UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

Département d'informatique et d'ingénierie

Une nouvelle approche de détection de communautés dans les réseaux sociaux

Mémoire (INF 6021) pour l'obtention du grade de Maîtrise en sciences et technologies de l'information

Par

Mohamed Talbi

Août 2013

Résumé

Un réseau social comme Facebook ou Twitter est un ensemble d'acteurs sociaux, tels des individus ou des organisations, reliés entre eux par des connexions représentant des interactions sociales. Il décrit une structure sociale dynamique par un ensemble de sommets et d'arêtes. L'analyse des réseaux sociaux, fondée principalement sur la théorie des graphes et l'analyse sociologique, vise à étudier diverses facettes de ces réseaux dont les principales sont : la détection de communautés, l'identification d'acteurs influents ainsi que l'étude et la prédiction de l'évolution des réseaux. La détection de communautés consiste à former des groupes (disjoints ou chevauchants) de sorte que les nœuds au sein d'un même groupe soient connectés d'une manière dense. Dans le cas particulier de communautés disjointes, cela signifie aussi que les liens entre groupes sont faibles.

Dans le cadre de ce mémoire de maîtrise, nous présentons une nouvelle méthode de détection de communautés qui ne nécessite pas la connaissance apriori du nombre de communautés et vise à réduire considérablement le nombre d'itérations à effectuer sur un réseau initial en éliminant plusieurs liens (au lieu d'un seul lien) inter-communautés au cours d'une même itération. La méthode comporte deux phases et exploite la covariance des liens entre les nœuds et l'inertie inter-classes pour identifier les arcs à éliminer. Les communautés sont identifiées par maximisation de la modularité.

Une analyse empirique de notre approche et de quatre autres méthodes connues dans la littérature montre l'efficacité de notre approche à détecter des communautés même dans des graphes complexes où les groupes ne sont pas aisément identifiables. Les tests ont été menés sur des réseaux tant réels que synthétiques de diverses tailles et configurations.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Contexte

Un réseau social comme Facebook ou Twitter est un ensemble d'acteurs sociaux, tels des individus ou des organisations, reliés entre eux par des connexions représentant des interactions sociales. Il décrit une structure sociale dynamique par un ensemble de sommets et d'arêtes. L'analyse des réseaux sociaux, fondée principalement sur la théorie des graphes et l'analyse sociologique, vise à étudier diverses facettes de ces réseaux dont les principales sont : la détection de communautés, l'identification d'acteurs influents, l'étude et la prédiction de l'évolution des réseaux. Dans le cadre de ce mémoire de maîtrise, nous nous intéressons à la détection de communautés.

La notion de réseau existe dans plusieurs domaines de recherche en informatique et même dans d'autres disciplines. La modélisation des réseaux par des graphes facilite l'étude et la compréhension de leur structure en faisant appel à la théorie des graphes. Les graphes sont constitués de nœuds et de liens avec possibilité d'orientation des arcs et de labels (étiquettes) et poids au niveau des nœuds et des liens. En biologie par exemple, il existe les réseaux métaboliques dont les nœuds sont des protéines et les liens sont les interactions chimiques entre elles [For10, SLM$^+$04]. En sociologie, les nœuds sont des individus ou entités sociales (associations, entreprises, pays, etc.) et les liens entre eux sont de différentes natures qui varient selon le type du réseau. Dans les réseaux de connaissances, deux individus sont liés s'ils se connaissent. Dans les réseaux de collaboration, deux individus sont liés s'ils travaillent ensemble [New01]. Dans le cas des réseaux d'échanges, deux nœuds sont reliés entre eux, dès qu'ils échangent un courrier électronique [For10, KRKSSR02].

En général, la densité des liens entre les nœuds du réseau varie d'une zone à une

autre, ce qui implique l'existence de groupes de nœuds fortement connectés entre eux mais faiblement reliés aux autres nœuds du réseau. Ces zones, appelées communautés, sont des ensembles de nœuds connexes dont la densité des liens est plus forte que dans le graphe dans son entier tel qu' illustré par l'exemple simple de la figure 1.1 [GN02].

Le problème d'identification de communautés est important puisqu'il peut être rencontré dans plusieurs domaines d'application et des situations du monde réel. À titre d'exemple, les réseaux sociaux peuvent dévoiler des communautés représentant des individus ayant des intérêts communs ou de fortes connexions entre eux. De ce fait, on peut prédire la consommation ou le comportement des individus en analysant les achats et les comportements des autres éléments de la même communauté. L'identification des communautés nous permet également de déterminer le rôle de différents acteurs au sein des communautés et dans le réseau dans sa globalité. Le nœud avec une position centrale dans sa communauté, comme un nœud qui partage un grand nombre de liens avec les autres nœuds de la même communauté, peut exercer un important rôle de contrôle et assurer la stabilité au sein du réseau. Notons que les nœuds situés sur les frontières entre les communautés jouent un rôle important de médiation et de contrôle de communications entre les communautés [Cse09].

Par ailleurs, il existe plusieurs méthodes de détection de communautés. Ces méthodes se classent en trois catégories [For10, PKVS12, Fer12]. La première contient les méthodes de classification hiérarchiques qui permettent de choisir une structure de communauté parmi plusieurs niveaux hiérarchiques représentant différentes structures possibles. La deuxième catégorie quant à elle englobe les méthodes d'optimisation d'une fonction objective, qui identifient les communautés en maximisant une fonction de qualité. Finalement, la dernière catégorie porte sur les méthodes à base du modèle, dont des formalismes (définis à l'avance) s'appliquent sur les nœuds du réseau d'une manière itérative jusqu'à avoir une structure de communautés stable.

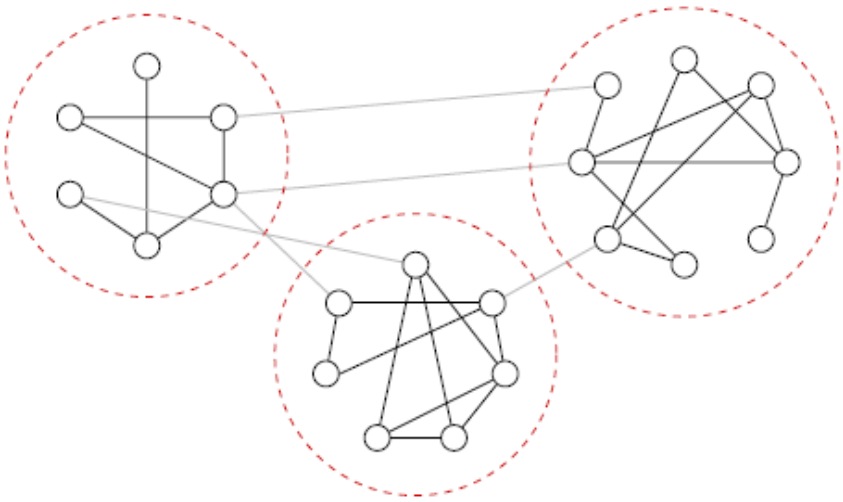

FIGURE 1.1 – Un réseau constitué de 3 communautés.

1.2 Mise en contexte et problématique

Soit G un graphe non orienté et non pondéré. G est défini par $G = (V, E)$, où V représente l'ensemble des nœuds et E correspond à l'ensemble des liens qui relient les différents nœuds de G. Le but de la détection de communauté est de trouver une partition $P = C_1, ..., C_k$ de k communautés de l'ensemble des nœuds de V. Chaque communauté C_i représente un sous-groupe de nœuds qui sont fortement connectés plus qu'ailleurs dans le graphe G. Il convient de noter que la valeur de k est inconnue et doit être identifiée automatiquement.

Parmi les travaux sur la détection de communautés, signalons l'étude de Girvan et Newman [For10] laquelle propose une approche permettant de déterminer les communautés à travers l'élimination itérative des liens se trouvant entre elles. Nous verrons dans le chapitre 2 que ces liens ont des mesures de centralité d'intermédiarité élevés. Le problème de cette approche est que le nombre de communautés k doit être connu à l'avance. Cependant, dans plusieurs applications, la valeur de k est inconnue. Afin de pallier à ce problème, Girvan et Newman ont proposé [New04] une approche itérative qui consiste à utiliser une fonction objective Q, appelée la modularité. Cette fonction permet de guider la recherche de la partition P. Spécifiquement, la modularité mesure pour chaque partition possible P une valeur $Q(P)$ qui fournit un indice sur la qualité de la partition générée. La maximisation de Q permet d'identifier la meilleure structure de communautés

dans un réseau donné. Toutefois, plusieurs autres travaux se basant sur la modularité ont été proposés [PL05, OL09]. La plupart de ces approches utilisent différentes techniques pour optimiser la modularité.

Ainsi, la problématique associée à notre étude concerne d'une part, le fait que certains travaux de recherche supposent que le nombre de communautés est fixé à priori et d'autre part, certains algorithmes effectuent un nombre impressionnant d'itérations pour éliminer les liens susceptibles d'être des relations inter-communautés. En outre, nous avons constaté que lorsque le nombre de liens inter-communautés est faible par rapport au nombre de liens intra-communauté, les algorithmes de détection de communautés finissent par identifier la même partition finale. Toutefois, la précision des résultats des divers algorithmes varie avec l'augmentation du nombre de liens inter-communautés et du paramètre de mixage (*mixing parameter*) lequel représente le ratio moyen des liens qui connectent un nœud à ceux d'autres communautés.

1.3 But

Le but de cette recherche est le développement d'une nouvelle approche de détection de communautés qui serait stable, précise et efficace même pour des réseaux complexes avec des liens inter-communautés élevés. Pour cela, nous avons défini une nouvelle méthode qui fonctionne en deux phases. Durant la première phase, nous utilisons l'inertie interclasses pour la détermination et l'élimination des liens inter-communautés en peu d'itérations. Dans la deuxième phase, nous proposons une procédure itérative ayant pour objectif la maximisation de la modularité, ce qui permet d'identifier les différentes communautés. L'approche proposée est évaluée sur différents types de réseaux (réels et synthétiques) en variant le nombre de liens inter-communautés. La performance de l'approche proposée est comparée à celle de quatre autres algorithmes de détection de communautés.

Le reste de ce mémoire est organisé comme suit : dans le deuxième chapitre, différents algorithmes de détection de communautés sont présentés. L'approche proposée est présentée dans le Chapitre 3. Le Chapitre 4 présente les résultats expérimentaux. Enfin, le Chapitre 5 conclut le mémoire.

Chapitre 2

État de l'art

Ce chapitre présente une brève revue de littérature sur la détection de communautés. Comme il existe de nombreuses approches proposées, nous allons retenir celles ayant reçu le plus d'intérêt de la part de la communauté scientifique. Ces approches illustrent aussi la diversité de méthodologies et donnent une vue d'ensemble des techniques proposées selon leurs principes méthodologiques [For10, PKVS12, LF09b, DDGDA05]. Nous commençons par présenter les méthodes hiérarchiques pour ensuite passer à celles relatives à l'optimisation d'une fonction objective. Ensuite, nous portons notre attention sur les méthodes à base de modèle. Avant de procéder à l'examen de différentes méthodes, nous allons clarifier la relation entre le problème de la détection de communautés et le partitionnement de graphes [PKVS12]. En effet, le partitionnement de graphes consiste à regrouper les nœuds d'un graphe en un nombre généralement prédéterminé de sous-groupes homogènes en minimisant le nombre de liens entre les différents groupes alors que la détection de communautés effectue la même opération avec ou sans exiger la connaissance à priori du nombre de communautés.

2.1 Les algorithmes de classification hiérarchique

Les algorithmes de classification hiérarchique cherchent à regrouper les nœuds d'un réseau dans différentes communautés, de telle sorte que les nœuds d'une même communauté se ressemblent le plus possible alors que les nœuds de communautés différentes sont les plus différents possible. Dans ce contexte, nous distinguons deux approches distinctes à savoir les algorithmes agglomératifs et les algorithmes divisifs.

2.1.1 Les algorithmes agglomératifs

L'approche agglomérative part d'une structure dans laquelle chaque nœud du graphe représente une communauté. Nous avons initialement n communautés (où n est le nombre de nœuds). On commence par calculer les distances entre les communautés et fusionner les deux communautés les plus proches pour former une nouvelle communauté. À chaque étape, on recalcule toutes les distances entre les communautés et on fusionne deux communautés. Lorsqu'il n'y a qu'une seule communauté représentant le graphe entier, il n'existe plus de distance à calculer. Tel qu'illustré par la figure 2.1 [PL05], les différentes étapes de ce processus peuvent être représentées par une forme arborescente appelée dendrogramme. Les feuilles sont les communautés avec un seul nœud et la racine représente le graphe entier. Dans ce qui suit, nous allons nous baser sur l'algorithme à marches aléatoires *Walktrap* [PL05] comme exemple d'algorithme agglomératif.

Walktrap :

Au début de l'algorithme, on a une partition $P_i = \{\{v_i\}, v_i \in V, i = 1...n\}$ du graphe G qui représente n communautés. Chaque communauté contient un seul nœud. La première étape consiste à calculer toutes les distances entre les n communautés. Ensuite, on répète les étapes suivantes jusqu'à l'obtention d'une seule communauté :

- Étape 1 : sélectionner les deux communautés C_1 et C_2 de la partition P_k qui ont une distance minimale entre elles (cf. équation 2.1).
- Étape 2 : fusionner les deux communautés en une seule communauté $C_3 = C_1 \bigcup C_2$ et créer un nouveau partitionnement $P_{k+1} = (P_k \backslash \{C_1, C_2\}) \bigcup \{C_3\}$.
- Étape 3 : calculer les distances entre les communautés et aller à l'étape 1.

Après $n - 1$ itérations, l'algorithme a comme partition $P_n = \{V\}$ représentant le graphe entier. Chaque fusion entre deux communautés engendre une nouvelle partition. Cet algorithme se représente à travers un dendrogramme dont P_1 représente les feuilles qui se trouvent en bas et P_n la plus haute racine comme l'illustre la figure 2.1. La différence entre cet algorithme et les autres approches agglomératives se manifeste à travers le choix de communautés à fusionner à chaque itération. *Walktrap* calcule pour chaque couple de communautés la variable $\Delta\sigma$ définie dans l'équation 2.1. Les deux communautés qui ont la valeur minimale de $\Delta\sigma$ (i.e., qui sont les plus homogènes) seront fusionnées constituant ainsi une seule nouvelle communauté.

$$\Delta\sigma(C_1, C_2) = \frac{1}{n} * \frac{|C1| * |C1|}{|C1| + |C2|} * r^2_{(C_1C_2)} \tag{2.1}$$

$|C_i|(i = 1, 2)$ désigne le nombre de nœuds dans la communauté C_i et $r_{(C_1C_2)}$ représente la distance entre les communautés C_1 et C_2. La distance r est grande lorsque deux nœuds sont dans deux communautés distinctes mais petite s'ils appartiennent à la même communauté.

$$r_{(C_1C_2)} = \sqrt{\sum_{k=1}^{n} \frac{(Pr^t_{C_1v_k} - Pr^t_{C_2v_k})^2}{d(v_k)}} \tag{2.2}$$

n est le nombre de nœuds du graphe, $d(v_k)$ est le nombre de voisins du nœud v_k et $Pr^t_{C_iv_k}(i = 1, 2)$ désigne la probabilité d'aller jusqu'au nœud v_k en partant d'un nœud (choisi aléatoirement) de la communauté C_i, et ce après un nombre de pas égal au paramètre t introduit. $Pr^t_{C_iv_k}$ se défini de la façon suivante :

$$Pr^t_{C_iv_k} = \frac{1}{|C_i|} \sum_{v_j \in C_i} Pr^t_{v_jv_k} \tag{2.3}$$

De même, $Pr^t_{v_jv_k}$ est la probabilité d'aller jusqu'au nœud v_k en partant du nœud v_j après une marche aléatoire de longueur t. L'idée de base de cet algorithme est que deux nœuds proches ayant un grand nombre de voisins ont des valeurs élevées et proches de $Pr_(v_jv_k)^t$. Donc, si deux communautés (C_1 et C_2) sont loin l'une de l'autre, elles auront de faibles liens entre elles mais par contre plus de liens à l'intérieur. Elles ont de ce fait une grande distance rC_1C_2 et donc une grande valeur de $\Delta\sigma(C_1, C_2)$ et donc ne seront pas fusionnées.

Après la dernière fusion, nous disposons du dendrogramme qui représente les différentes communautés obtenues après chaque itération. Le problème qui se pose est de choisir la bonne structure de communautés au cours des itérations. L'algorithme [AK08] fait appel à la modularité de Newman [New06] pour traiter le dendrogramme et dégager les communautés finales.

La modularité Q : Newman a défini la modularité Q d'un graphe comme suit :

$$Q = \frac{1}{2m} * \sum_{j,k} (a_{v_j,v_k} - \frac{d(v_j)d(v_k)}{2m}) * \delta(v_j, v_k) \quad j = 1...n \ \ et \ \ k = 1...n \tag{2.4}$$

Où m est le nombre de liens du graphe, n est le nombre de nœuds, a_{v_j,v_k} vaut 1 si les nœuds v_j et v_k sont liés et 0 dans le cas contraire. La variable $d(v_j)$ est le nombre de voisins du nœud v_j et δ est le symbole de Kronecker qui vaut 1 si v_j et v_k appartiennent à la même communauté et 0 sinon.

La modularité représente la différence entre la valeur d'adjacence entre deux nœuds de la même communauté (a_{v_j,v_k}) et la probabilité pour que ceux-ci soient connectés. Comme

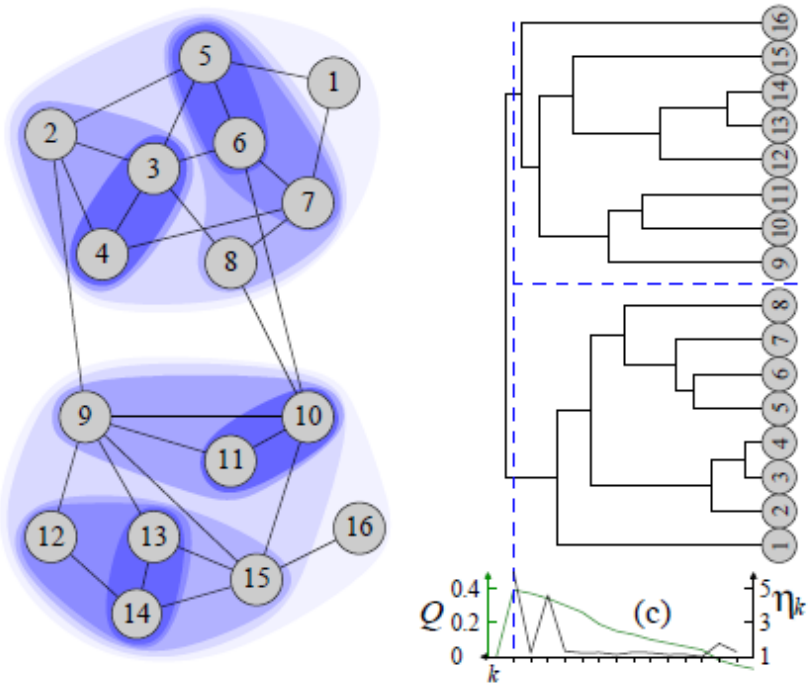

FIGURE 2.1 – Exemple de réseau à 16 sommets divisé en 2 communautés avec *Walktrap*.

$\frac{d(v_j)d(v_k)}{2m}$ représente la probabilité qu'il existe un lien entre v_j et v_k, cette valeur est importante si le nombre de liens à l'intérieur des communautés est élevé et le nombre de liens entre ces communautés est faible. Avec cette expression de modularité, une communauté est vue comme un ensemble de nœuds qui ont plus de liens entre eux que de liens avec des nœuds à l'extérieur de la communauté.

La procédure *Walktrap* la modularité pour chaque nouvelle structure de communautés (après chaque fusion). Une coupe du dendrogramme qui correspond à la valeur maximale de la modularité Q donne les communautés dégagées du graphe. La figure 2.1, extraite de [PL05], montre les résultats d'exécution de l'algorithme *Walktrap* avec un réseau de 16 nœuds. L'évolution de Q au cours des itérations de fusion montre qu'elle est maximale pour une valeur de 0.4. La projection de cette valeur sur le dendrogramme divise le réseau en deux communautés de huit nœuds chacune.

La complexité de *Walktrap* est de $O(nmH)$ où m est le nombre de liens, n le nombre de nœuds et H est la hauteur du dendrogramme. Dans le cas où les sommets sont fusionnés un par un, H atteint son maximum et vaut $n - 1$. L'inconvénient de cet algorithme cependant est le paramètre t qui représente le nombre de pas ou de liens entre les nœuds.

