Une contrainte globale pour le problème de l'isomorphisme de graphes

Sébastien Sorlin, Christine Solnon

LIRIS, CNRS FRE2672, bât. Nautibus, Université Lyon I
43 Bd du 11 novembre, 69622 Villeurbanne cedex, France
{sebastien.sorlin,christine.solnon}@liris.cnrs.fr

RÉSUMÉ. *Le problème de l'isomorphisme de graphes consiste à prouver que deux graphes donnés ont la même structure. Ce problème peut très facilement être modélisé en un problème de satisfaction de contraintes puis être résolu par un solveur de contraintes. Toutefois, sur ce type de problèmes, la programmation par contraintes est bien moins efficace que les algorithmes dédiés qui sont capables de tirer partie de la sémantique globale du problème. Nous introduisons dans cet article une nouvelle contrainte globale dédiée au problème de l'isomorphisme de graphes. Nous définissons ensuite l'algorithme de filtrage associé à cette contrainte. Celui-ci exploite les arêtes du graphe de façon globale afin de réduire le domaine des variables. Nous montrons aussi que cette contrainte globale est décomposable en un ensemble de "contraintes de distance" propageant mieux les réductions des domaines que les "contraintes d'arêtes" habituellement utilisées pour ce problème.*

ABSTRACT. *The graph isomorphism problem consists in deciding if two given graphs have an identical structure. This problem can be modeled as a constraint satisfaction problem in a very straightforward way, so that one can use constraint programming to solve it. However, constraint programming is a generic tool that may be less efficient than dedicated algorithms which can take advantage of the global semantic of the original problem. Hence, we introduce in this paper a new global constraint dedicated to graph isomorphism problems, and we define an associated filtering algorithm that exploits all edges of the graphs in a global way to narrow variable domains. We then show how this global constraint can be decomposed into a set of "distance" constraints which propagate more domain reductions than "edge" constraints that are usually generated for this problem.*

MOTS-CLÉS : *Isomorphisme de graphes, contrainte globale*

KEYWORDS: *Graph isomorphism, global constraint*

1. Introduction

Les graphes sont largement utilisés pour représenter des objets structurés tels que les molécules, les images, les réseaux... De nombreuses applications nécessitent de comparer des graphes afin de déterminer si leurs structures sont identiques : c'est le problème de l'isomorphisme de graphes (GIP).

Il existe de nombreux algorithmes dédiés au GIP ([ULL 76, MCK 81, COR 01]). Ces algorithmes sont très efficaces en pratique, même si leur complexité dans le pire des cas est exponentielle. Cependant, ces algorithmes sont difficilement utilisables pour résoudre des problèmes plus généraux (***e.g.***, un GIP auxquel des contraintes supplémentaires ont été ajoutées).

Une alternative intéressante à ces algorithmes dédiés est l'utilisation de la programmation par contrainte (PPC) qui fournit un cadre générique pour la résolution des problèmes de satisfaction de contraintes (CSP). En effet, les GIPs peuvent être aisément transformés en CSPs [MCG 79] et il est possible d'utiliser un solveur générique de contraintes pour les résoudre. Cependant, la transformation d'un GIP en CSP, entraîne la perte de la sémantique globale du problème et la PPC est moins efficace pour résoudre les problèmes d'isomorphisme que les algorithmes dédiés qui gardent une vue globale du problème.

Le but de notre travail est de permettre aux solveurs de contraintes d'appréhender les GIPs de façon globale afin qu'ils puissent les résoudre efficacement sans pour autant perdre la souplesse de la PPC. Pour cela, nous introduisons une contrainte globale dédiée aux GIPs et nous montrons comment exploiter cette globalité pour résoudre efficacement les GIPs.

La section 2 définit formellement le GIP et propose un rapide survol de la complexité des GIPs et des méthodes de résolution de ces problèmes. La section 3 présente les propriétés des GIPs utilisées par la suite pour définir notre algorithme de filtrage. Nous introduisons en section 4 une nouvelle contrainte globale pour modéliser les GIPs sur des graphes non-orientés ainsi que la technique de filtrage qui lui est associée. Lorsque ce filtrage n'est pas suffisant pour résoudre le problème, nous proposons de remplacer cette contrainte globale par un ensemble de contrainte binaires appelées "*contraintes de distance*". Nous discutons en section 5 de l'extension de notre travail aux graphes orientés et au problème de l'isomorphisme de sous-graphe.

2. Le problème de l'isomorphisme de graphes

Définitions : Un *graphe* G est défini par un couple (V, E), V étant l'ensemble fini des sommets de G et $E \subseteq V \times V$ l'ensemble des arcs de G. Nous nous intéresserons tout particulièrement aux graphes sans boucles, ***i.e.***, $\forall (u, v) \in E, u \neq v$. Deux graphes $G = (V, E)$ et $G' = (V', E')$ sont *isomorphes* s'il existe une fonction bijective $f : V \rightarrow V'$ telle que $(u, v) \in E \Leftrightarrow (f(u), f(v)) \in E'$. La fonction f est alors appelée

une *fonction d'isomorphisme*. Un GIP consiste à déterminer si deux graphes donnés sont isomorphes.

Complexité. Dans le cas général, la complexité théorique du GIP n'est pas précisement établie : le problème est clairement dans NP mais on ne sait pas s'il est dans P ou s'il est NP-complet [FOR 96] ; la classe des problèmes *isomorphisme-complets* a donc été définie. Cependant, il a été montré que pour certains types de graphes (les graphes planaires [HOP 74], les arbres [AHO 74], les graphes à degré borné [LUK 82]. ..), GIP est un problème de complexité polynomiale.

Algorithmes dédiés. Il est possible de résoudre un GIP en recherchant directement un appariement des sommets des deux graphes. L'espace de recherche composé de tous les appariements est exploré par *séparation et évaluation* et l'élaguage de l'arbre de recherche se fait en exploitant les propriétés des graphes manipulés [COR 01, ULL 76] (distribution des arcs, voisinage des sommets...). Cette approche, très efficace, permet de résoudre des problèmes de plus de 1000 sommets très rapidement (moins d'une seconde).

[MCK 81] propose une autre approche qui consiste à calculer, pour chaque sommet v_i d'un graphe, une étiquette unique calculée à partir d'un ensemble d'invariants de sommets (*i.e.*, un ensemble de caractéristiques décrivant les relations de v_i avec les autres sommets du graphe) et telle que deux sommets ont la même étiquette si et seulement s'ils peuvent être appariés dans une fonction d'isomorphisme. C'est l'approche utilisée par *nauty*, le solveur de problèmes d'isomorphismes le plus efficace. L'efficacité de *nauty* est comparable à celle des méthodes par séparation et évaluation mais *nauty* est toujours le plus rapide pour les gros graphes [FOG 01].

