Détection et suivi de l'évolution de communautés ego-centrées dans les réseaux sociaux dynamiques

Ahmed Ould Mohamed Moctar

UNIVERSITÉ CHEIKH ANTA DIOP DE DAKAR
ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

THÈSE DE DOCTORAT UNIQUE

Présentée par
Ahmed Ould Mohamed Moctar
Pour obtenir le grade de
Docteur de l'Université Cheikh Anta Diop de Dakar

Mention : Informatique
Spécialité : Analyse des Réseaux Sociaux

Détection et suivi de l'évolution de communautés ego-centrées dans les réseaux sociaux dynamiques

Soutenue publiquement le 11 Janvier 2019 devant le jury composé de :

Président :	Hamidou Dathe, Professeur, Université Cheikh Anta Diop
Rapporteurs :	Rokia Missaoui, Professeur, Université du Québec en Outaouais
	Cheikh Talibouya Diop, Professeur, Université Gaston Berger
Examinateurs :	Gervais Mendy, Maître de Conférences, Université Cheikh Anta Diop
	Joseph Ndong, Maître Assistant, Université Cheikh Anta Diop
Directeur de thèse :	Idrissa Sarr, Maître de Conférences, Université Cheikh Anta Diop

Année universitaire 2017-2018

III

يظَلُّ المرءُ حكيماً ما دام يبحثُ عن الحِكْمَة ؛
فإن اعتقد أنَّه حصل عليها فَقَدَ عقْله.

(مثل عربي)

« L'homme est sage tant qu'il cherche la sagesse,
Mais, à croire l'avoir acquise, il perd la tête. »

(Proverbe arabe)

Remerciements

Comme la thèse est le fruit d'un travail collectif, je tiens à remercier ici toutes les personnes ayant intervenu, de près ou de loin, dans l'accomplissement de ce travail.

Tout d'abord, je tiens à remercier vivement mon directeur de thèse Docteur Idrissa Sarr, sans qui rien n'aurait été réalisé. Outre son appui technique et ses conseils pertinents, j'en ai appris la ponctualité et la constance dans le travail, entre autres. Il a su m'orienter, me conseiller, tout en me laissant libre dans mes choix. Grâce à lui, j'ai pu faire mes premiers pas dans le très vaste monde de la recherche. Certes, il me reste beaucoup à apprendre mais, son encadrement me permet aujourd'hui d'être serein vis-à-vis du chemin à venir. La fin de cette thèse n'est évidement pas la fin de notre amitié encore moins de notre collaboration.

Ensuite, je tiens à exprimer toute ma gratitude aux Professeurs Rokia Missaoui et Cheikh Talibouya Diop qui m'ont fait l'honneur d'être rapporteurs de cette thèse, et qui ont bien voulu me consacrer une partie de leur temps malgré leurs nombreuses occupations. Mes remerciements vont également aux Docteurs Gervais Mendy et Joseph Ndong pour avoir accepté de participer au jury de cette thèse. Je remercie également Professeur Hamidou Dathe de bien avoir voulu présider ce jury.

De plus, je tiens à remercier tous les chercheurs qui fréquentaient la salle des doctorants à la Section Informatique surtout ceux avec qui j'ai eu à discuter à propos de la recherche. Spécifiquement, la bonne humeur de mon collègue Joel Tanzouak ainsi que Docteur Ndiouma Bame, leur ponctualité, leur dynamisme et les sujets très poussés que nous avons débattus qu'ils soient scientifiques, personnels ou politiques ont constitué des ingrédients essentiels au succès de ma thèse.

En outre, je remercie le directeur de l'école doctorale Professeur Hamidou Dathe ainsi que le responsable de la formation doctorale Professeur Karim Konaté sans oublier le directeur du laboratoire d'informatique Docteur Cheikh Ahmadou Bamba Gueye ; je les remercie pour leur disponibilité et leurs services administratifs multiples.

Enfin, je ne peux m'empêcher de manifester mes sentiments les plus intimes en faveur de ma famille, notamment mon cher papa Mohamed Moctar et très chère mère Souéidou Hamoud. Je les remercie pour leur affection sans faille, leur noble éducation, leurs conseils et leur assistance morale et matérielle. Que mon cher frère Cheikh Brahim, mes sœurs Selem et Diana trouvent l'expression de la finesse dont ils m'ont encadrée. Mes remerciements vont également à mes oncles, mes tantes, mes cousines, cousins et connaissances.

Résumé

Le développement des réseaux sociaux en ligne a donné naissance à plusieurs opportunités permettant de communiquer, d'accéder et de partager de l'information de n'importe où et à n'importe quel moment. Par exemple, les applications sociales telles que *Facebook*, *Viber*, *WhatsApp*, *Imo*, *Line*, entre autres, permettent de faire connaissance avec des amis et de discuter avec eux via des photos, des vidéos ou encore par messagerie instantanée.

Les données collectées à partir de ces applications intègrent le temps et la position géographique à partir desquels les messages ont été envoyés et/ou reçus. De ce fait, il serait intéressant de construire à partir de ces données un graphe social dont les nœuds représentent les individus et les liens leur relations.

L'expansion des réseaux sociaux en ligne s'est accompagnée d'un développement remarquable de l'Analyse des Réseaux Sociaux (ARS), qui est un processus d'exploration de réseaux visant à identifier des caractéristiques et d'en exploiter les informations. La détection de communautés est l'une des thématiques principales de l'ARS qui attirent de plus en plus l'attention de nombreux chercheurs. Les premiers travaux relatifs à la détection de communautés se limitaient aux communautés statiques. Avec le temps, la dimension temporelle qui désigne l'évolution de communautés a été ajoutée pour pouvoir gérer l'aspect dynamique des réseaux sociaux actuels.

En effet, la majorité des travaux existants cherchent à partitionner un réseau en plusieurs communautés « globales » et suivre leur évolution au fil du temps. L'inconvénient principale de cette approche est qu'elle nécessite l'état global du réseau. Or, il est souvent difficile d'obtenir un réseau dynamique dans sa globalité. Une autre approche consiste à détecter les communautés « locales » qui sont construites à partir de certains nœuds spécifiques, appelés « nœuds d'intérêt ». Une extension de communautés locales est appelé « communautés ego-centrées » et constitue le contexte d'étude de cette thèse. En effet, contrairement aux communautés locales, celles ego-centrées sont construites sur la base d'un voisinage direct ou indirect du nœud d'intérêt. Ainsi, elles permettent, entre autres, de mieux comprendre les relations établies entre le nœud d'intérêt et ses voisins directs et indirects.

Même si la longueur du voisinage constitue un atout de communautés ego-centrées, les travaux existants se limitent encore au voisinage direct lors de la détection de communautés. De plus, les solutions existantes ne prennent pas en compte les communautés ego-centrées dynamiques. L'objectif de cette thèse est de pallier à ces manquements dans l'étude des communautés égo-centrées. Pour ce faire, nous avons proposé une solution de détection de communautés dynamiques qui fonctionne en trois étapes, à savoir :

1. faire plusieurs captures du réseau dynamique à des instants précis. Chaque capture représente l'état du réseau dynamique au cours d'une période donnée ;
2. proposer et appliquer un algorithme de détection de communautés statiques sur chaque capture en vue de découvrir des communautés égo-centrées ;
3. prendre la structure d'une communauté égo-centrée sur deux instantanés successifs et vérifier s'il y a eu des changements majeurs. Pour interpréter la nature des changements possibles, nous

avons défini des règles de correspondance. Ainsi, au fil du temps nous établissons la manière d'évolution des différentes communautés.

Nous avons implémenté notre solution avec une pile d'outils dédiés à l'ARS et nous avons utilisé trois jeux de données. Les résultats d'expérimentation ont montré la faisabilité de notre solution, ses gains et son efficacité.

Mots-clés : analyse des réseaux sociaux, détection de communautés, réseaux sociaux dynamiques, communautés ego-centrées, applications sociales.

Abstract

The development of online social media has created many opportunities to communicate, access, and share information from anywhere and at anytime. This kind of application such as *Instagram*, *WhatsApp*, *Imo*, *Line*, as well as *Facebook* affords plenty of possibilities for getting in touch with friends, colleagues, and relatives at every moment with real-time messages, photos, videos, etc.

Data collected from those applications integrate two parameters such as the time and the geographical position from which they were sent and/or received. Therefore, it is worthwhile to build a social graph based on these data where individuals are the vertices and their interactions the edges.

Alongside the growth of online social media, there is a rapid expansion of social network analysis (SNA), which is a process of exploring the graph in order to discover knowledge leading to informative decision-making. At the heart of the SNA topics that attract many scientists, we have the community detection. The mean reason is related to the fact that individuals tend to form a community in many real-life situations.

Early works in community detection focused mainly on partitioning networks into several global communities before they target communities evolution over time. The main drawback of such approaches is the difficulty to obtain the entire network on which we may process a set of algorithms to track evolution. As a result, many researchers focused on detecting and tracking local dynamic communities. An extension of local communities is called "ego-communities". Unlike local communities, ego-centered ones can be built based on a large neighborhood of the ego. They allow, among others, to better identify and track the network elements activities from some particular nodes. Moreover, the existing works on ego-communities are built based on the direct neighborhood of the ego ignoring the fact that indirect neighbors may have special impact on the structure. In addition, existing works did not take into account dynamic ego-communities. To overcome these weaknesses, we have proposed a dynamic ego-community solution that works in three steps, namely :

1. create several snapshots of the dynamic network. Each snapshot represents the state of the dynamic network during a given period ;
2. propose and apply a static ego-community detection algorithm on each snapshot to discover ego-communities ;
3. get the structure of the detected ego-communities on two successive snapshots and check whether there are major changes. Depending on the nature of the change, the algorithm performs an interpretation corresponding to a characterization of the community evolution.

In order to validate our solution, we conducted a validation on three datasets. The results of experimentation showed the effectiveness of our solution and its benefits.

Keywords : social networks analysis, community detection, dynamic social networks, ego-community, social media.

Table des matières

Liste des figures

Liste des tableaux

Liste des acronymes

ARS Analyse des Réseaux Sociaux. 8, 13, 20

EVD Extraction du Voisinage Direct. 84

NDTV Network Dynamic Temporal Visualizations. 80

SAAB Sélection Aléatoire avec Ajout par Bloc. 81, 83

SAAN Sélection Aléatoire avec Ajout par Nœud. 81, 83

SDAB Sélection Déterministe avec Ajout par Bloc. 81, 83–85, 87, 88

SDAN Sélection Déterministe avec Ajout par Nœud. 81, 83

SDRB Sélection Déterministe avec Retrait par Bloc. 87, 88

Liste des notations

(u, v) Un lien entre deux nœuds u et v

β Taille de bloc de nœuds

δ Densité du graphe G

$\delta_(u, v)$ Vaut 1 si $u \in \mathcal{B}$ et $v \in \mathcal{C}$ ou vice versa, sinon, il est égal à 0

η Seuil modélisant le taux de changements que le réseau subit entre deux instants

γ Coefficient de prédiction

$\mathcal{B}$ Bordure de la communauté C

$\mathcal{C}$ Cœur de la communauté C

$\mathcal{N}$ Réseau dynamique, orienté et pondéré

$\mathcal{N}_t$ Capture d'un réseau dynamique à l'instant t

$\mathcal{U}$ Extérieur de la communauté C

μ Coefficient de l'erreur prédite

$\overline{d}(G)$ Degré moyen du graphe G

$\phi(C)$ Conductance d'une communauté

$\sum_{\forall u \in C} d(u)$ Somme des degrés des nœuds qui se trouvent dans C

$\sum_{w_C^{in}}$ Somme des poids des liens ayant leurs deux extrémités dans C

$\sum_{w_C^{out}}$ Somme des poids des liens dont une seule extrémité se trouve dans C

$\sum_{w_{\mathcal{N}_t}}$ Somme des poids de tous les liens d'un réseau dynamique à l'instant t

τ Seuil utilisé pour quantifier la qualité de communautés ego-centrées

C et $\overline{C}$ Respectivement, une communauté et son complémentaire dans le graphe G

$C_{\{u,k\}}$ Communauté ego-centrée en k-étapes du nœud u

$cut(C)$ Nombre de liens connectant une communauté C au reste du réseau $\overline{C}$

$d(u)$ Degré de centralité du nœud u, soit le nombre de voisins de u

$d_{Iso}(C)$ Degré d'isolation de la communauté C

Dm Diamètre du graphe G

E Ensemble de liens du graphe G

E_C Ensemble de liens de la communauté C

$E_{\mathcal{N}_t}$ Ensemble de liens d'un réseau dynamique à l'instant t

G Un graphe de n nœuds et m liens

I Nombre de liens n'ayant aucune extrémité dans $\mathcal{U}$

$N_{\{u,k\}}$ Ensemble de nœuds connectés au nœud u en k-étapes

$S_{\{u,k\}}$ Graine d'initialisation pour construire la communauté ego-centrée en k-étapes du nœud u

T Nombre de liens ayant au moins une extrémité dans $\mathcal{B}$

T_{max} Paramètre utilisé dans (Tsourakakis et al., 2013) indiquant le nombre maximal voulu d'itérations

V Ensemble de nœuds du graphe G

V_C Ensemble de nœuds de la communauté C

$V_{\mathcal{N}_t}$ Ensemble de nœuds d'un réseau dynamique à l'instant t

$wd(u)$ Degré pondéré du nœud u, soit la somme des poids des liens adjacents à u

$wd_v^{C_{\{u,k\}}}$ Somme des poids des liens dont l'une extrémité est v et l'autre appartient à $C_{\{u,k\}}$

Chapitre 1

Introduction

1.1 Motivations

Avec la prolifération des applications sociales et mobiles, les utilisateurs sont en interaction de manière continue, à travers des partages de documents, de photos/vidéos, de messages, etc. Ces interactions peuvent être modélisées via un réseau social représenté sous forme de graphe dont les nœuds désignent les utilisateurs et les liens représentent leurs interactions. Pour étudier les caractéristiques des éléments d'un réseau social, les chercheurs font souvent recours à l'analyse des réseaux sociaux.

L'une des thématiques les plus étudiées dans l'analyse des réseaux sociaux est la détection de communautés qui peut être vue comme une approche, à la fois graphique et analytique, permettant d'identifier les utilisateurs ayant des intérêts communs ou partageant des propriétés similaires. Une communauté peut être définie comme un ensemble de nœuds, densément connectés entre eux et légèrement liés au reste du réseau.

Aujourd'hui, la thématique de détection de communautés ne cesse d'attirer l'attention de la communauté scientifique. De ce fait, il y a eu des centaines de travaux autour de cette thématique durant les dernières années. Cet intérêt, de plus en plus croissant, s'explique par les usages divers et multiples du concept de communauté dans de nombreux domaines. Par exemple, elle est utilisée pour l'amélioration des stratégies de marketing car un individu s'intéresse, d'habitude, à un produit/service si la quasi-totalité de ses amis s'y intéresse. De même, la structure d'une communauté révèle une gamme d'informations pouvant être utilisées pour comprendre et prédire les caractéristiques de la diffusion des épidémies. En outre, la structure de communautés permet d'identifier les cercles sociaux et suggérer, éventuellement, des amis aux utilisateurs.

1.2 Problématiques

Dans cette thèse, nous nous intéressons au suivi de l'évolution de communautés dans les réseaux dynamiques issus des applications sociales. En effet, il existe deux manières de faire le suivi de l'évolution de communautés dans les réseaux dynamiques. La première conçoit les changements du réseau comme étant un flux de données qui évolue au fil du temps. Ainsi, elle consiste à mettre à jour la structure de communautés lors de chaque ajout et/ou suppression de nœuds et/ou liens. Cette approche peut entraîner un coût assez exorbitant surtout pour les réseaux qui changent très fréquemment. En outre, elle ne garantit pas la stabilité des communautés à long terme. Autrement dit, il est difficile d'assurer une cohérence de communautés sur l'ensemble des étapes d'évolution, car le suivi de communautés se fait uniquement lors du passage d'un instant à l'autre. La deuxième approche réalise le suivi de communautés par intervalle de temps. Elle considère le réseau dynamique comme un ensemble d'instantanés (réseaux statiques) dont chacun représente l'état du réseau au cours d'une période donnée. Nous avons

privilégié cette approche dans le cadre de cette thèse puisqu'elle permet de tirer profit des avancées et des solutions de détection de communautés existantes. Ainsi, nous pourrons ré-utiliser et/ou adapter des solutions existantes pour résoudre des problèmes spécifiques à notre contexte. Toutefois, le choix de cette approche soulève trois problématiques, à savoir :

1. Soit un réseau dynamique construit au fil du temps, comment décomposer le réseau en plusieurs instantanés de sorte que chacun représente un état sur lequel nous pouvons faire des analyses ? Ce problème est assez crucial car une mauvaise décomposition peut entraîner soit une multitude d'instantanés difficiles à exploiter, soit des instantanés impertinents pour le suivi de l'évolution. Un problème sous-jacent à la décomposition est de décider s'il faut la faire de manière périodique ou de manière dynamique en s'appuyant sur l'évolution réelle du réseau dynamique. De plus, il est important de réfléchir sur comment faire des instantanés pertinents à partir des caractères intrinsèques du réseau et sans effets de bord sur son évolution.
2. comment faire la détection de communautés statiques sur chaque instantané en supposant que celles-ci doivent être construites à partir de nœuds spécifiques, appelés égos ? Les questions soulevées par ce problème sont : comment définir une communauté dans notre contexte ? Comment évaluer la pertinence d'une communauté ? Quelle approche de détection de communautés faut-il privilégier ?
3. comment utiliser notre solution de détection de communautés sur les différents instantanés en vue de révéler les transformations des structures des communautés au fil du temps ? En s'intéressant à ce problème, on s'interroge d'abord sur les possibles évolutions des communautés dans notre contexte. Puis, nous nous focalisons sur comment caractériser chaque type d'évolution. Enfin, vient la question de comment identifier les évolutions entre les instantanés.

1.3 Contributions de la thèse

Nos principales contributions pour faire face aux problèmes soulignés peuvent être résumées autour de plusieurs points.

1.3.1 Architecture globale de la solution proposée

Nous avons proposé une architecture d'extraction et d'étude de communautés pour les réseaux sociaux. La solution est conçue autour de trois modules structurés sous forme de couches. Le premier module comprend les logiciels et bibliothèques nécessaires pour l'analyse des réseaux sociaux et l'implémentation des deux autres couches. Le deuxième module permet de : *i)* représenter les données sous forme d'un réseau social ; et *ii)* décomposer le réseau en plusieurs instantanés. Le dernier module consiste à gérer la détection des communautés. Pour ce faire, ce module inclut deux sous-modules : *i)* un sous-module de détection de communautés égo-centrées sur les instantanés ; et *(ii)* un sous-module qui assure le suivi de l'évolution des communautés au fil du temps. La figure 1.1 présente une illustration de l'architecture globale de notre solution.

Notre architecture est conçue de sorte que chaque couche soit autonome et puisse fonctionner avec des interfaces permettant les échanges de données et/ou de traitements avec les autres couches. Ainsi, l'architecture est plus facile à faire évoluer pour faire face aux futurs besoins.

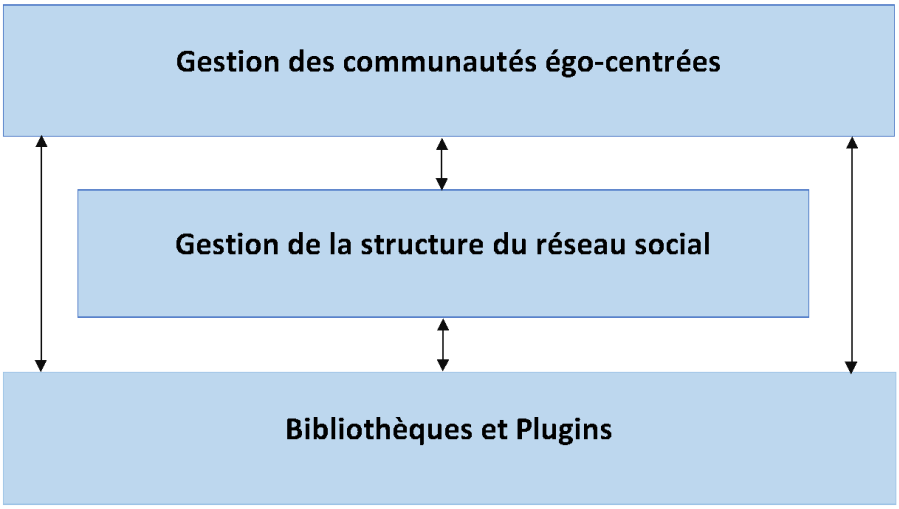


FIGURE 1.1 – Vue simplifiée de l'architecture globale de notre solution.

1.3.2 Représentation du graphe social dynamique

Nous nous plaçons dans le contexte des applications sociales permettant les interactions instantanées entre individus. Les interactions désignent la communication entre individus et peuvent se faire par échange de textes, d'images, de vidéos. De même, les utilisateurs peuvent commenter aussi les objets d'un utilisateur (son statut, ses photos, ses vidéos, etc.). Cette manière de commenter les objets des autres est aussi une sorte de communication directe ou indirecte entre utilisateurs. Nous représentons les données issues de ces applications par un réseau social orienté et pondéré. Concrètement, nous considérons qu'un lien est établi entre deux utilisateurs lorsque qu'il y a une communication entre eux. La direction du lien s'oriente de l'utilisateur ayant initié la communication vers l'utilisateur la recevant. Si un utilisateur soumet plusieurs communications à un autre, le poids du lien s'incrémente en indiquant le nombre de communications ayant eu lieu.

Pour gérer l'aspect évolutif du réseau, nous considérons que l'instant t indique l'état initial du réseau. Ensuite, si à l'instant $t+1$, l'un des utilisateurs du réseau communique avec un nouvel élément [1], ce dernier sera ajouté dans le réseau sous forme d'un nouveau nœud, muni du lien correspondant. Si, à l'instant $t+1$, une nouvelle interaction apparaît entre deux utilisateurs déjà existants dans le réseau, cela implique l'apparition d'un nouveau lien. C'est ainsi, donc, que le réseau social évolue dans le temps. Nous considérons que chaque lien est muni d'un attribut « durée » indiquant le temps que dure chaque communication. Notre manière de représenter le réseau vise à traduire au mieux la réalité du terrain et nous permet de proposer des solutions adaptées.

1.3.3 Décomposition du réseau dynamique en plusieurs instantanés

Il s'agit de décomposer le réseau dynamique en plusieurs instantanés dont chacun représente l'état du réseau au cours d'une période donnée. Le processus de décomposition est fait suivant le niveau des changements que le réseau subit au fil du temps. Autrement dit, entre deux instantanés successifs, nous devons avoir suffisamment de changements de la structure communautaire. L'intuition derrière cela est de ne pas perdre du temps en cherchant les changements de communautés sur deux instantanés

1. Un utilisateur qui n'existe pas encore dans le réseau.

topologiquement proches. Pour ce faire, nous avons proposé une fonction de qualité globale ψ permettant de mesurer le niveau des changements intervenus dans le réseau à un instant donné. Dans cette perspective, nous considérons que le réseau a connu des changements considérables si la différence de qualité entre deux instants t et $t+1$ devient supérieure à un seul donné ε. Formellement :

$$|\psi(\mathcal{N}_t) - \psi(\mathcal{N}_{t+1})| \geqslant \varepsilon \tag{1.1}$$

Nous remarquons que la formule 1.1 nécessite deux paramètres, à savoir, le réseau à l'instant actuel t et le réseau à l'instant $t+1$. Ainsi, pour prédire le prochain instant $t+1$, nous utilisons une série temporelle construite à partir des scores de qualité au fil du temps. Si l'instant prédit vérifie l'équation 1.1, nous le sauvegardons dans l'historique et nous prédisons le prochain. Sinon, nous supprimons l'instant prédit de l'historique et nous passons à un instant ultérieur. Ce protocole de décomposition présente l'avantage de ne pas être intrusif et propose des instantanés efficients contrairement aux instantanés qui sont pris périodiquement sans une évaluation de leur pertinence.

1.3.4 Algorithme de détection de communautés ego-centrées statiques

Après la décomposition du réseau dynamique en plusieurs instantanés, nous proposons une solution de détection de communautés dans un réseau orienté et pondéré. Notre algorithme consiste à détecter les communautés ego-centrées à partir de quelques nœuds d'intérêt appelés égos. Pour ce faire, il cherche à minimiser une fonction de qualité locale que nous avons proposée. La fonction de qualité a pour but d'évaluer la pertinence d'une communauté égo-centrée tout en prenant en compte la direction et le poids des liens. La fonction qualité a été conçue avec deux variantes. La première consiste en une métrique de portée locale dont le but est de mesurer la pertinence d'une communauté statique ego-centrée. La seconde manière comprend une métrique de portée globale permettant d'évaluer la qualité d'un réseau dynamique à un instant donné. Quant à la manière de construire les communautés, nous observons deux cas. Le premier cas détecte les communautés ego-centrées tout en se limitant au voisinage direct de l'égo. Par contre, le second cas suppose que le voisinage d'un nœud peut être élargi aux nœuds non directement liés à lui. L'idée d'élargir le voisinage au-delà des nœuds adjacents est de mesurer l'impact de l'égo dans un sous-graphe et vice-versa.

Notre approche de détection a l'avantage de prendre en compte les réseaux orientés et pondérés contrairement à la plupart des solutions existantes. De plus, nous avons défini une communauté égo-centrée de sorte que la présence du nœud égo soit indispensable dans ladite communauté. Cette manière de faire diffère des solutions existantes qui supposent que le nœud égo ne sert qu'à initialiser l'algorithme de détection et que sa présence n'a pas d'importance capitale sur la communauté trouvée. Enfin, notre algorithme qui structure une communauté en allant au-delà du voisinage direct permet de comprendre et d'analyser les relations qui existent entre le nœud d'intérêt et ses voisins directs ou indirects. Ces relations peuvent avoir un impact significatif sur non seulement le comportement du nœud d'intérêt, mais aussi sur sa communauté.

1.3.5 Algorithme de suivi de l'évolution de communautés ego-centrées dynamiques

Enfin, nous avons proposé une approche de suivi de l'évolution des communautés égo-centrées au fil du temps. A cette fin, nous avons défini l'ensemble des types d'évolutions possibles. Puis, nous

avons proposé une procédure de comparaison des états des communautés au fil des instantanés pour détecter les variations de leur structures. Notre approche consiste à évaluer tous les changements entre deux instantanés successifs sur tous les aspects du réseau, à savoir les nœuds, les liens, la structure des communautés, etc. Pour établir la nature des évolutions, nous avons introduit des règles de correspondance nous permettant d'interpréter les changements.

Notons que notre approche corrige des faiblesses identifiées dans les solutions existantes. En effet, nous incluons dans le processus d'évolution la variation de l'intensité des liens au fil du temps. Ceci a pour avantage de ne pas se limiter qu'aux changements topologiques comme le font les solutions actuelles de suivi des communautés. De même, les changements microscopiques relatifs aux nœuds et de leurs liens sont pris en compte en plus des changements macroscopiques considérées par les autres approches. Cette manière de faire ainsi que la définition d'une égo-communauté nous a permis d'exclure certains types d'évolutions proposés dans la littérature. En effet, compte tenu que le nœud égo est indispensable pour former une communauté, nous avons remarqué que la division d'une communauté en deux ne pouvait avoir lieu. De même, partant du fait que chaque communauté a un seul nœud égo, nous ne saurons fusionner deux égo-communautés au risque d'enlever le statut d'égo à un des nœuds ou de considérer des communautés multi-égo. Ce dernier cas, ne faisant pas partie des études de cette thèse, est un problème ouvert sur lequel il faudra réfléchir. Au demeurant, il convient de souligner que notre étude a permis de soulever l'impossibilité d'avoir certaines évolutions proposées dans la littérature et qu'il sied de revoir certaines études sur l'évolution des communautés pour mieux spécifier leur cadre d'application.

1.3.6 Validation et Résultats

Pour valider notre approche, nous avons implémenté l'architecture présentée au début en utilisant des outils d'analyse de réseau sociaux et de statistiques comme *R*, *NetworkDynamic* et *Igraph*. L'implémentation a nécessité l'utilisation de plugins d'extraction d'informations des réseaux sociaux comme *RFacebook* pour récupérer des données de Facebook. En fonction des différents besoins d'expérimentation, trois jeux de données ont été utilisés. Puis, nous avons effectué une série d'expérimentation pour évaluer nos algorithmes de détection de communautés statiques ego-centrées, notre méthode de gestion des instantanés et notre algorithme de suivi de l'évolution de communautés dynamiques. Les expériences ont eu pour but d'évaluer l'efficacité de notre fonction de qualité, la pertinence des communautés égo-centrées trouvées, l'exactitude de notre gestion des instantanés et la faisabilité de notre approche de suivi de communautés dynamiques.

Les résultats ont montré que la définition de notre fonction de qualité permettait d'évaluer la pertinence des communautés dans le contexte d'un graphe social orienté et pondéré. De plus, avec cette fonction, les expériences ont montré que notre algorithme d'extraction de communautés présente de meilleurs résultats qu'un algorithme basé sur un voisinage direct. De même, les expériences ont montré l'impact d'élargir l'extraction aux voisins indirects. Ceci est structurellement pertinent car il permet de voir l'impact qu'un nœud peut avoir au-delà de ses voisins en s'appuyant uniquement sur les communications. Dans le contexte d'une lutte contre la propagation d'une épidémie, cela aurait permis de voir les personnes non en contact direct avec un suspect et qui sont tout de même exposés par effet de diffusion. Nous avons également vu que la gestion dynamique des instantanés donne la garantie que les captures soient porteuses d'informations utiles en termes de changements. Enfin, nous

avons montré que l'évolution de communautés pouvait se faire avec nos règles de correspondance et que certaines évolutions ne pouvaient pas avoir lieu dans notre contexte.

Les travaux de cette thèse ont fait l'objet de sept publications, à savoir, un article de journal international (Moctar et Sarr, 2016), un chapitre de livre (Moctar et Sarr, 2017b) et cinq articles de conférences internationales (Moctar et Sarr, 2017a ; Moctar et Sarr, 2018a ; Moctar et Sarr, 2018b ; Moctar et Sarr, 2018c ; Moctar, Sarr et Vaumi Tanzouak, 2019). En outre, nous avons également un article de journal déjà soumis.

1.4 Organisation du manuscrit

Le manuscrit de notre thèse est structuré en trois parties en dehors de l'introduction (Chapitre 1) et de la conclusion (Chapitre 7).

1. **Première partie** : La première partie est relative à l'étude de l'existant et est découpée en deux chapitres comme suit :
   - le chapitre 2 présente les notions fondamentales relatives à l'analyse des réseau sociaux ;
   - le chapitre 3 présente un état de l'art sur la détection de communautés statiques et dynamiques ;
2. **Deuxième partie** : La deuxième partie présente les contributions de cette thèse résumées autour de deux chapitres à savoir :
   - le chapitre 4 qui décrit le modèle de représentation des données ainsi que nos algorithmes de détection de communautés statiques ;
   - le chapitre 5 qui explique notre méthode de gestion des instantanés et notre algorithme de suivi de l'évolution de communautés dynamiques ;
3. **Troisième partie** : La dernière partie est relative à la validation avec un seul chapitre (Chapitre 6) qui donne les détails de la validation de notre solution.

Première partie

Étude de l'Existant

Chapitre 2

Généralités sur l'Analyse des Réseaux Sociaux

Une caractéristique particulière des deux dernières décennies est l'utilisation quotidienne du Web 2.0 qui est devenu une véritable pépinière de médias sociaux permettant à des utilisateurs d'interagir, de partager, de se regrouper, de collaborer, etc. Du contenu de ces médias sociaux, peuvent être extraites des structures interconnectées et bien organisées. Ces structures appelées souvent « réseaux sociaux numériques » ou « réseaux sociaux en ligne » pour se démarquer des réseaux sociaux tels qu'ils sont abordés traditionnellement en sciences sociales. Ces réseaux sociaux sont formés sur la base soit, des individus et de leurs interactions soit, d'un ensemble d'objets des médias sociaux ayant des liens de nature quelconque. Ainsi, on peut avoir des réseaux d'individus, des réseaux de pages web, des réseaux d'entreprises, des réseaux de produits/services, etc. Ces différents exemples démontrent la complexité de la notion de réseaux sociaux, de leur omniprésence dans notre vie mais aussi de l'intérêt que ceux-ci présentent comme sujet d'étude. Le problème actuel n'est plus comment collecter et stocker les données des réseaux sociaux mais plutôt comment les utiliser de manière pertinente en vue d'une analyse porteuse de valeur ajoutée.

S'il est évident de noter que l'Analyse des Réseaux Sociaux (ARS) a connu un grand essor sous l'impulsion de l'avancée de la technologie, il convient de rappeler que sa paternité est, entre autres, liée à la sociométrie à travers les travaux de Jacob Levy Moreno au début des années 30 (Morevno, 1934). De nos jours, l'ARS est conçue comme étant à la frontière de plusieurs sciences, à savoir, les Mathématiques, la Sociologie, l'Anthropologie, la Psychologie, la Biologie, la Physique, etc. Elle a pour objectif de formaliser les éléments d'un réseau social, de les étudier quantitativement et/ou qualitativement pour une bonne compréhension de leur structure et une extraction efficiente de connaissance. Dans cette perspective, l'ARS se munit de deux cadres : un cadre théorique et un autre méthodologique. Le cadre théorique adresse la structure sociale avec l'idée que les interactions qui se passent entre une personne et ses amis influe implicitement sur son comportement social (Adaptation, Conformisme, Individualisme, Empathie, Obéissance, Développement du soi, etc.). Cette vision psycho-sociologique de l'ARS nécessite de formaliser les interactions sociales en termes de nœuds et de liens appelés graphe au sens mathématique. Le cadre méthodologique consiste en l'utilisation de données et de logiciels pour des traitements analytiques et l'expérimentation. En ce sens, l'ARS se distingue des sciences sociales et profite aujourd'hui des techniques du Big Data pour exploiter des réseaux de la taille de Facebook ou de LinkedIn.

Dans la suite de ce chapitre, nous présentons les étapes suivies pour analyser un réseau social. Puis, nous faisons un rappel de quelques concepts de la théorie de graphe pour habituer le lecteur aux termes que nous allons utiliser dans cette thèse.

2.1 Étapes du processus d'Analyse des Réseaux Sociaux

L'analyse d'un réseau social se fait souvent en suivant quatre étapes, à savoir, la collecte de données, la préparation des données, l'analyse des données et la visualisation du réseau extrait des données. Nous mentionnons également que cette analyse est faite par l'intermédiaire de logiciels facilitant la représentation, la visualisation et le traitement de statistiques indispensable pour la quantification ou la qualification des propriétés des réseaux. Nous présentons dans la suite les différentes étapes du processus d'analyse.

2.1.1 Collecte d'informations

La collecte vise à identifier les sources d'informations, à définir la stratégie de récupération et à les stocker sur un support. Elle dépend fortement des besoins de l'analyse ou des questions qui sont posées, du contexte de l'analyse ainsi que des moyens à disposition. C'est une étape cruciale car la qualité des données est gage de pertinence et de validité des connaissances qui en seront extraites. Pour une collecte de données qualitatives, le processus doit s'inscrire dans un cadre rigoureux, non linéaire et récursif. Le cadre comprend les domaines suivants : la localisation du site ou de l'individu, l'accès et l'établissement de relations, l'échantillonnage, la collecte de données, l'enregistrement d'informations, la résolution des problèmes sur le terrain et le stockage des données.

La collecte de données peut se faire de trois manières non mutuellement exclusives.

1. **Entrevue** : C'est un processus de collecte impliquant l'échange (séquence de questions/réponses) entre deux personnes ou groupes de personnes. Les entrevues peuvent être structurées, non structurées et semi-structurées. Les questions pour une entrevue structurée sont limitées, dans le même ordre et sont posées à tous les participants. Les entrevues non structurées sont limitées et permettent à celui qui collecte d'adapter les questions à la personne interrogée et au contexte. Un mélange de questions non-structurées et structurées forme une entrevue semi-structurée qui permet de recueillir des informations non verbales ainsi que des données verbales. Enfin, il convient de souligner l'importance de celui qui mène le débat car il influe sur la qualité des réponses des interrogés.
2. **Observation** : C'est un mode de collecte basé sur une attention auditive et/ou visuelle pour repérer les comportements d'un ensemble de personnes ou d'objets et créer un jeu de données en conséquence. Les observations sont définies comme des « témoignages » directs de l'activité sociale de tous les jours. Pour de bonnes observations, il faut accroître la compréhension du contexte, avoir une expérience personnelle et la connaissance du comportement des participants et être en mesure d'enrichir les premières perceptions. Avec les prises de note de terrain, il faut avoir un compte rendu exact.
3. **Document** : Cette méthode consiste à exploiter tout document pouvant contenir des informations sur les individus concernés. Ces documents proviennent des équipements, logiciels, sites web, voyages, et autres activités laissant des traces (positions géographiques, journal d'appels, cookies, etc.). Pour des informations riches, il est recommandé de croiser des documents provenant de la photographie de plusieurs disciplines (Santé, Études, Profession, Civil, etc.). Par ailleurs, l'utilisation des Systèmes d'Information Géographique (SIG) fournit une localisation

géographique des individus à tout moment. Il convient de souligner que le cadre juridique et des techniques de protection de données sensibles peuvent rendre peu fiables les données collectées.

En résumé, la collecte peut se faire avec différents moyens et selon les besoins, l'expert analyste doit faire recours aux méthodes les plus adaptés en fonction du contexte socio-culturel et juridique.

2.1.2 Préparation des données

Cette phase vise à préparer les données au processus d'analyse. Ainsi, les données sont nettoyées pour réduire les incohérences et supprimer les bruits. Les données manquantes sont complétées et des transformations sont faites pour simplifier ou agréger certaines informations. Ensuite, les données sont représentées sous forme d'une matrice d'adjacence, de liste de liens ou de liste d'adjacence selon le type de graphe. Il convient de noter que le stockage du graphe sur disque peut se faire suivant plusieurs formats [1] de fichiers dépendant du logiciel que l'on compte utilisé. Cette phase requiert une stratégie bien peaufinée pour définir les nœuds, les liens et les attributs des nœuds et/ou de liens pour pouvoir répondre aux questions à résoudre.

2.1.3 Analyse des données

Une fois les données préparées, la phase d'analyse consiste à explorer la structure du réseau en s'appuyant sur des opérations de la théorie des graphes ou de calculs statistiques. L'analyse est conduite souvent pour comprendre ou expliquer la structure du réseau. Elle permet d'identifier la position sociale des nœuds, la densité du réseau, le diamètre du réseau, la connectivité, l'impact de la structure sociale sur le comportement des individus, la dynamique de la structure sociale au fil du temps, etc. Ces questions sont couvertes par des axes de recherche assez conséquents. On peut citer, entre autres :

- la prédiction des liens ;
- l'analyse de sentiments ;
- la détection des communautés ;
- l'analyse de l'influence sociale ;
- la diffusion dans les réseaux sociaux ;
- la catégorisation des nœuds dans les réseaux sociaux ;
- la prédiction des changements du réseau au fil du temps ;
- etc.

Par ailleurs, il est à souligner que pour analyser un réseau social, on peut prendre en compte trois axes : l'échelle d'analyse, la période d'analyse et l'hétérogénéité des interactions.

1. **Échelle** : elle désigne la portée de l'analyse qui peut être structurée en trois niveaux.
    - Le *niveau nodal* ou *échelle microscopique* met l'accent sur les nœuds ou les liens du réseau. Ce niveau est utilisé lors du calcul des positions sociales ou la catégorisation des nœuds/liens au sein de la structure.

1. Exemples : GML, GraphML, CSV, PAJEK, GEXF, etc.

- Le *niveau groupal* ou *échelle mésoscopique* procède à l'analyse de la structure sous forme de groupes. Ce niveau est privilégié lors de la détection des communautés ou de l'étude de l'évolution des communautés.
- Le *niveau global* ou *échelle macroscopique*, quant à lui, correspond à une étude du réseau dans sa globalité. Ce niveau est intéressant lors de l'étude des processus de diffusion, la propagation de l'influence, etc.

2. **Temps** : il indique la période de référence de l'analyse du réseau. On part de l'hypothèse que la structure d'un réseau est construite et évolue au fil du temps au lieu de considérer qu'elle a été créée de manière instantanée. Ainsi, on peut analyser la structure dans le passé (rétrospection), le présent ou au futur (prospection). Il convient de souligner que cette analyse chronologique requiert un historique de l'évolution de la structure.
   - L'analyse *rétrospective* vise l'étude des antécédents de la structure pour comprendre ou tenter d'expliquer pourquoi le réseau actuel en mettant en exergue les phases significatives de sa formation. La taille de l'historique délimite la période d'antériorité que l'on peut considérer. Cette analyse est pratique dans le cas de lutte contre la délinquance ou la criminalité, l'identification de l'origine d'une maladie dans une population, etc.
   - L'analyse *courante* étudie l'état actuel du réseau en supposant que la structure est statique durant toute la durée de l'analyse. Cette forme d'analyse est la plus couramment rencontrée dans la littérature et est souvent à visée descriptive. Elle est utilisée pour la catégorisation des nœuds dans un réseau social, l'analyse de l'influence, le calcul des statistiques du graphe, etc.
   - L'analyse *prospective* cherche à prédire les potentiels changements du réseau au fil du temps sur la base de modèles établis. C'est une analyse faite pour prédire des liens (ex. la suggestion d'amis sur Facebook ou LinkedIn), la surveillance de l'évolution du réseau, etc.
3. **Hétérogénéité** : elle fait référence à la nature des liens et de leurs variétés. Nous distinguons ainsi les réseaux homogènes (*mono-couche*) des réseaux hétérogènes (*multi-couche*).
   - Le réseau *mono-couche* ou *monoplex* est formé d'un ensemble de nœuds avec la même catégorie de relations. Par exemple, un réseau d'individus fondé sur la base de leur relation familiale. Dans un tel réseau, deux individus partagent un lien ou encore font partie du réseau que s'ils partagent des liens de parenté.
   - Le réseau *multi-couche* ou *multiplex*, quant à lui, est une agrégation de réseaux *mono-couches*, chacun renvoyant à un type de relation spécifique entre les nœuds. Autrement dit, un nœud A peut être lié à un groupe de nœuds B par des relations amicales, à un groupe de nœuds C par des relations de parenté, à un groupe de nœuds D par un autre type de lien. Ainsi, chaque nature de lien correspond à un réseau mono-couche.

