Concepts de plus proches voisins dans des graphes de connaissances

Sébastien Ferré

IRISA/Université de Rennes 1
Campus de Beaulieu, 35042 Rennes cedex
ferre@irisa.fr

Résumé : Nous introduisons la notion de *concept de voisins* comme alternative à la notion de distance numérique dans le but d'identifier les objets les plus similaires à un objet requête, comme dans la méthode des plus proches voisins. Chaque concept de voisins est composé d'une intension qui décrit symboliquement ce que deux objets ont en commun et d'une extension qui englobe les objets qui se trouvent entre les deux. Nous définissons ces concepts de voisins pour des données complexes, les graphes de connaissances, où les nœuds jouent le rôle d'objets. Nous décrivons un algorithme *anytime* permettant d'affronter la complexité élevée de la tâche et nous présentons de premières expérimentations sur un graphe RDF de plus de 120.000 triplets.

Mots-clés : Graphes de connaissances, Web sémantique, Plus proches voisins, Analyse de concepts formels, Graph-FCA, Concepts de voisins, Hypergraphes, Homomorphismes.

1 Introduction

La méthode des k plus proches voisins (k-NN) (Mitchell, 1997) est à la fois simple et polyvalente. Elle permet la classification supervisée et le raisonnement à partir de cas (RàPC) (De Mantaras *et al.*, 2005). Son principe est, partant d'une instance, de trouver dans un jeu de données les instances les plus similaires, les *plus proches voisins*, et de s'appuyer sur ces voisins pour prédire la classe (classification) ou adapter une solution (RàPC). C'est une forme d'apprentissage paresseux (*lazy learning*) puisque les instances sont mémorisées tel quel, sans généralisation explicite.

Notre objectif est de pouvoir appliquer la méthode des k-NN aux graphes de connaissances, où chaque nœud est une instance et où le graphe entier participe à la description de chaque instance. Par exemple, dans les données de MONDIAL (May, 1999), les pays, continents, rivières, etc. sont mutuellement décrits les uns par les autres. Bisson (2000) distingue deux types de similarités. Les similarités *numériques* ont l'avantage d'être souples et économiques mais elles sont peu explicables et peuvent être trompeuses en cachant derrière une même valeur des similarités très différentes. De plus, il existe peu de telles mesures sur des données relationnelles (ex., RIBL (Horváth *et al.*, 2001)). Les similarités *symboliques* évitent ces inconvénients en produisant des représentations symboliques généralisant plusieurs instances. Néanmoins, à notre connaissances, ces similarités ont seulement été utilisées pour la formation de concepts et règles par généralisation (ex., PLI (Muggleton, 1995)), pas pour la recherche de plus proches voisins. De plus, les travaux existants considèrent généralement comme instances des petits graphes (ex. molécules (Kuznetsov, 2013)) plutôt que les nœuds d'un grand graphe. Un inconvénient des similarités symbolique est leur coût de calcul.

La contribution de ce papier est de proposer une approche à la fois symbolique, générale et efficace des plus proches voisins pour des graphes de connaissances (ex., graphes RDF, graphes

conceptuels (Chein & Mugnier, 2008)). Cette approche, CNN (*Concepts of Nearest Neighbours*), définit la similarité symbolique entre deux instances comme un *concept de voisins* dont l'intension représente ce que les deux instances ont en commun, et dont l'extension englobe les autres instances qui se trouvent "entre" les deux instances. La taille de l'extension peut tenir lieu de distance et la taille de l'intension de similarité numérique. Des clusters de plus proches voisins peuvent ainsi être identifiés et exploités comme dans l'approche classique. L'approche est purement *symbolique* parce que le graphe est exploité tel quel, sans recodage ou extraction de features, et parce que chaque plus proche voisin est accompagné d'une représentation intelligible de sa similarité (intension). L'approche est *générale* car elle couvre une large classe de graphes, les multi-hypergraphes, et ne requiert aucun paramètre (ex., pas de limite dans la profondeur d'exploration du graphe). Enfin, l'approche est *efficace* en focalisant les comparaisons sur l'instance requête et grâce à des techniques originales : *énumération de concepts par partitionnement* et *forme paresseuse de jointure* pour le calcul des extensions de concepts.

L'article est organisé comme suit. La section 2 situe notre approche par rapport aux approches existantes. La section 3 rappelle les définitions de Graph-FCA, le cadre dans lequel nos travaux se placent. La section 4 définit la notion de "concept de voisin" et la section 5 détaille un algorithme efficace pour les calculer. Enfin, la section 6 décrit nos premières expérimentations sur des données réelles, avant de conclure dans la section 7.

2 Travaux existants

La principale approche k-NN pour des données relationnelles est RIBL (*Relational Instance-Based Learning*) (Horváth *et al.*, 2001). Elle définit une distance numérique entre objets sur la base de l'exploration arborescente de l'hypergraphe à partir de cet objet jusqu'à une profondeur limite, sans prendre en compte les éventuels cycles. Nous avons appliqué une approche similaire pour guider des utilisateurs dans la production de descriptions RDF à partir d'exemples (Hermann *et al.*, 2012). À notre connaissance, aucune approche k-NN ne considère autre chose que des valeurs numériques comme représentation des distances entre objets.

Dans le domaine de l'Analyse de concepts formels (FCA) (Ganter & Wille, 1999), des notions proches de nos "concepts de voisins" ont été proposées mais elles s'appliquent à des données non-relationnelles. Kuznetsov (2013) utilise de tel concepts pour faire de la classification. Pour chaque exemple à classer, cela nécessite de calculer un concept pour chaque exemple déjà classé. Cette approche a été appliquée à des graphes mais où les objets sont de petits graphes (ex., molécules) et non pas les nœuds d'un grand graphe de connaissances. Nous avions proposé (Ferré & Ridoux, 2002) une approche similaire à celle de Kuznetsov mais avec un calcul des concepts de voisins dirigé par la description de l'objet à classer plutôt que par l'énumération des objets déjà classés.

