

A fast and general algorithm for Galois lattices building

Fatma Baklouti¹, Gérard Lévy¹, Richard Emilion²

¹ CERIA Laboratory, Paris Dauphine University Place du Maréchal de Lattre de Tassigny 75775 Paris cedex 16, France

fatma.baklouti@dauphine.fr, glevy@dauphine.fr

² MAPMO Laboratory, UMR 6628-MAPMO B.P. 6759, 45067 Orléans cedex 2 France

Richard.Emilion@univ-orleans.fr

Abstract

Standard Galois Lattices are effective tools for data analysis and knowledge discovery. They allow structuring data sets, by extracting concepts and rules to deduce concepts from other concepts. They focus on binary data arrays, called contexts. Several algorithms were proposed to generate concepts or concept lattices on a data context. However, the mining of large databases needs more efficient algorithms. Nowadays, we need to deal with contexts which are large and not necessarily binary. In this paper, we propose a new and fast Galois lattice-building algorithm, called ELL algorithm, for generating closed itemsets from objects having general descriptions; and a generalization of the Ganter algorithm (GGA). A comparison of performance between GGA, ELL, and another published algorithm, called Close By One, is presented.

1 Introduction

Data mining is applied in business to find new market opportunities from data stored in operational databases which are used for day-to-day management. This tool combines ideas from statistics, machine learning, data base technology and high performance computing to find nuggets of knowledge. Data mining is also applied in various domains like scientific research and management of health care. But large data still bothers us in the field of data mining and machine learning. The mining of very large database still need more efficient algorithms. It is proved that Galois lattices are effective tools for data mining. Several algorithms were proposed to generate the concept lattices on a data context. Bordat [2], Ganter [12], Chein [6], Norris [19], Godin [14] and Nourine [20] are examples of such algorithms. [18] describe some of these algorithms and compare their performance. Other comparative studies are presented in [15], [16] and [11].

We notice that most existing work focuses only on binary data. In order to generalize this work, the Galois lattice (GL) formalism was extended to symbolic data by [3] [4] and further developed by [22], [23], [24], [5]. Nevertheless, the general formalism of GL was addressed by [7], [8].

The rationale for this generalization is that nowadays, either descriptions of data are complex, or the size of datasets is drastically growing up so that if, for example, we want to deal with classes of data, we need descriptions that are much more complex than 0 or 1.

[21] proposed an extension of two classical algorithms (Ganter & Chein) and an incremental one (Godin & al) to multivariate, interval and histogram data with missing values.

In this paper we propose a fast Galois lattice-building algorithm, called ELL, based on dichotomic search and working for objects having general description. This work improves the framework started in [10].

Besides, we choose to generalize the Ganter algorithm to ordinal description because it requires little memory size at each step. Indeed, it considers only one concept when looking for the next one. Consequently, using this algorithm may be efficient in particular when considering large databases.

We choose also to implement Close By One (CBO) algorithm described in [18] because it is based on the same principles than ELL. It could be used to build lattices with general contexts. It is proved to be among the fastest algorithms with dense context [18].

The rest of this paper is organized as follows. The main definitions of general Galois lattices are presented in the next section. A complete generalization of Ganter's algorithm is proposed in section 3. Section 4 is devoted to the ELL algorithm, with a recursive version followed by an iterative one. In section 5, comparative performance measures of ELL, GGA and CBO are presented. Section 6 concludes this paper by listing future research directions.

2 General Galois Lattices

Concept lattice [13] and Closed itemset lattice are based on order theory and lattice theory [1], [26]. They are used to represent the order relation on concepts or closed itemsets. Concept lattice describes the character of the set pair: *intent* and *extent* of concept.

In this section, we define some notions generalizing usual ones: Data context, Closure operator, Closed itemset, etc.

Definition 1.1. A *lattice* is a mathematical structure $F = \langle F, \leq, \vee, \wedge, 0_F, 1_F \rangle$, where F is a partially ordered set by the relation \leq , with the largest element 1_F , a smallest element 0_F , and \vee, \wedge are internal composition laws of sup (or supremum), and inf (or infimum). In many situations F is the Cartesian product of several lattices $F_j = \langle F_j, \leq_j, \vee_j, \wedge_j, 0_{F_j}, 1_{F_j} \rangle$, for $j \in J = \{1, \dots, n\}$.

We write this $F = F_1 \times \dots \times F_n = \prod_{j=1}^n F_j$.

The relation \leq on F is defined by $z = (z_1, \dots, z_j, \dots, z_n) \leq t = (t_1, \dots, t_j, \dots, t_n)$ iff $z_j \leq_j t_j$ for each j of J .

