Accelerated algorithm for computation of all prime patterns in logical analysis of data

Arthur Chambon, Frédéric Lardeux, Frédéric Saubion, Tristan Boureau

▶ **To cite this version:**

Arthur Chambon, Frédéric Lardeux, Frédéric Saubion, Tristan Boureau. Accelerated algorithm for computation of all prime patterns in logical analysis of data. 8th International, Conference on Pattern Recognition Applications and Methods (ICPRAM), 2019, Prague, Czech Republic. pp.210-220, 10.5220/0007389702100220 . hal-02557441

HAL Id: hal-02557441

https://univ-angers.hal.science/hal-02557441

Submitted on 13 Oct 2021

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.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Accelerated algorithm for computation of all prime patterns in logical analysis of data

Arthur Chambon[1], Frédéric Lardeux[1], Frédéric Saubion[1], Tristan Boureau[2]
[1]*LERIA, Université d'Angers, Angers, France*
[2]*UMR1345 IRHS, Université d'Angers, Angers, France*
{*firstname.lastname*}*@univ-angers.fr*

Keywords: Logical analysis of data, Pattern generation, Logical characterization of data

Abstract: The analysis of groups of binary data can be achieved by logical based approaches. These approaches identify subsets of relevant Boolean variables to characterize observations and may help the user to better understand their properties. In logical analysis of data, given two groups of data, patterns of Boolean values are used to discriminate observations in these groups. In this work, our purpose is to highlight that different techniques may be used to compute these patterns. We present a new approach to compute prime patterns that do not provide redundant information. Experiments are conducted on real biological data.

1 Introduction

Context

Logical analysis of data (LAD), introduced for the first time by Peter Hammer (Hammer, 1986), is based on combinatorial optimization techniques and on the concept of partially defined Boolean functions. It may be considered as an alternative to conventional statistical classification methods. LAD (Crama et al., 1988) can be used in various application domains. One of its purpose is the characterization of data by means of patterns (and subsequently by logical formulas). Given two sets of data (groups), these patterns are indeed subsets of values that are present in several observations of one set, while not being present in the other set. Hence, patterns can be used to identify common characteristics of observations belonging to the same group. The idea is to focus on explicit justifications of groups of data, while classic classification approaches mainly focus on the construction or the identification of these groups. More precisely, the purposes of LAD are:

- to determine similarities within the same data set,
- to discriminate observations belonging to different data sets.
- to deduce logical rules/formulas that explain data sets.

As mentioned above, LAD focuses merely on explanation when classification techniques allow the user to build cluster and to assign groups to incoming data. Many applications of logical data analysis have been investigated in medicine (Reddy et al., 2008), industry (Mortada et al., 2012; Dupuis et al., 2012) or economy (Hammer et al., 2012).

Example

Let us consider two groups (sets) of observations P and N (respectively positive and negative observations) defined over a set $\mathcal{A}$ of Boolean variables. In this example, we consider a set of 8 Boolean variables (labeled from a to h) and 7 observations. Our purpose is to compute a subset of $\mathcal{A}$ that may be used to explain/justify a priori the membership of observations to their respective groups.

As mentioned above, contrary to classification approaches issued from machine learning techniques (e.g., clustering algorithms), the purpose here is to provide an explicit justification of the data instead of an algorithm that assigns groups to data. Note that we assume that the two groups are built by experts, or using expert knowledge (this is thus definitely not a classification nor a clustering problem).

Observ.	Groups	Variables							
		a	b	c	d	e	f	g	h
1	P	0	1	0	1	0	1	1	0
2		1	1	0	1	1	0	0	1
3		0	1	1	0	1	0	0	1
4	N	1	0	1	0	1	0	1	1
5		0	0	0	1	1	1	0	0
6		1	1	0	1	0	1	0	1
7		0	0	1	0	1	0	1	0

In LAD methodology, a key concept consists in identifying patterns of similar values in groups. For instance, $a = 0$ and $b = 1$ is a pattern that is shared by observations 1 and 3 in P and such that no observation in N is covered by this pattern. Therefore, this pattern could be interpreted as a partial explanation of the observations of group P. Among the sets of patterns, one has to decide which compromise has to be achieved between their size and their covering (i.e., the number of observations having the pattern in group P). Concerning the size of the patterns, some properties have been exhibited in order to focus on the most relevant ones. In particular, prime patterns are patterns whose number of variables cannot be reduced unless they are not patterns anymore. Prime patterns correspond to the simplicity requirement (in terms of variables), while strong patterns correspond to an evidential preference where a larger cover is preferred (we refer the reader to (Chikalov et al., 2013) for a survey on LAD).

Alternatively, variables f and g can also be used to generate a Boolean formula $\phi \equiv (f \wedge g) \vee (\neg f \wedge \neg g)$, which is true for observations in P (interpreted as Boolean assignments on variables) and false for observations in N. Note that the variable b is not sufficient to explain group P since observation 6 in N has also this variable set to 1. ϕ is presented here in disjunctive normal form. Note that such formula could be convenient for users, either by minimizing the number of variables (for instance, to simplify their practical implementation in diagnosis routines) or by minimizing the size of the formula (for instance, to improve their readability). Such an approach focuses on minimal characterizations in terms of number of variables and can be extended to consider several groups simultaneously (Chhel et al., 2012). In our work, it is important to consider prime patterns (without redundant variable) in order to minimize the number of variables, and to consider as many patterns as possible in order to have a broader view to minimize the size of the formula.

