Détection de communautés dynamiques dans des réseaux temporels

Rémy Cazabet

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue par :
Rémy CAZABET

le 22 Mars 2013

Titre :

Détection de communautés dynamiques dans des réseaux temporels

---

École doctorale et discipline ou spécialité :
ED MITT : Domaine STIC : Intelligence Artificielle

Unité de recherche :
IRIT - UMR CNRS 5505

Directeur(s) de Thèse :

Pr. Chihab Hanachi
Co-directeur : Frédéric Amblard

Jury :
Rapporteurs :
Jean-Loup GUILLAUME, Maître de conférence, LIP6, Paris VI
Pascale KUNTZ-COSPEREC, Professeur, Polytech'Nantes, Université de Nantes
---
Examinateurs :
Bertrand JOUVE, Professeur, Université Lumière Lyon 2
Lynda TAMINE-LECHANI, Professeur, IRIT, Université Toulouse III
Frédéric AMBLARD, Maître de conférence, IRIT, Université Toulouse I
Chihab HANACHI, Professeur, IRIT, Université Toulouse I

Remerciements

Ce travail de thèse, comme tout processus complexe, n'est pas le fruit d'un seul individu agissant seul. Bien au contraire, c'est tout un réseau d'acteurs qui auront été nécessaires à son achèvement, chacun intervenant sur des périodes de temps plus ou moins longues, apparaissant ou disparaissant du "système", dans un processus purement dynamique. C'est donc un plaisir pour moi de remercier ici chacune de ces personnes, sans qui le résultat final aurait été différent.

Frédéric Amblard tout d'abord, qui a encadré cette thèse au jour le jour, et qui m'a énormément apporté, à de nombreux niveaux. Tout d'abord, sur la science en général, car jouer avec des données pour obtenir des résultats amusants et obtenir des résultats scientifiques ne sont pas toujours équivalents. Il m'a également guidé et aidé dans l'élaboration de chacun de mes travaux. Enfin, il m'a également apporté son aide dans tous ces à-cotés qui sont si importants : trouver des financements, ouvrir de nouvelles portes, créer des contacts, bref, sortir de devant son écran. Je pense que beaucoup de doctorants manquent cruellement de cet aspect, et, sans lui, cela m'aurait très certainement manqué également. Je dois finir par le remercier de m'avoir accordé beaucoup de confiance, de liberté dans mes choix, et d'avoir supporté mon entêtement occasionnel.

Je remercie également Chihab Hanachi, qui a accepté de diriger cette thèse et qui en a supervisé le déroulement. Son aide a été particulièrement précieuse lors de la phase finale, et en particulier pour la rédaction du présent document, pour lequel il a beaucoup apporté.

Je remercie de plus les membres de mon jury, Pascale Kuntz, Jean-Loup Guillaume, Bertrand Jouve et Lynda Tamine-Lechani, pour avoir accepté d'en faire partie, ainsi que pour leurs remarques et commentaires. Je remercie également Camille Roth, ainsi que Bertrand Jouve à nouveau, pour m'avoir donné un premier retour sur mes travaux lors du comité de thèse.

Toujours sur le plan académique, de nombreuses personnes m'ont apporté d'enrichissantes discussions, et ont contribué à certains aspects de la thèse. Je peux notamment citer, dans le désordre, Emmanuel Navarro, avec qui j'ai travaillé sur le problème de la détection de communautés statiques, Georges Gonzales et Richard Bon qui ont accepté de partager leurs données sur les isards, et m'ont consacré du temps pour partager leur expertise du domaine, ou encore Maud Leguistin, avec qui j'ai collaboré pour le travail sur Facebook.

Une autre partie importante de ces remerciements va vers les personnes que j'ai côtoyé durant ces trois ans (et quelques mois) de thèse, et en particulier les résidents de la ME310, qui ont changé au cours du temps, mais qui qui font tous partie de cette "communauté dynamique", dont l'âme semble persister malgré le remplacement de ses éléments constituants. Merci donc à Tristram, Paul, John, Maroussia, Sylvain, Thomas, Joseph, Angela, Thai, Dengji, Eunate, Arnaud, Diane, Nicolas, Alex, Meriam, et tous les autres, parce que si j'essaye d'êtres exhaustif, je vais forcément oublier des gens

Au-delà de la ME310, je souhaite également remercier l'ensemble des personnes côtoyées à la manufacture des tabacs durant ces années. Leur convivialité fournit un cadre de travail particulièrement agréable et enrichissant.

Enfin, je remercie tous les membres de ma famille pour avoir essayé tant de fois de comprendre

ce que je pouvais bien faire. Plus particulièrement, mon père qui a directement participé à l'élaboration de ce document par sa relecture complète et efficace, ce qui n'était pas une mince tâche. Nathaly également, qui a du supporter les derniers mois de thèses où je n'avais pas beaucoup de temps à lui consacrer, et dont la présence m'a apporté beaucoup de joie dans les moments difficiles. Ma pensée finale sera pour ma mère, qui, elle aussi, m'a toujours soutenue, malgré la maladie ; jusqu'au bout elle a été enthousiaste et pleine de volonté, ce qui m'a certainement aidé à surmonter l'épreuve qu'à été sa perte.

Merci.

"Le vaisseau sur lequel Thésée s'était embarqué avec les autres jeunes gens, et qu'il ramena heureusement à Athènes, était une galère à trente rames, que les Athéniens conservèrent jusqu'au temps de Démétrios de Phalère. Ils en ôtaient les vieilles pièces, à mesure qu'elles se gâtaient, et les remplaçaient par des neuves qu'ils joignaient solidement aux anciennes. Aussi les philosophes, en se disputant sur ce genre de sophisme qu'ils appellent croissant, citent ce vaisseau comme un exemple de doute, et soutiennent les uns que c'était toujours le même, les autres que c'était un vaisseau différent."

(Plutarque, 75 AD [traduction de Dominique Ricard 1862], Vie de Thésée, paragraphe 23)

Table des matières

Table des figures

Introduction

Contexte

L'analyse de réseaux issus de données réelles, aussi appelés graphes de terrains, est un domaine ayant conduit à de nombreux travaux ces dernières années. Dans de très nombreuses disciplines, en effet, il est possible de représenter des systèmes réels sous la forme de réseaux. On peut notamment citer les réseaux de neurones en biologie, les réseaux sociaux dans les sciences humaines, ou encore les réseaux interbancaires en économie, pour ne donner que quelques exemples. Récemment, la popularité croissante d'internet et du web a conduit à la naissance d'immenses réseaux, pouvant contenir des milliards de nœuds et de liens, et pour lesquels il est possible d'obtenir de grandes quantités d'information. Ces **grands graphes** sont devenus un sujet d'étude à part entière. Une autre particularité des graphes de terrains, et cela est particulièrement visible dans le cas des réseaux issus du web, est que ce sont souvent des **graphes dynamiques**, ils évoluent au cours du temps, ce qui se traduit par l'ajout et la disparition de nœuds et de liens. La dynamique de ces réseaux est donc un facteur important à prendre en compte, et de plus en plus de travaux sont consacrés à ce sujet.

L'un des problèmes essentiels sur les réseaux, parce qu'il est complexe et a beaucoup d'applications, est la **détection de communautés**. Les communautés sont des structures mésoscopiques du réseau, connues pour être présentes dans la plupart des graphes de terrain, et en particulier dans les réseaux de type petit monde. On peut les définir comme des ensembles de nœuds fortement liés entre eux, et plus faiblement liés avec le reste du réseau [GN02]. Ces structures ont du sens dans les réseaux : par exemple, dans un réseau social, ces communautés pourront correspondre à des groupes sociaux (famille, groupe d'amis, etc.). Elles pourront correspondre à des champs lexicaux dans des réseaux de synonymie, ou encore à des blogs traitant de sujets similaires dans une analyse de pages web, pour donner quelques exemples.

Problématique

L'objectif de cette thèse est de proposer une nouvelle approche pour la détection de communautés dynamiques dans des réseaux dynamiques, et plus particulièrement dans des **réseaux temporels** [HS12]. Les réseaux dynamiques, qui changent au cours du temps, peuvent en effet être traités de plusieurs manières :

- On peut tout d'abord considérer qu'un réseau dynamique est une succession de réseaux statiques. Cette approche a longtemps été la plus utilisée. Pour suivre l'évolution d'une plateforme de réseau social en ligne de type Facebook, on peut par exemple capturer l'état complet du réseau chaque semaine, chaque mois ou chaque année. La succession de ces réseaux statiques ponctuels, appelés instantanés, va constituer l'évolution du réseau. Cette approche présente l'intérêt de pouvoir facilement réutiliser les méthodes statiques, chaque instantané pouvant être étudié de manière classique.

– Il est aussi possible d'utiliser, et c'est notre positionnement, le principe de réseaux temporels : le réseau est alors constitué de nœuds et de liens auxquels on associe des séquences d'intervalles, correspondant aux intervalles de temps durant lesquels le nœud ou le lien est présent dans le réseau. On parle également de graphes variants au cours du temps (Time-varying graphs [CFQS12]).

Si de nombreux algorithmes avaient déjà été proposées pour détecter des communautés sur des réseaux dynamiques constitués de séquences d'instantanés ([MRM+10, PBV07, GDC10, AG10a], pour n'en citer que quelques-unes), aucune méthode n'avait encore été proposée pour traiter des réseaux temporels, au moment où a débuté cette thèse.

La première question à laquelle on cherchera à répondre sera donc :

Comment détecter des communautés dans des réseaux temporels ?

Travailler sur des réseaux temporels plutôt que sur des séquences d'instantanés n'est cependant pas seulement une question de représentation de l'évolution du réseau. Un problème que soulève la plupart des méthodes présentées jusqu'ici est l'augmentation proportionnelle de la complexité de l'algorithme avec le nombre de pas de temps. En conséquence, augmenter le nombre d'étapes d'évolution pour avoir une plus faible granularité — des instantanés moins espacés dans le temps pour augmenter leur similarité — signifie augmenter d'autant le temps de calcul. Ceci signifie que traiter tous les détails de l'évolution d'un grand réseau devient très coûteux en temps, et généralement impossible en un temps fini. Or, dans des réseaux de terrain, le nombre d'évènements, c'est à dire de changements du réseau, est souvent très grand. Lorsque le nombre d'évènements devient grand comparativement au nombre de nœuds du réseau, nous parlerons de réseaux évoluant fortement. La plupart des réseaux réels, si on veut les étudier au niveau de granularité le plus bas, ce qui est notre objectif, sont des réseaux évoluant fortement.

Une autre question à laquelle nous voulions répondre en travaillant sur des réseaux temporels était donc :

Comment concevoir un algorithme de détection de communautés adapté au traitement des réseaux évoluant fortement ?

Enfin, des travaux précédents ont mis en évidence certaines difficultés particulières inhérentes à la détection de communautés dynamiques :

– Les recouvrements de communautés, c'est à dire la possibilité pour un nœud d'appartenir à plus d'une communauté, sont courants dans les réseaux réels [For10]. Plusieurs algorithmes proposés pour détecter des communautés dynamiques ne prennent pas ce problème en compte, et ne proposent que des communautés non recouvrantes. Nous voulons en revanche que notre algorithme soit capable de gérer de telles situations.
– L'évolution des communautés ne se résume pas à gagner ou perdre des nœuds, elles peuvent aussi effectuer des opérations plus complexes, telles que la fusion et la division [PBV07]
– Un problème important dans le suivi des communautés est le problème de la stabilité [AG10b]. Les algorithmes de détection de communautés statiques, sur lesquels beaucoup de méthodes dynamiques existantes sont basées, sont instables, dans le sens où ils peuvent donner des résultats très différents pour des réseaux pourtant semblables. Ce problème devient important dans le cas dynamique, car il est important de s'assurer que les évolutions des communautés détectées correspondent bien à des modifications structurelles, et ne sont pas des effets de bord dus à l'algorithme utilisé. Toutes les méthodes existantes souffrent de ce problème et essayent de le limiter par un moyen ou un autre.

L'algorithme que nous proposons doit donc, lui aussi, offrir une solution satisfaisante à ces problèmes.

Approche et contributions

La solution que nous proposons est caractérisée par une approche originale et novatrice. Comme nous ne souhaitons pas travailler sur des séquences d'instantanés, nous proposons de ne pas effectuer de détections, à intervalles réguliers, sur l'ensemble du réseau. Nous proposons au contraire de prendre en compte chaque modification du réseau, définie dans le réseau temporel, comme l'information à traiter.

La première contribution de cette thèse est donc la définition d'un méta-algorithme pour détecter des communautés dans des réseaux temporels : iLCD. Ce méta-algorithme décrit en détail la séquence des actions à effectuer pour détecter les communautés sur un réseau temporel, tout en laissant ouverte la manière dont ces actions elles-mêmes doivent être réalisées. Nous en proposons deux implémentations, mais d'autres seraient également possibles.

iLCD est donc la solution que nous proposons pour détecter des communautés dans des réseaux temporels.

Elle gère la façon dont de nouvelles communautés pourront être détectées, quels événements pourront amener à fusionner ou à supprimer des communautés, ainsi que l'impact de chaque modification du réseau sur les communautés existantes.

Une propriété importante de cette méthode est qu'elle n'effectue que des calculs à un niveau local au fur et à mesure des modifications du réseau. Quelles que soient les modifications du réseau, seules les communautés impactées pourront être modifiées, et de nouvelles communautés ne pourront apparaître qu'à l'endroit où la modification du réseau aura eut lieu. Une conséquence directe est que seul un calcul local doit être effectué pour chaque variation topologique du réseau, c'est à dire pour chaque lien ajouté ou retiré. En conséquence, la complexité de l'algorithme n'augmente pas avec le nombre d'instantanés dont on dispose, c'est à dire, avec le niveau de granularité de l'évolution. La complexité augmente simplement linéairement avec le nombre de modifications unitaires du réseau, de type ajout ou suppression de lien.

Pour que la complexité reste faible, même avec un graphe évoluant fortement, il faut donc que l'implantation de chacune des étapes d'iLCD soit de faible complexité, tout en permettant la détection de communautés pertinentes. C'est ce que nous avons essayé de faire dans les implémentations d'iLCD que nous proposons, iLCD-NRMH [CAH10] et iLCD-CDIE [CA11], et en particulier dans cette seconde.

Cette méthode permet donc de traiter de grands graphes évoluant fortement.

Il s'agit là de notre seconde contribution principale. Cette approche est basée sur des calculs locaux, et donc une vision strictement locale du réseau. De plus, les communautés ne sont jamais détectées en une fois sur la globalité du réseau. Un problème important est donc d'assurer que les communautés existant à un instant donné du réseau sont pertinentes, et d'une nature comparable à celles que pourraient trouver un algorithme statique sur un réseau correspondant à l'état du réseau dynamique à cet instant donné. Nous avons donc particulièrement travaillé sur la problématique de la validation de communautés, et en particulier, comme nous n'étions pas complètement satisfaits des méthodes existantes pour la génération de graphes avec communautés, nous avons travaillé sur la validation de communautés sur des réseaux réels. Ces travaux portent sur des réseaux statiques.

Nous présentons deux approches : l'une, constituée de deux métriques, permet de comparer

des solutions proposées par différents algorithmes sur le même réseau, y compris des méthodes avec et sans recouvrement.

L'autre concerne une évaluation empirique faite par des utilisateurs des communautés détectées par différents algorithmes sur des réseaux réels. [CLA12]

Nous avons ensuite travaillé sur deux applications à des cas concrets. Nous avons choisi des réseaux très différents : l'un correspond aux comportements sociaux d'isards, des animaux proches des chamois. Nous disposons de plus de quinze ans de données avec une granularité variant entre un jour et une semaine, ce qui en fait donc un réseau avec une dynamique très forte. Les mêmes données ayant déjà été étudiées par des éthologues spécialistes du domaine par des méthodes d'analyse spatiale[CBG+07], nous pouvons utiliser ces résultats pour valider les communautés que nous trouvons.

La seconde application est menée sur une plateforme japonaise d'échange de vidéos, type Youtube, appelée Nico Nico Douga [HTN08]. Nous disposons des informations pour la totalité des vidéos du réseau sur une période de plus de deux ans, soit plus de quatre millions. En utilisant les mots-clefs attribués à ces vidéos, nous créons un réseau temporel de termes, et détectons des communautés qui correspondent à des sujets populaires sur le réseau. Cela nous permet de valider certains aspects des communautés trouvées, comme la date à laquelle nous détectons des événements que nous pouvons facilement faire correspondre à des événements réels dont nous connaissons la date [CTHA12].

Ces deux applications nous ont permis d'identifier un nouveau type d'opération possible sur les communautés, la résurgence.

D'autres contributions sont également proposées, dans le domaine de la visualisation de communautés dynamiques ou de la création de réseaux temporels à partir de séquences d'interactions, par exemple. Elles découlent de besoins que nous avons rencontrés en essayant de répondre à notre problématique principale.

Structure du mémoire

Le premier chapitre de cette thèse est un état de l'art des méthodes existant pour la détection de communautés. Ses trois sections principales concernent le partitionnement (découpage du graphe en groupes de nœuds, chaque nœud appartenant à un et un seul groupe), la détection de communautés recouvrantes (un nœud peut appartenir à plusieurs communautés), et la détection de communautés dynamiques. Pour le partitionnement, seules les méthodes les plus connues seront présentées. L'état de l'art sera en revanche aussi proche que possible de l'exhaustivité pour les deux autres types d'algorithmes, qui concernent plus directement notre travail, et qui sont actuellement les domaines les plus actifs.

La dernière section de l'état de l'art est consacrée à une analyse critique des méthodes existantes, en mettant l'accent sur leurs forces et faiblesses.

Dans une deuxième partie, nous présenterons la proposition principale de cette thèse, un méta-algorithme pour la détection de communautés dynamiques dans des réseaux temporels, appelé iLCD. Nous commencerons par présenter la démarche qui nous a conduit à cette solution, en explicitant les problèmes à surmonter, ainsi que les points communs - et les différences - avec les solutions précédentes. Nous poursuivrons en proposant deux implémentations de cet algorithme, une première version ne permettant pas d'analyser tous types de réseaux, que nous avons historiquement développé en premier, et une seconde version plus élaborée.

La dernière section de cette partie est consacrée à présenter certains aspects pratiques de la détection de communautés dynamiques. En premier lieu, nous parlerons des données utilisées en entrée par un algorithme de détection dynamique, qui nécessitent un format spécifique, ainsi que

de la façon d'obtenir un réseau temporel à partir de séquences d'interactions. Nous finirons par discuter du traitement des résultats d'un algorithme de détection de communautés dynamiques, en particulier pourquoi des outils de visualisation sont indispensables à ce traitement. Nous présenterons les outils de visualisation que nous avons été amenés à concevoir durant cette thèse.

La troisième partie concerne la validation des communautés trouvées par iLCD sur des graphes réels. Notre approche consiste à dire qu'il n'existe pas une solution unique optimale sur des graphes réels complexes, contrairement aux graphes générés, pour lesquels la structure en communauté est unique. Nous proposons donc deux solutions pour évaluer la pertinence des communautés trouvées. La première solution est générique, et consiste à utiliser des métriques pour comparer différents découpages en communautés, sur n'importe quel réseau réel. La seconde solution consiste à faire évaluer les résultats de différents algorithmes connus — iLCD compris — sur des réseaux réels par des personnes aptes à juger de la qualité de ces solutions. Il s'agit plus précisément d'une application pour la plateforme Facebook, dans laquelle les utilisateurs peuvent évaluer les communautés découvertes au sein même de leur réseau social.

Enfin, dans la quatrième partie, nous verrons l'apport que représente la détection de communautés dynamiques au travers de deux applications à des cas réel, qui sont complémentaires. La première consiste à étudier un réseau d'animaux (des isards) ayant un comportement social. L'intérêt de ce réseau est qu'il court sur une longue période — plus de dix ans — et ne concerne que peu d'individus, ce qui permet d'étudier les résultats en détail. En collaboration avec des éthologues experts du domaine, nous étudions l'évolution des communautés au sein de la population.

La seconde application est réalisée sur un réseau de beaucoup plus grande taille, puisqu'il s'agit de données extraites depuis un large réseau social japonais de partage de vidéos, pour lequel nous possédons l'intégralité des informations sur une période de plus de deux ans. En utilisant les mots-clefs liés aux vidéos, nous en déduisons un réseau de termes, dans lequel les communautés représentent des sujets populaires au sein du réseau, c'est à dire les sujets à propos desquels les utilisateurs publient de nombreuses vidéos. Il s'agit à notre connaissance de la première application de la détection de communautés dynamiques à un réseau temporel réel de grande taille.

1

Notations et Définitions

Contents

Résumé du chapitre Ce chapitre est destiné à faciliter la lecture de ce document.

Il se décompose en deux sections.

Dans la première, nous présenterons les notations utilisées tout au long de ce document, notations concernant les réseaux, les propriétés des nœuds et des liens de ces réseaux, ainsi que des notations spécifiques aux communautés.

Dans la seconde section, nous donnerons un certain nombre de définitions de termes, que nous serons amenés à utiliser dans le document, et qui pourraient ne pas être suffisamment explicites par eux-mêmes, ou paraître imprécis.

1.1 Notations

Après avoir introduit les notations classiques sur les réseaux, nous introduirons une notation nous permettant de décrire avec plus de facilité les communautés et leurs propriétés.

1.1.1 Notations pour les graphes et leurs propriétés

Les graphes traités dans ce document sont des graphes non orientés et non pondérés, sauf indication contraire. Ce sont également des graphes simples, c'est à dire qu'ils ne contiennent pas de boucles (arêtes dont les deux extrémités correspondent au même nœud) ni d'arêtes parallèles (arêtes distinctes ayant les deux mêmes extrémités).

- $G(V, E)$, un graphe non dirigé, avec V l'ensemble de ses sommets (nœuds) et E l'ensemble de ses arêtes (liens), prises dans VxV.
- $n = |V|$ le nombre de nœuds du graphe
- $m = |E|$ le nombre de liens du graphe
- (i, j) une arête du graphe. i et j sont dits adjacents, ou voisins.

1.1.2 Notations des propriétés des nœuds et des liens

- $\xi_{i,j}$ une fonction booléenne qui associe à toute arête (i, j) de $G(V, E)$:
  - la valeur 1 si $(i, j) \in E$
  - 0 sinon

  Comme nous travaillons dans des graphes non dirigés, $\xi_{i,j} = \xi_{j,i}$
- $N(i)$ les voisins (ou successeurs) du nœud i.
- $d(i)$ le degré du nœud i, $|N(i)|$

1.1.3 Notations des communautés et de leurs propriétés

- $C(V_C, E_C)$, C est une communauté de G, correspondant au sous-graphe induit par V_C dans G. V_C est donc un ensemble de nœuds appartenants à V et E_C l'ensemble des liens appartenant à E dont les nœuds extrémités appartiennent à V_C.
- P, l'ensemble des communautés de G.
- $E(V_C)$ l'ensemble des liens existants entre tous les nœuds de V_C. Pour $C(V_C, E_C), E_C = E(V_C)$
- n_C le nombre de nœuds dans C, $n_C = |V_C|$
- $N_C(i)$ les voisins de i appartenant à C, $\{j \in N(i) \wedge j \in C\}$
- $d_C^{ext}(i)$ le nombre de liens externes de i par rapport à la communauté C, $d_C^{ext}(i) = |\{i : i \in V_C, \{(i, j) : j \notin V_C\}\}|$

- $d_C^{int}(i)$ le nombre de liens internes de i par rapport à la communauté C, $d_C^{int}(i) = |\{i : i \in C, \{(i,j) : j \in C\}\}|$
- C^{ext}, le nombre de liens externes de C, $C^{ext} = \sum_{i \in C} d_C^{ext}(i)$
- C^{int}, le nombre de liens internes de C, $C^{int} = |E_C| = \frac{\sum_{i \in C} d_C^{int}(i)}{2}$
- cs_i l'ensemble des communautés auxquelles i appartient, $cs_i = \{C : C \in P \wedge i \in V_C\}$
- $\delta(cs_i, cs_j)$ indique l'appartenance des nœuds i et j à la même communauté. $\delta(cs_i, cs_j) = 1$ si i et j ont au moins une communauté en commun, $\delta(cs_i, cs_j) = 0$ sinon.

1.2 Définitions

1.2.1 Réseau de terrain

Nous définissons un réseau de terrain par opposition à un réseau artificiel, ou réseau généré. Un réseau de terrain est un réseau constitué à partir de données collectées, et qui correspond à une réalité de terrain. Par opposition, les réseaux générés ne correspondent à aucune donnée du monde réel.

1.2.2 Grand graphe

Étudier des graphes composés d'une poignée de nœuds et des graphes qui en contiennent des millions demande des approches souvent différentes. Dans cette thèse, nous étudions ce que nous appelons de grands graphes. Cette expression est parfois utilisée dans la littérature, sans que l'on puisse vraiment y attacher une limite précise. La limite à partir de laquelle un graphe est considéré comme grand dépend par exemple du domaine dans lequel on se situe, et du traitement que l'on souhaite effectuer. Ici, il s'agit surtout de faire la distinction avec des graphes qui ne comporteraient que quelques dizaines de nœuds. Une définition arbitraire serait donc de dire qu'un graphe est grand à partir de quelques milliers de nœuds, bien que notre objectif soit de pouvoir traiter des graphes contenant des millions de nœuds et de liens.

1.2.3 Réseau évoluant fortement

Nous utilisons ce terme pour faire la distinction entre, d'un côté, un réseau pour lequel nous disposons d'un petit nombre d'étapes d'évolution, par exemple un réseau de citation pour lequel nous avons un instantané par an, et, d'un autre côté, des réseaux pour lesquels nous disposons de tous les détails de son évolution. S'il est toujours arbitraire de donner une définition précise de cette notion, on peut définir un réseau évoluant fortement comme un réseau pour lequel le nombre d'étapes d'évolution est supérieur au nombre total de nœuds du réseau.

1.2.4 Réseau petit monde (*Small-world network*)

Selon les auteurs, la notion de réseau petit monde peut être légèrement différente. Le consensus général est de les caractériser par les propriétés suivantes :

- Faible distance entre les nœuds : le nombre moyen de pas nécessaire pour joindre deux nœuds du réseau choisis aléatoirement par la plus courte chaîne reste très faible, quelle que soit la taille du réseau. Cette distance moyenne est de l'ordre de $ln(n)$, le logarithme du nombre de nœuds du réseau. En pratique, des études récentes sur le réseau Facebook [UKBM11] tendraient à montrer que ce nombre peut être encore inférieur. Les réseaux aléatoires vérifient également cette propriété.

– Loi de puissance de la distribution des degrés des nœuds du réseau [BA99]. Cette propriété n'est cependant pas vérifiée pour certains réseaux. Facebook, par exemple, présente une distribution en loi de puissance tronquée, étant donnée l'absence de hubs de très fort degré.
– Fort clustering : cette propriété signifie que les nœuds ont tendance à créer des structures locales denses. Cette propriété peut venir notamment de la transitivé : s'il existe un lien (a, b) et un lien (b, c), alors la probabilité qu'un lien (a, c) existe est renforcée.
– Structure en communauté : cette propriété est fortement liée à la précédente. Dans un réseau petit monde, on observe des structures mésoscopique, c'est à dire des ensembles de nœuds fortement liés entre eux et plus faiblement liés au reste du réseau. Se reporter au chapitre 3.2 pour une définition plus approfondie des communautés.

1.2.5 Centralité d'intermédiarité des liens (*edge betweenness*)

Cette mesure de centralité d'un lien représente le nombre de plus courtes chaînes entre deux nœuds distincts du graphe passant par ce lien. Elle est notamment utilisée par Girvan et Newman dans leur algorithme de détection de communautés [GN02].

1.2.6 Modularité

La modularité est une métrique souvent utilisée pour évaluer la qualité d'un découpage en communautés. Elle a initialement été introduite dans [GN02]. Elle est définie comme la différence entre la proportion des liens se trouvant à l'intérieur des communautés et la proportion des liens qui tomberaient à l'intérieur de ces mêmes communautés, en moyenne, dans un graphe aléatoire de même distribution de degrés :

$$Q = \frac{1}{2m} \sum_{i,j \in V} \left[\xi_{i,j} - \frac{d(i)d(j)}{2m} \right] \delta(cs_i, cs_j)$$

On peut également exprimer cette fonction comme la somme des modularités de chaque communauté :

$$Q = \sum_{C \in P} \left[\frac{C^{int}}{m} - \left(\frac{C^{ext} + 2C^{int}}{2m} \right)^2 \right]$$

1.2.7 Instantané de réseau dynamique

Une méthode classique pour étudier des réseaux qui changent au cours du temps est de découper cette évolution en plusieurs instantanés, des réseaux statiques qui représentent son état à un moment donné. Selon les cas, ces instantanés peuvent en fait correspondre à deux situations différentes : dans le cas d'un réseau dont les liens ont une existence durable, comme, par exemple, les liens formés par les listes de contacts de la plateforme Facebook, il suffit de collecter directement ce réseau à l'instant qui nous intéresse.

Dans d'autres cas, lorsque les liens correspondent à des interactions sans durées, comme, par exemple, un réseau d'envoi de mails, des réseaux statiques sont créés à partir de la concaténation de tous les liens ayant existé sur une période. Par exemple, le réseau statique correspondant à une période d'un mois sera défini comme l'ensemble des liens ayant existé au moins une fois durant cette période. (Dans le réseau de mail, deux personnes ayant échangé un mail au moins une fois durant cette période seront considérées comme connectées par un lien dans le réseau statique correspondant à la période.)

Bien que ces deux possibilités ne soient pas équivalentes, nous les dénommerons toutes deux instantanés de réseau, car ils correspondent *in fine* à la même chose : les relations considérées comme pertinentes à un moment donné de l'évolution du réseau.

1.2.8 Clique, Clique maximale

Dans un graphe, une clique est définie comme un ensemble de nœuds, tous connectés les uns aux autres, c'est à dire un sous-graphe complet. Une clique maximale est une clique de taille k non incluse dans une clique de taille $k+1$.

1.2.9 Composante connexe

Une composante connexe CC d'un graphe $G(V, E)$ est un sous-ensemble **maximal** de sommets tels que deux quelconques d'entre eux soient reliés par une chaîne, c'est à dire par une suite de nœuds et de liens. Si $x \in CC$:

- $\forall y \in CC$, il existe une chaîne reliant x à y
- $\forall z \in V \setminus C$, il n'existe pas de chaîne reliant x à z

2

Etat de l'art

Contents

Résumé du chapitre Dans ce chapitre, nous présentons l'état de l'art se rapportant à la détection de communautés, en couvrant la plupart de ses aspects, tout en nous consacrant plus particulièrement aux avancées les plus récentes du domaine, qui concernent directement la problématique de cette thèse.

Dans la section 1, nous introduisons le problème de la détection de communautés, ainsi que son historique. La section 2 sera consacrée aux détections de communautés statiques et sans recouvrement, qui ont représenté la plus grande partie des travaux publiés sur le sujet, mais que nous n'aborderons pas en profondeur, car ils sont assez loin de nos propres travaux. La section 3 sera consacrée aux méthodes statiques hiérarchiques, que nous aborderons également rapidement.

Nous approfondirons en revanche la question des algorithmes de détection de communautés statiques avec recouvrement (section 4) et celle des algorithmes de détection de communautés dynamiques (section 5), qui sont au cœur de la problématique de cette thèse. Enfin, dans la section 6, nous prendrons le temps de décrire les faiblesses des algorithmes existants, en particulier en ce qui concerne les aspects dynamiques.

2.1 Généralités sur la détection de communautés

2.1.1 Partitionnement de graphe

Le problème de la détection de communauté dans les réseaux est un sujet relativement récent, mais qui a très rapidement conduit à une grande quantité de travaux.

A l'origine, une version plus simple de ce problème avait été explorée avec le partitionnement de graphe (On peut noter que parfois, aujourd'hui, partitionnement de graphe, détection de communautés et *graph clustering* en anglais sont souvent utilisés indifféremment. Historiquement, le terme partitionnement de graphe se rapporte plutôt aux méthodes informées (nombre et taille des clusters connus), tandis que détection de communautés se rapporte aux méthodes non-informées.)

Le partitionnement de graphe consiste à trouver, pour un graphe donné, un partitionnement optimal en k partitions, k étant donné. Pour évaluer la qualité d'un partitionnement, on prend simplement le nombre de liens situés entre les partitions, c'est à dire les liens reliant un nœud appartenant à une partition P_1 à un second nœud appartenant à une partition P_2, avec $P_1 \neq P_2$.

Sans autres contraintes, une solution triviale consisterait à créer une partition contenant $n-(k-1)$ nœuds, et chacune des autres partitions contiendrait un seul nœud, parmi ceux ayant le moins de liens. Cette solution satisfait efficacement le problème de la minimisation des liens inter-communautés, mais ne présente évidemment que peu d'intérêt.

Dans un premier temps, une solution proposée consistait à chercher des partitions de taille donnée, souvent de même taille. La méthode la plus connue est celle de Kernighan-Lin [KL70]. Cette contrainte étant trop stricte pour être efficace dans des cas réels, elle a ensuite été relâchée [WC89], de manière à chercher un nombre donné de communautés, mais sans avoir à en préciser la taille exacte.

2.1.2 Détection de communautés

Le partitionnement de graphe a d'importantes limites, qui ont conduit à s'intéresser à un problème lié mais plus complexe, celui de la détection de communautés. Lorsque l'on étudie des réseaux de terrain, en particulier lorsque ceux-ci sont de grande taille et/ou représentent des données complexes (réseaux sociaux, réseaux biologiques, réseaux informatiques, etc.), le

nombre de groupes que l'on cherche à obtenir ne peut être connu à l'avance. Dans ces réseaux, le nombre de communautés existant est en fait, en lui-même, un résultat important.

C'est dans l'article de Girvan et Newman de 2002 [GN02] que le problème a pour la première fois été posé dans les termes que l'on connaît maintenant. Les auteurs partent du principe que dans les réseaux petit monde réels, il existe une structure en communauté. Ils se demandent alors comment détecter automatiquement cette structure.

Le nouveau problème ainsi posé peut se définir de la manière suivante : pour un réseau donné, comment le décomposer en un nombre inconnu de groupes de nœuds de manière à ce que ces groupes de nœuds correspondent à l'agencement topologique du réseau. Le problème de cette définition tient à ce que, contrairement au partitionnement de graphe, il n'y a pas de critère concret permettant d'évaluer la qualité du découpage, tel que, par exemple, le nombre de liens entre les communautés. En effet, il est très vite apparu qu'un critère aussi simple ne conduisait pas à des résultats concluants. Pour plus d'informations sur le problème de la nature des communautés, se reporter au chapitre 3.2.

2.1.3 Classification des méthodes de détection de communautés

Depuis 2002 et l'introduction du premier algorithme historique par Girvan et Newman [GN02], un nombre très important d'algorithmes ont été proposés. L'état de l'art de Santo Fortunato [For10] paru en 2009 référence plus de 50 méthodes, sans être exhaustif. L'article de Xie et al [XKS11] de 2011 compare 10 méthodes parues après 2009, dont iLCD, et seules les méthodes les plus connues et dont le code était disponible ont été testées. Selon certains auteurs, il y aurait plus de 250 algorithmes de détection de communautés publiés, et ce nombre continue à croître. Ne pouvant pas décrire ici toutes les méthodes, nous allons présenter une sélection d'entre elles, parmi les plus connues.

Il est également important d'adopter une catégorisation. On peut trouver de nombreuses méthodes pour les classifier, mais il est difficile d'en trouver une qui ne soit pas ambiguë.

Le classement le plus pertinent dans le cadre de cette thèse concerne les propriétés des communautés trouvées. La présentation de l'ensemble des méthodes existantes que nous avons adoptée va donc suivre l'organisation hiérarchique suivante :

- **Détection de communautés statiques**
  - Détection de communautés statiques, sans recouvrement
    - Optimisation de la modularité
    - Autres méthodes
  - Détection de communautés statiques, hiérarchiques
  - Détection de communautés statiques, avec recouvrement
    - Méthodes basées sur des cliques
    - Méthodes basées sur des communautés de liens
    - Méthodes basées sur des extensions/rétractions de graines
    - Méthodes basées sur des modèles génératifs
    - Méthodes basées sur la propagation de labels
- **Détection de communautés dynamiques**
  - Approches par détections statiques successives
  - Approches par étude simultanée de toutes les étapes d'évolution
  - Approches par détections statiques informées successives
  - Approches travaillant sur des réseaux temporels

Historiquement, dans les premières méthodes, la solution recherchée était un simple partitionnement : chaque nœud du réseau appartenait à une et une seule communauté, et il n'existait qu'un seul découpage possible.

Un peu plus tard, il est apparu qu'une telle décomposition ne pouvait suffire à représenter la complexité des réseaux de terrain. En effet, sur les grands graphes, il est fréquent que l'on puisse proposer différents niveaux de décomposition, le premier niveau étant composé de communautés contenant —relativement — peu de nœuds. Des niveaux supérieurs peuvent aussi être définis, en assemblant des communautés du niveau inférieur pour former de nouvelles communautés, plus étendues, qui sont donc des « communautés de communautés ». On appelle les algorithmes capables d'un tel découpage les algorithmes de détection de communautés hiérarchiques.

Le problème de l'appartenance multiple des nœuds s'est posé un peu plus tardivement, le premier algorithme permettant de fournir une telle solution ayant été publié en 2005 [PDFV05]. En effet, dans la plupart des graphes de terrain, il existe de nombreux nœuds qui peuvent logiquement appartenir à plusieurs communautés, comme cela était connu depuis longtemps dans le domaine des réseaux sociaux par exemple. Cette troisième catégorie est donc celle des algorithmes de détection de communautés avec recouvrement.

Enfin, la catégorie la plus récente concerne les algorithmes de détection de communautés dynamiques. De fait, les méthodes précédentes n'étaient capables que de trouver un découpage sur un graphe défini pour un moment donné. Or, beaucoup de graphe de terrain ont en fait la propriété de changer, d'évoluer au cours du temps, des nœuds et des liens pouvant apparaître ou disparaître. Trouver des communautés dans de tels réseaux demande de prendre en compte leurs différentes étapes d'évolution, de manière à donner des communautés cohérentes non pas à un moment donné, mais sur une période donnée, avec leurs éventuelles modifications au cours du temps. L'étude des réseaux dynamiques, ou réseaux temporels, est un domaine qui attire aujourd'hui de plus en plus d'intérêt, de par les nouvelles opportunités — et les nouveaux challenges — qu'il propose.

Si la méthode proposée dans cette thèse s'inscrit dans cette dernière catégorie, il est utile et même nécessaire de faire un état de l'art des techniques existantes pour les catégories précédentes, d'une part parce que la plupart des méthodes récentes sont basées sur des méthodes plus anciennes, et d'autre part parce que des méthodes pouvant fournir des résultats plus évolués — par exemple, des communautés avec recouvrement comparé à un algorithme sans recouvrement — ne sont pas forcément aussi efficaces. Sur certains réseaux, il est possible que les méthodes sans recouvrement donnent des résultats très intéressants alors que les méthodes avec recouvrement proposées jusqu'ici ne seront pas aussi pertinentes. Pour dire les choses autrement, les nouvelles méthodes qui ont été proposées au cours des dernières années n'ont, dans la plupart des cas, pas rendu obsolètes les méthodes précédentes. Elles permettent en revanche de traiter plus de problèmes différents, peuvent fournir des résultats pertinents, contenant souvent plus d'informations que les solutions antérieures (appartenances multiples, informations temporelles).

Dans les sections qui suivent, nous allons faire un tour d'horizon des méthodes existantes. Ne pouvant même nous approcher de l'exhaustivité, nous avons choisi de passer rapidement sur les méthodes sans recouvrement, pour nous concentrer sur les algorithmes les plus récents et dont les problématiques se rapprochent le plus de ce travail de thèse, les algorithmes avec recouvrement et les algorithmes détectant des communautés dynamiques. Dans chaque section, nous citerons ou présenterons succinctement plusieurs algorithmes, avant de présenter en détail un ou deux algorithmes, que nous pensons être les plus emblématiques. Une littérature abondante existe cependant sur le sujet de la comparaison et de la classification d'algorithmes de détection de communautés, se rapporter par exemple à [LF09, XKS11, For10, LLM10, DDGDA05, Sch07]

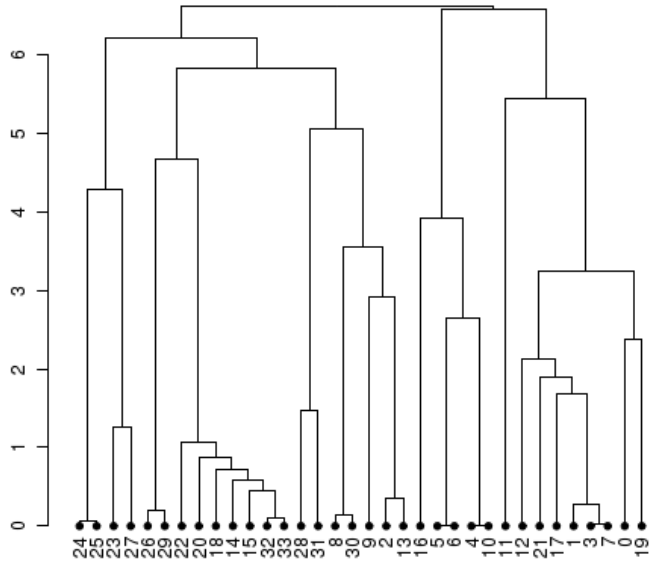


FIGURE 2.1 – *Exemple de dendrogramme : Zachary Karate Club.*

2.2 Détection de communautés statiques, sans recouvrement

La première méthode moderne pour la détection de communautés, encore utilisée dans plusieurs domaines et sur laquelle de nombreuses méthodes postérieures sont basées, est celle proposée en 2002 par Girvan et Newman. Il convient de présenter cette méthode, car, bien que certaines méthodes proposées ultérieurement donnent de meilleurs résultats, beaucoup reprennent son principe.

Il s'agit d'un algorithme divisif : au départ, on considère que tous les nœuds du réseau appartiennent à une seule communauté. On va ensuite retirer successivement les liens, un à un, en retirant toujours celui de centralité d'intermédiarité maximale. Petit à petit, le graphe devient donc non-connexe, et chacune des composantes connexes ainsi formées est une communauté. Le résultat est donc un dendrogramme, contenant à sa racine une communauté, deux à l'étape suivante, puis 3, et ainsi de suite jusqu'à ce qu'à ce que chaque nœud forme sa propre communauté (racines de l'arbre). On peut observer un exemple de dendrogramme généré avec la bibliothèque igraph dans la figure 2.1.

C'est en cherchant à couper ce dendrogramme de manière à obtenir un seul découpage en communauté « optimal » que les auteurs introduisirent une métrique qui occupe une place majeure dans le domaine de la détection de communauté : la modularité. (Définition dans le chapitre 1.2. Se reporter également au chapitre 2.6.1 pour une critique de cette métrique)

2.2.1 Optimisation de la modularité

Certains auteurs considérèrent alors que, puisque la modularité était utilisée pour choisir, parmi une série de découpages en communautés possibles (les différents niveaux du dendro-

gramme), quel était le meilleur, cette métrique pouvait être considérée comme une définition de ce qu'est une « bonne communauté ». Dès lors, ils eurent l'idée de chercher directement le découpage en communautés correspondant à la valeur maximale de la modularité pour un graphe donné. Le problème de la détection de communauté devenait donc un problème mathématique d'optimisation, consistant à explorer un espace de solutions pour trouver celle correspondant au maximum de modularité. Le problème ayant été prouvé NP-Complet, [BDG+07] des solutions furent proposées pour trouver, avec la complexité la plus faible possible, une solution considérée comme proche du maximum [New04, CNM04, DDGA06, WT07, BGLL08, GSPA04, DA05].

Cependant, les limites de ces méthodes apparurent bientôt : sur certains graphes de terrain, les résultats ne paraissaient pas correspondre à ce qui était attendu. Le problème de ces méthodes vient simplement du fait que la modularité n'est qu'une définition arbitraire de ce que peuvent être de bonnes communautés, qui se révéla fortement imparfaite. La preuve fut apportée par la publication de l'article de Fortunato et Barthelemy [FB07], qui démontra la limite de résolution de la modularité. Le problème mis en évidence dans ce travail est que la modularité présuppose certaines propriétés des communautés à partir des propriétés du réseau à étudier. Pour une taille de réseau donné ayant une densité donnée, la modularité ne pourra pas trouver de communautés inférieures à une certaine taille. Toute communauté plus petite, même nettement séparée du réseau, sera fusionnée avec d'autres communautés pour obtenir des communautés de la taille attendue par la modularité. Ces problèmes furent confirmés lors de tests sur des réseaux générés suffisamment réalistes [LF09].

2.2.2 Autres méthodes

Des dizaines d'autres méthodes ont été proposées, n'utilisant pas la modularité. On peut citer par exemple les algorithmes spectraux, tels que [CSCC05, DM04, JKV01], les modèles de spin [RB04, SJN06], les marches aléatoires [Zho03, PL05], ceux utilisant l'inférence statistique [Has06, NL07], et bien d'autres [RAK07, BB05, GMNN08]. Les références données ici ne le sont qu'à titre d'exemple, et sont loin d'être exhaustives. Pour une présentation très complète et détaillée d'un grand nombre d'algorithmes, se reporter à l'excellente *review* de Santo Fortunato : [For10]

Une méthode parmi les plus connues, et souvent considérée comme la plus efficace, est l'algorithme Infomap de Rosvall et Bergstrom [RB08]. Cette méthode est, tout d'abord, relativement rapide, ce qui permet de l'appliquer à des grands réseaux de terrain. Mais ce qui a fait sa popularité est sans conteste son efficacité sur des tests de benchmark. En 2009, Lancichinetti et al [LFR08] proposèrent un générateur de graphes avec communautés suffisamment réaliste pour que des algorithmes de détection de communautés puissent être comparés sur leur efficacité à retrouver les communautés correctes sur ces graphes[LF09, XKS11, NC10]. Et InfoMap a montré sur ces résultats qu'il était l'algorithme le plus performant, et ce sur la quasi-totalité des cas de figure, avec un écart important par rapport aux méthodes d'optimisation de la modularité (sauf l'algorithme Louvain, présenté dans la section suivante).

Le principe derrière InfoMap est très élégant : si l'on considère un marcheur aléatoire qui se déplace sur un réseau, et que ce réseau a une structure en communauté, alors le marcheur aléatoire va avoir tendance à rester à l'intérieur des communautés. Ce principe découle de la définition intuitive des communautés, selon laquelle elles sont des groupes de nœuds fortement connectés et plus faiblement connectés au reste du réseau. Une conséquence intéressante de cette définition est que si le graphe étudié est un graphe aléatoire, un marcheur aléatoire ne restera pas « coincé » dans une communauté, et InfoMap sera donc capable de dire que le graphe n'est pas divisible en communautés pertinentes, contrairement aux méthodes basées sur une optimisation

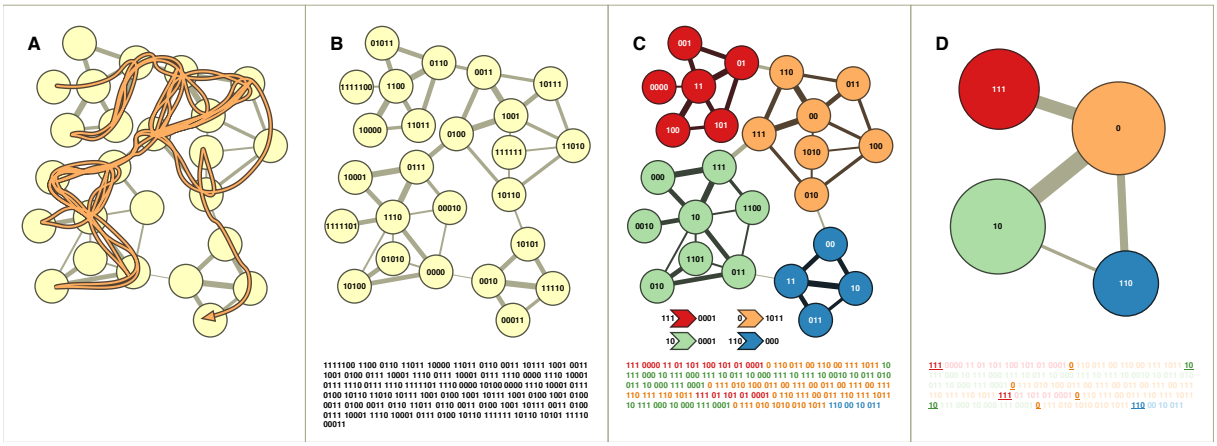


FIGURE 2.2 – *Illustration du fonctionnement d'InfoMap (Issu de l'article [RB08]). L'image A représente le parcours d'un marcheur aléatoire, que l'on souhaite encoder. L'image B présente un moyen basique de faire cet encodage : on attribue un identifiant binaire à chaque nœud, et on utilise la séquence des identifiants pour encoder le parcours (en noir, en-dessous). Les images C et D montrent comment optimiser cet encodage, en attribuant des identifiants aux clusters, et, pour chaque nœud, un identifiant interne a son cluster. On voit que l'encodage (en-dessous, coloré) est plus efficace que dans le cas B. C'est la solution conduisant à l'encodage minimal qu'Infomap cherche à trouver.*

de la modularité, qui trouvent toujours des communautés, quel que soit le graphe étudié.

Plus concrètement, InfoMap utilise des méthodes venues de la théorie de l'information, et cherche à trouver un encodage optimal du graphe. Pour ce faire, il cherche à minimiser la longueur de la description du chemin parcouru par un marcheur aléatoire, les nœuds étant décrits selon une numérotation préfixée. Chaque nœud est ainsi décrit par :

- l'identifiant de la communauté à laquelle il appartient
- un identifiant unique au sein de cette communauté

Lorsque l'on décrit le déplacement d'un marcheur, on commence par donner l'identifiant de la communauté dans laquelle il se trouve, puis l'identifiant du nœud sur lequel il est. Tant que le marcheur reste au sein de la même communauté, on peut décrire le nouveau nœud qu'il atteint par son identifiant « local » au sein de la communauté, sans répéter l'identifiant de celle-ci. Dès que son déplacement l'emmène hors de la communauté, pour décrire le nouveau nœud sur lequel il se trouve, il faut donner l'identifiant de la nouvelle communauté, puis l'identifiant du nœud au sein de la communauté. Il s'agit d'un système simple tel que celui utilisé pour les adresses postales : il existe de nombreuses rues Victor Hugo en France, (identifiant de nœud), mais elles peuvent être différenciées par l'élément dans lequel elles sont incluses (nom de ville, correspondant aux identifiants de communautés). Le fonctionnement de l'algorithme est illustré dans la figure 2.2.

On comprend aisément qu'avec ce système de description de déplacement, quitter des communautés se révèle très coûteux. Un bon découpage en communauté consistera donc à rendre ces déplacements les plus rares possible. InfoMap peut fournir encore une nouvelle information : on peut évaluer la « qualité » de la compression réalisée, et donc la pertinence du découpage correspondant, en évaluant le taux de compression de la description d'une marche aléatoire permis par le découpage en communautés. De par son efficacité démontrée sur des réseaux de terrain et générés, par sa rapidité lui permettant de traiter de très grands graphes, ainsi que par sa pos-

sibilité d'évaluer la qualité du découpage fourni, on peut considérer que cet algorithme est une solution complète au problème le plus simple de la détection de communauté, celui consistant à trouver un découpage unique, statique et sans recouvrement de communautés.

Cette méthode sera utilisée dans le chapitre Validation de ce travail (4), avec d'autres, pour comparer les communautés qu'elle trouve avec celles de notre algorithme.

2.3 Détection de communautés statiques, hiérarchiques

Dans la plupart des réseaux de terrain de grande taille, il n'existe pas un mais plusieurs niveaux pertinents de découpage en communautés. Par exemple dans un réseau social de grande taille, on pourrait trouver de petites communautés correspondant à des quartiers, à un niveau supérieur des communautés plus grandes correspondant à des villes, puis les départements, les régions et enfin les pays. Un résultat de ce type a par exemple été obtenu avec les régions et les départements par Vincent Blondel, en travaillant sur des réseaux de communications par téléphones cellulaires, sur plusieurs pays (France, Angleterre, Belgique). Des résultats particulièrement intéressants apparaissent parfois, comme la nette séparation, connue mais non institutionnelle, de la Belgique en communautés correspondant aux locuteurs des différents idiomes. [EEBL11, BGLL08].

Les premières approches proposées contenaient en fait déjà cette idée de hiérarchies : en utilisant des approches agglomératives [New04] ou divisives [Zho03], on crée des dendrogrammes, dans lesquels chaque niveau correspond à la division ou à la fusion de deux communautés du niveau précédent. Ces dendrogrammes correspondent bien à une décomposition hiérarchique des communautés, chaque petite communauté étant incluse dans une communauté de plus grande taille. Le problème de cette méthode est que, pour la plupart des niveaux du dendrogramme, les communautés formées ne sont pas significatives. On peut le comprendre aisément par un exemple imaginaire pour lequel on aurait un réseau composé de deux communautés très clairement définies, par exemple deux cliques, ou deux graphes aléatoires denses, reliées par un seul lien. Il n'existe dans ce cas qu'un seul niveau de découpage pertinent. Pourtant, les approches à base de dendrogrammes proposeront inévitablement des étapes intermédiaires, qui leur permettront de parvenir au résultat voulu. Par exemple, sur la figure 2.1, il est clair que toutes les étapes ne peuvent pas être pertinentes. Une solution consiste donc à travailler sur un dendrogramme, mais à chercher non plus une seule coupe pertinente, comme avec la modularité de Girvan et Newman, mais plusieurs [ABL10].

La solution ayant rencontré cependant le plus de succès est différente. Elle consiste à travailler sur plusieurs itérations de détection successive. La première détection va trouver des communautés relativement petites dans le graphe. Dans une deuxième étape, on va créer un méta-graphe dans lequel chaque communauté trouvée va devenir un nœud, et le nombre de liens existant entre deux communautés (liens existants entre des nœuds de ces communautés) va être utilisé pour créer un lien pondéré entre les nœuds représentants ces communautés. Ce processus est illustré dans la figure 2.3.

La méthode la plus connue qui l'utilise est sans conteste la méthode Louvain (aussi appelée BGL) Cette méthode est d'une grande importance dans le domaine, parce qu'elle est aujourd'hui la plus utilisée dans de nombreuses applications : en tant qu'application sur de grands réseaux [HP09, RH10, HS09], dans des applications professionnelles (LinkedIn, pour détecter des communautés dans les contacts des utilisateurs), ou même comme algorithme de base pour des solutions postérieures, par exemple [GDC10, MRM+10] et bien d'autres.

L'objectif recherché est, classiquement, une optimisation de la modularité. Cependant, ce qui

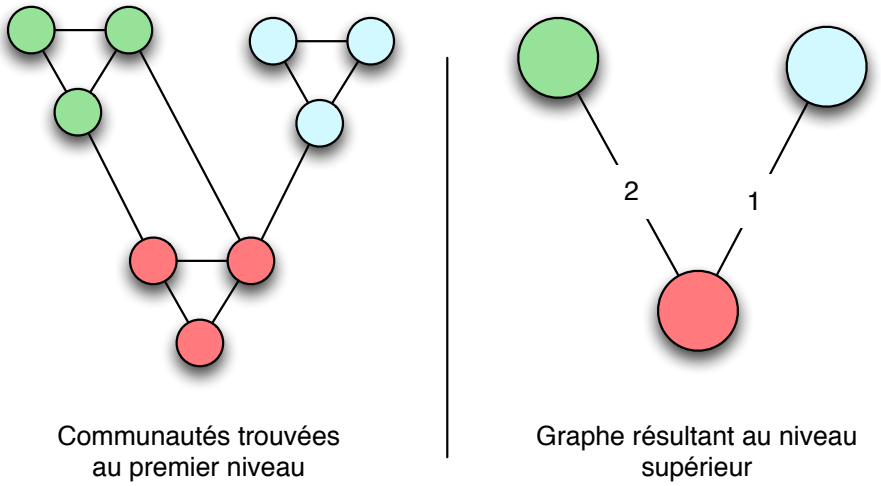


FIGURE 2.3 – *Processus permettant de passer d'un ensemble de communautés sur un graphe à un graphe de niveau supérieur. Chaque nœud du graphe au niveau supérieur correspond à une communauté du niveau inférieur, les liens entre les nœuds correspondent au nombre de liens entre ces communautés au niveau inférieur.*

fait l'originalité de cette méthode est qu'elle effectue une optimisation locale de cette métrique. C'est un algorithme glouton qui est utilisé (*greedy optimization*) : initialement, chaque nœud est considéré comme appartenant à une communauté composée d'un seul nœud : lui-même. Ensuite, pour chaque nœud, on calcule le gain en modularité que l'on peut obtenir en l'intégrant dans la communauté de l'un de ses voisins. Cette évaluation peut se faire localement, ce qui assure une grande rapidité. S'il existe une solution qui améliore la modularité, elle est prise. (Dans le cas où plusieurs solutions existent, c'est celle apportant le gain le plus important en modularité qui est choisie.) Lorsqu'il n'est plus possible d'améliorer la modularité de cette manière, une première étape de décomposition est atteinte. On calcule alors le méta-réseau, comme expliqué précédemment, et on relance le même algorithme sur ce nouveau réseau.

Au final, on obtient donc plusieurs solutions, représentant les différents niveaux de découpage. Même pour de très grands réseaux, le nombre de niveaux reste généralement faible (typiquement entre 2 et 5), ce qui permet une analyse assez simple.

Si cette méthode a connu une large diffusion, c'est tout d'abord grâce à sa très faible complexité. C'est une approche gloutonne, et elle s'est révélée capable de traiter des réseaux de centaines de millions de nœuds en quelques minutes. Il s'agit sans conteste de l'une des méthodes les plus rapides proposées à ce jour, avec un avantage de plusieurs ordres de grandeur comparée à la plupart des autres méthodes basées sur l'optimisation de la modularité.

De plus, la modularité globale obtenue pour le niveau de modularité maximal est souvent très bonne comparée aux autres méthodes. Enfin, pour les niveaux de décomposition les plus faibles, la méthode ne souffre pas du défaut connu de la modularité qu'est la limite de résolution. La méthode pourra trouver de petites communautés, même dans un très grand graphe.

Cette méthode sera utilisée dans le chapitre Validation de ce travail (4), avec d'autres, pour comparer les communautés qu'elle trouve avec celles de notre algorithme.

Année	Méthode	C	CL	EG	MG	PL	M
2005	Baumes et al. [BGMI05]			✓			
	Palla et al. [PDFV05]	✓					
2008	Kumpala et al.[KKKS08]	✓					
2009	Kelley et al. [Kel10]			✓			
	Lancichinetti et al. [LFK09]			✓			
	Shen et al. [SCCH09]	✓					✓
	Wang et al. [WJW09]			✓			
2010	Ahn et al. [ABL10]		✓				
	Chen et al. [CLSW10]			✓			✓
	Chen et al. [CSLF10]			✓			
	Evans et al. [EL09]		✓				✓
	Gregory et al. [Gre10]					✓	
	Lee et al. [LRMH10]			✓			
	McDaid et al. [MH10]				✓		
	Padrol et al. [PSPLPMM10]			✓			
	Wu et al. [WLWT10]		✓				
2011	Havemann et al. [HHSG11]			✓			
	Kim et al. [KJ11]		✓				
	Lancichinetti et al. [LRRF11]			✓			
	Latouche et al. [LBA11]				✓		
2012	Xie et al. [XSL11]					✓	

TABLE 2.1 – *Algorithmes de détection de communautés statiques avec recouvrement, classés par année de publication. On observe que la majorité des méthodes proposées l'on été après le début de cette thèse (2009). Les colonnes représentent l'approche utilisée. Légende : **C** : méthodes à base de cliques. **CL** : communautés de liens. **EG** : extension de graines. **MG** : Modèles génératifs. **PL** : Propagation de labels. **M** : méthodes faisant appel à la modularité.*

2.4 Détection de communautés statiques, avec recouvrement

Si l'idée de hiérarchie de communautés était déjà présente implicitement dans les premières méthodes de détection de communautés telles que celle de Newman, il a fallu attendre plus longtemps pour qu'une autre caractéristique des réseaux de terrain soit prise en compte : le recouvrement de communautés. Cette propriété, observée depuis longtemps dans les réseaux sociaux, est présente dans la plupart des réseaux de terrain.

Si l'on pense par exemple à un réseau d'individus dans lesquels les liens ne sont pas limités à un type précis d'interaction, mais peuvent correspondre à tout type de relation (comme on peut le trouver dans les réseaux sociaux Web 2.0 tels que Facebook), il est évident que chaque personne appartient à plusieurs groupes, ou communautés : les personnes de sa famille, ses collègues de travail, ses amis de lycée et de l'université, etc. Même si l'on ne considère que le réseau égocentré de cette personne, il est probable qu'au moins l'une des personnes de sa famille fasse aussi partie de son cercle d'amis du lycée, ou autre appartenance multiple.

Pourtant, la première méthode prenant en compte efficacement le recouvrement n'a été proposée qu'en 2005, par Palla et al. [PDFV05]. Et il a fallu attendre encore quelques années pour que l'importance de ce problème soit vraiment comprise et que de nouvelles solutions soient proposées.

Étant donné le plus faible nombre de méthodes entrant dans cette catégorie, nous pouvons présenter la majorité des travaux publiés à ce jour. Nous n'avons retenu que les méthodes ne nécessitant en entrée ni le nombre de communautés ni de travail de préparation préalable (choix de nœuds cibles, labellisation de nœuds, etc.). Une liste chronologique peut être trouvée dans la table 2.1.

Nous allons commencer par présenter les grandes approches permettant de faire de la détection de communautés avec recouvrement, en présentant rapidement les principaux algorithmes utilisant ces méthodes. Dans la suite, nous allons également présenter en détail deux méthodes particulièrement importantes : OSLOM [LRRF11] et CPM [PDFV05]. CPM tout d'abord, parce qu'elle est historiquement l'une des premières, et parce qu'elle reste encore aujourd'hui une référence largement utilisée. Nous présenterons ensuite OSLOM, parce que c'est un des algorithmes les plus récents, qui a su s'inspirer des travaux précédents, mais aussi parce qu'il semble que ce soit actuellement l'une des solutions les plus efficaces. De plus, ces deux méthodes peuvent être utilisées pour la détection de communautés dynamiques, comme cela sera présenté dans le prochain chapitre. Pour une présentation détaillée d'un certain nombre d'algorithmes, se reporter à [XKS13].

2.4.1 Difficulté à adapter les méthodes sans recouvrement

On pourrait penser que la différence entre méthodes avec communautés non recouvrantes et méthodes avec communautés recouvrantes est peu importante, et qu'il serait donc possible d'adapter facilement les premières pour gérer le recouvrement. Or, il n'en est rien. Sans recouvrement, les communautés sont définies les unes par rapport aux autres. Les limites d'une communauté sont naturellement définies par les limites des autres.

Le cas le plus évident est celui de la modularité. Si l'optimisation de la modularité est une des méthodes les plus populaire dans le cas sans recouvrement, aucune méthode avec communautés recouvrantes ne l'utilise. La modularité peut pourtant être utilisée pour évaluer la qualité d'une communauté prise séparément, cette métrique consistant en effet à comparer la densité interne de la communauté avec la densité moyenne qu'aurait la communauté dans une réseau aléatoire recablé. Cependant, dans le cas de l'optimisation de cette métrique, c'est le fait qu'agrandir une communauté oblige à en réduire une autre qui permet de parvenir à un résultat pertinent. Avec des recouvrements possibles, chaque communauté peut grandir tant qu'elle le souhaite, et le nombre de communautés trouvées peut être grand, par exemple supérieur au nombre de nœuds du réseau. Une version modifiée de la modularité a cependant été utilisée dans certains algorithmes avec recouvrement [NMCM09, SCCH09], mais pas simplement pour l'optimiser.

Ce problème d'adaptation au recouvrement se retrouve pour d'autres approches, telles que les méthodes spectrales, les méthodes agglomératives et divisives, etc.

2.4.2 Méthodes basées sur des cliques

Une première solution proposée pour trouver des communautés recouvrantes a été d'utiliser les cliques du réseau. Les cliques sont des ensembles de nœuds dont tous les membres sont liés à tous les autres. On sait que dans de nombreux réseaux, les relations ont tendance à être transitives : si les nœuds i et j sont connectés à un même nœud k, alors la probabilité qu'un lien existe entre i et j est augmentée, ce qui crée naturellement des cliques. Les communautés étant composées de nœuds fortement connectés entre eux, elles sont riches en cliques, dans les réseaux pour lesquels la transitivité est forte. Une définition caricaturale des communautés utilisée dans certains cas consiste d'ailleurs à considérer que les communautés correspondent aux

cliques maximales du réseau, ce qui fournit des communautés recouvrantes. Certaines approches ont donc essayé d'utiliser les cliques comme élément de base de leurs communautés.

La première de ces méthodes est celle de Palla et al. [PDFV05], CPM, que nous présenterons en détail dans la suite. Elle consiste à d'abord détecter toutes les cliques d'une taille donnée, puis à agglomérer celles ayant suffisamment de nœuds en commun, ce qui est moins restrictif que de simplement considérer les cliques maximales.

Kumpala et al. [KKKS08] ont proposé une optimisation de CPM, plus performante mais reprenant le même mécanisme.

Shen et al. [SCCH09] ont proposé un algorithme appelé EAGLE, qui utilise un dendrogramme de cliques. L'algorithme commence par identifier toutes les cliques maximales. Ce sont les communautés initiales. Ensuite, les communautés ayant le plus fort taux de similarité sont fusionnées, formant de nouvelles communautés, qui, à leur tour, pourront être fusionnées avec des communautés semblables. La coupe optimale du dendrogramme est déterminée en utilisant une version modifiée de la modularité, définie comme :

$$Q_{ov}^{E} = \frac{1}{2m} \sum_{C \in P} \sum_{i,j \in C} \left[\xi_{i,j} - \frac{d(i)d(j)}{2m} \right] \frac{1}{cs_i cs_j}$$

Cette version de la modularité pondère, pour chaque lien, le rôle joué dans le calcul de la modularité de sa communauté par le nombre de communautés auxquelles il appartient.

2.4.3 Méthodes basées sur des communautés de liens

Si les communautés sont définies comme des groupes de nœuds, dans un réseau, on peut également créer des groupes de liens. Une fois ces clusters de liens trouvés, on peut en déduire un découpage des nœuds en communautés recouvrantes. Il suffit de considérer qu'à chaque groupe de lien A correspond une communauté de nœuds C_A, à laquelle appartiennent tous les nœuds à l'extrémité d'au moins un lien de A.

Une première méthode consiste à créer directement un nouvel algorithme pour constituer des clusters de liens. Dans Ahn et al. [ABL10], un dendrogramme est créé en agglomérant les liens les plus semblables. La similarité entre les liens est définie, pour les liens ayant au moins un nœud k en commun, comme :

$$S((i,k),(j,k)) = \frac{|N(i) \bigcap N(j)|}{|N(i) \bigcup N(j)|}$$

Il suffit alors de couper le dendrogramme pour obtenir un découpage des liens en clusters.

Certains auteurs ont trouvé une méthode permettant de réutiliser les algorithmes classiques pour trouver des clusters de liens. Pour ce faire, ils appliquent l'algorithme non plus sur le graphe lui-même mais sur une transformation de ce graphe particulière : le *line graph*. La transformation consiste à faire correspondre, à chaque lien du graphe originel G, un nœud dans le graphe transformé $L(G)$. Un lien existe entre deux nœuds de $L(G)$ si et seulement si les liens correspondants dans G avaient une extrémité commune. On peut observer un exemple de transformation sur la figure 2.4.

Une fois le Line Graph obtenu, n'importe quelle méthode sans recouvrement peut être utilisée pour détecter des communautés, qui, une fois rapportées à G, correspondront à des communautés de liens. Cette méthode est utilisée par Evens et al. [EL09]. Kim et al. [KJ11] reprennent la même idée mais en utilisant l'algorithme InfoMap.

Le principe de ces méthodes est très intéressant, l'idée de considérer des communautés de liens semble pertinente. Si l'on prend l'exemple d'un réseau social, un individu A qui appartiendrait

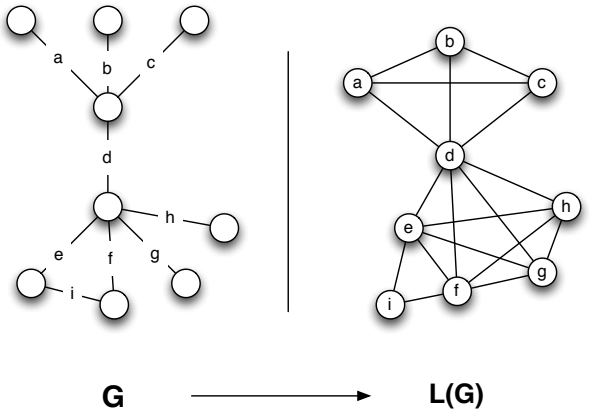


FIGURE 2.4 – *Exemple de transformation d'un graphe en line graph. G représente le graphe initial et L(G) le line graph correspondant*

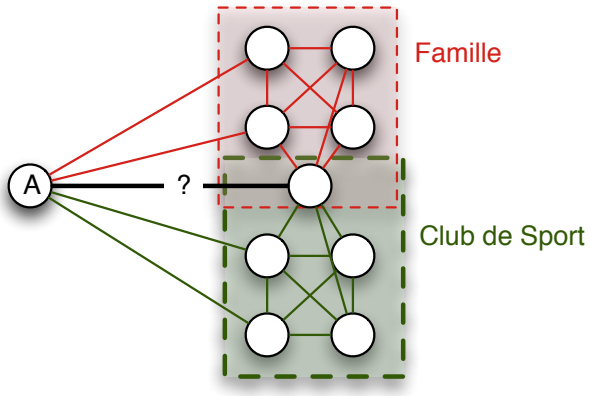


FIGURE 2.5 – *Illustration de l'intérêt des Line Graph. Les liens de la même couleur appartiennent au même cluster de liens. On voit ici que pour le nœud A, cela permet de représenter le recouvrement : une partie de ses liens sont dans un groupe et l'autre partie dans l'autre groupe. Cependant, on constate aussi un problème : le lien marqué d'un point d'interrogation devrait être à la fois vert et rouge. On voit donc que dans le cas des clusters de liens, le recouvrement de communautés est aussi nécessaire.*

à une communauté C_1 (club de sport) et une communauté C_2 famille, aurait bien des liens de type différent avec les individus de ces différents groupes. Si l'individu appartient bien aux deux groupes, les liens avec les nœuds de C_2 correspondent clairement à des liens de relation familiale alors que les autres sont d'une nature différente, comme illustré sur la figure 2.5.

