Chloé DESDOUITS
Guillerme DUVILLIE
Swan ROCHER

M1 Informatique

Projet Transversal - Coloration de Graphes

2012

UM2

Table des matières

Chapitre 1

Introduction

1.1 Remarques

1.1.1 Connexité

Dans la suite, tous les graphes seront supposés connexes, sauf si l'inverse est précisé. En effet, dans le cas où ce ne serait pas le cas, les preuves / algorithmes peuvent s'appliquer à chaque sous graphe correspondant aux composantes.

Chapitre 2

Propriétés Générales

2.1 Exercice 1 - Coloration des Sommets

Considérons H un sous-graphe de G, il paraît évident qu'une coloration de G est aussi une coloration de H. De plus, par le théroème de Brooks, si le degré maximal du sous graphe H est inférieur à $\chi(G)$, alors $\chi(H) \leq \Delta \leq \chi(G)$.

D'où, la propriété suivante :

Propriété 2.1.1 *Si H est un sous-graphe de G alors $\chi(H) \leq \chi(G)$.*

2.2 Exercice 2 - Coloration des Sommets

Chaque composante connexe de G peut être vue comme un sous-graphe de ce dernier. Donc d'après la propriété 2.1.1, on a : $\chi(G) \geq \max\{\chi(C), C \text{ composante connexe de } G\}$. Appelons $C_1, C_2, C_3, \ldots, C_k$ les composantes connexes de G. Pour $1 \leq i \leq k$, appelons c_i la coloration de la composante C_i avec les couleurs $1, 2, \ldots, \chi(C_i)$ et c définie pour tout v sommet de G par $c(v) = c_i(v)$ si $v \in C_i$, dans la mesure où il n'existe aucune arête entre deux composantes connexes de G, c est une coloration de G, ce qui implique $\chi(G) \leq \max\{\chi(C_i)\}$. On en déduit donc la propriété suivante :

Propriété 2.2.1 $\chi(G) = \max\{\chi(C), C \textit{ composante connexe de } G\}$

2.3 Exercice 3 - Coloration des Sommets

Deux sommets de même couleur ne sont reliés entre eux par aucune arête [1]. Ainsi tous les nœuds de la même couleur n'ont aucune arête entre eux et forment donc un stable. Ainsi, un graphe $k-$coloriable peut être divisé en k sous-ensembles des sommets formant des stables et donc on en déduit la propriété suivante :

Propriété 2.3.1 *Rechercher un $k-$stable est équivalent à montrer l'existence d'un graphe $k-$coloriable.*

1. Par définition de la coloration

2.4 Exercice 4 - Coloration des Sommets

Montrons que : $G = (V, E)$ est biparti $\leftrightarrow$ G est 2-colorable.
Un graphe biparti est défini de la manière suivante :

- $V = V_1 \cup V_2$ avec $V_1 \cap V_2 = \emptyset$,
- $\forall x_1, y_1 \in V1 (x_1, y_1) \notin E$,
- $\forall x_2, y_2 \in V2 (x_2, y_2) \notin E$.

$\rightarrow$ Soit $G = (V = V_1 \cup V_2, E)$.

Soit la coloration C suivante :

- $C(x) = 1 \forall x \in V_1$,
- $C(x) = 2 \forall x \in V_2$.

Par définition d'un graphe biparti, cette 2-coloration est bien valide (puisque il n'existe aucune arête à l'intérieur de chaque partition).

$\leftarrow$ Soit $G = (V, E)$ un graphe 2-colorable.
Soit C une 2-coloration valide de ses sommets.
On pose $V_1 = \{x \in V : C(x) = 1\}$ et $V_2 = \{x \in V : C(x) = 2\}$. Puisque cette coloration est valide $\forall x, y \in V$ si $C(x) = C(y)$ alors $(x, y) \notin E$.
Ainsi on peut noter $V = V_1 \cup V_2$, leur intersection est bien vide (puisque un même sommet ne peut pas avoir deux couleurs différentes) et il n'existe aucune arête ayant ses deux extrémités dans le même sous-ensemble de sommets.
G est donc un graphe biparti.