In our example, we consider only two possible states for each variables, so we work on binary data. However, in practice, this is not always the case. In (Boros et al., 1997), "binarization" of data is introduced, allowing to transform quantitative data into binary data. The basic idea of binarization is simple: each real-valued data is associated to a threshold. The binary value is 1 (respectively 0) if the real data is above this threshold (respectively below). Since a single threshold is too restrictive, it is important to study several thresholds combinations. One of the problems associated with binarization is therefore to find a minimal number of thresholds that would preserve most of the information contained in a data set (Hammer and Bonates, 2006; Boros et al., 1997).

Contributions
In this paper, we present a new algorithm that computes the set of all prime patterns. Other algorithms can be used to compute prime patterns, in particular the algorithm presented in (Boros et al., 2000). However our algorithm, based on the detection of solutions in logical characterization of data, is faster and works on a more important set of variables. Moreover our algorithm allows us to determine the coverage of each patterns.

Organization
In Section 2 we recall the concepts of the logical analysis of data and the computation of prime patterns. In Section 3 we recall the main concepts of logical characterization of data. In Section 4, we present our new algorithm. Finally, in Section 5 we compare the performances of our algorithm with the algorithm of (Boros et al., 2000) on different real instances (issued from biology) and handmade instances.

2 Logical Analysis of Data

2.1 Terminology and notation

Let us recall the main concepts of logical analysis of data (LAD) (Hammer et al., 2004; Hammer and Bonates, 2006; Boros et al., 2011; Chikalov et al., 2013). LAD is based on the notion of partially defined Boolean functions.

Definition 1. *A Boolean function f of n variables, $n \in \mathbb{N}$, is a function $f : \mathbb{B}^n \to \mathbb{B}$, where $\mathbb{B}$ is the set $\{0,1\}$.*

Definition 2. *A vector $x \in \mathbb{B}^n$ is a positive vector (resp. negative vector) of the Boolean function f if $f(x) = 1$ (resp $f(x) = 0$). $T(f)$ (resp. $F(f)$) is the set of positive vectors (resp. negative vectors) of the Boolean function f.*

In the rest of the paper, Boolean vectors correspond to observations. The set of observations is denoted Ω.

Definition 3. *A partially defined Boolean function (pdBf) on $\mathbb{B}^n$ is a pair (P,N) such that P, $N \subseteq \mathbb{B}^n$ and $P \cap N = \emptyset$.*

In a pdBf, we consider two groups of observations: P (positive group) and N (negative group). A *literal* is either a binary variable x_i or its negation $\bar{x}_i$. A term is a pdBf represented by a conjunction of distinct literals, such that a term does not contain a variable and its negation.

Definition 4. *Given: $\sigma^+, \sigma^- \subseteq \{1,2,...,n\}$, $\sigma^+ \cap \sigma^- = \emptyset$, a term t_{σ^+,σ^-} is a Boolean function whose positive set $T(t_{\sigma^+,\sigma^-})$ is of the form:*

$$T(t_{\sigma^+,\sigma^-}) = \{x \in \mathbb{B}^n | x_i = 1 \ \forall i \in \sigma^+ \text{ and } x_j = 0 \ \forall j \in \sigma^-\}$$

A term t_{σ^+,σ^-} can be represented by an *elementary conjunction*, i.e., a Boolean expression of the form:

$$t_{\sigma^+,\sigma^-}(x) = (\bigwedge_{i \in \sigma^+} x_i) \wedge (\bigwedge_{j \in \sigma^-} \bar{x}_j)$$

We say that a Boolean vector satisfies a term if it has the same binary values as the term on the variables of the term. The set of literals of a term t is $Lit(t)$. The *degree* of a term t is the number of literals that appear in this term, ie $|Lit(t)|$.

Definition 5. *A pattern of a pdBf (P,N) is a term t_{σ^+,σ^-} such that $|P \cap T(t_{\sigma^+,\sigma^-})| > 0$ and $|N \cap T(t_{\sigma^+,\sigma^-})| = 0$.*

A pattern is therefore a term satisfied by at least one positive vector (from P) and no negative vector (from N). Of course, patterns are associated to the positive group P. It is also possible to design patterns associated to the negative group by considering the pdBf (N, P) instead of the pdBf (P, N).

Definition 6. *The coverage of a pattern p, denoted $Cov(p)$, is the set $Cov(p) = P \cap T(p)$.*

In other words, the coverage of a pattern p is the set of positive Boolean vectors satisfying p.

Example 1. *Back to introductory example:*

Obs	*Groups*	*Variables*							
		a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
1	*P*	*0*	*1*	*0*	*1*	*0*	*1*	*1*	*0*
2		*1*	*1*	*0*	*1*	*1*	*0*	*0*	*1*
3		*0*	*1*	*1*	*0*	*1*	*0*	*0*	*1*
4	*N*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*1*
5		*0*	*0*	*0*	*1*	*1*	*1*	*0*	*0*
6		*1*	*1*	*0*	*1*	*0*	*1*	*0*	*1*
7		*0*	*0*	*1*	*0*	*1*	*0*	*1*	*0*

$p_1 = \bar{a} \wedge b$ and $p_2 = \bar{f} \wedge \bar{g}$ are two patterns covering observations 1 and 3 for p_1 and 2 and 3 for p_2.

There exist many possible patterns. Using the two sets $Lit(p)$ and $Cov(p)$, we can define 2 types of partial preorders on patterns: *the simplicity preference* and *the evidential preference*.

