Faculté des Sciences et Ingénierie - Sorbonne université

Master Informatique parcours - DAC / IMA

COMPLEX - Complexité, algorithmes randomisés et approchés

Rapport de projet

Couverture de graphe

Réalisé par :
SAID Faten Racha - M1 DAC
MAOUCHE Mounir - M1 IMA

Supervisé par :
Thomas Bellitto
Fanny Pascual

Octobre 2023

Table des matières

Table des figures

Introduction

Une couverture par sommets (Vertex Cover) d'un graphe est un sous-ensemble V de ses sommets tel que chaque arète du graphe possède au moins une de ses extrémités dans l'ensemble V.

FIGURE 1 – Exemples de couvertures par sommets.

Le problème de couverture minimale par sommets est un problème classique en optimisation informatique qui consiste à chercher, pour un graphe donné, le plus petit ensemble possible de sommets qui constitue une couverture de ses arètes.

Ce problème a la faculté d'être NP-complet, il est donc impossible, selon l'état technologique et intellectuel actuel, de résoudre ce problème en temps raisonnable pour des graphes de grande taille.

Afin de pallier ce problème, des méthodes ont été mises en place afin d'arriver à une solution approchée mais plus rapide à trouver, ou bien de chercher un moyen d'arriver à la solution optimale en un minimum de temps et/ou de ressources.

Nous allons donc procéder dans ce projet à l'étude de ces deux types d'approches, et évaluer leurs résultats en terme de qualité et de temps et de ressources de calcul.

Chapitre 1

Graphes

1.1 Modélisation

La partie algorithmique de ce projet a été implémentée en Python en utilisant les structures de données natives du langage. Nos graphes seront représentés sous forme de listes des sommets et des arêtes, ce qui nous a permis de faciliter certains calculs notamment au niveau des branchements.

1.2 Déroulement des tests

Lors de notre conception et analyse des algorithmes approchés et exacts, nous avons mis en place des testw selon la méthode suivante :

— Déterminer Nmax la taille maximale du graphe (i.e son nombre de sommets) pour laquelle l'algorithme tourne rapidement sur notre machine.

— Exécuter les algorithmes sur un ensemble $\{\frac{Nmax}{10}, \frac{2*Nmax}{10}, ..., Nmax\}$ de tailles d'instances.

— Pour chaque taille d'instance, plusieurs graphes sont générés aléatoirement et la moyenne de leurs résultats d'évaluation est comptabilisée.

— Nous ferons varier quelques fois la densité d'arêtes dans le graphe, et nous opterons dans d'autres cas pour une densité inversement proportionnelle au nombre de sommets du graphe afin d'avoir des graphes moins denses.

— Les graphiques représenteront principalement l'évolution du temps de calcul des algorithmes, leur coût en ressources mémoire, et la qualité des algorithmes approchés, en fonction de la taille et la densité en arêtes du graphe analysé.

Chapitre 2

Méthodes approchées

2.1 Méthode du couplage maximal

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

Le principe de cette méthode est de prendre une arête E_i quelconque du graphe. Si aucune de ses deux extrémités n'est dans l'ensemble constituant la solution courante, alors on les ajoute et on recommence pour le graphe amputé de l'arête E_i.

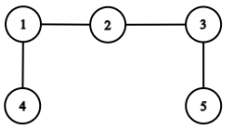
Cet algorithme est 2-approché. C'est à dire qu'il nous retournera au pire des cas une solution 2 fois supérieure à la solution optimale.

2.2 Algorithme glouton

Cette méthode-ci consiste à prendre un sommet V de degré maximum dans le graphe, l'ajouter à l'ensemble solution, et réitérer sur le graphe amputé du sommet V, et ce jusqu'à ce que l'ensemble créé constitue une couverture.

Cet algorithme n'est pas optimal. En effet, il est possible de trouver un graphe pour lequel la solution retournée sera supérieure à la solution optimale.

Exemple :



Sur cette instance de départ, l'algorithme aura le choix entre plusieurs sommets (1, 2 et 3) qui ont un degré maximum. Comme il n'y a aucun restriction à ce niveau-là, il est tout à fait possible que le programme sélectionne le sommet 2 en premier, ce qui nous donnera un graphe comme celui-ci :

Maintenant, l'algorithme est obligé de prendre un sommet de chaque sous-graphe, ce qui nous donnera une solution de taille 3 (par exemple l'ensemble $\{2, 1, 3\}$)

Or, il est évident que la solution optimale doit être de taille 2, et ce en prenant les sommets 1 et 3. Ceci prouve que l'algorithme n'est pas optimal.

En conséquence de cela, nous pouvons affirmer que l'algorithme est au moins $\frac{3}{2}$ approché, et qu'il n'est donc pas r-approché pour tout r $< \frac{3}{2}$, car pour tout r dans cet intervalle de valeurs, on trouvera comme contre-exemple l'instance précédente.

2.3 Comparaison des deux méthodes

Comparaison des temps de calcul

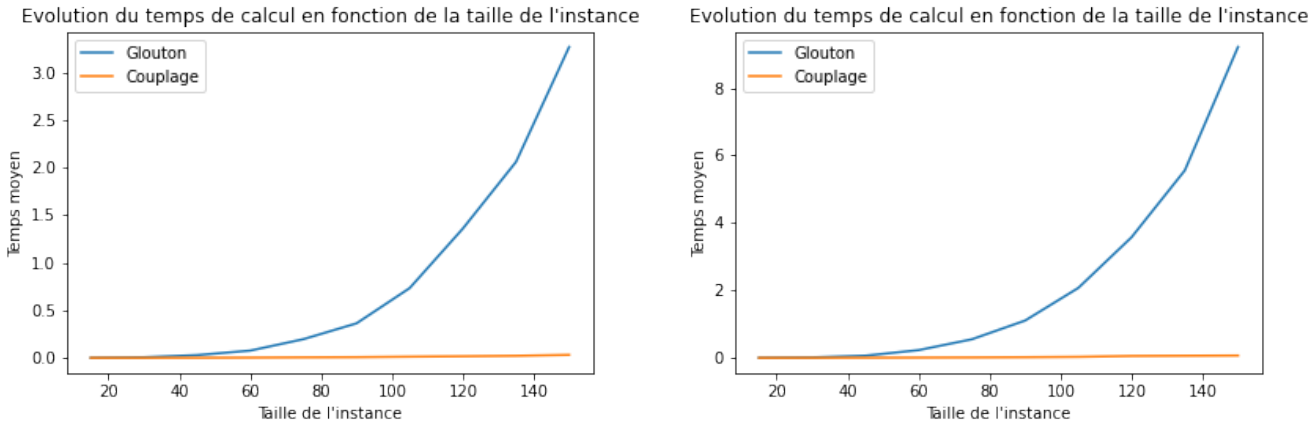


FIGURE 2.1 – Temps d'exécution des algorithmes en fonction de n avec p=0.2 et p=0.7

