Algorithme polynomial pour les graphes isomorphes, tous les cas

Mohamed mimouni
20 rue kadissia 60000 Oujda Maroc
mimouni.mohamed@gmail.com

1er août 2018

Résumé

Dans ce papier, je propose un algorithme capable de résoudre le problème de graphes isomorphes en temps polynomial. En premier lieu, je défini un pseudo-arbre qui nous permet de définir pour chaque sommet une étiquette ou un label. En second lieu, j'applique le pseudo-arbre pour le premier graphe puis je calcul les étiquette de chaque sommet de premier graphe, puis je fais la même chose pour le deuxième graphe. En troisième lieu je cherche pour chaque sommet de graphe1 les sommets de graphe2 qui ont la même étiquette, si ou moins un sommet de premier graphe son étiquette n'est pas dans les étiquette de deuxième graphe on déduit que les deux pseudo-arbres ne sont pas isomorphes. Dans d'autres cas je génère des solutions et je les vérifier en temps polynomiale... Cet algorithme permet donc pour les graphes isomorphes de calculer l'image de chaque sommet en temps polynomiale.

MOTS-CLES : graphes isomorphes, isomorphisme, automorphisme, hypergraphes isomorphes

Table des matières

1 Définitions

1.1 Un graphe

Un graphe est un ensemble de points nommés nœuds (parfois sommets ou cellules) reliés par des traits (segments) ou flèches nommées arêtes (ou liens ou arcs).

1.2 Un hypergraphe

Un hypergraphe est un graphe dont les arêtes relions un ou deux ou plusieurs sommets.

1.3 Un isomorphisme

Un isomorphisme **f** entre les graphes **G** et **H** est une bijection entre les sommets de **G** et ceux de **H**, telle qu'une paire de sommets **u**, **v** de **G** est une arête de **G** si et seulement si (**u**), (**v**) est une arête de **H**.

2 Construire un pseudo-arbre

Soit **G** un graphe simple et **s** un sommet. Le pseudo-arbre entête **s** se fait de la manière suivante :

1. Le sommet **s** dans **niveau 0**.
2. Les adjacents de **s** dans **niveau 1**.
3. Les adjacents de chaque sommet de **niveau 1** sans répétition, qui ne sont pas déjà dans les autres niveaux, dans le **niveau 2**.
4. Répéter l'étape 3 pour construire les autres niveaux.
5. Fin quand il n'y a pas des nouveaux adjacents.

Si dans le cas (cas de graphe non connecte) où il reste des sommets de graphe G, qui ne sont pas dans le pseudo-arbres on doit génère d'autres pseudo-arbres.

3 Étiquette

Pour chaque sommet de **G** une étiquette se forme de **quatre** éléments :

1. **n** le niveau où il se trouve.
2. **h** le nombre des adjacents qui sont dans le niveau en haut.
3. **m** le nombre des adjacents qui sont dans le même niveau.
4. **b** le nombre des adjacents qui sont dans le niveau en bas.

4 Algorithme

Soit **G** un graphe $\{g_1, g_2, g_3, , , g_n\}$ et **H** un graphe $\{h_1, h_2, h_3, , , h_n\}$
Les deux graphes **G** et **H** sont isomorphe si pour la fonction isomorphisme $\mathbf{f}(g_i) = h_i$, on a l'étiquette de g_i égal l'étiquette de h_i.
$label(g_i) = label(h_i) = n - h - m - b$

4.1 Pour le graphe G

Construire un ou plisseurs pseudo-arbres de manière d'avoir une étiquette pour chaque sommet de graphe **G**.

4.2 Pour le graphe H

Construire des pseudo-arbres et calculer les étiquettes pour chaque sommet de **H**. Chercher dans le graphe **G** les sommets qui ont la même étiquette. Si au moins un sommet de **H** a une étiquette inégale des étiquettes des sommet de **H**, on modifier l'entête et refaire les calculs des étiquettes des sommets de **H**. Si chaque sommet de **H** a une étiquette égale a des étiquettes de graphe **G**, on fait une validation ou une vérification des solutions. Si la solution n'est pas validée on change les entêtes. Fin si une solution est validée ou vérifier.

