Prépa CAPES NSI, 2019-2020 Université Claude Bernard Lyon1

Sujet 1 Blanc du 13/3/2020 - Partie Graphes et Complexité

Durée : 2h

# 1 Problèmes de décision sur les graphes

## Préambule

On rappelle les notions de base suivantes sur les graphes et la logique propositionnelle, suffisantes pour la bonne compréhension des trois problèmes à suivre.

**Graphes**

- Un graphe (orienté) $G$ est un couple $(S,A)$ où $S$ est un ensemble fini de sommets et $A \subseteq S \times S$ est l'ensemble de ses arcs.
- Deux sommets $x$ et $y$ d'un graphe $G=(S,A)$ seront dits *adjacents* si et seulement si $(x,y) \in A$ ou $(y,x) \in A$ (i.e., s'il existe un arc entre les deux sommets). L'adjacence est donc une relation symétrique.
- Soit un ensemble $C$ fini de $k$ *couleurs*. Un *$k$-coloriage* d'un graphe $G$ est une fonction qui affecte une et une seule couleur à chaque sommet. Il est "correct" si et seulement si deux sommets adjacents n'ont pas la même couleur. Un graphe est *$k$-coloriable* s'il existe un *$k$-coloriage* correct pour ce graphe.

**Logique propositionnelle**

- Étant donné un vocabulaire logique $V$ constitué de $v$ variables propositionnelles $p_1, \ldots, p_v$, une conjonction de clauses est une formule logique de la forme $E_1 \wedge \ldots \wedge E_i \ldots \wedge E_m$. Chaque clause $E_i$ est une disjonction de littéraux (différents deux à deux) $l_{i_1} \vee \ldots \vee l_{i_{k_i}}$, et un littéral est soit une variable propositionnelle $p$ de $V$ soit la négation $\neg p'$ d'une variable propositionnelle $p'$ de $V$.
- Une interprétation $I$ d'un ensemble $V = \{p_1, \ldots, p_v\}$ de variables propositionnelles est une fonction qui associe à chaque variable propositionnelle $p_i$ la valeur $Vrai$ ou $Faux$.
  - L'évaluation d'une conjonction de clauses dans une interprétation $I$ de ses variables propositionnelles renvoie $Vrai$ si *toutes* les clauses sont évaluées à $Vrai$ dans $I$, et $Faux$ sinon.
  - L'évaluation d'une clause dans une interprétation $I$ renvoie $Vrai$ si *au moins* un de ses littéraux est évalué à $Vrai$ dans $I$ et $Faux$ sinon.
  - Un littéral positif $p_i$ est évalué à $Vrai$ dans $I$ si $I(p_i) = Vrai$, à $Faux$ sinon.
  - Un littéral négatif $\neg p_i$ est évalué à $Faux$ dans $I$ si $I(p_i) = Vrai$, à $Vrai$ sinon.
- Une conjonction de clauses est satisfaisable s'il existe une interprétation de ses variables propositionnelles dans laquelle toutes les clauses sont évaluées à $Vrai$.

  **Exemple :** La conjonction de clauses $(\neg p_1 \vee p_3) \wedge (p_2 \vee \neg p_3) \wedge (\neg p_2)$
  - est évaluée à $Vrai$ dans l'interprétation $I$ définie par : $I(p_1) = I(p_2) = I(p_3) = Faux$,
  - est évaluée à $Faux$ dans l'interprétation $I'$ : $I'(p_1) = I'(p_2) = Faux, I'(p_3) = Vrai$.

  Elle est satisfaisable ($I$ suffit pour le prouver).

**NP-complétude, réduction polynomiale** Un problème $\mathcal{L}_2$ est NP-complet s'il est dans NP et s'il est NP-dur, i.e, si tout problème dans NP peut être réduit à $\mathcal{L}_2$, ce qui est équivalent à montrer qu'il existe une réduction polynômiale d'un problème NP-complet $\mathcal{L}_1$ *connu* vers $\mathcal{L}_2$.

