الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Systèmes Informatiques

Thème :

Détection des communautés dans les réseaux sociaux

Encadré Par :

Mme Louafi Wafa

Présenté par :

Elaggoune Achraf

Octobre 2020

Remerciements

Je remercie en tout premier lieu, le bon Dieu tout puissant de nous avoir donné la volonté, la force et le courage de réaliser le présent travail.

Je tiens à remercier chaleureusement notre directeur de recherche, Madame Louafi Wafa, pour sa patience, sa disponibilité et son aide incontestable. Je lui adresse toute ma reconnaissance quant à son accompagnement régulier tout au long de l'élaboration de ce travail.

J'adresse mes sincères remerciements aux membres de jury pour leurs efforts de lecture et d'évaluation de ce mémoire.

Je remercie également le chef de département de l'informatique, Monsieur Kouahla Zineddine.

Je saisisse l'opportunité pour remercier vivement tous les enseignants du département de l'informatique qui ont veillé à nous accorder une formation de qualité.

Mes sincères gratitudes vont aussi à tous ceux qui ont contribué de près ou de loin à l'accomplissement de ce travail.

Merci

Dédicace

Je dédie ce modeste travail ;

A mes chers parents qui n'ont jamais cessé de formuler des prières à mon égard, de me soutenir et de m'épauler pour que je puisse atteindre mes objectifs, aucune dédicace, aucun mot ne pourrait exprimer à sa juste valeur la gratitude et l'amour que je leur porte.

A mes deux uniques frères, Zaki et Amine pour ses soutien moral, ses encouragements et ses aide précieuse.

A la mémoire de ma grand-mère maternelle, ma grand-mère paternelle, qui restera à jamais dans mon esprit et dans mon cœur.

A Meriem, pour son aide inestimable.

Table des matières

Liste des figures

Liste des Tableaux

Résumé

Les réseaux du monde réel sont complexes, de grande dimension et multiformes, et peuvent donc être interprétés de nombreuses manières différentes.

De nombreux systèmes du monde réel peuvent être modélisés comme des réseaux dans lesquels les nœuds représentent les entités et les liens représentent les relations entre ces entités.

Parmi les principaux axes d'analyse des réseaux est la détection des communautés et notamment celui qui se chevauchent qui considère comme l'un des problèmes difficiles qui prend un intérêt croissant ces dernières années parce que le chevauchement est l'une des caractéristiques des réseaux du monde réel et devrait être prise en compte pour la détection des communautés.

Certaines méthodes de détection de communautés chevauchantes ont également été proposées récemment, quelques méthodes ont donné des résultats acceptables, mais jusqu'à présent, aucun algorithme ne donne des résultats complètement corrects. Notre travail consiste à réaliser une nouvelle approche dans le terme de la détection de communautés chevauchantes dans les réseaux sociaux qui se base principalement sur le concept du voisinage et de la probabilité qui donne au début un ensemble des groupes des nœuds et qui apporte des nombreux regroupements fréquents sur les groupes initiales pour obtenir les communautés finales.

Notre travail est la suite de thèse de Mastère de l'année passée « Analyse des réseaux sociaux et détection de communautés » [1].

Mots clés :

Détection de communautés, communautés chevauchantes, Réseaux sociaux, voisinage, probabilité.

Abstract

Real-world networks are complex, high-dimensional and multifaceted, thus can be interpreted in many different ways.

Many real-world systems can be modeled as networks in which the nodes represent entities and the links represent relationships between entities.

The detection of overlapping communities is a challenging problem which is gaining increasing interest in recent years because overlapping community is one of the characteristics of real-world networks and should be considered for community detection.

Some methods of detecting overlapping communities have also been proposed recently, most of the methods have given acceptable results, but so far no algorithm gives completely correct results. Our work consists in carrying out a new approach in the term of the detection of overlapping communities in the social networks which is based mainly on the concept of the neighborhood and the probability which gives at the beginning a set of the groups of the nodes and which brings many regroupings frequent on the initial groups to obtain the final communities.

Our work is the continuation of last year's Master thesis « Analysis of social networks and detection of communities» [1].

Keywords :

Community detection, Overlapping Communities, Social Networks, Neighborhood, Probability.

ملخص

شبكات العالم الحقيقي معقدة وعالية الأبعاد ومتعددة الأوجه وبالتالي يمكن تفسيرها بعدة طرق مختلفة.

يمكن نمذجة العديد من أنظمة العالم الحقيقي كشبكات تمثل فيها الرؤوس كيانات وتمثل الروابط العلاقات بين الكيانات.

يعد اكتشاف المجتمعات المتداخلة مشكلة صعبة تكتسب اهتمامًا متزايدًا في السنوات الأخيرة لأن المجتمع المتداخل هو أحد خصائص شبكات العالم الحقيقي ويجب مراعاته عند اكتشاف المجتمعات.

تم اقتراح بعض طرق اكتشاف المجتمعات المتداخلة مؤخرًا ، وقد أعطت معظم الطرق نتائج مقبولة ، ولكن حتى الآن لا توجد خوارزمية تعطي نتائج مثالية تماما. يتمثل عملنا في تنفيذ نهج جديد في مصطلح الكشف عن المجتمعات المتداخلة في الشبكات الاجتماعية والذي يعتمد بشكل أساسي على مفهومي التقارب والاحتمالات و الذي يعطي في البداية مجموعة من مجموعات العقد والتي يطبق عليها العديد من عمليات إعادة التجميع المتكررة على المجموعات الأولية للحصول على المجتمعات النهائية.

عملنا هو استمرار لأطروحة ماستير العام الماضي «تحليل الشبكات الاجتماعية والكشف عن المجتمعات» .[1]

كلمات مفتاحية

التعرف على المجتمعات، المجتمعات المتداخلة، الشبكات الاجتماعية، التقارب، الاحتمالات

Introduction générale

Les réseaux complexes constituent un formalisme efficace pour représenter les relations entre les objets composant de nombreux systèmes du monde réel. Les réseaux sont modélisés sous forme graphiques, où les nœuds représentent des objets et les arêtes représentent les interactions entre ces objets.

L'un des principaux problèmes dans l'étude des réseaux complexes est la détection de la structure communautaire, c'est-à-dire la division d'un réseau en groupes (clusters ou modules) de nœuds ayant des intra-connexions denses, et des interconnexions clairsemées.

Les réseaux du monde réel ont généralement une structure communautaire, c'est-à-dire que les nœuds sont regroupés en communautés densément connectées. La détection des communautés est l'un des sujets de recherche les plus populaires et les mieux étudiés en science des réseaux et a attiré l'attention dans de nombreux domaines différents, notamment l'informatique, les statistiques, les sciences sociales...etc.

La détection de communauté sans chevauchement divise un réseau en plusieurs communautés où chaque nœud appartenant à une seule communauté. De nombreux algorithmes ont été proposés pour la détection de communauté sans chevauchement [2][3][4]…etc.

Dans les réseaux du monde réel, un nœud appartient souvent à plus d'une communauté. Par exemple, une personne peut appartenir à plusieurs communautés dans un réseau social comme la famille, les amis et les collègues. Ces dernières années, de nombreuses approches différentes ont été proposées en ce qui concerne la détection des communautés chevauchantes dans les réseaux [5][6][7]…etc. Le grand nombre de méthodes disponibles conduit à une question fondamentale: si une méthode proposée est capable d'identifier correctement les communautés disjointes et les communautés chevauchantes.

L’objectif principal de cette thèse est de concevoir une approche de détection de communautés chevauchantes qui serait stable, précise et efficace même pour les réseaux réels. Pour ce but, nous avons implémenté une nouvelle méthode de détection des communautés chevauchante qui se base principalement sur le principe de voisinage qui était déjà proposé l’année passée [1] mais elle n’est pas implémenter en ajoutant quelque modification et faisons quelques corrections.

L’algorithme fonctionne sur trois phases principales; la première phase compose de deux étapes : l’initialisation et l’arrangement. Cette phase consiste à initialiser les classes pour chaque nœud du réseau, pour cela nous avons affecté chaque nœud à une classe avec tous ses voisins. Dans la deuxième phase nous proposons un ensemble de regroupement qui va regrouper les groupes les plus similaires de la première phase. Finalement, la dernière phase où nous avons calculé la probabilité d’appartenance pour chaque nœud dans les communautés pour extraire la structure communautaire finale.

L’approche proposée est évaluée sur différents types de réseaux et leur performance est comparée à celle d'autres algorithmes de détection de communautés.

Le mémoire se compose de trois chapitres, il est organisé de la manière suivante :

- **Le premier chapitre :** l'état de l'art
  Dans ce chapitre, nous donnerons quelques définitions et la terminologie principale dans la détection de communautés avec chevauchement. Une description des propositions les plus récentes pour la détection de communautés chevauchantes est donnée, et une classification en différentes catégories est fournie.
- **Le deuxième chapitre :** Conception du système
  Dons ce chapitre, nous présentons un schéma générale de notre approche avec une description générale de notre algorithme tout en donnons des exemples illustratif pour bien défini la démarche de notre approche.

- **Le troisième chapitre :** Implémentation du système
  Dans ce chapitre, nous allons montrer l'implémentation de notre système on montrant les environnements de développement, les fonctionnalités du système, la présentation du système, les tests que nous allons fait tout en finirons le chapitre avec une comparaison de nos résultats avec d'autre algorithme.

Nous terminerons notre le mémoire avec une conclusion générale et quelques perspectives.

Chapitre 01 : La détection de communautés chevauchante

1. Introduction

L'analyse des réseaux sociaux est un domaine répandu qui attire l'attention de nombreux experts en Data Mining. Dans ce contexte, la communauté est l'une des propriétés les plus courantes et les plus importantes pour révéler les structures cachées d'un réseau. Ainsi, la détection communautaire devient l'un des problèmes les plus cruciaux dans ce domaine. La détection de communauté est devenue de plus en plus populaire après le grand Apparaissant des réseaux sociaux comme Facebook, Google+, Tweeter…etc.

Comme le scénario du monde réel est dynamique et évolutif, le réseau social est utilisé de manière intensive dans une large gamme d'applications et représenté par un graphe avec des nœuds et des arêtes. Les nœuds représentent les utilisateurs, acteurs, ou les éléments tandis que les arêtes représentent les relations entre les utilisateurs. Détecter les communautés implique de trouver les nœuds fortement connectés. La détection de communautés chevauchantes est possibles si un nœud est membre de plusieurs communautés[8].

La détection de communauté peut aider à découvrir les structures communautaires dans les réseaux complexes, et à une signification théorique importante et de larges perspectives application, donc de plus en plus d'attention y a été accordée. La détection de communauté est divisée en détection de communauté sans chevauchement et détection de communauté avec chevauchement, mais dans les structures communautaires des réseaux la détection de communauté qui se chevauchant est plus réaliste.

Dans ce chapitre, nous présentons l'état de l'art se rapportant à la détection de communautés avec chevauchement, on va tout d'abord expliquer ce qu'une communauté et comment là représenter avant d'expliquer brièvement ce qu'une détection de communauté avec chevauchement. Après on va retenir les travaux connexes les plus connus dans le domaine de détection des communautés chevauchante.

2. Représentation des communautés

Dans l'analyse des réseaux sociaux, les graphes ont été largement utilisés comme moyens de représenter formellement les individus par des nœuds et les relations entre eux avec des liens.

2.1 Les nœud

Aussi appelé individu ou acteur c'est le sommet d'un graphe, il peut avoir des relations avec d'autres individus, Au niveau de la figure 1.1 A, B, C sont des nœuds (acteurs)[9].

Figure 1.1 Représentation des nœuds.

2.2 Lien

C'est une relation particulière, bien spécifiée entre deux acteurs, les liens peuvent être Non-dirigés lorsque la relation signifie la même chose aux deux acteurs ou bien dirigés dans le cas contraire[9].

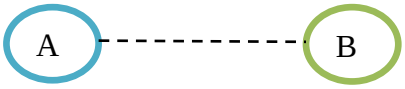

Figure 1.2 Représentation d'un lien entre deux acteurs.

3. Définition d'une communauté

D'après S. Fortunato: Une communauté est un ensemble d'entités ayant beaucoup d'interactions entre elles et peu d'interactions avec l'extérieur[10].

Girvan et Newman définissent une communauté comme un ensemble d'entités qui ont des relations internes plus que des relation externes[11].

Radicchi et al. Améliorent cette définition par la contrainte que chaque individu d'une communauté a plus de voisins à l'intérieur de sa communauté qu'à l'extérieur. Ils appellent ces structures les communautés fortes[12].

Généralement, une communauté est un ensemble d'individu qui ont beaucoup de relation entre eux à l'intérieure et moins de relation à l'extérieure.

La figure 1.3 ci-dessus montre une représentation graphique simple avec trois communautés, où chaque communauté est délimitée par des cercles en pointillés.

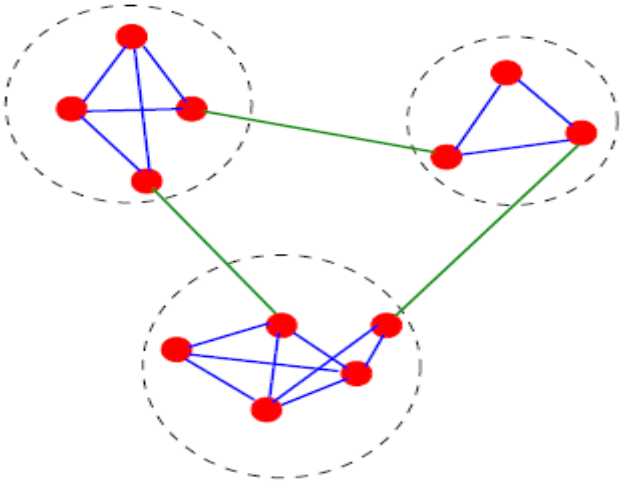

Figure 1.3 Représentation simple de trois communautés.

4. Détection de communauté avec chevauchement

4.1 Définition

Dans le monde réel, une personne appartient à plusieurs cercles sociaux d'un réseau social, tels que le cercle d'amis, le cercle familial et le cercle de collègues. Dans le réseau, les nœuds appartenant à plusieurs communautés sont appelés nœuds chevauchants[13]. La figure 1.4 nous présente la division d'un graph avec 6 nœuds on deux communautés chevauchante où on trouve que le nœud 4 est un nœud chevauchent qui appartient dans les deux communautés contrairement aux autres nœuds qui n'appartiennent dans une seule communauté.

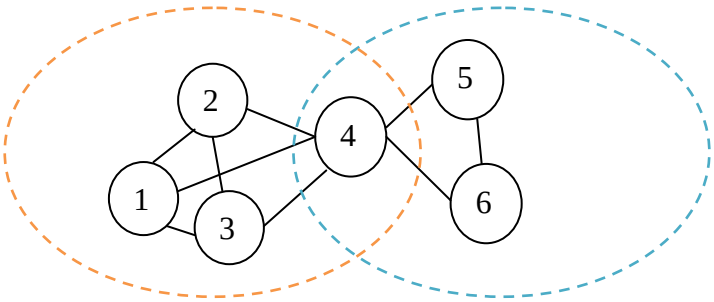

Figure 1.4 Structure d'une communauté chevauchante.

4.2 Méthodes

Dans cette section, on va décrites les méthodes de la détection des communautés avec chevauchement. Ils ont été classés en six catégories différentes sur la base de la méthodologie utilisée pour identifier les communautés. Les catégories sont les suivantes[14]:

1. Graines de nœuds et expansion locale
2. Expansion de clique
3. Groupement de liens
4. Propagation des étiquettes
5. Réseaux dynamiques
6. D'autres approches

Pour chaque catégorie, une brève description des principales caractéristiques communes aux algorithmes de cette classe est fournie.

4.2.1 Graines de nœuds et expansion locale

4.2.1.1 Définition

L'idée sous-jacente à ces approches est qu'à partir d'un nœud ou d'un petit ensemble de nœuds, une communauté peut être obtenue en ajoutant des nœuds voisins qui améliorent une fonction de qualité. La fonction qualité caractérise la structure du cluster obtenu [14].

4.2.1.2 Algorithmes

Baumes et al. [15] ont introduit le concept de fonction de densité et défini une communauté comme un sous-graphe localement optimal par rapport à cette fonction de densité. Les auteurs ont défini trois métriques pour attribuer un poids pour le graphe : La probabilité de lien interne, le rapport de lien et le rapport d'intensité. Ces métriques mesurent l'intensité de la communication au sein des clusters et peuvent être mises à jour efficacement lorsqu'un nouveau nœud est ajouté ou supprimé du cluster.

Baumes,Golberd et Magdon-Ismail [15] ont introduit un autre algorithme qui s'appelle la suppression de rang (RaRe), suppose qu'il existe des nœuds de rang élevé qu'ils sont supprimés du graphe, déconnectent le graphique en composants connectés plus petits, appelés noyaux. Les nœuds supprimés sont ensuite ajoutés à un ou plusieurs cœurs. Cela signifie que le chevauchement entre deux clusters n'est possible que par ces sommets.

Pour valider les méthodes, la distance de Hamming entre chaque cluster du vrai cluster et le cluster obtenu est calculée, puis la moyenne de toutes ces distances est considérée

Le choix d'une arête aléatoire peut influencer négativement le résultat du SI. Ainsi, les mêmes auteurs ont modifié leur méthode et ont proposé un algorithme plus efficace pour trouver des communautés chevauchante nommé IS², qui combine IS et RaRe. IS2 s'appuie également sur une nouvelle stratégie d'initialisation des clusters d'amorçage capable de calculer le classement de chaque nœud une seule fois.

La figure 1.4 ci-dessus représentent une comparaison entre les clusters trouvés dans une section d'un graphe web sémantique de 85 nœuds et 126 arêtes [15].

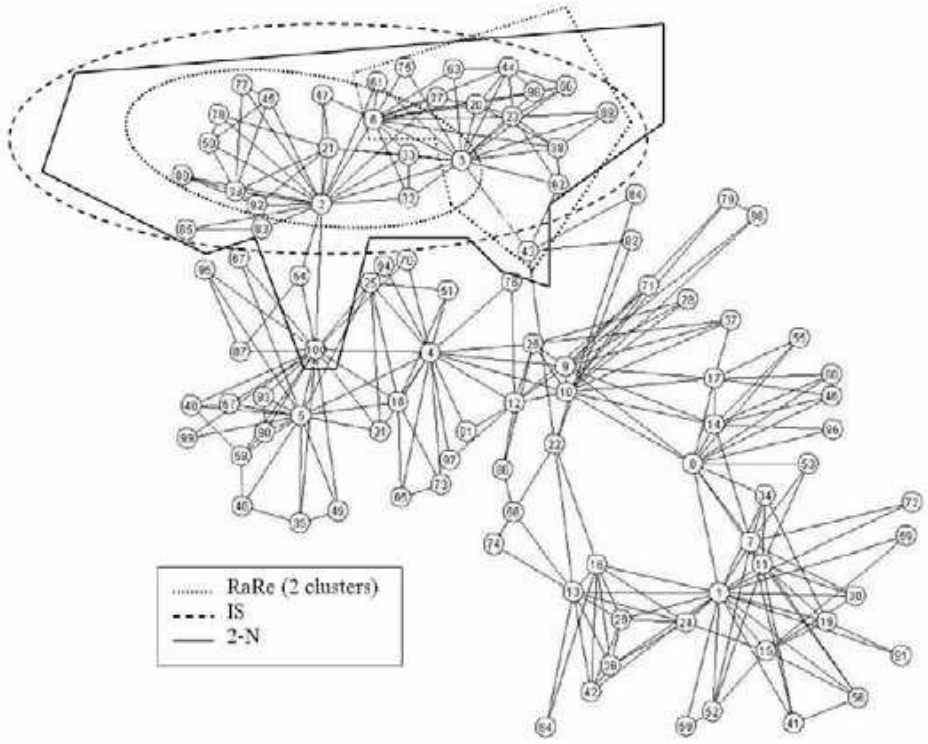

Figure 1.5 Structure d'une communauté chevauchante.

4.2.2 Expansion de clique

4.2.2.1 Définition

Les méthodes d'expansion des cliques sont similaires aux approches décrites dans la section précédente. Cependant, ils considèrent comme des noyaux germes des ensembles de nœuds hautement connectés constitués par des cliques, puis génèrent des communautés qui se chevauchent en fusionnant ces cliques, en appliquant des critères différents[14].

4.2.2.2 Algorithmes

S.Jabbour [16] a proposé une méthode efficace de détection des communautés avec chevauchement en utilisant une approche d'expansion de clique. En particulier, il a fait un usage original d'un concept particulier de théorie des graphes, appelé graphe en accords, pour découvrir des structures densément connectées dans des interactions sociales basées sur des cliques maximales. En effet, un graphe triangulé possède un certain nombre de propriétés intéressantes et utiles qui peuvent l'aider à récupérer efficacement toutes les cliques maximales d'un graphe donné. Ensuite, il développe de nouvelles stratégies d'ensemencement basées sur différentes fonctions de fitness pour découvrir des communautés significatives.

