ABSTRACT

Discovering pattern sets or global patterns is an attractive issue from the pattern mining community in order to provide useful information. By combining local patterns satisfying a joint meaning, this approach produces patterns of higher level and thus more useful for the end-user than the usual local patterns. In parallel, recent works investigating relationships between data mining and constraint programming (CP) show that the CP paradigm is a powerful framework to model and mine patterns in a declarative and generic way. We present a constraint-based language which enables us to define queries in a declarative way addressing pattern sets and global patterns. By specifying what the task is, rather than providing how the solution should be computed, it is easy to process by stepwise refinements to successfully discover global patterns. The usefulness of the approach is highlighted by several examples coming from the clustering based on associations. All primitive constraints of the language are modeled and solved using the SAT framework. We illustrate the efficiency of our approach through several experiments.

Categories and Subject Descriptors
H.2.8 [Information Systems]: data mining

General Terms
Declarative approach, Constrained-based language

Keywords
Pattern set mining, Clustering, SAT modeling and solving.

1. INTRODUCTION

The process of extracting useful patterns from data, called pattern mining, is an important tool for data analysis and has been used in a wide range of applications and domains. A large amount of work has been developed and many pattern extraction problems are now identified and understood from both theoretical and computational perspectives. Local pattern discovery has become a growing field [20] and several paradigms are available for producing extensive collections of patterns such as the constraint-based pattern mining [21] or condensed representations of patterns [4]. Because of the exhaustive nature of the techniques, the pattern collections provide a fairly complete picture of the information content of the data. However, the approach suffers from limitations. First, the collections of patterns still remain too large for an individual and global analysis performed by the data analyst. Secondly, the so-called local patterns represent fragmented information whereas patterns expected by the data analyst require to consider simultaneously several local patterns. That is why combining local patterns to get global patterns is highly attractive.

The data mining literature includes several methods to take into account the relationships between patterns and produce global patterns or pattern sets [5, 10]. Recent approaches - constraint-based pattern set mining [5], pattern teams [16] and selecting patterns according to the added value of a new pattern given the currently selected patterns [3] - 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. Nevertheless, these methods are mainly based on the reduction of the redundancy (i.e., basically, the overlap between the data covered by the patterns [5]) or specific aims such as classification processes. The difficulty of the task may explain the use of heuristic functions and the lack of complete and correct methods to mine global patterns. Indeed, mining local patterns requires the exploration of a large search space but mining global patterns is even harder because solutions satisfying each pattern must be compared. Clearly, the lack of generic approaches restrains the discovery of useful global patterns because the user has to develop a new method each time he wants to extract a new kind of global patterns. It explains why this issue deserves our attention.

In this paper, we propose a constraint-based language to discover patterns combining several local patterns. The key idea is to propose a declarative and generic approach to ask queries: the user models a problem by specifying a set of constraints and expresses his queries thanks to constraints over terms built from constants, variables, operators, and...
function symbols. Queries and built-in constraints of the language are encoded and solved using the SAT framework. Our approach takes benefit of the recent progress on cross-fertilization between data mining and Constraint Programming \cite{12, 13, 14, 23}. The definition of a constraint-based language offers the great advantage to provide a declarative method to address different pattern mining problems: it is enough to change the specification in term of constraints. We illustrate the approach by several examples coming from the clustering based on associations. With simple query refinements, the data analyst is able to easily produce clusterings satisfying different properties. By specifying what the task is, rather than how the solution should be computed, the process greatly facilitates the search of global patterns and the discovery of knowledge.

This paper is organized as follows. Section 2 describes the constraint-based language and shows how queries and constraints can be defined using terms and built-in constraints. Starting from the clustering example, Section 3 describes the process of successive refinements which enables us to easily address several kinds of clustering and the discovery of global models. Section 4 describes how queries and built-in constraints of the language are modeled and solved using the SAT framework. Section 5 demonstrates the efficiency of our approach through several experiments. We review related work in Section 6.

