CHEMOUNE Alaeddine
VILLAVICENCIO Carlos Alberto Avila

4TIN704U

DM Complexité

Calculabilité et Complexité

M1 Informatique

Université de Bordeaux

2017/2018

Introduction :

L'objectif de ce devoir est de mettre en place un outil qui résoud le problème HAC et donne une solution possible après l'avoir transformé en formule SAT.

HAC est NP-difficile (optionnel) :

1. Une réduction polynomiale du problème Chemin hamiltonien vers HAC.

Si on pose dans le problème HAC l'entier k donné égal à n (le nombre de sommets du graphe), la hauteur de l'arbre cherché est maximale, ce qui signifie que l'arbre est lui même le chemin hamiltonien.

2. Qu'en concluez-vous sur la complexité de HAC ?

La complexité de HAC est vérifiable en temps polynomiale.

Réduction de HAC vers SAT

1. Pour chaque sommet $v \in V$, il y a un unique entier h t.q. x(v,h) est vrai.

$$/\backslash(v \in V)\ [(\backslash/(h=0..k)\ x(v,h)\ \ /\backslash\ \ /\backslash(i=0..k)(j=i+1..k)(\text{-}x(v,i)\backslash/\text{-}x(v,j))]$$

2. Il y a un unique sommet v t.q. d(v) = 0 ("v est la racine").

$$/\backslash(v,u \in V \text{ tq } v!=u)(\text{-}x(u,0)\backslash/\text{-}x(v,0))\ \ /\backslash\ \ \backslash/(v \in V)\ x(v,0)$$

3. Il y a au moins un sommet v t.q. d(v) = k.

$$\backslash/(v \in V)\ x(v,k)$$

4. Pour chaque sommet v, si d(v) > 0, il existe un sommet u tel que $uv \in E$ et d(u) = d(v) − 1 ("le sommet u est un parent potentiel de v dans l'arbre").

$$/\backslash(\ v \in V)\ [/\backslash(i=1..k)(\text{-}x(v,i)\ \backslash/\ \backslash/(u \in V \text{ tq } u!=v \text{ et } uv \in E)\ x(u,i\text{-}1))]$$

I . Bilan

Nous avons mis tout le projet dans un dossier Complexité où il ya les graphes proposés dans le site http://www.labri.fr/perso/dorbec/CoCa/ et les graphes des camarades en plus de deux petits graphes test1.c et test2.c pour le test et le fichier all.h qui contient les fonctions qui définissent un graphe, nous avons mis aussi le programme glucose qui résoud les formules SAT qui se trouvent dans les fichiers DIMACS.
Nos implémentations se trouvent dans le fichier satfile.c.
satfile.c contient une fonction satfile() qui prend 4 paramètres :
1- Une chaîne de caractères qui sera le nom du fichier DIMACS à créer.
2- Un entier qui présente le nombre de sommets du graphe.
3- Un entier qui présente le nombre de profondeurs possibles dans le graphe.
4- Une fonction booléenne qui présente les arrêtes du graphe.
(au début de fonction, nous avons mis une description des variables et ce qu'elles représentent).
La fonction crée un fichier et commence par écrire des commentaires dedans (contenu d'un ficher DIMACS), et puis les clauses de 1 à 4, on met un espace où il y a (\/) avec un compteur qui s'incrémente à chaque fois où il y a un (/\) un zéro et retour à la ligne, en fin de fonction, on se place au début de fichier à la position 35 avec un fseek pour écrire le nombre de clauses qui existent. dans le main on vérifie si le nom de fichier a été donné en paramètres, et on fait appel à la fonction satfile, en donnant en premier paramètre le nom de fichier.
Par manque de temps, nous n'avons pas réalisé l'outil qui dessiné l'arbre couvrant après le résultat positif, mais cela ne pose pas de problèmes techniques particuliers.

II . Points délicats

Nous avons pris beaucoup de temps au moment de la réalisation des clauses et l'identification de chaque sommet avec chaque profondeur et la transformation des clauses en code.
Choisir la façon de création des fichiers et faire le test a pris du temps également.

III . Limitations

Le projet permet de tester un graphe s'il contient un arbre couvrant de profondeur k, si oui, il affiche la profondeur de chaque sommet.
Pour déterminer l'entier k, il faut modifier sizeG() du graphe, On pouvait faire un scanf pour demander à l'utilisateur d'introduire k à chaque fois, mais nous n'avons pas pensé à cela au début.
Notre projet ne permet pas de voir un graphe dessiné, ni son arbre couvrant s'il existe.

IV . Tests

Pour faire le test, nous avons ajouté quelques commandes dans le Makefile de simp pour générer les fichiers DIMACS dans un dossier glucose/tests pour tous les graphes, nous avons commenté quelques lignes, car ce sont des graphes hyper grands, qui durent lentemps pour s'exécuter.
Donc on se place dans glucose/simp et on tape make en ce moment tous les DIMACS sont créés, puis on tape ./glucose -model ../tests/test1.sat. par exemple, pour tester test1, on aura le résultat en suite SATISFIABLE et une suite de chiffres qui précisent chaque sommet et sa profondeur ou UNSATISFIABLE au cas ou l'arbre n'existe pas.