HEC MONTRÉAL

Algorithme séquentiel pour la détection de communautés à l'aide d'une interprétation matricielle de la modularité

par

Victor Putureanu

Sciences de la gestion
(Option Analytique d'affaires)

Mémoire présenté en vue de l'obtention
du grade de maîtrise ès sciences en gestion
(M. Sc.)

Août 2020

© Victor Putureanu, 2020

Résumé

La détection de communautés vise à identifier des structures dans les réseaux complexes. Ces structures sont importantes pour comprendre les caractéristiques des réseaux, les individus qui la composent ainsi que leurs interactions. Présentement, la détection de communautés disjointes dans les réseaux non directionnels est un problème de la classe NP-Difficile. Ce mémoire explore une interprétation matricielle de la modularité avec une approche en plusieurs étapes en utilisant la modularité pour évaluer l'efficacité de l'algorithme. L'algorithme proposé implémente 4 étapes en stades avec un retour au début lorsqu'une amélioration est détectée. Une initialisation est effectuée en parallèle pour tous les nœuds et regroupée dans les communautés semblables afin d'avoir une solution de départ. La première étape maximise la modularité au niveau des nœuds. La deuxième étape divise les communautés afin d'avoir un gain de modularité suivi de la troisième étape de fusionner des groupes. La dernière étape est une approche qui analyse des pairs de nœuds pour des gains. Les résultats illustrent une qualité efficace dans la maximisation de la modularité pour les petits réseaux jusqu'à 512 nœuds. Les résultats illustrent une valeur de modularité élevée, soit très proche de l'optimalité avec une concentration de certaines étapes à des moments clés.

Mots clés : Détection de communautés, Modularité, Matrice de modularité, Réseaux complexes, Matrice de modularité, apprentissage non supervisé.

Abstract

Community detection aims at finding structures in complex networks. These structures are important for understanding the characteristics of networks, the individuals who compose them and their interactions. Currently, disjointed community detection in non-directional networks is a NP-Difficult class problem. This thesis explores a matrix interpretation of modularity utilizing a multi-step approach using modularity to evaluate the algorithm's effectiveness. The proposed algorithm implements 4 steps in stages with a return to the first step when an improvement is detected. An initialization is carried out in parallel for all nodes and grouped in similar communities to have a solution from start. The first step maximizes modularity at the level of individual nodes. The second stage divides communities to detect a gain of modularity followed by the third step of merging groups. The final step is an approach that analyzes sets of nodes for gains. The results illustrate an effective quality in maximizing modularity for small networks up to 512 nodes. The results illustrate a high modularity value, very close to optimality with an concentration of certain stages at key moments in time.

Keywords: Community detection, modularity, Modularity matrix, multi-step process, complex networks, partitioning, modularity matrix, unsupervised learning.

Table des matières

Résumé ..... **iii**
Abstract ..... **v**
Table des matières ..... **vii**
Liste des figures ..... **ix**
Liste des tableaux ..... **xi**
Liste des algorithmes ..... **xiii**
Liste des abréviations, sigles et symboles ..... **xv**
Avant-propos ..... **xvii**
Remerciements ..... **xix**
Chapitre 1 : Introduction **1**
1.1. Théorie des graphes et des réseaux 1
1.2. Applications de la détection de communautés 3
1.3. Représentations des réseaux 4
Chapitre 2 : Revue de la littérature **7**
2.1. Classe de problème algorithmique 7
2.2. Taille du problème : Les chiffres de Bell 8
2.3. Indicateurs de communautés dans les réseaux complexes 9
2.4. Classes d'algorithmes de maximisation de la modularité 12
2.5. Liste de techniques les plus communs 12
Chapitre 3 : Définition de la problématique **19**
3.1. Paramètres du problème 19
3.2. Transformation en matrice de modularité et équivalences 19
3.3. Propriétés de la matrice de modularité 23
3.4. Fonction objective de l'algorithme 28
3.5. Indices de performance 29
Chapitre 4 : Méthodologie **31**
4.1. Méthodologie de développement 31
4.2. Initialisation 33
4.3. Étape 1 : Glouton 35
4.4. Étape 2 : Division de groupes 37
4.5. Étape 3 : Fusion de groupes 43
4.6. Étape 4 : Glouton compressé 45
Chapitre 5 : Résultats **47**
5.1. Temps de calcul 47
5.2. Qualité des résultats 49
5.3. Performance des étapes 50
5.4. Trajectoire de l'algorithme 52
5.5. Facteurs qui influencent la performance et la qualité 55
Chapitre 6 : Conclusion **57**
6.1. Résumé 57
6.2. Limites et travaux à venir 59
Bibliographie **61**

Liste des figures

Figure 1 : Représentation d'un réseau simple 2
Figure 2 : Exemple du réseau complexe *les misérables* 2
Figure 3 : Comparaison entre la forme matricielle et réseau 5
Figure 4 : Équivalence avec la liste d'adjacence 6
Figure 5 : Comparaison du réseau *Les Misérables* brute et partitionné en fonction de la modularité. 7
Figure 6 : Comparaison de la croissance du nombre de partitions avec d'autres fonctions 8
Figure 7 : Représentation graphique des approches à la détection de communauté 11
Figure 8 : Graphique conceptuel de la précision des algorithmes les plus populaires en fonction de la complexité 14
Figure 9 : Exemple du calcul de la modularité et la matrice de modularité 21
Figure 10 : Représentation simplifiée de la somme de la modularité 26
Figure 11 : Représentation simplifiée de la modularité et extraction de deux groupes 27
Figure 12 : Tableau de la comparaison des équivalences matricielles pour *Les misérables* 28
Figure 13 : Description de l'algorithme. 32
Figure 14 : Représentation simplifiée de l'étape du glouton 36
Figure 15 : Exemple d'un graphique des valeurs de séparation d'un groupe qui a une valeur ajoutée à la séparation 38
Figure 16 : Représentation graphique des formules de la division de groupes 39
Figure 17 : Graphiques simplifiés des itérations de la division de groupes 40
Figure 18 : Illustration d'un exemple des valeurs graphiques d'une séparation de groupe 41
Figure 19 : Graphique matriciel simplifié des aires pour la fusion de groupe 44
Figure 20 : Représentation des arcs adjacents et les arcs non adjacents 45
Figure 21 : Temps de calcul en fonction de la taille du réseau 48
Figure 22 : Graphique de la comparaison entre 4 réseaux sur l'amélioration en fonction du temps standardisé[1] à la valeur de départ 50
Figure 23 : Graphique de la comparaison de 4 réseaux de l'amélioration en fonction du temps standardisé[1] à la valeur de départ pour les premiers 300 milisec. 51
Figure 24 : Amélioration de la modularité en fonction du temps (ms) avec les étapes indiquées pour *Electronic Circuit (s838)* 52
Figure 25: Amélioration de la modularité en fonction du temps (ms) avec les étapes indiquées pour *Netscience_main* 53
Figure 26: Amélioration de la modularité en fonction du temps (ms) avec les étapes indiquées pour *USair97* 53
Figure 27: Amélioration de la modularité en fonction du temps (ms) avec les étapes indiquées pour *A01 Main* 54
Figure 28 : Représentation visuelle du problème de cycle 60

Liste des tableaux

Tableau 1 : Tableau des réseaux analysés avec les données de [1] 31
Tableau 2 : Temps de calcul en fonction de la taille du réseau..... 47
Tableau 3 : Tableau des résultats en *Q* de chaque réseau en comparaison avec le *Q* optimal..... 49

Liste des algorithmes

Algorithme 1 : Pseudo-code de la procédure d''initialisation 34

Liste des abréviations, sigles et symboles

1. n : Nombre de nœuds
2. m : Nombre d'arcs
3. k_i : Degré du noeud i (Nombre d'arc)
4. C_i : Classe du nœud i
5. Q : Modularité
6. Q_c : Modularité d'une classe
7. ΔQ : Delta Q
8. Q^- : Somme des arcs entre les classes
9. Q_{algo} : Modularité de l'algorithme
10. Q_{opt} : Modularité avec partition optimale
11. δ : Delta de kronecker
12. A_{ij} : Matrice d'adjacence
13. B_{ij} : Matrice de modularité
14. G : Réseaux (input)
15. $F_i(G)$: Fonction i sur le réseau G
16. $P_i = F_i(\mathrm{G})$: Partition i obtenue de l'algorithme i
17. $Z(P_i)$: Mesure de la qualité de la partition i
18. a : Scalaire qui représente la somme de la modularité du groupe 1
19. b : Scalaire qui représente la somme de la modularité du groupe 2
20. c : Scalaire qui représente la somme de la modularité de l' entre-deux groupes
21. g_1 : Emplacement du dernier noeud du groupe *1*
22. $QV_{i\,l\,k}$: Contribution net du nœud i du groupe l vers le groupe k
23. $QG_{\ i\,k}$: Contribution direct du noeud i au groupe k
24. $QG_{\ i\,l}$: Contribution direct du noeud i au groupe l
25. $O\big(g(n)\big)$ *:* Complexité d'un algorithme

Avant-propos

L'objectif initial de ce projet de mémoire a été une exploration du domaine de la détection de communauté. La maximisation de la modularité est un problème algorithmique fascinant vu qu'il n'y a pas de solution parfaite ou une approche satisfaisante qui est découverte à ce jour. Lors de l'exploration d'un problème qui apparaît intuitif à première vue, la maximisation de la modularité possède plusieurs facettes cachées et énigmatiques.

Pendant l'exploration du problème, j'ai créé un algorithme afin d'explorer davantage la modularité et la détection de communauté. Cet algorithme initial était différent que celles proposées par la littérature et fournissait des partitions intéressantes. Après plusieurs discussions avec Gilles Caporossi et Sylvain Perron sur l'algorithme, ce projet a pris naissance afin de comprendre l'algorithme, le codage et la détection de communauté.

Remerciements

Je tiens à remercier mes deux directeurs de recherche, Sylvain Perron et Gilles Caporossi pour m'avoir orienté et conseiller tout au long de la recherche malgré tous leurs tâches aux HEC. Je les remercie pour tout le temps qu'ils ont affecté pour discuter de mon projet de mémoire ainsi que les discussions d'algorithme très captivant et palpitant.

Je tiens à remercier mes parents pour leur support moral constant et inconditionnel tout au long de cette réalisation. De plus, je remercie le partage de leur sagesse lors des moments difficiles de ce manuscrit.

Chapitre 1 :
Introduction

Les communautés et les réseaux possèdent des structures internes et externes par leurs liens entre elles. L'étude des réseaux qui se retrouve naturellement dans tous les domaines révèle des informations énigmatiques et pragmatiques de leurs fonctionnements et interactions. Certains réseaux sont plus évidents, tandis que d'autres sont plutôt une abstraction humaine résultant d'une représentation mathématique. En bref, tout objet à un réseau plus ou moins complexe associé à ces interactions. Par exemple, un réseau de neurones est un exemple évident d'un réseau complexe qui forme la pensée chez les êtres vivants. De l'autre côté, un réseau social est une abstraction mathématique des interactions des individus. Les réseaux sociaux n'ont pas un lien physique ou une substance qui connecte les individus. Cependant, le réseau est issu des interactions entre les particuliers.

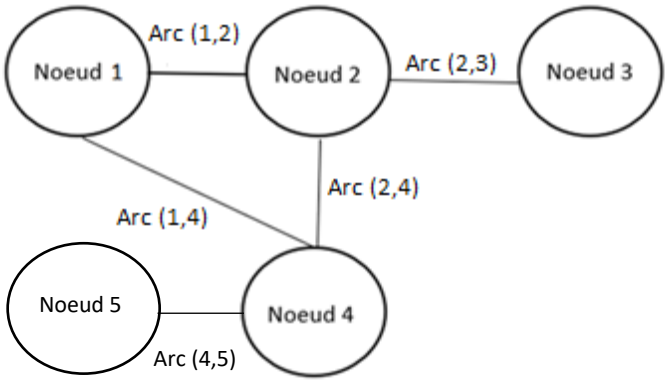
Les réseaux permettent d'étudier les phénomènes macroscopiques émergents des interactions et des liens entre les nœuds. Cette approche est partagée par plusieurs disciplines scientifiques et industrielles afin d'explorer des populations ou des phénomènes. Dans certains cas de manière exploratoires et d'autres pour retrouver des solutions. Par exemple, le domaine des mathématiques appliquées utilise couramment une représentation de réseaux pour simplifier et optimiser des solutions telles que le plus court chemin, le voyageur de commerce, la détection de communautés, etc. Le domaine de la biologie étudie les réseaux de neurones qui forment la conscience et les interactions de protéines par exemple.

1.1. Théorie des graphes et des réseaux

Les réseaux sont une représentation mathématique et visuelle d'un ensemble de systèmes et leurs liens entres-elle. Cette représentation est un objet mathématique défini par un ensemble de nœuds et d'arcs interconnectés qui forme un graphe. Les réseaux complexes sont une classe de réseaux qui ont la particularité d'avoir des structures topologiques non

triviales en lien avec la représentation. Une définition de quelques concepts clés est requise afin d'améliorer la compréhension de ce mémoire ainsi que réduire la confusion sur le vocabulaire parfois employé dans d'autres disciplines :

Figure 1 : Représentation d'un réseau simple



Définition :

Nœud : Objets mathématiques qui représentent un point potentiellement relié par un arc. La représentation graphique est normalement un cercle avec son identifiant comme la figure 1. Un nœud est aussi appelé sommet. La notation d'un graphe est *G(V,E)*, où *V* est l'ensemble des nœuds (*n*) et *E* est l'ensemble des arcs (*m*).

Arc : Un arc est un lien entre deux nœuds représentés par une ligne sur un graphe. Cette ligne illustre l'interaction entre deux nœuds qui peut être non directionnel ou directionnel avec ou sans valeur. Exemple dans la figure 1.

Figure 2 : Exemple du réseau complexe *les misérables*

Degré d'un nœud : Le degré d'un nœud est la somme des arcs connectés à un nœud. En autres mots, le degré d'un nœud est le nombre de liens avec d'autres nœuds il possède. Dans la suite du texte, le degré du nœud i sera noté par ki.

Réseaux : Un réseau est formé de nœuds connectés par des arcs entre eux qui représentent un ensemble de connexions ou d'interactions entre plusieurs nœuds. La figure 1 représente un réseau où les nœuds sont représentés par des cercles et les arcs par des lignes.

Réseaux complexes : Les réseaux complexes sont un ensemble de réseaux qui possède une ou plusieurs structures non triviale et non aléatoire. La figure 2 illustre un exemple d'un réseau complexe structuré en fonction de la modularité.

Communautés : Une communauté est une structure particulière d'un réseau complexe qui regroupe un nombre de nœuds avec des arcs entre eux qui se distingue de l'ensemble du réseau. Les couleurs du réseau de la figure 2 représentent les différentes communautés dans le réseau. Les communautés ont la propriété d'avoir une densité d'arcs plus élevée entres-elles qu'à l'extérieur.

Sous-communauté : Une sous-communauté est une structure qui se retrouve dans une structure de communauté. Une communauté peut regrouper des sous-structures de nœuds qui sont assemblés.

Indicateur de communautés : Un indicateur de communauté est une mesure de la force d'une structure dans un réseau dans une approche précise.

Nœuds adjacents : Sous-ensemble de nœuds qui sont considérés connectés entres-elles par au moins un arc.

1.2. Applications de la détection de communautés

La détection de communautés conduit à une panoplie d'applications. Un exemple contemporain d'applications de la détection de communautés est dans le contexte de l'épidémie de Covid-19. Les chercheurs qui tentent de créer un vaccin utilisent la détection de communautés pour comprendre les interactions des protéines du virus. Les gouvernements peuvent appliquer la détection de communautés pour comprendre la

propagation du virus et détecter les groupes les plus susceptibles par leurs interactions afin de les protéger. En bref, l'application des algorithmes de détection de communautés est nombreuse dans tous les domaines.

La détection de communautés une boite à outils qui existe depuis quelques décennies et qui a pris beaucoup d'ampleur dans la communauté scientifique à cause des innovations technologiques en informatique. La détection de communautés requiert des ordinateurs très puissants avec un stockage très grand afin de les collecter et les résoudre. Certaines limites contraignent la grosseur d'application de la détection de communauté telles que le nombre d'opérations par seconde des ordinateurs, la capacité de collecte des réseaux ainsi qu'un algorithme efficace afin de retrouver les meilleures partitions.

Ce mémoire va traiter la problématique de détecter les meilleures partitions ou des partitions efficaces dans les réseaux à l'aide d'un algorithme en étape. L'algorithme cherche à résoudre des problèmes intermédiaires des partitions afin d'améliorer la modularité globale. La modularité étant un indicateur de partitions basé sur une comparaison entre la probabilité d'un lien et l'observation d'un lien, une élaboration plus détaillée de la modularité est expliquée dans une section subséquente.

1.3. Représentations des réseaux

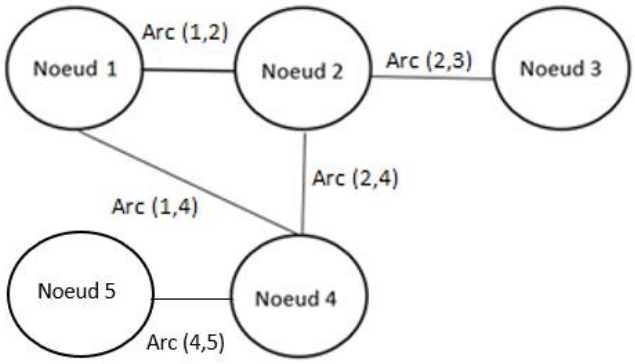
Les réseaux et les graphes peuvent être représentés de manière différente en fonction de l'approche, l'application et la performance voulues. La première est sous forme d'un réseau de nœuds et d'arcs tel qu'illustré dans les figure 1 et figure 2. Le défaut de cette approche intuitive en image est qu'elle est difficile à traiter lorsqu'on utilise un ordinateur ou des réseaux de grande taille. Alors, deux autres formats pour les réseaux sont utilisés pour effectuer des opérations et les algorithmes qui sont la matrice d'adjacence et la liste d'adjacence.

La matrice d'adjacence est une formulation matricielle d'un réseau qui met en relation chaque nœud avec tous les autres nœuds. Les arcs sont représentés par les valeurs à l'intersection des indices des colonnes et des lignes. La matrice d'adjacence représente l'ensemble de l'information dans un tableau.

Figure 3 : Comparaison entre la forme matricielle et la forme graphique

$$\boldsymbol{A : \text{Matrice d'adjacence}}$$

$$\boldsymbol{A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}}$$



La figure 3 illustre l'équivalence des deux formes où chaque nœud est représenté par les indices des colonnes et des lignes et la présence d'un arc est indiquée par un 1. De plus, la diagonale est habituellement zéro à part lorsqu'il y a un arc qui mène à lui-même. Dans un graphe non directionnel comme dans l'exemple, la matrice d'adjacence est symétrique.

a forme matricielle d'un réseau permet l'utilisation d'opérations matricielles. Ces techniques permettent la résolution de certains problèmes. Le défaut de la forme matricielle est qu'elle est souvent creuse nécessite beaucoup de mémoire pour son utilisation à cause de sa nature quadratique. Alors, une formulation plus compacte appelée la liste d'adjacence est utilisée pour compresser l'information tout en conservant la capacité d'effectuer des opérations.

