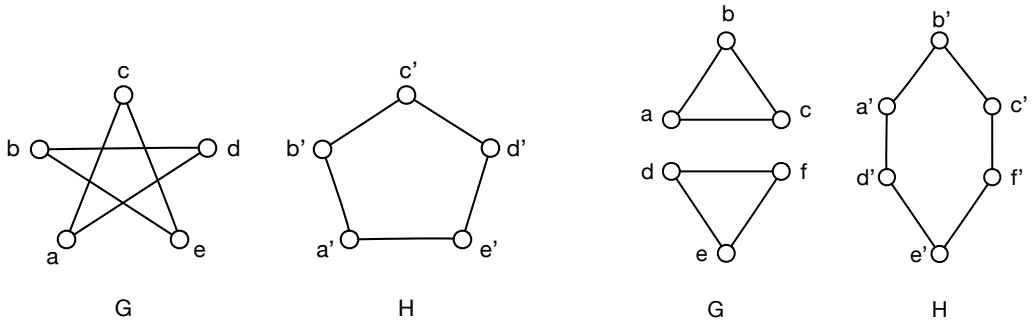
Programmation par Contraintes - Rapport

Vasquez Alessandro

15 février 2018

Table des matières



(a) Deux graphes isomorphes

(b) Graphes non-isomorphes

FIGURE 1 – Isomorphisme, examples

1 Introduction

Dans ce rapport nous allons proposer une modélisation pour trois problèmes d'isomorphisme sur graphe. Nous verrons d'abord les définitions préliminaires dans la section 1. Ensuite nous allons traiter les problèmes d'isomorphisme de graphe, sous-graphe et plus grandes sous structures communes dans la section 2. Enfin dans la section 3 nous citerons quelques articles qui proposent des modélisations différents.

Définition 1.1 (Graphe sans Boucle)**.** Un graphe G est un couple (V, E) où V est l'ensemble de sommets et E est l'ensemble d'arêtes. Une arête est une paire (u, v) de sommets $u, v \in V$ tq $u \neq v$.

Définition 1.2 (Isomorphisme de graphe)**.** Soient G, H deux graphes $G = (V, E)$ et $H = (V', E')$. Une fonction f entre F et G est un isomorphisme ssi f est une bijection entre les sommets de G et de H t.q. $(u, v) \in E$ ssi $(f(u), f(v)) \in H$.

Nous remarquons qu'il y a plusieurs propriétés qui doivent être satisfaites au même temps :

(i) $|V| = |V'|$,

(ii) $|E| = |E'|$,

(iii) G, H ont le même nombre de composantes connectées.

Au début on peut penser que si tous les sommets des deux graphes ont le même dégrée, le graphe sont forcement isomorphes. La figure 1b montre un exemple de graphes dont tous les sommets ont le même degré mais qui ne sont pas isomorphes.

Définition 1.3 (Isomorphisme de sous-graphe)**.** Soient G, H deux graphes $G = (V, E)$ et $H = (V', E')$. Nous nous demandons s'il existe un sous-graphe $G' = (V_s, E_s), V_s \subseteq V$ et $E_s \subseteq E \cap (V \times V)$ tq $G' =_{iso} H$.

Définition 1.4 (Plus grandes sous structures communes)**.** Entre les sous structures communes d'un graphe il y a différents types de sous-graphes sur lesquels nous pouvons tomber. Soit $G = (V, E)$ un graphe et $G' = (V', E')$ un sous-graphe de G. Alors :

1. G' est sous-graphe couvrant (ou *graphe partiel*) de G si $V = V'$,
2. G' est *sous-graphe partiel* de G si $V' \subseteq V \wedge \forall v, u \in V' : (u, v) \in E'$ ssi $(u, v) \in E$.

La sous-structure commune de deux graphe G, H peut être définie comme un graphe g tq il existe un isomorphisme de sous-graphe de G à g et de H à g.

Remarque 1.1. Nous supposons que les isomorphismes sont définies sur graphes non orientés.

