Faculté des Sciences et Ingénierie - Sorbonne université

Master Informatique parcours - DAC / ANDROIDE

COMPLEX - Complexité, algorithmes randomisés et approchés

Rapport de projet

Couverture de graphe

Réalisé par :
ATTABI Melissa Dahlia - M1 DAC
SLIMANI Nour Ismahane - M1 ANDROIDE

Octobre 2023

Table des matières

Table des figures

Introduction

L'étude des problèmes liés aux graphes occupe une place cruciale dans de nombreux domaines tels que l'informatique, les réseaux et les systèmes complexes en général. L'un de ces problèmes fondamentaux est le Problème de Couverture par sommets (Vertex Cover), qui consiste à trouver un ensemble minimal de sommets qui couvrent toutes les arêtes d'un graphe non orienté donné.

Dans le cadre de ce projet, nous nous concentrons sur la mise en œuvre de diverses approches algorithmiques heuristiques, visant à résoudre le problème Vertex Cover. Notre objectif principal est de concevoir un programme informatique capable de déployer ces algorithmes et de les soumettre à des tests expérimentaux sur diverses instances de graphes. L'analyse de ces résultats expérimentaux nous permettra d'évaluer l'efficacité et les performances relatives de ces méthodes de résolution face à la complexité du problème.

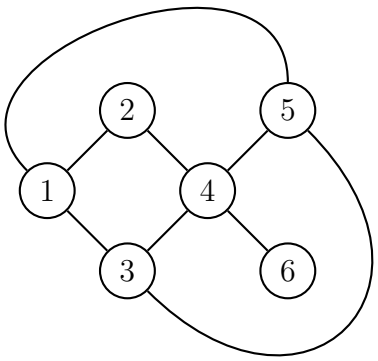
Nous présenterons en détail les algorithmes étudiés, en expliquant leur fonctionnement et en discutant de leur applicabilité dans la résolution du Problème Vertex Cover. Nous mettrons en avant les résultats obtenus lors des expérimentations. Enfin, nous présenterons une conclusion générale qui synthétisera les enseignements tirés de ce travail.

Chapitre 1

Graphes

1.1 Rpresentation des données

Dans le cadre de ce projet, nous avons choisi de représenter un graphe par des listes d'adjacences. Dans notre code, la structure de données utilisée est un dictionnaire en Python. Chaque clé du dictionnaire représente un sommet du graphe, et la valeur associée à chaque clé indique les sommets connectés à ce sommet spécifique.
Prenons un exemple pour mieux visualiser la structure :



Voici comment serait représenté ce graphe avec notre implémentation : {1 : [2,3,5], 2 : [1, 4], 3 : [1, 4, 5], 4 : [2, 3, 5, 6], 5 : [1, 3, 4], 6 : [4]}

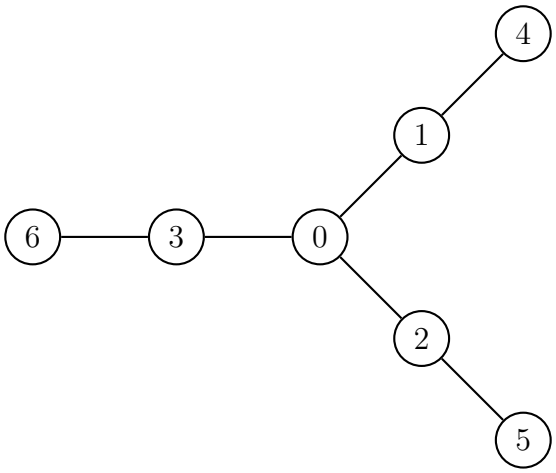
Chapitre 2

Méthodes approchées

2.1 Approximation de l'algorithme glouton

2.1.1 Montrer que l'algorithme glouton n'est pas optimal :

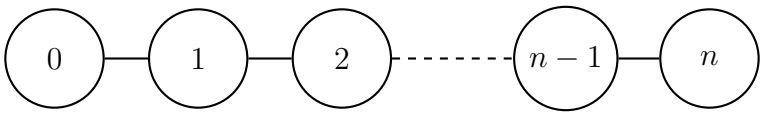
Soit le graphe $G = (V, E)$ suivant :