La liste d'adjacence est une formulation compacte de la matrice d'adjacence. Cette liste contient toute l'information strictement nécessaire de manière explicite et le reste de l'information est implicite à sa structure. Sans doute, les matrices d'adjacences sont souvent très peu denses avec beaucoup d'espace. Alors, la liste d'adjacence est une liste

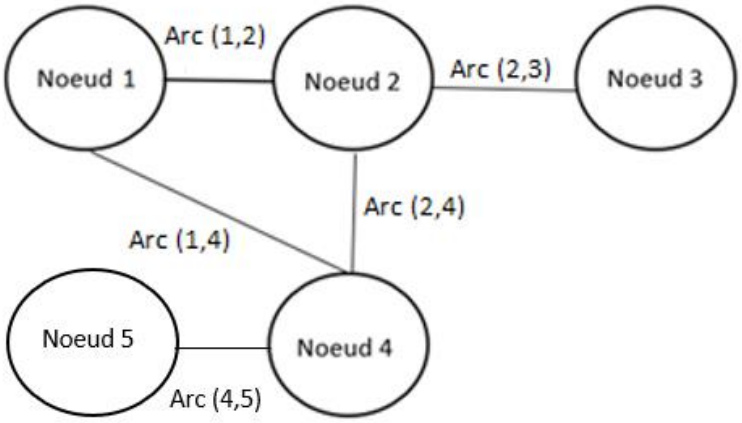
avec deux nœuds qui représente seulement les arcs. Toutes les informations non explicites sont considérées nulles.

Figure 4 : Équivalence avec la liste d'adjacence

Liste $\rightarrow \{(\mathbf{1},\mathbf{2}),(\mathbf{1},\mathbf{4}),(\mathbf{2},\mathbf{4}),(\mathbf{2},\mathbf{3}),(\{\mathbf{4},\mathbf{5}\})\}$

Arc	Noeud i	Noeud j
(1,2)	1	2
(1,4)	1	4
(3,4)	3	4
(2,3)	2	3
(4,5)	4	5

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$



Alors, la liste d'adjacence compresse le réseau ainsi que la matrice d'adjacence en trois lignes qui représente quel nœud est connecté à quel nœud. La figure 4 illustre la formulation en liste d'adjacence. Dans le cas d'un graphe peu dense, cette manière compacte est beaucoup moins taxant sur la mémoire des ordinateurs et les opérations peuvent être effectuées. De plus, les algorithmes sont généralement plus rapides puisque la liste d'adjacence limite le domaine de recherche de manière implicite.

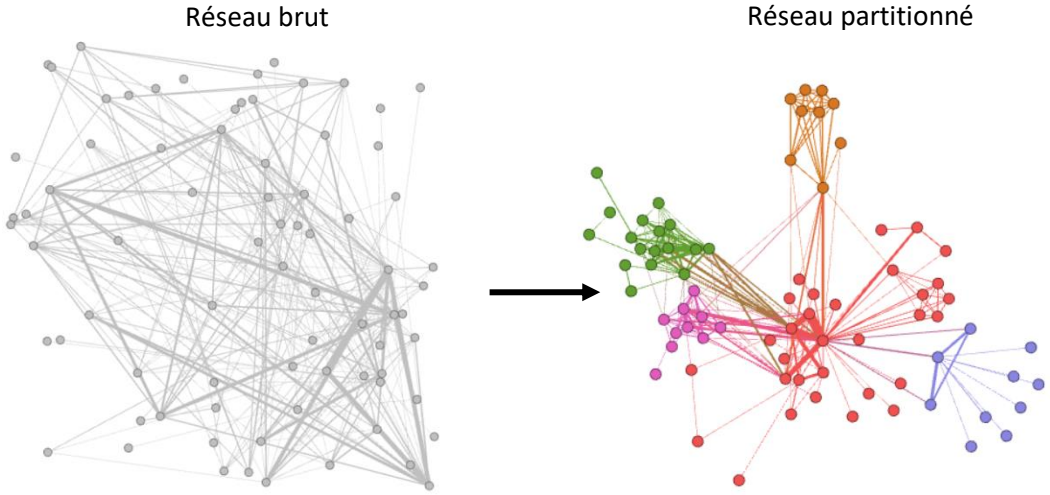
Chapitre 2 :
Revue de la littérature

2.1. Classe de problème algorithmique

La détection de communautés dans un réseau complexe non directionnel est un problème algorithmique en soi qui possède plusieurs axes d'approches afin d'arriver à des résultats efficaces dans un temps raisonnable. Plusieurs méthodes avec des philosophies divergentes ont été proposées avec différents degrés d'efficacité et de précision.

La propriété des réseaux complexes est qu'elle est divisible en sous-structures qui forme des ensembles de nœuds qui ont la propriété d'avoir une densité d'arcs plus élevée entres-elles qu'à l'extérieur. L'objectif de la détection de communautés est de détecter la meilleure partition du réseau complexe afin d'extraire et d'analyser des sous-structures qui ont la propriété de densité élevée. Le résultat de ce processus mène à une partition de nœuds dans divers regroupements comme l'illustration de la figure 5. L'image à gauche dans la figure 5 illustre un placement de nœuds aléatoire sans structure. L'image de droite est une partition basée sur la modularité où les sous-structures sont identifiées par leurs couleurs.

Figure 5 : Comparaison du réseau *Les Misérables* brute et partitionné en fonction de la modularité.



Deux questions émergent de l'objectif de la détection de communautés :

1. Métrique : Qu'elle est la définition d'une bonne partition ?
2. Algorithme : Comment détecter la meilleure partition ?

Nonobstant la réponse aux questions, la littérature illustre certains incontournables face à ce problème et la complexité en termes de temps de calcul afin de trouver la meilleure partition.

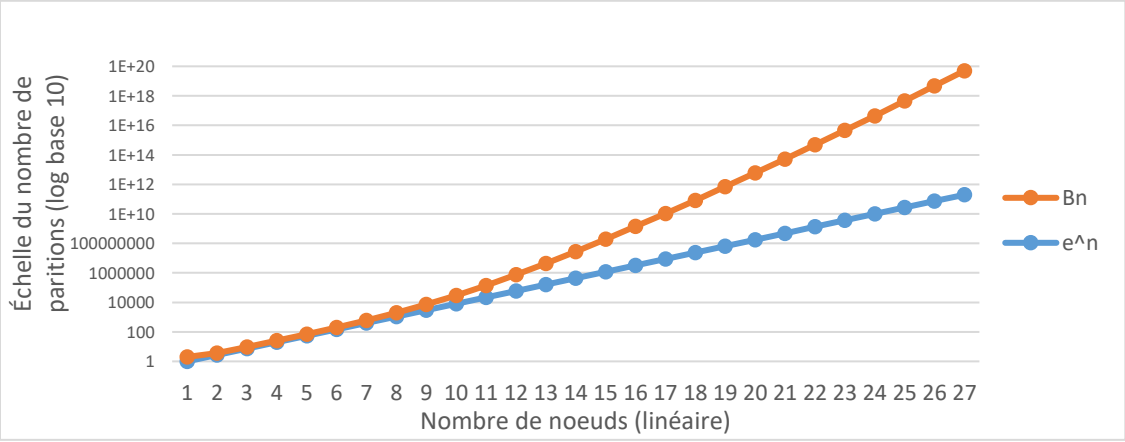
2.2. Taille du problème : Les chiffres de Bell

Lorsqu'on regarde les petits réseaux, certains regroupements peuvent émerger intuitivement. Cependant, le problème de partitionner le réseau est de la classe de problèmes combinatoire. Le nombre de partitions possible augmente très rapidement. Le nombre de partitions possible augmente à un rythme des nombres de Bell décrit par :

$$B_{n+1} = \sum_{k=0}^{n} \left\{ {n \atop k} \right\} B_k \qquad (1)$$

La fonction de Bell est le nombre de partitions non vide possible dans un ensemble de nœuds (Mezo 2010). La croissance du nombre de partitions suit une tendance supérieure à l'exponentielle comme la figure 6 le démontre.

Figure 6 : Comparaison de la croissance du nombre de partitions avec d'autres fonctions



2.3. Indicateurs de communautés dans les réseaux complexes

Plusieurs critères et métriques ont été proposés par divers auteurs afin de mesurer le degré ou la qualité de la partition d'un réseau. Aucun consensus n'a été établi sur la suprématie d'une métrique sur l'ensemble proposé par divers auteurs. Cependant, voici la définition de Clauset (2013) pour résumer l'ensemble des approches :

$$G : \text{Réseau (input)}$$
$$F_i(G): \text{Fonction i (Algorithm)}$$
$$P_i = F_i(\mathrm{G}) : \text{Partition i obtenue de l'algorithme i}$$
$$Z(P_i) : \text{Mesure de la qualité de la partition i}$$
$$si\ max\ F_1(G) = Z(P_1) > Z(P_2) = max\ F_2(G),$$
$$alors\ algorithme\ F_1\ produit\ de\ meilleurs\ partitions$$

La définition tirée de Clauset (2013) illustre qu'un réseau G est partitionné par un algorithme $F_i(G)$ et une métrique évalue la partition générée par l'algorithme. La comparaison du résultat des métriques est suffisante pour arbitrer la meilleure partition du réseau. Les critères de métriques utilisés ne sont pas des mesures objectives telles que la quantité de joules ou la longueur d'un gymnase qui sont inchangées, peu importe l'application ou l'emplacement de l'observateur. Les diverses mesures doivent être suivies d'une analyse de l'application, voir même une optimisation selon plusieurs critères pour avoir plus d'une perspective par l'analyste.

2.3.1. Partitions basées sur les coupes

Intuitivement, le premier réflexe afin de partitionner les réseaux complexes est de former des groupes en minimisant le nombre d'arcs entre les communautés (Wang, Huang, Feng, Liu 2015). Plusieurs variations existent telles que le ratio, le nombre de coupes nettes, la moyenne, etc. Cependant, tous les indicateurs utilisent des scores pour minimiser les arcs intra communautés. De plus, les nœuds peuvent aussi se faire mettre sur un plan cartésien avec une matrice de semblance afin de déterminer leur emplacement. Les fonctions qui minimisent ces coupes n'ont aucun regard sur la structure interne des communautés.

2.3.2. Maximisation de la modularité

La modularité est un indicateur de la qualité d'une partition proposé par Newman (2006). Cette approche a reçu une grande popularité vu la nature qualitative de la mesure. La modularité sera la mesure utilisée dans les techniques proposée dans ce mémoire. La maximisation de la modularité est un partitionnement d'un réseau basé sur la modularité, c'est-à-dire que l'algorithme recherche une partition avec la plus grande concentration d'arcs dans les communautés en comparaison à une distribution aléatoire. Cette densité d'arcs supérieurs à l'aléatoire dans les communautés maximise la métrique de modularité.

La modularité est une mesure qui compare la distribution des arcs dans le graphe par rapport à la probabilité d'avoir un arc entre deux nœuds. En autres mots, la modularité compare la proximité des nœuds par rapport à ce qui est attendu par chance (Newman 2006). Cette mesure incorpore une comparaison entre les partitions et l'hypothèse nulle afin de déterminer si les partitions sont significatives. Voici la formulation de Newman pour le calcul de la modularité :

Q : Modularité

C_i : Communauté à laquelle le noeud i appartient

$\delta(C_i, C_j)$: Delta de khronecker (si $\mathrm{C_i} = \mathrm{C_j}, \delta(\mathrm{C_i}, \mathrm{C_j}) = 1$, sinon $\delta(\mathrm{C_i}, \mathrm{C_j}) = 0$)

$$\mathrm{Q} = \frac{1}{2\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{i}=\mathrm{n}} \sum_{\mathrm{j}=1}^{\mathrm{j}=\mathrm{n}} \left(\mathrm{A_{ij}} - \frac{\mathrm{k_i} * \mathrm{k_j}}{2\mathrm{m}} \right) \delta(\mathrm{C_i}, \mathrm{C_j}) \qquad (2)$$

La modularité peut être utilisée comme mesure locale ou comme mesure globale. La mesure locale est employée pour évaluer la force d'une communauté tandis que la mesure globale mesure la force de la partition complète. La modularité utilise une comparaison des arcs observés avec une distribution aléatoire théorique. Si *Q* tend vers 0, alors le réseau se rapproche d'une distribution aléatoire sans aucune structure. Si *Q* tend vers 1, ceci démontre des structures fortes ou une déviation de l'aléatoire qui signifie des structures. De plus, la modularité a la propriété d'être additive. Donc, la somme de *Q* par communauté est égale à la somme globale (Fortunato 2010).

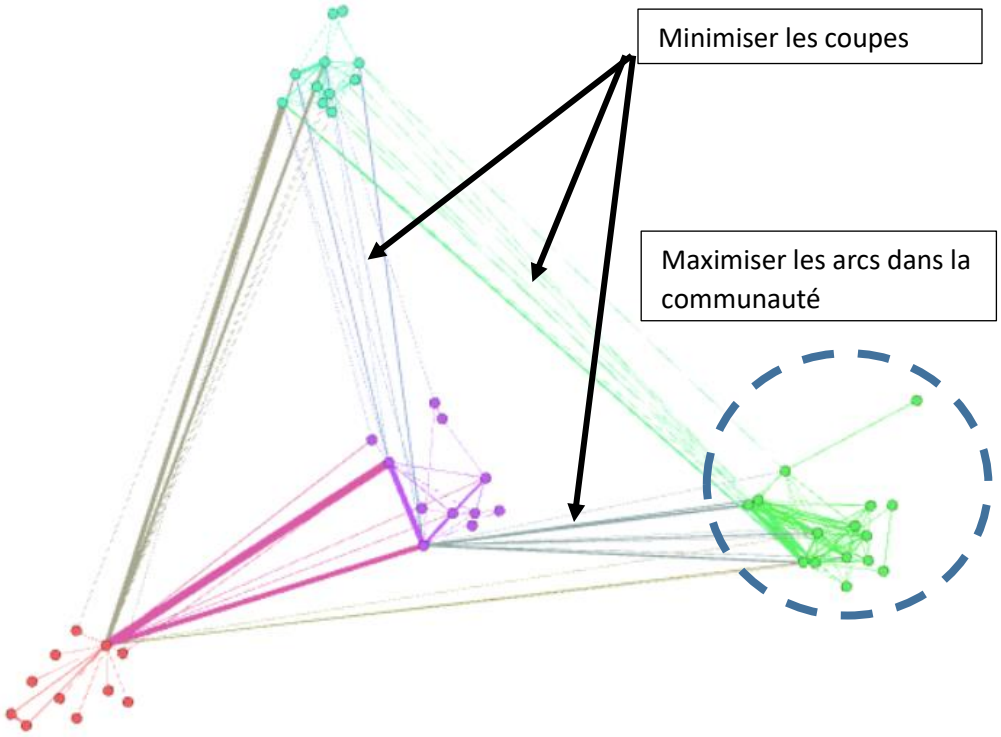
$$Q_c : Modularité\ de\ la\ communauté\ c$$

$$t : Nombre\ total\ de\ communautés$$

$$Q = \sum_{c=1}^{t} Q_c \tag{3}$$

La modularité est souvent utilisée pour évaluer la qualité d'un réseau complexe. Une plus grande modularité dans une partition signifie une meilleure partition. Le Q optimal (Q_{max}) peut être également comparé entre les réseaux afin de déterminer la qualité des structures.

Figure 7 : Représentation graphique des approches à la détection de communauté



La figure 7 est une représentation d'un réseau afin de voir visuellement la différence entre les approches de minimiser les coupes et maximiser les arcs dans une communauté. Évidemment, les deux approches ont un rapport entre elles parce qu'un arc ne peut pas être détruit. Alors, un arc est coupé par deux communautés ou un arc connecte deux nœuds dans une même communauté. Cependant, les deux approches vont avoir une préférence dans la fonction objective pour la minimisation ou la maximisation.

Malgré l'efficacité de la modularité comme indice de qualité de la partition d'un réseau, les modules interconnectés de la partition possèdent une limite de taille dépendamment de la taille du réseau entier (Fortunato 2006). De plus, la modularité est influencée par la taille totale du réseau. En autres mots, plus le réseau est grand (nombre de nœuds), il y a une décroissance de la probabilité ou attente d'un arc entre deux nœuds d'une même partition vu la division par l'ensemble des nœuds. Alors, le résultat de ce biais est que les partitions plus petites finissent par être fusionnées lorsqu'elles devraient être séparées. Alors, les modules de la partition finissent par avoir un minimum de taille dans les réseaux très grands.

Malgré les défauts de l'approche de la modularité, ce critère a atteint une popularité à cause de son caractère unimodal, son application intuitive et son universalité. Cette approche permet de ressortir une partition qui fonctionne et qui est comparable entres-elles à l'aide de la modularité.

2.4. Classes d'algorithmes de maximisation de la modularité

Une grande majorité de la littérature est concentrée sur la deuxième question de trouver un algorithme efficace pour trouver les bonnes partitions. Plusieurs critères sont utilisés par évaluer la performance des algorithmes de détection de communauté. Pour ce qui concerne ce mémoire, nous allons nous concentrer sur l'ensemble de la littérature qui utilise la modularité comme indice de qualité d'une partition et tente de la résoudre en maximisant la modularité.

Plusieurs classes et techniques avec une panoplie de variations ont été employés afin d'établir des algorithmes efficaces. Cependant, le problème de la maximisation de la modularité reste encore un problème NP-difficile.

2.5. Liste de techniques les plus communs

D'une perspective de complexité algorithmique, le dilemme entre la complexité (O) de l'algorithme et la précision de l'algorithme est universel dans tous les algorithmes. L'algorithme utopique le plus efficace est capable d'avoir une résolution exacte dans un

temps *O(1)*. Évidemment, un algorithme exact en *O(1)* est à l'extérieur de la sphère de connaissance actuelle. Ce troc entre précision et complexité est un dilemme essentiel lorsqu'on évalue la performance des algorithmes et souvent nous devons nous contenter du meilleur pour le but et l'application en question. Le graphique 1.3 est un schéma des classes d'algorithmes qui seront brièvement décrites et leur localisation à l'intérieur de ce compromis.

