Constraint Programming for Mining n-ary Patterns

Medhi Khiari, Patrice Boizumault, Bruno Crémilleux

▶ **To cite this version:**

Medhi Khiari, Patrice Boizumault, Bruno Crémilleux. Constraint Programming for Mining n-ary Patterns. Int. Conference on Principles and Practice of Constraint Programming (CP'10), Sep 2010, St Andrews, Scotland, France. pp.552-567. hal-01016652

HAL Id: hal-01016652
https://hal.science/hal-01016652

Submitted on 30 Jun 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Constraint Programming for Mining n-ary Patterns

Mehdi Khiari, Patrice Boizumault, and Bruno Crémilleux

GREYC, CNRS - UMR 6072, Université de Caen Basse-Normandie,
Campus Côte de Nacre, F-14032 Caen Cedex, France
{Forename.Surname}@info.unicaen.fr

Abstract. The aim of this paper is to model and mine patterns combining several local patterns (n-ary patterns). First, the user expresses his/her query under constraints involving n-ary patterns. Second, a constraint solver generates the correct and complete set of solutions. This approach enables to model in a flexible way sets of constraints combining several local patterns and it leads to discover patterns of higher level. Experiments show the feasibility and the interest of our approach.

1 Introduction

Knowledge Discovery in Databases involves different challenges, such as the discovery of patterns of a potential user's interest. The constraint paradigm brings useful techniques to express such an interest. If mining local patterns under constraints is now a rather well-mastered domain including generic approaches [3], these methods do not take into account the interest of a pattern with respect to the other patterns which are mined: the useful patterns are lost among too much trivial, noisy and redundant information. In practice, a lot of patterns which are expected by the data analyst (cf. Section 2.2) require to consider simultaneously several patterns to combine the fragmented information conveyed by the local patterns. It also explains why the question of how to turn collections of local patterns into global models such as classifiers or clustering receives a large attention [13]. That is why the discovery of patterns under constraints involving combinations of local patterns is a major issue. In the following, such constraints are called *n-ary constraints* and they define n-ary patterns.

There are very few attempts on mining patterns involving several local patterns and the existing methods tackle particular cases by using devoted techniques [19]. One explanation of the lack of generic methods may be the difficulty of the task: mining local patterns under constraints requires the exploration of a large search space but mining patterns under n-ary constraints is even harder because we have to take into account and compare the solutions satisfying each pattern involved in the constraint. We think that the lack of generic approaches restrains the discovery of useful patterns because the user has to develop a new method each time he wants to extract a new kind of patterns. It explains why this issue deserves our attention.

In this paper, we propose a generic approach for modeling and mining n-ary patterns using Constraint Programming (CP). Our approach proceeds in two steps. First, the user specifies the set of constraints which has to be satisfied. Such constraints handle set operations (e.g., inclusion, membership) but also numeric properties such as the frequency or the length of patterns. A constraint solver generates the correct and complete set of solutions. The great advantage of this modeling is its flexibility, it enables us to define a large broad of n-ary queries leading to discover patterns of higher level. It is no longer necessary to develop algorithms from scratch to mine new types of patterns. To the best of our knowledge, it is the first generic approach to express and mine patterns involving several local patterns.

Cross-fertilization between data mining and CP is a research field in emergence and there are very few attempts in this area. A seminal work [8] proposes a formulation in CP of constraints on local patterns but it does not address rich patterns such as n-ary patterns. We did a preliminary work [12] based on a joint use of local patterns constraint-based mining and CP in order to discover n-ary patterns and to investigate such relationships.

This paper is organized as follows. Section 2 sketches definitions and presents the context. Our approach is described in Section 3, we illustrate it from several kinds of n-ary queries. Section 4 describes the modeling of such n-ary queries as Constraint Satisfaction Problems (CSP). Section 5 presents a background on pattern discovery and our hybrid method for mining n-ary patterns. In Section 6, we go further in CP area and we propose a *full-CP* method to mine n-ary queries. Section 7 compares the two methods. Finally we conclude (Section 8) giving several research issues on using CP for pattern discovery.

2 Definitions and First Examples

2.1 Local Patterns

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 a *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 $A, \ldots, c_2$.

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. 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.

Example. Let X be a local pattern. The *frequency* constraint focuses on patterns occurring in the database a number of times exceeding a given minimal threshold: $freq(X) \geq minfr$. An other interesting measure to evaluate the relevance of patterns is the area [15]. The *area* of a pattern is the product of its frequency times its length: $area(X) = freq(X) \times length(X)$ where $length(X)$ denotes the cardinality of X.