Cependant, cet exemple simple permet de se rendre compte du problème principal : faire des groupes non recouvrants de liens ne permet que de repousser le problème. En effet, si l'un des nœuds de C_1 appartient aussi à C_2, alors le lien entre A et lui est en recouvrement entre C_1 et C_2.

Parmi les autres problèmes de ces méthodes, on peut répertorier la complexité élevée (le Line Graph d'un graphe dense peut vite devenir énorme), et l'incertitude quant à la nature des communautés ainsi découvertes, qui n'est pas nécessairement identique aux communautés de nœuds plus classiques.

2.4.4 Méthodes basées sur des extensions/rétractions de graines

Dans le cas de communautés recouvrantes, les communautés ne sont plus limitées les unes par les autres. Il devient alors possible de constituer chaque communauté indépendamment. Plusieurs solutions ont été proposées pour ce faire, basées sur un même principe en deux étapes :

1. Trouver des graines, c'est à dire des communautés initiales simples mais imparfaites.
2. Modifier ces graines, en les étendant et/ou éventuellement en les contractant, jusqu'à obtenir les communautés recouvrantes recherchées.

Pour certains, les graines peuvent être des communautés trouvées par un algorithme sans recouvrement, comme dans Wang et al. [WJW09]. Les auteurs proposent ensuite d'ajouter ou de retirer des nœuds de ces communautés pour en augmenter la valeur de $force$ (*community strength*), définie comme le ratio entre les liens internes et la somme des degrés des nœuds de la communauté :

$$strength(C) = \frac{C^{int}}{(C^{int} + C^{ext})^{\alpha}}$$

où α est un paramètre, appelé paramètre de résolution, qui permet de faire varier la taille des communautés trouvées. Un paramètre α d'une valeur inférieure à 0,5 conduira à de très larges communautés, tandis que des valeurs de α supérieures à 2 conduisent à des communautés de très petites tailles. [LFK09]

OSLOM (présenté en détail dans la suite, cf. 2.4.8), propose plusieurs méthodes pour obtenir des graines, dont l'une consiste aussi à prendre celles fournies par un algorithme sans recouvrement.

Baumes et al. [BGMI05] et Kelley et al. [Kel10] proposent une méthode ad hoc pour trouver des communautés non recouvrantes. Ils trient d'abord les nœuds selon une métrique (par exemple, PageRank [PBMW99]), puis retirent les nœuds de valeurs les plus élevées jusqu'à obtention de communautés bien distinctes. Ils utilisent ensuite, comme Wang et al. [WJW09] une optimisation de la force de communauté pour chaque communauté. L'ajout principal de la version de Kelley et al. est d'empêcher la formation de communautés non connexes.

L'inconvénient de ces deux premiers types de graines est qu'elles ne permettent pas forcément de trouver toutes les communautés. En effet, comme illustré sur la figure 2.6, certaines communautés peuvent être entièrement constituées de nœuds en recouvrement. De telles communautés ne pourront pas être découvertes en partant de graines sans recouvrement.

Lancichinetti et al. [LFK09] proposent de tirer successivement des nœuds choisis aléatoirement comme graines. Le processus est le suivant :

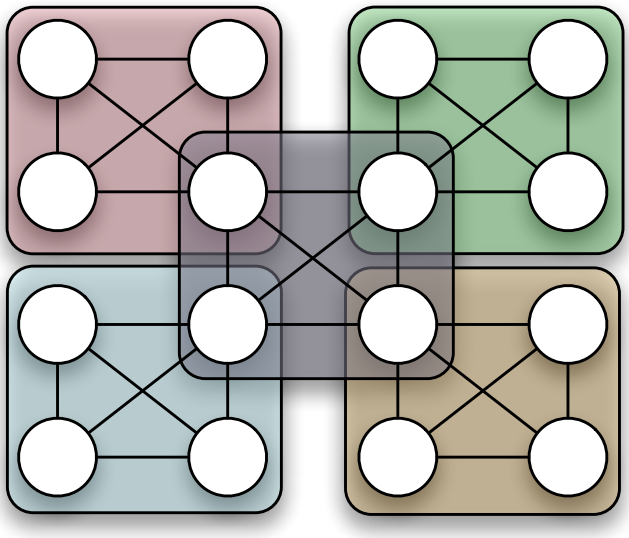


FIGURE 2.6 – *Exemple de communauté dont tous les nœuds appartiennent à d'autres communautés (au centre). Une telle communauté n'est pas du tout détectée par un algorithme sans recouvrement.*

- 1) choix d'un premier nœud
- 2) optimisation de la force de communauté jusqu'à obtention d'une communauté
- 3) choix d'un nœud parmi ceux n'appartenant à aucune communauté
- 4) optimisation de la force de communauté jusqu'à obtention d'une nouvelle communauté
- 5) tant qu'il reste des nœuds sans communauté, retour à 3)

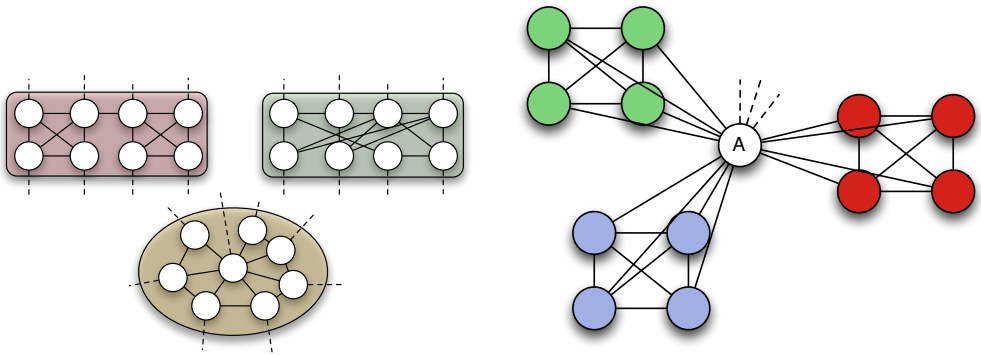
Le problème est le même que précédemment, certaines communautés risquant de ne pas être détectées en cas de recouvrement très important. Certaines communautés peuvent aussi, au final, être très semblables.

Toujours sur le même principe, Lee et al. [LRMH10], proposent d'utiliser les cliques maximales comme graines. Ici aussi, les communautés sont étendues en optimisant la force de communauté. Etant donné le nombre très important de graines initiales, un mécanisme est proposé pour fusionner les communautés devenues semblables (pourcentage de nœuds en commun supérieur à un paramètre).

MONC [HHSG11] s'inspire de ces méthodes, et les améliore, notamment en permettant de fusionner des communautés semblables au cours du temps. L'ajout le plus important est cependant la possibilité de ne pas définir une seule valeur du paramètre de résolution α, mais de laisser chaque communauté le fixer.

Le problème principal de ces méthodes est l'utilisation de la force de communauté comme définition de communauté. Cette métrique est effectivement intuitive : une communauté étant généralement définie comme un ensemble de nœuds fortement connectés entre eux et faiblement connectés au reste du réseau, comparer le nombre de liens internes au nombre de liens externes est judicieux. Cependant, seule, cette métrique peut conduire à des résultats peu pertinents, comme on peut le voir sur la figure 2.7. Ces problèmes sont :

- La communauté est considérée comme une boîte noire, dont seuls les nombres de liens internes et de liens externes sont pris en compte. Or, la topologie interne est parfois importante.



(a) *Réseaux de même force de communauté mais de topologies très différentes*

(b) *Le nœud A est un hub, il appartient à plusieurs communautés. Cependant, du point de vue de chacune des communautés, ajouter ce nœud ferait baisser la force de communauté (en considérant que A peut appartenir à n'importe quel nombre de communautés).*

FIGURE 2.7 – *Illustrations de limites de l'utilisation de la force de communauté comme définition des communautés.*

- Les hubs liés à de nombreuses communautés font diminuer la force de communauté de chacune, et ont donc tendance à ne pas être intégrés dans les communautés. Ceci est contre-intuitif, les hubs étant logiquement les nœuds devant appartenir au maximum de communautés.
- Plus généralement, les communautés trouvées vont avoir tendance à contenir des nœuds dont le rapport lien internes/liens externes sera élevé. Or, dans les cas de réseaux avec fort recouvrements, tels que des réseaux où chaque nœud appartient à plusieurs communautés, ce rapport peut être très faible. Si le réseau est hétérogène en terme de nombre d'appartenance des nœuds, ceci peut poser des problèmes, l'optimisation de la force de communauté ayant tendance à rejeter les nœuds ayant beaucoup de liens externes. Ceci peut d'ailleurs être la source d'un biais lors de tests sur des graphes générés. Dans le benchmark LFR [LFR08] par exemple, benchmark le plus utilisé pour la création de graphes avec communautés, tous les nœuds ont un ratio de liens internes à leurs communautés identique, ce qui est favorable à des processus utilisant la force de communauté, sans être réaliste.

D'autres solutions n'utilisent pas la force de communauté. Padrol-Sureda et al. [PSPLPMM10] proposent de partir de nœuds tirés aléatoirement comme graines, mais leur méthode d'expansion est plus originale : à chaque nœud est associé un vecteur. Chaque sous-ensemble de nœuds S est alors défini comme la somme des vecteurs des nœuds de ce sous-ensemble. La notion de fitness à optimiser est définie comme le Laplacien sur la fonction O, où O est le carré de la longueur Euclidienne du vecteur de S.

Chen et al. [CLSW10] définissent comme graines l'ensemble des nœuds, chacun dans sa propre communauté. Ensuite, l'algorithme consiste à prendre des nœuds au hasard et à leur faire choisir entre trois actions possibles : rejoindre une nouvelle communauté, quitter une communauté, ou changer une communauté pour une autre. La méthode étant basée sur la théorie des jeux, les décisions sont prises par les nœuds à l'aide de deux fonctions : une fonction de gain, basée sur une vision locale de la modularité (plus le nœud va augmenter la modularité de la communauté, plus la fonction de gain est élevée), et une fonction de perte définie comme $(|c_i| - 1)c$, où

pour rappel, c_i correspond à l'ensemble des communautés auxquelles appartient le nœud i, et c est un paramètre de l'algorithme.. Les auteurs prouvent alors que ces fonctions étant linéaires localement, il existe un équilibre de Nash pour le jeu de formation des communautés.

Enfin, Lancichinetti et al. [LRRF11] proposent une méthode plus complexe, OSLOM, que nous détaillerons dans la section 2.4.8.

2.4.5 Méthodes basées sur des modèles génératifs

Une solution pour détecter des communautés consiste à se baser sur un modèle génératif. Latouche et al. [LBA11] proposent un modèle, appelé OSBM (Overlapping Stochastic Block Models), basé sur le classique SBM [NS01]. MOSES, proposé par McDaid et al. [MH10], combine OSBM avec une optimisation locale. A partir de ce modèle, on peut définir une fonction de fitness à optimiser. Les auteurs définissent alors une suite d'actions à effectuer :

1. Sélectionner au hasard des liens dans le réseau. Des communautés initiales sont formées, composées des nœuds extrémité de ces liens. Ensuite, des nœuds sont ajoutés à ces communautés en maximisant la fonction de fitness de manière gloutonne.
2. A chaque fois qu'une tranche de 10 % des liens à été traitée, des communautés peuvent être supprimées, si cela conduit à une augmentation de la fonction objectif. Le test pour la suppression des communautés est fait 10 fois, à chaque fois que 10% des nœuds ont été traités.
3. Enfin, la dernière tâche, inspirée de l'algorithme de Louvain, consiste à retirer successivement les nœuds des communautés auxquels ils appartiennent, puis à voir si les intégrer dans une autre communauté augmenterait la fonction de fitness.

2.4.6 Méthodes basées sur la propagation de labels

La propagation de labels est une méthode qui avait déjà été proposée pour trouver des communautés sans recouvrement [RAK07]. Elle fonctionne de la manière suivante :

1. Chaque nœud est initialisé avec un label unique.
2. De manière successive, chaque nœud x remplace son label par celui utilisé par le maximum de ses voisins (ou un choisi au hasard en cas d'égalités). Après un certain nombre d'itérations, le même label tend à être associé aux membres d'une même communauté.
3. Tous les nœuds ayant le même label forment une communauté.

Gregory et al. [Gre10] proposent une adaptation de cette méthode aux cas avec recouvrement (COPRA). Pour ce faire, ils proposent, lors de la phase 2, de ne plus choisir seulement le label le plus courant chez ses voisins, mais de maintenir une liste des labels les plus courants dans son entourage. Un paramètre de l'algorithme fixe le nombre maximum de labels qu'un nœud peut retenir (sans quoi chaque label s'étendrait à l'infini).

SLPA, proposé par Xie et al. [XSL11], est une autre variante basée sur la propagation de labels. L'ajout principal est la mise en place d'une mémoire, pour chaque nœud, des labels qui lui ont été attribués. Ainsi, la force des appartenances aux diverses communautés est interprétée comme la probabilité de rencontrer le label dans la mémoire du nœud, et non plus seulement l'état des labels lors de la dernière itération comme dans les méthodes précédentes.

La limite de ces méthodes est la même que pour la modularité : la propagation de labels a été conçue dans le cas de communautés sans recouvrement, et le rôle implicite joué par les autres communautés y est très important. Dans la propagation de labels sans recouvrements, les communautés ne s'étendent pas à l'infini car les autres communautés les en empêchent. Or,

la possibilité d'avoir des recouvrements annule ce principe, et on peut alors se demander si les communautés trouvées seront bien pertinentes. Par exemple, selon les auteurs, des communautés non connexes tendent à apparaître, et nécessitent l'ajout d'un mécanisme de post-processing pour repartager les communautés trouvées en communautés connexes.

2.4.7 Clique Percolation Method

Dans cette partie, nous allons présenter en détail l'algorithme CPM [PDFV05], qui est l'un des algorithmes les plus classiques de détection de communautés, en précisant ses avantages et ses faiblesses.

CPM (Clique Percolation Method), est une solution qui a l'avantage d'être conceptuellement simple. Elle prend un paramètre, k, qui représente la taille des cliques qui serviront de « briques de base » à l'algorithme. Par défaut, l'algorithme calcule les solutions pour toutes les valeurs de k, et laisse l'utilisateur choisir celle qui lui semble la plus pertinente, au vu des résultats obtenus. Pour la plupart des réseaux de terrain, une valeur de 4 est la plus efficace. Cela signifiera que tout nœud n'appartenant pas à au moins une clique de taille au moins 4 ne sera classifié dans aucune communauté. Si le graphe est peu dense, il est parfois utile de prendre une valeur de 3. Cette méthode ne fonctionne donc pas sur des graphes ayant une structure arborescente, sans fermetures transitives.

L'algorithme fonctionne en deux étapes : premièrement, trouver dans le réseau toutes les cliques de taille exactement k. Deuxièmement, pour chaque clique trouvée, si elle a $k-1$ nœuds en commun avec une autre clique de taille k, mettre les 2 cliques dans la même communauté. Une communauté est donc formellement définie comme un ensemble de cliques dans lequel on peut passer de l'une choisie au hasard à n'importe quelle autre par une suite de déplacements, où un déplacement consiste à choisir une autre clique ayant $k-1$ nœuds communs avec cette clique.

Quand bien même le principe de cette méthode paraît simple, elle fonctionne remarquablement bien dans les réseaux de terrain. Ceci est lié au fort taux de clustering au sein des communautés, et en particulier le grand nombre de fermetures transitives qu'elles contiennent, dans la plupart des réseaux de terrain.

Le principal avantage par rapport à d'autres méthodes ultérieures, en particulier les méthodes de link clustering, est que cette méthode est capable de détecter des recouvrements très importants, parfois avec de nombreux nœuds, voire liens, en commun, ce qui est très important dans les réseaux de terrain [YL12b].

En revanche, cette méthode souffre de certaines faiblesses : dans des graphes très denses, elle peut avoir une complexité et une consommation mémoire prohibitives. De plus, elle est sensible à certaines configurations. Par exemple, si l'on imagine une suite de cliques de taille k, ayant $k-1$ nœuds en commun, et formant une chaine, CPM les détectera comme une communauté, alors que cela n'est généralement pas pertinent, comme illustré dans la figure 2.8.

Cette méthode sera utilisée dans le chapitre Validation de ce travail (4), avec d'autres, pour comparer les communautés qu'elle trouve avec celles de notre algorithme.

2.4.8 OSLOM

OSLOM [LRRF11] est assez différent des autres algorithmes que nous avons présenté précédemment parce qu'il n'est pas basé sur une idée unique, mais semble plutôt avoir été conçu en rassemblant les éléments les plus efficaces de plusieurs solutions existantes.

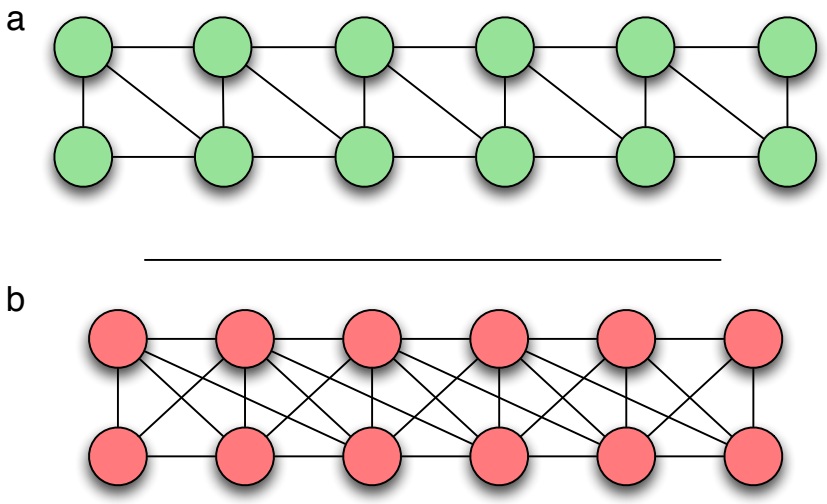


FIGURE 2.8 – *Exemples de communautés détectées par CPM, formées de chaines de cliques voisines. a) cas $k = 3$. b) cas $k = 4$.*

Comme certains de ses prédécesseurs, OSLOM utilise l'idée de communautés « graines », qui pourront ensuite s'étendre en intégrant des nœuds périphériques, ou au contraire se réduire en rejetant les nœuds les moins pertinents. Les auteurs proposent plusieurs méthodes pour obtenir des graines : pour une plus grande rapidité d'exécution, ils proposent d'utiliser InfoMap ou Louvain — des algorithmes rapides et efficaces mais ne tenant pas compte du recouvrement — pour obtenir un « premier jet » de communautés, qu'OSLOM va ensuite se charger d'améliorer. Comme expliqué précédemment, cette méthode à un inconvénient important, c'est qu'elle ne permet pas de détecter toutes les communautés dans des cas de recouvrement très important.

Dans ces situations, les auteurs préconisent d'utiliser comme graines de petites communautés créées en prenant un nœud aléatoirement dans le réseau, et en lui adjoignant un nombre arbitraire de voisins.

Dans un deuxième temps, pour chaque graine, OSLOM va appliquer des règles permettant de successivement ajouter puis retirer des nœuds, jusqu'à arriver à un état stable dans lequel il n'est plus intéressant de modifier la communauté.

Le principe est d'utiliser la probabilité statistique pour un nœud d'être connecté à une communauté : connaissant le degré d'un nœud et le nombre de nœuds dans une communauté, on peut estimer combien ce nœud devrait avoir de connexions avec les nœuds de cette communauté dans un modèle nul aléatoire. Si le nœud a plus de connexions que cette valeur moyenne, on aura tendance à l'ajouter à la communauté. S'il en a moins, on préfèrera le rejeter. Ce principe est similaire à celui utilisé pour la modularité.

Cependant, pour ne pas tomber dans des minimums locaux, le processus d'expansion des communautés est stochastique. Les auteurs proposent donc de répéter le processus d'optimisation N fois pour chaque graine (N étant un paramètre de l'algorithme), et proposent d'utiliser des heuristiques pour garder un résultat unique à partir de ces différentes expansions.

Le processus va ensuite être répété en prenant aléatoirement des graines dans le réseau, et ce tant que ce processus permet de trouver de nouvelles communautés. Pour éviter de trouver des communautés trop proches, chaque nouvelle communauté « optimisée » trouvée va être comparée avec les communautés existantes, et, si elle est considérée similaire, la plus petite des deux va être éliminée (deux communautés sont similaires si leur taux de nœuds en commun est supérieur

à un paramètre de l'algorithme)

Une fois ce processus terminé, on obtient un ensemble de communautés décrivant la totalité du réseau. Cependant les auteurs remarquent que le processus étant stochastique, le résultat peut fortement dépendre de l'ordre dans lequel les graines auront été tirées. L'algorithme demande donc un autre paramètre, qui va correspondre au nombre de fois où le processus complet va être reproduit. Une sélection est ensuite effectuée pour trouver la meilleure de ces solutions, et c'est ce résultat qui sera considéré comme la solution.

Comme on peut le constater, cette méthode n'est pas basée sur une idée nouvelle, simple mais efficace comme CPM, Louvain ou InfoMap. Elle a explicitement été élaborée en prenant en compte les algorithmes précédents, et en procédant par essais erreurs. Cependant, force est de constater que, là où la plupart des autres méthodes avec recouvrement obtenaient des résultats mitigés (notamment sur des graphes générés), cette méthode semble fournir des solutions plus pertinentes. On peut lui reprocher d'être lente à l'exécution, mais beaucoup de tâches étant mutuellement indépendantes, elles pourraient être effectuées en parallèle, ce qui annulerait pratiquement le problème.

Reste la question du paramétrage : avec pas moins de 4 paramètres modifiables, et au moins une valeur seuil arbitraire, on peut s'interroger sur la robustesse des résultats. Cependant, dans beaucoup de cas, en utilisant les paramètres par défaut, les résultats sont pertinents.

2.5 Détection de communautés dynamiques

Toutes les méthodes présentées jusqu'à maintenant s'appliquent uniquement à des réseaux statiques. Par réseau statique, nous entendons un réseau constitué d'un ensemble de nœuds et d'un ensemble de liens entre ces nœuds, sans notion de temps, ou d'ordre, pour les caractériser.

Mais, récemment, un intérêt croissant a été porté à ce que l'on appelle les réseaux temporels, réseaux diachroniques ou graphes variants au cours du temps (Time-varying graphs). Si les phénomènes les plus étudiés sont des phénomènes de dynamique sur les réseaux —infection, diffusion d'information, etc. — on trouve aussi un nombre croissant de travaux sur la dynamique de la topologie des réseaux elle-même, comme par exemple l'accessibilité temporelle dans[WDdACG12]. Une formalisation de ces réseaux ainsi qu'une présentation d'un certain nombre de travaux sur le sujet peuvent être trouvés dans l'excellent article de Holme et Saramäki [HS12].

Le sujet de la détection de communautés dynamiques est cependant si récent qu'il n'existe pas, à notre connaissance, d'article de *review* sur le sujet. L'ajout de la dynamique à la notion de communauté pose de nouvelles questions. L'une des plus importantes est celle des opérations de communautés.

Opérations de communautés Lorsque les communautés sont amenées à évoluer, elles peuvent le faire de plusieurs manières. Les opérations auxquelles on pense tout de suite sont la croissance et la contraction, correspondant à l'ajout et au retrait de nœuds d'une communauté existante. Viennent ensuite la naissance et la mort de communautés : au fur et à mesure que le réseau évolue, de nouvelles communautés peuvent apparaître, et d'anciennes communautés peuvent disparaître. Enfin, on peut identifier deux opérations un peu plus complexes : la fusion et la division. Deux communautés peuvent en effet, au cours du temps, devenir semblable. Elles sont alors fusionnées en une seule. De manière complémentaire, une communauté peut se diviser en deux nouvelles communautés, plus petites que la communauté dont elles sont issues. La façon d'implémenter ces différentes opérations n'a pas encore été fixée, et dépend des algorithmes. Par exemple, lors de la division de la communauté A en deux, doit-on considérer que l'une des

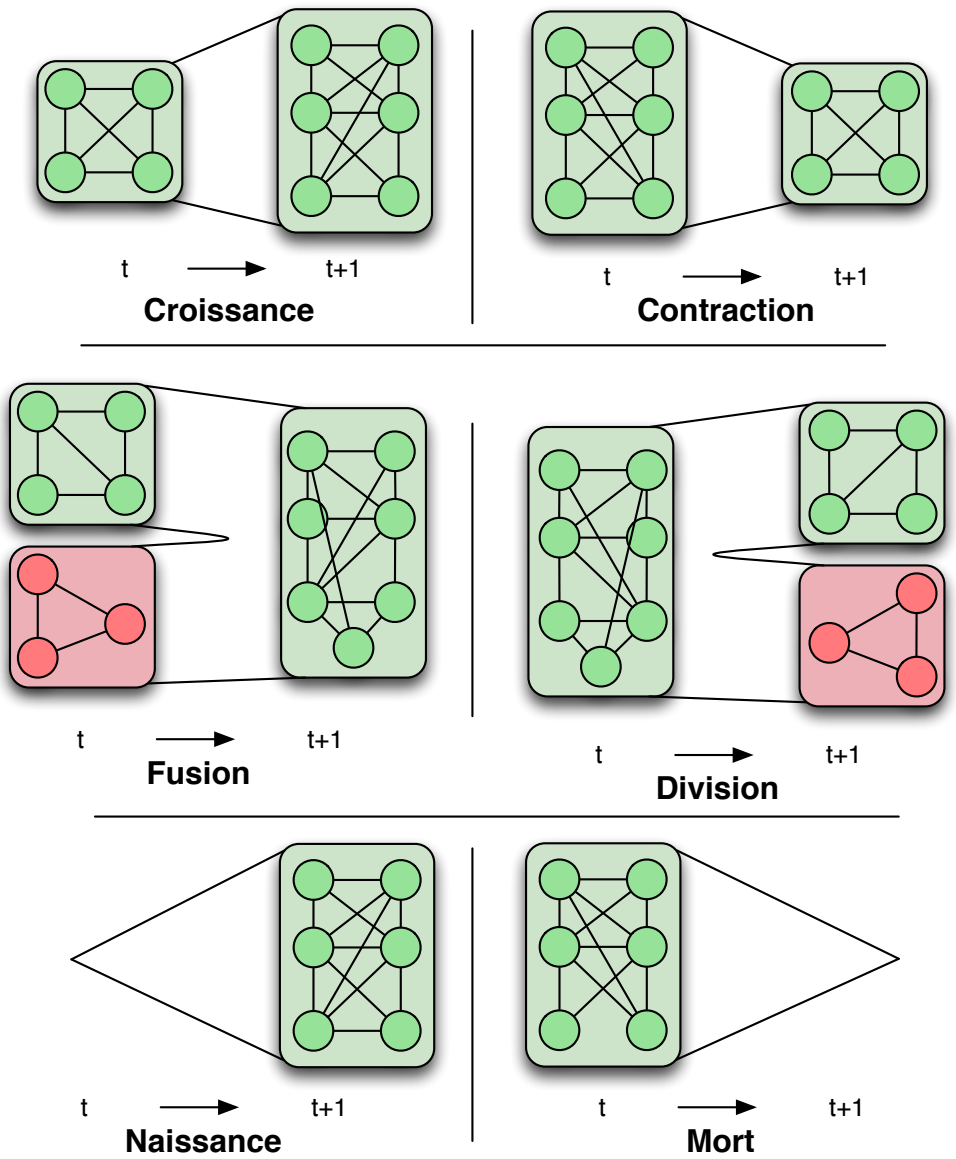


FIGURE 2.9 – *Représentation schématique des opérations possibles sur les communautés dynamiques.*

Date	Article	A	R	Op	F - D
2004	Hopcroft et al. [HKKS04]	1		✓	✓
2006	Chakrabati et al. [CKT06]	3		✓	
2007	Ben Jdidia et al. [BJRF07]	2		✓	
	Palla et al. [PBV07]	1	✓	✓	✓
	Tantipathananandh et al. [TBWK07]	2		✓	
2008	Falkowski et al. [FBS08]	4	✓	✓	✓
	Wang et al. [WWD08]	1		✓	✓
2009	Chan et al [CHX09]	3		✓	
	Lin et al. [LCZ+09]	3	✓		
2010	Aynaud et al. [AG10a]	2			
	Aynaud et al. [AG10b]	3		✓	
	Cazabet et al. [CAH10]	4	✓		
	Chen et al. [CWJ+10]	1	✓		
	Greene et al. [GDC10]	1	✓	✓	✓
	Mucha et al. [MRM+10]	2		✓	
	Wang et al. [WF10]	3		✓	
2011	Cazabet et al. [CA11]	4	✓	✓	✓
	Lancichinetti et al. [LRRF11]	3	✓	✓	
	Mitra et al. [MTR11]	4		✓	
	Xu et al [XKH11]	3		✓	
	Yang et al. [YCZ+11]	2		✓	
2012	Li et al. [LHB+12]	4		✓	
	Shang et al. [SLX+12]	4			✓

TABLE 2.2 – *Algorithmes de détection de communautés dynamiques, classés par année de publication. On observe que la majorité des méthodes proposées l'on été après le début de cette thèse (2009). Légende : **A** : Approche utilisée. 1 : Détections statiques successives. 2 : Étude simultanée de toutes les étapes d'évolution. 3 : Détections statiques informées successives. 4 : Réseaux temporels. **R** : Recouvrement de communautés. **Op** : Identification des opérations de communautés les plus simples : naissance, mort, croissance, contraction. **F - D** : Identification des fusions/divisions de communautés*

communautés résultantes reste la même communauté A, tandis que l'autre est une nouvelle communauté distincte A_2, ou doit-on considérer qu'une fois divisée, la communauté A n'existe plus, et a cédé la place à deux nouvelles communautés A_1 et A_2 ? Il n'existe pas à l'heure actuelle de solution claire à ce problème, et chaque algorithme adopte sa solution propre. Palla et al. ont, dans [PBV07], été les premiers à définir ces opérations. Ils en ont proposé une représentation schématique, dont nous proposons une version équivalente sur la figure 2.9.

Au cours de cette thèse, nous avons été amenés à identifier une autre opération possible, l'opération de résurgence, dont nous parlons au chapitre 5.4.

Différentes approches Plusieurs types d'approches assez différentes ont été proposées pour détecter des communautés dynamiques. Les premières approches proposaient d'appliquer directement les méthodes statiques : il suffisait de décomposer l'évolution du réseau en instantanés, d'appliquer un algorithme classique à chacun de ces instantanés, puis de faire correspondre les communautés trouvées dans un instantané avec celles de l'instantané précédent.

Transition	Notation	Indicateur
Le cluster survit	$x \to Y$	$match(X,Y) > \tau \wedge (\nexists Z : Z \neq X \wedge match(X,Y) > \tau)$
Le cluster est divisé	$X \to \{Y_1, \ldots, Y_2\}$	$match(X, \bigcup_u = 1^p > \tau \wedge \forall u = 1 \ldots p : match(X,Y_u) > \tau_{split} \wedge (\nexists Y : Y \notin Y_1, \ldots, Y_p \wedge match(X,Y) > \tau_{split})$
Le cluster est absorbé	$X \to Y$	$match(X,Y) > \tau \wedge (\exists Z : Z \neq X \wedge match(Z,Y))$
Le cluster disparait	$X \to \odot$	X ne satisfait à aucun des cas précédents
Un nouveau cluster apparaît	$\odot \to X$	

TABLE 2.3 – *Operations sur les communautés, selon [SNTS06]. τ et τ_{split} sont des paramètres.*

Cependant, l'instabilité des méthodes statiques rend cette technique peu efficace. Outre l'approche par détections statiques successives, on peut donc identifier d'autres approches : les approches cherchant des communautés en étudiant la globalité du réseau, les approches incrémentales par instantanés et les méthodes étudiant directement le réseau temporel.

Nous allons donc présenter succinctement les méthodes proposées, classées selon ce critère. Pour chaque méthode, nous ferons la différence entre les méthodes capables d'identifier des recouvrements de communautés et celles qui ne le peuvent pas, car cela joue aussi un rôle dans la qualité de la dynamique des communautés, en particulier pour identifier les fusions et les divisions. Nous proposons également une classification chronologique de ces méthodes dans la table 2.2.

2.5.1 Approches par détections statiques successives

Ce sont les méthodes les plus simples, puisqu'elles ne font intervenir que des détections de communautés statiques suivies d'un post-traitement. Cependant, toutes ces méthodes souffrent du problème de l'instabilité de la détection, comme expliqué au chapitre suivant. Le principe général de cette approche est illustré dans la figure 2.10.

2.5.1.1 Avec communautés non recouvrantes

Hopcroft et al. [HKKS04] proposent probablement la première méthode s'intéressant aux changements de communautés. Les auteurs considèrent seulement deux instantanés du réseau, et observent les changements entre ces deux instantanés en calculant une valeur d'appariement (*matching*) entre les communautés. Cet appariement est défini comme :

$$match(C_1, C_2) = min\left(\frac{|C_1 \bigcap C_2|}{|C_1|}, \frac{|C_1 \bigcap C_2|}{|C_2|}\right)$$

Les auteurs notent qu'il existe un gros problème d'instabilité de la détection, et, pour le limiter, décident de ne considérer les changements que des communautés stables, c'est à dire des communautés qui, même si l'on introduit un changement mineur du réseau, restent identiques. Les auteurs n'ont pas prévu d'opérations de communautés autres que la contraction et l'expansion. On peut cependant facilement relier ces travaux à ceux portant sur le clustering en général, et utiliser des méthodes telles que celles définies par [SNTS06] pour les clusters dynamiques.

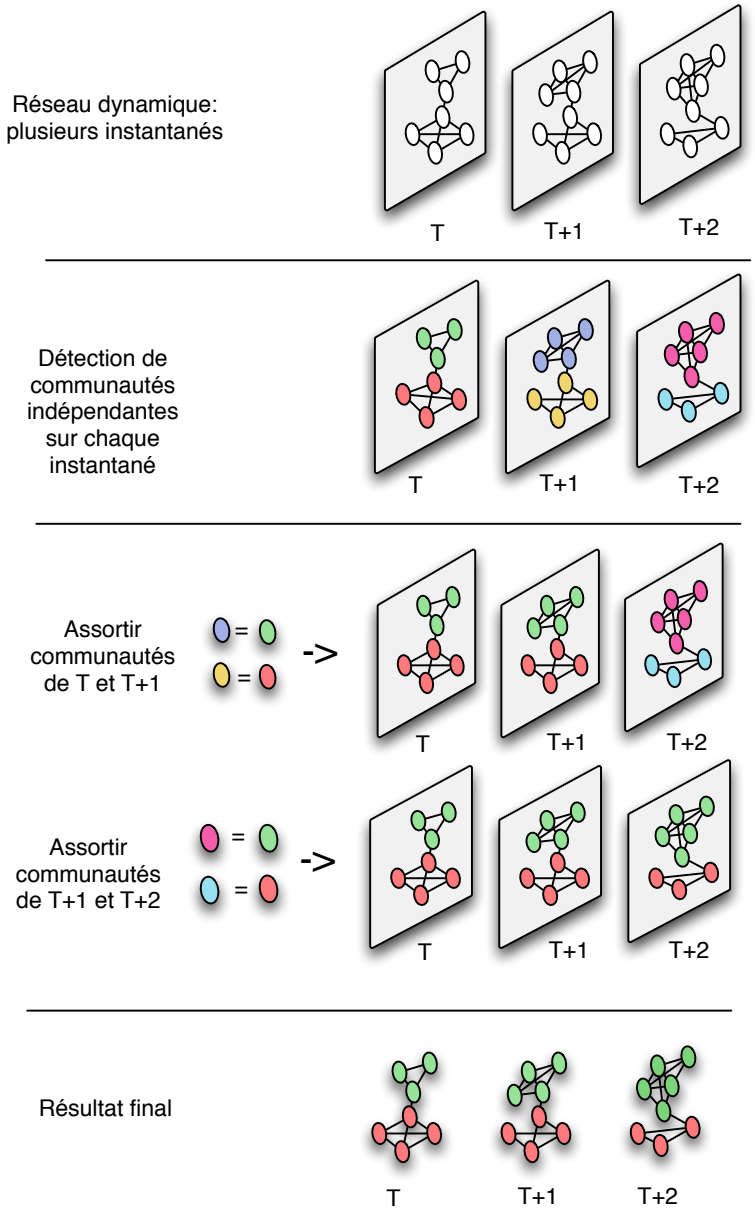


FIGURE 2.10 – *Illustration de l'approche par détections statiques successives.*

Dans ces travaux, les transitions sont définies tel que présenté dans la table 2.3. Ces propositions sont assez intuitives, et font intervenir deux paramètres permettant de spécifier à quel point on demande à des communautés d'être proches pour les considérer identiques. Le problème est qu'en étudiant des instantanés successifs, la valeur de ces paramètres va être très importante. Par exemple, si l'on prenait une valeur de τ de 0,9 la moindre variation conduirait à ne plus considérer les communautés comme pérennes, mais les ferait se diviser sans cesse. Avec une valeur faible, on aurait un comportement inverse.

Wang et al. [WWD08], notant que les communautés sont trop différentes d'un instantané à l'autre, ont l'idée d'utiliser des nœuds cœurs pour les identifier. Via une métrique propre, ils identifient les nœuds considérés comme caractéristiques de chaque communauté sur deux instantanés consécutifs. Pour observer comment les communautés ont évolué, il suffit alors de regarder comment les nœuds cœurs se sont comportés. Si deux d'entre eux initialement dans des communautés différentes sont maintenant rassemblés dans la même, c'est qu'il y a eu un processus de fusion. Si au contraire ils ont été séparés, c'est qu'il y a eu division. Cette méthode a cependant des faiblesses. Résumer l'identité des communautés à un faible pourcentage de leurs nœuds parait limitant, et pose en particulier problème pour des communautés assez homogènes, où il n'y a pas vraiment de nœuds centraux.

2.5.1.2 Avec communautés recouvrantes

Palla et al. [PBV07] proposent une adaptation de leur algorithme CPM pour les communautés dynamiques. Les communautés sont détectés sur chaque instantané à l'aide de CPM. Pour chaque paire d'instantanés successifs I_1 et I_2, ils créent alors un graphe union, U, contenant les nœuds et les liens de chacun d'eux. CPM est alors appliqué sur U. Une propriété intéressante est que, CPM étant basé sur des cliques, et l'union des deux instantanés contenant au moins toutes les cliques des deux instantanés, chaque communauté de I_1 et de I_2 sera contenue dans une et une seule communauté de U. Cette information est donc utilisée pour faire correspondre les communautés de I_2 à celles de I_1 : si à une communauté de U correspond exactement une communauté dans I_1 et une communauté dans I_2, ces deux communautés sont considérées comme la même. Si à une communauté de U correspondent deux communautés de I_1 et une de I_2, il s'agit d'une fusion, et inversement pour une division. Le problème se pose lorsqu'à une communauté de U correspondent plusieurs communautés de I_1 et plusieurs communautés de I_2. Les communautés sont alors identifiées en fonction de leurs taux de recouvrement.

Chen et al [CWJ+10] définissent les communautés comme étant les cliques maximales du réseau. Cette définition stricte rend la détection d'opérations sur les communautés très simples. Les auteurs proposent également d'identifier des nœuds cœurs au sein de ces communautés, définis comme les nœuds appartenant au plus petit nombre de communautés. Il est alors facile de suivre les communautés d'un instantané à l'autre. Évidemment, la faiblesse principale de cette méthode est qu'elle définit les communautés comme étant des cliques maximales, ce qui, d'une part, donne souvent des communautés peu pertinentes et, d'autre part, conduit à un nombre bien trop important de communautés dans des grands graphes ayant une densité élevée.

Greene et al. [GDC10] utilisent un algorithme statique pour détecter les communautés sur chaque instantané. L'algorithme utilisé dans l'article est MOSES, parce qu'il est capable de gérer du recouvrement. Ensuite, un système de matching est utilisé pour reconnaître les communautés des différents instantanés. Les opérations de communautés sont possibles.

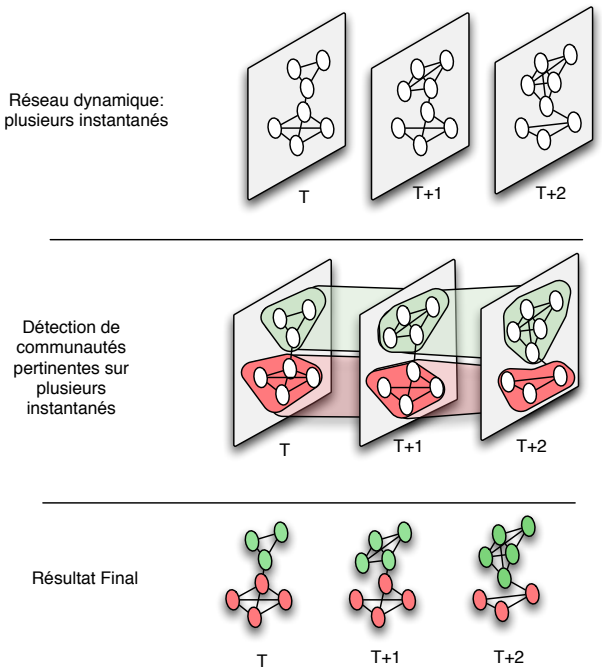


FIGURE 2.11 – *Illustration de l'approche par étude simultanée de toutes les étapes d'évolution.*

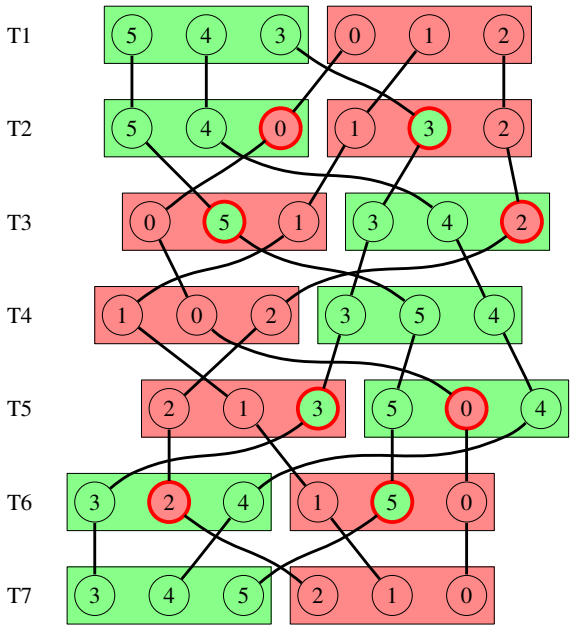


FIGURE 2.12 – *Exemple de détection de communautés dynamiques avec la méthode de Tantipathananandh et al. On peut observer (cercles entourés de rouge) des nœuds qui sont dans une communauté dans un instantané (rectangle de couleur), mais dans une communauté dynamique différente (couleur dans le cercle).*

2.5.2 Approches par étude simultanée de toutes les étapes d'évolution

Analyser chaque instantané séparément pose problème, à cause de l'instabilité des algorithmes de détection de communauté. Une autre approche consiste donc à analyser l'ensemble des étapes d'évolution simultanément. Dans les approches de cette section, le principe d'instantané est conservé, mais tous sont étudiés simultanément, afin de découvrir des communautés cohérentes sur plusieurs d'entre eux. Le principe général de cette approche est illustré dans la figure 2.11.

On notera qu'aucun algorithme utilisant cette méthode ne permet pour l'instant de trouver des communautés recouvrantes.

2.5.2.1 Avec communautés non recouvrantes

Tantipathananandh et al. [TBWK07] commencent par considérer que les communautés pour chaque instantané sont non ambiguës (des cliques non recouvrantes), la détermination de communautés sur chaque instantané n'est donc pas abordée. Ce a quoi ils s'intéressent est la détermination de communautés cohérentes dans le temps à partir de ces communautés d'instantanés. Ils proposent d'utiliser 3 coûts, pouvant être paramétrés :

- Le $i-cost$, coût individuel, va pénaliser, pour un nœud, le changement de communauté.
- Le $g-cost$, coût de groupe, va pénaliser, pour un nœud, le fait de ne pas apparaître dans sa communauté à un moment donné.
- Enfin, le $c-cost$, coût de couleur, va pénaliser toute couleur utilisée après la première pour un nœud, c'est à dire que ce coût représente le nombre total de communautés différentes auxquelles le nœud appartient au cours du temps.

Ils utilisent ensuite un algorithme pour optimiser ces fonctions de coût sur l'ensemble du réseau temporel. On voit ici une spécificité de cet algorithme : il y a un découpage en communautés lié à chaque instantané et, ensuite, est calculé un découpage en communautés dynamiques. Par exemple, si un nœud qui appartient à la communauté A dans l'instantané t_1, passe dans la communauté B dans l'instantané t_2, avant de revenir dans la communauté A, on pourra considérer que sa communauté reste la communauté A à l'instant t_1, malgré le découpage initial de t_2. On peut voir un exemple sur la figure 2.12.

Cette méthode, bien qu'intéressante, souffre de plusieurs limitations : d'abord, elle ne permet pas d'envisager des fusions ou des divisions de communautés. Ensuite, elle ne semble pas pouvoir passer à l'échelle pour de grands graphes, malgré les différentes approches proposées pour l'optimisation du coût. Enfin, les résultats dépendent beaucoup des valeurs que l'on donne aux différents coûts.

Yang et al [YCZ+11] proposent une modification du *stochastic bloc model* [HLL83] pour prendre en compte la dynamique. Bien que cette approche semble intéressante, elle ne permet pas les opérations de fusion ou de division, ni recouvrement de communautés, et n'est applicable que sur de petits réseaux.

Ben Jdidia et al. [BJRF07] créent un seul réseau à partir de différents instantanés en liant entre eux les nœuds similaires apparaissant à des instantanés différents, comme montré sur la figure 2.13. Ils créent également des liens entre les nœuds connectés à au moins un voisin commun dans deux instantanés consécutifs. Une version légèrement modifiée de l'algorithme statique WalkTrap [PL05] est alors utilisée sur ce réseau pour détecter des communautés. Cette méthode ne permet pas d'opérations de fusion ou division de communautés.

Mucha et al. [MRM+10] créent un seul réseau, de manière légèrement différente à Ben Jdida et al. : ils connectent entre eux les nœuds identiques entre deux instantanés différents, consécutifs

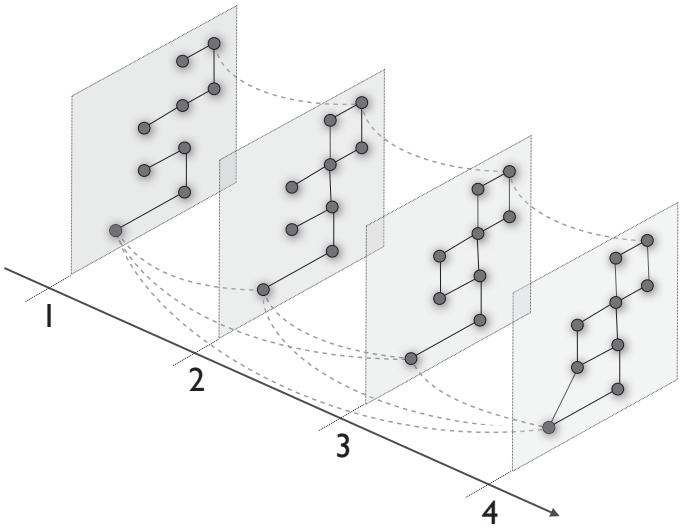


FIGURE 2.13 – *Création d'un seul réseau en connectant les nœuds identiques dans des instantanés successifs.*

ou non. Une version généralisée de l'algorithme Louvain est utilisée pour optimiser la modularité. Cette méthode ne permet pas d'opérations de fusion ou division de communautés.

Mitra et al. [MTR11] travaillent sur des réseaux temporels particuliers, composés de liens dont les extrémités ne sont pas dans le même instantané. Ces réseaux sont typiquement des réseaux de citations d'articles, ou de références de blogs vers d'autres blogs (qui citent des articles publiés précédemment dans d'autres blogs). Ces réseaux ont donc l'avantage de représenter toute l'évolution dans un seul graphe. Il ne s'agit cependant pas de réseaux classiques. Les auteurs montrent ensuite qu'il est directement possible d'appliquer une algorithme statique sur ce type de réseau, comme dans les deux méthodes précédentes. Dans leur article, ils utilisent l'algorithme Louvain, pour sa rapidité. Il serait cependant possible d'utiliser un algorithme avec recouvrement, d'après les auteurs.

Enfin, Aynaud et al. [AG10a] utilisent une version modifiée de l'algorithme Louvain pour maximiser la modularité, non pas sur un instantané, mais sur l'ensemble des instantanés, ou un sous-ensemble d'instantanés. Concrètement, l'algorithme cherche à optimiser la modularité moyenne d'un groupe de nœuds sur plusieurs instants du réseau. L'algorithme va donc chercher des communautés cohérentes sur le long terme. Cette méthode ne permet pas d'opérations de fusion ou division de communautés.

2.5.3 Approches par détections statiques informées successives

Ces approches utilisent toujours des instantanés, et effectuent une détection pour chacun d'entre eux. Cependant, afin de résoudre le problème de l'instabilité des algorithmes, ces méthodes proposent de prendre en compte les résultats obtenus à l'étape t lors de la détection des communautés à l'étape $t+1$. Ceci réduit l'instabilité, car, au cas où l'algorithme ne saurait lequel choisir entre deux découpages différent, il pourrait par exemple prendre le plus semblable au découpage précédent. Le principe général de cette approche est présenté dans la figure 2.14.

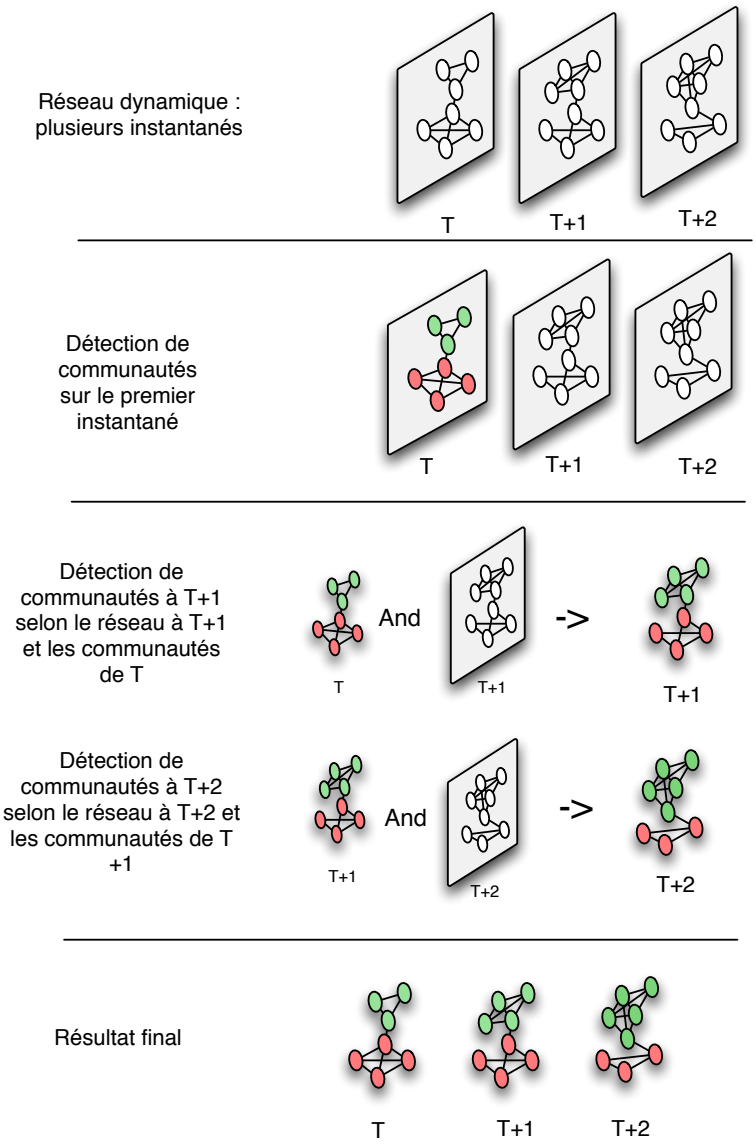


FIGURE 2.14 – *Illustration de l'approche par détections statiques informées successives.*

2.5.3.1 Avec communautés non recouvrantes

Wang et al. [WF10] réutilisent l'idée des nœuds cœurs, proposée précédemment par Wang et al. [WWD08], mais ont recours à une astuce pour réduire l'instabilité entre deux détections. Ils utilisent l'algorithme de Louvain pour les détections sur chaque instantané, et ils initialisent cet algorithme avec les nœuds cœurs trouvés à l'étape précédente, ce qui permet de limiter l'instabilité. Celle-ci reste cependant importante.

Dans cet algorithme, les nœuds cœurs sont définis comme ceux qui ne changent pas de communautés si on exécute plusieurs fois le même algorithme sur le même réseau légèrement modifié.

Aynaud et al. [AG10b] proposent une méthode au mécanisme proche : à chaque étape, les communautés sont détectées selon l'algorithme de Louvain, initialisé avec les communautés trouvées à l'étape précédente. Cette méthode, en revanche, n'utilise pas les nœuds cœurs.

Chakrabarti et al. [CKT06] ont proposé une méthode d'*evolutionary clustering* . Ils ne se sont donc pas préoccupés de l'évolution des communautés sur le long terme, ni des questions de fusion ou de division. En revanche, ils ont cherché à assurer que les *clusters* trouvées à l'instant $t+1$ soient cohérents par rapport à l'instant t. Pour ce faire, ils ont mit au point une fonction de qualité en deux composantes : la première est statique, et concerne donc le réseau de l'instant t étudié, tandis que l'autre sert à assurer la stabilité, et évalue donc la distance entre les *clusters* à l'étape précédente et les *clusters* à l'étape courante. La fonction de qualité peut donc s'exprimer de la manière suivante :

$$Q = Q_{instant} + \alpha Q_{stabilit}$$

Dans laquelle α représente un paramètre permettant de donner plus ou moins de poids à la cohérence avec le résultat précédent. Chan et al [CHX09] utilisent la même idée.

Xu et al [XKH11] utilisent une idée proche de la précédente, mais en modifiant directement la matrice d'adjacence correspondant à l'instantané t en tenant compte de l'instantané $t-1$. La matrice d'adjacence à l'instant t est donc définie par :

$$\bar{W}^t = \alpha^t \bar{(W)}^{t-1} + (1-\alpha^t) W^t$$

2.5.3.2 Avec communautés recouvrantes

Lin et al. [LCZ+09, LCZ+08] proposent une solution, basée sur un modèle génératif probabiliste, consistant à formuler une fonction de qualité comme un problème de factorisation de matrices non négatives qui optimise conjointement la qualité et la stabilité des communautés. Bien que cette méthode ait l'avantage de permettre la détection de communautés recouvrantes, elle impose cependant de fortes contraintes : le nombre de communautés doit être connu à l'avance, et il n'est *a priori* pas possible d'ajouter ou de supprimer des nœuds au cours du temps. Elle ne permet pas non plus d'opération telles que la fusion ou la division de communautés.

OSLOM [LRRF11] a déjà été présenté en détail dans la section sur les méthodes avec recouvrement. Mais les auteurs présentent aussi, sans s'y attarder, un mécanisme pour l'adapter aux réseaux dynamiques. Comme expliqué précédemment, il est possible de fournir à OSLOM, pour détecter les communautés sur un graphe statique, un jeu initial de communautés. OSLOM va alors se charger de modifier ces communautés, en les étendant ou les rétractant, pour les améliorer. Dans un graphe dynamique représenté par des séries d'instantanés, il est donc possible d'alimenter OSLOM avec les communautés de l'instantané précédent lorsque l'on veut étudier un instantané donné. Les auteurs ne donnent cependant pas de détails sur la façon dont des opérations sur les communautés pourraient être traitées. La gestion de l'apparition et de la mort des communautés pourrait également se révéler problématique.

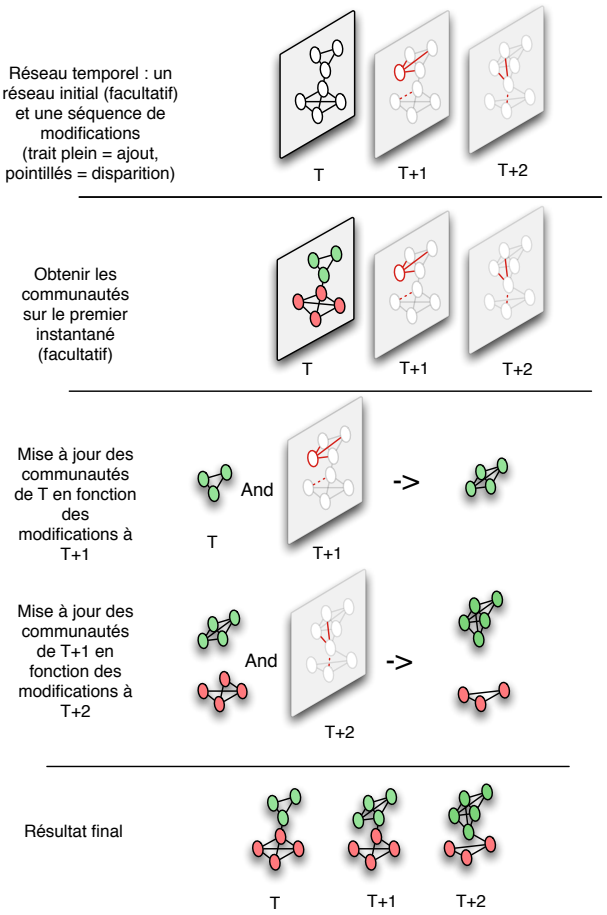


FIGURE 2.15 – *Illustration de l'approche par détection de communautés sur des réseaux temporels.*

2.5.4 Approches travaillant sur des réseaux temporels

La dernière section concerne les algorithmes travaillant directement sur le réseau temporel. Ici, l'évolution du réseau n'est plus considérée comme une succession d'instantanés, mais comme une succession de modifications sur le réseau. L'idée est donc de prendre en compte la ou les dernières modifications effectuées sur le réseau, et de modifier les communautés existantes en conséquence. Il n'y a plus ici de problème d'instabilité, les communautés perdurant naturellement. En revanche, d'autres problèmes se posent, comme l'aspect très local des modifications, qui peut entrainer une dérive vers des communautés qui ne sont plus valables à un moment donné, par rapport à l'état du réseau à cet instant (la détection de communauté n'étant jamais faite sur l'ensemble du réseau, mais uniquement par petites modifications locales successives).

Le principe général de cette approche est présenté sur la figure 2.15.

2.5.4.1 Avec communautés non recouvrantes

Li et al. [LHB+12] utilisent l'évolution du réseau lien par lien. A chaque étape, en fonction des modifications faites au réseau (ajout ou suppression d'un lien), les nœuds affectés par ces modifications vont pouvoir changer de communautés, par rapport à l'étape précédente. Le mécanisme utilisé est très simple, puisqu'il est uniquement basé sur le nombre de liens qu'a chaque

nœud avec les différentes communautés (degré interne aux communautés). Concrètement, le nœud va appartenir à la communauté avec laquelle il a le plus de liens. Cependant, si la différence entre deux communautés n'est pas nette (inférieure à un paramètre de l'algorithme), alors le nœud préfèrera rester dans la communauté dans laquelle il appartenait à l'étape précédente. Cet algorithme ne permet ni recouvrement de communautés ni opération sur les communautés.

Enfin, Shang et al. [SLX+12] utilisent également l'évolution du réseau lien par lien. En commençant avec des communautés données sur un graphe donné, trouvées par une optimisation de la modularité, la méthode consiste à ajouter (ou supprimer) chaque nouveau lien au fur et à mesure de son apparition (ou disparition), et à modifier localement les communautés affectées de manière à, si possible, augmenter la modularité, sinon, limiter autant que possible sa diminution. Les auteurs notent que la méthode ne fonctionne pas indéfiniment, les communautés trouvées finissant par avoir une modularité nettement inférieure au maximum possible. De plus, si les communautés peuvent être fusionnées, il n'est pas prévu de pouvoir en créer de nouvelles au cours de l'évolution.

2.5.4.2 Avec communautés recouvrantes

La méthode de Falkowski et al. [FBS08] est basée sur un algorithme utilisé habituellement pour trouver des clusters dans des données spatiales, DBSCAN. Les auteurs choisissent donc de définir une distance entre les nœuds du réseau, variant entre 0 et 1, et représentant le nombre de fois que ces nœuds ont interagi au cours de l'histoire du réseau. L'algorithme DBSCAN consiste ensuite à identifier des nœuds cœurs, qui sont des nœuds ayant plus de voisins qu'une limite η. Si c'est le cas, une communauté est définie à partir d'eux, constituée de tous les nœuds non cœurs situés à une distance inférieure à ϵ, un autre paramètre. Si un autre nœud cœur est compris dans ce voisinage, il fait lui aussi partie de cette communauté, ainsi que ses voisins, et ainsi de suite. Le processus d'évolution des communautés est alors assez simple : à chaque nouvelle étape, les valeurs de distance entre les nœuds sont mises à jour. Si l'une de ces modifications fait apparaître un nouveau nœud cœur, celui-ci est intégré à une nouvelle communauté, ou en crée une nouvelle s'il n'a pas de nœud cœur dans son voisinage. De même, si des nœuds qui étaient dans le rayon ϵ d'un nœud cœur passent au-delà de ce rayon, ils quittent la communauté, et ainsi de suite. Les opérations et évolutions de communautés sont de cette manière définis très simplement et naturellement. En revanche, le problème de cette solution est que la définition de communauté utilisée est très particulière, et est assez éloignée de ce que l'on considère généralement comme une bonne communauté. De plus, la solution est extrêmement dépendante des valeurs choisies pour les paramètres η et ϵ. Dans l'application donnée par les auteurs, ces valeurs sont choisies de manière à ce que 90% des nœuds soient en fait considérés comme du bruit, c'est à dire ni des nœuds cœurs, ni des nœuds situés à une distance ϵ de l'un d'eux. Les communautés trouvées sont donc généralement très denses, et la majorité du réseau n'est pas clusterisée.

2.6 Faiblesses des algorithmes existants pour la détection de communautés dynamiques

2.6.1 Utilisation de la modularité

Beaucoup d'algorithmes proposés pour la détection de communautés dynamiques font usage de la modularité. Qu'ils cherchent à l'optimiser sur chaque instantané du graphe dynamique, sur un graphe agrégé, ou bien à la maintenir à un niveau élevé au fil des modifications, l'idée reste la même : utiliser la modularité comme définition de ce que sont de bonnes communautés. Cette

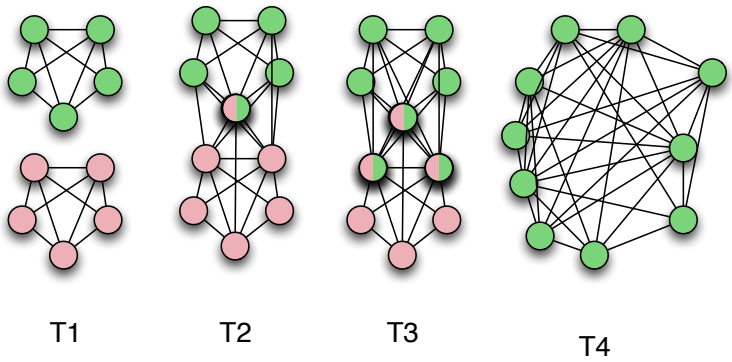


FIGURE 2.16 – *Illustration d'une fusion progressive de communautés.*

métrique occupe certes une place à part dans le domaine : proposée dans le premier algorithme moderne de détection de communauté, par Girvan et Newman, elle se comprend aisément et fait sens intuitivement : une communauté est bonne si elle contient plus de liens qu'elle ne devrait en contenir dans un graphe aléatoire de propriété similaire au graphe étudié. Les résultats donnés par une optimisation de cette métrique étant souvent pertinents, en particulier sur des petits graphes aux communautés clairement définies, elle a parfois été considérée comme la « définition » de ce que sont des communautés.

Or, il ne faut pas oublier que cette métrique n'est qu'une métrique. Il a maintenant été démontré plusieurs fois que les communautés correspondant au maximum de modularité étaient peu pertinentes dans de nombreux graphes.

Fortunato et al. [FB07] ont prouvé la limite de résolution de la modularité. Concrètement, pour un réseau d'une taille (nombre de nœuds) donnée et d'une densité donnée, la taille minimale des communautés qu'une optimisation de la modularité pourra trouver est fixe, et croît avec le réseau. Dit autrement, plus le graphe contient de nœuds et de liens, et plus les communautés correspondant à l'optimum de modularité vont être grandes. Ceci, bien sûr, ne correspond pas à ce que l'on trouve dans les réseaux de terrain. Des tests empiriques sur des graphes générés réalistes ont également montré que les méthodes essayant d'optimiser la modularité étaient moins efficaces que nombre d'autres méthodes. [LF09]

Enfin, plus récemment, une analyse de nombreux réseaux de terrain pour lesquels les communautés existantes étaient connues a montré que la modularité n'était pas un bon indicateur de la qualité des communautés prises individuellement, notamment parce qu'elle était incapable, dans beaucoup de cas, de différencier de véritables communautés d'ensemble de nœuds pris aléatoirement contenant de nombreux liens entre eux, contrairement à d'autres métriques. [YL12a]

2.6.2 Non prise en compte du recouvrement

De nombreuses méthodes, soit parce qu'elles utilisent la modularité, soit pour d'autres raisons, ne permettent pas aux communautés trouvées d'avoir simultanément des nœuds en commun. Or, cela va à l'encontre de ce que l'on sait sur les réseaux de terrain. Le recouvrement est un phénomène important, cela était connu depuis longtemps pour les réseaux sociaux, et a aussi été montré dans des travaux récents. Leskovec a montré [YL12b] que dans de nombreux réseaux tirés du Web 2.0, la majorité des nœuds appartiennent à plusieurs communautés, certains d'entre eux pouvant même avoir des dizaines d'appartenances différentes.

De plus, dans le cas des communautés dynamiques, il existe des phénomènes tels que la fusion de communautés, qui va amener deux communautés initialement distinctes à n'en former plus

qu'une. Mais ce processus est progressif : il va forcément exister un état ou les deux communautés seront partiellement fusionnées, comme présenté sur la figure 2.16. Sans recouvrement, il n'est pas possible de représenter un tel cas de figure, et la détection de ces opérations sera donc hasardeuse.

2.6.3 Instabilité des algorithmes statiques

Plusieurs approches, en particulier les plus anciennes, avaient comme idée de considérer le graphe temporel comme une succession d'instantanés indépendants. Une première étape consistait donc à appliquer un algorithme statique sur chacun de ces instantanés, et ensuite de trouver une correspondance entre les communautés existant dans des instantanés consécutifs. Cette méthode a un avantage certain : elle permet de réutiliser telles quelles les méthodes statiques existantes, qui arrivent maintenant à maturité.

Malheureusement, cette approche a une faiblesse intrinsèque importante : sur deux graphes relativement proches topologiquement, le même algorithme va parfois trouver des communautés différant fortement. La plupart des algorithmes performants faisant intervenir une part de stochastique (Louvain, OSLOM, InfoMap et d'autres), le même algorithme sur le même réseau peu déjà fournir des résultats sensiblement différents.

Cela pose deux problèmes :

- Instabilité : entre deux instantanés consécutifs, les communautés vont donc être différentes. L'algorithme va interpréter ces différences en terme d'évolution des communautés : il va considérer que certains nœuds sont entrés ou sortis de telle communauté, ou que ce qui ne formait auparavant qu'une seule communauté en forme désormais deux. Mais une partie de ces modifications étant seulement due à l'instabilité de l'algorithme, seule une partie des modifications constatées correspondront réellement à une évolution du réseau. Cette instabilité à notamment été mise en évidence dans [AG10b]. Les auteurs montrent que sur un réseau initial constitué de 9377 nœuds, en retirer un seul peut conduire à plus de 2500 changements unitaires (un nœud change de communauté) avec l'algorithme de Louvain, ainsi qu'à plus de 3000 et plus de 500 avec d'autres algorithmes testés. Cette instabilité est donc extrêmement forte.
- Oscillation : prenons le cas de deux communautés en cours de fusion. Une partie importante de leurs nœuds sont partagés, mais pas tous. Il peut donc être tout à fait pertinent de détecter déjà une seule communauté, ou les deux. Considérons qu'à l'étape $t1$, l'algorithme trouve les deux communautés distinctes. A l'étape $t2$, sans qu'il n'y ait eu de modification affectant cette partie du réseau, l'algorithme ne voit plus qu'une seule communauté. A l'étape suivante, il peut tout à fait trouver à nouveau deux communautés distinctes, et ainsi de suite. Dès que les communautés ne seront pas clairement définies, on pourra ainsi se retrouver dans un cas où il ne sera plus possible de se fier à des détections successives.

On pourrait cependant envisager de pallier au problème en utilisant le principe de communautés floues. Par exemple, Rosvall [RB10] a proposé une méthode de visualisation de communautés évoluant au cours du temps qui utilise son algorithme statique InfoMap. Sur chaque instantané, l'algorithme est lancé plusieurs fois. Un premier traitement est effectué pour ne garder dans les communautés que les nœuds qui apparaissent effectivement ensemble dans toutes les solutions (InfoMap étant sans recouvrement, il n'y a pas de problèmes de co-appartenance à plusieurs communautés). C'est sur ces communautés « fiables » que la seconde partie, visant à faire correspondre les communautés d'un instantané avec celles de l'instantané précédent, est effectuée. Le processus est illustré sur la figure 2.17.

