Actes du 2nd Atelier

GAOC

Graphes et Appariement d'Objets Complexes

En conjonction avec EGC 2011

25 Janvier 2011
Brest

Graphes et Appariement d'Objets Complexes

Contexte et Objectifs

Les applications telles que les bibliothèques numériques, les systèmes de médiation ou les architectures à base de services Web produisent des masses de données générant un grand nombre de modèles. Ces modèles sont destinés à être stockés dans l'objectif d'être accessibles pour des utilisateurs (clients) exprimant un besoin spécifique, traduit totalement ou en partie à l'aide d'une requête. Ils décrivent des objets complexes tels que les documents, les processus métiers ou les ontologies relatives à des domaines de connaissances, et sont de ce fait hétérogènes et riches du point de vue sémantique. Actuellement, la plupart des applications utilisant de tels modèles se basent sur une représentation sous forme de diagrammes ou graphes dont l'accès se décline par un processus d'appariement qui s'appuie uniquement sur des propriétés de structures (topologiques). La représentation sous forme de graphes devient complexe dès qu'elle tente de capturer toute la sémantique du modèle qu'elle représente. Par conséquent, les problèmes d'appariement de ces graphes deviennent difficiles à résoudre.

Par ailleurs, l'appariement de graphes est connu dans la littérature des graphes sous le nom de « graph matching problem ». Ce problème est difficile. Plusieurs algorithmes et heuristiques ont été développés pour des matching exacts (isomorphisme) ou inexacts (placement, plongement,.) sur des familles de graphes ou des graphes quelconques. Le but de cet atelier est de réunir différentes communautés de chercheurs académiques ou industriels travaillant sur des problèmes de matching et d'appariement de modèles de graphes. Des discussions auront lieu sur les aspects théoriques et appliqués de ces problèmes, et prolongeront les discussions initiées lors du premier atelier GAOC ayant eu lieu en 2010.

Thèmes

Les thématiques de l'atelier sont les suivantes :
- appariement de graphes et services Web
- appariement de graphes et documents XML, RDF,.
- appariement de graphes et Web sémantique
- appariement de graphes et ontologies
- appariement approximatif/flou de graphes
- fouille de graphes
- isomorphisme/homomorphisme de graphes
- plongement /placement de graphes
- résolution de requêtes à graphes
- etc.

Comités

Responsables de l'atelier

Allel Hadjali, ENSSAT/IRISA, Université de Rennes 1, France
Karen Pinel-Sauvagnat, IRIT, Université de Toulouse 3, France

Comité de programme

Boutheina Ben Yaghlane Ben Slimen, LARODEC, Tunis
Mokrane Bouzeghoub, PRISM - Université de Versailles
Sylvie Calabretto, LIRIS - INSA de Lyon
Madalina Croitoru, LIRMM Université de Montpellier
Khalil Drira, LAAS, Toulouse
Eric Duchêne, LIESP - Université Claude Bernard, Lyon
Brice Effantin, LIESP - Université Claude Bernard, Lyon
Daniela Grigori, PRISM - Université de Versailles
Mohand-Said Hacid, LIRIS - Université Claude Bernard, Lyon
Mohammed Haddad, LIESP - Université Claude Bernard, Lyon
Allel Hadjali, IRISA - Université de Rennes 1
Hamamache Kheddouci, LIESP - Université Claude Bernard, Lyon
Ludovic Liétard, IRISA - Université de Rennes 1
Guy Melançon, LABRI - Université de Bordeaux 1
Laurent Miclet, IRISA - Université de Rennes 1
Gabriella Pasi, Université de Milan, Italie
Jean-Marc Petit, LIRIS - INSA de Lyon
Daniel Rocacher, IRISA - Université de Rennes 1
Karen Pinel-Sauvagnat, IRIT - Université Paul Sabatier, Toulouse
Hamida Seba, LIESP - Université Claude Bernard, Lyon

Table des matières

Proportions analogiques de séquences et d'arbres. Définitions, algorithmes et applications.

Anouar Ben Hassena
Laurent Miclet

IRISA, Rennes
laurent.miclet@enssat.fr,
anouar.ben_hassena@inria.fr

Résumé. En intelligence artificielle, l'analogie est une technique de raisonnement empirique utilisée pour la résolution de problèmes, le traitement du langage, l'apprentissage, etc. Cette présentation traite de la proportion analogique (une analogie forte) entre quatre séquences, et entre quatre arbres. On présentera les définitions et les algorithmes qui permettent de mesurer le degré d'analogie et de résoudre des équations analogiques dans les domaines des séquences et des arbres. On donnera aussi une application à l'analyse syntaxiques de phrases extraites du corpus Penn Wall Street Journal Treebank.

Summary. In Artificial Intelligence, analogy is used as a non exact reasoning technique to solve problems, for natural language processing, for learning classification rules, etc. We focus in this paper on the matching by analogical proportion of sequences and trees. We firstly give definitions and algorithms which deal with the analogical proportion between four sequences, and between four trees. Secondly, we present an application to the learning of the syntactic tree (parsing) of a sentence. As an experiment, we measure the performance of our parser on a corpus extracted from the Penn Wall Street Journal Treebank.

Programmation linéaire en nombres entiers pour la recherche d'isomorphisme de sous-graphe : Application à la recherche de symboles graphiques

Pierre Le Bodic*, Pierre Héroux **
Sébastien Adam **, Yves Lecourtier **

*Université de Paris-Sud - LRI UMR 8623 , F-91405 Orsay cedex
Pierre.Lebodic@lri.fr
**Université de Rouen - LITIS , F-76800 Saint-Etienne du Rouvray
Prénom.Nom@univ-rouen.fr
http://www.litislab.eu

Résumé. Cet article propose une approche basée sur la programmation linéaire en nombres entiers pour la recherche d'isomorphisme de sous-graphes. L'originalité de l'approche réside dans la prise en compte d'un étiquetage du graphe par des vecteurs numériques pour lesquels les méthodes traditionnelles sont inopérantes. Cette approche est appliquée à la localisation de symboles dans des documents graphiques décrits par graphes d'adjacence de régions étiquetés par descripteurs de formes et des caractéristiques exprimant des relations géométriques. La globalité de l'approche est évaluée sur une base de documents synthétiques. Les résultats obtenus sont prometteurs.

1 Introduction

Les graphes étiquetés sont des structures de données appréciées pour leur aptitude à représenter des entités complexes. Au sein d'une représentation à base de graphe, les nœuds et leurs étiquettes décrivent des objets ou des parties tandis que les arcs représentent les relations entretenues entre les objets. En raison de la généricité intrinsèque des représentations à base de graphes, et grâce à l'augmentation de la puissance de calcul des ordinateurs, les représentations structurelles sont devenues de plus en plus utilisées dans de nombreux domaines d'application tels que la biologie, la chimie, l'analyse d'images de document ou la reconnaissance de symboles. Une conséquence de l'usage intensif des représentations à base de graphes est un intérêt croissant pour les problématiques scientifiques associées que sont la fouille de graphes, la classification de graphes (supervisée ou non) ou la recherche d'isomorphisme.

Cet article aborde le problème de la recherche d'isomorphisme de sous-graphe et propose une application dans le cadre de la détection de symboles dans des documents graphiques. En exploitant des graphes de régions adjacentes attribués pour représenter les documents et les symboles, nous présentons un nouveau cadre pour la recherche

d'isomorphisme de sous-graphe afin de trouver les instances de symboles dans les documents. Ce nouveau cadre modélise l'isomorphisme de sous-graphe comme un problème d'optimisation formulé comme un Programme Linéaire en Nombres Entiers (PLNE). La formulation proposée vise à trouver une correspondance exacte du point de vue de la topologie du graphe, tout en tolérant les erreurs du point de vue des étiquettes des nœuds et des arcs. Le problème est résolu par les méthodes d'optimisation combinatoire implémentées par des solveurs tels que, par exemple, SYMPHONY [1] qui est décrit dans Ralphs et Güzelsoy (2005).

Le système de recherche de symboles, dans sa globalité, est évalué sur un ensemble de 200 documents synthétiques contenant 5609 occurrences de symboles provenant de 16 classes. Ces documents sont produits par l'application décrite dans Delalandre et al. (2008).

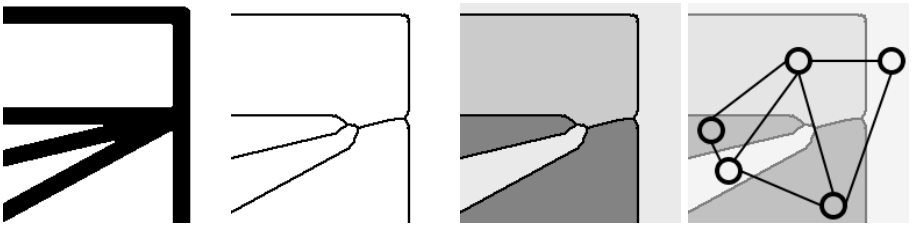
Le reste de cet article est organisé de la façon suivante. En section 2, nous présentons succinctement le processus permettant d'extraire les représentations structurelles des images de documents et de symboles. La section 3 présente la modélisation du problème de recherche d'isomorphisme par un programme linéaire en nombres entiers. En section 4, nous décrivons le protocole expérimental, les bases de données utilisées et les résultats obtenus lors de l'évaluation de l'approche, avant de dresser en section 5 les conclusions et de présenter quelques voies permettant de poursuivre ce travail.

2 Représentations structurelles

La détection de symboles est un des problèmes relevant de l'analyse d'image de document. Il s'agit de trouver l'emplacement de certains symboles dans une image de document. Ce type de problèmes revêt une difficulté supérieure à celui de la reconnaissance de symboles isolés dans le sens où il est nécessaire de simultanément segmenter et reconnaître le symbole. De fait, si la littérature abordant la reconnaissance de symboles est abondante, très peu d'approches sont proposées pour la détection de symboles (Rusiñol et al. (2009)). Dans cet article, nous présentons un système visant la détection de symboles dans un contexte particulier. Tout comme les travaux décrits dans Lladós et al. (2001), l'approche proposée s'appuie sur deux niveaux. Le premier niveau permet l'extraction d'une représentation structurelle à base de graphes d'adjacence de régions. Le second niveau vise à trouver les occurrences du graphe modèle représentant le symbole au sein du graphe représentant le document.

Nous décrivons, dans cette section, le premier niveau de l'approche proposée. Nous traitons des images de documents techniques (images binaires) où la composante blanche est associée au fond tandis que les composantes noires correspondent à la partie graphique. La segmentation de telles images peut être obtenue par étiquetage des composantes. Cependant, afin d'obtenir une représentation fine des relations d'adjacence pour chaque paire de régions, l'image binaire est soumise à une squelettisation. On fait alors correspondre à chaque composante blanche de cette image squelettisée un nœud dans le graphe en construction. Par ailleurs, un parcours des branches du squelette est exploité pour déterminer les relations d'adjacence entre les régions deux à deux. Cette

1. `http://www.coin-or.org/SYMPHONY/`



(a) Image originale (b) Squelette de l'image (a) (c) Composantes connexes de l'image (b) (d) Graphe d'adjacence de régions

FIG. 1 – *Création du graphe d'adjacence de régions*

relation d'adjacence est matérialisée par la création d'un arc entre les nœuds associés aux régions correspondantes. La figure 1 illustre, sur un extrait d'image de document, le processus de construction du graphe d'adjacence de régions.

Afin de caractériser les nœuds représentant les régions et de préciser la nature des relations d'adjacence, le graphe est étiqueté. Plusieurs types de caractéristiques ont été proposées dans la littérature pour décrire les formes et les relations spatiales. Parmi les descripteurs de formes, les moments de Zernike proposés dans Teague (1980) permettent d'atteindre de bonnes performances lors de la reconnaissance de formes soumises à des transformations affines ou des dégradations. Un vecteur de caractéristiques composé des 24 premiers moments de Zernike extraits de chaque composantes connexes et caractérisant la forme est donc utilisé pour étiqueter les nœuds correspondants dans le graphe. Le graphe construit est dirigé et les attributs affectés aux arcs ($source \rightarrow destination$) sont :

– une caractéristique liée au rapport des surfaces des composantes associées aux nœuds source et destination ;

$$\sqrt{\frac{A(destination)}{A(destination) + A(source)}}$$

– une caractéristique liée à la distance entre les centres de gravité des régions associées aux nœuds origine et destination.

$$\frac{d_e(g_{source}, g_{destination})}{\sqrt{A(source) + A(destination)}}$$

Au final, on obtient un graphe $G = (V, E, L_v, L_E)$ où V et $E \subseteq V \times V$ sont respectivement les ensembles de nœuds représentant les régions et d'arcs représentant les relations d'adjacence entre les régions. L_v et L_E sont les fonctions d'étiquetage des nœuds et des arcs.

– $L_V : V \rightarrow \mathbb{R}^d$, permet une description de la forme de la région associée.
– $L_E : E \rightarrow \mathbb{R}^2$, permet une description de la relation d'adjacence.

Du fait de la réciprocité de la relation d'adjacence, l'existence d'un arc ij implique l'existence d'un arc ji. Si l'étiquetage de ces arcs est identique du point de vue de la

caractéristique relative à la distance entre les centres de gravité des régions, chaque arc apporte une information différente sur la caractéristique relative au rapport des surfaces.

Un graphe d'adjacence de régions $\mathcal{G}$ est construit de la façon décrite précédemment à partir de l'image d'un document complet. La même méthode d'extraction est utilisée pour construire un graphe d'adjacence de régions $\mathcal{S}$ décrivant le symbole modèle. Dès lors, la détection des occurrences du symbole associé à $\mathcal{S}$ au sein du plan associé au graphe $\mathcal{G}$ se ramène à un problème de recherche d'isomorphismes de sous-graphe tolérante aux erreurs dans le sens où, du fait du bruit présent sur les images ou résultant de l'extraction, l'étiquetage des nœuds et des arcs peut différer entre le graphe décrivant le symbole isolé et ses différentes occurrences au sein du graphe décrivant le document.

Le recherche d'isomorphisme de sous-graphe est un problème connu pour être NP-complet Garey et Johnson (1979). Un des algorithmes faisant référence en la matière est l'algorithme d'Ullman, proposé dans Ullmann (1976). Cet algorithme permet une exploration de la combinatoire tout en permettant une réduction significative de la complexité par un principe de retour arrière. Un certain nombre d'articles ont proposé des algorithmes avec des complexités moindres exploitant certaines contraintes. C'est le cas des algorithmes proposés dans Hopcroft et Wong (1974) ou Cordella et al. (1999). D'autres approches telles que celle proposée dans Messmer et Bunke (1998) s'appuient sur un prétraitement de l'ensemble des graphes modèles. Certaines de ces approches sont des approches exactes. Les autres sont tolérantes aux erreurs. Les unes comme les autres imposent toutefois un étiquetage nominal des nœuds et des arcs du graphe. Leur utilisation pour traiter des graphes étiquetés par des attributs numériques est alors soumise à une étape préalable visant à transformer ces attributs numériques en attributs nominaux. Cette étape est généralement effectuée via une discrétisation ou une classification des attributs numériques. Cependant, une partie importante de l'information portée par les attributs numériques est alors perdue.

Nous n'avons pas connaissance d'algorithme permettant la recherche d'isomorphisme tolérant aux erreurs et apte à traiter directement des graphes où les nœuds et les arcs peuvent être étiquetés par des valeurs numériques vectorielles.

La section suivante présente la contribution principale de cet article, à savoir, un nouveau formalisme basé sur la programmation linéaire en nombres entiers pour la recherche d'isomorphismes de sous-graphe tolérante aux erreurs en présence de graphes étiquetés avec des valeurs numériques vectorielles, ce qui, à notre connaissance, présente une forte originalité.

3 Formulation 0-1 de l'isomorphisme de sous-graphe

3.1 Programmation linéaire en nombres entiers

Nous utilisons un programme linéaire en nombres entiers. Un programme linéaire en nombres entiers se présente sous la forme générique suivante :

$$\min_{x} \; c^t x \tag{1}$$

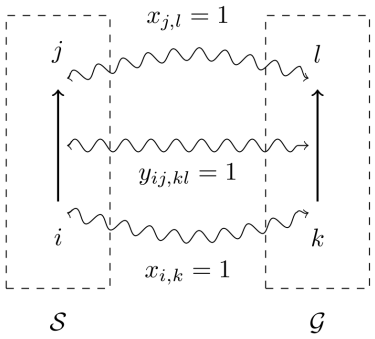


FIG. 2 – *Illustration de la mise en correspondance des nœuds et des arcs*

$$\text{sous la contrainte } Ax \leq b \tag{2}$$
$$x \in C \subseteq \mathbb{Z}^n \tag{3}$$

Dans cette formulation, $c \in \mathbb{R}^n, A \in \mathbb{R}^{n.m}, b \in \mathbb{R}^m$ sont les données du problème. Ce programme mathématique définit un ensemble de solutions pour le problème modélisé. La solution est un vecteur x de n variables, appartenant à $\mathbb{Z}$ dans le cas de la programmation en nombres entiers (3). Les variables permettent d'exprimer des contraintes linéaires (2). Une solution valide pour le problème est un vecteur x tel que les contraintes (2) et (3) sont respectées. Une telle solution est dite réalisable. Trouver une solution optimale consiste alors à minimiser la fonction objectif (1) sur l'ensemble des solutions réalisables.

3.2 Modélisation de la recherche d'isomorphismes de sous-graphe

La modélisation que nous proposons n'utilise que des variables binaires. Elles sont de deux sortes et illustrées sur la figure 2 :

- pour chaque nœud $i \in V_\mathcal{S}$ et pour chaque nœud $k \in V_\mathcal{G}$, il existe une variable $x_{i,k}$ telle que $x_{i,k} = 1$ si les nœuds i et k sont mis en correspondance et $x_{i,k} = 0$ sinon.
- pour chaque arc $ij \in E_\mathcal{S}$ et pour chaque arc $kl \in E_\mathcal{G}$, il existe une variable $y_{ij,kl}$ telle que $y_{ij,kl} = 1$ si les arcs ij et kl sont mis en correspondance et $y_{ij,kl} = 0$ sinon.

Étant donnés $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S})$ et $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$, supposons qu'on connaisse une distance $d_V : V_\mathcal{S} \times V_\mathcal{G} \to \mathbb{R}^+$ et une distance $d_E : E_\mathcal{S} \times E_\mathcal{G} \to \mathbb{R}^+$. Considérons deux nœuds $i \in V_\mathcal{S}$ et $k \in V_\mathcal{G}$. La mise en correspondance de i à k ($x_{i,k} = 1$) présente un coût de $d_V(i,k)$ alors que le fait de ne pas les associer ($x_{i,k} = 0$) présente un coût nul. Ce coût peut donc être noté $d_V(i,k) * x_{i,k}$. De même le coût d'association des arcs $ij \in E_\mathcal{S}$ et $kl \in E_\mathcal{G}$ peut être écrit $d_E(ij,kl) * y_{ij,kl}$. Il nous est alors possible d'écrire la fonction objectif qui vise à minimiser la somme des distances entre les éléments appariés :

$$\min_{x,y} \sum_{i \in V_\mathcal{S}} \sum_{k \in V_\mathcal{G}} d_V(i,k) * x_{i,k} + \sum_{ij \in E_\mathcal{S}} \sum_{kl \in E_\mathcal{G}} d_E(ij,kl) * y_{ij,kl} \tag{4}$$

Les contraintes du programme linéaire sont les suivantes :

– Chaque nœud de $\mathcal{S}$ doit être mis en correspondance avec un unique nœud de $\mathcal{G}$.

$$\sum_{k \in V_\mathcal{G}} x_{i,k} = 1 \quad \forall i \in V_\mathcal{S} \tag{5}$$

– Chaque arc de $\mathcal{S}$ doit être mis en correspondance avec un unique arc de $\mathcal{G}$.

$$\sum_{kl \in E_\mathcal{G}} y_{ij,kl} = 1 \quad \forall ij \in V_\mathcal{S} \tag{6}$$

– Chaque nœud de $\mathcal{G}$ doit être associé à au plus un nœud de $\mathcal{S}$.

$$\sum_{i \in V_\mathcal{S}} x_{i,k} \leq 1 \quad \forall k \in V_\mathcal{G} \tag{7}$$

– Si deux nœuds $i \in V_\mathcal{S}$ et $k \in V_\mathcal{G}$ sont mis en correspondance, un arc ayant i pour origine doit être mis en correspondance avec un arc ayant k pour origine.

$$\sum_{kl \in E_\mathcal{G}} y_{ij,kl} = x_{i,k} \quad \forall k \in V_\mathcal{G}, \forall ij \in E_\mathcal{S} \tag{8}$$

– Si deux nœuds $j \in V_\mathcal{S}$ et $l \in V_\mathcal{G}$ sont mis en correspondance, un arc ayant j pour destination doit être mis en correspondance avec un arc ayant l pour destination.

$$\sum_{kl \in E_\mathcal{G}} y_{ij,kl} = x_{j,l} \quad \forall l \in V_\mathcal{G}, \forall ij \in E_\mathcal{S} \tag{9}$$

– Enfin, nous précisons les contraintes indiquant que les variables du problème sont des variables binaires.

$$x_{i,k} \in \{0,1\} \quad \forall i \in V_\mathcal{S}, \forall k \in V_\mathcal{G} \tag{10}$$

$$y_{ij,kl} \in \{0,1\} \quad \forall ij \in E_\mathcal{S}, \forall kl \in E_\mathcal{G} \tag{11}$$

Les équations (4) à (11) forment le programme linéaire en nombres entiers utilisé pour résoudre la recherche d'isomorphisme. Toutes les solutions réalisables, celles qui satisfont les contraintes (5) à (11), correspondent toutes à un isomorphisme, cependant celle qui optimise la fonction objectif est celle pour laquelle le coût d'association est le plus faible.