We note $z \vee t = (\dots, z_j \vee_j t_j, \dots)$, $z \wedge t = (\dots, z_j \wedge_j t_j, \dots)$, $0_F = (\dots, 0_{F_j}, \dots)$, $1_F = (\dots, 1_{F_j}, \dots)$.

(For standard Galois lattices, we have for each j : $F_j = \{0, 1\}$, $0 < 1$, $0 \vee_j 0 = 0$, $0 \vee_j 1 = 1 \vee_j 0 = 1 \vee_j 1 = 1$, $0 \wedge_j 0 = 0 \wedge_j 1 = 1 \wedge_j 0 = 0$, $1 \wedge_j 1 = 1$. So $0_F = (\dots, 0, \dots)$, and $1_F = (\dots, 1, \dots)$.)

Definition 1.2. General contexts and descriptions: Let m be a finite positive integer, $I = \{1, \dots, m\} = [1, \dots, m]$, and F any lattice. Let $d: I \rightarrow F$ be any mapping from I to F . A **context** C is defined as an array with rows $d(i)$, $i = 1, \dots, m$.

Formalism: The context provides, for each individual or object i of I , its **description** $d(i) = (d_1(i), d_2(i), \dots, d_j(i), \dots, d_n(i)) \in F = F_1 \times \dots \times F_n$, according to the attributes or properties $j \in J$. (In the standard case, $d_j(i)=1$ means that i has property j , and $d_j(i)=0$ means that i has not the property j). So, in the general case, C is an $m \times n$ array of elements $d_j(i)$ of F , and for each individual i and each property j , $d_j(i)$ is the value of this property for i .

Definition 1.3. Galois connection: Let $C = \langle I, F, d \rangle$ be a context. We define $E = P(I)$, $|E| = 2^I$ and $f: E \rightarrow F$ as follows: for each subset X of I , $f(X) = \wedge d(i): i \in X$ if $X \neq \emptyset$, and $f(\emptyset) = 1_F$.

So, $f(X)$ is the infimum of the descriptions $d(i)$ of elements i of X . And in the standard case

$f(X)$ is an element $z = (z_1, \dots, z_j, \dots)$ of F , and $z_j = \wedge_j \{d_j(i) : i \in X\} = 1$, iff $d_j(i)=1$, for each i of X . This means that $z_j = 1$ iff j is a property which belongs to all i in X . For this reason, we call $f(X)$ the intent of X .

Remark:

For each i of I , we have $f(\{i\}) = d(i)$, and f is decreasing (if $X \subseteq X' \subseteq I$ then $f(X') \leq f(X)$).

We define $g: F \rightarrow E$ by $g(z) = \{i \in I: z \leq d(i)\}$, for each z of F .

We say that $g(z)$ is the extent of z . (In standard case, $g(z)$ is the set of all individuals who have all properties of z , $z_j = 1$).

We can see that g is also a decreasing mapping.

The ordered pair (f, g) is called a Galois connection. From it we derive two other mappings:

$h: P(I) \rightarrow P(I)$, by $h = g \circ f$, and $k: F \rightarrow F$ by $k = f \circ g$.

So, for each subset X of I , we have $h(X) = g(f(X)) = \{i \in I: f(X) \leq d(i)\}$, and for each z of F , we have $k(z) = f(g(z)) = \bigwedge \{d(i): i \in g(z)\}$.

We can see that h and k are closure operators. This means that each of them is an increasing, extensive, and idempotent operator. More explicitly, for each X, X' of E , and z, z' of F :

- $X \subseteq X'$ implies that $h(X) \subseteq h(X')$, $z \leq z'$ implies that $k(z) \leq k(z')$;
- $X \subseteq h(X)$, $z \leq k(z)$;
- $h(h(X)) = h(X)$, $k(k(z)) = k(z)$.

Any subset X of I such that $X = h(X)$ is called a I -closed set, and each z of F such that $z = k(z)$ is called **F -closed element**.

Let us define $H = \{X \subseteq I: h(X) = X\}$ the set of all closed subsets of I , and $K = \{z \in F: z = k(z)\}$, the set of all closed elements of F .

One can prove that there is a bijective mapping between H and K . The ordered pairs $(X, z) \in H \times K$ such that $f(X) = z$, and therefore such that $g(z) = X$, are called the concepts associated with the context C .

The set of all such concepts constitutes the **Galois lattice** $GL(C)$ associated with this context C . (the order relation on $GL(C)$ is defined by $(X, z) \leq (X', z')$ iff $X \subseteq X'$ and $z' \leq z$.)

3 A generalization of Ganter's Algorithm

In this section, we present the principles of a generalized version of the Ganter algorithm.