Cependant, cette technique, qui élimine effectivement une bonne partie du bruit, présente

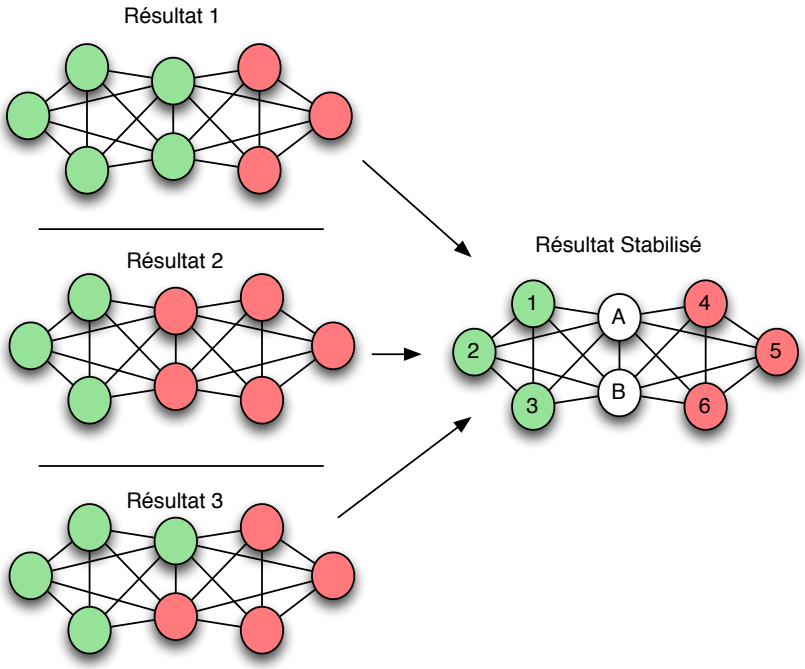


FIGURE 2.17 – *Exemple de gestion des oscillations par exécutions répétées. Le même algorithme a été lancé plusieurs fois sur le même réseau, conduisant à 3 résultats différents. A partir de ces résultats, une solution stable est construite. Les nœuds 1, 2 et 3 on toujours été détectés ensemble, et forment donc un cluster. De même pour 4, 5 et 6. En revanche, A et B n'ont pas toujours été placés dans les mêmes clusters : leur classement est oscillant. Notons que dans des cas plus complexes, il peut être difficile de trouver des clusters fiables. Se reporter à [RB10] pour plus de précisions*

quelques problèmes : d'une part elle demande beaucoup de calculs, et d'autre part elle semble beaucoup plus difficile à appliquer dans le cas de communautés avec recouvrement. Il devient alors difficile de déduire, pour deux nœuds n'appartenant plus exactement aux mêmes communautés, quelle différence vient des changements de l'une ou l'autre des communautés auxquelles ils appartiennent.

Seifi et al. [SG12] étudient également la stabilité de ce qu'ils appellent des communautés cœurs, c'est à dire les éléments des communautés restant stables dans plusieurs exécutions du même algorithme stochastique sur le même réseau, ou de légères variations du même réseau.

2.6.4 Utilisation d'instantanés

La quasi totalité des méthodes proposées, à l'exception de celles de Falkowski, Shang et Li, [FBS08, SLX+12, LHB+12] qui sont parmi les plus récentes, n'utilisent pas directement les données de graphes temporels, mais travaillent sur des instantanés.

Il y a un avantage évident à cette approche : elle permet de se retrouver en terrain connu, de réutiliser directement les acquis des travaux précédents sur les graphes statiques. Dans certains cas, sans même avoir à introduire de nouvelles notions : application d'un algorithme sur chaque instantané, auquel on adjoint un post-traitement. Nous avons expliqué précédemment le problème d'instabilité qui rend ces méthodes peu efficaces. D'autres méthodes ont alors été proposées :

- Définition de métriques prenant en compte les communautés obtenues à l'instant précédent pour le calcul des communautés de l'instant courant.
- Création d'un seul graphe constitué en agrégeant tous les instantanés.

La première méthode n'élimine pas forcément le problème de l'instabilité, car une nouvelle détection est bien effectuée à chaque nouvelle étape. Cependant, elle peut largement l'atténuer. Le problème est de trouver un compromis entre la correspondance au résultat précédent et la prise en compte des modifications.

La seconde méthode élimine bien le problème de l'instabilité, puisqu'une seule détection est faite. Cependant, cela rend la détection des fusions et des divisions de communautés difficile, puisque la détection est faite en une seule fois sur toutes les étapes d'évolution. De plus, toutes les méthodes présentées jusqu'à maintenant utilisant ce principe ne sont pas non plus capables de trouver des recouvrements entre les communautés à un même moment.

En définitive, nous pensons que l'utilisation d'instantanés doit être réservée aux données pour lesquelles ces instantanés constituent la seule information disponible. Lorsque l'on souhaite étudier un réseau évoluant au cours du temps, il est en effet possible que les données dont on dispose soient parcellaires. Par exemple, si le réseau a été obtenu en interrogeant des individus, ou si l'accès aux données est complexe et qu'il ne peut se faire qu'à intervalle régulier, on ne dispose parfois pas du détail des évènements survenus sur le réseau, mais juste de ces instantanés obtenus à intervalle plus ou moins régulier. Dans ce cas, étant donné les informations manquantes, il est clair que l'approche par instantanés fait sens.

Cependant, avec l'arrivée des technologies les plus récentes, il est fréquent d'avoir accès à l'intégralité des événements. Cela est évident pour les technologies liées à l'informatique, telles que les réseaux sociaux web 2.0, les réseaux d'ordinateurs ou de communication. Les données bibliométriques telles que réseaux de citations, de co-auteurs ou de co-citations, fournissent également des réseaux temporels détaillés.

Mais c'est également vrai dans d'autres domaines. Par exemple, si des éthologistes souhaitent monitorer le comportement social d'animaux, ils ont à leur disposition des outils tels que les GPS, qui permettrons de suivre les individus en temps réel. De même, des travaux récents, comme

ceux de Cattuto et Barrat [CVdBB+10] présentent des outils pour suivre les interactions sociales entre individus dans des hôpitaux, des musées ou des conférences à l'aide de puces RFID : lorsque deux personnes équipées sont dans un rayon donné — voire, avec les méthodes les plus évoluées, uniquement lorsqu'elles sont face à face durant une période assez longue — cela va déclencher un événement. Par un simple post-traitement, toutes ces informations peuvent être simplement converties en un réseau temporel détaillant chaque modification topologique.

On peut se demander quelle est la différence fondamentale entre une séquence d'instantanés et un réseau temporel : après tout, il peut y avoir une équivalence d'information entre les deux, si à chaque modification du réseau temporel on associe un instantané du réseau, identique à l'instantané précédent à l'exception de la modification introduite. Mais on se rend compte aisément que cette approche par instantané pour un suivi détaillé de l'évolution d'un réseau n'est pas adaptée. Imaginons un grand réseau statique stocké en mémoire, et dont le poids serait de 100Mo (ce qui reste raisonnable). Imaginons maintenant que nous disposons de 100.000 pas d'évolution de ce réseau, constitués par des opérations atomiques (addition/suppression d'un lien). L'information codant cette modification pourra aisément être représentée à l'aide d'1ko. Avec un réseau temporel, la totalité de l'information (réseau initial + toutes les étapes) pourra donc être stockée sur 200Mo. En revanche, stocker l'intégralité des instantanés correspondants représenterait un volume d'environs 100 To, ce qui est considérable.

Or, pour toutes les méthodes utilisant ce paradigme, il est nécessaire de détecter des communautés sur ces instantanés, que ce soit indépendamment, en séquence ou sur un graphe agrégé contenant toutes les informations. Or, même les algorithmes de détection de communautés les plus rapides ne sont pas capables aujourd'hui de traiter un tel volume d'information en un temps fini. Utiliser des instantanés signifiera donc obligatoirement perdre de l'information.

Une première conséquence est que les informations d'évolution seront moins fines, moins détaillées. La personne souhaitant travailler sur un réseau dynamique va de plus devoir faire un choix pour créer ses instantanés : à quelle fréquence les faire ?

Une fréquence élevée signifiera beaucoup d'information, des temps de traitement longs. Une fréquence plus faible, elle, entrainera une forte perte d'information. Il deviendra alors possible de perdre des données importantes : modifications topologiques trop grandes entre les instantanés consécutifs, évènements apparaissant et se terminant dans la période entre deux instantanés, etc.).

Définir une période d'intervalle pourra également se révéler complexe dans un réseau dont la vitesse d'évolution n'est pas linéaire. Par exemple, dans un réseau social devenant de plus en plus populaire, ou dans un réseau de télécommunications plus actif à certaines périodes qu'à d'autres (week-end, vacances, évènements particuliers, etc.). Dans ce cas, choisir une période fixe entre deux instantanés signifiera que le nombre de modifications ayant lieu durant la période pourra varier fortement. Choisir au contraire un nombre de modifications fixe amènera à avoir des instantanés séparés par des périodes de temps fort différents.

Pour toutes ces raisons, il nous semble que pour traiter le cas idéal de réseaux temporels pour lesquels nous disposons de l'information complète — et ce cas de figure devient de plus en plus fréquent — l'utilisation d'algorithmes basés sur des séquences d'instantanés n'est pas la solution la plus pertinente.

Cette approche est par contre évidemment valable pour les réseaux ayant été initialement collectés de cette manière.

2.7 Bilan de l'état de l'art

Comme nous l'avons exposé dans ce tour d'horizon des méthodes de détection de communautés, beaucoup de solutions ont été proposées. De par l'absence de définition précise de ce qu'est une bonne communauté, de nombreux travaux parvenant à des résultats fortement différents ont pu rencontrer un égal succès. Aujourd'hui cohabitent des méthodes capables ou non de détecter des hiérarchies de communautés, des communautés avec ou sans recouvrement, des communautés devant forcément contenir des fermetures transitives (cliques) ou, au contraire, centrées sur des hubs. Tous ces travaux sont intéressants, et donnent souvent des visions complémentaires d'un réseau.

Si le domaine de la détection de communautés statiques sans recouvrement semble aujourd'hui arrivé à maturité, avec plusieurs méthodes se révélant à la fois rapides, performantes et élégantes dans leur concept, nous pensons que le problème de la détection de communautés avec recouvrement n'a pas encore atteint ce stade.

Quant aux communautés dynamiques, nous pensons que l'on est encore dans une phase d'exploration, ce qui explique la grande variété d'approches utilisées.

Comme on peut le voir sur les deux tableaux présentant la chronologie des publications (tableaux 2.1 et 2.2), la majorité des travaux sur les communautés avec recouvrement et sur les communautés dynamiques sont très récents, la majorité ayant été faits au cours de ces 3 dernières années, et donc durant la période de cette thèse. C'est donc logiquement sur ces problèmes que nous nous sommes penchés, comme nous le présenterons dans la suite de ce document.

3

Conception d'un algorithme de détection de communautés dynamiques

Contents

Résumé du chapitre Ayant présenté l'état de l'art de la détection de communautés dans le chapitre précédent, nous allons maintenant pouvoir nous consacrer à décrire en détail notre propre solution. Dans la section 1, nous allons détailler les exigences que nous cherchons à prendre en compte, en particulier combler les faiblesses des solutions existantes identifiées dans l'état de l'art. Notre but est en effet de pouvoir traiter des grands graphes, ayant de plus une forte dynamique, tout en fournissant des communautés cohérentes dans le temps, ce qu'aucun algorithme actuel ne parvient vraiment à faire de manière tout à fait convaincante, comme exposé au chapitre précédent. La section 2 sera consacrée à une discussion sur les communautés que nous cherchons à trouver. Étant donné l'absence de définition consensuelle pour ce qu'est une communauté, on dit souvent que chaque algorithme a sa propre définition. Cette section vise donc à expliquer ce que nous considérons être une communauté, et les critères qui fondent ce jugement.

La section 3 est particulièrement importante puisqu'elle détaille la solution que nous proposons pour la détection de communautés dynamiques. Elle se décompose en la présentation d'un méta-algorithme, iLCD, puis de deux implémentations de ce méta-algorithme.

Enfin, la section 4 traite des aspects pratiques de la détection de communautés dynamiques. On y trouvera des précisions importantes, en particulier pour les lecteurs non familiers des graphes dynamiques. Y sont traités le problème du type de réseaux temporels pouvant être étudiés, la création de réseaux temporels adaptés, et le rôle joué par les paramètres de la solution que nous proposons.

3.1 Exigences à prendre en compte

Ayant étudié les différentes méthodes proposées précédemment pour la détection de communautés dynamiques, nous avions relevé un certain nombre de faiblesses, que nous avons présentées dans la dernière partie de l'état de l'art. Nous avons donc décidé de chercher à proposer une méthode qui comblerait au moins en partie ces faiblesses. Notre objectif est de pouvoir manipuler des réseaux temporels, évoluant en temps réel. Nous souhaitons également pouvoir appliquer notre méthode à de grands graphes, et, enfin, nous voulons assurer la stabilité dans le temps de nos communautés.

3.1.1 Traitement de réseaux temporels

Une des principales critiques que nous avons adressé aux méthodes existantes porte sur la question de l'utilisation de séquences d'instantanés, plutôt que de concevoir un algorithme qui traite directement un graphe temporel. Pour être plus clairs, reprenons la définition de ce qu'est un graphe temporel. Nous utilisons ici celle de l'article *Temporal Networks*, de Holme et Saramäki [HS12], qui distingue deux types de réseaux temporels :

- Les *séquences de contact* : dans ces réseaux, les liens n'ont pas d'existence dans la durée. Un lien est donc caractérisé par un triplet (i, j, t), où i et j sont des nœuds du graphe, et t l'instant où le lien a été activé. Ce type de graphe est particulièrement adapté à l'étude de la dynamique sur les graphes, comme par exemple la diffusion d'informations ou la propagation d'épidémies. On peu citer comme réseaux de terrain directement modélisables les réseaux de communications asynchrones (e-mails, courriers), ou les relations entre individus pour lesquelles la durée de l'interaction est négligeable (par exemple, réseaux de communications téléphoniques, réseaux de contacts pour diffusion de maladies, etc.)
- Les *graphes d'intervalles* : dans ces réseaux, on va considérer qu'un lien entre deux nœuds existe pour une ou plusieurs périodes. Le lien est donc défini par les deux nœuds qu'il re-

lie, mais également par une séquence d'intervalles $Te = \{(t1, t'1), \ldots, (tn, t'n)\}$ qui représentent les périodes durant lesquelles le lien existe. Le spectre d'utilisation de ce formalisme est plus large, puisqu'il est plus général (on peut représenter des séquences de contacts en définissant des séquences d'intervalles sans durée, de type $\{(t1, t1), \ldots, (tn, tn)\}$). On peut donc savoir à un instant donné les liens qui sont actifs : ce sont tous les liens pour lesquels l'instant courant est contenu dans l'une de ses séquences d'intervalle.

C'est à ce second type de graphes que l'on va s'intéresser. En effet, notre objectif est de détecter des communautés dynamiques. Pour une communauté ayant une existence prolongée dans le temps, il est logique que les nœuds et les liens qui la composent aient eux-mêmes une existence prolongée. Notons cependant que cela ne veut pas dire que l'on ne pourra pas traiter des réseaux temporels aux communications instantanées telles que des échanges d'e-mail. Mais dans ce cas, nous appliquerons un algorithme simple afin de transformer des suites d'interactions répétées, si elles satisfont certains critères de fréquence et de régularité, en un lien ayant une existence dans la durée. Nous proposerons une telle transformation dans la section 3.4.1.1.

Le formalisme que nous avons choisi d'utiliser est en fait légèrement différent de celui présenté. Il est équivalent en terme de représentativité, mais nous a semblé plus approprié dans le cas de données traitées en temps réel. En effet, lorsque l'on manipule des réseaux temporels de terrain, l'une des possibilités intéressantes est de travailler au fur et à mesure de l'évolution du réseau. Par exemple, dans le cas d'un réseau social Web 2.0, il serait intéressant de détecter les communautés d'utilisateurs, mais aussi de les maintenir au fur et à mesure que de nouveaux utilisateurs rejoignent ou quittent le réseau, que de nouveaux liens se créent, etc.

Le formalisme des graphes d'intervalles est bien approprié dans le cas de réseaux pour lesquels la totalité des évolutions sont connues à l'avance. Dans le cas où de nouvelles modifications peuvent venir se rajouter alors même que nous avons commencé la détection, il nous a semblé plus adéquat d'adopter une approche légèrement différente. Nous avons choisi de modéliser le réseau sous la forme d'une suite d'évènements. Un événement est un quadruplet :

- i, j : deux nœuds affectés par l'événement
- a : une action pouvant être une création ou une suppression de lien.
- t : un instant

Ainsi, l'événement (i, j, a, t) décrira l'action a(création ou suppression de lien) entre les nœuds i et j, à l'instant t. Nous avons choisi de nous limiter aux réseaux non dirigés et non pondérés. Les actions (i, j, a, t) et (j, i, a, t) sont donc équivalentes. Après une opération de création pour le lien (i, j), la seule opération possible sur ce lien est une suppression.

En représentant le réseau sous cette forme, on peut donc aisément connaître l'état du réseau à un instant donné, il suffit d'exécuter dans l'ordre tous les évènements depuis l'état actuel du réseau jusqu'à l'instant qui nous intéresse. La quantité d'information nécessaire pour représenter de nombreuses étapes d'évolution est donc bien inférieure à celle qui aurait été nécessaire avec des séquences d'instantanés.

Notre objectif est donc de concevoir un algorithme capable de traiter tous les détails de l'évolution du graphe, c'est à dire chaque ajout et chaque suppression de liens.

3.1.2 Traitement de réseaux évoluant en temps réel

Nous avons évoqué dans la section précédente la question des réseaux évoluant en temps réel. Un objectif de l'algorithme que nous voulions concevoir était de pouvoir être capable de traiter un réseau de cette manière, au fur et à mesure de son évolution. Ce n'est pas le cas de tous les algorithmes : certains, comme décrit dans l'état de l'art, étudient simultanément toutes les étapes d'évolution ([MRM+10], [BJRF07], [AG10b]), et nécessitent donc de travailler sur un

graphe composé en agglomérant toutes les données liées aux différentes étapes d'évolution du graphe. Cela permet d'appliquer un certain « lissage », en gommant par exemple des irrégularités dues à l'incertitude des algorithmes lors de détections successives statiques, mais empêche de travailler sur des réseaux évoluant en temps réel.

Dans notre cas, nous voulions être capable de trouver le nouvel état des communautés uniquement à partir de l'état précédent et des nouvelles évolutions du réseau.

3.1.3 Gérer de grands graphes

Comme expliqué en introduction, nous nous intéressons particulièrement aux données issues de grands graphes de terrains. Il est parfois tentant, face à la complexité de la détection de communautés sur les graphes de terrain, de concevoir des méthodes complexes, sans se soucier de leur applicabilité à de grands graphes. Déjà sur les graphes statiques, certains algorithmes efficaces ont des temps de calcul prohibitifs.

Prenons l'exemple d'OSLOM [LRRF11], qui est aujourd'hui l'un des meilleurs algorithmes pour détecter des communautés avec recouvrement. Son mécanisme étant à base d'essais-erreurs jusqu'à atteindre une stabilisation, et son caractère fortement stochastique nécessitant de multiples exécutions d'un même calcul en vue d'un renforcement, son exécution est très longue sur de larges graphes. Avec la croissance continuelle de la puissance des machines, il est cependant possible de l'utiliser sur des graphes composés de millions de nœuds, à raison de plusieurs heures de calcul. Cependant, il s'agit là de graphes statiques. Si l'on cherchait à appliquer le même algorithme sur des étapes successives d'un graphe évoluant fortement, les temps de calcul deviendraient trop longs. Dans ce cas, le temps de calcul nécessaire au traitement de toute l'évolution est égale au temps de calcul nécessaire pour traiter un graphe statique multiplié par le nombre d'évolutions du graphe.

Pour traiter de larges graphes dynamiques, il faut au contraire que la complexité de l'algorithme soit seulement déterminée par l'évolution du graphe (Plus éventuellement le coût d'une détection statique initiale).

3.1.4 Stabilité dans le temps des communautés : approche globale et approche itérative

Comme nous l'avons expliqué au chapitre précédent, la question de la stabilité des communautés au cours du temps est cruciale. Il ne faut pas que l'algorithme puisse décider, dans une configuration similaire, de diviser les communautés d'une manière à un instant et d'une autre manière à un autre instant. Il y a principalement deux façons de s'assurer de cette continuité :

- *Méthode globale* : on peut effectuer la détection de communauté en considérant simultanément toutes les étapes d'évolution. Cela peut se faire comme dans les méthodes de Mucha et de Ben Jdidia [BJRF07, MRM+10], en reliant les nœuds équivalents dans des instantanés consécutifs, comme dans la méthode d'Aynaud et al [AG10b], en optimisant une métrique sur tous les instantanés simultanément, ou enfin comme dans celle de Mitra [MTR11], en constituant un réseau statique qui reprend les informations du réseau dynamique. Un algorithme statique est ensuite utilisé. Les instabilités locales sont ainsi gommées par une vision globale dans le temps du réseau.
- *Méthode itérative* : on peut aussi considérer le réseau étapes par étapes, et, pour chaque nouvelle étape, prendre en compte le résultat de l'étape précédente dans le calcul de la nouvelle, de façon à gommer l'instabilité.

Chaque méthode a des avantages et des inconvénients, que l'on peut détailler :

3.1.4.1 Avantages de la méthode globale

- S'il y a beaucoup de bruit dans les données, comme des liens ou nœuds ayant tendance à apparaître et à disparaître alors qu'ils ne le devraient pas, ces algorithmes pourront facilement s'abstraire de ces informations en voyant qu'elles sont des « anomalies », en regard de l'ensemble des étapes d'évolution.
- Plus grande facilité à repérer des événements périodiques : en analysant tout l'historique, il est plus facile de se rendre compte de patterns apparaissant de manière similaire à des périodes données.

3.1.4.2 Inconvénients de la méthode globale

- Il n'est pas possible de traiter des réseaux évoluant en temps réel. En effet, la totalité des étapes d'évolution du graphe sont requises pour faire ce type de détection. Et si l'on faisait une première détection avec les données connues à un moment donné, puis une nouvelle détection à une étape plus tardive, en ayant eu connaissance des modifications les plus récentes, les résultats correspondant aux étapes antérieures pourraient tout à fait ne plus être cohérents avec ceux obtenus lors de la première détection.
- On change la nature du problème étudié. Avec les graphes statiques, on étudiait simplement la topologie du réseau. Des nœuds et des liens représentant une relation entre ces nœuds. Avec l'approche itérative, le problème étudié à chaque étape est, également, un problème purement topologique : des nœuds et des liens ayant toujours la même signification, dont certains peuvent être nouveaux, et les communautés de l'étape précédente. Avec l'approche globale, le problème étudié est composé de plusieurs nœuds temporels représentant le même nœud topologique, et parfois on fait même apparaître des liens qui ne représentent plus la relation classique entre les nœuds du réseau.
- Les données à traiter sont complexes. Toutes les données doivent être manipulées simultanément, ce qui va souvent demander plus de ressources mémoires, ou même amener une plus grande complexité.
- Il est difficile de se passer de la représentation par instantanés. Nous avons expliqué précédemment pourquoi nous ne considérions pas les instantanés comme une solution pertinente. Si de multiples approches globales ont été présentées en utilisant les instantanés, une seule d'entre elles [MTR11] à notre connaissance étudie directement le graphe temporel. Cependant, cette méthode est spécifique à des réseaux dans lesquels les deux extrémités des liens ne sont pas contemporains (réseaux de communications indirectes, réseaux de citations, etc.), et ne peut s'appliquer à tous les cas. Si l'on souhaite s'abstraire de la méthode par instantanés, la représentation adéquate du réseau temporel étudiable globalement reste à définir.

Les avantages et inconvénients de la méthode itérative sont donc symétriques :

- Possibilité de traiter des réseaux évoluant en temps réel : à chaque nouvelle modification du réseau, l'algorithme en est informé, et peut calculer les modifications correspondantes à apporter aux communautés de l'étape précédente.
- Conservation de la nature du problème étudié : à chaque étape, on retrouve bien un graphe statique ordinaire
- Difficulté à reconnaître les évènements périodiques : on ne dispose *a priori* que de l'état précédent du réseau. Il est bien sûr possible de faire un algorithme avec mémoire, mais ce n'est pas induit par la méthode
- Sensibilité au bruit : si un événement se produit, tel que la disparition accidentelle d'une

grande quantité de nœuds par (problème technique, ...), cela peut fortement affecter les résultats postérieurs.

En conclusion, la méthode à choisir dépend à notre avis du résultat que l'on souhaite obtenir. Dans notre cas, nous souhaitions pouvoir étudier de grands graphes de terrains, en particulier des réseaux sociaux web 2.0, pour lesquels nous désirions pouvoir calculer les communautés en temps réel. Le problème du passage à l'échelle était aussi important, puisque nous voulions que notre méthode puisse être utilisée sur de très grands graphes ayant de très nombreuses étapes d'évolution. Nous expliquerons dans la section 3.4.1.1 comment les problèmes de bruit dans les données peuvent être fortement réduits, puisque c'est l'un des défauts de cette méthode. En revanche, nous n'avons pas proposé de méthode pour repérer des patterns temporels ou des événements périodiques : cela pourrait faire l'objet d'un travail ultérieur, par exemple en utilisant une méthode globale sur les résultats obtenus par notre algorithme itératif.

3.2 Détection de communautés, oui, mais lesquelles ?

Comme nous l'avons expliqué précédemment, il n'existe pas de définition admise par tous de ce que sont des communautés. Dans ce chapitre, nous allons expliquer quelles sont les communautés que nous allons chercher dans un réseau, et pourquoi nous considérons qu'il s'agit d'une définition intéressante de ce que sont des communautés.

3.2.1 Définition de communauté, densité et séparation

La définition de communauté sur laquelle à peu près tout le monde s'accorde peut être énoncée de la manière suivante : « Une communauté est un ensemble de nœuds fortement connectés entre eux et plus faiblement connectés au reste du réseau ». Les communautés doivent être *denses*, et être clairement *séparées* du reste du réseau. Dans les réseaux de terrain, on pourrait rajouter — et c'est cette propriété qui fait que la détection de communautés a tant d'applications — que les communautés sont des ensembles de nœuds qui sont « pertinents au niveau du sens », c'est à dire qu'un spécialiste du réseau, en voyant les nœuds composant la communauté, sans rien connaître de la topologie du réseau, pourrait dire que la communauté est valide.

Le problème est que cette définition est extrêmement vague, et que, pour un même réseau, il peut souvent exister de nombreux groupes de nœuds fortement connectés entre eux et plus faiblement connectés au reste du réseau. Cette définition est en effet composée de deux critères contradictoires, et les solutions intéressantes se situent dans le compromis entre elles. Si l'on pousse chaque critère à son extrémité, on obtient des définitions caricaturales des communautés, qui sont parfois utilisées dans des cas simples :

- Les cliques maximales peuvent être considérées comme des communautés. Dans ce cas, les communautés sont les plus denses possibles. En revanche, elles vont rester fortement connectées avec le reste du réseau (notons que cela produit des communautés avec recouvrement)
- Les composantes connexes fournissent au contraire des communautés parfaitement séparées les unes des autres, mais rien ne garantit qu'elles vont être denses.

Si l'on prend l'exemple de quelques algorithmes, on s'aperçoit que tous respectent globalement la définition, avec l'idée de compromis entre densité et séparation :

- la modularité d'une communauté consiste à comparer la fraction de liens existant entre les nœuds de la communauté à la fraction qu'il y en aurait dans une version recablée aléatoirement du réseau. Plus la différence est grande, plus le score de modularité est élevé. On retrouve directement le fait que la communauté doit être dense. Le critère de communautés

faiblement connectées est, lui, implicite : comme le découpage en communauté est total et unique (chaque nœud appartient à une et une seule communauté), tout lien qui n'est pas dans une communauté ne pourra pas contribuer à augmenter le score de modularité global (la modularité globale que l'on cherche à optimiser est la somme des modularités de toutes les communautés). Tout lien inter-communauté conservé est donc pénalisant.

- Dans InfoMap [RB08], un bon découpage est tel que la description du déplacement d'un marcheur aléatoire sur le réseau soit la plus courte possible, sachant que chaque déplacement entre des communautés entraine une « pénalité », puisqu'il faut décrire ce changement. Le critère de séparation est donc clairement défini, puisque, plus il y aura de liens inter-communautés, plus un marcheur aléatoire aura de probabilités de les emprunter. C'est cette fois le critère de densité qui est implicite : plus une communauté est grande, plus la description d'un déplacement interne à la communauté devient pénalisant. Les communautés doivent donc être aussi petites que possibles tout en contenant un maximum de liens, ce qui, dans un découpage en communautés sans recouvrement, est similaire à la définition initiale des communautés.
- Dans OSLOM [LRRF11], les communautés sont constituées indépendamment les unes des autres, par un processus d'essais-erreurs :
  - on ajoute à une communauté les nœuds qui ont plus de liens avec la communauté qu'ils n'en auraient dans un réseau recablé : on ajoute donc à la communauté des nœuds créant beaucoup de liens internes, ce qui renforce la densité de la communauté.
  - Au contraire, on rejette les nœuds ayants moins de liens que ce qui est prévu : ces liens deviennent des liens inter-communautés. On sait donc par construction qu'il y en a relativement peu.

On peut ainsi retrouver dans chaque méthode de détection de communautés un moyen de faire un compromis entre la densité interne des communautés et la séparation du reste du réseau, y compris dans les algorithmes avec recouvrement. C'est en quelque sorte la définition de communautés adoptée par chaque solution.

Cette définition est cruciale, et une définition imparfaite peut amener à des communautés peu pertinentes. Si l'on prend l'exemple de l'algorithme CPM, on sait que les communautés sont composées de cliques ayant tous leurs nœuds en commun sauf un (cf présentation détaillée dans l'état de l'art, 2.4.8). On assure donc que les communautés sont composées d'éléments denses (les cliques). Cependant, rien n'assure que la communauté dans sa globalité est dense. C'est pour cela que l'on peut constituer des chaines de cliques, éventuellement très longues, qui conduiront donc à des densités internes globales faibles, comme cela avait été présenté dans la figure 2.8.

Il est donc clair que tout algorithme de détection de communautés doit offrir ce compromis entre *densité* et *séparation*. Nous reparlerons plus en détail de ces notions au chapitre 4.1, où nous proposerons également une méthode pour évaluer la qualité d'un découpage en communautés, basée sur ces notions de densité et de séparation.

3.2.2 Communautés : relatives ou intrinsèques

Il y a un autre critère entrant en jeu dans la définition de communautés que l'on adopte. Ce critère a pourtant peu été évoqué dans la littérature, mais il nous semble important : il s'agit de la question des communautés intrinsèques ou relatives.

Les communautés relatives sont des communautés dont la définition dépend des propriétés de la totalité du réseau étudié.

Les communautés intrinsèques ne dépendent pas du réseau dans lequel elles sont incluses, mais uniquement de leur topologie interne et de leur frontière avec le reste du réseau.

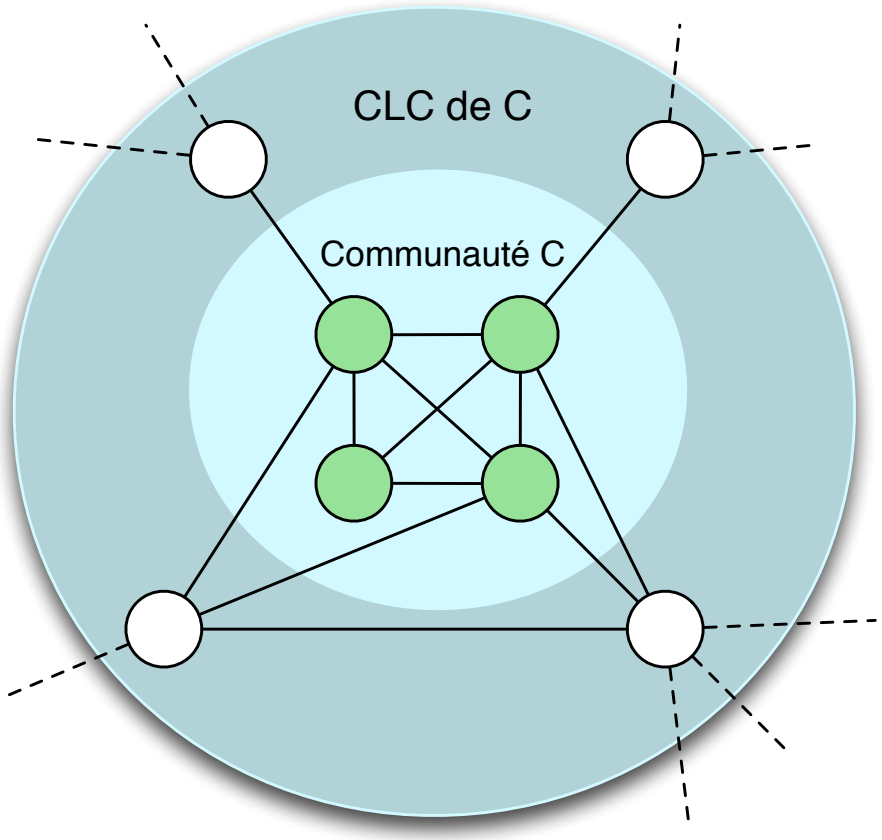


FIGURE 3.1 – *Illustration d'une communauté et de sa CLC. La CLC comprend les liens externes à la communauté ainsi que tous les nœuds aux extrémités de ces liens. Les liens représentés en pointillés, c'est à dire les liens de ces nœuds vers d'autres nœuds, n'en font pas partie. Un algorithme de détection de communautés intrinsèques, en connaissant seulement la CLC, peut savoir si la communauté correspondante est une bonne communauté. En revanche, un algorithme à communautés relatives ne le peut pas.*

Définissons la **Configuration Locale de Communauté de C (CLC(C))**, une configuration constituée par le sous réseau correspondant à une communauté C, à laquelle on ajoute tous les nœuds ayant au moins un lien avec cette communauté (exemple sur la figure 3.1). Est-ce qu'un algorithme, en connaissant la CLC de C, et sans autre information sur le réseau, est capable de dire si C est une communauté ou non ? Si c'est le cas, on peut parler d'algorithme à communautés intrinsèques, sinon, à communautés relatives.

On peut faire un parallèle entre la notion de communauté intrinsèque et celle de communauté naturelle (*natural community*) utilisée par Lancichinetti et al. dans [LFK09]. La communauté naturelle d'un nœud A, $CN(A)$, est définie en utilisant la fonction de fitness φ, comme un ensemble de nœuds incluant A, et pour lequel on ne puisse ni ajouter ni retirer un nœud sans faire baisser la valeur de φ. Dans leur procédure, $CN(A)$ est obtenue en partant d'une communauté initiale ne contenant que A, qui ne peut être modifiée que par l'ajout d'un nœud voisin ou le retrait d'un nœud interne. On peut donc, en ne considérant que la CLC de $CN(A)$, dire qu'il s'agit d'une communauté naturelle pertinente.

On peut cependant faire deux distinctions entre les communautés naturelles et les communautés intrinsèques :

- La fonction de fitness φ utilisée dans le calcul de $CN(A)$ peut utiliser des informations sur l'ensemble du réseau, comme le fait la modularité par exemple. Or, dans notre définition des communautés intrinsèques, nous spécifions que l'on peut déterminer si une communauté intrinsèque est une communauté uniquement en utilisant sa CLC, sans autres informations sur le reste du réseau.
- Une communauté naturelle correspond à un nœud d'origine. Or, rien n'assure que toutes les communautés pertinentes du réseau puissent être trouvées en commençant par un nœud particulier. Il est par exemple probable que la communauté centrale de la figure 2.6 ne puisse être obtenue à partir d'aucun nœud d'origine.

L'immense majorité des approches proposées jusqu'à maintenant sont des approches relatives. Les seuls exemples d'approches utilisant des communautés intrinsèques sont CPM, et les approches avec recouvrement consistant à partir d'une graine et à étendre cette graine en utilisant la force de communauté (*community strength*), qui peut être calculée en connaissant uniquement la CLC de la communauté. [WJW09, BGMI05, LFK09, LRMH10]

Les deux approches, relatives et intrinsèques, ont des propriétés différentes, qui peuvent être des avantages ou des inconvénients selon ce que l'on va chercher dans le réseau.

3.2.2.1 Communautés relatives

Communautés adaptées au réseau Dans les approches relatives, une communauté n'a pas de propriétés en elle-même, les communautés sont différentes pour chaque réseau. Par exemple, sur la figure 3.2, un algorithme à communautés relatives ne fait pas le même découpage pour des configurations identiques. Cela est tout simplement dû aux caractéristiques du reste du réseau. Avec des communautés relatives, l'algorithme va adapter ce qu'il considère comme une communauté aux caractéristiques du réseau étudié.

Ceci semble donc un avantage. En revanche, cela conduit aussi à certains résultats moins évidents. Le même algorithme pourra par exemple ne pas trouver les mêmes communautés dans un réseau de petite taille et dans un réseau de grande taille, même si la structure en communauté est en fait identique. En effet, plusieurs approches, comme la modularité, mais aussi OSLOM, et d'autres, utilisent la probabilité d'existence des liens comme critère d'appartenance de nœuds à des communautés. Or, à degré moyen équivalent, la probabilité qu'un lien existe entre deux nœuds est plus importante dans un petit réseau que dans un grand réseau. Une communauté

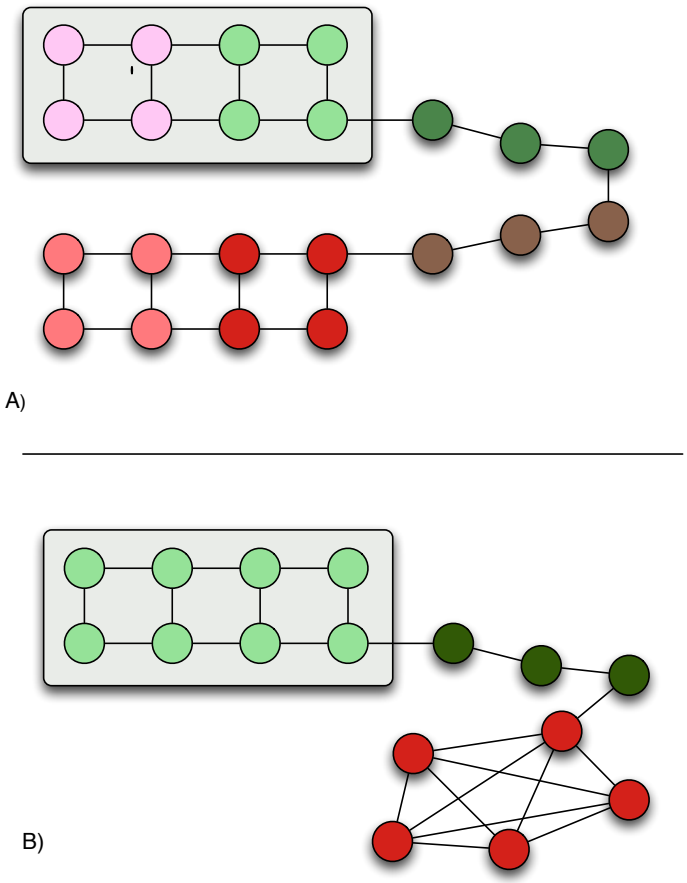


FIGURE 3.2 – *Exemple de cas illustrant le fait que les communautés relatives peuvent trouver des communautés différentes avec la même CLC. Dans le réseau A), les 8 nœuds entourés en gris ne sont pas découpés de la même manière que ceux du réseau B, ceci à cause de la différence dans une autre partie du réseau.*

donnée qui avait une densité interne supérieure à une configuration aléatoire dans un grand graphe pourra donc être considérée comme une configuration aléatoire dans un graphe plus petit. Cela semble fiable tant que l'on reste strictement sur cette limite entre configuration plus probable que dans l'aléatoire et moins probable. Mais cette limite n'étant généralement pas considérée comme suffisante, on va chercher des configurations largement au-dessus de l'aléatoire.

On en vient donc à décider que telle configuration était pertinente sur un grand graphe et ne l'est plus dans un petit graphe, ce qui peut paraitre contre-intuitif.

Problème des réseaux hétérogènes Le problème précédent est discutable, en revanche, sur les réseaux de terrain, il existe un autre problème plus difficile à écarter.

On sait maintenant, suite à des études sur les graphes de terrain, que ceux-ci sont généralement fortement hétérogènes. Concrètement, il existe des zones denses et des zones moins denses. Beaucoup de réseaux présentent une structure en core-periphery [BE00, LLDM09]. Le centre du réseau est alors une zone plus dense, constituée de nœuds de forts degrés souvent reliés entre eux. A la périphérie on trouvera des zones moins denses, avec souvent des nœuds de degrés plus faibles.

Certains réseaux sociaux sont également fortement hétérogènes. Par exemple dans des réseaux de blogs, on pourra constater des comportements différents selon les thèmes. Les blogs traitant —par exemple— de politique peuvent avoir tendance à être densément connectés alors que les blogs de littérature le seront moins.

Le problème est que les méthodes relatives considèrent le graphe comme un tout uniforme. La probabilité pour deux nœuds d'être liés l'un à l'autre sera la même en tout point du graphe. Dès lors, les communautés des zones moins denses du réseau risquent de ne pas être détectées, ou sur-découpées, tandis que les communautés des zones plus denses risquent d'être trop grandes, ou fusionnées.

Une autre propriété des communautés relatives est que si l'on détecte une communauté C dans un réseau R, puis que l'on considère la communauté C comme un réseau indépendant, et que l'on applique à nouveau l'algorithme uniquement sur ce nouveau réseau formé par la communauté C, l'algorithme va souvent trouver de nouvelles communautés à l'intérieur de cette communauté C.

3.2.2.2 Communautés intrinsèques

Lorsque les communautés sont définies de manière intrinsèque, la définition de chaque communauté est locale, relative à elle-même. L'exemple de CPM [PDFV05] est le plus simple : les communautés sont composées de cliques se chevauchant, qui resteront les mêmes quelles que soient les propriétés du reste du réseau. La communauté s'arrête là où il n'est plus possible de trouver de cliques en recouvrement avec les cliques de la communauté : là aussi, les propriétés du reste du réseau ne vont rien changer tant que la CLC de la communauté reste la même.

En conséquence, une CLC qui avait donné lieu à la détection d'une communauté dans un petit réseau peu dense donnera la même communauté dans un grand réseau dense.

De même, si l'on prend une communauté intrinsèque détectée dans un réseau quelconque, qu'on la considère comme un réseau et qu'on applique à nouveau l'algorithme de détection de communautés intrinsèques, celui-ci ne détectera qu'une seule communauté dans ce nouveau réseau, constitué de tout le réseau (sauf éventuellement si l'algorithme est stochastique).

Les algorithmes à communautés intrinsèques présentent un problème évident : ils ne sont adaptés qu'à certains types de communautés. Certains réseaux de terrain particuliers sont par exemple très peu denses, et présentent des structures en forme d'étoiles : un hub connecté à

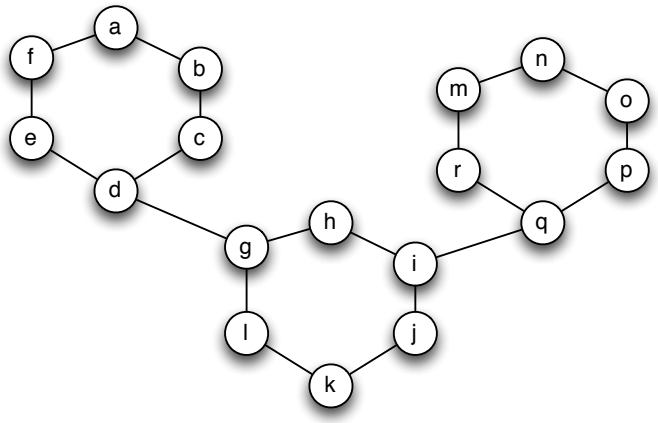


FIGURE 3.3 – *Exemple de communautés non trouvables par un algorithme intrinsèque. On peut ici identifier 3 communautés, en forme de cercles. Un algorithme à communautés relatives, qui peut décider de la nature des communautés en fonction de l'ensemble du réseau, peut en théorie identifier chaque cercle comme une communauté. En revanche, pour un algorithme à communautés intrinsèques, cela semble difficile : sortis du contexte de ce réseau, les nœuds a, b, c, d, e, f ne seraient probablement pas identifiés comme une communauté. On pourrait imaginer définir les communautés intrinsèques pour trouver des cycles, qui trouverait donc ces communautés. Mais, dans un graphe de terrain plus classique, cet algorithme ne trouverait pas des solutions correspondant à la définition habituelle de communautés.*

une douzaine de nœuds, avec quelques connexions entre eux, par exemple. Les algorithmes à communautés relatives peuvent détecter cela comme des communautés, car il s'agit de zones plus denses dans un réseau très peu dense. Au contraire, il n'est à priori pas possible pour un algorithme à communautés intrinsèques de détecter de telles communautés, puisque dans d'autres réseaux plus denses, cette structure est courante et non significative. Un algorithme à communautés intrinsèques qui détecterait de telles communautés dans un graphe peu dense trouverait énormément de communautés peu significatives dans un réseau plus dense.

De même, des communautés clairement séparées du reste du réseau mais présentant des propriétés particulières ne seront pas considérées comme des communautés, telles que des communautés formant des cercles de nœuds (voir figure 3.3)

A l'opposé, il est possible que dans certains réseaux, un algorithme détecte des communautés alors qu'il n'y en a pas, et que les structures soient simplement dues au hasard. Par exemple, dans un réseau aléatoire sans communautés, des zones plus denses vont inévitablement se former. Un algorithme à communautés relatives, pouvant voir l'intégralité du réseau, sera capable de dire que ces zones plus denses ne sont pas significatives dans ce réseau, étant donnée la densité moyenne.

En revanche, un algorithme à communautés intrinsèques ne pouvant se fier qu'à la communauté elle-même et sa CLC pour juger de sa pertinence, pourra se tromper et détecter cette irrégularité comme une communauté.

Par exemple, dans un réseau aléatoire dense sans communautés, des cliques vont inévitablement se former. L'algorithme CPM va donc les détecter comme des communautés, alors qu'un algorithme tel qu'InfoMap saura voir qu'il ne s'agit que d'un réseau aléatoire. On peut noter cependant que beaucoup d'algorithmes à communautés relatives ont aussi tendance à détecter des communautés dans des réseaux aléatoires. Mais il est théoriquement possible de l'éviter,

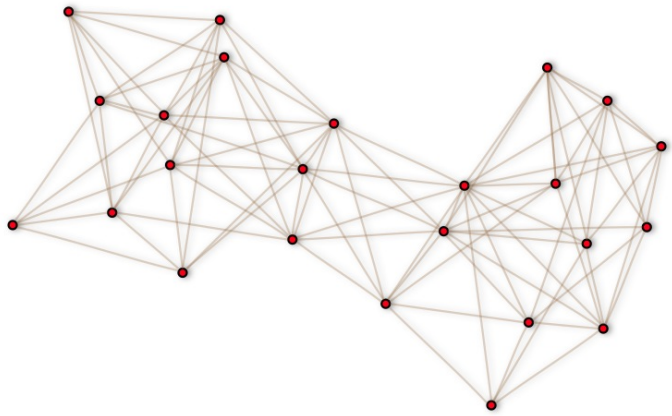


FIGURE 3.4 – *Exemple de communauté contre-intuitive trouvée par l'algorithme CPM avec une valeur de $k = 4$. Il semble y avoir deux groupes de nœuds distincts, mais ils sont reliés par une suite de cliques de taille 4.*

alors que ça ne l'est pas pour les communautés intrinsèques, qui par définition ne veulent pas tenir compte de l'ensemble du réseau.

Une fois encore, nous aurions tendance à dire que le choix de l'une ou l'autre des options dépend des réseaux que l'on souhaite étudier. Si la quasi-totalité des méthodes proposées à ce jour sont relatives, c'est à notre avis parce qu'à l'origine, on avait tendance à considérer que tous les réseaux pouvaient être traités de la même manière, que ce soient des réseaux biologiques, des réseaux informatiques, sociaux ou autres. Les réseaux étant clairement différents dans leurs propriétés, mais considérés comme suivant les mêmes règles, il était logique de chercher à proposer une méthode capable de trouver tout type de communautés dans tout type de réseaux, donc une méthode relative, pour pouvoir s'adapter aux différents réseaux.

Mais cette recherche de l'algorithme idéal capable de trouver toutes les communautés dans tous les réseaux est peut-être illusoire.

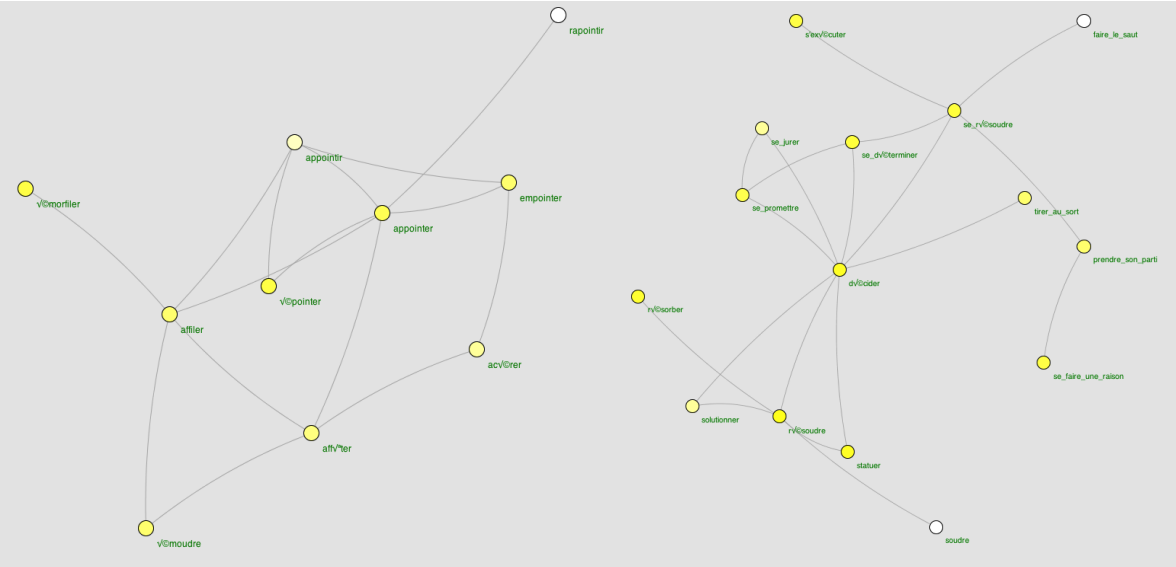
Les travaux récents commencent à différencier certains réseaux par leurs propriétés [YL12b, LKSF10]. Les communautés dans ces réseaux sont tellement différentes qu'il n'est pas aberrant de proposer des méthodes qui ne soient adaptées qu'à certains d'entre eux.

Pour notre part, en appliquant plusieurs algorithmes relatifs sur des réseaux de terrain à notre disposition, nous avons souvent remarqué la détection d'une super-communauté : une unique communauté contenant à elle seule un pourcentage élevé des nœuds du réseau, y compris dans de très grands réseaux. Ce résultat à été publié dans [NC10]. Il semblerait que ces communautés soient composées de la partie la plus dense du réseau, et, à notre avis, c'est un biais induit par le problème de l'hétérogénéité du réseau que nous avons décrit précédemment.

Comme nous voulions travailler sur des grands réseaux de terrain pouvant être très hétérogènes, nous avons penché pour la solution des communautés intrinsèques, avec l'idée de permettre à chaque communauté d'être définie par la topologie locale du réseau, et non par la topologie globale.

3.2.3 Communautés homogènes

En partant de l'idée qu'une communauté peut être définie uniquement à partir de sa CLC, sans connaître le reste du réseau, nous avons donc cherché ce qui pouvait définir une communauté, en considérant uniquement sa topologie interne et les liens qu'elle avait avec le reste du réseau.



(a) *avec Louvain* (b) *avec InfoMap*

FIGURE 3.5 – *Exemple de communautés trouvées avec des algorithmes à communautés relatives, dont la topologie interne ne semble pas très pertinente intuitivement. Ces communautés ont été trouvées dans un graphe de terrain avec, de gauche à droite, l'algorithme Louvain et l'algorithme InfoMap.*

Nous nous sommes rendu compte que l'un des problèmes qu'il y avait avec une définition telle que celle de CPM était qu'il était possible d'obtenir des communautés de « formes » extrêmement diverses. Ainsi, par exemple, l'illustration 2.8 montre des communautés de type chaînes de nœuds, qu'il est possible de détecter dans des réseaux à l'aide de CPM. L'illustration 3.4 montre un autre exemple, issu d'un réseau de terrain. Les nœuds sont disposés en utilisant un *force layout* (modèle masse ressort) [KK89]. On observe clairement que, ici, intuitivement, on souhaiterait trouver deux communautés. C'est d'ailleurs la solution que fournissent la majorité des algorithmes existant sur ce réseau.

Ce problème se retrouve avec les communautés relatives : les méthodes relatives n'impliquent pas d'avoir une quelconque propriété pour la structure interne des communautés. On peut donc trouver des communautés en étoile, en clique ou autre, selon le réseau. Si cela permet de trouver de nombreux types de communautés, cela peut aussi conduire à trouver des communautés peu intuitives. Sur la figure 3.5, deux exemples sont donnés de communautés trouvées sur un réseau de terrain (réseau de synonymie), dont la structure interne ne parait pas très convaincante, c'est à dire qu'il semble que, tant sémantiquement que topologiquement, ces communautés ne semblent pas complètement pertinentes.

En plus de la définition habituelle, la forte densité des communautés et leur séparation claire du reste du réseau, nous avons donc ajouté un critère, qui nous semble indispensable dans le cas des communautés intrinsèques : la cohérence topologique interne des communautés.

Nous n'allons pas donner ici de définition complète et explicite, parce que nous présenterons, dans la suite, deux implémentations de ce critère, que nous avons utilisé dans les deux versions de l'algorithme que nous avons proposé.

L'idée cependant est que les communautés que l'on cherche à trouver doivent avoir une topologie s'approchant de celle d'un graphe aléatoire :

– Elles ne doivent pas être composées de sous-parties pouvant être clairement séparées
– Elles ne doivent pas présenter une structure particulière, telle qu'une forme en chaine, en chaine bouclante (cercle), en étoile ou en grille.

Selon cette définition, une communauté est donc « atomique », dans le sens où on ne peut pas trouver d'autres communautés pertinentes à l'intérieur d'elle-même. Si l'on souhaitait faire une décomposition hiérarchique en communauté, rien ne l'empêcherait, mais on devrait dans ce cas utiliser la méthode employée par exemple par l'algorithme Louvain, consistant à détecter d'abord un premier jeu de communautés, transformer le réseau en un nouveau réseau dans lequel les communautés deviennent des nœuds, et appliquer à nouveau l'algorithme pour obtenir un niveau supérieur de décomposition, dans lequel les communautés seraient à nouveau atomiques.

On peut noter également que l'ajout de ce critère permet d'éviter la grande hétérogénéité des communautés trouvées par un algorithme relatif dans un grand réseau. Il est en effet fréquent qu'à un même niveau de décomposition, ces algorithmes trouvent de petites communautés, composées d'une poignée de nœuds et satisfaisant aux critères précédent, mais aussi de larges communautés composées de centaines de nœuds qui pourraient facilement être décomposées en communautés plus petites. Cette présence de communautés atomiques et de communautés décomposables dans un même découpage nous semble souvent peu pertinente.

3.3 Algorithme proposé

Dans cette section, nous allons présenter en détail l'algorithme que nous avons développé. Nous avons, en fait, développé et publié deux versions de l'algorithme, qui suivent le même méta-algorithme, le même principe général, mais diffèrent cependant sur plusieurs points. Nous allons d'abord présenter le méta-algorithme, utilisé par les deux versions et qui pourrait également être réutilisé par d'autres méthodes. Nous allons ensuite présenter succinctement la première version de l'algorithme [CAH10], ses résultats, ainsi que les limites qui nous ont conduit à proposer une deuxième version [CA11], que nous présenterons alors en détail.

3.3.1 Méta-algorithme iLCD : détection longitudinale de communautés

Comme expliqué précédemment, il existe plusieurs approches pour détecter des communautés dynamiques dans un réseau temporel. Comme nous ne souhaitions pas utiliser une méthode basée, comme les précédentes, sur des instantanés successifs, nous avons commencé par concevoir un méta-algorithme permettant de traiter des réseaux temporels directement, sans passer par un prétraitement. Nous l'avons appelé iLCD, pour intrinsic Longitudinal Community Detection.

3.3.1.1 Méta-algorithme inspiré des systèmes multi-agents

Le méta-algorithme proposé est inspiré de principes issus des systèmes multi-agents (SMA). Sans entrer dans les détails, un système multi-agent est défini comme un système composé d'un ensemble d'agents, qui évoluent dans un environnement et qui ont la possibilité d'interagir. Une caractéristique importante des agents est d'être, au moins partiellement, autonomes. Pour plus d'information, on peut par exemple se référer à [Fer99][SS02] [BGPP03] [Ser05] [DMSGK06].

Nous avons choisi cette approche car les SMA sont une solution particulièrement adaptée au traitement de problèmes dynamiques. Comme dans tout SMA, il nous faut définir un environnement, ainsi que des agents, situés dans cet environnement, capables d'interagir entre eux et avec celui-ci.

Environnement Le réseau constitue un environnement, qui évolue par lui-même. L'environnement est donc un ensemble de nœuds et de liens. Les évolutions de cet environnement sont décrites par une succession de modifications, comme décrit en détail dans la section 3.1.1.

Agents communautés Les communautés sont représentées par des agents situés dans l'environnement. Un agent communauté est une entité autonome qui contient un certain nombre de nœuds de l'environnement. A l'origine, l'environnement ne contient aucun agent, donc aucune communauté.

Perception des agents communautés Les agents communauté ne sont capables de percevoir que les nœuds ayant au moins un lien avec un nœud qu'elles contiennent. Autrement dit, pour un agent communauté AC contenant les nœuds $k1, \ldots, kn$, l'ensemble des nœuds que AC est capable de percevoir est défini par :

$$\bigcup_{k_i \in AC} \{k_i\} \cup N(k_i)$$

La vision de l'agent est donc limitée à son environnement local, il n'a pas de connaissance globale du réseau.

Etat interne des communautés L'état interne des communautés est caractérisé par les nœuds qu'elle contient. Chaque communauté a également la possibilité d'enregistrer des informations, par exemple sur son activité passée, ou sur les propriétés des nœuds qu'elle contient (densité interne, dates de création et de modification, etc.)

Communication des agents communautés Les agents communautés ne peuvent interagir qu'avec les autres communautés avec lesquelles elles ont au moins un nœud en commun. Ceci est un critère important, puisque, dans le cas d'une communication avec tous les agents, il serait possible de retomber dans une solution relative, puisque les communautés pourraient acquérir une vision globale du graphe via les autres communautés.

Naissance des agents communautés De nouvelles communautés peuvent être créées : à chaque modification de l'environnement (donc, à chaque ajout ou suppression de lien dans le réseau), on teste l'apparition d'une communauté embryonnaire. Autrement dit, si la modification qui vient de se produire a fait apparaître un pattern donné (par exemple, une nouvelle clique), celle-ci donne naissance à une nouvelle communauté, à condition qu'elle ne soit pas complètement incluse dans une communauté existante. Cette nouvelle communauté inclue automatiquement tous les nœuds constituant le pattern lui ayant donné naissance.

Modification des communautés Chaque modification de l'environnement a lieu entre un nœud i et un nœud j. Toute communauté contenant l'un de ces deux nœuds est alors activée et va devoir prendre la décision, selon son état interne et sa vision locale du réseau, d'ajouter ou de rejeter des nœuds. Il s'agit de la partie la plus complexe, qui peut être implémentée de nombreuses manières différentes.

Disparition des communautés A chaque fois qu'une communauté prend la décision de rejeter un nœud, elle va contrôler que son état interne vérifie encore une certaine propriété. Si ce n'est pas le cas, la communauté va être considérée comme morte et disparaître de l'environnement.

Fusion de communautés A chaque fois qu'une communauté va décider d'une modification interne (ajout ou suppression de nœud), elle va avoir la possibilité d'interagir avec les communautés voisines, afin de décider s'il est devenu pertinent de fusionner avec l'une d'entre elles.

3.3.1.2 Définition algorithmique

A partir des actions identifiées précédemment, nous pouvons définir un algorithme formel qui décrit la façon dont la détection de communautés doit être effectuée. Cet algorithme est présenté dans la figure 3.6. Il est centralisé, et représente donc l'ensemble des actions effectuées par les agents. Le résultat de l'algorithme n'est pas clairement défini, car, comme cela est expliqué au chapitre 6.2.2, il y a plusieurs manières de considérer le résultat d'un algorithme de communautés dynamiques. La plus complète consiste à prendre comme résultat l'ensemble des modifications ayant eut lieu sur l'ensemble des communautés.

Les fonctions surlignées d'une couleur différente ne sont pas définies dans le méta-algorithme, et doivent être implémentées. Leur rôle est décrit dans la section suivante.

Dans cet algorithme, T_e représente la séquence des actions sur le réseau, composée de quadruplets (i, j, a, t) avec i et j les nœuds affectés, a l'action ayant lieu, pouvant être un ajout ou une suppression de lien, et t l'instant auquel cette modification à lieu. Ces actions sont ordonnées par t croissant. Dans le cas où plusieurs actions ont lieu au même instant t, un ordre aléatoire leur sera attribué. Ceci introduit un processus aléatoire qui peut avoir un effet sur le résultat final.

Le principe de l'algorithme peut être résumé de la manière suivante :

- Pour chaque ajout d'un lien (i, j) dans le réseau, si i est dans une communauté C et j n'appartient pas à C, on regarde si j doit être intégré à C *(lignes 4-18)*. On trouve ensuite l'ensemble des nouvelles communautés formées par l'apparition de ce nouveau lien. Les nouvelles communautés n'étant pas incluses dans une communauté existante sont conservées *(lignes 20-26)*.
- Pour chaque suppression d'un lien (i, j), si ce lien se trouve à l'intérieur d'une communauté $(i \in C \wedge j \in C)$, on calcule si C perd un ou plusieurs nœuds, ce qui peut l'amener à se diviser *(lignes 30-43)*.
- Après chaque modification du réseau, si une ou plusieurs communautés ont été modifiées, on regarde si elles doivent être fusionnées avec d'autres communautés. Les communautés candidates sont celles qui partagent des nœuds avec la communauté modifiée *(lignes 46-50)*.

3.3.1.3 Procédure d'implémentation

Une implémentation d'un algorithme suivant ce méta-algorithme devra donc proposer une implémentation pour les fonctions suivantes :

1. NAISSANCE : fonction définissant le pattern qui déclenche l'apparition d'une nouvelle communauté. A ce stade, la communauté est appelé communauté embryonnaire, sa forme étant simple et connue, n'ayant pas encore eut le temps d'évoluer. Cette fonction prend en paramètre les deux nœuds extrémités d'un nouveau lien, et retourne un ensemble de nouvelles communautés.
2. CROISSANCE : fonction définissant les règles qui permettent à une communauté de choisir d'intégrer de nouveaux nœuds lorsqu'une modification les affecte. Elle prend en paramètre un nœuds candidat et une communauté, et retourne soit VRAI, si le nœud doit être intégré à la communauté, soit FAUX sinon.

FONCTION $iLCD(\ T_e,\ )$:

```
0:POUR TOUT (i, j, a, t) ∈ T_e
1:     MC ← ∅  //ensemble des communautés modifiées lors de cette étape

2:      //Cas creation d'un nouveau lien
3:     SI a = creation ALORS
4:         //Croissance de communautés
5:            POUR TOUT C ∈ cs_i
6:                 SI CROISSANCE(j, C) ALORS
7:                     V_C ← V_C ∪ {j}
8:                     E_C ← E_C ∪ {{j, k} ∈ V : k ∈ V_C }
9:                     MC ← MC ∪ {C}
10:                FIN SI
11:           FIN POUR TOUT
12:           POUR TOUT C ∈ cs_j
13:                SI CROISSANCE(i, C) ALORS
14:                    V_C ← V_C ∪ {i}
15:                    E_C ← E_C ∪ {{i, k} ∈ V : k ∈ V_C }
16:                    MC ← MC ∪ {C}
17:                FIN SI
18:           FIN POUR TOUT

19:           //Naissances de communautés
20:           SI NAISSANCE(i, j) ≠ ∅ ALORS
21:                POUR TOUT C ∈ NAISSANCE(i, j)
22:                     SI C ⊈ C_x : C_x ∈ (cs_i ∪ cs_j) ALORS
23:                          P ← P ∪ {C}
24:                MC ← MC ∪ {C}
25:                FIN POUR TOUT
26:           FIN SI

27:      //Cas suppression d'un lien
28:       SINON
29:           //Contractions ou divisions de communautés
30:           POUR TOUT C ∈ (cs_i ∩ cs_j)
31:               CPRIME ← CONTRACTION_DIVISION(C, i)
32:               POUR TOUT C_2 ∈ CPRIME
33:                    SI j ∈ C_2
34:                         CPRIME ← CONTRACTION_DIVISION(C, j)
35:                    FIN SI
36:               FIN POUR TOUT
37:               P ← P \ {C} ∪ CPRIME
38:               MC ← MC ∪ CPRIME

39:               //Mort de communautés
40:               SI MORT(C) ALORS
41:                    P ← P \ {C}
42:               FIN SI
43:           FIN POUR TOUT
44:       FIN SI

45:      //Fusions de communautés
46:       POUR TOUT C ∈ MC
47:           POUR TOUT C_2 : V_C2 ∩ V_C ≠ ∅
48:               P ← P \ {C, C_2} ∪ FUSION(C, C_2 )
49:           FIN POUR TOUT
50:        FIN POUR TOUT
51:FIN POUR TOUT
```

FIGURE 3.6 – *Pseudo-code du méta-algorithme iLCD. Les fonctions surlignées correspondent à des fonctions non définies, dont l'implémentation est libre. MC et $CPRIME$ sont des variables temporaires représentant des ensembles de communautés.*

3. CONTRACTION_DIVISION : lorsqu'un lien interne à une communauté est supprimé, quelles sont les règles qui vont lui faire rejeter des nœuds (contraction) et, éventuellement, la faire se diviser en plusieurs communautés. Elle prend en paramètre une communauté, et un nœud candidat à être retiré. Elle retourne un ensemble de communautés, composé d'une seule communauté dans le cas d'une contraction simple, ou de plusieurs si une division a également eu lieu.
4. MORT : fonction définissant quelles conditions provoque la mort d'un agent communauté. Elle prend en paramètre une communauté, retourne VRAI si la communauté doit mourir, et FAUX sinon.
5. FUSION : fonction définissant quelles conditions amènent deux communautés à être fusionnées, et comment cette fusion a lieu. Elle prend en paramètre deux communautés, et retourne un ensemble de communautés, pouvant contenir une seule communauté en cas de fusion, ou deux sinon.

3.3.1.4 Note sur la complexité

Plusieurs parties de l'algorithme n'ayant pas d'implémentations, on ne peut faire de calcul de complexité exact. En revanche, on peut déjà dire que la complexité ne dépendra pas des mêmes facteurs avec cet algorithme qu'avec les algorithmes fonctionnant à l'aide d'instantanés.

Pour quantifier l'évolution d'un réseau pour une période p donnée, on peut utiliser deux valeurs :

- Le nombre de modifications unitaires m (ajout ou suppression de lien)
- Le nombre d'étapes d'évolution, e.

m dépend uniquement du réseau, qui peut être plus ou moins grand et évoluer plus ou moins rapidement. e en revanche dépend principalement de la méthode de collecte et de la façon de modéliser le graphe. Si on utilise un graphe temporel, $e = m - c$, où c représente le nombre d'opérations ayant lieu simultanément à une autre opération n'appartenant pas à c. C'est le niveau de granularité le plus fin. En revanche, en utilisant des instantanés, e correspond au nombre d'instantanés. Si le graphe évolue de manière régulière, une étape correspondra donc à $\frac{m}{e}$ modifications unitaires. En diminuant e, on rend la granularité moins fine.

Lorsqu'une méthode dispose de e instantanés pour représenter l'évolution du réseau, et considère la totalité de ces réseaux statiques à chaque étape (comme toutes les méthodes présentées dans l'état de l'art utilisant des instantanés, quelle que soit l'approche), alors sa complexité augmente lorsque e augmente, toutes choses égales par ailleurs. Par contre, sa complexité ne dépend que partiellement de m (différent selon les méthodes). La conclusion de cette remarque est que plus on affine le détail de l'évolution, plus ces méthodes seront coûteuses en temps.

Au contraire, avec la méthode présentée ici, le nombre de calculs à faire va dépendre uniquement de m. Si l'on considère un graphe en expansion par exemple, avec $m = 500$, que l'on dispose de 5 pas de temps ($e = 5$) ayant 100 nouveaux liens entre eux (granularité élevée), 50 pas de temps ($e = 50$) avec 10 modifications ou 500 pas de temps ($e = 500$) avec 1 seul nouveau lien (faible granularité), la complexité restera la même. La complexité ne dépend que du nombre de modifications m.

On pourrait cependant faire la remarque suivante : dans la méthode avec instantanés de granularité la plus fine, $m = e$, et on pourrait donc penser que la complexité est similaire. Cependant, et c'est un point très important, la totalité du réseau est représentée à chaque instantané. La quantité de données à traiter est donc égale à $e \times nm$ (donc $m \times nm$), avec nm le nombre moyen de liens du réseau. En revanche, pour notre solution, la quantité de données à traiter est égale à m. Dans un grand graphe, $m \times nm \gg m$.

Cette méthode est donc parfaitement adaptée pour étudier le détail de l'évolution des communautés, avec la granularité la plus fine.

3.3.2 Première implémentation : iLCD-CDIE, Communautés à Diffusion d'Information Efficace

Dans cette section, nous allons présenter une première implémentation d'iLCD, appelée iLCD-CDIE, pour Communautés à Diffusion d'Information Efficace. Nous allons commencer par donner une idée intuitive de la méthode, avant de définir en détail l'implémentation d'iLCD que nous proposons. Après avoir discuté de la complexité et du temps de calcul de cette méthode, nous présenterons deux expérimentations, sur des réseaux générés d'abord, puis sur le réseau de citation de JASSS. Enfin, nous conclurons, et nous expliquerons pourquoi nous avons décidé de proposer une seconde implémentation.