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

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

2.2 N-ary Patterns

In practice, the data analyst is often interested in discovering richer patterns than local patterns and he/she is looking for patterns that reveal more complex characteristics from the database. The definitions relevant to such patterns rely on properties involving several local patterns and are formalized by the notions of *n-ary constraint* and *n-ary pattern* leading to *n-ary queries*:

Definition 1 (n-ary constraint). *A constraint q is said n-ary if it involves several local patterns.*

Definition 2 (n-ary pattern). *A pattern X is said n-ary if it is defined by a n-ary constraint.*

Definition 3 (n-ary query). *A n-ary query is a conjunction of n-ary constraints.*

2.3 Motivating Examples

N-ary queries enable us to design a lot of patterns requested by the users such as the discovery of pairs of exception rules without domain-specific information [19] or the simplest rules in the classification task based on associations [21].

Example 1. An exception rule is defined as a pattern combining a strong rule and a deviational pattern to the strong rule, the interest of a rule of the pattern is highlighted by the comparison with the other rule The comparison between rules means that these exception rules are *not* local patterns. This enables us to distinguish exception rules from rare rules where a rare rule is a rule with a very low frequency value. This is useful because in practice rare rules cannot be straightforwardly used because many of them arise by chance and are unreliable. 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):

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

Such a pair of rules is composed of a common sense rule $X \backslash Y \rightarrow I$ 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 (see 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 [19] to extract the exception rules, but this method is devoted to this kind of patterns.

Example 2. In the context of genomics, data are often noisy and the search of fault-tolerant patterns is very useful to cope with the intrinsic uncertainty embedded in the data [2]. Defining n-ary queries is a way to design such fault-tolerant patterns candidate to be synexpression groups: larger sets of genes with few exceptions are expressed by the union of several local patterns satisfying the *area* constraint previously introduced and having a large overlapping between them. It corresponds to the following n-ary query:

$$(area(X) > min_{area}) \wedge (area(Y) > min_{area}) \wedge (area(X \cap Y) > \alpha \times min_{area})$$

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 X and Y.

3 Examples of n-ary Queries

In this section, we present the modeling of several n-ary queries within our approach. Some of them were introduced in Section 2.2.

3.1 Exception Rules

Let $freq(X)$ be the frequency value of the pattern X. Let Y be a pattern, I and $\neg I \in \mathcal{I}$ (I and $\neg I$ can represent two class values of the dataset). Let $minfr$, $maxfr, \delta_1, \delta_2 \in \mathbb{N}$. The exception rule n-ary query is formulated as it follows:

- $X \backslash Y \rightarrow I$ is expressed by the conjunction: $freq((X \setminus Y) \sqcup$[1] $I) \geq minfr \wedge (freq(X \setminus Y) - freq((X \setminus Y) \sqcup I)) \leq \delta_1$ which means that $X \backslash Y \rightarrow I$ must be a frequent rule having a high confidence value.
- $X \rightarrow \neg I$ is expressed by the conjunction: $freq(X \sqcup \neg I) \leq maxfr \wedge (freq(X) - freq(X \sqcup \neg I)) \leq \delta_2$ which means that $X \rightarrow \neg I$ must be a rare rule having a high confidence value.

[1] The symbol $\sqcup$ denotes the disjoint union operator. It states that for a rule, patterns representing respectively premises and conclusion must be disjoint.

To sum up:

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

3.2 Unexpected Rules

Padmanabhan and Tuzhilin [17] propose the notion of *unexpected* rule $X \rightarrow Y$ with respect to a belief $U \rightarrow V$ where U and V are patterns. Basically, an unexpected rule means that Y and V logically contradict each other. It is defined in [17] by (1) $Y \wedge V \vDash False$, (2) $X \wedge U$ holds (it means XU frequent), (3) $XU \rightarrow Y$ holds ($XU \rightarrow Y$ frequent and has a sufficient confidence value), (4) $XU \rightarrow V$ does not hold ($XU \rightarrow V$ not frequent or $XU \rightarrow V$ has a low confidence value). Given a a belief $U \rightarrow V$, an unexpected rule $un.(X,Y)$ is modeled by the following n-ary query:

$$un.(X,Y) \equiv \begin{cases} freq(Y \cup V) = 0 \wedge \\ freq(X \cup U) \geq minfr_1 \wedge \\ freq(X \cup U \cup Y) \geq minfr_2 \ \wedge (freq(X \cup U \cup Y)/freq(X \cup U)) \geq minconf \wedge \\ (freq(X \cup U \cup V) < maxfr \ \vee (freq(X \cup U \cup V)/freq(X \cup U)) < maxconf) \end{cases}$$

3.3 Synexpression Groups

From n local patterns, the search of synexpression groups is expressed by the following n-ary query:

$$synexpr(X_1, ..., X_n) \equiv \begin{cases} \forall\ 1 \leq i < j \leq n, \\ area(X_i) > min_{area} \wedge \\ area(X_j) > min_{area} \wedge \\ area(X_i \cap X_j) > \alpha \times min_{area} \end{cases}$$

where min_{area} denotes the minimal area (defined in Section 2.1) and α is a threshold given by the user to fix the minimal overlapping between the local patterns. This example illustrates how a n-ary query enables us to easily express complex and fault-tolerant patterns such as candidate synexpression groups.

