Projet COMPLEX
Problème du VERTEX COVER

Esther CHOI (3800370)
Folco BERTINI
M1 DAC - Groupe 1

October 18, 2021

Contents

Abstract

Ce document constitue le rapport du projet de l'UE COMPLEX, suivie au premier semestre du M1 Informatique à Sorbonne Université.
Les études de performance ont été réalisés avec un processeur Intel ©Core ™i7-8550U 1.80GHz.

1 Définition du problème

Une couverture d'un graphe est un ensemble de sommets qui couvre tous les sommets du graphe.
Le problème VERTEX COVER est défini de la façon suivante :

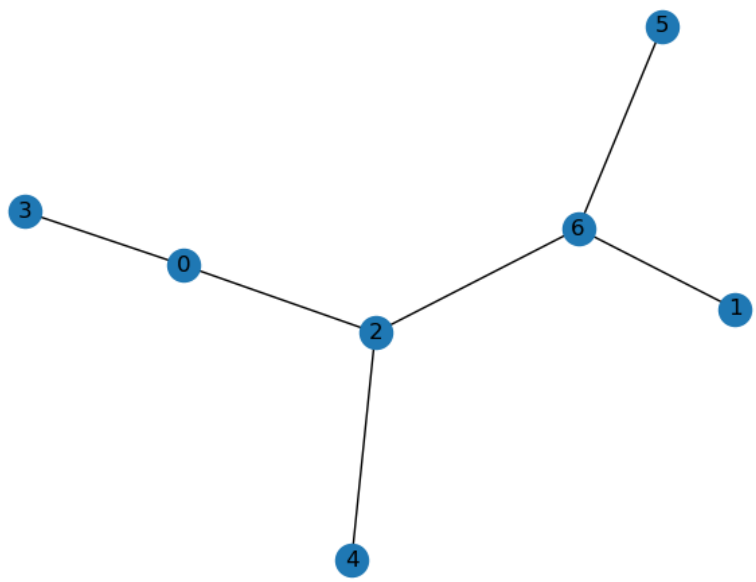
- entrée : un graphe non orienté G
- sortie : une couverture de G de taille minimale

Le but de ce projet est d'implémenter des algorithmes approchés et exacts pour résoudre le problème VERTEX COVER.

2 Méthodes approchées

1) Soit le graphe I suivant :