Z.Sun etB.Wang [17] ont proposé un nouveau modèle appelé OCDID (Détection de communautés chevauchante basé sur l'information dynamique) pour découvrir les communautés qui se chevauchent, qui traite le réseau comme un système dynamique permettant à un individu de communiquer et de partager des informations avec ses voisins. Le flux d'informations dans le réseau est contrôlé par la structure topologique sous-jacente (par exemple, la structure de la communauté), et la structure de la communauté est également reflétée par la dynamique de l'information. Les nœuds qui se chevauchent agissent comme des ponts entre plusieurs communautés et les informations provenant de plusieurs communautés transitent par ces nœuds. Ainsi, les nœuds qui se chevauchent peuvent être identifiés en analysant le flux d'informations entre les communautés. De plus, ils ont utilisé le théorème de convergence monotone pour confirmer la convergence de son modèle.

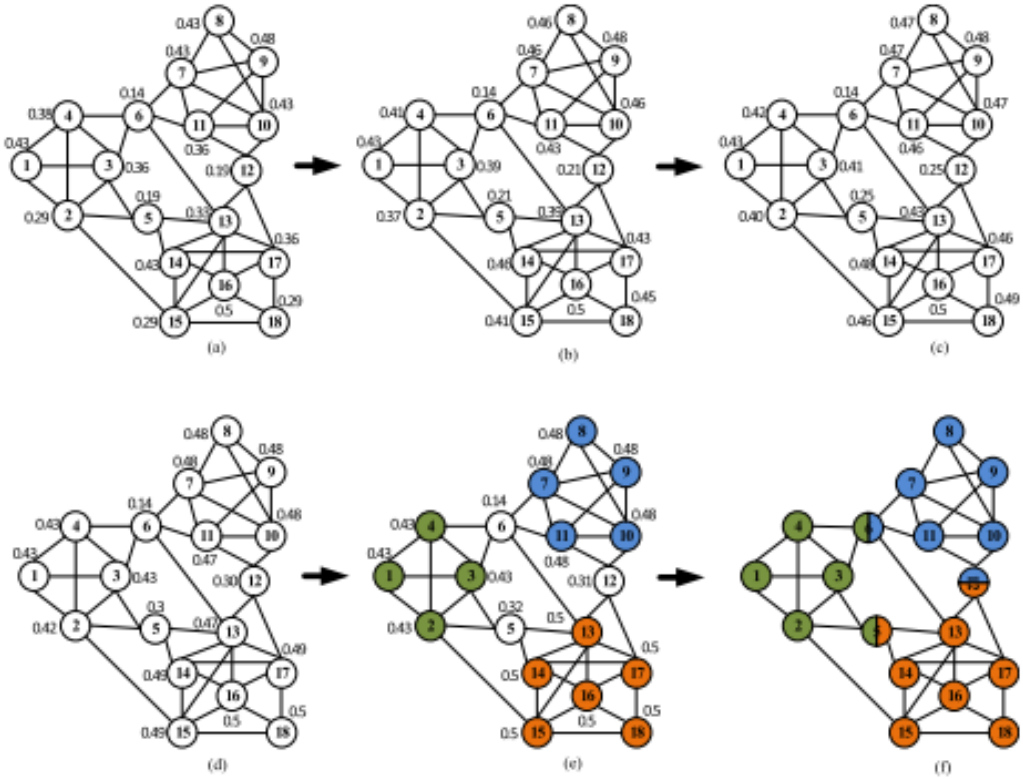

Figure 1.6 illustration de l'information dynamique [17]**.**

4.2.3 Groupement de lien

4.2.3.1 Définition

La méthode de regroupement de liens pour la détection de communautés qui se chevauchent a attiré beaucoup d'attention dans le domaine des applications de réseaux sociaux. Cependant, cela peut entraîner le regroupement avec un chevauchement excessif et un bord de pont et de bordure de cluster par erreur avec les communautés adjacentes. Ces méthodes ont proposé pour détecter les communautés qui se chevauchent en partitionnant l'ensemble des liens plutôt que l'ensemble des nœuds [18].

4.2.3.2 Algorithmes

Evans et Lambiotte [19] ont soutenu que tout algorithme qui partitionne un réseau peut être appliqué aux liens de partition pour découvrir la structure de communauté qui se chevauchent. Ils ont d'abord examiné une définition de la modularité qui utilise les propriétés statistiques des processus dynamiques se déroulant sur les bords d'un graphe, puis ont proposé un partitionnement de liens en appliquant le nouveau concept de modularité. En particulier, la modularité traditionnelle Q est définie en termes de marcheur aléatoire se déplaçant sur les liaisons du réseau. Un tel marcheur serait donc situé sur les liaisons au lieu des nœuds à chaque instant t, et ses déplacements se font entre des bords adjacents, c'est-à-dire des liaisons ayant un nœud en commun. Trois fonctions de qualité de partitionnement des liaisons d'un

réseau G ont été proposées. Chacun formalise un processus dynamique différent et explore la structure du graphe original G d'une manière différente.

Dans le premier processus dynamique, la marche aléatoire Link-Link, le marcheur saute sur l'un des bords adjacents avec une probabilité égale. Dans le deuxième processus, marche aléatoire Link-Node-Link, le marcheur se déplace d'abord vers un nœud voisin avec une probabilité égale, puis saute vers un nouveau lien, choisi avec une probabilité égale à partir de ces nouveaux bords incidents au nœud. Dans le dernier processus, les dynamiques sont conduites par la marche aléatoire originale mais sont projetées sur les liens du réseau. Les stabilités des trois processus ont été définies en généralisant le concept de modularité à des chemins de longueur arbitraire, afin d'ajuster la résolution des partitions optimales. Les partitions optimales de ces fonctions de qualité peuvent être découvertes en appliquant des algorithmes d'optimisation de modularité standard aux graphes linéaires correspondants.

Gabardo, Berretta et Moscato [20] ont proposé M-Link, un algorithme mimétique pour la détection de communautés avec chevauchement. Il maximise une fonction objective appelée densité de partition de lien. Les communautés d'arêtes obtenues avec cette méthode se traduisent naturellement par des communautés de nœuds qui se chevauchent. La méthode est basée sur une expansion locale et un mécanisme de recherche local spécialisé. Les méthodes de propagation d'étiquettes sont utilisées pour initialiser une structure de population d'arbres tertiaires multi-agents. Nous utilisons les informations mutuelles normalisées pour évaluer la similitude entre la structure de communauté connue dans un ensemble de réseaux de référence et la structure de communauté détectée par M-Link.

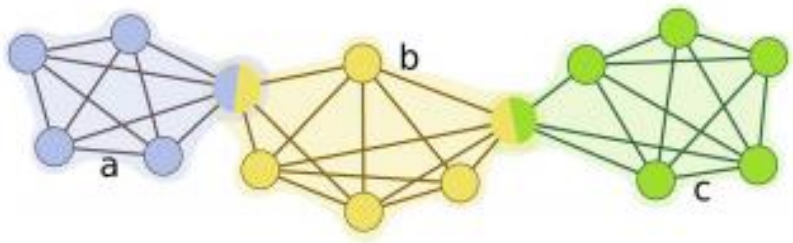

Figure 1.7 Un réseau divisé en trois communautés de liaison a, b et c avec deux nœuds qui se chevauchent [20]**.**

4.2.4 Propagation des étiquettes

4.2.4.1 Définition

Label Propagation Algorithm (LPA) [21] a été proposé pour la première fois par Raghavan et al., Qui a une complexité temporelle linéaire et s'est avéré capable de détecter rapidement et efficacement les structures communautaires dans des réseaux complexes à grande échelle. Cependant, LPA présente quelques lacunes:

1) il ne peut détecter que les communautés qui ne se chevauchent pas;

2) il souffre d'une faible précision dans de nombreux réseaux en raison du fait que l'importance de tous les nœuds voisins est la même lors de la propagation des étiquettes;

3) le caractère aléatoire de la propagation des étiquettes entraîne une instabilité de la détection de la communauté;

4) le caractère aléatoire de la propagation des étiquettes produit également le problème de la «diffusion des étiquettes».

4.2.4.2 Algorithmes

Meilian et Zhang [22] ont proposé un algorithme amélioré de détection de communauté qui se chevauchent, LPANNI (algorithme de propagation d'étiquette avec influence de nœud voisin), qui détecte les structures de communauté qui se chevauchent en adoptant une séquence de propagation d'étiquette fixe basée sur l'ordre croissant d'importance des nœuds et une stratégie de mise à jour d'étiquettes basée sur l'influence des nœuds voisins et l'étiquette historique préférée stratégie. Des résultats expérimentaux étendus à la fois sur des réseaux réels et des réseaux synthétiques montrent que le LPANNI peut améliorer considérablement la précision et la stabilité des algorithmes de détection communautaire basés sur la propagation d'étiquettes dans des réseaux complexes à grande échelle.

LPANNI se compose de deux phases. Dans la première phase, il calcule le NI (l'importance du nœud) de tous les nœuds et trie les nœuds par ordre croissant selon NI. Dans la deuxième phase, il propage les étiquettes en considérant à la fois la séquence de NI et NNI (influence du nœud voisin) jusqu'à ce que l'algorithme converge, puis la structure de communauté qui se chevauche est détectée. La condition de convergence courante est que toutes les étiquettes de nœuds ne changent plus. Cependant, cette condition peut ne pas être remplie dans certains réseaux et l'algorithme ne peut pas se terminer même après des centaines d'itérations. Par conséquent, la condition de convergence utilisée dans LPANNI est que la taille de l'ensemble d'étiquettes et les étiquettes dominantes de tous les nœuds sont stables, ou qu'un nombre d'itérations maximum spécifié est atteint.

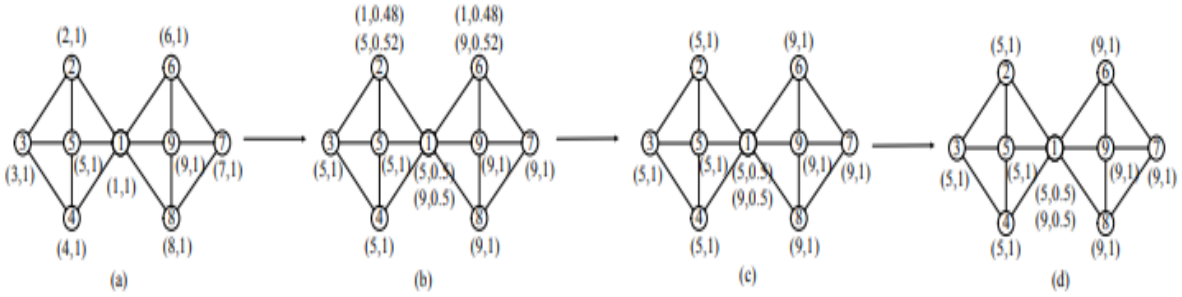

Figure 1.8 Processus de propagation d'étiquette de LPANNI sur un exemple de réseau (par ordre croissant de NI 2→3→ 4 →6→ 7→ 8→ 5→ 9→ 1) [22]**.**

Sun [23] a proposé un algorithme, appelé algorithme de propagation d'étiquettes basé sur les liens (LinkLPA), pour détecter les communautés qui se chevauchent. Parce que la partition de lien est conceptuellement naturelle pour le problème de détection de communauté qui se chevauchent, LinkLPA transforme d'abord le problème de partition de nœud en problème de partition de lien et utilise un nouvel algorithme de propagation d'étiquette avec une préférence

sur les liens plutôt que sur les nœuds pour détecter les communautés grâce à la simplicité et à l'efficacité de l'algorithme de propagation d'étiquettes. Ensuite, le LinkLPA proposé effectue un post-traitement pour affiner les communautés qui se chevauchent détectées en évitant le chevauchement et la partition incorrecte des liens faibles.

LinkLPA est composé de deux phase, la première phase est un Algorithme de propagation d'étiquettes avec préférence sur les liens, la deuxième phase c'est le Post-traitement ; Après la partition de lien, le nœud a tendance à être détecté comme chevauchant si ses liens d'incident appartiennent à des communautés différentes. Pour éviter les sur-chevauchements et mieux détecter les structures communautaires qui se chevauchent, nous adoptons une procédure de post-traitement en deux étapes: la gestion des chevauchements et la fusion de clusters similaires.

4.2.5 Réseaux dynamiques

4.2.5.1 Définition

Les méthodes décrites jusqu'à présent ne prennent pas en compte un aspect important caractérisant les réseaux: c'est-à-dire l'évolution qu'ils traversent dans le temps. La représentation de nombreux systèmes complexes à travers un graphe statique, même lorsque la dimension temporelle décrivant les interconnexions variables entre les nœuds est disponible, ne permet pas d'étudier la dynamique du réseau et les changements qu'il subit dans le temps. Les réseaux dynamiques, au contraire, capturent les modifications des interconnexions au fil du temps, permettant de retracer les changements de structure du réseau à différents pas de temps. L'analyse des réseaux et de leur évolution suscite depuis peu un intérêt croissant de la part des chercheurs [14].

Cependant, peu de propositions ont été faites pour la détection des communautés avec chevauchement dans les réseaux sociaux dynamiques. Dans ce qui suit, les méthodes les plus récentes visant à rechercher des communautés dynamiques sont décrites.

4.2.5.2 Algorithmes

Palla et al. [5] ont été parmi les premiers chercheurs à introduire une approche permettant d'analyser la dépendance temporelle des communautés qui se chevauchent à grande échelle et à ce titre, de découvrir les relations de base caractérisant l'évolution des communautés. En fait, ils ont fait valoir qu'à chaque étape de temps, les communautés peuvent être extraites en utilisant la méthode de percolation clique (CPM) [5]. Les événements qui caractérisent la durée de vie d'une communauté sont la croissance ou la contraction, à chaque pas de temps une nouvelle communauté peut apparaître, tandis que d'autres peuvent disparaître. De plus, les groupes peuvent fusionner ou se diviser. Afin d'identifier l'évolution de la communauté au cours du temps, les auteurs ont proposé de fusionner des réseaux de deux pas de temps consécutifs t et t+1, puis d'appliquer la méthode CPM pour extraire la nouvelle structure communautaire du réseau commun. Puisque le graphe joint contient l'union des liens des deux graphes, toute communauté de l'étape de temps t à t+1 ne peut que croître, fusionner ou rester inchangée, plus d'une communauté peut être fusionnée en une seule communauté, mais

aucune communauté ne peut perdre des membres. Par conséquent, toute communauté de l'un des réseaux d'origine peut être contenue dans exactement une communauté du réseau joint. Les communautés dans le graphique conjoint fournissent un moyen de faire correspondre les communautés entre les pas de temps t et t+1. Si une communauté dans le graphe conjoint contient une seule communauté de t et une seule communauté de t+1, alors elles sont appariées. Si le groupe conjoint contient plus d'une communauté de l'un ou l'autre des pas de temps, les communautés sont mises en correspondance dans l'ordre décroissant de leur chevauchement de nœuds relatif. Le chevauchement est calculé pour chaque paire de communautés à partir des deux pas de temps sous forme de fraction du nombre de nœuds communs à la somme du nombre de nœuds des deux communautés. Des expériences sur deux réseaux réels ont montré que de grands groupes restent vivants s'ils subissent des changements dynamiques. Au contraire, les petits groupes survivent s'ils sont stables.

Xiaoke et Di. [24] proposent deux cadres de factorisation matricielle évolutive non négative (ENMF) pour la détection de communautés dynamiques. Pour aborder la relation théorique entre les algorithmes de clustering évolutif, ils ont prouvé d'abord la relation d'équivalence entre ENMF et l'optimisation de la densité de modularité évolutive. Ensuite, ils ont étendu la théorie en prouvant l'équivalence entre le clustering spectral évolutif et ENMF, qui sert de fondement théorique aux algorithmes hybrides. Sur la base de l'équivalence, ils ont proposé un ENMF semi-supervisé (sE-NMF) en incorporant des informations a priori dans l'ENMF. Contrairement aux algorithmes semi-supervisés traditionnels, les informations a priori sont intégrées dans la fonction objective de l'algorithme. Le principal avantage de l'algorithme proposé est d'échapper à la solution optimale locale sans augmenter la complexité temporelle.

4.2.6 D'autres approches

La détection de communauté chevauchante de réseaux complexes a attiré de plus en plus l'attention des chercheurs. Les travaux représentatifs existants incluent principalement le clustering hiérarchique, basé sur le réseau et basé sur le nœud.

4.2.6.1 Algorithme de détection de communauté chevauchante basé sur les nœuds

L'idée de base des nœuds est que les nœuds ayant des caractéristiques similaires sont plus susceptibles d'être dans la même communauté. La méthode la plus courante est la méthode de percolation de cluster (CPM) [5]. Il peut trouver des communautés qui se chevauchent en recherchant des sous-graphiques complets maximum dans le réseau. Cependant, il a une limitation de complexité élevée.

4.2.6.2 Algorithme de détection de communauté chevauchante basé sur le réseau

Les méthodes basées sur le réseau sont basées sur un graphe de réseau social. Les méthodes existantes peuvent être divisées

1. Méthode de segmentation de graphique
2. Méthode d'extension locale
3. Méthode d'apprentissage en profondeur.

Méthode de segmentation de graphe :

Généralement, la plupart des méthodes de segmentation de graphe tentent de diviser le graphe en deux sous-graphes, puis d'obtenir le nombre requis de sous-graphes via des itérations. Les algorithmes classiques incluent l'algorithme kernighanlin (KL) [25] et l'algorithme de clustering spectral (SC) [26]. Cependant, ils doivent tous deux connaître le nombre de nœuds contenus dans la communauté et le nombre de communautés. Visant les problèmes, H.V Lierde [27] a étendu le concept de coupe normalisée et introduit les informations préalables de la communauté à laquelle appartient le nœud.

Méthode d'extension locale

L'algorithme LFM [28] était un algorithme de détection communautaire typique basé sur l'idée d'expansion locale. Bien que l'algorithme LFM puisse trouver des communautés qui se chevauchent et des relations hiérarchiques entre les communautés, il a encore des problèmes tels que des résultats instables et une dérive communautaire. Pour résoudre ces problèmes, C.Lee [29] a proposé l'algorithme GCE, qui recherchait des sous-groupes maximaux en tant que nœuds germes. Cependant, le coût de la recherche de sous-groupes maximaux dans des réseaux complexes était très élevé.

Méthode d'apprentissage en profondeur

Avec le développement de l'apprentissage en profondeur, les fonctionnalités du réseau d'apprentissage des réseaux neuronaux ont injecté une nouvelle vitalité dans la détection de communautés qui se chevauchent. B Perozzi [30] a proposé une méthode en ligne pour mapper les nœuds du réseau vers l'espace caché. LINE a été proposée par J Tang [31] pour cartographier de grands réseaux dans un espace vectoriel de faible dimension. La méthode était adaptée à la gestion de tout type de réseau, y compris les réseaux non dirigés, dirigés, impuissants et pondérés.

4.2.6.3 Algorithme de détection de communauté chevauchante basé sur le clustering hiérarchique

Le but du clustering hiérarchique est de construire une hiérarchie communautaire, qui peut analyser le réseau à différents niveaux. Les méthodes représentatives sont[32] :

1. Le clustering hiérarchique de division
2. Le clustering hiérarchique agglomérat.

Clustering hiérarchique de division

L'algorithme de clustering hiérarchique qui divise est une segmentation de la communauté de haut en bas. Généralement, cette idée considère d'abord le réseau comme une communauté, puis divise la communauté en plusieurs petites communautés via une segmentation hiérarchique telle que l'algorithme GN [11]. L'algorithme GN est l'une des méthodes les plus classiques, qui utilise des médiateurs d'arête. Étant donné que l'arête avec des médiateurs d'arête élevée sera supprimée à chaque itération, la vitesse est beaucoup plus rapide que la

vitesse de suppression aléatoire des arêtes. Cependant, il présente une complexité et une limitation temporelles élevées dans les réseaux à grande échelle.

clustering hiérarchique agglomérat

Newman [3] a proposé la définition de la modularité, puis a proposé un nouvel algorithme de détection de communauté basé sur le glouton. Les algorithmes basés sur l'optimisation de la modularité ont été largement étudiés. Cependant, il remet en question des résultats instables et ignore les relations entre les nœuds qui ne sont pas directement connectés

5. Ensembles de données et évaluations

5.1 Ensembles de données classiques

Les bases des données publiques pour la détection de la communauté sont présentées dans le tableau I. Le réseau de karaté [33] a été créé en observant les relations entre les membres des clubs de karaté dans les universités américaines. Les nœuds du réseau représentaient les membres du club de karaté, et les arêtes représentaient la connexion et l'interaction entre les membres. Le réseau Dolphins [34] a été obtenu par les auteurs. Et ils ont observé le mode de vie des dauphins qui vivent depuis longtemps dans les fjords magiques de la Nouvelle-Zélande. Dans le réseau Dolphins, chaque nœud représentait des Dolphins et chaque arête représentait des contacts fréquents entre Dolphins. Dans le Football Network [11], chaque nœud représentait une équipe, tandis que chaque arête représentait la relation entre deux équipes. Le réseau de produits Amazon et réseau social Youtube proviennent de la base de données SNAP. Dans le réseau de produits Amazon, les nœuds représentaient les produits de base et les arêtes représentaient deux produits généralement achetés par les clients en même temps. Dans le réseau social Youtube, les nœuds représentaient les utilisateurs et les arêtes représentaient l'interaction entre les utilisateurs. Le réseau Books [35] a été construit en vendant des livres politiques sur Amazon. Chaque nœud représentait les livres à vendre, tandis que les arêtes indiquaient qu'un grand nombre de clients ont acheté les deux livres en même temps.

