Mesurer la similarité de graphes étiquetés

Sébastien Sorlin, Pierre-Antoine Champin, Christine Solnon

Mesurer la similarité de graphes étiquetés

Sébastien Sorlin **Pierre-Antoine Champin**
Christine Solnon
LIRIS CNRS FRE 2672
bât. Nautibus, Université Lyon I
43 Bd du 11 novembre, 69622 Villeurbanne cedex
{sebastien.sorlin,champin,csolnon}@bat710.univ-lyon1.fr

Résumé

Cet article s'intéresse au problème de mesurer la similarité de graphes orientés étiquetés, i.e., des graphes tels que chaque arc et chaque sommet possèdent un ensemble d'étiquettes (de caractéristiques). Nous définissons tout d'abord le problème du calcul de la similarité de deux graphes étiquetés comme la recherche d'un meilleur appariement de leurs sommets. Ce problème se distingue du problème de l'isomorphisme de graphes par le fait que ces appariements autorisent la mise en correspondance d'un sommet d'un graphe avec 0, 1 ou plusieurs autres sommets de l'autre graphe. Nous étudions ensuite la complexité de notre problème, et nous proposons deux algorithmes : un algorithme basé sur une exploration complète par "séparation et évaluation", et un algorithme glouton.

1 Introduction

Le paradigme du raisonnement à partir de cas (RàPC) se base sur l'hypothèse que des problèmes similaires admettent des solutions similaires. Un système de RàPC résout un problème en réutilisant un problème similaire (nommé alors "cas") dont on connaît la solution [Aamodt et Plaza, 1994]. Rechercher un cas similaire nécessite une mesure de similarité entre cas. Lorsque les cas sont représentés par un vecteur de paires (attribut-valeur), la similarité entre cas est facilement évaluable, *e.g.*, en faisant une somme pondérée des similarités entre leurs paires [Diday, 1982]. Cependant, des problèmes plus complexes nécessitent une représentation plus stucturée des cas [Lieber et Napoli, 1996; Petrovic *et al.*, 2002]. On s'intéresse dans cet article à une application du raisonnement à partir de cas dans le domaine de la conception assistée par ordinateur (CAO) où les cas sont des objets de conception. Ces objets structurés sont décrits par des graphes orientés étiquetés, et nous étudions dans cet article le problème de mesurer la similarité de tels graphes.

Notons que si ce travail a trouvé son origine dans ce contexte du RàPC appliqué à la CAO, il existe bien d'autres applications nécessitant d'évaluer la similarité de deux objets, en expli-

citant leurs points communs et leurs différences ; un corollaire de ce problème étant de rechercher, parmi un ensemble d'objets, celui qui "ressemble" le plus à un objet donné. En particulier, lorsque l'on recherche des informations sur le "Web", il s'agit de localiser, dans le gigantesque réseau d'informations que constitue la "toile informatique", la partie qui répond au mieux à la requête de l'internaute. Une autre application possible de ce travail est l'extraction d'images où, étant donnée une base de données d'images, il s'agit de retrouver toutes les images similaires à une requête image.

Nous illustrons maintenant les spécificités de notre mesure de similarité sur un exemple tiré d'une application en CAO [Champin, 2002]. La figure 1 présente deux objets similaires : les poutres a, b, c et d de l'objet 1 jouent le même rôle que les poutres 1, 2, 3 et 4 de l'objet 2, tandis que les murs e et f correspondent au mur 5. Ces deux objets, bien que similaires, ne sont toutefois pas identiques : les poutres n'ont pas la même forme (en I à gauche et en U à droite) et le nombre de murs est différent (le mur 5 à droite joue seul le rôle des murs e et f à gauche).

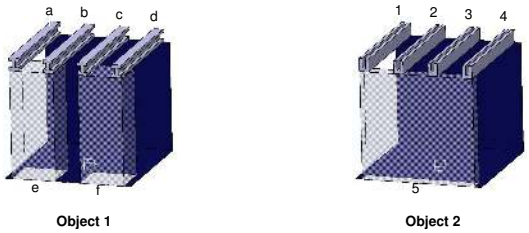


FIG. 1 – Deux objets de conception similaires

Cet exemple nous montre qu'afin de mesurer la similarité de deux objets, il est nécessaire d'établir une correspondance entre leurs composants. Plus précisémment, il s'agit de trouver le meilleur appariement possible : celui qui met en correspondance les composants les plus similaires, la similarité des composants étant fonction des caractéristiques qu'ils ont en commun. Par exemple, les composants a et 1 partagent les caractéristiques "être une poutre", "être à coté de" une autre poutre et "être sur" un mur, tandis qu'ils ne partagent pas la caractéristique de forme. Un premier point important de notre mesure de similarité est qu'elle n'est pas seulement quantitative (évaluant le degré de similarité de deux objets) mais aussi qualitative (explicitant en quoi les objets sont similaires et en quoi ils sont différents). Cette évaluation qualitative des différences entre deux objets est alors semblable à la notion d'*explications* proposée dans [Guéhéneuc et Jussien, 2001]. Un second point important de notre mesure de similarité est que l'importance relative des différentes caractéristiques n'est pas imposée, mais est définie par une fonction de pondération qui permet d'adapter la mesure au domaine d'application considéré : on peut par exemple décider que partager la caractéristique "être une poutre" est plus important que partager une caractéristique de forme. Enfin, notons que l'appariement ne doit pas nécessairement être univalent : il doit être possible d'associer un composant à plusieurs autres. Ainsi, le mur 5 de l'objet 2 joue seul le rôle des deux murs e et f de l'objet 1.

Organisation de l'article

En partie 2, nous définissons le formalisme des graphes étiquetés et nous illustrons leur expressivité sur les objets de la figure 1. Dans la partie 3, nous présentons notre mesure de similarité générique basée sur l'appariement des sommets des deux graphes. La partie 4 étudie la

complexité de notre problème et propose deux algorithmes : un algorithme basé sur une exploration complète de type "séparation et évaluation", et un algorithme glouton incomplet. Après la présentation de quelques résultats expérimentaux, nous concluons en montrant la généricité de notre approche par rapport aux autres méthodes de comparaison de graphes puis nous discutons quelques perspectives futures.