Il existe deux extensions de la FCA applicables à des graphes de connaissances : RCA (*Relational Concept Analysis*) (Rouane-Hacene *et al.*, 2013) et Graph-FCA (Ferré, 2015). Nous avons choisi de nous appuyer formellement sur Graph-FCA car elle permet de définir chaque concept de voisins indépendamment des autres concepts. De plus, Graph-FCA autorise les hypergraphes (relations n-aires) et prend en compte les cycles, contrairement à RCA.

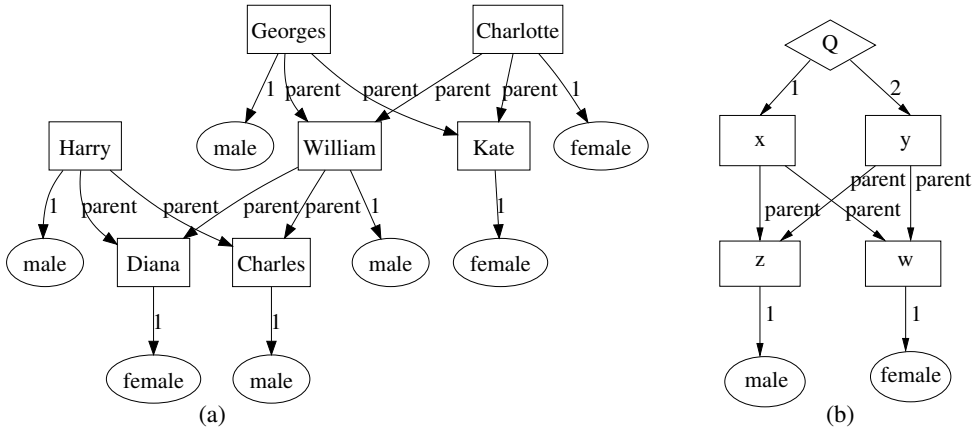

FIGURE 1 – (a) Contexte graphe sur la famille royale. Les rectangles sont les objets, les étiquettes d'arcs et d'ovales sont les attributs. (b) Un PGP définissant la relation binaire "frère-ou-sœur". Les rectangles sont des variables et le losange est le tuple de projection.

3 Graph-FCA : concepts formels d'un graphe de connaissances

Dans cette section, nous rappelons les principales définitions de Graph-FCA, une extension de l'analyse de concepts formels (FCA) (Ganter & Wille, 1999) aux graphes de connaissances, que nous avons introduite récemment (Ferré, 2015; Ferré & Cellier, 2016). Ces définitions sont illustrées par un petit exemple sur la famille royale anglaise.

Définition 1 (contexte graphe)
Un contexte graphe *est un triplet $K = (O, A, I)$, où O est l'ensemble des* objets, *A est l'ensemble des* attributs, *et $I \subseteq O^* \times A$ est une* relation d'incidence *entre des tuples d'objets (O^*) et des attributs.*

Un *contexte graphe* modélise un multi-hypergraphe orienté où les nœuds sont les *objets*, les étiquettes d'arcs sont les *attributs*, et les arcs sont des *incidences* $(\overline{o}, a)$ (aussi notée $a(\overline{o})$) entre tuples d'objets et attributs. L'aspect "orienté" vient de l'utilisation de tuples pour les arcs et l'aspect "hyper" vient de leur longueur quelconque. L'aspect "multi-" vient du fait qu'un même tuple peut être en incidence avec plusieurs attributs. La figure 1.(a) montre une représentation graphique d'un petit exemple. Des exemples d'arcs sont $male(William)$ et $parent(William, Diana)$. Différentes sortes de graphes de connaissances peuvent être traduit en contexte graphe : ex., graphes RDF, graphes conceptuels (Chein & Mugnier, 2008), contextes RCA (Rouane-Hacene *et al.*, 2013). Pour RDF, on a les traductions suivantes où o_x est l'objet représentant le nœud RDF x : triplets $\langle x \text{ a } c\rangle \mapsto a(o_x)$ (les classes deviennent des attributs unaires), autres triplets $\langle x\ p\ y\rangle \mapsto p(o_x, o_y)$ (les propriétés deviennent des attributs binaires), URI $u \mapsto u(o_u)$ et litéral $s \mapsto s(o_s)$ (les identités/valeurs deviennent des attributs unaires).

Définition 2 (pattern de graphe (projeté))
Un pattern de graphe *$P \subseteq \mathcal{V}^* \times A$ représente une généralisation d'un contexte graphe en abstrayant les nœuds-objets par des nœuds-variables pris dans un ensemble infini $\mathcal{V}$. Un* pattern de graphe projeté *(PGP) est un couple $Q = (\overline{x}, P)$ où P est un pattern de graphe et $\overline{x}$ est un tuple de projection sur certains nœuds-variables du pattern.*

Un PGP est analogue à une requête SPARQL de type SELECT-WHERE. La figure 1.(b) montre un PGP définissant la relation binaire "frère-ou-sœur" comme deux personnes partageant un père et une mère. Pour tout tuple d'objets $\overline{o} \in O$ d'un contexte $K = (O, A, I)$, l'expression $Q(\overline{o}) = (\overline{o}, I)$ dénote un PGP où les objets doivent être pris comme des variables et qui représente la description du tuple d'objets $\overline{o}$ par le graphe de connaissances K. Ainsi, dans l'exemple de la famille royale, chaque individu est décrit par le graphe entier, mais chacun d'un point de vue différent. Par exemple, William est un homme qui a un garçon et une fille qui ont tous deux une même mère, et qui a un père et une mère qui ont un fils en commun.