Au regard des trois axes que nous venons de développer, on peut déduire le caractère pluriel et non exhaustif des approches d'analyse dans les réseaux sociaux. Précisément, la combinaison des trois axes et de leur sous-points donne suite à dix-huit (18) approches possibles. La figure 2.1 en donne un aperçu. Les méthodes de prédiction des liens d'un nœud correspondent au point ***a*** et consiste en une analyse *prospective* et *nodale* dans un réseau *monocouche*. Les méthodes de partitionnement d'un réseau correspondent au point ***b*** et renvoient à une analyse *courante* et *globale* dans un réseau

multicouche. Le point ***c*** est un exemple d'analyse *rétrospective* d'un réseau *monocouche* pour une exploration de l'historique des propriétés d'un nœud.

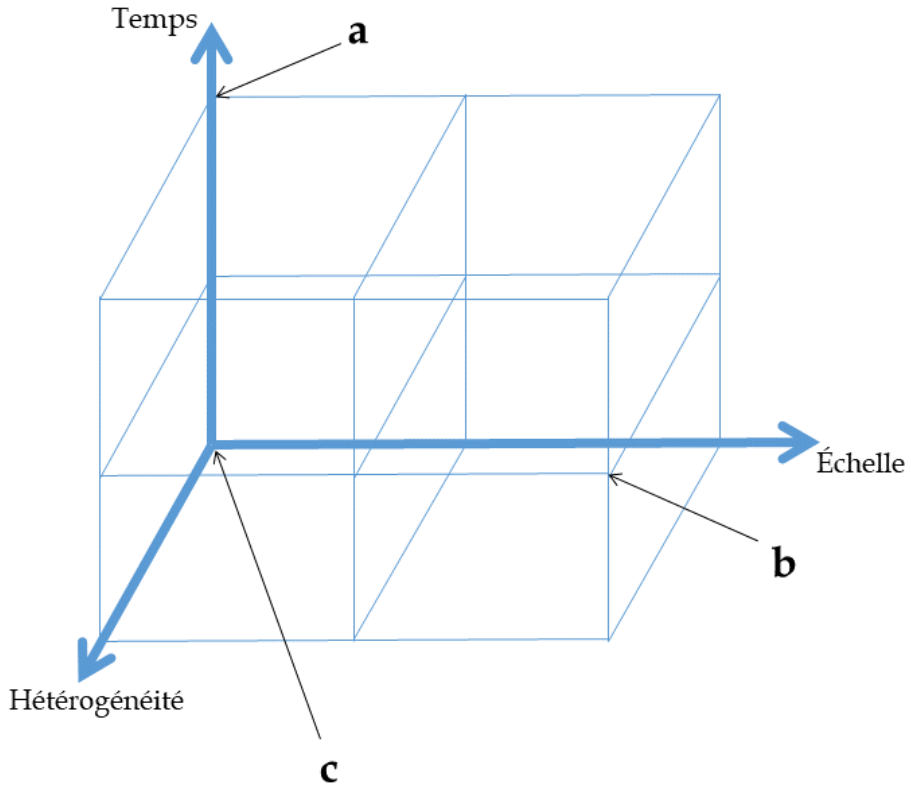


FIGURE 2.1 – Représentation de quelques approches de l'analyse des réseaux sociaux.

2.1.4 Visualisation du réseau

La visualisation du réseau est une étape fondamentale qui peut se faire à tout moment après l'obtention des données ou l'analyse de celles-ci. C'est un processus de transformation d'informations ou de connaissances au travers de représentations graphiques en vue d'une analyse visuelle, d'une exploration ou d'une explication. Dans cette perspective, la visualisation est un élément de renforcement de la perception ou de la compréhension de grands corpus. Effectuée avec un logiciel performant et interactif, elle donne à l'analyste une vue approfondie de la structure du réseau telle que l'existence de communautés, de nœuds centraux, etc. En ce sens, elle peut succéder à l'analyse comme un moyen de vérification des résultats ou de comparaison avec les perceptions.

2.2 Outils d'Analyse des Réseaux Sociaux

Il existe une multitude d'outils utilisés pour analyser les réseaux sociaux. Dans cette perspective, ils permettent le chargement des données, leur analyse et la visualisation des structures. Toutefois, il convient de souligner que certains outils ne permettent que, soit la visualisation, soit l'analyse des réseaux sociaux. De plus, les outils peuvent être sous forme de logiciels ou de bibliothèques. En général, les logiciels sont plus faciles à utiliser car disposent d'une interface graphique et interactive au profit d'un public plus élargi. Les bibliothèques requièrent des compétences en langage de programmation

et présentent plus d'opportunités d'extensions de fonctionnalités. Ainsi, l'analyste peut ajouter des fonctions conformément à ses besoins.

Nous présentons sur le Tableau 2.1 quelques outils utilisés pour l'ARS en essayant de les comparer suivant une grille de comparaison conçue autour de six critères. Le premier critère de comparaison indique la catégorie de l'outil, à savoir, s'il est un logiciel ou une bibliothèque. Le second critère désigne le type de la licence de l'outil : libre ou propriétaire. Le troisième critère met l'accent sur les systèmes d'exploitation supportant l'outil. Le quatrième critère donne une idée du (des) langage(s) d'implémentation de l'outil et donc avec lequel ou lesquels on peut étendre les fonctionnalités de l'outil. Le cinquième critère permet de savoir si l'outil prend en compte le calcul parallèle sur une architecture à mémoire partagée. Cet aspect est d'une importance capitale dans le cadre des traitements des graphes qui sont très gourmands en termes de mémoire. Le dernier critère spécifie si l'outil a des fonctionnalités permettant l'analyse de l'évolution des réseaux dynamiques. Nous rappelons, qu'il existe bien d'autres critères de comparaison de ces outils. Nos choix sont surtout guidés par le contexte de cette thèse qui adresse les réseaux dynamiques et que la performance comme l'extensibilité des outils sont de taille pour des traitements assez complexes telle que la détection des communautés. Par ailleurs, il convient de souligner qu'il est possible de combiner plusieurs outils lorsqu'un seul ne permet pas l'obtention satisfaisante des résultats escomptés.

Tableau 2.1 – Comparaison de quelques outils d'analyse des réseaux sociaux.

Outil	**Catégorie**	**Licence**	**Plateforme**	**Implémentation**	**Exécution parallèle**	**Réseaux dynamiques**
Gephi	Logiciel	Libre	Windows, Linux, MacOS	Java	Non	Oui
Statnet	Bibliothèque	Libre	Windows, Linux, MacOS	R	Non	Non
GraphStream	Bibliothèque	Libre	Windows, Linux, MacOS	Java	Oui	Oui
networkDynamic	Bibliothèque	Libre	Windows, Linux, MacOS	R	Non	Oui
NetMiner	Logiciel	Propriétaire	Windows	Java	Non	Non
iGraph	Bibliothèque	Libre	Windows, Linux, MacOS	C, R, Python	Non	Non
NetworkX	Bibliothèque	Libre	Windows, Linux	Python	Non	Non
Cytoscape	Logiciel		Windows, Linux	C++	Non	Non
Graph-tool	Bibliothèque	Libre	Linux, MacOS	Python, C++	Oui	Non
Commetrix	Logiciel	Propriétaire	Windows	Java, C++	Non	Oui

A la lecture du Tableau 2.1, nous constatons que la majorité des outils ne permet pas de faire un traitement parallèle. Pourtant, la prise en compte de cet aspect permet d'améliorer considérablement les résultats en terme de performance ou du temps d'exécution. Concernant le dernier critère, nous voyons qu'il existe quatre outils pouvant analyser les réseaux dynamiques. Cependant, l'analyse offerte n'est souvent pas satisfaisante surtout quand il s'agit des algorithmes de suivi de l'évolution de communautés ou encore les méthodes de prédiction de liens.

2.3 Termes et concepts

Comme nous l'avons souligné dans l'introduction, les réseaux sociaux sont représentés par des graphes. Par conséquent, on peut se reposer sur la théorie des graphes pour faire l'analyse de ceux-ci. Dans cette perspective, nous présentons les termes et concepts relatifs à la théorie des graphes permettant une compréhension de la suite de ce manuscrit.

2.3.1 Graphe et mesures

Définition 1 (Graphe)
Un graphe $G = (V,\ E)$ est un ensemble fini d'objets, noté $V = \{v_1, v_2, ..., v_n\}$, reliés par un ensemble d'arêtes $E = \{e_1, e_2, ..., e_m\}$. Les objets sont appelés les sommets ou nœuds du graphe et les arêtes modélisent les relations entre les objets.

Deux sommets v_i et v_j sont dit adjacents s'ils sont reliés par une arête. De plus, une arête $e_i \in E$ peut être une paire ordonnée (on parle d'arc) ou non-ordonnée (on parle de lien) formée par deux sommets v_i et v_j. Ainsi, le graphe est dit non-orienté si les arêtes sont non ordonnés. On parle aussi de graphe bidirectionnels ou symétriques. Autrement dit, un lien entre v_i et v_j indique une relation dans les deux sens. Tel est le cas des relations d'amitiés, d'appartenance à un groupe, etc. Par contre, le graphe est orienté si les arêtes sont ordonnées pour représenter l'asymétrie de la relation reliant deux sommets. Tel est le cas des relations de parenté, de prédateur-proie, de maître-esclave, etc. De plus, on parle de graphe pondéré si chaque lien est affecté d'un nombre réel positif, appelé « poids », représentant, par exemple, la distance entre deux sommets ou la densité de leur interaction. La figure 2.2 présente une illustration d'un graphe orienté et pondéré alors que la figure 2.3 illustre un graphe non orienté et non pondéré.

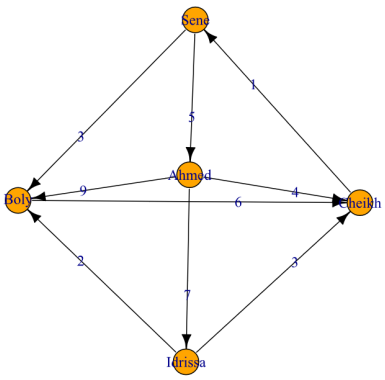


FIGURE 2.2 – Graphe orienté et pondéré.

Définition 2 (Graphe complet)
Un graphe est dit complet lorsque tous les sommets sont adjacents, autrement dit, tout couple de sommets distincts est relié par une arête. Si le graphe est orienté, il n'est complet que si chaque paire de sommets est reliée par exactement deux arcs (un dans chaque sens).

La figure 2.4 montre un graphe complet de cinq sommets, chacun étant adjacent à tous les autres sommets restants.

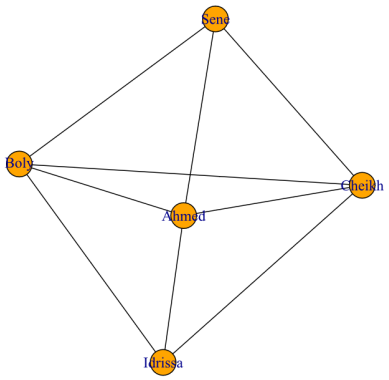


FIGURE 2.3 – Graphe non orienté et non pondéré.

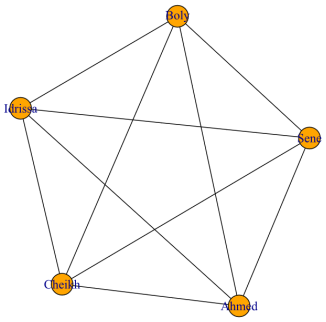


FIGURE 2.4 – Graphe complet de cinq nœuds.

Définition 3 (Clique)
Une clique d'un graphe G est un sous-ensemble des sommets de G dont le sous-graphe induit est complet. Autrement dit, chaque deux sommets quelconques sont toujours adjacents.

Une clique est dite maximale si elle possède le plus grand nombre de nœuds dans un graphe donné. L'identification de la clique maximale d'un graphe est un problème d'optimisation classé comme NP-complet. Pour ne pas subir les conséquences de la NP-dureté du problème tout en profitant de l'intérêt des cliques qui expriment une certaine cohésion ou stabilité d'un sous-ensemble du graphe, on peut faire recours à une k-Clique. En effet, une k-Clique est une clique formée par k nœuds et permet de borner la recherche aux sous-graphes de taille k en lieu et place d'une recherche exhaustive sur le graphe.

Définition 4 (Densité)
La densité d'un graphe est égale à la proportion de liens existants par rapport au nombre total de liens possibles.

En effet, le nombre maximal d'arêtes d'un graphe (ou d'un sous-graphe) de $\boldsymbol{n}$ sommets est de $\frac{n(n-1)}{2}$ s'il est non orienté et $n(n-1)$ dans le cas contraire.
Ainsi la densité d'un graphe orienté est $\delta = \frac{m}{n(n-1)}$ alors qu'elle est de $\delta = \frac{m}{\frac{n(n-1)}{2}}$ pour les graphes non-orientés, avec m le nombre de liens existants. La densité d'un graphe varie entre 0 (graphe composé de nœuds isolés) et 1 (graphe complet).

Définition 5 (Graphe et sous-graphe dense)
Un graphe ou un sous-graphe est dit dense lorsque le nombre d'arêtes (ou d'arcs) est égal ou proche au nombre maximal d'arêtes possibles.

Sur la figure 2.5, on décompte 6 arêtes pour le sous-graphe non-orienté coloré en jaune. Comme le maximum d'arêtes possibles est égal à 6, alors il est dit dense.

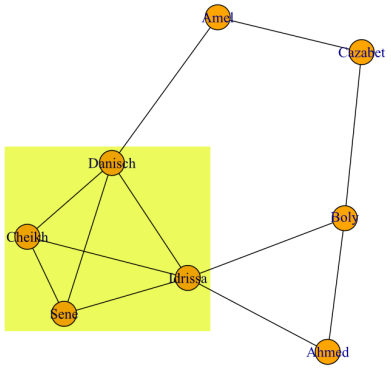


FIGURE 2.5 – Illustration du plus dense sous-graphe.

Définition 6 (Nœud d'intérêt)
Nous appelons un nœud d'intérêt ou bien « ego » tout nœud dont on s'intéresse à un instant donné.

Cet intérêt peut être lié à sa position sociale, son caractère atypique, son rôle, etc. Par exemple, le nœud d'intérêt est le porteur d'un virus dans le cadre d'un contrôle épidémiologique. Par contre, lors d'une étude sur le marketing viral, les individus ayant le plus d'influence constituent les nœuds d'intérêt.

Définition 7 (Voisinage)
Un voisinage ego-centré est constitué d'un nœud d'intérêt et de tout nœud adjacent ou étant à une distance donnée de l'égo. Les voisins du nœud d'intérêt peuvent être, éventuellement, liés entre eux.

En effet, le voisinage ego-centré d'un nœud d'intérêt u peut être de longueur 1 ou encore de longueur $k > 1$. Dans un voisinage ego-centré de longueur 1, l'ego est directement lié à ses voisins. En revanche, un voisinage ego-centré de longueur k est constitué du nœud u ainsi que tous les nœuds qui peuvent être atteints, à partir de u, via une longueur de trajet inférieure ou égale à k. La valeur de k est comprise entre 1 et l'excentricité [2] de u. Sur la figure 2.6, les nœuds colorés en orange font partie du voisinage de longueur 1 de l'égo qui est coloré en bleu. Si nous considérons les nœuds colorés en rose, nous obtenons le voisinage de longueur 2. De plus, si les nœuds colorés en vert sont inclus, nous obtenons le voisinage de longueur 3. Notons que, par abus de langage, le « voisinage ego-centré » est souvent appelé « réseau ego-centré ».

Définition 8 (Diamètre)
Le diamètre d'un graphe signifie la plus grande distance possible qui peut se trouver entre deux nœuds quelconques :

$$Dm = max_{u,v} d(u, v) \tag{2.1}$$

La distance $d(u, v)$ est dite géodésique et est définie par la longueur du plus court chemin entre deux nœuds.

2. Voir la définition 9.

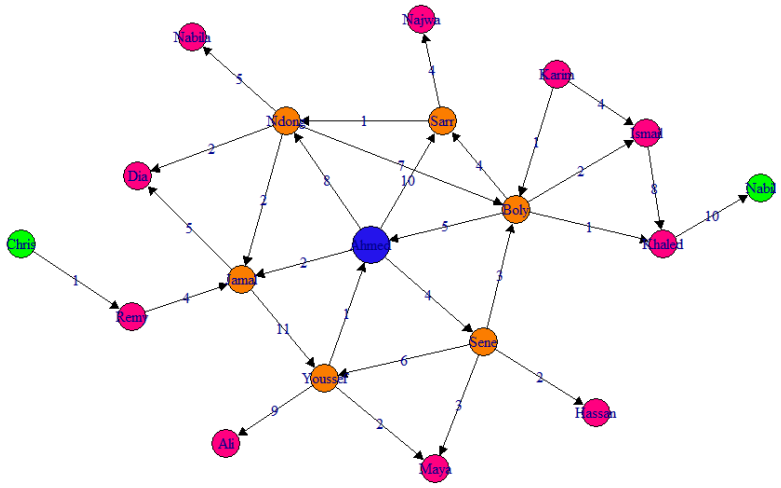


FIGURE 2.6 – Illustration du voisinage de longueur 3 de l'ego « Ahmed ».

Définition 9 (Excentricité)
L'excentricité d'un nœud u, notée $e(u)$, désigne le nombre de liens servant à connecter u au nœud le plus distant dans un graphe G. Autrement dit, $e(u)$ représente la distance maximale qui peut exister entre u et n'importe quel autre nœud du graphe. L'excentricité d'un nœud u est donnée par la formule suivante :

$$e(u) = max_{\forall v \in V} d(u, v) \tag{2.2}$$

Définition 10 (Centralité de degré)
La centralité de degré d'un nœud désigne le nombre de ses voisins directs, c'est-à-dire, le nombre de liens qui lui sont incidents.

Cette mesure évalue le niveau de connectivité d'un nœud dans un graphe donné. Formellement, la centralité de degré d'un nœud u est ainsi définie :

$$d(u) = \sum_{\forall v \in V,\ u \neq v} a_{u,v} \tag{2.3}$$

Où $a_{u,v}$ signifie l'indice d'adjacence, valant 1 si $(u, v) \in E$ et 0 dans le cas contraire.

Dans un graphe valué, le degré peut être pondéré par les poids des liens, on parle alors de degré pondéré. Cette mesure permet de prendre en compte l'intensité de communication entre les éléments du réseau. Toutefois, elle fait disparaître la dimension topologique du réseau (nombre de voisins). Soit W la matrice des poids dans laquelle w_{uv} représente le poids de lien entre le nœud u et le nœud v. Le degré pondéré d'un nœud u est défini par la somme des poids de tous les liens incidents à u :

$$wd(u) = \sum_{\forall v \in V,\ u \neq v} w_{uv} \tag{2.4}$$

On appelle « degré moyen » d'un graphe la somme de la centralité de degré de tous les nœuds divisée par le nombre de nœuds du graphe. Soit :

$$\overline{d}(G) = \frac{\sum_{u \in V} d(u)}{n} \tag{2.5}$$

Définition 11 (Centralité d'intermédiarité)
La centralité d'intermédiarité d'un nœud mesure le nombre de fois où il est l'intermédiaire sur le plus court chemin entre deux autres nœuds. La communication des nœuds non adjacents dépend d'autres acteurs qui ne se trouvent pas sur le chemin. Tels acteurs ont la capacité d'interrompre le passage d'information. Ainsi, plus un acteur se trouve au « milieu » du chemin, plus il est central de ce point de vue.

Formellement, la centralité d'un nœud u_i est définie ainsi :

$$C_B(u_i) = \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{g_{jk}(u_i)}{g_{jk}} \tag{2.6}$$

Tel que $g_{jk}(u_i)$ désigne le nombre total des chemins géodésiques entre les nœud u_j et u_k qui passent par le nœud u_i, et g_{jk} est le nombre total des plus courts chemins entre les nœud u_j et u_k.

2.3.2 Communautés dans les réseaux sociaux

Définition 12 (Communauté)
Une communauté désigne un ensemble de nœuds fortement liés entre eux et faiblement liés au reste du réseau. Autrement dit, c'est un sous-graphe dont la densité est plus forte à l'intérieur qu'à l'extérieur.

Cette définition montre que la découverte des communautés d'un réseau repose sur l'identification d'une valeur de la densité à partir de laquelle un ensemble de nœuds devient une communauté. Ainsi, la manière d'établir, d'estimer ou d'évaluer cette valeur donne suite à plusieurs techniques de détection sur lesquelles nous reviendrons au chapitre suivant. Toutefois, nous soulignons qu'importe la technique, sa visée demeure la découverte de communautés pertinentes avec une sémantique circonscrite dans le contexte étudié. La figure 2.7 montre un réseau avec trois communautés.

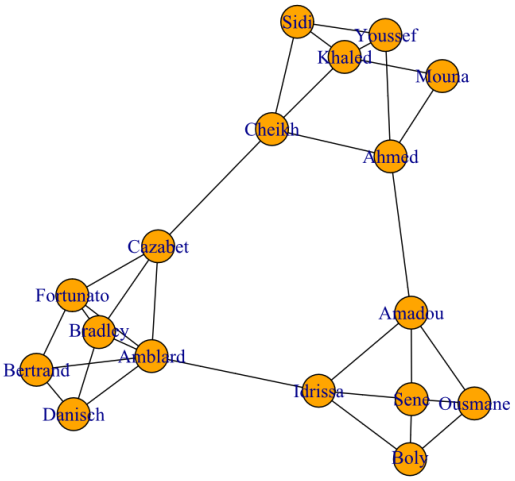


FIGURE 2.7 – Exemple d'un réseau structuré en trois communautés.

Définition 13 (Communautés globales)
Les communautés globales sont des sous-graphes qui résultent du découpage du réseau entier en plusieurs partitions.

Définition 14 (Communauté locale)
Une communauté locale est une communauté construite autour de quelques nœuds spécifiques.

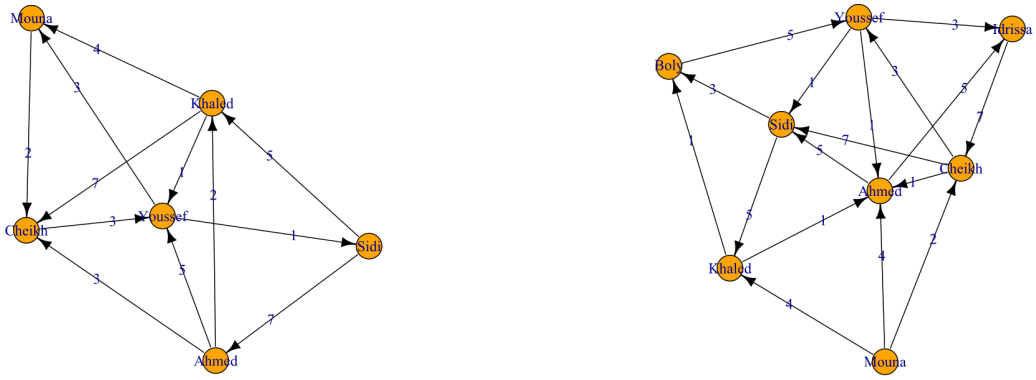
Définition 15 (Communauté ego-centrée)
Une communauté ego-centrée est une communauté construite autour d'un seul nœud d'intérêt, appelé « ego ».

En effet, la communauté ego-centrée peut être vue comme une extension de communauté locale. La première est obtenue en supposant qu'elle émerge à partir d'un nœud cible jugé d'intérêt. De plus, si une communauté est construite en considérant à la fois deux nœuds d'intérêt, alors nous parlons de communautés bi-ego-centrées ou de communautés multi-ego-centrées s'il y en a plusieurs.

Définition 16 (Communauté chevauchante)
Une communauté est chevauchante si une partie de ses nœuds appartient simultanément à plus d'une communauté.

Définition 17 (Communauté dynamique)
Une communauté est dite dynamique si sa structure topologique[3] *ou sa composition*[4] *change au fil du temps.*

En d'autres mots, la visualisation de la topologie de la communauté entre deux instants donne des rendus divergents. Sur la figure 2.8, nous notons l'augmentation du nombre de nœuds de la communauté C entre t et $t+$. Tel est aussi le cas pour l'adjacence des nœuds.



(a) Une communauté C à l'instant t.

(b) La communauté C à l'instant $t+1$.

FIGURE 2.8 – Illustration des changements d'une communauté dynamique.

Définition 18 (Algorithme de détection de communautés)
C'est une procédure qui vise à trouver les sous-groupes du réseau qui remplissent les critères d'une communauté. Il prend en entrée un réseau et sort une liste de communautés, chacune avec ses membres.

Définition 19 (Algorithme de détection de communautés non-déterministe)
Un algorithme est dit « non-déterministe » s'il détecte des communautés différentes en l'exécutant de manière répétitive sur le même jeu de données.

Définition 20 (Algorithme de détection de communautés instable)
Un algorithme est dit « instable » s'il trouve des communautés fortement différentes sur deux graphes topologiquement proches.

3. Apparition ou disparition de nœud/liens.
4. Citons par exemple le changement des valeurs des poids de liens ou aussi le changement de l'orientation des liens.

Cependant, on note que l'instabilité est encore plus évidente pour les algorithmes non-déterministes.

2.4 Conclusion

Dans ce chapitre, nous avons présenté les principales phases du processus d'ARS ainsi que les outils utilisés pour l'analyse. En outre, nous avons rappelé quelques concepts de la théorie des graphes nécessaires pour bien comprendre la suite de cette thèse ainsi qu'un certain nombre de métriques associées comme la centralité de degré, la densité, l'excentricité, etc.

Dans le chapitre suivant, nous présentons l'état de l'art relatif à la détection de communautés statiques et dynamiques. Nous mettons le focus plus précisément sur les approches de détection de communautés locales ou ego-centrées.

Chapitre 3

État de l'art

L'une des premières études de la structure de communautés date de 1955 et a été effectuée par (Weiss et Jacobson, 1955) dans le but de rechercher des groupes de travail dans une agence de gouvernement. Ensuite, le problème de la détection de communautés a connu un regain d'intérêt au début du troisième millénaire avec les travaux de (Girvan et Newman, 2002) qui ont conclu que la structure en communautés fait partie des caractéristiques des réseaux de type petit-monde (Jeffrey Travers, 1969). Autrement dit, les réseaux du monde réel sont structurés de sorte que les éléments partageant des propriétés similaires tendent à former des groupes denses qui sont peu connectés entre eux.

Aujourd'hui, la détection de communautés est utilisée dans plusieurs domaines, à savoir, les réseaux de neurones artificiels (Liu, Moser et Aviyente, 2014), les systèmes d'énergie électrique (Z. Chen, Xie et Q. Zhang, 2015), l'étude de propagation des maladies infectieuses (Petkova et al., 2016 ; Ratnayake et al., 2016 ; McColgan et al., 2017), l'amélioration des stratégies de marketing (W. Wang, 2016 ; Kaple, Kulkarni et Potika, 2017), les réseaux sociaux mobiles (Ke Xu et al., 2016 ; Wan et al., 2016 ; Zheng, Zhao et Z. Kang, 2017), etc.

En effet, la thématique de la détection de communautés peut être divisée en deux grandes catégories, à savoir, la détection de communautés globales et la détection de communautés locales. Les algorithmes de la première catégorie cherchent à partitionner le réseau en plusieurs communautés, tandis que les algorithmes de la seconde catégorie s'intéressent, plutôt, à identifier les communautés de quelques nœuds d'intérêt. Dans cette perspective, une communauté ego-centrée peut être vue comme une extension d'une communauté locale, à la différence que :

- Le centre d'intérêt d'une communauté ego-centrée est toujours constitué d'un seul nœud ;
- La portée d'une communauté ego-centrée peut être élargie au delà du voisinage direct du nœud d'intérêt.

La définition de « communauté » a toujours été sujet du débat. En effet, il n'y a pas de consensus sur la définition quantitative de ce qu'est une communauté (Fortunato, 2010). Chaque algorithme définit ses propres critères quantitatifs. Qualitativement, une communauté doit vérifier deux critères, à savoir :

1. La cohésion interne, qui indique que la densité de communauté est élevée ;
2. La séparation du reste du réseau, qui désigne que le nombre de liens à l'intérieur de communauté est considérablement supérieur au nombre de liens situés à l'extérieur.

La définition qualitative révèle deux contraintes discordantes, à savoir, la cohésion interne et la disjonction avec le reste du réseau. C'est pourquoi, la détection de communautés est souvent perçue comme étant un problème d'optimisation multiobjectif (D. Chen et al., 2016 ; Rahimi, Abdollahpouri et Moradi, 2017). Dans cette optique, la méthode X peut détecter des communautés bien

séparées du reste du réseau que la méthode Y. Mais, en revanche, la méthode Y peut avoir des communautés ayant une cohésion interne plus bonne. D'une part, les algorithmes qui cherchent à détecter les communautés en ne s'appuyant que sur la connectivité de nœuds, obtiennent des communautés fortement isolées mais qui sont peu denses. D'autre part, les algorithmes utilisant les cliques maximales trouvent les communautés les plus denses, mais, très liées au reste du réseau. Par conséquent, comme dans tout problème multiobjectif, il existerait un front de Pareto sur lequel se trouvent les méthodes dont la cohésion et la séparation sont les plus pertinentes possibles.

Aujourd'hui, on décompte un grand nombre de travaux relatifs à la détection de communautés (Fortunato et Hric, 2016 ; Rossetti et Rémy Cazabet, 2017 ; Khan et Niazi, 2017). Toutefois, il est à noter que la majorité des algorithmes existants reposent entièrement sur les propriétés statiques du réseau et négligent souvent le fait que les réseaux évoluent au fil du temps (Qi et al., 2017 ; M. Wang et al., 2017 ; W. Li et al., 2018). De ce fait, nous constatons que le nombre de travaux consacrés à la détection de communautés statiques est largement supérieur à celui visant les communautés dynamiques. Pourtant, les interactions entre les nœuds changent constamment dans le temps et se traduisent par l'ajout et/ou la suppression de nœuds et/ou de liens. Cette dynamique exerce un impact fort sur la structure de communautés.

L'objectif ce chapitre est de faire l'état de l'art des principales approches de détection de communautés statiques et dynamiques tout en ayant un regard sur les approches de détection de communautés locales ego-centrées. La suite du chapitre est structurée comme suit. Nous présentons les principales approches de détection de communautés locales statiques dans la section 3.1. Ensuite, nous décrivons les principales approches de suivi de l'évolution de communauté dynamiques dans la section 3.2. Nous terminons ce chapitre par un bilan global des solutions existantes dans la section 3.3.

3.1 Détection de communautés statiques

Après avoir passé en revue les travaux existants, nous retenons trois approches de détection de communautés, à savoir, les méthodes à base d'optimisation de fonction de qualité, de mesures de similarité et de triangles. Nous finissons cette section par une analyse des limites de ces solutions.

3.1.1 Approche basée sur une fonction de qualité

C'est l'une des approches les plus étudiées dans la littérature (Clauset, 2005 ; Ngonmang, Tchuente et Viennet, 2012 ; Xiang et al., 2016 ; Creusefond, Largillier et Peyronnet, 2016 ; Hollocou, Bonald et Lelarge, 2017). Elle consiste à exprimer les critères quantitatifs de communauté sous forme d'une métrique, appelée « fonction de qualité ». Cette métrique associe à toute communauté un score de qualité. La définition d'une fonction de qualité dépend de la façon dont l'algorithme perçoit les communautés. En outre, l'application d'une fonction de qualité une seule fois ne garantit pas la pertinence des communautés. Ainsi, le résultat de la fonction de qualité fait souvent l'objet d'une optimisation qui a pour objectif de minimiser ou de maximiser le score de qualité en vue de l'obtention de communautés plus pertinentes. Dans la section 3.1.1.1, nous présentons quelques exemples d'expression de fonctions de qualité avant d'aborder quelques heuristiques d'optimisation dans la section 3.1.1.2.

3.1.1.1 Exemples de fonctions de qualité

Modularité locale

La modularité locale est un exemple de fonction de qualité qui est définie par rapport à un découpage de l'entourage de communauté. Le premier exemple de modularité locale est celui de (Clauset, 2005), où l'auteur définit un découpage en trois ensembles, à savoir :

- le cœur de la communauté, noté $\mathcal{C}$, qui constitue l'ensemble des nœuds dont tous les voisins se trouvent dans la communauté ;
- la bordure de la communauté, notée $\mathcal{B}$, qui comprend les nœuds qui se trouvent dans la communauté et ayant au mois un voisin ailleurs ;
- l'extérieur de la communauté, noté $\mathcal{U}$, qui est formé des nœuds situés hors de la communauté mais y ayant au moins un voisin.

La Figure 3.1, extraite de (Clauset, 2005), en donne une illustration.

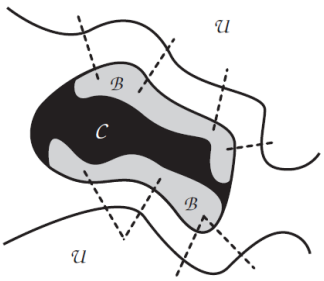


Figure 3.1 – Division de l'entourage d'une communauté en trois ensembles : le cœur $\mathcal{C}$, la bordure $\mathcal{B}$ et l'extérieur de la communauté, appelé $\mathcal{U}$.

Après avoir découpé l'entourage de communauté, Clauset propose sa fonction de qualité définie comme suit :

$$R = \frac{\sum_{u,v \in V, u \neq v} \mathcal{B}_{u,v} \delta_(u,v)}{\sum_{u,v \in V, u \neq v} \mathcal{B}_{u,v}} = \frac{I}{T} \tag{3.1}$$

Tel que :

$$\mathcal{B}_{u,v} = \begin{cases} 1 & \text{Si } (u,v) \in E \ \& \ u,v \in \mathcal{B} \\ 0 & \text{Sinon} \end{cases} \tag{3.2}$$

Et $\delta_(u,v)$ vaut 1 si $u \in \mathcal{B}$ et $v \in \mathcal{C}$ ou vice versa, sinon, il est égal à 0. I représente le nombre de liens n'ayant aucune extrémité dans $\mathcal{U}$ et T indique le nombre de liens ayant au moins une extrémité dans $\mathcal{B}$.

La faiblesse intrinsèque de la modularité locale a été exposée dans (J. Chen, Zaiane et Goebel, 2009). Cette faiblesse peut être expliquée comme suit : soient $\{O_1, O_2, ..., O_{11}\}$ des nœuds marginaux, c'est-à-dire qui sont à peine liés aux nœuds de D (voir la figure 3.2). Supposons que tous les nœuds de S (qui constitue la partie extérieure de D), à l'exception de O_1 et O_9, diminuent la valeur de R s'ils sont ajoutés à D. De ce fait, la tentative de maximiser la modularité locale entraine l'ajout des nœuds marginaux à D. La raison est que l'ajout de ces nœuds ne modifie pas le nombre de liens externes

et augmente de 1 le nombre de liens internes. Ainsi, en plus des membres effectifs, la communauté résultante contiendra des nœud faiblement liés et éloignés du nœud d'intérêt. La Figure 3.2 en présente une illustration.

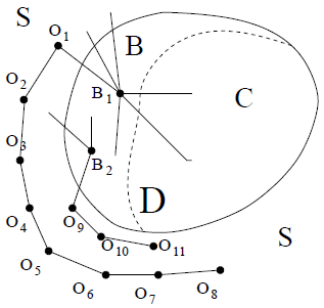


FIGURE 3.2 – Problème d'adhésion des éléments marginaux en utilisant la modularité locale. Figure extraite de (J. CHEN, ZAIANE et GOEBEL, 2009).

Des solutions ont été proposées pour pallier à cette faiblesse, notamment, les travaux présentés dans (J. CHEN, ZAIANE et GOEBEL, 2009 ; NGONMANG, TCHUENTE et VIENNET, 2012) qui visent à intégrer les distances des nœuds voisins par rapport au nœud d'intérêt.

Coupe normalisée

La coupe normalisée, appelée aussi la conductance, est définie par le nombre de liens connectant une communauté C au reste du réseau $\overline{C}$ divisé par la somme de degrés des nœuds qui se trouvent dans C (SHI et MALIK, 2000). Formellement, la conductance est exprimée ainsi :

$$\phi(C) = \frac{cut(C)}{\min(\sum_{\forall u \in C} d(u), \sum_{\forall v \in \overline{C}} d(v))} \tag{3.3}$$

Où $cut(C)$ désigne le nombre de liens entre C et les autres nœuds du réseau. $\sum_{\forall u \in C} d(u)$ est la somme des degrés des nœuds de C et $\sum_{\forall v \in \overline{C}} d(v)$ la somme des degrés des nœuds qui ne se trouvent pas dans C. En effet, quand on parle de communauté ego-centrée dans les réseaux sociaux, $\sum_{\forall u \in C} d(u)$ qui représente la somme des degrés des nœuds d'une communauté est généralement petit devant $\sum_{\forall v \in \overline{C}} d(v)$, soit la somme des degrés des nœuds dans le reste du réseau.

La valeur de $\phi(C)$ varie entre 0 (une composante connexe) et 1 (une communauté sans liens internes, ou ayant une infinité de liens externes).

Il est clair que la conductance fait intervenir les deux aspects qualitatifs de la définition de communauté : la cohésion interne (valeur du dénominateur) et la séparation du reste du réseau (valeur du numérateur). Ainsi, plus la conductance d'une communauté est proche de zéro, plus elle est censée être de bonne qualité.

Même si la conductance a été critiquée pour la production de « communautés de cavernes » [1] (LIM, U. KANG et FALOUTSOS, 2014), les algorithmes utilisant cette mesure sont capables de produire des communautés possédant les caractéristiques d'un réseau petit-monde (ABRAHAO et al., 2012). En

1. Les « communautés de cavernes » désignent un type particulier de structure en communautés où chaque individu connait bien ceux qui sont avec lui dans la même « caverne », alors qu'il en sait très peu sur les personnes qui se trouvent dans les autres « cavernes » (U. KANG et FALOUTSOS, 2011).

outre, la conductance est souvent utilisée dans les méthodes de détection de communautés chevauchantes (Whang, Gleich et Dhillon, 2016 ; Su et C.-C. Lee, 2017). Citons également que dans de nombreux articles et études empiriques, la conductance est considérée comme l'une des mesures efficaces (Jaewon Yang et Leskovec, 2015 ; Lim, W.-J. Lee et al., 2017).

Le principe de base de la conductance a été reformulé dans plusieurs travaux de recherche. Citons par exemple le travail de (Hamann, Röhrs et Wagner, 2017) qui est l'un des travaux les plus récents utilisant la conductance comme fonction de densité.

Enfin, pour comparer la différence conceptuelle entre la conductance et la modularité locale, nous pouvons dire que la première sélectionne tous les liens du voisinage, alors que la modularité locale ne considère que les liens ayant une extrémité dans la bordure $\mathcal{B}$. Ceci est particulièrement utile dans le cas où la densité ne se limite pas au centre de la communauté.

Conductance pondérée

La conductance pondérée adopte le même principe de la métrique précédente, sauf que celle-ci utilise les poids des liens au lieu de la somme des degrés. Un exemple de cette métrique est proposé dans (Lu, Wen et Cao, 2013) où la fonction de densité est ainsi définie :

$$\Phi(C) = \frac{\sum_{w_C}^{out}}{\sum_{w_C}^{in} + \sum_{w_C}^{out}} \tag{3.4}$$

avec :

- $\sum_{w_C}^{out}$ représentant la somme des poids des liens dont une seule extrémité se trouve dans C ;
- et $\sum_{w_C}^{in}$ désignant la somme des poids des liens ayant leurs deux extrémités dans C.

L'avantage principal de cette fonction est qu'elle prend en compte la pondération des liens. Cependant, elle n'est pas capable de gérer l'orientation de liens. Autrement dit, la fonction Φ ne permet pas de distinguer un lien de poids cinq ayant deux voisins d'un autre du même poids ayant trois voisins.

3.1.1.2 Optimisation de fonctions de qualité

Le processus d'optimisation d'une fonction de qualité est constitué de trois phases. De ce fait, toutes les heuristiques d'optimisation adoptent le même processus mais peuvent différer dans l'implémentation des trois phases. Nous présentons ci-dessous ces trois phases et nous discutons quelques variantes possibles.

1. La première phase est l'initialisation de communauté. Elle consiste à trouver l'ensemble de nœuds, appelé « graine », qui représente la composition initiale de la communauté. La graine peut être constituée d'un ou de plusieurs nœuds. Le choix de la graine dépend de la nature des communautés que nous voulons identifier. Par exemple, si nous cherchons à détecter une communauté spécifique, nous pouvons initialiser la graine avec le nœud d'intérêt (Lu, Wen et Cao, 2013). Par contre, si nous avons déjà une idée sur l'état initial de la communauté et nous cherchons à la compléter, il est plus utile de considérer que la graine comprend un groupe de nœuds. Cette variante est utilisée dans (Danisch, Guillaume et Le Grand, 2014a). En outre, si nous

nous intéressons aux différentes formes d'une communauté, il est recommandé d'initialiser aléatoirement la graine. Ainsi, à chaque construction de communauté, nous aurons une composition différente. C'est la variante choisie dans (Lancichinetti, Fortunato et Kertész, 2009) ;

2. La seconde phase est la définition d'une condition d'arrêt du processus d'optimisation. La condition d'arrêt la plus intuitive est de continuer tant que les voisins du nœud d'intérêt ne sont pas tous parcourus (Hamann, Röhrs et Wagner, 2017). Il est également possible, comme proposé dans (J. Chen, Zaiane et Goebel, 2009), de considérer que l'optimisation du score de qualité est bornée par un seuil donné. Ainsi, le processus d'optimisation s'arrête une fois que le seuil est atteint même s'il reste des voisins du nœud d'intérêt qui ne sont pas encore parcourus ;
3. La troisième phase est l'optimisation de la fonction de qualité. En effet, l'optimisation désigne une minimisation (J. Chen, Zaiane et Goebel, 2009) ou une maximisation (Lu, Wen et Cao, 2013) du score de qualité. L'objectif de cette phase est d'ajouter ou de supprimer les nœuds qui optimisent plus le score de qualité. Ce processus est répété jusqu'à ce que la condition d'arrêt soit vérifiée.

