Une algèbre floue pour l'interrogation flexible de bases de données graphes

Olivier Pivert, Virginie Thion, Hélène Jaudoin, Grégory Smits

Une algèbre floue pour l'interrogation flexible de bases de données graphes

Olivier Pivert
Université Rennes 1, IRISA
Lannion, France
Olivier.Pivert@irisa.fr

Virginie Thion
Université Rennes 1, IRISA
Lannion, France
Virginie.Thion@irisa.fr

Hélène Jaudoin
Université Rennes 1, IRISA
Lannion, France
Helene.Jaudoin@irisa.fr

Grégory Smits
Université Rennes 1, IRISA
Lannion, France
Gregory.Smits@irisa.fr

ABSTRACT

Cet article décrit une algèbre de requête floue adaptée à l'interrogation flexible de bases de données graphes. Cette algèbre, fondée sur la théorie des ensembles flous et sur la notion de graphe flou, se compose d'un ensemble d'opérateurs permettant de formuler des requêtes à préférences sur des objets de type graphe, flous ou non. Les préférences exprimables dans ce cadre peuvent concerner i) le contenu des nœuds du graphe et/ou ii) la structure du graphe (qui peut inclure des arcs pondérés quand le graphe est flou). De même que l'algèbre relationnelle constitue la base du langage SQL utilisé dans les systèmes commerciaux, l'algèbre floue proposée ici est destinée à servir de fondement à l'extension d'outils plus orientés utilisateur tels que le langage Cypher implanté dans le système Neo4j.

Keywords

Bases de données graphes, requêtes floues, algèbre, graphes flous

De nombreux travaux ont été consacrés à l'interrogation floue de bases de données *relationnelles*, voir par exemple [DP96, ZDDK08, PB12], qui ont débouché notamment sur une extension floue du langage SQL, nommée SQLf [BP95]. Cependant, bien que les bases de données relationnelles soient encore largement utilisées, en particulier pour les applications classiques de gestion dans les entreprises, le besoin de gérer des données plus complexes s'est fait sentir depuis déjà plusieurs décennies, et a conduit à l'émergence d'autres modèles de données. Ainsi, un concept a fait son apparition ces dernières années, et a attiré l'attention de nombreux chercheurs de la communauté des bases de données, celui de *base de données graphe* [GS02, HS08, DSUBGV$^+$10, VMZ$^+$10, Ang12, CAH12, BT12], dont la finalité première est de permettre une gestion efficace de réseaux d'entités où chaque nœud est décrit par un ensemble de caractéristiques (par exemple un ensemble d'attributs décrivant l'entité, mais une structure de données plus complexe peut aussi être associée à un nœud) et les arcs représentent les liens de différentes natures existant entre les entités. Un tel modèle de données a de nombreuses applications potentielles, et peut servir par exemple à représenter des réseaux sociaux, des données RDF [AG05], des bases de données cartographiques, des bases de données bibliographiques, etc.

Le schéma des données associé à une base de données graphe est généralement complexe à appréhender par les utilisateurs, ce qui les amène à écrire des requêtes "en aveugle", dont les réponses peuvent souvent s'avérer vides ou pléthoriques. Dans ce contexte, il est nécessaire de proposer à l'utilisateur des moyens d'interroger la base de données de façon flexible.

Les bases de données graphes offrent de nombreuses possibilités en termes d'interrogation flexible puisque deux aspects peuvent intervenir dans les préférences exprimables par un utilisateur : i) le contenu des nœuds et ii) la structure du graphe lui-même. Dans cet article, nous nous attachons à définir une algèbre de requête floue adaptée à ce type de base de données. Nos points de départ sont, d'une part, l'article [Yag13] de R.R. Yager dans lequel ce dernier recense un certain nombre de critères de recherche flexibles pertinents dans un tel contexte, sans toutefois entrer dans les détails de leur expression à l'aide d'un langage de requête formel, et l'article [HS08] de H. He et A.K. Singh dans lequel les auteurs proposent une algèbre permettant d'interroger (sans préférences ni gradualité d'aucune sorte) des bases de données graphes.

La suite de l'article est organisée de la façon suivante. La section 1 présente des notions de base sur les graphes flous et les requêtes à préférences floues. La section 2 décrit les éléments principaux pouvant intervenir dans une requête floue dans un contexte de base de données graphe (critères sur les nœuds, conditions graduelles sur la structure du graphe, connecteurs flous). La section 3 présente une algèbre de requête floue permettant d'exprimer des préférences sur une base de données représentant un graphe (ou un ensemble de graphes) classique(s) ou flou(s). Les travaux connexes les plus pertinents sont discutés en section 4. Finalement, la section 5 rappelle les contributions principales et esquisse quelques perspectives.

BDA 14 octobre 2014, Grenoble-Autrans, France.

1. NOTIONS DE BASE

1.1 Bases de données graphes

Un système de gestion de bases de données graphes (souvent appelé simplement et abusivement *base de données graphe* dans la littérature) permet de gérer des données telles que la structure du schéma et les instances sont modélisées par des graphes (ou des généralisations de ce concept) et la manipulation des données se fait au moyen d'opérations orientées graphe et de constructeurs de type [AG08]. Parmi les systèmes de ce genre, on peut citer AllegroGraph [All], InfiniteGraph [Inf], Neo4j [Neo] et Sparksee [Spa]. Il existe différent modèles de bases de données graphes (voir [AG08] pour une vue d'ensemble), notamment le *graphe d'attributs* (appelé aussi *graphe de propriétés*) permettant de modéliser un réseau d'entités encapsulant des données. Dans ce modèle, les nœuds représentent les entités et les arcs correspondent à des relations entre entités. La modélisation des nœuds et des arcs peut faire intervenir des attributs décrivant leurs propriétés. La figure 1 est un exemple de base de données graphe contenant des données provenant de DBLP[1].

1.2 Ensembles flous et graphes flous

1.2.1 Rappels sur les ensembles flous

La théorie des ensembles flous a été définie par L.A. Zadeh [Zad65] pour modéliser des classes d'objets aux frontières vagues. Pour de tels ensembles, la transition entre l'appartenance complète et la non-appartenance est graduelle plutôt que tranchée. Des exemples typiques de classes floues sont celles associées à des adjectifs du langage naturel tels que *jeune, bon marché, rapide*, etc. Formellement, un ensemble flou F défini sur un référentiel U est caractérisé par une fonction d'appartenance $\mu_F : U \to [0, 1]$ où $\mu_F(u)$ désigne le degré d'appartenance de u à F. En particulier, $\mu_F(u) = 1$ reflète l'appartenance totale de u à F, tandis que $\mu_F(u) = 0$ exprime la non-appartenance absolue. Lorsque $0 < \mu_F(u) < 1$, on parle d'appartenance partielle.

Deux ensembles classiques présentent un intérêt particulier lorsque l'on définit un ensemble flou F:

- le noyau $C(F) = \{u \in U \mid \mu_F(u) = 1\}$, constitué des *prototypes* de F,
- le support $S(F) = \{u \in U \mid \mu_F(u) > 0\}$.

La notion de coupe de niveau, ou α-coupe englobe ces deux concepts. L'α-coupe (resp. α-coupe stricte) F_α (resp. $F_{\overline{\alpha}}$) d'un ensemble flou F correspond à l'ensemble des éléments du référentiel qui possèdent un degré d'appartenance à F au moins égal à (resp. strictement plus grand que) α :

$$F_\alpha = \{u \in U \mid \mu_F(u) \geq \alpha\} \text{ et } F_{\overline{\alpha}} = \{u \in U \mid \mu_F(u) > \alpha\}.$$

De façon directe, on a : $C(F) = F_1$ et $S(F) = F_{\overline{0}}$.

En pratique, la fonction d'appartenance associée à un ensemble flou F est souvent choisie de forme trapézoïdale. Ainsi, F peut se représenter par le quadruplet (A, B, a, b) où $C(F) = [A, B]$ et $S(F) = [A - a, B + b]$, voir la figure 2.