Si les algorithmes dédiés sont très efficaces pour résoudre des GIPs (malgré leur complexité exponentielle dans le pire des cas), ils ne permettent pas la résolution de problèmes plus généraux tels que des GIPs auxquels des contraintes ont été ajoutées. En particulier, il est fréquent que les sommets et les arcs des graphes soient étiquetés et que l'on recherche une fonction d'isomorphisme respectant certaines contraintes sur ces étiquettes. C'est le cas, par exemple, dans [RÉG 95] où les graphes représentent des molécules, ou en CAO où les graphes représentent des objets de conception [CHA 03].

Programmation par contraintes. La programmation par contraintes (PPC) est un outil générique pour la résolution de problèmes de satisfaction de contraintes (CSP). Elle peut donc être utilisée pour résoudre des GIPs. Un *CSP* [TSA 93] est défini par un triplet (X, D, C) où :

– X est un ensemble fini de variables,

– D est une fonction associant à chaque variable $x_i \in X$ son domaine $D(x_i)$, *i.e.*, l'ensemble des valeurs qui peuvent lui être associées

– C est un ensemble de contraintes, *i.e.*, de relations entre des variables restreignant l'ensemble des valeurs que les variables peuvent prendre simultanément.

Un *CSP binaire* ne contient que des contraintes binaires, *i.e.*, des contraintes ne portant que sur exactement deux variables. La contrainte binaire entre les variables x_i et x_j est notée $C(x_i, x_j)$ et est définie comme l'ensemble des couples $(v_i, v_j) \in D(x_i) \times D(x_j)$ qui satisfont la contrainte.

Résoudre un CSP (X, D, C) consiste à trouver un assignement complet (*i.e.*, affectant une valeur $v_i \in D(x_i)$ à chacune des variables $x_i \in X$) et tel que toutes les contraintes de C sont satisfaites.

Les CSPs peuvent être résolus en utilisant un solveur de contraintes intégré dans un langage de programmation par contraintes (CHOCO [LAB 00], Ilog solver [ILO 00], CHIP [AGG 92]...). Ces solveurs sont basés sur l'exploration complète de l'espace de recherche et sur la réduction de cet espace par des techniques de filtrage du domaine des variables telle que la consistance d'arcs (AC) [TSA 93, MOH 86, BES 01].

Utilisation de la PPC pour résoudre un GIP. Un GIP peut être aisément formulé en CSP. Il est alors possible d'utiliser la PPC pour le résoudre [GAR 79, RÉG 95]. Etant donnés deux graphes $G = (V, E)$ et $G' = (V', E')$, nous définissons le CSP (X, D, C) suivant :

– une variable x_u est associée à chaque sommet $u \in V$, *i.e.*, $X = \{x_u / u \in V\}$,

– le domaine de chaque variable x_u est l'ensemble des sommets de G' ayant le même nombre d'arcs entrants et le même nombre d'arcs sortants que u, *i.e.*,

$$D(x_u) = \{u' \in V' \,/\quad |\{(u,v) \in E\}| = |\{(u',v') \in E'\}| \text{ et } |\{(v,u) \in E\}| = |\{(v',u') \in E'\}|\}$$

– Il existe une contrainte binaire $C_{edge}(x_u, x_v)$ entre chaque paire de variables (x_u, x_v) exprimant le fait que les sommets de G' assignés à x_u et x_v doivent être connectés par un arc si et seulement si $(u, v) \in E$, *i.e.*,

$$\begin{array}{ll} \text{si } (u,v) \in E, & C_{edge}(x_u, x_v) = E' \\ \text{sinon} & C_{edge}(x_u, x_v) = \{(u',v') \in V'^2 \mid u' \neq v' \text{ et } (u',v') \notin E'\} \end{array}$$

Discussion. Lors de la formulation d'un GIP en CSP, la sémantique globale du problème est décomposée en un ensemble de contraintes binaires d'arcs (C_{edge}), chacune d'elles exprimant localement la présence ou l'absence d'un arc. Cela a pour conséquence de rendre la PPC moins efficace que les algorithmes dédiés.

Afin d'améliorer la résolution des CSPs associés à des GIPs, il est possible d'ajouter une contrainte globale *allDiff* [RÉG 95]. La contrainte *allDiff* permet d'exprimer de façon globale qu'un ensemble de variables doivent avoir des valeurs différentes deux à deux. Par rapport aux définitions de globalité proposées par [BES 03], *allDiff* n'est pas sémantiquement globale : elle peut toujours être décomposée, sans modifier les solutions du CSP, en un ensemble de contraintes binaires sémantiquement équivalent exprimant la différence entre chaque couple de variables. La contrainte *allDiff* est par contre AC-opérationnellement globale : un filtrage par AC de cette contrainte permet de réduire plus fortement le domaine des variables qu'un filtrage par AC sur l'ensemble de contraintes binaires equivalent. [RÉG 95] propose un algorithme de filtrage

dédié à la contrainte *allDiff* : celui-ci, basé sur la théorie des couplages, exploite de façon globale la sémantique de la contrainte pour filtrer très efficacement le domaine des variables.

Dans cet article, nous introduisons une nouvelle contrainte globale pour modéliser les GIPs. Cette contrainte n'est pas sémantiquement globale, elle peut être remplacée par l'ensemble des contraintes binaires présenté plus haut. Cependant, elle permet de considérer les arcs des graphes de façon globale et ainsi filtrer plus efficacement l'espace de recherche. Notons que cette contrainte peut être combinée avec la contrainte *allDiff* afin de filtrer encore plus de valeurs.

3. Quelques propriétés du GIP

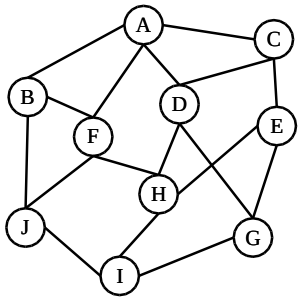
Une fonction d'isomorphisme n'associe que des sommets *similaires*. Les méthodes basées sur les *invariants de sommets* utilisent cette propriété pour filtrer l'espace de recherche d'un GIP. Un invariant de sommet est une étiquette $l(v)$ assignée à un sommet v telle que, s'il existe une fonction d'isomorphisme associant v et v', alors $l(v) = l(v')$ (l'inverse n'étant cependant pas toujours vrai). L'exemple le plus simple d'invariants de sommets est le degré d'un sommet (*i.e.*, ses nombres d'arcs entrants et sortants) : si f est une fonction d'isomorphisme entre $G = (V, E)$ et $G' = (V', E')$, alors pour chaque sommet $v \in V$, les sommets v et $f(v)$ ont le même degré.