Definition 7. *The simplicity preference σ, denoted $\succcurlyeq_\sigma$, is a binary relation over a set of patterns $\mathcal{P}$ such that for a couple $(p_1, p_2) \in \mathcal{P}^2$, we have $p_1 \succcurlyeq_\sigma p_2$ if and only if $Lit(p_1) \subseteq Lit(p_2)$.*

Definition 8. *The evidential preference $\mathcal{E}$, denoted $\succcurlyeq_\mathcal{E}$, is a binary relation over a set of patterns $\mathcal{P}$ such that for a couple $(p_1, p_2) \in \mathcal{P}^2$, we have $p_1 \succcurlyeq_\mathcal{E} p_2$ if and only if $Cov(p_2) \subseteq Cov(p_1)$.*

For a preference $\succcurlyeq_{pi}$ ($\pi \in \{\sigma, \mathcal{E}\}$) and two patterns p_1 and p_2, we will note the double relation $p_1 \succcurlyeq_\pi p_2$ and $p_2 \succcurlyeq_\pi p_1$ by $p_1 \approx_\pi p_2$.

In order to refine the comparison of patterns, we will consider combinations of preferences.

Definition 9. *Let two preferences π and ρ on a set of patterns $\mathcal{P}$, and let $(p_1, p_2) \in \mathcal{P}^2$, the pattern p_1 is preferred to the pattern p_2 with respect to the lexicographic refinement $\pi|\rho$, denoted $p_1 \succcurlyeq_{\pi|\rho} p_2$, if and only if $p_1 \succcurlyeq_\pi p_2$ or $p_1 \approx_\pi p_2$ and $p_1 \succcurlyeq_\rho p_2$.*

Definition 10. *Given a preference $\succcurlyeq_\pi$ on a set of patterns $\mathcal{P}$, a pattern $p_1 \in \mathcal{P}$ is Pareto-optimal with respect to π if and only if $\nexists p_2 \in \mathcal{P} \backslash \{p_1\}$ such that $p_2 \succ_\pi p_1$ (i.e $p_2 \succcurlyeq_\pi p_1$ and $p_2 \not\approx_\pi p_1$).*

We can thus define types of Pareto-optimal patterns, according to the preferences:

Preference	Pareto-optimal pattern
σ	Prime pattern
$\mathcal{E}$	Strong pattern
$\mathcal{E}\|\sigma$	Strong prime pattern

Note the following property demonstrated in (Hammer et al., 2004).

Property 1. *A pattern is pareto-optimal with respect to $\mathcal{E}|\sigma$ if and only if it is strong and prime.*

Example 2. *Let's take Example 1 to illustrate the different types of patterns:*

- *The pattern $a \wedge d \wedge e$ is a prime pattern because if you remove a literal, it is no longer a pattern.*
- *The $e \wedge \bar{f} \wedge \bar{g}$ pattern is a strong pattern because there is no pattern covering the same observations plus one.*
- *The $\bar{f} \wedge \bar{g}$ pattern is both strong and prime. It is therefore a strong prime pattern.*

In (Boros et al., 2000), the notion of *support sets* is used. A support set is a subset of variables such that, by working only on the selected variables, all positive Boolean vectors are different from negative Boolean vectors. In other words, we are looking for a set of variables that discriminate the whole group. A set support is said to be *irredundant* if the removal of any variable gives a set of variables that is no longer a set support.

2.2 Patterns Generation

The problem of generating optimal patterns can be solved by a linear program. In (Ryoo and Jang, 2009) integer linear programs are proposed, allowing to generate different types of patterns, in particular strong prime patterns. However, if the authors demonstrate that an optimal solution of their linear program is a Pareto-optimal pattern, the converse is not true. It is therefore not possible to generate the set of all prime patterns (or set of all strong prime patterns) by the linear program presented in this article.

If the linear programming approach does not allow us to generate all the optimal patterns, the algorithm proposed in (Boros et al., 2000) will generate the set of all prime patterns. Note also that in (Hammer et al., 2004) are described algorithms transforming a pattern into a prime pattern or strong pattern.

2.2.1 Prime patterns generation

Concerning the generation of prime patterns, the algorithm in (Boros et al., 2000) proposes the generation of all prime patterns less than or equal to a degree D. By choosing the highest degree D_{max} of prime patterns, we can generate all of them, but we do not know how to compute this degree without generating the set of patterns. We choose an upper bound to D_{max} to be certain of our results. By choosing $D = n$,i.e. the number of variables, we insure that $D \geqslant D_{max}$ and thus, the algorithm will return the set of prime patterns.

Boros's algorithm first generates prime patterns of degree 1, then 2 and so on up to degree D. In step 1, the algorithm tests all the terms of degree 1 and classifies them according to their interest. All terms that are patterns will be in a set P, and in a set C_1 the terms of size 1 covering positive and negative observations. In the following steps (E) we will only consider the terms of the set C_{E-1}, terms of degree $E-1$, for which we try to add a literal to transform it into a term of degree E. If this term is a pattern, it joins the set P and if it still covers both positive and negative observations, it joins the set C_E. Once the step $E = D$ has been completed, we will have the set P which will contain all prime patterns of degree less than or equal to D.

Algorithm 1 presents the pseudo-code of this algorithm.

Since we test all the combinations, we are certain to generate the set of prime patterns. In addition, to be generated, a term must be in the set P and thus check the conditions to be a pattern. As before being in the set P, the term was in a set C_i, if we remove a literal, the term is no longer a pattern. The set P therefore contains all prime patterns and contains only prime patterns.

ALGORITHM 1. Computation of all prime patterns from (Boros et al., 2000) (`PPC_1`)

Data: D the maximum degree of patterns that will be generated.
Result: P_D the set of prime patterns smaller than or equal to D.