BDD	Nœud	Arêtes	Nombre de communauté
Karate	34	78	2
Daulphins	62	159	2
Football	115	613	12
Books	105	441	3
Facebook	4039	88234	10

Tableau 1.1 : ensemble des bases de données publique [32]

5.2 Evaluation

Il est très important d'évaluer la qualité de la structure des communautés chevauchantes. La capacité d'un algorithme à détecter la structure d'une communauté est généralement validée en testant l'algorithme sur des réseaux artificiels ou réels pour lesquels la division en communautés est connue. Étant donné que la disponibilité d'une structure de communauté de vérité terrain pour les grands réseaux réels est plutôt difficile, des repères synthétiques construits en spécifiant des paramètres pour caractériser la structure du réseau sont préférés[14].

Dans cette section, certaines méthodes d'évaluation courantes sont présentées, telles que Normaliser les informations mutuelles (NMI), modularité de chevauchement et la mesure F1. En général, dans les évaluations de communautés qui se chevauchent, quand on connaît la partition réelle du réseau, la NMI est généralement appliquée pour évaluer l'exactitude de la détection de la communauté. Lorsqu'il est inconnu sur la partition réseau réelle, la modularité de chevauchement est appliquée pour évaluer. Dans l'évaluation des nœuds qui se chevauchent, les chercheurs effectuent généralement une mesure F1 comme méthode d'évaluation[32]. Les détails de ces méthodes sont les suivants :

L'information mutuelle normalisée (NMI) est appliquée pour quantifier le degré de similitude de deux communautés. Plus la valeur NMI est élevée, plus le degré de similitude entre les résultats de la partition de communauté et les résultats du monde réel est élevé [36].

La modularité de chevauchement introduit la formule optimale du coefficient d'appartenance. Plus la valeur est proche de 1, plus la structure communautaire divisée par le réseau est forte et meilleure [32].

Mesure F1: la mesure F1 mesure la précision de l'extraction des nœuds qui se chevauchent. C'est la moyenne harmonique de la précision et du rappel [37].

6. Travaux connexes

Comme nous avons dit précédemment, La détection de communautés qui se chevauchent est un problème difficile qui suscite un intérêt croissant ces dernières années en raison de l'attitude naturelle des individus, observés dans des réseaux du monde réel, pour participer à plusieurs groupes en même temps. Pour ce contexte, les chercheurs ont fourni beaucoup d'algorithmes qu'on a déjà vus, on va présenter encore quelques travaux connexes à notre travail.

L'algorithme de propagation de chevauchement des communautés (COPRA) [38] Les sommets ont des étiquettes qui se propagent entre leurs voisins de sorte que les membres d'une communauté parviennent à un consensus sur leur appartenance à la communauté. Par la suite, les coefficients d'appartenance des nœuds sont mis à jour de manière itérative en calculant la moyenne des coefficients des nœuds de leur voisinage.

Un algorithme appelé Speaker-listener Label Propagation Algorithm (SLPA) [39] présente un nouveau cadre général permettant de détecter et d'analyser à la fois les nœuds individuels qui

se chevauchent et des communautés entières, les nœuds échangent des étiquettes selon des règles d'interaction dynamiques. Ce processus est répété plusieurs fois pour calculer l'appartenance à la communauté de chaque nœud.

En 2019, S .Gupta et P.Kumar [40] ont proposé Un algorithme de détection de communauté qui se chevauchent basé sur des informations granulaires de liens et des concepts de théorie des ensembles approximatifs. Premièrement, des liens de voisinage autour de chaque paire de nœuds sont utilisés pour former des sous-ensembles de liens initiaux. Par la suite, l'approximation supérieure de liaison contrainte des sous-ensembles de liaison est calculée de manière itérative jusqu'à convergence. Les sous-ensembles d'approximations supérieurs obtenus lors de chaque itération sont contraints et fusionnés en utilisant la notion de réciprocité de lien mutuel.

M. Xu et Y.Li [41] ont proposé un regroupement étendu des pics de densité adaptative pour la détection de communautés qui se chevauchent, appelé EADP. Pour gérer à la fois les réseaux sociaux pondérés et non pondérés, EADP prend en compte les pondérations et incorpore une nouvelle fonction de distance basée sur des nœuds communs pour mesurer la distance entre les nœuds. De plus, EADP adopte une stratégie basée sur l'ajustement linéaire pour choisir les centres de cluster de manière adaptative.

7. Conclusion

La détection de communautés dans les réseaux sociaux est l’un des principaux défis d’analyse des réseaux sociaux.

Dans ce chapitre, nous passons en revue un large éventail d'algorithmes de détection de communauté avec chevauchement, ainsi que des ensembles de base de donné ouverts et plusieurs repère d’évaluations existants. En outre, nous proposons de nouveaux points d'amélioration dans la détection de communautés avec chevauchement de réseaux complexes.

Dans ce qui suit, nous allons décrire notre proposition pour la détection de communautés avec chevauchement dans les réseaux sociaux.

Chapitre 02 : conception du système

1. Introduction

Dans ce chapitre, Nous présentons notre approche dans la détection de communauté chevauchante qui est basé sur le voisinage et la probabilité.

On va commencer tout d'abord par la problématique et les objectifs de notre travail, ensuite on va détailler les étapes de notre conception qui est basé sur la détection des communautés dans les réseaux sociaux on montrant l'efficacité d'utilisation de la probabilité dans notre approche avec un exemple illustratif.

2. Problématique

D'après ce qu'on a vu dans le chapitre précédent, malgré il existe plusieurs algorithmes de détection des communautés dans les réseaux sociaux mais il existe toujours les problèmes de la détermination du nombre de classes et le nombre des nœuds chevauchants, es-ce qu'il est possible de proposer une nouvelle approche de détection des communautés chevauchante efficace, simple et facile à mettre en œuvre qui se base principalement sur le voisinage?

3. Objectif

L'objectif de notre travail est de réaliser un environnement pour la détection de communauté chevauchante dans les réseaux sociaux basant sur le principe de voisinage à l'aide de la probabilité qui nous aide à choisir les bonnes communautés pour les nœuds et de connaitre les nœuds qui se chevauchent et le comparer avec l'algorithme de Louvain qui était déjà implémenter de l'année passée[1].

4. Schéma et description générale de l'approche proposée

Dans l'analyse des réseaux sociaux, les graphes ont été largement utilisés comme moyens de représenter formellement les individus par des nœuds et les relations entre eux avec des liens.

Une communauté est généralement considérée comme un groupe de nœuds avec plus de connexions entre ses membres qu'entre les membres du reste du réseau. Les communautés dans les réseaux se chevauchent également car les nœuds appartiennent à plusieurs clusters à la fois. En raison des difficultés d'évaluation des communautés détectées et du manque d'algorithmes évolutifs, la tâche de détection de communautés qui se chevauchent dans les réseaux reste en grande partie un problème ouvert [42].

Notre but est d'utiliser d'une méthode basée sur le voisinage pour résoudre le problème de la détection de communautés dans les réseaux sociaux.

Pour répondre à ce problème, on a proposé une nouvelle méthode de détection des communautés locales ne prenant en compte que le voisinage d'un nœud donné et sans accéder à l'ensemble du réseau, il est principalement basé sur la notion de voisinage qui est la base du réseau social et qui affecte toutes les autres critères.

L'algorithme que nous proposons consiste à regrouper chaque nœuds de notre réseau avec ses voisin qui nous donnent à la fin des communautés disjointes et d'autres chevauchantes afin d'appliquer quelques étapes qui se répètent jusqu'au trouver les communautés finales. Quand la détection des communautés se termine, à l'aide de la probabilité notre algorithme permette à chaque nœuds d'appartient à des communautés différentes et aussi de détecter les nœuds chevauchants avec une probabilité d'appartenir dans chaque communautés entre 0 et 100.

Après la lecture de notre base de données et la transformer en graphe, l'algorithme est divisé en 3 phases :

Phase 01 : cette phase se déroule en deux étapes :

1. Initialisation
2. Arrangement

Phase 02 : cette phase se déroule en cinq étapes :

1. Regroupement des classes identiques
2. Regroupement des classes inclus
3. Regroupement des classes ou l'intersection est (n-1)
4. Regroupement des classes ou l'intersection est (n-2)
5. Regroupement des classes ou l'intersection est (n-3)
6. Regroupement des classes ou l'intersection est (n-4)

Phase 03 : cette phase se déroule en deux étapes

1. Calculer la probabilité d'appartenir d'un nœud à une communauté pour chaque nœud.
2. Extraction des communautés disjointes et chevauchantes.

La figure 1.2 ci-dessous représentent le schéma général de notre approche

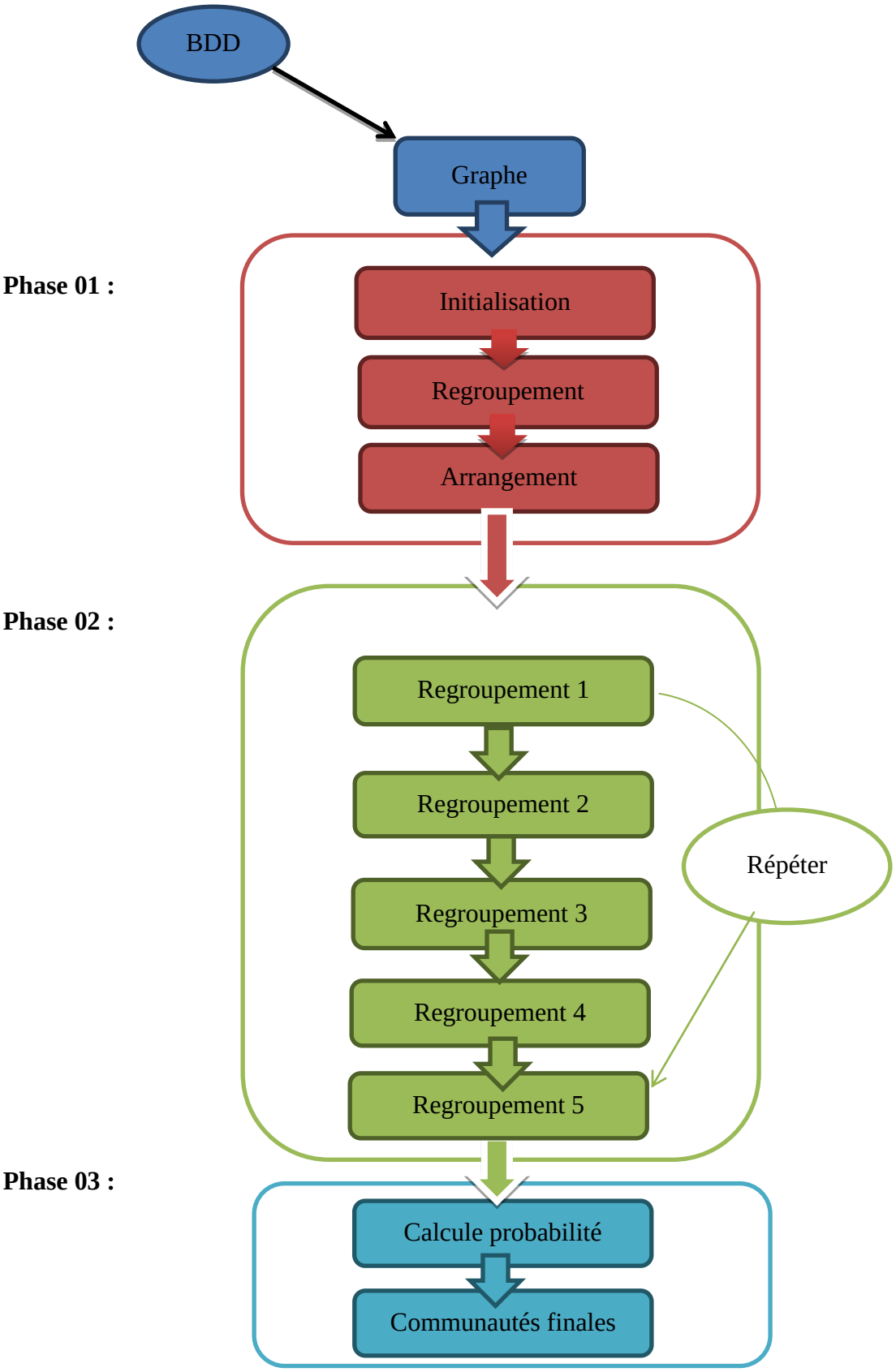

Figure 2.1 Schéma descriptive de notre approche.

5. Conception détaillée de l'approche proposée

Dans notre algorithme, on va commencer tout d'abord par créer des classes à partir des voisins des nœuds, le nombre des classes initiales égales aux nombres des nœuds où chaque classe contient un nœud avec ses voisins. Après passé au quelque étapes, des communautés disjointes sont découvertes lorsque la probabilité d'appartenir à une classe est forte. Des communautés chevauchantes sont trouver dans le cas où il y'a aucune forte probabilité d'appartenir à une classe. Après avoir utilisé les probabilités, il y a une possibilité de disparition de certaines classes qui contient que les nœuds chevauchantes ou des nœuds avec une très faible probabilité. Le nombre des regroupements et l'utilisation de la probabilité se diffère selon la base de données(le nombre des nœuds). Dans cette partie, on va détaillons les étapes de conception de notre approche qu'on a vue dans la figure 4.1. Commençons par un graphe et se terminant par des communautés qui peuvent être chevauchantes ou disjointe.

On considère un graphe G (V, E), où V est l'ensemble des nœuds et $E \subseteq V \times V$ est l'ensemble des arêtes. On notera N le nombre des nœuds de G, avec N = |V| et le nombre de voisinage du nœud Ni est Vi.

Dans ce qui suit, on va détailler les étapes qu'on a déjà vues.

5.1 La première phase

Cette phase consiste à initialiser les classes pour chaque nœud du réseau. Pour cela nous avons affecté chaque nœud à une classe avec tous ses voisins.

Le nombre de classe est N (nombre des nœuds), chaque classe est nommée par le nom de son nœud principal ajoutant le C (C_1, C_2, C_3, ..., C_k). Cette phase consiste à appliquer trois étapes sur les nœuds : l'initialisation, le regroupement et l'arrangement.

Les étapes de cette phase ont était déjà proposé l'année passé.

5.1.1 Initialisation

Dans cette première partie, on va créer C_n classes chevauchantes initiales où chaque classe C_i contient V_i+1 nœuds et on note la redondance de chaque nœud dans les classes (dans cette étape la redondance est égale à 1)

> Le nombre total des nœuds dans tous les classes est le nombre des nœuds N plus le nombre des liens fois deux.
> Le nombre total des classes initial est le nombre des nœuds $\boldsymbol{N}$

5.1.2 Regroupement

Dans cette partie on va regrouper les classes identique c'est-à-dire les classes où quel que soit Ni appartiens à Cj Ni appartiens aussi à C_k, le regroupement ce fait deux par deux selon une boucle qui suivre l'ordre qu'on à fait la phase d'arrangement, l'un des deux classes sera supprimer on ajoutant la redondance des nœuds de la classe supprimer à l'autre classe.

5.1.3 Arrangement

Dans cette étape la modification ne se fait pas sur les nœuds et les classes mais sur l'ordre des classes parce qu'il est très important et influe directement sur les résultats comme dans le cas de Louvain et de plusieurs autres algorithmes de détection des communautés, c'est pourquoi on a les trié avec un ordre décroissant où la classe qui contient le grand nombre des nœuds est placé la première classe.

5.2 La deuxième phase

Cette phase consiste à regrouper les classes selon leurs homogénéités, elle consiste à appliquer six étapes itératives sur les nœuds :

1. Regroupement 01 : regroupement des classes inclus
2. Regroupement 02 : regroupement des classes où la différence est un seul nœud
3. Regroupement 03 : regroupement des classes où la différence est deux nœuds
4. Regroupement 04 : regroupement des classes où la différence est trois nœuds
5. Regroupement 05 : regroupement des classes où la différence est quatre nœuds

Les regroupements ont effectué selon le nombre des nœuds :

- Si le nombre des nœuds est inférieur à 30 nœuds on répète les deux premiers regroupements 1 et 2 seulement.
- Si le nombre des nœuds est entre 30 et 60 nœuds on répète les trois regroupements 1,2 et 3
- Si le nombre des nœuds est entre 60 est 115 on répète tous les cinq regroupements

- Les deux premiers regroupements ont était déjà proposé dans le mémoire de l'année passée [1], et avec les tests que nous avons réalisé nous avons trouvé que les deux premiers regroupements donnent des bonne résultats seulement avec les petites bases de données, pour ce but nous avons ajouté les trois derniers regroupements.

5.2.1 Regroupement 01

Dans cette étape, on va regrouper les petites classes qui ont inclus à 100% dans des grandes classes ajoutant les redondances des nœuds de la petite classe aux redondances de la grande classe avec la suppression de la petite classe.

5.2.2 Regroupement 02

Cette phase consiste à regrouper deux classes où N est le nombre des nœuds de la grande classe est M et le nombre des nœuds de la petite classe, où la différence entre ces deux classe est égale à M-1, c'est-à-dire un seule nœud, en ajoutant les redondances des nœuds existant et le différent nœud avec sa redondance à la grande classe.

5.2.3 Regroupement 03

Cette phase consiste à regrouper deux classes où N est le nombre des nœuds de la grande classe est M et le nombre des nœuds de la petite classe, où la différence entre ces deux classe est égale à M-2, c'est-à-dire deux nœuds, en ajoutant les redondances des nœuds identique et les nœuds manquant avec ses redondance à la grande classe.

5.2.4 Regroupement 04

Cette phase consiste à regrouper deux classes où N est le nombre des nœuds de la grande classe est M et le nombre des nœuds de la petite classe, où la différence entre ces deux classe est égale à M-3 c'est-à-dire deux nœuds, en ajoutant les redondances des nœuds identique et les nœuds manquant avec ses redondance à la grande classe.

5.2.5 Regroupement 05

Cette phase consiste à regrouper deux classes où N est le nombre des nœuds de la grande classe est M et le nombre des nœuds de la petite classe, où la différence entre ces deux classe est égale à M-4 c'est-à-dire trois nœuds, en ajoutant les redondances des nœuds identique et les nœuds manquant avec ses redondance à la grande classe.

Le regroupement se fait avec la comparaison entre les nœuds de la dernière classe (la plus petite communauté) avec toutes les autres classes commençant la comparaison avec la première classe (la plus grande classe).

- L'arrangement se fait après chaque regroupement, si deux classes ont le même nombre des nœuds on prend en compte la somme des redondances de chaque classe.
- Répétez cette phase jusqu'à ce qu'aucun regroupement ou suppression des classes ne soit possible.
- Le résultat de cette phase est un nombre des communautés où chaque nœud peut faire partir d'une ou plusieurs communautés avec un nombre de répétition bien précis dans chaque communauté.

5.3 La troisième phase

C'est la dernière phase où on va faire la décision d'appartenir pour chaque nœud pour obtenir les communautés finales. Cette phase consiste à appliquer deux étapes sur les nœuds: calculs la probabilité d'appartenance des nœuds dans les communautés et extraire les communautés disjointes et chevauchantes.

5.3.1 Calcul de probabilité

Cette étape consiste à calculer la probabilité d'appartenir des nœuds dans les communautés qui nous aide à extraire les communautés finales.

La probabilité d'un nœud V_j dans une communauté C_i est calculé comme suit : $[(RC_i*100)/RV_j]$, où : RC_i est la redondance du nœud V_j dans la communauté C_i et RV_j est la redondance totale du nœud(le nombre totale de ces voisins).

5.3.2 Communautés finales

Cette étape consiste à extraire les communautés finales à l'aide de l'étape précédente où on va décider c'est un nœud est chevauchant ou il appartient à une certaine classe.

6. Algorithme de l'approche

Entrées :

G = (V, E) : le graphe initial
liste_des_noeuds : la liste des nœuds du graphe.
Liste_Rtotale : la liste de redondance de chaque nœud de la liste liste_des_noeuds

***Sorties* :**

C_1, C_2, C_3, ..., C_k : les communautés.