La couverture retournée par l'algorithme glouton sera : algo = [0,1,2,3].
Il commencera par ajouter le sommet (0) à la couverture car c'est le sommet de degré maximal ($deg(0) = 3$), il prendra ensuite les sommets $(1), (2)$ et (3) pour couvrir les arêtes restantes : (1,4), (2,5) et (3,6).
Or, la couverture optimale du graphe G est opt = [1,2,3]. On en déduit donc que l'algorithme glouton n'est pas optimal car : $|algo| > |opt|$, avec $|algo|$ le coût de la solution algo, et $|opt|$ le coût de la solution optimale.

2.1.2 Montrer que l'algorithme glouton n'est pas r-approché

Nous allons utiliser une généralisation des graphes en forme de chaîne.
Soit $G = (V, E)$ un graphe tel que : $V = \{1, 2, 3, ..., n\}$ l'ensemble des sommets, et $E = \{(1, 2), (2, 3), (3, 4), ..., (n - 1, n)\}$ l'ensemble des arêtes de G.



Pour cette forme de graphe, la stratégie optimale consiste à sélectionner les sommets de manière alternative, en commençant par le deuxième sommet et en s'arrêtant lorsque toutes les arêtes sont couvertes. La solution optimale, notée *opt*, serait donc égale à la partie entière inférieure de $\frac{n}{2}$ car on prend un sommet sur deux , soit $|opt| = \lfloor \frac{n}{2} \rfloor$.

On remarque bien que : $\forall i \in \{2, 3, \ldots, n-1\}, deg(i) = degré\_max = 2$.
Comme nous n'avons pas de contrainte dans la sélection des sommets dans l'algorithme glouton, excepté le fait qu'il faut choisir un sommet de degré maximum, supposons que la sélection se fait comme suit :
-Nous prenons dans la couverture le sommet $\lfloor \frac{n}{2} \rfloor + 1$ puis le supprimons, ce qui va générer 2 sous-chaînes $\{1, 2, ..., \lfloor \frac{n}{2} \rfloor\}$ et $\{\lfloor \frac{n}{2} \rfloor + 2, \lfloor \frac{n}{2} \rfloor + 3, ..., n\}$.
-On applique ensuite le même principe sur ces deux sous-chaînes jusqu'à l'obtention d'une couverture.
-Notons, *glouton* la solution retournée par l'algorithme glouton.

— Si n est pair : $|glouton| = |opt| = \lfloor \frac{n}{2} \rfloor$

— Si n est impair : $|glouton| = |opt| + 1 = \lfloor \frac{n}{2} \rfloor + 1$

Ce qui nous donne alors un rapport d'approximation :

$$r = \frac{\lfloor \frac{n}{2} \rfloor + 1}{\lfloor \frac{n}{2} \rfloor} = 1 + \lfloor \frac{2}{n} \rfloor$$

On voit bien que le rapport d'approximation dépend de la variable n, le nombre de sommets du graphe. Nous ne pouvons jamais trouver un rapport d'approximation fixe. Donc, l'algorithme glouton n'est pas r-approché, quel que soit r.

2.2 Comparaison des méthodes

2.2.1 Comparaison du temps de calcul

Complexité théorique

Soit un graphe $G = (V, E)$, avec $n = |V|$ le nombre de sommets de G, et $m = |E|$ le nombre d'arêtes.

1. L'algorithme de couplage a une complexité de $O(m)$ car il boucle sur les arêtes du graphe.
2. L'algorithme glouton quant à lui, a une complexité de $O(m * n)$:
   (a) Il boucle sur les arêtes du graphe : $O(m)$

(b) A chaque itération, Il calcule le sommet de degré maximum en $O(n)$ (parcourir tous les sommets du graphe et calculer leur degré)

(c) Puis supprime les arêtes couvertes par ce dernier en $O(n-1)$ (supprime le sommet de degré max de toutes les valeurs du dictionnaire, exceptée de celle du sommet en question).