2.5 Exercice 5 - Coloration des Sommets

Soit A le plus grand stable d'un graphe quelconque G à n sommets. On note $\alpha(G) = |A|$.
Montrons que $\lceil \frac{n}{\alpha(G)} \rceil \leq \chi(G) \leq n - \alpha(G) + 1$. On pose (i) la première inégalité et (ii) la seconde.

Montrons (i) par récurrence sur la taille $\alpha(G)$ de A.
Si $\alpha(G) = 1$, on a : $\lceil \frac{n}{1} \rceil = n \leq \chi(G) \leq n$.
On suppose que (i) est vraie pour $\alpha(G) = c$ et on cherche à prouver que c'est toujours le cas pour $\alpha(G) = c+1$. Il est évident que $\lceil \frac{n}{c+1} \rceil \leq \lceil \frac{n}{c} \rceil$, or par hypothèse on a : $\lceil \frac{n}{c} \rceil \leq \chi(G)$.
(i) est donc vérifiée.

Montrons maintenant (ii) :
On sait que $\chi(G) \leq n$ (il suffit d'attribuer une couleur différente à chaque sommet).
Soit $G' = G \setminus A$, on voit que le nombre de sommets de G' est égal à $n - \alpha(G)$, et on en déduit que $\chi(G') \leq n - \alpha(G)$. A étant stable, on a $\chi(A) = 1$.
De plus, $\chi(G) \leq \chi(G') + \chi(A)$, et donc on peut conclure :
$\chi(G) \leq n - \alpha(G) + 1$.

2.6 Exercice 6 - Coloration des Sommets

On cherche à montrer que le graphe complet à n sommets K_n est n-colorable.
Soit la coloration C suivante : $C(x_i) = i \ \forall i \in [1, n]$.
Puisque chaque sommet a une couleur différente, C est valide, de plus puisque K_n possède n sommets, n couleurs ont été utilisées.
K_n est bien n-colorable.

Chapitre 3

Complexité

3.1 Exercice 11 - Coloration des sommets d'un graphe et coloration des arêtes d'un graphe

Un graphe est $2-$coloriable si et seulement si il est possible de partitionner ce dernier en deux ensembles stables. Un graphe 2-coloriable est donc un graphe biparti, or le problème de décider si un graphe est biparti est un problème polynomial, le problème de la $2-$coloration est donc un problème polynomial.

Un graphe est $2-$arête$-$coloriable s'il l'ensemble V de ses arêtes est divisible en deux ensembles F et E distincts vérifiant :

1. $E \cup F = V$
2. $E \cap F = \emptyset$
3. $\forall((i,j),(k,l)) \in E^2, \quad i \neq k, \neq l, j \neq k, j \neq l$
4. $\forall((i,j),(k,l)) \in J^2, \quad i \neq k, \neq l, j \neq k, j \neq l$

E et F sont alors des couplages de G. Un algorithme résolvant ce problème consiste en la recherche d'un couplage maximum C puis en vérifiant que V C est aussi un couplage. Si tel est le cas, le graphe est $2-$arête$-$coloriable.

La recherche d'un couplage se faisant en temps polynomial, l'algorithme énoncé ci-dessus est polynomial.

3.2 Exercice 12 - Coloration des sommets d'un graphe

3.2.1 Appartenance à NP

1. COLOR :
   Considérons une solution, un moyen de tester s'il s'agit d'une solution réalisable est de tester pour chacun des sommets si ses voisins sont d'une couleur différente en gardant le s couleurs rencontrées en mémoire afin de vérifier que leur nombre n'excède pas k. La vérification se fait alors en $O(n^2)$ et est donc polynomiale.

2. 3-COLOR :
   On procède de la même manière que pour COLOR en s'assurant que le nombre de couleur ne dépasse pas 3, la vérification est alors aussi polynomiale
3. 3-COLOR-PLAN :
   La vérification de la validité de la 3−coloration se fait comme précédemment en $O(n^2)$. Il fait cependant tester si le graphe est planaire qui est un problème polynomial, la vérification est don cpolynomiale.