Nous introduisons dans cette section quelques définitions et théorèmes utilisés par la suite pour définir un nouvel invariant de sommets. Nous focaliserons notre attention sur les graphes non-orientés, *i.e.*, des graphes avec des arêtes non-orientées (les arêtes (u, v) et (v, u) sont considérées comme identiques). L'extension de notre travail aux graphes orientés est discutée en section 5.1. Nous supposerons que les graphes sont connexes, *i.e.*, que chaque sommet est accessible à partir de tous les sommets.

3.1. *Définitions et théorèmes*

Définition 1. Soit un graphe $G = (V, E)$, un *chemin* entre deux sommets u et v est une suite $< v_0, v_1, v_2, ..., v_k >$ de sommets telle que $v_0 = u$, $v_k = v$ et telle que pour tout $i \in [1, k]$, $(v_{i-1}, v_i) \in E$. La longueur d'un chemin π, notée $|\pi|$, est son nombre d'arêtes, *i.e.*, k.

Définition 2. Soit un graphe $G = (V, E)$, un *plus court chemin* entre deux sommets u et v est un chemin entre u et v de longueur minimale. La *longueur d'un plus court chemin* entre u et v est notée $\delta_G(u, v)$. Nous dirons que $\delta_G(u, v)$ est la *distance* entre u et v.



$\delta_G(u,v)$	A	B	C	D	E	F	G	H	I	J
A	0	1	1	1	2	1	2	2	3	2
B	1	0	2	2	3	1	3	2	2	1
C	1	2	0	1	1	2	2	2	3	3
D	1	2	1	0	2	2	1	1	2	3
E	2	3	1	2	0	2	1	1	2	3
F	1	1	2	2	2	0	3	1	2	1
G	2	3	2	1	1	3	0	2	1	2
H	2	2	2	1	1	1	2	0	1	2
I	3	2	3	2	2	2	1	1	0	1
J	2	1	3	3	3	1	2	2	1	0

Figure 1. *Un graphe $G = (V, E)$ et les distances entre chaque couple de sommets.*

Théorème 1. Soient deux graphes $G = (V, E)$ et $G' = (V', E')$ tels que $|V| = |V'|$, et une fonction bijective $f : V \rightarrow V'$, les deux propositions suivantes sont equivalentes :

$$f \text{ est une fonction d'isomorphisme, } \textit{i.e.,} (u, v) \in E \Leftrightarrow (f(u), f(v)) \in E' \qquad (1)$$

$$\forall (u, v) \in V^2, \delta_G(u, v) = \delta_{G'}(f(u), f(v)) \qquad (2)$$

Preuve. $(1) \Rightarrow (2)$: si f est une fonction d'isomorphisme, alors (u, v) est une arête de G si et seulement si $(f(u), f(v))$ est une arête de G' donc $< v_1, v_2, ..., v_n >$ est un chemin dans G si et seulement si $< f(v_1), f(v_2), ..., f(v_n) >$ est un chemin dans G', et de sorte que $< v_1, v_2, ..., v_n >$ est un plus court chemin dans G si et seulement si $< f(v_1), f(v_2), ..., f(v_n) >$ est un plus court chemin dans G'. La propriété 2 est donc vérifiée.
$(2) \Rightarrow (1)$: Pour chaque paire de sommets $(u, v) \in V \times V$, si (u, v) est une arête de G, alors $< u, v >$ est le plus court chemin entre u et v, $\delta_G(u, v) = 1$ et donc $\delta_{G'}(f(u), f(v)) = 1$. $(f(u), f(v))$ est donc un arc de G' (et vice versa).

Le théorème 1 sera utilisé pour définir les "*contraintes de distance*" permettant la propagation des réductions de domaines lors de la résolution des GIPs par un solveur de contraintes. Nous introduisons maintenant quelques définitions utilisées par la suite pour définir une consistance partielle et un algorithme de filtrage pour les GIPs.

Définition 3. Soit un graphe $G = (V, E)$, un sommet $u \in V$ et une distance $i \in [0, |V| - 1]$, nous notons $\Delta_G(u, i)$ l'ensemble des sommets à une distance de i du sommet u et $\#\Delta_G(u, i)$ le nombre de ces sommets, *i.e.,*

$$\Delta_G(u, i) = \{v \in V / \delta_G(u, v) = i\} \quad \text{et} \quad \#\Delta_G(u, i) = |\Delta_G(u, i)|$$

Par exemple, pour le graphe G de la figure 1, nous avons :

$$\begin{array}{ll}
\Delta_G(A, 0) = \{A\} & \#\Delta_G(A, 0) = 1 \\
\Delta_G(A, 1) = \{B, C, D, F\} & \#\Delta_G(A, 1) = 4 \\
\Delta_G(A, 2) = \{E, G, H, J\} & \#\Delta_G(A, 2) = 4 \\
\Delta_G(A, 3) = \{I\} & \#\Delta_G(A, 3) = 1 \\
\Delta_G(A, i) = \emptyset & \#\Delta_G(A, i) = 0, \quad \forall i \geq 4
\end{array}$$

Définition 4. Soit un graphe $G = (V, E)$ et un sommet $u \in V$, nous notons $\#\Delta_G(u)$ la séquence composée de $|V|$ nombres correspondant respectivement aux nombres de sommets à une distance de 0, 1, ... ,$|V| - 1$ de u, *i.e.*,

$$\#\Delta_G(u) = < \#\Delta_G(u,0), \#\Delta_G(u,1), ..., \#\Delta_G(u,|V|-1) >$$

Nous omettrons les zéros présents à la fin de la séquence.
Par exemple, les séquences des sommets du graphe G de la figure 1 sont :
$\#\Delta_G(A) = \#\Delta_G(D) = \#\Delta_G(F) = < 1,4,4,1 >$
$\#\Delta_G(B) = \#\Delta_G(C) = \#\Delta_G(E) = \#\Delta_G(G) = \#\Delta_G(I) = \ < 1,3,4,2 >$
$\#\Delta_G(H) = < 1,4,5,0 >$
$\#\Delta_G(J) = < 1,3,3,3 >$

