# Les treillis de Galois Alpha

**Véronique Ventos*** — **Henry Soldano****,*****

* *LRI, UMR-CNRS 8623, Université Paris-Sud*
*Bat. 490, F-91405 Orsay*
*ventos@lri.fr*

** *L.I.P.N, UMR-CNRS 7030, Université Paris-Nord*
*Av. J-B Clément, F-93430 Villetaneuse*
*soldano@lipn.univ-paris13.fr*

*** *Atelier de BioInformatique, 12 rue Cuvier, Université Paris 6*

RÉSUMÉ. *La représentation générique d'un ensemble de données que nous utilisons ici est un treillis de Galois, c'est-à-dire un treillis correspondant au partitionnement des termes d'un langage en classes d'équivalence relativement à leur extension (l'extension d'un terme est la partie d'un ensemble d'instances qui satisfait ce terme). Pour réduire la taille du treillis, nous proposons ici de simplifier la représentation des données, tout en conservant la structure formelle de treillis de Galois. Pour cela nous utilisons une partition préliminaire des données correspondant à l'association d'un type à chaque instance. En redéfinissant la notion d'extension d'un terme de manière à tenir compte, à un certain degré $\alpha$, de cette partition, nous aboutissons à des treillis de Galois particuliers appelés treillis de Galois Alpha. Nous étudions ici cette nouvelle notion d'extension, la construction directe ou incrémentale et l'ordonnancement de ces treillis ainsi que les règles d'implications associées.*

ABSTRACT. *Our basic representation of the data is a Galois lattice, i.e. a lattice in which the terms of a representation language are partitioned into equivalence classes w.r.t. their extent (the extent of a term is the part of the instance set that satisfies the term). We propose here to simplify our view of the data, still conserving the Galois lattice formal structure. For that purpose we use a preliminary partition of the instance set, representing the association of a* type *to each instance. By redefining the notion of extent of a term in order to cope, to a certain degree (denoted as $\alpha$), with this partition, we define a particular family of Galois lattices denoted as Alpha Galois lattices.*

MOTS-CLÉS : *classification non-supervisée, treillis de Galois, règles d'association.*

KEYWORDS: *conceptual clustering, Galois lattices, association rules.*

RSTI - RIA – 19/2005. Apprentissage automatique, pages 799 à 827

## 1. Introduction

Dans beaucoup d'applications il devient primordial d'aider les utilisateurs à accéder à une grande quantité de données. Une manière de procéder consiste à regrouper les instances en classes, elle-mêmes organisées en une hiérarchie, et décrites à un niveau d'abstraction adéquat. La représentation de départ que nous utilisons ici est un treillis de concepts. Dans un tel treillis chaque nœud correspond à une classe représentée par son *extension* (les instances de la classe) et son *intension* (les propriétés communes à la classe exprimées comme un terme d'un langage de classes). Le treillis de concepts constitue une représentation exhaustive des concepts sous-jacents à l'ensemble des instances par rapport au langage de classes : tout sous-ensemble des instances constituant l'extension d'un terme du langage est associé à un et un seul nœud du treillis. Son principal désavantage est sa taille qui peut être tres grande dans le cas d'applications réelles. Plusieurs techniques ont été proposées pour réduire la taille du treillis en éliminant une partie de ses nœuds (*e.g.* (Hereth *et al.*, 2000)). En particulier un treillis de concepts *fréquents* (Waiyamai *et al.*, 2000) représente la partie supérieure d'un treillis de concepts : seuls les nœuds dont l'extension est suffisament grande (relativement à un seuil) sont représentés. Dans l'approche présentée ici nous contrôlons le nombre de nœuds du treillis en tenant compte, dans une certaine mesure associée à un degré $\alpha$, d'une partition *a priori* des données. Cette partition est constituée d'un ensemble de *classes de base* telles que chaque classe de base regroupe les instances partageant un même *type de base*. Ainsi dans un jeu de données concernant le catalogue électronique de produits informatiques C/Net (http ://www.cnet.com), il y a 59 classes de base correspondant à 59 types de base différents (*e.g. Laptops, Harddrive, NetworkStorage*) et ce pour 2 274 instances. Les classes de base sont utilisées pour ajouter un critère de fréquence *locale* à la notion d'$extension$ de la manière suivante : une instance $i$ appartient à $ext_\alpha(T)$ (la $\alpha$-$extension$ d'un terme $T$ du langage de classes), lorsque, d'une part, elle appartient à l'extension de $T$, $ext(T)$, (*i.e.* $i$ a toutes les propriétés qu'exprime $T$), et d'autre part, au moins $\alpha$ % des instances de la classe de base de $i$ appartiennent aussi à $ext(T)$. Cette nouvelle notion d'$\alpha$-$extension$ induit de nouveaux treillis de concepts, plus flexibles, qui ont formellement la structure de treillis de Galois, et que nous appelons ici *treillis de Galois Alpha* ou plus simplement *treillis Alpha*. Pour en revenir à l'exemple du catalogue electronique, considérons un treillis de concepts fréquents. Même pour seuil de fréquence très bas, la propriété « support »n'apparaît dans aucun nœud car elle n'est pas globalement fréquente (13 produits sur 2274). Dans un treillis Alpha (avec $\alpha$ plus petit que 92) la propriété « support »apparaît dans au moins un terme, car dans la classe de base $HardDrives$ 92 % des instances sont vendues avec un « support ». Autrement dit le treillis de Galois Alpha introduit une notion de fréquence *locale* qui permet de faire apparaître dans le treillis des propriétés qui ne sont fréquentes que dans certaines classes de base.

Nous montrons que pour un même langage de classes et un même ensemble d'instances, le *treillis de Galois Alpha* est plus *grossier* que le treillis de concepts : les nœuds du *treillis de Galois Alpha* constituent un sous-ensemble des nœuds du treillis de concepts. Nous montrons également que les valeurs de $\alpha$ définissent un ordre total

sur les *treillis de Galois Alpha* : le *treillis de Galois Alpha* induit par $ext_{\alpha 1}$ est plus grossier que celui induit par $ext_{\alpha 2}$ si $\alpha 1 \geq \alpha 2$. Cet ordre total permet en particulier de faire des *zooms* successifs sur les *treillis de Galois Alpha* permettant ainsi de tenir compte à la fois des limites de ce que peut appréhender un utilisateur et de ce qui l'intéresse réellement. Cette approche interactive consiste à construire dans un premier temps un treillis grossier puis à raffiner une partie de ce treillis, en faisant décroître la valeur de $\alpha$. Une telle stratégie de zoom/raffinement successifs est implémentée dans le système ZooM (Pernelle *et al.*, 2002) dédié à la réprésentation de données par des treillis à différents niveaux d'abstraction. Le cadre général des treillis de Galois est donné en section 2. En section 3 nous présentons, et illustrons sur un exemple simple, les *treillis de Galois Alpha*. La section 4 presente des expériences, menées sur les données de C/net, qui mettent l'accent sur la robustesse de cette représentation en particulier en présence de données *exceptionnelles* relativement à leur classe de base (avec des valeurs de $\alpha$ proches de 0 ou de 100). La section 5 aborde la question de la construction incrémentale des treillis Alpha. Pour cela nous étendons préalablement l'opérateur de *subposition* des treillis de concepts de manière à pouvoir fusionner, de manière plus générale, des treillis de Galois selon leur dimension extensionnelle. La section 6 traite des règles d'implications particulières associées aux *treillis de Galois Alpha*, de la notion, qui en découle, de règles d'association, et de la construction de bases de telles règles. Enfin la section 7 traite des modifications à apporter aux définitions dans le cas où les classes de bases ne sont pas disjointes, interprète le treillis Alpha, en changeant la nature des instances, pour le replonger dans le cadre de l'*Analyse Formelle de Concepts*, et enfin compare le point de vue ensembliste associé à la notion d' $\alpha$-*extension* avec celui de la théorie des ensembles « rugueux »(rough sets). La conclusion tente alors de relier ce travail à d'autres travaux sur les treillis de concepts et trace quelques perspectives de recherche. Cet article est une version étendue de (Ventos *et al.*, 2004).

## 2. Préliminaires et définitions

Les principales définitions, preuves et résultats concernant les correspondances et treillis de Galois sont présentées dans (Barbu *et al.*, 1970; Birkhoff, 1973). D'autres résultats sur les treillis de Galois redéfinis dans le champ de l'analyse formelle de concepts apparaissent dans (Ganter *et al.*, 1999) et, dans le cadre de l'Analyse des Objets Symboliques, dans (Bock *et al.*, 2000). Nous utilisons ici une présentation plus large que celle de (Ganter *et al.*, 1999) dans la mesure où nous ferons varier par la suite la notion d'*extension*. Dans la suite du papier nous appelons treillis de Galois la structure formelle que nous définissons ci-dessous et réservons le terme *treillis de concepts* aux treillis de Galois présentés dans (Ganter *et al.*, 1999).

**Définition 1 (Ensembles ordonnés et treillis)** *Un ensemble ordonné est un ensemble muni d'une relation d'ordre $\leq$. Un ensemble ordonné (M,$\leq$) est un treillis ssi tout couple d'éléments (x,y) de M a un plus petit majorant (ou supremum) $x \vee y$, et un plus grand minorant (ou infimum) $x \wedge y$.*

**Définition 2 (Correspondance de Galois )** *Soient m1 : P → Q et m2 : Q → P des fonctions définies sur deux ensembles ordonnés (P,$\leq_P$) et (Q,$\leq_Q$). (m1,m2) est une correspondance de Galois si pour tout p, p1, p2 de P et pour tout q, q1, q2 de Q :*

*C1- p1 $\leq_P$ p2 ⇒ m1(p2) $\leq_Q$ m1(p1)*

*C2- q1 $\leq_Q$ q2 ⇒ m2(q2) $\leq_P$ m2(q1)*

*C3- p $\leq_P$ m2(m1(p)) et q $\leq_Q$ m1(m2(q))*

L'exemple ci-dessous sera utilisé pour illustrer les différentes notions presentées en section 2 et 3.

**Exemple 1** *Les deux ensembles ordonnés sont ($\mathcal{L}$,$\preceq$) et ($\mathcal{P}(I)$, $\subseteq$) :*
*- $\mathcal{L}$ est un langage dont un terme est un sous-ensemble d'un ensemble d'attribut $\mathcal{A}$ = {t1, t2, t3,a3,a4,a5,a6,a7,a8}. Ici c1 $\preceq$ c2 signifie que le terme c1 est moins spécifique que le terme c2 (par exemple, {a3,a4} $\preceq$ {a3,a4,a6}),*
*- I est un ensemble d'instances = {i1,i2,i3,i4,i5, i6,i7,i8}.*

*Soient $int$ et $ext$ deux fonctions telles que $int : \mathcal{P}(I) \rightarrow \mathcal{L}$ et $ext : \mathcal{L} \rightarrow \mathcal{P}(I)$ avec :*
*- $int(e1)$ est l' ensemble des attributs communs à toutes les instances de $e1$. $int(e1)$ est appelée l'intension de $e1$.*
*- $ext(c1)$ = {$i \in$ I tels que $i$ $isa$ c1} où $isa$ est l'appartenance usuelle d'une instance à un terme. $ext(c1)$ est donc l'ensemble des instances qui ont tous les attributs de $c1$.*
*L'exemple 1 est décrit en figure 1 : chaque ligne $i$ représente l'intension $int(\{i\})$ d'une instance de $I$ et chaque colonne $j$ représente l'extension $ext(\{j\})$ d'un attribut de $\mathcal{A}$.*
*- $int$ et $ext$ forment une correspondance de Galois sur $\mathcal{L}$ et $\mathcal{P}(I)$.*

| | t1 | t2 | t3 | a3 | a4 | a5 | a6 | a7 | a8 |
|---|---|---|---|---|---|---|---|---|---|
| i1 | 1 | | | 1 | 1 | | 1 | | 1 |
| i2 | 1 | | | 1 | | 1 | 1 | | |
| i3 | | 1 | | | 1 | | 1 | | 1 |
| i4 | | 1 | | | 1 | | 1 | 1 | |
| i5 | | 1 | | 1 | | | 1 | | 1 |
| i6 | | | 1 | 1 | | | 1 | | 1 |
| i7 | | | 1 | 1 | | | 1 | | 1 |
| i8 | | | 1 | 1 | | 1 | 1 | | 1 |

**Figure 1.** *Exemple 1. $Tab(i,j)$ = 1 si le $j^{eme}$ attribut appartient à la $i^{eme}$ instance*

Nous rappelons ci-dessous la définition d'un opérateur de *fermeture*.