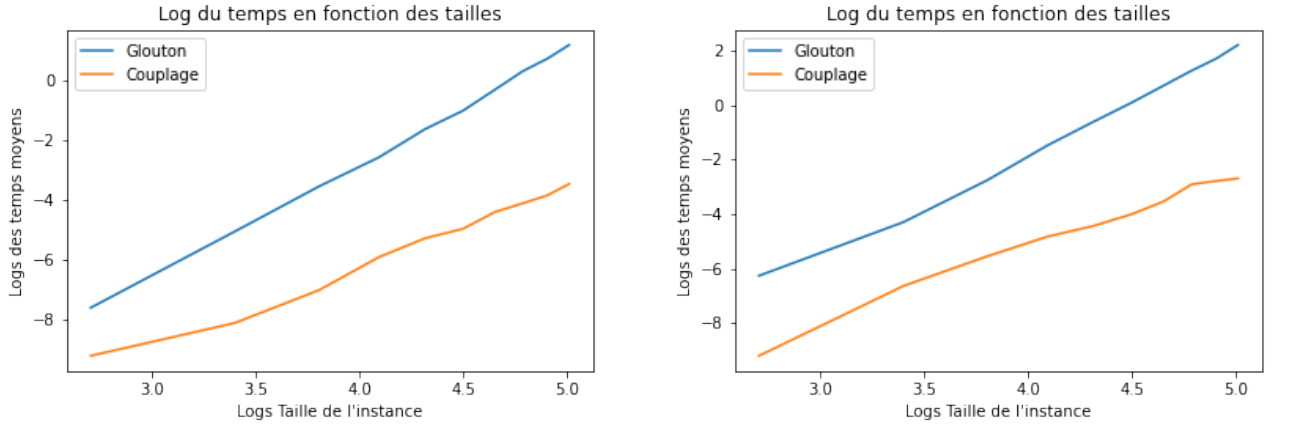


FIGURE 2.2 – Complexité des algorithmes avec p=0.2 et p=0.7

Nous pouvons voir à travers ces graphes et selon nos calculs que l'algorithme Glouton possède une complexité polynomiale supérieure à celle de Couplage, $O(n^3)$ vs $O(n^2)$. Il croît beaucoup plus vite pour instances de grandes tailles.

