THÈSE DE DOCTORAT DE
L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

INFORMATIQUE

Présentée par

Thomas Aynaud

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

DÉTECTION DE COMMUNAUTÉS DANS LES RÉSEAUX DYNAMIQUES

Thèse dirigée par
Jean-Loup GUILLAUME et Matthieu LATAPY

Soutenance : 30 Novembre 2011

Jury :

Rapporteurs :	Éric FLEURY	-	Professeur, ENS de Lyon
	Nicolas HANUSSE	-	DR CNRS, Université Bordeaux I
Examinateurs :	Vincent BLONDEL	-	Professeur, Université Catholique de Louvain
	Franck PETIT	-	Professeur, UPMC
Encadrant :	Jean-Loup GUILLAUME	-	MdC, UPMC
Directeur :	Matthieu LATAPY	-	DR CNRS, UPMC

Remerciements

Je tiens à remercier toutes les personnes qui m'ont accompagné au cours de cette thèse. Mes premières pensées vont aux membres de l'équipe ComplexNetworks qui ont été des collègues formidables. On y a le sentiment de pouvoir bien faire son travail tout en profitant d'une ambiance très agréable et j'ai eu beaucoup de plaisir à y travailler.

Je remercie tout particulièrement Jean-Loup Guillaume et Matthieu Latapy qui m'ont encadré en stage puis en thèse. Ils ont toujours été disponibles et ont su me guider tout au long de ma thèse tout en me laissant une grande liberté. Quand tout allait bien, je pouvais avancer librement et chaque fois que quelque chose bloquait, je savais que je pouvais aller dans leurs bureaux et que j'en ressortirais avec une nouvelle idée ou une nouvelle façon de voir les choses. Leurs confiance, compétence, gentillesse et imagination ont beaucoup compté pour moi.

Bien sûr, ils ne sont pas les seuls à avoir joué un rôle et je remercie beaucoup Bénédicte Le Grand, Clémence Magnien et Fabien Tarissan pour leurs relectures et leurs remarques, ainsi que Lamia, Sébastien, Christophe, Massoud, Frédéric et Guillaume, avec qui j'ai eu la chance de partager un bureau, pour leur bonne humeur et leur aide. J'ai eu l'occasion de rencontrer et de partager des cafés, repas et idées avec de nombreuses personnes comme Alice, Oussama, Arnaud, Daniel, Maximilien, Raphaël, Assia, Amélie, Stéphane, Élie, Abdelhamid, Lionel et beaucoup d'autres. Merci à eux pour ces bons moments très enrichissants. Merci tout particulièrement à Véronique Varenne qui rend les démarches administratives faciles. J'ai aussi été très heureux d'enseigner avec Olivier Marchetti et Laurent-Stéphane Didier.

En plus de ces collègues, j'ai la chance d'avoir de très bons amis. Merci à eux ainsi qu'à ma famille proche et particulièrement à Marielle et à Agnès pour leur soutien sans failles et à Philippe pour les mégas de RAM et m'avoir donné envie de faire de la recherche. J'ai beaucoup de chance de partager ma vie avec Sophia. Sa joie de vivre et sa motivation ont beaucoup compté pour moi durant ces trois années. Mes colocataires ont été de très bons compagnons de thèse et je me souviendrai longtemps de Clément et ses pâtes, de Maëva et sa Bretagne, de Régis et Antoine avec leur caméra ainsi que de Ruy et sa chance. Merci à Olivier, Cyril, Sylvain, Thomas et les autres pour toutes ces discussions sur la science, l'univers et le reste ainsi qu'à Lucie pour tout ça et son courage d'avoir lu cette thèse.

Je tiens enfin à exprimer toute ma gratitude à Éric Fleury et Nicolas Hanusse pour avoir accepté d'être mes rapporteurs ainsi qu'à Vincent Blondel et Franck Petit qui ont bien voulu faire partie de mon jury.

Toutes mes excuses à ceux que j'ai pu oublier. S'ils m'ont lu jusque-là, ils méritent sans doute mes remerciements.

Table des matières

Chapitre 1

Introduction

Sommaire

1.1 Graphes de terrain

De nombreux systèmes complexes sont composés d'objets individuellement assez simples mais qui interagissent ou sont en relation entre eux. La complexité du système résulte alors de ces multiples relations. On trouve de tels systèmes dans de très nombreuses disciplines : en informatique, le web peut être vu comme un ensemble de pages web (de simples fichiers textes) liées entre elles par l'intermédiaire de liens hypertextes pour former une structure complexe, l'Internet peut être vu comme un ensemble de routeurs connectés par des câbles ; en biologie, le cerveau est un ensemble de neurones interagissant entre eux et une cellule peut être modélisée comme un groupe de protéines interagissant pour obtenir une fonction ; en sociologie, l'analyse des réseaux sociaux conduit à étudier comment interagissent divers agents, etc.

Ces systèmes, entités liées par des relations, font immédiatement penser aux graphes qui ont été largement étudiés en informatique et en mathématiques. Les objets constituant le système sont modélisés par les nœuds du graphe et quand deux objets interagissent ou sont liés dans le système réel, il y a un lien, aussi appelé arête, entre les nœuds correspondants dans le graphe. Nous appellerons les graphes modélisant des objets réels des *graphes de terrain* car ils proviennent d'une étude de terrain. Ils sont aussi souvent appelés réseaux mais il faut dans ce cas faire attention à ne pas les confondre avec des réseaux de communication qui en sont un sous-ensemble.

Il est apparu que ces graphes de terrain sont assez différents des graphes communément étudiés qui sont soit très réguliers (cliques, graphes planaires, ...) soit au contraire aléatoires. Bien que provenant de disciplines très différentes, il est possible de trouver des propriétés non triviales qu'ils partagent

fréquemment. Par exemple, la distance moyenne entre deux nœuds est souvent faible comparée à la taille totale du graphe [2]. Cette propriété est souvent décrite comme le fait que ces graphes sont des *petits mondes* en référence à l'expérience de Milgram [53]. On sait aussi que le nombre de voisins d'un nœud appelé son degré est distribué suivant une loi hétérogène (souvent bien approximée par une loi de puissance) : il existe beaucoup de nœuds avec peu de voisins, mais quand même des nœuds avec un degré intermédiaire et quelques nœuds avec des degrés très élevés agissant comme des hubs.

L'étude de ces graphes de terrain pose de nombreuses questions scientifiques. Premièrement, leur mesure n'est jamais une activité triviale. Par exemple, leur taille est souvent telle que pendant que l'on mesure une partie, le reste évolue en même temps induisant de nombreux biais. La visualisation des réseaux est souvent délicate et rarement très efficace et il faut définir des outils d'analyse permettant de les décrire et de rendre compte aisément de leurs spécificités. La taille des réseaux et leurs particularités structurelles font aussi émerger de nouvelles questions d'algorithmique et leur modélisation qui permettrait de prévoir leur évolution reste encore limitée car les modèles sont souvent assez éloignés de la réalité.

1.2 Communautés

Une question importante parmi toutes celles soulevées par l'étude des graphes de terrain est l'analyse de leur structure : comment peut-on décrire la forme générale de ces systèmes très délicats à appréhender dans leur globalité ? Une particularité par exemple des réseaux est que leur densité est en général faible : deux nœuds pris au hasard ont très peu de chance d'être reliés. Par contre, si l'on choisit deux nœuds ayant un voisin en commun, cette probabilité augmente sensiblement. Deux de mes amis ont plus de chances d'être amis que deux personnes prises au hasard parmi six milliards d'individus. Là où globalement le réseau semble très peu connecté, localement il se comporte presque comme une clique. Cette structure particulière s'explique par la présence de groupes de nœuds appelés des *communautés*. Ils sont en général vus comme des groupes intérieurement denses, les gens se connaissent dans une communauté, et extérieurement peu reliés au reste du réseau, les gens connaissent nettement plus de gens dans leur communauté qu'à l'extérieur.

La détection automatique de tels groupes offre un éclairage intéressant sur la structure du réseau. Il devient possible de voir de grandes tendances et d'obtenir une vision macroscopique de systèmes complexes. Vu l'étendue des domaines où apparaissent des graphes de terrain (on peut encore citer l'analyse de texte, de vidéos, la finance, la littérature et l'histoire), les applications de la détection de communautés sont très variées. Ainsi une communauté de pages web regroupe des pages traitant du même sujet et une communauté de

protéines regroupe des protéines participant à une fonction commune dans la cellule.

Les principaux problèmes liés à la détection de communautés sont la définition de ces groupes, leur détection en pratique qui est un problème plus algorithmique et comment valider que les résultats sont effectivement pertinents. Plusieurs définitions existent, mais l'une, basée sur une fonction appelée la modularité, est aujourd'hui plus généralement utilisée que les autres. Il existe aussi de nombreux algorithmes pour les calculer cherchant un compromis entre efficacité et qualité des résultats. La détection de communautés dans des réseaux statiques est aujourd'hui un outil utilisable et a été appliquée, entre autres, à la visualisation de graphes [5], à l'analyse de codes sources [52], du web [19] ou pour étudier différents réseaux sociaux [8, 85].

Il reste cependant encore de nombreuses questions en suspens. La définition de communautés est en général limitée à une partition de l'ensemble des nœuds. Cela exclut par exemple qu'un nœud appartienne à plusieurs communautés à des degrés divers, ce qui pourtant semblerait naturel dans un réseau social. Il existe aussi des questions relatives aux échelles de communautés. On peut en effet parfaitement imaginer qu'une communauté va elle-même en contenir et ainsi de suite, créant une hiérarchie de communautés. Jusqu'à quel niveau décomposer ? Quel est le niveau pertinent ? Ce sont encore aujourd'hui des questions largement ouvertes, même si plusieurs travaux vont déjà dans ce sens [5, 7, 86]. Nous présenterons en annexe dans la section 7 des recherches effectuées sur la notion de multi-échelle où nous montrons que l'on peut effectivement identifier des sous-communautés mais qu'il faut arrêter la redécomposition assez tôt. Nous analyserons la structure de la hiérarchie et proposerons un critère d'arrêt.

1.3 Le cas dynamique

Une autre grande question restante est l'étude des réseaux et de leur structure communautaire *en tenant compte de leur dynamique*. Habituellement les gens travaillent sur un graphe correspondant à l'agrégation de toutes leurs données sur la durée de la mesure. Or, ces données sont très souvent dynamiques : le web évolue constamment de même que les affinités entre les gens dans un réseau social. Une grande quantité d'information est donc complètement ignorée si l'on ne tient pas compte de l'aspect temporel intrinsèque.

Cette thèse porte sur l'étude de réseaux *dynamiques* et la détection de communautés sur ceux-ci. La détection de communautés sur les graphes dynamiques amène les mêmes problèmes de définition, de validation, d'algorithmique et d'analyse avec une dimension temporelle supplémentaire. En effet, il n'existe pas de définition de communautés dynamiques faisant consensus. On peut les définir comme une succession de communautés statiques, mais aussi

plus directement, suivant la façon de modéliser les graphes dynamiques. Il faut aussi étudier comment trouver ces communautés efficacement. Là où un algorithme était satisfaisant pour étudier un graphe, étudier plusieurs centaines de pas de temps d'un réseau dynamique fait souvent atteindre les limites de ses performances. Avoir une définition et des méthodes de détection ne suffit pas. Il faut des jeux de données sur lesquels valider les résultats ainsi que des méthodes d'analyse.

Toutes ces questions sont loin d'être triviales. Une approche naïve consiste à se dire que si on a un algorithme statique efficace alors on peut l'appliquer sur chaque pas de temps. La figure 1.1 montre le pourcentage de nœuds qui changent de communauté entre deux pas de temps extrêmement resserrés (la dynamique est décrite précisément au chapitre 3). Même si la dynamique est très faible entre deux pas de temps consécutifs, on voit que pratiquement à chaque fois près de 20% des nœuds changent de communautés ce qui semble totalement irréaliste. Ceci est donc lié à l'instabilité des algorithmes et non à des changements structuraux du graphe. Les algorithmes statiques ne sont donc pas directement transposables aux réseaux dynamiques et il faudra en proposer de nouveau. Cela ne signifie pas que l'approche naïve soit inutilisable, simplement qu'il faudra prendre ces problèmes en compte pour l'adapter.

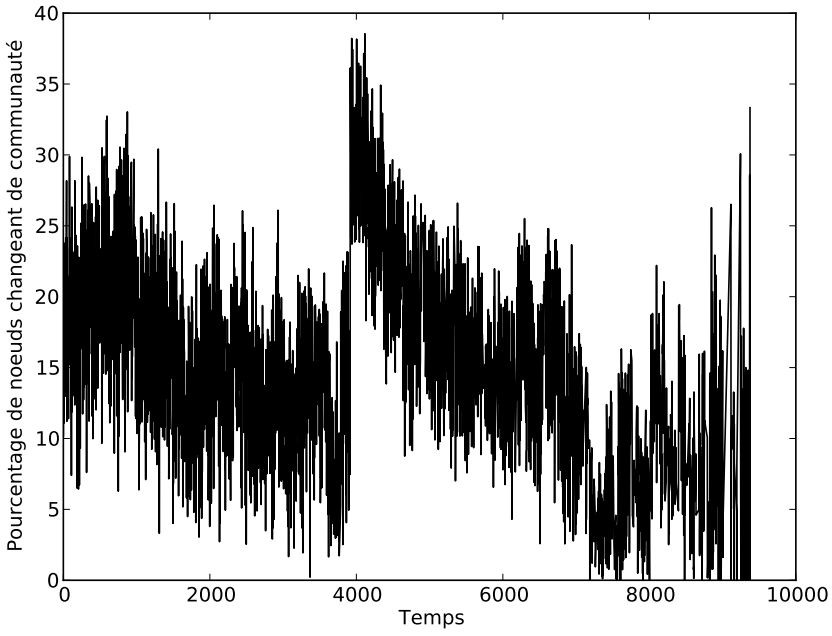


FIGURE 1.1 – Pourcentage des nœuds changeant de communauté entre deux pas de temps consécutifs dans le cas d'une dynamique très faible.

1.4 Contributions

Cette thèse contribue à l'étude des communautés dans les graphes dynamiques de plusieurs façons :

- **Instabilités** : Premièrement, nous avons mené une étude minutieuse de la stabilité de divers algorithmes statiques de détection de communautés. Cela nous a permis de montrer que l'approche consistant à les utiliser sur chaque pas de temps, bien que commune, est très dangereuse sans plus de précautions. En effet, les algorithmes se sont avérés très instables : une modification mineure du réseau conduit à une grande transformation des communautés. Les changements observés sont donc plus liés à l'algorithme employé qu'à un vrai changement de structure. Nous avons proposé une modification de l'un d'eux pour atteindre une stabilité plus satisfaisante et avons étudié les causes de son instabilité.
- **Analyse sur une fenêtre de temps** : Nous avons ensuite proposé une nouvelle approche pour la détection de communautés dans le cas dynamique. Là où habituellement on cherchait à détecter une communauté à chaque pas de temps, nous nous sommes attachés à détecter une seule partition de qualité durant toute une durée, que nous appelons la fenêtre de temps. Cette partition est donc une meilleure représentante de la structure globale du réseau, avec sa dynamique.
- **Un temps divisé hiérarchiquement** : Une autre question majeure est la définition de la durée de cette fenêtre de temps. Nous avons montré qu'il était possible de trouver une hiérarchie de fenêtres de temps intéressantes automatiquement. De plus, notre définition de fenêtre de temps permet de voir des phénomènes comme une structure répétée. Si l'on considère par exemple le réseau de qui parle avec qui, on imagine que la structure change en fonction du moment de la journée : on voit différentes personnes le jour (des collègues souvent) et le soir (famille et amis). Les changements de structure du réseau le soir et le jour peuvent être détectés par notre méthode.
- **Des applications sur des cas concrets** : Enfin, nous avons essayé d'appliquer ces différents outils à des cas concrets. L'étude des instabilités est appliquée à l'analyse de graphes représentant la topologie d'Internet afin d'y faire de la détection d'événements. Nous avons aussi appliqué la détection de communautés au suivi d'objets dans des vidéos mais pour cela nous avons en revanche dû définir des algorithmes spécifiques.

Cette thèse est organisée en quatre parties. La première regroupe la présentation de la majorité des outils que nous utiliserons, que ce soit définitions, algorithmes ou jeux de données ainsi qu'un état de l'art le plus exhaustif possible des travaux sur les communautés dans les graphes dynamiques existants lors de la rédaction de ce manuscrit. La deuxième partie contient notre analyse

de l'instabilité des algorithmes statiques et les solutions proposées. La troisième partie porte sur notre méthode de détection de communautés bonnes pendant plusieurs pas de temps et le calcul des fenêtres de temps intéressantes. La quatrième partie présente les différents cas concrets rencontrés, les problèmes posés, les solutions proposées et les résultats obtenus. Enfin, après une conclusion sur l'étude des communautés dans les graphes dynamiques, nous présenterons en annexe nos recherches sur la notion de multi-échelle.

Chapitre 2

Outils et état de l'art

Sommaire

2.1 Outils et définitions

2.1.1 Modularité

Une définition naturelle des communautés stipule qu'une communauté est dense, c'est-à-dire que ses membres sont fortement connectés entre eux et que, dans le même temps, ils sont peu liés à des membres en dehors de la communauté. Le problème de la détection de communautés est donc naturellement formalisé en la recherche d'une partition d'un graphe en sous-groupes denses peu connectés entre eux.

Définir les communautés comme les parties d'une partition est fréquemment admis, mais cela implique qu'un nœud n'appartient qu'à une et une seule communauté. Il existe des définitions de communautés où un nœud peut appartenir à diverses communautés [26, 67, 86]. On appelle de telles communautés des communautés *recouvrantes*. Néanmoins, aucune définition ne fait aujourd'hui consensus, il existe peu d'algorithmes pour les détecter, les outils d'analyses sont encore limités et la détection de communautés recouvrantes

est donc aujourd'hui encore considérée comme un problème ouvert. Nous nous restreindrons donc à l'étude de partitions.

Pour évaluer la qualité d'une partition, on utilise généralement une fonction dite de qualité donnant un score à une partition et qui capture de manière formelle l'intuition donnée précédemment. Ensuite, il "reste" à maximiser cette fonction pour trouver la meilleure partition pour un graphe donné. Il existe plusieurs fonctions de qualité [15,23], la plus utilisée étant la *modularité* définie dans [61].

Définition 1 *Pour une partition π de l'ensemble des nœuds d'un graphe $G = (V, E)$, en notant $L = |E|$ le nombre de liens du graphe, d_s le degré total d'une partie s de π et l_s le nombre de liens à l'intérieur de s, la modularité Q est définie par :*

$$Q(\pi) = \sum_{s \in \pi} \left(\frac{l_s}{L} - \left(\frac{d_s}{2L} \right)^2 \right)$$

Cette grandeur est la somme, sur toutes les communautés, des différences entre la proportion de liens à l'intérieur de la communauté s (soit $\frac{l_s}{L}$) et la proportion de liens que devrait avoir une communauté dans un graphe aléatoire de même distribution de degrés (soit $\left(\frac{d_s}{2L}\right)^2$ [1]). Une partition sera donc bonne s'il y a nettement plus de liens à l'intérieur des communautés que ce à quoi on s'attendait (et par conséquent moins de liens hors des communautés). La modularité est comprise entre -1 et 1 mais la partition triviale regroupant tous les nœuds en une seule partie a pour modularité 0 et donc les partitions intéressantes ont toutes une modularité positive.

La modularité permet de comparer deux partitions d'un même graphe mais pas vraiment des partitions de graphes différents. Elle n'est pas une mesure absolue de qualité, dans le sens où la meilleure partition pour un graphe n'aura pas la même modularité que la meilleure partition pour un autre graphe. En particulier, une modularité de 1 est inatteignable. Des bornes plus précises sont données dans [9]. Par exemple, la modularité est forcément inférieure à $1 - \frac{2}{L}$. Une analyse de la modularité de graphes aléatoires a été faite dans [73] où il est entre autres montré que pour un graphe d'Erdös-Renyi de N nœuds liés avec probabilité p, on peut espérer une modularité de $0.97\sqrt{\frac{1-p}{pN}}$.

En revanche, la meilleure modularité atteignable pour un graphe donne une information sur la structure de celui-ci. Si elle est élevée, cela signifie que le graphe est divisible en parties très denses et nettement séparables, contrairement à un graphe dont la meilleure partition a une modularité faible.

1. Il y a d_s demi-liens partant de la communauté qui ont chacun une probabilité $\frac{d_s}{2L}$ d'être reliés à un demi-lien dans la communauté. Cela fait donc $\frac{d_s^2}{2L}$ paires de demi-liens reliés dans la communauté soit $\frac{d_s^2}{4L}$ liens et donc une proportion de $\frac{1}{L}\frac{d_s^2}{4L} = \left(\frac{d_s}{2L}\right)^2$

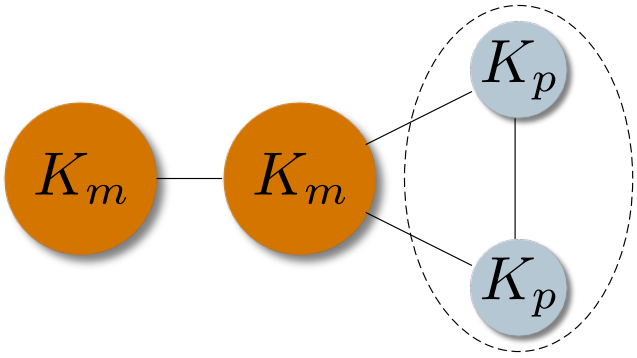


FIGURE 2.1 – Exemple d'effet de la résolution limite : il y a quatre cliques contenant m ou p nœuds. Normalement, il faudrait donc diviser le réseau en quatre. Or, quand m devient grand devant p (par exemple pour $m = 20, p = 5$), la modularité obtenue en regroupant les deux petites cliques est supérieure à celle obtenue en les séparant.

Il a aussi été montré que l'optimisation de la modularité a un effet non souhaité qui est de défavoriser les petites communautés, de tailles inférieures à $\sqrt{2L}$, lesquelles ont souvent (mais pas toujours) intérêt à fusionner pour améliorer la modularité [9]. Ce problème a reçu le nom de "résolution limite" et nous l'illustrons sur la figure 2.1. Les communautés qui maximisent la modularité sont donc parfois des regroupements de communautés plus petites et ce sont ces petites communautés qui sont pertinentes. Il arrive aussi que les algorithmes divisent des communautés arbitrairement car la modularité est un compromis entre beaucoup de communautés avec une faible qualité [2] ou peu de communautés avec une forte qualité. Il existe ainsi un nombre de communautés vers lequel tend l'algorithme afin de maximiser la qualité et non pour détecter des communautés pertinentes.

Des méthodes pour contrebalancer ce phénomène existent. Pour obtenir des plus petites communautés, il est possible de diviser à nouveau les communautés obtenues en sous-communautés jusqu'à un certain point comme nous le ferons en annexe au chapitre 7 ou de modifier la définition de la modularité. Cette dernière technique suppose en général de définir une nouvelle fonction de qualité ayant un paramètre supplémentaire de résolution que l'on fait varier pour obtenir des communautés plus ou moins petites. On peut citer par exemple [3, 72] dont les auteurs définissent des paramètres pour détecter de plus petites communautés que celles obtenues par la modularité. Ces résultats ont été généralisés dans [47] où les auteurs donnent une nouvelle définition de communauté. Ils définissent la stabilité d'une partition comme une statistique

2. La qualité d'une communauté s est le terme dans la modularité qui la concerne, à savoir $\frac{l_s}{L} - \left(\frac{d_s}{2L}\right)^2$.

comptant d'une part la probabilité d'une marche aléatoire d'une longueur donnée de rester dans un groupe minorée par la probabilité dans un graphe aléatoire de même distribution de degrés. Une bonne partition est alors une partition ayant une haute stabilité. Pour avoir une idée intuitive de la méthode, on peut envisager un réseau social. Sur celui-ci, le parcours aléatoire est la diffusion d'une information et une information a tendance à rester dans une communauté. Plus le chemin que peut parcourir cette information est long, plus la communauté dans laquelle elle doit rester est grande. La durée de la marche aléatoire est un nouveau paramètre, noté t dans l'article mais que nous noterons R pour ne pas le confondre avec l'instant considéré, et les auteurs montrent que l'optimisation de la stabilité se rapporte à la modularité et que les autres méthodes de gestion de la résolution en sont des cas particuliers.

En particulier, il faut alors optimiser :

$$Q_{NL}(R) = (1 - R) + \sum_{s \in \pi} \left(R \frac{l_s}{L} - \left(\frac{d_s}{2L} \right)^2 \right)$$

Quand R est égal à 1, leur généralisation est exactement la modularité tandis que quand $R \leq 1$ les communautés sont plus petites et quand $R \geq 1$ les communautés sont plus grandes. Nous utiliserons ce paramètre aux chapitres 3 et 5. Néanmoins, ces paramètres n'agissent que globalement et il est toujours problématique de détecter à la fois les petites communautés et les grandes [48].

2.1.2 Méthode de Louvain

Trouver la (ou les) partition qui maximise la modularité est un problème NP-difficile [14] et de nombreuses heuristiques ont donc été proposées, voir par exemple [20, 33, 60, 61, 70]. La plupart de ces algorithmes produisent des hiérarchies de communautés imbriquées, mais nous considérerons en général que leur résultat est la partition dans cette hiérarchie ayant la plus grande modularité.

Dans la suite de cette thèse, nous avons utilisé principalement la méthode de Louvain [11] qui produit des résultats ayant parmi les modularités les plus élevées tout en étant très rapide, ce qui est une nécessité dans l'étude de graphes dynamiques qui conduit souvent à calculer énormément de partitions.

Cette approche est composée d'un ensemble de passes, chacune composée de deux phases, qui sont répétées de manière itérative jusqu'à obtenir un maximum local de la modularité. L'algorithme part d'un graphe pondéré non orienté ayant N sommets. L'algorithme 1 présente une version en pseudo-code.

Première phase. La partition initiale consiste à placer chaque sommet dans une communauté distincte, cette partition est donc composée de N communautés. Nous considérons ensuite le premier sommet 0 et calculons la va-

riation de modularité obtenue en supprimant 0 de sa communauté et en le plaçant dans la communauté de l'un de ses voisins j. Cette variation est calculée pour chacun des voisins de 0 et le sommet 0 est ensuite déplacé dans la communauté pour laquelle cette variation est maximum, mais uniquement si elle est positive. Si tous les gains sont négatifs, alors le sommet 0 est replacé dans sa communauté d'origine. Ce processus est appliqué de manière séquentielle sur tous les sommets et on appelle cela une itération. Le processus est ensuite ré-appliqué à tous les sommets de manière répétée jusqu'à ce que plus aucun sommet ne soit déplacé pendant une itération complète. Après cette première phase, le réseau a donc été découpé en une partition P ayant N_c communautés. Si la première phase a regroupé des sommets, l'algorithme passe à la deuxième phase, sinon l'algorithme est terminé et le résultat est la partition P.

Deuxième phase. La deuxième phase consiste à construire un nouveau graphe dont les sommets sont les communautés découvertes durant la première phase. Pour cela, le poids des liens entre ces nouveaux sommets est donné par la somme des poids des liens qui existaient entre les sommets de ces deux communautés. Les liens qui existaient entre des sommets d'une même communauté créent des boucles sur cette communauté dans le nouveau graphe. Une fois cette seconde phase terminée, il est possible d'appliquer à nouveau la première phase de l'algorithme sur le graphe pondéré et d'itérer. Cela constitue une passe et à chaque passe la modularité croît par agrégation des nœuds dans des communautés de plus en plus grandes.

La dernière partition trouvée est celle qui a la meilleure modularité. Nous considérerons qu'elle est le résultat de la méthode de Louvain et oublierons la hiérarchie produite.

2.1.3 Autres algorithmes statiques

Afin de ne pas restreindre notre étude à la méthode de Louvain, nous avons utilisé d'autres algorithmes classiques de détection de communautés dans les graphes statiques. Cela permet de valider que les propriétés observées ne sont pas des propriétés induites par la méthode de Louvain, mais bien des propriétés des communautés telles que définies par la modularité.

Nous avons donc utilisé :

- Walktrap [70], un algorithme basé sur le fait qu'une marche aléatoire courte va rester dans la communauté du nœud initial avec forte probabilité, car il n'y a que peu de ponts vers les autres communautés. Cette propriété est utilisée pour définir une distance entre les nœuds et entre communautés utilisée pour construire un regroupement hiérarchique.
- Glouton Rapide [20], un algorithme glouton d'optimisation de la modularité : initialement, chaque nœud est seul dans sa communauté puis, à chaque étape, l'algorithme regroupe les deux communautés maximisant

Algorithme 1 Pseudo-code de l'algorithme de décomposition en communautés

```
1: G le graphe initial
2: répéter
3:    Placer chaque nœud de G dans une unique communauté
4:    Sauver la modularité de cette décomposition
5:    tantque il y a des sommets déplacés faire
6:       pour tout nœud n de G faire
7:          Chercher c la communauté voisine de n maximisant le gain de modularité si n est déplacé dans c
8:          si c induit un gain strictement positif alors
9:             déplacer n de sa communauté dans c
10:         finsi
11:      fin pour
12:   fin tantque
13:   si la modularité atteinte est supérieure à la modularité initiale alors
14:      fin ← faux
15:      Afficher la décomposition trouvée
16:      Transformer G en le graphe entre les communautés
17:   sinon
18:      fin ← vrai
19:   finsi
20: jusqu'à fin
```

le gain de modularité.

Ces deux algorithmes sont ou ont été largement utilisés et sont relativement rapides ce qui est une nécessité pour analyser des réseaux dynamiques. En effet, on va avoir besoin de les lancer un grand nombre de fois et ce serait impossible de le faire avec des algorithmes plus lents. Ils possèdent en outre des implémentations efficaces en C ce qui nous a permis de les utiliser sans trop de difficultés.

2.2 Communautés dynamiques

Toutes les méthodes et définitions que nous avons présentées jusque-là s'appliquent à des réseaux statiques. Or, la majorité des données sont dynamiques : les réseaux évoluent dans le temps. Cela pose de nouvelles questions scientifiques. On peut par exemple vouloir définir les communautés comme une succession de communautés statiques ou comme un objet directement dynamique. Nous allons maintenant passer en revue les différentes solutions proposées pour la détection de communautés dans des graphes dynamiques. Elles sont divisées en deux grandes familles. Premièrement, il y a les méthodes qui fonctionnent en deux temps : on détecte des communautés à chaque instant à l'aide d'un algorithme statique puis on fait le lien entre elles. Ensuite, il existe des définitions et des algorithmes utilisant plus directement la dynamique. Puis, nous verrons une classe particulière d'algorithmes de détection de communautés qui cherchent à faire évoluer une partition déjà trouvée en fonction des changements du réseau. Enfin, il est inutile de définir des communautés dynamiques sans ensuite chercher à décrire leur évolution et leurs propriétés. Nous présenterons donc les méthodes d'analyse de communautés dynamiques.

2.2.1 Notations

En règle générale, un graphe dynamique sera une succession de graphes statiques. Nous appellerons ces graphes statiques des *instantanés* car ils représentent l'image du réseau dynamique à un instant donné. Le temps sera divisé en *pas de temps* ou *instants*. Un pas de temps sera l'index de l'instantané. Si les instantanés ne sont pas répartis régulièrement on peut associer au pas de temps une durée. De plus, sauf mention contraire, nous noterons l'instant suivant t comme l'instant $t + 1$ quelle que soit la durée qui les sépare.

2.2.2 Utilisation d'algorithmes statiques

Comme le problème de la détection de communautés est largement étudié sur des graphes statiques, il est naturel de commencer par utiliser les solutions

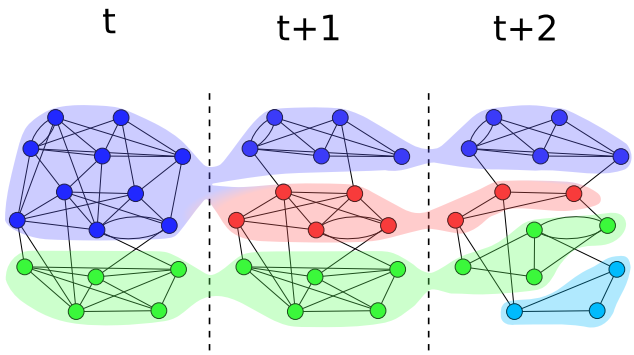


FIGURE 2.2 – Trois instantanés d'un réseau dynamique avec une association entre les communautés des différentes étapes.

statiques dans le cas dynamique. C'est pourquoi plusieurs tentatives d'utilisation des algorithmes statiques existent. L'idée générale est de diviser le réseau dynamique en une série d'instantanés qui sont tous des graphes statiques puis la détection se fait en deux temps : tout d'abord, il faut appliquer un algorithme classique sur chacun de ces instantanés ce qui permet d'obtenir une série de partitions, une pour chaque instantané. Dans un deuxième temps, il faut décider ce qui se passe entre deux pas de temps, par exemple en faisant la correspondance entre les communautés trouvées à l'instant t et les communautés de l'instant $t+1$. Des événements plus subtils peuvent apparaître comme des regroupements de communautés, des scissions ou encore des dissolutions de communautés par exemple. Nous appellerons le problème de la décision de ce qui se passe entre deux instants le problème du suivi. Il consiste à trouver une *association* entre deux partitions. La figure 2.2 montre un exemple d'association possible.

Une préoccupation fréquente sera de gérer l'instabilité des algorithmes. En effet, les algorithmes statiques de détection de communautés peuvent donner des résultats assez éloignés pour des graphes très proches. Ceci est lié à leur caractère heuristique, à leur non déterminisme ou au fait qu'il y a souvent plusieurs bonnes décompositions. Si les résultats varient trop, les événements suivis sont plus liés à l'algorithme qu'à un changement structurel et n'apportent donc pratiquement pas d'informations sur l'évolution du réseau.

2.2.2.1 Suivi et méthodes ensemblistes

La première méthode intuitive pour effectuer le suivi est de considérer chaque communauté comme un ensemble et d'utiliser des méthodes ensemblistes couplées à des règles pour décider si deux communautés de différentes partitions sont similaires ou non. Par exemple, si deux communautés de pas de temps successifs partagent une grande partie de leurs nœuds, elles sont

sans doute étroitement liées.

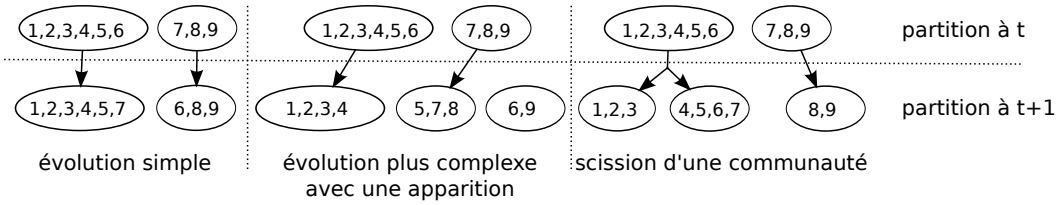


FIGURE 2.3 – Quelques cas particuliers d'événements dans la vie d'une communauté.

La première méthode de suivi est proposée dans [39]. Ses auteurs étudient un réseau de co-auteurs et utilisent la taille de l'intersection entre deux communautés successives : une communauté à l'instant $t+1$ succède à une communauté à l'instant t si elles partagent suffisamment de nœuds, soit si leur intersection est grande. Plus précisément, le *match* entre deux communautés est défini comme :

$$match(C, C') = min\left(\frac{|C \cap C'|}{|C|}, \frac{|C \cap C'|}{|C'|}\right)$$

Cette valeur est comprise entre 0 et 1 et plus les communautés sont proches, plus l'intersection est grande, plus grand le *match* est. Il vaut 1 si elles sont égales et 0 si elles sont disjointes. Les auteurs définissent la communauté correspondant à la communauté C de l'instant t comme la communauté C' à l'instant $t+1$ qui maximise cette valeur. À cause de l'instabilité de l'algorithme de détection de communautés utilisé, un grand nombre de communautés disparaissent ou sont grandement modifiées entre deux pas de temps. Par conséquent, les communautés sont initialement impossibles à suivre, car les changements sont causés par l'algorithme et non dus à une modification de la structure du réseau. Ce problème est résolu en ne considérant que les communautés stables, définies comme les communautés qui continuent à exister même après une modification mineure. De telles communautés sont appelées des *communautés naturelles* et, bien que cette restriction élimine de nombreuses communautés intéressantes, elle a permis une première analyse de communautés dynamiques. La validation est effectuée en observant, sur les communautés trouvées dans la base de données de *CiteSeer*, des faits déjà connus dans l'évolution des sciences comme l'apparition du sujet des réseaux ad-hoc dans la communauté réseau.

Cette méthodologie de suivi a été généralisée dans [77] où les auteurs définissent plusieurs règles pour gérer les autres cas comme les divisions, regroupements, apparitions et disparitions de communautés (comme ceux montrés

sur la figure 2.3). Étant donnée une communauté C à l'instant t, les auteurs définissent $match(C)$ comme la communauté à $t+1$ dont l'intersection est la plus grande, si cette intersection est plus grande qu'une limite donnée. Si une telle communauté n'existe pas, car les intersections sont toutes trop petites, alors $match(C) = \varnothing$. Ils utilisent ensuite cet ensemble de règles pour définir les événements régissant la vie des communautés :

- $C \in P_t$ devient $C' \in P_{t+1}$ si $C' = match(C)$ et $\forall C'' \in P_t \neq C, C' \neq match(C'')$.
- $C \in P_t$ se divise en plusieurs communautés $C_1, C_2, ..., C_k \in P_{t+1}$ if $\forall i$, $C_i \cap C$ est suffisamment grand et $(C_1 \cup C_2 \cup ... \cup C_k) \cap C$ est suffisamment grand.
- $C \in P_t$ s'est regroupée avec d'autres communautés pour former $C' \in P_{t+1}$ si $C' = match(C)$ et $\exists Z \in P_t \neq C, C' = match(Z)$.
- $C \in P_t$ a disparu si aucune des règles précédentes ne s'applique.
- $C' \in P_{t+1}$ a apparu si $\forall C \in P_t, C' \neq match(C)$.

Ces règles sont assez naturelles et faciles à comprendre mais ne sont pas vraiment satisfaisantes. En effet, le terme "suffisamment grand" n'est pas une définition formelle. Les auteurs de [77] en proposent une mais dans [4, 29, 36, 63, 64] des règles très similaires sont proposées en utilisant différentes notions de proximité. De plus, toutes ces règles demandent des paramètres souvent impossibles à fixer proprement. Si une intersection est "suffisamment grande" quand sa taille représente n% de chaque communauté, comment fixer n ? Est-ce que deux communautés sont similaires quand elles partagent 50% des nœuds, 70% ou 90% ? Comme les algorithmes statiques sont souvent instables, ce paramètre est souvent fixé à des valeurs que nous trouvons basses pour contrebalancer l'instabilité et avoir de vraies choses à suivre et ne pas avoir des effets étranges comme l'apparition de toutes les communautés de chaque instant. Ainsi, trouver un ensemble minimal et consensuel de règles semble impossible en pratique et les paramètres qu'ils impliquent ne peuvent pas être fixés sans connaissances a priori sur l'évolution des communautés.

Dans [88], les auteurs utilisent le même type de solution basée sur des comparaisons d'intersections mais, au lieu de calculer le *match* entre des communautés, ils proposent de les suivre à l'aide de quelques nœuds importants appelés nœuds cœurs. Ces nœuds cœurs, centraux pour la communauté, sont censés être de meilleurs représentants de celle-ci que les nœuds en bordure dont l'appartenance à la communauté est peut être plus floue. Quand des nœuds cœurs précédemment dans la même communauté se séparent, c'est qu'un changement important a lieu car ils sont définis de manière à être stables et importants pour la cohésion de la communauté. Les auteurs sélectionnent donc pour chaque communauté un certain nombre de représentants et suivent ces représentants au cours du temps. Les limitations liées au suivi de communautés comme la difficulté du consensus sur les règles et les paramètres

sont cependant toujours là et à celles-ci s'ajoutent le problème de la définition des nœuds cœurs. Dans [88], les nœuds cœurs sont les nœuds v qui vérifient $\sum_{n \in neighbours} degree(v) - degree(n) > 0$ tandis que dans [10] ils sont des nœuds dont le degré est supérieur à une valeur k donnée. Il y a bien d'autres mesures de centralité qui visent toutes à classer les nœuds en fonction de leur importance et en choisir une est donc une vraie difficulté et peut dépendre des applications visées.

Les auteurs de [18] utilisent des nœuds cœurs définis comme les nœuds existant aux instants $t-1$, t et $t+1$ pour réduire le nombre de communautés à considérer. Ils définissent les communautés initialement comme les cliques maximales et peuvent donc avoir des communautés recouvrantes. Néanmoins, le nombre de communautés peut être élevé (exponentiel en le nombre de nœuds) et ils utilisent la notion de nœuds cœurs pour ne considérer que les communautés contenant des nœuds cœurs et donc restreindre leur nombre. Les nœuds cœurs permettent aussi dans un deuxième temps de faire du suivi en appliquant des règles classiques.

Finalement, les auteurs de [87] définissent les nœuds cœurs comme les nœuds classés ensemble avec forte probabilité quand on applique plusieurs fois un même algorithme non déterministe (en l'occurrence la méthode de Louvain, mais d'autres algorithmes pourraient être utilisés). Une fois les nœuds cœurs obtenus à l'instant t, ils servent à initialiser l'algorithme de détection de communauté quand on l'applique à l'instant $t+1$. Cela sert à stabiliser l'algorithme qui comme nous le verrons au chapitre 3 est très instable et par conséquent inadapté en l'état au suivi de communautés.