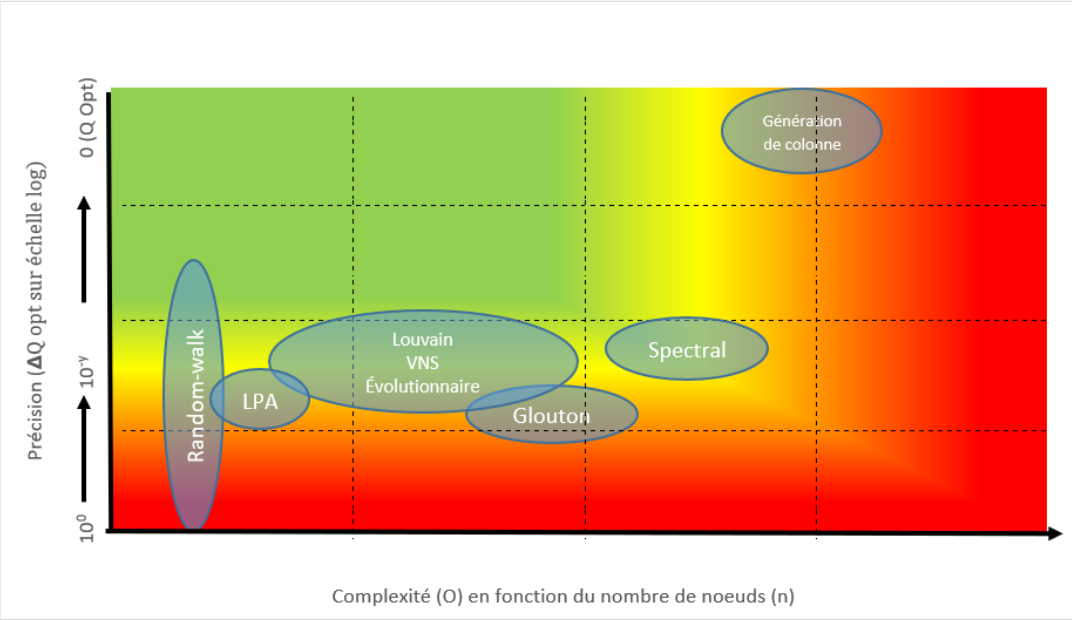
Définition de la précision :

La précision dans un contexte de modularité est basée sur la valeur de ΔQ_{max} (Good, 2010) qui représente la différence entre la valeur de Q retrouvée et la valeur de Q maximale. Les valeurs de Q représentent différentes partitions du réseau. Cependant, les réseaux de petite taille ont la preuve d'optimalité et facilite la comparaison de modularité. Pour les réseaux de très grande taille, cette comparaison est effectuée avec un Q_{max} théorique qui est représenté par la meilleure partition retrouvée à l'aide d'autres algorithmes.

Définition Complexité :

La complexité définit le nombre d'opérations nécessaires afin d'avoir une solution adéquate pour un réseau d'une taille fixe dans une recherche exhaustive ou le pire scénario. Par extension, si la complexité décrit une pente en nombre d'opérations en fonction de la taille du problème. D'une perspective plus pragmatique, la complexité est la quantité de temps nécessaire pour accomplir la tâche.

Figure 8 : Graphique conceptuel de la précision des algorithmes les plus populaires en fonction de la complexité



La figure 8 est une illustration conceptuelle des algorithmes les plus populaires en fonction de leur complexité et précision interprétée à partir des tableaux de Coscia (2011). Le graphique est une simplification générique des algorithmes les plus populaires. En réalité, les performances des algorithmes sont aussi variées qu'il y a d'auteurs. Cependant, l'évaluation de la complexité et de la précision des algorithmes est essentielle à déterminer leurs application et efficacité en fonction des ressources et des besoins. De plus, le graphique est divisé en trois régions (vert, jaune et rouge). Un algorithme qui serait dans la région verte rendrait le problème de la détection de communautés dans la catégorie de temps polynomial. La région rouge est l'ensemble des algorithmes qui ne sont ni précis ni rapides. Alors, tous les problèmes se retrouvent dans la région jaune des compromis entre précision et complexité.

2.5.1. Glouton (Newman, 2004)

L'algorithme glouton rapide de Newman est un algorithme glouton qui commence avec chaque nœud dans un groupe disjoint. La modularité d'un changement de groupe de tous les nœuds est calculée de manière itérative et la meilleure amélioration est choisie (Max ΔQ). Alors, la meilleure amélioration signifie un changement de groupe pour un nœud.

Cet algorithme va cycler jusqu'à aucun changement de groupe pour un nœud n'améliore *Q*.

La problématique de cet algorithme est qu'il ne calcule que l'amélioration d'un nœud à la fois. Alors, lorsqu'il y a une paire de nœuds qui peut améliorer la fonction, l'algorithme ne peut pas le détecter.

2.5.2. Propagation d'étiquettes (Raghavan et al., 2007)

La propagation d'étiquettes aussi appelée « Label Propagating Algorithm » (LPA) et ses variantes sont une technique qui commence avec un état ou chaque nœud appartient à un groupe unique. L'algorithme va changer les libellés (Changement de groupe) copiant le libellé le plus commun parmi ses voisins de manière aléatoire. Lorsque deux libellés ont la même fréquence, un libellé est choisi au hasard entre les plus hautes fréquences.

Cet algorithme est rapide par rapport aux autres, cependant c'est une heuristique comme le Glouton qui a des solutions très variables ainsi que des partitions d'une qualité inférieure généralement.

2.5.3. Louvain (Blondel et al., 2008)

La technique de Blondel[3] est un des algorithmes les plus rapides pour trouver des partitions à cause de la nature réductionniste de l'information. Cette technique peut être utilisée dans des très grands réseaux.

L'algorithme est initialisé en associant un groupe par nœud. Par la suite, l'algorithme effectue deux étapes à répétition jusqu'à ce que nulle amélioration ne soit possible :

1. Glouton (Newman, 2004) : L'algorithme effectue un algorithme de glouton jusqu'à ce qu'aucune amélioration ne soit possible.
2. Agglomération : Les groupes de noeuds sont fusionnés et calculés comme un seul noeud. Alors, chaque communauté est représentée comme un noeud et l'information de chacun des noeuds qui compose ce super-noeud est éliminée. Par la suite, l'algorithme recommence avec un glouton de ces super-noeuds.

2.5.4. VNS (Aloise, al., 2013)

« Variable neighbourhood search » (VNS) est une métaheuristique employée qui emploie une recherche dans le voisinage en modifiant une variable à la fois afin de détecter une amélioration. Cet algorithme est effectué en deux étapes :

1. Heuristique d'amélioration : L'heuristique d'amélioration employée dans l'article de (Aloise et al., 2013) est un LPAm+. Cependant tout autre algorithme peut être employé également.
2. Exploration du voisinage : L'étape d'exploration consiste d'un ensemble prédéterminé de perturbation ou de modifications suivi d'une évaluation pour déterminer la meilleure amélioration.

2.5.5. Spectral

Les techniques spectrales sont une classe d'algorithme qui se base sur les propriétés spectrales des graphes. Généralement, ces algorithmes vont détecter des communautés à travers les eigenvectors (Fortunato, 2010). Elles vont déterminer les nœuds qui contribuent le plus au eigenvector de la communauté et les classer en fonction de ces indicateurs.

Les techniques spectrales sont populaires pour leur nature flexible et robuste. Les approches spectrales ont tendance à avoir des partitions globales dans la direction optimale, mais l'efficacité est réduite par les erreurs au niveau des nœuds individuels (Coscia, 2010). Alors, les techniques spectrales plus récentes tentent de les jumeler avec un algorithme de perturbation afin d'améliorer la solution.

2.5.6. Méthodes exactes (Aloise, et al.,2010)

Les méthodes exactes sont une classe de techniques qui recherche la solution optimale ainsi que la preuve d'optimalité pour la maximisation de la modularité. Ces techniques se concentrent sur la qualité de la partition, cependant elles ont un temps de calcul supérieur.

Par exemple, la méthode de génération de colonne (Aloise et ass., 2010) emploie une réduction de la maximisation de la modularité par un « clique partitionning » pour initialiser une solution suivie d'une programmation en quadratique en nombre entier avec

une génération de colonnes qui prend en compte l'ensemble des configurations possible. L'initialisation est utilisée pour minimiser le temps de recherche de la deuxième étape plus ardue.

Ces techniques sont très lentes afin de trouver une solution optimale, cependant elles ont la capacité d'avoir la preuve d'optimalité ce que les heuristiques ne possèdent pas. Pour les réseaux de grandes tailles, ces algorithmes sont beaucoup trop lents pour être efficaces.

2.5.7. Marche aléatoire

Les techniques de marche aléatoires sont autant utilisées seules qu'en combinaison avec d'autres algorithmes afin d'injecter de l'aléatoire ou sortir des sommets locaux. Les techniques de marche aléatoire prennent des points aléatoires dans l'espace de solution dans un ordre ou aléatoire. Par la suite ces techniques déterminent une nouvelle direction à partir des points retrouvés dans les cycles antérieurs.

La force de ces techniques est qu'elles sont simples et faciles à implémenter pour identifier une solution à travers un très grand nombre d'itérations en retenant la meilleure réponse. Ces techniques ont de meilleurs résultats en combinaison avec un algorithme plus déterministe afin de restreindre l'ensemble de l'espace de solution.

2.5.8. Algorithmes évolutionnaires

Les algorithmes évolutionnaires sont une métaheuristique très populaire pour leur robustesse et leurs polyvalences avec plusieurs noms et variations. La classe de techniques évolutionnaire est une marche aléatoire plus sophistiquée qui est aussi facile à implémenter.

Les techniques évolutionnaires vont prendre plusieurs solutions réalisables et leurs imposer des variations afin de déterminer les meilleures mutations d'une solution. Par la suite, cette technique va croiser les solutions retenues afin de croiser les meilleures solutions pour se rapprocher d'une configuration plus propice en termes de qualité et de mutations. Suivi de plusieurs itérations, une sélection naturelle des solutions effectuera un triage et l'algorithme retient la meilleure solution.

Chapitre 3 :
Définition de la problématique

3.1. Paramètres du problème

Les réseaux analysés pour l'algorithme en étapes ont été restreints à certaines caractéristiques afin de concentrer les efforts et fournir une stabilité lors de l'exécution de l'algorithme. Tous les réseaux ont les caractéristiques suivantes :

1. Réseaux non directionnels
2. Réseaux sans poids
3. Groupes disjoints
4. Maximisation de la modularité

3.2. Transformation en matrice de modularité et équivalences

Les réseaux peuvent prendre plusieurs formes ou structures. Souvent, elles sont stockées en tuples vu que la matrice d'adjacente est une matrice creuse. Cependant, les réseaux peuvent être transformés en matrice d'adjacence qui met en relation les arcs entre les nœuds tel que décrit dans la section 1.3. Afin de comprendre la modularité, trois matrices doivent être créées :

$$\frac{A_{ij}}{2m} \text{ : élément de la matrice d'adjacence pondérée sur } 2m \quad (4)$$

La matrice pondérée sur *2m* est équivalente aux données observées pour calculer leurs poids.

$$\frac{k_i * k_j}{4m^2} \text{ : élément de la matrice d'espérance théorique} \quad (5)$$

La matrice d'espérance est l'espérance théorique ou l'attente dans le cas d'une hypothèse nulle ou aléatoire.

$$\delta(C_i, C_j) : \text{ élément de la matrice de kroenecker (binaire)} \quad (6)$$

La matrice de Kronecker indique pour chaque paire de nœuds si ceux-ci appartiennent ou non à la même classe. En effet, le delta vaut 1 si les deux nœuds appartiennent à la même classe et 0 dans le cas contraire.

$$B_{ij} = \frac{A_{ij}}{2m} - \frac{k_i * k_j}{4m^2} \text{ élément de la matrice de modularité} \quad (7)$$

La matrice de modularité est la déviation à une distribution aléatoire calculée à chaque arc par la différence entre les valeurs observées et théoriques.

$$\mathrm{Q} = \sum_{\mathrm{i=1}}^{\mathrm{i=n}} \sum_{\mathrm{j=1}}^{\mathrm{j=n}} \left(\frac{A_{ij}}{2m} - \frac{k_i * k_j}{4m^2} \right) \delta(\mathrm{C_i}, \mathrm{C_j}) \quad (8)$$

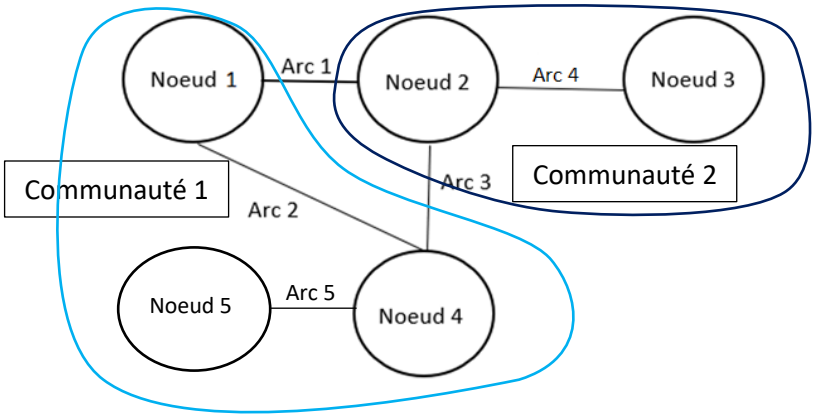
La combinaison des trois matrices dans une fonction de modularité qui calcule la somme complète de la modularité.

$$\mathrm{Q} = \frac{1}{2\mathrm{m}} \sum_{\mathrm{i=1}}^{\mathrm{i=n}} \sum_{\mathrm{j=1}}^{\mathrm{j=n}} \left(\mathrm{A_{ij}} - \frac{\mathrm{k_i} * \mathrm{k_j}}{2\mathrm{m}} \right) \delta(\mathrm{C_i}, \mathrm{C_j}) \quad (9)$$

Simplification de la formule (**8**) qui mène à la formule classique de la modularité de Newman.

La matrice de modularité B = (B_{ij}) est construite à partir de la matrice d'adjacence A en appliquant la fonction de modularité à chaque élément de la matrice. Un exemple des calculs des matrices est illustré à la figure 9.

Figure 9 : Exemple du calcul de la modularité et la matrice de modularité



La matrice d'adjacence :

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \end{bmatrix}$$

Vecteur des degrés

$$\boldsymbol{k} = [\mathbf{2} \quad \mathbf{3} \quad \mathbf{1} \quad \mathbf{3} \quad \mathbf{1}]$$

Nombre d'arcs : $\boldsymbol{m = 5}$

La matrice d'adjacence pondérée sur *2m* :

$$\begin{bmatrix} \mathbf{0} & \mathbf{0.1} & \mathbf{0} & \mathbf{0.1} & \mathbf{0} \\ \mathbf{0.1} & \mathbf{0} & \mathbf{0.1} & \mathbf{0.1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0.1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0.1} & \mathbf{0.1} & \mathbf{0} & \mathbf{0} & \mathbf{0.1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0.1} & \mathbf{0} \end{bmatrix}$$

La matrice d'espérance théorique :

$$\begin{bmatrix} \mathbf{0.04} & \mathbf{0.06} & \mathbf{0.02} & \mathbf{0.06} & \mathbf{0.02} \\ \mathbf{0.06} & \mathbf{0.09} & \mathbf{0.03} & \mathbf{0.09} & \mathbf{0.03} \\ \mathbf{0.02} & \mathbf{0.03} & \mathbf{0.01} & \mathbf{0.03} & \mathbf{0.01} \\ \mathbf{0.06} & \mathbf{0.09} & \mathbf{0.03} & \mathbf{0.09} & \mathbf{0.03} \\ \mathbf{0.02} & \mathbf{0.03} & \mathbf{0.01} & \mathbf{0.03} & \mathbf{0.01} \end{bmatrix}$$

La matrice de modularité :

$$\boldsymbol{B} = \begin{bmatrix} \mathbf{-0.04} & \mathbf{0.04} & \mathbf{-0.02} & \mathbf{0.04} & \mathbf{-0.02} \\ \mathbf{0.04} & \mathbf{-0.09} & \mathbf{0.07} & \mathbf{0.01} & \mathbf{-0.03} \\ \mathbf{-0.02} & \mathbf{0.07} & \mathbf{-0.01} & \mathbf{-0.03} & \mathbf{-0.01} \\ \mathbf{0.04} & \mathbf{0.01} & \mathbf{-0.03} & \mathbf{-0.09} & \mathbf{0.07} \\ \mathbf{-0.02} & \mathbf{-0.03} & \mathbf{-0.01} & \mathbf{0.07} & \mathbf{-0.01} \end{bmatrix}$$

Figure 9 : Exemple du calcul de la modularité et la matrice de modularité (suite)

Le vecteur de communautés : $\boldsymbol{C} = [\mathbf{1} \quad \mathbf{2} \quad \mathbf{2} \quad \mathbf{1} \quad \mathbf{1}]$

La matrice de Kronecker :

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

Le calcul de la modularité :

$$\mathbf{Q} = \sum_{\mathbf{i=1}}^{\mathbf{i=n}} \sum_{\mathbf{j=1}}^{\mathbf{j=n}} \left(\frac{\boldsymbol{A_{ij}}}{\boldsymbol{2m}} - \frac{\boldsymbol{k_i * k_j}}{\boldsymbol{4m^2}} \right) \boldsymbol{\delta}(\mathbf{C_i}, \mathbf{C_j})$$

$$\boldsymbol{Q} = \sum_{\mathbf{i=1}}^{\mathbf{i=n}} \sum_{\mathbf{j=1}}^{\mathbf{j=n}} \boldsymbol{B_{ij}} \ \boldsymbol{\delta}(\mathbf{C_i}, \mathbf{C_j})$$

En utilisant le produit de Hadamard, on obtient

$$\begin{bmatrix} \mathbf{-0.04} & \mathbf{0.04} & \mathbf{-0.02} & \mathbf{0.04} & \mathbf{-0.02} \\ \mathbf{0.04} & \mathbf{-0.09} & \mathbf{0.07} & \mathbf{0.01} & \mathbf{-0.03} \\ \mathbf{-0.02} & \mathbf{0.07} & \mathbf{-0.01} & \mathbf{-0.03} & \mathbf{-0.01} \\ \mathbf{0.04} & \mathbf{0.01} & \mathbf{-0.03} & \mathbf{-0.09} & \mathbf{0.07} \\ \mathbf{-0.02} & \mathbf{-0.03} & \mathbf{-0.01} & \mathbf{0.07} & \mathbf{-0.01} \end{bmatrix} \circ \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

$$\mathbf{Q} = \sum_{\mathbf{i=1}}^{\mathbf{i=n}} \sum_{\mathbf{j=1}}^{\mathbf{j=n}} \begin{bmatrix} \mathbf{-0.04} & \mathbf{0} & \mathbf{0} & \mathbf{0.04} & \mathbf{-0.02} \\ \mathbf{0} & \mathbf{-0.09} & \mathbf{0.07} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0.07} & \mathbf{-0.01} & \mathbf{0} & \mathbf{0} \\ \mathbf{0.04} & \mathbf{0} & \mathbf{0} & \mathbf{-0.09} & \mathbf{0.07} \\ \mathbf{-0.02} & \mathbf{0} & \mathbf{0} & \mathbf{0.07} & \mathbf{-0.01} \end{bmatrix}$$

En réalité, la matrice de modularité (B) est indépendante de la partition. La matrice de Kronecker est ce qui change en fonction des partitions. Alors, la modularité est dépendante de la matrice de Kronecker.

$$\boldsymbol{Q} = \mathbf{0.08}$$

Dans cet exemple simple, la modularité est positive parce qu'il y a deux communautés selon la modularité. Alors, une division en deux groupes :

1. Nœuds 1,4 et 5;
2. Nœuds 2 et 3

3.3. Propriétés de la matrice de modularité

1. Création d'arcs entre tous les nœuds

    a. La matrice B de modularité est une matrice pleine créée à partir d'une matrice creuse. La formule de Newman crée artificiellement des arcs entre tous les nœuds du réseau et chaque élément de la matrice représente le poids associé à chacun des arcs. Le poids de la matrice B est positif pour chaque arc de la matrice d'adjacence alors que le poids est négatif en l'absence d'un arc. Ces valeurs sont expliquées par le delta entre la valeur observée et la valeur espérée. Alors, le graphe original est représenté par un graphe complet pondéré par la valeur de la modularité. Donc, nous allons créer une terminologie pour simplifier la description et l'utilisation de ces nœuds dans le nouveau graph complet de modularité.

        i. Arcs adjacents : Les arcs du réseau original, c'est-à-dire, les arcs qui ont une modularité positive.

        ii. Arcs non adjacents : Les arcs absents du réseau original ayant une modularité négative.

        iii. Arcs index : Tous les arcs d'un nœud vers lui-même. La formule de Newman pondère une espérance d'un arc entre le nœud et lui-même. Puisque ces arcs sont absents du réseau original, leur modularité est nécessairement négative.

    b. Explication :

$$\forall\, i,j\ \in [1,n]\ \mathrm{B_{ij}} = \frac{\mathrm{A_{ij}}}{2\mathrm{m}} - \frac{\mathrm{k_i} * \mathrm{k_j}}{4\mathrm{m}^2} \qquad \textbf{(10)}$$

Formulation de la matrice de modularité

$$\forall\, i,j\ \in [1,n]\ \frac{k_i * k_j}{2m} > 0 \qquad \textbf{(11)}$$

Tout nœud a une espérance d'avoir un lien avec tous les autres nœuds et ne peut pas être nul. En reprenant la formule (**10**), les valeurs de la matrice de modularité

auront toujours une valeur non nulle à cause de la différence entre la valeur observée et la valeur théorique non nulle.

