Université Paris 7 – Denis Diderot
ufr d'informatique

Thèse de Doctorat
spécialité : Informatique

Détection de communautés dans les grands graphes de terrain

par

Pascal Pons

Thèse dirigée par Matthieu Latapy

soutenue le 20 Juillet 2007 devant le jury composé de :

M. Vincent Blondel, professeur, ucl, Louvain ..... rapporteur
M. Eric Fleury, professeur, insa, Lyon ..... rapporteur
M. Pierre Fraigniaud, directeur de recherches cnrs, Paris ..... examinateur
M. Patrick Gallinari, professeur, upmc, Paris ..... examinateur
M. Jens Gustedt, directeur de recherches inria, Nancy ..... examinateur
M. Michel Habib, professeur, Paris Diderot, Paris ..... examinateur
M. Matthieu Latapy, chargé de recherches cnrs, Paris ..... directeur
M. Michel Morvan, professeur, ens, Lyon ..... rapporteur

Remerciements

Trois ans de thèse, au début cette échéance paraît fort lointaine. Au final, on s'aperçoit que l'on n'a réussi à faire que le dixième de toute ce que l'on aurait souhaité faire. À chaque fois que l'on arrive enfin à déverrouiller un porte, on découvre immanquablement derrière elle dix nouvelles portes fermées. Ceci atteste sans aucun doute des excellentes conditions dans lesquelles se sont déroulés mes travaux de recherche. Je souhaite remercier tous ceux et toutes celles qui y ont contribué au cours de ces trois années.

En premier lieu, je remercie Matthieu Latapy pour m'avoir fait découvrir un domaine de recherche excitant, entre théorie et applications, à l'intersection de nombreuses disciplines. Il a toujours été disponible lorsque j'avais besoin d'un avis ou de conseils et a su diriger mes recherches tout en me laissant un grande part de liberté.

Cette thèse n'aurait pas pu être menée à terme correctement sans un aménagement de ma scolarité, effectuée en parallèle, au corps des télécommunications. Je remercie ceux qui m'ont permis de réaliser ce double cursus dans de bonnes conditions et en particulier Godeffroy Beauvallet pour sa grande souplesse. J'ai eu la chance de pouvoir effectuer mon stage d'ingénieur à France Télécom R&D, stage durant lequel j'ai pu appliquer une partie de mes travaux de thèse. Je remercie Dominique Cardon et Sébastien Bertrand de m'avoir accueilli pendant six mois durant lesquels ce fut un grand plaisir de travailler avec Christophe Prieur, Jean-Samuel Beuscart et Nicolas Pissard.

J'ai eu le plaisir de participer aux groupes de recherche PERSI puis AUTOGRAPH qui m'ont permis d'ouvrir mes travaux (et mon esprit) à d'autres disciplines. Je remercie notamment Julien Levrel, Céline Poudat et Nicolas Auray pour les travaux que nous avons effectués en collaboration.

Je remercie les rapporteurs de cette thèse, Vincent Blondel, Eric Fleury et Michel Morvan, qui m'ont accordé leur temps et leur attention. Il en va de même pour Pierre Fraigniaud, Patrick Gallinari, Jens Gustedt et Michel Habib qui ont accepté de faire partie de mon jury.

Outre mon directeur de thèse, je tiens particulièrement à remercier Clémence Magnien pour ses commentaires et remarques sur les versions préliminaires de mes articles et de cette thèse. Je tiens aussi à remercie Fabien Viger, Olivier Levillain et mon épouse qui ont aussi contribué à divers travaux de relecture.

De bonnes conditions de travail commencent sans aucun doute par une bonne ambiance de bureau. Je remercie donc les divers occupants du bureau 6A51 pour leur convivialité et leur bonne humeur. Nommons dans le désordre Fabien, Vincent, Mohssen, Benjamin, Jean-Loup, Mahendra, Michel, Frédéric, Philippe, Philippe, le lutin et le frigo.

Je n'oublie surtout pas de remercier mon épouse Élodie, qui a su me supporter et m'encourager dans les phases de rédaction. Et bien sûr ma fille Juliette qui m'a laissé le temps d'achever la rédaction de cette thèse de justesse avant de venir au monde et de perturber grandement notre rythme circadien.

Table des matières

1 Introduction **9**

1.1 Les grands graphes de terrain . . . 9

1.1.1 De nombreux domaines concernés . . . 10

1.1.2 Des caractéristiques communes . . . 11

1.1.3 Les enjeux algorithmiques . . . 13

1.2 La détection de communautés . . . 14

1.2.1 Définition formelle du problème . . . 15

1.2.2 Intérêt et applications . . . 15

1.3 État de l'art . . . 16

1.3.1 Les approches classiques . . . 17

1.3.2 Les approches séparatives . . . 19

1.3.3 Les approches agglomératives . . . 20

1.3.4 Les approches utilisant des marches aléatoires . . . 21

1.3.5 Les autres approches . . . 22

1.4 Travaux effectués et organisation de la thèse . . . 23

2 Graphes et marches aléatoires **25**

2.1 Définitions . . . 25

2.1.1 Graphes . . . 25

2.1.2 Marches aléatoires . . . 27

2.2 Propriétés . . . 28

2.3 Calcul des marches aléatoires . . . 31

2.3.1 Calcul exact pour des graphes denses . . . 31

2.3.2 Calcul exact pour des graphes peu denses . . . 32

2.3.3 Calcul approché par simulation . . . 32

2.4 Graphes bipartis et marches aléatoires . . . 33

2.4.1 Définition et contexte des graphes bipartis . . . 33

2.4.2 Projection d'un graphe biparti . . . 33

2.4.3 Marches aléatoires et projections de graphes bipartis . . . 34

2.5 Marches aléatoire en temps continu . . . 36

2.5.1 Définition du processus . . . 36

2.5.2 Calcul des marches aléatoires en temps continu . . . 37

2.6 Cas des graphes orientés . . . 38

3 Détection de communautés **41**
3.1 Marches aléatoires et similarité des sommets . . . 41
3.1.1 Définition d'une distance entre sommets . . . 41
3.1.2 Propriétés spectrales de la distance r . . . 43
3.1.3 Liens avec les autres approches spectrales . . . 45
3.1.4 Choix de la longueur t des marches . . . 46
3.1.5 Généralisations de la distance . . . 47
3.2 L'algorithme de clustering hiérarchique . . . 50
3.2.1 L'algorithme de clustering hiérarchique . . . 50
3.2.2 Complexité . . . 55
3.2.3 Améliorations et optimisations . . . 58

4 Partitions et coupes multi-échelles **61**
4.1 Problématique . . . 61
4.2 Fonctions de qualité . . . 62
4.2.1 Exemples de fonctions de qualité . . . 63
4.2.2 Fonctions de qualité additives . . . 66
4.3 Optimisation d'une fonction de qualité additive . . . 67
4.4 Détection multi-échelles de communautés . . . 69
4.4.1 Fonctions de qualité multi-échelles . . . 69
4.4.2 Meilleure partition pour toutes les échelles . . . 72
4.5 Détermination des échelles pertinentes . . . 76

5 Analyse expérimentale **79**
5.1 Protocole d'évaluation expérimentale . . . 79
5.1.1 Génération des graphes de test . . . 79
5.1.2 Évaluation de la qualité d'une partition . . . 81
5.2 Comparaison des différents algorithmes . . . 83
5.2.1 Comparaison sur des graphes homogènes . . . 84
5.2.2 Comparaison sur des graphes hétérogènes . . . 88
5.2.3 Comparaison sur des graphes réels . . . 90
5.3 Évaluation de la détection multi-échelles . . . 92
5.3.1 Évaluation des différentes fonctions de qualité . . . 92
5.3.2 L'apport des fonctions de qualité multi-échelles . . . 94
5.4 Influence de la longueur des marches aléatoires . . . 97

6 Conclusion **101**

Table des notations

Notion	Description	Section
A	Matrice d'adjacence d'un graphe	2.1.1
$\mathcal{C}$	Une communauté, c'est à dire un ensemble de sommets	
D	Matrice diagonale des poids des sommets ($D_{ii} = w(i)$)	2.1.1
E	Ensemble des arêtes d'un graphe	2.1.1
G	Un graphe G = (V, E, w)	2.1.1
λ_α	Valeurs propres de la matrice de transition P	2.2
m	Nombre d'arêtes du graphe ($m = \|E\|$)	2.1.1
n	Nombre de sommets du graphe ($n = \|V\|$)	2.1.1
P	Matrice des probabilités de transition entre sommets	2.1.2
P_{ij}^t	Probabilité d'aller du sommet i au sommet j en t étapes par une marche aléatoire	2.1.2
$P_{i\bullet}^t$	Vecteur colonne de probabilité de position ($P_{i\bullet}^t(j) = P_{ij}^t$)	3.1.1
$P_{\mathcal{C}\bullet}^t$	Vecteur colonne de probabilité de position ($P_{\mathcal{C}\bullet}^t = \frac{1}{\|\mathcal{C}\|}\sum_{i\in\mathcal{C}} P_{i\bullet}^t$)	3.1.5
$\mathcal{P}^k$	$k^{\text{ième}}$ partition créée lors du déroulement de notre algorithme	3.2.1
$\mathcal{P}_\alpha$	Partition $\mathcal{P} \in \Pi$ maximisant Q_α pour le facteur d'échelle α	4.4.1
π	Distribution limite de probabilité de position ($\lim_{t\to+\infty} P_{ij}^t = \pi_j = \frac{w(j)}{\sum_{k\in V} w(k)}$)	2.2
Π	Ensemble des partitions de V possibles à partir de l'ensemble des communautés $\mathcal{S}$ d'un dendrogramme	4.1
Q^M, Q^P, Q^S	Principales fonctions de qualité	4.2.1
$Q_\alpha^M, Q_\alpha^P, Q_\alpha^S$	Fonction de qualité multi-échelles correspondantes	4.4.1
$R(\alpha)$	Fonction donnant la pertinence du facteur d'échelle α	4.5
r_{ij}	Distance entre les sommets i et j, $r_{ij} = \left\|D^{-\frac{1}{2}}P_{i\bullet}^t - D^{-\frac{1}{2}}P_{j\bullet}^t\right\|$	3.1.1
$\rho(t)$	Vecteur de probabilité de position d'une marche aléatoire	2.1.2
$\bar{\rho}(t)$	Vecteur de probabilité de position d'une marche aléatoire en temps continu ($t \in \mathbb{R}^+$)	2.1.2
$\mathcal{S}$	Ensemble des communautés définies par un dendrogramme	4.1
σ_k	Hétérogénéité de la partition $\mathcal{P}^k$: $\sigma_k = \frac{1}{n}\sum_{\mathcal{C}\in\mathcal{P}_k}\sum_{i\in\mathcal{C}} r_{i\mathcal{C}}^2$	3.2.1
$\Delta\sigma(\mathcal{C}_1, \mathcal{C}_2)$	Variation de σ_k en cas de fusion des communautés $\mathcal{C}_1$ et $\mathcal{C}_2$	3.2.1
u_α	Vecteurs propres à gauche de la matrice de transition P	2.2
v_α	Vecteurs propres à droite de la matrice de transition P	2.2
V	Ensemble des sommets d'un graphe	2.1.1
w_{ij}	Poids de l'arête $\{i, j\} \in E$	2.1.1

Chapitre 1

Introduction

De récentes avancées dans le domaine des systèmes complexes ont fait ressortir le rôle central que jouent les graphes dans de nombreux phénomènes. Ces grands graphes permettent de modéliser les interactions entre les différents acteurs de ces phénomènes complexes, qui interviennent dans de très nombreux domaines : sociologie, biologie, linguistique, physique, informatique, épidémiologie, etc. De manière surprenante ces graphes, appelés *grands graphes de terrain* ou *complex networks*, possèdent des propriétés structurelles non triviales communes qui ont fait l'objet de nombreuses et récentes études [83, 77, 1, 57, 21]. Ces propriétés communes permettent d'envisager une nouvelle algorithmique des grands graphes de terrain. Les avancées qui en découlent ont des applications potentielles dans les nombreux domaines concernés.

Les travaux de cette thèse s'inscrivent dans ce contexte interdisciplinaire, en se concentrant sur la question algorithmique de détection de communautés, c'est-à-dire de groupes d'acteurs fortement liés entre eux et faiblement liés aux autres.

Nous allons d'abord présenter les grands graphes de terrain et leurs applications dans les différents domaines concernés. Nous introduirons ensuite la problématique de détection de communautés. Un état de l'art des travaux sur le sujet sera proposé et mis en parallèle avec les travaux effectués dans cette thèse.

> Afin de faciliter la lecture de cette thèse, nous proposerons à la fin de chaque section une synthèse facilement repérable par son cadre. Ces résumés peuvent être avantageusement utilisés en première lecture pour acquérir une vision rapide des points essentiels développés dans le texte.

1.1 Les grands graphes de terrain

Un graphe $G = (V, E)$ permet de modéliser des interactions entre des objets (représentés par un ensemble V de n sommets) reliés par des liens (représentés par un ensemble $E \subset V \times V$ de m paires de sommets nommées arêtes). Ainsi des *grands graphes de terrain* permettent de modéliser de nombreux phénomènes réels (le terme grand fait référence au nombre important de sommets de ces graphes). Ces graphes peuvent être, selon le contexte, orientés ou non-orientés, pondérés ou non-pondérés. Nous définirons formellement ces objets mathématiques au Chapitre 2, cependant nous nous permettrons d'utiliser par avance dans cette introduction les concepts simples et classiques de sommet, d'arête, de degré d'un sommet (nombre de sommets

qui lui sont liés), de chemin (succession de sommets liés par des arêtes) et de composante connexe (ensemble maximal de sommets reliés par des chemins). Nous encourageons le lecteur non familier avec ces notions à se référer au Chapitre 2 directement.

Les grands graphes de terrain permettent de modéliser de nombreux phénomènes provenant d'horizons très variés. L'appellation *graphe de terrain*, proposée initialement par Bruno Gaume dans [30], n'est pas universellement acceptée en France et l'on peut aussi rencontrer par exemple les termes *réseau d'interactions* ou *réseau complexe*. Ces appellations font référence aux mêmes objets qui possèdent la dénomination anglo-saxonne bien reconnue de *complex network*. Ils se caractérisent par le fait que des interactions simples et facilement compréhensibles entre sommets à l'échelle microscopique produisent des propriétés topologiques globales et des comportements macroscopiques non-triviaux difficilement interprétables. Du point de vue de l'algorithmique ou de la théorie des graphes, l'originalité du sujet tient au fait que les graphes, outre leur grandes tailles, sont obtenus empiriquement sur la base de données réelles (et non définis formellement).

Nous allons présenter quelques exemples dans lesquels ces graphes jouent un rôle majeur. Nous allons ensuite recenser leurs caractéristiques communes qui permettent d'envisager des algorithmes génériques qui peuvent s'appliquer à l'ensemble des domaines présentés. Le domaine des grands graphes de terrain a fait l'objet de plusieurs travaux de synthèse provenant de différentes disciplines [12, 82, 77, 1, 57, 21] auxquels nous renvoyons le lecteur qui souhaite obtenir une bibliographie exhaustive du sujet.

1.1.1 De nombreux domaines concernés

Nous allons maintenant présenter quelques exemples dans lesquels des graphes sont utilisés comme outil de modélisation de phénomènes complexes. Ces exemples vont illustrer la diversité des domaines d'applications possibles. Pour chaque cas, nous identifierons les acteurs du phénomène, modélisés par les sommets du graphe, et les interactions entre eux, modélisées par des liens ou arêtes entre les sommets. L'ensemble de ces domaines fait l'objet d'une littérature très abondante, nous nous contenterons de donner des pointeurs vers certaines publications clés. Une bibliographie plus complète peut être obtenue dans [82, 77, 1, 57, 21].

