Projet de Master

"Complexité, algorithmes randomisés et approchés"

Couverture de graphe

Binôme:

Johan-Luca Rossi
Baptiste Chevalier

Langage utilisé: Python

Sommaire

1 Représentation des graphes en Python

Représentation des graphes: Les graphes ont été représentés a l'aide de dictionnaires python. De cette manière si k est le plus grand d'indice d'un sommet du graphe, il n'est pas nécessaire de disposer d'un tableau à k entrées.
On a donc des graphes représentés de cette manière:

```
1 G = {8:[1,2], 1:[8,4], 2:[8], 3:[4], 4:[1,3]}
```

2 Méthodes approchées

2.1 Méthode du couplage maximal

```
1  def couplageMax(G):
2      C = []
3      for x in G:
4          if x not in C:
5              for y in G[x]:
6                  if y not in C:
7                      C.append(x)
8                      C.append(y)
9                      break
10     return C
```

2.1.1 Complexité

Pour un graphe $G(V, E)$ et un couplage C, soit $m = |E|$,$n = |V|$ et $c = |C|$.
Les conditions lignes 4 et 6 sont en $\mathcal{O}(c)$ et la boucle est exécutée $2m$ fois, on obtient donc une complexité en $\mathcal{O}(m \times c)$
Dans le meilleur cas Graphe stable $\mathcal{O}(1)$
Dans le pire cas Clique de taille n : tous les sommets sont reliés entre eux, on effectue $n(n-1)$ fois les primitives "not in" on a alors $\mathcal{O}(n^2c - nc)$.
En général on aura donc pour une entrée de taille N : l'algorithme de couplage maximum s'exécute en $\mathcal{O}(N^3)$.

2.1.2 Tests d'exécution

Note pour les tests : Pour toutes les courbes expérimentales qui vont suivre,chaque point est une moyenne sur 10 points. L'échelle de représentation est toujours précisée, la variable p représente toujours la probabilité de présence d'arêtes et n le nombre de noeuds de l'instance. Les tests sont réalisés sur des graphes générés aléatoirement. Dans les cas de comparaison, les exécutions des deux fonctions sont faites sur le même graphe (nouveau graphe généré pour chaque valeur de n et pour chacune des 10 itérations permettant de construire la moyenne).

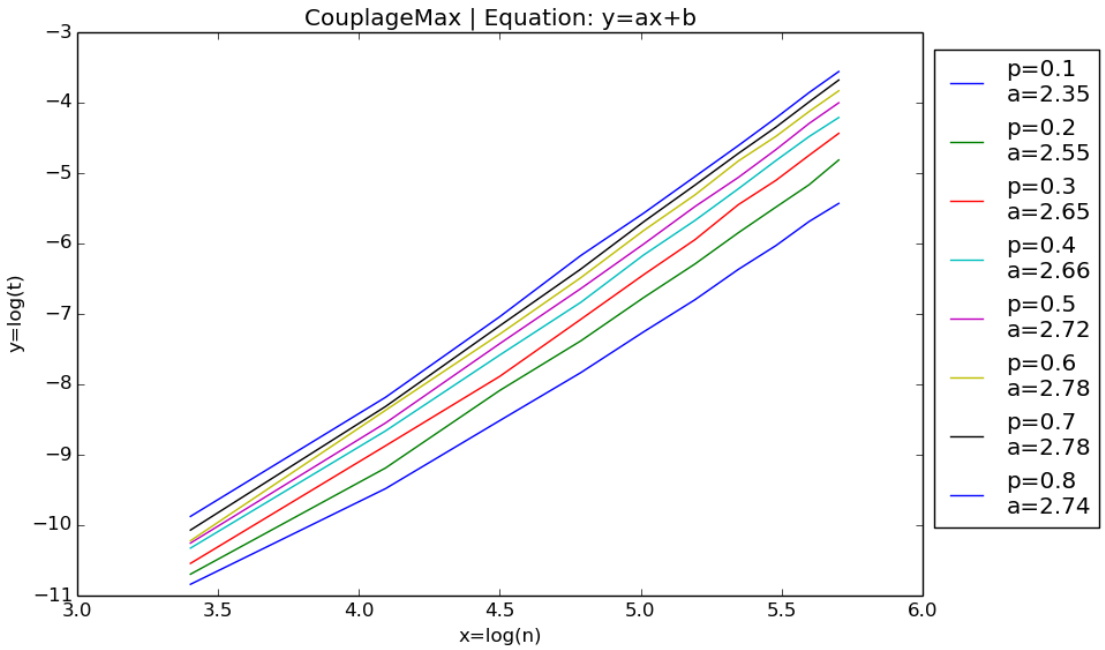


Figure 1: Complexité couplageMax, $n \in [30; 300]$ avec un pas de 30, $p \in [0.1; 0.8]$ avec un pas de 0.1. En ordonnée le logarithme du temps d'exécution moyen et en abscisse le logarithme de la taille de l'instance.

On remarque que les courbes s'apparentent fortement à des droites, cela confirme la complexité polynomiale de l'algorithme. Le coefficient directeur a tend vers 3 quand p augmente, cela est en accord avec la théorie car plus on augmente le nombre d'arêtes plus on se rapproche du pire cas de l'algorithme. On a donc bien une complexité en $\mathcal{O}(N^3)$.