[12] proposed a well-known GL construction algorithm which yields the closed sets in lexicographic order, each set being represented as an element of $F = \{0,1\} \times \dots \times \{0,1\}$.

We propose here a complete generalization for $F = F_1 \times \dots \times F_j \times \dots \times F_n$, where each F_j is totally ordered finite set represented, without loss of generality, as a set of integers, say:

$$F_j = \{0, 1, \dots, b_j\}, 0 < 1 < \dots < b_j.$$

3.1 Product order and lexicographic order

Two order relations \leq and \preceq are usually defined in F :

Let $y = (y_1, \dots, y_n) \in F$ and $z = (z_1, \dots, z_n) \in F$ then the product order \leq is defined by:

$$y \leq z \text{ iff } \forall j = 1, \dots, n : y_j \leq z_j$$

While the lexicographic order \preceq (or prefix order) is defined by:

$$y \preceq z \text{ iff } (y = z) \text{ or } (\exists j \in \{1, \dots, n\} : y_j < z_j \text{ and } (y_k = z_k \forall k < j)).$$

As usual, $y < z$ (vs. $y \prec z$) means that $y \leq z$ (vs. $y \preceq z$) and $y \neq z$.

It is easy to verify that F is partially ordered for \leq and totally ordered for \preceq .

The greatest element of F for \preceq is: $b = (b_1, \dots, b_n)$

3.2 Transition index

For any $a = (a_1, \dots, a_n) \in F$ such that $a < b$, let $a^+ = (a_1^+, \dots, a_n^+) \in F$ denote the successor of a w.r.t the lexicographic order. Define the transition index of a , say $i^+(a)$, as the greatest $j \in \{1, \dots, n\}$ such that $a_j < b_j$. It can be seen that:

$$\forall j < i^+(a) \quad a_j^+ = a_j$$

$$\text{if } j = i^+(a) \quad \text{then } a_j^+ = 1 + a_j$$

$$\forall j > i^+(a) \quad a_j^+ = 0.$$

By convention $i^+(b)$, the transition index of b , is 0.

If $t = i^+(a)$, let $a^* = (a_1^*, \dots, a_n^*)$ be the element of F defined as : $a^* = (a_1, \dots, a_{t-1}, b_t, \dots, b_n)$

It can be observed that we have

$$a = (a_1, \dots, a_{t-1}, a_t, b_{t+1}, \dots, b_n)$$

$$a^+ = (a_1, \dots, a_{t-1}, 1 + a_t, 0, \dots, 0_n)$$

$$a^* = (a_1, \dots, a_{t-1}, b_t, \dots, b_n)$$

The following properties will be useful.

Proposition:

For any $y, z \in F$, we have $y \leq z \Rightarrow y \preceq z$.

For any $a, y \in F$, such that $a < b$, we have $a^+ \leq y \leq a^* \Leftrightarrow a^+ \preceq y \preceq a^*$.

The proof of this proposition is proposed in [9].

To complete this subsection we propose a procedure, called successor and noted **succ**, which computes a^+ and the transition index of $(a_1, \dots, a_{t-1}, a_t)$ in the lattice $F_1 \times \dots \times F_i$.

In the following, we present the procedure **succ**.

Procedure succ ($a \in F; i \in \{1, \dots, n\}; \text{var } a^+ \in F; \text{var } i^+ \in \{1, \dots, n\}$)
<pre> Begin $j = i;$ while $((j > 0) \text{ and } (a_j = b_j))$ do $j = j - 1;$ if $(j > 0)$ then begin for $k = 1$ to $j - 1$ do $a_k^+ = a_k;$ $a_k^+ = 1 + a_j;$ for $k = j + 1$ to n do $a_k^+ = 0;$ $i^+ = j;$ end else $i^+ = j;$ end </pre>

Remark: If the second parameter i is equal to n then **succ** ($a; p; \text{var } a^+; \text{var } i^+$) returns a^+ and $i^+(a)$ as defined above.

3.3 The GGA algorithm

In this subsection, we present the GGA algorithm. Let us note that the variable GL below, which represents the Galois lattice, is a list of nodes.

Procedure GGA
<pre> Var $a, a^+, y \in F; i; j \in \{1, \dots, n\}$ begin </pre>

```

a = 0F; y = k(a); insert y in GL
while (i > 0) do
  begin
    succ (a, i, a+, i+); i = i+;
    if (i > 0) then
      begin
        y = k(a+)
        if (∀ j < i, yj = aj) //that is y ≦ a*
          then
            begin
              insert y in GL; a = y; i = n;
            end
          else i = i-1 //that is a = a*
        end if
      end while
    end
  end
end

```

4 The ELL algorithm

In this section, we describe the main feature of the ELL algorithm, including a recursive version and an iterative version of this algorithm.