3.4 Classification Conflicts

In classification based on associations [21], the quality of the classifier is based on the combination of its local patterns. By taking into account several local patterns, n-ary queries are very useful to help to design classifiers. Let c_1 and c_2 be the items denoting the class values. A classification conflict can be defined by a pair of frequent rules $X \rightarrow c_1$ and $Y \rightarrow c_2$ with a confidence greater than a minimal threshold $minconf$ and a large overlapping between their premises.

$$classif.\ conflict(X,Y) \equiv \begin{cases} freq(X) \geq minfr \wedge \\ freq(Y) \geq minfr \wedge \\ (freq(X \sqcup \{c_1\})/freq(X)) \geq minconf \wedge \\ (freq(Y \sqcup \{c_2\})/freq(Y)) \geq minconf \wedge \\ 2 \times length(X \cap Y) \geq (length(X) + length(Y))/2 \end{cases}$$

Table 2. Primitive constraints for the exception rules

N-ary constraints	Primitive constraints
$freq((X \setminus Y) \sqcup I) \geq minfr$	$F_2 \geq minfr$ $\wedge \quad I \in X_2$ $\wedge \quad X_1 \subsetneq X_3$
$freq(X \setminus Y) - freq((X \setminus Y) \sqcup I) \leq \delta_1$	$F_1 - F_2 \leq \delta_1$ $\wedge \quad X_2 = X_1 \sqcup I$
$freq(X \sqcup \neg I) \leq maxfr$	$F_4 \leq maxfr$ $\wedge \quad \neg I \in X_4$
$freq(X) - freq(X \sqcup \neg I) \leq \delta_2$	$F_3 - F_4 \leq \delta_2$ $\wedge \quad X_4 = X_3 \sqcup \neg I$

The first four constraints correspond to the usual frequent classification. The last constraint expresses the overlapping between the premises: the two rules share at least half of the items of their premises. When a rule of this pair of rules is triggered by an unseen example, it means that it is likely that the other rule of the pair concluding to the another class value is also triggered and thus a classification conflict appears. The user can modify the parameters of the n-ary query and/or add new constraint to model specific classification conflicts.

4 CSP Modeling

We present the CSP modeling shared by the two approaches. Both of the approaches are based on two inter-related CSPs that we introduce in this section. The next section sketches the hybrid approach. Its main feature is to distinguish local constraints from n-ary ones. On the contrary, the full-CP approach does not make such a distinction, but all constraints will have to be reformulated using decision variables and reified constraints (see Section 6).

4.1 General Overview

Let $\boldsymbol{r}$ be a dataset having m transactions, and $\mathcal{I}$ the set of all its items. The itemset mining problem is modeled by using two inter-related CSPs $\mathcal{P}$ and $\mathcal{P}'$:

1. Set CSP $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ where:
   - $\mathcal{X} = \{X_1, ..., X_n\}$. Each variable X_i represents an unknown itemset.
   - $\mathcal{D} = \{D_{X_1}, ..., D_{X_n}\}$. Initial domain of X_i is the set interval $[\{\} \,..\, \mathcal{I}]$.
   - $\mathcal{C}$ is a conjunction of set constraints handling set operators $(\cup, \cap, \in, ...)$
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.
   - $\mathcal{D}' = \{D_{F_1}, ..., D_{F_n}\}$. Initial domain of F_i is the integer interval $[1 \,..\, m]$.
   - $\mathcal{C}'$ is a conjunction of numeric constraints.

For each unknown itemset i, variables X_i and F_i are tied together (see Section 5.2 for the hybrid approach and Section 6.2 for the full-CP approach).

4.2 Example: Modeling the Exception Rules

Table 2 provides the primitive constraints for the exception rules n-ary query modelled in Section 3.1.

- Set variables $\{X_1, X_2, X_3, X_4\}$ represent the four unknown itemsets:
  - $X_1 : X \setminus Y$, and $X_2 : (X \setminus Y) \sqcup I$ for the common sense rule,
  - $X_3 : X$, and $X_4 : X \sqcup \neg I$ for the exception rule.
- Integer variables $\{F_1, F_2, F_3, F_4\}$ represent their frequency values.
- Set constraints: $\mathcal{C} = \{(I \in X_2), (X_2 = X_1 \sqcup I), (\neg I \in X_4), (X_4 = X_3 \sqcup \neg I), (X_1 \subsetneq X_3\})$
- Numeric constraints: $\mathcal{C}' = \{(F_2 \geq \mathit{minfr}), (F_1 - F_2 \leq \delta_1), (F_4 \leq \mathit{maxfr}), (F_3 - F_4 \leq \delta_2)\}$

5 Related Work

5.1 Pattern Discovery Approaches in Data Mining

There are a lot of works to discover local patterns under constraints [7,15] but there are no so many methods to combine local patterns. Recent approaches - pattern teams [14], constraint-based pattern set mining [9] 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 argue that n-ary queries 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. General data mining frameworks based on the notion of local patterns to design global models are presented in [13,11]. These frameworks help to analyze and improve current methods in the area.