Définition 5. Soit un graphe $G = (V, E)$, nous notons $\#\Delta_G$ l'ensemble de toutes les séquences associées aux sommets de G, *i.e.*,

$$\#\Delta_G = \{s | \exists u \in V, s = \#\Delta_G(u)\}$$

Par exemple, l'ensemble de toutes les séquences présentes dans le graphe G de la figure 1 est :

$$\#\Delta_G = \{< 1,3,3,3 >, < 1,3,4,2 >, < 1,4,4,1 >, < 1,4,5,0 >\}$$

Chaque séquence $\#\Delta_G(u)$ caractérise les relations en termes de distance entre le sommet u et les autres sommets de G. Lorsqu'on cherche un isomorphisme de graphes, il est possible d'utiliser ces séquences pour réduire l'espace de recherche en éliminant les appariements qui associent des sommets ayant des séquences différentes. Plusieurs sommets du même graphe peuvent cependant avoir la même séquence, ce critère n'est donc pas toujours suffisant pour élaguer efficacement l'espace de recherche. Par exemple, dans le graphe de la figure 1, cinq sommets ont pour séquence $< 1,3,4,2 >$. La définition 6 va une étape plus loin afin de caractériser plus précisément les relations d'un sommet u avec les autres sommets du graphe.

Définition 6. Soit un sommet $u \in V$, nous notons $label_G(u)$ l'ensemble de tous les triplets (i, s, k) tels que i est une distance, s est une séquence, et k est le nombre de sommets distants de i du sommet u et dont la séquence est s, *i.e.*,

$$label_G(u) = \{(i,s,k) \ / \quad \begin{array}{l} i \in [0, |V|-1], \\ s \in \#\Delta_G, \text{ and} \\ k = |\{v \in \Delta_G(u,i) / \#\Delta_G(v) = s\}| \} \end{array}$$

Nous ne considérerons que les triplets (i, s, k) tels que $k > 0$.
Par exemple, pour le graphe G de la figure 1, nous avons :

$$\begin{array}{ll} label_G(A) = \{ & (0, < 1,4,4,1 >, 1), \\ & (1, < 1,3,4,2 >, 2), (1, < 1,4,4,1 >, 2), \\ & (2, < 1,3,3,3 >, 1), (2, < 1,3,4,2 >, 2), (2, < 1,4,5,0 >, 1), \\ & (3, < 1,3,4,2 >, 1)\} \end{array}$$

car il y a un sommet (A) distant de 0 de A et dont la séquence est $< 1, 4, 4, 1 >$, deux sommets (B et C) distants de 1 de A dont la séquence est $< 1, 3, 4, 2 >$, deux autres sommets (D et F) distants eux aussi de 1 du sommet A mais dont la séquence est $< 1, 4, 4, 1 >$, etc...

Théorème 2. Soient deux graphes $G = (V, E)$ et $G' = (V', E')$. S'il existe une fonction d'isomorphisme $f : V \rightarrow V'$ entre les deux graphes G et G', alors, pour chaque sommet $u \in V$, $label_G(u) = label_{G'}(f(u))$.

Preuve. f est une bijection et la distance entre deux sommets u et v de G est égale à la distance entre leur images par f (voir le théorème 1), *i.e.*, $\delta_G(u, v) = \delta_{G'}(f(u), f(v))$. Par conséquent, le nombre de sommets de G distants de i du sommet u est égal au nombre de sommets de G' distants de i du sommet $f(u)$, donc $\#\Delta_G(u) = \#\Delta_{G'}(f(u))$. Par conséquent, les ensembles $\#\Delta$ de séquences des deux graphes sont égaux, *i.e.*, $\#\Delta_G = \#\Delta_{G'}$. Pour chaque séquence $s \in \#\Delta_G$, et pour chaque sommet $u \in V$, le nombre de sommets distants de i du sommet u et dont la séquence est s est égal au nombre de sommets distants de i du sommet $f(u)$ ayant également pour séquence s. Nous avons donc $label_G(u) = label_{G'}(f(u))$.

La réciproque du théorème 2 n'est cependant pas toujours vraie : en effet, il peut exister deux sommets ayant le même label et tels qu'il n'existe aucune fonction d'isomorphisme les reliant l'un à l'autre.

3.2. *Algorithmes et complexités*

Nous discutons dans cette section de la complexité en temps et en espace du calcul des différentes valeurs introduites en 3.1. Ces complexités sont données pour un graphe $G = (V, E)$ non-orienté et connexe tel que $|V| = n$ et $|E| = p$ avec $n - 1 \leq p < n^2$.

$\delta_G(u, v)$. Toutes les définitions introduites en section 3.1 sont basées sur les distances entre chaque paire de sommets. Pour calculer ces distances, il est nécessaire d'effectuer un parcours en largeur [COR 90] à partir de chacun des sommets de G : n parcours de complexité en $\mathcal{O}(p)$ sont donc nécessaires. La complexité en temps est donc en $\mathcal{O}(np)$. L'espace nécessaire à la mémorisation des distances est en $\mathcal{O}(n^2)$ (une distance par paire de sommets).

$\Delta_G(u, i)$, $\#\Delta_G(u, i)$ et $\#\Delta_G(u)$. Toutes ces valeurs peuvent être calculées de façon incrémentale pendant le calcul des plus courts chemins : à chaque calcul d'un nouveau $\delta_G(u, v)$, le sommet u (resp. v) est ajouté à l'ensemble $\Delta_G(v, \delta_G(u, v))$ (resp. $\Delta_G(u, \delta_G(u, v))$), les ensembles $\#\Delta_G(u, \delta_G(u, v))$ et $\#\Delta_G(v, \delta_G(u, v))$ sont incrémentés, et les séquences $\#\Delta_G(u)$ et $\#\Delta_G(v)$ sont mises à jour en incrémentant leur $\delta_G(u, v)^{\text{ième}}$ élément. Toutes ces opérations s'effectuent en temps constant. L'espace nécessaire à la mémorisation de ces informations est en $\mathcal{O}(n^2)$ (pour chaque sommet $u \in V$, les ensembles $\Delta_G(u, i)$ forment une partition des n sommets de $|V|$).