4.1 Main feature

For two disjoint subsets X_0 and K of I , let **ELL** (X_0, K) denote a procedure which lists all the closed sets of I obtained by extending X_0 with some elements of K .

In other words, **ELL** (X_0, K) lists all the closed sets, which strictly contain X_0 and are contained in $X_0 \cup K$. Obviously, **ELL** (\emptyset, I) lists all the non-empty closed sets of I . **ELL** (X, K) proceeds by dichotomy as follows:

```

If (K ≠ ∅)
  Choose an element i0 ∈ K
  Find the closed sets which contain i0
  Find the closed sets which do not contain i0
Endif

```

The key point of the proposed algorithm is that the search time of such closed sets is considerably reduced by using the following proposition.

Proposition:

Let X_0 and $K \neq \emptyset$ be two disjoint subsets of I . Let $i_0 \in K$.

a) We have

$$h(X_0 \cup \{i_0\}) = X_0 \cup A \text{ where } A = \{i \in \Lambda X_0 : f(X_0) \wedge f(i_0) \leq f(i)\}$$

If a closed set contains X_0 and i_0 , then it also contains A . Hence, if $A \subseteq K$ then $X_0 \cup A$ is the smallest closed set containing X_0 and i_0 and contained in $X_0 \cup K$.

If a closed set contains X_0 and does not contain i_0 , then it also does not contain any element of the set $R = \{i \in K : f(X_0) \wedge f(i) \leq f(i_0)\}$.

A (vs. R) is used for Attraction (vs. Rejection).

The proof of this proposition is proposed in [9].

4.2 The recursive version

The following pseudo-code is a recursive version of the algorithm.

<pre> Procedure ELL (X_0, K) $GL = \emptyset$ Var i_0: element of I, z, z_0: elements of F; X, A, R: subsets of I; begin $z_0 = f(X_0)$; if $K \neq \emptyset$ then begin choose an element i_0 of K; $z = z_0 \wedge f(i_0)$; $A = \{i \in IX_0 : z \leq f(i)\}$; if $A \subseteq K$ then begin $X = X_0 \cup A$; insert node (X, z) in GL; ELL ($X, K \setminus A$); end; $R = \{i \in K : z_0 \wedge f(i) \leq f(i_0)\}$; ELL ($X_0, K \setminus R$); end end </pre>

The procedure **ELL** (\emptyset, I) starts with any $i_0 \in I$ (I is non empty). Then it determines the set A . Observing that $i_0 \in A$, we have the strict inclusions $\emptyset \subsetneq X$ and $IA \subsetneq I$. Hence if $A \subseteq K$, **ELL**(X, IA) will run with a strictly smaller second parameter. Since $i_0 \in R$, we see that the same holds for **ELL** (X, IR).

More generally **ELL** ($X, K \setminus A$) and **ELL** (X_0, IR) run with a strictly smaller second parameter than that of **ELL** (X, K). Since I is finite, this parameter which is a subset of I , will reach the void set and the procedure will terminate.

This algorithm lists all closed sets without duplicates. Let us show that each closed set F occurs exactly once.

Starting with $GL = \emptyset$, $X_0 = \emptyset$ and $K = I$, let $i_0 \in I$ be fixed and consider two cases: $i_0 \in F$ and $i_0 \notin F$. If $i_0 \notin F$ then

Either F is the smallest closed set which contains i_0 . Then according to the previous proposition (a), $F = X = X_0 \cup A$ and F will be listed by **ELL**(X_0, K),

Or F is not the smallest one. In this case $X \subsetneq F$ and F will be listed by **ELL**($X, K \setminus A$).

If $i_0 \in F$, then according to the previous proposition (c), F will be listed by **ELL**($X_0, K \setminus R$). Since each insert only concerns the unique smallest closed set containing X_0 and i_0 , we see that F occurs exactly once. This will also be seen in the iterative implementation given below.

The only case not treated by the algorithm is whether the empty set is closed or not. The algorithm yields all the nodes (X, z) of the GL such that $X \neq \emptyset$. Since $f(\emptyset) = 1_F$ and $h(\emptyset) = \{i \in I : 1_F \leq f(i)\} = \{i \in I : f(i) = 1_F\}$, \emptyset is closed iff there is no $i \in I$ such that $f(i) = 1_F$.

4.3 The iterative version

The implementation of the recursive version of this algorithm has been successfully tested. Nevertheless for very large datasets, an iterative version is needed in order to speed up the search.

