Filtrage pour l'isomorphisme de sous-graphe

Stéphane Zampelli, Yves Deville, Christine Solnon, Sébastien Sorlin, Pierre Dupont

Filtrage pour l'isomorphisme de sous-graphe

S. Zampelli[1] **Y. Deville**[1] **C. Solnon**[2] **S. Sorlin**[2] **P. Dupont**[1]

Université catholique de Louvain[1]
Department of Computing Science and Engineering, Place Sainte-Barbe 2
1348 Louvain-la-Neuve (Belgium)
{sz,yde,pdupont}@info.ucl.ac.be

LIRIS, CNRS UMR 5205, Université de Lyon I[2]
43 Bd du 11 Novembre
69622 Villeurbanne Cedex (France)
{christine.solnon,sebastien.sorlin}@liris.cnrs.fr

Résumé

On introduit ici un algorithme de filtrage dédié au problème de l'isomorphisme de sous-graphe consistant à décider s'il existe une copie d'un graphe motif dans un graphe cible. L'idée principale est d'étiqueter chaque sommet en fonction de ses relations avec les autres sommets du graphe. Cet étiquetage peut être renforcé en ajoutant des informations sur les étiquettes des sommets voisins de chaque sommet. Un tel renforcement peut être effectué itérativement jusqu'à l'obtention d'un point fixe. On définit un ordre partiel sur les étiquettes afin d'exprimer leur compatibilité pour l'isomorphisme de sous-graphe. Cet ordre partiel est utilisé pour filtrer les domaines. Les résultats expérimentaux montrent que ce filtrage permet de résoudre plus efficacement le problème de l'isomorphisme de sous-graphe que des approches dédiées, pour des instances "scale free".

1 Introduction

Les graphes sont utilisés dans de nombreuses applications pour représenter des objets structurés, e.g., des molécules, des images, ou des réseaux biologiques. Dans beaucoup de ces applications, on recherche une copie d'un graphe motif dans un graphe cible [3]. Ce problème, appelé problème d'isomorphisme de sous-graphe, est NP-complet dans le cas général.

Il existe des algorithmes dédiés à ce problème, comme par exemple [18, 4]. Cependant, ces algorithmes dédiés sont difficilement utilisables pour résoudre des problèmes plus généraux, comportant des contraintes supplémentaires, comme par exemple celui introduit dans [20].

Une alternative élégante à ces approches dédiées réside dans la Programmation Par Contraintes (PPC), qui fournit un cadre générique pour la résolution de n'importe quel problème formulé en termes de contraintes. En effet, le problème d'isomorphisme de sous-graphe peut être aisément formulé sous la forme de contraintes [15, 16]. Afin de rendre la PPC compétitive avec les approches dédiées à ce problème, [11] a introduit une contrainte globale, et un algorithme de filtrage associé, combiné avec des contraintes globales de différence. [20] a étendu ce travail au problème plus général d'isomorphisme de sous-graphe approximé, et a montré que la PPC est compétitive avec les approches dédiées. La contrainte globale de [11] est *algorithmiquement* globale [2] : elle exploite un ensemble de contraintes binaires (exprimant des contraintes d'appariement d'arêtes) de façon globale afin d'établir la consistance d'arc plus efficacement ; cependant, elle établit la même consistance que si l'on avait considéré l'ensemble original de contraintes binaires.

Contribution. On introduit ici un nouvel algorithme de filtrage pour le problème d'isomorphisme de sous-graphe qui exploite la structure globale du graphe afin d'établir une consistance partielle plus forte. Ce travail est inspiré de la procédure de raffinement de partition utilisée dans Nauty [12] et Saucy [5] pour le problème d'automorphisme de graphes : l'idée est d'étiqueter

chaque sommet par une propriété invariante, comme par exemple le degré des sommets, et d'étendre itérativement ces étiquettes en considérant les étiquettes des sommets voisins. Un étiquetage similaire est également utilisé dans [17] pour définir un algorithme de filtrage pour le problème de l'isomorphisme de graphes, l'idée étant de supprimer du domaine d'une variable associée à un sommet v tout sommet ayant une étiquette différente de celle de v. L'extension d'un tel filtrage basé sur les étiquettes au problème d'isomorphisme de sous-graphe nécessite la définition d'un ordre partiel sur les étiquettes exprimant leur compatibilité pour le problème de l'isomorphisme de sous-graphe, cet ordre partiel étant alors utilisé pour supprimer du domaine d'une variable associée à un sommet v tout sommet ayant une étiquette non compatible avec celle de v.

Organisation de l'article. On présente en 2 le problème de l'isomorphisme de sous-graphe, et sa modélisation en PPC. On décrit en 3 le cadre théorique de notre filtrage : on introduit la notion d'étiquetage, et on montre comment utiliser ces étiquetages pour filtrer les domaines ; on montre ensuite comment renforcer un étiquetage en ajoutant des informations sur les étiquettes des voisins, ce renforcement pouvant être itéré jusqu'à l'obtention d'un point fixe. On discute ensuite en 4 de la mise-en-œuvre de cet algorithme de filtrage, et on propose deux algorithmes pour calculer efficacement l'ordre partiel sur les étiquettes : un algorithme exact et un algorithme approché, dont la complexité en temps est inférieure. On donne ensuite en 5 quelques résultats expérimentaux, montrant que cet algorithme de filtrage est plus efficace que les algorithmes dédiés pour les instances "scale free".

2 Isomorphisme de sous-graphe

2.1 Définitions

Un graphe non orienté $G = (S, A)$ est constitué d'un ensemble S de sommets et d'un ensemble $A \subseteq S \times S$ d'arêtes, où une arête (u, v) est une paire de sommets.

Un *problème d'isomorphisme de sous-graphe partiel* entre un graphe motif $G_m = (S_m, A_m)$ et un graphe cible $G_c = (S_c, A_c)$ consiste à décider s'il existe une injection $f : S_m \to S_c$ telle que

$$\forall (u, v) \in S_m \times S_m, (u, v) \in A_m \Rightarrow (f(u), f(v)) \in A_c$$

Notons que l'on impose de retrouver toutes les arêtes du graphe motif dans le graphe cible, mais qu'il est possible que des arêtes du graphe cible ne se retrouvent pas dans le graphe motif.

Dans un *problème d'isomorphisme de sous-graphe induit*, on impose une équivalence stricte entre les arêtes des sommets appariés, i.e., $\forall (u, v) \in S_m \times S_m, (u, v) \in A_m \Leftrightarrow (f(u), f(v)) \in A_c$.

Dans la suite de cet article, nous considérons le problème de l'isomorphisme de sous-graphe partiel, que nous appellerons "problème d'isomorphisme de sous-graphe" afin d'alléger la lecture. La fonction f sera appelée *fonction d'isomorphisme de sous-graphe*.

Dans la suite, on supposera que l'instance du problème d'isomorphisme de sous-graphe à résoudre est définie par le graphe motif $G_m = (S_m, A_m)$ et le graphe cible $G_c = (S_c, A_c)$. On définit également $S = S_m \cup S_c$, $A = A_m \cup A_c$, $s_m = \#S_m$, $s_c = \#S_c$, d_m et d_c le degré maximal des graphes G_m et G_c, et $d = max(d_m, d_c)$. L'ensemble des nombres naturels est dénoté par $\mathbb{N}$.

2.2 Modélisation en PPC

Un problème d'isomorphisme de sous-graphe peut être facilement formulé en termes de contraintes [15, 16, 11]. Une variable x_u est associée à chaque sommet u du graphe motif, son domaine étant l'ensemble des sommets du graphe cible. Une première contrainte globale de différence (Alldiff) [14] contraint la fonction d'isomorphisme à être injective. L'appariement des arêtes est assuré par un ensemble de contraintes binaires, appelées $c2$ dans [11] : $\forall (u, v) \in S_m \times S_m$,

$$c2(x_u, x_v) \equiv ((u, v) \in A_m \Rightarrow (x_u, x_v) \in A_c)$$

Afin de réduire la complexité en temps pour établir la consistance d'arc sur cet ensemble de contraintes binaires, [11] a introduit une contrainte globale. Cette contrainte est algorithmiquement globale, dans le sens où elle établit la même consistance que l'ensemble original de contraintes binaires, mais plus efficacement. [11] a également proposé d'ajouter des contraintes redondantes, appelées $c3$, pour contraindre le nombre de sommets disponibles dans l'union des domaines des voisins de x_u dans le graphe cible à être supérieur ou égal au nombre de voisins de u dans le graphe motif.

