Local Constraint-Based Mining and Set Constraint Programming for Pattern Discovery

Mehdi Khiari, Patrice Boizumault and Bruno Crémilleux

GREYC, Université de Caen, Campus Côte de Nacre,
F-14032 Caen Cedex, France
{Forename.Surname}@info.unicaen.fr

Abstract. Local patterns are at the core of the discovery of a lot of knowledge from data but their use is limited by their huge number. That it is why discovering global patterns and models from local patterns is an active research field. In this paper, we investigate the relationship between local constraint-based mining and constraint programming and we propose an approach to model and mine patterns combining several local patterns, i.e., patterns defined by global constraints. The user specifies a set of global constraints and a constraint solver generates the whole set of solutions. Our approach takes benefit from the recent progress on mining local patterns and all local constraints which can be inferred from the global ones are pushed by a solver on local patterns. This approach enables us to model in a flexible way *any* set of constraints combining several local patterns. Experiments show the feasibility of our approach.

1 Introduction

It is well-known that the "pattern flooding which follows data flooding" is an unfortunate consequence in exploratory Knowledge Discovery in Databases (KDD) processes. There is a large range of methods to discover the patterns of a potential user's interest but the most significant patterns are lost among too much trivial, noisy and redundant information. Many works propose methods to reduce the collection of patterns, such as the constraint-based paradigm [23], the pattern set discovery approach [10, 18], the so-called condensed representations [6] as well as the compression of the dataset by exploiting the Minimum Description Length Principle [25].

The constraint-based pattern mining framework is a powerful paradigm to discover new highly valuable knowledge [23]. Constraints provide a focus on the most promising knowledge by reducing the number of extracted patterns to those of potential interest for user. There are now generic approaches to discover *local patterns* under constraints [9, 26] and this issue is rather well-mastered, at least for data described by items (i.e., boolean attributes). We call *local constraints* the constraints addressing local patterns. Here, locality refers to the fact that checking whether a pattern satisfies or not a constraint can be performed independently of the other patterns holding in the data. Nevertheless, even if the number of produced local patterns is reduced thanks to the constraint, the output still remains too large for individual and global analysis by the end-user. Furthermore, local constraints are insufficient to define and discover a lot of global patterns and models expected by the end-users. Indeed, the interest of a pattern often depends on the other patterns and a lot of models such as a classifier or regression model require to consider simultaneously several patterns to combine the fragmented information conveyed by the local patterns. That is why we claim that discovering patterns under constraints involving comparisons between local patterns holding in the data is a major issue. In the following of this paper, we call *global constraints* such constraints. These constraints are at the basis of global patterns and models.

Mining patterns under local constraints requires the exploration of a large search space, even in the case of the simplest patterns, i.e., data described by items. Obviously, mining patterns under global constraints is even harder because we have to take into account and compare the solutions satisfying each pattern involved in a global constraint. In this paper, we investigate the relationship between constraint-based mining and constraint programming and we propose an approach to model and mine patterns under global constraints. As Constraint Satisfaction Problem (CSP) has the ability to define constraints on several variables [1], it is a natural way to model global constraints. We show that each pattern of a global constraint can be assimilated to a variable in the CSP framework. The great advantage of this modeling is its flexibility, it enables us to define a large broad of global constraints. Basically, with our approach, the user specifies the model, that is, the set of global constraints which has to be satisfied, and a constraint solver generates the correct and complete set of solutions. The CSP community has developed several efficient constraint solvers that we can reuse and the resolution can be performed at the level of this global modeling. But we think that it would be a pity not to take benefit from the recent progress on mining local patterns. That is why a key point of our approach is to divide a global constraint in two parts, i.e., a set of local constraints $\mathcal{C}_{loc}$ which is solved by a solver on local patterns and a set of global constraints $\mathcal{C}_{glob}$ which is solved by a CSP solver (cf. Section 4 for more details). We claim that is this combination between the local and global levels which enables us the discovery of patterns under global constraints. In other words, the contribution of this paper is to propose an approach joining local constraint mining and set constraint programming in order to model global constraints and discover patterns under such constraints. More generally, the paper investigates the relationship between constraint-based mining and set constraint programming.

This paper is organized as follows. Section 2 sketches definitions and presents the problem statement. The background on pattern discovery and set constraint programming is given in Section 3. We propose our approach to model and mine patterns under global constraints in Section 4. Section 5 details experiments. Section 6 deals with a discussion and research issues related to our approach.

2 Definitions and motivations

Below we give definitions used in the paper and the context and motivations.

2.1 Definitions

Let $\mathcal{I}$ be a set of distinct literals called *items*, an itemset (or pattern) is a non-null subset of $\mathcal{I}$. The language of itemsets corresponds to $\mathcal{L}_{\mathcal{I}} = 2^{\mathcal{I}} \backslash \emptyset$. A transactional dataset is a multi-set of itemsets of $\mathcal{L}_{\mathcal{I}}$. Each itemset, usually called transaction or object, is a database entry. For instance, Table 1 gives a transactional dataset $\boldsymbol{r}$ where 9 objects $o_1, \ldots, o_9$ are described by 6 items $A, \ldots, c_2$.

Trans.	Items					
o_1	A	B			c_1	
o_2	A	B			c_1	
o_3			C		c_1	
o_4			C		c_1	
o_5			C		c_1	
o_6	A	B	C	D		c_2
o_7			C	D		c_2
o_8			C			c_2
o_9				D		c_2

Table 1. Example of a transactional context $\boldsymbol{r}$.