$P_0 = \emptyset$
//C_i is the set of terms of degree i that can become patterns, which cover both positive and negative observations.
$C_0 = \{\emptyset\}$
for $i = 1$ *to* D **do**
  $P_i = P_{i-1}$
  $C_i = \emptyset$
  forall $t \in C_{i-1}$ **do**
    p=maximum index of variables in t
    for $s = p+1$ *to* n **do**
      forall $l \in \{x_s, \bar{x}_s\}$ **do**
        $T = t \wedge l$
        for $j = 1$ *to* $i-1$ **do**
          $t' = T$ with the j-th variable removed
          if $t' \notin C_{i-1}$ **then**
            go to $\Diamond$
          end
        end
        if T *covers a positive vector but no negative vector* **then**
          $P_i = P_i \cup \{T\}$
        end
        if T *covers both a positive and a negative vector* **then**
          $C_i = C_i \cup \{T\}$
        end
        $\Diamond$
      end
    end
  end
end
return P_D

Note that, for an instance with x observations and n variables, the complexity is :

$$O(3^n \times n \times (2^{n-\lfloor \frac{n}{3} \rfloor} \times \frac{n!}{\lfloor \frac{n}{3} \rfloor !(n-\lfloor \frac{n}{3} \rfloor)!} + x))$$

(3^n is the number of terms that can be created with n variables and $2^{n-\lfloor \frac{n}{3} \rfloor} \times \frac{n!}{\lfloor \frac{n}{3} \rfloor !(n-\lfloor \frac{n}{3} \rfloor)!}$ is the maximal number of prime patterns).

3 Multiple Characterization of Data

Multiple characterization of data is an extension of logical analysis of data where several groups of data are considered. The goal is to compute a set of variables, called solution, sufficient to discriminate each group of data from the others, simultaneously.

3.1 Presentation

As in LAD, multiple characterization of data (Chhel et al., 2012) aims to discriminate observations from different groups. The objective is to determine a set of variables discriminating all groups simultaneously. This approach is similar to support sets computation but the number of groups may be larger than 2.

Example 3. *Let us consider again introductory example with more than two groups:*

Obs	*Groups*	*Variables*							
		a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
1	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*1*	*0*
2		*1*	*1*	*0*	*1*	*1*	*0*	*0*	*1*
3		*0*	*1*	*1*	*0*	*1*	*0*	*0*	*1*
4	*2*	*1*	*0*	*1*	*0*	*1*	*0*	*1*	*1*
5		*0*	*0*	*0*	*1*	*1*	*1*	*0*	*0*
6	*3*	*1*	*1*	*0*	*1*	*0*	*1*	*0*	*1*
7		*0*	*0*	*1*	*0*	*1*	*0*	*1*	*0*

The variables a and b are not sufficient to discriminate groups 2 and 3 because the observation 2 is similar to the observation 6 on these variables. However, the variables a, b and f discriminate the 3 groups at the same time because no observations are identical on these 3 variables.

It is important to note that multiple characterization of data is different from feature selection since it does not focus on the most statistically informative variables, but rather on a combination of variables that exactly discriminates the groups.

3.2 Terminology and Notations

We use here the notations and formalization proposed in (Chambon et al., 2015). The observations belonging to Ω are expressed on Boolean variables belonging to the set $\mathcal{A}$. These observations are divided into several groups belonging to the set of groups $\mathcal{G}$. These data are represented by a matrix D =.

Definition 11. *An instance of the Multiple Characterization Problem (MCP) is a quadruplet $(\Omega, \mathcal{A}, D, G)$ defined by a set of Ω observations whose elements are expressed on a set of variables A , and are represented by a matrix of Boolean data $D_{|\Omega|\times|\mathcal{A}|}$ and a function $G : \Omega \to \mathcal{G}$, such that $G(o)$ is the group to which the observation $o \in \Omega$ belongs.*

The data matrix is defined as follows:

- The value $D[o,a]$ represents the presence/absence of the variable a for the observation o.
- A line $D[o,.]$ represents the Boolean vector of presence/absence of the different variables for the observation o.
- A column $D[.,a]$ represents the Boolean vector of presence/absence of the variable a in all observation.

Thus, two observations $o, o' \in \Omega$ can be represented by the same Boolean vector (so $D[o,.] = D[o',.]$) and yet be considered as two distinct observations.

In the following we are only interested in satisfiable MCP, i.e. such that D does not contain two identical observations in two different groups (generalization of the notion of support set to multiple groups).

Property 2. *A MCP instance $(\Omega, \mathcal{A}, D, G)$ is satisfiable iff: $\nexists (o,o') \in \Omega^2$ such that $D[o,.] = D[o',.]$ and $G(o) \neq G(o')$*

Definition 12. *Let $A \subset \mathcal{A}$, D^A is the data matrix reduced to the subset of variables A.*

We now introduce the concept of MCP solution.

Definition 13. *Given an instance $(\Omega, \mathcal{A}, D, G)$, a subset of variables $S \subseteq \mathcal{A}$ is a solution if and only if $\forall (o,o') \in \Omega^2, G(o) \neq G(o') \Rightarrow D^S[o,.] \neq D^S[o',.]$. In this case, the matrix D^S is called a solution matrix.*

In other words, S is a solution if two observations, coming from two different groups, are different on at least one variable $a \in S$.
Note that in the particular case where we only have two groups, a MCP solution is a support set (see Section 2.1).

An instance of the MCP may have several solutions of different sizes. It is therefore important to define an ordering on solutions in order to compare and classify them. In particular, for a given solution S, adding a variable generates a new solution $S' \supset S$. In this case we say that S' is dominated by S.

Definition 14. *A solution S is non-dominated iff $\forall s \in S$, $\exists (o,o') \in \Omega^2$ such that $G(o) \neq G(o')$ and $D^{S\setminus\{s\}}[o,.] = D^{S\setminus\{s\}}[o',.]$ (i.e. $S\setminus\{s\}$ is not a solution).*

The search for non-dominated solutions thus makes it possible to avoid searching for redundant information while limiting the number of solutions.

Among these solutions, we are interested in computing solutions of minimal size with regards to their variables.