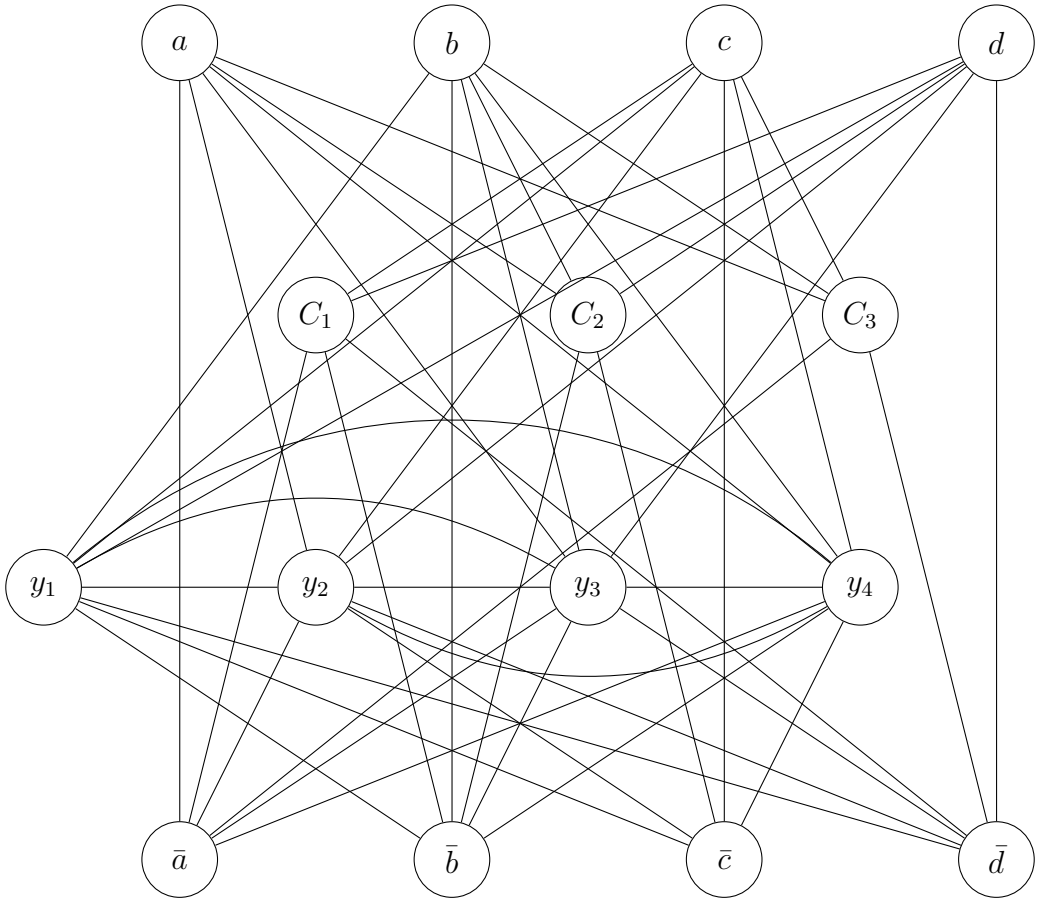
Tous ces problèmes appartiennent donc à NP.

3.2.2 Relation enre 3-COLOR et COLOR

Supposons qu'il existe un algorithme résolvant COLOR en temps polynomial quelque soit k, il existe alors un algorithme polynomial résolvant 3-COLOR. Par contraposition, on vient de démontrer que si 3-COLOR est NP-difficile alors COLOR est NP-difficile.

3.2.3 Étude du problème COLOR

Graphe G_ϕ



Chapitre 4

Heuristiques

4.1 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
1 Soit $x_1, ..., x_n$ une numérotation des sommets de G.
2 Soit $C = \{1, 2, ..., k\}$ un ensemble de couleurs.
3 **for all** x_i **do**
4 $\quad \chi(G) = min\{k \in C : \forall y \in voisinage(x_i), \chi(y) \neq k\}$
5 **end for**
6 **return** $\chi(G)$

4.1.1 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 : $\chi(G) = 3$.

4.1.2 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, soit le graphe biparti $G = (V, E)$ suivant :

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

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 $\chi_{opt}(G_{fig3}) = 2$.