En effet, l'algorithme doit fixer à l'avance cette valeur. En pratique, on affecte la valeur $log(n)$ à t ($t = log(n)$). Toutefois cette valeur est liée à la densité des liens dans un graphe et non pas à sa taille.

2.1.2 Les algorithmes divisifs

Les algorithmes divisifs consistent à scinder un réseau en plusieurs communautés en éliminant itérativement les liens entre les nœuds. Ils commencent par une seule communauté (le réseau entier), en haut du dendrogramme, jusqu'à avoir n communautés à un seul nœud représentant les feuilles du dendrogramme. Dans chaque itération, tout réseau connexe est considéré comme une communauté. Les méthodes existantes se distinguent par le choix des liens à éliminer et par les poids accordés aux liens. Nous retenons l'algorithme *Edge Betweenness* [NG04] comme exemple d'algorithme divisif.

Edge Betweenness :

Les méthodes de détection de communautés s'intéressent généralement aux nœuds du réseau. Newman [NG04] s'est penché sur les liens plutôt que sur les nœuds. En effet, l'identification des liens se trouvant entre les communautés, ainsi que leur élimination permet d'identifier les différentes communautés dans un réseau. Afin de trouver les liens inter-communautés, l'algorithme *Edge Betweenness* accorde à chaque lien une mesure de centralité d'intermédiarité (*Edge Betweenness Centrality*).

Mesure de centralité d'intermédiarité : La mesure de centralité d'intermédiarité se base sur le calcul du plus court chemin (distance géodésique) [AK08, Fre79]. En effet, un nœud intermédiaire est un nœud porteur de plusieurs chemins géodésiques reliant les différents couples de nœuds d'un réseau. La centralité d'intermédiarité d'un lien e_i se calcule comme suit :

$$EBC(e_i) = \sum_{j<k} \frac{NP_{v_j v_k}(e_i)}{NP_{v_j v_k}} \tag{2.5}$$

$NP_{v_j v_k}$ représente le nombre de plus courts chemins entre les deux nœuds v_j et v_k. $NP_{v_j v_k}(e_i)$ est le nombre de plus courts chemins entre les deux nœuds v_j et v_k passant par le lien e_i.

Notons qu'un lien intermédiaire est un élément influant dans le réseau. Il contrôle les flux d'information circulant entre tous les nœuds. Il a aussi une forte probabilité de se trouver entre les différentes communautés. Cette mesure de centralité peut être calculée

pour les nœuds et les liens de la même façon. Néanmoins, le calcul de cette mesure est un processus trop lent car un parcours de tous les chemins possibles entre tous les couples de nœuds doit être fait pour chaque lien. Dans ce sens, Newman a présenté une méthode plus rapide avec une justification des optimisations suivies [NG04, Bra01]. Dans [NG04], un algorithme très détaillé a été présenté pour mieux implémenter cette phase de calcul du degré d'intermédiarité. L'algorithme de calcul d'intermédiarité des liens est composé de deux principales phases. La première phase consiste à affecter pour chaque nœud un poids, tandis que la seconde calcule le degré d'intermédiarité des liens. Les détails relatifs à chaque phase se présentent comme suit :

Première phase :

1- Choisir un nœud v_s et mettre à 0 sa distance et à 1 son poids : $d_s = 0$, $w_s = 1$.

2- Pour tout nœud v_i adjacent à v_s, $d_i = d_s + 1$ et $w_i = w_s = 1$.

3- Pour tout nœud v_j adjacent à v_i :

i-Si la distance d_j de v_j est calculée pour la première fois, $d_j = d_i + 1$ et $w_j = w_i$.

ii-Sinon, $d_j = d_i + 1$, $w_j = w_j + w_i$.

4- S'il y a un nœud sans distance, répéter l'étape 3.

Deuxième phase :

1- Trouver tous les nœuds v_t se trouvant à l'extrémité du réseau (aucun chemin reliant deux nœuds ne passe à travers eux).

2- Pour tout nœud v_i adjacent à v_t, calculer le poids du lien entre v_i et v_t : $w_{it} = w_i/w_t$.

3- En commençant par le lien le plus loin de v_s, le poids accordé à chaque lien entre deux nœuds adjacents v_i et v_j (v_j plus loin de v_s que v_i) w_{ij} = (1+le poids de tous les liens adjacents) $* w_i/w_j$.

4- Répéter l'étape 3 jusqu'à atteindre le sommet v_s.

Toutes ces étapes ont été faites pour un seul nœud v_s du réseau. Le parcours de tous les sommets du réseau génère à chaque fois des poids accordés aux liens. La somme de ces scores est la mesure de centralité d'intermédiarité propre aux liens. La figure 2.2 illustre les mesures de centralité d'intermédiarité pour un exemple du réseau.

Description de l'algorithme *Edge Betweenness* : Cette approche se base sur le principe qu'un lien se trouvant entre les communautés fait partie d'un très grand nombre

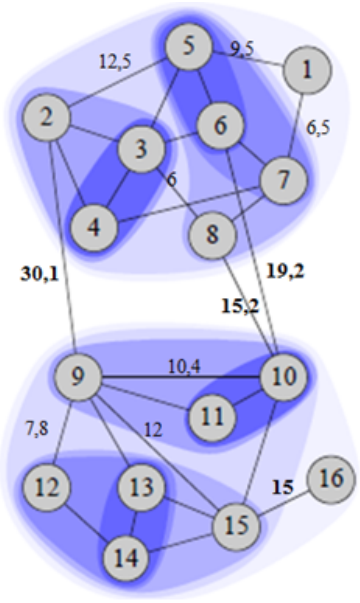

FIGURE 2.2 – Mesures de centralité d'intermédiarité d'un réseau.

de chemins géodésiques entre les différents nœuds. Il a de ce fait une grande valeur de centralité d'intermédiarité. Cependant, quelques liens se trouvant entre les communautés et qui sont plus longs que les chemins géodésiques, ont des EBC (centralité d'intermédiarité d'un lien) faibles. D'où l'idée d'éliminer, dans chaque itération, le lien le plus intermédiaire (i.e., ayant la plus grande valeur EBC), puis recalculer à nouveau les EBC de tous les liens restants.

L'algorithme est constitué de quatre étapes principales :

1- Calculer les EBC pour tous les liens du graphe.

2- Éliminer le lien le plus intermédiaire.

3- Recalculer les EBC de tous les liens restants.

4- Répéter les étapes 2 et 3 jusqu'à l'élimination de tous les liens.

La méthode *Edge Betweenness* commence par une seule communauté C contenant tous les nœuds du graphe. Après chaque itération, et s'il y a un nouveau sous-graphe qui n'est pas connexe au graphe initial, de nouvelles communautés C_1 et C_2 se génèrent avec $C = C_1 \bigcup C_2$. Ce processus se répète jusqu'à l'obtention des feuilles du dendrogramme qui sont constituées par n communautés. Chacune de ces communautés contient donc un seul nœud. Nous traitons ici comme exemple le réseau social du club de karaté de Zachary [Zac77], représenté dans la figure 2.3 prise de[NG04]. Ce réseau représente les liens d'amitié entre 34 membres d'un club dans une université aux États-Unis. La figure

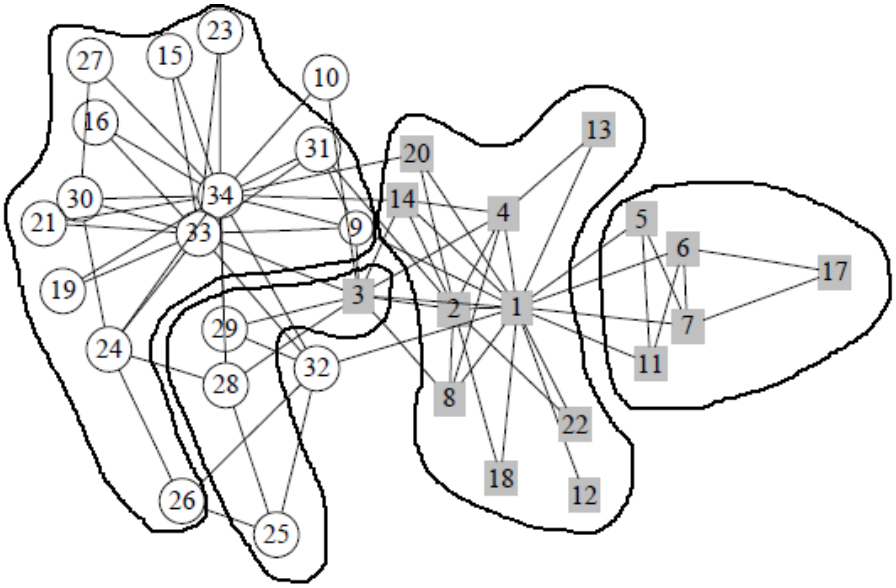

FIGURE 2.3 – Le réseau social du club de karaté de Zachary.

2.4, également prise de [NG04], montre le dendrogramme obtenu après l'exécution de l'algorithme avec ce réseau. Une meilleure structure de communautés, qui correspond à une valeur maximale de la modularité Q dans le dendrogramme, est représentée par cinq communautés (figure 2.4). Si nous ignorons le nœud 10, les communautés trouvées contiennent 5, 10, 4 et 14 membres. Elles sont délimitées dans la figure 2.3. Il faut noter que le nœud 32 fait en fait partie du groupe des quatorze nœuds.

Les résultats de cet algorithme prouvent sa performance pour la détection des communautés. Cependant, son temps de calcul est très élevé. La phase de calcul des degrés d'intermédiarité est trop lente. Ajoutons à cela qu'elle est exécutée à chaque itération. L'algorithme a une complexité de l'ordre de $O(m^2.n)$ où m est le nombre de liens et n est le nombre de nœuds [NG04]. Pour éviter cet inconvénient, plusieurs algorithmes [For10, GSPA04] ont proposé des mesures presque équivalentes au degré de centralité d'intermédiarité locales et qui se calculent rapidement. Ceci dit, ce processus reste un peu lent. Dans un esprit similaire, nous avons décidé de garder cette mesure d'intermédiarité tout en proposant une approche qui réduit énormément le nombre d'itérations mais s'avère évidemment inappropriée pour de grands réseaux.

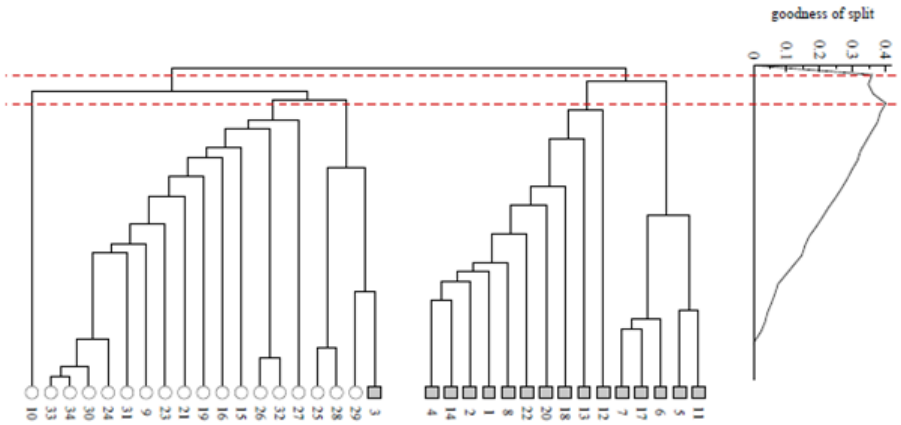

FIGURE 2.4 – Dendrogramme du réseau de Zachary avec les mesures de modularités.

2.2 Les algorithmes d'optimisation d'une fonction objective

Il existe un certain nombre d'algorithmes se basant sur des heuristiques pour définir la structure de communauté des réseaux. Ce type d'algorithme consiste à définir une fonction objective dont la valeur varie selon les communautés dégagées. La fonction est maximale pour la meilleure structure de communauté. Un exemple de ce type d'algorithme est *Fast Greedy* de Newman [New04].

***Fast Greedy* :**

L'algorithme utilise la fonction de modularité Q définie dans l'équation 2.4. Chaque division possible du graphe a une valeur propre Q. Cette valeur est élevée pour une bonne division du graphe et faible dans le cas contraire. En observant toutes les divisions possibles du graphe, on remarque que le calcul exhaustif de Q prend une quantité du temps exponentiel au nombre de nœuds. Cette méthode peut être utilisée pour les réseaux de taille faible (une vingtaine ou trentaine de nœuds) pour avoir un temps d'exécution raisonnable.

Des stratégies d'optimisation ont été proposées afin d'améliorer les délais d'exécution de cet algorithme [New04]. La présente approche commence par considérer chaque nœud comme une communauté à part, pour ensuite fusionner les communautés en paires, en choisissant à chaque étape la paire de communautés dont la fusion donne la plus forte augmentation de Q par rapport à l'itération antérieure. Puisque la fusion des communautés non adjacentes ne produit jamais une plus grande augmentation de Q, plusieurs fusions

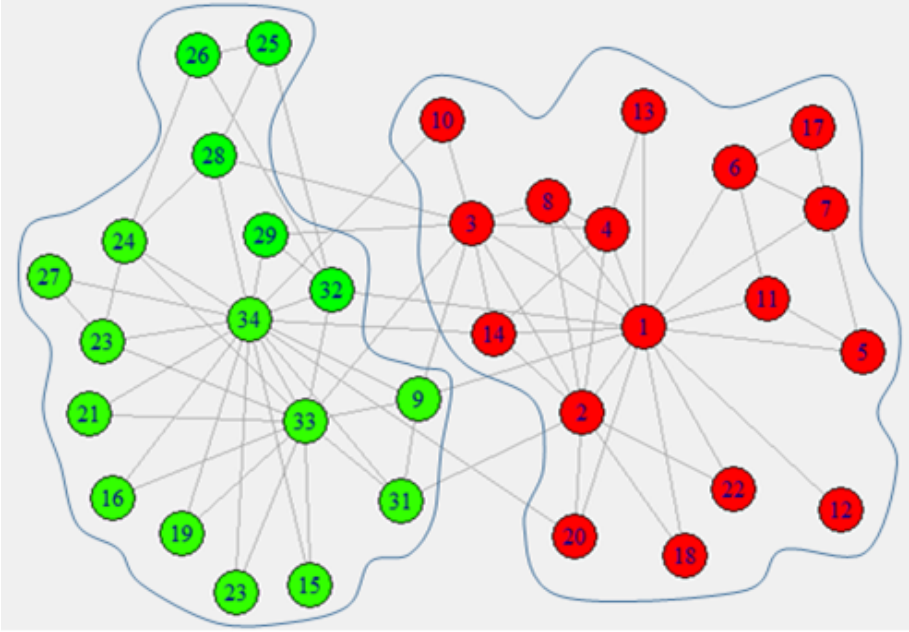

FIGURE 2.5 – Structure de communautés détectée par *Fast Greedy* pour le réseau de Zachary.

possibles sont évitées. Dans ce cas, il y a m paires et $n-1$ fusions possibles pour relier toutes les communautés entre-elles. Ainsi, le temps total d'exécution est en $O(mn)$. La valeur maximale de Q correspond à la meilleure structure de communautés. Par exemple, pour le réseau de Zachary et avec une valeur maximale de Q=0,38, l'algorithme *Fast Greedy* identifie deux communautés de 17 membres chacune. La figure 2.5 représente ces deux communautés.

Fast Greedy n'est pas un algorithme hiérarchique. Il n'accorde aucun score ou paramètre aux liens. Ajoutons à cela qu'il ne mesure pas la distance ou la similarité entre les communautés à fusionner. De même, l'inconvénient majeur de cette méthode réside dans sa qualité de classification qui est moins bonne en comparaison avec d'autres algorithmes utilisant la modularité, citons par exemple *Edge Betweenness*.

2.3 Les algorithmes à base du modèle

Les algorithmes à base du modèle sont des algorithmes de classification non supervisée utilisant des méthodes à base de prototypes exprimés dans un formalisme de modèles. Pour chaque type de données, un modèle d'apprentissage adapté à la nature des données traitées est proposé. Pour le traitement des graphes, nous détaillons l'algorithme décrit

en [RAK07].

***Label Propagation* :**

Cet algorithme se base sur le principe que chaque nœud change de communauté en fonction de la communauté à laquelle appartiennent ses voisins. Un nœud fait partie de la communauté qui contient le plus grand nombre de nœuds voisins. Ce processus est le modèle d'apprentissage pour *Label Propogation*, qui s'exécute sur tous les nœuds dans chaque itération. Au début de l'algorithme, chaque nœud se trouve dans une seule communauté. Puis, les nœuds changent de communautés tout en respectant le modèle d'apprentissage. Avec cette méthode, un groupe de nœuds, fortement liés entre eux, finit par se trouver dans la même communauté. L'exemple de la figure 2.6 illustre ceci avec son réseau à 7 nœuds. Au début de l'algorithme, chaque nœud se trouve dans une communauté (a, b, c, d, e, f, g) comme illustré dans la figure 2.6(i). L'algorithme choisit d'une façon aléatoire le traitement des nœuds. Supposons qu'il a choisi le nœud 3 de la communauté c. Comme chacun de ses voisins appartient à une seule communauté, ce nœud peut appartenir à n'importe quelle communauté (a, b, ou d) qui lui est voisine. Dans notre cas, l'algorithme a choisi aléatoirement d'associer le nœud 3 à la communauté b (figure 2.6(ii)). Il s'en suit que l'algorithme a choisi aléatoirement de traiter le nœud 7 se trouvant dans la communauté g. Ce dernier a un voisin dans la communauté e et un autre voisin dans la communauté f. Le deuxième nœud à traiter passe donc de la communauté g à la communauté e (figure 2.6(iii)). Le nouveau nœud à traiter est le nœud 4 qui a deux voisins de la communauté b et un voisin de la communauté a. Puisque le nombre de voisin dans la communauté b est plus grand que le nombre de voisin dans la communauté a, le nœud 4 fait partie de la communauté b. En appliquant ce processus avec tous les autres nœuds, nous avons eu deux communautés b et e pour le graphe entier (figure 2.6(vi)).

La communauté d'un nœud v_i à la $t^{ème}$ itération dépend des communautés des voisins de v_i à la $t-1^{ème}$ itération. Il existe aussi une autre façon asynchrone pour affecter les communautés. La communauté d'un nœud v_i à la $t^{ème}$ itération dépend des communautés des voisins de v_i à la tème itération. Idéalement, l'algorithme s'arrête lorsque les nœuds ne changent plus de communautés même si ce n'est pas toujours le cas. En effet, lorsqu'un nœud a un nombre maximal de voisins qui appartiennent à deux communautés ou plus, il change de communauté à chaque itération.

L'algorithme *Label Propagation* est décrit principalement par les étapes suivantes :

1-Initialement chaque nœud v_i, à l'itération $t = 0$, appartient à une seule communauté i :$C_{v_i}(0) = i$, $i = 1...n$.

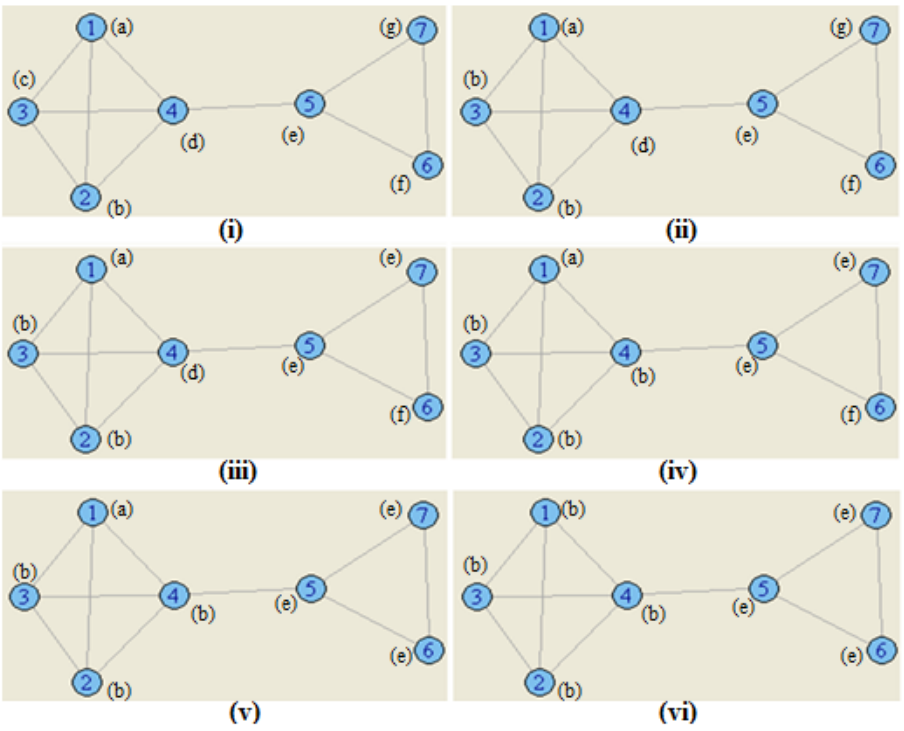

FIGURE 2.6 – Exemple d'exécution de l'algorithme *Label Propagation*.