3.1.2 Approche à base de similarité

Les algorithmes de cette catégorie définissent une mesure de similarité entre les nœuds. L'intuition derrière cette approche est que plus le score de similarité entre deux nœuds est élevé, plus la connectivité entre eux est jugée forte. C'est pourquoi les algorithmes de cette approche considèrent que les nœuds ayant des similarités élevées par rapport à un nœud d'intérêt constituent sa communauté.

Dans (Huang et al., 2011), les auteurs proposent une fonction de qualité basée sur une mesure de similarité introduite dans (Radicchi et al., 2004). Leur fonction de qualité est définie comme suit :

$$T(C) = \frac{S_{in}^{C}}{S_{in}^{C} + S_{out}^{C}} \tag{3.5}$$

où S_{in}^{C} désigne la similarité interne, soit la somme des similarités des nœuds de la communauté C et S_{out}^{C} représente la similarité externe, définie par la somme des similarités des nœuds adjacents à ceux de C. Par la suite, les auteurs proposent un optimisation de la fonction $T(C)$ en suivant les deux étapes suivantes. Premièrement, la communauté est initialisée avec le nœud d'intérêt. Deuxièmement, le voisin le plus similaire au nœud d'intérêt est ajouté à la communauté s'il fait augmenter le score de $T(C)$. Dans le cas contraire, l'algorithme passe au suivant jusqu'à ce que tous les voisins du nœud d'intérêt soient visités.

Un autre exemple utilisant les mesures de similarité est réalisé dans (Danisch, Guillaume et Le Grand, 2014b) où l'algorithme proposé fonctionne en trois étapes. Tout d'abord, il calcule la similarité des nœuds du réseau par rapport au nœud d'intérêt. Ensuite, il trie les nœuds selon leur score de similarité. Enfin, il recherche des irrégularités dans l'ordre des scores de similarité. L'intuition derrière cela est que les nœuds proches d'un nœud d'intérêt se caractérisent par leur score de similarité élevé contrairement aux autres nœuds qui auront un score faible. Par ailleurs, le tracé de la courbe des scores de similarité des différents nœuds fait observer une variation brutale de l'allure de la courbe dès que les scores sont assez distants. Cela indique que les nœuds se trouvant avant cette variation brusque forment une communauté du nœud d'intérêt. La Figure 3.3 en donne une illustration. En outre, si la courbe de similarité décroît régulièrement en suivant une loi sans-échelle, cela signifie que,

soit le nœud appartient à plusieurs communautés de différentes tailles, soit il n'appartient à aucune communauté.

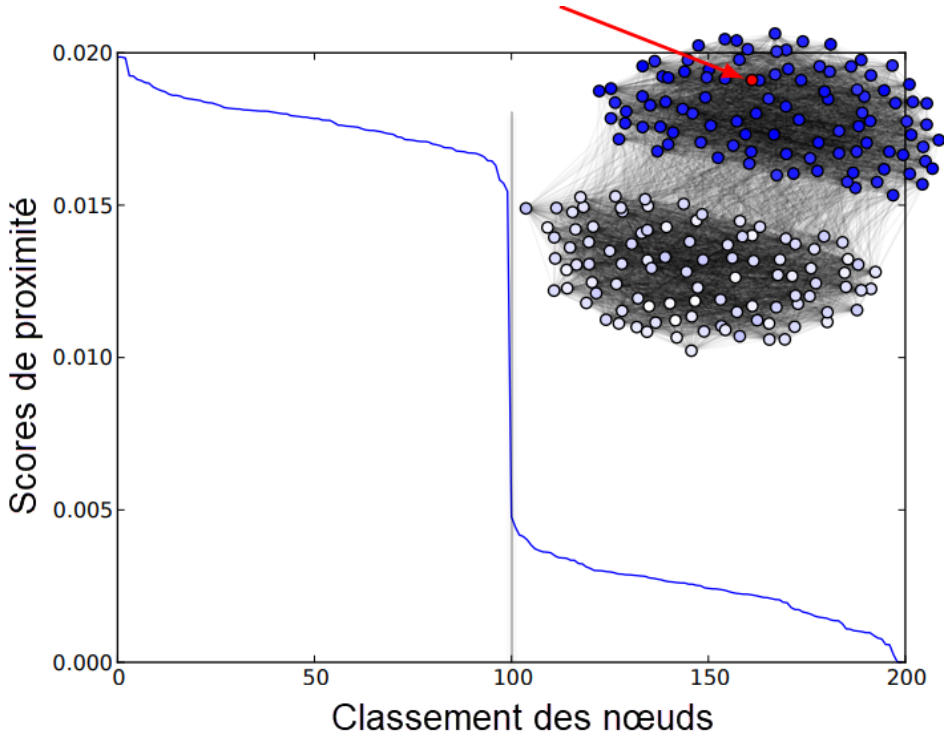


FIGURE 3.3 – Illustration d'une courbe de similarité des nœuds d'un graphe par rapport à un nœud d'intérêt. Figure extraite de (DANISCH, 2015).

Sur la Figure 3.3, l'axe des abscisses désigne le classement des nœuds du graphe en fonction de la similarité au nœud d'intérêt. L'axe des ordonnées indique la variation des scores de similarité des nœuds par rapport au nœud d'intérêt. Le graphe est composé de deux sous-graphes d'Erdos-Renyi $(100, 0.5)$. Deux nœuds dont chacun se trouve dans un sous-graphe différent ont une probabilité d'être liés de 0.1. Sur la représentation du graphe, le nœud d'intérêt est indiqué par une flèche et l'intensité de la couleur des nœuds est proportionnelle à leur degré de similarité au nœud d'intérêt. Sur la courbe montrant les scores de similarité triés par ordre décroissant, nous constatons deux plateaux séparés par une forte décroissance aux environs du point 100 sur l'axe des abscisses. En visualisant le graphe, nous voyons que les deux plateaux correspondent à deux communautés dont chacune comprend le nœud d'intérêt. La décroissance de la courbe, marquée par une ligne verticale, indique la fin des nœuds de la première communauté et le début des nœuds appartenant à la seconde.

Plusieurs autres travaux adoptant le principe de similarité ont été proposés. Citons par exemple : (HOLLOCOU, BONALD et LELARGE, 2016 ; PECH et DONG, 2017 ; CHOWDHARY, LÖFFLER et SMITH, 2017 ; DING, J. ZHANG et Jing YANG, 2018).

3.1.3 Approche basée sur les triangles

En théorie des graphes, un triangle désigne un sous-graphe de trois nœuds et trois liens. Les algorithmes basés sur les triangles cherchent à détecter toutes les communautés locales auxquelles appartient un nœud d'intérêt. L'intuition derrière l'utilisation des triangles est qu'ils sont souvent l'intersection de plusieurs sous-graphes. De plus, l'abondance de triangles dans un sous-graphe implique

un niveau élevé de cohésion. Rappelons que la cohésion interne est l'une des propriétés fondamentales de communauté.

Dans (Tsourakakis et al., 2013), les auteurs partent du constat que les communautés locales extraites via une maximisation du degré moyen sont généralement des grands sous-graphes ayant une faible densité et un diamètre élevé. Pour pallier à cette faiblesse, ils proposent une fonction dont l'optimisation permet d'obtenir des communautés locales de forte densité et de petit diamètre. Pour ce faire, les auteurs cherchent à extraire des k-quasi-cliques qui maximisent la fonction suivante :

$$f_k(C) = g(e[C]) - k.h(|C|),\ k \in [0,1] \tag{3.6}$$

L'intuition derrière la formule de $f_k(C)$ est d'équilibrer deux aspects : le premier terme $g(e[C])$ favorise la densité de communautés locales tandis que le second terme $-k.h(|C|)$ pénalise l'existence des grands sous-graphes. $e[C]$ représente le nombre de liens de la communauté C et $|C|$ le nombre de nœuds.

Après avoir proposé leur fonction de qualité, les auteurs proposent deux méthodes d'optimisation qui sont *GreedyOQC* et *LocalSearchOQC*. La première consiste à supprimer itérativement les nœuds ayant les plus petites valeurs du degré moyen tout en maximisant le score de qualité. La seconde heuristique cherche à trouver un local optimal de qualité de sorte que si un seul nœud est ajouté ou supprimé, le score de qualité diminue. Pour ce faire, *LocalSearchOQC* sélectionne, au début, un nœud aléatoire. Puis, elle continue d'ajouter des nœuds à l'ensemble C tant que le score de qualité augmente. Si aucun nœud ne peut être ajouté, *LocalSearchOQC* cherche à supprimer de C les nœuds qui maximisent le score de qualité. Ce processus est maintenu jusqu'à l'atteinte d'un optimum local ou d'une valeur T_{max} signalant le nombre maximal d'itérations.

L'aspect triangle dans la méthode précédente réside dans le fait d'utilisation des cliques vu qu'elles sont riches de triangles.

Dans (Fagnan, Zaiane et Barbosa, 2014), les auteurs définissent la qualité de communauté en fonction de triades internes et externes. La fonction de densité proposée est constituée de deux métriques, à savoir, T_{in} et T_{ex} :

$$T_{in} = \frac{1}{6} \sum_{i,j,k \in C} a_{i,j}\ a_{j,k}\ a_{i,k} \tag{3.7}$$

$$T_{ex} = \frac{1}{2} \sum_{i,j,k \in N_{\{u,2\}}} a_{i,j}\ a_{j,k}\ a_{i,k} \tag{3.8}$$

La première métrique désigne le nombre de triades dont tous les nœuds se trouvent dans C. La seconde représente le nombre de triades dont un seul nœud appartient à C. Ainsi, selon cette définition, les triades ayant exactement deux nœuds en C ne sont ni internes ni externes.

En se basant sur ces deux métriques, les auteurs expriment leur fonction de densité :

$$T = T_{in}. \begin{cases} T_{in} - T_{ex} & \text{Si } T_{in} \geqslant T_{ex} \\ 0 & \text{Sinon} \end{cases} \tag{3.9}$$

Le découpage proposé dans (Fagnan, Zaiane et Barbosa, 2014) s'inspire principalement de celui présenté dans (Clauset, 2005). Sauf que les méthodes de parcours des nœuds sont différentes.

L'algorithme d'optimisation proposé dans (Fagnan, Zaiane et Barbosa, 2014) consiste à initialiser la communauté avec un nœud d'intérêt u. Puis, l'algorithme place tous les voisins de u dans l'ensemble $N_{\{u,2\}}$. Ensuite, pour chaque itération, l'algorithme sélectionne un nœud v de $N_{\{u,2\}}$. Si son ajout maximise la fonction T, l'algorithme ajoute le nœud v à C_u ainsi que tous les voisins de v dans l'ensemble $N_{\{u,2\}}$. Ce processus est répété jusqu'à ce qu'aucun nœud de $N_{\{u,2\}}$ ne puisse maximiser le score de T.

Les auteurs de (Fanrong et al., 2014) cherchent à détecter l'ensemble de communautés locales dont chacune contient le nœud d'intérêt. Pour ce faire, ils ont proposé un algorithme basé sur les cliques maximales qui fonctionne en deux phases. La première consiste à utiliser l'algorithme de Bron-Kerbosch (Bron et Kerbosch, 1973) pour extraire toutes les cliques maximales contenant le nœud d'intérêt. Ces cliques maximales sont considérées comme des communautés locales initiales. Les cliques dont aucun nœud n'a été attribué à une communauté locale sont considérées comme des communautés locales non classées. Dans la seconde phase, les auteurs utilisent une fonction de densité proposée dans (Luo, J. Z. Wang et Promislow, 2008) pour étendre les communautés locales non classées jusqu'à ce que le score de qualité dépasse un seuil donné.

3.1.4 Bilan des algorithmes de détection de communautés statiques

Dans cette section, nous présentons un aperçu critique des solutions existantes. Le tableau 3.1 présente quatre critères pour évaluer les algorithmes présentés précédemment.

Le premier critère indique l'approche utilisée. Trois sous-critères sont considérés pour évaluer ce critère, à savoir, le type d'approche, ses avantages et ses inconvénients. Le second critère permet de savoir quels types de réseaux sont supportés par l'algorithme. Nous avons considéré deux types de réseaux : les réseaux orientés et ceux pondérés. Le troisième critère donne un aperçu sur les types de communautés couvertes : chevauchantes et ego-centrées. Le dernier critère présente des informations relatives au comportement de l'algorithme. Deux sous-critères sont utilisés pour évaluer ce critère, à savoir, le déterminisme et la complexité. Le premier sous-critère met l'accent sur le comportement déterministe de l'algorithme. Autrement dit, est-ce que l'algorithme retourne, toujours, la même communauté pour le même nœud d'intérêt ? Ou produit-il, plutôt, une communauté différente lors de chaque exécution. Le second sous-critère donne une idée de la rapidité des algorithmes. Les termes utilisés dans les expressions des complexités sont :

- $\overline{d}$ désigne le degré moyen du graphe.
- $|V_{N_{\{u,1\}}}|$ représente le nombre de nœuds dans le voisinage direct du nœud d'intérêt u.
- $|E_{N_{\{u,1\}}}|$ indique le nombre de liens dans le voisinage direct de u.
- $|V_{C_{\{u,1\}}}|$ représente le nombre de nœuds dans la communauté ego-centrée en une seule étape de u.
- $|E_{C_{\{u,1\}}}|$ désigne le nombre de liens dans la communauté ego-centrée en une seule étape de u.
- T_{max} indique le nombre maximal d'itérations.

Les critères utilisés dans le tableau 3.1 relèvent d'une importance capitale car ils nous permettent de savoir les intrants des algorithmes existants (types de réseaux supportés), les extrants (natures de communautés détectées), les méthodologies (type d'approche, avantages et inconvénients) et enfin, les

Tableau 3.1 – Présentation des points forts et points faibles des algorithmes de détection de communautés statiques locales.

Références	Approche			Types de réseaux		Nature de communautés		Comportement de l'algorithme	
	Catégorie	Avantages	Inconvénients	Orienté	Pondéré	Communautés ego-centrées	Communautés chevauchantes	Déter-minisme	Complexité
(Clauset, 2005)	Fonction de qualité	- L'utilisation d'un seul indicateur pour évaluer la pertinence d'une communauté. - La possibilité de basculer entre les communautés locales et globales.	- L'implication d'un ordre de passage entre les nœuds. - L'optimisation est NP-Complet.	×	×	×	✓	×	$\mathcal{O}(\lvert V_{N_{\{u,1\}}}\rvert^2 \overline{d})$
(J. Chen, Zaiane et Goebel, 2009)				×	×	×	×	✓	$\mathcal{O}(\lvert E_{C_{\{u,1\}}}\rvert \lvert V_{N_{\{u,1\}}}\rvert \overline{d})$
(Lu, Wen et Cao, 2013)				×	✓	×	✓	×	$\mathcal{O}(\lvert V_{N_{\{u,1\}}}\rvert^2)$
(Xiang et al., 2016)				×	×	×	✓	×	$\mathcal{O}(\lvert V_{N_{\{u,1\}}}\rvert^2 \log(\lvert V_{N_{\{u,1\}}}\rvert))$
(Hamann, Röhrs et Wagner, 2017)				×	✓	✓	×	✓	$\lvert E_{C_{\{u,1\}}}\rvert \log(\lvert V_{N_{\{u,1\}}}\rvert)$
(Huang et al., 2011)	Mesures de similarité	- L'existence de corrélation entre les mesures de similarité et la structure en communautés. - Contrairement à l'approche de fonction de qualité, celle-ci est déterministe et stable.	- Les mesures existantes se focalisent soit sur l'orientation de liens, soit sur leur pondération. - Les mesures de similarité actuelles sont conçues pour les réseaux statiques.	×	✓	×	×	×	$\mathcal{O}(\lvert V_{N_{\{u,1\}}}\rvert \log(\lvert V_{N_{\{u,1\}}}\rvert))$
(Danisch, Guillaume et Le Grand, 2014b)				×	×	×	✓	✓	$\mathcal{O}(\lvert E_{N_{\{u,1\}}}\rvert)$
(Hollocou, Bonald et Lelarge, 2016)				×	✓	×	✓	×	Non mentionnée
(Ding, J. Zhang et Jing Yang, 2018)				✓	×	✓	✓	✓	$\mathcal{O}(\overline{d}^3 \log(\lvert V_{N_{\{u,1\}}}\rvert))$
(Tsourakakis et al., 2013)	Approche à base de triangles	- La détection de communautés chevauchantes.	- La majorité des algorithmes de cette approche souffrent d'une complexité exponentielle.	×	×	×	✓	×	$\mathcal{O}(T_{max} \lvert E_{N_{\{u,1\}}}\rvert)$
(Fanrong et al., 2014)				×	×	×	✓	✓	$\mathcal{O}(\lvert V_{C_{\{u,1\}}}\rvert^2)$
(Fagnan, Zaiane et Barbosa, 2014)				×	×	×	✓	✓	Non mentionnée
(Tsourakakis, 2015)				×	×	×	✓	×	$\mathcal{O}(\lvert V_{N_{\{u,1\}}}\rvert + \lvert E_{N_{\{u,1\}}}\rvert)$

informations relatives au comportement de l'algorithme (déterminisme et complexité). Ainsi, en fonction du niveau de couverture des critères, il devient plus aisé de voir les manquements de l'algorithme en question. En outre, pour chaque approche, nous présentons les travaux connexes suivant l'ordre chronologique pour montrer au lecteur comment le sujet de la détection de communautés locales a évolué au fil des années.

A la lecture du Tableau 3.1, nous remarquons que chaque approche présente des avantages et des inconvénients. Nous observons également qu'aucun algorithme ne couvre simultanément tous les critères définis.

Concernant l'approche *Fonction de qualité*, nous pouvons dire que la représentation de la qualité de communauté en une seule valeur est pratique et permet d'en avoir une idée via un seul indicateur. En outre, cette approche peut être utilisée pour détecter des communautés de portée locale et globale (Hamann, Röhrs et Wagner, 2017 ; T. Chen, Singh et Bassler, 2017). Toutefois, l'optimisation de la fonction de qualité implique un ordre de passage entre les nœuds. De ce fait, la communauté construite dépend fortement de cet ordre, selon lequel, chaque communauté sera forcément meilleure ou plus mauvaise que n'importe quelle autre. En outre, l'optimisation de la fonction de qualité constitue un problème NP-Complet (Leskovec, Lang et Mahoney, 2010) et aucune fonction proposée dans la littérature ne fait l'unanimité.

Pour ce qui est de l'approche *Mesures de similarité*, l'un des avantages des mesures de similarité est que les communautés détectées sont structurellement bonnes grâce à la corrélation existante entre les mesures de similarité et la structure en communautés. De plus, les communautés détectées sont locales (à un ou plusieurs nœuds) ce qui permet de les étiqueter et de suivre facilement leur évolution. En outre, contrairement à l'approche basée sur la fonction de qualité, l'approche basée sur les mesures de similarité est déterministe et stable. Elle est donc peu sensible aux perturbations. Concernant les limites de cette approche, les mesures de similarité actuelles sont conçues pour les réseaux statiques. L'aspect dynamique semble être ignoré dans les travaux existants. De plus, les algorithmes actuels de cette approche mettent l'accent soit sur la structure topologique, soit sur l'aspect de pondération. A notre connaissance, il n'existe pas encore de méthode combinant ces deux aspects.

Quant à *l'approche à base de triangles*, son avantage principal est la détection de communautés chevauchantes. Toutefois, les méthodes d'extraction de cliques maximales sont généralement coûteuses.

Pour le reste des critères, nous pouvons dire que l'orientation de liens représente l'un des aspects dont la gestion est quasiment absente dans la littérature. De plus, nous remarquons que la pondération des liens et le déterminisme sont partiellement pris en compte.

Concernant l'aspect ego-centré, il peut être vu de deux manières :

1. l'ego est focal par rapport à ses voisins. Ainsi, une communauté ego-centrée est identifiée à partir d'un nœud d'intérêt et elle est constituée de tous les nœuds qui peuvent être atteints à partir de l'ego à travers k sauts où la valeur de k varie entre 1 et l'excentricité [2] du nœud d'intérêt. Un exemple de cette définition est proposée dans (Lu, Wen et Cao, 2013) ;
2. l'ego est focal par rapport aux communautés. Dans ce cas, le résultat est une communauté multi-ego-centrée : un ensemble de communautés dont chacune comprend le nœud d'intérêt. Cette définition est adoptée dans (Danisch, Guillaume et Le Grand, 2014b).

2. L'excentricité désigne la distance géodésique maximale qui peut exister entre u et n'importe quel autre nœud d'un graphe.

Comme nous nous intéressons aux communautés ego-centrées et non pas à celles multi-ego-centrées, nous avons considéré que le critère « *communautés ego-centrées* » désigne les travaux où l'ego est focal par rapport à ses voisins. Toutefois, la quasi-totalité de ces travaux considèrent que le nœud d'intérêt ne représente pas un élément indispensable pour sa communauté. Il sert uniquement comme graine pour initialiser l'algorithme.

Un autre aspect totalement absent dans la littérature est la détection de communautés ego-centrées à voisinage élargi en k sauts où la valeur de k varie entre 1 et l'excentricité du nœud d'intérêt. En effet, les travaux existants se limitent au voisinage direct. Pourtant, la détection de communauté ego-centrées à voisinage élargi permet de mieux comprendre et analyser les relations qui existent entre le nœud d'intérêt et ses voisins directs ou indirects. Sachant que ces relations peuvent avoir un impact significatif sur non seulement le comportement du nœud d'intérêt mais aussi, sur sa communauté.

Enfin, les algorithmes que nous avons présentés dans cette section ont été conçus uniquement pour traiter des réseaux statiques. De ce fait, ils ne sont pas capables d'interpréter les changements que les communautés subissent au fil du temps dans un réseau dynamique. Nous présentons dans la section suivante quelques approches de suivi de l'évolution de communautés dynamiques.

3.2 Détection de communautés dynamiques

La détection de communautés dynamiques locales consiste à interpréter les changements, le cas échéant, de la structure des communautés du réseau au fil du temps. Le processus d'interprétation de communautés exige deux aspects :

1. la connaissance du moment clé de l'occurrence de changements significatifs sur la structure des communautés. La maîtrise de cette connaissance passe par la gestion de fenêtres temporelles qui sont considérées comme des périodes inclusives dans lesquelles les changements surviennent. En d'autres mots, chaque changement d'une communauté se déroule exclusivement à l'intérieur d'une période et jamais entre deux périodes.
2. l'élaboration d'un ensemble de règles définissant les types d'évolution possibles en fonction des changements au fil du temps.

La suite de cette section est structurée au regard de ces deux aspects. Pour ce faire, nous présentons quelques définitions dans la section 3.2.1 avant de décrire la problématique de la gestion de fenêtre temporelle dans la section 3.2.2. Ensuite, les principales approches de suivi de l'évolution de communautés dynamiques sont passées en revue dans la section 3.2.3. Enfin, nous concluons par un bilan critique dans la section 3.2.4.

3.2.1 Communauté dynamique et types d'évolution

D'après la définition 17, une communauté est dite dynamique si sa structure topologique [3] ou sa composition sémantique [4] change au fil du temps. Ce changement peut se manifester de plusieurs façons. L'un des premiers travaux ayant traité cette question est celui de (Palla, Barabási et Vicsek,

3. Le changement de la structure topologique désigne l'apparition ou la disparition des nœud/liens.
4. Le changement de la composition sémantique désigne la variation des valeurs des poids de liens ou bien le changement de leur orientation.

2007). Dans ce travail, les auteurs ont défini six transformations possibles de communautés dynamiques, à savoir, la naissance, la croissance, la contraction, la fusion, la division et la mort. Ces transformations, illustrées sur la Figure 3.4, peuvent être expliquées comme suit :

- la naissance : une composition de nœuds/liens forme une nouvelle communauté à l'instant actuel ;
- la croissance : la communauté grandit en intégrant de nouveaux nœuds/liens ;
- la fusion : plusieurs communautés se regroupent en une seule ;
- la division : la communauté se divise en plusieurs petites communautés ;
- la contraction : la communauté rétrécit en perdant quelques nœuds/liens ;
- la mort : la communauté existante à l'instant précédent, disparaît à l'instant actuel.

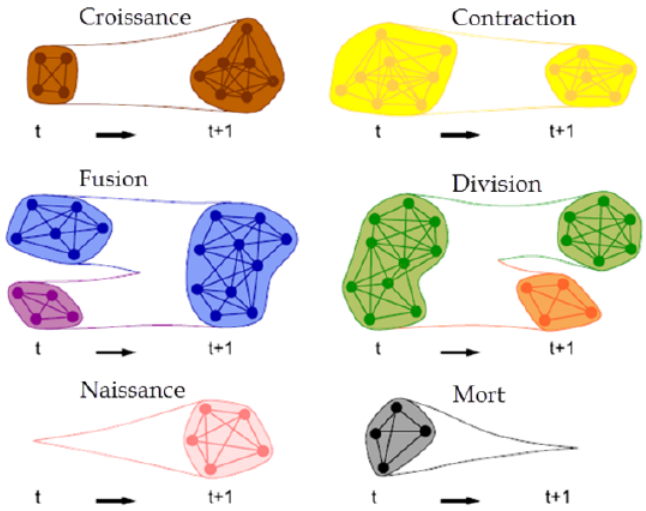


FIGURE 3.4 – Exemples de quelques transformations de communautés dynamiques. Figure extraite de (PALLA, BARABÁSI et VICSEK, 2007).

En effet, les transformations les plus faciles à identifier sont la naissance et la mort. Viennent ensuite la croissance et la contraction. En outre, les transformations de fusion et de division sont plus difficiles à détecter car elles dépendent de la définition adoptée par l'algorithme.

Des transformations similaires ont été également discutées dans (ASUR, PARTHASARATHY et UCAR, 2009 ; GREENE, DOYLE et CUNNINGHAM, 2010 ; BRÓDKA, SAGANOWSKI et KAZIENKO, 2013). Sauf que dans (BRÓDKA, SAGANOWSKI et KAZIENKO, 2013), les auteurs ont proposé une nouvelle transformation qu'ils appellent la « monotonie ». Elle signifie que la communauté garde la même composition de nœuds/liens d'un instantané à l'autre. En outre, dans (Rémy CAZABET, 2013), l'auteur a défini une nouvelle transformation qu'il appelle la « résurgence », au cours de laquelle la communauté vit un état d'extinction temporaire entre deux instantanés non successifs. Autrement dit, la communauté disparait à un moment donné, puis réapparait quelques temps plus tard sous une forme identique ou très proche. Un exemple de cette transformation se présente au niveau des communautés périodiques.

Malheureusement, il n'y a pas encore eu de consensus sur la façon d'implémenter ces transformations. Prenons un exemple simple, lors de la division d'une communauté X en deux, doit-on considérer que l'une des communautés résultantes reste la même communauté X, tandis que l'autre représente une nouvelle communauté distincte X_1 ? Ou doit-on, plutôt, considérer qu'une fois divisée, la communauté X n'existe plus et a cédé la place à deux nouvelles communautés X_1 et X_2 ? Ces questions

restent ouvertes et chaque algorithme définit ses propres hypothèses telles que nous allons le voir dans la suite.

3.2.2 Décomposition des réseaux dynamiques

L'analyse de l'évolution des réseaux dynamiques nécessite le choix d'un intervalle de temps, appelé souvent « fenêtre temporelle ». Les contours de cette fenêtre déterminent le délai à partir duquel des captures ou instantanés du réseau peuvent être relevés en vue d'une analyse de l'évolution des communautés. Cela montre l'importance que revêt la taille des fenêtres temporelles sur le résultat de l'analyse de l'évolution. Malheureusement, la taille des fenêtres temporelles n'est pas souvent étudiée en profondeur dans les travaux existants. En général, les auteurs se limitent à préciser l'intervalle de temps qu'ils jugent adéquat sans en fournir une démonstration rigoureuse. Bien qu'il n'existe pas beaucoup de travaux sur la détermination dynamique de la taille des fenêtres, il convient de noter la présence de plusieurs articles expliquant leur impact sur l'évolution des communautés au fil du temps.

Dans (Krings et al., 2012), l'objectif est d'étudier l'effet de l'intervalle de temps sur les caractéristiques structurelles des réseaux en utilisant un jeu de données issu des appels téléphoniques pour une période de six mois. Dans cette perspective, les auteurs se focalisent sur l'étude des fonctionnalités qui devraient avoir un impact sur les propriétés des réseaux agrégés si la taille de la fenêtre temporelle grandit.Ainsi, les auteurs examinent la croissance du réseau agrégé en termes du nombre de nœuds/liens et du degré moyen. Selon les observations, le nombre de nœuds accroit rapidement au début du processus d'agrégation, de sorte que la fenêtre de temps contient environ 90% des nœuds si le réseau est agrégé mensuellement. En revanche, la croissance du nombre de liens nécessite beaucoup plus de temps, une période d'environ cinq mois est nécessaire pour pouvoir capturer 90% des liens du réseau. En outre, contrairement au nombre de nœuds, pour les intervalles longs d'agrégation, le nombre de liens continue de croître de façon constante et aucune borne de croissance n'est observée. De même, pour des intervalles d'agrégation assez longs, le degré moyen continue d'accroitre en fonction de la longueur de l'intervalle de temps.

Après avoir mené plusieurs expériences sur l'impact du choix de l'intervalle de temps, les auteurs concluent que les intervalles d'agrégation courts produisent des instantanés où dominent les liens forts associés aux communautés denses, tandis que pour les intervalles longs, les effets de liens faibles deviennent de plus en plus forts.

Dans (Psorakis et al., 2012), les auteurs s'intéressent à la recherche d'une structure communautaire dans des jeux de données spatiotemporels construits à partir des déplacements des oiseaux. La méthode proposée consiste à parcourir, à travers des capteurs, les arrivées des oiseaux aux endroits précis au cours d'une période donnée. Ensuite, la méthode considère que les régions denses d'activité correspondent aux événements de rassemblement des oiseaux.

En résumé, la plupart des travaux existants considèrent une période fixe de la fenêtre temporelle. Cette taille peut prendre plusieurs échelles. Par exemple, un jour (Génois et al., 2015), un mois (Psorakis et al., 2012) ou encore différentes échelles de temps (Krings et al., 2012 ; Holme, 2015 ; Sekara, Stopczynski et Lehmann, 2016).

L'inconvénient principal des tailles statiques ou fixes est qu'il est probable que les communautés détectées durant la fenêtre $t+1$ ne soient pas si différentes de celles de la fenêtre t. Pour illustrer cette faiblesse, supposons un réseau de proximité géographique, composé de chercheurs scientifiques qui se

rencontrent moult fois dans l'année. Supposons une année d'activité durant laquelle les chercheurs se sont rencontrés quatre fois aux moments suivants : lors de la première semaine de l'année, après un mois, après trois mois et après six mois. Le choix d'une fenêtre fixe d'un mois implique une analyse non fructueuse pour au moins 8 fois. Une période de 6 mois ne donne pas un aperçu de l'évolution durant les 6 premiers mois et l'analyse faite au delà de 6 mois n'apporte rien. Ainsi, nous perdons du temps et des ressources en cherchant de nouvelles communautés alors que le réseau, en effet, n'a pas changé. De cette petite illustration, l'on peut observer qu'un petit intervalle de temps peut conduire à des analyses ou captures répétitives inutiles. Un grand intervalle, quant à lui, traduit peu fidèlement l'évolution ou les grandes étapes de celle-ci. C'est pourquoi, il est important de penser à une méthode dynamique pouvant donner plus de chance de faire des captures ou analyses que lors des moments indispensables.

En effet, la méthode dynamique subdivise le temps en plusieurs fenêtres où la taille de chacune peut, éventuellement, être différente des autres. Par conséquent, la méthode dynamique vise le choix d'une taille « optimale » en s'appuyant sur la quantité de changements intervenus au fil du temps. Ainsi, la taille d'une fenêtre varie selon la vélocité des changements.

3.2.3 Approches de suivi de l'évolution de communautés dynamiques

Une fois que les fenêtres sont définies et/ou que les captures sont faites, le processus de suivi consiste à analyser la structure des communautés pour chaque capture et d'en déduire une conclusion. La suite de cette section est rédigée pour présenter le principe des approches utilisées dans le suivi de l'évolution de communautés dynamiques.

3.2.3.1 Approche par instantanés uniformes successifs

Comme la détection de communautés statiques semble arrivée à maturité, la quasi-totalité des chercheurs ont eu l'idée de réutiliser les concepts des solutions statiques dans le cas dynamique. Ce qui a fait l'objet de plusieurs tentatives d'adaptation des algorithmes statiques aux réseaux dynamiques (Hopcroft et al., 2004 ; Palla, Barabási et Vicsek, 2007 ; Y. Wang, Wu et Du, 2008 ; Rosvall et Bergstrom, 2010 ; W. Chen et al., 2010 ; Greene, Doyle et Cunningham, 2010). L'idée générale est de considérer un réseau dynamique comme une succession de réseaux statiques, appelés « instantanés », dont chacun représente l'image du réseau dynamique à un instant donné. Le principe de cette approche s'articule donc autour de trois étapes. Tout d'abord, décomposer l'évolution du réseau en plusieurs instantanés. Ensuite, appliquer un algorithme statique à chaque instantané. Enfin, faire correspondre les communautés trouvées dans un instantané avec celles de l'instantané précédent. La Figure 3.5 montre trois instantanés d'un réseau dynamique avec une association entre les communautés des différentes étapes. L'instantané t contient uniquement deux communautés colorées, respectivement, en bleu et en vert. La communauté colorée en bleu à l'instantané t se divise à $t+1$ en deux sous-communautés, colorées en bleu et en rouge, tandis que la communauté de couleur verte reste intacte. A l'instantané $t+2$, la communauté de couleur bleu reste la même, celle en rouge et celle en vert se rétrécissent et une nouvelle communauté apparait, de couleur bleu clair.

En effet, nous pouvons dire qu'en termes de discrétisation/continuité du temps, il n'existe que deux grandes approches, celle d'instantanés et celle dynamique. Les autres méthodes reprennent l'idée de l'approche par instantanés en ajoutant ou modifiant quelques notions pour réduire l'instabilité des

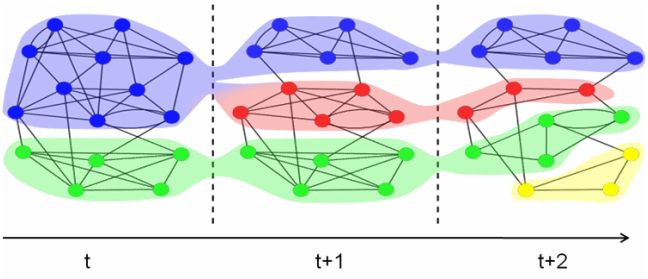


FIGURE 3.5 – Illustration de l'approche par instantanés uniformes successifs. Figure extraite de (AYNAUD, FLEURY et al., 2013).

algorithmes statiques. Dans les paragraphes suivants, nous abordons deux approches qui se basent sur le même principe d'instantanés mais l'interprètent différemment.

3.2.3.2 Approche par instantanés consécutifs, deux à deux interdépendants

Cette approche s'inspire de la précédente (CHAN, HUI et Kuang XU, 2009 ; LIN et al., 2009 ; LANCICHINETTI, FORTUNATO et KERTÉSZ, 2009 ; K. S. XU, KLIGER et HERO III, 2011). Toutefois, dans la perspective de surmonter la problématique d'instabilité des algorithmes statiques, les résultats obtenus, à l'instantané t, sont pris en compte à l'instantané $t+1$ en vérifiant si les communautés obtenues sont similaires ou en cohérence avec celles obtenues dans les fenêtres antérieures. Cette stratégie permet de prendre les bonnes décisions au regard des observations antérieures. La différence entre l'approche précédente et celle-ci se manifeste dans la phase d'interprétation de communautés : l'approche par instantanés uniformes successifs effectue la détection de communautés sur chaque instantané indépendamment des autres fenêtres de temps. Alors que l'approche par instantanés consécutifs profite de communautés détectées à l'instantané t pour mieux identifier celles que l'on devraient avoir à l'instantané $t+1$.

3.2.3.3 Approche par instantanés multi-pas

Analyser chaque instantané séparément pose un problème à cause de l'instabilité des algorithmes statiques. Une extension de l'approche par instantanés est de ne pas se focaliser sur la stabilité temporaire du partitionnement mais plutôt sur la qualité globale du découpage à long terme (TANTIPATHANANANDH, BERGER-WOLF et KEMPE, 2007 ; JDIDIA, ROBARDET et FLEURY, 2007 AYNAUD et GUILLAUME, 2010 ; MUCHA et al., 2010 ; T. YANG et al., 2011). Ainsi, on n'impose pas à une communauté, à l'instantané $t+1$, d'être proche de la précédente mais d'être pertinente à l'instantané $t+1$. Cela entraîne, implicitement, la stabilité puisqu'il s'agit de chercher des communautés cohérentes sur plusieurs fenêtres de temps, consécutives ou pas, et non pas seulement sur deux instantanés successifs. Sur la Figure 3.6, l'image A présente trois instantanés d'un réseau dynamique, l'image B révèle les communautés sur chaque instantané et l'image C représente les communautés finales.

Cette approche est pertinente dans le cas où nous nous intéressons à la détection de communautés à long terme. Au lieu de chercher des communautés à chaque instant, il convient de chercher celles qui sont plutôt bonnes sur une longue durée. Un réseau de propagation des maladies, où la contagion se déroule au cours d'une longue durée en est une illustration.

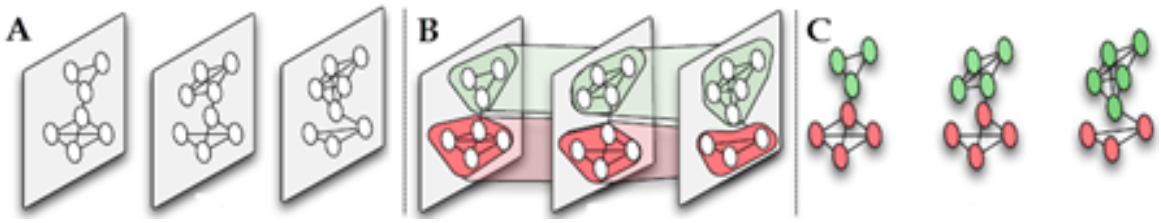


FIGURE 3.6 – Illustration de l'approche par instantanés multi-pas. Figure extraite de (Rémy CAZABET, 2013) avec une légère modification.

3.2.3.4 Approche incrémentale

Cette approche travaille directement sur le réseau dynamique, de ce fait, elle ne considère pas le réseau comme une succession d'instantanés, mais plutôt, comme une suite de modifications sur le même réseau (FALKOWSKI, BARTH et SPILIOPOULOU, 2008 ; Rémy CAZABET, Frédéric AMBLARD et HANACHI, 2010 ; Remy CAZABET et Frederic AMBLARD, 2011 ; J. LI et al., 2012 ; SHANG et al., 2014).

L'approche incrémentale vise donc à analyser un flux de données au lieu d'un instantané. L'entrée est, donc, une séquence d'événements sur le réseau et l'algorithme essaie de maintenir une décomposition de qualité en mettant à jour sa composition courante plutôt qu'en recalculant une composition à partir de zéro. Ceci est particulièrement utile dans le cas d'une analyse en temps réel.

En effet, si les mises à jour sont gérées efficacement, l'approche incrémentale aura bien des performances meilleures qu'un calcul de communautés à chaque intervalle de temps indépendamment. La manière dont les mises à jour sont faites, influe fortement sur les résultats obtenus. Nous discutons dans le paragraphe suivant les avantages et les inconvénients de l'approche incrémentale par rapport à l'approche par instantanés.

3.2.4 Bilan des approches de détection de communautés dynamiques

L'avantage certain de l'utilisation de l'approche par instantanés est de mettre à disposition énormément d'outils, sur lesquels beaucoup de chercheurs travaillent depuis longtemps. En revanche, cette approche souffre d'une faiblesse intrinsèque : sur deux graphes très proches topologiquement, le même algorithme peut trouver des communautés fortement différentes. Ainsi, les modifications obtenues ne sont donc pas liées aux changements structurels mais plutôt au caractère non-déterministe de l'algorithme employé.

De plus, l'utilisation de l'approche par instantanés entraîne la manipulation de plusieurs instantanés. Donc, si le réseau est de grande taille, le stockage des instantanés peut présenter un problème, vu leur volume considérable. Cependant, si nous parvenons à stocker l'intégralité des instantanés du réseau, nous aurons l'avantage d'avoir l'historique de détection de communautés aux différents moments, ce qui n'est pas le cas pour l'approche incrémentale.

Concernant la gestion des instantanés, la définition formelle de fenêtre de temps joue un facteur indispensable dans la détection de communautés. Une fenêtre de temps peut être considérée comme l'image statique du réseau à un instant donné et uniquement à cet instant. Dans ce cas, il est alors possible de perdre beaucoup de données importantes comme les modifications topologiques qui se déclenchent et se terminent dans une période entre deux instants successifs. Une autre définition plus réaliste de fenêtre de temps consiste à considérer que l'instantané actuel reprend les modifications de

là où l'instantané précédent s'est arrêté. Une telle définition a l'avantage de garder une vision contiguë de l'ensemble de modifications que le réseau subit dans le temps.

En somme, que ce soit l'approche par instantanés ou l'approche incrémentale, la question primordiale reste la même : à quel moment faut-il prélever les instantanés ou appliquer les mises à jour. Ce qui devrait pousser à étudier la fréquence optimale pour répertorier les changements. Hélas, cet aspect n'est pas encore bien étudié au moment où on écrit ce chapitre.

Tableau 3.2 – Critique des approches de détection de communautés dynamiques.

Approche	Avantages	Inconvénients
Approche par instantanés uniformes successifs	– la disponibilité de plusieurs outils et de techniques prêts à l'emploi ; – la possibilité de parallélisation du découpage du réseau pour accélérer le temps d'exécution.	– l'instabilité ; – le volume croissant des instantanés si le réseau est de grande taille et les instantanés sont nombreux.
Approche par instantanés consécutifs, deux à deux interdépendants	Réduction de l'instabilité.	Coût élevé.
Approche par instantanés multi-pasù	Réduction de l'instabilité en cherchant des communautés cohérentes sur plusieurs fenêtres de temps.	– la complexité très élevée notamment si les instantanés sont nombreux ; – aucun algorithme de cette approche ne permet de détecter efficacement les transformations de fusion et de division.
Approche incrémentale	– l'assurance, à court terme, de la stabilité de communautés ; – la bonne complexité : si deux instantanés consécutifs sont topologiquement proches, la mise à jour de communautés s'effectue dans un délai très court.	Difficulté d'assurer une cohérence de communautés sur l'ensemble des étapes d'évolution, car le suivi de communautés se fait uniquement lors du passage d'un instantané à l'autre.

