

A fast and general algorithm to build Galois lattices

Gérarg Lévy
CERIA Laboratory
glevy@dauphine.fr

Fatma Baklouti
CERIA Laboratory
fatma.baklouti@dauphine.fr

ABSTRACT

Standard Galois Lattices are effective tools for data analysis and knowledge discovery. Many research works in classification or association rules increase the interest of them for data mining. They allow structuring data sets, by extracting concepts and rules to deduce concepts from other concepts. They concern binary data arrays, called *contexts*. Several algorithms were proposed to generate concepts of lattices, among which Ganter algorithm is the best one for large contexts. However, it's also hard to face the complexity of large data with Ganter algorithm. So we propose a new and fast Galois lattice-building algorithm GL which works for objects having very general descriptions. And we propose a way to parallelise large context.

General Terms

Algorithms, Performance, Experimentation, Standardization, Languages, Theory.

Keywords

Concept lattice, Closure operator, Closed itemset, Data mining.

1. INTRODUCTION

Data mining is applied in business to find new market opportunities from data stored in operational data bases which are used for day-to-day management. The tool applied combine ideas from statistics, machine learning, data base technology and high performance computing to find nuggets of knowledge. Data mining is also applied in science for example to find taxonomies of variable stars and in the national administration for 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 algorithm. For example, in the context of association rule mining, it's a hard problem to find all frequent itemsets in large data. It needs more efficient algorithms and methods. Closed itemsets lattices are parts of concept lattices. The problem of finding frequent itemsets from data for association rules can be reduced to finding frequent closed itemsets with closed itemset lattice.

Several algorithms were proposed to generate concepts or concept lattices on a data context, for example: Bordat (Bordat, 1986) [2], Ganter (Ganter, 1984) [6], Chein (Chein, 1969) [3], Norris (Norris, 1978) [16], Godin (Godin et al., 1995) [10]

and Nourine (Nourine and Raynaud, 1999) [17], etc. Experimental comparisons of performance of these algorithms show that Ganter algorithm is the fastest one [12]. So in this paper we propose a new and fast Galois lattice-building algorithm GL based on dichotomic search and working for objects having very general description. Experimental results show that the proposed algorithm outperforms the earlier ones. In the rest of the paper, we propose a way to parallelise large contexts of general lattices since we need to treat huge contexts.

The paper is organized as follows. Section 2 contains the main definitions of general Galois lattices. A new algorithm for generating closed itemsets, called GL algorithm, will be proposed in section 3. In section 4, the performance of the new algorithm will be shown. A way to parallelise large context is proposed in section 5. The paper ends with a conclusion in section 6.

2. GENERAL GALOIS LATTICES

Concept lattice (Ganter and Wille, 1999) and Closed itemset lattice are based on order theory and lattice theory (Birkhoff, 1967; Wille, 1982). They are used to represent the order relation of concepts or closed itemsets. Concept lattice describes the character of the set pair: *intent* and *extent* of concept.

In this section, we define some basic notions: 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 relationship \leq , with a 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 $F = F_1 \times \dots \times F_j \times \dots \times F_n = \prod_{j=1}^n F_j$.

The relationship \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 .

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

(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. Contexts and descriptions

Let m be a finite positive integer, $I = \{1, \dots, m\} = [1..m]$, and F any lattice. Let $d: I \rightarrow F$ be any mapping from I to F . By definition, the array with rows $d(i), i = 1, \dots, m$, is a **context** C .

The context gives 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 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 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 $E = 2^I = P(I)$ and $f: E \rightarrow F$ as follows: for each subset X of I , f

$$f(X) = \bigwedge \{d(i) : i \in X\} \text{ if } X \neq \emptyset, \text{ and } f(\emptyset) = 1_F.$$

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

$f(X)$ is an element $z = (\dots, z_j, \dots)$ of F , and $z_j = \bigwedge_j \{d_j(i) : i \in X\} = 1$, 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 . And 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 $X \subset X' \subset I$ then $f(X') \leq f(X)$).