3 Cadre théorique

On introduit ici un nouvel algorithme de filtrage pour l'isomorphisme de sous-graphe ; on montrera dans la section suivante comment ce filtrage peut être mis en œuvre en pratique.

3.1 Etiquetage consistant pour l'isomorphisme de sous-graphe

Définition 1. Un *étiquetage* l est défini par un triplet $(\mathbb{L}, \preceq, \alpha)$ tel que

- $\mathbb{L}$ est un ensemble d'étiquettes pouvant être associées aux sommets ;

– $\preceq \subseteq \mathbb{L} \times \mathbb{L}$ est un ordre partiel sur $\mathbb{L}$;
– $\alpha : S \to \mathbb{L}$ est une application affectant une étiquette $\alpha(v)$ à chaque sommet v.

Un étiquetage induit une relation de compatibilité entre les sommets du graphe motif et ceux du graphe cible.

Définition 2. L'ensemble des *couples compatibles de sommets* induit par un étiquetage $l = (\mathbb{L}, \preceq, \alpha)$ est défini par $CC_l = \{(u, v) \subseteq S_m \times S_c \mid \alpha(u) \preceq \alpha(v)\}$

Cette relation de compatibilité peut être utilisée pour filtrer les domaines de la façon suivante :

FILTRE(l)
  Pour chaque $u \in S_m$ faire :
    $\mathrm{D}(x_u) \leftarrow \mathrm{D}(x_u) \cap \{v \in S_c \mid (u, v) \in CC_l\}$

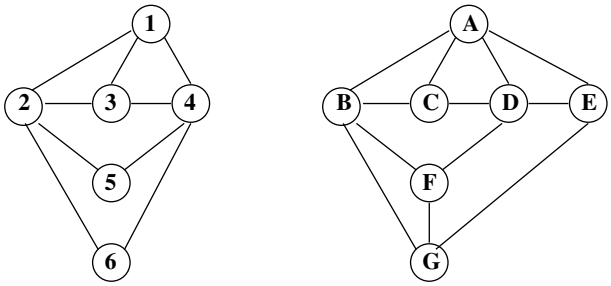
En supposant que la comparaison de deux étiquettes se fasse en temps constant, la complexité de FILTRE est en $\mathcal{O}(s_m.s_c)$.

Notre but est de trouver un étiquetage qui filtre le plus fortement possible les domaines des variables, sans pour autant supprimer de solutions, i.e., si un sommet v du graphe motif peut être apparié à un sommet u du graphe cible, alors l'étiquette de v doit être compatible avec celle de u. Cette propriété est appelée consistance d'isomorphisme de sous-graphe.

Définition 3. Un étiquetage l est *consistant pour l'isomorphisme de sous-graphe* (IS-consistant) ssi pour toute fonction d'isomorphisme de sous-graphe f, on a : $\forall v \in S_m, (v, f(v)) \in CC_l$.

Pour les problèmes d'isomorphisme et d'automorphisme de graphes, comme par exemple dans Nauty [12], un tel étiquetage est appelé un "invariant". Dans ce cas, l'ordre partiel est remplacé par une relation d'équivalence : deux sommets sont compatibles s'ils ont la même étiquette.

De nombreuses propriétés des graphes, qui sont "invariantes" pour l'isomorphisme de sous-graphe, peuvent être utilisées pour définir des étiquetages IS-consistants. Par exemple, les trois étiquetages suivants sont IS-consistants :
– $l_{deg} = (\mathbb{N}, \leq, deg)$ où deg est la fonction qui retourne le degré des sommets ;
– $l_{distance} = (\mathbb{N}, \leq, distance_k)$ où $distance_k$ est la fonction qui retourne le nombre de sommets qui sont atteignables par un chemin de longueur inférieure à k ;
– $l_{clique} = (\mathbb{N}, \leq, clique_k)$ où $clique_k$ est la fonction qui retourne le nombre de cliques de taille k contenant le sommet.



Graphe motif G_m **Graphe cible G_c**

FIG. 1 – Instance du problème d'isomorphisme de sous-graphe.

Exemple. Considérons par exemple l'instance de problème décrite dans la figure 1. Notons que cette instance n'a pas de solution car G_m ne peut être apparié à un sous-graphe de G_c. La figure 2 décrit l'étiquetage $l_{deg} = (\mathbb{N}, \leq, deg)$ pour ces graphes.

3.2 Renforcement d'étiquetages

Nous proposons de commencer à partir d'un étiquetage IS-consistant basique, comme par exemple l'étiquetage l_{deg} défini précédemment, et de le renforcer itérativement. La "force" d'un étiquetage est définie par rapport à l'ensemble des couples compatibles induits.

Définition 4. Etant donnés deux étiquetages l et l' :
- *l' est strictement plus fort que l* ssi $CC_{l'} \subset CC_l$,
- *l' est équivalent à l* ssi $CC_{l'} = CC_l$.

Un étiquetage plus fort permet un meilleur filtrage puisqu'il contient moins de couples compatibles.

Pour renforcer un étiquetage, l'idée est d'étendre l'étiquette de chaque sommet en ajoutant des informations sur les étiquettes de ses voisins. Comme plusieurs voisins peuvent avoir une même étiquette, cette information est un multi-ensemble. Nous utiliserons les notations suivantes pour ces multi-ensembles.

Définition 5. Etant donné un ensemble de référence A, un *multi-ensemble* est une fonction $m : A \to \mathbb{N}$, telle que $m(a)$ est la multiplicité (i.e., le nombre d'occurrences) de l'élément a dans m. Le multi-ensemble m peut également être représenté par le "bag" $\{a_0, \ldots, a_0, a_1, \ldots\}$ où chaque élément est répété autant de fois que d'occurrences.

Par exemple, le multi-ensemble m contenant 3 occurrences de a, 2 occurrences de b et 1 occurrence de c est défini par $m(a)$=3, $m(b)$=2, $m(c)$=1 et, $\forall x \notin \{a, b, c\}$, $m(x)$=0 ; ce multi-ensemble peut également être représenté par $\{a, a, a, b, b, c\}$.

L'étiquetage $l_{deg} = (\mathbb{N}, \leq, deg)$ affecte les étiquettes suivantes aux sommets des graphes de la figure 1 :

$$\begin{aligned} deg(A) = deg(B) = deg(D) = deg(2) = deg(4) &= 4 \\ deg(C) = deg(E) = deg(F) = deg(G) = deg(1) = deg(3) &= 3 \\ deg(5) = deg(6) &= 2 \end{aligned}$$

Par conséquent, l'ensemble des couples compatibles induit par cet étiquetage est

$$\begin{aligned} CC_{l_{deg}} &= \{(u,v) \mid u \in \{2,4\}, v \in \{A,B,D\}\} \\ &\cup \{(u,v) \mid u \in \{1,3,5,6\}, v \in \{A,B,C,D,E,F,G\}\} \end{aligned}$$

Cet ensemble de couples compatibles permet à FILTRE(l_{deg}) de supprimer les valeurs C, E, F et G des domaines des variables associées aux sommets 2 et 4.

FIG. 2 – Etiquetage $l_{deg} = (\mathbb{N}, \leq, deg)$ pour les sommets des graphes de la figure 1

Etant donné un ordre partiel défini sur un ensemble A, on étend cet ordre partiel aux multi-ensembles définis sur A comme suit.

Définition 6. Etant donnés deux multi-ensembles m et m' définis sur un ensemble A, et un ordre partiel $\preceq \subseteq A \times A$, on définit $m \preceq m'$ ssi il existe une injection $t : m \to m'$ telle que $\forall a_i \in m, a_i \preceq t(a_i)$.