Les PGPs sont partiellement ordonnées par une relation d'inclusion : $Q_1 \subseteq_q Q_2$ ssi il existe un homomorphisme de graphe ϕ (Hahn & Tardif, 1997) du pattern de Q_1 vers le pattern de Q_2 qui préserve le tuple de projection. On note $dom(\phi)$ le domaine de définition d'un homomorphisme ϕ. On appelle *homset* Φ un ensemble d'homomorphismes ayant un même domaine. On note $homs_K(P)$ l'ensemble des homomorphismes d'un pattern P dans un contexte graphe K. La jointure de deux homsets est défini par $\Phi_1 \bowtie \Phi_2 = \{\phi_1 \cup \phi_2 \mid \phi_1 \in \Phi_1, \phi_2 \in \Phi_2, \phi_1 \sim \phi_2\}$, où $\phi_1 \sim \phi_2$ ssi pour tout $x \in dom(\phi_1) \cap dom(\phi_2)$, $\phi_1(x) = \phi_2(x)$. L'intersection de deux PGPs Q_1, Q_2 est défini par l'opérateur $\cap_q$, basé sur le produit de graphe dit catégorique (Hahn & Tardif, 1997), et donne le plus grand PGP qui soit inclu au sens de $\subseteq_q$ dans Q_1 et Q_2.

Définition 3 (concept graphe)
Soit $K = (O, A, I)$ un contexte graphe. Un concept graphe *de K est une paire (R, Q), constituée d'une relation (*extension*) et d'un PGP (*intension*), tel que $R = \{\overline{o} \mid Q \subseteq_q Q(\overline{o})\}$ et $Q \equiv_q \cap_q \{Q(\overline{o})\}_{\overline{o} \in R}$. L'*arité *d'un concept est la longueur du tuple de projection de son intension, et donc aussi la longueur des tuples de son extension.*

Les concepts sont définis de façon similaire à la FCA classique mais avec des PGP à la place des ensembles d'attributs et avec des *relations*, c'est-à-dire des ensembles de tuples d'objets, à la place d'ensembles d'objets. Un résultat important est que l'ensemble des concepts de même arité k forme un treillis $(\mathcal{C}_k, \leq, \wedge, \vee)$ où $C_1 \leq C_2$ signifie que C_1 est plus spécifique que C_2, c'est-à-dire a une plus petite extension et une plus grande intension. Dans ce papier, on se limite aux concepts unaires ($k = 1$) dont les extensions sont des ensembles d'objets, même si les résultats présentés s'appliquent également aux concept n-aires. Un exemple de concept est le couple $C_{ex} = (\{Charlotte, George, Harry, William\}, (x, \{parent(x, f), male(f), parent(x, m), female(m), parent(y, f), parent(y, m), male(y)\}))$. Il représente le concept des "enfants", lesquels, dans ce contexte, ont tous un père et une mère connus, lesquels ont toujours un fils.

4 Concepts de (plus proches) voisins

L'idée de départ est de remplacer la notion de distance numérique par une notion de distance conceptuelle.

Définition 4 (distance conceptuelle)
Soit $K = (O, A, I)$ un contexte graphe. La distance conceptuelle *entre deux objets $u, v \in O$ est le plus petit concept graphe unaire qui contient ces deux objets, c'est-à-dire le concept $\delta(u, v) = (R, Q)$ où le PGP $Q = Q(u) \cap_q Q(v)$ représente tout ce que u et v ont en commun et où la relation R englobe tous les objets qui partagent cette description commune.*

La distance conceptuelle vérifie les propriétés d'une distance si on prend l'ordre partiel $\leq$ sur les concepts et le supremum $\vee$ de concepts comme addition :

1. (positivité) $\delta(u,u) \leq \delta(u,v)$; ($\delta(u,u)$ joue le rôle de distance zéro)
2. (symétrie) $\delta(u,v) = \delta(v,u)$;
3. (inégalité triangulaire) $\delta(u,w) \leq \delta(u,v) \vee \delta(v,w)$.

On notera qu'une inégalité entre concepts $C_1 \leq C_2$ implique une inclusion entre leurs extensions $C_1.ext \subseteq C_2.ext$ et donc une inégalité entre le cardinal de ces extensions : $|C_1.ext| \leq |C_2.ext|$. La réciproque n'est pas vraie car $\leq$ est un ordre partiel sur les concepts. On peut donc utiliser les cardinaux d'extensions comme forme numérique – et dégradée – des distances conceptuelles : $d(u,v) := |\delta(u,v).ext|$. Cette forme numérique mesure en quelque sorte le nombre d'objets qui se trouvent entre u, v. On peut faire une observation similaire avec les intensions des concepts : $C_1 \leq C_2$ implique $|C_1.int| \geq |C_2.int|$. On peut donc utiliser la taille des intensions comme mesure de similarité : $sim(u,v) := |\delta(u,v).int|$.

Définition 5 (concepts de voisins)
Soit $K = (O, A, I)$ un contexte graphe et $u \in O$ un objet. Les concepts de voisins *de l'objet requête u (CN pour "concepts of neighbours") sont l'ensemble des distances conceptuelles partant de u, c'est-à-dire*

$$CN(u) := \{\delta(u,v) \mid v \in O\}.$$

L'extension propre d'un concept de voisins $\delta \in CN(u)$ est l'ensemble des objets qui se trouvent exactement à la distance δ de u : $\delta.proper := \{v \in O \mid \delta(u,v) = \delta\}$.

Le concept de voisins à distance zéro $\delta(u,u)$ englobe, en plus de u, les objets w dont la description contient celle de u ($Q(u) \subseteq_q Q(w)$). Comme il est inférieur à tous les autres concepts de voisins (positivité), on peut lui attribuer le rang 0, puis définir le rang de tout autre concept de voisins $\delta \in CN(u)$ par $rank(\delta) := 1 + max\{rank(\delta') \mid \delta' \in CN(u), \delta' < \delta\}$. On peut ensuite définir les concepts de plus proches voisins comme les concepts de rang 1, c'est-à-dire ceux qui sont immédiatement supérieurs au concept zéro. Un avantage de notre approche est de pouvoir définir l'ensemble des k plus proches voisins comme les instances de ces concepts de rang 1, sans avoir à fixer de valeur pour k. Un autre avantage est que ces plus proches voisins sont partitionnés en un ou plusieurs concepts, chacun apportant une justification via son intension.

