Heuristique SL pour la coloration de graphes

SIO - Simulation et Optimisation

Nicolas Crausaz & Maxime Scharwath

23.10.2022

Table des matières

Introduction

Ce laboratoire consiste en l'élaboration d'un pseudo-code mettant en œuvre l'heuristique SL (smallest-last), utile pour la coloration de graphes. Cette mise en œuvre devra respecter des complexités temporelles et spatiales linéaires sur un graphe non orienté simple comptant n >= 1 sommets et m >= 0 arêtes. Ces complexités devront donc être de l'ordre de $\mathcal{O}(n + m)$.

Démarche et analyse de complexité

Notre approche se décompose en trois phases distinctes

- L'initialisation des structures nécessaires
- L'établissement d'un ordre de coloration Smallest-Last
- La coloration du graphe selon l'ordre obtenu

Soit un graphe $G = (V, E)$ simple et non orienté, avec $n = |V|, >= 1$ sommets et $m = |E|, <= 0$ arêtes. Nous utiliserons $\Delta(G)$ pour parler du sommet de plus grand degré du graphe G.

L'heuristique SL consiste à retirer le sommet de plus petit degré Vi à chaque itération du sous-graphe induit par $V \setminus \{v_1, \ldots, v_{k-1}\}$, que nous appellerons V'. Pour trouver ce sommet de plus petit degré, il est donc nécessaire d'avoir à disposition une structure triée des sommets du sous-graphe V', selon leur degré au fil de chaque itération. Un tri en temps linéaire n'est cependant pas envisageable car il devrait être effectué à chaque itération, la contrainte complexité temporelle imposée ne serait donc pas respectée, nous serions dans l'ordre de $\mathcal{O}(n^2)$.

Initialisation

Il va nous falloir construire une structure nous permettant de conserver un état des sommets de V' triés selon leur degré. Nous allons créer un tableau de pointeur, de taille $\Delta(G) + 1$ indicé de 0 à $\Delta(G)$. Chaque élément i du tableau est un pointeur vers une liste doublement chaînée, cette liste permet de stocker les sommets de degré i de V'. Nous déplacerons les sommets entre les différentes listes afin de maintenir un ordre trié des sommets selon leur degré sans devoir effectuer un tri complet de la structure.

Nous supposons que l'implémentation des listes doublement chaînées nous permet d'insérer en début et en fin de liste en temps constant et de connaître sa taille en temps constant.

Nous créons un tableau de taille n qui nous permettra de conserver l'ordre de coloration SL retenu pour la coloration après la deuxième étape de l'algorithme. On initialise un entier à 0 nous permettant de compter les insertions dans ce tableau.

Nous initialisons un entier nous permettant de conserver le degré minimum courant, nous lui attribuons la valeur initiale de $\Delta(G) + 1$. Il permet de savoir dans quelle liste de la structure se trouve le sommet de plus petit degré, sans devoir itérer sur le tableau complet et de vérifier si les listes sont vides.

Nous initialisons un tableau de degrés courant, de taille n, indicé de 0 à $n-1$. Ce tableau nous permet de savoir le degré courant de chaque sommet et s'il a déjà été retiré de V'.

Pour initialiser notre structure de sommets de V' triés par degré, nous parcourons le tableau de liste d'adjacence du graphe G, et nous ajoutons chaque sommet S de V dans la liste référencée par l'indice correspondant à $deg(V)$. Cette opération nécessite le parcours du tableau complet du graphe G, ainsi une complexité égale à $|V|$, c'est-à-dire en $\mathcal{O}(n)$. Durant cette construction, nous tenons à jour notre entier représentant le plus petit degré connu.