Now, 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 define 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 :

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

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 $H = \{X \subset I : h(X) = X\}$ 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 bijection 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 relationship on $GL(C)$ is defined by $(X, z) \leq (X', z')$ iff $X \subset X'$ and $z' \leq z$.)

3. GENERAL GALOIS LATTICE ALGORITHMS

Several algorithms were proposed to generate concepts or concept lattices on a binary data context, for example: Bordat (Bordat, 1986), Ganter (Ganter algorithm) (Ganter, 1984), Chein (Chein, 1969), Norris (Norris, 1978), Godin (Godin et al., 1995) and Nourine (Nourine and Raynaud, 1999), etc. Experimental comparisons of performance of these algorithms show that Ganter algorithm is the best one.

In this section, we present the principle of ganter algorithm and we propose a new, fast and general Galois lattice-building algorithm **GL** which works for objects having very general description (not necessarily binary data).

3.1 Ganter algorithm

This algorithm constructs all closed items for any closed relation; it has been developed in the field of concept analysis. The principle of Ganter algorithm uses the characteristic vector.

$A = (a_1, a_2, \dots, a_m)$ and $B = (b_1, b_2, \dots, b_m)$ are two binary characteristic vectors verifying the lexicographic order $<$. For example $(0,0,0,0) < (0,0,0,1) < (0,0,1,0) < (0,1,0,0)$ etc.

Suppose that $A < B$ and i is the index, search as for all $j < i$ we have $a_j = b_j$ and $a_i < b_i$. ($a_i = 0$ and $b_i = 1$). We note $A <_i B$.

If in $A = (a_1, a_2, \dots, a_m)$ there is $a_j = 0$, we note $A_j = (a_1, a_{j-1}, 1, 0, \dots, 0)$ and we define $A \oplus j = h'(A_j)$.

B. Ganter deduce three lemmas and a proposition:

- $A < A \oplus i$
- $A <_i B$ and B closed $\Rightarrow A \oplus i \leq B$
- $A <_i B$ and B closed $\Rightarrow A <_i A \oplus i$

Proposition: The smallest closed item placed after A for the lexicographical order is $A \oplus i$ where i is the greatest index search as $A <_i A \oplus i$.

Ganter algorithm is found on this proposition.

Algorithm

$A = (0, 0, \dots, 0)$

While $A \neq (1, 1, \dots, 1)$ do

$i = m$;

1/ if $a_i = 1$ do $i = i-1$; go to 1/

else do

For $j = i+1$ to m do $a_j = 0$

$A' = h'(A)$

For $j = 1$ to $i-1$ do

If $a_j < a'_j$; then $i = i-1$; go to 1/
 End;
 $A = A'$;

End.

Standard Ganter algorithm concern binary data but today we need to treat contexts which are large and not necessarily binary. In [5] a general version of Ganter algorithm is presented. We will use the general version for the comparison with the GL algorithm.

Ganter algorithm can generate closed item sets or concepts more rapidly than other lattice algorithms for large data, but it still takes very high time cost to deal with large data.

3.2 GL algorithm

3.2.1 Main feature

For two disjoint subsets X_0 and K of I let $\text{Closed}(X_0, K)$ denote a procedure which lists all the closed sets of I obtained by extending X_0 with some element of K .

In other words $\text{Closed}(X_0, K)$ lists all the closed sets, which strictly contain X_0 and are contained within $X_0 \cup K$. Obviously $\text{Closed}(\emptyset, I)$ will then list all the non-empty closed sets of I .

