# COUVERTURE DE GRAPHE

Jérémy DUFOURMANTELLE et Antoine TOULLALAN

Octobre 2021

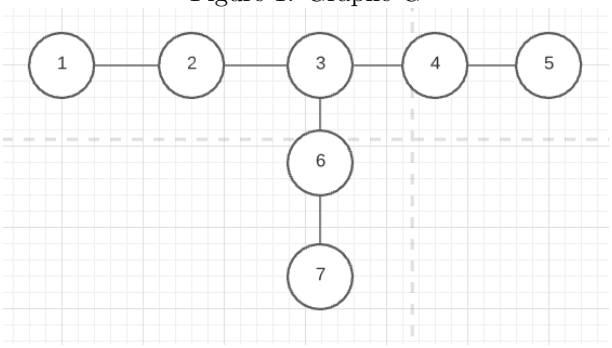
## 1 Graphes

On écrit notre code en python. Ainsi on représente les graphes sous la forme d'un tuple (V,E) où V est un set qui contient les identifiants entiers des sommets et E est un set contenant des tuples de la forme (a,b) représentant les arêtes, et où a et b sont les extrémités de l'arête.

## 2 Méthodes approchées

**1** Soit G le graphe suivant :

Figure 1: Graphe G




L'algorithme glouton renvoie la solution {3,2,4,6} de taille 4 qui est bien une couverture. Mais la solution optimale est {2,4,6} de taille 3.

$$val(algo\_glouton(G)) \leq \alpha * opt(G)$$

$$\alpha = 4/3$$

L'algorithme glouton n'est donc pas optimal et il n'est pas r-approché pour $r \leq 4/3$.

2. Pour l'algorithme de couplage, $Nmax \approx 120$ (avec un temps de calcul=2s). Pour l'algorithme glouton, $Nmax \approx 500$ (avec un temps de calcul=2s). On observe les temps de calcul des algorithmes en fonction de la taille de l'entrée et de p

Figure 2:

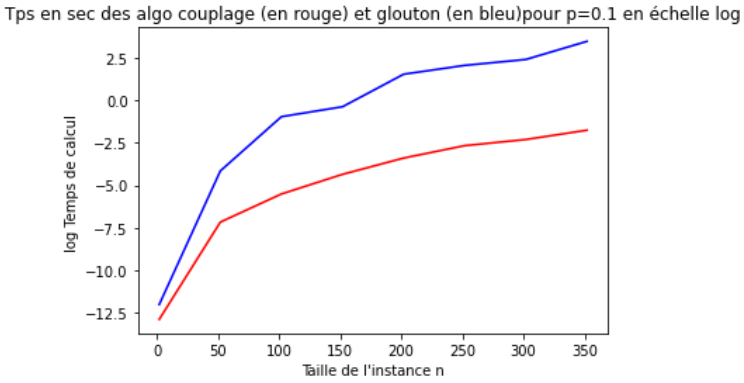
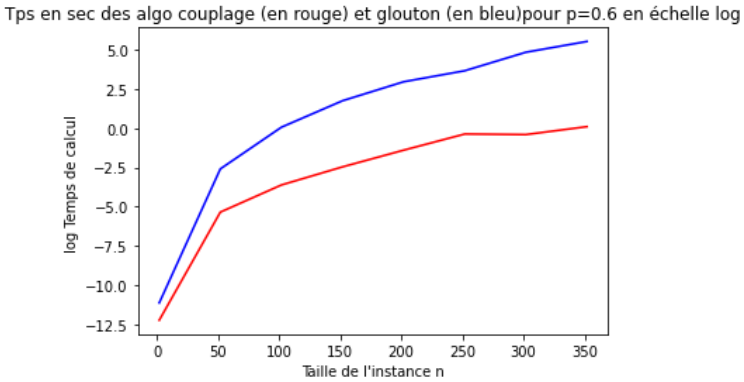



Figure 3:




D'après les figure 2 et 3, on visualise les courbes des 2 algorithmes avec le temps de calcul en fonction de log(n) avec n le nombre de sommets du graphe en entrée.

On voit que les 2 algorithmes ont une complexité polynomiale car les courbes sont concaves.

