Complexité et Calculabilité 2017/18

Le problème de graphes *Hauteur d'arbre couvrant (HAC)* décrit ci-dessous est un problème NP-complet. Nous souhaitons dans ce devoir (1) réduire le problème HAC vers le problème SAT (satisfaisabilité de formules booléennes) et (2) utiliser un SAT-solveur pour résoudre HAC.

1 Définition du problème *HAC*

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 [1]. Nos arbres couvrants seront toujours *enracinés* : un sommet est distingué et appelé là *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 [2], $max_{v \in V}(d(v))$. 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 ?

2 *HAC* est NP-difficile (optionnel)

On rappelle qu'un *chemin hamiltonien* d'un graphe G est un chemin constitué d'arêtes du graphe visitant chaque sommet exactement une fois. Déterminer si un graphe possède un chemin hamiltonien est un problème NP-complet.

1. Montrez qu'il existe une réduction polynomiale du problème *Chemin hamiltonien* vers HAC.
2. Qu'en concluez-vous sur la complexité de HAC ?

1. En particulier, tout graphe admettant un arbre couvrant doit être connexe.
2. Remarquez qu'il suffit de regarder les feuilles.

Indication : quel arbre couvrant particulier possède tout graphe qui a un chemin hamiltonien ?

3 Réduction de *HAC* vers *SAT*

Décrivez une réduction polynomiale de *HAC* vers *SAT*.

Pour cela, étant donné G un graphe avec n sommets, $V = \{1, \ldots, n\}$, vous utiliserez des variables booléennes de la forme $x_{v,h}$, où $v \in V$ et $0 \leq h \leq k$. L'intuition est d'avoir $x_{v,h}$ vrai si la profondeur $d(v)$ de v dans l'arbre est égale à h.

Indications : Vous pouvez écrire des clauses (disjonctions de littéraux) pour exprimer les contraintes suivantes :

1. Pour chaque sommet $v \in V$, il y a un unique entier h t.q. $x_{v,h}$ est vrai.
2. Il y a un unique sommet v t.q. $d(v) = 0$ ("v est la racine").
3. Il y a au moins un sommet v t.q. $d(v) = k$.
4. Pour chaque sommet v, si $d(v) > 0$, alors 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").

4 Format des graphes

Pour vous épargner le travail consistant à écrire un parseur de graphe et à créer une structure de donnée graphe, vous permettant ainsi de vous focaliser sur l'efficacité de votre réduction, nous vous fournissons des fichiers C comportant chacun les fonctions suivantes :

```
int orderG();
int sizeG();
```

vous donnant respectivement le nombre de sommets et d'arêtes du graphe, et

```
int are_adjacent(int u, int v);
```

prenant en paramètre deux numéros de sommets (entre 0 et `orderG()-1`) et renvoyant 1 si u et v sont adjacents, 0 sinon.

Étant donnés un graphe et un entier k, vous devez générer une formule SAT qui représente l'assertion "le graphe fourni dispose d'un arbre couvrant de profondeur k" pour la donner à un solveur, qui décidera pour vous si la

formule est satisfaisable ou non. En outre, si un arbre couvrant existe et que le solveur retourne une affectation des variables qui satisfait la formule, vous devrez convertir cette affectation en un arbre couvrant.

Des exemples de fichiers C traduisant des graphes seront proposés sur la page suivante : `http://www.labri.fr/perso/dorbec/CoCa/`. Ils vous permettront de tester votre codage.

5 Solveur SAT

Le solveur que vous allez utiliser est `Glucose`, un solveur basé sur `Minisat` développé par G. Audemard et L. Simon, dont voici la page web [3]. Le format d'entrée de ce programme est un format DIMACS, utilisé pour les compétitions. Vous pouvez en trouver une description ici [4].

Le principe est simple, le contenu d'un fichier doit ressembler à cela :

```
c
c start with comments
c
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

où les lignes au début commençant par 'c' sont des commentaires, puis une ligne `p cnf nbvar nbclauses` qui annonce le nombre de variables et de clauses, enfin une ligne pour chaque clause (terminant par 0), indiquant les numéros des variables contenues dans les clauses précédées du signe `-` si la négation est utilisée.

6 Évaluation

Vous réaliserez le travail par binômes. Vous remettrez par mail votre code à vos responsables au plus tard le 3 novembre 2017, ainsi qu'une description (et justification !) de la réduction que vous utilisez et des optimisations que vous avez mises en place. Les évaluations auront lieu durant les TDs après les vacances de Toussaint. Votre code sera testé sur les machines du CREMI et devra donc pouvoir s'y executer.

3. `http://www.labri.fr/perso/lsimon/glucose/`
4. `http://www.satcompetition.org/2009/format-benchmarks2009.html`

Pour tester le code, nous vous fournirons de nouveaux programmes de génération de graphes pour tester les limites de votre programme.