3.3.2.1 Idée intuitive

Nous avons expliqué dans le chapitre 3.2 la notion de communautés homogènes. Nous cherchions donc un moyen de garantir cette propriété pour nos communautés. Dans cette section, nous proposons une première implémentation d'iLCD appelée **iLCD-CDIE**, pour Communautés à Diffusion d'Information Efficace. Elle se base sur deux critères :

- EMNSN : « Estimation of the Mean Number of Second Neighbors » ou estimation du nombre moyen de voisins différents à rang deux ou moins.
- EMNRSN : « Estimation of the Mean Number of Robust Second Neighbors » ou estimation du nombre moyen de voisins différents à rang deux accessibles de manière robuste.

EMNSN va servir à s'assurer que chaque nœud de la communauté va pouvoir atteindre facilement, par une chaîne courte, tout autre nœud de la communauté. EMNSN représente également la facilité que tout nœud de la communauté peut avoir pour communiquer avec tout autre nœud de la même communauté. Concrètement, des nœuds ne seront intégrés à une communauté C que s'ils ont au moins autant de voisins à rang deux dans C que la moyenne des autres nœuds de C, et cette valeur devra rester haute. Une communauté est donc définie comme un ensemble de nœuds au sein duquel la communication peut être rapide et efficace, ce qui élimine les communautés en chaîne, en cercle (chaîne bouclante), ou avec plusieurs modules faiblement reliés.

EMNRSN vient compléter EMNSN. Elle définit la notion de voisin robuste : i est un voisin robuste de j s'il existe au moins deux chaînes différentes de longueur deux ou moins, qui mènent de i jusqu'à j. Cette métrique vient contrebalancer le rôle que peuvent jouer les hubs. En effet, si l'on envisage l'existence d'un hub connecté à la totalité des nœuds de deux communautés par ailleurs clairement séparées, on se rend compte qu'EMNSN n'est plus suffisant : les nœuds de l'une des communautés peuvent accéder aux nœuds de l'autre communauté par une chaîne de longueur deux, en passant par le hub. Mais on se rend compte que cette communication n'est pas robuste : il suffirait de supprimer le hub pour perdre toutes les possibilités de communication. En imposant que les nouveaux nœuds ajoutés au réseau soient capables d'accéder à une majorité des autres nœuds de la communauté de manière robuste, on limite ce problème. Notre définition de communauté dans cet algorithme est donc basée sur le principe de la communication interne : on souhaite que tous les nœuds d'une communauté soient capables de communiquer de manière rapide, efficace et robuste.

```
//Fonction qui retourne les communautés créées lors de l'ajout
//d'un nouveau lien dans le réseau
FONCTION NAISSANCE(i, j) :

    NOUVELLESC ← ∅
    POUR TOUT D ⊂ V : i ∈ D ∧ j ∈ D ∧ |D| = k ∧ (∀x, y ∈ D : (x, y) ∈ E)
        NOUVELLESC ← NOUVELLESC ∪ D
    FIN POUR TOUT
    RETOURNER(NOUVELLESC)
FIN FONCTION
```

FIGURE 3.7 – *Définition de la fonction NAISSANCE. k est un paramètre de l'algorithme, définissant la taille des cliques formant les communautés initiales.*

```
//Fonction qui retourne une variable booléenne
//représentant la décision d'ajouter ou non le nœud i à la communauté C

FONCTION CROISSANCE(i, C) :

    V_rang2 ← |k : k ∈ C ∧ N(k) ∩ N(i) ≠ ∅ |
    V_robustes_rang2 ← |k : k ∈ C ∧ |N(k) ∩ N(i)| ≥ 2 |

    SI V_rang2 ≥ EMNSN(C) ∧ V_robustes_rang2 ≥ EMNRSN(C) ALORS
        RETOURNER(VRAI)
    SINON
        RETOURNER(FAUX)
    FINSI
FIN FONCTION
```

FIGURE 3.8 – *Définition de la fonction CROISSANCE.*

3.3.2.2 Implémentation des différentes étapes

1. NAISSANCE
   Une communauté embryonnaire est composée d'une clique de taille k, où k est un paramètre de l'algorithme, généralement 3 ou 4. Il peut être intéressant de faire varier cette valeur selon les graphes étudiés : pour un graphe très peu dense, une clique de 3 nœuds (un triangle) peut déjà être significative, alors que pour un graphe plus dense, il vaudra mieux préférer une valeur plus élevée. En choisissant la clique comme communauté embryonnaire, on s'assure de valeurs d'EMNSN et d'EMNRSN maximales à l'origine, puisque, par définition, dans une clique, tout nœud peut accéder à tout autre de manière robuste.
   Fonction définie en 3.7.
2. CROISSANCE
   Le problème est le suivant : lorsqu'un nouveau lien apparaît entre un nœud i de la communauté C et un nœud j n'appartenant pas à C, faut-il que C intègre j ? Le critère adopté est simple : lorsqu'on veut tester si un nœud doit être intégré à une communauté, on

```
//Fonction qui retourne un ensemble de communautés,
//le résultat de la fusion entre C_new et C_old, si elle est pertinente
FONCTION FUSION(C_old, C_new) :

      SI TR(C_old, C_new) ≥ λ ALORS
           RETOURNER(C_old)
      SINON
           RETOURNER({C_old, C_new})
       FINSI
FIN FONCTION
```

FIGURE 3.9 – *Définition de la fonction FUSION.*

compte le nombre de voisins accessibles à rang 2 dans la communauté et le nombre de voisins à rang deux accessibles de manière robuste. Si ces deux valeurs sont supérieures ou égales aux valeurs d'EMNSN et d'EMNRSN de la communauté, le nœud est intégré à la communauté. Ainsi, le premier nœud qui sera intégré à une clique de taille 4 devra obligatoirement être connecté à au moins 3 nœuds de cette communauté. Au fur et à mesure des ajouts de nœuds, selon que ceux-ci auront plus de liens avec la communauté que le strict minimum, ou au contraire ne satisferont le critère que de manière minimale, les valeurs d'EMNSN et d'EMNRSN pourront rester élevées ou diminuer. Ces valeurs étant définies de manière indépendante pour chaque communauté, on pourra avoir au sein du même réseau des communautés plus ou moins denses, dans laquelle la communication sera plus ou moins facile et robuste. Les communautés s'adaptent donc à l'hétérogénéité de l'environnement (du réseau) dans lesquelles elles sont.

Fonction définie en 3.8.

3. CONTRACTION_DIVISION
   Dans cette première version, seul l'ajout de liens dans le réseau était considéré, et les communautés ne pouvaient donc qu'ajouter de nouveaux liens. Le cas contraction ou division ne peut donc pas se produire.
4. MORT
   Dans cette première solution, les réseaux et les communautés ne peuvent que croitre. Les seules disparitions de communautés possibles se font donc par fusion de communautés, comme décrit ci-après
5. FUSION
   Au fur et à mesure que les communautés grandissent, il est possible qu'elles deviennent de plus en plus similaires, c'est à dire qu'elles intègrent un grand nombre de nœuds en commun.

   Nous avons donc proposé que, lorsque deux communautés partagent un certain pourcentage de leurs nœuds, elles soient fusionnées. Dans cette première version, comme nous voulions garder une heuristique simple, la fusion de communauté est simplifiée en une suppression de l'une des communautés. Le critère d'ancienneté de la communauté est utilisé pour choisir laquelle est conservée. Deux raisons nous ont portées à faire ce choix :
   - En gardant la plus ancienne, on garde celle ayant le plus d'historique, on conserve donc plus d'informations

– Il est plus probable que la communauté la plus récente ne soit qu'un épiphénomène, une conséquence du bruit dans le réseau, tandis qu'une communauté plus ancienne, qui existe depuis plus longtemps, a une existence plus fiable. Dans la majorité des cas, la communauté la plus récente est également la plus petite.

Ce taux de recouvrement TR est calculé de la manière suivante :

$$TR(C_{old}, C_{new}) = \frac{\mid C_{old} \cap C_{new} \mid}{\mid C_{new} \mid}$$

Où C_{old} correspond à la plus ancienne communauté et C_{new} à la plus récente.

On utilise le nombre de nœuds de la communauté la plus récente comme dénominateur car, comme c'est elle qui pourra être supprimée, il est important que ce soit pour elle que le taux de recouvrement soit atteint. Nous avons d'abord cherché une valeur idéale pour le taux de recouvrement à partir duquel deux communautés doivent être fusionnées, mais des essais sur des cas réels ont montré que parfois, différentes valeurs pouvaient être pertinentes dans certains cas et pas dans d'autres, ou même que différentes valeurs sur un même réseau pouvaient donner des résultats différents mais également intéressants. Ceci est dû au fait que dans des graphes de terrain, il n'existe pas un découpage unique et clair des communautés, leurs frontières sont floues et non uniques, et dans de nombreux cas, considérer un ensemble de nœuds comme une ou plusieurs communautés est plus une question de choix que de critère absolu (communautés partiellement fusionnées, ou partageant une grande partie de leurs membres). En conséquence, le choix d'une valeur faible va augmenter le nombre de fusions de communautés, et donc diminuer le nombre de communautés dans le résultat final. On a donc une vision générale du réseau, avec des communautés peu similaires, se rapprochant d'un découpage sans recouvrement. Au contraire, plus on augmente cette valeur, et plus on va avoir de communautés proches les unes des autres. On va donc avoir un résultat plus détaillé, avec de nombreuses communautés souvent fortement enchevêtrées. Le grand nombre de communautés trouvé va cependant rendre les résultats plus difficilement exploitables. Connaître dix versions d'une communauté ne différant que par quelques nœuds n'est généralement que peu informatif. Nous définissons donc un paramètre λ compris entre 0 et 1, tel que si $TR(C_{old}, C_{new}) \geq \lambda$, on fusionne les communautés.

Cette fonction est définie en 3.9.

3.3.2.3 Optimisation du temps de calcul

Un problème qui s'est présenté avec cet algorithme a été celui du calcul pour chaque communauté d'EMNSN et d'EMNRSN. Nous avons proposé une méthode de calcul permettant d'approximer cette valeur pour une version recablée des communautés, afin d'éviter le calcul exhaustif de cette métrique à chaque étape. De manière surprenante, nous n'avons pas pu trouver dans la littérature de formule permettant d'estimer le nombre moyen de seconds voisins dans un petit réseau en tenant compte de la redondance (deux premiers voisins ayant des voisins en commun), qui est très forte dans les communautés. Nous avons donc proposé un moyen de calculer EMNSN pour un réseau quelconque :

Définissons d_C comme le degré moyen des nœuds de la communauté C

$$d_C = \left\lfloor \frac{2C^{int}}{n_C} \right\rfloor$$

Nous calculons récursivement EMNSN sur un réseau aléatoire de mêmes propriétés (nombre de liens, nombre de nœuds) que la communauté, soit $f(d_C)$ avec

$$f : [0; d_C] \to \Re$$
$$f(i) = \begin{cases} i = 0 \to d_C \\ i > 0 \to f(i-1) + (d_C - 1) * \frac{n_C - f(i-1)}{n_C} \end{cases}$$

L'idée est de compter successivement le nombre de voisins rencontrés une fois seulement à rang deux. En commençant d'un nœud quelconque ($i = 0$), nous avons en moyenne d_C voisins directs. Puis, pour chacun de ces premiers voisins, on peut ajouter en moyenne $d_C - 1$ nouveaux voisins (l'un de ses voisins étant le nœud d'origine). La probabilité pour ces voisins de ne pas avoir déjà été comptés correspond au nombre de nœuds n'ayant pas déjà été comptés comme voisins : $n_C - f(i-1)$ sur le nombre total de nœuds n_C.
L'estimation d'EMNRSN est faite selon le même principe.

3.3.2.4 Estimation de la complexité

Il est difficile de déterminer de manière réaliste la complexité de cet algorithme, car il dépend fortement de la taille et du nombre de communautés, du nombre de cliques, de la valeur des paramètres, de l'imbrication des communautés, etc. Tout d'abord, la complexité va au moins être proportionnelle au nombre de liens : à chaque nouveau lien ajouté dans le réseau, un calcul est nécessaire. D'un autre côté, ce calcul ne dépend que d'informations locales. On peut donc être certain que la complexité ne va pas croitre exponentiellement avec la taille du réseau. La complexité peut cependant être assez grande dans le cas de réseaux avec de grandes communautés et des nœuds de forts degrés. En terme de mémoire, l'algorithme ne nécessite de mémoriser que le réseau statique actuel (nœuds et liens), et les communautés actuellement vivantes, avec leur état interne. Nous avons effectué des calculs sur des graphes de différentes tailles. Sur les graphes générés par le LFR benchmark — que nous présenterons dans la suite — qui comptent jusqu'à 5000 nœuds, l'algorithme termine en quelques secondes, sur une machine de bureau ordinaire récente (Core i5, 4Go de RAM). L'exécution sur un grand réseau de terrain (le réseau de citation issu d'Arxiv, « High Energy Physic - Theory », avec les données de 1992 à 2003), composé d'approximativement 27000 nœuds et 350000 liens, nécessite moins de 10 minutes. Sur la même machine, CPM ne peut pas s'exécuter, à cause d'un dépassement mémoire (trop grand nombre de cliques).

3.3.2.5 Expérimentations

On peut considérer que cette première version de l'algorithme n'est encore que partiellement dynamique. En effet, elle n'est capable de gérer que la croissance du réseau, et l'ajout de nœuds à des communautés. Les phénomènes de disparition de liens, nœuds et communautés ne sont pas gérés. Nous avons donc testé cet algorithme sur des réseaux adaptés à ces contraintes. Parmi ceux-ci, les réseaux de citations sont particulièrement intéressants. Nous allons commencer par présenter le type de résultats que l'on peut obtenir en étudiant un réseau de citation sans traitement, en utilisant le réseau du journal JASSS. Dans un deuxième temps, nous allons montrer les résultats obtenus sur des réseaux générés à l'aide du Benchmark LFR [LFR08]. Ces résultats, bien que ne portant pas sur des réseaux dynamiques, permettent de montrer que les communautés trouvées par un algorithme de ce type sont cohérentes avec celles que l'on peut trouver avec un algorithme plus classique.

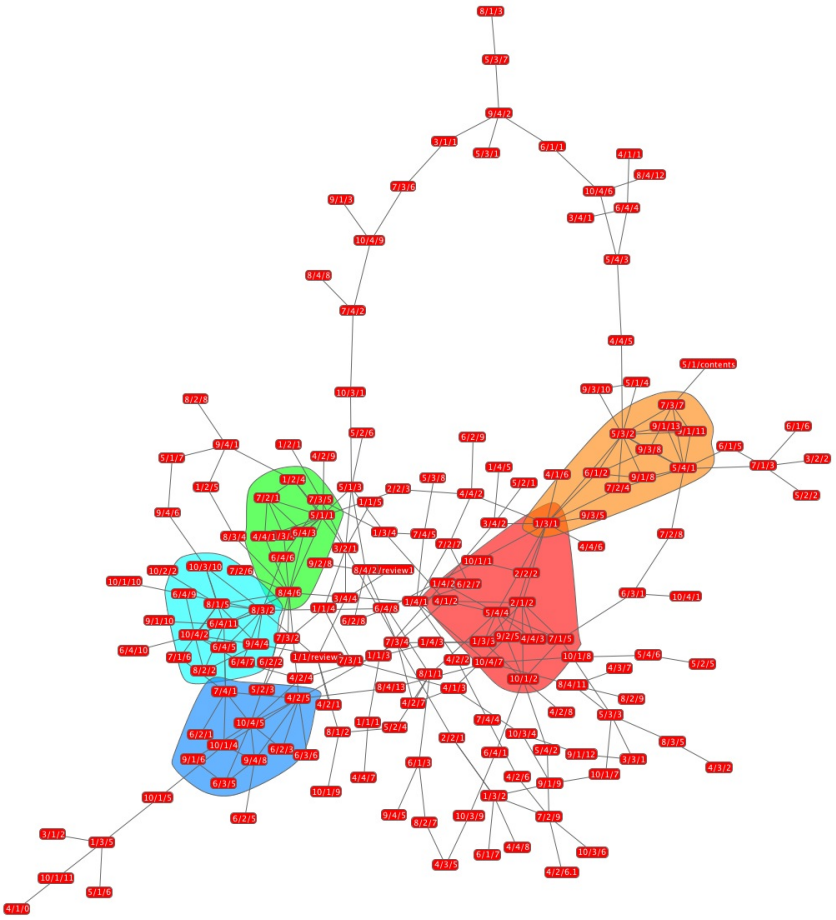


FIGURE 3.10 – *Visualisation des communautés dans le réseau de citation de JASSS, avec la première version d'iLCD*

Expérimentations sur le réseau de citation de JASSS Le « Journal of Artificial Societies and Social Simulation », JASSS, avait déjà été le sujet d'une étude sur l'évolution de ses communautés par Meyer et Lorscheid [MLT09].

Dans cet article, les auteurs avaient créé deux versions statiques du réseau, en agrégeant les données de co-citations de deux périodes différentes, détecté les communautés dans chacun de ces réseaux agrégés, et ensuite comparé manuellement les communautés, de manière à faire apparaître les changements dans les principaux domaines de recherche publiés dans JASSS. Nous avions donc décidé d'utiliser notre propre méthode sur les mêmes données, de manière à pouvoir comparer qualitativement nos résultats à ceux d'un travail précédent.

Le jeu de données utilisé est constitué de tous les articles publiés dans ce journal entre 1998 (date de la création du journal) et 2008, citant au moins un autre article paru dans JASSS. Les nœuds représentent donc des articles et les liens des citations entre ces articles. Nous avons considéré les liens comme non dirigés, l'orientation des liens ne jouant pas de rôle dans notre algorithme. Comme on peut le voir sur la figure 3.10, le réseau est peu dense, avec beaucoup de nœuds n'ayant qu'un ou deux liens avec les autres articles de JASSS. Le but d'une détection de communautés sur un tel réseau est d'identifier les sujets les plus importants, et donc traités par plusieurs articles. Les algorithmes de détection de communautés sans recouvrement ne sont

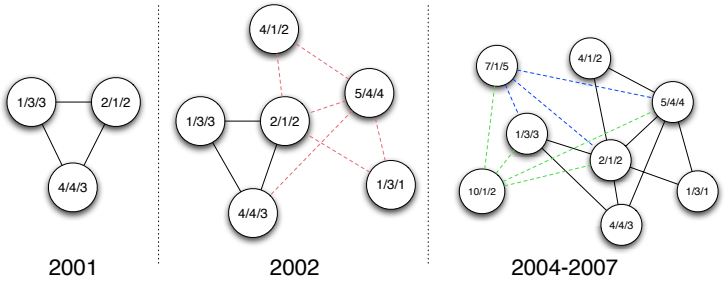


FIGURE 3.11 – *Evolution de la communauté "norme et réputation" de JASSS*

pour la plupart pas adaptées à cette tâche : ils essayent de faire une partition totale du réseau, alors qu'ici il ne semble pas y avoir de communauté claire pour tous les nœuds. L'avantage des communautés intrinsèque dans ce cas est que ne sont détectées comme communautés que les structures présentant effectivement une structure particulière.

Sur la figure 3.10, nous avons également représenté les communautés détectées par notre algorithme lors de la dernière étape d'évolution du réseau. L'algorithme détecte 5 communautés.

En recoupant simplement les mots clefs de ces publications, on peut établir aisément à quoi elles correspondent : les 3 communautés sur la gauche sont directement liées aux systèmes multi-agents (SMA) : SMA et industrie, réplications de SMA, SMA et jeux de rôles. Les deux sur la droite concernent les normes et réputations et les dynamiques d'opinion.

Même si la méthode et le réseau étudiés étaient fortement différents de l'article original de Meyer et Lorscheid (qui étudiaient le réseau de co-citations), les communautés détectées sont assez similaires. Elles semblent de plus cohérentes avec les connaissances que l'on a sur le réseau.

Par rapport aux approches traditionnelles, l'ajout principal est cependant dans le côté dynamique de la détection. Là où l'approche par instantané se contentait de constater les différences entre la première moitié de l'histoire du journal et la deuxième moitié, avec notre algorithme, nous sommes capable d'avoir beaucoup plus d'information. Nous pouvons ainsi dire que la communauté normes et réputations est apparue en 2001, alors que la communauté sur la réplication de modèles n'est apparue qu'en 2006. En observant l'évolution de la communauté normes et réputations, nous sommes capables de dire que si elle n'était composée que de trois nœuds en 2001, elle en comptait 6 en 2002, 7 en 2004 et 8 en 2007, et ainsi caractériser la croissance de certaines communautés, comme montré dans la figure 3.11.

Validation sur les réseaux générés avec le Benchmark LFR Dans le chapitre précédent, nous avons montré que notre solution donnait des résultats intéressants sur un réseau de terrain. Cependant, le fait qu'un algorithme soit capable de retrouver les sujets principaux dans un réseau de citation n'en fait pas forcément un bon algorithme de détection de communautés. La méthode qui nous a paru la plus pertinente pour juger de la qualité de l'algorithme a été de la comparer aux autres méthodes existantes, en utilisant une procédure standard. L'article de Lancichinetti [LF09] a rencontré un grand succès dans le domaine de l'évaluation des algorithmes de détection de communautés. Le principe de l'article est assez classique : des réseaux aléatoires sont générés, contenant une structure en communauté clairement définie. N'importe quel algorithme peut ensuite être utilisé pour, à partir de ces réseaux, essayer de retrouver les communautés. La différence entre le résultat correct — connu par construction — et le résultat obtenu par l'algorithme est calculée via une métrique, l'IMN (Information Mutuelle Normalisée). [DDGDA05]

Le succès de l'article est principalement dû à deux raisons : d'une part, l'utilisation d'un benchmark novateur pour créer des graphes aléatoires avec structure en communauté. En effet, les générateurs utilisés précédemment dans des articles similaires [CK01, GN02, FLZ+07] se contentaient de petits réseaux peu réalistes. Avec le générateur proposé par Lancichinetti et al [LFR08], le Benchmark LFR, plusieurs paramètres du réseau peuvent facilement être modifiés, tels que la taille des communautés, la densité du réseau, la netteté de la séparation des communautés, etc.

L'autre point intéressant de l'article est qu'il compare un assez grand nombre d'algorithmes et fait ressortir de fortes différences d'efficacité entre eux.

Nous avons donc repris exactement la même procédure que celle utilisée dans l'article, et avons appliqué notre propre algorithme sur les réseaux à tester.

Avant de présenter les résultats, il est nécessaire d'expliquer comment notre algorithme, qui effectue une détection de communautés dynamiques sur un réseau temporel, a pu être comparé avec des résultats d'algorithmes statiques sur des réseaux statiques. Pour ce faire, nous avons tout simplement attribué un ordre aléatoire aux liens du réseau. Il est bien entendu que cela peut poser des problèmes à notre algorithme, et ne peut en tout cas pas l'avantager. Dans un réseau dynamique de terrain, il est évident que les communautés ne se forment pas dans un ordre quelconque, elles commencent généralement par une petite structure, et grossissent ensuite progressivement, en intégrant de nouveaux nœuds. En prenant les liens dans un ordre aléatoire, cette croissance réaliste n'est plus respectée, les communautés se forment de toute part simultanément.

Les résultats donnés ici sont donc à prendre avec les précautions nécessaires. Ils sont cependant assez concluant pour dire que cette méthode, bien que basée sur des communautés intrinsèques et d'une nature assez différente des méthodes précédentes, trouve des communautés qui peuvent être comparées avec celles des algorithmes classiques.

Un premier test a été effectué sur des graphes sans recouvrement. Le benchmark LFR permet de faire varier de nombreux paramètres. Pour les paramètres invariants, nous avons repris les mêmes valeurs que dans l'article original[LF09]. Ces paramètres définissent notamment la densité du réseau (25 liens par nœud), la loi de puissance de distribution des degrés, ainsi que la loi de puissance de distribution de la taille des communautés. En revanche, trois paramètres étaient utilisés dans l'article original pour tester l'efficacité des algorithmes dans des cas différents :

- La taille du réseau, qui peut être petit (1000 nœuds) ou grand (5000 nœuds).
- La taille des communautés, là aussi petites ou grandes (respectivement entre 10 et 50 nœuds ou entre 20 et 100).
- le coefficient de mélange(*mixing parameter*) μ, qui représente le ratio, pour chaque nœud, entre les liens internes à sa communauté et les liens externes. Plus cette valeur est basse, moins il y a de liens inter-communautés, et plus les communautés sont clairement définies.

La figure 3.12 compare notre algorithme avec l'algorithme CPM, qui était le seul à être capable de gérer le recouvrement dans l'article original (car c'était également le seul dont le code source était disponible). Comme on peut le voir, les résultats sont globalement comparables, avec un léger avantage à notre algorithme. On note que l'algorithme est plus efficace sur les petites communautés que sur les grandes. Ceci s'explique facilement : la densité restant la même (25 liens par nœuds), lorsque les communauté sont grandes et que μ commence à être élevé, les communautés deviennent très peu denses et ne sont donc logiquement plus considérées comme des communautés par notre algorithme. Par exemple avec 100 nœuds par communauté et un μ de 0,5, chaque nœud n'est connecté en moyenne qu'à 13% des nœuds de sa communauté. On peut se représenter une telle communauté sur la figure 3.13.

La figure 3.14 compare notre algorithme avec les méthodes les plus efficaces présentées dans

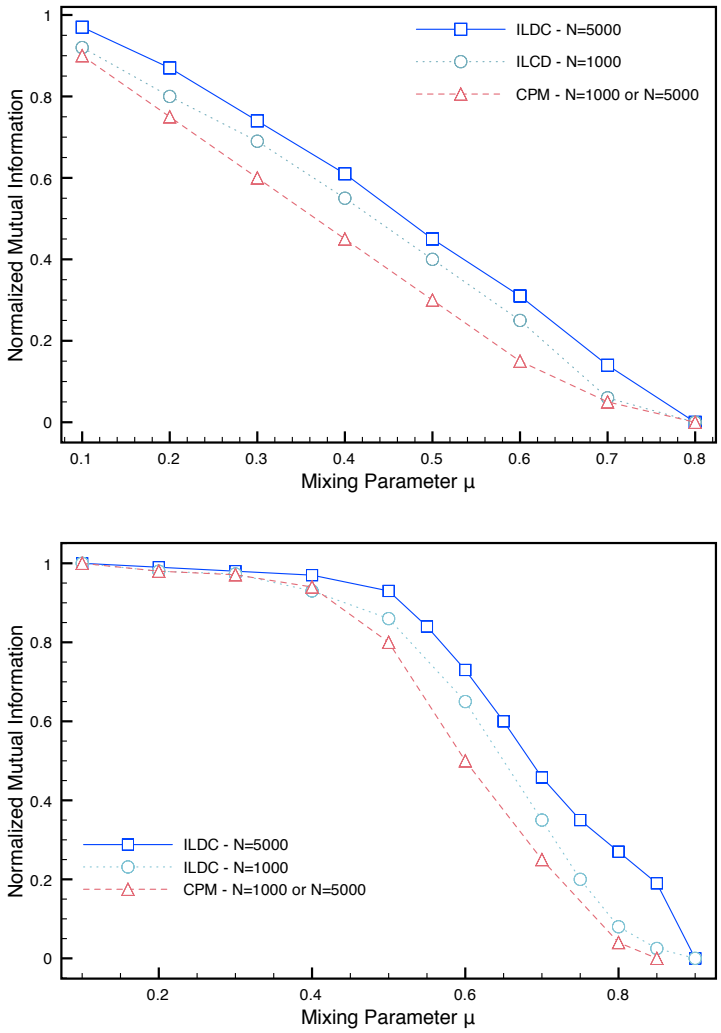


FIGURE 3.12 – *Comparaison entre iLCD et CPM sur le benchmark LFR. En haut, avec de grandes communautés. En bas, avec de petites communautés*

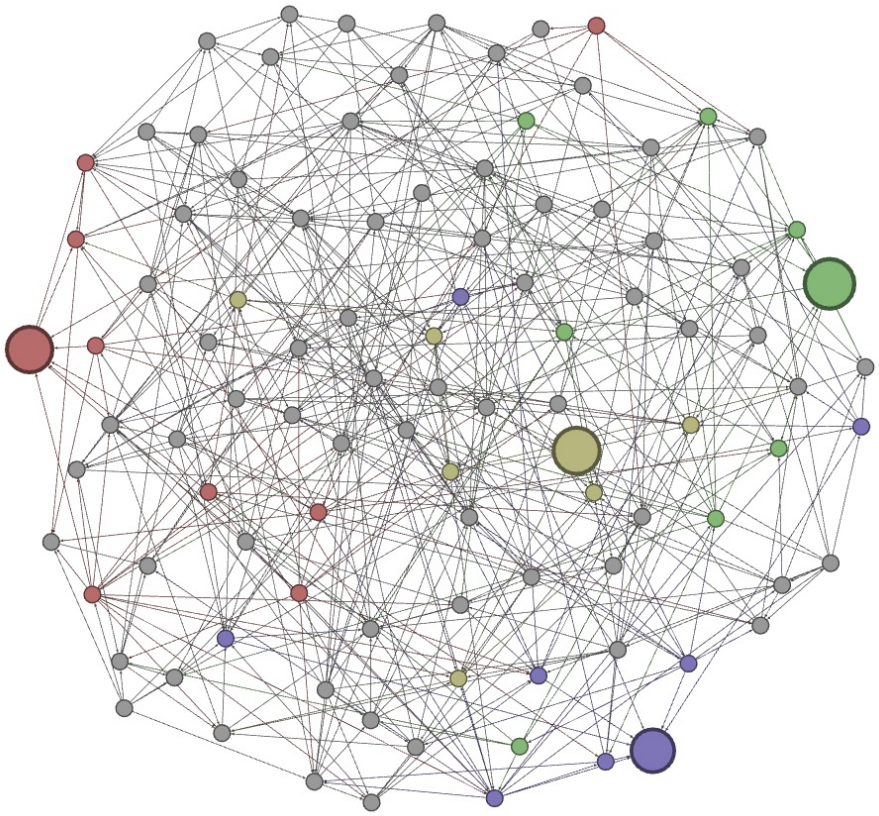


FIGURE 3.13 – *Communauté générée par le LFR benchmark, contenant 100 nœuds et pour une valeur de $\mu = 0,5$. Quatre nœuds sont représentés de plus grande taille. Ces nœuds et leurs voisins ont été colorés, afin de montrer que certains nœuds ne sont connectés qu'à une petite partie de la communauté. On remarque qu'aucun de ces quatre nœuds n'a de voisins communs avec les autres. Le diamètre de ce graphe est de 6.*

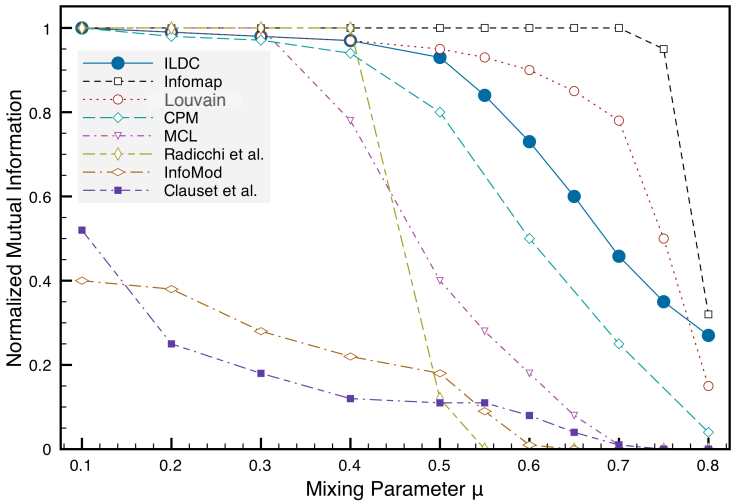


FIGURE 3.14 – *Comparaison de plusieurs algorithmes de détection de communauté sur le même graphe généré par LFR.*

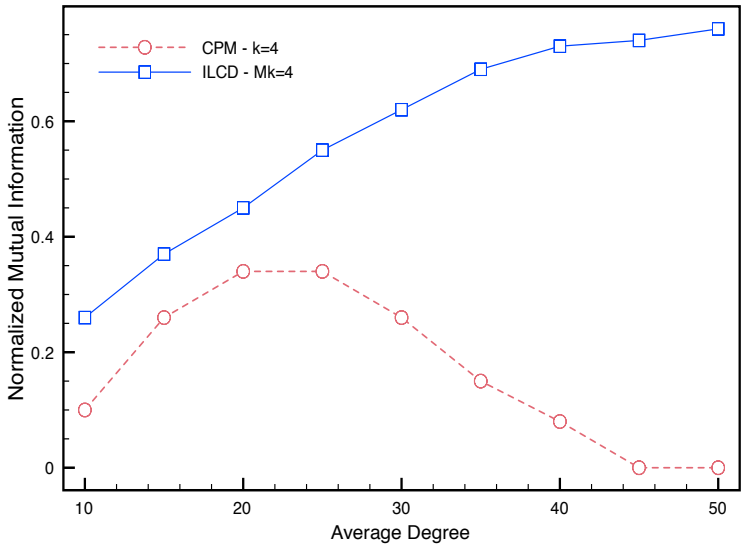


FIGURE 3.15 – *Comparaison entre iLCD et CPM sur le benchmark LFR avec recouvrement. En faisant varier la densité du graphe, on voit clairement les résultats de CPM s'effondrer.*

l'article original. Ces algorithmes sont : InfoMap [RB08], Louvain [BGLL08], CPM [PDFV05], MCL [EVDO02], Radicchi [RCC+04], InfoMod [RB07], Clauset et al. [CNM04] Se rapporter à l'article original pour une description de ces algorithmes.

Seul le cas où les communautés sont le plus clairement définies est représenté : ce sont de petites communautés (taille entre 10 et 50 nœuds) dans de grands réseaux. On constate cette fois que les meilleurs algorithmes statiques sont largement plus efficaces que notre algorithme sur ce cas. En revanche, on peut également se rendre compte que plusieurs algorithmes connus donnent de moins bons résultats, notamment des algorithmes basés sur l'optimisation de la modularité (Clauset et al., Radicchi et al.).

Ceci est donc une confirmation que notre algorithme permet bien une détection de communautés qui, quoique différente des approches classiques, permet bien de reconnaître des structures de type communautés selon la définition habituelle.

Dans une dernière expérience utilisant le Benchmark LFR, nous nous sommes intéressés au problème de la détection de communautés avec recouvrement. Nous avons choisi une situation de recouvrement très important : dans les réseaux générés, chaque nœud devait appartenir à exactement trois communautés différentes. Pour ne pas rendre le problème trop complexe, les nœuds n'avaient pas de liens en-dehors de ces communautés (μ=0). Cependant, les communautés sont difficiles à détecter : chaque nœud a, par rapport à une communauté, plus de liens à l'extérieur de celle-ci qu'à l'intérieur. Les communautés restent cependant assez claires pour être détectées (sans recouvrement, cela correspond à une valeur de μ de 0,66).

Cette fois, le paramètre que nous faisons varier est la densité moyenne du graphe.

Sur la figure 3.15, nous pouvons observer que plus le réseau est dense, et plus notre algorithme est efficace. Pour CPM en revanche, la tendance est inverse. Si, pour des valeurs de densités inférieures à 20 liens par nœuds, on constate bien cette tendance, en revanche, pour les valeurs supérieures à 25, l'efficacité s'effondre rapidement. Ceci est tout simplement dû au mécanisme de CPM, uniquement basé sur des cliques. Passée une certaine densité, la probabilité de rencontrer des cliques devient importante y compris entre les communautés, et des communautés séparées sont fusionnées à tort. Il est certes possible de faire varier la taille des cliques de base, le paramètre

k, pour CPM. Avec un graphe plus dense, une valeur de cliques plus élevée donne de meilleurs résultats. Cependant, trouver la bonne valeur de k sur un graphe de terrain pour lequel on ne connaît pas à l'avance les communautés est un problème complexe. Et surtout, le problème devient insurmontable dès lors que l'on a affaire à des graphes avec une forte hétérogénéité : si une partie du graphe est très dense et l'autre partie beaucoup moins, choisir une valeur ou une autre reviendra à mal détecter les communautés dans une partie du graphe.

Notre algorithme au contraire, ayant une définition de communauté s'adaptant à chaque instance de communauté, peut détecter efficacement à la fois des communautés de densité moyenne et des communautés de densité forte dans le même réseau. On pourra cependant remarquer que notre algorithme, pas plus que CPM, n'est efficace sur les communautés très peu denses. Cela n'est pas surprenant, ces communautés ne vérifiant ni les critères de forte transitivité interne (formation de cliques), ni le critère de connexion robuste entre les nœuds utilisés comme définition de communauté par notre algorithme.

3.3.2.6 Bilan sur la première implémentation d'iLCD

Cette première approche était très intéressante parce qu'elle validait le principe du méta-algorithme de détection de communautés intrinsèques longitudinales. D'une part, les tests sur des réseaux de terrain et sur des réseaux générés ont montré que l'on pouvait obtenir des communautés sémantiquement valables, et d'une nature proche des algorithmes classiques. D'autre part, les tests sur des réseaux réellement dynamiques (réseaux de citations), ont mis en évidence l'intérêt certain d'une telle approche dynamique : la possibilité de suivre, sur des réseaux évoluant fortement, la naissance de nouvelles communautés et l'évolution de communautés existantes, sans se limiter à quelques instantanés du réseau.

Enfin, les communautés trouvées par cette implémentation correspondent aux caractéristiques des communautés recherchées :

- Les communautés sont denses, puisqu'elles sont initialement constituées de cliques et ne peuvent s'agrandir qu'en ajoutant des nœuds ayant de nombreuses connexions avec les nœuds de la communauté, comme imposé par EMNSN et EMNRSN.
- Les communautés sont nettement séparées du reste du réseau, puisque tout nœud ayant de nombreux liens avec elle est intégré, selon EMNSN et EMNRSN.
- Les communautés sont intrinsèques, puisque l'ensemble du réseau n'est pas pris en compte, les ajouts ne se font qu'en considérant la frontière de la communauté.
- Les communautés sont homogènes puisque, par construction, il faut que les nœuds aient tous un accès important aux autres nœuds de la communauté. Il n'est donc pas possible de créer des cercles de nœuds, des lignes de nœuds, ou des composantes clairement séparées, puisque cela créerait des communautés dans lesquelles l'information ne pourrait pas se diffuser efficacement, caractérisées par de faibles valeurs de EMNSN et EMNRSN.

Cependant, si, sur le principe, les résultats étaient satisfaisants, dans le détail, cette version souffrait d'un certain nombre de faiblesses. Certaines étaient connues dès le départ, d'autres se sont révélées en travaillant sur des graphes de terrain de plus grande taille ou de plus grande complexité.

Pas de gestion de la disparition de liens, de nœuds et de communautés Cette faiblesse est la plus évidente. Il était volontaire d'avoir laissé, dans un premier temps, ce problème de côté, pour se concentrer sur la réalisation d'une première version simplifiée, permettant de valider le principe de fonctionnement. Il aurait été possible de garder le même mécanisme utilisant EMNSN et EMNRSN, et de rajouter la gestion des disparitions de liens. Cependant, cela posait

certains problèmes, notamment pour assurer la conservation de valeurs élevées de ces deux métriques au sein des communautés. Comme ces métriques présentaient par ailleurs d'autres faiblesses, que nous allons expliquer en-dessous, nous avons préféré nous orienter sur une nouvelle implémentation utilisant un principe légèrement différent, et n'avons donc finalement jamais implémenté la disparition de liens dans cette version.

EMNRSN et hubs multiples Comme expliqué dans le chapitre 3.3.2.1, le rôle de la métrique EMNRSN était d'empêcher de considérer deux communautés différentes comme similaires. En effet, lorsque deux communautés séparées contiennent un hub commun, la plus courte chaîne moyenne entre les nœuds de ces différentes communautés devient très courte. La plupart des nœuds se trouvent alors à distance deux les uns des autres, via le hub. Le rôle d'EMNRSN était de réduire ce problème en annulant le rôle de ces hubs. Cependant, il suffit que les deux communautés aient plusieurs hubs en commun, et l'on retombe exactement sur le même problème. Dans un premier temps, nous avions pensé que ce problème était mineur et ne se présenterait que peu dans des graphes de terrain. Effectivement, sur les graphes générés et les réseaux de citations étudiés, les résultats ne semblaient pas en pâtir particulièrement. En revanche, lorsque nous avons commencé à étudier des réseaux de terrain de plus grande taille, nous nous sommes rendu compte du rôle prédominent que pouvaient parfois prendre les hubs, et nous sommes effectivement retrouvé confrontés aux problèmes de « hubs géants » appartenant à une proportion importante des nœuds du réseau. De nombreuses communautés partageaient alors les mêmes hubs, et l'algorithme peinait à les séparer. Nous avons donc pris soin de garder ce problème à l'esprit en concevant une deuxième implémentation de l'algorithme.

Fusion de communautés trop naïve Dans cette version, lorsque deux communautés deviennent trop semblables, l'une d'entre elles est tout simplement supprimée. Les autres solutions simples, telles qu'intégrer tous les nœuds de l'une des communautés dans l'autre, ou au contraire ne retenir que les nœuds partagés entre les deux, posaient encore plus de problèmes, surtout dans le cas où les communautés fusionnées sont encore assez différentes (rappelons que ce critère est contrôlé par un paramètre de l'algorithme). La communauté résultante peut, dans ce cas, se retrouver dans un état « incohérent », ne respectant plus les critères de forte EMNSN et EMNRSN.

Cependant la solution choisie, si elle garantit que la communauté résultant de la « fusion » forme toujours un tout cohérent, est souvent la cause de perte d'information.

C'est en voulant étudier plus en détail la dynamique des communautés, en essayant de décrire l'évolution de chaque communauté particulière, que nous nous sommes rendu compte que parfois cette fusion était fortement porteuse de sens, et qu'il était nécessaire de développer une méthode adéquate pour la fusion de communautés, adaptée à l'analyse en temps réel (le problème est fort différent lorsque l'on travaille avec des instantanés, car il n'y a pas cette fusion progressive, qui oblige à prendre la décision, à un moment donné, de fusionner deux communautés qui ne sont presque jamais rigoureusement identiques.)

Complexité En théorie, la complexité d'un algorithme basé sur iLCD est relativement faible, puisque seul un calcul local est à effectuer à chaque modification du réseau. Cependant, dans les faits, cette complexité dépend beaucoup du réseau étudié. Avec iLCD-CDIE, sur des réseaux de terrain de grande taille, le temps de calcul devenait prohibitif. Une première raison est due au calcul d'EMNSN et d'EMNRSN, qui peuvent être coûteux à calculer sur de grosses communautés. L'autre situation ralentissant fortement l'algorithme était la présence de nombreux hubs de très forts degrés. Ces hubs appartiennent à beaucoup de communautés, créent beaucoup de liens et,

en conséquence, amènent à faire à chaque étape de nombreux calculs sur de très nombreuses communautés.

Imaginons un hub appartenant à toutes les communautés d'un grand réseau. A chaque fois qu'un lien est créé entre un nœud quelconque et ce hub, ce qui est fréquent étant donné que ce hub a un degré très élevé, toutes les communautés du réseau vont donc être « réveillées » par cette opération et vont calculer si elles doivent ou non intégrer le nœud auquel le hub vient d'être connecté. Paradoxalement, iLCD-CDIE était donc très rapide sur les réseaux générés et certains graphes de terrain ayant une forte densité, du moment qu'ils ne contenaient pas de hubs d'importance, alors qu'il pouvait se révéler beaucoup plus lent sur un graphe de terrain de densité inférieure. Cette limite a aussi été prise en compte dans la seconde implémentation, puisque l'un des objectifs de ce travail était de pouvoir traiter des réseaux de terrain de très grande taille, quelles que soient leurs propriétés.

3.3.3 Deuxième implémentation : iLCD-NRMH, Nœuds Représentatifs et Minimisation des Hubs

Dans cette section, nous allons présenter la deuxième implémentation d'iLCD, appelée iLCD-NRMH, pour Nœuds Représentatifs et Minimisation des Hubs. Nous allons commencer par présenter l'idée intuitive du fonctionnement de cette implémentation, qui sera suivie de la présentation détaillée des métriques utilisées. L'étape suivante consistera à décrire en détail l'implémentation d'iLCD, qui utilise ces métriques. Nous discuterons ensuite de la complexité de cette méthode, ainsi que sur ses limites. Enfin, avant de conclure, nous présenterons une variante du calcul de la métrique de représentativité.

3.3.3.1 Idée intuitive

La plupart des méthodes classiques, pour évaluer la pertinence de l'ajout d'un nœud à une communauté, se basent simplement sur le nombre de liens que ce nœud entretient avec la communauté, sans se soucier des propriétés des nœuds auxquels ces liens le connectent. Ainsi, avec la modularité par exemple, si un nœud a plus de liens avec une communauté qu'il n'en aurait dans un réseau aléatoire, l'ajouter à la communauté va avoir tendance à augmenter la modularité, au moins de cette communauté, et va donc être considéré comme une action intéressante.

Dans la méthode avec recouvrement OSLOM, cela est encore plus explicite : lorsque l'on souhaite savoir si l'on doit ajouter un nœud à une communauté, on évalue le nombre de liens qu'il devrait avoir dans un réseau aléatoire, et la comparaison avec le nombre de liens effectivement présents va décider de l'ajout ou non du nœud à la communauté.

La plupart des méthodes vont avoir une démarche similaire, ne s'intéressant qu'au nombre de liens internes, sans se soucier des nœuds auxquels ces liens correspondent. Or, on peut facilement montrer que cette vision est un peu réductrice dès que l'on considère que tous les nœuds n'appartiennent pas à leurs communautés de la même manière.

Prenons l'exemple d'une communauté C composée de 10 nœuds. 6 d'entre eux ne sont reliés qu'à des nœuds de C, les 4 autres sont des hubs du réseau, fortement connectés aux nœuds de C, mais tout aussi fortement connectés à plusieurs autres communautés. Admettons maintenant qu'un nœud, non encore inclus dans C, est relié à 4 nœuds de C, et que le fait d'avoir 4 liens avec C soit considéré comme significatif (supérieur au nombre de liens moyens théoriquement présents, par exemple.) (Illustration 3.16).

On peut raisonnablement admettre que si les 4 nœuds auxquels le lien est connecté sont les 4 hubs, alors la probabilité pour que le nœud appartienne effectivement à la communauté C est

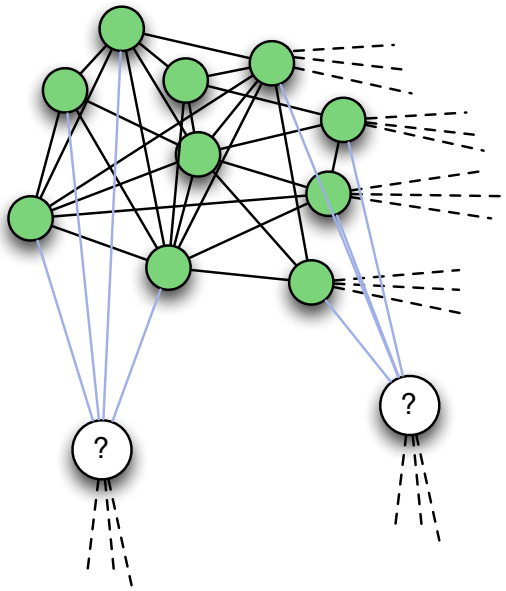


FIGURE 3.16 – *Illustration de l'importance des propriétés des nœuds auxquels un nœud candidat est connecté. Les deux nœuds marqués d'un point d'interrogation pourraient être ajoutés à la communauté. Cependant, celui de droite est connecté seulement à des hubs, alors que celui de gauche n'est connecté qu'à des nœuds n'appartenant qu'à cette seule communauté. Les mesures classiques telles que la modularité ou la densité interne ne prennent pas cette différence en compte.*

plus faible que s'il est connecté à 3 nœuds qui n'appartiennent qu'à cette communauté. C'est sur cette idée qu'est basée cette version de l'algorithme. A chaque nœud et pour chacune de ses communautés va être attribuée une valeur de représentativité, décrivant à quel point ce nœud est représentatif de cette communauté. Dès lors, lorsque l'on voudra ajouter un nœud à une communauté, on va se baser sur la représentativité des nœuds auxquels il est connecté au sein de la communauté.

Une étude empirique, menée par Backstrom et al. [BHKL06], vient donner un argument empirique. Les auteurs ont étudié plusieurs réseaux réels, venant du web 2.0, dans lesquels il existait une structure en communautés explicite. En connaissant le réseau et les communautés à un moment donné, ils ont essayé de trouver quelles étaient les propriétés qui permettaient de prédire si un nœud n allait être intégré à une communauté ou non. Le nombre de liens avec la communauté est un premier indicateur. Mais les auteurs ont montré que cet indicateur était à pondérer : si les nœuds internes à la communauté à laquelle n est lié sont connectés entre eux, la probabilité pour n de se joindre à la communauté est beaucoup plus élevée que s'ils ne le sont pas. Compter simplement le nombre de liens externes et de liens internes n'est donc pas suffisant, il faut prendre en compte la topologie.

De la même manière, on sait que la définition classique des communautés édicte que celles-ci doivent être bien séparées du reste du réseau. Généralement, les algorithmes essayent donc d'avoir un minimum de liens entre les communautés. Nous avons vu dans l'état de l'art que beaucoup de méthodes avec recouvrement font appel à la densité interne pour trouver des communautés pertinentes, définir comme le ratio entre le nombre de liens internes de la communauté et la somme des degrés des nœuds de la communauté (donc, liens internes + liens externes). Mais là encore, en considérant que tous les nœuds n'appartiennent pas de la même façon à leurs

communautés, cette vision peut être trompeuse. Prenons deux communautés de 10 nœuds, $C1$ et $C2$. Dans $C1$, tous les nœuds ont 5 liens internes et 5 liens externes. On a donc 50 liens externes. Dans $C2$, les nœuds ont également tous 5 liens internes. En revanche, 9 d'entre eux n'ont aucun lien externe. Le dernier est par contre un hub conséquent du réseau, et possède 50 liens vers d'autres communautés.

On voit ici que, malgré le fait que les communautés aient rigoureusement le même ratio entre liens externes et liens internes, la façon dont elles sont séparées du reste du réseau est très différente. Ce sont ces observations qui ont guidé la conception de cette seconde version de l'algorithme, iLCD-NRMH, pour **Nœuds Représentatifs et Minimisation des Hubs**.

3.3.3.2 Introduction des métriques

Nous allons donc définir des métriques, qui seront utilisées par l'algorithme, et qui nous permettrons de prendre en compte la topologie des communautés.

Représentativité Nous commençons par introduire la représentativité, une métrique qui indique à quel point un nœud est représentatif d'une communauté.

On définit la représentativité Rp d'un nœud i par rapport à une communauté C comme étant la métrique suivante :

$$Rp(i, C) = \frac{d_C^{int}(i)}{d(i)}$$

Logiquement, les nœuds possédant de nombreux liens externes, tels que les hubs, vont avoir une valeur de représentativité faible, tandis que ceux n'étant liés qu'à une communauté auront une valeur de représentativité maximale, de 1.

Cohésion intrinsèque Une métrique est utilisée pour calculer la cohésion intrinsèque CI d'une communauté, elle est fortement liée à la séparation de la communauté du reste du réseau. Elle est définie comme étant la somme des représentativités des nœuds qui la composent :

$$CI(C) = \sum_{i \in C} Rp(i, C)$$

En conséquence, si la communauté est principalement composée de hubs, sa valeur de cohésion intrinsèque va être très faible. Au contraire, si les nœuds ont chacun peu de liens externes, la valeur va être élevée, jusqu'à atteindre n_C pour une composante connexe du réseau. Pour deux communautés ayant le même nombre de liens externes, également répartis pour l'une et concentrés sur un nœud pour l'autre, la valeur de cohésion intrinsèque sera largement plus élevée pour celle où un seul nœud concentrera tous les liens externes.

Force d'appartenance Cette métrique quantifie, pour un nœud i et une communauté C donnés, la force de l'appartenance FA de i à C. Elle est définie comme la somme des valeurs de représentativité pour C des voisins de i appartenant à C :

$$FA(i, C) = \sum_{j \in N_C(i)} Rp(j, C)$$

On peut noter que cette métrique est une somme, et pas une moyenne. Elle n'est donc pas bornée entre 0 et 1, comme l'était Rp. S'il s'agissait d'une moyenne, un nœud n connecté à un seul nœud

$n2$ de C pourrait avoir une valeur de FA maximale si $n2$ avait une représentativité maximale. Ici au contraire, pour avoir une FA maximale, n doit être connecté à tous les nœuds de C. La valeur de FA représente donc bien l'attachement de n à C : plus il est connecté à de nombreux nœuds, plus la valeur sera élevée. Plus les nœuds auxquels il est connecté sont représentatifs de C, et plus la valeur de FA sera également élevée.

3.3.3.3 Implémentation des différentes étapes

```
//Fonction qui décide d'ajouter ou non le nœud i à la communauté C
FONCTION CROISSANCE(i, C) :

      SI  FA(i, C) ≥ Δ * CI(C) ALORS
          RETOURNER(VRAI)
      SINON
          RETOURNER(FAUX)
       FINSI
FIN FONCTION
```

FIGURE 3.17 – *Définition de la fonction CROISSANCE.*

```
//Fonction qui détermine si le nœud i doit rester dans C,
//et retourne la ou les communautés résultant de ce retrait le cas échéant.

FONCTION CONTRACTION_DIVISION(C, i) :
      SI  FA(i, C) < Δ * CI(C \ i) ALORS
          V_C ← V_C \ i
          E_C ← E(V_C)
          POUR TOUT k ∈ N_C(i)
              C ← CONTRACTION_DIVISION(C, k)
          FIN POUR TOUT
     FIN SI
     RETOURNER({C})
FIN FONCTION
```

FIGURE 3.18 – *Définition de la fonction CONTRACTION_DIVISION.*

1. NAISSANCE :
   Comme dans la version précédente, les communautés embryonnaires sont des cliques d'une taille donnée, de préférence 3 ou 4. Cela permet de s'assurer qu'à l'origine, la communauté est cohérente dans sa structure interne (chaque nœud est connecté au maximum possible d'autres nœuds de la communauté). Au niveau de sa séparation du reste du réseau en revanche, tout est possible, certains nœuds peuvent être des hubs et d'autres non.
   La définition de cette fonction avait déjà été faite pour la première implémentation : 3.7

```
//Fonction qui détermine si la communauté C doit disparaître.
FONCTION MORT(C) :

      SI n_C > k ALORS
           RETOURNER(Faux)
      SINON
           RETOURNER(Vrai)
      FINSI
FIN FONCTION
```

FIGURE 3.19 – *Définition de la fonction MORT.*

```
//Fonction qui retourne le résultat de la fusion entre C_new et C_old,
//si elle est pertinente
FONCTION FUSION(C_old, C_new) :

      SI λ * CI(C_new) < Σ_{i|i ∈ (C_new ∩ C_old)} Rp(i, C_new) ALORS

           V_{C_old} ← V_{C_old} ∪ (n_2|n_2 ∈ C_new , FA(n_2, C_old) > Δ * CI(C_old))
           E_{C_old} ← E(V_{C_old})
           RETOURNER( {C_old} )
      SINON
           RETOURNER({C_old, C_new} )
      FINSI
FIN FONCTION
```

FIGURE 3.20 – *Définition de la fonction FUSION.*

2. CROISSANCE :
   Lors de l'ajout d'un nouveau lien, comme expliqué dans le méta-algorithme, certaines communautés peuvent chercher à intégrer l'un des nœuds affectés par le lien. Lorsqu'une communauté C cherche à intégrer un nœud i, on va comparer les deux grandeurs suivantes : d'une part, la force d'appartenance de i à C, $FA(i, C)$. Cette valeur représente à quel point i est connecté à C, et en particulier à ses nœuds les plus représentatifs. D'autre part, la cohésion intrinsèque de C, $CI(C)$. Si i était connecté à tous les nœuds de C, $FA(i, C)$ serait donc égale à $CI(C)$. Plus la valeur de $FA(i, C)$ sera proche de la valeur de $CI(C)$, plus on considérera comme probable que i appartienne à C. Pour que i soit ajouté à C, il faut donc qu'il satisfasse au critère suivant :

$$FA(i, C) \geq \Delta * CI(C)$$

   Où Δ est un paramètre de l'algorithme, variant entre 0 et 1. Une valeur de Δ de 1 signifiera donc que les communautés trouvées seront obligatoirement des cliques, puisque n devra être connecté à tous les nœuds de C pour y être rattaché. Au contraire, une valeur de Δ de 0 permettra de découvrir des communautés proches des composantes connexes : dès qu'un nœud testera s'il peut être rattaché à une communauté, il y sera effectivement

rattaché. Notons que les communautés trouvées ne seront généralement pas exactement les composantes connexes du réseau final, à cause de l'ordre dans lequel les liens sont ajoutés.
Cette fonction est définie en 3.17

3. CONTRACTION_DIVISION :
Comme décrit dans le méta-algorithme, lorsqu'un lien est supprimé au sein d'une communauté, les deux nœuds affectés par ce changement sont susceptibles d'être rejetés de la communauté. Le même critère que précédemment est adopté, et pour chacun des nœuds, on vérifie donc la propriété suivante :

$$FA(i, C) \geq \Delta * CI(C)$$

Où $CI(C)$ est calculé en considérant que i n'appartient pas à C. Autrement dit, on calcule si i serait ajouté à C s'il n'en faisait pas déjà partie. Dans le cas où l'un des deux nœuds serait rejeté par ce processus, on va alors répéter récursivement le calcul pour chacun de ses voisins.
En effet, lorsqu'un nœud est rejeté d'une communauté, cela peut remettre en cause la cohérence de la communauté. Les nœuds appartenant à C et connectés à n perdent tous un lien interne et gagnent donc un lien externe, ce qui peut remettre en cause leur appartenance à la communauté. Par ce procédé récursif, une simple suppression d'un lien peut donc, dans certains cas, amener à retirer successivement tous les nœuds de la communauté, et donc à supprimer cette dernière.
Nous n'avons pas proposé d'implémentation particulière pour la division. En effet, dans l'approche classique à base d'instantanés, le processus de division est évident. Pour donner un exemple, à un instant, les nœuds $\{a, b, c, d, e, f\}$ forment un groupe, puis, à l'instant suivant, on peut observer un groupe $\{a, b, c\}$ et un groupe $\{d, e, f\}$, par exemple. Cependant, dans notre cas, où l'évolution est traitée pas à pas, cette division n'a normalement pas lieu d'un seul coup. Elle se fait progressivement, les liens étant coupés l'un après l'autre. Les nœuds sont donc retirés successivement, et il n'y a pas de moment de division clairement défini. Lorsqu'une division d'une communauté à lieu, celle-ci est donc simplement vue comme la contraction d'une communauté ($\{a, b, c, d, e, f\} \rightarrow \{a, b, c\}$) et la naissance d'une autre ($\varnothing \rightarrow \{d, e, f\}$).
Cette fonction est définie en 3.18

4. MORT
On considère tout simplement qu'une communauté disparaît lorsque le nombre de nœuds qu'elle contient tombe en dessous du nombre minimum de nœuds nécessaires pour créer une communauté embryonnaire. Par exemple, si l'on considère qu'une communauté embryonnaire est une clique de taille 3, alors une communauté disparaitra si elle ne contient plus que 2 nœuds. Cette disparition peut avoir lieu suite à une suppression de nœuds en cascade comme expliqué en 2).
Cette fonction est définie en 3.19

5. FUSION
L'un des problèmes de la version précédente de l'algorithme était la fusion de communautés partageant des hubs. Ce même problème peut se retrouver à l'identique dans d'autres méthodes de détection de communautés, telle OSLOM [LRRF11], qui utilisent la même méthode : deux communautés sont fusionnées si un certain pourcentage de leurs nœuds sont en commun. Le problème est qu'une fois de plus, les propriétés des nœuds en communs sont ignorées. Dans les réseaux de terrain pourtant, le cas peut être très différent si les

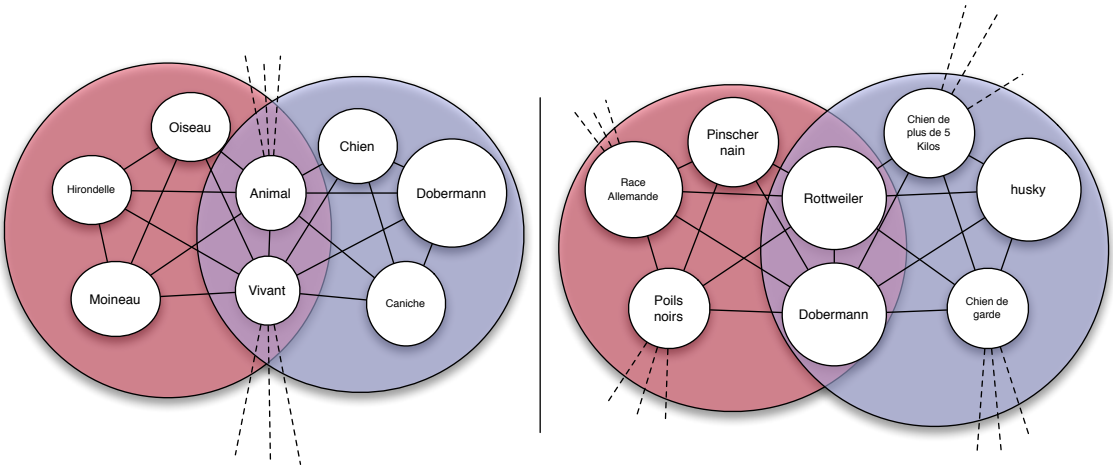


FIGURE 3.21 – *Illustration du rôle des hubs dans la fusion de communautés. Dans le schéma de gauche, les nœuds en commun sont des hubs, des nœuds très généraux. Il n'y a pas vraiment de raison de fusionner les deux communautés (Chiens, Oiseaux). Dans le schéma de droite, le taux de recouvrement est le même, mais les nœuds en commun ne sont pas des hubs. Dans ce cas, il pourrait être raisonnable de fusionner les deux communautés (nouvelle communauté Chien)*

nœuds en commun sont des hubs ou au contraire si ce sont des nœuds qui n'appartiennent qu'à ces communautés.

Considérons le cas d'un réseau de coauteurs, dans lequel les nœuds correspondent donc à des auteurs. Imaginons que nous détectons deux communautés proches, correspondant pour l'une aux auteurs de méthodes de détection de communautés utilisant la modularité, et pour l'autre aux auteurs utilisant des marches aléatoires pour le même but. Les auteurs travaillant sur les mêmes problèmes sont logiquement dans les mêmes communautés. Mais les deux communautés vont aussi contenir un certain nombre d'auteurs en commun, qui correspondent aux auteurs « classiques » ou généralistes. Ce sont des hubs, qui appartiennent également à toutes les autres communautés d'auteurs liées à la détection de communauté, et le fait qu'ils soient en commun entre deux communautés n'est pas un argument pour les fusionner, du moins pas à un premier niveau de décomposition.

Au contraire, il est parfois possible que lorsque l'on se demande si deux communautés doivent être fusionnées, on trouve que malgré de nombreux nœuds en commun, elles contiennent aussi des hubs qui ne sont pas partagés. Bien que cela dénote effectivement une différence entre les groupes, il est souvent judicieux de fusionner les groupes, car des hubs différents peuvent être des artefacts. Si nous reprenons le cas de nos scientifiques, deux groupes de chercheurs peuvent travailler sur la détection de communautés avec modularité, mais en venant de domaines différents (informatique et physique, par exemple). Lorsqu'ils vont citer des auteurs de référence (hubs), ils vont certes citer des auteurs clefs de la détection de communauté, que l'on retrouvera en commun, mais aussi probablement des auteurs de référence de leurs domaines respectifs.

En résumé, pour deux paires de communautés qui partagent le même nombre de nœuds, si l'une des paires partage des hubs, et l'autre des nœuds qui n'appartiennent qu'à ces deux communautés, il est plus raisonnable de fusionner les nœuds de la première communauté. Un exemple illustré est présenté en 3.21

Pour pallier à ce problème, nous avons proposé une méthode de fusion de communautés

qui utilise à nouveau les métriques définies précédemment. Tout d'abord, comme dans la méthode précédente, la fusion de deux communautés C_{new} et C_{old} se définie comme la disparition de la plus récente d'entre elles. Cette fois cependant, tout ou partie des nœuds de cette communauté pourra être intégré dans la plus ancienne.

Lorsque l'on doit tester si deux communautés doivent être fusionnées, on va tester à quel point elles sont semblables. Pour cela, on ne va pas utiliser simplement le nombre de nœuds, mais leurs valeurs de représentativité. On calcule la similarité de la communauté la plus récente C_{new} avec la communauté C_{old} comme étant la somme des représentativités des nœuds de C_{new} appartenant également à C_{old}. En divisant cette valeur par la somme des représentativités de tous les nœuds de C_{new}, on obtient un ratio. Comme dans le cas du nombre de nœuds, on demande en paramètre de l'algorithme le ratio λ à partir duquel on souhaite fusionner les communautés, afin de pouvoir choisir entre un résultat synthétique (peu de communautés bien distinctes) ou détaillé (beaucoup de communautés pouvant être semblables). Le critère pour fusionner C_{new} à C_{old} peut donc être défini par la formule suivante :

$$\lambda * CI(C_{new}) < \sum_{i|i \in (C_{new} \cap C_{old})} Rp(i, C_{new})$$

On voit que par ce procédé, le fait d'avoir des hubs en commun ne sera pas déterminant au moment de décider si les communautés sont semblables ou non.

Dans le cas où l'on décide de fusionner les communautés, la plus récente est donc supprimée. Mais, pour chacun de ses nœuds, on va vérifier s'il ne pourrait pas être intégré par la communauté conservée. On essaye tout simplement d'ajouter le nœud selon la méthode décrite précédemment, en comparant la force d'appartenance à la cohésion intrinsèque. L'ensemble des nœuds de la nouvelle communauté est donc défini par la formule suivante :

$$\{n_1 | n_1 \in C_{old} \cup (n_2 | n_2 \in C_{new}, FA(n_2, C_{old}) > \Delta * CI(C_{old}))\}$$

Cette fonction est définie en 3.20 On peut se demander pourquoi, si ces nœuds peuvent être ajoutés par le calcul habituel à C_{old}, ils n'ont pas déjà été intégrés dans cette communauté. Cela est tout simplement dû à l'ordre dans lequel le réseau a été créé. Cet ordre peut faire que la question d'intégrer ces nœuds à cette communauté ne s'est encore jamais posée. Ceci est expliqué en détail dans la section 3.3.3.5.

3.3.3.4 Discussion sur la complexité

Pour cette version de l'algorithme, il n'est pas non plus possible de calculer une complexité qui puisse refléter la rapidité de l'algorithme dans des cas réels. Nous avons pu l'appliquer sur des graphes comportant des millions de nœuds en un temps limité. Nous l'avons également comparé avec des algorithmes classiques du domaine sur des graphes générés. Comme on peut le voir sur la figure 3.22, il est plus rapide que de nombreux algorithmes, en particulier des algorithmes avec recouvrement. En revanche, certains algorithmes utilisant des techniques gloutonnes, comme la méthode Louvain sont encore plus rapide. Le point important que l'on observe est que, sur ces réseaux, générés pour avoir les mêmes propriétés que des réseaux de terrain, la complexité est linéaire avec le nombre de liens (le degré des nœuds étant constant). Un autre point important est que la consommation mémoire est limitée. Certains algorithmes, comme CFinder, ont une consommation mémoire qui peut être prohibitive, car exponentielle avec la taille ou la densité du réseau. Cette implémentation d'iLCD n'a pas ce problème puisqu'il ne stocke que le réseau dans son état d'évolution courant et les communautés qu'il a détecté.

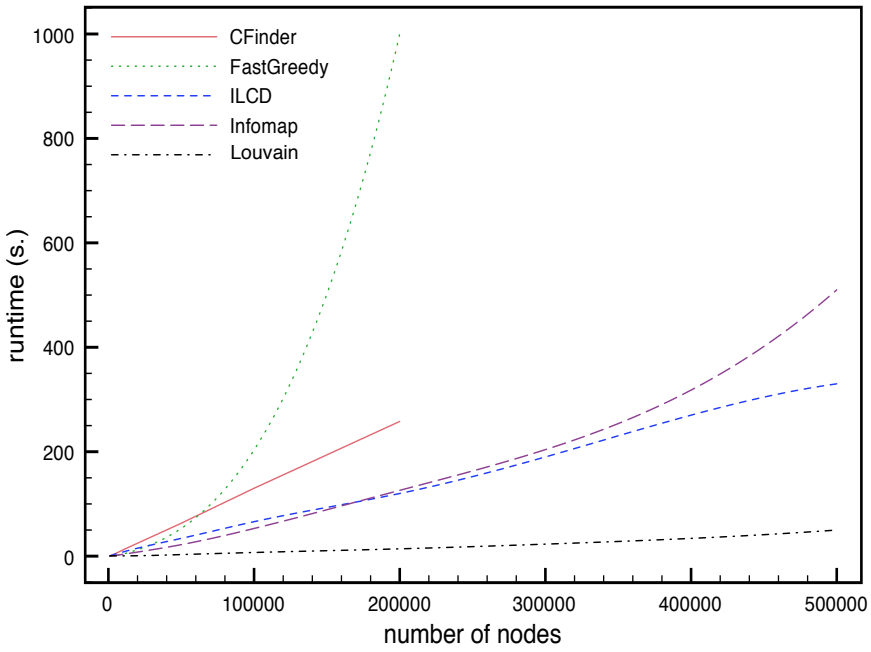


FIGURE 3.22 – *Comparaison de la vitesse d'exécution de plusieurs algorithmes connus. iLCD est linéaire avec le nombre de nœuds et de liens du réseau.*

3.3.3.5 Limites

Bien que cette seconde version de l'algorithme corrige plusieurs problèmes de la version précédente, et qu'elle ait des avantages par rapport aux algorithmes existants par son aspect dynamique, il reste quelques limites qu'il est bon d'énoncer ici.