**Définition 3 (Fermeture)** *w est un opérateur de fermeture sur un ensemble ordonné (M,$\leq$) ssi pour tout couple (x,y) d'éléments de M on a :*

*- $x \leq w(x)$ (extensivité)*
*- Si $x \leq y$ alors $w(x) \leq w(y)$ (monotonie)*
*- $w(x) = w(w(x))$ (idempotence)*
*Un élement x de M tel que x=w(x) est appelé un terme fermé de M relativement à w.*

Dans une correspondance de Galois, $m1 \circ m2$ et $m2 \circ m1$ sont des opérateurs de fermeture pour $P$ et $Q$.
**Exemple** : *Dans l'exemple 1, $ext(\{a4\}) = \{i1, i3, i4\}$, $int(\{i1, i3, i4\}) = \{a4, a6\}$. Le terme $\{a4, a6\}$ est fermé car $int(ext(\{a4\})) = \{a4, a6\}$.*

**Définition 4 (Treillis de Galois)** *Soient m1 : P → Q et m2 : Q → P deux fonctions définies sur les treillis (P,$\leq_P$) et (Q,$\leq_Q$), telles que (m1,m2) est une correspondance de Galois.*
*Soit G={ (p,q), où p est un élément de P et q un élément de Q, tels que p=m2(q) et q = m1(p)}.*
*Soit $\leq$ la relation définie par : (p1,q1) $\leq$ (p2,q2) ssi q1 $\leq_Q$ q2.*
*(G,$\leq$) est un treillis appelé treillis de Galois. Nous utiliserons si nécessaire la notation complète $G(P, m1, Q, m2)$.*
**Exemple** *: Dans l'exemple 1, G={(c,e) où $c$ appartient à $\mathcal{L}$, et $e$ appartient à $\mathcal{P}(\mathcal{I})$ et tels que e=ext(c) et c=int(e)}. (G, $\leq$) est un treillis de Galois avec $\leq$ définie par : (c,e) $\leq$ (c',e') ssi e $\subseteq$ e' (ce qui est en fait équivalent à c $\succeq$ c'). Le treillis de Galois correspondant à l'exemple 1 est presenté figure 2.*

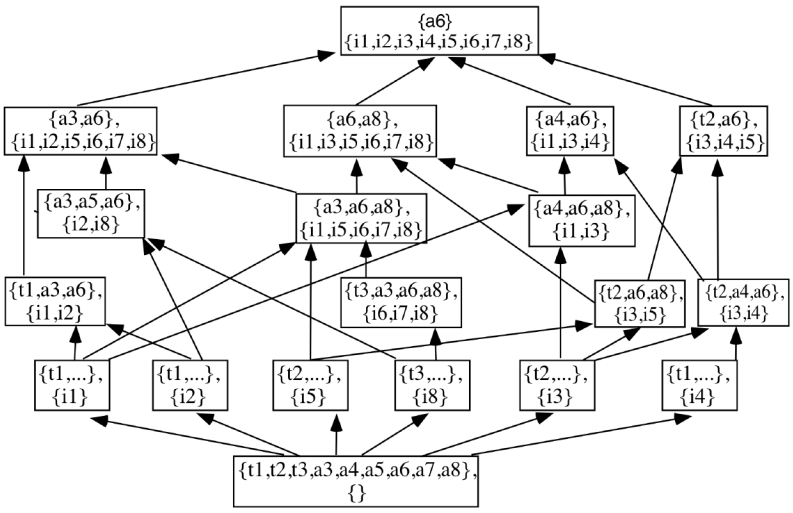


**Figure 2.** *Le treillis de Galois correspondant à l'exemple 1*

Remarquons qu'un nœud d'un treillis de Galois est un couple d'éléments fermés de $P$ et $Q$. De plus les fonctions $m1$ et $m2$ définissent des relations d'équivalence sur les treillis $P$ et $Q$ :

**Définition 5 (Relations d'équivalence sur $P$ et $Q$)** *Soient $\equiv_P$ et $\equiv_Q$ les relations d'équivalence définies sur $P$ et sur $Q$ par $m1$ et $m2$, c'est-à-dire soient p1 et p2 des éléments de $P$ et q1, q2 des élements de $Q$, alors :*

$$p1 \equiv_P p2 \text{ ssi } m1(p1) = m1(p2), \text{ et } q1 \equiv_Q q2 \text{ ssi } m2(q1) = m2(q2)$$

**Lemme 1** *Soient p un élément de P, et q un élément de Q, m2(m1(p)) est l'unique plus grand élément de la classe d'équivalence de $\equiv_P$ contenant p et m1(m2(q)) est l'unique plus grand élément de la classe d'équivalence de $\equiv_Q$ contenant q.*

Ainsi, une propriété caractéristique des treillis de Galois est que chaque nœud $(p, q)$ est constitué de représentants de classes d'équivalence de $\equiv_P$ et de $\equiv_Q$.

Dans notre exemple, nous avons utilisé le langage $\mathcal{L}$, défini comme l'ensemble $\mathcal{P}(\mathcal{A})$ des parties d'un ensemble d'attributs $A$, comme premier treillis, et l'ensemble $\mathcal{P}(\mathcal{I})$ des parties d'un ensemble d'instances $I$ comme second treillis. Ces treillis, associés aux deux fonctions $int$ et $ext$, définissent un treillis de Galois particulier nommé *treillis de concepts* (Ganter *et al.*, 1999). Dans un treillis de concepts, un nœud $(c, e)$ est un concept, $c$ est *l'intension* et $e$ est l'*extension* du concept. Les treillis de concepts sont intéressants à la fois d'un point de vue pratique, dans la mesure où ils expriment d'une manière rigoureuse les deux facettes d'un concept, et d'un point de vue théorique car il a été montré que tout treillis fini est isomorphe à un treillis de concepts (Ganter *et al.*, 1999).

## 3. Treillis de Galois Alpha

Dans ce qui suit nous considérons, sans perte de généralité, $\mathcal{L} = \mathcal{P}(A)$ et prenons comme point de départ le treillis de concepts $G(\mathcal{L}, ext, \mathcal{P}(I), int)$ pris comme exemple ci-dessus. Puis nous présentons une variante de $ext$ dont la relation d'équivalence associée $\equiv'_{\mathcal{L}}$ est plus grossière que celle associée à $ext$ (*cf* la définition 11) ce qui conduit à des classes d'équivalence plus grandes sur $\mathcal{L}$ et donc à un treillis de Galois constitué d'une partie seulement des nœuds du treillis de concepts associé à $ext$.

La nouvelle notion d'extension repose sur l'association d'un type prédéfini à chaque instance. Les instances sont regroupées en *classes de base* correspondant à ces types. La première idée consiste alors à considérer, pour construire le treillis de concepts, les *classes de base* plutôt que les instances (cf (Pernelle *et al.*, 2001)).

Supposons par exemple que les attributs $t1, t2, t3$ correspondent aux trois types possibles pour les instances de l'ensemble $I$ de l'exemple 1. A partir de ces types on engendre 3 classes de base $BC1$, $BC2$, $BC3$ dont les descriptions sont les suivantes : BC1={i1,i2}, int(BC1)= {t1,a3,a6} (les attributs communs à $i1$ et $i2$); BC2={i3,i4,i5}, int(BC2)= {t2,a6} ; BC3={i6,i7,i8}, int(BC3)= {t3,a3,a6,a8}.

Il est maintenant possible de construire un treillis de concepts avec un nouvel ensemble de trois instances : {bc1,bc2,bc3} (appelons-les les *prototypes* de leur classe

de base respective) tels que, pour tout index i, $int(BCi)$ = $int(\{bci\})$. Ce treillis de Galois, représenté figure 3, est un cas particulier de treillis de Galois Alpha, plus petit que le treillis de concepts original.

Cependant il semble intéressant de concevoir une approche intermédiaire, dans laquelle on conserverait une influence de la partition des données en classes de base sans pour autant se restreindre, comme ci-dessus, à ne considérer que des réunions de classes de base dans les extensions.

C'est cette approche intermédiaire qui conduit à la définition des *treillis de Galois Alpha*.

### 3.1. *Alpha définitions*

**Définition 6 (Alpha satisfaction)** *Soient $\alpha \in$ [0,100], $e = \{i_1, \ldots, i_n\}$ un ensemble d'instances, et $T$ un terme de $\mathcal{L}$.*

$$e\ \alpha - satisfait\ T\ (e\ sat_\alpha\ T)\ ssi\ \mid ext(T) \cap e \mid \geq \frac{\mid e \mid .\alpha}{100}$$

La $\alpha$-satisfaction d'un terme de $\mathcal{L}$ permet de tester si un pourcentage $\alpha$ % des instances d'un sous-ensemble d'instances satisfait un terme de $\mathcal{L}$. Nous considérons les classes de bases comme sous-ensembles d'instances et ajoutons cette contrainte à la relation d'appartenance $isa$ entre les instances et les termes de $\mathcal{L}$. Nous appelons cette nouvelle notion (appartenance de l'instance à un terme et $\alpha$-satisfaction de ce terme par la classe de base de l'instance) la $\alpha$-*appartenance*.

**Définition 7 (Alpha appartenance)** *Soit $I$ un ensemble d'instances et $\mathcal{BC}$ une partition*[1] *finie de $I$ en classes de base. Soit BCl : $I \rightarrow \mathcal{BC}$ la fonction telle que $BCl(i)$ est la classe de base de l' instance $i$, et soit $T$ un terme de $\mathcal{L}$, alors :*

$$i\ isa_\alpha\ T\ ssi\ i\ isa\ T\ et\ BCl(i)\ sat_\alpha\ T$$

**Exemple (exemple 1)** *: Soit T={a6,a8}, ext(T) = {i1,i3,i5,i6,i7,i8}.*
$BC1\ sat_{50}\ T$ *puisque* $i1\ isa\ T$ *et* $|BC1|= 2$. *En conséquence* $i1\ isa_{50}\ T$.
*D'autre part* $BC2\ sat_{60}\ T$ *puisque* $\mid ext(T) \cap BC2 \mid \geq \frac{|BC2|.60}{100}$. *Ainsi nous avons* $i3$ *et* $i5\ isa_{60}\ T$. *Finalement* $BC3\ sat_{100}\ T$ *puisque 100 % des instances de* $BC3$ *appartiennent à l'extension de* $T$. *Ainsi nous avons* $i6$, $i7$, *et* $i8\ isa_{100}\ T$.

Nous utilisons maintenant la *Alpha appartenance* pour définir la notion d'extension que nous utiliserons dans les treillis de Galois Alpha.

1. Par souci de simplicité, nous imposons ici aux données d'être partitionnées. En réalité les classes peuvent sans inconvénient avoir des intersections non vides, il faut alors légèrement modifier la présente définition (voir section 7.1).

**Définition 8 (Alpha extension d'un terme)** *La $\alpha$-extension d'un terme $T$ dans $I$, relativement à la partition $\mathcal{BC}$, est définie comme suit :*

$$ext_\alpha(T) = \{i \in I \mid i \ isa_\alpha \ T\}$$

**Exemple (exemple 1)** *: Soit T={a6,a8}, $ext_0$(T)= ext(T) = {i1, i3,i5, i6,i7,i8}*
*$ext_{60}$(T)= {i3,i5,i6,i7,i8} et $ext_{100}$(T)= {i6,i7,i8}*

La proposition suivante définit une nouvelle correspondance de Galois à partir des fonctions $int$ et $ext_\alpha$. Elle nécessite de définir un treillis $E_\alpha$ de sous-ensembles de $I$ correspondant à une partie seulement de $\mathcal{P}(\mathcal{I})$. Un élément de $E_\alpha$ est constitué de parties suffisamment grandes de différentes classes de base (c'est-à-dire de parties dont la cardinalité est supérieure ou égale à $\alpha$ % de la classe de base correspondante).

**Proposition 1** *Soit $E_\alpha$ le sous-ensemble de $\mathcal{P}(\mathcal{I})$ défini comme suit :*
$E_\alpha = \{e \in \mathcal{P}(I) \mid \ \forall i \in e, \mid e \cap BCl(i) \mid \geq \frac{|BCl(i)|.\alpha}{100}\}$.
*Alors : $int$ et $ext_\alpha$ définissent une correspondance de Galois entre $\mathcal{L}$ et $E_\alpha$.*
**Preuve** *: La preuve de cette proposition repose sur le 1) du théorème 1 présenté à la section suivante et est en conséquence donnée après l'énoncé de ce théorème, sous la forme de la preuve d'un corollaire.*

Nous pouvons maintenant définir les *treillis de Galois Alpha* associés à cette nouvelle correspondance de Galois.

