Rapport:
Isomorphismes de graphes

Fanny Dufossé

1 Introduction

Le problème d'isomorphisme de graphes est pose de nombreuses questions. On sait qu'il appartient à la classe NP. Mais contrairement à la majorité des problèmes de NP, on n'a de preuve ni de son appartenance à la classe P, ni de sa NP-complétude. Le meilleur algorithme connu est de complexité $O(e^{\sqrt{(c.n.\ln(n))}})$. Cependant, pour certaines classes de complexité, il existe des algorithmes polynomiaux, voire linéaires. Je vais présenter quelques algorithmes polynomiaux concernant les isomorphismes de graphes.

2 Isomorphisme d'arbres

2.1 Isomorphismes d'arbres enracinés

L'algorithme que je vais présenter pour l'isomorphisme d'arbres enracinés vient du livre The Design and analysis of computer algorithms de Aho, Hopcroft et Ullman. Il consiste essentiellement à représenter de façon unique l'arbre. Il faut donc trier les fils de chaque nœud en fonction de leurs sous-graphes induits.

Entrées: Deux arbres T_1 et T_2
Sorties: vrai si T_1 et T_2 sont isomorphes, faux sinon

À chaque feuille de T_1 et T_2, on associe la valeur 0.
Soit H_i la hauteur de l'arbre T_i
pour i *de* $H-1$ *à* 0 **faire**
    On associe à chaque sommet de T_1 et T_2 et de hauteur i le uplet de valeurs associés à ses fils triés par ordre croissant.
    Soit S_1 et S_2 les listes des uplets obtenus pour T_1 et T_2.
    si $S_1 \neq S_2$ **alors**
        Renvoyer faux
    sinon
        Trier S_1 et associer à chaque sommet de T_1 et T_2 et de hauteur i l'indice de son uplet dans la liste triée.
        (les indices sont comptés à partir de 1)
    fin
fin
Renvoyer vrai

Algorithm 1: Algorithme pour des arbres enracinés

Le tri des uplets peut se faire par exemple par ordre lexicographique. Tous les ordres conviennent. De même, dans les uplets, on peut trier par ordre décroissant tout aussi bien que par ordre croissant. L'important est uniquement de trier les fils de chaque nœud. Les auteurs donnent une complexité linéaire à cet algorithme.

2.2 Algorithmes généralisés

On veut pouvoir généraliser l'algorithme précédent pour des arbres non enracinés. Pour cela, on veut repérer un faible nombre de sommets d'un arbre. On va donc calculer le centre de l'arbre :

tant que $n > 2$ **faire**
    enlever toutes les feuilles de T
fin

Algorithm 2: Algorithme de recherche du centre

On obtient ainsi un ou deux sommets particuliers d'un arbre. Il suffit maintenant de l'appliquer au problème d'isomorphisme :

Cet algorithme calcule donc en temps linéaire l'isomorphisme d'arbres, si on sait trouver en temps linéaire les feuilles d'un arbre.

Une autre solution consisterait à parcourir les graphes de la même manière que pour la recherche du centre, mais au lieu d'enlever les nœud à chaque étape, on leur assigne des uplets, puis des entiers, de la même manière que

Entrées: T_1 et T_2
Calculer les centres C_1 et C_2 de T_1 et T_2
si $|C_1| \neq |C_2|$ **alors**
    Renvoyer faux
fin
si $|C_1| = 1$ **alors**
    Enraciner les deux arbre en leur centre et lancer le premier algorithme
fin
si $|C_1| = 2$ **alors**
    Enraciner T_1 en $C_1[1]$
    Enraciner T_2 en $C_2[1]$
    lancer le premier algorithme
    Si il renvoie faux, enraciner T_2 en $C_2[2]$
    lancer le premier algorithme et renvoyer son résultat
fin

Algorithm 3: Algorithme généralisé

pour l'algorithme d'isomorphisme d'arbres enracinés. Cela revient à regrouper les deux algorithmes, au lieu de les développer l'un après l'autre comme dans l'algorithme précédent.

3 L'isomorphisme de cographes

Les cographes présentent l'avantage de pouvoir être représenté de façon unique par des arbres : les coarbres. Il faut donc adapter l'algorithme précédent aux coarbres dont les nœud internes sont indexés par S ou P. Il faut donc tenir compte de ces paramètres dans le tri. On peut donc résoudre l'isomorphisme de cographes en temps linéaire à partir des coarbres.

4 Algorithmes de prolongement

Le problème de prolongement est le suivant : étant donné 2 graphes G_1 et G_2, G_1 est-il un sous-graphe de G_2 ?

Ce problème est NP-complet. La preuve de NP-complétude se fait par réduction à partir de CLIQUE. En effet, chercher une clique de taille k dans un graphe G revient à se demander si la clique de taille k est isomorphe à un sous graphe de G.

Appliqué à des arbres, ce problème est polynomial. On trouve des algorithmes en $O(n_1^{1.5} \times n_2)$.

5 Conclusion

Le problème d'isomorphisme, extrêmement compliqué pour les graphes, est plus accessible pour les arbres ou les cographes. On obtient pour ces classes de graphes des algorithmes polynomiaux. Cependant, on ignore toujours si, dans le cas général, il est ou non NP-complet.

6 Bibliographie

The design and analysis of computer algorithms, Aho, Hopcroft et Ullman, Addison-Wesley (1974)
Algorithms on trees and graphs, G. Valiente, Springer (2002)