- Taille et densité des communautés : comparé aux algorithmes habituels, cet algorithme a tendance à trouver des communautés plus petites et plus denses. Ceci est dû au fait que seules des communautés intrinsèques sont détectées, et donc des communautés atomiques, dans lesquelles on ne trouve pas de sous-structure interne. Pour obtenir des communautés moins denses, il faudrait s'intéresser à des communautés de communautés.
- Ordre des liens : les communautés obtenues au final dépendent évidemment de l'ordre dans lequel les liens sont apparus et ont disparu. Pour un réseau final identique produit par deux évolutions différentes, les communautés finales peuvent donc être différentes. Cela n'est pas un problème en soi, mais une conséquence de la méthode d'intégration des nœuds aux communautés peut paraître surprenante : quand un nouveau lien est ajouté entre les nœuds $n1$ et $n2$, seuls ces nœuds pourront être intégrés dans une communauté ce tour-ci. Or, cette intégration de $n1$ ou $n2$ à une communauté peut affecter les nœuds qui lui sont liés, et pourraient éventuellement les faire intégrer une communauté. Cependant, dans cet algorithme nous ne faisons pas ces tests récursifs. Cela permet d'une part de réduire la complexité, et, d'autre part, nous avons considéré que cela représentait autant un avantage qu'un inconvénient. L'inconvénient est que des nœuds qui auraient des valeurs de Force d'Appartenance suffisantes pour être intégrés dans une communauté ne le sont pas. L'avantage est que cela assure qu'un nœud doit interagir directement avec une communauté pour l'intégrer. Concrètement, un nœud doit créer un lien avec un nœud à l'intérieur

d'une communauté pour l'intégrer. Le fait qu'il soit lié avec des nœuds qui intègrent cette communauté ne suffit pas ; à un moment donné, il faudra que lui-même crée explicitement un lien dans la communauté. L'ordre des liens est donc utilisé comme une information qui permet d'améliorer la qualité des communautés. Prenons un exemple : A, B et C sont des chercheurs qui travaillent ensemble, formant une communauté. A effectue un déplacement, et va travailler avec un quatrième chercheur D. Un peu plus tard, B les rejoint, et vient travailler avec D. $\{A, B, D\}$ forment donc maintenant une nouvelle communauté. Si les liens (A, C) et (B, C) sont toujours actifs, alors, une vision statique à ce moment là aurait tendance à vouloir insérer également C dans cette nouvelle communauté $\{A, B, D\}$. Cependant, avec notre algorithme, tant que le lien C, D n'aura pas effectivement été créé, C ne sera pas intégré à la communauté, la séquence des événements ne nous donnant pas de raisons de le faire.

3.3.3.6 Variante de la métrique de représentativité

La présence de nœuds ayant peu de liens mais uniquement des liens internes à une communauté peut parfois conduire à des communautés moins stables. En effet, si l'on prend l'exemple d'une communauté de 10 nœuds, dans laquelle l'un des nœuds, i, n'aurait que deux liens, tous deux internes, i aurait donc une représentativité de 1. Si les autres nœuds ont plus de liens internes mais également de nombreux liens externes, on peut se retrouver dans un cas où la représentativité de i serait surévaluée. Bien que faiblement connecté à la communauté, c'est lui qui serait considéré comme le plus représentatif. Pour compenser cela, nous avons essayé une autre méthode dans laquelle la représentativité est calculée comme étant

$$RpBis = \frac{d_C^{int}(i)}{d(i)} * d_C^{int}(i)$$

Les différences n'étaient cependant pas très importantes, dans le reste de ce document nous traiterons donc les résultats avec la version originale, que nous avons testée plus exhaustivement.

3.3.3.7 Bilan de la seconde implémentation d'iLCD

iLCD-NRMH a donc plusieurs avantages par rapport à la version précédente, et en particulier la possibilité d'étudier de vrais réseaux temporels, sur lesquels on peut à la fois ajouter et retirer des liens.

Les communautés trouvées par cette implémentation correspondent également à ce que nous avions défini comme les communautés que l'on cherchait :

- Les communautés sont denses, puisqu'elles sont initialement constituées de cliques et ne peuvent s'agrandir qu'en ajoutant des nœuds connectés à une proposition importante de ces nœuds. La densité dépend du paramètre Δ, comme cela sera expliqué dans la section 3.4.2. Plus ce paramètre est haut, plus les communautés seront bien définies.
- Les communautés sont nettement séparées du reste du réseau, puisque tout nœud ayant de nombreux liens avec elle est intégré. La séparation dépend également du paramètre Δ. Plus ce paramètre est haut, moins les communautés seront clairement séparées.
- Les communautés sont intrinsèques, puisque l'ensemble du réseau n'est pas pris en compte, les ajouts ne se font qu'en considérant la frontière de la communauté.
- Les communautés sont homogènes puisque, par construction, pour qu'un nœud soit ajouté (et conservé), il faut qu'il soit connecté à une fraction suffisamment importante de nœuds représentatifs du réseau. Les formations en chaîne ou en cercle ne sont pas possible, car

chaque nœud n'est connecté qu'à une faible fraction des nœuds de la communauté. Une communauté qui présenterais des sous-communautés claires ne peut pas non plus se former, car un nœud ne peut être ajouté que s'il est connecté à une fraction suffisamment importante de nœuds représentatifs. Or, si la communauté présente une sous-structure en communauté, cela signifie que les nœuds de chaque sous-communauté ne sont connectés qu'entre eux, et donc à une faible proportion de la communauté. Dans la première implémentation, un faible nombre de liens entre les hubs des sous-communautés pouvait amener à ce type de sous-structure, mais, dans iLCD-NRMH, avec la minimisation des hubs, chaque nœud doit être directement connecté à suffisamment de nœuds représentatifs, assurant l'homogénéité de la communauté.

Dans la suite de ce document, nous allons présenter les différents travaux que nous avons pu faire autour de cet algorithme : tout d'abord valider les communautés détectées et leur pertinence, ce qui nous a conduit à essayer de répondre à la question « Qu'est-ce qu'une bonne communauté ? », ainsi qu'à une question transversale : « Si plusieurs algorithmes trouvent des résultats différents, lequel est le meilleur ? ». Une fois la pertinence des communautés trouvées assurée, nous avons cherché à appliquer cet algorithme sur des cas réels de manière à montrer ce qu'il pouvait apporter par rapport aux algorithmes existants, surtout par son côté fortement dynamique, qui n'avait pas d'équivalent au moment de sa publication.

3.4 Aspects pratiques de l'application d'iLCD à des graphes temporels

Dans cette section, nous allons discuter de deux aspects pratiques de l'utilisation de notre algorithme. Le premier concerne les propriétés qu'un graphe doit avoir pour qu'on puisse le traiter avec notre algorithme, et une proposition de méthode pour transformer des séquences d'interactions —données non manipulables efficacement par notre algorithme— en un graphe temporel lissé, adapté à la détection de communautés dynamiques.

Le second aspect concerne les valeurs des paramètres de la deuxième implémentation proposée, iLCD-NRMH, leurs rôles, impacts, et les valeurs que l'on peut leur donner.

3.4.1 Graphes aptes à être étudiés

La littérature est riche de nombreux travaux consistant à transformer des données réelles collectées sur le terrain en un réseau pouvant être étudié de manière statique. On peut citer, par exemple, la création de réseaux de co-auteurs à partir de bases de publications, ou les mécanismes d'agrégations permettant de créer un réseau statique à partir d'interactions répétées. Par exemple, lorsque l'on souhaite étudier le réseau formé par des appels téléphoniques entre individus sur une période donnée, on peut attribuer un lien entre chaque individu ayant appelé un autre au moins une fois. On peut également attribuer un poids à chacun des liens pour représenter le nombre d'appels entre ces individus. Pour éliminer du bruit dans le réseau, il est également possible de définir un seuil, et tout lien dont le poids sera en-dessous de ce seuil sera éliminé. Il existe ainsi de nombreux procédés pour créer des réseaux statiques à partir de données qui n'en sont pas sous leur forme d'origine, ou pour rendre des réseaux de terrain plus pertinents, plus faciles à étudier.

Dans le cas des réseaux temporels, peu de travaux ont été faits sur le sujet. Au cours de nos recherches, nous avons été amenés à travailler sur ces aspects, et certains d'entre eux nous ont semblé importants à décrire ici.

3.4.1.1 Création d'un réseau temporel à partir de séquences d'interactions

Si les données que l'on étudie sont parfois explicitement structurées sous la forme d'un réseau, comme par exemple dans le cas des relations « d'amitié » d'un réseau social Web 2.0, il est beaucoup plus courant que le réseau que l'on étudie soit un modèle représentant des données observées sur le terrain. Un cas classique de transformation consiste à prendre des séries d'interactions entre des agents, et à traduire ces observations sous forme de liens dans un réseau. Les réseaux temporels sont particulièrement adaptés à ce type de modélisation. Prenons le cas de communications entre utilisateurs, via un système tel que des mails, des appels téléphoniques ou tout autre média. Un réseau créé sans traitement consisterait à considérer l'existence d'un lien lors de chaque interaction, pour la durée de l'interaction seulement. Mais cela ne nous semblait que peu pertinent dans le cadre de la détection de communautés. De même que, dans les études classiques à l'aide de réseaux statiques, on applique un traitement pour obtenir un réseau fortement porteur de sens, il nous semble judicieux de prétraiter ce réseau initial pour obtenir des liens significatifs. L'idée est donc qu'un lien ne représente pas une interaction unique sur une courte durée, mais plutôt un lien perdurant dans le temps.

Si l'on prend le cas des appels téléphoniques, un simple appel isolé à un numéro ne constituera pas un lien entre moi et le détenteur de ce numéro. Cela peut être une erreur, un appel effectué par quelqu'un d'autre ou autre cas particulier. En revanche, il existe probablement des numéros que nous appelons régulièrement, c'est à dire plusieurs fois dans des intervalles plus ou moins courts. Nous avons donc mis en place une procédure pour créer, à partir de séquences d'interactions, un réseau temporel aux liens significatifs.

La procédure est la suivante : on défini une période P et une fréquence F. Si une interaction entre deux agents se produit plus de F fois au cours d'une période P, on considère qu'il existe un lien entre ces deux agents. Le lien commence lors de la première interaction, existe tant que l'on peut trouver au moins F interactions au cours de cette période P, et se termine lors de la dernière interaction.

Dans le cas où toutes les interactions sont connues à l'avance, le calcul de cette procédure ne pose pas de problèmes particuliers. En revanche, l'un des objectifs de nos travaux étant d'être capable de traiter des réseaux évoluant en temps réel, c'est à dire pour lesquels on connaît les actions passées mais non les actions futures, il est parfois difficile de dire si un lien est toujours actif ou non.

En effet, imaginons que la période P soit de 1 mois, et que l'on étudie la présence d'un lien entre les agents a et b. Au cours du mois passé, a et b ont interagi très fréquemment et un lien a donc été créé entre eux. Mais au moment présent, cela fait 25 jours qu'ils n'ont pas interagi. Est-ce que le lien doit rester actif ou est-ce que l'on doit le couper ? En effet, il est possible que si, demain, a et b interagissent F fois, nous soyons bien au sein d'une période satisfaisant les critères de création d'un lien, mais nous ne pouvons évidemment pas le savoir à l'avance, et, si ce n'est pas le cas, notre lien aura existé un mois sans interaction aucune.

Dans le cas de ces réseaux étudiés en temps réel, on crée les liens d'une manière un peu particulière : on considère à chaque instant les F dernières interactions entre chaque paire de nœuds. Si la période entre la plus ancienne de ces F dernières interactions et l'instant présent est inférieure à P, alors le lien est toujours actif. Sinon, le lien ne doit plus exister. Cela a pour effet négatif de « retarder » la création des liens par rapport à l'approche plus exacte, mais il est impossible de faire autrement dans un réseau pour lequel on ne connaît pas l'évolution future. Les deux cas sont illustrés sur la figure 3.23.

Le choix des paramètres F et P est évidemment important, dépend à la fois des propriétés du réseau et de ce que l'on souhaite étudier. Par exemple, dans un réseau qui correspondrait à

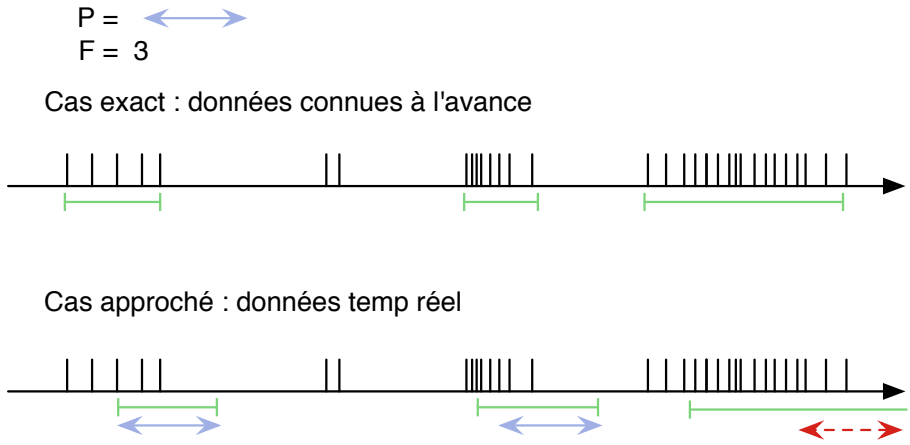


FIGURE 3.23 – *Processus de transformation de séquences d'interactions en un réseau temporel. Les lignes horizontales correspondent au temps, les lignes verticales à des interactions entre les deux même nœuds. Les lignes horizontales bornées (vertes) correspondent à la durée d'existence d'un lien. Le cas exact, en haut, est simple : le lien est créé lors de la première interaction et se termine lors de la dernière. Tant qu'il est resté présent, il n'a pas existé une période P sans au moins F interactions. Sur le cas temps réel, on observe bien le décalage : on ne peut savoir qu'il faut créer un lien qu'une fois que l'on a vu 3 interactions au cours d'une période P. De plus, le lien existe plus longtemps qu'il ne le devrait : les flèches bleues et rouges représentent ce délai : tant qu'elles existent, il y a eu au moins 3 interactions au cours de la période P passée.*

des appels téléphoniques, on peut vouloir étudier les liens extrêmement forts, et demander par exemple d'avoir au moins 3 communications par semaine pour maintenir un lien. Si par contre on voulait s'intéresser à des contacts réguliers mais moins intenses, on pourrait demander à avoir au moins 4 interactions par mois. Toutes les combinaisons sont possibles, et il est bien sûr nécessaire d'avoir une connaissance préalable du réseau pour pouvoir choisir des valeurs pertinentes.

3.4.1.2 Propriétés nécessaires du réseau étudié

Quel que soit le moyen par lequel le réseau temporel est obtenu, il devra satisfaire au moins deux critères pour que l'algorithme iLCD fonctionne efficacement :

Données fiables de manière continue Comme expliqué dans le chapitre 2.5, certains algorithmes de détection de communautés temporelles essayent de trouver des communautés pertinentes sur l'ensemble des données. Si, pendant une période, les données sont manquantes ou altérées (suite à un problème technique, ou autre), ces algorithmes peuvent théoriquement faire abstraction du problème et faire le lien entre les communautés avant et après cette période. En revanche, pour iLCD, la procédure étant itérative, si des données sont fausses ou manquantes à un moment donné, cela va forcément affecter la continuité des communautés. Dans le cas de données manquantes par exemple, on observera sur la période incriminée une forte quantité de disparitions de communautés, suivie d'une période ou de nouvelles communautés — identiques à celles existant avant le problème — vont être créées. Cela fausse donc beaucoup les résultats.

Persistance des liens Une communauté dynamique selon iLCD est une communauté dans laquelle les liens entre ses membres sont maintenus de manière durable. Si le réseau temporel étu-

dié voit tous ses liens être régulièrement coupés puis recréés, les communautés détectées auront également tendance à être détruites puis reconstruites régulièrement. Si l'on utilise la méthode présentée précédemment, un tel cas de figure peut se présenter si l'on choisit des paramètres trop restrictifs. Dans le cas des communications téléphoniques par exemple, demander 2 communications par jour conduirait à un réseau sans persistance des liens. On voit que selon ce paramètre, la plupart des liens auraient tendance à exister par intermittence : présents 1 jour, absents 2, à nouveau présents, puis absents, et ainsi de suite. Cela donnera bien des communautés, mais souvent peu pertinentes. En règle générale, on préfèrera que la période d'existence des liens soit raisonnablement grande par rapport à la durée du réseau. Par exemple, pour un réseau couvrant une période d'un an, des liens qui n'existeraient que quelques minutes ne seraient sans doute pas pertinents. Dans le cas où le réseau dont on dispose présenterait un tel profil de liens existant par intermittence, on peut justement appliquer la procédure présentée au-dessus afin d'obtenir des liens à l'existence plus significative.

3.4.2 Influence des paramètres de l'algorithme

Lorsque nous avons présenté iLCD, nous avons évoqué la présence de deux paramètres. Dans cette section, nous allons présenter plus en détail leurs rôles et leurs implications.

3.4.2.1 Coefficient de fusion λ

Ce paramètre est utilisé au moment où l'on doit décider si deux communautés ayant au moins un nœud en commun doivent être fusionnées. Ce paramètre varie entre 0 et 1.

0 Signifie que toutes communautés partageant au moins 1 nœud seront fusionnées. Au contraire, une valeur de 1 va amener à ne fusionner que les communautés rigoureusement identiques. Ces deux valeurs extrêmes sont généralement à exclure, car les résultats obtenus sont peu intéressants : peu de communautés et beaucoup de nœuds sans communautés dans un cas, ou nombreuses communautés redondantes dans l'autre.

Les valeurs les plus intéressantes sont situées dans la tranche 0,4 - 0,8. On assure ainsi que les communautés trouvées ne soient pas trop redondantes, tout en évitant de fusionner des groupes clairement différents. Si l'utilisateur n'a pas d'idées quand au résultat qu'il souhaite obtenir, une valeur de 0,6 sera généralement un bon compromis, plus de la moitié des nœuds représentatifs devant donc être en commun pour obtenir une fusion.

3.4.2.2 Δ : paramètres d'intégration des nœuds aux communautés

Cet autre paramètre, appelé Δ, représente à quel point un nœud doit être fortement connecté à une communauté pour que celle-ci l'intègre. Sa valeur peut varier de 0 à 1.

0 signifie que n'importe quel lien entre le nœud et la communauté permettra à celle-ci de l'intégrer. 1 demandera au contraire que le nœud soit lié à tous les nœuds de la communauté pour que celle-ci l'intègre. Avec une valeur de 0, les communautés obtenues seront donc de type composantes connexes, aux fusions de communautés près.

Avec une valeur de 1, les communautés seront des cliques. Ces deux types de communautés correspondent aux types extrêmes de communauté que nous avons présentés dans la section 3.2.

Le choix de cette valeur est donc très important, puisqu'il va déterminer à quel point les communautés détectées seront ou non fortement cohérentes. Il faut remarquer que plus la valeur est basse, et plus l'ordre des liens aura une forte importance. Ici aussi, les valeurs extrêmes ne seront généralement pas pertinentes, et des résultats dans la plage 0,3 - 0,7 seront les plus intéressants. Sans désidérata particulier, une valeur moyenne de 0,5 sera un bon compromis.

3.4.3 Autres aspects pratiques

D'autres aspects pratiques pourraient être traités. Nous consacrons deux chapitres de l'annexe à cette fin. Nous y parlons de la visualisation de communautés dynamiques, qui, étant donné la complexité des résultats retournés par un algorithme de détection de communautés dynamiques, est particulièrement importante. Nous discutons également des formats de fichiers, en entrée et en sortie. Là encore, le cas dynamique se révèle plus complexe que le cas statique, et nécessite des formats appropriés.

3.5 Bilan

Dans ce chapitre, nous avons présenté en détail notre solution. Nous avons d'abord expliqué quels étaient nos objectifs, en reprenant les faiblesses de l'état de l'art. Nous avons ensuite proposé un méta-algorithme, iLCD, adapté au traitement des réseaux temporels, ce qui était notre premier objectif.

Nous avons ensuite proposé deux implémentations d'iLCD, iLCD-CDIE et iLCD-NRMH, dont la deuxième répond de manière satisfaisante à tous nos objectifs, puisqu'elle trouve des communautés intrinsèques et homogènes, tout en étant peu coûteuse en temps de calcul.

La dernière partie était consacrée aux aspects pratiques de la détection de communautés, ce qui sera utilisé dans la suite pour expliquer comment nous avons appliqué iLCD sur des cas réels.

Dans le prochain chapitre, nous allons nous consacrer à montrer que les communautés trouvées par iLCD-NRMH sont pertinentes sur des réseaux réels, ce qui faisait également partie de nos objectifs.

4

Validation des résultats : qu'est-ce qu'une bonne communauté ?

Contents

Résumé du chapitre Au chapitre précédent, nous avons présenté en détail un nouvel algorithme de détection de communautés, iLCD. La première étape est de montrer que les communautés qu'il trouve sont pertinentes. Comme nous nous intéressons particulièrement aux réseaux de terrain, ce chapitre est consacré à la validation des communautés trouvées par iLCD-NRMH, la deuxième implémentation d'iLCD, sur de tels réseaux. Afin de pouvoir comparer les résultats obtenus avec iLCD-NRMH à ceux des méthodes existantes reconnues, nous ne travaillons pour l'instant que sur des réseaux statiques.

Le problème auquel nous sommes confrontés est qu'il n'existe pas de procédure bien établie pour comparer des découpages en communautés sur des réseaux de terrain. La section 1 est donc une première réponse à ce problème : nous introduisons des métriques et des outils nous permettant de comparer des découpages en communautés proposés par différents algorithmes sur des réseaux de terrain pour lesquels on ne peut pas connaître les communautés correctes. Nous utilisons ensuite ces outils pour comparer iLCD-NRMH à d'autres méthodes, et montrer que les communautés sont pertinentes, comparées à celles trouvées par les autres solutions.

Dans la section 2, nous prenons une approche différente : toujours en travaillant sur des réseaux de terrain, nous souhaitons cette fois pouvoir évaluer directement la qualité des communautés trouvées, et non pas seulement les comparer avec d'autres algorithmes. Pour ce faire, nous demandons à des individus de noter la qualité des communautés trouvées. Le cadre applicatif est la plateforme de réseau social Web 2.0 Facebook, car cela nous permet d'avoir des utilisateurs qui évaluent des communautés sur un réseau qu'ils connaissent très bien : leur propre réseau social. Dans ce cas aussi, les communautés trouvées par iLCD-NRMH sont validées.

Introduction Comme nous l'avons présenté au chapitre précédent, l'une des problématiques majeures de la détection de communauté est la définition de ce que sont de bonnes communautés. Pour qu'un algorithme de détection de communautés soit considéré intéressant, il faut que les communautés qu'il trouve soient pertinentes, ce qui est souvent difficile à démontrer étant donné l'absence de définition précise de ce qu'est une bonne communauté. Notre algorithme est dynamique, et, pour montrer que les communautés qu'il trouve sont pertinentes, nous allons montrer qu'elles sont pertinentes comparées à d'autres algorithmes existants reconnus, qui sont donc des algorithmes statiques. En effet, pour chaque instant t du réseau dynamique, notre algorithme dispose d'un découpage en communautés, P_t. Si un algorithme de détection de communautés statique est exécuté sur le réseau statique correspondant à l'état du réseau à l'instant t, les communautés qu'il trouve peuvent être comparées à celles de P_t. Si, pour chaque instant t, les communautés de P_t sont pertinentes, comparées à celles trouvées par l'algorithme statique, alors, nous pourrons dire que les communautés trouvées par notre algorithme sont pertinentes, d'un point de vue statique.

Pour pouvoir travailler sur un large éventail de graphes, au lieu de travailler sur des instants donnés d'un graphe dynamique, nous travaillons sur des réseaux initialement statiques, qui seront vus par notre algorithme comme le dernier état d'un graphe dynamique, dont les étapes d'évolutions sont constituées de l'ajout des liens du réseau.

Nous avons montré que la modularité, de par ses points faibles tels que la limite de résolution, ne constituait pas une définition valable de bonnes communautés. Dans le cas de communautés avec recouvrement, il est encore plus délicat d'en donner une définition.

Lorsque nous avons donné, au chapitre 3.2, la définition de communauté, nous l'avons divisée en deux parties. Le première était la définition classique : « Une communauté est un ensemble de nœuds fortement connectés entre eux et plus faiblement connectés au reste du réseau ». Le premier travail cherche donc à valider que les communautés trouvées par notre algorithme sont

valides à ce niveau, c'est à dire qu'elles offrent un compromis pertinent entre une bonne densité interne et une bonne séparation du reste du réseau.

La deuxième partie de la définition de communauté que nous avons donnée, qui ne concerne que les réseaux de terrain, énonce que les communautés sont des ensembles de nœuds qui sont « pertinents au niveau du sens ».

C'est ce que nous montrons dans la deuxième partie de ce chapitre, en conduisant une expérience sur un réseau social web 2.0 faisant intervenir des utilisateurs de ce réseau, aptes à juger des communautés trouvées. Ce travail a été présenté lors du workshop *Web Intelligence and Communities*, et a été effectué avec Maud Leguistin (sociologue) et Frédéric Amblard. [CLA12]

4.1 Comparaison de communautés sur des réseaux de terrain dont les communautés sont inconnues

Nous avons vu dans l'état de l'art qu'il existe de très nombreux algorithmes de détection de communautés. On peut observer facilement que les résultats fournis par différents algorithmes sur des réseaux identiques sont souvent extrêmement différents, dès lors que les communautés ne sont pas triviales. Or, nombre de ces algorithmes sont considérés comme bons, et leurs résultats semblent souvent pertinents. Que doit-on donc déduire ce cet état de fait, lorsque l'on travaille sur des réseaux de terrain, dont les communautés « correctes » ne sont pas connues *a priori* ?

Est-ce que l'on doit considérer qu'il n'existe qu'un et un seul résultat optimal, et que les autres sont forcément moins bons ? Ou est-ce que, au contraire, chaque résultat est une solution qui n'est ni plus, ni moins pertinente que les autres ?

La juste réponse se trouve certainement entre ces deux extrêmes : s'il existe évidemment des résultats plus pertinents que d'autres, plusieurs résultats différents peuvent tout à fait être aussi valables les uns que les autres, en offrant des perspectives différentes.

Le problème que l'on pose donc est celui de l'évaluation d'algorithmes de détection de communautés sur des graphes dont on ne connaît pas le découpage optimal. Pouvoir évaluer la qualité des communautés trouvées nous permettra de valider l'algorithme iLCD-NRMH par rapport aux approches connues.

4.1.1 Travaux existants

De manière assez surprenante, s'il existe plusieurs travaux consistant à comparer des algorithmes sur des graphes générés, et donc à solutions connues [DDGDA05, LF09, XKS13], les travaux étudiant notre problème sont assez rares.

L'approche adoptée par certains chercheurs pour comparer des découpages sur des graphes de terrain est d'utiliser tout simplement la modularité pour comparer les différentes solutions. Cependant, outre les fortes limites de la modularité présentées précédemment, il n'est pas envisageable de comparer via la modularité des méthodes basées sur l'optimisation de cette même modularité à d'autres qui ne l'utilisent pas.

Dans [NC10], nous avions déjà cherché à comparer les solutions fournies par plusieurs algorithmes sur des graphes de terrain. Ceci nous avait permis de mettre en évidence la grande différence qu'il pouvait y avoir entre les différents résultats, ainsi qu'à révéler le problème des super-communautés : certains algorithmes ont tendance à trouver une communauté de très grande taille, contenant par exemple plus de 30% des nœuds du réseau, y compris dans de très grands réseaux, ce qui semble généralement peu pertinent (La majorité des autres communautés étant de tailles comparativement très petites). Un autre travail peut être rapproché de ce

problème, par Leskovec et al [LLM10]. Dans cet article, les auteurs utilisent plusieurs métriques pour comparer les solutions fournies par différents algorithmes. Leur proposition principale est d'utiliser la conductance et le diamètre des communautés.

La conductance [KVV04, SM00] d'une communauté est simplement définie comme le rapport entre le nombre de liens sortants de la communauté et la somme des degrés de ses nœuds. Elle varie donc entre 0 (composante connexe) et 1 (communauté sans liens internes, ou ayant une infinité de liens externes).

$$f(S) = \frac{C^{ext}}{2C^{int} + C^{ext}}$$

On voit que cette valeur fait déjà intervenir les deux aspects dont nous avions parlé dans notre définition d'une communauté : la séparation du reste du réseau (représentée par le nombre de liens externes) et la cohésion interne des communautés (représentée par le nombre de liens internes).

Cette métrique ne peut cependant pas être utilisée, seule, comme un moyen d'évaluer la qualité d'une communauté. On peut donner au moins deux raisons pour laquelle cette métrique seule n'est pas pertinente :

Ignorance de la structure interne Pour deux communautés ayant une même conductance de 0.5 par exemple, c'est à dire autant de demi-liens internes que de demi-liens externes (Les demi-liens sont des moitiés de lien, c'est à dire qu'à chaque lien du réseau correspondent deux demi-liens, chacun connecté à l'un des nœuds du lien), on peut avoir affaire à des cas très différents. On ne connait tout d'abord rien de la façon dont les liens internes sont répartis. Si la communauté contient 6 nœuds et 6 liens, on peut avoir affaire à deux cliques disjointes ou au contraire à des liens répartis régulièrement entre tous les nœuds. L'une de ces communautés n'est pas du tout pertinente et l'autre oui, pour une même valeur de conductance.

Mais on ne connait pas non plus la façon dont les liens externes sont répartis. Tous peuvent être concentrés sur un seul nœud, très mal séparé, alors que le reste de la communauté est parfaitement séparé du reste du réseau. Au contraire, ces liens externes peuvent être répartis équitablement entre tous les nœuds, ce qui change beaucoup la vraisemblance de la communauté.

Ignorance du reste du réseau Même dans le cas où deux communautés auraient des distributions de liens internes et de liens externes similaires, la qualité de la communauté dépend énormément du reste du réseau, des nœuds auxquels les liens externes sont connectés. Prenons le cas de deux communautés composées d'une clique de 3 nœuds, ayant chacun un lien externe. La communauté A est incluse dans un graphe de 1000 nœuds, et ses 3 liens sont liés à des nœuds pris aléatoirement dans le réseau. Cette communauté parait donc bien définie. La communauté B, elle, appartient à un réseau de 4 nœuds, et ses 3 liens externes pointent vers le même nœud, le quatrième, formant une clique de taille 4. On voit que dans ce cas, cette communauté n'a pas de sens. Pourtant, A et B ont la même conductance (et le même diamètre).

Autres métriques De nombreuses autres métriques existent pour évaluer la qualité d'une communauté. Nombre d'entre elles combinent les deux critères cités précédemment. On peut citer certaines de ces métriques :

– **Expansion**, mesure le nombre de liens par nœud qui pointent en-dehors du cluster :

$$f(C) = \frac{C^{ext}}{n_C}$$

– **Internal density** [RCC+04], représente la densité de liens internes du cluster, c'est à dire le nombre de liens présents par rapport au nombre maximal de liens possibles :

$$f(C) = \frac{2C^{int}}{n_C(n_C - 1)}$$

– **Cut Ratio** [For10], la fraction de tous les liens sortants possibles qui sont effectivement présents

$$f(C) = \frac{C^{ext}}{n_C(n - n_C)}$$

– **Normalized Cut** [SM00], qui peut être vue comme la conductance du cluster auquel on ajoute la conductance du cluster formé par le reste du réseau

$$f(C) = \frac{C^{ext}}{2C^{int} + C^{ext}} + \frac{C^{ext}}{2(m - C^{int}) + 2C^{ext}}$$

– **Maximum-ODF (Out Degree Fraction)** [FLG00], la valeur maximale, pour les nœuds du cluster, de la fraction de liens externes

$$max_{u \in C} \frac{d_C^{ext}(u)}{d(u)}$$

– **Average-ODF** [FLG00], la valeur moyenne, pour les nœuds du cluster, de la fraction de liens externes

$$f(C) = \frac{1}{n_C} \sum_{u \in C} \frac{d_C^{ext}(u)}{d(u)}$$

– **Flake-ODF** [FLG00], la fraction des nœuds de S ayant moins de liens internes que de liens externes

$$f(C) = \frac{|\{u : u \in C, |\{(u, v) : v \in C\}| < d(u)/2\}|}{n_S}$$

Toutes ces métriques ont l'avantage et l'inconvénient de ne représenter la qualité d'un découpage en communautés que par une seule valeur, ce qui est certes pratique, mais implique une hiérarchie stricte entre la qualité des communautés. Si l'on construit, à partir de l'une d'elles, une métrique globale pour évaluer la qualité d'un découpage, chaque découpage sera forcément meilleur ou plus mauvais que n'importe quel autre.

4.1.2 Approche multiobjectif pour comparer des communautés

La plupart des méthodes proposées pour évaluer la qualité d'un découpage en communautés proposent donc une métrique unique, couvrant à la fois le critère de séparation et celui de connectivité interne. Nous pensons cependant qu'une telle approche va avoir tendance à poser un *a priori* sur le type de communautés que l'on souhaite trouver.

Nous pensons donc que chacun de ces deux éléments doit être calculé séparément, et qu'à un découpage en communautés doit correspondre 2 valeurs : l'une représentant la qualité de la séparation des communautés du reste du réseau, et l'autre caractérisant la connectivité interne des communautés.

La définition de la qualité d'un découpage se rapporte donc à un problème multiobjectif classique. L'algorithme A peut très bien avoir des communautés mieux séparées du reste du réseau que l'algorithme B, mais celui-ci peut en revanche avoir des communautés avec une plus

grande densité interne. Comme dans tout problème multiobjectif, il existerait donc un front de Pareto, et plusieurs méthodes pourraient être sur ce front, c'est à dire différentes, mais pertinentes. En revanche, une méthode qui ne serait pas sur ce front de Pareto ne serait pas aussi intéressante sur le réseau étudié.

La première étape consistait donc à définir une métrique pour évaluer chacun de ces deux aspects.

4.1.3 Définition d'une métrique pour la séparation des communautés

Dans le cas de communautés sans recouvrement, une métrique s'impose logiquement pour calculer la séparation des communautés, il s'agit tout simplement du ratio entre le nombre de liens internes aux communautés et le nombre total de liens :

$$separation = \frac{\sum_{C:C\in P} |C^{int}|}{n}$$

On peut également exprimer ce calcul de la manière suivante :

$$separation = \frac{\sum_{e\in E} \theta(e)}{m}$$

où $\theta(e) = 1$ si e est un lien interne à une communauté, et $\theta(e) = 0$ si e est un lien intercommunautaire.

Une valeur de *separation* de 1 signifie donc que les communautés sont parfaitement séparées (composantes connexes), tandis qu'une valeur de 0 représente des communautés sans liens internes.

Dans notre cas, nous souhaitions pouvoir également comparer des méthodes produisant des communautés avec recouvrement, ce qui complexifie un peu le problème.

Nous avons donc développé une métrique inspirée de la précédente. A chaque lien, une valeur est attribuée, appelée *internalite*, comprise entre 0 et 1. 0 signifie que le lien est complètement externe, tandis que 1 signifie que le lien est systématiquement interne. On calcule cette valeur en fonction du rôle joué par le lien dans les communautés auxquelles les nœuds extrémités appartiennent, de la manière suivante :

Pour un lien $e = (x, y)$

$$roleNoeudLien(x, e) = \frac{|\{C:C\in P, x\in C, y\in C\}|}{|\{C:C\in P, x\in C\}|}$$

$$internalite(e) = \frac{roleNoeudLien(x,e)+roleNoeudLien(y,e)}{2}$$

où $roleNoeudLien(x, e)$ représente la vision qu'à le nœud x du lien e, c'est à dire le ratio entre le nombre de fois où il voit e comme un lien interne et le nombre de communautés auxquelles il appartient. Une fois cette valeur d'internalité calculée, il suffit de l'utiliser en lieu et place de θ dans le calcul de la séparation :

$$separation = \frac{\sum_{e\in E} internalite(e)}{m}$$

On notera que dans le cas de communautés non recouvrantes, cette formulation de la séparation est équivalente à la précédente.

A titre d'exemple, le calcul de ce ratio sur un réseau simple, selon plusieurs découpages en communautés, est présenté dans la figure 4.1

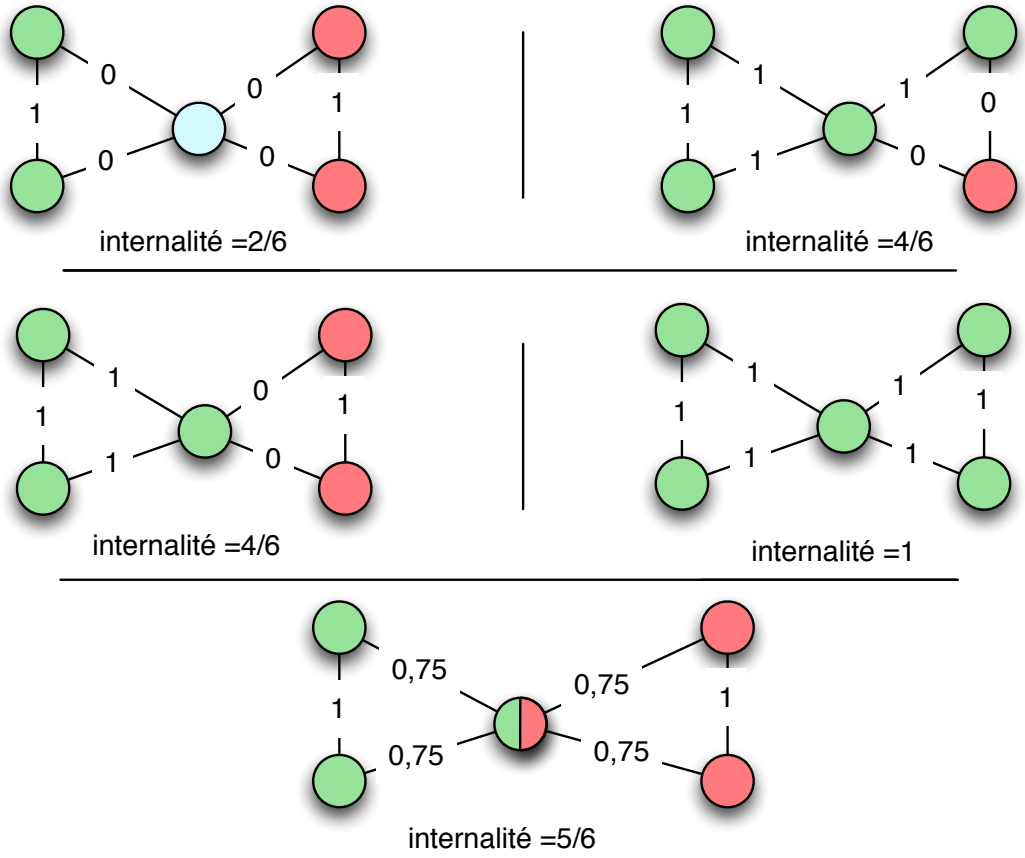


FIGURE 4.1 – *Exemples de calculs de l'internalité. On observe que des communautés de plus grande taille ont tendance à avoir une internalité plus forte. On voit aussi que le recouvrement peut améliorer l'internalité, dans certains cas.*

4.1.4 Définition d'une métrique pour calculer la cohésion interne des communautés

Pour calculer la cohésion interne des communautés, nous aurions souhaité utiliser la même approche. Il nous fallait donc d'abord trouver une métrique satisfaisante dans le cas sans recouvrement. La densité interne est généralement définie de la manière suivante :

$$densite(C) = \frac{2C^{int}}{n_C(n_C - 1)}$$

Ce qui correspond au nombre de liens dans la communauté divisé par le nombre de liens internes possibles au maximum.

La densité interne normalisée est parfois également utilisée [LKSF10] :

$$densiteNormalisee(C) = \frac{2C^{int}}{n_C - 1}$$

Si l'on souhaite calculer une valeur unique pour tout le réseau, on voit qu'on ne peut pas faire simplement la moyenne des valeurs de chaque communauté :

$$\frac{\sum_{C \in P} densite(C)}{|P|}$$

car un découpage contenant une communauté géante à faible valeur et plusieurs petites communautés de valeurs maximales (cliques) donnerait un bon score.

On souhaiterait donc que le score de chaque communauté soit pondéré par sa taille :

$$\frac{\sum_{C \in P} densite(C) n_C}{n}$$

Ou, énoncé autrement, on calcule la moyenne de la densité, pour chaque nœud, de la communauté à laquelle il appartient :

$$\frac{\sum_{v \in V} densite(C) : v \in V_C}{n}$$

On peut donc ensuite l'adapter facilement à des communautés avec recouvrement en calculant pour chaque nœud, la moyenne des densités des communautés auxquelles il appartient :

$$\frac{\sum_{v \in V} \frac{\sum_{C \in cs_v} densite(C)}{|cs_v|}}{n}$$

où, pour rappel, cs_v représente les communautés auxquelles le noeud v appartient.

Cependant, un problème se pose avec cette formule : elle a tendance à écraser les différences pour les grandes communautés. Pour illustrer ce phénomène, prenons une communauté A de 100 nœuds ayant 1000 liens internes, ce qui est courant pour une communauté trouvée par un algorithme classique sur un réseau de terrain, et une communauté B de 1000 nœuds ayant également 1000 liens internes (Ce qui est très faible, puisque le sous réseau correspondant à la communauté peut à peine être connexe). Pour A, les nœuds auront une valeur de cohésion d'approximativement 0,1. Pour B, leur valeur de cohésion sera d'environ 0,001. Ces deux valeurs sont évidemment fort différentes. Cependant, lorsque l'on va faire la moyenne de tous les scores de cohésion, les nœuds appartenant à de petites communautés auront généralement des scores beaucoup plus élevés, proches de 1. Lorsque l'on va faire la moyenne, avoir 10 communautés

telles que A, et donc raisonnables, ou une communauté peu cohérente comme B aura un impact réduit si le réseau est grand par ailleurs.

C'est pourquoi nous avons choisi de développer une métrique qui reflèterait mieux l'importante différence entre les communautés peu denses et les très peu denses, appelée la fraction de connexion interne (FCI).

Cette métrique est définie, pour chaque communauté, comme le ratio, pour chaque nœud, entre le nombre de nœuds auxquels il n'est pas lié et le nombre de nœuds auxquels il est lié :

$$FCI(C) = \frac{\frac{(n_C(n_C-1))-2C^{int}}{n_C}}{\frac{2C^{int}}{n_C}} = \frac{(n_C(n_C-1))-2C^{int}}{2C^{int}} = \frac{(n_C(n_C-1))}{2C^{int}} - 1$$

On voit donc que cette métrique est liée à l'inverse de la densité. On conservera cependant le -1 pour garder la sémantique. Cette métrique a l'inconvénient d'être une métrique ouverte : sa valeur minimale est 0, pour une clique, mais peut atteindre au maximum le nombre de liens possibles -1. Si une communauté n'a aucun lien interne, on ne peut calculer sa FCI, ce qui est cohérent, cette valeur devant alors être maximale (et la communauté étant alors complètement incohérente). Pour résumer, plus la communauté est grande, plus la valeur maximale de la métrique est élevée. Une valeur de 1 correspond à une communauté pour laquelle, en moyenne, ses nœuds ne sont connectés qu'à la moitié des autres.

Pour calculer la valeur globale pour le réseau, nous utilisons le même principe qu'expliqué précédemment, en remplaçant la densité par la FCI :

$$\frac{\sum_{v \in V} \frac{\sum_{C \in cs_v} FCI(C)}{|cs_v|}}{n}$$

Le calcul de la FCI sur quelques réseaux est illustré dans la figure 4.2, dans laquelle on reprend les mêmes exemples que pour le calcul de l'internalité.

On remarque que pour l'internalité, plus les communautés sont grandes, et plus il est facile d'obtenir une valeur élevée. Pour la FCI, c'est le contraire. Enfin, on observe que dans ce cas simple, la solution avec recouvrement est la seule à obtenir la valeur maximale à la fois pour la FCI et pour l'internalité. Cela semble pertinent, ce découpage en communautés semblant plus logique qu'un partage en communautés de tailles inégales.

4.1.5 Application à des réseaux de terrain

Une application de ces mesures est de comparer des résultats fournis par des algorithmes différents sur un même réseau de terrain. En comparant les résultats, nous pourrons savoir immédiatement quels algorithmes proposent des solutions où les communautés sont très denses, quels algorithmes proposent des solutions où les communautés sont bien séparées, et quelles solutions ne sont pas intéressantes, parce qu'elles sont à la fois moins denses et plus mal séparées qu'un autre algorithme (Elles ne sont donc pas sur le front de Pareto. Se reporter à la figure 4.3 pour un exemple de front de Pareto).

Il est intéressant de noter que l'on peut facilement trouver les extrémités du front : en cherchant les composantes connexes, nous aurons une solution pour laquelle la séparation est maximale, mais où la densité interne est faible (dépendante du réseau).

La solution dans laquelle chaque clique maximale est une communauté fournira une solution de densité interne maximale, mais de séparation faible (dépendante du réseau). Ces deux solutions fourniront donc les extrémités du front de Pareto.

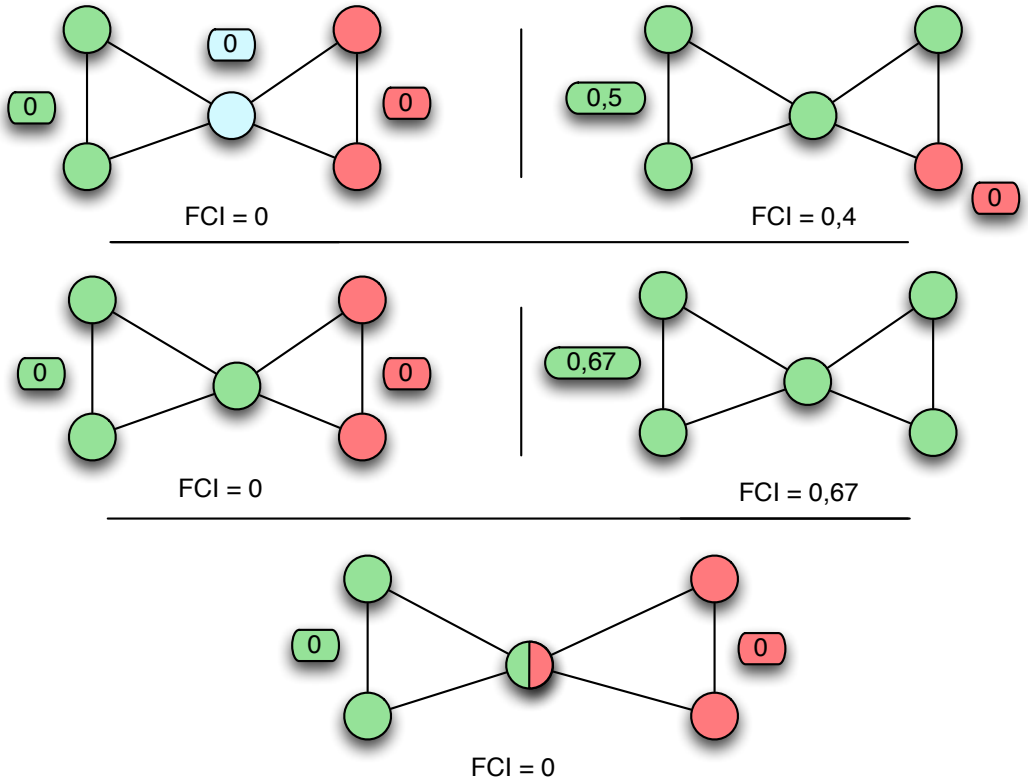


FIGURE 4.2 – *Exemples de calculs de la FCI. Les nombres sur le diagramme correspondent à la FCI des communautés prises indépendamment. On observe que des communautés de plus grande taille ont tendance à avoir une moins bonne FCI, c'est à dire une valeur plus élevée.*

Nom	Nombre de nœuds	Nombre de liens
Réseau de synonymie de verbes en français (DicoSyn) [PV98]	9147	51423
Réseau de synonymie de verbes en anglais [SNG+11]	11484	40919
Zachary Karaté Club [Zac77]	34	78
Réseau des personnages des misérables [NG04]	77	254

TABLE 4.1 – *Caractéristiques de quelques réseaux étudiés*

Dans un premier temps, nous allons montrer le type de résultats que l'on peut obtenir avec cette méthode sur quelques réseaux de terrain, dont les caractéristiques sont données dans la table 4.1.

Les algorithmes testés sont :

- **Algorithme de Louvain** [BGLL08] : Cet algorithme étant hiérarchique, nous utilisons les trois premiers niveaux de décompositions, lorsqu'ils existent. Il est intéressant de noter que plus le niveau hiérarchique est élevé, plus les communautés sont bien séparées du reste du réseau, mais plus elles sont moins denses.
- **CFinder**[PDFV05] : Cet algorithme propose un paramètre ayant une grande importance, k, représentant la taille des cliques utilisées comme élément de base des communautés. Nous fournissons les résultats pour $k = 3$, $k = 4$ et $k = 5$ lorsque cela est possible. Il est intéressant de voir que plus k augmente, plus les communautés sont denses, mais moins elles sont clairement séparées du reste du réseau.
- **InfoMap**[RB08] : Cet algorithme a été présenté en détail dans l'état de l'art.
- **iLCD-NRMH** [CAH10, CA11] : Nous avons intégré les résultats iLCD-NRMH avec les paramètres de base conseillés au chapitre 3.4.2.

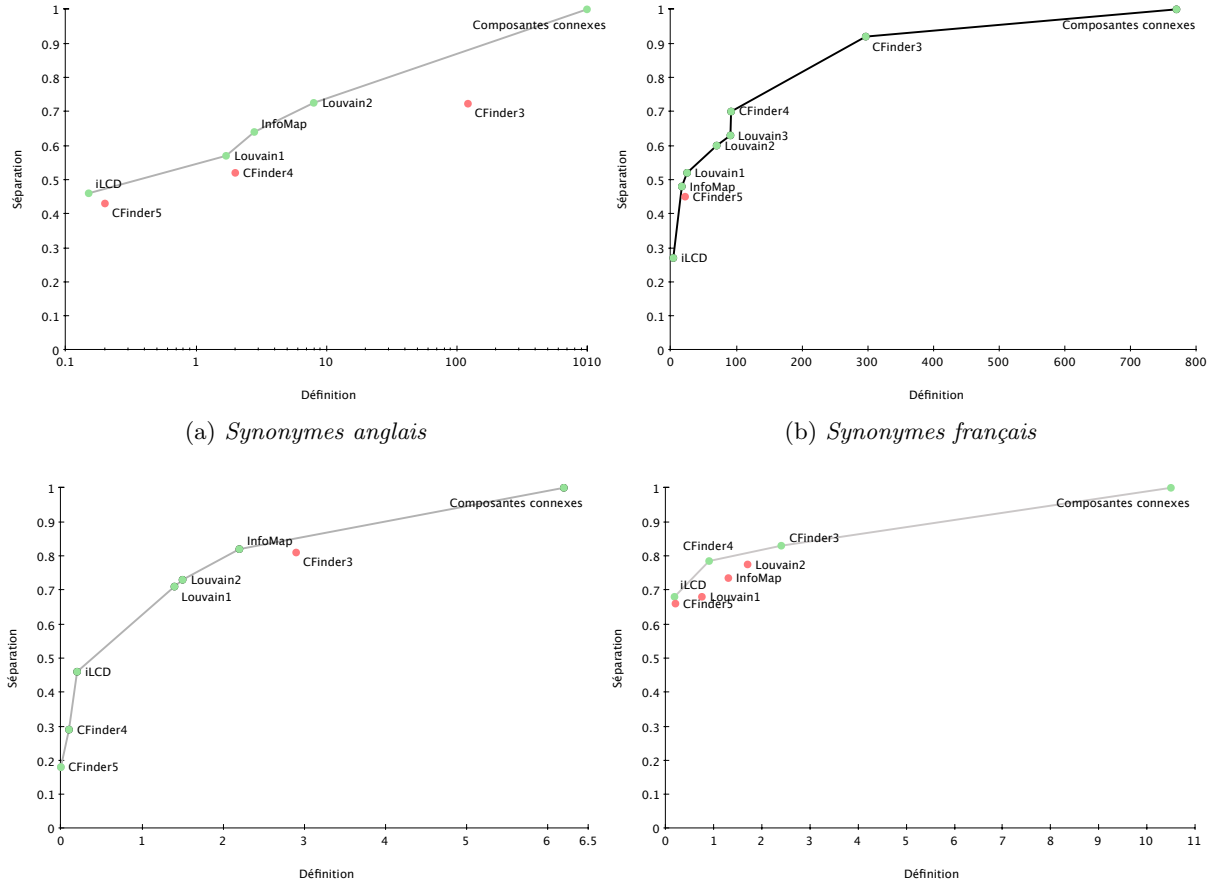
Les résultats sont présentés dans la figure 4.3, pour tous les réseaux.

Nous pouvons observer sur ces graphiques la grande différence entre les résultats. On voit que certains algorithmes trouvent des solutions ayant une très bonne séparation, mais une mauvaise densité interne, alors que d'autres proposent une solution où les communautés sont denses, mais mal séparées du reste du réseau. Les mêmes algorithmes ont tendance à fournir le même type de résultats sur tous les réseaux. Par exemple, CFinder avec $k = 3$ propose, dans tous ces exemples, une solution avec des communautés biens séparées, tandis que CFinder avec $k = 5$ trouve dans ces exemples des communautés avec une bonne densité interne.

Une première observation est que CFinder propose assez souvent des solutions qui ne sont pas sur le front de Pareto. Il semble donc que les solutions proposées par CFinder ne soient souvent pas les plus pertinentes.

Une deuxième observation est le renversement de situation sur le réseau des personnages des Misérables. Dans ce cas, CFinder donne de bons résultats, et ce sont les algorithmes Louvain et InfoMap qui ne sont plus sur le front de Pareto. Lorsque l'on étudie les communautés trouvées, on se rend effectivement compte que les résultats de Louvain et InfoMap ne sont pas très intuitifs, comme cela sera illustré dans la figure 4.9. Une explication probable est que ce réseau a une structure typique d'un réseau social complexe, et est en particulier caractérisé par du recouvrement. Les méthodes ne prenant pas en compte les appartenances multiples des nœuds sont donc défavorisées, et donnent de moins bons résultats.

La solution proposée par iLCD est souvent celle ayant la séparation entre communautés la



(a) *Synonymes anglais*

(b) *Synonymes français*

(c) *Zachary karaté club*

(d) *Personnages des Misérables*

FIGURE 4.3 – *Comparaison de différentes méthodes sur des graphes de terrain. Les nœuds en vert sont sur le front de Pareto, ceux en rouge n'y sont pas, c'est à dire que sur le réseau étudié, il existe une solution dont les communautés sont plus denses et mieux séparée du reste du réseau.*

plus faible mais, en contrepartie, les communautés les plus denses. Elle se trouve en tout cas sur le front de Pareto, car offrant une séparation des communautés meilleure que les cliques maximales, et une densité interne plus élevée que la plupart des autres solutions. On pourrait en fait rapprocher les communautés trouvées par iLCD des communautés cœurs proposées par certains auteurs, [SG12] par exemple, qui pointent du doigt l'instabilité, l'incertitude des communautés détectées par les algorithmes habituels . Les communautés détectées par iLCD sont en effet plus denses, et donc plus fiables que les communautés proposées par la plupart des algorithmes.

4.1.6 Visualisation de profils de communautés, pour une analyse détaillée

Si pouvoir comparer d'un coup d'œil les propriétés des algorithmes, et évaluer leur qualité, était un résultat appréciable, nous étions également intéressés pour voir le détail des propriétés de chacune des communautés. Nous avons donc proposé un outil permettant de visualiser ces propriétés pour l'ensemble des communautés trouvées par un algorithme. Pour ce faire, nous utilisons à nouveau un graphique, reprenant les deux mêmes dimensions, mais, cette fois, chaque point correspond à une communauté trouvée par un algorithme. Les points sont en fait constitués d'un cercle dont le diamètre est proportionnel au nombre de nœuds de la communauté. Pour chaque algorithme, on peut ainsi obtenir un profil caractéristique.

La figure 4.4 présente ces profils pour cinq algorithmes sur le réseau des verbes de la langue française. On peut observer des caractéristiques intéressantes : pour CFinder 4, le résultat est constitué d'une très grande communauté principale, peu dense mais très bien séparée, et d'un ensemble de petites communautés très denses (visibles à gauche).

La solution de Louvain est constituée d'une dizaine de très grandes communautés, assez bien séparées du reste du réseau mais très peu denses, puis de nombreuses petites communautés de forte densité interne.

InfoMap propose un profil un peu similaire, mais les grosses communautés sont beaucoup moins marquées.

iLCD, en revanche, n'a pas trouvé de communautés de grande taille sur ce réseau, mais plutôt de très nombreuses communautés, denses mais mal séparées.

Ce qui ressort surtout de cette visualisation est la grande variabilité des résultats des différents algorithmes.

4.1.7 Note sur la séparation des communautés

Le critère de séparation des communautés est une question délicate pouvant parfois prêter à confusion. Dans les premières tentatives de définition des communautés, on trouvait des règles de type : « Une communauté doit avoir plus de liens internes que de liens externes ». Aujourd'hui encore, certains auteurs ont tendance à considérer qu'un nœud appartenant à une communauté, pour bien lui appartenir, doit avoir plus de liens internes qu'externes.

Or, si cela n'est pas une propriété spécifique des communautés que l'on cherche à trouver, cette logique est erronée. Prenons un contre exemple :

La communauté C est composée d'une clique de 6 nœuds. Chacun des nœuds a donc 5 liens internes. Imaginons maintenant que chacun de ces nœuds ait 10 liens externes. La communauté a donc beaucoup plus de liens externes que de liens internes. Chaque nœud pris indépendamment a, lui aussi, plus de liens externes que de liens internes. Cependant, si cette communauté est incluse dans un réseau comportant 1 million de nœuds, et que ces liens externes sont répartis aléatoirement parmi ces derniers, on voit bien que la structure en communauté de C s'impose

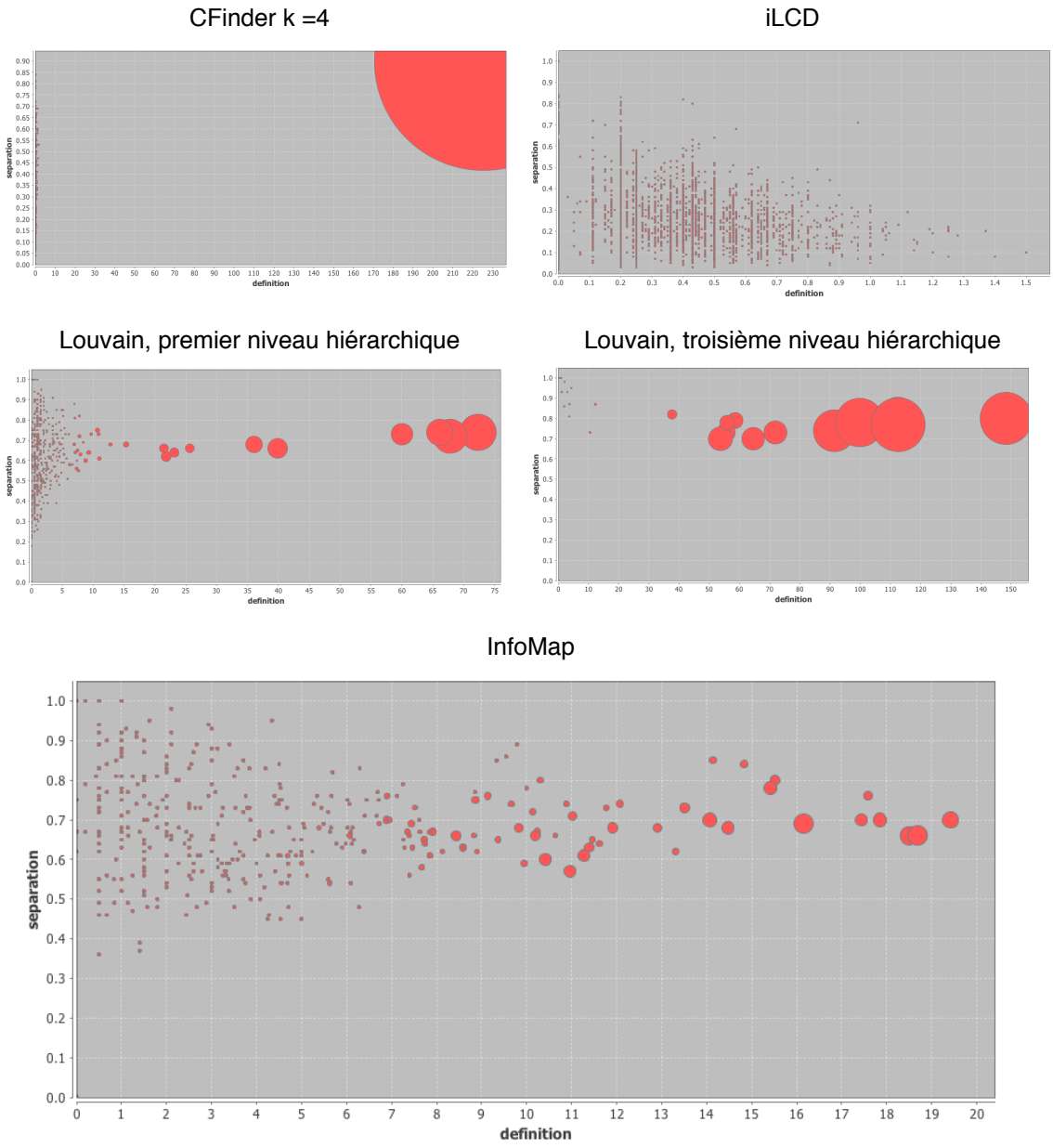


FIGURE 4.4 – *Visualisation des profils de différents algorithmes sur un même réseau. Chaque cercle correspond à une communauté, son périmètre correspond au nombre de nœuds qu'elle contient.*

clairement. Le fait que le ratio entre les liens externes et les liens internes soit élevé n'est donc pas un argument suffisant pour dire que la communauté n'est pas pertinente.

4.1.8 Outils complémentaires

Les deux outils présentés précédemment permettent d'avoir un aperçu des résultats trouvés par différents algorithmes. Lorsque l'on s'intéresse à la détection de communautés sur des réseaux de terrain, il est souvent nécessaire d'avoir une idée du résultat fourni par les algorithmes. La représentation visuelle est l'approche la plus simple pour se faire une idée.

Lorsque les réseaux contiennent peu de nœuds et de liens, nous pouvons utiliser une visualisation complète du réseau, les nœuds étant colorés de manière à faire ressortir les communautés.

Cependant, dès que le nombre de nœuds dépasse quelques milliers, il devient difficile de trouver une visualisation convaincante. Pourtant, lorsque l'on souhaite, par exemple, comparer intuitivement les résultats fournis par deux algorithmes, ou tout simplement essayer d'évaluer manuellement une solution, il est nécessaire d'avoir une visualisation. Certains travaux récents [GPK12b] proposent d'utiliser une visualisation en 3 dimensions, éventuellement en faisant intervenir de la stéréoscopie, pour aider l'utilisateur à mieux voir le réseau, et en particulier ses communautés. Cependant, dans le cas de réseaux de très grande taille et dont les communautés sont très complexes (en particulier, avec recouvrement), cette méthode n'est pas encore satisfaisante.

Ne trouvant pas d'outils adaptés à une telle exploration pour les grands graphes, nous avons développé une application, intégrant les deux outils présentés ci-dessus ainsi que d'autres visualisations. Nous allons rapidement présenter ces nouvelles visualisations dans cette section, ainsi que des exemples où leur utilisation peut nous aider à appréhender les caractéristiques de différents découpages en communautés.

4.1.8.1 Distribution des nœuds par la taille de leurs communautés

Dans une publication précédente, [NC10], nous avions montré que l'un des problèmes de plusieurs algorithmes était la présence de « super-communautés », des communautés de trop grande taille non pertinentes. Il existe un problème opposé, qui est celui de la présence d'un grand nombre de nœuds sans communautés. Dans un algorithme comme CFinder, en effet, tout nœud doit appartenir à au moins une clique de taille k pour appartenir à une communauté. Dans certains réseaux, de nombreux nœuds n'appartiennent à aucune clique de taille supérieure ou égale à 3, et resteront donc esseulés. Nous avons donc développé un outil pour visualiser la proportion de nœuds affectés par ces problèmes.

Concrètement, l'utilisateur définit deux valeurs, correspondant aux tailles minimales et maximales des communautés qu'il souhaite trouver. Par exemple, il pourra décider que des communautés de taille inférieure à 3 ne seront pas pertinentes, de même que les communautés contenant plus de 100 nœuds. On peut observer le résultat obtenu pour les réseaux de synonymie sur la figure 4.5.

Comme on peut le voir, selon les algorithmes, plus de la moitié des nœuds peuvent appartenir à des communautés trop petites ou trop grandes (Il est bien entendu que la définition de communautés trop grandes est ici purement arbitraire, et donnée à des fins d'illustration).

4.1.8.2 Visualisation de la topologie interne des communautés

Cet outil très simple est extrêmement utile pour jauger la cohésion interne des communautés. Il s'agit tout simplement de proposer, pour chacune des communautés trouvées, une visualisation

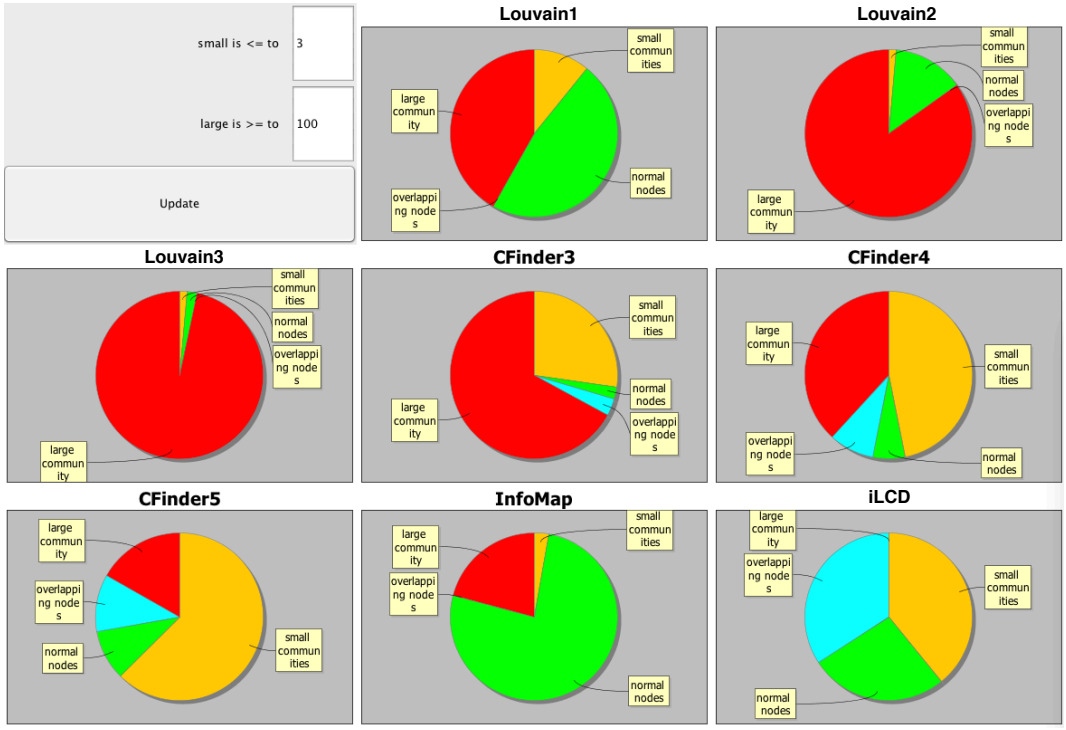
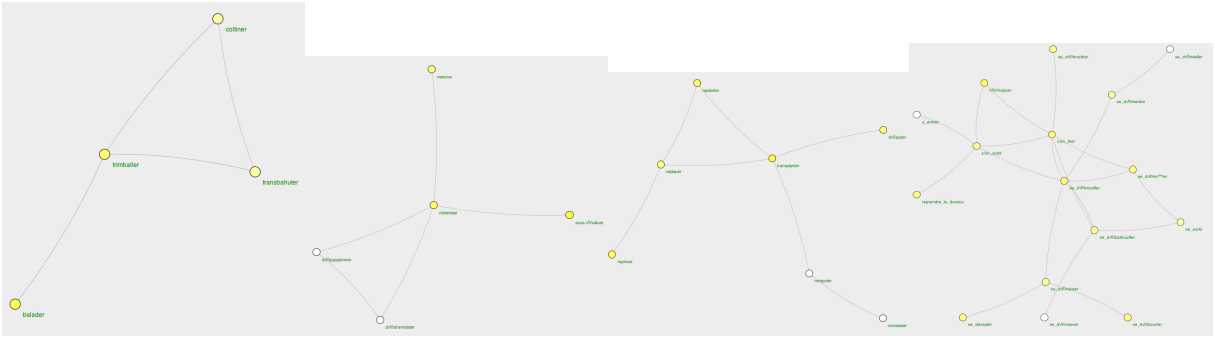
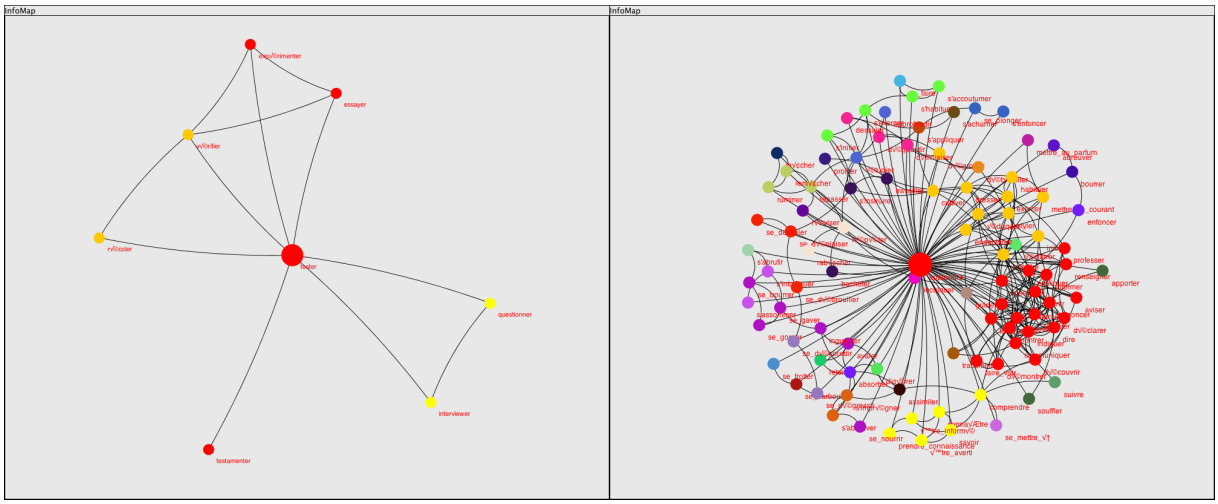


FIGURE 4.5 – *Distribution des nœuds par la taille des communautés auxquelles ils appartiennent. On voit ici que la plupart des nœuds appartiennent à des communautés de plus de 100 nœuds pour les algorithmes Louvain et CFinder avec k=3. Pour les autres valeurs de k et pour iLCD, de nombreux nœuds appartiennent à des communautés de taille inférieure à 4, ils sont en fait laissé seuls (sans communautés).*



(a) *Cohésion interne : 0,5* (b) *Cohésion interne : 1* (c) *Cohésion interne : 2* (d) *Cohésion interne : 5*

FIGURE 4.6 – *Exemples de visualisation topologique de communautés, pour différentes valeurs de cohésion interne.*



(a) *Réseau égocentré du nœud "tester"* (b) *Réseau égocentré du nœud "apprendre"*

FIGURE 4.7 – *Exemples de réseaux égocentrés dans le réseau de synonymie.*

de ses nœuds et de ses liens internes en utilisant un algorithme de positionnement classique de type masse ressort. Cet outil est particulièrement pratique combiné à la visualisation introduite en 6.1 : chaque cercle correspondant à une communauté peut être cliqué pour atteindre la visualisation de sa topologie interne. Cela permet de voir plus concrètement ce que signifie une cohésion interne faible ou élevée. Par exemple, la figure 4.6 présente des communautés trouvées par l'algorithme InfoMap sur les réseaux de synonymie. Sur ces réseaux, on peut intuitivement se faire une idée de la pertinence ou non des communautés.

