rapport de projet
Résolution de problème à l'aide de la coloration de graphe
UE Optimisation dans les graphes
Encadré par Mme Dominique Quadri

BA Alhassane
Master 1 ANO

30 octobre 2021

Table des matières

1

Introduction

Le problème traité ici dans ce projet est l'étude et la résolution des problèmes de la vie réelle à partir des algorithmes de graphes. Notre sujet est spécifique aux problèmes dont la résolution est adapté aux algorithmes glouton de coloration dans les graphes.
Il s'agit du problème de gestion de la mémoire des variables des programme informatique à partir des allocations de ressource par des registres. Le but est d'allouer le minimum possible de nombre de registre dont les variables vont partager.

Dans la suite on détaillera la résolution de ce problème par des modélisations formelles , l'algorithme spécifique et son pseudo code.

1.1 Modélisation formelle du problème par un graphe

- **Que representera les sommets et les arêtes du graphe ?**
  Comme le problème sera résolut par des graphes. Composé uniquement de sommets et d'arêtes. La question qui se pose est qu'est ce qui représentera les sommets et les arcs et la signification des sommets reliés.
  Si deux variables ne sont pas utilisées en même il est possible de les allouer dans un même registre.La notion d'usage simultané est schématisé par la figure suivante. Une horizontales représentent le le début et la fin d'activité des variables. C'est ces lignes de temps qui correspondront les **sommets** de notre graphe. Leurs intersections représenteront une **arête** du graphe. Les lettres ce sont les noms des variables.

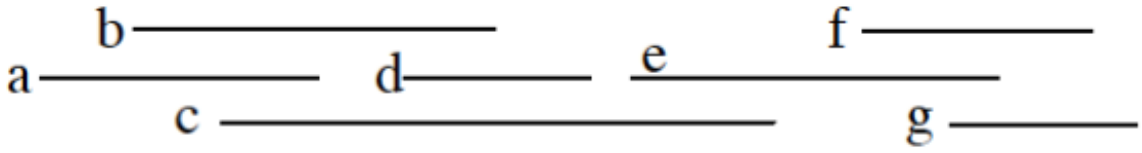

FIGURE 1.1 – Intervales d'activités des variables

Par exeemple on voit sur la figure 1.1 que l'intervalle d'activités de la variables c coïncide avec celle de des variables a,b,d,e. Donc on pourra pas allouer simultanément ces dernières et c dans un même registre.

- **Le graphe associé**
  Le la figure suivante est la représentation de notre graphe. On avait dit que les temps d'intervalles a,b,d,e et celle de c se coïncidaient donc les variables associées vont être relié par une arête. C'est pour cela on une arête entre le sommet C et les autres.

  On a maintenant à travers ce graphe une idée plus claire du problème. On a une vue d'ensemble du problème et on sait qui fait quoi en même temps que les autres.