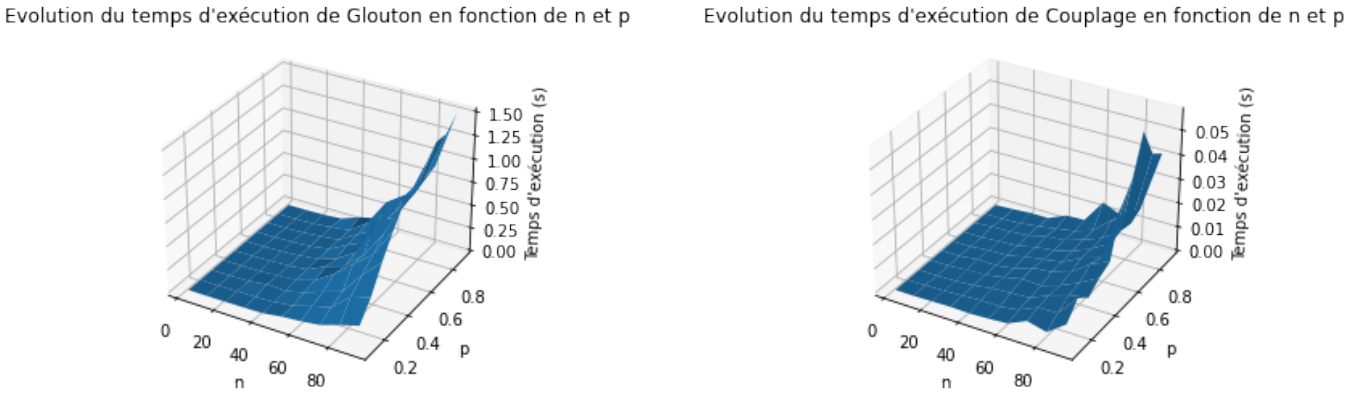


FIGURE 2.3 – Temps d'exécution de Glouton et Couplage en fonction de n et p

Comparaison de la qualité des solutions

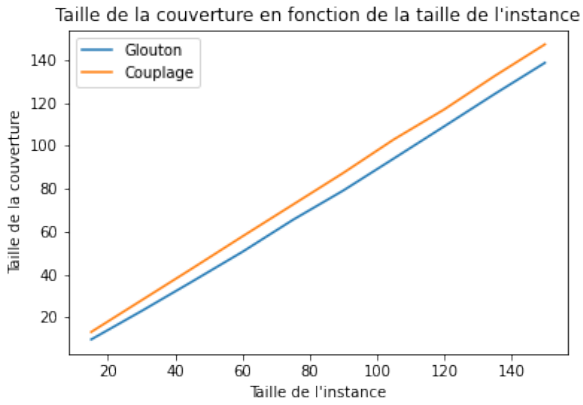


FIGURE 2.4 – Qualite de Couplage et Glouton avec p=0.2

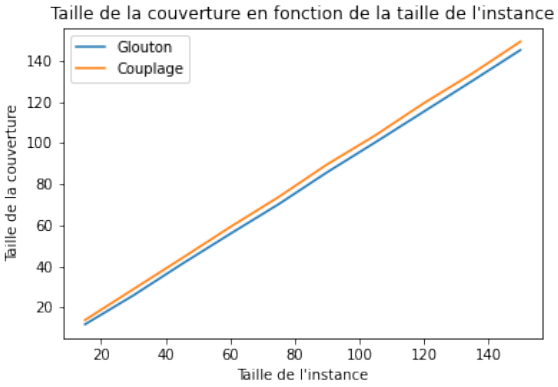


FIGURE 2.5 – Qualite de Couplage et Glouton avec p=0.5

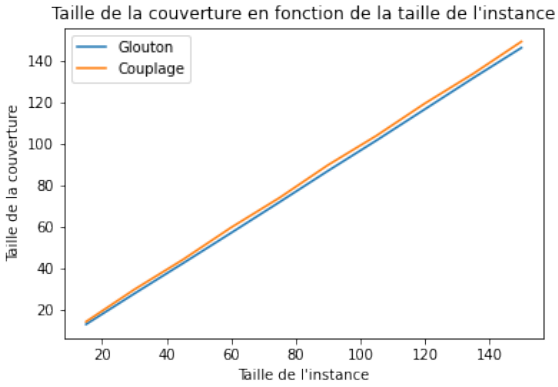


FIGURE 2.6 – Qualite de Couplage et Glouton avec p=0.7

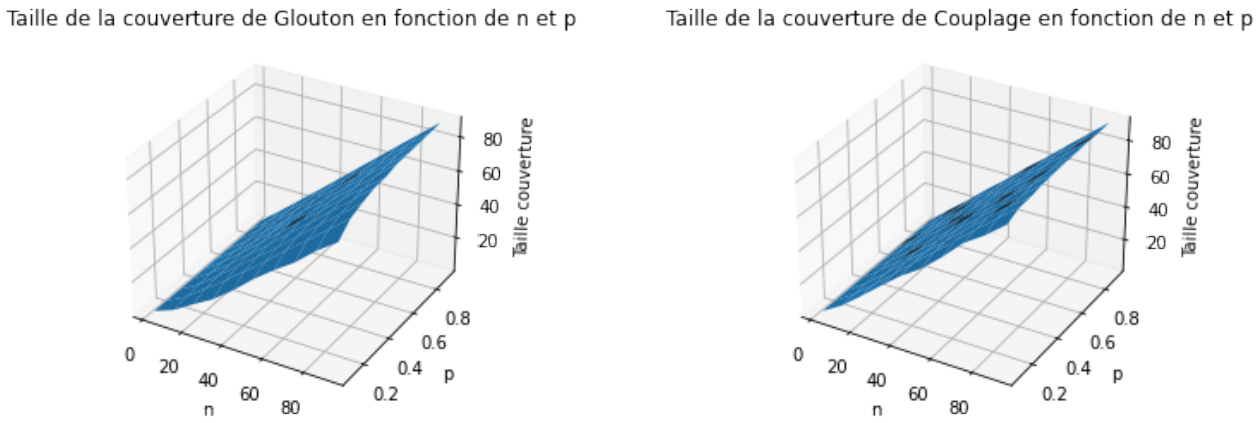


FIGURE 2.7 – Tailles des couvertures en fonction de n et de p

Nous pouvons voir à travers ces graphes que parmi les deux algorithmes c'est Glouton qui retourne de meilleures solutions, mais l'écart entre les deux diminue significativement à mesure que p augmente, c'est à dire que la densité du graphe en arêtes augmente. Cela dû au fait que la qualité de Couplage s'améliore plus le graphe est dense ; car éliminer deux sommets d'un graphe complet retire plus d'arêtes que dans un graphe moins dense.

Séparation et évaluation

Nous allons dans ce qui suit tester les fonctions basées sur des algorithmes de séparation et évaluation (Branch and bound), en partant d'une version "basique" vers une versions plus évoluée.

3.1 Branchement

Evaluation du temps de calcul

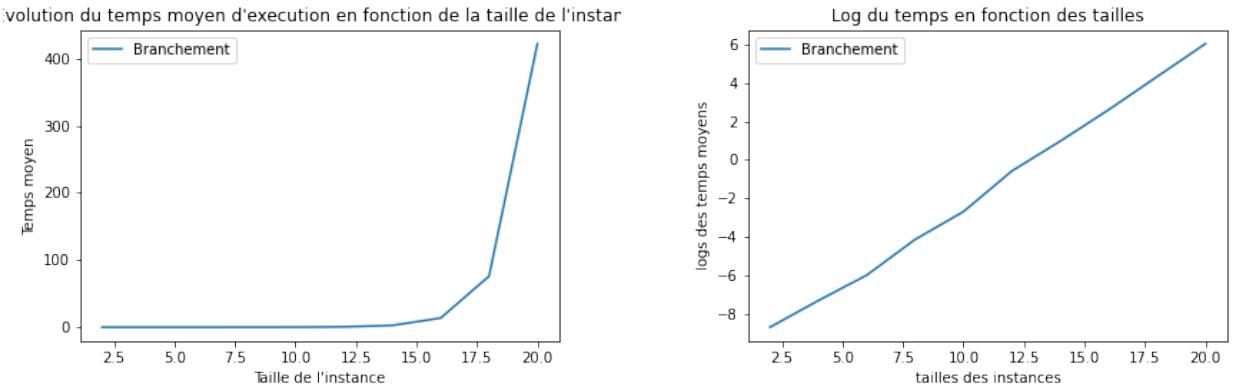


Figure 3.1 – Temps d'execution de Branchement simple avec $p = \frac{1}{\sqrt{n}}$.

La fonction Branchement simple est très inefficace en terme de temps de calcul et a une complexité en $O(2.3^n)$.