2.2 Algorithme glouton

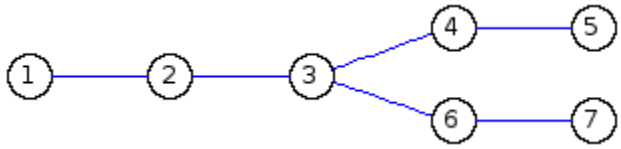
```
1  def algo_glouton(G):
2      C=[]
3      while arete(G) :
4          v = degMax(G)
5          C.append(v)
6          G=supprimerSommet(G,v)
7      return C
```

2.2.1 Complexité

Pour un graphe $G(V, E)$, soit $m = |E|$ et $n = |V|$.
arete(G) est en $\mathcal{O}(n)$, degMax(G) est en $\mathcal{O}(n)$, supprimerSommet(G,v) exécute $2m$ fois la primitive remove(u) de Python, celle-ci est en $\mathcal{O}(deg(u))$ (degré de u) que l'on simplifie en $\mathcal{O}(n)$, enfin la boucle itère au plus m fois.
On obtient donc une complexité en $\mathcal{O}(m \times (mn + n))$.
Dans le meilleur cas Graphe stable $\mathcal{O}(1)$
Dans le pire cas (cycle) toutes les arêtes sont de degrés 2 et on a $n = m$ on a alors une complexité pire cas $\mathcal{O}(n^2 + n^3)$
En général on aura donc pour une entrée de taille N : l'algorithme glouton s'exécute en $\mathcal{O}(N^3)$

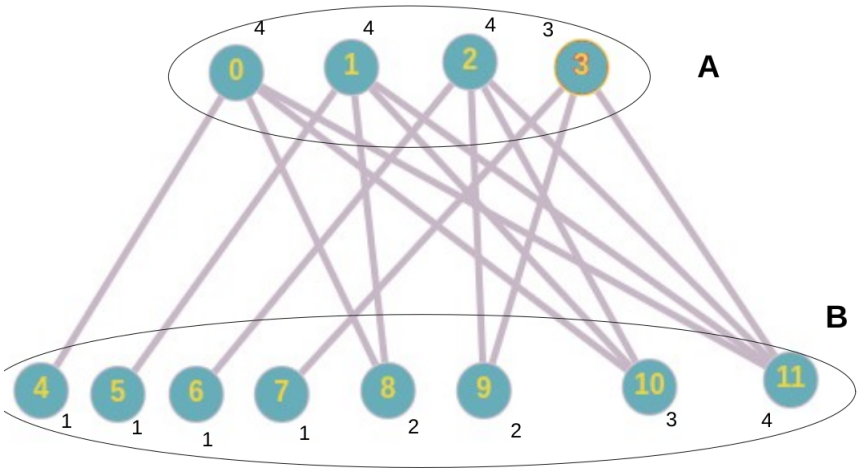
2.2.2 Optimalité et approximation :

Soit le Graphe G suivant:



L'exécution de l'algorithme glouton sur G retourne la solution $S = \{3, 2, 4, 6\}$, or une solution optimale est $S^* = \{2, 4, 6\}$, on a $|S| > |S^*|$, **l'algorithme glouton n'est pas optimal**. On remarque que $|S| = \frac{4}{3}|S^*|$. On peux en déduire directement que **l'algorithme n'est pas r-approché pour** $r < \frac{4}{3}$.

Preuve que Glouton n'est pas r-approché : Nous allons prouver que l'algorithme glouton précédent n'est pas r-approché et ce pour tout r.
Soit le G graphe bipartie suivant :



Dans ce graphe il existe plusieurs sommets de degré maximal, l'algorithme va devoir choisir l'un de ces sommets mais on ne nous dit pas lequel. Il alors possible de considérer le cas (qui peut être rare mais existe) qui va nous arranger.
Ici par exemple on va considérer que l'algorithme commence par prendre le sommet 11. Ainsi, le degré des sommets 0,1,2 va baisser, nous amenant dans une situation identique à la précédente, c'est à dire, l'algorithme doit choisir entre plusieurs sommets de degré maximal.
On répète l'opération jusqu'à obtenir pour couverture l'ensemble B.
Or comme G est bipartie, A est une couverture meilleure et optimale de G.
Ainsi $|C| = |B| = 8 \geq |C^*| = |A| = 4$, on en déduit que la solution retournée est deux fois moins bonne en considérant le rapport d'approximation : $\alpha = \frac{|B|}{|A|}$

Généralisons en reprenant ce concept de graphe bipartie.
Supposons que l'algorithme glouton soit r-approché pour un certain r. Alors r borne notre rapport d'approximation α. Nous allons montrer qu'un tel r ne peut pas exister en faisant augmenter la taille de notre graphe bipartie.
Soit $k = |A|$ la taille de notre ensemble A, qui est également la taille de la couverture optimale.
On va chercher à maximiser le nombre de sommets dans B, afin de faire croître α, tout en faisant en sorte que le degré maximal d'un sommet de B ne dépasse pas k.
On va alors placer dans B, k sommets de degré 1, $\lfloor \frac{k}{2} \rfloor$ sommets de degré 2, ... et d'une façon plus générale $\lfloor \frac{k}{d} \rfloor$ sommets de degré d.
On obtient alors la relation suivante :

$$|B| = \sum_{d=1}^{k} \lfloor \frac{k}{d} \rfloor$$

La division euclidienne de k par d peut s'écrire comme :

$$k = \lfloor \frac{k}{d} \rfloor d + r \tag{1}$$
$$0 \leq r < d \tag{2}$$
$$\implies \lfloor \frac{k}{d} \rfloor = \frac{k-r}{d}$$

Donc

$$|B| = \sum_{d=1}^{k} \frac{k-r}{d}$$

On en déduit :

$$\begin{aligned}
\alpha &= \frac{|B|}{|A|} = \frac{|B|}{k} \\
&= \sum_{d=1}^{k} \frac{k-r}{kd} \\
&= \sum_{d=1}^{k} \frac{1}{d} - \frac{1}{k} \sum_{d=1}^{k} \frac{r}{d} \\
&= H_k - < \frac{r}{d} >
\end{aligned}$$

Or on remarque que le terme de gauche est en fait la série harmonique.
Cette somme peut être approximée par un développement asymptotique comme $H_k = \log k$.
Le terme de droite est une constante valant la valeur moyenne des $\frac{r}{d}$. Comme on a toujours $r < d$ alors $\frac{r}{d} < 1$ et donc cette valeur moyenne vaut moins de 1.
On en déduit donc que $\alpha = \mathcal{O}(\log k)$.
Ainsi lorsque la taille de notre ensemble A : $k \to +\infty$, α diverge vers $+\infty$ car la série diverge. Il n'existe donc pas de r bornant le rapport d'approximation. Ceci conclut notre preuve comme quoi l'algorithme glouton n'est pas r-approché.