(d) Ce qui nous donne $O(m*(n+n-1)) = O(m*n)$

Complexité expérimentale

Pour comparer les temps de calcul de l'algorithme de couplage et l'algorithme glouton, nous avons mesuré les temps d'exécution moyens sur 50 exécutions pour différentes tailles de graphe avec une probabilité de présence de chaque arête p fixée. Et ce, pour 100 tailles de graphes. Voici les résultats obtenus :

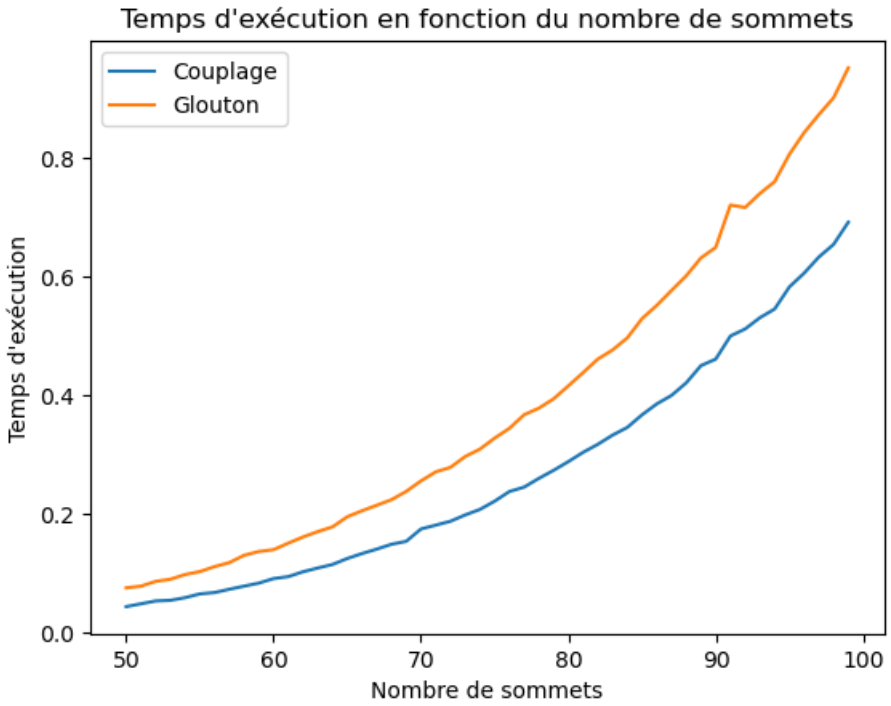


FIGURE 2.1 – Temps d'exécution de l'algo de couplage et l'algo glouton en fonction du nombre de sommets

On remarque que la courbe prend une forme polynomiale, le temps de calcul augmente plus le nombre de sommets augmente. Il est à noter aussi que la courbe de l'algorithme glouton est au dessus de celle de l'algorithme de couplage. On explique cela par le calcul de la complexité théorique effectué plus haut.

Nous avons ensuite fait la même chose pour différentes valeurs de probabilité p, en lui attribuant ces 5 valeurs suivantes : $[0.1; 0.3; 0.5; 0.7; 0.9]$, avec un nombre de sommets n fixé. Nous avons obtenu les résultats ci-dessous :

