Rapport AC 2017 : Motifs interdits

A.Lanoix, J.Luc, Q.Sonrel

16 Janvier 2016

Prise en main du problème

Question 1

Les assignations qui satisfont C sont :

- $x1 \to V, x2 \to V$
- $x1 \to V, x2 \to B$
- $x1 \to B, x2 \to V$
- $x1 \to B, x2 \to R$

Question 2

Supposons que C soit satisfiable donc qu'il existe une assignation ϕ telle que ϕ satisfait C.

Donc, selon ϕ, $x1 \to a$ tel que $a \in \{R, V, B\}$ et selon les contraintes

[(x1, R),(x2, R)] ,

[(x1, V),(x2, V)] ,

[(x1, B),(x2, B)] ,

[(x1, R),(x3, R)] ,

[(x1, V),(x3, V)] ,

[(x1, B),(x3, B)] ,

[(x1, R),(x4, R)] ,

[(x1, V),(x4, V)] ,

[(x1, B),(x4, B)],

$x2$, $x3$ et $x4 \not\to a$.

Donc selon ϕ, $x2 \to b$ tel que $b \neq a$ et $b \in \{R, V, B\}$ et selon les contraintes

[(x2, R),(x3, R)] ,

[(x2, V),(x3, V)] ,

[(x2, B),(x3, B)] ,

[(x2, R),(x4, R)] ,

[(x2, V),(x4, V)] ,

[(x2, B),(x4, B)] ,

$x3$ et $x4 \not\to b$.

Donc selon ϕ, $x3 \to c$ tel que $c \neq a$ et $c \neq b$ et $c \in \{R, V, B\}$ et selon les contraintes

[(x3, R),(x4, R)] ,

[(x3, V),(x4, V)] ,

[(x3, B),(x4, B)],

$x4 \not\to c$.

Donc $x4 \to d$ tel que $d \neq c$ et $d \neq b$ et $d \neq a$ et $d \in \{R, V, B\}$.
Or, contradiction : parmi $\{R, V, B\}$ il est impossible de trouver a, b, c et d tels qu'ils soient tous différents.
Donc C n'est pas satisfiable.

Algorithme randomisé

Question 3

Si on a deux contraintes [(x, a)] et [(x, b)], et $a \neq b$, alors on peut supprimer les contraintes de la forme [(x, a), (y, ·)] ou [(x, b), (y, ·)] car ces contraintes sont ainsi satisfaites quel que soit (y, ·) vu que x sera nécessairement assigné à c, et transformer les contraintes [(x, c), (y, ·)] en [(y, ·)].

Question 4

- Eliminer toutes les contraintes de la forme [(x, R),(y, ·)]

  La contrainte [(x, R)] nous garantit que x ne sera pas assigné à R - toutes les contraintes de la forme [(x, R),(y, ·)] seront du coup automatiquement satisfaites.

- Pour tout couple de contraintes [(x, V),(y, b)] et [(x, B),(z, c)], ajouter la contrainte [(y, b),(z, c)]

  La contrainte [(x, R)] nous garantit que x sera soit assigné à B soit assigné à V, on aura donc forcément pour un couple de contraintes [(x, V), (y, b)] et [(x, B),(z, c)] :

  [(x, V), (y, b)] satisfaite si x est assigné à B;

  [(x, B), (z, c)] satisfaite si x est assigné à V;

  Ensuite, selon la valeur qu'on aura assignée à x, on aura besoin de valider soit (y, b), soit (z, c), on peut donc ajouter la contrainte [(y, b),(z, c)] sans changer la valeur du problème initial.

Question 5

- Si C n'est pas satisfiable alors C' n'est pas satisfiable

  Si C n'est pas satisfiable aucune assignation ne satisfait C.
  Or $C' = C \cup \{c1, c2\}$ avec $c1$ et $c2$ étant deux contraintes unaires. Donc pour que C' soit satisfiable alors C et $\{c1, c2\}$ doivent être satisfiables (avec la même assignation).
  C n'est pas satisfiable donc C' n'est pas satisfiable non plus.