Evaluation du nombre de noeuds générés

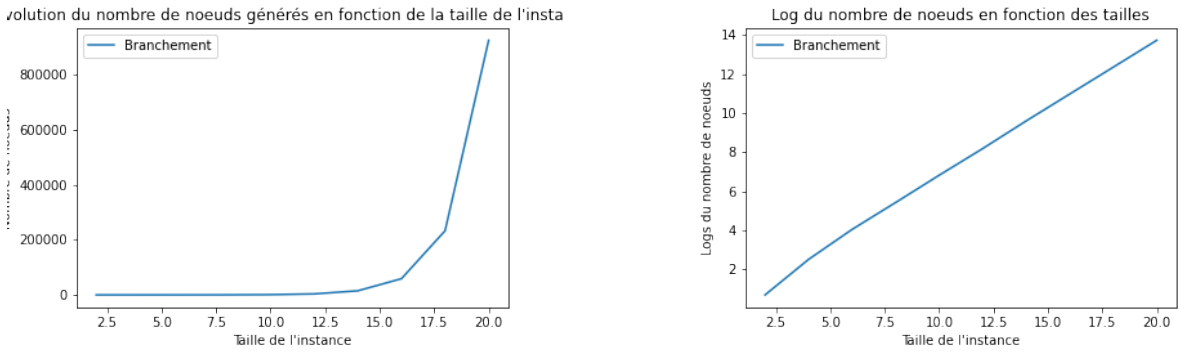


Figure 3.2 – Noeuds générés par Branchement simple avec $p = \frac{1}{\sqrt{n}}$.

Ici, nous voyons pourquoi la fonction met beaucoup de temps à s'exécuter, car il apparaît qu'elle possède une grande complexité spaciale ($O(2^n)$) .

En effet, cette méthode, devant retourner la solution optimale, doit parcourir toutes les possibilités et donc créer un nombre exponentiel de noeuds.

3.2 Ajout des bornes

Soit G un graphe, M un couplage de G et C une couverture de G. Nous exploitons la propriété suivante :

$$|C| \geq max\{\text{b1, b2, b3}\}$$

avec $b1 = \lceil \frac{\Delta}{m} \rceil$ (où Δ est le degré maximum des sommets du graphe), $b2 = |\text{M}|$, et $b3 = \frac{2n-1-\sqrt{(2n-1)^2-8m}}{2}$

Il est possible de prouver ces bornes comme suit :

Preuve de $b1$

Nous savons que $|C|$ est un nombre suffisant de sommets pour couvrir toutes les arêtes du graphe. Chaque sommet de C couvre entre 1 et Δ arêtes du graphe. Donc au maximum chaque sommet de C couvre Δ arêtes. Ce qui nous donne $|\text{C}|*\Delta \geq \text{m}$

Notons que le seul cas où chaque sommmet de C couvre exactement Δ arêtes, soit $|C|*\Delta = m$, est celui où $|C| = 1$. Autrement, avec cette inégalité, on est sûr qu'au moins une arête est couverte par plus d'un sommet. Et comme $|C|$ est un nombre entier, alors nous avons forcément : $|C| \geq \lceil \frac{m}{\Delta} \rceil$

Preuve de $b2$

Soient $|M|$ le nombre d'arêtes du couplage, et $S = 2*|M|$ le nombre de sommets du couplage (car par définition du couplage, ses arêtes sont non-adjacentes).

On sait que couplage retourne toujours au plus $2 * opt$ car il est 2-approché.
Donc : $\text{S} = 2 * |M| \leq 2*\text{opt} \rightarrow 2*|\text{M}| \leq 2*|\text{C}| \rightarrow |\text{M}| \leq |\text{C}|$

Preuve de $b3$

Pour prouver cette borne, nous utiliserons le nombre maximal d'arêtes d'un graphe dont C est une couverture. Posons $x = |C|$, $n =$ nombre de sommets du graphe.

nous pouvons distinguer deux cas :

— Les arètes dont les deux extrémités sont dans C : il y a en a au maximum $\frac{x(x-1)}{2}$, qui est le nombre d'arètes dans un graphe complet à x sommets

— Les arètes dont seule une extrémité est dans C : il y en a au maximum $x(n-x)$ car chaque sommet de C est relié aux $n-x$ autres sommets qui n'y appartiennent pas.

Nous trouvons alors que :

$$\frac{x(x-1)}{2} + x(n-x) \geq m \tag{3.1}$$

ce qui nous donne après simplification :

$$x^2 - x(2n-1)x + 2m \leq 0 \tag{3.2}$$

En supposant l'égalité et en résolvant l'équation de second degré on trouve :

$$x = \frac{2n - 1 \pm \sqrt{(2n-1)^2 - 8m}}{2} \geq b_3 \tag{3.3}$$

Ce qui prouve que $|C|$ ≥b3

Conclusion : Puisque $|C|$ est supérieur aux bornes $b1$, $b2$, $b3$ prises individuellement, alors il est supérieur à leur maximum.

Evaluation du temps de calcul

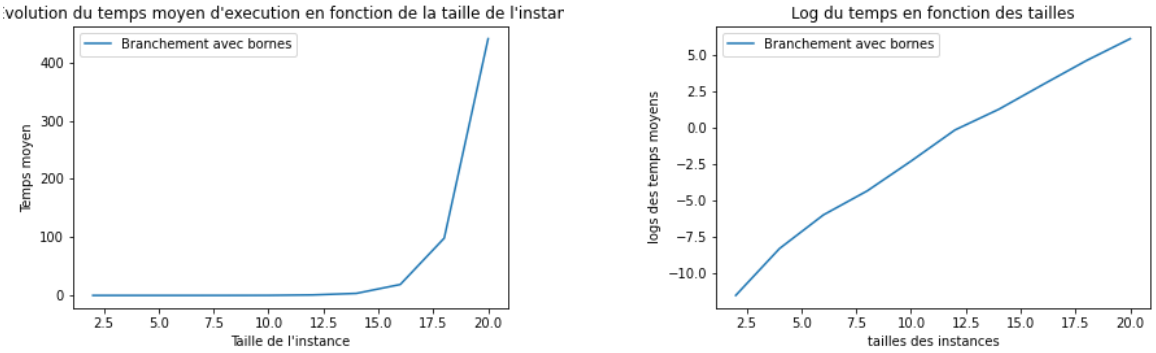


FIGURE 3.3 – Temps de calcul et complexité de Branchement Avec Bornes avec $p = \frac{1}{\sqrt{n}}$