Les fonctions:

- Taille : une fonction qui retourne la taille d'une liste
- Tout(L1,L2) : une fonction qui retourne vrai si deux liste ont la même taille et partage les mêmes éléments.
- Voisin() : une fonction qui retourne une liste des voisins d'un nœud.
- Ajouter() : une fonction qui ajoute une variable à une liste.
- Ajouter_liste() : une fonction qui ajoute une liste à une liste des liste.
- Tri_com() : une fonction pour trier les communautés prenant en compte les redondances.
- Inclu() : une fonction booléenne qui retourne vrai si la première communauté est inclus dans la deuxième.
- CalculD : une fonction qui calcule la différence des nœuds entre deux communautés
- Supprimer() : une fonction qui supprime les communautés et laisse une liste vide.
- Elimine_vide : une fonction qui supprime les listes vides
- Max() : une fonction qui retourne l'index du max d'une liste

6.1 La première phase

Entré :

- G = (V, E) : le graphe initial
- liste_des_noeuds : la liste des nœuds du graphe.

Sortie :

- N classe trié avec un ordre décroissant (chaque classe contient un nœud avec ses voisins).

L'algorithme de la première phase :

```
Début :
   t= Taille(liste_des_noeuds)
   Pour i allant de 1 à t
     liste_de_voisin=[ ]
     nœud=liste_des_noeuds[i]
     liste_de_voisin=voisin(nœud)
     Ajouter(nœud,liste_de_voisin)
     Ajouter_com(liste_de_voisin,liste_de_communauté)
   finPour ;
   Pour i allant de 1 à Taille(liste_de_communauté)
     Pour i allant de i+1 à Taille(liste_de_communauté)
       Si Tout(liste_de_communaute(i),liste_de_communaute(j)) faire :
         Regroupement(liste_de_communaute(i),liste_de_communaute(j))
    finPour ;
     finPour ;
   Tri_com(liste_de_communauté)
 fin.
```

6.2 La deuxième phase

Les fonctions :

- Regroupement() : une fonction qui regroupe deux communautés identique où une communauté inclut dans une autre
- Regroupement1() : une fonction qui regroupe deux communautés qui ont de un jusqu'à quatre nœuds différents

L'algorithme de la deuxième phase:

```
Début :

   nouveau_nb_com=taille(liste_de_communauté)
   ancien_nb_com =nouveau_nb_comm+1
   Tant que ancien_nb_com>nouveau_nb_com:
     Pour i allant de Taille(liste_de_communauté[]) à 1
       petite_comm=liste_de_communauté[i]
       Pour j allant de 1 à (liste_de_communauté[]-1) :
         grande_comm=liste_de_communaut[j]
         si inclu(petite_comm,grande_comm) alors :
           regroupement(petite_comm_grande_comm)
          supprimer(petite_comm)
         Fin si ;
       Fin pour ;
     Fin pour ;
     Elimine_vide (liste_des_communauté[])
    Trier_com(liste_des_communauté[])
   Si Taille(liste_des_noeuds)<30 faire :
```

```
    Pour i allant de taille(liste_de_communauté[]) à 1
      petite_comm=liste_de_communauté[i]
      Pour j allant de 1 à (liste_de_communauté[]-1) :
        grande_comm=liste_de_communaut[j]
        D=calculD(grande_comm,petite_comm)
        Si D=1 faire :
          Regroupement1(grande_comm,petite_comm)
          Supprimer(petite_comm)
          Trier_comm(liste_de_communauté)
        Fin Si ;
     Fin Pour ;
  Fin Pour ;
  Elimine_vide(liste_de_communauté)
Sinon si Taille(liste_des_noeuds)=>30 et Taille(liste_des_noeuds)<60 faire :
    Pour i allant de taille(liste_de_communauté[]) à 1
      petite_comm=liste_de_communauté[i]
      Pour j allant de 1 à (liste_de_communauté[]-1) :
        grande_comm=liste_de_communaut[j]
        D=calculD(grande_comm,petite_comm)
        Si D=1 faire :
          Regroupement1(grande_comm,petite_comm)
          Supprimer(petite_comm)
          Trier_comm(liste_de_communauté)
        Fin Si ;
      Fin Pour ;
    Fin Pour
  Fin Si ;
  Elimine_vide(liste_de_communauté)
    Pour i allant de taille(liste_de_communauté[]) à 1
      petite_comm=liste_de_communauté[i]
      Pour j allant de 1 à (liste_de_communauté[]-1) :
        grande_comm=liste_de_communaut[j]
        D=calculD(grande_comm,petite_comm)
        Si D=2 faire :
          Regroupement1(grande_comm,petite_comm)
          Supprimer(petite_comm)
          Trier_comm(liste_de_communauté)
        Fin Si ;
      Fin Pour ;
   Fin Pour ;
  Elimine_vide(liste_de_communauté)
Sinon si Taille(liste_des_noeuds)=>60 faire :
    Pour i allant de taille(liste_de_communauté[]) à 1
      petite_comm=liste_de_communauté[i]
      Pour j allant de 1 à (liste_de_communauté[]-1) :
        grande_comm=liste_de_communaut[j]
        D=calculD(grande_comm,petite_comm)
        Si D=1 faire :
          Regroupement1(grande_comm,petite_comm)
          Supprimer(petite_comm)
```

```
            Trier_comm(liste_de_communauté)
          Fin Si ;
        Fin Pour ;
      Fin Pour ;
      Elimine_vide(liste_de_communauté)
        Pour i allant de taille(liste_de_communauté[]) à 1
          petite_comm=liste_de_communauté[i]
          Pour j allant de 1 à (liste_de_communauté[]-1) :
            grande_comm=liste_de_communaut[j]
            D=calculD(grande_comm,petite_comm)
            Si D=2 faire :
              Regroupement1(grande_comm,petite_comm)
              Supprimer(petite_comm)
              Trier_comm(liste_de_communauté)
            Fin Si ;
        Fin Pour ;
      Fin Pour ;
      Elimine_vide(liste_de_communauté)
        Pour i allant de taille(liste_de_communauté[]) à 1
          petite_comm=liste_de_communauté[i]
          Pour j allant de 1 à (liste_de_communauté[]-1) :
            grande_comm=liste_de_communaut[j]
            D=calculD(grande_comm,petite_comm)
            Si D=3 faire :
              Regroupement1(grande_comm,petite_comm)
              Supprimer(petite_comm)
              Trier_comm(liste_de_communauté)
            Fin Si ;
          Fin Pour
        Fin Pour ;
      Elimine_vide(liste_de_communauté)
        Pour i allant de taille(liste_de_communauté[]) à 1
          petite_comm=liste_de_communauté[i]
          Pour j allant de 1 à (liste_de_communauté[]-1) :
            grande_comm=liste_de_communaut[j]
            D=calculD(grande_comm,petite_comm)
            Si D=4 faire :
              Regroupement1(grande_comm,petite_comm)
              Supprimer(petite_comm)
              Trier_comm(liste_de_communauté)
            Fin Si ;
          Fin Pour :
        Fin Pour ;
      Elimine_vide(liste_de_communauté)
    Fin Si ;
Fin.
```

6.3 La troisième phase

L'algorithme de la troisième phase:

```
Début :
  Pour i allant de 1 à (Taille(liste_des_noeuds)) faire :
    nœud :=liste_des_noeuds[i]
    totale=liste_Rtotal[i]
    pour j allant de1 à (Taille(liste_de_communauté)) faire :
       communaute := liste_de_communauté[j]
       pour x allant de 1 à (Taille(communaute)) faire :
          Si nœud=communaute[x] faire :
             Pos=x
             repitition_noeudC := repitition[x]
             proba_noeudC=(repitition_noeudC*100)totale
             Ajouter(proba_noeudC,liste_de_proba)
          Sinon si (x=(Taille(communaute)-1) et nœud != communaute[x]) faire :
             Ajouter(0,liste_de_proba)
         Ajouter_liste(liste_de_proba,Liste_des_proba)
         Fin Si ;
       Fin pour ;
    Fin pour ;
  Fin pour ;
  Pour i allant de 1 à Taille(liste_des_noeuds) faire :
    liste_de_proba=liste_des_proba[i]
    maxpos= Max(liste_de_proba)
    Si Taille(liste_des_noeuds)<60 faire :
       Si liste_de_proba[maxpos]=>70 faire :
          Ajouter_noeud_comm(nœud,liste_de_communauté,maxpos)
       Sinon si liste_de_proba[maxpos]<70 faire :
         Ajouter(nœud,liste_noeud_chevauch)
       Fin Si ;
    Sinon Si Taille(liste_des_noeuds)=>60
       Si liste_de_proba[maxpos]=>30 faire :
          Ajouter_noeud_comm(nœud,liste_de_communauté,maxpos)
       Sinon si liste_de_proba[maxpos]<30 faire :
         Ajouter(nœud,liste_noeud_chevauch)
       Fin Si ;
     Fin Si ;
  Fin Pour ;
Fin.
```

7. Exemples illustratifs

On va utiliser deux exemples l'un donne des communautés disjointes et l'autre donne des communautés chevauchantes pour monter l'efficacité de notre approche dans les deux cas.

Tous les schémas et les résultats qu'on va l'utiliser au cours de ces exemples sont de notre application.

7.1 Exemple 01

Notre premier exemple est un graphe artificiel, connexe et non orienté constitué de quinze nœuds et quatorze liens.

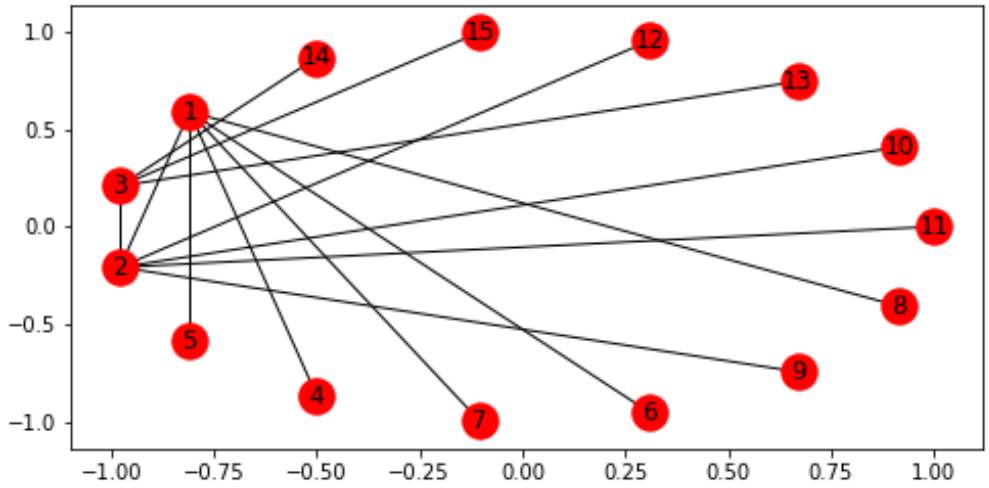

Figure 2.1 La représentation graphique de l'exemple 01

7.1.1 La première phase

Comme on a déjà mentionné, cette phase consiste à initialiser des classes pour chaque nœud du réseau, regrouper les communautés identiques et les trier avec un ordre décroissement par rapport au nombre des nœuds. Pour cela nous avons affecté chaque nœud à une classe avec tous ses voisins.

Dans le tableau suivant on va représenter les communautés initiales après l'initialisation :

Initialisation	
Les Nœuds Ni	Les communautés Ci
1	1 2 4 5 6 7 8
2	2 1 3 9 10 11 12
3	3 2 13 14 15
4	4 1
5	5 1
6	6 1
7	7 1
8	8 1
9	9 2
10	10 2
11	11 2
12	12 2
13	13 3

14	14 3
15	15 3

Tableau 2.1 les communautés résultantes de l'initialisation

Dans cet exemple, le résultat ne change pas après l'initialisation car il n'existe pas des communautés identiques et les communautés ont déjà trié par ordre décroissant.

Dans cette phase, tout les redondances des nœuds dans les communautés est égale à 1.

Les communautés résultantes de cette étape sont :

C1 : 1 2 4 5 6 7 8
C2 : 2 1 3 9 10 11 12
C3 : 3 2 13 14 15
C4 : 4 1
C5 : 5 1
C6 : 6 1
C7 : 7 1
C8 : 8 1
C9 : 9 2
C10 : 10 2
C11 : 11 2
C12 : 12 3
C13 : 13 3
C14 : 14 3
C15 : 15 3

7.1.2 La deuxième phase

Dans cet exemple, on va répéter que les deux premiers regroupements car le nombre totale des nœuds ne dépasse pas 30 nœuds.

Regroupement 01 :

Dans cette étape, on va regrouper les communautés qui ont inclus dans d'autre communauté :

- Les communautés C4, C5, C6, C7 et C8 ont inclus dans C1.
- Les communautés C9, C10, C11 et C12 ont inclus dans C2
- Les communautés C13, C14 et C15 ont inclus dans C3

Les communautés résultantes de cette étape sont :

C1 : 1 2 4 5 6 7 8
C2 : 2 1 3 9 10 11 12
C3 : 3 2 13 14 15

Regroupement 02 :

Il y a aucun changement dans cette étape car il n'existe aucune communauté où l'intersection est seulement un nœud avec d'autres communautés.

Donc le résultat de cette phase sont les trois classes C1, C2 et C3 du deuxième regroupement.

7.1.3 La troisième phase

Dans cette phase, on va calculer la probabilité d'appartenance des nœuds dans les communautés et extraire les communautés finales.

Le tableau ci-dessus nous montre la probabilité des nœuds avec la décision d'appartenance :

Les Nœuds Ni	Redondance	Proba(C1)	Proba(C2)	Proba(C3)	Décision
1	7	6/7(85.1%)	1/7(14.9%)	0/7(0%)	C1
2	7	1/7(14.3%)	5/7(71.4%)	1/7(14.3%)	C2
3	5	0/5(0%)	1/5(20%)	4/5(80%)	C3
4	2	2/2(100%)	0/2(0%)	0/2(0%)	C1
5	2	2/2(100%)	0/2(0%)	0/2(0%)	C1
6	2	2/2(100%)	0/2(0%)	0/2(0%)	C1
7	2	2/2(100%)	0/2(0%)	0/2(0%)	C1
8	2	2/2(100%)	0/2(0%)	0/2(0%)	C1
9	2	0/2(0%)	2/2(100%)	0/2(0%)	C2
10	2	0/2(0%)	2/2(100%)	0/2(0%)	C2
11	2	0/2(0%)	2/2(100%)	0/2(0%)	C2
12	2	0/2(0%)	2/2(100%)	0/2(0%)	C2
13	2	0/2(0%)	0/2(0%)	2/2(100%)	C3
14	2	0/2(0%)	0/2(0%)	2/2(100%)	C3
15	2	0/2(0%)	0/2(0%)	2/2(100%)	C3

Tableau 2.2 Table de probabilité avec décision.

Dans cet exemple, le nombre des nœuds est égal à 15, donc si le pourcentage d'appartenance d'un nœud dans une communauté est supérieur ou égale à 70% le nœud est appartient dans cette communauté.

Donc le résultat de cet exemple est 3 communautés disjointes car chaque nœud appartient à une communauté avec un pourcentage qui dépasse 70%.

La figure ci-dessus nous montre le résultat final de notre algorithme dans cet exemple :

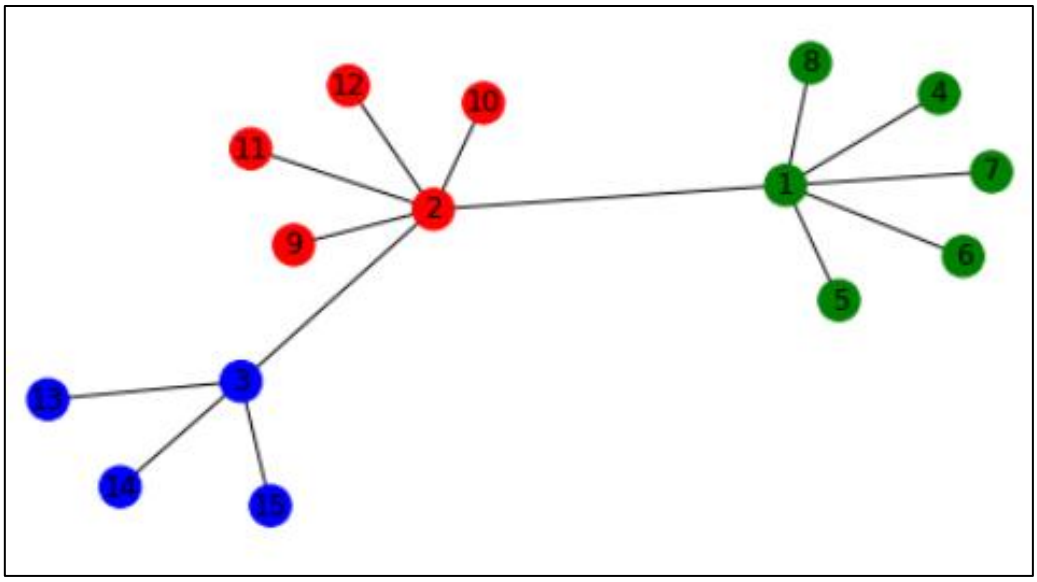

Figure 2.2 résultat de l'exemple 01

7.2 Exemple 02

Dans le deuxième exemple, on va prendre le même graphe du premier exemple on supprime le lien entre les nœuds 1 et 2, on ajoutant deux nœuds qui ont des liens avec les nœuds 1 et 2.

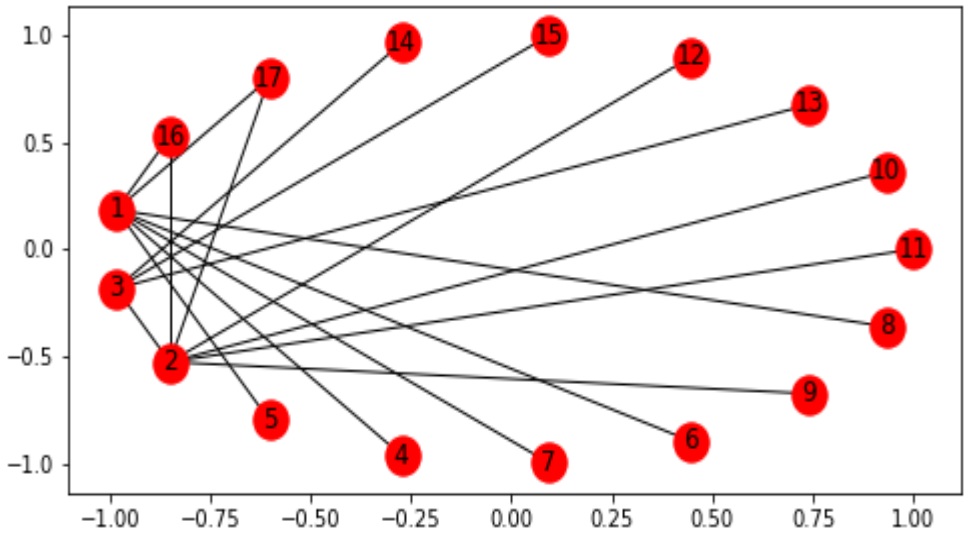

Figure 2.3 la représentation graphique de l'exemple 02

7.2.1 La première phase :

Comme on a déjà mentionné, cette phase consiste à initialiser des classes pour chaque nœud du réseau, regrouper les communautés identiques et les trier avec un ordre décroissement par rapport au nombre des nœuds. Pour cela nous avons affecté chaque nœud à une classe avec tous ses voisins.

Dans le tableau suivant on va représenter les communautés initiales après l'initialisation :

Initialisation	
Les Nœuds Ni	Les communautés Ci
1	1 4 5 6 7 8 16 17
2	2 3 9 10 11 12 16 17
3	3 2 13 14 15
4	4 1
5	5 1
6	6 1
7	7 1
8	8 1
9	9 2
10	10 2
11	11 2
12	12 2
13	13 3
14	14 3
15	15 3
16	16 1 2
17	17 1 2

Tableau 2.3 les communautés résultantes de l'initialisation

Dans cet exemple, on passe directement à l'arrangement, car il n'existe pas des communautés identiques.

Les communautés résultantes après l'arrangement (on change l'emplacement de C16 et C17) sont :

C1 : 1 4 5 6 7 8 16 17
C2 : 2 3 9 10 11 12 16 17
C3 : 3 2 13 14 15
C4 : 16 1 2
C5 : 17 1 2
C6 : 4 1
C7 : 5 1
C8 : 6 1
C9 : 7 1
C10 : 8 1
C11 : 9 2
C12 : 10 2
C13 : 11 2
C14 : 12 2
C15 : 13 3
C16 : 14 3
C17 : 15 3
Dans cette phase, tous les redondances des nœuds dans les communautés est égale à 1.

7.2.2 La deuxième phase :

Dans cet exemple, on va répéter que les deux premiers regroupements car le nombre totale des nœuds ne dépasse pas 30 nœuds.