$label_G(u)$**.** Afin de calculer et de comparer efficacement les labels, nous commençons par trier les ensembles de séquences afin qu'un entier unique soit associé à chaque séquence. Cette opération nécessite $\mathcal{O}(n^2.log(n))$ opérations car il y a au plus n séquences différentes et que la comparaison de deux séquences est une opération en $\mathcal{O}(n)$. Une fois cette opération effectuée, le calcul des labels est réalisé en $\mathcal{O}(n^2)$ opérations (il y a n labels à calculer et chaque label contient au plus n différents triplets) et leur mémorisation nécessite une mémoire en $\mathcal{O}(n^2)$ (car chaque label contient au plus n triplets de taille constante si les séquences sont remplacées par un entier unique). Enfin, la comparaison de deux labels se fait en $\mathcal{O}(n)$ opérations, en supposant que les triplets (i, s, k) des labels soient triés.

Par conséquent, le calcul de toutes les valeurs introduites en section 3.1 pour un graphes $G = (V, E)$ nécessite $\mathcal{O}(|V|.|E| + |V|^2.log(|V|))$ opérations et une taille mémoire en $\mathcal{O}(|V|^2)$.

4. Une contrainte globale pour les GIP

Nous introduisons ici une nouvelle contrainte globale dédiée aux problèmes d'isomorphisme de graphes. Syntaxiquement, cette contrainte est définie par la relation $gip(V, E, V', E', L)$ où

– V et V' sont deux ensembles de valeurs tels que $|V| = |V'|$,

– $E \subseteq V \times V$ est un ensemble de couples de valeurs de V,

– $E' \subseteq V' \times V'$ est un ensemble de couples de valeurs de V',

– L est un ensemble de couples qui associent une variable différente du CSP à chaque valeur de V, *i.e.*, L est un ensemble de $|V|$ couples (x_u, u) où x_u est une variable du CSP, u une valeur de V et pour toute paire de couples (x_u, u) et (x_v, v) différents de L, x_u et x_v sont des variables différentes et $u \neq v$.

Sémantiquement, la contrainte globale $gip(V, E, V', E', L)$ est consistante si et seulement s'il existe une fonction d'isomorphisme $f : V \rightarrow V'$ telle que pour chaque couple $(x_u, u) \in L$, il existe une valeur $u' \in D(x_u)$ telle que $u' = f(u)$.

Cette contrainte globale n'est pas sémantiquement globale dans le sens où elle est sémantiquement équivalente à l'ensemble des contraintes binaires décrit en section 2. La contrainte gip nous permet cependant d'utiliser la sémantique globale des GIPs afin de les résoudre plus efficacement. Nous définissons maintenant une consistance partielle ainsi que l'algorithme de filtrage qui lui est associé (section 4.1). Nous décrirons ensuite comment propager les contraintes (section 4.2).

4.1. *Label-consistance et filtrage par labels pour la contrainte* gip

Le théorème 2 montre qu'une fonction d'isomorphisme n'associe que des sommets qui ont le même label. Nous définissons donc une consistance partielle (la *label-consistance*) pour la contrainte gip, qui garantit que pour chaque couple $(x_i, v) \in L$, le domaine de x_i ne contient que des valeurs ayant le même label que v.

Définition 7. La contrainte globale $gip(V, E, V', E', L)$ est label-consistante si et seulement si :

$$\forall (x_u, u) \in L,\ \forall u' \in D(x_u),\ label_{(V,E)}(u) = label_{(V',E')}(u')$$

Pour rendre la contrainte label-consistante, il suffit de calculer les labels de chaque sommet des deux graphes (section 3.2) et d'enlever du domaine de chaque variable x_u associée à un sommet $u \in V$ toutes les valeurs $u' \in D(x_u)$ telles que $label_{(V,E)}(u) \neq label_{(V',E')}(u')$.

Le filtrage par labels réduit souvent très fortement les domaines des variables. Considérons par exemple le graphe G de la figure 1. Les trois premiers triplets des labels de chaque sommet (triés par distance, puis par séquence croissante) sont :

$$\begin{array}{ll}
label_G(A) = & \{(0,<1,4,4,1>,1),(1,<1,3,4,2>,2),(1,<1,4,4,1>,2),...\} \\
label_G(B) = & \{(0,<1,3,4,2>,1),(1,<1,3,3,3>,1),(1,<1,4,4,1>,2),...\} \\
label_G(C) = & \{(0,<1,3,4,2>,1),(1,<1,3,4,2>,1),(1,<1,4,4,1>,2),...\} \\
label_G(D) = & \{(0,<1,4,4,1>,1),(1,<1,3,4,2>,2),(1,<1,4,4,1>,1),...\} \\
label_G(E) = & \{(0,<1,3,4,2>,1),(1,<1,3,4,2>,2),(1,<1,4,5,0>,1),...\} \\
label_G(F) = & \{(0,<1,4,4,1>,1),(1,<1,3,3,3>,1),(1,<1,3,4,2>,1),...\} \\
label_G(G) = & \{(0,<1,3,4,2>,1),(1,<1,3,4,2>,2),(1,<1,4,4,1>,1),...\} \\
label_G(H) = & \{(0,<1,4,5,0>,1),(1,<1,3,4,2>,2),(1,<1,4,4,1>,2),...\} \\
label_G(I) = & \{(0,<1,3,4,2>,1),(1,<1,3,3,3>,1),(1,<1,3,4,2>,1),...\} \\
label_G(J) = & \{(0,<1,3,3,3>,1),(1,<1,3,4,2>,2),(1,<1,4,4,1>,1),...\}
\end{array}$$

Tous les sommets de G ont des labels différents. Par conséquent, pour toute contrainte gip entre G et un autre graphe G', le filtrage par label détectera une inconsistance (si des labels de G ne sont pas présents dans G') ou réduira le domaine de chaque variable à un singleton rendant alors la consistance globale du problème facile à vérifier.

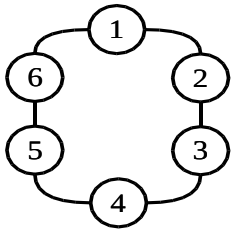
Nous pouvons comparer la consistance de labels sur la contrainte gip avec la consistance d'arcs sur le CSP de la section 2. Prenons par exemple un graphe $G' = (V', E')$ isomorphe au graphe $G = (V, E)$ de la figure 1, et tel que chaque sommet $u \in V$ est renommé en u' dans G'. Considérons maintenant le CSP "classique" modélisant le problème qui consiste à trouver une fonction d'isomorphisme entre ces deux graphes (section 2). Dans ce CSP, le domaine $D(x_u)$ de chaque variable x_u contient tous les sommets $u' \in V'$ tels que u et u' ont le même nombre d'arcs, i.e.,

$$\begin{array}{rcl}
D(x_A) = D(x_D) = D(x_F) = D(x_H) & = & \{A', D', F', H'\} \\
D(x_B) = D(x_C) = D(x_E) = D(x_G) = D(x_I) = D(x_J) & = & \{B', C', E', G', I', J'\}
\end{array}$$

Ce CSP est déja arc-consistant : le filtrage par arc consistance ne reduira donc aucun domaine. Notons aussi que sur cet exemple, ajouter une contrainte *a*llDiff ne permet pas non plus un meilleur filtrage des domaines.