- Les réseaux sociaux. Ils constituent un champ d'application ancien et important [82] dans lequel les acteurs sont des individus ou entités sociales (associations, entreprises, pays, etc). Les liens entre eux peuvent être de différentes natures. Nous pouvons ainsi observer plusieurs types de réseaux : les réseaux de connaissance (deux individus sont reliés s'ils se connaissent), les réseaux de contact physique (deux individus sont reliés s'ils ont été physiquement en contact), les réseaux de collaboration (deux individus sont reliés s'ils ont travaillé ensemble, en particulier de nombreux travaux ont étudié les collaborations scientifiques [60]), les réseaux d'appels téléphoniques [72] (deux individus ou numéros de téléphones sont reliés s'il y a eu un appel entre eux), les réseaux d'échanges (deux entités sont reliées si elles ont échangé un fichier [34] ou un courrier électronique [23] par exemple), etc.
- Les réseaux biologiques. Il en existe plusieurs types parmi lesquels nous pouvons citer les réseaux métaboliques [45] (les sommets sont des gènes ou des protéines qui sont liés par leurs interactions chimiques), les réseaux de neurones (chaque neurone est connecté à plusieurs autres neurones) ou les réseaux trophiques [84] (les espèces d'un écosystème sont reliées pour représenter les chaînes alimentaire).
- Les réseaux d'infrastructure. Ils représentent des connections matérielles entre objets

distribués dans un espace géographique. Nous pouvons citer les réseaux de transport (routes entre villes ou liaisons aériennes entre aéroports), les réseaux de distribution électrique (câbles entre les lieux de production et de consommation) ou encore le réseau physique de l'Internet [26] (câbles entre ordinateurs).
- Les réseaux d'information. Ils représentent des liens abstraits de référencement entre des supports d'information. Parmi eux, les réseaux de citation d'articles ou les graphes du Web [49] (les sommets sont des pages Web liées par des liens hypertextes).
- Les réseaux linguistiques. Ils relient les mots d'un langage donné et regroupent entre autres les réseaux des synonymie (deux mots sont reliés s'ils sont synonymes), les réseaux de co-occurrences [42] (deux mots sont reliés s'ils apparaissent dans une même phrase d'un ouvrage) ou encore les réseaux de dictionnaires [10] (deux mots sont liés si l'un est utilisé dans la définition de l'autre).

Les problématiques d'étude peuvent être très variées selon les disciplines. Par exemple on étudie la propagation des épidémies [62] grâce à un réseau social de contact physique modélisant les possibilités de contamination. Un opérateur de télécommunications peut vouloir établir les profils de ses clients en analysant le réseau des appels téléphoniques. Les réseaux métaboliques sont utilisés pour comprendre le fonctionnement de la cellule [45]. L'analyse des réseaux d'infrastructure permet de détecter leurs points faibles susceptibles d'être la cible d'attaques ou de pannes ayant des conséquences majeures [2, 14]. Les moteurs de recherche du Web se basent sur des graphes du Web pour référencer de manière pertinente les pages Web.

Nous allons montrer dans les sections suivantes que ces objets issus d'horizons différents possèdent toutefois des caractéristiques communes et soulèvent des problématiques similaires, ce qui permet d'envisager des méthodes algorithmiques générales.

> Les grands graphes de terrain sont utilisés pour modéliser de nombreux phénomènes dans des contextes très variés. Citons en particulier les réseaux sociaux, les réseaux biologiques, les réseaux d'infrastructure, les réseaux d'information et les réseaux linguistiques.

1.1.2 Des caractéristiques communes

Les grands graphes de terrain que l'on peut rencontrer dans les différentes disciplines que nous venons de citer n'ont, à première vue, pas de raison de se ressembler. Cependant, plusieurs études ont révélé l'existence de caractéristiques structurelles communes et non triviales [83, 77, 1, 57, 21]. Nous présenterons dans cette section un survol de ces caractéristiques communes.

Tout d'abord du point de vue de la taille, le terme "grand" graphe de terrain fait référence au nombre de sommets qui peut aller de quelques centaines à plusieurs milliards (par exemple pour les graphes du Web ou la structure neuronale du cerveau). Cependant chaque sommet est lié à relativement peu d'autres sommets : une des caractéristiques des grands graphes de terrain est de posséder un *faible degré moyen* d_{moy} (ou nombre moyen de voisins d'un sommet dans le graphe : $d_{moy} = \frac{2m}{n}$). Cette faible moyenne correspond généralement à des limites individuelles de chaque acteur du réseau. De nombreuses études et modèles visent à considérer que les grands graphes de terrain possèdent un degré moyen borné (indépendamment de la taille du graphe). Ceci implique en particulier que le nombre d'arêtes d'un grand

graphe de terrain peut être considéré comme linéaire par rapport au nombre de sommets du graphe ($m = \mathcal{O}(n)$) ; on parle alors de graphes creux ou peu denses.

Bien que le degré moyen soit relativement faible par rapport à la taille du graphe, *la distribution des degrés est en général très hétérogène* : il existe dans les grands graphes de terrain un nombre non négligeable de sommets possédant un très fort degré par rapport à une majorité de sommets possédant un très faible degré. Cette distribution est souvent bien approximée par une loi de puissance pour laquelle le nombre P_k de sommets de degré k est proportionnelle à $k^{-\alpha}$, avec des puissances α mesurées entre 2 et 3 pour la plupart des réseaux réels. Ceci a engendré l'appellation *réseaux sans-échelle* (ou *scale-free networks*), issue de la physique, pour certains grands graphes de terrain. Cette caractéristique implique qu'il existe quelques rares (mais non exceptionnels) sommets de très fort degré qui jouent nécessairement des rôles particuliers par rapport aux autres sommets. Cette propriété est l'une des premières propriétés surprenantes qui rapprochent les topologies des grands graphes de terrain.

La première propriété sur le degré moyen borné implique que *la densité des grands graphes de terrain est faible*. Cette densité δ est la proportion d'arêtes existantes par rapport au nombre total d'arêtes possibles $\delta = \frac{2m}{n(n-1)}$. En effet, comme nous l'avons vu, le nombre d'arêtes de ces graphes croît linéairement par rapport au nombre de sommets alors que le nombre d'arêtes possibles croît de manière quadratique. Ceci implique que la densité globale tend à être très faible lorsque la taille du graphe croît. Cependant *une des propriétés essentielles des grands graphes de terrain, qui va jouer un rôle essentiel dans cette thèse, est l'existence d'une forte densité locale qui s'oppose à la faible densité globale du graphe.* Cette forte densité traduit le fait que les liens entre les sommets qui sont proches (possédant des voisins communs par exemple) sont beaucoup plus probables que les liens entre sommets éloignés. Elle s'explique par la tendance des acteurs à se regrouper en modules ou *communautés*. Cette densité est souvent capturée par le coefficient de clustering qui compte la probabilité que deux voisins d'un même sommet soient eux-mêmes liés par une arête.

La différence entre forte densité locale et faible densité globale fonde la problématique de détection de communautés que nous allons développer dans cette thèse. En effet cette différence de densité entre les niveaux microscopiques et macroscopiques des grands graphes de terrain est largement expliquée par la présence de groupes de sommets (communautés) fortement liés entre eux et faiblement liés avec l'extérieur.

Une dernière caractéristique attribuée aux grands graphes de terrain est celle de *graphes petits-mondes* (en référence à l'effet *small-word* de l'expérience de Milgram [78]). Celle-ci traduit le fait qu'il existe des très courts chemins pour relier chaque paire de sommets (un chemin étant une succession de sommets reliés par des arêtes).

Cette dernière propriété, bien qu'intuitivement surprenante, peut en réalité s'expliquer de manière statistique. En effet elle est vérifiée par le modèle de graphes aléatoires d'Erdős-Rényi [24]. Ce modèle construit un graphe en plaçant aléatoirement des arêtes entre les paires de sommets, la probabilité (identique pour chaque paire de sommets) fixant la densité du graphe aléatoire construit. Au-delà, tout graphe auquel est ajouté une faible quantité de liens aléatoires voit sa distance moyenne devenir très faible [50, 83]. La propriété "petits-mondes" peut donc être considérée comme une propriété mécanique découlant de la présence d'aléatoire plutôt qu'une propriété caractéristique des graphes de terrain. Les trois premières propriétés (faible densité globale, loi de puissance, forte densité locale) sont bien des propriétés caractéristiques que l'on ne retrouve pas dans les graphes aléatoires. Un axe important de recherche, sur lequel nous ne nous attarderons pas dans cette thèse, s'attache d'ailleurs à construire des modèles de graphes permettant de reproduire ces propriétés (voir [8, 55, 37, 83]

par exemple).

> Les grands graphes de terrain rencontrés dans différentes disciplines possèdent des caractéristiques communes non-triviales. Ils possèdent en particulier un faible degré moyen, une forte hétérogénéité des degrés, des chemins courts entre tous les sommets et une faible densité globale couplée à une forte densité locale. Cette dernière caractéristique, souvent mesurée par le coefficient de clustering, est essentielle pour la problématique développée dans cette thèse car elle traduit l'existence de zones denses faiblement interconnectées appelées communautés.

1.1.3 Les enjeux algorithmiques

Il existe de nombreuses problématiques de recherche autour des grands graphes de terrain. En effet, chaque discipline possède ses propres interrogations sur ses objets d'étude qui sont modélisés par ces graphes. Cependant les grandes tailles des données manipulées rendent impossible tout traitement manuel et demandent une automatisation faisant intervenir des algorithmes de traitement adéquats. Les grandes tailles des données représentent une fois de plus une forte limitation sur le choix des algorithmes utilisables. En effet, dans ce contexte, il est difficilement envisageable de faire fonctionner un algorithme dont la complexité soit plus que quadratique en la taille du graphe. Une telle complexité permet de manipuler des graphes de quelques dizaines de milliers de sommets ; au delà il est préférable de rechercher des méthodes de traitement linéaires. Cette forte contrainte limite grandement le choix algorithmique pour répondre aux problématiques des différentes disciplines.

Plutôt que de résoudre séparément chaque problème spécifique concernant les grands graphes de terrain, nous sommes convaincus de l'utilité d'une nouvelle algorithmique spécialement adaptée à ce domaine en général. Les caractéristiques communes présentées dans la section précédente permettent en effet d'envisager des méthodes de traitement et d'analyse générales qui tireront parti des spécificités communes de ces grands graphes. Dans cette nouvelle visée, l'algorithmicien devra proposer des outils et méthodes d'analyse et de traitement de graphes qui devront s'adapter aux contraintes des grands graphes de terrain, notamment à la contrainte de complexité. Nous devons alors tirer profit de leurs caractéristiques pour proposer des algorithmes génériques qui serviront de pièces de bases pour la construction de méthodes de traitement plus complexes adaptées à chaque problématique particulière. es premiers travaux transverses commencent à apparaître dans cette direction [11, 52].

Un autre enjeu important provient de la qualité des données. En effet l'ampleur et la complexité de la collecte des données entraînent l'apparition fréquente d'erreurs dans les graphes que l'on a à traiter. Il peut s'agir d'informations manquantes que l'on n'a pas pu mesurer ou même des informations erronées qui sont généralement des conséquences indésirables de la méthode de mesure mise en œuvre. Idéalement, un algorithme devra s'adapter aux inévitables erreurs présentes dans les données ou même tenir compte des erreurs de mesure prévisibles. De manière générale, nous devons chercher à mesurer les conséquences des erreurs présentes dans les données sur les résultats calculés. Il émerge ainsi une nouvelle problématique de métrologie [36, 35] des grands graphes et des conséquences de la qualité des données. Cette problématique, que nous n'aborderons pas dans cette thèse et peu traitée jusqu'à présent, possède une grande importance pratique et ouvre de nombreuses perspectives de recherche.

Les caractéristiques communes des grands graphes de terrain amènent à envisager une nouvelle algorithmique qui leur soit spécialement adaptée. Elle proposera des outils et méthodes d'analyse et de traitement qui serviront de pièces élémentaires pour la résolution des problématiques de chaque domaine d'application. Ces algorithmes devront tirer profit des caractéristiques spécifiques des grands graphes de terrain pour répondre aux contraintes que posent la taille et la qualité des données.

1.2 La détection de communautés

L'existence dans les grands graphes de terrain de zones plus densément connectées que d'autres constitue une des caractéristiques non triviales que l'on retrouve dans de nombreux cas. Ces zones sont appelées *communautés* (par analogie avec les réseaux sociaux) et correspondent intuitivement à des groupes de sommets plus fortement connectés entre eux qu'avec les autres sommets, comme illustré dans la Figure 1.1. Afin de disposer d'un cadre rigoureux, nous allons donner une définition du problème basée sur l'optimisation de fonctions de qualité. Nous discuterons ensuite de quelques enjeux pratiques de la détection de communautés.

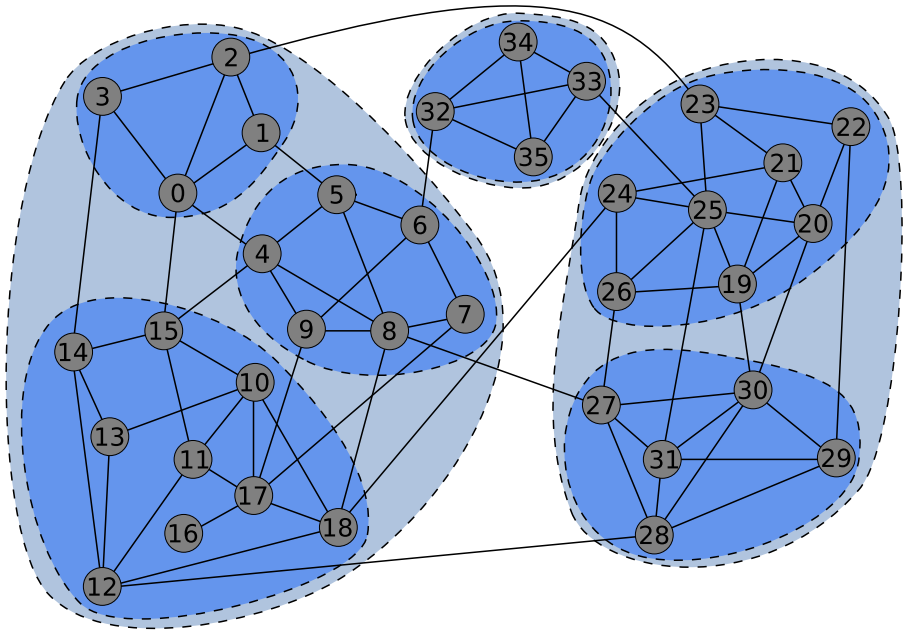

FIG. 1.1 – Exemple de structures de communautés dans un graphe : deux partitions en communautés correspondant à deux échelles différentes sont représentées.

1.2.1 Définition formelle du problème

On se donne une fonction Q, appelée *fonction de qualité*, qui à toute partition $\mathcal{P}$ de V associe un indice de sa qualité du point de vue de la notion de communauté considérée. Le but de la détection de communautés dans un graphe $G = (V, E)$ est de trouver une partition $\mathcal{P} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ de l'ensemble V des sommets telle que $Q(\mathcal{P})$ est maximale. Les communautés $\mathcal{C}_i$ sont donc disjointes ($\forall i \neq j, \mathcal{C}_i \cap \mathcal{C}_j = \emptyset$) et recouvrent V ($\cup_i \mathcal{C}_i = V$). Notons que nous ne connaissons a priori ni le nombre ni la taille des communautés qui composent la partition recherchée.

Comme le montre cette définition, on n'a pas a priori de définition de la notion de communauté : tout repose sur la fonction de qualité Q qui joue donc un rôle clé. La fonction de qualité donne un score à toute partition qui mesure à quel point les communautés de cette partition vérifient des critères caractéristiques des communautés. En particulier, ces fonctions des qualité $Q(\mathcal{P})$ tiennent généralement compte des densités de liens à l'intérieur et entre les communautés. Il existe plusieurs fonctions de qualité pour la détection de communautés, la plus communément utilisée étant la *modularité* [59]. Il n'existe toutefois pas de consensus sur ce point, et nous consacrerons donc la Section 4.2 à la présentation de différentes fonctions de qualité existantes.

L'optimisation exacte d'une fonction de qualité sur l'ensemble de toutes les partitions est généralement un problème NP-difficile. Notre approche (comme toutes les autres) aura donc pour objectif de trouver en une complexité raisonnable une partition donnant la meilleure valeur possible pour la fonction de qualité choisie.

Notons enfin que cette définition impose que les communautés $\mathcal{C}_i$ sont disjointes, or en pratique rien n'empêche l'existence de communautés qui se chevauchent. La détection de telles communautés est un problème bien plus difficile que la détection de communautés disjointes. Ce problème a pour le moment été très peu traité et sera abordé à la fin de cette thèse, le reste de la thèse étant consacré au problème de détection de communautés disjointes.