Soit F et G deux ensembles flous définis sur l'univers U. On dit que $F \subseteq G$ ssi $\mu_F(u) \leq \mu_G(u)$, $\forall u \in U$. Le complément de F, noté F^c, est défini par $\mu_{F^c}(u) = 1 - \mu_F(u)$. De plus, $F \cap G$ (resp. $F \cup G$) est défini de la

[1] `http://www.informatik.uni-trier.de/~ley/db/`

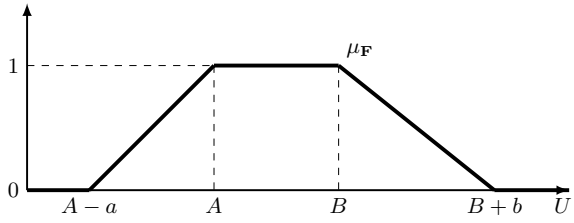

Figure 2: Fonction d'appartenance trapézoïdale

façon suivante : $\mu_{F\cap G} = min(\mu_F(u), \mu_G(u))$ (resp. $\mu_{F\cup G} = max(\mu_F(u), \mu_G(u))$).

Comme dans le cas booléen, les contreparties logiques des opérateurs ensemblistes $\cap$, $\cup$ et la complémentation sont respectivement la conjonction $\wedge$, la disjonction $\vee$ et la négation $\neg$. Voir [DP00] pour de plus amples détails.

1.2.2 Graphes flous

Un graphe est un couple (V, R), où V est un ensemble et R est une relation sur V. Les éléments de V (resp. R) sont les nœuds (resp. arcs) du graphe. Une relation floue ρ sur un ensemble V peut être vue comme définissant un graphe pondéré, ou graphe flou [Ros75, MN00], où un arc $(x, y) \in V \times V$ a un *poids* ou *force* de valeur $\rho(x, y) \in [0, 1]$. Ce *degré* peut exprimer l'intensité de n'importe quelle relation graduelle entre deux nœuds.

REMARQUE 1. *Le graphe peut être flou au départ — c-à-d que la relation ρ est connue au départ — ou peut être issu d'un traitement le rendant flou. Un graphe flou peut modéliser des relations statiques ou dynamiques. Si un graphe flou modélise un réseau social de style Twitter et notamment les relations entre utilisateurs de ce réseau, $\rho(x, y)$ peut par exemple être défini en fonction du nombre d'articles de y que x a retransmis.*

La relation floue ρ peut être vue comme un sous-ensemble flou sur $V \times V$, ceci permettant d'utiliser les formalismes des ensembles flous [Yag13]. On peut ainsi dire que $\rho_1 \subseteq \rho_2$ si $\forall(x, y)$, $\rho_1(x, y) \leq \rho_2(x, y)$.

Quelques propriétés d'intérêt peuvent êtres associées aux relations floues, comme la réflexivité ($\rho(x, x) = 1$, $\forall x$), la symétrie ($\rho(x, y) = \rho(y, x)$), ou encore la transitivité ($\rho(x, z) \geq \max_y \min(\rho(x, y), \rho(y, z))$).

La composition est une importante opération associée aux relations floues. Supposons que ρ_1 et ρ_2 soient deux relations floues sur V. Alors, la composition $\rho = \rho_1 \circ \rho_2$ est une relation floue sur V tq. $\rho(x, z) = \max_y \min(\rho_1(x, y), \rho_2(y, z))$. La composition est associative : $(\rho_1 \circ \rho_2) \circ \rho_3 = \rho_1 \circ (\rho_2 \circ \rho_3)$. Cette propriété d'associativité permet d'utiliser la notation $\rho^k = \rho \circ \rho \circ \ldots \circ \rho$ pour représenter la composition de ρ avec elle-même $k - 1$ fois. En complément, on définit ρ^0 comme étant $\rho^0(x, y) = 0$, $\forall(x, y)$ [Yag13]. Si ρ est réflexive alors $\rho^{k_2} \supseteq \rho^{k_1}$ pour $k_2 > k_1$. Si ρ est transitive, on peut montrer que $\rho^{k_2} \subseteq \rho^{k_1}$ si $k_2 > k_1$. On peut montrer que si ρ est réflexive et transitive alors $\rho^{k_2} = \rho^{k_1}$ pour tous k_1 et $k_2 \neq 0$.

REMARQUE 2. *Les graphes flous comme définis ci-dessus peuvent être généralisés au cas des graphes contenant des nœuds flous. En notant toujours V l'ensemble des nœuds et F le sous ensemble flou de V, un graphe flou contenant des*

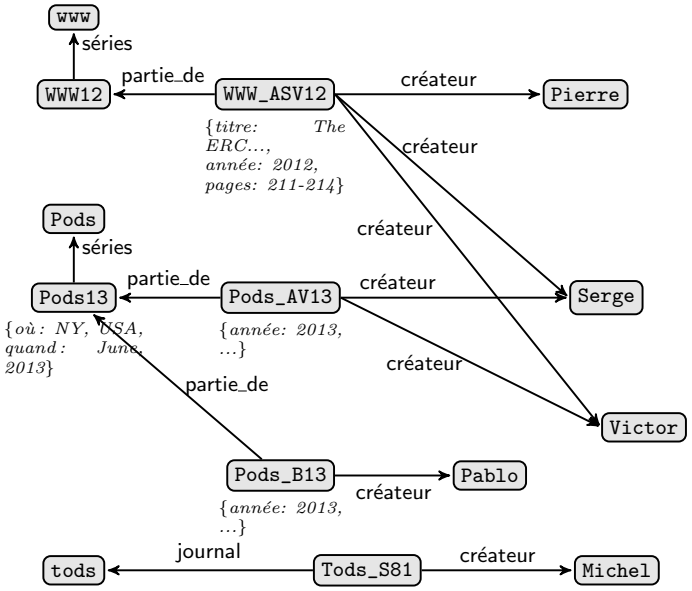

Figure 1: Base de données graphe inspirée d'une extraction de données de DBLP

nœuds flous est un triplet (V, F, ρ_F) *où* ρ_F *est une relation sur* V *définie par* $\rho_F(x, y) = \min(\rho(x, y), \mu_F(x), \mu_F(y))$ *où* μ_F *est la fonction d'appartenance associée à* F. *Dans la suite, on ne considère que le cas de nœuds non flous.*

Si ρ est symétrique alors on peut dire que (V, ρ) est un graphe non orienté. Si ρ n'est pas symétrique alors (V, ρ) est un graphe orienté. Sans perte de généralité, on ne considère que des graphes orientés dans la suite.

2. PRÉFÉRENCES FLOUES

Nous décrivons dans cette section les principaux éléments qui devraient pouvoir être exprimés dans une requête floue adressée à une base de données graphe. Deux types de préférences doivent pouvoir être considérés : des préférences concernant le contenu (attributs) du graphe et des préférences concernant la structure du graphe.

2.1 Concernant le contenu du graphe

L'idée est d'exprimer des conditions flexibles concernant les attributs associés aux nœuds (et éventuellement aux arcs) du graphe. Un exemple de requête flexible de ce type est "trouver les personnes *jeunes*, *hautement diplômées*, vivant en *Europe de l'Est*" ; en supposant qu'un nœud décrivant une personne contienne des informations concernant son âge, son niveau d'étude, son adresse, etc. Des conditions composées plus complexes peuvent être exprimées en utilisant des connecteurs flous (dont une large gamme est disponible). Nous ne détaillons pas cet aspect déjà intensivement étudié dans le cadre de l'interrogation floue des bases de données relationnelles [PB12].

2.2 Concernant la structure du graphe

Nous décrivons maintenant les concepts issus de la théorie des graphes flous qui nous semblent les plus utiles dans la perspective d'interrogation d'une base de données graphe. Soit un graphe flou $G = (V, \rho)$.
Un chemin p dans G est une suite $x_0 \to x_1 \to \ldots \to x_n$ $(n \geq 0)$ telle que $\rho(x_{i-1}, x_i) > 0$, $1 \leq i \leq n$. Le nombre de pas dans le chemin est n.