Let X be a local pattern. Pattern mining aims at discovering information from all the patterns or a subset of $\mathcal{L}_{\mathcal{I}}$. More precise, constraint-based mining task selects all the itemsets of $\mathcal{L}_{\mathcal{I}}$ present in $\boldsymbol{r}$ and satisfying a predicate which is named *constraint*. Local patterns are regularities that hold for a particular part of the data. A local pattern is of special interest if it exhibits a deviating behavior w.r.t. the underlying global model of the data [15] because we are seeking for surprising knowledge which deviates from the already known background model. There are a lot of constraints to evaluate the relevance of local patterns. A well-known example is the frequency constraint which focuses on patterns having a frequency in the database exceeding a given minimal threshold $\gamma > 0$: $\mathit{freq}(X) \geq \gamma$. Many works [23] replace the frequency by other interestingness measures to evaluate the relevance of patterns such as the area of a pattern ($\mathit{area}(X)$ is the product of the frequency of the pattern times its length, i.e., $\mathit{area}(X) = \mathit{freq}(X) \times \mathit{count}(X)$ where $\mathit{count}(X)$ denotes the cardinality of X).

In practice, the user is often interested in discovering more complex patterns such as the simplest rules in the classification task based on associations [2, 30], pairs of exception rules [28] which may reveal global characteristics from the database. The definition of such patterns relies on properties involving several local patterns and are here called global patterns. They are formalized by the notion of *global constraint* [7].

Definition 1 (Global constraint). *A constraint q is said global if several local patterns have to be compared to check if q is satisfied or not.*

The next section provides more precise examples of global constraints.

2.2 Context and motivations

Global constraints are very useful to design a lot of patterns requested by the users. For instance, the discovery of exception rules from a data set without domain-specific information is of a great interest [28]. An exception rule is defined as a deviational pattern to a strong rule and the interest of an exception rule is evaluated according to another rule. The comparison between rules means that these exception rules are not local patterns. More formally, an exception rule is defined within the context of a pair of rules as follows (I is an item, for instance a class value, X and Y are local patterns):

$$exception(X \rightarrow \neg I) \equiv \begin{cases} true \quad \text{if } \exists Y \in \mathcal{L}_{\mathcal{I}} \text{ such that } Y \subset X, \text{ one have} \\ \qquad\qquad\qquad\qquad\qquad\qquad (X\backslash Y \rightarrow I) \wedge (X \rightarrow \neg I) \\ false \text{ otherwise} \end{cases}$$

Such a pair of rules is composed of a common sense rule $X\backslash Y \rightarrow I$ (the term "common sense rule" represents a user-given belief) and an exception rule $X \rightarrow \neg I$ since usually if $X\backslash Y$ then I. The exception rule isolates unexpected information. This definition assumes that the common sense rule has a high frequency and a rather high confidence and the exception rule has a low frequency and a very high confidence (the confidence of a rule $X \rightarrow Y$ is $freq(X \cup Y)/freq(X)$). Assuming that a rule $X \rightarrow Y$ holds iff at least 2/3 of the transactions containing X also contains Y, the rule $AC \rightarrow \neg c_1$ is an exception rule in our running example (cf. Table 1) because we jointly have $A \rightarrow c_1$ and $AC \rightarrow \neg c_1$. Note that Suzuki proposes a method based on sound pruning and probabilistic estimation [28] to extract the exception rules. Nevertheless, this method is devoted to this kind of patterns.

In the context of genomics, local patterns defined by groups of genes and satisfying the *area* constraint previously introduced above are at the core of the discovery of synexpression groups [16]. Nevertheless, in noisy data such as transcriptomic data, the search of fault-tolerant patterns is very useful to cope with the intrinsic uncertainty embedded in the data [4]. Global constraints are a way to design such fault-tolerant patterns: larger sets of genes with few exceptions are expressed by the union of several local patterns satisfying an area constraint and having a large overlapping between them. From two local patterns, it corresponds to the following global constraint: $area(X) > min_{area} \wedge area(Y) > min_{area} \wedge (area(X \cap Y) > \alpha \times min_{area})$ where min_{area} denotes the minimal area and α is a threshold given by the user to fix the minimal overlapping between the local patterns. The set of global constraints can also be extended by the use of the universal quantifier (see Section 7).

Section 4 presents our approach to model patterns satisfying such global constraints and how we combine local constraint mining and set constraint programming to extract these patterns.

3 Background: related works and set CSP

3.1 Local patterns and pattern sets discovery

As said in the introduction, there are a lot of works to discover local patterns under constraints. A key issue of these works is the use of the property of monotonicity because pruning conditions are straightforwardly deduced [21]. A constraint q is anti-monotone w.r.t. the item specialization iff for all $X \in \mathcal{L}_{\mathcal{I}}$ satisfying q, any subset of X also satisfies q. In this paper, we use the Music-dfs [1] prototype because it offers a set of syntactic and aggregate primitives to specify a broad spectrum of constraints in a flexible way [27]. Music-dfs mines soundly and completely all the patterns satisfying a given set of input local constraints. The efficiency of Music-dfs lies in its depth-first search strategy and a safe pruning of the pattern space exploiting the anti-monotonicity property to push the local constraints as early as possible. The pruning conditions are based on intervals representing several local patterns. The local patterns satisfying all the local constraints are provided in a condensed representation made of intervals (each interval represents a set of patterns satisfying the constraint and each pattern appears in only one interval). The lower bound of an interval is a prefix-free pattern and its upper bound is the prefix-closure of the lower bound [27].