4.2 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

```
1  C = ∅
2  k = 0
3  d_1 >= d_2 >= ... >= d_n les degrés des sommets
4  repeat
5      k = k + 1
6      colorier le sommet x ayant le plus haut degré avec k
7      marquer les voisins de x
8      C = C ∪ {x}
9      while ∃ un sommet y non marqué et non coloré do
10         colorier y ayant le plus haut degré avec k
11         marquer les voisins de y
12     end while
13     effacer toutes les marques
14 until tous les sommets sont colorés
15 return k
```

4.2.1 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 : $\chi(G) = 3$.

4.2.2 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) :

Chapitre 5

Approximations

5.1 Exercice 16 - Coloration des Sommets d'un Graphe

5.1.1 Question 1

Nous devons montrer que l'algorithme 1 n'est pas r-approché quel que soit $r \in \mathbb{N}$ (celui-ci est détaillé en 4.1).

On cherche donc que $\forall r \in \mathbb{N}$, $\exists G$ tel que $\chi_{algo}(G) > r\chi_{opt}(G)$.
Soit le graphe biparti G défini dans la question 4.1.2 :
- $V = \{1, 2, ..., n\}$;
- $E = \{(i, j) \forall i, j \in V : i\ impair,\ j\ pair,\ et\ j \neq i + 1\}$

Etant biparti, $\chi_{opt}(G) = 2$, et d'après 4.1.2 $\chi_{algo}(G) = \frac{n}{2}$.
Donc $\chi_{algo}(G) = \frac{n}{4} \times \chi_{opt}(G)$.

Pour chaque $r \in \mathbb{N}$ créons le graphe ci-dessus de taille $4 \times r + 4$.
$\chi_{algo}(G) = \frac{n}{4} \times \chi_{opt}(G) = (r + 1) \times \chi_{opt}(G) > r \times \chi_{opt}(G)$.
Ainsi pour chaque $r \in \mathbb{N}$ il existe un graphe qui pour lequel l'algorithme ne renvoie pas une r-approximation.

5.1.2 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 $couleur(y_i) \leq couleur(y_{i+1})$.
Il est évident qu'une telle numérotation existe et qu'en suivant celle-ci l'algorithme renverra toujours la solution optimale.

5.1.3 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 (preuve en **??**).

5.1.4 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 $couleur(y) \leq 3$.
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 2.4).
G_x est donc un graphe biparti.

5.1.5 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 $\forall x \in G$

```
1  trier les x_i ∈ G par ordre décroissant de degré
2  k ← 1
3  couleur ← ∅
4  repeat
5      x ← sommet(G)
6      G' ← voisinage(x)
7      couleur ← couleur ∪ 2coloration(G', k)
8      k ← k + 2
9      G ← G \ G'
10 until deg(sommet(G)) < √n
11 couleur ← couleur ∪ DMaxPlus1coloration(G, k)
12 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 (5.1.3).
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 (5.1.4).
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.

5.1.6 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