2.2.3 Tests d'exécution

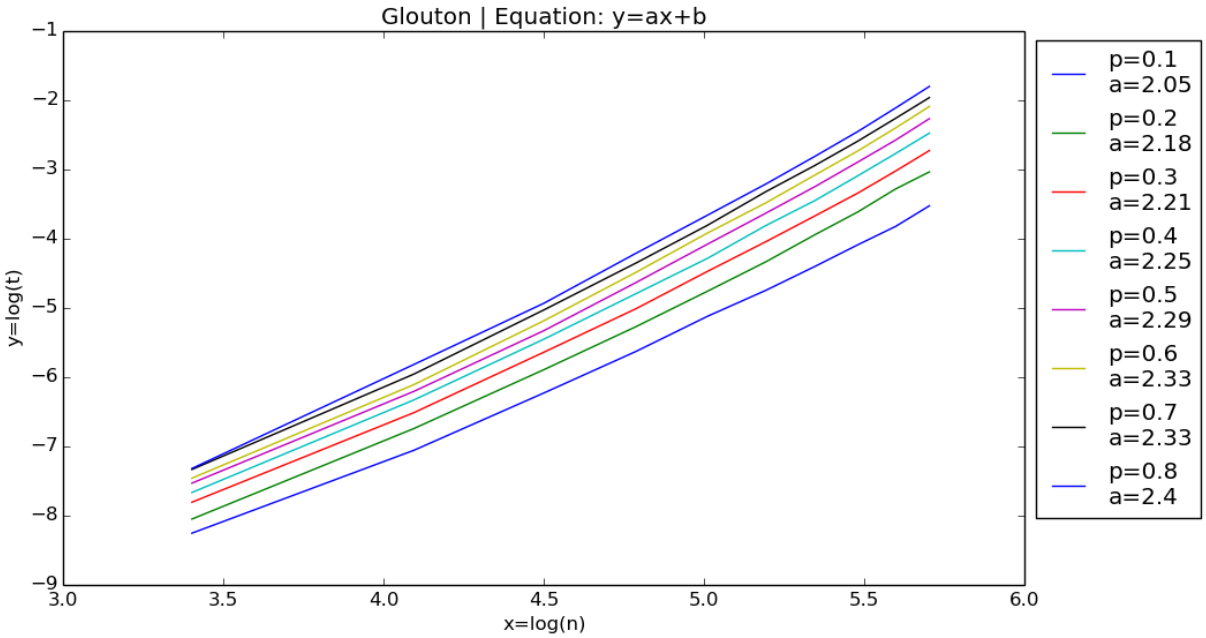


Figure 2: Complexité Glouton, même configuration que pour couplageMax (cf. Figure 1)

On remarque que les courbes s'apparentent à des droites, ce qui confirme déjà la complexité polynomiale de l'algorithme glouton, de plus on remarque que le coefficient directeur des droites tend vers 3 quand p augmente, on retrouve donc bien comme prédit une complexité en $\mathcal{O}(N^3)$.

2.3 Comparaison des méthodes

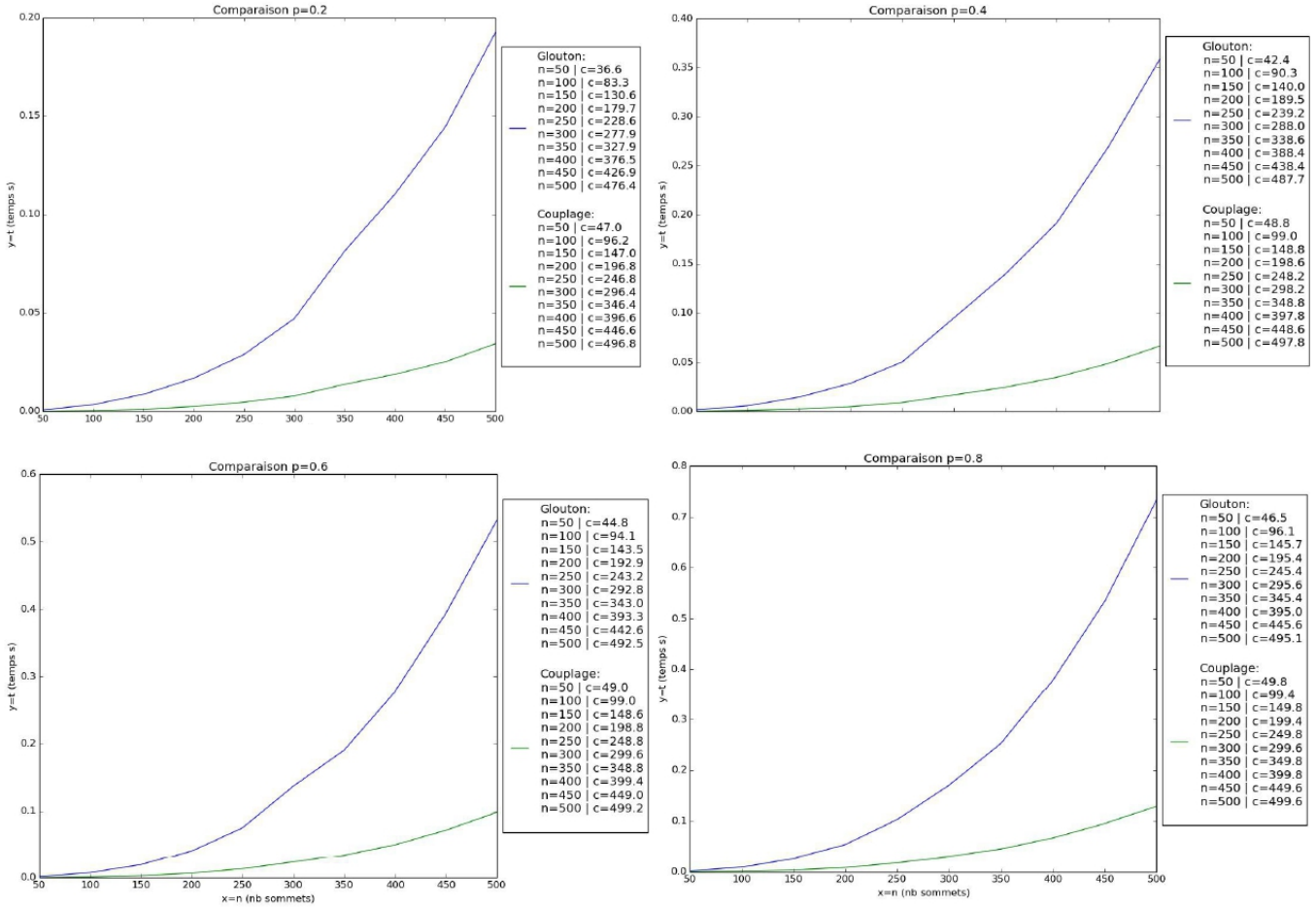


Figure 3: Comparaison temps d'exécution et solutions retournées pour Couplage-Max/Glouton, $n \in [50; 500]$ avec un pas de 50, $p \in [0.2; 0.8]$ avec un pas de 2. On définit la variable c représentant la moyenne de la taille du couplage C retourné.