> Le but de la détection de communautés est de trouver une partition $\mathcal{P} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ des sommets du graphe, dans laquelle chaque sous-ensemble $\mathcal{C}_i$ représente une communauté (nous ne connaissons a priori ni le nombre ni la taille des communautés). Le choix de la partition est fait de manière à optimiser une fonction de qualité $Q(\mathcal{P})$ qui mesure des critères caractéristiques des communautés. Les fonctions de qualité classiques seront présentées en 4.2.

1.2.2 Intérêt et applications

La détection de communautés entre dans la problématique générale de l'algorithmique des grands graphes de terrain présentée en Section 1.1.3. En effet l'existence de telles structures constitue une caractéristique commune indépendante du domaine d'origine du graphe. La détection de communautés peut donc être vue comme un outil générique qui peut s'appliquer à tous les grands graphes de terrain rencontrés. Les intérêts principaux de tels algorithmes sont doubles : d'une part ils permettent de mieux comprendre la structure de ces graphes et d'autre part ils peuvent être utilisés comme briques de base dans l'élaboration d'autres traitements plus complexes (parallélisation, visualisation ou compression par exemple).

Notons que les communautés peuvent avoir des interprétations différentes suivant le type de réseau considéré. Ainsi dans les réseaux sociaux elles correspondent à des ensembles d'individus possédant des points communs et dont les liens sociaux sont naturellement plus forts. De manière totalement différente, dans les réseaux métaboliques les communautés correspondent à des fonctions biologiques de la cellule [70, 38]. Pour les réseaux d'information, elles peuvent correspondre à des thématiques. Par exemple les pages Web traitant d'un même sujet se référencent mutuellement et la détection de communautés dans le graphe du Web est une piste envisagée pour améliorer les moteurs de recherche [28]. De manière générale, la détection de communautés est un outil important pour la compréhension des structures et des fonctionnements des grands graphes de terrain.

En effet, il est souvent impossible d'appréhender la structure d'un graphe en ne connaissant que ses propriétés locales, c'est à dire des propriétés à l'échelle des sommets et de leurs voisins directs. Les communautés permettent de donner un point de vue macroscopique sur la structure des graphes. Elle permettent par exemple de regrouper et d'identifier les sommets qui jouent potentiellement des rôles similaires. Ce genre d'analyse peut se rapprocher des problématiques de *data-mining* lorsque les données peuvent être représentées par un graphe.

Parallèlement, la détection de communautés peut jouer le rôle de brique élémentaire dans la constitution d'algorithmes plus complexes. En effet la grande taille des graphes considérés est une limite majeure en terme de complexité des algorithmes utilisables. Dans certain cas, utiliser les communautés pour diviser le graphe permet d'effectuer des calculs séparés moins coûteux sur chaque communauté. Ce procédé de parallélisation de calcul permet d'envisager des gains de complexité pour les algorithmes qui s'y prêtent. La détection de communautés peut aussi être utilisée pour la visualisation des grands graphes de terrain [6]. On peut par exemple envisager des visualisations multi-échelles des communautés permettant d'apprécier les structures du graphe à différentes échelles. Enfin, des tentatives ont récemment été faites pour améliorer les méthodes de compression de graphes [54] qui sont d'une grande importance pour l'étude des grands graphes de terrain.

> La détection de communautés permet d'une part de comprendre les structures et les fonctionnements macroscopiques des grands graphes de terrain. D'autre part une telle méthode peut être utilisée comme brique de base pour l'élaboration d'algorithmes plus complexes comme la parallélisation, la visualisation ou la compression.

1.3 État de l'art

Nous allons, dans cette section, faire un état de l'art relatif à la détection de communautés. Il s'agit d'une problématique proche des problématiques classiques de clustering de données et de partitionnement de graphes. Une orientation explicite vers l'analyse des grands graphes de terrain n'a été introduite qu'en 2002 par Girvan et Newman [33]. Outre le nouveau domaine d'application que constitue les grands graphes de terrain, la différence principale dans la problématique réside dans le fait que l'on ne connaît pas à l'avance le nombre de communautés que l'on cherche. Depuis cet article qui fait référence dans le domaine, le sujet a reçu une extraordinaire attention de la part de la communauté scientifique et de très nombreuses nouvelles approches sont sans cesse proposées. La difficulté réside dans le fait de minimiser

les coûts (en temps et en espace) du calcul tout en maximisant la qualité des partitions en communautés trouvées. Suivant les contextes, différents compromis sont acceptables, ce qui justifie en partie la prolifération de méthodes.

Nous allons lister ici les principales approches qui ont été proposées à ce jour. Bien que la liste soit importante, elle est non exhaustive : afin d'en limiter la longueur, nous n'avons retenu que les approches qui ont reçu le plus d'attention de la part de la communauté scientifique. Notre but est de donner une vue d'ensemble des méthodes proposées, et d'en illustrer la diversité. Pour organiser la présentation, nous avons essayé de les regrouper en fonction du type d'approche utilisée. Tout d'abord les méthodes itératives (séparatives ou agglomératives) qui divisent ou fusionnent successivement des communautés représentent les approches les plus répandues. Nous allons aussi nous intéresser aux méthodes utilisant des marches aléatoires, qui jouent un rôle important dans cette thèse car nous allons nous-même fonder notre approche sur les marches aléatoires. Nous finirons par présenter diverses autres approches qui ne rentrent pas dans les deux premières catégories.

1.3.1 Les approches classiques

La détection de communautés s'approche des deux thématiques classiques en informatique que sont le partitionnement de graphe et le clustering de données. La première, initialement introduite pour la parallélisation de processus, cherche à répartir des tâches, représentées par les sommets d'un graphe, tout en minimisant les échanges, représentés par les arêtes. La seconde thématique de clustering de données est une thématique générale plus vaste dans laquelle on cherche à regrouper des données possédant des caractères communs. Cette problématique possède des applications dans de très nombreux domaines et on peut considérer la détection de communautés comme une de ses applications pour les grands graphes de terrain.

Le partitionnement de graphe

Le but du partitionnement de graphe est de grouper les sommets d'un graphe en un nombre prédéterminé de parties (et de tailles elles aussi prédéterminées) tout en minimisant le nombre d'arêtes tombant entre les différents groupes. Cette approche ne convient pas totalement à la détection de communautés car elle a l'inconvénient de requérir une connaissance préalable du nombre de communautés recherchées ainsi que de leurs tailles. Nous citons les deux méthodes générales ayant eu le plus de succès.

La méthode de bissection spectrale [27, 67]. Elle consiste à calculer le vecteur propre correspondant à la plus petite valeur propre non nulle de la matrice Laplacienne du graphe, $L = D - A$ (selon les notations qui seront introduites au Chapitre 2). Le graphe est alors séparé en deux parties en fonction du signe de leur composante selon ce vecteur propre. Cette méthode fonctionne en temps $\mathcal{O}(n^3)$ et obtient de bons résultats lorsque le graphe possède effectivement deux grandes communautés de tailles similaires, ce qui n'est qu'un cas particulier dans l'optique de détection de communautés.

La méthode de Kernighan et Lin [48]. Il s'agit d'un algorithme de bissection visant à trouver la coupe du graphe minimisant le nombre d'arêtes tombant entre les deux groupes. L'algorithme nécessite comme paramètre la taille des communautés à détecter. Une coupe de la bonne taille est choisie arbitrairement comme point de départ en utilisant une heuristique

gloutonne. La bissection est améliorée itérativement en faisant des échanges de sommets entre les communautés. A chaque étape on échange les deux sommets procurant la meilleure réduction du nombre d'arêtes externes, avec la condition imposée de ne jamais changer deux fois un même sommet. Il y a donc exactement $\frac{n}{2}$ étapes de calcul et la meilleure partition rencontrée au cours du déroulement de l'algorithme est retenue. La complexité du pire cas est en $\mathcal{O}(n^3)$.

Le clustering de données

Cette problématique a été introduite pour analyser des données [4, 25]. Elle cherche à partitionner des données en se basant sur une mesure de distance (ou de similarité) entre celles-ci. Cette distance représente la similarité des données et on cherche à former des groupes d'objets proches les uns des autres. Nous renvoyons le lecteur à l'article de synthèse [43] comme point d'entrée dans ce très vaste sujet. Cette problématique peut être modélisée par des graphes pondérés : chaque donnée est un sommet et les arêtes sont pondérées par la distance entre ces données.

Le problème de détection de communautés peut être vu comme un problème de clustering de données pour lequel il faut choisir une distance adéquate. Cependant, les graphes considérés par les applications usuelles de clustering ne possèdent pas les caractéristiques spécifiques des grands graphes de terrain. Par conséquent de nombreuses approches classiques de clustering de données sont inadaptées aux grands graphes de terrain, principalement pour des questions de taille. En pratique seules les méthodes de clustering hiérarchique ont permis d'obtenir de bons résultats. Nous allons donc présenter ces méthodes, qui ont été adaptées pour la détection de communautés par plusieurs approches récentes, dont la nôtre.

Les méthodes de clustering hiérarchique. L'algorithme part d'une structure dans laquelle chaque sommet est identifié comme une petite communauté. On itère alors les étapes suivantes : on calcule des distances entre communautés et on fusionne les deux communautés les plus proches en une nouvelle communauté. Le nombre de communautés est réduit de un à chaque étape, et le processus s'arrête lorsqu'il n'y a plus qu'une seule communauté correspondant au graphe entier. On obtient ainsi une structure hiérarchique de communautés qui peut être représentée sous une forme arborescente appelée dendrogramme (nous en donnerons une illustration en Figure 3.2, page 52) : les feuilles sont les sommets du graphe tandis que les noeuds représentent les communautés créées et sont reliés en fonction des fusions de communautés. La racine de la structure correspond au graphe entier.

Il existe plusieurs façons de définir la distance entre deux communautés. La plus simple (*single linkage*) considère que la distance entre deux communautés est la distance minimale entre deux sommets de celles-ci. A l'opposé, on peut considérer la distance maximale (*complete linkage*). De manière intermédiaire (*average linkage*) on peut considérer que la distance entre deux communautés est la moyenne des distances entre chaque paire de leurs sommets. Il est encore possible dans certains cas de représenter chaque communauté par un sommet moyen et de considérer leurs distances respectives (*centroïd*). La dernière méthode (dite de *Ward*) [81], que nous utiliserons et détaillerons en Section 3.2.1, consiste à minimiser la somme des carrés des distances de chaque sommet au sommet moyen représentant sa communauté.

1.3.2 Les approches séparatives

L'idée commune à toutes ces méthodes est d'essayer de scinder le graphe en plusieurs communautés en retirant progressivement les arêtes reliant des communautés distinctes. Les arêtes sont retirées une à une, et à chaque étape les composantes connexes du graphe obtenu sont identifiées à des communautés. Le processus est répété jusqu'au retrait de toutes les arêtes. On obtient alors une structure hiérarchique de communautés (dendrogramme), comme pour les méthodes de clustering hiérarchique. Les méthodes existantes diffèrent par la façon de choisir les arêtes à retirer.

L'algorithme de Girvan et Newman basé sur la centralité d'intermédiarité [33, 59]. Cette approche retire les arêtes de plus forte centralité d'intermédiarité. Cette centralité est définie pour une arête comme le nombre de plus courts chemins passant par cette arête. Il existe en effet peu d'arêtes reliant les différentes communautés et les plus courts chemins entre deux sommets de deux communautés différentes ont de grandes chances de passer par ces arêtes. Un algorithme calculant la centralité de toutes les arêtes en $\mathcal{O}(mn)$ est proposé. Ce calcul est effectué à chaque étape sur le graphe obtenu après retrait des arêtes. La complexité de l'algorithme est donc $\mathcal{O}(m^2n)$. Une variante considérant des marches aléatoires à la place des plus courts chemins est aussi introduite. Elle donne des résultats légèrement meilleurs mais demande plus de calculs.

Il est à noter que le même algorithme de calcul de la centralité d'intermédiarité des arêtes a aussi été introduit en même temps par Brandes [11].

Les approches de Radicchi *et al* [68] et d'Auber *et al* [6] basées sur le clustering d'arêtes. La détection des arêtes intercommunautaires est ici basée sur le fait que de telles arêtes sont dans des zones peu clusterisées. Radicchi *et al* [68] proposent un coefficient de clustering (d'ordre g) d'arêtes. Il est défini comme étant le nombre de cycles de longueur g passant par l'arête divisé par le nombre total de tels cycles possibles (étant donné les degrés des extrémités de l'arête). Cet algorithme retire donc à chaque étape l'arête de plus faible clustering (d'un ordre donné, 3 ou 4 en pratique). Chaque suppression d'arête ne demande alors qu'une mise à jour locale des coefficients de clustering, ce qui lui permet d'être bien plus rapide que le précédent algorithme. La complexité totale est en $\mathcal{O}(m^2)$. L'approche d'Auber *et al* [6] utilise un clustering d'arêtes d'ordre 3 pour proposer un outils de visualisation de graphes.

L'algorithme de Fortunato *et al* basé sur la centralité d'information [29]. Il s'agit d'une variante de l'algorithme de Girvan et Newman basé sur une autre notion de centralité : la centralité d'information. Les auteurs définissent l'efficacité de communication ϵ_{ij} entre deux sommets i et j du graphe comme étant l'inverse leur distance (en terme de plus courts chemins). L'efficacité E du réseau est alors définie comme la moyenne des ϵ_{ij} pour tous les couples de sommets. Enfin, la centralité d'information d'une arête est définie comme étant la diminution relative de l'efficacité du réseau lorsque l'on retire cette arête du graphe. Cette approche donne de meilleurs résultats que l'approche de Girvan et Newman mais a une complexité en $\mathcal{O}(m^3n)$.

1.3.3 Les approches agglomératives

L'idée commune de toutes ces méthodes est d'utiliser une approche, s'apparentant à celle du clustering hiérarchique, dans laquelle les sommets sont regroupés itérativement en communautés en partant d'une partition de n communautés composées d'un seul sommet. Les regroupements de communautés sont poursuivis jusqu'à obtenir une seule communauté regroupant tous les sommets, et une structure hiérarchique de communautés (dendrogramme) est ainsi construite. Ces approches sont similaires aux approches séparatives à la différence qu'elles travaillent de bas en haut dans la hiérarchie des communautés au lieu de travailler de haut en bas.

L'algorithme d'optimisation de la modularité proposé par Newman [58] et amélioré par Clauset *et al* [16]. Newman introduit une notion de modularité ; il s'agit d'une fonction Q mesurant la qualité d'une partition du graphe en communautés. Nous présentons en détail cette fonction en Section 4.2.1 (elle se base sur les proportions d'arêtes internes aux communautés et les proportions d'arêtes liées à chaque communauté). Afin de maximiser cette quantité l'algorithme glouton proposé fusionne à chaque étape les communautés permettant d'avoir la plus grande augmentation de la modularité.

Pour améliorer les performances de l'algorithme, seules les communautés ayant une arête entre elles peuvent être fusionnées à chaque étape. Chaque fusion se fait en $\mathcal{O}(n)$ et la mise à jour des valeurs des variations de Q (pour chaque nouvelle fusion possible) peut être effectuée facilement en $\mathcal{O}(m)$. La complexité globale est donc $\mathcal{O}((m+n)n) = \mathcal{O}(mn)$. Cette méthode est très rapide et permet de traiter de très grands graphes. La qualité des partitions obtenues est cependant moins bonne qu'avec des méthodes plus coûteuses.

La complexité de cette méthode a été améliorée par Clauset [16] en utilisant une structure de données adaptée. Une autre optimisation de cette méthode a récemment été proposée par Wakita et Tsurumi [80] afin de traiter des graphes de taille encore plus importante.

L'algorithme de Donetti et Muñoz basé sur les propriétés spectrales de la matrice Laplacienne du graphe [19, 20]. Les propriétés spectrales de la matrice Laplacienne du graphe, $L = D - A$ (avec les notations qui seront introduites au Chapitre 2) sont utilisées pour détecter des communautés. En effet les coordonnées i et j des vecteurs propres correspondant aux plus petites valeurs propres non nulles sont corrélées lorsque les sommets i et j sont dans la même communauté. Une distance entre sommets est alors calculée à partir de ces vecteurs propres, cette distance étant ensuite utilisée dans un algorithme de clustering hiérarchique. Le nombre de vecteurs propres à considérer est a priori inconnu. Plusieurs calculs sont successivement effectués en prenant en compte différents nombres de vecteurs propres, et le meilleur résultat est retenu. Les performances de l'algorithme sont limitées par les calculs des vecteurs propres qui se fait en $\mathcal{O}(n^3)$ pour une matrice creuse. Une amélioration de cette approche a été proposée et utilise une version normalisée de la matrice Laplacienne [20]. Notons que les vecteurs propres de cette nouvelle matrice sont les mêmes vecteurs propres que ceux de la matrice de transition des marches aléatoires que nous allons utiliser dans notre approche (Chapitre 3).