Regroupement 01 :

Dans cette étape, on va regrouper les communautés qui ont inclus dans d'autre communauté :

- Les communautés C6, C7, C8, C9 et C10 ont inclus dans C1.
- Les communautés C11, C12, C13 et C14 ont inclus dans C2
- Les communautés C15, C16 et C17 ont inclus dans C3

Les communautés résultantes de cette étape sont :

C1 : 1 4 5 6 7 8 16 17
C2 : 2 3 9 10 11 12 16 17
C3 : 3 2 13 14 15
C4 : 16 1 2
C5 : 17 1 2

Regroupement 02 :

Dans cette étape, on va regrouper les communautés où l'intersection est seulement un nœud avec d'autres communautés. Donc les communautés C5 va regrouper avec C1 on ajoutant le nœud 2 avec sa redondance de C5 dans C1, et C4 va regrouper avec C2 on ajoutant le nœud 1 avec sa redondance de C4 dans C2.

Les communautés résultantes de cette étape sont :

C1 : 1 4 5 6 7 8 16 17 2
C2 : 2 3 9 10 11 12 16 17 1
C3 : 3 2 13 14 15

7.2.3 La troisième phase

Dans cette phase, on va calculer la probabilité d'appartenance des nœuds dans les communautés et extraire les communautés finales.

Le tableau ci-dessus nous montre la probabilité des nœuds avec la décision d'appartenance :

Les Nœuds Ni	Redondance	Proba(C1)	Proba(C2)	Proba(C3)	Décision
1	8	6/8(75%)	2/8(25%)	0/8(0%)	C1
2	8	1/8(12.5%)	6/8(75%)	1/8(12.5%)	C2
3	5	0/5(0%)	1/5(20%)	4/5(80%)	C3
4	2	2/2(100%)	0/2(0%)	0/2(0%)	C1
5	2	2/2(100%)	0/2(0%)	0/2(0%)	C1
6	2	2/2(100%)	0/2(0%)	0/2(0%)	C1
7	2	2/2(100%)	0/2(0%)	0/2(0%)	C1
8	2	2/2(100%)	0/2(0%)	0/2(0%)	C1
9	2	0/2(0%)	2/2(100%)	0/2(0%)	C2
10	2	0/2(0%)	2/2(100%)	0/2(0%)	C2
11	2	0/2(0%)	2/2(100%)	0/2(0%)	C2
12	2	0/2(0%)	2/2(100%)	0/2(0%)	C2

13	2	0/2(0%)	0/2(0%)	2/2(100%)	C3
14	2	0/2(0%)	0/2(0%)	2/2(100%)	C3
15	2	0/2(0%)	0/2(0%)	2/2(100%)	C3
16	3	1/3(33%)	2/3(67%)	0/3(0%)	C1, C2
17	3	2/3(67%)	1/3(33%)	0/3(0%)	C1, C2

Tableau 2.4 Table de probabilité avec décision.

Dans cet exemple, le nombre des nœuds est égal à 17, donc si le pourcentage d'appartenance d'un nœud dans une communauté est supérieur ou égale à 70% le nœud est appartient dans cette communauté.

Donc le résultat de cet exemple est 3 communautés, où les deux communautés C1 et C2 sont des communautés chevauchante qui partage les nœuds 16 et 17.

La figure ci-dessus nous montre le résultat final de notre algorithme dans cet exemple où les nœuds qui ont en blanc sont les nœuds chevauchants :

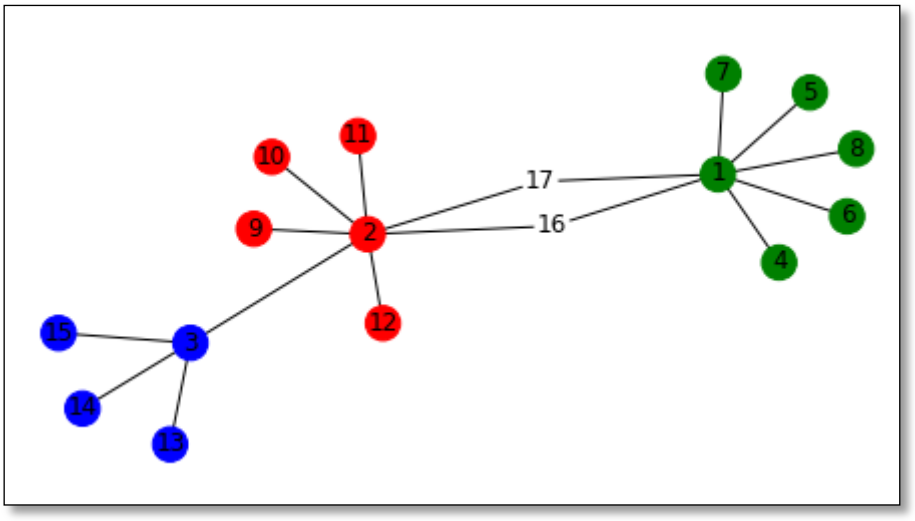

Figure 2.4 résultat de l'exemple 02

8. Conclusion

Dans ce chapitre, nous avons présenté notre nouvelle méthode de détection de communauté chevauchante dans les réseaux sociaux avec deux exemples illustratifs où nous avons montré que notre méthode est capable de détecter facilement les communautés disjointes et chevauchante dans le résultat ne change pas même si on répète l'exécution plusieurs fois sur le même réseau à cause de l'arrangement que nous avons fait après chaque étape.

Dans ce qui suit, on va définir les environnements de travail matériel et logiciel, implémenté les principaux algorithmes de détection de communauté et les comparer ses résultats avec notre algorithme avec quelques exemples.

Chapitre 03 : Implémentation du système

1. Introduction

Dans ce chapitre, nous allons montrer l'implémentation de notre système on montrant les environnements de développement. Nous présentons en premier lieu l'environnement de développement matériel. Ensuite, nous allons définir les logiciels et les outils utilisés pour implémenter notre application et comment les installer.

Nous terminerons ce chapitre par une discussion des résultats obtenue on comparant les résultats de notre algorithme avec d'autre algorithme.

2. Environnement de développement

Avant de présenter nos tests pour obtenir notre algorithme final, nous devons présenter en premier lieu les logiciels et les langages nécessaires pour cet algorithme.

Toutes nos installations et nos tests seront réalisés sur une machine physique (un micro-ordinateur) dont la configuration est la suivantes :

Processeur : Intel® Core™ i5-4210U CPU @ 1.70 GHz
Mémoire Installée(RAM) : 4 Go
Type de système : système d'exploitation windows7 64 bits

Notre application a été développée sous un système d'exploitation Windows 7 de 64 bits où on a utilisé le langage de programmation python 2.7 sous Anaconda.

3. Logiciel et langage de programmation

3.1 Langage de programmation

Le langage utilisé pour le codage de notre application est principalement python 2.7.

Python est un langage de programmation interprété, de haut niveau et polyvalent. Créée par Guido van Rossum et publiée pour la première fois en 1991, la philosophie de conception de Python met l'accent sur la lisibilité du code avec son utilisation notable d'espaces blancs importants. Ses constructions de langage et son approche orientée objet visent à aider les programmeurs à écrire un code clair et logique pour les projets à petite et grande échelle. Python est typé dynamiquement. Il prend en charge plusieurs paradigmes de programmation, y compris la programmation structurée (en particulier, procédurale), orientée objet et fonctionnelle. Python est souvent décrit comme un langage «piles incluses» en raison de sa bibliothèque standard complète [python – Dave Kuhlman].

L'installation de python est gratuite et facile, il suffit de le télécharger dans le site: **https ://www.python.org/downloads/** [w1].

Figure 3.1 Le site d'installation de python.

3.2 Plateforme et IDE

Après l'installation de Python, on passe à l'installation de la plateforme avec tous les logiciels qu'on a besoin, pour ce but on a choisir Anaconda.

ANACONDA est l'une des nombreuses plates-formes open source qui facilitent l'utilisation de langages de programmation open source (R, Python) pour le traitement de données à grande échelle, l'analyse prédictive et le calcul scientifique. La communauté de recherche environnementale peut choisir d'adapter l'utilisation des langages de programmation R ou Python pour analyser les problèmes de science des données sur la plate-forme Anaconda[43].

Téléchargez et installez la dernière version appropriée de la plate-forme Anaconda basée sur le système d'exploitation de l'ordinateur de l'utilisateur et la dernière version de Python à partir du site Web d'Anaconda : https ://www.anaconda.com/products/individual [w2].

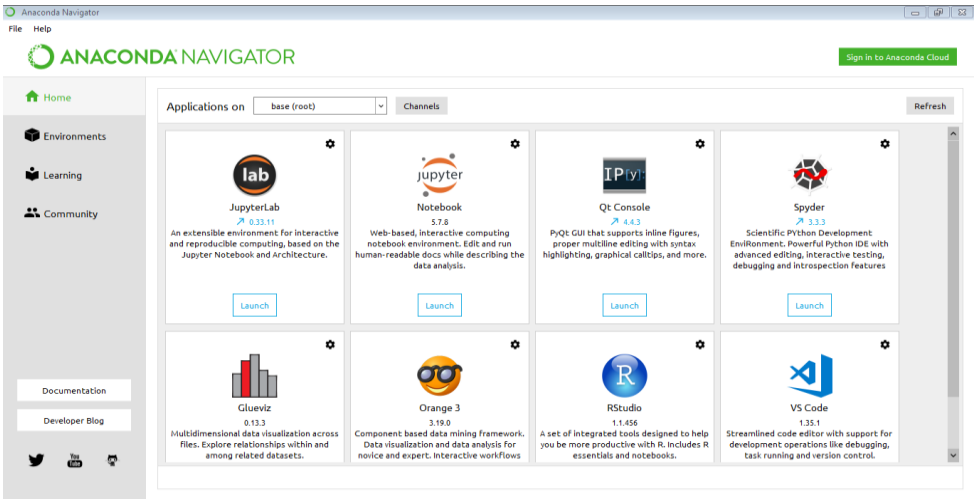

Figure 3.2 Capture d'écran d'Anaconda Navigator.

est un IDE multiplateforme open source pour la science des données qui fournit un environnement de développement scientifique Python qui facilite les fonctionnalités avancées d'édition, de test interactif, de débogage et d'introspection sans utiliser les commandes de ligne de commande qui se trouve dans la barre des taches[43].

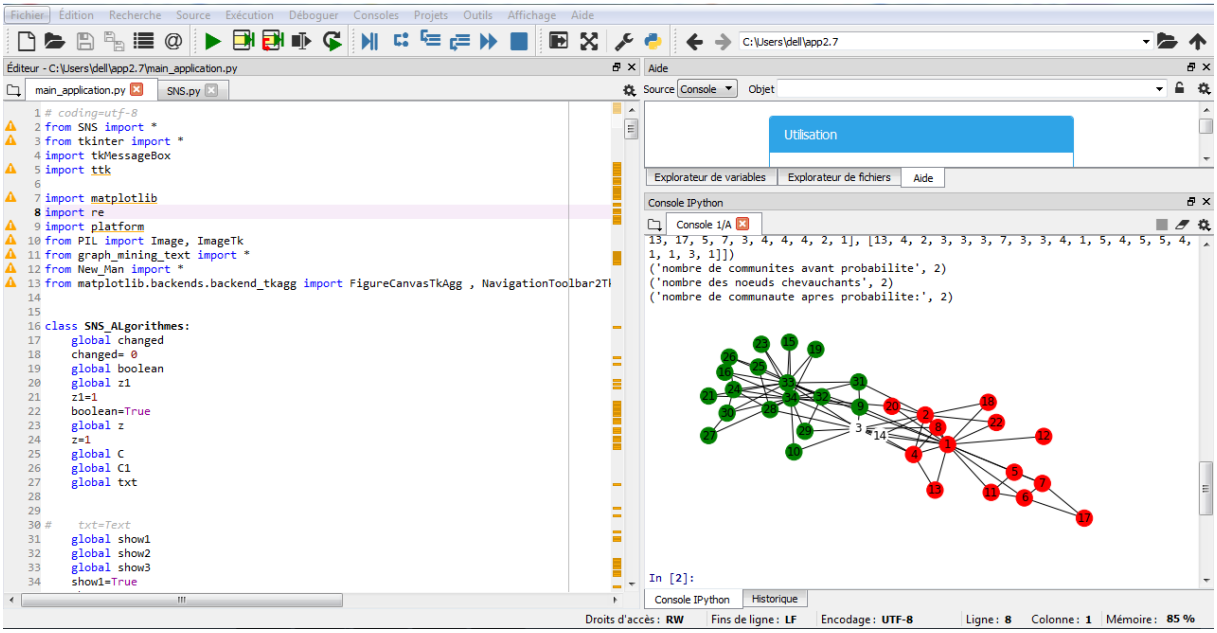

Figure 3.3 Capture d'écran de l'IDE Spyder.

3.3 Package

Après l'installation des logiciel et plateforme, on passe maintenant à l'installation des packages qu'on a besoin pour développer notre application.

NetworkX : est un package de langage Python pour l'exploration et l'analyse des réseaux et des algorithmes de réseau. Le package principal fournit des structures de données pour représenter de nombreux types de réseaux ou de graphiques, y compris des graphiques simples, des graphiques dirigés et des graphiques avec des arêtes parallèles et des boucles automatiques[44].

Tkinter : est une interface graphique pour Python

Matplotlib : est une bibliothèque complète pour créer des visualisations statiques, animées et interactives en Python.

Pour l'installation des packages nécessaires il suffit de taper les codes ci-dessus dans Anaconda Powershell Prompt :

NetworkX : `conda install -c anaconda networkx.`

Python-louvain conda install –c anaconda python-louvain.

Matplotlib: python –m pip install –U matplotlib

Le téléchargement des packages nécessite l'accès à l'internet.

4. Architecture de notre application

Notre application compose de deux parties principales : Une partie pour afficher le graphe et l'exécution de notre algorithme et une partie pour afficher le graphe et exécuter l'algorithme de Louvain qui était déjà implémenter l'année passée[1].

L'architecture de notre application est présentée dans le schéma ci-dessus :

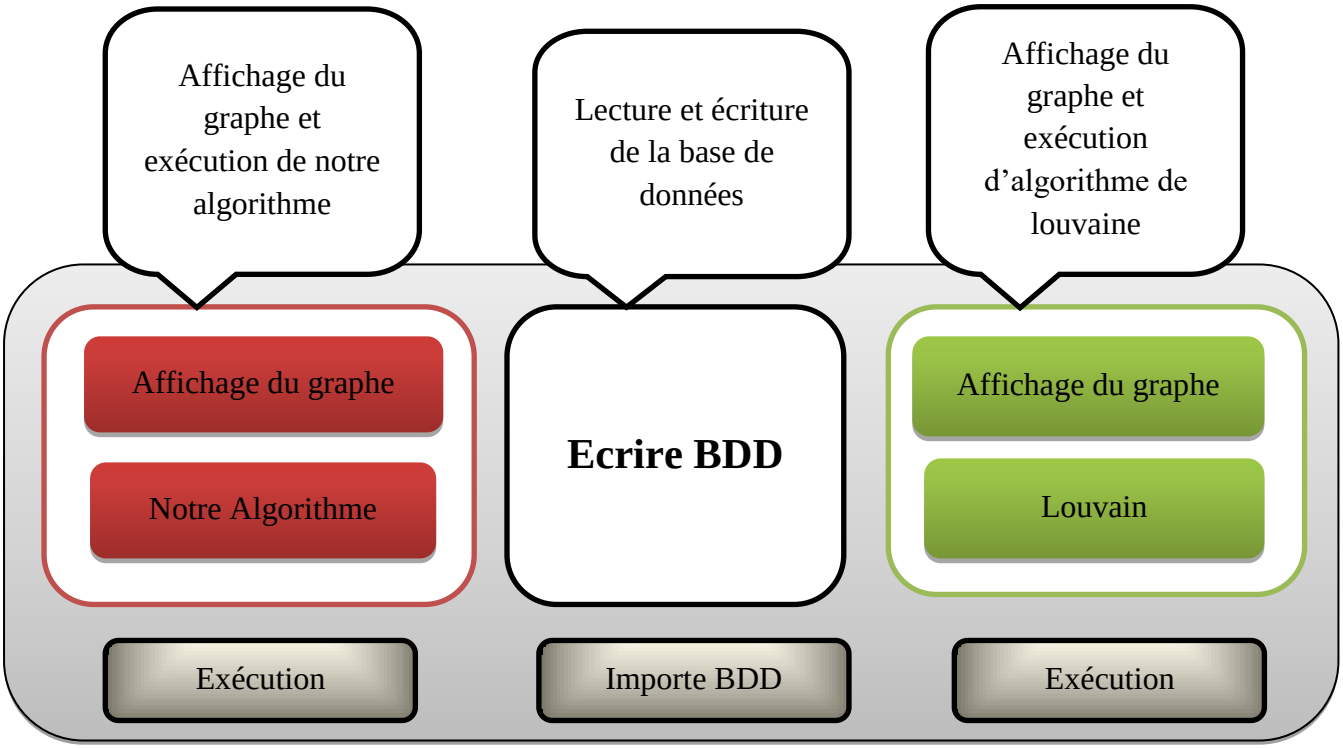

Figure 3.4 Schéma d'architecture générale du système.

5. Fonctionnalités du système

Comme on a dit précédemment, notre système compose de deux parties principales où l'utilisateur peut choisir entre exécuter l'algorithme de Louvain[2] qui était déjà implémenter dans le mémoire de l'année passée [1] ; afficher le graphe ou exécuter notre algorithme.

Concernant les bases de données, l'utilisateur peut écrire les données comme il peut les importer directement. Les bases de données doit être un fichier texte où il y'a un espace entre chaque couple des nœuds qui représente un lien entre ses deux nœuds.

En haut à gauche, l'utilisateur trouvera un bouton ficher qui contient les fonctionnalités suivante (Nouveau, Ouvrir, Enregistrez- sous, quitter) et un autre bouton Edit qui contient Édition contient (Copier, Coller, Couper).

Après l'exécution, une nouvelle fenêtre apparaît pour afficher les résultats des bases de données et l'algorithme que l'utilisateur a choisi.

6. Présentation du système

Dans cette section on va présenter quelque module de notre application.

La figure ci-dessus nous montre l'interface principale de notre application, ou il y'a deux espace pour choisir les algorithmes que l'utilisateur peut exécuter et un espace pour écrire ou importer la base des données qu'on va traiter.

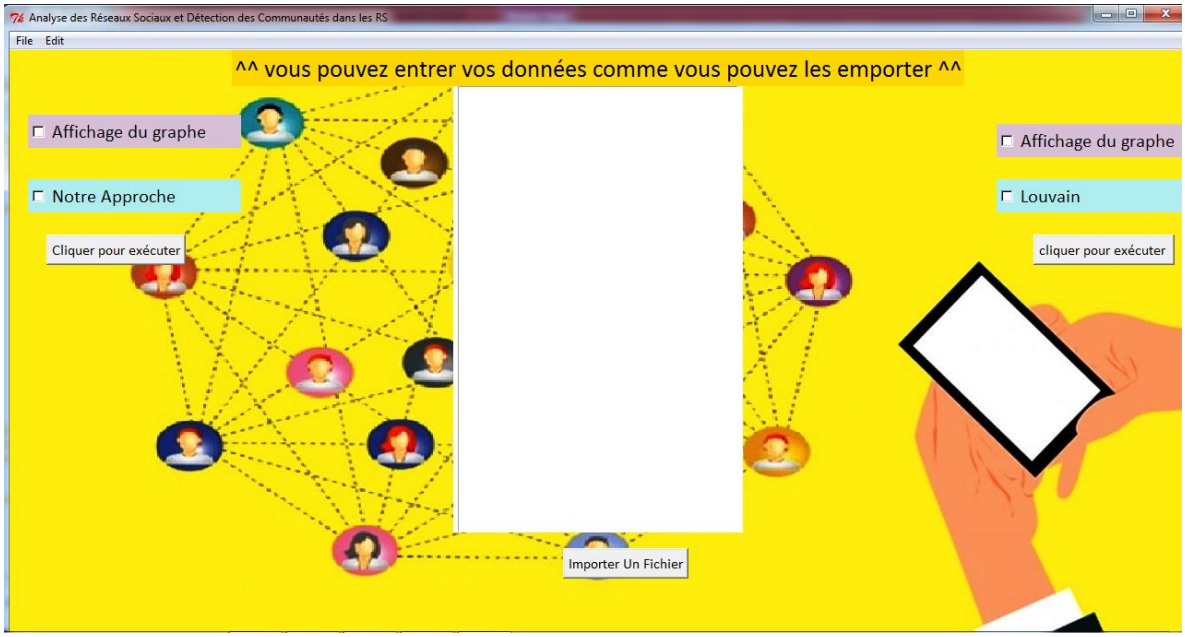

Figure 3.5 Interface générale de l'application

La figure 3.6 ci-dessus nous montre comment l'utilisateur peut saisir les données à traiter et choisir entre exécuter les algorithmes de détection des communautés ou l'affichage du graphe et l'exécution de notre approche

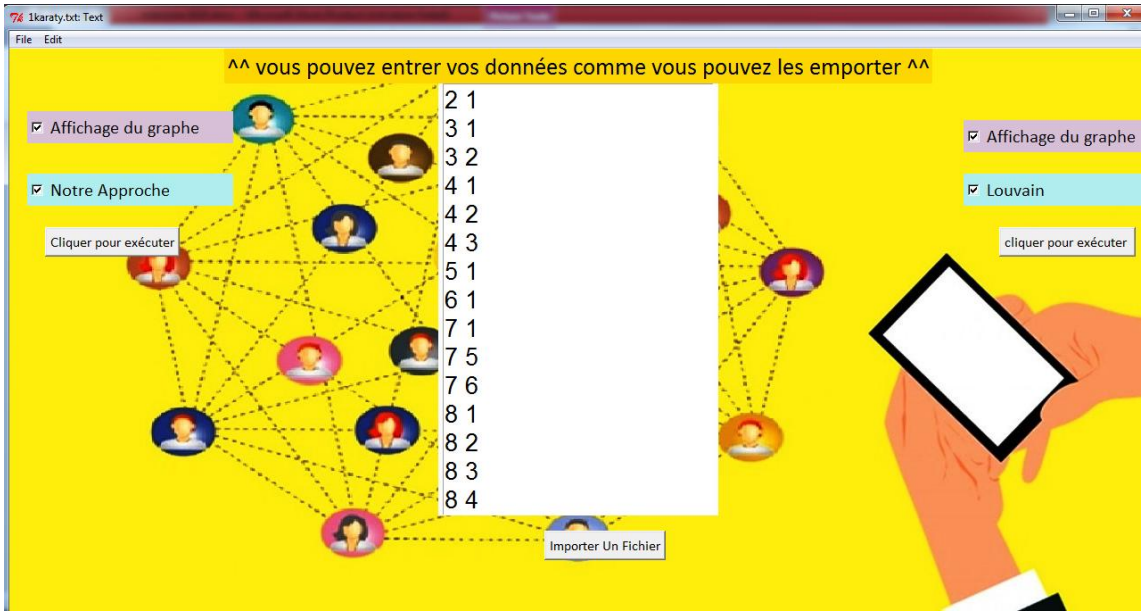

Figure 3.6 La saisie des données et le choix des algorithmes à exécuter

La figure ci-dessus nous montre la nouvelle fenêtre après choisir d'exécuter l'algorithme de louvain et l'affichage du graphe.