Dès lors que la recherche d'isomorphisme est modélisée sous la forme d'un programme linéaire qu'il est possible de résoudre en utilisant un solveur mathématique. Dans cette étude, nous utilisons un solveur disponible sous licence CPL appelé SYMPHONY décrit dans Ralphs et Güzelsoy (2005). Dans la section suivante, nous présentons les expérimentations réalisées et discutons des résultats obtenus.

FIG. 3 – *Exemples d'images de la base* `floorplans` *correspondant à différents fonds de plan*

4 Évaluation

4.1 Présentation des données

Les données utilisées pour évaluer l'approche proposée sont extraites de la base `floorplans` [2]. Cette base est constituée de données synthétiques représentant différentes dispositions de symboles placés sur 10 fonds de plans architecturaux. Notre évaluation se base sur 200 images synthétiques de plans architecturaux correspondant aux 20 premières dispositions proposées pour chacun des fonds. Des exemples de ces images sont donnés sur la figure 3.

La tâche associée à cette base de données consiste à retrouver les occurrences des 16 symboles modèles présentés sur la figure 4.

Dans un premier temps, nous avons extrait la représentation structurelle de chacun des plans comme décrit en section 2.

Grâce à une interface graphique développée pour l'occasion, il a été possible de constituer une vérité terrain pour la recherche d'isomorphisme de sous-graphe en identifiant au sein des 200 représentations structurelles les sous-graphes correspondant à des occurrences de symboles. Nous avons ainsi pu identifier que la base de plans contenait 5609 occurrences de symboles, soit environ 28 symboles par document en moyenne. Les sous-graphes correspondant aux symboles contenaient en moyenne 4 nœuds et 7 arcs. En comparaison, les représentations structurelles des plans complets contiennent en moyenne 121 nœuds et 525 arcs.

2. `http://mathieu.delalandre.free.fr/projects/sesyd/`

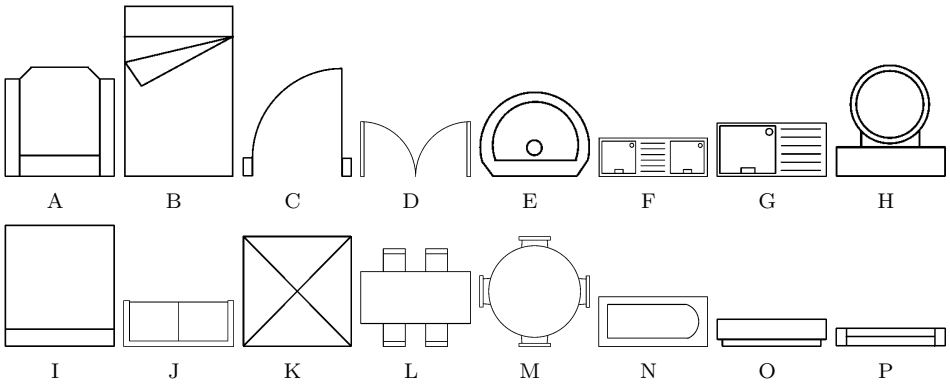


FIG. 4 – *Modèles des symboles recherchés*

4.2 Mesures de performance

À l'issue de la résolution par le solveur du programme linéaire en nombres entiers, nous disposons d'une proposition d'appariement entre le graphe représentant le symbole et celui représentant le plan en examinant lesquelles des variables x_{ik} et $y_{ij,kl}$ ont été mises à 1. Pour juger de la pertinence de cette proposition, il faut la comparer à la vérité terrain établie. Cette comparaison peut donner lieu à différents cas de figure :

- Dans le meilleur des cas, l'appariement proposé correspond exactement à celui de la vérité terrain (noté $=$)
- Certains des appariements proposés correspondent à la vérité terrain, alors que d'autres ne correspondent pas. (noté $\approx$)
- Aucun des appariements proposés ne correspond à la vérité terrain (noté $\neq$).
- Aucune proposition d'appariement n'est proposée. Ce cas de figure intervient lorsqu'aucune solution ne remplit les contraintes, c'est à dire qu'il n'existe pas de solution réalisable (noté $\emptyset$).

Dans le cas d'une proposition erronée d'isomorphisme, il convient de distinguer le cas où le symbole recherché est en réalité absent du document (noté $\neq |-$), auquel cas l'erreur est naturelle, du cas où le symbole est effectivement présent dans la vérité terrain (noté $\neq |+$).

Par ailleurs, si le solveur permet de retrouver la solution qui minimise la fonction objectif, il est possible de rechercher plusieurs occurrences en relançant une nouvelle recherche dans laquelle sera exclue la solution déjà trouvée, et ainsi de suite. Il est ainsi possible de rechercher plusieurs isomorphismes. Bien que d'autres configurations soient possibles, compte-tenu du contexte applicatif, lors de la recherche d'un nouvel isomorphisme, nous excluons des solutions possibles tout isomorphisme dans lequel apparaît un nœud déjà apparié dans un isomorphisme précédent.

4.3 Recherche d'une unique occurrence

Dans une première expérimentation, nous avons recherché, dans chacun des plans, l'isomorphisme de coût minimal pour chacun des symboles modèles. Les résultats quan-

Symbole	$=$	$\approx$	$\neq \mid +$	$\neq \mid -$	$\emptyset$
A	88	3	15	94	0
B	48	72	40	40	0
C	56	72	72	0	0
D	0	11	49	140	0
E	112	0	0	88	0
F	96	34	0	70	0
G	51	21	79	49	0
H	40	128	20	12	0
I	114	0	62	24	0
J	170	2	20	8	0
K	154	30	16	0	0
L	123	0	0	77	0
M	85	6	0	109	0
N	179	0	1	20	0
O	124	0	59	17	0
P	172	1	20	7	0
Total	1612	380	413	755	0

TAB. 1 – *Résultats de la recherche d'une occurrence de symbole par plan*

titatifs sont présentés dans le tableau 1. Globalement, ces résultats indiquent que sur les 3200 occurrences recherchées, 1612 correspondent exactement à des symboles présents sur les plans, 380 correspondent partiellement (au moins 1 nœud apparié à bon escient et au moins un nœud apparié à mauvais escient). 755 recherches ont été effectuées alors que le symbole n'était pas présent dans le plan. Finalement, le système n'a produit de véritables erreurs ($\neq \mid +$) que dans 453 des 2445 cas où un symbole pouvait être trouvé. On remarque également que les deux dernières colonnes du tableau indiquent que le système a toujours proposé une solution réalisable lors de la recherche d'une unique occurrence.

Le tableau 1 révèle également des disparités entre des classes proposant de bons résultats (par exemple les classes E, M et N) et d'autres pour lesquelles les résultats sont très faibles (classes C, D et G). Deux explications peuvent être données. Concernant les classes C et D, la représentation en graphe d'adjacence de régions n'est pas adaptée à ces symboles qui ne présentent que deux régions blanches non adjacentes (cf. Fig. 4C et 4D). Le graphe correspondant est donc restreint à deux nœuds non reliés par un arc. De ce fait, il n'existe pas de contrainte topologique pour l'appariement ce qui conduit à de nombreuses erreurs. Concernant la classe G, on peut observer que le symbole correspondant est complètement inclus dans celui de la classe F, ce qui, là aussi, est source de confusions.

4.4 Recherche de plusieurs occurrences

Étant donnés les résultats obtenus lors de la recherche d'une unique occurrence et considérant le fait qu'un même symbole est susceptible d'apparaître à plusieurs reprises

Symbole	$=$	$\approx$	$\neq \mid +$	$\neq \mid -$	$\emptyset$	rappel	précision
A	210	7	328	2090	7365	0,88	0,08
B	157	131	166	2367	7179	0,96	0,10
C	158	669	1882	6849	442	0,98	0,09
D	19	61	1092	8386	442	0,80	0,01
E	112	0	0	5325	4563	1,00	0,02
F	104	46	0	892	8958	1,00	0,14
G	98	110	126	2642	7024	1,00	0,07
H	40	168	32	3615	6145	1,00	0,05
I	397	6	1368	6784	1445	0,93	0,05
J	233	2	503	2935	6327	0,92	0,06
K	473	427	987	1063	7050	0,83	0,30
L	132	0	0	954	8914	1,00	0,12
M	90	6	0	1080	8824	1,00	0,08
N	180	0	18	8438	1364	1,00	0,02
O	467	0	1899	6257	1377	0,92	0,05
P	648	3	311	1706	7332	0,86	0,24
Total	3518	1636	8712	60383	84751		

TAB. 2 – *Résultats de la recherche de 50 occurrences de symbole par plan*

sur un même document, nous avons souhaité évaluer la recherche de plusieurs isomorphismes. Compte-tenu du fait qu'une composante connexe ne peut appartenir qu'à un seul symbole, nous avons, dans cette expérimentation, paramétré la recherche de telle sorte que soit exclu des solutions réalisables tout isomorphisme faisant apparaître un nœud déjà apparié dans un isomorphisme précédent.

Le tableau 2 présente les résultats d'une recherche de 50 occurrences de chaque symbole modèle pour chacun des 200 plans de la base `floorplans`. Dans ce tableau, les colonnes $\neq \mid -$ et $\emptyset$ indiquent respectivement le nombre d'erreurs et de recherches infructueuses pour cause d'absence de solution réalisable sachant que toutes les occurrences réellement présentes ont été trouvées précédemment. Il ne s'agit donc pas à proprement parler d'erreur. Même s'il persiste, comme dans le cas de la recherche d'une unique occurrence, des disparités entre classes, globalement, on retrouve exactement 3518 (62,7%) et partiellement 1636 (29,2%) des 5609 occurrences de symboles réellement présentes dans les documents. En estimant qu'une correspondance partielle suffit à considérer que l'occurrence du symbole est détectée, on atteint un rappel de 92%.

Ces bons résultats doivent cependant être tempérés. En effet, un nombre non négligeable de fausses détections sont opérées avant d'avoir retrouvé la dernière occurrence réelle du symbole cherché (colonne $\neq \mid +$). Parallèlement, en l'état actuel, notre système n'est pas en mesure de déterminer le nombre de symboles de chaque type présents sur un plan. De ce fait, dans cette expérimentation, nous avons volontairement choisi une valeur importante du nombre de détections à proposer au regard du nombre de symboles réellement présents (recherche de 50 occurrences par type de symbole pour chaque plan) sans que cela soit suffisant pour les retrouver tous (rappel < 1). Mais, *a contra-*

rio, cela a conduit le système à proposer des occurrences possibles alors que toutes les occurrences réelles avaient déjà été trouvées (colonne $\neq |-$). Enfin, la recherche s'est dans certains cas interrompue d'elle-même lorsqu'il n'existait plus de solution réalisable (colonne $\emptyset$). Compte-tenu de ce nombre important de fausses détections, la recherche de 50 occurrences de chaque symbole pour chaque plan a aboutit à un rappel de 92% pour une précision de 7%. Nos travaux futurs s'emploieront à proposer des compromis précision/rappel permettant de proposer des modes de fonctionnement intermédiaires en adaptant le nombre d'occurrences cherchées.

5 Conclusion

Dans cet article, nous présentons une approche pour la recherche d'isomorphisme exact de sous-graphe tolérant aux erreurs basée sur une programmation linéaire en nombres entiers. Cette approche est implantée grâce l'utilisation d'un solveur libre et appliquée à la recherche d'occurrences de symbole dans des documents architecturaux en utilisant des représentations à partir de graphe pour décrire aussi bien le plan que les symboles à chercher.

Les expérimentations réalisées sur des bases publiques montrent des performances intéressantes. En effet, lors de la recherche d'une occurrence de symbole, si celui-ci est effectivement présent, une de ses occurrences est retrouvée au moins partiellement dans plus de 80% des cas. De même, lors de recherche de multiples occurrences, les symboles sont localisés dans plus de 60% des cas exactement et dans presque 30% des cas partiellement.

Cependant, ce taux de rappel est obtenu au détriment d'une précision relativement faible. En effet, notre système ne permet pas en l'état de déterminer automatiquement le nombre d'occurrences à localiser, et si certaines occurrences sont trouvées tardivement, de nombreuses fausses détections sont expliquées par le fait bon nombre de recherches sont lancées sans qu'il n'y ait plus de nouveaux symboles à retrouver.

Nous envisageons de mener une étude qui permettrait de palier cette carence. Pour ce faire, nous projetons d'établir une étude statistique sur la valeur des coûts des isomorphismes (valeur de la fonction objectif). De cette façon, nous pourrions pour chaque classe de symbole établir des valeurs seuils de la fonction objectif qui permettraient éventuellement de prendre des décisions de rejet des isomorphismes associés à un coût supérieur.

Références

Cordella, L. P., P. Foggia, C. Sansone, et M. Vento (1999). Performance evaluation of the vf graph matching algorithm. In *Proccedings of the International Conference on Image Analysis and Processing*, pp. 1172–1177.

Delalandre, M., T. Pridmore, E. Valveny, H. Locteau, et E. Trupin (2008). Building synthetic graphical documents for performance evaluation. In *Graphics Recognition. Recent Advances and New Opportunities*, Volume 5046 of *Lecture Notes in Computer Science*, pp. 288–298.

Garey, M. R. et D. S. Johnson (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness*. Freeman & co.

Hopcroft, J. et J. Wong (1974). Linear time algorithm for isomorphism of planar graphs. In *Proceedings of the sixth annual ACM Symp. Theory of Computing*, pp. 172–184.

Lladós, J., E. Martí, et J. J. Villanueva (2001). Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*(10), 1137–1143.

Messmer, B. T. et H. Bunke (1998). A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Inteligence 20*(5), 493–504.

Ralphs, T. K. et M. Güzelsoy (2005). *The Next Wave in Computing, Optimization, and Decision Technologies*, Volume 29 of *Operations Research/Computer Science Interfaces Series*, Chapter The Symphony Callable Library for Mixed Integer Programming, pp. 61–76. Springer US.

Rusiñol, M., J. Lladós, et G. Sánchez (2009). Symbol spotting in vectorized technical drawings throug a lookup table of region strings. *Pattern Analysis and Applications*.

Teague, M. (1980). Image analysis via the general theory of moments. *Journal of the Optical Society of America 70*(8), 920–930.

Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *Journal of the ACM 23*(1), 31–42.

Summary

This paper proposes an approach based on integer linear programming to solve the subgraph isomorphism problem. This approach is able to handle graphs labeled with numerical feature vectors for which traditional methods can not be used. This approach is applied for symbol localization in technical document images decribed with region adjacency graphs labeled with shape descriptors and feature vectors describing geometric relations. The whole approach is evaluated on a synthetic document database. Preliminary results are encouraging.

Une Approche Skyline pour l'Interrogation de Bases de Données de Graphes

Katia Abbaci*, Allel Hadjali*,
Ludovic Liétard**, Daniel Rocacher*

*IRISA/ENSSAT
Rue de Kérampont BP 80518 Lannion, France
{Katia.Abbaci, Allel.Hadjali, Daniel.Rocacher}@enssat.fr
**IRISA/IUT
Rue Edouard Branly BP 30219 Lannion, France
Ludovic.Lietard@univ-rennes1.fr

Résumé. La recherche de graphes similaires à une requête à graphe constitue l'un des problèmes fondamentaux dans les bases de données de graphes. Les approches existantes traitant ce problème s'appuient, généralement, sur une seule mesure de similarité entre les structures de graphes. Dans cet article, nous proposons une approche permettant de rechercher les graphes similaires au graphe d'une requête où la similarité entre graphes n'est plus un scalaire unique mais un vecteur de scalaires. Pour cela, nous introduisons le concept de *skyline par similarité* d'une requête à graphe défini par un sous-ensemble de graphes, de la base de données interrogée, qui sont *les plus similaires* à la requête au *sens de Pareto*. Une méthode permettant de raffiner le résultat de la recherche est aussi proposée en s'appuyant sur le critère de *diversité* entre graphes.

1 Introduction

Les graphes sont devenus de plus en plus importants dans la modélisation des données structurées et complexes dans différents domaines d'applications récents : la bio-informatique (Hu et al., 2005), la reconnaissance de formes (Conte et al., 2004), les documents XML (Zhang et al., 2006), la chimie (Klinger et Austin, 2005), les réseaux sociaux (Cai et al., 2005), etc. Toutes ces applications indiquent l'importance et la large utilisation du paradigme des Bases de Données de Graphes (BDGs). D'une manière générale, on peut classer les requêtes adressées à une BDG en deux catégories (Zeng et al., 2009) : (1) *la recherche de graphes fondée sur une relation d'inclusion* et (2) *la recherche de graphes par similarité*. La première catégorie se compose de deux sous-problèmes suivants : (i) *la recherche de sous-graphes* : soit une BDG $D = \{g_1, g_2, \cdots, g_n\}$ et une requête à graphe q (dite requête sous-graphe), il s'agit de rechercher tous les graphes $g_i \in D$ tel que q est un sous-graphe de g_i (i.e., $q \subseteq g_i$) ; (ii) *la recherche de super-graphes* : soit une BDG $D = \{g_1, g_2, \cdots, g_n\}$ et une requête à graphe q (dite requête super-graphe), il s'agit de rechercher tous les graphes $g_i \in D$ tel que q est un super-graphe de g_i (i.e., $q \supseteq g_i$). Les deux sous-problèmes font appel à la *procédure de vérification d'isomorphisme de sous-graphes*, qui est NP-Complet. Ainsi, plusieurs approches

de traitement de requêtes à graphe, utilisant des techniques d'indexation, ont été développées pour réduire l'espace de recherche et résoudre efficacement ces deux sous-problèmes (Chen et al., 2007; Yan et al., 2004; Zhang et al., 2007, 2009).

Quant à la deuxième catégorie (i.e., la recherche de graphes par similarité), qui consiste à rechercher tous les graphes d'une BDG structurellement similaires au graphe de la requête, est apparue comme une nouvelle tendance pour les raisons suivantes. Premièrement, de nombreuses et réelles BDGs sont de nature bruitée et incomplète, d'où la nécessité d'un *appariement approximatif* de graphes. Deuxièmement, plusieurs applications modernes préfèrent les résultats d'un appariement approximatif plutôt que ceux d'un appariement exact car ils véhiculent davantage d'informations, comme ce qui pourrait être manquant ou superflu dans un graphe de requête ou dans une BDG. Ainsi, plusieurs approches ont été proposées pour répondre aux requêtes de recherche de graphes par similarité (ou simplement, requêtes par similarité). Voir (Yan et al., 2005; He et Singh, 2006; Tian et Patel, 2008; Shang et al., 2010).

Le point commun entre toutes ces approches est l'utilisation d'une *seule* mesure pour évaluer la similarité entre graphes. Toutefois, un graphe est une structure complexe et comprend une multitude de caractéristiques de base. Il est alors difficile de donner une définition significative de la similarité entre graphes en utilisant un seul scalaire.

Dans cet article, nous préconisons que plusieurs indices sont nécessaires pour évaluer d'une manière significative et efficace la similarité entre graphes. Chaque indice est dédié à mesurer une distance (ou similarité) locale afférente à un aspect donné dans la structure du graphe. Ainsi, la similarité entre graphes est caractérisée par un vecteur de mesures de distances locales au lieu d'une seule mesure. De cette façon, on peut préserver l'information concernant la similarité sur différentes caractéristiques, lorsque l'on compare deux graphes.

Nous proposons une nouvelle approche de traitement de requêtes par similarité en utilisant la notion de *skyline par similarité*. D'une manière générale, le skyline par similarité d'une requête à graphe est défini par le sous-ensemble des graphes de la BDG interrogée *les plus similaires* à la requête au sens de Pareto. L'idée est d'effectuer une comparaison multidimensionnelle entre graphes en termes de d mesures de distances locales et d'identifier les graphes qui sont *maximalement similaires* au sens d'une relation de *dominance par similarité*. Les principales contributions de l'article sont :

1. Nous introduisons la notion de *similarité composée entre graphes* (SCG) et définissons ensuite la *relation de dominance par similarité entre graphes*.
2. En se basant sur cette relation de dominance, nous proposons une définition formelle du *skyline de graphes par similarité*, i.e., les graphes de la base interrogée maximalement similaires au graphe de la requête au sens de Pareto.
3. Pour réduire la taille du skyline, qui est souvent très importante, nous proposons une méthode permettant d'extraire un sous-ensemble de graphes qui est *aussi divers que possible*, mais dont la taille est raisonnable.

