



UNIVERSITÉ SORBONNE

MASTER 2 STL - INSTA

ANALYSE D'ALGORITHMES, GRAPHES ET APPLICATIONS

Accélérer l'algorithme de Girvan-Newman

Auteurs :

Merouane Bouafia
Anis Halfaoui
Rubén Coll Sánchez

Professeurs :

Lionel Tabourier
Mehdi Naïma

5 novembre 2025

Table des matières

1	Introduction	1
2	Context de base	1
3	Optimisations	2
3.1	Recalcul de la centralité sur les composantes des arêtes affectées	2
3.2	Omission du recalcu de la centralité	2
3.3	Approximation de la centralité par échantillonnage	3
4	Résultats pratiques	5
4.1	Ensembles de données	5
4.2	Experiences	5
5	Conclusion	7
6	Bibliographie	8

1 Introduction

L'algorithme de Girvan-Newman est une méthode classique de détection de communautés dans les graphes, fondée sur le calcul de la centralité des arêtes. Bien qu'efficace pour de petits réseaux, il devient rapidement trop lent pour les graphes de grande taille en raison du coût élevé de ce calcul répété à chaque itération, dont la complexité croît avec la taille du graphe.

Dans ce rapport, nous présentons une implémentation de l'algorithme standard, puis nous proposons une approche pour l'accélérer, comme la mise à jour partielle ou l'approximation des centralités. Enfin, nous évaluons les résultats de ces variantes sur différentes données, en comparant les mesures de performance et la qualité des partitions mesurée par la modularité.

2 Context de base

L'algorithme de Girvan-Newman est une méthode de détection de communautés reposant sur la suppression progressive des arêtes ayant la plus forte centralité intermédiaire ou edge-betweenness. Cette centralité mesure la taux de plus courts chemins qui passent par une arête donnée ; elle évalue donc son importance dans la connectivité entre les communautés du graphe.

Le fonctionnement général de l'algorithme est le suivant :

1. Calculer la centralité pour toutes les arêtes du graphe.
2. Supprimer l'arête ayant la valeur la plus élevée.
3. Calculer la modularité de la partition actuelle.
4. Recalculer les centralités après chaque suppression.
5. Répéter le processus jusqu'à ce que toutes les arêtes soient supprimées.
6. Choisir la partition avec la valeur de modularité la plus élevée.

Chaque suppression pouvant diviser le graphe en plusieurs composantes connexes, on obtient ainsi un dendrogramme représentant les communautés à chaque niveau de découpage.

Algorithm 1 Algorithme de Girvan-Newman [1]

- 1: **Entrée** : Un graphe non orienté $G = (V, E)$
 - 2: **Sortie** : Un ensemble hiérarchique de communautés
 - 3: **while** E n'est pas vide **do**
 - 4: Calculer la *centralité intermédiaire* de chaque arête $e \in E$
 - 5: Trouver l'arête e_{max} ayant la plus grande centralité
 - 6: Supprimer e_{max} du graphe G
 - 7: Enregistrer la nouvelle partition communautaire
 - 8: **end while**
 - 9: Construire le *dendrogramme* représentant les différentes partitions obtenues
-

Le calcul complet de la centralité pour toutes les arêtes à chaque étape entraîne un coût en temps de l'ordre de $O(|E| \cdot (|V| + |E|))$ pour un graphe non orienté. Par conséquent, l'algorithme

de Girvan Newman présente une complexité temporelle de $O(|E|^2 \cdot (|V| + |E|))$, et ça le rend peu adapté aux grands réseaux.

Pour cela, il est primordial de réfléchir à une solution plus performante afin de pouvoir l'appliquer à des graphes de grande ampleur, reflétant les réseaux à l'échelle réelle.

Graphes	Noeuds	Arêtes	Temps (approx.)
Club de karaté de Zachary	34	78	~ 1 s
Réseau politique (Blogs US)	1 222	16 714	~ 10 min
Réseau social Facebook (échantillon)[3]	4 039	88 234	~ 45 min
Réseau Twitter (sous-graphe)	81 306	1 768 149	> 10 h

TABLE 1 – Exemples de temps d'exécution de l'algorithme de Girvan–Newman

3 Optimisations

Dans la section suivante, nous présentons une série de modifications possibles à l'algorithme de Girvan-Newman, dans le but d'accélérer son exécution. Certaines consistent en de petits changements qui rendent l'algorithme plus rapide dans des cas particuliers, tandis que d'autres permettent de réduire l'ordre général de complexité au détriment d'une réduction de la précision des résultats.