2.2.2.2 Utilisation de l'algorithme de détection de communautés pour le suivi

Une autre approche pour effectuer le suivi, proposée dans [66], consiste à utiliser l'algorithme de détection de communautés pour effectuer le suivi. Pour cela, les auteurs construisent l'union des graphes à t et $t+1$ et détectent des communautés sur ce nouveau graphe. Ils utilisent une définition particulière des communautés qui leur garantit que les communautés trouvées à t ou $t+1$ seront intégralement contenues dans les communautés de l'union. Quand des communautés à t sont regroupées sur l'union avec des parties à $t+1$, elles sont en relation et on peut en déduire leur évolution (voir figure 2.4). Les auteurs valident enfin l'approche en vérifiant l'intérêt de leurs communautés sur un réseau de téléphonie mobile et un réseau de co-auteurs.

Cependant, cette technique suppose une propriété rarement vérifiée par les définitions classiques de communautés : si des liens sont ajoutés au réseau, les communautés ne peuvent que grossir, se regrouper ou rester inchangées. Cette propriété assure que les communautés détectées sur l'union des instantanés

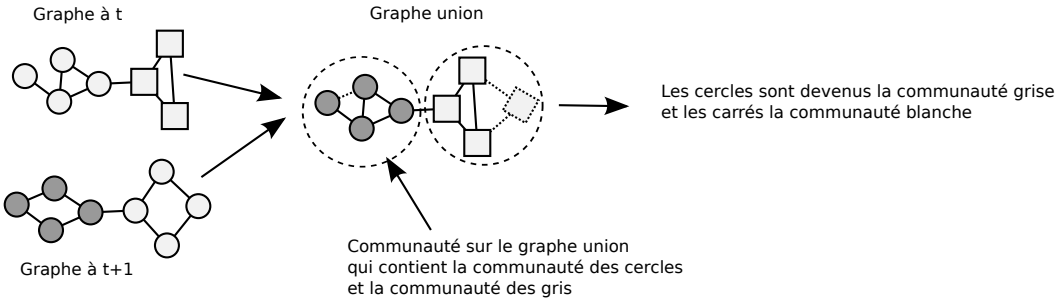


FIGURE 2.4 – Utilisation de l'union et de l'algorithme de détection pour le suivi

ne contiendront que des communautés entières parmi celles de t ou $t+1$. Par conséquent, quand on détecte des communautés sur le graphe union, on peut clairement décider de mettre en relation des communautés des instants t et $t+1$. Avec une définition plus relâchée de communautés, il est possible que les communautés sur le graphe union ne contiennent que des portions de communautés à t ou $t+1$ ce qui rendrait l'association délicate. Néanmoins, c'est une définition très restrictive de communautés qui ne va pas permettre de détecter toutes les communautés intéressantes sur un réseau complexe. Elle implique aussi quelques paramètres pour gérer les cas des regroupements et des scissions et n'est donc pas pleinement satisfaisante.

2.2.2.3 Réseau d'évolution entre communautés

Une autre façon de suivre les communautés est de construire un réseau temporel représentant les relations entre les communautés à chaque pas de temps. On peut alors décomposer ce réseau entre communautés pour obtenir des groupes de communautés appartenant à plusieurs pas de temps. L'algorithme est donc en deux phases : premièrement, tous les instantanés du réseau analysé sont décomposés en communautés statiques, puis deuxièmement ces communautés deviennent les nœuds d'un réseau représentant les rapports entre les communautés à différents pas de temps. Ce réseau est ensuite analysé, souvent à l'aide encore d'un algorithme de détection de communautés. Nous appellerons les communautés obtenues initialement sur chaque instantané des *instances* et les communautés obtenues sur le réseau entre *instances* des *chronologies* (voir la figure 2.5 pour un exemple).

Dans [28], les *instances* sont calculées à l'aide d'un algorithme classique de clustering hiérarchique [61] à chaque pas de temps. Ensuite, le réseau entre *instances* est construit en liant les *instances* de tous les pas de temps à l'aide d'un poids similaire au *match* de la méthode de suivi présentée précédemment [77]. Pour éviter des liens entre des *instances* de périodes trop distantes, les liens

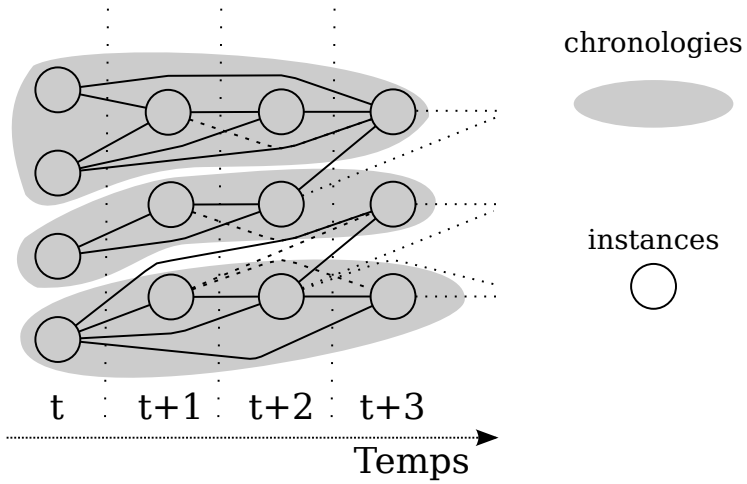


FIGURE 2.5 – Un exemple de graphe entre *instances* (les nœuds noirs) et des *chronologies* (en gris).

entre *instances* appartenant à des pas de temps distant dans le temps de plus d'une limite donnée sont oubliés. Ce réseau est ensuite divisé en communautés afin d'obtenir les *chronologies*. Celles-ci contiennent des *instances* de plusieurs pas de temps, et définissent les communautés finales regroupant des nœuds de plusieurs pas de temps. Mais avec cette définition, un nœud peut appartenir à une *chronologie* différente à chaque pas de temps, car les *instances* auxquelles il appartient n'ont pas été regroupées dans la même *chronologie*. Afin de décrire ce phénomène, les auteurs définissent une nouvelle métrique, la *participation* qui décrit combien un nœud appartient à une *chronologie*.

Dans [19], les auteurs utilisent le même type de décomposition en deux étapes. Ils calculent premièrement des *instances*, représentées par un *tenseur*. Les *instances* sont ensuite pondérées, afin de représenter leur importance dans les *chronologies*. Au final, ils obtiennent une fonction à optimiser pour obtenir des *chronologies* recouvrantes existant durant plusieurs pas de temps. Ils proposent une méthode d'optimisation de cette fonction et valident celle-ci en retrouvant des faits connus sur le réseau de blogs de NEC.

Dans [80], les auteurs utilisent une approche similaire mais redéfinissent la notion d'*instance* en se basant sur des hypothèses tirées de l'analyse des réseaux sociaux : une *instance* est un groupe de personnes qui se voient souvent, qui voient rarement des personnes d'autres groupes et qui ne changent pas trop au cours du temps. Les auteurs proposent donc une fonction de qualité pour assigner les nœuds à des *instances* et les *instances* aux *chronologies*. De plus, les nœuds peuvent changer de *chronologie* au cours du temps et la fonction de qualité est au final la somme des coûts :

– α, le coût individuel quand un nœud change de *chronologie* entre deux

pas de temps. Cela impose aux nœuds de ne pas changer trop souvent de communauté.
- β, le coût de groupe quand un nœud et son *instance* ne sont pas dans la même *chronologie* à un moment donné.

Finalement, ils ajoutent un coût "couleur" qui est proportionnel, pour chaque nœud, au nombre de *chronologies* auxquelles il appartient pendant toute la durée pour s'assurer qu'un nœud particulier n'appartient pas à trop de *chronologies*. Optimiser cette fonction est un problème NP-complet mais des algorithmes avec des facteurs d'approximation garantis sont proposés dans [79].

2.2.2.4 Conclusion

Aucune des techniques mentionnées n'est complètement satisfaisante. Le premier problème vient du fait qu'un consensus sur les règles et les paramètres est sans doute impossible. Le second problème, souvent négligé, est l'instabilité : un algorithme de classification peut fournir des résultats très différents sur des réseaux très proches, voire isomorphes en modifiant l'initialisation. Cela implique que les transformations détectées sont souvent des changements de la sortie de l'algorithme et non de vrais changements structuraux. Nous discuterons ces problèmes en détail dans le chapitre 3.

2.2.3 Détection directe de communautés dynamiques

Puisque les méthodes basées sur des algorithmes statiques ne sont pas entièrement satisfaisantes, des solutions directement dynamiques ont été proposées. La première idée est de redéfinir les communautés comme des objets dynamiques.

2.2.3.1 Modification des fonctions de qualité

La première solution a été présentée par [46] et propose une fonction de qualité séparée en deux termes : un pour la qualité statique sur l'instantané considéré et un second pour assurer la stabilité :

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

où $Q_{instant}$ est une fonction de qualité statique (comme la modularité, mais d'autres fonctions sont utilisés), $Q_{stabilite}$ est un terme qui évalue la distance entre la nouvelle partition et la précédente et α un paramètre caractérisant l'importance de la stabilité. Cette nouvelle fonction de qualité permet d'obtenir une série de partitions plus intéressantes en réduisant le nombre d'artefacts causés par l'algorithme d'optimisation. Les auteurs utilisent cette

fonction pour étendre divers algorithmes tels que *k-means* et un algorithme de partitionnement hiérarchique. Des idées similaires sont utilisées par [17].

Une extension de cette approche [76] est de non pas utiliser un terme de stabilité mais un terme de qualité globale. Ainsi, on n'impose pas à la partition à l'instant $t+1$ d'être proche de la partition précédente mais d'être pertinente à l'instant $t+1$ et la plus pertinente possible à l'instant t. Cela apporte indirectement de la stabilité. Nous présentons au chapitre 4 une généralisation de cette idée en cherchant des partitions bonnes sur plusieurs pas de temps, consécutifs ou pas, et non sur seulement deux pas de temps.

Une approche similaire est décrite dans [89] où les auteurs tiennent compte de l'histoire non pas dans la fonction de qualité mais dans la matrice d'adjacence. En considérant que la matrice d'adjacence à l'instant t est W_t, les auteurs cherchent donc à maximiser la qualité sur un graphe ayant pour matrice d'adjacence $W = \alpha W_t + (1-\alpha)W_{t+1}$. Ils proposent également des moyens de trouver le meilleur α.

D'autres modifications existent telles que [78] pour étudier des réseaux multi-mode, c'est à dire des réseaux ou les nœuds et les liens peuvent correspondre à des objets de types différents. Un exemple serait un réseau d'auteurs et des papiers de recherches. Des nœuds seraient de type auteurs et d'autres de types papiers et il y a un lien entre un papier et ses auteurs et des liens de types différents entre deux papiers qui se citent.

Le problème de la stabilité est aussi étudié dans [42]. Au lieu de modifier la fonction de qualité, la stabilité est forcée en modifiant les distances entre les objets à classer. Leur méthodologie ne s'applique pas qu'aux graphes mais à tout problème de classification. La distance considérée entre deux objets u et v, notée $dist'(u,v)$, qui sert à classer les objets en communautés d'objets proches est modifiée à partir de la distance réelle à l'instant t, notée $dist_t(u,v)$, en :

$$dist'(u,v) = dist_{t+1}(u,v) + \alpha dist_t(u,v)$$

Ils peuvent ensuite appliquer tout algorithme de classification basé sur les distances comme DBSCAN. On peut aussi appliquer des méthodes de détection de communautés utilisant la structure du graphe en modifiant les poids de la même manière que les distances.

2.2.3.2 Approches modèles

Les approches modèles consistent à décrire un modèle aléatoire de génération de graphes ayant un certain nombre de paramètres puis à trouver les paramètres qui permettent de mieux rendre compte des données. Parmi ces paramètres, il peut y avoir l'existence de communauté, par exemple en ayant des probabilités différentes de relier des nœuds suivant leurs communautés.

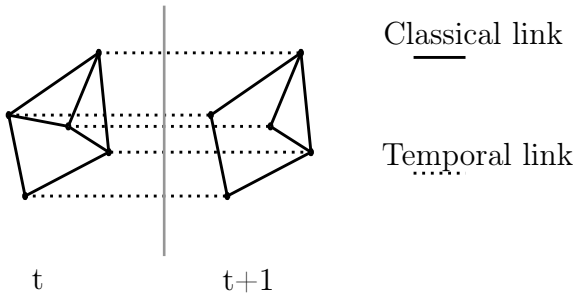


FIGURE 2.6 – Exemple de graphe temporel : deux instantanés d'un même réseau sont placés côte à côte à deux pas de temps, et les communautés sont calculées sur ce nouveau réseau.

La définition du modèle dépend souvent de connaissances préalables des auteurs et l'évaluation des paramètres repose sur des méthodes d'évaluation variées. Ces approches ont en général de solides bases théoriques comparées aux nombreuses heuristiques que nous avons présentées mais passent par contre souvent assez mal à l'échelle et leur technicité les rend souvent difficiles à appréhender pour les utilisateurs ce qui freine leur adoption.

Une proposition de modèle dynamique a été faite dans [83, 84] dont les auteurs formulent une fonction de qualité comme un problème de factorisation de matrices non négatives qui optimise conjointement la qualité et la stabilité des communautés. Ils proposent divers algorithmes ensuite pour trouver en pratique les communautés et testent leur méthode, qu'ils appellent FacetNet, sur plusieurs jeux de données. Leur méthode est très générale car elle permet à la fois d'avoir des communautés recouvrantes, dynamiques et de trouver une association.

Dans [90], les auteurs étudient une modification tenant compte de la dynamique du *stochastic block model*. Il s'agit d'un modèle de génération de graphes ajoutant des nœuds un par un. Quand un nœud est ajouté, il est affecté à une communauté selon une loi de probabilité π et des liens sont ensuite ajoutés selon une loi de probabilité dépendant des communautés des extrémités. Ils changent donc l'assignement dans une communauté pour tenir compte de la décomposition précédente. Leur évaluation semble prometteuse mais reste limitée à des cas relativement petits (quelques dizaines de pas de temps et quelques centaines de nœuds).

2.2.3.3 Graphes en tranches temporelles

L'information temporelle peut aussi être codée dans le graphe lui-même. Étant donné plusieurs instantanés d'un graphe dynamique, ils peuvent être placés côte à côte pour construire un nouveau réseau. Ainsi, les nœuds qui

existent à plusieurs pas de temps dans le graphe dynamique apparaissent plusieurs fois dans le nouveau réseau. Chaque instantané correspond à une tranche du nouveau réseau et il est possible d'ajouter des liens inter-tranches, par exemple entre un nœud existant dans la tranche à t et lui-même dans la tranche à $t + 1$. Il est ensuite possible d'oublier temporairement les tranches et le nouveau réseau n'est plus qu'un graphe statique particulier avec deux types de liens : ceux qui existent à un moment donné et ceux inter-tranches (voir figure 2.6). Ce graphe peut ensuite être étudié normalement et on peut détecter des communautés dessus. Ces communautés contiendront des nœuds de tranches distinctes et donc regrouperont des nœuds existant à des moments différents. Le problème principal vient du choix des liens inter-tranches à ajouter.

La première tentative dans cette direction est proposée dans [41] où les auteurs construisent un graphe inter-tranche et appliquent dessus l'algorithme *WalkTrap* [70] en mettant des liens entre le même nœud aux instants successifs. Les communautés sur ce réseau contiennent des nœuds de différents pas de temps et sont des communautés temporelles. Le problème du suivi est alors automatiquement résolu, les communautés existent sur plusieurs pas de temps par défaut.

Cette idée est étendue dans [54] où une méthode plus générale de connections entre les tranches est proposée. Ses auteurs proposent aussi une modification de la modularité pour intégrer directement la notion de tranche et l'utilisent pour détecter des communautés à plusieurs échelles et dans des réseaux dynamiques.

2.2.3.4 Conclusion

Plusieurs idées ont été proposées pour étudier les communautés, certaines basées sur des algorithmes statiques et d'autres prenant directement en compte la dynamique. Un des manques majeurs actuels concerne les outils de validation. En effet, il existe des fonctions de qualité dans le cas statique faisant consensus et que les auteurs d'algorithmes peuvent utiliser pour les valider ainsi que des réseaux de tests, qu'ils soient artificiels [49] ou réels associés à une connaissance extérieure d'une bonne décomposition [91]. Cependant, de tels outils n'existent pas dans le cas dynamique. Un modèle basé sur le modèle statique de [49] et un autre basé sur [61] ont été proposés respectivement dans [36] et dans [42] mais ils sont trop récents pour avoir été utilisés.

2.2.4 Algorithmes incrémentaux

Les algorithmes incrémentaux visent à analyser un flux de données plutôt qu'un instantané complet. L'entrée est alors une séquence d'événements sur le réseau et l'algorithme essaye de maintenir une décomposition de qualité

en mettant à jour sa décomposition courante plutôt qu'en recalculant une décomposition à partir de rien. Ceci est particulièrement utile dans le cas d'une analyse en temps réel d'un réseau ou pour le cas de jeux de données énormes tels que le web ou l'Internet par exemple. En effet, si les mises à jour peuvent être prises en compte très efficacement, l'algorithme incrémental peut avoir des performances bien meilleures qu'un calcul de communautés à chaque pas de temps indépendamment. De plus, les algorithmes incrémentaux ont souvent naturellement une mémoire de ce qui s'est passé et donc tiennent au moins partiellement compte de l'historique et de la dynamique.

Dans [62], les auteurs proposent un algorithme incrémental qui calcule les communautés à l'aide des valeurs propres du réseau. En ne considérant que les changements, ils atteignent une décomposition en communautés dix fois plus rapide qu'un calcul direct sur le nouvel instantané au prix d'une petite baisse de qualité. Néanmoins, cette comparaison est faite par rapport à un algorithme de classification spectrale qui n'est pas le plus rapide et donc l'algorithme incrémental est plus lent sur certaines instances que d'autres algorithmes classiques.

Un algorithme du même type est proposé dans [27]. Les auteurs définissent tout d'abord une distance entre les nœuds d'un réseau puis définissent le voisinage d'un nœud comme la boule topologique de rayon ε. Ils ne considèrent que les nœuds dont le voisinage est plus grand qu'une limite donnée et les considèrent comme des nœuds cœurs. Les nœuds présents dans le voisinage d'un nœud cœur sont des nœuds frontière. Ils définissent ensuite les communautés comme les unions des voisinages partageant des nœuds. Cela définit un algorithme de détection de communautés statiques très similaire à l'algorithme de clustering standard DBSCAN. Ils proposent ensuite des techniques pour mettre à jour les voisinages et les communautés quand des nœuds sont ajoutés ou enlevés.

Deux algorithmes d'optimisation de la modularité, la méthode de Louvain et Glouton Rapide, sont modifiés dans [35] pour gérer des petites modifications du réseau. Ces deux algorithmes sont des heuristiques qui déplacent des nœuds de communautés en communautés de manière gloutonne et regroupent des communautés pour optimiser la modularité. En considérant qu'ils ont déjà une partition, les auteurs appliquent les mêmes heuristiques mais en fixant les nœuds non affectés par les changements du réseau dans leur communauté précédente. Ainsi, seulement un petit nombre de nœuds sont déplaçables et les algorithmes s'exécutent donc bien plus vite que s'ils étaient appliqués au réseau entier.

Finalement, un autre algorithme incrémental est proposé dans [24]. Au lieu de réagir aux événements, les auteurs encodent le réseau en un réseau entre communautés particulier : chaque communauté correspond en fait à deux nœuds, reliés par un lien dont le poids est la somme des poids des

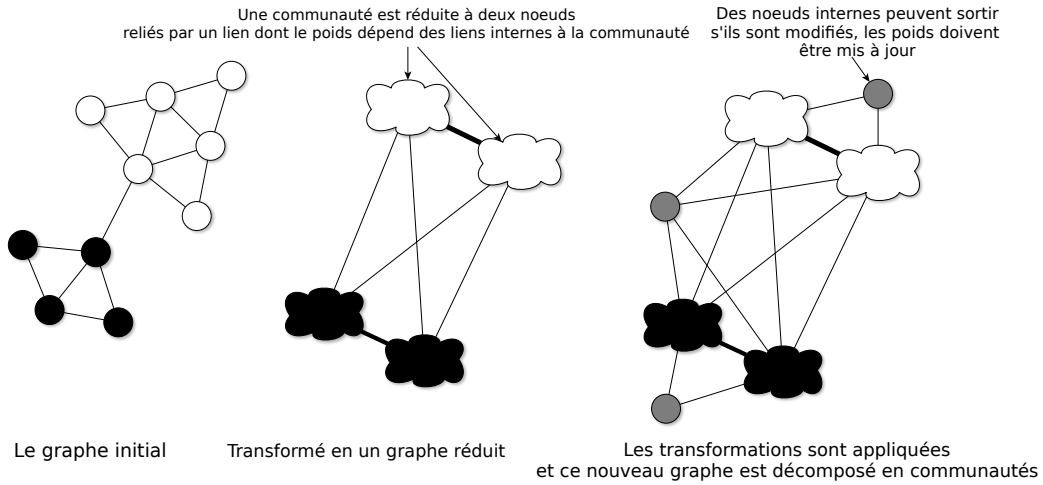


FIGURE 2.7 – Processus d'évolution des communautés de [24]. On construit un graphe entre communautés, on intègre les modifications puis on recalcule les communautés sur ce nouveau graphe.

liens internes. Il y a aussi un lien entre deux communautés dont le poids est la somme des poids les reliant. Cela produit un réseau bien plus petit que le réseau original. Ensuite, quand une transformation a lieu, les auteurs transforment ce graphe entre communautés. Ils appliquent ensuite l'algorithme de détection de communautés sur ce réseau. Le processus est décrit sur la figure 2.7. Le réseau entre communautés reste en général bien plus petit que le réseau original et donc les performances sont meilleures. Une particularité de cette technique est qu'elle ne dépend pas de l'algorithme de décomposition employé car seul le réseau d'entrée est modifié.

2.2.5 Analyse de l'évolution

Analyser les communautés dans des réseaux dynamiques apporte de nouveaux problèmes. En effet, de nouvelles propriétés directement liées à l'évolution apparaissent, comme l'espérance de vie ou la vitesse de croissance des communautés. En général, il n'existe pas de consensus sur les propriétés intéressantes et de nombreuses définitions sont très similaires, ne changeant que la normalisation par exemple.

2.2.5.1 Stabilité des partitions

La première question pour analyser l'évolution de communautés est la stabilité : comment réagit la décomposition en communautés quand le réseau change ? S'il y a beaucoup de transformations, cela peut vouloir dire soit qu'il

y a des changements structuraux dans le réseau, soit que l'algorithme est instable et que les transformations observées sont liées à l'algorithme. Pour cela, des métriques pour comparer deux partitionnements ont été définies. Cela a aussi de nombreuses applications pour évaluer la qualité d'un partitionnement en comparant par exemple les résultats obtenus avec des résultats connus.

Vu que c'est une question cruciale, de nombreuses distances ou mesures de similarité ont été définies. On peut trouver une revue des méthodes de comparaison de partitions dans [69].

Les premières métriques sont basées sur la comparaison deux à deux des nœuds : elles testent si deux nœuds sont tous deux ensembles ou tous deux séparés dans les deux partitions. Une première mesure de ceci est l'indice de Rand [71].

Définition 2 *Étant donné un ensemble S de n éléments et deux décompositions en communautés, X et Y, à comparer et étant donnés :*
- *a, le nombre de paires d'éléments de S qui sont dans la même communauté dans X et dans Y.*
- *b, le nombre de paires d'éléments de S qui sont dans des communautés différentes dans X et dans Y.*
- *c, le nombre de paires d'éléments de S.*

L'indice de Rand est défini comme $\frac{a+b}{c}$.

Intuitivement, $a + b$ est le nombre de paires d'éléments pour lesquelles les deux décompositions sont d'accord. L'indice prend des valeurs comprises entre 0 et 1 et est maximum quand les deux partitions sont égales. Il existe beaucoup d'autres mesures similaires basées sur cette idée de comparaison des paires d'éléments mis ensemble ou non. On peut citer l'indice de Jaccard, le coefficient de Minkowski ou le coefficient de similarité de Fowlkes-Mallows ainsi que leurs normalisations. Un grand nombre de tels indices est répertorié et ils sont utilisés pour comparer des partitionnements dans [82].

Une autre idée similaire basée sur des probabilités est utilisée dans [12]. Les auteurs calculent des communautés à différents pas de temps et définissent la stabilité comme l'espérance pour deux nœuds dans la même communauté de rester ensemble. Plus précisément ils calculent étant donnés deux nœuds i et j et étant donné que $C_k(t)$ est la communauté du nœud k à l'instant t :

$$\mathbb{E}\left[\mathbb{P}\left(C_i\left(t\right) = C_j\left(t\right) | C_i\left(t-1\right) = C_j\left(t-1\right)\right)\right]$$

Calculer la stabilité en se basant sur les paires de nœuds peut donner une très grande importance à la division d'une grande communauté car alors de très nombreuses paires sont cassées. C'est une des raisons pour laquelle une autre métrique basée sur l'information mutuelle définie initialement pour la théorie de l'information a été introduite. On la trouve par exemple dans [22]

pour comparer des algorithmes statiques entre eux et dans [31, 32] pour analyser la dynamique de réseaux financiers.

Définition 3 *Étant données deux partitions A et B d'un ensemble de n nœuds, l'information mutuelle entre A et B est définie comme :*

$$MI\left(A,B\right)=\sum_{a,b\in A\times B}\frac{\left|a\cap b\right|}{n}\cdot\log\left(\frac{\left|a\cap b\right|\cdot n}{\left|a\right|\cdot\left|b\right|}\right)$$

C'est une application aux partitions de l'information mutuelle de la théorie de l'information qui compte le nombre de bits partagés par deux variables aléatoires. Il existe plusieurs normalisations de celle-ci. Par exemple :

$$NMI\left(A,B\right)=\frac{-2\sum_{a,b\in A\times B}\left|a\cap b\right|\cdot\log\left(\frac{\left|a\cap b\right|\cdot n}{\left|a\right|\cdot\left|b\right|}\right)}{\sum_{a\in A}\left|a\right|\cdot\log\left(\frac{\left|a\right|}{n}\right)+\sum_{b\in B}\left|b\right|\cdot\log\left(\frac{\left|b\right|}{n}\right)}$$

Une valeur de 0 signifie que les partitions sont indépendantes et une valeur de 1 qu'elles sont égales. Cette métrique normalisée est intéressante mais souffre d'un certain nombre de problèmes. Tout d'abord, il y a plusieurs façons de normaliser, qui produisent des résultats différents, sans qu'aucune ne soit clairement la bonne. Elle est aussi corrélée au nombre de communautés [25] et est très difficile à interpréter. À part pour les valeurs extrêmes, il est impossible de décider facilement si deux partitions sont proches. Nous définirons une autre métrique plus intuitive au chapitre 3.

Dans [31], les auteurs utilisent une notion de stabilité des nœuds : ils étudient si un nœud reste avec les mêmes nœuds dans sa communauté durant toute la durée de mesure. Ils définissent donc le coefficient d'auto-corrélation d'un nœud i à l'instant t étant donné que $c_i\left(t\right)$ est la communauté du nœud i à l'instant t comme :

$$a_i^t\left(\tau\right)=\frac{\left|c_i\left(t\right)\cap c_i\left(t+\tau\right)\right|}{\left|c_i\left(t\right)\cup c_i\left(t+\tau\right)\right|}$$

Compris entre 0 et 1, il est d'autant plus petit que la communauté change entre t et $t+\tau$. Ensuite, ils prennent la moyenne de celui-ci quand t varie et comparent cette moyenne pour plusieurs valeurs de τ et plusieurs nœuds. Ils établissent que les nœuds fortement connectés à leur communauté ont tendance à effectivement conserver plus longtemps leur communauté.

Dans [66], les auteurs définissent une métrique similaire mais pour une communauté, étant donné que ses successeurs sont connus :

$$Corr_C^t(\tau)=\frac{\left|C(t)\cap C(t+\tau)\right|}{\left|C(t)\cup C(t+\tau)\right|}$$

Ensuite, ils calculent la moyenne de $Corr(t)$ sur toutes les communautés nées avec la même taille et compare son évolution en fonction de τ. Cette autocorrélation décroît plus vite si les communautés sont plus grandes, ce qui signifie que les membres changent plus rapidement dans les grandes communautés. Ils quantifient ensuite cela en définissant la stationarité d'une communauté comme la corrélation moyenne entre ses différents états :

$$\xi(c) = \frac{\sum_{t=t_0}^{t_{max}-1} Corr_C^t(t+1)}{t_{max} - t_0 - 1}$$

Cela caractérise combien une communauté est stable durant sa vie. Il apparaît que les plus grosses communautés sont moins stables que les petites.

2.2.5.2 Évolution des communautés

Mesurer la stabilité est important mais un autre problème consiste à mesurer à quelle vitesse les événements ont lieu et comment les communautés évoluent. Dans [81], les auteurs étudient une série de quatre instantanés du web mesurés entre 1999 et 2002 et définissent certains paramètres à un pas de temps donné pour décrire à quelle vitesse évolue une communauté (voir le tableau 2.1).

Nom	Définition	Représente
taux de croissance	$\frac{\lvert c' \rvert - \lvert c \rvert}{\delta_t}$	vitesse d'évolution de la taille d'une communauté
nouveauté	$\frac{\lvert c' - c \rvert}{\delta_t}$	nombre de nouveaux nœuds dans la communauté
taux de disparition	$\frac{\lvert c' - c \rvert}{\delta_t}$	nombre de disparitions de nœuds dans la communauté
taux de regroupement	$\frac{\lvert c' \cap C - c \rvert}{\delta_t}$	nombre de nœuds absorbés venant d'une autre communauté
taux de scission	$\frac{\lvert c \cap C' - c' \rvert}{\delta_t}$	nombre de divisions de c

TABLE 2.1 – Propriétés proposées par [81] avec c une communauté au temps t, c' la communauté correspondante au temps $t+1$, C l'union de toutes les communautés à t et C' l'union des communautés à $t+1$.

Des mesures similaires sont définies par [77] pour étudier la dynamique de communautés suivies par la méthode MONIC. Les auteurs proposent le ratio de survie (proportion de communautés à t qui existent encore à $t+1$), le taux d'absorption (la proportion de communautés absorbées) et le *pass-forward ratio* (la somme des survies et des absorptions) afin d'analyser des communautés à l'intérieur de la bibliothèque de l'ACM.

Des métriques décrivant les changements à l'intérieur d'une communauté telle que sa *popularité* sont définies dans [4]. La popularité est le nombre de nœuds rejoignant la communauté moins le nombre de nœuds la quittant. Dans un jeu de données de papiers scientifiques, il apparaît que la communauté XML est devenue très populaire dans les années 2000 ce qui correspond à un fait connu et tend à prouver la pertinence de la métrique.

2.2.5.3 Détection d'événements

La détection d'événements est un autre problème lié à la stabilité. L'objectif est d'être capable de faire la distinction entre des transformations normales, liées à l'évolution habituelle du réseau, et les changements anormaux.

Dans [30], les auteurs calculent des communautés à plusieurs pas de temps en suivant le principe de compression maximale : toute régularité dans un jeu de données permet de le compresser, c'est-à-dire de le décrire en moins de symboles qu'initialement. Une communauté est donc un regroupement de nœuds qui permet une meilleure compression du graphe : elle maximise la compression. Ils construisent donc une liste de communautés à chaque étape. Si une décomposition en communauté est proche de la précédente, elle peut être efficacement compressée en les regroupant tandis que si elle est sensiblement différente, son codage direct va être aussi efficace. Ces moments où la compression n'est pas bonne leur permettent de détecter des événements.

2.2.5.4 Évolution des nœuds parmi les communautés

Une autre question est de savoir comment les nœuds bougent de communautés en communautés. Pour étudier cette évolution, les auteurs de [4] définissent plusieurs métriques telles que la *sociabilité* ou l'*influence* d'un nœud. La sociabilité est définie comme le nombre de communautés auxquelles un nœud appartient au cours de sa vie. Cette métrique peut être utilisée par exemple pour faire de la prédiction de liens.

Ils définissent aussi l'influence d'un nœud, qui caractérise si d'autres nœuds rejoignent ou quittent une communauté quand il le fait. Ainsi, en notant n un nœud, *companions* l'ensemble des nœuds qui rejoignent ou quittent une communauté quand n le fait et m le nombre de fois ou n rejoint ou quitte une communauté, on défini l'influence de n comme : $influence(n) = \frac{|companions|}{m}$. Cela a pour effet de rendre influents les suiveurs d'un nœud influent, ce qui est problématique. Les auteurs définissent donc des règles pour diminuer cet effet : soit n' le nœud qui a le plus d'interactions avec n, alors l'influence de b est nulle si le degré ou l'influence de n' est supérieur à ceux de n. Cela exclu les suiveurs, mais réduit l'étude aux nœuds très influents car tous les autres ont une influence nulle et cela limite les interactions possibles entre des nœuds influents.

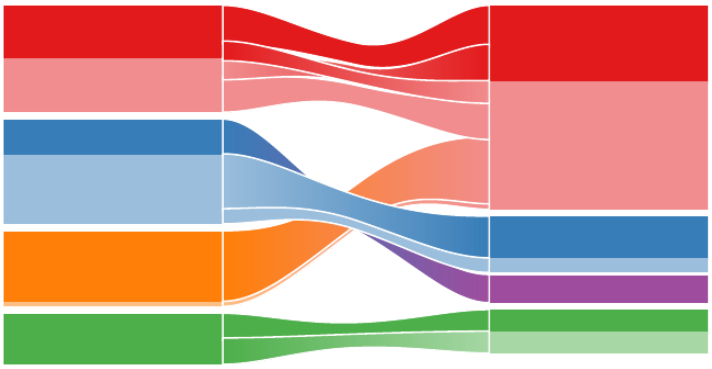


FIGURE 2.8 – Visualisation de la dynamique entre communautés (Extrait de [75])

Enfin, [75] propose un outil de visualisation des dynamiques de communautés. Ils proposent des diagrammes de flots représentant bien les changements structurels entre deux pas de temps ainsi que l'évolution des nœuds parmi les communautés (voir figure 2.8).

2.2.5.5 Analyse de cas réels

Finalement, quelques études de cas existent et apportent des éclairages et des explications sur divers comportements des communautés.

Premièrement, [45] analyse les caractéristiques globales d'un réseau de blog entre 1999 et 2002 et étudie le comportement des liens intra-communautés. Ils utilisent la notion de *burstiness* définie par Kleinberg dans [43]. Les liens intra-communautaires apparaissent en général plus par paquets (une grande quantité de liens apparaissent rapidement au même moment) et il y a de plus en plus de communautés qui ont tendance à grossir. Ce résultat est en fait plutôt un résultat historique et n'est donc pas général car leur jeu de données correspond à l'apparition de la blogo-sphère mais est un éclaircissement intéressant sur l'émergence des blogs.

Dans [8], les auteurs étudient deux jeux de données. Le premier est extrait du dépôt de papiers scientifiques *dblp* et le second d'un site web de nouvelles et de communications, *LiveJournal*. Ils ne font pas de détection de communautés sur les données, mais utilisent des connaissances sur le réseau pour définir des groupes. L'article est donc intéressant non pas par la définition des groupes mais par les méthodes d'analyse de l'évolution de ceux-ci. Dans les jeux de données *dblp*, les auteurs sont regroupés s'ils participent à la même conférence et dans les jeux de données LiveJournal, les utilisateurs peuvent eux-mêmes décider de rejoindre des groupes de discussions. Les auteurs se demandent quelles sont les raisons structurelles qui conduisent un nœud à

rejoindre un groupe et pourquoi certains groupes grossissent. Pour expliquer cela, ils définissent de nombreuses propriétés d'un nœud et construisent un arbre de décision à l'aide de ces propriétés. Les premières propriétés de l'arbre sont celles qui permettent de discriminer le plus un nœud. Dans LiveJournal, la propriété la plus importante est la proportion d'amis dans la communauté qui sont eux-mêmes amis. De la même manière, ils construisent un arbre de décision pour prédire la croissance ou non de groupes. Ils s'interrogent aussi sur le lien entre sujets d'un groupe de discussion et utilisateurs : est-ce que le sujet existe et les utilisateurs le rejoignent ou est-ce les utilisateurs qui apportent le sujet après s'être regroupés.

Finalement, [66] étudie l'espérance de vie de communautés en fonction de leur taille et de leur stabilité. L'espérance de vie est la durée entre l'apparition de la communauté et sa dissolution. Ils étudient un réseau d'appels téléphoniques ainsi qu'un réseau de co-auteurs de papiers scientifiques et découvrent qu'il existe deux cas suivant que la communauté est grande ou petite. Si la communauté est petite, elle doit être très stable si elle veut survivre longtemps, tandis qu'une grande communauté doit au contraire accepter des changements comme des nouveaux membres et des départs.

2.3 Réseaux étudiés

Nous présentons maintenant les différents jeux de données utilisés durant cette thèse. Le tableau 2.2 résume les caractéristiques de ces réseaux.

Blogs [21]. Pendant quatre mois, approximativement six mille blogs ont été monitorés afin de suivre les différents articles ou commentaires postés et les liens de citations entre eux. Nous avons utilisé les réseaux entre les blogs contenant les données agrégées jusqu'au jour considéré. Nous commençons donc par un réseau vide au jour 0, puis ajoutons chaque jour les nouveaux blogs et liens entre eux. Nous obtenons donc un réseau croissant constitué de 120 pas de temps. Ce réseau croît lentement et régulièrement et il y a un événement de mesure connu au jour 40 (de nouveaux blogs ont été ajoutés au pool de blogs monitorés).

Mrinfo [68]. Le deuxième réseau correspond à une cartographie de la topologie des routeurs multicast sur Internet. Elle a été mesurée grâce à l'outil `mrinfo`. Ce programme permet de demander à un routeur multicast la liste de ses voisins. Chaque jour, `mrinfo` a été lancé sur un premier routeur puis ensuite récursivement sur chacun de ses voisins à la manière d'un parcours en largeur. Cette mesure a été menée durant plusieurs années ce qui a permis d'obtenir une carte dynamique des routeurs multicasts. Pour plus de détails à propos de la mesure, nous renvoyons à [68]. Nous n'utiliserons ici que les données de l'année 2005, ce qui représente 365 pas de temps contenant chacun 3100 nœuds en moyenne. L'évolution de ce réseau est constituée de 3 phases.

La première phase dure pendant les 52 premiers jours durant lesquels le réseau est assez instable et contient de nombreux événements. La seconde phase est entre les jours 52 et 117 et est bien plus stable. Enfin, la troisième phase dure du jour 117 à la fin et est la plus longue et est plus stable que la première mais moins que la seconde. La structure des graphes instantanés est en général plus proche entre la première et la troisième phase qu'avec la seconde phase.

Radar [50]. Il s'agit d'une autre cartographie de la topologie entre les routeurs sur Internet. Au lieu d'une mesure relativement lente de la topologie des routeurs multicast telle qu'obtenue à l'aide de `mrinfo`, Radar est une mesure rapide, prenant en compte tout type de machines, effectuée à l'aide de `traceroute`. Une machine sert de source et un ensemble de machines servent de destinations. Par l'intermédiaire d'une modification de `traceroute`, les chemins entre la source et chacune des destinations sont obtenus, construisant un arbre enraciné sur la source représentant sa vision *égocentrée* du réseau et qui est appelé une passe. Nous utilisons ici une mesure effectuée toutes les quinze minutes environ et nous avons un peu plus de 5000 passes. Afin d'étudier des graphes et non des arbres, nous ajoutons à chaque passe les 9 suivantes pour avoir une union glissante de 10 passes. Ce réseau est très dynamique : les routes changent énormément sur Internet et donc les routeurs intermédiaires vus aussi.

Imote [40]. Il s'agit d'un réseau de capteurs représentant la proximité entre les participants de la conférence Infocom en 2005. Plusieurs participants se sont vus remettre un périphérique *Bluetooth*, appelé Imote, permettant de détecter les autres Imotes à proximité et de conserver cette information. L'expérience a duré trois jours à partir du soir précédant la conférence jusqu'à la fin de la troisième matinée. La mesure est extrêmement bruitée car les périphériques Bluetooth sont très imprécis et échouent régulièrement à détecter les périphériques alentour. Ainsi, un tout petit mouvement d'un participant peut suffire à changer pour un court instant ses voisins. De plus, les participants bougent et les liens changent donc parfois très vite. Le réseau est donc extrêmement dynamique et comporte de nombreux changements. Il contient 41 nœuds et est divisé en 25000 pas de temps. Tous les pas de temps ne sont pas séparés par la même durée. Le réseau est en outre très stable et peu dense durant la nuit et très instable le jour.