2. A CONSTRAINT-BASED LANGUAGE

This section describes the constraint-based language we propose. Terms are built using constants, variables, operators, and function symbols. Constraints are relations over terms that can be satisfied or not. The data analyst can define new function symbols. We show how queries and constraints can be written using built-in constraints.

2.1 Definitions and example

Let $I$ be a set of $n$ distinct literals called items, an itemset (or pattern) is a non-null subset of $I$. The language of itemsets corresponds to $L_T = 2^I \setminus \emptyset$. A transactional dataset is a multi-set of m itemsets of $L_T$. Each itemset, usually called a transaction or object, is a database entry. For instance, Table 1 gives a transactional dataset $T$. The language are encoded and solved using the SAT framework.

`Table 1: Transactional dataset $T$.`

<table>
<thead>
<tr>
<th>Trans.</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>A D F</td>
</tr>
<tr>
<td>$t_2$</td>
<td>A E F</td>
</tr>
<tr>
<td>$t_3$</td>
<td>A E G</td>
</tr>
<tr>
<td>$t_4$</td>
<td>A E G</td>
</tr>
<tr>
<td>$t_5$</td>
<td>B E G</td>
</tr>
<tr>
<td>$t_6$</td>
<td>B E G</td>
</tr>
<tr>
<td>$t_7$</td>
<td>C E G</td>
</tr>
<tr>
<td>$t_8$</td>
<td>C E G</td>
</tr>
<tr>
<td>$t_9$</td>
<td>C E H</td>
</tr>
<tr>
<td>$t_{10}$</td>
<td>C E H</td>
</tr>
<tr>
<td>$t_{11}$</td>
<td>C F G H</td>
</tr>
</tbody>
</table>

The constraint based language owns predefined (built-in) function symbols\footnote{Only function symbols (and constraints) used in Section 3 are introduced in this paper. } like:

- $\text{cover}(X_i) = \{ t \mid t \in T, X_i \subseteq t \}$ is the set of transactions covered by $X_i$.
- $\text{freq}(X_i) = | \{ t \mid t \in T, X_i \subseteq t \} |$
- $\text{size}(X_i) = | \{ j \mid j \in I, j \notin X_i \} |$
- $\text{overlapItems}(X_i, X_j) = | X_i \cap X_j |$ is the number of items shared by both $X_i$ and $X_j$.
- $\text{overlapTransactions}(X_i, X_j) = | \text{cover}(X_i) \cap \text{cover}(X_j) |$ is the number of transactions covered by $X_i$ and $X_j$.

2.2.1 Built-in function symbols

The data analyst can define new function symbols using constants, variables, operators and existing function symbols (built-in or previously defined ones). Examples:

- $\text{area}(X_i) = \text{freq}(X_i) \times \text{size}(X_i)$
- $\text{coverage}(X_i, X_j) = \text{freq}(X_i \cup X_j) \times \text{size}(X_i \cap X_j)$

Interestingness measures can be straightforwardly defined. Here is the example of the growth-rate well-used in contrast mining \cite{22}. Let $D_1, D_2 \subset T$ be 2 sets of transactions (i.e., classes) and $\text{freq}^2(X_i, D_j)$ the frequency of $X_i$ into $D_j$, the growth-rate of $X_i$ in $D_1$ is $\text{gr}_1(X_i) = |D_2| \times \text{freq}^2(X_i, D_2) / |D_1| \times \text{freq}^2(X_i, D_1)$

2.3 Constraints and Queries

Constraints are relations over terms that can be satisfied or not. There are three kinds of built-in constraints:

i) **numerical constraints**: $<, \leq, =, \neq, \geq, >$. Examples:

- $\text{freq}(X_i) \leq 10$
- $\text{size}(X_2) = 2 \times \text{size}(X_3)$
- $\text{area}(X_1) < \text{size}(X_2) \times \text{freq}(X_3)$

ii) **set constraints** like: $\in, \notin, \notin, \subset, \subseteq$. Examples:

- $A \in X_1$
- $X_1 \cup X_2 \subseteq X_3$
- $X_1 = X_2 \cap X_4$

iii) **dedicated constraints** like:

- $\text{closed} (X_1)$ is satisfied iff $X_1$ is a closed\footnote{Let $T_{X_1}$ be the set of transactions covered by pattern $X_1$. $X_1$ is closed iff $X_1$ is the largest (c) pattern covering $T_{X_1}$. } pattern.
- $\text{coverTransactions}([X_1, ..., X_k])$ is satisfied iff each transaction is covered by at least one pattern $\bigcup_{1 \leq i \leq k} \text{cover}(X_i) = T$.
- $\text{coverItems}([X_1, ..., X_k])$ is satisfied if every item belongs to at least one pattern (i.e. $\bigcup_{1 \leq i \leq k} X_i = T$).
- $\text{canonical}([X_1, ..., X_k])$ is satisfied iff for all i s.t. $1 \leq i \leq k$, pattern $X_i$ is less than pattern $X_{i+1}$ with respect to the lexicographic order.


Queries are formulae built using constraints and logical connectors: ∧ (conjunction) and ∨ (disjunction).

3. MINING BY REFINING QUERIES

A major strength of our approach is to provide a simple and efficient way to declare and refine queries. In practice, the data analyst starts by writing a first query. Then, he successively refines the query (deriving Q_{i+1} from Q_i) until he considers that relevant information has been extracted. We illustrate this approach with the clustering problem. Clustering aims at partitioning data into groups (clusters) so that transactions are similar inside each cluster but different between clusters [8]. We selected clustering because it is an important and popular data mining task and, by nature, clustering proceeds by iteratively refining queries until a satisfactory solution is found. Our approach is also well-suited to integrate constraints handled in constraint-based clustering [1, 2].

3.1 Modeling a clustering query

The closed patterns are well-designed for clustering based on associations because a closed pattern gathers the maximum amount of similarity between a set of transactions. Thus, a closed pattern is a candidate cluster. The standard clustering problem can then be formalized as: “to find a set of k closed patterns X_1, X_2, ..., X_k (i.e., clusters) covering all transactions without any overlap on these transactions”.

Our constraint-based language offers the constraints to express this query: the closed(X_i) constraints enforce each unknown pattern X_i to be closed, the coverTransactions constraint ensures to cover all the transactions. To avoid an overlap between transactions, we add for each couple of patterns (X_i, X_j) s.t. i<j the overlapTransactions(X_i, X_j)=0 constraint. This constraint states that there exists no transaction covered by both X_i and X_j.

Moreover, a clustering problem intrinsically owns a lot of symmetrical solutions. Let s = (p_1, p_2, ..., p_k) be a solution containing k patterns p_i, any permutation σ of these k patterns σ(s) = (p_{σ(1)}, p_{σ(2)}, ..., p_{σ(k)}) is also a solution. The canonical([X_1, ..., X_k]) constraint is used to avoid computing symmetrical solutions. This constraint ensures that, for all i s.t. 1 ≤ i < k, pattern X_i is before pattern X_{i+1} with respect to the lexicographic order. The canonical([X_1, ..., X_k]) constraint plays an important role. As for a clustering involving k clusters, the number of symmetrical solutions is k!, it is crucial to break the symmetries to avoid obtaining a huge number of redundant solutions. Moreover, this constraint performs an efficient filtering by drastically reducing the size of the search space.

Finally, we get the following query (Q_1) modeling the initial clustering problem:

\[ \left\{ \begin{align*}
 \land_{1 \leq i \leq k} \text{closed}(X_i) \\
 \land_{1 \leq i \leq k} \text{coverTransactions}([X_1, ..., X_k]) \\
 \land_{1 \leq i \leq j < k} \text{overlapTransactions}(X_i, X_j) = 0 \\
 \land_{1 \leq i \leq j < k} \text{canonical}([X_1, ..., X_k])
\end{align*} \]

On our running example, with k=3 patterns, query Q_1 provides 5 solutions (see Table 2).

3.2 Refining queries

By only refining queries on a clustering, the data analyst can easily produce other clusterings satisfying different properties. This section illustrates this feature of our approach that facilitates the building of global patterns and the discovery of knowledge. From the initial query Q_1, we derive queries Q_2 and Q_3 avoiding clusterings with non-frequent patterns and clusterings with small size patterns.