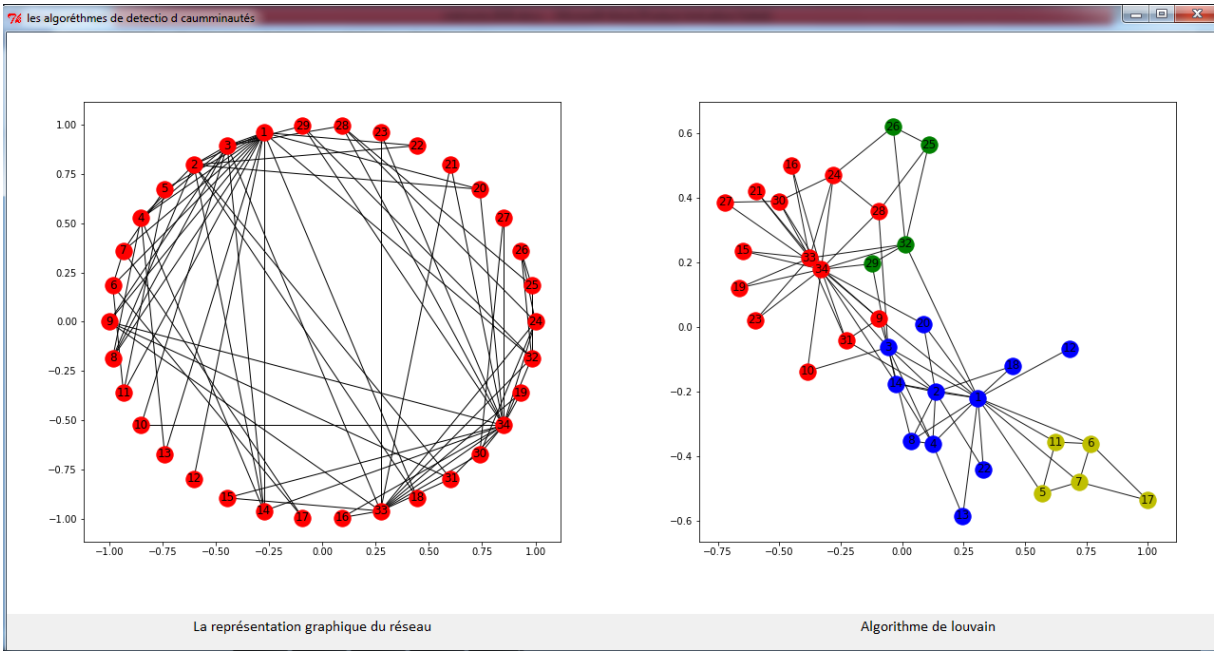

Figure 3.7 Résultat d'affichage du graphe et l'exécution de l'algorithme de Louvain.

La figure ci-dessus nous montre l'affichage du graphe et l'exécution de notre algorithme.

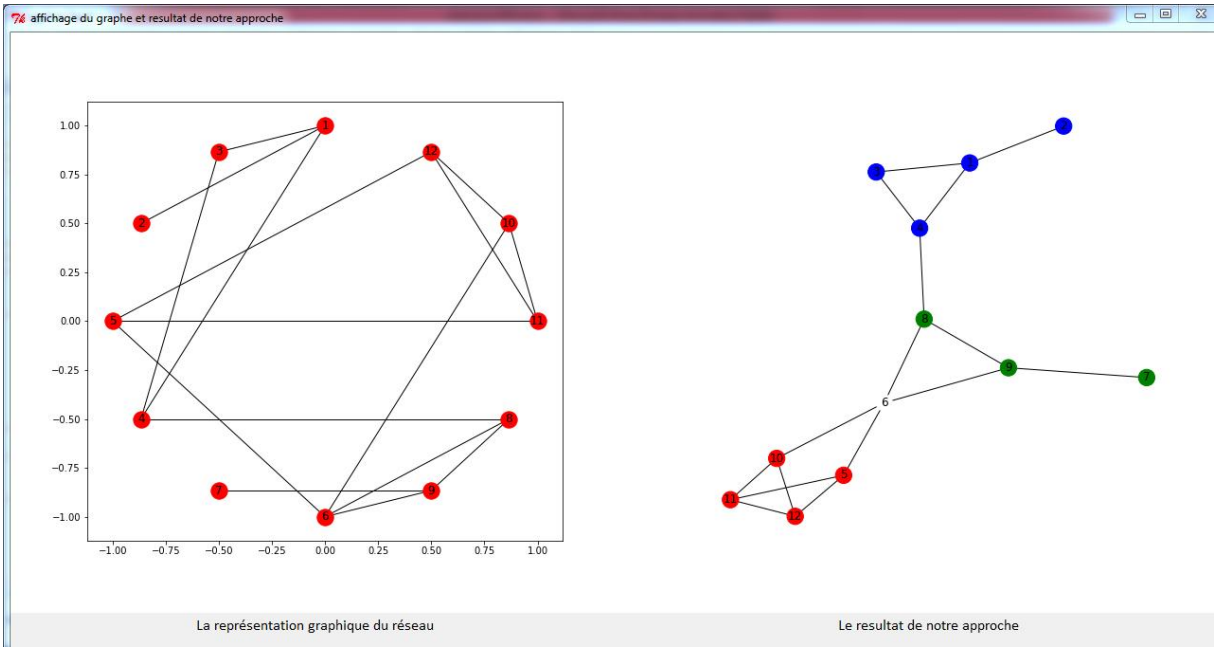

Figure 3.8 Résultat d'affichage du graphe et exécution de notre approche

7. Tests

Dans cette partie, on va discuter les tests que nous avons faits avant de passer au version finale de notre algorithme

7.1 Test sur les réseaux artificiel

Au début, Nous avons fait des tests sur une base de données artificielle de onze nœuds et dix-huit liens pour confirmer que notre algorithme peut donner des bons résultats où il détecte les communautés disjointes et les communautés chevauchantes.

Le résultat d'exécution de notre algorithme sur le graphe est représenté dans la figure suivante.

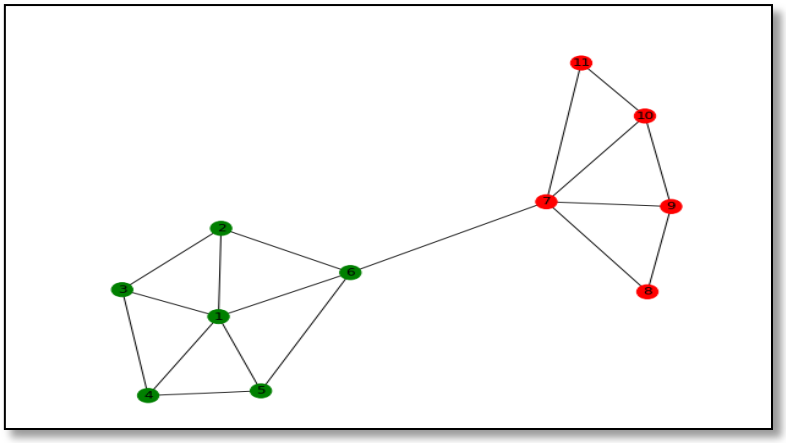

Figure 3.9 résultat d'algorithme sur la base de données artificiel.

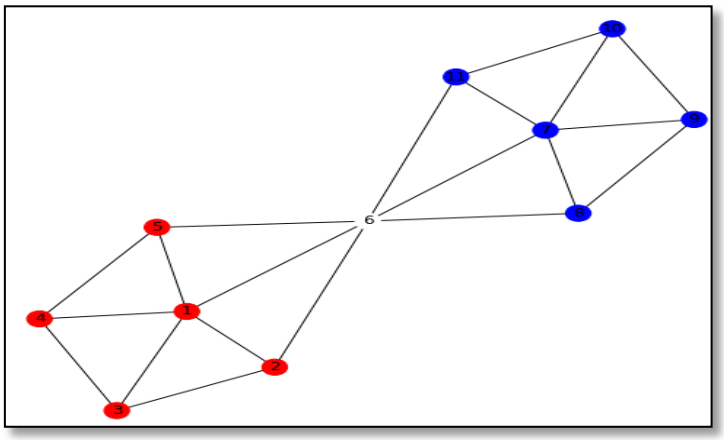

Figure 3.10 résultat d'algorithme après l'ajoute des arcs (6-8) et (6-11).

7.2 Tests sur des réseaux réelle

Dans cette partie, on va tester notre avec des réseaux réelle dont la structure de communautés est connue d'avance en comparant les résultats avec d'autres algorithmes.

Avant d'obtenir notre algorithme final, nous avons fait plusieurs tests pour trouver les valeurs qui nous donneraient les meilleurs résultats.

Les tableaux suivants nous montrent les tests que nous avons faits avant de décider le nombre de regroupement et la probabilité d'appartenance d'un nœud dans une communauté.

BDD(nbr des nœuds)	Différence des nœuds	Nombre de communauté avant proba	Probabilité	Nombre de communauté après proba	Nombre des nœuds chevauchants
Karaty(35 nœuds)	1	3	30	2	0
	1	3	40	2	0
	1	3	50	2	1
	1	3	60	2	2
	1	3	70	2	4
	2	2	30	2	1
	2	2	40	2	1
	2	2	50	2	1
	2	2	60	2	1
	2	**2**	**70**	**2**	**2**
	3	2	30	2	1
	3	2	40	2	1
	3	2	50	2	1
	3	2	60	2	1
	3	2	70	2	2
	4	2	30	2	1
	4	2	40	2	1
	4	2	50	2	1
	4	2	60	2	1
	4	2	70	2	5

Tableau 3.1 Résultat des tests de notre algorithme sur Karaty

BDD	Différence des nœuds	Nbr com avant prob	Prob	Nbr com après prob	Nbr nœuds chevauchants
Daulphins(62 nœuds)	1	27	30	13	17
	1	27	40	12	23
	1	27	50	11	27
	1	27	60	10	39
	1	27	70	9	48
	2	6	30	5	1
	2	6	40	5	2
	2	6	50	5	2
	2	6	60	5	8
	2	6	70	4	22
	3	6	30	5	1
	3	6	40	5	2
	3	6	50	5	2
	3	6	60	5	8
	3	6	70	4	22
	4	**3**	**30**	**3**	**1**
	4	3	40	3	1
	4	3	50	3	1
	4	3	60	3	3
	4	3	70	3	10

Tableau 3.2 Résultat des tests de notre algorithme sur Daulphins.

BDD(nbr des nœuds)	Différence des nœuds	Nombre de communauté avant proba	Probabilité	Nombre de communauté après proba	Nombre des nœuds chevauchants
Football(115)	1	112	30	0	115
	1	112	40	0	115
	1	112	50	0	115
	1	112	60	0	115
	1	112	70	0	115
	2	94	30	3	87
	2	94	40	3	89
	2	94	50	1	105
	2	94	60	1	111
	2	94	70	0	115
	3	27	30	12	10
	3	27	40	9	33
	3	27	50	8	36
	3	27	60	7	60
	3	27	70	6	83
	4	**24**	**30**	**10**	**17**
	4	24	40	10	19
	4	24	50	9	39
	4	24	60	8	68
	4	24	70	3	93

Tableau 3.3 Résultat des tests de notre algorithme sur Football

BDD(nbr des nœuds)	Différence des nœuds	Nombre de communauté avant proba	Probabilité	Nombre de communauté après proba	Nombre des nœuds chevauchants
Books(105)	1	40	30	11	30
	1	40	40	10	40
	1	40	50	10	40
	1	40	60	7	77
	1	40	70	4	89
	2	13	30	9	6
	2	13	40	8	11
	2	13	50	9	18
	2	13	60	8	33
	2	13	70	5	57
	3	6	30	5	4
	3	6	40	5	4
	3	6	50	5	5
	3	6	60	5	14
	3	6	70	5	33
	4	**6**	**30**	**6**	**7**
	4	6	40	6	7
	4	6	50	6	9
	4	6	60	6	22
	4	6	70	5	44

Tableau 3.4 Résultat des tests de notre algorithme sur Books

8. Résultat et discussion

Pour qu'un algorithme de détection de communautés soit considéré intéressant, il faut que les communautés qu'il trouve soient pertinentes, ce qui est souvent difficile à démontrer étant donné l'absence de définition précise de ce qu'est une bonne communauté. Pour cela, nous avons testé notre algorithme sur des réseaux réelle dans le nombre de communauté est déjà connu en comparant nos résultats avec ceux des algorithmes connus, soit Newman [45] , Edge Betweenness[11], Label Propagation [21] et l'agorithme de Louvain[2], et d'autres algorithmes récent, soit MCLC[40], C-means [47]et l'algorithme de Nedioui[48].

8.1 Club de karaté de Zachary

La première comparaison a fait sur le club de karaté de Zachary [33]. C'est un réseau très populaire et très utilisé par les algorithmes de détection de communauté qui contient 34 nœuds et 78 arêtes. Ce réseau comporte deux communautés.

La figure ci-dessus nous montre le résultat d'exécution de notre algorithme sur le réseau de Zachary(les nœuds blancs représentent les nœuds chevauchants).

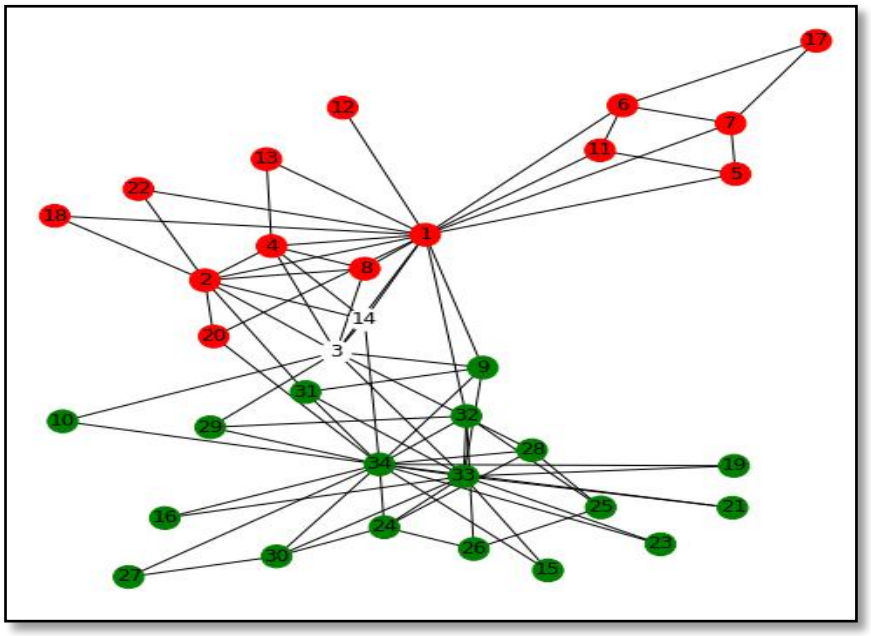

Figure 3.11 Structure de communautés trouvée par notre méthode pour le réseau de Zachary

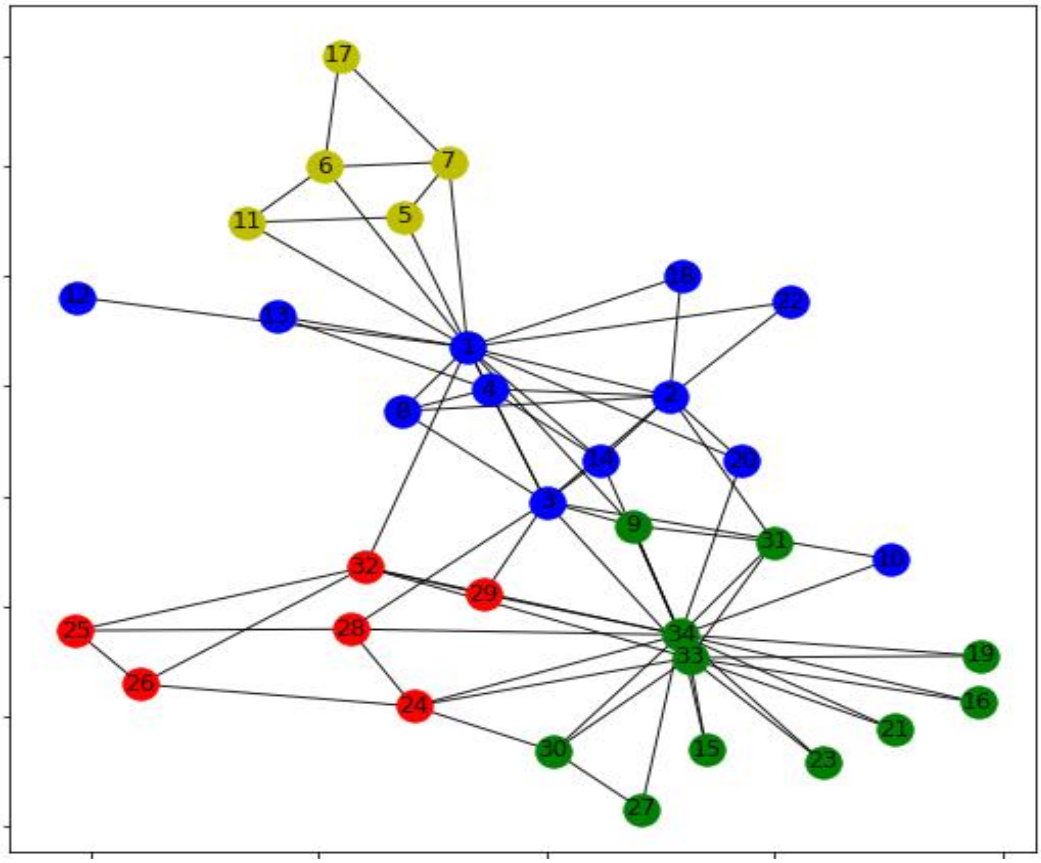

Figure 3.12 Structure de communautés trouvée par l'algorithme de Louvain pour le réseau de Zachary.