There are also other approaches to combine local patterns. Recent approaches - pattern teams [18], constraint-based pattern set mining [10] and selecting patterns according to the added value of a new pattern given the currently selected patterns [5] - aim at reducing the redundancy by selecting patterns from the initial large set of local patterns on the basis of their usefulness in the context of the other selected patterns. Even if these approaches explicitly compare patterns between them, they are mainly based on the reduction of the redundancy or specific aims such as classification processes. We think that global constraints are a flexible way to take into account a bias given by the user to direct the final set of patterns toward a specific aim such as the search of exceptions. A general data mining framework based on the notion of local patterns to design global models is presented in [17]. This abstract framework helps analyzing and improving current methods in the area. In our approach (cf. Section 4), we show the interest of the set constraint programming in this general issue of combining local patterns.

Constraint programming is a powerful declarative paradigm for solving difficult combinatorial problems. In a constraint programming approach, one specifies constraints on acceptable solutions and search is used to find a solution that satisfies the constraints. A first approach using Constraint Programming for itemset mining has been proposed in [8]. In this work, constraints such as frequency, closedness, maximality, and constraints that are monotonic or anti-monotonic or variations of these constraints are modeled using 0/1 Linear Programming. Then patterns satisfying these constraints are obtained by using the constraint solver Gecode [12]. This work presents in a unified framework a large

[1] `http://www.info.univ-tours.fr/~soulet/music-dfs/music-dfs.html`

set of patterns but does not address patterns modeled by relationships between several local patterns as those described in Section 2. Recently, this work has been extended in order to find correlated patterns (i.e., patterns having the highest score w.r.t. a correlation measure) [24].

3.2 Set CSP

Formally a Constraint Satisfaction Problem (CSP) is a 3-uple $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{X}$ is a set of variables, $\mathcal{D}$ is a set of finite domains and $\mathcal{C}$ is a set of constraints that restrict certain simultaneous variables assignments. There are several types of CSPs such as numerical CSPs, boolean CSPs, set CSPs, etc. They differ fundamentally from the domain types and filtering techniques. We present here more precisely set CSPs that are used in our modeling. First, we define Set Intervals. Then we introduce set CSPs, and give an example. Finally we present some filtering rules for set CSPs.

Definition 2 (Set Interval). *let lb and ub be two sets such that* $lb \subset ub$*, the set interval* $[lb..ub]$ *is defined as follows:* $[lb..ub] = \{E$ *such that* $lb \subseteq E$ *and* $E \subseteq ub\}$.

Set intervals avoid data storage problems due to the size of domains: they model the domain and encapsulate all the possible values of the variables. For example: $[\{1\}..\{1,2,3\}]$ summarizes $\{\{1\},\{1,2\},\{1,3\},\{1,2,3\}\}$ and $[\{\}..\{1,2,3\}]$ summarizes $2^{\{1,2,3\}}$.

Definition 3 (Set CSP). *A set constraint satisfaction problem (set CSP) is a 3-uple* $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ *where* $\mathcal{C} = \{c_1, ..., c_m\}$ *is a set of constraints associated to a set* $\mathcal{X} = \{X_1, ..., X_n\}$ *of variables. For each variable* X_i*, an initial domain of set intervals (or union of set intervals)* D_{X_i} *is given and* $D = \{D_{X_i}, ..., D_{X_n}\}$.

In order to illustrate the declarative feature and the expressiveness of set CSPs, we give the following example.

Example [29] Two transmitters have to be assigned to two radio frequencies each. Available frequencies are $\{1,2,3,4\}$ for the first transmitter and $\{3,4,5,6\}$ for the second one. The distance between these two frequencies is equal to the absolute value of the difference between these frequencies. The constraints are:

- two radio frequencies have to be assigned to each transmitter: $c_1 \wedge c_2$.
- both transmitters do not share frequencies: c_3
- two frequencies within a transmitter must have at least a distance equals to 2: c_4
- the first transmitter requires the frequency 3: c_5
- the second transmitter requires the frequency 4: c_6

It can be expressed as a set CSP $(\mathcal{X}, \mathcal{D}, \mathcal{C})$, where:

- $\mathcal{X} = \{t_1, t_2\}$ where t_1 and t_2 are the two transmitters.

– $D(t_1) = [\{\} \mathrel{..} \{1,2,3,4\}]$ and $D(t_2) = [\{\} \mathrel{..} \{3,4,5,6\}]$.
– $\mathcal{C} = \{c_1, c_2, c_3, c_4, c_5, c_6\}$ where:
  - $c_1 \quad | t_1 | = 2$
  - $c_2 \quad | t_2 | = 2$
  - $c_3 \quad t_1 \cap t_2 = \emptyset$
  - $c_4 \quad \forall v_1, v_2 \in t_i, \quad | v_1 - v_2 | \geq 2 \quad i = 1, 2$
  - $c_5 \quad 3 \in t_1$
  - $c_6 \quad 4 \in t_2$

This problem has a unique solution where the first transmitter is assigned to the frequencies $\{1,3\}$ and the second to $\{4,6\}$.