L'article est structuré comme suit. La section 2 introduit quelques notions préliminaires. La section 3 discute quelques travaux apparentés. La section 4 décrit certaines mesures de similarité entre graphes et leurs interprétations. Dans la section 5, nous introduisons la notion de skyline par similarité dédiée aux requêtes à graphe. La section 6 présente un exemple détaillé. Dans la section 7, nous proposons une méthode pour raffiner les résultats retournés dans le skyline. La section 8 conclut l'article.

2 Notions préliminaires

2.1 Rappel sur les requêtes skyline

Les requêtes skyline (Borzsonyi et al., 2001) représentent un paradigme très populaire et puissant pour extraire des objets d'un ensemble de données multidimensionnelles. Elles s'appuient sur le principe de dominance de Pareto qui peut être défini comme suit :

Définition 1. Soit *r* un ensemble de points multidimensionnels et $p = (p_1, p_2, \ldots, p_d)$ et $q = (q_1, q_2, \ldots, q_d)$ deux points de *r*. On dit que *p domine* (au sens de Pareto) *q* ssi sur chaque dimension $p_i \leq q_i$ (pour $1 \leq i \leq d$) et sur au moins une dimension $p_j < q_j$.

Par souci de simplicité et sans perte de généralité, nous supposons que plus la valeur p_i est petite, meilleure elle est. On dit alors que *p domine (est préféré à) q* et on note $p \succ q$.

Définition 2. Le skyline de r est le sous-ensemble de r des points non-dominés par aucun autre point de *r*.

Les requêtes skyline calculent donc l'ensemble des tuples optimaux au sens de Pareto dans une relation, i.e., les tuples qui ne sont dominés par aucun autre tuple de la même relation.

Exemple 1. Considérons une base de données contenant des informations sur des hôtels comme indiqué dans le tableau 1 (où la dimension d = 2).

Hôtel	Prix(€)	Distance (Km)
H_1	4.0	150
H_2	3.0	110
H_3	2.5	240

TAB. 1 – *Exemple d'hôtels.*

Considérons une personne qui cherche un hôtel aussi proche que possible de la plage et ayant un prix faible. On peut vérifier que le skyline résultant *S* contient les hôtels H_2 et H_3, car H_1 est dominé par H_2.

2.2 Quelques définitions de base

Définition 3 (Graphe). Un graphe *g* est défini par un quadruplet *(V, E, L, l)* où *V* est l'ensemble des nœuds, *E* est l'ensemble des arêtes, *L* est l'ensemble des étiquettes et *l* est la fonction d'étiquetage qui met en correspondance chaque nœud ou arête avec une étiquette de *L*.

Par souci de clarité, les graphes considérés sont étiquetés et non-orientés (différents nœuds peuvent avoir la même étiquette). La taille d'un graphe *g* est définie comme suit : |g| = |E(g)| (i.e., la taille d'un graphe est le nombre de ses arêtes).

Définition 4 (Isomorphisme de graphes). Soit deux graphes *g = (V, E, L, l)* et *g' = (V', E', L', l')*, *g* est *isomorphe à g'* (dénoté par $g \approx g'$) s'il existe une bijection *f : V → V'*, telle que

1. $\forall$ *v* $\in$ *V, f(v)* $\in$ *V' et l(v) = l'(f(v))* ;
2. $\forall$ *(u, v)* $\in$ *E, (f(u), f(v))* $\in$ *E' et l(u, v) = l'(f(u), f(v))*.

Définition 5 (Isomorphisme de sous-graphes). Soit deux graphes *g = (V, E, L, l)* et *g' = (V', E', L', l')*, *g* est *isomorphe de sous-graphe à g'* s'il existe une injection *f : V → V'*, telle que

1. $\forall$ *v* $\in$ *V, f(v)* $\in$ *V' et l(v) = l'(f(v))* ;
2. $\forall$ *(u, v)* $\in$ *E, (f(u), f(v))* $\in$ *E' et l(u, v) = l'(f(u), f(v))*.

Définition 6 (Sous-graphe v.s. super-graphe). Soit deux graphe *g = (V, E, L, l)* et *g' = (V', E', L', l')*, *g* est dit *sous-graphe de g'* (ou *g'* est un *super-graphe de g*), dénoté par $g \subseteq g'$ (ou $g' \supseteq g$), s'il existe un isomorphisme de sous-graphes de *g* à *g'*.

Définition 7 (Sous-graphe commun maximal, SCM). Soit deux graphes g_1 et g_2, le sous-graphe commun maximal de g_1 et g_2 est le plus grand sous-graphe connexe de g_1 qui est isomorphe de sous-graphe à g_2, dénoté par *g'= SCM(*g_1*,* g_2*)*.

3 Travaux apparentés

L'étude présentée dans cet article peut être apparentée avec les travaux effectués dans les domaines des requêtes skyline et des requêtes de recherche de graphes par similarité.

Requêtes skylines. Durant ces dernières années, les requêtes skylines ont reçu l'attention de nombreux chercheurs. Plusieurs études ont été menées pour développer des algorithmes efficaces et introduire des variantes pour les requêtes skyline (Pei et al., 2007; Yiu et Mamoulis, 2007; Khalefa et al., 2008; Hadjali et al., 2010). Pour autant que nous le sachions, il n'existe pas de travaux portant sur les requêtes skyline dans un contexte d'interrogation de BDGs, excepté, le travail de Zou et al. (2010) qui étudie les requêtes skyline dynamique dans le cadre d'un graphe de grande taille. Dans notre cas, c'est un autre type de skyline (i.e. skyline par similarité) sur un ensemble de graphes qui est étudié.

Requêtes par similarité. Plusieurs approches ont été développées pour traiter les requêtes par similarité. *Grafil* (Yan et al., 2005) applique une recherche par similarité de sous-structures dans une BDG à large échelle. Il retourne tous les graphes de la base de données qui contiennent approximativement le graphe de la requête. *C-Tree* (He et Singh, 2006) est un autre outil de recherche par similarité de sous-graphes. Il est basé sur la distance d'édition entre la requête et les graphes candidats. *Tale* (Tian et Patel, 2008) propose une technique d'appariement innovante qui distingue les nœuds par leur importance dans la structure des graphes. Cette technique met d'abord en correspondance les nœuds importants d'une requête à graphe, ensuite, elle étend progressivement ces correspondances. Récemment, Shang et al. (2010) ont proposé une technique répondant aux requêtes super-graphes où le problème de recherche par similarité de super-graphes est converti en un problème de détection de σ-sous-graphes manquants, où σ est un seuil de tolérance d'erreurs. Les deux approches *C-Tree* et *Tale* utilisent la distance d'édition pour mesurer la similarité entre graphes, tandis que les travaux réalisés par Yan et al. (2005) et Shang et al. (2010) utilisent la notion de *sous-graphe commun maximal*.

Comme on peut le constater, toutes les approches proposées et dédiées aux requêtes par similarité utilisent un seul indice pour mesurer la similarité entre deux graphes. En procédant ainsi, la similarité entre deux graphes n'est pas entièrement capturée car des similitudes relatives à certaines caractéristiques du graphe pourraient être ignorées. Ceci est principalement dû au fait que chaque indice de similarité entre graphes peut être vu comme une mesure locale qui exprime seulement une ressemblance au regard d'un seul aspect dans la structure des graphes (voir la section 4). Par comparaison avec les travaux ci-dessus, notre approche, d'une part, repose sur une mesure de similarité composée entre graphes, et d'autre part, retourne un ensemble de graphes dominants par similarité au sens de Pareto pour répondre à une requête.

4 Similarité de graphes : une vue sémantique

Plusieurs modèles ont été proposés (Bunke, 1997; Bunke et Shearer, 1998; Wallis et al., 2001) pour mesurer la similarité (ou la distance) entre deux graphes. Ci-après, nous présentons les mesures les plus utilisées pour déterminer les similarités entre graphes[1].

4.1 La distance d'édition de graphes

La distance d'édition de graphes (Bunke, 1997; He et Singh, 2006) se base sur les opérations d'édition de graphes nécessaires pour transformer un graphe en un autre. Généralement, l'ensemble d'opérations d'édition considéré inclut : l'insertion, la suppression et le ré-étiquetage de nœuds/d'arêtes. Chaque opération d'édition est associée à un coût (un nombre réel non négatif) en fonction de l'intensité de distorsion induite par la transformation. Soit *e_op* une opération d'édition et *c(e_op)* son coût. Le coût d'une séquence d'opérations d'édition, $s = e\_op_1, \ldots, e\_op_n$ est donné par

$$c(s) = \sum_{i=1}^{n} c(e\_op_i).$$

En pratique, il est difficile de déterminer le coût de chaque opération d'édition élémentaire. Par souci de simplicité, on considère, ici, ce coût égal à une mesure de distance *uniforme* : la distance entre deux nœuds/arêtes est *1* si leurs étiquettes sont différentes ; *0* sinon.

Définition 8 (Distance d'édition de graphes). La distance d'édition entre deux graphes g_1 et g_2 est égale au *coût minimal*, résultant de toutes les séquences d'opérations d'édition possibles, qui transforment g_1 en g_2, i.e.,

$$Dist_{Ed}(g_1, g_2) = min_{s \in E\_op} c(s) \quad (1)$$

où *E_op* dénote l'ensemble de toutes les séquences d'opérations d'édition possibles qui transforment g_1 en g_2. Plus $Dist_{Ed}(g_1, g_2)$ est faible, plus les deux graphes sont similaires. Cette mesure de distance s'applique à tout type de graphes sans aucune restriction.

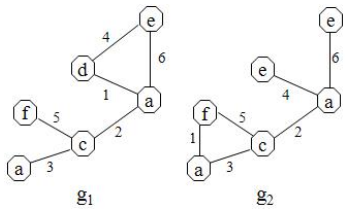


FIG. 1 – *Exemple de graphes étiquetés.*

Exemple 2. Considérons les graphes étiquetés de la figure 1. La séquence d'opérations d'édition nécessaire pour transformer g_1 en g_2 est : (i) une suppression d'arêtes, i.e., l'arête *(d, e)*, (ii) un ré-étiquetage d'arêtes, i.e., changer l'étiquette de l'arête *(a, d)* de *1* à *4*, (iii) un ré-étiquetage de nœuds, i.e., changer l'étiquette du nœud *d* de *d* à *e*, (iv) une insertion d'arêtes

[1] Par souci d'espace, l'analyse de la complexité du calcul de chaque mesure n'est pas abordée dans cet article.

i.e., l'arête *(a, f)* avec l'étiquette *1*. En utilisant des mesures de distance uniformes, on peut vérifier que cette séquence est la meilleure (i.e., la minimale). Ainsi, $Dist_{Ed}(g_1, g_2) = 4$.

Dans le contexte d'interrogation de BDGs, cette mesure de distance nous renseigne sur les caractéristiques non partagées par un graphe cible de la BDG et le graphe de la requête.

4.2 La distance basée sur le SCM

Bunke et Shearer (1998) ont développé un autre type de mesures de similarité entre graphes qui est basée sur *le sous-graphe commun maximal (SCM)*.

Définition 9 (Similarité basée sur le SCM). Soit deux graphes g_1 et g_2, la similarité entre graphes basée sur le *SCM* est définie comme suit,

$$Sim_{SCM}(g_1, g_2) = \frac{|SCM(g_1,g_2)|}{max(|g_1|,|g_2|)},$$

où $|SCM(g_1, g_2)|$ dénote le nombre d'arêtes dans $SCM(g_1, g_2)$.

Plus le *SCM* de deux graphes est large, plus leur similarité est élevée. La mesure Sim_{SCM} est normalisée (i.e., $0 \leq Sim_{SCM}(g_1, g_2) \leq 1$) car $|SCM(g_1, g_2)| \leq max(|g_1|, |g_2|)$. Ainsi, la mesure de distance entre graphes, $Dist_{SCM}$, dérivée de Sim_{SCM} peut s'écrire :

$$Dist_{SCM}(g_1, g_2) = 1 - Sim_{SCM}(g_1, g_2) \quad (2)$$

L'avantage principal de l'approche basée sur le *SCM* est la non-utilisation de fonctions de coût, palliant ainsi l'inconvénient principal de l'approche basée sur la distance d'édition.

Example 3. Reprenons l'exemple 2. La distance basée sur le *SCM* entre g_1 et g_2 est calculée comme suit. Premièrement, le *SCM* (g_1, g_2) est identifié, voir la figure 2. Ensuite, par application de (2), nous obtenons

$Dist_{SCM}(g_1, g_2) = 1 - \frac{|SCM(g_1,g_2)|}{max(|g_1|,|g_2|)} = 0.33,$

où $|SCM(g_1, g_2)| = 4$ et $max(|g_1|, |g_2|) = 6$.

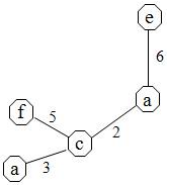


FIG. 2 – *SCM de g_1 et g_2.*

Dans le contexte d'interrogation de BDGs, le *SCM* de deux graphes véhicule de l'information sur les caractéristiques partagées par un graphe de la base interrogée et la requête.

4.3 La distance basée sur l'UG

La mesure de *distance basée sur l'union de graphes (UG)*, proposée par Wallis et al. (2001), est basée sur le principe de l'union de graphes.

Définition 10 (Similarité basée sur l'UG). Soit deux graphes g_1 et g_2, la similarité entre graphes basée sur l'union de graphes est définie comme suit,

$$Sim_{UG}(g_1, g_2) = \frac{|SCM(g_1,g_2)|}{|g_1|+|g_2|-|SCM(g_1,g_2)|},$$

où le dénominateur représente la taille de l'union des deux graphes selon une vue ensembliste[2].

Cette mesure de similarité est aussi normalisée et son comportement est assez proche de celui de Sim_{SCM}. Il est facile de voir que $Sim_{UG}(g_1, g_2) \leq Sim_{SCM}(g_1, g_2)$ (ce qui signifie que Sim_{UG} est une mesure plus exigeante que Sim_{SCM}). L'utilisation de l'union de graphes (Wallis et al., 2001) est motivée par le fait que les changements dans la taille du *plus petit graphe* qui préservent le *SCM*(g_1, g_2) constant ne sont pas pris en compte dans $Sim_{SCM}(g_1, g_2)$, tandis que la mesure $Sim_{UG}(g_1, g_2)$ prend en compte cette variante.

La mesure de distance de graphes dérivée de Sim_{UG} s'écrit :

$$Dist_{UG}(g_1, g_2) = 1 - Sim_{UG}(g_1, g_2) \tag{3}$$

Exemple 4. Reprenons encore une fois les graphes de l'exemple 2. Par application de (3), la distance basée sur l'UG entre g_1 et g_2 est

$Dist_{UG}(g_1, g_2) = 1 - \frac{|SCM(g_1,g_2)|}{|g_1|+|g_2|-|SCM(g_1,g_2)|} = 0.50,$

où $|SCM(g_1, g_2)| = 4$ (voir l'example 3) et $|g_1| = |g_2| = 6$.

Dans le cadre d'interrogation de BDGs, cette similarité donne également des informations sur les aspects communs entre un graphe de la base interrogée et le graphe de la requête.

5 Skyline de graphes par similarité

Dans ce qui suit, nous supposons que la similarité entre graphes est une notion composée, i.e., un vecteur de mesures de distances.

Définition 11 (Similarité composée entre graphes, SCG). Soit *g* et *g'* deux graphes, une similarité composée entre *g* et *g'* est un vecteur de mesures de distances locales, dénotée par

$$SCG(g, g') = (Dist_1(g, g'), Dist_2(g, g'), \ldots, Dist_d(g, g')),$$

où $Dist_i(g, g')$, pour i = 1, . . ., d, représente une mesure de distance locale entre *g* et *g'*.

Soit $D = \{g_1, g_2, \ldots, g_n\}$ une BDG et q une requête par similarité. Pour répondre à q, l'idée est de procéder à une comparaison multidimensionnelle entre graphes en termes de d mesures de distances (locales) pour rechercher les graphes qui sont maximalement similaires à q au sens de la relation de dominance par similarité définie ci-dessous.

Définition 12 (Relation de dominance par similarité). Soit une requête à graphe q et deux graphes *g* et *g'*, on dit que *g' est dominé par similarité* par *g* dans le contexte de q, dénoté par $g \succ_q g'$, ssi les deux conditions suivantes sont vérifiées :

1. $\forall i \in \{1, \ldots, d\}, Dist_i(g, q) \leq Dist_i(g', q)$,
2. $\exists k \in \{1, \ldots, d\}, Dist_k(g, q) < Dist_k(g', q)$.

[2]Cette mesure de similarité ressemble à l'indice de Jaccard utilisé pour mesurer la similarité entre deux ensembles *A* et *B*, i.e., $J(A, B) = |A \cap B| \,/\, |A \cup B|$.

Plus simplement, la relation $g \succ_q g'$ est vérifiée si g n'est pas moins similaire à q que g' dans toutes les dimensions et (strictement) plus similaire à q que g' dans au moins une dimension. On peut observer que g est potentiellement plus intéressant que g' comme graphe réponse. Par conséquent, l'ensemble des graphes les plus similaires à q sont ceux qui ne sont pas dominés (au sens de la définition 13). De tels graphes, appelés *graphes optimaux au sens de Pareto*, représentent ce que nous dénotons par *le skyline de graphes par similarité (SGS)* :

$$SGS(D, q) = \{g \in D | \nexists g' \in D, g' \succ_q g\} \quad (4)$$

où $g' \succ_q g$ signifie que g est dominé par similarité par g' dans le contexte de q.

Pour illustrer notre approche, nous présentons dans la section suivante un exemple où $d = 3$. $SCG(g, q)$ est alors un vecteur de trois composantes exprimées en termes de mesures de distances locales décrites dans la section 4, i.e.,

$$SCG(g, q) = (Dist_{Ed}(g, q), Dist_{SCM}(g, q), Dist_{UG}(g, q)).$$

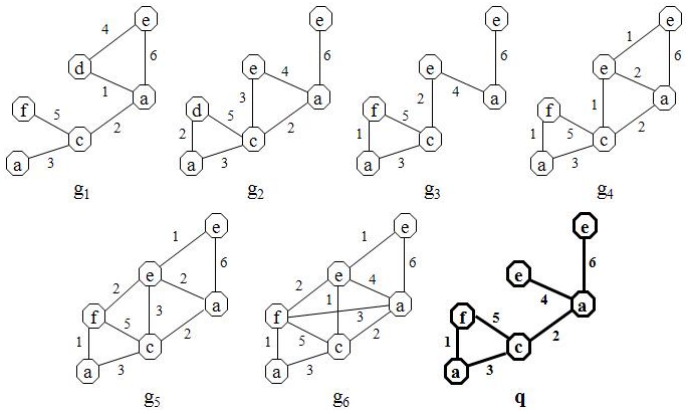


FIG. 3 – *La base de données D et le graphe requête q.*

6 Un exemple illustratif

Soit $D = \{g_1, g_2, g_3, g_4, g_5, g_6\}$ une BDG et q un requête par similarité (voir la figure 3). Afin de retourner les réponses les plus intéressantes à q, on calcule le skyline de graphes par similarité *SGS(D, q)*. Les valeurs de $|SCM(g_i, q)|$, pour i =1, . . .,6, sont données dans le tableau 2 et les vecteurs de similarité entre graphes $SCG(g_i, q)$, pour i=1, . . .,6, sont résumés dans le tableau 3. Par application de (4), l'ensemble des graphes optimaux au sens de Pareto, i.e. le skyline de graphes par similarité, est donné par $SGS(D, q) = \{g_1, g_3, g_4, g_6\}$.

Il est aisé de voir que $g_2 \notin SGS(D, q)$ car il est dominé par g_4 et $g_5 \notin SGS(D, q)$ car il est dominé par g_1. Les graphes de D maximalement similaires à q sont g_1, g_3, g_4 et g_6. En effet,

Paire de graphes	$\|SCM(g_i, q)\|$
(g_1, q)	4
(g_2, q)	4
(g_3, q)	3
(g_4, q)	5
(g_5, q)	5
(g_6, q)	6

TAB. 2 – *Informations sur* $|SCM(g_i, q)|$.

	$Dist_{Ed}(g_i, q)$	$Dist_{SCM}(g_i, q)$	$Dist_{UG}(g_i, q)$
(g_1, q)	**4**	**0.33**	**0.50**
(g_2, q)	3	0.43	0.56
(g_3, q)	**2**	**0.50**	**0.67**
(g_4, q)	**3**	**0.38**	**0.44**
(g_5, q)	4	0.44	0.50
(g_6, q)	**4**	**0.40**	**0.40**

TAB. 3 – *Mesures de distance.*

- Le graphe g_1 est le plus intéressant au regard de la mesure $Dist_{SCM}$. Cela est dû aux raisons suivantes : i) g_1 satisfait un maximum de caractéristiques requises par q que d'autres graphes de même taille ; ii) g_1 et q sont de même taille. Mais, g_1 est le moins intéressant au regard des caractéristiques manquantes et superflues (i.e., $Dist_{Ed}$).
- Le graphe g_3 est le meilleur au regard de la mesure $Dist_{Ed}$. Cela signifie, d'une part, qu'il est le plus intéressant au regard du nombre de désaccords avec q. D'autre part, g_3 est beaucoup moins satisfaisant au regard des concordances avec q au sens de *SCM*.
- Le graphe g_6 est le plus intéressant au regard de la mesure $Dist_{UG}$. Cela est dû au fait que $g_6 \supset q$. Mais, c'est le moins intéressant au regard du critère basé sur les caractéristiques superflues (i.e., $Dist_{Ed}$).
- Le graphe g_4 peut être considéré comme un bon compromis entre les trois mesures $Dist_{Ed}$, $Dist_{SCM}$ et $Dist_{UG}$.