Force d'un chemin. — La *force* d'un chemin p de G est définie par :

$$ST(p) = \min_{i=1..n} \rho(x_{i-1}, x_i). \quad (1)$$

En d'autres termes, la force d'un chemin est le degré du plus faible des arcs entrant dans la composition du chemin. Deux nœuds pouvant être reliés par un chemin p tel que $ST(p) > 0$ sont dits *connectés*. Un chemin p est un cycle si $n \geq 1$ et $x_0 = x_n$. Il est possible de montrer que $\rho^k(x, y)$ est la force du plus fort chemin allant de x à y et contenant au plus k pas. Ainsi, la force du plus fort chemin reliant deux nœuds x et y, quel que soit le nombre de pas considéré, peut être écrite $\rho^\infty(x, y)$. Un algorithme permettant de calculer ρ^∞ en $O(|V|^4)$ est proposé dans [BS91].

Longueur et *distance*. — La *longueur* d'un chemin $p = x_0 \to x_1 \to \ldots \to x_n$ (au sens de ρ) est un concept défini par Rosenfeld [Ros75] par :

$$Length(p) = \sum_{i=1}^{n} \frac{1}{\rho(x_{i-1}, x_i)}. \quad (2)$$

En toute généralité, on a $Length(p) \geq n$. Dans le cas particulier où G est non flou (c-à-d dans le cas où ρ est booléenne), on a $Length(p) = n$. On peut définir la *distance* entre deux nœuds x et y dans G par :

$$\delta(x, y) = \min_{\text{tous les chemins allant de } x \text{ à } y} Length(p). \quad (3)$$

Il s'agit de la longueur du plus court chemin allant de x à y. On peut monter que δ est une métrique [Ros75] c-à-d que

$\delta(x, x) = 0$, $\delta(x, y) = \delta(y, x)$, et $\delta(x, z) \leq \delta(x, y) + \delta(y, z)$.

α-coupe d'une relation. — Elle est définie par : $\rho^\alpha = \{(x, y) \mid \rho(x, y) \geq \alpha\}$ où $\alpha \in [0^+, 1]$. On peut noter que ρ^α est une relation non floue.

2.3 Combinaison de préférences

La théorie des ensembles flous fournit une large gamme d'opérateurs permettant de connecter des conditions flexibles. Dans le cadre considéré, ces connecteurs permettent de combiner des prédicats sur le contenu et/ou la structure du graphe. Outre le minimum et le maximum qui généralisent la conjonction et la disjonction usuelles, on peut utiliser différents opérateurs de compromis tels que les moyennes ou leurs variantes pondérées, les opérateurs de conjonction et de disjonction pondérées proposées dans [DP86], l'opérateur de moyenne pondérée dynamique (OWA) introduit dans [Yag88], des quantificateurs flous [LK98], ou des opérateurs hiérarchiques permettant d'affecter des priorités aux différentes conditions que l'on combine [Yag08, BP12]. Nous invitons le lecteur à consulter [FY00] pour une présentation détaillée des divers connecteurs flous.

3. BASE DE DONNÉES GRAPHE FLOUE ET ALGÈBRE FLOUE

Dans cette section, nous définissons une algèbre permettant une interrogation flexible de bases de données graphes, dans lesquelles les graphes peuvent être flous ou non. Nous introduisons le modèle de données, puis les opérateurs de l'algèbre. Un exemple illustre les notions définies au fur et à mesure de leur introduction.

3.1 Modèle de données

Nous nous intéressons dans la suite à la manipulation de bases de données graphes floues dans lesquelles les nœuds et les arcs peuvent être porteurs de données (pe. des paires clef-valeur dans les graphes d'attributs évoqués dans la section 1.1). Nous commençons donc par proposer une extension de la notion de graphe flou à celle de *graphe de données flou*.

Définition 1 (Graphe de données flou). *Soit E un ensemble d'étiquettes (destinées à étiqueter les arcs du graphe). Un* graphe de données flou *$\mathcal{G}$ est un quadruplet (V, R, κ, ζ), où V est un ensemble fini de nœuds pour lequel chaque nœud n a un identifiant id, aussi noté n.id, $R = \bigcup_{e \in E}\{\rho_e : V \times V \to [0, 1]\}$ est un ensemble d'arcs flous reliant les nœuds de V, et κ (resp. ζ) est une fonction associant des données, par exemple un ensemble d'éléments de type clef-valeur, aux nœuds (resp. arcs) de $\mathcal{G}$.*

Remarque 3. (Arcs et graphe de données non flous) *Un graphe de données flou peut contenir à la fois des arcs flous et des arcs non flous puisqu'un arc flou ayant un degré de 0 ou 1 peut être considéré non flou. Par extension, un graphe de données non flou est simplement un cas particulier de graphe de données flou pour lequel $\rho_e : V \times V \to \{0, 1\}$ pour tout $e \in E$. Nous ne considérerons donc que des arcs et graphes de données flous, dont le cas non flou n'est qu'une spécialisation.*

Dans la suite, le terme *base de données graphe* est utilisé pour désigner un graphe de données flou. L'exemple suivant illustre cette notion.

Exemple 1 (Graphe de données flou). *La figure 3 est un exemple de graphe de données flou, contenant à la fois des arcs flous (dont le degré est par convention indiqué entre parenthèses à côté de l'étiquette) et des arcs non flous (équivalents à des arcs ayant un degré associé de valeur* `1`*). Dans cet exemple, le degré associé à un arc de la forme* `A -contributeur-> B` *correspond à la proportion de papiers de journaux écrits par* `B` *qui ont été co-écrits par* `A`*. Cette notion de degré est fondée sur une notion statistique simple, qui peut être rendue plus fine par l'application d'opérations floues ou par l'intégration de connaissances expertes.* ⋄

Cette fonctionnalité étant offerte par de nombreux systèmes, les nœuds sont supposés typés. Dans la figure 3, les nœuds `WWW12` et `Pods13` sont de type *Conférence*, les nœuds `Pods_AV13`, `Pods_B13`, `Tods_S81`, et `WWW_ASV12` sont de type *Article*, les nœuds `Pods`, `Tods`, et `WWW` sont de type *Série* et les autres nœuds sont de type *Auteur*. Si n est un nœud de V, alors $Type(n)$ représente son type.

3.2 Algèbre

Nous passons maintenant à la définition de l'algèbre floue permettant l'interrogation de graphes de données éventuellement flous. Notre algèbre est en partie inspirée de la notion de requête (non floue) à base de patron de graphe de [FLM$^+$12] et de l'algèbre non floue proposée dans [HS08]. L'unité d'information de base de l'algèbre est le graphe.

L'opérateur de sélection est fondé sur le concept de *patron flou de graphe*, une extension de la notion de patron de graphe non flou proposé dans [FLM$^+$12]. Nous commençons donc par introduire la notion de *patron flou de graphe*, s'appuyant elle-même sur la notion d'*expression régulière floue*.

Définition 2. *(*(Expression régulière floue) *Une* expression régulière floue *est une expression de la forme:*

$$F ::= e \mid F{\cdot}F \mid F \cup F \mid F^* \mid F^{Cond}$$

où

(i) $e \in E \cup \{\_\}$ représente un arc étiqueté par e, le caractère souligné (_) représentant n'importe quelle étiquette de E;
(ii) $F{\cdot}F$ est une concaténation d'expressions;
(iii) $F \cup F$ représente des expressions alternatives;
(iv) F^* représente la répétition d'expression;
(v) F^{Cond} représente un chemin p satisfaisant F et la condition Cond, où Cond est une combinaison booléenne de formules atomiques de la forme `Property is Fterm` *où* `Property` *est une propriété définie sur p et* `Fterm` *est un terme flou pré-défini ou défini par un utilisateur comme le terme flou* `short` *dont une représentation de la fonction d'appartenance est proposée en figure 4.*

Dans la suite, nous limitons les propriétés des chemins à l'ensemble $\{\texttt{ST}, \texttt{Length}\}$ représentant respectivement $ST(p)$ (voir équation 1) et $Length(p)$ (voir équation 2). Ainsi, des exemples de conditions sur la forme d'un chemin sont `Length is short` et `ST is strong`. Observons qu'une condition booléenne de la forme `Property op a` où `Property` est une propriété sur p, `a` est une constante et `op` est un opérateur de comparaison (<,=,...) est un cas particulier de condition floue.