Examples of filtering rules for set CSPs: For CSPs, filtering consists of reducing the variable domains in order to remove values that cannot occur in any solution. As soon as a domain D_{X_i} becomes empty (i.e., there is no available value for X_i), a failure is generated for the search. Filtering rules for integer intervals and set intervals are presented in [22, 19, 14]. We now present two examples of filtering rules for set intervals, the *inclusion* and the *intersection* constraints:

Let $D_x = [a_x..b_x]$, $D_y = [a_y \mathrel{..} b_y]$ and $D_z = [a_z \mathrel{..} b_z]$ three domains represented by set intervals and D'_x, D'_y and D'_z the filtered domains.

– **Constraint:** $X \subset Y$
Filtering rule:
```
if $a_x \subset b_y$ then
    $D'_x = [a_x \mathrel{..} b_x \cap b_y]$
    $D'_y = [a_x \cup a_y \mathrel{..} b_y]$
else
    $D'_x = \emptyset, D'_y = \emptyset$
```

– **Constraint:** $Z = X \cap Y$
Filtering rule:
```
if $(b_x \cap b_y) \subset b_z$ and $(b_x \cap b_y) \neq \emptyset$ then
    $D'_x = [a_x \cup a_z \mathrel{..} b_x \setminus ((b_x \cap a_y) \setminus b_z]$
    $D'_y = [a_y \cup a_z \mathrel{..} b_y \setminus ((b_y \cap a_x) \setminus b_z]$
    $D'_z = [a_z \cup (a_x \cap a_y) \mathrel{..} b_z \cap b_x \cap b_y]$
else
    $D'_x = D'_y = D'_z = \emptyset$
```

Programming tool: ***ECL^iPS^e*** [11] is a Constraint Programming Tool supporting the most common techniques used in solving constraints satisfaction (or optimization) problems: Constraint Satisfaction Problems, Mathematical Programming, Local Search and combinations of those. ECL^iPS^e is built around the Constraint Logic Programming paradigm [1]. Different domains of constraints as numeric CSP and Set CSPs can be used together. Finally, libraries for solving set CSPs, as *ic-sets* or *conjunto* [13], are available in ECL^iPS^e.

4 Set Constraint Programming for Pattern Discovery

Our approach is based on two major points. First, we use the wide possibilities of modelization and resolution given by the CSPs, in particular the set CSPs and numeric CSPs. Second, we take benefit from the recent progress on mining local patterns. The last choice is also strengthened by the fact that local constraints can be solved before and regardless global constraints.

In this section, we start by giving an overview of our approach. Then we describe each of the three steps of our method by considering the example of the exception rules described in Section 2.2.

4.1 General overview

Figure 1 provides a general overview of the three steps of our approach:

1. Modeling the query as CSPs, then splitting constraints into local ones and global ones.
2. Solving local constraints using a local pattern extractor (Music-dfs, introduced in Section 3.1) which produces an interval condensed representation of all patterns satisfying the local constraints.
3. Solving global constraints of the CSPs by using ECL^iPS^e (introduced in Section 3.2) where the domain of each variable results from the interval condensed representation (computed in the Step-2).

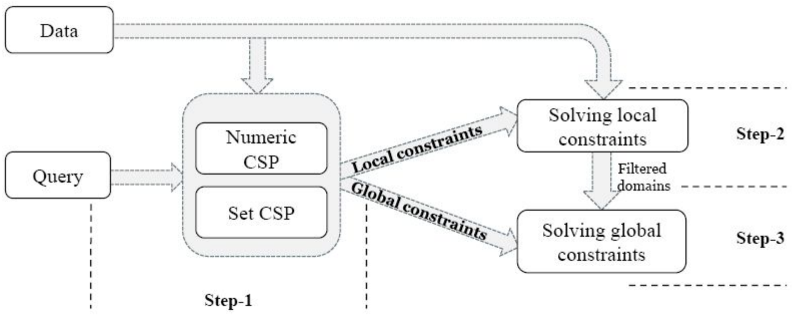

Fig. 1. General overview of our 3-steps method

4.2 Step-1: Modelling the query as CSPs

Let $\boldsymbol{r}$ be a dataset having nb transactions, and $\mathcal{I}$ the set of all its items. We model the problem by using two CSPs $\mathcal{P}$ and $\mathcal{P}'$ that are inter-related:

1. Set interval CSP $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ where:
   - $\mathcal{X} = \{X_1, ..., X_n\}$. Each variable X_i represents an unknown itemset.
   - $\mathcal{D} = \{D_{X_1}, ..., D_{X_n}\}$. The initial domain of each variable X_i is the set interval $[\{\} \,..\, \mathcal{I}]$.
   - $\mathcal{C}$ is a conjunction of set constraints by using set operators ($\cup$, $\cap$, $\setminus$, $\in$, $\notin$, ...)
2. Numeric CSP $\mathcal{P}' = (\mathcal{F}, \mathcal{D}', \mathcal{C}')$ where:
   - $\mathcal{F} = \{F_1, ..., F_n\}$. Each variable F_i is the frequency of the itemset X_i.
   - $\mathcal{D}' = \{D_{F_1}, ..., D_{F_n}\}$. The initial domain of each variable F_i is the integer interval $[1 \,..\, nb]$.
   - $\mathcal{C}'$ is a conjunction of arithmetic constraints.