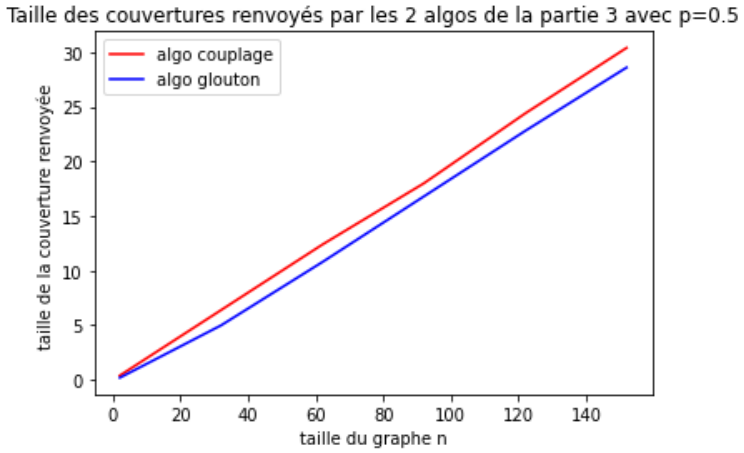
Le temps de calcul de l'algorithme glouton est très supérieure à celui de l'algorithme de couplage pour N grand (N>200). Lorsqu'on augmente p, on voit que le temps de calcul des 2 algorithmes augmente car il y a plus d'arêtes à traiter.

D'après nos fonctions de test, les 2 algorithmes renvoient bien des solutions valides. On a vu que les résultats de l'algorithme glouton ne sont pas tous optimaux et les résultats de l'algorithme couplage aussi ne sont pas tous optimaux. Par exemple avec un graphe à 2 sommets $s_1$ et $s_2$ reliés par une unique arête, c'est à dire de la forme : ($s_1$,$s_2$) , la solution optimale a une taille de 1 (on prend $s_1$ ou $s_2$) mais l'algorithme de couplage renvoie une solution de taille 2 ($s_1$ et $s_2$).

Donc les algorithmes renvoient des solutions valides mais pas optimales. L'algo glouton a un temps de calcul très supérieur à l'algo de couplage pour N grand.

On regarde la taille des couvertures renvoyées par les 2 algorithmes en fonction de la taille du graphe d'entrée avec le graphe de la figure 4.

Figure 4:




- On voit que l'algorithme de couplage semble renvoyer une couverture de taille supérieur à celle renvoyée par l'algorithme glouton. En effet, dans l'algorithme de couplage, à chaque arêtes qui n'a aucune de ses 2 extrémités dans l'ensemble C qui est la couverture "en construction", or une seule extrémité pourrait être ajoutée car une couverture est un ensemble de sommets C tel que chaque arête a au moins un sommet dans C. A l'inverse, dans l'algorithme glouton, chaque sommet qu'on ajoute est nécessaire pour avoir une couverture car on ajoute le sommet de degré maimal à chaque itération. Donc la taille de la couverture est de taille au plus égale à la couverture renvoyée par l'algo couplage

Donc l'algorithme glouton renvoit de meilleurs solutions que l'algorithme de couplage.

**3** On suppose que l'algorithme glouton est r-approché avec r>1 constant. On montre qu'il existe un graphe tel que la solution de taille N de l'algorithme

glouton et la solution optimale est de taille OPT tel que N/OPT>r.

# 3 Séparation et évaluation

## 3.1 Branchement

]

D'après les figures 5 et 6, nous voyons que la courbe de temps de calcul en fonction du log de la taille de l'entrée a la forme d'une droite pour différentes probabilité p (on rappelle que p est la probabilité de présence d'arêtes).On en déduit que l'algorithme BranchAndBound a une complexité exponentielle. Cette droite a un coefficient directeur d'environ 2/5, or $10^{2/5} \approx 2.5$, donc la complexité de l'algorithme branchAndBound est $0(3^n)$.