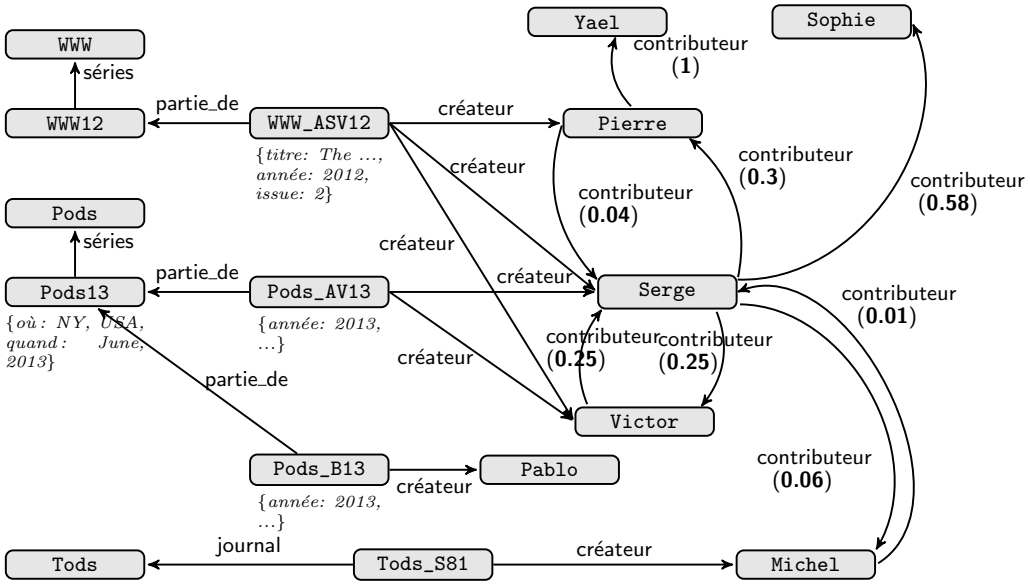

Figure 3: Graphe de données flou $\mathcal{BD}$ inspiré d'un extrait de DBLP data

Nous utilisons les notations suivantes : étant donnée une expression régulière floue f, f^+ est une notation raccourcie pour $f{\cdot}f^*$, f^k est une notation raccourcie pour $f{\cdot}f{\cdot}\cdots{\cdot}f$ ayant k occurrences de f et $f^{n,m}$ est une notation raccourcie pour $\bigcup_{i=n}^{m} f^i$.

Une expression régulière floue est dite *simple* si elle est de la forme e où $e \in E \cup \{\_\}$, c'est-à-dire qu'elle fait explicitement référence à un arc seul.

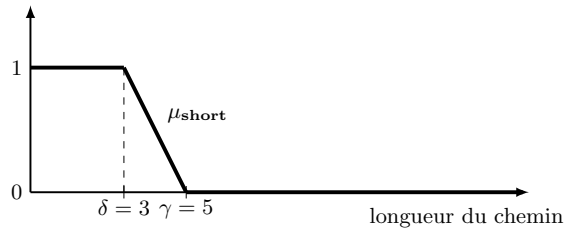

Figure 4: Représentation du terme flou *short*

Remarque 4. *Même si on ne désire considérer que des graphes classiques non flous, les concepts présentés ci-dessus peuvent toujours être utilisés en arguments des conditions floues (par exemple une longueur courte (short)) puisqu'ils restent valides sur des graphes non flous (avec $\rho(x) \in \{0, 1\}$).*

Définition 3. (Satisfaction d'une expression régulière floue) *Soient un chemin p et une expression régulière floue exp; p* satisfait *exp* avec un degré de satisfaction $\mu_{exp}(p)$ *defini comme suit, en fonction de la forme de exp (dans la suite, f, f_1 et f_2 sont des expressions régulières floues):*

exp **est de la forme** *e* **avec** *$e \in E$* **(resp. "_").** *Si p est de la forme $v_1 \xrightarrow{e'} v_1'$ où $e' = e$ (resp. où $e' \in E$) alors $\mu_{exp}(p) = 1$ sinon $\mu_{exp}(p) = 0$.*

exp **est de la forme** *$f_1{\cdot}f_2$***.** *Soit P l'ensemble des couples de chemins (p_1, p_2) tq p est de la forme p_1p_2. On a alors $\mu_{exp}(p) = \max_P(min(\mu_{f_1}(p_1), \mu_{f_2}(p_2)))$.*

exp **est de la forme** *$f_1 \cup f_2$***.** *Dans ce cas, on a $\mu_{exp}(p) = \max(\mu_{f_1}(p), \mu_{f_2}(p))$.*

exp **est de la forme** *f^****.** *Si p est un chemin vide alors $\mu_{exp}(p) = 1$. Sinon, on note P l'ensemble des listes de chemins $(p_1, \cdots, p_n)$ $(n > 0)$ telle que p est de la forme $p_1 \cdots p_n$. On a alors $\mu_{exp}(p) = \max_P(\min_{i\in[1..n]}(\mu_f(p_i)))$.*

exp **est de la forme** *f^{Cond} où $Cond$ est une condition floue telle que définie dans la définition 2. Dans ce cas, on a $\mu_{exp}(p) = \min(\mu_f(p), \mu_{Cond}(p))$ où $\mu_{Cond}(p)$ est le degré de satisfaction de $Cond$ par p.*

Dans la suite, on dira qu'une expression régulière est satisfaite si elle l'est à un degré strictement supérieur à 0.

Exemple 2. *Les chemins représentés dans la figure 5 sont extraits de la base de données graphe de la figure 3.*

- *L'expression $e_1 = créateur \cdot contributeur^+$ est une expression régulière floue. Tous les chemins p_i ($i \in [1..4]$) de la figure 5 satisfont e_1 avec un degré de égal à $\mu_{e_1}(p_i) = 1$.*
- *L'expression $e_2 = (créateur \cdot contributeur^+)^{\texttt{ST>0.4}}$ est une expression régulière floue. Le chemin p_4 est le seul chemin de la figure 5 satisfaisant e_2 (car $ST(p_1) = 0.3$, $ST(p_2) = 0.3$, $ST(p_3) = 0.01$ et $ST(p_4) = 0.58$), avec $\mu_{e_2}(p_4) = 1$.*
- *L'expression $e_3 = créateur{\cdot}(contributeur^+)^{\texttt{Length is short}}$ où short est le terme flou représenté en figure 4, est une expression régulière floue. Les chemins p_1, p_2 et p_4 de la figure 5 satisfont e_3 avec $\mu_{e_3}(p_1) = 0.83$ puisque $\mu_{short}(1/0.3) = 0.83$ (où $1/0.3$ est la longueur*

du chemin allant de `Serge` *à* `Pierre`*),* $\mu_{e_3}(p_2) = 0.67$ *puisque* $\mu_{short}(1/0.3 + 1) = 0.67$ *(où* $1/0.67$ *est le longueur du chemin allant de* `Serge` *à* `Yael`*) et* $\mu_{e_3}(p_4) = 1$ *puisque* $\mu_{short}(1/0.58) = 1$*. Le chemin* p_3 *ne satisfait pas* e_3 *puisque* $\mu_{short}(1/0.01) = 0$*.* $\diamond$

Nous introduisons maintenant la notion de *patron flou de graphe*, correspondant à un graphe (classique non flou) pour lequel les arcs sont étiquetés avec des expressions régulières floues, d'éventuelles conditions peuvent être posées sur les nœuds et les arcs, et un type peut être associé à un nœud.