4.1.8.3 Visualisation du réseau égocentré des nœuds

Les réseaux égocentrés sont couramment utilisés dans l'étude des réseaux sociaux, en particulier par les sociologues. Le réseau égocentré d'un nœud correspond à ce nœud, ses voisins, les liens entre ce nœud et ses voisins, et les liens entre ses voisins.

Visualiser le réseau égocentré d'un nœud peut se révéler particulièrement utile pour visualiser la pertinence d'un découpage en communautés. La Figure 4.7 présente ce type de réseau pour deux nœuds issus du réseau de synonymie, avec l'algorithme InfoMap. A chaque communauté différente est attribuée une couleur, et le réseau égocentré du nœud choisi est alors représenté, toujours en utilisant un positionnement masse-ressort [KK89]. Cela permet de se rendre compte si le nœud appartient bien à la communauté paraissant la plus pertinente, ainsi que de la présence ou non de plusieurs communautés auxquelles le nœud pourrait potentiellement appartenir.

Un aspect qui nous paraissait également intéressant était de pouvoir comparer les communautés trouvées sur le réseau égocentré d'un nœud selon des algorithmes de détection de communautés différents. Pour ce faire, nous avons proposé un outil permettant de les comparer simultanément en essayant de garantir une certaine uniformité de couleurs, ainsi qu'un positionnement similaire des nœuds, afin de facilement reconnaître les communautés détectées de manière identique par les différents algorithmes. La similitude des couleurs se fait par un mécanisme très simple : la plus grosse communauté est toujours représentée de la même couleur, quelle que soit la solution. Il en est de même pour la seconde et la troisième plus grande communautés, s'il y en a une. Si cela n'assure pas avec certitude que les communautés similaires seront

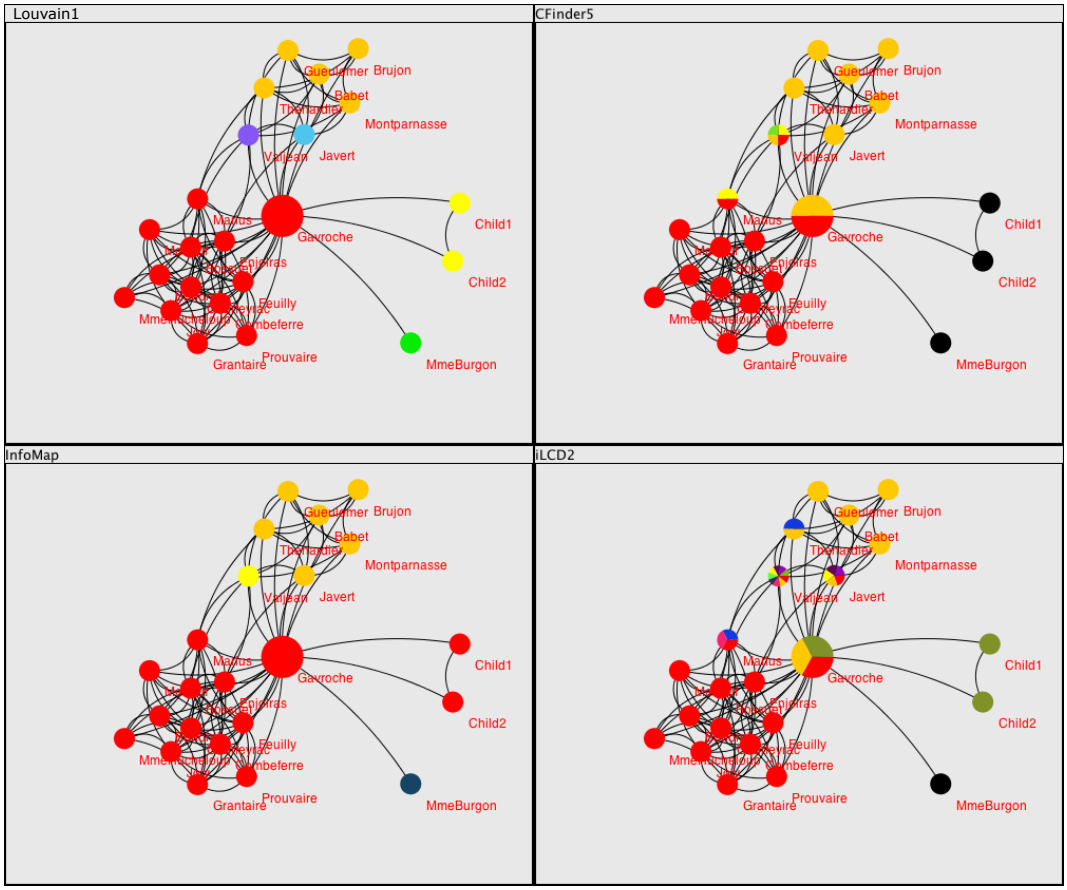


FIGURE 4.8 – *Réseau égocentré du nœud correspondant à Gavroche, dans le réseau des Misérables. On observe que les 4 algorithmes étudiés trouvent les mêmes communauté principales (rouges et jaunes). En revanche, l'intérêt des communautés avec recouvrement est évident : Gavroche appartient clairement à ces 2 communautés.*

de la même couleur, dans les faits, cela semble efficace. Une solution basée sur une similarité en terme de nœuds inclus dans les communautés serait très complexe à mettre en œuvre dès que l'on aurait plus de deux solutions à comparer.

La figure 4.8 donne un exemple intéressant pour le personnage de Gavroche dans les misérables. On observe que pour les algorithmes sans recouvrement de communautés, l'attribution de ce personnage à une seule communauté semble contre-intuitive. Au contraire, les algorithmes avec recouvrement (ici, iLCD et CFinder) proposent un résultat intuitivement cohérent.

La figure 4.9 montre ce même réseau pour Jean Valjean, le personnage principal du roman dont le réseau est issu. On peut observer que pour infoMap et Louvain, son attribution semble incohérente, la communauté choisie paraît être la moins pertinente. Il s'agit d'une communauté en étoile, dont ce nœud est le centre, et contenant tous les nœuds qui n'appartiennent à aucune autre communauté évidente. Au contraire, pour iLCD, on s'aperçoit que l'on touche aux limites de la visualisation, ainsi qu'à l'un des problèmes de la détection de communautés avec recouvrement : les nœuds peuvent appartenir à de nombreuses communautés, ce qui rend ce type de résultat difficile à exploiter.

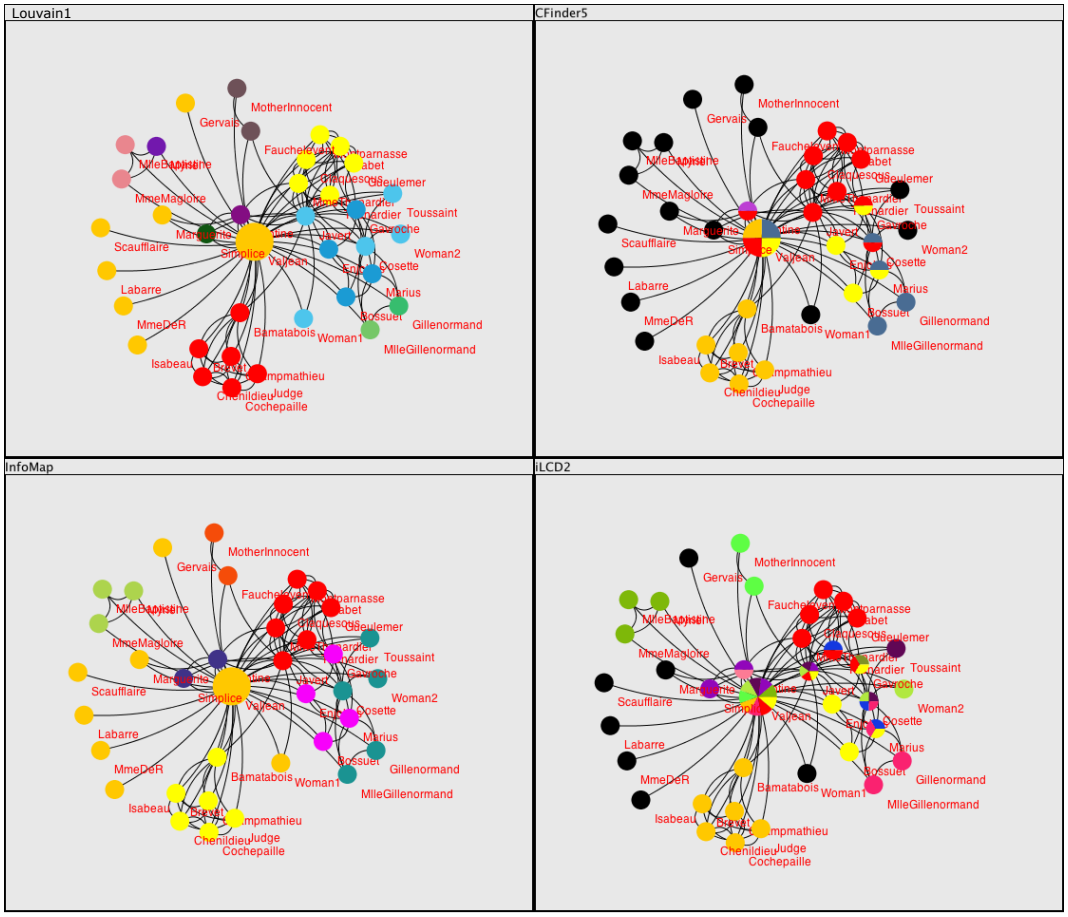


FIGURE 4.9 – *Réseau égocentré du nœud correspondant à Jean Valjean, dans le réseau des Misérables. On voit ici les inconvénients des différentes méthodes. Sans recouvrement, pour Louvain et InfoMap, le personnage appartient à une communauté qui ne parait pas être la plus pertinente, une communauté en étoile dont il est le hub. Pour CFinder et surtout iLCD, le problème est opposé, le nœud appartenant à de nombreuses communautés, ce qui pose le problème d'une complexité excessive du résultat, trop de recouvrement pouvant le rendre difficile à traiter.*

4.1.9 Bilan

Dans cette section, nous avons tout d'abord présenté une méthode, basée sur deux métriques, permettant de comparer les résultats obtenus par différentes algorithmes sur des réseaux de terrain. Ceci nous a permis de montrer que les communautés détectées par iLCD étaient intéressantes comparées aux approches existantes. Bien qu'étant dans la plupart des cas plus mal séparées du reste du réseau que les communautés classiques, elles sont en général plus denses. Le profil caractéristique des communautés d'iLCD sont donc de petites communautés denses, avec un recouvrement qui peut être important. Les résultats sur le réseau des personnages des Misérables a montré la pertinence de telles communautés dans des réseaux avec un recouvrement important. Les autres outils présentés nous ont permis de montrer les grandes différences qu'il peut y avoir entre des résultats produits par des algorithmes différents, et de montrer les forces et faiblesses d'iLCD : s'il n'est pas gêné par les communautés avec un fort recouvrement, le nombre de communautés différentes peut conduire à des résultats difficiles à exploiter. De même qu'il existe une balance entre communautés peu denses et communautés mal séparées, il existe un compromis entre un résultat détaillé et un résultat synthétique plus facilement exploitable.

4.2 Evaluation de la pertinence sémantique des communautés

Dans la définition des communautés que nous avons donnée dans le chapitre 3.2, nous présentons les communautés dans des réseaux de terrain comme des ensembles de nœuds qui sont « pertinents au niveau du sens ». Ce point me semble important car si la détection de communautés a autant d'applications, depuis la biologie jusqu'à la sociologie en passant par des utilisations grand public, c'est que les résultats obtenus correspondent à une réalité que l'humain peut généralement comprendre, mais qui n'apparaît pas naturellement par la seule connaissance du réseau.

Or, certains algorithmes sont testés extensivement sur des réseaux générés automatiquement, et semblent donner de bons résultats, mais n'ont parfois pas été validés sur des graphes de terrain, en-dehors des cas simples tels que le Zachary karaté club [Zac77] ou le réseau de personnages du livre « Les Misérables » [NG04].

Valider les résultats d'algorithmes sur des réseaux de terrain pose en effet un problème de taille : il est rare de disposer de la classification des nœuds en communautés pour de tels graphes, en dehors des cas simplistes tels que le Zachary karaté club. A notre connaissance, il n'existe pas de travaux majeurs visant, sur des réseaux de terrain, à comparer les résultats de différents algorithmes par rapport à une solution de référence connue venant du terrain. Le travail qui s'en approche le plus a été fait par Yang et Leskovec [YL12a]. Dans cet article, plusieurs grands réseaux sociaux Web 2.0 sont étudiés, et, pour chacun d'eux, une donnée existante du réseau est utilisée pour définir des communautés. Dans certains réseaux par exemple, des communautés sont explicitement créées par les utilisateurs. Un problème est, évidemment, que rien ne garantit que ces communautés soient fiables, ni qu'elles soient en rapport avec la topologie du réseau.

Un problème que l'on rencontrera de toute façon avec ce type de méthodes, est que l'on devra sélectionner un seul découpage en communautés, alors que, comme expliqué dans la section précédente, une décomposition unique et parfaite d'un réseau de terrain n'existe tout simplement pas. En conséquence, si l'on cherche à comparer les algorithmes avec une seule solution, quelle que soit la méthode employée pour l'obtenir, on aura un biais en faveur de l'un ou l'autre des algorithmes.

L'approche que nous avons adoptée est donc différente. Plutôt que d'essayer d'obtenir préalablement un ensemble de communautés considérées dans la suite comme optimales, nous avons

préféré commencer par effectuer, sur un même réseau, plusieurs détections de communautés avec des algorithmes reconnus, et, dans une seconde étape, demander à des utilisateurs d'évaluer la pertinence de ces communautés.

Il nous fallait tout d'abord trouver un domaine d'application. Nous voulions pouvoir tester sur de nombreux réseaux, pour éviter les cas particuliers. Il nous fallait aussi pouvoir faire évaluer les communautés par des utilisateurs qui seraient experts des réseaux à étudier. La solution que nous avons adoptée a été de travailler sur le réseau social Facebook.

Facebook est aujourd'hui, en 2013, la plateforme de réseau social la plus utilisée dans le monde, venant de dépasser le milliard d'utilisateurs réguliers. Facebook présente pour moi plusieurs intérêts :

- Les utilisateurs définissent une liste de contacts, appelés « amis » par le réseau social. Bien que la nature de ces liens ne soit en principe pas définie (famille, ami, relation professionnelle, accointance, etc.), ils permettent de définir les connexions d'un nœud du réseau (un individu) avec d'autres individus de ce réseau.
- D'autre part, Facebook propose une API facile à utiliser et permettant d'accéder à de multiples informations des utilisateurs, à condition que ces derniers en donnent l'autorisation.

L'idée de notre méthode était donc d'adapter la détection de communautés aux réseaux personnels des utilisateurs (réseaux égocentrés). En effet, les réseaux égocentrés des individus présentent naturellement une structure en clusters : pour un individu donné, ses contacts de l'université vont former un cluster, les personnes de sa famille un autre cluster, ses collègues de bureau un autre, etc. Comme dans la plupart des réseaux de terrain, les communautés rencontrées peuvent être recouvrantes : il est tout à fait possible qu'un collègue de travail fasse également partie de nos anciens camarades de l'université, par exemple.

Nous avons donc développé une application pour Facebook chargée de trouver ces communautés. Elle procède en trois étapes :

1. L'utilisateur qui accepte d'installer l'application lui permet de récupérer la liste de ses contacts ainsi que les liens d'amitié entre eux, ce qui permet de créer le réseau d'étude.
2. Plusieurs algorithmes de détection de communautés sont appliqués sur ce réseau d'étude.
3. L'utilisateur a la possibilité de consulter, via une interface adaptée, les différents résultats. On peut alors lui donner la possibilité de les évaluer et/ou de les comparer.

4.2.1 Travaux similaires

Il existe déjà, sur Facebook, des applications dont le rôle est de présenter à l'utilisateur ses contacts sous la forme d'un réseau, avec éventuellement une coloration des nœuds correspondant à des communautés. On peut notamment citer les applications « Friends Set [1] », « Friend Wheel [2] » et « TouchGraph [3] » .

Cependant, ces applications n'étant pas des travaux de recherche, on ne peut savoir quels algorithmes elles utilisent. Aucune d'entre elles ne propose de communautés avec recouvrement.

Le réseau Linked-In a également intégré un module permettant à ses membres de visualiser les communautés de leur réseau de collaborateurs. Ce module utilise l'algorithme de Louvain pour réaliser sa détection.

Il existe également, à notre connaissance, deux travaux de recherche que l'on peut rapprocher de celui-ci :

1. http ://apps.facebook.com/friendsets
2. http ://apps.facebook.com/friendwheel
3. http ://apps.facebook.com/touchgraph

– L'application « Fellows » [FCF+11] vise à tester un algorithme conçu par les auteurs sur les réseaux égocentrés des utilisateurs de Facebook. Les limites de ce travail sont que, d'une part, aucune comparaison n'est faite avec d'autres méthodes reconnues, et, d'autre part, aucune visualisation n'est fournie aux utilisateurs, les listes de noms étant proposées successivement aux utilisateurs qui doivent juger de leur pertinence, ce qui s'avère complexe dès lors que le nombre de contacts devient important.
– L'application [LVM+12], qui reprend également le même mécanisme, sur Facebook, mais, à nouveau, en utilisant un seul algorithme. Ce travail est également postérieur à celui que nous présentons ici.

4.2.2 Choix des algorithmes à comparer

Le choix des algorithmes à comparer était un premier critère important. Pour un utilisateur, évaluer la qualité d'un découpage en communautés et, plus encore, comparer différents résultats entre eux, est un procédé chronophage. Il nous fallait donc choisir un échantillon représentatif mais le plus limité possible. Nous avons donc sélectionné les algorithmes suivants :

– Composantes connexes, telles que définies en 1.2.
– InfoMap [RB08], décrit en détail dans l'état de l'art (2.2.2).
– CFinder [PDFV05], décrit en détail dans l'état de l'art (2.4.7).
– iLCD-NRMH, tel que défini en 3.3.3.
– Facebook Smart Lists, méthode propre à la plateforme Facebook, dont ce qui est connu est décrit ci-dessous.

Se reporter à l'état de l'art pour une présentation détaillée de ces méthodes.

Le choix de ces méthodes s'est fait sur les critères suivants :

– Nous voulions tout d'abord disposer d'une méthode qui servirait d'étalon, ou d'élément de comparaison. Proposer des communautés aléatoires n'ayant pas vraiment de sens, nous nous sommes orientés vers l'utilisation de composantes connexes. Sur un réseau égocentré, dans lequel on ne considère pas le nœud central (ego), il est fréquent de trouver plusieurs composantes connexes. Nous avons donc adopté cette méthode simple mais théoriquement pertinente pour vérifier que les résultats de la détection de communautés pouvaient être meilleurs.
– Nous voulions disposer d'une méthode considérée comme très efficace. InfoMap est souvent considéré comme l'une des meilleures méthodes existantes, nous l'avons donc sélectionnée malgré le fait qu'elle ne soit pas capable de gérer les recouvrements de communautés.
– Il nous fallait donc également considérer la meilleure méthode possible capable de gérer le recouvrement. Au moment où nous avons fait les tests utilisateurs, CFinder était la méthode la plus reconnue dans le domaine.
– iLCD était la troisième méthode utilisée. Les réseaux obtenus étant statiques, nous avons donné un ordre aléatoire aux liens. Comme dans le cas précédent, nous rappelons que cela peut affecter négativement les résultats, par rapport à une analyse dynamique.
– Le réseau social Facebook propose lui-même un outil pour trouver des groupes pertinents d'individus dans sa liste de contact, appelé Smart Lists. La méthode utilisée par Facebook n'est pas basée sur une analyse réseau, mais procède par recoupement d'informations à partir des profils des contacts de l'utilisateur. Il nous a semblé intéressant de comparer les résultats obtenus par les algorithmes de détection de communautés avec ceux de la méthode actuellement utilisée par Facebook.

4.2.3 Procédure d'évaluation de la pertinence des résultats

Les utilisateurs devaient comparer, sur leur propre réseau social, les communautés trouvées par les différentes méthodes, et évaluer leurs pertinences. Nous avons donc proposé deux types d'évaluations différentes : l'une par note et l'autre par classement.

4.2.3.1 Évaluation par note

Dans l'évaluation par note, il s'agit tout simplement d'attribuer à chaque résultat une note correspondant à sa pertinence. Pour ne pas laisser trop de place au caractère subjectif d'une telle notation, la signification de chaque note était explicitement décrite. Les différents choix possibles étaient :

- **1** : Solution incohérente et/ou incompréhensible
- **2** : Solution non convaincante, mais avec des éléments pertinents
- **3** : Résultat moyen
- **4** : La solution est logique et pertinente, avec de petites erreurs
- **5** : Le résultat est parfait

Une telle notation, quoique simpliste, permet d'avoir une idée de la qualité des résultats. Comme une note est attribuée à chaque algorithme, et que différents algorithmes peuvent avoir la même note, on peut voir si des résultats sont différents mais aussi pertinents les uns que les autres par exemple.

4.2.3.2 Évaluation par classement

Dans cette deuxième étape, les utilisateurs vont se contenter de classer les solutions de la meilleure à la moins bonne. Il y a au moins deux avantages à cette méthode par rapport à la précédente :

- Elle est moins sujette au problème de la variabilité des résultats d'un utilisateur à l'autre. La notion de « résultat plus pertinent » est la même quel que soit l'utilisateur, et il est donc plus facile de comparer les résultats donnés par des utilisateurs différents.
- Elle permet de départager des résultats proches. Par exemple, on pourrait concevoir que les utilisateurs attribuent la note 3 ou 4 à la plupart des algorithmes de détection de communautés. Dans ce cas, les obliger à classer les résultats permet de se rendre compte si l'un des algorithmes a tendance à être souvent considéré comme le meilleur.

En revanche, cette méthode présente un défaut important : les résultats n'étant que relatifs, on ne sait pas si la méthode classée comme la meilleure à été considérée comme convaincante par l'utilisateur. De plus, s'il y a de grands écarts entre l'efficacité des méthodes, ils peuvent ne pas être reflétés par un simple classement : la méthode classée deuxième peut-être à peine moins bonne que la première, alors que celle classée troisième sera presque complètement inefficace. Ces lacunes sont comblées par la première évaluation, à l'aide de notes.

4.2.4 Conditions de l'expérience

Afin que les utilisateurs soient en mesure de juger de manière efficace la qualité du découpage en communautés, nous nous sommes rendus compte qu'il était nécessaire de développer une interface graphique intuitive et simple à comprendre.

Afin que les utilisateurs ne soient pas influencés par l'aspect du réseau dans sa visualisation classique, ni qu'ils puissent comprendre le fonctionnement des algorithmes, nous avons décidé de nous abstraire de la visualisation classique du réseau, dans laquelle les nœuds sont positionnés,

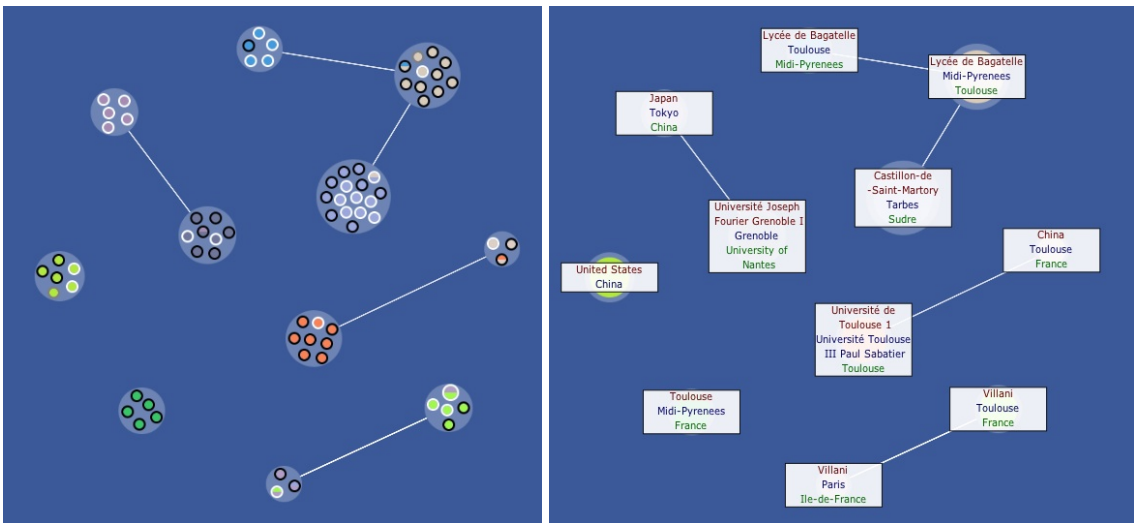


FIGURE 4.10 – *Captures d'écran de l'interface de l'application Facebook. A gauche, la vue présentant le détail par individu. A droite, la vue par communautés seulement. Les 3 mots clefs les plus pertinents sont associés à chaque communauté.*

par exemple, via un modèle masse-ressort. Nous avons préféré une mise en forme mettant en exergue les communautés. On peut observer dans la figure 4.10 cette interface.

Les communautés sont représentées de manière très claire par des cercles dont la taille représente le nombre d'individus qu'elles contiennent. Chaque individu est positionné dans le cercle correspondant à la communauté à laquelle il appartient. Dans le cas d'appartenances multiples, le nœud est placé dans l'une de ses communautés, choisie comme étant la plus pertinente. On définit la pertinence d'un nœud v pour une communauté C comme étant le ratio entre le nombre de nœuds de C liés à v et le nombre de nœuds de C

$$pertinence(v, C) = \frac{|\{u : u \in C, (u, v) \in E\}|}{n_C}$$

Chaque communauté se voit attribuée une couleur, et chaque nœud est ensuite coloré par la ou les couleurs correspondant à sa ou ses communautés. L'interface a ensuite été agrémentée de fonctionnalités visant à aider l'utilisateur à reconnaître les communautés et à juger de leur pertinence :

- Le survol d'un nœud fait apparaître le nom, la photo et quelques informations de l'individu qu'il représente.
- Un cercle de couleur est ajouté autour de chaque individu, noir pour les hommes et blanc pour les femmes.
- Une option a été ajoutée pour proposer un nom à chacune des communautés.

Ce processus de nommage est effectué de la manière suivante : pour chaque individu, on obtient de son profil toutes les informations qui peuvent être recoupées : nom de famille, universités fréquentées, employeurs, lieux de résidence et de naissance, etc. A chaque communauté C, nous associons la liste des termes liés t_C.

Ensuite, pour chaque communauté et pour chaque terme, nous calculons le ratio représentant la fréquence de l'apparition du terme t dans la communauté C :

$$FrequenceCommunaute(t, C) = \frac{count(t, t_C)}{|t_C|}$$

où $count(t, t_C)$ correspond au nombre de fois où t apparaît dans les termes de la communauté C. Nous calculons de la même manière la fréquence d'apparition du terme pour la totalité des individus du réseau

$$FrequenceReseau(t) = \frac{count(t, t_G)}{|t_G|}$$

où t_G correspond à l'ensemble des termes apparaissant pour tous les individus. Enfin, le score de pertinence de ce terme pour cette communauté est calculé par :

$$Pertinence(t, C) = \frac{FrequenceCommunaute(t, C)}{FrequenceReseau(t)}$$

L'idée de cette formule est qu'un terme apparaissant fréquemment dans une communauté sera pertinent pour cette communauté. Si, en plus, ce terme apparaît principalement dans cette communauté, sa pertinence sera augmentée. Si, au contraire, ce terme a tendance à beaucoup apparaître dans le réseau, cela diminuera sa pertinence.

Par exemple, pour une personne française, il est probable que le terme France apparaisse dans le profil d'une majorité de ses contacts. Ce terme ne sera donc que rarement jugé très pertinent. Au contraire, s'il a un petit nombre de contacts anglais par exemple, et que ces individus apparaissent dans un même groupe, le terme Angleterre sera jugé très pertinent pour le caractériser.

Afin de ne pas perdre l'utilisateur dans une trop grande quantité d'information, nous caractérisons chaque communauté par les 3 termes jugés les plus pertinents. (Figure 4.10)

Le déroulement de l'expérience était le suivant : tout d'abord, chaque utilisateur se connectait sur son profil Facebook et ouvrait l'application. Une première solution s'affichait alors, et nous lui expliquions comment utiliser l'interface. Une fois familiarisé, nous lui montrions comment appliquer des algorithmes différents et obtenir des résultats différents. Les différentes solutions étaient labellisées Résultat1, Résultat2, . . . , Résultat5 de manière à ne pas risquer de l' influencer. Les utilisateurs inscrivaient finalement leurs conclusions sur un questionnaire papier.

4.2.5 Résultats

Nous avons sélectionné un panel d'utilisateurs, en essayant de prendre des personnes aux profils variés afin d'obtenir tous types de réseaux.

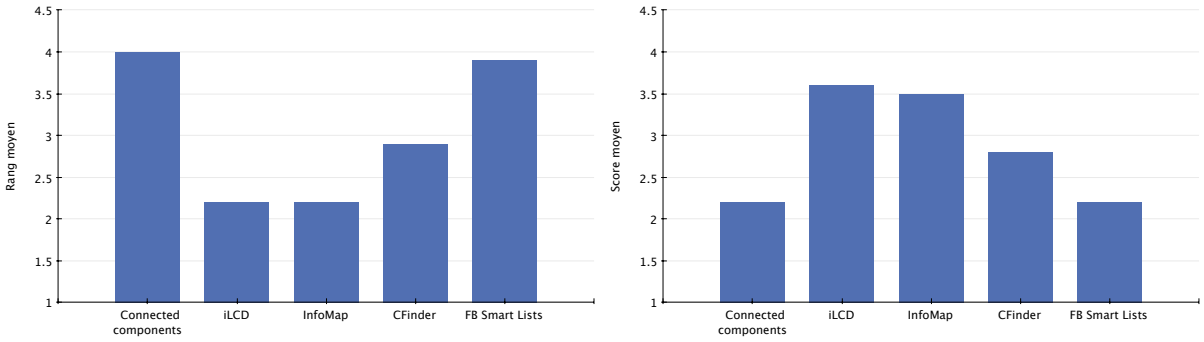
Bien que l'application Facebook permettant de tester les différents algorithmes soit disponible en ligne, nous n'avons pas jugé pertinent de considérer dans nos résultats des expériences faites par des utilisateurs inconnus. Nous avons préféré faire passer le test à des utilisateurs que nous pouvions observer, afin de nous assurer qu'ils ne rencontraient pas de problèmes techniques et qu'ils comprenaient bien ce qui leur était demandé.

La procédure étant relativement longue (généralement entre 20 et 30 minutes), nous avons dû nous restreindre à une population limitée, bien que suffisante pour faire apparaître des tendances claires : 30 utilisateurs ont participé.

50% des individus étaient des hommes, 50% des femmes. 60% des individus avaient entre 20 et 30 ans, 10% moins de 20 ans et 30% au-dessus de 30 ans. La figure 4.12 représente la répartition des utilisateurs en terme de nombre de contacts.

Si l'on regarde les profils des utilisateurs de Facebook, par exemple dans [Har07], on pourra se rendre compte que notre panel n'est pas parfait, mais s'approche cependant d'un échantillon représentatif.

Sur la figure 4.11, on peut observer le classement moyen ainsi que le score moyen des différents algorithmes. Les deux graphes semblent mener aux mêmes conclusions :



(a) *Classements moyens* (b) *Scores moyens*

FIGURE 4.11 – *Moyennes des appréciations des utilisateurs sur la qualité des communautés.*

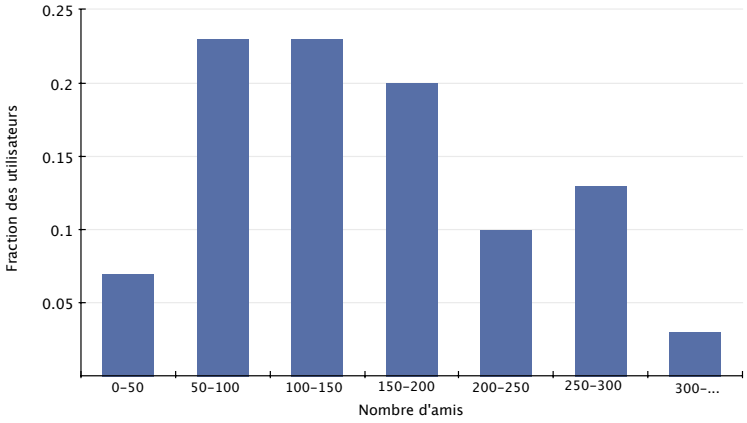


FIGURE 4.12 – *Répartition des testeurs en terme de nombre de contacts*

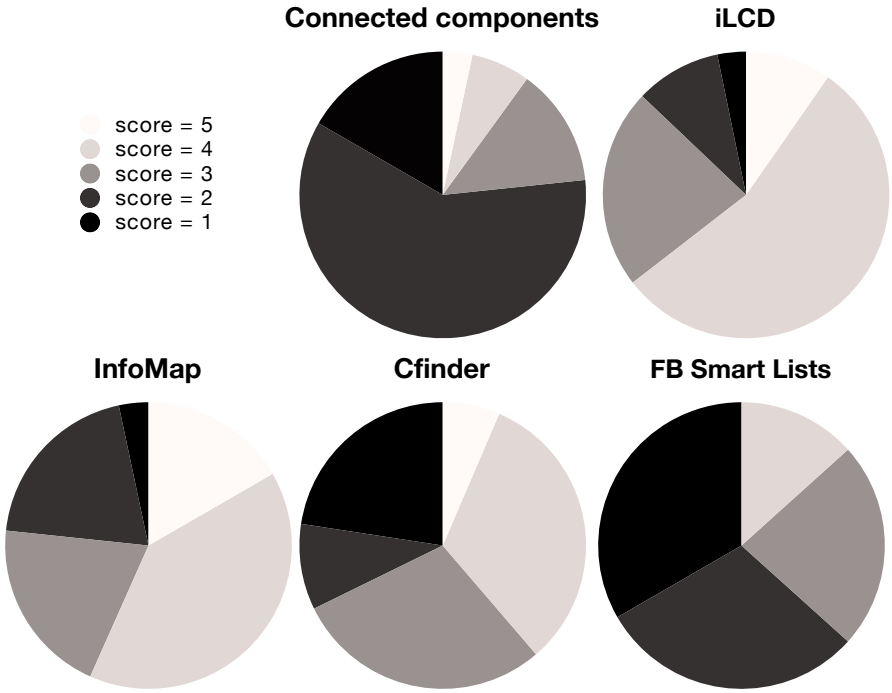


FIGURE 4.13 – *Distribution des notes pour chaque algorithme. Les couleurs les plus claires sont les meilleurs notes.*

- L'algorithme par composantes connexes est, comme prévu, le moins efficace. Avec une note moyenne de 2,2, il est considéré par la plupart des utilisateurs comme non pertinent. Les résultats ne sont pas incohérents mais, dans la plupart des cas, la majorité des contacts de l'utilisateur sont agrégés en une seule et même communauté, avec quelques utilisateurs formant de petits groupes séparés.
- Les résultats fournis par les Smart Lists de Facebook ont constitué une surprise : en moyenne, les résultats n'étaient pas meilleurs. En fait, si l'on ne prend que les utilisateurs pour lesquels aucune liste n'avait été définie manuellement auparavant, la note moyenne était de 1,78 soit une note pire que celle donnée par les composantes connexes. 40% de ces utilisateurs ont attribués une note de 1, correspondant à des communautés incohérentes. Nous discuterons des causes et conséquences de ce résultat surprenant dans la section discussion de ce chapitre.
- iLCD et Infomap ont des scores équivalents, que ce soit pour le score ou le classement. Le score moyen pour ces algorithmes est d'approximativement 3,5, ce qui est clairement meilleur que les autres. La population du panel d'utilisateur était trop faible pour que les différences entre ces deux méthodes soient significatives. En regardant dans le détail, certaines disparités apparaissent cependant, que nous allons exposer un peu plus loin.
- Les résultats de CFinder sont plus ou moins à mi-chemin entre ceux des méthodes les plus efficaces et ceux des autres méthodes. Cela montre que, certes, les algorithmes de détection de communauté fournissement un résultat plus pertinent que des approches plus simples, mais également que tous ne se valent pas, certains algorithmes pouvant être jugés plus pertinents par les utilisateurs.

Nous avons ensuite décidé de regarder la répartition des scores pour chaque algorithme. En

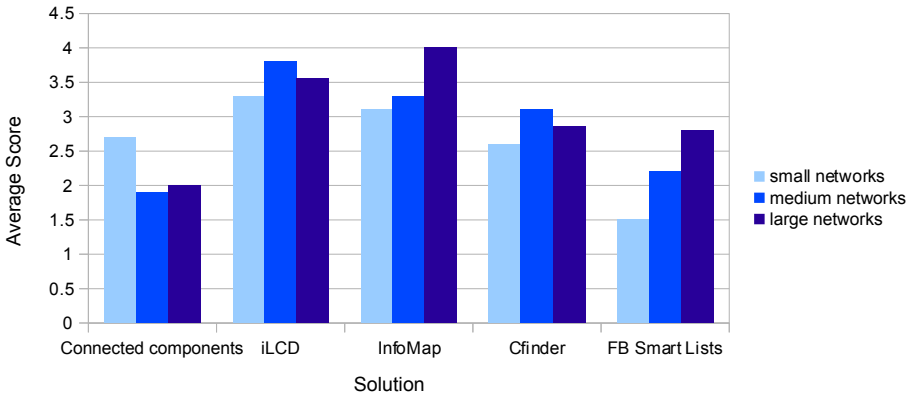


FIGURE 4.14 – *Corrélation entre la taille des communautés et l'efficacité de chaque algorithme. On n'observe pas de corrélation évidente.*

effet, se contenter de comparer des moyennes peut parfois être trompeur. La figure 4.13 représente donc la distribution, pour chacun des algorithmes, des notes qui lui ont été attribuées.

La méthode par composante connexe a une forte majorité de scores égaux à 2. Cela paraît logique, car les résultats ne sont pas incohérents, mais manquent fortement de précision. Pour iLCD et InfoMap, plus de 60% des scores sont des 4 ou des 5, ce qui correspond à des résultats pertinents (On peut rappeler qu'un score de 4 signifie « La solution est logique et pertinente, avec de petites erreurs »). Si InfoMap a obtenu un plus grand nombre de 5 — score correspondant à un résultat parfait — iLCD a obtenu moins de scores correspondants à un mauvais découpage, c'est à dire inférieurs à 3. L'un des résultats les plus intéressants est fournit par CFinder. Bien qu'il ait clairement beaucoup plus de résultats positifs que la solution des composantes connexes, il a cependant reçu plus de 1 que cette méthode, c'est à dire des résultats considérés comme incohérents par les utilisateurs. Nous avons effectivement observé que parfois, sur certains profils, l'algorithme semblait en échec. Il avait tendance à diviser de manière incohérente des communautés qui ne l'étaient par aucun autre résultat, ou au contraire rassemblait des individus qui n'auraient clairement pas dû l'être.

Finalement, nous avons cherché à voir si certains algorithmes étaient plus efficaces que d'autres sur certains types de réseaux. Tout d'abord, nous avons observé que les deux meilleurs algorithmes avaient un ratio de bons scores (scores supérieurs à 3) autour de 60%. Nous nous sommes demandés si ces algorithmes étaient performants sur les mêmes réseaux ou non.

90% des utilisateurs ayant attribué au moins une fois une note de 4 ou plus, nous pouvons déduire qu'il n'y a pas une distinction entre des réseaux simples, sur lesquels tous les algorithmes seraient efficaces, et des réseaux complexes, n'étant efficacement analysés par aucun. Dans 90% des cas, au moins l'un des algorithmes a donné un bon résultat, mais, souvent, un seul des algorithmes testé l'a fait.

Nous aurions aimé étudier quelles propriétés des réseaux avantageaient l'un ou l'autre des algorithmes. Cependant, pour des problèmes de droits et de confidentialité, nous n'avons pas pu conserver les réseaux des différents utilisateurs. La seule donnée dont nous disposions était le nombre de nœuds dans chaque réseau. La figure 4.14 montre cependant qu'il n'y a pas de corrélation évidente entre la taille du réseau et l'efficacité des différents algorithmes.

4.2.6 Discussion sur l'application de la détection de communautés à la formation de groupes

Si ce travail permettait de valider empiriquement la pertinence sémantique des communautés trouvées avec iLCD, il s'agissait également d'une expérience portant sur l'intérêt que pouvait avoir la détection de communautés appliquée à la détection de groupes pertinents d'individus dans un réseau social Web 2.0. Pour ce travail, nous avons collaboré avec une sociologue, Maud Leguistin, qui travaille sur les réseaux sociaux Web 2.0.

Si l'utilisation massive des réseaux sociaux de type Facebook a créé de nombreux usages, elle a également posé certains problèmes. L'un des plus cités est celui du respect de la vie privée. Les sociologues expliquent [Gof05, Lég12], que dans les relations habituelles, nous avons différents groupes de connaissance, et ces groupes sont remarquablement imperméables : les membres de notre famille ne rencontrent pour ainsi dire jamais nos collègues de bureau, qui eux-mêmes ne connaissent pas nos amis du lycée, qui, à leur tour, n'ont aucun contact avec les membres de notre club de sport. Il existe bien sûr quelques exceptions, des individus qui peuvent appartenir à plusieurs groupes, mais cela ne constitue pas la règle.

Sur les réseaux de type Facebook en revanche, c'est tout le contraire. Par défaut, tous les individus font partie de la même liste, et sont traités de manière identique. Cela provoque de nombreuses collisions, qui peuvent être dommageables à l'utilisateur.

Ce problème est maintenant assez connu, et un réseau plus récent, Google+, a mis l'accent sur une segmentation plus forte des contacts, via l'utilisation des « Cercles », c'est à dire des groupes de contacts avec lesquels on peut communiquer de manière exclusive, sans que d'autres contacts ne puissent voir ce qui leur est dit.

Facebook propose également des outils similaires, bien qu'ils soient moins connus et utilisés.

Peu avant que nous ne fassions notre expérimentation, Facebook a mis en ligne un nouvel outil visant à créer automatiquement des listes de contact, en fonction des informations de profil de ces derniers. On peut en effet supposer que la raison pour laquelle ces listes, que l'on peut créer manuellement, ne sont que peu utilisées, est que le processus de leur création et de leur maintenance est non intuitif, voire fastidieux.

L'autre objectif de cette expérience était donc d'évaluer l'intérêt que la création automatique de groupes de contact suscitait ou non chez les utilisateurs, ainsi que l'efficacité pratique de la méthode par détection de communautés.

Les résultats présentés dans la section précédente montrent déjà que les utilisateurs jugent les résultats obtenus par détection de communautés comme pertinents. Nous avons également demandé aux utilisateurs si les noms attribués aux communautés étaient significatifs. Des noms de communautés pertinents sont très positifs, puisque cela veut dire que, d'une part, l'utilisateur comprend bien à quoi correspond le groupe sémantiquement et, d'autre part, que le mécanisme de nommage donne de bons résultats.

Nous avons demandé aux utilisateurs de calculer, pour le résultat qu'ils considéraient comme le meilleur, le pourcentage de communautés trouvées pour lesquelles les noms étaient significatifs. Les résultats sont présentés dans le graphique 4.15. Nous avons créé trois catégories, nommées Bon, Correct et Faible, correspondant respectivement à des pourcentages de bonnes communautés de 80%-100%, 50%-80% et 0%-50%.

Comme on peut le voir, près de la moitié des sujets estiment que plus de 80% des communautés sont correctement nommées, ce qui signifie qu'elles correspondent bien à un groupe sémantiquement valide. Parmi les groupes mal nommés, beaucoup d'entre eux sont des groupes de peu d'individus qui n'ont pas rempli extensivement leurs informations de profil. De plus, le nom de certains groupes ne peut pas être déduit à partir des informations de profil.

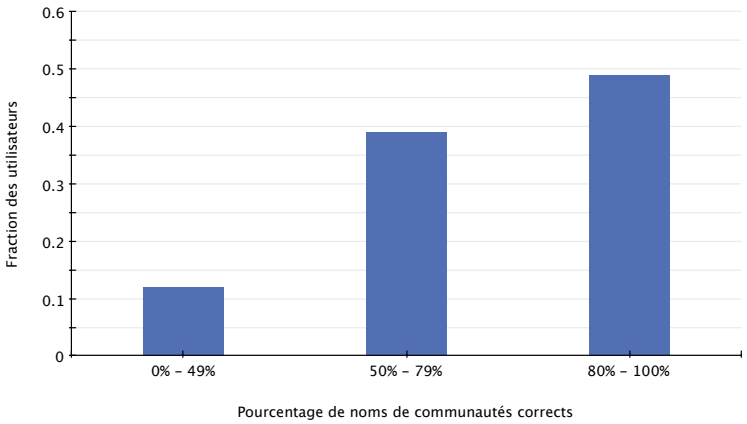


FIGURE 4.15 – *Evaluation de la qualité du nommage automatique des communautés. On observe que près de la moitié des utilisateurs jugent les noms très pertinents (plus de 80% des noms corrects), et près de 90% des utilisateurs jugent que plus de la moitié des noms sont corrects.*

Si par exemple un individu est inscrit dans un club d'échecs, ses contacts faisant partis du même club seront probablement regroupés ensemble, mais l'application ne pourra trouver le lien à partir des informations de profil. La communauté sera donc probablement nommée du nom de la ville où est situé le club, si les autres membres y résident ou y travaillent, mais les utilisateurs ne jugeront pas un tel nommage pertinent, puisqu'ils savent bien que ce n'est pas cet élément qui constitue vraiment le lien entre ces individus.

La pertinence de ces résultats et leur non-trivialité ont été illustrées par les réactions des sujets. Comme expliqué précédemment, les algorithmes utilisés n'étaient ni labellisés ni expliqués. Plusieurs utilisateurs se sont donc fait une fausse idée sur leur fonctionnement, et ont pensé que les communautés étaient formées à partir des informations de profil. Ils ont alors remarqué qu'une personne avec qui, par exemple, ils étaient au lycée n'apparaissait pas dans la communauté correspondant aux camarades de lycée. Ils montraient le profil de la personne et montraient que pourtant, cette information était bien indiquée dans son profil.

Ils interprétaient ce résultat comme un bogue, puisqu'ils pensaient que les communautés étaient simplement constituées en recoupant les informations de profil. Il est intéressant de noter que l'une des méthodes fonctionne pourtant exactement de cette manière : les Smart Lists de Facebook. Pourtant, ces mêmes utilisateurs jugeaient ces listes automatiques comme illogiques et non pertinentes.

En effet, cette méthode qui semble intuitivement pertinente, se révèle dans les faits inefficace. Si l'on créait par exemple une communauté des personnes ayant été à la même université que l'utilisateur, elle contiendrait certes les personnes qui devraient y être, à condition qu'elles aient remplies leur profil correctement (dans les faits, beaucoup d'utilisateurs ne complètent pas correctement leurs informations de profil, par omission ou même, souvent, volontairement, par souci de respect de la vie privée). Le problème est cependant qu'en général, plusieurs utilisateurs sont ajoutés à la communauté alors qu'ils n'appartiennent pas du tout au même groupe. Il s'agit tout simplement de tous les individus qui sont allés à la même université, mais qui ne sont pas en contact avec l'utilisateur pour cette raison. Ils ont pu y aller à une toute autre période, ou simplement ne pas avoir été en contact à ce moment là. Ce type de problème se rencontre presque systématiquement et explique pourquoi les utilisateurs considèrent non pertinents les résultats fournis par une méthode utilisant ce mécanisme de recoupement.

Il est donc certain que l'utilisation de mécanismes réseaux, et en particulier de la détection de communautés, puisse se révéler utile pour assister les utilisateurs de plateformes Web 2.0.

Aspects dynamiques Pour finir, nous pouvons faire remarquer que l'utilisation d'un algorithme dynamique pourrait, dans ce cas d'utilisation, se révéler pertinent. En effet, les listes de contact des utilisateurs de Facebook évoluent en permanence. Ceux-ci ajoutent de nouveaux contacts, en retirent, sans compter que des liens peuvent également se créer ou disparaître entre des contacts d'un utilisateur. Un algorithme de détection de communautés dynamiques serait donc capable de tirer parti de ces informations pour proposer à l'utilisateur des modifications pertinentes de ses groupes, en particulier lors de l'ajout de nouveaux « amis ». Si un algorithme statique pourrait éventuellement être utilisé, le problème est qu'à cause de l'instabilité, les groupes existants à l'étape précédente risquent de ne pas se retrouver à l'étape suivante.

L'utilisation d'un algorithme dynamique permettrait aussi de prendre en compte les préférences de l'utilisateur. Par exemple, si lors de la première détection, l'utilisateur n'est pas entièrement satisfait du résultat, il peut modifier manuellement les communautés trouvées pour qu'elles soient plus pertinentes. L'algorithme dynamique peut alors partir de ces communautés modifiées lorsque la prochaine modification du réseau entrainera un recalcul des communautés.

4.2.7 Bilan

En conclusion de cette partie, nous avons montré que les communautés trouvées par iLCD-NRMH sur ce type de réseaux de terrain étaient particulièrement pertinentes. Non seulement les utilisateurs jugent que les résultats fournis sont pertinents, mais, comparés avec d'autres algorithmes connus du domaine, les résultats sont aussi bons, voire meilleurs. La conclusion selon laquelle aucun algorithme n'est le meilleur dans tous les cas, mais qu'au contraire les solutions se complètent semble corroborer l'intuition selon laquelle chercher un algorithme qui serait le meilleur pour tout type de communauté dans tout type de réseau est une utopie.

4.3 Bilan du chapitre

Dans ce chapitre, nous avons montré, par deux approches différentes, que les communautés trouvées par notre méthode étaient comparables avec celles que pouvaient trouver des algorithmes statiques sur des réseaux statiques. Dans le prochain chapitre, nous allons donc nous consacrer à valider les aspects dynamiques de notre algorithme, en l'appliquant à des réseaux réels et en validant les résultats trouvés.

5

Applications de la détection de communautés dynamiques à des cas réels

Contents

Résumé du chapitre Nous avons consacré le chapitre précédent à montrer que les communautés trouvées par iLCD-NRMH sur des réseaux statiques étaient pertinentes. Dans ce dernier chapitre, nous allons donc valider l'aspect dynamique de notre algorithme, en l'appliquant à des réseaux temporels de terrain. Nous présentons deux cas, qui sont complémentaires : le premier est un réseau de petite taille, ayant déjà été étudié, sur lequel nous pouvons valider en détail toutes les communautés. L'autre est un grand réseau issu du Web 2.0, ce qui nous permet de valider le passage à l'échelle ainsi que l'intérêt de notre méthode sur un cas réel nouveau.

Le premier travail, en section 1, porte sur un graphe pour lequel nous avons des observations faites plusieurs fois par semaine, sur une période de plus de 20 ans. Il s'agit d'un réseau d'isards, des animaux au comportement social. Après avoir présenté la méthode utilisée pour transformer les données d'origine en un réseau temporel, nous présenterons les résultats obtenus, et les validerons, notamment en les comparant à des résultats obtenus précédemment dans une autre étude.

Dans la section 2, nous prendrons un autre jeu de données, constitué d'un réseau de termes issu d'une plateforme japonaise de partage de vidéos appelée Nico Nico Douga. Nous montrerons comment, en appliquant iLCD-NRMH à ces données, il est possible de mettre en évidence les sujets populaires sur la plateforme. Nous détaillerons les résultats, et, dans le cas des sujets populaires de courte durée, nous pourrons valider les périodes auxquelles ces communautés sont détectées en les comparant aux dates auxquelles ces événements ont eut lieu au Japon (dates de sorties dans le commerce de jeux vidéos).

Enfin, un point important de la conclusion de ce chapitre sera la mise en évidence d'un phénomène pouvant apparaître lors de l'évolution de communautés dynamiques : la résurgence.

5.1 Introduction

Si la détection de communautés a rencontré la popularité qu'on lui connaît aujourd'hui, c'est notamment par le grand nombre d'applications qu'il est possible d'en faire. Nous avons déjà évoqué nombre d'exemples dans les chapitres précédents. On peut en particulier citer deux applications particulièrement importantes : la première est l'application aux réseaux sociaux, dans le sens original du terme, c'est à dire à des réseaux d'individus tels qu'ils peuvent être collectés par des sociologues. A l'origine, beaucoup de travaux sur la détection de communautés portaient sur ce type de réseaux. Le terme de détection de communautés, introduit par Girvan et Newman, vient d'ailleurs de l'application à de tels réseaux. Les réseaux étudiés dans ce cas sont généralement des réseaux de terrain, comportant des communautés assez complexes, mais dans des réseaux de petites tailles.

Les autres réseaux tenant une place particulière sont les grands réseaux issus du web 2.0, c'est à dire des réseaux constitués par les actions conjointes de milliers —voire millions— d'utilisateurs. On peut facilement donner nombre d'exemples de ces réseaux : Facebook, Linked-In ou Google+ bien sûr, qui sont des plateformes de réseaux sociaux Web 2.0, mais également des sites plus divers tels que Twitter, YouTube, Wikipedia, les réseaux de blogs, etc. Ces réseaux, de par leur taille et leur quantité d'information, ont relancé l'intérêt de la détection de communautés, en lui imposant de nouvelles contraintes. Aujourd'hui, avec l'arrivée des algorithmes de détection de communautés dynamiques, il est possible de reprendre ces deux types de réseaux, et de les traiter d'une nouvelle manière pour en tirer de nouvelles informations. C'est ce que nous avons essayé de faire durant cette thèse, une fois que l'efficacité de l'algorithme développé a été démontrée sur des réseaux statiques.

Nous avons essayé de prendre des réseaux issus de cas réels, pour lesquels nous disposions

des données de leur évolution, et de les étudier à la lumière des résultats fournis par iLCD.

Le premier de ces réseaux est donc de petite taille, et ressemble aux réseaux issus de la sociologie utilisés dans les premiers temps. Il s'agit cependant d'un réseau venant de l'éthologie, modélisant l'évolution des interactions entre les individus d'une population d'isards sur une période de plusieurs années.

Le second réseau que nous avons extensivement étudié est issu d'une plateforme japonaise de publication de vidéos, de type YouTube, appelé Nico Nico Douga. L'intérêt de ce réseau est qu'il est vaste, et que nous disposons des informations complètes sur toutes les vidéos publiées par les utilisateurs sur une période de 3 ans. Ce travail a été publié dans le journal Social Network Analysis and Mining, avec Frédéric Amblard, Hideaki Takeda et Masahiro Hamasaki [CTHA12].

Nous avons choisis de présenter ici ces deux résultats, que nous pensons complémentaires. Le premier est un réseau de petite taille ayant déjà été étudié, et pour lequel nous travaillons en collaboration avec des experts du domaine de ce réseau. La petite taille nous permet d'étudier les résultats en détail. Le fait que le réseau ait déjà été étudié et que des experts travaillent avec nous permet également de valider les résultats dans leur globalité. En revanche, le réseau étudié est relativement simple et de petite taille. C'est pourquoi nous avons fait une deuxième application, portant elle sur un grand réseau issu issue du Web 2.0. Étant donné la masse de données, il n'est plus possible de valider exhaustivement chaque communauté. En revanche, on vérifiera qu'iLCD est applicable a ce cas réel complexe, et qu'il permet d'en extraire des informations utiles, tout en étant capable de passer à l'échelle.

L'analyse approfondie des résultats d'iLCD-NRMH sur ces deux réseaux nous a également permis de valider la pertinence de l'aspect dynamique de notre algorithme. En effet, s'il existe à l'heure actuelle des générateurs de réseaux avec communautés qui, sans être tout à fait convaincants, fournissent déjà un support crédible pour la comparaison d'algorithmes, il n'existe à notre connaissance rien de tel pour des réseaux temporels. S'il serait en effet possible de faire évoluer des réseaux statiques pour produire des réseaux dynamiques, en l'état actuel des connaissances, la façon dont on ferait évoluer ces réseaux serait certainement bien loin de la façon dont des réseaux de terrain évoluent, et tester des algorithmes de détection de communautés dynamiques sur ces réseaux n'aurait pas beaucoup de sens. C'est pourquoi, jusqu'ici, nous avons préféré la solution expérimentale consistant à travailler sur des réseaux issus de données réelles et à essayer de valider — ou invalider — les communautés en se basant sur la sémantique des réseaux ou sur des résultats antérieurs.

5.2 Analyse d'un réseau d'isards

Dans cette première application, nous nous sommes intéressés à un réseau de petite taille mais ayant un grand avantage : nous disposons de données le concernant sur une période très longue. Ce réseau est issu de recherches faites par des éthologues sur une population d'isards vivant dans les Pyrénées (montagnes du sud de la France). Les isards (5.1) sont des animaux proches des chamois, dont les femelles en particulier ont un comportement social. La population étudiée est une population vivant en liberté dans le parc national des Pyrénées, plus précisément dans la zone de Cauterets.

5.2.1 Création du jeu de données

Les données dont nous disposons couvrent la période de 1992 à 2008, soit une période de 16 ans. La méthode utilisée pour la collecte de ce jeu de données, qui est une méthode standard, est basée sur le marquage visuel et l'observation. Concrètement, des animaux sont capturés

FIGURE 5.1 – *Photographie d'un isard, Rupicapra pyrenaica*

de manière aléatoire, et sont ensuite identifiés de manière unique, au moyen de colliers et de marqueurs posés sur les oreilles. Par la suite, des observateurs effectuent, plusieurs fois par semaine, des sorties d'observation, durant lesquelles ils notent tous les individus observés ainsi que leurs positions.

5.2.1.1 Périodes hivernales et périodes estivales

Les isards ont un comportement très différent entre l'hiver et l'été. En hiver, la présence d'importantes quantités de neige et la difficulté à se nourrir obligent les individus à rester cantonnés à une région particulière, de petite taille, appelée le Clot Cayan. Durant cette période, tous les individus se rassemblent donc dans cette zone. En été au contraire, les individus peuvent couvrir une zone beaucoup plus étendue 5.2. Nous avons donc traité différemment ces deux périodes.

5.2.1.2 Différences Mâles/Femelles

Les comportements des mâles et ceux des femelles sont très différents. Alors que les femelles sont connues pour leur comportement social, formant des hardes (groupes d'isards) et n'étant que rarement observées seules, les mâles ont souvent tendance à être solitaires, en dehors de la période de reproduction (Octobre/Novembre), durant laquelle le mâle rend visite à des femelles, ce qui conduit à des confrontations avec d'autres mâles (rarement observées, étant donné la méthode de collecte). Dans la suite, comme recommandé par les éthologues, nous travaillons principalement sur les femelles, bien que quelques remarques pourront également être faites sur les mâles, puisque nous disposons aussi des données les concernant.

5.2.1.3 Méthode de positionnement des positions des individus

La méthode utilisée pour identifier la position des individus est importante, puisqu'elle a un effet sur les liens que nous créons entre les individus. Deux grilles ont été définies pour découper la zone, une pour le Clot Cayan (zone hivernale), pour laquelle les carrés sont de 50 m x 50 m, et une autre pour le reste de la carte, dont les carrés sont de 250 m x 250 m. Tout individu vu dans

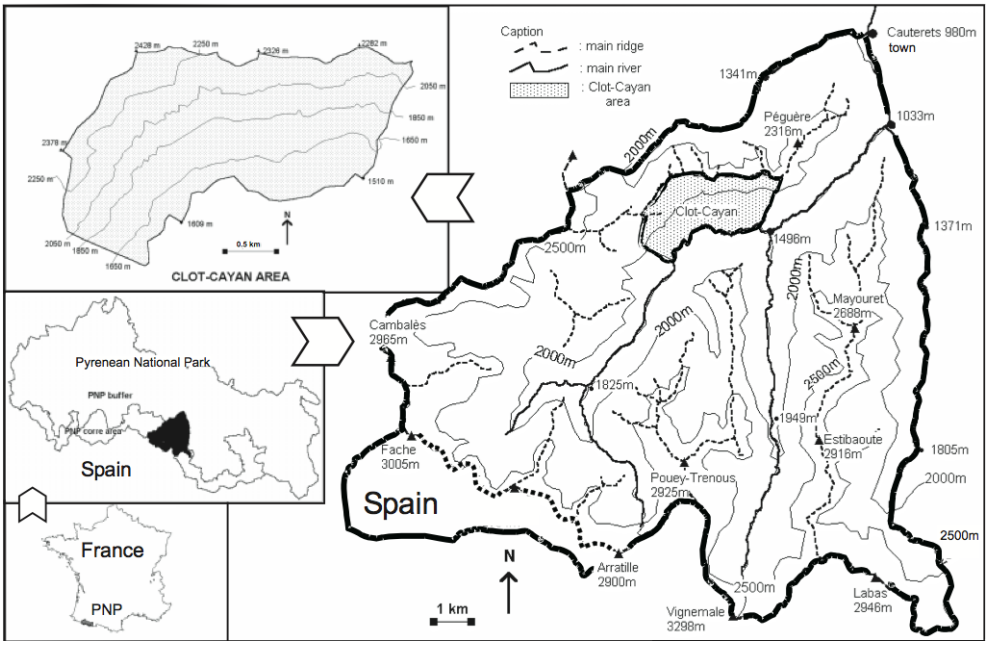


FIGURE 5.2 – *Illustration de la zone de déplacement des isards. A droite, grande carte représentant la zone accessible par les individus en été. En haut à gauche, détail de Clot Cayan, la zone dans laquelle les animaux se trouvent en hiver.*

l'un de ces carrés virtuels est donc identifié comme étant situé dans celui-ci. En conséquence, si plusieurs individus sont dans le même carré, nous les considérons comme localisés au même emplacement, et donc potentiellement en interaction.

5.2.1.4 Définition des interactions entre individus

Comme nous souhaitons construire un réseau social dynamique de ces individus, nous souhaitons savoir quels individus sont en interaction avec quels autres et à quels moments. Le problème consiste donc à définir ce à quoi correspond une interaction pour des isards. Dans le cas présent, en l'absence d'autres données, notre critère d'interaction est purement spatial : si deux individus sont observés dans le même carré au même moment, alors ils sont en interaction ce jour là (Un individu peut être observé au maximum une fois au cours d'une même journée).

5.2.1.5 Incomplétude des données

Il est évident que les données obtenues ne sont que fragmentaires : les observations n'ont pas été menées tous les jours, ce qui signifie qu'il existe des jours sans aucune observation. Lorsqu'une sortie d'observation est menée, seule une fraction de la population marquée est observée, il est donc rare que l'on connaisse la position de tous les individus pour un jour donné.

Tous les individus de la population n'ont pas ailleurs pas pu être marqués. Par exemple, les jeunes ne peuvent pas être capturés pour identification dès leur naissance, et d'autres individus n'ont tout simplement jamais été capturés. 119 individus ont été identifiés au cours de la période d'étude, ce qui couvre, selon le moment, entre 18% et 41% de la population totale estimée.