On rappelle la définition formelle d'une réduction polynomiale d'un langage vers un autre langage : Une réduction polynomiale de $L_1$ *vers* $L_2$ est une fonction $f : \Sigma^* \to \Sigma^*$ *calculable en temps polynomial* telle que $x \in \mathcal{L}_1$ ssi $f(x) \in \mathcal{L}_2$ (transfert des solutions).

**Il est demandé d'expliquer et justifier les exemples et les algorithmes.**

## 1.1 Exemple 1 : coloriage de graphe non orienté

On code la relation d'adjacence associée à un graphe $G$ de $n$ sommets par un tableau T de taille $n \times n$ tel que : $T[i,j] = 1$ si les sommets $i$ et $j$ sont adjacents (cf. préambule), et $T[i,j] = 0$ sinon. On code un $k$-coloriage de $G$ par un tableau d'entiers $C$ à une dimension de taille $n$ tel que $C[i]$ vaut la couleur affectée au sommet $i$ dans le coloriage.

**Question #1**

Dessiner le graphe dont la matrice d'adjacence est déclarée par : (on numérote les sommets de 0 à 5, les indices débutent à 0) :

```
adj = np.array ([[0, 1, 1, 0, 0, 0],
  [1, 0, 0, 1, 0, 0],
  [1, 0, 0, 0, 1, 0],
  [0, 1, 0, 0, 1, 0],
  [0, 0, 1, 1, 0, 1],
  [0, 0, 0, 0, 1, 0]])
```

Dans la suite, nous nommons ce graphe $G_1$.

**Question #2**

Écrire un programme Python prenant en entrée la matrice d'adjacence d'un graphe, un tableau représentant un coloriage, le nombre de noeuds du graphe, et qui vérifie si ce coloriage est correct :

```
coloriage=[1,1,2,2,1,0]
print is_ok (adj,coloriage,6)
#False
coloriage=[0,1,2,3,4,5]
print is_ok (adj,coloriage,6)
#True
```

**Question #3**

Quel est l'ordre de grandeur de la complexité de ce programme en fonction du nombre $n$ de sommets du graphe ?

Une instance du problème de $k-$*coloriabilité* d'un graphe est la donnée d'un graphe et d'un entier, ie une paire $(G,k)$. La question associée est "existe-t-il un $k$-coloriage pour $G$ ?"

**Question #4**

Montrer que ce problème est NP.

Intéressons nous maintenant à la 3-coloriabilité : étant donné un graphe $G$ à $n$ sommets, on modélise par la variable propositionnelle $x_{i,c}$ ($i \in [0..n-1]$ et $c \in [0..2]$) que le sommet $i$ est colorié avec la couleur de numéro $c$.

**Question #5**

(a) Traduire par une conjonction de clauses que chaque sommet est colorié par une et une et une seule des 3 couleurs.
**Indication :** on exprimera par la clause $K(i)$ que le sommet $i$ est colorié en au moins une des 3 couleurs. On exprimera ensuite par la clause $U(i)$ (respectivement $V(i)$ et $W(i)$) que le sommet $i$ ne peut pas être colorié à la fois par la couleur 0 et la couleur 1 (respectivement la couleur 0 et la couleur 2, la couleur 1 et la couleur 2).

(b) Écrire cette conjonction de clauses dans le cas particulier du graphe $G_1$ de la question 1. *Pour des raisons de longueur, vous pouvez vous restreindre aux sommets 3 et 4.*

(c) Traduire par une conjonction de clauses que deux sommets voisins n'ont pas la même couleur.

(d) Écrire cette conjonction de clauses dans le cas particulier du graphe $G_1$ de la question 1. *Pour des raisons de longueur, vous pouvez vous restreindre au sommet 4 et à ses arêtes adjacentes.*

**Question #6**

En déduire l'existence d'une réduction polynômiale du problème de 3-coloriabilité de graphes vers SAT.

**Question #7**

Peut-on en déduire que le problème de 3-coloriabilité de graphes est NP-complet ?