<pre> Procedure ELL GL = \emptyset Var i_0: element of I; s: element of F; A, R: Subset of I; l: [1..$n+1$]; i: array [1..$n+1$] of elements I; z: array [1..$n+1$] of elements of F; X, K: array [1..$n+1$] of subsets of I; Begin $l = 1$; $X_1 = \emptyset$; $K_1 = I$; $z_1 = 1F$; while ($l > 0$) do begin if $K_l \neq \emptyset$ then begin Choose an element i_0 of K_l; $s = z_l \wedge f(i_0)$; $A = \{r \in \bigcap X_l : s \leq f(r)\}$; if $A \subseteq K_l$ then begin $i_l = i_0$; $l = l + 1$; // push up the stack $X_l = X_{l-1} \cup A$; $K_l = K_{l-1} \setminus A$; $z_l = s$; Insert node (X_l, z_l) in GL; // X_l is a closed set end else $R = \{r \in K_l : z_l \wedge f(r) \leq f(i_0)\}$; $K_l = K_{l-1} \setminus R$; end else // $K_l = \emptyset$ begin $l = l - 1$; // Push down the stack if ($l > 0$) then begin $i_0 = i_l$; $R := \{r \in K_l : z_l \wedge f(r) \wedge f(i_0)\}$; $K_{l+1} := K_{l+1} \setminus R$; end end end end while end </pre>

4.4 The choice of i_0

At each iteration of the procedure ELL, we have to choose an element i_0 of K . If at each iteration we choose i_0 as the smallest element of K then the concepts will be generated in the lexicographic order. We remark that:

$$\forall i_0, i \in \bigcap X_0 : f(X_0) \wedge d(i) \leq f(X_0) \wedge d(i_0) \Rightarrow A(X_0, i_0) \subseteq A(X_0, i);$$

$$\text{and } X_0 \cup A(X_0, i_0) \subseteq X_0 \cup A(X_0, i) \text{ or } h(X_0 \cup \{i_0\}) \subseteq h(X_0 \cup \{i\})$$

To get the smallest closed sets that contain strictly X_0 , it is necessary to add to X_0 all elements $i_0 \in \bigcap X_0$ which maximize the function $\delta: \bigcap X_0 \rightarrow F$ defined by: $\delta(i) = f(X_0) \wedge d(i)$

This will permit constructing the Hass diagram.

4.5 Complexity

According to the recursive version, obtaining a closed set requires $O(|M|^2|N|)$ operations, where $|M|$ is the number of objects and $|N|$ is the number of attributes. Therefore the complexity of this algorithm is $O(|M|^2|N||L|)$, where $|L|$ is the size of the concept lattice.

The implementation of the recursive version shows that the memory size required is $O(|M|2)$. Indeed, the vectors X_0 and K_0 , whose size is $|M|+|M|$, are stored in a stack at each iteration of the algorithm which requires $|M|$ steps for processing all the closed itemsets.

On the other hand, the implementation of the iterative version shows that the memory size required is $O(|M|)$. Indeed, the vector constituted of all the objects ($X_0 \cup K_0 \cup R$), whose size is $O(|M|)$, is stored only one time during the execution of the algorithm. Only the order of the components of the vectors is modified at each iteration.

4.6 Application:

Since the algorithm can deal with general data, we have just to redefine the set F , the order relation \leq , the operation of infimum \wedge and the largest element 1_F . The table below presents some examples of such redefinitions.

Variable	Set of values F	\leq	\wedge	1_F
Boolean	$\{0,1\}$	$0 < 1$	Min	1
Ordinal	Finite ordered set	$0 < 1 < 2 \dots < b$	Min	b
Dataset	Set of subsets	\subseteq	\cap	E
Interval	Subset interval	\subseteq	\cap	E
Fuzzy	Mapping: $E \rightarrow [0,1]$	$f \leq g$	$\text{Min}(f(a), g(a))$	1

The following example is an application of our algorithm in the case of interval variable.

Let us use the context C from the example described below defined by the following array, where $m=4$, $n=3$ and $b=([0,5],[0,1],[0,4]) = 1_F$.

$j \rightarrow$	A	B	C
1	[3,4]	[0,1]	[0,4]
2	[0,2]	[0,1]	[4,4]
3	[2,5]	[1,1]	[4,4]
4	[2,3]	[1,1]	[0,4]

This application leads to a list of closed itemsets. When we chose i_0 as described in the subsection 4.4, we construct the following Hass diagram.



5 Experimental results

We have implemented the generalized version of the Ganter algorithm, CBO and the iterative version of the ELL algorithm. The CBO algorithm was implemented as described in [18]. The preliminary results of these implementations were performed on a PIII 1.8 GHz computer with 1GB RAM using binary random data, and real data.

In the case of binary data, we consider that the density (D) of a context is the proportion of 1 in the entire context ($|M| \times |N|$).

