Graphe et coloration

BELDILMI Guillaume - KHENTICHE Wissam - DJERA Feriel

Introduction

La coloration k-impropre

La coloration des sommets d'un graphe consiste à associer une couleur à chaque sommet de manière à ce que deux sommets adjacents n'aient pas la même couleur. Une coloration k-impropre est une coloration dans laquelle chaque sommet a au plus k voisins de même couleur que lui.

Le nombre chromatique k-impropre d'un graphe G, noté $\chi k(G)$, est le nombre minimum de couleurs nécessaires pour une coloration k-impropre de G. Si k = 0, cela correspond simplement au nombre chromatique usuel de G.

En d'autres termes, pour déterminer le nombre chromatique k-impropre d'un graphe, il faut trouver le nombre minimum de couleurs nécessaires pour colorier les sommets du graphe de telle sorte que chaque sommet ait au plus k voisins de même couleur. Cela implique que si un sommet a plus de k voisins de même couleur, il doit être colorié avec une couleur différente de celle de ses voisins. Le but est de minimiser le nombre total de couleurs utilisées pour la coloration du graphe tout en satisfaisant cette contrainte de k-impropreté.

La méthode DSATUR (Degree of Saturation) est un algorithme de coloration de graphe qui utilise une heuristique gloutonne. Dans l'algorithme de base, chaque sommet est d'abord trié en fonction de son degré de saturation, qui est défini comme le nombre de couleurs différentes utilisées par ses voisins. Le sommet avec le degré de saturation le plus élevé est ensuite choisi et coloré avec la plus petite couleur disponible. Le processus se répète jusqu'à ce que tous les sommets soient colorés

Exercice 1 :

1. Algorithme

La principale modification consiste à ajouter un troisième paramètre k à la fonction convientDSAT(x, c, k) qui est appelée pour vérifier si la couleur c convient au sommet x ou pas. Ce paramètre k permet de s'assurer que le degré d'impropreté est respecté, c'est-à-dire qu'un sommet ne peut avoir plus de k voisins de la même couleur.

Donc il suffit d'utiliser cette fonction dans la boucle while de la fonction DSATUR, qui renvoie (nb<=k), si elle retourne vrai donc ça convient sinon ça ne convient pas.

2. Observation des résultats

On remarque que le nombre de couleurs utilisées est plus petit que dans l'algorithme de base et diminue à mesure que k augmente, ce qui est normal car on a ajouté une contrainte qui est le degré d'impropreté. Ceci nous permet d'utiliser moins de couleurs que dans l'algorithme de base.

Exercice 2

1. Algorithme

La modification demandée consiste à changer le calculer du degré de saturation DSAT des sommets dans l'algorithme DSAUTR, de manière à ce qu'il représente le nombre de couleurs différentes c pour lesquls plus de k voisins sont de cette couleur c.

Pour cela, la fonction convientDSAT2(x, c, k) a été ajoutée, qui teste si la couleur c peut être donnée au sommet x en vérifiant si plus de k de ses voisins ont déjà cette couleur c, en utilisant la condition (nb>k).

Ensuite nous avons créé DSATUR2(k) qui utilise le nouveau critère de DSAT, qui initialise les tableaux couleur2, DSAT et Degre avec les nouvelles valeurs de DSATUR2, ensuite la boucle principale qjui colorie les sommets un par un en choisissant pour chaque sommet la couleur c la plus petite possible qui vérifie la condition de la nouvelle définition DSAT avec la deuxième boucle imbriquée, ensuite la mise à jour des valeurs de DSAT pour chaque sommet j voisin du sommet x colorié.

2. Observation des résultats

On remarque que le nombre de couleurs utilisées est relativement similaire à celui de l'algorithme précédent, on pourra remarquer que le nombre de couleurs utilisées est parfois plus petit que celui de l'algorithme précédent. Ceci est dû au fait que le nouveau critère de DSAT est plus restrictif que le précédent, ce qui permet d'utiliser moins de couleurs.

Exercice 3

1. Algorithme

La fonction degmoy calcule le degré moyen des sommets dans le graphe en effectuant une somme des degrés de tous les sommets, puis en la divisant par le nombre total de sommets.

On commence par déclarer une variable deg de type float initialisée à zéro pour stocker la somme des degrés. Ensuite, cette fonction utilise deux boucles for imbriquées pour parcourir la matrice d'adjacence adj et ajoute le nombre de connexions (ou arêtes) entre chaque paire de sommets dans deg. Comme la matrice d'adjacence est symétrique, seuls les éléments situés au-dessus ou en diagonale de la diagonale principale sont additionnés.