Temps d'exécution et qualité des solutions retournées : On observe clairement pour chaque valeur de p que l'exécution de l'algorithme glouton prend nettement plus de temps que pour couplage (presque un ordre de grandeur au dessus), au niveau de la qualité des solutions retournées pour chaque taille d'instances et pour chaque p, glouton renvoie systématiquement une meilleure solution en moyenne.

Note sur les complexités: Il est intéressant d'observer que lors du calcul des coefficients pour les courbes de complexité (cf. Figure 1/2), les coefficients de couplage

sont plus importants que ceux de glouton, cependant on a un temps d'exécution nettement plus important pour glouton. Cela est sûrement du aux termes polynomiaux supplémentaires dans la complexité de Glouton, il faudrait alors tester sur un très grand nombre de sommets pour observer un temps d'exécution plus important pour couplageMax (nous avons testé jusqu'à $n = 1000$, mais n'avons pas vu de réelle évolution).

3 Séparation et évaluation

3.1 Branchement naïf

Le branchement naïf a été effectué récursivement. A chaque appel on génère deux noeuds et on rappelle sur ces deux noeuds. On obtient donc évidemment une **complexité exponentielle** en $\mathcal{O}(2^m)$ (Pour un graphe $G(V, E)$, $m = |E|$).

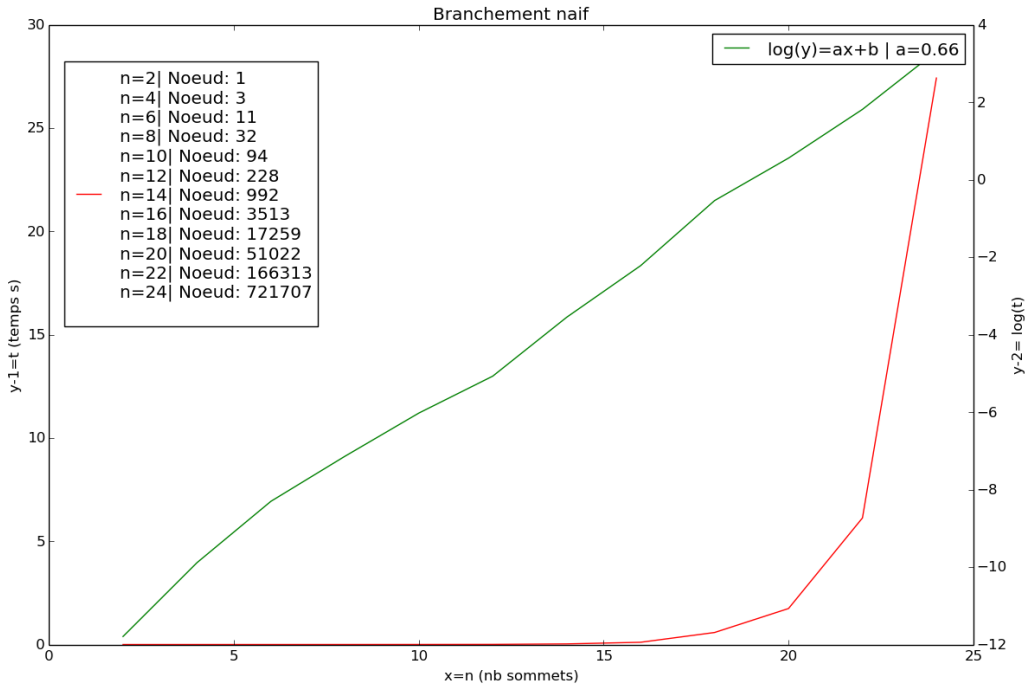


Figure 4: Temps d'exécution (rouge), complexité(vert) et noeuds générés, $n \in [2; 26]$ avec un pas de 2, $p = 1/\sqrt{n_{max}}$

On remarque que la courbe $y_2 = log(t)$ s'apparente bien a une droite, on retrouve bien la complexité exponentielle prévue.

3.2 Ajout de bornes

3.2.1 Définition des bornes

Borne Supérieure Pour notre borne supérieure nous utiliserons une solution réalisable retournée par l'algorithme de couplage dans un premier temps. Nous comparerons par la suite les résultats obtenus en utilisant une solution réalisable de l'algorithme glouton comme borne supérieure.

Borne Inférieure Nous définissons notre borne inférieure de la façon suivante :

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

avec $b1 = \lceil \frac{m}{\Delta} \rceil$ (ou Δ est le degré maximum d'un sommet du graphe), $b2 = |M|$ et $b3 = \frac{2n-1-\sqrt{(2n-1)^2-8m}}{2}$

Preuve des bornes Inférieures Soit G=(V,E) un graphe comportant n sommets et m arêtes.