2 Graphes étiquetés

Définitions et notations : Un *graphe étiqueté* est un graphe orienté auquel on a associé à chacun des sommets et chacun des arcs un ensemble non vide d'étiquettes. Plus formellement, étant donné L_V un ensemble fini d'étiquettes de sommets et L_E un ensemble fini d'étiquettes d'arcs, un graphe étiqueté est défini par un triplet $G = \langle V, r_V, r_E \rangle$ tel que :

- V est un ensemble fini de sommets,
- $r_V \subseteq V \times L_V$ est la relation associant sommets et étiquettes, i.e., r_V est l'ensemble des couples (v_i, l) tels que le sommet v_i est étiqueté par l,
- $r_E \subseteq V \times V \times L_E$ est la relation associant arcs et étiquettes, i.e., r_E est l'ensemble des triplets (v_i, v_j, l) tels que l'arc (v_i, v_j) est étiqueté par l. L'ensemble E des arcs peut être défini par $E = \{(v_i, v_j) | \exists l, (v_i, v_j, l) \in r_E\}$.

Nous appelons les tuples de r_V les caractéristiques des sommets de G et ceux de r_E les caractéristiques des arcs de G. On définit le descripteur d'un graphe $G = \langle V, r_V, r_E \rangle$ comme l'ensemble de toutes ses caractéristiques de sommet et d'arc : $descr(G) = r_V \cup r_E$. Cet ensemble décrit entièrement le graphe et sera utilisé pour mesurer la similarité de deux graphes.

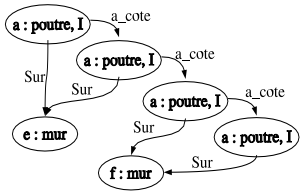
Un *problème de similarité* est défini par les deux ensembles L_V et L_E des étiquettes de sommets et d'arcs et par deux graphes $G_1 = \langle V_1, r_{V_1}, r_{E_1} \rangle$ et $G_2 = \langle V_2, r_{V_2}, r_{E_2} \rangle$ n'ayant pas de sommet en commun (i.e., $V_1 \cap V_2 = \emptyset$).

Exemple : Considérons les deux objets de la figure 1. Afin de représenter ces objets avec des graphes étiquetés, définissons tout d'abord les ensembles d'étiquettes des sommets et des arcs :

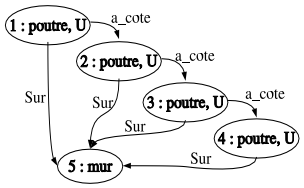
$$\begin{aligned} L_V &= \{ poutre, I, U, mur \} \\ L_E &= \{ sur, a\_cote \} \end{aligned}$$

Etant donnés ces ensembles d'étiquettes, les deux objets peuvent alors être représentés par les deux graphes étiquetés de la figure 2. Nous constatons que les sommets a, b, c et d possèdent les deux étiquettes $poutre$ et I tandis que les sommets 1, 2, 3 et 4 possèdent les étiquettes $poutre$ et U. Cela nous permet d'exprimer le fait que les objets correspondant à ces sommets partagent la propriété d'être une poutre bien qu'ils aient une forme différente.

De façon plus générale, le fait que les arcs et les sommets puissent avoir plusieurs étiquettes peut être utilisé pour exprimer des relations d'héritage en termes d'inclusion des ensembles d'étiquettes. La similarité de deux sommets ou arcs sera alors calculée par rapport à leurs étiquettes communes, correspondant à leurs ancêtres communs dans la hiérarchie d'héritage.



$$
\begin{aligned}
G_1 = \langle\, V_1 &= \{\, a, b, c, d, e, f \,\} \\
r_{V_1} &= \{\, (a, poutre), (b, poutre), (c, poutre), (d, poutre), \\
&\quad\ (a, I), (b, I), (c, I), (d, I), \\
&\quad\ (e, mur), (f, mur) \,\} \\
r_{E_1} &= \{\, (a, b, a\_cote), (b, c, a\_cote), (c, d, a\_cote), \\
&\quad\ (a, e, sur), (b, e, sur), (c, f, sur), (d, f, sur) \,\} \,\rangle
\end{aligned}
$$



$$
\begin{aligned}
G_2 = \langle\, V_2 &= \{\, 1, 2, 3, 4, 5 \,\} \\
r_{V_2} &= \{\, (1, poutre), (2, poutre), (3, poutre), (4, poutre), \\
&\quad\ (1, U), (2, U), (3, U), (4, U), \\
&\quad\ (5, mur) \,\} \\
r_{E_2} &= \{\, (1, 2, a\_cote), (2, 3, a\_cote), (3, 4, a\_cote), \\
&\quad\ (1, 5, sur), (2, 5, sur), (3, 5, sur), (4, 5, sur) \,\} \,\rangle
\end{aligned}
$$

FIG. 2 – Les graphes étiquetés $G1$ et $G2$ décrivant les objets 1 et 2 de la figure 1 : à gauche, leur représentation graphique, à droite leur représentation dans le formalisme présenté en 2.

3 Une mesure de similarité de graphes étiquetés

3.1 Appariement de graphes étiquetés

Pour mesurer la similarité de deux graphes étiquetés, il est nécessaire de mettre leurs sommets en correspondance. Plus formellement, un appariement entre deux graphes $G_1 = \langle V_1, r_{V_1}, r_{E_1} \rangle$ et $G_2 = \langle V_2, r_{V_2}, r_{E_2} \rangle$, tels que $V_1 \cap V_2 = \emptyset$, est une relation $m \subseteq V_1 \times V_2$. Un tel appariement associe à chaque sommet d'un graphe 0, 1 ou plusieurs sommets de l'autre graphe. Par extension, nous noterons $m(v)$ l'ensemble des sommets associés au sommet v dans l'appariement m, *i.e.*,

$$
\begin{aligned}
\forall v_1 \in V_1, \quad m(v_1) &\doteq \{v_2 \in V_2 |\ (v_1, v_2) \in m\} \\
\forall v_2 \in V_2, \quad m(v_2) &\doteq \{v_1 \in V_1 |\ (v_1, v_2) \in m\}
\end{aligned}
$$

Exemple : Pour les graphes étiquetés de la figure 2, nous pouvons par exemple définir les deux appariements suivants :

$$
\begin{aligned}
m_A = &\quad \{(a,1), (b,2), (c,3), (d,4), (e,5), (f,5)\} \\
m_B = &\quad \{(a,1), (b,2), (a,3), (b,4)), (e,5)\}
\end{aligned}
$$

L'appariement m_A met respectivement en correspondance les sommets a, b, c, et d aux sommets 1, 2, 3 et 4 ainsi que les deux sommets e et f au sommet 5 (*i.e.*, $m_A(5) = \{e, f\}$). L'appariement m_B associe les sommets 1 et 3 au sommet a, les sommets 2 et 4 au sommet b et le sommet e au sommet 5. Dans cet appariement, aucun sommet n'a été mis en relation avec f (*i.e.*, $m_B(f) = \emptyset$).

