Représentation de la variabilité par des treillis de concepts. Etude de la faisabilité pour des descriptions issues de matrices de comparaison de produits

Jessie Carbonnel, Marianne Huchard, Alain Gutierrez

Représentation de la variabilité par des treillis de concepts

Etude de la faisabilité pour des descriptions issues de matrices de comparaison de produits

Jessie Carbonnel* — **Marianne Huchard*** — **Alain Gutierrez***

* *LIRMM, CNRS & Université de Montpellier, France*
jessiecarbonnel@gmail.com, prenom.nomd@lirmm.fr

RÉSUMÉ. *Les matrices de comparaison de produits (MCP) croisent un ensemble de produits et un ensemble de caractéristiques. Si elles présentent une solution simple de lecture pour effectuer certaines tâches, elles gagnent à être accompagnées de représentations alternatives telles que les modèles de caractéristiques (feature model ou FM). Dans cet article, nous nous intéressons aux treillis de concepts qui constituent un autre outil pour la représentation de la variabilité. Une forme dérivée de l'Analyse Formelle de Concepts (AFC) permet en outre de représenter la variabilité d'une famille de produits décrite en particulier par une ou plusieurs autres familles de produits, telle que l'on peut la trouver dans Wikipedia. Nous montrons comment traduire les données en partie informelles des MCP afin de les analyser par l'AFC. Ces approches étant connues pour être combinatoires, nous étudions leur passage à l'échelle sur des MCP de Wikipedia ou générées aléatoirement. Nous montrons que l'un des types de structure conceptuelle (AOC-poset) possède de bonnes propriétés pour les études que nous désirons mener.*

ABSTRACT. *Product Comparison Matrices (PCMs) provide a simple way to express variability in a product family, but sometimes need to be completed by more formal alternatives like Feature Models (FMs). In this paper, we focus on concept lattices, another alternative already explored in several works to express variability. Besides, an extension of Formal Concept Analysis (FCA) can represent the case where a product family is described by other product families. We propose a way to translate data from PCMs to be analyzed by FCA-based approaches. Because of the combinatorial aspect of these approaches, we evaluate the scalability of the produced structures. We show that a particular structure (AOC-poset) possesses interesting properties for the studies that we envision.*

MOTS-CLÉS : *Ligne de produits, Analyse formelle de concepts, Représentation de la variabilité.*

KEYWORDS: *Product lines, Formal Concept Analysis, Variability Representation.*

1re soumission à *Journées lignes de produits 2014*, le 31/10/2014

1. Introduction

En ingénierie des lignes de produits, plusieurs formalismes peuvent être utilisés pour représenter la variabilité. Un grand nombre de données sont susceptibles d'être prises en compte pour la modéliser et il est donc important de fournir aux concepteurs des outils pour faciliter leur représentation ainsi que leur traitement.

Parmi les formalismes existants, le plus couramment utilisé est la *matrice de comparaison de produits* (MCP). Elle se présente sous la forme d'un tableau croisant un ensemble de produits d'une même famille avec un ensemble de caractéristiques : elle offre ainsi un moyen simple et visuel d'exposer les caractéristiques de chaque produit et de les comparer entre eux. On peut en trouver librement sur des sites d'associations de consommateurs ou bien sur Wikipédia. Une étude sur des matrices de comparaison extraites de Wikipédia (Sannier *et al.,*, 2013) met en avant certaines des limites de ce formalisme : parmi elles, on note un manque de formalisation du contenu des cellules de la matrice. Ce dernier point est dû à l'absence de format prédéfini ou de bonnes pratiques pour le remplissage de ces cellules, alors laissé au bon vouloir des utilisateurs. Une des conséquences est la difficulté à interpréter automatiquement ces informations : le modèle est compréhensible par un être humain mais n'est pas assez formel pour une analyse automatique. Ce même article offre des pistes pour passer de matrices de comparaison de produits à des modèles plus formels, tels que les *features models* (FM), et ce afin de dépasser ces limitations. Les FM modélisent la variabilité en décrivant les différentes caractéristiques existantes ainsi que leurs inter-dépendances, et font donc état des différentes configurations possibles pour les produits d'une même famille. Cependant, ils sont exclusivement centrés sur les caractéristiques et dans leur forme standard, ils ne montrent pas les liens avec les produits et ne proposent pas non plus de liens avec d'autres types de produits, à la manière des MCP.