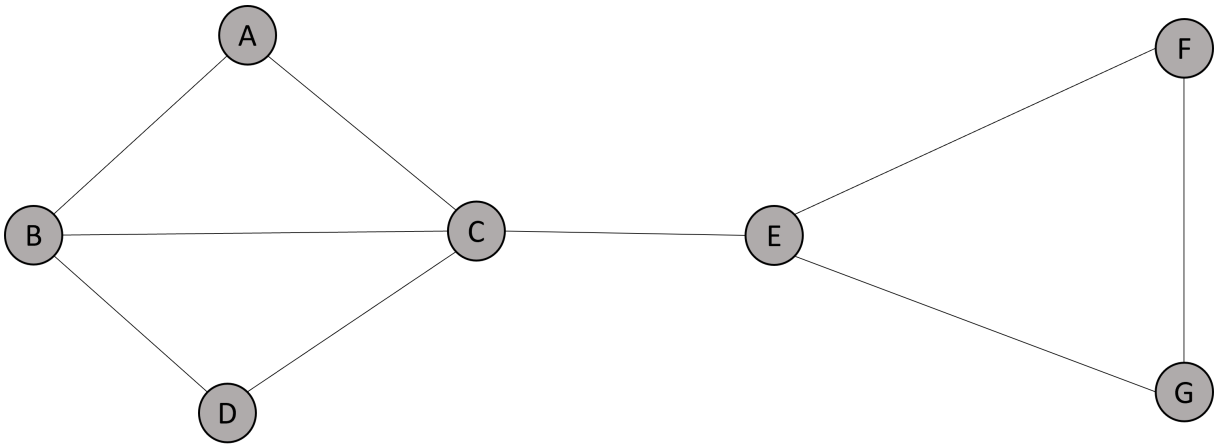

FIGURE 1.2 – La représentation formelle de la figure 1.1

- **Remarque**
  On peut dire que à partir de cette modélisation que un sommet est équivalente à une variable et une arête une est équivalent par une relation synchronisation ou une simultanéité entre les variables.
- **Les couleurs et les nombre de registres minimum.**
  L'objectif du problème est d'allouer le minimum de registres possible pour ces 7 variables. C'est là que va intervenir les couleurs. Ce qui fait ramener au problème de couleur minimale dans les graphes.
  Donc on vas colorer les sommets du graphes avec le nombre de couleur minimale. Le nombre de couleur obtenu sera le nombre de registres minimal nécessaire pour allouer ces variables.

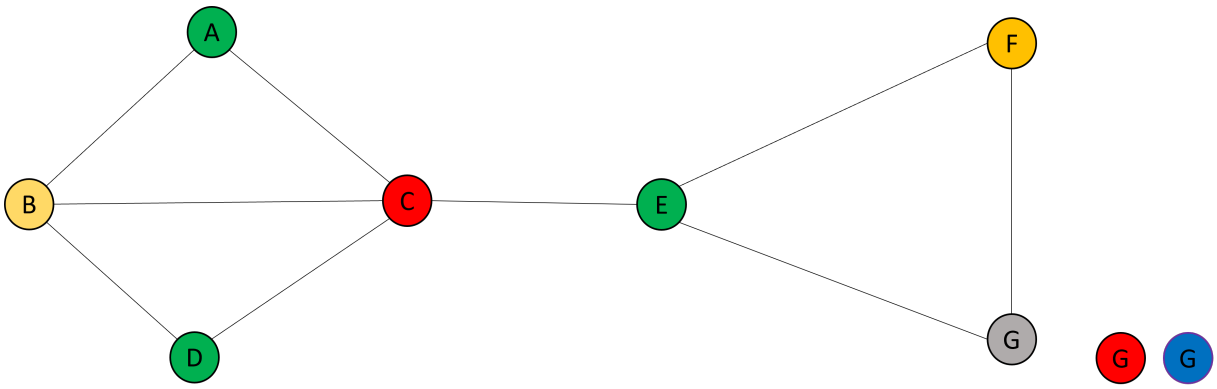

FIGURE 1.3 – Le coloriage du graphe

On a colorié le graphe avec les couleurs ci-dessous. Des sommets de couleurs identiques identiques indique qu'on pourra allouer les variable dans une même registres. Évidemment

on voit que deux sommets adjacent n'on jamais la meême couleurs. Car ce sont des variables qui s'exécutent mème tempps.

Pendant le coloriage on pouvait colorier sommet G avec une nouvelle couleur comme le bleu ou une autre couleur dont il n'est pas adjacent. Mais l'optimisation du problème impose à minimiser le nombre de couleurs utilisé.

