Projet : Couverture de graphe (Vertex Cover)
UE Complexité – M1 Informatique, Sorbonne Université

Yuxiang ZHANG
Kenan ALSAFAD

Année universitaire 2025-2026

1 Introduction

Le problème de *couverture de graphe* (*Vertex Cover*) est un problème fondamental en théorie des graphes et en optimisation combinatoire. Étant donné un graphe non orienté $G = (V, E)$, une **couverture de sommets** est un sous-ensemble $V' \subseteq V$ tel que chaque arête $e \in E$ ait au moins une extrémité dans V'. L'objectif est de trouver une couverture de taille minimale.

Ce problème est **NP-difficile**. Sa version décisionnelle, consistant à déterminer s'il existe une couverture de taille au plus k, est **NP-complète**. Le but de ce projet est d'implémenter plusieurs approches — exactes et approchées — permettant de résoudre ce problème, et d'en analyser les performances empiriques.

2 Représentation du graphe

2.1 Structure de données

Le graphe est représenté par une **liste d'adjacence** sous forme de dictionnaire Python :

$$\text{adj} = \{\, v : [\,\text{voisins de } v\,] \,\},$$

offrant un bon compromis entre accès rapide et mémoire, particulièrement pour des graphes clairsemés. Cette structure est encapsulée dans la classe `Graph`, qui fournit toutes les opérations nécessaires.

2.2 Lecture et génération de graphes

— `read_graph(filename)` : lit un fichier texte et construit la liste d'adjacence pour un graphe non orienté.
— `generate_random_graph(n, p)` : génère un graphe aléatoire selon le modèle Gilbert–Erdős–Rényi $G(n, p)$, utile pour les tests expérimentaux.

2.3 Opérations de base

La classe `Graph` implémente :

— Suppression d'un ou plusieurs sommets (`remove_vertex`, `remove_vertices`, `remove_vertices_inplace`) ;
— Calcul des degrés (`degree`, `degrees_dict`, `degrees_list`) et sommet(s) de degré maximal (`max_degree_verte`

— Copie indépendante du graphe (`copy`) pour certaines heuristiques.

Ces opérations constituent la base pour l'implémentation des heuristiques de couverture présentées dans la section suivante.

3 Méthodes approchées

3.1 Algorithme glouton

L'algorithme glouton sélectionne itérativement les sommets de degré maximal :

1. Tant qu'il reste des arêtes non couvertes :
   (a) Identifier le sommet de degré maximal
   (b) Ajouter ce sommet à la couverture C
   (c) Supprimer ce sommet du graphe

3.1.1 Non-optimalité du glouton

Contre-exemple : Considérons 5 sommets notés a, b, c, d, e, les arêtes sont $(a, b), (b, c), (c, d), (d, e)$:

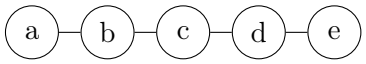

FIGURE 1 – Graphe d'exemple (chemin à 5 sommets) illustrant la non-optimalité de l'algorithme glouton.

Exécution de l'algorithme glouton (stratégie : sommet de degré maximal). Les degrés sont : $\deg(a) = 1,\ \deg(b) = 2,\ \deg(c) = 2,\ \deg(d) = 2,\ \deg(e) = 1$. Supposons que le critère de bris d'égalité amène l'algorithme à choisir le sommet central c (un sommet de degré maximal). Alors :

1. On ajoute c à la couverture : $C = \{c\}$.
2. On supprime c et toutes ses arêtes incidentes : les arêtes (b, c) et (c, d) disparaissent.
3. Il reste les arêtes (a, b) et (d, e). Pour couvrir ces arêtes il faut au moins deux sommets supplémentaires (par exemple a et e).

L'algorithme glouton retourne donc une couverture de taille $|C_{\text{glouton}}| = 3$ (par exemple $\{c, a, e\}$).