Examinons maintenant les résultats obtenus en utilisant seulement une seule mesure de similarité entre graphes. Si on s'intéresse aux *k* (= 3) meilleures réponses, g_2 est retourné comme graphe candidat en utilisant l'approche basée sur la distance d'édition. Tandis qu'avec l'approche basée sur le skyline, g_2 n'est pas retourné à l'utilisateur car g_4 est meilleur.

7 Raffinement du Skyline

Un des problèmes qui peut survenir lors du calcul de l'ensemble SGS (et d'un skyline en général) est sa taille qui est souvent très importante. D'un point de vue utilisateur, il est très souhaitable de disposer d'un critère pertinent permettant de sélectionner un sous-ensemble, de taille raisonnable, des graphes les plus intéressants parmi ceux du skyline SGS. Une solution à

ce problème est d'utiliser *le critère de diversité* (McSherry, 2002) pour sélectionner un sous-ensemble de graphes qui *est aussi divers que possible*, et ainsi fournir à l'utilisateur une image globale de l'ensemble des éléments de *SGS*.

Soit *S* un sous-ensemble de *SGS*. La diversité de *S* signifie que les graphes qu'il contient doivent être dissimilaires. L'objectif est d'extraire à partir de *SGS* un sous-ensemble $\mathbb{S}$ de taille *k* (*k* est un paramètre défini par l'utilisateur) avec *une diversité maximale*. En s'inspirant des travaux de Kukkonen et Lampinen (2007), l'approche proposée définit la diversité de *S* ($\subseteq SGS$) de taille *k* par un vecteur $Div(S) = (v_1, v_2, v_3)$ tel que

$$v_i = min\{Dist_i(g, g') | g, g' \in S\},$$

où, pour i=1, . . ., 3, $Dist_1 = Dist_{Ed}/(Dist_{Ed}+1)$, $Dist_2 = Dist_{SCM}$ et $Dist_3 = Dist_{UG}$. La valeur v_i exprime la diversité sur la $i^{ème}$ dimension du sous-ensemble *S*.

Afin d'identifier le sous-ensemble $\mathbb{S}$, nous considérons tous les sous-ensembles $S \subseteq SGS$ avec $|S| = k$ comme candidats et appliquons les étapes suivantes :

Étape 1. Pour chaque dimension *i* (i=1, . . ., 3), ordonner d'une manière décroissante tous les sous-ensembles candidats S selon leur diversité v_i. Soit $rang_i(S)$ le rang de *S* au regard de la $i^{ème}$ dimension. Rang de valeur *1* signifie la meilleure valeur de diversité et rang de valeur $C^k_{|SGS|}$ signifie la plus mauvaise valeur de diversité.

Etape 2. Évaluer un candidat *S* par : $val(S) = \sum_{i=1,...,3} rang_i(S)$.

Le sous-ensemble minimisant la somme de ses positions relativement à tous les rangs est considéré comme le sous-ensemble ayant la diversité maximale. Ainsi, $\mathbb{S}$ est caractérisé par

$val(\mathbb{S}) = min_S val(S)$, où $S \subseteq SGS$ et $|S| = k$.

Exemple 5. Reprenons l'exemple donné dans la section 6 où le skyline $SGS(D, q) = \{g_1, g_3, g_4, g_6\}$. Supposons maintenant que l'utilisateur est intéressé par les *k (=2)* meilleurs graphes au regard du critère de la diversité. On peut facilement vérifier que l'ensemble de tous les candidats contient 6 sous-ensembles de taille *k*, voir le tableau 4.

	v_1	v_2	v_3
$S_1 = \{g_1, g_3\}$	0.86	0.67	0.80
$S_2 = \{g_1, g_4\}$	0.83	0.50	0.60
$S_3 = \{g_1, g_6\}$	0.87	0.60	0.67
$S_4 = \{g_3, g_4\}$	0.80	0.62	0.73
$S_5 = \{g_3, g_6\}$	0.83	0.70	0.77
$S_6 = \{g_4, g_6\}$	0.75	0.50	0.61

TAB. 4 – *Candidats avec leur diversité.*

L'étape 1 et 2 conduisent aux résultats décrits dans le tableau 5 à partir duquel on peut voir que $val(S_1)$ est la valeur minimale. Ainsi, $\mathbb{S} = S_1 = \{g_1, g_3\}$.

8 Conclusion

Dans cet article, nous avons proposé une approche permettant la recherche de graphes par similarité. Le concept clé de cette approche est la notion de *skyline de graphes par similarité*.

	r_1	r_2	r_3	$Val(S_i) = \sum_{i=1,\cdots,3} r_i$
S_1	2	2	1	5
S_2	3	5	6	14
S_3	1	4	4	9
S_4	4	3	3	10
S_5	3	1	2	6
S_6	5	5	5	15

TAB. 5 – *Évaluation des Candidats ($r_i = rang_i$).*

Ce type de skyline permet l'extraction de tous les graphes de la base de données interrogée qui ne sont dominés par aucun autre graphe de la base au sens de la relation de dominance par similarité définie. Chaque graphe réponse est retourné à l'utilisateur avec un vecteur de scores montrant les différentes similarités correspondant aux différentes caractéristiques avec le graphe de la requête. Nous avons aussi montré comment sélectionner un sous-ensemble de diversité maximale à partir d'un skyline de graphes par similarité. Nous travaillons actuellement sur l'implémentation de l'approche pour démontrer son efficacité et sa pertinence.

Références

Borzsonyi, S., D. Kossmann, et K. Stocker (2001). The skyline operator. In *Proc. of ICDE*, pp. 421–430.

Bunke, H. (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern Recogn. Letters 18 (9)*, 689–697.

Bunke, H. et K. Shearer (1998). A graph distance metric based on the maximal common subgraph. *Pattern Recogn. Letters 19 (3-4)*, 255–259.

Cai, D., Z. Shao, X. He, X. Yan, et J. Han (2005). Community mining from multirelational networks. In *Proc. of PPKDD*, pp. 445–452.

Chen, C., X. Yan, P. S. Yu, J. Han, D.-Q. Zhang, et X. Gu (2007). *Towards Graph Containment Search and Indexing*. In *Proc. of VLDB*, Vienna, Austria, pp. 926–937.

Conte, D., P. Foggia, C. Sansone, et M. Vento (2004). Thirty years of graph matching in pattern recognition. *Inter. J. of Pattern Recogn. and Art. Intell. 18 (3)*, 265–298.

Hadjali, A., O. Pivert, et H. Prade (2010). Possibilistic contextual skylines with incomplete preferences. In *Proc. of SoCPaR, Paris, France*.

He, H. et A. K. Singh (2006). Closure-tree: An index structure for graph queries. In *Proc. of ICDE*, pp. 38–54.

Hu, H., Y. Hang, J. Han, et X. Zhou (2005). Mining coherent dense subgraphs across massive biological network for fonctional discovery. *Bioinformatics 1(1)*, 1–9.

Khalefa, M. E., M. F. Mokbel, et J. J. Levandoski (2008). Skyline query processing for incomplete data. In *Proc. of ICDE*, pp. 556–565.

Klinger, S. et J. Austin (2005). Chemical similarity searching using a neural graph matcher. In *Proc. of ESANN*, pp. 479–484.

Kukkonen, S. et J. Lampinen (2007). Ranking-dominance and many-objective optimization. In *IEEE Congress on Evolutionary Computation*, pp. 3983–3990.

McSherry, D. (2002). Diversity-conscious retrieval. In *Proc. of ECCBR*, pp. 219–233.

Pei, J., B. Jiang, X. Lin, et Y. Yuan (2007). Probabilistic skylines on uncertain data. In *Proc. of VLDB*, pp. 15–26.

Shang, H., K. Zhu, X. Lin, Y. Zhang, et R. Ichise (2010). Similarity search on supergraph containment. In *Proc. of ICDE*, pp. 637–648.

Tian, Y. et J. M. Patel (2008). Tale : A tool for approximate large graph matching. In *Proc. of ICDE, Cancun, Mexico*, pp. 963–972.

Wallis, W. D., P. Shoubridge, M. Kraetz, et D. Ray (2001). Graph distances using graph union. *Pattern Recogn. Letters 22 (6-7)*, 701–704.

Yan, X., P. S. Yu, et J. Han (2004). Graph indexing: A frequent structurebased approach. In *Proc. of0 ACM SIGMOD*, pp. 335–346.

Yan, X., P. S. Yu, et J. Han (2005). Substructure similarity search in graph databases. In *Proc. of ACM SIGMOD*, pp. 766–777.

Yiu, M. L. et N. Mamoulis (2007). Efficient processing of top-k dominating queries on multi-dimensional data. In *Proc. of VLDB*, pp. 483–494.

Zeng, Z., A. Tung, J. Wang, J. Feng, et L. Zhou (2009). Comparing stars: On approximating graph edit distance. In *Proc. of VLDB*, pp. 25–36.

Zhang, N., T. Özsu, I. Ilyas, et A. Aboulnaga (2006). Fix: Feature-based indexing technique for xml documents. In *Proc. of VLDB*, pp. 259–270.

Zhang, S., M. Hu, et J. Yang (2007). Treepi: A novel graph indexing method. In *Proc. of ICDE*, pp. 966–975.

Zhang, S., J. Z. Li, H. Gao, et Z. Zou (2009). A novel approach for efficient supergraph query processing on graph databases. In *Proc. of EDBT*, pp. 204–215.

Zou, L., L. Chen, M. T. Ozsu, et D. Zhao (2010). Dynamic skyline queries in large graphs. In *Proc. of DASFAA*, pp. 62–78.

Summary

One of the fundamental problems in graph databases is similarity search for graphs of interest. Existing approaches dealing with this problem rely on a single similarity measure between graph structures. In this paper, we suggest an approach allowing for searching similar graphs to a query graph where similarity between graphs is rather modelled by a vector of scalars than a unique scalar. To this end, we introduce the concept of *similarity skyline* of a query graph defined by the subset of graphs of the target database that are *the most similar* to the query in *a Pareto sense*. A diversity-based method for refining the retrieval result is proposed as well.

Adding Preferences to Semantic Process Model Matchmaking

Fernando Lemos*, Ahmed Gater*, Daniela Grigori*, Mokrane Bouzeghoub*

*Laboratoire PRiSM, Université de Versailles St-Quentin en Yvelines
firstname.lastname@prism.uvsq.fr,
http://prism.uvsq.fr

Abstract. The capability to easily find useful services satisfying non-functional constraints is critical for building service oriented systems. Current approaches for service retrieval with these kinds of constraints are mostly limited to atomic services. However, these approaches are not sufficient and do not fulfill user needs since many non-functional factors are hidden within the specification of service behavior. We argue that, in many situations, the service discovery process should be based on both behavior specification (that is, the process model that describes each composite service) and non-functional features of services. Moreover, taking into account user preferences allows ranking similar responses by functional and non-functional aspects. In this paper, we extend our semantic graph model for business process to consider non-functional aspects and preferences. We also propose matching techniques that allow delivery of inexact matches and evaluation of semantic distance between these matches and the user query. Consequently, even if a service satisfying exactly the user requirements does not exist, the most similar ones will be retrieved.

1 Introduction

Selection of an appropriate Web service for a particular task has become a difficult challenge due to the increasing number of Web services offering similar functionalities. These services may be offered, for example, at different QoS (quality of service) levels, which refers to various non-functional characteristics such as response time, throughput, availability and reliability.

In general, retrieving an appropriate web service for a particular task is done in two steps: *service discovery* and *service selection*. In the first step, web services that meet certain functional criteria are discovered. Then, these services are evaluated and ranked to identify the ones that better fulfill a set of non-functional properties requested by the user. Among user non-functional requests, it can be distinguished two kinds: those requests that *must* be satisfied and those ones that are *desired*, but not imposed.

Current approaches for service retrieval with non-functional characteristics consider only atomic services Wang et al. (2006); Mokhtar et al. (2008); Şora et al. (2010); D'Mello et al. (2008); Zhang et al. (2009); Kritikos and Plexousakis (2006). However, these approaches are

not sufficient and do not fulfill user needs as non-functional aspects can be *hidden within the specification of the service behavior*. We believe that, in some situations, the service discovery process should be based on both behavior specification and non-functional aspects of services. We present two such examples.

- *Web services integration*. Consider a company that uses a service S for one of its applications and wants to replace S with an equivalent service. Thus the company will search in a service registry for a service having the same business protocol as service S. Moreover, if the existing application requires some non-functional requirements for individual operations of service S, or for all the service, these constraints have to be fulfilled by the retrieved service. Thus, we need a method to discover services given a behavior model and its non-functional requirements.
- *Retrieving scientific workflows*. Scientists have a great need for process-based discovery method. Given a workflow, users are interested in finding similar workflows (published by them or by other users). Moreover, they could be interested to find, among these similar workflows, the ones that satisfies some global or local quality requirements. For instance, the workflow that takes the shorter time or which guarantees a given correctness for the results of a specific activity.

These scenarios show the need for a discovery approach taking into account both process models and non-functional requirements. Thus, the technical challenges for such a discovery approach for this kind of applications are the following:

- defining a model allowing to specify both global and local non-functional requirements and differentiate between required factors and preferred ones;
- allowing an inexact match, i.e., defining a similarity measure taking into account structural differences (sequencing order differences) and non-functional satisfaction degree;
- combining the behavior-based matching algorithm and non-functional factors matching. While, for atomic services, the discovery process applies first functional criteria and then non-functional factors, in our case, the behavioral matching and non-functional matching should be interwoven because required non-functional features can be specified at a fine grain level, at the operation level;
- dynamically recalculating some global non-functional factors that depends on the process structure. If we consider the possibility of a partial matching, i.e., returning only a path that match with the required protocol, then some global non-functional factors advertised for the service should be recalculated (for instance, the execution time).

In this paper, we propose a method for service discovery and selection taking into account process model and preferences. We suppose that user gives as input a process model annotated with non-functional factors (local and global, required and preferred) and our method retrieves, in a repository, the services respecting all (or most of) user constraints and preferences. For this, we propose a formal model to annotate process models and their activities with non-functional properties and to annotate user queries and their activities with preferences over the non-functional properties. We also present the algorithms to evaluate the preferences both in service discovery and service selection tasks.

Section (2) presents our model to annotate process models and user queries. Section (3) presents an algorithm to add preference evaluation in a service matchmaking algorithm. Section (4) presents a technique to use preferences in the service selection task. Finally, Section (5) presents the conclusions.

	Response Time
sequence of nodes $s = \{n_1, \ldots, n_k\}$	$respTime\,(s) = \sum_{i=1}^{k} respTime\,(n_i)$
connector node $\wedge$ *with branches* $\{b_1, \ldots, b_k\}$	$respTime\,(\wedge) = \max_{i=1}^{k} respTime\,(b_i)$
connector node $\vee$ *with branches* $\{b_1, \ldots, b_k\}$	$respTime\,(\wedge) = \max_{i=1}^{k} respTime\,(b_i)$
activity node n	The value of attribute *respTime*

TAB. 1 – *Response Time Aggregation Function.*

2 Adding Preferences to Business Process Specification

In this paper, we use preferences as one important aspect to select and rank business processes, besides the structural aspect. We consider preferences defined over non-functional properties described in the business process specification. We illustrate our approach by using QoS attributes as the non-functional properties of the business process.

In Section 1, we specified a service to be described by a process model. A process model consists of a set of related primitive activities that are combined using usual control flow structures (Sequence, Flow, If-Then-Else, Choice, Pick, Any-order and Repeat) to construct complex processes. Many languages are available to describe process models, as WSCL Banerji et al. (2002), BPEL4WS Andrews et al. (2003), OWL-S Martin et al. (2004), etc. In this section, we aim to abstract as much as possible from a specific description language by introducing a graph-based model capable of capturing major control flow structures and non-functional aspects. Moreover, we extend the abstract model to feature preference specification.

Adding QoS Information to Process Models QoS information is added as graph annotations. User can specify QoS attributes that characterize the process model as a whole (*global annotations*) or specify QoS attributes for specific activities (*activity annotation*s). Each annotation is a pair (m, r), where m is a QoS attribute and r is a pertinent value [1]. Figure 1 shows an example of a process model annotated with QoS attributes. The example presents a global annotation indicating the cost of the process model and several activity annotations indicating the response time and reliability of some activities.

User can also define aggregation functions to automatic calculate global QoS information from activity annotations. For example, consider the aggregation function in Table 1, which determines the response time of the whole process model from the response time of its component activities. In this case, the response time of the process model in Figure 1 is 90ms.

Adding Preferences to User Queries User queries are represented as graphs annotated with hard/soft preferences. Hard preferences are constraints that must be satisfied by the target graph, while soft preferences are those whose satisfaction is optional, although a better satisfaction means a more desirable graph. User can specify preferences for the process model as a whole (*global preferences*) or specify preferences for specific activities (*activity preferences*). Hard preferences are specified as relational expressions and soft preferences are specified using the preference constructors defined in Kießling (2002).

1. We abstract from the different units in which a value is described.

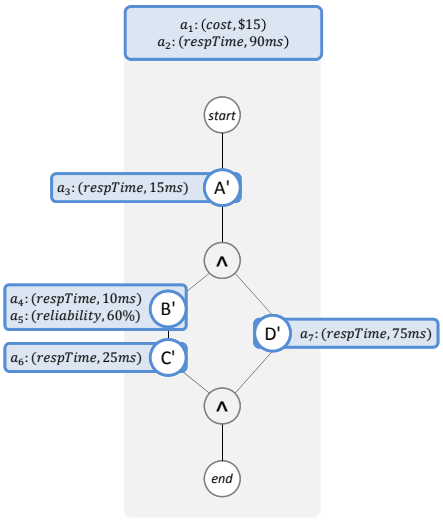


FIG. 1 – *Target Graph* G_T.

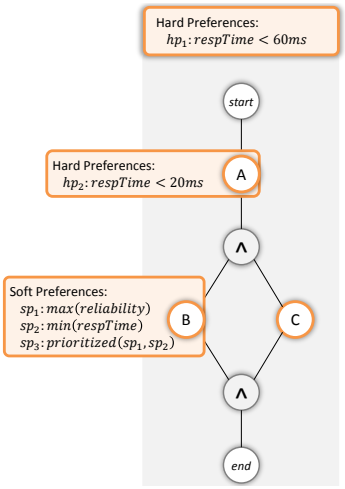


FIG. 2 – *Query Graph* G_{T_1}

Figure 2 shows an example of a user query annotated with hard and soft constraints. The example presents a global hard preference indicating the response time must be less than 60ms. The example also presents a hard preference for activity A and some soft preferences for activity B.

Semantic Graph Model for Business Process Annotated with Preferences The formal description of a process model annotated with QoS information and preferences is given below. Our model is an extension over a lighten version of the process model formal description given by Ahmed et al. (2009).

In what follows, let m be a QoS attribute.

Definition 1 *An **annotation** is a pair* (m, r)*, where* r *is a value for* m*.*

Definition 2 *A **preference** is an expression that represents a desire of the user over the QoS attributes of a process model or activity. It can be of two types:*

- *A **hard preference** is a relational expression of the form* (m, o, r)*, where* $o \in \{<, >, \leq, \geq, =, \neq\}$ *is a relational operator and* r *is a value for* m*.*
- *A **soft preference** is an expression of one the following forms* [2]*:*
  - *atomic soft preferences:*
    - $around\,(m, r_{desired})$*: for attribute* m*, this expression favors the value* $r_{desired}$*; otherwise, it favors the values close to* $r_{desired}$*;*
    - $between\,(m, r_{low}, r_{up})$*: for attribute* m*, this expression favors the values inside the interval* $[r_{low}, r_{up}]$*; otherwise, it favors the values close to the limits;*

2. Based on the preference constructors defined in Kießling (2002).

  - $max\,(m)$*: for attribute* m*, this expression favors for the highest value; otherwise, the closest value to the maximum is favored;*
  - $min\,(m)$*: for attribute* m*, this expression favors for the lowest value; otherwise, the closest value to the minimum is favored;*
  - $likes\,(m, r_{desired})$*: for attribute* m*, this expression favors the value* $r_{desired}$*; otherwise, any other value is accepted;*
  - $deslikes\,(m, r_{undesired})$*: for attribute* m*, this expression favors the values that are not equal to* $r_{undesired}$*; otherwise,* $r_{undesired}$ *is accepted;*
- *complex soft preferences:*
  - $pareto\,(p_i, p_j)$*: this expression states that the two soft preference expressions* p_i *and* p_j *are equally important;*
  - $prioritized\,(p_i, p_j)$*: this expression states that the soft preference expression* p_i *is more important than the soft preference expression* p_j*.*

Definition 3 ***Semantic Process Model Graph*** *is a tuple* $SPMG = (A, C, E, O, \varphi, S_a, S_p)$*, where:*
- A *is the set of activity descriptor nodes;*
- C *is the set of connector nodes;*
- $E \subseteq ((A \cup C) \times (A \cup C))$ *is the set of edges;*
- O *is the set of ontological concepts;*
- φ *is a set of aggregation functions;*
- S_a *is a set of annotations;*
- S_p *is a set of preference expressions.*

Definition 4 ***Activity Node Descriptor*** *is a tuple* $AND = (N, In, Out, \rho, S_a, S_p)$*, where:*
- N *is the name of activity;*
- In *is the set of input attributes;*
- Out *is the set of output attributes;*
- $\rho : (In \cup Out) \rightarrow O$ *is the function that maps input/output attributes to ontological concepts;*
- S_a *is a set of annotations;*
- S_p *is a set of preference expressions.*

In the case of a target process model or activity, the set S_p is empty and the set S_a contains the QoS annotations characterizing the model. In the case of a query process model or activity, the set S_a is empty and the set S_p contains the user preferences.