Definition 15. *A solution S is minimal iff $\nexists S'$ with $|S'| < |S|$ s.t. S' is a solution.*

According to our notion of dominance between solutions, a minimal solution in not dominated by any other solutions. Intuitively, a minimal (non dominated) solution cannot be reduced unless two identical lines appear in two different groups (and consequently the reduced set of variables is not a solution).

As already mentioned, a solution of the MCP is a generalization of the notion of support set. A non-dominated solution is thus a generalization of an irredundant support set. Note that the union of irredundant support sets for each couple of groups is not necessary a non-dominated solution.

3.3 Converting Characterization Requirements into Constraints

The minimum multiple characterization problem can be formulated as a linear program (Chhel et al., 2013; Boros et al., 2000). In fact, finding solutions correspond here to a set covering problem. Given an instance $(\Omega, \mathcal{A}, D, G)$, let us consider the following $0/1$ linear program.

$$
\begin{aligned}
& min : \sum_{i=1}^{|\mathcal{A}|} y_i \\
& s.t. : \\
& C \,.\, Y^t \geqslant \mathbb{1}^t \\
& Y \in \{0,1\}^{|\mathcal{A}|}, Y = [y_1, ..., y_{|\mathcal{A}|}]
\end{aligned}
$$

where Y is a Boolean vector that encodes the presence/absence of the set of variables in the solution. C is a matrix that defines the constraints that must be satisfied in order to insure that Y is a solution. Let us denote Θ the set of all pairs $(o,o') \in \Omega^2$ such that $G(o) \neq G(o')$. For each pair of observations (o,o') that do not belong to the same group, defined by an element of Θ, one must insure that the value of at least one variable differ from o to o'. This will be insured by the inequality constraint involving the $\mathbb{1}$ vector (here a vector of dimension $|\Theta|$ that contains only 1 values).

More formally, C is a Boolean matrix of size $|\Theta| \times |\mathcal{A}|$ define as:

- Each line is numbered by a couple of observations $(o,o') \in \Omega^2$ such that $G(o) \neq G(o')$ ($(o,o') \in \Theta$).
- Each column represents indeed a variable.
- $C[(o,o'),a] = 1$ if $D[o,c] \neq D[o',c]$, $C[(o,o'),a] = 0$ otherwise.
- We denote $C[(o,o'),.]$ the Boolean vector representing the differences between observations o and o' on each variable. This Boolean vector is called constraint since one variable a such $C[(o,o'),a] = 1$ must be selected in order to insure that no identical observations can be found in different groups .

3.4 Computation of all Non-dominated Solutions

In (Chambon et al., 2015), Algorithm 2 is presented in order to compute the set of all non-dominated solutions according to Definition 14.

The idea is thus to select variables a such that there exists a couple of observations $(o,o') \in \Theta$[1] satisfying $C[(o,o'),a] = 1$ in the constraint matrix C and $C[(o,o'),a'] = 0$ for any variable $a' \neq a$.

ALGORITHM 2. Non-Dominated Solutions: NDS

```
Data: C: Constraints matrix of size |Θ| × |A|.
Result: Sol: Non-dominated solutions set.
Sol = {∅}
for i = 1 to |Θ| do
    //Build a subset of solutions ND_i
    ND_i = ∅
    forall j ∈ A s.t. C[θ_i, j] = 1 do
        forall S ∈ Sol\ND_i do
            if j ∈ S then
                ND_i = ND_i ∪ {S}
            end
        end
    end
    //Build a subset of solutions ES_i
    ES_i = ∅
    forall j ∈ A s.t. C[θ_i, j] = 1 do
        forall S ∈ Sol\ND_i do
            if ∄S' ∈ ND_i s.t. S' ⊆ S ∪ {j} then
                ES_i = ES_i ∪ {S ∪ {j}}
            end
        end
    end
    Sol = ND_i ∪ ES_i
end
return Sol;
```

Algorithm 2 builds incrementally the set Sol of non-dominated solutions. Each element of $\Theta = \{\theta_1, \theta_2, ..., \theta_{|\Theta|}\}$ is a constraint that must be satisfied. At each iteration, the solutions are updated in order to satisfy the constraint θ_i. The main idea consists in

[1] Remind that Θ is a set of couples of observations defined in 3.3 for indexing lines of the constraint matrix C.

distinguishing solutions that already satisfy this constraint (they are put in a set of non dominated solutions ND_i) and solutions that need to be modified in order to satisfy the constraint, (they belong to the set ES_i). Note that the modification of these latest solutions is performed by adding one variable while maintaining the non-domination property.

This algorithm is related to the Berge's algorithm (Berge, 1984) that may be used for computing hitting sets. Berge's algorithm consists in incrementing the solution at each iteration with an element of the constraint, and to compare pairs of solutions to remove dominated solution. Algorithm 2 avoid constructing dominated solutions.

Note that, as the algorithm builds incrementally the set of non-dominated solutions. We can use a maximal bound B for computing only non-dominated solutions with a number of variables smaller than this bound. Given a bound B, Algorithm 2 can be modified when updating ES_i as $ES_i = ES_i \cup \{S \cup \{j\}\}$. This update can be performed only if $\forall s \in ES_i$, $|s| \leq B$ in order to improve the performance of the algorithm.

4 Computation of Prime Patterns and Group Covers

We propose now a new algorithm that uses the computation of all non dominated solutions of the MCP problem in order to compute prime patterns.

In LAD, the aim is to find a pattern that covers a maximum number of observations of P, such as no observation of N contains this pattern. From MCP point of view, the notion of solution is rather different. Given a solution S of a MCP instance $(\Omega, \mathcal{A}, D, G)$ defined as above, the variables of S do not generally correspond to a pattern for the observations in P, unless all observations are identical on S. In this case a solution of the MCP obviously coincides with a prime pattern in terms of variables.