2-Définir $t = 1$.

3- Pour chaque nœud v_i, trouver la communauté du nœud v_i à la $t^{ème}$ itération $C_{v_i}(t)$ en fonction des communautés de voisins de v_i à l'itération t ou $t-1$.

4-Si tous les nœuds ne changent plus de communautés ($C_{v_i}(t) = C_{v_i}(t-1)$ pour $i = 1...n$), l'algorithme s'arrête. Sinon, $t = t+1$ et aller à l'étape (3).

Comme l'algorithme choisit le nœud à traiter aléatoirement et avec une condition d'arrêt (non pas une mesure), plusieurs résultats finaux peuvent être obtenus. Dans ce sens, la figure 2.7 prise de [RAK07], contient trois structures de communautés différentes dégagées par Label Propagation pour le réseau du club de karaté. Dans la figure 2.7(i) et 2.7(ii), *Label Propagation* a trouvé deux communautés, alors que dans la figure 2.7(iii), il a détecté trois communautés.

2.4 Conclusion

Nous avons détaillé quatre algorithmes appartenant à des catégories de méthodes de détection de communautés différentes. Ils ont différentes caractéristiques comme illustré dans le tableau 2.1. Le seule algorithme qui a un paramètre d'entreé est *Walktrap*. Il est à

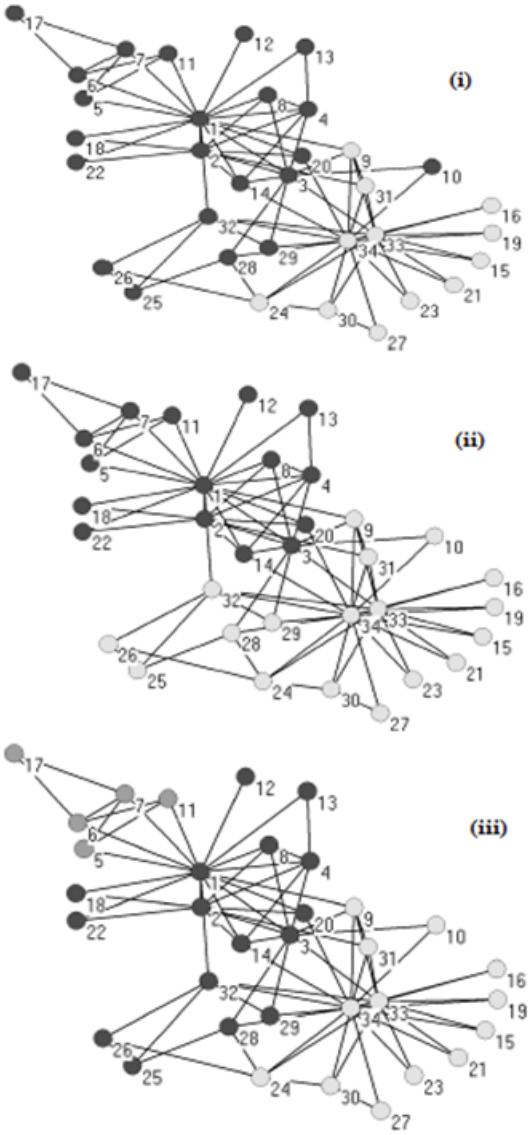

FIGURE 2.7 – Les différentes structures de communautés trouvées par *Label Propagation* pour le réseau du club de karaté.

Nom de l'algorithme	Classification hiérarchique	Paramètres requis	Utilisation de la modularité	Déterministe
Walktrap	Oui	Longueur de la marche aléatoire	Avec le dendrogramme	Oui
Edge Betweenness	Oui	Aucun	Avec le dendrogramme	Oui
Fast Greedy	Non	Aucun	Optimisation	Oui
Label Propagation	Non	Aucun	Non	Non

Tableau 2.1 – Principales propriétés des algorithmes étudiés.

noter aussi que l'algorithme *Label Propagation* est le seul à être non déterministe puisque son exécution avec le même réseau peut donner plusieurs résultats.

Lorsque le nombre de liens inter-communautés est faible, tous ces algorithmes finissent par avoir la même structure de communautés. En augmentant le ratio des liens inter-communautés, les résultats de détection de communautés vont diverger et la qualité du regroupement se dégrade. Dans le chapitre suivant, nous allons présenter une nouvelle méthode de détection de communautés qui se comporte mieux avec des réseaux complexes où le nombre de liens inter-communautés est élevé. Une analyse empirique de notre méthode avec celles présentées dans ce chapitre est ensuite effectuée.

Chapitre 3

Méthode proposée

Ce chapitre présente une nouvelle méthode de détection de communautés dans les réseaux sociaux. La méthode proposée est applicable sur des réseaux non orientés et non pondérés. Il s'agit d'une méthode d'optimisation d'une fonction de qualité qui permet de trouver de meilleures structures de communautés de graphe. La détection de communautés se fait en deux phases. La première phase vise à éliminer les liens inter-communautés afin de décomposer le réseau initial en petits groupes élémentaires dont les nœuds sont bien reliés entre eux. La deuxième phase vise à identifier une structure de communautés qui maximise la modularité Q via un processus itératif. Les détails de chaque phase sont présentés dans les sous-sections 3.1 et 3.2.

3.1 La première phase

En général, tel qu'indiqué dans le chapitre 2, les algorithmes existants qui se basent sur l'élimination des liens, comme la procédure *Edge Betweenness*, enlèvent un seul lien dans chaque itération et calculent une fonction objective de qualité. L'algorithme *Edge Betweenness* finit par enlever tous les liens du réseau tout en gardant la meilleure structure de communautés qui maximise la fonction de qualité. L'inconvénient de cette approche réside dans l'élimination d'un seul lien à la fois, ce qui est fastidieux dans des réseaux avec des nœuds fortement connectés. Afin de pallier à ce problème, nous avons développé une méthode qui vise à réduire considérablement le nombre d'itérations et éliminer plusieurs liens inter-communautés en une seule itération. Dans ce qui suit, nous allons d'abord présenter les mesures retenues pour distinguer entre les liens inter-communautés et intra-communautés. Par la suite, nous présentons une approche itérative qui se base sur l'inertie inter-classes pour identifier et éliminer les liens inter-communautés.

Pour éliminer les liens au cours de la première phase, quelques mesures propres aux liens ont été étudiées. Tel est le cas de la centralité d'intermédiarité, le nombre de voisins communs, le coefficient de Jaccard et la covariance. La mesure de centralité d'intermédiarité [GA08, Fre79], déjà présentée en sous-section 2.1.2, est le rapport entre le nombre de plus courts chemins passant par un lien et le nombre total de plus courts chemins. Le nombre de voisins communs (*Common neighbors*) [LNK07] pour un lien reliant deux nœuds est le nombre de nœuds se retrouvant comme voisins de ces deux nœuds. Le coefficient de Jaccard [LNK07] est un indice de similarité qui mesure le chevauchement de deux ensembles contenant les voisins de deux nœuds reliés à un lien donné. Le coefficient de Jaccard est défini comme la taille de l'intersection des ensembles divisée par la taille de leur union. La mesure de Czekanovski-dice [BCM+04] se base aussi sur le calcul du nombre de voisins. La mesure que nous retenons dans le cadre de ce travail est celle de la covariance car nos tests empiriques sur les métriques précédemment mentionnées ont montré la supériorité de la covariance en terme de précision et de performance pour diverses configurations de réseaux.

3.1.1 Covariance

La mesure de covariance [WIK13] permet d'évaluer le sens de variation de deux séries de données numériques (ou deux variables aléatoires) et de qualifier le degré de leur indépendance. Ainsi, si les grandes (respectivement petites) valeurs d'une variable correspondent principalement aux grandes (respectivement petites) valeurs de l'autre variable, alors la covariance est élevée et les deux variables tendent à montrer un comportement similaire. Dans le cas contraire, lorsque les valeurs supérieures d'une variable correspondent principalement aux petites valeurs de l'autre, les variables tendent à montrer un comportement opposé, et la covariance est donc faible. Les valeurs de covariance sont comprises entre -1 et 1. Lorsque deux variables sont indépendantes, la covariance est nulle mais la réciproque n'est pas vraie.

Nous avons choisi d'utiliser cette mesure de covariance sur les liens reliant les nœuds du réseau. En effet, si un lien existe entre deux nœuds ayant un nombre important de voisins communs (c.-à-d. les deux nœuds ont le même comportement), alors ce lien a une covariance élevée et représente lien intra-communauté. Dans le cas contraire, si un lien relie deux nœuds ayant un nombre faible de voisins communs, ce lien a alors une covariance faible. Il s'agit alors d'un lien inter-communautés. Afin de calculer la covariance des liens, il faut calculer la matrice d'adjacence A du réseau. Si deux nœuds v_i et v_j de ce réseau sont directement connectés, $A[i, j]$ est égal à un et vaut zéro dans le cas contraire. La

covariance d'un lien e_{ij} reliant les nœuds v_i et v_j est élevée si les uns et les zéros du vecteur $A[i]$ correspondent respectivement aux uns et zéros du vecteur $A[j]$ (c.-à-d. les nœuds v_i et v_j ont les mêmes voisins et se trouvent probablement dans la même communauté). La covariance est faible si les uns et les zéros de $A[i]$ correspondent respectivement aux zéros et uns de $A[j]$. Dans ce dernier cas, les nœuds v_i et v_j n'ont pas les mêmes voisins et ils se trouvent donc dans différentes communautés. La covariance d'un lien e_{ij} est définie par l'équation suivante :

$$cov(e_{ij}) = \frac{1}{n}\sum_{k=1}^{n}(A[i,k] - \bar{A[i]})(A[j,k] - \bar{A[j]}) \tag{3.1}$$

Où $\bar{A[i]}$ est la moyenne du vecteur $A[i]$ et n est le nombre de nœuds dans le réseau. Dans ce qui suit, les valeurs de covariance sont normalisées.

En regardant la figure 3.1, on remarque bien que les liens inter-communautés 5-12, 6-7 et 10-13 ont les plus faibles valeurs de covariance. Les liens intra-communauté 1-3, 2-3, 3-4, 7-8, 7-9 et 7-11 ont des valeurs de covariance maximales.

Après plusieurs expérimentations, nous avons retenu la covariance comme mesure de similarité entre deux nœuds et donc des liens. En effet, cette mesure a donné de bons résultats et de bonnes performances.

Nous présentons dans la section suivante la première phase de la méthode proposée, la prise en compte de l'inertie inter-classes et le mode d'utilisation de la mesure de covariance.

3.1.2 Inertie inter-classes

Cette mesure est généralement utilisée pour le regroupement des données. Elle sert à l'étude d'un ensemble de données, en le regroupant en plusieurs sous-ensembles de sorte que les éléments d'un même groupe soient le plus semblables ou proches possible et que deux groupes distincts soient les plus hétérogènes possible. Cette mesure est déployée pour déterminer deux groupes de liens : les liens avec covariance élevée et les liens avec covariance faible.

La délimitation de deux groupes de liens se fait en trouvant la valeur maximale de l'inertie inter-classes, I, pour toutes les combinaisons possibles des données classées par ordre décroissant. L'inertie inter-classes entre deux groupes G_1 et G_2 est définie comme suit :

$$I(G_1, G_2) = |G_1|(\mu_1 - \mu)^2 + |G_2|(\mu_2 - \mu)^2 \tag{3.2}$$

-$|G_1|$ et $|G_2|$ sont, respectivement, le nombre d'éléments (liens) dans les deux groupes G_1 et G_2.

-μ_1, μ_2 et μ représentent, respectivement, la moyenne de la covariance pour G_1, G_2 et l'ensemble des groupes.

3.1.3 Algorithme de la première phase

L'algorithme de la première phase consiste essentiellement à éliminer un ensemble de liens à chaque itération. Ce processus se répète jusqu'à l'obtention d'un nombre de sous-réseaux supérieur à $\sqrt{n}$, où n est le nombre de nœuds du graphe initial. Nous avons déduit le nombre minimal de sous-réseaux en fonction de $\sqrt{n}$ car nous utilisons la modularité dans la deuxième phase de la méthode proposée. En effet, dans [GSPA04], il a été prouvé pour un nombre important de réseaux que la modularité est maximale lorsque le nombre de communautés est proche de $\sqrt{n}$. L'algorithme 1 décrit les étapes de la première phase.

Algorithme 1 : *Première phase*

Entrées : $G = (V, E)$: le graphe initial.

Sorties : $G' = (V, E')$: le graphe après l'élimination des liens inter-communautés.

Variables : NSR : le nombre de sous-réseaux dans G', $MinSR$: le nombre minimal de sous-réseaux à identifier, *liste_cov* : liste de réels contenant les covariances de liens, *liste_inertie* : liste de réels contenant les valeurs de l'inertie inter-classes, *indice_seuil* : nb. de liens à garder, In : valeur de l'inertie inter-classes.

```
1  début
2      E' = E ;
3      NSR = 1 ;
4      MinSR = √n ;
5      tant que NSR < MinSR faire
6          pour chaque lien e_ij dans E' faire
7              Ajouter cov(e_ij) dans liste_cov ;
8          Trier liste_cov par ordre décroissant ;
9          pour i de 1 à |E'| − 1 faire
10             In =
               CalculerInertie(liste_cov[1...i], liste_cov[i + 1...taille(liste_cov)]) ;
11             Ajouter In dans liste_inertie ;
12         indice_seuil = IndiceValeurMaximale(liste_inertie) ;
13         E' = SupprimerLiens(E', liste_cov[indice_seuil...|E'|]) ;
14         NSR = SousRes(G') ;
15     retourner (V, E')
```

L'algorithme a pour entrée le graphe $G = (V, E)$. Le résultat de cet algorithme est un nouveau graphe $G' = (V, E')$ dont l'ensemble de liens E' est inclus dans E. Au début de l'algorithme (ligne 2), E' contient l'ensemble de liens initiaux du graphe G. Le nombre de sous-réseaux (NSR) est initialement égal à un puisqu'on a toujours comme entrée un réseau connexe. Le nombre minimal de sous-réseaux ($MinSR$) est égal à $\sqrt{n}$. Dans la boucle *tant que* que (ligne 5 jusqu'à la ligne 14), l'algorithme élimine à chaque itération un ensemble de liens et calcule le nombre de sous-réseaux. Si ce nombre est inférieur à $MinSR$, il élimine encore d'autres liens. Dans le cas contraire, l'algorithme arrête la phase itérative et retourne le graphe $G' = (V, E')$. L'algorithme commence donc par

calculer les covariances de tous les liens et les mettre dans *liste_cov* (ligne 6 et 7). Puis, la liste *liste_cov* est triée par ordre décroissant. Ensuite, l'algorithme calcule les différentes valeurs de l'inertie inter-classes avec la fonction $CalculerInertie(l1, l2)$ (ligne 9 à 11). L'indice de la valeur maximale des inerties inter-classes (*liste_inertie*) est retourné par la fonction $IndiceValeurMaximale(liste\_inertie)$. Cette valeur, sauvegardée dans *indice_seuil*, représente le nombre de liens à ne pas éliminer. Donc l'algorithme utilise la fonction $SupprimerLiens(E', liste\_cov[indice\_seuil, \ldots, |E'|])$ pour éliminer $|E'| - indice\_seuil + 1$ liens qui ont les plus faibles valeurs de covariance.

Algorithme 2 : *Procédure SupprimerLiens(G', l)*

Entrées : $G' = (V, E')$: le graphe avant l'élimination de liens, l : les valeurs de covariance des liens à éliminer.

Sorties : $G' = (V, E')$: le graphe après l'élimination de liens.

```
1 début
2     pour chaque lien e_ij dans E' faire
3         si cov(e_ij) ∈ l alors
4             éliminer e_ij ;
5     retourner G'
```

3.1.4 Un exemple illustratif

L'algorithme de la première phase est exécuté avec l'exemple du réseau représenté dans la figure 3.1. Il s'agit d'un réseau de 14 nœuds et 24 liens. Au début de l'algorithme, NSR est égal à un et inférieur à $MinSR$ ($MinSR = \sqrt{n} = 3.74$). L'algorithme commence donc par calculer les covariances de tous les liens dans E', les classer par ordre décroissant, et calculer les différentes valeurs de l'inertie inter-classes comme le montre le tableau 3.1. Les valeurs de covariances sont normalisées entre 0 et 1. La valeur maximale de l'inertie inter-classes (1.53) est obtenue avec un *indice_seuil* égal à 17. Le nombre de liens à éliminer dans cette première itération est donc égal à huit ($|E'| - indice\_seuil + 1$=24-17+1=8). Ce sont les liens qui ont les plus faibles valeurs de covariance (5-2, 6-4, 8-6, 10-7, 6-5, 13-10, 7-6 et 12-5). Le nouveau graphe obtenu à la fin de la première itération est représenté dans la figure 3.2. Le nombre de sous-réseaux NSR est égal à trois et il est encore inférieur à $MinSR$. Une deuxième itération est alors amorcée et le même processus décrit précédemment est répété.

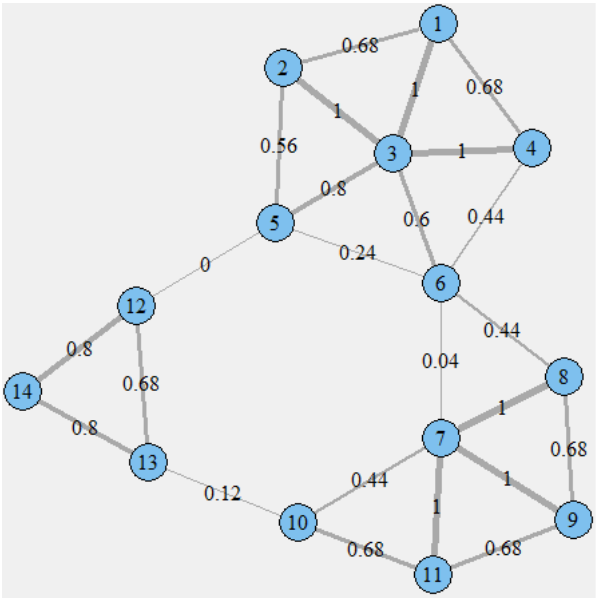

FIGURE 3.1 – Exemple de réseau

Liens	*liste_cov*	*liste_inertie*
3–1	1.00	0.13
3–2	1.00	0.28
4–3	1.00	0.44
8–7	1.00	0.62
9–7	1.00	0.81
11–7	1.00	1.03
3–5	0.80	1.08
14–12	0.80	1.15
14–13	0.80	1.23
2–1	0.68	1.23
4–1	0.68	1.24
9–8	0.68	1.26
11–9	0.68	1.31
11–10	0.68	1.38
13–12	0.68	1.47
6–3	0.60	1.51
5–2	0.56	*1.53*
6–4	0.44	1.45
8–6	0.44	1.40
10–7	0.44	1.39
6–5	0.24	1.18
13–10	0.12	0.83
7–6	0.04	0.42
12–5	0.00	–

Tableau 3.1 – Valeurs de covariances et de l'inertie inter-classes durant la première itération

À la fin de la deuxième itération, le nombre de sous-réseaux NSR est égal à six et il est supérieur à $MinSR$ qui vaut 3.74. Donc c'est la fin de l'algorithme de la première phase dont la sortie est le graphe $G' = (V, E')$ avec V représentant les quatorze nœuds du graphe et E' contenant les onze liens qui n'ont pas été éliminés durant les différentes itérations. Le graphe obtenu à la fin de la première phase est illustré par la figure 3.3.