$\text{Closed}(X, K)$ will proceed by dichotomy as follows:

If ($K \neq \emptyset$)

 Choose an element $i_0 \in K$
 Find the closed sets which contain i_0
 Find the closed sets which do not contain i_0

Endif

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

Proposition:

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

a) We have

$$h(X_0 \cup \{i_0\}) = X_0 \cup A$$

where

$$A = \{i \in I \setminus X_0; f(X_0) \wedge f(i_0) \leq f(i)\}$$

b) 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 within $X_0 \cup K$.

c) If a closed set contains X_0 and does not contain i_0 , then it also does not contain any element of the set.

$$R = \{i \in K; f(X_0) \wedge f(i) \leq f(i_0)\}$$

3.2.2 A recursive version

The following pseudo-code is a recursive version of the algorithm. The variable GL below, which represents the Galois lattice, is a list of nodes.

$GL = \emptyset$.

Procedure: *Closed* (X_0, K)

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 ;

$z = z_0 \wedge f(i_0)$; $A = \{i \in I \setminus X_0; z \leq f(i)\}$;

 if $A \subseteq K$ then

 begin

$X = X_0 \cup A$; insert node (X, z) in GL ;

Closed ($X, K \setminus A$);

 end;

$R = \{i \in K; z_0 \wedge f(i) \leq f(i_0)\}$; **Closed** ($X_0, K \setminus R$);

 end;

end;

The implementation of the recursive version of this algorithm has been successfully tested.

Example:

Let us consider the context C which is given by the following array, where $m = 5, n = 3$,
 $I = [1 \dots 7], b_1 = 3, b_2 = 2, b_3 = 3. (b = 1_F)$

$j \rightarrow$ $i \downarrow$	1	2	3
1	0	1	1
2	2	0	1
3	3	0	3
4	1	2	3
5	2	1	0

- $X_0 = \emptyset$ and $K = I = \{1, 2, 3, 4, 5\}$

▪ We choose the element $i_0 = 1$ of K ;

▪ $f(X_0) = z = 1_F = (3, 2, 3)$

▪ $z = z_0 \wedge f(i_0) = (3, 2, 3) \wedge (0, 1, 1) = (0, 1, 1)$

▪ $A = \{i \in I \setminus X_0; z \leq f(i)\} = \{1, 4\}$

▪ $A \subseteq K$

▪ $X = \{1, 4\}$ and $z = (0, 1, 1)$, (X, z) *Closed item pair*

▪ $X = X_0 \cup A = \emptyset \cup \{1, 4\}$

- $X_0 = \{1, 4\}$ and $K = \{2, 3, 5\}$

▪ We choose the element $i_0 = 2$ of K ;

▪ $f(X_0) = z = (0, 1, 1)$

▪ $z = z_0 \wedge f(i_0) = (0, 1, 1) \wedge (2, 0, 1) = (0, 0, 1)$

▪ $A = \{i \in I - X_0; z \leq f(i)\} = \{2, 3\} \subseteq K$

▪ $X = X_0 \cup A = \{1, 4, 2, 3\}$

▪ $X = \{1, 4, 2, 3\}$ and $z = (0, 0, 1)$, (X, z) *Closed item pair*

-

Total number of closed pairs (X, z) of lattice $T = GL(C) = 14$.

- pair (1)= $x = \{1,4,\}$ $z = \{0,1,1,\}$
- pair (2)= $x = \{1,4,3,2,\}$ $z = \{0,0,1,\}$
- pair (3)= $x = \{1,4,3,2,5,\}$ $z = \{0,0,0,\}$
- pair (4)= $x = \{1,4,5,\}$ $z = \{0,1,0,\}$
- pair (5)= $x = \{2,3,\}$ $z = \{2,0,1,\}$
- pair (6)= $x = \{2,3,5,\}$ $z = \{2,0,0,\}$
- pair (7)= $x = \{2,3,5,4,\}$ $z = \{1,0,0,\}$
- pair (8)= $x = \{2,3,4,\}$ $z = \{1,0,1,\}$
- pair (9)= $x = \{5,\}$ $z = \{2,1,0,\}$
- pair (10)= $x = \{5,4,\}$ $z = \{1,1,0,\}$
- pair (11)= $x = \{3,\}$ $z = \{3,0,3,\}$
- pair (12)= $x = \{3,4,\}$ $z = \{1,0,3,\}$
- pair (13)= $x = \{4,\}$ $z = \{1,2,3,\}$
- pair (14)= $x = \{\}$ $z = \{3,2,3,\}$

The case $X_0 = \emptyset$ isn't treated by the algorithm so the test must be added.