3.1 Recalcul de la centralité sur les composantes des arêtes affectées

Une approche mentionnée dans le même article que l'algorithme de Girvan–Newman consiste à réduire le coût du recalcul de la centralité après chaque suppression d'arête.

Fonctionnement

En effet, cette solution reprend le même fonctionnement de l'algorithme Girvan-Newman, à un détail près. Plutôt que de recalculer la centralité pour l'ensemble du graphe, les auteurs suggèrent de ne recalculer la centralité que sur les composantes connexes directement affectées par la suppression de l'arête ayant la plus grande valeur de centralité.

Cette optimisation repose sur le fait qu'après la suppression d'une arête, le graphe se divise souvent en plusieurs sous-graphes indépendants. Ainsi, seules les arêtes appartenant à ces nouvelles composantes ont besoin d'une mise à jour de leur valeur de centralité, car les chemins les plus courts dans les autres sous-parties du graphe restent in affectés.

Cela permet donc, dans le meilleur des cas, de légèrement réduire le nombre de calculs nécessaires à chaque itération de l'algorithme, tout en conservant la même précision dans la détection des communautés.

3.2 Omission du recalcul de la centralité

Les auteurs de l'article de Girvan-Newman évoquent une proposition : au lieu de calculer la centralité à chaque itération. Pour ce faire, on calcule les valeurs une première fois seulement,

ensuite, on supprime les arrêtes une par une par ordre décroissant de leur valeur de centralité.

Malgré que cette solution permet de réduire considérablement la complexité de l'algorithme de base, les auteurs découragent fortement son utilisation [1] vu que cette méthode ne fonctionne pas correctement avec des communautés reliées par plus d'une arête.

C'est pourquoi nous ne l'avons ni mis en œuvre ni testé.

3.3 Approximation de la centralité par échantillonnage

Une approche prometteuse pour accélérer l'algorithme de Girvan-Newman consiste à remplacer le calcul de la centralité pour des méthodes d'approximation par échantillonnage (Riondato et al., [2]), qui permettent d'estimer la centralité de manière efficace sans parcourir tous les plus courts chemins du graphe.

Fonctionnement

La méthode consiste à échantillonner aléatoirement un certain nombre de chemins les plus courts, et pour chaque chemin sélectionné, à augmenter la BC approximative de ses arêtes d'une constante. Intuitivement, une arête avec une BC plus élevée sera sélectionnée plus souvent, car elle fait partie d'un plus grand nombre de chemins les plus courts, et donc sa BC approximative sera également plus élevée.

Formellement, le fonctionnement de la méthode est définie par un théorème qui permet de relier les hyperparamètres de la méthode à la précision obtenue. :

Théorème 1. *Soit $G = (V, E)$ un graphe, $\varepsilon \in (0, \infty)$ la marge d'erreur, $\delta \in (0, 1)$ la probabilité d'erreur, et $c \in (0, \infty)$ constante. Soit $b_c : E \rightarrow [0, 1]$ la mesure de centralité intermédiaire et $b'_c : E \rightarrow [0, 1]$ l'approximation obtenu par la méthode. Alors,*

$$P(|b_c(v) - b'_c(v)| < \varepsilon, \forall v) \geq 1 - \delta$$

Démonstration. Voir le théorème 2 de [2]. □

Le déroulement du calcul peut être résumé comme suit :

1. **Calcul de la taille d'échantillon r :** La taille de l'échantillon r est déterminée selon la formule suivante :

$$r = \left\lceil \frac{c}{\varepsilon^2} \left(\lfloor \log_2(VD(G) - 2) \rfloor + 1 + \ln \frac{1}{\delta} \right) \right\rceil$$

où $VD(G)$ est le diamètre en sommets du graphe, et les paramètres ε (précision souhaitée), δ (probabilité d'erreur) et c (constante). Coût : $O(1)$

Si le diamètre du graphe est inconnu, il est calculé à partir d'une méthode approximative, également par échantillonnage. Chaque échantillon choisit un nœud au hasard et prend la somme des deux chemins les plus longs parmi tous les chemins les plus courts à partir de celui-ci. La moyenne des échantillons est renvoyée. Dix échantillons sont prélevés, ce

qui fait que le coût de cette étape devient $O(|V| + |E|)$. On peut démontrer que chaque échantillon prend une valeur comprise entre $VD(G)$ et $2 \cdot VD(G)$ ([2]).