Le Tableau 3.2 résume les avantages et les inconvénients des approches de détection/suivi de communautés dans les réseaux dynamiques.

3.3 Conclusion

Dans ce chapitre, nous avons passé en revue les principales solutions de détection des communautés locales statiques et dynamiques. L'étude de l'existant nous a permis de dégager cinq manquements dont la prise en compte permettra de mieux faire face au suivi de communautés ego-centrées dynamiques issues des applications mobiles sociales.

Premièrement, les algorithmes de détection de communautés se focalisent généralement sur les caractéristique topologiques du réseau en étudiant, par exemple, les mesures de centralité ou les distances entre les nœuds. Or, cette modélisation néglige l'intensité de communication entre les éléments du réseau. De ce fait, ces algorithmes ne sont pas capables de distinguer deux nœuds ayant le même nombre de voisins mais l'un d'eux, contrairement à l'autre, possède une fréquence d'échanges élevée à l'égard des nœuds du réseau.

Deuxièmement, une bonne partie des algorithmes existants ne prennent pas en compte l'orientation de liens. Par conséquent, ils ne peuvent pas saisir la réalité de communications entre les nœuds. Par exemple, si nous prenons le cas d'un réseau d'appels téléphoniques, en considérant que les liens sont bidirectionnels, ils ne peuvent pas distinguer les appels reçus de ceux émis. Ainsi, il est clair que la

prise en compte de la direction de liens est indispensable pour pouvoir différencier les communications entrantes de celles sortantes.

Troisièmement, l'aspect ego-centré au-delà du voisinage direct constitue l'un des manquements de l'existant. En effet, les travaux actuels s'intéressent, généralement, au partitionnement des réseaux en plusieurs communautés globales. Concernant les quelques travaux traitant les communautés locales, ils se limitent toujours au voisinage direct. De ce fait, ils ne sont pas aptes à comprendre et à analyser les relations qui existent entre le nœud d'intérêt et ses voisins indirects. Pourtant, ces relations peuvent avoir un impact significatif non seulement sur le comportement du nœud d'intérêt, mais aussi sur sa communauté. De plus, l'aspect évolutif semble ignoré quand il s'agit de communautés ego-centrées.

Quatrièmement, le processus de suivi de communautés dynamiques dépend fortement des types d'évolution définis. En effet, les types d'évolution existants souffrent de deux faiblesses, à savoir :

1. Elles négligent l'aspect de l'intensité de communications et se limitent aux changements topologiques ;
2. Elles sont de portée globale, c'est-à-dire qu'elles ne sont pas censées interpréter les changements microscopiques du réseau (relatifs aux nœuds/liens).

Cinquièmement, comme nous l'avons vu dans la section 3.2.2, il n'existe pas encore une méthode de gestion de la taille de fenêtre temporelle. Pourtant, cette dernière influe considérablement sur les communautés détectées vu qu'elle décrit l'ensemble de nœuds/liens qui appartient à chaque instantané. Ce manquement nous a poussé à proposer une nouvelle méthode capable de décomposer l'évolution d'un réseau dynamique en plusieurs instantanés tout en prenant en compte l'intensité de communication et l'orientation de liens.

Après avoir identifié ces cinq manquements, notre objectif est de proposer une méthode de détection et de suivi de l'évolution de communautés ego-centrées dans les réseaux orientés et pondérés. Dans le prochain chapitre, nous décrivons la philosophie générale de notre solution.

Deuxième partie

Contributions

Chapitre 4

Détection de communautés égo-centrées statiques

Pour rappel, nous nous intéressons au suivi de l'évolution de communautés ego-centrées dans les réseaux orientés et pondérés. Notre méthode consiste à construire des communautés ego-centrées autour de quelques nœuds d'intérêt. Ensuite, elle cherche à suivre les changements de ces communautés au fil du temps. Comme présenté dans l'introduction, notre solution est structurée en trois étapes, à savoir :

1. faire plusieurs captures du réseau dynamique à des instants précis. Chaque capture, appelée aussi instantané, est un réseau statique. L'agrégation de l'ensemble des instantanés pris jusqu'à l'instant t constitue le réseau agrégé à cet instant ;
2. appliquer un algorithme de détection de communautés statiques sur chaque instantané en vue de découvrir des communautés égo-centrées ;
3. prendre la structure d'une communauté égo-centrée sur deux instantanés successifs et vérifier s'il y a eu des changements majeurs. Selon la nature du changement, l'algorithme procède à une interprétation correspondant à une caractérisation de l'évolution de la communauté.

Dans la perspective de gérer les différentes étapes de notre solution, nous avons conçu une architecture structurée autour de trois grands modules comme illustré sur la figure 4.1.

Le premier module est le plus bas, il comprend l'ensemble des logiciels et bibliothèques permettant d'implémenter les modules au-dessus. Le second module est constitué de deux sous-modules. Le premier permet d'extraire les données d'une application sociale et de représenter ces données sous forme d'un réseau social. Le second sous-module est chargé de la décomposition du réseau en plusieurs instantanés. Le deuxième module est responsable de tout ce qui concerne la gestion de communautés. Il comprend deux sous-modules, l'objectif du premier est de détecter les communautés ego-centrées statiques sur l'ensemble des instantanés produits par le module précédent. Le second sous-module sert à comparer les communautés trouvées sur les instantanés afin d'identifier les types d'évolution de communautés au fil du temps.

Après ce bref rappel et aperçu global de notre contribution, l'objectif principal de ce chapitre est de décrire notre solution de détection de communautés statiques. Cette partie est d'une importance capitale car elle sert de soubassement à notre solution de suivi des communautés qui sera présentée au prochain chapitre. Au demeurant, nous décrivons d'abord notre manière de structurer un réseau social sur la base des applications cibles. Puis, nous détaillons notre stratégie de détection de communautés statiques.

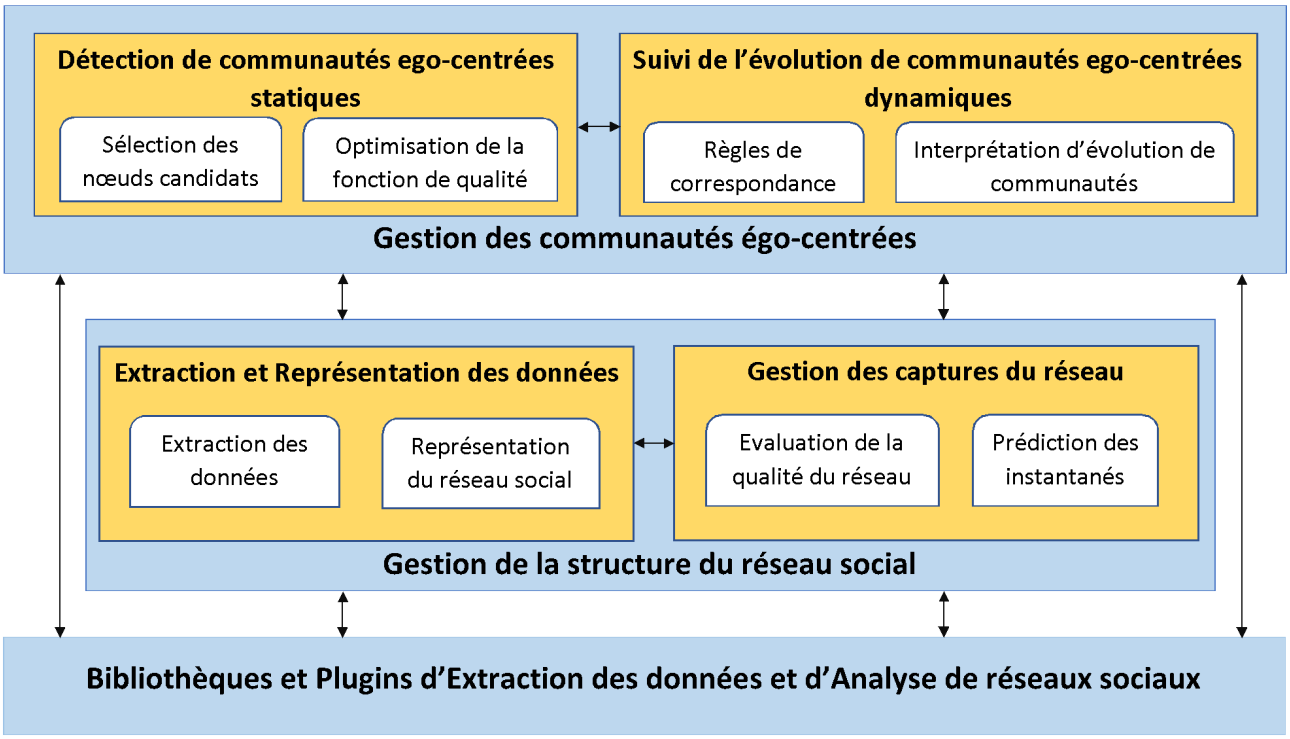


FIGURE 4.1 – Vue simplifiée des modules qui constituent l'architecture globale de notre solution.

4.1 Gestion de la structure du réseau

Nous commençons cette section par la manière dont nous représentons un réseau social à partir d'un jeu de données. Il convient de souligner que cette représentation est très importante car elle constitue le soubassement de la conception de notre solution.

4.1.1 Construction du réseau social

Nous nous plaçons dans le contexte des applications sociales mobiles permettant les interactions simultanées entre utilisateurs. Par exemple, *Facebook, Whatsapp, Twitter*, entre autres, font partie de ces applications qui offrent la possibilité d'échanger par du texte, des images, des vidéos de manière synchrone ou non. Deux utilisateurs peuvent communiquer via un appel vidéo (synchrone) ou des vidéos envoyées sous forme de messages que chacun peut visualiser de manière asynchrone. De même, les utilisateurs peuvent commenter aussi les objets d'un utilisateur (son statut, ses photos, ses vidéos, etc.). Cette manière de commenter les objets des autres est aussi une sorte de communication directe ou indirecte entre utilisateurs. Par conséquent, en prenant un jeu de données qui matérialise la communication entre les utilisateurs des applications sociales, nous extrayons un réseau social dont les nœuds sont les utilisateurs et les liens leurs interactions.

En effet, nous considérons l'existence d'un lien entre deux nœuds dès que ceux-ci échangent par l'un des canaux décrits précédemment, à savoir, des appels, des messages, des commentaires, des partages, etc. Pour refléter une sémantique au sein de la classe des utilisateurs, nous avons introduit l'orientation des liens pour distinguer l'émetteur du récepteur. Autrement dit, la direction du lien matérialise le nœud qui a émis et celui qui a reçu une communication. De plus, si un nœud soumet plusieurs communications à un autre, le poids du lien est incrémenté pour caractériser le nombre de

fois que les deux concernés ont échangé. Si à l'instant $t+1$, l'un des nœuds du réseau communique avec un nœud qui n'existait pas à l'instant t, le nouveau nœud sera ajouté dans le réseau muni du lien correspondant. Une nouvelle interaction entre deux nœuds déjà existants dans le réseau implique l'apparition d'un nouveau lien. C'est ainsi que le réseau évolue dans le temps.

Pour un meilleur suivi du réseau dynamique, nous avons ajouté des attributs de nœuds et de liens.

Attributs de nœuds. Chaque nœud est caractérisé par les informations suivantes :

- un identifiant unique : cet attribut permet d'identifier chaque nœud de manière univoque ;
- une date d'apparition : cet attribut indique l'intervalle de vie d'un nœud au sein du réseau.

Attributs de liens. Chaque lien est muni des informations suivantes :

- sa direction, qui montre le nœud initiant l'interaction et le nœud le recevant ;
- son poids, qui indique l'intensité de l'interaction entre les nœuds durant une période donnée ;
- sa durée, qui représente le temps que dure chaque interaction entre deux utilisateurs. Pour les communications de type appel téléphonique ou vidéo, la durée correspond au temps d'échanges. Par contre, s'il s'agit d'une interaction instantanée, comme le cas d'échange par textes, nous supposons l'interaction continue tant que le temps séparant les envois et les réponses ne dépasse pas un seuil. Autrement dit, si le temps entre un envoi et une réponse est au delà de ce seuil, nous considérons que la première communication est achevée et qu'une deuxième a été initialisée. L'intuition derrière cette hypothèse est bien adaptée à la philosophie de la messagerie instantanée où les individus ont tendance à envoyer des messages textes en temps réel de sorte que les utilisateurs reçoivent leur réponse instantanément.

4.1.2 Exemple de construction de réseau

Nous nous appuyons sur Facebook en considérant les commentaires autour d'une discussion. Nous supposons deux types de commentaires : 1) un utilisateur commentant la publication de quelqu'un ; 2) un autre utilisateur réagissant au premier commentaire. Ainsi, le réseau est construit à partir des commentaires des utilisateurs. La figure 4.2 montre une illustration d'une discussion instantanée d'un groupe sur Facebook. Pour des raisons de confidentialité, nous avons caché les messages. A partir de cette discussion, un graphe social est extrait comme représenté sur la figure 4.3. Le tableau 4.1 présente les identifiants des utilisateurs. Ces identifiants sont utilisés comme labels des nœuds du graphe.

4.2 Détection de communautés statiques ego-centrées

Nous avons décrit dans la section précédente la manière de représenter un réseau à partir de données d'applications sociales. A présent, nous présentons notre solution de détection de communautés dans un tel réseau. Nous rappelons que nous nous intéressons aux communautés égo-centrées. C'est pourquoi notre algorithme de détection de communautés statiques reçoit en entrée un nœud d'intérêt et retourne la communauté ego-centrée correspondante. Pour ce faire, l'algorithme crée tout d'abord la graine d'initialisation qui représente l'état initial de la communauté. Cette graine est constituée de l'ego et de quelques-uns de ses voisins. Ensuite, l'algorithme parcourt les voisins restants du nœud d'intérêt

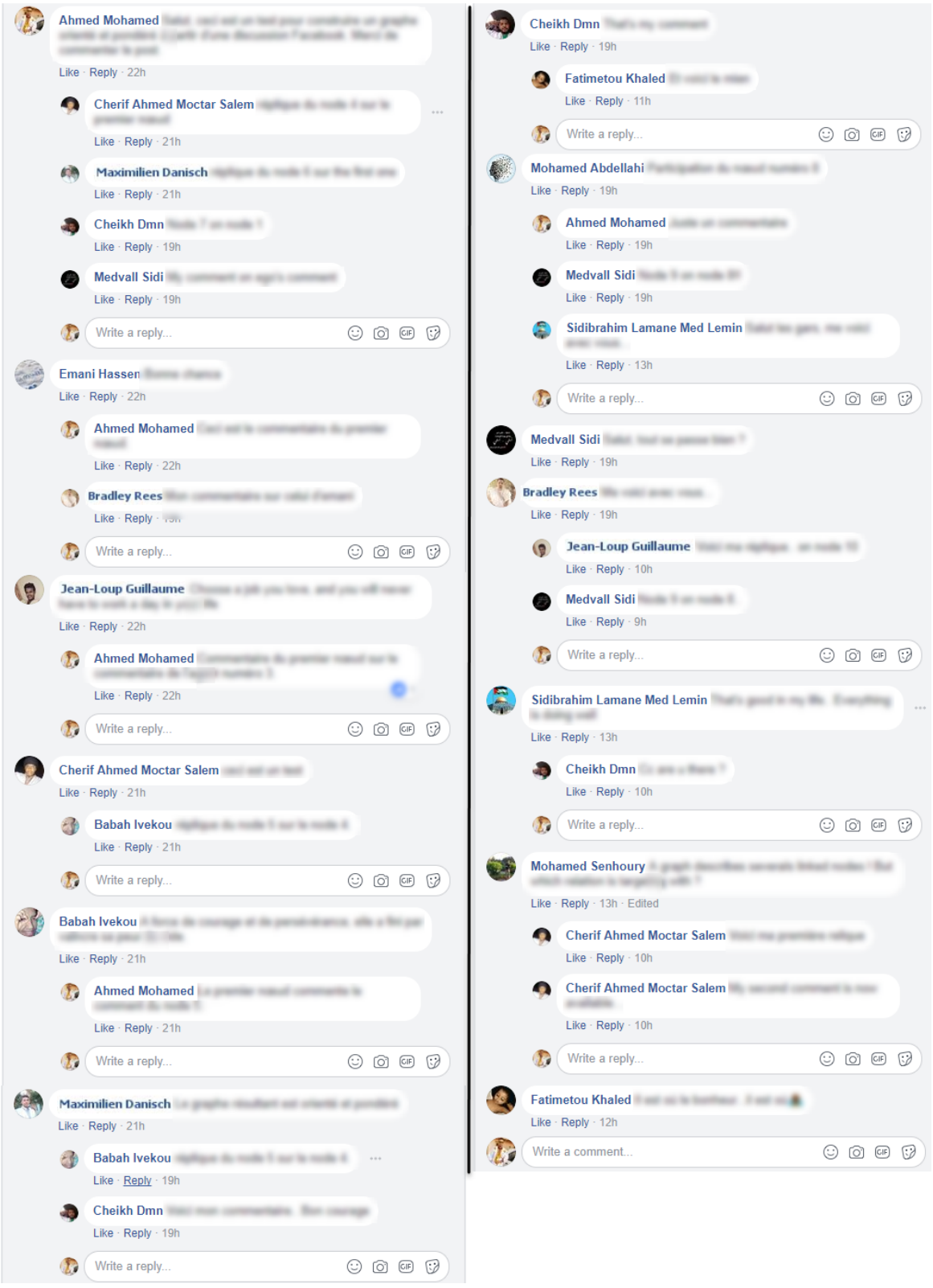


FIGURE 4.2 – Capture d'écran d'une discussion instantanée dans un groupe fermé sur Facebook.

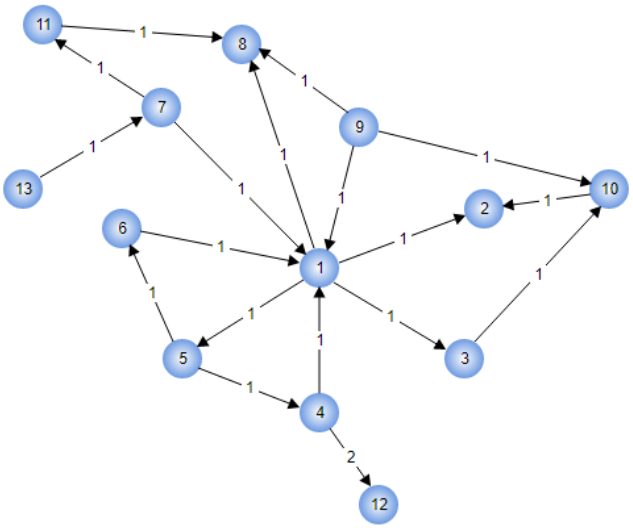


FIGURE 4.3 – Illustration du graphe social extrait de la discussion présentée sur la figure 4.2.

Tableau 4.1 – Identifiants des utilisateurs.

Nom d'utilisateur	**Identifiant**
Ahmed Mohamed	1
Emani Hassen	2
Jean-Loup Guillaume	3
Cherif Ahmed Moctar Salem	4
Babah Ivekou	5
Maximilien Danisch	6
Cheikh Dmn	7
Mohamede Abdellahi	8
MedVall Sidi	9
Bradley Rees	10
SidiBrahim Lamane Med Lemin	11
Mohamed Senhoury	12
Fatimetou Khaled	13

et ajoute ceux qui optimisent le score de qualité. Ce score est calculé par une fonction de qualité que nous allons détailler dans la suite.

En effet, les problèmes adressés par notre solution peuvent être résumés autour de trois points : 1) comment identifier et/ou sélectionner les nœuds candidats à l'initialisation ou ceux à ajouter/supprimer pour avoir une bonne communauté ; 2) comment évaluer la pertinence d'une communauté ; et enfin 3) comment construire la communauté égo-centrée. Nous allons détailler notre contribution par rapport à ces trois aspects.

4.2.1 Identification des nœuds candidats

L'identification des nœuds peut être faite de manière aléatoire ou déterministe. Soit u un nœud d'intérêt, $N_{\{u,k\}}$ son voisinage de longueur k et $C_{\{u,k\}}$ sa communauté ego-centrée de longueur k.

4.2.1.1 Méthode aléatoire

Il s'agit de sélectionner aléatoirement un nœud parmi les voisins de l'ego. L'avantage de cette méthode est sa rapidité puisqu'elle ne nécessite aucun calcul préalable. De plus, la sélection aléatoire des voisins d'un nœud d'intérêt u est d'une complexité linéaire égale à $|V_{N_{\{u,k\}}}| - 1$ où $N_{\{u,k\}}$ désigne le voisinage en k étapes du nœud d'intérêt u et $|V_{N_{\{u,k\}}}|$ le nombre de voisins dans ce voisinage. Nous rappelons que le voisinage du nœud u peut être de longueur 1, *i.e.*, l'ensemble des nœuds connexes à u, ou de longueur k, *i.e.*, l'ensemble des nœuds distants de u de k sauts au plus. Toutefois, avec cette méthode, on peut choisir un nœud impertinent qui ne favorise pas une construction efficace de la communauté. De plus, la méthode aléatoire ne garantit pas une reproductibilité des résultats de l'exécution successive de l'algorithme. En effet, comme le choix est aléatoire, d'une exécution à une autre la graine peut changer ainsi que la forme de la communauté. Pour pallier cette insuffisance, nous avons proposé une méthode déterministe.

4.2.1.2 Méthode déterministe

Pour faire une sélection déterministe, nous avons défini une métrique, appelée « degré d'affinité », qui évalue le niveau d'appartenance ou de proximité d'un nœud du voisinage à une ego-communauté ou à l'égo lui même. L'intuition derrière cette approche est qu'un nœud ayant un score élevé d'affinité est plus susceptible de participer à la communauté.

En effet, le degré d'affinité d'un nœud $v \in N_{\{u,k\}}$ à une communauté $C_{\{u,k\}}$ est mesuré suivant deux aspects, à savoir :

- La séparation du reste du réseau : le nœud v possède-t-il plus de liens au sein de la communauté qu'en dehors de celle-ci ? Autrement dit, communique-t-il plus avec les nœuds de la communauté qu'avec ceux qui se trouvent à l'extérieur ?
- Le niveau de connectivité à la communauté : à quel degré v est-il connecté aux nœuds de la communauté ? Communique-t-il assez avec eux ?

Notons que pour chaque aspect, nous prenons en compte le nombre de liens et l'intensité de leur poids. Pour gérer le premier aspect, nous proposons la métrique $Sep._v(C_{\{u,k\}})$ qui mesure la séparation

d'un nœud v par rapport à une communauté $C_{\{u,k\}}$:

$$Sep._v(C_{\{u,k\}}) = \frac{d_v^{C_{\{u,k\}}}}{d(v)} \times \frac{wd_v^{C_{\{u,k\}}}}{wd(v)} \tag{4.1}$$

avec,

- $|N_{\{v,k\}} \cap C_{\{u,k\}}|$ désigne le nombre de voisins que v possède dans $C_{\{u,k\}}$;
- $d(v)$ représente le nombre total des voisins de v ;
- $wd_v^{C_{\{u,k\}}}$ indique la somme des poids des liens dont l'une extrémité est v et l'autre appartient à $C_{\{u,k\}}$;
- $wd(v)$ est la somme des poids de tous les liens auxquels le nœud v est adjacent.

La valeur de $Sep._v(C_{\{u,k\}})$ varie entre 0 et 1. Dans le pire des cas, la valeur de $Sep._v(C_{\{u,k\}})$ sera proche de zéro, ce qui signifie que v possède beaucoup de voisins dans le graphe et un seul voisin dans $C_{\{u,k\}}$ avec un poids de lien égal à 1. En effet, vu que les poids modélisent l'intensité de communication entre les nœuds, nous considérons qu'ils sont des entiers strictement positifs. Un poids nul désigne qu'il n'y a pas de communication entre les nœuds. Dans le meilleur des cas, $Sep._v(C_{\{u,k\}}) = 1$, ce qui signifie que tous les voisins de v se trouvent dans $C_{\{u,k\}}$).

En effet, la faiblesse de la métrique $Sep.$ se manifeste lorsque certains nœuds n'ont pas de liens à l'extérieur de la communauté. Dans un tel cas, cette métrique considère que tous ces nœuds sont au même niveau de pertinence. Ce qui n'est pas toujours forcément le cas. C'est pourquoi nous proposons une seconde métrique dont le but est d'évaluer le degré de connectivité d'un nœud v par rapport aux nœuds de la communauté $C_{\{u,k\}}$:

$$Conn._v(C_{\{u,k\}}) = \frac{1}{d_v^{C_{\{u,k\}}} \times wd_v^{C_{\{u,k\}}}} \tag{4.2}$$

L'intuition derrière est que plus le nœud v est très connecté [1] à $C_{\{u,k\}}$, plus le score de $Conn._v$ est proche de zéro.

La valeur de $Sep._v(C_{\{u,k\}})$ varie entre 0 et 1. Au pire des cas, $Conn._v(C_{\{u,k\}}) = 1$, ce qui correspond à un nœud ayant un seul lien dans $C_{\{u,k\}}$ et dont le poids est égal à 1. Dans le meilleur des cas, la valeur de $Conn._v(C_{\{u,k\}}) \simeq 0$, ce qui veut dire que v possède un grand nombre de voisins dans $C_{\{u,k\}}$ avec des poids de liens très élevés.

Sur la base des métriques précédentes, nous définissons le degré d'affinité comme suit :

$$\mathcal{A}._v(C_{\{u,k\}}) = Sep._v(C_{\{u,k\}}) + (1 - Conn._v(C_{\{u,k\}})) \tag{4.3}$$

Dans la formule ci-dessus, nous calculons $1 - Conn._v(C_{\{u,k\}})$ pour que les bornes de $Sep.$ et $Conn.$ soient cohérentes. La valeur de $\mathcal{A}._v(C_{\{u,k\}})$ varie entre 0 et 2.

En effet, l'avantage de cette métrique est qu'elle définit un ordre de passage des nœuds suivant leur niveau d'appartenance à la communauté. Ainsi, le nœud ayant le degré d'affinité le plus élevé est sélectionné en premier lieu. Si jamais deux nœuds ont le même degré d'affinité, nous nous référons à leur identifiant pour les ordonner. Cependant, cette métrique nécessite du temps pour faire les calculs

1. Il possède beaucoup de voisins dans $C_{\{u,k\}}$ et communique autant avec eux.

vu qu'elle doit estimer, à chaque itération, le degré d'affinité des nœuds. La sélection d'un voisin de u est d'une complexité de $2(n + |V_{N_{\{u,k\}}}|)$ tel que n désigne le nombre de nœuds du réseau.

4.2.2 Évaluation de la pertinence de communauté

Notre méthode d'évaluation de la pertinence de communauté est basée sur le principe d'optimisation d'une fonction de qualité. Cette fonction permet d'évaluer la qualité d'une communauté suivant deux aspects, à savoir, la cohésion interne et l'intensité de communication. Plus le score de qualité est proche de zéro, plus la communauté est jugée pertinente. C'est dans cette perspective que notre algorithme de détection de communautés cherche à optimiser la fonction de qualité en ajoutant les nœuds qui minimisent le plus le score de qualité. Dans la suite, nous présentons notre fonction de qualité avant de décrire le processus d'optimisation.

4.2.2.1 Fonction de qualité

Avant de rentrer dans les détails, nous définissons tout d'abord deux notions indispensables sur lesquelles se base notre fonction de qualité.

Définition 21 (Lien externe)
Un lien est dit « externe » par rapport à une communauté C s'il possède une seule extrémité dans C. Autrement dit, un lien externe relie un nœud de la communauté à un nœud du reste du réseau.

Définition 22 (Lien interne)
Un lien est considéré « interne » par rapport à une communauté C si ses deux extrémités se trouvent dans C. Autrement dit, un lien interne relie deux nœuds de la même communauté.

Notons que les liens externes peuvent être entrants ou sortants. De même pour les liens internes.

Notre fonction de qualité s'inspire des deux critères qualitatifs de définition de communauté, à savoir, la séparation avec le reste du réseau et la cohésion interne.

Le premier critère exige que les nœuds soient fortement liés entre eux et faiblement connectés au reste du réseau. Pour prendre en compte ce critère, nous supposons que les nœuds de la communauté doivent communiquer plus intensément entre eux qu'avec l'extérieur. Pour ce faire, nous divisons la somme des poids des liens externes par la somme des poids des liens internes. Ainsi, plus la valeur de cette fraction tend vers zéro, plus la communauté est considérée bien séparée du reste du réseau.

En effet, l'intuition derrière les définitions 21 et 22 est de pouvoir comparer l'intensité de communication à l'intérieur de la communauté (la somme des poids des liens internes) avec l'intensité de communication entre les nœuds de la communauté et l'extérieur du réseau (la somme des poids des liens externes). Rappelons que les liens internes/externes peuvent être entrants ou sortants. C'est donc dans ce sens qu'intervient l'aspect d'orientation de liens.

Concernant le second critère, une communauté est jugée cohésive si sa structure est proche d'une clique, c'est-à-dire que chaque nœud est (presque) lié à tous les autres. De ce fait, la communauté comprend un nombre de liens largement supérieur au nombre de nœuds. Nous utilisons donc la proportion $\frac{|V_C|}{|E_C|}$ pour évaluer la cohésion de communauté tel que $|V_C|$ désigne le nombre de nœuds de la communauté C et $|E_C|$ le nombre de liens.

Notre fonction qualité est définie par la formule suivante :

$$\psi(C) = \frac{\sum_{w_C^{out}}}{\sum_{w_C^{in}}} \times \frac{|V_C|}{|E_C|} \tag{4.4}$$

avec,

- C étant une communauté ego-centrée ;
- $\sum_{w_C^{out}}$ représentant la somme des poids des liens externes ;
- $\sum_{w_C^{in}}$ désignant la somme des poids des liens internes ;
- $|V_C|$ indiquant le nombre de nœuds de la communauté C.
- $|E_C|$ étant le nombre de liens de la communauté C.

Notons que si $\psi(C) \approx 0$, alors, la cohésion et la séparation de C sont pertinentes. Autrement dit, plus le score d'une communauté tend vers zéro, plus cette dernière est considérée « meilleure ».

Notre fonction de qualité est conçue, à la base, pour les réseaux orientés et pondérés. Toutefois, elle est bien compatible avec les réseaux non orientés et/ou non pondérés. Ceci est l'une des propriétés qui distingue notre fonction de qualité de celles existantes qui ne supportent que les réseaux orienté ou les réseaux pondérés. Dans la suite, nous expliquons comment notre fonction de qualité peut être appliquée aux réseaux non orientés et/ou non pondérés.

Pour les réseaux non orientés. Il est claire que la notion de liens internes/externes ne dépend pas de l'orientation de liens. Elle fonctionne aussi bien pour les réseaux orientés que pour ceux non orientés.

Pour les réseaux non pondérés. Dans ce cas, vu que les liens ne sont pas munis de poids, on utilise le nombre de liens internes/externes au lieu de leur poids. Ainsi, avec une légère reformulation, notre fonction de qualité devient :

$$\psi(C) = \frac{|E_C^{out}|}{|E_C^{in}|} \times \frac{|V_C|}{|E_C|} \tag{4.5}$$

Tels que $|E_C^{out}|$ désigne le nombre de liens externes et $|E_C^{in}|$ le nombre de liens internes.

4.2.2.2 Optimisation de la fonction de qualité

Dans cette section, nous proposons deux algorithmes d'optimisation de la fonction de qualité. Cette optimisation consiste à minimiser la fonction de qualité jusqu'à arriver à un seuil donné, noté τ. En effet, le processus d'optimisation est constitué de deux phases, à savoir, l'identification des nœuds candidats et leur ajout ou suppression à la communauté. Nous avons décrit deux méthodes d'identification de nœuds candidats dans la section 4.2.1. A présent, nous allons montrer comment nous allons ajouter les nœuds candidats pour construire une communauté. Nous distinguons deux possibilités : l'ajout par nœud ou l'ajout par bloc de nœuds

Ajout nœud par nœud. Il s'agit d'ajouter les nœuds individuellement après avoir vérifié s'ils diminuent le score de qualité. Pour ce faire, nous calculons le score de qualité lors de l'ajout de chaque voisin du nœud d'intérêt. Dans le cas d'une sélection aléatoire des nœuds, la complexité de l'ajout est de $|V_{N_{\{u,k\}}}|$.

Par contre, pour une sélection déterministe, la complexité est de $\rho.\sum_{i=1}^{|V_{N_{\{u,k\}}}|-1}(n-i)+(|V_{N_{\{u,k\}}}|-i)$, où n désigne le nombre de nœuds du réseau et ρ désigne le nombre d'itérations nécessaires pour atteindre le seuil τ.

En effet, la complexité de l'ajout par nœud devient inefficace pour les réseaux à large échelle. Pour faire face à cet inconvénient, nous avons proposé une seconde méthode d'ajout de nœuds à base de blocs.

Ajout par bloc. Cette méthode permet d'éviter le calcul individuel du score de qualité pour chaque nœud. Ainsi, le calcul est reporté et exécuté par groupe de nœuds. Cela a pour avantage de diminuer le nombre nécessaire d'itérations pour minimiser le score de qualité. En effet, le parcours par bloc consiste à sélectionner et ajouter les nœuds par groupe. Dans cette perspective, nous créons à chaque itération un groupe constitué de β nœuds. Pour définir un bloc de nœuds, nous calculons le degré d'affinité de tous les voisins du nœud d'intérêt en utilisant la métrique 4.3. Ensuite, nous trions ces voisins par ordre décroissant. Enfin, nous considérons que le bloc de nœuds est constitué des premiers β nœuds. Nous ajoutons le bloc identifié si le score de qualité diminue. Ce processus est répété jusqu'à l'atteinte d'un seuil τ fixé.

La complexité, cette fois-ci, est considérablement réduite à cause du parcours par bloc. Ainsi, elle devient $\frac{|V_{N_{\{u,k\}}}|}{\beta}$ pour une sélection aléatoire et $\rho.\frac{\sum_{i=1}^{|V_{N_{\{u,k\}}}|-1}(n-i)+(|V_{N_{\{u,k\}}}|-i)}{\beta}$ pour une sélection déterministe. Notons que la variation de la taille du bloc a un impact sur les communautés trouvées en termes de taille et de qualité.

En résumé, nous avons présenté deux méthodes d'ajout dont l'une peut être préférée à l'autre selon la taille du réseau. De plus, ces deux méthodes d'ajout, combinées aux méthodes d'identification, donnent quatre combinaisons possibles :

- une sélection déterministe avec ajout par nœud ;
- une sélection aléatoire avec ajout par nœud ;
- une sélection déterministe avec ajout par bloc de nœuds ;
- une sélection aléatoire avec ajout par bloc de nœuds.

Dans le chapitre relatif à la validation, nous allons comparer les résultats de chaque combinaison.

4.2.3 Algorithmes de détection de communautés égo-centrées

Après avoir décrit les principes de notre approche, nous présentons nos algorithmes de détection de communautés. Rappelons que le processus de détection de la communauté $C_{\{u,k\}}$ consiste à chercher les nœuds minimisant le score de qualité $\psi(C_{\{u,k\}})$ jusqu'à atteindre un seuil $\tau \geqslant 0$.

L'algorithme 1 présente la forme générale de notre méthode de détection de communautés.

De la ligne 1 à 14, nous voyons la présence d'une boucle *Pour* dont le but est de parcourir tous les nœuds d'intérêt du graphe et détecter la communauté ego-centrée de chacun.

Pour chaque nœud d'intérêt u, l'algorithme sauvegarde les voisins immédiats du u dans le vecteur M (ligne 3). Ensuite, l'algorithme initialise la communauté de u (ligne 4). Cette initialisation est faite suivant la méthode aléatoire (section 4.2.1.1) ou déterministe (section 4.2.1.2).

Dans la boucle *Pour*, il existe une autre boucle *Tant que* (lignes 5-12) qui constitue le cœur de l'algorithme. Cette boucle cherche à parcourir tous les voisins de l'ego et à ajouter ceux qui minimisent

Algorithme 1 Pseudocode du squelette global de notre mécanisme de détection de communautés égo-centrées.

Entrée: un graphe orienté et pondéré, des nœuds d'intérêt
Sortie: un ensemble de communautés ego-centrées

Données
G : un graphe orienté et pondéré,
$u \in G$: un nœud d'intérêt,
B : un bloc des voisins de u,
M : l'ensemble des voisins de u,
$\mathcal{L} = \emptyset$: liste de communautés ego-centrées détectées.

1: **Pour tout** $u \in G$ **Faire**
2:
3: Initialiser l'ensemble M
4: Initialiser la communauté ego-centrée de u
5: **Tant que** le seuil n'est pas encore atteint **Faire**
6:
7: Sélectionner un bloc B de l'ensemble M
8: **Si** l'ajout de B à la communauté diminue le score de qualité **Alors**
9: Ajouter B à la communauté
10: **Fin Si**
11: Supprimer B de l'ensemble M
12: **Fin Tant que**
13: Ajouter la communauté détectée à la liste $\mathcal{L}$
14: **Fin Pour**
15: Retourner la liste $\mathcal{L}$

le score de qualité. Cette boucle s'arrête une fois le seuil atteint. Si le seuil ne peut pas être atteint, la boucle s'arrête après avoir parcouru tous les voisins de l'ego.

En effet, la boucle *Tant que* comprend trois principales instructions. La première (ligne 7) désigne la sélection d'un ou de plusieurs nœuds voisins, cela dépend de la variante utilisée : l'ajout par nœud (paragraphe 4.2.2.2) ou l'ajout par bloc (paragraphe 4.2.2.2). La deuxième instruction (lignes 8-10) consiste à ajouter le(s) nœud(s) voisin(s) si son (leur) ajout fait diminuer le score de qualité. La troisième instruction (ligne 11) permet de supprimer le(s) nœud(s) sélectionné(s) de l'ensemble des voisins qui restent à parcourir. L'objectif de cette instruction est de ne pas rentrer dans une boucle infinie en ré-sélectionnant toujours le même élément.

Une fois la boucle *Tant que* est terminée, l'algorithme sauvegarde la communauté ego-centrée détectée dans la liste $\mathcal{L}$ (ligne 13). A la sortie de la boucle *Pour*, nous aurons dans la liste $\mathcal{L}$ l'ensemble des communautés ego-centrées des nœuds d'intérêt du graphe.

Le squelette 1 présente la forme globale de notre mécanisme de détection de communautés. A partir de ce squelette, nous avons conçu deux algorithmes dépendant de la taille du voisinage de l'égo. Le premier algorithme se limite au voisinage direct de l'ego alors que le second prend en compte le voisinage de longueur k, *i.e.,* tous les nœuds accessibles à partir de l'ego via un nombre de liens inférieur ou égal à k. Dans la suite, nous présentons les détails de chaque algorithme.

4.2.3.1 Détection de communautés à voisinage direct

Pour cet algorithme, la détection de communautés ego-centrées se limite au voisinage direct, c'est-à-dire, $k = 1$.

Définition 23 (Communauté ego-centrée à voisinage direct)
Soit u un nœud d'intérêt et $C_{\{u,1\}}$ sa communauté ego-centrée à voisinage direct. $C_{\{u,1\}}$ est constitué des nœuds du voisinage direct $N_{\{u,1\}}$ dont l'ajout minimise le score de qualité $\psi(C_{\{u,1\}})$ jusqu'à atteindre un seuil $\tau \geqslant 0$. Formellement, $C_{\{u,1\}}$ est définie comme suit :

$$C_{\{u,1\}} = S \cup \{B \in N_{\{u,1\}} \mid \psi(C_{\{u,1\}} \cup B) \leqslant \tau\} \tag{4.6}$$

Tel que S désigne la graine de la communauté et $B = \{v_1, v_2, ..., v_i\}$ un bloc de nœuds. Si B comprend un seul nœud ($|B| = 1$), alors il s'agit d'un ajout par nœud. En revanche, si $|B| > 1$, cela signifie que l'ajout est par bloc.

La figure 4.4 présente une vue globale de notre algorithme de détection de communautés ego-centrées à voisinage direct. Cette figure illustre de manière abstraite le principe de fonctionnement de l'algorithme. Autrement dit, elle ne cherche pas à évoquer les détails du type d'ajout (par nœud ou par bloc) et du type de sélection (aléatoire ou déterministe).

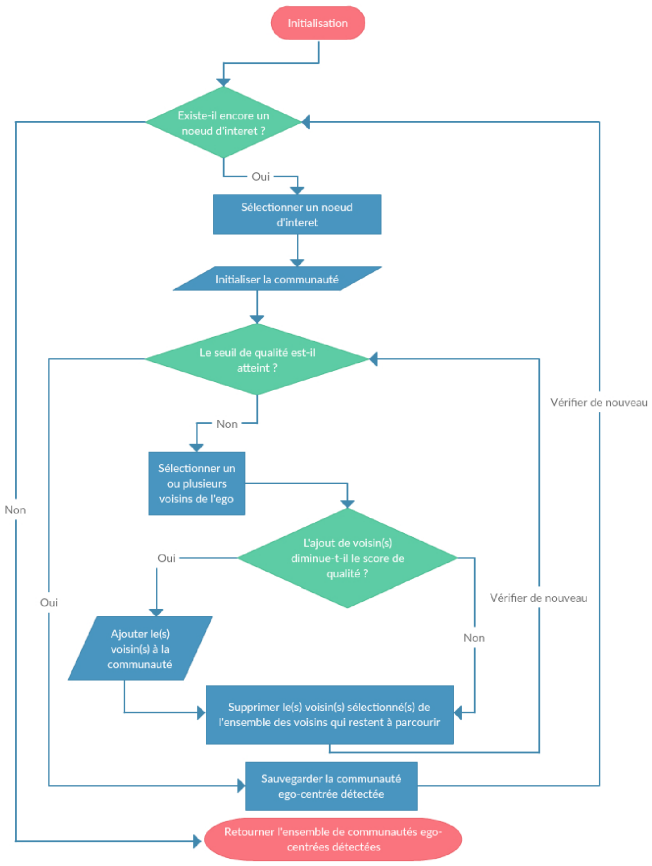


FIGURE 4.4 – Diagramme simplifié illustrant les grandes lignes de notre algorithme de détection de communautés ego-centrées à voisinage direct.