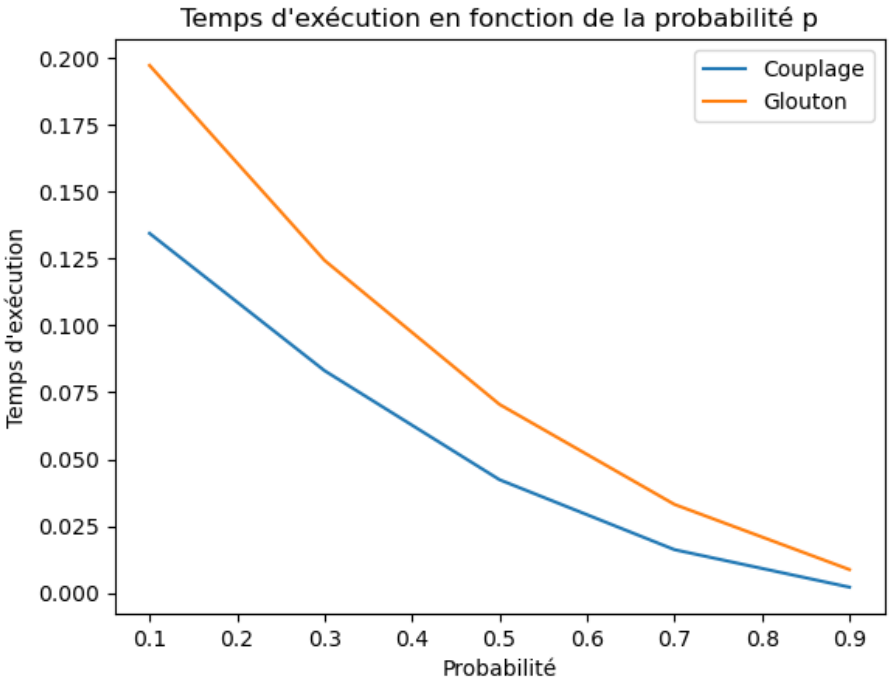


FIGURE 2.2 – Temps d'execution de l'algo de couplage et l'algo glouton en fonction de la probabilité p

Nous remarquons que le temps d'exécution diminue plus la probabilité de création d'arêtes augmente surtout pour l'algorithme glouton. Cela s'explique par le fait que lorsque p augmente, la suppression d'un sommet entraînera la suppression d'un nombre plus important d'arêtes (toutes les arêtes où le sommet apparaît). Nous trouverons donc une couverture plus rapidement.

2.2.2 Comparaison de la qualité des solutions

Nous avons effectué le même raisonnement précédent, mais cette fois-ci en calculant la taille des solutions retournées par les deux algorithmes.

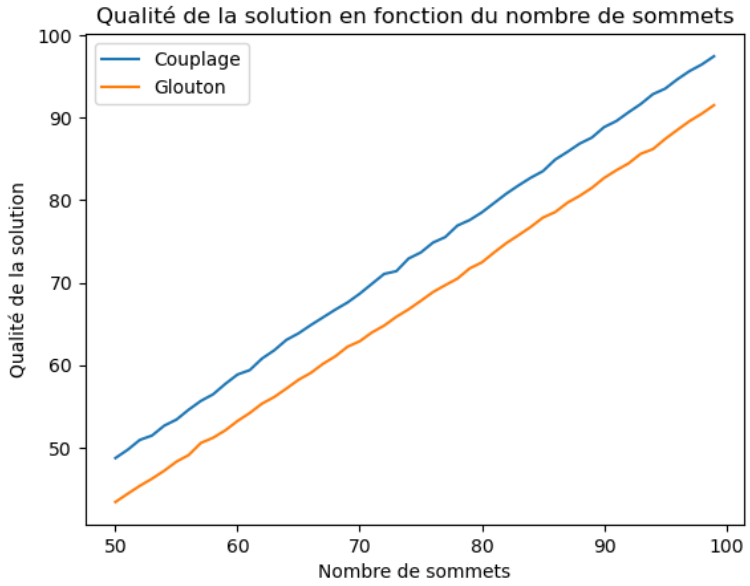


FIGURE 2.3 – Qualité de la solution de l'algo de couplage et l'algo glouton en fonction du nombre de sommets

On observe sur la figure que la qualité des solutions de l'algorithme de couplage est moins bonne que celle de l'algorithme glouton. Alors que glouton choisit à chaque itération les arêtes de degré max, l'algorithme couplage fais un choix aléatoire. On obtient donc une meilleure solution avec l'algorithme glouton qu'avec l'algorithme de couplage.