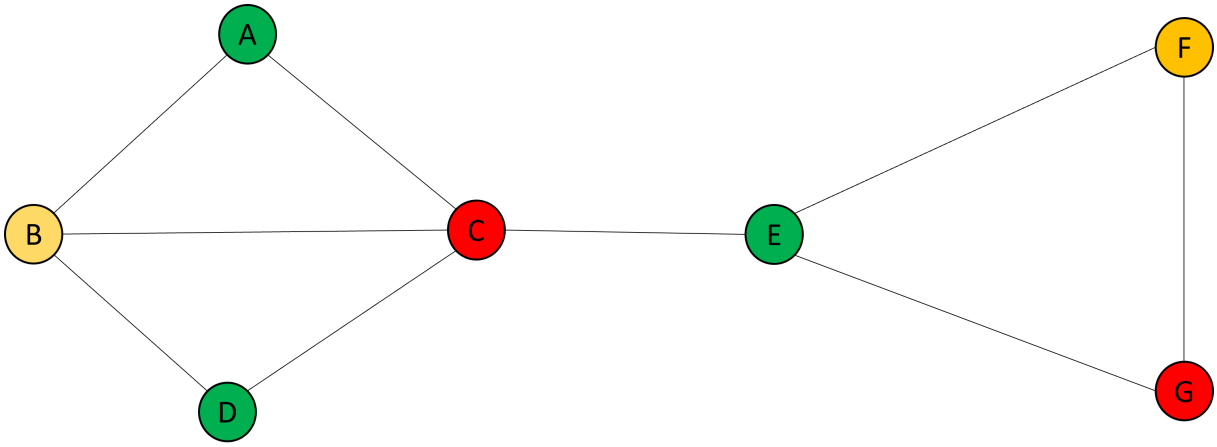

FIGURE 1.4 – Le coloriage optimum avec le minimum de couleurs

2

Algorithme de coloration et Pseudo code

Il existe plusieurs algorithmes différentes qui donnent le nombre de couleurs optimal ou pas optimale. On va présenter dans cette partie le choix de notre algorithme facile pour l'implémentation parmi plusieurs.

2.1 L'algorithme séquencielle : algorithme pas forcément optimale

C'est une méthode facile simple. Mais pas optimale. Il s'agit de détermner un un ordre $O = (x_1, x_2, ..., x_n)$ des sommets du graphe .

- Pour $i := 1$ n faire.
- Donner à x_i la lus petite couleur possible ;
  - ie la plus petite couleur qui n'est pas utilisées par x_j adjacent à x_i
  - Pour tout $j < i$
- Fin Pour

2.2 Algorithme de DSATUR de Belaz : optimal

Soit $G = (X, E)$ Un graphe.
C'est algorithme glouton qui attribue à chaque étape une couleur à un sommet du graphe.

A une étape quelconque, la solution courante consisste à une coloration partielle des sommets du graphe.

On introduira une nouvelle variable de plus pour informer le nombre de couleurs différentes colorié des sommets du voisin. On l'appelle niveau de saturation . Et on la notera σ. Donc Pour un sommet $i \in X$ du graphe on note $\sigma(i)$ son nombre de couleurs différentes dèja attribué à son voisin.

Avant de décrire le pseudo code on va spécifier les structure de données.

- Soit $\delta(i)$ le degré d'un sommet i de G
- Soit $\sigma(i)$ son nombre de couleurs différentes déjà attribué à son voisin.
- Soit $c(i)$ son numéro de couleur.
- Soit $\Gamma(i)$ son nombre de voisin.

L'algorithme est basé sur les observations suivantes :

- Un sommet est d'autant plus difficile à colorier qu'il a beaucoup de voisin. On cherchera donc à colorier en priorité les sommets de degré plus élevés.

- Il est d'autant plus urgent de colorier un sommet i que le nombre de sommets voisins déjà colorié avec des couleurs différentes. Autrement dit son niveau de saturation $\sigma(i)$ est élevé.
- Aini le critère de sélection du sommet à colorier tient compte des deux paramètre $\sigma(i)$ et $\delta(i)$

1. — Pour i = 1, 2,3,..,..,n
   - — $\delta(i) \longleftarrow$ degrés des sommets du graphe.
   - — $\sigma(i) \longleftarrow 0$
   - — $c(i) \longleftarrow 0$
   - — Fin pour
   - — Choisir un sommet $_i0$ de degré maximum dans G et lui attribué la couleur 1 :
   - — $c(i_0) \longleftarrow 1$; $k \longleftarrow 0$
   - — Pour $j \in \Gamma(i_0)$ faire :
     - — $\delta(j) \longleftarrow \delta(j) - 1$
     - — $\sigma(i) \longleftarrow 1$
   - — Fin pour
2. Tant que $k < n$ faire :
   - — Sélectionner un sommet i tel que $c(i) = 0$ maximisant $\sigma(i)$; au cas ou il existe plusieurs sommets de valeurs $\sigma(i)$ maximale , choisir un sommet de de valeur $\delta(i)$ maximale.
   - — calculer :
     - — q = Min $(j/j \in (1, 2, 3, ..., n) \quad tel \quad que \quad j \notin c(\Gamma_i) \quad)$ La plus petite couleur pouvant être attribué à i
     - — $c(i) \longleftarrow q$
     - — $k \longleftarrow k + 1$
     - — Pour $j \in \Gamma_{i_0}$ faire :
       - — $\delta(j) \longleftarrow \delta(j) - 1$
       - — recalculer : $\sigma(i) \longleftarrow |c(\Gamma_j|$
     - — Fin Pour
   - — Fin Tant que
3. Fin Algorithme

Lorsque l'algorithme se termine le contenu du tableau donne la coloration du graphe.