Preuve de la borne b1 :

Soit $V' \subset V$ un sous ensemble de sommets de V. Alors le nombre d'arêtes couvertes par un sous-ensemble V' dans G vaut le nombre de voisins de chaque sommets $x \in V'$ auquel il faut retirer les doublons (les sommets de V' qui sont voisins). On a donc la relation suivante pour le nombre d'arêtes couvertes par V':

$$\#nombre\ aretes\ couvertes = \sum_{x \in V'} deg(x) - |\{uv \in E | u\ et\ v \in V'\}|$$

Maintenant considérons le cas ou $V' = C$ est une couverture de G. Alors :

$$\begin{aligned} &\#nombre\ aretes\ couvertes = m \\ &\sum_{x \in C} deg(x) - |\{uv \in E | u\ et\ v \in C\}| = m \\ &\sum_{x \in C} deg(x) \geq m \end{aligned}$$

En posant $\Delta = \max_{x \in V} \ deg(x)$, alors $\forall x \in C, deg(x) \leq \Delta$ Ainsi :

$$|C|\Delta \geq \sum_{x \in C} deg(x) \geq m$$
$$|C| \geq \lceil \frac{m}{\Delta} \rceil$$

Preuve de la borne b2 :

Soit $M \subset E$ un couplage de G. Alors $\forall uv \in M$, une couverture C de G doit contenir au moins l'un des deux sommets u ou v. Ainsi $|C| \geq |\{u | uv \in M\}| = |M|$

Preuve de la borne b3 :

Soit $C = \{x_1, x_2, ..., x_d\}$ une couverture.
On appelle m_{max} le nombre d'arêtes maximum qui peuvent être couvertes par C.
Soit $x_1 \in C$, afin de maximiser m_{max}, on doit faire en sorte que x_1 couvre le plus d'arêtes possible, soit n-1 arêtes. On répète l'opération pour chaque arête de C, ainsi toute arête x_i couvre n-i arêtes.

$$\begin{aligned} m_{max} &= \sum_{i=1}^{d} \ n - d = nd - \sum_{i=1}^{d} d = nd - \frac{d(d+1)}{2} \geq m \\ &\implies 2nd - d(d+1) \geq 2m \\ &\implies 2nd - d^2 - d - 2m \geq 0 \\ &\implies -d^2 + (2n-1)d - 2m \geq 0 \end{aligned}$$

On trouve une équation d'un polynôme du second degré en fonction de la taille d de notre couverture. En solvant l'inéquation on en déduit :

$$|C| = d \geq \frac{2n - 1 - \sqrt{(2n-1)^2 - 8m}}{2}$$

3.2.2 Branchement avec bornes

Borne inférieure et solution réalisable : Dans un premier temps nous allons définir deux cas de figure pour les tests avec les bornes définies ci-dessus:

- Borne inf=max$\{b1, b2, b3\}$ | Borne sup gloutonne
- Borne inf =max$\{b1, b2, b3\}$ | Borne sup couplage

Ci dessous les courbes d'exécution en échelle linéaire des deux cas (borne sup Couplage:= "Sup:C", borne sup Glouton:="Sup:G" et borne inf=max$\{b1, b2, b3\}$:="Inf:C").

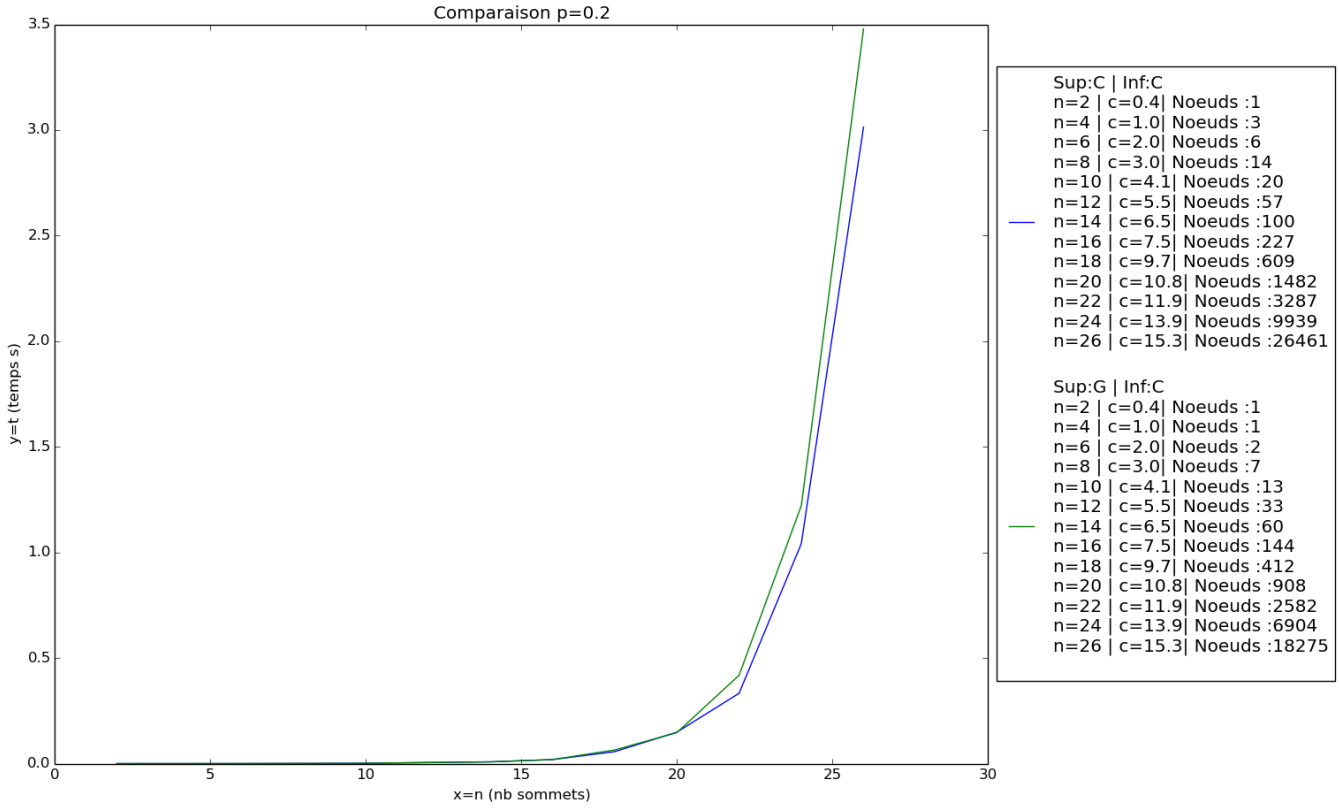


Figure 5: Comparaison borne supérieure Glouton/Couplage , en temps d'exécution, solutions retournées et noeuds générés ($c = |C|$), $n \in [2; 26]$ avec un pas de 2 et $p = 0.2$