2 Développement

2.1 Isomorphisme de graphe

2.1.1 Modélisation

Soient $G = (V, E),\ G' = (V', E')$ deux graphes. La définition d'isomorphisme nous permet d'obtenir directement une modélisation. Soit $CSP = (X, D, C)$ le problème de PPC que nous voulons définir. Nous définissons l'ensemble de variables de la façon suivante :

$$X = \{x_v \mid v \in V\},\ \forall v \in V.$$

Puisque nous cherchons une fonction $f : V \to V'$ t.q. $(u, v) \in E$ ssi $(f(u), f(v)) \in E'$, nous savons déjà que deux sommets $u \in V, u' \in V'$ peuvent être liés par f uniquement si $deg(u) = deg(u')$. Donc, nous allons considérer pour chaque $u \in V$ l'ensemble des sommets $\{v_1, v_2..v_k\}$ tq $v_1, v_2..v_k \in V'$ et $deg(v_1) = deg(v_2) = .. = deg(v_k) = deg(u),$ pour un certain $k : 0 \leq k \leq |V'|$. Nous remarquons tout de suite que si $k = 0$ pour un sommet $u \in V$, il n'y aura pas une solution à notre problème. Pour chaque variable $x \in X$, le domaine est ainsi définis comme l'ensemble :

$$D_{x_v} = \{u \mid deg(u) = deg(v) \wedge u \in V'\}.$$

Pour définir les contraintes, nous suivrons la définition d'isomorphisme, donc $\forall u, v \in V$ tq $(u, v) \in E$:

$$c(x_u, x_v) \triangleq \text{ " pour chaque sommets } u' \in D_{x_u} : \exists v' \in D_{x_v} \text{ tq } (u', v') \in E' \wedge u' \neq v' \text{ ".}$$

Remarque 2.1. Nous allons appeler ce type de contrainte *contrainte d'adjacence des sommets.* Ici les contraintes sont binaires et définies localement. À partir de l'ensemble de contraintes locales il est possible de définir un seul contrainte de cardinalité globale C [1]. Puisque nous cherchons une fonction isomorphe, nous savons que un sommet de G doit être associé à chaque sommet de G'. Donc l'idée est d'avoir une contrainte globale dans lequel chaque valeur des domaines peut être assumée une et une seul fois par une variable. Nous pouvons par exemple utiliser le système d'étiquetage proposé dans [3]. Par ailleurs, il est utile de préciser que une contrainte globale *alldiff* peut aussi être utilisé une fois pour toute plutôt que indiquer $u' \neq v'$ pour chaque contrainte $c(x_u, x_v)$.

2.1.2 Complexité du modèle

La complexité du problème $CSP = (X, D, C)$ est donné par le nombre de variables, contraintes et par la taille des domaines. Pour ce problème nous avons que par construction $|X| = |V|$, c'est à dire que le nombre de variables est égal au nombre de sommets de G. Chaque domaine est défini comme un sous-ensemble de l'ensemble de sommets de G', donc $|D_x| \leq |V'|, \forall x \in X$. Puisque les contraintes sont définies comme l'ensemble $C \subseteq (V \times V')$, le nombre de contraintes est toujours $|C| \leq |V| \cdot |V'|$.

2.1.3 Résolution

En considérant le contrainte globale C obtenu à partir de l'ensemble de contraintes globales, nous allons introduire maintenant le réseau de valeurs [1] associé à C. Puisque $\forall x \in X : D_x \subseteq V'$ nous savons que

$$U(C) = \bigcup_{x \in X(C)} D_x = V'.$$

Nous savons aussi que $|X| = |V|$, parce que une et une seule variable est associé à chaque sommet de G. Nous définissions le graphe bipartite $GV(C) = (X(C), U(C), A)$ où $A \subseteq X(C) \times U(C)$ est l'ensemble d'arêtes défini de la manière suivante :

$$A = \{(u, x_v) \in A \mid u \in U(C), x_v \in X(C) \wedge deg(v) = deg(u)\},$$

c'est à dire que dans le graphe $GV(C)$ il y aura un arc entre une variable x_v et un sommet $u \in G'$ seulement si v et u ont le même degré. Pour construire le réseau de valeurs $N(C)$, nous devons introduire des nouveaux éléments :

1. un sommet s et un sommet t,
2. un arc (s, u) pour chaque $u \in U(C)$,
3. un arc (x_v, t) pour chaque $x_v \in X(C)$,
4. une direction pour chaque arc $a = (u, v) \in A$: nous supposons que a est sortant de u et entrant dans v.