Après avoir terminé la somme, la fonction divise la somme totale par le nombre de sommets n pour obtenir le degré moyen. La fonction retourne le résultat en tant que nombre à virgule flottante.

2. Évolution du degré moyen

Le degré moyen d'un graphe aléatoire dépend de n, le nombre de sommets, et p, la probabilité de la présence d'une arête entre deux sommets.

En augmentant n, le degré moyen va également augmenter, car le nombre d'arêtes dans le graphe va également augmenter. Cependant, si p reste constant, l'augmentation du degré moyen sera relativement faible, car chaque sommet ne sera relié qu'à une fraction de plus en plus petite des autres sommets.

En augmentant p, le degré moyen va également augmenter rapidement. Si p = 1, le graphe sera complet et le degré moyen sera égal à n-1, le maximum possible. Cependant, si p est proche de 0, le graphe sera très clairsemé et le degré moyen sera très faible.

En résumé, le degré moyen d'un graphe aléatoire G(n, p) augmente avec n et p.

3. Évolution du nombre de couleurs utilisées

Dans l'exercice 1, on modifie l'algorithme pour qu'il prenne en compte un paramètre k, qui représente le degré d'impropreté souhaité. Cela signifie que les sommets avec un degré de saturation inférieur à k peuvent être considérés comme non saturés et donc avoir une couleur non optimale. Cela peut permettre d'utiliser moins de couleurs que dans l'algorithme de base, car certains sommets peuvent être colorés avec une couleur non optimale.

Dans l'exercice 2, on modifie encore l'algorithme pour que le degré de saturation des sommets représente le nombre de couleurs c pour lesquelles plus de k voisins sont de couleur c. Cela permet de prendre en compte plus de contraintes lors de la coloration, ce qui peut conduire à une réduction du nombre de couleurs utilisées.

Si on prend k égal au degré moyen, cela signifie que la plupart des sommets ont un degré similaire et donc des contraintes similaires. Cela peut réduire le nombre de couleurs utilisées par rapport à l'algorithme de base DSATUR, mais cela dépend toujours du graphe en question.

Exercice 4

1. Algorithme

On nous demande de modifier l'algorithme exact de coloration de graphe pour calculer le nombre chromatique k-impropre d'un graphe. Le nombre chromatique k-impropre d'un graphe est le nombre minimum de couleurs nécessaires pour colorer le graphe de sorte que chaque sommet ait au moins k couleurs différentes parmi ses voisins.

La fonction colorexact est modifiée pour prendre en compte le degré d'impropreté k dans ses paramètres. La fonction colorRR, qui est appelée à partir de colorexact, est responsable de tester toutes les combinaisons possibles de couleurs pour chaque sommet, en veillant à ce que les couleurs des sommets adjacents soient différentes. Si une coloration en k couleurs est trouvée, la variable globale trouve est mise à vrai.

La fonction nbChromatique calcule le nombre chromatique en testant à partir de d couleurs et en diminuant k tant que c'est possible. Elle utilise la fonction colorexact pour tester toutes les combinaisons de couleurs jusqu'à ce qu'une coloration en k couleurs soit trouvée. Si une telle coloration est trouvée, la fonction colorexact mettra la variable trouve à vrai, indiquant que le graphe peut être coloré avec k couleurs. Si aucune coloration n'est trouvée, la fonction nbChromatique continuera à diminuer k jusqu'à ce qu'une coloration soit trouvée. En résumé, l'algorithme exact modifié fonctionne en essayant toutes les combinaisons de couleurs possibles jusqu'à ce qu'une coloration en k couleurs soit trouvée. La variable k est d'abord initialisée à d+1, puis diminuée jusqu'à ce qu'une coloration soit trouvée ou que k soit égal à 1. Ainsi, la fonction nbChromatique calcule le nombre chromatique k-impropre en trouvant le nombre minimum de couleurs nécessaires pour colorer le graphe de sorte que chaque sommet ait au moins k couleurs différentes parmi ses voisins.

2. Comparaison des résultats

En comparant les résultats de l'algorithme exact modifié avec ceux de DSATUR2, on peut voir que les deux algorithmes donnent des résultats sensiblement identiques.