3.2 Similarité par rapport à un appariement

Afin de mesurer la similarité entre deux objets, il est courant et assez intuitif de comparer la quantité de caractéristiques communes à ces deux objets par rapport à l'ensemble de toutes leurs caractéristiques [Lin, 1998]. [Tversky, 1977] démontre la plausibilité cognitive de cette intuition

et définit un modèle mathématique : la similarité de deux objets a et b, décrits respectivement par les deux ensembles A et B de leurs caractéristiques, est définie par la formule

$$sim_{\text{Tversky}}(a,b) = \frac{f(A \cap B)}{f(A \cup B)}$$

où f est une fonction positive croissante et monotone par rapport à l'inclusion, permettant de pondérer l'importance des différentes caractéristiques.

Nous proposons ici d'adapter la formule de Tversky pour évaluer la similarité de graphes étiquetés tels que définis en partie 2. Un graphe étiqueté G est décrit par l'ensemble $descr(G)$ des caractéristiques de ses sommets et arcs. Ainsi, la similarité de deux graphes $G_1 = \langle V_1, r_{V_1}, r_{E_1} \rangle$ et $G_2 = \langle V_2, r_{V_2}, r_{E_2} \rangle$ dépend des caractéristiques communes des ensembles $descr(G_1)$ et $descr(G_2)$. Cependant, étant donné que $V_1 \cap V_2 = \emptyset$, l'intersection de ces deux ensembles de descripteurs sera toujours vide. Nous devons donc calculer cette intersection par rapport à un appariement m associant les sommets de G_1 à ceux de G_2, i.e.,

$$\begin{aligned} descr(G_1) \sqcap_m descr(G_2) \quad \doteq \quad & \{(v,l) \in r_{V_1} |\ \exists v' \in m(v),\ (v',l) \in r_{V_2}\} \\ \cup \quad & \{(v,l) \in r_{V_2} |\ \exists v' \in m(v),\ (v',l) \in r_{V_1}\} \\ \cup \quad & \{(v_i,v_j,l) \in r_{E_1} |\ \exists v'_i \in m(v_i), \exists v'_j \in m(v_j)\ (v'_i,v'_j,l) \in r_{E_2}\} \\ \cup \quad & \{(v_i,v_j,l) \in r_{E_2} |\ \exists v'_i \in m(v_i), \exists v'_j \in m(v_j)\ (v'_i,v'_j,l) \in r_{E_1}\} \end{aligned}$$

Cet ensemble contient toutes les caractéristiques des éléments de G_1 (resp. G_2) retrouvées au moins une fois dans G_2 (resp. G_1) via l'appariement m.

Etant donnée cette définition des caractéristiques communes de G_1 et G_2, nous pouvons utiliser la formule de Tversky pour définir la similarité entre G_1 et G_2 par rapport à l'appariement m :

$$sim1_m(G_1,G_2) \quad = \quad \frac{f(descr(G_1) \sqcap_m descr(G_2))}{f(descr(G_1) \cup descr(G_2))} \tag{1}$$

Exemple : Considérons les deux graphes étiquetés et leurs ensembles de descripteurs de la figure 2. L'intersection de leurs caractéristiques communes par rapport aux appariements m_A et m_B proposés en partie 3.1, sont :

$$\begin{aligned} descr(G_1) \sqcap_{m_A} descr(G_2) \quad &= \ descr(G_1) \cup descr(G_2) \\ &\quad - \{\ (a,I),(b,I),(c,I),(d,I),(1,U),(2,U),(3,U),(4,U)\ \} \\ descr(G_1) \sqcap_{m_B} descr(G_2) \quad &= \{(a,poutre),(b,poutre),(e,mur), \\ &\quad (1,poutre),(2,poutre),(3,poutre),(4,poutre),(5,mur), \\ &\quad (a,b,a\_cote),(1,2,a\_cote),(3,4,a\_cote),(a,e,sur), \\ &\quad (b,e,sur),(1,5,sur),(2,5,sur),(3,5,sur),(4,5,sur)\ \} \end{aligned}$$

3.3 Prise en compte des sommets éclatés

La définition de la similarité $sim1_m$ de l'équation (1) n'est pas entièrement satisfaisante dans le sens où elle ne tient pas compte du fait qu'un sommet peut être "éclaté", i.e., apparié à plusieurs sommets. Par exemple, dans les deux graphes étiquetés de la figure 2, l'appariement $m_A = \{(a,1),(b,2),(c,3),(d,4),(e,5),(f,5)\}$ apparie le sommet 5 avec les deux sommets e et f. Il s'agit de prendre en compte ces éclatements de sommets dans notre mesure. Pour cela, nous

introduisons la fonction $splits$ qui retourne l'ensemble des sommets "éclatés" avec l'ensemble des sommets auxquels ils sont reliés :

$$splits(m) \doteq \{(v, s_v) |\ v \in V_1 \cup V_2,\ s_v = m(v),\ |m(v)| \geq 2\}$$

et nous définissons un opérateur d'inclusion sur ces ensembles de sommets éclatés :

$$splits(m1) \sqsubseteq splits(m2) \Leftrightarrow \forall (v, s_v) \in splits(m_1),\ \exists (v, s'_v) \in splits(m_2),\ s_v \subseteq s'_v$$

Par exemple, les appariements de la partie 3.1 génèrent les splits suivants : $splits(m_A) = \{\ (5, \{e, f\})\ \}$ et $splits(m_B) = \{\ (a, \{1, 3\}),\ (b, \{2, 4\})\ \}$.

Nous pouvons alors modifier l'équation (1) afin qu'elle prenne en compte les éclatements de sommets :

$$sim_m(G_1, G_2) \quad = \quad \frac{f(descr(G_1) \sqcap_m descr(G_2)) - g(splits(m))}{f(descr(G_1) \cup descr(G_2))} \tag{2}$$

où la fonction g possède les mêmes propriétés que f : elle est positive et croissante par rapport à la relation d'inclusion $\sqsubseteq$. Dès lors, la similarité ne peut que décroître lorsque le nombre d'éclatements de sommets dans l'appariement m croît.

