A Generic Approach for Modeling and Mining n-ary Patterns

Medhi Khiari, Patrice Boizumault, Bruno Crémilleux

A Generic Approach for Modeling and 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 [1], these methods do not take into account the interest of a pattern with respect to the other patterns which are mined. In practice, a lot of patterns which are expected by the data analyst (cf. Section 2.2) require to consider simultaneously and to combine several patterns. In the following, such patterns are called *n-ary patterns*, and a query involving n-ary patterns is called a *n-ary query*.

There are very few attempts on mining n-ary patterns and the existing methods tackle particular cases by using devoted techniques [8,9]. One explanation of the lack of generic methods is likely the difficulty of the task. Mining n-ary patterns requires to compare the solutions satisfying each pattern involved in the constraint, it is drastically harder than mining local patterns. The lack of generic methods 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.

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 and also numeric properties such as the frequency or the length of patterns. Then, 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 set 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.

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* $\boldsymbol{r}$ is a multi-set of itemsets of $\mathcal{L}_{\mathcal{I}}$. Each itemset, usually called a *transaction* or object, is a database entry. 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 (i.e., 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 well-known *frequency* constraint focuses on patterns occurring in the database a number of times exceeding a given minimal threshold: $freq(X) \geq minfr$. There are many other constraints [7] to evaluate the relevance of patterns, like the *area* ($area(X)$ is the product of its frequency times its length: $area(X) = freq(X) \times length(X)$).

2.2 N-ary Patterns

In practice, the data analyst is often interested in discovering richer patterns than local patterns. The definitions relevant to such more complex 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 pattern). A *n-ary pattern* is defined by a query involving several patterns.
Definition 2 (n-ary query). A *n-ary query* is a set of constraints over n-ary patterns.

2.3 Motivating Example

N-ary queries straightforwardly enable us to design rich patterns requested by the users such as the discovery of pairs of exception rules without domain-specific information [9]. 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. 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 surprising 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 $freq(X \cup Y)/freq(X)$). Suzuki proposes a method based on sound pruning and probabilistic estimation [9] to extract the exception rules, but this method is devoted to this kind of patterns.

2.4 Related Work

There are a lot of works to discover local patterns under constraints [7] but there are not so many methods to combine local patterns: pattern teams [6], constraint-based pattern set mining [3] to name a few. 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. Our work is in the new trend on investigations of relationships between data mining and constraint programming [2,4].

3 Modeling and Mining n-ary Queries Using CP

3.1 Examples of n-ary Queries

Exception Rules. (see Section 2.3). Let X and Y be two patterns. Let I and $\neg I \in \mathcal{I}$. 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: $freq((X \setminus Y) \sqcup [1] I) \geq minfr \wedge (freq(X \setminus Y) - freq((X \setminus Y) \sqcup I)) \leq \delta_1$ ($X \backslash Y \rightarrow I$ is a frequent rule having a high confidence value).
- $X \rightarrow \neg I$ is expressed by the conjunction: $freq(X \sqcup \neg I) \leq maxfr \wedge (freq(X) - freq(X \sqcup \neg I)) \leq \delta_2$ ($X \rightarrow \neg I$ is a rare rule having a high confidence value).

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

Unexpected Rules. Another example of n-ary queries is the *unexpected* rule $X \rightarrow Y$ with respect to a belief $U \rightarrow V$ where U and V are patterns [8]. Basically, an unexpected rule means that Y and V logically contradict each other. It is defined more formally as: (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:

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

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

Classification Conflicts. Classification based on associations [11] is an other area where n-ary queries enable us to combine local patterns to help to design classifiers. Let C and C' be the items denoting the class values. The following example detects classification conflicts, here a pair of frequent classification rules $X \rightarrow C$ and $Y \rightarrow C'$ having confidences greater than a minimal threshold $minconf$. The rules have a large overlapping between their premises that may introduce classification conflicts on unseen examples.

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

3.2 Soving n-ary Queries Using CP

After having formulated n-ary queries in a high level modeling as a set of numeric and set constraints as previously seen, these constraints are solved by a CP solver (the Gecode system[2]). Our method has 3 steps. Firstly, the dataset and the patterns involved in the n-ary query are linked. Then, unknown patterns are modeled using variables. Finally, numeric constraints and set constraints are reformulated in a low level way (for more details, see [5]). 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.

4 Experiments

Experiments were performed on several datasets from the UCI repository[3] and a real-world dataset `Meningitis` coming from the Grenoble Central Hospital (329 transactions described by 84 items). 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.

