Chloé DESDOUITS
Guillerme DUVILLIE
Swan ROCHER

M1 Informatique

Projet Transversal - Coloration de Graphes

2012

UM2

Table des matières

0.1 Exercice 14 - Coloration de Sommets

Algorithm 1 Approximation de χ(G) séquentielle

Require: G : graphe de n sommets
Ensure: $\chi(G)$: nombre de couleurs nécessaires pour colorer G

```
Soit x_1, ..., x_n une numérotation des sommets de G.
Soit C = {1, 2, ..., k} un ensemble de couleurs.
for all x_i do
    χ(G) = min{k ∈ C : ∀y ∈ voisinage(x_i), χ(y) ≠ k}
end for
return χ(G)
```

0.1.1 Question 1

Soit la numérotation suivante (parcours en profondeur à partir du premier sommet) :

En appliquant l'algorithme ci-dessus on obtient la coloration suivante :

On remarque que sur cet exemple, la solution est optimale : $\chi(G) = 3$.

0.1.2 Question 2

Malheureusement la solution renvoyée n'est pas toujours tr'es bonne, en effet il est possible de construire un graphe 2-colorable pour lequel l'algorithme renverra $\frac{n}{2}$.

En effet, soient les graphe bipartis $G = (V, E)$ suivants :

– $V = \{1, 2, ..., n\}$;
– $E = \{(i, j) \forall i, j \in V \ : \ i \ impair, \ \ j \ pair, \ \ et \ j \neq i + 1\}$

En exécutant l'algorithme sur un tel graphe, on voit rapidement que le nombre de couleurs utilisées sera bien égal au nombre de sommets divisé par deux. Sur l'exemple de la figure, on voit que pour $n = 6$, la solution $\chi_{alg}(G_{fig3}) = 3$ tandis que $\chi_{opt}(G_{fig3}) = 2$.

0.2 Exercice 15 - Coloration de Sommets

Algorithm 2 Approximation de χ(G) via les degrés des sommets

Require: G : graphe de n sommets
Ensure: $\chi(G)$: nombre de couleurs nécessaires pour colorer G

```
C = ∅
k = 0
d_1 >= d_2 >= ... >= d_n les degrés des sommets
repeat
    k = k + 1
    colorier le sommet x ayant le plus haut degré avec k
    marquer les voisins de x
    C = C ∪ {x}
    while ∃ un sommet y non marqué et non coloré do
        colorier y ayant le plus haut degré avec k
        marquer les voisins de y
    end while
    effacer toutes les marques
until tous les sommets sont colorés
return k
```

0.2.1 Question 1

Soit la numérotation suivante (en fonction des degrés) :

En appliquant l'algorithme ci-dessus on obtient la coloration suivante :

On remarque que sur cet exemple, la solution est également optimale : $\chi(G) = 3$.

0.2.2 Question 2

On doit trouver un graphe 2-colorable pour lequel l'algorithme renvoie n.
Prouvons qu'il n'en existe aucun. Soit $G = (V, E)$ un tel graphe.
La seule manière pour que cet algorithme renvoie n, est qu'à chaque itération, k soit incrémenté, i.e. il ne doit jamais rentrer dans la boucle while.
Soit x le sommet de degré maximum de G.
Si degré(x)$< n - 1$, alors soit y un sommet de G tel que $(x, y) \notin E$. L'algorithme lui attribut alors la même couleur (1) que x (en passant dans le while). Ne restant que n-2 sommets à colorer tout en ayant utilisé qu'une seule couleur, l'algoritme renverra donc au plus n-1 couleurs. Ce qui est absurde par hypothèse.
Donc degré(x) = n-1.
Soit $G_2 = (V_2, E_2)$ le sous graphe de G privé de x.
On note x_2 le sommet de degré maximum de G_2.
Si x_2 n'est pas adjacent à tous les sommets non colorés alors comme précédemment deux sommets possèderont la même couleur. Ce qui empêchera l'algorithme de renvyé n. x_2 est donc adjacent à tous les sommets de G_2, ie degré(x_2) $>= n - 2$.
Or x est adjacent à x_2 (cf plus haut), et donc degré(x_2) = n - 1.
En raisonnant de la même manière pour tous les sommets, nous arrivons donc à la conclusion que G est le graphe complet à n sommets.
C'est à dire que G ne peut être coloré au mieux qu'avec n couleurs. Et donc lorsque l'algorithme renvoie n, le graphe ne peut pas être 2-colorable.