L'*analyse formelle de concepts* (AFC) et les treillis de concepts ont déjà été étudiés comme une solution pour représenter la variabilité (Loesch *et al.,*, 2007). À partir d'un ensemble de produits décrits par des caractéristiques, l'AFC calcule des sous-ensembles maximaux de produits partageant des caractéristiques communes et les organise sous forme de hiérarchie. Contrairement aux MCPs, les treillis de concepts vont présenter de façon plus formelle les informations relatives à la variabilité. De par leur structure, ils mettent en évidence des propriétés sur les caractéristiques à la manière des FMs : les treillis conservent cependant le lien entre les différentes caractéristiques possibles et les produits existants, et ils organisent les produits entre eux de manière à mettre en évidence leurs similarités et leurs différences. De plus, l'AFC possède des aspects qui peuvent être intéressants pour modéliser certains cas particuliers. On peut trouver une MCP $\mathcal{A}$ qui possède une caractéristique dont le domaine de valeurs correspond à l'ensemble des produits d'une MCP $\mathcal{B}$: on se retrouve donc dans le cas où un ensemble de produits est décrit par un autre ensemble de produits. L'*analyse relationnelle de concepts* (ARC) est une extension de l'AFC qui va permettre de prendre en compte plusieurs ensembles d'objets ainsi que des liens entre les objets de ces différents ensembles, fournissant de cette façon un ensemble de treillis interconnectant la représentation de la variabilité d'un type de produit avec celle d'un autre type de

produit. Les treillis de concepts obtenus n'ont pas pour objectif d'être visualisés par un utilisateur : ils sont complexes et très peu lisibles dans leur intégralité. Cependant, leurs aspects formel et structurel en font de bons candidats pour leur appliquer des traitements automatiques : comparaison de produits, recherche par caractéristiques, visualisations partielles autour de points d'intérêts ou encore aide à la décision.

Les questions qui ont orienté nos recherches sont les suivantes : Comment représenter la variabilité exprimée sous la forme d'une MCP par un treillis de concepts ? Dans quelle mesure l'ARC peut-elle modéliser le cas où un ensemble de produits est décrit par un autre ? Les structures ainsi obtenues sont-elles exploitables pour effectuer différents traitements, tels qu'une analyse des produits ou l'extraction assistée d'un FM ? Nous allons d'abord faire un rappel sur l'utilisation de l'AFC et proposer une méthode pour passer d'une MCP à un treillis de concepts (Sect. 2). Nous étudierons alors comment étendre cette méthode pour prendre en compte des cas rencontrés dans Wikipedia où une MCP est décrite par un ensemble de produits faisant l'objet d'une autre MCP, et ce grâce à l'ARC (Sect. 3). Nous réaliserons ensuite des tests sur des matrices de comparaison de Wikipédia ou générées selon le même principe pour obtenir un ordre de grandeur sur le nombre de concepts composant les structures obtenues avec de telles techniques (Sect. 4). Nous évoquerons les travaux connexes en Section 5. Enfin, la conclusion et les perspectives seront exposées en Section 6.

2. AFC et matrices de comparaison de produits

Cette section propose une approche pour représenter la variabilité exprimée sous la forme d'une MCP par un treillis de concepts. Une matrice de comparaison se présente sous la forme d'un tableau à double entrée, qui croise un ensemble de produits avec un ensemble de caractéristiques. La figure 1 présente une matrice de comparaison de quatre produits décrits par deux caractéristiques telle que recueillie sur Wikipédia.

Language	Standardized	Paradigm
Java	Yes	Functional, Imperative, OO
Perl	No	Functional & Procedural & Imperative
Php	No	Functional, Procedural, Imperative and Object-Oriented
C#	Yes	functional, procedural, imperative, Object Oriented

Figure 1. *Extrait d'une matrice de comparaison de produits sur des Langages (http ://en.wikipedia.org/wiki/Comparison_of_programming_languages, Juil. 2014)*

On peut noter que le contenu des cellules ne respecte pas de format particulier : par exemple, les valeurs de la caractéristique *Paradigm* des produits *Php* et *C#* ont la même sémantique mais sont différentes syntaxiquement. Pour rendre le traitement des MCPs plus évident, nous les avons nettoyées pour qu'elles respectent un même format. L'étude de (Sannier *et al.*,, 2013) montre que le contenu d'une cellule peut être de huit types différents : *booléen* lorsque la valeur est prise dans {yes, no}, *booléen contraint, partiel ou ambigu* si elle est en plus accompagnée de valeurs qui viennent la contraindre, *simple* si une seule valeur peut satisfaire le critère, *multiple* si plusieurs

valeurs peuvent satisfaire le critère, *inconnu*, *nul* si la cellule est vide, *incohérent* s'il n'a pas de lien avec le critère ou *ajouté* si la cellule présente des informations complémentaires. Nous avons traité chaque cas comme suit. Les cellules vides, bien que dénotant un manque d'informations, ne présentent pas d'inconvénients. Les cellules incohérentes devront être repérées et leurs informations effacées ou modifiées. Les informations ajoutées ne seront pas prises en compte. Les autres types de cellules pourront posséder soit une seule valeur soit une liste de valeurs, séparées par des virgules. Une matrice correctement nettoyée possèdera donc trois types de caractéristiques : *booléenne simple*, dont le domaine de valeurs est uniquement {Yes, No}, *booléenne contrainte* si l'on peut trouver des valeurs en plus de *Yes* et *No* apportant des informations complémentaires, et *valeur* qui indique une liste de valeurs non booléennes. La figure 2 représente la matrice de la figure 1 respectant ce format.