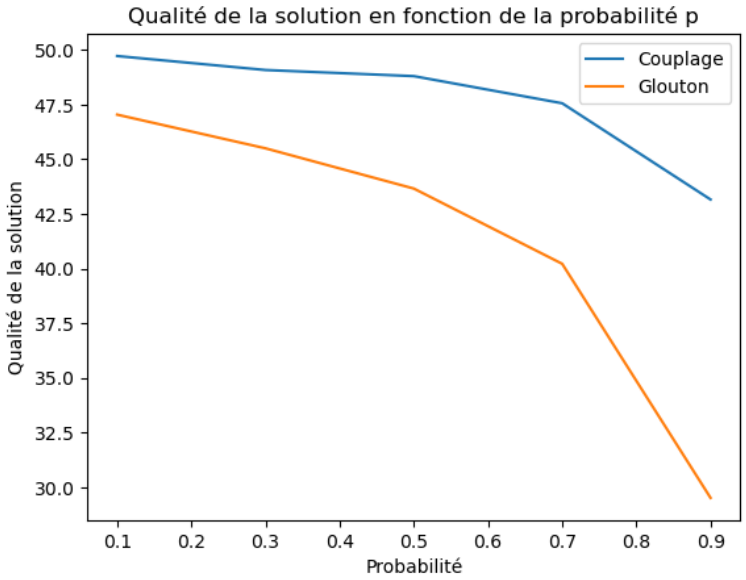


FIGURE 2.4 – Qualité de la solution de l'algo de couplage et l'algo glouton en fonction de la probabilité p

Comme expliqué précédemment, en supprimant un sommet pour une probabilité de création d'arête plus grande, un nombre plus grand d'arêtes sera également supprimé. Par conséquent, nous pourrons obtenir une couverture de manière plus efficace.

Chapitre 3

Séparation et évaluation

3.1 Branchement

Nous avons evalué le temps de calcul de l'algorithme de branchement en fonction du nombre de sommets, puis en fonction de la probabilité de présence des arêtes. Voici les résultats observés :

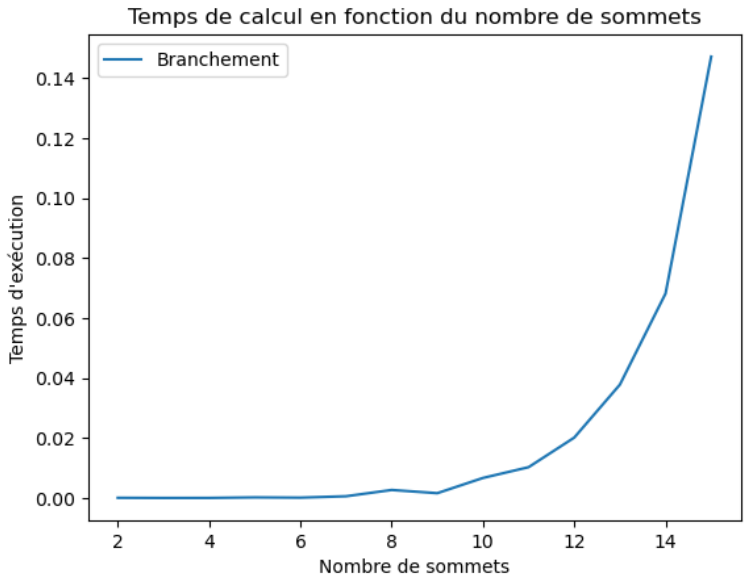


Figure 3.1 – Temps de calcul en fonction du nombre de sommets

On voit sur le graphe que le temps de calcul augmente exponentiellement plus le nombre de sommets augmente, ce qui confirme la complexité théorique de l'algorithme de branchement qui est égale à $O(2^n)$.