In this example:

- $X = \emptyset$
- $f(X) = z = 1_F = (3, 2, 3)$
- $g(z) = \{i \in I / z \leq d(i)\} = \emptyset$
- $X = \emptyset$ and $z = 1_F = (3, 2, 3)$, (X, z) : *Closed item pair*

The recursive version use a great memory place, so we will use an iterative version, detailed in [5].

4. EXPERIMENTAL RESULTS

We have implemented the generalized version of the Ganter algorithm and the iterative version of the GL algorithm in C++. Preliminary results of our implementation show that the iterative version has efficient performance.

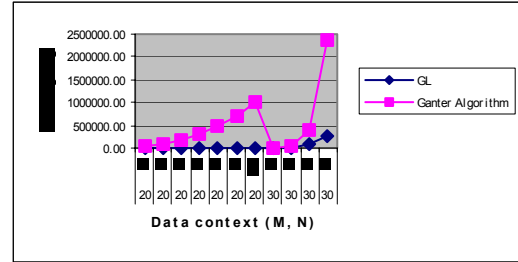
We have tested these two algorithms with random data contexts (M, N) .

M	N	Time GL	Time NC
20	40	2640,60	36573,50
20	50	4222,00	88149,90
20	60	5575,00	170220,30
20	70	8384,60	324807,60
20	80	9581,40	499229,50
20	90	11007,90	714281,20
20	100	12525,10	1018060,80
30	10	143,90	223,30
30	20	7953,20	22231,10
30	30	76332,90	415517,1
30	40	276812,7	2369779,5

- M: number of individuals
- N: number of items
- Time GL: Time in millisecond with GL algorithm
- Time NC: Time in millisecond with Ganter algorithm
- NF: Number of closed pairs

The comparison between them shows that the total time of computing all closed pairs with GL algorithm is remarkably lower than that of the Ganter algorithm. (Figure 1)

Figure 1: Performance of comparison for general Ganter algorithm and GL algorithm with 10 test data



We have also tested our algorithm in the worst case, and it succeeds in computing some large data that were impossible to be compute with other algorithm. For example: the worst case with 30 items is very hard to compute with other algorithms. Other lattice-based algorithm failed to build closed item sets for data with more than 20 items. Using GL algorithm, we have generated all closed items for worst-case data sets with 30 items.

5. A PARALLELISATION OF LARGE CONTEXTS

Today, we need to treat contexts which are huge and not necessarily binary. Since it is difficult to determine Galois lattice for large contexts C on one only computer, we propose to partition C depending on its rows or columns, to build on different computers Galois lattices associated to sub-contexts, and then to determine the global lattice from these lattices [14].

Row-sharing / Column-sharing:

If the context C is too large to be solved with one only computer; we partition the array associated with $C = < I, F, d >$ into two sub-arrays C_1 and C_2 .

We suppose that we have three workstation M, M_1 and M_2 .

We apply the GL algorithm to the context C_1 on a machine M_1 and C_2 on a machine M_2 to build respectively lattice $T_1 = GL(C_1)$ and lattice $T_2 = GL(C_2)$.

We apply an algorithm to compute the $T = GL(C)$ from T_1 and T_2 .

The algorithm applied corresponds with row-sharing (column-sharing), if we share the context on rows (on columns). These two algorithms are detailed in [14].

The idea of sharing contexts into two subsets could be easily extended to multipartitions and to many workstations; one could imagine building different architectures for networks of stations to manage very complex situations.

We are implementing this feature on SDDS systems which have been developed at CERIA center (University Paris IX Dauphine).