Language	Standardized	Paradigm
Java	Yes	Functional, Imperative, Object-Oriented
Perl	No	Functional, Procedural, Imperative
Php	No	Functional, Procedural, Imperative, Object-Oriented
C#	Yes	Functional, Procedural, Imperative, Object-Oriented

Figure 2. *MCP de la figure 1. nettoyée manuellement*

Un contexte formel est semblable à une MCP, mais les objets qu'il présente sont décrits par des attributs binaires. Cette étape peut donc se résumer à convertir un ensemble de caractéristiques multivaluées en attributs binaires. La figure 3 présente le contexte formel obtenu après conversion de la matrice de la figure 2.

Language	Standardized :Yes	Standardized :No	Functional	Procedural	Imperative	Object Oriented
Java	x		x		x	x
Perl		x	x	x	x	
Php		x	x	x	x	x
C#	x		x	x	x	x

Figure 3. *Contexte formel obtenu après scaling de la MCP de la figure 2.*

Cette conversion est réalisée avec la technique de ***scaling*** (Ganter *et al.,*, 1999). Le *scaling* d'un attribut multivalué consiste à créer un attribut binaire pour chaque valeur ou groupe de valeurs possible. Les caractéristiques booléennes peuvent être traduites en un unique attribut binaire indiquant la présence de la caractéristique. Cependant, du fait de la présence de cellules vides, le fait qu'un produit ne possède pas d'attribut peut signifier deux choses distinctes : l'absence de l'attribut (valeur *No*) ou bien une information non renseignée (cellule vide). Si l'on souhaite être plus précis, on peut choisir de produire deux attributs binaires, l'un indiquant la présence de la caractéristique, l'autre son absence. Si elles sont contraintes, on peut garder ces contraintes sous forme d'attributs binaires qui viendront compléter les deux attributs existants. Les valeurs peuvent être discrètes ou continues. Dans le premier cas, on associe un attribut à chaque valeur discrète, et dans le second cas, à chaque intervalle de valeurs.

L'analyse formelle de concepts (Ganter *et al.,*, 1999) permet de construire un treillis de concepts à partir d'un contexte formel. L'AFC est un cadre mathématique qui permet de structurer un ensemble d'objets pour mettre en évidence leurs différences et leurs similitudes. On extrait pour cela un ensemble de concepts, que l'on munit d'un ordre partiel. Un contexte formel est un triplet (O, A, R) où O et A sont deux ensembles finis et R $\subseteq$ O x A une relation binaire. Les éléments de O sont appelés les objets et ceux de A les attributs. Un couple de R définit que "l'objet o possède l'attribut a". Pour un contexte formel K = (O, A, R), un concept est une paire (E, I) telle que $E \subseteq O$ et $I \subseteq A$. Il représente un ensemble maximal d'objets partageant un ensemble maximal d'attributs communs. $E = \{o \in O | \forall a \in I, (o, a) \in R\}$ est l'extension du concept et $I = \{a \in A | \forall o \in E, (o, a) \in R\}$ est l'intension du concept. Pour un contexte formel K = (O, A, R) et deux concepts $C_1 = (E_1, I_1)$ et $C_2 = (E_2, I_2)$ de K, $C_1 \leq C_2$ si et seulement si $E_1 \subseteq E_2$ et $I_2 \subseteq I_1$. C_1 est un sous-concept de C_2 et C_2 est un super-concept de C_1. L'ensemble des concepts issus de K muni de l'ordre partiel $\leq$ forme une structure de treillis, appelée **treillis de concepts**. La figure 5 (droite) présente le treillis de concepts du contexte de la figure 3. Dans ce treillis, on peut lire que tous les langages permettent d'écrire selon les paradigmes fonctionnel et impératif ; le produit *Php* possède tous les attributs de *Perl* plus l'attribut *Object Oriented* ; les langages standardisés sont tous orientés objet.

On choisit de représenter l'extension et l'intension d'un concept de manière optimisée en supprimant les éléments hérités (les attributs étant hérités en descendant et les objets en montant dans le treillis) : on ne fait apparaître un attribut ou un objet qu'une seule fois, dans le concept dans lequel il est introduit. On parle de concept-objet et de concept-attribut pour des concepts qui introduisent respectivement au moins un objet et au moins un attribut. Un concept peut être les deux à la fois. L'AOC-poset (pour Attribute-Object-Concept poset) associé au contexte formel K est le sous-ordre de (C_K,$\leq$) induit par les concepts-objet et les concepts-attribut.