Par exemple, le concept C_{ex} de la section précédente est la distance conceptuelle entre $Charlotte$ et $William$ (ils ont en commun d'avoir un père et une mère partageant un fils) et fait donc partie des concepts de voisins de chacun des deux individus. Pour $u = William$, l'extension propre est réduite à $\{Charlotte\}$ car $George$ et $Harry$ ont aussi en commun avec $William$ d'être des garçons. Pour $u = Charlotte$, l'extension propre est $\{Harry, William\}$ car $George$ a aussi en commun avec $Charlotte$ d'avoir des grands-parents dans le contexte graphe. Les k plus proches voisins de $William$ sont : (1) $Kate$ (parent d'un garçon et d'une fille), (2) $Charles$ (père), et (3) $Harry$ et $George$ (fils). Dans ce cas $k = 4$.

5 Algorithme

D'après les définitions de la section 4, l'algorithme naïf pour énumérer l'ensemble des concepts de voisins $CN(u)$ est le suivant.

Algorithme 1 (énumération naïve)
Require: *un graphe contexte $K = (O, A, I)$, un objet $u \in O$*
Ensure: *l'ensemble de concepts de voisins CN*

1: $CN \leftarrow \emptyset$
2: ***for all*** $v \in O$ ***do***
3: $Q \leftarrow Q(u) \cap_q Q(v)$ *// calcul intension*
4: $R \leftarrow \{o \in O \mid Q \subseteq_q Q(o)\}$ *// calcul extension*
5: $CN \leftarrow CN \cup \{\delta\}$ *where* $\delta = (R, Q)$
6: $\delta.proper \leftarrow \delta.proper \cup \{v\}$ *// v est dans l'extension propre de δ*
7: ***end for***

Dans le cas des graphes de connaissances de type Web sémantique, on doit supposer que le graphe a une seule composante connexe et donc que les descriptions des objets $Q(u), Q(v)$ sont de la taille du graphe. Le calcul de Q par l'intersection $\cap_q$ est donc quadratique dans le nombre d'arcs du contexte graphe, sans compter l'extraction du sous-pattern équivalent minimal (noyau) du pattern produit, ce qui est NP-complet. Le calcul de l'extension R implique un test d'inclusion $\subseteq_q$ NP-complet [1] pour chaque objet et donc pour chaque nœud du graphe. Le tout doit être calculé pour chaque objet.

Tout cela rend l'utilisation de cet algorithme naïf impratiquable et la complexité en NP-complet des opérations sur les graphes semble laisser peu d'espoir. Il est tentant de simplifier le problème en restreignant la description des objets à une certaine profondeur et en se ramenant à des arbres (pas de cycles), voire à des ensembles de chaines (pas de branchements). Cependant, cela signifirait que l'on ne travaille plus vraiment sur des graphes, mais sur des strucures plus pauvres (ex., arbres, chaines). Notre objectif est de calculer les concepts de voisins directement sur le graphe et sans restriction de profondeur ou autre. La nature des graphes de connaissances et des distances conceptuelles rend la tâche moins complexe qu'elle ne l'est dans le cas général :

1. les graphes de connaissances sont des (hyper)graphes *orientés* et *étiquetés*, ce qui pose des contraintes fortes sur les homomorphismes et réduit donc leur combinatoire ;
2. les descriptions d'objets et les intensions de concepts sont des PGPs, i.e. des graphes *centrés* sur un nœud qui sert d'ancrage et contraint encore davantage les homomorphismes.

Pour rendre pratiquable le calcul des concepts de voisins, nous combinons deux techniques originales. La première technique (section 2) consiste à énumérer ces concepts en partitionnant de façon de plus en plus fine l'ensemble des objets, plutôt que de traiter les objets un à un. Cela permet de grandement factoriser le calcul des intensions et extensions de concepts. La deuxième technique (section 3) consiste en une forme "paresseuse" de jointure entre homsets pour le calcul des extensions. Elle permet de représenter de façon compacte le homset d'un pattern dont la taille peut rapidement exploser.

5.1 Énumération par partitionnement

Le parti pris est de discriminer de plus en plus finement l'ensemble des objets par partitionnements successifs, en formant des sous-PGP de plus en plus grands de $Q(u)$. Un *pré-concept* matérialise une étape de ce processus avec une intension susceptible d'être agrandie et avec une extension propre susceptible d'être partitionnée davantage.

1. Il s'agit du problème de l'existence d'un homomorphisme entre deux graphes (Hahn & Tardif, 1997).

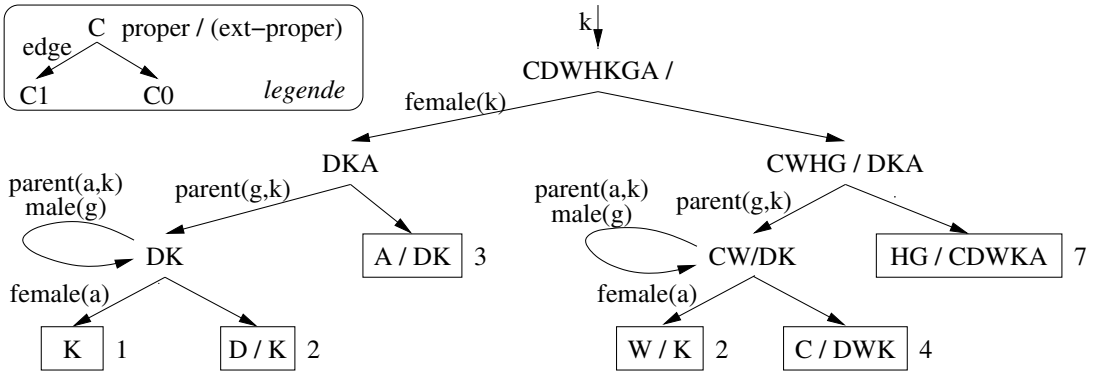

FIGURE 2 – Énumération par partition pour le contexte de la famille royale et avec $u = Kate$.

