Couverture de graphe

1 Introduction

Soit un graphe $G = (V, E)$ non orienté, où V est l'ensemble des n sommets et E l'ensemble des m arêtes de G. Une couverture de G est un ensemble $V' \subseteq V$ tel que toute arête $e \in E$ a (au moins) une de ses extrémités dans V. Le but du problème Vertex Cover est de trouver une couverture contenant un nombre minimum de sommets.

Ce problème est NP-difficile (sa version décision [1] est NP-complète). Le but de ce projet est d'implémenter différents algorithmes, exacts et approchés, pour résoudre le problème Vertex Cover, et de les tester expérimentalement sur différentes instances.

Travail demandé

Chacune des sections suivantes comporte un certain nombre de questions. Il vous est demandé de :
- Réaliser un programme implantant les différents algorithmes et permettant de réaliser les tests demandés. Le code devra être commenté afin de permettre à un lecteur d'en comprendre le fonctionnement. Le choix du langage est libre.
- Ecrire un rapport (autour de 8 pages, 10 maximum) qui contient les réponses aux questions théoriques, les courbes/résultats numériques obtenus et leurs explications, interprétations,... Il est inutile d'inclure le code dans le rapport.

Le travail est à réaliser **en binôme**.

VOUS DEVEZ ETRE EN BINOME AVEC QUELQU'UN DE VOTRE GROUPE DE TD.

La date de rendu (rapport et code) sera donnée par votre chargé de TD. Tout retard sera pénalisé.

Le projet donnera lieu à une soutenance en salle machine. Lors de la soutenance, il pourra vous être demandé d'exécuter une ou plusieurs de vos fonctions sur un graphe fourni dans un fichier .txt dans un format identique au fichier "exempleinstance.txt" fourni sur le moodle. Vous devez donc écrire une méthode permettant de créer un graphe à partir d'un tel fichier texte.

2 Graphes

On pourra représenter un graphe par un tableau de listes d'adjacence. D'autres représentations sont également possibles. Il est également autorisé d'utiliser une bibliothèque de graphes pour gérer vos graphes (dans ce cas vous pouvez utiliser des méthodes existantes pour les opérations de base, mais pas pour la suite bien sûr).

2.1 Opérations de base

1. Coder une méthode prenant en paramètres un graphe G et un sommet v de G, et qui retourne un nouveau graphe G' obtenu à partir de G en supprimant le sommet v (et les arêtes incidentes). Précision : la méthode doit fonctionner même si les n sommets du graphe ne sont pas les entiers de 0 à $n-1$ (par exemple on peut vouloir supprimer le sommet 3 d'un graphe avec 4 sommets $\{0, 3, 6, 8\}$).
2. Généraliser la méthode précédente lorsqu'un ensemble de sommets est supprimé.

1. Etant donné un graphe G et un entier k, existe-t-il une couverture de taille au plus k ?

3. Coder une méthode prenant en entrée un graphe et renvoyant un tableau contenant les degrés des sommets du graphe, et une autre permettant de déterminer un sommet de degré maximum. Ici aussi, la méthode doit fonctionner même si les n sommets du graphe ne sont pas les entiers de 0 à $n-1$.

2.2 Génération d'instances

Afin de tester vos algorithmes, vous allez générer des graphes aléatoires.

1. Coder une méthode prenant en entrée un entier $n > 0$ et un paramètre $p \in]0,1[$, et qui renvoie un graphe sur n sommets, et où chaque arête (i,j) est présente avec probabilité p. Les tirages au sort seront indépendants pour les différentes arêtes.

3 Méthodes approchées

Méthode du couplage maximal

Un couplage est un ensemble d'arêtes n'ayant pas d'extrémité en commun.

```
algo_couplage(G)
----------------
Entrée: G=(V,E=(e_1,...,e_m)) un graphe
Sortie: Une couverture
----------------
  C = emptyset
  Pour i de 1 à m:
      Si aucune des deux extrémités de e_i n'est dans C, alors:
          Ajouter les deux extrémités de e_i à C
      Fin Si
  Fin Pour
  Renvoyer C
```

Algorithme glouton

```
algo_glouton(G)
---------------
Entrée: G=(V,E=(e_1,...,e_m)) un graphe
Sortie: Une couverture
----------------
  C = emptyset
  Tant que E n'est pas vide:
      Prendre un sommet v de degré maximum
          Ajouter v à C, et supprimer de E les arêtes couvertes par v
      Fin Si
  Fin Pour
  Renvoyer C
```

On rappelle que l'algorithme `algo-couplage` est 2-approché.

1. Montrer que l'algorithme glouton précédent n'est pas optimal. En déduire que l'algorithme glouton n'est pas r-approché, pour un certain r que à préciser.

2. Coder les deux méthodes précédentes. Les comparer (en fonction de n et p) du point de vue (1) de leur temps de calcul (2) de la qualité des solutions retournées.
3. (Facultatif) Montrer que l'algorithme glouton précédent n'est pas r-approché, quel que soit r.