3. ARC et treillis de produits interconnectés

Dans cette section, nous présentons une méthode pour modéliser le cas particulier où un ensemble de produits est décrit par un ensemble de caractéristiques représentant un autre ensemble de produits. Par exemple, on peut trouver sur Wikipédia une MCP *Wikisoftware* qui fait référence à *Language* : on veut pouvoir structurer les *Wikisoftwares* en fonction des langages dans lesquels ils sont écrits. Supposons la MCP *Wikisoftware* possédant deux caractéristiques : *Open Source* (booléenne) et *Spam Prevention* (booléenne contrainte). En appliquant l'approche de la Section 2, la matrice *Wikisoftware* forme alors un contexte objets-attributs (figure 4). La figure 5 (gauche) présente le treillis de concepts associé au contexte de la figure 4.

L'*analyse relationnelle de concepts* (ARC) (Hacène *et al.,*, 2013) est une extension de l'AFC qui va permettre de prendre en compte plusieurs ensembles d'objets. Chaque ensemble d'objets reste défini par ses attributs propres et peut être lié à un autre ensemble d'objets par un contexte relationnel (contexte objets-objets). Ce der-

Wikisoftware	OS :Yes	OS :No	SP :Yes	SP :No	SP :Captcha	SP :Blacklist
TeamPage		x	x		x	
Socialtext			x			
MindTouch		x	x			
DokuWiki	x		x			x
EditMe		x	x		x	

Figure 4. *Contextes objets-attributs de Wikisoftware et de Language.*

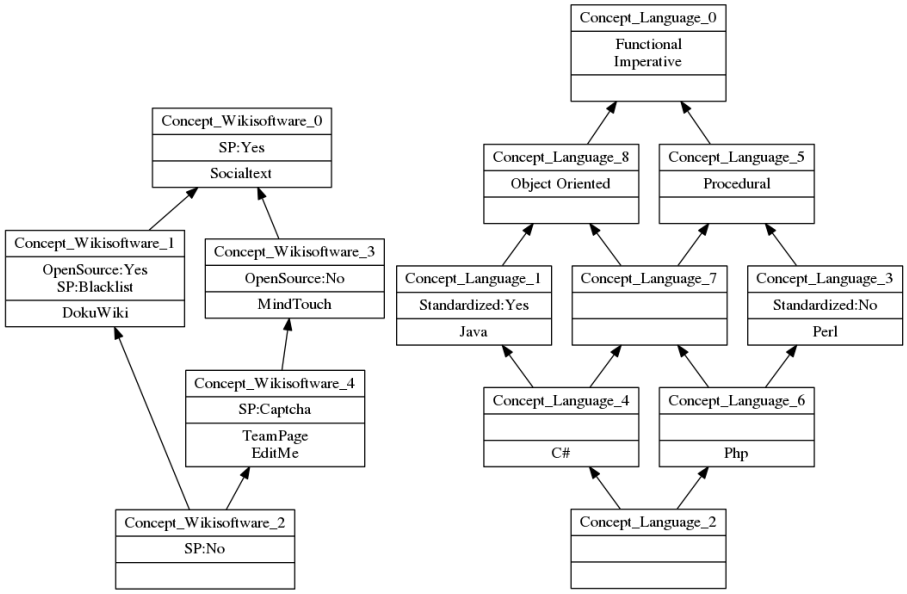

Figure 5. *Treillis de concepts de Wikisoftwares et de Language à l'étape 0 (construits avec RCAExplore : http ://dolques.free.fr/rcaexplore.php)*

nier est un triplet (O_1, O_2, I) où O_1 (source) et O_2 (cible) sont deux ensembles finis d'objets tels qu'il existe deux contextes formels (O_1, A_1, R_1) et (O_2, A_2, R_2), et où $I \subseteq O_1 \times O_2$ est une relation binaire. On exprime alors la relation entre *Wikisoftware* et *Language* par le contexte objets-objets de la figure 6.

isWritten	Java	Perl	Php	C#
TeamPage	x			
Socialtext		x		
MindTouch			x	x
DokuWiki			x	
EditMe	x			

Figure 6. *Contexte objets-objets décrivant la relation entre les objets des deux contextes de la figure 4.*

Dans l'approche ARC, les contextes objets-attributs sont étendus en fonction des contextes objets-objets pour prendre en compte les liens entre objets. Soit un contexte objets-objets $R_j = (O_k, O_l, I_j)$, l'ARC va étoffer le contexte objets-attributs de l'en-

semble O_k en fonction des concepts d'un treillis associé au contexte objets K_l en y ajoutant des attributs relationnels. Ils vont apparaître dans le treillis au même titre que les autres attributs, à la différence que ceux-ci vont pouvoir être considérés comme des *références* vers les concepts qu'ils désignent, dans d'autres treillis. Le treillis obtenu après extension du contexte de *Wikisoftware* est présenté dans la figure 7.