In particular, if $|P| = 1$, the set of all solutions of the MCP coincides in terms of variables with the set of all prime patterns that cover the only observation in P, because in both cases no variable can be removed.

Given a non-dominated solution S of the MCP (computed by previously mentioned algorithms for instance), it is easy to transform an observation o of the group P into prime pattern p. Each variable a of S appears positively (resp. negatively) in p if $D[o,a] = 1$ (resp. $D[o,a] = 0$).

The following simple procedure (algorithm 3) transforms a non-dominated solution of MCP, considering only one observation $x \in P$ into a prime pattern.

ALGORITHM 3. Pattern_Transformation

Data: s: non-dominated solution of the MCP;
x: the only observation in the group P.
Result: p: prime pattern.
$pos = \emptyset$
$neg = \emptyset$
forall $a \in s$ **do**
  if $x_a = 1$ **then**
    $pos = pos \cup \{a\}$
  end
  else
    $neg = neg \cup \{a\}$
  end
end
$p = (\bigwedge_{i \in pos} x_i) \wedge (\bigwedge_{j \in neg} \neg x_j)$
return p;

For each observation, we can generate all prime patterns that cover this observation. If we generate all prime patterns for all observations, we generate prime patterns p, and determine $Cov(p)$ for each one.

Algorithm 4 returns the set Pat of all prime patterns, and the set Cov of coverage of all patterns $p \in Pat$. Cov is a set of elements V_p, $\forall p \in Pat$. Each element V_p is a set of all observations covered by p. Note that it is not necessary to compute the set Cov to generate the set Pat. Hence, each step that involves the set Cov can be removed.

All algorithms that can compute the set of all non-dominated solutions (like algorithm NDS) can be used to determine the set Sol. Note that since we are working on one group against all the others, we can also use an algorithm that computes all the irredundant support sets. However, the algorithms presented in (Boros et al., 2000) only allow to compute a subset of these irredundant support sets.

Note that Algorithm NDS (Chambon et al., 2015) can also generate solutions of smaller size than a given bound B. Given a bound B, we can only generate prime patterns with a size inferior to B.

Also note, if we use the algorithm NDS for computing the set Sol, for an instance with x observations and n variables the complexity is :

$$O(x^2 \times 2^{n-\lfloor \frac{n}{3} \rfloor} \times \frac{n!}{\lfloor \frac{n}{3} \rfloor !(n-\lfloor \frac{n}{3} \rfloor)!} \times \frac{n!}{\lfloor \frac{n}{2} \rfloor !(n-\lfloor \frac{n}{2} \rfloor)!})$$

($\frac{n!}{\lfloor \frac{n}{2} \rfloor !(n-\lfloor \frac{n}{2} \rfloor)!}$ is the maximal number of non-dominated solutions)

Now, using the set Cov we can run Algorithm 5 to compute only strong prime patterns. From the set of all covers, we can compute the subset of strong patterns among prime patterns.

ALGORITHM 4. Prime Patterns Computation (PPC_2)

Data: D: matrix of data, with two groups {P,N}.
Result: *Pat*: set of all prime patterns
Result: *Cov*: set of covers of each prime pattern.
$Pat = \emptyset$
$Cov = \emptyset$
forall $o \in P$ **do**
  Generate the constraint matrix C_o as if o was the only one observation in P
  Sol={set of all non dominated solutions for C_o}
  forall $s \in Sol$ **do**
    p=Transformation_Pattern(s,o)
    if $p \notin Pat$ **then**
      $Pat = Pat \cup \{p\}$
      //Create a new element V_p of *Cov* which will be a set of observations covered by p.
      $V_p = \{o\}$
      $Cov = Cov \cup \{V_p\}$
    end
    else
      //V_p is already in *Cov*; update
      $V_p = V_p \cup \{o\}$
    end
  end
end
return *Pat* and *Cov*;

ALGORITHM 5. Strong Prime Patterns Computation

Data: *Cov*: set of coverage of each prime pattern.
Pat: set of all prime patterns
Result: *SPP*: set of all strong prime patterns.
$SPP = \emptyset$
forall $p \in Pat$ **do**
  if $\nexists p' \in Pat$ *s.t.* $Cov(p) \subset Cov(p')$ **then**
    $SPP = SPP \cup \{p\}$
  end
end
return SPP;

5 Experiments

The main purpose of our experiments is to compare the performance of our new algorithm PPC_2 to the algorithm PPC_1 for computing sets of prime patterns.

PPC_2 (Algorithm 4) uses the principles presented in Section 4, encoded in C++ with data structures and operators from the library *boost*[2].

PPC_1 has been recalled in Section 2.2.1. Note that the source code of this algorithm presented in (Chikalov et al., 2013) was not available. It has been implemented in C++ using the same data structures and operators from the library *boost*

Experiments have been run on a computer with Intel Core i7-4910MQ CPU (8×2.90 GHz), 31.3 GB RAM.

5.1 Data Instances

We consider several sets of observations issued from different case studies.