5.2 A Hybrid Method

We did a preliminary work [12] based on a joint use of local patterns constraint-based mining and CP in order to discover n-ary patterns and to investigate such relationships. In this section, we provide an overview of the method highlighted by the example of the exception rules n-ary query (see Section 4.2).

i) General overview. The hybrid approach consists in three steps:

1. Modeling the n-ary query as a CSP, then splitting constraints into local ones and n-ary ones.
2. Solving the local constraints using a local patterns extractor (Music-dfs [2]) [18] which produces an interval condensed representation of all patterns satisfying the local constraints.

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

3. Solving the n-ary constraints using ECL^iPS^e[3]. The domain of each variable results from the interval condensed representation (computed in the Step-2).

ii) Splitting the set of constraints. The whole set of constraints $(\mathcal{C} \cup \mathcal{C}')$ is split 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.
- $\mathcal{C}_n$ is the set of n-ary constraints to be solved where the domains of the variables X_i and F_i will be deduced from the interval condensed representation.

For the exception rules (see Table 2) this splitting is performed as follows:

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

iii) Solving the local constraints. MUSIC-DFS is a local patterns extractor which offers a set of syntactic and aggregate primitives to specify a broad spectrum of local constraints in a flexible way [18]. MUSIC-DFS mines soundly and completely all the patterns satisfying a given set of local constraints. 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 [18].

For the hybrid method, local constraints are solved before and regardless n-ary constraints. So that, the search space for n-ary constraints will be reduced to the space of solutions of local constraints. 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: $\forall x \in I_p,\ \mathit{freq}(x) = f_p$.

So domains 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 $\mathcal{C}_{loc}$ the set of local constraints for the exception rules. The respective values for $(I, \neg I, \mathit{minfr}, \delta_1, \mathit{maxfr}, \delta_2)$ are $(c_1, c_2, 2, 1, 1, 0)$. The set of local constraints related to X_2 and F_2, $\mathcal{C}_2 = \{c_1 \in X_2, F_2 \geq 2\}$, is solved by MUSIC-DFS with the following query showing that the parameters can be 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
---------------
```

iv) Solving the n-ary constraints. Domains of the variables X_i and F_i are deduced from the condensed representation of all patterns satisfying local constraints. Solving n-ary constraints using ECL^iPS^e enables then to obtain all the solutions satisfying the whole set of constraints (local ones and n-ary ones).

[3] http://www.eclipse-clp.org

Example: Let us consider $\mathcal{C}_n$ the set of n-ary constraints for the exception rules. The respective values for $(I, \neg I, minfr, \delta_1, maxfr, \delta_2)$ are still the same. The following ECL^iPS^e session illustrates how all pairs of exception rules can be directly 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];
.../...
---------------
```

6 A Full-CP Approach

This section presents a new approach keeping the previous modeling (see Section 4) but using a new solving technique. We call *full-CP* this approach because it only uses a CP solver and no longer a local patterns extractor. Section 7 analyzes in depth the two methods.

Numeric constraints and set constraints are modeled in three steps: linking the data and the patterns involved in the n-ary constraint, then modeling the patterns and finally formulating numeric constraints and set constraints.

We use the CP system Gecode[4]. For the first step (see Section 6.1), we use the implementation of the Itemset Mining system FIM_CP[5] which is an approach using CP for pattern mining [8]. This approach addresses in a unified framework a large set of local patterns and constraints such as frequency, closedness, maximality, constraints that are monotonic or anti-monotonic or variations of these constraints but it does not deal with n-ary constraints.

In the remainder of this section, let $\boldsymbol{r}$ be a dataset where $\mathcal{I}$ is the set of its n items and $\mathcal{T}$ the set of its m transactions, and let $\boldsymbol{d}$ be the 0/1 (m,n) matrix where $\forall t \in \mathcal{T}, \forall i \in \mathcal{I}$, $(d_{t,i} = 1) \Leftrightarrow (i \in t)$.

6.1 Modeling an Unknown Local Pattern

Let M be the unknown local pattern. FIM_CP establishes the link between the data set and M by introducing two kinds of variables, each of them having $\{0, 1\}$ for domain: $\{M_1, M_2, ..., M_n\}$ where $(M_i = 1) \Leftrightarrow (i \in M)$, and $\{T_1, T_2, ..., T_m\}$ where $(T_t = 1) \Leftrightarrow (M \subseteq t)$. So, $freq(M) = \sum_{t \in \mathcal{T}} T_t$ and $length(M) = \sum_{i \in \mathcal{I}} M_i$.