3.2.1 Removing solutions with non-frequent patterns

When a cluster has a low frequency, it lacks of representativity and the clustering is not considered as reliable. From the initial query Q_1, it is easy to add frequency constraints ensuring that each pattern is frequent. With a frequency threshold δ_1=2, we get Q_2:

\[ \left\{ \begin{align*}
 \land_{1 \leq i \leq k} \text{closed}(X_i) \\
 \land_{1 \leq i \leq k} \text{coverTransactions}([X_1, ..., X_k]) \\
 \land_{1 \leq i \leq j < k} \text{overlapTransactions}(X_i, X_j) = 0 \\
 \land_{1 \leq i \leq j < k} \text{canonical}([X_1, ..., X_k]) \\
 \land_{1 \leq i \leq k} \text{freq}(X_i) \geq \delta_1
\end{align*} \]

Pattern [C, F, G, H] of solution s_1 has a frequency of 1 and thus is removed. With Q_2, there remain 4 solutions (s_2, s_3, s_4, and s_5, see Table 2).

3.2.2 Removing solutions with small size patterns

A clustering with at least one cluster X_i of small size\(^3\) is not considered as useful because X_i does not ensure enough similarity between transactions associated to X_i. It is simple to add constraints requiring that the size of each pattern is higher than a minimal size. From Q_2, with a minimal size threshold δ_2=2, we obtain the query Q_3:

\[ \left\{ \begin{align*}
 \land_{1 \leq i \leq k} \text{closed}(X_i) \\
 \land_{1 \leq i \leq k} \text{coverTransactions}([X_1, ..., X_k]) \\
 \land_{1 \leq i \leq j < k} \text{overlapTransactions}(X_i, X_j) = 0 \\
 \land_{1 \leq i \leq j < k} \text{canonical}([X_1, ..., X_k]) \\
 \land_{1 \leq i \leq k} \text{size}(X_i) \geq \delta_2
\end{align*} \]

With the query Q_3, there is only one solution: s_2 with X_1=[A, F], X_2=[C, H] and X_3=[E, G] (cf. Table 2).

\(^3\)Moreover, clustering with clusters of size 1 often reflects values coming from the binarization of an attribute (such as A, B and C in Table 1) and are useless.

<table>
<thead>
<tr>
<th>Sol.</th>
<th>X_1</th>
<th>X_2</th>
<th>X_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_1</td>
<td>{C, F, G, H}</td>
<td>{E}</td>
<td>{A, D, F}</td>
</tr>
<tr>
<td>s_2</td>
<td>{A, D, F}</td>
<td>{C, F, G, H}</td>
<td>{E}</td>
</tr>
<tr>
<td>s_3</td>
<td>{A, F}</td>
<td>{C, F, G, H}</td>
<td>{E}</td>
</tr>
<tr>
<td>s_4</td>
<td>{A, E, F}</td>
<td>{C, F, G, H}</td>
<td>{E}</td>
</tr>
<tr>
<td>s_5</td>
<td>{A, D, F}</td>
<td>{C}</td>
<td>{F}</td>
</tr>
</tbody>
</table>

Table 2: Set of different clusterings (Q_1).

<table>
<thead>
<tr>
<th>Sol.</th>
<th>X_1</th>
<th>X_2</th>
<th>X_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_1</td>
<td>{C, F, G, H}</td>
<td>{E}</td>
<td>{A, D, F}</td>
</tr>
<tr>
<td>s_2</td>
<td>{A, D, F}</td>
<td>{C, F, G, H}</td>
<td>{E}</td>
</tr>
<tr>
<td>s_3</td>
<td>{A, F}</td>
<td>{C, F, G, H}</td>
<td>{E}</td>
</tr>
<tr>
<td>s_4</td>
<td>{A, E, F}</td>
<td>{E}</td>
<td>{F}</td>
</tr>
<tr>
<td>s_5</td>
<td>{A, D, F}</td>
<td>{E}</td>
<td>{F}</td>
</tr>
</tbody>
</table>

Table 3: Set of different clusterings (Q_4).
3.3 Solving other Clustering Problems

In the same way, it is easy to express other clustering problems [2] such as soft clustering, co-clustering, and soft co-clustering.