Evaluation du nombre de noeuds générés

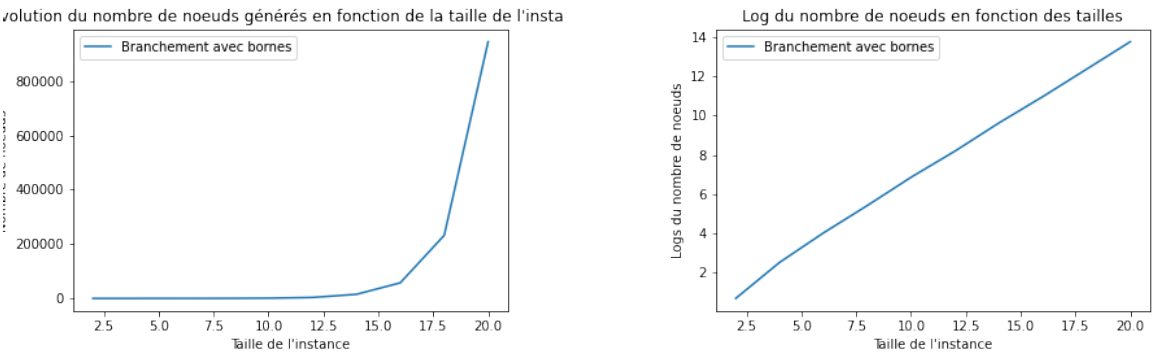


FIGURE 3.4 – Noeuds créés par Branchement avec Bornes et complexité spaciale avec $p = \frac{1}{\sqrt{n}}$.

La complexité temporelle de cette fonction est en $O(2.2^n)$, et sa complexité spatiale est quant à elle en $O(2^n)$

Nous constatons que l'algorithme Branchement avec ajout des bornes ne réalise pas de meilleures performances que son prédécesseur, nous pouvons supposer que ses critères ne sont pas assez discriminants pour permettre d'élaguer un nombre significatif de branches par rapport à la taille de l'arbre créé.

3.3 Amélioration du branchement

3.3.1 Branchement amélioré

Evaluation du temps de calcul

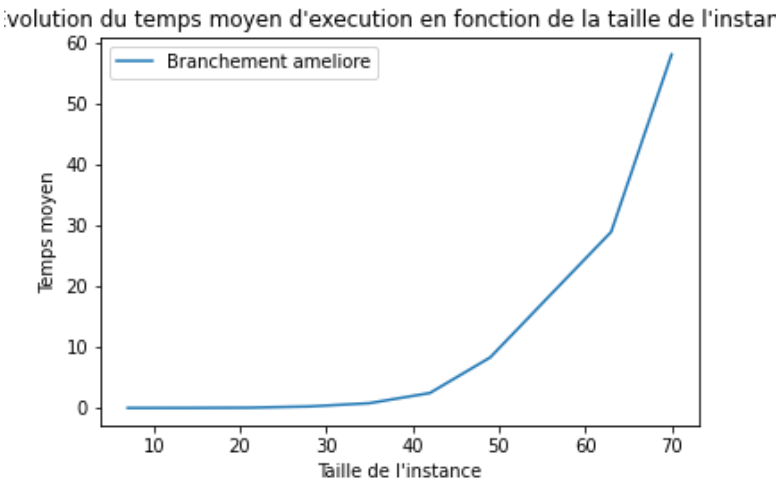


FIGURE 3.5 – Temps d'exécution de Branchement Ameliorée $p = \frac{1}{\sqrt{n}}$

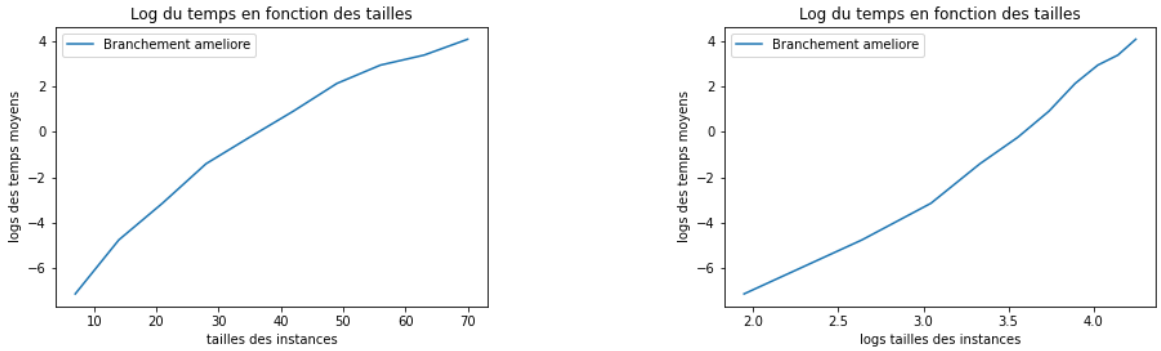


FIGURE 3.6 – Complexité de Branchement Améliorée avec $p = \frac{1}{\sqrt{n}}$.

La complexité est incertaine entre exponentielle en $O(1.2^n)$ ou polynomiale en $O(n^5)$, mais il semble plus correcte de dire que sa complexite est exponentielle qui tend vers un temps polynomial. Dans tous les cas, cela représente une amélioration significative par rapport aux deux précedents algorithmes de branchement.

Evaluation du nombre de noeuds générés

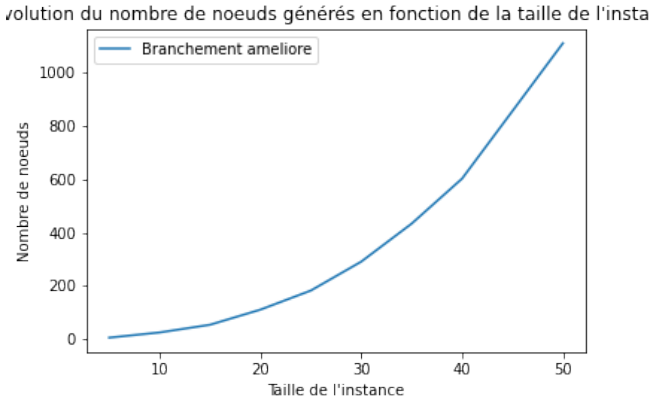


FIGURE 3.7 – Noeuds générés par Branchement Améliorée avec $p = \frac{1}{\sqrt{n}}$