- Random is a random instance built with only one observation in the positive group, and random value in $\{0,1\}$. This instance is used as a basic test case. The group P is restricted to only 1 value since otherwise it would have been difficult to identify common patterns for several randomly generated observations.
- Instances ra100_phv, ra100_phy, ralsto, ra_phv, ra_phy, ra_rep1, ra_rep2 and rch8 are matrices built from biological data that correspond to bacterial strains. Each observation is a bacterial strain and variables are genes (housekeeping gene, resistance gene or specific effectors). These bacteria are responsible of serious plant diseases. Therefore it is important to be able to identify precisely different groups of bacteria using a restricted set of variables and to identify common gene profiles. Such identification are very helpful for building simple and cheap diagnosis routines (Boureau et al., 2013). The original files are available[3]. Initially, several groups are considered in these instances. Therefore, we have considered the first group of bacteria as the positive group and the union of the other groups as the negative group. Note that similar results have been obtained when considering others groups as positive group.
- Instances vote_r[4] are also binary data used as benchmarks for classification purpose. Note that these instances have missing data and have been completed randomly.
- Instances cr60, os1 and rel1 are datasets corresponding to patients suffering from leukemia. Ob-

[2] http://www.boost.org/doc/libs/1_36_0/libs/dynamic_bitset/dynamic_bitset.html

[3] http://www.info.univ-angers.fr/~gh/Idas/Ccd/ce_f.php

[4] http://tunedit.org/repo/UCI/vote.arff

servations correspond to specific mutated variants of genes that are suspected to play a role in the disease. Here the goal is to find genes that could help to improve prognosis and to select the most suitable treatments according to the patients profiles.

The instances are described in Table 1 with their number of observations, number of observations in the positive group (the negative group is of course the complement) and the maximal number of variables. For each instance, we consider consider different values x of variables in order to evaluate the performance of the algorithms with regards to this number of variables.

Instances	Obs	Positive group size	Var
Random	20	1	35
ra100_phv	100	21	50
ra100_phy	105	31	51
ralsto	73	27	23
ra_phv	108	22	70
ra_phy	112	31	73
ra_rep1	112	38	155
ra_rep2	112	37	73
rch8	132	5	37
vote_r	435	168	16
cr60	289	58	14
os1	289	224	14
rel1	259	200	14

Table 1: Characteristics of the instances

5.2 Results

Table 2 provides the results obtained on the instances for computing the set of prime patterns. The first column corresponds to the name of the instance, with the number x of used variables (remind that we consider different sizes for each the instances). The second column corresponds to the number of prime patterns for each instance. Next two columns are execution time (in seconds) for our algorithm `PPC_2` (Algorithm 4), and execution time (in seconds) for `PPC_1` (Algorithm 1). The last two columns correspond to the maximal size of the computed patterns (in terms of number of variables) and the execution time of `PPC_1` used with an initial bound equal to this maximal size. Of course, when using this bound we get the same results but `PPC_1` is faster since it may stop earlier. Note that in practice the value of the bound is not known until the set of prime patterns has been computed. Running time is limited to 24 hours. "-" corresponds to execution times greater than this limit.
The execution time of `PPC_1` increases as the number of observations increases and especially as the number of variables increases. `PPC_2` is less sensitive to the number of observations. Nevertheless its computation time also increases according to the number of variables.

Let us remark that `PPC_1` is able to compute all prime patterns for instance Random(35) in three days. Nevertheless, one week is not enough for the instance rch8(37). `PPC_2` is able to compute all prime patterns for instance ra_rep2(65) in a bit more than one hour while `PPC_1` is not able to solve the same instance with only 20 variables.

In `PPC_1`, the number of iterations is related to the number of variables. We observe in the last column that if a good bound is available (equal to the size of the largest prime pattern), prime patterns can be computed more efficiently. Nevertheless, execution time is still high compared to `PPC_2`. Moreover, it requires to know the size of the largest pattern.

6 Conclusion

In this paper we have defined a new algorithm to generate complete sets of prime patterns and strong prime patterns in LAD context. Compared to the state of the art algorithms for these problems, our algorithm is now able to handle larger data sets. The main idea of its resolution process is to use an extension of the LAD (multiple characterization of data) in order to first compute the non dominated solutions and finally obtain all the prime and strong prime patterns. Experiments show the efficiency of our algorithm in term of running times and instance sizes.

REFERENCES

Berge, C. (1984). *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier.

Boros, E., Crama, Y., Hammer, P. L., Ibaraki, T., Kogan, A., and Makino, K. (2011). Logical analysis of data: classification with justification. *Annals OR*, 188(1):33–61.

Boros, E., Hammer, P. L., Ibaraki, T., and Kogan, A. (1997). Logical analysis of numerical data. *Mathematical Programming*, 79(1-3):163–190.

Boros, E., Hammer, P. L., Ibaraki, T., Kogan, A., Mayoraz, E., and Muchnik, I. (2000). An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):292–306.

Boureau, T., Kerkoud, M., Chhel, F., Hunault, G., Darrasse, A., Brin, C., Durand, K., Hajri, A.,