Définition 6 (pré-concept)
Un pré-concept *est une structure* $C = (pattern, homs, proper, edges)$*, où :*
- $pattern \subseteq I$ *est un sous-graphe connexe du contexte graphe contenant le nœud* u *;*
- $homs = homs_K(pattern)$ *représente l'ensemble des occurences du pattern dans* K *;*
- $proper \subseteq O$ *est l'extension propre du pré-concept ;*
- $edges \subseteq I$ *est l'ensemble des arcs restant disponibles pour partitionner le pré-concept.*

On définit l'extension et l'intension du pré-concept à partir du pattern et du homset en les projetant sur u *:* $int := (u, pattern)$ *;* $ext := \{\phi(u) \mid \phi \in homs\}$.

Le processus de partitionnement commence avec le pattern $\{\top(u)\}$, où $\top(u)$ est l'arc neutre qui ne pose aucune contrainte sur u, et l'ensemble O comme extension propre. Ensuite, chaque pré-concept peut être partitionné en deux pré-concepts par tout arc disponible et connexe au pattern. Le processus de partitionnement s'arrête pour un pré-concept quand son extension propre est vide ou bien quand il n'y a plus d'arcs disponibles.

Algorithme 2 (énumération par partitionnement)
Require: *un graphe contexte* $K = (O, A, I)$*, un objet requête* $u \in O$
Ensure: *un ensemble de pré-concepts de voisins* CN
1: $CN \leftarrow \{C_{init} := (\emptyset, homs_K(\{\top(u)\}), O, I)\}$
2: ***while*** *il existe* $C \in CN$ *et un arc* $e = (\overline{x}, a) \in C.edges$ *connexe avec* $C.pattern$ ***do***
3: $\quad C_1 = (C.pattern \cup \{e\},\ C.homs \bowtie homs_K(\{e\}),\ C.proper \cap C_1.ext,\ C.edges \setminus \{e\})$
4: $\quad C_0 = (C.pattern,\ C.homs,\ C.proper \setminus C_1.proper,\ C.edges \setminus \{e\})$
5: $\quad CN \leftarrow CN \setminus \{C\}$
6: $\quad$ **if** $C_1.proper \neq \emptyset$ **then** $CN \leftarrow CN \cup \{C_1\}$
7: $\quad$ **if** $C_0.proper \neq \emptyset$ **then** $CN \leftarrow CN \cup \{C_0\}$
8: ***end while***

La figure 2 illustre cet algorithme avec le calcul des concepts de voisins de Kate dans le contexte de la famille royale. Chaque pré-concept est représenté par son extension, partagée entre son extension propre et le reste. Les lettres représentent les 7 personnes par leur initiale (sauf A pour Charlotte). Les lettres minuscules dans les arcs représentent l'abstraction de ces personnes par des variables. Le pattern de l'intension d'un pré-concept est l'ensemble des arcs sur le chemin menant à ce pré-concept. Les concepts de voisins, ceux qui ne peuvent être par-

titionnés davantage, sont encadrés et la taille de leur extension est affichée comme distance numérique. On trouve 5 concepts de voisins en plus du concept zéro K. Par ordre de distance croissante, on a le concept D/K des mères avec Diana, le concept W/K des parents d'un garçon et d'une fille avec William, le concepts A/DK des personnes de sexe féminin avec Charlotte en plus de Diana, le concept C/DWK des parents avec Charles en plus de Diana et William, et enfin le concept $GH/CDWKA$ de ceux qui ne partagent rien avec Kate.

Comparé à l'algorithme naïf, cet algorithme peut être incomplet au sens où certains concepts de voisins peuvent ne pas être produits, conduisant à placer certains objets à une plus grande distance conceptuelle qu'ils ne le sont réellement. En effet, les patterns générés P sont des sous-ensemble de I, ce qui contraint les homomorphismes de P vers I à être injectifs, c'est-à-dire que deux variables de P ne peuvent pas se projeter sur un même objet. Par exemple, si $Q(u) = (u, \{p(u, u_1), a(u_1), b(u_1)\})$, alors on va générer les patterns $P_{ab} = \{p(u, u_1), a(u_1), b(u_1)\}$, $P_a = \{p(u, u_1), a(u_1)\}$ et $P_b = \{p(u, u_1), b(u_1)\}$, mais pas le pattern $P_* = \{p(u, u'_1), a(u'_1), p(u, u''_1), b(u''_1)\}$. Si des objets contiennent P_* mais pas P_{ab}, alors ils seront placés dans l'extension propre de P_a ou P_b. Rétablir la complétude suppose de permettre la duplication de variable dans un pattern, ce qui accroît considérablement l'espace de recherche. La forme actuelle de l'algorithme a aussi l'avantage de faciliter l'interprétation des intensions de concepts car ce sont des sous-ensembles de la description de l'objet u.

L'énumération par partitionnement offre plusieurs avantages importants. Tout d'abord, grâce au partitionnement des extensions propres, le nombre de pré-concepts est borné à tout moment par le nombre d'objets, alors même que le nombre de patterns possibles est exponentiel avec le nombre d'arcs. Ensuite, l'algorithme offre beaucoup de flexibilité. Les pré-concepts peuvent être partitionnés dans n'importe quelle ordre, autorisant des stratégie en profondeur d'abord ou en largeur d'abord ou l'emploi d'une heuristique. Le choix de l'hyperarc est également libre et indépendant d'un pré-concept à l'autre. Des optimisations peuvent être appliquées telles que : éliminer u de l'extension propre initiale, stopper le partitionnement sur les pré-concepts dont l'extension propre est un singleton. De plus, on peut tout de même utiliser cet algorithme pour calculer la distance conceptuelle avec un objet donné v, simplement en posant $C_{init}.proper = \{v\}$. Enfin, et c'est peut-être le plus important, l'algorithme est *anytime* puisqu'une partition des objets en concepts est définie à tout moment. Cette partition est simplement moins fine si on arrête le processus avant la fin. Pour un parcours en largeur, cela permet de limiter la profondeur d'exploration du graphe par un temps de calcul plutôt que par une profondeur limite.