5 Preuve mathématique

5.1 théorème principale

Soit $P(a)$ et $Q(a')$ deux pseudo-arbres isomorphes.
Si $f(a) = a'$ on a :
Pour chaque sommet **s** de **P(a)** : $label(s) = label(f(s))$

5.1.1 preuve

L'isomorphisme conserve la distance, donc :

1. $\mathbf{n} = d(a, s) = d(a', f(s))$ donc f(s) se trouve aussi dans niveau **n**.
2. pour **h**, il existe **h** sommets de **niveau n-1** adjacents a **s** tel que $d(s, t_h) = d(s', t'_h)$ et $d(a, t_h) = d(a', t'_h) = n - 1$, d'après la définition il y a aussi **h** sommets dans **Q(a')**.
3. pour **m**, il existe **m** sommets de **niveau n** adjacents a **s** donc $d(s, t_m) = d(s', t'_m)$ et $d(a, t_m) = d(a', t'_m) = n$, d'après la définition il y a aussi **m** sommets dans **Q(a')**.
4. pour **b**, il existe **b** sommets de **niveau n+1** adjacents a **s** donc $d(s, t_b) = d(s', t'_b)$ et $d(a, t_b) = d(a', t'_b) = n + 1$, d'après la définition il y a aussi **b** sommets dans **Q(a')**.

se qui démontre que $label(s) = label(f(s))$

5.2 L'algorithme

D'après la théorème principale on 2 cas :

Cas 1 $label(s) \neq label(s') \Longrightarrow f(s) \neq s'$.

Cas 2 $label(s) = label(s')$, il faut vérifier f(s)=s' (On ne peut pas affirmer que $f(s) = s'$).

5.2.1 La vérification

Donc après avoir trouver pour chaque sommet de **G** une ou plusieurs images de l'isomorphisme **f**, on doit faire une dernière étape : **la vérification des solutions**, cette vérification et **polynomiale**.

5.2.2 La complexité

Tous les étapes de l'algorithme sont polynomiale, donc L'algorithme proposé dans se papier et Polynomiale, donc **l'isomorphisme est dans P**.

6 Complexité d'algorithme

Chaque sommet doit être dans un pseudo-arbres, la complexité est donc $\mathbf{O}(n^2)$.
Pour le deuxième graphe la complexité peut atteindre $\mathbf{O}(n^3)$.
La calcule des étiquettes de chaque sommet est de $\mathbf{O}(n)$.
La validation de solution est de $\mathbf{O}(n^2)$.
Donc cet algorithme est polynomial.

7 Exemples

7.1 graphes connectes

Soit les deux graphe **G** et **H**, présenter par listes des adjacents suivants :

Graphe **G**	Graphe **H**
a :b,d,e,h	1 :4,6,7,8,9
b :a,c,e,g,i	2 :6,7,8,9
c :b,f,g,h,i	3 :5,6
d :a,e,f,i	4 :1,5,6,9
e :a,b,d,h,i	5 :3,4,7,9
f :c,d	6 :1,2,3,4,8
g :b,c,h	7 :1,2,5,9
h :a,c,e,g	8 :1,2,6
i :b,c,d,e	9 :1,2,4,5,7

7.1.1 Pseudo-arbre de tête a pour graphe G

Graphe **G**	niveaux
a	niveau 0
b,d,e,h	niveau 1
f,i,c,g	niveau 2

7.1.2 Label de chaque sommet de graphe G

sommet	adjacents	niveau	haut	même	bas	label
a	b,d,e,h	0	vide	a	b,d,e,h	0-0-0-4
b	a,c,e,g,i	1	a	b,d,e,h	f,i,c,g	1-1-1-3
d	a,e,f,i	1	a	b,d,e,h	f,i,c,g	1-1-1-2
e	a,b,d,h,i	1	a	b,d,e,h	f,i,c,g	1-1-3-1
h	a,c,e,g	1	a	b,d,e,h	f,i,c,g	1-1-1-2
f	c,d	2	b,d,e,h	f,i,c,g	vide	2-1-1-0
i	b,c,d,e	2	b,d,e,h	f,i,c,g	vide	2-3-1-0
c	b,f,g,h,i	2	b,d,e,h	f,i,c,g	vide	2-2-3-0
g	b,c,h	2	b,d,e,h	f,i,c,g	vide	2-2-1-0

explications Après avoir fini pour graphe **G**, on passe au graphe **H** : Label (a)=0-0-0-4 donc pour graphe **H**, on teste f(a)=2, f(a)=4, f(a)=5 et f(a)=7.