Then, the whole set of constraints ($\mathcal{C} \cup \mathcal{C}'$) is divided into two subsets as follows:

- $\mathcal{C}_{loc}$ is the set of local constraints to be solved (by Music-dfs). Solutions are given in the form of an interval condensed representation.
- $\mathcal{C}_{glob}$ is the set of global constraints to be solved (by ECL^iPS^e), where the domain of the variables X_i and F_i will be deduced from the interval condensed representation computed in the previous step.

Local (unary) constraints can be solved before and regardless global constraints. The search space of the global constraints is reduced by the space of solutions satisfying local constraints. This ensures that every solution verifies both local and global constraints.

4.3 Example: modeling the exception rules as CSPs

Recall that the definition of the pairs of exception rules is given in Section 2.2.

Reformulation: Let $freq(X)$ be the frequency value of the itemset X. Let I and $\neg I \in \mathcal{I}$ (in this example, I and $\neg I$ represent the two class values of the data set). Let $\gamma_1, \gamma_2, \delta_1, \delta_2 \in \mathbb{N}$. The exception rules constraint can be formulated as it follows:

- $X \backslash Y \rightarrow I$ can be expressed by the conjunction: $freq((X \setminus Y) \sqcup$[2] $I) \geq \gamma_1 \wedge (freq(X \setminus Y) - freq((X \setminus Y) \sqcup I)) \leq \delta_1$ which means that $X \backslash Y \rightarrow I$ must be a frequent rule having a high confidence value.
- $X \rightarrow \neg I$ can be expressed by the conjunction: $freq(X \sqcup \neg I) \leq \gamma_2 \wedge (freq(X) - freq(X \sqcup \neg I)) \leq \delta_2$ which means that $X \rightarrow \neg I$ must be a rare rule having a high confidence value.

To sum up:

$$exception(X \rightarrow \neg I) \equiv \begin{cases} \exists Y \subset X \text{ such that:} \\ freq((X \setminus Y) \sqcup I) \geq \gamma_1 \wedge \\ (freq(X \setminus Y) - freq((X \setminus Y) \sqcup I)) \leq \delta_1 \wedge \\ freq(X \sqcup \neg I) \leq \gamma_2 \wedge \\ (freq(X) - freq(X \sqcup \neg I)) \leq \delta_2 \end{cases}$$

[2] the symbol $\sqcup$ denotes the disjoint union operator

CSP modelisation: The CSP variables are defined as follows:

- Set variables $\{X_1, X_2, X_3, X_4\}$ representing unknown itemsets:
  - $X_1 : X \setminus Y$,
  - $X_2 : (X \setminus Y) \sqcup I$ (common sense rule),
  - $X_3 : X$,
  - $X_4 : X \sqcup \neg I$ (exception rule).
- Integer variables $\{F_1, F_2, F_3, F_4\}$ representing their frequency values (variable F_i denotes the frequency of the itemset X_i).

Table 2 provides the constraints modeling the exception rules.

Constraints	**CSP formulation**	**Local**	**Global**
$freq((X \setminus Y) \sqcup I) \geq \gamma_1$	$F_2 \geq \gamma_1$	$\times$	
	$\wedge$		
	$I \in X_2$	$\times$	
	$\wedge$		
	$X_1 \subsetneq X_3$		$\times$
$freq(X \setminus Y) - freq((X \setminus Y) \sqcup I) \leq \delta_1$	$F_1 - F_2 \leq \delta_1$		$\times$
	$\wedge$		
	$X_2 = X_1 \sqcup I$		$\times$
$freq(X \sqcup \neg I) \leq \gamma_2$	$F_4 \leq \gamma_2$	$\times$	
	$\wedge$		
	$\neg I \in X_4$	$\times$	
$freq(X) - freq(X \sqcup \neg I) \leq \delta_2$	$F_3 - F_4 \leq \delta_2$		$\times$
	$\wedge$		
	$X_4 = X_3 \sqcup \neg I$		$\times$

Table 2. Exception rules modeled as CSP constraints

Summary:

- Set interval CSP
  - $\mathcal{X} = \{X_1, ..., X_4\}$
  - $\mathcal{C} = \{(I \in X_2), (X_2 = X_1 \sqcup I), (\neg I \in X_4), (X_4 = X_3 \sqcup \neg I), (X_1 \subsetneq X_3)\}$
- Numeric CSP
  - $\mathcal{F} = \{F_1, ..., F_4\}$
  - $\mathcal{C}' = \{(F_2 \geq \gamma_1), (F_1 - F_2 \leq \delta_1), (F_4 \leq \gamma_2), (F_3 - F_4 \leq \delta_2)\}$

- $\mathcal{C}_{loc} = \{(I \in X_2), (F_2 \geq \gamma_1), (F_4 \leq \gamma_2), (\neg I \in X_4)\}$
- $\mathcal{C}_{glob} = \{(F_1 - F_2 \leq \delta_1), (X_2 = X_1 \sqcup I), (F_3 - F_4 \leq \delta_2), (X_4 = X_3 \sqcup \neg I), (X_1 \subsetneq X_3)\}$