Figure 5:

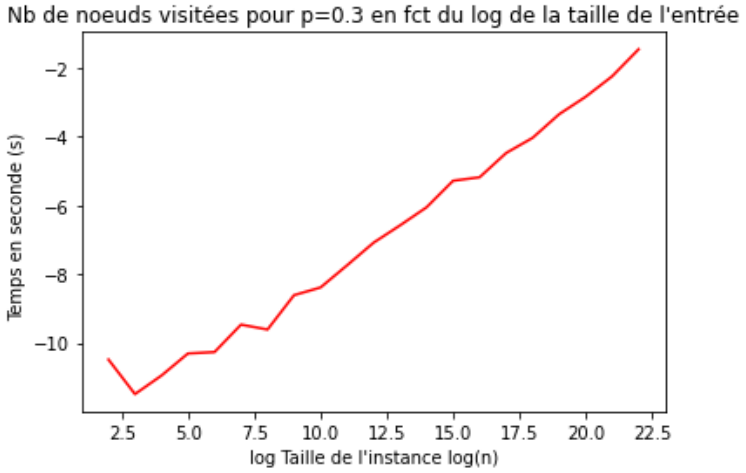
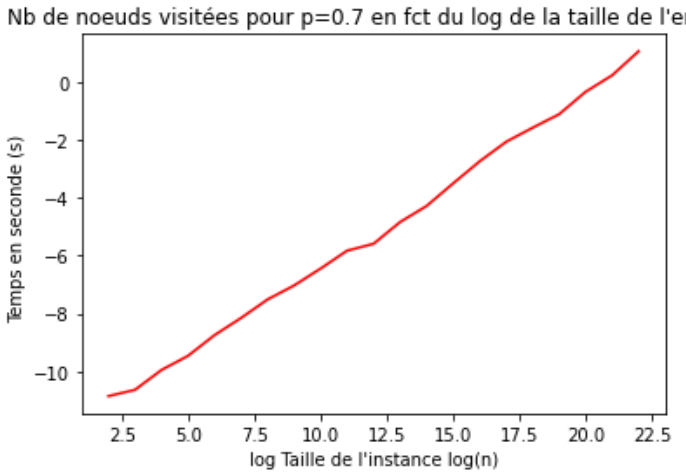


Figure 6:




- En effet la complexité de l'algorithme Branch and Bound est exponentielle car on doit explorer l'arbre entier des différentes possibilités de couvertures, ainsi si on ajoute une arête dans le graphe d'entrée, on multiplie la taille de l'arbre et le temps de calcul par deux.

On a donc bien une complexité temporelle exponentielle en fonction de l'entrée, cette algorithme n'est donc pas utilisable pour calculer l'optimum pour tout les graphes car le temps de calcul devient rapidement très grand lorsqu'on augmente la taille du graphe. (plus d'1 min secondes pour un graphe avec 30 sommets et 15 arêtes).

## 3.2 Ajout de bornes

### 3.2.1 Preuve de la borne $b_1$

Soit C une couverture , $\Delta$ le degré max, et m le nombre d'arêtes du graphe. On peut écrire $|C| * \Delta \geq m$ car les sommets d'une couverture sont connectés à toutes les arêtes.
C'est à dire : $|C| \geq m/\Delta$, comme C est entier et $m/\Delta$ n'est pas obligé d'être entier on a : $|C| \geq \lceil m/\Delta \rceil$

### 3.2.2 Preuve de la borne $b_2$

Montrons que pour toute couverture C de G nous avons : $|C| \geq |M|$ avec M un couplage de G.
Soient k les sommets de G de degré $\geq 1$. On peut alors écrire :

$M = \{(s_1, s_2); (s_3, s_4); ...; (s_{k-1}, s_k)\}$ (avec k pair : une arete relie forcément 2 sommets).
(1) Nous avons donc $0 \leq |M| = \frac{k}{2} \leq k \leq n$ (avec n, nombre de sommets de