7.1.3 Pseudo-arbre de tête 2,4,5 et 7 pour graphe H

2	4	5	7	niveau 0	a
6,7,8,9	1,5,6,9	3,4,7,9	1,2,5,9	niveau 1	b,d,e,h
3,1,4,5	3,8,2,7	6,1,2	6,8,3,4	niveau 2	f,i,c,g
		8		niveau 3	

Nous avant pour sommet 5 quatre niveaux donc $f(a) \neq 5$.

7.1.4 Les labels

tête 2

sommet	adjacents	niveau	haut	même	bas	label	images
2	6,7,8,9	0	vide	2	6,7,8,9	0-0-0-4	a
6	1,2,3,4,8	1	2	6,7,8,9	3,1,4,5	1-1-1-2	d,h
7	1,2,5,9	1	2	6,7,8,9	3,1,4,5	1-1-1-2	d,h
8	1,2,6	1	2	6,7,8,9	3,1,4,5	1-1-1-1	vide

Pour sommet 8 le label (1-1-1-1) n'existe pas dans les labels de graphe **G**, donc $f(a) \neq 2$.

tête 4

sommet	adjacents	niveau	haut	même	bas	label	images
4	1,5,6,9	0	vide	4	1,5,6,9	0-0-0-4	a
1	4,6,7,8,9	1	4	1,5,6,9	3,8,2,7	1-1-2-2	vide

Pour sommet 1 le label (1-1-2-2) n'existe pas dans les labels de graphe G, donc $f(a) \neq 4$.

tête 7

sommet	adjacents	niveau	haut	même	bas	label	images
7	1,2,5,9	0	vide	7	1,2,5,9	0-0-0-4	a
1	4,6,7,8,9	1	7	1,2,5,9	6,8,3,4	1-1-1-3	b
2	6,7,8,9	1	7	1,2,5,9	6,8,3,4	1-1-1-2	d,h
5	3,4,7,9	1	7	1,2,5,9	6,8,3,4	1-1-1-2	d,h
9	1,2,4,5,7	1	7	1,2,5,9	6,8,3,4	1-1-3-1	e
6	1,2,3,4,8	2	1,2,5,9	6,8,3,4	vide	2-2-3-0	c
8	1,2,6	2	1,2,5,9	6,8,3,4	vide	2-2-1-0	g
3	5,6	2	1,2,5,9	6,8,3,4	vide	2-1-1-0	f
4	1,5,6,9	2	1,2,5,9	6,8,3,4	vide	2-3-1-0	i

7.1.5 Vérifier les solutions

sommet	adjacents	images	images adjacents	sommet	adjacents	etat
7	1,2,5,9	a	(b),(d,h),(d,h),(e)	a	b,d,e,h	ok
1	4,6,7,8,9	b	(i),(c),(a),(g),(e)	b	a,c,e,g,i	ok
2	6,7,8,9	d	(c),(a),(g),(e)	d	a,e,f,i	non
2	6,7,8,9	h	(c),(a),(g),(e)	h	a,c,e,g	ok
5	3,4,7,9	d	(f),(i),(a),(e)	d	a,e,f,i	ok
5	3,4,7,9	h	(f),(i),(a),(e)	h	a,c,e,g	non
9	1,2,4,5,7	e	(b),(d,h),(i),(d,h),(a)	e	a,b,d,h,i	ok
6	1,2,3,4,8	c	(b),(d,h),(f),(i),(g)	c	b,f,g,h,i	ok
8	1,2,6	g	(b),(d,h),(c)	g	b,c,h	ok
3	5,6	f	(d,h),(c)	f	c,d	ok
4	1,5,6,9	i	(b), (d,h),(c),(e)	i	b,c,d,e	ok

explications Dans ce tableau on a $f(2) = h$ et pas d. et $f(5) = d$ et pas h.
La vérification à donner 2 erreurs, on doit alors faire une autre vérification

