Problème de coloration de graphe

Dans cette démonstration nous tâcherons de démontrer que le nombre chromatique est obtenable en temps polynomial.

1. Annexes

Soit le graphe tels que :

1. Définition du problème du coloration de graphe

Définition :

-Le nombre chromatique est le nombre minimal de couleurs qu'on doit utiliser pour colorier tous les sommets d'un graphe en s'assurant que deux sommets adjacents ne soient pas de la même couleur.

1. Détermination de l’ensemble des nœuds qui sont associés à une couleur.
2. Introduction à l’ensemble

Soit l’ensemble qui est une injection de chaque couleur au travers un des nœuds de G portant cette couleur et .

Soit :

En raisonnant par classe d’équivalence :

Exemples :

Si  :

1. Détermination de l’ensemble I

l’ensemble qui associe à un nœud k, tous les nœuds différents de lui-même dans l’ensemble des nœuds du graphe (V) qui peuvent potentiellement avoir la même couleur en respectant l’ensemble des arrêtes (E).

Soit :

On déduit facilement que :

Sachant que :

Du fait que :

Donc :

Remarque :

Donc :

1. Décomposition en élément simple de l’ensemble maximale.

Dans cette partie on cherchera à démontrer que peut se décomposer sous plusieurs ensembles et que cette décomposition reviendra à un problème coloriage de graphe.

1. Lemme :

La proposition logique P contient n variables et v clauses, chacune composée de deux variables engagées dans une opération logique avec les opérateurs "ou" et "et". P peut être représenté comme un problème de coloriage du graphe G(V,E), où V représente les n sommets correspondant aux variables et E représente les arêtes correspondant aux clauses contenant l'opérateur "ou". Par exemple, si , elle peut être représentée par trois sommets (un pour 1, un pour 2 et un pour 3) et les sommets 1 et 2 ainsi que les sommets 2 et 3 seront reliés par une arête.

Pour résoudre ce problème de logique booléenne, il faut trouver une coloration des sommets de G telle que deux sommets reliés par une arête ne sont de même couleur. Chaque couleur représente une valeur de vérité pour les variables correspondantes. Si deux sommets reliés par une arête ont la même couleur, cela signifie que la clause contenant l'opération "ou" est fausse, car deux variables contradictoires ne peuvent pas être vraies en même temps. Le nombre minimal de couleurs nécessaires pour colorier le graphe de vérité doit être utilisé pour rendre la proposition logique vraie et simplifier l'expression de la proposition P autant que possible.

Haut du formulaire

1. Démonstration

et du lemme précédent on peut en déduire que :

On a le graphe de qui est avec :

De ce fait on introduit l’ensemble des repartions des couleurs minimales dans :

On a alors :

On remarque donc une récurrence qui prendra fin quand l’ensemble maximal de nœuds ne contiendra plus aucune clause avec l’opérateur logique (l’ensemble n’engendrera pas un nouveau sous graphe).

1. Extraction d’un algorithme et calcul de sa complexité

Tous ce qui a été écrit précédemment peut être déterminer réaliser par un algorithme.

J’ai choisi de codé celui-ci en python :

1. def nbh(n,E,w=1):
2. F=[]
3. V=list(range(n))
4. v=n
5. while v>0:
6. M=[]
7. m=0
8. k=0
9. DM=[]
10. for a in V[1:]:
11. if E[V[0]][a-V[0]-1]==0:
12. M.append(a)
13. m+=1
14. a=0
15. for A in M[:m-1]:
16. b=0
17. for B in M[a+1:]:
18. if E[A][B-A-1]==1:
19. DM.append([a,b])
20. k+=1
21. b+=1
22. a+=1
23. if k!=0:
24. M\_=nbh(m, [[1 if [i,j] in DM else 0 for j in range(m-i-1)] for i in range(m)],0)[0][0]
25. M=[M[i] for i in M\_]
26. m=len(M\_)
27. v-=m+1
28. F.append([V[0]]+M)
29. v=w\*v
30. if v==0:
31. return F,len(F)
32. else:
33. V=[x for x in V if x not in [V[0]] + M]
34. Utilisation du programme

En entrer de ce programme il nous faut renseigner :

-: le nombres de nœuds du graphe.

-: l’ensemble des liaisons telle que et si les nœuds a et b sont relié alors tels que

-: w sera toujours égale à 1

En sortie :

L’algorithme nous renverra F qui est l’ensemble de répartition des couleurs au sein de G et len (F) sera par conséquent le nombre chromatique de G.

1. Calcul de la complexité de l’algorithme

Pour le calcul de la complexité de l’algorithme on utilisera la notation de Landau .

Ligne 2 : O(1) (initialisation de la liste F)

Ligne 3 : O(n) (initialisation de la liste V)

Ligne 4 : O(1) (initialisation de la variable v + détermination de la valeur initiales de v)

Ligne 5 : O(n) (boucle sur la valeur de v)

Ligne 6 : O(1) (initialisation de la liste M)

Ligne 7 : O(1) (initialisation de la variable m)

Ligne 8 : O(1) (initialisation de la variable k)

Ligne 9 : O(1) initialisation de la liste DM)

Ligne 10 : O(n) (boucle sur la valeur de V)

Ligne 11 : O(1) (accès à l’élément E[V[0]][a-V[0]-1])

Ligne 12 : O(1) (ajout de l’élément « a » à la liste M)

Ligne 13 : O(1) (incrémentation de la variable m)

Ligne 14 : O(1) (initialisation de a)

Ligne 15 : O() (boucle imbriquée sur les éléments de M)

Ligne 16 : O(1) ( initialisation de la valeur b)

Ligne 17 : O() (boucle imbriquée sur les éléments de M)

Ligne 18 : O(1) (accès à l’élément E[A][B-A-1])

Ligne 19 : O(1) (ajout de la paire [a,b] à la liste DM)

Ligne 20 : O(1) (incrémentation de la variable k)

Ligne 21 : O(1) (incrémentation de la variable b)

Ligne 22 : O(1) (incrémentation de la variable a)

Ligne 23 : O(1) (accès à l’élément k)

Ligne 24 : O() (construction de la matrice d’adjacence à partir de la liste DM) + récursion sur une sous instance de taille m<n. On peut considérer que la complexité est T(m) (appelle récursif de la fonction nombre\_chromatique avec des arguments de taille m)

Ligne 25 : O(n) (construction de la liste M à partir de la liste M\_)

Ligne 26 : O(1) (mise à jour de la variable)

Ligne 27 : O(1) (incrémentation de la variable v)

Ligne 28 : O(1) (ajout de la liste [V[0]]+M à la liste F)

Ligne 29 : O(1) (mise à jour de la variable v)

Ligne 30 : O(1) (accès à la variable v)

Ligne 31 : O(1) (retour de l’information)

Ligne 32-33 : O(n) (construction de la liste V à partir de la liste [V[0]]+M

La complexité du nombre\_chromatique dépend de l’appel récursif à la ligne 24. Si on note T(n) la complexité du nombre\_chromatique avec une entrée de taille n, alors la complexité de l’algorithme est donnée par la relation de récurrence suivante :

Car :

*Ou est la complexité du sous graphe de T(m) et ainsi de suite jusqu’à .*

On a donc bien démontré que le nombre chromatique est obtenable en temps polynomial.

1. Remarque

On peut alors faire un lien avec la question si P=NP, et du fait des résultats en déduire que P=NP.