4.2. *Propagation de contraintes*

Le filtrage par labels ne permet pas toujours de réduire le domaine de chacune des variables à un singleton. Considérons par exemple le graphe de la figure 2. Ce graphe comporte plusieurs symétries (il est isomorphe à n'importe quel graphe obtenu par une permutation circulaire de ses sommets). Tous les sommets sont donc associés à une même séquence et un même label. Dans ce cas, le filtrage par labels ne réduira aucun domaine.



Pour tous les sommets $u \in V$,
$\#\Delta_G(u) = < 1, 2, 2, 1 >$ et
$label_G(u) = \{ \quad (0, < 1, 2, 2, 1 >, 1),$
$(1, < 1, 2, 2, 1 >, 2),$
$(2, < 1, 2, 2, 1 >, 2),$
$(3, < 1, 2, 2, 1 >, 1)\}$

Figure 2. *Un graphe $G = (V, E)$ circulaire et les séquences et labels de ses sommets*

Quand le filtrage par labels ne réduit par le domaine de chaque variable à un singleton, il est nécessaire d'explorer l'espace de recherche composé de toutes les affectations possibles en construisant un arbre de recherche. A chaque nœud de cet arbre, le domaine d'une variable est découpé en deux sous-domaines, puis des techniques de filtrage relatives à des consistances partielles sont utilisées pour réduire le domaine des variables. Ces techniques de filtrage utilisent les contraintes afin de propager la réduction du domaine d'une variable sur les domaines des autres variables jusqu'à ce qu'un domaine devienne vide (le nœud peut alors être coupé), ou qu'un point fixe soit atteint (soit une solution est trouvée, soit le nœud doit être une fois de plus développé).

Pour utiliser une contrainte gip pour propager les réductions de domaines, une première possibilité consisterait à utiliser l'ensemble des contraintes C_{edge} tel que défini en section 2. Cependant, il est possible de tirer partie des résultats obtenus lors du filtrage par consistance de labels pour définir un ensemble de contraintes plus "fortes", *i.e.*, des contraintes définies par un nombre plus petit (ou égal) de couples de valeurs autorisées. La propagation de ces contraintes permet alors de réduire plus fortement les domaines des variables.

L'idée est de contraindre chaque paire (x_u, x_v) de variables associée à la paire (u, v) de sommets du premier graphe à prendre leurs valeurs parmi l'ensemble des paires de sommets (u', v') du second graphe tels que la distance entre u et v est égale

à la distance entre u' et v'. Comme le prouve le théorème 1, une fonction bijective entre les sommets de deux graphes est une fonction d'isomorphisme si et seulement si cette fonction préserve les distances entre chaque paire de sommets des deux graphes. La contrainte globale $gip(V, E, V', E', L)$ est donc sémantiquement équivalente à l'ensemble des "*contraintes de distance*" défini par : pour tout $((x_u, u), (x_v, v)) \in L \times L$ tel que $u \neq v$,

$$C_{distance}(x_u, x_v) = \{(u', v') \in V' \times V' \mid \delta_{(V,E)}(u, v) = \delta_{(V',E')}(u', v')\}$$

Nous pouvons facilement montrer que chaque contrainte binaire $C_{distance}(x_u, x_v)$ est au moins aussi forte que la contrainte binaire $C_{edge}(x_u, x_v)$ correspondante :

– si les sommets de G associés aux variables x_u et x_v sont reliés par une arête, alors $C_{distance}(x_u, x_v) = C_{edge}(x_u, x_v) = E'$,

– sinon, $C_{distance}(x_u, x_v) \subseteq C_{edge}(x_u, x_v)$ car $C_{edge}(x_u, x_v)$ contient tous les couples de sommets de G' qui ne sont pas connectés par un arc alors que $C_{distance}(x_u, x_v)$ ne contient que les couples de sommets de G' tels que la distance entre leurs sommets est égale à la distance entre les sommets de G associés à x_u et x_v.

Par conséquent, quelle que soit la consistance locale considérée, propager une contrainte $C_{distance}$ permettra de réduire au moins autant les domaines que la propagation de la contrainte C_{edge} correspondante, et, dans certains cas, réduira même plus fortement les domaines.

Considérons par exemple le graphe G de la figure 2 et définissons un autre graphe $G' = (V', E')$ isomorphe à G et tel que chaque sommet $u \in V$ est renommé en u' dans G'. Nous notons x_u la variable associée à un sommet $u \in V$. La contrainte d'arc entre x_1 et x_4 contient tous les couples de sommets de G' qui ne sont pas reliés par un arc, *i.e.*,

$$\begin{aligned} C_{edge}(x_1, x_4) = \{ \quad & (1', 3'), (1', 4'), (1', 5'), (2', 4'), (2', 5'), (2', 6'), \\ & (3', 5'), (3', 6'), (3', 1'), (4', 6'), (4', 1'), (4', 2'), \\ & (5', 1'), (5', 2'), (5', 3'), (6', 2'), (6', 3'), (6', 4')\} \end{aligned}$$

alors que la contrainte de distance entre x_1 et x_4 contient seulement les couples de sommets de G' qui sont à une distance de 3 l'un de l'autre car la distance entre les sommets 1 et 4 est égale à 3, *i.e.*,

$$C_{distance}(x_1, x_4) = \{(1', 4'), (2', 5'), (3', 6'), (4', 1'), (5', 2'), (6', 3')\}$$

La contrainte de distance entre x_1 et x_4 étant plus stricte que la contrainte d'arc correspondante, elle permet une meilleur propagation des réductions de domaines. Par exemple, si x_1 est assignée à $1'$, une propagation par forward-checking de le contrainte $C_{distance}(x_1, x_4)$ réduit le domaine de x_4 au singleton $\{4'\}$ alors que la propagation par forward-checking de $C_{edge}(x_1, x_4)$ ne réduit le domaine de x_4 qu'à $\{3', 4', 5'\}$, l'ensemble des sommets qui ne sont pas connectés à $1'$.

De même, lorsque la valeur $1'$ est supprimée du domaine de x_1, une propagation par consistance d'arc de la contrainte $C_{distance}(x_1, x_4)$ permet d'enlever la valeur $4'$ au domaine de x_4 alors que la même propagation de la contrainte $C_{edge}(x_1, x_4)$ ne permet d'enlever aucune valeur.