2.3 Le choix de cet algorithme et les difficulté du codage

2.3.1 Le choix de cet algorithme

On voit que cette algorithme sort bien toutes toutes les difficulté du problème de coloriage. Il met aussi en évidence toutes les structures de données adaptées aux problème de coloriage. Il indique une relation d'ordre de priorité sur les choix du sommet à colorier. Il est facile à lire et à comprendre.Et garantit une solution optimale. Mais on verra par la suite même si il est simple à lire et comprendre pour un humain mais il n'est pas facile à traduire en langage machine.

2.3.2 Les difficultés du codage

Cet algorithme codé dans ce projet et plusieurs autres algorithmes du cours d'optimisaton sont codés en C++. Il 'y a eu plusieurs difficultés en terme d'aspect technique et codage et le choix des structure de données. D'une manière générale le C++ fait beaucoup d'effet de bord. Il fait pas parfois ce qu' on croit qu'il va faire.

J'ai implementé cette algorithe avec plus de 3 codes différents. La difficulté réside à se souvnir et à recensér les couleurs des sommets adjacentes dont il reste des sommets non encore coloriés tout en leur attribué un numéro inférieur qui n'est pas dans le voisinage.

3

Résolution à la main du problème par théorie des graphe

Dans cette partie on va essayer de répondre aux questions théoriques relatives aux propriétés de coloration.

1. Tracez le graphe d'intervalles correspondant 'a l'exemple de la Figure : vor figure 1.1.
2. Montrez que minimiser le nombre de registres utilisées correspond à calculer $\chi(G)$ pour le graphe d'intervalles correspondant aux intervalles d'activité des variables du programme.
   Il $\exists$ uen arête entre A et B sont utilisé en même temps. Dons Elles ne peuvent pas être mises dans un même registre.Donc ces deux sommets adjacentes ne peuvent pas avoir la même couleur. On voit ici que le nombre de registre est équivalent aux nombre de couleurs. Et le plus petit nombre de registre revient à chercher la valeur $\chi(G)$, le nombre de couleurs minimum.

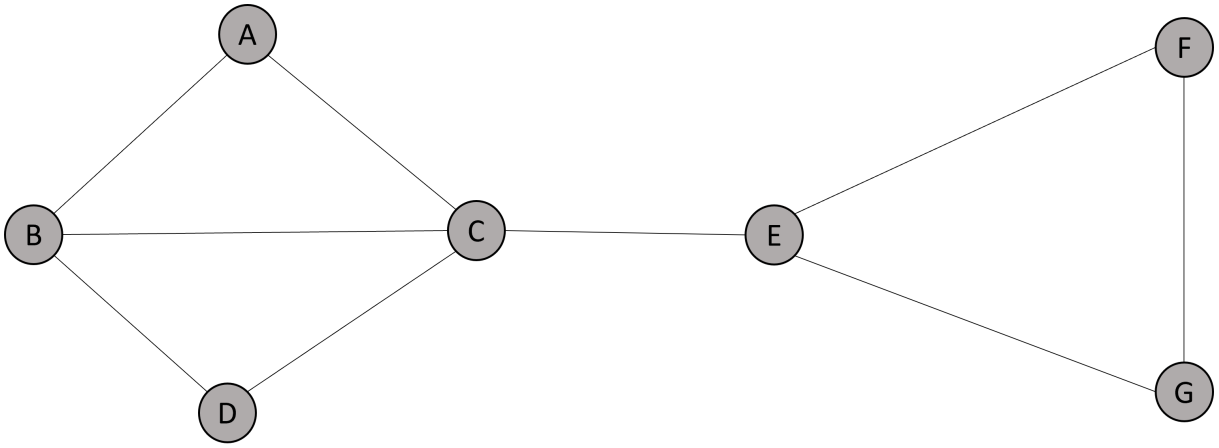

FIGURE 3.1 – La représentation formelle de la figure 1.1

3. Montrez que pour tout graphe G on a $\chi(G) \geq \omega(G)$ Dèja uniquement dans le clique on a forcément autant de couleur que la taille du clique. Vu que la clique est un sous graphe du graphe, Donc le nombre de couleur global est forcément supérieure ou égale à la taille du clique. Ici notre clique C'est E, F, G de taille 3. Donc on aura au moins 3 couleurs.
4. Tri des sommets par ordre de début de temps d'intervalle. a, b, c,d,e f,g
5. Montrez que x a au moins $k-1$ sommets voisins après l'application de l'algorithme séquentiel
   Obtenir K couleurs différentes après coloration $\Rightarrow$ on a quelque part ou on ne peut pas utilisé des couleurs inférieurs et sune nouvelle couleur est nécessaire. C'est les endroits ou

on a le plus grand degré du graphe qui son adjacents à tous les couleurs utilisées jusqu'à là.