3 Extending Process Model Matchmaking with Preferences

In this section, we present an algorithm to verify user hard preferences in the process model matchmaking. The algorithm is composed of two phases: (i) Matching between activity hard preferences and activity annotations; (ii) Matching between global hard preferences and global annotations.

Matching between Activity Hard Preferences and Activity Annotations A first step necessary to verify the hard preferences of a query activity is to discover the target activity that

semantically corresponds to it, i.e., to find a structural mapping between query and target process models.

In Ahmed et al. (2009), Error-Correcting Subgraph Isomorphism (ECSI) algorithm Messmer (1995) was adapted to discover the best mapping between two process models represented as graphs. The sub-graph isomorphism detection is based on a state-space searching using an algorithm similar to A* Bunke (2000). The basic idea of a state-space search is to have states representing partial solutions of the given problem and to define transitions from one state to another. Thus, the latter state represents a more complete solution than the previous state. For each state s there is an evaluation function $f(s)$ that describes the quality of the represented solution. In the case of sub-graph isomorphism detection, given a target graph G_Q and a query graph G_T, a state s in the search space represents a partial mapping from G_Q to G_T.

More precisely, the algorithm starts by mapping the first activity node of G_Q with all the activity nodes of $G_T \cup \{\$\}$ (symbol $\$$ denotes deleting activity nodes). This represents a partial mapping that will be extended by adding one activity node at a time. The process terminates when either a state representing an optimal error-correct-subgraph isomorphism from G_Q to G_T has been reached or all states in the search space have costs that exceed a given acceptance threshold.

In Ahmed et al. (2009), a pruning technique was used to reduce the search space since non-promising mappings can be early identified in the search. The pruning technique presented in Ahmed et al. (2009) considers two metrics to prune the search space tree:

- *Semantic Similarity* $SS(v, w)$, which checks if a semantic similarity holds between two activities v and w. This metric considers the activity label, its inputs and outputs;
- *Cost* $C(M)$, which computes the cost of the mapping M. Mappings trespassing a defined threshold t are considered non-promising and they are discarted..

We propose to extend this technique by verifying unacceptable mappings according to the user hard preferences. We propose a metric called *Hard Preference Satisfaction* $\theta(v, w)$ which checks whether activity w satisfies the hard preferences specified for activity v. In the following functions, $v.S_{hp}$ stands for the hard preferences specified in $v.S_p$.

Hard preference satisfaction function between two activities v and w:

$$\theta(v, w) = \begin{cases} \text{true}, & v.S_{hp} = \emptyset \\ \theta(v.S_{hp}, w.S_a), & v.S_{hp} \neq \emptyset \wedge w \neq \$ \\ \text{false}, & \text{otherwise} \end{cases} \quad (1)$$

Satisfaction function between a set S_{hp} of hard preferences and a set S_a of offers:

$$\theta(S_{hp}, S_a) = \bigwedge_{e_a \in S_a} \theta(e_{hp}, e_a) \,|\, e_{hp} \in S_{hp} \wedge e_a.m = e_a.m \quad (2)$$

Satisfaction function between a hard preferences expression e_{hp} and an offer expression e_a:

$$\theta(e_{hp}, e_a) = \begin{cases} \text{true}, & e_a \text{ satisfies } e_{hp} \\ \text{false}, & \text{otherwise} \end{cases} \quad (3)$$

To verify if that e_a satisfies e_{hp} is trivial and will not be discussed in this paper.

Hard Preference Satisfaction metric is checked after the Semantic Similarity metric and before the Cost metric. Figure 3 represents the matching process between two process models

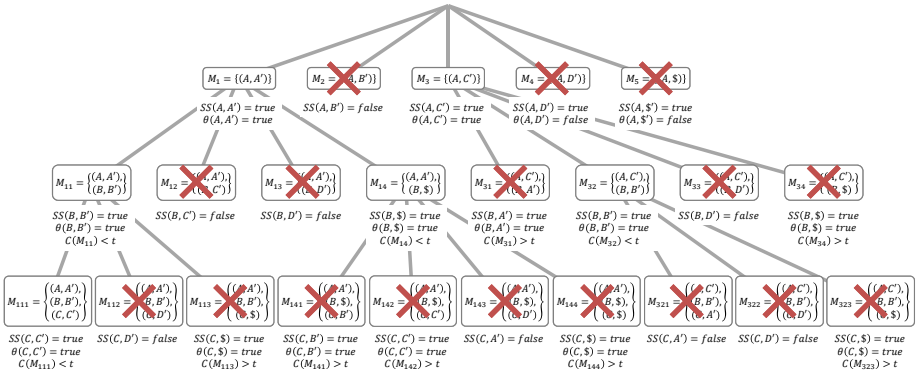


FIG. 3 – *Matching between query G_Q and target G_{T_1}.*

G_Q and G_{T_1}. Mappings M_2, M_{12}, M_{13}, M_{33}, M_{112}, M_{143}, M_{321} and M_{332} are considered non-promising according to the semantic similarity metric. Mappings M_4 and M_5 are considered unacceptable according to the hard preference satisfaction metric. Mappings M_{31}, M_{34}, M_{113}, M_{141}, M_{142}, M_{144} and M_{323} are considered non-promising according to the cost metric. In this example, matching M_{111} is the best mapping.

Matching between Global Hard Preferences and Global Annotations Once a target satisfying all activity preferences is discovered, it is subject to the global hard preferences matching. A simple approach would verify the query global preferences against the target global annotations. However, based on the structural mapping discovered between query and target, some target global annotations can change.

Consider, for example, the matching presented in Figure 4. The response time of target graph G_T is annotated as 90ms. However, considering the mapping between the query graph G_Q and G_{T_1}, the dotted trace containing activity D' will never be consumed. Thus, recalculating the response time of the target graph ignoring activity D' results in a global response time of 50ms. According to the global hard preference of the query, if the recalculation had not been done, the target G_{T_1} would be discarded.

Once the global attributes are recalculated, we proceed to the proper matching between global preferences and global annotations. Again we use the hard preference satisfaction metric to filter the targets. In this case, the target G_{T_1} is considered as a positive response.

4 Process Model Selection using Preferences

The cost of the mapping M between query and target implies a structural similarity $SS(M)$ [3] that can be used to define a total order between targets. However, two targets having the same structural similarity cannot be distinguished. Moreover, targets very similar to the query and better satisfying the user preferences should appear among the firsts in the ranking. In this case, we propose to use the soft preferences defined in the query to calculate the satisfiability

3. $SS(M)$ can be calculated as $SS(M) = \frac{1}{1+C(M)}$.

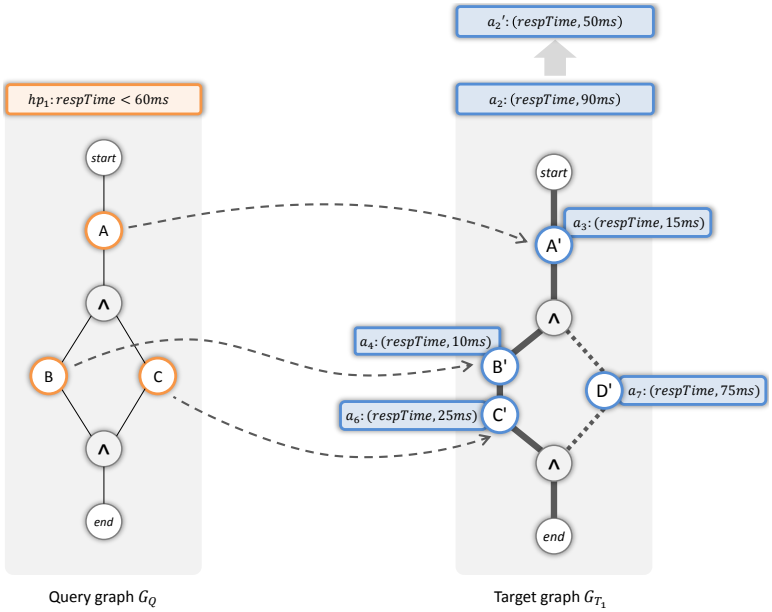


FIG. 4 – *Recalculation of a target global annotation based on the graph matchmaking result.*

degree between query and target and aggregated it with the structural similarity to provide a ranking metric that takes into account structure and preferences.

Satisfiability degree (δ) measures how well the annotations of a target process model or activity satisfy the soft preferences of a user query or activity, respectively. Given a set S_{sp} of soft preferences and a set S_a of annotations, $\delta\,(S_{sp}, S_a)$ measures how well S_a satisfies S_{sp}. The calculus is done in two phases: (i) for atomic soft preferences, we calculate the satisfiability degree between each of them and a pertinent annotation; (ii) for complex preferences, we aggregate the satisfiability degree of the atomic preferences.

Satisfiability Degree for Atomic Preferences Given an atomic preference p and an annotation $a = (m, r)$, the satisfiability degree $\delta\,(p, a)$ between them is given by the equation (4). The *Satisfiability Distance* $d\,(p, a)$ measures how far is the value r in annotation a from what value prefers the preference p. The satisfiability distance depends on the preference type of p and it is described in Table 2.

$$\delta\,(p, a) = \frac{1}{1 + d\,(p, a)} \tag{4}$$

Suppose an attribute m over which an atomic preference p is specified by activity v and an annotation a is specified by activity w. Consider r the value assigned by a to m. Table 2 presents the *satisfiability distance* of m according to the type of p.

In our example, consider the set of soft preferences $B.S_{sp}$ of query activity B and the set of annotations $B'.S_a$ of target activity B'. First, we calculate the satisfiability distances between the atomic soft preferences sp_1 and sp_2 in $B.S_{sp}$ and their corresponding annotations

Preference p over Metric m	*Satisfiability Distance $d(p,a)$*
$around\,(m, r_{desired})$	$d\,(p,a) = \lvert r - r_{desired}\rvert$
$between\,(m, r_{low}, r_{up})$	$d\,(p,a) = \begin{cases} 0, & r \in [low, up] \\ low - r, & r < low \\ r - up, & r > up \end{cases}$
$max\,(m)$	$d\,(p,a) = r_{max} - r$, where r_{max} is the highest value
$min\,(m)$	$d\,(p,a) = r - r_{min}$, where r_{min} is the lowest value
$likes\,(m, r_{desired})$	$d\,(p,a) = \begin{cases} 0, & r = r_{desired} \\ \mu, & \text{otherwise} \end{cases}$, where μ is a weight
$deslikes\,(m, r_{undesired})$	$d_m = \begin{cases} 0, & r \neq r_{undesired} \\ \mu, & \text{otherwise} \end{cases}$, where μ is a weight

TAB. 2 – *Satisfiability distance of a preference p according to an annotation a.*

Soft Preference	*Satisfiability Distance*	*Satisfiability Degree*
sp_1	$d\,(sp_1, a_5) = 40$	$\delta\,(sp_1, a_5) = 0.024$
sp_2	$d\,(sp_2, a_4) = 10$	$\delta\,(sp_2, a_4) = 0.09$

TAB. 3 – *Satisfiability distances and degrees of atomic soft preferences of query activity B.*

in $B'.S_a$. Table 3 presents the satisfiability distances and degrees of atomic soft preferences of query activity B.

Satisfiability Degree for Complex Preferences First, we construct a *preference tree* t_{sp} to represent to the complex preferences of S_{sp}. In a preference tree, the nodes represent atomic preferences and the edges represent a *more important than* relation (*prioritized* preference) from parent to child. Preferences belonging to the same level and having the same parent means that a *pareto* preference stands for them. Each level l_i of the tree has a weight μ_{l_i} and $\mu_{l_i} > \mu_{l_j}$ iff $j > i$.

Let h be the height of t_{sp}. The satisfiability degree $\delta\,(S_{sp}, S_a)$ between S_a and S_{sp} is given by:

$$\delta\,(S_{sp}, S_a) = \frac{\sum_{i=1}^{h} \left(\mu_{l_i} \times \sum_{p \in l_i} \delta\,(p,a)\right)}{\sum_{i=1}^{h} \sum_{p \in l_i} \mu_{l_i}} \tag{5}$$

where a in $\delta\,(p,a)$ is the annotation that corresponds to the QoS attribute defined in preference p and $p \in l_i$ means that preference p belongs to level l_i. As we can see, this function is a weighted average of the satisfiability degree of each atomic preference. The weight assignment of each level can be defined by the user or automatically according to the height of the tree.

For our example, in the preference tree t_{sp} generated according to complex preference sp_3 in $B.S_{sp}$, the atomic preference sp_1 (level l_1) is the parent of the atomic preference sp_2 (level $l_{2.}$). If we assign weight 2 to l_1 and weight 1 to l_2, we have $\delta\,(B.S_{sp}, B'.S_a) = 0.046$.

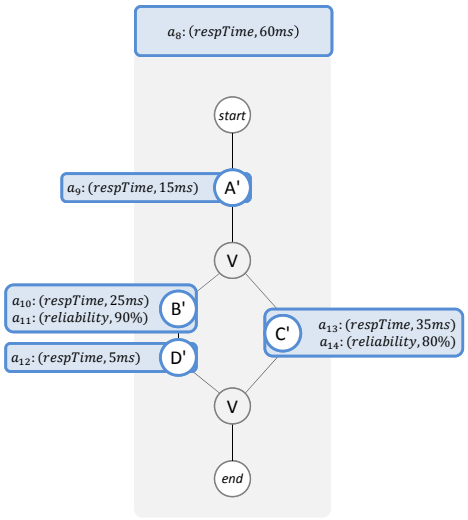


FIG. 5 – *Target graph* G_{T_2}.

Satisfiability Degree Between Graphs The satisfiability degree between two graphs G_Q and G_T according to a mapping M between them is given by:

$$\delta\left(G_Q, G_T, M\right) = \frac{\delta\left(G_Q.S_{sp}, G_T.S_a\right) + \sum_{(v,w)\in M} \delta\left(v.S_{sp}, w.S_a\right)}{|M^*| + 1} \tag{6}$$

where $M^* = \{(v,w) \,|\, (v,w \in M) \wedge v.S_{sp} \neq \emptyset\}$.

In our example, as we have only one activity defining soft preferences, then the satisfiability degree between G_Q and G_{T_1} is equal to satisfiability degree between activity B and B': $\delta\left(G_Q, G_{T_1}, M_{111}\right) = \delta\left(B.S_{sp}, B'.S_a\right) = 0.046$.

Ranking Position of a Target Graph The position of a target G_T in the ranking according to its structural similarity $SS\left(M\right)$ and satisfiability degree $\delta\left(G_Q, G_T, M\right)$ is given by:

$$rank\left(G_T\right) = \mu_{SS} \times SS\left(M\right) + \left(1 - \mu_{SS}\right) \times \delta\left(G_Q, G_T, M\right) \tag{7}$$

where $0 < \mu_{SS} < 1$ is a weight assigned to the semantic similarity metric.

Consider the target graph G_{T_2} presented in Figure (5). Consider the mapping $M_{G_Q,G_{T_2}} = \{(A,A'), (B,B'), (C,C')\}$ as the best mapping between query G_Q and target G_{T_2}. The satisfiability degree between G_Q and target G_{T_2} according to $M_{G_Q,G_{T_2}}$ is given by $\delta\left(G_Q, G_{T_2}, M_{G_Q,G_{T_2}}\right) = 0.073$. In this case, if we suppose both targets have the same structural similarity, then we can say that target G_{T_2} is better than target G_{T_1}.

5 Conclusions

In this paper, we presented a method for service discovery and selection taking into account process model and preference requirements. First, we proposed a formal model to annotate

process models and their activities with non-functional properties and to annotate user queries and their activities with preferences over the non-functional properties. Then, we presented an approach to evaluate user preferences in a service matchmaking algorithm. In this case, we presented how user preferences can help in pruning the space search and how the matching between two process models can change the non-functional global annotations of a service. Finally, we presented a method to consider user preferences in the service selection task.

As future works, we pretend to study the impact of different types of preference evaluations in the service matchmaking. We also pretend to study different types of preferences, e.g., structural preferences. As the satisfiability functions presented are based on weigthed averages, we also pretend to define other types of aggregation to provide different interpretations to the merging of structural and satisfiability similarities.

Acknowledgments

This work has received support from the French National Agency for Research (ANR) on the reference ANR- 08-CORD-009.

References

Ahmed, G., D. Grigori, and M. Bouzeghoub (2009). Ranking and similarity evaluation of owl-s process models. Technical report, PRiSM Laboratory, University of Versailles.

Andrews, T., F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana (2003). *BPEL4WS, Business Process Execution Language for Web Services Version 1.1*. IBM.

Banerji, A., C. Bartolini, D. Beringer, V. Chopella, and Et (2002). Web services conversation language (wscl) 1.0. Technical report.

Bunke, H. (2000). Recent developments in graph matching. In *Proc. of 15th Int. Conf. on Pattern Recognition*, pp. 117 – 124.

Şora, I., G. Lazăr, and S. Lung (2010). Mapping a fuzzy logic approach for qos-aware service selection on current web service standards. In *IEEE International Joint Conference on Computational Cybernetics and Technical Informatics (ICCC-CONTI 2010)*, pp. 553 – 558.

D'Mello, D. A., I. Kaur, N. Ram, and A. V.S. (2008). Semantic web service selection based on business offering. In *Proceedings of the 2008 Second UKSIM European Symposium on Computer Modeling and Simulation (EMS '08)*, Washington, DC, USA, pp. 476–481. IEEE Computer Society.

Kießling, W. (2002). Foundations of preferences in database systems. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pp. 311–322. VLDB Endowment.

Kritikos, K. and D. Plexousakis (2006). Semantic qos metric matching. In *Proceedings of the European Conference on Web Services (ECOWS'06)*, Washington, DC, USA, pp. 265–274. IEEE Computer Society.

Martin, D., M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. R. Payne, E. Sirin, N. Srinivasan, and K. Sycara (2004). Owl-s: Semantic markup for web services.

Messmer, B. (1995). *Graph Matching Algorithms and Applications*. Ph. D. thesis, University of Bern.

Mokhtar, S. B., D. Preuveneers, N. Georgantas, V. Issarny, and Y. Berbers (2008). Easy: Efficient semantic service discovery in pervasive computing environments with qos and context support. *Journal of Systems and Software 81*(5), 785–808. Software Process and Product Measurement.

Wang, X., T. Vitvar, M. Kerrigan, and I. Toma (2006). A qos-aware selection model for semantic web services. In *4th International Conference on Service Oriented Computing (ICSOC)*, pp. 390–401. Springer.

Zhang, Y., H. Huang, D. Yang, H. Zhang, H.-C. Chao, and Y.-M. Huang (2009). Bring qos to p2p-based semantic service discovery for the universal network. *Personal Ubiquitous Computing 13*(7), 471–477.

Résumé

La capacité de trouver facilement des services utiles satisfaisant des contraintes non-fonctionnelles est essentielle pour le renforcement des systèmes orientées service. Les approches actuelles pour la recherche de services avec ces sortes de contraintes sont essentiellement limités aux services atomique. Toutefois, ces approches ne sont pas suffisants et ne répondent pas aux besoins des utilisateurs une fois que de nombreux facteurs non-fonctionnels sont cachés dans la spécification du comportement du service. Nous soutenons que, dans de nombreuses situations, le processus de découverte de services devrait être basé sur les deux spécifications de comportement (qui est, le modèle de processus qui décrit chaque service composite) et des caractéristiques non-fonctionnelle des services. En outre, préférences de l'utilisateur permetent de classer les réponses similaires par aspects fonctionnels et non fonctionnels. Dans cet article, nous étendons notre modèle de graphe sémantique défini pour les processus métier pour examiner les aspects non-fonctionnels et les préférences. Nous proposons aussi des techniques d'appariement qui permettent la prestation des matches inexacte et d'évaluation de la distance sémantique entre ces matches et la requête de l'utilisateur. Par conséquent, même si un service répondant exactement aux exigences d'utilisateur n'existe pas, les plus similaires seront récupérées.