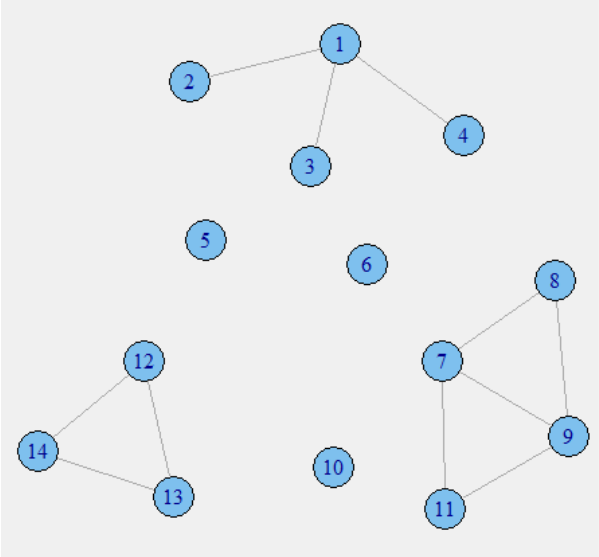

FIGURE 3.3 – Le graphe G' après la deuxième itération

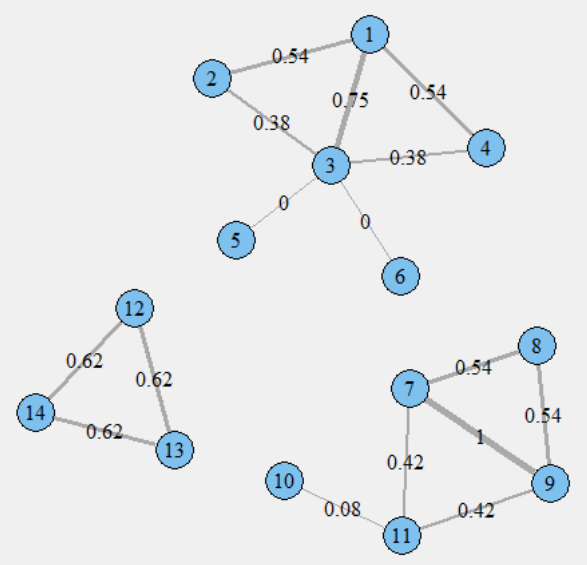

FIGURE 3.2 – Le graphe G' après la première itération

Liens	*liste_cov*	*liste_inertie*
9–7	1.00	0.30
3–1	0.75	0.38
13–12	0.62	0.39
14–12	0.62	0.42
14–13	0.62	0.48
2–1	0.54	0.50
4–1	0.54	0.53
8—7	0.54	0.57
9–8	0.54	0.64
11–7	0.41	0.63
11–9	0.41	0.65
3–2	0.37	0.65
4–3	0.37	*0.70*
11–10	0.08	0.49
5–3	0.00	0.23
6–3	0.00	–

Tableau 3.2 – Valeurs de covariances et de l'inertie inter-classes durant la deuxième itération

3.2 La deuxième phase

La deuxième phase vise à combiner les sous-réseaux obtenus dans la première phase afin de trouver la structure de communautés optimale. Combiner deux sous-réseaux implique l'ajout de liens (se trouvant dans le réseau initial) entre les nœuds de ces sous-réseaux afin de former un seul réseau connexe. Pour k sous-réseaux, il existe $k*(k-1)/2$ combinaisons possibles. Dans le but de rendre l'algorithme plus rapide, plusieurs combinaisons peuvent être évitées. En effet, la modularité du réseau, après une combinaison de deux sous-réseaux qui n'ont initialement pas de liens entre eux, ne va pas augmenter la modularité. Donc, le nombre de combinaisons à tester est largement inférieur à $k*(k-1)/2$. À chaque itération, la combinaison à traiter est choisie aléatoirement parmi l'ensemble des combinaisons restantes.

Pour chaque combinaison, il faut estimer la valeur de la modularité du réseau et garder seulement la combinaison ayant la plus grande valeur de la modularité. Cette dernière combinaison est admise et la fusion de deux sous-réseaux a lieu si la valeur de la modularité de cette fusion est supérieure à la valeur de la précédente. Dans le cas contraire, il ne faut pas procéder à la fusion et les sous-réseaux obtenus représentent les différentes communautés du réseau initial.

3.2.1 La modularité

La modularité de Newman [NG04], présentée en sous-section 2.1.1, est un indice de la qualité du partitionnement d'un graphe donné. Elle est définie par l'équation 2.4. Cet indice est important si le nombre de liens entre les communautés est faible et le nombre de liens intra-communautés est élevé. La meilleure structure de communautés est celle qui maximise la modularité. Une étude comparative a d'ailleurs montré l'efficacité de la fonction de modularité de Newman par rapport à d'autres fonctions de qualité de partitionnement comme *Modularity ration*, *Volume* et *Edge cut* [LLM10]. Nous utilisons donc la modularité de Newman dans la deuxième phase pour identifier la structure optimale de communautés.

3.2.2 Algorithme de la deuxième phase

Cet algorithme a deux paramètres d'entrées à savoir le graphe $G' = (V, E')$, obtenu de la première phase, et le graphe initial $G = (V, E)$. L'algorithme retourne un ensemble de sous-réseaux $G_1, G_2, G_3, ..., G_{NSR}$ représentant les communautés détectées. Avant de commencer la phase itérative de combinaison, il faut avoir un nombre de sous-réseaux

supérieur à un. Si le nombre de sous-réseaux est égal à un, c'est la fin l'algorithme de la deuxième phase. Dans la boucle *tant que*, de la ligne 2 à la ligne 20 de l'algorithme 3, les valeurs de modularité sont calculées pour les différentes combinaisons possibles entre les sous-réseaux adjacents. La structure optimale et sa modularité durant cette itération sont sauvegardées respectivement dans $Gtemp1$ et $mod2$. La procédure de combinaison est assurée par la fonction $CombinerSousGraphes(G_i, G_j, G)$ qui permet de fusionner les sous-réseaux G_i et G_j se trouvant dans G en un seul sous-réseau. Le calcul de la modularité se fait à travers la fonction $CalculerModularite(G)$. Les étapes décrites par la procédure suivante illustrent le processus de combinaison de sous-réseaux. L'algorithme 3 décrit les étapes de la deuxième phase.

3.2.3 Un exemple illustratif

Dans cette section nous poursuivons l'exécution de la deuxième partie de notre algorithme en nous basant sur le même exemple du graphe déjà exécuté avec la première partie (cf. sous-section 3.1.4). Le graphe est constitué initialement de quatorze nœuds et vingt-quatre liens et il est connexe. La première phase a retourné un graphe de dix liens et six sous-réseaux non-connectés (figure 3.3). Au début de la deuxième phase, les liens qui se trouvent à l'intérieur de chaque sous-réseau et qui ont été éliminés durant la première phase sont ajoutés. Le graphe au début de la deuxième phase est représenté par la figure 3.4. Avec cette structure, la modularité Q du graphe vaut 0.32.

Au cours de la première itération, le nombre de sous-réseaux NSR est égal à six. On calcule la modularité Q pour les combinaisons possibles. Une valeur maximale de Q égale à 0.36 est obtenue en fusionnant le sous-réseau qui contient le nœud 10 avec le sous-réseau qui contient les nœuds 7, 8, 9 et 11. La fusion de ces deux sous-réseaux consiste à ajouter tous les liens entre les nœuds de ces deux sous-réseaux qui se trouvent dans le graphe initial $G = (V, E)$. La modularité du graphe de la figure 3.5 est 0.36 alors que la modularité du graphe de la figure 3.4 est 0.32. Dans ce cas, nous gardons la structure du graphe de la figure 3.5 car sa modularité est bien supérieure.

Le processus itératif de fusion se poursuit pour avoir une modularité de 0.40 avec le graphe de la figure 3.6 et une modularité de 0.45 avec le graphe de la figure 3.7.

A ce stade, le nombre de sous-réseaux ($NSR = 3$) est toujours supérieur à un. En essayant les trois combinaisons possibles entre les sous-réseaux restants, les trois modularités correspondantes aux différentes combinaisons (0.19, 0.33 et 0.37) sont inférieures à la modularité de l'itération précédente (Q=0.45). Le processus de fusion s'arrête donc à ce point, ce qui signifie la fin de la deuxième phase de la méthode proposée. Les commu-

Algorithme 3 : *Algorithme de la deuxième phase*

Entrées : $G = (V, E)$: le graphe initial, $G' = (V, E')$: le graphe obtenu à la première phase constitué d'un ensemble de sous-réseaux non connectés entre eux $\{G_1, G_2, G_3, ..., G_{NSR}\}$.

Sorties : $G_1, G_2, G_3, ..., G_{NSR}$: des sous-réseaux représentant les différentes communautés détectées.

Variables : NSR : le nombre de sous-réseaux dans G', $Gtemp1$ et $Gtemp2$: deux graphes, *tab_modularite* : tableau de modularité, $mod1$l, $mod2$l, et $mod3$: valeurs de modularité.

```
1  début
2      tant que NSR > 1 faire
3          Gtemp1 = G' ;
4          mod1 = CalculerModularite(Gtemp1) ;
5          mod2 = −1 ;
6          mod3 = −1 ;
7          pour i de 1 à NSR − 1 faire
8              pour j de i + 1 à NSR faire
9                  si (adjacent(G_i, G_j, G)) alors
10                     Gtemp2 = G'\{G_i, G_j} ⋃ CombinerSousGraphes(G_i, G_j, G) ;
11                     mod3 = CalculerModularite(Gtemp2) ;
12                     si (mod2 < mod3) alors
13                         mod2 = mod3 ;
14                         Gtemp1 = Gtemp2 ;
15         si (mod1 < mod2) alors
16             mod1 = mod2 ;
17             G' = Gtemp1 ;
18         sinon
19             retourner {G_1, G_2, G_3, ..., G_NSR} ;
20     retourner {G_1, G_2, G_3, ..., G_NSR} ;
```

Algorithme 4 : *Procédure CombinerSousGraphes(G_i,G_j,G)*

Entrées : $G_i = (V_i, E_i), G_j = (V_j, E_j)$: les deux sous-réseaux à combiner ;
$G = (V, E)$: le graphe initial.

Sorties : $Gtemp = (Vtemp, Etemp)$: le sous-réseau après combinaison.

```
1 début
2     Vtemp = Vi ⋃ Vj
3     pour chaque nœud vi dans Vi faire
4         pour chaque nœud vj dans Vj faire
5             si eij ∈ E alors
6                 Ajouter eij dans Etemp ;
7     retourner Gtemp
```

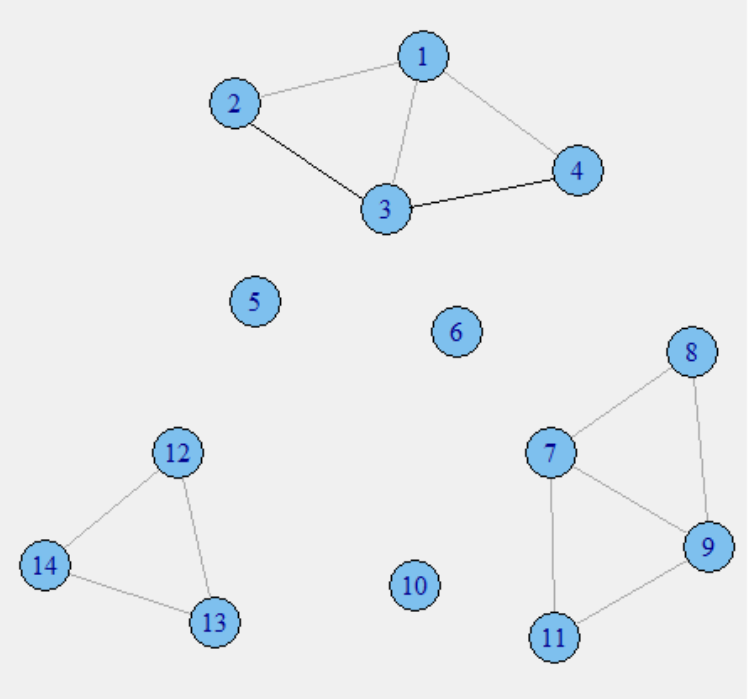

FIGURE 3.4 – La structure du graphe au début de la deuxième phase

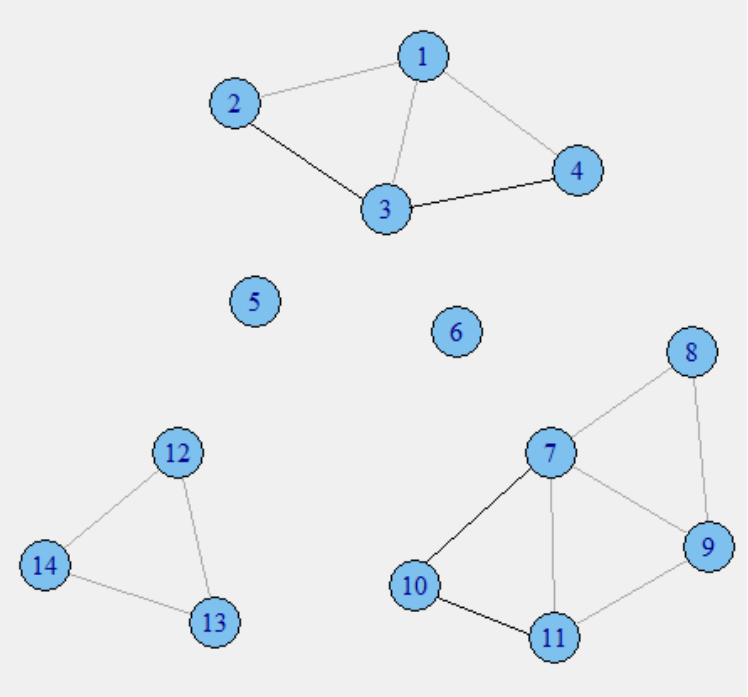

FIGURE 3.5 – La structure du graphe après la première itération de la deuxième phase

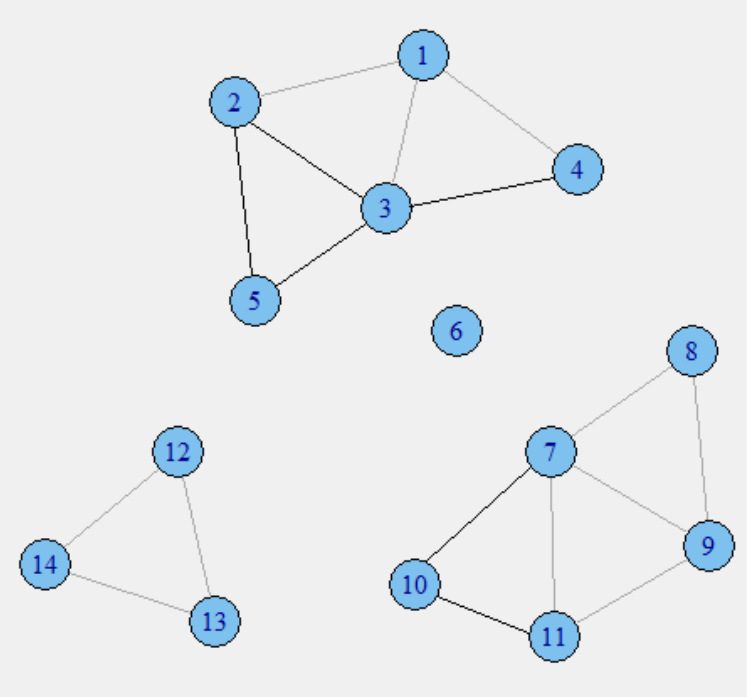

FIGURE 3.6 – La structure du graphe après la deuxième itération de la deuxième phase

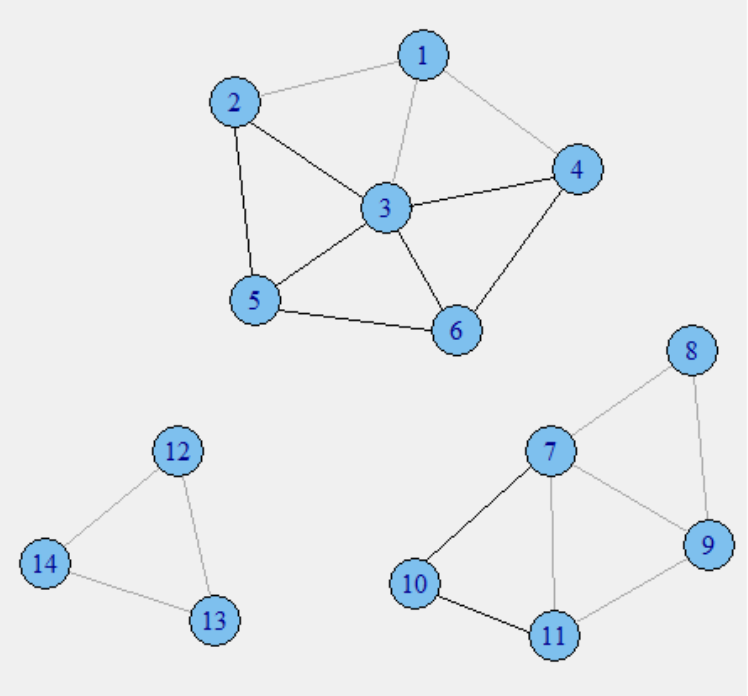

FIGURE 3.7 – La structure du graphe après la troisième itération de la deuxième phase

nautés détectées dans le graphe $G = (V, E)$ sont les derniers sous-réseaux non connectés. La configuration finale du réseau en trois communautés est représentée en figure 3.7.

3.3 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode de détection de communautés dans les réseaux sociaux. La méthode proposée contient deux phases. Dans la première phase, la mesure de covariance est utilisée pour éliminer les liens inter-communautés et diviser le réseau initial en sous-réseaux. La deuxième phase consiste à trouver la structure de communautés optimale en gardant les fusions entre les sous-réseaux qui maximisent la modularité de Newman.

Chapitre 4

Évaluation de la méthode proposée

Ce chapitre présente une évaluation empirique de la méthode proposée de détection de communautés sur des réseaux synthétiques et réels. La performance de notre méthode est comparée à quatre algorithmes connus dans la littérature, soit *Walktrap* [PL05], *Edge Betweenness* [NG04], *Fast Greedy* [New04] et *Label Propagation* [RAK07].

4.1 Expérimentations sur les réseaux synthétiques

4.1.1 Génération des réseaux

Nous utilisons un modèle de génération de réseaux artificiels défini dans [LF09b, LF09a]. Ce modèle a deux paramètres principaux : Pin et $Pout$. Le premier est la probabilité d'avoir un lien entre deux nœuds se trouvant dans la même communauté tandis que le deuxième se définit comme la probabilité d'avoir un lien entre deux nœuds de différentes communautés. Le nombre n de nœuds dans le réseau, la moyenne d de degrés de centralité (nombre de nœuds voisins), et la centralité de degré maximale autorisée d_{max} sont d'autres paramètres d'entrée du modèle. En outre, le modèle utilise les paramètres α et β qui sont respectivement l'exposant de la distribution de la centralité de degré et l'exposant de la distribution de la taille des communautés. Le paramètre le plus intéressant dans la génération aléatoire des réseaux est le paramètre de mixage (*mixing parameter*) mp. Ce dernier représente la moyenne (pour tous les nœuds) du rapport entre le nombre de liens propres à un nœud se trouvant à l'extérieur de sa communauté et le nombre total de liens de ce nœud. Lorsque cette valeur est faible, les communautés au sein du réseau sont bien distinguées et peuvent être vues même visuellement. Dans le cas contraire, les communautés se superposent et ont plusieurs nœuds en commun, ce qui rend leur identification plus difficile. À titre illustratif, les réseaux présentés par les figures 4.1, 4.2 et

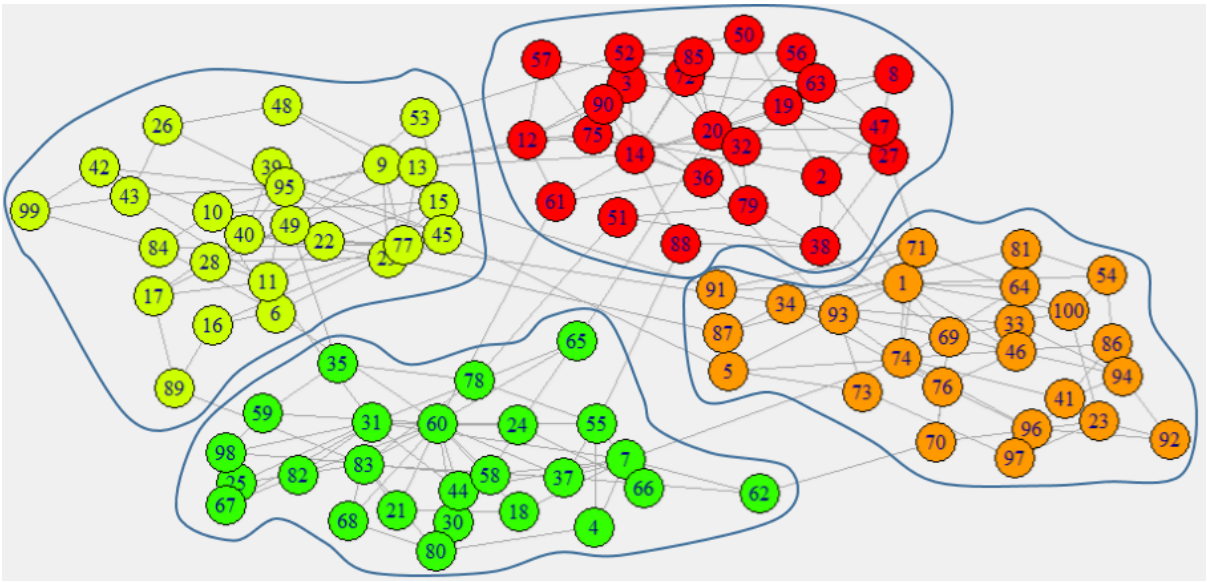

FIGURE 4.1 – Exemple de réseau généré avec mp=0.1

4.3 contiennent 100 nœuds et sont générés avec des paramètres de mixage de 0.1, 0.3 et 0.5 respectivement. Les différentes communautés dans chaque réseau sont encerclées et représentées par différentes couleurs.