2.4 Conclusions

Nous avons tout d'abord défini dans ce chapitre la notion de communautés dans les graphes statiques sur laquelle nous allons nous baser par la suite. Il s'agit des éléments d'une partition ayant la plus grande modularité possible. Nous avons vu qu'il existe de très bons algorithmes pour chercher cette partition dans les graphes statiques. Les principales questions restantes sont du

Réseau	Nombre de pas de temps	Nombre de nœuds moyens	Nombre d'arêtes moyen	Nombre moyen de changements de nœuds	Nombre moyen de changement d'arêtes
Blogs	120	3503	13782	37	215
Mrinfo	364	3114	7523	456	1539
Radar	499	11783	14254	725	3667
Imote	20379	20.2	29.5	0.7	2.2

TABLE 2.2 – Tableau des caractéristiques des différents réseaux étudiés.

point de vue statique comment gérer les différentes échelles et le recouvrement c'est-à-dire de permettre aux communautés d'en contenir elles-mêmes et aux nœuds d'appartenir à plusieurs communautés. Une autre problématique est l'étude des communautés sur des graphes dynamiques.

Cette thèse portant sur cet aspect, nous avons passé en revue les différentes méthodes proposées pour intégrer la notion de dynamique dans la détection de communautés. Nous avons tout d'abord étudié les méthodes reposant sur des algorithmes statiques consistant à détecter des communautés à chaque instant et ensuite faire le lien entre ces partitions. Cela pose encore problème car aucun consensus sur la bonne façon de procéder pour faire cette association n'émerge. Les méthodes proposées reposent sur des ensembles de règles et trouver un ensemble minimum, complet et consensuel de règles semble difficile. De plus, ces règles impliquent souvent des paramètres délicats à fixer. En particulier, les algorithmes ont tendance à être instables, c'est à dire qu'ils produisent des résultats différents dans des situations similaires et donc il faut souvent un paramètre pour décider que deux communautés sont les mêmes quand elles partagent $x\%$ de leurs nœuds. Mais ce x semble souvent fixé à des valeurs qui nous semblent trop relâchées comme 70 où 80% des nœuds. On verra dans le chapitre 3 que l'instabilité des algorithmes statiques est effectivement trop grande pour une utilisation directe dans le cas dynamique et nous proposerons une méthode de stabilisation de la méthode de Louvain.

La deuxième approche consiste à considérer directement la dynamique dans la définition de communautés. Ici encore de nombreuses techniques concurrentes existent sans qu'aucune ne soit clairement la bonne. Toutes ont tendance à considérer la structure comme quelque chose de continu dans le temps et cherchent donc à décrire l'évolution selon une ligne de temps. Nous proposons un autre algorithme au chapitre 4 permettant de se détacher de cette limite. Cela permet par exemple de chercher des structures répétées ce qui est difficile à mettre en évidence en se focalisant uniquement sur le passage de l'instant t à l'instant $t + 1$.

La multitude d'algorithmes sans vrai unificateur est sans doute due au manque général d'outils de validation. Chaque article vient avec ses propres présupposés et objectifs et cherche à les retrouver dans les données. Cette méthode de validation est très limitée car d'une part on n'a pas forcément d'hypothèses à valider et on cherche juste à analyser un réseau sans présupposés et d'autre part elle est assez biaisée car on a toujours tendance à trouver ce que l'on cherche. Nous avons utilisé cette méthode de validation peu satisfaisante au chapitre 4 car c'est encore aujourd'hui la seule existante. Définir une méthode d'évaluation plus rigoureuse constitue selon nous une des pistes majeures d'améliorations de nos algorithmes mais cela aurait été impossible sans les analyses que nous allons effectuer par la suite.

Chapitre 3

Instabilités

Sommaire

La première approche que nous avons étudiée pour détecter des communautés sur les graphes dynamiques consiste à appliquer des algorithmes statiques connus à chaque pas de temps du réseau dynamique. Cela produit un grand nombre de partitions, et la difficulté est d'expliciter ce qui se passe entre deux pas de temps consécutifs. Notre première préoccupation a été la stabilité des algorithmes : comment réagissent-ils quand on modifie le réseau ? Si les résultats des algorithmes de détection de communautés sont instables, il est illusoire de vouloir suivre des communautés, car on observera essentiellement des artefacts liés à l'algorithme et non des changements structurels. Nous avons analysé trois algorithmes : la méthode de Louvain, *WalkTrap* et Glouton Rapide. Parmi ceux-ci, la méthode de Louvain a la particularité de ne pas être déterministe. Elle requiert un ordre sur les nœuds, des ordres différents pouvant induire des résultats différents. *WalkTrap* et Glouton Rapide

sont déterministes sauf quand il y a à choisir entre plusieurs maxima. Cependant, une petite modification de la topologie peut aussi avoir une grande influence sur leur exécution et il est tout à fait possible que ces algorithmes produisent des résultats très éloignés pour des graphes très proches.

Dans la suite, nous allons montrer tout d'abord qu'effectivement ces trois algorithmes classiques de détection de communautés sont trop instables pour être utilisés tels quels pour analyser un réseau dynamique. Pour cela, nous allons définir une distance entre partitions et analyser la distance entre des partitions de graphes similaires. Puis, nous proposerons une modification de la méthode de Louvain pour la rendre plus stable et permettre une analyse de la dynamique plus réaliste.

3.1 Distance d'édition

La première tache, lorsque l'on veut comparer des partitions, est de mesurer la distance entre elles. Il existe de nombreuses métriques de distance entre partitions comme l'information mutuelle décrite au chapitre précédent. Elle est comprise entre 0 et 1 : une information mutuelle de 0 implique que les partitions sont indépendantes et de 1 qu'elles sont égales. Cette métrique est largement utilisée mais souffre d'un défaut important : elle est très délicate à interpréter. En dehors de valeurs extrêmes (très proches de 1 ou de 0) il est difficile de conclure à une proximité ou un éloignement entre partitions. De plus, il a été prouvé que l'information mutuelle est corrélée à la taille de la partition [25] et n'est donc pas adaptée pour comparer des algorithmes qui vont donner un nombre différent de communautés.

Nous avons par conséquent décidé d'utiliser en parallèle de l'information mutuelle une autre mesure, qui nous semble plus facile à interpréter : une distance d'édition entre partitions. Étant données deux partitions, nous allons donc compter combien de nœuds doivent être déplacés pour passer dz l'une à l'autre. Pour définir ceci précisément, nous allons considérer une bijection entre les parties de deux partitions (nous ajoutons des parties vides s'il n'y en a pas le même nombre, mais nous verrons que dans le cas d'une évolution assez stable, le nombre de communautés change rarement). Pour chaque bijection, il y a un coût correspondant au nombre de nœuds n'étant pas dans l'image de leur communauté. Ainsi, la figure 3.1 montre deux bijections possibles. La première induit 6 mouvements tandis que la seconde en induit seulement 2. Nous cherchons ensuite parmi toutes les bijections possibles celle minimisant ce nombre de mouvements.

Plus précisément, nous allons d'abord définir le coût de l'association entre deux communautés :

Définition 4 *Étant donnés deux ensembles S_1 et S_2, le coût de leur association est $|S_1 \backslash S_2|$.*

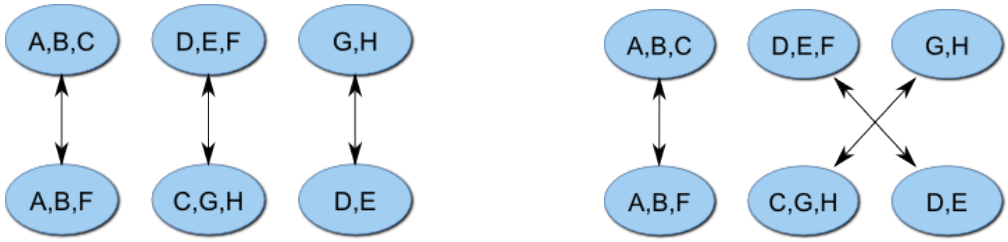


FIGURE 3.1 – Exemple d'associations via des bijections. La bijection de gauche implique 6 mouvements (C, D, E, F, G et H) tandis que la bijection de droite implique 2 mouvements (C et F). Elle est la bijection optimale et la distance entre les deux partitions est donc 2.

Cela dénombre le nombre d'éléments de S_1 qui devront changer de communautés si S_1 est associé à S_2. Une association entre deux partitions π_1 et π_2 est la fonction permettant de dire quelle communauté de π_2 est devenue une communauté de π_1. Nous allons supposer que les partitions ont le même nombre de communautés, quitte à en ajouter des vides si nécessaire.

Définition 5 *Une association entre deux partitions π_1 et π_2 ayant le même nombre de parties est une bijection associant à chaque partie de π_1 une partie unique de π_2.*

Nous pouvons maintenant définir le coût d'une association qui compte le nombre de nœuds à déplacer pour passer d'une partition à l'autre si leurs parties sont associées suivant la bijection.

Définition 6 *Le coût d'une association δ entre les partitions π_1 et π_2 est la somme des coûts entre les parties et leur image :*

$$cout(\delta) = \sum_{p \in \pi_1} |p \backslash \delta(p)|$$

On peut ensuite définir la distance entre deux partitions comme le coût minimal atteignable :

Définition 7 *La distance entre deux partitions est le coût minimum parmi toutes les associations :*

$$dist(\pi_1, \pi_2) = \min_{\delta \ association}(cout(\delta))$$

Si deux partitions n'ont pas le même nombre de parties, nous ajoutons les parties vides nécessaires pour pouvoir avoir une bijection. Cela définit aussi

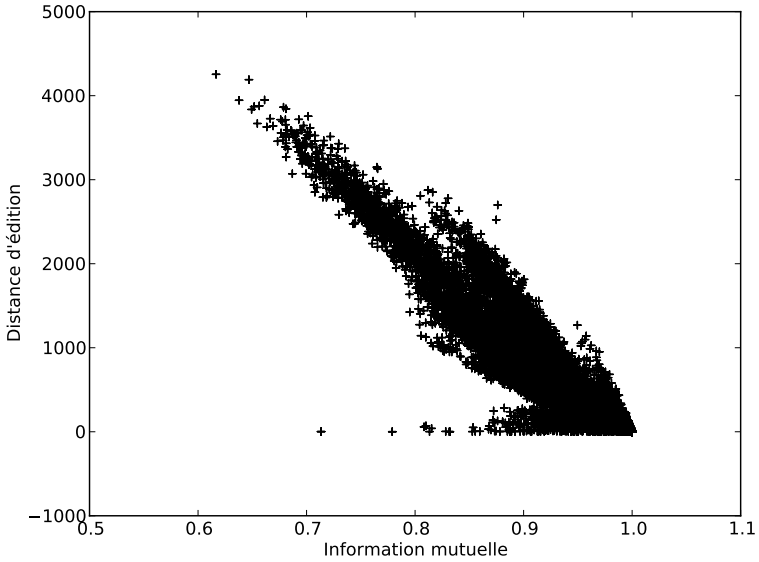


FIGURE 3.2 – Corrélation entre l'information mutuelle et la distance d'édition.

une association entre les communautés et permet de dire, dans un premier temps, ce qu'il s'est passé entre deux pas de temps. L'association peut être calculée en $O(n^3)$ avec n le nombre de parties avec l'algorithme de Kuhn-Munkres [44].

La figure 3.2 présente la corrélation entre l'information mutuelle et la distance d'édition entre partitions sur les dynamiques simulées que nous détaillons par la suite. Les informations mutuelles proches de 1 ont lieu quand la distance d'édition est faible et réciproquement, les informations mutuelles faibles ont lieu quand la distance d'édition est haute. Les deux avantages de la distance d'édition sont qu'elle donne une association et une valeur bien plus intuitive. Un désavantage est le besoin d'une bijection : il n'y a pas de fusion, de scission ou de vraie apparition d'une communauté et par conséquent la distance peut énormément défavoriser les moments où cela arrive.

3.2 Dynamique simulée

Maintenant que nous sommes en mesure d'estimer la distance entre des partitions, nous avons besoin de réseaux dynamiques. Afin d'étudier la stabilité des algorithmes, nous avons besoin d'une évolution contrôlable, pour laquelle nous savons si la dynamique est stable ou non et qui soit aussi simple que possible. Nous avons malgré tout besoin que les pas de temps contiennent des communautés, ce qui exclut les méthodes de génération de graphes aléa-

toires comme l'attachement préférentiel.

Nous allons donc simuler une dynamique très simple à partir d'un réseau statique existant : nous allons enlever les nœuds un par un, dans un ordre aléatoire, tout en gardant uniquement la composante connexe la plus grande. C'est clairement une dynamique irréaliste, mais enlever un seul nœud ne devrait pas changer la structure globale du réseau sauf peut-être lors du retrait de nœuds centraux ou de hubs importants et les partitions calculées par les algorithmes devraient être très stables.

3.3 Instabilité

Nous présentons maintenant les résultats obtenus en utilisant Arxiv comme réseau initial. Il s'agit d'un graphe entre scientifiques, connectés s'ils ont écrit un papier ensemble, tiré de la base de données *arxiv.org* et extrait par Newman dans [56]. Afin de simplifier le réseau et comme ils n'ont pas d'influence sur les communautés, nous avons enlevé les nœuds connectés à un seul nœud et avons ajouté des boucles sur leur voisin. Au final, le réseau contient 9377 nœuds et 24107 liens. Les résultats obtenus avec d'autres réseaux initiaux sont qualitativement très similaires. De même, enlever les arêtes une par une et non les nœuds produit les mêmes résultats.

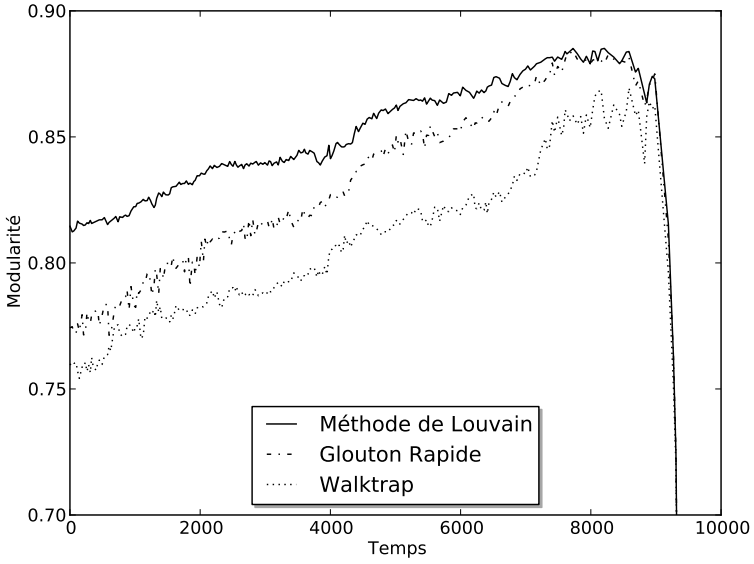


FIGURE 3.3 – Modularité de chaque partition durant une exécution.

Nous avons tout d'abord calculé à chaque étape la modularité. Les résultats sont représentés sur la figure 3.3. Pour chaque algorithme, la modularité croît

lentement jusqu'à un point de rupture où elle s'effondre. À ce moment, le nombre de nœuds du réseau s'écroule aussi car chaque suppression tend à déconnecter de grandes parties. La méthode de Louvain produit les résultats ayant la meilleure qualité, suivie par Glouton Rapide puis par *WalkTrap*. En terme de vitesse, l'ordre reste le même. Un point remarquable est que la modularité varie très peu entre deux pas de temps consécutifs. Cela tend à confirmer que retirer un nœud choisi aléatoirement ne change effectivement pas la structure globale du réseau.

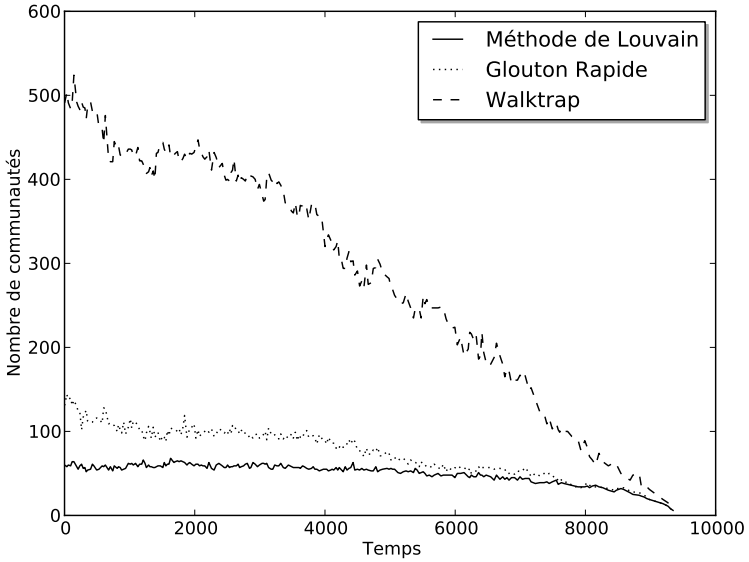


FIGURE 3.4 – Nombre de communautés durant une exécution (suppression de tous les nœuds).

Le nombre de communautés, représenté sur la figure 3.4, est aussi très stable pour la méthode de Louvain et Glouton Rapide. Walktrap est moins stable si on considère l'exécution complète, mais localement il n'y a que peu de variations. Nous sommes donc dans le cas ou la distance d'édition est une bonne mesure car il y a sans doute peu de fusions et scissions.

La figure 3.5 présente l'information mutuelle entre chaque partition et sa précédente. Pour *WalkTrap*, elle est proche de 1 à chaque fois, et est donc très stable ce qui est encourageant. En revanche, pour les deux autres algorithmes, l'information mutuelle varie beaucoup et est comprise entre 0.75 et 0.98 pour la méthode de Louvain et entre 0.65 et 0.97 pour Glouton Rapide. La distance varie donc énormément mais reste dans des plages de valeurs souvent considérées comme indiquant que les partitions sont similaires. En revanche, la distance d'édition présentée sur la figure 3.6 donne une vision complètement

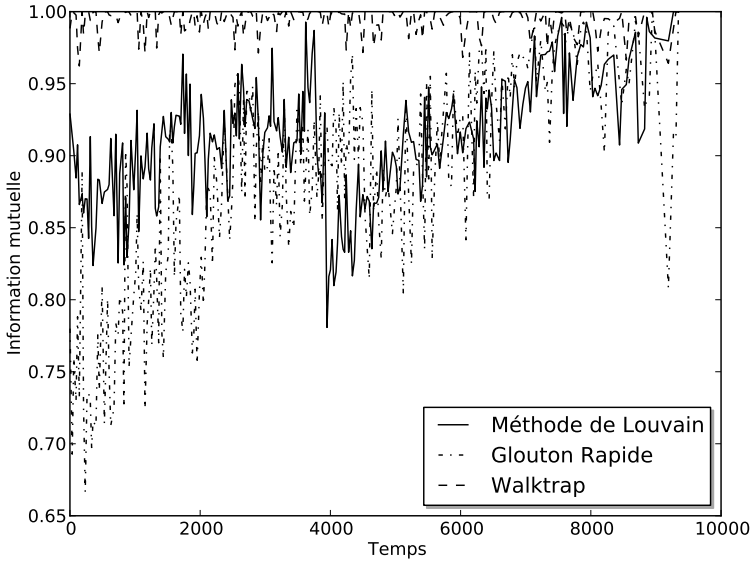


FIGURE 3.5 – Information mutuelle entre deux partitions successives durant une exécution.

différente : après la suppression d'un nœud parmi initialement 9377, entre 2000 et 3000 nœuds changent de communautés avec Glouton Rapide et entre 1500 et 2500 avec la méthode de Louvain. Cela veut dire que les communautés détectées entre chaque pas de temps sont très différentes (plus du quart des sommets ont bougé !) et que donc ces deux algorithmes sont extrêmement instables. Les résultats de *WalkTrap* sont clairement meilleurs, mais il y a régulièrement 500 changements ce qui est toujours beaucoup pour une simple suppression de nœud. Nous verrons par la suite que *WalkTrap* est en fait très instable quand le réseau change un peu plus.

La conséquence principale de cette instabilité est que ces algorithmes sont totalement inexploitables tels quels pour faire de la détection de communautés dynamiques sur plusieurs pas de temps. Il est en effet impossible de savoir si les événements détectés sont liés à des changements structurels dans le réseau étudié ou juste causés par l'algorithme. Nous allons par la suite étudier les instabilités de la méthode de Louvain puis allons proposer une modification mineure permettant un gain de stabilité très grand avec une perte de qualité très faible.

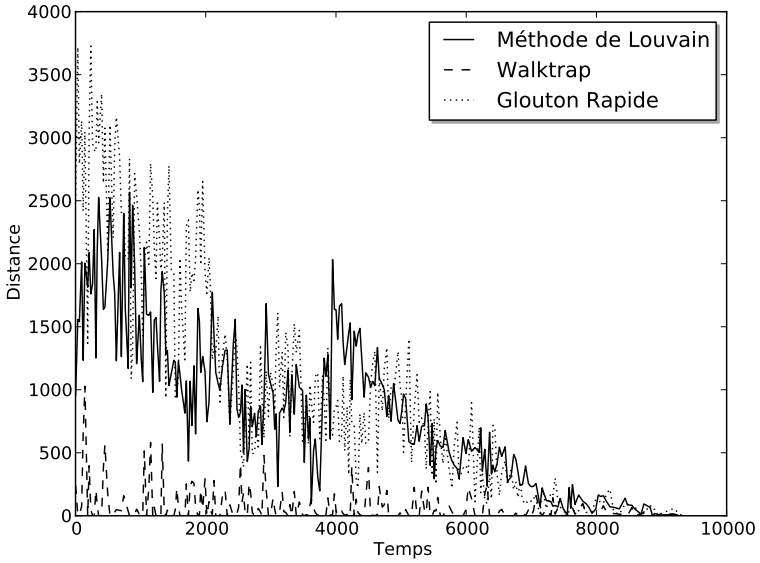


FIGURE 3.6 – Distance d'édition entre deux partitions successives durant une exécution.

3.4 Les instabilités de la méthode de Louvain

3.4.1 Origines

La méthode de Louvain fait des choix non déterministes à plusieurs moments :

- Le premier choix est l'ordre d'évaluation des nœuds durant une itération. En effet, les nœuds sont évalués un par un et placés dans la communauté voisine maximisant le gain de modularité. Suivant l'ordre d'évaluation les choix vont être différents et les fusions aussi. Si au début de l'algorithme, deux nœuds particuliers sont regroupés, ils forment une communauté et d'autres nœuds évalués après pourraient la rejoindre alors que sans ce regroupement ils auraient rejoint une autre communauté voisine.
- Un deuxième choix, plus classique, a lieu quand il existe plusieurs communautés voisines permettant le même gain de modularité. Là encore, un choix au début peut avoir beaucoup d'influence sur la suite.
- Enfin, au moment du regroupement du graphe pour construire le graphe entre communautés, on crée un nouveau graphe entre communautés. Il existe beaucoup de représentations possibles de ce graphe, toutes isomorphes mais sur lesquelles les ordres d'évaluation des nœuds vont sans doute varier par exemple.

Nous allons montrer que bien que chacune de ces étapes soit "déterminisable", c'est-à-dire que l'on peut ajouter quelques règles pour que l'algorithme effectue toujours le même choix, l'instabilité reste assez grande. Nous allons aussi voir que divers paramètres ont de l'influence sur la stabilité et que toutes les communautés n'ont pas la même stabilité, mais qu'aucune de ces solutions ne permet d'améliorer de manière satisfaisante la stabilité.

3.4.2 Déterminisation

Nous allons tout d'abord déterminiser l'algorithme en utilisant un ordre donné sur les nœuds. Cela permet de toujours considérer les nœuds dans le même ordre, de choisir toujours la même communauté en cas d'égalité de gain de modularité (on choisit celle contenant le plus petit nœud) et on peut continuer à trier les regroupements en les classant suivant le plus petit nœud qu'ils contiennent.

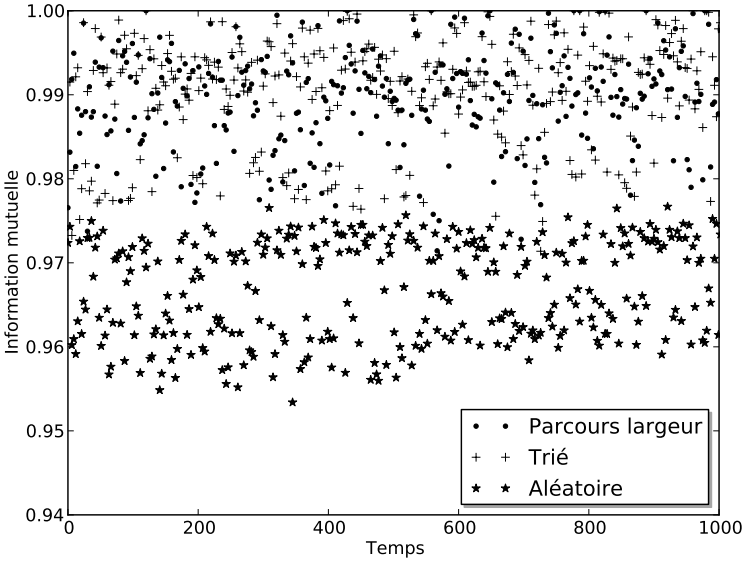


FIGURE 3.7 – Distance entre partitions successives avec différents ordres. Soit un ordre fixé, soit des parcours en largeur à partir du même nœud, soit un ordre aléatoire.

Nous allons d'abord étudier exclusivement l'influence de l'ordre d'évaluation des nœuds indépendamment des regroupements. Pour cela, nous allons arrêter l'algorithme avant le regroupement du graphe dans le graphe entre communauté, ce qui correspond à la première phase qui contient le plus de

communautés. La figure 3.7 montre l'information mutuelle [1] entre les partitions successives lors de la suppression de 1000 nœuds (à chaque fois les mêmes et dans le même ordre) en parcourant les nœuds suivant des ordres particuliers :

- Un ordre fixé a priori sur les nœuds pour les 1000 calculs.
- Un parcours en profondeur à partir d'une source fixée. Nous avons effectué l'expérience avec des parcours en largeur et les résultats sont similaires. Pour des raisons de lisibilité, nous ne les affichons pas.
- Un ordre aléatoire à chaque calcul.

Le meilleur ordre est l'ordre fixé car l'information mutuelle est plus élevée ($0,991$ en moyenne). Ensuite viennent les parcours en largeur ($0,989$ en moyenne) puis l'ordre aléatoire changé à chaque fois ($0,967$). L'important semble donc de suivre toujours le même ordre, mais cela est insuffisant pour obtenir un résultat stable et a peu d'influence. Nous présentons ici la stabilité pour le premier niveau qui a donc le plus de communautés et la qualité la plus faible et comme il s'agit du début de l'algorithme, les petites transformations ne sont pas encore vraiment propagées.

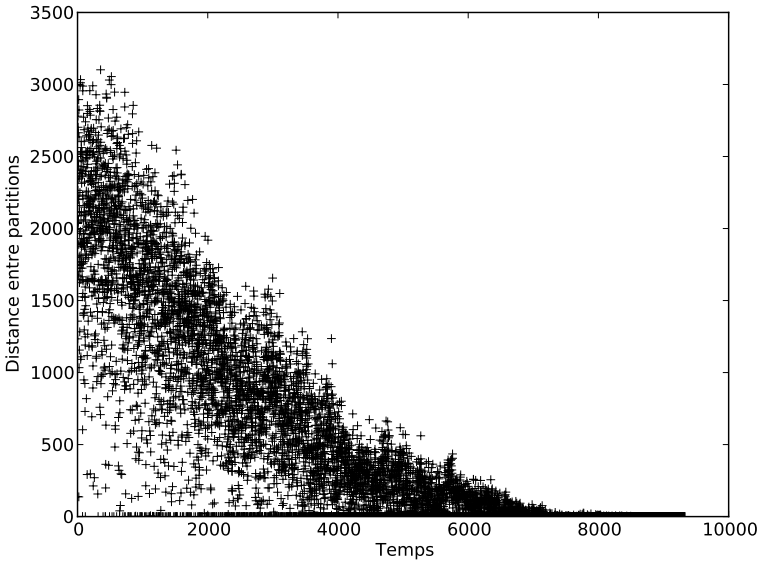


FIGURE 3.8 – Distance entre partitions successives en déterminisant l'algorithme.

Si nous continuons jusqu'au dernier niveau, la stabilité reste trop faible en déterminisant la méthode de Louvain. On a effectué la même expérience

1. Nous utilisons ici l'information mutuelle car le nombre de communautés au premier niveau est plus élevé et nous atteignons les limites de la distance d'édition.

mais avec une version déterminisée de la méthode de Louvain pour obtenir la figure 3.8. Il y a un gain de stabilité au final assez limité qui ne nous semble pas suffisant. En effet, il y a en moyenne encore 13% des nœuds qui changent de communautés pour une suppression de nœud.

Ainsi, il semble que déterminiser l'algorithme n'est pas suffisant et qu'il reste trop instable. Quand des nœuds apparaissent, l'ordre d'évaluation change beaucoup plus et donc le comportement de l'algorithme aussi. Nous allons maintenant étudier quelques propriétés ayant une influence sur la stabilité.

3.4.3 Résolution

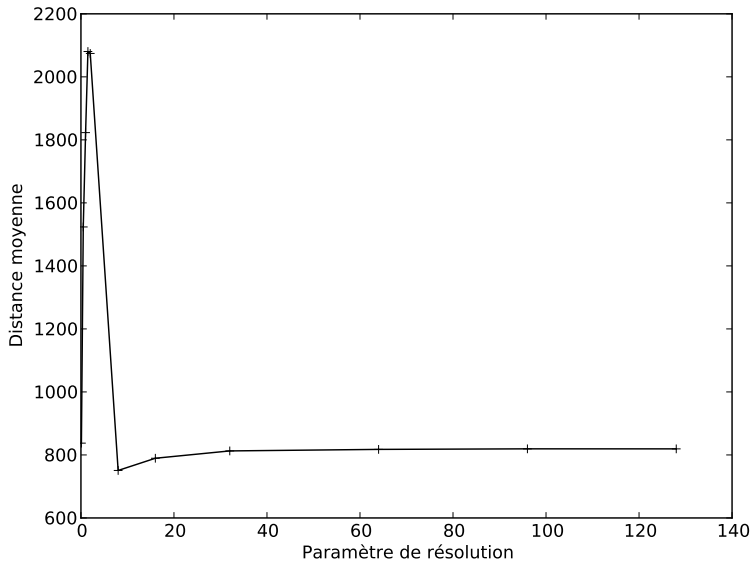


FIGURE 3.9 – Distance moyenne durant 1000 suppressions de nœuds en fonction du paramètre de résolution.

Nous allons maintenant étudier l'effet de la résolution limite, expliquée en détail en section 2.1.1 : la définition des communautés induit une taille intrinsèque qui fait que les communautés vont avoir tendance à être regroupées où divisée arbitrairement pour avoir la taille induite et augmenter la modularité. Nous avons utilisé le paramètre défini dans [47] pour contrôler la résolution des communautés et nous avons appliqué le même suivi à différentes résolutions. La moyenne des distances entre partitions successives en fonction du paramètre utilisé est donnée sur la figure 3.9. On voit qu'une augmentation du paramètre, ce qui correspond à une augmentation de la taille des communautés, conduit à une plus grande stabilité sans toutefois obtenir une bonne

stabilité. Il y a aussi une perte de qualité (on n'optimise plus la modularité) jusqu'à un cas dégénéré où il y a très peu de communautés.

L'amélioration de la stabilité peut être liée au changement de taille des communautés : les petits mouvements entre les petites communautés disparaissent car ces communautés sont absorbées dans une plus grande. Un cas dégénéré est quand il n'y a plus qu'une seule communauté, la stabilité est alors trivialement maximale.

La résolution est donc une part du problème, mais il nous semble difficile de se baser dessus pour stabiliser la détection de communautés : c'est à la fois insuffisant et on ne peut pas se restreindre aux seules résolutions stables.

3.4.4 Restriction à des groupes stables

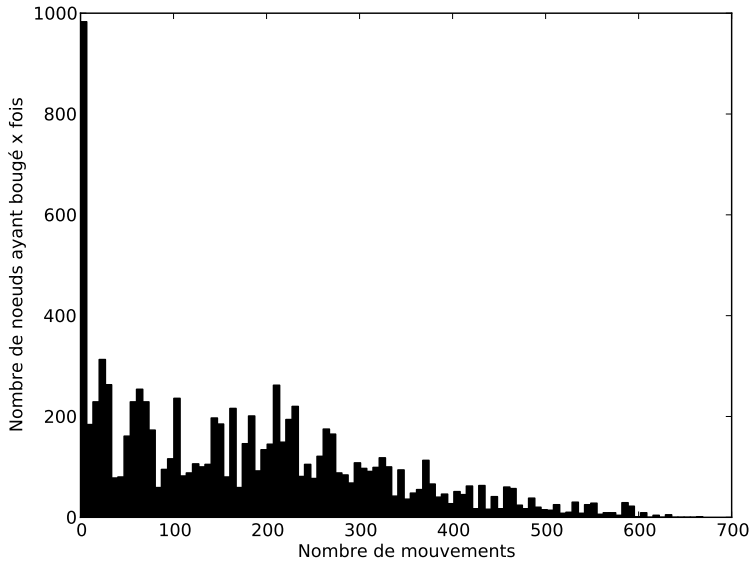


FIGURE 3.10 – Distribution du nombre de mouvements des nœuds.

Avant de chercher à améliorer la stabilité de l'algorithme, il convient de se demander quels sont les nœuds qui se déplacent. En particulier, il y a intuitivement deux types de nœuds : des nœuds cœurs qui bougent peu et des nœuds bordures plus instables. Pour étudier cela, nous avons calculé après 1000 suppressions combien de fois chaque nœud était considéré comme un nœud ayant bougé, c'est-à-dire combien de fois comptait chaque nœud dans la distance d'édition. Nous avons représenté la distribution du nombre de fois ou un nœud bouge sur la figure 3.10. Sur l'axe des abscisses il y a donc un nombre de mouvements et sur l'axe des ordonnées le nombre de nœuds ayant bougé ce

nombre de fois. Il s'avère qu'il n'y a pas vraiment de classification possible. Il y a en effet quelques nœuds qui ne bougent jamais, des nœuds qui bougent de temps en temps et des nœuds qui bougent très souvent, mais pas de rupture nette entre ces groupes, si ce n'est entre bougent/ne bougent jamais. De plus, on voit qu'il y a en fait assez peu de nœuds qui ne bougent jamais : moins de 800 soit moins de 10% des nœuds. Se contenter des nœuds stables revient à oublier beaucoup de nœuds et serait une grande limitation. Par conséquent, une stabilisation en séparant nœuds presque stables et nœuds instables semble illusoire, à moins d'imposer une limite qui au vu de la distribution ne pourra qu'être arbitraire.

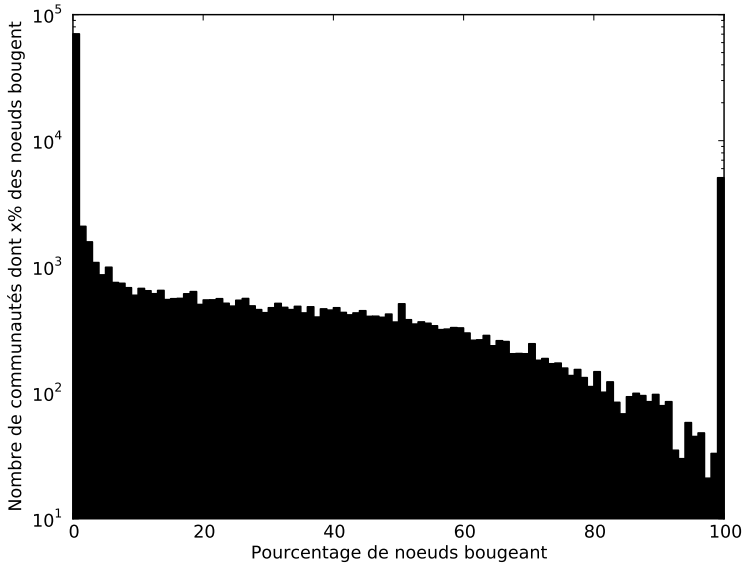


FIGURE 3.11 – Distribution de la proportion de nœuds changeant de communauté.

Nous avons aussi regardé s'il existait des communautés stables et d'autres instables. La figure 3.11 présente la distribution de la proportion de mouvements entre une communauté c_1 à un instant et la communauté à laquelle elle est associée à l'instant suivant. L'axe des abscisses est donc les valeurs de $\frac{|c_2-c_1|+|c_1-c_2|}{|c_1 \cup c_2|}$ (c'est-à-dire $\frac{nouveaux+perdus}{taille\,de\,l\,union}$) et l'axe des ordonnées le nombre d'assignements $c_1 \rightarrow c_2$ avec cette proportion de mouvements. L'axe des ordonnées est en échelle logarithmique sinon les valeurs intermédiaires n'apparaîtraient pas. Il y a deux cas extrêmes : quand l'association est "parfaite" et qu'il n'y a aucune perte ou apparition de nœuds dans la communauté ($|c_2-c_1|+|c_1-c_2| = 0$) et quand l'association est vraiment mauvaise ($c_1 \cap c_2 = \emptyset$ et $\frac{|c_2-c_1|+|c_1-c_2|}{|c_1 \cup c_2|} = 1$). Comme on peut le constater, ces deux cas se produisent

fréquemment. En particulier, la pire association possible est très représentée car elle peut correspondre au cas où on crée une communauté vide. Au milieu, il n'y a pas vraiment de rupture et il est donc impossible de séparer les communautés en des communautés stables et des instables sans soit être extrêmement rigide (une communauté est stable si elle n'a pas bougé du tout, or un seul nœud qui s'en va semble être une modification non importante) soit en fixant une limite qui est arbitraire au vu de la distribution.

Ainsi, il ne semble pas y avoir de vraies distinctions souples entre des nœuds ou des communautés stables et d'autres non. On observe tous les comportements sans saut permettant de les classer. Par conséquent, appliquer des méthodes ne suivant que les communautés stables force, soit à appliquer un critère arbitraire pour décider de qui est stable et qui ne l'est pas, soit à ne considérer que des communautés parfaitement stables. Ceci n'étant pas satisfaisant, nous proposons par la suite une méthode de stabilisation de la méthode de Louvain donnant des résultats bien plus réalistes.

3.5 Stabilisation par intégration de la partition précédente

3.5.1 Méthode

Afin de pouvoir faire du suivi de communautés, nous proposons une modification mineure de la méthode de Louvain. Le principe général est de changer l'initialisation afin de forcer la stabilité. Au lieu d'initialiser l'algorithme en plaçant chaque nœud tout seul dans sa communauté propre, nous allons le placer dans sa communauté précédente. L'algorithme commence donc dès le début avec de vrais groupes et non avec des communautés de taille 1. On peut ensuite appliquer exactement le même algorithme, en déplaçant les nœuds un par un dans leurs communautés voisines afin de maximiser le gain de modularité.

La figure 3.12 est similaire à la figure 3.6 mais avec la version modifiée de l'algorithme. La distance d'édition entre les partitions successives est considérablement réduite et donc les communautés sont bien plus stables. La plupart du temps, il n'y a aucun changement et parfois il y a des pics ou de petites modifications se produisent. C'est un résultat qui semble nettement plus réaliste pour des suppressions d'un seul nœud car, en général, enlever un nœud ne change pas la structure, sauf parfois avec quelques nœuds centraux.

Cependant, la stabilité n'est pas suffisante pour avoir une bonne partition. La figure 3.13 présente la modularité de la méthode de Louvain, de *WalkTrap* et de la méthode de Louvain modifiée. La plupart du temps, la méthode de Louvain modifiée est meilleure que *WalkTrap* et est proche de la méthode de Louvain. Néanmoins, la modularité de la méthode de Louvain modifiée évolue

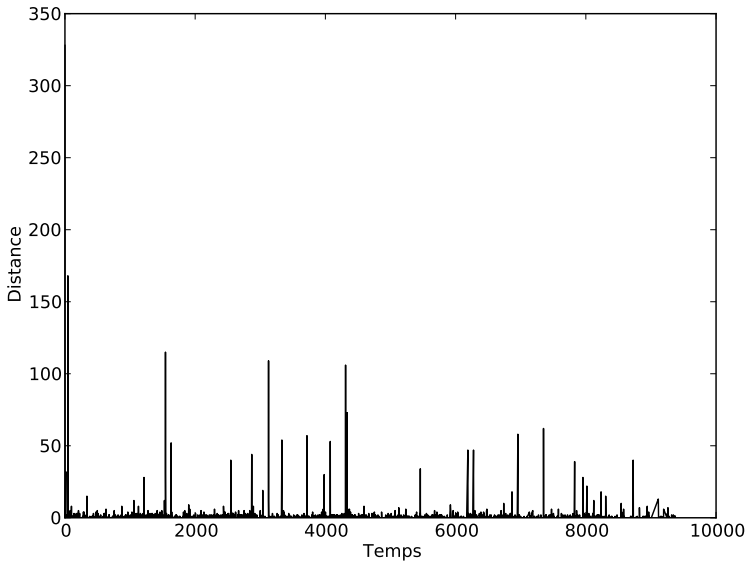


FIGURE 3.12 – Stabilité de la méthode de Louvain modifiée.

très peu au cours du temps. En fait, la partition change très peu et l'algorithme a du mal à faire évoluer la décomposition en communautés pour continuer à refléter la structure. La version modifiée reste meilleure car elle a un avantage initial en termes de modularité, mais quand suffisamment de nœuds ont été supprimés et que la structure a suffisamment changé, la méthode de Louvain modifiée devient moins bonne que *WalkTrap*. Cela est dû au fait que l'on fait commencer l'algorithme dans ce qui est très proche d'un maximum local, ce qui est rarement une bonne idée. Nous allons donc essayer de relâcher un peu la contrainte sur l'algorithme en ajoutant un peu d'aléatoire par l'intermédiaire d'un paramètre de qualité.