Fig. 1. Concept set $|M| = 100$, $D = 25\%$

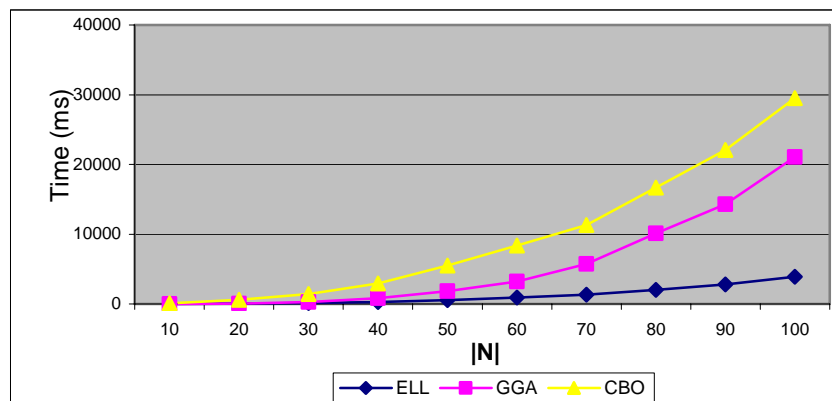


Fig. 2. Concept set $|M| = 100, D = 25 \%$

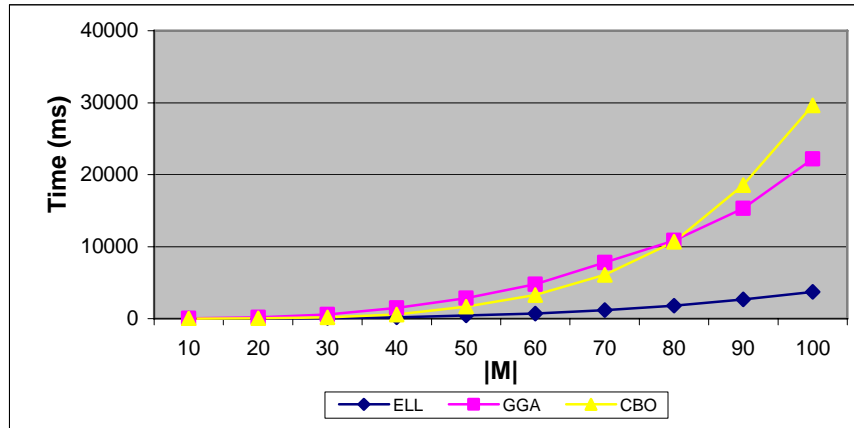


Fig. 3. Concept set $|M| = 100, D = 50 \%$

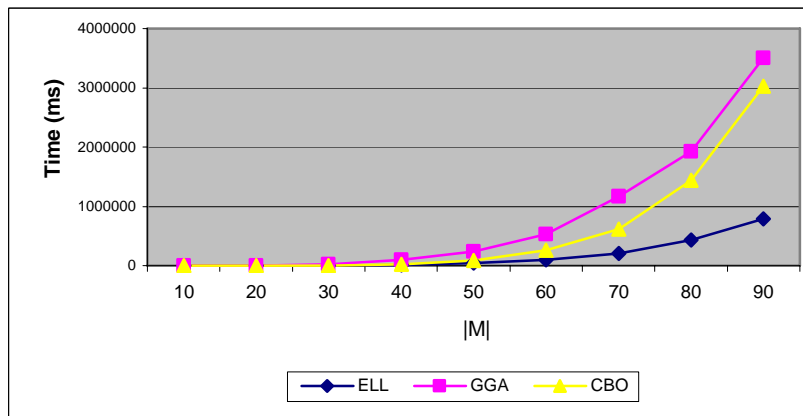
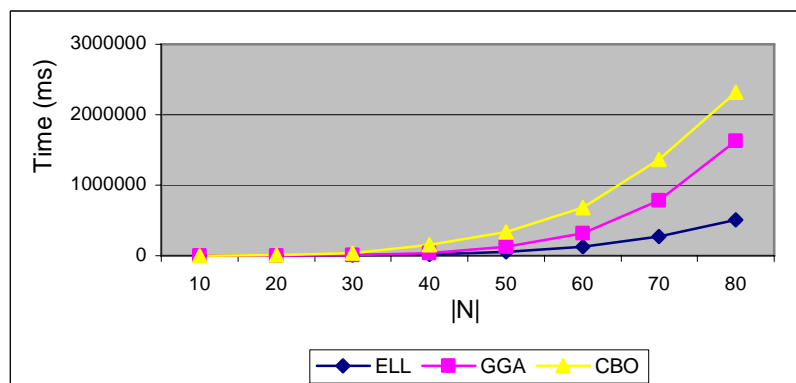


Fig. 4. Concept set $|M| = 100, D = 50 \%$



The charts above show the running time of ELL, GGA and CBO in the case of binary data.