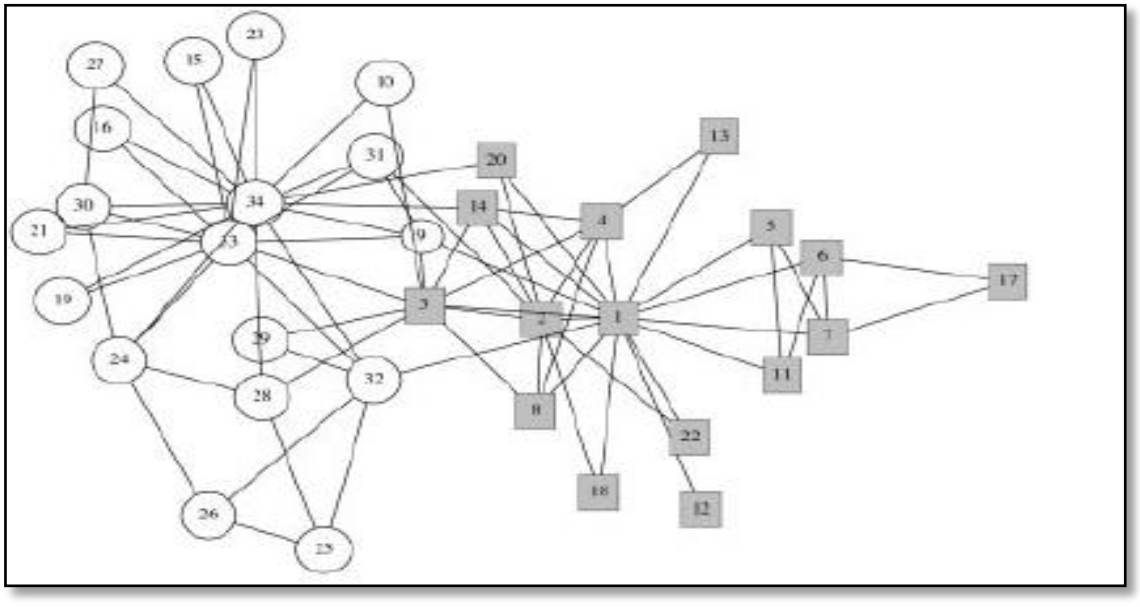

Figure 3.13 Structure de communautés trouvée par Girvan et Newman pour le réseau de Zachary

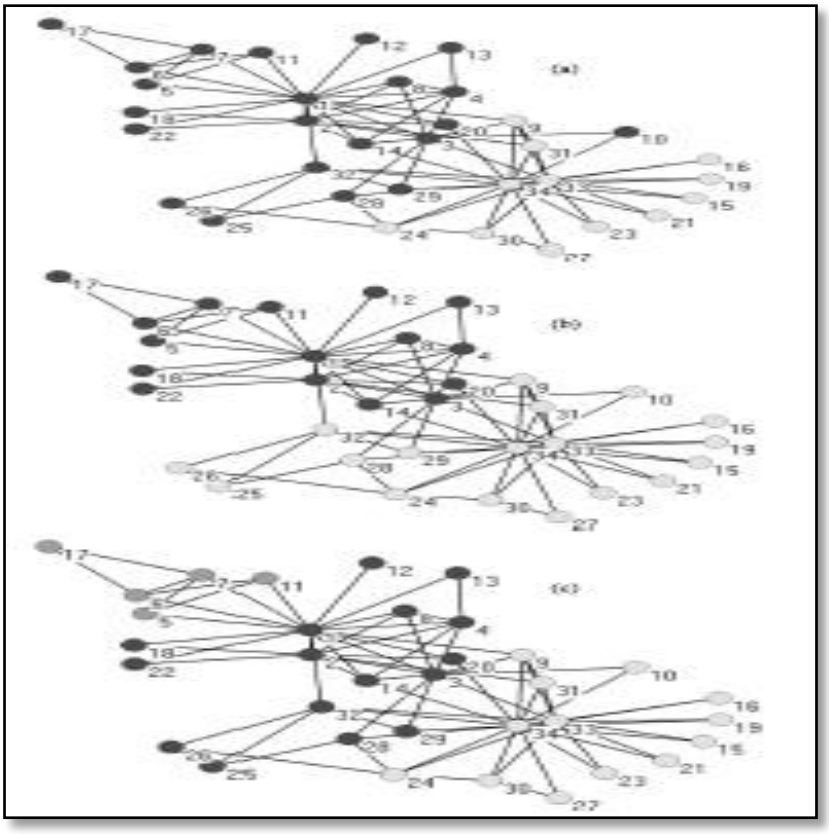

Figure 3.14 Différentes structures de communautés trouvées par Label Propagation pour le réseau de Zachary

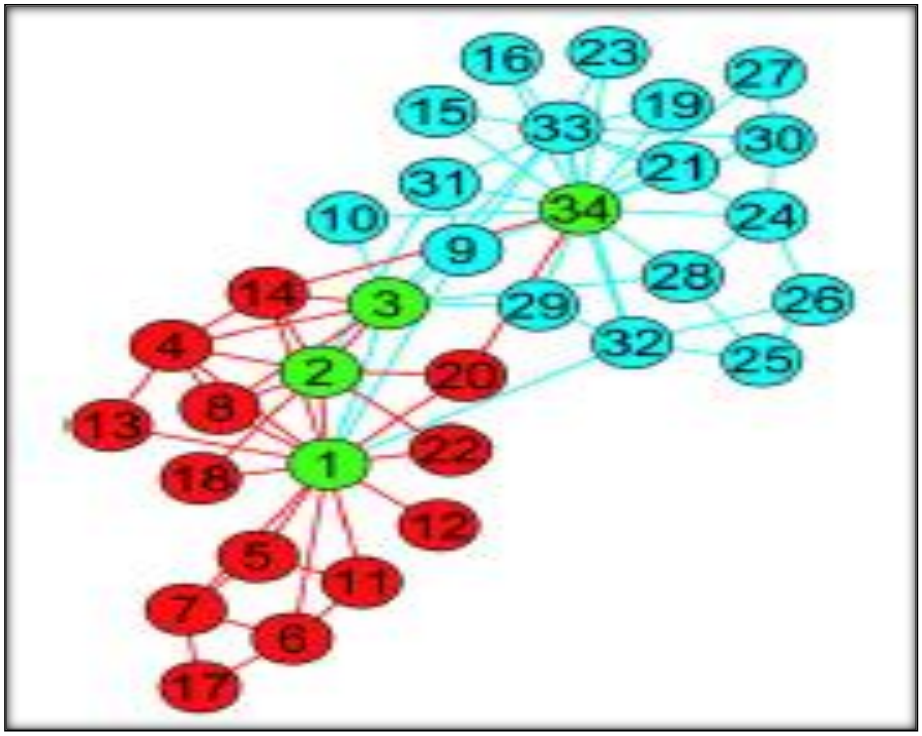

Figure 3.15 Deux communautés (Rouge et bleu) et 4 nœuds chevauchants trouvés par MCLC pour le réseau de Zachary

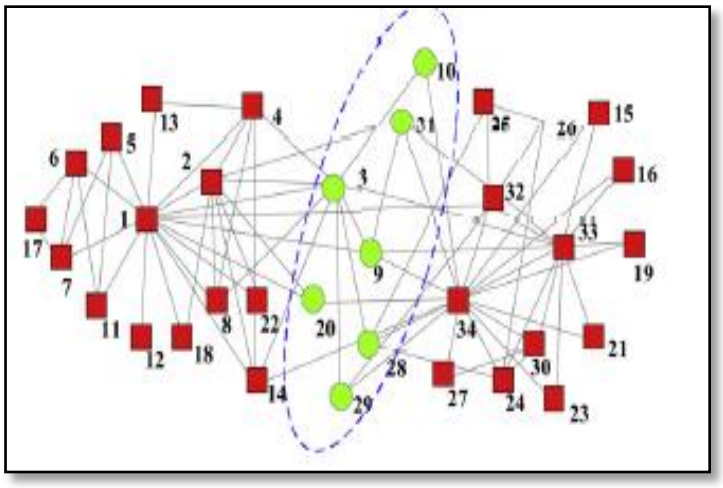

Figure 3.16 Deux communautés et 7 nœuds chevauchants trouver par C-means pour le réseau de Zachary

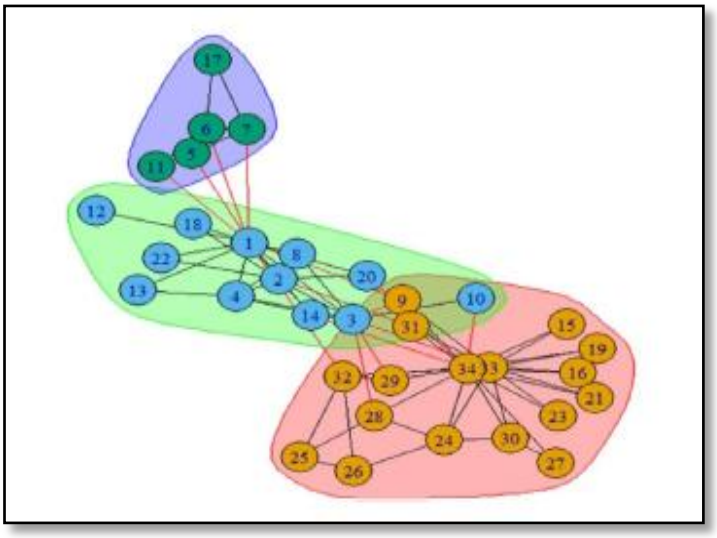

Figure 3.17 Structure de communautés trouvée par Nidioui pour le réseau de Zachary

Méthode	Nombre de communauté	Nombre des nœuds chevauchants
Méthode proposé	2	2
Newman	2	/
Edge Betweness	5	/
Louvain	4	/
Label Propagation	3	/
MCLC	2	4
C-means	2	7

Algorithme de Nidioui	3	4

Tableau 3.5 Résultat d'exécution des algorithmes sur le réseau de Zachary.

8.2 Les dauphins de Lusseau

La deuxième comparaison a fait sur le réseau de dauphins de Lusseau. Ce réseau contient 62 nœuds et 159 liens. La méthode proposée a détecté trois communautés avec des trois nœuds chevauchants. Ce réseau est constitué essentiellement de deux communautés.

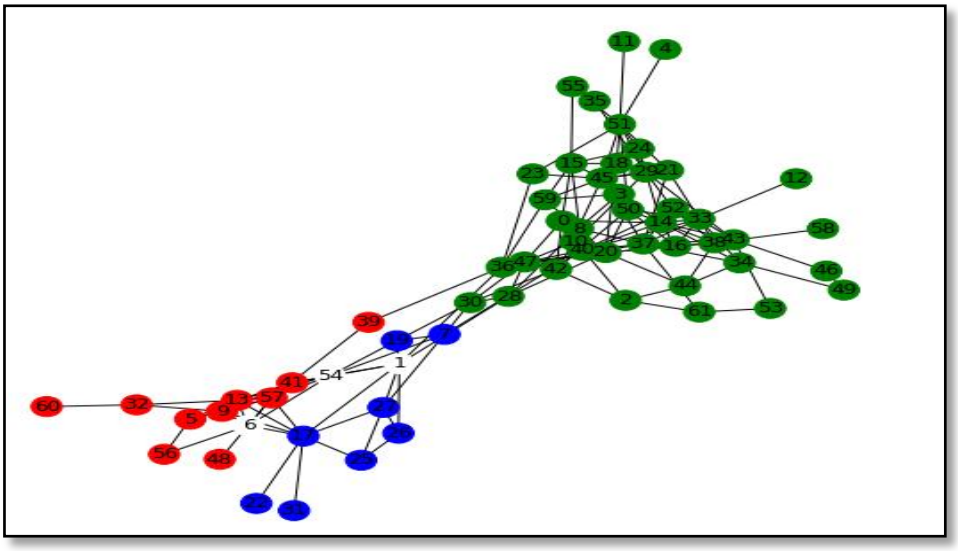

Figure 3.18 Structure de communautés trouvée par notre méthode pour le réseau de dauphins.

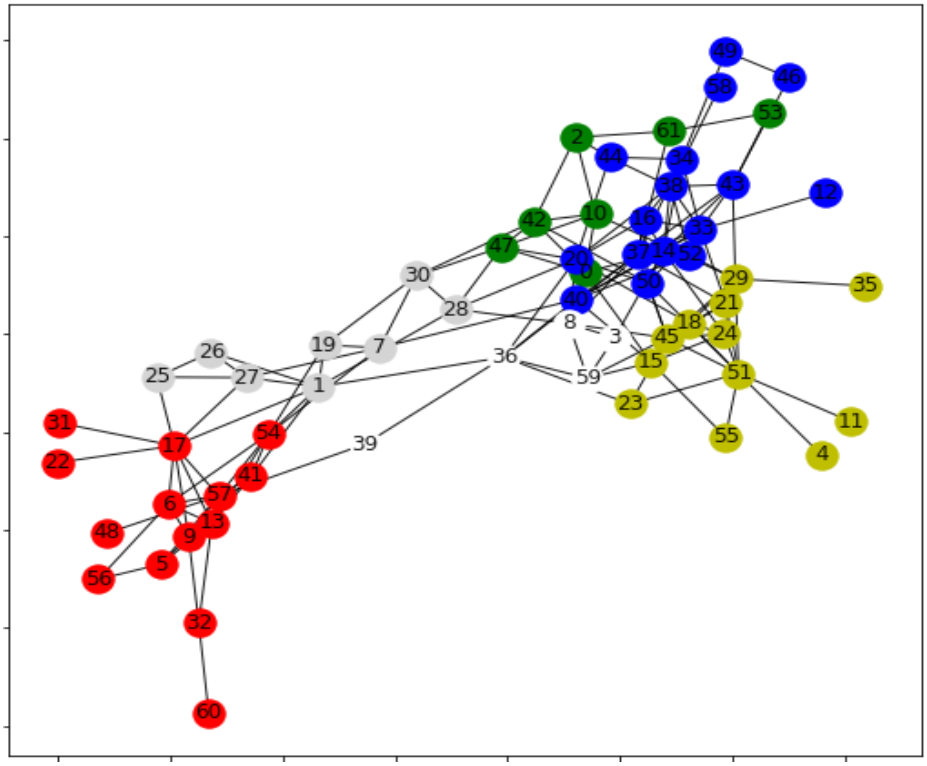

Figure 3.19 Structure de communautés trouvée par l'algorithme de Louvain pour le réseau des dauphins.

Méthode	Nombre de communauté	Nombre des nœuds chevauchants
Méthode proposé	3	3
Newman	5	/
Edge Betweness	6	/
Label Propagation	3	/
Louvain	6	/
MCLC	2	4
C-means	2	5
Algorithme de Nidioui	4	/

Tableau 3.6 Résultat d'exécution des algorithmes sur le réseau de dauphins.

8.3 Les livres politique

Une troisième comparaison que nous avons faite sur le réseau de livres politiques. Le réseau est constitué de 105 livres avec 441 liens se trouvant entre les livres achetés ensemble. Ce réseau a initialement trois groupes.

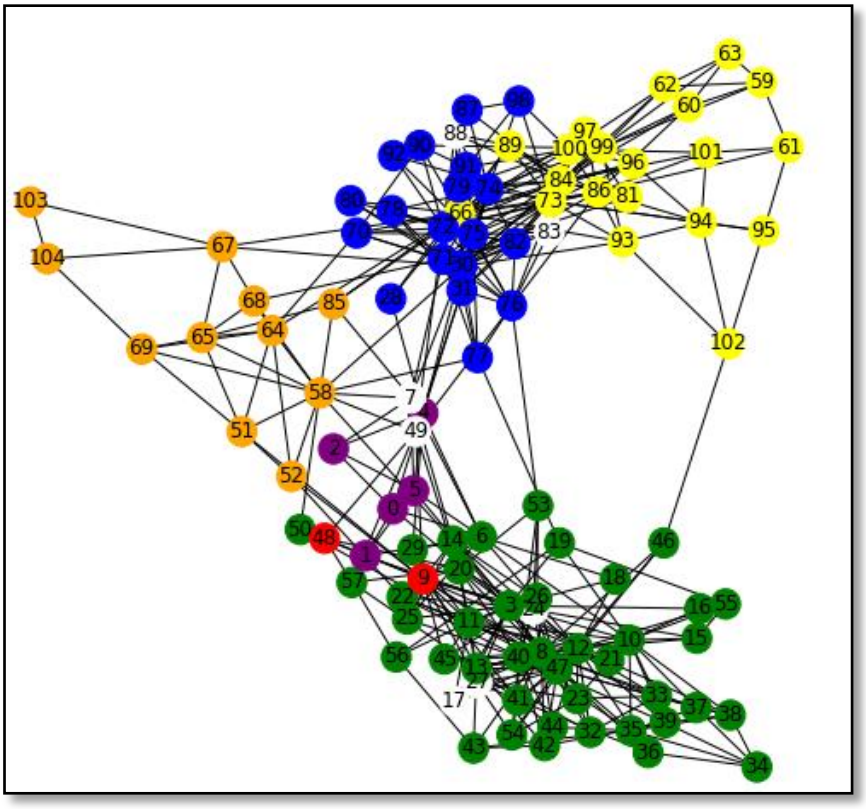

Figure 3.20 Structure de communautés trouvée par notre méthode pour le réseau livres politique

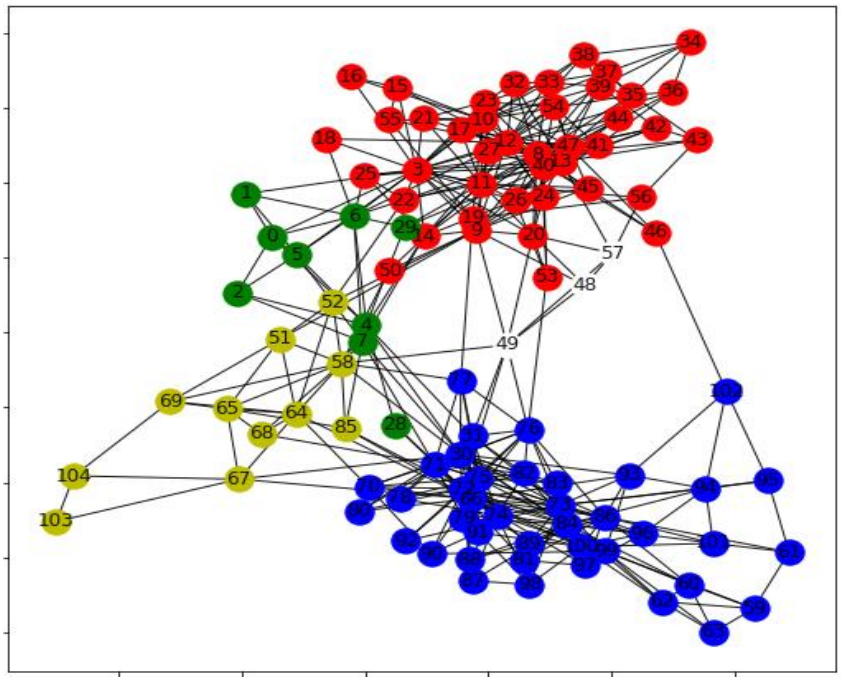

Figure 3.21. Structure des communautés trouvées par l'algorithme de Louvain pour le réseau des livres politiques.

Méthode	Nombre de communauté	Nombre des nœuds chevauchants
Méthode proposé	6	7
Newman	5	/
Edge Betweness	5	/
Label Propagation	3	/
Louvain	5	/
MCLC	/	/
C-means	/	/
Algorithme de Nidioui	4	/

Tableau 3.7 Résultat d'exécution des algorithmes sur le réseau livres politique

8.4 Football américain

Une autre comparaison que nous avons faite d'un réseau réel étudié dans le cadre de ce mémoire est le réseau de jeux du football américain. Ce réseau est constitué de douze communautés, 115 nœuds et 613 liens.

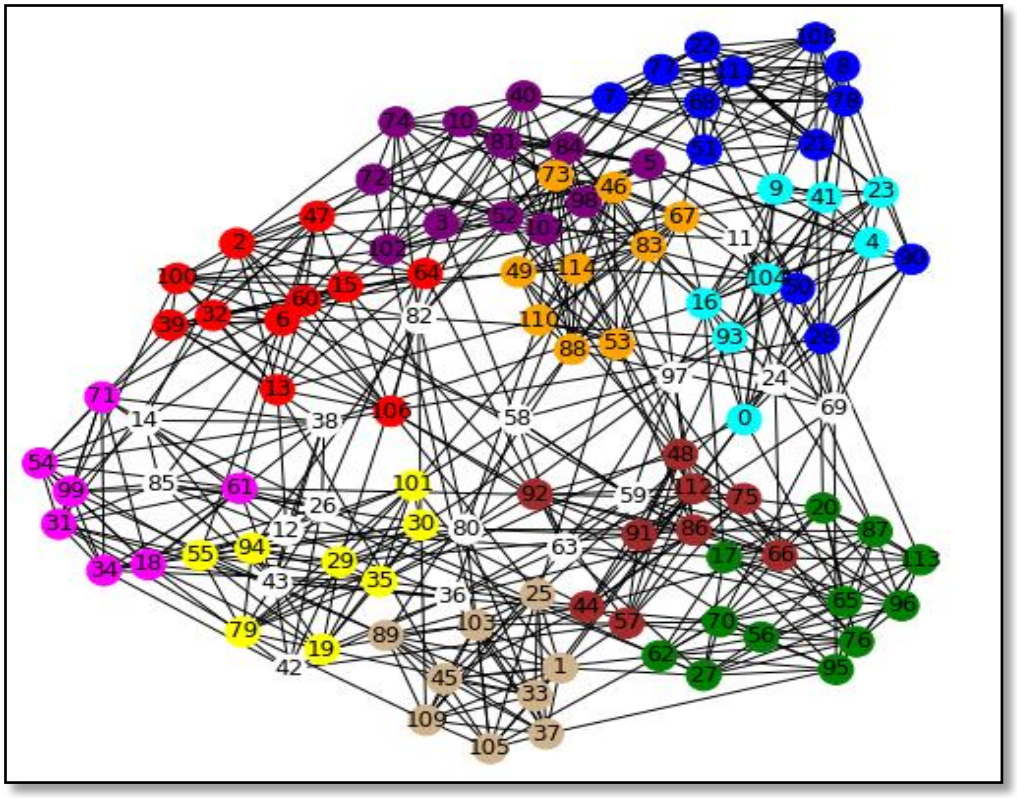

Figure 3.22 Structure de communautés trouvée par notre méthode pour le réseau football.