3.5.2 Relaxation de l'initialisation

Afin de limiter les contraintes sur l'algorithme modifié, nous allons d'abord placer tous les nœuds dans leur communauté précédente. Ensuite, nous allons choisir aléatoirement $x\%$ des nœuds et nous allons les sortir de leur communauté pour les mettre seuls. Plus x est grand, plus l'algorithme peut modifier sa sortie, car il déplacera forcément les nœuds seuls (placer un nœud seul dans sa communauté est toujours une mauvaise idée en terme de modularité). Si $x = 100\%$, on retrouve exactement la méthode de Louvain tandis que si $x = 0\%$ on retrouve exactement la version modifiée. Ainsi, le paramètre x est une sorte de curseur permettant de privilégier la modularité ou la stabilité.

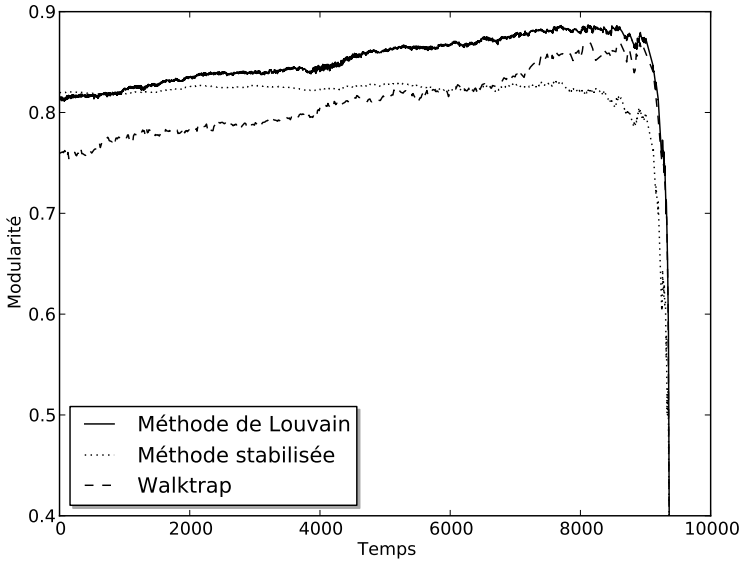


FIGURE 3.13 – Modularité de la méthode de Louvain modifiée, de la méthode de Louvain et de *WalkTrap*.

La figure 3.14 montre la modularité pour différentes valeurs de x. On voit que même pour des valeurs assez faibles, par exemple 2%, la modularité commence à croître et devient nettement meilleure.

La figure 3.15 montre la moyenne de la distance d'édition entre partitions durant les 1000 premières suppressions. Pour des petites valeurs de x, la stabilité n'est qu'un tout petit peu réduite tandis que pour de grandes valeurs, supérieures à 50, la stabilité décroit fortement jusqu'à atteindre la stabilité de la méthode de Louvain. Utiliser $x = 2\%$ semble un bon compromis ici : il y a une bonne stabilité et une haute modularité. Le choix de ce paramètre est malgré tout fortement lié à l'évolution du réseau. Si par exemple le réseau bouge suffisamment entre deux pas de temps, la partition précédente sera suffisamment éloignée du maximum local et donc la méthode de Louvain modifiée arrivera directement à la faire évoluer sans ajouter de l'aléatoire. Par contre, s'il y a peu de modifications entre chaque pas de temps, la partition précédente peut devenir une contrainte trop forte et il faut alors faire croître le paramètre x. Une bonne méthode semble d'être de lancer l'algorithme d'une part avec $x = 0\%$ et avec $x = 100\%$. Le cas $x = 100\%$ permet d'avoir une référence sur la modularité atteignable et on peut ensuite augmenter x petit à petit jusqu'à ce que la qualité soit satisfaisante en connaissant une référence. Par la suite, tous nos réseaux vont avoir une dynamique suffisamment grande pour pouvoir se contenter de $x = 0\%$.

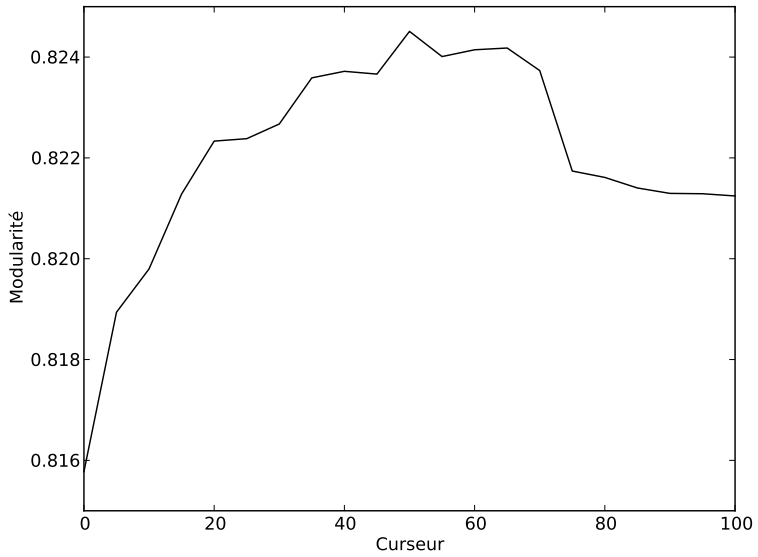


FIGURE 3.14 – Modularité en fonction du paramètre de qualité.

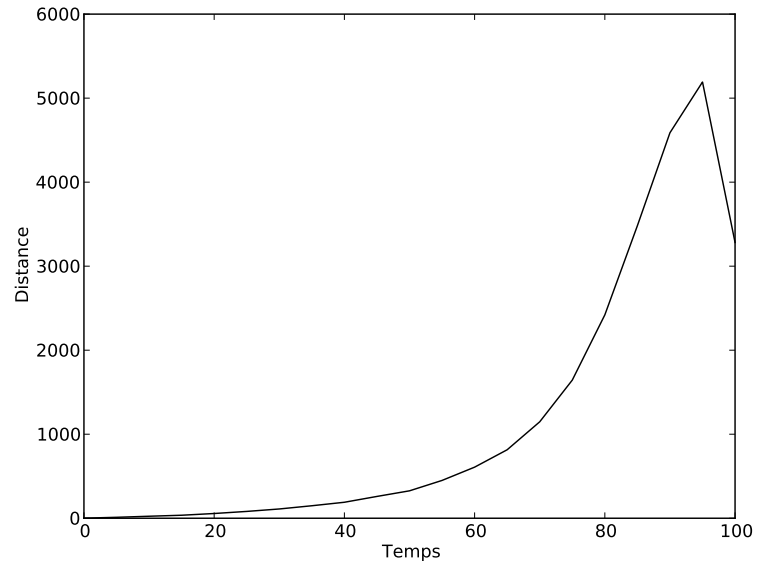


FIGURE 3.15 – Distance moyenne durant les 1000 premières suppressions avec différentes valeurs du paramètre de qualité.

3.6 Des événements ?

On observe sur la figure 3.12 quelques pics durant une exécution. Ce sont des moments où la structure communautaire évolue et donc les moments censés être importants. Ils peuvent être de vrais événements sur le réseau comme la suppression d'un nœud très important, mais ils peuvent aussi être des artefacts causés par l'algorithme. Pour analyser cela indépendamment de l'ordre de suppression, nous avons suivi une autre approche. Au lieu d'enlever les sommets un par un en réduisant à chaque fois la taille du réseau, nous les avons chacun enlevés en partant du réseau complet et avons donc regardé l'effet de la suppression du nœud.

3.6.1 Corrélation des événements à des propriétés des nœuds

Nous avons tout d'abord cherché à savoir si les pics pouvaient s'expliquer par une propriété des nœuds, comme une forte centralité. Nous avons donc calculé une partition π du graphe initial puis pour chaque nœud i une partition π_i obtenue à l'aide de la méthode de Louvain modifiée initialisée par π sur le graphe initial privé de i.

Ensuite, nous avons calculé plusieurs propriétés connues pour chaque nœud i sur le graphe initial ainsi que sur le graphe restreint à la communauté de i. Nous avons ainsi calculé pour chaque nœud sur le graphe initial et sur le graphe restreint à sa communauté :

- son degré, à savoir le nombre de liens reliés à ce nœud
- le nombre de triangles auxquels il participe. Le nombre de triangles est une mesure importante de cohésion et répond à la question "est-ce que mes amis sont amis ?" par exemple dans un réseau social.
- sa *centralité d'intermédiarité* ou *betweeness centrality* [13]. Il s'agit d'une métrique caractérisant l'importance d'un nœud. Elle compte la somme sur toutes les paires de nœuds de la fraction des plus courts chemins entre cette paire qui passe par le nœud considéré. Si une forte proportion de plus courts chemins passent par le nœud considéré, c'est que ce nœud est important.
- sa *centralité de proximité* ou *closeness centrality* [59]. Correspond à la distance moyenne des autres nœuds dans le réseau. On peut la voir comme le temps moyen que prendra une information apparaissant au nœud considéré pour atteindre les autres nœuds et une caractérisation de son importance.
- sa *load centrality* [57]. Elle correspond la fraction de tous les plus courts chemins qui traversent le nœud considéré (c'est une métrique rarement utilisée assez proche de la *betweeness centrality*).

– son *PageRank* [65]. Il s'agit d'une mesure d'importance utilisée initialement pour classer les pages Web. Elle mesure la probabilité d'être sur ce nœud pour un marcheur aléatoire en faisant quelques ajustements pour être certain de pouvoir la calculer.

Ensuite, nous avons essayé de corréler ces propriétés avec la distance entre π et π_i. Nous avons donc tracé pour chaque propriété la figure avec pour chaque nœud un point d'abscisse la propriété et d'ordonnée la distance. Si les deux propriétés sont liées, les courbes obtenues doivent faire apparaître une forme de corrélation. Nous avons représenté sur la figure 3.16 le résultat avec la centralité d'intermédiarité. Il ne semble pas y avoir de vraie corrélation, certains nœuds induisant une grande distance ont une faible centralité d'intermédiarité et réciproquement. Nous avons obtenu le même genre de courbes pour chaque propriété, ce qui tend à montrer que les grandes variations ne sont pas liées directement à des propriétés simples des nœuds.

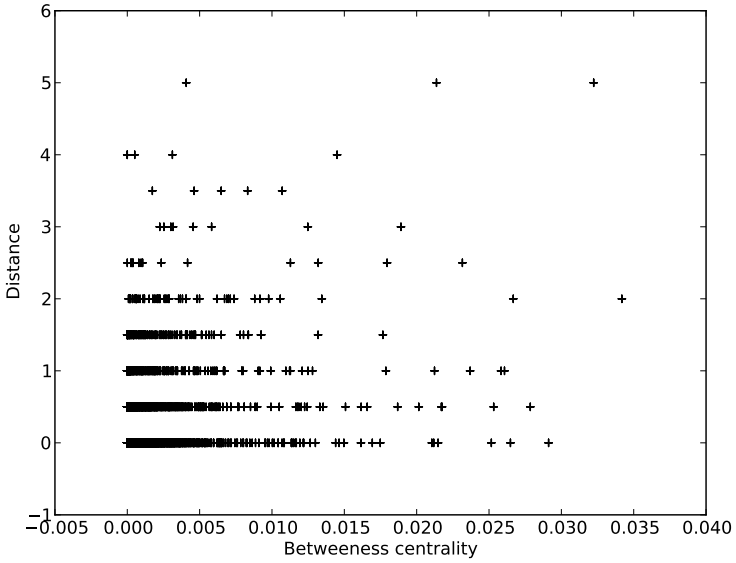


FIGURE 3.16 – Corrélation entre la centralité d'intermédiarité et la distance entre partitions.

3.6.2 Effet des ordres et conditions nécessaires pour obtenir les mêmes pics

Vu que les pics ne semblent pas pouvoir s'expliquer directement par une propriété du nœud enlevé, nous avons ensuite essayé de voir ce qui était né-

cessaire pour les expliquer. Nous avons donc regardé la distance après la suppression de chaque nœud indépendamment dans plusieurs situations :

- Avec deux ordres différents pour la méthode de Louvain et deux partitions initiales différentes.
- En repartant de la même partition mais avec deux ordres différents.
- Avec toujours le même ordre mais avec deux partitions initiales différentes.

Nous avons ensuite regardé pour chaque situation la corrélation entre les distances entre partitions obtenues pour chaque cas. Les résultats sont donnés sur les figures 3.17, 3.18 et 3.19. Pour chaque situation, nous avons tracé pour chaque nœud enlevé le point ayant pour abscisse la distance entre partitions pour le premier cas et pour ordonnée la distance dans le deuxième cas.

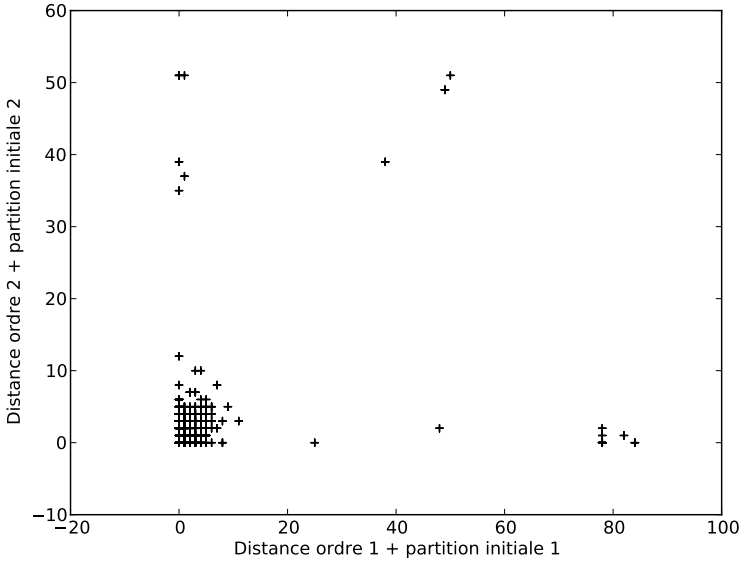


FIGURE 3.17 – Corrélation entre les distances entre partitions pour chaque nœud observées avec deux partitions initiales et deux ordres d'évaluation différents.

Le résultat pour le cas où les ordres et les partitions initiales sont différents est donné sur la figure 3.17. Comme on peut le voir, les variations de partitions ne sont pas corrélées, il y a des cas où la distance est grande avec le premier ordre et la première partition tandis qu'il est faible avec le deuxième ordre et la deuxième partition et réciproquement. Par conséquent, le nœud enlevé seul ne peut pas suffire à expliquer les variations car quand il est le seul invariant, on n'obtient pas les mêmes résultats.

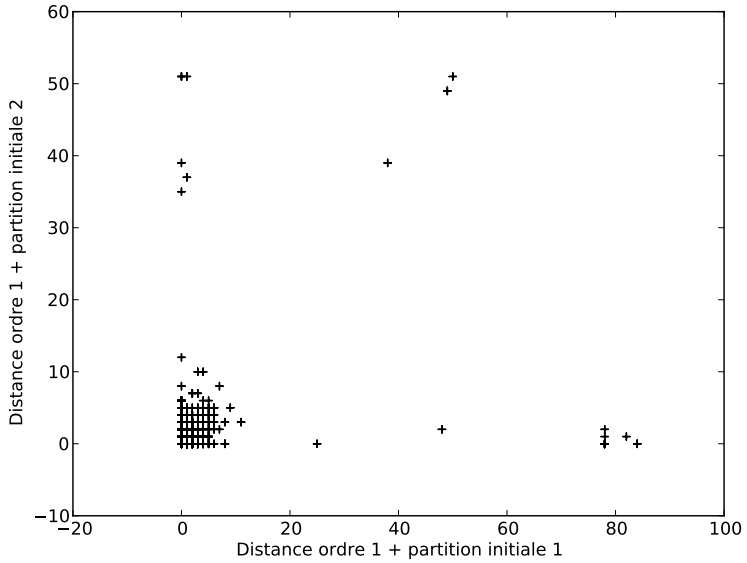


FIGURE 3.18 – Corrélation entre les distances entre partitions pour chaque nœud observées avec deux partitions initiales différentes.

Nous avons ensuite fait de même mais en utilisant à chaque fois le même ordre d'évaluation mais des partitions initiales différentes. Là encore, comme le montre la figure 3.18, la corrélation est faible. L'ordre et le nœud ensemble ne permettent donc pas d'expliquer les pics.

Par contre, quand on regarde sur la figure 3.19 la corrélation avec la même partition initiale mais des ordres d'évaluation différents, la corrélation entre les distances est très nette. On en déduit qu'il faut regarder la partition initiale et le nœud enlevé pour commencer à expliquer les pics et que l'ordre utilisé par la méthode de Louvain est nettement moins important.

3.6.3 Analyse des transformations durant les événements

3.6.3.1 Des groupes de nœuds déplacés ensemble

Sans réussir à identifier des causes à ces événements, nous avons aussi remarqué qu'ils correspondaient à des groupes de nœuds se déplaçant ensemble. Pour voir cela, nous avons sélectionné 1000 nœuds que nous avons enlevés un par un, mais en suivant des ordres différents et avec la même partition initiale (nous ne supprimons que 1000 nœuds, soit environ 10% du graphe, pour ne pas le déstructurer complètement). Nous avons remarqué que les nœuds entraînant des événements quand on les supprime étaient différents suivant les ordres de suppressions utilisés. Cela est cohérent avec le fait que les évé-

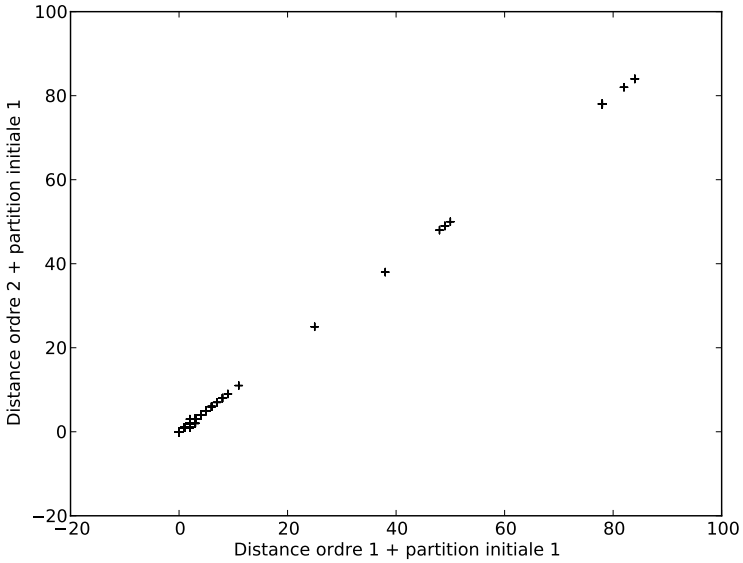


FIGURE 3.19 – Corrélation entre les distances entre partitions pour chaque nœud observées avec deux ordres d'évaluation différents.

nements ne sont pas liés uniquement au nœud enlevé, mais liés à la fois au nœud et à la partition. Néanmoins, pour chaque événement important dans un ordre, on pouvait trouver un événement proche dans un autre ordre. C'est à dire que pour chaque événement, on a identifié l'ensemble des nœuds déplacés et avons pu retrouver pour un autre ordre un événement dont l'ensemble des nœuds déplacés était proche. Ce sont donc potentiellement des groupes de nœuds qui bougent ensemble qui constituent un événement. Nous avons essayé d'identifier des ensembles de nœuds minimaux causant les événements en question mais sans succès.

3.6.3.2 Position des nœuds déplacés

Nous avons enfin étudié ce qu'il se passait durant les événements. Pour cela, nous avons regardé la position des nœuds changeant de communauté par rapport au nœud enlevé. Nous avons représenté sur la figure 3.20 la distribution de la distance moyenne des nœuds changeant de communauté. L'axe des abscisses est donc une distance moyenne et l'axe des ordonnées le nombre de pas de temps où cette distance moyenne a été observée. On remarque qu'en général, la distance moyenne est très proche de 1 ce qui signifie que les modifications de communautés sont proches de la transformation du graphe. Il y a quand même des cas où cette distance moyenne est grande (9 est une distance grande dans le cas d'un graphe petit monde comme un réseau social), mais

ceci arrive en fait quand il y a peu de transformations : un ou deux nœuds lointains changent de communautés et sont donc des artefacts de l'algorithme.

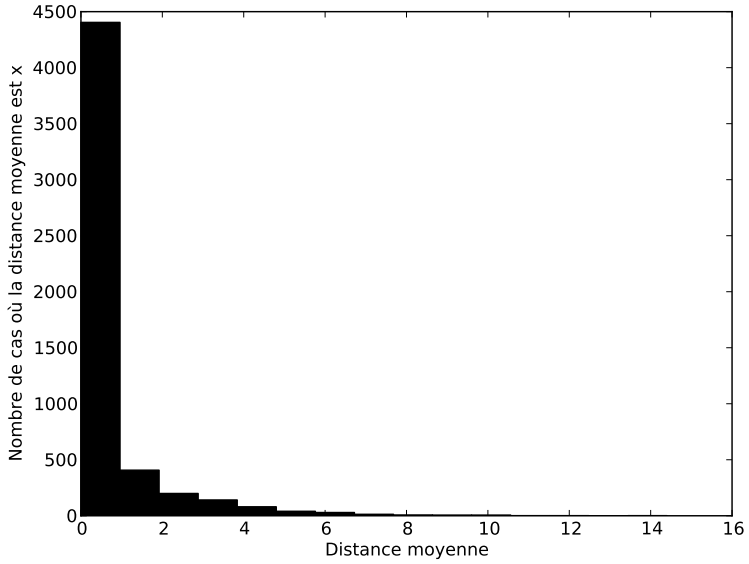


FIGURE 3.20 – Distribution de la distance moyenne des nœuds changeant de communauté avec le nœud enlevé.

Ainsi, les événements sont encore mystérieux. Ils sont liés à la fois au nœud enlevé et à la partition précédente, ce qui les rend difficiles à analyser. Leur comportement semble plus réaliste, mais sans outils de validation plus poussés il est difficile d'aller plus loin. Comme la dynamique étudiée précédemment était complètement artificielle, nous avons ensuite utilisé la même méthode mais sur nos jeux de données réelles.

3.7 Résultats sur de vrais réseaux

La figure 3.21 montre la modularité de chaque décomposition du réseau Blogs à chaque pas de temps. La méthode de Louvain a toujours la meilleure qualité, mais la version modifiée est très proche. Il n'y a pas besoin de changer le paramètre de qualité car le réseau évolue suffisamment entre chaque pas de temps. La figure 3.22 présente la distance d'édition entre partitions successives. La version modifiée est en revanche beaucoup plus stable que les autres, ce qui permet un suivi potentiel et son comportement est bien plus satisfaisant. Au départ, la partition est assez instable car le réseau se construit et est très dynamique : il y a beaucoup plus de différences entre le premier instantané qui est vide et le second par exemple qu'entre deux instantanés successifs de

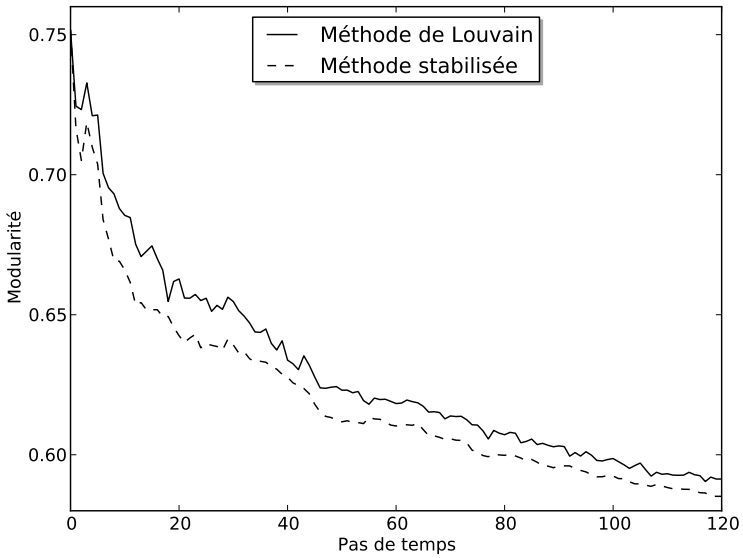


FIGURE 3.21 – Modularité de la méthode de Louvain modifiée sur le réseau Blogs.

la fin. Ensuite, il y a une stabilisation des communautés après le jour 50. Ce comportement est plus réaliste que celui des communautés trouvées par les autres algorithmes car il semble irréaliste que plus de 10% de la blogosphère change chaque jour. Il y a aussi un événement au jour 40 qui correspond à un événement de mesure connu : des blogs ont été ajoutés à la liste des blogs monitorés.

On constate des résultats similaires sur les réseaux Radar, Mrinfo et Imote avec quelques petites variations. Le réseau Mrinfo est assez stable et ses communautés sont particulièrement marquées donc la méthode de Louvain était plus stable. La méthode stabilisée a une modularité très proche et est un peu plus stable. Concernant Radar, la méthode stabilisée n'est encore pas stable ce qui est sans doute lié au fait que c'est une mesure extrêmement dynamique. La bonne partition change donc pratiquement à chaque étape. Néanmoins, il y a quand même en général presque quatre fois moins de déplacements après stabilisation : 77% des nœuds sont déplacés en moyenne contre 21% pour la version stabilisée. Concernant Imote il est difficile de conclure vu le nombre très grand de pas de temps et les nombreuses petites modifications rendant le réseau instable. Nous obtenons une stabilité meilleure et une perte de qualité faible (la différence de modularité est de $0,017$ en moyenne avec un écart type de $0,027$). Il y a en moyenne seulement $0,28$ changements de communautés pour la méthode de Louvain modifiée et $1,31$ pour la méthode de Louvain (il

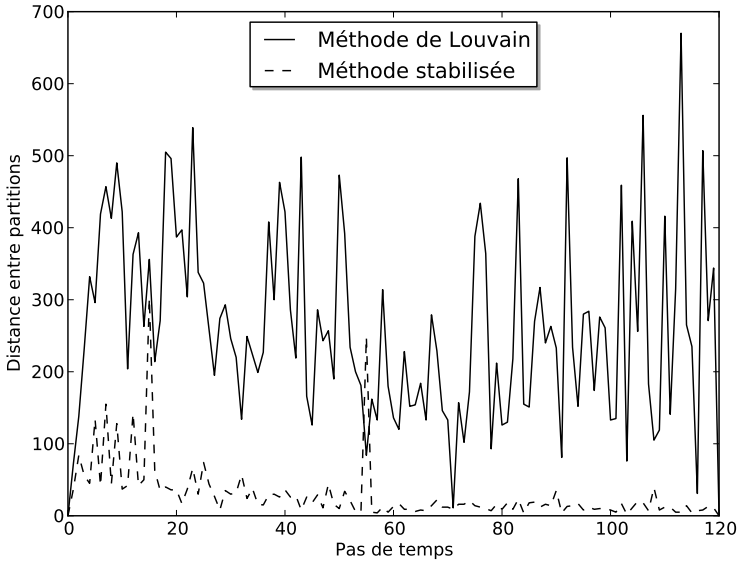


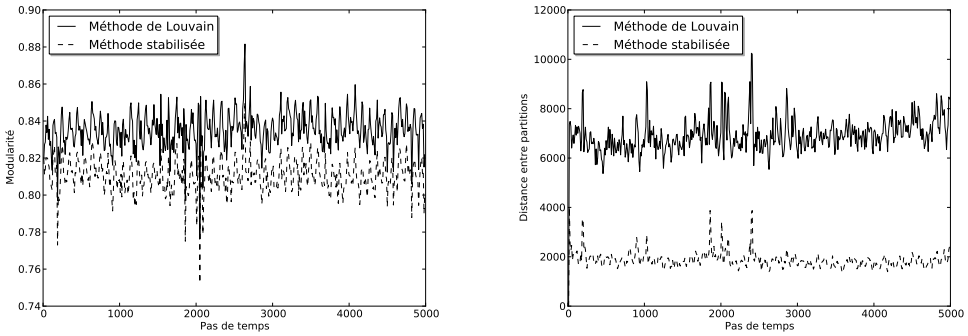
FIGURE 3.22 – Distance entre deux partitions successives de la méthode de Louvain modifiée sur le réseau Blogs.

y a énormément de pas de temps où il ne se passe pas grand chose et où la structure communautaire est très marquée entrainant une relativement bonne stabilité), par contre les écarts types sont bien plus grands pour la méthode de Louvain ($2,29$ contre $0,78$) indiquant de bien plus nombreux moments où les partitions changent.

Ainsi, en partant d'un réseau vide, la méthode de Louvain modifiée est stable, rapide et produit des résultats de grande qualité. Nous avons essayé de faire débuter l'algorithme plus tard, pour commencer directement par un réseau structuré et non un réseau vide, et les résultats se sont avérés très similaires. La méthode de Louvain modifiée semble donc toujours capable de s'adapter et conserve son gain de stabilité dans le cas du réseau Blogs.

3.8 Conclusions

Nous avons montré que l'application directe de méthodes statiques de détection de communautés était inappropriée dans le cas de graphes dynamiques. Ces méthodes sont généralement trop instables pour fournir un résultat utile. En effet, de très faibles variations du réseau d'entrée impliquent de grandes transformations de la sortie. Par conséquent, suivre les communautés au cours du temps à l'aide de ces méthodes est vain : on observe des modifications liées à l'algorithme et non des modifications de la structure du réseau. La distance



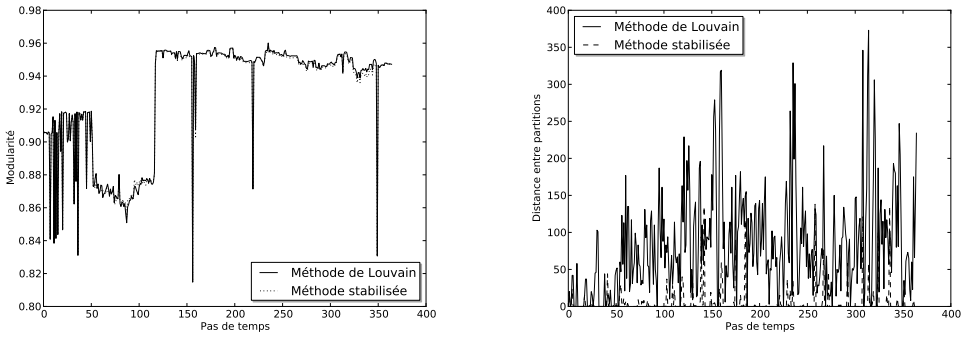
(a) Modularité (b) Distance

FIGURE 3.23 – Modularité et distance entre deux partitions successives de la méthode de Louvain modifiée sur le réseau Radar.

d'édition utilisée permet une mesure plus intuitive de cette instabilité que l'information mutuelle utilisée classiquement.

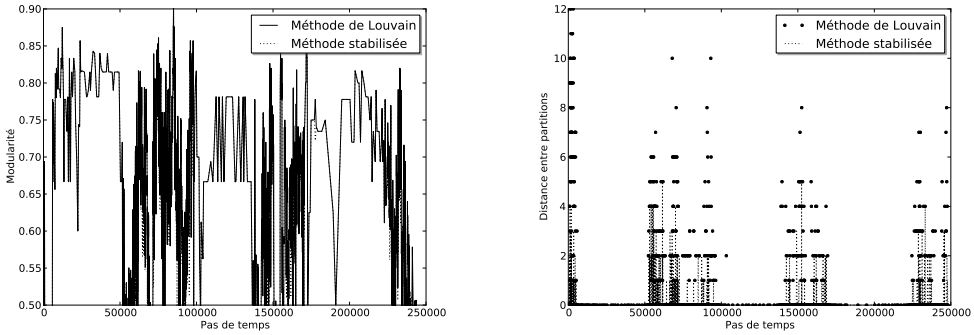
Nous avons proposé une modification de la méthode de Louvain qui permet de gagner énormément en stabilité au prix d'une faible perte de modularité. Cette modification est ajustable avec un paramètre de qualité permettant de contrôler finement le compromis stabilité/qualité. Si l'évolution est constituée d'une série de petites transformations, il faut accroître un peu la qualité au prix de la stabilité pour permettre une évolution continue de la partition. En revanche, si le réseau évolue suffisamment entre chaque pas de temps, on peut se permettre de régler la stabilité au maximum. De manière heuristique, il faut regarder la perte de qualité : on commence avec la stabilité au maximum et quand il y a une vraie divergence de qualité, il faut réduire la stabilité.

Les événements apparaissant ne sont pas encore parfaitement compris. Ils sont liés à la fois aux transformations du réseau et à la partition précédente et par conséquent la partition initiale a encore beaucoup d'importance. Différencier artefacts liés à l'algorithme et vraies transformations du graphe de terrain est encore un défi. Une limitation de cette approche vient de la méthode de détection : il n'y a pas de recouvrements de communautés. Cela implique qu'un nœud est dans une et une seule communauté. Or, on peut facilement imaginer dans le cas d'un réseau social que l'évolution de la structure n'est pas faite de ruptures nettes, mais au contraire d'une lente évolution : un nœud change petit à petit de communauté. Le chapitre suivant étudie une autre approche de la dynamique. Nous allons d'une part chercher des communautés bonnes sur une longue durée et chercher quels sont les regroupements de pas de temps cohérents structurellement.



(a) Modularité (b) Distance

FIGURE 3.24 – Modularité et distance entre deux partitions successives de la méthode de Louvain modifiée sur le réseau Mrinfo. Les modularités sont presque confondues (la différence moyenne est de 0.00066 avec une déviation de 0.0014).



(a) Modularité (b) Distance

FIGURE 3.25 – Modularité et distance entre deux partitions successives de la méthode de Louvain modifiée sur le réseau Imote. Les modularités sont ici aussi très proches (la différence moyenne est de 0.018 avec une déviation de 0.027).

Chapitre 4

Détection sur une période donnée

Sommaire

Nous proposons maintenant une méthode de détection de communautés dynamiques très différente de la précédente. Au lieu de chercher pour chaque pas de temps quelle est la bonne partition, nous allons au contraire chercher une seule partition, mais qui devra être bonne pendant plusieurs pas de temps. Ce ne seront plus des communautés instantanées, mais plutôt des partitions représentant la structure du réseau sur le long terme. Cela pose un certain nombre de problèmes. Tout d'abord, il faut définir ces communautés que nous appellerons communautés multi-pas, puis il faut des algorithmes pour effectivement les détecter. Il faut ensuite des outils pour étudier ces communautés car bien que la partition soit fixée, les nœuds et liens du réseau peuvent bouger. Enfin, une question importante est de savoir sur quelle durée on doit calculer les communautés. Si on calcule des communautés sur des petites durées, on revient au problème précédent tandis que si on cherche des communautés sur une durée trop longue, celles-ci pourraient ne pas exister ou ne pas être significatives.

4.1 Définition de communautés multi-pas

Nous considérons ici encore qu'un graphe dynamique est une succession de graphes statiques représentant chacun l'état du réseau à un moment donné. On notera $S = \{1, 2, ..., n\}$ l'ensemble des pas de temps et $G = \{G_1, G_2, ..., G_n\}$ le graphe dynamique avec $G_i = (V_i, E_i)$ l'instantané i (ou au pas de temps i) ayant pour nœuds V_i et pour arêtes E_i. On notera aussi $V = \cup_{i\in\{1,...,n\}} V_i$ l'ensemble de tous les nœuds et on cherchera une partition de V. Une fenêtre de temps T est un sous ensemble de S. Nous discuterons ceci plus tard et pour le moment nous aurons toujours $T = S$. Nous noterons $Q(G_i, \pi)$ la modularité de la partition π sur le graphe statique G_i en restreignant π à une partition de V_i et on l'appellera la modularité statique pour ne pas la confondre avec la fonction de qualité multi-pas que nous cherchons à définir, à savoir la modularité moyenne :

Définition 8 *La modularité moyenne, $Q_{avg}(G, \pi, T)$, de la partition π de V durant la fenêtre de temps T est définie comme :*

$$Q_{avg}(G, \pi, T) = \frac{1}{|T|} \sum_{i\in T} Q(G_i, \pi)$$

Dans de nombreuses situations, les pas de temps n'ont pas tous la même importance et il est possible d'assigner un poids w_i au pas de temps i. Cela permet par exemple de considérer le cas où les pas de temps ne sont pas régulièrement espacés en tenant compte de l'espacement dans les poids. Il est aussi possible de donner plus d'importance aux pas de temps récents en utilisant des poids comme $w_i = \frac{i}{n}$.

Définition 9 *La modularité moyenne pondérée de la partition π de V durant la fenêtre de temps T est définie comme :*

$$Q_{avg}(G, \pi, T) = \frac{1}{\sum_{i\in T} w_i} \sum_{i\in T} w_i.Q(G_i, \pi)$$

Détecter des communautés revient alors à trouver la partition maximisant la modularité moyenne. Tout comme l'optimisation de la modularité statique [14], optimiser la modularité moyenne est NP-complet et nous allons donc en fait chercher des partitions ayant la plus haute modularité moyenne possible.

4.2 Méthodes de détection

Nous proposons ici deux algorithmes pour trouver des partitions ayant une haute modularité moyenne sur une fenêtre de temps donnée. La première

consiste à construire un nouveau graphe qui sera une représentation moyenne du réseau dynamique puis à détecter des communautés sur ce nouveau graphe et les considérer comme la structure moyenne. La deuxième méthode est une modification de la méthode de Louvain pour optimiser directement la modularité moyenne.

4.2.1 Méthode-somme

Plutôt que d'optimiser directement la modularité moyenne, nous proposons premièrement la *méthode-somme* pour optimiser la modularité statique sur une représentation moyenne du réseau dynamique. Étant donné un graphe dynamique $G = \{G_1, G_2, ..., G_n\}$ et une fenêtre de temps $T \subseteq \{1, ..., n\}$, nous construisons un nouveau graphe pondéré, que nous appelons le graphe somme, qui est l'union de tous les instantanés de T : chaque arête dans le graphe somme a pour poids le nombre total de pas de temps de T durant lesquels cette arête existe. Puisque le graphe somme est un graphe pondéré statique classique, on peut appliquer la méthode de Louvain dessus (ou n'importe quel autre algorithme) et considérer son résultat comme un ensemble de communautés multi-pas.

Cependant, la métrique optimisée de cette façon est un peu différente de la modularité moyenne. Étant donnée une partition π, en notant l_{tc} le nombre de liens à l'intérieur de la communauté c au temps t, L_t le nombre de liens de G_t et d_{tc} le degré total de la communauté c au temps t. Le poids total des liens à l'intérieur de c est alors $\sum_{t\in T} l_{tc}$, le poids total du graphe somme est $\sum_{t\in T} L_t$ et le degré pondéré total de la communauté c est $\sum_{t\in T} d_{tc}$. Alors, la modularité statique de la partition π sur le graphe somme est égale à :

$$Q_{somme}(\pi, G, T) = \sum_{c\in\Pi} \frac{\sum_{t\in T} l_{tc}}{\sum_{t\in T} L_t} - \left(\frac{\sum_{t\in T} d_{tc}}{2.\sum_{t\in T} L_t}\right)^2$$

Si nous revenons à la définition de la modularité moyenne, et après réordonnancement de la somme, nous avons :

$$Q_{avg}(\pi, G, T) = \frac{1}{\sum_{i\in T} w_i} \sum_{c\in\Pi} \left(\sum_{t\in T} \frac{l_{tc}}{L_t} - \sum_{t\in T} \left(\frac{d_{tc}}{2.L_t}\right)^2\right)$$

Par conséquent, la modularité statique sur le graphe somme Q_{somme} n'est pas exactement la modularité moyenne Q_{avg}. L'ordre entre certaines sommations et fractions est inversé. Nous allons maintenant proposer une seconde méthode pour optimiser la modularité moyenne directement. Néanmoins, cette première méthode est très rapide car les graphes sommes sont en général bien plus petits que la somme des tailles des instantanés. Nous verrons par la suite que cette méthode est une bonne première approximation puisque les résultats obtenus ont une haute modularité moyenne.

4.2.2 Modification de la méthode de Louvain

Pour optimiser la modularité moyenne, nous allons modifier la méthode de Louvain. Deux raisons expliquent l'efficacité de la méthode de Louvain. Premièrement, quand il faut décider dans quelle communauté un nœud doit être déplacé, il est possible de calculer très rapidement le gain de modularité statique induit par chaque déplacement. Deuxièmement, la taille du graphe étudié est rapidement réduite durant la phase 2 et par conséquent les passes suivantes sont très rapides. Il faut donc changer deux éléments pour adapter la méthode de Louvain pour optimiser la modularité moyenne durant une fenêtre de temps T : le calcul du gain de qualité dans la première phase et la construction du réseau entre communautés durant la seconde phase.

Il faut tout d'abord remarquer que la différence de modularité moyenne entre deux partitions est la somme des différences de modularité statique sur chaque instantané. Par conséquent, le gain de modularité moyenne peut aussi être calculé localement et rapidement : c'est la somme des gains statiques sur chaque instantané de T. La transformation du réseau en un réseau entre communautés peut aussi être modifiée : il suffit en fait de transformer chaque pas de temps de T suivant la même procédure que la méthode de Louvain indépendamment, en prenant juste garde à nommer les nœuds représentant la même communauté à des pas de temps similaires de la même façon et en calculant les poids indépendamment pour chaque pas de temps. On obtient ainsi un réseau dynamique entre les communautés. Nous appellerons cette méthode la méthode-moyenne.