Étant donnée une instance $\phi$ de 3-SAT, c'est-à-dire une conjonction $E_0 \wedge \ldots \wedge E_i \ldots \wedge E_{m-1}$ de $m$ clauses, construite à partir de $v$ variables propositionnelles $(p_0, \ldots, p_{v-1})$, où chaque clause contient exactement 3 littéraux (différents), on construit le graphe $G_\phi$ à $3v + m$ sommets défini de la manière suivante :

- Ses sommets sont $p_0, \ldots, p_{v-1}$, $\neg p_0, \ldots, \neg p_{v-1}$, $y_0, \ldots, y_{v-1}$, $E_0, \ldots, E_{m-1}$.
- Chaque sommet $p_i$ est relié au sommet $\neg p_i$.
- Chaque sommet $y_i$ est relié :
  * à tous les sommets $y_j$ tels que $j \neq i$,
  * à tous les sommets $p_j$ tels que $j \neq i$,
  * et à tous les sommets $\neg p_j$ tels que $j \neq i$.
- le sommet $p_i$ est relié au sommet $E_j$ si $p_i$ *n'est pas un littéral* de la clause $E_j$.
- le sommet $\neg p_i$ est relié au sommet $E_j$ si $\neg p_i$ *n'est pas un littéral* de la clause $E_j$.

Dans la suite on considérera l'ordre des sommets du graphe de taille $3v+m$ dans l'ordre suivant : d'abord les $p_i$, puis les $\neg p_i$, puis les $y_i$, puis les $E_i$. $p_0$ est donc le sommet de numéro 0, $\neg p_0$ le sommet de numéro $v$, ....

**Question #8**

Soit $\phi_1$ l'instance de 3-SAT suivante :

$$(p_0 \vee \neg p_1 \vee \neg p_2) \wedge (\neg p_0 \vee p_1 \vee \neg p_2).$$

Que valent $v$, $m$ ? Dessiner le graphe $G_{\phi_1}$ et donner sa matrice d'adjacence. *Les points de suspension sont autorisés...*

On code en Python une instance de 3-SAT $\phi$ (avec les notations précédentes) sous la forme d'une matrice de $F$ taille $m \times v$ telle que $F[i, j]$ vaut :

- 1 si $p_i$ est un littéral de la clause $E_j$,
- $-1$ si $\neg p_i$ est un littéral de la clause $E_j$,
- et 0 si ni $p_i$ ni $\neg p_i$ n'est un littéral de la clause $E_j$,

**Question #9**

(a) Que vaut $F$ dans le cas de la formule $\phi_1$ ? On nomme cette matrice $F_1$.

(b) Quel est le numéro du noeud représentant la clause $E_j$ ?

(c) Coder dans un programme Python la transformation qui prend le codage d'une instance de 3-sat (F, matrice de taille $m \times v$) et qui construit en sortie la matrice codant le graphe correspondant :

```
1 def transforme_formule(F,m,v):
2     T=np.zeros((3*v+m,3*v+m),dtype=np.int32) #init de T avec des 0
3     #à compléter
4     return T
```

L'appel `transforme_formule(F1,..,..)` doit retourner la matrice d'adjacence de $G_{\phi_1}$ obtenue à la question 8.

(d) Quelle est la taille de l'entrée, et le nombre d'instructions effectuées par votre programme en fonction de $m$ et $v$ ?

**Question #10**

Montrer que si $k < v$, alors il n'existe pas de $k$-coloriage correct du graphe.

**Question #11**

On considère un $(v+1)$-coloriage qui associe la couleur $i$ à l'un des deux membres de chaque paire de sommets $\{p_i, \neg p_i\}$, et la couleur $v+1$ à l'autre.
Montrer que si $v > 4$, alors, dans tout $(v+1)$-coloriage correct qui étend aux sommets $E_j$ le $(v+1)$-coloriage précédent, aucun sommet $E_j$ n'a la couleur $v+1$.

**Question #12**

Montrer comment associer une couleur parmi $1, \ldots, v$ à chaque sommet $E_j$ de sorte que $G$ soit $(v+1)$-coloriable si et seulement si l'ensemble de toutes les clauses $E_j$ est satisfaisable. On pourra raisonner sur un exemple.

**Question #13**

Peut-on en déduire que le problème de $k$-coloriage d'un graphe est NP-complet ?