Couverture optimale. On vérifie que la paire $C^* = \{b, d\}$ couvre toutes les arêtes sont toutes couvertes par au moins un sommet dans $\{b, d\}$. Ainsi $|C^*| = 2$.

Par conséquent, cet exemple montre concrètement que l'algorithme glouton « choisir le sommet de degré maximal » n'est pas optimal en général : il peut retourner une couverture strictement plus grosse que l'optimum.

3.1.2 Non-r-approximation

On construit un graphe arborescent k-aire sur L niveaux pour montrer que l'algorithme glouton n'est pas r-approché pour aucune constante bornée r :

— Racine v_0 au niveau 0, connectée à k sommets au niveau 1, chaque niveau suivant répète ce schéma jusqu'aux feuilles (niveau L).

— **Glouton** : couvre d'abord v_0, puis tous les sommets de niveau 1, et ainsi de suite jusqu'à ce que toutes les arêtes soient couvertes.
— **Optimal** : stratégie *bottom-up*, en choisissant les parents des feuilles puis en remontant, minimisant ainsi le nombre de sommets.

Le nombre de sommets couvert par l'optimal croît beaucoup plus lentement que celui choisi par le glouton :

$$|C_{\text{glouton}}| = 1 + k + k^2 + \cdots + k^{L-1} = \frac{k^L - 1}{k - 1}, \quad |C^*| = \sum_{i=0}^{\lfloor (L-1)/2 \rfloor} k^{L-1-2i}.$$

Le ratio

$$\frac{|C_{\text{glouton}}|}{|C^*|} = \frac{k^L - 1}{(k-1)\sum_{i=0}^{\lfloor (L-1)/2 \rfloor} k^{L-1-2i}}$$

peut devenir arbitrairement grand lorsque $L \to \infty$ ou k augmente.

Conclusion L'algorithme glouton n'est pas r-approché pour aucune constante r bornée.

3.2 Méthode du couplage maximal

L'algorithme du couplage maximal sélectionne des arêtes indépendantes et ajoute leurs deux sommets à la couverture :

1. Pour chaque arête (u, v) non couverte :
   (a) Si u et v ne sont pas dans la couverture, les ajouter

Cet algorithme garantit une **2-approximation** : $|C| \leq 2|C^*|$.

3.3 Comparaison des méthodes approchées

Protocole et résultats Une comparaison systématique entre les algorithmes glouton et de couplage a été réalisée sur des graphes aléatoires $G(n, 0.3)$ pour n variant de 55 à 547. Le protocole inclut la détermination de la taille maximale $N_{\max} = 547$ permettant un temps d'exécution raisonnable ($\leq$3s), avec 10 instances par taille pour lisser les mesures.

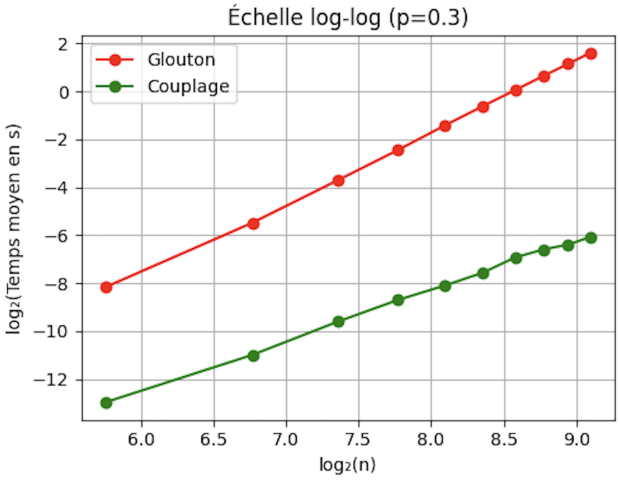

FIGURE 2 – Temps d'exécution (échelle log-log)