3.3.1 Soft clustering

This problem is a relaxed version of the clustering where small overlaps on transactions (less than a threshold \( \delta_T \)) are allowed. The query \( Q_4 \) (soft version of \( Q_1 \)) models this problem:

\[
\begin{cases}
\land_{1 \leq i \leq k} \text{closed}(X_i) \\
\land_{1 \leq i < j \leq k} \text{overlapTransactions}(X_i, X_j) \leq \delta_T \\
\text{canonical}([X_1, \ldots, X_k])
\end{cases}
\]

With \( k=3 \) and a maximal overlap between transactions \( \delta_T=1 \), \( Q_4 \) produces 13 solutions (see Table 3).

3.3.2 Co-clustering

The co-clustering task consists in finding \( k \) clusters covering both the set of transactions and the set of items, without any overlap on transactions or on items. Query \( Q_5 \) expresses this problem:

\[
\begin{cases}
\land_{1 \leq i \leq k} \text{closed}(X_i) \\
\land_{1 \leq i < j \leq k} \text{overlapTransactions}(X_i, X_j) = 0 \\
\land_{1 \leq i \leq k} \text{coverTransactions}(X_i, T) \\
\land_{1 \leq i \leq k} \text{overlapTransactions}(X_i, I) = 0 \\
\land_{1 \leq i < j \leq k} \text{overlapItems}(X_i, X_j) = 0 \\
\text{canonical}([X_1, \ldots, X_k])
\end{cases}
\]

3.3.3 Soft co-clustering

This problem is a relaxed version of the co-clustering, allowing small overlaps on transactions (less than \( \delta_T \)) and on items (less than \( \delta_I \)). The query \( Q_6 \) (soft version of \( Q_4 \) and \( Q_5 \)) models this task:

\[
\begin{cases}
\land_{1 \leq i \leq k} \text{closed}(X_i) \\
\land_{1 \leq i < j \leq k} \text{overlapTransactions}(X_i, X_j) \leq \delta_T \\
\land_{1 \leq i \leq k} \text{coverTransactions}(X_i, I) \\
\land_{1 \leq i < j \leq k} \text{overlapItems}(X_i, X_j) \leq \delta_I \\
\land_{1 \leq i \leq k} \lvert \text{freq}(X_i) - \text{freq}(X_j) \rvert \leq \Delta \times m
\end{cases}
\]

3.3.4 Balanced clustering

In clustering, we generally prefer solutions in which the frequencies of the clusters do not differ too much from each other. Query \( Q_7 \) describes clusterings with balanced frequencies. For any couple of clusters \((X_i, X_j)\), their difference of frequencies must be lower than a threshold \( \Delta \times m \) where \( \Delta \) is a percentage. Looking for clusterings with balanced size clusters could be achieved in a same way:

\[
\begin{cases}
\land_{1 \leq i \leq k} \text{closed}(X_i) \\
\land_{1 \leq i < j \leq k} \text{overlapTransactions}(X_i, X_j) = 0 \\
\land_{1 \leq i \leq k} \text{coverTransactions}(X_i, I) \\
\land_{1 \leq i < j \leq k} \lvert \text{freq}(X_i) - \text{freq}(X_j) \rvert \leq \Delta \times m
\end{cases}
\]

4. Modeling and Solving Using SAT

Satisfiability (SAT) is the problem of determining if the variables of a given boolean formula can be assigned in such a way as to make the formula be evaluated to True. A formula is in conjunctive normal form (CNF) if it is a conjunction of clauses, where a clause is a disjunction of literals and a literal is either a variable \( x_i \) or its negation \( \neg x_i \).