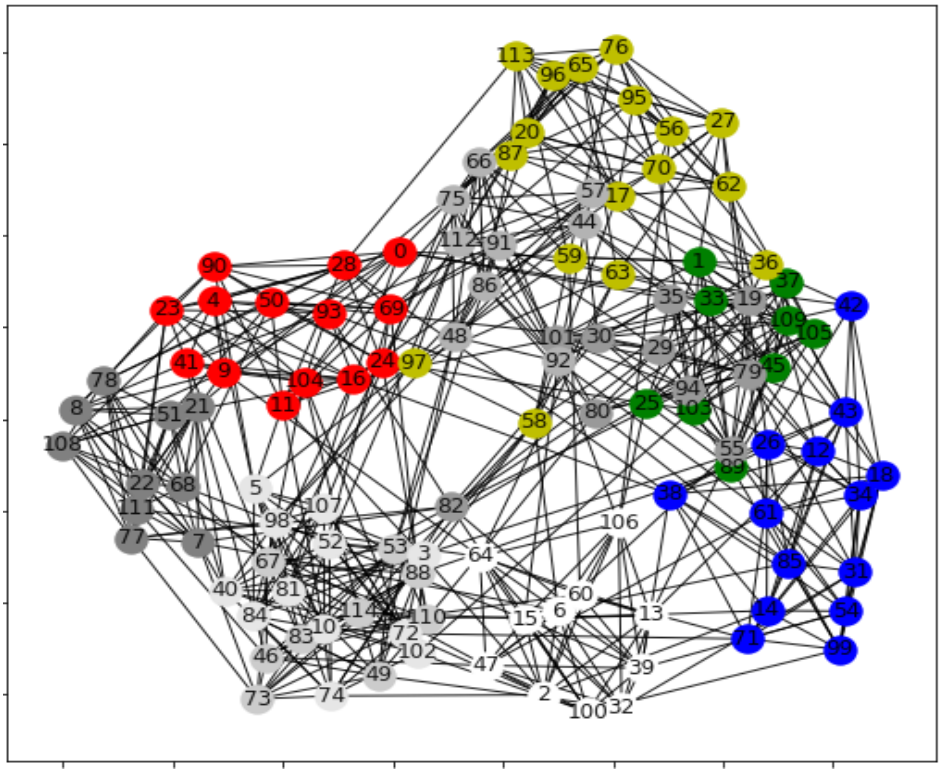

Figure 3.23 Structure de communautés trouvée par l'algorithme de Louvain pour le réseau football.

Méthode	Nombre de communauté	Nombre des nœuds chevauchants
Méthode proposé	10	17
Newman	9	/
Edge Betweness	12	/
Label Propagation	11	/
Louvain	8	/
MCLC	/	/
C-means	/	/
Algorithme de Nidioui	9	/

Tableau 3.8 Résultat d'exécution des algorithmes sur le réseau football

8.5 Ego-Network Facebook

Dans les réseaux sociaux, la fusion des petits réseaux est appelée Ego-Network est un ensemble des petites communautés où chaque nœuds est centrées sur un nœud[49].

Un autre test que nous avons fait est sur le réseau Facebook, qui est le résultat de fusion de dix Facebook Ego-network dans un seul réseau qui s'appelle Facebook. Ce réseau est constitué 4039 nœuds et 88234 liens [50].

Avant la troisième phase de la probabilité, notre algorithme à détecter 51 qui partagent un tels nombre des nœuds entre eux. Dans la troisième phase où nous avons calculé la probabilité d'appartenance des nœuds dans les communautés, le nombre des communautés a diminué à 9 communautés avec 9 nœuds chevauchant dans un temps d'exécution de 15 minutes.

Les résultats d'exécution étaient comme suite :

- nombre de communautés avant la probabilité : 51 communautés
- nombre de communautés après la probabilité : 9 communautés
- nombre des nœuds chevauchants : 9
- temps d'exécution : environ 15 minutes

La figure suivante nous montre le résultat d'exécution de notre algorithme sur les réseaux Facebook

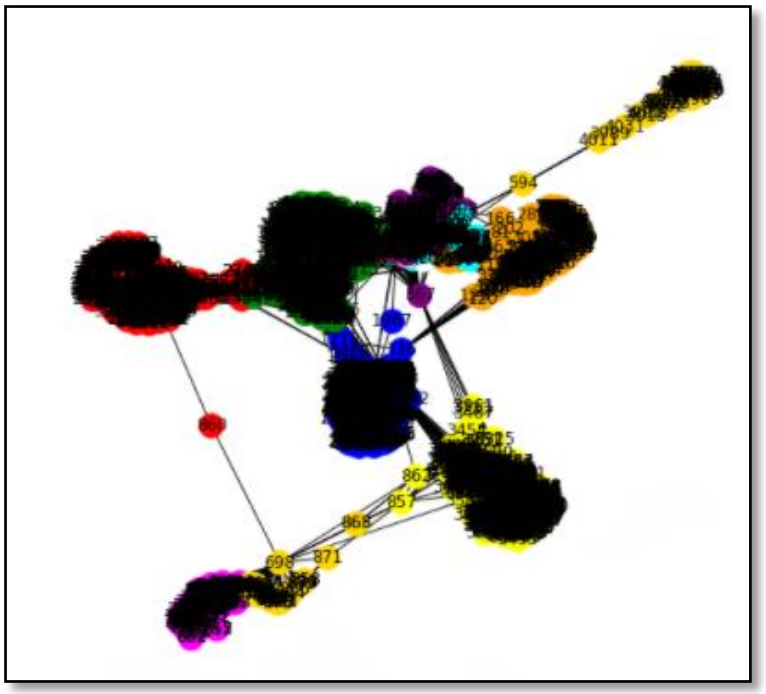

Figure 3.24 Structure de communautés trouvée par notre algorithme pour le réseau Facebook.

8.6 Complexité

Dans cette section, le calcul de la complexité n'était pas facile à calculer, parce que dans la deuxième phase le nombre de répétitions des regroupements dépendent aux nombres des communautés trouver dans la première phase et la relation entre ses communautés.

La complexité globale estimée de notre algorithme est égale à : **O (n^3 log m),** où m représente le nombre des arêtes et n représente le nombre des nœuds.

Le tableau suivant nous montre la comparaison des complexités de notre algorithme avec d'autres algorithmes connus dans la détection des communautés.

Méthodes	Complexité
Notre methode	O(n^3 log m)
Walktrap[books_2005]	O(n^4)
Clauset et Newman[50]	O(n^2 d log n)
Girvan et Newman [11]	O(n^5)
Fortunato[51]	O($n^{7)}$
Louvain[2]	O(m n^3)

Tableau 3.9 comparaison des complexités des algorithmes

9. Avantages et Inconvénients

En plus de la facilité de l'implémentation, nous avons montré que notre méthode est capable de détecter des communautés disjointes et chevauchantes comparable à celles que pouvaient trouver d'autres algorithmes de détection de communauté.

Les résultats d'exécution de notre algorithme restent toujours stables.

L'arrangement nous a aidés à inclure les petites communautés dans d'autres communautés appropriées.

En ce qui concerne le temps d'exécution, notre algorithme a fait preuve de sa performance par rapport aux d'autres algorithme de détection de communautés.

10. Conclusion

Dans ce chapitre, nous avons tout d'abord présenté les outils que nous avons utilisés pour développer notre application. Ensuite, nous avons abordé un schéma général et les fonctionnalités de notre système, et finalement nous avons exposé les résultats des expérimentations que nous avons réalisées après l'analyse des test que nous avons fait avant d'obtenir les meilleurs regroupement et la valeur de la probabilité d'appartenance d'un nœud dans une communauté.

Conclusion et perspective

Ce mémoire avait comme objectif la détection de communauté chevauchante dans les réseaux sociaux. Pour cela, le premier chapitre a passé en revue les approches de l'état de l'art pour la détection des communautés qui se chevauchent. Nous commençant l'état de l'art avec une représentation des communautés et quelques définitions essentielles dans la détection des communautés. Ensuite nous avons abordé les différentes catégories des algorithmes de la détection de communauté en décrivant les travaux les plus récents dans le domaine.

Nous avons présenté une nouvelle approche de détection de communautés chevauchantes qui se concentre principalement sur le principe de voisinage et la probabilité.

Notre méthode est simple et facile à comprendre, elle est capable de détecter les communautés disjointes dans lesquelles les nœuds de chaque communauté sont fortement liés entre eux et de détecter les communautés chevauchantes qui peuvent partager un ou plusieurs nœuds entre eux.

D'après les tests que nous avons réalisés sur des réseaux artificiels et d'autres réseaux réels, notre algorithme montre l'efficacité des regroupements que nous avons réalisés à détecter les communautés disjointe et l'efficacité de la probabilité à détecter les nœuds chevauchants avec une probabilité d'appartenance dans les communautés entre 0% et 100%.

La comparaison des résultats de notre approche avec d'autres montre que notre algorithme donne des bons résultats.

Les communautés trouvées par notre méthode restent stables dans plusieurs exécutions sur le même réseau et la mise en œuvre de la méthode proposée est très facile.

En ce qui concerne le temps d'exécution, notre algorithme a fait preuve de sa performance par rapport aux d'autres algorithme de détection de communautés.

Nos buts de future sont :

- Optimiser l'algorithme pour obtenir des résultats identiques ou meilleurs.
- Optimiser le temps d'exécution.
- Ajouter une méthode pour évaluer les communautés résultantes et liées notre méthode avec l'intelligence artificielle pour changer le nombre des regroupements utilisé et la probabilité d'appartenance d'un nœud dans une communauté.

Bibliographie

[1] W.Louafi and N. Louafi, “Analyse des réseaux sociaux et détection de communautés.” these Master,Univ 08 Mai 45, Guelma, 2019.

[2] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *J. Stat. Mech. Theory Exp.*, vol. 2008, no. 10, 2008, doi: 10.1088/1742-5468/2008/10/P10008.

[3] M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 69, no. 2 2, pp. 1–15, 2004, doi: 10.1103/PhysRevE.69.026113.

[4] W. Luo, Z. Yan, C. Bu, and D. Zhang, “Community Detection by Fuzzy Relations,” *IEEE Trans. Emerg. Top. Comput.*, vol. 6750, no. 1, 2017, doi: 10.1109/TETC.2017.2751101.

[5] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, “Uncovering the overlapping community structure of complex networks in nature and society,” *Nature*, vol. 435, no. 7043, pp. 814–818, 2005, doi: 10.1038/nature03607.

[6] H. Shen, X. Cheng, K. Cai, and M. Bin Hu, “Detect overlapping and hierarchical community structure in networks,” *Phys. A Stat. Mech. its Appl.*, vol. 388, no. 8, pp. 1706–1712, 2009, doi: 10.1016/j.physa.2008.12.021.

[7] A. Kumar, D. Barman, R. Sarkar, and N. Chowdhury, “Overlapping Community Detection Using Multiobjective Genetic Algorithm,” *IEEE Trans. Comput. Soc. Syst.*, pp. 1–16, 2020, doi: 10.1109/TCSS.2020.2989295.

[8] Z. Liu, B. Xiang, W. Guo, Y. Chen, K. Guo, and J. Zheng, “Overlapping Community Detection Algorithm Based on Coarsening and Local Overlapping Modularity,” *IEEE Access*, vol. 7, pp. 57943–57955, 2019, doi: 10.1109/ACCESS.2019.2912182.

[9] C. Dawson and C. Dawson, “Social network analysis,” *A–Z Digit. Res. Methods*, pp. 356–361, 2019, doi: 10.4324/9781351044677-54.

[10] S. Fortunato, “Community detection in graphs,” *Phys. Rep.*, vol. 486, no. 3–5, pp. 75–174, 2010, doi: 10.1016/j.physrep.2009.11.002.

[11] M. Girvan and M. E. J. Newman, “Community structure in social and biological networks,” vol. 99, no. 12, 2002.

[12] A. Lancichinetti, S. Fortunato, and F. Radicchi, “Benchmark graphs for testing community detection algorithms,” *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 78, no. 4, pp. 1–5, 2008, doi: 10.1103/PhysRevE.78.046110.

[13] L. Ni, W. Luo, W. Zhu, and B. Hua, “Local overlapping community detection,” *ACM Trans. Knowl. Discov. Data*, vol. 14, no. 1, 2019, doi: 10.1145/3361739.

[14] R. Alhajj and J. Rokne, *Encyclopedia of social network analysis and mining*. Springer Publishing Company, Incorporated, 2014.

[15] J. Baumes, M. Goldberg, M. Krishnamoorthy, M. Magdon-Ismail, and N. Preston, “Finding communities by clustering a graph into overlapping subgraphs,” *Int. Conf. Appl. Comput. (IADIS 2005)*, pp. 97–104, 2005.

[16] S. Jabbour, N. Mhadhbi, B. Radaoui, and L. Sais, “Detecting Highly Overlapping Community Structure by Model-based Maximal Clique Expansion,” *Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018*, vol. 2, pp. 1031–1036, 2019, doi: 10.1109/BigData.2018.8621868.

[17] Z. Sun, B. Wang, J. Sheng, Z. Yu, and J. Shao, “Overlapping community detection based on information dynamics,” *IEEE Access*, vol. 6, pp. 70919–70934, 2018, doi: 10.1109/ACCESS.2018.2879648.

[18] X. Zhou, Y. Liu, J. Wang, and C. Li, “A density based link clustering algorithm for overlapping community detection in networks,” *Phys. A Stat. Mech. its Appl.*, vol. 486, pp. 65–78, 2017, doi: 10.1016/j.physa.2017.05.032.

[19] T. S. Evans and R. Lambiotte, “Line graphs, link partitions, and overlapping communities,” *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 80, no. 1, pp. 1–8, 2009, doi: 10.1103/PhysRevE.80.016105.

[20] A. C. Gabardo, R. Berretta, and P. Moscato, “M-Link: a link clustering memetic algorithm for overlapping community detection,” *Memetic Comput.*, vol. 12, no. 2, pp. 87–99, 2020, doi: 10.1007/s12293-020-00300-x.

[21] U. N. Raghavan, R. Albert, and S. Kumara, “Near linear time algorithm to detect community structures in large-scale networks,” *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 76, no. 3, pp. 1–11, 2007, doi: 10.1103/PhysRevE.76.036106.

[22] M. Lu, Z. Zhang, Z. Qu, and Y. Kang, “LPANNI: Overlapping Community Detection Using Label Propagation in Large-Scale Complex Networks,” *IEEE Trans. Knowl. Data Eng.*, vol. 4347, no. c, pp. 1–14, 2018, doi: 10.1109/TKDE.2018.2866424.

[23] H. Sun, J. Liu, J. Huang, G. Wang, X. Jia, and Q. Song, “LinkLPA: A Link-Based Label Propagation Algorithm for Overlapping Community Detection in Networks,” *Comput. Intell.*, vol. 33, no. 2, pp. 308–331, 2017, doi: 10.1111/coin.12087.

[24] X. Ma and D. Dong, “Evolutionary Nonnegative Matrix Factorization Algorithms for Community Detection in Dynamic Networks,” *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 1045–1058, 2017, doi: 10.1109/TKDE.2017.2657752.

[25] B. Kernighan and S. Lin, “An effective heuristic procedure for partitioning graphs,” *Bell Syst. Tech. J.*, pp. 291–308, 1970.

[26] A. Pothenf and H. D. Simon, “With Eigenvectors of Graphs *,” vol. 11, no. 3, pp. 430–452, 1990.

[27] H. Van Lierde, G. Chen, and T. W. S. Chow, “Scalable Spectral Clustering for Overlapping Community Detection in Large-Scale Networks,” *IEEE Trans. Knowl. Data Eng.*, vol. PP, no. c, pp. 1–1, 2019, doi: 10.1109/tkde.2019.2892096.

[28] A. Lancichinetti, S. Fortunato, and J. Kertész, “Detecting the overlapping and hierarchical community structure in complex networks,” *New J. Phys.*, vol. 11, 2009, doi: 10.1088/1367-2630/11/3/033015.

[29] C. Lee, F. Reid, A. McDaid, and N. Hurley, “Detecting highly overlapping community structure by greedy clique expansion,” vol. 10, 2010.

[30] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pp. 701–710, 2014, doi: 10.1145/2623330.2623732.

[31] J. Tang and M. Qu, “P1067-Tang,” pp. 1067–1077.

[32] Q. Chen and L. Wei, “Overlapping community detection of complex network: A survey,” *Proc. - 2019 20th Int. Conf. Parallel Distrib. Comput. Appl. Technol. PDCAT 2019*, pp. 513–516, 2019, doi: 10.1109/PDCAT46702.2019.00102.

[33] W. W. Zachary, “An Information Flow Model for Conflict and Fission in Small Groups,” *J. Anthropol. Res.*, vol. 33, no. 4, pp. 452–473, 1977, doi: 10.1086/jar.33.4.3629752.

[34] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, “The bottlenose dolphin community” *Behav. Ecol. Sociobiol.*, vol. 54, no. 4, pp. 396–405, 2003, doi: 10.1007/s00265-003-0651-y.

[35] D. Mehrle, A. Strosser, and A. Harkin, “Walk-modularity and community structure in networks,” *Netw. Sci.*, vol. 3, no. 3, pp. 348–360, 2015, doi: 10.1017/nws.2015.20.

[36] A. F. McDaid, D. Greene, and N. Hurley, “Normalized Mutual Information to evaluate overlapping community finding algorithms,” pp. 1–3, 2011.

[37] K. Erciyes, *Complex Networks*. 2018.

[38] S. Gregory, “Finding overlapping communities in networks by label propagation,” *New J. Phys.*, vol. 12, 2010, doi: 10.1088/1367-2630/12/10/103018.

[39] J. Xie, B. K. Szymanski, and X. Liu, “SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process,” *Proc. - IEEE Int. Conf. Data Mining, ICDM*, pp. 344–349, 2011, doi: 10.1109/ICDMW.2011.154.

[40] S. Gupta and P. Kumar, “An overlapping community detection algorithm based on rough clustering of links,” *Data Knowl. Eng.*, vol. 125, p. 101777, 2020, doi: 10.1016/j.datak.2019.101777.

[41] M. Xu, Y. Li, R. Li, F. Zou, and X. Gu, “EADP: An extended adaptive density peaks clustering for overlapping community detection in social networks,” *Neurocomputing*, vol. 337, pp. 287–302, 2019, doi: 10.1016/j.neucom.2019.01.074.

[42] J. Yang and J. Leskovec, “Overlapping community detection at scale: A nonnegative matrix factorization approach,” *WSDM 2013 - Proc. 6th ACM Int. Conf. Web Search Data Min.*, pp. 587–596, 2013, doi: 10.1145/2433396.2433471.

[43] A. Kadiyala and A. Kumar, “Applications of Python to evaluate environmental data science problems,” *Environ. Prog. Sustain. Energy*, vol. 36, no. 6, pp. 1580–1586, 2017, doi: 10.1002/ep.12786.

[44] D. Schult, “LA-UR- EXPLORING NETWORK STRUCTU Exploring network

structure , dynamics , and function using NetworkX," vol. 836, 1943.

[45] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," vol. 066133, no. September 2003, pp. 1–5, 2004, doi: 10.1103/PhysRevE.69.066133.

[46] X. Deng, G. Li, M. Dong, and K. Ota, "Finding overlapping communities based on Markov chain and link clustering," pp. 411–420, 2017, doi: 10.1007/s12083-016-0457-0.

[47] Y. Lei, Y. Zhou, and J. Shi, "Overlapping communities detection of social network based on hybrid C- means clustering algorithm," *Sustain. Cities Soc.*, vol. 47, no. December 2018, p. 101436, 2019, doi: 10.1016/j.scs.2019.101436.

[48] M. Abdelhamid, A. Moussaoui, and B. Saoud, "Detecting communities in social networks based on cliques," *Physica A*, vol. 551, p. 124100, 2020, doi: 10.1016/j.physa.2019.124100.

[49] Y. Li, C. Sha, X. Huang, and Y. Zhang, "Community detection in attributed graphs: An embedding approach," *32nd AAAI Conf. Artif. Intell. AAAI 2018*, pp. 338–345, 2018.

[50] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.*, vol. 70, no. 6, p. 6, 2004, doi: 10.1103/PhysRevE.70.066111.

[51] S. Fortunato, V. Latora, and M. Marchiori, "Method to find community structures based on information centrality," *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.*, vol. 70, no. 5, p. 13, 2004, doi: 10.1103/PhysRevE.70.056104.

Webographie

[W1] **https ://www.python.org/downloads/**

[W2] https ://www.anaconda.com/products/individual