Une contrainte de distance peut être vue comme un invariant d'un couple de sommets : c'est une étiquette $l(u, v)$ affectée à un couple de sommets (u, v) telle que, s'il existe une fonction d'isomorphisme reliant les sommets (u, v) aux sommets (u', v'), alors $l(u, v) = l(u', v')$. Il est possible d'utiliser des invariants plus performants (***i.e.***, définissant des contraintes plus strictes) que celui basé sur la distance entre deux sommets. Par exemple, il est possible de définir l'étiquette $l'(u, v)$ d'un couple (u, v) de sommets du graphe G comme un ensemble de triplets tels que $(d_u, d_v, n) \in l'(u, v)$ s'il existe n sommets dans G à une distance d_u du sommet u et à une distance d_v du sommet v. Des invariants plus sophistiqués permettent un plus fort filtrage de l'espace de recherche mais peuvent être aussi plus couteux à calculer et à comparer. L'invariant de couples de sommets qui offre le meilleur compromis entre l'efficacité et le temps de calcul devra être déterminé expérimentalement.

5. Extensions aux graphes orientés et au problème de l'isomorphisme de sous-graphe

5.1. *Graphes orientés.*

Les définitions et les théorèmes introduits en section 3 restent valables dans le cas des graphes orientés où les chemins doivent respecter le sens des arcs. Cependant, dans ce cas, il peut exister beaucoup de couples de sommets qui ne sont pas connectés par de tels chemins. Les séquences décrivant les sommets peuvent alors être très courtes, et, dans ce cas, le filtrage par label ne réduira pas beaucoup l'espace de recherche.

Une deuxième façon d'étendre notre travail aux graphes orientés consiste à générer le graphe non-orienté correspondant (en ignorant le sens des arcs) et à calculer les séquences et les labels sur ce graphe non-orienté. Des contraintes peuvent alors être ajoutées pour exprimer le sens des arcs.

Enfin, une dernière façon d'étendre notre travail aux graphes orientés est de considérer en parallèle différentes distances, basées chacunes sur différents chemins, ***e.g.***, des chemins orientés, qui respectent le sens des arcs, des chemins non-orientés, qui ignorent le sens des arcs... Chacune de ces distances peut alors être utilisée pour calculer un label.

Evidemment, si la troisième possibilité permet un filtrage plus important des domaines, c'est aussi la plus coûteuse à réaliser. Ces trois possibilités devront être comparées expérimentalement.

5.2. *Problèmes d'isomorphisme de sous-graphe*

Un graphe $G = (V, E)$ est un *sous-graphe* d'un autre graphe $G' = (V', E')$ (noté $G \subseteq G'$) si $V \subseteq V'$ et $E = E' \cap (V \times V)$. Un graphe $G = (V, E)$ est un sous-graphe isomorphe d'un autre graphe $G' = (V', E')$ s'il existe un sous-graphe $G'' \subseteq G'$ isomorphe à G. Le problème de l'isomorphisme de sous-graphe (SGIP) consiste à vérifier qu'un graphe $G = (V, E)$ est un sous-graphe isomorphe à un autre graphe $G' = (V', E')$.

Si la complexité théorique du GIP n'est pas encore parfaitement déterminée, SGIP est connu comme étant un problème NP-complet [GAR 79]. De fait, SGIP est un problème plus difficile et nombreuses instances ne peuvent être résolues en un temps raisonnable.

La modélisation des SGIPs en CSPs est très similaire à celle des GIPs. Comme pour les GIPs, il est possible de tirer partie de la sémantique globale du problème pour définir des algorithmes de filtrage puissants. Cependant, contrairement aux fonctions d'isomorphisme de graphes (théorème 1), une fonction d'isomorphisme de sous-graphe $f : V \rightarrow V'$ ne préserve pas les distances entre les sommets des graphes : pour chaque chemin $< v_1, ..., v_n >$ de G, il existe un chemin $< f(v_1), ..., f(v_n) >$ dans G' mais l'opposé n'est pas toujours vrai (f n'est pas une bijection et il peut donc exister des sommets de V' qui ne sont reliés à aucun sommet de V). Par conséquent, la distance $\delta_G(u, v)$ entre deux sommets u et v de G peut être plus grande que la distance $\delta_{G'}(f(u), f(v))$ entre $f(u)$ et $f(v)$.

Les techniques de filtrage permettant de résoudre efficacement un SGIP peuvent être basées sur la propriété suivante : étant donnés deux graphes non-orientés $G = (V, E)$ et $G' = (V', E')$, si f est une fonction d'isomorphisme de sous-graphe entre G et un sous-graphe de G', alors :

$$\forall (u, v) \in V \times V,\ \delta_G(u, v) \geq \delta_{G'}(f(u), f(v))$$

Cette propriété peut être utilisée pour définir une consistence partielle et l'algorithme de filtrage qui lui est associé. L'idée consiste à vérifier que pour tous les sommets u de G, le domaine de la variable x_u associée à u ne contient que les sommets u' tels que, pour toutes les distances $k \in 1..|V| - 1$, le nombre de sommets $v \in G$ pour lesquels $\delta_G(u, v) \leq k$ est plus petit ou égal aux nombre de sommets $v' \in G'$ pour lesquels $\delta_{G'}(u', v') \leq k$.

Cette proporiété peut aussi être utilisée pour définir des contraintes et propager les réductions de domaines pendant l'exploration d'un arbre de recherche.

6. Conclusion

Nous introduisons dans cet article une nouvelle contrainte globale pour les problèmes d'isomorphisme de graphes. Pour une utilisation efficace de cette contrainte globale, nous définissons une consistance partielle, la consistance de labels, et l'algorithme de filtrage associé, utilisable pour réduire le domaine des variables avant la résolution du CSP. Cette consistance de labels est basée sur le calcul, pour chaque sommet u, d'un label qui caractérise les relations en termes de plus court chemin entre u et les autres sommets du graphe. Dans de nombreux cas, la consistance de labels permet à un solveur de contraintes de détecter une inconsistance ou de réduire les domaines des variables à des singletons rendant alors la consistance globale du problème facile à vérifier.