G) (2) Une couverture C peut contenir au plus k sommets, ce qui nous donne : $|C| \leq k$. De plus, nous pouvons définir $C = \{c_1, c_2, ..., c_{|C|}\}$ et pour chaque $c_i, i \in [1, |C|]$ nous avons $d_i$ qui représente le degré du sommet $i$. Or, on sait que chaque $d_i \geq 1$ d'après l'hypothése de la construction d'une couverture. Donc on en déduit que $c_i$ implique un autre sommet de G différent des sommets de C de degré $\geq 1$ d'après la définition d'une couverture. (3) Donc on en déduit : $|C| \geq \frac{k}{2}$ Donc d'après (1),(2) et (3) : $0 \leq |M| = \frac{k}{2} \leq |C| \leq k \leq n \leftrightarrow |M| \leq |C|$

### 3.2.3 Preuve de la borne $b_3$

Majorons le nombre de sommet du graphe G par le nombre de sommets maximal de la couverture C reliés aussi avec les sommets de $G/C$

Nous avons donc : $\frac{|C|^2-|C|}{2}$ le nombre maximal de sommet de la couverture C, $(n - |C|)$ correspond au nombre de sommets de $G/C$ et donc $|C| * (n - |C|)$ correspond au nombre maximal d'arête des sommets de C et des sommets de $G/C$.

$$m \leq \frac{|C|^2-|C|}{2} + |C| * (n - |C|)$$

$$m \leq \frac{1}{2} * |C|^2 - \frac{1}{2} * |C| + |C| * n - |C|^2$$

$$\frac{1}{2} * |C|^2 - |C|(n - \frac{1}{2}) + m \leq 0$$

$$\Delta = (n - \frac{1}{2})^2 - 4 * \frac{1}{2} * m$$

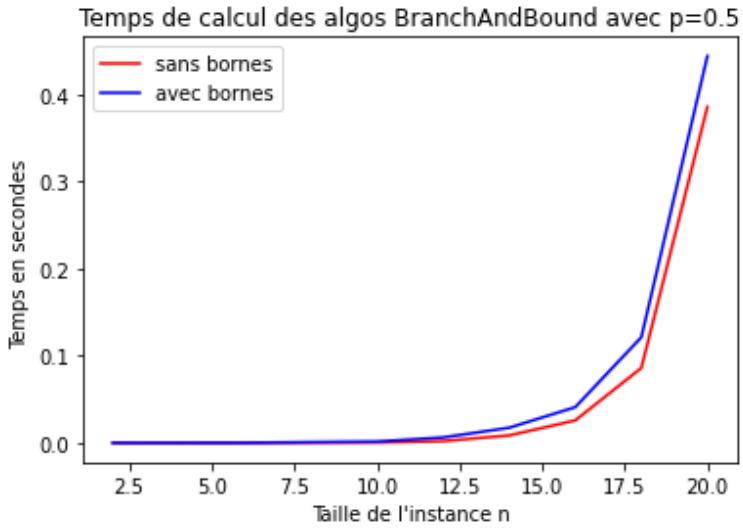
$$= n^2 - n + \frac{1}{n} - 2m$$

$$x_1 = \frac{(n-\frac{1}{2})-\sqrt{n^2-n+\frac{1}{4}-2m}}{2*\frac{1}{2}} \; x_2 = \frac{(n-\frac{1}{2})+\sqrt{n^2-n+\frac{1}{4}-2m}}{2*\frac{1}{2}}$$

En multipliant $x_1$ par $\frac{2}{2}$, nous obtenons :
$x_1 = \frac{2n-1-\sqrt{(2n-1)^2-8m}}{2}$
$x \in [x_1, x_2]$ Ici nous cherchons le plus petit minorant pour $|C|$, donc nous gardons la valeur de $x_1$, ce qui nous donne : $|C| \geq x_1$