Dans la suite, nous présentons les pseudocodes des quatre variantes de l'algorithme de détection de communautés ego-centrées à voisinage direct. En effet, nous avons regroupé les quatre variantes en

deux, selon le type de sélection, à savoir :

1. Sélection aléatoire avec ajout par nœud (si $|B| = 1$) ou avec ajout par bloc (si $|B| > 1$) ;
2. Sélection déterministe avec ajout par nœud (si $|B| = 1$) ou avec ajout par bloc (si $|B| > 1$).

Au cours de la description de ces variantes, nous considérons que l'ajout est réalisé par bloc de nœuds sachant que l'ajout par nœud n'est qu'un cas particulier ($|B| > 1$) de l'ajout par bloc.

Sélection aléatoire avec ajout par nœud ou par bloc. L'algorithme 2 présente le pseudocode de la variante « Sélection Aléatoire » avec ajout par nœud (si $|B| = 1$) ou par bloc de nœuds (si $|B| > 1$).

Algorithme 2 Pseudocode de la variante « Sélection Aléatoire » de l'algorithme de détection de communautés égo-centrées à voisinage direct.

Entrée: un graphe orienté et pondéré, des nœuds d'intérêt
Sortie: un ensemble de communautés ego-centrées

Données
G : un graphe orienté et pondéré,
$u \in G$: un nœud d'intérêt,
$N_{\{u,1\}}$: le voisinage direct de u,
$C_{\{u,1\}}$: la communauté ego-centrée à voisinage direct,
$B \in N_{\{u,1\}}$: un bloc des voisins de u,
$\beta = |B|$: taille du bloc de nœuds,
M : l'ensemble des voisins de u,
τ : le seuil pré-défini,
$\mathcal{L} = \emptyset$: liste de communautés ego-centrées détectées.

1: **Pour tout** $u \in G$ **Faire**
2:
3: $M = N_{\{u,1\}}$
4: $B = \text{Random}(M, \beta)$
5: $C_{\{u,1\}} = u \cup B$
6: $M = M - B$
7: **Tant que** $\psi(C_{\{u,1\}}) \leqslant \tau$ **Faire**
8:
9: $B = \text{Random}(M, \beta)$
10: **Si** $\psi(C_{\{u,1\}} \cup B) \leqslant \psi(C_{\{u,1\}})$ **Alors**
11: $C_{\{u,1\}} = C_{\{u,1\}} \cup B$
12: **Fin Si**
13: $M = M - B$
14: **Fin Tant que**
15: $\mathcal{L} = \mathcal{L} \cup C_{\{u,1\}}$
16: **Fin Pour**
17: Retourner $\mathcal{L}$

Pour chaque nœud d'intérêt u de G, l'algorithme sauvegarde l'ensemble des voisins directs de u dans le vecteur M (ligne 3). Ensuite, il sélectionne aléatoirement un bloc B constitué de β nœuds du vecteur M (ligne 4). Cette sélection est faite via la fonction *Random* qui prend en paramètre un vecteur de nœuds M et un entier β. Puis, elle retourne β nœuds de M choisis au hasard. La ligne 5 permet d'initialiser la communauté avec le bloc B. Et pour ne pas rester bloqué dans une boucle infinie, l'algorithme supprime le bloc B de l'ensemble des nœuds qui restent à parcourir (ligne 6).

Si le score de qualité atteint le seuil τ, la boucle *Tant que* s'achève et la communauté résultante est ajoutée à la liste des communautés détectées (ligne 15). Sinon, l'algorithme sélectionne un autre bloc de nœuds (ligne 9) et l'ajoute à la communauté si son ajout fait diminuer le score de qualité (lignes 10-12). Ensuite, le bloc B est retiré des nœuds à parcourir (ligne 13). Ce processus itératif (lignes 7-14) est répété tant que le score de qualité est inférieur au seuil. Si le score de qualité ne peut pas être atteint, le processus itératif se termine une fois tous les voisins directs de u sont parcours.

Après avoir examiné tous les nœuds d'intérêt de G, l'algorithme retourne les communautés ego-centrées correspondantes (ligne 17).

Sélection déterministe avec ajout par nœud ou par bloc.

Hypothèse 1

Soit u un nœud d'intérêt. Nous considérons que la graine initiale de la communauté ego-centrée de u est constitué de u et du nœud v le plus connecté au voisinage de u.

L'intuition derrière est que le nœud v est le plus susceptible de faire partie à la communauté vu qu'il est le plus actif en terme d'intensité de communication et de liaisons topologiques. Techniquement, v désigne le nœud ayant la plus grande valeur du degré d'affinité par rapport au voisinage du nœud d'intérêt.

L'algorithme 3 présente le pseudocode de la variante « Sélection Déterministe » avec ajout par nœud (si $\beta = 1$) ou par bloc de nœuds (si $\beta > 1$).

Algorithme 3 Pseudocode de la variante « Sélection Déterministe » de l'algorithme de détection de communautés égo-centrées à voisinage direct.

Entrée: un graphe orienté et pondéré, des nœuds d'intérêt
Sortie: un ensemble de communautés ego-centrées

Données
G : un graphe orienté et pondéré,
$u \in G$: un nœud d'intérêt,
$N_{\{u,1\}}$: le voisinage direct de u,
$C_{\{u,1\}}$: la communauté ego-centrée à voisinage direct,
$B \in N_{\{u,1\}}$: un bloc des voisins de u,
$\beta = |B|$: taille du bloc de nœuds,
A : un vecteur contenant les degrés d'affinités des voisins de u,
M : une liste de type clé-valeur où les clés représentent les nœuds à parcourir et les valeurs les degrés d'affinité,
τ : le seuil pré-défini,
$\mathcal{L} = \emptyset$: liste de communautés ego-centrées détectées.

1: **Pour tout** $u \in G$ **Faire**
2:
3: $A = \mathcal{A}._v(N_{\{u,1\}}),\ \forall\ v \in N_{\{u,1\}}$
4: $M = \{N_{\{u,1\}}, A\}$
5: MergeSort(M, A)
6: $B = M[1..\beta]$
7: $C_{\{u,1\}} = u \cup B$
8: $M = M - B$
9: **Tant que** $\psi(C_{\{u,1\}}) \leqslant \tau$ **Faire**
10:
11: $A = \mathcal{A}._v(C_{\{u,1\}}),\ \forall\ v \in M$
12: MergeSort(M, A)
13: $B = M[1..\beta]$
14: **Si** $\psi(C_{\{u,1\}} \cup B) \leqslant \psi(C_{\{u,1\}})$ **Alors**
15: $C_{\{u,1\}} = C_{\{u,1\}} \cup B$
16: **Fin Si**
17: $M = M - B$
18: **Fin Tant que**
19: $\mathcal{L} = \mathcal{L} \cup C_{\{u,1\}}$
20: **Fin Pour**
21: Retourner $\mathcal{L}$

La première étape de l'algorithme est l'initialisation de la communauté ego-centrée de u (lignes 3-7). Au cours de cette étape, l'algorithme en s'appuyant sur l'hypothèse 1 cherche à initialiser la communauté avec le bloc de nœuds le plus connecté au voisinage de u. Pour ce faire, l'algorithme crée une liste M de type clé-valeur, l'initialise avec les voisins de u et la trie dans l'ordre décroissant suivant le degré d'affinité (lignes 3-5).

En détails, l'algorithme calcule le degré d'affinité de tous les voisins de l'ego par rapport à son voisinage direct (ligne 3). Ensuite, il initialise la liste M avec les clés : voisins de u et les valeurs : degrés d'affinité des voisins de u (ligne 4). La ligne 5 permet de trier M suivant le degré d'affinité. Le tri est réalisé dans l'ordre décroissant en utilisant l'algorithme *Tri fusion* (Kumar, Dutt et Saini, 2014).

Après avoir ordonné les voisins de l'ego selon leur degré d'affinité, la prochaine étape consiste à sélectionner un bloc de nœuds B constitué des premiers β nœuds de la liste M (ligne 6). Ces nœuds constituent les voisins les plus connectés au voisinage de l'ego. La communauté est donc initialisée avec l'ego et les nœuds de B (ligne 7). La ligne 8 permet supprimer le bloc B des voisins qui restent à parcourir. L'objectif de cette étape est de ne pas rester coincé dans une boucle infinie en re-sélectionnant à chaque itération le même bloc de nœuds.

Si le score de qualité n'est pas encore atteint, l'algorithme calcule le degré d'affinité des voisins restants par rapport aux nœuds de la communauté (ligne 11). Puis, il les trie suivant leur degré d'affinité (ligne 12). Ensuite, il sélectionne un bloc de nœuds B constitué des premiers β nœuds ayant les scores les plus élevés du degré d'affinité (ligne 13). Les nœuds de B ne sont ajoutés à la communauté que si leur ajout diminue le score de qualité (lignes 14-16). La ligne 17 permet de supprimer le bloc B des voisins qui restent à parcourir.

Le processus itératif (ligne 9-18) s'arrête dans deux cas : si le score de qualité devient supérieur au seuil donné ou si tous les voisins de l'ego sont parcourus. Une fois le processus itératif est achevé, l'algorithme ajoute la communauté résultante à la liste des communautés détectées. Les instructions du processus itératif racine (lignes 1-20) sont appliquées à chaque nœud d'intérêt du graphe G. A la sortie de ce processus, l'algorithme retourne l'ensemble des communautés ego-centrées détectées.

4.2.3.2 Détection de communautés à voisinage élargi

Cet algorithme permet de détecter les communautés ego-centrées sur un voisinage de longueur k. Pour ce faire, nous avons utilisé la variante de sélection déterministe mais, au lieu de l'ajout par bloc, nous utilisons un retrait par bloc de nœuds. En effet, le retrait par bloc implique un changement léger (mais très significatif) au niveau de la phase d'initialisation de communauté. Au lieu de commencer avec l'ego et quelques nœuds voisins, nous considérons que la graine initiale comprend tout le voisinage. Puis, nous cherchons à retirer les nœuds impertinents. L'intuition derrière est qu'il est plus pratique de commencer avec le voisinage et de retirer les nœuds faiblement liés à la communauté que de commencer avec le nœud d'intérêt et d'ajouter tous les voisins qui minimisent le score de qualité. Cette hypothèse se manifeste logique si les nœuds à retirer sont peu nombreux pas rapport à ceux restants, ce qui est généralement le cas d'une communauté par rapport au voisinage.

Définition 24 (Communauté ego-centrée de longueur k)

Soit u un nœud d'intérêt et $C_{\{u,k\}}$ sa communauté ego-centrée de longueur k définie par :

$$C_{\{u,k\}} = N_{\{u,k\}} - \{B \in N_{\{u,k\}} \mid \psi(C_{\{u,k\}} - B) \leqslant \tau\} \tag{4.7}$$

En effet, le processus de détection de $C_{\{u,k\}}$ consiste à initialiser la communauté par le voisinage élargi de u, noté $N_{\{u,k\}}$, et à supprimer de ce voisinage les blocs de nœuds B dont le retrait minimise le score de qualité jusqu'à atteindre un seuil donné.

L'algorithme 4 présente le pseudocode de notre algorithme de détection de communautés ego-centrées sur un voisinage de longueur k. Cet algorithme fonctionne en deux phases. La première phase comprend l'initialisation de communauté (lignes 3-4). La graine initiale est constituée du voisinage du nœud d'intérêt de longueur k. L'extraction du voisinage est réalisée via la fonction *Extractk-StepsEgoNeighborhood*. La seconde phase (lignes 5-15) concerne l'optimisation de la fonction de qualité sur l'ensemble du voisinage extrait. Dans la suite, nous expliquons les détails de chaque phase.

Algorithme 4 Pseudocode de l'algorithme de détection de communautés égo-centrées de longueur k.

Entrée: un graphe orienté et pondéré, des nœuds d'intérêt
Sortie: un ensemble de communautés ego-centrées

Données
G : un graphe orienté et pondéré,
$u \in G$: un nœud d'intérêt,
$N_{\{u,k\}}$: le voisinage ego-centré de longueur k du nœud u,
$C_{\{u,k\}}$: la communauté ego-centrée de longueur k,
$B \in N_{\{u,k\}}$: un bloc des voisins de u,
$\beta = |B|$: taille du bloc de nœuds,
T : l'ensemble des nœuds accessibles à partir de u via un nombre de liens inférieur ou égal à k,
A : un vecteur contenant les degrés d'affinités des voisins de u,
M : une liste de type clé-valeur où les clés représentent les nœuds à parcourir et les valeurs les degrés d'affinité,
τ : le seuil pré-défini,
$\mathcal{L} = \emptyset$: liste de communautés ego-centrées détectées.

1: **Pour tout** $u \in G$ **Faire**
2:
3: $T =$Extract$k$-StepsEgoNeighborhood(u, k)
4: $C_{\{u,k\}} = T$
5: **Tant que** $\psi(C_{\{u,k\}}) \leqslant \tau$ **Faire**
6:
7: $A = \mathcal{A}_{.v}(C_{\{u,k\}}),\ \forall\ v \in T$
8: $M = \{T, A\}$
9: MergeSort(M, A)
10: $B = M[1..\beta]$
11: **Si** $\psi(C_{\{u,k\}} - B) \leqslant \psi(C_{\{u,k\}})$ **Alors**
12: $C_{\{u,k\}} = C_{\{u,k\}} - B$
13: **Fin Si**
14: $T = T - B$
15: **Fin Tant que**
16: $\mathcal{L} = \mathcal{L} \cup C_{\{u,k\}}$
17: **Fin Pour**
18: Retourner $\mathcal{L}$

Extraction du voisinage de longueur k. Pour extraire le voisinage de longueur k du nœud d'intérêt, nous utilisons l'algorithme Breadth-First Search (Bundy et Wallen, 1984). Notre implémentation de BFS consiste à extraire les voisins de l'ego par étape du voisinage. Autrement dit, nous commençons d'abord par l'ensemble des nœuds N_1 qui sont directement liés à l'ego. Puis, nous extrayons l'ensemble des nœuds N_2 qui sont liés à N_1. Ensuite, nous extrayons l'ensemble des nœuds N_3 qui sont liés à N_2. Le processus d'extraction du voisinage continue jusqu'à ce qu'on arrive à l'ensemble de nœuds N_k qui sont liés à l'ego à travers k sauts ou liens. Les nœuds extraits à chaque itération sont sauvegardés dans le vecteur T à condition qu'ils ne soient pas déjà présents.

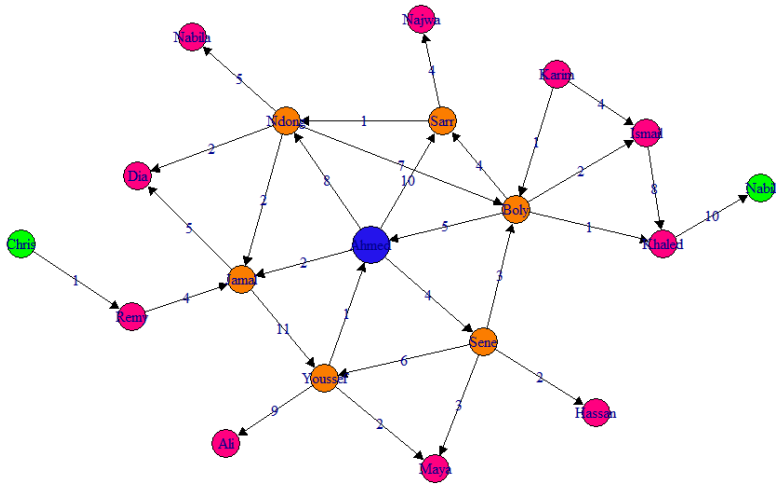


Figure 4.5 – Illustration du voisinage de longueur 3 de l'ego « Ahmed ».

Soit le réseau présenté sur la figure 4.5, nous présentons sur le tableau 4.2 le processus d'extraction du voisinage de longueur 3 du nœud « Ahmed ». Lors de la première itération, l'algorithme récupère les voisins directs du nœud «Ahmed ». A la seconde itération, il récupère les voisins de chacun des voisins directs de « Ahmed ». Par exemple, les voisins de « Sene » sont « Hassan » et « Maya » et les voisins de « Youssef » sont « Maya » et « Ali ». Mais, comme « Maya » est déjà ajouté à la liste des voisins, l'algorithme se contente d'ajoute uniquement « Ali ». Le même cas se répète avec « Jamal » et « Ndong » puisqu'ils ont « Dia » comme ami commun. Le résultat final est l'union des résultats de toutes les itérations après avoir supprimé les voisins répétés.

La complexité de BFS est de $O(|V_{N_{\{u,k\}}}| + |E_{N_{\{u,k\}}}|)$ tel que $|V_{N_{\{u,k\}}}|$ désigne le nombre de voisins de u dans un voisinage de longueur k et $|E_{N_{\{u,k\}}}|$ le nombre de liens.

Après avoir extrait et sauvegardé les nœuds du voisinage, l'algorithme les trie en ordre croissant par rapport à leur degré d'affinité. Le tri est effectué en utilisant l'algorithme *Tri fusion*. La complexité du tri est de $O(|M| \log(|M|))$. A ce stade, le voisinage de longueur k du nœud u est extrait (ligne 3). L'instruction suivante (ligne 4) consiste à initialiser la communauté de u avec le voisinage extrait.

Optimisation de fonction de qualité sur les nœuds du voisinage. Après avoir extrait le voisinage et initialisé la communauté. Le reste du pseudocode 4 consiste à optimiser la fonction de qualité sur les nœuds du voisinage. En d'autres mots, il s'agit de parcourir les voisins de u et de supprimer ceux dont le retrait diminue le score de qualité (lignes 5-15).

En effet, le processus d'optimisation de fonction de qualité consiste à :

Tableau 4.2 – Illustration du processus d'extraction du voisinage de longueur 3 de l'ego « Ahmed ».

Première itération	
Résultat : Boly, Sene, Youssef, Jamal, Ndong, Sarr	
Seconde itération	
Nœud	*Voisins*
Boly	Karim, Ismail, Khaled
Sene	Hassan, Maya
Youssef	Maya, Ali
Jamal	Remy, Dia
Ndong	Dia, Nabila
Sarr	Najwa
Résultat : Karim, Ismail, Khaled, Hassan, Maya, Ali, Remy, Dia, Nabila, Najwa	
Troisième itération	
Nœud	*Voisins*
Khaled	Nabil
Remy	Chris
Résultat : Nabil, Chris	

1. Calculer le degré d'affinité des voisins de l'ego qui ne sont pas encore parcourus, à savoir, les nœuds du vecteur T (ligne 7).
2. Construire la liste M en lui attribuant les voisins de u et leurs degrés d'affinité (ligne 8).
3. Trier M suivant le degré d'affinité (ligne 9). Le tri est réalisé en ordre croissant et fait en utilisant l'algorithme *Tri fusion*.
4. Sélectionner un bloc de nœuds $B \in M$ (ligne 10). Ce bloc est constitué des premiers β nœuds de M ayant les plus petites valeurs d'affinité. Autrement dit, ce sont les nœuds les plus faiblement liés à la communauté.
5. Retirer le bloc B de la communauté si son retrait fait diminuer le score de qualité. L'algorithme supprime également le bloc B de l'ensemble des voisins qui restent à parcourir.

Le processus itératif (lignes 5-15) est appliqué à chaque voisin de u qui se trouve dans le voisinage de longueur k. Ce processus s'arrête une fois le seuil atteint ou après avoir parcouru tous les nœuds du voisinage. A la ligne 16, l'algorithme récupère la communauté résultante et la sauvegarde dans la liste des communautés ego-centrées détectées. Le processus itératif père (lignes 1-17) est appliqué à tout nœud d'intérêt du graphe G. Après avoir parcouru tous les nœuds d'intérêt du graphe, l'algorithme retourne la liste des communautés ego-centrées correspondantes (ligne 18).

4.2.4 Illustration des algorithmes

Dans cette section, nous illustrons le processus de détection de communautés ego-centrées à voisinage direct et à voisinage élargi. Pour ce faire, nous présentons trois exemples, à savoir :

1. Détection de communautés ego-centrées à voisinage direct en utilisant un ajout par nœud.
2. Détection de communautés ego-centrées à voisinage direct en utilisant un ajout par bloc.
3. Détection de communautés ego-centrées à voisinage élargi en utilisant un retrait par bloc.

Le type de sélection utilisé dans ces exemples étant la sélection déterministe. Dans la suite de cette section, nous décrivons le réseau d'illustration avant de présenter les détails de chaque exemple.

La figure 4.6 présente le réseau d'illustration. Ce réseau est orienté et pondéré. Il est constitué de 18 nœuds et 39 liens. Les valeurs qui figurent sur les liens représentent les poids. A partir de ce réseau, nous avons choisi deux nœuds d'intérêt grâce à leurs positions centrales, à savoir, les nœuds 1 et 8.

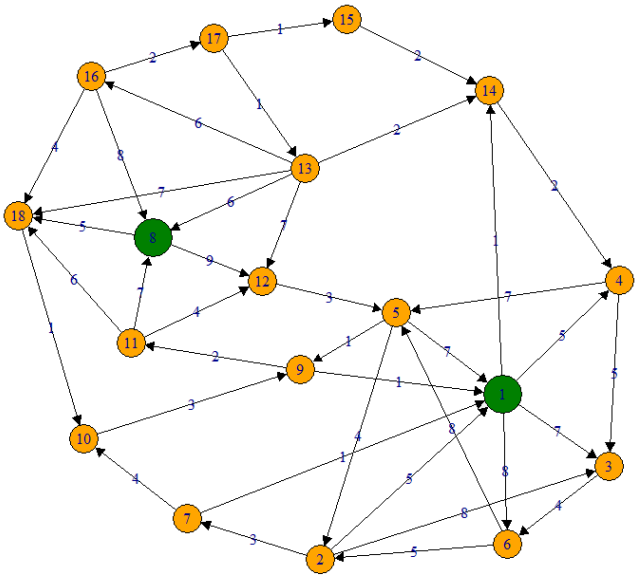


FIGURE 4.6 – Réseau d'illustration représenté sous forme de graphe orienté et pondéré.

4.2.4.1 Détection de communautés à voisinage direct via ajout par nœud

Le tableau 4.3 décrit le processus d'exécution de l'algorithme de détection de communautés ego-centrées à voisinage direct. Le nœud d'intérêt choisi est celui de label 1 sur la figure 4.6. La variante utilisée de cet algorithme étant l'ajout par nœud avec sélection déterministe.

Tableau 4.3 – Illustration de la procédure de détection de communauté ego-centrée à voisinage direct du nœud 1 en utilisant la variante d'ajout par noeud avec sélection déterministe.

Nœuds de communauté	Voisins à parcourir	Degrés d'affinité	Nœud sélectionné	Score de qualité	Décision	Seuil atteint ?
{1}	{2, 3, 4, 5, 6, 7, 9, 14}	{1.992, 1.98, 1.65, 1.74, 1.99, 1.20, 0.89, 1.04}	2	20	—	—
{1, 2}	{3, 4, 5, 6, 7, 9, 14}	{1.27, 0.86, 1.07, 1.22, 1.20, 0.03, 0.03}	3	2.45	Ajouté	Non
{1, 2, 3}	{4, 5, 6, 7, 9, 14}	{1.21, 1.07, 1.49, 1.20, 0.03, 0.03}	6	0.63	Ajouté	Non
{1, 2, 3, 6}	{4, 5, 7, 9, 14}	{1.21, 1.29, 1.20, 0.03, 0.03}	5	0.26	Ajouté	Non
{1, 2, 3, 5, 6}	{4, 7, 9, 14}	{1.65, 1.20, 0.89, 0.03}	4	0.098	Ajouté	Non
{1, 2, 3, 4, 5, 6}	{7, 9, 14}	{1.20, 0.89, 1.04}	7	0.090	Ajouté	Non
{1, 2, 3, 4, 5, 6, 7}	{9, 14}	{0.89, 1.04}	14	0.08	Ajouté	Non
{1, 2, 3, 4, 5, 6, 7, 14}	{9}	{0.89}	9	0.10	Rejeté	Non
Communauté détectée : {1, 2, 3, 4, 5, 6, 7, 14}						

Le tableau 4.3 est constitué de sept colonnes. La première présente les nœuds de la communauté à

chaque itération. La seconde colonne désigne les voisins de l'ego qui restent à parcourir. La troisième colonne comprend les degrés d'affinité de ces voisins. La quatrième colonne représente le nœud sélectionné à chaque itération. La cinquième colonne spécifie la décision prise, est-ce un ajout ou un rejet. La dernière colonne donne un aperçu sur la condition d'arrêt, à savoir, si le score de qualité a atteint le seuil ou non. Dans cet exemple, nous avons considéré que le seuil de qualité $\tau = 0.05$. De ce fait, le processus d'optimisation s'arrête si ce seuil est atteint, sinon, après avoir parcouru tous les voisins de l'ego.

La première ligne du tableau 4.3 correspond à l'étape d'initialisation de communauté. Le voisin sélectionné au cours de cette itération est le nœud 2 vu qu'il a le degré d'affinité le plus élevé (1.992). Ainsi, la graine initiale est constituée de l'ego et le nœud 2.

Après avoir initialisé la communauté, l'algorithme procède à un processus itératif qui fonctionne comme suit : à chaque itération, l'algorithme calcule le degré d'affinité des voisins de l'ego. Puis, il sélectionne le voisin ayant la valeur la plus élevée du degré d'affinité. Ce voisin est ajouté à la communauté si son ajout diminue le score de qualité. Ensuite, le voisin sélectionné est retiré de l'ensemble des voisins qui restent à parcourir. Ce processus est appliqué à chaque voisin de l'ego tant le seuil n'est pas atteint. Suite à ce processus, nous remarquons que l'ajout des nœuds 3, 4, 5, 6, 7 et 14 a impliqué une diminution du score de qualité contrairement au nœud 9.

Une fois la communauté du nœud 1 est construite, l'algorithme répète le même processus pour détecter la communauté du nœud 8. La figure 4.7 présente les deux communautés détectées. Notons que tous les voisins du nœud 8 ont été ajoutés à la communauté. Cela peut s'expliquer par le niveau élevé d'intensité de communication au sein de la communauté. Autrement dit, ces voisins communiquent autant entre eux et faiblement avec le reste du réseau. Ainsi, leur ajout fait diminuer le score de qualité.

(a) Communauté ego-centrée du nœud 1.

(b) Communauté ego-centrée du nœud 8.

FIGURE 4.7 – Communautés ego-centrées à voisinage direct des nœuds 1 et 8. La variante utilisée étant l'ajout par noeud avec sélection déterministe.

4.2.4.2 Détection de communautés à voisinage direct via ajout par bloc

Le tableau 4.4 décrit le processus d'exécution de l'algorithme de détection de communautés ego-centrées à voisinage direct. Le nœud d'intérêt choisi est celui de label 1 sur la figure 4.6. La variante utilisée de cet algorithme étant l'ajout par bloc avec sélection déterministe.

Dans cet exemple, nous avons considéré que la taille de bloc égale à 2. Ainsi, à chaque itération, nous sélectionnons deux voisins de l'ego. Le seuil de qualité que nous avons choisi égal à zéro, c'est-

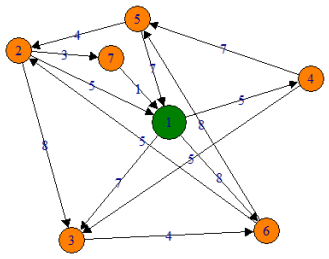
Tableau 4.4 – Illustration de la procédure de détection de communauté ego-centrée à voisinage direct du nœud 1 en utilisant la variante d'ajout par bloc avec sélection déterministe.

Nœuds de communauté	Voisins à parcourir	Degrés d'affinité	Bloc sélectionné	Score de qualité	Décision
{1}	{2, 3, 4, 5, 6, 7, 9, 14}	{1.992, 1.98, 1.65, 1.74, 1.99, 1.20, 0.89, 1.04}	{2, 6}	2.72	—
{1, 2, 6}	{3, 4, 5, 7, 9, 14}	{1.27, 0.86, 1.29, 1.20, 0.03, 0.03}	{5, 3}	0.098	Ajouté
{1, 2, 3, 5, 6}	{4, 7, 9, 14}	{1.65, 1.20, 0.89, 0.03}	{4, 7}	0.090	Ajouté
{1, 2, 3, 4, 5, 6, 7}	{9, 14}	{0.89, 1.04}	{14, 9}	0.091	Rejeté
Communauté détectée : {1, 2, 3, 4, 5, 6, 7}					

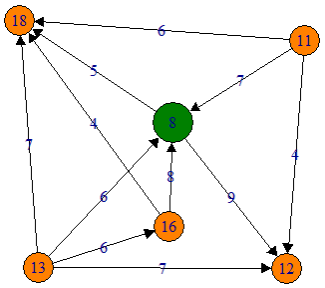
à-dire que nous cherchons la minimisation la plus forte de la fonction de qualité. Ainsi, le processus d'optimisation ne s'arrête qu'après avoir parcouru tous les voisins de l'ego. En effet, théoriquement, le score de qualité atteint zéro si la somme des poids des liens entrants égale à l'infini ou le nombre de liens de communauté égal à l'infini. De ce fait, les scores de qualité tendent vers zéro mais ne peuvent pas l'attendre. Vu que le seuil 0, en pratique, ne peut pas être atteint, nous n'avons pas mis la colonne « Seuil atteint » contrairement au tableau 4.3 où le seuil était 0.05.

Sur le tableau 4.4, nous remarquons que la variante d'ajout par bloc a pu détecter une communauté similaire à celle de l'ajout par nœud tout en faisant moins d'itérations (3 itérations au lieu de 7). C'est là où se manifeste l'aspect de réduction de complexité de l'ajout par bloc.

Après avoir construit la communauté du nœud 1, l'algorithme répète le même processus pour détecter la communauté du nœud 8. La figure 4.8 présente les deux communautés détectées.



(a) Communauté ego-centrée du nœud 1.



(b) Communauté ego-centrée du nœud 8.

FIGURE 4.8 – Communautés ego-centrées à voisinage direct des nœuds 1 et 8. La variante utilisée étant l'ajout par bloc avec sélection déterministe.

4.2.4.3 Détection de communautés à voisinage élargi via retrait par bloc

Le tableau 4.5 décrit le processus d'exécution de l'algorithme de détection de communautés ego-centrées à voisinage élargi. Le nœud d'intérêt choisi est celui de label 1 sur la figure 4.6. Cet algorithme se base sur un retrait par bloc avec sélection déterministe.

La première ligne du tableau 4.5 présente les valeurs des paramètres que nous avons choisies dans cet exemple. La seconde ligne correspond à l'initialisation de communauté. L'algorithme considère que la graine initiale comprend tout le voisinage en deux étapes. Puis, à chaque itération, il calcule le degré

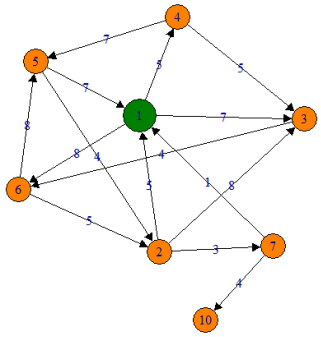
Tableau 4.5 – Illustration de la procédure de détection de communauté ego-centrée à voisinage élargi du nœud 1.

Nœuds de communauté	**Voisins à parcourir**	**Degrés d'affinité**	**Bloc sélectionné**	**Score de qualité**	**Décision**
Taille de bloc = 3, Seuil pré-défini = 0.					
{1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15}	—	—	—	0.21	—
{1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15}	{2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15}	{1.992, 1.989, 1.986, 1.994, 1.99, 1.95, 1.96, 1.51, 1.07, 1.43, 1.04, 1.96, 0.83}	{15, 13, 11}	0.15	Supprimé
{1, 2, 3, 4, 5, 6, 7, 9, 10, 12, 14}	{2, 3, 4, 5, 6, 7, 9, 10, 12, 14}	{1.992, 1.989, 1.986, 1.994, 1.99, 1.95, 1.46, 1.51, 0.69, 1.04}	{12, 14, 9}	0.07	Supprimé
{1, 2, 3, 4, 5, 6, 7, 10}	{2, 3, 4, 5, 6, 7, 10}	{1.992, 1.989, 1.65, 1.56, 1.99, 1.95, 0.91}	{10, 5, 4}	0.152	Gardé
{1, 2, 3, 4, 5, 6, 7, 10}	{2, 3, 6, 7}	{1.992, 1.989, 1.99, 1.95}	{7, 3, 6}	0.31	Gardé
{1, 2, 3, 4, 5, 6, 7, 10}	{2}	{1.992}	{2}	0.43	Gardé
Communauté détectée : $\{1, 2, 3, 4, 5, 6, 7, 10\}$					

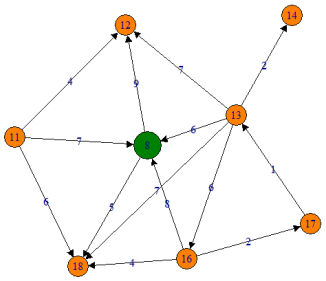
d'affinité des voisins de l'ego et sélectionne un bloc de trois voisins ayant les plus petites valeurs du degré d'affinité. Ce bloc est retiré de la communauté si son retrait diminue le score de qualité. Sinon, l'algorithme garde le bloc dans la communauté et passe au bloc suivant. Suite à ce processus, nous remarquons que le retrait des blocs {15, 13, 11} et {12, 14, 9} implique une diminution du score de qualité contrairement aux autres blocs.

Notons que d'une itération à l'autre, certaines nœuds ont gardé le même degré d'affinité vu que leurs voisins dans la communauté n'ont pas changé.

Après avoir construit la communauté du nœud 1, l'algorithme répète le même processus pour détecter la communauté du nœud 8. La figure 4.9 présente les deux communautés détectées.



(a) Communauté ego-centrée du nœud 1.



(b) Communauté ego-centrée du nœud 8.

FIGURE 4.9 – Communautés ego-centrées à voisinage de longueur 2 des nœuds 1 et 8.

4.3 Conclusion

Dans ce chapitre, nous avons présenté nos algorithmes de détection de communautés égo-centrées statiques dans un réseau orienté et pondéré. Pour y arriver, nous avons commencé par décrire et représenter un réseau social sur la base de données issues des applications sociales. Ensuite, nous avons identifié les trois briques de base de notre méthode détection de communautés, à savoir, l'initialisation

d'une communauté, l'évaluation de sa pertinence et enfin les algorithmes permettant de construire une communauté ego-centrée à partir du voisinage direct ou élargi. Pour chacune des briques de base, nous avons présenté différentes variantes. Nous évaluons ces différentes propositions dans le chapitre relatif à la validation et nous parlerons au prochain chapitre du suivi de communautés dynamiques.

Chapitre 5

Gestion de la dynamique des communautés ego-centrées

Dans le chapitre précédent, nous avons présenté notre solution de détection de communautés statiques ego-centrées. L'objectif de ce chapitre est d'expliquer comment nous faisons le suivi de l'évolution de communautés dynamiques ego-centrées. Rappelons que notre stratégie de gestion de communautés dynamiques consiste à :

1. faire plusieurs captures (ou instantanés) du réseau au fil du temps ;
2. détecter des communautés statiques sur chaque instantané ;
3. identifier la nature d'évolution de communautés entre deux instantanés successifs.

Dans cette perspective, nous commençons ce chapitre par la description de notre méthode de gestion des instantanés. Ensuite, nous présentons notre approche de suivi de l'évolution des communautés dynamiques. En effet, la suite du chapitre est organisée en plusieurs sections. Premièrement, nous présentons les concepts et hypothèses sur lesquels notre solution s'appuie dans la section 5.1.

Deuxièmement, dans la section 5.1.2, nous passons en revue les différents modes de représentation de l'aspect temps dans les réseaux dynamiques. De plus, nous expliquons dans la section 5.2 la stratégie de décomposition du réseau dynamique. Enfin, nous présentons notre algorithme de suivi de communautés dynamiques dans la section 5.3 avant de conclure dans la section 5.4.

5.1 Rappels des notions et concepts des réseaux dynamiques

Dans cette section, nous définissons l'ensemble des notions nécessaires pour bien comprendre la suite.

5.1.1 Réseau dynamique

Définition 25 (Réseau dynamique)
Un réseau est dit dynamique s'il subit des changements au fil du temps. Ces changements sont marqués par l'apparition ou la disparition de nœuds/liens de façon continue dans le temps. De même, ces changements peuvent être matérialisés par l'intensification de l'interaction entre les nœuds.

Un des aspects importants dans l'étude des réseaux dynamiques est la représentation de l'évolution de ceux-ci au fil du temps.

5.1.2 Modes de représentation de l'aspect temps

L'évolution d'un réseau dynamique au fil du temps peut être modélisée de plusieurs manières (Holme, 2015), parmi lesquelles, nous retenons deux que nous allons présenter.

- Le modèle à temps agrégé : il considère qu'un réseau dynamique est une suite de réseaux statiques. De ce fait, il consiste à agréger les états du réseau par période d'une année, ou d'un mois, ou d'une semaine, etc.
- Le modèle à temps réel : il définit le réseau comme étant un objet dynamique qui change au fil du temps. Ainsi, chaque nœud/lien est muni d'une durée de vie au cours de laquelle il apparaît dans le réseau. Avec ce modèle, le temps peut être discrétisé ou continu :
  - le modèle à temps discret : il consiste à décrire les liens sous forme d'un triplet (u, v, t) qui indique qu'il y a eu une interaction entre les nœuds u et v à l'instant t. Si le réseau est orienté, l'ordre des nœuds implique que le sens d'interaction s'oriente de u vers v. Ce modèle est souvent utilisé pour modéliser les interactions instantanées puisque celles-ci ne durent pas dans le temps. Des exemples de ce type d'interactions incluent les réseaux de communication asynchrone comme dans les applications de type courrier électronique, la messagerie instantanée, etc. ;
  - le modèle à temps continu : il conçoit les interactions entre les nœuds comme étant des événements. De ce fait, chaque événement est muni d'une date d'apparition et d'une date de disparition. Formellement, le quadruplet (u, v, s, e) indique l'existence d'un lien entre u et v dans l'intervalle de temps $[s, e]$. Les réseaux d'appel téléphoniques représentent un exemple de ce type d'interactions.

Quant à la représentation des nœuds au fil du temps, deux cas sont possibles :

- le premier cas considère que le réseau comprend un nombre fixe de nœuds ayant tous la même date d'apparition et disparaîtront ensemble. Ainsi, seuls les liens sont les éléments évoluant dans le temps. Les nœuds n'ayant pas d'interactions au cours d'un intervalle de temps sont considérés comme isolés, *i.e. ;* sans voisins.
- le second cas définit les nœuds comme des objets dynamiques qui apparaissent/disparaissent tout au long du processus d'évolution. De ce fait, une (ou plusieurs) durée (s) de vie est (sont) attribuée (s) à chaque nœud du réseau.

La première méthode est utilisée généralement pour observer les interactions entre un nombre fixe de nœuds au cours d'une période donnée. Par contre, la seconde méthode est plus adaptée quand l'ensemble des nœuds du réseau varie dans le temps. Autrement dit, c'est dans le cas des réseaux avec des nœuds pouvant joindre le réseau à tout moment.

Dans le cadre de notre thèse, nous nous plaçons dans le contexte des réseaux à temps réel avec un modèle à temps continu. De ce fait, même si le réseau est à temps discret, nous considérons que chaque lien est muni d'un intervalle de temps dont le début est égal à la fin.

5.1.3 Instantanés d'un réseau dynamique

Nous rappelons que notre objectif est décomposer l'évolution d'un réseau dynamique au fil du temps à travers des instantanés. Pour ce faire, nous définissons un instantané comme suit.

Définition 26 (Instantané)
Un instantané est une capture de la structure d'un réseau dynamique au cours d'une période donnée. En d'autres mots, il s'agit de représenter l'ensemble des nœuds/liens du réseau dynamique sur un intervalle de temps sous forme d'un réseau statique.

Autrement dit, un instantané consiste à concaténer toutes les interactions qui ont eu lieu de l'instant t jusqu'à $t^{'}$, même si certaines d'elles ne durent pas tout au long de l'intervalle $[t, t^{'}]$. Au cours de cette période, si une interaction apparait à plusieurs reprises, nous la représentons par un seul lien dont le poids est égal à la somme des poids des liens qui correspondent aux différentes apparitions.

L'intervalle de temps qui délimite les contours de l'instantané est appelé une fenêtre temporelle. En effet, la gestion de fenêtre temporelle peut se faire de deux manières : soit par découpage en parts égales du temps global, soit par découpage en parts différentes. Le découpage en parts égales, qui correspond à des fenêtres temporelles de taille identique, a l'avantage d'être simple et facile à implémenter. Toutefois, si le réseau ne change pas au cours de k fenêtres, cela conduit à des captures identiques dont l'analyse ne donnera aucun résultat intéressant. De même, si la structure du réseau évolue de manière irrégulière, la fixation de la taille de la fenêtre peut poser problème. En effet, de petites tailles multiplient les instantanés et de grandes tailles cachent les moments précis d'occurrence des changements significatifs de la structure du réseau dynamique. Pour éviter ce problème de calibrage de la fenêtre temporelle ainsi que favoriser des captures pertinentes qui intègrent suffisamment de changements, nous proposons une stratégie de gestion dynamique de la taille des fenêtres temporelles. Pour ce faire, nous avons défini une fonction de qualité globale dont le but est de quantifier les changements que le réseau subit au cours d'une période donnée. Nous considérons qu'il est temps de capturer une nouvelle fenêtre temporelle si le réseau subit un changement considérable : techniquement, si la différence de qualité entre deux instants dépasse un seuil donné.