The relationship between M and $\mathcal{T}$ is modeled by reified constraints, stating that, for each transaction t, $(T_t = 1)$ iff t is covered by M:

$$\forall t \in \mathcal{T}, (T_t = 1) \Leftrightarrow \sum_{i \in \mathcal{I}} M_i \times (1 - d_{t,i}) = 0 \quad (1)$$

[4] http://www.gecode.org

[5] http://www.cs.kuleuven.be/~dtai/CP4IM/fim_cp.php

Each reified constraint (see Equation 1) is solved as follows: if the solver deduces (T_t=1) (resp. T_t=0), then the sum must be equal to 0 (resp. must be different from 0). The propagation is also performed, in a same way, from the sum constraint toward the equality constraint.

6.2 Modeling the k Patterns We Are Looking for

Let $X_1, X_2, ..., X_k$ be the k patterns we are looking for. First, each pattern X_j is modeled by n variables $\{X_{1,j}, X_{2,j}, ..., X_{n,j}\}$ having $\{0, 1\}$ for domain and such that $(X_{i,j} = 1)$ iff item i belongs to pattern X_j:

$$\forall i \in \mathcal{I}, (X_{i,j} = 1) \Leftrightarrow (i \in X_j) \tag{2}$$

m variables $\{T_{1,j}, T_{2,j}, ..., T_{m,j}\}$ having $\{0, 1\}$ for domain are associated with each pattern X_j such that $(T_{t,j} = 1)$ iff transaction t is covered by pattern X_j:

$$\forall t \in \mathcal{T}, (T_{t,j} = 1) \Leftrightarrow (X_j \subseteq t) \tag{3}$$

So, $freq(X_j)$=$\sum_{t \in \mathcal{T}} T_{t,j}$ and $length(X_j)$=$\sum_{i \in \mathcal{I}} X_{i,j}$. The relationship between each pattern X_j and $\mathcal{T}$ is modeled by reified sum constraints, stating that, for each transaction t, $(T_{t,j} = 1)$ iff t is covered by X_j:

$$\forall j \in [1..k], \forall t \in \mathcal{T}, (T_{t,j} = 1) \Leftrightarrow \sum_{i \in \mathcal{I}} X_{i,j} \times (1 - d_{t,i}) = 0 \tag{4}$$

6.3 Reformulating Numeric and Set Constraints

Let operator $\texttt{op} \in \{<, \leq, >, \geq, =, \neq\}$; numeric constraints are reformulated as follows:

- $freq(X_p)$ $\texttt{op}$ $\alpha \rightarrow \sum_{t \in \mathcal{T}} T_{t,p}$ $\texttt{op}$ α
- $length(X_p)$ $\texttt{op}$ $\alpha \rightarrow \sum_{i \in \mathcal{I}} X_{i,p}$ $\texttt{op}$ α

Some set constraints (such that equality, inclusion, membership) are directly reformulated using linear numeric constraints:

- $X_p = X_q \rightarrow \forall i \in \mathcal{I}, X_{i,p} = X_{i,q}$
- $X_p \subseteq X_q \rightarrow \forall i \in \mathcal{I}, X_{i,p} \leq X_{i,q}$
- $i_o \in X_p \rightarrow X_{i_0,p} = 1$

Other set constraints (such that intersection, union, difference) can easily be reformulated into boolean constraints using the conversion function ($b :: \{0, 1\} \rightarrow \{False, True\}$) and the usual boolean operators:

- $X_p \cap X_q = X_r \rightarrow \forall i \in \mathcal{I}, b(X_{i,r}) = b(X_{i,p}) \wedge b(X_{i,q})$
- $X_p \cup X_q = X_r \rightarrow \forall i \in \mathcal{I}, b(X_{i,r}) = b(X_{i,p}) \vee b(X_{i,q})$
- $X_p \backslash X_q = X_r \rightarrow \forall i \in \mathcal{I}, b(X_{i,r}) = b(X_{i,p}) \wedge \neg b(X_{i,q})$

Finally, reified constraints (linking the dataset and the patterns, see Equation 4), numeric constraints and set constraints are all managed by *Gecode* .

6.4 Experiments

This section shows the practical usage and the feasibility of our approach.

i) Experimental setup. Experiments were performed on several datasets from the UCI repository[6] and a real-world dataset `Meningitis` coming from the Grenoble Central Hospital[7]. This last dataset gathers children hospitalized for bacterial or viral meningitis. Table 3 summarizes the characteristics of these datasets. Experiments were conducted with several kinds of n-ary queries: exception rules, unexpected rules and classification conflicts. We use a PC having a 2.83 GHz Intel Core 2 Duo processor and 4 GB of RAM, running Ubuntu Linux.