Le problème de trouver un isomorphisme peut être transformé en un problème de flot sur $N(C)$ en définissant la fonction de capacité $c : A \to [\mathbb{R}, \mathbb{R}]$, où $[\mathbb{R}, \mathbb{R}]$ est un intervalle fermé dans lequel le flot qui passe dans l'arc a peut prendre une valeur, comprise la borne inférieure et supérieure. Dans notre cas, la fonction de capacité c est définie par

$$\forall (u, v) \in A : c((u, v)) = \begin{cases} [1, 1] & \text{if } u = s \vee v = t, \\ [0, 1] & \text{sinon.} \end{cases}$$

L'idée est de trouver un flot tq l'arc (t, q) est saturé. L'algorithme pour le trouver est décrit in [1]. Lorsque nous injections un flot de s à $u \in U(C)$ et puis de u à $x_v \in X(C)$ nous allons couper les arcs qui sont incompatible avec la contrainte globale où $u = x_v$ est fixé : par exemple, si un sommet $v' \in V$ est adjacent à v, mais $u' \in V'$ n'est pas adjacent à u, l'arc $(x_{v'}, u')$ sera coupé. Ici un algorithme d'arc-consistance peut être utilisé. Ensuite nous allons itérer cette procédure jusqu'à nous trouvions une solution. Si pendant le processus un sommet perd tous ses arcs (i.e. son degré devient 0), nous recommençons la procédure de zéro mais en choisissant au début un arc différent de (u, x_v). Un exemple de cette procédure de résolution pour le cas de figure 1a est présenté dans la figure 2 . Nous insistons sur le fait que à chaque pas, dès que un arc est choisi, il y a des arcs qui sont coupés à cause au moins d'une des contraintes suivantes :

1. les contraintes *alldiff,*
2. les contraintes d'adjacence des sommets.

Par exemple, dès que un flot est injecté sur le chemin $s \to a' \to x_a \to t \to s$, nous coupons tous les autres arcs qui partent de a' et ceux qui arrivent en x_a pour la contrainte *alldiff.* Ensuite pour les contraintes d'adjacence des sommets nous considérons que

1. $neighbors(a) = \{c, d\}$,
2. $neighbors(a') = \{b', e'\}$.

Donc nous allons couper les arcs

$$a = (b', x_v) \text{ tq } v \notin \{c, d\},$$
$$a = (e', x_v) \text{ tq } v \notin \{c, d\},$$
$$a = (u, c) \text{ tq } u \notin \{b', e'\},$$
$$a = (u, d) \text{ tq } u \notin \{b', e'\}.$$