2. **Precalcul des composantes connectés avec plus d'une sommet** : Ça permet de sélectionner aléatoirement des paires de sommets de façon efficient. Coût : $O(|V| + |E|)$
3. **Échantillonnage de paires de sommets** : Répéter r fois :
 - (a) **Sélection aléatoire** : choisir uniformément une paire (u, v) de sommets distincts du graphe dans la même composant connexe. Coût : $O(1)$ moyen à cause de la randomisation.
 - (b) **Calcul des plus courts chemins** : Pour cette paire (u, v) , déterminer l'ensemble S_{uv} des plus courts chemins reliant ces deux sommets. Coût : $O(|V| + |E|)$
 - (c) **Sélection d'un chemin aléatoire** : Choisir uniformément un chemin p dans S_{uv} . Pour ce faire, on part du noeud v et on choisit l'un de ses prédécesseurs p_1 . Pour celui-ci, on choisit un autre de ses prédécesseurs, p_2 , et ainsi de suite jusqu'à atteindre u . À chaque choix, si n est le dernier noeud choisi, chaque prédécesseur p de n a une probabilité de $\frac{\sigma_p}{\sigma_n}$. Cela garantit que la probabilité de choix de chaque chemin de u à v est uniforme ([2]). Coût : $O(|V| + |E|)$
 - (d) **Mise à jour des estimations** : Incrémenter de $1/r$ la centralité approximée de chaque arête appartenant au chemin p . Coût : $O(|V|)$

Ainsi, après r itérations, les valeurs de centralité estimées convergent vers une approximation fiable de la centralité intermédiaire exacte. En conséquence on obtient une complexité de $O(r \cdot (|V| + |E|))$.

Algorithm 2 Approximation de la centralité par échantillonnage [2]

- 1: **Entrée** : Graphe non orienté $G = (V, E)$, nombre d'échantillons k
 - 2: **Sortie** : Approximation de la centralité d'intermédiaire de chaque arête
 - 3: **for** $i = 1$ à k **do**
 - 4: Choisir aléatoirement une paire de sommets (s, t) dans V
 - 5: Calculer un plus court chemin P entre s et t
 - 6: **for** chaque arête e appartenant à P **do**
 - 7: Incrémenter la centralité approximée de e : $C[e] \leftarrow C[e] + \frac{1}{r}$
 - 8: **end for**
 - 9: **end for**
-

Pour adapter l'algorithme de Girvan Newman, le calcul exact de la centralité intermédiaire est remplacé par la méthode approximative. La complexité finale de l'algorithme adapté passe de $O(|E|^2 \cdot (|V| + |E|))$ à $O(|E| \cdot r \cdot (|V| + |E|))$. Même dans les graphes les plus grands que l'on peut trouver dans des ensembles de données réels, le diamètre du graphe n'est généralement pas supérieur à 50, de sorte que l'on peut même supposer qu'avec ce changement, l'ordre de complexité devient $O(|E| \cdot (|V| + |E|))$.

4 Résultats pratiques

Ce chapitre est consacré à l'évaluation expérimentale de l'algorithme de Girvan–Newman classique, ainsi que de la version optimisée fondée sur l'échantillonnage, la plus prometteuse.

Nous cherchons à mesurer les gains en performance obtenus grâce aux optimisations proposées, tout en analysant leur impact sur la qualité de la détection de communautés.

4.1 Ensembles de données

Pour cela, plusieurs graphes réels et synthétiques de tailles variées ont été utilisés afin d'observer le comportement des algorithmes selon la complexité du réseau et la densité des connexions. Nous avons testé principalement avec 3 type de graphes allant de simple, moyen à complexe.