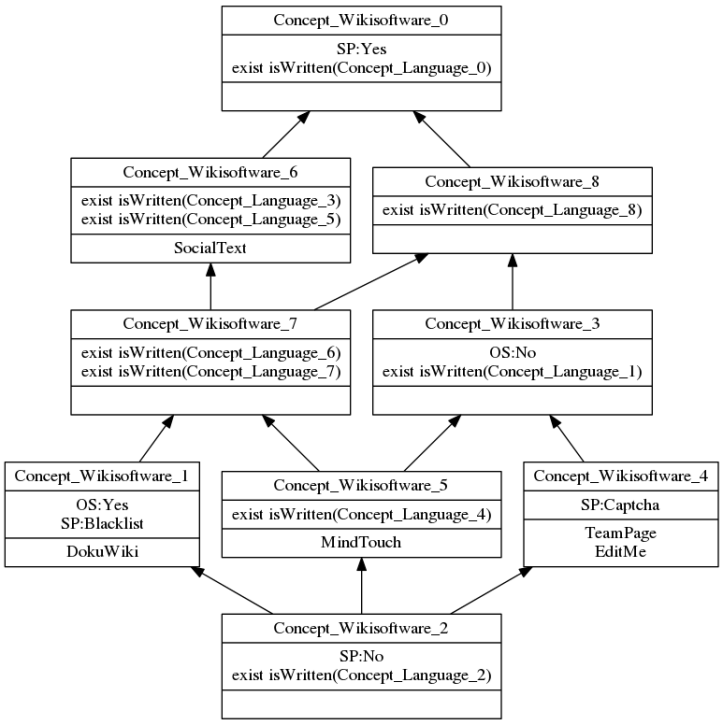

Figure 7. *Treillis de concepts de Wikisoftwares à l'étape 1*

Dans ce dernier, les attributs relationnels se lisent comme des références vers le treillis de *Language* (figure 5). Par exemple, *DokuWiki* (concept 1) est écrit dans un langage caractérisé par les concepts 6, 7, 8, 3, 5 et 0 de *Language*, ce qui correspond aux attributs *Standardized :No*, *Procedural*, *Functional*, *Object Oriented* et *Imperative*. On peut aussi extraire des informations à partir de la structure d'un treillis obtenu avec l'ARC au même titre que l'AFC. Par exemple, avec les données étudiées, *TeamPage* et *EditMe* sont des produits équivalents car ils sont introduits dans le même concept (mêmes attributs et attributs relationnels), un logiciel non *open source* sera écrit dans un langage standardisé et un *open source* dans un langage non standardisé.

Il existe différentes manières pour un objet du domaine de O_k d'être en relation avec un concept formé sur O_l, par exemple : un objet est lié (par I_j) à au moins un des objets de l'extension d'un concept (*scaling existentiel*) ; un objet est lié (par I_j) seulement aux objets de l'extension d'un concept (*scaling universel*). On précise pour

chaque relation de R quel *scaling* est utilisé. Chaque concept c formé sur O_l forme alors un attribut relationnel de la forme *q r :*c où *q* est un opérateur de *scaling* et *r* est la relation entre O_k et O_l.

Une Famille de Contextes Relationnelle (FCR) est une paire (K, R) telle que K est un ensemble de contextes objets-attributs $K_i = (O_i, A_i, I_i)$, $1 \leq i \leq n$ et R est un ensemble de contextes objets-objets $R_j = (O_k, O_l, I_j)$, $1 \leq j \leq m$, avec $I_j \subseteq O_k \times O_l$. Dans notre exemple, la FCR se compose de *Wikisoftware*, *Language* et du contexte objets-objets *isWritten*. Etant donné un contexte formel $\mathcal{K} = (O, A, I)$, on définit $rel(\mathcal{K})$ l'ensemble des relations (contextes objets-objets) de R qui ont pour domaine O, et ρ une fonction qui associe un opérateur de *scaling* à chaque contexte objets-objets de R. À chaque étape, on augmente le contexte $\mathcal{K}$ en ajoutant les attributs relationnels issus de chaque contexte de $rel(\mathcal{K})$: on obtient alors l'**extension relationnelle complète de** $\mathcal{K}$. Lorsque l'on calcule l'extension relationnelle complète de chaque contexte de K, on obtient l'**extension relationnelle complète de la FCR**.

Le processus de construction de l'ARC consiste à générer une série de contextes et de treillis associés à la FCR de départ (K, R) et ρ. À l'étape 0, on génère les treillis associés aux contextes de K. $K^0 = K$. À une étape $e + 1$: On calcule l'extension relationnelle complète des contextes de K^e. Les contextes objets-attributs étendus ainsi obtenus (K^{e+1}) possèdent des attributs relationnels qui référencent les concepts des treillis obtenus à l'étape précédente, comme montré sur l'exemple.