Table 3. Description of the datasets

dataset	#trans	#items	density
`Mushroom`	8142	117	0.18
`Australian`	690	55	0.25
`Meningitis`	329	84	0.27

ii) Soundness and Flexibility. As the resolution performed by the CP solver is sound and complete, our approach is able to mine the correct and complete set of patterns satisfying n-ary queries. Figure 1 (the upper part) depicts the number of pairs of exception rules according to $minfr$ and δ_1 and Figure 1 (the bottom part) indicates the number of classification conflict rules according to the two parameters $minfr$ and $minconf$. These figures show the feasibility of our approach that mines the correct and complete set of all patterns satisfying the n-ary queries from these various sets of parameters. We tested other combinations of the parameters: as they provided similar results, they are not indicated here. As expected, the lower $minfr$ is, the larger the number of pairs of exception rules. Results are similar when δ_1 varies: 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).

iii) Highlighting useful patterns. As already said, exception rules are a particular case of rare rules (cf. Section 2.2). There are few attempts to extract the whole set of rare rules [20]. But, even if these rules can be extracted, it is impossible to pick the exception rules among the set of all the rare rules. That is a strong limitation because most of the rare rules are unreliable and it highlights the interest of pairs of exceptions rules. Figure 2 quantifies the number of pairs of exception rules on the `Meningitis` dataset versus the number of rare rules (the number of rare rules which depends on $maxfr$ corresponds to the line at the top of the figure). Looking for pairs of exception rules reduces on several orders of magnitude the number of outputted patterns (the Y-axis is on a logarithmic scale). A n-ary constraint enables to straightforwardly discover the proper set of exception rules.

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

[7] The authors would like to thank Dr P. François who provided the meningitis dataset.

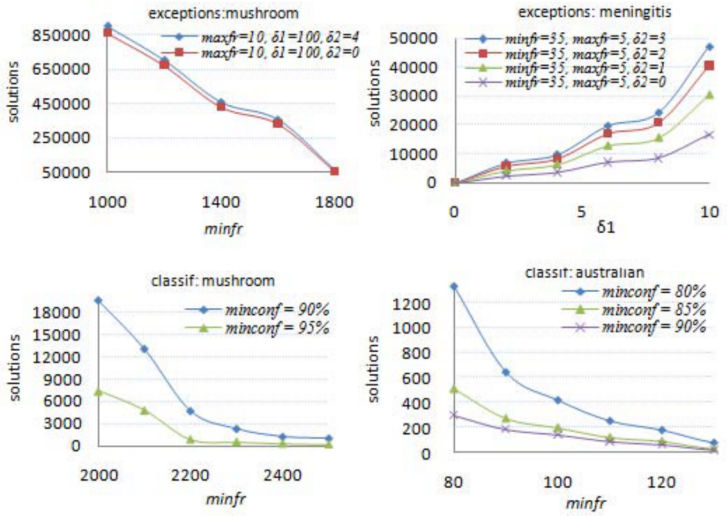

Fig. 1. Number of solutions

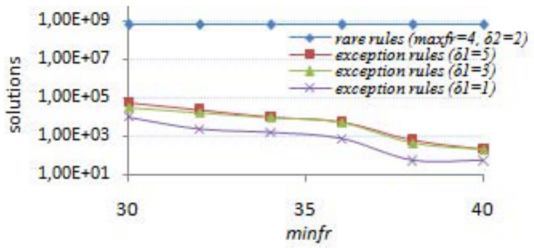

Fig. 2. Number of pairs of exception rules versus number of rare rules (`Meningitis`)

Unexpected rules may reveal useful information. For example, on `Meningitis` dataset, such a rule has a premise made of a high percentage of immature band cells and the absence of neurological deficiency and its conclusion is a normal value of the polynuclear neutrophil level. This rule is unexpected with the belief that high values of the white cells count and the polynuclear percentage lead to a bacterial etiological type. Experts appreciate to have n-ary constraints to address such patterns.

iv) Computational Efficiency. This experiment quantify runtimes and the scalability of our approach. In practice, runtimes vary according to the size of the datasets and the tightness of constraints (a constraint is said *tight* if its number of solutions is low compared to the cardinality of the cartesian product of the variable domains, such as constraints defined by high frequency and confidence thresholds).