Par contre, il existe des graphes 2-colorables pour lequel l'algorithme renvoie $\frac{n}{2}$, par exemple la couronne à n sommets (en supposant que la numérotation des sommets fait les pires choix possibles) :

0.3 Exercice 16 - Coloration des Sommets d'un Graphe

0.3.1 Question 1

Nous devons montrer que l'algorithme 1 n'est pas r-approché quel que soit $r \in N$.

Soit la propriété P_r : "l'algorithme n'est pas r-approché". Par récurrence sur $r \in N$:

P_1 : Si P_1 est fausse, la solution renvoyée est optimale, l'algorithme étant polynomial et le problème de coloration de sommets appartenant à la classe NP, il est évident que P_1 est vraie (si $P \neq NP$).

Sous l'hypothèse que P_r est vraie, nous allons montrer que P_{r+1} l'est également :
On suppose que P_{r+1} est fausse (démonstration par l'absurde).
L'algorithme renvoie donc $k \times (r+1)$ couleurs. Soit $G' = G \setminus \{v : couleur(v) > k \times r\}$ (c'est à dire qu'on retire de G tous les sommets colorés par les k dernières couleurs).
Il est trivial de voir que l'algorithme renvoie donc une solution r-approchée pour G' (tout en restant polynomial en fonction du temps).
Or ceci est impossible par hypothèse de récurrence $\rightarrow$ absurde.
P_{r+1} est donc vraie.

0.3.2 Question 2

Nous devons maintenant montrer qu'il existe toujours un ordre des sommets tel que l'algorithme 1 renvoie une solution optimale.

Soit G un graphe quelconque.
Soit $couleur(x_i) \forall x_i \in G$ une coloration optimale de G.
Soit la numérotation $\{y_i\} \forall y_i \in G$ telle que $couleur(y_i) <= couleur(y_i + 1)$.
Il est évident qu'une telle numérotation existe et qu'en suivant celle-ci l'algorithme renverra toujours la solution optimale.

0.3.3 Question 3

Le problème de savoir si un graphe est 2-colorable est polynomial puisque l'algorithme suivant permet toujours de donner la réponse :
On attribue au premier sommet la couleur 1, puis la couleur 2 à tous ses voisins, à nouveau la couleur 1 à leurs voisins respectifs, et on continue ainsi tant qu'il n'existe pas deux sommets adjacents partageant la même couleur.
Si aucune erreur n'est repérée a la fin de l'algorithme, le graphe est 2-colorable.

0.3.4 Question 4

Nous devons montrer que si G est 3-colorable, alors $\forall x \in G$ $voisinage(x)$ est un graphe biparti.

Soit G un graphe 3-colorable.
Soit $x \in G$ un sommet quelconque. On note 1 la couleur de x.
Soit G_x le graphe du voisinage de x.
Puisque tous les sommets $y \in G_x$ sont adjacents à x par construction, ils ne peuvent pas être de sa couleur. C'est à dire que $couleur(y) > 1$.
Or G est 3-colorable, donc $couleur(y) <= 3$.
On a donc que $couleur(y) \in \{2, 3\}$. Ie, G_x est 2-colorable, et tout graphe 2-colorable admet une bipartition de ses sommets (cf **??**).
G_x est donc un graphe biparti.

0.3.5 Question 5

Il est question de trouver un algorithme $\sqrt{n}$-approché permettant de colorer un graphe 3-colorable. Soit l'algorithme suivant :

Algorithm 3 Coloration de graphe 3-colorable

Require: G : graphe de n sommets 3-colorable
Ensure: $couleur(x)$: couleur attribué au sommet x $\forall x \in G$

```
trier les x_i ∈ G par ordre décroissant de degré
k ← 1
couleur ← ∅
repeat
    x ← sommet(G)
    G' ← voisinage(x)
    couleur ← couleur ∪ 2coloration(G', k)
    k ← k + 2
    G ← G \ G'
until deg(sommet(G)) < √n
couleur ← couleur ∪ DMaxPlus1coloration(G, k)
return couleur
```

Il faut maintenant prouver que cet algorithme renvoie bien une $\sqrt{n}$-approximation.