Une approche d'alignement sémantique de concepts complexes d'ontologies

Myriam Lamolle*, Chan Le Duc*
Rim Touhami*,**

*LIASD - Université Paris8/IUT de Montreuil
140 rue de la Nouvelle France 93100 Montreuil
m.lamolle, c.leduc@iut.univ-paris8.fr,
http://www.iut.univ-paris8.fr
**INRA, unité mét@risk
AgroParisTech, 16 rue Claude Bernard 75231 PARIS CEDEX 05
rim.touhami@agroparistech.fr

Résumé. Le but du Web sémantique est de transformer le Web en un système ayant un contenu compréhensible aussi bien par des hommes que par des machines ou des agents informatiques. Aussi, la sémantique du contenu des ressources du Web doit être rendue explicite dans une représentation formelle ontologique et standardisée. Cependant, avec la prolifération d'ontologies accessibles par le Web, bien souvent modélisées différemment pour un même domaine, il devient nécessaire de fournir un processus d'alignement pour favoriser leur interopérabilité. Bien peu de travaux sur l'alignement d'ontologies proposent des solutions pour générer des correspondances entre concepts complexes. Nous présentons dans cet article une méthode d'alignement permettant de détecter à la fois des correspondances simples et complexes. Elle consiste, dans un premier temps, à générer les correspondances simples selon trois techniques ; puis, dans un deuxième temps, à générer des correspondances complexes à partir de graphes représentant les ontologies décrites en OWL-DL.

1 Introduction

Il n'est plus à démontrer qu'un domaine du monde réel peut avoir de multiples modélisations dans divers langages de représentations. Dans le contexte du web, ces différentes représentations sont de plus en plus souvent des ontologies. La terminologie choisie pour modéliser une ontologie dépend du point de vue et de la connaissance du domaine que l'on en a. Il est possible de décrire un même domaine d'application en utilisant des ontologies ayant un vocabulaire différent et en utilisant des niveaux de granularité différents. Ces modèles ontologiques sont liés à des systèmes formels qui permettent d'induire à partir des connaissances acquises de nouvelles connaissances. Pour favoriser l'interopérabilité des applications qui s'appuient sur ces ontologies, l'hétérogénéité entre les connaissances exprimées au sein de chacune d'entre elles doit être réduite. Pour atteindre cet objectif, il est nécessaire d'établir des

correspondances sémantiques entre les différentes entités appartenant à deux ontologies différentes ; c'est le but de l'alignement d'ontologies (Euzenat et Shvaiko, 2007). Ainsi, l'alignement permet-il de produire un ensemble de correspondances entre des paires d'entités de deux ontologies. Ces paires d'entités sont liées par une relation sémantique (équivalence, subsomption, incompatibilité, etc.). De plus, une mesure de confiance peut être associée à chaque correspondance. L'ensemble des correspondances (ou alignement) peut alors être utilisé pour fusionner des ontologies, migrer des données entre ontologies ou traduire des requêtes formulées à partir d'une ontologie vers une autre.

Dans la littérature, plusieurs méthodes d'alignement ont été proposées. Certaines s'appuient sur la comparaison des expressions linguistiques (Euzenat, 2004), d'autres sur la comparaison des instances et des annotations (David et al., 2007) ; enfin, il y a des méthodes qui tiennent compte de la structure (Djoufak Kengue et al., 2008), (Zghal et al., 2009), (Bach, 2006), etc. Ces méthodes, appelées méthodes d'alignement simple, sont les plus répandues à l'heure actuelle. Elles permettent la détection des correspondances simples entre entités atomiques (ou concepts simples) (i.e $Human \sqsubseteq Person, Female \sqsubseteq Person$). Plusieurs types d'hétérogénéité sémantique entre les différentes ontologies peuvent être résolus en utilisant ces méthodes d'alignement classiques. Cependant, des correspondances nécessitent des techniques d'appariement plus évoluées afin d'aligner des entités simples avec des entités complexes ou des entités complexes avec d'autres entités complexes. Il nous paraît intéressant de détecter ce genre de correspondances pertinentes où un concept atomique est aligné avec une formule représentant un concept complexe. Par exemple, le concept *Parent* peut être défini par la notion *"personne ayant au moins un enfant"*. Dans cet exemple, aligner le concept simple *Parent* avec la formule qui représente une personne ayant au moins un enfant (i.e. $Parent \sqsubseteq Personne \sqcap \geqslant 1\ AvoirEnfant$) est plus riche sémantiquement qu'aligner simplement le concept *Parent* avec le concept *Personne* (i.e. $Parent \sqsubseteq Personne$). Un autre point important à souligner est que la génération d'un alignement complexe a une certaine répercussion lors de la vérification de la cohérence du système. En effet, un raisonneur, tel que Pellet (Sirin et al., 2007) ou Fact++ (Tsarkov et Horrocks, 2006), lancé sur un système constitué de deux ontologies O_1 et O_2 et d'un alignement simple *As*, peut répondre que le système est non cohérent. Or, ce raisonneur, sur les mêmes ontologies O_1 et O_2 avec un alignement complexe *Ac*, peut déduire que le système est cohérent.

En conséquence, de nouveaux travaux se tournent vers des solutions d'alignements complexes telles que (Ritze et al., 2009). Mais ces travaux, à l'heure actuelle, ne concernent au mieux que l'alignement d'un concept simple avec un concept complexe. La méthode que nous proposons dans cet article permet de détecter à la fois des correspondances simples et complexes. Pour cela, nous procédons en plusieurs étapes. La première étape consiste à enrichir les deux ontologies à aligner en utilisant le raisonneur IDDL (Zimmermann et Le Duc, 2008). Ce raisonneur permet d'ajouter aux ontologies des relations de subsomptions qui sont implicites. La deuxième étape est la détection des correspondances simples, effectuée en combinant efficacement trois aligneurs classiques. La dernière étape consiste à détecter les correspondances complexes, l'idée, ici, est inspirée des méthodes d'alignement simple qui sont fondées sur les graphes (Djoufak Kengue et al., 2008),(Zghal et al., 2009). Puisqu'à partir d'un vocabulaire fini, un nombre infini de formules peut être construit, il est impossible de savoir quelles sont les formules pertinentes à aligner. Une solution possible est d'essayer de capturer la sémantique OWL et de représenter les constructeurs (par exemple, subsomptions, disjonctions,

restrictions de cardinalité) sous forme de graphe. Puis, à partir des graphes correspondants aux deux ontologies à aligner, il s'agit de chercher des sous-graphes pertinents pouvant être alignés en prenant en considération leurs structures et en s'appuyant sur une mesure de similarité terminologique.

Cet article est organisé comme suit. Après avoir présenté le raisonneur IDDL que nous utilisons pour nos expérimentations, nous proposons une nouvelle méthode d'alignement d'ontologies par leur transformation en graphe. Nous détaillons ensuite l'alignement des objets complexes de ces ontologies et un exemple montre l'application des algorithmes proposés. Enfin, nous concluons et donnons quelques perspectives de ce travail.

2 Le raisonneur IDDL

Raisonner sur des ontologies est essentiel pour inférer de nouvelles connaissances. Les raisonneurs sont utilisés pour contrôler la consistance des ontologies, vérifier si certaines classes sont insatisfaisables et gérer la hiérarchie des classes et des relations. Ce sont les propriétés concernant les concepts représentés et les relations entre ces concepts qui permettent aux raisonneurs tels que Pellet (Sirin et al., 2007), FaCT++ (Tsarkov et Horrocks, 2006) d'inférer de nouvelles connaissances (notamment par la notion de subsumption).

Le raisonneur IDDL (Integrated Distributed Description Logics) est un nouveau formalisme pour représenter un ensemble d'ontologies et leurs alignements, i.e. des ontologies en réseau interconnectées par des alignements. IDDL se démarque des autres formalismes par :

1. Dans IDDL, les alignements sont considérés comme des connaissances indépendantes des connaissances provenant de l'ontologie. Aussi, de nouvelles conséquences sémantiques peuvent apparaître quand une connaissance est propagée par des alignements à partir des ontologies locales.
2. IDDL ne fait aucune hypothèse sur l'expressivité des formalismes utilisés dans une ontologie locale hormis la décidabilité. Ceci permet une hétérogénéité des mécanismes et des formalismes de raisonnement utilisés dans les ontologies locales. Par exemple, un raisonneur local utilise un algorithme basé tableau alors qu'un autre peut rechercher la consistance par un algorithme basé automate.
3. IDDL supporte un vrai raisonnement distribué ; i.e. tous les raisonnements sur les ontologies locales peuvent être exécutés indépendamment.

Le raisonneur IDDL implémente l'algorithme présenté dans (Zimmermann et Le Duc, 2008) et permet de faire un raisonnement distribué en adoptant deux types de sémantiques :

1. La sémantique IDDL qui utilise des primitives spécifiques permettant de (i) manipuler des ontologies et des alignements (i.e. le réseau des ontologies constitué des ontologies locales avec leurs alignements), (ii) vérifier la consistance globale ou locale du réseau ;
2. La sémantique DL (logique des descriptions classique) pour intégrer toutes les ontologies et les alignements et former ainsi une ontologie unique. IDDL permet ensuite de faire un raisonnement classique sur cette ontologie unique.

Lors de la première étape de notre méthode d'alignement, nous lançons ce raisonneur afin d'enrichir les deux ontologies à aligner en ajoutant des relations de subsomptions qui sont implicites et qui n'existaient pas dans les ontologies sources. En effet, ces nouveaux axiomes peuvent permettre l'augmentation du nombre de correspondances détectées.

3 Une nouvelle méthode d'alignement d'ontologies

Notre méthode d'alignement permet de détecter à la fois des correspondances simples et des correspondances complexes entre deux ontologies. La détection des correspondances simples est effectuée en combinant efficacement les trois aligneurs classiques OLA[1] (Djoufak Kengue et al., 2008), AROMA[2](David et al., 2007) et WN[3]. Chacun d'eux est fondé sur une approche particulière. Le premier aligneur est un aligneur basique qui utilise la ressource linguistique *WordNet*, le deuxième est fondé sur une approche structurelle et le troisième sur les annotations associées aux entités. Soulignons que les deux derniers aligneurs choisis sont considérés par l'OAEI[4] parmi les meilleurs systèmes d'alignement.

Puis, pour faciliter la détection des correspondances complexes, la sémantique des deux ontologies OWL est transformée sous forme de graphes. A partir des deux graphes obtenus, il s'agit de chercher des sous-graphes pertinents pouvant être alignés. Les étapes de l'alignement des deux ontologies sont résumées dans l'algorithme suivant :

Entrées : O_1 et O_2, deux ontologies à aligner
Sorties : *A*, l'alignement généré
Début :
Enrichir O_1 et O_2 par un raisonneur pour obtenir O'_1 et O'_2.
Combiner trois aligneurs existants pour obtenir l'alignement A_{Simple}.
Transformer les deux ontologies O'_1 et O'_2 sous forme de graphes.
Détecter des correspondances complexes : $A_{Complexe}$.
$A \leftarrow A_{Simple} \bigcap A_{Complexe}$.
Retourner *A*.
Fin.

3.1 Enrichissement des ontologies

La première étape consiste à lancer le raisonneur IDDL sur les deux ontologies afin de les enrichir. En effet, le raisonneur permet de déduire de nouvelles relations de subsomptions qui sont implicites. Soit l'ontologie O_2 représentée dans le tableau 1, on a par exemple $AdultFemale \sqsubseteq Adult$ et $Adult \sqsubseteq Person$, le raisonneur déduit donc que : $AdultFemale \sqsubseteq Person$. Le raisonneur peut aussi déduire d'autres relations plus complexes.

Par exemple, si on ajoute à l'ontologie O_2 une nouvelle formule qui représente les personnes qui ont fait plus de 3 publications et dont une est de type *Book*. Cette formule s'écrit comme suit : $PersonSup3 \equiv Person \sqcap (\geqslant .\top \sqcap \exists\, write.Book)$.

Et si, d'autre part, $Person \sqcap (\geqslant 3\, write.\top \sqcap \exists\, write.Book) \sqsubseteq Person \sqcap \geqslant 3\, write.\top$, le raisonneur déduit donc que $PersonSup3 \equiv Person \sqcap \geqslant 3\, write.$

Le raisonneur peut être exploité aussi pour générer des axiomes qui s'écrivent sous la forme : $\exists\, R.C$ où *R* représente un rôle et *C* un concept simple. Il s'agit donc de lui demander pour chaque couple (*C, C'*) de concepts et pour chaque rôle *R*, si l'axiome $C \sqsubseteq \exists\, R.C'$ peut

[1]OWL-Lite Alignment
[2]Association Rule Ontology Matching Approach
[3]Aligneur basique de l'API alignement appelé JWNLAlignment
[4]Ontology Alignment Evaluation Inititiave - 2009

Ontologie O_1	Ontologie O_2
$Man \sqsubseteq Human$	$AdultFemale \sqsubseteq Adult$
$Woman \sqsubseteq Human$	$Adult \sqsubseteq Person$
$write \sqsubseteq create$	$Man \sqsubseteq Adult_Man$
$\exists\, create.booklet \sqsubseteq Human$	$AdultMale \sqsubseteq Person$
$\top \sqsubseteq \forall\, create.booklet$	$Book \sqsubseteq Publication$
$Paul : Man$	$Magazin \sqsubseteq Publication$
	$Book \sqsubseteq \neg Magazin$
	$\exists\, write.\top \sqsubseteq Person$
	$\top \sqsubseteq \forall write.Publication$

TAB. 1 – *Exemple d'ontologies à aligner.*

être déduit. Ce dernier type d'axiome nous permet de détecter un nombre plus important de correspondances complexes.

3.2 Génération de l'alignement simple

La deuxième étape de l'algorithme consiste à chercher un alignement simple en combinant les résultats des trois aligneurs AROMA, OLA et WN. Afin de générer un alignement plus riche, il est intéressant de combiner les résultats de ces trois aligneurs dans la mesure où ils suivent des approches différentes. Il s'agit donc de lancer ces trois aligneurs, de comparer toutes les correspondances obtenues et de les ajouter ou non dans l'ensemble des alignements. Si une correspondance entre deux concepts est détectée par plus d'un aligneur, nous mémorisons la correspondance avec sa plus grande mesure de confiance. Puis, les correspondances détectées par un seul aligneur ayant des mesures de confiance qui dépassent un seuil donné (par exemple, 0,9) sont ajoutées aux précédentes.

3.3 Transformation des ontologies en graphe

La troisième étape de l'algorithme consiste à transformer les ontologies OWL-DL (Patel-Schneider et al., 2004) enrichies sous forme de graphes GRAPH-DL. Rappelons qu'OWL-DL est une version décidable d'OWL correspondant à la logique de description SHOIN (Baader et al., 2003). Un graphe GRAPH-DL représente les informations sémantiques contenues dans une ontologie OWL-DL. Les noeuds représentent les entités de l'ontologie, concepts ou individus. Les arcs du graphe sont les relations qui existent entre ces différentes entités et qui permettent de relier une entité source à une entité cible. Une relation peut être simple (subsomption ou disjonction) ou une propriété. Les concepts et les individus sont représentés sous forme d'ovale, et les propriétés qui relient une entité source à une entité cible sont représentées sous forme de rectangle. Les différents constructeurs et axiomes OWL qui sont pris en compte par le GRAPH-DL sont la subsomption, l'équivalence, la disjonction, la restriction existentielle, la restriction des cardinalités et la cardinalité exacte. La figure 1 montre un exemple d'ontologie OWL représentée sous forme d'un graphe GRAPH-DL.

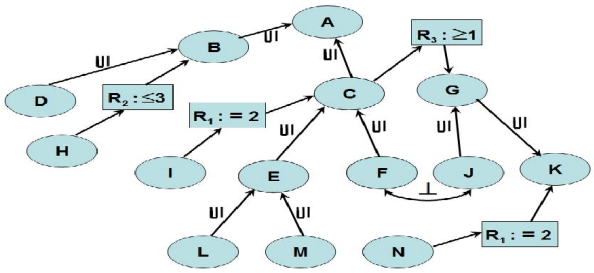


FIG. 1 – *Représentation d'une ontologie sous forme de graphe.*

4 Détection des correspondances complexes

4.1 Mesure de similarité

Pour vérifier la subsomption entre deux sous-graphes, la définition d'une mesure de similarité est nécessaire. Comme nous l'avons mentionné dans l'introduction, nous avons choisi d'utiliser une mesure de similarité terminologique. Plusieurs méthodes de mesures de similarité terminologiques peuvent être définies (Slimani et al., 2007). Citons par exemple la méthode *String Matching* (SM) (Maedche et Staab, 2002) ou les méthodes utilisant la technique des n-grammes (Blanchon et Boitet, 2007). Dans cette étude, nous nous sommes limités à tester trois mesures utilisant *Wordnet* pour faciliter la mise en correspondance de concepts complexes. Nous présentons ici celle qui a été la plus efficace à savoir une mesure de similarité "*souple*".

Cette méthode de calcul de similarité consiste en premier lieu à éliminer les mots vides (e.g. is, has, of, etc.), puis à segmenter les labels à comparer en un ensemble de tokens. Puis, nous appliquons une fonction *Sim* définie par :

$Sim = max(Sim_1, Sim_2)$ où

- Sim_1 : est une mesure qui utilise la ressource linguistique *WordNet* et qui exploite les synonymes des synsets ;
- Sim_2 : est la mesure Wu et Palmer (Wu et Palmer, 1994) appliquée sur la hiérarchie de *WordNet* et qui permet d'exploiter les liens d'hyperonymies.

En effet, on peut trouver deux labels qui ne sont pas synonymes mais qui sont hiérarchiquement très proches.

Cette mesure retourne une valeur entre 0 et 1. Il est donc nécessaire de définir un seuil acceptable pour considérer que deux labels sont similaires même si la fonction *Sim* ne renvoie pas exactement 1.

Remarque. Lors de la vérification des similarités entre propriétés voisines des deux sous-graphes à aligner, il est intéressant d'attribuer un poids plus fort aux concepts (domaine ou co-domaine des propriétés) qu'aux propriétés elles-mêmes pour mieux prendre en compte la sémantique des propriétés.

Cette mesure de similarité est appliquée dans notre algorithme d'alignement que nous allons détailler dans les propositions 1 et 2.

4.2 Proposition 1

Notre première proposition permet de détecter des correspondances entre deux concepts complexes (ou formules) (e.g. $Paper \sqcap \exists\, write^{-}.Author \sqcap \exists\, Accept^{-}.Reviewer \sqsubseteq\ Paper \sqcap \exists\, write^{-}.Author$).

La détection de ce genre de formule est effectuée en exploitant la structure des graphes qui expriment bien la sémantique des deux ontologies OWL-DL. Chaque graphe est constitué d'un ensemble de sous-graphes dont chacun représente une formule. Il s'agit donc de parcourir tous les couples de concepts des deux ontologies et vérifier si leurs sous-graphes correspondants peuvent être alignés.

Un sous-graphe correspondant à un concept donné *C*, est constitué de l'ensemble des concepts directement liés à *C* par des relations simples (subsomptions ou disjonctions) ou par des propriétés.

Pour aligner deux sous-graphes, l'un des trois cas suivants doit être vérifié :

1. le premier sous-graphe subsume le deuxième (i.e. $sousGraphe1 \sqsupseteq sousGraphe2$) et dans ce cas, une relation de subsomption sera générée ;
2. le deuxième sous-graphe subsume le premier (i.e. $sousGraphe1 \sqsubseteq sousGraphe2$) et dans ce cas une relation de subsomption dans le sens inverse de (1) sera générée ;
3. les deux sous-graphes sont équivalents : c'est-à-dire le premier sous-graphe subsume le deuxième et le deuxième subsume aussi le premier ; dans ce cas, une relation d'équivalence sera générée (i.e. $sousGraphe1 \equiv sousGraphe2$).

Un sous-graphe SG_1 subsume un autre sous-graphe SG_2 si les conditions suivantes sont vérifiées :

1. Toutes ses sous-classes directes sont similaires à des sous-classes directes de SG_2 ;
2. Toutes ses classes disjointes sont similaires à des classes disjointes de SG_2 ;
3. Toutes ses super-classes directes ou leurs généralisations sont similaires à des super-classes directes ou à leurs généralisations dans SG_2 ;
4. Toutes ses propriétés directes sont similaires à des propriétés directes de SG_2 ;
5. Toutes les cardinalités de ses propriétés sont équivalentes ou subsumées par les cardinalités des propriétés de SG_2 ;
6. Tous les domaines ou co-domaines de ses propriétés ou leurs généalisations sont similaires à des domaines ou co-domaines ou leurs généralisations dans SG_2.

4.3 Proposition 2

Cette proposition permet de détecter des correspondances entre un concept simple et une formule (e.g. $SubmittedPaper \sqsubseteq\ \exists\, submit^{-}.Author$). Elle est inspirée du travail présenté dans (Ritze et al., 2009) mais simplifiée et généralisée. Il s'agit d'essayer de trouver des ressemblances entre des concepts simples et des formules en se basant sur les similarités syntaxiques entre des concepts et des propriétés. Il est nécessaire d'utiliser une mesure de similarité qui soit purement syntaxique pour comparer les labels des concepts aux labels des propriétés. De plus, le concept à aligner avec la formule ayant une propriété qui lui ressemble syntaxiquement, doit être un concept spécialisant un concept déjà aligné avec le concept source ou cible de la propriété en question (ou l'un de ses concepts père).