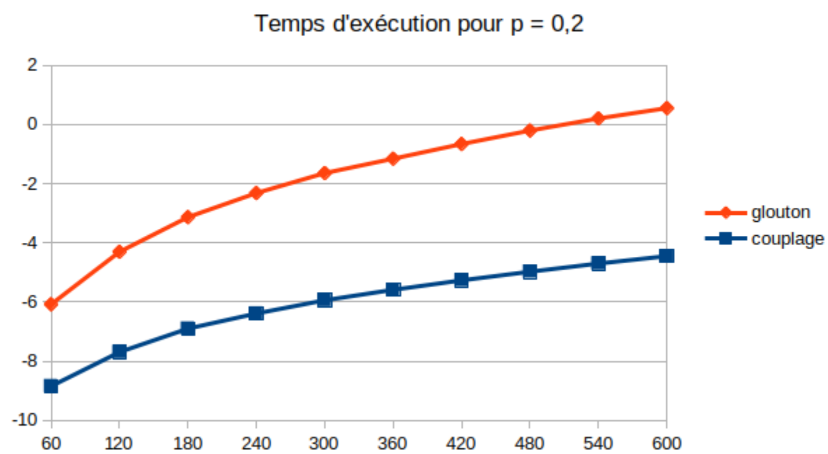
Figure 1: Graphe I

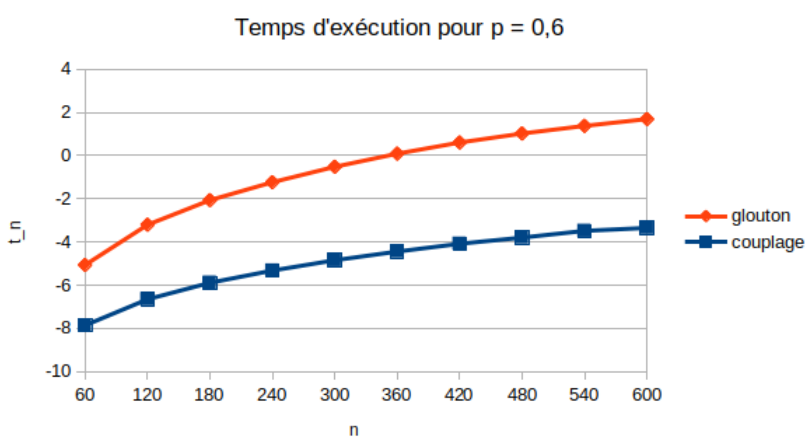


Une couverture optimale est $C_{opt} = \{0, 2, 6\}$. L'algorithme glouton renvoie la solution $C = \{0, 1, 2, 5\}$ qui a un sommet de plus (l'exécution complète est donnée en annexe). Ceci montre que algo_glouton n'est pas optimal et qu'il n'est pas 1.2-approché.
En effet, s'il l'était, alors pour toute instance de VERTEX COVER, le rapport d'approximation entre la solution retournée et une solution optimale serait inférieur ou égal à 1.2.
Or pour l'instance I précédente, ce rapport vaut $r = \frac{|C|}{|C_{opt}|} = \frac{4}{3} \approx 1.33 > 1.2$.
Donc algo_glouton n'est pas 1.2-approché.

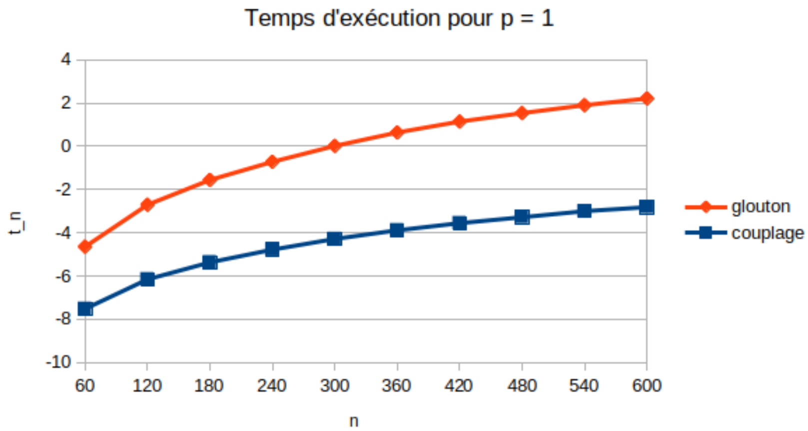
2) Comparons les deux algorithmes algo_couplage et algo_glouton.
Pour cela, nous avons commencé par calculer N_{max} comme suggéré dans l'énoncé. Nous avons pris $p = 1$ et nous nous sommes limités à un temps d'éxécution de 10 secondes. Nous avons ainsi trouvé $N_{max} = 600$.

1. *Comparaison du point de vue du temps de calcul* :
   Les graphiques suivants montrent les temps de calcul pris par les deux algorithmes en fonction de n le nombre de sommets et p la probabilité d'apparition d'une arête (les valeurs sont les valeurs logarithmiques).
   Nous avons pris comme ensemble valeur pour n l'ensemble $\{N_{max}/10, 2N_{max}/10, ..., N_{max}\}$.
   Pour chaque n, nous avons généré aléatoirement 10 graphes sur lesquels nous avons appliqué les fonctions algo_couplage et algo_glouton. Nous avons pris la moyenne des temps d'exécution.





De ces graphiques, nous pouvons observer que pour les deux algorithmes, le temps de calcul augmente de façon linéaire avec n et p, et que algo_glouton est nettement



meilleur que algo_couplage.
Nous pouvions nous y attendre puisque algo_couplage est en $O(nm)$ et algo_glouton en $O(??)$

2. *Comparaison du point de vue de la qualité de la solution* :

3 Méthodes exactes : algorithme de branch-and-bound

1.2) Voici le tableau contenant le temps de calcul de la fonction branch en fonction de n.

2.1) Soit $G = (V, E)$ un graphe non orienté, où V est l'ensemble des sommets de et E l'ensemble des arêtes. On note $n = |V|$ et $m = |E|$.
Soit M un couplage et C une couverture de G.
Posons $b_1 = \lceil \frac{m}{\Delta} \rceil$, $b_2 = |M|$ et $b_3 = \frac{2n-1-\sqrt{(2n-1)^2-8m}}{2}$.
Montrons que $|C| \geq b_1, b_2, b_3$.

- Soit Δ le degré maximum des sommets de G. Comme chaque sommet de C est une extrémité d'au plus Δ arêtes, on a : $|C| \times \Delta \geq m \implies |C| \geq \frac{m}{\Delta}$.
  $|C|$ étant un entier, on peut prendre $b_2 = \lceil \frac{m}{\Delta} \rceil$ comme borne inférieure (???).
- Pour toute arête de M, une de ses extrémités est dans C (sinon elle ne serait pas couverte et C ne serait pas une couverture). De plus, ces extrémités sont toutes distinctes par définition d'un couplage. Ainsi on a bien $\boxed{|C| \geq b_2}$
- ??? [ça ressemble à la formule d'une racine d'un polynôme du second degré.]