6. Soit t la date de début de l'intervalle correspondant au sommet x. Montrez que t appartient k intervalles si k est son numéro de couleur.

   Si le temps de début de x piétine un autre temps d'intervalle alors on peut pas les mettre dans un même registre. Donc de couleurs différentes. Il appartient utant d'intervalle que son numéro couleur.

7. Déduire de la question précédente que $\omega(G) \geq k$.

   Ici si k désigne le nombre d'intervalle donc dans un clique de taille k on est impliqué dans $k-1$ intervalles pour chaque sommet. Donc $\omega(G) \geq k$ est supérieur ou égale au nombre d'intervalle en intersection.

8. Je ne comprend pas bien la suite des questions.

4

Organisation du code du projet

J'ai mis ensemble les codes des algorithme de parcours et le celui de la coloration. J'ai utilisé les classes. Les codes pour la coloration sont dans une classe appelée Coloriage et pour le parcours les codes sont dans une classe appelés Parcours. Il une classe particulière en plus appelé Matrice pour coder le graphe et que toutes les autres classes d'algorithme vont utiliser pour instancier le graphe. Chaque classe a son fichier .h et .cpp.
Dans la classe Parcours il est codé l'algorithme de Dijsktra. Le code exécute correctement.

Dans la classe Coloriage j'ai codé plusieurs méthodes dont deux sont la même choses et code l'algorithme que j'ai cité en amont. Ces codes n s'exécutent correctement. Car je viens de me rendre en compte que j'ai utilisé la classe Matrice pour instancier les graphes non orientés. A la base J'ai construit cette classe Matrice pour les graphes orientés depuis le début.

J'ai essayé de casser cette notion de sens mais la structure de donnée que j'ai utilisé donne toujours des effets de bords. J'ai utilisé un pointeur de tableau 1D pour représenter les matrices.J'ai mis beaucoup de jour à débeuguer ça mais j'ai pas pu. Et finalement je me rend compte que c'est du à ce pointeur. Par exemple Dans la classe Coloriage à la ligne 516 j'ai essayé d'affiché les sommets adjacentes du sommet 0. On vois que dans le terminal il affiche le sommet 11 et même dans l'affiche du graphe on vois que les deux sommets ne sont pas adjecentes. J'ai compris le problème comme j'ai utilisé un tableau 1 D c'est comme si les le tableau se boucle du sommet 1 $\longrightarrow$2$\longrightarrow$ 3$\longrightarrow$4... $\longrightarrow$11 et du sommet 11 $\longrightarrow$. sommet 1.

Ce n'est pas probablement les mêmes raisons que mon code ne s'exécutent pas correctement , il y'a aussi les codes de sélections des sommets qui sont un peux difficiles à coder.

J'ai écrit les code avec l'éditeur CLion. Toutes classes sont contenus dans un documents appelé ParcoursCraphe.

5

Conclusion

Ce projet de modélisation et de résolution de problème à l'aide des graphes est plein d'enseignement en terme d'algorithme, d'optimisation et de codage de programme informatique. Ca nous a permis de distinguer les propriétés utiles du problème et des les adapter à un algorithme spécifique parmi les innombrables algorithmes qui existent.
Il nous a permis aussi de faire une analogie entre les méthodes formelles et leurs implémentations en code.

Ca nous a permis de voir que passer à l'étape de l'implémentation fait apparaître plusieurs concepts cachés dans les algorithmes ou dit en une seule ligne mais dont leurs représentation en langage de programmation demande plusieurs lignes et des difficultés énormes.

L'algorithme de coloriage c'est l'algorithme le plus difficile à mon avis. Car il tient comptes de plusieurs paramètres et se rappelles de plusieurs choses des lignes précédentes et actuelle du programme en même temps. Contrairement aux autres algorithmes de parcours où on se rappelle juste des prédécesseurs. D'après mes recherches je ne voyais même pas des codes qui implémentent cet algorithme correctement.