Définition 4. (Patron flou de graphe) *Soit* $\mathcal{F}$ *un ensemble de termes flous. Un* patron flou de graphe *est défini par un sextuplet* $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, f_e^{path}, f_n^{cond}, f_e^{cond}, f_n^{type})$ *où*

(i) $V_\mathcal{P}$ *est un ensemble fini de nœuds ;*
(ii) $E_\mathcal{P} \subseteq V_\mathcal{P} \times V_\mathcal{P}$ *est un ensemble fini d'arcs* (u, u')*;*
(iii) f_e^{path} *est une fonction définie sur* $E_\mathcal{P}$ *telle que pour tout* (u, u') *dans* $E_\mathcal{P}$*,* $f_e^{path}(u, u')$ *est une expression régulière floue ;*
(iv) f_n^{cond} *est une fonction définie sur* $V_\mathcal{P}$ *telle que pour tout nœud* u*,* $f_n^{cond}(u)$ *est une condition sur les attributs de* u*, définie par une combinaison de formules atomiques de la forme* `A is Fterm` *où* `A` *est un attribut et* `Fterm` *est un terme flou (pe.* `année is recent`*). De nouveau, un prédicat booléen de la forme* `A op a` *(où* `A` *est un attribut,* `a` *est une constante et* `op` *est un opérateur de comparaison, comme pe.* `année`$>$`2012` *est un cas particulier de prédicat flou.*
(v) f_e^{cond} *est la contrepartie de* f_n^{cond} *pour les arcs. Pour tout* (u, u') *dans* $E_\mathcal{P}$ *pour lequel* $f_e^{path}(u, u')$ *est simple,* f_e^{cond} *est une condition sur les attributs de* (u, u') *; et*
(vi) f_n^{type} *est une fonction définie sur* $V_\mathcal{P}$ *tq pour tout nœud* u*,* $f_n^{type}(u)$ *est le type de* u*.*

Dans la suite, nous choisissons d'adopter une syntaxe *à la Cypher* pour la représentation des patrons. Deux raisons motivent ce choix : 1) cette syntaxe inspirée de l'art ASCII pour la représentation de graphes est intuitive, et 2) adopter cette syntaxe constitue un premier pas vers la définition d'un langage orienté utilisateur fondé sur l'algèbre, qui constitue une perspective naturelle à court terme. Un patron flou de graphe exprimé *à la Cypher* consiste en un ensemble d'expressions de la forme

```
(n1:Type1)-[exp]->(n2:Type2)
ou  (n1:Type1)-[e:label]->(n2:Type2)
```

où `n1`, `n2` sont des variables de nœuds, `e` est une variable d'arc, `label` est une étiquette de E, `exp` est une expression régulière floue, et `Type1` et `Type2` sont des types de nœuds. Une telle expression représente un chemin satisfaisant une expression régulière floue (qui s'avère être *simple* dans la seconde forme) allant d'un nœud de type `Type1` à un nœud de type `Type2`. Tous les arguments de l'expression sont individuellement optionnels, ce qui signifie que la forme la plus dépouillée d'une telle expression est `()-[]->()` représentant un chemin constitué d'un arc quelconque reliant deux nœuds quelconques.

Les conditions sur les attributs sont exprimées sur les variables de nœuds et d'arcs dans une clause `where`.

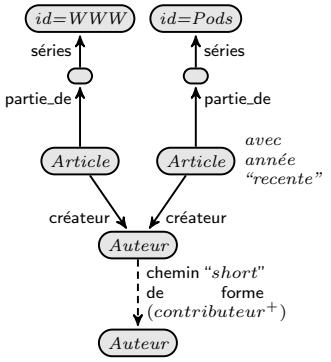

Figure 6: Patron $\mathcal{P}$

Exemple 3. *On note* $\mathcal{P}$ *le patron flou de graphe suivant :*

```
1 (art1:Article)-[partie_de.séries]->(s1),
2 (art2:Article)-[partie_de.séries]->(s2),
3 (art1)-[:créateur]->(auth1:Auteur),
4 (art2)-[:créateur]->(auth1:Auteur),
5 (auth1)-[(contributeur+)|Length is short]->(auth2:Auteur),
6 where
7 s1.id=WWW, s2.id=Pods,
8 art2.année is recent.
```

Le graphe de la figure 6 est une représentation graphique du patron $\mathcal{P}$ *dans laquelle l'arc en pointillés représente un chemin, les informations en italique représentent des conditions sur les attributs des nœuds ou arcs et le type associé à un nœud est indiqué en italique dans celui-ci. Ce patron capture des informations concernant des auteurs (*`auth2`*) qui ont parmi leurs contributeurs proches un auteur (*`auth1`*) ayant publié un papier (*`art1`*) à* `WWW` *et un autre papier (*`art2`*) à* `Pods` *récemment (*`art2.année is recent`*).* $\diamond$

Nous passons maintenant à la notion de la satisfaction d'un patron flou de graphe.

Définition 5. (Satisfaction d'un patron flou de graphe) *Un graphe de données flou* $G = (V, R, \kappa, \zeta)$ satisfait *un patron flou de graphe* $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P}, f_e^{path}, f_n^{cond}, f_e^{cond}, f_n^{type})$ *avec un degré de satisfaction noté* $\mu_\mathcal{P}(G)$ *s'il existe une relation binaire* $S \subseteq V_\mathcal{P} \times V$ *représentant une fonction injective de* $V_\mathcal{P}$ *dans* V *telle que :*

1. *(correspondance des nœuds) pour tout nœud* $u \in V_\mathcal{P}$*, il existe un nœud* $v \in V$ *tel que* $(u, v) \in S$ *;*
2. *(correspondance des arcs) pour tout arc* $(u, u') \in E_\mathcal{P}$*, il existe deux nœuds* v *et* v' *de* V *tels que* $\{(u, v), (u', v')\} \subseteq S$ *et il existe un chemin* p *dans* G *de* v *à* v' *tel que* p *satisfait* $f_e^{path}(u, u')$ *(on rappelle que, d'après la définition 3, un degré de satisfaction de valeur strictement supérieure à zéro est associé en cas de satisfaction) ;*
3. *(vérification des conditions sur les attributs et des types des nœuds) pour tout tuple* $(u, v) \in S$*,* $\kappa(v) \vdash f_n^{cond}(u)$ *(la sémantique de* $\vdash$ *découle trivialement du contexte ici) et* $f_n^{type}(u) = Type(v)$*.*
4. *(vérification des conditions sur les attributs des arcs) le même raisonnement peut être appliqué en ce qui concerne les conditions sur les attributs des arcs étiquetés par une expression régulière floue simple dans* $E_\mathcal{P}$*, c-à-d* $\zeta(v, v') \vdash f_e^{cond}(u, u')$*.*

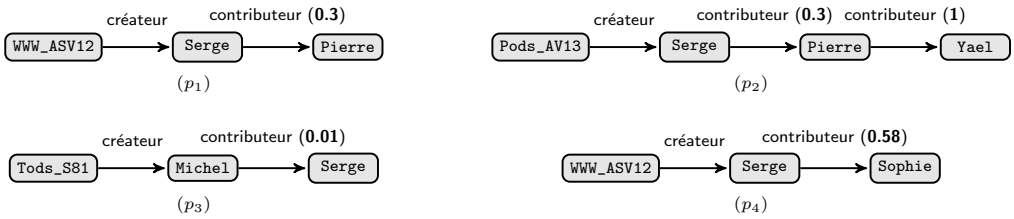

Figure 5: Satisfaction d'une expression régulière floue

$\mu_{\mathcal{P}}(G)$ est le degré de satisfaction minimal induit des correspondances et vérifications de 2., 3. et 4. Si aucune relation S satisfaisant ces conditions n'existe alors $\mu_{\mathcal{P}}(G) = 0$, autrement dit, G ne satisfait pas $\mathcal{P}$.

Exemple 4. *La figure 7 donne l'ensemble des sous-graphes de $\mathcal{BD}$ satisfaisant le patron $\mathcal{P}$ de l'exemple 3 avec le degré de satisfaction associé (par manque de place, les données véhiculées par les nœuds ne sont pas reportées).* `auth1` *est nécessairement* `Serge` *ou* `Victor` *qui sont les seuls auteurs de $\mathcal{BD}$ ayant écrit à la fois un papier à* `WWW` *et un papier récent à* `Pods`*. De façon à pouvoir traiter la ligne 8, on suppose que $\mu_{recent}(2013) = 0.7$. On note ici que le degré de satisfaction d'un graphe satisfaisant $\mathcal{P}$ est le minimum des degrés de satisfaction induits par les lignes 5 et 8.* $\diamond$

Nous donnons maintenant la définition des opérateurs flous de l'algèbre.