In the following the expression “ $X > Y$ ” means that the algorithm X is faster than the algorithm Y . Therefore,

When $M > N, \forall D$ (Fig.1, Fig. 4):

ELL $>$ CBO and GGA $>$ CBO

If $M \gg N$ then GGA $>$ ELL else ELL $>$ GGA

When $M < N, \forall D$ (Fig. 2, Fig. 3)

ELL $>$ CBO $>$ GGA

If $N \gg M$ then CBO $>$ ELL $>$ GGA

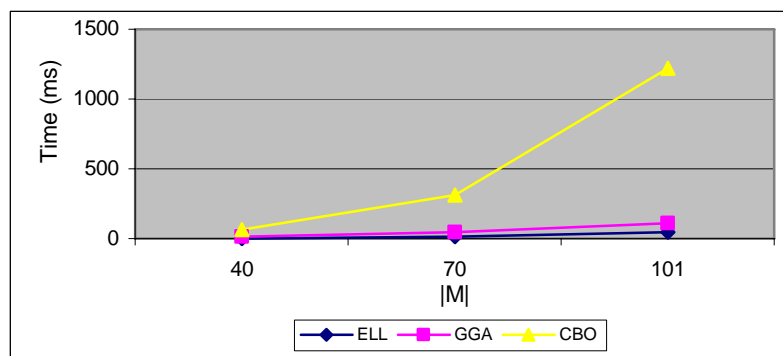
When $M \# N$:

If the density is high (the worst case): CBO $>$ ELL $>$ GGA else ELL $>$ GGA $>$ CBO.

We also performed comparisons on the Zoo database constituted of 101 objects and 18 attributes which are: animal name, 15 Boolean attributes, and 2 numerics (set of values and integer values in range). This database is available from the UCI Machine learning Repository [17]. The results are illustrated by fig. 5 which shows that ELL $>$ GGA $>$ CBO.

The charts above show the running time of ELL, GGA and CBO in the case of binary data.

Fig. 5. Evolution of the CPU-time for the three algorithms (Zoo database)



In the rest of this section, we try to provide a synthetic explanation of the results listed above. CBO and ELL are based on the search of the set of subsets of $I = \{1, \dots, n\}$. While GGA is based on the search of the set of strings of $F = F_1 \times F_2 \times \dots \times F_n$. These two algorithms consist in completing the set X_0 with an element i_0 .

CBO constructs the Galois lattice in the lexicographic order. Indeed, to complete X_0 with i_0 , the algorithm proceeds by eliminating elements $i < i_0$.

By another way, to complete X_0 with i_0 , ELL determines all the closed sets which contain X_0 and i_0 (the set A in the algorithm), as well as all closed sets which contain X_0 and not contain i_0 (the set R in the algorithm).

It seems that ELL eliminates more elements than CBO. This may explain why ELL is faster than CBO.

In his turn, GGA explores partially the set F in the lexicographic order. In that way, it does not process all the elements of F .

The number of elements eliminated by ELL may be greater than those eliminated by GGA, this may explain why ELL is faster than GGA.

On the other hand, $|E| = 2^m$ and $|F| = b_1 \times b_2 \times \dots \times b_n$.

If each attribute j gets its values in the set $F_j = \{0, 1, \dots, b_{j-1}\}$ where $0 < 1 < \dots < b_{j-1}$ then $|F_j| = b_j$ and $|F| = |F_1| \times \dots \times |F_n| = b_1 \times b_2 \times \dots \times b_n$
For example, in the binary case $b_j = 2$, so $|F| = 2^n$ (where n is the number of attributes).

6 Conclusion

In this paper, we presented an algorithm, ELL, which had been proposed first in [10]. We have also compared it with GGA and CBO, for general Galois lattices building.

Let us note that GGA was validated only in the case of ordinal data, we think that it may be useful to extend this validation to symbolic data.

From the results presented above, one can see that the choice of an algorithm should be based on the properties of the context such as the number of attributes, the number of objects, and the density. This is compliant with the conclusions of [18]. One can also note that in most cases ELL is faster than GGA and CBO. In cases where ELL is not the fastest one, its speed is comparable with that of the best performer.

A wider study is needed, to compare ELL to other algorithms which provide the list of concepts, and to those which construct the diagram graph. Indeed, the ELL's formulation is that of a local procedure, so it allows us to obtain sons of any closed set easily and thus the Hasse diagram. This formulation could also be used to obtain distributed version of ELL in order to work with very large data. This is ongoing research.