Autrement dit, $m \preceq m'$ ssi pour chaque occurrence d'élément de m il existe une occurrence d'élément différente de m' qui est plus grande ou égale. Par exemple, si on considère l'ordre classique sur $\mathbb{N}$, on a $\{3,3,4\} \preceq \{2,3,5,5\}$, tandis que $\{3,3,4\}$ n'est pas comparable avec $\{2,5,6\}$. Notons que la comparaison de deux multi-ensembles n'est pas triviale dans le cas général, notamment si la relation d'ordre sur l'ensemble de départ A n'est pas totale. Ce point est abordé dans la section suivante.

Nous pouvons maintenant définir la procédure de renforcement d'étiquetage.

Définition 7. Etant donné un étiquetage $l = (\mathbb{L}, \preceq, \alpha)$, l'*étiquetage renforcé de* l est l'étiquetage $l' = (\mathbb{L}', \preceq', \alpha')$ tel que :

- chaque étiquette de $\mathbb{L}'$ est composée d'une étiquette de $\mathbb{L}$ suivie d'un multi-ensemble d'étiquettes de $\mathbb{L}$, i.e., $\mathbb{L}' = \mathbb{L} \cdot (\mathbb{L} \to \mathbb{N})$;
- la fonction d'étiquetage α' étend chaque étiquette $\alpha(v)$ par le multi-ensemble des étiquettes des voisins de v, i.e., $\alpha'(v) = \alpha(v) \cdot m$ où $\forall l_i \in \mathbb{L}$, $m(l_i) = \#\{u \mid (u,v) \in A \wedge \alpha(u) = l_i\}$;
- l'ordre partiel sur les étiquettes étendues de $\mathbb{L}'$ est défini par $l_1 \cdot m_1 \preceq' l_2 \cdot m_2$ ssi $l_1 \preceq l_2$ et $m_1 \preceq m_2$.

Le théorème suivant montre que l'étiquetage renforcé d'un étiquetage IS-consistant est également IS-consistant et est plus fort que (ou équivalent à) l'étiquetage initial.

Théorème 1. Soient deux étiquetages $l = (\mathbb{L}, \preceq, \alpha)$ et $l' = (\mathbb{L}', \preceq', \alpha')$ tels que l soit IS-consistant et l' soit l'étiquetage renforcé de l.
(i) l' est aussi IS-consistant
(ii) l' est plus fort que (ou équivalent à) l.

Preuve. (i) : Soient une fonction d'isomorphisme de sous-graphe f et un sommet $v \in S_m$. Il s'agit de montrer que $\alpha'(v) \preceq' \alpha'(f(v))$, autrement dit $\alpha(v) \preceq \alpha(f(v))$ et $m \preceq m'$, où m (resp. m') est le multi-ensemble contenant les étiquettes des voisins de v dans G_m (resp. des voisins de $f(v)$ dans G_c). On a $\alpha(v) \preceq \alpha(f(v))$ car l est IS-consistant. On a $m \preceq m'$ car m' contient, pour chaque voisin u de v, l'étiquette $\alpha(f(u))$ du sommet apparié à u par la fonction d'isomorphisme de sous-graphe f ; comme l est IS-consistant, $\alpha(u) \preceq \alpha(f(u))$ et donc $m \preceq m'$.
(ii) : C'est une conséquence directe de la définition 7 qui étend l'ordre partiel aux étiquettes étendues : $\alpha(u) \preceq \alpha(v)$ est une condition nécessaire pour avoir $\alpha'(u) \preceq \alpha'(v)$. □

Exemple. Considérons de nouveau l'instance de la figure 1, et l'étiquetage l_{deg} défini en 3.1. L'étiquetage renforcé de l_{deg} est décrit dans la figure 3.

3.3 Renforcement itératif d'étiquetages

L'idée est de renforcer itérativement les étiquettes, en partant d'un étiquetage IS-consistant l donné.

Définition 8. Soit un étiquetage IS-consistant initial $l = (\mathbb{L}, \preceq, \alpha)$. On définit la suite d'étiquetages IS-consistants $l^k = (\mathbb{L}^k, \preceq^k, \alpha^k)$:

- $l^0 = l$
- l^{k+1} = étiquetage renforcé de l^k $(k \geq 0)$

L'étiquetage renforcé de l_{deg} est l'étiquetage $l' = (\mathbb{L}', \preceq', \alpha')$ décrit ci-dessous. Notons que l'on ne mentionne que les relations de compatibilité $l_i \preceq' l_j$ pour lesquelles l_i est une étiquette du graphe motif et l_j une étiquette du graphe cible, les autres relations n'étant d'aucune utilité pour le filtrage.

$$\begin{array}{rll}
\alpha'(A) & = 4 \cdot \{3,3,4,4\} & \\
\alpha'(B) = \alpha'(D) & = 4 \cdot \{3,3,3,4\} & \\
\alpha'(2) = \alpha'(4) & = 4 \cdot \{2,2,3,3\} & \preceq' 4 \cdot \{3,3,4,4\} \text{ et } 4 \cdot \{3,3,3,4\} \\
\alpha'(C) & = 3 \cdot \{4,4,4\} & \\
\alpha'(E) = \alpha'(F) = \alpha'(1) = \alpha'(3) & = 3 \cdot \{3,4,4\} & \preceq' 4 \cdot \{3,3,4,4\}, 3 \cdot \{4,4,4\} \text{ et } 3 \cdot \{3,4,4\} \\
\alpha'(G) & = 3 \cdot \{3,3,4\} & \\
\alpha'(5) = \alpha'(6) & = 2 \cdot \{4,4\} & \preceq' 4 \cdot \{3,3,4,4\}, 3 \cdot \{4,4,4\} \text{ et } 3 \cdot \{3,4,4\}
\end{array}$$

Ainsi, l'ensemble des couples compatibles induits par cet étiquetage est

$$CC_{l'} = \{(u,v) \mid u \in \{2,4\}, v \in \{A,B,D\}\} \cup \{(u,v) \mid u \in \{1,3,5,6\}, v \in \{A,C,E,F\}\}$$

Cet ensemble de couples compatibles permet à FILTRE(l') de supprimer les valeurs B, D et G des domaines des variables associées aux sommets 1, 3, 5 et 6.

FIG. 3 – Renforcement de l'étiquetage l_{deg} pour l'instance de la figure 1

Un filtrage théorique peut être imaginé à partir de cette suite. Partant d'un étiquetage IS-consistant initial $l = l^0$, on calcule itérativement l^{k+1} à partir de l^k, et on applique la procédure FILTRE(l^{k+1}) jusqu'à ce qu'un domaine devienne vide (indiquant ainsi que l'instance n'a pas de solution), ou bien jusqu'à ce qu'une condition d'arrêt soit vérifiée.

Une condition d'arrêt peut être l'atteinte d'un point fixe, i.e., une itération à partir de laquelle les ré-étiquetages suivants ne renforceront plus l'étiquetage. On montre dans [22] que ce point fixe est atteint lorsque à la fois l'ensemble des couples compatibles induits et le nombre d'étiquettes différentes sont inchangés par le ré-étiquetage. Ce point fixe est atteint en au plus $O(s_m.s_c)$ itérations.

Exemple. Considérons de nouveau l'instance de la figure 1, et supposons que la suite d'étiquetages IS-consistants est commencée à partir de $l^0 = l_{deg}$ comme défini en 3.1. La figure 4 montre que le renforcement itéré de cet étiquetage initial permet de détecter l'inconsistance de cette instance en 2 itérations.

Notons cependant que pour d'autres instances, la procédure de filtrage peut ne réduire aucun domaine. C'est le cas en particulier si l'étiquetage initial est l_{deg}, et si le degré maximal du graphe motif est inférieur ou égal au degré minimal du graphe cible, ou plus généralement, si l'étiquetage initial est tel que toute étiquette associée à un sommet du graphe motif est inférieure ou égale à toute étiquette associée à un sommet du graphe cible. Dans ce cas, chaque sommet du graphe motif est compatible avec tous les sommets du graphe cible de sorte qu'aucun domaine n'est réduit.