3.4 Similarité maximum de deux graphes étiquetés

Nous avons défini la similarité de deux graphes par rapport à un appariement entre leurs sommets. Maintenant, nous pouvons définir la *similarité maximum* de deux graphes G_1 et G_2 comme la plus grande similarité par rapport à tous les appariements possibles :

$$sim(G_1, G_2) \quad = \quad \max_{m \subseteq V_1 \times V_2} \frac{f(descr(G_1) \sqcap_m descr(G_2)) - g(splits(m))}{f(descr(G_1) \cup descr(G_2))} \tag{3}$$

Notons que, si le degré de similarité sim_m obtenu pour un appariement m ne contenant pas de sommet éclaté est toujours compris entre 0 et 1, sim_m peut devenir négatif lorsque beaucoup de sommets sont éclatés : le poids de la fonction $splits$, défini par g, peut alors être plus grand que le poids des caractéristiques communes, défini par f. Cependant, dans tous les cas, la similarité maximum sera toujours comprise entre 0 et 1 car la similarité obtenue par l'appariement vide est nulle.

Notons aussi que c'est le choix des fonctions f et g qui permet de déterminer le meilleur appariement. f et g seront donc choisies en fonction du domaine d'application. Ce point est discuté dans la partie 3.5.

3.5 Connaissances de similarité

La mesure de similarité présentée plus haut est générique dans le sens où elle est paramétrée par les deux fonctions f et g : ces deux fonctions définissent l'importance relative des caractéristiques les unes par rapport aux autres (*e.g.*, rendre le concept *"être une poutre"* plus important que le concept *"avoir une forme en U"*). Le choix de f et g permet donc l'introduction de connaissances de similarité propres au domaine de l'application considérée. Ces fonctions sont

souvent définies comme une somme pondérée. On introduit pour cela les trois fonctions de pondérations suivantes, définissant les poids des caractéristiques de sommets et d'arcs et les poids des éclatements :

$$\begin{aligned} poids_V &: V \times L_V \rightarrow \Re^+ \\ poids_E &: V \times V \times L_E \rightarrow \Re^+ \\ poids_g &: V \times \wp(V) \rightarrow \Re^+ \end{aligned}$$

On définit alors les fonctions f et g par :

$$f(F) = \sum_{(v,l)\in F} poids_V(v,l) + \sum_{(v_1,v_2,l)\in F} poids_E(v_1,v_2,l) \quad (4)$$

$$g(S) = \sum_{(v,s_v)\in S} poids_g(v,s_v) \quad (5)$$

Sans connaissances particulières du domaine, les fonctions de pondération $poids_V$ et $poids_E$ peuvent retourner une valeur constante, ce qui revient à considérer que toutes les caractéristiques sont également importantes. La fonction $poids_g(v, s_v)$ peut retourner une valeur proportionnelle à la cardinalité de s_v.

A contrario, si des connaissances contextuelles sont disponibles, il est possible d'associer un poids particulier à chacune des caractéristiques et à chacun des éclatements de sommets. Par exemple, le fait d'être une poutre est peut être plus important que le fait d'avoir une forme en U ; il est peut être plus génant d'éclater le composant 5 que le composant 1 ...

On peut également choisir des fonctions de pondération permettant de mesurer des similarités "assymétriques" : en assignant un poids nul à toutes les caractéristiques du graphe G_2, il est possible de mesurer à quel point G_1 est un sous-graphe de G_2.

Dans la suite de cette article, nous considérons les fonctions f et g définies dans les formules (4) et (5), fonctions qui possèdent la propriété intéressante d'être calculables de manière incrémentale.

4 Calcul de la similarité maximum

Etant donnés deux graphes étiquetés G_1 et G_2, on considère maintenant le problème de calculer leur similarité maximum, i.e., trouver l'appariement m qui maximise l'équation 2. Notons que le dénominateur $f(descr(G_1) \cup descr(G_2))$ de cette équation ne dépend pas d'un appariement : il est introduit afin de normaliser le degré de similarité entre zéro et un. Il est donc suffisant, pour déterminer la similarité maximum entre deux graphes, de trouver l'appariement m qui maximise la fonction

$$score(m) = f(descr(G_1) \sqcap_m descr(G_2)) - g(splits(m))$$

Ce problème est fortement combinatoire : il est plus général que le problème NP-complet d'isomorphisme de sous-graphes. En effet, pour déterminer si B est un sous graphe de A, il suffit de définir deux graphes étiquetés G_A et G_B tels que chaque arc et chaque sommet aient la même étiquette, de choisir g telle que $g(\emptyset) = 0$ et $\forall s \neq \emptyset, g(s) = 1$ et f qui associe un poids de 1 (resp. 0) à toutes les caractéristiques de G_B (resp. G_A). Si la similarité maximum entre G_A et G_B est égale à 1, alors B est un sous-graphe de A.

Ce problème peut être formulé sous la forme d'un problème de satisfaction de contraintes valué (VCSP) [Shiex *et al.*, 1995] : on associe une variable à chaque sommet de G_1, son domaine étant l'ensemble des parties de l'ensemble des sommets de G_2 ; les contraintes expriment le fait que chaque caractéristique de G_1 ou de G_2 doit appartenir à l'ensemble des caractéristiques communes aux deux graphes $descr(G_1) \sqcap_m descr(G_2)$, et qu'aucun sommet ne doit être éclaté ; la valuation des contraintes est définie par les fonctions de pondération. Cependant, dans cette modélisation, le domaine de chaque variable comporte $2^{|V_2|}$ éléments de sorte que la recherche risque fort de ne pas terminer en un temps "raisonnable".

Dans cette partie, on décrit tout d'abord une autre approche complète pour la résolution de notre problème, basée sur une exploration par "séparation et évaluation". On décrit ensuite un algorithme incomplet utilisant une heuristique gloutonne. On commente enfin quelques résultats expérimentaux.

4.1 Recherche complète par "séparation et évaluation"

L'espace de recherche du problème de la similarité maximum entre deux graphes étiquetés est composé de tous les appariements possibles de ces deux graphes, c'est à dire de tous les sous-ensembles de $V_1 \times V_2$. Il existe alors $2^{|V_1|*|V_2|}$ états possibles dans cet espace de recherche. Celui-ci peut être structuré en treillis par rapport à la relation d'inclusion : on note n_m un nœud du treillis associé à un appariement m ; chaque nœud n_m a pour fils l'ensemble des nœuds $n_{m'}$ tels que m' est un appariement contenant un couple de sommets de plus que m.