4. Tests sur la taille des structures obtenues

Jusqu'ici, nous avons proposé une méthode pour obtenir un treillis de concepts à partir d'une MCP, ainsi qu'une solution pour représenter plus précisément le cas où des caractéristiques font l'objet de leur propre MCP, et ce grâce à un ensemble de treillis interconnectés. Dans cette section, nous testons sur des matrices existantes les méthodes précédentes afin d'avoir une idée de leur exploitabilité. Le nombre de concepts générés à partir d'un contexte formel dépend à la fois du nombre d'objets, du nombre d'attributs, et de la forme de ce contexte : ce nombre peut atteindre au maximum $2^{min(|O|,|A|)}$ dans le cas d'un treillis, et $|O| + |A|$ pour un AOC-poset. Dans les tests suivants, nous avons généré à la fois des treillis de concepts et des AOC-posets, et ce dans le but de mettre en évidence l'impact de la suppression des concepts n'introduisant ni produits, ni attributs sur la taille de la structure finale. Nous avons conduit ces tests une première fois sur des contextes entiers (*scaling* générant deux attributs pour les caractéristiques booléennes et un attribut pour chaque contrainte pour les booléennes contraintes) et une seconde fois sur des contextes réduits (*scaling* ne générant qu'un seul attribut pour les caractéristiques booléennes et booléennes contraintes).

Nous avons d'abord conduit des tests sur plus de 40 matrices de comparaison issues de Wikipédia sans considérer de relations entre produits, que nous avons traduites en contextes formels afin d'obtenir un ordre de grandeur sur les structures obtenues avec la méthode de *scaling* (Sect. 2). Les résultats sont présentés dans la figure 8.

Nous avons analysé 1298 produits, construit 6528 attributs binaires et généré un total de 28464 concepts formels. Certains treillis peuvent atteindre les 5000 concepts. Ce nombre étant très élevé, il est difficile de traiter rapidement ces données (recherches, parcours ...). Les contextes réduits donnent des structures moins importantes, mais le nombre de concepts reste là encore considérable. Cependant, les résultats obtenus avec les AOC-posets sont encourageants : le nombre le plus élevé de concepts obtenus pour les mêmes matrices ne dépasse pas les 170.

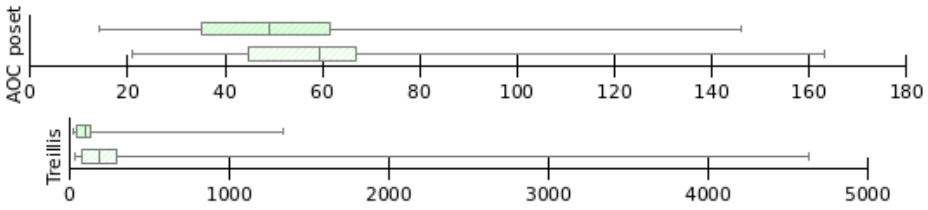

Figure 8. *Diagrammes de Tukey sur le nombre de concepts générés pour un AOC-poset et un treillis de concepts, sur des contextes réduits (haut) et entiers (bas).*

Par la suite, nous avons effectué le même type de tests sur des structures dont les contextes avaient été étendus selon des relations (Sect. 3). Afin d'obtenir des chiffres significatifs pour l'utilisation de l'ARC, nous avons voulu conduire des tests sur un nombre assez important de matrices. Or, alors qu'il est simple de trouver des matrices de comparaison sur Wikipédia, il est plus difficile de trouver des relations entre celles-ci. Nous avons donc généré des contextes objets-objets aléatoirement (selon le modèle de ceux existants) pour simuler des relations entre des matrices déjà existantes. Les résultats sont présentés dans la figure 9. Les treillis de concepts sont immenses (jusqu'à 60000 concepts), alors que les AOC-posets restent relativement abordables (moins de 250 concepts). On peut en déduire que les treillis de concepts ainsi générés possèdent une majorité de concepts « vides » : l'AOC-poset se présente comme une bonne alternative pour obtenir des structures bien moins complexes que les treillis. Nous avons également réalisé deux tests sur des relations existantes entre des MCP de Wikipédia. Le premier fait le lien entre *Wikisoftwares* (41 produits et 36 critères) et *ProgrammingLanguages* (119 produits et 15 critères). Le contexte *Wikisoftwares* une fois étendu génère 21055 concepts dans le cas d'un treillis, et 167 dans celui d'un AOC-poset. Le second test relie *LinuxDistribution* (77 produits, 25 critères) avec *FileSystem* (103 produits, 60 critères) et *InitSystem* (7 produits, 10 critères). Le contexte étendu de *LinuxDistribution* produit un treillis de 1310 concepts et un AOC-poset de 189 concepts. Dans ces deux tests, nous avons généré des contextes entiers. Les résultats sur ces deux cas réels concordent avec les résultats obtenus sur les données générées aléatoirement.