5.2 Gestion de la taille des fenêtres temporelles

5.2.1 Principe de gestion des fenêtres

L'idée de base de notre solution est de ne faire des captures que lorsque cela est nécessaire. Autrement dit, nous quantifions le changement de la structure au fil du temps de sorte qu'une capture ne soit possible que si un seuil de modifications est atteint. De plus, nous proposons une modélisation de la variation du réseau de manière à prédire les instants à partir desquels la quantité de changements atteint le seuil fixé. Ainsi, sur la base de cette modélisation, on est capable de prédire les instants de captures sans avoir à fixer une périodicité fixe. L'avantage de notre solution est qu'elle permet d'avoir des captures dépendant intégralement de l'évolution du réseau et non de la méthode de capture. De fait, notre approche est semblable à un processus d'identification de la taille optimale d'une fenêtre temporelle. Ce processus peut être résumé en deux étapes :

1. utiliser une fonction de qualité globale pour mesurer le taux de changements du réseau à un instant donné ;
2. comparer le taux des changements avec un seuil η. Si la différence de qualité du réseau entre deux instants dépasse la valeur du seuil, alors on conclut que le réseau a été considérablement modifié et qu'il est temps de faire une capture.

Nous présentons dans la suite de cette section la fonction de qualité globale que nous utilisons.

5.2.2 Qualité des instantanés

Comme énoncé précédemment, nous utilisons une fonction ψ_G pour mesurer la qualité du réseau dynamique au cours d'une période donnée tout en prenant en compte la direction des liens et l'intensité de communications entre les nœuds. En effet, cette fonction est une variante de celle que nous avons proposée dans le chapitre précédent. Leur différence principale réside dans le fait que la dernière évalue la qualité d'une communauté ego-centrée dans un réseau statique tandis que ψ_G évalue la qualité d'un réseau dynamique au cours d'une période donnée.

A l'image de la fonction de qualité présentée dans le chapitre précédent, deux critères qualitatifs sont retenus pour définir ψ_G, à savoir, la densité du réseau et l'intensité de communication entre les nœuds. Nous rappelons que pour le premier critère, le réseau est jugé dense si sa structure est proche d'une clique, c'est-à-dire que chaque nœud est (presque) lié à tous les autres. De ce fait, le réseau comprend un nombre de liens largement supérieur au nombre de nœuds. Quant au second critère, il désigne l'abondance d'interactions entre les nœuds du réseau au fil du temps.

Soit $\mathcal{N}$ un réseau dynamique et $\mathcal{N}_t$ la capture du réseau à l'instant t. Pour avoir une idée de l'intensité de communication entre les nœuds au sein de $\mathcal{N}_t$, nous calculons l'inverse de la somme des poids de tous les liens $\frac{1}{\sum_{w_{\mathcal{N}_t}}}$. L'idée est que plus la communication est intense au sein de $\mathcal{N}_t$ (valeur élevée des poids), plus $\frac{1}{\sum_{w_{\mathcal{N}_t}}}$ tend vers 0.

Concernant la densité de $\mathcal{N}_t$, nous reprenons l'idée de la fonction de qualité locale : plus la structure topologique du réseau à l'instant t s'approche d'une clique, plus l'instantané est jugé cohésif. Par conséquent, la proportion $\frac{|V_{\mathcal{N}_t}|}{|E_{\mathcal{N}_t}|}$ permet d'évaluer la cohésion de $\mathcal{N}_t$. Tels que $|V_{\mathcal{N}_t}|$ désigne le nombre de nœuds dans $\mathcal{N}_t$ et $|E_{\mathcal{N}_t}|$ le nombre de liens.

Notre fonction de qualité qui est de portée globale est définie comme suit :

$$\psi_G(\mathcal{N}_t) = \frac{1}{\sum_{w_{\mathcal{N}_t}}} + \frac{|V_{\mathcal{N}_t}|}{|E_{\mathcal{N}_t}|} \tag{5.1}$$

Comme notre fonction de qualité dépend du temps, il est trivial de comparer ses valeurs aux instants t et $t+1$. Si la variation des valeurs est considérable, cela montre que le réseau a grandement changé. Ce changement est le fruit de la variation du nombre de nœuds, de liens, de leur direction ou de leur poids. Formellement, nous considérons que le réseau a subi un changement considérable si la différence entre sa qualité à l'instant t et celle à l'instant $t+1$ devient supérieur au seuil η :

$$|\psi_G(\mathcal{N}_t) - \psi_G(\mathcal{N}_{t+1})| \geqslant \eta \tag{5.2}$$

Notons que plus la valeur du seuil est grande, plus l'est le nombre de changements entre deux instantanés successifs.

5.2.3 Délimitation des fenêtres temporelles

Une fois que nous avons décrit comment nous évaluons la qualité à l'instant t d'un réseau, il nous reste à voir comment modéliser la variation de la fonction au fil du temps. L'idée est de voir comment utiliser les valeurs récentes pour prédire les futures valeurs de la fonction, et de procéder, à priori, à

la comparaison de celles-ci entre des instants successifs. En effet, connaissant les valeurs à l'instant t, nous voulons estimer la valeur à l'instant $t+1$ afin de vérifier si leur différence dépasse le seuil. Dans l'affirmative, on marque ces deux instants comme périmètres de deux captures. Autrement dit, on décide de faire une nouvelle capture à $t+1$. De plus, comme nous ne visons pas à prédire les valeurs de la fonction sur une longue période à l'horizon, mais plutôt des valeurs à court terme, nous faisons appel aux séries temporelles. Nous rappelons, par conséquent, le concept de série temporelle avant de mettre l'accent sur comment nous prédisons les instants de capture.

5.2.3.1 Notions de série temporelle

On peut comprendre une série temporelle comme une suite d'observations répétées d'un même phénomène à des dates différentes. Les dates sont le plus souvent équidistantes (séries journalières, hebdomadaires, mensuelles, trimestrielles etc.). Par contre, dans certaines situations, il est possible d'avoir une séquence irrégulière comme le cas des banques au regard des jours non ouvrables.

Définition 27 (Série temporelle)
Une série temporelle est une suite finie de données indexées par le temps, notées $(x_1, ..., x_n)$*. Le pas du temps désigne un intervalle de temps fixe. Il peut être une minute, une heure, un jour, une année, etc.*

Une série temporelle est aussi appelée série chronologique et permet de représenter l'évolution d'une quantité spécifique au cours du temps. Une telle série peut permettre d'analyser le comportement d'une quantité en s'appuyant sur son évolution passée pour prévoir son futur. La figure 5.1 présente un exemple d'une série temporelle montrant la consommation d'électricité des français entre les mois de juin et juillet 1991. Dans cette série, nous avons une donnée toutes les demi-heures pendant 35 jours, soit 1680 données. On observe deux composantes périodiques correspondant à la journée (48 unités de temps) et à la semaine ainsi qu'un effet massif relatif au week-end (Viano et Philippe, p.d.).

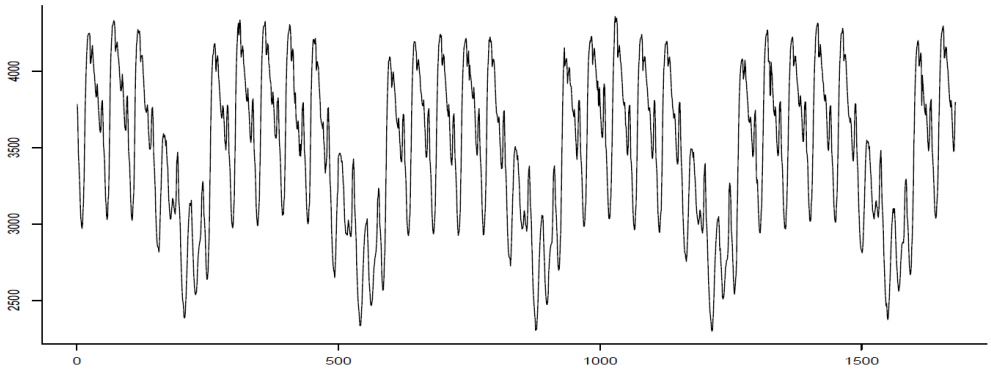


Figure 5.1 – Exemple d'une série temporelle montrant la consommation d'électricité des français au cours de deux mois.

Une utilisation courante des séries temporelles, est leur capacité à permettre des prévisions de ventes de produits, d'adhésion à un service, de variation d'une clientèle, etc. Ces prévisions sont faites sur la base de données périodiques pour un horizon à court terme. Les techniques de projections utilisant ce principe sont regroupées sous la rubrique du lissage exponentiel. Il existe plusieurs techniques de lissage permettant la prise en compte de situations variant du simple au complexe.

Dans le cadre de cette thèse, nous nous limitons au lissage exponentiel simple. Ce choix est motivé par le fait que nous ne visons pas à donner une tendance d'évolution de la fonction de qualité, mais plutôt d'approximer les prochaines valeurs immédiates. Nous rappelons au lecteur que pour la prédiction de série avec une tendance, comme la vente de produits, il est recommandé d'utiliser le lissage exponentiel double, ou le lissage de Holt, ou celui de Winters. Par ailleurs, nous n'avons pas choisi une régression simple car elle tente de prolonger une série alors que nous visons à obtenir une valeur lissée en t pour la reporter tout simplement en $t+1$. Mathématiquement, le lissage exponentiel simple est défini comme suit.

$$\hat{x}_{t+1} = \alpha x_t + (1-\alpha)\hat{x}_t \tag{5.3}$$

Tels que :

- $\hat{x}_{t+1}$ désigne la prédiction à l'instant $t+1$;
- $\alpha \in]0,1[$ est appelé le coefficient de prédiction ;
- x_t représente l'observation à l'instant t ;
- $\hat{x}_t$ indique la prédiction à l'instant t.

Plus la valeur de α est proche de 0, plus la prédiction dépendra des observations du passé lointain. En revanche, plus α est proche de 1, plus la prédiction dépendra des observations du passé proche.

5.2.3.2 Prédiction du prochain score de qualité

Après ce rappel de l'outil mathématique que nous comptons utiliser, nous allons à présent montrer comment nous nous en servons pour prédire les moments de capture. De fait, nous nous intéressons à la prédiction du score de qualité à l'instant $t+1$ à partir de l'instant t. Autrement dit, on cherche à prédire le prochain score de qualité avant que le réseau atteigne le prochain instant. Ainsi, nous saurons à l'avance si $t+1$ sera un moment de capture ou non en comparant la différence des scores de qualité des deux instants successifs. Formellement, nous proposons la formule suivante pour prédire le prochain score de qualité :

$$\hat{P}_{t+1} = \gamma Q_t + (1-\gamma)\hat{P}_t \tag{5.4}$$

Tels que :

- $\hat{P}_{t+1}$ désigne la prédiction du score de qualité à l'instant $t+1$;
- $\gamma \in]0,1[$ est le coefficient de prédiction ;
- Q_t indique le score de qualité observé à l'instant t et correspond alors à $\psi_G(\mathcal{N}_t)$;
- $\hat{P}_t$ représente la prédiction du score de qualité à l'instant t.

Avec cette formule, on peut alors fixer un premier instant de capture à partir duquel les autres instants seront déduits. Le premier instant représente l'état initial du réseau. Dans l'optique de réduire les erreurs de prédiction dues à l'initialisation du processus ou à l'irrégularité des valeurs, nous avons introduit une seconde série temporelle qui cherche à fournir de meilleures prédictions en se basant sur l'historique des erreurs de prédiction. L'idée est de faire des prédictions, de mesurer les erreurs et d'apprendre de celles-ci pour corriger les premières.

5.2.3.3 Correction de la prédiction

La correction de la prédiction consiste à minimiser davantage l'erreur de prédiction de sorte que la valeur prédite soit la plus proche de la valeur observée. Pour corriger la prédiction, notre méthode calcule l'erreur de prédiction qui est définie par la différence entre le score de qualité observé et le score prédit. Ensuite, notre méthode prédit la prochaine erreur de prédiction en utilisant une série temporelle basée sur l'historique des erreurs de prédiction. Formellement, la série temporelle qui prédit l'erreur de prédiction est définie comme suit :

$$\hat{L}_t = \mu \hat{E}_{t-1} + (1-\mu) L_t \tag{5.5}$$

Tels que :

- $\hat{L}_t$ désigne l'erreur de prédiction estimée à l'instant $t+1$;
- $\mu \in]0,1[$ est le coefficient de l'erreur prédite ;
- $\hat{E}_{t-1}$ représente l'erreur de prédiction à l'instant t, définie par $Q(t) - P(t)$;
- L_t indique l'erreur de prédiction tableau à l'instant t.

Après avoir prédit l'erreur de prédiction, nous considérons que la prédiction corrigée est égale au score prédit additionné par l'erreur prédite (si la valeur prédite est plus petite que celle observée) ou bien l'erreur prédite soustraite du score prédit (si la valeur prédite est supérieure à celle observée).

En somme, rappelons que l'objectif de notre méthode est de prédire l'instant à partir duquel on doit capturer un nouvel instantané. Notre méthode de prédiction fonctionne en deux phases, à savoir :

1. Phase d'apprentissage : durant cette phase, on cherche à accumuler de l'historique. A ce stade, les prédictions ne sont pas aussi pertinentes ;
2. Phase d'exploitation : avec le temps, on commence à avoir de l'historique. Par conséquent, la prédiction devient plus bonne vu qu'elle dépend de la grandeur de l'historique. A ce niveau, la différence entre le score prédit et le score observé devient très minime. De ce fait, on pourrait faire confiance aux résultats prédits à l'instant $t+1$ avant même que le réseau atteigne cet instant.

5.3 Suivi de l'évolution de la structure des communautés

Notre algorithme de suivi consiste à identifier la nature d'évolution de communautés entre deux instantanés successifs. Pour ce faire, nous définissons tout d'abord les hypothèses sur lesquelles nous nous sommes appuyés.

Hypothèse 2

Le nœud d'intérêt est un élément indispensable pour sa communauté et à partir duquel elle est définie et construite. Autrement dit, on ne peut pas avoir une communauté ego-centrée sans nœud d'intérêt.

Hypothèse 3

Chaque communauté ego-centrée comprend strictement un seul nœud d'intérêt.

Hypothèse 4

Les nœuds d'intérêt sont connus au début de l'analyse du réseau. Ils sont donc statiques. De ce fait, on ne peut pas avoir d'autres nœuds d'intérêt qui apparaissent dans le réseau au fil du temps.

Dans la suite, nous définissons un ensemble de règles de correspondance qui indiquent les types des changements. Nous distinguons les changements microscopiques de ceux macroscopiques. En outre, alors que les travaux existants se limitent à interpréter les changements topologiques, notre solution permet d'introduire l'aspect d'intensité de communication dans les types des changements.

Dans cette perspective, nous présentons les types des changements dans les sections 5.3.1 et 5.3.2 avant de présenter dans la section 5.3.3 l'apport de notre solution par rapport à l'existant.

5.3.1 Types d'évolution microscopique

Les changements microscopiques concernent les modifications relatives aux nœuds/liens. Le tableau 5.1 présente huit types d'évolution ayant un fort impact sur la structure des communautés ou leur évolution au fil du temps.

Tableau 5.1 – Types d'évolution des nœuds/liens.

Type d'évolution	Description
Apparition de nœud	Un nœud intègre le réseau s'il reçoit une communication de la part de l'un des nœuds du réseau
Disparition de nœud	Un nœud disparait du réseau si sa durée d'apparition s'achève
Apparition de lien	Un lien apparait si c'est la première fois que deux nœuds du réseau commencent à échanger au cours d'une fenêtre temporelle
Disparition de lien	Un lien disparait du réseau si la communication entre les nœuds s'achève ou si l'un des nœuds disparait
Croissance de lien	Un lien marque une croissance si ses deux extrémités communiquent plus entre eux à l'instant $t+1$ qu'à l'instant t
Contraction de lien	Un lien présente une contraction si ses deux extrémités communiquent plus entre eux à l'instant t qu'à l'instant $t+1$
Monotonie de lien	Un lien est dit monotone si ses deux extrémités gardent la même fréquence d'échange de l'instant t à l'instant $t+1$
Résurgence de lien	Un lien resurgit s'il disparait de l'instant t puis réapparait à un instant t' de sorte que t et t' ne soient pas successifs

5.3.2 Types d'évolution macroscopique

Les types d'évolution macroscopique concernent les changements que les communautés subissent au fil du temps. Le tableau 5.2 donne un aperçu des différents types d'évolution que notre algorithme interprète.

En effet, nous considérons qu'une communauté peut disparaitre dans deux cas :

1. si elle n'est plus séparée du reste du réseau ou bien son degré d'isolation n'est plus significatif. L'intuition derrière est qu'une communauté, par définition, doit être cohésive et bien séparée du reste du réseau. Ainsi, si cette contrainte n'est plus remplie, nous supposons que la communauté a disparu.
2. si le nœud d'intérêt de la communauté disparait. Ce raisonnement est guidé par l'hypothèse 2 qui considère que toute communauté ego-centrée est définie sur la base de son nœud d'intérêt. Par conséquent, si ce dernier disparait, la communauté n'est plus ego-centrée, elle devient plutôt plate (n'ayant pas de centre).

Tableau 5.2 – Types d'évolution des communautés.

Type d'évolution	Description
Disparition de communauté	Une communauté C qui existait à l'instant t disparait de l'instant $t+1$ dans deux cas : - si les membres de C deviennent plus connectés à l'extérieur qu'à l'intérieur de C - si le nœud d'intérêt de C disparait
Réapparition de communauté	Une communauté qui existait à un instant précédent disparait de l'instant actuel et réapparait à l'instant prochain
Croissance de communauté	Croissance de densité : augmentation du nombre de nœuds/liens entre deux instants
	Croissance d'intensité de communication : augmentation de la fréquence d'échange entre les nœuds d'un instant à l'autre
Rétrécissement de communauté	Rétrécissement de densité : diminution du nombre de nœuds/liens entre deux instants
	Rétrécissement de cohésion : diminution de la fréquence d'échange entre les nœuds d'un instant à l'autre
Monotonie de communauté	Monotonie de cohésion : constance du nombre de nœuds/liens entre deux instants
	Monotonie d'intensité de communication : constance de la fréquence d'échange entre les nœuds d'un instant à l'autre
Résurgence de communauté	la communauté disparait à un moment donné, puis réapparait quelques temps plus tard sous une forme identique ou très proche

Pour mesurer le degré d'isolation d'une communauté, noté d_{Sepa}, nous prenons en compte le nombre et le poids des liens à l'intérieur de C et à l'extérieur. Ainsi, nous représentons l'intensité de communication entrante $\sum_{w_C^{in}}$ par la somme des poids des liens dont les deux extrémités se trouvent dans C et nous représentons l'intensité de communication sortante $\sum_{w_C^{out}}$ par la somme des poids des liens dont une seule extrémité appartient à C. Nous considérons que l'intensité des liens internes/externes égale au nombre de ces liens multiplié par la somme des poids. Ainsi, nous divisons la proportion d'intensité entrante par la proportion d'intensité totale :

$$d_{Sepa}(C) = \frac{|E_C| \sum_{w_C^{in}}}{|E_C| \sum_{w_C^{in}} + |E_C| \sum_{w_C^{out}}} \tag{5.6}$$

La valeur de d_{Sepa} varie entre 0 (une communauté fortement liée au reste du réseau) et 1 (une communauté totalement isolée du reste du réseau).

Hypothèse 5

Une communauté C est considérée séparée du reste du réseau si ses nœuds communiquent entre eux autant qu'avec l'extérieure du réseau. Autrement dit, si la proportion de communication entrante est strictement supérieure à la proportion de communication sortante. Formellement, si le degré d'isolation est supérieure à la moitié $d_{Sepa}(C) > 0.5$.

De plus, il convient de noter que la résurgence est différente de la réapparition même si elles semblent être similaires. La différence entre les deux peut être expliquée comme suit :

1. la communauté source et celle ayant resurgi n'apparaissent pas sur deux instantanés successifs. Il

faut qu'elles soient séparées par plusieurs instantanés dans lesquels ne figure pas la communauté qui va resurgir ;

2. la résurgence donne lieu à une communauté identique ou très proche de la communauté source. Par contre, la réapparition d'une communauté peut être sous une forme très différente de la communauté source.

En quelques sorte, nous pouvons dire que la résurgence peut être vue comme une apparition suivie de plusieurs disparitions et une réapparition. La résurgence permet de modéliser l'apparition de communautés saisonnières.

L'algorithme 5 présente l'implémentation des types d'évolution présentés sur le tableau 5.2. Cet algorithme fonctionne comme suit : étant donné deux instantanés successifs $\mathcal{N}_t$ et $\mathcal{N}_{t+1}$, il cherche à trouver une correspondance entre les communautés de ces instantanés. Pour ce faire, il utilise un ensemble de règles de correspondance pour identifier l'évolution de communautés. Ces règles sont appliquées à chaque communauté et décrivent la manière dont les types d'évolution sont définis.

De la ligne 3 à 6 de l'algorithme 5, nous voyons la règle correspondant à la disparition de communauté. D'après cette règle, une communauté est considérée disparue si ses membres deviennent plus attachés au reste du réseau qu'à l'intérieur de la communauté ($d_{Sepa}(C_u^{\mathcal{N}_t}) > 0.5$) ou bien si le noeud d'intérêt disparait de la communauté ($u \notin C_u^{\mathcal{N}_t}$). Ensuite, nous trouvons la règle de croissance de densité (lignes 7-10), une communauté devient plus dense si elle comprend plus de liens qu'avant ou si elle garde le même nombre de liens et perd quelques nœuds isolés [1]. Le contraire de l'opération précédente désigne le rétrécissement de densité (lignes 15-18). Une communauté devient moins dense si le nombre de liens diminuent ou si le nombre de liens ne change pas mais le nombre de nœuds isolés s'incrémente. Aux lignes 11-14, nous trouvons la règle de croissance d'intensité de communication qui indique que les nœuds à l'instant actuel communiquent plus entre eux qu'à l'instant précédent. L'opération inverse signifie naturellement la diminution de fréquence d'échange de la communauté (lignes 19-22). Si la communauté garde la même structure (lignes 23-26) ou garde la même fréquence d'échange entre les nœuds (lignes 27-30), on parle de monotonie topologique ou comportementale (relative à l'intensité de communication entre les nœuds). La dernière règle de correspondance est celle de la résurgence (lignes 31-34), ce type d'évolution signifie tout simplement que la communauté C a apparu sur deux instantanés non successifs. Autrement dit, la communauté était présente à l'instantané $\mathcal{N}_t$, puis elle a disparu sur un ensemble d'instantanés $\mathcal{N}_{t+i,\ 1\leqslant i<j}$. Ensuite, elle est réapparue sur l'instantané $\mathcal{N}_{t+j}$.

5.3.3 Discussion sur les types d'évolution

La première chose que nous constatons dans notre algorithme est l'absence de deux types d'évolution très fréquemment abordés dans la littérature, à savoir, la division et la fusion de communautés.

En effet, la division de communauté ne peut pas être appliquée dans notre contexte d'étude. La raison est que si une communauté se divise en deux sous-communautés ego-centrées, nous ferons face à l'une de deux possibilités :

1. soit considérer que les communautés résultantes n'ont pas d'ego. Ce qui contredit l'hypothèse 2 ;
2. soit élire pour chaque sous-communauté un ego correspondant. Cela contredit également l'hypothèse 5.

1. Un nœud est dit isolé s'il n'a pas de voisin.

Algorithme 5 Pseudocode de l'algorithme de suivi de l'évolution de communautés ego-centrées.

Entrée: Des instantanés du réseau dynamique
Sortie: Une interprétation de l'évolution de communautés ego-centrées au fil des instantanés

Données
$\mathcal{N}_t$: un réseau dynamique, orienté et pondéré,
u : un nœud d'intérêt,
$\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, ..., \mathcal{N}_n$: des captures du réseau dynamique $\mathcal{N}_t$ aux différents instants,
$C_u^{\mathcal{N}_t}$: la communauté ego-centrée de u à l'instant t,
$|V_{C_u^{\mathcal{N}_t}}|$: nombre de nœuds de la communauté de u à l'instant t,
$|E_{C_u^{\mathcal{N}_t}}|$: nombre de liens de la communauté de u à l'instant t,
$\sum w(C_u^{\mathcal{N}_t})$: somme des poids des liens de la communauté de u à l'instant t,
i et j : deux entiers naturel positifs.

Pour tout $u \in \mathcal{N}_t$ **Faire**

  Si $d_{Sepa}(C_u^{\mathcal{N}_t}) > 0.5 \mid u \notin C_u^{\mathcal{N}_t}$ **Alors**

    Afficher « La communauté C a disparu. »
  Fin Si
  Si $|E_{C_u^{\mathcal{N}_{t+1}}}| > |E_{C_u^{\mathcal{N}_t}}| \mid (|E_{C_u^{\mathcal{N}_{t+1}}}| = |E_{C_u^{\mathcal{N}_t}}| \ \&\ |V_{C_u^{\mathcal{N}_{t+1}}}| < |V_{C_u^{\mathcal{N}_t}}|)$ **Alors**

    Afficher « La communauté C est devenue plus dense. »
  Fin Si
  Si $\sum w(C_u^{\mathcal{N}_{t+1}}) > \sum w(C_u^{\mathcal{N}_t})$ **Alors**

    Afficher « Croissance de C en terme de fréquence d'échanges entre les nœuds. »
  Fin Si
  Si $|E_{C_u^{\mathcal{N}_{t+1}}}| < |E_{C_u^{\mathcal{N}_t}}| \mid (|E_{C_u^{\mathcal{N}_{t+1}}}| = |E_{C_u^{\mathcal{N}_t}}| \ \&\ |V_{C_u^{\mathcal{N}_{t+1}}}| > |V_{C_u^{\mathcal{N}_t}}|)$ **Alors**

    Afficher « La communauté C est devenue moins dense. »
  Fin Si
  Si $\sum w(C_u^{\mathcal{N}_{t+1}}) < \sum w(C_u^{\mathcal{N}_t})$ **Alors**

    Afficher « Rétrécissement de C en terme de fréquence d'échanges entre les nœuds. »
  Fin Si
  Si $|E_{C_u^{\mathcal{N}_{t+1}}}| = |E_{C_u^{\mathcal{N}_t}}|$ **Alors**

    Afficher « La cohésion de communauté C n'a pas changé. »
  Fin Si
  Si $\sum w(C_u^{\mathcal{N}_{t+1}}) = \sum w(C_u^{\mathcal{N}_t})$ **Alors**

    Afficher « La fréquence d'échanges entre les nœuds de C n'a pas changé. »
  Fin Si
  Si $C_u \in \mathcal{N}_t \ \&\ C_u \notin \mathcal{N}_{t+i,\ 1 \leqslant i < j} \ \&\ C_u \in \mathcal{N}_{t+j}$ **Alors**

    Afficher « La communauté C a resurgit. »
  Fin Si
Fin Pour

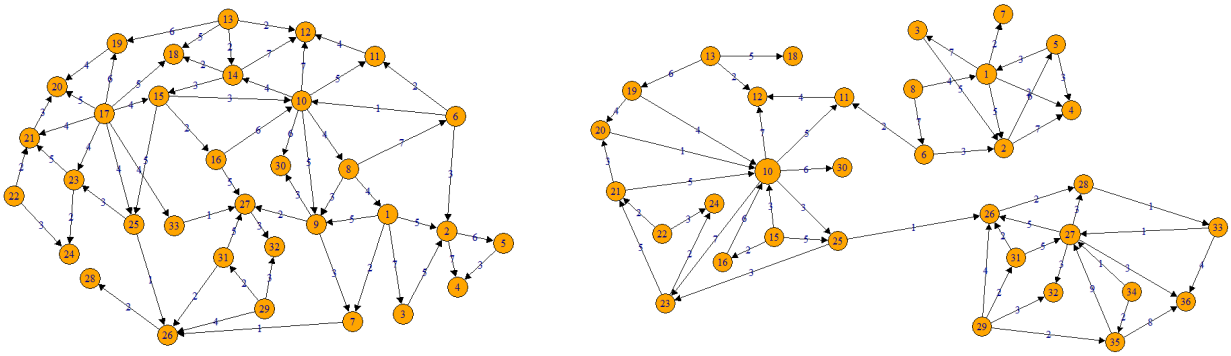
En outre, la fusion de deux communautés ego-centrées produit une nouvelle communauté ayant deux egos. Or, selon l'hypothèse 3, une communauté est dite ego-centrée si elle est définie sur la base d'un seul ego. En bref, la leçon à retenir est que les types d'évolution de communautés ne se comportent pas toujours de la même manière. Elles dépendent du contexte d'étude et des hypothèses considérées.

Il convient de noter également que les travaux existants se limitent à l'interprétation des changements topologiques. Pourtant, une communauté peut grandir ou rétrécir sans que sa structure topologique change (nombre de nœuds/liens ou direction de liens). C'est dans ce cadre que nous avons décidé d'intégrer l'aspect d'intensité de communication dans les types d'évolution de communautés. Cela constitue l'un des atouts de notre solution. Pour chacune des opérations suivantes : la croissance, le rétrécissement et la monotonie, nous avons distingué deux types d'évolution dont l'un correspond aux changements topologiques et l'autre concerne le changement de fréquence d'échanges entre les nœuds.

Enfin, contrairement aux solutions existantes, notre interprétation permet non seulement de préciser la classe de l'évolution de communauté mais aussi, le type de changements en termes de nœuds et de liens. Cela nous permettra d'expliquer les différents facteurs ayant causé l'évolution. Ces facteurs sont d'une importance capitale car l'évolution de communautés n'est pas toujours menée par les aspects topologiques mais aussi par le comportement d'un nœud vis-à-vis aux autres au cours d'une fenêtre temporelle.

5.3.4 Illustration du suivi de l'évolution de communautés dynamiques

L'objectif de cette section est de présenter une illustration de notre méthode de suivi de l'évolution de communautés dynamiques. Soient deux instantanés capturés respectivement à l'instantané t et $t+1$, représentés sur la Figure 5.2. L'instantané t constitue l'état initial du réseau et dont les communautés sont présentées sur la Figure 5.3.



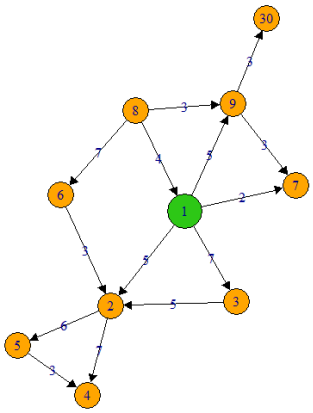
(a) Structure du réseau dynamique à l'instant t.

(b) Structure du réseau dynamique à l'instant $t+1$.

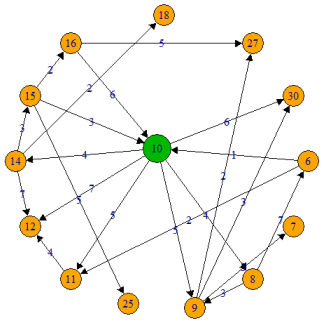
FIGURE 5.2 – Deux instantanés successifs d'un réseau dynamique.

D'après la Figure 5.2, nous constatons que les changements microscopiques (relatifs aux nœuds/liens) qui ont eu lieu à l'instant $t+1$ sont :

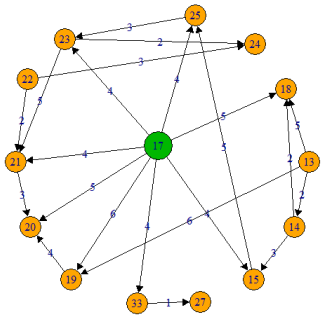
- la disparition de 3 nœuds : 9, 14 et 17.
- la disparition des liens (16, 27) et (10, 8) ainsi que tous les liens qui étaient adjacents aux nœuds disparus.
- l'apparition de 3 nœuds : 34, 35 et 36.



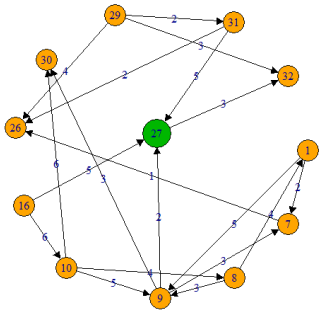
(a) Communauté ego-centrée du nœud 1.



(b) Communauté ego-centrée du nœud 10.



(c) Communauté ego-centrée du nœud 17.



(d) Communauté ego-centrée du nœud 27.

FIGURE 5.3 – Communautés ego-centrées en deux étapes détectées sur le réseau à l'instant t

- l'apparition de 16 liens : (5,1) ; (1,4) ; (27,26) ; (29,35) ; (27,28) ; (28,33) ; (33,36) ; (35,36) ; (34,35) ; (27,36) ; (35,27) ; (34,27) ; (20,10) ; (10,23) ; (21,10) ; (19,10).

Quant aux changements macroscopiques, après avoir appliqué notre algorithme de suivi, il détecte trois communautés et indique cinq types d'évolution :

1. la contraction de $C_{\{a,2\}}$ après avoir perdu 2 nœuds : 9 et 30.
2. la croissance de $C_{\{a,2\}}$ après avoir intégré deux nouveau liens (5,1) et (1,4). Ainsi, $C_{\{a,2\}}$ ={1,2,3,4,5,6,7,8}.
3. la mort de la communauté $C_{\{c,2\}}$.
4. la croissance de $C_{\{b,2\}}$ après avoir intégré les nœuds qui étaient dans la communauté de $C_{\{c,2\}}$. Par conséquent, $C_{\{b,2\}}$ ={10,11,12,13,15,16,18,19,20,21,22,23,24,25,30}.
5. la croissance de $C_{\{d,2\}}$ grâce à l'arrivée des nouveau nœuds et liens. De ce fait $C_{\{d,2\}}$ ={26,27,28,29,31,32,33,34,35,36}.

Nous montrons sur la Figure 5.4 les communautés détectées à l'instantané $t+1$.

5.4 Conclusion

Dans ce chapitre, nous avons proposé une solution qui répond à trois problématiques fondamentales, à savoir :

1. L'évaluation de l'état d'un réseau dynamique à un instant donné ;
2. La prédiction de l'instant à partir duquel on doit capturer un nouvel instantané ;

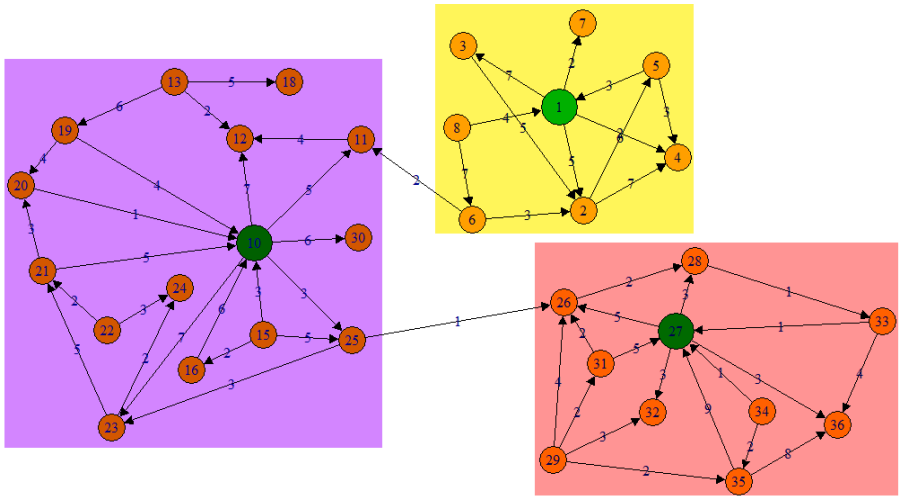


FIGURE 5.4 – Communautés ego-centrées en deux étapes détectées à l'instant $t + 1$.

3. le suivi de l'évolution de communautés dynamiques au fil du temps.

Pour répondre à la première problématique, nous avons proposé une variante de la fonction de qualité présentée dans le précédent chapitre (section 4.2.2.1). Cette variante permet d'évaluer l'état d'un réseau dynamique à un instant donné tout en prenant en compte la direction des liens, leur nombre et leur poids. Concernant la seconde problématique, nous avons présenté une nouvelle stratégie qui se base sur les séries temporelles pour prédire l'instant de capture de la prochaine fenêtre temporelle. Pour ce qui est de la troisième problématique, nous avons proposé un algorithme qui utilise des règles d'association pour interpréter l'évolution de communautés dynamiques au fil du temps.

Troisième partie

Validation

Chapitre 6

Validation de la solution proposée

Après avoir présenté notre solution, nous consacrons ce chapitre à la validation de notre approche de détection de communautés qui comprend :

1. Deux algorithmes de détection de communautés statiques ego-centrées ;
2. Une méthode de gestion de fenêtre temporelle ;
3. Un algorithme de suivi de l'évolution de communautés dynamiques.

La suite de ce chapitre est organisée comme suit : tout d'abord, nous présentons les outils d'implémentation dans la section 6.1. Puis, nous évaluons l'efficacité des algorithmes de détection de communautés statiques dans la section 6.2. Ensuite, nous expérimentons la méthode de gestion de fenêtre temporelle dans la section 6.3. Enfin, nous présentons les résultats de notre algorithme de suivi de communautés dynamiques dans la section 6.4.

6.1 Outils d'implémentation

Nous avons implémenté notre solution en utilisant les outils décrits ci-dessous.

R R est un logiciel dédié au traitement et à l'analyse des données. Il est libre, gratuit et multiplateforme (Il fonctionne sur Windows, Linux et Mac OS X). R est utilisé dans de nombreuses disciplines (biologie, écologie, finances, sciences sociales, etc). Il bénéficie d'un grand nombre d'extensions (plus de dix milles actuellement) développées par la communauté des utilisateurs et permettent d'ajouter des fonctionnalités à celles proposées par défaut.

RStudio RStudio est un environnement de développement intégré conçu spécialement pour R. Il est disponible en deux versions, à savoir, RStudio Desktop, pour une exécution locale du code, et RStudio Server qui fonctionne sur un serveur Linux et permet d'accéder à RStudio par un navigateur web. Naturellement, RStudio est libre, gratuit et multiplateforme. Son objectif est de combler le manque des fonctionnalités de R, citons par exemple, la coloration syntaxique, l'autocomplétion, le débogage, la création de packages, le contrôle de version sur Git, etc.

RFacebook RFacebook est un package R permettant d'extraire les données des utilisateurs sur Facebook. Ces données comprennent les informations personnelles des utilisateurs (identifiant, photo de profil, date de naissance, etc) ainsi que les posts (texte, image, vidéo, etc) et les réactions aux posts. Il existe également d'autres packages R pour extraire des données à partir de Twitter [1] et Instagram [2].

1. Package twitterR :https://cran.r-project.org/web/packages/twitteR/index.html
2. Package instaR :https://cran.r-project.org/web/packages/instaR/index.html

network network est un package R permettant de créer et de manipuler les réseaux statiques. Il fait partie du projet statnet [3].

networkDynamic Conçu sur la base du package network, networkDynamic est un package R conçu spécialement pour la gestion des réseaux dynamiques. Cette gestion comprend la création, la manipulation, l'exportation et l'importation des réseaux.

NDTV Network Dynamic Temporal Visualizations (NDTV) est une extension du package networkDynamic. Elle permet de visualiser, en mode réel ou discret, les réseaux dynamiques.

intergraph intergraph est un package R permettant de convertir un objet de la classe network vers igraph et vice versa.

iGraph iGraph est une bibliothèque d'analyse des réseau sociaux statiques, disponible en trois langages : C, Python et R.

Le Tableau 6.1 présente le rôle des outils que nous avons utilisés.

Tableau 6.1 – Description des outils utilisés dans l'implémentation de notre solution.

Package	Référence	Rôle
R	(R. C. Team, 2013)	Plateforme de base de notre implémentation
RStudio	(R. Team, 2015)	Développement des scripts et analyse des données
RFacebook	(P. Barbera, Piccirilli et M. P. Barbera, 2017)	Extraction des données sur Facebook
networkDynamic	(Skye Bender-deMoll, 2016)	Manipulation du réseau dynamique
NDTV	(S Bender-deMoll, 2018)	Visualisation du réseau dynamique
network	(Butts, 2015)	Extraction des instantanés à partir du réseau dynamique
intergraph	(Bojanowski, 2015)	Conversion d'un objet *network* en *iGraph*
iGraph	(Csárdi, Nepusz et Airoldi, 2016)	Traitement et visualisation des instantanés

En effet, nous avons choisi Facebook comme application sociale. Nous nous sommes intéressés aux commentaires faits autour des publications d'un utilisateur donné. Pour ce faire, nous avons récupéré les données en utilisant le package RFacebook. Sur la base des données extraites par ce package, nous avons pu illustrer le principe de fonctionnement de notre modèle de représentation des données dans la section 4.1.2.

Notons que n'ayant pas le temps pour développer une application Facebook répondant aux exigences récentes de la politique de confidentialité des données, nous avons opté pour d'autres jeux de données qui répondent à nos besoins et les avons utilisés dans la validation.

Après avoir eu le réseau social dynamique, nous l'avons manipulé et visualisé grâce aux packages *networkDynamic* et *NDTV*. Ensuite, nous avons extrait des instantanés du réseau via le package

3. Software Tools for the Analysis, Simulation and Visualization of Network Data.

network. Ces instantanés sont des objets de la classe *network*. Mais, comme cette classe ne fournit pas les fonctionnalités[4] dont nous avons besoin, nous avons eu recours au package *intergraph* pour convertir les instantanés de *network* en *iGraph*. Grâce à ce dernier, nous avons pu implémenter nos algorithmes statiques et nous avons extrait l'ensemble de communautés ego-centrées.

6.2 Évaluation des algorithmes de détection de communautés statiques

Dans cette section, nous présentons tout d'abord les résultats de l'évaluation de l'algorithme de détection de communautés ego-centrées sur un voisinage direct. Ensuite, nous nous intéressons à l'évaluation de l'algorithme de détection de communautés ego-centrées sur un voisinage de longueur k.

6.2.1 Détection de communautés ego-centrées sur un voisinage direct

Pour évaluer l'efficacité de cet algorithme, nous décrivons tout d'abord dans la section 6.2.1.1 le jeu de données sur lequel l'algorithme est appliqué. Puis, nous comparons les quatre variantes de cet algorithme dans la section 6.2.1.2 pour en choisir la meilleure. Ensuite, dans la section 6.2.1.3, nous comparons la meilleure variante avec un algorithme d'extraction du voisinage direct. Enfin, nous montrons dans la section 6.2.1.4 un exemple illustratif de l'impact des algorithmes sur la structure de communautés.