Exécuter la méthode-moyenne n'est pas en général plus long que d'appliquer la méthode de Louvain sur chaque pas de temps de T car moins de nœuds sont considérés durant la phase 1. La complexité de la méthode de Louvain est assez difficile à analyser. Dans le pire des cas, c'est un algorithme assez lent en $O(L^3)$ avec L le nombre de liens. Dans la pratique, il a un comportement quasi-linéaire sur les graphes possédant une structure communautaire. La méthode-moyenne a la même complexité et est donc parfois plus lente que la méthode de Louvain appliquée sur chaque pas de temps (quand la structure moyenne n'est pas claire et que la première phase dure longtemps) et souvent bien plus rapide. Par conséquent, cela reste applicable à de grands jeux de données composés de milliers de nœuds dans des milliers de pas de temps. Une implémentation est disponible sur la page [37]. Nous allons maintenant analyser les résultats de ces deux algorithmes.

4.3 Modularité moyenne des différents réseaux

Pour étudier l'efficacité des algorithmes, nous avons comparé les qualités de plusieurs partitions en utilisant toute la durée de la mesure comme fenêtre de temps. T est donc $\{1, ..., n\}$. La s-partition est le résultat de méthode-

somme (voir section 4.2.1) et la a-partition le résultat de la méthode-moyenne (voir section 4.2.2). Nous avons aussi calculé les partitions trouvées par la méthode de Louvain pour chacun des instantanés pris indépendamment les uns des autres. Ce sont les partitions statiques. Comme elles ont la meilleure modularité statique que l'on peut espérer sur leurs instantanés respectifs, leur moyenne est un maximum, sans doute inatteignable, pour la modularité moyenne. On compare ici la qualité d'une partition unique (s-partition ou a-partition) avec la moyenne des qualités de plusieurs partitions ayant chacune moins de contraintes.

	s-partition	a-partition	moyenne des modularités statiques
Blogs	0.6	0.61	0.62
Mrinfo	0.91	0.92	0.923
Imote	0.38	0.39	0.56
Radar	0.70	0.70	0.84

TABLE 4.1 – Modularités moyennes des partitions étudiées durant la totalité de la mesure.

Le tableau 4.1 présente les modularités obtenues pour chacune des trois méthodes. Bien que la a-partition et la s-partition aient des qualités très similaires, la a-partition est toujours meilleure. Comme la s-partition est nettement plus rapide à calculer, elle reste intéressante, mais par la suite nous allons toujours utiliser les a-partitions et nous allons les appeler les partitions multi-pas. Pour information, les temps pris pour optimiser la modularité moyenne sont donnés dans le tableau 4.2. La méthode-somme est pratiquement instantanée sur les graphes considérés, une fois le graphe moyen construit.

Blogs	Imote	Mrinfo	Radar
30 s	50 s	4 minutes	4 heures 30 minutes

TABLE 4.2 – Temps d'exécution de la méthode de Louvain pour optimiser la modularité moyenne.

4.4 Une dynamique interne

4.4.1 Modularités statiques de partitions multi-pas

Pour obtenir des informations plus précises à propos des qualités des partitions, les figures 4.1, 4.2, 4.3 et 4.4 présentent la modularité statique de ces partitions au cours du temps. Nous avons donc calculé d'abord la partition

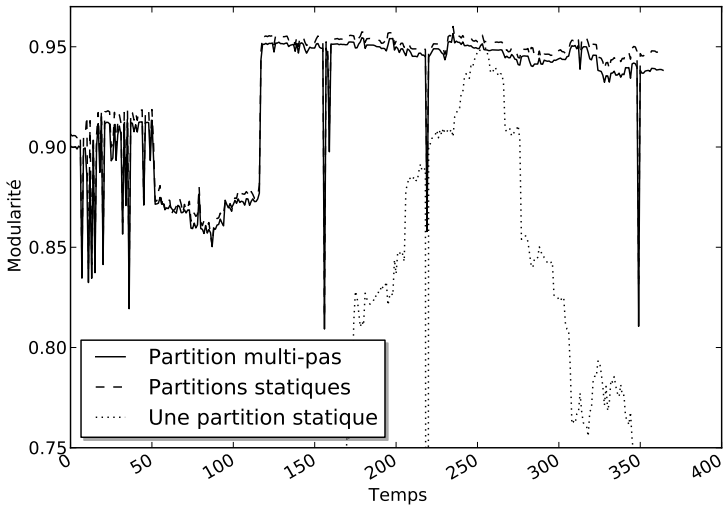


FIGURE 4.1 – Modularités statiques des partitions étudiées sur Mrinfo.

multi-pas et ensuite, pour chaque pas de temps, nous avons regardé la modularité de la partition multi-pas et celle de la meilleure partition possible obtenue par la méthode de Louvain. Nous avons aussi ajouté la meilleure partition statique. Il s'agit, parmi toutes les partitions statiques, de celle qui a la meilleure modularité moyenne quand on la considère comme une partition multi-pas sur toute la durée. L'objectif est d'étudier si on aurait pu trouver une bonne structure globale en considérant simplement un instantané. Comme tous les nœuds n'étaient pas forcément présents sur le pas de temps où elle a été calculée, nous avons regroupé tous les nœuds non classés dans une communauté unique. C'est un désavantage important, mais d'autres solutions comme mettre chaque nœud seul dans sa communauté auraient été encore plus désavantageuses.

Pour information, les modularités moyennes des ces meilleures partitions statiques sont 0.6 pour Blogs, 0.61 pour Mrinfo et 0.34 pour Imote. Excepté avec Blogs, les meilleures partitions statiques sont donc de faible qualité comparées aux s-partitions et aux a-partitions.

Dans le réseau Blogs, la partition multi-pas et les partitions statiques se rapprochent régulièrement, confirmant que les structures instantanées et moyennes sont de plus en plus proches. De même, la meilleure partition statique est très proche des autres approches. Ceci peut s'expliquer par la façon dont est construit le réseau : il croît régulièrement et par conséquent les derniers instantanés sont très similaires au graphe somme défini précédemment.

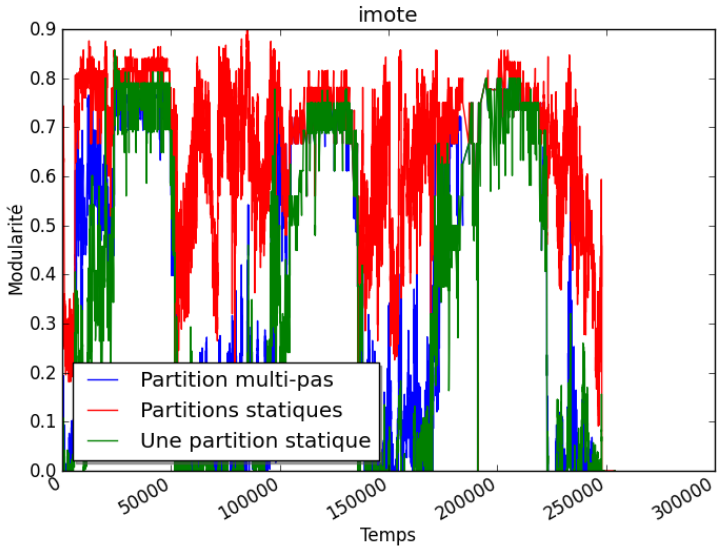


FIGURE 4.2 – Modularités statiques des partitions étudiées sur Imote.

Leurs décompositions ont donc une grande modularité moyenne.

Avec le réseau Mrinfo, la partition multi-pas a pratiquement toujours une qualité comparable à la partition statique. Ainsi, on capture la structure globale du graphe avec seulement une partition. Cela continue à fonctionner même durant la phase 2 entre les jours 52 et 117. Il semble que des parties de l'ensemble des nœuds change à ce moment et par conséquent cet événement n'affecte pas les partitions des autres nœuds. La meilleure partition statique ici échoue clairement à décrire une structure globale car elle n'a une bonne modularité que durant quelques pas de temps. Cela démontre l'intérêt de chercher une partition multi-pas intéressante sur toute la mesure plutôt que juste sélectionner une partition d'un pas de temps qui n'est pas représentative de la globalité.

Dans le réseau Imote, il apparaît que la structure change énormément au court du temps, et il y a un effet jour/nuit très prononcé. La partition multi-pas est de très bonne qualité durant la nuit, car le graphe est quasi statique et les groupes clairement séparés à ce moment-là. Inversement, elle échoue durant la journée, car la structure est beaucoup moins prononcée et très instable. On constate d'ailleurs que la meilleure partition statique souffre du même défaut et a une moins bonne modularité moyenne.

Dans le réseau Radar, nous n'arrivons pas à trouver une partition multi-pas qui ait une qualité similaire aux partitions statiques. Étant donné que Radar évolue rapidement, il est possible que cette structure générale n'existe

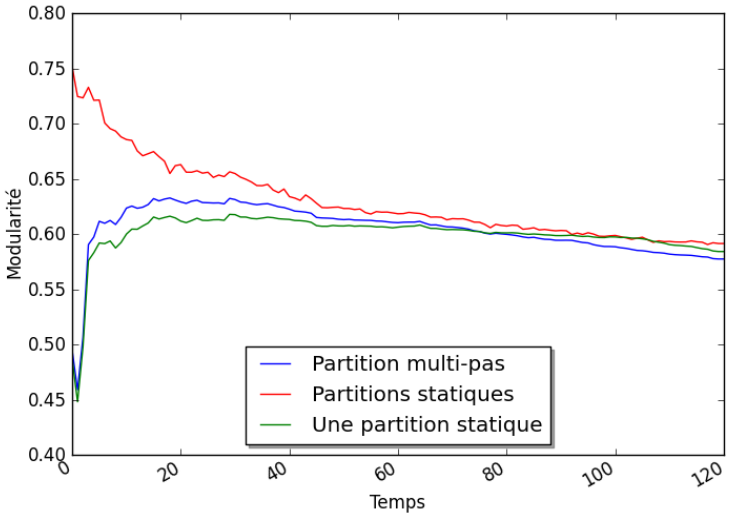


FIGURE 4.3 – Modularités statiques des partitions étudiées sur Blogs.

pas, mais nous ne pouvons pas savoir avec certitude si cette structure existe et nous échouons à la trouver ou si cette structure n'existe pas. On peut remarquer que la partition statique prise sur l'ensemble de la durée est aussi de très faible qualité.

Finalement, il semble que l'efficacité de la modularité moyenne dépend énormément des caractéristiques du graphe. Avec Mrinfo et Imote, la partition multi-pas donne de nouvelles informations car il n'y a pas de pas de temps représentatif de toute la vie du réseau tandis que pour Blogs, détecter la structure à la fin est suffisant. Avec Radar, la partition multi-pas ne semble pas rendre compte efficacement de la structure globale qui peut ne pas exister.

4.4.2 Visualisation de la dynamique interne

Un éclairage intéressant est l'existence ou non de nœuds durant une fenêtre de temps. La partition est unique mais le graphe lui évolue. Nœuds et liens apparaissent et disparaissent et bien que nous n'ayons qu'une seule partition englobant tous les nœuds qui ont existé à un moment, le contenu des communautés peut évoluer. Certains nœuds peuvent même ne jamais exister au même moment, mais être fortement connectés à un noyau dur d'une communauté. Pour représenter ceci, nous avons choisi une représentation pour chaque communauté pour laquelle chaque ligne correspond à un nœud et chaque colonne à un pas de temps (voir figure 4.5). Si le n^{ieme} nœud de la communauté

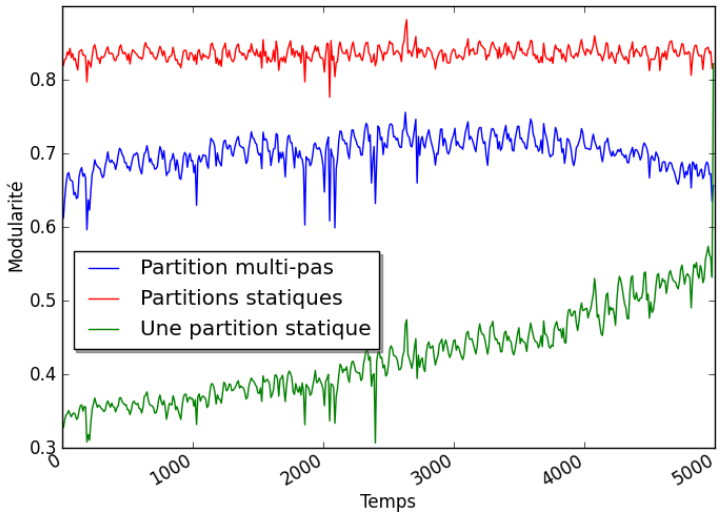


FIGURE 4.4 – Modularités statiques des partitions étudiées sur Radar.

existe au temps t, le point (t, n) est blanc et inversement le point est noir s'il n'existe pas. Pour avoir un résultat interprétable et donc pas trop bruité, nous devons choisir un bon ordre pour les nœuds. Nous avons choisi un ordre simple qui donne quand même de bons résultats : les nœuds sont d'abord ordonnés en fonction de leur moment d'apparition, car des nœuds qui apparaissent ensemble sont souvent similaires puis nous les avons triés selon le nombre d'instantanés pendant lesquels ils existent (cela permet de grouper raisonnablement les nœuds qui participent aux mêmes événements).

Cette représentation graphique n'est pas intéressante avec tous les graphes ni toutes les communautés, car il est parfois impossible de représenter tous les nœuds ou tous les pas de temps. De plus les partitions ne sont pas toutes intéressantes et essayer de les représenter est alors inutile. Avec le réseau Blogs, les communautés croissent toutes lentement et il n'y a rien de plus intéressant à voir. Chacune des courbes ressemble à un triangle où les nœuds apparaissent un à un. Il est tout de même remarquable que toutes les communautés grandissent lentement, il n'y a pas par exemple une séparation entre les communautés apparaissant tôt et les communautés apparaissant tard : toutes les communautés commencent vides et croissent régulièrement et à des rythmes comparables. Pour le réseau Imote, il y a bien trop de pas de temps et trop peu de nœuds pour observer quelque chose, d'autant plus que la partition n'est pas bonne pendant toute une partie du temps, et notamment la journée qui est plus intéressante. On obtient les résultats les plus intéressants avec Mrinfo.

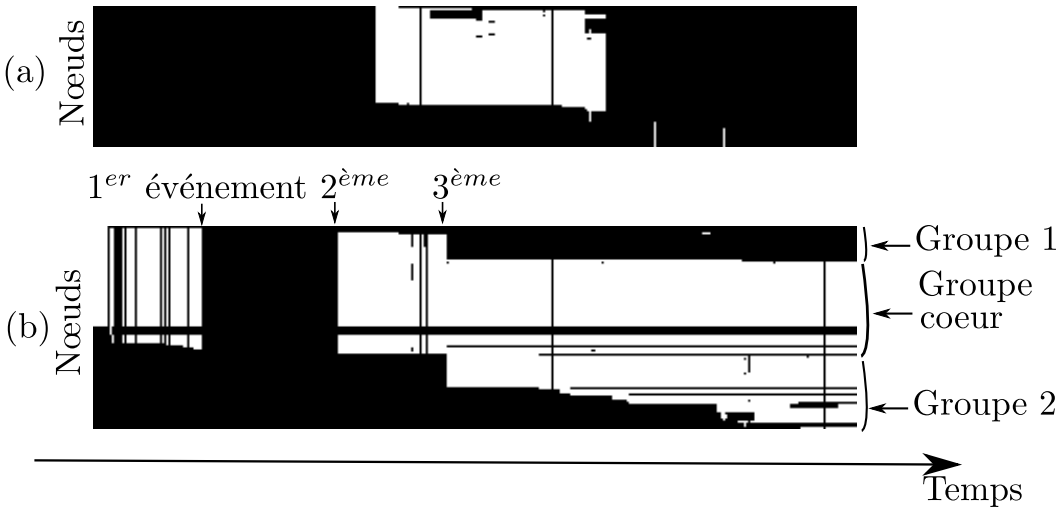


FIGURE 4.5 – Diagrammes d'existence des nœuds de deux communautés de Mrinfo.

La communauté (a) de la figure 4.5 illustre le cas le plus simple : tous les nœuds existent approximativement au même moment, à part quelques exceptions. Les nœuds apparaissent et disparaissent en même temps, ce qui semble révéler que le regroupement est pertinent. La communauté (b) correspond à un cas plus compliqué et on peut la diviser en 3 grands groupes : un noyau central composé des nœuds les plus présents et deux groupes qui existent à des moments différents. Ainsi, au début, le groupe 1 et le noyau central existent puis disparaissent tout deux durant la deuxième phase entre les jours 52 et 117 puis réapparaissent pendant quelques jours. Ensuite, le groupe 1 disparaît et le groupe 2 le remplace.

De tels diagrammes illustrent que même si la partition est unique, les nœuds et les liens peuvent exister ou disparaître durant la mesure et que par conséquent les communautés évoluent. Cela révèle de nouvelles informations à propos de la structure des communautés et étudier localement à l'intérieur d'une communauté permet d'observer des évolutions.

4.5 Détection automatique des fenêtres de temps

4.5.1 Définition des fenêtres de temps

Jusqu'à ce point, nous avons optimisé la modularité moyenne seulement sur toute la période des expériences et nous avons vu que dans certains cas on obtient des communautés de moins bonne qualité car la structure moyenne

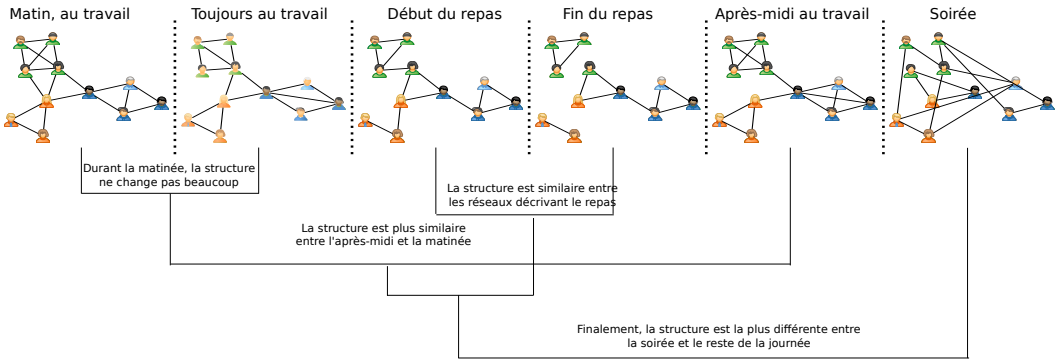


FIGURE 4.6 – Un exemple naïf de décomposition en pas de temps. On a représenté plusieurs pas de temps et en dessous l'arbre de regroupement des pas de temps. On considère un réseau social de qui communique avec qui au cours du temps. Durant la matinée, les gens sont avec leurs équipes et les pas de temps correspondant sont regroupés. Pendant le repas la structure change et les pas de temps correspondant au repas sont regroupés. Durant l'après-midi, les gens retournent au travail et donc la structure ressemble à la matinée. Ensuite, le repas et les moments "au travail" se ressemblent plus que la soirée où les gens sont entre amis ou en famille.

change trop. Nous allons maintenant exploiter le fait que nous pouvons choisir la fenêtre de temps T. Elle peut être un intervalle si nous ne voulons que des pas de temps consécutifs mais ce n'est pas obligatoire. Par exemple, on peut vouloir retrouver une structure qui se répète chaque jour et ne serait pas composée de pas de temps contigus. Une fenêtre de temps est donc un sous-ensemble de l'ensemble de pas de temps.

Dans Imote, la structure change entre les jours et la nuit, mais elle change aussi durant la journée en fonction des différentes sessions, repas, keynote, etc. Ainsi, une fenêtre de temps doit naturellement pouvoir en contenir des plus petites. Ce genre de segmentation hiérarchique est habituellement représentée par un arbre dont les feuilles sont les constituants de base (ici les pas de temps) et dont chaque nœud correspond au regroupement des deux sous-arbres fils. Un exemple (hypothétique) de décomposition est donné sur la figure 4.6.

4.5.2 Détection automatique

4.5.2.1 Algorithme

Pour construire un tel arbre, nous proposons un algorithme de regroupement hiérarchique des fenêtres de temps qui consiste à regrouper à chaque étape les deux fenêtres de temps les plus similaires : au début, chaque pas de

temps est tout seul et on regroupe récursivement les deux fenêtres les plus similaires jusqu'à ce qu'il ne reste aucune possibilité. Cet algorithme très standard nécessite une fonction de similarité pour décider quelles fenêtres regrouper.

Algorithme 2 Algorithme de regroupement hiérarchique de fenêtres de temps

```
1: G le graphe initial
2: L la liste des pas de temps potentiels, initialement vide
3: pour tout instantané t de G faire
4:    Chercher les communautés π_{t} sur l'instantané t
5:    Mettre {t} dans L
6: fin pour
7: tantque L n'est pas vide faire
8:    Trouver T_i et T_j dans L qui maximise Sim(T_i, T_j)
9:    Enlever T_i et T_j de L et ajouter T = T_i ∪ T_j dans L
10:   Calculer les communautés π_T de G pour la fenêtre T
11: fin tantque
```

4.5.2.2 Fonction de similarité

Pour définir une similarité entre fenêtres de temps, nous allons utiliser les partitions associées. Nous utilisons comme hypothèse que si deux fenêtres de temps sont structurellement similaires, la partition multi-pas de l'une, qui résume sa structure, sera aussi une relativement bonne décomposition pour l'autre et inversement. Cela peut être évalué pour deux fenêtres T_i et T_j par la fonction de similarité définie comme suit :

Définition 10 *Étant données deux fenêtres de temps T_i et T_j ainsi que leur décomposition en communautés π_i et π_j, on définit la similarité entre T_i et T_j comme :*

$$Sim(T_i, T_j) = Q_{avg}(G, \pi_i, T_j) + Q_{avg}(G, \pi_j, T_i)$$

On n'effectue pas une comparaison des partitions à cause de l'instabilité des algorithmes de décomposition comme discuté précédemment [6]. Inversement, la modularité est stable et par conséquent nous l'avons utilisée comme test d'efficacité d'une partition pour dire si deux fenêtres de temps ont des structures similaires.

Quelques extensions de l'algorithme sont possibles. Par exemple, on peut vouloir forcer les fenêtres regroupées à être contiguës : dans ce cas, les résultats sont plus simples à interpréter, mais des structures répétées ne seront pas détectées. Par la suite, nous allons aussi interdire de regrouper des fenêtres

de temps dont la similarité est négative [1]. En utilisant ces règles, l'algorithme produit une forêt d'arbres disjoints plutôt qu'un seul arbre.

4.5.3 Filtrage

Les arbres obtenus par cette méthode sont des arbres binaires ayant un nombre de feuilles égal au nombre de pas de temps. Cela produit donc des arbres de grandes tailles et difficiles à analyser tels quels. Ils ont aussi la particularité de contenir de longs peignes correspondants aux ajouts successifs de pas de temps dans une fenêtre. En effet, vu que l'arbre est binaire, chaque pas de temps est ajouté un par un. Ces longs peignes ne nous semblent pas très intéressants, les points pertinents sont plus les moments où deux fenêtres de temps sont regroupées. Pour obtenir plus directement un arbre exploitable, nous avons développé une technique de filtrage réduisant la taille de l'arbre. Elle consiste à d'abord enlever toutes les feuilles, car elles apportent peu d'information, puis à réduire tous les chemins triviaux (les successions de nœuds de l'arbre n'ayant qu'un parent et qu'un successeur) en leurs extrémités (voir figure 4.7).

Nous allons aussi noter une fenêtre de temps de manière concise en regroupant les intervalles. Par exemple, "32,34,45,[52-86],[88-116]" correspond à la fenêtre de temps contenant les pas de temps 32, 34, 45, de 52 à 86 et de 88 à 116. Quand les pas de temps ne sont pas régulièrement espacés comme pour Imote nous afficherons les durées réelles plutôt que les indices des pas de temps.

4.5.4 Résultats sur les réseaux.

La validation des résultats est une question importante et nous la faisons encore manuellement. Nous avons vérifié que les fenêtres de temps identifiées par l'algorithme correspondent à des phénomènes compréhensibles sur le graphe obtenus soit par l'intermédiaire de connaissances externes sur le réseau (une idée du déroulement de la conférence sur Imote par exemple) soit par l'intermédiaire de visualisations du réseau. Cela permet de valider à la fois l'algorithme et la fonction de similarité. Nous avons construit les arbres pour chaque jeu de données sans et avec l'obligation de regrouper des fenêtres contiguës. Il faut remarquer que si on oblige les fenêtres de temps à être contiguës, le moindre problème de mesure ou un grand changement soudain de structure pendant un court laps de temps empêchent les fenêtres d'avant et d'après de se regrouper et implique une coupure en deux arbres disjoints.

1. la décomposition triviale qui regroupe tous les nœuds dans une seule communauté a une modularité nulle et par conséquent une modularité négative est signe d'une décomposition vraiment mauvaise !

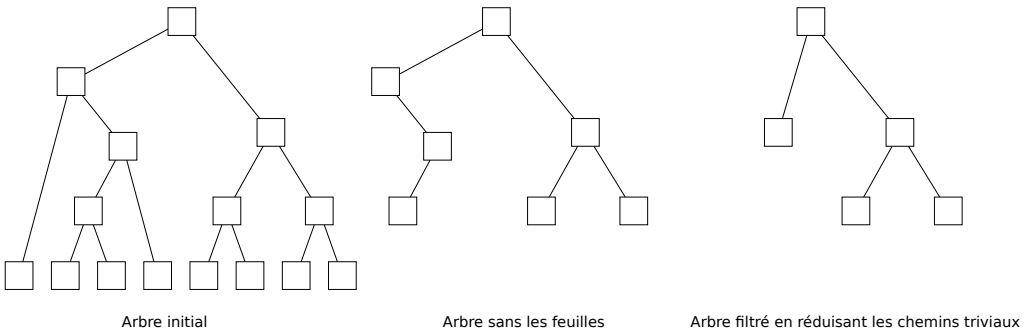


FIGURE 4.7 – Schéma du filtrage de l'arbre : on enlève les feuilles puis on réduit les chemins (branche de gauche par exemple). Ce filtrage peut être itéré si l'arbre reste trop grand.

Avec Mrinfo en utilisant des fenêtres contiguës (figure 4.9), le niveau le plus élevé qui contient les fenêtres de temps les plus longues est plus divisé que les 3 phases décrites précédemment. En effet, il y a de nombreux événements pendant la première phase qui ne peut par conséquent pas être identifiée comme une grande fenêtre. Le groupe de pas de temps entre le jour 52 et le jour 116 correspond à la phase 2 et les fenêtres de temps après correspondent à la phase 3 entrecoupée de 3 événements. Le premier niveau semble donc bien correspondre à l'impression que l'on avait de la dynamique et il est aussi possible d'expliquer les séparations aux niveaux inférieurs correspondant à des disparitions et des apparitions de gros blocs dans le réseau.

Quand on regarde la décomposition avec des fenêtres non contiguës (figure 4.8), l'effet causé par les événements ponctuels disparaît. Le premier niveau est composé de deux groupes : un contenant la phase 2 et quelques événements et un contenant les phases 1 et 3, qui sont structurellement similaires. Ce dernier groupe est ensuite divisé en deux sous-groupes contenant la phase 1 et la phase 3. Ensuite, les sous niveaux correspondent à des fenêtres contiguës, séparées par les événements. On peut remarquer que bien qu'on n'impose aucune contrainte de contiguïté, les fenêtres contiennent des pas de temps très rapprochés et sont quasi contiguës. Il est en effet raisonnable de penser que deux instantanés proches dans le temps ont plus de chance d'être similaires structurellement.

Nous obtenons des résultats similaires avec Imote. En utilisant des fenêtres de temps contiguës, le niveau le plus haut contient plusieurs fenêtres qui correspondent à de vrais moments (voir figure 4.9). Le premier correspond au début de l'expérience et contient trois groupes : les 45 minutes après le début, une longue période correspondant à la première nuit et deux autres heures et

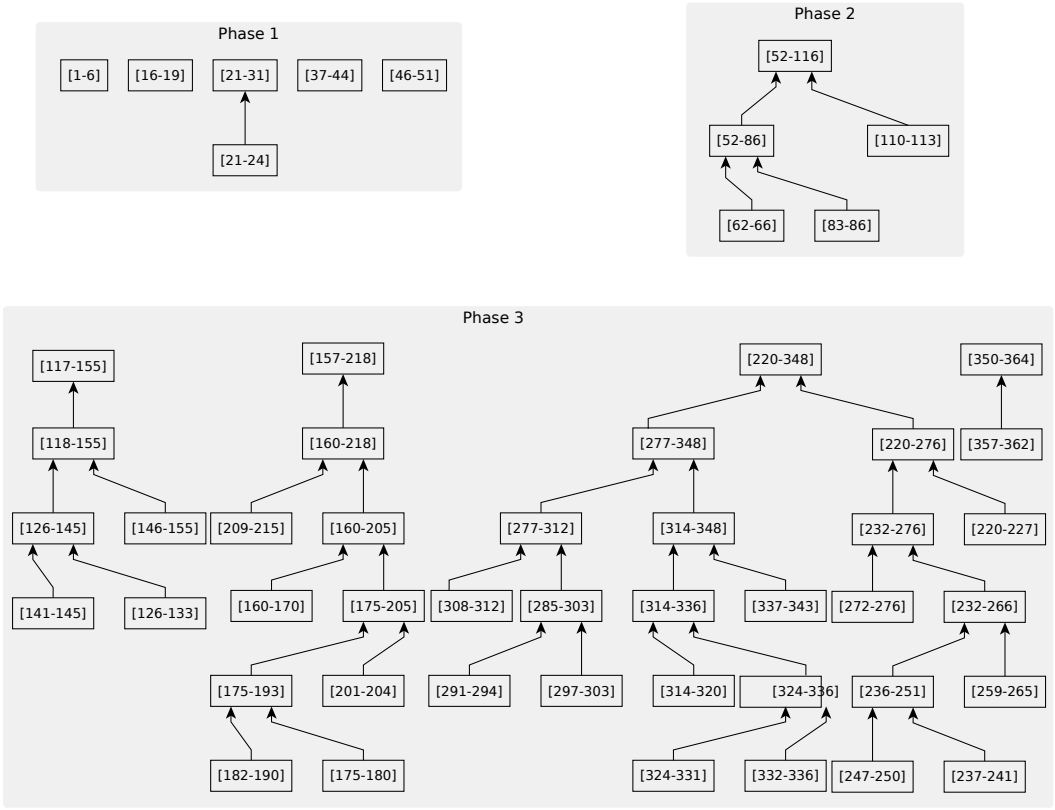


FIGURE 4.8 – Arbre obtenu après filtrage (répété deux fois car l'arbre était encore trop grand) avec Mrinfo en forçant les fenêtres à être connexes. Les nombres dans les nœuds de l'arbre représentent les fenêtres de temps. Nous avons groupé les fenêtres de temps en blocs avec une explication potentielle.

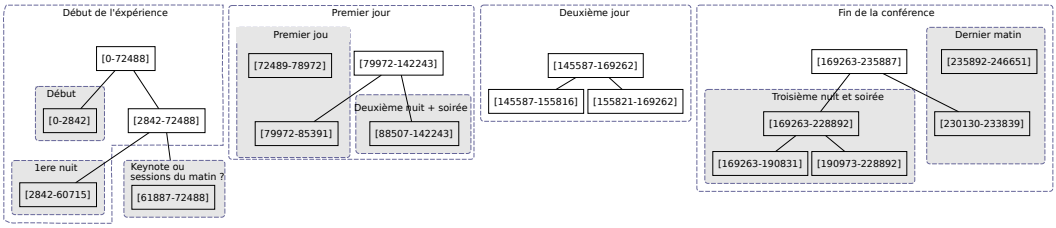


FIGURE 4.9 – Premiers niveaux de la décomposition hiérarchique du temps sur Imote. Les temps sont en secondes depuis le début de l'expérience. Même après filtrage l'arbre reste trop grand pour un affichage sur une page, nous représentons donc ici que les premiers niveaux après un filtrage manuel.

demie qui pourraient correspondre au petit déjeuner et au premier keynote (nous n'avons pas la correspondance exacte entre les temps de l'expérience et de la conférence et ne pouvons qu'émettre des hypothèses). Ensuite, il y a une petite fenêtre qui peut correspondre à une partie du premier jour. La

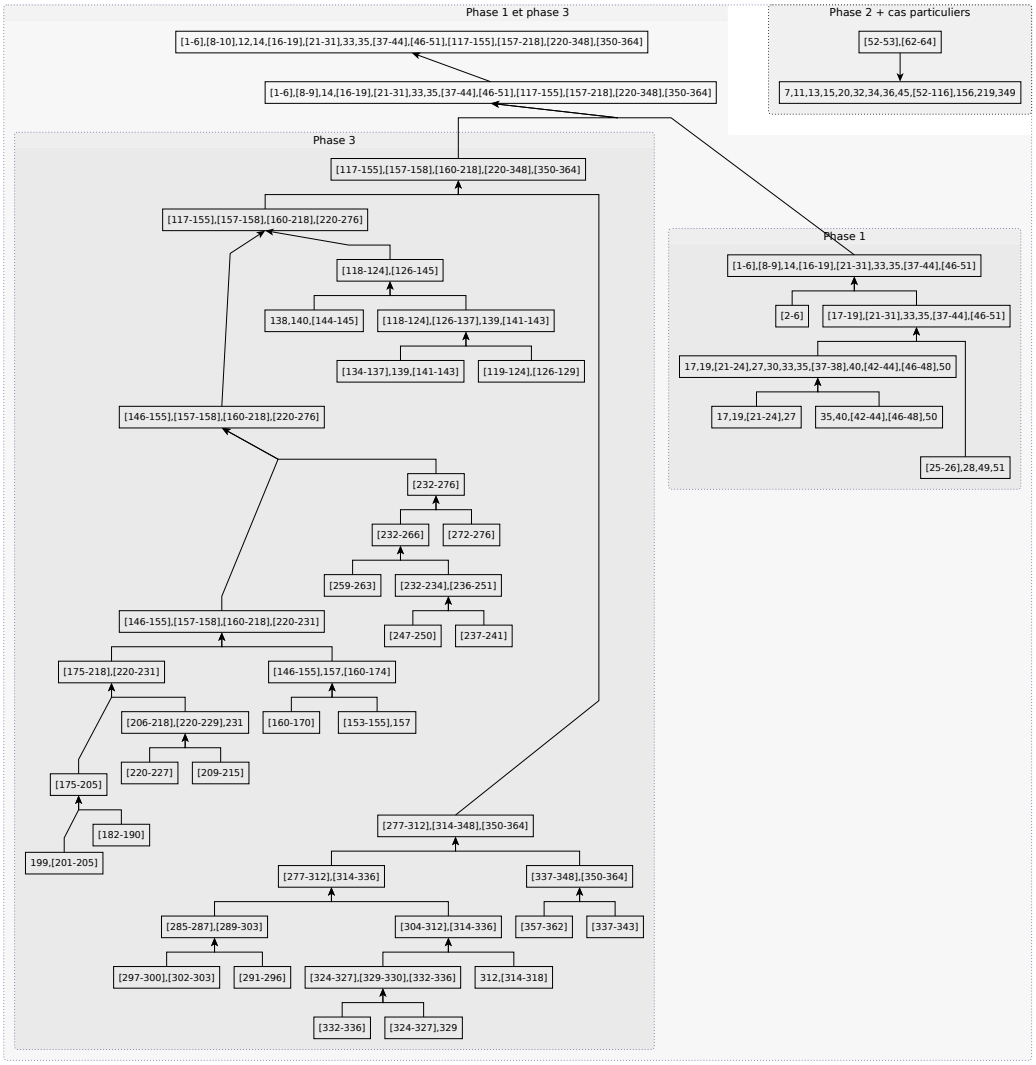


FIGURE 4.10 – Arbre obtenu avec Mrinfo après filtrage (répété deux fois). Les nombres dans les nœuds de l'arbre représentent les fenêtres de temps. Nous avons groupé les fenêtres de temps en blocs avec une explication potentielle.

fenêtre suivante est la suite du premier jour avec la seconde nuit. La fenêtre suivante contient majoritairement le second jour, qui est lui-même divisé en deux parties qui correspondent pratiquement avec le matin et l'après-midi. La fenêtre suivante est composée de la dernière nuit et de la dernière matinée et enfin la dernière fenêtre contient les quelques moments restants de la dernière matinée, correspondants potentiellement au retour des périphériques et la fin de l'expérience.

Nous donnons sur la figure 4.11 une représentation simplifiée du plus grand arbre (les autres sont très petits) de la décomposition obtenue sans

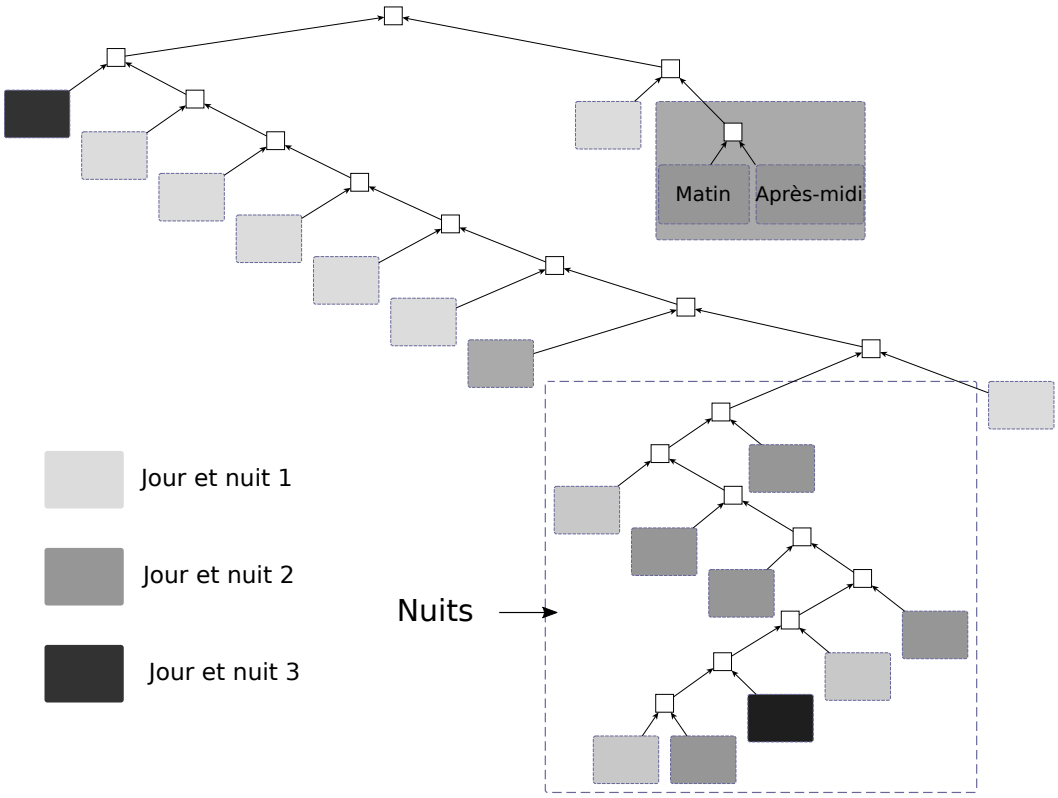


FIGURE 4.11 – Plus grand arbre de la décomposition hiérarchique du temps de Imote avec des fenêtres de temps non contraintes à être contiguës. Les blocs représentent de grands sous arbres afin de représenter la structure globale.

la contrainte de contiguïté. L'arbre est vraiment grand et il y a énormément de sous niveaux, nous ne pouvons donc pas le représenter en entier mais juste donner une idée de sa structure. Ainsi, chaque bloc de la figure 4.11 correspond en fait à un sous arbre. On constate que les sous arbres sont tous constitués de pas de temps appartenant à la même journée, ce qui tend à montrer que la structure identifiée a du sens. Les deux plus grands groupes sont le jour 2 et les nuits. Le jour 2 est lui décomposé en deux parties, matin et après-midi, tandis que le groupe contenant les nuits contient plusieurs sous arbres contenant des pas de temps du même jour. Les données changeant souvent de structure et étant très bruitées, retrouver ces informations dans la structure semble très positif.