3.4 Filtrage dans le contexte d'une recherche par séparation et propagation

Nous introduisons ici différentes optimisations qui peuvent être apportées dans le contexte d'une recherche par séparation et propagation, quand une valeur est affectée à une variable à chaque étape de la recherche.

Au départ, le domaine d'une variable x_u (associée à un sommet u du graphe motif) contient l'ensemble des sommets du graphe cible, i.e., $D(x_u) = S_c$. Si au cours de la recherche un sommet du graphe cible n'appartient à aucun domaine $D(x_u)$, alors ce sommet peut être ignoré. Plus précisément, la cible à considérer est le sous-graphe de G_c induit par les sommets apparaissant dans les domaines des variables x_u.

Définition 9. A chaque étape de la recherche, le graphe cible courant est le graphe $G'_c = (S'_c, A'_c)$ tel que $S'_c = \cup_{i \in S_m} D(x_i)$ et $A'_c = \{(u,v) \in A_c \mid u,v \in S'_c\}$

Une première optimisation est faite lorsque les graphes cibles courants issus de deux étapes successives de la recherche sont isomorphes : étant donné que le graphe motif ne change pas, il n'est pas nécessaire d'exécuter de nouveau l'algorithme de filtrage. Notons que pour tester si le graphe cible courant n'a pas changé, il suffit de tester si le nombre de sommets a changé.

Une seconde optimisation est faite lorsqu'à une étape de la recherche la plus grande étiquette du graphe motif est inférieure ou égale à la plus petite étiquette du graphe cible : dans ce cas, chaque sommet du graphe motif a une étiquette compatible avec les étiquettes de tous les sommets du graphe cible de

Après la première itération, l'étiquetage renforcé de l'étiquetage l^0 décrit dans la figure 3 est l'étiquetage $l^1 = (\mathbb{L}^1, \preceq^1, \alpha^1)$ décrit ci-dessous (afin de faciliter la lecture, nous avons renommé les étiquettes).

$$\begin{array}{rll}
\alpha^1(A) & = 4 \cdot \{3,3,4,4\} & \text{renommé en } m_1 \\
\alpha^1(B) = \alpha^1(D) & = 4 \cdot \{3,3,3,4\} & \text{renommé en } m_2 \\
\alpha^1(2) = \alpha^1(4) & = 4 \cdot \{2,2,3,3\} & \text{renommé en } m_3 \quad \preceq^1 \{m_1, m_2\} \\
\alpha^1(C) & = 3 \cdot \{4,4,4\} & \text{renommé en } m_4 \\
\alpha^1(E) = \alpha^1(F) = \alpha^1(1) = \alpha^1(3) & = 3 \cdot \{3,4,4\} & \text{renommé en } m_5 \quad \preceq^1 \{m_1, m_4, m_5\} \\
\alpha^1(G) & = 3 \cdot \{3,3,4\} & \text{renommé en } m_6 \\
\alpha^1(5) = \alpha^1(6) & = 2 \cdot \{4,4\} & \text{renommé en } m_7 \quad \preceq^1 \{m_1, m_4, m_5\}
\end{array}$$

L'étiquetage renforcé de l^1 est l'étiquetage $l^2 = (\mathbb{L}^2, \preceq^2, \alpha^2)$ tel que

$$\begin{array}{rll}
\alpha^2(A) & = m_1 \cdot \{m_2, m_2, m_4, m_5\} & \text{renommé en } n_1 \\
\alpha^2(B) & = m_2 \cdot \{m_1, m_4, m_5, m_6\} & \text{renommé en } n_2 \\
\alpha^2(C) & = m_4 \cdot \{m_1, m_2, m_2\} & \text{renommé en } n_3 \\
\alpha^2(D) & = m_2 \cdot \{m_1, m_4, m_5, m_5\} & \text{renommé en } n_4 \\
\alpha^2(E) & = m_5 \cdot \{m_1, m_2, m_6\} & \text{renommé en } n_5 \\
\alpha^2(F) & = m_5 \cdot \{m_2, m_2, m_6\} & \text{renommé en } n_6 \\
\alpha^2(G) & = m_6 \cdot \{m_2, m_5, m_5\} & \text{renommé en } n_7 \\
\alpha^2(1) = \alpha^2(3) & = m_5 \cdot \{m_3, m_3, m_5\} & \text{renommé en } n_8 \quad \preceq^2 \{n_1, n_3\} \\
\alpha^2(2) = \alpha^2(4) & = m_3 \cdot \{m_5, m_5, m_7, m_7\} & \text{renommé en } n_9 \quad \preceq^2 \{n_4\} \\
\alpha^2(5) = \alpha^2(6) & = m_7 \cdot \{m_3, m_3\} & \text{renommé en } n_{10} \quad \preceq^2 \{n_1, n_3, n_5, n_6\}
\end{array}$$

L'ensemble des couples compatibles induits par cet étiquetage est

$$CC_{l^2} = \{(2,D),(4,D),(1,A),(1,C),(3,A),(3,C)\} \cup \{(u,v) \mid u \in \{5,6\}, v \in \{A,C,E,F\}\}$$

Cet ensemble de couples compatibles permet à FILTRE(l^2) de supprimer les valeurs A et B des domaines des variables associées aux sommets 2 et 4. Par conséquent, les domaines de ces deux variables ne contiennent plus qu'une seule valeur (D), et grâce à la contrainte globale *alldiff*, une inconsistance est détectée.

FIG. 4 – Suite d'étiquetages IS-consistants partant de $l^0 = l_{deg} = (\mathbb{N}, \leq, deg)$ pour l'instance de la figure 1.

sorte qu'aucun domaine ne pourra être réduit.

Une troisième optimisation est faite lorsqu'au cours de la recherche une variable associée à un sommet du graphe motif est affectée à un sommet du graphe cible. Dans ce cas, la procédure de renforcement d'étiquetage est modifiée afin d'obliger les deux sommets à avoir la même nouvelle étiquette (incompatible avec les autres étiquettes) de la façon suivante :

Définition 10. Soit $l = (\mathbb{L}, \preceq, \alpha)$, un étiquetage IS-consistant, et soit (u, v), un couple de sommets de $S_m \times S_c$ tel que $v \in x_u$. La propagation de $x_u = v$ sur l est le nouvel étiquetage $l' = (\mathbb{L}', \preceq', \alpha')$ tel que

- $\mathbb{L}' = \mathbb{L} \cup \{l_{uv}\}$ où l_{uv} est une nouvelle étiquette telle que $l_{uv} \notin \mathbb{L}$;
- $\preceq' = \preceq \cup \{(l_{uv}, l_{uv})\}$ de sorte que la nouvelle étiquette l_{uv} n'est comparable avec aucune autre étiquette à part elle-même ;
- $\alpha'(u) = \alpha'(v) = l_{uv}$ et $\forall w \in S \setminus \{u, v\}, \alpha'(w) = \alpha(w)$.

Cet étiquetage l' est utilisé comme point de départ d'une nouvelle suite de renforcements d'étiquetages. Notons que cette propagation est faite à chaque fois qu'un domaine est réduit à un singleton.

4 Mise-en-œuvre pratique

L'objectif de la procédure de filtrage est de calculer, à chaque itération k, l'ensemble CC_{l^k} des couples compatibles induits par l'étiquetage l^k. Cet ensemble ne peut que diminuer d'une itération à l'autre : si un couple n'est pas compatible à l'itération k, alors il ne le sera plus aux itérations suivantes. Ainsi, il s'agit de déterminer à chaque itération k, pour chaque couple $(u, v) \in CC_{l^{k-1}}$, si l'étiquette de u est compatible avec l'étiquette de v, i.e., si $\alpha^k(u) \preceq^k \alpha^k(v)$.

Le calcul de la fonction d'étiquetage α^k à partir de la fonction d'étiquetage α^{k-1} peut être fait en temps linéaire par rapport au nombre d'arêtes des deux graphes, sous réserve que les étiquettes soient renommées à chaque itération (afin de permettre leur comparaison en temps constant).