5. Travaux connexes

À notre connaissance, les premiers travaux utilisant l'Analyse Formelle de Concepts pour analyser les relations entre des configurations de produits et des ca-

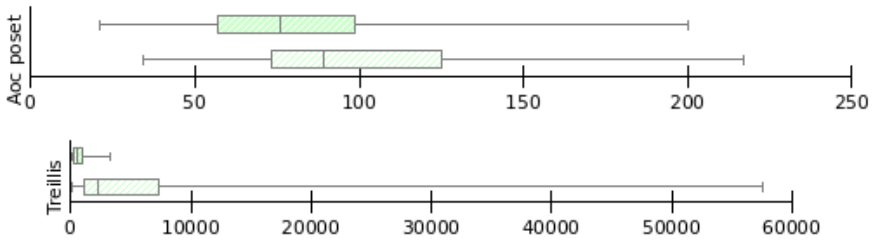

Figure 9. *Diagrammes de Tukey sur le nombre de concepts générés après extension des contextes formels avec l'ARC.*

ractéristiques sujettes à variation pour assister la construction d'une ligne de produits se trouvent dans Loesch et Ploederer (Loesch *et al.*,, 2007). Dans cette approche, le treillis de concepts est analysé pour en extraire des informations sur des groupes de caractéristiques soit toujours présentes, soit jamais utilisées, soit toujours présentes ensemble, soit jamais présentes ensemble. Les auteurs utilisent cette information pour décrire des contraintes entre caractéristiques et proposer des restructurations sur les caractéristiques (comme fusion, disparition ou identification de groupes de caractéristiques alternatives). Dans (Niu *et al.*,, 2009), les auteurs étudient, dans le contexte de l'ingenierie des exigences orientée aspect, un treillis de concepts qui classe les scénarios par les exigences fonctionnelles et analysent d'une part la relation entre les concepts et les exigences de qualité (usabilité, maintenance, accessibilité), et d'autre part les interférences entre les exigences de qualité. Ils analysent aussi l'impact des changements. Cette réflexion a été étendue par la suite par des remarques sur l'information que contient le treillis (ou le sous-ordre particulier que constitue un AOC-poset) sur les produits et sur la manière dont des travaux n'utilisant pas l'AFC en sont cependant proches (AL-msie'deen *et al.*,, 2014b). Dans le présent travail, nous montrons comment étendre l'approche d'origine, qui analyse des produits décrits par des caractéristiques, à un cas plus général dans lequel certaines des caractéristiques sont elles-mêmes d'autres produits. De plus nous évaluons le passage à l'échelle de l'AFC et de l'ARC sur des produits décrits dans des matrices de comparaison de produits tirées de wikipedia et connectés par des relations générées aléatoirement.

Dans le domaine des lignes de produits, une autre catégorie de travaux s'intéresse à l'identification de caractéristiques dans du code source en utilisant l'AFC (Xue *et al.*,, 2012 ; Al-Msie'deen *et al.*,, 2013). Dans ce cas, les entités décrites sont des variantes de logiciels, leur description est donnée par des éléments de code source et on cherche à produire des groupes d'éléments de code source candidats à être des caractéristiques. Des travaux cherchent des liens de traçabilité entre les caractéristiques et le code (Salman *et al.*,, 2013). Dans (Eisenbarth *et al.*,, 2003), les auteurs croisent des parties de code source et des scénarios qui les exécutent et qui mettent en œuvre des caractéristiques. L'objectif est d'identifier les parties du code correspondant à l'implémentation des caractéristiques. Dans notre cas, nous ne travaillons pas sur du code source, ni avec des scénarios, mais avec une description des produits déjà connue.

Enfin un dernier groupe de travaux étudie l'organisation de caractéristiques dans un FM grâce à l'AFC. Certaines approches (Yang *et al.*,, 2009 ; Ryssel *et al.*,, 2011) utilisent la structure conceptuelle (treillis ou AOC-poset) à la fois pour connaître les contraintes, mais également pour suggérer des relations de type *sous-caractéristique* liées au domaine métier. Dans l'article (Ryssel *et al.*,, 2011), les auteurs étudient de manière très approfondie l'extraction d'implications dans le treillis et les relations de couverture, pour déterminer par exemple si un ensemble de caractéristiques couvre tous les produits. Des travaux récents (AL-msie'deen *et al.*,, 2014a ; Shatnawi *et al.*,, 2015) se focalisent plutôt sur la mise en valeur de relations logiques entre caractéristiques dans le FM et sur l'identification de contraintes transverses. Dans le présent travail, nous construisons une structure ou plusieurs structures conceptuelles connectées. Ces structures sont créées à des fins d'analyse de la variabilité, mais nous ne nous sommes pas intéressés au problème de la construction d'un FM.