Il convient de noter que même si une base de données graphe contient un seul graphe, une requête de l'algèbre peut retourner un ensemble de graphes puisque plusieurs sous-graphes peuvent satisfaire un patron comme le montre l'exemple 4. Un degré de satisfaction est associé à chaque graphe. Un ensemble de couples $\langle$ *graphe*, *degré* $\rangle$ est tout simplement un ensemble flou de graphes. Chaque opérateur de l'algèbre prend ainsi en entrée un ou plusieurs (selon l'arité de l'opérateur) ensemble(s) flou(s) de graphes et génère un ensemble flou de graphes. L'algèbre est close. En cas de production de graphes doublons (un même graphe apparaissant avec différents degrés de satisfaction), seul le plus haut degré de satisfaction est conservé. Appliquer un opérateur à la base de données initiale $\mathcal{BD}$ revient à appliquer l'opérateur au singleton $\{\langle\, \mathcal{BD}\,,\, 1\,\rangle\}$. Il est à noter que les graphes d'un ensemble ne partagent pas nécessairement la même structure. Les *expressions de l'algèbre* sont récursivement définies par: (i) une base de données graphe $\mathcal{BD}$ est une expression de l'algèbre, et (ii) si $e_1, \cdots, e_n$ sont des expressions et O est un opérateur d'arité n alors $O(e_1, \cdots, e_n)$ est une opération de l'algèbre.

Définition 6. (Opérateur de sélection) *L'opérateur de sélection σ prend en entrée un patron flou de graphe $\mathcal{P}$ et un ensemble flou $\mathcal{G}$ de graphes. Il renvoit un ensemble flou constitué des sous-graphes de $\mathcal{G}$ satisfaisant $\mathcal{P}$:*

$$\sigma_{\mathcal{P}}(\mathcal{G}) = \{\langle\, s\,,\, \min(d,\, \mu_{\mathcal{P}}(s))\,\rangle \,|\, \mu_{\mathcal{P}}(s) > 0\}$$

où s est un sous-graphe de g tel que $\langle\, g\,,\, d\,\rangle \in \mathcal{G}$.

Exemple 5. *La figure 7 présente l'ensemble des réponses de $\sigma_{\mathcal{P}}(\mathcal{BD})$ où $\mathcal{P}$ est le patron de l'exemple 3 et $\mathcal{BD}$ est la base de données graphe de la figure 3.*

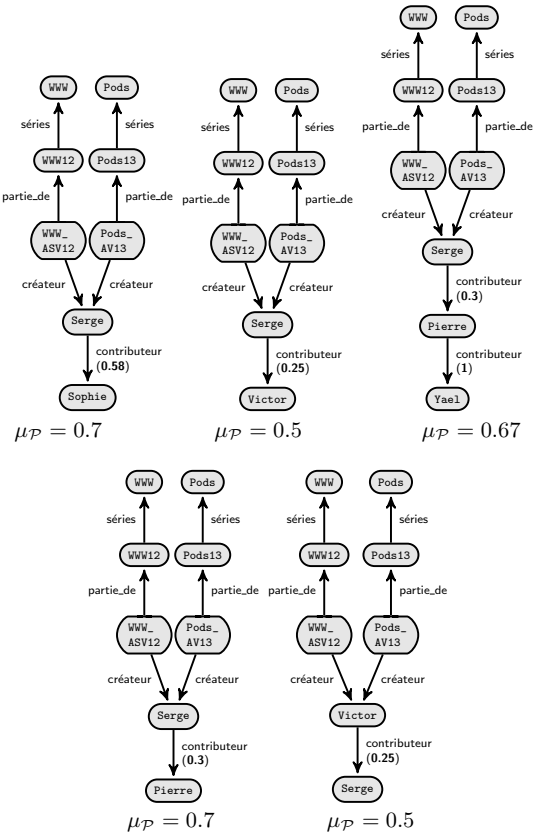

Figure 7: Sous-graphes de $\mathcal{BD}$ satisfaisant $\mathcal{P}$

Définition 7. (Opérateurs de projection) *L'opérateur de projection "sur les arcs" noté Π^{arcs} permet de supprimer des relations d'un graphe (sans supprimer de nœud du graphe). Cet opérateur prend en entrée un ensemble flou $\mathcal{G}$ de graphes, un ensemble d'étiquettes $\mathcal{L} \subseteq E$, et renvoie un ensemble flou de graphes défini comme suit:*

$$\Pi^{arcs}_{\mathcal{L}}(\mathcal{G}) = \{\langle\, (V,\, R',\, \kappa,\, \zeta)\,,\, d\,\rangle \,|\, \langle\, (V,\, R,\, \kappa,\, \zeta)\,,\, d\,\rangle \in \mathcal{G}\}$$

où $R' = \{\rho_e \,|\, \rho_e \in R \text{ et } e \in \mathcal{L}\}$.

L'opérateur de projection "sur les nœuds" noté $\Pi^{nœuds}$ permet de supprimer des nœuds d'un graphe. Cet opérateur prend en entrée un ensemble flou $\mathcal{G}$ de graphes, et un ensemble de types $\mathcal{T}$, et renvoie un ensemble flou de graphes

défini comme suit:

$$\Pi_{\mathcal{T}}^{nœuds}(\mathcal{G}) = \{\langle (V',\ R,\ \kappa,\ \zeta)\ ,\ d\rangle \mid \langle (V,\ R,\ \kappa,\ \zeta)\ ,\ d\rangle \in \mathcal{G}\}$$

où $V' = \{v \mid v \in V$ *et* $Type(v) \in \mathcal{T}\}$ *et* R' *est la restriction de* R *sur* $V' \times V'$.

Exemple 6. *La figure 8 contient les réponses de* $\Pi_{Auteur}^{nœuds}(\sigma_{\mathcal{P}}(\mathcal{BD}))$ *où* $\sigma_{\mathcal{P}}(\mathcal{BD})$ *est l'expression de l'exemple 5 (c-à-d que* $\mathcal{P}$ *est le patron de l'exemple 3 et* $\mathcal{BD}$ *est la base de données graphe de la figure 3).*

Définition 8. (Opérateur d'alpha-coupe) *Cette opération effectue une* α*-coupe sur une relation floue du graphe (voir section 2.2). L'opérateur d'alpha-coupe Cut prend en entrée un ensemble flou de graphes* $\mathcal{G}$*, une étiquette* $e \in E$ *et un nombre réel* $\alpha \in [0^+, 1]$*. Il produit un ensemble flou de graphes constitué des graphes de* $\mathcal{G}$ *où une alpha-coupe a été effectuée sur la relation* ρ_e *:*

$$Cut_{e,\alpha}(\mathcal{G}) = \{\langle (V,\ R',\ \kappa,\ \zeta)\ ,\ d\rangle \mid \langle (V,\ R,\ \kappa,\ \zeta)\ ,\ d\rangle \in \mathcal{G}\}$$

où $R' = \{\rho_l \mid \rho_l \in R$ *et* $l \neq e\} \cup \{\rho_e^{\alpha}\}$.

L'opérateur d'alpha-coupe met à jour la relation ρ_e en lui appliquant une α-coupe, et donc rend ρ_e non floue. Cette opération n'a pas d'impact sur les données véhiculées par les arcs étiquetés par e.

Exemple 7. *La figure 9 donne les réponses renvoyées par l'expression* $Cut_{contributeur,0.3}(\Pi_{Auteur}^{nœuds}(\sigma_{\mathcal{P}}(\mathcal{BD})))$ *où* $\Pi_{Auteur}^{nœuds}(\sigma_{\mathcal{P}}(\mathcal{BD}))$ *est l'expression de l'exemple 6 (c-à-d que* $\mathcal{P}$ *est le patron de l'exemple 3 et* $\mathcal{BD}$ *est la base de données graphe de la figure 3).*

Si le graphe comportait des nœuds flous, l'opérateur Cut pourrait être étendu pour permettre une α-coupe sur les nœuds.

Définition 9 (Opérateurs ensemblistes). *Les opérateurs classiques d'union, intersection et différence sont définis à partir des opérations classiques des ensembles flous (voir section 1.2.1 pour les références).*

Les auteurs de [HS08] définissent des opérateurs permettant de combiner, fusionner et restructurer des graphes, qui pourraient compléter l'algèbre.