Pour les cas où la consistance de labels ne permet pas de résoudre à elle seule le problème de l'isomorphisme de graphes, nous définissons un ensemble de contraintes de distance, sémantiquement équivalent à la contrainte globale gip, et permettant de propager les réductions de domaines. Nous montrons que ces contraintes de distances sont plus "strictes" que les contraintes d'arcs traditionnellement utilisées. Ces contraintes de distance permettent alors de réduire plus fortement les domaines des variables lors d'une propagation de contraintes. Les contraintes de distance peuvent être combinées avec une contrainte globale *allDiff* pour propager encore mieux les réductions.

Le filtrage par labels et la génération de l'ensemble des contraintes de distance peuvent être réalisés en $\mathcal{O}(np + n^2 log(n))$ opérations pour les graphes composés de n sommets et p arcs ($n - 1 \leq p \leq n^2$). En comparaison, la consistance d'arcs avec AC2001 sur un CSP décrivant le problème de l'isomorphisme avec des contraintes d'arcs nécessite $\mathcal{O}(ed^2)$ opérations [BES 93] où e est le nombre de contraintes, *i.e.*, $e = n(n-1)/2$, et d est la taille du plus grand domaine, *i.e.*, $d = n$. Etablir la consistance de labels sur notre contrainte globale est donc d'un ordre moins coûteux que la consistance d'arcs sur les contraintes binaires traditionnellement utilisées. Notons cependant que ces consistances ne sont pas comparables : pour certains graphes, tel que le graphe de la figure 1, la consistance de labels permet de résoudre le problème alors que AC sur les contraintes d'arcs ne réduit aucun domaine. Inversement, sur le graphe de la figure 2, la consistance de labels ne réduit le domaine d'aucune variable alors que la consistance d'arcs permet une réduction des domaines dès qu'une variable est affectée à une valeur.

La suite de notre travail concernera l'intégration de notre algorithme de filtrage dans un solveur de contraintes (*e.g.*, CHOCO [LAB 00]) afin d'évaluer les performances de ce filtrage. Ce travail permettra ensuite d'effectuer des expérimentations pour déterminer quel invariant d'un couple de sommets offre un bon équilibre entre puissance de filtrage et temps de calcul. Nous aimerions enfin clarifier les relations qu'il peut exister entre différentes consistances partielles sur les contraintes $C_{distance}$ et C_{edge}. En particulier, lors de tous nos tests, nous avons pu constater qu'après l'affectation d'une valeur à une variable, une propagation par forward checking des

contraintes $C_{distance}$ réduisait au moins autant les domaines qu'une propagation par consistance d'arc des contraintes C_{edge}. Nous aimerions donc prouver cette assertion ou lui trouver un contre-exemple.

7. Bibliographie

[AGG 92] AGGOUN A., BELDICEANU N., « Extending CHIP in order to solve complex scheduling and placement problems », *Actes des Journées Francophones de Programmation et Logique, Lille, France*, 1992.

[AHO 74] AHO A., HOPCROFT J., ULLMAN J., *The design and analysis of computer algorithms*, Addison Wesley, 1974.

[BES 93] BESSIÈRE C., CORDIER M.-O., « Arc-Consistency and Arc-Consistency Again », *Proceedings of the 11th National Conference on Artificial Intelligence*, Menlo Park, CA, USA, juillet 1993, AAAI Press, p. 108–113.

[BES 01] BESSIÈRE C., RÉGIN J., « Refi ning the Basic Constraint Propagation Algorithm », NEBEL B., Ed., *Proceedings of the seventeenth International Conference on Artificial Intelligence (IJCAI-01)*, San Francisco, CA, août 4–10 2001, Morgan Kaufmann Publishers, Inc., p. 309–315.

[BES 03] BESSIÈRE C., HENTENRYCK P. V., « To be or not to be... a global constraint », *CP'03, Kinsale, Ireland*, vol. LNCS N°2833, 2003, p. 789-794.

[CHA 03] CHAMPIN P.-A., SOLNON C., « Measuring the similarity of labeled graphs », *5th International Conference on Case-Based Reasoning (ICCBR 2003)*, vol. LNAI N°2689, 2003, p. 80-95.

[COR 90] CORMEN T., LEISERSON C., RIVEST R., *Introduction to Algorithms*, MIT Press, 1990.

[COR 01] CORDELLA L., FOGGIA P., SANSONE C., VENTO M., « An Improved Algorithm for Matching Large Graphs », *3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, Cuen, 2001, p. 149-159.

[FOG 01] FOGGIA P., SANSONE C., VENTO M., « A Performance Comparison of Five Algorithm for Graph Isomorphism », *3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, Cuen, 2001, p. 188-199.

[FOR 96] FORTIN S., « The Graph Isomorphism Problem », rapport, 1996, Dept of Computing Science, Univ. Alberta, Edmonton, Alberta, Canada.

[GAR 79] GAREY M., JOHNSON D., *Computers and Intractability : A Guide to The Theory of NP-Completness*, W.H. Freeman, San Francisco, 1979.

[HOP 74] HOPCROFT J., WONG J., « Linear time algorithm for isomorphism of planar graphs », *Proc. 6^{th} Annual ACM Symp. theory of Computing*, vol. ACM Press, 1974, p. 172-184.

[ILO 00] ILOG S., *ILOG Solver 5.0 User's Manual and Reference Manual*, ILOG,S.A., 2000.

[LAB 00] LABURTHE F., THE OCRE PROJECT TEAM, « CHOCO : implementing a CP kernel », *Proc. of the CP'2000 workshop on techniques for implementing constraint programming systems, Singapore*, 2000.

[LUK 82] LUKS E., « Isomorphism of Graphs of bounded valence can be tested in polynomial time », *Journal of Computer System Science*, vol. 25, 1982, p. 42-65.

[MCG 79] MCGREGOR J., « Relational Consistency Algorithms and Their Applications in Finding Subgraph and Graph Isomorphisms », *Information Science*, vol. 19, 1979, p. 229-250.

[MCK 81] MCKAY B., « Practical Graph Isomorphism », *Congressus Numerantium*, vol. 30, 1981, p. 45-87.

[MOH 86] MOHR R., HENDERSON T., « Arc and path consistency revisited », *Artificial Intelligence*, vol. 28, 1986, p. 65-74.

[RÉG 95] RÉGIN J., « Développement d'Outils Algorithmiques pour l'Intelligence Artificielle. Application à la Chimie Organique », PhD thesis, Université Montpellier II, 1995.

[TSA 93] TSANG E., *Foundations of Constraint Satisfaction*, Academic Press, 1993.

[ULL 76] ULLMAN J., « An algorithm for subgraph isomorphism », *Journal of the Association of Computing Machinery*, vol. 23, n° 1, 1976, p. 31-42.