3 NP-complétude

Dans cette partie, on cherche à montrer que le problème général du calcul du nombre chromatique d'un graphe est difficile. Pour ce faire, on étudie plutôt des problèmes de décision liés à la coloration. On définit les problèmes de décision suivants :

- Problème k-coloration (pour k fixé)
  - Entrée : un graphe G.
  - Question : le graphe G est-il k-colorable ?
- Problème Coloration
  - Entrée : un graphe G et un entier k.
  - Question : le graphe G est-il k-colorable ?

Par ailleurs, pour A et B deux problèmes de décision, la notation $A \leqslant_{\mathsf{m}}^{\mathsf{p}} B$ signifie qu'il existe une réduction many-one polynomiale de A à B.

Question 15 Montrer que pour tout entier k, k-coloration $\leqslant_{\mathsf{m}}^{\mathsf{p}}$ Coloration.

Question 16 Montrer que pour $h \leqslant k$ deux entiers fixés, h-coloration $\leqslant_{\mathsf{m}}^{\mathsf{p}}$ k-coloration.

Question 17 Montrer que 2-coloration $\in$ P. On décrira en français un algorithme polynomial permettant de résoudre le problème.

Question 18 Écrire une fonction `bool deux_colorable(liste** G, int n)` qui prend en argument un graphe `G` et renvoie `true` si G est 2-colorable et `false` sinon. Cette fonction devra avoir une complexité linéaire en le nombre de sommets et d'arêtes de G.

Les deux questions précédentes montrent que les problèmes 1-coloration et 2-coloration sont « faciles ». Les questions suivantes montrent la NP-complétude de 3-coloration.

Question 19 Montrer que Coloration $\in$ NP.

On rappelle les définitions suivantes : pour un ensemble de variables $V = \{v_1, \ldots, v_n\}$, un *littéral* est une variable ou sa négation, c'est-à-dire de la forme v_i ou $\overline{v_i}$. Une *clause* est une disjonction de littéraux (séparés par des $\vee$). Une formule est dite en *forme normale conjonctive* si c'est une conjonction de clauses (séparées par des $\wedge$). Elle est dite en 3-FNC si de plus chaque clause contient exactement 3 littéraux. Par exemple, la formule φ_0 suivante est en 3-FNC :

$$\varphi_0 = (a \vee b \vee c) \wedge \overbrace{(\overline{a} \vee c \vee \overline{d})}^{\text{clause}} \wedge (b \vee \underbrace{\overline{c}}_{\text{littéral}} \vee d)$$

On définit alors le problème de décision :

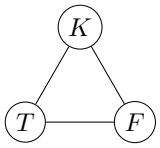
- Problème 3-SAT
  - Entrée : une formule booléenne φ en 3-FNC.
  - Question : la formule φ est-elle satisfiable ?

On admet que ce problème est NP-complet. La suite de cette partie a pour objectif de montrer qu'il se réduit en temps polynomial au problème 3-coloration. Pour ce faire, en considérant une formule booléenne φ en 3-FNC, on construit un graphe G_φ qui est 3-colorable si et seulement si φ est satisfiable. L'idée est que deux des couleurs utilisées correspondent à l'affectation *Vrai* ou *Faux* pour chaque littéral, et la troisième couleur est une couleur de contrôle.

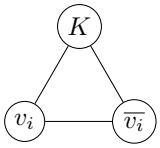
Dans le détail, si φ est une formule en 3-FNC sur un ensemble $V = \{v_1, \ldots, v_n\}$ de n variables, alors on définit le graphe $G_\varphi = (S, A)$ par :

- S contient trois sommets T (*True*), F (*False*) et K (*Control*), un sommet par littéral possible, et 5 sommets supplémentaires par clause ;

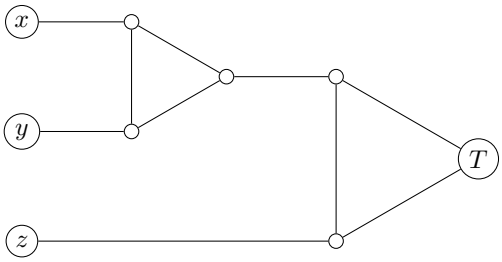
– A contient les arêtes suivantes :
  * $\{T, F\}, \{T, K\}, \{F, K\}$



  * pour chaque variable v_i, les arêtes $\{v_i, \overline{v_i}\}, \{v_i, K\}, \{\overline{v_i}, K\}$



  * pour chaque clause $C = x \vee y \vee z$ apparaissant dans φ (où x, y, z sont des littéraux), on rajoute les 10 arêtes telles que décrites dans le graphe ci-dessous, en utilisant les 5 sommets supplémentaires associés à la clause.



Par exemple, la figure 3 représente le graphe G_{φ_0} obtenu pour la formule booléenne φ_0 sur l'ensemble de variables $\{a, b, c, d\}$.