- Si C est satisfiable alors C' est satisfiable avec probabilité au moins $1/2$

  Soit $c0$ la première contrainte de C qui est donc de la forme : [(x, a), (y, b)] avec $a, b, c \in \{R, V, B\}$ et $a \neq b \neq c$

  On peut alors :

  - ajouter [(x, a)], [(y, a)] (1)
  - ajouter [(x, a)], [(y, c)] (2)
  - ajouter [(x, c)], [(y, b)] (3)
  - ajouter [(x, b)], [(y, b)] (4)

  Les assignations satisfaisant $c0$ sont :

  $x \to a$ et $y \to a$
  $x \to a$ et $y \to c$
  $x \to b$ et $y \to a$
  $x \to b$ et $y \to b$
  $x \to b$ et $y \to c$
  $x \to c$ et $y \to a$
  $x \to c$ et $y \to b$
  $x \to c$ et $y \to c$

  Il y a alors $4/8$ assignations qui satisfont $c0$ et (1) :

  $x \to b$ et $y \to b$
  $x \to b$ et $y \to c$
  $x \to c$ et $y \to b$
  $x \to c$ et $y \to c$

  Il y a alors $4/8$ assignations qui satisfont $c0$ et (2) :

  $x \to b$ et $y \to a$
  $x \to b$ et $y \to b$

$x \to c$ et $y \to a$

$x \to c$ et $y \to b$

Il y a alors 4/8 assignations qui satisfont $c0$ et (3) :

$x \to a$ et $y \to a$

$x \to a$ et $y \to c$

$x \to b$ et $y \to a$

$x \to b$ et $y \to c$

Il y a alors 4/8 assignations qui satisfont $c0$ et (4) :

$x \to a$ et $y \to a$

$x \to a$ et $y \to c$

$x \to c$ et $y \to a$

$x \to c$ et $y \to c$

On voit alors que, s'il n'existe qu'une seule assignation A possible pour C, alors il y a toujours deux possibilités d'ajout de C' qui sont satisfaites par A (chaque possibilité a une assignation en commun avec les trois autres).

S'il existe plusieurs assignations possibles pour C, alors il y a plus de deux possibilités d'ajout de C' qui sont satisfaites par une de ces assignations.

Donc, comme il y a toujours au moins deux possibilités d'ajout pour C' qui sont satisfaites par l'assignation satisfaisant C, on peut dire que si C est satisfiable alors C' est satisfiable avec probabilité au moins $1/2$.

Question 6

Utilisation du programme :
`python3 algo_random.py [-r nb_repetitions] [-b] nom_fichier`
`-r nb_repetitions` : Nombre de répétitions de l'algorithme qu'on veut effectuer
`-b` : Active le mode brief (désactive l'affichage de la progression pour plus de rapidité)
`nom_fichier` : Fichier contenant la matrice d'adjacence d'un graphe (fournis dans le dossier `graphes`)

- Graphe 1 : non coloriable
- Graphe 2 : coloriable à 100%
- Graphe 3 : coloriable à 0%

- Graphe 4 : coloriable à 95.37%
- Graphe 5 : coloriable à 87.86%
- Graphe 6 : non coloriable
- Graphe 7 : non coloriable
- Graphe 8 : non coloriable
- Graphe 9 : coloriable à 100%
- Graphe 10 : non coloriable
- Graphe bonus 1 : non coloriable
- Graphe bonus 2 : coloriable à 100%

Motifs interdits pondéres

NP-complétude

Question 7

Problème de décision $W - SSS$: Est-ce qu'il existe une assignation des variables dont le poids cumulé des contraintes satisfaites est supérieur à k pour un ensemble de contraintes pondéré donné ?

Question 8

Rappel du problème $(3,2) - SSS$: Instance : Un ensemble de contraintes C (= une liste de couples (variable, couleur))
Problème : Existe-t'il une assignation satisfaisant l'ensemble de contraintes, c'est-à-dire : il existe une variable qui prend une valeur différente dans l'assignation et dans la contrainte.

Pour $3 - COLOR$:
Rappel du problème $3 - COLOR$:
Instance : Un graphe $G = (V, E)$
Problème : Existe-t'il une façon de colorier les sommets avec 3 couleurs de telle sorte qu'aucune arête n'aie les extrémités de la même couleur ?