The Scalable and Distributed Data Structures (SDDSs) are one class of data structures that is defined specifically for multicomputers. An SDDS file is distributed over the server nodes. The number of nodes dynamically scales with the file growth. The application accesses the file through a client node that makes the data distribution transparent. For scalability, the data address calculations do not involve any centralized directory. The data are also basically stored in the distributed RAM, for faster access than to the traditional disk-based structures [15].

6. CONCLUSION

The work proposed in this paper, generalizes results known for binary description, and shows that the proposed algorithm (GL) outperforms the earlier ones.

Although the GL algorithm succeeds in computing some large data in worst case that are impossible to be computed with other algorithms, today we need to treat huge context.

Since it is difficult to determine Galois lattice for too large contexts C on one only computer, we propose to partition C depending on its rows or columns.

Then we use the GL algorithm to build on different computers Galois lattices associated to sub-contexts, and then we determine the global lattice from these lattices [14]. The aim was to propose a way to parallelise large contexts of general lattices.

The idea of sharing contexts into two subsets could be easily extended to multipartitions and to many workstations; one could imagine building different architectures for networks of stations to manage very complex situations.

7. 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] Chein. M. Algorithme de recherche des sous matrices premières d'une matrice, *Bull.Math.R.S.* 13, 1969.
- [4] Diday. E, Emilion. R. Treillis de Galois maximaux et capacités de Choquet. *C.R.Acad.Sci. Paris*, t.325, Série I, p.261-266, 1997.
- [5] Emilion. R, Lambert G, Lévy. G. Algorithms for general Galois lattice building. Technical report. CERIA, University PARIS IX Dauphine. 2001.
- [6] Ganter. B. Two basic algorithms in concept analysis. Preprint 831, Technische Hochschule Darmstadt, 1984.
- [7] Ganter. B, Wille. R. Formal Concept Analysis. *Mathematical Foundations*, Springer. 1999.
- [8] Ganter. B. Formal Concept Analysis: algorithmic aspects. TU Dersden. 2002.

[9] Godin. R. Complexité de structures de treillis. *Ann. SC. Math. Quebec*, 13 (1): 19-38, 1989.

[10] 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.

[11] Godin. R, Mineau. G.W, Missaoui. R. Incremental structuring of knowledge bases. *Proceedings of the International knowledge retrieval, use, and storage for efficiency Symposium (KRUSE' 95)*, Santa Cruz: pp 179-198.

[12] Guénoche. A. Construction du treillis de Galois d'une relation binaire. *Math. Inf. Si. Hum.* (28ème année, n° 109, 1990, p 41-53).

[13] Lévy. G., Correspondances et treillis de Galois. Technical report.

[14] Lévy. G., Baklouti. F. Parallel algorithms for general Galois lattices building. *Workshop WDAS. CERIA, University PARIS IX Dauphine.* 2003.

[15] Litwin. W., With Neimat M-A., Schneider. D., RP*: A Family of Order-Preserving Scalable Distributed Data Structures. *VLDB-94, Chile.*

[16] Norris. E. An algorithm for computing the maximal rectangles in a binary relation. *Revue Romaine Math. Pures et Appl.* XXIII (2), 243. 1978.

[17] Nourine. L., Raynaud. O. A fast algorithm for building Lattices. *Information Processing Letters* 71, 199. 1999.

[18] Njiwoua. P, Mephu Nguifo. E. A parallel algorithm to build concept lattice. In *Proceedings of 4th Groningen Intl. Information Tech. Conf. For Students*, pp. 103-107, 1997.

[19] Wille. R. Concept Lattices and Knowledge Systems. *Computer Mathematic Applied*, 23 (6-9): 493-515, 1992.

[20] Wille. R. Restructuring Lattice Theory. in I. Rival (ed.), *Symposium on Ordered Sets*, pp 445-470, University of Calgary, Boston. 1982.