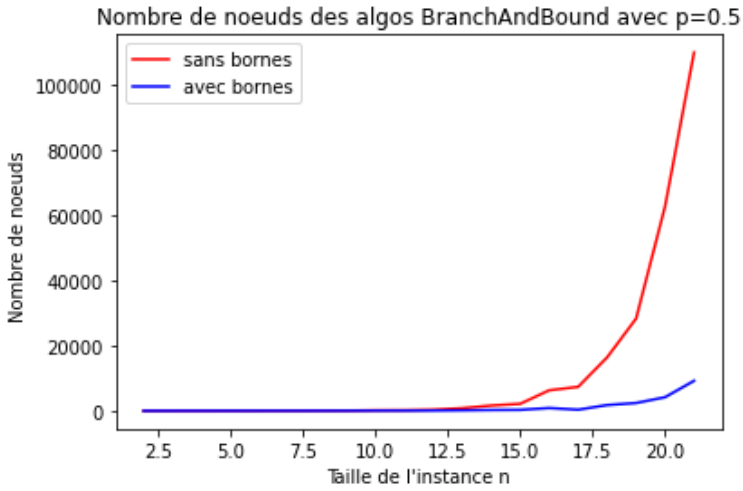
**2** - On voit que cette méthode avec bornes a une complexité équivalente à celle de la méthode précédente car les temps de calcul sont égaux (environ), comme on peut le voir sur la figure 7:

Figure 7:




- Néanmoins, on voit que la méthode avec les bornes permet de visiter beaucoup moins de noeuds que sans les bornes.

Figure 8:




- Cela peut s'expliquer par le fait que les "élagages" permettent de réduire significativement le nombre de noeuds visités mais à chaque noeud, on doit appliquer l'algorithme de couplage sur le graphe correspondant à ce noeud ce qui consomme beaucoups de temps de calcul (même si cet algorithme a une

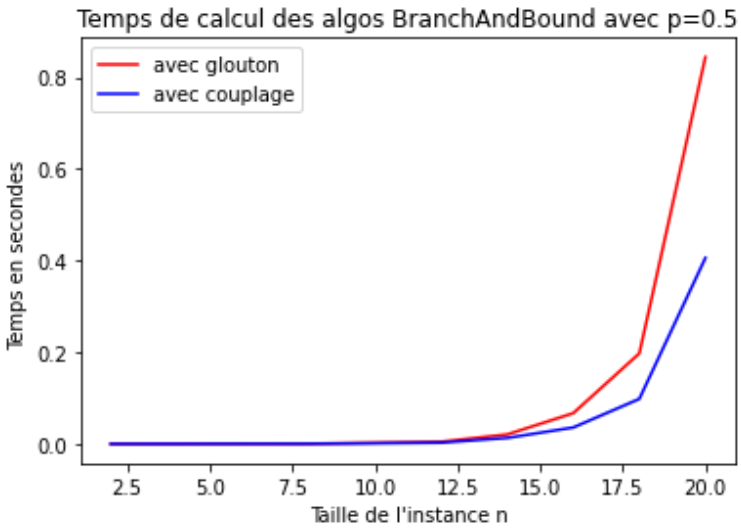
complexité polynomiale). Pour améliorer notre algorithme de branchement, il faudrait une méthode renvoyant une solution réalisable plus économique en temps de calcul.

Malgré les bornes dans notre dernière méthode, celle-ci garde donc une complexité exponentielle car sa complexité est equivalente à celle de la méthode sans les bornes.

En effet, avec l'ajout de bornes, on a toujours un nombre de noeuds qui évolue de manière exponentielle en fonction de la taille de l'entrée, donc même si on en visite qu'une fraction avec notre dernière méthode, la complexité reste exponentielle.

**3** L'algorithme glouton renvoie de meilleures solutions que l'algo couplage comme on l'a vu dans la question 3.2. Si on remplace l'algo couplage par l'algo glouton dans BranchAndBound, on aurait ainsi une meilleure borne supérieure car la solution réalisable serait plus petite, donc on peut "élaguer" plus de branches dans l'arbre de possibilités. Néammoins le temps de calcul pour glouton est très supérieur à bouclage pour un même graphe, ainsi si on remplace couplage dans l'algorithme BranchAndBound avec bornes on a les temps de calculs de la figure 10

Figure 9:

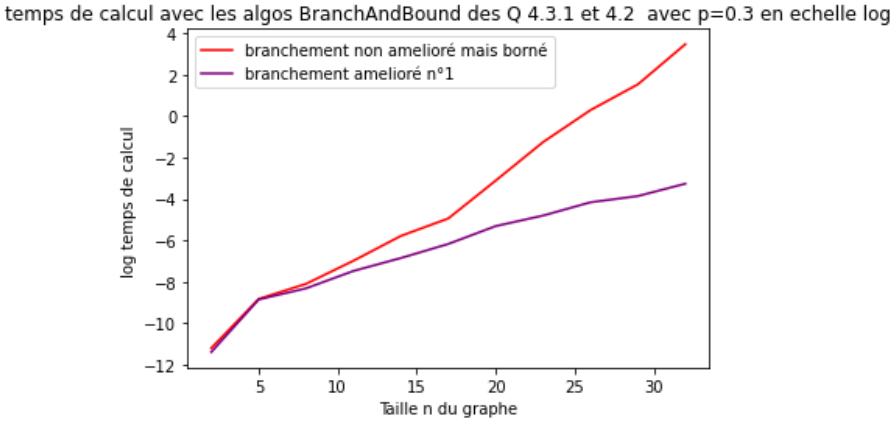



- Ainsi, remplacer notre heuristique par l'algorithme glouton ne réduirait pas notre temps de calcul même s'il renvoie des solutions plus proches de la solution optimale donc réduit le nombre de branches à chaque noeud.

### 3.3 Amélioration du branchement

**1** - Si on branche sur une arête $(u, v)$, on ne prend pas $u$ dans la 2eme branche et on prend tout les voisins de u (à part v) donc on réduit le nombre de noeuds dans cette branche par rapport aux branchements utilisés précédemment.Dans cette méthode, on ajoute cette technique à la méthode de branchement avec borne de la partie 4.2. On voit ainsi que le nombre de noeuds visités avec ce branchement est très inférieure à la technique de branchement de la partie 4.2.

Figure 10:




**-** On voit ainsi que le nombre de noeuds visités avec ce branchement est très inférieur à la technique de branchement de la partie 4.2. On voit d'après la figure 10 que les deux courbes forment approximativement des droites en échelle log, donc on reste sur complexité exponentielle avec le branchement amélioré.

On a bien vérifié lors des tests que cette méthode renvoie une solution optimale car c'est une couverture de même taille que la solution par l'algorithme BranchAndBound borné.

**2** On voit que la fonctionnalité ajoutée (commencer le branchemement avec le noeud de degré maximal), réduit bien le temps de calcul par rapport à la méthode précédente même si on reste sur une complexité exponentielle (figure 11).

Figure 11:

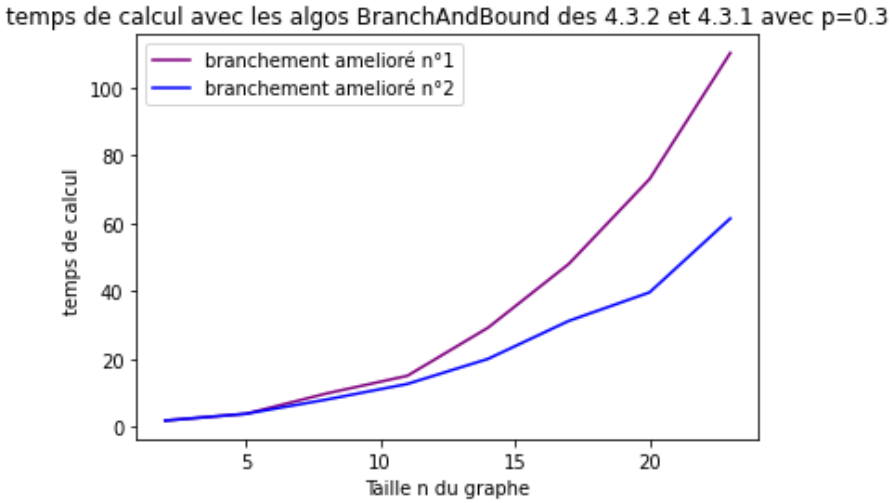
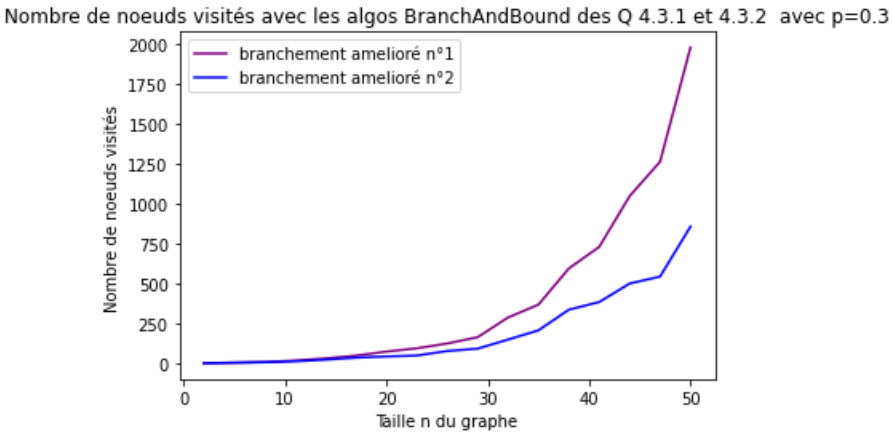