2. Somme de la matrice complète est nulle

    a. La matrice de la modularité B_{ij} est une matrice où chaque élément de la matrice est la différence entre la valeur de l'arc (0 ou 1) et la valeur espérée avec l'hypothèse nulle. Alors, la somme de la matrice au complet est égale à 0 parce qu'il ne peut pas avoir une espérance plus élevée ou plus faible que l'ensemble des observations des nœuds.

    b. Explication :

$$Q = \frac{1}{2m} \sum_{i,j=1}^{i,j=n} \left(A_{ij} - \frac{k_i * k_j}{2m} \right) \delta(C_i, C_i) \quad \textbf{(12)}$$

$$\forall\, i,j \in [1,n]\ \delta(C_i, C_i) = 1, \text{ si M=1 (1 groupe)} \quad \textbf{(13)}$$

Matrice de Kronecker dans une configuration incluant tous les nœuds dans le même groupe afin d'inclure l'ensemble de la matrice dans le calcul.

$$\sum_{i=1}^{i=n} \sum_{j=1}^{j=n} \left(\frac{A_{ij}}{2m} \right) = 1 \text{ , vu que } m = \frac{\sum_{i=1}^{i=n} \sum_{j=1}^{j=n} (A_{ij})}{2} \quad \textbf{(14)}$$

Division par un scalaire issu de la somme de la matrice est égale à 1.

$$\forall\, i \in [1,n] \sum_{j=1}^{j=n} A_{ij} = k_i \quad \textbf{(15)}$$

Vecteur de la somme de chaque ligne. La simple somme de A_{ij} est le vecteur de ligne ou colonne.

$$\sum_{i=1}^{i=n} k_i = \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} A_{ij} \; ou \sum_{j=1}^{j=n} k_j = \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} A_{ij} \quad \textbf{(16)}$$

Équivalence entre la somme des lignes et des colonnes pour les réseaux non directionnels.

$$Q = \frac{1}{2m}\sum_{i,j=1}^{i,j=n}\left(A_{ij} - \frac{k_i * k_j}{2m}\right)\delta(C_i, C_i) = \sum_{i,j=1}^{i,j=n}\left(\frac{A_{ij}}{2m} - \frac{k_i}{2m} * \frac{k_j}{2m}\right)\delta(C_i, C_i) \quad \textbf{(17)}$$

Formulation de Newman convertie dans une forme développée.

$$Si\ M = 1, \qquad Q = \sum_{i,j=1}^{i,j=n}\left(\frac{A_{ij}}{2m} - \frac{A_{ij}}{2m} * \frac{A_{ij}}{2m}\right)\delta(C_i, C_i) = (1 - 1 * 1) * 1 = 0 \quad \textbf{(18)}$$

Substitution de la somme des lignes et des colonnes par la double somme de la matrice d'adjacence pour le cas d'un groupe. En autres mots, la double somme de A_{ij} est la somme de $k_i\ ou\ k_j$ (**16**).

3. Maximisation de modularité

    a. La somme totale de la matrice de la modularité est égale à 0. Alors, la maximisation de la modularité recherche une solution qui maximise l'extraction d'une diagonale de regroupements d'arcs. Cette maximisation est la somme des liens intragroupes ou la minimisation de la somme extra groupes.

$$Q = Q + Q^{-} = 0 \quad \textbf{(19)}$$

La somme de l'ensemble de tous les arcs est nulle (**18**)

$$Q = \frac{1}{2m}\sum_{i,j=1}^{i,j=n}\left(A_{ij} - \frac{k_i * k_j}{2m}\right)\delta(C_i, C_i) \quad \textbf{(20)}$$

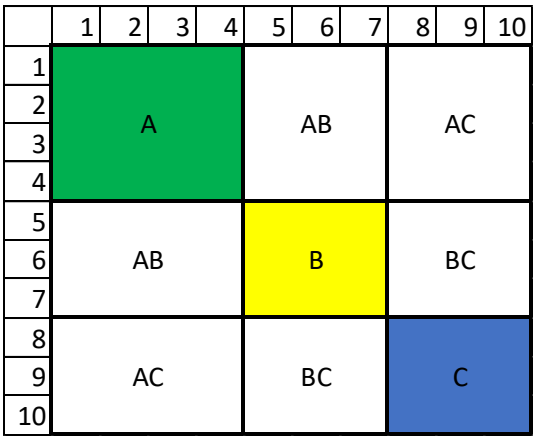
Formule de Newman de la modularité

$$Q^{-} = \frac{1}{2m}\sum_{i,j=1}^{i,j=n}\left(A_{ij} - \frac{k_i * k_j}{2m}\right)\left(1 - \delta\left(C_i, C_j\right)\right), (lorsque\ C_i \neq C_i) \quad \textbf{(21)}$$

Alors, la transformation du delta de Kronecker qui exclut tous les arcs qui ne sont pas dans la même communauté décrit la composante du Q^{-}. Le Q^{-} décrit l'absence de structures entre les communautés en comparaison à une distribution aléatoire.

b. Explication :

Figure 10 : Représentation simplifiée de la somme de la modularité



A, B, C : *Somme intragroupe de modularité*

AB, AC, BC : *Somme intergroupe*

$Q = A + B + C$

$Q^- = 2 * (AB + AC + BC)$

$Q + Q^- = A + B + C + 2 * (AB + AC + BC)$=0

L'illustration géométrique permet la séparation entre la somme des arcs captés par Q et le reste représenté par Q^- qui est l'ensemble des arcs qui ne sont pas dans la même communauté.

La figure 10 est une représentation visuelle géométrique des sommes de modularité comme une aire afin d'illustrer les coupes. Cette simplification est basée sur la transformation des cellules de la matrice en une superficie que l'on peut additionner. Les régions A, B et C sont les régions des arcs qui sont dans les mêmes communautés. Par la suite, les régions AB, AC et BC sont l'ensemble des arcs entre les groupes. De plus, la somme totale l'ensemble des régions est nulle. De plus, la maximisation de la somme des régions A, B et C est équivalente à la minimisation de la somme des régions AB, AC et BC.

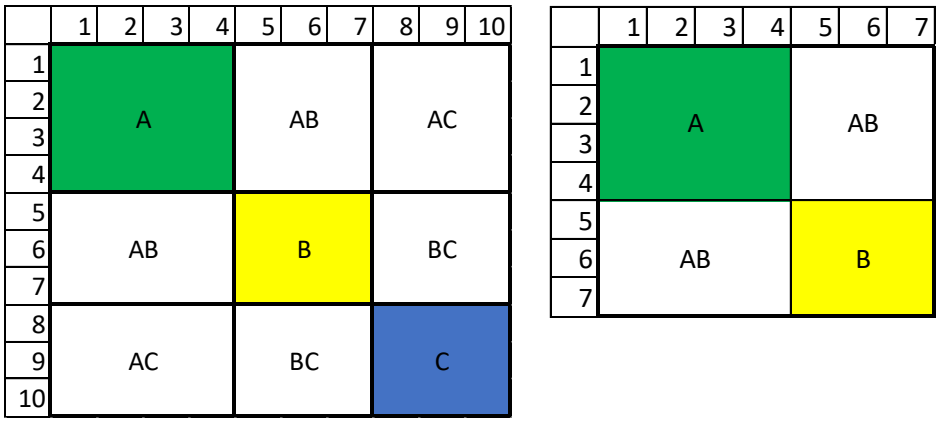
4. Division de la matrice et addition de modularité (localité)

   a. La maximisation de la modularité revient à effectuer des coupes dans la matrice de modularité afin d'avoir une diagonale positive symétrique. La modularité de chacune des communautés est calculée à partir de la somme de tous les arcs non adjacents d'une communauté de manière séparée. De plus, deux communautés peuvent être extraites et mises en relation sans influencer les restes de la matrice.

   b. En théorie, la matrice de modularité peut être divisée dans des sous-matrices où il y a deux types de matrices. Dans un premier temps, la matrice intragroupe et la matrice extra groupe.

c. Illustration :

Figure 11 : Représentation simplifiée de la modularité et extraction de deux groupes



Alors, l'extraction de matrices peut simplifier certains calculs ou codages dans certains cas avec des objets qui représentent les nœuds dans un groupe ainsi qu'une matrice de la relation intragroupe. Toutes les opérations dans les groupes extraits peuvent être effectuées sans affecter le reste de la matrice. Par exemple, les opérations de fusion, déplacement de nœuds et divisions de groupes peuvent être effectués de manière locale entre A et B.

Variables pour le calcul de *Q* global.

a. L'ordre des nœuds

b. Taille du groupe

Les deux variables pour la détection de communautés peuvent avoir plusieurs formes. La forme classique est qu'on associe un libellé commun à tous les nœuds d'une même communauté. En autres mots, nous créons des pots où l'on dépose des nœuds.

Dans le cas de la matrice de modularité, la formulation mathématique conserve l'intégrité complète de la matrice. Dans un premier temps, la variable de l'ordre et la coupe permet à l'algorithme de construire des communautés par la création de sous-matrices. Ce qui complexifie le problème de résolution est la détection d'améliorations sous différentes combinaisons. Bien que l'ordre à l'intérieur d'une communauté soit arbitraire et n'a pas d'influence sur la modularité, l'ordre à l'extérieur des groupes est important.

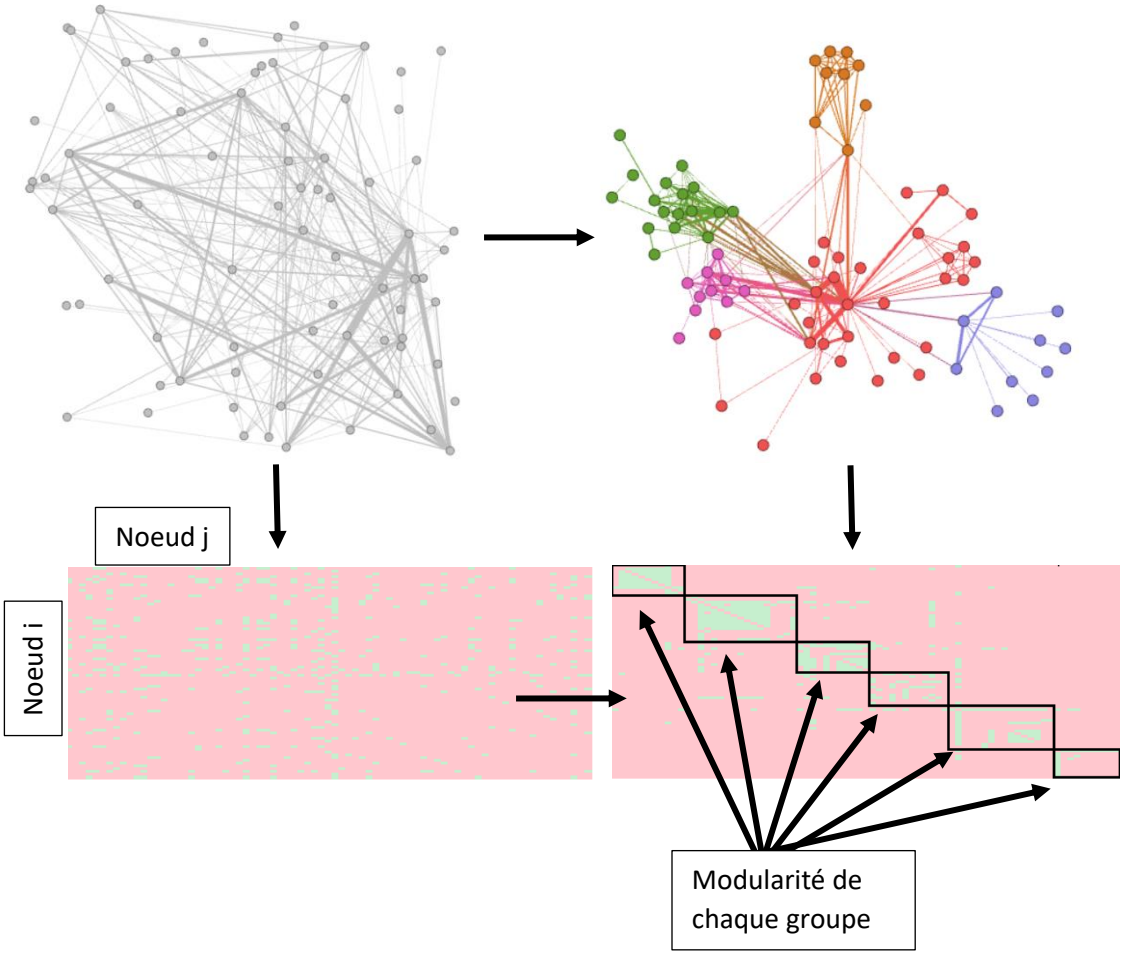
3.4. Fonction objective de l'algorithme

L'ensemble des algorithmes proposé dans ce mémoire n'ont pas de preuve d'optimalité Les algorithmes discutés dans ce mémoire suivent une maximisation de la modularité avec une approche de comparaison suivant l'idée de Clauset (2004). Voici la formulation de la fonction objective généralisée :

$$si\ max\ F_1(G) = Z(P_1) > Z(P_2) = max\ F_2(G),$$

alors algorithme F_1 produit de meilleurs partitions

Figure 12 : Représentation visuelle des transformations lors de la détection de communauté pour le réseau *Les misérables*



La figure 12 est un consolidé des concepts à ce jour. Les pixels verts signifient des arcs positifs et les pixels roses les arcs négatifs dans les deux matrices. Les arcs positifs sont

les nœuds adjacents et les arcs négatifs sont les arcs non adjacents. Les deux figures à gauche illustrent la matrice de modularité (B_{ij}) avec un réseau non partitionné tandis que les figures à droite sont partitionnées en se fiant sur la modularité. La formation de structures dans les figures à droite est évidente. De plus, ces structures sont différentiables d'une partition aléatoire et les arcs positifs sont plus fréquents entre les pairs d'une même communauté.

3.5. Indices de performance

1. ΔQ
    a. L'indice de performance pour la qualité de l'algorithme est le ΔQ qui représente l'écart entre la modularité de l'algorithme et la modularité optimale rapportée de [1].

$$\Delta Q = Q_{algo} - Q_{opt} \quad \textbf{(22)}$$

2. Le nombre d'itérations
    a. Le nombre d'itérations est important comme mesure de performance afin de déterminer la traction de l'algorithme pour les réseaux utilisés. Alors, peu d'itérations avec de meilleurs gains en modularité sont préférables à beaucoup d'itérations avec peu d'amélioration.
3. Temps de calcul
    a. Le temps de calcul (en ms) est la mesure de performance la plus importante pour observer la performance d'un algorithme pour différents réseaux. Ce critère de performance est également important afin de déterminer l'applicabilité de l'algorithme dans les plus gros problèmes. Cependant, cette mesure de performance est très sensible à la performance des ordinateurs utilisés.

Chapitre 4 :
Méthodologie

4.1. Méthodologie de développement

La méthodologie employée tout au long du développement de l'algorithme a été une méthode basée sur des comparaisons avec la solution optimale fournie dans [1] ainsi que les solutions (groupe de chaque nœud) en csv. L'analyse des écarts entre l'algorithme développé et les clusters optimaux ont été à une profondeur des structures de nœuds et les nœuds individuels. De plus, certains ensembles de données ont été utilisés qu'à des fins de validation afin de minimiser le biais de l'algorithme pendant que d'autres ont été exhaustivement disséqués. Voici les réseaux tirés de [1] pour l'analyse :

Tableau 1 : Tableau des réseaux analysés avec les données de [1]

Problem ID	Name	n	m	Densité	Q opt	M	Utilisation
1	Zachary's karate club	34	74	0.1319	0.4198	4	Développement
2	Dolphins social network	62	159	0.0841	0.5285	5	Développement
3	Les Misérables	77	254	0.0868	0.56	6	Développement
4	A00_main	83	135	0.0397	0.5309	9	Développement
5	protein p53	104	226	0.0422	0.5351	7	Test
6	Books about US politics	105	441	0.0808	0.5272	5	Développement
7	American College Football	115	613	0.0935	0.6046	10	Développement
8	A01_main	249	635	0.0206	0.6329	14	Développement
9	USAir97	332	2126	0.0387	0.3682	6	Test
10	netscience_main	332	2126	0.0387	0.8486	19	Test
11	Electronic Circuit (s838)	512	819	0.0063	0.8194	12	Test

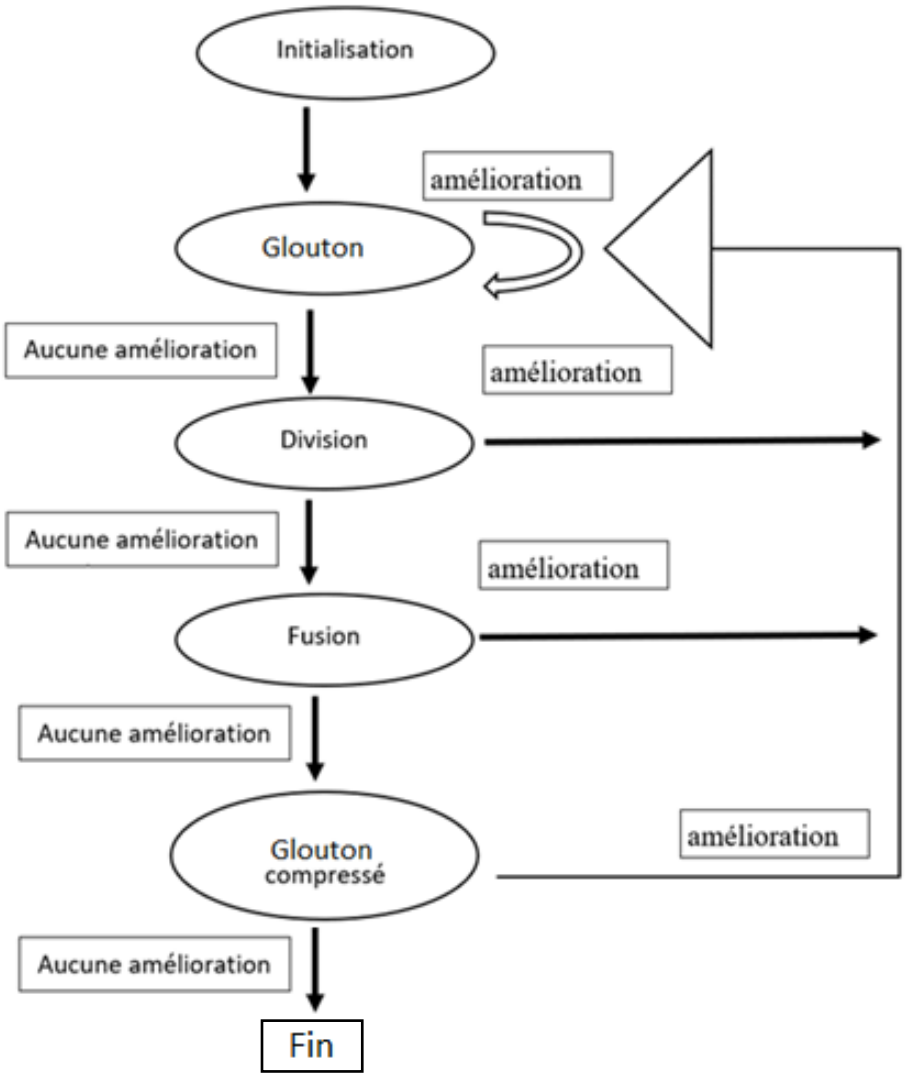
* Développement = Donnée qui a été analysée au niveau des nœuds pour l'algorithme.
* Test = Données qui ont seulement été utilisées par l'algorithme sans analyse approfondie des partitions.