## 1.2 Exemple 2 : chemins et circuits dans un graphe orienté

On considère ici la représentation des graphes orientés par liste d'adjacence. En Python on propose l'implémentation à l'aide d'un dictionnaire, par exemple :

```
g = {"a": ["d"],          "b": ["c"],          "c": ["b", "d"],
     "d": ["c", "b"],     "e": ["c"],          "f": []}
```

**Recherche en Python**

**Question #1**

Donner un algorithme en Python qui permet de décider si un sommet $d$ est accessible à partir d'un sommet $s$ dans un graphe orienté.

**Question #2**

Quelle est la complexité algorithmique de votre algorithme en considérant la représentation par liste d'adjacence ?

**Question #3**

Donner un algorithme en Python qui permet de décider si deux sommets donnés font partie d'un même circuit. Quelle est sa complexité ?

**Complexité de la recherche** On s'intéresse à la réduction polynomiale de 2-SAT vers la recherche de chemin dans un graphe orienté. On rappelle que 2-SAT est la satisfiabilité d'une FNC (Forme Normale Conjonctive) $F$ comportant **au maximum** 2 littéraux par clause.

On considère la transformation suivante d'une instance $F$ de 2-SAT en un graphe orienté $G$ appelé le *graphe d'implications* de $F$.

- Pour chaque variable propositionnelle $x_i$ apparaissant dans $F$, $G$ a deux sommets étiquetés $x_i$ et $\overline{x_i}$.
- Pour chaque clause binaire $x_i \vee x_j$ de $F$, on crée une arête du sommet $\overline{x_i}$ vers le sommet $x_j$ et une arête du sommet $\overline{x_j}$ vers le sommet $x_i$ traduisant les implications "si $x_i$ est faux alors $x_j$ doit être vrai" et "si $x_j$ est faux alors $x_i$ doit être vrai". On rappelle que $\overline{\overline{x_i}} = x_i$.
- Pour chaque clause unaire $x_i$, on crée une arête du sommet $\overline{x_i}$ vers le sommet $x_i$.

**Question #4**

Dessinez le graphe d'implications correspondant à la conjonction de clauses
$(\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge (\overline{x_3} \vee x_1) \wedge x_1$.

**Question #5**

Donnez l'ordre de grandeur de la complexité de la construction du graphe d'implications en fonction du nombre $n$ de variables et du nombre $m$ de clauses de l'instance 2-SAT à transformer.

**Question #6**

Soit $F$ une instance de 2-SAT et $G$ le graphe d'implications correspondant.
Montrez que, s'il existe dans $G$ un circuit passant par deux sommets $x_i$ et $\overline{x_i}$, alors $F$ est insatisfiable.

**Indication :** Montrez (par double implication) que les littéraux reliés par une chaîne fermée d'implications (un circuit du graphe $G$) ne peuvent qu'avoir la même valeur de vérité (Vrai ou Faux), dans une interprétation satisfiant $F$.

**Question #7**

On considère que deux sommets sont équivalents ($u \approx v$) s'ils sont sur un même circuit. On construit le graphe $G'$ à partir de $G$ en remplaçant chaque ensemble de sommets reliés 2 à 2 par un circuit par un sommet unique $[u]$ correspondant à tous les sommets $v$ tels que $u \approx v$), et en conservant les sommets non impliqués dans un circuit.

Dans $G'$, une arête relie deux sommets distincts $[u]$ et $[v]$ si et seulement si il existe une arête dans $G$ reliant un des sommets regroupés dans $[u]$ avec un des sommets regroupés dans $[v]$.

**Remarques.** Si $G$ est sans circuit, $G' = G$. Par construction, si $G$ a des circuits, $G'$ est sans circuit. $G'$ a donc au moins un sommet sans prédécesseur, on peut donc l'explorer en largeur à partir d'un tel sommet.

Dessinez le graphe $G'$ du graphe d'implications $G$ de la question 1. Pourquoi $G'$ est-il ainsi réduit ?

**Question #8**

Les résultats des deux parties (chemins dans un graphe orientés) sont-ils cohérents ?