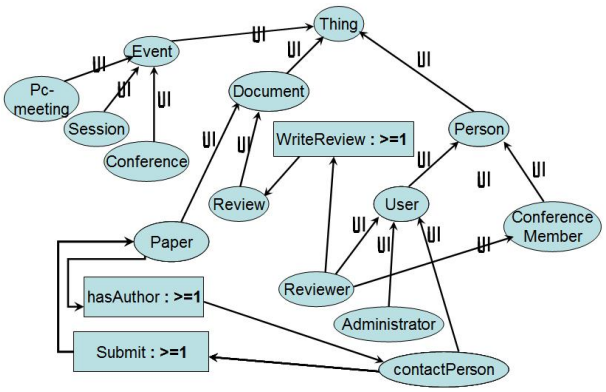


FIG. 2 – *GRAPH-DL de l'ontologie* O_1.

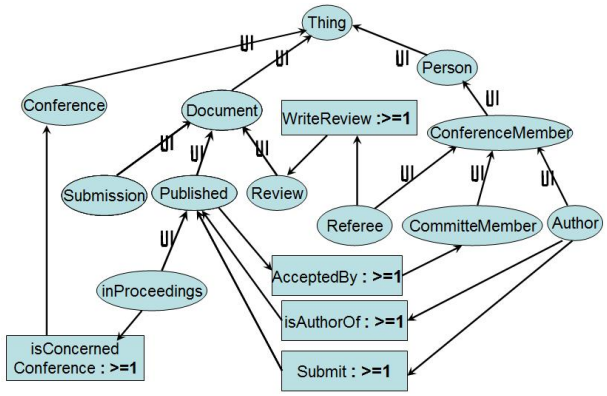


FIG. 3 – *GRAPH-DL de l'ontologie* O_2.

Pour générer les correspondances complexes détectées par la première et la deuxième proposition, nous avons utilisé lors de l'implémentation le langage expressif *EDOAL*[5] qui étend le format de l'alignement proposé par l'INRIA. Ce langage permet d'exprimer des structures complexes entre entités de différentes ontologies.

5 Exemple

Soient deux ontologies à aligner représentant le domaine des conférences. La sémantique OWL des deux ontologies est représentée sous forme de graphe (cf. les figures 2 et 3).

Les correspondances simples suivantes sont détectées lors de l'étape 1 :

[5]http ://alignapi.gforge.inria.fr/edoal.html

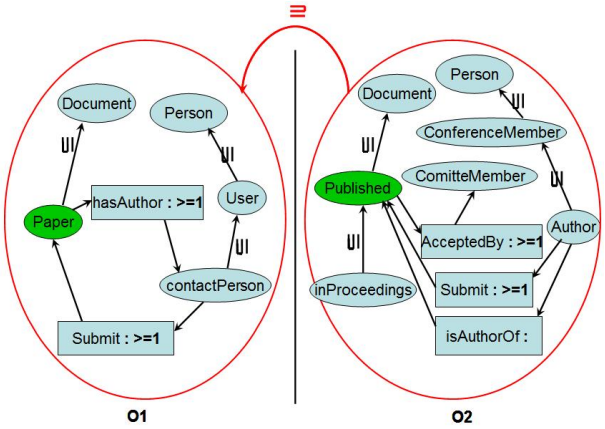


FIG. 4 – *Sous-graphes des concepts O_1 : Paper et O_2 : Published alignés.*

$O_1 : Document \equiv O_2 : Document$
$O_1 : Person \equiv O_2 : Person$
$O_1 : Reviewer \equiv O_2 : Referee$
$O_1 : Review \equiv O_2 : Review$
$O_1 : Conference \equiv O_2 : Conference$
$O_1 : Submit \equiv O_2 : Submit$
$O_1 : WriteReview \equiv O_2 : WriteReview$
$O_1 : ConferenceMember \equiv O_2 : ConferenceMember$

La deuxième étape, consistant à parcourir les sous-graphes pertinents, détecte des correspondances complexes telles que celles présentées dans les figures 4 et 5. Pour cela, le voisinage des noeuds des graphes est pris en considération. Par exemple, la correspondance générée à partir des deux sous-graphes de la figure 4 ayant comme point d'entrée les noeuds O_1 : Paper et O_2 : Published est :

$O_1 : Paper \sqcap \geqslant 1\ O_1 : hasAuthor.O_1 : contactPerson \sqcap \geqslant 1\ O_1 : Submit^-.O_1 : contactPerson \sqsupseteq O_2 : Published \sqcap \geqslant 1\ O_2 : Submit^-.O_2 : Author \sqcap \geqslant 1 O_2 : isAuthorOf^-.O_2 : Author \sqcap O_2 : AcceptedBy.O_2 : ComitteMember$

De la même façon, la correspondance générée à partir des deux sous-graphes de la figure 5 ayant comme point d'entrée les noeuds $O_1 : Reviewer$ et $O_2 : Referee$ est :

$O_1 : Reviewer \sqcap \geqslant 1\ O_1 : WriteReview.O_1 : Review \sqsubseteq O_2 : Referee \sqcap \exists\, WriteReview.O_2 : Review$

Notre méthode permet aussi de détecter des correspondances entre un concept simple et une formule. Les figures 6 et 7 sont des exemples de ce cas. Dans la figure 6, la correspondance générée est $\geqslant 1\ Submit^-.contactPerson \equiv Submission$ et dans la figure 7, la correspondance générée est $\geqslant 1\ hasAuthor^-.Paper \equiv Author$.

Nous avons implémenté les deux propositions présentées dans la section précédente sous

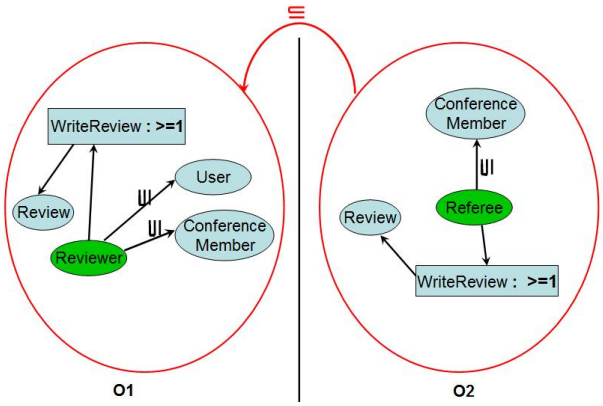


FIG. 5 – *Sous-graphes des concepts O_1 : Reviewer et O_2 : Referee alignés.*

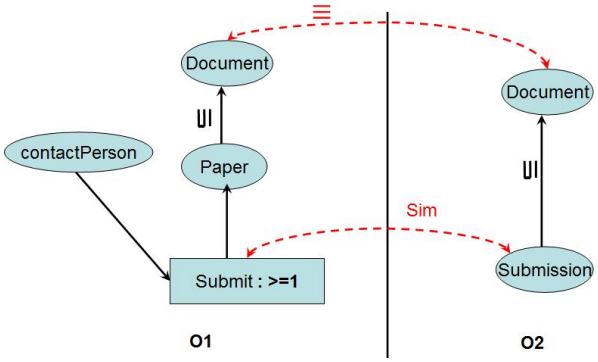


FIG. 6 – *Correspondance entre le concept O_2 : $Submission$ et une formule.*

forme de modules Java. Pour ce faire, nous avons utilisé l'API OWL[6] qui permet de manipuler facilement des ontologies OWL et l'API Align[7] qui génère l'alignement simple obtenu par la combinaison des trois aligneurs OLA, AROMA et WN. Cette API permet aussi de mémoriser l'alignement complexe en utilisant le langage expressif EDOAL[8].

6 Conclusions et perspectives

Dans cet article, nous avons présenté une nouvelle méthode d'alignement d'ontologies OWL-DL par la détection de correspondances complexes. Cette nouvelle méthode d'aligne-

[6]http ://owlapi.sourceforge.net/

[7]http ://alignapi.gforge.inria.fr/

[8]par manque de place, les résultats des expérimentations ne sont pas présentés ici mais ils sont en général meilleurs notamment par rapport à l'approche de (Ritze et al., 2009).

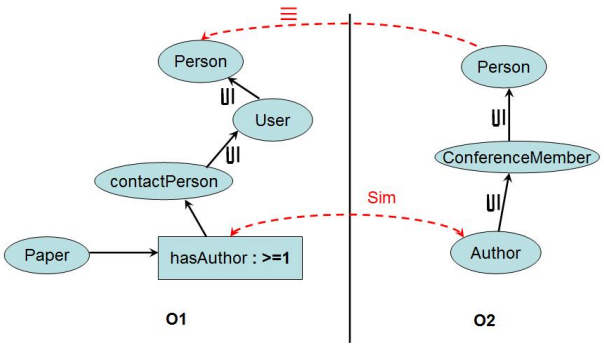


FIG. 7 – *Correspondance entre le concept O_2 : $Author$ et une formule.*

ment permet de rechercher les formules pertinentes à apparier en exploitant la sémantique OWL représentées sous forme de graphe et en utilisant une mesure de similarité terminologique. En effet, afin de trouver ces formules pertinentes, il s'agit de capturer des axiomes et des constructeurs OWL à partir de deux ontologies et de les représenter sous forme de graphes. Puis, à partir de ces graphes intégrant au mieux la sémantique des ontologies, il s'agit de rechercher des sous-graphes similaires. L'ensemble des correspondances complexes détectées permet de réduire l'hétérogénéité sémantique entre les ontologies et permet aux raisonneurs d'inférer un ensemble plus important de nouvelles connaissances.

Plusieurs améliorations sont en cours sur cette méthode d'alignement permettant de la rendre plus pertinente. Ces améliorations incluent le calcul plus riche et plus complet de la similarité terminologique, la représentation sous forme de graphes d'autres constructeurs et axiomes OWL (e.g. la restriction universelle, les propriétés de transitivité, symétrie, fonctionnelles, etc.). D'autre part, nous envisageons d'étendre le raisonneur IDDL pour qu'il prenne en considération les correspondances complexes afin de vérifier la cohérence d'un réseau d'ontologies comportant des correspondances complexes.

Références

Baader, F., D. Calvanese, D. L. McGuiness, D. Nardi, et P. F. Patel-Schneider (2003). *The Description Logic Handbook: Theory, Implementation, Applications*. Cambridge, UK: Cambridge University Press.

Bach, T. (2006). *Construction d'un Web sémantique multi-points de vue*. Thèse de doctorat, École des Mines de Paris à Sophia-Antipolis.

Blanchon, H. et C. Boitet (2007). Pour l'évaluation externe des systèmes de TA par des méthodes fondées sur la tâche. In *Revue Traitement Automatique des Langues*, Volume Vol.48-1, pp. 33–65. Atala.

David, J., F. Guillet, et H. Briand (2007). Association rule ontology matching approach. *International Journal Semantic Web Information Systems 2*, 27–49.

Djoufak Kengue, J.-F., J. Euzenat, et P. Valtchev (2008). Alignement d'ontologies dirigé par la structure. In Y. A. Ameur (Ed.), *Conférence Francophones sur les Architectures Logicielles, CAL 2008*, Volume RNTI-L-2 of *RNTI*, pp. 155–. Cépaduès-Éditions.

Euzenat, J. (2004). An api for ontology alignment. *In 3rd conference on international semantic web conference (ISWC)*, 698–712.

Euzenat, J. et P. Shvaiko (2007). *Ontology matching*. Heidelberg (DE): Springer Verlag.

Maedche, E. et S. Staab (2002). Measuring similarity between ontologies. In *Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW)*, pp. 251–263. Springer.

Patel-Schneider, P., P. Hayes, et I. Horrocks (2004). Owl web ontology language semantics and abstract syntax. In *W3C Recommendation*.

Ritze, D., C. Meilicke, O. Šváb Zamazal, et H. Stuckenschmidt (2009). A pattern-based ontology matching approach for detecting complex correspondences. *Proceedings of the ISWC 2009 Workshop on Ontology Matching*.

Sirin, E., B. Parsia, B. C. Grau, A. Kalyanpur, et Y. Katz (2007). Pellet: A practical owl-dl reasoner. *Journal of Web Semantics 5*(2), 51–53.

Slimani, T., B. B. Yaghlane, et K. Mellouli (2007). Une extension de mesure de similarité entre les concepts d'une ontologie. In *Proceedings of the 4rth International Conference: Sciences of Electronic,Technologies of Information and Telecommunications (SETIT)*, pp. 1–10.

Tsarkov, D. et I. Horrocks (2006). FaCT++ description logic reasoner: System description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006)*, Volume 4130 of *Lecture Notes in Artificial Intelligence*, pp. 292–297. Springer.

Wu, Z. et M. Palmer (1994). Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Associations for Computational Linguistics*, pp. 133–138.

Zghal, S., S. B. Yahia, E. M. Nguifo, et Y. Slimani (2009). SODA: an OWL-DL based ontology matching system,
disponible à : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.142.6984.

Zimmermann, A. et C. Le Duc (2008). Reasoning with a network of aligned ontologies. *Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems (ICWRRS)*, 43–57.

Summary

This paper presents a new approach to align a simple concept with a complex concept or two complex concepts of ontologies described in OWL-DL. Our method runs in two steps. The first one combines three classical alignments to generate simple mappings. The main goal is to obtain a richer alignment. Then, in a second step, complex correspondences are generated. So, the ontologies to align are transformed in graph integrating the OWL semantic. From these graphs, the method detects the relevant subgraphs representing alignable formulas, taking into account their structure and using a similarity measure terminology.

Gestion du conflit dans l'appariement des ontologies

Amira Essaid*, Boutheina Ben Yaghlane**
Arnaud Martin***

*LARODEC, Université de Tunis/ISG Tunis
essaid_amira@yahoo.fr
**LARODEC, Université 7 Novembre à Carthage/IHEC Carthage
boutheina.yaghlane@ihec.rnu.tn
***UMR 6074 IRISA, Université de Rennes1 / IUT de Lannion
Arnaud.Martin@univ-rennes1.fr

Résumé. L'alignement des ontologies représente un grand intérêt dans le web sémantique. En effet, il permet de pallier le problème d'hétérogénéité sémantique et d'assurer une interopérabilité entre les systèmes utilisant des ontologies distribuées et hétérogènes. Cet alignement est réalisé grâce aux différentes approches fondées sur des mesures de similarité. L'appariement des ontologies fait naturellement apparaître du conflit entre les résultats fournis par ces mesures. Dans cet article, nous proposons d'appliquer la théorie des fonctions de croyance pour gérer ce conflit et ceci par la combinaison des résultats des différentes mesures. Plusieurs règles de combinaison peuvent être appliquées à cet effet.

1 Introduction

Le *web sémantique* introduit par Berners-Lee et al. (2001), tend à rendre le contenu des documents web accessible par les applications afin d'assurer un échange d'information et une interopérabilité entre les différents systèmes. Les ontologies proposées comme élément central du web sémantique, sont définies par Gruber (1993) comme "*une spécification explicite d'une conceptualisation*". En effet, selon Kengue et al. (2008), les ontologies permettent de décrire un domaine en définissant un ensemble de concepts et les relations qu'ils entretiennent avec d'autres concepts par spécialisation ou à travers des propriétés.

Les ontologies sont utilisées pour la représentation des connaissances. Plusieurs ontologies couvrant totalement ou partiellement un même domaine d'application peuvent être développées indépendamment les unes des autres. Les entités de ces ontologies peuvent être définies selon différents niveaux de granularité ou simplement décrites grâce à des différents langages de représentation. Afin de permettre l'interopérabilité et le partage des données, il est primordial que l'hétérogénéité entre les connaissances décrites dans les ontologies soit résolue et ceci par la création des liens sémantiques entre les entités, c'est le but de l'*alignement*.

Etant données deux ontologies, l'alignement (appariement ou mise en correspondance) consiste en la production d'un ensemble de correspondances entre les entités. Ces entités

peuvent être des concepts, des propriétés ou encore des instances. Cet ensemble de correspondances ou encore alignement peut être par la suite utilisé pour fusionner les ontologies, créer une troisième ontologie à partir des entités des deux ontologies en entrée et la liste des correspondances ou encore effectuer des tâches de raisonnement entre les ontologies appariées.

La découverte des alignements d'une manière manuelle est une tâche coûteuse en temps, inefficace et peut produire des erreurs. C'est pour cette raison que plusieurs méthodes d'appariement ont été proposées. Euzenat et Shvaiko (2007) donnent un état de l'art exhaustif des différentes techniques d'alignement. Ces méthodes sont fondées sur des mesures de similarité. Dans le but d'améliorer le résultat de l'alignement, plusieurs mesures peuvent être utilisées simultanément ce qui fait apparaitre inévitablement un conflit entre les différents résultats produits par chacune de ces mesures.

La théorie des fonctions de croyance introduite par Dempster (1967) et Shafer (1976) permet de combiner des informations provenant de plusieurs sources hétérogènes. Ces informations fournies peuvent être en conflit. Dans le cadre de cette théorie, la gestion du conflit peut se faire de deux manières différentes : soit nous nous intéressons à gérer le conflit avant la combinaison des informations des sources ce qui revient à le réduire et donc d'affaiblir ces informations selon le degré de fiabilité de la source, soit nous pouvons tenir compte du conflit lors de la combinaison et ceci par application des règles de combinaison qui permettent de supprimer le conflit en le redistribuant de manières différentes sur les informations disponibles.

Dans cet article, nous proposons de gérer le conflit dans l'alignement des ontologies par l'utilisation de la théorie des fonctions de croyance. Tout d'abord, nous considérons chacune des mesures de similarité comme une source d'évidence ayant un degré de confiance par rapport aux résultats d'alignement fournis. Ces résultats font naturellement apparaitre du conflit qu'on gérera lors de l'étape de combinaison de ces résultats et ceci par application des règles de combinaison adéquates.

La suite de cet article est organisée comme suit : dans la deuxième section, nous rappelons la définition de l'alignement des ontologies ainsi que les mesures de similarité que nous avons utilisées. La troisième section est dédiée à la présentation des concepts de base de la théorie des fonctions de croyance. Enfin, dans la quatrième section nous décrivons notre approche de gestion du conflit dans l'alignement des ontologies pour ainsi présenter dans la section suivante quelques directives des travaux de recherche envisageables dans le futur.

2 L'alignement des ontologies et les mesures de similarité

2.1 L'alignement des ontologies

Afin de faciliter l'échange entre les applications, plusieurs langages de représentation des connaissances ont été développées parmi lesquels le langage *OWL*[1]. Ce dernier fournit un grand nombre de constructeurs pour représenter les entités (*e.g.* "owl :class", "rdfs :subClassOf"...).

Selon Ehrig et Staab (2004), une ontologie est définie formellement par le tuple suivant :

$$O := (C,\ H_C,\ R_C,\ H_R,\ I,\ R_I,\ A)$$

1. http ://www.w3.org/TR/owl-ref/

Une ontologie est un ensemble de concepts *C* (des instances de *'owl :class'*) reliés par une hiérarchie de subsomptions H_C (relation binaire correspondant à *'rdfs :subClassOf'*). Les relations R_C (des instances de *'owl :objectProperty'*) relient les concepts. Ces relations sont organisées selon une hiérarchie de subsomption H_R (*'rdfs :subPropertyOf'*). Un individu *i* appartenant à l'ensemble des individus *I* est une instance de la classe *c* tel que $c \in C$. Cette instance *i* peut être reliée à une instance *j* par une relation *r* appartenant à R_I. Le triplet (*i, r, j*) est une instance de propriété. Une ontologie permet aussi d'inférer de nouvelles connaissances grâce à l'ensemble des axiomes noté par *A*.

Aujourd'hui, il existe un très grand nombre d'ontologies disponibles sur le web. L'utilisation d'une seule ontologie ne permet pas de soutenir les tâches demandées par le web sémantique. Les applications doivent alors utiliser plusieurs ontologies distribuées et hétérogènes. Ceci est réalisé par un appariement des ontologies qui selon Euzenat et Shvaiko (2007) consiste en une fonction *f* définie par :

$$A = f\ (O_1, O_2, A', p, r)$$

Cette fonction tend à partir de deux ontologies O_1 et O_2, d'un ensemble d'alignements *A'*, d'un paramètre *p* (seuil minimal de similarité) ainsi que des ressources externes *r* (*e.g.* thesaurus) à renvoyer un alignement *A* : en d'autres termes à chercher pour chaque entité *e* (concept, relation, ou encore instance) de l'ontologie O_1, les entités correspondantes se trouvant dans O_2 ayant des sémantiques similaires. La figure 1 montre l'appariement de deux ontologies relatives aux références bibliographiques. L'alignement doit en effet identifier la classe *Congress* comme correspondante à la classe *Conference* et que la classe *Address* doit être reliée à la classe *Directions*.

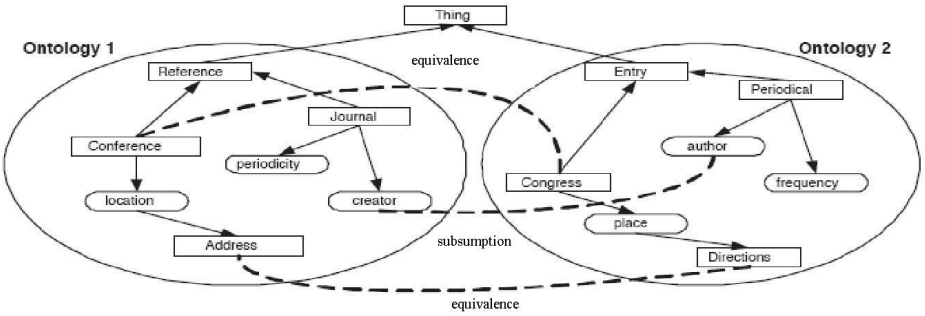


FIG. 1 – *Extrait d'appariement de deux ontologies bibliographiques.*

Dans cet article, nous nous intéressons à aligner uniquement les concepts représentés dans une ontologie en *OWL* par *"owl :class"*.