Avec le jeu de données de Blogs, les arbres de fenêtres contiguës et non contiguës sont très proches. Ils sont tous deux très déséquilibrés et presque dégénérés en une sorte de liste de pas de temps, s'ajoutant un par un pour former un grand peigne. Le filtrage réduit beaucoup la taille de l'arbre. Ceci est cohérent avec le fait que le réseau ne fait que croître : il n'y a pas ici de fenêtres de temps clairement identifiées. C'est pourquoi nous n'avons que

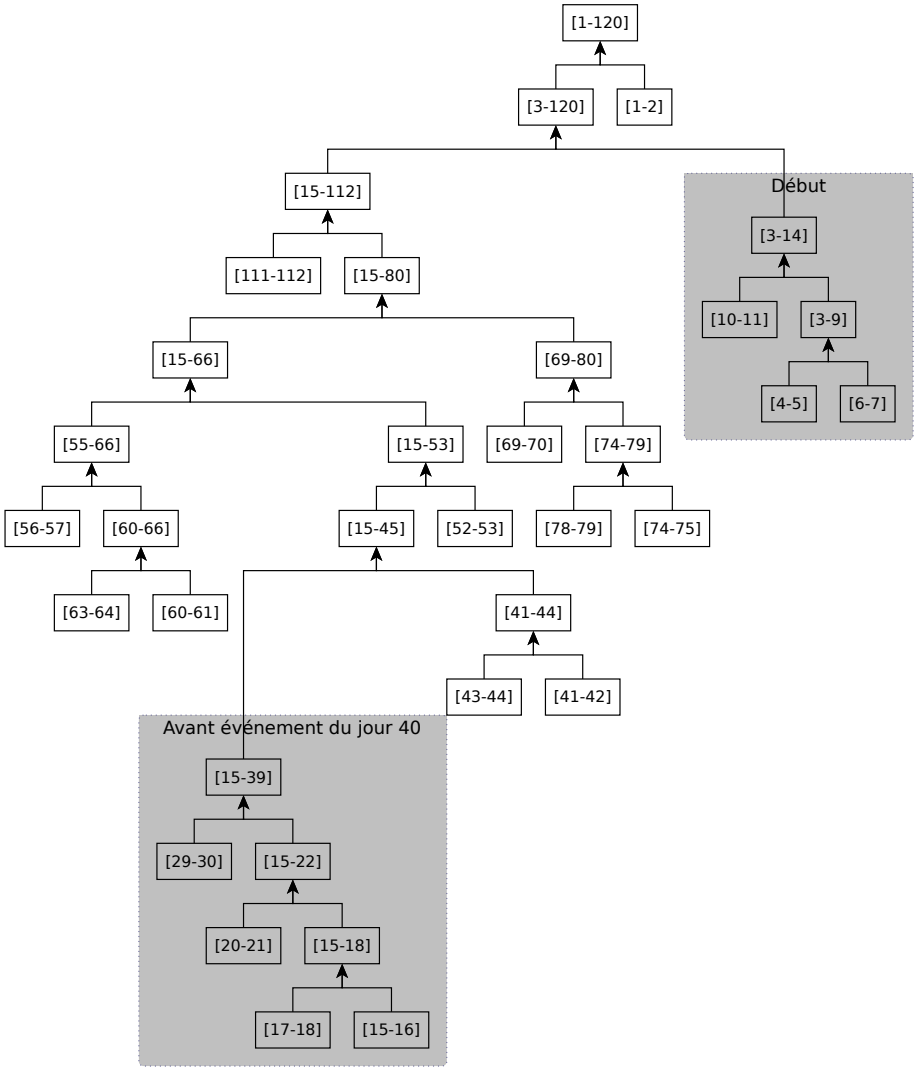


FIGURE 4.12 – Arbre après filtrage obtenu sur Blogs en forçant les fenêtres à être contiguës. Les blocs gris représentent explications potentielles.

dessiné une représentation de l'arbre du cas non contigu sur la figure 4.12. Il y a très peu de longues fenêtres de temps. La plus grande fenêtre contient les premiers pas de temps, qui sont en effet les plus différents de la structure moyenne. Nous n'avons pas réussi à interpréter les autres fenêtres par manque de connaissance sur l'objet mesuré. Nous savons uniquement qu'un événement de mesure a eu lieu le jour 40 et cet événement apparaît dans les deux fenêtres les plus basses.

La figure 4.13 présente l'arbre obtenu en cherchant les fenêtres de temps sur le réseau Radar avec des fenêtres contiguës. En effet, même si on échoue à détecter une bonne partition moyenne sur toute la mesure, il est potentiel-

lement pertinent d'en chercher une sur des sous-ensembles. Nous verrons au chapitre suivant en faisant de la détection d'événements sur Radar que les points de coupures correspondent à des événements identifiés indépendamment par d'autres méthodes ce qui valide en partie la décomposition. Nous avons assez peu d'information sur le réseau et nous n'avons pas réussi à interpréter l'arbre en ne forçant pas les fenêtres à être contiguës.

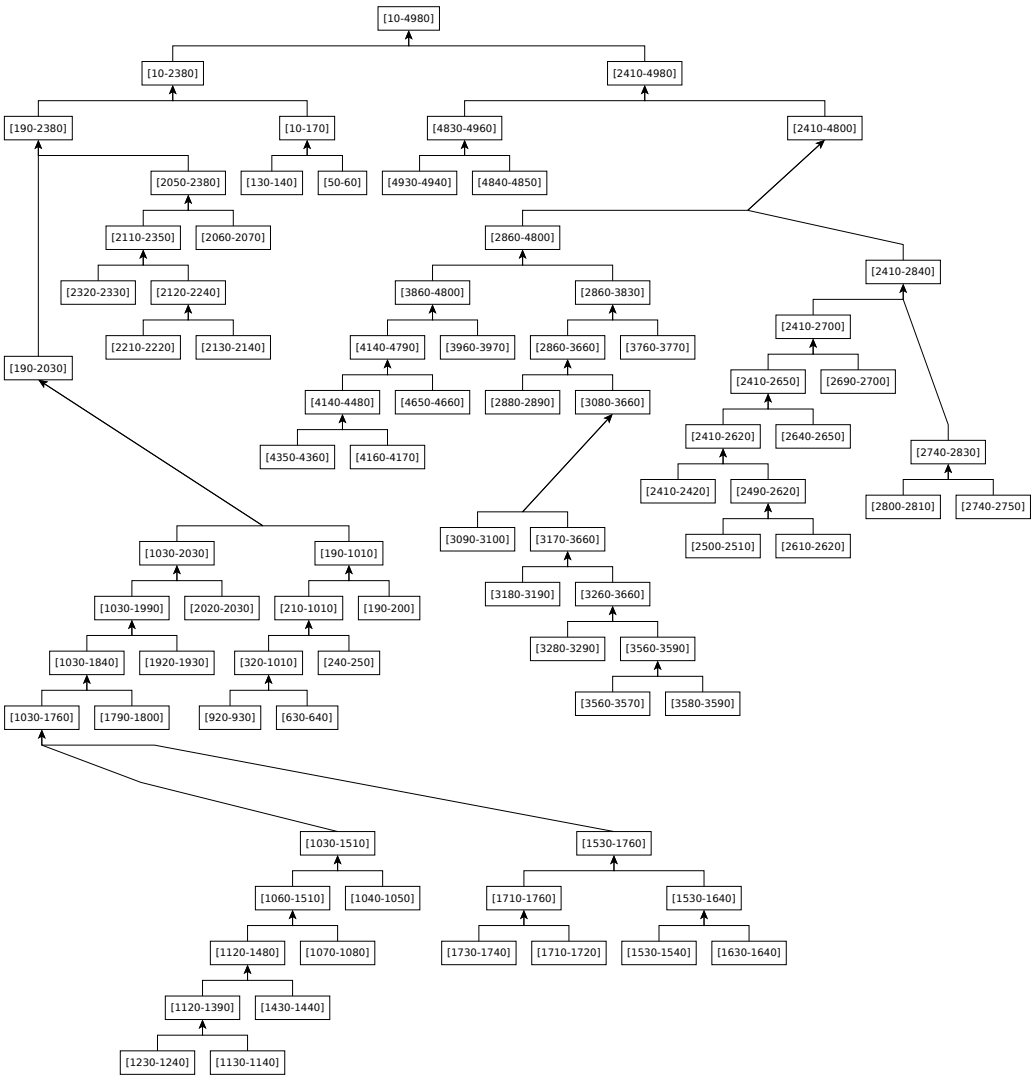


FIGURE 4.13 – Arbre de la décomposition du temps de Radar dans le cas de fenêtres contiguës après filtrage.

Finalement, ces résultats semblent assez prometteurs. La décomposition est souvent pertinente quand les graphes ont une évolution divisée en phases. Les deux approches, contiguë ou non, sont complémentaires : les fenêtres contiguës sont plus simples à interpréter et produisent plus directement des arbres

exploitables tandis que les cas non contigus sont moins sensibles aux événements et peuvent détecter des répétitions de structures, comme entre la phase 1 et 3 de Mrinfo.

4.5.4.1 Vitesse

L'algorithme produit ces segmentations du temps relativement rapidement. Si le graphe dynamique est composé de n instantanés, il y a en tout $2.n$ détections de communautés à effectuer. Il y a au départ une détection à faire sur chaque instantané, soit n (mais sur des petites parties du graphe dynamique) puis il y a au plus n regroupements donc au plus n décompositions et on arrive au final à $2.n$ applications de la méthode de Louvain. La complexité de la méthode de Louvain est délicate à évaluer. En effet, il existe une borne supérieure élevée connue ($O(L^3)$ avec L le nombre de liens), mais l'algorithme se comporte de manière quasi linéaire en pratique sur des réseaux ayant des communautés. Il faut aussi compter que chaque décomposition est évaluée sur tous les instantanés, mais un calcul de modularité revient à faire un parcours de toutes les arêtes et est linéaire. Il y a donc $2.n$ calculs qui sont linéaires. Dans la pratique, la principale limitation est la mémoire pour stocker le graphe.

4.6 Conclusions

Nous avons montré que des communautés peuvent être détectées non seulement sur un instantané donné mais aussi sur une longue période en optimisant la qualité moyenne. Nous avons proposé deux algorithmes pour effectuer en pratique cette optimisation : un premier très rapide et un second offrant une meilleure qualité tout en restant applicable à de larges jeux de données.

La nouvelle fonction de qualité est très générique et les poids permettent de privilégier certains pas de temps ou de prendre en compte leur durée. Nous avons utilisé la modularité comme fonction de qualité instantanée de base, mais d'autres peuvent remplir ce rôle. Utiliser une fonction de qualité statique connue est un atout pour la validation des résultats : si on admet que la fonction de qualité sur un instantané est une bonne estimation de la qualité des partitions et que nous avons la même qualité sur un pas de temps, alors notre nouvelle méthode est aussi valide sur ce pas de temps. Avoir des communautés qui existent sur plusieurs pas de temps et des diagrammes comme celui d'existence des nœuds au cours du temps fournit de nouveaux outils d'analyse et permet d'obtenir de nouvelles informations sur la vie des communautés.

Quand on optimise la modularité moyenne sur une période donnée, la question de la durée de la fenêtre de temps devient cruciale. Nous avons proposé

une méthode pour extraire automatiquement des fenêtres de temps intéressantes qui peut s'appliquer sur des réseaux dynamiques composés de milliers de nœuds et de pas de temps. Nous avons aussi défini une nouvelle fonction de similarité entre graphes dynamiques basée sur leur structure : si deux graphes ont la même structure pendant une fenêtre de temps, la partition de l'un sera bonne sur l'autre et inversement.

Ces résultats donnent de nouvelles informations sur les réseaux étudiés. Le processus de construction de Blogs a un effet significatif sur les conclusions que l'on peut obtenir et on peut imaginer qu'il faut diminuer le poids des liens vus longtemps auparavant ou considérer le réseau des liens vus sur une durée donnée plutôt que depuis le début de la mesure. Nos algorithmes sont plus efficaces sur Mrinfo et Imote où des phases que nous arrivons à détecter existent. Les résultats sont plus mitigés avec le réseau Radar. C'est une mesure très bruitée ayant de nombreux biais et ayant une structure particulière, car obtenue en prenant l'union de plusieurs arbres Par conséquent, il n'est pas étonnant que les résultats y soient moins bons. On verra par la suite que l'arbre est quand même découpé à des moments déjà identifiés comme des événements.

D'autres fonctions de similarité peuvent exister. La fonction utilisée a le défaut de mesurer à la fois si deux fenêtres sont similaires et modulaires. Ainsi, deux fenêtres de temps identiques seront d'autant plus proches que leur structure est modulaire et une normalisation adaptée pourrait corriger ce défaut. Nous en avons essayé quelques-unes sans que l'amélioration des résultats soit probante. Une autre limitation de la modularité moyenne est la sommation elle-même. Étant donné que la somme est commutative, l'ordre des instantanés n'a aucune influence. Cela implique qu'il n'y a pas de causalité et que, par exemple, la succession d'instantanés G_1, G_2, G_3 est strictement équivalente à la succession G_3, G_2, G_1. Dans une première approximation, c'est acceptable, mais à terme une métrique rendant compte de la causalité serait bien plus intéressante. On peut envisager premièrement d'en tenir compte par exemple en augmentant les poids des pas de temps les plus tardifs. On peut aussi changer la moyenne pondérée en toute fonction dépendant des modularités instantanées des pas de temps de la fenêtre considérée et potentiellement définir une fonction tenant compte de la causalité. Une telle fonction semble encore très difficile à définir de manière consensuelle dans le cas général.

Chapitre 5

Applications

Sommaire

Nous allons maintenant présenter plusieurs applications de la détection de communautés à des réseaux dynamiques que nous avons effectuées en collaboration. Nous allons l'appliquer à la détection d'événements sur Internet et à la segmentation de vidéos. Nous démontrons ainsi la variété des applications possibles de la détection de communautés dans les graphes dynamiques. Pour la détection d'événements, nous allons nous baser essentiellement sur les méthodes décrites au chapitre 3 tandis que nous verrons que la segmentation de vidéos va nécessiter des algorithmes ad hoc. En collaboration avec Mario Chavez de l'Université Pierre et Marie Curie, nous sommes aussi en train d'étudier comment appliquer nos résultats à l'analyse de la strucutre de réseaux neuronaux afin de prédire certaines crises d'épilepsie. Ces travaux sont hélas encore trop préliminaires pour que nous les présentions ici.

5.1 Détection d'événements

La détection d'événements consiste à identifier des phénomènes sortant de l'ordinaire dans le réseau comme par exemple un accroissement soudain de la taille, la disparition d'un grand nombre de nœuds ou l'apparition d'un lien entre deux parties sans aucun rapport. Il faut donc d'une part définir cet "ordinaire" et proposer des méthodes pour détecter les cas sortant de l'ordinaire.

La détection d'événements est un problème très important pour les réseaux de télécommunication où un événement est sans doute une panne nécessitant une intervention. Faire cette détection automatiquement est une nécessité, car avec l'accroissement de la taille des réseaux, une surveillance humaine de tous les éléments est impossible. Cette notion d'événements peut aussi s'appliquer à d'autres types de réseaux. Un événement sur un réseau social peut être la diffusion d'une nouvelle marquante et sur un réseau de bibliométrie peut correspondre à l'apparition d'une nouvelle idée innovante.

Détecter des événements est assez proche des méthodes vues au chapitre précédent et se ramène souvent à détecter les points de rupture. Les communautés servant à décrire la structure du réseau, il est alors naturel de vouloir étudier les moments où cette structure change particulièrement. Les méthodes décrites dans cette section sont des extensions des méthodes de détection d'événements statistiquement significatifs développées dans la thèse d'Assia Hamzaoui [38], mais sous le prisme des communautés.

5.1.1 Méthodologie

La méthodologie proposée par Assia Hamzaoui consiste à suivre des propriétés du réseau au cours du temps. Un exemple de propriété classique est le nombre de nœuds. Ensuite, nous allons étudier la distribution des valeurs de cette propriété. Il peut alors y avoir plusieurs cas :

- La propriété prend des valeurs très centrées autour de la moyenne sans exception. On parle alors de distribution homogène. Dans ce cas, il n'y a pas d'événements détectables, toutes les valeurs sont normales. Cela peut être dû à l'absence d'événements réels dans le réseau ou au fait que la propriété considérée ne permet pas de les détecter. Si par exemple seuls les liens changent beaucoup et que les nœuds sont fixes, compter le nombre de nœuds ne permettra pas de détecter d'événements, le nombre de nœuds dans le réseau étant toujours le même. Un exemple d'une telle distribution de valeurs est donné sur la figure 5.1.
- À l'inverse, la propriété peut prendre des valeurs très écartées de la moyenne. On parle alors de distributions hétérogènes. L'exception est alors la norme et la notion de normalité n'existe tout simplement pas. Si la mesure était infinie, la moyenne pourrait ne pas être définie, comme pour certaines lois de puissance. Un exemple d'une telle distribution de valeurs est donné sur la figure 5.2.
- Enfin, la distribution des valeurs peut être bien centrée autour d'une moyenne avec quelques exceptions que nous appellerons événements. On parlera alors de distribution homogène avec événements. Dans ce cas, les moments où la propriété prend une valeur atypique sont les moments où se produit un événement. Nous allons donc chercher des métriques ayant ce type de distribution. Un exemple d'une telle distribution de

valeurs est donné sur la figure 5.3.

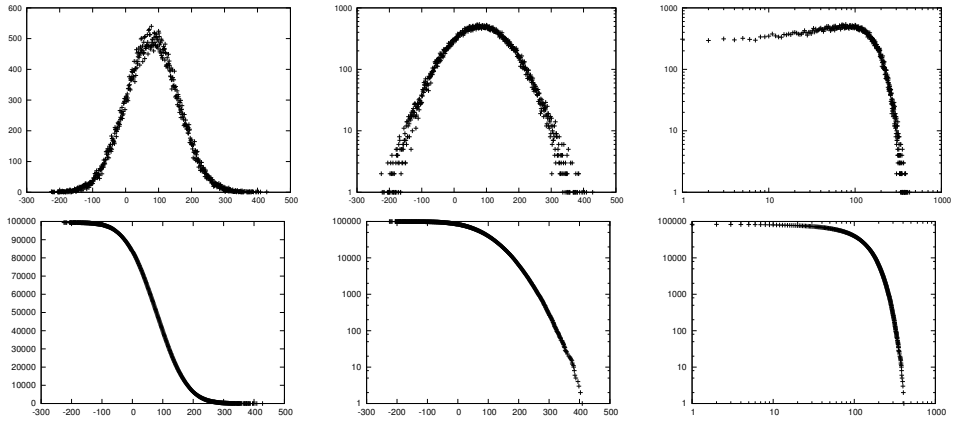


FIGURE 5.1 – Cas typique d'une distribution homogène. Première ligne (de gauche à droite) : la distribution dans les échelles lin-lin, lin-log et log-log. Deuxième ligne : la cumulative inverse de la distribution dans les mêmes échelles.

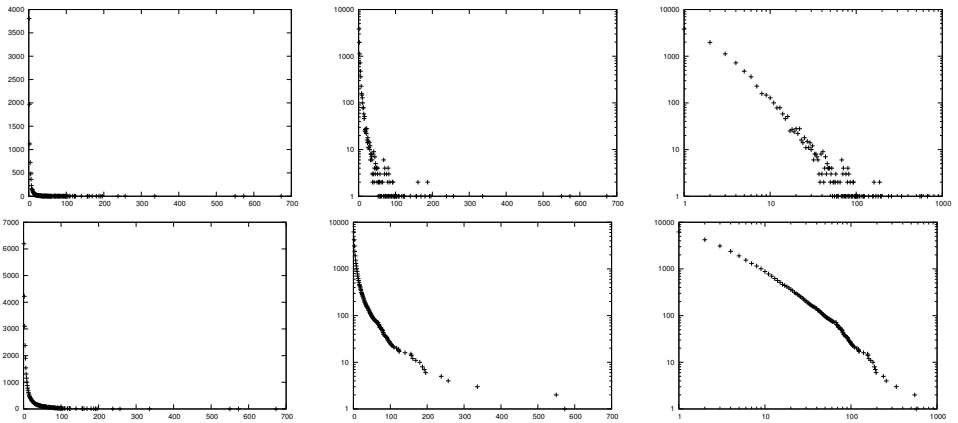


FIGURE 5.2 – Cas typique d'une distribution hétérogène. Première ligne (de gauche à droite) : la distribution dans les échelles lin-lin, lin-log et log-log. Deuxième ligne : la cumulative inverse de la distribution dans les mêmes échelles.

Des métriques ayant une distribution normale avec événements sont nécessaires pour détecter des événements mais pas suffisantes pour détecter des événements intéressants. La majorité de ce travail a été effectué sur Radar, et nous rappelons qu'il s'agit d'une mesure de la topologie de l'Internet tel que vue depuis une source appelée le moniteur. Si la connectivité du moniteur n'est pas bonne, tout Internet peut sembler disparaître d'un coup et ce qui ressemble à un événement majeur n'est qu'un épiphénomène.

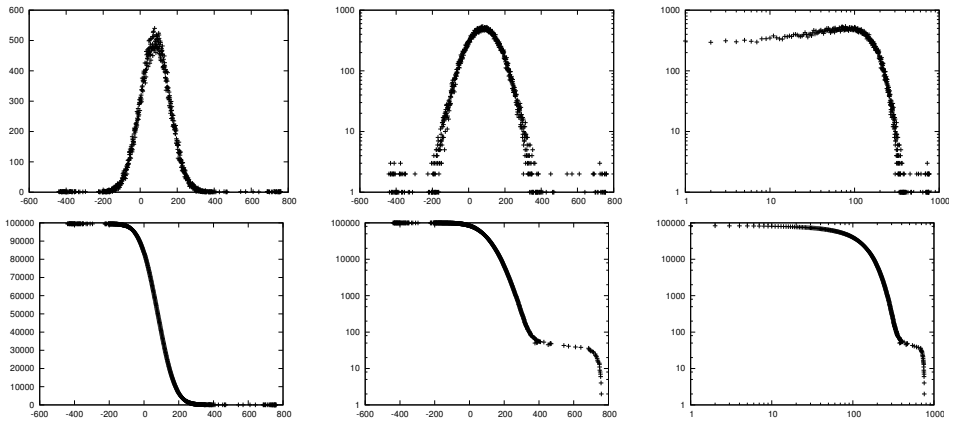


FIGURE 5.3 – Cas typique d'une distribution homogène avec *événements*. Première ligne (de gauche à droite) : la distribution dans les échelles lin-lin, lin-log et log-log. Deuxième ligne : la cumulative inverse de la distribution dans les mêmes échelles.

Radar est une mesure obtenue par traceroute et à chaque pas de temps, on mesure un arbre des chemins de la source vers les destinations. Il est inintéressant de détecter des communautés sur un arbre qui ne peut pas contenir de groupes de nœuds fortement liés et a une structure très particulière. Par contre une union d'arbres donne un graphe qui peut contenir une structure intéressante. Pour cela, nous allons définir, pour chaque pas de temps i et étant donnés deux entiers p et c, $G_i^c = (V_i^c, E_i^c)$ le graphe *courant* et $G_i^p = (V_i^p, E_i^p)$ le graphe *précédent*. Le graphe courant est l'union de c visions instantanées à partir de la i-ème, donc de i à $i + c - 1$ et le graphe précédent est l'union des p visions instantanées précédant la i-ème, donc de $i - p$ à $i - 1$. [1]

Nous nous sommes ensuite attachés à trouver de nouvelles métriques utilisant plus la structure des graphes, et notamment la structure communautaire, que des propriétés classiques comme le nombre de nœuds, le nombre de liens, les distances entre des nœuds, etc. Nous classons les nouvelles métriques en deux grandes catégories : celles basées sur les distances entre partitions et celles basées sur la modularité des partitions. Après leur définition et vérification que nous obtenons bien des métriques ayant une distribution homogène avec événements, nous verrons quels événements sont détectés et les corrélations avec les événements détectés par d'autres métriques.

1. Ce ne sont pas les mêmes graphes extraits de Radar que précédemment (nous utilisions des unions disjointes de 10 arbres) car nous voulons pouvoir comparer les événements trouvés par les méthodes utilisant des communautés à ceux déjà identifiés.

5.1.2 Métriques

5.1.2.1 Métriques liées aux distances entre partitions

Dans le cadre des métriques basées sur la distance entre deux partitions, nous utiliserons toujours la méthode de Louvain telle que modifiée au chapitre 3. En effet, la distance entre deux partitions sans stabilisation est toujours grande et par conséquent ne permet pas de détecter d'événements intéressants : l'instabilité occulte d'éventuels événements.

Nous avons testé deux métriques basées sur la distance entre partitions. La première, dont les résultats sont présentés sur la figure 5.4 est naturellement la distance entre la partition π^c_{t-1} du graphe G^c_{t-1} et la partition π^c_t du graphe G^c_t calculée en initialisant la méthode de Louvain avec π^c_{t-1}. La distribution n'est pas totalement comme celle des distributions homogènes avec événements car il n'y a que des valeurs atypiques supérieures à la moyenne, à savoir des moments où la partition à t est plus éloignée de la partition à $t-1$ que d'habitude.

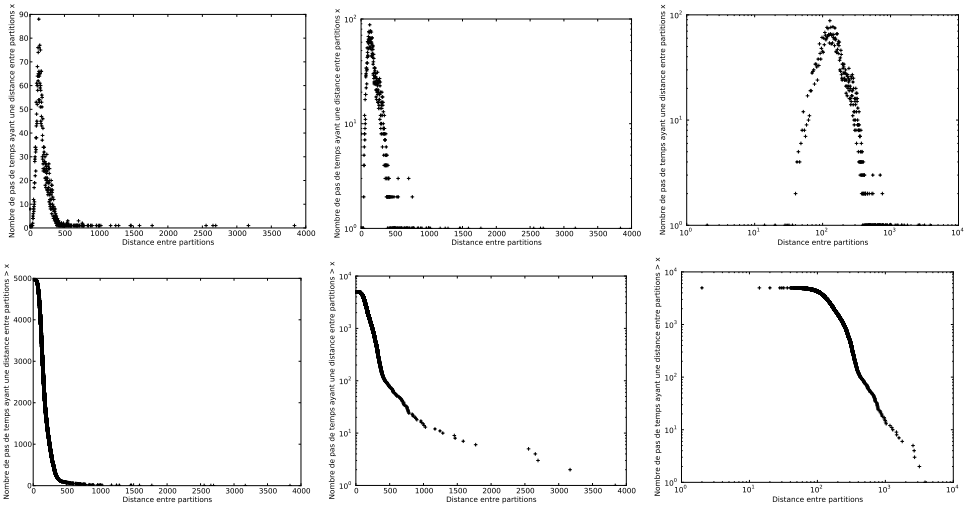


FIGURE 5.4 – Distribution des distances entre la partition de $G_c(t-1)$ et celle de $G_c(t)$. On observe des événements mais exclusivement supérieurs à la moyenne (avoir des moments où la partition est notablement trop stable serait étrange). Première ligne (de gauche à droite) : la distribution dans les échelles lin-lin, lin-log et log-log. Deuxième ligne : la cumulative inverse de la distribution dans les mêmes échelles.

La deuxième métrique testée est la distance entre la partition π^p_t du graphe G^p_t et la partition π^c_t de G^c_t. Là encore, nous avons calculé la partition de G^c_t en initialisant la méthode de Louvain avec la partition de G^p_t, sinon les distances sont irréalistes. La mesure semble homogène avec des événements (tous avec des valeurs supérieures à la moyenne).

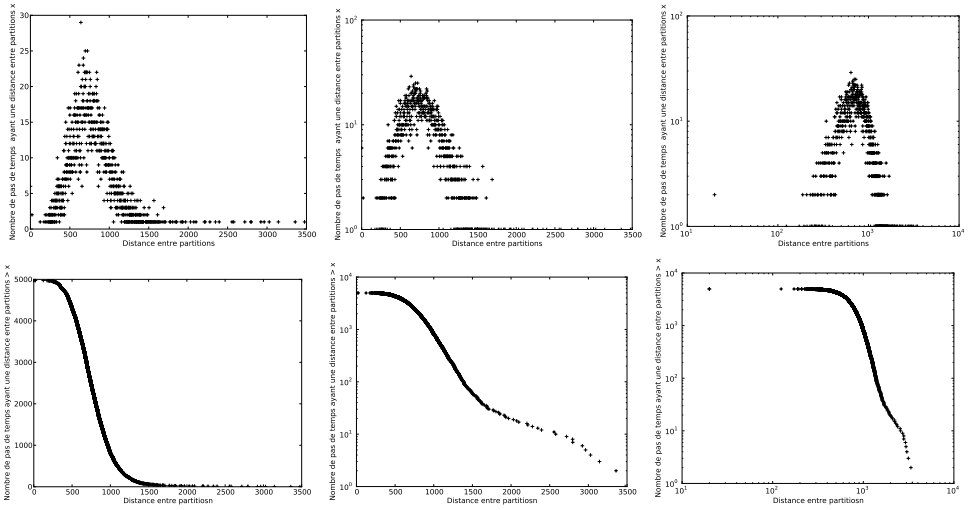


FIGURE 5.5 – Distribution des distances entre la partition de G_p et celle de G_c. On observe des événements mais exclusivement supérieurs à la moyenne. Première ligne (de gauche à droite) : la distribution dans les échelles lin-lin, lin-log et log-log. Deuxième ligne : la cumulative inverse de la distribution dans les mêmes échelles.

Nous utilisons ces deux approches car l'une correspond à une fenêtre *glissante* et l'autre non. En considérant les partitions de G^c_{t-1} et G^c_t, les graphes se recouvrent en partie et on peut donc s'attendre à ce que la partition soit plus stable. En revanche, les graphes G^p_t et G^c_t ne se recouvrent pas du tout et on peut s'attendre à une plus grande instabilité. Comme nous les verrons par la suite, ces deux approches détectent effectivement des événements différents.

5.1.2.2 Métriques liées à la modularité

Nous avons aussi étudié certaines métriques basées sur la modularité des partitions. On s'intéresse ici à la modularité comme mesure caractéristique du graphe et non à comparer des partitions. Nous cherchons donc la modularité maximale que l'on arrive à atteindre sur le graphe et par conséquent, nous allons utiliser la méthode de Louvain non modifiée qui optimise le plus la modularité. Nous avons testé plusieurs combinaisons de partitions de G^p, G^c et G^c_{t-1} utilisées sur les différents graphes. Les distributions des valeurs sont données sur les figures 5.6, 5.7 et 5.8. Le résultat est toujours le même : la modularité est en général quasi constante et les distributions très resserrées autour d'une valeur moyenne avec quelques événements très éloignés. L'éloignement est tel qu'il semble plutôt lié à des événements locaux (par exemple une déconnexion de la machine de mesure) et les événements ne sont donc pas forcément intéressants. Par conséquent, nous n'utiliserons pas ces métriques dans la partie suivante où nous analysons quels événements sont détectés par

quelles métriques.

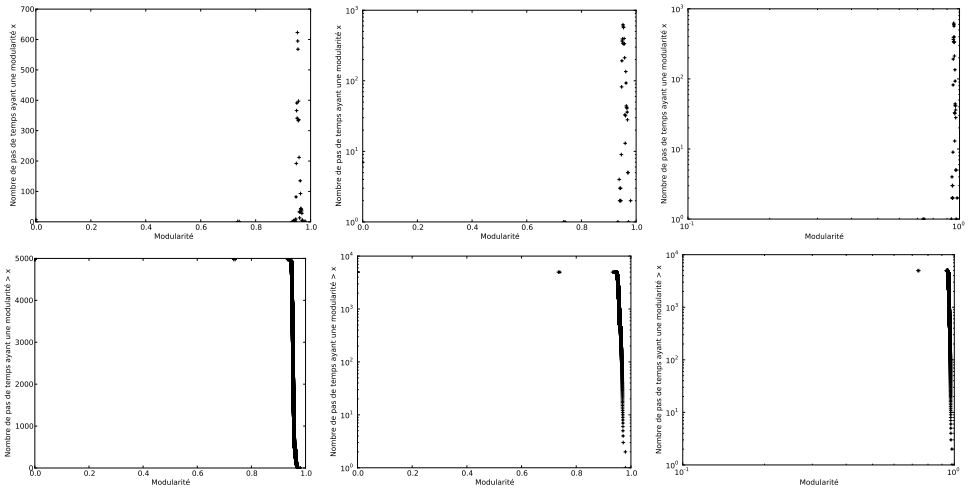


FIGURE 5.6 – Modularité de la partition de G_c. Les valeurs sont fortement regroupées autour de la moyenne, exceptés des événements qui sont très éloignés et font penser à des événements locaux et peu pertinents. Première ligne (de gauche à droite) : la distribution dans les échelles lin-lin, lin-log et log-log. Deuxième ligne : la cumulative inverse de la distribution dans les mêmes échelles.

5.1.3 Corrélation des événements

Nous avons représenté sur la table 5.1 la liste des événements détectés par les différentes métriques. Pour chaque métrique, nous avons trié les pas de temps du plus éloigné de la normale (la moyenne) au plus proche et avons retenu les 10 plus éloignés. C'est une première approximation de la notion d'événements. On ne peut alors que dire que s'il y a des événements, ce sont ceux-ci. Pour déterminer l'existence d'événements, il existe des tests statistiques comme le test de Grubbs. Son application montre qu'il y a en général moins de 10 événements, mais nous avons voulu prendre une définition un peu plus large. Nous avons par conséquent indiqué dans le tableau quel est le rang du pas temps dans la liste d'événements. Nous avons aussi regroupé les pas de temps correspondant sans doute au même événement car très peu séparés.

Cette liste permet de regrouper les métriques selon les événements détectés. On voit que par exemple le nombre de nouveaux nœuds et de nouveaux liens détectent pratiquement les mêmes événements. Un graphe des relations entre les différentes métriques est donné sur la figure 5.9. On constate que globalement, les métriques sont toutes assez proches exceptées la distance entre la partition de G_c^{t-1} et la partition de G_c^t. Les cinq autres ont chacune des évé-

t	Nouveaux nœuds	Nouveaux liens	Nombre comp. connexes	Taille plus grande comp. connexe	Distance partition G^c_{t-1} et G^c_t	Distance partition G_p et G_c
203	9			3		
204	5			1		
1014	4	9	4			
1015	2	3	2	5		4
1050				9		
1511	10			8		3
1610		8				
1644	8	5	10			
1645	6	2	9	10	10	2
1926					5	
1998					8	
2279				6		
2387					3	
2389					6	
2392					2	
2393					9	
2394					7	
2624			6		1	
2625			5		4	
2626			8			
2838	3	7	3	4		10
2839	1	4	1	2		7
2840						5
3664		10				6
3665	7	1	7			1
3666		6				
4759				7		
4859						8
4860						9

TABLE 5.1 – Liste pour chaque métrique des dix moments s'éloignant le plus de la normale. Les moments sont la colonne de gauche et dans chaque case il y a le rang de l'événement pour la métrique considérée. Ainsi, s'il y a 1 de marqué, c'est le moment où la métrique est la plus éloignée de sa moyenne, s'il y a 2 c'est la deuxième valeur la plus éloignée et ainsi de suite. Nous n'avons pas mis les moments au dessus de 10 car des tests comme le Grubb's test semblent montrer qu'il y a moins de 10 événements en général.

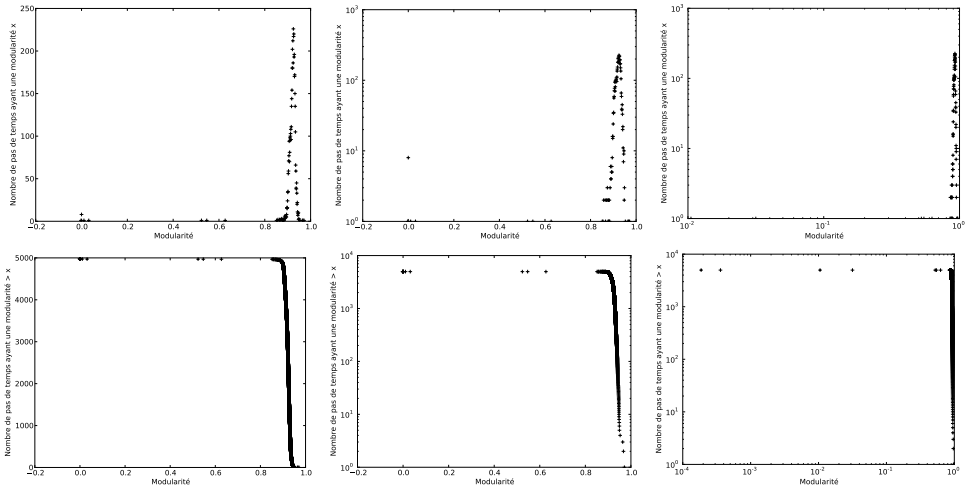


FIGURE 5.7 – Modularité de la partition de G^c_{t-1} utilisée sur G^c_t. Les valeurs sont fortement regroupées autour de la moyenne, exceptés des événements qui sont très éloignés et font penser à des événements locaux et peu pertinents. Première ligne (de gauche à droite) : la distribution dans les échelles lin-lin, lin-log et log-log. Deuxième ligne : la cumulative inverse de la distribution dans les mêmes échelles.

nements qu'elles détectent seules et d'autres événements qu'elles partagent, bien que ce ne soient pas toujours les mêmes.

5.1.4 Lien avec les fenêtres de temps

Nous avons vu au chapitre précédent une méthode pour décomposer le temps en différentes fenêtres. Les moments où l'algorithme décide de séparer une grande fenêtre de temps correspondent à des moments importants où la structure change. L'arbre est représenté sur la figure 4.13. Il s'agit d'une version un peu différente du réseau étudié ici car les arbres sont regroupés par 10 sans qu'il n'y ait de recouvrement et par conséquent les pas de temps considérés peuvent varier un peu. Nous pouvons remarquer dans l'arbre des changements de fenêtres aux instants, entre autres :

- autour de 2390 (regroupement des fenêtres [10-2380] et [2410-4980])
- autour de 1020 (regroupement de [190-1010] et [1030-2030])
- autour de 2850 (regroupement de [241-2840] et [2860-4800])
- autour de 180 (regroupement de [10-170] et [190-2380])

Chacun semble bien correspondre à des événements identifiés par les méthodes précédentes. Par conséquent, les événements et les fenêtres de temps semblent concorder bien qu'ils aient été obtenus indépendamment.

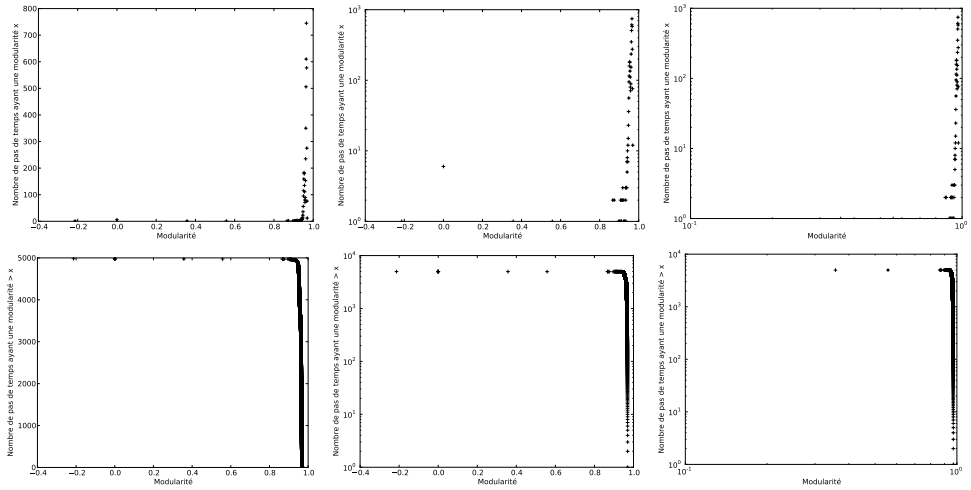


FIGURE 5.8 – Modularité de la partition de G_p utilisée sur G_c. Les valeurs sont fortement regroupées autour de la moyenne, exceptés des événements qui sont très éloignés et font penser à des événements locaux et peu pertinents. Première ligne (de gauche à droite) : la distribution dans les échelles lin-lin, lin-log et log-log. Deuxième ligne : la cumulative inverse de la distribution dans les mêmes échelles.

5.1.5 Conclusion

Nous avons donc étendu des travaux précédents sur la détection d'événements en utilisant des métriques plus proches de la structure du graphe. La modularité s'est avérée peu utilisable pour détecter des événements car elle a des variations très brutales avec des événements locaux. Nous aurions pu éliminer ces événements locaux, ce qui donne effectivement des distributions plus intéressantes, mais cela nous ferait sortir du cadre d'une détection d'événements par valeurs atypiques. La distance entre partitions s'est avérée bien plus efficace et permet une recherche effective d'événements. Les événements trouvés recouvrent certains événements déjà connus mais on en détecte des nouveaux ce qui confirme l'intérêt d'une recherche d'événements plus structuraux. Ces événements semblent souvent correspondre avec les changements de fenêtres de temps décrits précédemment ce qui est encourageant.

5.2 Analyse de vidéos

5.2.1 Introduction

Nous avons aussi appliqué la détection de communautés dans les réseaux dynamiques pour essayer d'étendre des outils d'analyse d'images (les objets statiques) à de l'analyse de vidéo (la dynamique). Une tâche fréquente en ana-

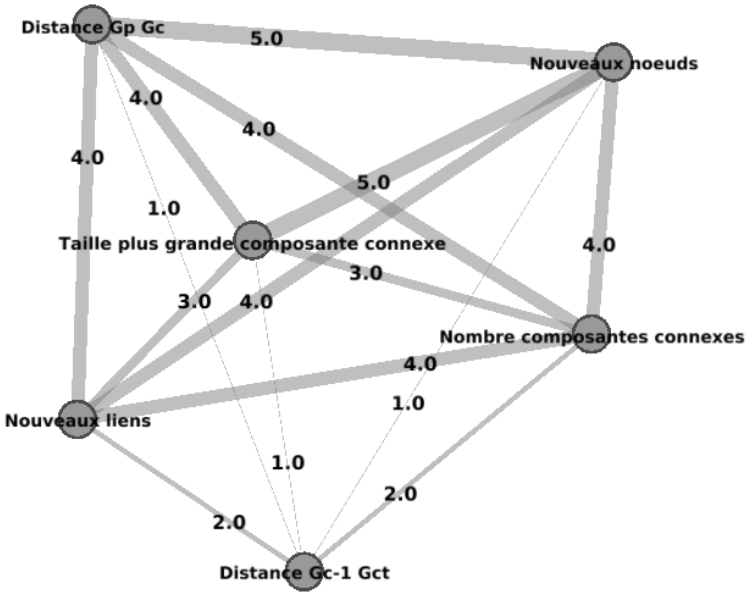


FIGURE 5.9 – Relations entre les métriques. Le poids est le nombre d'événements détectés par les deux métriques.

lyse d'image est la segmentation : il s'agit de déterminer quels sont les objets apparaissant dans une scène et donc de découper l'image en différentes zones, idéalement une par objet. Bien que cette tache semble absolument triviale pour n'importe quel homo sapiens, elle est particulièrement difficile à faire réaliser par un ordinateur. Un humain peut utiliser toutes ses expériences et ses capacités à mettre des concepts en parallèle (une voiture rouge et une voiture noire sont des voitures, là où l'ordinateur voit un tas de pixels noirs et un tas de pixels rouges). Des techniques informatiques basées sur les méthodes d'apprentissage automatique existent pour détecter un type d'objets particulier mais la segmentation en général et sans préapprentissage reste un problème difficile. Segmenter les images a de nombreuses applications comme la détection d'objets ou de personnes pour éviter une collision par une voiture, l'analyse automatique de données expérimentales (l'astronomie produit par exemple une tellement grande quantité d'images qu'une analyse manuelle est impossible), on peut imaginer des techniques de compression utilisant une meilleure compréhension de l'image ou des outils de recherches d'images. Ces travaux ont été réalisés en collaboration avec Arnaud Browet.