6. Conclusion

Dans cet article, nous avons choisi d'étudier l'analyse formelle de concepts et les treillis de concepts comme complément aux matrices de comparaison de produits pour représenter la variabilité au sein d'une ligne de produits. Nous proposons une méthode pour traduire une matrice de comparaison en contexte formel en utilisant la méthode de *scaling*. Nous proposons d'utiliser l'ARC pour modéliser le cas particulier où le domaine de valeurs d'une caractéristique appartenant à une MCP est un ensemble de produits possédant sa propre matrice de comparaison. On obtient alors un ensemble de treillis interconnectés, apportant une nouvelle dimension à la recherche et à la classification de produits lorsque ceux-ci sont en relation avec d'autres familles de produits.

Nous avons suite à cela conduit une série de tests visant à déterminer un ordre de grandeur sur le nombre de concepts composant les structures finales, obtenues d'abord avec l'AFC en traduisant des matrices de comparaison en contextes formels, puis avec l'ARC en introduisant des liens entre les contextes. Nous les avons réalisés en comparant deux structures différentes : les treillis de concepts qui établissent un sous-ordre parmi tous les concepts obtenus, et les AOC-posets qui accomplissent la même tâche sans prendre en compte les concepts non introducteurs d'information (produit ou caractéristique). Ceux-ci représentant la majorité des concepts au sein des treillis générés, l'AOC-poset s'impose comme une alternative largement plus avantageuse.

À l'avenir, nous souhaitons étudier les travaux de (Bécan *et al.*,, 2014) pour améliorer la traduction des MCP en données pour l'AFC et l'ARC. De plus, des recherches sur les traitements et leur application sur de telles structures seront effectuées pour venir compléter cette première analyse. Aussi, une étude plus approfondie des possibilités offertes par l'ARC pour modéliser d'autres cas que celui présenté ici est envisagée.

7. Bibliographie

Al-Msie'deen R., Seriai A., Huchard M., Urtado C., Vauttier S., Salman H. E., « Mining Features from the Object-Oriented Source Code of a Collection of Software Variants Using Formal Concept Analysis and Latent Semantic Indexing », *Proc. of The 25th SEKE*, p. 244-249, 2013.

AL-msie'deen R., Huchard M., Seriai A.-D., Urtado C., Vauttier S., « Concept lattices : a representation space to structure software variability », *Proc. of the 5th IEEE ICICS*, p. 1-6, 2014a.

AL-msie'deen R., Huchard M., Seriai A.-D., Urtado C., Vauttier S., Al-Khlifat A., « Concept lattices : a representation space to structure software variability », *Proc. of the 11th CLA*, vol. 1252, CEUR Workshops, p. 11-22, 2014b.

Bécan G., Sannier N., Acher M., Barais O., Blouin A., Baudry B., « Automating the formalization of product comparison matrices », *Proc. of the 29th ACM/IEEE ASE '14*, p. 433-444, 2014.

Eisenbarth T., Koschke R., Simon D., « Locating Features in Source Code », *IEEE Trans. Softw. Eng.*, vol. 29, n° 3, p. 210-224, 2003.

Ganter B., Wille R., *Formal concept analysis - mathematical foundations*, Springer, 1999.

Hacène M. R., Huchard M., Napoli A., Valtchev P., « Relational concept analysis : mining concept lattices from multi-relational data », *Ann. Math. Artif. Intell.*, vol. 67, n° 1, p. 81-108, 2013.

Loesch F., Ploedereder E., « Restructuring Variability in Software Product Lines using Concept Analysis of Product Configurations », *Proc. of the 11th IEEE ECSMR*, p. 159-170, 2007.

Niu N., Easterbrook S. M., « Concept analysis for product line requirements », *Proc. of the 8th AOSD 2009*, p. 137-148, 2009.

Ryssel U., Ploennigs J., Kabitzsch K., « Extraction of feature models from formal contexts », *Proc. of ACM SPLC '11*, p. 4 :1-4 :8, 2011.

Salman H. E., Seriai A., Dony C., « Feature-to-code traceability in a collection of software variants : Combining formal concept analysis and information retrieval », *Proc. of the14th IEEE IRI*, p. 209-216, 2013.

Sannier N., Acher M., Baudry B., « From comparison matrix to Variability Model : The Wikipedia case study », *Proc. of the 28th IEEE/ACM ASE 2013*, p. 580-585, 2013.

Shatnawi A., Seriai A.-D., Sahraoui H., « Recovering Architectural Variability of a Family of Product Variants », *To appear in Proc. of the 14th ICSR*, 2015.

Xue Y., Xing Z., Jarzabek S., « Feature Location in a Collection of Product Variants », *Proc. of the 19th IEEE WCRE*, p. 145-154, 2012.

Yang Y., Peng X., Zhao W., « Domain Feature Model Recovery from Multiple Applications Using Data Access Semantics and Formal Concept Analysis », *Proc. of the 16th IEEE WCRE*, p. 215-224, 2009.