2.2 Les mesures de similarité

Rahm et Bernstein (2001) proposent la classification suivante des différentes techniques d'alignement :

1. **Les méthodes terminologiques** : Elles comparent les labels des différentes entités. Elles sont décomposées en :
   (a) *Les méthodes syntaxiques* : Elles effectuent la correspondance à travers les mesures de dissimilarité des chaînes.
   (b) *Les méthodes lexicales* : Elles effectuent la correspondance à travers les relations lexicales (synonymie, hyponymie ...).
2. **Les méthodes structurelles** : Elles se fondent sur la comparaison des structures des entités. On distingue :
   (a) *Les méthodes de comparaison de structures internes* : Elles se fondent sur la structure interne des entités (cardinalité des attributs, transitivité des propriétés, ...).
   (b) *Les méthodes de comparaison de structures externes* : La comparaison de similarité entre deux entités des deux ontologies dépend des relations de ces entités avec d'autres entités. Des types de relations existent : les relations taxonomiques, les relations méréologiques, ...
3. **Les méthodes sémantiques** : Elles sont fondées sur le modèle des entités utilisé pour valider et justifier les résultats d'alignement. Ce sont des méthodes déductives. Parmi ces techniques, nous citons la satisfiabilité propositionnelle qui tend à trouver les alignements en transformant deux ontologies ainsi que la requête d'alignement (le couple d'entités à apparier et la relation possible entre ces entités) en des formules propositionnelles et par la suite de vérifier sa validité.

Dans cet article, nous avons appliqué des méthodes terminologiques afin de chercher les correspondances entre les entités de deux ontologies. Nous avons utilisé trois mesures syntaxiques à savoir *la distance de Levenshtein*, *la distance Jaro* ainsi que *la distance de Hamming*. Quant à la mesure lexicale, nous avons eu accès au dictionnaire *WordNet 2.0* proposé par Miller (1995) pour identifier les correspondances. Ci dessous, nous rappelons brièvement les mesures syntaxiques utilisées.

– **Distance de Levenshtein** : Hall et Dowling (1980) définissent cette distance comme étant la mesure la similarité entre deux chaînes de caractères. Elle est égale au nombre d'opérations nécessaires pour transformer une chaîne en une autre. Les opérations consistent en suppression, insertion ou remplacement. Nous définissons la mesure de similarité entre deux mots w_1, w_2 comme :

$$sim_{ed}(w_1, w_2) = \frac{1}{1 + ed(w_1, w_2)} \quad (1)$$

Notons que $ed(w_1, w_2)$ est la distance de Levenshtein entre les deux mots w_1 et w_2.

– **Distance Jaro** : Elle mesure le nombre et l'ordre des caractères communs entre deux chaînes de caractères. Nous définissons la distance de similarité de Jaro entre deux mots w_1, w_2 comme la distance Jaro entre les deux chaînes de caractères des deux mots.

– **Distance de Hamming** : Elle mesure le nombre de positions au niveau desquelles les deux chaînes de caractères diffèrent.

3 La théorie des fonctions de croyance

3.1 Les concepts de base

La théorie des fonctions de croyance appelée aussi théorie de l'évidence est issue des travaux de Dempster (1967) et fut reprise par Shafer (1976). C'est un outil qui permet de modéliser aussi bien l'incertitude que l'imprécision. Il prend en compte des ambiguïtés et des conflits entre les sources. Nous présentons dans ce qui suit les concepts de base de cette théorie.

Pour un problème donné, la théorie des fonctions de croyance définit un *cadre de discernement* noté Θ comme étant l'ensemble des *N* hypothèses exhaustives et exclusives.

$$\Theta = \{H_1, H_2, \ldots H_N\}$$

Un ensemble 2^Θ, défini à partir de Θ contient les hypothèses singletons de Θ, toutes les disjonctions possibles de ces hypothèses ainsi que l'ensemble vide.

La théorie des fonctions de croyance est fondée sur la manipulation des fonctions de masse appelée encore *masse élémentaire de croyance*. La fonction de masse *m* définie sur 2^Θ et à valeurs dans $[0, 1]$, vérifie les propriétés suivantes :

$$m(\emptyset) = 0 \tag{2}$$

$$\sum_{A \subseteq \Theta} m(A) = 1 \tag{3}$$

Les éléments *A* ayant une masse de croyance non nulle sont appelés *les éléments focaux*.

D'autres mesures de croyance peuvent être calculées à partir des fonctions de masse, à savoir la crédibilité (*bel*) et la plausibilité (*pl*).

La fonction de crédibilité (*bel*) mesure la croyance minimale apportée à un sous ensemble de 2^Θ . Cette fonction est définie pour tout $A \in 2^\Theta$ et à valeurs dans $[0, 1]$ par :

$$bel(A) = \sum_{B \subseteq A, B \neq \emptyset} m(B) \tag{4}$$

La fonction de plausibilité (*pl*) mesure la croyance maximale apportée à un sous ensemble de 2^Θ. Cette fonction est définie pour tout $A \in 2^\Theta$ et à valeurs dans $[0, 1]$ par :

$$pl(A) = \sum_{A \cap B \neq \emptyset} m(B) \tag{5}$$

Afin de bien illustrer les notions présentées ci dessus, nous proposons dans ce qui suit un exemple d'organisation des conférences. Afin de participer à une conférence, les auteurs soumettent leurs articles. Ces contributions seront par la suite évaluées par des relecteurs. Nous pouvons identifier trois types de travaux à savoir un poster (PS), un papier long (PL) ou encore un papier court (PC). Suite à l'évaluation, un papier long peut être accepté, par exemple, à ce qu'il soit réécrit comme un papier court. Le problème peut être modélisé par un cadre de discernement $\Theta = \{PS, PL, PC\}$. L'ensemble 2^Θ correspondant est :
$2^\Theta = \{\emptyset, PS, PL, PC, PS \cup PL, PS \cup PC, PL \cup PC, \Theta\}$.

Supposons qu'un des relecteurs a évalué un papier et a exprimé son degré de croyance par les masses suivantes :

$$m(PL) = 0.3,\ \ m(PL \cup PC) = 0.5,\ \ m(PS \cup PC \cup PL) = 0.2$$

Le relecteur a un degré de croyance de 0.3 que le papier doit être long, de 0.5 qu'il peut être accepté en tant que papier long ou réécrit en tant que papier court et un degré d'ignorance totale (le papier peut être un poster, un papier long ou encore un papier court) de 0.2. Nous présentons dans le tableau 1 les fonctions de masse fournies par le relecteur ainsi que les fonctions de crédibilité et de plausibilité.

	m	bel	pl
$\emptyset$	0	0	0
PL	0.3	0.3	1
PC	0	0	0.7
PL $\cup$ PC	0.5	0.8	1
PS	0	0	0.2
PL $\cup$ PS	0	0.3	1
PC $\cup$ PS	0	0	0.7
Θ	0.2	1	1

TAB. 1 – *Les fonctions de masse, de crédibilité et de plausibilité relatives à l'évaluation d'un article.*

3.2 Les règles de combinaison

La fusion des données imparfaites (incertaines, imprécises et incomplètes) est une solution pour obtenir une information plus pertinente et plus fiable. La théorie des fonctions de croyance est un outil intéressant de fusion de données. En effet, pour un problème donné et pour un même cadre de discernement, il est possible d'obtenir une fonction de masse synthétisant les différentes connaissances issues des différentes sources d'information *distinctes* et *indépendantes* et ceci par application d'une règle de combinaison.

Il existe plusieurs modes de combinaison développés dans le cadre de la théorie des fonctions de croyance. Nous présentons dans ce qui suit la combinaison conjonctive, la combinaison disjonctive ainsi que la combinaison mixte. Afin d'illustrer les différentes règles, nous supposons que nous avons deux sources S_1 et S_2 distinctes et indépendantes. Ces sources fournissent des masses m_1 et m_2 pour un même cadre de discernement Θ.

3.2.1 Combinaison conjonctive

Elle fut introduite par Dempster (1967) et reprise par Shafer (1976), elle se fonde sur la combinaison des fonctions de masse en considérant les intersections des éléments de 2^{Θ}.

Règle orthogonale normalisée de Dempster-Shafer Etant données deux fonctions de masse m_1 et m_2, cette règle de combinaison est présentée comme suit :

$$m_{1\oplus 2}(A) = \begin{cases} \dfrac{\sum\limits_{B\cap C=A} m_1(B)\times m_2(C)}{1-\sum\limits_{B\cap C=\emptyset} m_1(B)\times m_2(C)} & \forall A\subseteq\Theta,\ A\neq\emptyset \\ 0 & si\ A=\emptyset \end{cases} \tag{6}$$

Notons que m_1(B) et m_2(C) sont les fonctions de masse associées aux deux sources S_1 et S_2 respectivement et que $\sum\limits_{B\cap C=\emptyset} m_1(B)\times m_2(C)$ est une mesure de conflit entre les sources. Cette règle est normalisée par $1-\sum\limits_{B\cap C=\emptyset} m_1(B)\times m_2(C)$. Cette normalisation masque le conflit et n'est intéressante que sous l'hypothèse du monde fermé (toutes les hypothèses possibles du problème appartiennent au cadre de discernement Θ) et aussi lorsqu'il n'y a pas de conflit entre les sources. En effet, la fonction de masse obtenue suite à la combinaison renforce la croyance sur les décisions pour lesquelles les sources sont concordantes et l'atténue en cas de conflit.

Règle de Smets Smets propose la forme non normalisée de la règle précédemment décrite. Cette règle est utilisée sous l'hypothèse du monde ouvert où une masse non nulle est affectée à l'ensemble vide. La règle conjonctive de combinaison est définie alors comme suit :

$$m_{1\bigcirc\!\!\!\cap 2}(A) = \sum_{B\cap C=A} m_1(B)\times m_2(C) \tag{7}$$

Cette règle est principalement utilisée lorsque les deux sources sont fiables.

Règle de Yager Yager (1987) propose une règle où la mesure de conflit est affectée au cadre de discernement Θ. Pour deux fonctions de masse m_1 et m_2, la règle de Yager est définie comme suit :

$$\begin{cases} m_Y(A) = m_{1\bigcirc\!\!\!\cap 2}(A) & \forall A\in 2^\Theta,\ A\neq\Theta\ et\ A\neq\emptyset \\ m_Y(\Theta) = m_{1\bigcirc\!\!\!\cap 2}(\Theta) + m_{1\bigcirc\!\!\!\cap 2}(\emptyset) & \\ m_Y(\emptyset) = 0 & \end{cases} \tag{8}$$

3.2.2 Combinaison disjonctive

La règle de combinaison disjonctive a été proposée par Smets (1993) pour combiner des fonctions de masse dont l'une au moins est fiable, sans pour autant quantifier la fiabilité ni identifier laquelle des deux sources est fiable. Cette règle est définie pour $\forall A\in 2^\Theta$ par l'équation suivante :

$$m_{1\bigcirc\!\!\!\cup 2}(A) = \sum_{B\cup C=A} m_1(B)\times m_2(C) \tag{9}$$

3.2.3 Combinaison mixte

Dubois et Prade (1988) ont proposé un compromis afin de tenir compte des avantages des deux modes de combinaison à savoir la combinaison conjonctive et la combinaison disjonctive. Cette combinaison est donnée pour tout $A \in 2^{\Theta}$ par la formule suivante :

$$\begin{cases} m_{DP}(X) = m_{1\textcircled{\cap}2}(X) + \sum\limits_{A\cap X=\emptyset,\ A\cup X=B} m_1(X)m_2(A) \quad \forall A \subseteq \Theta,\ A \neq \emptyset \\ m_{DP}(\emptyset) = 0 \end{cases} \tag{10}$$

Ce mode de combinaison suppose que le conflit provient de la non fiabilité des sources.

4 Une approche de gestion du conflit dans l'alignement des ontologies

Dans le but d'apparier les ontologies, plusieurs mesures de similarité peuvent être utilisées afin de tenir compte des aspects terminologiques des différents concepts de deux ontologies à aligner. L'utilisation de ces mesures peut faire apparaitre du conflit. Dans le cadre de la théorie des fonctions de croyance, nous proposons de gérer le conflit une fois que nous avons appliqué ces mesures. Pour ce faire, nous devons modéliser le problème d'appariement dans le cadre de cette théorie. En effet, les mesures de similarité utilisées seront considérées comme les sources d'évidence fournissant une information quant aux classes à apparier. Les résultats de ces mesures seront interprétés comme des fonctions de masse fournies par une source.

Afin d'illustrer notre modélisation, nous nous fondons sur deux ontologies *cmt* et *conference* relatives à l'organisation des conférences [2]. *cmt* est une ontologie source pour laquelle nous cherchons pour chacune de ses classes ses correspondantes au niveau de l'ontologie cible *conference*. Le tableau 2 met en évidence quelques classes de l'ontologie *cmt* (Paper, PaperAbstract, Document, Preference) ainsi que leurs correspondantes de l'ontologie *conference* suite à l'application des mesures terminologiques.

	Distance Levenshtein	Distance Jaro	Distance Hamming	Méthode Lexicale
Paper	Paper	Paper	Paper	Paper
PaperAbstract	Abstract	Paper	Paper	Abstract
Document	Conference_document	Committee	Co-chair	Paper
Preference	Conference	Review_preference	Conference	Review_preference

TAB. 2 – *Appariement des ontologies cmt et conference.*

La distance Levenshtein, la distance jaro, la distance Hamming ainsi que la méthode lexicale sont représentées comme les sources d'évidence ($S_{lev}, S_{jaro}, S_{ham}, S_{lex}$). Les quatre sources considèrent que la classe *Paper* de l'ontologie *cmt* doit être appariée à la classe *Paper* de l'ontologie *conference*. S_{lev} et S_{lex} relient la classe PaperAbstract de l'ontologie source à la classe Abstract de l'ontologie cible tantdis que les deux autres sources S_{jaro} et S_{ham} relient

2. http ://nb.vse.cz/svabo/oaei2010/

PaperAbstract à *Paper*. Quant à la classe *Document*, le tableau 2 montre bien que les quatre sources ont des points de vue différents.

Nous définissons le cadre de discernement comme étant l'ensemble de tous les couples formés à partir des classes des deux ontologies. $\Theta = \{(c_i, e_1), (c_i, e_2), \ldots (c_i, e_n)\}$ où c_i et e_j représentent respectivement une classe de l'ontologie source et son correspondant de l'ontologie cible fournie par la mesure de similarité. Les classes pour lesquelles les quatre sources donnent un même résultat seront exclues de cette modélisation comme le cas de *Paper*. Par exemple, le Θ correspondant au tableau 2 est représenté comme suit :

$\Theta = \{(PaperAbstract, Abstract), (PaperAbstract, Paper),$
$(Document, Conference\_document), (Document, Committee),$
$(Document, Co - chair), (Document, Paper),$
$(Preference, Conference), (Preference, Review\_preference)$

Le tableau 3 illustre les résultats de ces différentes mesures. Dans le cadre de la théorie des fonctions de croyance, ces résultats sont interprétés comme des fonctions de masse.

	Distance Levenshtein	Distance Jaro	Distance Hamming	Méthode Lexicale
Paper	1.000	1.000	1.000	1.000
PaperAbstract	0.615	0.795	0.385	0.762
Document	0.421	0.638	0.125	0.910
Preference	0.700	0.670	0.700	0.740

TAB. 3 – *Résultats d'appariement des ontologies cmt et conference.*

Par exemple pour la classe *Paper*, les quatre sources sont en accord et attribuent une masse égale à 1.000 à la classe *Paper* de l'ontologie cible comme suit :

$m^{S_{lev}}$(Paper, Paper) = 1.000, $m^{S_{jaro}}$(Paper, Paper) = 1.000,
$m^{S_{ham}}$(Paper, Paper) = 1.000, $m^{S_{lex}}$(Paper, Paper) = 1.000.

Quant à la classe *PaperAbstract* et malgré le choix d'une même classe *Paper* à apparier à *PaperAbstract*, les deux sources S_{jaro} et S_{ham} n'attribuent pas une même masse. En effet, nous avons :

$m^{S_{lev}}$(PaperAbstract, Abstract) = 0.615, $m^{S_{jaro}}$(PaperAbstract, Paper) = 0.795,
$m^{S_{ham}}$(PaperAbstract, Paper) = 0.385, $m^{S_{lex}}$(PaperAbstract, Abstract) = 0.762.

Etant donné le conflit entre les différentes mesures, on gérera ce conflit lors de la combinaison des résultats des différentes sources et ceci par application des règles de combinaison précédemment présentées.

5 Discussion

Dans cet article, nous nous sommes limités à apparier les ontologies tout en cherchant à relier les concepts qui sont équivalents en appliquant des techniques terminologiques. Nous

pouvons chercher dans le futur à apparier les individus ou encore les propriétés de deux ontologies tout en utilisant des techniques structurelles ou sémantiques. Le fait d'appliquer uniquement les techniques terminologiques a permis de détecter un conflit. L'utilisation d'autres méthodes d'alignement permettra de tenir compte des différents aspects des entités tel que la relation d'une entité avec les entités qui lui sont directement reliées. Dans ce cas, la gestion du conflit sera intéressante.

Dans cet article, nous nous sommes contentés de proposer une approche où la gestion du conflit est effectuée lors la combinaison des informations issues des différentes mesures de similarité. Cette approche peut être étendue en s'intéressant à gérer le conflit avant même de combiner en estimant la fiabilité des mesures de similarité tout en affaiblissant les informations des mesures et donc de réduire le conflit.

6 Conclusion

Dans cet article, nous avons proposé une approche de gestion du conflit dans l'appariement des ontologies. Nous nous sommes limités à chercher les correspondances entre les classes de deux ontologies en utilisant quatre mesures de similarité terminologiques. Une fois que nous avons identifié pour chaque classe de l'ontologie source son correspondant de l'ontologie cible, une gestion du conflit est effectuée par application de la théorie des fonctions de croyance. En effet, les mesures de similarité sont assimilées à des sources d'évidence et les résultats fournis par ces mesures sont traités comme des fonctions de masse. La gestion du conflit est effectuée lors de la combinaison des fonctions de masse des différentes sources par application des règles de combinaison.

Comme perspective, nous envisageons à appliquer différentes règles de combinaison et de comparer les résultats fournis par ces règles. Des tests expérimentaux seront effectués sur différents couples d'ontologies fournis par l'OAEI [3].

Références

Berners-Lee, T., J. Hendler, et O. Lassila (2001). The semantic web. *Scientific American 284*, 34–43.

Dempster, A. (1967). Upper and Lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics 38*, 325–339.

Dubois, D. et H. Prade (1988). Representation and combination of uncertainty with belief functions and possibility measures. *Computational Intelligence 4*, 244–264.

Ehrig, M. et S. Staab (2004). Qom - quick ontology mapping. In *Proceedings of the International Semantic Web Conference (ISWC04), Springer-Verlag LNCS 3298*, pp. 289–303.

Euzenat, J. et P. Shvaiko (2007). Ontology Matching. Springer-Verlag.

Gruber (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition 5*, 199–220.

3. Ontology Alignment Evaluation Initiative : http ://oaei.ontologymatching.org/2010/

Hall, P. A. V. et G. R. Dowling (1980). Approximate string matching. *ACM Computing Surveys 12*(4), 381–402.

Kengue, J. F. D., J. Euzenat, et P. Valtchev (2008). Alignement d'ontologies dirigé par la structure. In Y. A. Ameur (Ed.), *CAL*, Volume RNTI-L-2 of *Revue des Nouvelles Technologies de l'Information*, pp. 155. Cépaduès-Éditions.

Miller, G. A. (1995). Wordnet : A lexical database for english. *Communications of the ACM 38*, 39–41.

Rahm, E. et P. A. Bernstein (2001). A survey of approaches to automatic schema matching. *The International Journal on Very Large Data Bases 10*, 334–350.

Shafer, G. (1976). A Mathematical Theory of Evidence. *Princeton University Press*.

Smets, P. (1993). Belief functions : The disjunctive rule of combination and the generalized bayesian theorem. *International Journal of Approximate Reasoning 9*(1), 1–35.

Yager, R. R. (1987). On the dempster-shafer framework and new combination rules. *Informations Sciences 41*, 93–137.

Summary

Mapping ontologies is a crucial step to facilitate semantic interoperability between systems. Different matchers can be used in order to find the correspondences between two ontologies. These matchers can be contradictory, thus leading to a conflict. In order to manage this conflict, we propose an approach by using the Dempster-Shafer theory. Every matcher is considered as a source of evidence and the match results as the mass functions. Managing conflict is dealt when combining between the different results using for that purpose different rules of combination.