Even if the SAT problem is NP-complete, efficient and scalable algorithms for SAT, that were developed over the last decade, have contributed to dramatic advances in the ability to automatically solve problem instances involving tens of thousands of variables and millions of constraints. That is why, we have chosen to transform a query into a CNF and then use a SAT solver to find its solutions.

In the remainder of this section, let \((d_{t,i})\) be the \((m,n)\) boolean matrix where \((d_{t,i}=True)\) iff \((i \in t)\). Queries are modeled in two steps. First, unknown patterns are modeled using boolean variables and matrix \((d_{t,i})\). Then, each built-in constraint its expressed using a CNF.

4.1 Modeling unknown patterns

Let \( X_1, X_2, \ldots, X_k \) be the \( k \) patterns we are looking for. In a same way as [14, 23], the link between the data set \( T \) and an unknown pattern \( X_j \) is performed by introducing two kinds of boolean variables:

- \( X_{i,j}, X_{2,j}, \ldots, X_{n,j} \) s.t. \((X_{i,j} = True)\) iff \((i \in X_j)\)
- \( T_{i,j}, T_{2,j}, \ldots, T_{m,j} \) s.t. \((T_{i,j} = True)\) iff \((X_j \subseteq t)\)

The relation \((X_j \subseteq t)\) can be transformed into the following CNF:

\[
\bigwedge_{i \in X_j} \neg X_{i,j} \tag{1}
\]

The relationship between \( X_j \) and \( T \) is modeled by stating that, for each transaction \( t \), \((T_{i,j} = True)\) iff \( X_j \) covers \( t \):

\[
\forall t \in T, T_{i,j} \iff (X_j \subseteq t) \tag{2}
\]

Using Eq. 1, the left to right implication of Eq. 2 can be transformed into the following CNF:

\[
\bigwedge_{i \in T} \bigwedge_{(i \in X_j)} (\lnot T_{i,j} \lor \lnot X_{i,j}) \tag{3}
\]

Using Eq. 1, the right to left implication of Eq. 2 can be transformed into the following CNF:

\[
\bigwedge_{i \in T} (\bigvee_{(i \in X_j)} X_{i,j} \lor T_{i,j}) \tag{4}
\]

Finally, Eq. 3 and Eq. 4 must hold for every \( X_j \), \( 1 \leq j \leq k \).
So, the SAT encoding of a query with \( k \) unknown patterns requires \( k \times m \times n \) binary clauses.

4.2 Constraints as boolean formulae

This section provides the boolean formulae associated to built-in constraints. The following ones have a straightforward encoding:

- \( X_p = X_q \rightarrow \bigwedge_{i \in X_p} (X_{i,p} \leftrightarrow X_{i,q}) \)
- \( i \in X_p \rightarrow X_{i,p} \)
- \( X_p \subseteq X_q \rightarrow \bigwedge_{i \in X_p} (X_{i,p} \leftrightarrow X_{i,q}) \)
- \( X_p \cup X_q = X_r \rightarrow \bigwedge_{i \in X_p} (X_{i,p} \lor X_{i,q}) \)
- \( X_p \cap X_q = X_r \rightarrow \bigwedge_{i \in X_p} (X_{i,p} \land X_{i,q}) \)
- \( X_p \setminus X_q = X_r \rightarrow \bigwedge_{i \in X_p} (X_{i,p} \land \neg X_{i,q}) \)
- \( \text{coverItems}([X_1, \ldots, X_k]) \rightarrow \bigwedge_{i \in T} (X_{i,1} \lor \ldots \lor X_{i,k}) \)
- \( \text{coverTransactions}([X_1, \ldots, X_k]) \rightarrow \bigwedge_{j \in 1..k} (T_{j,1} \lor \ldots \lor T_{j,t}) \)

All threshold constraints are modeled using the sorting network approach (for technical details, see [7]) in order to
prevent prohibitive grounding. Using such an encoding, the size of the CNF modelling a threshold constraint is independent from the value of the threshold but depends on the maximal value for the considered measure. For example, the size of the CNF for constraint \( \text{freq}(X_i) \geq \delta_1 \) is \( O(m \times \log(m)) \), and does not depend on \( \delta_1 \). The size of the CNF for constraint \( \text{size}(X_i) \leq \delta_2 \) is \( O(n \times \log(n)) \), and does not depend on \( \delta_2 \). Threshold constraints for measures like \( \text{overlapItems} \) and \( \text{overlapTransactions} \) are encoded in the same way.