La calcul de l'ordre partiel $\preceq^k$ est plus délicat. Nous montrons en 4.1 comment le calculer de façon exacte en $\mathcal{O}(s_m \cdot s_c \cdot d^{5/2})$. On montre ensuite en 4.2 comment calculer un ordre induisant un filtrage plus faible, en $\mathcal{O}(s_c \cdot d_m \cdot (s_m + d_c \cdot \log s_c))$. Ces deux algorithmes sont comparés expérimentalement en 5.

4.1 Calcul exact de l'ordre partiel

Etant donnés l'ordre partiel $\preceq^{k-1}$ et l'ensemble de couples compatibles $CC_{l^{k-1}}$ calculés à l'itération $k-1$, il s'agit de déterminer, pour chaque couple de sommets $(u, v) \in CC_{l^{k-1}}$, si $\alpha^k(u) \preceq^k \alpha^k(v)$. Notons m_u^k et m_v^k les multi-ensembles contenant les étiquettes des voisins des sommets u et v à l'itération k, de sorte que $\alpha^k(u) = \alpha^{k-1}(u).m_u^k$ et $\alpha^k(v) = \alpha^{k-1}(v).m_v^k$. Pour déterminer si $\alpha^k(u) \preceq^k \alpha^k(v)$, il suffit donc de déterminer si $m_u^k \preceq^k m_v^k$, i.e., s'il existe pour chaque occurrence d'élément de m_u^k une occurrence d'élément (distincte) dans m_v^k qui soit supérieure ou égale selon l'ordre partiel $\preceq^{k-1}$.

Propriété 1. Soit $G = (S = (S_u, S_v), A)$, le graphe non orienté bipartie tel que S_u (resp. S_v) associe un sommet différent à chaque occurrence d'élément du multi-ensemble m_u^k (resp. m_v^k), et A contient l'ensemble des arêtes (i, j) telles que $i \preceq^{k-1} j$. On a : $m_u^k \preceq_k m_v^k$ ssi il existe un appariement couvrant l'ensemble de sommets S_u dans le graphe bipartie G.

Hopcroft [10] propose un algorithme pour ce problème en $O(d_m.d_c\sqrt{d_m + d_c})$, c'est-à-dire $O(d^{5/2})$.

Ce calcul devant être effectué pour chaque couple d'étiquettes $(\alpha^k(u), \alpha^k(v))$ tel que $(u, v) \in CC_{l^{k-1}}$, et le nombre d'étiquettes différentes pour chaque graphe étant borné par son nombre de sommets, le calcul de $\preceq^k$ à partir de $\preceq^{k-1}$ est en $O(s_m.s_c.d^{5/2})$.

4.2 Transformation de l'ordre partiel en un ordre total

Le calcul de l'ordre partiel $\preceq^k$ est simplifié lorsque la relation d'ordre $\preceq^{k-1}$ de l'itération précédente est un ordre total : dans ce cas, il suffit de parcourir le multi-ensemble m_u^k, de son plus petit élément à son plus grand élément, en cherchant à chaque fois le plus petit élément du multi-ensemble m_v^k qui soit compatible et qui n'ait pas encore été apparié à un élément de m_u^k. Chaque multi-ensemble comportant $\mathcal{O}(d)$ occurrences d'éléments, et le nombre d'étiquettes étant $s_m + s_c$, le tri de tous les multi-ensembles est en $\mathcal{O}((s_m + s_c) \cdot d \cdot \log d)$. Il faut ensuite comparer $\mathcal{O}(s_m \cdot s_c)$ paires d'étiquettes, et chaque comparaison est en $\mathcal{O}(d)$ (les multi-ensembles étant triés). Par conséquent, lorsque $\preceq^{k-1}$ est une relation d'ordre totale, le calcul de $\preceq^k$ est en $\mathcal{O}(s_m.s_c.d)$.

On montre maintenant que lorsque $\preceq^{k-1}$ n'est pas un ordre total, on peut l'étendre en un ordre total[1], et utiliser cet ordre total pour calculer un sur-ensemble de l'ensemble CC_{l^k} des couples compatibles induits par l'étiquetage l^k. Ce sur-ensemble, calculé plus efficacement que l'ensemble exact, induira cependant un filtrage moins fort.

La définition suivante donne une condition simple sur l'ordre total nous assurant sa validité par rapport à l'ordre partiel.

[1] Les nouvelles étiquettes introduites pour propager une affectation (cf définition 10) sont exclues du calcul de l'ordre total car elles ne sont compatibles avec aucune autre étiquette.

Définition 11. Soit un étiquetage $l = (\mathbb{L}, \preceq, \alpha)$. Un *ordre total valide* pour l est un ordre total $\leq$ sur $\mathbb{L}$ tel que $\forall u \in S_m, v \in S_c : \alpha(u) \preceq \alpha(v) \Rightarrow \alpha(u) \leq \alpha(v)$

On étend l'ordre $\leq$ aux multi-ensembles, de façon similaire à ce qui est fait pour les ordres partiels dans la définition 6, i.e., $m \leq m'$ ssi il existe une injection $t : m \to m'$ telle que $\forall a_i \in m, a_i \leq t(a_i)$. Ainsi, nous avons $m \preceq m' \Rightarrow m \leq m'$. Notons cependant que cette extension de $\leq$ aux multi-ensembles n'induit qu'un ordre partiel sur les multi-ensembles, certains multi-ensembles pouvant être non comparables.

On peut alors définir une nouvelle procédure de renforcement d'étiquetage basée sur un ordre total valide.

Définition 12. Soient $l = (\mathbb{L}, \preceq, \alpha)$, un étiquetage, et $\leq$, un ordre total valide pour l. Le *renforcement de l basé sur* $\leq$ est l'étiquetage $l'_{\leq} = (\mathbb{L}', \preceq'_{\leq}, \alpha')$ où

- $\mathbb{L}'$ et α' sont définis comme pour l'étiquetage renforcé de l (cf définition 7) ;
- la relation d'ordre $\preceq'_{\leq} \subseteq \mathbb{L}' \times \mathbb{L}'$ est définie par :

$$l_1 \cdot m_1 \preceq'_{\leq} l_2 \cdot m_2 \text{ ssi } l_1 \preceq l_2 \wedge m_1 \leq m_2$$

Le théorème suivant montre que l'étiquetage renforcé $l'_{\leq}$ basé sur l'ordre total $\leq$ peut être utilisé dans notre procédure d'étiquetage itératif, et permet plus (ou autant) de filtrage que l. Cependant, ce filtrage peut être moins bon que celui obtenu avec l'étiquetage renforcé basé sur l'ordre partiel $\preceq$: de fait, l'ordre total induit plus de couples compatibles que l'ordre partiel.

Théorème 3. Soient $l = (\mathbb{L}, \preceq, \alpha)$, $l' = (\mathbb{L}', \preceq', \alpha')$ et $l'_{\leq} = (\mathbb{L}', \preceq'_{\leq}, \alpha')$, trois étiquetages tels que l' soit l'étiquetage renforcé de l et $l'_{\leq}$ soit l'étiquetage renforcé de l basé sur un ordre total valide $\leq$.

Si l est IS-consistant, alors (i) $l'_{\leq}$ est IS-consistant, (ii) $l'_{\leq}$ est plus fort que (ou égal à) l, et (iii) l' est plus fort que (ou égal à) $l'_{\leq}$.

Preuve. (ii) et (iii) : Pour l'étiquetage l', on a $l_1 \cdot m_1 \preceq' l_2 \cdot m_2$ ssi $l_1 \preceq l_2 \wedge m_1 \preceq m_2$. Etant donné que $\leq$ est un ordre total valide pour l, on a $m \preceq m' \Rightarrow m \leq m'$. Par conséquent $CC_{l'} \subseteq CC_{l'_{\leq}} \subseteq CC_l$.
(i) est une conséquence directe de (ii), étant donné que l' est IS-consistant (propriété (i) du Théorème 1). □

Différents ordres totaux valides peuvent être dérivés à partir d'un ordre partiel, permettant des filtrages plus ou moins forts : le filtrage est d'autant plus fort que l'ordre total introduit le moins de nouveaux couples de sommets compatibles. Dans le cas particulier où tous les sommets ont des étiquettes différentes, cela revient à chercher l'ordre total introduisant le moins de nouveaux couples d'étiquettes compatibles. Malheureusement, le problème consistant à chercher l'ordre total valide minimisant le nombre de nouveaux couples d'étiquettes compatibles est NP-difficile.