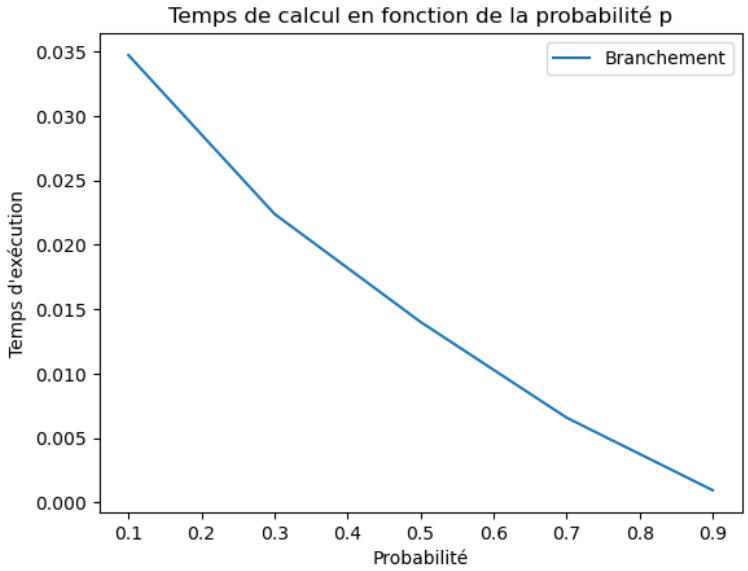


FIGURE 3.2 – Temps de calcul en fonction de la probabilité p

Les résultats obtenus ne nous semblent pas corrects, bien que nous sommes sures du bon fonctionnement de notre code. En effet, nous pensons que le temps de calcul de l'algorithme de branchement devrait augmenter plus la probabilité p augmente, et ce car la suppression d'un sommet entraîne la suppression d'un plus grand nombre d'arêtes, ce qui prendra plus de temps.

3.2 Ajout de bornes et amélioration du branchement

3.2.1 Validité des bornes

1. **Borne $\mathbf{b_1}$:**

   $b_1 = max\{b_1, b_2, b_3\}$

   On veut montrer que $|C| \geqslant b_1$ donc $|C| \geqslant \lceil \frac{m}{\Delta} \rceil$.

   C est la couverture minimale du graphe G. Chaque sommet de C doit couvrir au maximum Δ arêtes.

   De plus, la somme des arêtes couvertes par chaque sommet de C est supérieure ou égale au nombre total d'arêtes : $\sum_{i=0}^{|C|} m_i \geqslant m$ avec m_i, le degré du sommet x_i.

   Alors $|C| \times \Delta \geqslant \sum_{i=0}^{|C|} m_i \geqslant m$ donc $|C| \times \Delta \geqslant m$ car dans les cas où tous les sommets de C ont un degré $= \Delta$ et Δ est le degré max qu'un sommet de la couverture peut avoir donc c'est forcément supérieur ou égale à $\sum_{i=0}^{|C|} m_i$.

   Donc en divisant les deux membres par Δ, on obtient $|C| \geqslant \frac{m}{\Delta}$. Comme $|C|$ est un entier donc : $|C| \geqslant \lceil \frac{m}{\Delta} \rceil$.

2. **Borne $\mathbf{b_2}$:**

$b_2 = max\{b_1, b_2, b_3\}$
Posons $|M| = x$, il existe donc x arêtes de la forme $(\{x_i, x_j\})$ dans l'ensemble M n'ayant pas d'extrémités en commun, tel que tous les x_i et x_j soient différents.
Raisonnons par l'absurde :
Supposons que $|C| < x$. Le graphe G contient alors x arêtes indépendantes, il faudrait alors qu'il y ait au moins x sommets dans la couverture C, donc que $|C| \geqslant x$: contradiction.
Conclusion : $|C| \geqslant |M|$

3.2.2 Amélioration

Nous avons généré trois autres algorithmes de branchements :

1. **Ajout de bornes :** Nous avons ajouté à l'algorithme précèdent l'utilisation de bornes inférieures et supérieures, ainsi que le calcul d'une solution réalisable obtenue avec l'algorithme de couplage. Ces paramètres nous permettront d'effectuer des tests à chaque noeud, qui déterminerons si l'on explore ou pas, le sous-arbre enraciné au noeud courant.
2. **Ajout des voisins :** Dans cet algorithme de branchement, lorsque l'on branche sur une arête (u, v), nous ajoutons les voisins du sommet u à la couverture de la branche droite.
3. **Sommet de degré maximum :** La sélection de l'arête branchée se fait en prenant l'arête qui contient le sommet de degré maximum dans le graphe du noeud courant.

3.3 Temps de calcul des méthodes de branchement

3.3.1 Complexité théorique

Soit un graphe $G = (V, E)$, avec $n = |V|$ le nombre de sommets de G, et $m = |E|$ le nombre d'arêtes.
La complexité des 4 méthodes est de l'ordre de $O(2^n)$, car dans le pire des cas, il faudra explorer toutes les combinaisons possible de couvertures, ce qui nous donnera un arbre de 2^n feuilles. Cependant, cette complexité est rarement atteinte par la méthode de branchement utilisant les voisins, et la méthode de branchement utilisant les sommets de degré maximum.