### 4.3 Ensuring completeness

Given a CNF, SAT solvers either find one instanciation (and only one) for the variables evaluating the formula to True, or prove there is no such an instanciation. In order to ensure the completeness of our approach, restarts are performed.

Let \( \mathcal{F} \) be the CNF modeling a query \( Q \). \( \mathcal{F} \) is the conjunction of the CNF associated to the modeling of unknown patterns (see Section 4.1) and the CNFs associated to the modeling of the constraints involved in \( Q \) (see Section 4.2). Resolution begins with \( \mathcal{F} \). Then, after having obtained the \( i \)-th solution \( s_i \), its negation \( \neg s_i \) is added to the (current) CNF and resolution is restarted in order to look for another solution. The process ends when a failure occurs, i.e. when all solutions have been found.

Using restarts may seem too naive, but in practice is powerful (see experiments in Section 5.1). This is because the CNF \( \mathcal{F} \) contains much binary clauses (see Section 4.1), and so, filtering by unit propagation will be very effective.

The SAT solver MiniSat \(^4\) [6] has been used for experiments, because its implementation proved easy to modify and MiniSat is one of the most efficient SAT solvers.

### 5. EXPERIMENTS

Recalling that the key contribution of this paper is to propose a constraint-based language to model and mine global patterns in a declarative way. Therefore, the goal of the experiments is to provide better insights on the use of our approach in order to discover global patterns with the example of clustering based on associations.

Experiments were performed on several benchmarks from the UCI repository\(^7\) and also a real-world dataset Meningitis gathering 329 children hospitalized for acute meningitis. Characteristics of datasets are presented in Table 4. Experiments were conducted on a PC having a 2.83 GHz Intel Core 2 Duo Processor and 4GB of RAM, running Ubuntu Linux.

#### 5.1 Clustering Queries

This section illustrates the successive query refinement depicted in Section 3 and leading to the query \( Q_3 \).

\(^{4}\)http://minisat.se/

\(^{7}\)http://www.ics.uci.edu/~mlearn/MLRepository.html

---

### Table 4: Description of the datasets.

<table>
<thead>
<tr>
<th>dataset</th>
<th>#transactions</th>
<th>#items</th>
<th>density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australian</td>
<td>690</td>
<td>125</td>
<td>0.40</td>
</tr>
<tr>
<td>Mushroom</td>
<td>8124</td>
<td>119</td>
<td>0.19</td>
</tr>
<tr>
<td>Soybean</td>
<td>630</td>
<td>50</td>
<td>0.32</td>
</tr>
<tr>
<td>Primary-Tumor</td>
<td>336</td>
<td>36</td>
<td>0.48</td>
</tr>
<tr>
<td>Zoo</td>
<td>101</td>
<td>36</td>
<td>0.44</td>
</tr>
<tr>
<td>Meningitis</td>
<td>329</td>
<td>82</td>
<td>0.26</td>
</tr>
</tbody>
</table>

---

**Figure 1:** NoS. for refinements \( Q_1 \) to \( Q_3 \) (Soybean).

**Figure 2:** NoS. for refinements \( Q_1 \) to \( Q_3 \) (Meningitis).

**Figure 3:** NoS. to query \( Q_3 \) on several datasets.

Fig. 1 (resp. Fig. 2) gives the total number of solutions for queries \( Q_1 \), \( Q_2 \), and \( Q_3 \) for the dataset \textit{Soybean} (resp. \textit{Meningitis}) according to \( k \) (i.e., the number of patterns). Applying the stepwise refinements from query \( Q_1 \) to query \( Q_3 \) drastically reduces the number of clusterings and highlights on the most promising ones (note that the y-axis is a logarithmic scale).