Le tableau 1 illustre tous les réseaux analysés. Les réseaux plus petits ont été utilisés pour l'analyse et le développement à cause de leur taille pragmatique à l'analyse. Les plus gros réseaux ont été utilisés pour seulement l'évaluation de l'algorithme pour avoir des réseaux vierges.

Les étapes de l'algorithme ont été élaborées avec certains principes directifs afin d'augmenter la performance de l'algorithme ainsi que la précision. Les étapes successives sont basées sur une suite de la moins couteuse à calculer en temps de calcul vers les étapes plus lourdes. Cependant, il faut faire attention qu'une étape n'alourdit pas le calcul de l'étape successive. Le benchmark est le temps de calcul global.

L'algorithme utilise plusieurs étapes afin de contourner les sommets locaux en utilisant une fonction objective différente, mais qui améliore la solution à chaque étape.

Figure 13 : Description de l'algorithme.



L'algorithme effectue les étapes en séquence dans un ordre basé sur les améliorations de modularité. L'initialisation est effectuée une fois au début seulement. Par la suite, le

glouton va itérer tant qu'il y a amélioration. La division de groupe est entamée. Si une amélioration est trouvée, elle est appliquée et l'algorithme retourne à l'étape du glouton pour optimiser le réseau et recommencer le processus. Si aucune amélioration n'est retrouvée, l'algorithme procède aux fusions de groupes avec la même logique de répétition en cas d'amélioration. Le glouton compressé effectue le même processus que les étapes de division et de fusion au niveau de la séquence. Si aucune amélioration est constatée pour cette dernière étape, l'algorithme s'arrête.

4.2. Initialisation

L'objectif de l'initialisation est de trouver une solution réalisable avec une modularité suffisamment élevée afin de raccourcir le temps de la recherche de l'algorithme. Une modularité de départ plus basse augmente généralement le temps de calcul à cause des itérations subséquentes dans les autres paliers. L'initialisation est effectuée une fois au début de l'algorithme.

La procédure d'initialisation est basée sur l'itération de chaque nœud qui est laissé à dresser une liste de ses meilleurs amis potentiels en fonction du plus grand gain de modularité. Cette procédure s'apparente au choix de joueurs dans un cours d'éducation physique où chaque nœud aurait l'opportunité d'être le capitaine de l'équipe. Cependant, lorsque le groupe devient supérieur à 1 nœud, les membres rajoutés dans le groupe contribuent aux liens du capitaine par leur gain de modularité. Le critère d'arrêt est lorsqu'aucun gain en modularité n'est possible en rajoutant un membre. Vu l'indépendance des listes, cette partie est parallélisée afin d'améliorer le temps de calcul.

Par la suite, les listes de choix sont comparées et les nœuds qui ont les mêmes listes sont consolidés dans des clusters. Étonnement, la variation des équipes ou des listes sont très basses et elles sont souvent identiques pour les nœuds qui sont à proximité. En bref, l'initialisation est basée sur le fait que les nœuds qui se ressemblent, s'assemblent si nous enlevant toute autre contrainte.

Algorithme 1 : Pseudo-code de la procédure d'initialisation

Variables:

$noeud_i$: $position\ du\ noeud\ i\ dans\ B_{ij}$
$groupefinal_i$: $groupe\ final\ du\ noeud\ i$
$listfinale_i$: $Liste\ en\ ordre\ des\ noeuds\ appartenant\ au\ noeud\ i$
$links_i$: $liste\ d'arcs\ du\ groupe$
$groupe_i$: $Liste\ de\ noeuds\ dans\ le\ groupe$ i

1: **For all** $nœuds$ (en parallèle)
2: **switch** $nœud_i$ et $nœud_1$
3: Stop=0
4: **While** (stop == 0)
5: **for all** $links_i$
6: $s = \max_{1<j<l} 2 * \sum_{i=1}^{i=l} B_{groupe_i links_j} - B_{links_j links_j}$
7: **If** (s < 0)
8: stop = 1
9: **else**
10: **add** $noeud_i\ dans\ groupe_i$
11: **add all different** $links_i$ de $noeud_i$ à $links_i$
12: **Sort** (ordre croissant) $groupe_i$
13: $listfinale_i = groupe_i$
14: **for all** $nœuds$
15: **if** ($listfinal_i$ existe parmi les groupes)
16: nœud i est dans le groupe
17: **else**
18: créer nouveau groupe

4.3. Étape 1 : Glouton

Une fois l'initialisation terminée, l'étape du glouton est activée pour stabiliser le réseau advenant qu'un nœud ne soit pas placé dans le meilleur groupe ou un gain de modularité soit possible en bougeant un nœud d'un groupe à un autre avec un minimum de coût. De plus, cette étape est activée le plus souvent et lorsqu'une amélioration n'est détectée dans les étapes suivantes. L'algorithme global recommence à cette étape.

L'algorithme du glouton va itérer à travers de chaque nœud et va vérifier si un gain de modularité est possible en changeant un nœud de groupe. Alors, le glouton va calculer la valeur rajoutée du déplacement d'un nœud d'un groupe envers tous les groupes avec la matrice de la modularité jusqu'à ce qu'aucun gain ne soit possible au niveau des nœuds individuels.

Les fonctions de l'algorithme du glouton :

Variables :

$$k : Groupe\ k, \qquad \forall\ k\ \in N\ [1, g]$$
$$l : Groupe\ l\ actuel\ du\ noeud\ i, \qquad \forall\ l\ \in N\ [1, g]$$

$$QV_{i\,l\,k} : Contribution\ net\ du\ noeud\ i\ du\ groupe\ l\ vers\ le\ groupe\ k\ (Note^{1})$$
$$QG_{\ i\,k} = Contribution\ direct\ du\ noeud\ i\ au\ groupe\ k$$
$$QG_{\ i\,l} = Contribution\ direct\ du\ noeud\ i\ au\ groupe\ l\ (\text{Note}^{1})$$

Formules :

$$QG_{i\,k} = \sum_{j=1}^{j=n} B_{ij} * \delta(C_k, C_j),\ o\grave{u}\ C_k = communit\acute{e}\ k \tag{23}$$

$$QV_{i\,l\,k} = 2(QG_{\ i\,k} - QG_{\ i\,l} - B_{i\,i}) \tag{24}$$

[1] Les variables $QV_{i\,l\,k}$ et $QG_{\ i\,l}$ ont une redondance d'information par rapport aux indices afin de faciliter la compréhension. En réalité, $QV_{i\,l\,k}$ peut être noter $QV_{i\,k}$ et $QG_{\ i\,l}$ peut être généralisé à $QG_{\ i\,k}$.

Fonction objective :

$$\max_{i,l,k}(QV_{i\,l\,k}) \qquad (25)$$

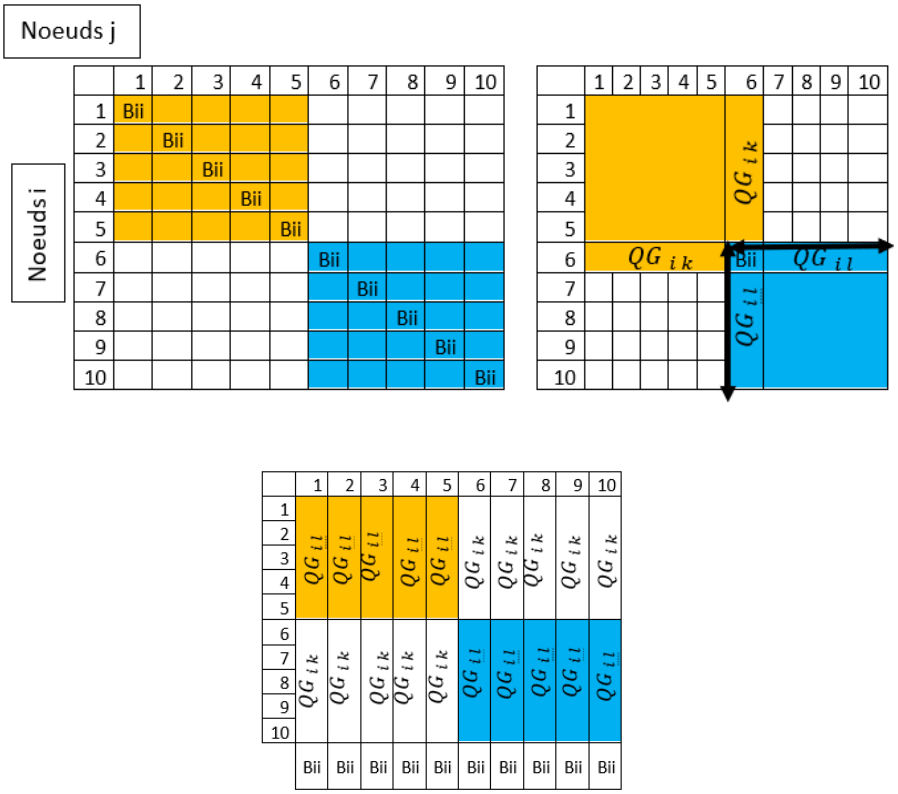
$$Nouveau\ Q = Q + QV_{i\,l\,k}\ (\text{Q si mouvement du noeud i vers le groupe } l)$$

$$Si\ QV_{i\,l\,k} > 0, ceci\ veut\ dire\ un\ gain\ de\ modularité$$

Donc, deux situations sont possibles pour un nœud i:

1. $QG_{\,i\,k} > QG_{\,i\,l} - B_{i\,i}$: Gain de modularité du déplacement du nœud i vers un autre groupe
2. $QG_{\,i\,k} < QG_{\,i\,l} - B_{i\,i}$: Baisse de modularité du déplacement du nœud i vers un autre groupe

Figure 14 : Représentation simplifiée de l'étape du glouton



La figure 14 est une illustration graphique des formules sur la matrice de modularité. La partie colorée représente les arcs qui sont partagés entre deux nœuds dans une même

communauté. La diagonale de $B_{i\,i}$ représente le poids de modularité d'un arc entre un nœud et lui-même. Les régions $QG_{i\,k}\ et\ QG_{i\,l}$ représente les régions sommées pour le calcul d'une valeur ajoutée.

Plusieurs optimisations de l'algorithme sont possibles afin de limiter le domaine de recherche : une parallélisation ainsi qu'une compression avec une nouvelle matrice *i* par *k* avec l'ensemble des $QG_{i\,k}$ qui requiert des mises à jour. Dans l'implémentation de l'algorithme, cette étape est suffisamment rapide pour ne pas avoir à suroptimiser avec un gain minimal surtout avec une parallélisation de la recherche. De plus, ces formules sont employées dans l'étape 3 d'une manière abrégée pour générer les fusions de groupes. Une parallélisation a été employée afin de gagner du temps. La complexité initiale de l'algorithme est de $O(n^2)$ à cause de la sommation de chaque ligne pour chaque nœud. Cependant, une optimisation de l'algorithme pourra le réduire à O(n*k) pour les itérations subséquentes si la matrice est seulement amendée au lieu de recalculer. Ceci enlèverait la nécessité de calculer chaque ligne. Alors, le calcul s'effectue pour chaque groupe sur chaque ligne.

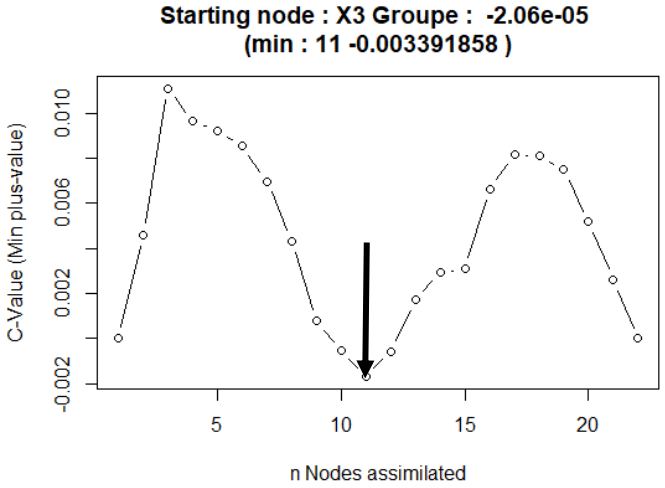
4.4. Étape 2 : Division de groupes

La division de groupes est la partie critique de l'algorithme avec plus des trois quarts du temps de calcul. Malgré la lourdeur, la procédure de division de groupes a un rôle important pour la qualité de la solution. De plus, la division de groupes permet l'ajout ou la liberté à l'algorithme de créer un nouveau groupe au besoin tout en améliorant la modularité.

La division de groupe est fondée sur une minimisation d'une coupe de la matrice de modularité. Dans un premier temps, l'algorithme va extraire une communauté de manière itérative et va procéder à tester tous les nœuds pour commencer la division. Par la suite, de manière itérative sur chaque nœud, la procédure va séparer le groupe avec un groupe ayant un seul nœud (Groupe 1) et le reste (Groupe 2). L'algorithme choisit le candidat du groupe 2 qui minimise la réduction de modularité. Une fois que le moins pire nœud est annexé au groupe 1, un nœud subséquent est pris du groupe 2 avec la même condition d'être le moins pire en termes de réduction de modularité. L'algorithme est ensuite obligé

d’effectuer ces échanges de groupes avec un nœud à la fois. L’obligation de choisir un candidat à chaque itération s’explique par l’obligation de dépasser le sommet local initial. Les premières itérations réduisent la modularité locale suivie d’une amélioration s’il y a lieu. À chaque itération, la somme de l’entre-deux groupes est enregistré.

Figure 15 : Exemple d’un graphique des valeurs de séparation d'un groupe qui a une valeur ajoutée à la séparation



Une fois que tous les nœuds du groupe 2 ont traversé au groupe 1, si la somme des arcs à l’extérieur des communautés est devenue négative, ceci veut dire une amélioration de *Q* globale par une séparation telle que la figure 15 illustre. Cette figure démontre le processus itératif de la séparation d’un groupe en forçant les nœuds à changer de groupe. À l’initialisation, la valeur ajoutée est égale à zéro. Par la suite, une augmentation des pertes en modularité en séparant le groupe est enregistrée. Dans le cas d’une séparation d’un groupe qui consistait de deux sous-groupes, un minimum négatif est détecté.

Variables :

$$k : Groupe\ k, \qquad \forall\ k\ \in N\ [1, m]$$

$a : Scalaire\ qui\ représente\ la\ somme\ de\ la\ modularité\ du\ groupe\ 1$

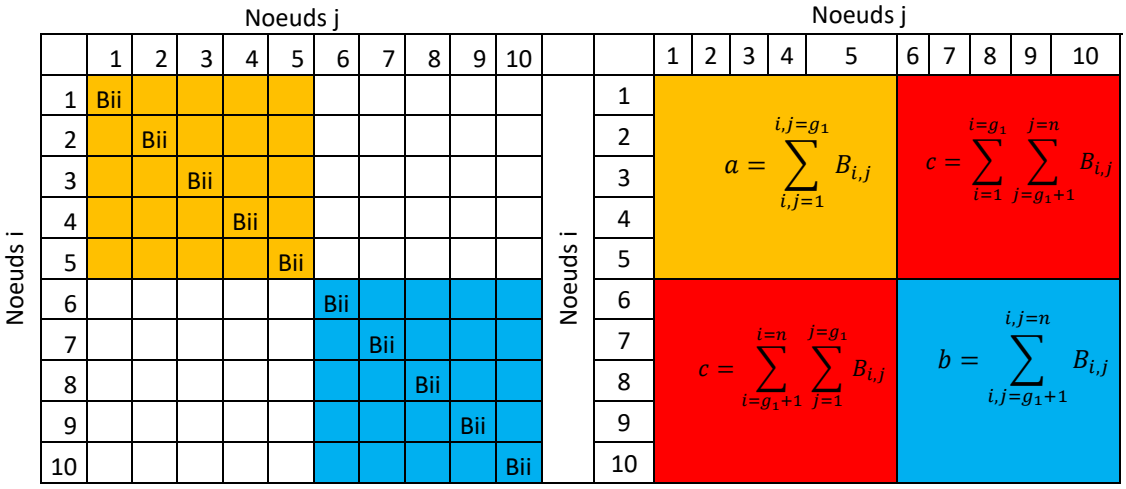
$b : Scalaire\ qui\ représente\ la\ somme\ de\ la\ modularité\ du\ groupe\ 2$

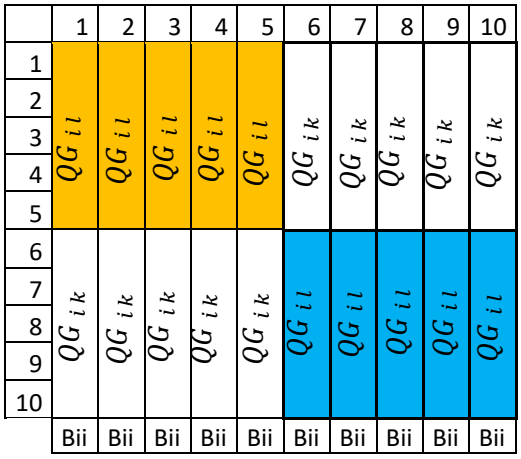
$c : Scalaire\ qui\ représente\ la\ somme\ de\ la\ modularité\ de\ l'entre\ deux\ groupes$