1.3.4 Les approches utilisant des marches aléatoires

Les marches aléatoires dans les graphes sont des processus aléatoires dans lesquels un marcheur est positionné sur un sommet du graphe et peut à chaque étape se déplacer vers un des sommets voisins. Ce processus sera étudié en détail au Chapitre 2. Le comportement des marches aléatoires est étroitement lié à la structure du graphe, ainsi plusieurs approches de détection de communautés se basent sur ces comportements. Nous avons souhaité distinguer les approches utilisant les marches aléatoires pour les mettre en parallèle de l'approche que nous proposerons qui est elle aussi basée sur des marches aléatoires. Notons que beaucoup des algorithmes présentés dans cette section sont aussi des algorithmes agglomératifs qui auraient pu être classés dans la section précédente.

Markov Cluster Algorithm de van Dongen [79]. Cette approche calcule des probabilités de transition entre tous les sommets du graphe en partant de la matrice de transition des marches aléatoires. Deux opérations matricielles sont successivement itérées. La première calcule les probabilités de transition par des marches aléatoires de longueur fixée r et correspond à une élévation de la matrice à la puissance r. La seconde consiste à amplifier les différences en augmentant les transitions les plus probables et en diminuant les transitions les moins probables. Les transitions entre sommets d'une même communauté sont alors favorisées et les itérations successives des deux opérations conduisent à une situation limite dans laquelle seules les transitions entre sommets d'une même communauté sont possibles. La complexité totale de l'algorithme est en $\mathcal{O}(n^3)$

L'approche de Harel et Koren [39]. Cette approche itère une opération de séparation sur les arêtes du graphe qui amplifie le poids des arêtes intérieures aux communautés et atténue celui des arêtes entre les communautés. Le poids des arêtes est modifié par un opérateur en fonction de la proximité de ses deux extrémités. Pour mesurer cette proximité, deux méthodes sont proposées. La première compare les vecteurs de probabilité de position pour des marches aléatoires courtes (cette méthode s'approche de la distance entre sommets que nous allons introduire dans notre approche). La seconde méthode utilise la probabilité d'échappement d'une marche aléatoire, c'est-à-dire la probabilité pour un marcheur d'atteindre un sommet d'arrivée sans repasser par le sommet de départ. Chaque itération du premier opérateur est faite en $\mathcal{O}(mn)$, le second opérateur nécessite la résolution d'un système d'équations linéaires en $\mathcal{O}(n^3)$. Les communautés peuvent être déduites en utilisant un seuil ou en appliquant un algorithme de clustering hiérarchique sur le graphe modifié.

Deux approches basées sur le temps moyen pour atteindre un sommet [88, 86]. Pour mesurer la proximité structurelle de deux sommets, ces approches utilisent le nombre moyen d'étapes pour qu'une marche aléatoire atteigne un sommet donné en partant d'un autre sommet. Cette longueur moyenne peut être calculée grâce à une inversion de matrice en $\mathcal{O}(n^3)$. Les deux approches, l'une proposée par Zhou et Lipowsky [88] et l'autre proposée par Yen *et al* [86] utilisent cette quantité pour définir deux distances mesurant la similarité des sommets. Cette distance est utilisée dans la première approche dans un méthode de clustering hiérarchique pour constituer un algorithme de détection de communautés nommé *netwalk*. La seconde approche utilise leur distance dans un algorithme de clustering *k-mean*.

1.3.5 Les autres approches

Une méthode d'optimisation par recuit simulé proposée par Guimerà et Amaral [38]. Ces travaux appliqués à la cartographie de réseaux métaboliques utilisent une approche directe d'optimisation de la modularité par recuit simulé. Cette méthode envisage des modifications aléatoires d'une partition, les probabilités de transition étant définies en fonction de l'amélioration de la modularité. Les transformations de partitions envisagées sont des fusions et divisions de communautés ainsi que des déplacements de sommets d'une communauté vers une autre.

Une méthode d'optimisation proposée par Duch et Arenas [22]. Cette méthode consiste à diviser le graphe en deux communautés et à diviser récursivement les deux communautés ainsi trouvées. Chaque étape part d'une division arbitraire et des sommets sont ensuite changés de communauté de façon à améliorer la modularité (mesurée comme une somme de participations élémentaires pour chaque sommet). Le sommet à déplacer est choisi aléatoirement (selon des probabilités adéquates) parmi les sommets ayant les moins bonnes contributions à la modularité globale de la coupe. La coupe obtenant la meilleure modularité est retenue tout au long du processus.

Cosmoweb de Bennouas, Bouklit et de Montgolfier [9]. Cette approche originellement proposée pour clusteriser des graphes du Web est basée sur une simulation gravitationnelle dans laquelle les pages Web liées s'attirent mutuellement dans un espace métrique. Les objets évoluent dans l'espace et s'agrègent en amas sous l'effet des forces d'attraction. Après quelques itérations, un seuil sur les distances est choisi pour déterminer les communautés.

Une méthode d'optimisation proposée par Reichardt et Bornholdt [71]. Cette approche est basée sur un modèle physique de Potts. Chaque sommet est caractérisé par un spin prenant q valeurs possibles, et les communautés correspondent aux classes de sommets ayant des valeurs de spin égales. Une énergie du système est définie et doit être minimisée par recuit simulé. Cette minimisation favorise les paires de sommets liés par une arête et possédant le même état de spin tout en pénalisant les trop grandes classes de spins. Le nombre q de spins possibles correspond au nombre maximal de communautés que l'on peut trouver et doit être choisi de manière à ce qu'il soit supérieur au nombre effectif de communautés.

Et de nombreuses autres approches... Il existe de nombreuses autres méthodes reliées à la problématique de détection de communautés dans les grands graphes de terrain. Sans en établir une liste exhaustive, citons [47, 85, 18, 7, 15].

La détection de communautés dans les grands graphes de terrain s'approche du problème classique de partitionnement de graphe et du problème général de clustering de données, dont la méthode de clustering hiérarchique a été adaptée avec succès par plusieurs approches récentes (dont la nôtre).
Cette problématique récente a fait l'objet de très nombreuses études au cours des dernières années. Parmi les nombreuses approches récentes, nous retrouvons principalement des approches séparatives qui divisent successivement le graphe et des approches agglomératives qui fusionnent successivement des communautés. Les marches aléatoires, que nous allons utiliser dans notre approche, ont aussi été utilisées dans d'autres méthodes de détection de communautés et problèmes de clustering. Finalement un dernier type d'approche est basé sur l'optimisation directe d'une fonction de qualité.

1.4 Travaux effectués et organisation de la thèse

L'apport principal de cette thèse consiste en deux algorithmes complémentaires.

Tout d'abord, nous proposons un nouvel algorithme de détection de communautés basé sur les marches aléatoires. Nous introduirons pour cela une distance entre sommets basée sur les marches aléatoires dans les graphes. Cette distance sera reliée aux approches spectrales existantes et permet de tirer profit de leurs avantages en s'affranchissant des calculs pénalisants de valeurs et vecteurs propres. Nous adapterons alors une méthode de clustering hiérarchique basée sur cette distance pour construire une structure hiérarchique de communautés appelée dendrogramme. L'algorithme proposé a été comparé expérimentalement aux principaux algorithmes existants : il se situe parmi les meilleurs algorithmes existants dans la recherche d'un compromis ente la qualité des résultats et le coût de calcul.

En se basant sur le dendrogramme trouvé, nous proposerons ensuite une nouvelle méthode d'optimisation de fonctions de qualité qui permet de trouver de meilleures partitions en communautés et de détecter des structures de communautés à différentes échelles. Cette seconde méthode permet d'améliorer significativement la sortie de tout algorithme produisant une structure hiérarchique de communautés.

Le Chapitre 2 définira formellement les notions de théorie des graphes utilisées dans cette thèse et effectuera une étude des processus de marche aléatoire dans les graphes. Ce chapitre est indépendant de la problématique de détection de communautés.

Le Chapitre 3 présentera notre algorithme de détection de communautés qui utilise une distance entre sommets, basée sur les marches aléatoires. Nous établirons un lien fort entre cette distance et les approches spectrales existantes. Finalement nous évaluerons la complexité globale de l'approche, qui se classe parmi les méthodes les plus efficaces existantes.

Le Chapitre 4 présentera une méthode d'optimisation de fonctions de qualité sur les partitions que l'on peut construire à partir d'un dendrogramme. Plusieurs fonctions de qualité seront envisagées et nous définirons une classe générale de fonctions de qualité additives et multi-échelles qui permet de détecter des structures de communautés à différentes échelles. Cette méthode peut s'appliquer en sortie de notre méthode de détection de communautés

ou peut être utilisée pour améliorer la sortie de toute méthode qui produit une structure hiérarchique de communautés.

Le Chapitre 5 présentera de nombreuses études expérimentales. Nous y effectuerons principalement une comparaison directe des performances (en terme de temps de calcul et de qualité des résultats) entre notre approche de détection de communautés et les principales autres approches existantes. Nous évaluerons aussi les bénéfices apportés par la méthode d'optimisation de fonction de qualité proposée au Chapitre 4.

Chapitre 2

Graphes et marches aléatoires

Nous allons dans ce chapitre introduire et étudier les concepts fondamentaux des graphes et des marches aléatoires dans les graphes. Nous établirons les principales propriétés que nous allons utiliser par la suite pour la conception d'un algorithme de détection de communautés. Le lecteur familier avec ces concepts peut se contenter de lire la dernière section récapitulative du chapitre et passer au chapitre suivant.

2.1 Définitions

Nous introduisons dans un premier temps les notations, hypothèses et concepts relatifs aux graphes, et définissons dans un second temps un processus de marches aléatoires.

2.1.1 Graphes

Un graphe non-orienté $G = (V, E)$ est composé d'un ensemble V de sommets (ou noeuds) et d'un ensemble E de paires (non ordonnées) de sommets nommées arêtes (ou liens). Nous noterons n le nombre de sommets ($n = |V|$) et m le nombre d'arêtes ($m = |E|$). Les arêtes du graphe peuvent être pondérées grâce à une fonction de poids $w : E \to \mathbb{R}^+$ permettant de modéliser plus finement les interactions entre sommets, nous obtenons ainsi un graphe pondéré $G = (V, E, w)$. Le poids d'une arête $\{i, j\}$ entre deux sommets i et j sera noté $w_{ij} > 0$. Par convention, un poids nul est attribué dans le cas où l'arête n'existe pas ($w_{ij} = 0$ si $\{i, j\} \notin E$). Dans le cas d'un graphe non pondéré les poids des arêtes de E sont fixés à 1, ainsi dans ce cas particulier $\forall i, j \in V, w_{ij} \in \{0, 1\}$.

Dans l'optique d'une problématique de détection de communautés, nous pouvons sans perte de généralité considérer que les graphes manipulés sont connexes (c'est à dire qu'il existe un chemin entre toute paire de sommets, un chemin étant une suite de sommets reliés par des arêtes). En effet il est facile de calculer les composantes connexes (sous-ensembles de sommets connexes maximaux) d'un graphe (en temps linéaire). Chaque composante étant totalement déconnectée des autres, nous ne pourrons pas faire de rapprochement entre leurs sommets de manière non arbitraire. Le problème de détection de communautés peut alors être réduit à détecter séparément des communautés dans chaque composante connexe. Pour cela, nous supposons qu'un pré-calcul de composantes connexes est effectué, et nous appliquerons ensuite l'algorithme de détection de communautés séparément sur les sous graphes connexes. Pour éviter des cas particuliers nous rejetons aussi les sommets de degré nul qui ne présentent

dans notre contexte qu'un intérêt limité.

Des graphes orientés peuvent aussi être définis, leur ensemble E est alors composé de couples (ordonnés) de sommets nommés arcs. Un graphe non-orienté peut être vu comme un graphe orienté dans lequel tous les liens sont réciproques. Les propriétés des marches aléatoires diffèrent grandement entre les graphes orientés et non-orientés, comme nous le verrons dans la Section 2.6. Dans la suite de cette thèse, nous ne considérerons pour ces raisons que des graphes non-orientés.

Le degré $d(v)$ d'un sommet $v \in V$ est le nombre d'arêtes incidentes au sommet v ; il s'agit du nombre de sommets voisins de v. Nous définissons aussi le poids $w(i)$ d'un sommet i comme la somme des poids de ses arêtes incidentes :

$$w(i) = \sum_{j \in V} w_{ij}$$

Notons que le poids d'un sommet coïncide avec la définition du degré d'un sommet dans le cas des graphes non-pondérés.

La matrice d'adjacence A représentant les arêtes d'un graphe pondéré non-orienté est définie par :

$$A_{ij} = A_{ji} = \begin{cases} 0 & \text{si } \{i,j\} \notin E \\ w_{ij} & \text{si } \{i,j\} \in E \end{cases}$$

Dans le cas d'un graphe non-orienté, la matrice d'adjacence A est symétrique ($A^T = A$). Par ailleurs, dans le cas d'un graphe non pondéré $A_{ij} \in \{0,1\}$ (car nous fixons un poids w_{ij} égal à 1 pour toutes les arêtes de E).

Dans de nombreux cas, le poids d'un sommet d'un graphe pondéré est une bonne généralisation du degré d'un sommet d'un graphe non-pondéré. Cependant nous pouvons toujours définir la notion de degré d'un sommet dans un graphe pondéré comme le nombre de ses voisins, cette notion sera alors différente de la notion de poids. Nous définissons aussi la matrice diagonale D des poids des sommets de la façon suivante :

$$D_{ij} = \begin{cases} w(i) & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

Cette matrice D peut aussi être vue, dans le cas particulier des graphes non-pondérés, comme la matrice des degrés.

Nous allons de plus considérer que le graphe G est apériodique pour éviter des comportements particuliers dans l'analyse des marches aléatoires. Un graphe est apériodique lorsque le pgcd des longueurs de tous ses cycles est égal à 1. Il est facile de garantir cette condition d'apériodicité en ajoutant par exemple des boucles sur les sommets comme nous allons le faire dans le chapitre suivant. De plus les graphes rencontrés en pratique n'ont généralement aucune raison de présenter une période. Cette contrainte n'est donc pas une limite en pratique, mais nous pouvons cependant citer le cas des graphes bipartis (dont la période est paire) qui sont des graphes périodiques. Nous discuterons en Section 2.4 du cas particulier des graphes bipartis qui peuvent modéliser des phénomènes d'affiliation.

Nous considérerons dans cette thèse un graphe connexe, non-orienté, apériodique et pondéré $G = (V, E, w)$ composé de $n = |V|$ sommets et $m = |E|$ arêtes et de poids w_{ij} sur les arêtes $\{i,j\} \in E$.

2.1.2 Marches aléatoires

Nous définissons dans ce paragraphe un processus de marche aléatoire à temps discret dans G. Le temps est discrétisé ($t = 0, 1, 2, \ldots$) et un marcheur est localisé à chaque instant t sur un sommet du graphe G. Le marcheur se déplace à chaque instant aléatoirement et uniformément vers l'un de ses sommets voisins. La suite des sommets visités constitue une marche aléatoire.

Lorsque le marcheur se trouve à un instant donné sur le sommet i, nous notons P_{ij} la probabilité d'aller du sommet i au sommet j à l'instant suivant. Dans le cas d'un graphe non pondéré, le marcheur a une probabilité de $\frac{1}{d(i)}$ d'aller vers l'un des $d(i)$ voisins du sommet i. Lorsque nous considérons un graphe pondéré, les différentes arêtes empruntées pour relier les différents voisins n'ont pas le même poids et donc n'ont pas la même importance. Nous considérons alors que la probabilité d'emprunter chaque arête est proportionnelle au poids de l'arête. La probabilité de transition P_{ij} d'un sommet i à un sommet j est donc donnée par l'expression suivante :

$$P_{ij} = \frac{A_{ij}}{w(i)} = \frac{A_{ij}}{\sum_{k=1}^{n} A_{ik}} = \frac{w_{ij}}{\sum_{k=1}^{n} w_{ik}}$$

Ces probabilités définissent la matrice de transition P de la chaîne de Markov associée au processus de marche aléatoire. Notons qu'il n'y a pas de problème de définition car les contraintes que nous avons fixées sur le graphe imposent $w(i) > 0$ pour tous les sommets i. Nous pouvons définir cette matrice de transition à partir de la matrice d'adjacence A du graphe et de la matrice diagonale D des poids des sommets selon la formule suivante :

$$P = D^{-1}A \tag{2.1}$$

Étant donnée une position initiale du marcheur, les probabilités de position du marcheur après t étapes vont jouer un rôle essentiel dans la suite. Un exemple est proposé en Figure 2.1.