**Définition 9 (Treillis de Galois Alpha)** *Soit G={ (c,e) où $c$ est un élément de $\mathcal{L}$, $e$ est un élément de $E_\alpha$, e=$ext_\alpha$(c) et c= int(e)}. Le treillis de Galois correspondant $G(\mathcal{L}, ext_\alpha, E_\alpha, int)$ est appelé treillis de Galois Alpha et est noté $G_\alpha$.*

Lorsque $\alpha$ = 0, on a $E_\alpha = \mathcal{P}(I)$ et $ext_\alpha = ext$. Ainsi le treillis de Galois Alpha est dans ce cas le treillis de concepts associé aux mêmes attributs et aux mêmes instances. Lorsque $\alpha$ = 100, l'extension associée à un nœud de ce treillis de Galois Alpha est constituée de classes de base entières. En conséquence le treillis de Galois Alpha est le treillis de concepts obtenu en considérant comme instances les *prototypes* des classes de base.

Le treillis de Galois Alpha $G_{100}$ de l'exemple 1 est représenté figure 3. La figure 4 présente la partie supérieure de $G_{60}$. Notons que les *intensions* des nœuds de $G_{100}$ sont aussi les *intensions* des nœuds de $G_{60}$ qui se projettent sur eux. Les intensions de ces derniers sont également celles des nœuds correspondants du treillis de concepts initial $G_0$ (voir figure 2).

Dans la section suivante nous détaillons comment s'ordonnent les treillis Alpha.

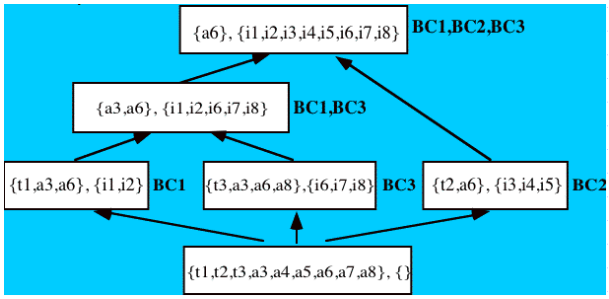


**Figure 3.** *$\alpha$ =100 : le treillis de Galois Alpha $G_{100}$ de l'exemple 1 est plus petit que le treillis de concepts initial présenté figure 2*

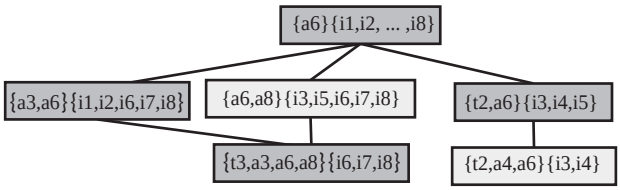


**Figure 4.** *$\alpha$ = 60 : la partie supérieure de $G_{60}$ de l'exemple 1. Les nouveaux nœuds, relativement à $G_{100}$ sont d'un gris plus clair*

### 3.2. *Ordres sur les treillis de Galois Alpha*

Dans (Ganter *et al.*, 2001) les auteurs formalisent l'extension de l'analyse formelle de concepts à des langages plus sophistiqués. Une notion de *projection* est utilisée pour réduire le langage et ainsi réduire la taille du treillis de Galois. Indépendamment, (Pernelle *et al.*, 2002) utilise la même notion de projection pour faire varier le langage mais l'utilise également en la nommant projection *extensionelle*, pour modifier la fonction d'extension *ext*.

Nous rappelons ci-dessous la notion générale de projection, ainsi qu'une relation d'ordre sur les relations d'équivalence :

**Définition 10 (Projection)** *Proj est une projection d'un ensemble ordonné (M,$\leq$) ssi pour tout couple (x,y) d'éléments de M, on a :*
*$x \geq Proj(x)$ (minimalité).*
*if $x \leq y$ then $Proj(x) \leq Proj(y)$ (monotonie).*
*$Proj(x) = Proj(Proj(x))$ (idempotence).*

En changeant *ext*, une projection extensionnelle transforme un treillis de Galois en un treillis plus petit pour lequel la relation d' équivalence $\equiv'_{\mathcal{L}}$ est plus grossière et dont les classes d'équivalence sur $\mathcal{L}$ incluent donc celles de la relation d'origine :

**Définition 11** *Soit $\equiv^1_{\mathcal{L}}$ la relation associée à la fonction ext1 et $\equiv^2_{\mathcal{L}}$ celle associée à ext2, alors, $\equiv^2_{\mathcal{L}}$ est dite plus grossière que $\equiv^1_{\mathcal{L}}$ ssi pour tout couple (c1,c2) d' éléments de $_{\mathcal{L}}$ on a :*
*si ext1(c1) = ext1(c2) alors ext2(c1) = ext2(c2).*

Le théorème suivant est prouvé dans (Pernelle *et al.*, 2002) :

**Théorème 1 (Un ordre extensionnel sur les correspondances de Galois)** *Soient $int$ et $ext$ deux fonctions définissant une correspondance de Galois sur $\mathcal{L}$ et $E$, et soit $proj$ une projection de $E$. Notons $E'$ = $proj$(E) et $ext'$ = $proj \circ ext$. Alors :*

*1) $int$, $ext$' définissent une correspondance de Galois sur $\mathcal{L}$ et $E'$.*

*2) Le treillis de Galois $G'(\mathcal{L}, ext', E', int)$ a la propriété suivante : pour tout nœud g'=(c,e') de $G'$ il existe un nœud g=(c,e) de $G(\mathcal{L}, ext, E, int)$, avec la même $intension$ c, tel que e'=proj(e).*

*3) $\equiv'_{\mathcal{L}}$ est plus grossière que $\equiv_{\mathcal{L}}$.*

*Nous dirons alors que G' est plus grossier extensionnellement que G.*

Le corollaire suivant de ce théorème démontre la proposition 1.

**Corollaire 1** *Soit $G(\mathcal{L}, ext$,P(I)$, int)$ un treillis de Galois. Soit $ext_\alpha$=$proj_\alpha \circ ext$ et $E_\alpha$ = $proj_\alpha(\mathcal{P}(\mathcal{I}))$ avec pour tout e de $\mathcal{P}(\mathcal{I})$ :*

$$proj_\alpha(e) = e - \{i \mid i \in e \text{ et } \mid e \cap BCl(i) \mid < \frac{\mid BCl(i) \mid .\alpha}{100}\}$$

*1) $int$, $ext_\alpha$ définissent une correspondance de Galois sur $\mathcal{L}$ et $E_\alpha$.*
*2) Le treillis de Galois $G'(\mathcal{L}$, $ext_\alpha$, $E_\alpha$, int) est tel que : pour tout nœud g'=(c,e') de $G'$ il existe un nœud g=(c,e) de $G(\mathcal{L}, ext$,P(I)$, int)$, avec la même $intension$ c, tel que e'=$proj_\alpha(e)$*
**Preuve :** *Le corollaire découle directement du théorème dans la mesure où on prouve que $proj_\alpha$ est une projection (on démontre ainsi la proposition 1).*
*$proj_\alpha$(e) est inclus dans e puisqu'on retire des éléments de e, donc $proj_\alpha$ est minimal. Si $e$ est inclus dans $e'$, chaque élément de $e$ retiré par la projection sera aussi retiré de $e'$, $proj_\alpha$(e) est donc inclus dans $proj_\alpha$(e') et $proj_\alpha$ est monotone. Finalement, $proj_\alpha$ est idempotent puisque plus aucun élément de $proj_\alpha$(e) ne peut être retiré par une seconde application de $proj_\alpha$.*

De plus nous pouvons ordonner ces fonctions d'extensions selon la valeur de $\alpha$ : Pour tout couple ($\alpha 1$, $\alpha 2$) tel que $\alpha 1 \leq \alpha 2$, on a $ext_{\alpha 2} = proj_{\alpha} \circ ext_{\alpha 1}$ avec $\alpha = \alpha 2$. En consequence, la valeur de $\alpha$ définit un ordre total sur les treillis de Galois Alpha :

**Proposition 2 (Un ordre total sur les treillis de Galois Alpha)** *Notons $\equiv_{\alpha}$ la relation d' équivalence sur $\mathcal{L}$ associée à $ext_{\alpha}$. Alors pour tout couple ($\alpha 1$, $\alpha 2$) tel que $\alpha 1 \leq \alpha 2$, $\equiv_{\mathcal{L}}^{\alpha 2}$ est plus grossière que $\equiv_{\mathcal{L}}^{\alpha 1}$.*
**Preuve :** *$proj_{\alpha}$ est une projection pour toute valeur de $\alpha$ appartenant à [0,100]. $ext_{\alpha 1}$=$proj_{\alpha} \circ ext$ avec $\alpha = \alpha 1$ et $ext_{\alpha 2}$=$proj_{\alpha} \circ ext_{\alpha 1}$ avec $\alpha = \alpha 2$. Selon le 3) du théorème 1, $\equiv_{\mathcal{L}}^{\alpha 2}$ est alors plus grossière que $\equiv_{\mathcal{L}}^{\alpha 1}$.*

**Exemple :** *$\equiv_{\mathcal{L}}^{100}$ est plus grossière que $\equiv_{\mathcal{L}}^{60}$ qui est elle-même plus grossière que $\equiv_{\mathcal{L}}^{0}$ qui est la relation d'équivalence $\equiv_{\mathcal{L}}$ du treillis de concepts.*

Cette dernière proposition permet de faire des raffinements successifs de treillis de Galois Alpha (voir la section 4).

Il existe aussi un ordre partiel associé à la partition initiale $\mathcal{BC}$ de $I$ en classes de base. Supposons que nous retirions certaines classes de base de $I$, réduisant ainsi l'ensemble d'instances à $I'$ associé à une partition réduite $\mathcal{BC}'$ constituée des classes de base restantes. Il est aisé de montrer (la preuve est ici omise) qu'il y a une projection $proj$ telle que le treillis $E'_{\alpha}$ correspondant s'écrit simplement $proj(E_{\alpha})$. En conséquence nous avons la propriété suivante :

**Proposition 3 (Un ordre partiel sur les treillis de Galois Alpha)** *Soit $\mathcal{BC}'$ un sous-ensemble des classes constituant $\mathcal{BC}$, alors, pour la même valeur $\alpha$, le treillis de Galois Alpha $G'$ associé à $\mathcal{BC}'$ est plus grossier que le treillis de Galois Alpha $G$ associé à $\mathcal{BC}$.*

Un cas particulier intéressant est celui de la partition $\{I\}$ constituée d'une seule classe, c'est-à-dire le cas où toutes les instances partagent le même type. Le treillis de Galois Alpha correspondant est la partie supérieure du treillis de concepts défini par le même langage $\mathcal{L}$ et le même ensemble $I$ d'instances, et est constitué des nœuds dont l'extension est plus grande que $\frac{\alpha}{100}|I|$. Nous retrouvons les structures de *treillis Iceberg* ou *treillis de concepts fréquents* (Stumme *et al.*, 2002; Waiyamai *et al.*, 2000) récemment étudiées. Dans ces structures $\frac{\alpha}{100}$ correspond à la valeur du seuil sur le support *minsupp*.

Notons que du fait de la proposition 3, le treillis fréquent de chaque classe de base $BCi$ d'une partition $\mathcal{BC}$ est toujours plus grossier que le treillis de Galois Alpha correspondant à $\mathcal{BC}$.

Enfin ces deux ordres se combinent pour donner un ordre partiel entre treillis Alpha.

## 4. Implémentation et expériences

Le programme ALPHA qui construit les treillis de Galois Alpha utilise une approche top-down classique dans laquelle on spécialise l'intension $c$ d'un nœud en ajoutant d'abord un nouvel attribut $a$ puis en appliquant l'opérateur de fermeture $int \circ ext_\alpha$ à $c \cup \{a\}$. Chaque nœud ainsi engendré doit être comparé aux nœuds déjà construits (et rangés dans une file) de manière à éviter les doublons.