L'algorithme exact modifié donne parfois des résultats légèrement meilleurs que DSATUR2 pour certains graphes.

Cependant, sur de grands graphes, l'algorithme exact modifié est beaucoup plus lent que DSATUR2 et nécessite beaucoup plus d'iteration et de temps de calcul pour donner un résultat. Là où DSATUR2 donne un résultat en une seule itération de ce dernier, l'algorithme exact modifié peut prendre plusieurs minutes pour donner le même résultat.

Exercice 5

Les graphes de disques, également appelés "graphes circulaires", sont une représentation visuelle des données sous forme de cercles ou d'anneaux. Dans un graphe de disques, chaque cercle représente une catégorie de données et la taille de chaque cercle est proportionnelle à la quantité de données qu'il représente. Les données sont souvent présentées sous forme de pourcentages ou de fractions pour faciliter la comparaison entre les différentes catégories. Les graphes de disques sont souvent utilisés pour représenter des données de manière claire et concise, et pour mettre en évidence les proportions relatives entre les différentes catégories. Ils peuvent être utiles pour visualiser les parts de marché d'une entreprise, la répartition des dépenses dans un budget, les résultats d'un sondage ou toute autre information qui peut être catégorisée en groupes distincts.
Pour cela nous avons rajouté les fonctions dist, calccord, et nous avons modifié la fonction genere

Nous avons créé la fonction dist prend en entrée deux indices de sommets a et b, et une distance maximale d. Elle calcule la distance Euclidienne (ou la norme L2) entre les points du plan correspondant aux coordonnées des sommets a et b. Si cette distance est inférieure ou égale à d, la fonction renvoie 1 (pour indiquer que les sommets sont voisins), sinon elle renvoie 0 (pour indiquer qu'il n'y a pas d'arête entre les sommets).

La fonction calccoord calcule les distances entre chaque paire de sommets dans un graphe non orienté et pondéré représenté par la matrice d'adjacence adj.

La fonction parcourt chaque paire de sommets distincts (i et j) en utilisant deux boucles for. Pour chaque paire, elle appelle la fonction dist(i, j, d) qui calcule la distance entre les sommets i et j en utilisant la distance euclidienne dans un espace à d dimensions. La distance calculée est stockée dans la matrice d'adjacence adj aux positions (i,j) et (j,i) pour refléter le fait que le graphe est non orienté.

La fonction genere génère aléatoirement un graphe non orienté avec n sommets, place les sommets à des positions aléatoires dans un plan cartésien et relie chaque paire de sommets dont la distance est inférieure ou égale à d.

Elle prend en entrée un entier d qui représente la probabilité d'avoir une arête entre deux sommets d'un graphe non orienté généré aléatoirement. Cette fonction utilise également deux variables globales coord et adj.

La première boucle for itère de 0 à n-1 (où n est le nombre de sommets du graphe), et à chaque itération, elle génère des coordonnées aléatoires pour le sommet correspondant en utilisant la fonction rand() % L pour la coordonnée x et rand() % H pour la coordonnée y, où L et H sont des constantes définies ailleurs dans le code. Ainsi, chaque sommet du graphe est placé à une position aléatoire dans un plan cartésien.

La deuxième boucle for commence à partir de i+1 et itère jusqu'à n-1. À chaque itération, elle calcule la distance entre le sommet i et le sommet j en utilisant la fonction dist(i, j, d). Cette fonction renvoie une distance aléatoire entre 0 et d pour deux sommets donnés. Si la distance calculée est inférieure ou égale à d, cela signifie que les sommets i et j sont connectés par une arête. Dans ce cas, la matrice d'adjacence adj est mise à jour en conséquence pour refléter cette arête.

En observant les résultats obtenus après plusieurs essais avec h et l constants, nous constatons que le nombre d'arêtes du graphe augmente avec d et n.

Augmenter n signifie augmenter le nombre de sommets dans le graphe et donc, le nombre de potentielles arêtes.

Augmenter d signifie que la probabilité d'avoir une arête entre deux sommets augmente avec d. Cela est dû au fait que la fonction dist renvoie une distance aléatoire entre 0 et d, et que la probabilité d'obtenir une distance inférieure ou égale à d augmente avec d. Ainsi, plus d augmente, plus la probabilité d'avoir une arête entre deux sommets augmente car le graphe se rapproche de plus en plus d'un graphe complet augmentant ainsi le nombre chromatique.