4.4 Step-2: Solving local constraints

As already said, we use for this task Music-dfs (see Section 3.1) which mines soundly and completely local patterns. In order to fully benefit from the efficiency of the local pattern mining, the set of local constraints $\mathcal{C}_{loc}$ is split into a disjoint union of $\mathcal{C}_i$ (for $i \in [1..n]$) where each $\mathcal{C}_i$ is the set of local constraints related to X_i and F_i. Each C_i can be separately solved. Let CR_i be the interval condensed representation of all the solutions of $\mathcal{C}_i$. $CR_i = \bigcup_p (f_p, I_p)$ where I_p is a set interval verifying: $\forall x \in I_p,\ \mathit{freq}(x) = f_p$. Then the filtered domains (see Section 4.3) for variable X_i and variable F_i are:

- D_{F_i} : the set of all f_p in CR_i
- D_{X_i} : $\bigcup_{I_p \in CR_i} I_p$

Example: Let us consider the dataset $\boldsymbol{r}$ (see Table 1) and the local constraints for the exception rules $\mathcal{C}_{loc} = \{(I \in X_2), (F_2 \geq \gamma_1), (F_4 \leq \gamma_2), (\neg I \in X_4)\}$ (see Section 4.3). The respective values for $(I, \neg I, \gamma_1, \delta_1, \gamma_2, \delta_2)$ are $(c_1, c_2, 2, 1, 1, 0)$. The local constraints set related to X_2 is $\mathcal{C}_2 = \{c_1 \in X_2, F_2 \geq 2\}$ is solved by Music-dfs with the following query showing that the parameters given to Music-dfs are straightforwardly deduced from $\mathcal{C}_{loc}$.

```
---------------
./music-dfs -i donn.bin -q "{c1} subset X2 and freq(X2)>=2;"
X2 in [A, c1]..[A, c1, B ] U [B, c1] -- F2 = 2 ;
X2 in [C, c1] -- F2 = 3
---------------
```

4.5 Step-3: Solving global constraints

Then, from the condensed representation of all patterns satisfying local constraints, domains of the variables X_i and F_i (for $i \in \{1, 2, 3, 4\}$) are updated.

Given the parameters $I = c_1, \neg I = c_2, \delta_1 = 1$ and $\delta_2 = 0$ ($\gamma_1 = 2$ and $\gamma_2 = 1$ are already used in Step-2) and the data set in Table 1, the following ECL^iPS^e session illustrates how all pairs of exception rules can be obtained by using backtracking:

```
---------------
[eclipse 1]:
?- exceptions(X1, X2, X3, X4).
Sol1 : X1 = [A,B], X2=[A,B,c1], X3=[A,B,C], X4=[A,B,C,c2];
Sol2 : X1 = [A,B], X2=[A,B,c1], X3=[A,B,D], X4=[A,B,D,c2];
.../...
---------------
```

5 Experiments

This section shows the practical usage and the feasibility of our approach. This experimental study is conducted on the *postoperative-patient-data* coming from the UCI machine learning repository[3]. This data set gathers 90 objects described by 23 items and characterized by two classes (two objects of a third class value were put aside). We test our approach by using the exception rules as a global constraint (in the following, we use a class value for the item I given in the definition of an exception rule). As previously said, we use Music-dfs (see Section 3.1) and ECL^iPS^e (see Section 3.2). All the tests were performed on a 2 GHz Intel Centrino Duo processor with Linux operating system and 2GB of RAM memory.

These experiments show the feasibility of our approach. Given $(I, \gamma_1, \delta_1, \gamma_2, \delta_2)$ a set of values, our method is able to mine the correct and complete set of all pairs of exception rules.

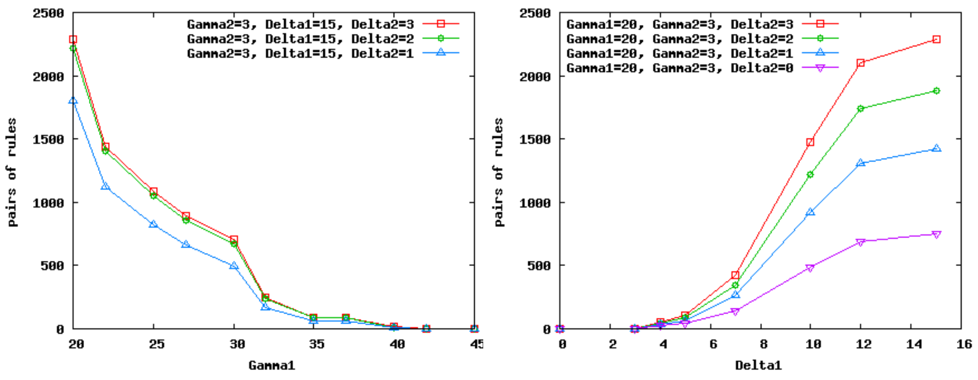

Fig. 2. Number of rules according to γ_1 (left) and δ_1 (right)

Figure 2 depicts the number of pairs of rules according to γ_1 (left part of the figure) and δ_1 (right part of the figure). We tested several combinations of the parameters. As expected, the lower γ_1 is, the larger the number of pairs of exception rules. Note that the decreasing of the curves is approximatively the same for all the combinations of parameters. The result is similar when δ_1 varies (right part of Figure 2): the higher δ_1 is, the larger the number of pairs of exception rules (when δ_1 increases, the confidence decreases so that there are more common sense rules). Interestingly, these curves quantify the number of pairs of exception rules according to the sets of parameters. Some cases seem to point out pairs of rules of good quality. For instance, with $(\gamma_1 = 20, \delta_1 = 5, \gamma_2 = 1, \delta_2 = 0)$, we obtain 25 pairs of rules with a common sense rule having a confidence value greater than or equal to 83% and an exact exception rule (i.e., confidence value equals 100%). Moreover, our approach enables us in a natural way to add new properties such as the control of the sizes of rules. If the user

[3] `www.ics.uci.edu/~mlearn/MLRepository.html`

wants that the number of items added to an exception rule remains small with regards to the size of the common sense rule, it can be easily modeled by a new constraint: for instance, the number of added items to an exception rule must be lower than the minimum of a number (e.g., 3) and the size of the common sense rule. It highlights the flexibility of our approach.