Fig. 3 depicts the number of solutions to query \( Q_3 \) on several datasets. In some cases, the query can be satisfied by a large number of solutions and we do not report results when there are more than one million of solutions. It especially happens with \textit{Australian} because this dataset provides a huge number of closed patterns. In such situations, the query should be refined (by increasing the minimal frequency and/or size thresholds or adding new constraints).

The completeness of our approach is ensured by restarting the SAT solver. Each time a new solution is found, its negation is stored and added to the current CNF. Memory consumption becomes too high above millions of solutions. From a practical point of view, as the data analyst is inter-
**5.2 Soft Clustering**

Fig. 5 and Fig. 6 provide soft clustering results with query $Q_3$ on datasets **Mushroom** and **Primary-Tumor**. Curves indicate the number of solutions according to $k$ and the maximal overlap threshold $\delta_T$ on transactions. As expected, the number of solutions increases with the size of the overlap. The number of solutions can be easily controlled by adjusting (decreasing/increasing) $\delta_T$.

**5.3 Balanced Clustering**

We performed experiments with query $Q_\gamma$. Fig. 7 and Fig. 8 display curves indicating the number of solutions on datasets **Zoo** and **Primary-Tumor** according to $k$ and $\Delta$. Adding the balanced frequencies constraint strongly reduces the number of solutions. For dataset **Zoo**, only a gap $\Delta=40\%$ allows to find a solution for $k=2$. Moreover, with $k=3$ and $k=4$, there is no solution with a gap $\Delta=10\%$.

**6. RELATED WORK**

The data mining literature includes many approaches to reduce the number of produced patterns such as the condensed representations of patterns [4], the compression of the dataset by exploiting Minimum Description Length Principle [25], the discovery of $k$ representative patterns with probabilistic models for summarizing frequent patterns [19]. These approaches mainly aim at reducing the redundancy between patterns and often focus on frequent patterns.

Taking into account the relationships between patterns is explicitly required to produce global patterns or pattern sets [3, 5, 10]. A large number of these methods are based on two-step techniques. The first step generates an exhaustive collection of local patterns and then the patterns are heuristically post-processed to select a smaller subset of complementary relevant patterns such as in associative classification [18]. In [5], the authors propose a framework to mine constraint-based pattern sets by adapting existing techniques for itemset mining to pattern set mining. Nevertheless, the extraction of the addressed pattern sets have to satisfy properties such as the (anti-)monotonicity. There are few methods without heuristic to mine complete and correct pattern sets or global patterns and in practice running tech-
niques are devoted to specific kinds of global patterns [17, 24]. General data mining frameworks based on the notion of local patterns to design global models are presented in [10, 15]. These frameworks help to analyze and improve current methods in the area.

Recent works [12, 13, 14, 23] have shown the cross-fertilization between data mining and Constraint Programming (CP). Indeed, CP provides a general declarative methodology for modeling and solving constraint problems. CP facilitates the design of generic methods handling several patterns, that is a key point in pattern set mining. Techniques for mining itemsets and n-ary patterns (i.e., the combination of n patterns) have been proposed. Looking for declarative techniques in pattern mining was also recently investigated in [11]: by using relational algebra, the authors propose an algebraic framework for pattern discovery for expressing a wide range of queries.

7. CONCLUSION AND FUTURE WORK

We have proposed a constraint-based language allowing to easily express different mining tasks in a declarative way. Thanks to the declarative process, extending or changing the specification to refine the results and discover more relevant patterns or address new global patterns is very simple. Moreover, all constraints can be combined together and new constraints can be added. The efficiency and the flexibility of our approach is shown on several examples coming from clustering based on associations. Thanks to query refinements, the user is able to produce clusterings satisfying different constraints and generating more meaningful clusters.

As future work, we want to enrich our constraint-based language with further constraints to capture and model a wider range of data mining tasks. The scalability of the approach to larger values of k and larger datasets will also be investigated. Another promising direction is to integrate optimisation criteria in our framework.

Acknowledgements: This work is partly supported by Region Basse-Normandie (France) and ANR funded project Bingo2 ANR-07-MDCO-014.

8. REFERENCES