Ce treillis peut être exploré de façon complète grâce à une approche de type "séparation et évaluation" (branch and bound). Une telle approche peut être envisagée s'il est possible de trouver une bonne fonction d'évaluation. Une telle fonction permet, à partir d'un nœud n_m du treillis (correspondant à un appariement m) de trouver une borne maximale de la qualité de tous les nœuds $n_{m'}$ descendants de n_m (correspondant à des appariements m' tels que $m \subset m'$). Si cette borne maximale est inférieure à la qualité du meilleur appariement connu, alors il ne sera pas nécessaire de développer le treillis à partir du nœud n_m et l'espace de recherche à explorer deviendra plus restreint. Malheureusement, la fonction $score$ n'est pas monotone par rapport à l'inclusion : le score d'un appariement peut diminuer ou augmenter quand on lui ajoute un nouveau couple de sommets. Cela vient du fait que la fonction $score$ est définie comme une différence de deux fonctions toutes deux croissantes lors de l'ajout d'un nouveau couple. Plus précisemment, nous pouvons voir que pour tous les appariements m et m' tels que $m \subseteq m'$:

$$\begin{aligned} descr(G_1) \sqcap_m descr(G_2) &\subseteq descr(G_1) \sqcap_{m'} descr(G_2) \\ \text{et} \qquad splits(m) &\sqsubseteq splits(m') \end{aligned}$$

donc, comme f et g sont monotones croissantes,

$$\begin{aligned} f(descr(G_1) \sqcap_m descr(G_2)) &\leq f(descr(G_1) \sqcap_{m'} descr(G_2)) \\ \text{et} \qquad g(splits(m)) &\leq g(splits(m')) \end{aligned}$$

Nous pouvons cependant utiliser le fait que l'intersection des ensembles de caractéristiques des graphes est bornée par l'ensemble de toutes les caractéristiques de ces graphes pour déterminer une fonction d'évaluation. Pour tous les appariements m, il est possible de montrer que

$$descr(G_1) \sqcap_m descr(G_2) \subseteq descr(G_1) \cup descr(G_2)$$

et, comme f et g sont monotones, pour tous les appariements m' tels que $m' \supseteq m$,

$$\begin{aligned} score_{m'}(G_1, G_2) &= f(descr(G_1) \sqcap_{m'} descr(G_2)) - g(splits(m')) \\ &\leq f(descr(G_1) \cup descr(G_2)) - g(splits(m)) \end{aligned}$$

Afin de réduire l'espace de recherche, il est possible d'utiliser quelques règles *ad-hoc*. Par exemple, prendre en compte le fait qu'un couple de sommets peut devenir "inutile". A partir d'un nœud n_m du treillis, un couple c de sommets devient inutile quand le poids de l'ensemble des caractéristiques non encore satisfaites et communes à ses deux sommets est inférieur au poids des splits générés par l'insertion de c dans l'appariement m. L'ensemble $useless_m$ des couples inutiles à partir d'un appariement m est alors défini par :

$$useless_m \doteq \{c \in (V_1 \times V_2 - m), \forall m' \subseteq (V_1 \times V_2), score(m \cup m') > score(m \cup m' \cup \{c\})\}$$

Il ne sera alors pas nécessaire de développer à partir de n_m les nœuds du treillis correspondant à des appariements contenant un couple inutile c, le score de ces appariements étant nécessairement inférieur au score des mêmes appariements privés de c. Notons qu'un couple de sommets c peut être inutile pour tous les appariements ($c \in useless_\emptyset$) : cela survient quand ses sommets ne partagent aucune caractéristique.

Certaines caractéristiques d d'un graphe peuvent alors être considérées "irrécupérables" à partir d'un nœud n_m du treillis : quand tous les couples permettant l'insertion de d dans l'ensemble des caractéristiques communes aux deux graphes sont inutiles, alors d est irrécupérable.

$$\begin{aligned} irrecup_m(G_1, G_2) \doteq \; & \{d \in (descr(G_1) \cup descr(G_2)) - descr(G_1) \sqcap_m descr(G_2) \text{ tel que} \\ & \forall c \in V_1 \times V_2, d \in descr(G_1) \sqcap_{m \cup \{c\}} descr(G_2), c \in useless_m\} \end{aligned}$$

Un appariement $m_1 \supseteq m$ tel que $\exists d \in (irrec_m(G_1, G_2) \cap descr(G_1) \sqcap_{m_1} descr(G_2))$ aura alors toujours un score inférieur à celui du meilleur appariement $m_2 \supseteq m$ tel que $d \notin descr(G_1) \sqcap_{m_2} descr(G_2)$. Il est donc maintenant possible de dire que, pour tous les appariements $m' \supseteq m$:

$$\begin{aligned} score_{m'}(G_1, G_2) &= f(descr(G_1) \sqcap_{m'} descr(G_2)) - g(splits(m')) \\ &\leq f(descr(G_1) \cup descr(G_2)) - g(splits(m)) - f(irrec_m(G_1, G_2)) \end{aligned}$$

Cette formule définit une borne maximum du score de tous les appariements $m' \supseteq m$. Par conséquent, si le nœud n_m du treillis associé à un appariement m est tel que $f(descr(G_1) \cup descr(G_2)) - g(splits(m)) - f(irrecup_m(G_1, G_2))$ est plus petit ou égal au score du meilleur appariement connu, il ne sera pas nécessaire de développer le treillis à partir de ce nœud n_m ; tous les nœuds correspondants à des appariements qui sont des sur-ensembles de m (les descendants de m dans le treillis) ne seront pas explorés et la recherche sera accélérée.

L'efficacité de cette fonction d'évaluation pour réduire l'espace de recherche à explorer dépend du poids de la fonction g par rapport à celui de la fonction f : plus le poids associé aux splits est élevé et plus le nombre de nœuds éliminés sera élevé. Cette fonction d'évaluation est générique et peut être utilisée avec toutes sortes de fonctions f et g (sous réserves que celles-ci soient croissantes). Pour des cas plus spécifiques, il est possible de trouver une fonction d'évaluation plus efficace.

Par ailleurs, la faisabilité d'une approche de résolution complète de type "branch and bound" dépend également de l'ordre dans lequel les appariements sont explorés : une bonne heuristique

d'ordonnancement permet de trouver rapidement un bon appariement. Plus tôt un appariement de score élevé est connu et plus élevé sera le nombre d'appariements éliminés durant la recherche. En partie suivante, nous décrivons un algorithme glouton capable de trouver rapidement un "bon" appariement.

4.2 Algorithme glouton

La figure 3 présente un algorithme glouton utilisant une heuristique simple pour construire rapidement un appariement m de "bonne" qualité.