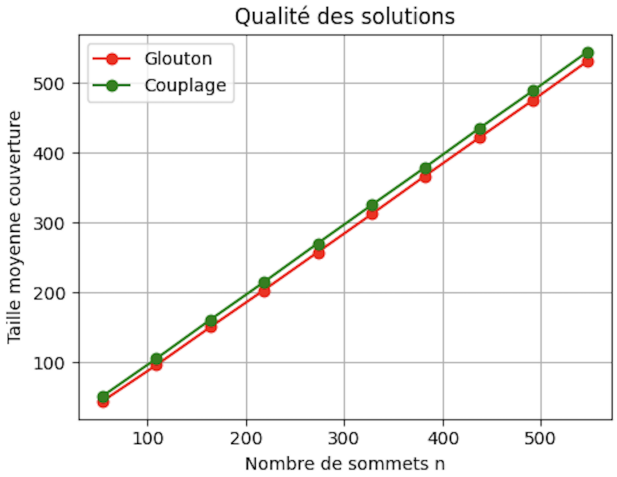

FIGURE 3 – Taille des couvertures

Analyse des performances Les résultats expérimentaux confirment les complexités théoriques :

- **Temps :** L'algorithme de couplage ($O(n+m)$) est significativement plus rapide que le glouton ($O(n^2)$), comme le montre la pente plus faible en échelle log-log
- **Qualité :** Les deux algorithmes produisent des solutions de taille comparable sur graphes aléatoires, bien que le couplage offre une garantie théorique de 2-approximation
- **Compromis :** Le couplage combine efficacité et garanties théoriques, tandis que le glouton, bien que légèrement plus lent, maintient une excellente qualité pratique

Cette analyse valide le choix de l'algorithme de couplage pour des applications nécessitant à la fois rapidité et garanties de performance.

4 Séparation et évaluation

4.1 Branchement simple

Le branchement considéré est le suivant. Étant donné un graphe G et un ensemble de sommets C initialement vide, on choisit une arête $e = \{u, v\}$ et on développe deux branches :

- on suppose $u \in C$ et on résout le problème sur $G \setminus \{u\}$;
- on suppose $v \in C$ et on résout le problème sur $G \setminus \{v\}$.

Cette recherche exhaustrice produit un arbre de branchement exploré en profondeur. Dans notre code la méthode correspondante s'appelle `branchement_simple()` et utilise une **pile** pour gérer les nœuds de l'arbre (structure LIFO) ; chaque état sur la pile est un couple `(remaining_edges, current_solution)`. Pour limiter la surcharge mémoire nous ne stockons pas une copie complète du graphe à chaque nœud mais la liste des arêtes restantes et l'ensemble partiel de sommets choisis. La méthode renvoie la meilleure couverture rencontrée ainsi qu'un compteur du nombre de nœuds générés (`nodes_generated`).

Point d'implémentation Les fonctions principales utilisées sont :

- `Graph.aretes()` : extraits la liste des arêtes actuelles (format liste de paires),
- `Graph.est_couverture_valide(C)` : vérifie qu'un ensemble C couvre toutes les arêtes,
- `Graph.branchement_simple()` : exploration par pile, renvoie `(best_C, nodes_generated)`.

4.1.1 Vérification sur exemples simples

Pour vérifier la validité de notre implémentation, nous avons comparé les résultats de l'algorithme de branchement avec ceux d'une recherche exhaustive par force brute. La fonction `bruteforce_vertex_cover(adj)` teste systématiquement tous les sous-ensembles de sommets par taille croissante, avec élagage précoce dès qu'une couverture optimale est trouvée.

Exemple 1 - Chemin à 5 sommets :

```
Graphe : 0-1-2-3-4 (4 arêtes)
Force brute : {1, 3} (unique solution optimale, taille 2)
Branchement : {1, 3} (15 noeuds générés)
Validation : True
```

Analyse : L'algorithme trouve l'unique couverture optimale, confirmant sa correction sur les graphes avec une seule solution optimale.