Théorème 4. Le problème de la recherche d'un ordre total valide introduisant au plus k nouveaux couples d'étiquettes compatibles est NP-complet.

Preuve. On montre que le problème de la recherche d'un "Feedback Arc Set" dans une orientation d'un graphe bipartie complet (FAS), qui est NP-complet [8, 9], se ramène à notre problème. Etant donnée une orientation d'un graphe complet bipartie $G = (S = (S1, S2), A)$ tel que pour tout couple de sommets $(u_1, u_2) \in S1 \times S2$, soit $(u_1, u_2) \in A$, soit $(u_2, u_1) \in A$, le problème FAS consiste à décider s'il est possible de rendre G acyclique en supprimant au plus k arcs. Pour résoudre une instance de ce problème, on peut définir l'ordre partiel $\preceq \subseteq S1 \times S2$ tel que $u_1 \preceq u_2$ ssi $(u_1, u_2) \in A$. Il s'agit alors de décider s'il existe un ordre total $\leq$ valide pour $\preceq$ qui introduit au plus k nouveaux couples compatibles $(u_1, u_2) \in S1 \times S2$ pour lesquels $u_1 \leq u_2$ mais $u_1 \not\preceq u_2$. □

Algorithme heuristique pour calculer un ordre total valide. Nous proposons maintenant un algorithme heuristique visant à construire un ordre total introduisant peu de nouveaux couples de sommets compatibles (sans garantie d'optimalité). Notons L_m (resp. L_c) l'ensemble des étiquettes associées à des sommets du graphe motif G_m (resp. cible G_c). Nous supposerons que $L_m \cap L_c = \emptyset$ sans perte de généralité[2]. L'idée est alors de séquencer l'ensemble des étiquettes de $L_m \cup L_c$, définissant de la sorte un ordre total entre ces étiquettes, selon le principe glouton suivant : partant d'une séquence vide, on ajoute itérativement une ou plusieurs nouvelles étiquettes en fin de séquence, et on les supprime des ensembles L_m et L_c, jusqu'à ce que $L_m \cup L_c = \emptyset$.

Pour choisir les étiquettes ajoutées dans la séquence à chaque itération, notre heuristique se base sur le fait que les nouveaux couples de sommets compatibles sont introduits par des nouveaux couples d'étiquettes compatibles (e_m, e_c) tels que $e_m \in L_m$ et $e_c \in L_c$. Par conséquent, l'objectif est de séquencer le plus tard possible les étiquettes de L_m. Pour cela, on commence par chercher l'ensemble des étiquettes $e_c \in L_c$ qui minimisent le nombre d'étiquettes $e_m \in L_m$ pour lesquelles $e_m \preceq e_c$. Dans le cas où cet ensemble comporte plusieurs étiquettes, on choisit alors une étiquette e_c qui minimise le nombre moyen d'étiquettes $e'_c \in L_c$

[2]Si une étiquette e appartient à la fois à L_m et à L_c, alors il est toujours possible de renommer e en e' dans L_c (où e' est une nouvelle étiquette), et d'ajouter l'ensemble des relations $e'' \preceq e'$ pour chaque étiquette $e'' \in L_m$ telle que $e'' \preceq e$.

telles que $e_m \preceq e'_c$, pour chaque étiquette $e_m \in L_m$ telle que $e_m \preceq e_c$. On introduit alors dans la séquence l'étiquette e_c précédée de l'ensemble des étiquettes $e_m \in L_m$ pour lesquelles $e_m \preceq e_c$.

La complexité en temps de cet algorithme heuristique est en $\mathcal{O}(s_c \cdot \log s_c \cdot d_m \cdot d_c)$; il doit être exécuté à chaque itération. Une fois l'ordre total $\leq$ déterminé, il faut encore calculer $\preceq^k_{\leq}$, ce qui peut être fait en $\mathcal{O}(s_m.s_c.d)$. Ainsi, chaque renforcement d'étiquetage basé sur un ordre total a une complexité en temps en $\mathcal{O}(s_c.d.(s_m + d_c \cdot \log s_c))$.

5 Résultats expérimentaux

L'objectif des premières expérimentations rapportées ici est d'apporter des éléments de réponse aux questions suivantes :

- Pour quelles familles de graphes notre algorithme de filtrage donne-t'il de meilleurs résultats que les approches existantes pour l'isomorphisme de sous-graphe ?
- Quelles sont les performances du renforcement approché basé sur l'ordre total, par rapport au renforcement exact basé sur l'ordre partiel ?
- Quel est le nombre optimal d'itérations pour obtenir un bon équilibre entre la force du filtrage et son temps d'exécution ?

Approches considérées. Pour résoudre le problème d'isomorphisme de sous-graphe à l'aide de la PPC, on peut considérer différentes modélisations et différents niveaux de consistance.

- On note "c2+*AllDiff*" la formulation PPC basée sur un ensemble de contraintes $c2$ telles que définies en 2.2 et une contrainte globale *AllDiff*.
- On note "c2+*AllDiff*+c3" la formulation PPC ajoutant en plus la contrainte redondante $c3$ de [11] (cf 2.2).

Pour ces deux formulations, on peut considérer deux niveaux de propagation :

- on note "FC" l'approche basée sur une simple propagation des contraintes par forward checking ;
- on note "AC" l'approche filtrant les domaines pour établir la consistance d'arc, en considérant pour cela l'algorithme proposé dans [11].

On compare également des filtrages basés sur différents étiquetages. On considère tout d'abord l'étiquetage $l^0 = l_{deg}$ tel que défini en 3.1. On considère ensuite différents étiquetages renforcés l^k, obtenus après k itérations de la procédure de renforcement, et on note l^∞ l'étiquetage obtenu au point fixe. On considère enfin différents étiquetages $l^k_{\leq}$ obtenus après k itérations de la procédure de renforcement basée sur un ordre total, induisant un filtrage moins fort que l^k. Notons cependant que pour $k = 1$, $l^1 = l^1_{\leq}$ puisque la relation d'ordre dans $l^0 = l_{deg}$ est totale. Pour chacun de ces étiquetages différents, on combine le filtrage tel que défini par la procédure FILTRE avec une propagation par forward checking sur le modèle "c2+*AllDiff*".

Tous les modèles PPC comparés ici ont été implémentés en Gecode (http ://www.gecode.org), en utilisant CP(Graph) et CP(Map) [7] [6]. CP(Graph) introduit des variables de type "graphe" tandis que CP(Map) introduit des variables de type "fonction". L'implémentation est en C++.

Nous avons par ailleurs comparé ces différentes approches PPC avec `vflib`, une implémentation en C++ d'un algorithme de l'état de l'art dédié au problème d'isomorphisme de sous-graphe [4].

Instances considérées. Nous avons considéré des instances générées aléatoirement afin de pouvoir observer le passage à l'échelle des algorithmes comparés, ainsi que la sensibilité de leurs performances à différentes caractéristiques. On regroupe sous l'appellation `SIP-n-`d_{min}`-`d_{max}`-`λ`-`p_s`-`p_a un ensemble de 20 instances générées de la façon suivante.

- On génère un graphe cible non orienté connexe comportant n sommets en utilisant l'algorithme de [19]. Les degrés des sommets, bornés entre d_{min} et d_{max}, sont générés selon une loi de distribution exponentielle $P(x = k) = k^{(-\lambda)}$: cette distribution correspond aux réseaux "scale-free", qui modélisent en pratique un grand nombre de réseaux réels, e.g., les réseaux sociaux, les réseaux d'interaction ou les réseaux neuronaux [1].
- Un graphe motif connexe est extrait à partir du graphe cible en sélectionnant aléatoirement un pourcentage p_s de sommets, puis un pourcentage p_a d'arêtes entre les couples de sommets sélectionnés.