Pour assurer une meilleure fiabilité des résultats empiriques, nous avons généré aléatoirement pour chaque même ensemble de paramètres d'entrée 25 réseaux et avons estimé le comportement moyen pour chacune des méthodes de détection de communautés. La variation retenue de la complexité des réseaux se base sur les valeurs suivantes de paramètres : n=100, $dmax$=30, α=3, β=2 et mp=0.1*i avec i variant de 1 à 9. Cela a permis de générer 225 (=9*25) réseaux. Le nombre de communautés a été fixé à quatre pour tous les réseaux.

Le générateur de réseaux aléatoire fournit également des fichiers contenant les communautés auxquelles appartient chaque nœud de chaque réseau. Cela facilite par la suite la tâche de comparaison de la structure de communautés identifiée par chaque algorithme avec la structure initiale. Cette démarche a été utilisée dans [OL09] dans le but de comparer cinq algorithmes.

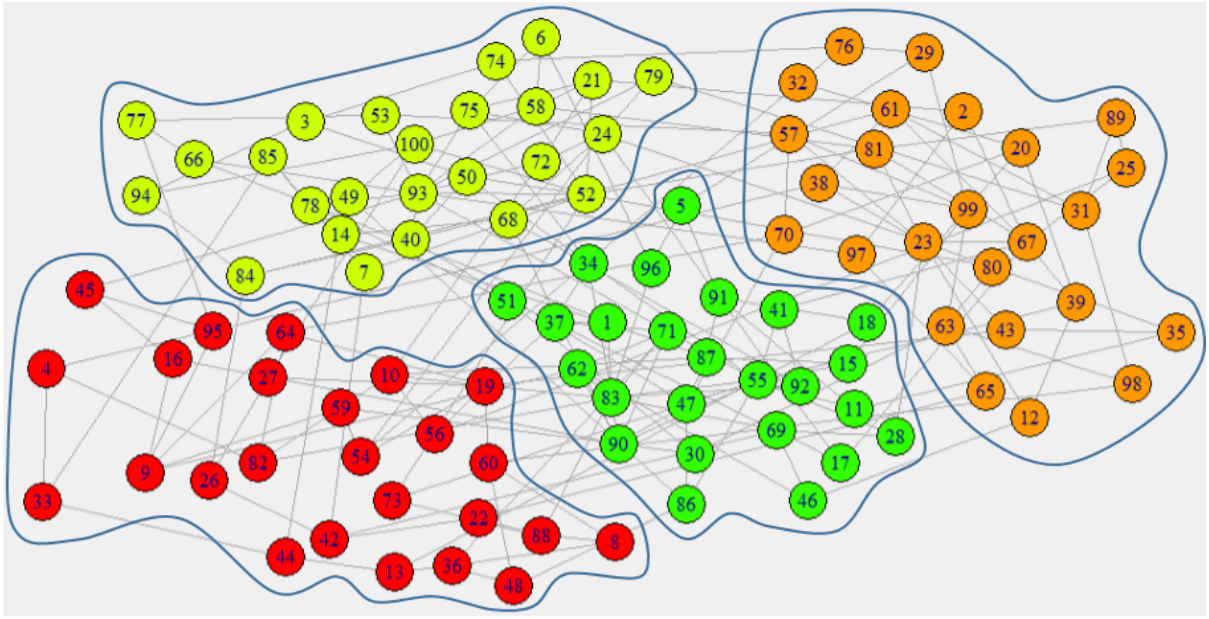

FIGURE 4.2 – Exemple de réseau généré avec mp=0.3

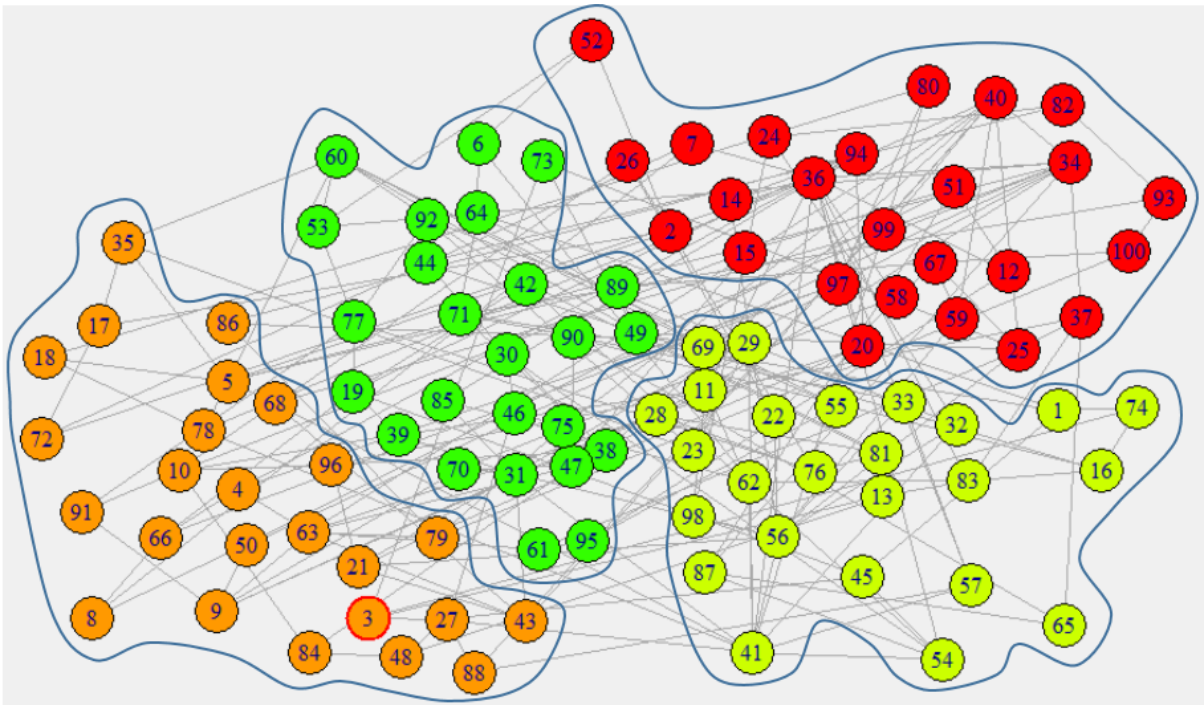

FIGURE 4.3 – Exemple de réseau généré avec mp=0.5

4.1.2 Comparaison des résultats

Nous rappelons que la méthode proposée est comparée avec celles présentées dans le chapitre 2, soit *Walktrap*, *Edge Betweenness*, *Fast Greedy* et *Label Propagation*. Le choix de ces algorithmes est dicté par le fait qu'ils représentent d'une part les différents types d'algorithmes (agglomératif, divisif, à base du modèle et heuristique) de détection de communautés, et d'autre part, ils sont également très connus et considérés comme les plus performants et de bons étalons pour une comparaison de performance tant en terme de précision du processus de délimitation des communautés que des temps d'exécution [For10, Fer12].

Pour chaque réseau généré, l'algorithme proposé dans ce mémoire ainsi que les quatre algorithmes choisis pour la comparaison sont exécutés. Les structures de communautés dégagées sont ensuite comparées à la structure de communautés suggérée par la procédure de génération aléatoire de réseaux. Afin de comparer ces résultats, une mesure de qualité de la partition trouvée par chaque méthode est nécessaire. Plusieurs mesures peuvent être utilisées telles que *Rand Index* [YS04], Jaccard [YS04], *Purity* et la mesure F [Bra10, RH07]. Nous avons choisi la mesure F car elle est la plus utilisée. Cette dernière permet de mesurer le degré de similarité entre deux partitions, à savoir la partition initialement suggérée et la partition dégagée par un algorithme de détection de communautés. Cette mesure est donc utilisée pour comparer les différents résultats trouvés avec la structure de communautés suggérée par l'algorithme de génération de réseaux synthétiques.

La mesure *F*

Pour utiliser cette mesure, nous supposons que C'=$C'_1, C'_2, C'_3 ... C'_{k'}$ est l'ensemble initial de communautés identifiées par le modèle de génération de réseaux artificiels et que C=$C_1, C_2, C_3 ... C_k$ est l'ensemble de communautés déterminées par un des algorithmes à comparer. Initialement, chaque nœud du réseau fait partie d'une seule communauté C'_i. À la fin de l'algorithme, chaque nœud du réseau fait partie d'une seule communauté C_j. Un élément a_{ij} de la matrice de correspondance $MC = [i, j]$ représente le nombre de nœuds se trouvant à la fois dans la classe C'_i et C_j. Donc, i varie de 1 à k' et j varie de 1 à k, où k' et k sont respectivement le nombre de communautés initialement identifiées et le nombre de communautés calculées par l'un des algorithmes.

On rappelle que la mesure F retenue pour l'évaluation de la correspondance entre deux communautés C'_i et C_j est définie par l'équation suivante :

$$F(C'_i, C_j) = \frac{2 * R(C'_i, C_j) * P(C'_i, C_j)}{R(C'_i, C_j) + P(C'_i, C_j)} \tag{4.1}$$

Où $R(C'_i, C_j)$ et $P(C'_i, C_j)$ représentent respectivement le rappel et la précision.

$$R(C'_i, C_j) = \frac{a_{ij}}{|C'_i|} \tag{4.2}$$

$$P(C'_i, C_j) = \frac{a_{ij}}{|C_j|} \tag{4.3}$$

Afin de déterminer la mesure F des structures C et C' entières de communautés, on fait appel à l'équation 4.1 donnée précédemment :

$$F(C', C) = \sum_{C'_i} \frac{|C'_i|}{n} \max\left(F(C'_i, C_j)\right) \tag{4.4}$$

Puisque le générateur de réseaux fournit pour chaque réseau les communautés auxquelles appartient un nœud, la mesure F est utilisée pour comparer les communautés détectées par les algorithmes avec la structure initiale de communautés. Cette mesure varie entre 0 et 1. Si la mesure F pour une structure donnée est proche de 1, cette dernière est proche de la structure initiale et donc l'algorithme qui l'a identifiée a donné de bons résultats. Par contre, si la mesure F est proche de 0, la décomposition identifiée est distincte de la décomposition originale en communautés, et l'algorithme est peu précis.

Plan d'expérimentations

Le comportement de la méthode proposée est étudié selon la variation des quatre paramètres suivants :
- la taille des réseaux
- leur densité
- le nombre de communautés (nc)
- le degré de chevauchement entre les communautés via le paramètre de mixage (mp) qui représente la proportion moyenne de liens entre un nœud et les nœuds situés à l'extérieur de sa communauté.

L'outil de génération de réseaux [SAN13] est installé sur la plateforme *Ubuntu* 10.04.4. Il est utilisé sur une machine qui a un processeur $Core2Duo$ d'$Intel$ (1.80 GHz) et une mémoire vive de 2 Go. L'exécution des algorithmes est faite avec le logiciel R [JMSY92], un logiciel gratuit de type *opensource* (code source ouvert). R sert à la fois à manipuler des données, tracer des graphiques et faire des analyses statistiques. Le logiciel R inclut

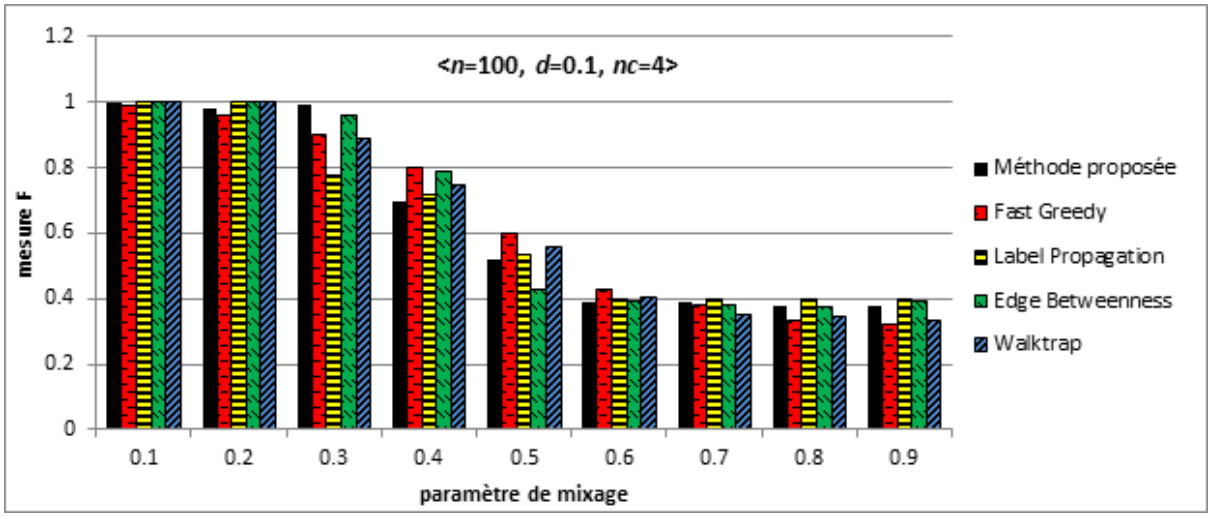

FIGURE 4.4 – Valeurs de la mesure F en fonction du paramètre de mixage pour des réseaux de $<n$=100, d=0.1, nc=4$>$

le module *igraph* [CRA13] spécialement conçu pour l'analyse et la visualisation des réseaux. Dans ce dernier, les quatre algorithmes étudiés (*Walktrap*, *Edge Betweenness*, *Fast Greedy* et *Label Propagation*) sont déjà prédéfinis. Nous avons donc développé la méthode proposée à l'aide du langage R afin d'utiliser la même plateforme de programmation pour tous les algorithmes. La documentation complète, les paquets (*packages*) et les sources d'installation propres au logiciel R sont disponibles pour le lecteur [RPR13].

Variation de la taille des réseaux

Le premier aspect à étudier est l'effet du changement de la taille du réseau sur la qualité de détection de communautés par la méthode proposée et les algorithmes retenus. Quatre différentes valeurs de n (nombre de nœuds du réseau) ont été déployées, soit 100, 1 000, 3 000 et 5 000. Initialement, la densité des réseaux est fixée à 0.1 (près de 500 liens avec un réseau de 100 nœuds et 1 250 000 liens avec un réseau de 5 000 nœuds), et le nombre de communautés nc est mis à quatre. Pour chaque valeur de n, on fait varier la complexité de détection de communautés en choisissant un paramètre de mixage passant de 0.1 à 0.2,..., 0.9. En effet, si le paramètre de mixage mp est proche de 0.1, le nombre de liens inter-communautés est faible et la détection de communautés est facile. Si le mp est plutôt proche de 0.9, le nombre de liens inter-communautés est élevé et la détection de communautés est plus difficile. Pour un même ensemble de paramètres donnés (n, d, nc et mp), on a généré 25 réseaux et la moyenne de 25 mesures F calculées pour chaque algorithme est retenue.

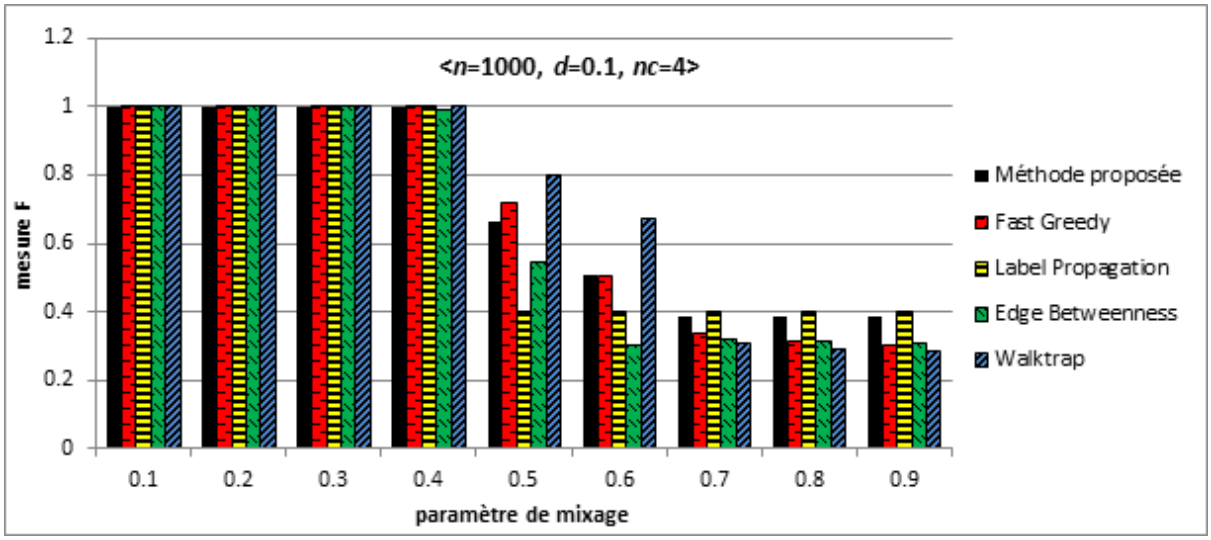

FIGURE 4.5 – Valeurs de la mesure F en fonction du paramètre de mixage pour des réseaux de $<n$=1 000, d=0.1, nc=4$>$

Comme illustré par la figure 4.4, les diverses méthodes donnent des résultats comparables lorsque le paramètre de mixage est inférieur à 0.4. Pour des valeurs du paramètre de mixage supérieures à 0.4, la qualité de tous les résultats se dégrade. Lorsque le paramètre de mixage vaut 0.4, 0.5 ou 0.6, la méthode proposée ne fournit pas les meilleurs résultats comme les algorithmes *Fast Greedy* et *Label Propagation*, mais des résultats assez proches d'eux. La méthode proposée et *Label propagation* ont les meilleurs résultats avec des valeurs du paramètre de mixage supérieures à 0.6.

Dans le deuxième ensemble de réseaux générés, il s'agit de la même densité et du même nombre de communautés, mais le nombre de nœuds est fixé à 1 000. Les mesures F pour tous les algorithmes avec ces réseaux sont représentées par la figure 4.5. Pour les différentes valeurs du paramètre de mixage inférieures à 0.5, toutes les méthodes ont donné la plus grande valeur de la mesure F. Avec un paramètre de mixage égal à 0.5 ou 0.6, les résultats de la méthode proposée sont moins bons que *Walktrap* et *Fast Greedy*. Dans les cas les plus complexes et avec un paramètre de mixage supérieur à 0.6, la méthode proposée est la meilleure avec *Label Propoagation*.

En augmentant davantage le nombre de nœuds dans les réseaux générés (3 000 et 5 000 nœuds), les résultats de tous les algorithmes, spécifiquement la méthode proposée, sont améliorés. En effet, la méthode proposée a toujours la mesure F la plus élevée pour un paramètre de mixage allant de 0.1 à 0.9 à l'exception des réseaux de 3 000 nœuds avec un paramètre de mixage de 0.4 (figure 4.6). Les valeurs de la mesure F pour tous les algorithmes ont augmenté avec l'accroissement du nombre de nœuds. Par exemple,

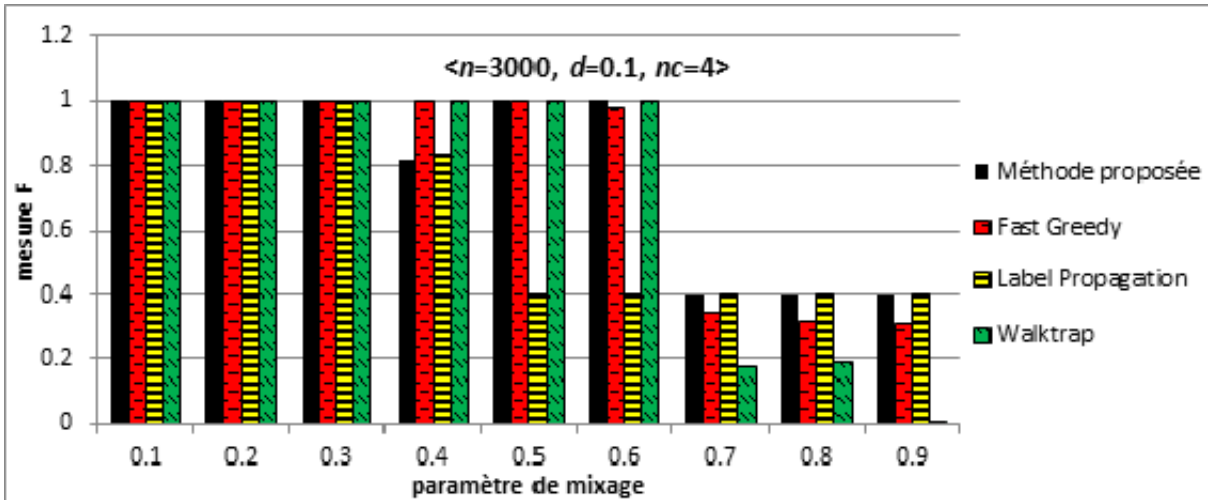

FIGURE 4.6 – Valeurs de la mesure F en fonction du paramètre de mixage pour des réseaux de $<n$=3 000, d=0.1, nc=4$>$

dans la figure 4.4 (n=100) et pour un paramètre de mixage supérieur à 0.3, toutes les mesures F n'ont pas dépassé le seuil de 0.8 alors que dans les figures 4.6 et 4.7 (n=3 000 et n=5 000), la méthode proposée et *Walktrap* ont une mesure F égale à 1 avec un paramètre de mixage de 0.6.