Nous définissons le vecteur de distribution de probabilité de position $\rho(t)$ du marcheur après t étapes (il satisfait $\forall i \in V, \rho_i(t) \geq 0$ et $\sum_{i \in V} \rho_i(t) = 1$). La distribution $\rho(0)$ donne la position initiale du marcheur, par exemple si la marcheur part d'un unique sommet i, $\rho_i(0) = 1$ et $\forall j \neq i, \rho_j(0) = 0$. Le marcheur peut aussi partir de plusieurs sommets à la fois avec différentes probabilités pour chaque sommet.

Pour calculer la probabilité de position $\rho(t+1)$ nous devons connaître les probabilités de position $\rho(t)$ à l'étape précédente. Ainsi la probabilité d'atteindre un sommet j à l'instant $t+1$ est directement lié aux probabilités de position à l'instant t et aux probabilités de transitions P_{ij} :

$$\forall j, \rho_j(t+1) = \sum_{i \in V} \rho_i(t) P_{ij}$$

En notant P^T la matrice transposée de la matrice P, nous obtenons la relation matricielle suivante :

$$\rho(t+1) = P^T \rho(t)$$

qui conduit à l'expression suivante pour $\rho(t)$:

$$\rho(t) = (P^T)^t \rho(0) \text{ soit } \forall j, \rho_j(t) = \sum_{i \in V} (P^t)_{ij} \rho_i(0)$$

Nous utiliserons par la suite la notation P^t_{ij} pour $(P^t)_{ij}$, et nous remarquons que la formule précédente permet d'interpréter la valeur P^t_{ij} comme étant la probabilité d'aller du sommet i au sommet j en t étapes.

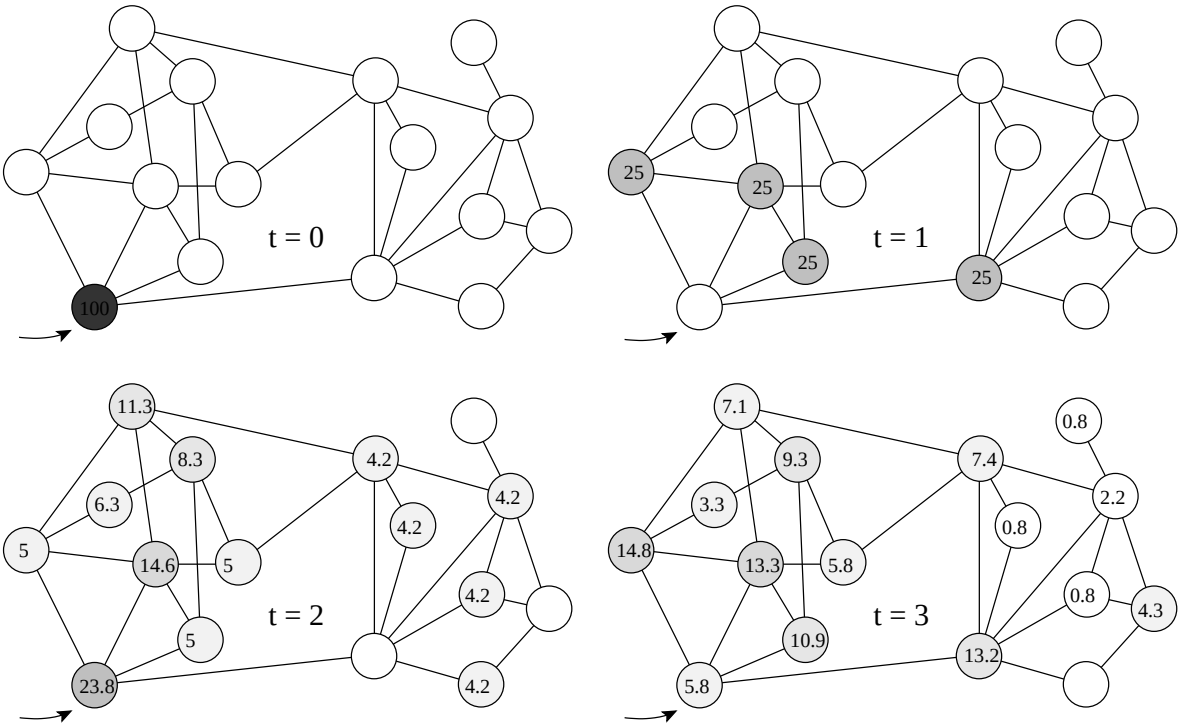

FIG. 2.1 – Exemple de dynamique d'une marche aléatoire sur un graphe non-pondéré. Le marcheur part du sommet indiqué par la flèche, les probabilités de position aux temps $t = 0, 1, 2$ et 3 sont indiquées en pourcentage sur chaque sommet.

Une marche aléatoire dans un graphe $G = (V, E, w)$ est un processus en temps discret sur l'ensemble des sommets V. Sa matrice de transition P est donnée par

$$P_{ij} = \frac{A_{ij}}{w(i)} = \frac{w_{ij}}{\sum_{j=1}^{n} w_{ij}}$$

Nous noterons $P_{ij}^t = (P^t)_{ij}$ la probabilité d'aller d'un sommet i à un sommet j par une marche aléatoire de longueur t.

2.2 Propriétés

Nous allons maintenant nous concentrer sur certaines propriétés des marches aléatoires dans un graphe. Cette section n'est pas une étude complète mais plutôt une présentation des principales propriétés qui vont être utilisées par la suite. Nous encourageons le lecteur voulant approfondir cette étude à se référer à l'ouvrage d'Aldous et Fill en cours de rédaction [5] ainsi qu'à la synthèse de Lovász [53], tous deux consacrés aux marches aléatoires dans les graphes.

Propriété 1 (Réversibilité des marches aléatoires) *Lors d'une marche aléatoire de longueur fixée t dans un graphe non-orienté, le rapport entre la probabilité P_{ij}^t d'aller du sommet*

i au sommet j et la probabilité P^t_{ji} d'aller du sommet j au sommet i ne dépend que des poids des sommets :

$$\forall i,j \in V, \frac{P^t_{ij}}{w(j)} = \frac{P^t_{ji}}{w(i)}$$

Cette propriété correspond à la propriété de réversibilité de la chaîne de Markov sous-jacente. *Preuve :* L'égalité à montrer peut s'écrire matriciellement $DP^tD^{-1} = (P^t)^T$. En utilisant la définition de la matrice de transition $P = D^{-1}A$ donnée par l'Équation 2.1, nous avons : $DP^tD^{-1} = D(D^{-1}A)^tD^{-1} = (AD^{-1})^t$. La matrice d'adjacence A est symétrique car le graphe est non-orienté ($\forall i,j \in V, A_{ij} = A_{ji}$ donc $A = A^T$). La matrice D est elle aussi symétrique car diagonale, donc $D = D^T$. Nous avons donc $(AD^{-1})^t = (A^T(D^T)^{-1})^t = (A^T(D^{-1})^T)^t = ((D^{-1}A)^T)^t = (P^T)^t = (P^t)^T$. Ce qui démontre la relation souhaitée $DP^tD^{-1} = (P^t)^T$. □

Propriété 2 (Distribution limite) *Dans un graphe non-orienté, apériodique et connexe, la distribution de probabilité de position d'une marche aléatoire converge lorsque t tend vers l'infini vers un distribution limite stationnaire π vérifiant $P^T\pi = \pi$. La probabilité d'atteindre un sommet donné est alors proportionnelle à son poids :*

$$\forall i \in V, \lim_{t\to+\infty} P^t_{ij} = \frac{w(j)}{\sum_{k\in V} w(k)} = \pi_j \text{ et donc } \forall \rho(0), \lim_{t\to+\infty} \rho(t) = \pi$$

Cette propriété traduit la propriété d'ergodicité de la chaîne de Markov sous-jacente sous les conditions de connexité et d'apériodicité du graphe G. Nous allons proposer une preuve de cette propriété, en nous basant sur le lemme suivant :

Lemme 1 *Les valeurs propres de la matrice de transition P sont réelles et satisfont :*

$$1 = \lambda_1 > \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_n > -1$$

Il existe de plus une famille de vecteurs orthonormés $(s_\alpha)_{1\leq\alpha\leq n}$ telle que les vecteurs $v_\alpha = D^{-\frac{1}{2}}s_\alpha$ et $u_\alpha = D^{\frac{1}{2}}s_\alpha$ sont respectivement des vecteurs propres à droite et à gauche des valeurs propres λ_α :

$$\forall \alpha, Pv_\alpha = \lambda_\alpha v_\alpha \text{ et } P^Tu_\alpha = \lambda_\alpha u_\alpha$$

ces vecteurs vérifient alors :

$$\forall \alpha,\beta, v_\alpha^T u_\beta = \delta_{\alpha\beta}$$

Preuve :

La matrice de transition P possède les mêmes valeurs propres que sa matrice similaire $S = D^{\frac{1}{2}}PD^{-\frac{1}{2}} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$.

La matrice S est symétrique car les matrices A et D sont elles aussi symétriques : $S^T = (D^{-\frac{1}{2}}AD^{-\frac{1}{2}})^T = (D^{-\frac{1}{2}})^TA^T(D^{-\frac{1}{2}})^T = D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = S$. La matrice S est une matrice réelle et symétrique, ses valeurs propres (qui sont aussi celles de P) sont donc réelles.

La matrice P est une matrice stochastique ($\forall i, \sum_{j=1}^n P_{ij} = 1$) donc sa valeur propre de module maximal est $\lambda_1 = 1$. Le graphe G étant connexe et apériodique, nous pouvons appliquer le théorème de Perron-Frobenius impliquant que la matrice P possède une unique valeur propre dominante. Nous avons donc $|\lambda_\alpha| < 1$ pour $2 \leq \alpha \leq n$.

La symétrie de la matrice S implique l'existence d'une famille de vecteurs propres orthonormés satisfaisant $\forall \alpha, \beta, s_\alpha^T s_\beta = \delta_{\alpha\beta}$ où $\delta_{\alpha\beta}$ est le symbole de Kronecker ($\delta_{\alpha\beta} = 0$ si $\alpha \neq \beta$ et $\delta_{\alpha\beta} = 1$ si $\alpha = \beta$). Nous obtenons, en utilisant la relation $S = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, que les vecteurs $v_\alpha = D^{-\frac{1}{2}} s_\alpha$ et $u_\alpha = D^{\frac{1}{2}} s_\alpha$ sont des vecteurs propres de P satisfaisant $\forall \alpha, \beta, v_\alpha^T u_\alpha = \delta_{\alpha\beta}$. □

Nous pouvons grâce à ce Lemme donner une preuve de la Propriété 2 :
Preuve : Le Lemme 1 permet de réaliser une décomposition spectrale de la matrice P :

$$P = \sum_{\alpha=1}^{n} \lambda_\alpha v_\alpha u_\alpha^T$$

Ceci permet en utilisant l'orthonormalité des vecteurs v et u d'obtenir l'expression suivante pour les puissances successives de la matrice P :

$$P^t = \sum_{\alpha=1}^{n} \lambda_\alpha^t v_\alpha u_\alpha^T$$

Nous obtenons ainsi l'expression simple pour les différentes probabilités de transition P_{ij}^t :

$$P_{ij}^t = \sum_{\alpha=1}^{n} \lambda_\alpha^t v_\alpha(i) u_\alpha(j)$$

Lorsque t tend vers l'infini, tous les termes λ_α^t pour $\alpha \geq 2$ tendent vers 0 et disparaissent car $|\lambda_\alpha| < 1$ pour $2 \leq \alpha \leq n$. De plus $\lambda_1 = 1$ donc :

$$\forall i, j, \lim_{t \to +\infty} P_{ij}^t = v_1(i) u_1(j)$$

Il est facile de vérifier que le vecteur propre à droite v_1 associé à la valeur propre principale $\lambda_1 = 1$ est un vecteur constant. En normalisant le vecteur $s_\alpha = D^{\frac{1}{2}} v_\alpha$ associé, nous obtenons $\forall i, v_1(i) = \frac{1}{\sqrt{\sum_{k \in V} w(k)}}$. Nous obtenons alors les composantes du vecteur u_1 en utilisant la relation $u_1 = D v_1$ soit $\forall i, u_1(j) = \frac{w(j)}{\sqrt{\sum_{k \in V} w(k)}}$. Les probabilités de transition P_{ij}^t convergent donc vers la valeur limite π_j ne dépendant que du sommet d'arrivée :

$$\lim_{t \to +\infty} P_{ij}^t = v_1(i) u_1(j) = \frac{w(j)}{\sum_{k \in V} w(k)} = \pi_j$$

Il est facile de vérifier que le vecteur de probabilité de position π est un vecteur stationnaire pour le processus de marche aléatoire : il vérifie $P^T \pi = \pi$. Quelle que soit la distribution initiale de position $\rho(0)$, les distributions de position $\rho(t)$ convergent vers la distribution stationnaire limite π car $\rho_j(t) = \sum_{i \in V} P_{ij}^t \rho_i(0)$ et ainsi

$$\lim_{t \to +\infty} \rho_j(t) = \sum_{i \in V} \lim_{t \to +\infty} P_{ij}^t \rho_i(0) = \sum_{i \in V} \pi_j \rho_i(0) = \pi_j$$

□

Dans un graphe connexe et apériodique, lorsque la longueur t d'une marche aléatoire tend vers l'infini, la probabilité limite π_j de se trouver sur un sommet j donné est proportionnelle à son poids $w(j)$:

$$\forall i \in V, \lim_{t \to +\infty} P_{ij}^t = \frac{w(j)}{\sum_{k \in V} w(k)} = \pi_j$$

Pour des marches de même longueur t, les probabilités de transition aller (P_{ij}^t) et retour (P_{ji}^t) entre deux sommets i et j sont liées par la relation suivante :

$$\frac{P_{ij}^t}{w(j)} = \frac{P_{ji}^t}{w(i)} \text{ soit encore : } \frac{P_{ij}^t}{\pi_j} = \frac{P_{ji}^t}{\pi_i}$$

2.3 Calcul des marches aléatoires

Nous allons exposer dans cette partie les méthodes de calcul des marches aléatoires. En effet l'algorithme que nous proposerons dans le chapitre suivant sera basé sur des calculs de probabilités de transition P_{ij}^t de marches aléatoires courtes. Nous devons garder à l'esprit que dans le contexte dans lequel nous nous plaçons, les graphes considérés sont des graphes peu denses ayant généralement un degré moyen borné (et donc un nombre d'arêtes linéaire en fonction du nombre de sommets $m = \mathcal{O}(n)$). Nous verrons de plus que nous n'utiliserons que des marches aléatoires courtes pour éviter d'atteindre le régime stationnaire limite : en pratique les longueurs de marche t utilisées sont inférieures à dix pas.

Le problème que nous considérons ici est le suivant : étant donné un vecteur de probabilité de position initiale $\rho(0)$ ($\rho_i(0)$ représente la probabilité de commencer la marche aléatoire du sommet i) nous voulons connaître le vecteur de probabilité de position $\rho(t)$ après t étapes ($\rho(t) = (P^T)^t \rho(0)$). Cette problématique est équivalente, à peu de choses près, au calcul des probabilités de transition P_{ij}^t que nous utiliserons dans notre algorithme.

2.3.1 Calcul exact pour des graphes denses

Lorsque nous n'avons aucune hypothèse sur la densité du graphe, la méthode la plus efficace est la méthode naïve de pré-calcul de la matrice P^t. Nous pouvons pour cela effectuer une exponentiation rapide qui nécessitera $\mathcal{O}(\log(t))$ multiplications de matrices de taille n. La méthode basique donnera alors une complexité en temps $\mathcal{O}(n^3 \log(t))$ qui peut être ramenée à $\mathcal{O}(n^\omega \log(t))$ en utilisant la meilleure technique de multiplication rapide de matrices actuellement connue [17] donnant $\omega = 2,376$. La complexité en espace est de $\mathcal{O}(n^2)$ pour le stockage d'une matrice de taille n.