$g_1 : Emplacement\ du\ dernier\ noeud\ du\ groupe\ 1$

$QP_k : Modularité\ (partiel)\ groupe\ initial\ k$

Figure 16 : Représentation graphique des formules de la division de groupes





Rappel des formules :

$$QG_{i\,k} = \sum_{j=1}^{j=n} B_{ij} * \delta(C_k, C_j),$$

$$où\ C_k = communité\ k$$

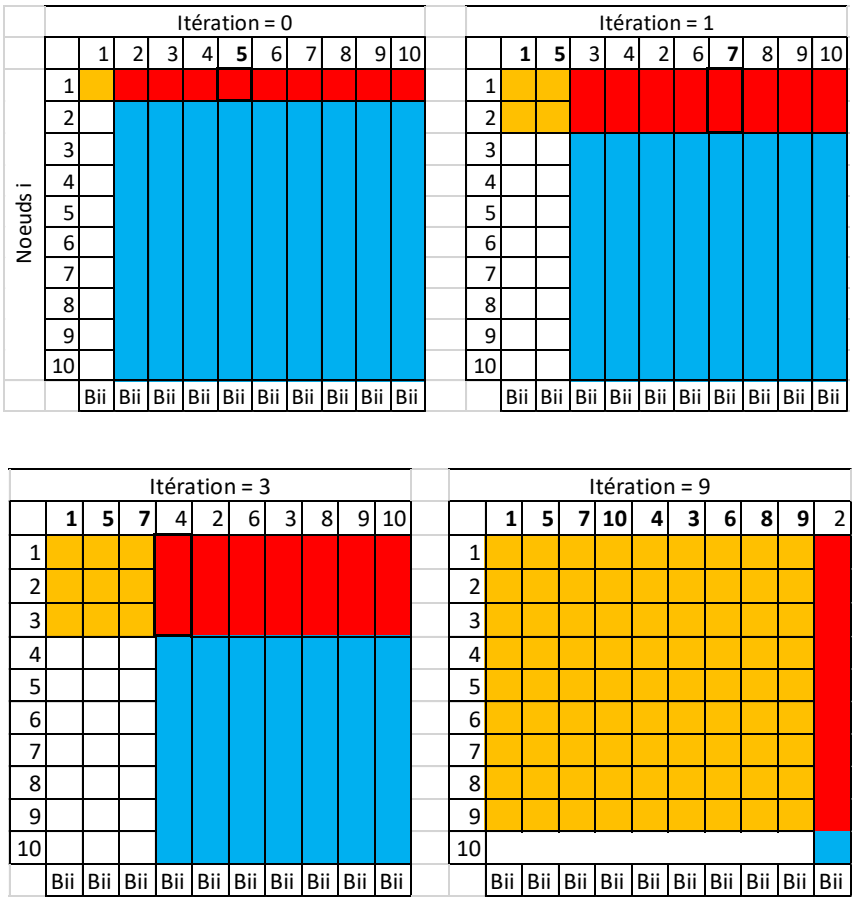
$$QG_{i\,l} = \sum_{j=1}^{j=n} B_{ij} * \delta(C_l, C_j),$$

$$où\ C_l = communité\ l$$

La figure 16 est une représentation graphique des formules employées à chaque itération afin de diviser le groupe avec un gain de modularité si applicable. Trois régions sont importantes lors du calcul : la région du groupe 1 (orange), la région du groupe 2 (bleu)

et la région entre les deux groupes (rouge). Vu que l'ensemble de la matrice est une communauté, la division de groupe est un problème de minimisation de c. En autres mots, la division de groupes est l'inverse de la fusion de groupes et nous cherchons la meilleure coupe de la matrice. Cependant, la division de groupe est plus lourde à calculer que la fusion à cause d'un ensemble plus gros de combinaisons.

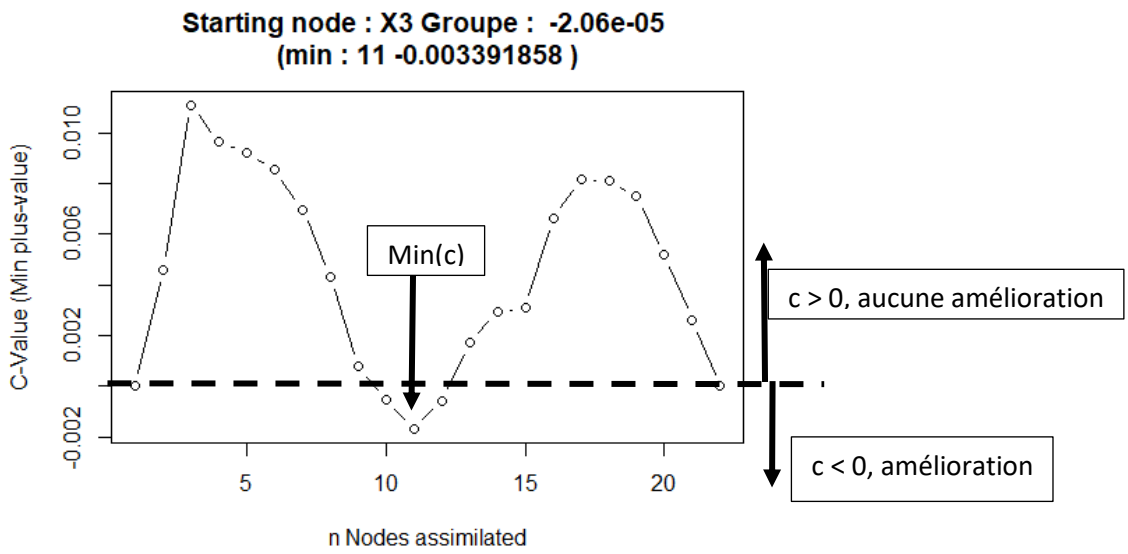
Figure 17 : Graphiques simplifiés des itérations de la division de groupes



$$\min_{iter<i\leq n} (QG_{i\,l} - QG_{i\,k} - B_{ii}) \text{ , où } iter : Iteration \qquad \textbf{(26)}$$

Les itérations de séparations de groupes sont effectuées tel qu'illustré dans la figure 17 avec une itération à la fois qui minimise c avec l'obligation de pivoter un individu vers le nouveau groupe. Le choix est évalué par la formule **(26)** où le nœud ayant la valeur la plus petite est pivoté dans le groupe 1 et la prochaine itération débute avec une rangée supplémentaire. Tous les résultats sont enregistrés et produisent un graphique comme la figure 18 :

Figure 18 : Illustration d'un exemple des valeurs graphiques d'une séparation de groupe



Formules :

$$a = \sum_{i,j=1}^{i,j=g_1} B_{i,j} \tag{27}$$

Formule de la modularité du groupe 1.

$$b = \sum_{i,j=g_1+1}^{i,j=n} B_{i,j} \tag{28}$$

Formule de la modularité du groupe 2.

$$c = \sum_{i=g_1+1}^{i=n} \sum_{j=1}^{j=g_1} B_{i,j} = \sum_{i=1}^{i=g_1} \sum_{j=g_1+1}^{j=n} B_{i,j} \tag{29}$$

Formule de l'entre-deux groupes (groupe 1 et groupe 2). Mêmes formules que la Figure 16.

$$QP_k = \sum_{i,j=1}^{i,j=n} B_{i,j} = a + b + 2c \tag{30}$$

Formule des composantes de la modularité partielle totale du groupe mère

$$-2c = a + b - QP_k \tag{31}$$

Formule de la mise en évidence de l'entre-deux groupes.

Fonction objective :

$$\max(-2c) = \max(a + b - QP_k) \quad \textbf{(32)}$$

$$\min(c) = \frac{-\max(a + b - QP_k)}{2} \quad \textbf{(33)}$$

Donc, 2 situations sont possibles:

1. $c > 0$: Aucun gain de modularité de la séparation
2. $c < 0$: Gain de modularité de la séparation

L'algorithme de la séparation de groupes n'est pas sans difficulté, car les solutions sont sensibles aux premiers nœuds qui se joignent au groupe 1. Plusieurs candidats au début ont des valeurs inférieures à l'epsilon ou son équivalent. Alors, afin de contourner ce problème, l'algorithme va itérer à travers tous les nœuds en mettant chacun comme nœud initial jusqu'à une première solution avec un c inférieur à 0 est identifié. De plus, l'implémentation va garder un registre des groupes sur lesquels une division sans succès a été effectuée pour les éviter. L'algorithme donne généralement de bons résultats et la parallélisation a été implémentée où chaque cœur de processeur traite un groupe.

4.5. Étape 3 : Fusion de groupes

La procédure de fusion de groupes est la portion la plus facile et rapide à implémenter. En reprenant les formules du glouton, la somme de l'entre-deux groupes va produire la valeur ajoutée en *Q* de la fusion de groupes. Alors, l'algorithme va itérer avec une sommation de tous les entre-groupes afin de trouver une valeur positive. Lorsque la valeur positive est retrouvée, les deux groupes sont fusionnés. Lorsque tous les entre-groupes sont testés sans amélioration, l'algorithme procède à la prochaine étape. Les fonctions de l'algorithme de fusion de groupe sont présentées ci-dessous :

Variables :

$$g : Nombre\ total\ de\ groupes\ actuels$$

$$k : Groupe\ k,\ \ k\ \in N\ [1, g]$$

$$g_{1\,k} = noeud\ de\ départ\ du\ groupe\ k$$

$$g_{2\,k} = noeud\ de\ fin\ du\ groupe\ k$$

$$QV_{k1,k2} : Valeur\ ajouté\ de\ la\ fusion\ du\ groupe\ k1\ et\ le\ groupe\ k2$$

Formules :

$$QV_{k1\,k2} = \sum_{i=g_{1\,k1}}^{g_{2\,k1}} \sum_{j=g_{1\,k2}}^{g_{2\,k2}} B_{i\,j} \tag{34}$$

Fonction objective :

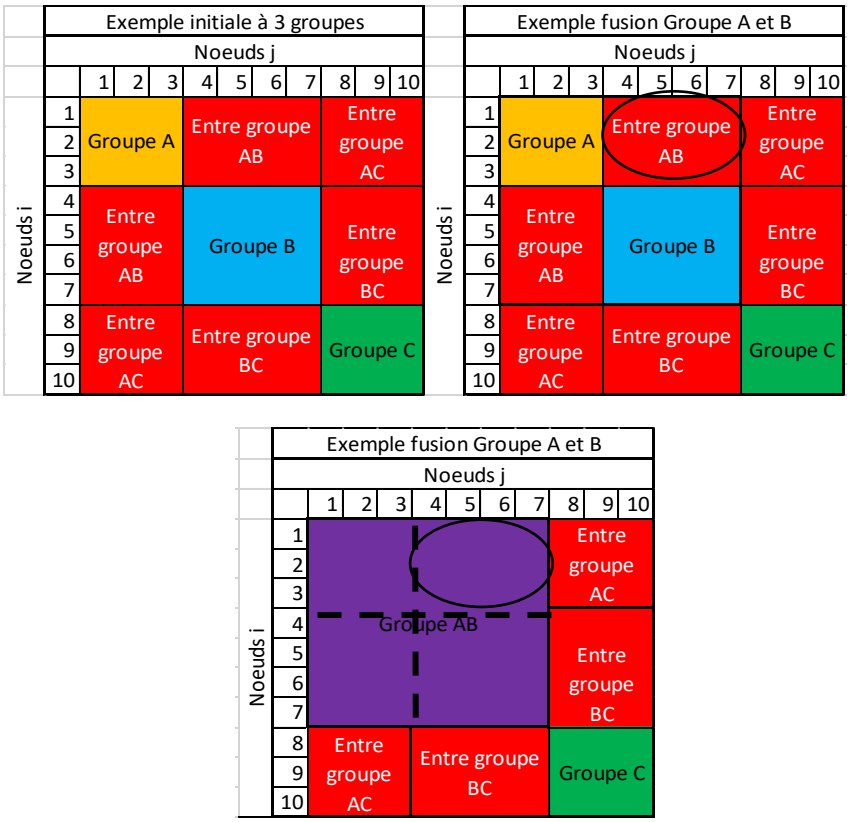
$$\max_{k1,k2}(QV_{k1\,k2}) \tag{35}$$

$$Q + Gain = 2 * QV_{k1\,k2} \tag{36}$$

Donc, deux situations sont possibles :

1. $QV_{k1\,k2} > 0$: Gain de modularité de la fusion du groupe k1 et k2
2. $QV_{k1\,k2} < 0$: Baisse de modularité de la fusion du groupe k1 et k2

Figure 19 : Graphique matriciel simplifié des aires pour la fusion de groupe



La figure 19 est une simplification de l'espace matricielle en superficie. Les régions orange, bleues et vertes sont les régions qui décrivent les arcs intragroupes de la matrice de modularité. Toute la région rouge est la superficie d'arcs entre les groupes. La région mauve est une fusion des groupes A et B. Le calcul de la fusion de groupes est la somme de l'entre-groupe d'une paire de groupes vu que la seule différence est cette région multipliée par deux lorsque deux groupes sont fusionnés. Puisqu'on considère un objectif de maximisation et que la matrice est symétrique, seul un entre-groupe par paire de groupes est calculé.

Une parallélisation est aussi implémentée puisque la sommation est indépendante pour chaque entre-groupe. De plus, une matrice simplifiée du glouton est utilisée pour réduire le nombre de calculs redondants. Il est possible, mais non implémenté de seulement ajuster les totaux des entre-groupes afin de réduire davantage le nombre de calculs.

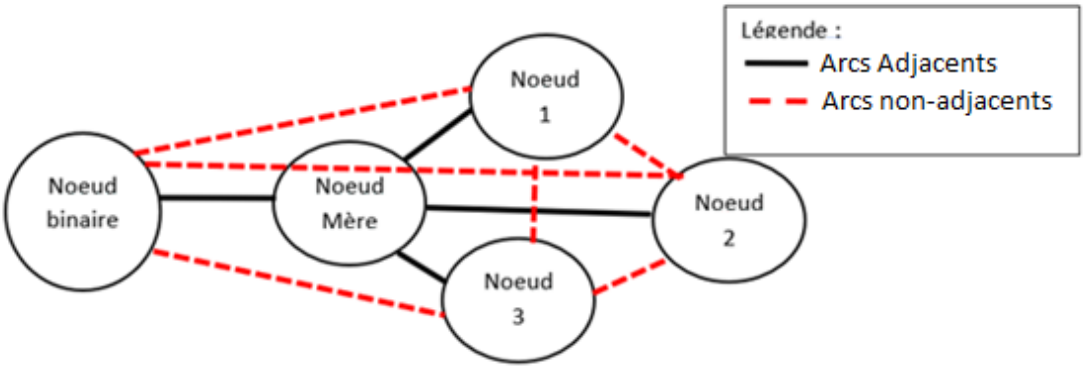
4.6. Étape 4 : Glouton compressé

Le glouton compressé est un algorithme qui utilise la même approche que le glouton initial avec la variation d'être calculé avec la fusion de deux ou plusieurs nœuds qui sont dépendants. Initialement, le glouton compressé était une tentative d'exploration et d'amélioration au premier algorithme du glouton. Malgré les attentes d'une amélioration vu la capacité de déplacer plusieurs nœuds à la fois, l'algorithme ne retrouve pas une qualité de réponse aussi élevée lorsqu'il est mis à la première étape à cause des arcs négatifs. Le glouton compressé améliore davantage la modularité lorsque réalisé à la fin de l'algorithme.

L'algorithme prend la liste d'adjacence initiale et fusionne les nœuds qui ont une dépendance forte telle que les nœuds qui ont seulement un lien. Par la suite, les nœuds qui ont seulement un arc envers un autre nœud sont fusionnés pour le calcul avec un glouton simple. Vu que le glouton simple évalue seulement un nœud à la fois, lorsqu'il y a un nœud binaire (nœud ayant un seul arc adjacent), le nœud mère n'a aucun gain à cause de son nœud dépendant. Alors, la fusion de ces deux nœuds est effectuée pour le calcul du glouton. Une fois qu'aucune amélioration est détectée après une recherche exhaustive, l'algorithme a terminé.

Cependant, le problème de cette approche est que lorsque la modularité est calculée pour tous les nœuds, tous les nœuds reçoivent des arcs avec des probabilités envers tous les autres nœuds comme le montre la figure 20.

Figure 20 : Représentation des arcs adjacents et les arcs non adjacents



La figure 20 illustre les arcs présents en noir ainsi que les arcs produits de la modularité. Ces deux types d'arcs sont les arcs positifs et négatifs qui lient tous les nœuds entre eux. À cause des arcs négatifs, la compression des nœuds mène à une réduction de l'efficacité du glouton compressé.

Lorsque cette variation du glouton est appliquée à la fin, le glouton compressé est capable de détecter des nœuds manqués par l'algorithme sans réduire la qualité des résultats finaux. Le glouton compressé est alors utilisé comme procédé de finition. Cette approche fusionnée pourrait être utilisée conjointement avec l'algorithme de Blondel [3] ou avec une détection de géométries de nœuds. Avec une bibliothèque ou librairie de géométrie des clusters de nœuds, il est envisageable de grouper des nœuds qui ont une certitude ou une grande probabilité d'être ensemble. Ces clusters avec un arrangement géométrique spécifique pourraient être détectés dès le départ avec un optimum local pour ce cluster qui ne changera pas.

Chapitre 5 :
Résultats

Tous les tests et les simulations ont été effectués en C++ avec un Lenovo yoga 720 qui possède un processeur quad-cœur i7-7700HQ cadencé à 2.8GHz et 2GB de mémoire. Les algorithmes sont parallélisés avec une librairie « Open MP » en C++.

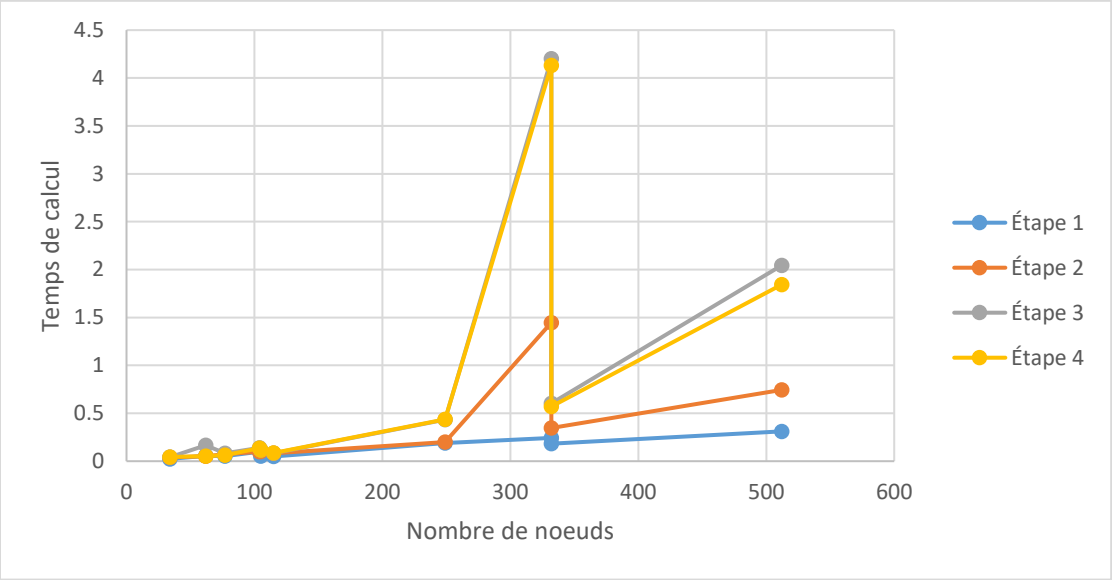
5.1. Temps de calcul

L'algorithme dans son ensemble performe avec une valeur de modularité dans les environs de quelques millièmes de la solution optimale. De plus, la structure du réseau fait varier considérablement le temps de recherche d'une solution. Lorsque la valeur est faible, le temps de calcul est agrandi et la solution ne converge pas facilement vers une solution. À l'inverse, l'algorithme va converger plus rapidement lorsque la structure modulaire du réseau est claire, i.e. lorsque la valeur de la modularité optimale est grande. La figure 21 illustre les résultats de temps de calcul en fonction de la taille du réseau.