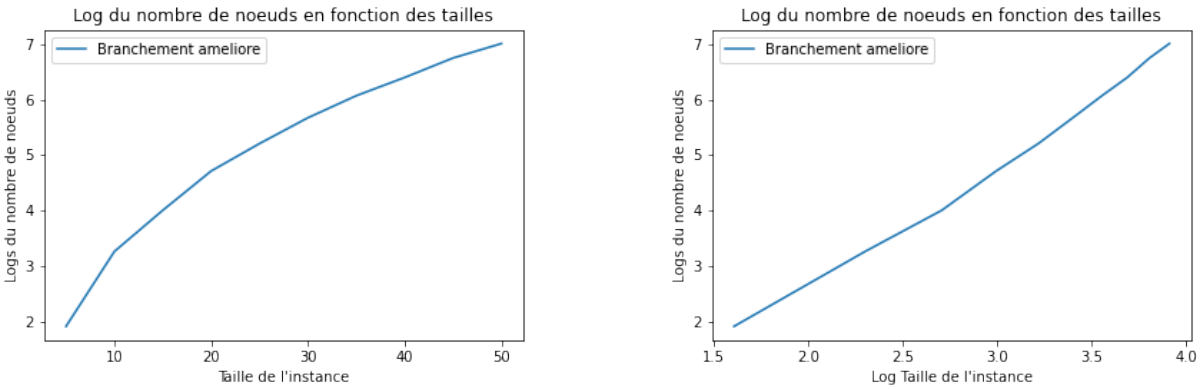


FIGURE 3.8 – Complexité spatiale de Branchement Améliorée avec P= $\frac{1}{\sqrt{n}}$.

Nous remarquons que la complexité spatiale est passée de $O(2^n)$ avec les algorithmes précédents à $O(n^{2.2})$ soit d'une complexité exponentielle à polynomiale.

Les résultats montrent que cette fonction est effectivement une amélioration, car elle inclut la notion de voisinage qui permet de construire une solution non pas sommet par sommet mais en ajoutant plusieurs sommets à la fois.

3.3.2 Branchement Finale

Evaluation du temps de calcul

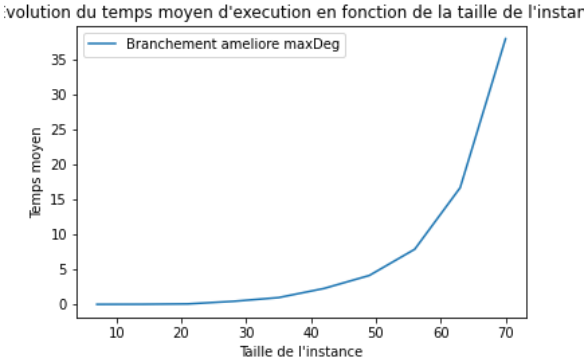


FIGURE 3.9 – Temps d'exécution de Branchement Finale avec $p = \frac{1}{\sqrt{n}}$.

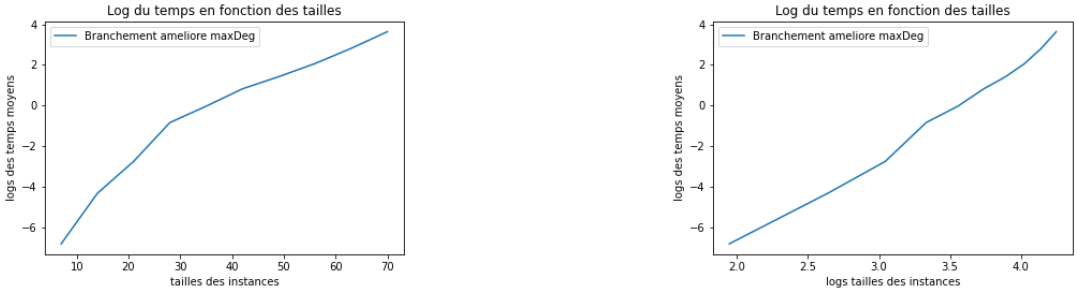


FIGURE 3.10 – Complexité de Branchement Finale avec P= $\frac{1}{\sqrt{n}}$

La complexite est incertaine entre exponentielle en $O(1.15^n)$ ou polynomiale en $O(n^{4.5})$, mais il semble plus correcte de dire que sa complexite est exponentielle mais tend vers un temps polynomial.

Evaluation du nombre de noeuds générés

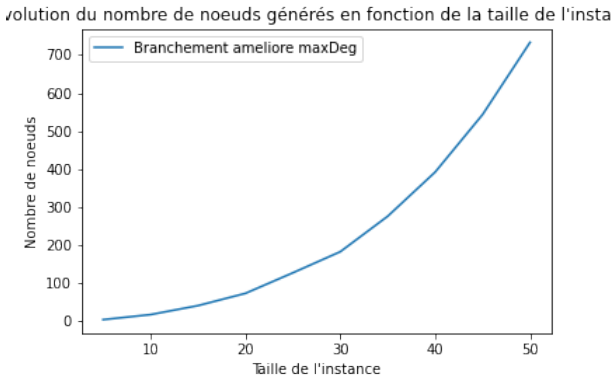


FIGURE 3.11 – Nombre de noeuds générés par Branchement Finale avec $p = \frac{1}{\sqrt{n}}$

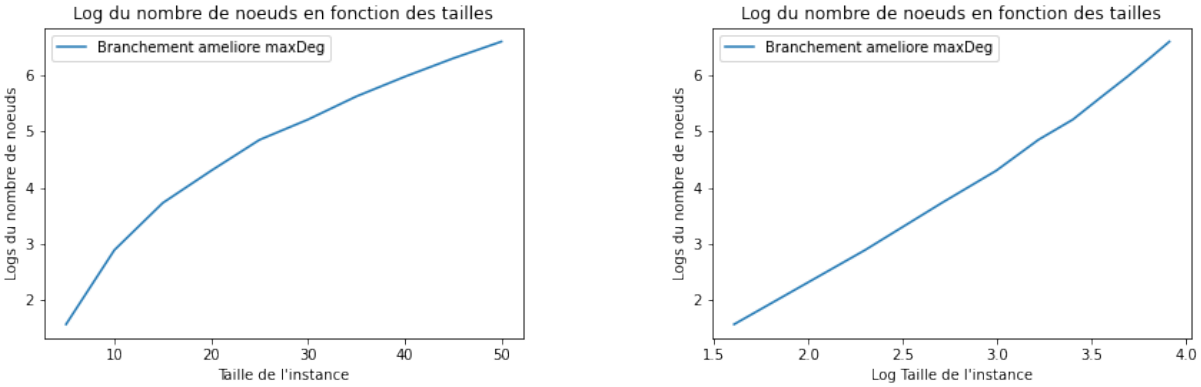


FIGURE 3.12 – Complexité spatiale de Branchement Finale avec P= $\frac{1}{\sqrt{n}}$

Nous remarquons une légère amélioration par rapport à la fonction précédente et conserve une complexité polynomiale en $O(n^{2.1})$.