Les graphes motifs étant des sous-graphes partiels des graphes cibles, ces instances sont toutes satisfiables.

Nous avons également généré des instances non satisfiables, dénotées par `SIP-n-`d_{min}`-`d_{max}`-`λ`-`p_s`-`$(p_a + p'_a)$, selon le principe précédent, mais en ajoutant dans le graphe motif un pourcentage p'_a de nouvelles arêtes entre les couples de sommets sélectionnés. Notons que si rien ne garantit que des instances générées de la sorte n'admettent pas de solution, toutes les instances de notre jeu d'essai se sont avérées insatisfiables.

Enfin, nous avons aussi considéré une classe d'instances orientées : le graphe cible est orienté en choisissant pour chaque arête une orientation de façon équiprobable ; le graphe motif orienté est extrait comme dans le cas non orienté. La procédure de renforcement d'étiquetage est adaptée au cas des graphes orientés en étendant l'étiquetage par deux multi-ensembles conte-

	vflib	c2+*AllDiff*		c2+*AllDiff*+c3		l^*+FC(c2+*AllDiff*)			$l^*_{\leq}$+FC(c2+*AllDiff*)			
		FC	AC	FC	AC	l^0	l^1	l^2	$l^2_{\leq}$	$l^4_{\leq}$	$l^8_{\leq}$	$l^{\infty}_{\leq}$
SIP-200	35%	100%	100%	100%	100%	100%	**100%**	100%	100%	100%	55%	100%
5-8-2.5	251.43	57.05	38.66	26.89	22.28	2.17	**0.61**	23.38	1.27	1.87	3.19	321.91
90-90	-	165239	19	67	0	440	**14**	0	13	0	0	0
SIP-600	0%	0%	0%	0%	0%	100%	**100%**	100%	100%	100%	100%	0%
5-8-2.5	-	-	-	-	-	61.35	**5.63**	144.22	24.50	28.66	59.64	-
90-90	-	-	-	-	-	1314	**8**	0	3	0	0	-
SIP-1000	0%	0%	0%	0%	0%	45%	**100%**	45%	100%	100%	100%	0%
5-8-2.5	-	-	-	-	-	439.19	**26.26**	495.84	101.84	110,36	227,81	-
90-90	-	-	-	-	-	1750	**13**	0	2	0	0	-
SIPdir-600	100%	100%	100%	5%	0%	**100%**	100%	100%	100%	100%	100%	0%
5-8-2.5	0.84	7.89	81.70	542.72	0.00	**0.71**	2.58	99.64	7.49	24.67	56.32	-
90-90	-	2402	0	0	0	**2**	0	0	0	0	0	-

FIG. 5 – Résultats sur des instances pour lesquelles le nombre n de sommets du graphe cible varie entre 200 et 1000 (avec d_{min}=5, d_{max}=8, λ=2.5 et p_s=p_a=90%). Les trois premières classes d'instances sont non orientées ; la quatrième est orientée. Pour chaque classe d'instances, on donne le pourcentage d'instances résolues en moins de 600s, le temps d'exécution moyen et le nombre de feuilles échec dans l'arbre de recherche pour les instances résolues.

nant respectivement les étiquettes des prédécesseurs et des successeurs.

Résultats détaillés. Chaque exécution consiste à rechercher toutes les solutions d'une instance. Toutes les exécutions ont été faites sur un Intel Xeon 3,06 Ghz avec 2Go de RAM, et ont été limitées à 600 secondes de temps CPU.

Nous avons comparé les résultats obtenus sur des instances générées avec différentes valeurs du paramètre λ (variant entre 1 et 5), afin de varier la distribution des degrés du graphe cible, et nous avons remarqué que les résultats étaient similaires pour toutes les valeurs de λ considérées, indiquant que les approches comparées ne sont pas sensibles à ce paramètre. Dans les figures 5 et 6, nous rapportons les résultats obtenus avec $\lambda = 2.5$.

Nous avons comparé les résultats obtenus avec différentes valeurs pour le degré minimum, d_{min}, et différentes valeurs du paramètre p_s contrôlant le nombre de sommets du graphe motif par rapport au graphe cible. Nous avons constaté que lorsque $d_{min} \leq 4$ ou lorsque $p_s \leq 70\%$ les instances sont "faciles" et sont donc mieux résolues par des filtrages élémentaires, tels qu'un simple forward checking sans la contrainte c3 redondante, ou un étiquetage élémentaire par les degrés des sommets l_{deg}. Pour ces instances, des propagations plus poussées, comme la consistance d'arc ou nos étiquetages renforcés, ne font qu'augmenter les temps de résolution. Dans les figures 5 et 6, nous rapportons les résultats obtenus sur des instances plus difficiles, pour lesquelles $p_s = 90\%$ et d_{min} a été fixé à 5 puis à 20.

La figure 5 compare les résultats obtenus avec des instances de taille croissante, pour lesquelles le nombre n de sommets du graphe cible varie de 200 à 1000. Pour les instances non orientées des trois premières classes, on remarque que les résultats sont relativement similaires pour les trois valeurs de n. L'approche dédiée vflib est clairement moins performante que les approches PPC. Les approches PPC basées sur une propagation des contraintes c2, allDiff et c3 permettent de résoudre les instances de petite taille, quand $n = 200$, mais pas les instances plus grandes (dans un temps limite de 600s). Pour ces instances, une propagation par consistance d'arc diminue très fortement le nombre de feuilles échec par rapport à un simple forward checking, mais comme l'établissement de la consistance d'arc est coûteux en temps, le temps CPU n'est que légèrement réduit.

Si l'on étudie maintenant les résultats obtenus avec des filtrages basés sur les étiquetages, on constate qu'ils réduisent très fortement à la fois le nombre de feuilles échec et le temps d'exécution. Un simple étiquetage par les degrés des sommets améliore déjà considérablement les performances. Le meilleur compromis entre force et temps de filtrage est obtenu ici avec l^1, correspondant à une seule extension par renforcement de l'étiquetage initial ; les étiquetages obtenus après un plus grand nombre d'itérations permettent de réduire encore le nombre de points de choix, mais le temps mis pour effectuer les renforcements étant plus important, les temps d'exécution sont augmentés par rapport à l^1.

La figure 5 nous permet également de comparer l'efficacité des étiquetages $l^*_{\leq}$ basés sur des ordres totaux par rapport à ceux basés sur des ordres partiels. En effet, l'utilisation d'un ordre total introduit une première approximation, et l'algorithme calculant l'ordre

	vflib	c2+*AllDiff*		c2+*AllDiff*+c3		l^*+FC(c2+*AllDiff*)			$l^*_{\leq}$+FC(c2+*AllDiff*)			
		FC	AC	FC	AC	l^0	l^1	l^2	$l^2_{\leq}$	$l^4_{\leq}$	$l^8_{\leq}$	$l^\infty_{\leq}$
SIP-300	0%	0%	5%	33%	20%	80%	60%	0%	75%	80%	**85%**	0%
20-300-2.5	-	-	361.98	319.54	397.64	126.72	98.57	-	35.16	18.77	**36.68**	-
90-90	-	-	154	21	7	4438	159	-	39	13	**7**	-
	-	-	-	-	-	-	-	-	1.93	3.87	**7.84**	-
SIP-300	0%	0%	0%	10%	5%	23%	20%	0%	38%	63%	68%	**73%**
20-300-2.5	-	-	-	381.74	346.50	186.40	109.91	-	45.01	10.45	3.94	**5.15**
90-(90+10)	-	-	-	52	14	18958	3304	-	2323	481	107	**20**
	-	-	-	-	-	-	-	-	1.78	3.87	5.74	**6.1**