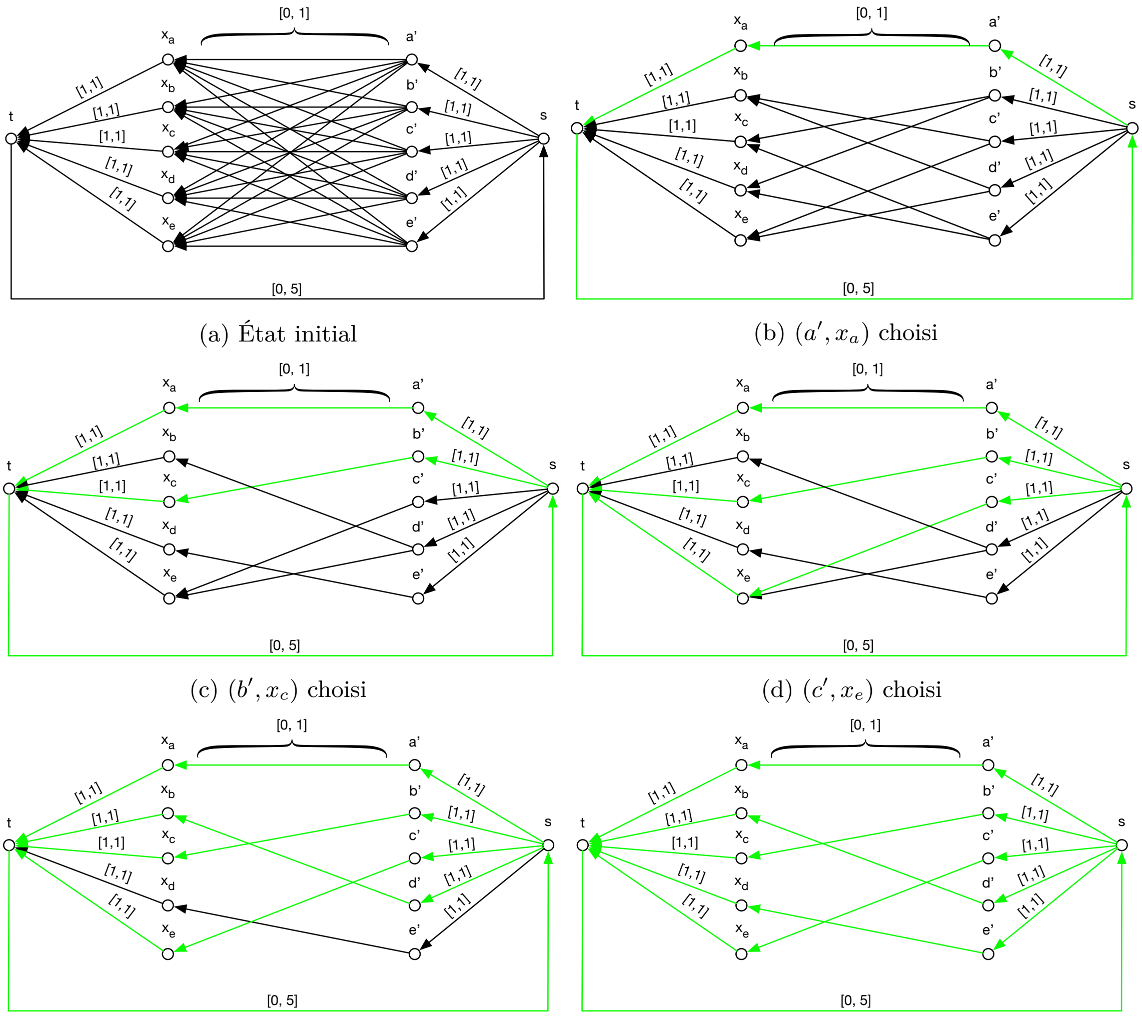
Ensuite nous allons itérer la procédure sur un autre sommets jusqu'à le flot qui entre dans s est égal à $|U(C)|$, i.e. tous les sommets de G' sont couverts. Finalement, la procédure termine dès que le flot qui entre dans le sommet s est égal au nombre de sommets de G'. S'il n'y a pas de solutions, la procédure termine quand toutes les possibles combinaisons ont été essayées.

2.1.4 Résolution, complexité

Sans compter l'initialisation des structures de données, l'algorithme de résolution va essayer d'injecter le flot pour toutes les permutations admissibles des couples $(u, x_v), u \in U(C), x \in X(C)$, dans le pire des cas. Chaque fois que le flot est injecté, nous allons chercher des arcs à couper en utilisant par exemple un algorithme d'arc-consistance. Supposons que AC est la complexité de l'algorithme d'arc-consistance. Alors la complexité sera

$$O(AC \cdot (|X(C)|^2 \cdot |U(C)|)).$$

En fonction de quel algorithme d'arc-consistance et quel structure de données ont été choisis, la complexité de l'algorithme peut arriver à être exponentielle, au pire. Par contre, nous savons que l'algorithme de résolution va s'arrêter dés qu'une solution a été trouvée. Il y a beaucoup d'exemples (comme dans la figure 2) dans lesquels l'algorithme va trouver une solution en temps polynomial par rapport à l'ensemble de sommets.



(a) État initial

(b) (a', x_a) choisi

(c) (b', x_c) choisi

(d) (c', x_e) choisi

(e) (d', x_b) choisi

(f) (e', x_d) choisi

FIGURE 2 – Résolution pour l'exemple dans la figure 1a.

2.2 Isomorphisme de sous-graphe

2.2.1 Modélisation

Comme pour le premier problème, nous pouvons définir un $CSP = (X, D, C)$ tout de suite en utilisant la définition du problème d'isomorphisme de sous-graphe que nous avons introduit dans la section 1. Soient $G = (V, E)$, $G' = (V', E')$ deux graphes. Les variables et les domaines sont définies toujours de la manière suivante :

$$X = \{x_v \mid v \in V\}, \ \forall v \in V,$$

$$D_{x_v} = \{u \mid deg(u) \leq deg(v) \wedge u \in V'\}.$$

Le degré de $v \in V$ doit être supérieur où égal au degré de $u \in V'$ parce que nous cherchons un sous-graphe de G qui soit isomorphe à G', donc si nous choisissons une couple (x_v, u) où