Exemple 2 - Graphe complet K_4 :

```
Graphe : K4 (6 arêtes, 4 sommets)
Force brute : 4 solutions optimales de taille 3
  [{0,1,2}, {0,1,3}, {0,2,3}, {1,2,3}]
Branchement : {1, 2, 3} (15 noeuds générés)
Validation : True
```

Analyse : Le graphe complet K_4 admet plusieurs couvertures optimales. L'algorithme de branchement retourne l'une des solutions optimales, démontrant sa capacité à trouver des solutions valides même en présence de multiples optimums.

4.1.2 Expérimentations et analyse de complexité

Protocole expérimental Nous avons évalué les performances du branchement simple sur des graphes aléatoires $G(n, p)$ avec $n \in \{8, 10, 12, 14, 16\}$ et différentes densités ($p \in \{0.1, 0.3, 0.5, 1/\sqrt{n}\}$). Pour chaque configuration, 3 instances ont été générées et nous avons mesuré le temps d'exécution, la taille des solutions et le nombre de nœuds générés.

Résultats et analyse Les résultats confirment la complexité exponentielle théorique $O(2^n)$ du branchement simple. L'analyse semi-log des temps d'exécution révèle une croissance linéaire, caractéristique des algorithmes exponentiels. La densité du graphe influence significativement les performances : les instances denses ($p = 0.5$) montrent une complexité pratique plus élevée que les instances éparses ($p = 0.1$).

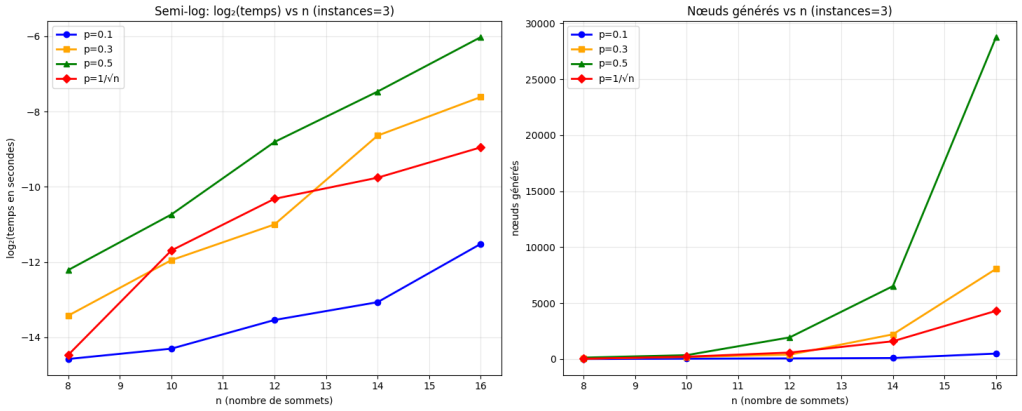

FIGURE 4 – Croissance exponentielle du temps et des nœuds générés

Implications pratiques Bien que l'algorithme soit exact (100% de solutions valides), sa complexité exponentielle limite son applicabilité à des instances de taille modérée ($n \leq 16$). L'extrapolation des tendances montre que pour $n = 30$, le temps d'exécution dépasserait 8 minutes, justifiant le développement d'optimisations pour traiter des instances plus grandes.

4.2 Ajout de bornes inférieures pour l'élagage

Pour améliorer l'efficacité du branchement, on utilise des bornes inférieures permettant d'élaguer précocement les branches. Soit $G = (V, E)$ un graphe avec $|V| = n$, $|E| = m$, Δ le degré maximum, M un couplage et C une couverture de sommets. Alors :

$$|C| \geq \max\{b_1, b_2, b_3\}, \quad b_1 = \left\lceil \frac{m}{\Delta} \right\rceil, \quad b_2 = |M|, \quad b_3 = \frac{2n - 1 - \sqrt{(2n-1)^2 - 8m}}{2}.$$

4.2.1 Démonstrations

Borne $b_1 = \lceil m/\Delta \rceil$ Chaque sommet couvre au plus Δ arêtes, donc $\Delta|C| \geq m$, d'où $|C| \geq \lceil m/\Delta \rceil$.