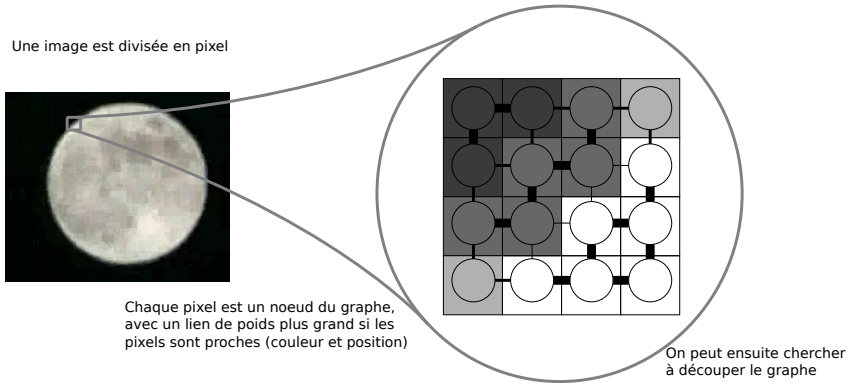


FIGURE 5.10 – Schéma du processus de segmentation. On construit un graphe à partir de l'image et on segmente ce graphe. Ici, $d_{max} = 1$ et donc on ne construit que les liens entre des pixels directement adjacents, mais on peut avoir des liens plus longs.

5.2.2 Méthodologie statique

Nous allons commencer par décrire rapidement la méthodologie utilisée pour segmenter des images décrite dans [16]. Tout d'abord, une image est une matrice de pixels et chaque pixel i est associé à un vecteur $F(i)$ qui peut décrire, suivant les situations, sa couleur en codage RGB ou une intensité dans le cas de niveaux de gris. Les techniques de segmentation d'images basées sur les graphes associent à chaque pixel un nœud dans le graphe G et construisent une matrice d'adjacence en utilisant les relations entre pixels : proximité dans l'image et proximité de couleurs principalement (voir figure 5.10). Une façon classique de définir la matrice d'adjacence W est par exemple :

$$w_{ij} = \begin{cases} e^{-\frac{d(i,j)^2}{\sigma_x^2}} e^{-\frac{||F(i)-F(j)||}{\sigma_i^2}} & \text{si } d(i,j) \leq d_{max}, \\ 0 & \text{sinon.} \end{cases}$$

d_{max}, σ_x et les σ_i étant des paramètres donnés par l'utilisateur et $d(i,j)$ la distance entre les pixels i et j. Ainsi, le poids d'un lien entre deux pixels sera d'autant plus grand qu'ils sont proches dans l'image et ont une couleur proche. σ_x correspond à l'importance donnée à la proximité des pixels (plus σ_x est petit, plus la distance va avoir d'importance) et σ_i à la proximité des couleurs (plus σ_i est petit, plus la couleur va avoir d'importance). Le paramètre d_{max} sert surtout à contrôler le temps d'exécution en diminuant le nombre de liens possibles.

Une fois le graphe représentant l'image construit, on cherche à regrouper l'ensemble de ses nœuds dans des ensembles contenant chacun un objet. De nombreuses techniques existent pour effectuer ce partitionnement de graphe.

Le nombre de nœuds étant très élevé, même pour des petites images, il faut des algorithmes très rapides et c'est donc ici que peuvent intervenir des algorithmes de détection de communautés comme la méthode de Louvain.

Pour aller plus loin, une vidéo peut ensuite être vue comme une série d'images : à chaque image est associé un graphe et nous pouvons donc associer à une vidéo un graphe dynamique. Les mêmes problèmes de stabilité et de suivi existent avec une difficulté supplémentaire : le pixel i à l'instant t et le pixel i à l'instant $t+1$ ne représentent pas forcément le même objet, car le pixel est fixe alors que l'objet a pu bouger. Par conséquent, on ne peut pas garantir que deux nœuds sont des représentants du même objet et le suivi est plus difficile.

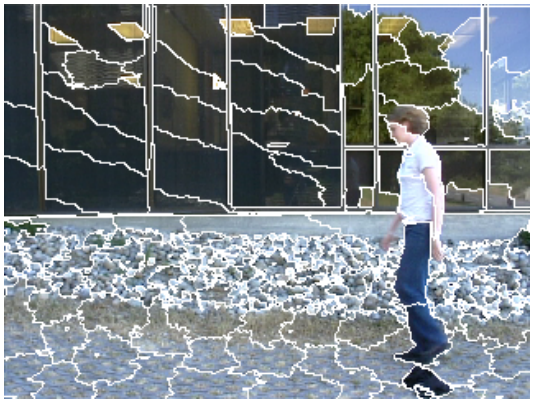
5.2.3 Méthodes déjà présentées

Nous avons tout d'abord commencé par appliquer les méthodes que nous avons déjà présentées à différentes vidéos. Il s'avère que la majorité ne fonctionne pas telles quelles, car les nœuds ne représentent pas toujours le même objet.

La première méthode consiste simplement à appliquer le même algorithme à chaque étape. Un exemple de résultat est représenté sur la figure 5.11. Deux problèmes émergent : instabilité et problème de résolution. Le problème de résolution est le fait que la maximisation de la modularité fait obtenir des communautés ayant une taille caractéristique fonction du nombre de liens. On constate ce phénomène par exemple sur les fenêtres noires en arrière-plan : elles sont découpées en plusieurs parties sans la moindre raison. Ces coupures sans raison sont une source d'instabilité, car l'algorithme ne les fait pas toujours au même endroit. La fenêtre la plus à gauche est d'abord coupée en 3 à l'instant t, puis en 4 à l'instant $t+2$ puis encore en 3 à l'instant $t+4$ par exemple.

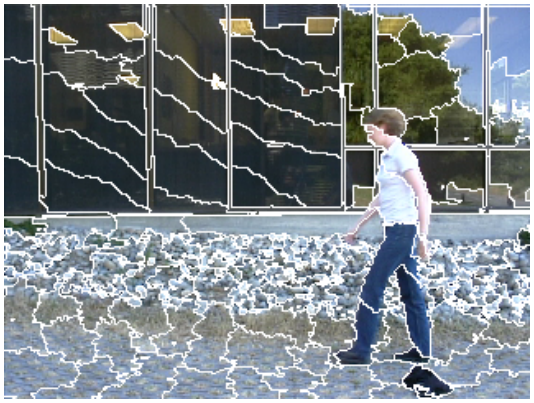
Afin d'améliorer la stabilité, nous avons essayé d'appliquer la méthode proposée au chapitre 3 qui consiste à initialiser l'algorithme avec la partition précédente. Néanmoins, le fait que le même nœud dans le graphe à des instants différents représente un objet différent fait que cette méthode est inefficace. Les nœuds commencent en effet directement dans de grandes communautés, et il n'est pas rentable de sortir un nœud d'une grande communauté, car on perd les liens qui l'y rattachaient sans gagner beaucoup de liens internes. Même en augmentant le paramètre d'aléatoire, les communautés n'évoluent pas et ne permettent pas de suivre les objets comme le montre l'exemple de la figure 5.12. On suit par exemple très mal le personnage, car sa communauté qui existait à l'instant t ne change pas et donc cesse rapidement de le couvrir.

Nous avons aussi un problème proche de la résolution limite décrite dans [9]. Les communautés trouvées ont une taille intrinsèque liée à la taille du graphe. Ici, il semble que les communautés trouvées sont trop petites et par conséquent nous pouvons encore appliquer la méthode de gestion de la résolution



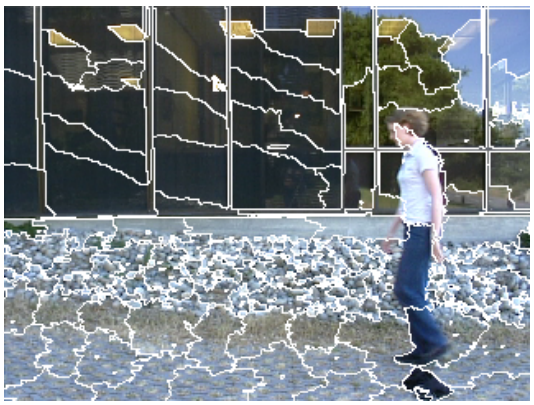
(a) t

(b) $t + 2$

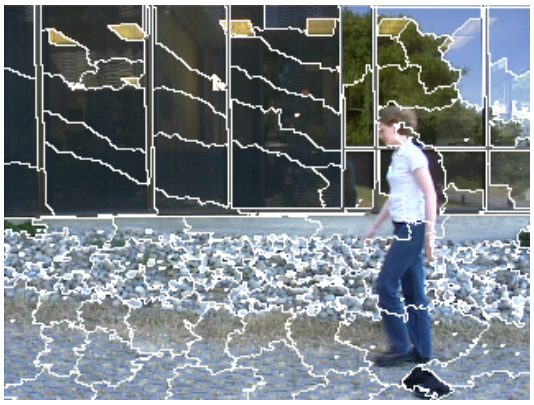


(c) $t + 4$

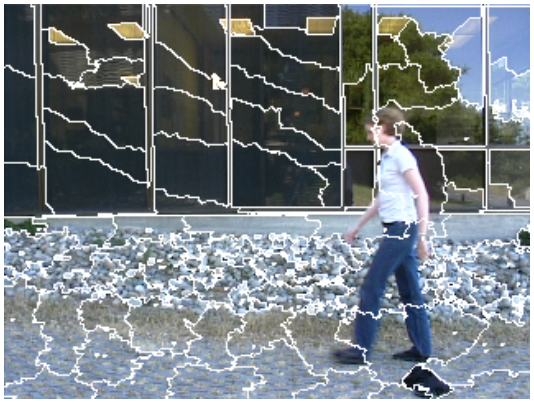
FIGURE 5.11 – Des communautés sur une vidéo en appliquant la méthode de Louvain directement. Les communautés sont délimitées par les traits blancs. On représente une image sur deux pour accentuer les différences.



(a) t



(b) $t + 2$



(c) $t + 4$

FIGURE 5.12 – Des communautés sur une vidéo en appliquant la méthode de Louvain modifiée pour repartir avec la partition précédente et un aléa de 50%. Les communautés sont délimitées par les traits blancs.

décrite dans [47] qui propose une nouvelle définition de la modularité. Elle généralise la modularité et les autres méthodes de gestion de la résolution. Elle définit un paramètre R[2]. Quand R vaut 1, nous optimisons exactement la modularité et plus R est grand, plus le nombre de communautés est faible et les communautés grandes.

Nous avons donc utilisé ce paramètre afin de diminuer le nombre de communautés. Les résultats sont représentés sur la figure 5.13. Le suivi est tout de suite bien plus stable et la qualité un peu meilleure sur les grandes communautés comme les fenêtres du fond. Par contre des détails disparaissent comme le bras droit qui n'est plus identifié. Il y a aussi encore des problèmes, comme avec la fenêtre de droite qui est découpée en 2 ou 3 suivant les images. De plus, il est délicat de fixer la valeur du paramètre R.

Nous avons aussi appliqué la méthode de détection de communautés présentée au chapitre 4 pour voir quel pouvait être son résultat. Le résultat d'une optimisation sur 20 images est représenté sur la figure 5.14. Comme on pouvait s'y attendre, la personne marchant n'est pas du tout prise en compte et seul le fond est segmenté. Cela est valide tant que le fond ne bouge pas, en cas de travelling par exemple le résultat est de bien moins bonne qualité.

5.2.4 Suivi multi-tranches

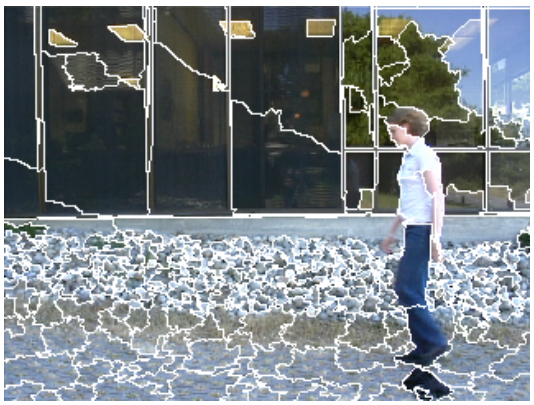
5.2.4.1 Présentation

Vu qu'il est difficile de faire l'association nœud à t, nœud à $t+1$, nous avons développé une méthode particulière de suivi via des graphes en tranches temporelles. L'objectif est d'intégrer la partition à t dans le calcul à $t+1$ afin d'améliorer la stabilité et de permettre un suivi plus efficace.

Pour cela, nous avons mis en parallèle le graphe à t et le graphe à $t+1$. Ensuite, nous avons utilisé la même fonction de poids pour mettre des liens entre les nœuds du temps t et les nœuds de $t+1$. Ainsi, les nœuds représentant des pixels qui sont proches dans l'image et en colorimétrie à t et $t+1$ sont plus fortement connectés. On obtient alors un nouveau graphe avec deux tranches. Le graphe de la tranche à t est alors réduit à son graphe entre communautés et on détecte alors les communautés sur ce nouveau graphe multi-tranches. Le processus est décrit sur la figure 5.15. La réduction du graphe à t à son graphe entre communautés a deux avantages :

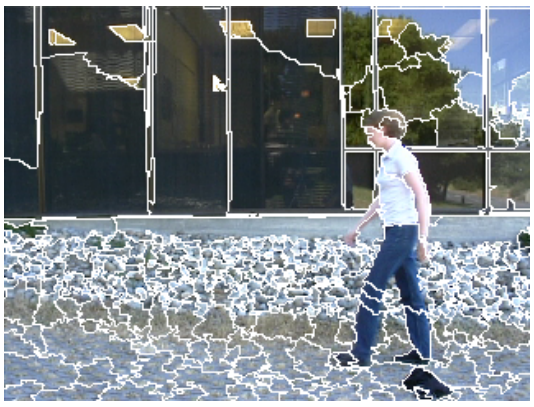
- On réduit grandement la taille du réseau car le nombre de communautés est très faible par rapport au nombre de pixels. Le gain de performance est donc non négligeable.
- Cela permet automatiquement d'intégrer les communautés précédentes dans le calcul tout en autorisant la création de nouvelles communautés.

2. il est noté t dans [47] car lié à une notion de temps de parcours mais pour éviter de le confondre avec le numéro du pas de temps, nous le noterons R



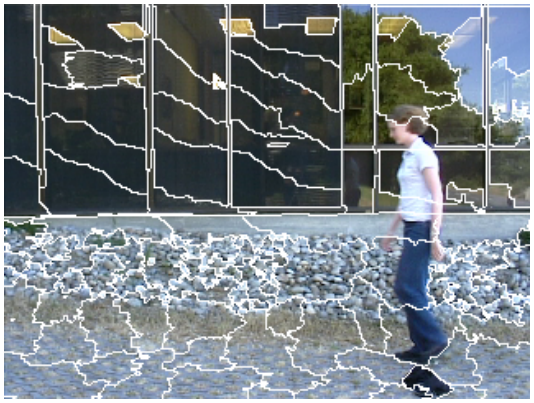
(a) t

(b) $t + 2$

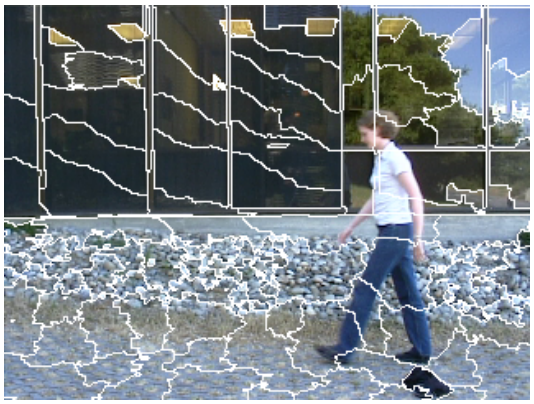


(c) $t + 4$

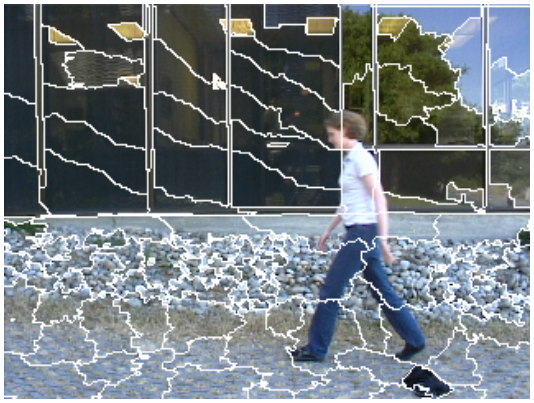
FIGURE 5.13 – Des communautés sur une vidéo en appliquant la méthode de Louvain modifiée avec le paramètre R à 50. Les communautés sont délimitées par les traits blancs.



(a) t



(b) $t + 5$



(c) $t + 10$

FIGURE 5.14 – Des communautés sur une vidéo en cherchant à maximiser la modularité moyenne. Les communautés sont délimitées par les traits blancs. Les images sont prises à des instants écartés pour mieux voir ce qui bouge ou pas.

Les nœuds de $t+1$ qui seront regroupés avec une communauté de t sont la suite de celle-ci. De plus, les communautés à t ne peuvent plus être divisées car chacune n'est qu'un seul meta-nœud.

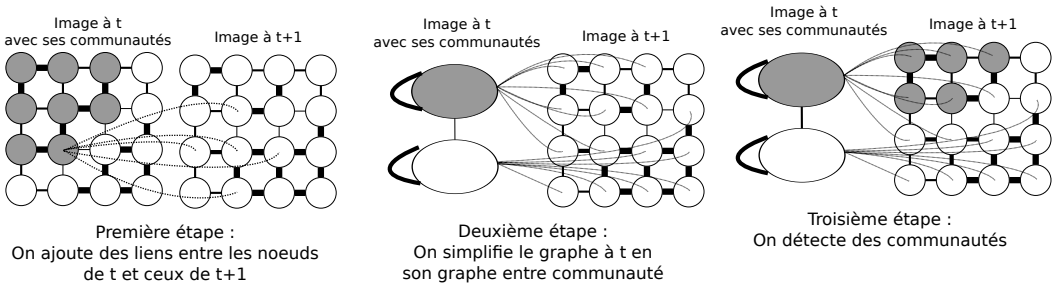
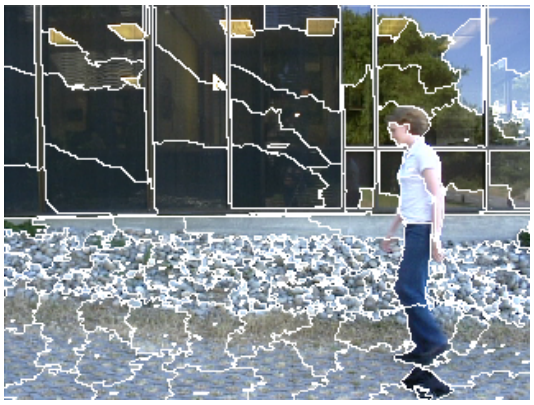


FIGURE 5.15 – Schéma du processus de suivi multi-tranches appliqué au suivi de vidéos.

5.2.4.2 Résultats

Les premiers résultats sont représentés sur la figure 5.16. Il reste des problèmes liés à la résolution des communautés, par exemple sur la fenêtre de gauche. Un phénomène intéressant est que les communautés continuent à exister même si la division est moins marquée sur l'image. Par exemple si l'on observe les cheveux de la personne qui marche, avec la méthode de Louvain avec un peu d'optimisation de la résolution, les cheveux sont regroupés avec le fond et cessent donc d'être distincts tandis qu'avec du suivi multi-tranches, ils continuent d'être dans une communauté à part. Le fait de changer la taille du réseau en ajoutant les communautés précédentes change la taille des communautés. Il est encore possible de jouer sur la résolution via le paramètre R, mais cela ne semble pas nécessaire ici.

Au final, la détection de communautés dans les vidéos pose encore de nombreux problèmes. Le suivi multi-tranches semble être le plus prometteur et résout assez facilement et élégamment le problème du suivi, mais ne résout pas tous les problèmes de résolution. La taille des réseaux (une image de 800×600 conduit directement à des graphes de 480000 nœuds et plusieurs millions d'arêtes) et leur structure très particulière (excepté au bord de l'image, tous les nœuds ont le même degré) font que les algorithmes de détection de communautés doivent être adaptés. Par exemple, la méthode de Louvain n'est plus aussi rapide sur de tels réseaux que sur des réseaux sociaux. La validation est aussi délicate car si l'on est capable de voir les défauts des méthodes à l'œil, il manque un critère objectif de comparaison.



(a) t

(b) $t + 2$

(c) $t + 4$

FIGURE 5.16 – Des communautés sur une vidéo en appliquant l'algorithme de suivi de vidéo multi-tranches.

Chapitre 6

Conclusions et perspectives

Nous nous sommes concentrés dans cette thèse sur la problématique de la détection de communautés, c'est-à-dire de zones localement denses et faiblement interconnectées, dans des graphes de terrain dynamiques. La contribution principale de cette thèse réside dans l'étude de deux approches complémentaires pour l'étude de la structure de ces réseaux. La première approche consiste à regarder la structure *instantanée* : on découpe le graphe en pas de temps et on étudie la structure à chaque instant. La deuxième approche consiste à regarder la structure sur le long terme : au lieu de chercher une division à chaque instant, on en cherche une qui est plutôt bonne sur une longue durée.

L'idée d'utiliser les algorithmes statiques sur chaque pas de temps comme présenté au chapitre 3 est très naturelle et déjà appliquée. En revanche, peu de travaux se sont attachés à étudier des cas simples comme la suppression d'un seul nœud. Cela nous a permis de montrer que bien que naturelle, c'est une idée dangereuse si l'on ne prend pas plus de précautions. En effet, tous les algorithmes utilisés se sont avérés beaucoup trop instables pour fournir des informations pertinentes. Ainsi, en supprimant juste un nœud et donc en nous plaçant dans un cas contrôlé, nous avons pu montrer que les résultats étaient totalement irréalistes. Si les algorithmes échouent dans des conditions simples, il semble illusoire d'espérer qu'ils produisent des résultats valides dans des cas plus complexes. Nous avons ensuite étudié en détail les causes de cette instabilité dans le cas particulier de la méthode de Louvain et avons proposé une solution permettant des résultats bien plus réalistes.

En creusant cette question de la stabilité vers l'exemple extrême d'une partition unique durant toute la mesure, nous avons abouti au résultat du chapitre 4 qui est une définition de partitions qui sont de haute qualité sur une longue durée appelée fenêtre de temps. Nous avons proposé deux méthodes de calcul de cette partition : une très rapide et une produisant des résultats de meilleure qualité. Cela nous a permis de mettre en évidence de nouveaux phénomènes et de proposer une méthode de décomposition du temps en fenêtres pertinentes. Une originalité de cette méthode est qu'elle n'impose pas aux fenêtres de temps d'être d'un seul tenant et par conséquent des phénomènes répétés ou de simple transitions temporaires peuvent être observés là où la

majorité des méthodes ne permettaient que de découper à certains moments considérés comme des événements.

Nous nous sommes attachés à appliquer ces algorithmes au chapitre 5. Ainsi, l'algorithme de stabilisation du chapitre 3 nous a permis de détecter des événements en utilisant la structure du réseau et ceci de manière plus satisfaisante. Sans cette modification, il y aurait eu constamment des événements sur le réseau. Ceux-ci n'auraient par conséquent plus été des événements. Nous savons aussi que l'algorithme a été employé pour le service *inmaps* de *LinkedIn* [51] qui permet à leurs clients de représenter leur réseau professionnel avec une classification en groupes. La première coloration en groupe est effectuée avec la méthode de Louvain et la mise à jour des groupes quand le réseau évolue par la modification proposée.

L'application à la fouille de vidéo a nécessité par contre le développement d'un nouvel algorithme basé sur l'idée de tranches temporelles. Cela est lié au fait que dans le cas de vidéo, un nœud donné ne représente pas forcément le même objet à différents pas de temps ce qui est une hypothèse implicite de nos deux algorithmes. Néanmoins, nous avons pu obtenir des résultats intéressants ouvrant la voie à une segmentation suivie de vidéos.

Parmi les questions en suspens, la plus importante est celle de la validation. Dans le cas des communautés statiques, la méthode standard est d'utiliser un réseau avec une structure connue et de vérifier que le résultat fourni par l'algorithme concorde. Il existe des données réelles sur lesquelles certains phénomènes sont connus comme le graphe *Karate* dont les nœuds représentent les membres d'un club de karaté reliés suivant le nombre d'activités qu'ils font en commun en dehors du club. Il s'avère que suite à une dispute de deux représentants du club, celui-ci va se scinder en deux et la façon dont cette scission a lieu est bien prévue par la détection de communautés. Cela valide en partie l'approche. On peut trouver d'autres réseaux similaires sur lesquels existe une vérité connue ainsi que des études permettant de valider que les regroupements ont du sens pour des spécialistes du domaine des données représentées par le réseau. Il existe aussi des outils permettant de générer des réseaux semi-aléatoires avec une structure imposée et on peut alors étudier si les algorithmes la détectent. Ces outils contiennent souvent des paramètres de difficulté ce qui permet de comparer les algorithmes entre eux. En revanche, presque rien de tel n'existe pour les réseaux dynamiques. Quelques réseaux existent avec une structure connue comme le réseau de mails entre des employés d'Enron mais certaines de ses propriétés font douter de sa validité (il n'est par exemple jamais connexe et trop petit pour que des communautés y aient vraiment du sens).

Il est peu probable qu'un seul outil de génération de graphes avec une structure connue suffise à valider les algorithmes dynamiques, car ils ne sont pas valides dans les mêmes situations. Par exemple, la détection de fenêtre n'a

de sens que quand des fenêtres existent réellement, ce qui n'est sans doute pas le cas sur des réseaux croissant comme le réseau de blogs. Par contre, ceux-ci ne contiendront sans doute pas de structures répétées comme un effet jour-nuit sur un réseau de communication. Sans une première analyse des dynamiques observables, construire un tel outil était donc impossible. Cette thèse est une première analyse et nous espérons qu'elle facilitera l'émergence de méthodes rigoureuses d'évaluation.

Un autre verrou à l'utilisation de la dynamique est l'absence de prise en compte du recouvrement des communautés. Il n'existe pas aujourd'hui de méthode consensuelle permettant de classer les nœuds en permettant à certains d'appartenir à plusieurs communautés. Cela semble pourtant un comportement standard dans les réseaux sociaux par exemple. Ainsi, dans le cas d'un graphe dynamique, on imagine facilement qu'une grande partie des transformations se font continuellement : un sous-groupe d'une communauté la quitte peu à peu pour en rejoindre une autre par exemple. Il est impossible de représenter ceci sans recouvrement et cela impose des évolutions faites d'importantes ruptures comme la scission d'un coup d'une communauté. Ainsi, une compréhension profonde de la dynamique nous semble difficile sans recouvrement.

Chapitre 7

Annexe : structure multi-échelle

Sommaire

7.1 Introduction

Quelle que soit la technique de détection de communautés utilisée, on peut trouver plusieurs niveaux de hiérarchie : les communautés sont elles-mêmes composées de sous-communautés. Ainsi, si l'on considère le graphe des citations entre articles, on devrait retrouver une communauté d'articles pour chaque domaine, par exemple l'informatique. L'informatique elle-même contient l'intelligence artificielle, la cryptographie, la théorie de l'information et bien d'autres. Le site arxiv.org hiérarchise ainsi les papiers suivant différentes disciplines, qui, elles-mêmes, contiennent des sous-domaines qui, parfois, sont eux-mêmes catégorisés.

Si cette structure hiérarchique n'est pas connue *a priori*, elle peut être obtenue en appliquant à nouveau de manière récursive une technique de décomposition sur les communautés puis sur les sous-communautés obtenues, etc. Cette structure se représente efficacement par un arbre donc chaque nœud est une communauté et les fils sont les sous-communautés. Cette annexe présente quelques résultats obtenus sur la structure multi-échelle d'un graphe obtenue en rédécomposant récursivement chaque communauté. Nous les présentons en annexe car il n'y a pas de lien direct avec la notion de dynamique. Nous pensons néanmoins qu'à terme les deux notions vont devoir se rejoindre car il est raisonnable de penser que beaucoup des changements structuraux s'opèrent d'abord dans des sous-communautés.

Dans la suite, nous commençons par décrire précisément la méthode utilisée pour construire effectivement l'arbre représentant la structure multi-échelle. Ensuite, nous présentons plusieurs propriétés de l'arbre afin de mettre

en évidence la structure multi-échelle. Enfin, nous montrons que la structure ainsi obtenue décompose des communautés à l'excès et que, s'il est nécessaire d'étudier la structure hiérarchique, il ne faut pas vouloir trouver des sous-communautés là où il n'y en a pas.

7.2 Méthode de décomposition

Afin d'étudier la structure multi-échelle des graphes de terrain et notamment des inclusions de communautés, nous avons donc utilisé une approche récursive consistant, à partir d'un graphe donné, à le décomposer en communautés, puis à redécomposer chaque communauté en sous-communautés, et ainsi de suite. Ce procédé est répété récursivement sur chaque communauté ou sous-communauté, jusqu'à obtenir des communautés qui ne peuvent plus être décomposées. Bien que la méthode soit indépendante de la méthode de décomposition utilisée, nous utiliserons exclusivement la méthode de Louvain. Il nous faut une méthode passant bien à l'échelle car pour avoir plusieurs niveaux intéressants il faut que les communautés initiales soient grandes pour être susceptibles d'en contenir et par conséquent les graphes doivent être grands et les algorithmes doivent pouvoir s'appliquer à eux.

Les communautés non décomposables ne sont pas forcément des sommets seuls, en effet certains graphes ne peuvent pas être décomposés en plusieurs communautés sans obtenir une modularité plus faible que si on regroupait tous les nœuds ensemble (voir figure 7.1 pour quelques exemples). Il faut aussi remarquer que, bien que certains graphes soient décomposables, vouloir décomposer à tout prix dès lors qu'il y a un gain de modularité peut amener à des communautés peu ou pas naturelles du fait de la résolution limite déjà évoquée.

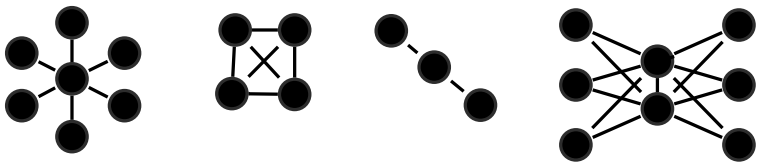


FIGURE 7.1 – Exemples de graphes pour lesquels toute tentative de décomposition en communautés fournit une modularité inférieure à celle du graphe considéré comme une seule communauté

Les résultats de ces décompositions successives peuvent être synthétisés dans un arbre, généralement appelé dendrogramme. Chaque sommet s de l'arbre correspond à une communauté calculée par l'algorithme, c'est-à-dire à un ensemble de sommets. Les fils d'un sommet s sont les sous-communautés de la communauté s telles que calculées par l'algorithme. On complète l'arbre

avec une racine qui correspond au graphe entier et, de plus, les communautés non décomposables ont comme nœuds fils les sommets du graphe qui les composent. Les seules feuilles de l'arbre sont donc les sommets du graphe original. On appelle niveau d'un sommet sa profondeur dans l'arbre. Ainsi la racine est au niveau 0, les communautés calculées initialement par l'algorithme sont au niveau 1, les sous-communautés au niveau 2, etc. La figure 7.2 présente un graphe et l'arbre obtenu avec la méthode de décomposition.

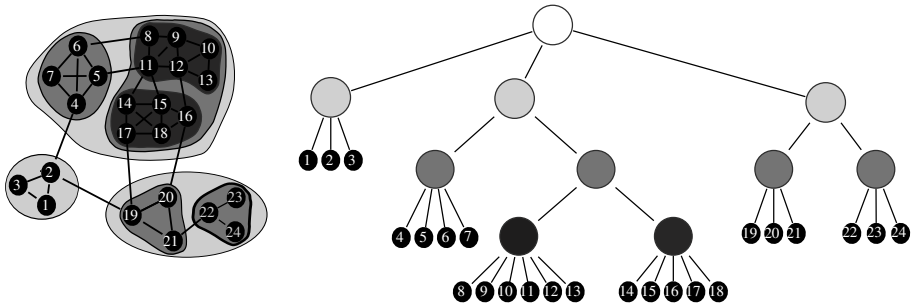


FIGURE 7.2 – Graphe avec représentation de la décomposition niveau par niveau et de l'arbre obtenu. Les feuilles de l'arbre sont les sommets du graphe initial

7.2.1 Graphes étudiés

Les travaux présentés ont été réalisés sur deux graphes de terrain statiques :

- *webndu* est un graphe du web pour lequel chaque nœud est une page web, et il y a un lien entre deux nœuds s'il y a un lien hypertexte de la première page vers la seconde. Le graphe considéré est une carte complète du domaine .nd.edu obtenue en 1999 et qui contient 325 729 pages et plus d'un million de liens [1]. Initialement orienté, on l'a considéré comme un graphe non orienté car il existe peu d'approches tenant compte de l'orientation. La détection des communautés sur ce graphe prend quatre secondes et la construction de l'arbre associé dure elle deux minutes trente secondes ;
- *arxiv* est un réseau de coauteurs d'articles de recherche dans le domaine de la physique extrait de la base de données *arxiv.org* [55]. À chaque auteur correspond un nœud du graphe, et il y a un lien entre deux auteurs s'ils ont cosigné un ou plusieurs articles [1]. La détection des communautés sur ce graphe prend moins de une seconde et la construction de l'arbre associé dure elle trois secondes.

1. Les nœuds de degré 1 ont été supprimés dans ce graphe, ce qui explique la différence avec les données de l'article original, mais cela n'a pas d'incidence sur les calculs de communautés et donc sur les analyses faites ci-après.

En parallèle nous avons effectué les mêmes expériences sur des graphes aléatoires ayant la même distribution de degrés que nos graphes de terrain. Ces graphes ont été générés selon la méthode décrite dans [58]. On leur a ensuite enlevé les arêtes multiples et les boucles.

7.3 Analyse de l'arbre

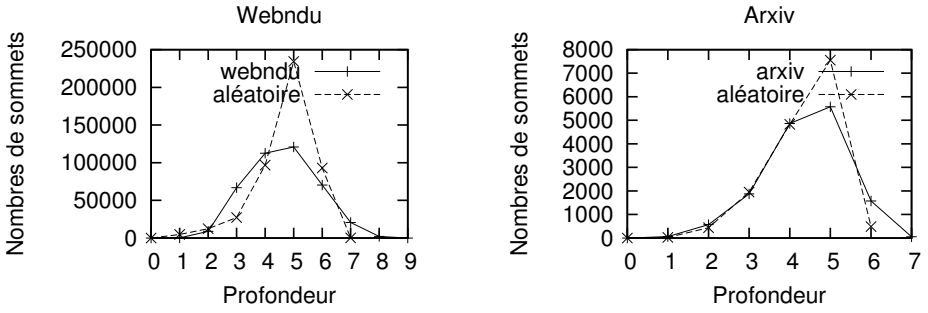


FIGURE 7.3 – Nombre de sommets (feuilles et sommets internes) dans l'arbre pour le graphe de terrain et le graphe aléatoire correspondant

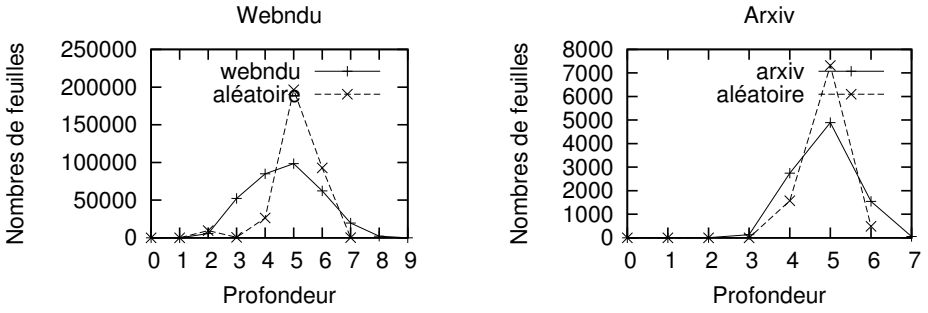


FIGURE 7.4 – Nombre de feuilles dans l'arbre pour le graphe de terrain et le graphe aléatoire correspondant

Le dendrogramme s'avère en général peu profond pour les graphes de terrain considérés. Bien que parmi nos graphes de test nous obtenons au maximum un arbre de profondeur 9, cette profondeur ne concerne que très peu de nœuds. La figure 7.3 représente le nombre de sommets (internes et feuilles) se trouvant à une profondeur donnée. On constate un rapide élargissement de l'arbre, laissant supposer que beaucoup des nœuds du graphe sont classés

bien avant la dernière décomposition. Si des feuilles sont à profondeur i, c'est qu'elles sont la terminaison de l'imbrication de $i-1$ communautés, la dernière n'étant pas décomposable. Les arbres contiennent des feuilles dès le niveau 2 ce qui signifie que même parmi les communautés du premier niveau, celles qui maximisent la modularité, il en existe qui ne peuvent pas être redécomposées. On constate sur chacun de nos graphes tests qu'approximativement 80 % des feuilles sont à un niveau inférieur ou égal à 5, soit qu'il y a en général moins de 4 niveaux d'imbrication de communautés pertinents.

Les arbres tirés des graphes aléatoires n'ont pas exactement le même comportement. Les feuilles apparaissent plus tard et sont plus localisées. Cela peut s'expliquer par le fait qu'il y ait moins de structure communautaire. En l'absence de groupes marqués, les nœuds se comportent plus uniformément et les groupes se cassent plus régulièrement. Les communautés cessent donc d'être décomposables au même moment car elles sont moins différentes initialement. Il y a néanmoins plusieurs niveaux avec des feuilles, et l'arbre reste assez déséquilibré.

La figure 7.5 présente la distribution du nombre de sous-communautés, c'est-à-dire la distribution du nombre de fils pour tous les sommets de l'arbre. La distribution est très hétérogène et cette disparité s'explique en grande partie par le fait que la taille des communautés, dont la distribution est représentée sur la figure 7.6, est aussi très hétérogène. Or, le nombre de sous-communautés d'une communauté est lié à sa taille comme l'indique la figure 7.7 qui représente la corrélation entre la taille d'une communauté et le nombre de sous-communautés.

Les graphes aléatoires ont ici un comportement assez similaire aux graphes de terrain. Le nombre de sous-communautés et leur taille semblent aussi hétérogènes.

Le dendrogramme des graphes de terrain est donc déséquilibré, mais ce déséquilibre est le reflet de la structure du graphe décomposé. Il semble n'y avoir qu'un petit nombre de niveaux d'imbrication de communautés, nombre qui dépend de la taille de la première communauté. Le dendogramme des graphes aléatoires est plus régulier mais reste déséquilibré.

7.4 La structure communautaire

Chaque niveau dans l'arbre définit aussi une partition du graphe initial. Ainsi, le premier niveau de l'arbre correspond aux communautés maximisant la modularité, le second niveau correspondant à l'ensemble des sous-communautés des précédentes, etc.