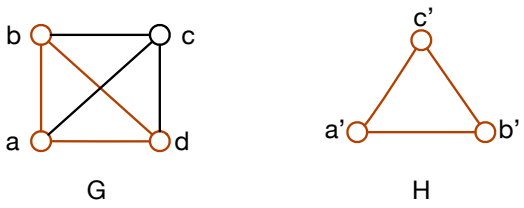


FIGURE 3 – Exemple d'isomorphisme de sous-graphe

$deg(u) < deg(v)$, nous pouvons toujours supprimer $deg(v) - deg(u)$ arcs adjacent à v pour obtenir un sous-graphe tq $deg(u) = deg(v)$. Comme avant, les contraintes peuvent être définies $\forall u, v \in V$ tq $(u, v) \in E$: :

$$C(x_u, x_v) \triangleq \text{ " pour chaque sommet } u' \in D_{x_u} : \exists v' \in D_{x_v} \text{ tq } (u', v') \in E' \wedge u' \neq v' \text{ ".}$$

Les remarques 2.1 s'appliquent toujours.

2.2.2 Complexité du modèle

Puisque nous avons défini la base du problème en utilisant la modélisation pour l'isomorphisme de graphe, les remarques sur la complexité dans le paragraphe 2.1.2 s'appliquent toujours.

2.2.3 Résolution

Afin de déterminer une solution nous pouvons reprendre la construction du réseau de valeurs du premier problème en appliquant quelque petite modification. Supposons que le réseau $N(C) = (X(C), U(C), A)$ est défini comme avant. Maintenant nous allons définir une nouvelle fonction de capacité des arcs :

$$\forall (u, v) \in A : c((u, v)) = \begin{cases} [1, 1] & \text{if } u = s, \\ [0, 1] & \text{sinon.} \end{cases}$$

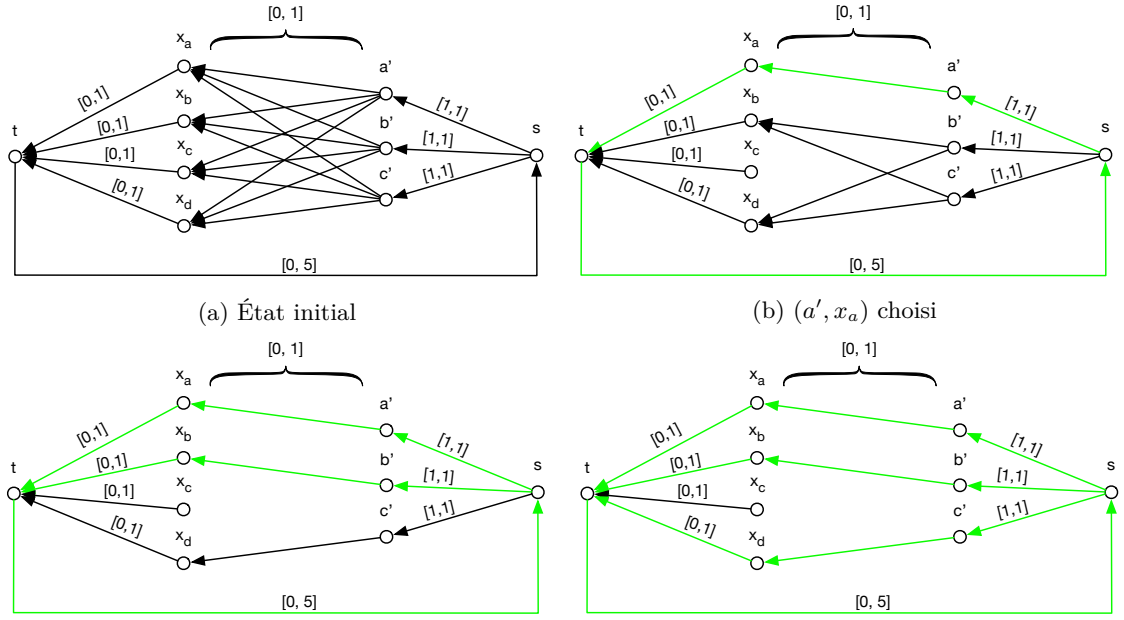
Donc maintenant les arcs entrants de t ont capacité : $[0, 1]$. Dans le sens de la modélisation du problème, ça signifie que le but n'est plus d'essayer de couvrir entièrement les sommets des deux graphes, mais seulement ceux du deuxième graphe, c'est à dire G'. De cette façon, dès que nous réussissons à couvrir tous les sommets $u \in U(C)$ avec le flot, nous allons trouver une solution. Comme avant, un exemple de résolution pour le cas présenté dans la figure 3 est indiqué à la figure 4 .