Ces deux évaluations nous permettent de dire que Branchement Finale est plus performante que sa version précédent. Cela vient du fait que commencer par sélectionner un sommet de degré maximum supprime un nombre maximum d'arêtes à la fois ce qui permet de converger plus rapidement vers une solution.

3.4 Qualité des algorithmes approchés

Dans cette section nous allons évaluer le rapport d'approximation des algorithmes approchés vus plus haut, en comparant les rapports respectifs de leurs résultats avec l'Optimale, donnée par la fonction Branchement Finale compte tenu qu'elle est la plus rapide et retourne des résultats exacts.

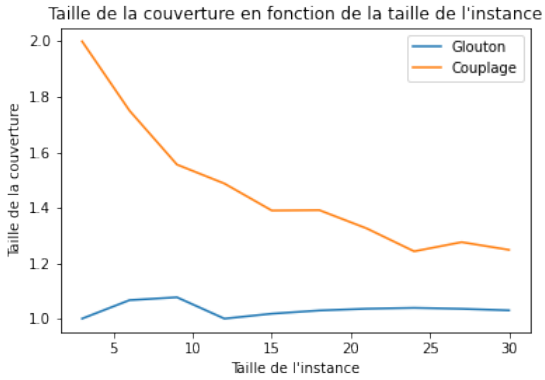


FIGURE 3.13 – Rapport d'approximation de Glouton et Couplage avec p=0.2

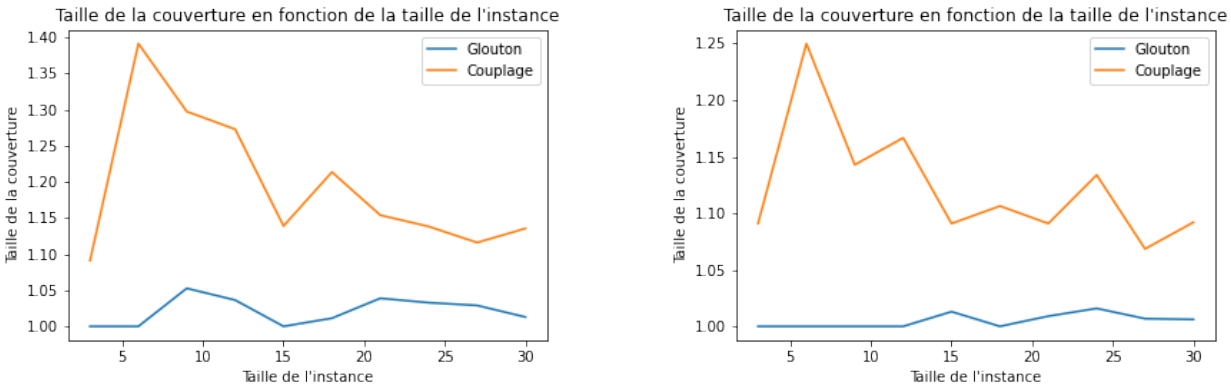


FIGURE 3.14 – Rapport d'approximation de Glouton et Couplage avec p=0.5 et p=0.7

Pire rapport d'approximation de Glouton : 1.08 (obtenu avec $p = 0.2$)
Pire rapport d'approximation de Couplage : 2 (obtenu avec $p = 0.2$)

Les graphes montrent que l'algorithme Glouton retourne toujours de meilleures solutions que Couplage, et son taux d'erreur fluctue moins. Aussi, la solution retournée par Couplage s'améliore proportionnellement à la densité des graphes, ce qui est en accord avec les résultas obtenus sur les figures 2.4 à 2.6 et les conclusions émises à leur sujet.

D'un autre coté, les résultats vérifient bien le fait que l'algorithme Couplage est 2-approché puisque son erreur n'a jamais dépassé 2 fois la solution optimale. Quant à Glouton, il n'a lui aussi pas dépassé une erreur de 2 mais cela ne garantit pas un rapport d'approximation inférieur à 2, car les graphes susceptibles de provoquer cela sont des cas particuliers qui ne sont pas apparus lors de la génération aléatoire.

3.5 Proposition et évaluation d'heuristiques

Principe :

Cette méthode repose sur l'observation que la méthode de branchement utilisée analyse deux fois les mêmes ensembles de sommets. Afin de pallier ce problème, nous avons proposé d'ajouter une vérification lors de l'empilement des sommets dans la pile afin de ne pas parcourir deux fois la même solution.

Evaluation du temps d'exécution :

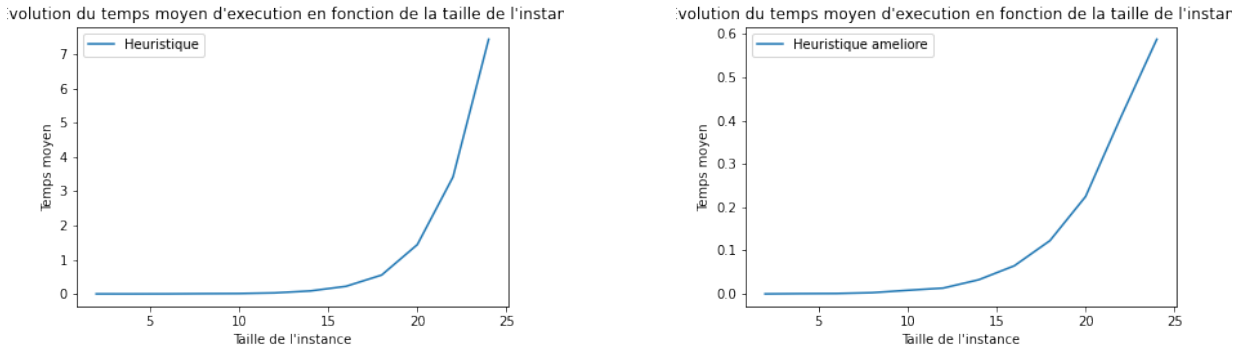


FIGURE 3.15 – Temps d'exécution de l'Heuristique et l'Heuristique Améliorée avec $p = \frac{1}{\sqrt{n}}$