4. TRAVAUX CONNEXES

Comme de nombreux modèles ont été proposés pour représenter des données ayant implicitement ou explicitement une structure de graphe [AG08], il existe de nombreux langages pour l'interrogation de graphes. Les auteurs de [AG08], [Woo12] et [BB13] proposent des états de l'art complémentaires des langages de requête pour les graphes proposés au cours de ces vingt-cinq dernières années incluant des langages pour l'interrogation de bases de données objet, de données semi-structurées, de réseaux sociaux, ou de données du web sémantique. Les travaux de [BB13] abordent les langages pour les bases de données graphes sous un angle théorique, en soulignant le fait que les systèmes de gestion de bases de données graphes actuels reposent sur des langages pour lesquels une syntaxe et une sémantique précise doivent encore être définies.

Certains langages de requête pour des bases de données graphes sont fondés sur une algèbre. Dans le cadre des graphes de données RDF, les auteurs de [AP11] proposent une formalisation algébrique dédiée à SPARQL, avec extension navigationnelle du langage. Dans [SEH14], les auteurs proposent un langage *à la SPARQL* permettant d'interroger des graphes d'attributs, en y associant un mécanisme d'évaluation fondé sur une algèbre. Dans [AG05], les auteurs proposent de nouvelles primitives inspirées du monde des bases de données graphes pour l'interrogation de données RDF. Une algèbre étendant l'algèbre relationnelle pour l'interrogation de bases de données graphes est proposée dans [HS08]. Comme mentionné précédemment, notre travail est partiellement inspiré de cette contribution. Les langages de requête proposés dans tous ces travaux concernent des bases de données graphes ou RDF non floues, sans interrogation flexible des données.

Concernant l'interrogation flexible de la topologie des bases de données graphes, il existe trois grandes catégories d'approches : (i) les approches du type *interrogation par mot-clef* ignorant le schéma des données (voir par exemple [HWYY07]) souffrant d'un manque d'expressivité dans une grande partie des cas d'utilisation d'une base de données graphe [MMVP09] ; (ii) les approches consistant à proposer à l'utilisateur des *réponses approximatives* à une requête (non floue), par exemple par la mise en œuvre d'un mécanisme d'extension ou de relâchement de la requête ou par un mécanisme calculant des réponses correspondant approximativement à la requête (voir par exemple [KS01], [BDBH08], ou [MMVP09]) ; (iii) les approches consistant à permettre à un utilisateur d'*introduire une forme de flexibilité dans les requêtes*, famille d'approches dans laquelle ce présent travail se situe.

Parmi le dernier type d'approche, de nombreuses contributions concernent l'extension flexible de XPath [DMP07, CDG$^+$09, AJLM11] pour l'interrogation de données semi-structurées (arbre). Même si les langages navigationnels *à la XPath* sont jugés adéquats pour l'interrogation de données graphes (voir [LMV13]), aucune extension flexible de ce type de langage n'a encore été proposée pour le modèle des bases de données graphes.

Dans [CMY10], les auteurs proposent une extension de la syntaxe du langage SPARQL permettant d'introduire des termes et des relations floues dans une requête SPARQL, en focalisant les travaux sur l'aspect d'enrichissement syntaxique du langage SPARQL et la faisabilité d'une implantation. Enfin, les auteurs de [CnC11] proposent une extension de SPARQL permettant d'interroger des graphes contenant des arcs pondérés, dans l'objectif final de pouvoir ordonner les réponses à une requête. La contribution est axée sur la mise en œuvre de l'extension via un moteur SPARQL. De notre côté, nous nous plaçons dans un cadre plus général permettant l'interrogation de bases de données graphes éventuellement floues en proposant une algèbre (qui pourrait servir de fondement à d'autres langages) fondée sur la théorie des ensembles flous et la théorie des graphes flous.

5. CONCLUSION ET PERSPECTIVES

Nous présentons dans ce papier une algèbre permettant l'interrogation flexible de bases de données graphes, dans lesquelles un graphe peut être flou ou non. Cette algèbre,

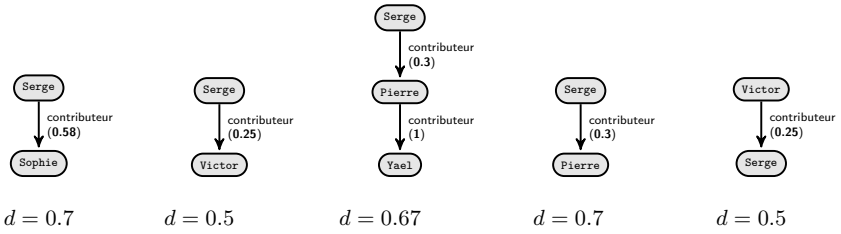

Figure 8: Réponse de $\Pi^{nœuds}_{Auteur}(\sigma_{\mathcal{P}}(\mathcal{BD}))$

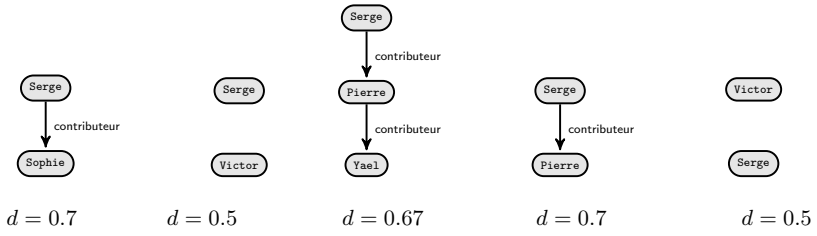

Figure 9: Réponse de $Cut_{contributor,0.3}(\Pi^{nœuds}_{Auteur}(\sigma_{\mathcal{P}}(\mathcal{BD})))$

fondée sur la théorie des ensembles flous et la théorie des graphes flous, propose un modèle de données et une famille d'opérateurs permettant d'exprimer des préférences flexibles sur 1) les données contenues dans le graphe, et 2) la structure du graphe.

De nombreuses perspectives théoriques et appliquées sont ouvertes par ce travail. De même que l'algèbre relationnelle constitue la base du langage SQL utilisé dans les systèmes commerciaux, l'algèbre floue proposée ici est destinée à servir de fondement à l'extension d'outils plus orientés utilisateur tels que le langage Cypher [Neo13] implanté dans le système Neo4j [Neo] (permettant actuellement la gestion de graphes d'attributs non flous). Comme dans le cadre relationnel, des fonctionnalités d'ordonnancement et d'agrégation seraient offertes par le langage, ouvrant la porte à la prise en compte de nouvelles notions des graphes flous telles que la *distance* ou le *diamètre* des chemins entre les nœuds, et également les *degrés entrant et sortant* ou la *centralité* des nœuds [Yag13], qui sont des notions régulièrement utilisées pe. dans le cadre de l'analyse structurelle de réseaux sociaux. Ces fonctionnalités permettraient également de pourvoir supprimer les nœuds faiblement connectés des réponses.

La mise en œuvre soulève (au moins) deux problèmes. Le premier problème, a priori facile à résoudre, concerne l'éventuelle modélisation d'une base de données floue dans un système gérant des bases de données graphes non floues. Une relation $\rho_e(x, y)$ d'une base de données floue peut être représentée dans une base de donnée non floue en considérant que l'étiquette reliant deux nœuds x et y n'est pas juste e mais un couple e, v où v est la valeur de $\rho_e(x, y)$. Ainsi, un graphe non flou être utilisé pour représenter un graphe flou. Dans le cas des graphes d'attributs, ce mécanisme peut être très simplement implanté en ajoutant à chaque arc étiqueté par e un attribut *fdegree* portant la valeur de v (en supposant ensuite que *fdegree* devient un mot clef du langage).

Un second problème concerne l'optimisation de l'évaluation des requêtes. À l'image du cadre relationnel, des règles d'optimisation permettant d'optimiser les requêtes en exploitant au mieux les index existants et la sélectivité des opérateurs seront à définir.

6. REFERENCES

[AG05] Renzo Angles and Claudio Gutiérrez. Querying RDF Data from a Graph Database Perspective. In *Proc. of European Semantic Web Conf. (ESWC)*, pages 346–360, 2005.