5.2 Jointure "paresseuse"

Le calcul explicite de l'ensemble $C.homs$ des homomorphismes d'un pattern par jointures successives peut être impratiquable, même dans des cas simples. Par exemple, le pattern $\{film(u), acteur(u, u_1), ..., acteur(u, u_n)\}$ aura de l'ordre de Nn^n homomorphismes, en considérant qu'il y a N films, chacun relié à n acteurs. Pour 1000 films reliés chacun à 10 acteurs, cela fait déjà 10^{13} homomorphismes ! Ce nombre peut diminuer quand des contraintes sont ajoutées aux acteurs, mais peut aussi augmenter de façon exponentielle avec l'ajout de relations, par exemple des acteurs vers leurs films. Il est possible de faire mieux car ce qui nous intéresse *in fine* est $C.ext \subseteq O$, qui est de taille bornée par le nombre d'objets. L'idée est de représenter le homset $C.homs$ par une structure contenant plusieurs jointures locales au lieu d'une jointure globale, en joignant juste ce qu'il faut pour que $C.ext$ soit correct par rapport à $C.pattern$.

Définition 7 (arbre d'homsets)
Un homset factorisé *est une structure arborescente* $\psi = (e, D, \Phi, \Delta, \Psi)$ *où :*
— e *est un hyperarc et* $vars(e)$ *est l'ensemble de ses nœuds-variables ;*
— $D \subseteq vars(e)$ *est l'ensemble des variables introduites par cet hyperarc ;*
— Φ *est un homset dont le domaine contient* $vars(e)$ *;*
— $\Delta \subseteq dom(\Phi)$ *est le sous-domaine utile aux structures englobantes ;*
— Ψ *est l'ensemble des sous-structures arborescentes (structures filles).*

Le homset explicite est égal à la jointure des homsets de l'ensemble des (sous-)structures de l'arbre d'homsets. L'arbre d'homsets du pré-concept initial correspond à l'arc neutre $\top(u)$.

L'*arbre d'homsets initial* correspond au pré-concept initial C_{init} et est défini comme $\psi_{init} := (\top(u), \{u\}, homs_K(\{\top(u)\}), \{u\}, \emptyset)$. Lors de la partition d'un pré-concept, la jointure d'homsets $C.homs \bowtie homs(\{e^*\})$ doit être remplacée par l'évaluation de $lazyjoin(\psi, \psi^*)$ (voir Algorithm 3), avec $\psi = C.homs$ et $\psi^* = (e^*, vars(e^*) \setminus vars(C.pattern), homs_K(\{e^*\}), vars(e^*) \cap vars(C.pattern), \emptyset)$. Cet jointure "paresseuse" consiste à insérer la structure feuille ψ^* correspondant au nouvel arc dans l'arbre d'homsets ψ.

Algorithme 3 (Définition récursive de la fonction $lazyjoin(\psi, \psi^*)$**)**
Require: *deux homset factorisés* $\psi = (e, D, \Phi, \Delta, \Psi)$ *et* $\psi^* = (e^*, D^*, \Phi^*, \Delta^*, \Psi^*)$
Ensure: *deux ensembles* $\Delta^+, \Delta^- \subseteq \mathcal{V}$*, un nouvel homset factorisé* $\psi' = (e, D, \Phi', \Delta', \Psi')$
1: $\Delta^+ \leftarrow \emptyset$; $\Delta^- \leftarrow \emptyset$; $\Phi' \leftarrow \Phi$; $\Psi' \leftarrow \emptyset$; $inserted \leftarrow false$
2: ***for all*** $\psi_c \in \Psi$ ***do***
3: $\Delta_c^+, \Delta_c^-, \psi_c' \leftarrow lazyjoin(\psi_c, e^*)$ *where* $\psi_c' = (e_c, D_c, \Phi_c', \Delta_c', \Psi_c')$
4: $\Delta^+ \leftarrow \Delta^+ \cup \Delta_c^+$; $\Delta^- \leftarrow \Delta^- \cup \Delta_c^-$; $\Phi' \leftarrow \Phi' \bowtie \pi_{\Delta_c'} \Phi_c'$; $\Psi' \leftarrow \Psi' \cup \{\psi_c'\}$
5: ***end for***
6: ***if*** $D \cap \Delta^* \neq \emptyset$ ***then***
7: ***if*** $\neg inserted$ ***then***
8: $\Delta^- \leftarrow \Delta^- \cup (\Delta^* \setminus D)$; $\Phi' \leftarrow \Phi' \bowtie \pi_{\Delta^*} \Phi^*$; $\Psi' \leftarrow \Psi' \cup \{\psi^*\}$; $inserted \leftarrow true$
9: ***else***
10: $\Delta^+ \leftarrow \Delta^+ \cup (\Delta^* \cap D)$
11: ***end if***
12: ***end if***
13: $\Delta^+, \Delta^- \leftarrow \Delta^+ \setminus \Delta^-,\ \Delta^- \setminus \Delta^+$
14: $\Delta' \leftarrow \Delta' \cup \Delta^+ \cup \Delta^-$
15: ***return*** $\Delta^+, \Delta^-, \psi'$

La figure 3 donne un exemple de jointure paresseuse partant de l'arbre d'homsets du PGP $(k, \{female(k), parent(g, k), male(g), parent(g, w), parent(a, k), female(a)\})$ et ajoutant l'arc $e^* = parent(a, w)$ qui forme un cycle en joignant a et w. Les modifications sont propagées à partir des deux sous-structures définissant (voir D) les variables a, w et la nouvelle sous-structure ψ^* est insérée dans l'une d'elle (ici, celle définissant a). Les insertions des autres arcs, menant au premier arbre de la figure, ne modifient qu'une branche de l'arbre à la fois car elles n'introduisent pas de cycle. Dans l'algorithme 3, le calcul des Δ^+, Δ^- permet de déterminer jusqu'où propager dans les Δ les variables définies dans les autres branches que la branche d'insertion (ici, la variable w) pour correctement prendre en compte le cycle dans le calcul des homsets.