Déroulement de l'algorithme

L'algorithme démarre avec un appariement m vide et ajoute, à chaque itération, un couple de sommets dans m selon un principe glouton. On choisit le couple à insérer parmi l'ensemble $cand$ des couples qui maximisent la fonction $score$. Cet ensemble comportant généralement plus d'un candidat, on départage les ex-æquo en anticipant le potentiel des candidats : on calcule l'ensemble $look\_ahead$ des propriétés d'arcs (départs ou arrivées d'arcs) partagées par les deux sommets du couple candidat et n'appartenant pas à $descr(G_1) \sqcap_m descr(G_2)$, et on choisit aléatoirement le prochain sommet à insérer parmi ceux qui maximisent $f(look\_ahead)$.

fonction Glouton(G_1=$\langle V_1, r_{V_1}, r_{E_1}\rangle$, G_2=$\langle V_2, r_{V_2}, r_{E_2}\rangle$)
retourne un appariement $m \subseteq V_1 \times V_2$
  $m \leftarrow \emptyset$
  $best_m \leftarrow \emptyset$
  itérer
    $cand \leftarrow \{(u_1,u_2) \in V_1 \times V_2 - m \mid score(m \cup \{(u_1,u_2)\})$ est maximal$\}$
    $cand' \leftarrow \{(u_1,u_2) \in cand \mid f(look\_ahead(u_1,u_2))$ est maximal$\}$
      où $look\_ahead(u_1,u_2) = \{(u_1,v_1,l) \in r_{E_1} \mid \exists v_2 \in V_2, (u_2,v_2,l) \in r_{E_2}$
        $\cup \{(u_2,v_2,l) \in r_{E_2} \mid \exists v_1 \in V_1, (u_1,v_1,l) \in r_{E_1}$
        $\cup \{(v_1,u_1,l) \in r_{E_1} \mid \exists v_2 \in V_2, (v_2,u_2,l) \in r_{E_2}$
        $\cup \{(v_2,u_2,l) \in r_{E_2} \mid \exists v_1 \in V_1, (v_1,u_1,l) \in r_{E_1}$
        $- descr(G_1) \sqcap_{m\{(u_1,u_2)\}} descr(G_2)$
  sortir si $\forall (u_1,u_2) \in cand'$, $score(m \cup \{(u_1,u_2)\}) \leq score(m)$
      et $look\_ahead(u_1,u_2) = \emptyset$
    choisir aléatoirement un couple (u_1,u_2) dans $cand'$
    $m \leftarrow m \cup \{(u_1,u_2)\}$
    si $score(m) > score(best_m)$ **alors** $best_m \leftarrow m$ **finsi**
  fin itérer
  retourner m

FIG. 3 – Recherche gloutonne d'un appariement

Cet algorithme glouton s'arrête lorsque plus aucun couple n'améliore directement la fonction score ni n'a de caractéristique potentielle dans l'ensemble $look\_ahead$. Notons que le score de l'appariement m en construction peut décroître d'une itération à l'autre (quand le couple qui est ajouté à m introduit plus d'éclatements que de nouvelles caractéristiques communes, mais a des caractéristiques potentielles dans l'ensemble $look\_ahead$, de sorte que l'on espère que la fonction score sera incrémentée à l'itération suivante). Ainsi, l'algorithme retourne le meilleur appariement trouvé ($best_m$) depuis le début de l'exécution.

Exemple d'exécution de l'algorithme

Prenons l'exemple des deux graphes de la figure 2 et définissons les fonctions f et g comme les fonctions de cardinalité. A la première itération, lorsque l'appariement courant est vide, l'ensemble $cand$ des couples qui font le plus croître la fonction score contient tous les couples formés de deux sommets qui possèdent une étiquette commune, $cand = \{a, b, c, d\} \times \{1, 2, 3, 4\} \cup \{e, f\} \times \{5\}$.

Parmi ces candidats, $(b, 2)$ et $(c, 3)$ ont $3 + 3$ caractéristiques d'arcs communes, de sorte que $f(look\_ahead(b, 2)) = f(look\_ahead(c, 3)) = 6$. Les couples $(e, 5)$ et $(f, 5)$ ont eux aussi 6 propriétés d'arcs en commun. Les autres candidats ont tous un nombre plus petit de caractéristiques communes. L'algorithme glouton choisit alors aléatoirement un couple parmi l'ensemble : $cand' = \{(b, 2), (c, 3), (e, 5), (f, 5)\}$.

Supposons maintenant que le couple $(e, 5)$ soit sélectionné. A la seconde itération, $cand$ contient $(a, 1)$ et $(b, 2)$ alors que $cand'$ ne contient que $(b, 2)$ car b partage plus de caractéristiques communes avec 2 que a avec 1. Le prochain couple à être inséré dans m est donc $(b, 2)$. A la troisième itération, $cand$ ne contient que $(a, 1)$ qui augmente le score de $3 + 3$, donc $(a, 1)$ est inséré dans m et ainsi de suite... Sur cet exemple, l'algorithme glouton retourne à chaque fois l'appariement $m_A = \{(a, 1), (b, 2), (c, 3), (d, 4), (e, 5), (f, 5)\}$, m_A étant effectivement le meilleur appariement du point de vue de la fonction $score$ comme d'un point de vue intuitif.

Complexité de l'algorithme

Cet algorithme glouton est de complexité polynomiale en $\mathcal{O}((|V_1| \times |V_2|)^2)$, sous réserve que le calcul des fonctions f et g s'effectue en temps linéaire. En contrepartie de cette faible complexité, cet algorithme n'est pas complet et ne garantit pas l'optimalité de son résultat. N'étant pas déterministe, cet algorithme peut être exécuté plusieurs fois afin de conserver le meilleur appariement trouvé.

Dans certains contextes d'application de notre mesure de similarité, il sera plus pertinent de proposer à l'utilisateur un ensemble d'appariements de bonne qualité plutôt qu'un seul appariement obtenant une similarité maximale. En effet, l'appariement de qualité maximale est extrêmement dépendant de la fonction d'évaluation choisie, or, selon l'application, cette fonction d'évaluation sera difficile à exprimer et pourra alors parfois amener à des erreurs. L'algorithme glouton que nous proposons est capable de générer rapidement un ensemble de bons appariements que l'utilisateur peut alors trier.