2.2.4 Résolution, complexité

Comme avant, la complexité va toujours être

$$O(AC \cdot (|X(C)|^2 \cdot |U(C)|)),$$

avec une complexité potentiellement exponentielle, toujours en fonction de quel algorithme d'arc-consistance et quel structure de données ont été choisis. Par contre, dans ce cas, puisque nous allons nous arrêter dès que nous trouvons un sous-graphe de G pour lequel il y a une solution, l'algorithme de résolution peut être plus vite de l'algorithme pour l'isomorphisme simple. De toute façon l'idée c'est qu'il y a un bon nombre d'exemples pour lesquels cet algorithme arrive à trouver une solution en temps polynomial.



(a) État initial

(b) (a', x_a) choisi

(c) (b', x_b) choisi

(d) (c', x_d) choisi

FIGURE 4 – Résolution pour l'exemple dans la figure 3.

2.3 Plus grandes sous structures communes

2.3.1 Introduction

Grâce à la modélisation utilisée pour les problèmes de isomorphisme de graphe et sous-graphe, nous allons tout de suite arriver à une solution. Nous assumons que $CSP = (X, D, C)$ est le problème de programmation par contrainte défini comme dans 2.1.1 et que $N(C)$ est le réseau de valeurs défini comme dans 2.2.3.

2.3.2 Résolution

Une solution à ce problème peut être trouvée en itérant l'algorithme de résolution du réseau de valeurs et en essayant tous les cas et en gardant toutes les solutions admissibles dans un ensemble $S = \{S_0, S_1..\}$, où chaque S_i est une solution définie par l'ensemble de couple $\{(u_0, x_{v_0}), (u_1, x_{v_1})..\}$. Enfin nous allons simplement choisir la solution qui correspond à l'isomorphisme de sous-graphe dont l'ensemble de sommets est le plus grand, c'est à dire :

$$\arg\max_{|S_i|}(S).$$

2.3.3 Résolution, complexité

Avec cette tactique, l'algorithme de résolution va explorer toujours tous les possibles solutions, c'est sûr que la complexité sera exponentielle par rapport au nombre de sommets, peu importe quel algorithme d'arc-consistance ou structure de données sont utilisés.

3 Conclusion

Nous avons présenté dans la section 2 trois simples modélisations et un algorithme de résolution qui utilise le réseau de valeurs pour trouver une solution. Malheureusement, la complexité de cet algorithme est exponentielle (dans le pire des cas) et, pour le problème du plus grande sous structure commune, elle est même exponentielle pour tout le cas. Tout à fait, ces problèmes sont encore ouverts, même s'il y a déjà nombreux articles qui proposent des nouveaux algorithmes et modélisations afin de réduire la complexité, par exemple [2, 4, 5]. Le problème c'est que il y a déjà beaucoup d'algorithmes qui arrivent à avoir une complexité polynomial pour un certain nombre d'exemples, mais il sont toujours exponentiels dans le pire des cas. Donc, le but de la recherche sur ce sujet est de trouver un algorithme ou une modélisation tells que la complexité soit réduite pour tous les cas (ou pour la plus part, au moins), c'est à dire garantir une complexité plus basse même dans le pire des cas.

Références

[1] Jean-Charles Régin. Generalized arc consistency for global cardinality constraint. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, August 4-8, 1996, Volume 1.*, pages 209–215, 1996.

[2] Christine Solnon. Alldifferent-based filtering for subgraph isomorphism. *Artif. Intell.*, 174(12-13) :850–864, 2010.

[3] Sébastien Sorlin and Christine Solnon. Une contrainte globale pour le problème de l'isomorphisme de graphes. In *Programmation en logique avec contraintes, JFPLC 2004, 21, 22 et 23 Juin 2004, Angers, France*, 2004.

[4] Sébastien Sorlin and Christine Solnon. A parametric filtering algorithm for the graph isomorphism problem. *Constraints*, 13(4) :518–537, 2008.

[5] Reza Takapoui and Stephen P. Boyd. Linear programming heuristics for the graph isomorphism problem. *CoRR*, abs/1611.00711, 2016.