Tout d'abord l'opération $2coloration(G, k)$ (qui colore un graphe 2-colorable avec les couleurs k et k+1) est expliquée en question 3 (0.3.3).
Quant à $DMaxPlus1coloration(G, k)$ qui colore un graphe avec les couleurs $[k, k + deg_{max}(G)]$, il suffit de voir qu'il est simple de colorer G si on s'autorise 1 couleur de plus que le degré maximum de G. En effet puisque chaque sommet a au plus D voisins, il reste toujours au moins une couleur non utilisée.
De plus, ces deux opérations sont évidemment polynomiales.

Il faut donc compter le nombre de couleurs utilisées.
A chaque fois qu'on entre dans la boucle on élimine $\sqrt{n}$ sommets (puisque le sommet x a un degré d'au moins $\sqrt{n}$).
On entrera donc dans cette boucle au plus $\frac{n}{\sqrt{n}} = \sqrt{n}$ fois. En effet, une fois ce nombre de passages effectués le degré maximum de G est forcément inférieur à $\sqrt{n}$.
Or à chaque itération on utilise seulement deux couleurs, en effet G' est 2-colorable (0.3.4).
Ce qui amène le nombre de couleurs à $2 \times \sqrt{n}$ à la fin de la boucle.
Une fois celle-ci terminée il ne nous reste plus que la dernière coloration qui utilise donc une couleur de plus que le degré maximum de G.
Or $deg_{max}(g) < \sqrt{n}$, ce qui au pire ajoutera encore $\sqrt{n}$ couleurs.
En faisant la somme nous arrivons donc à $3 \times \sqrt{n}$ couleurs.

Le graphe étant 3-colorable, le nombre de couleurs optimal est 3, donc en calculant le ratio de cet algorithme par rapport à la meilleure solution nous obtenons bien : $\frac{3\times\sqrt{n}}{3} = \sqrt{n}$.

Cet algorithme est bien $\sqrt{n}$-approché.

0.3.6 Question 6

Si un graphe dont le voisinage de tout sommet est un graphe biparti impliquait que ce graphe est 3-colorable, alors il existerait un algorithme polynomial permettant de savoir si un graphe est 3-colorable (problème NP) :

Algorithm 4 Graphe 3-colorable ?

Require: G : graphe de n sommets
Ensure: vrai ssi G est 3-colorable, faux sinon

for all sommet $x \in G$ **do**
    $G_x \leftarrow voisinage(x)$
    if G_x n'est pas 2-colorable **then**
        return faux
    end if
end for
return vrai

Et donc en supposant que $P \neq NP$, ceci est impossible.

0.3.7 Question 7

0.4 Exercice 21 - Coloration et Homomorphisme

0.4.1 Question 1

Soit $G = (V_G, E_G)$ un graphe quelconque connexe (s'il n'est pas connexe, il suffit d'appliquer la démonstration à chaque composante).
Montrons que G est k-colorable $\leftrightarrow \exists$ un homomorphisme de G dans une k-clique $K_k = (V_K, E_K)$.

$\rightarrow$ G est k-colorable.
Soit $W = (V_W, E_W)$ la plus grande clique de G.
On sait que $\chi(W) \leq \chi(G)$ (cf **??**). On note $x_1, ..., x_w$ les sommets de W, $x_{w+1}, ..., x_n$ les sommets de $G \setminus W$. Et $y_1, ..., y_k$ les sommets de K_k.
Puisque $\chi(K_n) = n$ on remarque que $w \leq k$.
Soit h une fonction de V_G dans V_K construite de la manière suivante :

- (i) $h(x_i) = y_i \forall i \in [1, w]$
- (ii) $h(x_i) \in V_K \setminus h(x_j) : \exists (x_i, x_j) \in E_G$

Les arêtes de W sont bien conservées par (i), K_k étant complet par définition. Quant à celles ayant une extrémité $x \in V_G \setminus V_W$, on sait qu'il existe au moins un sommet x' de V_W tel qu'il n'existe pas $(x, x') \in E_G$. En effet si tel était le cas, le sommet x ferait partie de W.
Ainsi il suffit de poser $h(x) = h(x')$ et ses arêtes seront également respectées.
La fonction h étant bien définie pour tout $x \in V_G$ et conservant les arêtes, celle-ci est bien un homomorphisme de G dans K_k.