sommet	adjacents	images	images adjacents	sommet	adjacents	etat
7	1,2,5,9	a	(b),(,h),(d),(e)	a	b,d,e,,h	ok
1	4,6,7,8,9	b	(i),(c),(a),(g),(e)	b	a,c,e,g,i	ok
2	6,7,8,9	h	(c),(a),(g),(e)	h	a,c,e,g	ok
5	3,4,7,9	d	(f),(i),(a),(e)	d	a,e,f,i	ok
9	1,2,4,5,7	e	(b),(,h),(i),(d),(a)	e	a,b,d,,h,i	ok
6	1,2,3,4,8	c	(b),(,h),(f),(i),(g)	c	b,f,g,,h,i	ok
8	1,2,6	g	(b),(,h),(c)	g	b,c,,h	ok
3	5,6	f	(d),(c)	f	c,d	ok
4	1,5,6,9	i	(b),(d),(c),(e)	i	b,c,d,e	ok

explications Maintenant aucune erreur donc les deux graphes **G** et **H** sont isomorphes et voici la solution finale :

sommet	Image
a	7
b	1
c	6
d	5
e	9
f	3
g	8
h	2
i	4

8 pseudo codes

8.1 pseudo-arbres

Voici un pseudo-code qui permet de générer un pseudo-arbre.

Algorithme 1 : générer un pseudo-arbre **P**(**a**) de tête **a** pour un graphe **G**

Données : Un graphe **G** et un sommet **a** de **G**
Résultat : Un pseudo-arbre **P**(**a**)

```
initialisation;
le sommet a dans le niveau 0;
les adjacents de a dans le niveau 1;
les sommets de G dans array graph[];
supprimer les sommets des niveau 0 et 1 de array graph[];
niv ← 1 ;
adjacents ← les adjacents de a ;
tant que graph[] n'est pas vide faire
|   initialiser array liste[] vide ;
|   niv ← niv + 1 ;
|   pour chaque element de adjacents faire
|   |   adjacents1 ← les adjacents de element ;
|   |   adjacents1 ← les intersection entre adjacents1 et graph[] ;
|   |   liste[] ← liste[]+adjacents1 ;
|   |   liste[] ← liste[] ordonner et sans doublon ;
|   |   graph[] ← la différence entre graph[] et liste[] ;
|   |   si graph[] est vide alors
|   |   |   sortir du boucle(break);
|   |   fin
|   fin
|   si liste[] est vide alors
|   |   sortir du boucle(break);
|   sinon
|   |   liste dans niveau niv ;
|   fin
fin
```

Pour les graphes non connectes il faut générer d'autres pseudo-arbres pour les autres sommets restants.

8.2 L'étiquette

Voici un pseudo-code qui calcule l'étiquette d'un sommet dans un pseudo-arbres.

Algorithme 2 : Calculer l'étiquette d'un sommet **s** dans un pseudo-arbre

Données : Un sommet **s** dans un pseudo-arbre
Résultat : Étiquette de **s** pour un pseudo-arbre

```
n ← déterminer le niveau de s;
nivh ← déterminer les sommets qui se trouvant dans le niveau n+1;
nivm ← déterminer les sommets qui se trouvant dans le niveau n;
nivb ← déterminer les sommets qui se trouvant dans le niveau n-1;
h ← le nombre des sommet adjacents s qui sont dans nivh;
m ← le nombre des sommet adjacents s qui sont dans nivm;
b ← le nombre des sommet adjacents s qui sont dans nivb;
label(s) ← "n-h-m-b";
```

8.3 Valider une solution

Voici un pseudo-code vérifier une solution proposé.

Algorithme 3 : Vérifier une solution

```
Données : Une solution proposé
Résultat : solution validé ou erroné
initialisation;
err ← 0;
pour Chaque f(a)=a' proposé faire
    adjacent(a) ← les adjacents de sommet a;
    adjacent(a') ← les adjacents de sommet a';
    pour chaque adjacents de a faire
        x ← un adjacent de sommet a;
        x' ← les images de x par l'isomorphisme;
        intersection ← l'intersection entre les images de x par l'isomorphisme et les
          adjacents de a';
        si intersection est vide alors
            err ← err+1;
            sortir du boucle(break);
        fin
    fin
    si err ≠ 0 alors
        sortir du boucle(break);
    fin
fin
si err = 0 alors
    "La solution est validé";
sinon
    "La solution n'est pas validé";
fin
```

8.4 remarque

Pour les graphes non connectes il faut bien sur étudier l'isomorphe entres les composants connectes.

9 Conclusion

Dans se papier j'ai présenté une version d'un algorithme polynomiale, qui nous donne les images de chaque sommets par la bijection isomorphisme, ou de dire que les deux graphes ne sont pas isomorphes.