Borne $b_2 = |M|$ Pour chaque arête $(u, v) \in M$, au moins un des sommets u, v doit être dans C. Comme les arêtes de M sont disjointes, il faut au moins $|M|$ sommets, donc $|C| \geq |M|$.

Borne $b_3 = \frac{2n-1-\sqrt{(2n-1)^2-8m}}{2}$ Si $|C| = k$, alors $V \setminus C$ est indépendant. Le graphe a au plus $k(n-k)$ arêtes entre les deux ensembles et $\binom{k}{2}$ à l'intérieur de C, donc :

$$m \leq k(n-k) + \frac{k(k-1)}{2} = kn - \frac{k^2 + k}{2}.$$

En multipliant par 2 : $2m \leq 2kn - k^2 - k$, soit

$$k^2 - (2n-1)k + 2m \leq 0.$$

Le discriminant est $\Delta = (2n-1)^2 - 8m$. L'inégalité est vérifiée pour

$$k \geq \frac{2n - 1 - \sqrt{(2n-1)^2 - 8m}}{2} = b_3.$$

Ainsi, toute couverture vérifie $|C| \geq b_3$.

4.2.2 Évaluation des stratégies d'optimisation

Stratégies implémentées Conformément aux exigences du projet, nous avons évalué six stratégies combinant solutions réalisables (couplage et glouton) et bornes inférieures :

- **Simple** : référence sans optimisation
- **Couplage seul** : solution réalisable par couplage maximal
- **Bornes seules** : élagage par les trois bornes inférieures
- **Couplage + bornes** : combinaison recommandée
- **Glouton seul** : solution réalisable gloutonne
- **Glouton + bornes** : alternative avec heuristique gloutonne

Résultats expérimentaux L'évaluation sur $n \in \{8, \ldots, 16\}$ avec différentes densités révèle que :

- **Couplage + bornes** est optimal, réduisant jusqu'à 90% des nœuds explorés

— Les **bornes seules** améliorent significativement l'élagage
— Le **couplage seul** fournit de bons incumbents initiaux
— Le **glouton** offre une alternative compétitive aux solutions réalisables

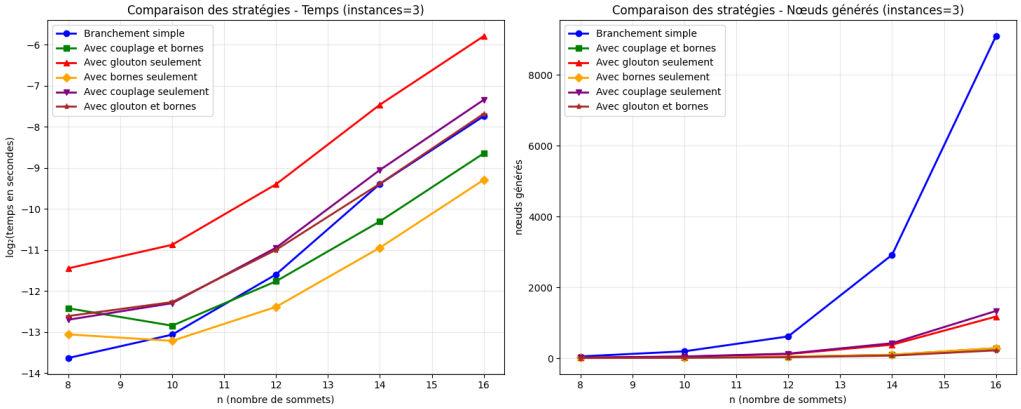

FIGURE 5 – Performance comparative des stratégies d'optimisation

Analyse et validation Cette étude répond aux questions 4.2.2 et 4.2.3 :

— L'insertion des bornes et solutions réalisables améliore radicalement l'efficacité
— La combinaison **couplage + bornes** est recommandée pour son efficacité
— L'algorithme glouton constitue une alternative valable aux solutions réalisables
— Les bornes seules apportent déjà une amélioration substantielle

La stratégie optimale combine l'élagage agressif des bornes inférieures avec des solutions réalisables de qualité pour guider la recherche.