$\leftarrow \exists$ un homomorphisme h de G dans K_k.
Intuitivement, un homomorphisme est une fonction qui "contracte" les sommets du graphe de départ dans celui d'arrivée. Ainsi en appliquant h de G dans K_k, il suffit de colorer chaque y_i avec la couleur i, puis de "redéplier" G à partir de son image dans K_k, et tous les sommets sont colorés avec seulement k couleurs.
Plus formellement, soit la coloration classique de K_k suivante : $couleur(y_i) = i \ \forall i \in [1, k]$.
Attribuons à chaque $x_i \in V_G$ la couleur de $h(x_i)$.
Puisque h conserve les arêtes, il est trivial de voir que la coloration de G est valide. De plus k couleurs ont été utilisées.
Donc G est k-colorable.

0.4.2 Question 2

Nous savons que le problème k-colorable est NP-complet.
De plus, nous pouvons définir l'algorithme suivant :

Algorithm 5 k-colorable ?

Require: G : graphe de n sommets, $k \in N$
Ensure: vrai ssi G est k-colorable, faux sinon

```
K_k ← k − clique
if ∃h un homomorphisme de G dans K_k then
    return vrai
else
    return faux
end if
```

Sous l'hypothèse que $P \neq NP$, et que la recherche d'homomorphisme dans un graphe est polynomiale, nous avons donc un algorithme polynomial permettant de décider si un graphe est k-colorable. Ce qui est absurde.
HOM appartient donc à la classe de problème NP-complet.

0.5 Exercice 22 - Homomorphisme et Satisfaction de Contraintes

Ici, nous nous intéressons à la réduction polynomiale du problème de l'existence d'une solution d'un réseau de contraintes (Constraint Satisfaction Problem) vers le problème de recherche d'homomorphisme dans un graphe (HOM).

Soit le CSP suivant : C = (X,D,C,R) avec :
- $X = \{x_i : \forall i \in [1, n]\}$ l'ensemble des variables de C,
- $D = \cup_{i \in [1,n]} D_i$ où chaque D_i est le domaine de la variable $x_i \in X$,
- $C = \{C_j : \forall j \in [1, p]\}$ où chaque C_j est une contrainte définie par un ensemble ordonné de variables $x_{ji} \in X_j \in X \forall i \in [1, q]$,
- $R = \{R_j : \forall j \in [1, q]\}$ où $R_j \in \{X_{i \in [1,q]} D_{ji}\}$ est la définition de la contrainte C_j.

Résoudre C revient à déterminer si ce réseau est consistant.

Nous allons construire deux graphes bipartis différents $G_X = (V_X, E_X, \omega_X)$ (la "requête") et $G_R = (V_R, E_R, \omega_R)$ (la base de connaissance).

Soit G_X défini comme suit :

- $V_X = \{x_i \in X\} \cup \{C_j \in C\}$,
- $E_X = \{(x_i, C_j) : x_i$ est une variable apparaissant dans $C_j\}$,
- $\omega_X : E_X \to N$ une fonction définie telle que $\omega_X(x_i, C_j) = p$ avec p la position de x_i dans C_j.

Soit G_R défini comme suit :
- $V_R = \{D_{ji} \in R\} \cup \{C_j \in C\}$,
- $E_R = \{(D_{ji}, C_j)\}$,
- $\omega_R : E_R \to N$ une fonction définie telle que $\omega_R(D_{ji}, C_j) = q$ avec q la position de D_{ji} dans C_j.

C est consistant $\leftrightarrow \exists h$ un homomorphisme de G_X dans G_R.

Preuve ? Bijection entre h et la fonction solution de C...