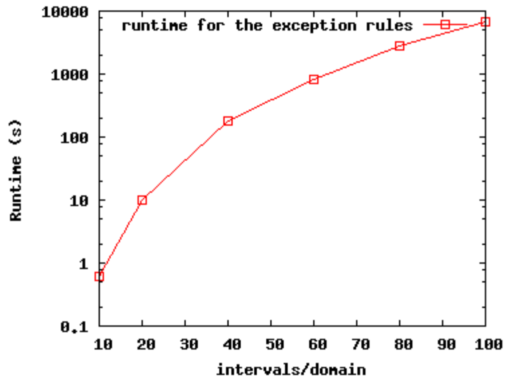

Fig. 3. Runtime according to the number of intervals of the condensed representations

Figure 3 details the runtime of our method according to the number of intervals of the condensed representation, i.e., the size of the condensed representation. In this experiment, for each dot of the curve, the four variables have the same domain and thus the same number of intervals. Obviously, the larger the number of intervals is, the higher the runtime (note that we use a logarithmic scale on the Y axis). In the case of exception rules, it is interesting to note that the runtime decreases when the quality of the exception rule pairs increases. Indeed, looking for common sense rules with high frequency and reliable exception rules leads to infer local constraints giving more powerful pruning conditions and thus less intervals. Table 3 indicates the number of intervals of the variable X_2 in the condensed representation (see Section 4.3) according to several local constraints. It shows the interest of an approach based on local constraint mining.

Local constraint	Number of intervals in D_{X_2}
-	3002
$I \in X_2$	1029
$I \in X_2 \wedge freq(X_2) >= 20$	52
$I \in X_2 \wedge freq(X_2) >= 25$	32

Table 3. Number of intervals according to several local constraints (case of D_{X_2})

6 Discussion

In this section, we discuss current research that is related to our approach. The key point is the set union operator for set CSPs and the techniques which could be developed to improve this step.

Set Union vs convexe closure for set Intervals. In order to perform bound consistency filtering, set CSP solvers approximate the union of two set intervals by their convex closure. The convex closure of $[lb_1 .. ub_1]$ and $[lb_2 .. ub_2]$ is defined as $[lb_1 \cap lb_2 .. ub_1 \cup ub_2]$. So, if filtering is applied a lot of times on a same variable domain, this domain could reach the whole set $[\emptyset .. \mathcal{I}]$ and specific information gathered during the search would be lost whereas this information is useful to limit the size of intervals.

To circumvent this problem, for each variable X_i with the condensed representation $CR_i = \bigcup_p (f_p, I_p)$, a search is successively performed upon each I_p. This approach is sound and complete and we use it in our experiments. Nevertheless, with this method, we do not fully profit from filtering because removing a value is propagated only in the treated intervals and not in the whole domains. It explains the results of Section 5 showing that the runtime strongly increases when the number of intervals increases.

Other solutions: a first alternative solution consists of implementing a set interval union operator in the kernel of the solver. Another solution is to use non-exact condensed representations to reduce the number of produced intervals such as a condensed representation based on maximal frequent itemsets. The number of intervals representing the domains will be smaller, but, due to the approximations, it should be necessary to memorize forbidden values.

7 Conclusions and future work

In this paper we have presented a new approach for pattern discovery. Its great interest is to model in a flexible way any set of constraints combining several local patterns. The complete and sound set of patterns satisfying the constraints is mined thanks to a joint cooperation between a solver on set constraint programming which copes with global constraints and a solver on local patterns to take benefit on the well-mastered methods on local constraint mining. We think that it is this combination between the local and global levels which enables us the discovery of such patterns. Experiments show the feasibility of our approach.

Further work is to introduce the universal quantification ($\forall$) that classic CSPs are unable to manage. This quantifier would be precious to model global constraints such as the peak constraint (the peak constraint compares neighbor patterns and a peak pattern is a pattern whose all neighbors have a value for a measure lower than a threshold). For that purpose, we think that recent works as Quantified Constraints Satisfaction Problems (QCSP) [3, 20] could be useful.

Acknowledgments. The authors would like to thank Arnaud Soulet for very fruitful discussions and Music-dfs prototype. This work is partly supported by the ANR (French Research National Agency) funded project Bingo2 ANR-07-MDCO-014.