Figure 12:




- En effet, avec cette nouvelle fonctionnalité, on améliore notre borne supérieure qui est la meilleure solution qu'on a déja trouvée, on trouve des meilleurs solutions au début de l'exploration de l'arbre (car c'est la ou on regarde les sommets qui ont les plus grand degrés). Donc le nombre de noeuds visités est inférieure au nombre de noeuds visités à la méthode précédente (figure 12).

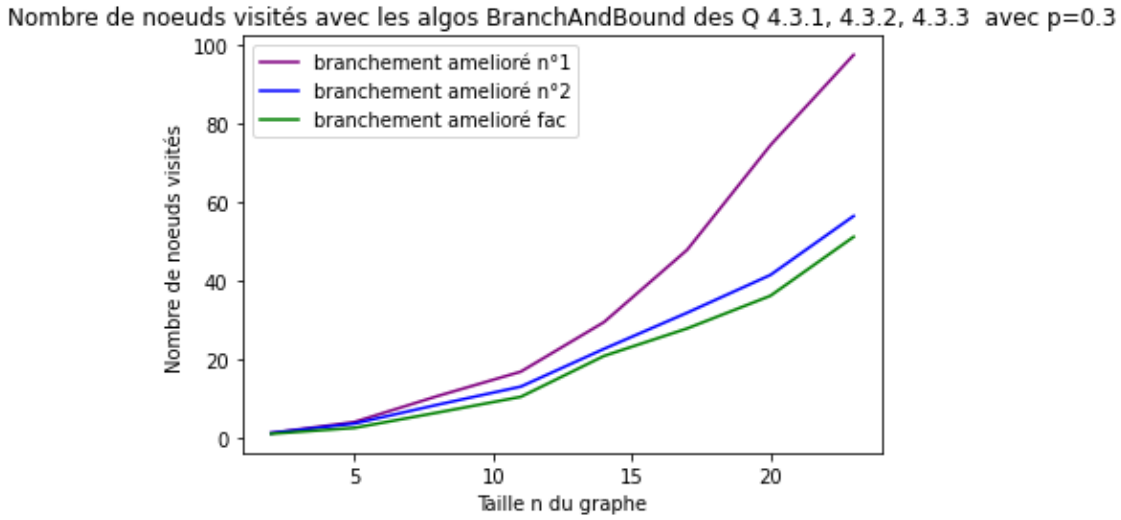
- On a bien vérifié que les solutions renvoyées par la méthode étaient polynomiales en les comparant avec les solutions de l'algorithme BranchAndBound borné de la partie 4.2.

**3** Dans un graphe, si un sommet u est de degré 1, alors il est l'extrémité d'une unique arête A .L'autre extrémité v de cette arête est de degré au moins 1 car elle est déjà relié à A. Ainsi si on prend u dans la couverture, cela ne permettra d'atteindre que 1 arête (l'arête A), or si on prend v on atteindra au moins une arête (A mais potentiellement d'autres arêtes).

Soit dans C1 une couverture qui tel que u appartient à C1 et C2 une couverture qui tel que v appartient à C2, on a $|C1| \geq |C2|$. Donc il existera toujours une couverture optimale qui ne contient pas u, cette couverture comprend v.

On applique cette fonctionnalité à la dernière méthode, on voit bien qu'on a moins de noeuds visités d'après la figure 13.