[AG08] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1):1:1–1:39, 2008.

[AJLM11] Jesús M. Almendros-Jiménez, Alejandro Luna, and Ginés Moreno. A Flexible XPath-based Query Language Implemented with Fuzzy Logic Programming. In *Proc. of the Intl. Conf. on Rule-based Reasoning, Programming, and Applications (RuleML)*, pages 186–193. Springer-Verlag, 2011.

[All] AllegroGraph web site. http://franz.com/agraph/allegrograph.

[Ang12] Renzo Angles. A comparison of current graph database models. In *Proc. of IEEE Intl. Conf. on Data Engineering Workshops*, pages 171–177, 2012.

[AP11] Marcelo Arenas and Jorge Pérez. Querying semantic web data with SPARQL. In *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, pages 305–316, 2011.

[BB13] Pablo Barceló Baeza. Querying graph databases. In *Proc. of the ACM Symposium on Principles of Database Systems (PODS)*, pages 175–188, 2013.

[BDBH08] Patrice Buche, Juliette Dibie-Barthélemy, and Gaëlle Hignette. Flexible Querying of Fuzzy RDF Annotations Using Fuzzy Conceptual Graphs. In *Proc. of Intl. Conf. on Conceptual Structures (ICCS)*, pages 133–146, 2008.

[BP95] Patrick Bosc and Olivier Pivert. SQLf: a relational database language for fuzzy querying.

IEEE Trans. on Fuzzy Systems, 3:1–17, 1995.

[BP12] Patrick Bosc and Olivier Pivert. On four noncommutative fuzzy connectives and their axiomatization. *Fuzzy Sets and Systems*, 202:42–60, 2012.

[BS91] Prabir Bhattacharya and Francis Suraweera. An algorithm to compute the supremum of max-min powers and a property of fuzzy graphs. *Pattern Recognition Letters*, 12(7):413–420, 1991.

[BT12] Shalini Batra and Charu Tyagi. Comparative analysis of relational and graph databases. *Intl. Journal of Soft Computing and Engineering*, 2(2), 2012.

[CAH12] Marek Ciglan, Alex Averbuch, and Ladislav Hluchý. Benchmarking traversal operations over graph databases. In *IEEE ICDE Workshops*, pages 186–189, 2012.

[CDG+09] Alessandro Campi, Ernesto Damiani, Sam Guinea, Stefania Marrara, Gabriella Pasi, and Paola Spoletini. A Fuzzy Extension of the XPath Query Language. *J. Intell. Inf. Syst.*, 33(3):285–305, 2009.

[CMY10] Jingwei Cheng, Z. M. Ma, and Li Yan. f-SPARQL: A flexible extension of SPARQL. In *Proc. of the Intl. Conf. on Database and Expert Systems Applications*, pages 487–494, 2010.

[CnC11] Juan P. Cedeño and K. Selçuk Candan. R2DF Framework for Ranked Path Queries over Weighted RDF Graphs. In *Proc. of the Intl. Conf. on Web Intelligence, Mining and Semantics*, pages 40:1–40:12, 2011.

[DMP07] Ernesto Damiani, Stefania Marrara, and Gabriella Pasi. FuzzyXPath: Using Fuzzy Logic an IR Features to Approximately Query XML Documents. In *Foundations of Fuzzy Logic and Soft Computing*, LNCS, pages 199–208. Springer, 2007.

[DP86] Didier Dubois and Henri Prade. Weighted minimum and maximum operations in fuzzy set theory. *Information Sciences*, 39:205–210, 1986.

[DP96] Didier Dubois and Henri Prade. Using fuzzy sets in database systems: Why and how? In *Proc. of FQAS'96*, pages 89–103, 1996.

[DP00] Didier Dubois and Henri Prade. *Fundamentals of fuzzy sets*, volume 7 of *The Handbooks of Fuzzy Sets*. Kluwer Academic Pub, 2000.

[DSUBGV+10] David Dominguez-Sal, P. Urbón-Bayes, Aleix Giménez-Vañó, Sergio Gómez-Villamor, Norbert Martínez-Bazan, and Josep-Lluis Larriba-Pey. Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark. In *Proc. of WAIM'10 Workshops*, pages 37–48, 2010.

[FLM+12] Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Yinghui Wu. Adding regular expressions to graph reachability and pattern queries. *Frontiers of Computer Science*, 6(3):313–338, 2012.

[FY00] János Fodor and Ronald R. Yager. Fuzzy-set theoretic operators and quantifiers. In *The Handbooks of Fuzzy Sets Series, vol. 1: Fundamentals of Fuzzy Sets*, pages 125–193. Kluwer Academic Publishers, 2000.

[GS02] Rosalba Giugno and Dennis Shasha. Graphgrep: A fast and universal method for querying graphs. In *ICPR (2)*, pages 112–115, 2002.

[HS08] Huahai He and Ambuj K. Singh. Graphs-at-a-time: query language and access methods for graph databases. In *Proc. of SIGMOD'08*, pages 405–418, 2008.

[HWYY07] Hao He, Haixun Wang, Jun Yang, and Philip S. Yu. Blinks: Ranked keyword searches on graphs. In *Proc. of the ACM SIGMOD*, pages 305–316, 2007.

[Inf] InfiniteGraph web site. http://www.objectivity.com/infinitegraph.

[KS01] Yaron Kanza and Yehoshua Sagiv. Flexible queries over semistructured data. In *Proc. of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '01, pages 40–51, 2001.

[LK98] Yaxin Liu and Etienne E. Kerre. An overview of fuzzy quantifiers. (i). interpretations. *Fuzzy Sets and Systems*, 95(1):1–21, 1998.

[LMV13] Leonid Libkin, Wim Martens, and Domagoj Vrgoč. Querying Graph Databases with XPath. In *Proc. of ICDT'13*, pages 129–140, 2013.

[MMVP09] Federica Mandreoli, Riccardo Martoglia, Giorgio Villani, and Wilma Penzo. Flexible query answering on graph-modeled data. In *Proc. of the Intl. Conf. on Extending Database Technology: Advances in Database Technology (EDBT)*, pages 216–227, 2009.

[MN00] John N. Mordeson and Premchand S. Nair. *Fuzzy Graphs and Fuzzy Hypergraphs*, volume 46 of *Studies in Fuzziness and Soft Computing*. Springer, 2000.

[Neo] Neo4j web site. http://www.neo4j.org.

[Neo13] Neo4j Team of Neo Technology. The Neo4j Manual v2.0.0, 2013. Part III.

[PB12] Olivier Pivert and Patrick Bosc. *Fuzzy Preference Queries to Relational Databases*. Imperial College Press, London, UK, 2012.

[Ros75] Azriel Rosenfeld. Fuzzy graphs. In *Fuzzy Sets and their Applications to Cognitive and Decision Processes*, pages 77–97. Academic Press, 1975.

[SEH14] Sherif Sakr, Sameh Elnikety, and Yuxiong He. Hybrid query execution engine for large attributed graphs. *Inf. Syst.*, 41:45–73, 2014.

[Spa] Sparksee (formerly known as DEX) web site. http://sparsity-technologies.com.

[VMZ+10] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *ACM Southeast Regional Conf.*, page 42, 2010.

[Woo12] Peter T. Wood. Query languages for graph databases. *SIGMOD Record*, 41(1):50–60, 2012.

[Yag88] Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):183–190, 1988.

[Yag08] Ronald R. Yager. Prioritized aggregation operators. *Int. J. Approx. Reasoning*, 48(1):263–274, 2008.

[Yag13] Ronald R. Yager. Social network database querying based on computing with words. In *Flexible Approaches in Data, Information and Knowledge Management*, Studies in Computational Intelligence. Springer, 2013.

[Zad65] Lotfi A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[ZDDK08] Slawomir Zadrozny, Guy De Tré, Rita M. De Caluwe, and Janusz Kacprzyk. An overview of fuzzy approaches to flexible database querying. In José Galindo, editor, *Handbook of Research on Fuzzy Information Processing in Databases*, pages 34–54. IGI Global, 2008.