References

1. K. R. Apt and M. Wallace. *Constraint Logic Programming using Eclipse*. Cambridge University Press, New York, NY, USA, 2007.
2. E. Baralis and S. Chiusano. Essential cmassification rule sets. *ACM Transactions on Database Systems*, 29(4):635–674, 2004.
3. F. Benhamou and F. Goualard. Universally quantified interval constraints. In *proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP'00)*, pages 67–82, London, UK, 2000. Springer-Verlag.
4. J. Besson, C. Robardet, and J.-F. Boulicaut. Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In *14th International Conference on Conceptual Structures (ICCS'06)*, pages 144–157, Aalborg, Denmark, 2006. Springer-Verlag.
5. B. Bringmann and A. Zimmermann. The chosen few: On identifying valuable patterns. In *proceedings of the 12th IEEE International Conference on Data Mining (ICDM-07)*, pages 63–72, Omaha, NE, 2007.
6. T. Calders, C. Rigotti, and J.-F. Boulicaut. A survey on condensed representations for frequent sets. In J.-F. Boulicaut, L. De Raedt, and H. Mannila, editors, *Constraint-Based Mining and Inductive Databases*, volume 3848 of *LNAI*, pages 64–80. Springer, 2005.
7. B. Crémilleux and A. Soulet. Discovering knowledge from local patterns with global constraints. In *8th International conference on Computational Science and Applications (ICCSA'08)*, volume 5073 of *Lecture Notes in Computer Science*, pages 1242–1257, Perugia, Italy, June 2008. Springer.
8. L. De Raedt, T. Guns, and S. Nijssen. Constraint Programming for Itemset Mining. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining edition:14 location*, Las Vegas, Nevada, USA, 2008.
9. L. De Raedt, M. Jäger, S. D. Lee, and H. Mannila. A theory of inductive query answering. In *proceedings of the IEEE Conference on Data Mining (ICDM'02)*, pages 123–130, Maebashi, Japan, 2002.
10. L. De Raedt and A. Zimmermann. Constraint-based pattern set mining. In *Proceedings of the Seventh SIAM International Conference on Data Mining*, Minneapolis, Minnesota, USA, April 2007. SIAM.
11. ECLiPSe. Eclipse documentation. *http://www.eclipse-clp.org*.
12. Gecode Team. Gecode: Generic constraint development environment, 2006. Available from `http://www.gecode.org`.
13. C. Gervet. Conjunto: constraint logic programming with finite set domains. In *ILPS '94: Proceedings of the 1994 International Symposium on Logic programming*, pages 339–358, Cambridge, MA, USA, 1994. MIT Press.
14. C. Gervet. Interval Propagation to Reason about Sets: Definition and Implementation of a Practical Language. *Constraints*, 1(3):191–244, 1997.
15. D. J. Hand. *ESF exploratory workshop on Pattern Detection and Discovery in Data Mining*, volume 2447 of *LNCS*, chapter Pattern detection and discovery, pages 1–12. Springer, 2002.

16. J. Kléma, S. Blachon, A. Soulet, B. Crémilleux, and O. Gandrillon. Constraint-based knowledge discovery from sage data. *In Silico Biology*, 8(0014), 2008.
17. A. Knobbe, B. Crémilleux, J. Fürnkranz, and M. Scholz. From local patterns to global models: The lego approach to data mining. In *International Workshop "From Local Patterns to Global Models" co-located with ECML/PKDD'08*, pages 1–16, Antwerp, Belgium, September 2008.
18. A. Knobbe and E. Ho. Pattern teams. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'06)*, volume 4213 of *LNAI*, pages 577–584, Berlin, Germany, September 2006. Springer-Verlag.
19. O. Lhomme. Consistency techniques for numeric csps. In *Proc. of the 13th IJCAI*, pages 232–238, Chambery, France, 1993.
20. N. Mamoulis and K. Stergiou. Algorithms for quantified constraint satisfaction problems. In *proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP'04)*, pages 752–756, 2004.
21. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
22. R. E. Moore. *Interval analysis.* Prentice Hall, 1966.
23. R. T. Ng, V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *proceedings of ACM SIGMOD'98*, pages 13–24. ACM Press, 1998.
24. S. Nijssen, T. Guns, and L. De Raedt. Correlated itemset mining in roc space: a constraint programming approach. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 647–655, Paris, France, June 2009.
25. A. Siebes, J. Vreeken, and M. van Leeuwen. Item sets that compress. In *Proceedings of the Sixth SIAM International Conference on Data Mining*, Bethesda, MD, USA, April 2006. SIAM.
26. A. Soulet and B. Crémilleux. An efficient framework for mining flexible constraints. In H. T. Bao, D. Cheung, and H. Liu, editors, *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'05)*, volume 3518 of *LNAI*, pages 661–671, Hanoi, Vietnam, May 2005. Springer.
27. A. Soulet, J. Klema, and B. Crémilleux. *Post-proceedings of the 5th International Workshop on Knowledge Discovery in Inductive Databases in conjunction with ECML/PKDD 2006 (KDID'06)*, volume 4747 of *Lecture Notes in Computer Science*, chapter Efficient Mining under Rich Constraints Derived from Various Datasets, pages 223–239. Springer, 2007.
28. E. Suzuki. Undirected Discovery of Interesting Exception Rules. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(8):1065–1086, 2002.
29. V. Thornary, J. Gensel, and P. Sherpa. An hybrid representation for set constraint satisfaction problems. In *Workshop on Set Constraints co-located with the fourth Int. Conf. on Principles and Practice of Constraint Programming*, Pisa, Italy, 1998.
30. X. Yin and J. Han. CPAR: classification based on predictive association rules. In *proceedings of the 2003 SIAM Int. Conf. on Data Mining (SDM'03)*, San Fransisco, CA, May 2003.