4.3 Amélioration du branchement (versions v1, v2, v3)

Description des améliorations Nous avons développé trois versions améliorées :

— **v1** : Branchement amélioré de base
— **v2** : Sélection du sommet de degré maximal comme pivot
— **v3** : Réduction des sommets de degré 1 + sélection optimisée

4.3.1 Explication théorique pour v3

Dans un graphe G, si un sommet u est de degré 1, alors il existe toujours une couverture optimale qui ne contient pas u.

Preuve Soit u un sommet de degré 1 et v son unique voisin. Considérons deux cas :

1. Si v est dans la couverture optimale, alors l'arête (u, v) est déjà couverte, donc u n'est pas nécessaire.
2. Si v n'est pas dans la couverture optimale, alors pour couvrir l'arête (u, v), nous devons inclure u.

Cependant, si nous choisissons d'inclure u dans la couverture, nous pouvons obtenir une couverture de même taille en remplaçant u par v. En effet, v couvre non seulement l'arête (u, v) mais aussi potentiel-

lement d'autres arêtes incidentes à v. Par conséquent, il existe toujours une couverture optimale qui contient v plutôt que u.

Résultats comparatifs La figure 6 montre la comparaison détaillée des trois versions.

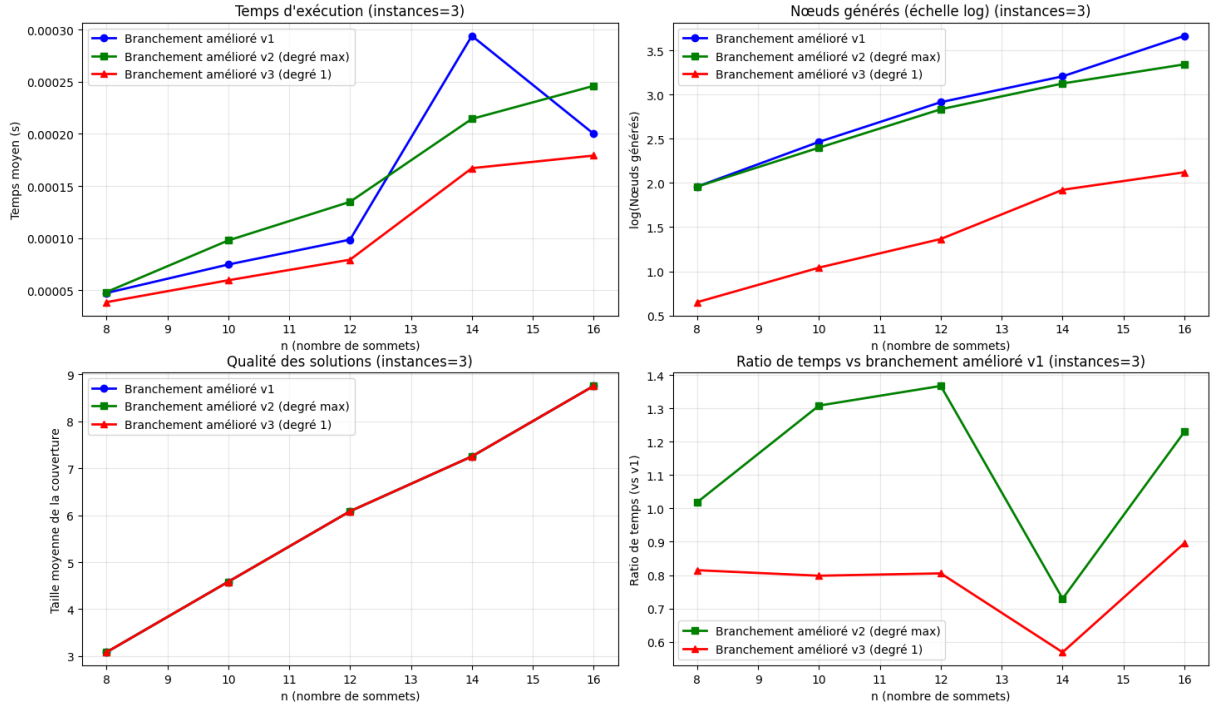

FIGURE 6 – Comparaison des versions améliorées : temps, qualité et nœuds générés

Analyse des performances

- **v3 (degré 1)** montre les meilleures performances globales