Nous avons expérimenté ALPHA sur un ensemble de 2 274produits informatiques, partitionnés en 59 types, extraits du catalogue C/Net. Chaque produit peut être décrit à l'aide de 234 attributs. Dans notre première expérience nous avons construit le treillis $G_{100}$ à partir de l'ensemble des 2 274produits (ainsi pratiquement réduit à 59 instances prototypiques), puis nous avons progressivement baissé la valeur de $\alpha$ et calculé les treillis $G_\alpha$ correspondants. Comme on peut le constater ci-dessous le nombre de nœuds (et donc le temps CPU) croît exponentiellement de 211 concepts à 165 369 lorsque $\alpha$ varie de 100 % à 91 %. Il est donc ici clairement impossible d'avoir une vue complète des données au niveau des instances (c'est-à-dire pour $\alpha$=0) et même le relâchement de la contrainte liée aux classes de base (en partant de $\alpha$=100) doit être limité :

| Alpha | 100 | 98 | 96 | 94 | 92 | 91 |
|---|---|---|---|---|---|---|
| Noeuds | 211 | 664 | 8198 | 44021 | 107734 | 165369 |

Notre seconde expérience concerne la partie de $G_{100}$ comprise entre, d'une part, le nœud dont l'extension contient les 3 classes de base (*Laptop* (252 instances pour 39 attributs impliqués dans les descriptions), *Hard-drive*(45 instances, pour 22 attributs impliqués), *Network-storage* (4 instances pour 16 attributs impliqués)) et, d'autre part, le nœud *Bottom* qui ne contient aucune instance. Le nouveau $G_{100}$ contient maintenant 5 nœuds (nombre à comparer au nombre maximum de nœuds $2^3 = 8$). Le treillis $G_\alpha$ est calculé pour toutes les valeurs entières de $\alpha$ entre 100 % et 0 % et sa taille comparée à celle du treillis fréquent correspondant (cf. figure 5).

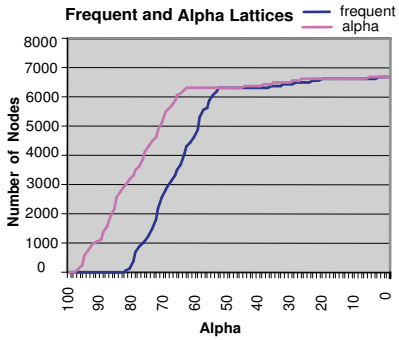


**Figure 5.** *Nombre de nœuds vs Alpha pour les treillis de Galois Alpha et les treillis fréquents*

Nous nous intéressons d'abord aux valeurs élevées de $\alpha$. Partant du treillis $G_{100}$, de nouveaux nœuds apparaissent lorsque $\alpha$ décroît. Ainsi pour $\alpha$ = 99 %, un nouveau nœud apparaît sous le nœud de $G_{100}$ représentant la classe *Laptop*. L'intension de ce nouveau nœud contient maintenant l'attribut « network-card ». Ceci est dû au fait que la plupart des instances de la classe *Laptop*, mais pas toutes, possède une carte réseau. Ainsi en affaiblissant la contrainte sur la classe de base nous évitons les quelques instances exceptionnelles de *Laptop* présentes dans le catalogue et qui masquaient cette propriété « par défaut »de *Laptop* dans le treillis $G_{100}$. De la même manière les instances de *hard-drives* sont vendues avec une maintenance (la propriété *support*). Ainsi pour $\alpha = 92\%$, un nouveau nœud représentant les instances de *hard-drives* possédant la propriété *support* apparaît. Remarquons que dans ce cas l'attribut *support* est rare si l'on considère toutes les instances (*support* apparaît dans 13 produits sur 301) et ne serait pas considéré dans un *treillis de concepts fréquents*, alors qu'il est fréquent dans la classe *hard-drive* (13 produits sur 15) et ainsi apparaît dans le treillis de Galois Alpha $G_{90}$. En résumé, en faisant diminuer $\alpha$ à partir de 100 % on a une vision plus fine des données en tenant compte de propriétés qui sont pertinentes (*i.e.* fréquentes) dans certaines classes de base.

Si on considère maintenant que l'on part du treillis de concepts original et que l'on fait croître lentement $\alpha$ de 0 vers des valeurs faibles (de l'ordre de 10 %), certaines instances, *exceptionnelles* dans leur classe de base relativement à un certain terme $t$ de $\mathcal{L}$, disparaîtront de la $\alpha$-extension de $t$. Ces instances sont exceptionnelles car elles appartiennent à l'extension du terme $t$ alors même que peu d'instances de cette même classe de base appartiennent à cette extension. En conséquence, certaines propriétés très rares dans certaines classes de base, ne seront plus utilisables pour distinguer les concepts. Par exemple, quelques *Laptops* seulement ont la propriété *Digital-Signal-Protocol*, et ainsi pour $\alpha = 6\%$ , les nœuds dont l'intension contient la propriété *Digital-Signal-Protocol* ne contiennent plus d'instances de *Laptop* dans leur extension. Ainsi les termes de la forme {Laptop, Digital-Signal-Protocol, ... , } deviennent équivalents au nœud $Bottom$, alors que d'autres termes incluant *Digital-Signal-Protocol* deviennent équivalents (car leur extensions ne diffèrent que par la présence d'instances de *Laptop*). L'effet résultant est une diminution du nombre de nœuds. Une lecture plus précise de la figure 5 montre qu'il peut y avoir beaucoup de nœuds même pour des valeurs élevées de $\alpha$. Dans cet exemple particulier ceci est dû au fait qu'une classe de base, *Laptop*, a un treillis fréquent qui envahit le treillis Alpha. Une expérience menée sur une partie seulement des données du catalogue (24 classes de base représentant en tout 1 187 objets) obtenues en retirant certaines classes envahissantes, montre que le treillis Alpha résultant est plus détaillé (pour une même valeur $\alpha$) que le treillis fréquent correspondant (voir aussi figure 6) :

| Alpha | 100 | 80 | 50 | 30 | 0 |
|---|---|---|---|---|---|
| Nœuds Alpha | 158 | 842 | 1493 | 1900 | 2202 |
| Nœuds Frequent | 2 | 18 | 18 | 50 | 2202 |

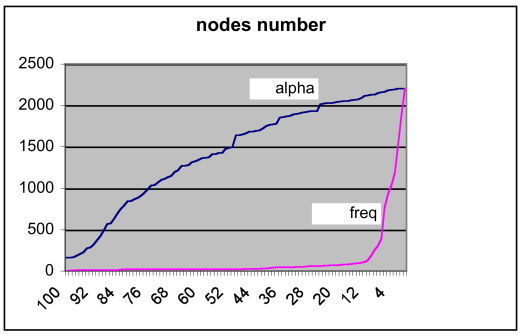


**Figure 6.** *Nombre de nœuds vs Alpha pour les treillis de Galois Alpha et les treillis fréquents*

## 5. Fusion de treillis de Galois et incrémentalité

Nous montrons ici que les treillis de Galois Alpha peuvent être construits en fusionnant les treillis fréquents associés à chaque classe de base. Ce résultat s'appuie sur la propriété 3 sur l'ordre partiel des treillis Alpha et permet une construction incrémentale *par classe de base* du treillis Alpha. Nous décrivons ensuite une stratégie générale de construction incrémentale d'un treillis Alpha s'appuyant sur la mise à jour de treillis fréquents. Nous commençons par définir ci-dessous un opérateur général de fusion extensionnelle que nous instancions pour la fusion des treillis Alpha. Ceci suppose au préalable d'étendre la notion de fusion telle qu'elle apparaît en analyse formelle de concepts ((Ganter *et al.*, 1999)).

### 5.1. *Fusion extensionnelle de treillis de Galois*

L'idée à la base des opérateurs de fusion tels qu'ils sont définis en analyse formelle de concepts est de diviser le *contexte* (voir figure 1) représentant la relation $\mathcal{R}(\mathcal{I}, \mathcal{A})$ entre les instances de $I$ et les attributs de$A$ soit sur les instances (horizontalement), soit sur les attributs (verticalement). Le résultat de la division est la formation de deux sous-contextes. En fusionnant les deux treillis de concepts correspondants on obtient le treillis de concept correspondant au contexte original. Dans le premier cas l'opérateur de *subposition* prend en entrée les deux treillis respectivement associés aux sous-contextes de $\mathcal{R}(I_1, A)$ et $\mathcal{R}(I_2, A)$ et donne en sortie le treillis de concept correspondant à $\mathcal{R}(I_1 \cup I_2, A)$. Le résultat est appellé *semi-produit* dans (Ganter *et al.*, 1999) et le passage d'un couple de treillis de concepts à leur semi-produit a été étudié en détail dans (Valtchev *et al.*, 2002b).

Cependant de notre point de vue, diviser un contexte sur les instances revient à appliquer deux projections extensionnelles particulières à $\mathcal{P}(\mathcal{I})$ conduisant aux deux ensembles de parties $\mathcal{P}(I_1)$ et $\mathcal{P}(I_2)$. Cela signifie que, dans certaines conditions, à

partir de deux treillis de Galois projetés on peut construire le treillis de Galois de départ. Repartant alors des mêmes fondements formels que (Ganter *et al.*, 1999; Valtchev *et al.*, 2002b), nous étendons ci-dessous la notion de semi-produit à toute forme de projection extensionnelle.

**Proposition 4** *Soit $G(\mathcal{L}, ext, E, int)$ un treillis de Galois et soit $Proj_1$ et $Proj_2$ deux projections extensionnelles sur $E$ telles que :*

*- Pour tout élément $e$ de $E$, on a $e = Proj_1(e) \vee_E Proj_2(e)$.*

*Soient $E_1 = Proj_1$ (E) et $E_2 = Proj_2$ (E) et soient $ext_1 = Proj_1 \circ ext$ et $ext_2 = Proj_2 \circ ext$. Nous avons alors :*

*a) Tout terme $c$ de $\mathcal{L}$, fermé relativement à. $int \circ ext$, s'écrit $c_1 \wedge_{\mathcal{L}} c_2$ où $c_1 = int \circ ext_1(c)$ et $c_2 = int \circ ext_2(c)$, et est tel que $ext(c) = ext_1(c_1) \vee_E ext_2(c_2)$.*

*b) Tout couple de termes $(c'_1, c'_2)$, tels que $c'_1$ est fermé relativement à $int \circ ext_1$ et $c'_2$ est fermé relativement à $int \circ ext_2$, est tel que :*

*b-1) $c'_1 \wedge_{\mathcal{L}} c'_2$ est fermé relativement à $int \circ ext$ et*

*b-2) $ext_1(c'_1) \vee_E ext_2(c'_2)$ est plus petit que $ext(c)$*

**Preuve** *:*

*a) Nous montrons d'abord qu'un terme $c_1$ fermé relativement à $int \circ ext_1$ est aussi fermé relativement à $int \circ ext$. En premier lieu $ext_1(c_1) = Proj_1 \circ ext(c_1) \subseteq ext(c_1)$ car $Proj_1$ est une projection. La condition $C_2$ de la définition de la correspondance de Galois impose $int(ext(c_1)) \leq_{\mathcal{L}} int(ext_1(c_1)) = c_1$. Puisque $int \circ ext$ est un opérateur de fermeture nous avons également $c_1 \leq_{\mathcal{L}} int(ext(c_1))$. Donc $c_1 = int(ext(c_1))$, c-à-d. $c_1$ est fermé relativement à $int \circ ext$. De la même manière un terme $c_2$ fermé relativement à $int \circ ext_2$ est aussi fermé relativement à $int \circ ext$. Utilisant alors une propriété bien connue des treillis de Galois, nous en concluons que $c_1 \wedge_{\mathcal{L}} c_2$ est fermé relativement à $int \circ ext$.*
*Pour tout terme $c$, nous savons que $e = ext(c)$ et si nous utilisons la première condition de la proposition, nous obtenons $ext(c) = ext_1(c) \vee_E ext_2(c)$. D'autre part d'après les propriétés de la correspondance de Galois, nous avons $ext_1(c) = ext_1(int \circ ext_1(c)) = ext_1(c_1)$ et $ext_2(c) = ext_2(int \circ ext_2(c)) = ext_2(c_2)$. En conclusion $ext(c) = ext_1(c_1) \vee_E ext_2(c_2)$.*

*b) Soit $c = c'_1 \wedge_{\mathcal{L}} c'_2$, nous avons vu que $c$ est fermé relativement à $int \circ ext$. Nous avons également $c \leq_{\mathcal{L}} c'_1$ et par conséquent $ext_1(c'_1) \subseteq ext_1(c)$ et donc $ext_1(c'_1) \subseteq ext(c)$. De la même manière nous obtenons $ext_2(c'_2) \subseteq ext(c)$. En conclusion $ext_1(c'_1) \vee_E ext_2(c'_2) \subseteq ext(c)$.*

De cette proposition on conclut que les intensions des nœuds $(c, e)$ du treillis de Galois $G(\mathcal{L}, ext, E, int)$ sont les conjonctions des intensions des nœuds $(c_1, e_1)$ du treillis de Galois projeté $Proj_1(G) = G(\mathcal{L}, ext_1, E_1, int)$ avec les intensions des nœuds $(c_2, e_2)$ de $Proj_2(G) = G(\mathcal{L}, ext_2, E_2, int)$. Plusieurs des intensions ainsi ob-

tenues pouvant être alors identiques, un seul nœud sera alors créé, et l'extension correspondante sera la plus grande obtenue en faisant la disjonction des extensions $e_1$ et $e_2$. Nous noterons $Fusion_E$ l'opérateur ainsi obtenu :

**Définition 12 (Un opérateur de fusion extensionnelle)** *Soit $G(\mathcal{L},\ ext,\ E, int)$ un treillis de Galois et soient $Proj_1$ et $Proj_2$ deux projections extensionelles sur $E$ telles que pour tout élément $e$ de $E$, on a $e$ = $Proj_1$(e) $\vee_E$ $Proj_2$(e). L'operateur $Fusion_E$ défini selon la proposition 4 est alors tel que :*

$G(\mathcal{L}, ext, E, int)$ = $Proj_1(G)$ $Fusion_E$ $Proj_2(G)$

De la même manière on peut définir un opérateur de *fusion intensionnelle* en définissant deux projections $P_1$ and $P_2$ sur le langage $\mathcal{L}$ telles que pour tout $c$ de $\mathcal{L}$ on a $c$= $P_1(c)$ $\vee_{\mathcal{L}}$ $P_2(c)$. Dans le cas des treillis de concepts l'opérateur correspondant est l'opérateur d'*apposition* qui est appliqué sur les treillis de concepts obtenus en divisant l'ensemble des attributs $A$ en deux sous-ensembles $A_1$ et $A_2$.