Par rapport a au branchement naïf on remarque que le nombre de noeuds est de l'ordre de 10× plus important pour la méthode naïve que pour les méthodes bornées, le temps d'exécution en est donc directement impacté avec un ordre de grandeur de différence.
Pour les solution retournées on remarque que la qualité des solutions retournées est toujours la même, ce qui semble logique car dans les deux cas, on doit retourner la solution optimale.
Au niveau des noeuds générés on remarque qu'avec la borne G on obtient un nombre de noeuds générés inférieur a la borne sup C, presque 1.5× plus pour C que pour G, ce qui est parfaitement en accord avec les test précédents sachant que

la qualité des solutions de glouton est meilleure que celle de couplage (cf. Figure 3). Cependant les temps d'exécution sont très proches et pour les plus grandes instances le temps d'exécution de glouton devient supérieur, cela est dû au fait que le temps d'exécution de glouton est nettement plus important que celui de couplage (cf. Figure 3). **Le temps résultant des deux méthodes se compense donc entre nombre de noeuds générés et temps d'exécution des algorithmes de borne supérieure.**

3.2.3 Comparaison

Il est intéressant de comparer l'intérêt de l'utilisation de chacune des bornes. Pour cela, on exécutera l'algorithme de branchement en utilisant dans un premier temps une borne supérieure triviale, une borne inférieure triviale et les deux (cas Figure 5).

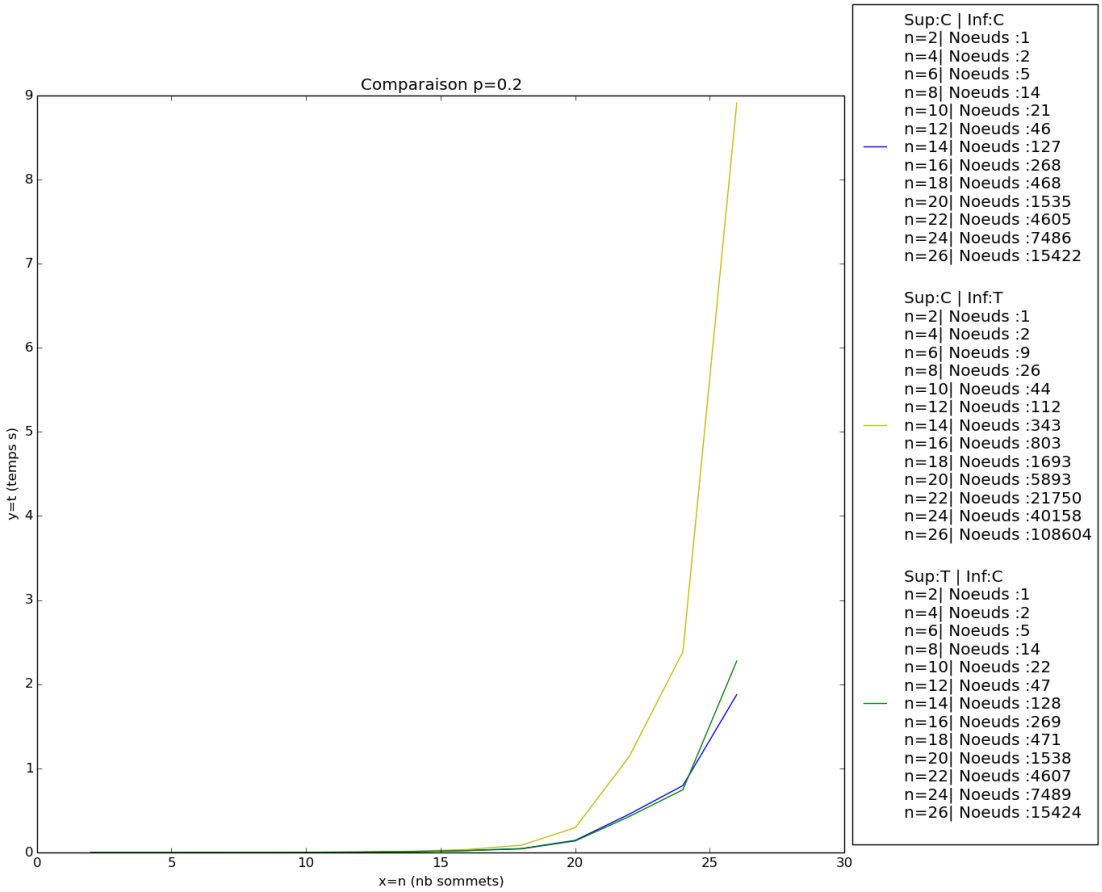


Figure 6: Comparaison des temps d'exécution, des noeuds générés avec uniquement la bornes inférieures(vert)/uniquement les solutions réalisables(jaune)/et les deux(bleu), $n \in [2; 26]$ avec un pas de 2 et $p = 0.2$.

L'utilisation des deux bornes permet d'obtenir le temps de calcul et le nombre de noeud générés les plus faibles.
Utiliser une seule des deux bornes semble tout de même intéressant et permet d'obtenir des temps de calcul réduits par rapport à l'algorithme naïf.
L'utilisation de la borne inférieure est bien plus avantageuse pour la réduction du temps de calcul en comparaison à celle de la borne supérieure. Le nombre de noeuds générés par l'utilisation de la borne supérieure seule étant environ 7× moins important que l'algorithme naïf contre environ 50× pour la borne inférieure. **Le temps de calcul de la borne inférieure est donc négligeable par rapport au gain final**.

3.3 Amélioration du branchement

Nous allons comparer les temps d'exécution, la complexité et les rapports d'approximation des trois méthodes d'amélioration présentées dans le sujet.

Ces méthodes améliorées représentent des méthodes approximées, la solution retournée ne sera donc pas toujours la solution optimale! En effet, en regardant le graphe de la figure 7 ci-dessous, on voit que l'algorithme amélioré présenté dans l'énoncé, en commençant par traiter l'arête {0,1} retournera la solution $S = \{0, 3, 4\}$ ou $S = \{1, 2, 3\}$ alors que la solution optimale est $S^* = \{0, 1\}$

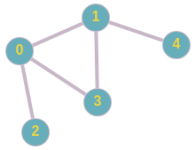


Figure 7: Exemple

Cependant l'utilisation de tels algorithmes devrait nous permettre de trouver une solution "correcte" (et nous allons voir par la suite ce que cela signifie) **en temps sous-exponentiel / polynomial**.
En effet, alors que l'algorithme branchement avec bornes explorait toujours deux branches supplémentaires pour chaque noeud valide (prendre le sommet u ou le sommet v). Ici dans l'algorithme de branchement amélioré, en prenant le sommet u on force l'algorithme à prendre tous les voisins de v, ainsi toutes les arêtes contenant un voisin de v ne sont plus traitées car il ne reste plus qu'une seule possibilité : prendre le voisin de v. Ceci réduit très fortement le nombre de possibilités et donc le nombre de noeuds explorés.