Figure 13:




- Néanmoins le temps de calcul n'est pas amélioré, on suppose que c'est à cause de la recherche du degré d'un sommet à chaque branchement coûte plus que branché avec un sommet de degré 1.

## 3.4 Qualité des algorithmes approchés

Figure 14:

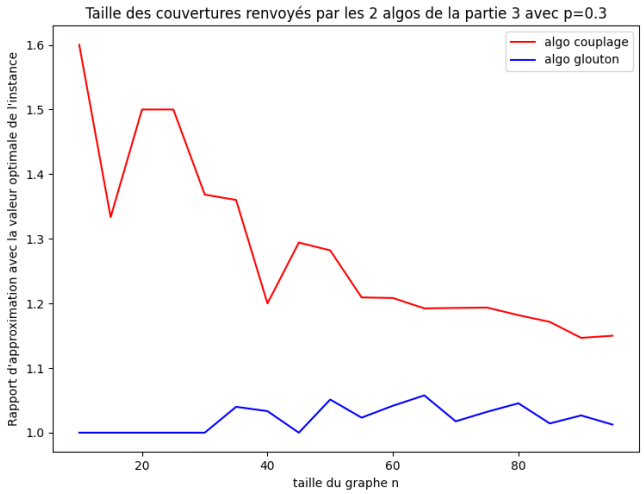
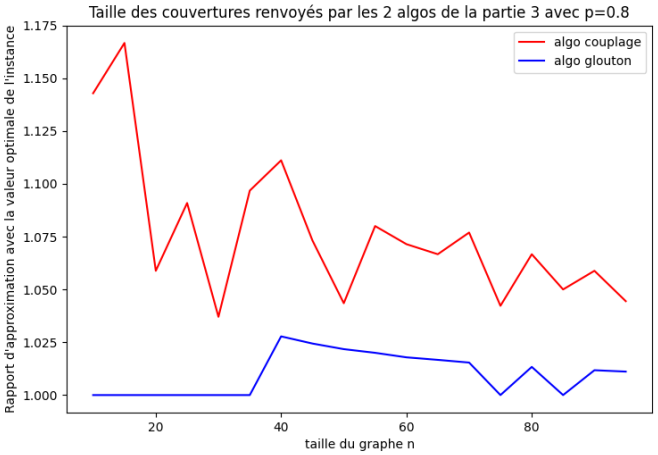



Figure 15:




**1** Nous avons evalués le rapport d'approximation des algorithmes glouton et de couplage en nous basant sur la solution optimale renvoyé par l'algorithme branch and bound. D'après les figures 14 et 15, nous avons constaté que l'algo de couplage présente un plus grand rapport d'approximation comme attendu.

- Nous avons réparti nos expérimentation sur deux probitilités de branchement $p_1 = 0.3$ et $p_2 = 0.8$ avec un nombre de sommets n maximum de 100. Nous avons calculer le rapport maximal d'approximation avec cette formule :

$$Rapport_{max}glouton = max_n(val(algoGlouton(I_n))/algoBB(I_n))$$
$$Rapport_{max}couplage = max_n(val(algoCouplage(I_n))/algoBB(I_n))$$

$$n \in \{10, 15, 20, 25, ..., 95, 100\}, I_n : \text{instance de taille n}$$

- Voici un tableau (figure 16) qui présente les rapport d'approximation maximaux de nos algorithmes en fonction des instances.

Figure 16:

| P | Algorithme | Rapport d'approximation maximum |
|---|---|---|
| P1 = 0,3 | glouton | 1,125 |
| | couplage | 1,5 |
| P2 = 0,8 | glouton | 1,026 |
| | couplage | 1,428 |

- Le rapport d'approximation maximum est de 1.5 pour l'algorithme de couplage et 1.125 pour l'algorithme glouton. Donc nous déduisons, selon des experimentations, que l'algorithme glouton est 1.125-approché et l'algorithme de couplage est $\frac{3}{2}$-approché Nous remarquons aussi que plus le taux de branchement est faible, plus le rapport est grand ce qui conjecture que ces deux grandeurs sont inversement proportionnelles.