6.2.1.1 Description du jeu de données

Nous expérimentons cet algorithme sur le réseau des Misérables (Knuth, 1993) qui comprend 77 nœuds et 254 liens (voir la figure 6.1). Les Misérables est un réseau non orienté, constitué de cooccurrence des personnages cités dans le roman des Misérables de Victor Hugo. Les nœuds représentent les personnages et les liens désignent les cooccurrences entre eux. Les liens sont munis de poids indiquant le nombre de cooccurrences des personnages dans le même chapitre du roman. Nous avons choisi ce réseau non orienté pour illustrer le bon fonctionnement de nos algorithmes sur les réseaux non orientés. Dans la section 6.2.2, nous expérimentons nos algorithmes sur un réseau orienté.

En effet, le réseau des Misérables comprend plusieurs communautés ego-centrées. Toutefois, comme le principe de l'algorithme reste le même quelle que soit la communauté ego-centrée sélectionnée, nous avons décidé de choisir la communauté de « Jean Valjean » vu sa prédominance.

6.2.1.2 Ajout par nœud vs. Ajout par bloc

Notre première expérience vise à comparer la manière dont se comportent nos algorithmes par rapport à la cohésion interne et à la séparation du reste du réseau. Rappelons que notre fonction qualité est définie sur la base de ces deux critères. Pour chacun, nous évaluons la sélection aléatoire et déterministe avec l'ajout par nœud et par bloc. Ainsi, nous obtenons quatre combinaisons, à savoir, Sélection Aléatoire avec Ajout par Nœud (SAAN), Sélection Déterministe avec Ajout par Nœud (SDAN), Sélection Aléatoire avec Ajout par Bloc (SAAB) et Sélection Déterministe avec Ajout par

4. Exemples : liste des nœuds, liste des liens, degré des nœuds, etc.

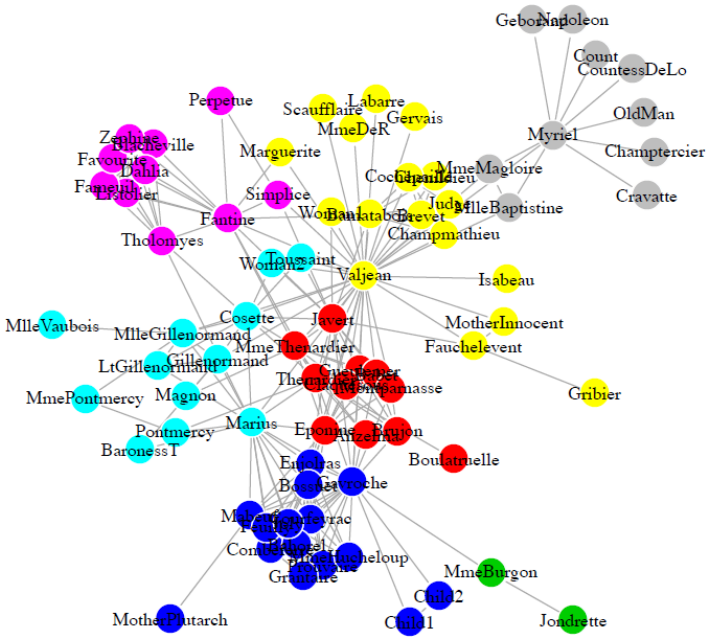


FIGURE 6.1 – Réseau des Misérables.

Bloc (SDAB). Comme les algorithmes basés sur la sélection aléatoire peuvent donner des scores de qualité différents pour chaque exécution, nous en avons effectué une série d'expériences, puis, nous avons calculé le score moyen de qualité. Quant aux algorithmes à base de bloc, nous considérons que la taille du bloc β égale à 5.

Dans la première partie de cette expérience, nous faisons varier la cohésion interne (densité de communauté) de 10% à 100%, puis, nous observons les valeurs correspondantes du score de qualité. La figure 6.2 présente les scores de qualité par rapport à la cohésion.

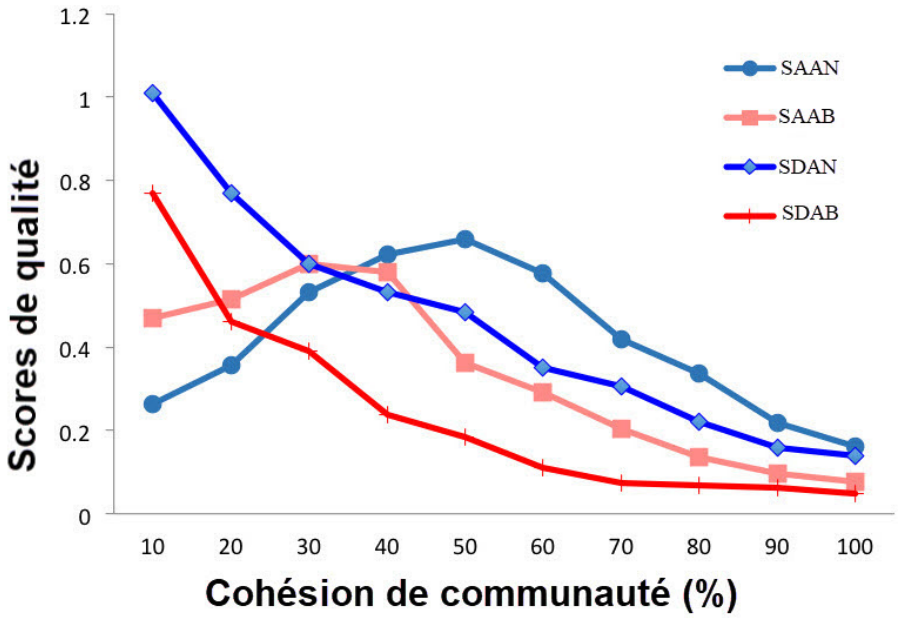


FIGURE 6.2 – Scores de qualité des quatre variantes par rapport à la cohésion interne.

Pour les quatre algorithmes, nous observons que les scores de qualité diminuent lorsque la cohésion

augmente. Cette observation prouve l'efficacité de nos stratégies d'optimisation de la fonction qualité comme sa valeur reste faible pour les partitions cohésives. Quant à la comparaison entre les quatre algorithmes, nous constatons que le score des algorithmes déterministes (SDAN, SDAB) diminue plus rapidement que ceux à base aléatoire (SAAN, SAAB). Ceci est dû au fait que les nœuds pertinents (qui font augmenter la cohésion de communauté et donc baisser le score de qualité) sont les premiers à choisir lors des sélections déterministes, alors que les sélections aléatoires n'exigent pas un ordre de passage de nœuds suivant leur pertinence, elles les sélectionnent plutôt au hasard. Nous remarquons également que la variante SDAB est plus efficace que SDAN car au lieu de faire l'ajout par nœud, la première le fait par bloc. Ainsi, au cours des itérations, les blocs ajoutés diminuent plus fortement le score de qualité.

Par analogie, nous effectuons une autre expérience afin d'évaluer l'optimisation de la fonction de qualité par rapport à la séparation de communauté. Dans cette perspective, nous varions le degré d'isolation de communauté de 10% (isolation faible, c'est-à-dire que les nœuds de la communauté communiquent plus avec l'extérieure du réseau qu'à l'intérieur de communauté) à 100% (isolation totale, c'est-à-dire que les nœuds de la communauté communiquent uniquement entre eux). Le degré d'isolation d'une communauté est calculé via la métrique d_{Sepa} définie dans la section 5.4.2.

Nous observons sur la figure 6.3 les mêmes tendances vues précédemment. Autrement dit, la SDAB surpasse les autres variantes.

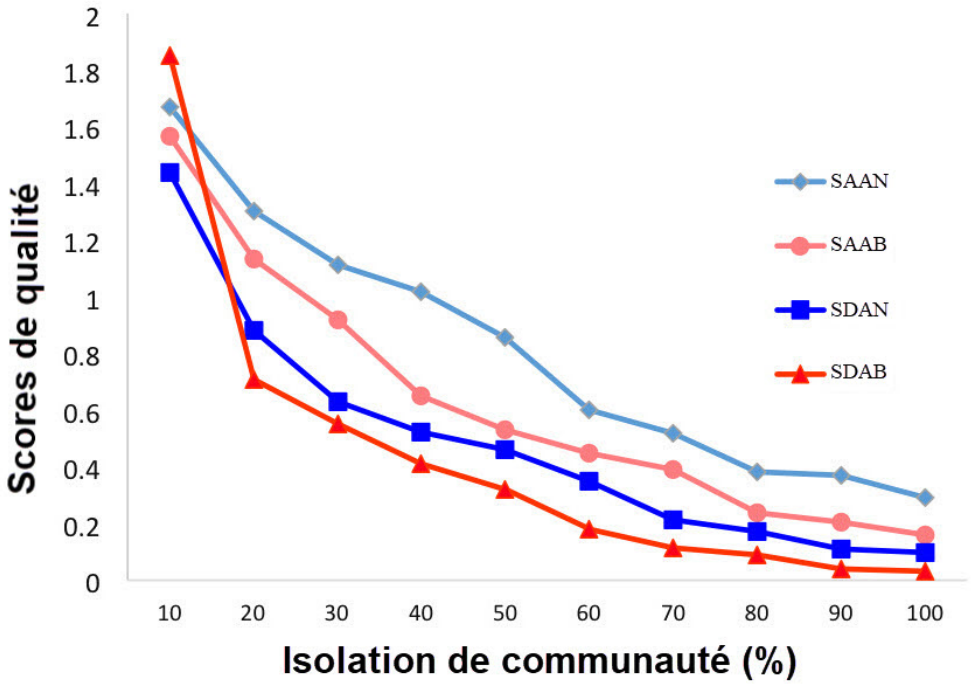


FIGURE 6.3 – Scores de qualité des quatre variantes par rapport à la séparation du reste du réseau.

En somme, il s'avère que la SDAB est la meilleure variante de l'algorithme par rapport à la cohésion interne et à la séparation du reste du réseau. Dans la suite, nous comparons la SDAB avec l'algorithme d'extraction du voisinage direct.

6.2.1.3 Efficacité de notre solution

Dans cette partie, nous comparons l'efficacité de SDAB avec celle de l'algorithme d'Extraction du Voisinage Direct (EVD) qui considère que la communauté est constituée de tous les voisins directs du nœud d'intérêt. Nous comparons SDAB à EVD suivant la cohésion interne et la séparation du reste du réseau. La figure 6.4 et 6.5 montrent les scores de qualité des deux algorithmes.

Sur la figure 6.4, nous observons que plus la cohésion augmente, plus le score de SDAB diminue plus rapidement que celui d'EVD. Ceci s'explique par le fait que SDAB filtre le voisinage direct en n'ajoutant que les blocs de nœuds pertinents, soient ceux qui diminuent plus le score de qualité. Par contre, EVD ajoute tous les nœuds du voisinage sans exception.

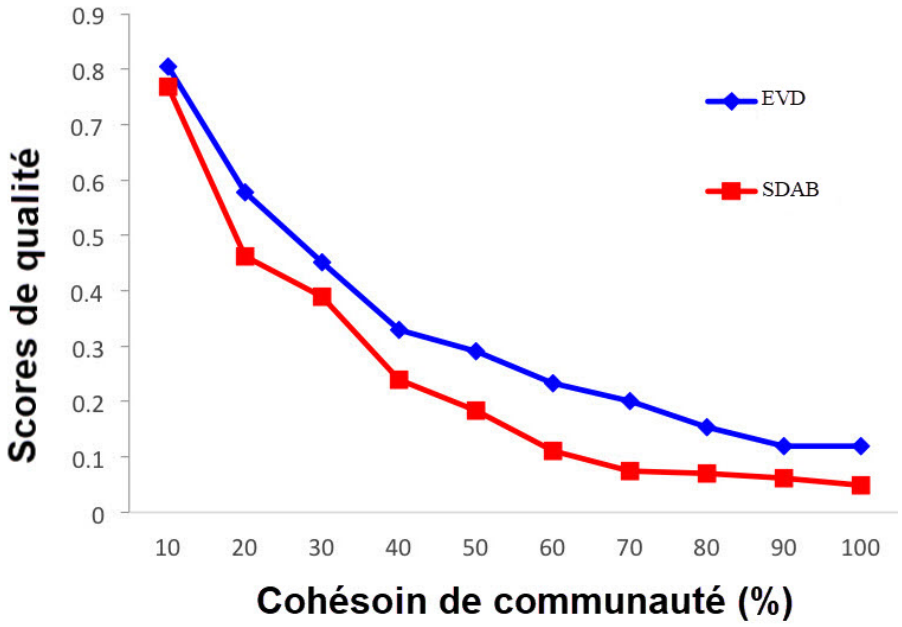


FIGURE 6.4 – Scores de qualité de SDAB et EVD par rapport à la cohésion interne.

Sur la figure 6.5, les scores de qualité correspondent aux variations du degré d'isolation de communauté. Nous constatons que le score d'EVD est meilleur si le degré d'isolation est inférieur à 50%. La raison en est que SDAB a tendance à rejeter les nœuds par bloc si leur taux de communication avec l'ego est faible tandis qu'EVD les inclut tous. Toutefois, nous voyons que si l'isolation dépasse 50%, SDAB se comporte mieux. Ceci est dû au fait que les nœuds ayant un taux de communication plus élevé avec la communauté sont ajoutés par bloc alors que les autres nœuds sont rejetés aussi par bloc.

6.2.1.4 Impact des algorithmes sur la structure de communauté

Pour montrer la pertinence de notre algorithme par rapport à la structure de communauté résultante, nous nous focalisons sur la communauté du personnage « Jean Valjean » du réseau représenté sur la figure 6.1. Ensuite, nous extrayons la communauté en utilisant EVD et SDAB.

La figure 6.6 montre la communauté de Valjean extraite par EVD et SDAB. Afin que les figures 6.6a et 6.6b soient lisibles, nous n'avons pas affiché ni la direction ni le poids des liens. Pour SDAB, nous avons utilisé un seuil limité à 0.65 pour réduire la taille de communauté.

Sur la figure 6.6b, nous constatons l'absence des nœuds faiblement liés à la communauté, comme

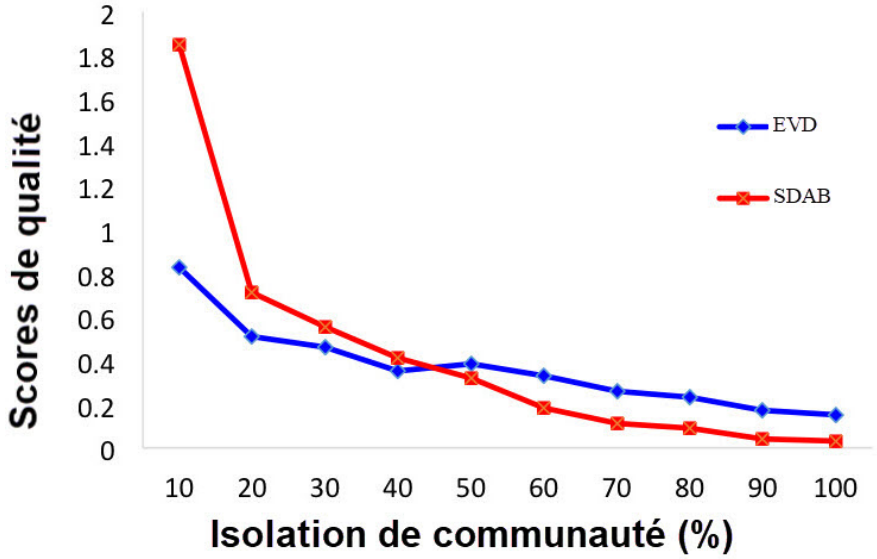
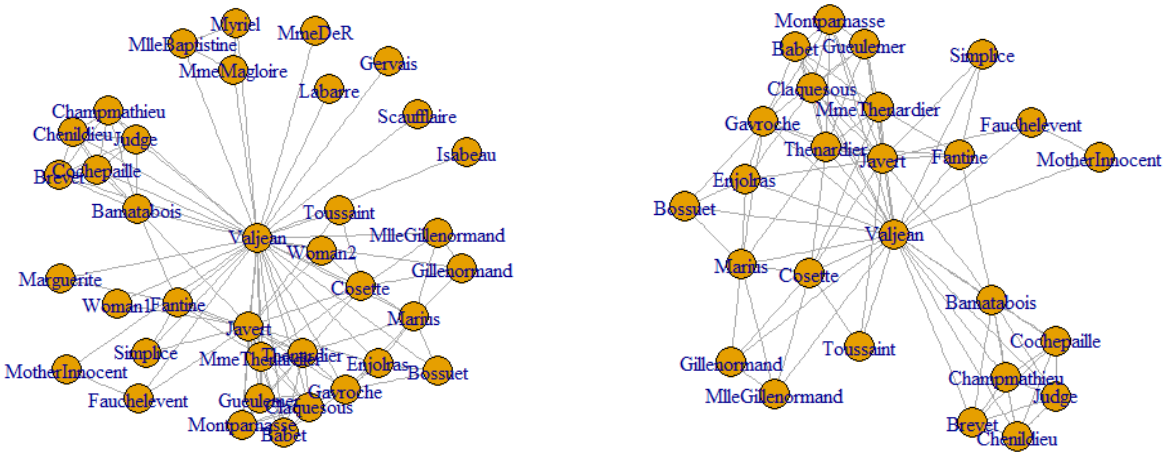


FIGURE 6.5 – Scores de qualité de SDAB et EVD par rapport à la séparation du reste du réseau.



(a) Communauté ego-centrée de Valjean extraite par EVD.

(b) Communauté ego-centrée de Valjean détectée par SDAB.

FIGURE 6.6 – Communauté ego-centrée de Valjean.

par exemple, Gervais, Isabeau, Labarre, Scaufflaire et MmeDeR.

En conclusion, la variante SDAB cherche à filtrer le voisinage direct tout en prenant en compte la cohésion interne et l'intensité de communication. Ainsi, une communauté ne signifie pas forcément le voisinage, elle en est plus pertinente.

6.2.2 Détection de communautés ego-centrées sur un voisinage de longueur k

Dans cette section, nous nous intéressons à la détection de communautés ego-centrées au-delà du voisinage direct. Pour ce faire, nous décrivons tout d'abord le jeu de données dans la section 6.2.2.1. Ensuite, nous présentons l'efficacité de notre algorithme dans la section 6.2.2.2. Enfin, nous illustrons l'impact de l'algorithme sur la structure de communautés dans la section 6.2.2.3.

6.2.2.1 Description du jeu de données

Le jeu de données que nous utilisons cette fois-ci est un réseau d'amitié des adolescents (*Adolescent health network dataset – KONECT* 2016) réalisé par Add Health [5] et comprend 2539 nœuds et 12969 liens. La figure 6.7 présente une illustration du réseau. Pour des raisons de lisibilité, nous n'avons pas affiché ni la direction de liens ni leur poids.

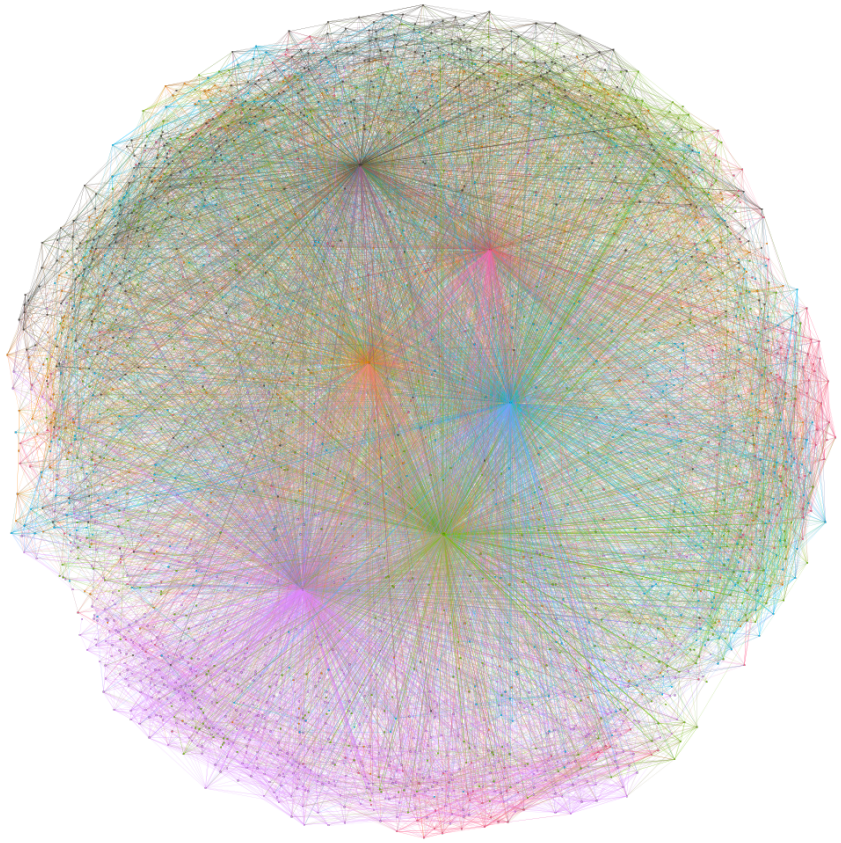


FIGURE 6.7 – Réseau d'amitié des adolescents.

Ce jeu de données a été créé lors d'une enquête qui a eu lieu entre 1994 et 1995 dans le cadre d'une étude socio-démographique menée sur un échantillon national des écoles secondaires publiques et privées aux États-Unis. Cette étude comprend un entretien à domicile de 90 minutes et vise à améliorer la santé des adolescents aux États-Unis. L'étude a été réalisée par Add Health dans le but de décrire le pourcentage des adolescents qui déclarent annuellement des soins de santé abandonnés et l'influence des facteurs socio-démographiques [6] sur la santé des adolescents. La procédure de collecte des données a été faite comme suit : chaque élève énumère ses cinq meilleurs amis des deux sexes. Les nœuds représentent les étudiants et les liens désignent que l'étudiant de gauche a choisi celui de droite comme ami. Plus la valeur du poids augmente, plus les élèves interagissent entre eux.

Comme nous pouvons remarquer sur la figure 6.7, il existe six principaux voisinages ego-centrés, nous nous intéressons au voisinage du nœud dont le label est « 1 » [7] en raison de sa prédominance. Nous travaillons sur un voisinage de longueur 2 ($k = 2$) dans les expériences que nous présentons dans la suite.

5. The National Longitudinal Study of Adolescent to Adult Health.
6. Exemples : le statut d'assurance, les soins de santé antérieurs et les risques/comportements sanitaires, etc.
7. Il s'agit du voisinage, coloré en violet qui se trouve en bas sur la figure 6.7.

6.2.2.2 Efficacité des algorithmes

Pour détecter des communautés ego-centrées sur un voisinage de longueur 2, nous utilisons la variante Sélection Déterministe avec Retrait par Bloc (SDRB)-2. Dans la suite, nous comparons SDRB-2 avec SDAB par rapport à la cohésion interne et à la séparation du reste du réseau.

Sur la figure 6.8, nous remarquons que plus la cohésion augmente, plus les scores de qualité diminuent. Ceci s'explique par le fait que plus on se rapproche de la cohésion totale (sous-graphe complet), plus la majorité des nœuds devient pertinente en termes de liaisons topologiques. Cependant, nous remarquons que la qualité de l'algorithme SDRB-2 est toujours meilleure que celle de SDAB. La différence de qualité entre les deux courbes peut être interprétée comme suit : comme la communauté ego-centrée sur un voisinage de longueur 2 comprend plus de nœuds que celle sur un voisinage direct, la qualité de la première sera toujours plus bonne en augmentant la cohésion.

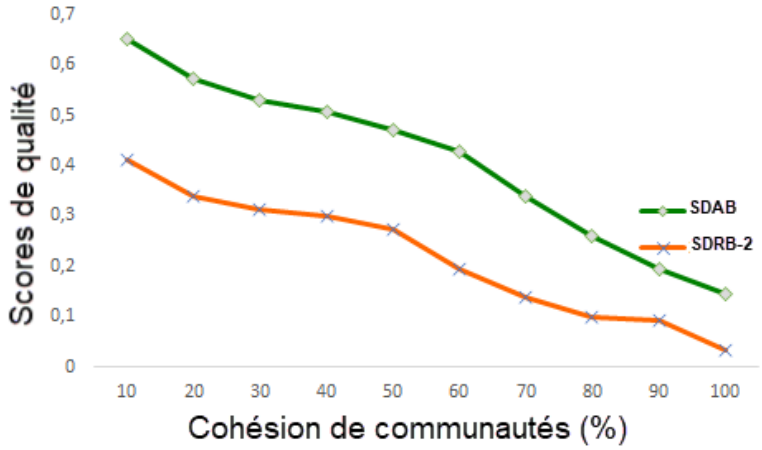


FIGURE 6.8 – Scores de qualité de SDAB et SDRB-2 par rapport à la cohésion interne.

Pour comparer SDAB et SDRB-2 suivant la séparation du reste du réseau, nous reprenons la même démarche déjà expliquée dans la section 6.2.1.2. La figure 6.9 présente les score de qualité de SDAB et SDRB-2 par rapport à la séparation du reste du réseau.

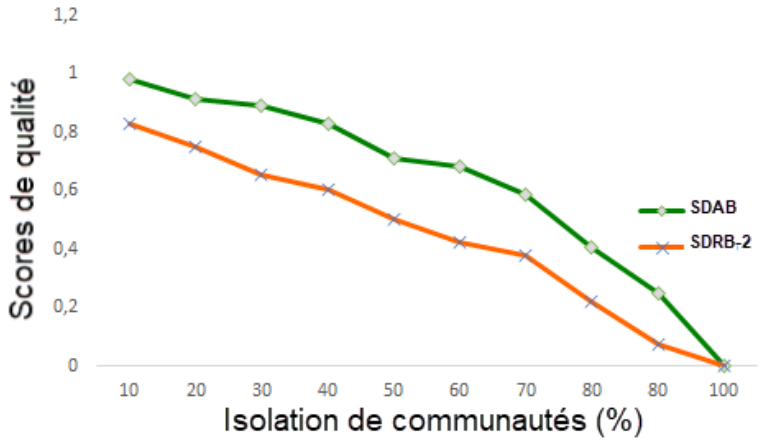


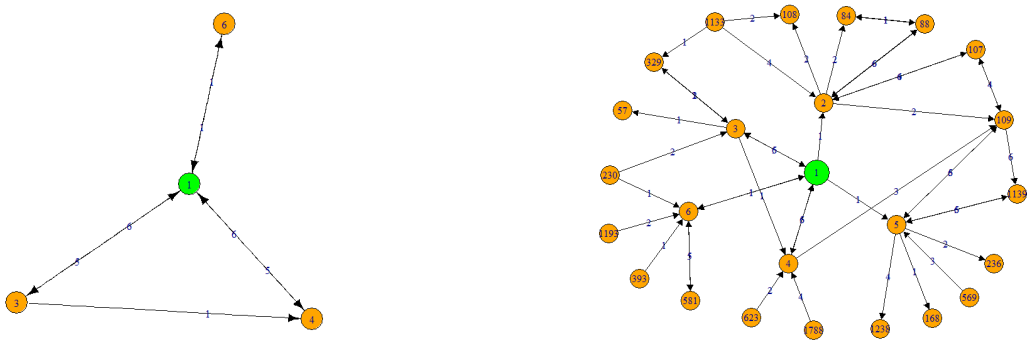
FIGURE 6.9 – Scores de qualité de SDAB et SDRB-2 par rapport à la séparation du reste du réseau.

Sur cette figure, nous constatons trois choses importantes. Premièrement, plus l'intensité de communication augmente, plus les scores de qualité diminuent. Deuxièmement, la qualité de l'algorithme

SDRB-2 est toujours meilleure que celle de SDAB. Troisièmement, lorsque l'isolation de communauté devient maximale, les deux algorithmes tendent vers un score de qualité nul. Ceci est tout à fait logique car si l'isolation atteint 100 %, les poids des liens sortants deviennent nuls, de même le score de qualité.

6.2.2.3 Impact des algorithmes sur la structure de communauté

Pour illustrer l'impact de SDRB-2 sur la structure de communauté, nous nous intéressons au voisinage de longueur 2 du nœud dont le label est « 1 » du réseau représenté sur la figure 6.7. Ainsi, nous extrayons la communauté ego-centrée sur un voisinage direct en utilisant la variante SDAB et la communauté ego-centrée sur un voisinage de longueur 2 en utilisant la variante SDRB-2 tel qu'illustré par la figure 6.10.



(a) Communauté ego-centrée sur un voisinage direct du nœud dont le label est « 1 », détectée par SDAB.

(b) Communauté ego-centrée du nœud « 1 » sur un voisinage de longueur 2, détectée par SDRB-2.

FIGURE 6.10 – Illustration de la communauté ego-centrée du nœud « 1 » sur un voisinage direct et sur un voisinage de longueur 2.

Sur la figure 6.10a, nous présentons la communauté détectée par SDAB. En effet, comme chaque élève énumère ses cinq meilleurs amis des deux sexes, le voisinage direct de chaque nœud sera toujours composé strictement de six nœuds y compris l'ego et de 15 liens s'il s'agit d'une clique. Notons que la minimisation de la fonction qualité effectuée par SDAB a conduit au rejet des nœuds étiquetés par « 2 » et « 5 ». Par conséquent, la communauté ego-centrée du nœud « 1 » sur un voisinage direct est constituée de trois voisins de l'ego. Cela remet en cause la faiblesse de SDAB surtout quand le voisinage direct n'est pas riche d'informations. C'est pourquoi il est souvent nécessaire de pouvoir remonter en haut de la hiérarchie du voisinage afin de mieux comprendre les connexions topologiques et sémantiques du nœud d'intérêt.

Sur la figure 6.10b, nous montrons la communauté ego-centrée sur un voisinage de longueur 2 détectée par SDRB-2. Pour que cette communauté soit lisible, nous avons mis le seuil de qualité τ à 0.4. D'après la figure 6.10b, il est clair donc qu'avec la possibilité d'élargir le voisinage, nous aurons une idée sur les amis des amis du nœud d'intérêt, ce qui nous permettra par exemple de pouvoir suggérer des amis, entre autres.

6.3 Évaluation de la méthode de gestion de fenêtre temporelle

Le plan de cette partie se présente comme suit ; tout d'abord, nous décrivons dans la section 6.3.1 le jeu de données utilisé dans l'expérimentation. Ensuite, nous présentons deux expériences. L'objectif de la première est de montrer que nos prédictions permettent de fournir des valeurs très proches des scores de qualité observés. Rappelons que sur la base de ces prédictions qu'on fait la capture des instantanés. Autrement dit, si les prédictions des scores de qualité sont bonnes, on pourra faire confiance aux valeurs prédites pour capturer les instantanés au fil du temps. La seconde expérience consiste à comparer les instantanés capturés par notre méthode et ceux capturés par une méthode statique utilisant un intervalle de temps régulier pour décomposer l'évolution du réseau. Ces deux expériences sont présentées dans les sections 6.3.2 et 6.3.3.

6.3.1 Description du jeu de données

Le jeu de données que nous utilisons dans la suite est orienté et pondéré. Il comprend plus de deux millions interactions des téléphones mobiles entre 80 étudiants au sein d'un campus (Madan et al., 2012). Ces interactions ont été collectées entre le 05 septembre 2007 et le 16 juillet 2009. Un lien est établi entre deux étudiants si le téléphone de l'un d'eux rentre dans la zone Bluetooth du téléphone de l'autre. Le sens du lien s'oriente de l'étudiant qui envoie le signal Bluetooth vers l'étudiant qui le reçoit. Ce signal indique qu'à l'instant t, les deux étudiants se sont trouvés à une distance moins de dix mètres.

Les données décrivant une interaction sont : l'identifiant de l'expéditeur, l'identifiant du destinataire, l'instant d'envoi du signal et la probabilité p que l'expéditeur soit dans le même étage que le destinataire. En effet, la valeur de p est très significative, car si deux étudiants se trouvent à une proximité de moins de dix mètres et qu'ils soient aussi dans le même étage, cela implique que la possibilité de contact direct devient très probable. Nous considérons que les valeurs de p désignent les poids des liens.

Pour calculer p, les auteurs ont eu recours à l'indication de la force du signal reçu « Received Signal Strength Indication » par rapport au point d'accès Wi-Fi. Cependant, ils n'ont pas présenté le modèle sur lequel ils se sont basés pour calculer p. Afin que le lecteur ait une vision plus claire de la façon dont ce jeu de données a été construit, nous présentons un modèle simple pour calculer la probabilité p.

Soient E un espace euclidien engendré par deux vecteurs $\overrightarrow{i}$ et $\overrightarrow{j}$ perpendiculaires, O l'origine de E qui représente le point d'accès Wi-Fi qui se trouve au centre de l'immeuble dans lequel logent les étudiants et $R = (O, \overrightarrow{i}, \overrightarrow{j})$ un repère orthonormal de base $(\overrightarrow{i}, \overrightarrow{j})$. Nous considérons que les deux points A et $B \in E$, de coordonnées respectives (x_A, y_A) et (x_B, y_B), représentent les emplacements de deux étudiants quelconques dans l'immeuble. Ainsi, la distance entre A et B peut être calculée comme suit :

$$AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2} \tag{6.1}$$

Soient l la largeur de l'immeuble, e la hauteur d'étage et ϱ le nombre d'étages. Vu que les dépla-

cements des étudiants sont surveillés uniquement dans l'immeuble, nous avons :

$$\forall\ A, B \in E, \begin{cases} 0 \leqslant ||x_B| - |x_A|| \leqslant l \\ 0 \leqslant ||y_B| - |y_A|| \leqslant \varrho.e \end{cases} \tag{6.2}$$

Pour que le téléphone d'un étudiant A rentre dans la zone Bluetooth du téléphone d'un étudiant B, il faut que la distance entre leur abscisses soit inférieure à dix mètres, $||x_B| - |x_A|| \leqslant 10m$. Une fois cette contrainte est vérifiée, nous distinguons trois cas, illustrés sur le tableau 6.2. La valeur de $n \in]0, \varrho[$ désigne le nombre d'étages qui séparent deux étudiants.

Tableau 6.2 – Description des trois cas possibles de la distance entre deux étudiants.

Expression	Explication
$\|\|y_B\| - \|y_A\|\| = 0$ $\Rightarrow AB = 100$	A et B se trouvent dans le même étage
$0 < \|\|y_B\| - \|y_A\|\| < n.e$ $\Rightarrow 0 < AB < \sqrt{100 - (n.e)^2}$	A et B ne se trouvent ni dans le même étage ni dans deux étages différents. Ce cas se manifeste si A est dans un étage et B est entrain de se déplacer en ascenseur ou aux escaliers.
$\|\|y_B\| - \|y_A\|\| = \varrho.e$ $\Rightarrow AB = \sqrt{100 - (\varrho.e)^2}$	A et B se trouvent dans deux étages différents

6.3.2 Prédiction des scores de qualité

L'utilisation des séries temporelles P et L nécessite deux paramètres, à savoir, le coefficient de prédiction γ et le coefficient de l'erreur prédite μ. Dans nos expérimentations, nous avons choisi le coefficient $\gamma = 0.5$ pour que la prochaine prédiction soit influencée équitablement par le passé lointain et le passé proche. Concernant le coefficient μ, nous lui avons attribué une petite valeur $\mu = 0.1$ pour que les fluctuations n'aient pas d'impact significatif sur l'erreur prédite de la prochaine prédiction.

Pour évaluer l'efficacité de notre méthode de prédiction, nous avons réalisé deux expériences. L'objectif de la première est d'évaluer l'exactitude de la prédiction à court terme. Pour ce faire, nous avons prédit les scores de qualité du réseau entre le 05 septembre 2007 et le 17 avril 2008, soit une période d'environ 7 mois et deux semaines. Au cours de cette période, il y a eu 2000 interactions. La figure 6.11 présente les scores de qualité observés (courbe colorée en noir), les scores de qualité prédits (courbe colorée en rouge) et les prédictions corrigées (courbe colorée en vert). L'axe des abscisses indique les instants auxquels le réseau a subi un changement. L'axe des ordonnées représente les scores de qualités correspondants.

Sur la figure 6.11, nous constatons que lorsque le nombre de fluctuations est élevé, la prédiction devient moins bonne. Cela s'explique par le fait qu'il est carrément difficile de prédire le comportement des scores de qualité s'il est quasi-aléatoire. En revanche, lorsque le nombre de fluctuations commence à baisser, la prédiction devient de plus en plus bonne.

En outre, nous remarquons sur la figure 6.11 que la prédiction est parfois plus bonne que sa correction. En effet, comme la correction de prédiction est basée sur l'erreur prédite, cela signifie que si cette dernière est mauvaise, la première l'est aussi. Par conséquent, lorsque la courbe de la fonction de qualité présente plusieurs fluctuations en peu de temps, l'erreur prédite devient inexacte et cette inexactitude se propage à la correction de prédiction.

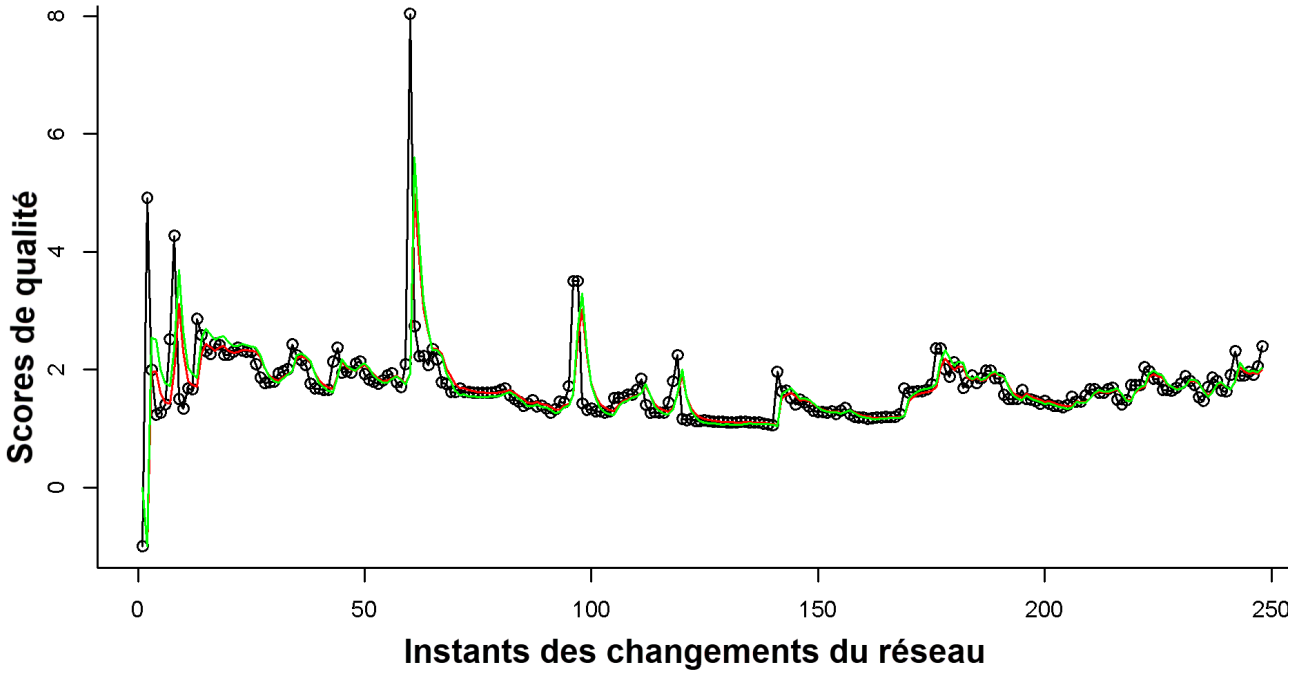


FIGURE 6.11 – Scores de qualité observés et prédits entre le 05 septembre 2007 et le 17 avril 2008.

La seconde expérience consiste à évaluer l'efficacité de notre prédiction à long terme. Pour ce faire, nous avons observé toutes les interactions qui ont eu lieu entre le 05 septembre 2007 et le 03 octobre 2008, soit une période d'une année et quatre semaines. Cette période comprend dix milles interactions. Le nombre d'instants où le réseau a subi un changement est égal à 2205. La figure 6.12 présente les courbes de cette expérience.

Remarquons qu'à long terme, la correction de prédiction devient presque identique avec les scores de qualité observés. Cela prouve donc l'efficacité de notre méthode de prédiction.

6.3.3 Exemples des instantanés détectés

L'objectif de cette expérience est de montrer la différence entre les instantanés détectés par notre méthode et ceux produits par une méthode statique qui décompose l'évolution du réseau sur la base d'un intervalle de temps régulier. Pour ce faire, nous considérons que la taille de fenêtre temporelle est de trente minutes. Ainsi, cette méthode consiste à capturer un nouvel instantané toutes les trente minutes.

Les figures 6.13a, 6.13b, 6.13c et 6.13d présentent l'état du réseau, respectivement, après 30 minutes, une heure, une heure et demi et deux heures. Les valeurs indiquées sur les liens représentent les poids. Pour des raisons de lisibilité, nous n'avons pas affiché les poids des liens sur les figures 6.13c et 6.13d. Sur la base de ces figures, nous constatons que le réseau n'a pas changé durant les deux premiers instantanés et un changement très léger a eu lieu au cours des deux derniers instantanés.