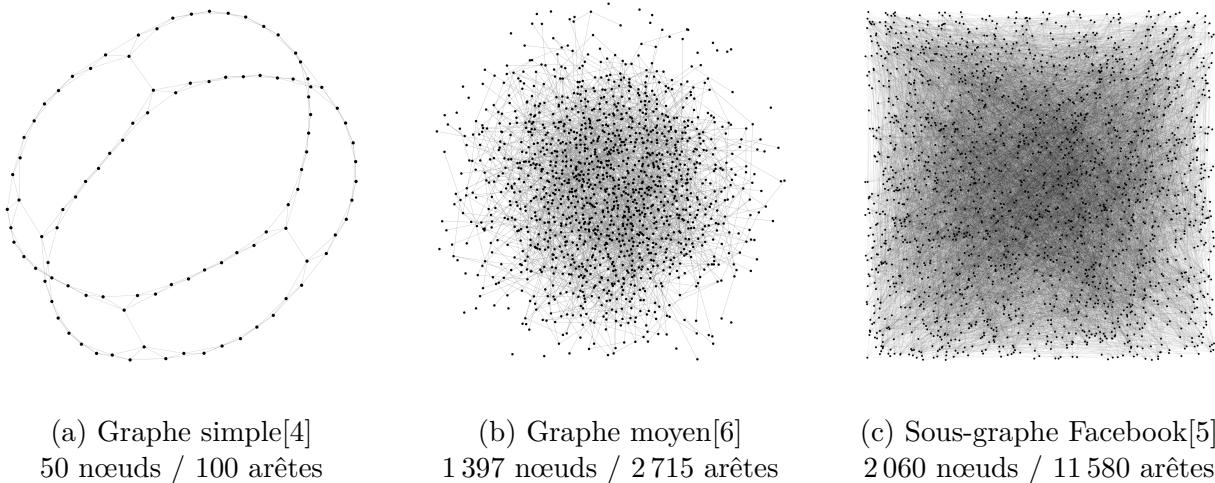


FIGURE 1 – Jeux de données de test.

4.2 Experiences

Afin d'évaluer les performances de l'algorithme de cette variation d'échantillonnage de l'algorithme de Girvan–Newman. On observe concrètement l'impact de la taille du graphe et des optimisations proposées sur le temps d'exécution et la qualité des partitions détectées.

Exemple

La valeur de la modularité Q de la meilleure partition, pour le graphe simple[4] de 50 noeuds et 100 arêtes, en utilisant l'algorithme de Girvan-Newman standard est de :

$$Q = 0.7332897085068407$$

Tandis qu'avec la méthode d'échantillonnage, vu qu'il le processus est non déterministe, on obtient un intervalle mais qui reste cohérent et assez proche de la valeur exacte :

$$Q' = [0.68 \sim 0.73]$$

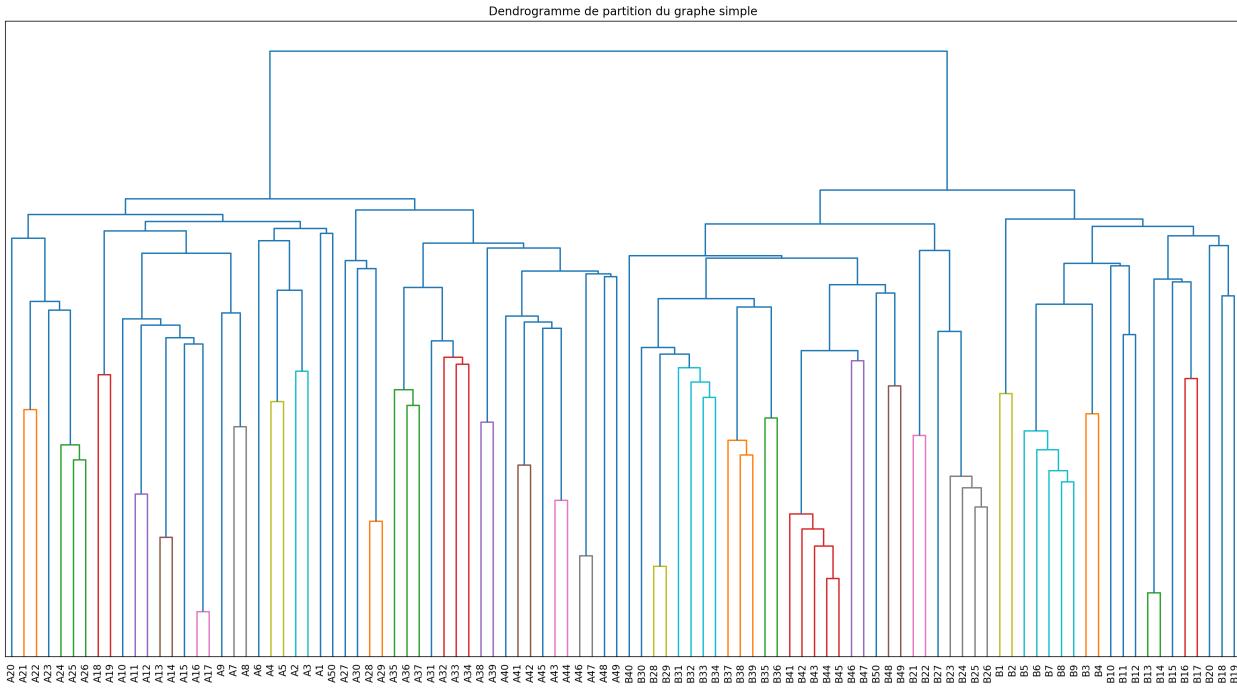


FIGURE 2 – Dendrogramme de partition du graphe simple[4]

Ceci démontre que l’approximation utilisée par l’algorithme dérivé reste assez proche de la valeur réelle, avec un faible taux d’erreur ; l’intervalle obtenu traduit la variabilité liée au processus d’échantillonnage (sélection aléatoire des paires et des chemins), et dépend des paramètres de précision et de confiance (ε, δ) choisis. En pratique, cette légère incertitude peut être compensée en augmentant r (le nombre d’échantillons) ou en moyennant plusieurs exécutions indépendantes : on obtient alors une estimation de la modularité avec une borne d’erreur plus étroite, tout en conservant un gain de temps significatif par rapport au calcul exact. Pour l’utilisateur, cela signifie que l’on peut accepter une petite perte de précision sur Q en échange d’une scalabilité bien meilleure sur des graphes de grande taille.