Tableau 2 : Temps de calcul en fonction de la taille du réseau

				Temps par étape (en secondes)			
Nom du réseau	Q opt	Densité	n	1	2	3	4
Zachary's karate club	0.4198	0.1319	34	0.0240	0.0390	0.0420	0.0440
Dolphins social network	0.5285	0.0841	62	0.0570	0.0520	0.1650	0.0540
Les Misérables	0.5600	0.0868	77	0.0510	0.0730	0.0810	0.0640
protein p53	0.5351	0.0422	104	0.0990	0.1000	0.1390	0.1350
Books about US politics	0.5272	0.0808	105	0.0510	0.1020	0.1160	0.1160
American College Football	0.6046	0.0935	115	0.0500	0.0820	0.0840	0.0870
A01_main	0.6329	0.0206	249	0.1880	0.1990	0.4340	0.4380
USAir97	0.3682	0.0387	332	0.2420	1.4470	4.2000	4.1310
netscience_main	0.8486	0.0387	332	0.1840	0.3460	0.6050	0.5690
Electronic Circuit (s838)	0.8194	0.0063	512	0.3110	0.7450	2.0430	1.8430

Figure 21 : Temps de calcul en fonction de la taille du réseau



Le graphique de la figure 21 extrait du Tableau 2 illustre une tendance positive entre la grandeur des réseaux et le temps de calcul. Naturellement, une croissance de la grosseur d'un réseau mène à un temps supérieur de calculs. Cette tendance est plus marquée pour l'étape 3 à cause de la lourdeur de la division de groupes. De plus, une anomalie est détectée avec le réseau de «US Air 97» à 332 nœuds. Ce réseau a une modularité très basse et un temps de calcul très grand. Le réseau est reconnu pour être choisi pour son absence de structure ou de communautés fortes. L'analyse des itérations de «US Air 97» illustre un très grand nombre d'itérations avec un gain marginal de modularité. Alors, l'algorithme a eu un temps de calcul plus grand et un nombre d'itérations supérieur.

Le compromis entre vitesse, qualité et robustesse d'un algorithme a mené à l'expérimentation d'un algorithme en étapes. Bien que les techniques employées soient différentes de la littérature, certaines conclusions ou constatations sont intéressantes. De plus, effectuer un algorithme avec des étapes distinctes avec une variation dans les fonctions objectives est efficace afin de contourner les sommets locaux. Malgré l'amélioration marginale de l'étape 4, le coût en temps d'une itération est une des plus petites malgré qu'elle doit attendre que l'étape 3 cherche exhaustivement l'ensemble de solutions. L'étape 4 a un temps de calcul marginalement plus élevé que l'étape 3 dans les Figure 20Figure 21. Alors, cette étape est conservée pour avoir de meilleurs résultats.

5.2. Qualité des résultats

La qualité des résultats provient d'une comparaison entre le *Q* optimal et les résultats. Rappelons les méthodes présentées dans ce mémoire sont des heuristiques, sans preuve d'optimalité. Les résultats en étapes illustrent une amélioration en fonction de la grosseur du réseau. Le Tableau 3 illustre les résultats de l'algorithme en étapes :

Tableau 3 : Tableau des résultats en *Q* de chaque réseau en comparaison avec le *Q* optimal

		Étapes					
Nom du réseau	Q opt	1	2	3	4	Delta max	% delta max
Zachary's karate club	0.4198	0.4198	0.4198	0.4198	0.4198	0.0000	0.00%
Dolphins social network	0.5285	0.5277	0.5277	0.5277	0.5277	-0.0008	-0.15%
Les Misérables	0.5600	0.5600	0.5600	0.5600	0.5600	0.0000	0.00%
Protein p53	0.5351	0.4859	0.5170	0.5235	0.5309	-0.0042	-0.79%
Books about US politics	0.5272	0.4979	0.5220	0.5272	0.5272	0.0000	0.01%
American Col. Football	0.6046	0.5412	0.6046	0.6046	0.6046	0.0000	0.00%
A01_main	0.6329	0.5990	0.6134	0.6283	0.6288	-0.0041	-0.64%
USAir97	0.3682	0.3405	0.3506	0.3630	0.3630	-0.0052	-1.41%
Netscience_main	0.8486	0.7732	0.8368	0.8471	0.8472	-0.0014	-0.16%
Electronic Circuit (s838)	0.8194	0.7144	0.7958	0.8165	0.8167	-0.0027	-0.33%
Min						-0.0042	-0.79%

Les résultats attendus sont qu'à chaque étape rajoutée, une amélioration de modularité est accompagnée à part dans le cas d'une modularité maximale atteinte. Les problèmes plus petits sont résolus par les premières étapes rapidement généralement. Cependant, les étapes subséquentes sont importantes lorsque les problèmes augmentent en nombre de nœuds.

L'algorithme en étapes a eu de la difficulté sur les réseaux qui ont une basse modularité ou une qualité de réseau faible. L'analyse de clusters et des différences entre les clusters optimaux illustrent que les sommets sont très éloignés entre eux avec de grands creux lors des itérations. La structure peu dense ou moins structurée indiquée par une faible modularité du réseau résulte dans des configurations à haute modularité très distinctes. La qualité du réseau marqué par un *Q* élevé à l'optimalité ont un effet de faire converger

l'algorithme plus rapidement. Le nombre de configurations de partitions pour les valeurs semblables de modularité est plus faible également.

5.3. Performance des étapes

L'algorithme va emprunter une trajectoire de résolution à chaque exécution et l'analyse de chaque itération produit un graphique de l'amélioration en fonction du temps. La figure 22 illustre l'amélioration en fonction du temps pour un réseau test. Tel qu'attendu, le gain suit une courbe de rendement décroissant avec les premières itérations qui ont la plus grande amélioration. Les sauts dans les graphiques sont dus à des divisions de groupes et aux fusions qui dépassent un sommet local. Toutes les courbes produisent un aplatissement de la courbe lorsque la modularité se rapproche de son maximum. De plus, la majorité du temps de calcul est lié à l'allongement des itérations en temps à cause de l'exhaustivité des recherches vers la fin de l'algorithme. Ces itérations sont nombreuses et longues avec un gain très faible de modularité. La figure 22 illustre un gain plus grand qu'attendu initialement lorsqu'il y a des divisions de groupes et des fusions. Ces étapes forment la majorité des gains en modularité dans un temps rapide. De plus, les étapes de division et de gain sont visuellement détectables sur un graphique à cause des sauts en modularité. Tous les graphiques de la modularité en fonction du temps ressemblent à des marches avec un gain considérable au début.

Figure 22 : Graphique de la comparaison entre 4 réseaux sur l'amélioration en fonction du temps standardisé[1] à la valeur de départ

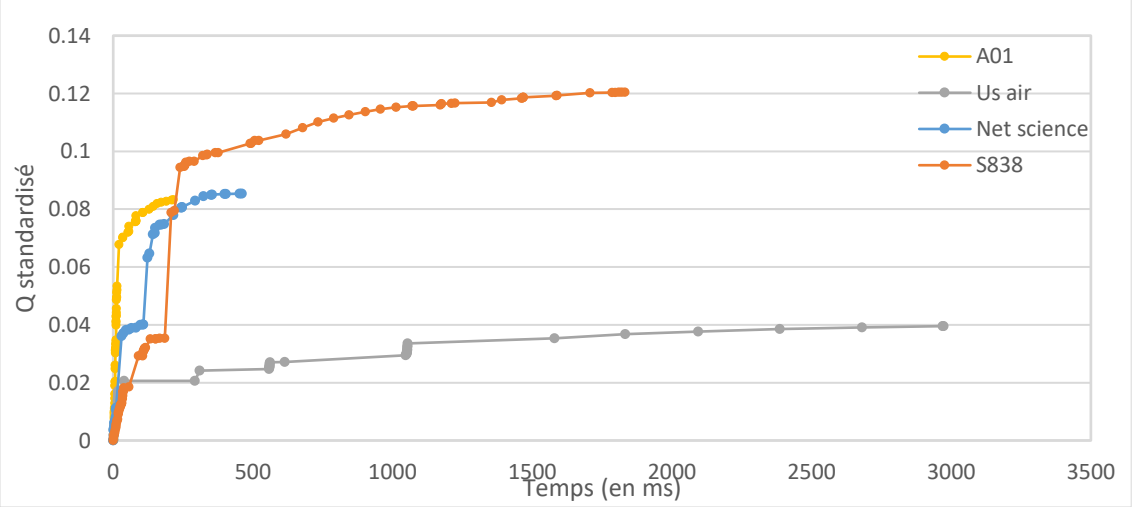
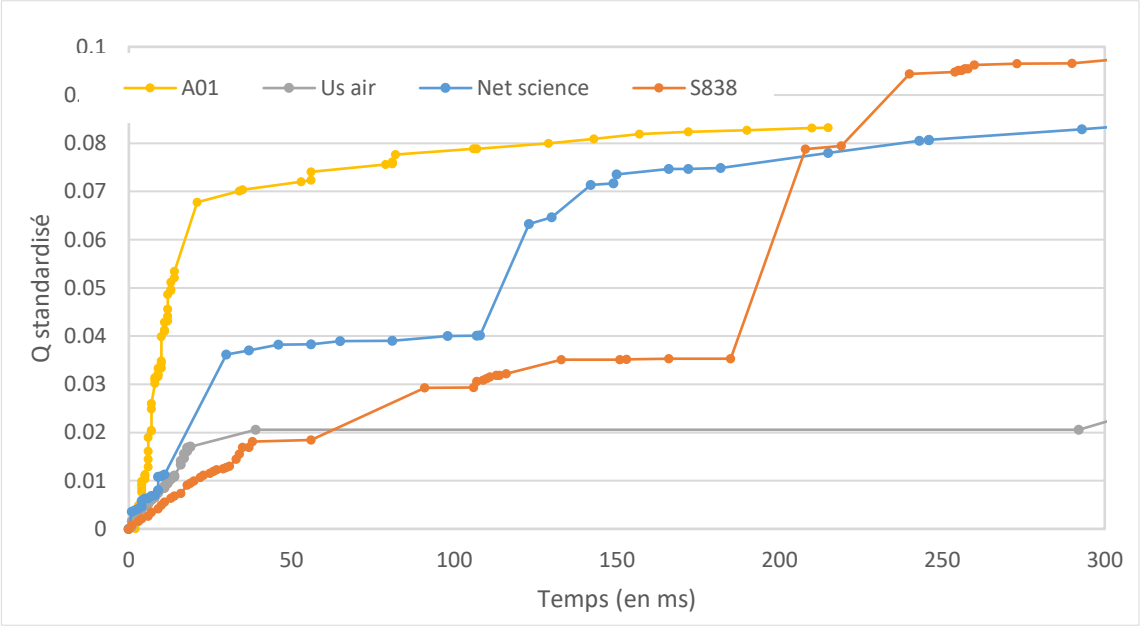


Figure 23 : Graphique de la comparaison de 4 réseaux de l'amélioration en fonction du temps standardisé[1] à la valeur de départ pour les premiers 300 ms.



1. La figure 22 est standardisée avec une soustraction de la valeur *Q* de départ et le temps de l'initialisation afin de comparer les améliorations des itérations sans le décalage de modularité naturel des réseaux.

La figure 22 illustre la modularité de l'ensemble des partitions d'un réseau en fonction du temps. En réalité, ces courbes sont discrètes où les points sont les mesures importantes afin d'évaluer la modularité. Les courbes continues sont seulement présentes afin de mieux visualiser la performance de l'algorithme dans le temps. Les quatre courbes démontrent le rendement décroissant de chaque itération représentée par les points. Le rendement décroissant est dû à l'allongement du temps de calcul de chaque itération et non à une augmentation du nombre d'itérations. Cet allongement du temps de calcul de chaque itération est une conséquence de recherche plus exhaustive de la matrice de modularité. Alors, un échange de temps pour plus de précision est présent. Après l'initialisation, le gain de modularité a un gain de modularité très rapide suivi d'un aplatissement. La figure 23 est un élargissement de la figure 22 qui illustre les points dans les premiers 300 ms. La majorité des points ou itérations sont dans les premiers 300 ms à partir du début de l'étape 1. Un autre effet intéressant est que dans certains cas une

amélioration est suivie d'un petit groupe d'améliorations subséquentes. Les sauts de modularité dans les figures sont dus à la division de groupe ou d'un nœud qui a été mis en évidence par le déplacement d'un nœud adjacent.

5.4. Trajectoire de l'algorithme

La trajectoire de l'algorithme a une influence considérable sur les résultats. L'algorithme a été construit afin de permettre la plus grande flexibilité de choix des étapes. Cependant, la suite présentée au chapitre 3 est la série qui produit les meilleurs résultats avec un compromis raisonnable en temps de calcul. Certains phénomènes émergents apparaissent tel que les phases de glouton, division, fusion et glouton compressé avec très peu de chevauchement.

L'algorithme doit effectuer les étapes précédentes à chaque fois qu'une amélioration est détectée. Cependant, un fait intéressant est qu'une fois qu'une étape est allée exhaustivement chercher son sommet local, elle est rarement réutilisée par la suite. En autres mots, les étapes sont regroupées dans certains moments de l'algorithme malgré qu'elles soient évaluées dans les étapes subséquentes. Ce phénomène persiste même lorsqu'on permute les étapes. Les figures 26 à 29 illustrent les regroupements d'étapes à travers les itérations. Une hypothèse est que les étapes sont moins corrélées que l'on pense. L'ordre des étapes ont une influence minime sur les résultats finaux.

Figure 24 : Amélioration de la modularité en fonction du temps (ms) avec les étapes indiquées pour *Electronic Circuit (s838)*

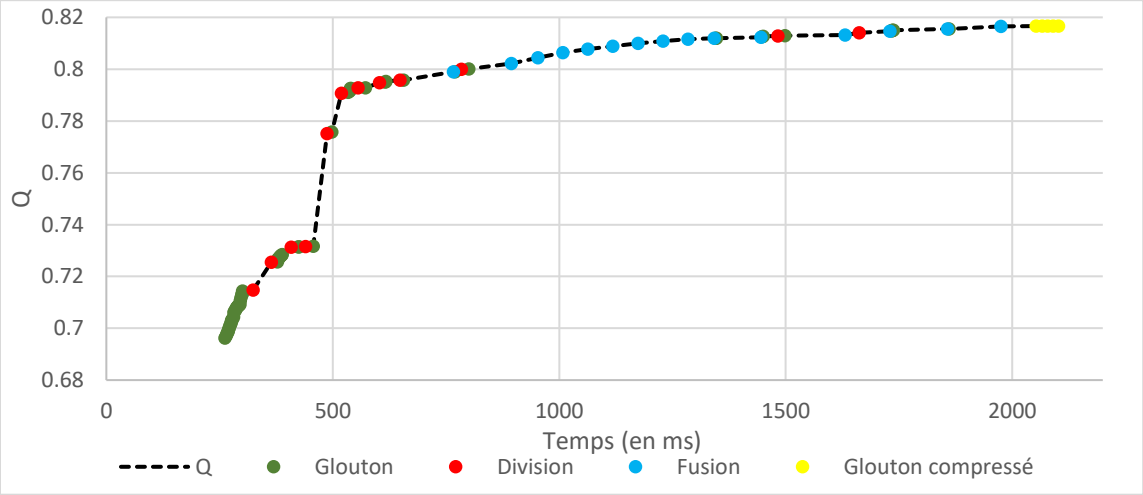


Figure 25: Amélioration de la modularité en fonction du temps (ms) avec les étapes indiquées pour *Netscience_main*

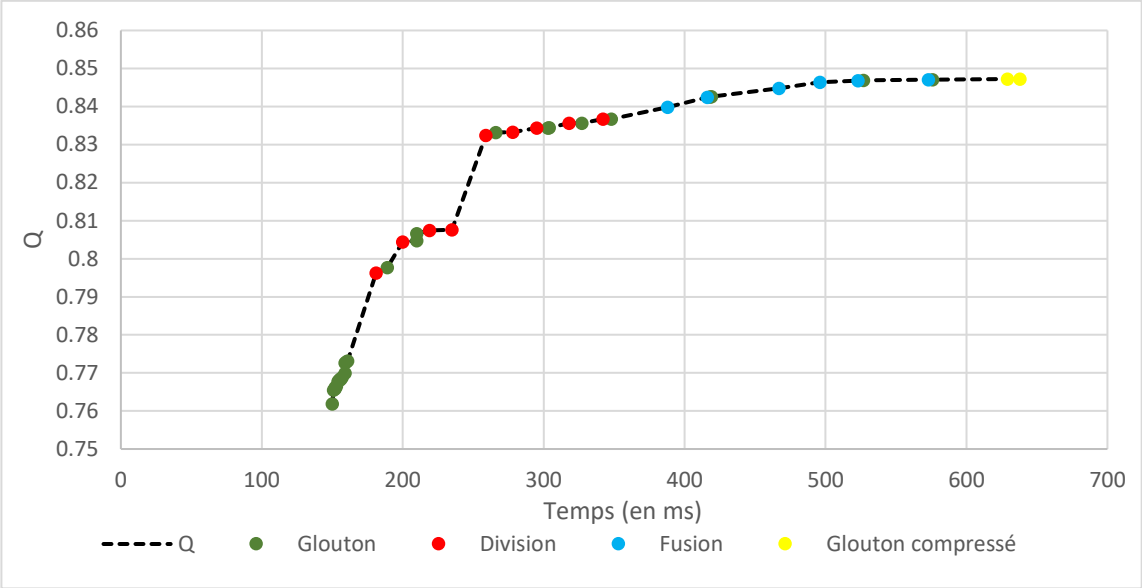


Figure 26: Amélioration de la modularité en fonction du temps (ms) avec les étapes indiquées pour *USair97*