Une fois ce pré-calcul effectué, nous pouvons obtenir le vecteur de probabilité de position $\rho(t)$ à partir du vecteur initial $\rho(0)$ par une multiplication du vecteur $\rho(0)$ par la matrice $(P^t)^T$. Chacun de ces calculs représente alors une complexité en temps $\mathcal{O}(n^2)$.

Cette approche possède le gros désavantage de nécessiter un espace mémoire important pour stocker un matrice entière. Elle demeure tout de même la meilleure approche lorsque nous sommes confrontés à un graphe dense pour lequel il faudra de toute façon une espace mémoire en $\mathcal{O}(n^2)$ pour stocker le graphe.

2.3.2 Calcul exact pour des graphes peu denses

Nous supposons ici que le graphe G est peu dense ; en pratique il est souvent fait l'hypothèse d'un degré moyen borné qui implique $m = \mathcal{O}(n)$. Nous donnerons donc les complexités en fonction du nombre d'arêtes m du graphe. Nous supposons aussi que la longueur t des marches à calculer est courte, typiquement dans les cas pratiques nous utiliserons des marches d'une longueur de l'ordre de 3 à 8. Une étude sur les raisons du choix de la longueur t est proposée en 3.1.4.

La méthode consiste simplement ici à calculer $\rho(t)$ en multipliant successivement t fois le vecteur initial $\rho(0)$ par la matrice creuse P^T. Nous pouvons stocker la matrice P^T en utilisant un espace $\mathcal{O}(m)$ grâce à des listes d'adjacence. Chaque multiplication de vecteur demande alors $\mathcal{O}(m)$ opérations. Le temps nécessaire pour calculer un vecteur de probabilité de position $\rho(t)$ est donc en $\mathcal{O}(tm)$. L'espace mémoire utilisé est limité à la taille du vecteur $\mathcal{O}(n)$ plus la taille de stockage de la matrice creuse $\mathcal{O}(m)$.

Cette méthode est particulièrement adaptée aux cas qui nous concernent, à savoir des graphes peu denses et des marches aléatoires de faible longueur t. Nous avons utilisé cette méthode dans l'implémentation de l'algorithme de détection de communautés présenté au chapitre suivant.

2.3.3 Calcul approché par simulation

Une autre méthode est de simuler des marches aléatoires pour calculer de manière approchée les probabilités de position souhaitées. Il est facile de simuler une marche aléatoire de longueur t qui part d'un sommet donné en un temps linéaire $\mathcal{O}(t)$ du nombre de pas à simuler. Cette complexité est possible lorsqu'on utilise une structure de stockage du graphe par tableau d'adjacence qui permet de tirer aléatoirement un voisin d'un sommet en temps constant $\mathcal{O}(1)$.

Il suffit alors de simuler un nombre K de marches aléatoires de longueur t et de comptabiliser le nombre N_i de marches qui finissent sur chaque sommet $i \in V$. Les sommets de départ sont pris selon les proportions données par le vecteur initial $\rho(0)$ (le cas fréquent d'une marche partant d'un seul sommet est plus facile à traiter). Nous estimons alors la probabilité de position finale par la valeur $\tilde{\rho}_i(t) = \frac{N_i}{K}$.

Le théorème central limite indique alors que les probabilités estimées $\tilde{\rho}_i(t)$ convergent vers les probabilités exactes $\rho_i(t)$ avec un erreur relative qui décroît à une vitesse en $\mathcal{O}(\frac{1}{\sqrt{K}})$ lorsque K tend vers l'infini. Le choix de K dépend alors de la précision souhaitée et du temps de calcul disponible.

Le calcul d'un vecteur $\tilde{\rho}(t)$ nécessite donc un temps de calcul $\mathcal{O}(tK)$ et un espace mémoire $\mathcal{O}(n)$ pour stocker les compteurs N_i comptabilisant le nombre de marches arrivant sur chaque sommet i. Cette méthode ne possède donc un avantage que dans les cas de très grands graphes et lorsque nous prenons $K < n$. Nous n'avons en pratique jamais rencontré de cas pour lesquels cette approche se révèle particulièrement attractive.

> Il est possible de calculer un vecteur de distribution de probabilité de position $\rho(t)$ d'une marche aléatoire en temps $\mathcal{O}(tm)$ et en espace $\mathcal{O}(n)$. Cette méthode est particulièrement adaptée aux calculs de marches de faible longueur t et aux graphes peu denses qui sont stockés sous forme de tableaux d'adjacences dans un espace mémoire $\mathcal{O}(m)$.

2.4 Graphes bipartis et marches aléatoires

Nous allons consacrer cette partie au cas particulier des marches aléatoires dans les graphes bipartis. Après avoir présenté le contexte d'utilisation des graphes bipartis, nous allons analyser les méthodes de projection qui permettent d'obtenir des graphes entre sommets de même classe. Nous étudierons ensuite les marches aléatoires dans les graphes bipartis et leurs projections.

2.4.1 Définition et contexte des graphes bipartis

Un graphe biparti $G = (V, E)$ est un graphe possédant deux classes de sommets disjointes ($V = V^1 \cup V^2$ et $V^1 \cap V^2 = \emptyset$) telles qu'il n'existe aucune arête reliant deux sommets de la même classe ($E \cap (V^1 \times V^1) = \emptyset$ et $E \cap (V^2 \times V^2) = \emptyset$). Pour illustrer ceci, nous pouvons voir un graphe biparti comme un graphe où il est possible de colorier les sommets avec deux couleurs différentes sans jamais colorier deux sommets voisins avec la même couleur. Les deux classes de sommets correspondent alors aux deux couleurs.

Les graphes bipartis peuvent modéliser des phénomènes d'affiliation où des objets d'un certain type peuvent être reliés à d'autres objets d'un type différent. Par exemple nous avons eu l'occasion de manipuler des réseaux d'acteurs reliés aux films dans lesquels ils ont joué, des réseaux d'auteurs reliés aux publications qu'ils ont signées, des réseaux de signataires de pétitions, des réseaux de mots clés reliés à des pages Web ou photos, etc. De nombreux phénomènes, dès lors qu'ils font intervenir des objets de nature différente (acteurs/films, auteurs/publications, signataires/pétitions, etc.), peuvent être modélisés par des graphes bipartis.

2.4.2 Projection d'un graphe biparti

Les graphes bipartis sont souvent utilisés comme une étape de modélisation qui permet par la suite de considérer d'autres relations. En effet il est souvent plus significatif de considérer les relations entre sommets d'une même classe qui sont alors reliés par l'intermédiaire de leurs voisins de l'autre classe. Dans les exemples cités ci-dessus on s'intéresse typiquement aux réseaux de co-apparition d'acteurs dans les films, de collaborations d'auteurs, de co-signataires de pétitions ou de de co-occurrences de mots clés.

Pour prendre en compte ces liens entre sommets d'une même classe il suffit de réaliser une projection du graphe biparti sur la classe de sommets que l'on désire étudier (nous prendrons ici l'exemple d'une projection sur V^1). Nous créons ainsi un nouveau graphe $G^1 = (V^1, E^1)$ sur l'ensemble des sommets V^1. Il existe une arête entre deux sommets i et j si ces deux sommets sont liés à au moins un voisin commun de V^2 : $\{i, j\} \in E^1 \Leftrightarrow \exists u \in V^2 / \{i, u\} \in E$ et $\{j, u\} \in E$. Ainsi dans les exemples cités, deux acteurs sont liés s'ils ont joué ensemble dans au moins un film et deux auteurs sont liés s'ils ont au moins signé une publication en commun.

Cette méthode de projection est la projection la plus simple produisant un graphe non-pondéré. Il n'est fait aucune différence entre deux acteurs qui ont joué un seule fois dans un même film et deux acteurs qui ont joué plusieurs fois ensemble. Pour pallier ce manque nous pouvons simplement pondérer un lien $\{i, j\}$ du nouveau graphe G^1 par le nombre de sommets communs de V^2 auxquels sont liés les sommets i et j. Cette seconde approche prend en compte les liens multiples créés par la projection, cependant elle peut donner un poids trop important aux sommets de plus fort degré de V^2.

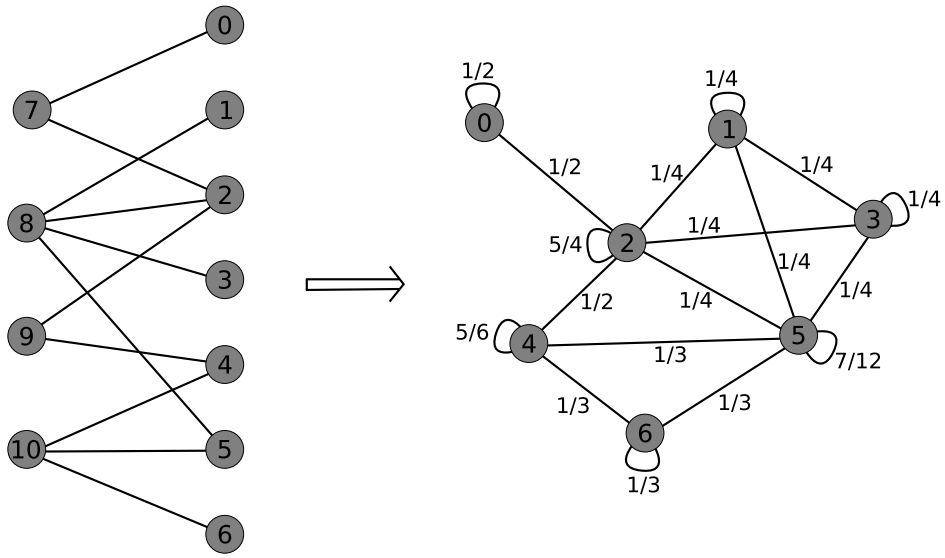

FIG. 2.2 – Exemple de projection d'un graphe biparti. Les poids des arêtes du graphe biparti de départ valent 1.

En effet nous pouvons voir cette transformation comme l'addition pour chaque sommet $v \in V^2$ d'une clique (sous-ensemble de sommets tous reliés entre eux) regroupant ses sommets voisins de V^1. Ainsi un sommet $v \in V^2$ de degré $d(v)$ correspond à l'ajout de $\frac{1}{2}(d(v)+1)d(v)$ arêtes (en comptant les boucles) entre les voisins de v dans le graphe projeté et engendre donc un poids total de $\frac{1}{2}(d(v)+1)d(v)$. Les degrés des graphes rencontrés en pratique étant très hétérogènes, les arêtes du graphe projeté dépendront majoritairement des quelques sommets de plus fort degré. Pour limiter ce phénomène, nous pouvons pondérer les liens créés par la taille des cliques générées : un sommet de degré $d(v)$ générera une clique dont toutes les arêtes auront un poids $\frac{1}{d(v)}$. Le poids total de la clique sera alors égal au poids du sommet initial, c'est à dire $d(v)$.

Nous allons généraliser cette approche de projection pondérée en considérant le cas d'un graphe biparti pondéré en entrée. La définition suivante formalise l'analyse que nous venons de faire :

Définition 1 *Soit $G = (V, E, w)$ un graphe biparti pondéré ($V = V^1 \cup V^2$). La projection pondérée de G sur V^1 produit un graphe $G^1 = (V^1, E^1, w^1)$ dont les poids sont définis par :*

$$\forall i, j \in V^1, w^1_{ij} = \sum_{v \in V^2} \frac{w_{iv} w_{vj}}{w(v)}$$

L'ensemble des arêtes E^1 est défini par la convention $\{i, j\} \in E^1 \Leftrightarrow w^1_{ij} > 0$.

Notons que cette transformation (illustrée à la Figure 2.2)crée des boucles sur les sommets du graphe projeté.

2.4.3 Marches aléatoires et projections de graphes bipartis

Nous allons relier dans cette partie les marches aléatoires dans les graphes bipartis aux marches aléatoires dans leurs projections.

Un graphe biparti est un graphe périodique : tous ces cycles sont de longueur paire. Une marche aléatoire dans un graphe biparti possède un comportement particulier : en partant d'un sommet donné, le marcheur se retrouvera en fonction de la parité de la longueur de la marche soit dans la classe du sommet initial soit dans l'autre classe de sommet. Ainsi les probabilités de position ne peuvent plus converger vers une distribution limite car elle vont osciller entre une classe de sommets et l'autre classe de sommets à chaque pas. Ces comportements peuvent être mis à profit pour calculer de manière efficace des marches aléatoires dans les projections des graphes bipartis.

Nous avons vu que les projections des graphes bipartis représentent souvent des objets d'étude plus pertinents que le graphe biparti lui même. Nous pouvons donc être amenés à rechercher des structures de communautés dans ces graphes. La projection pose cependant un problème de taille des données. En effet lorsque nous projetons un sommet de degré $d(v)$ en une clique nous transformons $d(v)$ arêtes du graphe biparti initial en $d(v)^2$ arêtes. Les graphes rencontrés en pratique ayant souvent une distribution hétérogène des degrés, il est fréquent de rencontrer des sommets de très fort degré qui produirons des cliques de taille trop importante dans le graphe projeté. En pratique il n'est pas toujours possible de construire le graphe projeté qui possède trop d'arêtes.

Nous allons cependant montrer qu'il est possible de calculer des marches aléatoires (et donc d'utiliser notre approche de détection de communautés) dans le graphe projeté sans le construire (nous utiliserons uniquement le graphe biparti). Le théorème suivant indique qu'il est possible de simuler une marche équivalente de longueur t dans le graphe projeté par une marche de longueur $2t$ dans le graphe biparti original.

Théorème 1 *Soit un graphe biparti G et son graphe projeté G^1 sur la classe des sommets V^1. Soit une marche aléatoire sur le graphe projeté G^1 de vecteur position initiale $\rho(0)$. Alors la marche aléatoire sur le graphe biparti G de même position initiale μ^0 ($\forall i \in V_1, \mu_i^0 = \rho_i(0)$ et $\forall i \in V_2, \mu_i^0 = 0$) est reliée à la première marche par :*

$$\forall t, \forall i \in V^1, \rho_i(t) = \mu_i^{2t}$$

Preuve : Pour montrer le théorème, il suffit de montrer que les probabilités de transition en deux pas dans le graphe biparti sont égales aux probabilités de transition en un pas dans le graphe projeté. Dans le graphe projeté, la probabilité de transition d'un sommet i à un sommet j est :

$$P_{ij} = \frac{w_{ij}^1}{w^1(i)} = \frac{\sum_{v \in V^2} \frac{w_{iv} w_{vj}}{w(v)}}{\sum_{k \in V^1} \sum_{v \in V^2} \frac{w_{iv} w_{kv}}{w(v)}} = \frac{\sum_{v \in V^2} \frac{w_{iv} w_{vj}}{w(v)}}{w_{iv} \sum_{v \in V^2} \frac{\sum_{k \in V^1} w_{kv}}{w(v)}} = \frac{\sum_{v \in V^2} \frac{w_{iv} w_{vj}}{w(v)}}{\sum_{v \in V^2} w_{iv}} = \sum_{v \in V^2} \frac{w_{iv}}{w(i)} \frac{w_{vj}}{w(v)}$$

Cette expression montre bien que la probabilité de transfert dans le graphe projeté correspond à la probabilité d'une marche de longueur deux dans le graphe biparti. Nous retrouvons en effet les probabilités $\frac{w_{iv}}{w(i)}$ d'aller du sommet i à un sommet $v \in V^2$ puis la probabilité $\frac{w_{vj}}{w(v)}$ de revenir du sommet v vers le sommet j. □

Il est souvent intéressant de considérer une projection d'un graphe biparti sur l'une de ses deux classes de sommets. Nous pouvons calculer efficacement des marches aléatoires dans ce graphe projeté sans avoir à construire explicitement le graphe projeté qui possède généralement un nombre très important d'arêtes.
Ceci permet d'utiliser efficacement notre approche de détection de communautés sur des projections de graphes bipartis.

2.5 Marches aléatoire en temps continu

Nous allons montrer ici comment passer des marches aléatoires en temps discret à des marches aléatoires en temps continu. Cette méthode permet d'envisager des marches de longueur intermédiaire entre les longueurs entières. Cette approche a principalement un intérêt théorique car nous n'avons en pratique pas rencontré de cas pour lesquels une différence suffisante de comportement entre deux longueurs t successives justifiait d'avoir recours à des longueurs intermédiaires.