For `Meningitis` and `Australian`, the set of all solutions is computed in a few seconds (less than one second in most of the cases). On `Mushroom`, runtimes

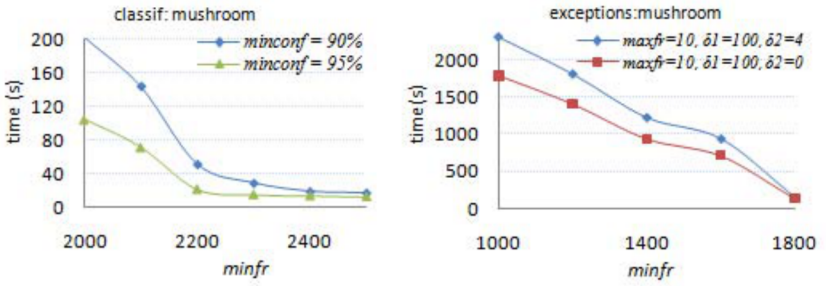

Fig. 3. Runtimes

vary from few seconds for tight constraints to about an hour for low frequency and confidence thresholds. So we have conducted further experiments on this dataset. Figure 3 details the runtime of our method on `Mushroom` according to different thresholds of confidence and frequency. We observe that the tighter the constraint is, the smaller the runtime is. Indeed, tight constraints enable a better filtering of the domains and then a more efficient pruning of the search tree. Runtimes also depend on the size of the dataset: the larger the dataset is, the larger the resulting CP program (cf. Section 6).

Obviously, our generic n-ary approach can be used for mining local patterns. We obtain on this task the same runtimes as [8] which were competitive with state of the art miners. With exception rules, we cannot compare runtimes because they are not indicated in [19].

7 Discussion

The hybrid approach (see Section 5.2) and the full-CP approach proposed in this paper (see Section 6) follow the same modeling described in Section 4. But, they have different solving methods. The hybrid approach uses a local patterns extractor in order to produce an interval condensed representation (of all patterns satisfying the local constraints) that will be used for constituting the domains of the Set CSP variables. The full-CP approach only uses a CP solver and no longer a local patterns extractor.

7.1 Hybrid Approach: Pros and Cons

With the hybrid approach, the modeling of the n-ary query can be directly provided to a Set CSP solver without any reformulation. Thus, a prototype can quickly be developed by using a Set CSP solver such as ECL^iPS^e.

But, Set CSP solvers [10] do not well manage the union of set intervals. In order to establish bound consistency, the union of two set intervals is approximated by its convex closure[8]. To circumvent this problem, for each variable

[8] The convex closure of $[lb_1 .. ub_1]$ and $[lb_2 .. ub_2]$ is defined as $[lb_1 \cap lb_2 .. ub_1 \cup ub_2]$.

X_i with the condensed representation $CR_i = \bigcup_p (f_p, I_p)$, a search is successively performed upon each I_p. If this approach is sound and complete, it does not fully profit from filtering because value removals are propagated only in the handled intervals and not in the whole domains.

This fact could seem to be prohibitive, but the number of set intervals strongly decreases according to local constraints. Table 4 indicates the number of set intervals constituting the domain of variable X_2 according to several local constraints (see the exception rules example Section 5.2).

An alternative solution would be to use non-exact condensed representations to reduce the number of produced intervals (e.g., a condensed representation based on maximal frequent itemsets [6]). In this case, the number of intervals representing the domains will be rather small, but, due to the approximations, it should be necessary to memorize forbidden values.

7.2 Full-CP Approach: Pros and Cons

First, the reformulation of n-ary constraints to low level constraints can be performed in an automatic way. Moreover, the filtering of reified constraints performed by *Gecode* is very effective and performant. But, the resulting number of constraints must be high for very large datasets. Let us consider a dataset with n items and m transactions and a n-ary query involving k unknown patterns. Linking the dataset and the unknown patterns requires $(k \times m)$ constraints, each of them involving at most $(n+1)$ variables (see Equation 4 in Section 6.2). So, the number of constraints necessary to model very large datasets could be prohibitive. In practice, this approach is able to tackle a large set of databases.

To summarize. The two approaches mainly differ in the way they consider the whole dataset. The hybrid approach uses a local patterns extractor and the resulting CSP owns a very small number of constraints, but variables with large domains. On the other hand, the full-CP approach requires a large number of constraints over decision variables. Experiments show that the full-CP approach is significantly more performant than the hybrid one on large datasets. Ratios between their respective runtimes are up to several orders of magnitude. This is due to the lower quality of the filtering of the hybrid approach (see Section 7.1).

Table 4. Number of intervals according to several local constraints (case of D_{X_2})

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

8 Conclusion and Future Works

In this paper, we have presented an approach to model and mine n-ary patterns. To the best of our knowledge, it is the first generic approach to express and mine patterns involving several local patterns. The examples described in Section 3 illustrate the generality and the flexibility of our approach. Experiments show its relevance and its feasibility in spite of its generic scope.

For CSPs, all variables are existentially quantified. Further work is to introduce the universal quantification: this quantifier would be precious to model important constraints such as the *peak* constraint (the *peak* constraint compares neighbor patterns; 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 Quantified CSPs [1,4] could be appropriate and useful.

On the other hand, extracting actionable and interesting knowledge from data is a human-guided, iterative and interactive process. The data analyst should only consider a high-level vision of the pattern discovery system and handle a set of primitives to declaratively specify the pattern discovery task. Even if CP is a new approach to tackle this problem [8,12,16], it appears to be very promising for building such a high level and interactive system.