```
1 for all sommet x ∈ G do
2     G_x ← voisinage(x)
3     if G_x n'est pas 2-colorable then
4         return faux
5     end if
6 end for
7 return vrai
```

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

5.1.7 Question 7

Chapitre 6

Contraintes

6.1 Exercice 19 - Mélangeons les reines et les sudokus

6.1.1 Modèles

Pour le modèle orienté case de l'échiquier à remplir, on pose l'ensemble des $X_{ij} \in \{0,1\}, \forall i,j \in \{1,\ldots,n\}$. Ces variables booléennes X_{ij} représentent la présence ou non d'une reine sur une case ij. Les contraintes sont les suivantes :

$$Lignes : \forall i \in \{1,\ldots,n\}, \sum_{j=1}^{n} X_{ij} = 1$$

$$Colonnes : \forall j \in \{1,\ldots,n\}, \sum_{i=1}^{n} X_{ij} = 1$$

$$Diagonales : \forall i,j,k,l : X_{ij} = 1 \wedge X_{kl} = 1, |i-k| \neq |j-l|$$

Pour le modèle orienté position de la reine sur une colonne, on pose l'ensemble des $X_i \in \{1,\ldots,n\}, \forall i \in \{1,\ldots,n\}$. Ces variables X_i représentent la position d'une reine sur la colonne i. Les contraintes sont les suivantes :

$$Lignes : \forall i,j : 1 \leq i < j \leq n, X_i \neq X_j$$

$$Diagonales : \forall i,j : 1 \leq i < j \leq n, |X_i - X_j| \neq |j-i|$$

6.1.2 Dual du modèle position sur la colonne

Le dual de ce modèle est celui de la position des reines sur les lignes. On pose l'ensemble des $Y_i \in \{1,\ldots,n\}, \forall i \in \{1,\ldots,n\}$. Ces variables Y_i représentent la position d'une reine sur la ligne i. Les contraintes sont les suivantes :

$$Colonnes : \forall i,j : 1 \leq i < j \leq n, Y_i \neq Y_j$$

$$Diagonales : \forall i,j : 1 \leq i < j \leq n, |Y_i - Y_j| \neq |j-i|$$

On remarque que les deux modèles sont strictement équivalents grâce à la relation suivante : $X_i = j \iff Y_j = i$.

6.1.3 Stratégie de recherche

Comme ce problème génère un grand espace de recherche, il est nécessaire de chercher à en explorer le moins possible (dans le cas de recherche d'une seule solution). Afin de réduire le facteur de branchement, on va chercher à affecter en priorité les variables dont le domaine est le plus réduit par les contraintes posées précédemment.

D'autre part, on va également chercher à affecter d'abord les variables qui génèrent le plus de contraintes car ce sont celles qui réduiront le plus les possibilités restantes.

6.1.4 Modèle pour la coloration des reines

Les modèles proposés précédemment permettent de résoudre ce problème avec une seule couleur, il faut donc ajouter une dimension à notre modèle : la dimension des couleurs. Reprenons donc le modèle orienté case de l'échiquier à remplir avec des variables entières cette fois et les contraintes suivantes :

$$Lignes : \forall i, j, k \in \{1, \ldots, n\} : j \neq k, X_{ij} \neq X_{ik}$$

$$Colonnes : \forall i, j, k \in \{1, \ldots, n\} : i \neq j, X_{ik} \neq X_{jk}$$

$$Diagonales : \forall i, j, k, l : |i - k| = |j - l|, X_{ij} \neq X_{kl}$$

Notons que chaque X_{ij} est affectée à la plus petite valeur possible de façon à minimiser le nombre de couleurs utilisées.

6.1.5 Solutions pour $n = 5$

Voici la solution donnée par l'énoncé :

0	1	2	3	4
3	4	0	1	2
1	2	3	4	0
4	0	1	2	3
2	3	4	0	1

Il existe d'autres solutions comme celle-ci par exemple :

0	2	4	1	3
1	3	0	2	4
2	4	1	3	0
3	0	2	4	1
4	1	3	0	2

6.1.6 Solution pour $n > 5$

Pour $n = 6$, il n'y a pas de solution. Pour $n = 7$ en voici une :

0	2	4	6	1	3	5
1	3	5	0	2	4	6
2	4	6	1	3	5	0
3	5	0	2	4	6	1
4	6	1	3	5	0	2
5	0	2	4	6	1	3
6	1	3	5	0	2	4

6.1.7 Conclusion

Nous remarquons que ces solutions répètent les mêmes motifs d'une ligne sur l'autre et que par conséquent, elles doivent être étendables à des grilles plus grandes.

Chapitre 7

Combinatoire des mots

7.1 Exercice 20 - Coloriage non répétitif

7.1.1 $\pi(P_n)$