ε	δ	c	r	Modularité Q
0.1	0.3	1.0	421	~ 0.4723
0.1	0.1	1.0	531	~ 0.4750
0.1	0.3	0.5	211	~ 0.4717
0.3	0.3	1.0	47	~ 0.4744

 TABLE 2 – Effet des paramètres sur r et Q pour le graphe moyen[6].

Graphes	Noeuds	Arêtes	Temps standard	Temps échantill.
Graphe simple[4]	50	100	~ 211 ms	~ 183 ms
Graphe moyen[6]	1 397	2 715	~ 3 min	~ 3 s
Sous-graphe Facebook[5]	2 060	11 580	-	~ 15 min

TABLE 3 – Comparaison des temps d’exécution avec les jeux de données testés

On observe que sur des graphes à petite échelle, l'échantillonnage réduit légèrement le temps d'exécution par rapport à l'algorithme de Girvan-Newman standard, mais qui restent assez proches. Pour des graphes plus volumineux, comme le sous-graphe Facebook, l'algorithme standard n'est plus praticable, tandis que l'échantillonnage reste réalisable dans un temps acceptable. Cela illustre l'efficacité de l'approximation pour des réseaux de moyenne à grande taille.

5 Conclusion

Dans ce rapport, nous avons présenté l'algorithme de Girvan-Newman pour la détection de communautés dans les graphes, ainsi que ses limitations en termes de complexité pour des réseaux de grande taille. Nous avons ensuite étudié une approche d'approximation basée sur l'échantillonnage, qui permet d'estimer efficacement la centralité des arêtes tout en réduisant significativement le temps de calcul. Pour ensuite implémenter l'algorithme de base de Girvan-Newman ainsi que la solution proposée de l'échantillonnage[7].

Les expériences menées par les auteurs de cette solution démontrent déjà le gain d'efficacité apporté, et qui est plus renforcé par notre propre expérience sur différents graphes, allant d'un petit réseau de 50 noeuds à un sous-graphe Facebook de plus de 2 000 noeuds, montrent que l'algorithme par échantillonnage conserve une qualité de partition approximative comparable à l'algorithme de base, tout en étant beaucoup plus rapide et scalable. Cette approche rend ainsi possible l'application de Girvan-Newman à des réseaux de moyenne à grande taille, là où l'algorithme standard devient impraticable.

En conclusion, l'approximation par échantillonnage constitue une solution efficace, précise et performante dans la détection de communautés dans les graphes volumineux à échelle réelle.

6 Bibliographie

- [1] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12) :7821–7826, 2002.
- [2] Matteo Riondato and Evgenios M. Kornaropoulos. Fast approximation of betweenness centrality through sampling. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 413–422. IEEE, 2014.
- [3] Emre Okcular. Facebook social network data. *GitHub repository*, https://github.com/emreokcular/social-circle/blob/main/facebook_combined.txt.gz, 24-10-2025.
- [4] Merouane Bouafia and Anis Halfaoui and Rubén Coll Sánchez. Fichier de test de données du graphe simple. *Fichier graphe de test*, <https://github.com/TheNuber/AAGA-Projet/blob/main/data/sample.edgelist>, 17-10-2025.
- [5] Merouane Bouafia and Anis Halfaoui and Rubén Coll Sánchez. Fichier de test de données du sous-graphe Facebook. *Fichier graphe de test*, https://github.com/TheNuber/AAGA-Projet/blob/main/data/facebook_combined.txt, 04-11-2025.
- [6] Merouane Bouafia and Anis Halfaoui and Rubén Coll Sánchez. Fichier de test de données du graphe moyen. *Fichier graphe de test*, <https://github.com/TheNuber/AAGA-Projet/blob/main/data/graphmoyen.txt>, 05-11-2025.
- [7] Merouane Bouafia and Anis Halfaoui and Rubén Coll Sánchez. Betweenness sampling algorithm implementation. *GitHub repository*, <https://github.com/TheNuber/AAGA-Projet/tree/main>, 05-11-2025.