En termes de mémoire, nous avons initialisé deux tableaux de taille n ainsi que notre structure composée de $\Delta(G)$ listes et $\Delta(G)$ pointeurs. Ces listes pourront contenir chacune au maximum n sommets et ainsi dans la structure contenir $\Delta(G) * n$ sommets et m doublons maximum. Ce cas maximal arriverait si tous les degrés des sommets sont égaux à $\Delta(G)$ (graphe complet). Dans le cas du graphe complet m étant linéaire à n^2, on peut accepter cette complexité comme étant linéaire. Cette complexité est donc de l'ordre de $\mathcal{O}(n^2 + m)$ dans le cas du graphe complet et de $\mathcal{O}(n + m)$ dans le cas général. La complexité spatiale est donc de $\mathcal{O}(2 * n) + \mathcal{O}(n + m)$, ainsi linéaire en $\mathcal{O}(n + m)$

Établissement de l'ordre de coloration SL

Nous allons ensuite établir l'ordre SL nécessaire à la coloration des sommets de notre graphe.

Nous itérons jusqu'à que chaque sommet ait été ajouter dans le tableau d'ordre SL, c'est à dire tant que l'indice d'insertion dans le tableau d'ordre n'est pas égal au nombre de sommets du graphe $G(|V| = n)$.

L'idée générale est de retirer le sommet de plus petit degré à chaque itération, il s'agit du premier élément de la liste du degré minimum courant, si celle-ci n'est pas vide. On va ensuite vérifier qu'il s'agit d'un sommet valide, c'est-à-dire qu'il n'a pas déjà été traité. Si ce n'est pas le cas, nous avons notre candidat de degré minimum pour l'ordre SL, nous l'ajoutons dans le tableau de résultat d'ordre, à l'indice courant et on incrémente cet indice. Nous marquons ce sommet dans la liste des degrés courants avec une valeurs de -1 pour indiquer que ce sommet à été traité. Cette série d'opérations s'effectue en temps constant.

Ensuite, il va falloir mettre à jour les degrés des voisins de ce sommet candidat. Pour se faire, nous parcourons la liste d'adjacence de ce sommet dans le graphe originel G, pour chaque sommet n'étant pas encore traité dans V', on l'ajoute dans la liste à l'indice de son sommet courant - 1. Cela diminue le degré des voisins du sommet de 1 unité. Ici, on ne retire pas les sommets des listes de leur anciens degrés, car cela demanderait une recherche dans la liste ainsi qu'une suppression, ce qui implique une complexité linéaire et ne permettrait pas d'obtenir une complexité finale linéaire à notre application. Le fait de ne pas retirer les sommets lors de leur diminution de leur degré implique que des doublons se trouveront parmi les différentes listes, c'est pour cette raison que nous utilisons un tableau auxiliaire pour conserver le vrai degré courant de chaque sommet, ce qui nous permet de vérifier que le sommet retiré est le véritable candidat de degré minimal à traiter, tout ceci en temps constant. Si lorsque que l'on retire un sommet, l'indice de sa liste (son degré selon V') et plus grand que son degré dans le tableau de degrés, ou alors qu'il est marqué comme traité (-1), il s'agit donc d'un doublon et

nous pouvons le retirer. Comme il s'agit du premier élément de la liste, le retirer s'exécute en temps constant.

Lorsque l'on diminue le degré d'un voisin, si son nouveau degré est minimal on l'assigne au degré minimal courant.

Il est donc nécessaire pour trouver cet ordre de coloration de boucler sur chaque sommet du graphe V', donc n fois et de décrémenter les degrés des voisins d'un sommet, opération effectuée au plus m fois durant l'exécution, les sommets diminuant à chaque itération ainsi que les arêtes. La complexité finale de cette étape est ainsi en $\mathcal{O}(n + m)$.

Dans cette étape, nous utilisons les structures initialisées à la première étape, de ce fait nous n'utilisons pas de mémoire supplémentaire, $\mathcal{O}(1)$.

Coloration des sommets

Notre ordre SL étant maintenant défini dans notre tableau d'ordre, nous allons pouvoir attribuer une couleur à chacun de ces sommets. Nous initialisons deux tableaux supplémentaires, un tableau d'entier permettant d'attribuer une couleur à un sommet, de taille n, il s'agira du tableau retourné comme résultat à la fin de notre application. Le second tableau est constitué de n entiers permettant de savoir si un sommet a déjà été colorié.