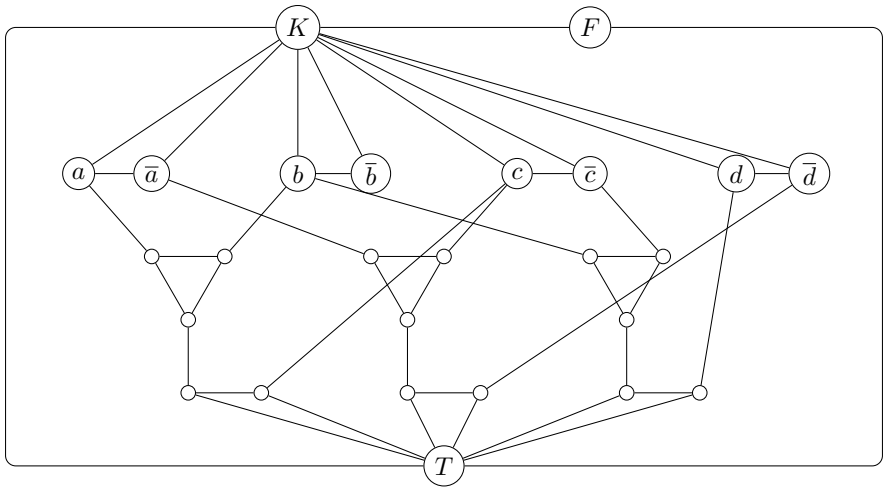


FIGURE 3 – Le graphe G_{φ_0} pour $\varphi_0 = (a \vee b \vee c) \wedge (\overline{a} \vee c \vee \overline{d}) \wedge (b \vee \overline{c} \vee d)$.

Question 20 Représenter graphiquement le graphe G_{φ_1} pour $\varphi_1 = (\overline{a} \vee b \vee \overline{c}) \wedge (a \vee b \vee \overline{c}) \wedge (a \vee \overline{b} \vee c)$.

Question 21 Justifier que la construction de G_φ à partir d'une formule φ en 3-FNC est en temps polynomial en fonction de la taille de φ (la taille d'une formule en 3-FNC est son nombre de clauses).

Question 22 Montrer que le graphe de la figure 4 est 3-colorable et que pour toute 3-coloration, au moins l'un des sommets x, y ou z a la même couleur que T.

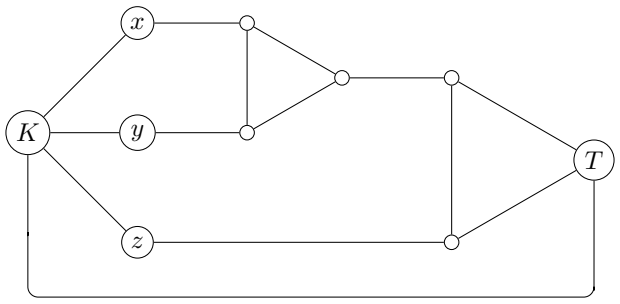


FIGURE 4 – Un gadget de clause.

Question 23 On suppose que φ est satisfiable. Montrer que G_φ est 3-colorable. On expliquera comment construire une 3-coloration de G_φ à partir d'une valuation de φ.

Question 24 On suppose que G_φ est 3-colorable. Montrer que φ est satisfiable. On expliquera comment construire une valuation de φ à partir d'une coloration de G_φ.

Question 25 En déduire que 3-coloration est NP-difficile, puis qu'il est NP-complet.

4 Algorithmes gloutons

Comme vu précédemment, le calcul du nombre chromatique d'un graphe est un problème difficile. On se propose donc d'étudier dans cette partie des algorithmes gloutons efficaces permettant de déterminer des k-colorations d'un graphe G, sans garantir que $k = \chi(G)$. Le schéma général de tous ces algorithmes est le même, seul l'ordre de traitement des sommets pouvant différer.

Début algorithme
  Entrée : graphe $G = (S, A)$
  Pour chaque sommet $s \in S$ non coloré **Faire**
    Colorer s en utilisant la plus petite couleur possible

La ligne « Colorer s en utilisant la plus petite couleur possible » consiste à parcourir tous les sommets colorés adjacents à s et à numéroter s en utilisant le plus petit numéro n'apparaissant pas parmi les couleurs des voisins de s. On attribue toujours la couleur 0 au premier sommet traité.

Question 26 Écrire une fonction `int plus_petit_absent(liste* lst)` qui prend en argument une liste `lst` d'entiers et renvoie le plus petit entier naturel n'apparaissant pas dans `lst`. Cette fonction devra avoir une complexité linéaire en la taille de `lst` dans tous les cas et on demande de justifier cette complexité.

Question 27 Écrire une fonction `liste* couleurs_voisins(liste** G, int* c, int s)` qui prend en argument un graphe `G`, une numérotation `c` et un sommet $s \in S$ et renvoie la liste chaînée des couleurs des voisins de s.

L'algorithme glouton le plus élémentaire consiste à parcourir les sommets dans l'ordre croissant des indices.

Question 28 Déterminer sans justifier la coloration renvoyée par cet algorithme sur le graphe G_1 de la figure 2.

Question 29 Écrire une fonction `int* colo_glouton(liste** G, int n)` qui prend en argument un graphe `G` d'ordre `n` et renvoie une coloration de G en suivant l'algorithme glouton.