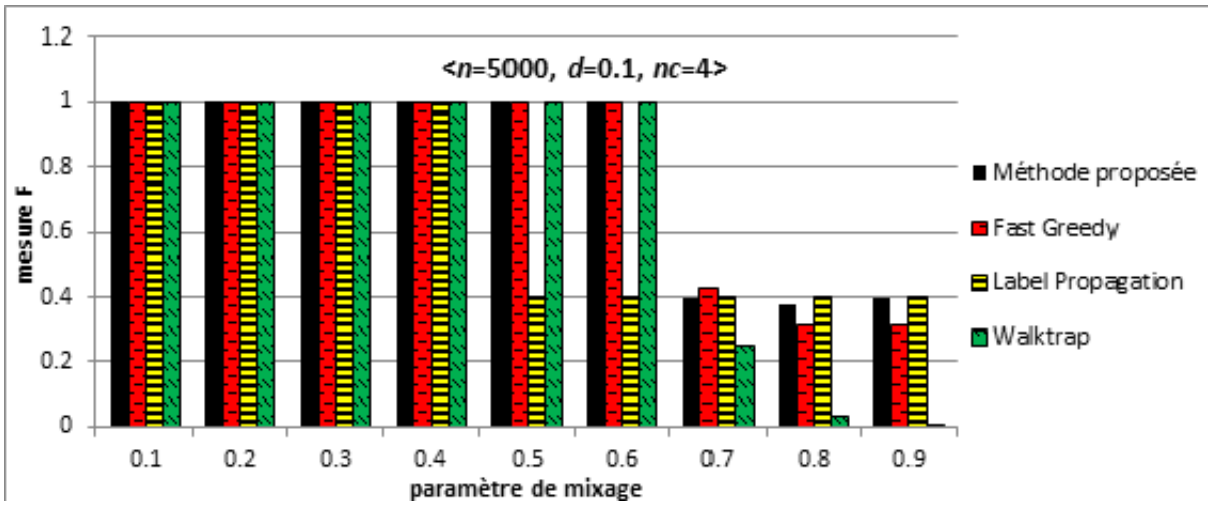

FIGURE 4.7 – Valeurs de la mesure F en fonction du paramètre de mixage pour des réseaux de $<n$=5 000, d=0.1, nc=4$>$

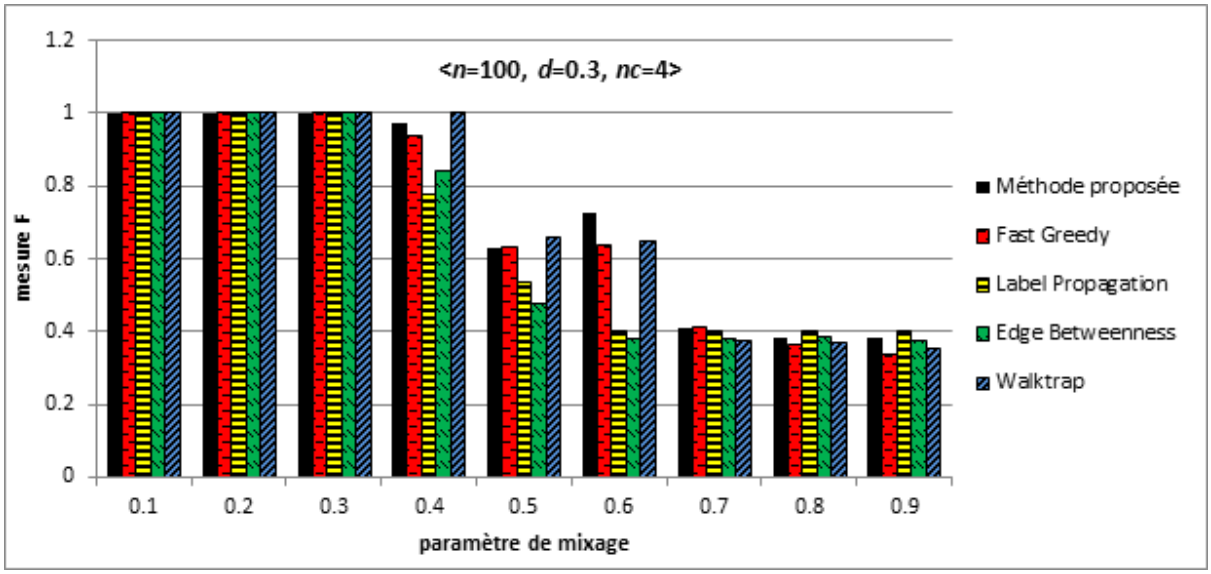

FIGURE 4.8 – Valeurs de la mesure F en fonction du paramètre de mixage pour des réseaux de $<n$=100, d=0.3, nc=4$>$

Variation de la densité des réseaux

La taille des réseaux est fixée maintenant à 100 nœuds et le nombre de communautés à quatre. Les réseaux à tester dans la figure 4.8 ont trois fois plus de liens (près de 1 500 liens) que les réseaux de la figure 4.6 puisque la densité d est égale à 0.3. La méthode proposée a donné les meilleurs résultats pour les différents ratios de liens inter-communautés à l'exception des valeurs 0.4 et 0.5 du paramètre de mixage où *Walktrap* a donné de meilleurs résultats (figure 4.8).

Nous avons ensuite généré des réseaux avec une densité de 0.5, soit 2 500 liens pour 100 nœuds et quatre communautés. Pour un paramètre de mixage de 0.6, la méthode proposée, *Walktrap* et *Fast Greedy* ont des mesures F plus enlevées (figure 4.9) qu'avec la figure 4.8. Lorsque le paramètre de mixage est de 0.5 (respectivement 0.7), la méthode le deuxième (respectivement le troisième) meilleur résultat. Dans le reste des cas, elle a donné la meilleure valeur de la mesure F (figure 4.9).

En augmentant la densité avec des réseaux de 1 000 nœuds pour avoir 150 000 liens (d=0.3) puis 250 000 liens (d=0.5) tout en gardant cinq communautés, nous obtenons les résultats montrés par les figures 4.10 et 4.11. Dans les deux figures, la méthode proposée donne les valeurs les plus élevées des mesures F à l'exception de la figure 4.10 avec un paramètre de mixage de 0.7 où *Walktrap* donne le meilleur résultat. La qualité des résultats trouvés par les différents algorithmes s'améliore avec l'augmentation de la densité. Dans

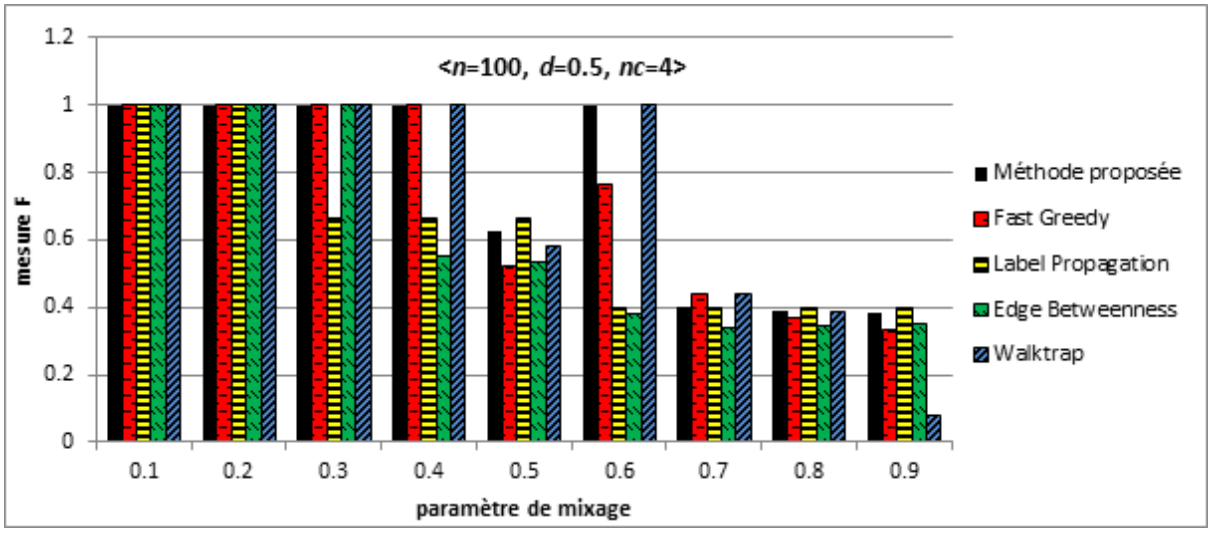

FIGURE 4.9 – Valeurs de la mesure F en fonction du paramètre de mixage pour des réseaux de $<n$=100, d=0.5, nc=4$>$

la figure 4.5 (d=0.1), et à partir d'un paramètre de mixage égal à 0.5, toutes les mesures F ne dépassent pas 0.8 alors que dans la figure 4.10 (d=0.3) et la figure 4.11 (d=0.5), les mesures F valent 1 pour un paramètre de mixage de 0.7.

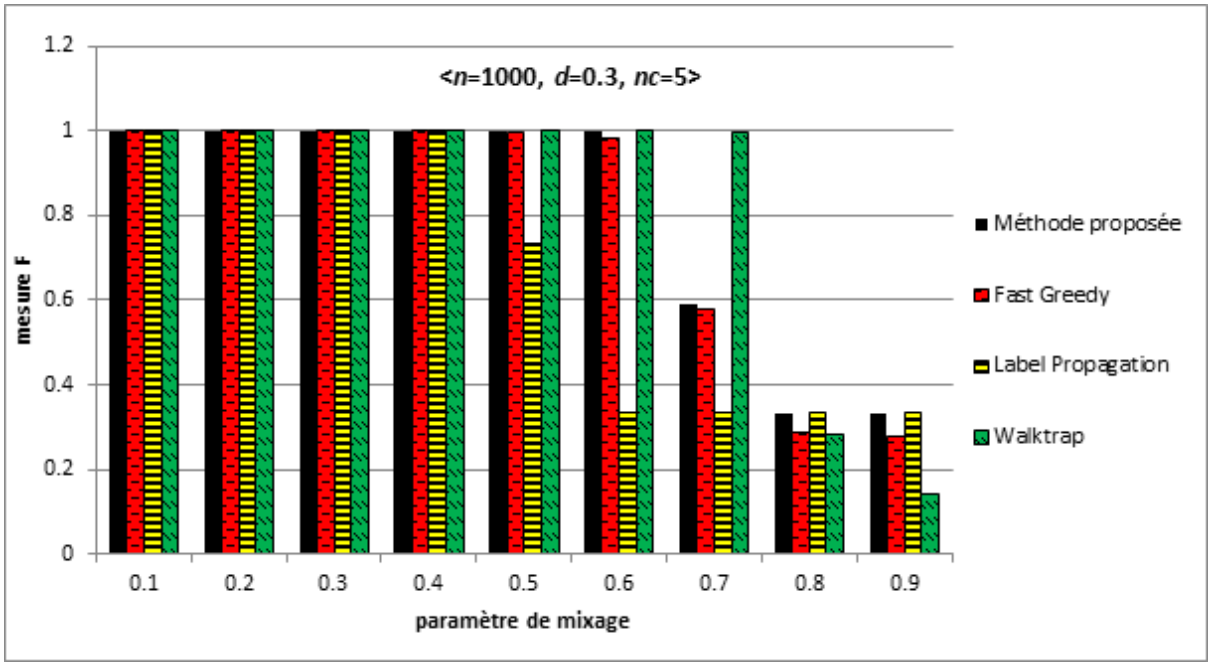

FIGURE 4.10 – Valeurs de la mesure F en fonction du paramètre de mixage pour des réseaux de $<n$=1 000, d=0.3, nc=5$>$

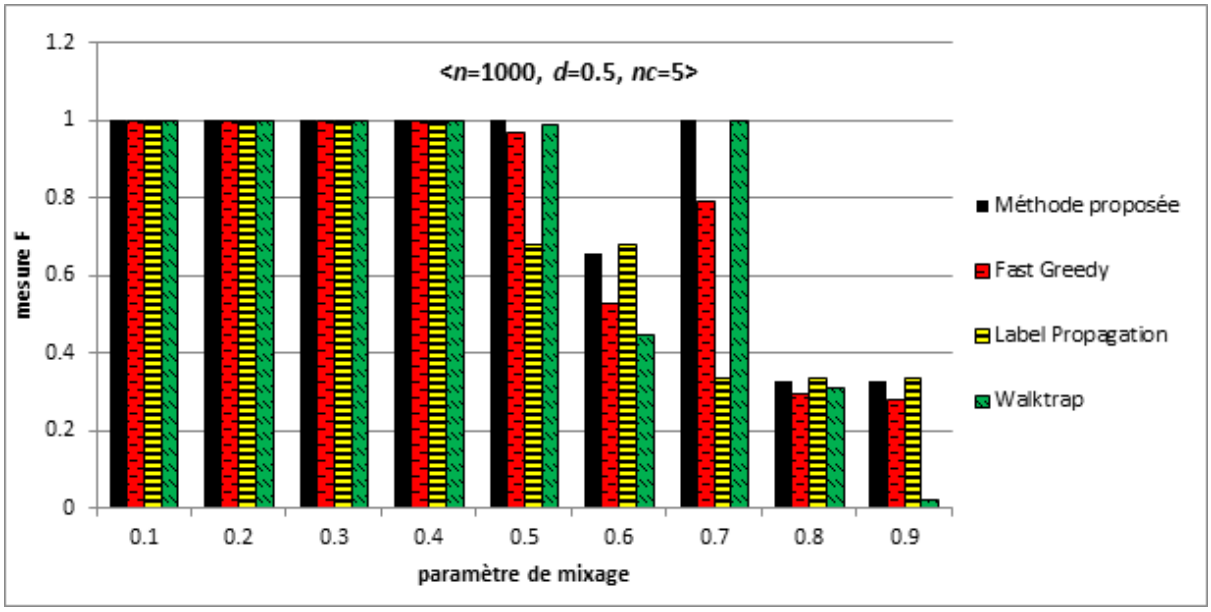

FIGURE 4.11 – Valeurs de la mesure F en fonction du paramètre de mixage pour des réseaux de <n=1 000, d=0.5, nc=5>

Variation du nombre de communautés dans les réseaux

Le dernier paramètre à modifier est le nombre de communautés nc. Tous les algorithmes sont exécutés sur des réseaux de 100 nœuds et huit communautés. Les mesures F calculées dans ce cas sont représentées dans la figure 4.12. Pour des valeurs du paramètre de mixage supérieures à 0.6, toutes les méthodes sont proches avec une supériorité infime pour l'algorithme *Fast Geedy*. Dans les autres cas et pour un paramètre de mixage inférieur à 0.7, la méthode proposée a toujours donné la meilleure mesure F avant *Walktrap*.

Dans le dernier ensemble de réseaux à tester, le nombre de communautés est fixé à dix avec un nombre de nœuds égal à 1 000 et une densité de 0.1. Les résultats trouvés avec ces réseaux sont représentés dans la figure 4.13. L'augmentation du nombre de communautés a influencé l'efficacité de tous les algorithmes de la même manière. En effet, pour des paramètres de mixage de 0.5, 0.6 et 0.7, les mesures F de tous les algorithmes (sauf *Label Propagation*) ont augmenté de la figure 4.5 (nc=4) à la figure 4.13 (nc=10). En regardant les mêmes figures (4.5 et 4.13), on remarque aussi que les valeurs de la mesure F ont baissé avec l'augmentation du nombre de communautés pour un paramètre de mixage supérieur à 0.7.

Dans quelques figures, les mesures F de l'algorithme *Edge Betweenness* ne sont pas incluses car cet algorithme a un temps d'exécution assez important pour des réseaux qui dépassent 50 000 liens.

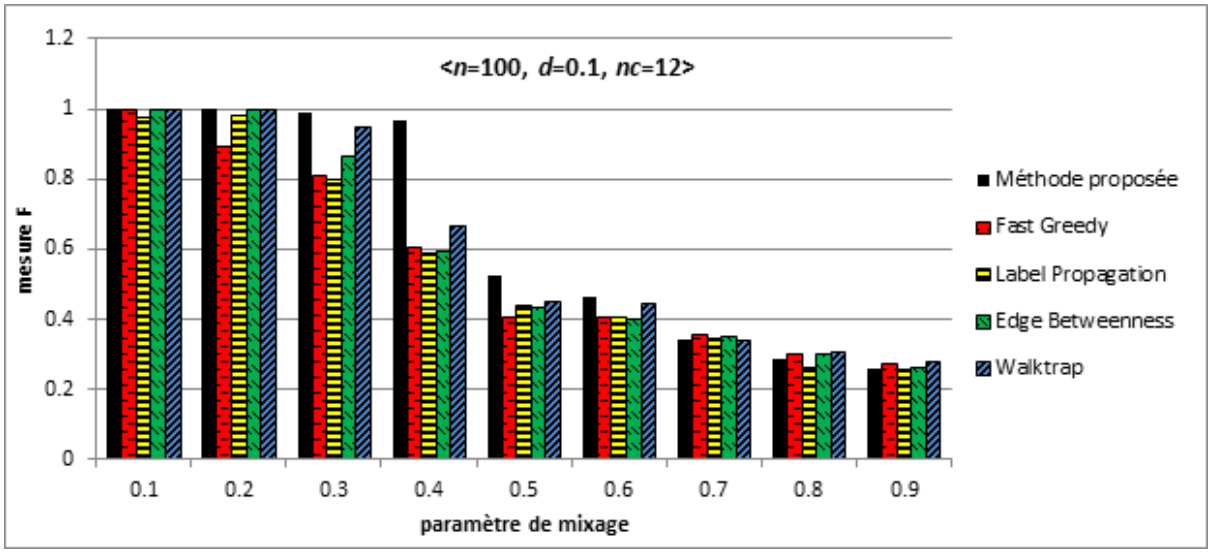

FIGURE 4.12 – Valeurs de la mesure F en fonction du paramètre de mixage pour des réseaux de $<n$=100, d=0.1, nc=12$>$

De ce qui précède, il ressort que tous les algorithmes et spécifiquement la méthode proposée sont plus efficaces avec des réseaux de grande taille et une densité allant jusqu'à 50%. Cependant, seule la méthode proposée maintient ses performances avec l'augmentation du nombre de communautés dans les réseaux (figures 4.4 et 4.12). En consultant tous les cas étudiés, on constate bien que la méthode proposée donne de bons résultats en ce qui concerne la qualité de communautés détectées. En effet, elle fournit presque dans les différentes situations la première ou la deuxième meilleure solution alors que ce n'est pas le cas pour les autres algorithmes.

La méthode proposée a également fait preuve d'une grande stabilité face aux différents types de réseaux. À titre d'exemple, avec un petit réseau, un nombre réduit de communautés, un nombre de liens inter-communautés faible et précisément un paramètre de mixage inférieur à 0.3 (cf. figure 4.4), notre méthode se compare avec *Edge Betweenness*. Tel qu'indiqué en figure 4.8, elle se compare avec *Walktrap* pour des paramètres de mixage inférieurs à 0.7, mais avec des communautés davantage reliées entre elles et des valeurs du paramètre de mixage supérieures à 0.7, elle se compare avec *Label Propagation*. Dans la figure 4.10, on constate que pour un réseau de grande taille, une densité plus élevée et pour des paramètres de mixage de 0.8 et 0.9, c.-à-d. avec un nombre de liens inter-communautés élevé, la méthode définie se compare avec *Fast Greedy* et *Label Propagation*. Dans la même figure 4.10 et pour un paramètre de mixage inférieur à 0.8, la méthode proposée se compare avec *Walktrap*.