2.5.1 Définition du processus

Considérons une distribution de probabilité de position en temps continu $(\bar{\rho}(t))_{t\in\mathbb{R}^+}$. Pendant une durée infinitésimale dt, le marcheur va changer de sommet selon la matrice de transition P avec une probabilité dt et va rester sur le même sommet avec une probabilité $1-dt$. La variation de probabilité de position sur chaque sommet j doit prendre en compte la probabilité que le marcheur arrive d'un sommet voisin et celle que le marcheur quitte le sommet durant dt. Ainsi :

$$\frac{d\bar{\rho}_j(t)}{dt} = \sum_{i=1}^{n} \bar{\rho}_i(t)P_{ij} - \bar{\rho}_j(t)$$

Ce qui donne sous forme matricielle :

$$\frac{d\bar{\rho}(t)}{dt} = P^T\bar{\rho}(t) - \bar{\rho}(t) = (P^T - Id)\bar{\rho}(t)$$

Cette équation différentielle linéaire du premier ordre se résout facilement et nous donne la chaîne de Markov en temps continu associée :

$$\bar{\rho}(t) = e^{t(P^T-Id)}\rho(0)$$

Cette définition utilise l'exponentielle d'une matrice définie par :

$$e^M = \sum_{k=0}^{\infty} \frac{M^k}{k!}$$

Ce processus a les mêmes propriétés que le processus discret initial car les vecteurs propres de la matrice $e^{t(P^T-Id)}$ sont les mêmes que ceux de la matrice P^T et les valeurs propres $\bar{\lambda}_\alpha^{(t)}$

sont déduites de celles de la matrice P^T par : $\bar{\lambda}_\alpha^{(t)} = e^{t(\lambda_\alpha - 1)}$. Nous pouvons donc écrire une décomposition spectrale de $e^{t(P^T - Id)}$:

$$e^{t(P^T - Id)} = \sum_{\alpha=1}^{n} e^{t(\lambda_\alpha - 1)} u_\alpha v_\alpha^T = D^{\frac{1}{2}} \sum_{\alpha=1}^{n} e^{t(\lambda_\alpha - 1)} s_\alpha v_\alpha^T$$

Ceci permettra d'obtenir des propriétés similaires au cas discret en nous basant sur les études de la Section 2.2 : seules les valeurs propres changent par rapport à la décomposition spectrale de la matrice $(P^T)^t$ donnée dans la preuve de la Propriété 2. La valeur propre principale est toujours $e^{t(\lambda_1 - 1)} = e^0 = 1$ comme précédemment, ce qui conduit à la même distribution stationnaire limite. Les quantités $e^{\lambda_\alpha - 1}$ conservent le même ordre que les valeurs propres λ_α mais sont toutes positives : les valeurs propres négatives λ_α correspondant à des phénomènes oscillatoires indésirables deviennent donc maintenant négligeables. De plus les valeurs propres $e^{t(\lambda_\alpha - 1)}$ conservent le même type de décroissance exponentielle que λ_α^t. Ces similitudes fortes justifient la possibilité d'utiliser un processus en temps continu à la place d'un processus en temps discret.

2.5.2 Calcul des marches aléatoires en temps continu

L'exponentielle de la matrice creuse $t(P^T - Id)$ provenant d'une chaîne de Markov peut être calculée de la manière suivante [75] :

$$e^{t(P^T - Id)} \simeq e^{-t} \sum_{k=0}^{r} \frac{t^k}{k!} (P^T)^k$$

Ceci donne pour un vecteur position approché $\bar{\rho}(t)$ suivant :

$$\bar{\rho}(t) \simeq e^{-t} \sum_{k=0}^{r} \frac{t^k}{k!} (P^T)^k \rho(0) = e^{-t} \sum_{k=0}^{r} \frac{t^k}{k!} \rho(k)$$

où $\rho(k)$ (à ne pas confondre avec $\bar{\rho}(t)$) est le vecteur position des marches aléatoires discrètes de longueur k précédemment défini. Il suffit donc de calculer ces vecteurs $\rho(k)$ successivement jusqu'à la longueur $k = r$ avec la méthode proposée en Section 2.3.2. Nous obtenons alors une complexité en temps de $\mathcal{O}(rm)$ pour chaque vecteur et une complexité en espace de $\mathcal{O}(n)$.

La série est tronquée à un rang r choisi pour obtenir la précision souhaitée : l'erreur commise sur chaque coordonnée est au plus $\varepsilon = e^{-t} \sum_{k=r+1}^{+\infty} \frac{t^k}{k!}$. Cette méthode simple est numériquement stable pour une matrice P stochastique et est efficace pour des graphes peu denses.

Il est possible de définir un processus de marches aléatoires en temps continu possédant la distribution de probabilité de position $(\bar{\rho}(t))_{t \in \mathbb{R}^+}$ suivante :

$$\bar{\rho}(t) = e^{t(P^T - Id)} \rho(0)$$

Ce processus conserve des propriétés similaires au processus discret et possède l'avantage (principalement théorique) de pouvoir considérer des longueurs de marche t non entières.

2.6 Cas des graphes orientés

Nous allons étudier ici les différences de comportement des marches aléatoires dans les graphes orientés et non-orientés.

Tout d'abord la propriété de réversibilité (Propriété 1) ne s'applique plus : la matrice d'adjacence A n'est plus symétrique. De plus dans le cas général où le graphe G n'est pas fortement connexe, certains sommets ne peuvent pas être atteints. Il est ainsi possible que les marches aléatoires se fassent piéger dans des composantes fortement connexes desquelles elles ne peuvent plus ressortir. La notion de distribution limite n'est dans le cas général plus définissable car les marches peuvent, pour une même position initiale et selon les premiers pas du processus, terminer piégées dans différentes des composantes fortement connexes disjointes.

Nous allons donc nous intéresser au cas d'un graphe G fortement connexe. Sous cette condition, et en ajoutant encore la condition d'apériodicité du graphe, les marches aléatoires convergent vers une distribution limite stationnaire π. Nous retrouvons ainsi une partie de la Propriété 2, cependant cette distribution limite ne dépend plus des degrés des sommets mais elle peut tout de même mesurer leur importance. Par exemple, cette distribution limite π correspond au Pagerank [13] mesuré par le moteur de recherche Google (graphes de pages Web reliées par les liens hypertexte orientés). Les graphes dans ce contexte ne sont pas fortement connexes mais ils sont modifiés en ajoutant un sommet virtuel doublement relié à tous les sommets. Ceci permet de rendre artificiellement le graphe fortement connexe.

Les marches aléatoires peuvent donc, dans l'hypothèse de la connexité forte du graphe, retrouver des propriétés limites similaires au cas non-orienté. Cependant les comportements transitoires observés pour des marches de faible longueur sont beaucoup plus difficiles à appréhender. Nous interprétons dans notre approche de détection de communautés la possibilité d'aller d'un sommet à un autre par une marche aléatoire comme une notion de similarité ou de proximité. Dans le cas des graphes orientés, cette notion de similarité ne serait plus symétrique (un sommet i peut être proche de j car il existe un arc de i à j mais ce même sommet j pourrait être très éloigné de i car il n'existe pas de chemin court de j vers i), ceci rend délicate l'interprétation de communautés dans lesquelles nous essayons de regrouper des sommets similaires.

De plus, l'orientation des arcs peut avoir des significations très différentes selon l'objet que l'on modélise par un graphe. Il est souvent, dans le contexte de détection de communautés, plus pratique d'ignorer les orientations des liens et de considérer que le graphe est non orienté. Il faut tout de même veiller lorsque l'on effectue cette transformation à ne pas perdre trop d'information sur l'objet modélisé par le graphe. Par exemple, si l'on considère un réseau d'appels téléphoniques, nous pouvons ignorer le sens des appels en considérant qu'une fois l'appel effectué les deux interlocuteurs sont en communication symétrique. Mais contrairement, si l'on considère un réseau de flux financiers entre sociétés, l'orientation du flux est une information primordiale qui ne peut pas être ignorée sans altérer le sens de la modélisation.

Les marches aléatoires dans un graphe orienté conservent leur propriété de distribution stationnaire sous la condition de forte connexité du graphe. Cependant les comportements des marches aléatoires de faible longueur diffèrent grandement et nous empêchent d'utiliser cette approche dans le contexte de détection de communautés.
Il est souvent possible de transformer un graphe orienté en un graphe non-orienté mais il faut être vigilant à l'impact de la perte d'information sur la modélisation de l'objet étudié par un graphe.

Chapitre 3

Détection de communautés

Ce chapitre est consacré à une méthode de détection de communautés basée sur des marches aléatoires dans les graphes. Nous allons dans un premier temps introduire une distance qui mesure la proximité structurelle des sommets d'un graphe. Cette distance est basée sur l'analyse des marches aléatoires qui tendent à être piégées dans les zones denses que constituent les communautés. Nous utilisons alors cette distance dans un algorithme de clustering hiérarchique pour trouver une structure hiérarchique de communautés. Cette méthode regroupe successivement les communautés les plus proches pour former une structure hiérarchique. Le résultat peut être vu comme une structure arborescente, appelée dendrogramme, entre les communautés qui se divisent en sous-communautés. Nous utiliserons cette structure hiérarchique dans le chapitre suivant pour déterminer les partitions les plus significatives qui maximisent les différentes fonctions de qualité envisageables.

3.1 Marches aléatoires et similarité des sommets

Nous considérons un graphe non-orienté pondéré $G = (V, E, w)$ et nous allons définir dans cette partie une distance entre sommets qui sera la base de notre approche de détection de communautés. Cette distance basée sur les marches aléatoires sera analysée et reliée aux propriétés spectrales du processus de Markov sous-jacent, ce qui permettra de relier notre approche à d'autres approches existantes.

3.1.1 Définition d'une distance entre sommets

Pour garantir l'apériodicité du graphe G nous rajoutons une boucle sur chaque sommet. En effet, ces boucles sont des cycles de longueur 1, le pgcd des cycles sera donc forcément 1. Cette condition permet de garantir les propriétés des marches aléatoires montrées au chapitre précédent. Notons de plus que ces boucles permettent de prendre en compte un lien naturel de similarité entre un sommet et lui même. Cet ajout de boucles n'est pas indispensable pour le bon fonctionnement général de notre approche, mais nous avons constaté qu'il procurait une légère amélioration des performances. Pour un graphe pondéré, nous choisissons un poids w_{ii} pour les boucles de ces liens égal au poids moyen des liens du graphe : $\forall k \in V, w_{kk} = \frac{1}{|E|} \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij}$.

Nous cherchons à détecter des communautés dans les graphes, c'est à dire des zones denses fortement reliées entre elles et plus faiblement reliées vers l'extérieur. L'intuition initiale de

l'utilisation des marches aléatoires est basée sur l'idée que les marches aléatoires vont se faire piéger dans ces zones denses. Cette intuition a déjà été introduite par Bruno Gaume [30] pour calculer une proximité sémantique de mots dans des réseaux linguistiques. Plus précisément, lorsqu'un marcheur sera dans une communauté il possédera une forte probabilité de rester dans la même communauté à l'étape suivante (grâce à la forte densité de liens internes et la faible densité de liens externes). Ainsi un marcheur possède de grandes chances de rester lors d'une marche de courte distance dans sa communauté d'origine. L'idée pour comparer la proximité de deux sommets est alors de comparer les distributions de probabilité des marches aléatoires partant de ces deux sommets. Nous nous basons alors sur le principe que deux marches qui partent d'une même communauté vont avoir des comportements similaires et se concentrer sur les mêmes zones du graphe.

Considérons des marches aléatoires d'une longueur donnée t. Chaque probabilité de transition P^t_{ij} nous donne de l'information sur les sommets i et j et leur proximité par une marche aléatoire de longueur t. La Propriété 1 de réversibilité des marches aléatoires nous indique que les probabilités P^t_{ij} et P^t_{ji} sont directement reliées ; elles sont donc porteuses de la même information. Nous pouvons donc considérer que toute l'information des marches aléatoires concernant un sommet donné $i \in V$ est contenue dans les probabilités $(P^t_{ik})_{k \in V}$. Ces probabilités correspondent à la $i^{\text{ème}}$ ligne de la matrice P^t, et nous noterons ces probabilités par un vecteur colonne $P^t_{i\bullet}$ défini comme suit.

Définition 2 *Chaque sommet i est caractérisé par le vecteur colonne de probabilité de position $P^t_{i\bullet}$ donné par des marches aléatoires de longueur t partant de i :*

$$\forall k \in V, P^t_{i\bullet}(k) = P^t_{ik}$$

Notons que le vecteur $P^t_{i\bullet}$ n'est autre que le vecteur de probabilité de position $\rho(t)$ donné par une position initiale $\rho(0)$ partant d'un unique sommet i ($\rho_i(0) = 1$ et $\rho_j(0) = 0$ pour $i \neq j$).

La longueur t des marches considérées doit être assez grande pour que la marche puisse atteindre suffisamment de sommets et ne pas dépendre uniquement du voisinage immédiat du sommet de départ. Cependant la longueur ne doit pas non plus être trop grande car les probabilités de transition vont dans ce cas tendre vers la probabilité stationnaire limite (Propriété 2) qui ne dépend que des poids des sommets. Nous perdrions ainsi toute l'information locale autour du sommet de départ et sur ses communautés. Le choix délicat de la longueur t des marches aléatoires sera discuté plus en détail au paragraphe 3.1.4.

Pour définir une distance entre deux sommets i et j, nous allons donc comparer les deux vecteurs de probabilité $P^t_{i\bullet}$ et $P^t_{j\bullet}$ en utilisant les remarques suivantes :

- Si deux sommets i et j sont dans la même communauté, la probabilité P^t_{ij} sera sûrement élevée, cependant une probabilité P^t_{ij} importante ne signifie pas forcément que i et j sont dans une même communauté. La Figure 2.1 (page 28) illustre bien ce point (notons que l'absence de boucles de cet exemple n'a que peu d'impact sur cette propriété).
- La probabilité P^t_{ij} est influencée par le poids $w(j)$ du sommet d'arrivée. Dans le cas limite où t tend vers l'infini, cette probabilité est proportionnelle à ce poids. De manière générale il est plus facile d'atteindre les sommets de fort poids par une marche aléatoire.
- Deux sommets i et j d'une même communauté tendent à "voir" les autres sommets de la même manière. En effet les marches à très courte distance vont principalement s'étendre dans leur communauté initiale puis diffuser à partir de là vers le reste du graphe. Nous pouvons supposer que deux sommets structurellement proches i et j vont atteindre les autres sommets avec des probabilités similaires ($\forall k \in V, P^t_{ik} \simeq P^t_{jk}$).

Ces considérations nous permettent de définir une distance r_{ij} entre les sommets du graphe qui prend en compte toutes les remarques que nous venons de faire :

Définition 3 *Soit i et j deux sommets du graphe, nous définissons la distance r_{ij} entre les deux sommets par :*

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{w(k)}} = \left\| D^{-\frac{1}{2}} P_{i\bullet}^t - D^{-\frac{1}{2}} P_{j\bullet}^t \right\|$$

où $\|.\|$ est la norme Euclidienne de $\mathbb{R}^n$

Remarquons que cette distance peut aussi être vue comme la distance L^2 définie par [5] entre les deux vecteurs de probabilité $P_{i\bullet}^t$ et $P_{j\bullet}^t$. Notons aussi que cette distance dépend du paramètre t de longueur des marches choisi. Nous devrions donc de manière plus rigoureuse utiliser la notation $r_{ij}(t)$, cependant dans une optique de simplification des notations nous considérerons cette dépendance comme implicite et utiliserons r_{ij} au lieu de $r_{ij}(t)$. De plus il est important de remarquer que la distance r ainsi définie est une distance Euclidienne vue sur l'ensemble $\mathbb{R}^n$ contenant les vecteurs de probabilité $P_{i\bullet}^t$. Cette propriété est cruciale pour les Théorèmes 4 et 5 qui permettront des calculs efficaces dans le processus de clustering hiérarchique.

Nous avons pondéré dans la distance l'influence de chaque probabilité P_{ik}^t par $\frac{1}{w(k)}$, ce qui permet d'équilibrer les contributions de chaque sommet, car sinon les sommets de plus fort poids (qui attirent plus facilement les marches) auraient une importance prépondérante dans le calcul de la distance.