Précision sur la 3ème amélioration : Si u est un sommet de degré 1, il existe toujours une couverture optimale ne contenant pas u car:
Si $dg(u) = 1$ alors u possède un unique voisin nommé v.

- Soit $dg(v) = 1$ alors la couverture contenant v plutôt que u est équivalente.
- Soit $dg(v) > 1$ alors la couverture contenant v plutôt que u est meilleure.

3.3.1 Temps d'exécution, rapport d'approximation

Pour les trois améliorations on définit leur rapport d'approximation $r = \frac{|C|}{|OPT|}$ (la valeur de OPT étant calculée avec une des méthodes bornées ci-dessus).

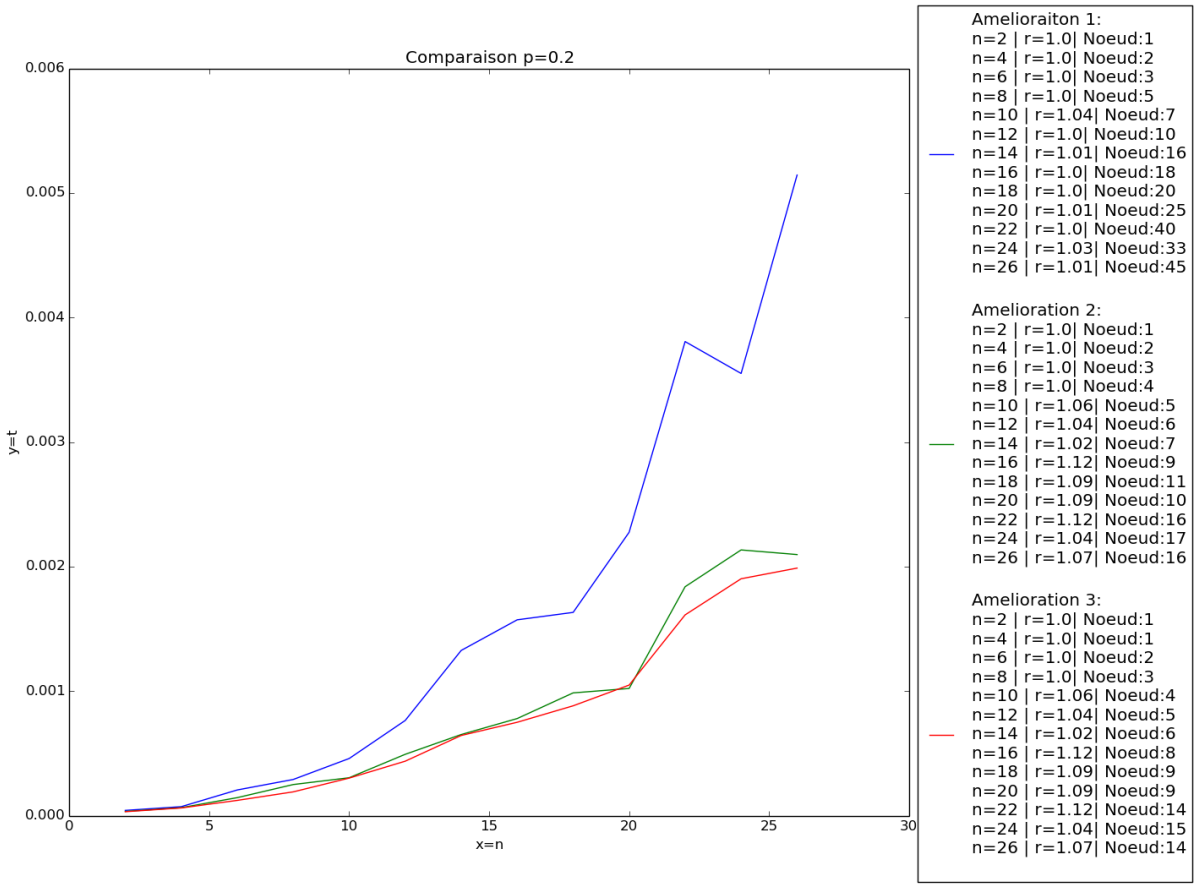


Figure 8: Comparaison temps d'exécution, noeuds générés et rapport approximation pour les améliorations 1(bleu)/2(vert)/3(rouge), $n \in [2; 26]$ avec un pas de 2 et $p = 0.2$.

Par rapport aux algorithmes de branchements précédents On remarque que le temps d'exécution est bien plus faible pour chacun des 3 algorithmes améliorés. Le nombre de noeuds générés est également bien plus faible, de l'ordre 100× inférieure par rapport au branchement avec bornes et donc de l'ordre 1000× inférieure pour le branchement naïf.
En comparant les différentes améliorations On remarque que chaque amélioration nous permet de gagner un peu de temps d'exécution. **L'amélioration 2** permet de diviser par deux le nombre de noeuds générés.**L'amélioration 3** apporte seulement un très léger gain de temps, en effet elle permet uniquement de traiter 1 à 2 noeuds de moins que l'amélioration 2.
Les rapports d'approximation sont supérieurs à 1 et s'en écartent quand le nombre de sommets augmente, montrant ainsi que la solution n'est pas toujours optimale. Cependant, les rapports restent tout de même proche de 1, on peut donc supposer que les solutions retournées par ces algorithmes sont très proches des solutions optimales.
On remarque que **les rapports d'approximation de l'amélioration 2 sont moins bons que ceux de l'amélioration 1**. En effet l'amélioration 2 permet de gagner du temps de calcul seulement en s'éloignant un peu plus de la solution optimale.