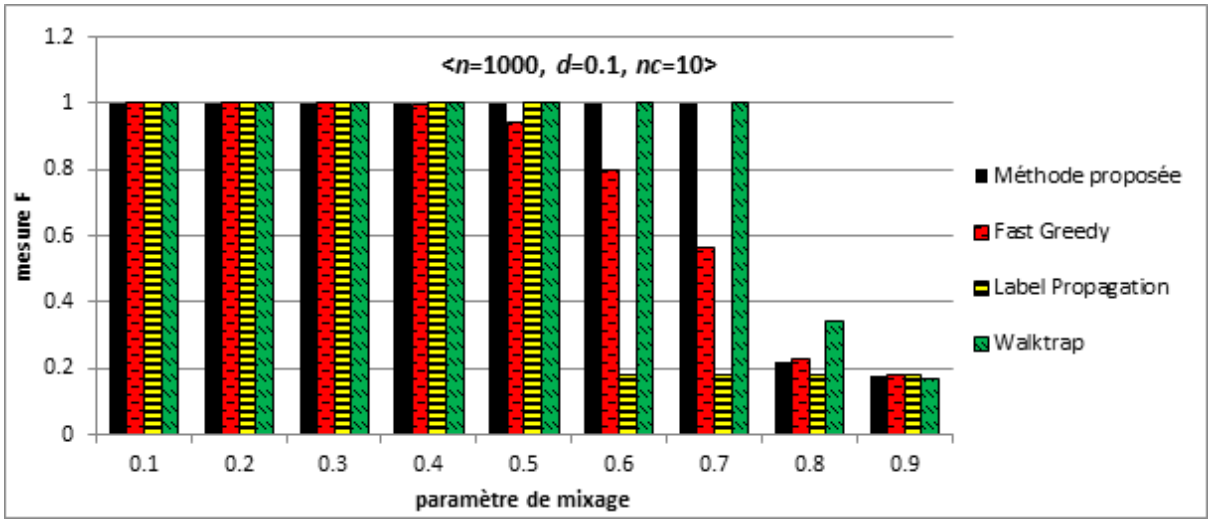

FIGURE 4.13 – Valeurs de la mesure F en fonction du paramètre de mixage pour des réseaux de $<n$=1 000, d=0.1, nc=10$>$

4.2 Expérimentations sur les réseaux réels

Outre les réseaux synthétiques, les algorithmes sont aussi testés sur des réseaux réels. Pour les réseaux dont la structure de communautés est connue à l'avance, nous avons calculé la mesure F pour toutes les solutions. Comme l'algorithme *Label Propagation* n'est pas déterministe, il est exécuté dix fois pour chaque exemple de réseau. La mesure F de cet algorithme est la moyenne des valeurs des dix solutions trouvées. La modularité de Newman est calculée aussi pour tous les résultats des algorithmes étudiés.

4.2.1 Club de karaté de Zachary

Le premier exemple est le club de karaté de Zachary [Zac77]. C'est un réseau construit à partir des relations entre 34 membres d'un club de karaté dans une université aux États-Unis. Il s'agit d'un réseau très populaire et très utilisé par plusieurs algorithmes afin de tester leurs performances puisque sa structure de communautés est connue à l'avance. Ce réseau comporte deux groupes comme l'illustre la figure 4.14. Les deux communautés sont séparées par une ligne verticale.

La méthode proposée a divisé ce réseau en quatre communautés. Les quatre communautés sont délimitées et représentées par différentes couleurs dans la figure 4.14. Cette structure dégagée est très proche de la structure existante à deux communautés. En effet, la fusion de deux communautés à gauche et de deux communautés à droite donne la même

structure initiale du réseau. En utilisant la mesure F, la méthode proposée a donné la meilleure structure de communautés après *Label Propagation* avec une mesure F=0.82. La moyenne de la mesure F pour *Label Propagation* vaut 0.85 avec un minimum de 0.63 et un maximum de 0.97. La figure 2.7 contient trois exemples de partitions trouvées par *Label Propagation*. Les valeurs de la mesure F pour les autres algorithmes sont inscrites dans le tableau 4.1. Mais, en regardant les valeurs de la modularité Q figurant dans le tableau 4.2, la méthode proposée a la meilleure modularité avec Q=0.41.

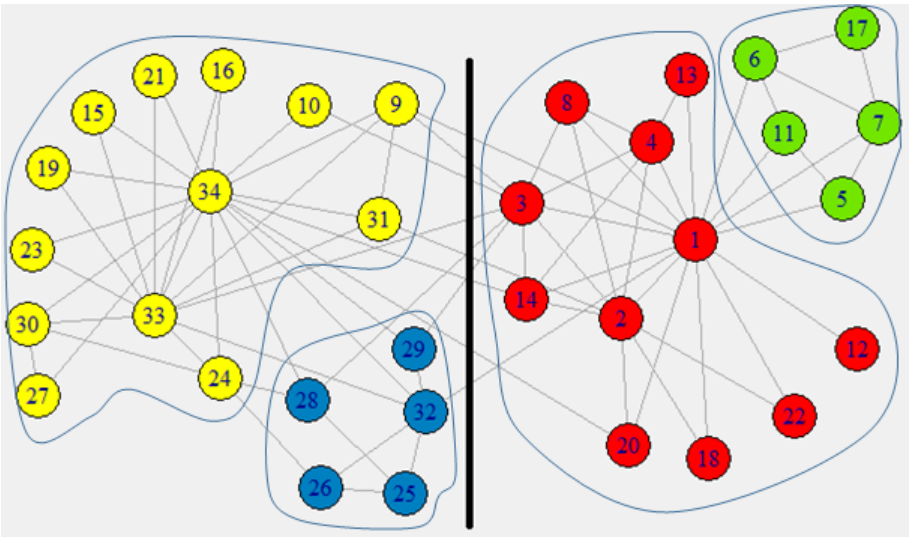

FIGURE 4.14 – Structure de communautés trouvée par la méthode proposée pour le réseau de Zachary

4.2.2 Les dauphins de Lusseau

Le deuxième exemple à traiter est le réseau de dauphins de Lusseau [LSB+03]. Ce réseau est constitué essentiellement de deux communautés qui sont séparées par une ligne verticale dans la figure 4.15. Ce réseau contient 62 nœuds et 159 liens. La valeur maximale de la mesure F est donnée par *Fast Greedy* (0.79) alors que la méthode proposée a une mesure F de 0.72. Les communautés détectées par *Fast Greedy* sont représentées dans la figure 4.16. La méthode proposée a détecté quatre communautés : la première contient les mêmes éléments de la première communauté initiale et les trois autres communautés contiennent les nœuds de la deuxième communauté initiale (figure 4.15). En ce qui concerne les mesures de modularité, la méthode proposée a la meilleure valeur de Q=0.52 que l'algorithme *Edge Betweenness* comme illustré par le tableau 4.2.

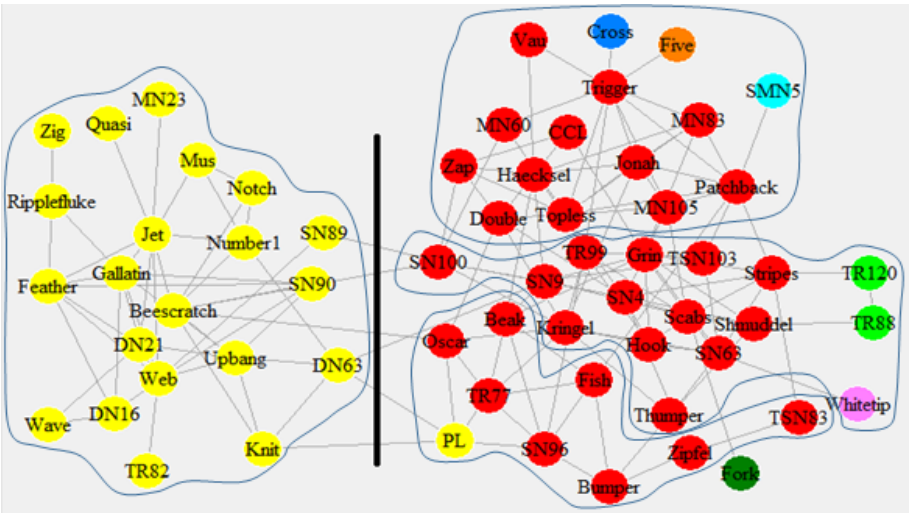

FIGURE 4.15 – les communautés détectées par la méthode proposée pour le réseau de dauphins de Lusseau

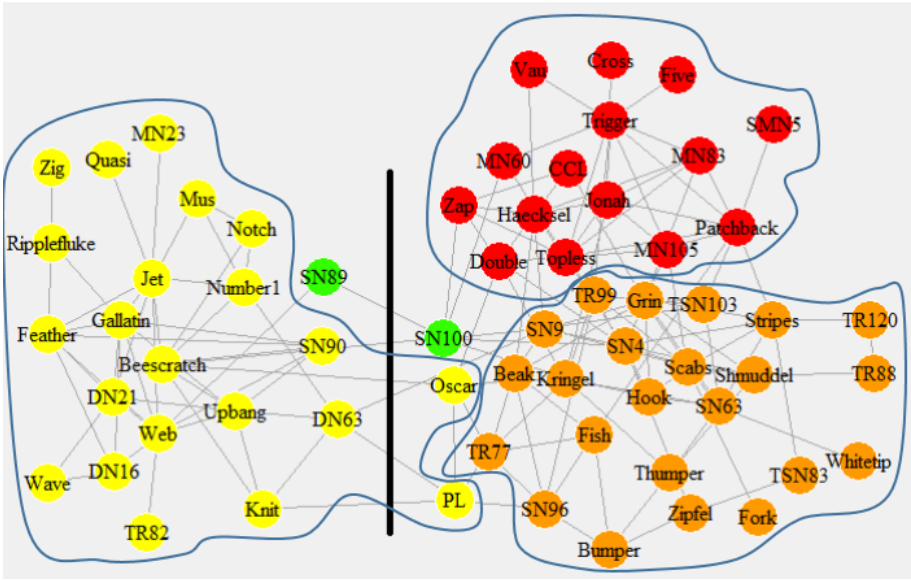

FIGURE 4.16 – Les communautés détectées par *Fast Greedy* pour le réseau de dauphins de Lusseau

4.2.3 Les livres politiques

Dans cette section nous allons traiter un autre type de relations. Il ne s'agit pas de relations entre les êtres humains ou les animaux, mais plutôt de relations entre l'achat de plusieurs titres sur Amazon.com [PER13]. Le réseau est constitué de 105 livres avec 441

liens se trouvant entre les livres achetés ensemble. Ce réseau a initialement trois groupes de livres : libéraux (l), conservateurs (c) et neutres (n). Les deux premiers groupes sont les plus intéressants. Ils ont ensemble 92 nœuds alors que le troisième compte 13 nœuds seulement. La méthode proposée a détecté quatre communautés qui sont encerclées dans la figure 4.17 et a abouti à une meilleure mesure de F égale à 0.84. Le deuxième meilleur résultat est trouvé par *Edge Betweenness* avec une mesure F=0.83 (cf. figure 4.18). Notre méthode a la meilleure valeur de Q=0.52 (cf. tableau 4.2).

Mesure F	Club de Zachary (n=34, m=78)	dauphins de Lusseau (n=62, m=159)	Livres politiques (n=105, m=441)	*Football* américain (n=115, m=615)	Championnat d'Angleterre (n=248, m=883)	Députés au Royaume-Uni (n=419, m=1 907)	Joueurs et clubs de rugby (n=854, m=3 361)
Méthode proposée	0.82	0.72	0.84	0.80	0.92	0.94	0.44
Edge Betweenness	0.78	0.78	0.83	0.84	0.91	0.83	0.45
Label Propagation	0.85	0.77	0.78	0.80	0.92	0.95	0.34
Fast Greedy	0.82	0.79	0.81	0.18	0.84	0.92	0.52
Walktrap	0.69	0.78	0.82	0.86	0.92	0.82	0.40

Tableau 4.1 – Valeurs de la mesure F pour les différents réseaux

Modularité Q	Club de Zachary (n=34, m=78)	dauphins de Lusseau (n=62, m=159)	Livres politiques (n=105, m=441)	*Football* américain (n=115, m=615)	Championnat d'Angleterre (n=248, m=883)	Députés au Royaume-Uni (n=419, m=1 907)	Joueurs et clubs de rugby (n=854, m=3 361)
Méthode proposée	0.41	0.52	0.52	0.60	0.86	0.63	0.88
Edge Betweenness	0.40	0.52	0.51	0.59	0.86	0.63	0.87
Label Propagation	0.35	0.41	0.50	0.60	0.85	0.62	0.84
Fast Greedy	0.38	0.49	0.50	0	0.85	0.62	0.85
Walktrap	0.35	0.48	0.50	0.60	0.86	0.63	0.85

Tableau 4.2 – Valeurs de la modularité pour les différents réseaux

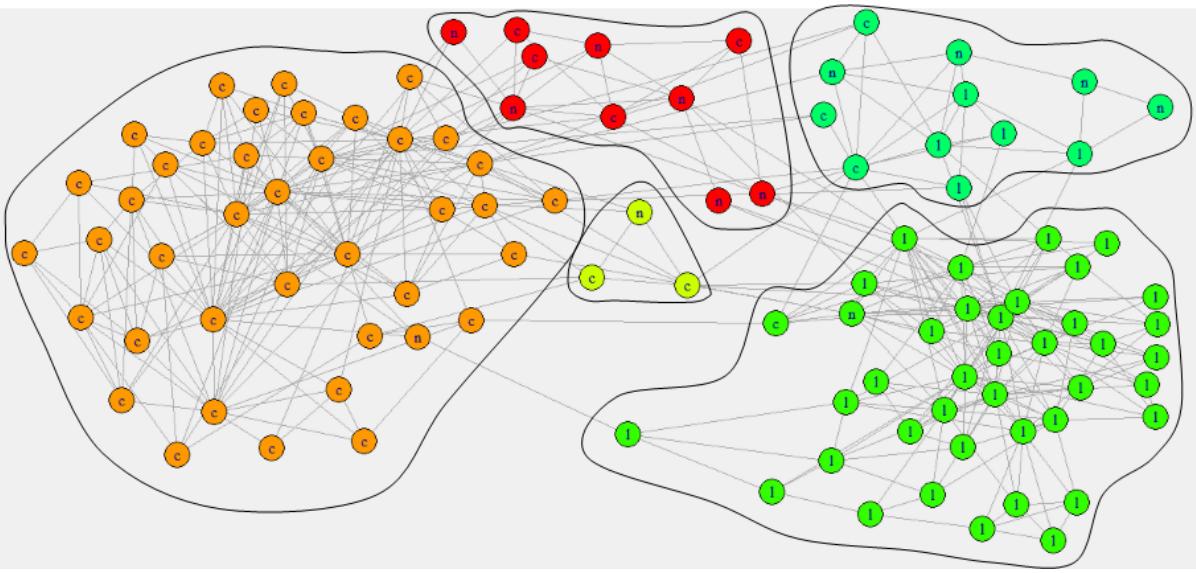

FIGURE 4.17 – Structure de communautés trouvée par la méthode proposée pour le réseau de livres politiques

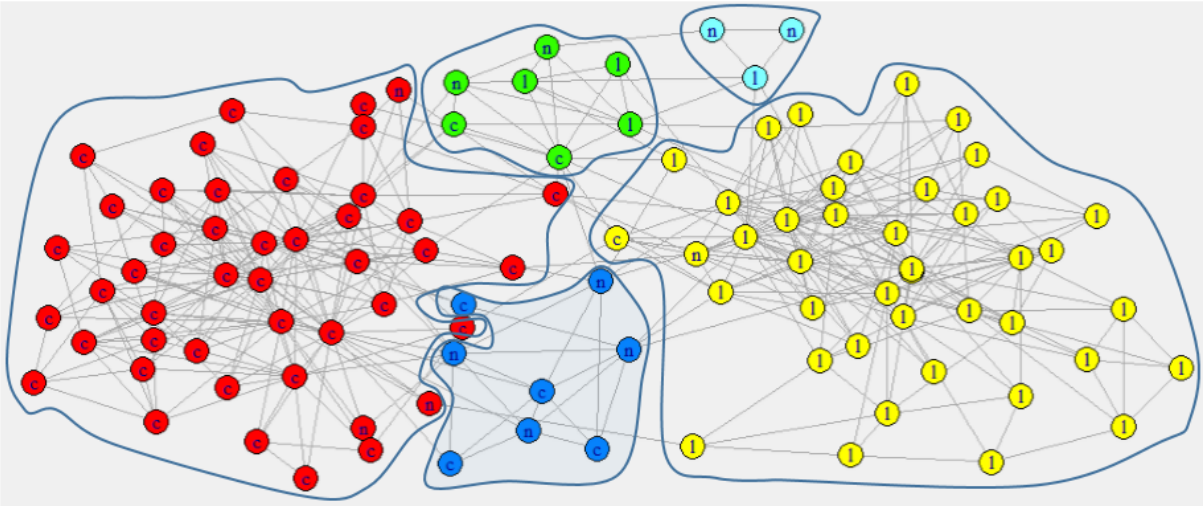

FIGURE 4.18 – Structure de communautés trouvée par la méthode *Edge Betweenness* pour le réseau de livres politiques

4.2.4 Football américain

L'autre exemple de réseau réel étudié dans le cadre de ce mémoire est le réseau de jeux de football américain (*American football games*) [GN02]. Il représente le calendrier des matchs entre des équipes américaines de football durant l'année 2 000. Ce réseau est constitué de douze communautés, 115 nœuds et 613 liens. L'algorithme *Walktrap* a

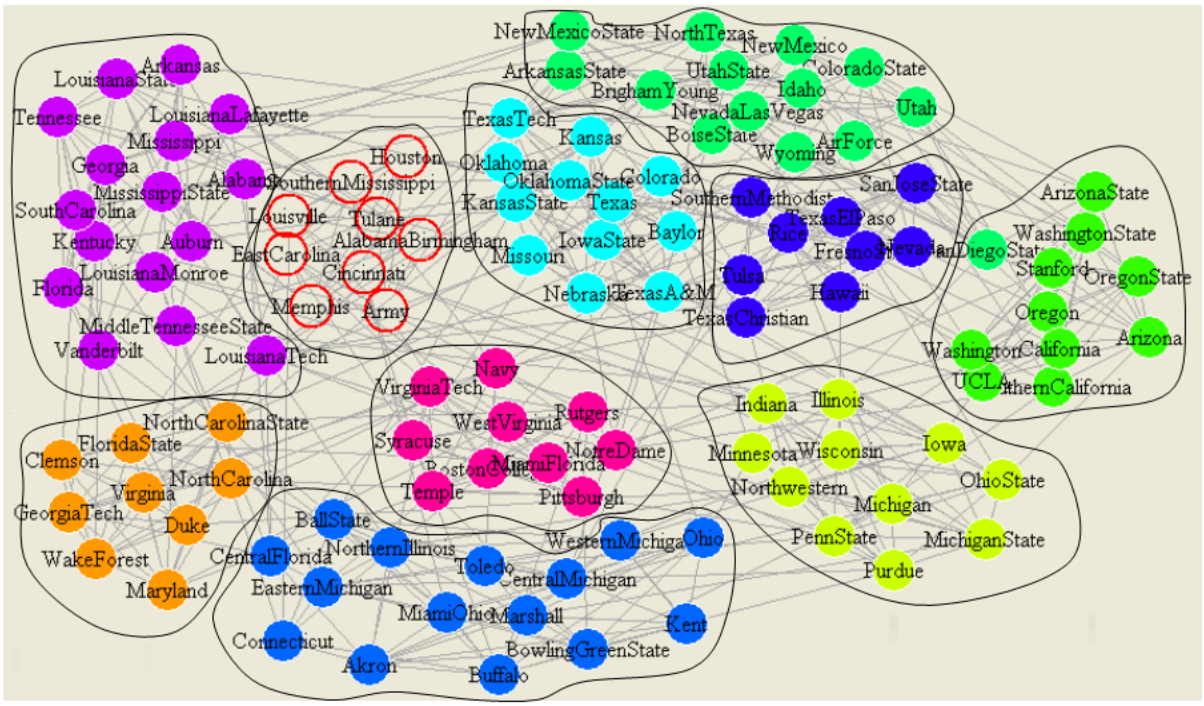

FIGURE 4.19 – Structure de communautés identifiée par *Walktrap* pour le réseau du Football américain

détecté la meilleure structure de communautés représentée par la figure 4.19. Il a détecté dix communautés parmi les douze communautés initiales. Notre méthode a, quant à elle, détecté une structure de communauté proche de *Walktrap* avec seulement neuf communautés. Elle a une mesure F=0.80 pour ce résultat. Mais, en regardant le tableau 4.2 de la modularité, *Walktrap* et la méthode proposée ont la meilleure valeur de Q correspondant à 0.60.

