Rapport DM
Complexité et Calculabilité

Serigne Amsatou SEYE

Etudiant en Master 1 Informatique - Groupe 2

Vendredi 03 Novembre 2017

I. Définition du problème

Un arbre couvrant d'un graphe non orienté G est un arbre constitué uniquement d'arêtes de G et contenant tous les sommets de G. Nos arbres couvrants seront toujours enracinés : un sommet est distingué et appelé la racine de l'arbre. La profondeur d(v) d'un sommet dans l'arbre est la distance par rapport à la racine, c'est-à-dire le nombre d'arêtes sur le chemin direct de la racine à v. La hauteur d'un arbre est la profondeur maximale d'un de ses sommets, $\max_{v \in V}(d(v))$. Formellement, le problème HAC est le suivant :

Entrée : Un graphe non-orienté G et un entier k.

Sortie : Existe-t-il un arbre couvrant de G de hauteur exactement k ?

II. Bilan

Lors de ce devoir maison, nous avons travaillé sur la réduction du problème HAC vers le problème SAT et la conversion de l'affectation des variables que nous a renvoyé le solveur en un arbre couvrant.

Pour les réductions, nous avons tout d'abord allouer un tableau à deux dimensions ((n+1)*(k+1)) dont les lignes sont les sommets (1 à n) et les colonnes les différentes profondeurs (0 à k). Ce tableau va être par la suite initialisé avec les n*(k+1) premiers entiers naturels qui seront les différentes variables dans le fichier au format DIMACS.

Pour l'écriture des clauses, les contraintes ont été exprimées comme suit:

1. **Pour chaque sommet $v \in V$, il y a un unique entier h t.q. $x_{v,h}$ est vrai.**

Cette contrainte regroupe deux "sous-contraintes":

- Chaque sommet a une profondeur h:
  Donc chaque sommet du graphe a soit une profondeur 0 ou 1 ou 2, etc jusqu'à k
- Un sommet ne peut pas avoir deux profondeurs différentes (pour appuyer l'unicité)
  Pour chaque sommet du graphe, on parcourt toutes les différentes profondeurs, de 0 à k, pour exprimer le fait qu'on ne puisse pas avoir par exemple $x_{v,h}$ et $x_{v,l}$
  ($0 <= h \neq l <= k$)
  donc une négation pour chaque variable.

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

Tout comme la première contrainte, celle-ci peut être subdivisée en deux autres "sous-contraintes":

- Il y a au moins un sommet de profondeur 0;
  Ici, chaque sommet peut avoir 0 comme profondeur; v_1 ou v_2 ou v_3 … ou v_n
- Deux sommets différents ne peuvent pas avoir une profondeur 0
  Pour s'assurer que la racine du graphe soit unique. Dans ce cas, nous ne pouvons plus avoir ce cas: $x_{v,0}$ et $x_{u,0}$ pour tout u,v ∈ V tq u≠v

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

Cette contrainte ressemble beaucoup à la première "sous-contrainte" de la deuxième contrainte sauf que ici nous avons k au lieu de 0. Donc nous pouvons avoir un sommet de profondeur k ou même un ou plusieurs autres différents du premier avec la même profondeur.

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

Cette contrainte nous a pris beaucoup de temps du fait qu'elle était la plus compliquée à écrire sa clause. Toutes les trois précédentes ont de quelques petites ressemblances mais celle-ci a la particularité d'avoir une autre condition. La question pour cette contrainte est bien comprise mais sa conversion reste un peu délicate. La solution que nous avons enfin trouvée est celle-ci: pour chaque sommet, nous parcourons ses possibles profondeurs en commençant par 1 (puisque d(v) > 0 puis nous faisons encore un autre parcours des sommets pour tester l'adjacence de deux sommets dont leurs profondeurs seront d(v) et d(v)-1. Après chaque sommet visité dans la deuxième boucle, nous avons une nouvelle clause.

Les variables et clauses sont écrites dans un fichier nommé formule.cnf au format DIMACS décrit dans le cahier des charges et un compteur s'incrémente pour avoir le nombre de clauses.

À la fin de l'écriture des clauses dans le fichier, nous nous plaçons au début de celui-ci pour mettre à jour le nombre de clauses.

En ce qui concerne la conversion de l'affectation des variables en arbre couvrant, nous avons créé un programme lexeur écrit en lex qui prend en entrée le la sortie du solveur, extrait juste l'affectation des variables, le transforme en un tableau pour avoir l'ensemble des sommets de notre nouvel arbre couvrant et leurs profondeurs. Après avoir récupérer toutes ces données, nous générons, dans un fichier nommé hac.c le nouvel arbre obtenu. Ce fichier contient trois fonctions qui sont dans tous les exemples de graphes qui nous ont été donnés avec le sujet.

Enfin, nous avons créé un script bash qui va nous permettre de choisir lequel des graphes nous voulons tester avec la profondeur de notre choix (saisi dans la console), lancer la compilation avec make, exécuter le solveur avec le fichier formule.cnf généré, et convertir l'affectation des variables en un arbre couvrant grâce à notre lexeur. Ce processus est répété autant que nous le souhaitons.

II. Points Délicats

La conversion de la quatrième contrainte pour la réduction en clause a été la plus compliquée pour les raisons expliquées dans le paragraphe précédant.
Plusieurs versions de cette contrainte ont été écrites mais nous en avons finalement choisir une qui nous paraît plus juste car testée avec succès avec l'arbre C10.c avec k = 6 par exemple et d'autres.

Une des problématiques dans ce devoir était d'extraire l'affectation des variable pour en faire un arbre. La fonction int are_adjacent(int u, int v) a été difficile à écrire car faire une bijection qui va nous permettre de connaitre le sommet et la profondeur de la variable n'était pas du tout évidente. Mais néanmoins nous y en sommes bien sortis.

III. Limitations

La partie 2 du devoir (optionnel) n'a pas été faite, par manque de temps et vu que ce n'était pas obligatoire de la faire.
La conversion de la quatrième contrainte a été implémentée mais il n'y a aucune assurance que ça marche totalement. En tout cas, avec certains graphes, ça semble correcte et bien faite.

IV. Tests

Nous n'avions pas pu faire le test avec tous les graphes qui nous est fournis. Le seul bémol est la réduction (à cause de la 4ème contrainte) mais tout le reste semble fonctionner très bien.
Certains tests effectués avec la réduction ont donné des résultats satisfaisants même si parfois il est difficile à le vérifier.
Nous aurions vouloir connaître pour chaque arbre parmi les tests, connaître ceux qui donneront des résultats positifs et ceux qui ne le donneront pas et pour quelle profondeur.