Highlighting Useful Patterns. Exception rules are a particular case of rare rules. Even when rare rules can be extracted [10], it is impossible to pick the exception rules among the set of all the rare rules. It is a pity because most of the rare rules are unreliable and it is much more interesting to get the exceptions rules. Fig. 1 quantifies the number of exception rules on the `Meningitis`

[2] `http://www.gecode.org`
[3] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

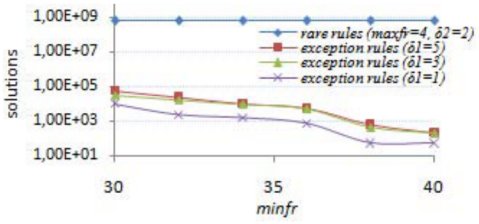

Fig. 1. Number of pairs of exception rules versus number of rare rules (`Meningitis`)

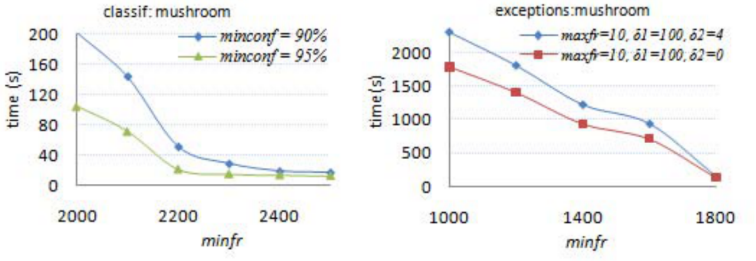

Fig. 2. Runtimes

dataset versus the number of rare rules (the number of rare rules depends on $maxfr$ and corresponds to the line at the top of the figure). Looking for exception rules reduces on several orders of magnitude the number of outputted patterns. Unexpected rules may also reveal useful information. For example, still on `Meningitis`, 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.

Computational Efficiency. These experiments quantify runtimes and the scalability of our approach. Runtimes vary according to the size of the datasets but also the tightness of constraints[4]. On `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 vary from few seconds for tight constraints to about an hour for low frequency and confidence thresholds. These results suggest to conduct further experiments on this dataset to better evaluate the runtimes. Fig. 2 details the runtime of our method on `Mushroom` according to different thresholds of confidence and frequency. We observe that the tighter the

[4] 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

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.

Obviously, our generic n-ary approach can be used for mining local patterns. We obtain on this task the same runtimes as [2] which were competitive with state of the art miners. With exception rules, we cannot compare runtimes because they are not indicated in [9].

5 Conclusion and Future Works

In this paper, we have presented a correct and complete approach to model and mine n-ary patterns. The examples described in Section 3.1 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 queries such as the *peak* query[5].

References

1. 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)
2. De Raedt, L., Guns, T., Nijssen, S.: Constraint Programming for Itemset Mining. In: ACM SIGKDD Int. Conf. KDD 2008, Las Vegas, Nevada, USA (2008)
3. De Raedt, L., Zimmermann, A.: Constraint-based pattern set mining. In: 7th SIAM Int. Conf. on Data Mining. SIAM, Philadelphia (2007)
4. Khiari, M., Boizumault, P., Crémilleux, B.: Local constraint-based mining and set constraint programming for pattern discovery. In: From Local Patterns to Global Models (LeGo 2009), ECML/PKDD 2009 Workshop, Bled, Slovenia, pp. 61–76 (2009)
5. Khiari, M., Boizumault, P., Crémilleux, B.: Constraint programming for mining n-ary patterns. In: Cohen, D. (ed.) CP 2010. LNCS, vol. 6308, pp. 552–567. Springer, Heidelberg (2010)
6. 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)
7. 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)
8. Padmanabhan, B., Tuzhilin, A.: A belief-driven method for discovering unexpected patterns. In: KDD, pp. 94–100 (1998)
9. Suzuki, E.: Undirected Discovery of Interesting Exception Rules. Int. Journal of Pattern Recognition and Artificial Intelligence 16(8), 1065–1086 (2002)
10. Szathmary, L., Valtchev, P., Napoli, A.: Generating Rare Association Rules Using the Minimal Rare Itemsets Family. Int. J. of Software and Informatics 4(3), 219–238 (2010)
11. 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)

[5] The *peak* query compares neighbor patterns; a *peak* pattern is a pattern whose all neighbors have a value for a measure lower than a threshold