$3 - COLOR \Rightarrow (3,2) - SSS$
Chaque sommet du graphe V correspond à une variable de $(3,2) - SSS$. Soit la variable Xi correspondant au sommet Si. Pour chaque arête reliant deux sommets (nommons les $S1$ et $S2$) dans E, on ajoute les contraintes suivantes :

[(X1, R), (X2, R)],

[(X1, V), (X2, V)],

[(X1, B), (X2, B)]

Question 9

Dans la suite, on appelle n le nombre de variables du problème.
Preuve que $(3,2) - SSS$ appartient à NP :

- Certificat : une assignation qui correspond à une fonction associant chaque variable à une couleur donc :
  Taille certificat = $nlog3$ (n * taille couleur)
- Instance : liste de contraintes binaires ou unaires = tableau de booléens comme dans l'implémentation de la question 6 (contraintes[n][n][3][3], pour les contraintes unaires, on peut les mettre dans le même tableau)
  Taille instance = $poly(9n^2)$
- Algorithme de vérification :

```
Vérifier $(3,2)-SSS$ : données : certificat [0..n-1] (entier (0 ou 1 ou 2))
  certificatOk = true
  pour i de 0 à n faire
    pour j de 0 à n faire
      pour ci de 0 à 2 faire
        pour cj de 0 à 2 faire
          si contraintes[i][j][ci][cj] = true ET certificat[i] == ci
                           ET certificat[j] == cj
            certificatOk = false
  retourner certificatOk
```

(On peut optimiser en arrêtant le parcours quand certificatOk est faux.)
Complexité de l'algo de vérification : $9n^2$ donc polynomial en la taille de l'instance.
Donc $(3,2) - SSS$ appartient à NP.
D'après la question 8), il existe une réduction polynômiale entre $3 - COLOR$ et $(3,2) - SSS$.
Or, $3 - COLOR$ est NP-difficile donc $(3,2) - SSS$ l'est également.
Donc, comme $(3,2) - SSS$ appartient à NP et est NP-difficile, on peut déduire que $(3,2) - SSS$ est NP-complet.

Algorithme glouton

Question 10

- On choisit de trier les contraintes par ordre décroissant de poids divisé par le nombre de couples de la contrainte. Cela nous permet de prendre en

premier les contraintes qui ont le meilleur rapport poids/"contraignance", autrement dit les contraintes qui ont le plus de poids et contraignent le moins de variables, donc probablement les plus intéressantes.

- Algorithme glouton :

```
Wtotal = 0
EnsAssignations = toutes les assignations possibles
                  (différentes en fonction du nombre de variables)
Pour chaque contrainte C de la liste de contraintes :
  EnsAssignationsTemp = EnsAssignations - l'ensemble
                  d'assignations qui ne satisfont pas la contrainte C
  si EnsAssignationsTemp est non vide
    EnsAssignations = EnAssignationsTemp
    WTotal = WTotal + Cw
  fsi
fpour
retourner WTotal
```

Question 11

Cet algorithme n'est pas optimal car pour l'ensemble de contraintes suivant :

(2, ([X, R], [Y, B]))

(2, ([X, V], [Y, B]))

(2, ([X, B], [Y, B]))

(3, ([Y, B]))

(4, ([Y, V]))

(4, ([Y, R]))

Si on assigne $Y \to V$, alors la somme des poids vaut $2 + 2 + 2 + 3 + 4 = 13$.
Or, avec cette liste de contraintes, l'algorithme va considérer les deux variables de poids 4 et va donc assigner $Y \to B$. De ce fait, il ne va pas prendre la contrainte de poids 3 et ne pourra prendre que deux des contraintes de poids 2 (puisque X va "forcer" une contrainte a ne pas être satisfaite).
L'algorithme va donc nous renvoyer $4 + 4 + 2 + 2 = 12$, ce qui ne peut pas être la solution optimale puisqu'on a trouvé une solution de poids 13.
L'algorithme n'est donc pas optimal.

Question 12

Les ensembles de contraintes de test utilisés sont en commentaire dans la fonction principale du code python.
Utilisation de l'algorithme : `python3 glouton.py`