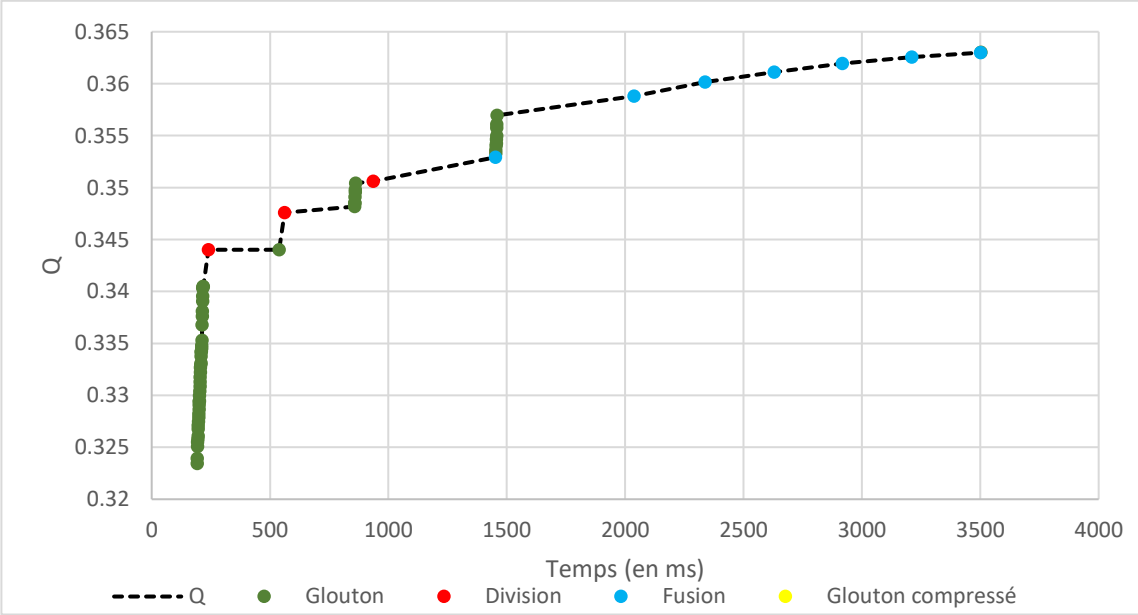
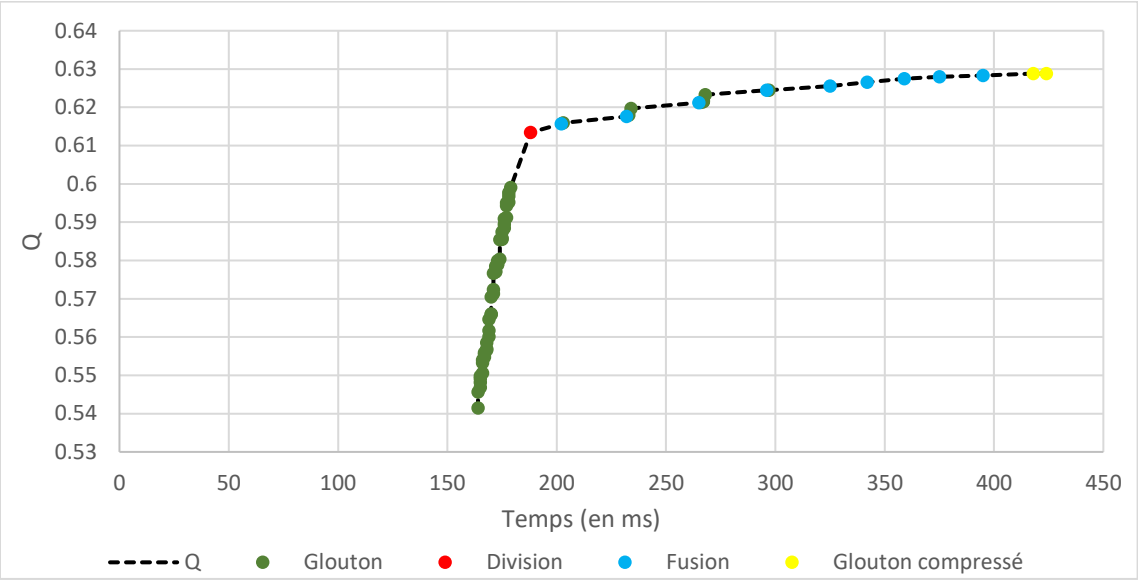


Figure 27: Amélioration de la modularité en fonction du temps (ms) avec les étapes indiquées pour *A01 Main*



Les graphiques ci-haut (fig. 26 à 29) illustrent la trajectoire de l'algorithme à travers les étapes avec un code de couleur distinct pour chaque étape. Après l'initialisation, l'algorithme va chercher à placer tous les nœuds afin d'optimiser l'ensemble des nœuds individuels dans les groupes qui maximiseront la modularité. Par la suite, les divisions de groupes vont séparer certains groupes et une autre optimisation des nœuds individuels suit pour atteindre un sommet local des nœuds individuels. Après la phase de divisions de groupes, chaque itération de fusion de groupe est précédée par les étapes de glouton et de division de groupe. À première vue, l'étape de fusion de groupe semble très longue. Cependant, cet effet est expliqué par le fait que la division de groupe doit exhaustivement conclure qu'il n'y a pas de groupe à diviser. Cette étape de vérification exhaustive de la division est la plus lourde.

L'étape de la division de groupe est l'étape la plus payante et elle est souvent la cause d'un saut considérable en modularité. Cette étape semble débloquer l'algorithme à un certain temps. Elle fournit la liberté à l'algorithme en créant de nouveaux groupes. L'étape du glouton est efficace à placer de nombreux nœuds, cependant elle a tendance à converger vers un maximum local sans être capable de sortir. Le glouton est utilisé afin

de « stabiliser » le réseau en assurant qu'aucun nœud individuel ne peut avoir une amélioration.

L'algorithme va parfois fusionner deux groupes suivis d'une division pour trouver un sous-cluster d'un des deux groupes avec un gain substantiel de modularité. La pente choisie de cette étape est toujours positive en améliorant la modularité avec un rendement décroissant. Alors, une itération est toujours améliorante.

5.5. Facteurs qui influencent la performance et la qualité

1. La trajectoire de recherche : lorsqu'on regarde plus précisément les itérations, on peut remarquer rapidement que la trajectoire empruntée par l'algorithme va avoir une grande influence sur la vitesse et la qualité de la solution. Lorsqu'on fait varier aléatoirement certaines sélections (ex : ordre de recherche des nœuds ou une permutation des étapes) les solutions finales sont influencées grandement par les premières étapes.
2. La qualité du réseau (Q optimal) a une influence considérable sur le temps de recherche. Les réseaux qui ont une haute modularité ont plusieurs trajectoires qui convergent vers la solution tandis que les réseaux à basse modularité ont des configurations de groupes qui passent par plusieurs minimums locaux dispersés. Alors, la distance entre les sommets locaux est plus dispersée lorsque la qualité du réseau est inférieure.
3. Les sous-clusters sont souvent un lien avec les sommets locaux. Par observation des configurations optimales et des solutions à chaque itération, plusieurs sommets locaux sont causés par des sous-clusters ou groupes de nœuds qui agissent comme un nœud. La majorité des algorithmes vont évaluer un nœud à la fois au lieu d'évaluer un groupe de nœuds ensemble pour établir une amélioration.
4. Les nœuds non adjacents ont une influence importante lorsque ΔQ se rapproche des valeurs de modularité des arcs non adjacents. Un algorithme devra prendre en compte les nœuds non adjacents afin d'arriver à l'optimalité. L'effet des arcs non

adjacents explique en partie la variation de solutions à haute modularité avec des configurations distinctes.

5. Le problème des cycles de groupes de nœuds : il y a certains cycles de groupes de nœuds qui sont présents dans les réseaux plus grands et complexes. Par exemple, un sous-cluster de nœuds dans un groupe va éviter qu'un autre soit déplacé de manière réciproque. Un algorithme robuste doit être capable de détecter et évaluer un déplacement de plusieurs sous-groupes dans ce cycle comme l'algorithme du « stepping-stone ».

Chapitre 6 :
Conclusion

6.1. Résumé

L'objectif de ce mémoire est l'étude d'une méthode en étapes de la maximisation de la modularité basée sur une approche matricielle. L'algorithme présenté a été construit d'une approche du bas vers le haut. En d'autres mots, l'algorithme a été construit à partir des particularités des partitions optimales des réseaux. L'algorithme est spécialisé pour maximiser la modularité des réseaux complexes classiques non directionnels et disjoints de la littérature.

La modularité est une mesure ou un critère de partition populaire, mais qui n'est pas universelle. D'autres critères existent, cependant les difficultés de la détection de communautés sont semblables à détecter des formes ou les contours de nuages afin de stipuler que le nuage ressemble à un objet. Un des défauts de la modularité est la limite de résolution de l'indicateur. Il existe des techniques pour rajouter une dimension supplémentaire pour prendre en compte cette contrainte qui n'a pas été prise en compte dans ce mémoire. Néanmoins, la modularité est l'indicateur de partitions le plus efficace à ce jour. Plusieurs algorithmes ont été élaborés dans la littérature afin de résoudre la maximisation de la modularité tout maximisant la qualité et amoindrir le coût en complexité.

L'exploration des propriétés de la matrice de modularité est impopulaire dans la littérature à ce jour pour résoudre la maximisation de la modularité à cause des limites de stockage en mémoire et la nature quadratique de la taille des matrices. De plus, très peu d'auteurs discutent des particularités des sommets locaux et des configurations internes des communautés. Un projet sur les particularités des sommets et des configurations serait une exploration intéressante dans l'avenir.

De plus, l'utilisation de la matrice de la modularité ainsi que ses propriétés illustrent un potentiel pour résoudre la maximisation de la modularité. L'algorithme présenté en quatre étapes distinctes a eu des résultats intéressants. Les objectifs individuels des quatre étapes

ne sont discutés dans la littérature, mais aucunement présentés dans une séquence semblable à celle présentée dans ce mémoire.

Dans l'algorithme présenté, il y avait 4 étapes avec des fonctions objectives distinctes qui visaient à optimiser différentes interactions des partitions de communautés. En réalité, ces étapes peuvent être remplacées par d'autres algorithmes avec potentiellement un meilleur rendement. Cependant, la démonstration d'un algorithme en étapes à explorer l'idée de résultats efficace en mettant des algorithmes en séquence. De plus, la séquence prise par l'algorithme par rapport aux étapes est importante pour réduire le temps de calcul et la qualité de la solution.

Par ailleurs, l'étude approfondie des configurations des nœuds et des itérations au niveau des nœuds illustre qu'un algorithme qui réussit à maximiser la modularité à l'optimalité dans un temps polynomial devra prendre en compte :

1. Les nœuds non adjacents lorsque ΔQ se rapproche de l'ordre de grandeur des valeurs d'arcs négatifs.
2. L'optimisation des cycles de sous-communautés de nœuds afin de détecter la valeur ajoutée de ces cycles.
3. L'étude des sommets locaux qui ont certaines configurations particulières qui pourront potentiellement être généralisées.

Les résultats de l'algorithme ainsi que sont implémentation illustre des résultats intéressants en termes de qualité. Le temps de calcul a été réduit considérablement des premiers prototypes. Cependant, la vitesse de l'algorithme est encore beaucoup trop lente pour envisager des problèmes de grande taille dans un délai raisonnable. Une approche de compression des nœuds de Blondel [3] avec une recherche locale en étape pourrait potentiellement avec un temps de calcul beaucoup plus rapide à cause de la réduction du problème.

6.2. Limites et travaux à venir

6.2.1. Les sommets locaux

Plusieurs types de sommets locaux sont rencontrés et peuvent être classifiés. Il y a un manque dans la littérature d'analyse de sommets locaux et des structures dans la détection des communautés dans une perspective de résolution. La grande majorité des algorithmes tentent une combinaison d'heuristiques et d'aléatoire avec différentes fonctions objectives pour améliorer la solution.

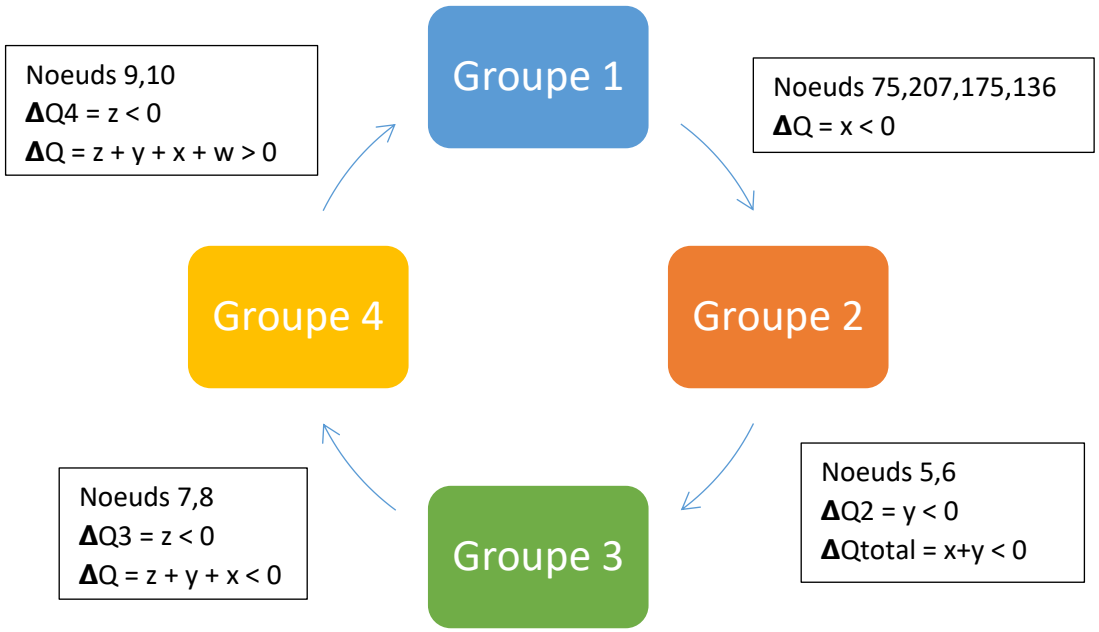
Une hypothèse est que lorsque toutes les catégories de sommets sont analysées et leur raison d'être est résolue, un algorithme pourra les contourner facilement en modifiant sa fonction objective. Une idée intéressante à expérimenter est un algorithme avec une fonction objective qui s'adapte aux sommets et réseaux en calculant un nouvel ensemble de solutions à chaque fois qu'il est piégé dans un sommet local. Par exemple, lorsqu'il a atteint un sommet local avec un nœud à la fois, il utilise des pairs de nœuds pour effectuer une amélioration. Par la suite, il peut utiliser des structures prédéfinies de nœuds pour effectuer les itérations. D'autres algorithmes ont eu de bons résultats avec différentes étapes, cependant les étapes sont nécessaires à une résolution rapide. Par expérimentation et comparaison avec les partitions optimales, le nombre de types de sommets est fini et en fonction du nombre de nœuds qui sont analysé. Tous les algorithmes pourront tirer bénéfices d'un domaine de l'analyse des sommets dans l'optimisation combinatoire. Cependant, des outils plus avancés ne sont pas encore disponibles ou répandus comme dans l'analyse continue des sommets.

6.2.2. Problème de cycle

La problématique de cycles émerge lorsque la modularité se rapproche de l'optimalité. La majorité des algorithmes se concentre sur les gains de modularité des arcs de la matrice d'adjacence. Lorsque Q est très proche de l'optimalité, les infimes valeurs de modularité des arcs négatifs vont avoir une incidence sur la répartition des nœuds dans les clusters. Ces petites valeurs de modularité négatives deviennent significatives lorsque le ΔQ est dans le même ordre de grandeur. Ces cycles s'excluent mutuellement, c'est-à-dire qu'ils

empêchent un sous-cluster de pouvoir changer de groupe et le sous-cluster l'empêche de changer de groupe également. Alors, les sous-clusters doivent être détectés et échangés entre les communautés de manière holistique comme illustrées à la figure 28.

Figure 28 : Représentation visuelle du problème de cycle



La figure 28 démontre un exemple commun de cycle rencontré. Par exemple, quatre communautés qui ont atteint un équilibre au niveau des nœuds individuels. Ces quatre communautés ont un sous-cluster qui individuellement n'a aucun gain en modularité si pivoté vers un nouveau groupe. Une amélioration différentielle est seulement obtenue par un changement simultané de ces quatre sous-clusters. Donc, l'amélioration en modularité provient de la non-adjacence de ces sous-clusters qui créent une boucle autodépendante.

Lorsque le delta *Q* est suffisamment petit, l'hypothèse est que l'algorithme devra analyser la valeur différentielle créée par les arcs négatifs ou les arcs absents qui ont une probabilité négative. Les cycles de sous-clusters apparaissent comme un problème de « stepping-stone ». L'algorithme devra détecter les sous-clusters et calculer le gain différentiel d'un cycle en prenant en compte la modularité des arcs positifs et des arcs négatifs. Cette problématique est présente lorsque la modularité se rapproche de son maximum pour un réseau. La figure 28 illustre un exemple de la problématique de cycles. La trajectoire de recherche pour améliorer la solution devra franchir une diminution (creux) de modularité pour trouver un voisinage qui améliore la modularité.

Bibliographie

1. Aloise, D., Cafieri, S., Caporossi, G., Hansen, P., Perron, S., & Liberti, L. (2010). Column generation algorithms for exact modularity maximization in networks. Physical Review E, 82(4).
2. Aloise, D., Caporossi, G., Hansen, P., Liberti, L., Perron, S., & Ruiz, M. (2013). Modularity maximization in networks by variable neighborhood search. Graph Partitioning and Graph Clustering, 113-127.
3. Blondel, V. D., Guillaume, J., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10), P10008.
4. Coscia, M., Giannotti, F., & Pedreschi, D. (2011). A classification for community discovery methods in complex networks. Statistical Analysis and Data Mining, 4(5), 512-546.
5. Fortunato, S., & Barthélemy, M. (2007). Resolution limit in community detection. Proceedings of the National Academy of Sciences of the United States of America, 104(1), 36–41.
6. Good, B. H., De Montjoye, Y., & Clauset, A. (2010). Performance of modularity maximization in practical contexts. Physical Review E, 81(4).
7. Henzinger, M., Noe, A., Schulz, C., & Strash, D. (2018). Practical minimum cut algorithms. 2018 Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments (ALENEX), 48-61.
8. Latora, V., Nicosia, V., & Russo, G. (2017). Complex networks: Principles, methods and applications. Cambridge University Press.
9. M.E.J. Newman, M. Girvan, "Finding and Evaluating Community Structure in Networks", Physical Rev. E, no. 026113, 2004.
10. Newman, M. (2006). Modularity and Community Structure in Networks. Proceedings of the National Academy of Sciences of the United States of America, 103(23), 8577-8582.
11. Thomas Pontoizeau. Community detection: computational complexity and approximation. Other [cs.OH]. PSL Research University, 2018.
12. Traag, V. A. (2015). Faster unfolding of communities: Speeding up the Louvain algorithm. Physical Review E, 92(3).
13. Vieira, V. D., Xavier, C. R., Ebecken, N. F., & Evsukoff, A. G. (2014). Performance evaluation of modularity based community detection algorithms in large scale networks. Mathematical Problems in Engineering, 2014, 1-15.
14. Wang Y., Huang H., Feng C., Liu Z. (2015) Community Detection Based on Minimum-Cut Graph Partitioning. In: Dong X., Yu X., Li J., Sun Y. (eds) Web-Age Information Management. WAIM 2015. Lecture Notes in Computer Science, vol 9098. Springer, Cham