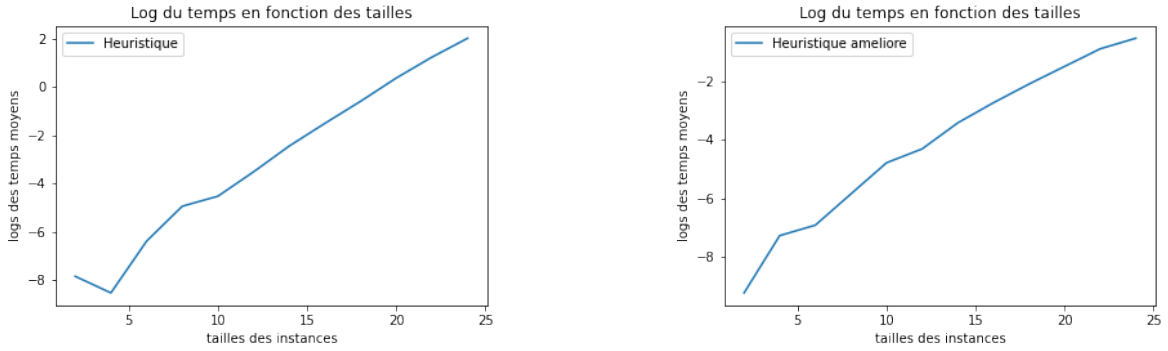


FIGURE 3.16 – Complexité de l'Heuristique et l'Heuristique Améliorée avec $p = \frac{1}{\sqrt{n}}$

Nos deux heuristiques ont des complexités respectives de $O(1.7^n)$ et $O(1.4^n)$.

Evaluation du nombre de noeuds créés :

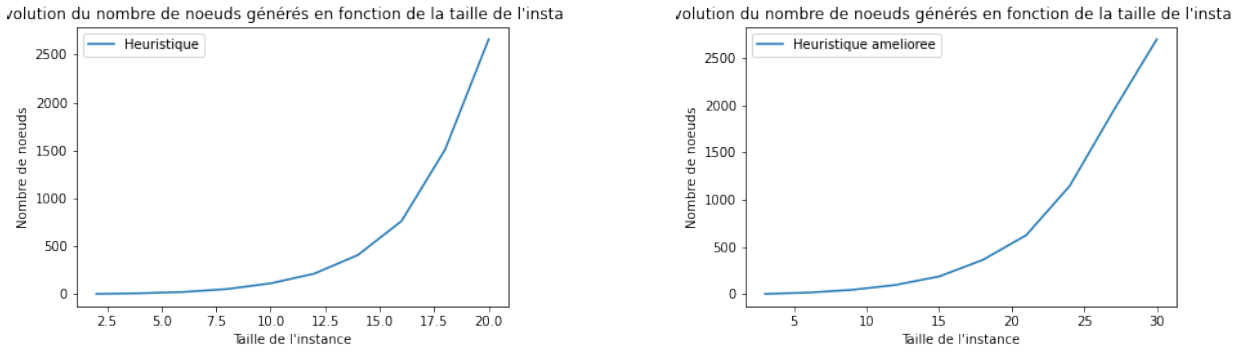


FIGURE 3.17 – Nombre de noeuds créés par les Heuristiques avec $p = \frac{1}{\sqrt{n}}$.

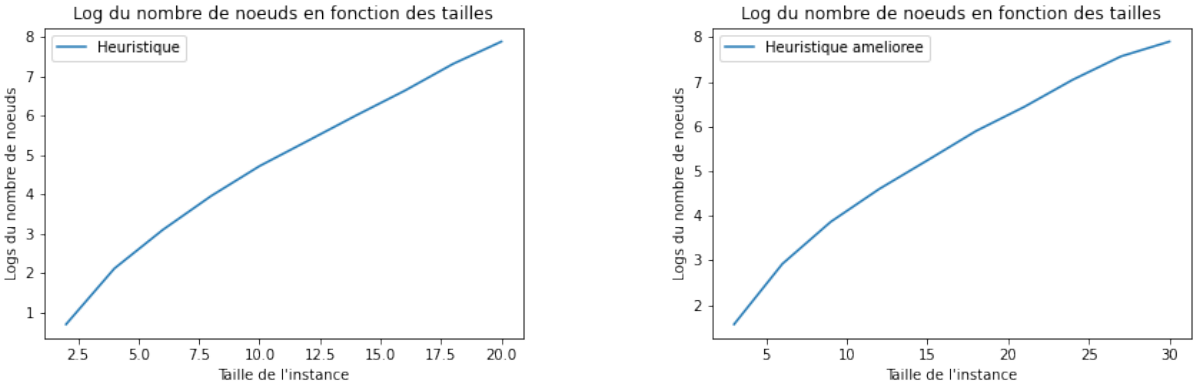


FIGURE 3.18 – Complexité spatiale des Heuristiques avec $p = \frac{1}{\sqrt{n}}$

Ici, nous avons complexités spatiales respectives de $O(1.4^n)$ et $O(1.2^n)$.
Comme il est possible de le constater, cette approche a retourné des résultats concluants quant au temps d'exécution ainsi que le nombre de noeuds créés.

Conclusion générale

L'élaboration de ce projet nous a permis de constater à quel point la résolution d'un problème NP-Complet comme le Vertex Cover est très gourmande en temps et en ressources. Ce coût peut cependant être réduit en utilisant des méthodes approchées, plus rapides mais au dépens de la précision.

Plus encore, la solution optimale d'un problème combinatoire peut être atteinte en temps réduit mais toujours exponentiel, en utilisant des moyens pour juger si un certain cheminement peut aboutir à une solution ou pas, ce qui permet d'écourter l'analyse, comme illustré lors de l'introduction des bornes et l'amélioration du branchement.