### 5.2. *Incrémentalité* par classe *: la fusion de treillis de Galois Alpha*

Nous montrons ici que si un ensemble d'instances $I$ est séparé en deux sous-ensembles de classes de base $\mathcal{BC}^1$ et $\mathcal{BC}^2$, alors la fusion extensionnelle des treillis Alpha correspondant aboutit au treillis Alpha associé à $\mathcal{BC}$= $\mathcal{BC}^1 \cup \mathcal{BC}^2$.

Soit $E_\alpha$ le sous-ensemble de $\mathcal{P}(\mathcal{I})$, associé à l'ensemble de classes de bases $\mathcal{BC}$, comme défini dans la proposition 1, et soient $Proj^1_\alpha$ et $Proj^2_\alpha$ définies comme suit :

$Proj^1_\alpha(e)$ = $\{i/\ i\ \in e\ et\ BCl(i)\ \in \mathcal{BC}^1\}$, et

$Proj^2_\alpha(e)$ = $\{i/\ i\ \in e\ et\ BCl(i)\ \in \mathcal{BC}^2\}$.

$Proj^1_\alpha$ et $Proj^2_\alpha$ sont alors des projections sur $E_\alpha$, et pour tout $e$ de $E_\alpha$nous avons :

e = $Proj^1_\alpha(e) \cup Proj^1_\alpha(e)$

En conséquence nous avons la propriété suivante :

**Proposition 5** *Le treillis Alpha $G_\alpha(\mathcal{L}, ext_\alpha, E_\alpha,$ int) associé à l'ensemble de classes de bases $\mathcal{BC}$= $\mathcal{BC}^1 \cup \mathcal{BC}^2$ est tel que :*

$G_\alpha(\mathcal{L}, ext_\alpha, E_\alpha,$ int) =

$G^1_\alpha(\mathcal{L}, ext^1_\alpha, E^1_\alpha,$ int) $Fusion_E$ $G^2_\alpha(\mathcal{L}, ext^2_\alpha, E^2_\alpha,$ int),

*où $G^1_\alpha(\mathcal{L}, ext^1_\alpha, E^1_\alpha,$ int) et $G^2_\alpha(\mathcal{L}, ext^2_\alpha, E^2_\alpha,$ int) représentent les treillis Alpha correspondant respectivement aux ensembles de classes de bases $\mathcal{BC}^1$ et $\mathcal{BC}^2$.*

Nous pouvons calculer ainsi un treillis Alpha pas à pas, en calculant d'abord les treillis fréquents $G^1_\alpha$, ..., $G^n_\alpha$ associés à chaque classe $BC_1$, .., $BC_n$, puis en utilisant répétitivement $Fusion_E$. Le schéma le plus simple pour ce calcul consiste à fusionner dabord $G^1_\alpha$ avec $G^2_\alpha$, puis à fusionner le treillis Alpha résultant avec $G^3_\alpha$ et ainsi de suite. Le principal avantage de ce schéma est l'incrémentalité *par classe*, c'est-à-dire la capacité de mettre à jour le treillis Alpha associé à l'ensemble de classes $\mathcal{BC}$ lorsqu'une nouvelle classe de base $BC_{n+1}$ est ajoutée à $\mathcal{BC}$. La mise à jour consiste alors simplement à calculer le treillis fréquent $G^{n+1}_\alpha$ puis à calculer $G_\alpha \; Fusion_E \; G^{n+1}_\alpha$.

D'un point de vue algorithmique la proposition 4 conduit à un algorithme générique identique à l'algorithme *Merge* pour la subposition décrit dans (Valtchev *et al.*, 2002b). En particulier le test de *canonicité*, qui évite les doublons lorsque l'on fait l'intersection des intensions, peut être exécuté de la même manière. Toutefois dans le cas des opérateurs de subposition/apposition l'algorithme tire profit de propriétés qui ne sont pas vraies en général, et qui ne le sont pas en particulier dans le cas des treillis Alpha. En effet dans le cas général les projections satisfont la propriété suivante (Pernelle *et al.*, 2002) :

**Proposition 6 (Treillis projetés)** *Soit P un treillis muni des opérateurs supremum $\vee_P$ et infimum $\wedge_P$, et soit $proj_P$ une projection. Dans ce cas :*

*E= $proj_P(P)$ est aussi un treillis avec les opérateurs infimum $\wedge_E$ et supremum $\vee_E$ définis de la manière suivante :*

*p1 $\wedge_E$ p2 = $proj_P$ (p1 $\wedge_P$ p2)*

*p1 $\vee_E$ p2 = p1 $\vee_P$ p2*

Considérons alors $P = \mathcal{P}(I)$ et $E_\alpha = Proj_\alpha(\mathcal{P}(I))$, et notons $\wedge_\alpha$ l'opérateur infimum sur $E_\alpha$. Dans ce cas, pour tout couple d'élements $(p1, p2)$ de $E_\alpha$ nous avons $p1 \wedge_\alpha p2 = proj_\alpha(p1 \cap p2)$ qui est en général différent de $(p1 \cap p2)$. Cela a comme conséquence que le treillis $E_\alpha$ n'est pas distributif : pour tout triplet $(p1, p2, p3)$ d'éléments de $E_\alpha$, $(p1 \vee_\alpha p2) \wedge_\alpha p3$ est en général différent de $(p1 \wedge_\alpha p3) \cup (p2 \wedge_\alpha p3)$.

5.2.1. *Implémentation et expériences*

L'opérateur $Fusion_\alpha$ est obtenu en partant de l'algorithme de subposition décrit dans (Valtchev *et al.*, 2001). Une description plus détaillée de l'algorithme d'apposition est présentée dans (Valtchev *et al.*, 2002b), et son adaptation à l'opérateur dual de subposition qui nous intéresse ne présente pas de difficulté. Notre implémentation de $Fusion_\alpha$ est une adaptation de celle de l'opérateur de subposition dans le logiciel **Galicia**[2]. En pratique la seule difficulté concerne la procédure de mise à jour des liens dans le treillis lors de l'ajout d'un nœud, *Update-Order*. Cette procédure, décrite dans (Valtchev *et al.*, 2002b), repose sur la distributivité de $\mathcal{P}(I)$. Nous avons donc dû

2. Galicia est un projet Open Source visant à construire une plate-forme Java intégrant un ensemble d'outils et méthodes relevant de l'analyse formelle de concepts ou de l'étude des treillis de Galois : http ://www.iro.umontreal.ca/ galicia/

revenir à une version plus directe et moins efficace de mise à jour des liens dans le cas de $E_\alpha$. La figure 7 représente l'évolution du nombre de nœuds du treillis Alpha en fonction du nombre de classes de bases dans le cas des 24 premières classes des données C/net décrites auparavant. Le nombre d'instances dans chaque classe est donné ci-dessous :

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Classe | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Instances | 16 | 21 | 15 | 45 | 30 | 20 | 7 | 36 |
| – | | | | | | | | |
| Classe | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Instances | 12 | 54 | 12 | 19 | 29 | 45 | 8 | 26 |
| – | | | | | | | | |
| Classe | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| Instances | 17 | 50 | 10 | 23 | 75 | 23 | 28 | 13 |

On remarquera une augmentation significative du nombre de nœuds lorsque la fusion intègre la classe 21 qui est celle dont le treillis fréquent contient le plus de nœuds.

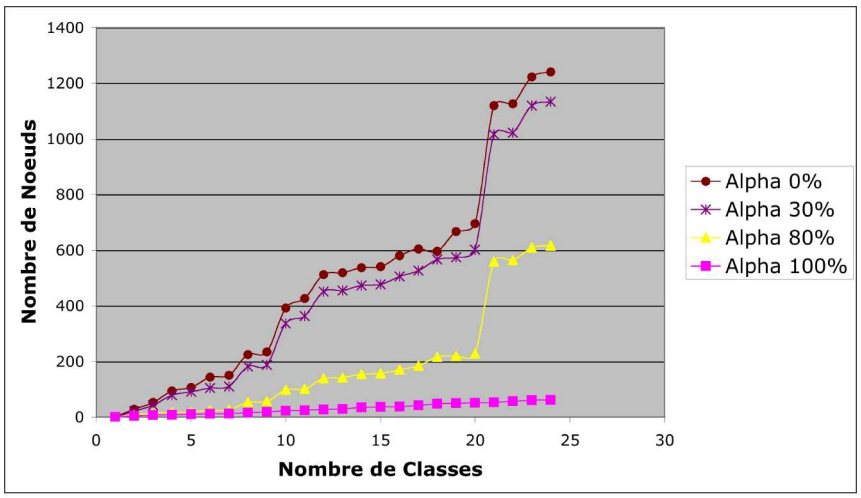


**Figure 7.** *Construction incrémentale d'un treillis Alpha par fusion de 24 treillis fréquents : évolution du nombre total de nœuds lors des fusions successives*

### 5.3. *Incrémentalité générale : mise à jour de treillis Alpha*

Nous avons décrit ci-dessus la construction incrémentale *par classe* des treillis Alpha. Nous nous intéressons ici à l'incrémentalité classique, c'est-à-dire la capacité de

mettre à jour un treillis Alpha lorsqu'arrivent de nouvelles instances appartenant éventuellement à des classes de bases présentes dans $\mathcal{BC}$. Nous allons décrire un protocole permettant d'effectuer cette mise à jour.

Remarquons tout d'abord que la mise à jour d'un treillis fréquent en présence de nouvelles instances est un problème délicat. Récemment (Valtchev *et al.*, 2002a) a proposé une méthode de mise à jour d'un treillis fréquent lorsqu'une nouvelle instance apparaît. Dans ce cas la mise à jour d'un treillis fréquent, en présence d'un ensemble de nouvelles instances, est effectuée en ajoutant ces instances une à une. Dans ce qui suit nous supposerons ainsi que nous disposons d'un opérateur $UpdateFrequent(G_{minsupp}, I_U)$ qui met à jour le treillis frequent $G_{minsupp}$ pour tenir compte d'un ensemble $I_U$ de nouvelles instances.

Soient $G_\alpha$ le treillis Alpha associé à $\mathcal{BC}$ = $BC_1$, .., $BC_n$, et $I_U$ l'ensemble des nouvelles instances divisé en sous-ensembles contenant des instances appartenant à des classes existantes de $\mathcal{BC}$, et en un ensemble $\mathcal{BC}_U$ de nouvelles classes de bases $BC_{n+1}$, .., $BC_m$. Nous avons ainsi $I_U = I_{U1} \cup I_{U2} \cup ... I_{Un} \cup I_{Un+1} \cup .... I_{Um}$. Le nouvel ensemble d'instances $I'$ = $I \cup I_U$ est maintenant partitionné en $\mathcal{BC}'$ = $\mathcal{BC} \cup \mathcal{BC}_U$ dans lequel les anciennes classes $BC_j$ sont mises à jour en $BC_j \cup I_{Uj}$, et les nouvelles classes $BC_k$ sont intialisées à $I_{Uk}$.