3.3.2 Complexité

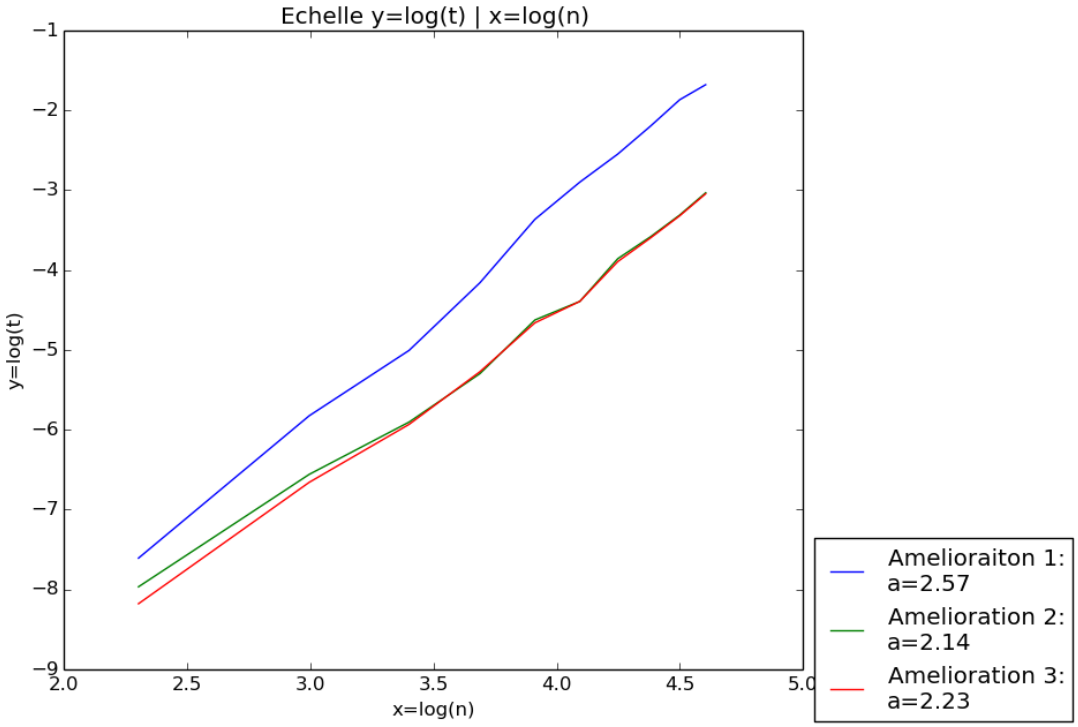


Figure 9: Complexité améliorations, $n \in [10; 100]$ avec un pas de 10, $p = 0.2$. En ordonnée le logarithme du temps d'exécution moyen et en abscisse le logarithme de la taille de l'instance.

On retrouve bien des courbes s'apparentant à des droites avec l'échelle $y = log(t), x = log(n)$, les algorithmes améliorés sont donc bien polynomiaux (le tracé a aussi été fait avec l'échelle $y = log(t)/x = n$, et on retrouvait bien des courbes se comportant comme des logarithmes, ce qui confirme d'autant plus la complexité polynomiale)

3.4 Qualité des algorithmes approchés

Dans cette dernière partie, nous avons testé la qualité des algorithmes approchés en exprimant leur rapport d'approximation r moyen sur 10 instances de taille n pour $n \in [2; 26]$. Comme dans la partie précédente, on utilise l'algorithme de branchement pour calculer une solution optimale puis on calcule le rapport entre les deux solutions (l'utilisation de l'algorithme de branchement empêche de tester sur des valeurs de n et p élevées). L'opération est répétée pour différentes valeurs de p, fixes à chaque fois.

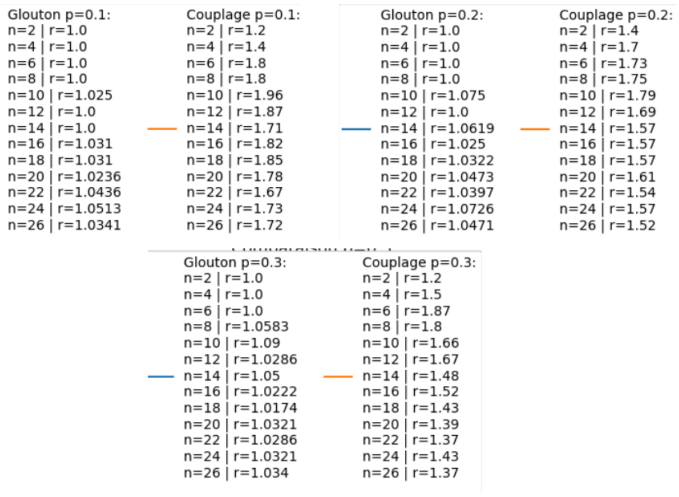


Figure 10: Rapport d'approximation couplage et glouton

Pour couplage **Le pire rapport d'approximation trouvé est 2** (qui n'apparaît pas sur les tableaux car ce sont des valeurs moyennées), ce qui est cohérent avec le fait que couplage soit **2-approché**.
On remarque que le **rapport d'approximation est décroissant pour p fixe lorsque n augmente**. Ceci s'explique par le fait que $m << n$, on se rapproche du meilleur cas pour l'algorithme de couplage : le graphe stable.
Il peut être intéressant de calculer dans chaque cas une régression linéaire :
Pour $p = 0,2$ on trouve : $r(n) = -0,01218n + 1,82309$ avec une corrélation de $R^2 = 0,74$
Pour $p = 0,3$ on trouve : $r(n) = -0,02481n + 1,94163$ avec une corrélation de $R^2 = 0,85$
On peut donc supposer que lorsque p augmente, on se rapproche d'une évolution linéaire pour la rapport d'approximation. Cette évolution est évidemment minorée par 1.

Pour Glouton Le rapport d'approximation est proche de 1 dans tous les cas, ceci montre que glouton est intéressant pour de tels n.
L'évolution semble chaotique, elle ne suit pas de fonction spécifique, et évolue de façon croissante (non stricte). Ceci est cohérent avec le fait que glouton ne soit pas r-approché pour tout r, mais les cas avec un rapport élevé sont rares.