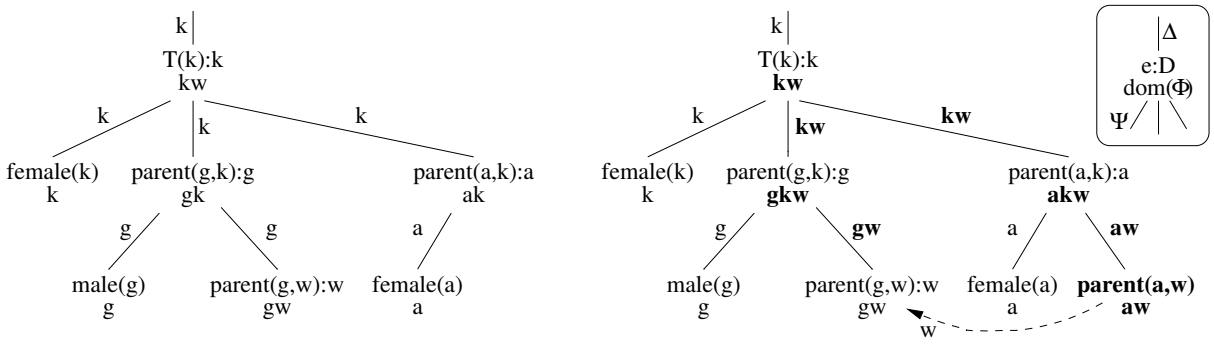

FIGURE 3 – Arbre d'homsets avant et après jointure paresseuse avec l'arc $parent(a, w)$.

Si on reprend l'exemple des films reliés à leurs acteurs, on constate que chaque nouvel arc $e^* = acteur(u, u_i)$ n'entraîne que la projection $\Phi_{e^*} = \pi_{\{u\}} homs_K(\{e^*\})$, c'est-à-dire le calcul du domaine d'une relation binaire (ici, les films ayant un acteur), puis la jointure $\Phi_{\top(u)} \bowtie \Phi_{e^*}$, c'est-à-dire une intersection de deux ensembles d'objets (ici, des films). L'arbre d'homsets final contient donc un homset de cardinal N (arc $\top(u)$) et n homsets de cardinal Nn (arcs $acteur(u, u_i)$), soit un total de l'ordre de Nn^2 homomorphismes au lieu de Nn^n. Pour 1000 films reliés à 10 acteurs chacun, cela fait 10^5 au lieu de 10^{13} ! De plus, les homomorphismes portent sur 1 ou 2 variables au lieu de $(n + 1)$.

6 Implémentation et premières expérimentations

Nous avons implémenté les algorithmes ci-dessus en ∼700 lignes de code OCaml et les avons intégré dans SEWELIS [2] comme amélioration d'UTILIS (Hermann *et al.*, 2012) pour la saisie guidée de descriptions RDF. UTILIS est aussi une méthode de plus proches voisins, mais basé sur une distance numérique et ne traitant ni les cycles, ni l'exploration profonde du graphe de connaissances. Les graphes de connaissances sont ici des graphes RDF. Dans notre implémentation, nous avons adopté les heuristiques suivantes : exploration en largeur d'abord de l'arbre de partitionnement des concepts de voisins ; choix de l'arc de partitionnement favorisant à la fois les arcs les plus proches de l'objet requête u et la diversité des prédicats dans les patterns. Le dernier critère permet d'éviter, par exemple, de considérer 10 acteurs d'un film avant même d'avoir considéré d'autres critères tels que le directeur ou la date de sortie.

Nous avons conduit de premières expérimentations de l'approche CNN sur deux versions de la base MONDIAL (May, 1999), qui contient des données géographiques (ex., pays, continents, rivières, montagnes, langues, groupes ethniques). La version *complète* est un contexte graphe de 41577 nœuds et 120546 arcs. La version *réduite* en est un sous-graphe de 9692 nœuds et 11691 arcs, soit avec environ 5 fois moins de nœuds et 10 fois moins d'arcs. Elle a été obtenue en éliminant les données numériques car elles sont pour l'instant traitées comme des données catégorielles et ne permettent donc guère de trouver des points communs entre objets [3]. La figure 4 montre pour les deux versions de MONDIAL l'évolution de deux mesures selon le temps de calcul alloué (*timeout*), allant de 1s à 200s. La première mesure (graphique gauche) est le

2. `http://www.irisa.fr/LIS/softwares/sewelis/`

3. Par exemple, il y a peu de chances que deux pays aient exactement la même population.

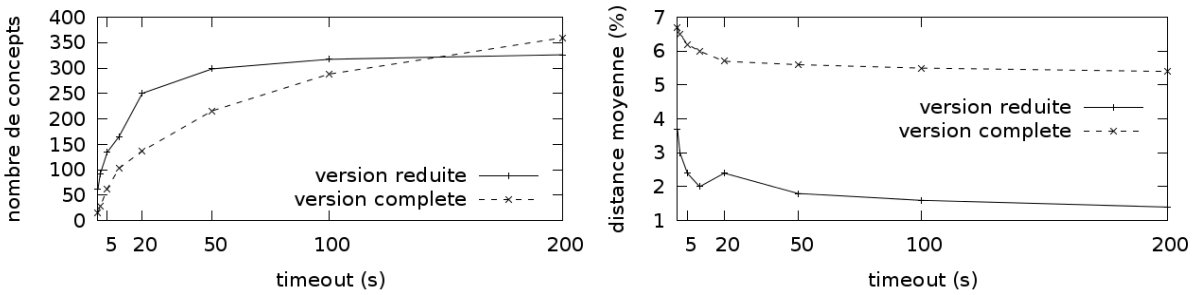

FIGURE 4 – Évolution du nombre de concepts de voisins (à gauche) et de la distance numérique moyenne des objets (à droite) en fonction du *timeout* et de la version de MONDIAL.

nombre de concepts de voisins obtenus. La deuxième mesure (graphique droit) est la distance numérique $d(u, v) = |\delta(u, v).ext|$ moyenne entre l'objet requête u et les autres objets v. Cette dernière est la moyenne des tailles des extensions des concepts de voisins, pondérée par le nombre d'objets dans l'extension propre de chaque concept. La distance numérique est donnée en pourcentage de la distance maximale possible, à savoir le nombre d'objets du jeu de données. Ces mesures sont moyennées sur un échantillon de 100 objets pour les timeouts inférieurs à 20 et sur 10 objets au-delà. Nous avons pu vérifié que ces mesures sont peu sensibles à la taille de cet échantillon : $\pm 30\%$ au maximum en faisant varier cette taille de 1 à 100.