3.3.2 Complexité expérimentale

Nous avons testé ces 4 méthodes de branchement sur les mêmes graphes en faisant varier le nombre de sommets. Pour cela, nous avons calculé le temps d'exécution ainsi que le nombre de noeuds générés.

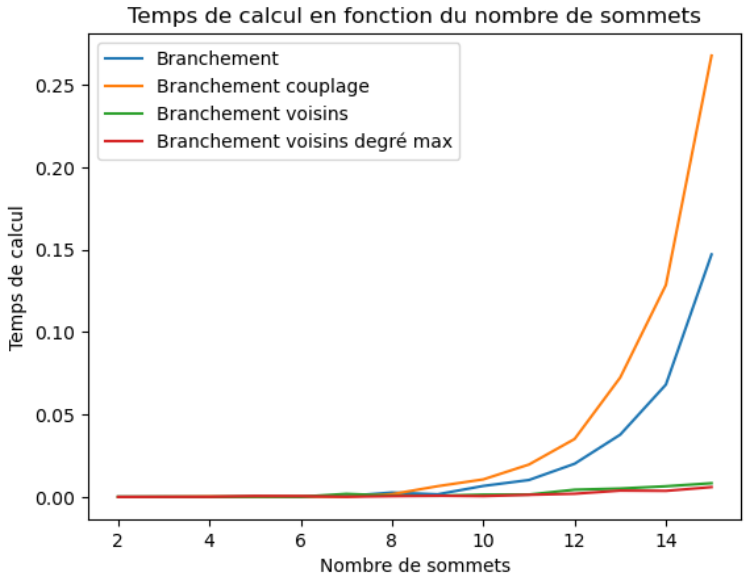


FIGURE 3.3 – Temps de calcul en fonction du nombre de sommets

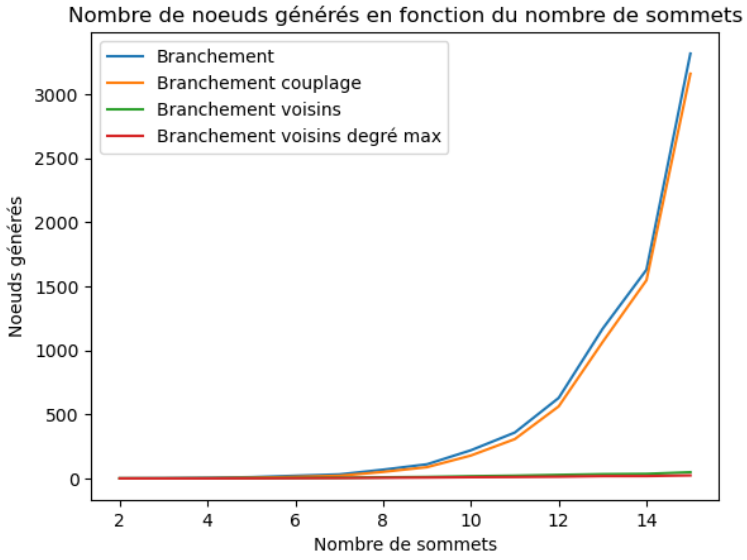


FIGURE 3.4 – Le nombre de noeuds générés en fonction du nombre de sommets

On observe sur les deux graphes que le temps de calcul et le nombre de noeuds générés augmente en temps exponentiel avec le nombre de sommets. La comparaison des quatre algorithmes de branchements montre que les deux derniers (branchements améliorés avec voisins et degré max) présente de bien meilleurs résultats (plus rapide et beaucoup moins de noeuds générés). Comme les deux premiers algorithmes de branchement explorent

beaucoup plus de noeuds donc le temps de calcul est beaucoup plus lent que pour les deux derniers.