Structures de données utilisées

Afin d'améliorer les temps de calcul pour le choix du meilleur couple de sommets parmi tous les couples possibles, on peut rendre les calculs de la fonction f et des ensembles $look\_ahead$ incrémentaux. On a utilisé pour cela des structures de données adaptées créées lors d'une phase d'initialisation de l'algorithme.

Pour chacune des caractéristiques de sommets (resp. d'arcs), sont mémorisés l'ensemble des couples de sommets (resp. l'ensemble des paires de couples de sommets) permettant l'insertion de cette caractéristique dans l'ensemble $descr(G_1) \sqcap_m descr(G_2)$ (et de l'ensemble $look\_ahead$ pour les caractéristiques d'arcs).

Pour chacun des couples de sommets, sont mémorisés l'ensemble des caractéristiques des sommets, des arcs sortants et des arcs entrants que ces sommets ont en commun. Toutes ces données sont mémorisées dans des listes accessibles en temps constant grâce à l'utilisation de tables de hashage. Elles sont indépendantes de l'appariement utilisé et peuvent donc être calculées dans une phase d'initialisation de l'algorithme.

Parallèlement à ces structures de données, on utilise des compteurs, permettant de déterminer si une caractéristique d'un des deux graphes est retrouvée dans l'autre graphe par l'appariement courant. D'autres compteurs mesurent pour chacun des couples de sommets l'accroissement de $f(descr(G_1) \sqcap_m descr(G_2))$ et de $f(look\_ahead)$ obtenu si le couple est inséré dans l'appariement. Ces deux types de compteurs évoluent à chaque insertion d'un couple de sommets dans l'appariement courant et sont mis à jour en un temps proportionnel au nombre de caractéristiques que les sommets du couple inséré ont en commun multiplié par le nombre de couples de sommets qui ont ces caractéristiques en commun.

La valeur de la fonction g est calculée de façon incrémentale : si l'insertion d'un couple génère un éclatement (resp. deux éclatements), on ajoute le poids d'un éclatement (resp. celui de deux éclatements) à la valeur de la fonction g.

4.3 Résultats expérimentaux

Les deux algorithmes présentés (l'algorithme complet et le glouton) ont été implémentés en C++. Les résultats obtenus par l'algorithme complet ont montré qu'une exploration par "séparation et évaluation" n'est envisageable que pour de tous petits graphes (une dizaines de sommets dans le cas général) et que son efficacité dépend énormément du poids des splits par rapport aux poids attribués aux caractéristiques des graphes.

L'algorithme glouton est bien plus efficace : l'appariement de graphes de plus de 100 sommets est tout à fait envisageable. Quelques tests ont été réalisés, ces tests faisant varier le type du problème à résoudre (meilleur appariement de graphes différents, isomorphisme de graphes, isomorphisme de sous-graphes), la connectivité des graphes (de un arc par sommet à une dizaine), le degré de différence entre les deux graphes comparés (le nombre de caractéristiques communes aux deux graphes) et l'étiquetage des éléments des graphes (d'une étiquette commune à tous les éléments à trois étiquettes par éléments parmi une dizaine d'étiquettes possibles).

Ces tests ont montré que la difficulté du problème varie en fonction de la connectivité des graphes, du nombre moyen d'étiquettes associées à chacun des sommets et des arcs et de la taille des ensembles L_V et L_E d'étiquettes. Le problème devient difficile lorsque la connectivité est faible et que tous les sommets des graphes ont approximativement le même nombre d'arcs entrants et le même nombre d'arcs sortants. A contrario, le problème devient d'autant plus simple que les sommets diffèrent les uns des autres : plus le nombre moyen d'étiquettes par élément du graphe est faible par rapport à la taille des ensembles de toutes les étiquettes et plus l'algorithme est performant. Ce phénomène s'explique par le fait que plus les sommets des graphes sont similaires et plus l'ensemble $cand'$ contient de couples de sommets : le choix parmi ces couples étant purement aléatoire, l'algorithme peut être amené à choisir un mauvais couple de sommets, l'erreur commise se répercutant alors sur les choix ultérieurs des couples de sommets.

5 Conclusion

Nous proposons dans cet article un modèle de représentation de données structurées par des graphes orientés étiquetés. Nous définissons ensuite une mesure de similarité entre de tels graphes et nous présentons deux algorithmes pour calculer cette similarité : un algorithme de recherche complète de type séparation/évaluation qui ne semble pas utilisable dans le cas général et un algorithme glouton qui, s'il ne garantit pas l'optimalité de la solution, permet d'approximer rapidement la similarité de graphes possèdant une centaine de sommets.

5.1 Un modèle générique

Les graphes permettent la représentation de données structurées et sont utilisés dans de nombreux domaines. Beaucoup de problèmes relatifs aux graphes ont déjà été étudiés et la comparaison de graphes est un problème courant qui peut prendre des formes diverses : deux graphes sont-ils identiques (problème de l'isomorphisme de graphe), l'un des graphes est-il plus général que l'autre (problème de la subsumption), à quel point sont-ils différents (quantitativement et qualitativement) ?

La plupart de ces problèmes de comparaison de graphes peuvent être transformés très simplement en un problème de mesure de similarité telle que nous l'avons définie. Le problème d'isomorphisme est équivalent à la recherche d'un appariement parfait entre deux graphes (similarité = 1) où les fonctions f et g sont strictement positives pour toutes les valeurs différentes de l'ensemble vide. Le problème de l'isomorphisme de sous-graphe peut, comme discuté en partie 4, être lui aussi transformé en un problème de mesure de similarité. Le problème de k-partitionnement, consistant à rechercher un ensemble de partitions d'un graphe $G(V, A)$ contenant le même nombre de sommets et tel que le poids des arcs allant d'une partition vers une autre soit minimal, peut être résolu par une recherche de similarité : il suffit de rechercher la similarité maximale entre le graphe G et un graphe de k sommets ayant chacun un arc sortant pointant vers eux même, en définissant la fonction f comme la fonction de cardinalité et la fonction g comme la fonction associant une valeur nulle à tous les ensembles de n/k sommets et une valeur infinie à tous les autres.

D'autres travaux ont étudié des problèmes de comparaison "approximative" de graphes : dans le cadre de la représentation de connaissances, de la reconnaissance de formes en imagerie ou de l'approximation de requêtes en bases de données semi-structurées, la recherche d'éléments strictement similaires est parfois trop contraignante. Des mesures de subsumption approximatives [Bisson, 1995; Valtchev, 1999] ont été proposées pour résoudre ce problème. Dans le cadre de la reconnaissance d'image, [Mulhem *et al.*, 2001] propose un algorithme de *projection floue* en étendant le formalisme des graphes conceptuels. [Bunke, 1999] traite de l'isomorphisme à tolérance d'erreurs, en calculant la similarité à partir du nombre de transformations nécessaires pour rendre les graphes isomorphes. Cependant, aucune de ces approches ne prend en compte toutes les caractéristiques de notre modèle, *e.g.*, le multi-étiquetage des arcs et des sommets, la possibilité d'éclater les sommets ou d'introduire des connaissances de similarité dépendantes du contexte à travers les fonctions f et g.