Cette expérience illustre la non pertinence de la méthode statique vu qu'elle peut capturer des instantanés presque identiques au fil du temps. Cela implique une perte de temps et de ressources. Pour pallier à cette faiblesse, nous présentons sur la figure 6.13 des exemples des instantanés capturés par notre méthode dynamique. Pour ce faire, nous considérons que le taux de changements nécessaires pour capturer un nouvel instantané égal de $\eta = 0.8$. Rappelons que notre méthode permet de décomposer

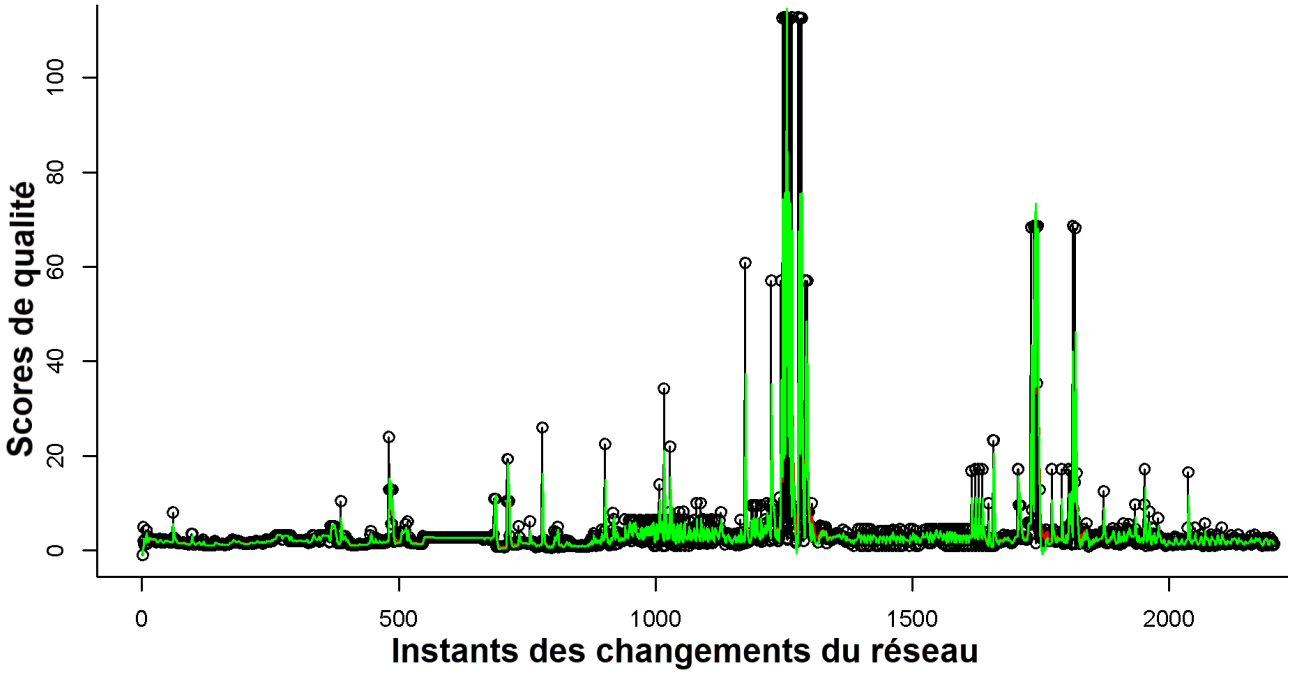


FIGURE 6.12 – Scores de qualité observés et prédits entre le 05 septembre 2007 et le 03 octobre 2008.

l'évolution d'un réseau dynamique suivant le degré des changements que le réseau a subi dans le temps.

Les figures 6.13e, 6.13f, 6.13g et 6.13h présentent quatre instantanés capturés aux différents moments. Les valeurs indiquées sur les liens représentent les poids. Pour des raisons de lisibilité, nous n'avons pas affiché les poids des liens sur les figures 6.13g et 6.13h. Sur ces figures, nous voyons clairement que la taille de la fenêtre temporelle est dynamique. Elle varie en fonction du degré des changements du réseau. La différence considérable entre la taille de ces fenêtres est due au degré des changements du réseau au fil du temps.

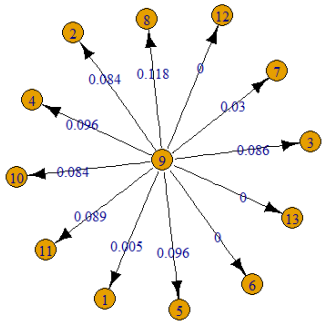
Concernant l'évolution des voisinages ego-centrés dans le temps, nous constatons qu'au début de l'évolution du réseau, il y avait un seul voisinage ego-centré, à savoir, celui du nœud 56 (voir la figure 6.13e). Puis, au fil du temps, le nœud 58 est devenu très actif et un voisinage ego-centré s'est créé autour de lui. La figure 6.13f en présente une illustration. Petit à petit, le voisinage du nœud 19 grandit et les nœuds $20, 35$ et 37 commencent à avoir un voisinage remarquable comme illustré sur la figure 6.13g. La même situation précédente se répète sur la figure 6.13h mais cette fois-ci avec les nœuds $5, 12$ et 68.

6.4 Suivi de communautés ego-centrées dynamiques

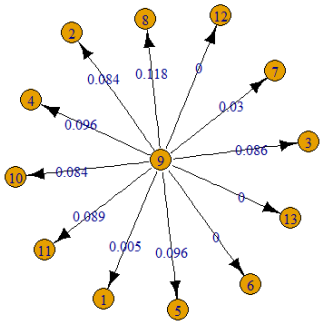
Dans cette section, nous expérimentons notre algorithme de suivi de l'évolution de communautés dynamiques ego-centrées. Nous nous intéressons au voisinage de longueur 2 ($k = 2$).

Grâce à notre méthode de gestion des instantanés, nous avons capturé quatre instantanés du réseau dynamique présenté dans la section 6.3.1. La figure 6.14 montre ces quatre instantanés.

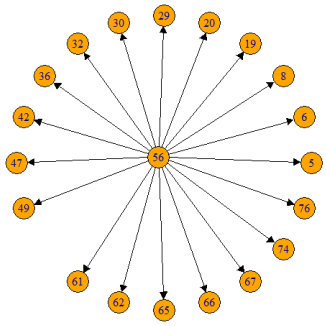
Le tableau 6.3 présente les communautés statiques détectées sur chaque instantané. Afin de réduire la taille de communautés détectées pour qu'elles soient lisibles, nous avons utilisé un seuil de qualité égal à 0.53. Les valeurs affichées sur les liens désignent les poids des liens. Pour des raisons de lisibilité,



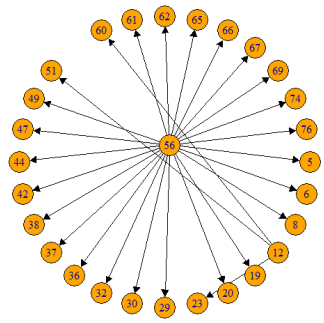
(a) Premier instantané capturé après 30 minutes.



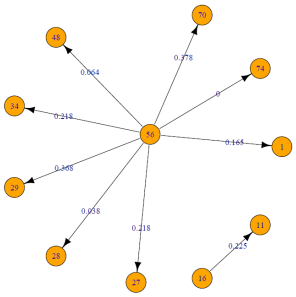
(b) Deuxième instantané capturé après 60 minutes.



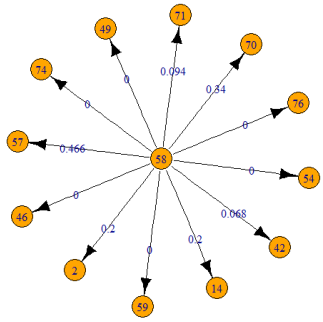
(c) Troisième instantané capturé après 90 minutes.



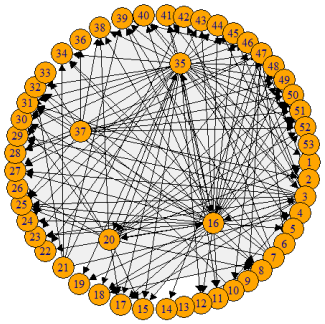
(d) Quatrième instantané capturé après 120 minutes.



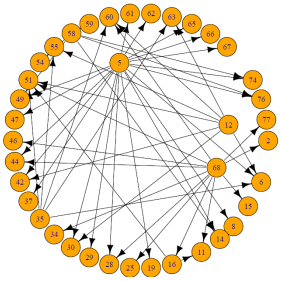
(e) Première fenêtre temporelle représentant l'état initial du réseau. Cette fenêtre temporelle est capturée le 05 septembre 2007 à 14 :02 :11.



(f) Deuxième fenêtre temporelle capturée le 05 septembre 2007 à 14 :12 :33. La taille de cette fenêtre temporelle est de 10 minutes.

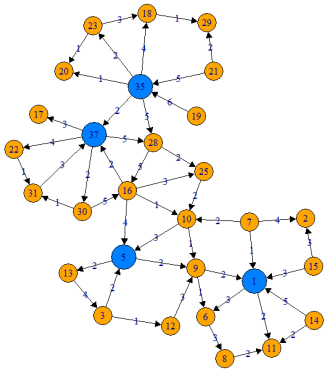


(g) Troisième fenêtre temporelle capturée le 23 janvier 2008 à 14 :27 :42. La taille de cette fenêtre temporelle est de 4 mois.

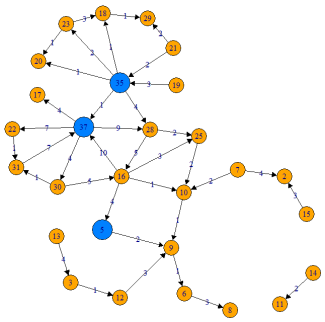


(h) Quatrième fenêtre temporelle capturée le 26 janvier 2008 à 06 :37 :50. La taille de cette fenêtre temporelle est de 3 jours.

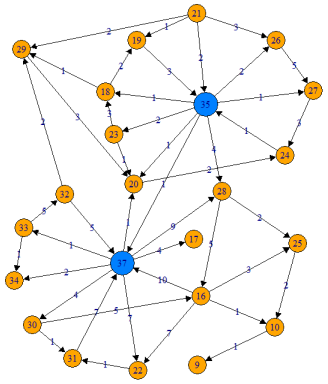
FIGURE 6.13 – Exemples des instantanés capturés par la méthode statique VS ceux produits par notre méthode.



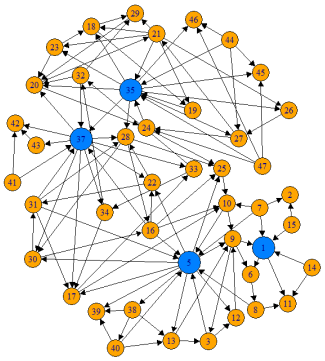
(a) Premier instantané.



(b) Deuxième instantané.



(c) Troisième instantané.



(d) Quatrième instantané.

FIGURE 6.14 – Quatre instantanés successifs capturés par notre méthode de gestion des instantanés.

nous n'avons pas affiché les poids des liens sur la figure 6.14d. Notons que cette fois-ci, nous avons normalisé la valeur du poids entre 0 (deux utilisateurs ne sont pas dans le même étage) et 10 (deux utilisateurs se trouvent dans le même étage).

Tableau 6.3 – Communautés statiques détectées sur les quatre instantanés.

Instantané	Communautés détectées
Premier instantané	$C_1 = \{1, 2, 6, 7, 8, 9, 10, 11, 12, 14, 15\}$ $C_5 = \{5, 10, 9, 12, 3, 13, 16, 25\}$ $C_{35} = \{18, 19, 20, 21, 23, 28, 29, 35, 37\}$ $C_{37} = \{16, 17, 22, 25, 28, 30, 31, 35, 37\}$
Deuxième instantané	$C_{35} = \{18, 19, 20, 21, 23, 28, 29, 35, 37\}$ $C_{37} = \{16, 17, 22, 25, 28, 30, 31, 35, 37\}$
Troisième instantané	$C_{35} = \{18, 19, 20, 21, 23, 24, 26, 27, 28, 29, 35\}$ $C_{37} = \{9, 10, 16, 17, 20, 22, 25, 28, 30, 31, 32, 33, 34, 37\}$
Quatrième instantané	$C_1 = \{1, 2, 6, 7, 8, 9, 10, 11, 12, 14, 15\}$ $C_5 = \{3, 5, 6, 7, 8, 9, 10, 12, 13, 16, 17, 22, 25, 30, 31, 37, 38, 39, 40\}$ $C_{35} = \{18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 32, 33, 35, 44, 45, 46, 47\}$ $C_{37} = \{16, 17, 22, 24, 28, 30, 31, 32, 33, 34, 37, 41, 42, 43\}$

Après avoir exécuté notre algorithme de suivi de communautés sur les quatre instantanés, nous avons obtenu le résultat présenté sur le tableau 6.4.

Ci-dessous quelques commentaires sur le suivi des changements de communautés.

Première transition (Instantané 1 $\longrightarrow$ instantané 2)

1. La disparition de C_1 est due à la disparition de son nœud d'intérêt. Par contre, C_5 a disparu

Tableau 6.4 – Description des changements qui ont eu lieu au cours des quatre instantanés.

Transition	Changements macroscopiques	Changements microscopiques
Instantané 1 $\longrightarrow$ instantané 2	- Disparition de C_1 - Disparition de C_5 - Monotonie de la cohésion de C_{35} - Rétrécissement de l'intensité de communication de C_{35} - Croissance de l'intensité de communication de C_{37} - Monotonie de la cohésion de C_{37}	- Disparition du nœud d'intérêt de C_1 - Disparition des liens (5,10), (5,3) et (5,13) - Monotonie des liens de C_{35} - Croissance des liens de C_{37} - Rétrécissement des liens (35,28), (19,35), (21,35), (35,18) et (35,37)
Instantané 2 $\longrightarrow$ instantané 3	Croissance de la cohésion de C_{35} et C_{37}	- Disparition des nœuds/liens restants des communautés C_1 et C_5 - Apparition de plusieurs liens dans C_{35} et C_{37}
Instantané 3 $\longrightarrow$ instantané 4	- Résurgence de C_1 - Réapparition de C_5 - Croissance de la cohésion de C_{35} et C_{37}	- Apparition du nœud d'intérêt de C_1 - Résurgence des liens de C_1 - Apparition du nœud d'intérêt de C_5 - Croissance des liens de C_5 - Apparition de plusieurs liens dans C_{35} et C_{37}

puisqu'elle n'est plus séparée du reste du réseau.

2. La communauté C_{37} a marqué une croissance de fréquence d'échange entre les nœuds et une monotonie en terme de cohésion.

Deuxième transition (Instantané 2 $\longrightarrow$ instantané 3)

1. La croissance de cohésion de C_{35} et C_{37} implique en quelque sorte une croissance d'intensité de communication vu que la somme des poids des liens a augmenté.
2. Notre interprétation de la disparition du reste des communautés C_1 et C_5 est que ces egos constituent le centre d'intérêt des membres de communauté. Par conséquent, lorsque les egos disparaissent, les membres commencent à « se désabonner » de la communauté au fil du temps.

Troisième transition (Instantané 3 $\longrightarrow$ instantané 4)

1. La communauté C_1 a resurgi puisqu'elle a réapparu sous une forme très proche de la communauté source avec un changement léger des valeurs des poids des liens.
2. Par contre, le type d'évolution de C_5 est « réapparition » car sa nouvelle forme est très différente de la communauté source.

6.5 Conclusion

Au cours de ce chapitre, nous avons réalisé trois principales expériences. Au niveau de la première, nous nous sommes limités au voisinage direct, les communautés détectées étant statiques. Nous avons comparé les quatre variantes de notre algorithme de détection de communautés sur la base de deux critères, à savoir, la cohésion interne et la séparation du reste du réseau. Les résultats d'expérience ont prouvé que SDAB est la meilleure variante. Ensuite, nous nous sommes intéressés aux communautés ego-centrées au-delà du voisinage direct. Pour ce faire, nous avons appliqué notre algorithme SDRB sur un voisinage en deux étapes. Nous avons également prouvé que les communautés détectées en deux étapes sont plus pertinentes que celles trouvées en une étape.

La seconde expérience consiste à évaluer l'efficacité de notre méthode de décomposition de l'évolution du réseau dynamique. Pour ce faire, nous avons utilisé une série temporelle qui cherche à prédire les instants qui correspondent aux nouveaux instantanés au fil du temps. L'objectif était de montrer l'exactitude de nos prédictions et qu'ils produisent des instantanés significatifs. En plus, nous avons comparé notre méthode dynamique avec une méthode statique qui décompose l'évolution du réseau sur la base d'un intervalle de temps régulier. Les résultats ont prouvé la supériorité de notre méthode.

La troisième expérience porte sur le suivi de l'évolution de communautés dynamiques au fil du temps. Nous avons pu expliqué les types d'évolution suivant deux facettes, à savoir, l'aspect topologique et la fréquence d'échange entre les nœuds. Nous avons également vu que les changements (microscopiques) relatifs aux nœuds/liens peuvent aider à expliquer l'origine d'une évolution (macroscopique) de communauté.

Chapitre 7

Conclusion et perspectives

Cette thèse s'intéresse à la détection des communautés dynamiques dans les réseaux sociaux. Pour la réaliser, plusieurs étapes ont été suivies. La première étape était d'effectuer des recherches sur les méthodes de détection de communautés globales dont les résultats nous ont permis non seulement de comprendre les préliminaires du domaine mais aussi de faire le premier pas vers les méthodes de détection de communautés locales qui constituent la base de notre sujet de thèse. Après avoir étudié l'état de l'art, nous avons pu faire les observations suivantes sur l'existant, à savoir :

- il existe peu de travaux sur les communautés ego-centrées statiques. La majorité des travaux existants s'intéressent soit au partitionnement des réseaux, soit aux communautés locales ;
- les algorithmes actuels se limitent généralement à l'aspect topologique de communautés ;
- le découpage des réseaux dynamiques se faisait de manière statique. Autrement dit, les instantanés sont capturés sur la base d'un intervalle de temps régulier ;
- il n'existe pas encore, à notre connaissance, des travaux portant sur le suivi de l'évolution de communautés ego-centrées dynamiques.

Dans l'optique de remédier à ces limites, nous avons proposé une solution de détection/suivi de l'évolution de communautés ego-centrées dans les réseaux dynamiques, orientés et pondérés. Cette solution fonctionne en trois étapes consistant à :

1. faire plusieurs captures du réseau de sorte qu'il y ait des changements considérables entre les captures successives ;
2. détecter des communautés ego-centrées statiques sur chaque capture ;
3. comparer la structure de communautés sur les captures successives afin d'identifier les types d'évolution de communautés au fil du temps.

Après avoir conçu la solution, nous avons procédé à sa validation sur des jeux de données réels. Les expérimentations ont montré que notre solution fournit des résultats conformes aux objectifs visés.

Perspectives

L'analyse de notre solution en termes de gains et de limites nous a permis de dégager un certain nombres de perspectives.

Pour commencer, le principe de fonctionnement de notre algorithme statique est facilement parallélisable vu que la création de chaque communauté ego-centrée se fait indépendamment des autres. Par conséquent, il serait possible de créer un algorithme de type *map-reduce* pour la construction d'un ensemble de communautés ego-centrées pour faire face à des jeux de données plus volumineux.

Concernant la gestion des instantanés, nous savons que le coefficient de prédiction joue un rôle important dans le processus du découpage du réseau. De ce fait, nous souhaitons proposer dans l'avenir

une méthode permettant de mettre à jour la valeur du coefficient de prédiction de sorte que l'erreur de prédiction soit minimale au fil du temps. Cela permettrait d'améliorer considérablement la qualité de la prédiction, et par conséquent, d'avoir des instantanés plus pertinents.

Pour ce qui est du suivi des communautés dynamiques, il serait pertinent de considérer que les egos sont dynamiques. Autrement dit, de nouveaux nœuds d'intérêt peuvent à tout moment apparaitre tout au long de l'évolution du réseau. Cette hypothèse nous permettra de gérer la division de communautés. Nous comptons proposer une stratégie permettant d'élire dynamiquement des egos pour les sous-communautés qui résultent d'une division. Toujours par rapport au suivi de communautés, un autre type d'évolution très intéressant est la fusion de communautés. La prise en compte de ce type d'évolution conduit à la gestion de communautés bi-ego-centrées. Le centre d'intérêt de la communauté résultante sera constitué de deux egos des communautés fusionnées.

Un autre aspect, sur lequel nous espérons avoir l'occasion de travailler et qui nous semble important, est la détection et le suivi de communautés ego-centrées hiérarchiques multi-niveaux. En effet, nous avons constaté que lorsque les communautés deviennent très larges, elles commencent à contenir des sous-communautés ego-centrées construites à partir des nœuds qui sont fortement liés à l'ego. Ces sous-communautés comprennent également des sous-communautés et ainsi de suite. Il s'agit donc de communautés ego-centrées imbriquées. Ainsi, le suivi des changements de la communauté mère ne permet de comprendre que les interactions au sens global. Or, il serait plus pertinent de creuser encore au sein de la communauté mère afin d'identifier les sous-communautés, de comprendre la sémantique de chacune et de pouvoir suivre ses changements de façon plus précise. Là, nous pouvons appliquer notre algorithme statique sur la communauté mère. Ensuite, nous identifions les voisins les plus connectés à l'ego et nous appliquons de nouveau notre algorithme sur ces voisins dans l'optique de détecter les sous-communautés correspondantes. De manière récursive, nous arrivons à trouver les communautés hiérarchiques de la communauté mère.

Liste de publications

- **Article de revue internationale**
  1. **Ahmed OULD MOHAMED MOCTAR**, Idrissa SARR (2016), « Détection de communautés statiques et dynamiques. », *in : Revue d'Intelligence Artificielle* 30.*4*, p. 469–496, DOI : https://doi.org/10.3166/ria.30.469-496.
- **Chapitre de livre**
  1. **Ahmed OULD MOHAMED MOCTAR**, Idrissa SARR (2017), « Instant Messaging for Detecting Dynamic Ego-Centered Communities », *in : Encyclopedia of Social Network Analysis and Mining*, sous la direction de Reda Alhajj et Jon Rokne, New York, NY : Springer New York, p. 1–12, DOI : https://doi.org/10.1007/978-1-4614-7163-9_110216-1.
- **Conférence internationale avec comité de sélection**
  1. **Ahmed OULD MOHAMED MOCTAR**, Idrissa SARR (2017), « Ego-centered Community Detection in Directed and Weighted Networks », *in : Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, Sydney, Australia : ACM, p. 1201–1208, DOI : http://doi.org/10.1145/3110025.3121243.
  2. **Ahmed OULD MOHAMED MOCTAR**, Idrissa SARR (2018), « Ego-Community Evolution Tracking in Instant Messaging Networks », *in : Innovations and Interdisciplinary Solutions for Underserved Areas*, InterSol 2018, (Best paper award), Kigali, Rwanda : Springer International Publishing, p. 13–22, DOI : https://doi.org/10.1007/978-3-319-98878-8_2.
  3. **Ahmed OULD MOHAMED MOCTAR**, Idrissa SARR (2018), « Building Ego-Comm-unity Based on a Non-Closed Neighborhood », *in : 14th African Conference on Research in Computer Science and Applied Mathematics*, Stellenbosch, South Africa, p. 226–235, URL : http://www.cari-info.org/South-africa-2018.
  4. **Ahmed OULD MOHAMED MOCTAR**, Idrissa SARR (2018), « Survey on Social Ego-Community Detection », *in : 7th International Conference on Complex Networks and Their Appications*, Cambridge, United Kingdom, p. 388–399, DOI : https://doi.org/10.1007/978-3-030-05414-4_31.
  5. **Ahmed OULD MOHAMED MOCTAR**, Idrissa SARR, Joel VAUMI TANZOUAK (2019), «Snapshot Setting for Temporal Networks Analysis», *in : 10th EAI International Conference on e-Infrastructure and e-Services for Developing Countries*, Dakar, Senegal, (to appear).

Bibliographie

Abrahao, Bruno et al. (2012), « On the separability of structural classes of communities », *in* : *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, p. 624–632.

Adolescent health network dataset – KONECT (2016), Available at http://konect.uni-koblenz.de/networks/moreno_health.

Asur, Sitaram, Srinivasan Parthasarathy et Duygu Ucar (2009), « An event-based framework for characterizing the evolutionary behavior of interaction graphs », *in* : *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3.*4*, p. 16.

Aynaud, Thomas, Eric Fleury et al. (2013), « Communities in evolving networks : definitions, detection, and analysis techniques », *in* : *Dynamics On and Of Complex Networks, Volume 2*, Springer, p. 159–200.

Aynaud, Thomas et Jean-Loup Guillaume (2010), « Long range community detection », *in* : *LAWDN-Latin-American Workshop on Dynamic Networks*, 4–p.

Barbera, Pablo, Michael Piccirilli et Maintainer Pablo Barbera (2017), « Package 'Rfacebook' », *in* : *R package version 0.6* 15.

Bender-deMoll, S (2018), *NDTV : Network Dynamic Temporal Visualizations. R package version 0.12. 2*.

Bender-deMoll, Skye (2016), *networkDynamic : Dynamic Extensions for Network Objects*, R package version 0.9.0, The Statnet Project (http://statnet.org), URL : https://cran.r-project.org/package=networkDynamic.

Bojanowski, Michal (2015), *intergraph : Coercion Routines for Network Data Objects*, R package version 2.0-2, URL : http://mbojan.github.io/intergraph.

Bródka, Piotr, Stanisław Saganowski et Przemysław Kazienko (2013), « GED : the method for group evolution discovery in social networks », *in* : *Social Network Analysis and Mining* 3.*1*, p. 1–14.

Bron, Coen et Joep Kerbosch (1973), « Algorithm 457 : finding all cliques of an undirected graph », *in* : *Communications of the ACM* 16.*9*, p. 575–577.

Bundy, Alan et Lincoln Wallen (1984), « Breadth-first search », *in* : *Catalogue of Artificial Intelligence Tools*, Springer, p. 13–13.

Butts, Carter T. (2015), *network : Classes for Relational Data*, R package version 1.13.0, The Statnet Project (http://statnet.org), URL : http://CRAN.R-project.org/package=network.

Cazabet, Rémy (2013), « Détection de communautés dynamiques dans des réseaux temporels », thèse de doct., Université Paul Sabatier-Toulouse III.

Cazabet, Remy et Frederic Amblard (2011), « Simulate to detect : a multi-agent system for community detection », *in* : *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on*, t. 2, IEEE, p. 402–408.

Cazabet, Rémy, Frédéric Amblard et Chihab Hanachi (2010), « Detection of overlapping communities in dynamical social networks », *in* : *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, IEEE, p. 309–314.

Chan, Shu-Yan, Pan Hui et Kuang Xu (2009), « Community detection of time-varying mobile social networks », *in* : *Complex Sciences*, Springer, p. 1154–1159.

Chen, Debao et al. (2016), « Multi-objective optimization of community detection using discrete teaching–learning-based optimization with decomposition », *in* : *Information Sciences* 369, p. 402–418.

Chen, Jiyang, Osmar Zaiane et Randy Goebel (2009), « Local community identification in social networks », *in* : *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*, IEEE, p. 237–242.

Chen, Tianlong, Pramesh Singh et Kevin E Bassler (2017), « Network community detection using modularity density measures », *in* : *arXiv preprint arXiv :1708.06810*.

Chen, Wei et al. (2010), « A game-theoretic framework to identify overlapping communities in social networks », *in* : *Data Mining and Knowledge Discovery* 21.*2*, p. 224–240.

Chen, Zengqiang, Zheng Xie et Qing Zhang (2015), « Community detection based on local topological information and its application in power grid », *in* : *Neurocomputing* 170, p. 384–392.

Chowdhary, Janamejaya, Frank E Löffler et Jeremy C Smith (2017), « Community detection in sequence similarity networks based on attribute clustering », *in* : *PloS one* 12.*7*, e0178650.

Clauset, Aaron (2005), « Finding local community structure in networks », *in* : *Physical review E* 72.*2*, p. 026132.

Creusefond, Jean, Thomas Largillier et Sylvain Peyronnet (2016), « On the evaluation potential of quality functions in community detection for different contexts », *in* : *International Conference and School on Network Science*, Springer, p. 111–125.

Csárdi, Gábor, Tamás Nepusz et Edoardo M Airoldi (2016), « Statistical Network Analysis with igraph », *in* :

Danisch, Maximilien (2015), « Mesures de proximité appliquées à la détection de communautés dans les grands graphes de terrain », thèse de doct., Paris 6.

Danisch, Maximilien, Jean-Loup Guillaume et Bénédicte Le Grand (2014a), « Learning a proximity measure to complete a community », *in* : *Data Science and Advanced Analytics (DSAA), 2014 International Conference on*, IEEE, p. 90–96.

— (2014b), « Multi-ego-centered communities in practice », *in* : *Social network analysis and mining* 4.*1*, p. 1–10.

Ding, Xiaoyou, Jinapel Zhang et Jing Yang (2018), « A robust two-stage algorithm for local community detection », *in* : *Knowledge-Based Systems* 152.

Fagnan, Justin, Osmar Zaiane et Denilson Barbosa (2014), « Using triads to identify local community structure in social networks », *in* : *Advances in Social Networks Analysis and Mining (ASONAM), 2014 IEEE/ACM International Conference on*, IEEE, p. 108–112.

Falkowski, Tanja, Anja Barth et Myra Spiliopoulou (2008), « Studying community dynamics with an incremental graph mining algorithm », *in* : *AMCIS 2008 Proceedings*, p. 29.

Fanrong, Meng et al. (2014), « Local community detection in complex networks based on maximum cliques extension », *in* : *Mathematical Problems in Engineering* 2014.

Fortunato, Santo (2010), « Community detection in graphs », *in* : *Physics reports* 486.*3*, p. 75–174.

Fortunato, Santo et Darko Hric (2016), « Community detection in networks : A user guide », *in* : *Physics Reports* 659, p. 1–44.

Génois, Mathieu et al. (2015), « Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers », *in* : *Network Science* 3.*3*, p. 326–347.

Girvan, Michelle et Mark EJ Newman (2002), « Community structure in social and biological networks », *in* : *Proceedings of the national academy of sciences* 99.*12*, p. 7821–7826.

Greene, Derek, Donal Doyle et Padraig Cunningham (2010), « Tracking the evolution of communities in dynamic social networks », *in* : *Advances in social networks analysis and mining (ASONAM), 2010 international conference on*, IEEE, p. 176–183.

Hamann, Michael, Eike Röhrs et Dorothea Wagner (2017), « Local Community Detection Based on Small Cliques », *in* : *Algorithms* 10.*3*, p. 90.

Hollocou, Alexandre, Thomas Bonald et Marc Lelarge (2016), « Improving PageRank for local community detection », *in* : *arXiv preprint arXiv :1610.08722*.

— (2017), « Multiple Local Community Detection », *in* : *Performance Evaluation*.

Holme, Petter (2015), « Modern temporal network theory : a colloquium », *in* : *The European Physical Journal B* 88.*9*, p. 234.

Hopcroft, John et al. (2004), « Tracking evolving communities in large linked networks », *in* : *Proceedings of the National Academy of Sciences* 101.*suppl 1*, p. 5249–5253.

Huang, Jianbin et al. (2011), « Towards online multiresolution community detection in large-scale networks », *in* : *PloS one* 6.*8*, e23829.

Jdidia, Manel Ben, Céline Robardet et Eric Fleury (2007), « Communities detection and analysis of their dynamics in collaborative networks. », *in* : *ICDIM*, p. 744–749.

Jeffrey Travers, Stanley Milgram (1969), « An Experimental Study of the Small World Problem », *in* : *Sociometry* 32.*4*, p. 425–443, issn : 00380431.

Kang, U et Christos Faloutsos (2011), « Beyond'caveman communities' : Hubs and spokes for graph compression and mining », *in* : *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, IEEE, p. 300–309.

Kaple, Madhura, Ketki Kulkarni et Katerina Potika (2017), « Viral Marketing for Smart Cities : Influencers in Social Network Communities », *in* : *9th IEEE International Workshop on Big Data Appications in Smart City Development*.

Khan, Bisma S et Muaz A Niazi (2017), « Network Community Detection : A Review and Visual Survey », *in* : *arXiv preprint arXiv :1708.00977*.

Knuth, Donald Ervin (1993), *The Stanford GraphBase : a platform for combinatorial computing*, t. 37, Addison-Wesley Reading.

Krings, Gautier et al. (2012), « Effects of time window size and placement on the structure of an aggregated communication network », *in* : *EPJ Data Science* 1.*4*, p. 1–16.

Kumar, Akash, Akshay Dutt et Gautam Saini (2014), « Merge Sort Algorithm », *in* : *International Journal of Research* 1.*11*, p. 16–21.

Lancichinetti, Andrea, Santo Fortunato et János Kertész (2009), « Detecting the overlapping and hierarchical community structure in complex networks », *in* : *New Journal of Physics* 11.*3*, p. 033015.

Leskovec, Jure, Kevin J Lang et Michael Mahoney (2010), « Empirical comparison of algorithms for network community detection », *in* : *Proceedings of the 19th international conference on World wide web*, ACM, p. 631–640.

Li, Jingyong et al. (2012), « Cdbia : a dynamic community detection method based on incremental analysis », *in* : *Systems and Informatics (ICSAI), 2012 International Conference on*, IEEE, p. 2224–2228.

Li, Weimin et al. (2018), « An overlapping network community partition algorithm based on semi-supervised matrix factorization and random walk », *in* : *Expert Systems with Applications* 91, p. 277–285.

Lim, Yongsub, U Kang et Christos Faloutsos (2014), « Slashburn : Graph compression and mining beyond caveman communities », *in* : *IEEE Transactions on Knowledge and Data Engineering* 26.*12*, p. 3077–3089.

Lim, Yongsub, Won-Jo Lee et al. (2017), « MTP : discovering high quality partitions in real world graphs », *in* : *World Wide Web* 20.*3*, p. 491–514.

Lin, Yu-Ru et al. (2009), « Analyzing communities and their evolutions in dynamic social networks », *in* : *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3.*2*, p. 8.

Liu, Ying, Jason Moser et Selin Aviyente (2014), « Network community structure detection for directional neural networks inferred from multichannel multisubject EEG data », *in* : *IEEE Transactions on Biomedical Engineering* 61.*7*, p. 1919–1930.

Lu, Zongqing, Yonggang Wen et Guohong Cao (2013), « Community detection in weighted networks : Algorithms and applications », *in* : *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*, IEEE, p. 179–184.

Luo, Feng, James Z Wang et Eric Promislow (2008), « Exploring local community structures in large networks », *in* : *Web Intelligence and Agent Systems : An International Journal* 6.*4*, p. 387–400.

Madan, Anmol et al. (2012), « Sensing the" health state" of a community », *in* : *IEEE Pervasive Computing* 11.*4*, p. 36–45.

McColgan, Peter et al. (2017), « Structural and functional brain network correlates of depressive symptoms in premanifest Huntington's disease », *in* : *Human Brain Mapping* 38.*6*, p. 2819–2829.

Moctar, Ahmed Ould Mohamed et Idrissa Sarr (2016), « Détection de communautés statiques et dynamiques. », *in* : *Revue d'Intelligence Artificielle* 30.*4*, p. 469–496, doi : 10.3166/ria.30.469-496.

— (2017a), « Ego-centered Community Detection in Directed and Weighted Networks », *in* : *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, Sydney, Australia : ACM, p. 1201–1208, doi : 10.1145/3110025.3121243.

— (2017b), « Instant Messaging for Detecting Dynamic Ego-Centered Communities », *in* : *Encyclopedia of Social Network Analysis and Mining*, sous la dir. de Reda Alhajj et Jon Rokne, New York, NY : Springer New York, p. 1–12, isbn : 978-1-4614-7163-9, doi : 10.1007/978-1-4614-7163-9_110216-1.

— (2018a), « Building ego-community based on a non-closed neighborhood », *in* : *14th African Conference on Research in Computer Science and Applied Mathematics*, Stellenbosch, South Africa, p. 226–235, URL : http://www.cari-info.org/South-africa-2018.

MOCTAR, Ahmed Ould Mohamed et Idrissa SARR (2018b), « Ego-Community Evolution Tracking inInstant Messaging Networks », *in* : *Innovations and Interdisciplinary Solutions for Underserved Areas*, InterSol 2018, (Best paper award), Kigali, Rwanda : Springer International Publishing, p. 13–22, DOI : 10.1007/978-3-319-98878-8_2.

— (2018c), « Survey on Social Ego-Community Detection », *in* : *The 7th International Conference on Complex Networks and Their Applications)*, Cambridge, United Kingdom, p. 388–399, DOI : 10.1007/978-3-030-05414-4_31.

MOCTAR, Ahmed Ould Mohamed, Idrissa SARR et Joel VAUMI TANZOUAK (2019), « Snapshot setting for Temporal Networks Analysis », *in* : *10th EAI International Conference on e-Infrastructure and e-Services for Developing Countries*, (to appear), Dakar, Senegal.

MOREVNO, Jacob Levy (1934), « Who Shall Survive ? », *in* : *The Journal of Nervous And Mental Disease*.

MUCHA, Peter J et al. (2010), « Community structure in time-dependent, multiscale, and multiplex networks », *in* : *science* 328.*5980*, p. 876–878.

NGONMANG, Blaise, Maurice TCHUENTE et Emmanuel VIENNET (2012), « Local community identification in social networks », *in* : *Parallel Processing Letters* 22.*01*, p. 1240004.

PALLA, Gergely, Albert-László BARABÁSI et Tamás VICSEK (2007), « Quantifying social group evolution », *in* : *Nature* 446.*7136*, p. 664–667.

PECH, Ratha et Hao DONG (2017), « Local similarity and community paradigm : The robust methods toward link prediction », *in* : *Big Data Analysis (ICBDA), 2017 IEEE 2nd International Conference on*, IEEE, p. 827–831.

PETKOVA, Antoniya et al. (2016), « Accelerating the distributed simulations of agent-based models using community detection », *in* : *Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2016 IEEE RIVF International Conference on*, IEEE, p. 25–30.

PSORAKIS, Ioannis et al. (2012), « Inferring social network structure in ecological systems from spatio-temporal data streams », *in* : *Journal of the Royal Society Interface*, rsif20120223.

QI, Xingqin et al. (2017), « Eb&D : A new clustering approach for signed social networks based on both edge-betweenness centrality and density of subgraphs », *in* : *Physica A : Statistical Mechanics and its Applications* 482, p. 147–157.

RADICCHI, Filippo et al. (2004), « Defining and identifying communities in networks », *in* : *Proceedings of the National Academy of Sciences* 101.*9*, p. 2685–2663.

RAHIMI, Shadi, Alireza ABDOLLAHPOURI et Parham MORADI (2017), « A multi-objective particle swarm optimization algorithm for community detection in complex networks », *in* : *Swarm and Evolutionary Computation*.

RATNAYAKE, Ruwan et al. (2016), « Assessment of Community Event–Based Surveillance for Ebola Virus Disease, Sierra Leone, 2015 », *in* : *Emerging infectious diseases* 22.*8*, p. 1431.

ROSSETTI, Giulio et Rémy CAZABET (2017), « Community Discovery in Dynamic Networks : a Survey », *in* : *arXiv preprint arXiv :1707.03186*.

Rosvall, Martin et Carl T Bergstrom (2010), « Mapping change in large networks », *in* : *PloS one* 5.*1*, e8694.

Sekara, Vedran, Arkadiusz Stopczynski et Sune Lehmann (2016), « Fundamental structures of dynamic social networks », *in* : *Proceedings of the national academy of sciences* 113.*36*, p. 9977–9982.

Shang, Jiaxing et al. (2014), « A real-time detecting algorithm for tracking community structure of dynamic networks », *in* : *arXiv preprint arXiv :1407.2683*.

Shi, Jianbo et Jitendra Malik (2000), « Normalized cuts and image segmentation », *in* : *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.*8*, p. 888–905.

Su, Yi-Jen et Che-Chun Lee (2017), « Overlapping community detection with seed set expansion by local cluster coefficient », *in* : *Consumer Electronics-Taiwan (ICCE-TW), 2017 IEEE International Conference on*, IEEE, p. 73–74.

Tantipathananandh, Chayant, Tanya Berger-Wolf et David Kempe (2007), « A framework for community identification in dynamic social networks », *in* : *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, p. 717–726.

Team, R Core et al. (2013), « R : A language and environment for statistical computing », *in* :

Team, RStudio et al. (2015), « RStudio : integrated development for R », *in* : *RStudio, Inc., Boston, MA URL http ://www. rstudio. com* 42.

Tsourakakis, Charalampos (2015), « The k-Clique Densest Subgraph Problem », *in* : *Proceedings of the 24th international conference on world wide web*, International World Wide Web Conferences Steering Committee, p. 1122–1132.

Tsourakakis, Charalampos et al. (2013), « Denser than the densest subgraph : extracting optimal quasi-cliques with quality guarantees », *in* : *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, p. 104–112.

Viano, M.-C et A. Philippe, *Maitrise d'Économétrie, Cours de Séries Temporelles*, Université des Sciences et Technologies de Lille.

Wan, Cong et al. (2016), « Communities Detection Algorithm Based on General Stochastic Block Model in Mobile Social Networks », *in* : *Advanced Cloud and Big Data (CBD), 2016 International Conference on*, IEEE, p. 178–185.

Wang, Meng et al. (2017), « A Community Detection Algorithm Based on Jaccard Similarity Label Propagation », *in* : *International Conference on Intelligent Data Engineering and Automated Learning*, Springer, p. 45–52.

Wang, Wenjun (2016), « Modeling influence diffusion in networks for community detection, resilience analysis and viral marketing », thèse de doct., The University of Iowa.

Wang, Yi, Bin Wu et Nan Du (2008), « Community evolution of social network : feature, algorithm and model », *in* : *arXiv preprint arXiv :0804.4356*.

Weiss, Robert S et Eugene Jacobson (1955), « A method for the analysis of the structure of complex organizations », *in* : *American Sociological Review* 20.*6*, p. 661–668.

Whang, Joyce Jiyoung, David F Gleich et Inderjit S Dhillon (2016), « Overlapping community detection using neighborhood-inflated seed expansion », *in* : *IEEE Transactions on Knowledge and Data Engineering* 28.*5*, p. 1272–1284.

Xiang, Ju et al. (2016), « Local modularity for community detection in complex networks », *in* : *Physica A : Statistical Mechanics and its Applications* 443, p. 451–459.

Xu, Kevin S, Mark Kliger et Alfred O Hero III (2011), « Tracking communities in dynamic social networks », *in* : *Social computing, behavioral-cultural modeling and prediction*, Springer, p. 219–226.

Xu, Ke et al. (2016), « Mining community and inferring friendship in mobile social networks », *in* : *Neurocomputing* 174, p. 605–616.

Yang, Jaewon et Jure Leskovec (2015), « Defining and evaluating network communities based on ground-truth », *in* : *Knowledge and Information Systems* 42.*1*, p. 181–213.

Yang, Tianbao et al. (2011), « Detecting communities and their evolutions in dynamic social networks-a Bayesian approach », *in* : *Machine learning* 82.*2*, p. 157–189.

Zheng, Wei, XiaoKang Zhao et Zhao Kang (2017), « Analysis of Associtivity and Community Structure in Mobile Social Networks », *in* : *Procedia Computer Science* 107, p. 630–635.