Complexité et Calculabilité : projet 2018/19

Le problème de graphes *Triangles* décrit ci-dessous est un problème NP-complet. Vous allez dans ce devoir (1) réduire le problème *Triangles* vers le problème SAT (satisfaisabilité de formules booléennes en CNF) et (2) utiliser un SAT-solveur pour résoudre *Triangles*.

1 Définition du problème *Triangles*

Soit $G = (V, E)$ un graphe non-orienté. Un triangle T dans G est une clique de taille 3, autrement dit un ensemble $T = \{u, v, w\}$, tel que $u, v, w \in V$ sont distincts et $\{(u, v), (v, w), (w, u)\} \subseteq E$.

Une partition de G en triangles est une partition $V = T_1 \cup T_2 \cdots \cup T_k$ des sommets de G telle que chaque T_i est un triangle.

Problème *Triangles*

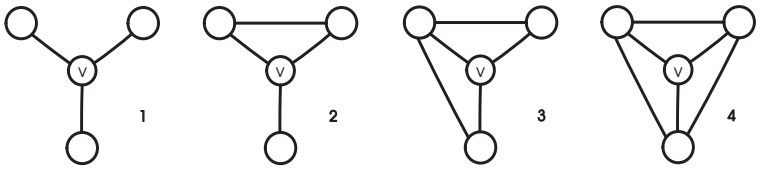
Entrée : Un graphe non-orienté G.

Sortie : Est-ce que G possède une partition en triangles ?

2 Degré 3 (optionnel)

Vous allez montrer que le problème *Triangles* peut être résolu en temps $O(|V| + |E|)$ (linéaire) si le degré de chaque sommet est au plus 3. Le degré d'un sommet est le nombre de voisins.

1. Que peut-on dire si G contient un sommet de degré 1 ?
2. Supposez que v est un sommet de degré 2. Montrez qu'on peut soit répondre "non" tout de suite, ou enlever des sommets de G, en obtenant G' tel que le G est instance positive de *Triangles* ssi G' l'est.
3. On suppose maintenant que tous les sommets ont degré 3.



Les 4 cas de la figure ci-dessus indiquent quel est le voisinage possible du sommet v. Raisonnez comme au point précédent.

4. Proposez un algorithme linéaire pour résoudre *Triangles* sur les graphes de degré maximal au plus 3.

3 Réduction de *Triangles* vers *SAT*

Le but de cette section est une réduction polynomiale de *Triangles* vers *SAT*.

Pour cela, étant donné G un graphe avec n sommets, $V = \{1, \ldots, n\}$ et ensemble d'arêtes $E \subseteq V \times V$, vous pouvez choisir une des deux réductions suivantes :

Réduction A. Vous utiliserez des variables booléennes de la forme $x_{u,v,1}$ et $x_{u,v,2}$, où $u, v \in V$. L'intuition est la suivante : chaque sommet doit avoir 2 voisins dans "son" triangle. Exemple : pour le triangle $\{1, 3, 6\}$ on peut choisir 6 comme premier voisin de 1, et 3 comme deuxième voisin. La variable $x_{u,v,1}$ est vraie si $uv \in E$ et v est le premier voisin dans le triangle de u ; la variable $x_{u,v,2}$ est vraie si $uv \in E$ et v est le deuxième voisin dans le triangle de u.

Réduction B. Vous utiliserez des variables booléennes de la forme $x_{u,v}$. L'intuition est la suivante : on peut donner une orientation à chaque triangle, ce qui définit un "successeur" pour chaque noeud. Par exemple, si $\{1, 3, 6\}$ forme un triangle, alors une orientation possible est $3, 1, 6$, le "successeur" de 1 étant 6, le "successeur" de 6 étant 3, et le "successeur" de 3 étant 1. La variable $x_{u,v}$ est vraie si v est le "successeur" de u (dans le sens décrit précedemment).

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

A Chaque sommet a exactement un premier voisin et un deuxième voisin, et ils sont différents ; si v est premier ou deuxième voisin de u, alors u est premier ou deuxième voisin de v, etc.

B Chaque sommet a exactement un successeur ; le successeur du successeur de v est égal à v, etc.

Ecrivez une réduction polynomiale de *Triangles* vers *SAT*, en utilisant soit la réduction A ou la réduction B.

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 vous devez générer une formule SAT qui représente l'assertion "le graphe possède une partition en triangles" pour la donner à un solveur, qui décidera pour vous si la formule est satisfaisable ou non. En outre, si le graphe possède une partition en triangles et que le solveur retourne une affectation des variables qui satisfait la formule, vous devrez convertir cette affectation en liste des arêtes appartenant à la partition. Facultativement, vous pourrez affichez graphiquement cette solution au problème *Triangles*, par exemple en colorant en rouge les arêtes de la triangulation et les autres en noir. Pour cela, il vous suffira de créer un fichier .dot décrivant votre graphe et de l'afficher grâce à la commande dot (de graphviz).

Des exemples de fichiers C traduisant des graphes seront proposés sur la page `http://www.labri.fr/perso/anca/MC/SAT/`. 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 [1]. Le format

1. `http://www.labri.fr/perso/lsimon/glucose/`

d'entrée de ce programme est le format DIMACS, utilisé pour les compétitions. Vous pouvez en trouver une description ici [2].

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 *obligatoirement* le travail par binômes. Vous déclarerez ces binômes sur la plateforme moodle [3] avant le 19 octobre 2018, en vous répartissant *à l'intérieur* de votre groupe de TD. Vous remettrez au plus tard le 5 novembre 2018 sur la plateforme moodle une archive contenant votre code, 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.

Attention, par «description et justification de votre réduction», nous entendons :

- La donnée explicite de la réduction,
- Une preuve qu'il s'agit d'une réduction *polynomiale*,
- Une preuve qu'il s'agit bien d'une réduction (i.e., la formule et satisfiable *si et seulement si* le graphe d'entrée est découpable en triangle).

Tout rapport ne contenant pas ces trois éléments verra sa note largement amputée.

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

2. `http://www.satcompetition.org/2009/format-benchmarks2009.html`
3. `https://moodle1.u-bordeaux.fr/course/view.php?id=4263`