References

1. Benedetti, M., Lallouet, A., Vautard, J.: Quantified constraint optimization. In: Stuckey, P.J. (ed.) CP 2008. LNCS, vol. 5202, pp. 463–477. Springer, Heidelberg (2008)
2. Besson, J., Robardet, C., Boulicaut, J.-F.: Mining a new fault-tolerant pattern type as an alternative to formal concept discovery. In: ICCS 2006, Aalborg, Denmark, pp. 144–157. Springer, Heidelberg (2006)
3. Bonchi, F., Giannotti, F., Lucchese, C., Orlando, S., Perego, R., Trasarti, R.: A constraint-based querying system for exploratory pattern discovery. Inf. Syst. 34(1), 3–27 (2009)
4. Bordeaux, L., Monfroy, E.: Beyond NP: Arc-consistency for quantified constraints. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 371–386. Springer, Heidelberg (2002)
5. Bringmann, B., Zimmermann, A.: The chosen few: On identifying valuable patterns. In: 12th IEEE Int. Conf. on Data Mining (ICDM 2007), pp. 63–72 (2007)
6. Burdick, D., Calimlim, M., Flannick, J., Gehrke, J., Yiu, T.: Mafia: A performance study of mining maximal frequent itemsets. In: FIMI 2003. CEUR Workshop Proceedings, vol. 90, CEUR-WS.org (2003)
7. Calders, T., Rigotti, C., Boulicaut, J.-F.: A survey on condensed representations for frequent sets. In: Boulicaut, J.-F., De Raedt, L., Mannila, H. (eds.) Constraint-Based Mining and Inductive Databases. LNCS (LNAI), vol. 3848, pp. 64–80. Springer, Heidelberg (2006)
8. De Raedt, L., Guns, T., Nijssen, S.: Constraint Programming for Itemset Mining. In: ACM SIGKDD Int. Conf. KDD 2008, Las Vegas, Nevada, USA (2008)
9. De Raedt, L., Zimmermann, A.: Constraint-based pattern set mining. In: 7th SIAM Int. Conf. on Data Mining. SIAM, Philadelphia (2007)

10. Gervet, C.: Interval Propagation to Reason about Sets: Definition and Implementation of a Practical Language. Constraints 1(3), 191–244 (1997)
11. Giacometti, A., Khanjari Miyaneh, E., Marcel, P., Soulet, A.: A framework for pattern-based global models. In: Corchado, E., Yin, H. (eds.) IDEAL 2009. LNCS, vol. 5788, pp. 433–440. Springer, Heidelberg (2009)
12. Khiari, M., Boizumault, P., Crémilleux, B.: Combining CSP and constraint-based mining for pattern discovery. In: Taniar, D., Gervasi, O., Murgante, B., Pardede, E., Apduhan, B.O. (eds.) ICCSA 2010. LNCS, vol. 6017, pp. 432–447. Springer, Heidelberg (2010)
13. Knobbe, A., Crémilleux, B., Fürnkranz, J., Scholz, M.: From local patterns to global models: The lego approach to data mining. In: Int. Workshop LeGo co-located with ECML/PKDD 2008, Antwerp, Belgium, pp. 1–16 (2008)
14. Knobbe, A., Ho, E.: Pattern teams. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 577–584. Springer, Heidelberg (2006)
15. Ng, R.T., Lakshmanan, V.S., Han, J., Pang, A.: Exploratory mining and pruning optimizations of constrained associations rules. In: Proceedings of ACM SIGMOD 1998, pp. 13–24. ACM Press, New York (1998)
16. Nijssen, S., Guns, T., De Raedt, L.: Correlated itemset mining in ROC space: a constraint programming approach. In: KDD 2009, pp. 647–655 (2009)
17. Padmanabhan, B., Tuzhilin, A.: A belief-driven method for discovering unexpected patterns. In: KDD, pp. 94–100 (1998)
18. Soulet, A., Klema, J., Crémilleux, B.: Efficient mining under rich constraints derived from various datasets. In: Džeroski, S., Struyf, J. (eds.) KDID 2006. LNCS, vol. 4747, pp. 223–239. Springer, Heidelberg (2007)
19. Suzuki, E.: Undirected Discovery of Interesting Exception Rules. Int. Journal of Pattern Recognition and Artificial Intelligence 16(8), 1065–1086 (2002)
20. Szathmary, L., Napoli, A., Valtchev, P.: Towards Rare Itemset Mining. In: Proc. of the 19th IEEE ICTAI 2007, Patras, Greece, vol. 1 (2007)
21. Yin, X., Han, J.: CPAR: classification based on predictive association rules. In: Proceedings of the 2003 SIAM Int. Conf. on Data Mining, SDM 2003 (2003)