Nous voulons ici mettre à jour $G_\alpha$ pour construire le treillis Alpha $G'_\alpha$ associé au nouvel ensemble $I'$ et à sa partition $\mathcal{BC}'$. Notre stratégie est la suivante :

– $G_\alpha$ est projeté sur chaque treillis fréquent $G^j_\alpha$ à mettre à jour, c'est-à-dire tel que $I_{Uj}$ est non vide. L'ensemble des classes de base à mettre à jour est noté $\mathcal{BC}^1$, et l'ensemble des classes inchangées est noté $\mathcal{BC}^0$= $\mathcal{BC} - \mathcal{BC}^1$.

– Chaque $G^j_\alpha$ est mis à jour : $G'^j_\alpha$= $UpdateFrequent(G^j_\alpha, I_{Uj})$.

– Le treillis Alpha mis à jour $G^1_\alpha$ associé à $\mathcal{BC}^1$ est calculé en fusionnant les treillis fréquents mis à jour $G'^j_\alpha$.

– Le treillis Alpha $G^0_\alpha$ associé à $\mathcal{BC}^0$ est calculé en projetant $G_\alpha$.

– Le treillis Alpha $G^U_\alpha$ associé à $\mathcal{BC}_U$ est calculé (directement ou en utilisant l'incrémentalité *par classe*).

– Le treillis Alpha mis à jour $G'_\alpha$ associé à $\mathcal{BC}'$ est calculé en fusionnant $G^0_\alpha$, $G^1_\alpha$ et $G^U_\alpha$.

## 6. Treillis Alpha fréquents et règles d'associations

### 6.1. *Combiner les contraintes locales et globale de fréquence : les treillis Alpha fréquents*

Le treillis fréquent est obtenu en appliquant une contrainte globale de fréquence à un treillis de concepts : les nœuds dont l'extension est trop petite sont éliminés (c'est-à-dire projetés sur le nœud *Bottom*). Lorsque le seuil d'élimination est trop haut cela tend malheureusement à éliminer des attributs et des concepts qui seraient intéressants

pour certaines instances. D'un autre côté, comme nous l'avons vu ci-dessus, le treillis Alpha est obtenu en appliquant une contrainte de fréquence qui est locale à chaque classe, c'est-à-dire telle que les attributs pertinents pour au moins une classe apparaissent. En contrepartie un treillis Alpha peut être très grand en particulier pour de faibles valeurs de $alpha$. Nous proposons ici de combiner ces deux contraintes : nous ne considérerons que les nœuds dont la $\alpha$-*extension* est assez grande. Appliquer cette contrainte globale permet d'éliminer des nœuds qui sont localement fréquents pour certaines classes mais représentent trop peu d'instances et peuvent être ainsi écartés lorsque nous voulons simplifier notre vision des données.

Le résultat d'un tel filtrage est encore un treillis de Galois. Plus précisément, considérons la fonction $proj^f$ définie sur $E_\alpha$ et telle que $proj^f(e) = e$ lorsque $\frac{|e|}{|I|} \geq f$ et $proj^f(e) = \emptyset$ sinon. $proj^f$ est une projection sur $E_\alpha$, et donc $G_\alpha^f = proj^f(G_\alpha)$ est un treillis de Galois plus grossier que $G_\alpha$. Plus précisément il correspond à la partie supérieure de $G_\alpha$ à laquelle on ajoute un nœud *Bottom*.

Nous dirons que $G_\alpha^f$ est un *treillis Alpha fréquent* associé à l'ensemble d'instances $I$, à la partition $\mathcal{BC}$ de $I$, à la valeur Alpha $\alpha$ et au seuil de fréquence $f$. Les règles d' implication correspondantes (voir section suivante) ont un support plus grand que $f$. Notons qu'il faut ici parler d'un $\alpha$-support puisque le support est calculé sur la $\alpha$-extension.

### 6.2. *Règles d'implication et d'association Alpha*

Les règles d'association, dans le domaine de la fouille de données, sont des implications dont la valeur de vérité est observée sur un ensemble d'instances $I$. Chaque règle est associée à une valeur de *support* (la fréquence de sa partie prémisse dans $I$), et à une valeur de *confiance*. Lorsque la confiance est fixée à 1, les règles sont dites *règles d'implication*. Lorsque l'on considère les treillis de concepts, l'ordre partiel induit sur les termes du langage par la correspondance de Galois, correspond à un ensemble de règles d'implication. Plus précisément, $T_2 \preceq_I T_1$ signifie que $ext_I(T_1) \subseteq ext_I(T_2)$ *i.e.* l'implication logique $T_1 \rightarrow T_2$ est vraie sur $I$. Pour une telle règle, on appelera $T_1$ la partie gauche (ou prémisse) et $T_2$ la partie droite (ou conclusion) de la règle.

Les règles d'association correspondant à un ensemble d'instances $I$ sont souvent construites en deux étapes : on construit le treillis de concepts associé à $I$, puis on extrait du treillis les règles d'association. Plus précisément le nombre de règles pouvant être très grand on cherche à extraire une *base* de règles à partir de laquelle toutes les règles peuvent être dérivées. En ce qui concerne les règles d'implication, l'idée générale est qu'un nœud d'un treillis de concepts correspond à une classe d'équivalence de termes partageant tous la même extension. L'intension du nœud, c'est-à-dire son plus grand élément (le plus spécifique), a donc la même extension que les termes les plus petits (les plus généraux) de la classe d'équivalence appelés aussi *générateurs*. En prenant comme partie gauche un terme générateur et comme partie droite un terme fermé on obtient un ensemble de règles d'implications, et l'ensemble

de ces règles forme la base *génératrice* de l'ensemble de toutes les règles d'implications. Pour rendre les règles plus lisibles, la partie droite de la règle est souvent retirée de la partie gauche. Par exemple, considérons le nœud $(abc, \{i1, i2, i3\})$ et supposons que les plus petits termes de la classe d'équivalence sont $\{a, b\}$, avec $ext_I(a)$= $ext_I(b)$ = $\{i1, i2, i3\}$. On obtient alors les règles d'implication $\{a \rightarrow abc, b \rightarrow abc\}$ qui se réécrivent $\{a \rightarrow bc, b \rightarrow ac\}$. Une base de règles d'association non exactes (dites *informatives*) est obtenue en prenant comme partie gauche un générateur et comme partie droite un terme fermé plus spécifique que le terme fermé associé au générateur (c'est-à-dire qu'une règle non exacte relie deux nœuds du treillis). La réduction transitive de cette base est obtenue en se limitant aux règles reliant deux nœuds en relation directe père-fils. Nous renvoyons à (Bastide *et al.*, 2002) pour une présentation plus détaillée des règles d'association et des bases associées.

Dans les treillis fréquents, l'extension d'un terme est redéfinie comme vide lorsque le terme n'est pas fréquent dans $I$, c'est-à-dire lorsque son extension contient moins de $minsupp * |I|$ instances de $I$. Par conséquent, les règles d'implication extraites des nœuds restants ont toutes un support plus grand que $minsupp$.

En ce qui concerne les treillis de Galois Alpha, $T_2 \preceq_\alpha T_1$ signifie que $ext_\alpha(T_1) \subseteq ext_\alpha(T_2)$ et nous dirons que la $\alpha\text{-}implication$ $T_1 \rightarrow_\alpha T_2$ est vraie sur le couple $(I, \mathcal{BC})$. Ces règles dérivant d'un treillis de Galois, c'est-à-dire correspondant à l'inclusion de l'extension de la partie gauche dans l'extension de la partie droite, les $\alpha\text{-}implications$ ont les propriétés suivantes :

-Si $T_1 \rightarrow_\alpha T_2$ et $T_2 \rightarrow_\alpha T_3$, alors $T_1 \rightarrow_\alpha T_3$ (Transitivité)

-SI $T_1 \rightarrow_\alpha T_2$ , et $T_1 \preceq$ T, alors $T \rightarrow_\alpha T_2$ (Monotonie)

-SI $T_1 \rightarrow_\alpha T_2$ et $T_1' \rightarrow_\alpha T_2'$, alors $T_1 \cup T_1' \rightarrow_\alpha T_2 \cup T_2'$ (Additivité)

De plus on dispose du *modus ponens* en tant que règle d'inférence :

Si $i\ isa_\alpha\ T_1$ et $T_1 \rightarrow_\alpha T_2$, alors $i\ isa_\alpha\ T_2$

La propriété d'additivité au niveau des instances s'écrit alors comme suit :

Si $i\ isa_\alpha\ T_1$ et $i\ isa_\alpha T_2$ et $BCl(i) sat_\alpha T_1 \cup T_2$ alors $i\ isa_\alpha T_1 \cup T_2$.

Ces propriétés permettent la construction de règles d'association Alpha de la même manière que les règles d'association classiques et de définir les bases de règles correspondantes, comme les bases génératrices et informatives, ainsi que la réduction transitive de cette dernière.

Nous adaptons ci-dessous les notions de support et de confiance et nous définissons ainsi ce que nous appellerons les règles d'association Alpha :

**Définition 13** *Une règle d'$\alpha$-association (ou règle d'association Alpha) est un couple de termes $T_1$ et $T_2$, noté $T_1 \rightarrow_\alpha T_2$.*

*Le support et la confiance d'une règle d'association Alpha* $r = T_1 \rightarrow_\alpha T_2$ *sont définis comme suit :*

$\alpha$-*supp(r)* = $\frac{|ext_\alpha(T_1 \cup T_2)|}{|I|}$

$\alpha$-*conf(r)* = $\frac{|ext_\alpha(T_1 \cup T_2)|}{|ext_\alpha(T_1)|}$

*La règle d'association Alpha* $r = T_1 \rightarrow_\alpha T_2$ *est valide sur le couple* $(I, \mathcal{BC})$ *lorsque* $\alpha$-$supp(r) \geq minsupp$ *et* $\alpha$-$conf(r) \geq minconf$.

Remarquons que lorsque nous extrayons les règles des bases génératrices ou informatives, la partie droite $T_2$ de la règle est un terme fermé et contient donc toujours la partie gauche de la règle (nous n'utilisons pas ici la simplification évoquée en début de section). En conséquence nous avons $T_1 \cup T_2 = T_2$ et le $\alpha$-support s'écrit $\frac{|ext_\alpha(T_2)|}{|I|}$. Ceci signifie que l'ensemble des règles dont le $\alpha$-support est plus grand que *minsupp* est extrait des nœuds du treillis Alpha fréquent $G_\alpha^{minsupp}$ évoqué à la section précédente.

Dans ce qui suit nous illustrons sur notre exemple initial les règles Alpha. Plus précisément nous construirons la base $\alpha$-GDL, plus compacte que les bases génératrices et informatives évoquées ci-dessus. En effet une partie de la base génératrice forme la base de Guigues-Duquenne des règles d'implications associées à $I$ (Guigues *et al.*, 1986) qui tire partie de la propriété d'additivité mentionnée plus haut pour réduire le nombre de règles. Cette base a été étendue aux règles d'implication ayant un support minimal $minsupp$. De même la base de Luxemberger des règles d'association non exactes de confiance plus grande que $minconf$ (et constituée de règles dont l'antécédent et le conséquent sont les termes fermés de deux nœuds en relation père-fils dans le treillis) a été étendue aux règles de support minimal $minsupp$. Ces deux bases sont calculées à partir des termes fermés (Pasquier *et al.*, 1999; Stumme *et al.*, 2001). La réunion de ces deux bases (GD et L) est la base GDL de règles d'associations. Nous noterons en conséquence base $\alpha$-GDL la base correspondante pour les règles d'association Alpha.

Les algorithmes proposés par (Pasquier *et al.*, 1999; Stumme *et al.*, 2001; Diatta, 2003) construisant la base GDL s'adaptent aisément à la construction de la base $\alpha$-GDL. Nous avons adapté l'algorithme de J. Diatta (Diatta, 2003)[3]. Plus précisément nous sommes partis de l'implémentation incluse dans le projet **Galicia**. L'adaptation consiste essentiellement à redéfinir les notions d'extension, de support et de confiance comme indiqué précédemment. Nous donnons ci-dessous le nombre de règles des bases $\alpha$-GDL pour l'exemple initial (voir figure 1) et pour les 4 valeurs de $\alpha$ conduisant à des treillis différents :

3. Une coquille s'est glissée dans cet article, voici la correction par l'auteur : « les éléments de $ERP_k$ (page 289 de mon papier, actes CAp03) doivent être choisis dans $FI_k$ (k-motifs fréquents) au lieu de $CG_k$ (k-motifs candidats générateurs) »

| Alpha | 0 | 40 | 60 | 100 |
|---|---|---|---|---|
| – | | | | |
| Règles GD | 9 | 7 | 5 | 4 |
| Règles L | 19 | 10 | 6 | 2 |

Voici la base $\alpha$-GDL pour $\alpha$ = 100 et $\alpha$ = 60 avec $minsupp$ = 0,1 (voir aussi figure 3 le treillis correspondant).