Note sur les tests : Lorsque vous ferez des tests pour mesurer par exemple le temps de calcul sur des instances aléatoires, il vous est suggéré de :
- mesurer la taille N_{max} jusqu'à laquelle l'algorithme tourne rapidement, disons de l'ordre de quelques secondes.
- pour chaque taille d'instances $n \in \{N_{max}/10, 2N_{max}/10, 3N_{max}/10, \ldots, N_{max}\}$: générer quelques dizaines d'instances de taille n, mesurer le temps de calcul moyen t_n sur ces instances. Tracer la courbe t_n en fonction de n sur la base de ces 10 points. On pourra penser à utiliser des échelles logarithmiques si besoin.

NB : avant de faire des tests sur des instances générées, vérifiez toujours que votre code fonctionne sur quelques exemples simples !

4 Séparation et évaluation

4.1 Branchement

Le but de cette section est de faire un premier algorithme de branchement simple. Nous l'améliorerons dans les sections suivantes. Le branchement considéré ici est le suivant : prendre une arête $e = \{u, v\}$, et considérer deux cas : soit u est dans la couverture, soit v est dans la couverture.

Ainsi, partant d'un graphe initial G et d'un ensemble de sommets C vide représentant la solution, la première branche consiste à mettre u dans C et à raisonner sur le graphe où l'on a supprimé u (les arêtes incidentes à u sont couvertes) ; symétriquement, la deuxième consiste à mettre v dans C et à raisonner sur le graphe où l'on a supprimé v.

1. Coder cette méthode. Vous utiliserez une pile pour gérer les nœuds de l'arbre. Votre méthode devra renvoyer la meilleure solution. **Vérifier la validité de votre méthode sur des exemples simples** (vérifier que l'ensemble des noeuds visités est bien correct).
2. Tester cette méthode, en évaluant son temps de calcul en fonction de n. Vous pourrez faire varier la probabilité p de présence des arêtes. Commentez.

NB1 : il vous est demandé de faire des tests sur les temps de calcul. Il est également intéressant de tester le nombre de nœuds générés par les différentes méthodes. Cela est facultatif.

NB2 : dans les tests numériques, il est intéressant de regarder l'influence de p également (en testant plusieurs valeurs par exemple). On pourra aussi tester avec des valeurs comme $p = 1/\sqrt{n}$ pour avoir des graphes moins denses.

4.2 Ajout de bornes

Soit G un graphe, M un couplage de G et C une couverture de G. Alors

$$|C| \geq \max\{b_1, b_2, b_3\}$$

avec $b_1 = \lceil \frac{m}{\Delta} \rceil$ (où Δ est le degré maximum des sommets du graphe), $b_2 = |M|$, $b_3 = \frac{2n-1-\sqrt{(2n-1)^2-8m}}{2}$.

1. Montrer la validité de chacune des bornes b_1, b_2, b_3. Pour b_3, on pourra s'interroger sur le nombre maximal d'arêtes d'un graphe dont C est une couverture.

2. Insérer dans l'algorithme précédent (1) le calcul (en chaque nœud) d'une solution réalisable obtenue avec l'algorithme de couplage (2) le calcul d'une borne inférieure basée sur la relation précédente. Tester cette nouvelle méthode, et commenter.
3. (Facultatif) Evaluer l'intérêt d'utiliser aussi l'algorithme glouton pour générer des solutions réalisables. Evaluer l'efficacité des méthodes si l'on n'utilise que les bornes inférieures, ou que le calcul de solution réalisable (avec une borne inférieure triviale).

4.3 Amélioration du branchement

Lorsque l'on branche sur une arête $e = \{u, v\}$, dans la deuxième branche où l'on prend le sommet v dans la couverture, on peut supposer que l'on ne prend pas le sommet u (le cas où on le prend étant traité dans la première branche. Dans la 2ème branche, ne prenant pas u dans la couverture on doit alors prendre tous les voisins de u (et on peut les supprimer du graphe donc).

1. Modifier la méthode précédente en utilisant ce branchement amélioré. Tester la nouvelle méthode, et commenter les résultats obtenus.
2. Afin d'éliminer un maximum de sommets dans la deuxième branche, il semble intéressant de choisir le branchement de manière à ce que le sommet u soit de degré maximum dans le graphe restant. Tester la nouvelle méthode, et commenter les résultats obtenus.
3. (Facultatif) Dans un graphe G, si un sommet u est de degré 1, expliquer pourquoi il existe toujours une couverture optimale qui ne contient pas u. Insérer alors ce test permettant de supprimer un sommet sans brancher. Evaluer expérimentalement l'intérêt d'ajouter ce test.

4.4 Qualité des algorithmes approchés

1. Evaluer expérimentalement (en fonction de n) le rapport d'approximation des algorithmes de couplage et glouton. Donner également le pire rapport d'approximation obtenu lors des expérimentations, pour chacun des deux algorithmes.
2. (Facultatif) Proposer une ou plusieurs autres heuristiques, et les évaluer expérimentalement.