Acknowledgements

We would like to express special thanks to Pr. Witold Litwin for his precious remarks, to Pr. G. Lambert, one of the fathers of ELL algorithm, and to Pr. D. Zégour (I.N.I, Algiers University) who has helped us to implement our experimentations.

We also thank anonymous reviewers for their constructive and very useful remarks.

References

- [1] Birkhoff. G. Lattice Theory. American Mathematical Society, Providence, RI, 3rd edition. 1967.
- [2] Bordat. J. Calcul pratique du treillis de Galois d'une correspondance, *Mathématique, Informatique et Sciences Humaines* 24(94), 31. 1986.
- [3] Brito. P. Analyse de données symboliques. Pyramides d'héritage, thèse Paris IX Dauphine. 1991.
- [4] Brito. P., "Order Structure of Symbolic Assertion Objects", in *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, N. 5, (1994), 830-835.
- [5] Brito. P, Polaiillon. G, "Structuring probabilistic data by Galois lattices", *revue MSH-MSS* 2005, to appear.
- [6] Chein. M. Algorithme de recherche des sous matrices premières d'une matrice, *Bull.Math.R.S.* 13, 1969.
- [7] Diday. E, Emilion. R. Treillis de Galois maximaux et capacités de Choquet. *Cahier de Recherche de l'Académie des Sciences. Paris*, t.325, Série I, p.261-266, 1997.
- [8] Diday. E, Emilion. R. "Maximal and stochastic Galois lattices", *Discrete Applied Mathematics*, 127, (2003), 271-284.
- [9] Emilion. R, Lambert G, Lévy. G. Algorithms for general Galois lattice building. Technical report. CERIA, University PARIS IX Dauphine. 2001.
- [10] Emilion. R, Lambert. G, Lévy. G, Algorithmes pour les treillis de Galois, *Indo-French Workshop.*, University Paris IX-Dauphine. 1997.

- [11] Fu. H., Mephu Nguifo E., "Etude et conception d'algorithmes de génération de concepts formels", dans : J.-F. Boulicault and B. Crémilleux, *Revue d'Ingénierie des Systèmes d'Information (ISI)*, Édition Hermès, Lavoisier, vol. 9, RSTI, n° 3-4, pp 109-132, Paris, France, Septembre 2004
- [12] Ganter. B. Two basic algorithms in concept analysis. Preprint 831, Technische Hochschule Darmstadt, 1984.
- [13] Ganter. B, Wille. R. *Formal Concept Analysis*. Mathematical Foundations, Springer. 1999.
- [14] Godin. R., Mineau. G. and al. Méthodes de classification conceptuelle bases sur les treillis de Galois et application. *Revue d'intelligence artificielle* pp 105-137. 1995.
- [15] Godin. R, Chau T. T. Comparaison d'algorithmes de construction de hiérarchies de classes, rapport, 1998, Université de Québec.
- [16] Guénoche. A. Construction du treillis de Galois d'une relation binaire. *Math. Inf. Si. Hum.* (28ème année, n° 109, 1990, p 41-53).
- [17] <http://www.ics.uci.edu/~mlearn/MLSummary.html>
- [18] Kuznetsov. S, Ob'edkov. S. Comparing the performance of algorithms for generating concept lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14 (2-3): 189–216, 2002.
- [19] Norris. E. An algorithm for computing the maximal rectangles in a binary relation. *Revue Romaine Math. Pures et Appl.* XXIII (2), 243. 1978.
- [20] Nourine. L., Raynaud. O. A fast algorithm for building Lattices. *Information Processing Letters* 71, 199. 1999.
- [21] Polaillon. G, Diday. E, Symbolic Galois Lattices of multivariate and interval tables, *Proceedings of OSDA'97*, Darmstadt.1997.
- [22] Polaillon G., Organisation et interprétation par les treillis de Galois de données de type multivalué, intervalle ou histogramme, Thèse de Doctorat, Université Paris IX Dauphine, 1998-a.
- [23] Polaillon G., Interpretation and reduction of Galois lattices of complex data, in *Advances in Data Science and Classification*, eds. Rizzi A., Vichi M., Bock H.-H., Springer-Verlag, (1998-b), 433-440.
- [24] Polaillon G., Diday E., Reduction of symbolic Galois lattices via hierarchies, in: *Proceedings of the Conference on Knowledge Extraction and Symbolic Data Analysis (KESDA'98)*, Office for Official Publications of the European Communities, Luxembourg, (1999), 137-143.
- [25] Wille. R. Restructuring Lattice Theory. in I. Rival (ed.), *Symposium on Ordered Sets*, pp 445-470, University of Calgary, Boston. 1982.