Alpha =100
Règles exactes GD

| | | | |
|---|---|---|---|
| R0 : | → a6 | Supp = 1,0 | Conf = 1,0 |
| R1 : | t1 → a3 | Supp = 0.25 | Conf = 1,0 |
| R2 : | t3 → a3 a8 | Supp = 0.37 | Conf = 1,0 |
| R3 : | a8 → t3 a3 | Supp = 0.75 | Conf = 1,0 |

Règles approximatives L

| | | | |
|---|---|---|---|
| R4 : | a6 → a3 | Supp = 0.62 | Conf = 0.62 |
| R5 : | a3 a6 → t3 a8 | Supp = 0.37 | Conf = 0.6 |

–
Alpha = 60
Règles exactes GD

| | | | |
|---|---|---|---|
| R0 : | → a6 | Supp = 1,0 | Conf = 1,0 |
| R1 : | t1 → a3 | Supp = 0.25 | Conf = 1,0 |
| R2 : | t3 → a3 a8 | Supp = 0.37 | Conf = 1,0 |
| R3 : | a4 → t2 | Supp = 0.37 | Conf = 1,0 |
| R4 : | a3 a8 → t3 | Supp = 0.62 | Conf = 1,0 |

Règles approximatives L

| | | | |
|---|---|---|---|
| R5 : | a6 → a3 | Supp = 0.62 | Conf = 0.62 |
| R6 : | a6 → a8 | Supp = 0.62 | Conf = 0.62 |
| R7 : | t2 a6 → a4 | Supp = 0.25 | Conf = 0.66 |
| R8 : | t2 a6 → a8 | Supp = 0.25 | Conf = 0.66 |
| R9 : | a3 a6 → t3 a8 | Supp = 0.37 | Conf = 0.6 |
| R10 : | a6 a8 → t3 a3 | Supp = 0.37 | Conf = 0.6 |

On remarque que le nombre de règles diminue lorsque l'on rend plus grossière la perception des données. Cependant bien que ce soit en général le cas, ce n'est pas formellement toujours vrai. D'une manière générale il y a même là un paradoxe dans le sens où le nombre total d'$\alpha$-implications ne peut qu'augmenter avec la valeur de $\alpha$. En effet le nombre de termes du langage ordonnés par l'inclusion de leur $\alpha$-extension augmente puisque la relation d'équivalence entre termes du langage devient plus grossière lorsque $alpha$ augmente : lorsque deux termes $c1$ et $c2$ deviennent équivalents, les règles $c1 \rightarrow_\alpha c2$ et $c2 \rightarrow_\alpha c1$ deviennent valides sur le couple $(I, \mathcal{BC})$. Tout se passe donc comme si en pratique la taille des bases de règles avait tendance à diminuer lorsque $\alpha$ augmente, alors que le nombre total d'implications, lui augmente. Nous avons le résultat suivant :

**Proposition 7** *Le nombre de règles Alpha génératrices, si on inclut celles où le terme générateur est égal au terme fermé, diminue lorsque la valeur de $\alpha$ augmente.*
**Preuve** *: la propriété est vraie pour toute projection extensionnelle. Ici, lorsque $\alpha$ augmente, la relation d'équivalence entre termes devient plus grossière et des classes sont réunies en une seule. Les générateurs de la nouvelle classe sont ceux des classes constituantes moins ceux qui ne sont plus minimaux, or chaque règle Alpha génératrice est associée à un générateur. Le nombre de règles génératrices diminue donc. Cependant il peut se produire que deux classes dont chaque unique générateur est identique au terme fermé de la classe, en se réunissant, donnent naissance à une règle génératrice non triviale (c'est-à-dire dont l'antécédent et le conséquent sont différents). Le résultat ne tient donc pas si on omet ces règles triviales du dénombrement.*

Illustrons cette décroissance sur l'exemple ci-dessus : considérons la règle $R2$ pour $\alpha = 60$, $a4 \rightarrow t2$. Celle-ci sous sa forme « générateur → fermé »s'écrit $a4 \rightarrow t2a4a6$. Pour $\alpha = 0$ l'implication est fausse. Elle apparaît pour $\alpha = 60$ car deux nœuds du treillis $G_0$ (voir figure 2) se réunissent et les deux implications associées $t2a4 \rightarrow t2a4a6$ et $a4 \rightarrow a4a6$ ne sont plus génératrices, au bénéfice de $a4 \rightarrow t2a4a6$ (en effet $t2a4$ n'est plus générateur et $a4a6$ n'est plus fermé). En ce qui concerne $\alpha = 100$ l'implication est toujours vraie mais n'est plus génératrice, et son support est maintenant nul. En effet $ext_{100}(a4) = 0$ (voir le tableau en figure 1).

Nous n'avons pas la même monotonie concernant les règles non exactes. Ceci est dû au fait que la $\alpha$-confiance d'une règle peut diminuer ou augmenter lorsque la valeur de $\alpha$ augmente. Aussi même si le nombre de nœuds diminue lorsque $\alpha$ augmente, c'est-à-dire que le nombre de termes fermés diminue, le nombre des règles de confiance supérieure à $minconf$ peut augmenter.

Nous voudrions maintenant illustrer sur un exemple ce en quoi les $\alpha$-implications permettent d'exprimer une notion d'exception lorsque les instances sont étiquetées par des types.

Pour cela supposons que nous séparions des espèces animales (chacune correspondant à une instance) en les classes de base suivantes : *mammifères, oiseaux, insectes*. Nous cherchons à extraire des règles générales de ces données. Une règle intuitive est par exemple : « un animal qui *vole* devrait avoir des *ailes* ». Cette implication est vraie aussi bien pour les oiseaux (les oiseaux coureurs, comme l'autruche, ne contredisent pas la règle) que pour les insectes. Pour être universelle, la règle devrait aussi être vraie pour les mammifères, qui en général ne volent pas, mais est contredite par l'écureuil volant. L'approche Alpha permet ici de tirer parti de ce que peu de mammifères volent (en d'autre termes, la prémisse de la règle n'est pas fréquente dans la classe *mammifère* à laquelle appartient l'instance qui falsifie la règle). Utiliser une $\alpha$-extension permet de retirer l'écureuil-volant de l'extension de la prémisse de la règle. Ici, une valeur faible de $\alpha$ est suffisante pour obtenir une règle d'$\alpha$-implication (avec donc une confiance de 1) exprimant que les animaux volant ont des ailes. Bien sûr des valeurs plus élevées de $\alpha$, (proches de 100) évitent aussi la falsification de cette règle. Cependant dans ce dernier cas, une règle d' $\alpha$-implication exprime quelque chose de

différent : elle ne s'applique à une instance que lorsque la prémisse de la règle est commune à la *plupart* des instances de la même classe de base. Dans notre exemple, seuls les *oiseaux* seraient ainsi concernés par une telle règle mais pas les *insectes*.

## 7. Aspects théoriques

### 7.1. *Cas des classes de bases non disjointes et liens avec l'analyse formelle de concepts*

Nous avons défini les treillis de Galois Alpha en considérant que les classes de base formaient une partition de l'ensemble des instances. Nous considérons maintenant que ces classes recouvrent l'ensemble des instances $I$ mais ne sont pas nécessairement disjointes. Dans ce cas une modification naturelle de la définition 7 consiste à imposer qu'au moins une des classes de base auxquelles appartient l'instance $i$ $\alpha$-satisfasse le terme $T$. La $\alpha$-appartenance est alors définie comme suit : $i\ isa_\alpha\ T$ ssi $i\ isa\ T$ et il existe une classe de base $BC$ telle que $i \in BC$ et $BC\ sat_\alpha\ T$. En changeant de la même manière $proj_\alpha$ (une instance $i$ dans $e$ appartient à $proj_\alpha(e)$ s'il existe une classe de base $BC$ telle que $i \in BC$ et $\mid e \cap BC \mid \geq \frac{|BC|.\alpha}{100}$) nous obtenons encore une projection extensionnelle, donc une correspondance de Galois et un treillis de Galois. Les ordres partiels et totaux mentionnés à la section 3.2 sont préservés ainsi que les résultats concernant l'incrémentalité de la section 5.

Une autre question est celle de la relation entre les treillis Alpha et l'analyse formelle de concepts. Nous cherchons ici un contexte formel *représentatif* (Ganter *et al.*, 2001) d'un treillis Alpha donné $G_\alpha$. Il s'agit d'un contexte formel dont le treillis de concepts est isomorphe à $G_\alpha$. Nous considérons pour cela comme *objets*[4] de ce contexte formel des sous-ensembles particuliers des classes de base. Plus précisément, pour chaque classe de base $BCi$ nous considérons les plus petits éléments de $proj_\alpha(\mathcal{P}(BCi))$ différents de $\emptyset$. Nous notons $I_\alpha$ l'ensemble de tous ces sous-ensembles. Ainsi si nous reprenons l'exemple 1, nous obtenons :

$$I_{60} = \{\{i1, i2\}, \{i3, i4\}, \{i4, i5\}, \{i3, i5\}, \{i6, i7\}, \{i7, i8\}, \{i6, i8\}\}$$

La relation d'incidence $R_\alpha$ entre l'ensemble $I_\alpha$ des objets et l'ensemble $A$ des attributs est alors définie comme suit : $oR_\alpha a$ ssi $o \subseteq ext_\alpha(\{a\})$. Notons alors $ext_{I_\alpha}$ et $int_{I_\alpha}$ les fonctions d'extension et d'intension de ce contexte formel. Dans l'exemple 1 nous obtenons : $ext_{60}(a8) = \{i3, i5, i6, i7, i8\}$ et $ext_{60}(a4) = \{i3, i4\}$ et par conséquent $ext_{60}(a8a4)$=$proj_{60}(\{i3\})$=$\emptyset$. Nous obtenons alors $ext_{I_{60}}(a8)$= $\{\{i3, i5\}, \{i6, i7\}, \{i7, i8\}, \{i6, i8\}\}$, $ext_{I_{60}}(a4)$= $\{\{i3, i4\}\}$ et donc $ext_{I_{60}}(a8a4)$ = $ext_{I_{60}}(a8) \cap ext_{I_{60}}(a4) = \emptyset$. Remarquons que $I_0$ est constitué des singletons de $I$ et que $I_{100}$ est l'ensemble des *prototypes* des classes de base. $I_\alpha$ est alors considéré comme l'ensemble des *$\alpha$-prototypes* de $\mathcal{BC}$. Il est clair que pour tout $\alpha$-prototype $o$, $int_{I_\alpha}(\{o\}) = int(o)$ et plus généralement que $int_{I_\alpha}(\{o1, o2, ..., on\}) = int(o1 \cup o2 \cup ... \cup on)$.