L'ordre obtenu dans le tableau d'ordre est trié selon le critère SL, mais il faudra parcourir ce tableau depuis la fin pour la coloration, car on doit colorier les sommets de plus petits degrés en dernier.

Pour appliquer une coloration à chaque sommet de notre graphe, nous allons itérer sur chaque sommet et sur chacun de ses voisins pour trouver une couleur compatible à chaque sommet.

Cette opération va nécessiter le parcours des n sommets du graphe G avec un total de $2m$ voisins pouvant être parcouru au maximum sur l'ensemble du graphe (car le graphe est non orienté). On obtient finalement une complexité en $\mathcal{O}(n + m)$.

Au niveau de la complexité spatiale, nous initialisons simplement deux nouveaux tableaux de taille n, donc $\mathcal{O}(n)$.

Conclusion

Pour conclure, nos trois étapes s'exécutant l'une après l'autre, la complexité temporelle de notre solution s'élève en, $\mathcal{O}(3*(n+m))$ est respecté donc la contrainte de linéarité souhaitée en $\mathcal{O}(n+m)$.

La complexité spatiale de notre solution s'élève quant à elle à $\mathcal{O}(n + m)$ + $\mathcal{O}(n)$, donc respecté également la contrainte linéaire en $\mathcal{O}(n + m)$.

Pseudocode

Algorithme 1 : Etablissement d'un ordre de coloration SL

Result : Tableau de sommets selon un ordre SL (triés selon le degré des sommets croissant)

Input : Un graphe simple et non orienté $G = (V, E)$ comptant $n = |V|$ sommets (numérotés de 1 à n) et $m = |E|$ arêtes (n $\geq$ 3 et m $\geq$ 3) représente sous forme d'un tableau de liste d'adjascence.

```
1  degréMin := Δ(G) + 1
2  sousGraphe := Tableau de taille Δ(G) de Liste doublement chaînée de sommets
3  degrésCourant := [0, ..., n − 1]
4  ordre := [0, ..., n − 1]
5  indexOrdre := 0
   /* Initialisation du tableau de degrés courant                    */
6  for S ∈ V do
7      sousGraphe[degréMin] ← S
8      degrésCourant[S] ← Δ(S)
9      if Δ(S) < degréMin then
10         degréMin ← Δ(S)
   /* Détermination de l'ordre de traitement des sommets             */
11 while indexOrdre < n do
12     if sousGraphe[degréMin] est vide then
13         degréMin ← degréMin + 1
14         continue
15     S := Retire premier élément de sousGraphe[degréMin]
16     if degréMin > degrésCourant[S] then
           /* Le sommet S a déjà été traité                          */
17         continue
18     degrésCourant[S] ← −1
19     ordre[indexOrdre] ← S
20     indexOrdre ← indexOrdre + 1
21     for U ∈ voisins(S) do
22         if degrésCourant[U] = −1 then
23             continue
24         degrés := degrésCourant[U] − 1
25         Ajoute U à la fin sousGraphe[degrés])
26         degrésCourant[U] ← degrés
27         if degrés < degréMin then
28             degréMin ← degrés
29 return ordre
```

Algorithme 2 : Coloration des sommets

Input : Tableau de sommets selon un ordre SL (triés selon le degré des sommets croissant)

Result : Coloration des sommets du tableau en entrée

```
1  sommetsUtilisés := [0, ..., Δ(G)] (initialisé à -1)
2  couleurs := [1, ..., n]
3  for S ∈ ordre do
4      for U ∈ voisins(S) do
5          if sommetsUtilisés[U]! = −1 then
6              sommetsUtilisés[couleurs[U]] ← S
7      for k ∈ [0, ..., Δ(G)] do
8          if sommetsUtilisés[k]! = S then
9              couleurs[S] ← k
10             break
11 return couleurs
```