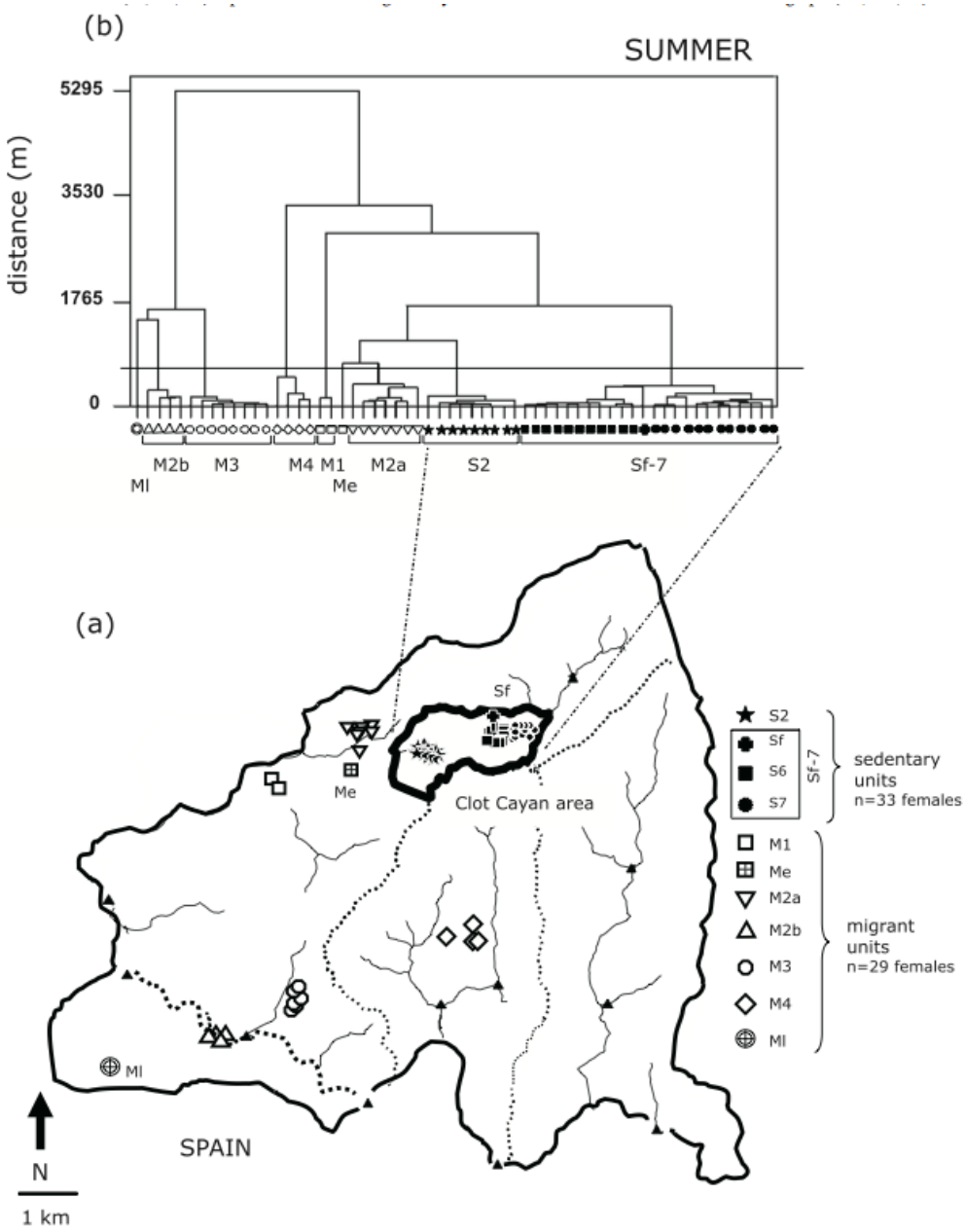


FIGURE 5.3 – *Clusters trouvés par la méthode statique des barycentres en été. En haut, le dendrograme de clustering. En bas, les positions correspondantes des barycentres des individus.*

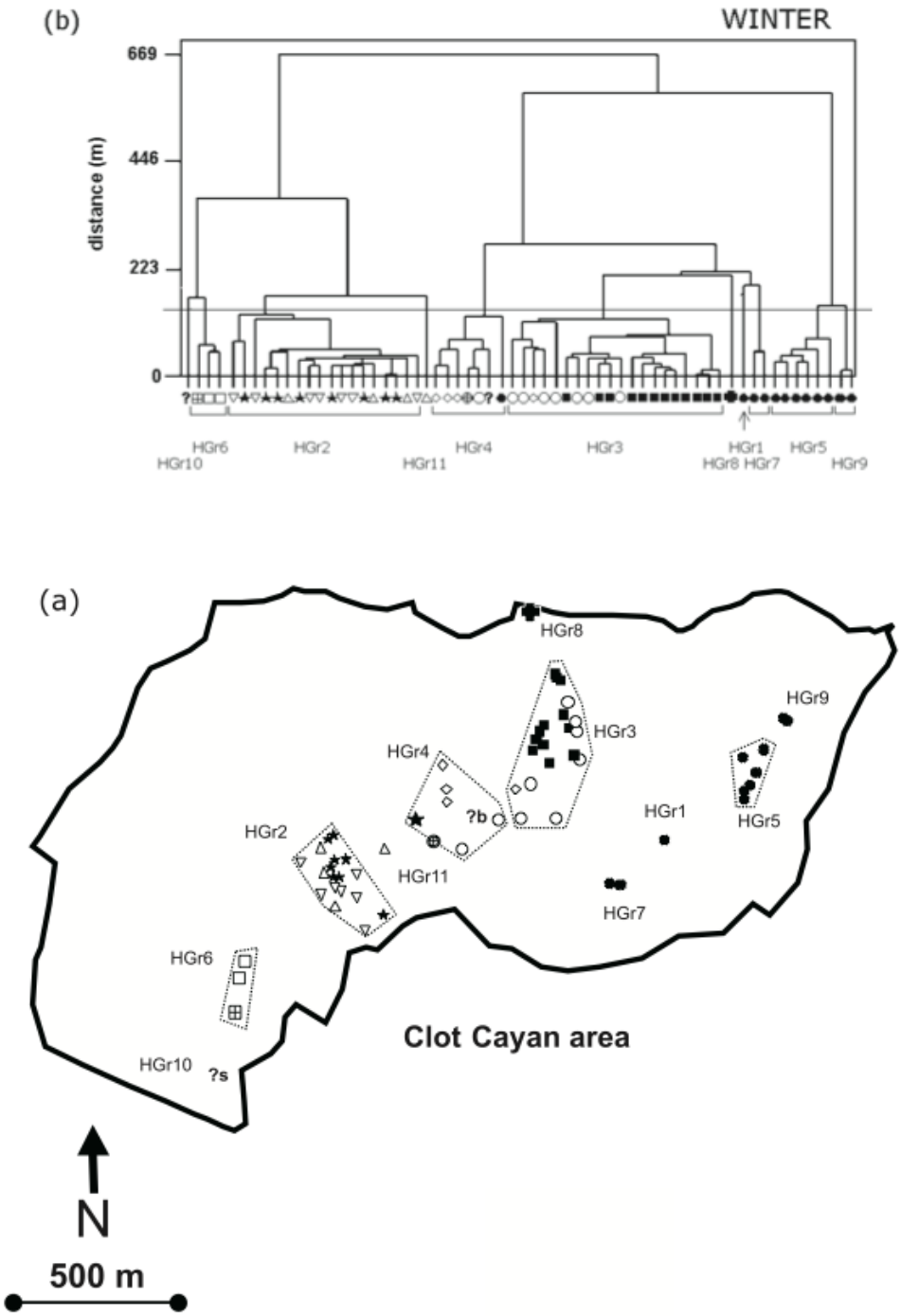


FIGURE 5.4 – *Clusters trouvés par la méthode statique des barycentres en hiver. En haut, le dendrograme de clustering. En bas, les positions correspondantes des barycentres des individus. Les formes représentant les barycentres correspondent à celles définies pour l'été (figure 5.3).*

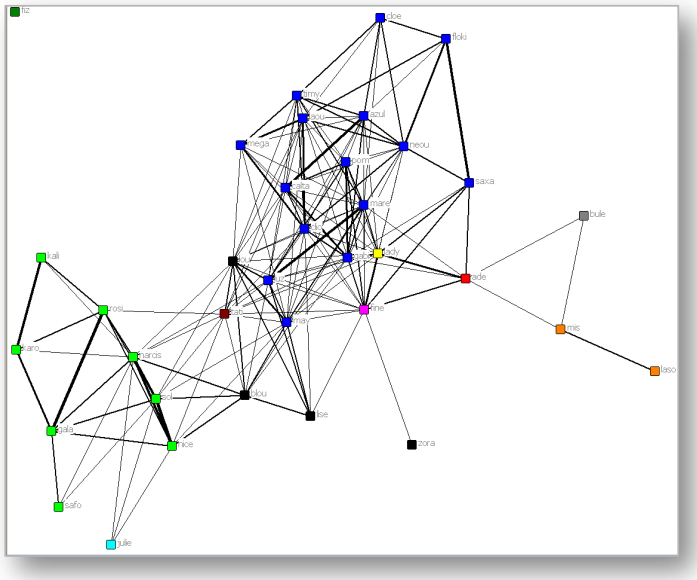


FIGURE 5.5 – *Réseau correspondant aux données cumulées sur la période allant de 1993 à 1999.*

5.2.2 Résultats obtenus précédemment

Ce jeu de données avait déjà donné lieu à des études approfondies [GC01, CLG+06, CCDP97, CGL02, CBG+07]. Dans Crampe et al. [CBG+07] l'organisation spatiale de la population avait été étudiée en détail. Les chercheurs, avec qui nous avons travaillé, avaient classé les individus en clusters en utilisant la méthode suivante : pour chaque individu, ils avaient calculé le barycentre de ses positions en hiver, et le barycentre de ses positions en été. Ensuite, pour chacune de ces saisons, ils avaient adopté une méthode de partitionnement (clustering) classique pour trouver les groupes. Une illustration du type de résultat obtenu peut être observé sur les figures 5.3 et 5.4, qui représentent les clusters trouvés en été et en hiver. Ce premier résultat permettait de tirer certaines conclusions :

- Les clusters identifiés correspondent bien à des groupes connus des éthologues par observation.
- Les comportements des individus peuvent être classifiés en 2 groupes : sédentaires et migrateurs. Les sédentaires restent dans la zone d'hivernage, y compris en été, alors que les autres peuvent se déplacer sur de grandes distances lors de cette saison, durant laquelle la neige ne les empêche pas de se déplacer.
- Les groupes d'individus ont un fort attachement spatial, puisqu'en faisant un barycentre sur plusieurs années entre les positions d'individus, on trouve des groupes très clairement définis.

L'article traite évidemment de plusieurs autres aspects, plus comportementaux, qui ne sont pas pris en compte pour ce travail. L'aspect de cette première recherche qui pouvait être amélioré concernait son caractère statique : les barycentres étaient constitués à partir des données cumulées sur toute la période d'étude. Or, en atteignant une période d'étude de plus de 10 ans, et les isards n'ayant une espérance de vie moyenne ne dépassant pas les quinze ans, de fortes modifications étaient susceptibles d'intervenir au cours du temps. Des premiers travaux avaient été effectués par les mêmes auteurs en travaillant à partir de réseaux, et en étudiant les chan-

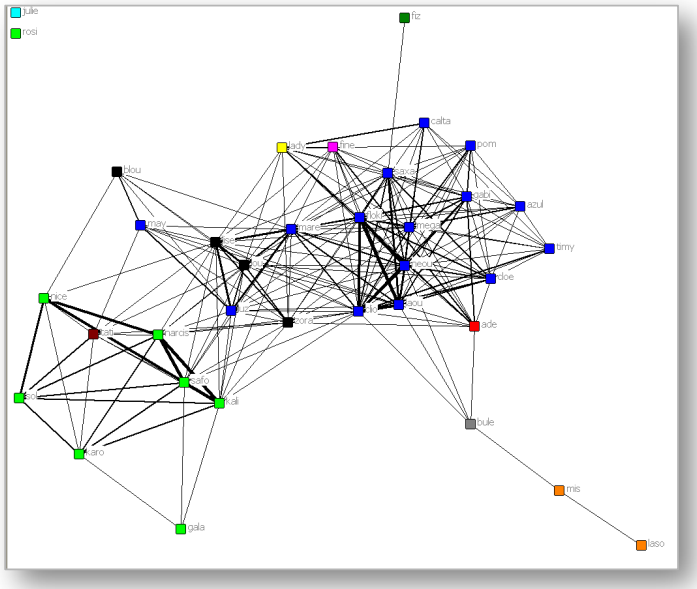


FIGURE 5.6 – *Réseau correspondant aux données cumulées sur la période allant de 2000 à 2006.*

gements intervenus entre deux périodes. Sur les figures 5.5 et 5.6, on peut observer un réseau constitué des relations en hiver entre les femelles sur la période 1993 - 1999, comparé aux mêmes données sur la période 2000 - 2006.

Ce premier travail semblait indiquer que les communautés d'individus restaient relativement stables au cours du temps. Nous avons proposé une solution pour appliquer la détection de communautés dynamiques à ce problème, afin d'essayer de fournir de nouveaux éléments pouvant aider à la compréhension des comportements sociaux de cette population.

5.2.3 Création du réseau dynamique

Les données originales sur lesquelles nous travaillons sont donc constituées de séquences de co-observations d'individus. Par exemple, nous pouvons savoir que le jour j, les individus $i1$, $i2$ et $i3$ ont été vus ensembles, et, séparés de ces derniers, les individus $i4$ et $i5$ étaient également co-localisés. Le jour $j+1$, ce sont par exemple les individus $i1$ et $i2$ qui sont observés au même endroit.

Comme cela a été décrit au chapitre 3.4.1.1, nous disposons d'une procédure nous permettant de transformer ces séquences d'interactions en un réseau dynamique. Il suffit pour cela de spécifier les paramètres F et P, qui indiquent sur quelle période P il nous faut au moins F interactions pour créer un lien.

Dans notre cas, en concertation avec les éthologues, nous avons créé plusieurs réseaux à partir de ces données de base. Le problème qui se pose, déjà évoqué, est que la nature des observations en été et des observations en hiver sont assez différentes. Il y a par exemple moins d'observations d'individus en été qu'en hiver, à cause de la plus grande dispersion de ces derniers.

Nous avons donc proposé une solution dans laquelle les périodes hivernales et estivales ont été séparées les unes des autres, et où nous avons créé deux réseaux dynamiques, l'un correspondant aux données d'hiver, l'autre aux données d'été.

Pour la période estivale, nous disposons de 888 observations, comptabilisant entre 2 et 11

individus. En hiver, nous en avons 2490, comptant entre 2 et 15 animaux. Pour ces réseaux ne concernant pas la totalité des mois de l'année, nous n'avions pas beaucoup de liberté dans le choix du paramètre P. En effet, nous ne pouvions pas mettre une valeur de quelques mois seulement, sinon, lors de la période non traitée, tous les liens se seraient terminés. Par exemple, en traitant les données hivernales, il n'y a aucune interaction durant les mois d'été. Nous avons donc choisi une valeur de P égale à 12 mois ce qui signifie qu'il nous fallait au moins F observations sur une période glissante d'un an pour créer un lien entre deux individus. Pour ne pas fausser les résultats, nous n'avons pas traité les données en temps réel mais selon la méthode la plus précise, c'est à dire que le lien entre deux individus se termine lorsqu'a lieu la dernière des F interactions satisfaisant la condition. Étant donné le faible nombre d'observations disponibles, nous avons choisi des paramètres de F peu restrictifs. Pour l'été, où nous disposions du plus faible nombre d'informations, nous avons choisi $F = 2$, c'est à dire qu'il suffit d'observer 2 fois un couple d'individus ensemble pour que cela constitue un lien entre eux. Nous ne voulions pas choisir une valeur $F = 1$, car cela donnerait une trop grande sensibilité au bruit. Dans le cas par exemple ou un individu aurait été incorrectement identifié, la probabilité que la même erreur se reproduise deux fois est très faible. Pour la période hivernale, comme nous disposions d'environ 3 fois plus d'observations, nous avons multiplié également la valeur de F par 3, ceci afin de rester cohérents en ce qui concerne le niveau d'interaction nécessaire pour créer un lien entre deux individus.

Nous avons également expérimenté une étude sur toutes les périodes, sans séparer les saisons, et dans ce cas nous avons pris $P = 3$ mois, et $F = 2$.

5.2.4 Résultats

5.2.4.1 Validation des communautés trouvées

La première étape consistait à montrer que les communautés trouvées par l'algorithme étaient cohérentes. Pour ce faire, nous les avons comparées avec celles qui avaient été découvertes —et jugées pertinentes— dans le premier article publié. Il faut rappeler que les méthodes utilisées sont très différentes : dans la méthode originale, seuls les barycentres des positions successives des individus sur toute la période d'étude étaient utilisés, rassemblés en groupes par un algorithme de clustering. Au contraire, notre méthode consistait à créer un réseau social à partir des collocations d'individus, puis d'appliquer l'algorithme de détection de communautés dynamiques iLCD. Nous commençons par détailler la correspondance entre les communautés trouvées en été par les deux méthodes. Avec la méthode spatiale, 8 clusters avaient été trouvés. 3 d'entre eux comportent entre 19 et 20 individus ($S6, S7, M2a$), les autres entre 4 et 8 individus($S2, M1, M2b, M3, M4$). On retrouve ces communautés dans l'illustration 5.3.

Les communautés trouvées par iLCD sont dynamiques, pour les comparer avec les communautés trouvées précédemment, nous considérons l'ensemble des nœuds ayant été présents dans une communauté, peu importe à quel moment. Cinq communautés sont trouvées par notre méthode. Le tableau 5.1 présente la correspondance entre les communautés trouvées et les communautés connues.

Comme on peut le voir, la correspondance est très forte. La communauté $M1$ n'est certes pas détectée par notre algorithme, mais cela provient probablement du faible nombre de données pour cette communauté. De plus, nous reparlerons du cas de cette communauté dans la section suivante, car elle présente une anomalie. Pour les trois grandes communautés en revanche, il est clair que nous disposons de groupes très proches. On peut voir que les groupes $S6$ et $S7$, bien que nettement distincts, ont quelques membres interchangés. Effectivement, avec la méthode des barycentres, on pouvait observer une proximité géographique très forte entre ces deux groupes, ce

	C_1 1994-2007	C_2 1994-2007	C_3 1999-2005	C_4 2003-2007	C_5 2001-2003
$M1$ (6)					
$M2a$ (20)		77%			
$M2b$ (4)		7%			
$M3$ (8)			100%		
$M4$ (4)					100%
$S2$ (4)		16%			
$S6$ (19)	80%			10%	
$S7$ (19)	20%			90%	

TABLE 5.1 – *Correspondance entre les communautés trouvées et les communautés connues. Le nombre indiqué entre parenthèse après le nom des communautés correspond au nombre d'individus de la communauté. Le nom des communautés initiales indique si la communauté est sédentaire (S) ou migrante (M). La similarité est calculé de la manière suivante :* $\frac{|\{C_x\}\cap\{MSX\}|}{|\{C_x\}|}$, *avec* C_x *les communautés trouvées par iLCD et* MSX *les groupes connus. L'absence de la communauté* $M1$ *est expliqué au chapitre 5.2.4.2*

qui peut expliquer ce phénomène. Nous observons également que la communauté correspondant majoritairement à $M2a$ a contenu, à certaines périodes, des individus des communautés $M2b$ et $S2$. Il y a également une explication simple à cela : en hiver, comme cela apparaît avec la méthode des barycentres, ces trois communautés n'en forment plus qu'une.

Pour la période hivernale, la comparaison est plus compliquée car dans les données d'origine, tous les individus n'avaient pas été catégorisés. Les observations claires étaient que les communautés $S2$, $M2a$ et $M2b$ n'en formaient plus qu'une lors de la période d'hivernage, que les communautés $S6$ et $M3$ étaient également rassemblées, tandis que les communautés $M1$ et $S7$ restaient clairement définies. Ces résultats se sont retrouvés à l'identique dans notre solution.

Nous avons donc validé que nous trouvions des résultats fortement semblables à ceux obtenus dans la première étude. Cependant, uniquement à partir de ce premier résultat, nous pouvons déjà faire une observation simple : si toutes les communautés contenant de nombreux individus ($M2a, S6, S7$) existent jusqu'à la fin du jeu de données, et ce depuis presque le commencement, en revanche, celles qui sont de plus petite taille n'ont en fait existé qu'à un moment donné : de 1999 à 2005 pour $M3$, de 2001 à 2003 pour $M4$. En étudiant les résultats plus en détail, nous pouvons faire d'autres observations.

5.2.4.2 Groupe sans coexistence temporelle

La communauté $M1$ n'avait pas été détectée en été par notre méthode. Cependant, en hiver, nous avons observé deux communautés qui contenaient chacune une partie des individus de $M1$. Mais l'une de ces communautés n'est présente que de 1994 à 1996 tandis que l'autre n'existe qu'entre 2006 et 2008. De plus, aucun individu n'apparaît à la fois dans la première et dans la seconde. Vérification faite, ces individus ne sont en effet pas contemporains. Rien ne permet donc d'affirmer qu'il s'agisse d'un même groupe. On peut donc se rendre compte d'un premier avantage de l'analyse dynamique, qui évite le risque de la création de relations entre individus qui ne sont que des biais dus à l'agrégation de données sur une trop longue période. Dans ce cas particulier, nous ne pouvons pas affirmer que ces deux groupes d'individus n'appartiennent pas, en fait, à la même communauté, car tous les individus ne sont pas marqués, la communauté

a donc peut-être continué à exister à travers des individus non identifiables. Cependant, rien ne permet non plus d'affirmer que ces individus aient pu faire partie d'un même groupe social, même à des époques différentes.

5.2.4.3 Groupe changeant de comportement

Le renouvellement important des individus, nombre d'entre eux disparaissant et de nombreux autres apparaissant, peut facilement conduire, dans une étude agrégative, à détecter des communautés correspondant en fait à des individus constituant une génération (les individus de la même génération ayant un comportement très proche), alors que la communauté elle même a une existence perdurant au fil des générations, avec des changements dans le comportement de ses membres.

Dans les communautés détectées à l'origine, les communautés $M2a$ et $S2$ étaient séparées en été et similaires en hiver. Le groupe $S2$ est un groupe sédentaire, qui ne se déplace pas en dehors de la zone d'hivernage en été, alors que les individus de $M2a$ sont migrants et en sortent donc légèrement.

Cependant, parmi les 8 individus de $S2$, 4 semblaient adopter le comportement du groupe $M2a$ à partir d'une certaine date (2003).

Dans notre résultat, les individus de $M2a$ et de $S2$ semblaient ne former qu'une seule communauté, quelle que soit la période. En vérifiant au cas par cas, nous nous sommes aperçus que les individus de $S2$ qui ne changeaient pas de comportement avaient disparu du jeu de données avant 2003. De plus, les individus de $M2a$ ont aussi tendance à être parfois observés dans la zone d'hivernage en été. Il semble donc que ces deux groupes en forment en fait un seul, dont le comportement est un peu plus complexe qu'une simple migration en été. Il semble que les individus puissent parfois migrer ou ne pas migrer, et que ce comportement de migration soit devenu de plus en plus courant au cours du temps. Les individus qui ont disparu au début du jeu de données n'ont donc jamais été observés en migration, et ont été considérés comme faisant partie d'une communauté différente, alors qu'en fait, il s'est toujours agit du même groupe, ayant eu de plus en plus tendance à migrer au cours du temps. Ici aussi, c'est grâce à l'analyse dynamique qu'une telle particularité a pu être détectée.

5.2.4.4 Individus appartenant à plusieurs communautés

L'une des questions à laquelle il était difficile de répondre avec l'étude initiale était : est-ce que les communautés sont imperméables, ou est-ce que des individus peuvent appartenir à plusieurs groupes, simultanément ou à différentes périodes de leurs vies ?

La détection initiale en cluster ne permettait pas de recouvrement. Notre méthode, en revanche, ne l'empêche pas. Pourtant, on n'observe que très peu de communautés en recouvrement. Ceci semble donc venir confirmer une certaine immiscibilité des groupes d'individus.

En revanche, et ceci n'avait pas été découvert par l'approche statique, un individu change clairement de groupe au fil du temps. Ce cas semble unique, et concerne un seul animal. Cet individu (identifiant : *zazi*) fait partie de la communauté $M3$ de 1994 à 1998. Puis, de 2000 à 2004, il est intégré à la communauté $S6$. Finalement, de 2004 à 2006, il appartient à la communauté $M2a$. Une observation attentive des données est venue confirmer cette particularité.

5.2.5 Analyse de la dynamique

L'intérêt le plus important de l'analyse des communautés temporelles est évidemment la possibilité de suivre l'évolution de ces communautés au cours du temps.

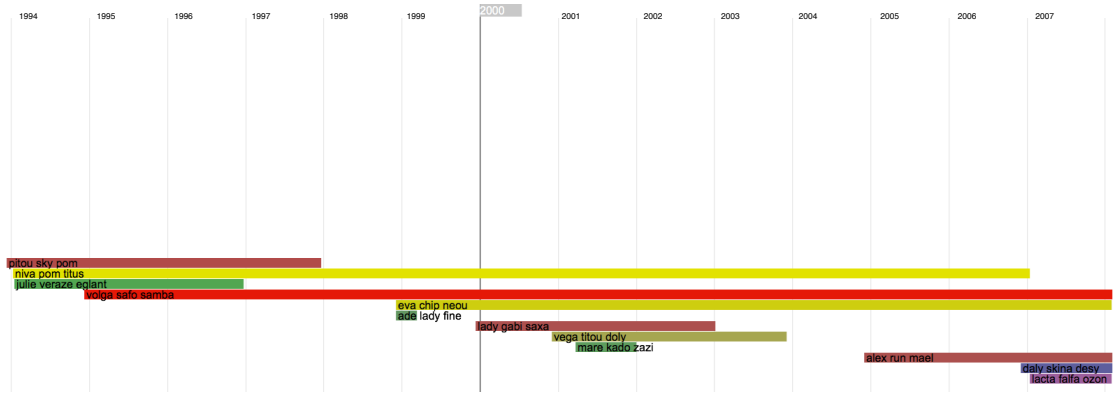


FIGURE 5.7 – *Visualisation de l'ensemble des communautés d'isards en Hiver. Chaque ligne horizontale correspond à une communauté. Le nom des communautés est un sous-ensemble des noms des individus qui la composent.*

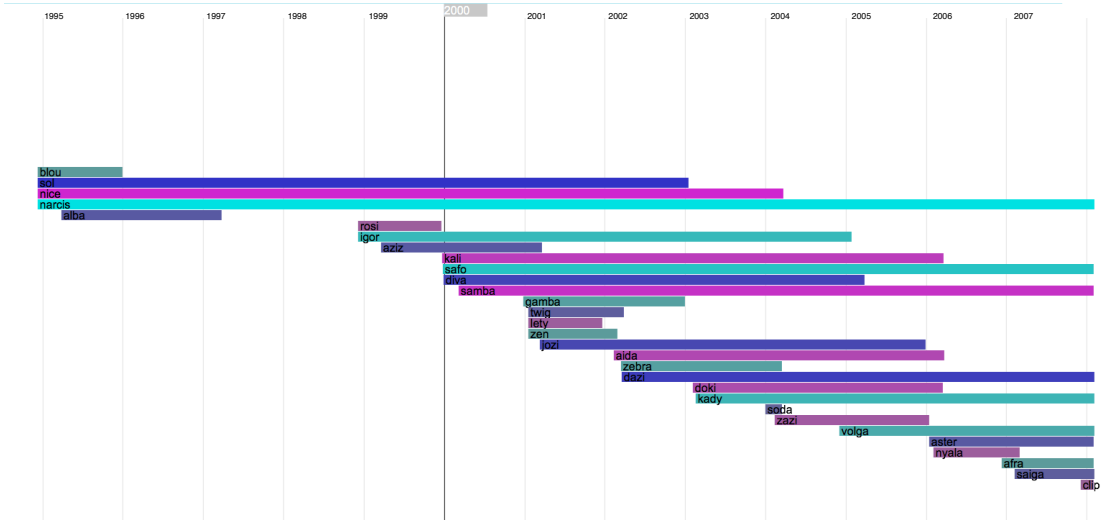


FIGURE 5.8 – *Visualisation de la communauté M2a en Hiver. Chaque ligne horizontale correspond à un individu, on peut observer la date à laquelle il intègre la communauté et celle à laquelle il la quitte.*

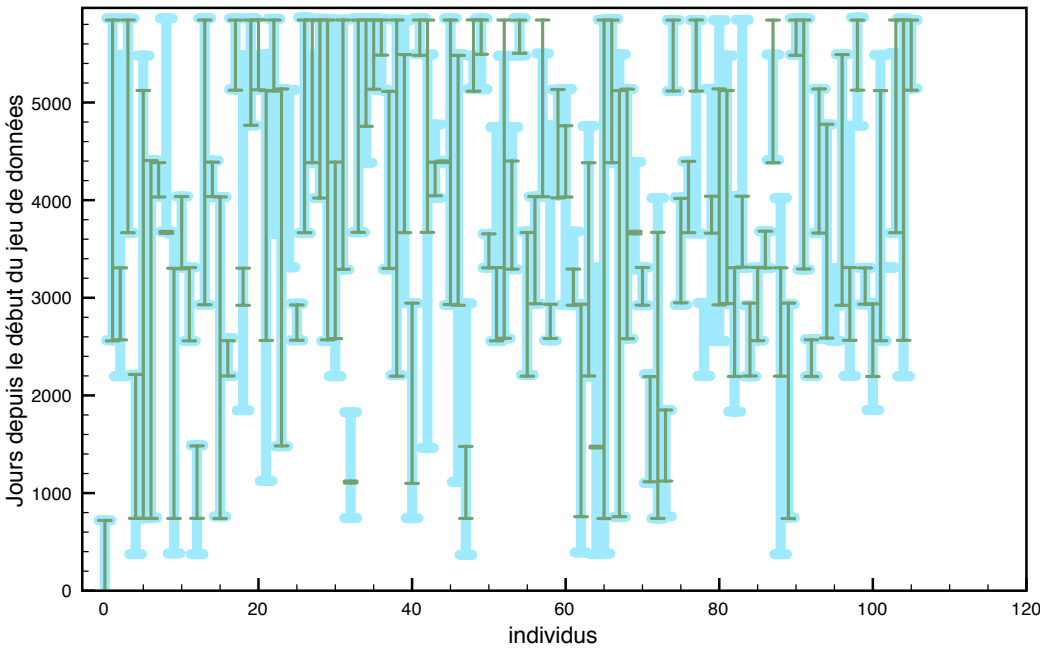


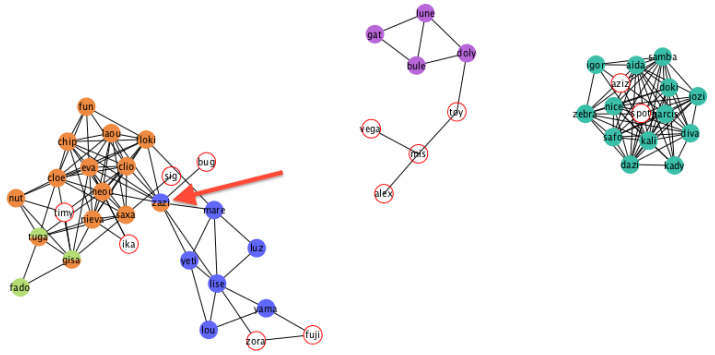
FIGURE 5.9 – *Visualisation de l'appartenance des individus aux communautés. Chaque ligne verticale bleu représente la durée de vie d'un individu. Chaque ligne verticale verte représente la période durant laquelle cet individu fait partie d'une communauté. On remarque que pour la majorité des individus, une communauté leur est attribuée durant la majorité de leur vie.*

Nous avons évoqué, au chapitre 6.1, des méthodes de visualisation de cette évolution. A titre d'exemple, nous pouvons montrer ce que donne la visualisation de l'ensemble des communautés sur la période hivernale 5.7. On observe bien l'existence de communautés stables, existant sur toute la période, ainsi que la présence de quelques communautés de durée de vie plus courte.

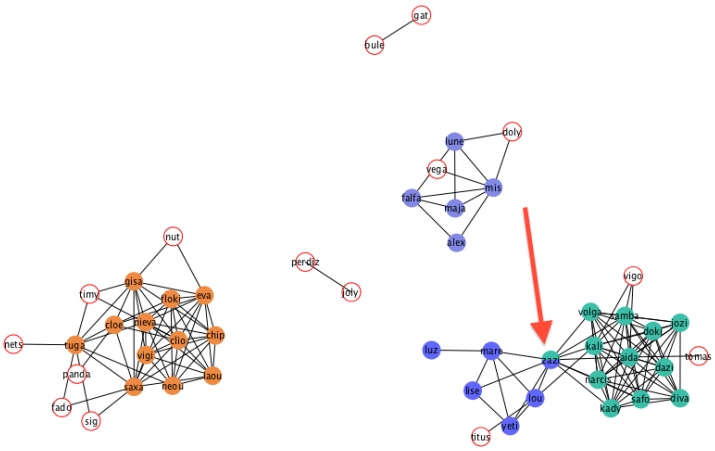
Mais l'aspect le plus intéressant dans cette application est la visualisation du détail de l'évolution d'une seule et même communauté. Sur la figure 5.8, on peut suivre l'évolution de la communauté correspondant au groupe $M2a$. On voit nettement que la majorité des individus présents à la fin du jeu de données ne sont pas les mêmes que ceux du début. Un seul individu (*narcis*) est présente durant toute la période. On note ensuite que chaque individu est présent plus ou moins longtemps. On peut également remarquer l'apparition de *zazi*, l'individu qui change de communauté au cours du temps. On note que sa présence dans la communauté est plus courte que celle de la plupart des autres animaux.

Ici, une information importante pour l'analyse des données traitées est le caractère continu de l'appartenance de chaque individu à la communauté. Si les animaux étudiés n'avaient pas d'attachement fort à leur groupe, on pourrait observer de nombreux animaux présents une année, absents l'année suivante, revenant l'année d'après, et ainsi de suite. Par exemple, si l'on intègre les mâles à ces données, on peut faire ce type d'observation, les mâles ayant un comportement moins social que les femelles.

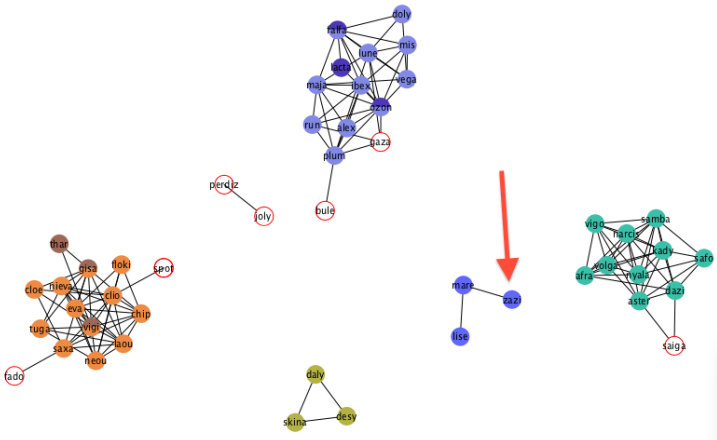
Toujours au chapitre 6.1, nous avions proposé une autre visualisation, plus axée sur la représentation du réseau lui-même. Cet outil est particulièrement pertinent ici, le réseau ne comportant pas trop de nœuds. A titre d'exemple, la figure 5.10 présente quelques captures d'écran mettant en exergue le comportement de l'individu *zazi*.



(a) *04/01/2002*



(b) *21/03/2005*



(c) *19/01/2007*

FIGURE 5.10 – *Exemples de visualisation de la dynamique du réseau. On peut ici observer le comportement du nœud zazi, identifié par une flèche, qui change de communauté au cours du temps.*

5.2.5.1 Individus sans communauté

Un autre aspect pouvant être étudié avec cette vision dynamique concerne l'intégration des individus à des communautés. Comme la plupart des algorithmes acceptant le recouvrement de communautés, iLCD propose un découpage qui n'est pas forcément complet, des nœuds pouvant rester sans communauté. Ici, cette information peut être importante : est-ce que les individus, lorsqu'ils apparaissent dans le jeu de données, sont immédiatement intégrés à une communauté ? Peuvent-ils parfois avoir une existence solitaire, en-dehors de toute communauté ? Existe-t-il des individus qui sont indépendants de tout groupe social ? Pour répondre à cette question, nous avons réalisé l'illustration 5.9. On peut voir, pour l'ensemble des individus femelles, leur période d'observation dans le jeu de données (bleu clair), ainsi que le temps qu'ils passent au sein d'une communauté (vert). La période d'observation est définie comme la période entre la première et la dernière observation de l'individu. Comme expliqué précédemment, cela ne correspond pas forcément à la totalité de la durée de vie de l'individu, celui-ci n'étant que rarement marqué dans sa jeunesse.

La conclusion ici est clairement que la plupart des individus passent toute leur vie au sein d'une communauté. Il existe quelques exceptions, mais, étudiées au cas par cas, elles ne révèlent pas de comportement clairement analysable.

5.2.5.2 Etude inter-saisonnière et communautés résurgentes

La dernière étude que nous avons fait consistait à ne plus tenir compte des saisons, et donc à créer un réseau dynamique tenant compte de toutes les informations, couvrant toute l'année. Cela nous a permis de réduire le paramètre P, c'est à dire la période durant laquelle nous devions observer des individus pour conserver le lien actif. Nous l'avons fixé à 90 jours, soit environ une saison. Pour le paramètre F, nous avons été obligés de rester à une valeur faible, étant donné le peu d'observations estivales : $F = 2$.

Le premier résultat observable, intéressant, est que l'on observe toujours des communautés ayant une existence continue sur la période, malgré la diminution de P. En particulier, les communautés correspondant aux grands groupes d'individus ($M2a, S6, S7$) se retrouvent clairement, quasiment sans interruption. En revanche, cette méthode a aussi mis en évidence une faiblesse de notre algorithme.

En effet, comme cela était prévisible, de nouvelles communautés se forment chaque été. Cela est logique, car nous savons que des groupes d'individus séparés en été n'en forment plus qu'un en hiver. A la fin de chaque été, donc, les communautés correspondant aux groupes estivaux sont fusionnées avec les autres groupes. Mais, lorsque l'été arrive à nouveau et que des groupes d'individus commencent à migrer, ces groupes estivaux sont correctement détectés, mais sont considérés comme de nouvelles communautés par notre algorithme. Ainsi, chaque été apparaissent de nouvelles communautés qui sont en fait les mêmes que les années précédentes, ce que notre algorithme ne peut pas percevoir. On peut donc se rendre compte qu'il serait pertinent d'intégrer un mécanisme permettant de faire correspondre, le cas échéant, une nouvelle communauté à une communauté ayant déjà existé, si cette communauté était similaire. Ce sont ces communautés que nous qualifions de résurgentes.

5.2.6 Bilan

Dans cette section, nous avons appliqué avec succès notre méthode à un cas réel complexe. Nous avons tout d'abord montré que les résultats que nous obtenions étaient tout à fait cohérents avec les résultats connus et validés. Dans un second temps, en poussant plus loin l'analyse des

résultats obtenus par notre méthode, nous avons mis en évidence l'intérêt d'utiliser une méthode d'analyse des communautés temporelles, notamment pour éviter les risques de confusion possibles liés à des changements de génération dans la population, ou plus simplement pour éviter les pièges que peut tendre l'agrégation de données. Les résultats obtenues ont été validés par les ethnologues avec lesquels nous avons travaillé, qui étaient à l'origine du jeu de données.

Mais nous avons également montré qu'une telle analyse ne permettait pas simplement d'obtenir un résultat plus fiable, mais ouvrait aussi de nouvelles possibilités d'étude. La visualisation de l'évolution d'une communauté, c'est à dire l'ensemble de ses modifications au cours du temps, notamment, est très riche sémantiquement.

5.3 Analyse d'une plateforme d'échange de vidéos : Nico Nico Douga

5.3.1 Présentation du réseau

Nico Nico Douga est une plateforme d'échange de vidéos japonaise, permettant à n'importe qui de mettre à disposition de tout le monde ses vidéos, à l'instar de YouTube ou de Daily Motion par exemple. Le site est très populaire au Japon, puisqu'il est aujourd'hui à la douzième place des sites les plus consultés du pays, et compte plus de 22 millions de membres.

Ses fonctionnalités sont un peu plus avancées que celles de la plupart des plateformes d'échange de vidéos concurrentes. Il est par exemple possible d'écrire des commentaires apparaissant directement en surimpression sur les vidéos, à un moment précis et à un endroit précis, ce qui permet d'interagir avec la vidéo elle-même via les commentaires. Une autre possibilité offerte aux créateurs de la vidéo, que nous n'avons pas exploitée dans ce travail mais qui avait déjà été étudiée par Hamasaki et al. [HTN08], est l'ajout de références (liens hypertextes) vers d'autres vidéos présentes sur Nico Nico Douga, et dont, par exemple, des extraits sont présents dans la vidéo publiée. Cette possibilité est largement utilisée pour certains usages, en particulier la création collaborative de vidéos. Ce sont ces phénomènes qui ont été étudiés dans l'article de Hamasaki et al., et en particulier l'apparition du phénomène Hatsune Miku, une chanteuse virtuelle dont des clips ont été réalisés de manière collaborative, certains contributeurs composant une chanson et la faisant chanter à un synthétiseur vocal adapté (VOCALOID), d'autres améliorant l'arrangement musical, un autre créant une vidéo à l'aide d'animations 3D, et ainsi de suite.

Enfin, une dernière fonctionnalité nous a fourni un sujet d'étude particulièrement intéressant : à chaque vidéo peuvent être associés des mots-clefs, ou tags. Ces mots-clefs sont en nombre limité, entre 0 et 10, et peuvent être définis librement par les utilisateurs. Ils peuvent être très génériques (*jeux-vidéos*, *musique*, ...) ou plus spécifiques (*chaton*, le nom d'un club de baseball, le nom d'un jeux vidéo, etc.)

Nous avons donc décidé d'exploiter ces tags pour essayer d'en extraire les tendances au sein de la plateforme, comme par exemple les sujets les plus populaires, le moment auquel ils apparaissent, auquel ils disparaissent, et comment ils évoluent. Pour ce faire, nous avons créé, à partir des tags attribués aux vidéos, un réseau temporel de termes.

5.3.2 Travaux existants sur ce domaine

La détection de tendances dans les réseaux (*trend detection*), est un sujet de plus en plus populaire. Confronté à un corpus de termes créés par des utilisateurs, de très grande taille, et évoluant au gré des actions de ces utilisateurs, il est intéressant d'être en mesure de pouvoir

détecter automatiquement quels sont les *hot topics*, les sujets les plus discutés du moment, ou encore les axes de discussion généraux de la plateforme.

Parmi les premiers travaux en ce sens, on peut citer la détection de *bursting terms*, des termes dont l'utilisation est très importante durant une période de temps, beaucoup plus qu'avant et après cette période. Dans le travail de Kleinberg [Kle03], un automate à état infini est utilisé pour identifier ce type de phénomènes dans n'importe quel type de corpus. L'intérêt principal de ce travail est d'être capable de mesurer la durée d'une explosion, l'analyse étant faite sur un corpus dynamique, et pas sur des séquences d'instantanés. Appliquée à des réseaux sociaux Web 2.0, une telle solution détecterait efficacement les termes clefs. Le problème est que pour un événement, plusieurs termes sont généralement employés, et, cette méthode n'ayant pas d'aspect réseau, chaque terme serait considéré comme un résultat.

Dans l'article de Laniado et al [LM10] l'objectif est d'identifier des termes importants sur le réseau social Twitter. En utilisant une analyse statique sur des fenêtres d'une journée, les auteurs détectent des termes clefs, certains présentant un profil explosif, d'autres un usage plus soutenu. Cependant, ce travail était fait, à nouveau, en traitant chaque terme séparément. Seuls les hashtags de Twitter étaient pris en compte (les hashtags Twitter sont des mots clefs inclus par les utilisateurs dans leurs messages, commençant par le caractère #, et servant à identifier leur message comme relié à un sujet (exemple : ajouter le hashtag #electionsAmericaines à un message signifie que ce message est en rapport avec les élections américaines). Benhardus et al. [Ben10] ont utilisé le même type de mécanisme, mais en utilisant tous les termes des tweets, ainsi que des bigrammes (couples de mots consécutifs), et non plus seulement les hashtags. Cependant cette méthode ne peut pas non plus rassembler les termes sémantiquement liés.

La solution proposée par Li et al. [LGZ08] et appliquée au site web delicious.com sur des tags liés aux URLs partagés permet de rassembler ensemble des termes qui apparaissent simultanément pour les mêmes URLs, et pas seulement les bigrammes. Par exemple, si l'on cherche pour des bigrammes, on ne va jamais considérer les termes *nourriture* et *recette* comme liés, alors que, si l'on cherche des co-occurrences, nous pourrons détecter ce lien sémantique. Cependant, cette méthode ne considère simplement que les co-occurrences plus fréquentes qu'un certain seuil, sans aspect réseau, et ne pourra donc pas mettre dans un même cluster des termes utilisés de manière exclusive, tels que deux orthographes différentes d'un même mot par exemple. De plus, comme indiqué par les auteurs, ce mécanisme conduit à l'identification d'un très grand nombre de sujets (148.000 pour un jeu de données plus petit que celui que nous étudions dans cet article.)

Weng et al. [WYLL11] utilisent des transformations par vaguelettes pour détecter des explosions d'usage de termes uniques. Ensuite, les auteurs proposent d'utiliser des similarités dans les patterns d'explosion pour assigner une similarité à des paires de termes. Dans une seconde étape, ils créent un réseau dans lequel les nœuds représentent des termes et les liens la similarité entre leurs patterns d'explosion. Un algorithme de détection de communautés classique est ensuite utilisé sur ce réseau pour trouver des clusters de termes similaires. Cette méthode a l'avantage sur toutes les autres d'être capable de mettre dans un même cluster des termes qui ne sont pas nécessairement utilisés simultanément. Cependant, nous pouvons relever certaines limitations : d'abord, l'algorithme de détection de communautés utilisé est statique, en conséquence la détection est faite sur des fenêtres d'une journée. Il n'y a donc pas de continuité dans les communautés détectées (dans leur application sur Twitter, aucun événement n'est détecté plus d'une journée). Nous pouvons aussi remarquer que les communautés contiennent très peu de termes (2,3), ce qui n'est parfois pas suffisant pour comprendre leurs sens. Enfin, la méthode utilisée pour la détection de communauté ne permet pas de recouvrement. Ainsi, ils détectent lors d'une journée d'analyse une communauté composée des deux termes *Vuvuzela* et *Soccer* (l'analyse a été menée durant la coupe du monde de football). Cela signifie que ce jour là, aucune autre communauté

ne pourra contenir le terme *Soccer*.

Deux autres méthodes (TwitterStand par Sankaranayan et al. [SST+09] , Becker et al. [BNG11]) proposent une approche assez différente des précédentes, avec cependant les mêmes objectifs. L'idée n'est plus de détecter des groupes de termes, mais des groupes de contenus publiés par les utilisateurs. Par exemple sur Twitter, ils essaient de regrouper des tweets qui semblent similaires, et ensuite de déduire le sujet commun de ces tweets. Le gros avantage de ces méthodes comparées aux précédentes est qu'elles sont capables de traiter la plateforme étudiée en continu, c'est à dire qu'à tout nouveau contenu posté par un utilisateur, est assigné soit une communauté existante soit une nouvelle communauté si aucune de celles qui existent ne semble pertinente. En contrepartie, comme cela a été identifié par les auteurs de TwitterStand, il y a un problème de fragmentation des sujets. Si, au départ, deux tweets différents mais traitant du même sujet ne sont pas identifiés comme appartenant au même cluster, tous les nouveaux tweets traitant de ce sujet vont avoir tendance à être regroupés avec l'un ou l'autre de ces sujets, conduisant à une fragmentation importante. De plus, chaque pièce de contenu ne peut appartenir qu'à un et un seul cluster.

Enfin, TwitterStand a l'avantage important par rapport aux autres méthodes de pouvoir détecter des événements de durée indéfinie, c'est à dire, dans un même réseau, des événements courts (quelques jours) et des événements plus longs. Cependant, cela est implémenté par une méthode assez stricte : un événement se termine si le barycentre temporel de tous ses composants est plus ancien que 3 jours. Cela signifie qu'un événement, pour continuer à exister après cette période, doit continuer à croître exponentiellement sans ralentir.

La méthode que nous proposons ici a plusieurs avantages par rapport aux méthodes proposées précédemment. Premièrement, elle permet d'analyser les événements au fur et à mesure de l'évolution du réseau, en temps réel, grâce à l'utilisation d'iLCD. Ensuite, elle permet de trouver des communautés qui sont non seulement composées de plusieurs termes, mais qui peuvent aussi être recouvrantes, ce qui se révélera très important. Enfin, cette méthode est capable de détecter aussi bien des événements très courts que des sujets stables dans la durée, du moment qu'il s'agit de sujets importants dans les échanges des utilisateurs.

5.3.3 Création du réseau de termes

L'intérêt principal de ce jeu de données était que nous disposions de la totalité des données pour une période assez étendue dans le temps. Les données originales avaient été extraites entre Février 2007 et Mai 2009, soit une période de plus de deux ans. Plus de quatre millions de vidéos avaient été collectées, utilisant plus de 3 millions de tags différents. Cependant, la majorité de ces tags sont utilisés moins de dix fois. Les vidéos publiées —et donc, les tags— concernent à peu près tous les sujets possibles. Les sujets les plus populaires sont cependant les jeux vidéos et les vidéos musicales, qui représentent plus de la moitié de la totalité des contenus publiés.

Pour créer le réseau temporel de termes, nous avons utilisé la méthode décrite en 3.4.1.1, d'une manière très similaire à l'application sur les isards.

Nos nœuds correspondent à l'ensemble des termes utilisés, et nous voulons qu'un lien reflète une proximité sémantique entre deux termes dans le réseau. Nous avons considéré que le fait que deux termes apparaissent dans les mots clefs de la même vidéo était un indice de cette proximité sémantique. En effet, si par exemple une vidéo présente une scène d'action du film de guerre FI réalisé par le réalisateur RE, alors les mots clefs associés à la vidéo seront *film, guerre*, FI et RE.

La co-occurrence de termes dans une même vidéo est donc l'interaction de base que nous utilisons pour passer d'une séquence d'interactions à un réseau dynamique. Si ce film est un

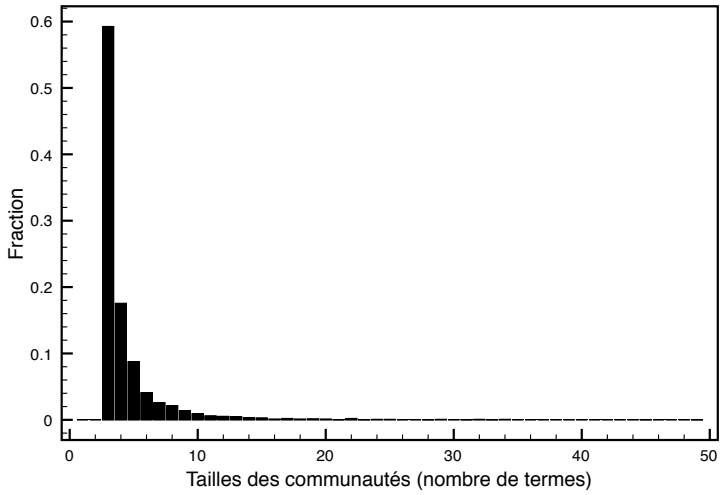


FIGURE 5.11 – *Fraction des communautés en fonction de leur taille.*

sujet populaire, de nombreuses personnes vont mettre en ligne des vidéos contenant les termes FI et RE, et nous allons considérer ce lien comme significatif.

Après avoir extrait les données de manière à savoir quel terme avait été utilisé avec quel autre à quelle date, nous avons utilisé l'algorithme de création de réseau temporel. Les paramètres ont été les suivants :

$$F = 100$$

$$P = 30$$

Ce qui signifie qu'il faut que deux termes présentent au moins 100 co-occurrences durant une période de 30 jours pour créer un lien entre eux. Nous pourrions changer ces paramètres pour nous concentrer sur des liens plus forts, et donc obtenir des communautés plus claires et mieux définies. Nous pourrions également les modifier dans le sens inverse, ce qui conserverait plus de liens entre les termes, mais au prix d'un bruit plus important dans le réseau à traiter. Il n'y a pas de bonne ou de mauvaise valeur de F et P, su moment qu'elles ne sont pas trop restrictives. Des valeurs différentes donnent des résultats différents.

Les valeurs choisies semblent offrir un bon compromis, et ont été choisies après plusieurs essais.

Le réseau ainsi créé contient 12865 nœuds, et est composé de 52366 étapes d'évolution, c'est à dire créations ou disparitions de liens.

5.3.4 Détection de communautés avec iLCD-NRMH

L'algorithme iLCD-NRMH a été appliqué directement sur ce réseau. Un total de 2865 communautés a été détecté, dont 467 n'ont pas disparu lors de la dernière modification du réseau (correspondant à la fin du jeu de données).

Il y a cependant une grande disparité entre ces communautés. Lorsque l'on travaille avec des communautés dynamiques, il y a en effet un critère très important à prendre en compte : la durée de vie des communautés. Certaines d'entre elles peuvent n'exister que pour une très courte durée alors que d'autres peuvent être présentes durant la totalité du jeu de données. A cela s'ajoute des différences déjà présentes dans le cas de communautés statiques, la diversité

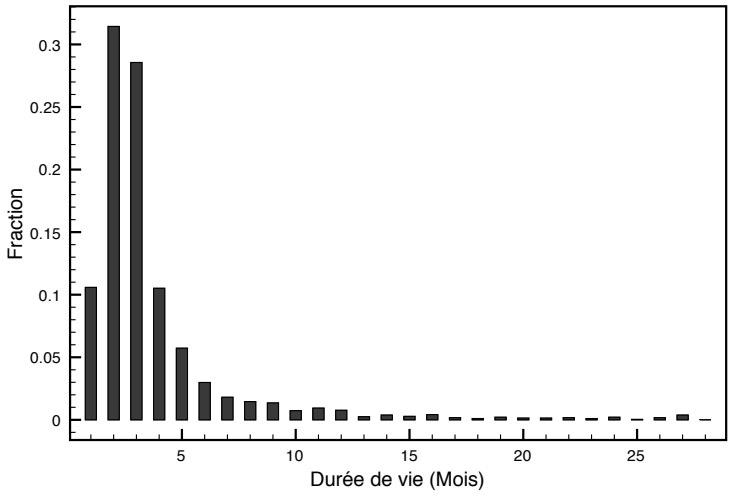


FIGURE 5.12 – *Fraction des communautés en fonction de leur durée de vie.*

de taille des communautés. Sur les figures 5.11 et 5.12, nous pouvons observer la répartition des communautés selon leur nombre de nœuds et leur durée de vie.

5.3.4.1 Durée de vie des communautés

La majorité des communautés ont une durée de vie assez limitée, entre deux et trois mois. Cela signifie que lorsqu'un nouveau sujet apparaît, les gens en parlent intensivement durant un mois ou deux, puis l'intérêt des utilisateurs diminue, ce qui conduit à sa disparition de la liste de nos communautés. Comme la durée de ces communautés est plus longue que notre paramètre P (30 jours), nous pouvons être sûr qu'il ne s'agit pas d'un biais qui viendrait du processus de création du réseau. Il s'agit bien d'une caractéristique des communautés de ce réseau : les gens s'intéressent à un sujet durant une période de plus d'un mois, mais s'en désintéressent ensuite naturellement.

Il y a d'ailleurs également de nombreuses communautés d'une durée de vie plus longue. On peut compter 378 communautés existant plus de 6 mois, et 21 communautés dont la durée de vie dépasse deux ans, c'est à dire pratiquement la totalité de notre jeu de données.

5.3.4.2 Taille des communautés

Comme on peut le voir sur la figure 5.11, la grande majorité des communautés sont petites, puisqu'elles contiennent moins de 5 nœuds. Cela n'est pas vraiment surprenant, car, pour un sujet donné, les utilisateurs tendent à toujours utiliser les mêmes termes. Par exemple, pour un jeu vidéo, les utilisateurs vont typiquement taguer leurs vidéos avec le nom du jeu —parfois avec deux ou trois variations, comme un acronyme, un tag pour la catégorie de jeu ou pour le nom de la série si le jeu est une suite d'un autre jeu connu, et enfin un tag général de type *jeu vidéo*.

Il existe cependant quelques communautés incluant un grand nombre de nœuds. Ainsi, 15 communautés sont constituées de plus de 30 nœuds. Cependant, tous n'appartiennent pas forcément simultanément à la communauté. Un nœud peut apparaître à un moment donné, disparaître, et être remplacé par un autre plus tard. Un bon exemple est la communauté correspondant à la série de jeux vidéos *Final Fantasy*. Elle inclue les noms de plusieurs épisodes de la série,

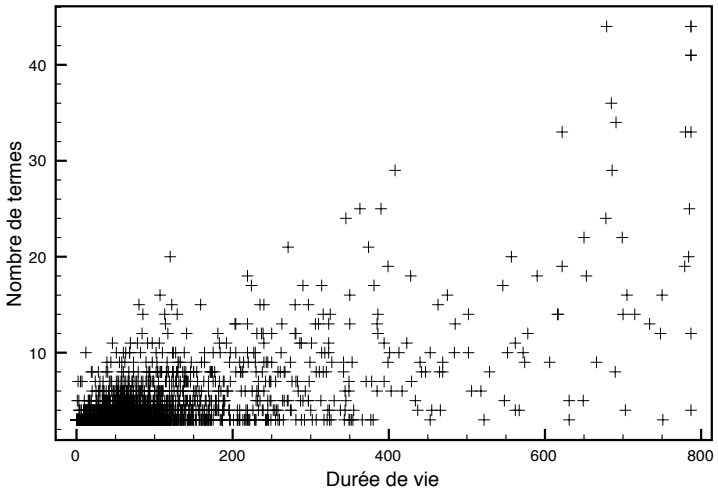


FIGURE 5.13 – *Corrélation entre la taille des communautés et leur durée de vie.*

mais pas nécessairement simultanément. Par exemple, lorsqu'un nouveau jeu est publié, son nom devient populaire et est intégré à la communauté. Mais après quelques semaines, ce nom va disparaître, jusqu'à ce qu'un autre épisode de la série voit le jour, et la communauté intègrera alors son nom. Comme cette série est très populaire au Japon, les tags généraux concernant la série seront, eux, présents en permanence, assurant le maintien de la communauté sur une longue période.

5.3.4.3 Corrélation entre taille et durée de vie

Il serait raisonnable de supposer que les communautés avec une longue durée de vie seraient celles ayant le plus de nœuds, comme dans le cas des séries présenté ci-dessus. La figure 5.13 montre la corrélation entre ces deux valeurs.

On observe qu'en effet, les communautés avec une durée de vie courte ont tendance à être composées de peu de nœuds. Cependant, si celles dont le nombre de nœuds est le plus élevé sont des communautés à longue durée de vie, toutes celles ayant une longue durée de vie n'ont pas forcément de nombreux nœuds. Il existe donc des communautés restant stables, sans intégration de nouveaux nœuds, sur de longues périodes. Le meilleur exemple en est une communauté existant durant tout le jeu de données, mais qui ne contient que 4 nœuds. Ceux-ci peuvent être traduits par : *vidéo, Nico Nico vidéo, Nico Nico commentaires* et *commentaires*. Ces tags très généraux sont beaucoup utilisés, durant tout le jeu de données, et souvent ensemble, mais n'ont pas de raison d'être liés à d'autres tags. Cette communauté reste donc stable jusqu'à la fin du jeu de données.

5.3.5 Catégories de communautés

L'exploration des événements détectés nous a permis de les classer en catégories. Dans cette section, nous décrivons ces différentes catégories et donnons des exemples d'événements correspondants.

Événement détecté	Date de création	Date de fin	Date de sortie	Délai de détection (j)
Devil May Cry	02/12/2007	08/08/2008	31/01/2008	-60
Fable 2	06/12/2008	03/02/2009	18/12/2008	-12
Gears Of War 2	14/10/2008	29/12/2008	07/11/2008	-24
Assassin's Creed	25/01/2008	26/02/2008	31/01/2008	-6
Soul Calibur IV	07/07/2008	15/11/2008	31/07/2008	-24
Uncharted	11/11/2007	02/01/2008	16/11/2007	-5

TABLE 5.2 – *Exemple d'événements détectés, correspondant à des jeux-vidéos. On peut observer que leur apparition est fortement corrélée à la date de sortie dans le commerce du jeu correspondant.*

5.3.5.1 Événements ponctuels

Une première catégorie est celle des événements ponctuels, détectés sous la forme de communautés à courte durée de vie. Nico Nico Douga est une plate forme utilisée majoritairement par de jeunes japonais, qui échangent beaucoup sur les jeux vidéos, les films ou séries d'animation, et les musiques. La majorité des sujets est donc en rapport avec ces centres d'intérêts. Dans la table 5.2, nous donnons quelques exemples de ces événements, pour des jeux vidéos, identifiés par le terme le plus représentatif, qui est naturellement le nom du jeu. Pour chacune des communautés, nous donnons ensuite sa date d'apparition et sa date de disparition. Afin de montrer la validité des communautés ainsi découvertes, nous donnons dans une troisième colonne la date de sortie commerciale de ce jeu au japon. Enfin, la quatrième colonne indique le nombre de jours entre la date de sortie du jeu et la date d'apparition de la communauté correspondante. Il ne s'agit que de quelques exemples pris parmi de nombreux autres.

Comme nous pouvons l'observer, la plupart des communautés apparaissent à une date proche de la date de sortie du jeu auquel elles correspondent. La communauté apparaît même avant la sortie du jeu. Lorsqu'un nouveau jeu à succès va bientôt être en vente, de nombreux utilisateurs échangent à propos du peu d'informations qu'ils ont à son propos. Dès que le jeu en en vente, de nombreux joueurs échangent des vidéos d'eux en train d'y jouer, montrant comment réussir une épreuve complexe par exemple, ou juste pour diffuser leur passage favori. Tout ceci conduit à de nombreuses vidéos taguées des mêmes mots clefs liés au jeu, et donc à l'apparition de la communauté correspondante.

5.3.5.2 Sujets génériques

Une seconde catégorie concerne les sujets existants sur le long terme, et de caractère plus général. Les utilisateurs ont tendance à partager beaucoup sur certains sujets, qui ne sont pas directement liés à un événement précis. Par exemple, nous donnons dans la table 5.3 certains de ces événements, avec leurs dates de commencement et leurs dates de fin lorsqu'elles existent.

Les communautés pour lesquelles il n'y a pas de date de fin sont les communautés qui ne sont pas encore terminées à la fin du jeu de données. Nous pouvons observer que certaines de ces communautés de termes existent depuis le début du jeu de données (février 2007) jusqu'à la fin, et donc que les utilisateurs partagent des vidéos sur ces sujets continuellement sur cette période. L'un des événements présenté cependant, celui concernant le catch, se termine avant la fin. Il est probable que l'ensemble des utilisateurs du réseau se sont peu à peu désintéressés de ce sujet au cours du temps. Ceci est confirmé en étudiant, par exemple, la diminution de la fréquence à laquelle le tag WWE (le nom de la ligue de catch la plus connue) est utilisé, en particulier après

Termes composant l'événement	Date de création	Date de fin
Chats, Chaton, Animal, vidéos de chat de Nico Nico	06/03/2007	-
J-League, Football, Sport	13/04/2007	-
Vocal, arrangement, Dojin Music	24/01/2008	-
Catch, WWE,WWF, Sport	18/07/2007	25/10/2008

TABLE 5.3 – *Exemples d'événements ayant une longue durée de vie. Pour la plupart, ces événements correspondent à des sujets plus généraux qu'un événement ponctuel.*

Sujet	Première apparition	Deuxième apparition	Troisième apparition
Noël	27/10/2007 À 14/01/2008	07/11/2008 À 19/01/2009	
Tour de France	17/06/2007 À 30/07/2007	02/06/2008 À 30/12/2008	
Jazz	19/09/2007 À 13/11/2007	26/11/2007 À 04/12/2008	21/03/2009 À -
Figure Skating	20/10/2007 À 11/09/2008	17/10/2008 À 04/01/2009	15/01/2009 À -

TABLE 5.4 – *Exemples d'événements répétitifs.*

la fin de la communauté.

5.3.5.3 Événements répétitifs

Une troisième catégorie que nous pouvons définir correspond aux évènements répétitifs. Certaines communautés sont détectées une première fois, disparaissent après un certain temps d'existence, se reforment un peu plus tard, disparaissent à nouveau, réapparaissent, et ainsi de suite. Nous donnons quelques exemples dans la table 5.4. Pour certains d'entre eux, ce comportement est complètement normal, et intéressant, comme par exemple pour Noël ou pour le Tour de France, qui a lieu tous les ans. Cependant, pour d'autres, cela peut être considéré comme un biais. Intuitivement, nous pourrions penser que ces événements devraient en constituer un seul, qui existerait sans interruption depuis le début de sa première occurrence jusqu'à la fin de la dernière, c'est à dire, pour la plupart, durant la totalité du jeu de données. Ces communautés sont probablement fragmentées parce que certains de leurs termes, durant une période donnée et sans vraiment de raisons, sont utilisés moins fréquemment. Cette utilisation plus rare conduit donc à couper le lien sémantique dans le réseau dynamique pour cette période, et donc dans certains cas, à faire disparaître la communauté. Dès que le lien réapparait, cela conduit à reformer la communauté, telle qu'elle était avant que le lien ne disparaisse. Si la fréquence de co-occurrence de deux termes est proche de la limite que nous avons utilisée dans notre paramétrage pour la création du graphe dynamique, ce phénomène peut se produire plusieurs fois, conduisant à une communauté existant par intermittence.

Du point de vue de la détection d'événements en temps réel, ce comportement n'est pas vraiment gênant : à un moment donné, ce sont bien les événements les plus populaires, ceux à propos desquels les utilisateurs échangent le plus de vidéos, qui forment des communautés. Cependant, nous perdons les informations à propos de l'historique de ces communautés. Du point de vue de l'algorithme de détection de communautés, la communauté de termes correspondant à Noël 2008 n'a pas de lien avec celle détectée pour Noël 2007.

Nous pouvons noter qu'il serait pertinent, pour certains réseaux, d'ajouter aux algorithmes de détection de communautés dynamiques la possibilité de détecter ce type de résurgence. Pour notre algorithme, par exemple, il serait possible de mémoriser, lorsqu'une communauté disparaît, son dernier état connu. Si une nouvelle communauté apparaissait un peu plus tard contenant ces mêmes nœuds, il serait alors possible de la considérer comme une résurgence de la communauté disparue. On peut remarquer que nous avions déjà noté l'intérêt d'une telle possibilité dans le premier cas applicatif.

5.3.5.4 Communautés de termes propres à la plateforme

Toutes les catégories d'événements présentés jusqu'ici étaient propres à un sujet spécifique. Parfois, le sujet était précis et limité (par exemple, un jeu), parfois un sujet plus large (Jazz, chats, etc.), mais il ne s'agissait jamais de communautés liées au réseau social lui-même. Les mêmes événements pourraient être détectés de manière similaire dans un autre réseau social, et ne reflètent que les intérêts des utilisateurs. Mais nous avons également eu la surprise de détecter certaines communautés qui n'étaient pas explicites par elles mêmes, et nous avons dû comprendre certaines particularités de la plateforme, certains usages des utilisateurs pour les comprendre. Nous les présentons ici. La première de ces communautés existe durant toute la période d'étude.

- *Divertissement, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10* : Sur Nico Nico Douga, il y a une limite à la longueur d'une vidéo. Mais beaucoup d'utilisateurs souhaitent partager des vidéos plus longues, parfois des films entiers par exemple, ou juste des vidéos personnelles particulièrement longues. Pour ce faire, ils utilisent une astuce : ils découpent le long fichier en plusieurs parties, et identifient chaque partie comme un élément du même ensemble. Ils utilisent, en sus des mots clefs habituels, des nombres correspondant à la quantité de séquences composant la vidéo complète, ainsi qu'au numéro de la vidéo actuellement visionnée dans cette chronologie. De cette pratique résulte la communauté trouvée.
- *@, ;, [,], test* : quelques communautés de cette sorte apparaissent durant la période de Novembre - Décembre 2008. Elles contiennent toutes le mot *test* et quelques caractères spéciaux. Nous avons cherché sur le réseau quelles vidéos étaient responsables de ces liens entre ces tags, mais, bien que nous en ayons trouvé la trace, les vidéos ne semblaient plus visibles. Il s'agissait donc probablement d'un test mené sur le réseau, peut-être pour vérifier la possibilité d'ajouter des caractères spéciaux dans les tags.
- *Animation, Jeux-Vidéos, Musique, Classement, Besoin de commentaires.*

Cette communauté apparaît à partir du milieu de la période étudiée. Elle contient trois des termes les plus utilisés dans le réseau, *Animation, Jeux-vidéos, Musique*, et le terme *Classement*. Pour comprendre cette communauté, il est nécessaire de savoir qu'un comportement courant sur Nico Nico Douga consiste à publier des vidéos qui sont des compilations d'extraits d'autres vidéos populaires sur le réseau, telles que les vidéos les plus regardées de la semaine par exemple, ou les vidéos favorites d'un utilisateur. Certains utilisateurs sont mêmes célèbres et suivis pour la publication de vidéos de ce type de bonne qualité, au travers desquelles d'autres personnes peuvent découvrir des vidéos intéressantes. Le mot clef pour identifier ce type de vidéos, qui

sont des best of, ou des Top 10, est le mot clef *Classement*.

5.3.5.5 Sous-événements

Si la plupart des jeux-vidéos, des groupes de musiques ou des sujets tels que le Jazz ou le Base-ball sont représentés par une et une seule communauté, d'autres, que nous aurions tendance à classer également comme un sujet unique, se retrouvent divisés en sujets multiples. Ceci est généralement le cas pour des sujets très appréciés, ce qui mène à un grand nombre de vidéos très différentes les unes des autres.

Deux de ces sujets particulièrement partagés sont le phénomène Hatsune Miku/VOCALOID et le jeu vidéo Idol Master.

Hatsune Miku est un phénomène très intéressant de collaboration entre utilisateurs sur un réseau social. VOCALOID est le nom d'un synthétiseur vocal de chant, un logiciel qui, en lui fournissant des paroles et une mélodie, produit une chanson correspondante. Hatsune Miku est le nom donné à un personnage imaginaire incarnant la voix des chants produits par VOCALOID 2, la première version de VOCALOID à devenir vraiment connue. Ce logiciel rencontra un grand succès, et en particulier sur Nico Nico Douga, où des centaines d'utilisateurs commencèrent à poster des milliers de vidéos. Certaines de ces vidéos contiennent une nouvelle chanson créée par l'utilisateur, ou une chanson populaire dont la voix a été refaite par Hatsune Miku. D'autres reprennent une chanson déjà mise en ligne, et modifient la vidéo, en ajoutant des images par exemple. Dans les travaux les plus évolués, une personne va composer la musique, une autre va créer le chant d'Hatsune Miku, et d'autres vont se charger du clip, allant jusqu'à créer des personnages en 3D, pouvant danser selon des chorégraphies inventées par d'autres utilisateurs.

Dans notre jeu de données, plus de 40 communautés sont détectées qui contiennent le terme *VOCALOID*. La plus longue commence le 19/08/2008 et continue jusqu'à la fin de la période étudiée. Cette période de début correspond bien avec la sortie commerciale de VOCALOID 2 (31/08/2008).

Cette communauté durable comporte de nombreux termes, notamment le nom d'autres personnages de la série VOCALOID (les successeurs de Hatsune Miku), ainsi que des termes assez généraux comme *musique, album original, Miku-vidéo*.

Dans le même temps, il y a de nombreuses autres communautés, généralement plus petites, et centrées sur des sujets plus précis. Par exemple, l'une d'elle est centrée sur le terme *VOCAROCK* (rock chanté par VOCALOID), une autre sur *Miku Miku Dance* (Un logiciel créé pour réaliser des vidéos en 3D de Hatsune Miku), et ainsi de suite. On peut noter qu'il pourrait être intéressant dans ce cas d'avoir une vision hiérarchique des communautés : toutes ces communautés ayant un rapport avec VOCALOID pourraient ainsi être regroupées comme une seule communauté de niveau supérieur. Mais, au niveau le plus bas, il est logique que ces différentes sous-communautés apparaissent comme distinctes, car il s'agit bien de centres d'intérêts différents.

Ces cas d'événements fragmentés en sous-événements semblent cependant assez limités.

5.3.6 Suivre l'évolution d'une communauté

L'un des points les plus intéressants de la détection de communautés dynamiques est la possibilité de suivre en détail l'évolution des communautés. Nous pouvons savoir précisément à quel moment un nœud rejoint ou quitte une communauté. Dans le cas présent, cela signifie que l'on peut savoir, pour un sujet populaire, quels termes liés sont apparus ou quels termes sont tombés en désuétude. La figure 5.14 montre un exemple de visualisation de l'évolution d'un sujet au cours du temps, pour la série de jeux vidéos appelée Metal Gear Solid.

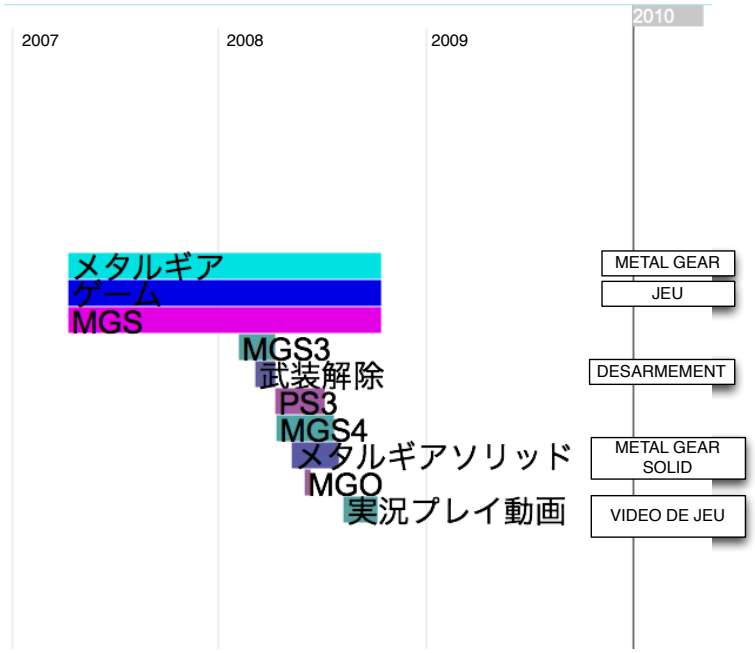


FIGURE 5.14 – *Détail de l'évolution d'une communauté.*

Chaque ligne horizontale représente un terme, commence lorsque ce terme est intégré à la communauté et se termine lorsque l'algorithme considère qu'il ne doit plus y être. Comme nous pouvons le voir, certains termes sont vraiment représentatifs et appartiennent donc à la communauté pour toute la période, alors que d'autres ne sont utilisés qu'à certains moments, et ne sont donc intégrés qu'épisodiquement.