Instance	Number of prime patterns	Time PPC_2	Time PPC_1	Maximal size	PPC_1 time using max size
Random(11)	61	0.003	0.006	4	0.003
Random(15)	170	0.011	0.029	5	0.021
Random(20)	476	0.020	2.306	6	1.185
Random(25)	1143	0.028	113.469	6	16.343
Random(27)	1529	0.052	338.708	6	38.045
Random(30)	2382	0.110	4154.002	6	170.906
Random(35)	4505	0.283	-	6	1668.373
ra100_phv(10)	6	0.001	0.053	3	0.001
ra100_phv(20)	105	0.013	16478.822	5	11.269
ra100_phv(30)	304	0.025	-	6	40646.487
ra100_phv(40)	1227	0.089	-	8	-
ra100_phv(50)	12162	1.381	-	10	-
ra100_phy(10)	5	0.001	0.039	4	0.014
ra100_phy(20)	131	0.015	-	6	555.829
ra100_phy(30)	583	0.037	-	6	85608.555
ra100_phy(40)	1982	0.123	-	8	-
ra100_phy(51)	13112	2.119	-	10	-
ralsto(10)	22	0.003	0.047	5	0.029
ralsto(15)	132	0.008	35.286	6	6.839
ralsto(20)	361	0.013	39725.665	6	436.524
ralsto(23)	1073	0.040	-	8	-
ra_phv(10)	6	0.004	0.042	3	0.004
ra_phv(25)	194	0.024	-	5	94.069
ra_phv(40)	1227	0.101	-	8	-
ra_phv(55)	28163	6.061	-	10	-
ra_phv(70)	384629	1777.386	-	14	-
ra_phy(10)	5	0.002	0.085	4	0.016
ra_phy(25)	252	0.016	-	6	8253.219
ra_phy(40)	1982	0.121	-	8	-
ra_phy(55)	23504	4.914	-	10	-
ra_phy(73)	449220	2328.949	-	14	-
ra_rep1(10)	4	0.000	0.107	4	0.013
ra_rep1(30)	11	0.000	-	4	6.385
ra_rep1(60)	415	0.021	-	7	-
ra_rep1(90)	9993	1.420	-	10	-
ra_rep1(120)	243156	1081.313	-	12	-
ra_rep1(155)	2762593	-	-	15	-
ra_rep2(10)	11	0.002	0.096	4	0.008
ra_rep2(15)	46	0.005	133.096	4	0.158
ra_rep2(20)	126	0.007	-	6	413.723
ra_rep2(25)	303	0.009	-	7	-
ra_rep2(30)	745	0.017	-	7	-
ra_rep2(35)	2309	0.060	-	9	-
ra_rep2(40)	6461	0.403	-	10	-
ra_rep2(45)	17048	2.315	-	10	-
ra_rep2(50)	43762	14.596	-	10	-
ra_rep2(55)	101026	68.378	-	11	-
ra_rep2(60)	254042	840.734	-	12	-
ra_rep2(65)	720753	4617.556	-	14	-
ra_rep2(73)	2474630	60740.333	-	15	-
rch8(15)	1	0.004	5.988	2	0.004
rch8(20)	7	0.009	4261.190	4	0.163
rch8(25)	26	0.012	-	4	0.676
rch8(30)	43	0.016	-	4	2.384
rch8(37)	131	0.021	-	6	77836.071
vote_r(10)	169	0.060	0.876	6	0.669
vote_r(13)	1047	0.250	52.711	8	48.427
vote_r(16)	4454	0.842	4138.240	9	3466.591
cr60(10)	55	0.018	0.923	6	0.585
cr60(14)	196	0.026	350.849	8	219.009
os1(10)	107	0.018	0.958	6	0.571
os1(14)	286	0.035	379.423	7	102.318
rel1(10)	135	0.024	0.976	6	0.604
rel1(14)	388	0.033	376.540	7	107.410

Table 2: Results for prime patterns computation.

Poussier, S., Manceau, C., et al. (2013). A multiplex-pcr assay for identification of the quarantine plant pathogen xanthomonas axonopodis pv. phaseoli. *Journal of microbiological methods*, 92(1):42–50.

Chambon, A., Boureau, T., Lardeux, F., Saubion, F., and Le Saux, M. (2015). Characterization of multiple groups of data. In *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*, pages 1021–1028. IEEE.

Chhel, F., Lardeux, F., Goëffon, A., and Saubion, F. (2012). Minimum multiple characterization of biological data using partially defined boolean formulas. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1399–1405. ACM.

Chhel, F., Lardeux, F., Saubion, F., and Zanuttini, B. (2013). Application du problème de caractérisation multiple à la conception de tests de diagnostic pour la biologie végétale. *Revue des Sciences et Technologies de l'Information-Série RIA: Revue d'Intelligence Artificielle*, pages 649–668.

Chikalov, I., Lozin, V., Lozina, I., Moshkov, M., Nguyen, H. S., Skowron, A., and Zielosko, B. (2013). Logical analysis of data: theory, methodology and applications. In *Three Approaches to Data Analysis*, pages 147–192. Springer.

Crama, Y., Hammer, P. L., and Ibaraki, T. (1988). Cause-effect relationships and partially defined boolean functions. *Annals of Operations Research*, 16(1):299–325.

Dupuis, C., Gamache, M., and Pagé, J.-F. (2012). Logical analysis of data for estimating passenger show rates at air canada. *Journal of Air Transport Management*, 18(1):78–81.

Hammer, P. L. (1986). Partially defined boolean functions and cause-effect relationships. In *International conference on multi-attribute decision making via or-based expert systems*.

Hammer, P. L. and Bonates, T. O. (2006). Logical analysis of data—an overview: from combinatorial optimization to medical applications. *Annals of Operations Research*, 148(1):203–225.

Hammer, P. L., Kogan, A., and Lejeune, M. A. (2012). A logical analysis of banks' financial strength ratings. *Expert Systems with Applications*, 39(9):7808–7821.

Hammer, P. L., Kogan, A., Simeone, B., and Szedmák, S. (2004). Pareto-optimal patterns in logical analysis of data. *Discrete Applied Mathematics*, 144(1):79–102.

Mortada, M.-A., Carroll, T., Yacout, S., and Lakis, A. (2012). Rogue components: their effect and control using logical analysis of data. *Journal of Intelligent Manufacturing*, 23(2):289–302.

Reddy, A., Wang, H., Yu, H., Bonates, T. O., Gulabani, V., Azok, J., Hoehn, G., Hammer, P. L., Baird, A. E., and Li, K. C. (2008). Logical analysis of data (lad) model for the early diagnosis of acute ischemic stroke. *BMC medical informatics and decision making*, 8(1):30.

Ryoo, H. S. and Jang, I.-Y. (2009). Milp approach to pattern generation in logical analysis of data. *Discrete Applied Mathematics*, 157(4):749–761.