3.4 Qualité des algorithmes approchés

Nous avons calculé le rapport d'approximation de l'algorithme de couplage et de l'algorithme glouton pour différentes valeurs de n, avec un $p = 0.5$.
Voici les résultats obtenus :

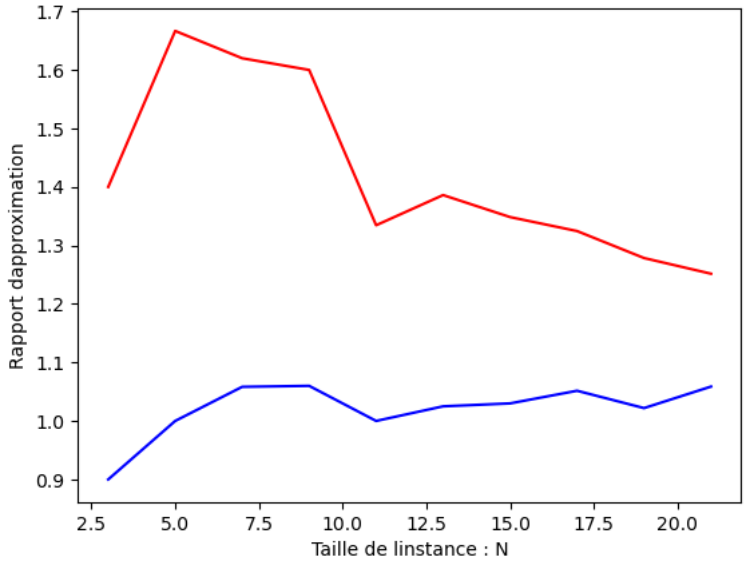


FIGURE 3.5 – Rapports d'approximation en fonction du nombre de sommets

Le pire rapport d'approximation enregistré pour chacun des algorithmes :

1. $r\_couplage = 1.8$
2. $r\_glouton = 1.038$

Nous n'avons pas pu testé les rapports pour un plus grand nombre de sommets car l'exécution prend un temps considérable.

On observe que le rapport d'approximation de l'algorithme glouton est plus ou moins stable est très proche de 1. En revanche, le rapport d'approximation de l'algorithme de couplage diminue avec l'augmentation de n. Plus précisément, il se rapproche de 1. On en conclut que l'algorithme glouton donne des solutions proches de la solution optimale quel que soit le nombre de sommets du graphe. Quant à l'algorithme de couplage, les solutions retournées par celui-ci sont acceptables pour un grand nombre de sommet (>= *qlqchose*). Cependant, le temps de calcul de ces algorithmes augmente considérablement avec l'augmentation de n, il ne serait donc pas intéressant de les exploiter pour des valeurs de n élevées.

Conclusion générale

A travers ce rapport, nous avons effectué une analyse approfondie des méthodes de résolutions pour le problème de couverture minimale (Vertex Cover). En évaluant les performances des algorithmes de couplage et glouton, ainsi que les variantes méthodes de branchement, nous avons observé des comportements distincts en fonction de la taille du graphe et de la probabilité de présence des arêtes. L'analyse des complexités théoriques a été confirmée par des expérimentations pratiques, révélant des tendances de croissance significatives pour le temps de calcul et le nombre de nœuds générés. Les deux premiers algorithmes de branchement (branchement classique et branchement utilisant les bornes) sont coûteux en terme de temps et leur temps de calcul croissent exponentiellement avec l'augmentation de la taille du graphe. Quant aux deux autres méthodes de branchement (branchement utilisant les voisins et branchement utilisant le degré max), ces dernières retournent une solution optimale de manière significativement plus rapide.

En ce qui concerne la qualité des solutions, l'algorithme glouton a généralement fourni des résultats plus proches de l'optimum par rapport à l'algorithme de couplage. Cependant, pour des graphes de grande taille, le rapport d'approximation de l'algorithme de couplage a montré une convergence vers 1 lorsque le nombre de sommets augmente, et celui de l'algorithme glouton est toujours au voisinage de 1, soulignant l'importance de considérer attentivement les caractéristiques spécifiques des graphes étudiés lors de la sélection des méthodes.

En résumé, la compréhension approfondie des performances et des limites de chaque algorithme constitue un aspect crucial pour l'application efficace de ces méthodes dans des contextes pratiques. La sélection de l'algorithme approprié dépendra largement des propriétés spécifiques du problème en question, y compris la taille du graphe, la densité des arêtes et les contraintes temporelles. Ainsi, ce rapport fournit des informations utiles pour guider les choix d'approches dans des scénarios réels.