Soit $\pi(P_n)$ le nombre de Thue du graphe P_n c-à-d le plus petit nombre de couleurs nécessaires pour colorier la chaîne à n sommets de manière non-répétitive.

Voici les graphes P_2, P_3 et P_4 ainsi que leur nombre de Thue.

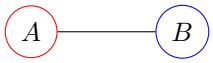


FIGURE 7.1 – $\pi(P_2) = 2$

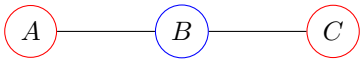


FIGURE 7.2 – $\pi(P_3) = 2$

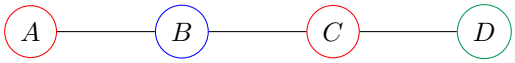


FIGURE 7.3 – $\pi(P_4) = 3$

Soit $H(n)$ l'hypothèse suivante : $\pi(P_n) \geq 3$. Montrons par récurrence que $H(n)$ est vraie pour tout $n \geq 4$.

D'après la figure du graphe P_4 ci-dessus, on a $H(4)$ vraie. On peut donc initialiser la récurrence à $n = 4$.

Supposons que $H(n)$ est vraie et penchons-nous sur le graphe P_{n+1}. Le graphe P_{n+1} étant la chaîne à $n+1$ sommets, on peut colorier une sous-chaîne de P_n à n sommets avec 3 couleurs ou plus de manière non-répétitive d'après l'hypothèse de récurrence. En coloriant le sommet restant avec une nouvelle couleur, on obtient donc un coloriage non-répétitif à 4 couleurs ou plus donc on a bien $H(n+1)$ vraie.

On vient de prouver que $H(n) \rightarrow H(n+1)$ donc on a bien $\pi(P_n) \geq 3, \forall n \geq 4$.

7.1.2 Mot ω sans chevauchement

Montrons que pour tout mot fini ω sur $\{a, b\}$, ω est sans chevauchement si et seulement si $\mu(\omega)$ est sans chevauchement.

Tout d'abord, supposons (sans perte de généralité) que $\omega = mno$ avec m, n, o des mots sur $\{a, b\}$ qui peuvent être le mot vide. Supposons que $n = \alpha u \alpha u \alpha$ avec α une lettre et u un mot. On a alors : $\mu(\omega) = \mu(m)\mu(n)\mu(o)$ par définition. Or $\mu(n) = \mu(\alpha)\mu(u)\mu(\alpha)\mu(u)\mu(\alpha)$. Si l'on décompose le mot avec $\mu(\alpha)_i$ la i^{me} lettre du mot $\mu(\alpha)$, on obtient : $\mu(n) = \mu(\alpha)_1 [\mu(\alpha)_2\mu(u)] \mu(\alpha)_1 [\mu(\alpha)_2\mu(u)] \mu(\alpha)_1\mu(\alpha)_2$. $\mu(n)$ privé de sa dernière lettre est donc bien un facteur de la forme $\alpha u \alpha u \alpha$ et on a :

$$\omega \textit{ est avec chevauchement} \rightarrow \mu(\omega) \textit{ est avec chevauchement}$$

et donc :

$$\mu(\omega) \textit{ est sans chevauchement} \rightarrow \omega \textit{ est sans chevauchement}$$

7.1.3 $\mu^n(a)$ sans chevauchement

Montrons par récurrence que les mots $\mu^n(a)$ sont sans chevauchement pour tout entier $n \geq 0$.

On a $\mu^0(a) = a$ qui est sans chevauchement. Donc au rang 0 la propriété est bien vérifiée.

Supposons $\mu^n(a)$ est sans chevauchement.

On a $\mu^{n+1}(a) = \mu(\mu^n(a))$. Or d'après l'hypothèse de récurrence, $\mu^n(a)$ est sans chevauchement et d'après la question précédente, $\mu(\omega)$ est sans chevauchement ssi ω est sans chevauchement. On a donc bien $\mu^{n+1}(a)$ est sans chevauchement si $\mu^n(a)$ est sans chevauchement. cqfd

7.1.4 Mot infini sans chevauchement

D'après la question précédente, les mots $\mu^n(a)$ sont sans chevauchement pour tout entier $n \geq 0$. Donc $\lim_{n\to\infty} \mu^n(a)$ est sans chevauchement. Or ce mot est un mot infini. Il existe donc bien un mot infini sans chevauchement.

7.1.5 Chevauchements et carrés

7.1.6 Réciproque

7.1.7 Mot infini sans carré

Chapitre 8

Représentation des Connaissances

8.1 Exercice 21 - Coloration et Homomorphisme

8.1.1 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 2.6). 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.

8.1.2 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

```
1 K_k ← k − clique
2 if ∃h un homomorphisme de G dans K_k then
3     return vrai
4 else
5     return faux
6 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.

8.2 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 $R_j \in \{\times_{i\in[1,q]} D_{ji}\}$ 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...

Réalisé avec LaTeX