Nous mentionnerons aussi le récent travail de [Petrovic *et al.*, 2002] sur la recherche de graphes similaires dans le cadre du RàPC. Leur approche, bien qu'assez proche de la notre (la similarité de deux graphes est exprimée en fonction du meilleur appariement entre les sommets de ces graphes), est moins générique : les éclatements de sommets ne sont pas autorisés, les éléments n'ont qu'une étiquette et la fonction f est prédéfinie en tant que la fonction cardinalité. Ils ont cependant implémentés un algorithme efficace de recherche Tabou.

Enfin, [Guéhéneuc et Jussien, 2001], propose de comparer des diagrammes de classes en utilisant la programmation par contraintes avec explications afin d'identifier des patrons de conception (ou des formes dégradées de ceux-ci) dans du code orienté-objet. La méthode utilisée dans [Guéhéneuc et Jussien, 2001], comme notre méthode, propose sous forme d'*explications* des informations relatives aux différences existantes entre le code source et le patron de conception detecté. Les auteurs montrent alors comment ces explications permettent d'aider le concepteur à améliorer son code.

5.2 Recherche taboue réactive

Une méthode de recherche locale taboue réactive à été développée très récemment. Cette méthode s'inspire de l'algorithme proposé dans [Battiti et Protasi, 2001] : à partir d'un bon appariement (obtenu par la recherche gloutonne), en appliquant une suite de *mouvements* sur cet appariement (l'ajout ou le retrait d'un couple de sommets), l'espace de recherche est exploré de proche en proche. A chaque itération, le mouvement générant le meilleur appariement est selectionné et ce même si ce mouvement dégrade la solution courante. Afin de diversifier la recherche on utilise une liste taboue : le mouvement inverse à un mouvement récemment appliqué ne sera pas autorisé pendant un certain nombre d'itérations. Cette méthode est réactive : lorsque l'algorithme produit des appariements déjà explorés, la longueur de la liste taboue est augmentée ce qui entraîne une diversification de la recherche. A contrario, quand la diversification est suffisante, la longueur de la liste taboue est réduite afin de permettre une exploration plus fine du voisinage de l'appariement courant.

Les premiers résultats expérimentaux montrent que ce type de recherche donne de bien meilleurs résultats que l'algorithme glouton, permettant d'aborder des graphes de tailles supérieures et d'accélérer la recherche de solutions.

5.3 Travaux futurs

La généricité de notre modèle de mesure de similarité rend le problème de la recherche de la similarité maximum très difficile. Nos efforts futurs se dirigeront vers deux directions.

Tout d'abord, nous aimerions essayer de nouvelles heuristiques de recherche : l'anticipation (*look_ahead*) dans l'algorithme glouton pourrait être approfondie afin de trouver un bon équilibre entre la qualité des résultats obtenus et le temps d'exécution nécessaire.

Dans le cas où les graphes étudiés ont certaines propriétés particulières, des heuristiques *ad hoc* pourraient être utilisées. Nous prévoyons d'étudier ces heuristiques en vue de leur intégration dans des applications spécifiques. L'adaptation des fonctions f et g pour ces applications sera de même nécessaire.

Références

[Aamodt et Plaza, 1994] Agnar Aamodt et Enric Plaza. Case-Based Reasoning : Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1) :39–59, 1994.

[Battiti et Protasi, 2001] Roberto Battiti et Marco Protasi. Reactive local search for the maximum clique problem. *Algorithmica*, 29 :610–637, 2001.

[Bisson, 1995] Gilles Bisson. *Why and How to Define a Similarity Measure for Object Based Representation Systems*, pages 236–246. IOS Press, Amsterdam (NL), 1995.

[Bunke, 1999] Horst Bunke. Error Correcting Graph Matching : On the Influence of the Underlying Cost Function. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 21(9) :917–922, 1999.

[Champin, 2002] Pierre-Antoine Champin. *Modéliser l'expérience pour en assister la réutilisation : de la Conception Assistée par Ordinateur au Web Sémantique*. Thèse de doctorat en informatique, Université Claude Bernard, Lyon (FR), 2002.

[Diday, 1982] Edwin Diday. *Éléments d'analyse de données*. Dunod, Paris (FR), 1982.

[Guéhéneuc et Jussien, 2001] Yann-Gaël Guéhéneuc et Narendra Jussien. Quelques explications pour les patrons – Une application de la PPC avec explications pour l'identification de patrons de conception. Dans Bertrand Neveu, editor, *actes des 7e JNPC*, pages 111–122. ONERA, juin 2001.

[Lieber et Napoli, 1996] Jean Lieber et Amedeo Napoli. Using Classification in Case-Based Planning. Dans *proceedings of ECAI 96*, pages 132–136, 1996.

[Lin, 1998] Dekang Lin. An Information-Theoretic Definition of Similarity. Dans *proceedings of ICML 1998, Fifteenth International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann, 1998.

[Mulhem *et al.*, 2001] Philippe Mulhem, Wee Kheng Leow, et Yoong Keok Lee. Fuzzy Conceptual Graphs for Matching Images of Natural Scenes. Dans *proceedings of IJCAI 01*, pages 1397–1404, 2001.

[Petrovic *et al.*, 2002] Sanja Petrovic, Graham Kendall, et Yong Yang. A Tabu Search Approach for Graph-Structured Case Retrieval. Dans *proc. of STAIRS 02*, volume 78 of *Frontiers in Artificial Intelligence and Applications*, pages 55–64. IOS Press, 2002.

[Shiex *et al.*, 1995] T. Shiex, H. Fargier, et G. Verfaillie. Valued constraint satisfaction problem : hard and easy problems. Dans *Proc. of IJCAI-95*, pages 631–637, 1995.

[Tversky, 1977] Amos Tversky. Features of Similarity. *Psychological Review*, 84(4) :327–352, 1977.

[Valtchev, 1999] Petko Valtchev. *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Thèse de doctorat en informatique, Université Joseph Fourier, Grenoble (FR), 1999.