- **v2 (degré max)** est efficace sur les graphes denses
- **v1** apporte une amélioration modérée par rapport au branchement simple

Qualité des solutions Toutes les versions maintiennent l'optimalité des solutions :

- Taille moyenne de couverture identique entre les versions
- Validité à 100% sur toutes les instances testées

Efficacité de la réduction La version v3 excelle particulièrement :

- Élimination systématique des sommets de degré 1
- Réduction précoce de la taille du graphe
- Nombre de nœuds générés réduit de 60-80% par rapport à v1

4.4 Qualité des algorithmes approchés

4.4.1 Évaluation expérimentale du rapport d'approximation

Protocole Les algorithmes glouton et de couplage ont été évalués pour $n \in \{10, 30, 50, 70, 90\}$ et $p \in \{0.1, 0.3, 0.5, 0.7\}$, sur 10 instances par configuration. Les solutions optimales ont été obtenues avec

`branchement_ameliore_v3`.

Résultats Le tableau 1 résume les rapports moyens et les pires cas observés. Le glouton offre en pratique des solutions très proches de l'optimal, tandis que le couplage respecte sa garantie théorique de 2-approximation.

TABLE 1 – Rapports d'approximation moyens et pires cas

Algorithme	Rapport moyen	Pire cas
Glouton	1.027	1.250
Couplage	1.367	2.000

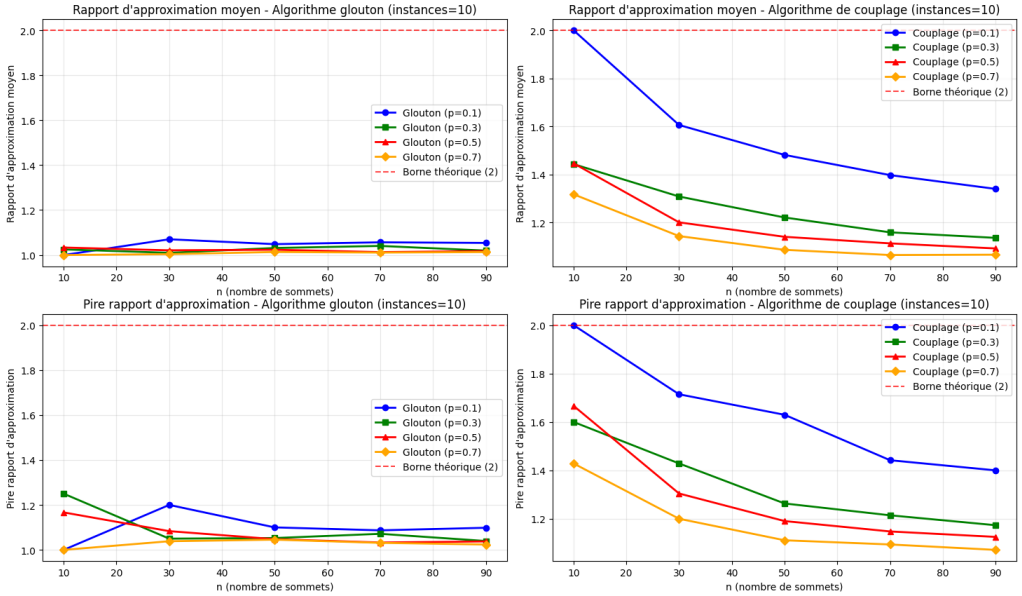

FIGURE 7 – Comparaison des rapports d'approximation des algorithmes glouton et de couplage