Dans cet exemple, nous voyons que la communauté est originalement formée autour des termes *Jeu vidéo, Metal Gear* et *MGS*, l'acronyme du nom du jeu. Au début de l'année 2008, *MGS3* est ajouté à la communauté. Nous n'avons pas trouvé de raison évidente pour l'apparition de ce terme, qui correspond à un jeu publié en 2004, à part éventuellement la sortie proche d'un nouvel épisode de la série. Un peu plus tard, le terme *Désarmement* apparaît pour une courte période. MGS est un jeu d'infiltration militaire, dans lequel désarmer un adversaire est un acte difficile mais gratifiant, que les utilisateurs aiment partager en vidéo. Puis, en Avril 2008, les deux termes *PS3* et *MGS4* apparaissent simultanément. MGS4 est le nom du nouvel opus de la série, vendu à partir de Juin 2008, sur la console PS3. Peu de temps après, un nouveau terme apparaît, qui est juste une nouvelle façon d'écrire le nom du jeu en japonais. Le terme *MGO*, qui apparaît en Juin 2008, est un acronyme pour Metal Gear Online, le nom de la version multi-joueurs en ligne du jeu, commercialisée ce même mois. Finalement, le dernier terme, traduit en *vidéo de jeu*, est un terme beaucoup utilisé sur le réseau correspondant à des vidéos du jeu prises pendant qu'un joueur est en train d'y jouer (à la différence de vidéos publicitaires, ou d'extraits de cinématiques, par exemple).

Cette visualisation peut être générée automatiquement pour chaque communauté, ce qui permet une exploration aisée des résultats. Les communautés dont la durée de vie est courte comptent généralement peu de ces modifications, alors qu'il est courant pour les plus longues d'observer de nombreuses entrées et sorties de nœuds. Nous pouvons, de la même façon, visualiser l'ensemble des communautés, de manière à capturer d'un coup d'œil leurs durées, date de début et date de fin. Sur la figure 5.15, nous montrons un exemple de visualisation obtenu en ne conservant

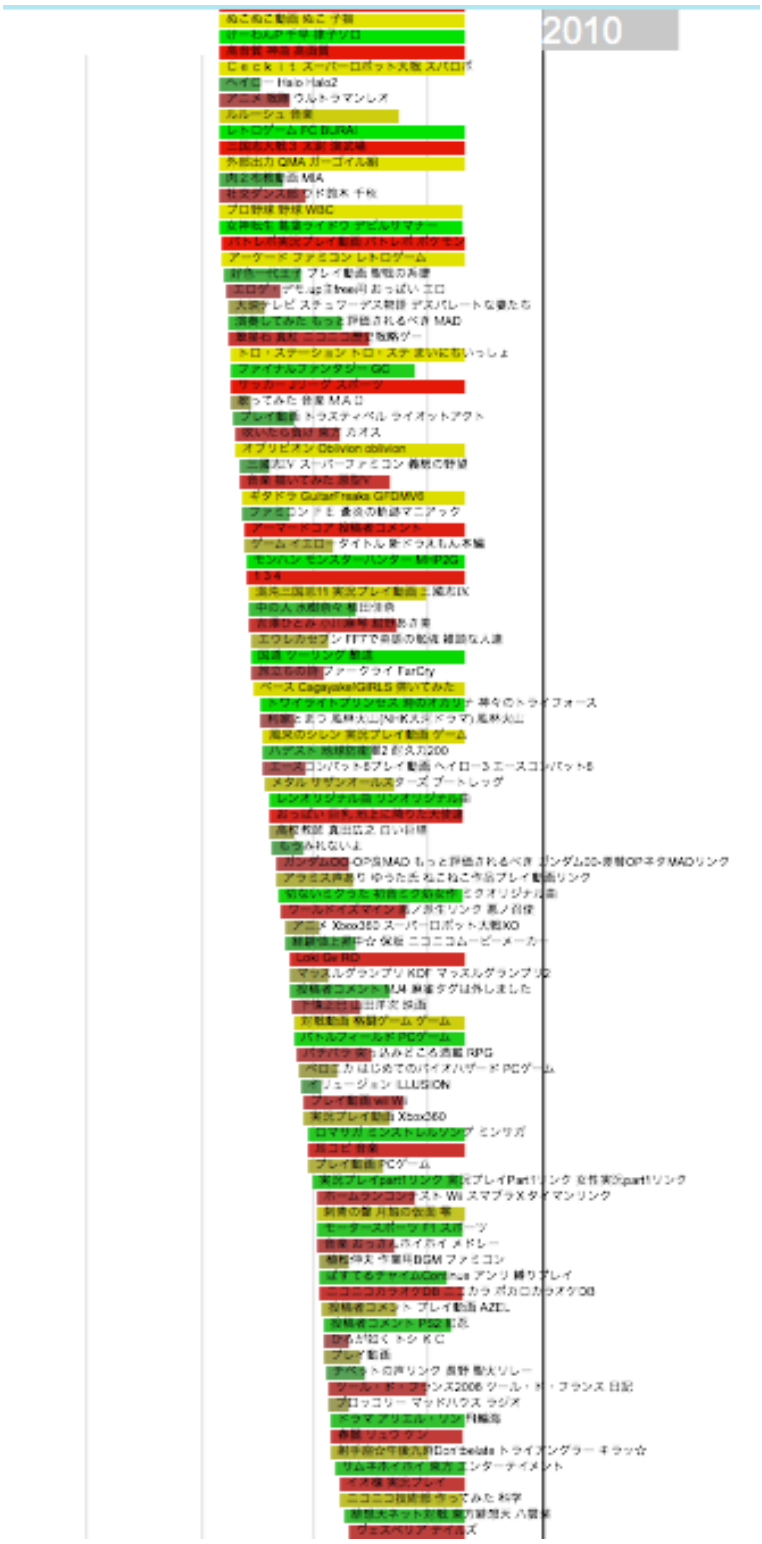


FIGURE 5.15 – *Visualisation de l'ensemble des communautés ayant plus de 10 termes. Chaque ligne horizontale correspond à une communauté, et l'on peut observer aisément leurs dates d'apparition et de disparition.*

que les communautés comptant plus de 10 termes. On peut faire la différence immédiatement entre des communautés ayant des caractéristiques différentes (les noms des communautés sont des sous-ensembles des noms des nœuds qu'elles contiennent, par souci de lisibilité).

5.3.7 Discussion

Nous avons vu dans les parties précédentes que l'utilisation de la détection de communauté en général et d'iLCD en particulier pour la détection d'événements dans un réseau social Web 2.0 semblait donner de très bons résultats. Comparés aux autres méthodes existantes, les avantages sont multiples. La possibilité de détecter des événements en temps réel, la possibilité d'avoir des termes apparaissant dans plusieurs événements, de pouvoir observer des événements à courte durée de vie et des événements sur le long terme et, enfin, de pouvoir suivre l'évolution d'un événement, sont autant d'atouts qui peuvent aider à mieux saisir le contenu des plateformes étudiées.

On remarquera que certains aspects pourraient encore être améliorés, comme, par exemple, proposer une décomposition hiérarchique des événements, comme dans le cas de VOCALOID. Il pourrait également être utile de pouvoir détecter des événements composés d'un seul terme ou de deux. En utilisant uniquement iLCD, pour qu'une communauté soit créée, il faut qu'au moins trois nœuds forment une clique. Si une communauté semble apparaître très clairement, mais n'est composée que de 2 termes, elle ne pourra pas être détectée. Cependant, prendre en compte de tels cas peut être risqué dans le sens où cela laisse beaucoup plus de place au bruit dans les données.

5.3.7.1 Adaptation à d'autres réseaux

On peut se demander si la même méthode ne pourrait pas être adaptée à d'autres réseaux sociaux Web 2.0, et en particulier à des réseaux plus complexes et plus riches comme Twitter. L'application de la même méthode à des services fonctionnant aussi sur le système des contenus identifiés par des mots clefs, tels que FlickR ou Delicious, ne poserait évidemment aucun problème : un réseau pourrait être créé à partir de ces mots-clefs, et le traitement serait exactement identique. En revanche, pour des plateformes dont les contenus sont écrits en langage naturel, tels que Twitter ou les statuts de Facebook, l'opération serait plus complexe. Un premier problème se poserait avec les termes non significatifs, tels que *et, ou, le*, qui provoqueraient beaucoup de bruit dans les données. Un autre problème serait les conjugaisons, pluriels et autres modifications de mots, qui empêcheraient un matching correct. Enfin, la présence de mots composés tels que Los Angeles ou Amy Winehouse est aussi un problème classique du traitement du langage naturel. Cependant, une méthode comme TwitterStand, qui est confrontée aux mêmes difficultés, semble obtenir des résultats convaincants en utilisant des solutions classiques issues du domaine du traitement automatique du langage. On peut voir en effet le problème réduit à l'énoncé suivant : comment transformer un message écrit en langage naturel en une suite de mots clefs pertinents. Une fois cette liste de mots clefs obtenue à partir d'un message tel qu'un tweet, on se retrouve dans une situation connue, pour laquelle la méthode que nous avons utilisée ici serait à nouveau applicable.

5.4 Bilan

Dans ce chapitre, nous avons montré que la méthode iLCD que nous avons proposée pouvait être efficacement employée pour traiter de grands réseaux de terrain issus du Web 2.0. En

utilisant ses propriétés dynamiques, nous avons obtenu des résultats particulièrement intéressant dont les méthodes antérieures n'étaient pas capables de rendre compte. La présence de communautés qu'il est facile d'identifier à des événements concrets, comme les communautés correspondant à des jeux vidéos, nous permet de vérifier que les communautés sont pertinentes à la fois du point de vue de leur sémantique et du point de vue temporel, comme nous l'avons montré avec la forte correspondance entre les dates de sorties de jeux et les dates de création des communautés correspondantes.

Enfin, nous avons montré un cas concret d'utilisation des algorithmes de détection de communautés dynamiques, et donc de l'intérêt de telles solutions. Un élément intéressant, qui est apparu dans les deux exemples, est la présence de communautés résurgentes, c'est à dire de communautés qui disparaissent à un moment donné, puis qui réapparaissent quelque temps plus tard sous une forme identique ou très proche. Pour les isards, il s'agissait de communautés liées aux saisons, tandis que pour Nico Nico Douga, il s'agissait soit de communautés périodiques (Noël, Tour de France, etc.), soit de communautés devenant régulièrement plus ou moins importantes, et pouvant donc disparaitre de nos données de base lorsqu'elles devenaient moins populaires. Nous proposons donc d'ajouter à la liste des opérations possibles sur les communautés dynamiques couramment admises un nouveau type d'opération, la résurgence. En reprenant les illustrations proposées par Palla et al. [PBV07], on peut représenter cette opération de résurgence de la manière présentée sur la figure 5.16.

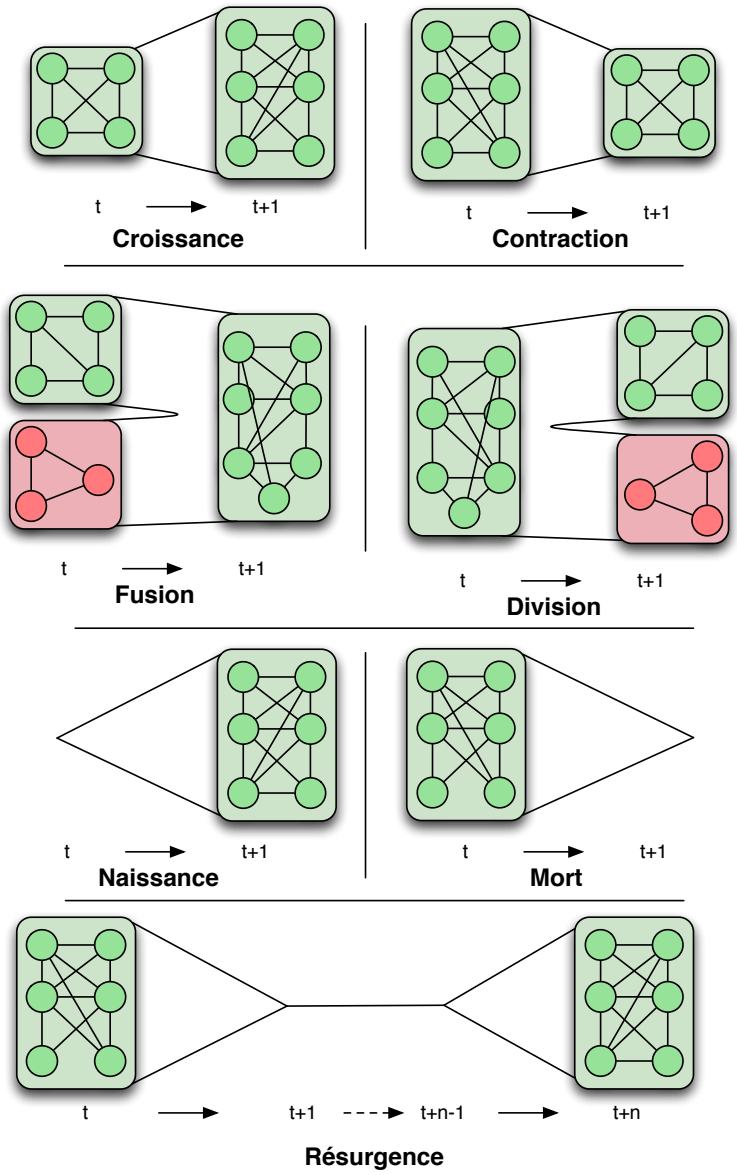


FIGURE 5.16 – *Représentation des opérations possibles sur les communautés, dont la résurgence.*

Conclusion

La détection de communautés dynamiques est, à notre avis, un domaine qui est encore dans une phase d'exploration, et pour lequel il faudra encore attendre quelques années avant d'arriver à un stade de maturation. Cette relative jeunesse du domaine a, d'une part, représenté un challenge et, d'autre part, été un facteur de motivation important. Nous avons en effet trouvé passionnant de se confronter à des problèmes qui ont encore peu été étudiés, et pour lesquels il reste beaucoup à inventer.

Nous avons en effet été frappés au commencement de la thèse par le fait que, malgré la présence de plusieurs algorithmes conçus pour trouver des communautés dynamiques, il n'en existait aucun capable de travailler directement sur des réseaux temporels, qui sont pourtant les plus adaptés pour représenter l'évolution d'un réseau. C'est dans cette optique que nous avons conçu le méta-algorithme iLCD et ses implémentations, iLCD-NRMH et iLCD-CDIE. Nous nous sommes inspirés des méthodes existantes, mais aucune n'était en mesure de résoudre exactement le problème que nous souhaitions étudier. Il nous a fallu de nombreux essais avant d'arriver à la version présentée dans ce mémoire, qui donne de bons résultats.

L'évaluation de la méthode a d'ailleurs été en elle-même une étape complexe. Il y avait deux aspects à analyser : premièrement, que les communautés trouvées soient cohérentes, comparativement aux autres algorithmes existants. Il fallait donc la comparer aux meilleurs algorithmes existants, qui étaient statiques. Nous avons commencé par une première validation en utilisant la méthode conventionnelle, c'est à dire essayer de détecter des communautés clairement définies sur un réseau généré. Cependant, nous nous sommes rendus compte, et d'autres travaux l'ont montré, que ces graphes générés, même les plus réalistes, étaient assez différents des graphes de terrain dans leurs propriétés. Comme expliqué en introduction, notre objectif était clairement d'obtenir des résultats convaincants sur des graphes de terrain. Nous avons donc cherché à utiliser des méthodes adaptées pour valider nos résultats sur de tels graphes. C'est le résultat de ces expérimentations que l'on trouve dans le troisième chapitre de ce mémoire. Là aussi, ce qui nous a motivé était le faible nombre de travaux portant sur ce problème. Comme nous l'avons présenté, il n'existe que peu de travaux proposant des méthodes pour comparer des découpages en communautés sur des graphes de terrain. Nous avons donc adopté deux approches, novatrices, qui nous ont permis de valider que les communautés trouvées par iLCD-CDIE étaient pertinentes. La première est basée sur deux métriques, de densité interne et de séparation des communautés, et permet de trouver des algorithmes moins peu pertinents sur un réseau donné. La seconde consistait à faire évaluer et comparer par des utilisateurs les résultats de différents algorithmes.

Mais il y avait un deuxième aspect à prendre en compte dans la validation de notre proposition : la composante dynamique. En effet, une fois montré qu'iLCD-CDIE pouvait détecter des communautés pertinentes sur des réseaux statiques, encore fallait-il montrer que l'évolution de ces communautés était elle-même pertinente. Le problème était l'absence d'approche publiée pour faire une telle validation. Les algorithmes de détection de communautés dynamiques précé-

dents travaillaient généralement sur un faible nombre d'étapes d'évolution. Certaines validaient leurs résultats en vérifiant que leur fonction objectif (la modularité) restait élevée, d'autres en testant sur de petits réseaux que les résultats étaient cohérents.

Une solution intéressante aurait été de tester notre algorithme sur des graphes générés, dynamiques et avec une structure en communautés, cependant il n'existe pas de générateurs de ce type à l'heure actuelle, et il serait délicat de valider un algorithme que l'on a conçu avec un générateur que nous aurions également proposé.

Comme, de plus, nous étions toujours intéressé par la validation de nos résultats sur des graphes de terrain, nous avons décidé de travailler sur des jeux de données réels, dynamiques et de grande taille. Il nous a donc fallu trouver ces jeux de données et un moyen de les étudier de manière fiable. Ceci nous a permis, d'une part, de valider que les résultats trouvés par iLCD étaient pertinents et exploitables et, d'autre part, que la détection de communautés dynamiques avait des applications concrètes, et pouvait être très intéressante dans de nombreux domaines. Les réseaux étudiés étant de grande taille et portant sur de longues périodes, ces tests ont permis également de confirmer qu'iLCD était adapté au traitement de grands réseaux de terrain.

Contributions principales

Au cours de cette thèse, nous avons travaillé sur des sujets divers, et nous pensons que plusieurs d'entre eux peuvent être dignes d'intérêt.

iLCD et ses implantations La première contribution est évidemment l'algorithme de détection de communautés dynamiques en lui-même (3.3). Si nous pensons que cet algorithme peut encore être amélioré, il s'agissait du premier algorithme capable de trouver des communautés sur des graphes temporels, et non pas sur des séquences d'instantanés. Il s'agit également du premier algorithme de détection de communautés dynamiques à avoir été appliqué sur des graphes ayant une dynamique aussi importante. Les premières et secondes implémentations ont été publiées, respectivement dans [CAH10] et [CA11].

Nature des communautés Les travaux que nous avons été amenés à faire sur la nature des communautés, et en particulier la distinction entre communautés intrinsèques et communautés relatives (3.2), pourraient venir participer à une définition précise de ce que l'on qualifie de communauté dans un réseau de terrain, qui reste encore à établir à ce jour.

Comparaison de communautés sur des réseaux de terrain Le premier outil que nous avons présenté dans la partie validation (4.1)est une solution complète pour explorer les solutions fournies par des algorithmes différents sur un même réseau de terrain. S'il existe maintenant de nombreux outils logiciels pour étudier les réseaux, il en existe peu adaptés à travailler sur les communautés, au-delà de la classique visualisation par coloration des nœuds, qui n'est pertinente que sur des réseaux très simples. Or, pour comprendre les résultats d'un algorithme sur un réseau de terrain, il est nécessaire de pouvoir l'étudier en détail.

La partie la plus intéressante de cet outil, que nous avons présenté en détail, permet de comparer, à l'aide de deux métriques complémentaires, les solutions fournies par des algorithmes différents, y compris des solutions avec recouvrement. Avec cet outil, qui pourrait être enrichi, il serait possible d'être capable de choisir, parmi plusieurs solutions, celle qui serait la plus efficace sur le réseau que l'on souhaite étudier. Car, comme nous l'avons expliqué, nous ne pensons pas qu'un algorithme puisse être le plus efficace pour tout type de réseaux, la nature des communautés à trouver pouvant être aussi diverse que la nature des réseaux.

Communautés résurgentes Lors de nos applications d'iLCD sur des réseaux de terrain, nous avons mis en évidence un phénomène qui semble assez courant : les communautés résurgentes (5.4). Les catégories d'évolution de communautés que l'on peut trouver sur un réseau dynamique avaient déjà été définies : croissance et affaiblissement de communautés, fusion et division, naissance et apparition étaient les six opérations possibles sur une communauté, et lorsque nous avons conçu iLCD, ce sont ces opérations que nous avons cherché à rendre possible. Or, il semble qu'il en existe une septième, la résurgence. Une communauté peut en effet disparaître pour un certain temps, avant de réapparaitre plus tard, exactement identique ou de manière légèrement différente. Si, lorsqu'elle réapparait, on peut la considérer comme une nouvelle communauté, il nous semble plus pertinent de tenir compte de son existence passé, comme cela apparait clairement dans le cas des isards et de Nico Nico Douga.

Autres contributions On peut trouver d'autres contributions dans cette thèse. Tout d'abord, un protocole permettant de transformer des séquences d'interactions en un réseau temporel (3.4.1.1), autorisant de plus à travailler sur des réseaux évoluant en temps réel. Deux propositions de visualisation de communautés dynamiques, ainsi que de l'évolution de ces communautés, sont présentées dans la section 6.1. Enfin, une méthode originale est également proposée pour la détection de sujets populaires sur les plateformes de réseau social (section 5.3, publié dans [CTHA12]).

Certaines applications présentées sont également dignes d'intérêt. L'application de la détection de communautés à des profils d'utilisateurs de réseaux sociaux (ici, Facebook) donne de bons résultats, et se révèle un outil intéressant pour comparer des algorithmes de détection de communautés (section 4.2, publié dans [CLA12]). L'autre application portant sur une plateforme de réseau social (Nico Nico Douga, section 5.3.4, publié dans [CTHA12]) nous fournit également de nombreux résultats intéressants, sur les différents types d'événements sur ces réseaux, et sur ce qu'il est possible de détecter en utilisant uniquement la détection de communautés dynamiques.

Perspectives

Il y aurait encore de nombreux travaux à faire, non seulement sur l'algorithme de détection de communautés proposé, mais aussi sur les autres propositions que nous avons faites.

Pour commencer par iLCD, les deux implémentations de ce méta-algorithme que nous avons proposées ici sont certainement imparfaites. Il existerait de nombreuses façon de les améliorer. Une première étape importante serait de proposer une décomposition hiérarchique des communautés. Comme expliqué dans le chapitre portant sur la conception de l'algorithme, les communautés détectées par l'implémentation actuelle sont nécessairement les communautés de plus bas niveau que l'on puisse trouver dans les réseaux, c'est à dire qu'elles ne peuvent pas être décomposées en communautés pertinentes de plus petite taille. Cela veut dire qu'il serait tout à fait possible de trouver des communautés de taille plus importante, en créant des communautés de communautés. Comme expliqué au chapitre validation, les algorithmes existants, à communautés relatives, ont tendance à trouver de petites communautés dans de petits réseaux et de grandes communautés dans de grands réseaux. En cherchant des communautés de communautés, on trouverait probablement des communautés plus proches de celles trouvées avec les autres algorithmes sur de grands réseaux. Nous avons déjà commencé à travailler sur cette question, et nous pensons qu'il ne serait pas très difficile de l'implémenter, en utilisant une méthode inspirée de celle que peut utiliser l'algorithme de Louvain, ou l'algorithme OSLOM, qui, tous deux, utilisent le mécanisme consistant à détecter des communautés de communautés.

Toujours par rapport à iLCD, il serait pertinent d'ajouter l'opération de division aux communautés, qui n'est pour l'instant que partielle, une division de communauté étant constituée d'une perte d'une fraction des nœuds suivie de la création d'une nouvelle communauté. L'opération de résurgence, identifiée durant la thèse, pourrait également être aisément ajoutée.

Un autre travail lié à la détection de communautés dynamiques, sur lequel nous espérons avoir l'occasion de travailler et qui nous semble désormais très important, est la conception d'un générateur de graphes temporels avec structure en communauté. Bien qu'il s'agisse d'une tâche complexe, un tel outil sera indispensable dans le futur afin de comparer les différentes méthodes. Outre le nécessaire réalisme des graphes générés, il faudrait également qu'il puisse reproduire les différentes opérations connues sur les communautés. Plus généralement, un travail de comparaison des différents algorithmes de détection de communautés dynamiques nous parait aujourd'hui indispensable, afin de mettre en exergue les forces et faiblesses des différentes approches.

En élargissant aux différents travaux que nous avons effectués durant la thèse, il y aurait également de nombreuses opportunités d'évolution. Le travail effectué sur Facebook, par exemple, pourrait prendre tout son sens en intégrant la dynamique de création des réseaux individuels. Le plus intéressant serait de ne plus étudier simplement les liens explicites entre les individus, mais la dynamique de leur communication, ce qui fournirait un cadre passionnant pour détecter des communautés dynamiques.

Enfin, il serait possible d'enrichir l'outil proposé pour comparer, sur des réseaux de terrain dont les communautés ne sont pas connues, des résultats d'algorithmes différents. Parmi les possibilités, on pourrait proposer d'autres métriques que les deux définies ici. On pourrait également utiliser une méthode naïve de référence, fournissant des communautés d'une taille donnée, pour évaluer le gain entre la solution donnée par chaque algorithme et cette solution naïve. En effet, comme on sait que des communautés plus grandes sont généralement moins bien définies intérieurement, et mieux séparées du reste du réseau que des communautés plus petites, disposer de solutions de référence avec des communautés de différentes tailles permettrait de mieux se rendre compte de la qualité des découpages.

Pour conclure, le travail présenté ici est une petite pierre apportée à l'édifice du vaste problème de la détection de communautés dans les graphes. Nous espérons que d'autres y trouveront un intérêt.

6

Annexes

Contents

6.1 Visualisation de communautés dynamiques

Le résultat obtenu lorsque l'on utilise un algorithme de détection de communautés dynamiques sur un réseau temporel est, comme on l'a vu, complexe. Si l'on souhaite utiliser ce résultat, l'étudier, ou simplement essayer de le comprendre, il peut être utile de le visualiser. Le problème est que le résultat auquel on a affaire est dynamique.

6.1.1 Représentation dynamique de communautés dynamiques

Une première solution que nous avons utilisée, la plus simple, consiste à représenter l'évolution des communautés sous la forme d'une vidéo.

De telles approches existent déjà, notamment pour la visualisation d'interactions dans des réseaux sociaux, pouvant même utiliser une représentation en 3 dimensions pour aider l'utilisateur à mieux apprécier un réseau complexe [GPK12a]

Dans notre cas, chaque image de la vidéo correspond à une étape de l'évolution du réseau, donc à un réseau statique. Les communautés sur ce réseau statique sont représentées de manière simple : à chaque communauté est attribuée une couleur unique, et à chaque nœud est attribué la couleur correspondant à sa communauté. Dans le cas d'appartenances multiples, le nœud est représenté par un diagramme en camembert dont chaque tranche correspond à une communauté.

De manière à ce que le graphe reste cohérent au cours de la vidéo, les nœuds sont positionnés à l'aide d'un modèle masse ressort qui se met à jour à chaque modification du réseau. Les nœuds et liens non modifiés par la dernière modification du réseau restent donc approximativement à la même place. La bibliothèque GraphStream [DGO+07], spécialement développée pour travailler avec des graphes dynamiques, a été utilisée pour développer cet outil.

On peut voir quelques illustrations tirées d'une vidéo représentant l'évolution d'un réseau de terrain sur la figure 6.1 Cette vidéo peut être générée automatiquement à partir du format de fichier TNF que nous avons décrit précédemment.

L'inconvénient principal de cette méthode est qu'elle ne permet pas de visualiser de grands réseaux complexes. Comme pour toute visualisation de réseaux, les graphes très denses et comportant beaucoup de nœuds ne peuvent être représentés sous cette forme, *a fortiori* un réseau dynamique.

6.1.2 Représentation statique de communautés dynamiques

Pour visualiser plus efficacement des communautés dynamiques, une autre solution consiste à en proposer une visualisation statique. Cette approche a déjà été adoptée dans la littérature auparavant. Deux solutions ont été proposées :

- Mucha et al. [MRM+10] utilisent une visualisation dans laquelle les nœuds ont une place fixe sur l'axe des ordonnées, tandis que le temps est sur l'axe des abscisses. 6.2. Cette visualisation permet de bien voir le renouvellement des nœuds des communautés, elle a cependant plusieurs inconvénients : il est difficile de trouver une information précise dessus, elle est parfois difficile à lire, et elle reste limitée à de petits graphes. Surtout, elle ne pourrait pas être utilisé pour des communautés vraiment complexes, dans lesquelles les nœuds pourraient appartenir à plusieurs communautés, et en changeraient. La position de chaque nœud sur l'axe vertical est en effet décidée en fonction de la communauté à laquelle il appartient, et c'est cette position qui assure la visibilité des communautés. Si un nœud appartient fortement à plusieurs communautés, il n'est plus possible de le placer à une position pertinente.

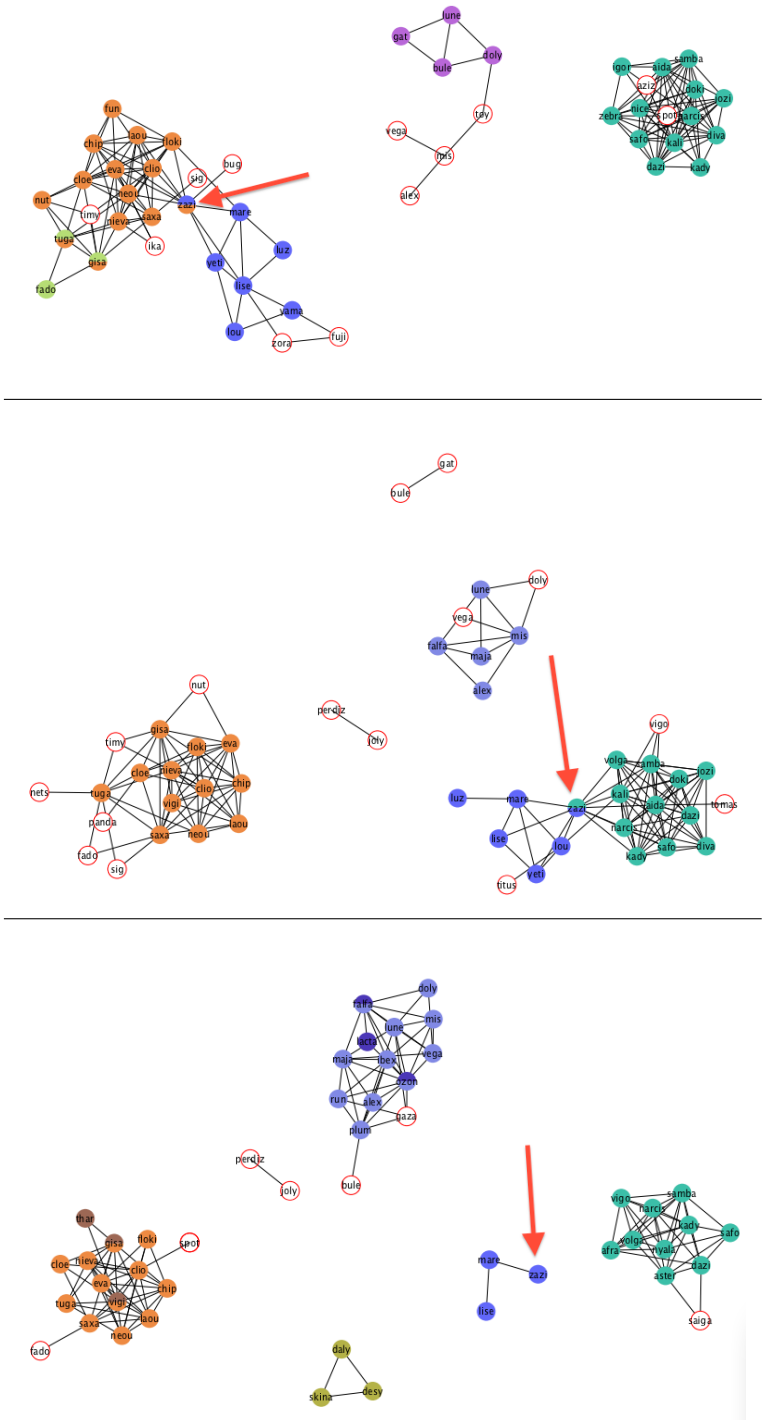


FIGURE 6.1 – *Captures d'écran de la visualisation dynamique de communautés dynamiques. Les communautés restent faciles à suivre malgré de multiples étapes d'évolution. On peut observer des nœuds en recouvrement, tel que celui indiqué par une flèche dans les deux premières captures d'écran*

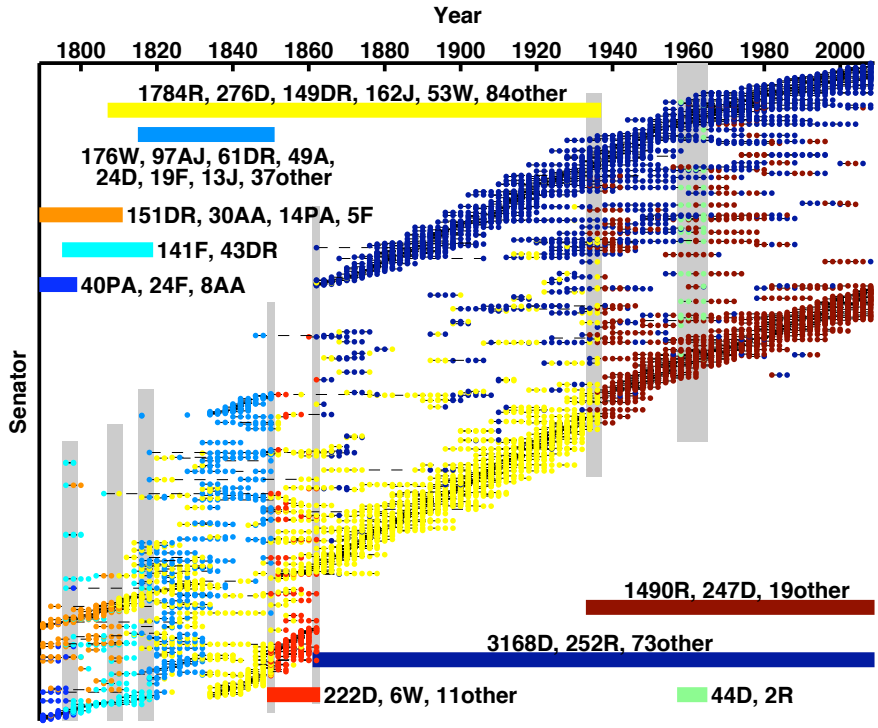


FIGURE 6.2 – *Visualisation de communautés dynamiques, par Mucha et al. [MRM+10]*

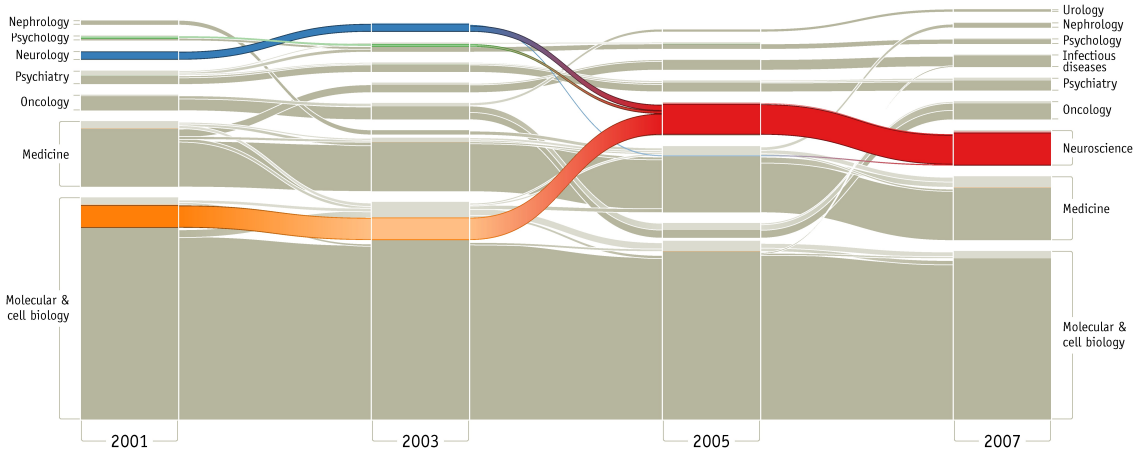


FIGURE 6.3 – *Visualisation de communautés dynamiques, par Rosvall et al. [RB10]*

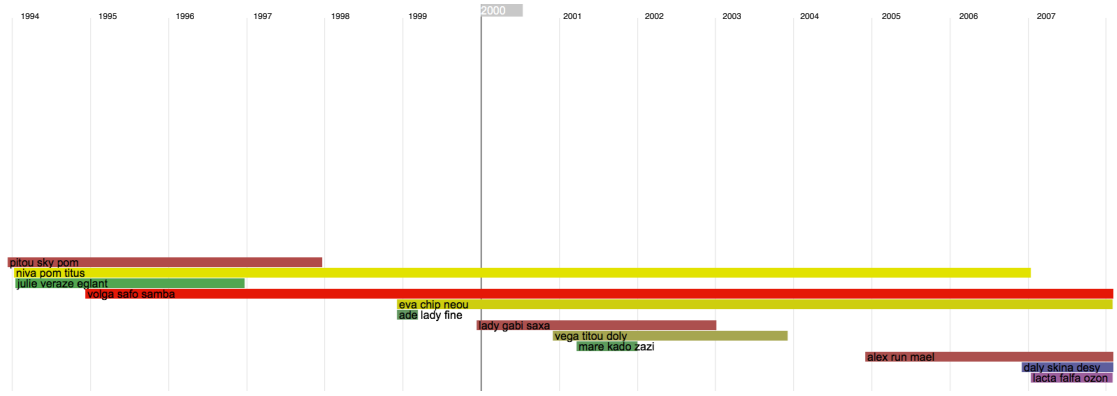


FIGURE 6.4 – *Visualisation statique de communautés dynamiques. Chaque ligne horizontale correspond à une communauté. On peut observer les dates de début et de fin des communautés. Un clic sur une de ces communautés affiche le détail de son évolution.*

– Rosvall et al [RB10] proposent une visualisation de très grande qualité, très pertinente dans le cas de réseaux contenant relativement peu de communautés (Figure 6.3. Le point le plus intéressant est que l'on peut représenter des fusions et des divisions de communautés. Le problème est cependant qu'ici aussi, il n'est pas possible de prendre en compte le recouvrement, un nœud ne peut appartenir qu'à une et une seule communauté. De plus, comme on peut le voir sur l'illustration, cette méthode n'est adaptée qu'à des graphes dont l'évolution n'est constituée que de quelques instantanés. Dans le cas de graphes contenant de très nombreuses étapes, la visualisation deviendrait trop complexe pour vraiment être utile. (Croisement des différentes lignes, etc.)

La solution que nous avons adoptée devait répondre à trois exigences :

– Permettre de prendre en compte des communautés avec recouvrement
– Permettre de suivre le détail de l'évolution des communautés
– Permettre de représenter de grands graphes

Nous avons finalement opté pour une visualisation interactive. Au premier niveau de visualisation, seules les communautés sont représentées, comme montré sur la figure 6.4. Chaque ligne horizontale correspond à une communauté, et on voit clairement ses dates de naissance et de disparition. Il peut y avoir de nombreuses communautés sans que cela ne rende la visualisation complexe.

Il est ensuite possible d'observer le détail de l'évolution de chacune de ces communautés (via un simple clic dans la version intéractive). La visualisation est alors semblable, mais chaque ligne correspond désormais à un nœud, et ses dates de début et de fin correspondent au moment où il a intégré puis quitté la communauté.

Le recouvrement, bien que n'étant pas expressément représenté, ne pose donc pas de problème de visualisation : le nœud apparait dans le détail de chacune des communautés auxquelles il appartient.

6.2 Formats de fichiers

Pour des algorithmes de détection de communautés classique, on sait que le réseau à fournir en entrée, sur lequel l'algorithme va travailler, est un réseau statique. Il existe de nombreux formats standards pour représenter des réseaux, certains permettant de limiter le volume de

données pour les grands graphes, tandis que d'autres vont permettre de décrire quantité de détails, tels que les propriétés des liens et des nœuds. Une liste non exhaustive comprendrait les formats Pajek [BM98], graphML [BELP10], Edge List, lgl, ncol, GEXF [BHJ09].

6.2.1 Formats d'entrée

Il n'existe à notre connaissance qu'un seul format standard permettant de décrire un réseau temporel autrement que comme une séquence d'instantanés : il s'agit du format GEXF, utilisé dans le logiciel Gephi. Ce format pourrait être utilisé pour étudier les communautés dynamiques, mais il présente deux inconvénients : d'une part, étant basé sur du XML, il a l'inconvénient d'être assez verbeux. Les fichiers décrivant de grands réseaux évoluant fortement peuvent donc rapidement devenir très volumineux. Deuxièmement, et c'est le plus gênant, le graphe temporel n'est pas organisé de manière chronologique. Chaque lien est défini de manière indépendante, comme pour un graphe statique, et les périodes d'existence du lien sont définies comme attributs dans la description du lien.

En conséquence, traiter l'évolution dans son ordre logique nécessite de charger préalablement tout le graphe en mémoire. Nous avons donc décidé de nous orienter vers un choix qui nous semblait plus rationnel dans notre cas, et nous avons donc défini un format de fichier spécifiquement conçu pour décrire l'évolution d'un graphe, que nous avons nommé TNF (.tnf) pour Temporal Network Format. Ce format est pour l'instant limité aux graphes non dirigés et non pondérés mais pourrait être facilement étendu. Il retranscrit simplement le formalisme décrit en section 3.1.1.

Chaque ligne correspond à un événement sur le réseau, et le fichier peut être parcouru dans l'ordre pour connaître la suite d'événements ayant amené à l'état actuel du réseau. Les événements peuvent être de deux types : création de lien ou suppression de lien. Une ligne correspondant à une création de lien commencera par le caractère + quand une ligne correspondant à une suppression sera notée - . Le lien que l'on crée ou supprime est simplement représenté par les identifiants du couple de nœuds affecté. Enfin, on peut insérer à tout moment une indication précise de temps en utilisant le caractère # en début de ligne. Un exemple de fichier sera donc :

```
#01/01/2012
+ a b
+ a c
#05/01/2012
+ a d
-a c
#06/01/2012
+ b d
```

On notera que l'apparition et la disparition de nœuds ne sont pas représentés. On considère simplement qu'un nœud n'a d'existence que tant qu'il possède au moins un lien dans le réseau. Nous avons pris cette décision car, dans le domaine de la détection de communautés, un nœud sans lien ne joue pas de rôle.

6.2.2 Formats de sortie

La question du format de sortie est plus complexe. Dans le cas des réseaux statiques, les résultats des algorithmes de détection de communautés sont un (ou plusieurs) ensemble de communautés. Un ensemble de communautés est représenté par un fichier, et les communautés peuvent être représentées de différentes manières. Les formats les plus utilisés sont :

– Une ligne par communauté, une ligne étant constituée par la suite des nœuds de la communauté séparés par un caractère prédéfini, tel qu'une tabulation.
– Des couples nœuds/communautés, où chaque ligne correspond à l'appartenance d'un nœud à une communauté.

Dans le cas des communautés dynamiques, ces solutions ne sont pas pertinentes. Dans la littérature, la plupart des méthodes n'expliquent pas sous quelle forme sont fournis les résultats, et à notre connaissance aucun autre algorithme de détection de communautés dynamiques n'est fourni sous la forme d'un logiciel directement utilisable fournissant un résultat dynamique exploitable.

Des questions se posent pourtant lorsque l'on veut fournir des communautés dynamiques sous une forme exploitable :

– Comment décrire une communauté qui change au cours du temps : une communauté dynamique ne peut pas être réduite à un ensemble de nœuds, c'est une succession de changements dans un ensemble de nœuds. Le résultat doit refléter ces changements.
– Si l'on fournit tous les détails de l'évolution des communautés, le résultat devient très vite inexploitable, car il y a trop d'informations. Dès lors, il faut envisager d'avoir nécessairement à utiliser un outil logiciel pour manipuler ces résultats. Or, de tels outils n'existent pas encore, du moins pas pour un grand nombre de communautés évoluant fortement.

Dans notre recherche d'un bon compromis entre ces deux problèmes, nous avons été amenés à produire trois formats de résultats ainsi que des outils pour la visualisation de communautés dynamiques. Nous présenterons ces outils dans la section visualisation de communautés dynamiques. Nous allons ici décrire les formats de sortie que nous avons définis et que l'on peut générer à partir de notre algorithme.

6.2.2.1 Résumé condensé

Ce fichier ne contient pas toutes les informations correspondant aux communautés détectées, mais en est un condensé. Il permet de fournir un premier résultat qui peut être parcouru manuellement, sans logiciel dédié. Pour chaque communauté, on va décrire sa date d'apparition et sa date de disparition (aucune si la communauté existe encore à la fin du jeu de données). Pour décrire son contenu, on va simplement citer tous les nœuds ayant été intégrés à un moment où à un autre de l'évolution de la communauté. On reprend le système du fichier contenant une communauté par ligne. Le format d'une communauté est le suivant :

```
Date de début - date de fin : noeud1 noeud2 noeud3 ...
```

Et ce pour chaque communauté.

6.2.2.2 Description statique des communautés dynamiques

Un fichier XML peut simplement être utilisé pour décrire toutes les étapes d'évolution des communautés. On obtient un fichier qui est difficilement lisible par un être humain, mais qui pourrait facilement être traité par un logiciel, et qui a l'avantage de décrire le résultat de l'algorithme de manière exhaustive. Ce fichier est composé de la manière suivante :

```
//section décrivant toutes les communautés
<communities>

//description d'une communauté
<c>
```

```
<idC> identifiant </idC>
<beginning>apparition</ beginning >
<end>apparition</ end >
<name>com1</name>
</c>

//fin de la section décrivant les communautés
</communities>

//début de la section décrivant toutes les appartenances de liens à des communautés
<nodes>

//description de l'appartenance d'un nœud donné à une communauté donnée
<n>
<idNode> identifiant </idNode>
<id_com> identifiant de la communauté parent </id_com>
<date1> date d'apparition dans la communauté </date1>
<date2> date de fin d'appartenance à la communauté </date2>
</n>

//fin de la section décrivant les nœuds
</nodes>

//début de la section décrivant les fusions
<mergings>

//description d'une fusion entre deux communautés
<m>
<id_kept> identifiant de la communauté qui perdure</id_kept>
<id_lost> identifiant de la communauté englobée par l'autre </id_lost>
<date> date de la fusion </date>
</m>

</mergings>
```

6.2.2.3 Description par modifications successives

Si la version précédente avait l'avantage d'utiliser le format XML, standard et donc facile à lire par un logiciel, elle présentait aussi les mêmes inconvénients que nous avons reproché précédemment au format de description de graphes dynamiques GEXF, à savoir la verbosité du fichier, qui peut être pénalisante sur de grands graphes, ainsi que la nécessité de charger tout le fichier pour pouvoir le traiter. Dans le cas de communautés dynamiques, il peut parfois être pratique de trouver les informations dans leur ordre d'apparition. Dans ce cas, nous avons proposé une extension au format TNF présenté précédemment. Il est possible de n'utiliser que les primitives présentées ici, de manière à décrire uniquement les communautés dynamiques, ou alors de synthétiser dans un même fichier TNF la description de l'évolution du graphe et de ses communautés, en vue, par exemple, d'une visualisation.

```
//descripteur de date des événements
```

```
#date de l'événement

//événement d'ajout du noeud node à la communauté com.
+nc node com

//événement de suppression du noeud node de la communauté com.
-nc node com

//événement de la création de la communauté com
+c com

//événement de la disparition de la communauté com
-c com

//événement de fusion des communauté com1 et com2
= c1 c2
```

On peut noter que les événements complexes, tels que la fusion de communautés — amenant à supprimer l'une des communautés et à, éventuellement, intégrer une partie de ses nœuds dans l'autre — peuvent être décrits par une succession d'événements ayant lieu au même instant.

Bibliographie

[ABL10] Y.Y. Ahn, J.P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307) :761–764, 2010.

[AG10a] T. Aynaud and J.L. Guillaume. Long range community detection. In *LAWDN-Latin-American Workshop on Dynamic Networks*, 2010.

[AG10b] T. Aynaud and J.L. Guillaume. Static community detection algorithms for evolving networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on*, pages 513–519. IEEE, 2010.

[BA99] A.L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439) :509–512, 1999.

[BB05] J.P. Bagrow and E.M. Bollt. Local method for detecting communities. *Physical Review E*, 72(4) :046108, 2005.

[BDG+07] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On finding graph clusterings with maximum modularity. In *Graph-Theoretic Concepts in Computer Science*, pages 121–132. Springer, 2007.

[BE00] S.P. Borgatti and M.G. Everett. Models of core/periphery structures. *Social networks*, 21(4) :375–395, 2000.

[BELP10] U. Brandes, M. Eiglsperger, J. Lerner, and C. Pich. Graph markup language (graphml). *Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications). Chapman & Hall/CRC*, 2010.

[Ben10] J. Benhardus. Streaming trend detection in twitter. *National Science Foundation REU for Artificial Intelligence, Natural Language Processing and Information Retrieval, University of Colarado*, 2010.

[BGLL08] V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 2008(10) :P10008, 2008.

[BGMI05] J. Baumes, M. Goldberg, and M. Magdon-Ismail. Efficient identification of overlapping communities. *Intelligence and Security Informatics*, pages 1–5, 2005.

[BGPP03] C. Bernon, M.P. Gleizes, S. Peyruqueou, and G. Picard. Adelfe : A methodology for adaptive multi-agent systems engineering. *Engineering Societies in the Agents World III*, pages 70–81, 2003.

[BHJ09] M. Bastian, S. Heymann, and M. Jacomy. Gephi : An open source software for exploring and manipulating networks. In *International AAAI conference on weblogs and social media*. Citeseer, 2009.

[BHKL06] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks : membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54. ACM, 2006.

[BJRF07] M. Ben Jdidia, C. Robardet, and E. Fleury. Communities detection and analysis of their dynamics in collaborative networks. In *Digital Information Management, 2007. ICDIM'07. 2nd International Conference on*, volume 2, pages 744–749. IEEE, 2007.

[BM98] V. Batagelj and A. Mrvar. Pajek-program for large network analysis. *Connections*, 21(2) :47–57, 1998.

[BNG11] H. Becker, M. Naaman, and L. Gravano. Beyond trending topics : Real-world event identification on twitter. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM11)*, 2011.

[CA11] R. Cazabet and F. Amblard. 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*, volume 2, pages 402–408. IEEE, 2011.

[CAH10] R. Cazabet, F. Amblard, and C. Hanachi. Detection of overlapping communities in dynamical social networks. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 309–314. IEEE, 2010.

[CBG+07] J.P. Crampe, R. Bon, J.F. Gerard, E. Serrano, P. Caens, E. Florence, and G. Gonzalez. Site fidelity, migratory behaviour, and spatial organization of female isards (rupicapra pyrenaica) in the pyrenees national park, france. *Canadian journal of zoology*, 85(1) :16–25, 2007.

[CCDP97] J.P. Crampe, J.C. Caens, J.L. Dumerc, and D. Pepin. La masse corporelle comme indicateur de la condition physique hivernale de l'isard, rupicapra pyrenaica (artiodactyla, bovidae). *Mammalia*, 61(1) :73–86, 1997.

[CFQS12] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. 2012.

[CGL02] J.P. Crampe, J.M. Gaillard, and A. Loison. L'enneigement hivernal : un facteur de variation du recrutement chez l'isard (rupicapra pyrenaica pyrenaica). *Canadian journal of zoology*, 80(7) :1306–1312, 2002.

[CHX09] S.Y. Chan, P. Hui, and K. Xu. Community detection of time-varying mobile social networks. *Complex Sciences*, pages 1154–1159, 2009.

[CK01] A. Condon and R.M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2) :116–140, 2001.

[CKT06] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 554–560. ACM, 2006.

[CLA12] R. Cazabet, M. Leguistin, and F. Amblard. Automated community detection on social networks : useful ? efficient ? asking the users. In *Proceedings of the 4th International Workshop on Web Intelligence & Communities*, pages 1–6. ACM, 2012.

[CLG+06] J.P. Crampe, A. Loison, J.M. Gaillard, E. Florence, P. Caens, and J. Appolinaire. Patrons de reproduction des femelles d'isard dans une population non chassee et consequences demographiques. *Canadian journal of zoology*, 84(9) :1263–1268, 2006.

[CLSW10] W. Chen, Z. Liu, X. Sun, and Y. Wang. A game-theoretic framework to identify overlapping communities in social networks. *Data Mining and Knowledge Discovery*, 21(2) :224–240, 2010.

[CNM04] A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6) :066111, 2004.

[CSCC05] A. Capocci, V.D.P. Servedio, G. Caldarelli, and F. Colaiori. Detecting communities in large networks. *Physica A : Statistical Mechanics and its Applications*, 352(2) :669–676, 2005.

[CSLF10] D. Chen, M. Shang, Z. Lv, and Y. Fu. Detecting overlapping communities of weighted networks via a local algorithm. *Physica A : Statistical Mechanics and its Applications*, 389(19) :4177–4187, 2010.

[CTHA12] R. Cazabet, H. Takeda, M. Hamasaki, and F. Amblard. Using dynamic community detection to identify trends in user-generated content. *Social Network Analysis and Mining*, 2 :361–371, 2012.

[CVdBB+10] C. Cattuto, W. Van den Broeck, A. Barrat, V. Colizza, J.F. Pinton, and A. Vespignani. Dynamics of person-to-person interactions from distributed rfid sensor networks. *PloS one*, 5(7) :e11596, 2010.

[CWJ+10] Z. Chen, K.A. Wilson, Y. Jin, W. Hendrix, and N.F. Samatova. Detecting and tracking community dynamics in evolutionary networks. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 318–327. IEEE, 2010.

[DA05] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical review E*, 72(2) :027104, 2005.

[DDGA06] L. Danon, A. Díaz-Guilera, and A. Arenas. The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics : Theory and Experiment*, 2006(11) :P11010, 2006.

[DDGDA05] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics : Theory and Experiment*, 2005(09) :P09008, 2005.

[DGO+07] A. Dutot, F. Guinand, D. Olivier, Y. Pigné, et al. Graphstream : A tool for bridging the gap between complex systems and dynamic graphs. In *Emergent Properties in Natural and Artificial Complex Systems. Satellite Conference within the 4th European Conference on Complex Systems (ECCS'2007)*, 2007.

[DM04] L. Donetti and M.A. Munoz. Detecting network communities : a new systematic and efficient algorithm. *Journal of Statistical Mechanics : Theory and Experiment*, 2004(10) :P10012, 2004.

[DMSGK06] G. Di Marzo Serugendo, M.P. Gleizes, and A. Karageorgos. Self-organisation and emergence in mas : An overview. *Informatica*, 30(1) :45–54, 2006.

[EEBL11] P. Expert, T.S. Evans, V.D. Blondel, and R. Lambiotte. Uncovering space-independent communities in spatial networks. *Proceedings of the National Academy of Sciences*, 108(19) :7663–7668, 2011.

[EL09] TS Evans and R. Lambiotte. Line graphs, link partitions, and overlapping communities. *Physical Review E*, 80(1) :016105, 2009.

[EVDO02] A.J. Enright, S. Van Dongen, and C.A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic acids research*, 30(7) :1575–1584, 2002.

[FB07] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1) :36–41, 2007.

[FBS08] T. Falkowski, A. Barth, and M. Spiliopoulou. Studying community dynamics with an incremental graph mining algorithm. *AMCIS 2008 Proceedings*, page 29, 2008.

[FCF+11] A. Friggeri, G. Chelius, E. Fleury, et al. Egomunities, exploring socially cohesive person-based communities. In *NetSci 2011 The International School and Conference on Network Science*, 2011.

[Fer99] J. Ferber. *Multi-agent systems : an introduction to distributed artificial intelligence*, volume 33. Addison-Wesley Reading, MA, 1999.

[FLG00] G.W. Flake, S. Lawrence, and C.L. Giles. Efficient identification of web communities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–160. ACM, 2000.

[FLZ+07] Y. Fan, M. Li, P. Zhang, J. Wu, and Z. Di. Accuracy and precision of methods for community identification in weighted networks. *Physica A : Statistical Mechanics and its Applications*, 377(1) :363–372, 2007.

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

[GC01] G. Gonzalez and J.P. Crampe. Mortality patterns in a protected population of isards (rupicapra pyrenaica). *Canadian journal of zoology*, 79(11) :2072–2079, 2001.

[GDC10] D. Greene, D. Doyle, and P. Cunningham. Tracking the evolution of communities in dynamic social networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*, pages 176–183. IEEE, 2010.

[GMNN08] V. Gudkov, V. Montealegre, S. Nussinov, and Z. Nussinov. Community detection in complex networks by dynamical simplex evolution. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 78(1 Pt 2) :016113, 2008.

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

[Gof05] E. Goffman. *The presentation of self in everyday life*. Penguin Harmondsworth, 2005.

[GPK12a] N. Greffard, F. Picarougne, and P. Kuntz. Immersive dynamic visualization of interactions in a social network. In *Challenges at the Interface of Data Analysis, Computer Science, and Optimization : Proceedings of the 34th Annual Conference of the Gesellschaft für Klassifikation e. V., Karlsruhe, July 21-23, 2010*, page 255. Springer, 2012.

[GPK12b] N. Greffard, F. Picarougne, and P. Kuntz. Visual community detection : An evaluation of 2d, 3d perspective and 3d stereoscopic displays. In *Graph Drawing*, pages 215–225. Springer, 2012.

[Gre10] S. Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10) :103018, 2010.

[GSPA04] R. Guimera, M. Sales-Pardo, and L.A.N. Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2) :025101, 2004.

[Har07] E. Hargittai. Whose space ? differences among users and non-users of social network sites. *Journal of Computer-Mediated Communication*, 13(1) :276–297, 2007.

[Has06] M.B. Hastings. Community detection as an inference problem. *Physical Review E*, 74(3) :35102, 2006.

[HHSG11] F. Havemann, M. Heinz, A. Struck, and J. Gläser. Identification of overlapping communities and their hierarchy by locally calculating community-changing resolution levels. *Journal of Statistical Mechanics : Theory and Experiment*, 2011(01) :P01023, 2011.

[HKKS04] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Tracking evolving communities in large linked networks. *Proceedings of the national academy of sciences of the United States of America*, 101(Suppl 1) :5249–5253, 2004.

[HLL83] P.W. Holland, K.B. Laskey, and S. Leinhardt. Stochastic blockmodels : first steps. *Social networks*, 5(2) :109–137, 1983.

[HP09] J. Haynes and I. Perisic. Mapping search relevance to social networks. In *Proceedings of the 3rd Workshop on Social Network Mining and Analysis*, page 2. ACM, 2009.

[HS09] P. Hui and N. Sastry. Real world routing using virtual world information. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 1103–1108. IEEE, 2009.

[HS12] P. Holme and J. Saramäki. Temporal networks. *Physics Reports*, 2012.

[HTN08] M. Hamasaki, H. Takeda, and T. Nishimura. Network analysis of massively collaborative creation of multimedia contents : case study of hatsune miku videos on nico nico douga. In *Proceedings of the 1st international conference on Designing interactive user experiences for TV and video*, pages 165–168. ACM, 2008.

[JKV01] B. Jouve, P. Kuntz, and F. Velin. Extraction de structures macroscopiques dans des grands graphes par une approche spectrale. volume 1, pages 173–184, 2001.

[Kel10] S. Kelley. *The existence and discovery of overlapping communities in large-scale networks*. PhD thesis, RENSSELAER POLYTECHNIC INSTITUTE, 2010.

[KJ11] Y. Kim and H. Jeong. Map equation for link communities. *Physical Review E*, 84(2) :026110, 2011.

[KK89] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1) :7–15, 1989.

[KKKS08] J.M. Kumpula, M. Kivelä, K. Kaski, and J. Saramäki. Sequential algorithm for fast clique percolation. *Physical Review E*, 78(2) :026109, 2008.

[KL70] B.W. Kernighan and S. Lin. An eflicient heuristic procedure for partitioning graphs. *Bell system technical journal*, 1970.

[Kle03] J. Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4) :373–397, 2003.

[KVV04] R. Kannan, S. Vempala, and A. Vetta. On clusterings : Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3) :497–515, 2004.

[LBA11] P. Latouche, E. Birmelé, and C. Ambroise. Overlapping stochastic block models with application to the french political blogosphere. *The Annals of Applied Statistics*, 5(1) :309–336, 2011.

[LCZ+08] Y.R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B.L. Tseng. Facetnet : a framework for analyzing communities and their evolutions in dynamic networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 685–694. ACM, 2008.

[LCZ+09] Y.R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B.L. Tseng. Analyzing communities and their evolutions in dynamic social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(2) :8, 2009.

[Lég12] M. Léguistin. L'amour 2.0, changements du masculin réels ou virtuels ? *Service social*, 58(1), 2012.

[LF09] A. Lancichinetti and S. Fortunato. Community detection algorithms : a comparative analysis. *Physical Review E*, 80(5) :056117, 2009.

[LFK09] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3) :033015, 2009.

[LFR08] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4) :046110, 2008.

[LGZ08] X. Li, L. Guo, and Y.E. Zhao. Tag-based social interest discovery. In *Proceedings of the 17th international conference on World Wide Web*, pages 675–684. ACM, 2008.

[LHB+12] J. Li, L. Huang, T. Bai, Z. Wang, and H. Chen. Cdbia : A dynamic community detection method based on incremental analysis. In *Systems and Informatics (ICSAI), 2012 International Conference on*, pages 2224–2228. IEEE, 2012.

[LKSF10] A. Lancichinetti, M. Kivelä, J. Saramäki, and S. Fortunato. Characterizing the community structure of complex networks. *PLoS One*, 5(8) :e11976, 2010.

[LLDM09] J. Leskovec, K.J. Lang, A. Dasgupta, and M.W. Mahoney. Community structure in large networks : Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1) :29–123, 2009.

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

[LM10] D. Laniado and P. Mika. Making sense of twitter. *The Semantic Web–ISWC 2010*, pages 470–485, 2010.

[LRMH10] C. Lee, F. Reid, A. McDaid, and N. Hurley. Detecting highly overlapping community structure by greedy clique expansion. *SNAKDD Workshop*, pages 33–42, 2010.

[LRRF11] A. Lancichinetti, F. Radicchi, J.J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4) :e18961, 2011.

[LVM+12] Y. Liu, B. Viswanath, M. Mondal, K.P. Gummadi, and A. Mislove. Simplifying friendlist management. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 385–388. ACM, 2012.

[MH10] A. McDaid and N. Hurley. Detecting highly overlapping communities with model-based overlapping seed expansion. In *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*, pages 112–119. IEEE, 2010.

[MLT09] M. Meyer, I. Lorscheid, and K.G. Troitzsch. The development of social simulation as reflected in the first ten years of jasss : a citation and co-citation analysis. *Journal of Artificial Societies and Social Simulation*, 12(4) :12, 2009.

[MRM+10] P.J. Mucha, T. Richardson, K. Macon, M.A. Porter, and J.P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980) :876–878, 2010.

[MTR11] B. Mitra, L. Tabourier, and C. Roth. Intrinsically dynamic network communities. *Computer Networks*, 2011.

[NC10] E. Navarro and R. Cazabet. Community detection : A comparative analysis on real-life networks. *Interaction, Intelligence, Information, an International Journal*, 2010.

[New04] M.E.J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6) :066133, 2004.

[NG04] M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2) :026113, 2004.

[NL07] M.E.J. Newman and E.A. Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences*, 104(23) :9564–9569, 2007.

[NMCM09] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri. Extending the definition of modularity to directed graphs with overlapping communities. *Journal of Statistical Mechanics : Theory and Experiment*, 2009(03) :P03024, 2009.

[NS01] K. Nowicki and T.A.B. Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96(455) :1077–1087, 2001.

[PBMW99] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking : bringing order to the web. 1999.

[PBV07] G. Palla, A.L. Barabasi, and T. Vicsek. Quantifying social group evolution. *Nature*, 446(7136) :664–667, 2007.

[PDFV05] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043) :814–818, 2005.

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

[PSPLPMM10] A. Padrol-Sureda, G. Perarnau-Llobet, J. Pfeifle, and V. Muntés-Mulero. Overlapping community search for social networks. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 992–995. IEEE, 2010.

[PV98] S. Ploux and B. Victorri. Construction d'espaces sémantiques à l'aide de dictionnaires de synonymes. *Traitement automatique des langues*, (39) :161–182, 1998.

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

[RB04] J. Reichardt and S. Bornholdt. Detecting fuzzy community structures in complex networks with a potts model. *Physical Review Letters*, 93(21) :218701, 2004.

[RB07] M. Rosvall and C.T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences*, 104(18) :7327–7331, 2007.

[RB08] M. Rosvall and C.T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4) :1118–1123, 2008.

[RB10] M. Rosvall and C.T. Bergstrom. Mapping change in large networks. *PloS one*, 5(1) :e8694, 2010.

[RCC+04] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9) :2658–2663, 2004.

[RH10] G. Roma and P. Herrera. Community structure in audio clip sharing. In *Intelligent Networking and Collaborative Systems (INCOS), 2010 2nd International Conference on*, pages 200–205. IEEE, 2010.

[SCCH09] H. Shen, X. Cheng, K. Cai, and M.B. Hu. Detect overlapping and hierarchical community structure in networks. *Physica A : Statistical Mechanics and its Applications*, 388(8) :1706–1712, 2009.

[Sch07] S.E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1) :27–64, 2007.

[Ser05] G.D.M. Serugendo. Self-organisation and emergence in multi-agent systems. *The Knowledge Engineering Review*, 20(2) :165–189, 2005.

[SG12] M. Seifi and J.L. Guillaume. Community cores in evolving networks. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 1173–1180. ACM, 2012.

[SJN06] S.W. Son, H. Jeong, and J.D. Noh. Random field ising model and community structure in complex networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 50(3) :431–437, 2006.

[SLX+12] J. Shang, L. Liu, F. Xie, Z. Chen, J. Miao, X. Fang, and C. Wu. A real-time detecting algorithm for tracking community structure of dynamic networks. *SNAKDD Workshop*, 2012.

[SM00] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8) :888–905, 2000.

[SNG+11] F. Sajous, E. Navarro, B. Gaume, L. Prévot, and Y. Chudy. Semi-automatic enrichment of crowdsourced synonymy networks : the wisigoth system applied to wiktionary. *Language Resources and Evaluation*, pages 1–34, 2011.

[SNTS06] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult. Monic : modeling and monitoring cluster transitions. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 706–711. ACM, 2006.

[SS02] J. Sabater and C. Sierra. Reputation and social network analysis in multiagent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems : part 1*, pages 475–482. ACM, 2002.

[SST+09] J. Sankaranarayanan, H. Samet, B.E. Teitler, M.D. Lieberman, and J. Sperling. Twitterstand : news in tweets. *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 42–51, 2009.

[TBWK07] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 717–726. ACM, 2007.

[UKBM11] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv :1111.4503*, 2011.

[WC89] Y.C. Wei and C.K. Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *Computer-Aided Design, 1989. ICCAD-89. Digest of Technical Papers., 1989 IEEE International Conference on*, pages 298–301. IEEE, 1989.

[WDdACG12] J. Whitbeck, M. Dias de Amorim, V. Conan, and J.L. Guillaume. Temporal reachability graphs. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 377–388. ACM, 2012.

[WF10] Q. Wang and E. Fleury. Mining time-dependent communities. In *LAWDN - Latin-American Workshop on Dynamic Networks*, 2010.

[WJW09] X. Wang, L. Jiao, and J. Wu. Adjusting from disjoint to overlapping community detection of complex networks. *Physica A : Statistical Mechanics and its Applications*, 388(24) :5045–5056, 2009.

[WLWT10] Z. Wu, Y. Lin, H. Wan, and S. Tian. A fast and reasonable method for community detection with adjustable extent of overlapping. In *Intelligent Systems and Knowledge Engineering (ISKE), 2010 International Conference on*, pages 376–379. IEEE, 2010.

[WT07] K. Wakita and T. Tsurumi. Finding community structure in mega-scale social networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 1275–1276. ACM, 2007.

[WWD08] Y. Wang, B. Wu, and N. Du. Community evolution of social network : feature, algorithm and model. *Science And Technology*, 2008.

[WYLL11] J. Weng, Y. Yao, E. Leonardi, and F. Lee. Event Detection in Twitter. *Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[XKH11] K. Xu, M. Kliger, and A. Hero. Tracking communities in dynamic social networks. *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 219–226, 2011.

[XKS11] J. Xie, S. Kelley, and B.K. Szymanski. Overlapping community detection in networks : the state of the art and comparative study. *arXiv preprint arXiv :1110.5813*, 2011.

[XKS13] Jierui Xie, Stephen Kelley, and B. K. Szymanski. Overlapping community detection in networks : the state of the art and comparative study. *ACM Computing Surveys*, 45(4), 2013.

[XSL11] J. Xie, B.K. Szymanski, and X. Liu. Slpa : Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 344–349. IEEE, 2011.

[YCZ+11] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin. Detecting communities and their evolutions in dynamic social networks ? a bayesian approach. *Machine learning*, 82(2) :157–189, 2011.

[YL12a] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 3. ACM, 2012.

[YL12b] J. Yang and J. Leskovec. Structure and overlaps of communities in networks. *arXiv preprint arXiv :1205.6228*, 2012.

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

[Zho03] H. Zhou. Distance, dissimilarity index, and network community structure. *Physical review e*, 67(6) :061901, 2003.