4. C'est-à-dire comme nouvelles instances.

### 7.2. *Alpha extensions et* rough sets

Nous traitons ici du point de vue ensembliste de ce travail. Nous considérons qu'une instance $i$ $\alpha$-appartient à un sous-ensemble $e$ de $I$ (nous noterons $i \in_\alpha e$) lorsque $i$ appartient à $proj_\alpha(e)$. Dans ce cas $i\ isa_\alpha\ T$ s'écrit $i \in_\alpha ext(T)$. Un autre point de vue ensembliste traite de données partitionnées *a priori* en classes de base : il s'agit de la théorie des ensembles *rugueux* (la théorie des *rough sets* ((Pawlak, 1996)). Dans cette théorie, tout sous-ensemble $e$ de $I$ a un plus grand minorant $inf(e)$ et un plus petit majorant $sup(e)$ pris parmi les réunions de classes de base. Si on compare la vue ensembliste Alpha avec celle des ensembles rugueux, on constate immédiatement que pour tout sous-ensemble $e$ de $I$, on a $proj_{100}(e) = inf(e)$. Dans la théorie des ensembles rugueux l'appartenance rugueuse $\mu_e(i)$ est la proportion d'instances de la même classe de base que $i$ qui appartiennent à $e$. Pour comparer ces deux visions ensemblistes nous utilisons $\alpha$ pour discrétiser l'appartenance $\mu$. Ceci conduit à la relation d'appartenance suivante :

**Définition 14** *Soient $i$ un élément de $I$ et $e$ un sous-ensemble de $I$. La fonction d'appartenance rugueuse seuillée $\in^R_\alpha$ se définit comme suit :*
$i \in^R_\alpha e$ *ssi* $\mu_e(i) \geq \frac{\alpha}{100}$

Remarquons que lorsque $i \notin ext(T)$, on a $i \notin_\alpha ext(T)$, alors qu'il est parfaitement possible d'avoir $i \in^R_\alpha ext(T)$. En effet, le partitionnement sur les ensembles rugueux exprime une indiscernabilité entre instances de la même classe de base. Ainsi, dans la théorie des ensembles rugueux il y a un certain degré d'appartenance de $i$ à $e$ dès que $e \cap Bc(i)$ n'est pas vide. Au contraire, du point de vue ensembliste Alpha, l'appartenance de $i$ à un ensemble $e$ est un prérequis pour la $\alpha$-appartenance. La $\alpha$-appartenance exprime donc à quel point l'appartenance à $e$ est fréquente *aussi* parmi les instances de la même classe de base que $i$ : si $i$ est exceptionnel relativement à un terme $t$, alors $i$ n'appartient pas à $ext_\alpha(t)$ même si $i$ appartient à l'extension classique de $t$. Une conséquence heureuse de la vue ensembliste Alpha est l'opportunité de construire des treillis de Galois. Il est aisé de voir que pour $\alpha \neq 100$ la fonction $ext$ correspondant à la vue ensembliste rugueuse (donc utilisant $\in^R_\alpha$) ne forme pas, avec la fonction $int$, une correspondance de Galois (voir la condition C3 dans la définition 2).

## 8. Travaux proches, conclusion et perspectives

Des travaux récents en Représentation des Connaissances et en Apprentissage par Machine se sont intéressés aux correspondances et structures de Galois avec des langages de termes plus sophistiqués que les ensembles d'attributs (Ganter *et al.*, 1999; Ganascia, 1993; Liquiere *et al.*, 1998; Ganter *et al.*, 2001). Différents travaux ont également exploité en apprentissage supervisé la notion de treillis de concepts pour limiter l'espace de recherche à explorer. Par exemple les auteurs de (Njiwoua *et al.*, 1997) construisent partiellement la partie supérieure du treillis de concept des instances positives, et ceux de (Brézellec *et al.*, 1998) explorent à l'aide

d'une métaheuristique l'espace de recherche en se déplaçant dans celui-ci de termes fermés en termes fermés. Cependant, peu de travaux ont formellement examiné la possibilité de simplifier les treillis de concepts en modifiant la fonction d'*extension*, et donc la partie extensionnelle d'un concept. Nous avons vu qu'en modifiant la notion d'extension en se référant à une partition de l'ensemble des instances $I$, on modifie le treillis des extensions qui n'est plus $\mathcal{P}(I)$ et on obtient une nouvelle famille de treillis de Galois. En particulier nous avons vu que les treillis Iceberg (ou treillis de concepts fréquents) (Waiyamai *et al.*, 2000; Stumme *et al.*, 2002) sont formellement des treillis de Galois Alpha particuliers pour lesquels toutes les instances appartiennent à la même classe de base. En outre, les règles d'implication associées aux treillis Alpha correspondent à l'inclusion des $\alpha$-extensions. Ainsi, des bases de telles $\alpha$-implications peuvent-elles être extraites d'un treillis Alpha de la même manière que leurs contreparties classiques le sont d'un treillis de concepts. Ceci conduit à considérer les *treillis Alpha fréquents* qui, utilisant une notion de fréquence globale, (le $\alpha$-support), sont le pendant, pour les *règles d'association Alpha* des treillis fréquents pour les règles d'associations classiques. Remarquons que les $\alpha$-implications héritent de la structure de treillis de Galois des propriétés intéressantes (comme la transitivité) inhabituelles lorsqu'on travaille avec des règles *approximatives*.

En ce qui concerne la construction des treillis Alpha, nous n'avons proposé qu'un algorithme direct et il serait évidemment intéressant d'adapter des algorithmes efficaces de construction de treillis de concepts (*e.g.* (Kuznetsov *et al.*, 2002)). Cependant, la propriété 3 conduit à une autre manière d'engendrer les treillis de Galois Alpha en construisant d'abord les treillis fréquents correspondant à chaque classe de base, puis en les fusionnant à l'aide d'un opérateur analogue à l'opérateur de *subposition* utilisé en analyse formelle de concepts. (Valtchev *et al.*, 2002b) ont utilisé cet opérateur pour construire et maintenir efficacement un treillis de concepts. Nous avons donc une *incrémentalité par classes de base* des treillis Alpha que nous avons expérimentée ici et, en utilisant la projection et la fusion de treillis Alpha, un schéma général de mise à jour de ces treillis Alpha. Il y a en réalité encore beaucoup à faire pour expérimenter et examiner les problèmes théoriques et pratiques concernant les treillis Alpha et les $\alpha$-implications correspondantes. Ainsi il reste par exemple à produire des opérateurs permettant de passer d'un treillis grossier à un treillis plus fin sans tout reconstruire (nous savons remonter vers un treillis plus fin (Pernelle *et al.*, 2002), mais la reconstruction pose problème). Egalement, les conséquences, d'un point de vue ensembliste de la notion d'$\alpha$-extension sont encore à explorer, ainsi qu'un approfondissement de l'étude initiée ici sur l'ordre induit sur les représentations par la valeur de $\alpha$ et la partition initiale. Nous nous intéressons particulièrement en ce moment aux aspects logiques associés à cette fonction d'extension, et à la forme particulière que prend ici la déduction. De plus, beaucoup de travaux utilisant le paradigme de l'analyse formelle de concepts et de ses dérivés peuvent être étendus aux treillis Alpha, comme par exemple les travaux sur la navigation dans les systèmes d'information logique (Ferré *et al.*, 2004) ou encore un travail récent sur la reconnaissance de pièces en robotique mobile (Zenou *et al.*, 2004) qui utilise également une partition des instances (ici des images) pour extraire des caractéristiques discriminantes à l'aide d'un treillis

de concepts. En conclusion nous pensons que cette structure et, peut-être surtout, la notion d'extension qui lui est associée, forment une approche nouvelle de l'investigation des données permettant de faire varier en finesse la perception des données et de traiter les exceptions relatives à un point de vue préliminaire sur les données. Ce point de vue préliminaire est une partie du biais que tout utilisateur averti applique pour extraire et apprendre des connaissances à partir de données.

Remerciements

Nous remercions Nathalie Pernelle pour son importante contribution aux premiers travaux sur l'extension Alpha. Nicolas Tchitchek a finalisé l'adaptation de **Galicia** initiée par Francisco Cabrera, Aitor Echevarría et Abel Rodríguez pendant un séjour ERASMUS (pour les règles Alpha) et Dorayd Arraji (pour la fusion de treillis Alpha). Marc Champesme a supervisé l'ensemble de ces récents développements. Thibaut Lamadon est l'auteur du programme ALPHA et a mené les expériences utilisant ce programme.

## 9. Bibliographie

Barbu M., Montjardet M., *Ordre et Classification, Algèbre et Combinatoire 2*, Hachette Université, 1970.

Bastide Y., Taouil R., Pasquier N., Stumme G., Lakhal L., « PASCAL : un algorithme d'extraction de motifs fréquents », *TSI*, vol. 21, n° 1, p. 65-95, 2002.

Birkhoff G., *Lattice Theory*, American Mathematical Society Colloquium Publications, Rhode Island, 1973.

Bock H., Diday E., *Analysis of Symbolic Data*, H.H. Bock and E. Diday, Springer Verlag, 2000.

Brézellec P., Soldano H., « Tabata : a learning algorithm performing a bidirectional search in a reduced search space using a tabu strategy », *Eur. Conf. in Art. Int., ECAI-98*, J. Wiley, Brighton, England, p. 420-424, 1998.

Diatta J., « Génération de la base de Guigues-Duquenne-Luxembuger pour les règles d'association par une approche utilisant des mesures de similarité multivoies », *Conférence d'Apprentissage (CAP'2003)*, 1-4 juillet-2003, Laval, France, p. 281-298, 2003.

Ferré S., Ridoux O., « Introduction to logical information systems », *Information Processing and Management*, vol. 40, n° 3, p. 383-419, 2004.

Ganascia J., « TDIS : an Algebraic Formalization », *Int. Joint Conf. on Art. Int.*, vol. 2, p. 1008-1013, 1993.

Ganter B., Kuznetsov S. O., « Pattern Structures and Their Projections », *ICCS-01, LNCS*, vol. 2120, p. 129-142, 2001.

Ganter B., Wille R., *Formal Concept Analysis : Logical Foundations*, Springer Verlag., 1999.

Guigues J., Duquenne V., « Famille non redondante d'implications informatives résultant d'un tableau de données binaires », *Mathématiques et Sciences humaines*, vol. 95, p. 5-18, 1986.

Hereth J., Stumme G., Wille R., Wille U., « Conceptual Knowledge Discovery and Data Analysis », *Int. Conf. on Conceptual Structure.*, p. 421-437, 2000.

Kuznetsov S., Obiedkov S., « Comparing performance of algorithms for generating concept lattices », *J. of Experimental and Theoretical Art. Int.*, vol. 2/3, n° 14, p. 189-216, 2002.

Liquiere M., Sallantin J., « Structural Machine Learning with Galois lattice and Graphs », *ICML98, Morgan Kaufmann*, 1998.

Njiwoua P., Nguifo E., « IGLUE : An Instance-based Learning System over Lattice Theory », *Proceedings of ICTAI'97*, IEEE Press, California, USA, p. 255-274, 1997.

Pasquier N., Bastide Y., Taouil R., Lakhal L., « Efficient mining of association rules using closed itemset lattices », *Information Systems*, vol. 24, n° 1, p. 25-46, 1999.

Pawlak Z., « Rough Sets, Rough Relations and Rough Functions », *Fundamenta Informaticae*, vol. 27, n° 2/3, p. 103-108, 1996.

Pernelle N., Rousset M.-C., Soldano H., Ventos V., « ZooM : a nested Galois lattices-based system for conceptual clustering », *J. of Experimental and Theoretical Artificial Intelligence*, vol. 2/3, n° 14, p. 157-187, 2002.

Pernelle N., Rousset M.-C., Ventos V., « Automatic Construction and Refinement of a Class Hierarchy over Multi-valued Data », *5th Conference on Principles and practice of Knowlewdge Discovery in Databases, PKDD'01, Lecture Notes in Artificial Intelligence*, Springer-Verlag, p. 386-398, 2001.

Stumme G., Taouil R., Bastide Y., Pasquier N., Lakhal L., « Intelligent Structuring and Reducing of Association Rules with Formal Concept Analysis », *Lecture Notes in Computer Science*, vol. 2174, p. 335-349, 2001.

Stumme G., Taouil R., Bastide Y., Pasquier N., Lakhal L., « Computing iceberg concept lattices with Titanic », *Data and Knowledge Engineering*, vol. 42, n° 2, p. 189-222, 2002.

Valtchev P., Missaoui R., « Building concept (Galois) lattices from parts : generalizing the incremental methods », *proceeding of ICCS-01, Lecture Notes in Computer Science*, vol. 2120, p. 290-303, 2001.

Valtchev P., Missaoui R., Godin R., Meridji M., « Generating frequent itemsets incrementally : two novel approaches based on Galois lattice theory », *J. of Experimental and Theoretical Artificial Intelligence*, vol. 2/3, n° 14, p. 115-142, 2002a.

Valtchev P., Missaoui R., Lebrun P., « A partition-based approach towards building Galois (concept) lattices », *Discrete Mathematics*, vol. 256, n° 3, p. 801-829, 2002b.

Ventos V., Soldano H., Lamadon T., « Treillis de Galois Alpha », *Conférence d'Apprentissage (CAP'04)*, Presses Universitaires de Grenoble, Montpellier, France, p. 175-190, 2004.

Waiyamai K., Lakhal L., « Knowledge discovery from very large databases using frequent concept lattices », *11th Eur. Conf. on Machine Learning, ECML'2000*, p. 437-445, 2000.

Zenou E., Ghallab M., Samuelides M., « Topological Visual Localization Using Decentralized Galois Lattices. », *European Conference on Artificial Intelligence, ECAI 04*, p. 1117-1120, 2004.