La figure 7.8 présente la modularité des partitions niveau par niveau et on constate qu'elle diminue régulièrement et assez vite passé le deuxième niveau, en raison notamment du problème de résolution limite évoqué précédemment

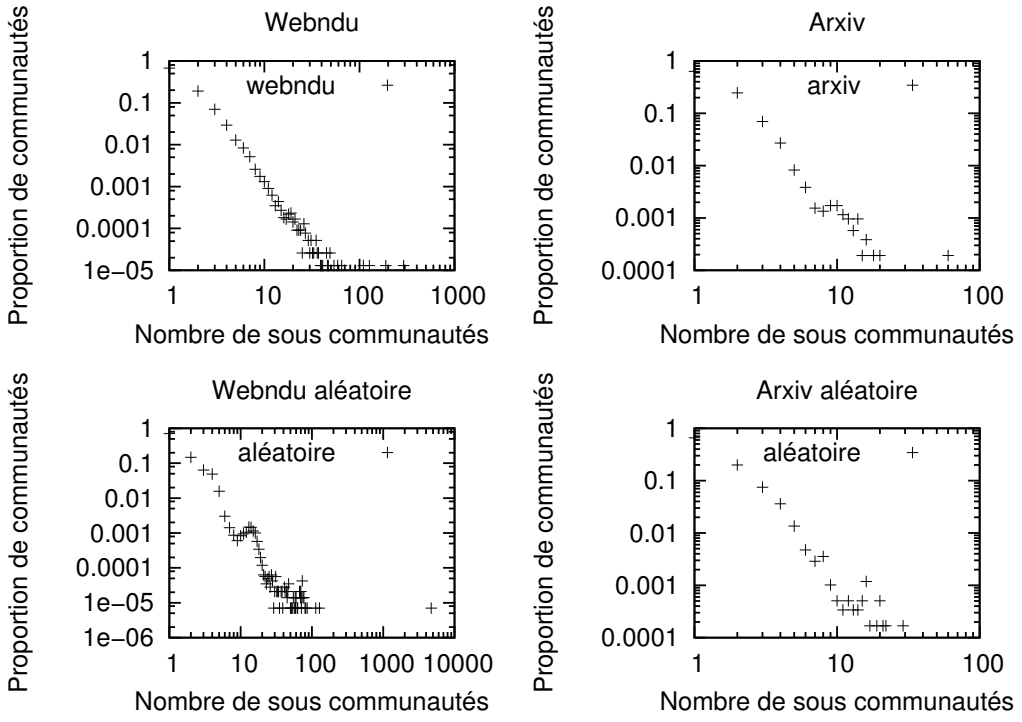


FIGURE 7.5 – Distribution du nombre de sous-communautés. Chaque point correspond en abscisse à un nombre de sous-communautés rencontré et en ordonnée à la proportion de communautés trouvées ayant ce nombre de sous-communautés. Tous les niveaux ont été agrégés

qui défavorise les petites communautés. Mais en plus de cela, d'autres problèmes se greffent :

- les nœuds isolés, qui ne font plus partie d'un regroupement mais sont les composants d'une communauté non décomposable, ont un effet strictement négatif sur la modularité car il n'y a alors aucun lien interne et que des liens sortants ;
- la modularité est aussi connue pour ne pas donner de bons scores aux partitions mélangeant des parties de tailles très différentes [34]. Par exemple, deux petites cliques à côté d'une grosse vont parfois être fusionnées pour avoir une modularité plus forte.

Cette modularité globale basse pour les partitions à partir du troisième niveau n'est donc pas forcément le signe qu'il n'y a plus de communautés significatives à partir de là, mais simplement que ces éventuelles communautés sont noyées dans la masse à cause de la résolution limite.

La modularité étant calculée en comparant une proportion de liens in-

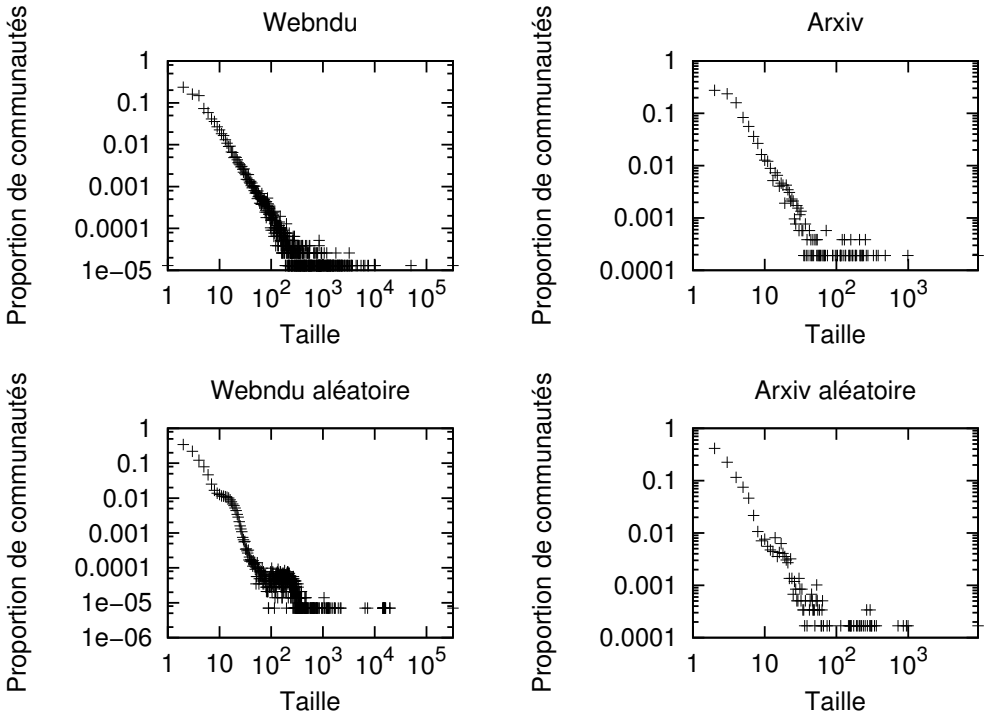


FIGURE 7.6 – Distribution des tailles des communautés. Chaque point correspond en abscisse à une taille rencontrée et en ordonnée à la proportion de communautés trouvées ayant cette taille. Tous les niveaux ont été agrégés

ternes à une communauté à ce que l'on aurait dans un graphe aléatoire de même taille, on espère que la modularité reste élevée et supérieure à celle d'un graphe aléatoire si le graphe est réellement modulaire. Si l'on étudie les graphes aléatoires, on constate que leur modularité part de bien moins haut, mais descend bien plus lentement signe d'une structure différente. Malgré la résolution limite, la modularité reste nettement supérieure durant plusieurs niveaux, ce qui montre que l'on continue à identifier des groupes. La décroissance de la modularité serait donc due à des conséquences de cette mesure plutôt qu'à l'absence d'identification de structure communautaire.

En analysant plus en détail les communautés à chaque niveau, on constate que seulement certaines communautés de tailles non négligeables ont une structure communautaire interne. En particulier, on peut considérer de manière indépendante chaque communauté de l'arbre puis la décomposer pour calculer sa modularité. La figure 7.9 représente la distribution cumulée de la modularité niveau par niveau. Chaque courbe correspond à un niveau dans l'arbre. On y lit en abscisse les valeurs prises par la modularité et en ordon-

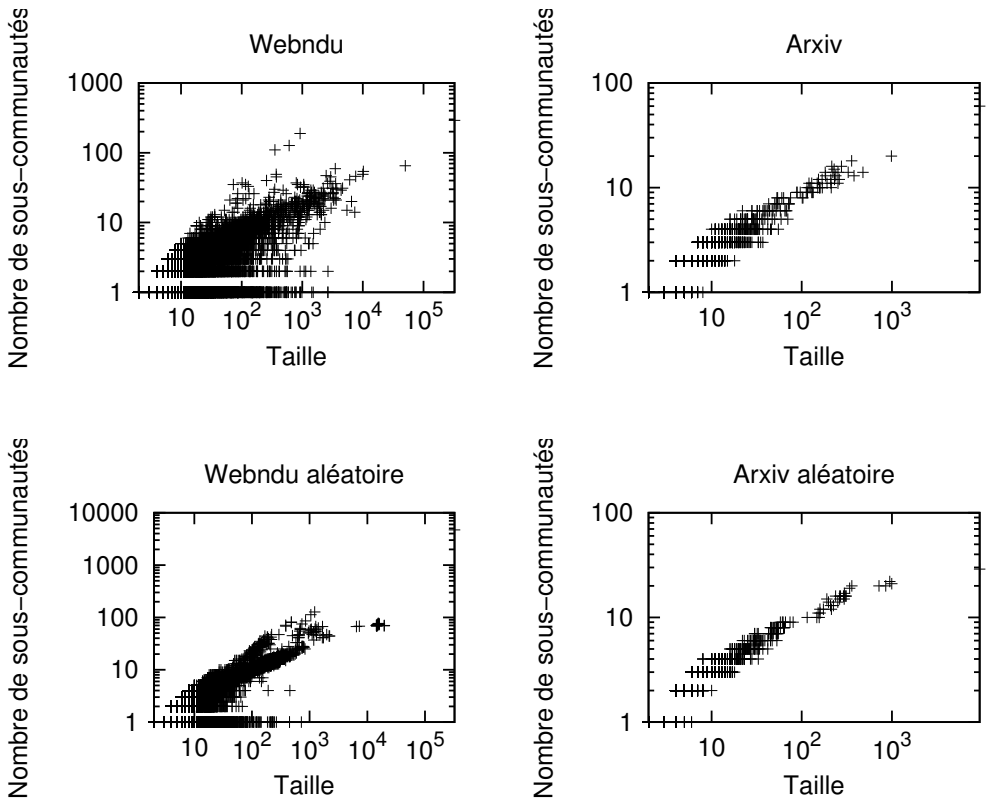


FIGURE 7.7 – corrélation taille-nombre de sous-communautés pour toutes les communautés de l'arbre. Les communautés qui ne peuvent pas être décomposées ont été exclues car leur taille est exactement égale à leur nombre de sous-communautés par définition

née la proportion de communautés qui sont décomposées en une partition de modularité inférieure : si moins de 40 % des communautés ont une modularité inférieure à $0,5$ au niveau 1, elles sont presque 70 % au niveau 2 et plus de 90 % au niveau 3. Ainsi, plus on s'enfonce dans l'arbre, plus la majorité des communautés ont une faible modularité, mais malgré tout, on trouve toujours quelques communautés avec une modularité élevée même dans les niveaux bas de l'arbre.

Si l'on regarde les graphes aléatoires, leur comportement est un peu différent. Vu leur taille, nous trouvons quand même des parties modulaires. Néanmoins, la variation est plus brutale : il y a moins de communautés ayant une modularité intermédiaire. La modularité décroît moins vite suivant les niveaux. Ainsi, sur le graphe aléatoire associé à webndu, plus de 98 % des communautés ont une modularité quasi nulle, et les autres ont une modularité proche de 1. Les graphes aléatoires ont donc bien une structure moins

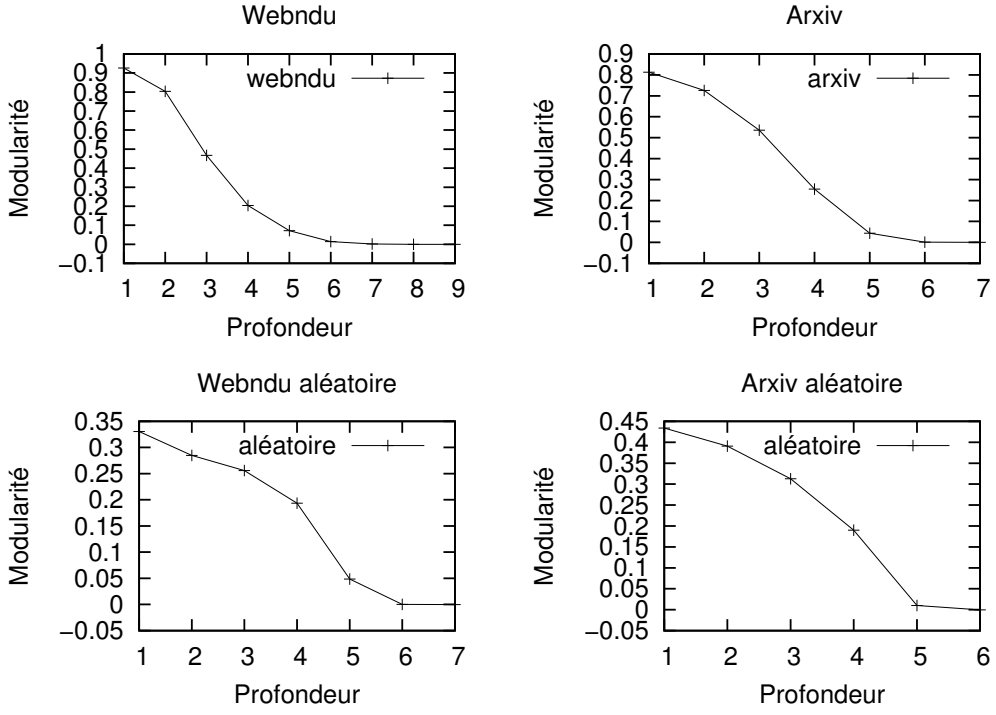


FIGURE 7.8 – Modularité de la décomposition globale niveau par niveau

modulaire, et ceci à chaque niveau. On trouve peu de communautés décomposables, et les décompositions trouvées sont très régulières et se retrouvent plus profondément.

Nous avons ensuite extrait une série de sous-graphes imbriqués en détectant à chaque étape les communautés et en sélectionnant à chaque fois la plus grande jusqu'à ce qu'il n'y en ait plus qu'une seule. La modularité obtenue à chaque étape est représentée sur la figure 7.10. On constate que la modularité reste élevée aux premiers niveaux mais décroît assez rapidement par la suite, alors que le sous-graphe associé reste de taille non négligeable comme indiqué sur la figure 7.11.

A titre comparatif, la modularité moyenne sur des graphes aléatoires de même distribution de degré pour chaque étape est indiquée sur la figure 7.10. La modularité des graphes de terrain reste nettement supérieure à celle des graphes aléatoires durant les premiers niveaux, ce qui montre qu'il y a bien une structure communautaire à plusieurs échelles. Par contre, ce n'est plus le cas pour les derniers niveaux où la modularité trouvée est similaire ou inférieure à ce que l'on trouve sur un graphe aléatoire, signe que nous n'arrivons pas à trouver dans ces sous-graphes des regroupements pertinents.

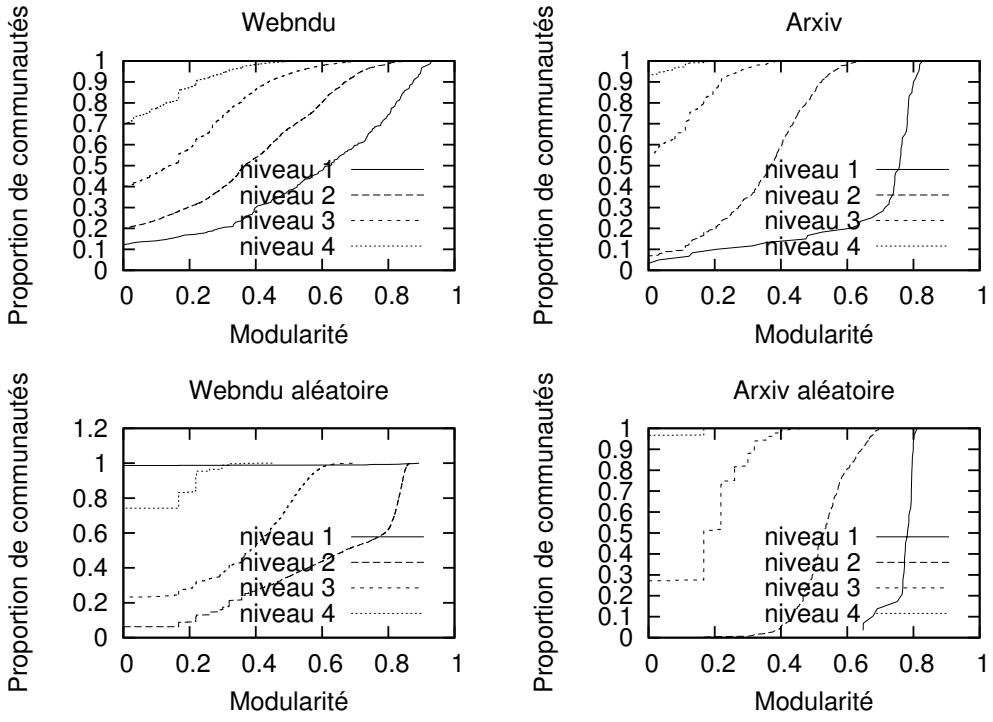


FIGURE 7.9 – Distribution cumulée relative de la modularité aux différents niveaux. Chaque courbe correspond à un niveau et passe par les points (x, y) tels qu'une proportion y de communautés peut se décomposer avec une modularité inférieure ou égale à x

Si l'on effectue la même expérience avec des graphes initiaux aléatoires, le comportement est très différent. Il y a moins de niveaux et nous n'arrivons pas à identifier des communautés significatives quel que soit le niveau. Cela montre qu'ils n'ont pas de structure multi-échelle identifiable. On constate que la modularité trouvée est assez élevée, ce qui montre que la comparaison à l'aléatoire est vraiment nécessaire pour identifier si les communautés trouvées ont du sens.

La comparaison peut aussi se faire directement grâce à des méthodes formelles permettant de prédire la modularité obtenue sur un graphe aléatoire [74]. Dans ce cas aussi, il est clair qu'un graphe qui n'est pas clairement plus modulaire qu'un graphe aléatoire de même taille n'est tout simplement pas modulaire.

Une méthode plus générale pour évaluer le score d'une partition est d'en prendre le z-score. C'est une normalisation habituelle ramenant la moyenne à zéro et l'écart type à un. Elle vaut :

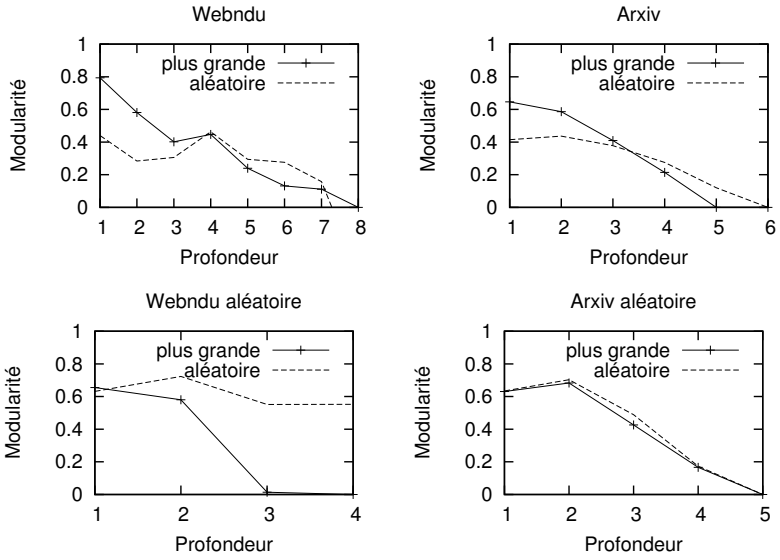


FIGURE 7.10 – Pour chaque graphe, modularité à l'intérieur de la plus grande communauté extraite récursivement. Pour chaque communauté extraite, un graphe aléatoire de même distribution de degré a été généré et sa modularité est représentée sur la même courbe

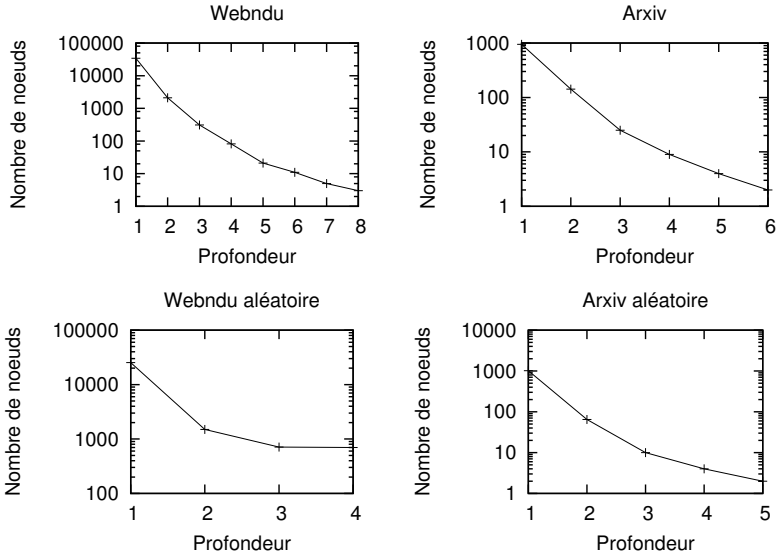


FIGURE 7.11 – Nombre de nœuds à l'intérieur de la plus grande communauté extraite récursivement

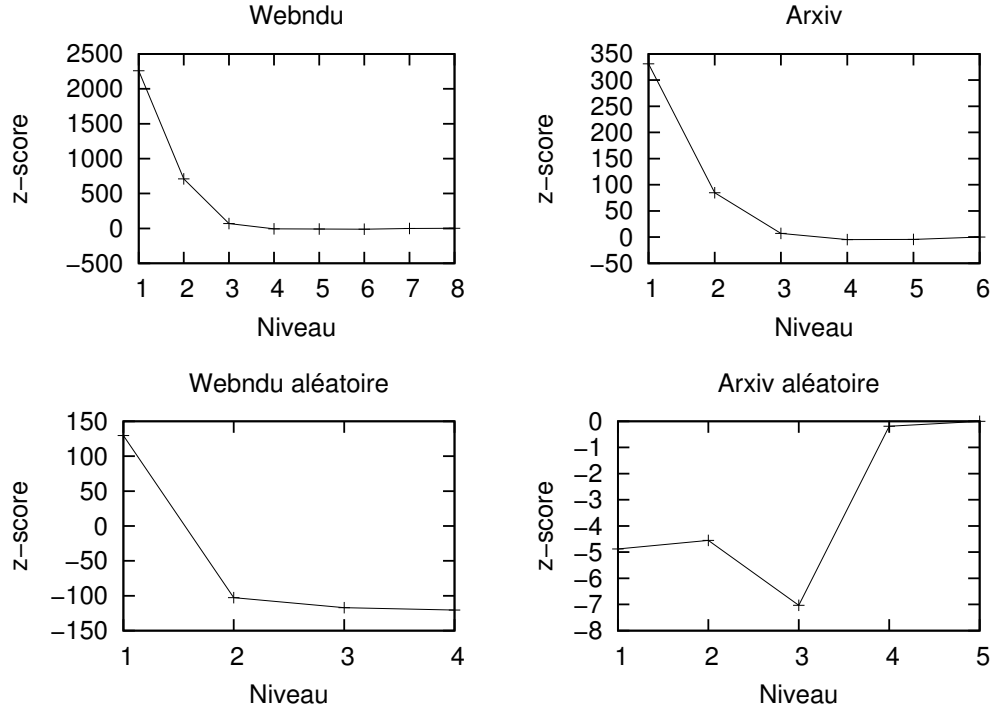


FIGURE 7.12 – z-score de la plus grande communauté extraite récursivement

$$z = \frac{Q - < Q >_{random}}{\sigma_{random}}$$

Avec Q la modularité considérée, $< Q >_{random}$ la modularité moyenne observée sur les graphes aléatoires équivalents et σ_{random} la déviation standard de cette valeur. Un graphe sera alors fortement modulaire si $z \gg 1$, des valeurs proches de 0 indiquent que la modularité du graphe est très proche de la valeur moyenne et n'est donc pas signe d'une structure fortement modulaire.

Les valeurs obtenues en approximant la moyenne sur 50 graphes aléatoires de même distribution de degrés sont représentées sur la figure 7.12. On confirme les constatations faites figure 7.10 : on trouve plusieurs niveaux significatifs avant de tomber dans des graphes pour lesquels on n'identifie plus de structure modulaire pertinente.

Ici encore, les graphes aléatoires n'ont pas la même structure. Le z-score du graphe de même distribution de degré qu'Arxiv est toujours proche de 0 ou négatif, indiquant qu'il n'y a pas de communautés identifiées. Celui du graphe aléatoire similaire à webndu s'effondre tout de suite après un premier niveau étonnamment élevé où les modularités sont très proches mais la déviation standard très faible. Les graphes aléatoires n'ont donc plus de communautés

identifiables immédiatement ou très vite après le premier niveau, alors que leur taille décroît elle beaucoup moins vite.

Pour conclure, il semblerait donc que l'on identifie bien une sous-structure communautaire dans les graphes de terrain. La modularité globale n'est pas très élevée, mais cette grandeur est limitée et n'est pas adaptée pour juger des communautés de tailles très variées. Il n'est néanmoins pas pertinent de prolonger trop profondément la détection de communautés car on décompose alors des graphes pour lesquels on n'identifie plus de structure communautaire. Le z-score permet de voir à quel moment les communautés trouvées cessent d'être significatives.

7.5 Conclusions et perspectives

Nous avons ici présenté quelques résultats sur la structure modulaire multi-échelle des grands graphes de terrain basés sur plusieurs graphes réels. Sur tous les graphes, les résultats sont similaires et mettent en évidence une structure multi-échelle dans laquelle les communautés peuvent effectivement être décomposées en sous-communautés qui ont du sens.

Cette structure multi-échelle est complexe et a de nombreuses propriétés très hétérogènes, notamment la taille des communautés ou le nombre de sous-communautés. De plus, dès les premiers niveaux, on trouve très rapidement des communautés non décomposables. L'étude de cet objet qui décrit le graphe de manière structurelle est donc à lui seul un problème intéressant.

Le problème de résolution limite exposé dans [34] incite à ne pas considérer uniquement la partition d'un graphe qui maximise la modularité car cette partition est généralement peu favorable aux communautés de petites tailles. Il est donc nécessaire d'explorer la structure hiérarchique complète des graphes étudiés. Au contraire, nous avons montré que s'il est possible de redécomposer de manière récursive les communautés, on arrive rapidement soit à des communautés non décomposables, soit à des communautés qui ne sont plus modulaires et n'ont donc pas de raison d'être décomposées. Il faut donc trouver un juste équilibre entre les problèmes de résolution limite et de décomposition excessive.

La technique utilisée pour mettre en évidence le problème de la décomposition excessive, à savoir la comparaison à des graphes aléatoires de même taille, peut certainement être utilisée comme critère d'arrêt supplémentaire lors de la décomposition : tout graphe trop peu modulaire ne doit plus être décomposé.

Il ne faut pas non plus perdre de vue qu'une structure hiérarchique est limitée. Il n'y a pas de possibilité de communautés recouvrantes : un nœud fait partie d'une et une seule série de communautés imbriquées. Si l'on considère deux communautés voisines, celles-ci sont susceptibles de contenir chacune une

partie d'une communauté à cheval entre elles, sans doute moins modulaire mais potentiellement intéressante. En extrayant une communauté, on perd toute cette information et on ne trouvera pas certaines sous-communautés. Intégrer le recouvrement d'une manière ou d'une autre dans le dendrogramme permettrait d'affiner la compréhension que l'on peut avoir de la structure du graphe.

Bibliographie

[1] R. Albert, H. Jeong, and A.-L. Barabasi. The diameter of the world wide web. *Nature*, 401 :130, 1999. (Cité en page 117.)

[2] R. Albert, H. Jeong, and A.-L. Barabasi. The diameter of the world wide web. *Nature*, 401 :130, 1999. (Cité en page 8.)

[3] A. Arenas, A. Fernandez, and S. Gomez. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10 :053039, 2008. (Cité en page 15.)

[4] S. Asur, S. Parthasarathy, and D. Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 921. ACM, 2007. (Cité en pages 22 et 35.)

[5] D. Auber, Y. Chiricota, F. Jourdan, and G. Melançon. Multiscale visualization of small world networks. In *Proc. IEEE Symposium on Information Visualization*, pages 75–81. Citeseer, 2003. (Cité en page 9.)

[6] T. Aynaud and J.-L. Guillaume. Static community detection algorithms for evolving networks. In *Wireless Networks*, volume 2010, pages 508–514, 2010. (Cité en page 80.)

[7] T. Aynaud and J.-L. Guillaume. Structure communautaire multi-échelle de grands graphes de terrain. *Techniques et sciences informatiques*, 30(2) :137–154, Feb. 2011. (Cité en page 9.)

[8] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks : membership, growth, and evolution. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 54, 2006. (Cité en pages 9 et 36.)

[9] M. Barthélemy and S. Fortunato. Resolution limit in community detection. *Proceedings of the National Academy of Sciences of the United States of America*, 104(1) :36–41, Jan. 2007. (Cité en pages 14, 15 et 103.)

[10] M. Beiró and J. Busch. Visualizing communities in dynamic networks. In *Latin American Workshop on Dynamic Networks*, volume 1, 2010. (Cité en page 23.)

[11] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech*, 10008 :1–12, 2008. (Cité en page 16.)

[12] M. Bommarito, D. Katz, and J. Zelner. On the Stability of Community Detection Algorithms on Longitudinal Citation Data. In *Proceedings of the 6th Conference on Applications of Social Network Analysis*, volume 1001, 2009. (Cité en page 32.)

[13] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2) :163–177, 2001. (Cité en page 58.)

[14] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, and D. Wagner. Maximizing Modularity is hard. *ArXiv Physics e-prints*, 2006. (Cité en pages 16 et 70.)

[15] U. Brandes and T. Erlebach. *Network Analysis : methodological foundations*. Springer Verlag, 2005. (Cité en page 14.)

[16] A. Browet, P. Absil, and P. Van Dooren. Community Detection for Hierarchical Image Segmentation. *Combinatorial Image Analysis*, pages 358–371, 2011. (Cité en page 102.)

[17] S.-Y. Chan, P. Hui, and K. Xu. Community Detection of Time-Varying Mobile Social Networks. *Complex Sciences*, pages 1154–1159, 2009. (Cité en page 27.)

[18] Z. Chen, K. a. Wilson, Y. Jin, W. Hendrix, and N. F. Samatova. Detecting and Tracking Community Dynamics in Evolutionary Networks. *2010 IEEE International Conference on Data Mining Workshops*, pages 318–327, Dec. 2010. (Cité en page 23.)

[19] Y. Chi, S. Zhu, X. Song, J. Tatemura, and B. L. Tseng. Structural and temporal analysis of the blogosphere through community factorization. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, page 163, 2007. (Cité en pages 9 et 25.)

[20] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70066111, 2004. (Cité en pages 16 et 17.)

[21] J.-P. Cointet and C. Roth. Socio-semantic Dynamics in a Blog Network. *2009 International Conference on Computational Science and Engineering*, (6) :114–121, 2009. (Cité en page 37.)

[22] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics : Theory and Experiment*, 2005(09) :P09008–P09008, Sept. 2005. (Cité en page 32.)

[23] M. Delest, J. Fedou, and G. Melançon. A quality measure for multi-level community structure. *Symbolic and Numeric Algorithms for Scientific*, 2006. (Cité en page 14.)

[24] T. Dinh, I. Shin, N. Thai, and M. Thai. A General Approach for Modules Identification in Evolving Networks. *Dynamics of Information*, 40(4) :83–100, 2010. (Cité en pages 30 et 31.)

[25] J. Epps and J. Bailey. Information Theoretic Measures for Clusterings Comparison : Is a Correction for Chance Necessary ? In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009. (Cité en pages 33 et 42.)

[26] T. S. Evans and R. Lambiotte. Line graphs, link partitions, and overlapping communities. *Physical Review E*, 80(1) :016105, July 2009. (Cité en page 13.)

[27] T. Falkowski, A. Barth, and M. Spiliopoulou. Studying community dynamics with an incremental graph mining algorithm. In *Proc. of the 14 th Americas Conference on Information Systems (AMCIS 2008)*, pages 1–11, 2008. (Cité en page 30.)

[28] T. Falkowski and M. Spiliopoulou. Users in volatile communities : Studying active participation and community evolution. *Lecture Notes in Computer Science*, 4511 :47, 2007. (Cité en page 24.)

[29] T. Falkowski, M. Spiliopoulou, and J. Bartelheimer. Community dynamics mining. In *Proceedings of 14th European Conference on Information Systems (ECIS 2006)*, Goteborg, 2006. Citeseer. (Cité en page 22.)

[30] C. Faloutsos, J. Sun, S. Papadimitriou, and P. Yu. Graphscope : parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 687–696. ACM New York, NY, USA, 2007. (Cité en page 35.)

[31] D. J. Fenn, M. A. Porter, M. McDonald, S. Williams, N. F. Johnson, and N. S. Jones. Dynamic communities in multichannel data. *arXiv*, 811 :1–4, 2008. (Cité en page 33.)

[32] D. J. Fenn, M. A. Porter, P. J. Mucha, M. McDonald, S. Williams, N. F. Johnson, and N. S. Jones. Dynamical Clustering of Exchange Rates. *Exchange Organizational Behavior Teaching Journal*, (21) :1–35, 2009. (Cité en page 33.)

[33] S. Fortunato. Community detection in graphs. *Physics Reports*, 2009. (Cité en page 16.)

[34] S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1) :36–41, 2007. (Cité en pages 120 et 127.)

[35] R. Görke, P. Maillard, and C. Staudt. Modularity-Driven Clustering of Dynamic Graphs. *Experimental Algorithms*, Cl(1), 2010. (Cité en page 30.)

[36] D. Greene and D. Doyle. Tracking the evolution of communities in dynamic social networks. In *Advances in Social Networks Analysis and Mining (ASONAM)*, volume 2010, pages 1–13. IEEE, 2010. (Cité en pages 22 et 29.)

[37] J.-l. Guillaume. Web Page, http ://jl-guillaume.free.fr/www/programs.php. (Cité en page 72.)

[38] A. Hamzaoui and M. Latapy. Detecting events in the dynamics of ego-centered measurements of the internet topology. In *Wireless Networks*, 2010. (Cité en page 92.)

[39] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Tracking evolving communities in large linked networks. In *National Academy of Sciences of the United States of America*, volume 101, page 5249. National Acad Sciences, 2004. (Cité en page 21.)

[40] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. *Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking - WDTN '05*, pages 244–251, 2005. (Cité en page 38.)

[41] M. B. Jdidia, C. Robardet, and E. Fleury. Communities detection and analysis of their dynamics in collaborative networks. In *ICDIM*, pages 744–749. IEEE, 2007. (Cité en page 29.)

[42] M. Kim and J. Han. A particle-and-density based evolutionary clustering method for dynamic networks. *Proceedings of the VLDB Endowment*, 2(1) :622–633, 2009. (Cité en pages 27 et 29.)

[43] J. Kleinberg. Bursty and Hierarchical Structure in Streams. *Data Mining and Knowledge Discovery*, 7(4) :373 – 397, 2003. (Cité en page 36.)

[44] H. Kuhn, P. Haas, I. Ilyas, G. Lohman, and V. Markl. The Hungarian method for the assignment problem. *Masthead*, 23(3) :151210, 1993. (Cité en page 44.)

[45] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. *World Wide Web*, 8(2) :159–178, 2005. (Cité en page 36.)

[46] R. Kumar, A. Tomkins, and D. Chakrabarti. Evolutionary clustering. In *Proc. of the 12th ACM SIGKDD Conference*, 2006. (Cité en page 26.)

[47] R. Lambiotte. Multi-scale modularity in complex networks. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2010 Proceedings of the 8th International Symposium on*, pages 546–553. IEEE, Apr. 2010. (Cité en pages 15, 51 et 106.)

[48] A. Lancichinetti and S. Fortunato. Limits of modularity maximization in community detection. *Networks*, pages 1–7. (Cité en page 16.)

[49] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 93046110, 2008. (Cité en page 29.)

[50] M. Latapy, C. Magnien, and F. Ouédraogo. A Radar for the Internet. In *2008 IEEE International Conference on Data Mining Workshops*, number 1, pages 901–908. IEEE, Dec. 2008. (Cité en page 38.)

[51] Linkedin. LinkedIn http ://inmaps.linkedin.com. (Cité en page 112.)

[52] S. Mancoridis, B. Mitchell, C. Rorres, Y. Chen, and E. Gansner. Using automatic clustering to produce high-level system organizations of source code. In *Proc. 6th Intl. Workshop on Program Comprehension*, pages 45–53, 1998. (Cité en page 9.)

[53] S. Milgram. The small world problem. *Psychology Today*, 1 :61, 1967. (Cité en page 8.)

[54] P. J. Mucha, T. Richardson, K. Macon, and M. A. Porter. Community structure in time-dependent, multiscale, and multiplex networks. *science*, 876 :10–13, 2010. (Cité en page 29.)

[55] M. E. J. Newman. The structure of scientific collaboration networks. Working Papers 00-07-037, Santa Fe Institute, July 2000. (Cité en page 117.)

[56] M. E. J. Newman. The structure of scientific collaboration networks. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 98, pages 404–9, 2000. (Cité en page 45.)

[57] M. E. J. Newman. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Physical Review E*, 64(1) :1–7, June 2001. (Cité en page 58.)

[58] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2) :167–256, 2003. (Cité en page 118.)

[59] M. E. J. Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1) :39–54, 2005. (Cité en page 58.)

[60] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 74 :36104, 2006. (Cité en page 16.)

[61] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2) :26113, 2004. (Cité en pages 14, 16, 24 et 29.)

[62] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. Huang. Incremental spectral clustering with application to monitoring of evolving blog communities. *In SIAM Int. Conf. on Data Mining*, 2007. (Cité en page 30.)

[63] M. Oliveira and J. Gama. Bipartite graphs for monitoring clusters transitions. *Advances in Intelligent Data Analysis IX*, M :114–124, 2010. (Cité en page 22.)

[64] M. Oliveira and J. Gama. Understanding Clusters Evolution. In *Workshop on Ubiquitous Data Mining*, volume D, pages 16 – 20, 2010. (Cité en page 22.)

[65] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking : Bringing order to the web. *In Technical report, Stanford Digital Library Technologies Project, Stanford University, Stanford, CA, USA*, 1998. (Cité en page 59.)

[66] G. Palla, A.-L. Barabasi, and T. Vicsek. Quantifying social group evolution. *Nature*, 446 :664–667, 2007. (Cité en pages 23, 33 et 37.)

[67] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Arxiv preprint physics/0506133*, (m) :1–10, 2005. (Cité en page 13.)

[68] J.-J. Pansiot, P. Mérindol, B. Donnet, and O. Bonaventure. Extracting Intra-Domain Topology from mrinfo Probing. In *Passive and Active Measurement*, 2009. (Cité en page 37.)

[69] D. Pfitzner, R. Leibbrandt, and D. Powers. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowledge and Information Systems*, 19(3) :361–394, July 2008. (Cité en page 32.)

[70] P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10 :191–218, 2006. (Cité en pages 16, 17 et 29.)

[71] W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336) :846–850, 1971. (Cité en page 32.)

[72] J. Reichardt and S. Bornholdt. Detecting fuzzy community structures in complex networks with a Potts model. *Physical Review Letters*, 93218701, 2004. (Cité en page 15.)

[73] J. Reichardt and S. Bornholdt. When are networks truly modular ? *Physica D Nonlinear Phenomena*, 224 :20–26, 2006. (Cité en page 14.)

[74] J. Reichardt and S. Bornholdt. When are networks truly modular ? *Physica D Nonlinear Phenomena*, 224 :20–26, Dec. 2006. (Cité en page 124.)

[75] M. Rosvall. Mapping change in large networks. *PLoS One*, pages 1–9, 2010. (Cité en page 36.)

[76] X. Song, Y. Chi, B. L. Tseng, D. Zhou, and K. Hino. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 153–162. ACM New York, NY, USA, 2007. (Cité en page 27.)

[77] 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 New York, NY, USA, 2006. (Cité en pages 21, 22, 24 et 34.)

[78] L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multi-mode networks. In *International Conference on Knowledge Discovery and Data Mining*, page 8, 2008. (Cité en page 27.)

[79] C. Tantipathananandh and T. Y. Berger-Wolf. Constant-factor approximation algorithms for identifying dynamic communities. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, volume Id, pages 827–836. ACM, 2009. (Cité en page 26.)

[80] C. Tantipathananandh, T. Y. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, page 717, 2007. (Cité en page 25.)

[81] M. Toyoda and M. Kitsuregawa. Extracting evolution of web communities from a series of web archives. In *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 28–37. ACM New York, NY, USA, 2003. (Cité en page 34.)

[82] A. L. Traud, E. D. Kelsic, P. J. Mucha, and M. A. Porter. Community Structure in Online Collegiate Social Networks. *North*, pages 1–15, 2008. (Cité en page 32.)

[83] B. L. Tseng, Y.-R. Lin, Y. Chi, S. Zhu, and H. Sundaram. Facetnet : a framework for analyzing communities and their evolutions in dynamic networks. *Social Networks*, pages 685–694, 2008. (Cité en page 28.)

[84] B. L. Tseng, Y.-R. Lin, Y. Chi, S. Zhu, and H. Sundaram. Analyzing communities and their evolutions in dynamic social networks. *ACM Transactions on Knowledge Discovery from Data*, 3(2) :1–31, 2009. (Cité en page 28.)

[85] M. L. Wallace, Y. Gingras, and R. Duhon. A new approach for detecting scientific specialties from raw cocitation networks. *J. Am. Soc. Inf. Sci. Technol.*, 60 :240–246, 2009. (Cité en page 9.)

[86] Q. Wang and E. Fleury. Detecting overlapping communities in graphs. In *European Conference on Complex Systems*, 2009. (Cité en pages 9 et 13.)

[87] Q. Wang and E. Fleury. Mining time-dependent communities. In *Latin American Workshop on Dynamic Networks*, 2010. (Cité en page 23.)

[88] Y. Wang, B. Wu, and N. Du. Community Evolution of Social Network : Feature, Algorithm and Model. *Science And Technology*, (60402011), 2008. (Cité en pages 22 et 23.)

[89] K. Xu, M. Kliger, and A. Hero. Tracking Communities in Dynamic Social Networks. *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 219–226, 2011. (Cité en page 27.)

[90] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin. A Bayesian Approach Toward Finding Communities and Their Evolutions in Dynamics Social Networks. In *Proceedings of the SIAM International Conference on Data Mining*, pages 990–1001. SIAM, 2009. (Cité en page 28.)

[91] W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33 :452–473, 1977. (Cité en page 29.)