La forme asymptotique des courbes de la figure 4 montre la validité de l'utilisation *anytime* de notre algorithme puisque la plupart des concepts sont produits rapidement et que la distance moyenne décroît d'abord rapidement puis très lentement. Le nombre de concepts est assez grand pour être discriminant et en même temps assez petit comparé au nombre d'objets pour démontrer une capacité d'abstraction. Nous avons aussi regardé le nombre de *plus proches* concepts : il évolue de 2 à 4 pour les deux versions. Les distances moyennes peuvent paraître faibles, mais elles correspondent tout de même à des centaines d'objets dans la version réduite et à des milliers d'objets dans la version complète. Toutes ces mesures montrent que l'on a pas d'écrasement du spectre des distances, ni vers le bas ni vers le haut. Enfin, nous avons évalué l'impact de nos deux optimisations en les désactivant. Si on utilise des jointures explicites plutôt que paresseuses, on produit environ 5 fois moins de concepts avec une distance moyenne environ 3 fois supérieure pour des timeouts de 10s et 100s. Si on applique notre algorithme à chaque objet v isolément ($C_{init}.proper = \{v\}$), en partageant un timeout de 100s entre les 9692 objets de la version réduite, on observe qu'on ne parvient à calculer une distance conceptuelle non-triviale (i.e., pattern non vide) que pour 16% des objets, et que la distance moyenne est 8 fois supérieure.

Pour illustrer notre approche, nous donnons ci-dessous quelques-uns des concepts de voisins les plus proches du pays France dans la version réduite, avec leurs extensions (même notation qu'en figure 2) et leurs intensions (informellement par souci de clarté et de concision) :

— Pologne : une république européenne, voisine de l'Allemagne ;
— Italie : une république européenne où on parle français ;
— Espagne : un pays européen sur l'Atlantique, ayant un voisin et des dépendances (ex., Melilla)
— Royaume-Uni / Espagne : un pays européen sur l'Atlantique avec des dépendences (ex., Cayman).
— Portugal / Espagne : un pays européen sur l'Atlantique, ayant un voisin et une ancienne colonie (ex., Cap vert).

7 Conclusion et perspectives

Nous avons introduit la notion de "concepts de voisins" comme forme conceptuelle de distance entre objets, où ces objets sont les nœuds d'un graphe de connaissances. Nous avons proposé un algorithme pour les calculer et montré qu'il était assez efficace pour être utilisé sur des données réelles et de taille conséquente. Ces résultats sont encore préliminaires et il reste surtout à explorer comment ces concepts de voisins peuvent être exploités au mieux pour des tâches d'apprentissage, supervisé ou non. Il reste des marges d'optimisation de l'algorithme et il sera nécessaire de mieux traiter les valeurs telles que nombres, dates ou chaines.

Références

BISSON G. (2000). La similarité : une notion symbolique/numérique. *Apprentissage symbolique-numérique*, **2**, 169–201.

CHEIN M. & MUGNIER M.-L. (2008). *Graph-based knowledge representation : computational foundations of conceptual graphs*. Advanced Information and Knowledge Processing. Springer.

DE MANTARAS R. L., MCSHERRY D., BRIDGE D., LEAKE D., SMYTH B., CRAW S., FALTINGS B., MAHER M. L., T. COX M., FORBUS K. *et al.* (2005). Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, **20**(03), 215–240.

FERRÉ S. (2015). A proposal for extending formal concept analysis to knowledge graphs. In J. BAIXERIES, C. SACAREA & M. OJEDA-ACIEGO, Eds., *Int. Conf. Formal Concept Analysis (ICFCA)*, LNCS 9113, p. 271–286 : Springer.

FERRÉ S. & CELLIER P. (2016). Graph-FCA in practice. In O. HAEMMERLÉ, G. STAPLETON & C. FARON-ZUCKER, Eds., *Int. Conf. Conceptual Structures (ICCS) - Graph-Based Representation and Reasoning*, LNCS 9717, p. 107–121 : Springer.

FERRÉ S. & RIDOUX O. (2002). The use of associative concepts in the incremental building of a logical context. In G. A. U. PRISS, D. CORBETT, Ed., *Int. Conf. Conceptual Structures*, LNCS 2393, p. 299–313 : Springer.

GANTER B. & WILLE R. (1999). *Formal Concept Analysis — Mathematical Foundations*. Springer.

HAHN G. & TARDIF C. (1997). Graph homomorphisms : structure and symmetry. In *Graph symmetry*, p. 107–166. Springer.

HERMANN A., FERRÉ S. & DUCASSÉ M. (2012). An interactive guidance process supporting consistent updates of RDFS graphs. In A. TEN TEIJE ET AL., Ed., *Int. Conf. Knowledge Engineering and Knowledge Management (EKAW)*, LNAI 7603, p. 185–199 : Springer.

HORVÁTH T., WROBEL S. & BOHNEBECK U. (2001). Relational instance-based learning with lists and terms. *Machine Learning*, **43**(1-2), 53–80.

KUZNETSOV S. (2013). Fitting pattern structures to knowledge discovery in big data. In P. CELLIER, F. DISTEL & B. GANTER, Eds., *Int. Conf. Formal Concept Analysis*, LNAI 7880, p. 254–266. Springer.

MAY W. (1999). *Information Extraction and Integration with* FLORID *: The* MONDIAL *Case Study*. Rapport interne 131, Universität Freiburg, Institut für Informatik. Available from `http://dbis.informatik.uni-goettingen.de/Mondial`.

MITCHELL T. (1997). *Machine Learning*. McGraw-Hill.

MUGGLETON S. (1995). Inverse entailment and progol. *New Generation Computation*, **13**, 245–286.

ROUANE-HACENE M., HUCHARD M., NAPOLI A. & VALTCHEV P. (2013). Relational concept analysis : mining concept lattices from multi-relational data. *Annals of Mathematics and Artificial Intelligence*, **67**(1), 81–108.