FIG. 6 – Résultats sur des instances pour lesquelles $n = 300$, d_{min}=20, d_{max}=300, λ=2.5 et p_s=p_a=90%). La première classe contient des instances satisfiables, la seconde des instances insatisfiables pour lesquelles on a ajouté aléatoirement 10% d'arêtes dans le graphe motif. Pour chaque classe d'instances, on donne le pourcentage d'instances résolues en moins de 600s, le temps d'exécution moyen et le nombre de feuilles échec dans l'arbre de recherche pour les instances résolues ; pour les approches basées sur un étiquetage renforcé itérativement, on donne le nombre moyen d'itérations effectuées à chaque étape.

total est basé sur une heuristique sans garantie d'optimalité, de sorte que la qualité de l'approximation ne peut être évaluée qu'expérimentalement. On constate en pratique que l^2 est plus fort que $l^2_{\leq}$ qui est lui même plus fort que l^1. Si pour n=200 le nombre de feuilles échec pour $l^2_{\leq}$ est à peine inférieur au nombre de feuilles échec pour l^1, pour n=600 et n=1000, le nombre de feuilles échec pour $l^2_{\leq}$ est plus proche de l^2 que de l^1, montrant que l'approximation est plutôt bonne. En ce qui concerne les temps d'exécution, on constate en pratique que les temps de résolution obtenus avec $l^2_{\leq}$ sont très nettement inférieurs à ceux obtenus avec l^2.

La quatrième classe de la figure 5 contient des instances orientées. Pour ces instances faciles, vflib et l^0 obtiennent les meilleurs résultats, l'augmentation du nombre d'itérations ne faisant qu'augmenter les temps de résolution pour l^* et $l^*_{\leq}$. Notons que pour ces instances, l'introduction de la contrainte redondante c3 a un effet très négatif sur la résolution.

La figure 6 permet de comparer les approches sur des graphes plus denses et moins réguliers : les instances de la figure 5 sont assez peu denses et relativement "régulières", les degrés des sommets étant toujours compris entre 5 et 8 ; on considère maintenant des graphes plus denses et moins réguliers, où les degrés des sommets varient entre 20 et 300, le nombre de sommets étant égal à 300. Ces instances apparaissent clairement plus difficiles, et ne sont jamais résolues en moins de 600 secondes par vflib, et que rarement résolues par les modèles classiques de la PPC, les meilleurs résultats étant obtenus par une propagation par forward checking sur les contraintes c2, c3 et *AllDiff*. En revanche, sur ces instances, un filtrage basé sur des étiquetages renforcés permet de réduire à la fois la combinatoire et les temps d'exécution de façon drastique. Il est intéressant de noter que pour ces instances les meilleurs résultats sont obtenus avec des étiquetages obtenus après un grand nombre de renforcements (basés sur des ordres totaux, les renforcements basés sur des ordres partiels augmentant trop les temps de résolution). Ce comportement est également observé pour les instances non satisfiables de la seconde ligne : pour ces instances, les meilleurs résultats sont obtenus lorsque les étiquetage sont renforcés jusqu'à l'obtention du point fixe ; ce point fixe est obtenu, en moyenne, après 6 itérations. Notons que pour les instances satisfiables de la première ligne, $l^\infty_{\leq}$ ne permet de résoudre aucune instance en moins de 600s car le nombre d'itérations nécessaires pour atteindre le point fixe est trop élevé.

6 Conclusion

Nous avons introduit un nouvel algorithme de filtrage pour le problème de l'isomorphisme de sous-graphe, basé sur un renforcement itératif d'étiquetages. Les premières expérimentations montrent que ce filtrage est plus efficace que les approches de l'état de l'art pour les graphes "scale free" dont les degrés suivent une distribution exponentielle. Cependant, le nombre d'itérations de renforcement apparait un paramètre critique : pour les instances très faciles, les meilleurs résultats sont obtenus par l'étiquetage initial $l^0 = l_{deg}$; pour les instances de difficulté intermédiaire, les meilleurs résultats sont obtenus par l'étiquetage renforcé une fois l^1 ; pour les instances les plus difficiles, les meilleurs résultats sont obtenus par des étiquetages obtenus après un plus grand nombre de renforcements, pouvant aller jusqu'au point fixe.

Ainsi, nos perspectives concernent principalement la recherche de critères de terminaison dynamiques, per-

mettant d'adapter le nombre d'itérations de renforcement à l'instance à résoudre. Nous souhaitons également étudier les possibilités d'intégration de ces techniques de filtrage avec des techniques d'élimination de symétries [13, 21].

Références

[1] Albert-Laszlo Barabasi. *Linked : How Everything Is Connected to Everything Else and What It Means.* Plume, 2003.

[2] Christian Bessière and Pascal Van Hentenryck. To be or not to be a global constraint. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming ? CP 2003 : 9th International Conference, CP 2003*, pages 789–794. Springer, September 2003.

[3] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 18(3) :265–298, 2004.

[4] Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pages 149–159. Cuen, 2001.

[5] P. T. Darga, M. H. Liffiton, K. A. Sakallah, and I. L. Markov. Exploiting structure in symmetry detection for cnf. In *Proc. Design Automation Conference (DAC)*, pages 530–534. IEEE/ACM, June 2004.

[6] Yves Deville, Grégoire Dooms, Stéphane Zampelli, and Pierre Dupont. Cp(graph+map) for approximate graph matching. *1st International Workshop on Constraint Programming Beyond Finite Integer Domains, CP2005*, pages 33–48, 2005.

[7] Grégoire Dooms, Yves Deville, and Pierre Dupont. Cp(graph) : Introducing a graph computation domain in constraint programming. *Principles and Pratice of Constraint Programming*, pages 211–225, 2005.

[8] M. R. Garey and David S. Johnson. *Computer and Intractability : A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[9] Jiong Guo, Falk Hüffner, and Hannes Moser. Feedback arc set in bipartite tournaments is np-complete. *Information Processing Letters*, 102(2-3) :62–65, 2007.

[10] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4) :225–231, 1973.

[11] Javier Larrosa and Gabriel Valiente. Constraint satisfaction algorithms for graph pattern matching. *Mathematical. Structures in Comp. Sci.*, 12(4) :403–422, 2002.

[12] B. D. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30 :45–87, 1981.

[13] Jean-François Puget. Breaking symmetries in all different problems. *Fourth International Workshop on Symmetry in Constraint Satisfaction Problems (SymCon'04)*, 2004.

[14] J.-C. Regin. A filtering algorithm for constraints of difference in CSPs. In *Proc. 12th Conf. American Assoc. Artificial Intelligence*, volume 1, pages 362–367. Amer. Assoc. Artificial Intelligence, 1994.

[15] Jean-Charles Régin. *Développement d'Outils Algorithmiques pour l'Intelligence Artificielle. Application à la Chimie Organique.* PhD thesis, 1995.

[16] Michael Rudolf. Utilizing constraint satisfaction techniques for efficient graph pattern matching. In *Theory and Application of Graph Transformations*, number 1764 in Lecture Notes in Computer Science, pages 238–252. Springer, 1998.

[17] Sébastien Sorlin and Christine Solnon. A new filtering algorithm for the graph isomorphism problem. *3rd International Workshop on Constraint Propagation and Implementation, CP2006*, 2006.

[18] Julian R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1) :31–42, 1976.

[19] Fabien Viger and Matthieu Latapy. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. In Lusheng Wang, editor, *COCOON*, volume 3595 of *Lecture Notes in Computer Science*, pages 440–449. Springer, 2005.

[20] Stéphane Zampelli, Yves Deville, and Pierre Dupont. Approximate constrained subgraph matching. *Principles and Pratice of Constraint Programming*, 2005.

[21] Stéphane Zampelli, Yves Deville, and Pierre Dupont. Symmetry breaking in subgraph pattern matching. *Sixth International Workshop on Symmetry in Constraint Satisfaction Problems (SymCon'06)*, 2006.

[22] Stéphane Zampelli, Yves Deville, Christine Solnon, Sébastien Sorlin, and Pierre Dupont. Filtering for subgraph matching. Technical Report 1054, Université Catholique de Louvain, 2007. available at http ://www.info.ucl.ac.be/∼sz/FSI-INGIReport1054.pdf.