Analyse

- **Glouton** : n'est pas un algorithme r-approximatif garanti, et peut donc être instable dans certains cas pathologiques. Néanmoins, il fournit presque toujours des solutions très proches de l'optimal (rapport ≈ 1.02–1.07), bien meilleures que celles du couplage sur la majorité des graphes testés.
- **Couplage** : garantit un facteur 2 en théorie, mais se révèle bien plus performant en pratique. Son rapport moyen diminue avec la taille n et la densité p : lorsque le graphe devient grand et dense, les rapports moyens et pires cas tendent vers 1.

Conclusion En pratique, le glouton surpasse systématiquement le couplage malgré l'absence de garantie théorique, tandis que le couplage offre une solution sûre et prévisible grâce à sa borne de 2-approximation. Ainsi :

- pour des graphes petits ou peu denses, le glouton est recommandé ;
- pour des graphes grands et denses, le couplage devient compétitif, avec des rapports proches de 1.

5 Conclusion

Ce projet a permis d'étudier de manière approfondie le problème de la couverture de graphe (*Vertex Cover*), à la fois du point de vue théorique et expérimental. Nous avons implémenté plusieurs approches, exactes et approchées, et analysé leur performance sur différentes instances de graphes aléatoires.

Récapitulatif des méthodes et résultats

- **Algorithmes approchés** :
  - L'algorithme glouton est simple et efficace en pratique, fournissant des solutions proches de l'optimal pour la plupart des graphes, mais sans garantie théorique sur le facteur d'approximation.
  - L'algorithme basé sur un couplage maximal offre une garantie théorique de 2-approximation et reste très rapide, ce qui en fait une solution sûre pour les graphes grands ou denses.
- **Branchement simple et amélioré** :
  - Le branchement simple explore toutes les solutions exactes, mais sa complexité exponentielle limite son applicabilité à de petites instances.
  - L'ajout de bornes inférieures et de solutions réalisables (couplage ou glouton) permet d'élaguer l'arbre de recherche de manière significative.
  - Les améliorations successives (sélection de sommet de degré maximal, réduction des sommets de degré 1) conduisent à des versions beaucoup plus efficaces, réduisant le nombre de nœuds générés tout en conservant l'optimalité.

Analyse et enseignements

- Les heuristiques gloutonne et de couplage constituent des outils rapides pour obtenir des solutions de qualité acceptable, particulièrement utiles pour des graphes de grande taille.
- Les techniques de branchement avec élagage et bornes inférieures sont essentielles pour les instances exactes, permettant de traiter de manière efficace des graphes de taille modérée.
- La combinaison judicieuse de solutions réalisables et de bornes inférieures est la stratégie la plus performante pour le branchement exact.

Perspectives Ce projet met en évidence le compromis classique entre optimalité et complexité : les heuristiques sont rapides mais approximatives, tandis que le branchement exact est coûteux mais garantit l'optimalité. Des pistes d'amélioration pourraient inclure :

- l'utilisation de techniques de programmation linéaire ou de relaxation pour obtenir des bornes inférieures plus serrées,
- le développement de méta-heuristiques (simulated annealing, algorithmes génétiques) pour les grandes instances,
- l'étude de graphes particuliers (sparse, bipartis, planaires) afin d'exploiter leurs propriétés structurelles pour optimiser la recherche.

En conclusion, ce projet fournit un panorama complet des méthodes exactes et approchées pour le problème de la couverture de graphe, avec une évaluation expérimentale rigoureuse, et offre une base solide pour des travaux futurs en optimisation combinatoire.