4.2.5 Championnat de football d'Angleterre

Ce réseau [GC13] a 248 nœuds représentant des joueurs et des clubs actifs sur le réseau Twitter à la fin de 2012. Il est constitué de vingt communautés qui sont les clubs du championnat de football d'Angleterre. De point de vue mesure F, la méthode proposée, *Label Propagation* et *Walktrap* ont le meilleur résultat (0.92) comme l'indique le tableau 4.1. Quant à la modularité, la méthode proposée a aussi la valeur maximale de Q (=0.86) avec *Edge Betweenness* et *Walktrap*. Les communautés détectées par la méthode proposée sont représentées dans la figure 4.20.

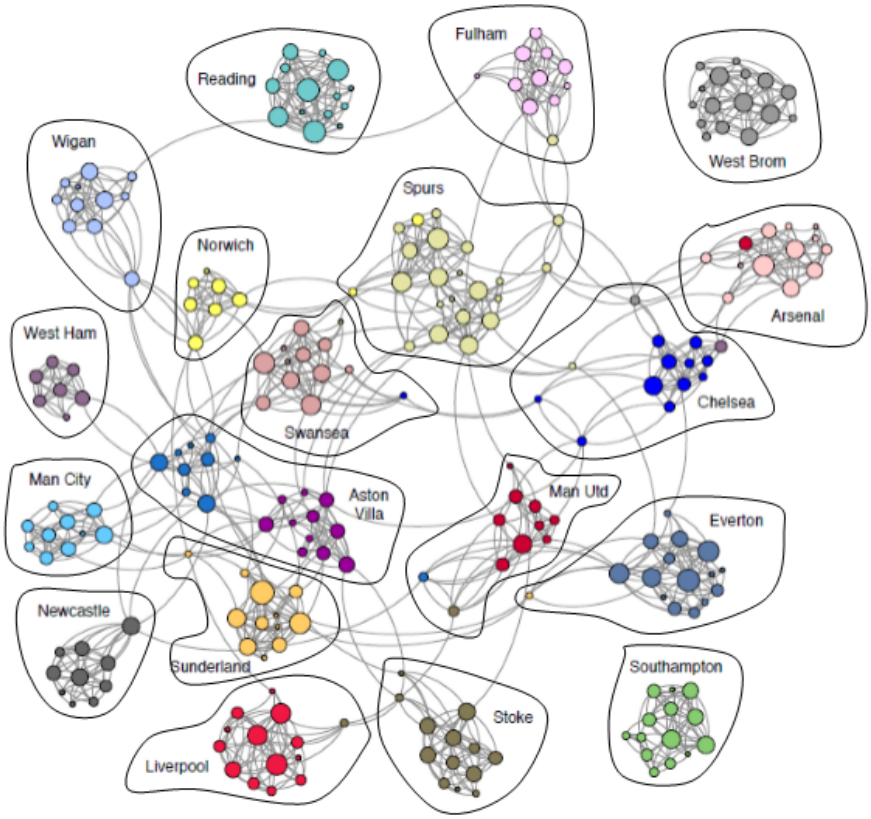

FIGURE 4.20 – Structure de communautés identifiée par la méthode proposée pour le réseau du championnat d'Angleterre

4.2.6 Les députés au Royaume-Uni

Les différentes interactions sur Tweeter entre les comptes des membres du parlement au Royaume-Uni sont représentées dans ce réseau [GC13]. Il s'agit d'un réseau de 419 nœuds et 1 907 liens. Il est constitué de cinq communautés à savoir *Conservative*, *Labour*, $LiberalDemocrat$, SNP et autre. L'algorithme *Label Propagation* a la valeur maximale de mesure F avec 0.95. Le deuxième meilleur résultat est trouvé par la méthode proposée avec une mesure F de 0.94 comme l'illustre le tableau 4.1. En ce qui concerne la modularité, la méthode proposée a la meilleure valeur de Q égale à 0.63 avec *Edge Betweenness* et *Walktrap*. Notre méthode a identifiée cinq communautés qui sont légèrement distinctes des communautés originales (cf. figure 4.21).

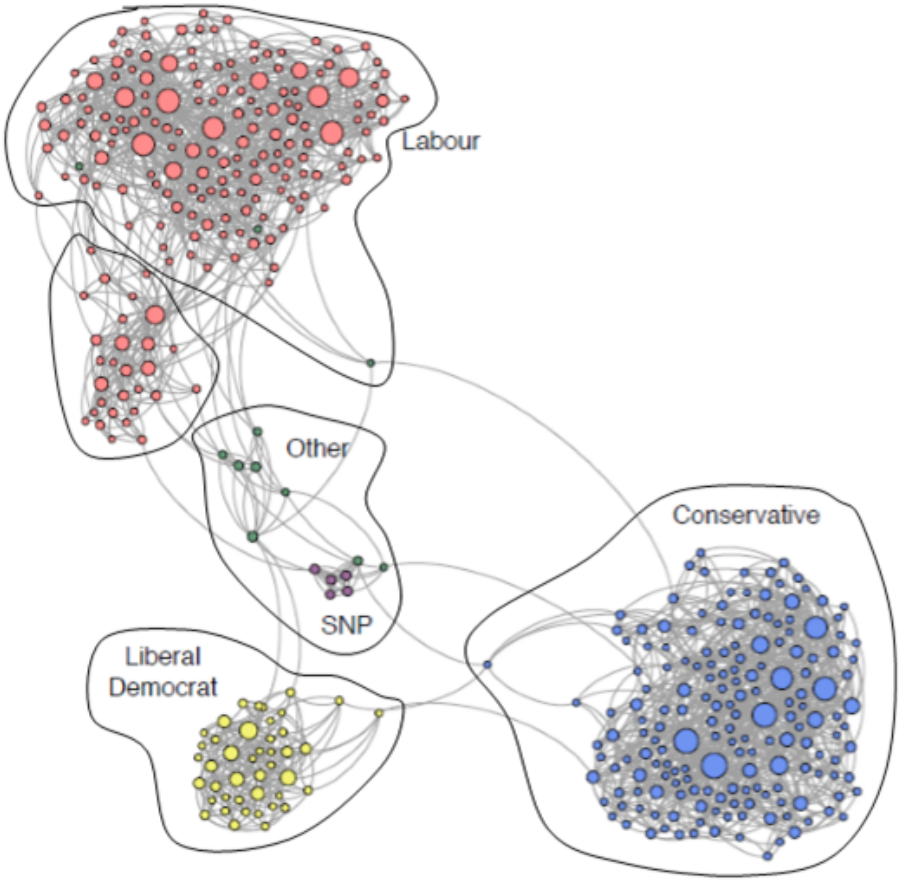

FIGURE 4.21 – Structure de communautés identifiée par la méthode proposée pour le réseau des députés au Royaume-Uni

4.2.7 Les joueurs et clubs de rugby

Le dernier exemple à tester est un réseau plus volumineux [GC13]. C'est une collection de 854 joueurs internationaux de l'Union de rugby, clubs et organisations actives sur Twitter. Les membres du réseau font partie de quinze communautés représentant les différents pays. La structure de communautés identifiée par *Fast Greedy* a une mesure F maximale de 0.52. La méthode proposée a une mesure F de 0.44 mais elle a une modularité maximale Q de 0.88. La faible valeur de la mesure F est due au nombre élevé de communautés détectées. En effet la méthode proposée a identifié vingt-sept (contre quinze) communautés comme l'indique la figure 4.22. Quant à *Fast Greedy* a identifié dix-huit communautés.

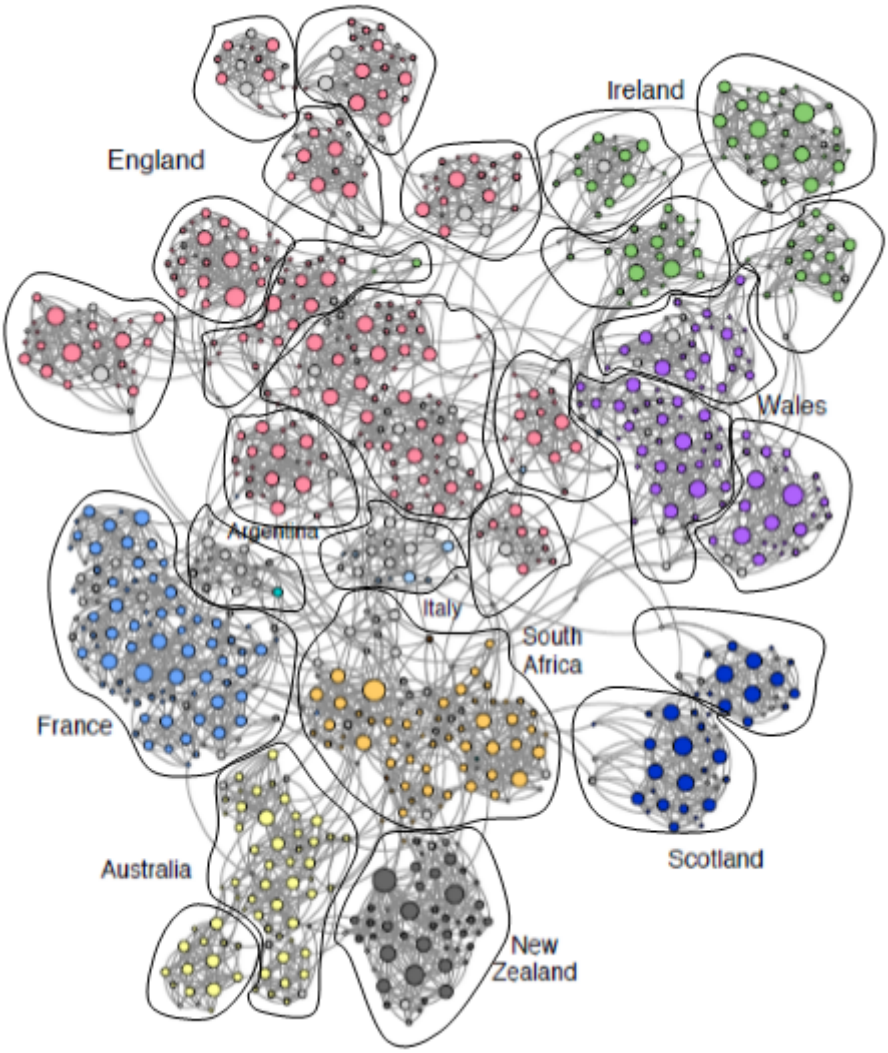

FIGURE 4.22 – Structure de communautés identifiée par la méthode proposée pour le réseau des joueurs et clubs de rugby

La méthode proposée a gardé les mêmes caractéristiques déduites avec les réseaux synthétiques. En effet, la méthode a de bonnes mesures F et des valeurs de modularité Q maximales comme l'indiquent les tableaux 4.1 et 4.2. La méthode est stable aussi bien pour les réseaux synthétiques que réels et tant au niveau de la modularité Q que de la mesure F. Avec le réseau de livres politiques, elle se compare avec la modularité et la mesure F de l'algorithme *Edge Betweenness*. Avec le réseau de *Football* américain, elle se compare avec la mesure F de *Walktrap* et les modularités obtenues pour *Label propagation* et *Walktrap*. La méthode proposée a une mesure F maximale et une modularité maximale pour le réseau du championnat d'Angleterre.

4.3 Temps d'exécution

En ce qui concerne le temps d'exécution, la méthode proposée a fait preuve de sa performance. En effet, elle est toujours plus rapide que l'algorithme *Edge Betweenness*. Avec des réseaux de petite taille de moins de 500 nœuds, elle est plus lente que les trois autres algorithmes. Mais en augmentant le nombre de nœuds (n) et le nombre de liens (m), elle devient plus performante que *Fast Greedy* et *Walktrap*. Pour les différents types de réseaux, *Label Propagation* reste toujours le plus rapide. L'utilisation de la mesure de centralité d'intermédiarité est le premier facteur qui ralentit l'algorithme *Edge Betweenness*.

Réseaux / Algorithmes	(n=100, m=450)	(n=500, m=12 450)	(n=1 000, m=49 950)	(n=3 000, m=449 850)	(n=5 000, m=1 250 000)
Méthode proposée	0.21	3.30	5.33	6.66	63
Edge Betweenness	0.54	480	1800	–	–
Label Propagation	0.00	2.70	3.52	3.77	10.90
Fast Greedy	0.02	3.93	7.50	15	240
Walktrap	0.00	3.75	5.50	8.30	147

Tableau 4.3 – Temps d'exécution en seconde des cinq algorithmes avec différents réseaux

Chapitre 5

Conclusion

Dans le cadre de ce mémoire, nous avons présenté une méthode de détection de communautés dans les réseaux sociaux en procédant à une élimination, en peu d'itérations, des liens entre les nœuds et en exploitant la fonction de modularité de Newman. Au départ, une étude de mesures propres aux nœuds est faite dans le but de regrouper les nœuds similaires dans des communautés. Ensuite, nous nous sommes intéressé aux liens plutôt qu'aux nœuds. En effet, il est plus facile de classer les liens en deux groupes : le groupe de liens inter-communautés et celui des liens intra-communauté plutôt que de classer les nœuds en un nombre inconnu de groupes. L'identification des différentes communautés se fait donc par l'élimination des liens inter-communautés.

La covariance est adoptée comme mesure propre aux liens des réseaux. Cette mesure a l'avantage de réduire le nombre d'itérations durant l'élimination des liens et l'amélioration de la qualité des communautés détectées. Afin de valider notre travail, nous avons mené des comparaisons avec quatre algorithmes de détection de communautés, soit *Walktrap*, *Edge Betweenness*, *Fast Greedy* et *Label Propagation*. Le choix de ces algorithmes est dicté par le fait qu'ils représentent d'une part les différents types d'algorithmes de détection de communautés (agglomératif, divisif, à base du modèle et heuristique), et d'autre part, ils sont également très connus et considérés comme les plus performants et de bons étalons pour une comparaison de performance tant en terme de précision du processus de délimitation de communautés que des temps d'exécution.

Tous les algorithmes ont été exécutés sur des réseaux synthétiques de différentes tailles (100 à 5 000 nœuds), effectifs de communautés (4 à 12 communautés), densités (500 à 1 250 000 liens) et degrés de complexité (degrés de chevauchement entre communautés). Pour chaque valeur des quatre paramètres, 25 réseaux ont été générés et une moyenne des valeurs de la mesure F a été calculée pour estimer la qualité des résultats de chaque

algorithme. Des tests ont été également effectués sur des réseaux réels de diverses tailles et ont confirmé la stabilité et la performance de l'approche proposée.

En conclusion, la qualité de partitionnement de notre méthode était souvent meilleure que celle des autres algorithmes aussi bien pour des réseaux synthétiques que pour des réseaux réels. Elle est relativement stable puisqu'elle se compare avec les meilleurs algorithmes dans plusieurs cas de figures. Les mesures de modularité avec les réseaux réels confirment aussi ces remarques. En ce qui concerne le temps d'exécution, notre méthode est la plus rapide après *Label propagation*. C'est donc une méthode compétitive avec les méthodes connues dans la littérature. Elle est particulièrement attrayante lorsque les paramètres de mixage sont élevés et donc en présence de situations extrêmes où la délimitation des communautés n'est pas si évidente. Elle favorise également les réseaux de grande taille et/ou de densité élevée.

Dans le but d'améliorer ce travail, trois suggestions peuvent être émises. La première consiste à développer de nouvelles mesures propres aux liens qui permettent de distinguer les liens inter-communautés des liens intra-communauté sans itérations répétitives. Une deuxième idée est de remplacer l'inertie inter-classes par une autre fonction permettant de mieux identifier les liens inter-communautés. Finalement, il serait souhaitable de développer un module qui choisit l'algorithme de détection de communautés à exécuter en fonction de la configuration du réseau puisque les tests expérimentaux montrent qu'il n'y a pas un meilleur algorithme dans tous les cas de figures mais que chacun des algorithmes peut être performant dans des cas très spécifiques. Des tests empiriques supplémentaires s'imposent donc pour mieux cerner les forces et faiblesses des divers algorithmes étudiés.

Bibliographie

[AK08] Gaurav Agarwal and David Kempe. Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B-Condensed Matter and Complex Systems*, 66(3) :409–418, 2008.

[BCM+04] Christine Brun, François Chevenet, David Martin, Jérôme Wojcik, Alain Guénoche, and Bernard Jacq. Functional classification of proteins for the prediction of cellular function from a protein-protein interaction network. *Genome biology*, 5(1) :R6–R6, 2004.

[Bra01] Ulrik Brandes. A faster algorithm for betweenness centrality*. *Journal of Mathematical Sociology*, 25(2) :163–177, 2001.

[Bra10] L Branting. Information theoretic criteria for community detection. *Advances in Social Network Mining and Analysis*, pages 114–130, 2010.

[CRA13] Igraph library. `http://cran.r-project.org/web/packages/igraph/igraph.pdf`, 2013. consulté le : 10/07/2013.

[Cse09] Peter Csermely. *Weak links : the universal key to the stability of networks and complex systems*. Springer, 2009.

[DDGDA05] Leon Danon, Albert Díaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics : Theory and Experiment*, (09), 2005.

[Fer12] Emilio Ferrara. Community structure discovery in facebook. *International Journal of Social Network Mining*, 1(1) :67–90, 2012.

[For10] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3) :75–174, 2010.

[Fre79] Linton C Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1(3) :215–239, 1979.

[GA08] David Kempe Gaurav Agarwal. Modularity-maximizing network communities via mathematical programming. *The European Physical Journal B*, pages 409–418, 2008.

[GC13] Derek Greene and Pádraig Cunningham. Producing a unified graph representation from multiple social network views. *arXiv preprint arXiv :1301.5809*, 2013.

[GN02] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12) :7821–7826, 2002.

[GSPA04] Roger Guimera, Marta Sales-Pardo, and Luís A Nunes Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2) :025101, 2004.

[JMSY92] Joxan Jaffar, Spiro Michaylov, Peter J Stuckey, and Roland HC Yap. The clp (r) language and system. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 14(3) :339–395, 1992.

[KRKSSR02] P Krishna Reddy, Masaru Kitsuregawa, P Sreekanth, and S Srinivasa Rao. A graph based approach to extract a neighborhood customer community for collaborative filtering. *Databases in Networked Information Systems*, pages 188–200, 2002.

[LF09a] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1) :016118, 2009.

[LF09b] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms : a comparative analysis. *Physical Review E*, 80(5) :056117, 2009.

[LLM10] Jure Leskovec, Kevin J Lang, and Michael Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.

[LNK07] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7) :1019–1031, 2007.

[LSB+03] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4) :396–405, 2003.

[New01] Mark EJ Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical Review E*, 64(1) :016–132, 2001.

[New04] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), 2004.

[New06] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23) :8577–8582, 2006.

[NG04] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2) :026113, 2004.

[OL09] Günce Orman and Vincent Labatut. A comparison of community detection algorithms on artificial networks. In *Discovery Science*, pages 242–256. Springer, 2009.

[PER13] Newman real networks. `http://www-personal.umich.edu/~mejn/netdata/`, 2013. consulté le : 10/07/2013.

[PKVS12] Symeon Papadopoulos, Yiannis Kompatsiaris, Athena Vakali, and Ploutarchos Spyridonos. Community detection in social media. *Data Mining and Knowledge Discovery*, 24(3) :515–554, 2012.

[PL05] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *Computer and Information Sciences-ISCIS 2005*, pages 284–293, 2005.

[RAK07] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3) :036106, 2007.

[RH07] Andrew Rosenberg and Julia Hirschberg. V-measure : A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, volume 410, page 420, 2007.

[RPR13] R project. `http://www.r-project.org/`, 2013. consulté le : 10/07/2013.

[SAN13] Graphs benchmark. `https://sites.google.com/site/santofortunato/inthepress2`, 2013. consulté le : 10/07/2013.

[SLM+04] Clement A Stanyon, Guozhen Liu, Bernardo A Mangiola, Nishi Patel, Loic Giot, Bing Kuang, Huamei Zhang, Jinhui Zhong, and Russell L Finley Jr. A drosophila protein-interaction map centered on cell-cycle regulators. *Genome Biol*, 5(12) :R96, 2004.

[WIK13] Covariance. http://en.wikipedia.org/wiki/Covariance, 2013. consulté le : 10/07/2013.

[YS04] Genane Youness and Gilbert Saporta. Une méthodologie pour la comparaison de partitions. *Revue de statistique appliquée*, 52(1) :97–120, 2004.

[Zac77] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.