Deux sommets seront, selon cette définition, d'autant plus proches que le comportement des marches aléatoires partant d'eux seront similaires. Il est à noter que nous ne pouvons pas comparer des distances entre sommets qui seraient basées sur deux longueurs t différentes. En effet la distribution limite de marches aléatoires implique que toutes les distances r_{ij} tendent vers 0 lorsque t tend vers l'infini : les ordres de grandeur des r_{ij} dépendent donc directement de t.

Nous définissons une distance r_{ij} entre toute paire de sommets i et j du graphe qui utilise des marches aléatoires d'une longueur fixée t :

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{w(k)}} = \left\| D^{-\frac{1}{2}} P_{i\bullet}^t - D^{-\frac{1}{2}} P_{j\bullet}^t \right\|$$

Cette distance est d'autant plus faible que les marches aléatoires partant de i et j ont des comportements similaires. Elle traduit donc une proximité structurelle des sommets. Nous allons montrer dans les sections suivantes qu'elle a le double avantage de pouvoir être calculée efficacement et de tirer profit de propriétés spectrales connues.

3.1.2 Propriétés spectrales de la distance r

Nous allons relier la distance r que nous venons de définir aux propriétés spectrales de la matrice de transition P. Ceci nous permettra d'établir un lien fort entre la distance r proposée

et les autres approches basées sur les propriétés spectrales du graphe proposées précédemment [19, 20, 47, 27, 67].

Théorème 2 *La distance r est reliée aux valeurs propres λ_α et aux vecteurs propres (à droite) v_α de la matrice de transition P par :*

$$r_{ij}^2 = \sum_{\alpha=2}^{n} \lambda_\alpha^{2t} (v_\alpha(i) - v_\alpha(j))^2$$

Preuve : Pour prouver ce théorème, nous partons du Lemme 1 et de la décomposition spectrale de la matrice P qu'il permet :

$$P_{ij}^t = \sum_{\alpha=1}^{n} \lambda_\alpha^t v_\alpha(i) u_\alpha(j)$$

où v_α et u_α sont respectivement des vecteurs propres à droite et à gauche de la matrice P. Ils sont définis à l'aide de la famille de vecteurs orthonormés $(s_\alpha)_{1\leq\alpha\leq n}$ par $v_\alpha = D^{-\frac{1}{2}} s_\alpha$ et $u_\alpha = D^{\frac{1}{2}} s_\alpha$. Ceci conduit à l'expression du vecteur $P_{i\bullet}^t$ suivante :

$$P_{i\bullet}^t = \sum_{\alpha=1}^{n} \lambda_\alpha^t v_\alpha(i) u_\alpha = D^{\frac{1}{2}} \sum_{\alpha=1}^{n} \lambda_\alpha^t v_\alpha(i) s_\alpha$$

Nous avons vu dans la preuve de la Propriété 2 que le vecteur propre v_1 associé à la valeur propre principale $\lambda_1 = 1$ est un vecteur constant. Ceci assure que le premier terme de la somme associé à $\alpha = 1$ est nul. La somme vaut donc :

$$P_{i\bullet}^t = D^{\frac{1}{2}} \sum_{\alpha=2}^{n} \lambda_\alpha^t v_\alpha(i) s_\alpha$$

La distance r_{ij} définie à l'aide la norme Euclidienne a donc pour expression :

$$r_{ij} = \left\| D^{-\frac{1}{2}} P_{i\bullet}^t - D^{-\frac{1}{2}} P_{j\bullet}^t \right\| = \left\| \sum_{\alpha=2}^{n} \lambda_\alpha^t v_\alpha(i) s_\alpha - \sum_{\alpha=2}^{n} \lambda_\alpha^t v_\alpha(j) s_\alpha \right\| = \left\| \sum_{\alpha=2}^{n} \lambda_\alpha^t (v_\alpha(i) - v_\alpha(j)) s_\alpha \right\|$$

La famille de vecteurs $(s_\alpha)_{1\leq\alpha\leq n}$ étant orthonormée, nous pouvons appliquer le théorème de Pythagore et obtenir :

$$r_{ij}^2 = \sum_{\alpha=2}^{n} \left\| \lambda_\alpha^t (v_\alpha(i) - v_\alpha(j)) s_\alpha \right\|^2 = \sum_{\alpha=2}^{n} \lambda_\alpha^{2t} (v_\alpha(i) - v_\alpha(j))^2$$

□

Ce théorème montre que la distance r dépend principalement des vecteurs propres associés aux plus grandes valeurs propres. En effet, dans la somme les termes sont pondérés par λ_α^{2t}, ce qui donne un poids plus important aux valeurs propres les plus grandes. Ceci relie la distance aux approches spectrales qui sont basées sur le fait que deux sommets proches (d'une même communauté) ont des composantes similaires sur les vecteurs propres principaux. Nous allons étudier ce lien dans le paragraphe suivant.

La distance r est reliée aux propriétés spectrales de la matrice de transition P par :

$$r_{ij}^2 = \sum_{\alpha=2}^{n} \lambda_\alpha^{2t} (v_\alpha(i) - v_\alpha(j))^2$$

Elle dépend donc principalement des vecteurs propres v_α associés au plus grandes valeurs propres λ_α.

3.1.3 Liens avec les autres approches spectrales

Le théorème précédent relie les marches aléatoires avec plusieurs travaux qui utilisent des propriétés spectrales des graphes pour la détection de structures de communautés.

Tout d'abord citons les travaux de Stéphane Lafon *et al* [51, 56] sur des problèmes de diffusion appliqués au clustering de données. Ces travaux, qui ne sont apparus que très récemment, ont été effectués de manière indépendante et simultanée à nos travaux. Un processus de diffusion dans un graphe équivalent à notre processus de marches aléatoires y est défini. La même distance r_{ij} est aussi définie sous le nom de "distance de diffusion" et le Théorème 2 liant la distance aux propriétés spectrales y est aussi montré. Cette distance a été utilisée dans des méthodes de clustering de données, problème plus général que la détection de communautés dans les graphes. Ces travaux indépendants et simultanés qui ont conduit à la découverte des mêmes résultats, justifient doublement la qualité de la distance r_{ij} proposée qui possède des atouts indéniables dans une thématique de recherche émergente.

De nombreuses études de détection de communautés se sont basées sur les propriétés spectrales des graphes. Par exemple, Simonsen [76] remarque que la structure modulaire d'un graphe est exprimée dans les vecteurs propres (autre que v_1) de la matrice P correspondant aux plus grandes valeurs propres. Cette expression implique que deux sommets i et j d'une même communauté vont avoir des composantes similaires sur les vecteur propres v_α ($v_\alpha(i) \simeq v_\alpha(j)$) pour les plus grandes valeurs propres λ_α.

De plus Schulman *et al* [31, 73] montrent dans un cadre plus théorique que lorsqu'une valeur propre λ_α tend vers 1, les coordonnées du vecteur propre v_α associé tendent à être constantes sur des sous-ensembles de sommets qui correspondent aux communautés. Une distance similaire à la nôtre a aussi été introduite : $d_t^2(i,j) = \sum_{\alpha=2}^{n} \frac{(v_\alpha(i)-v_\alpha(j))^2}{1-|\lambda_\alpha|^t}$. Cependant cette distance ne peut pas être calculée à partir de marches aléatoires et nécessite un calcul de valeurs et vecteurs propres.

Finalement, Donetti et Muñoz [19] utilisent une approche similaire en définissant des distances basées sur les vecteurs propres de la matrice Laplacienne du graphe : $L = D - A$. Ces distances sont ensuite utilisées dans un algorithme de clustering hiérarchique. Leur approche est améliorée dans [20] en utilisant une version normalisée de la matrice Laplacienne du graphe. Une étude rapide permet de montrer que les vecteurs propres de cette matrice normalisée sont les mêmes que ceux de la matrice de transition P que nous utilisons.

Toutes ces études montrent que la distance r que nous avons définie est reliée à des propriétés spectrales reconnues et utilisées dans d'autres approches de détection de communautés. Cependant notre approche possède le grand avantage de tirer partie de ces propriétés spectrales sans avoir recours à un calcul explicite de valeurs et vecteurs propres. Toutes les autres approches qui reposent sur ce calcul sont pénalisées en terme de complexité car la détermi-

nation des valeurs et vecteurs propres d'une matrice creuse nécessite un temps de calcul en $\mathcal{O}(n^3)$. Cette complexité peut rapidement devenir inabordable dès lors que la taille du graphe dépasse quelques milliers de sommets. Notre approche permet de tirer profit de ces propriétés spectrales tout en gardant une complexité raisonnable grâce aux calculs de marches aléatoires (une analyse de complexité de notre approche sera proposée à la Section 3.2.2).

Notre approche basée sur des marches aléatoires permet de tirer profit de propriétés spectrales reconnues dans le domaine de détection de communautés tout en s'affranchissant du calcul pénalisant de vecteurs et valeurs propres. Ceci constitue un avantage par rapport aux autres approches basées sur des propriétés spectrales.

3.1.4 Choix de la longueur t des marches

Nous avons défini une distance r entre sommets basée sur des marches aléatoires d'une longueur fixée t. Nous allons maintenant discuter du choix de cette longueur de marche t. Il s'agit en fait d'un point complexe qui n'est pas totalement résolu, cependant en pratique la longueur t ne possède pas une influence prépondérante sur les résultats.

Les marches aléatoires tendent vers une distribution stationnaire lorsque leur longueur t tend vers l'infini (Propriété 2). Ainsi si nous prenons des marches trop longues qui s'approchent de cette distribution stationnaire, les vecteurs de probabilités $P^t_{i\bullet}$ tendent tous vers la même distribution stationnaire π quel que soit le sommet de départ i. Nous pourrions donc difficilement comparer deux sommets i et j qui posséderaient des vecteurs de position $P^t_{i\bullet}$ et $P^t_{j\bullet}$ très proches. Les distributions de probabilité convergent de manière exponentielle vers la distribution limite π, ceci peut se voir grâce à la décomposition spectrale qui découle du Lemme 1. Ce lemme permet également de montrer que la vitesse de convergence dépend aussi du trou spectral $\lambda_1 - \lambda_2$ entre la valeur propre principale $\lambda_1 = 1$ et la seconde plus grande valeur propre. Nous devons donc choisir une distance t courte.

De manière opposée, si nous choisissons une longueur t des marches trop courte, l'information recueillie par les vecteurs de probabilité $P^t_{i\bullet}$ ne concernera que les sommets les plus proches du point de départ. Il faut donc aussi veiller à choisir une longueur de marche suffisamment longue pour "voir" plus loin que les sommets à proximité immédiate du sommet de départ.

Nous avons l'intuition que le choix de la longueur t la plus adaptée dépend de la taille des structures de communautés que nous cherchons à détecter. Nous pouvons en effet définir des longueurs caractéristiques telles que la distance moyenne ou le diamètre des communautés que nous voulons trouver. Une marche aléatoire partant d'un sommet intérieur à cette communauté va dans les premiers pas correspondant à cette longueur caractéristique principalement s'étendre dans la communauté et y rester "piégée". Dans un second temps, le marcheur va petit à petit arriver à quitter la communauté de départ pour "diffuser" depuis la communauté d'origine.

Si nous voulons comparer deux sommets d'une même communauté il faut laisser le temps aux marches qui partent de ces deux sommets de s'étendre dans la communauté. L'intuition peut alors être que, piégées dans cette structure locale, les marches vont atteindre rapidement un distribution stationnaire locale (à la manière du cas global) qui ne dépendra que de la

communauté. Une fois cette position atteinte, la marche diffuse dans le reste du graphe à une vitesse plus faible. Dans ces conditions les vecteurs position obtenus seront similaires, nous en concluons donc qu'il faut choisir des longueurs de marches de l'ordre des longueurs caractéristiques des structures que nous voulons détecter (deux fois la distances moyenne par exemple).

Le problème majeur est que nous ne connaissons pas a priori les tailles des structures de communautés que nous souhaitons découvrir. Le second problème est que nous pouvons être confrontés à des structures de communautés d'échelles différentes dans un même graphe. Ces structures de tailles différentes peuvent apparaître dans des parties disjointes du graphe ou même se superposer. Dans ces conditions le choix optimal de t pourrait être différent selon le sommet considéré, ceci n'est malheureusement pas faisable car il n'est pas possible de comparer des vecteurs de probabilités $P^t_{i\bullet}$ pour des valeurs de t différentes. En effet les vecteurs convergeant vers la distribution limite π à la même vitesse, deux marches de longueurs différentes auront atteint des niveaux de diffusion différents qui ne pourront pas êtres comparés directement par leur distance Euclidienne.

Nous avons cependant constaté expérimentalement que les variations de performances liées au choix de t étaient relativement faibles. Mis à part les choix de longueurs manifestement trop courtes ($t = 1$ ou $t = 2$) et les choix de longueurs bien trop grandes (plusieurs fois la distance moyenne du graphe entier), les performances restent relativement stables pour tout choix de t entre ces deux extrêmes. En pratique, nous obtenons souvent des résultats similaires pour des marches de longueur entre 3 et 10. Nous suggérons donc aux utilisateurs de notre méthode de prendre une longueur t intermédiaire qui permettra de tomber dans ce palier pour la plupart des graphes. Nous avons en pratique fixé le paramètre par défaut de l'implémentation que nous proposons à $t = 5$, ce paramètre est un bon compromis qui conduit le plus souvent à de très bons résultats. Une étude expérimentale de l'influence de la longueur des marches sera proposée en 5.4.

> Intuitivement, la meilleure longueur t des marches aléatoires est de l'ordre de grandeur des distances moyennes des structures de communautés que nous souhaitons détecter. Il n'est cependant pas possible de connaître a priori ces longueurs caractéristiques qui de plus peuvent varier d'un sommet à l'autre pour un même graphe. Cette problématique reste un sujet de recherche intéressant.
> En pratique nous avons observé que la longueur t a peu d'influence sur les performances dès lors qu'elle n'est ni exagérément trop courte ($t < 3$) ou exagérément trop longue ($t > 10$). Nous suggérons aux utilisateurs de notre méthode de prendre une longueur $t = 5$ qui semble expérimentalement conduire à de bons résultats dans la plupart des cas.

3.1.5 Généralisations de la distance

Nous avons défini une distance r_{ij} entre sommets, nous allons dans un premier temps généraliser cette distance aux communautés (c'est à dire entre des sous-ensembles de sommets). Nous proposerons dans un second temps une méthode qui permet de définir et de calculer efficacement d'autres distances basées sur les propriétés spectrales du graphe.

Distance entre communautés

Notre approche de clustering hiérarchique (détaillée en 3.2) nécessitera de mesurer la proximité entre sommets mais aussi entre communautés (c'est à dire des sous ensembles de sommets de V). Nous allons généraliser l'approche en utilisant des marches aléatoires partant des communautés pour les comparer. Il suffit pour cela de considérer des marches qui partent uniformément de l'ensemble d'une communauté plutôt que d'un sommet unique, nous définissons pour cela le vecteur de probabilités $P^t_{\mathcal{C}\bullet}$ qui caractérisera une communauté $\mathcal{C}$:

Définition 4 *Soit une communauté de sommets $\mathcal{C} \subset V$, nous définissons le vecteur de probabilité de position $P^t_{\mathcal{C}\bullet}$ correspondant à une marche de longueur t partant uniformément des sommets de $\mathcal{C}$:*

$$P^t_{\mathcal{C}\bullet} = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} P^t_{i\bullet}$$

Nous notons que ce vecteur $P^t_{\mathcal{C}\bullet}$ correspond au vecteur position $\rho(t)$ d'une marche aléatoire partant uniformément des sommets de la communauté $\mathcal{C}$, c'est à dire avec une position initiale $\rho(0)$ satisfaisant $\rho_i(0) = \frac{1}{|\mathcal{C}|}$ pour $i \in \mathcal{C}$ et $\rho_i(0) = 0$ pour $i \notin \mathcal{C}$. La relation avec les vecteurs de position $P^t_{i\bullet}$ découle du fait que le position initiale considérée est une combinaison linéaire des positions initiales des marches partant de chaque sommet de la communauté $\mathcal{C}$.

Nous pouvons ainsi définir une distance $r_{\mathcal{C}_1\mathcal{C}_2}$ entre communautés qui généralise la distance entre sommets :

Définition 5 *Nous généralisons la distance r entre deux communautés $\mathcal{C}_1$ et $\mathcal{C}_2$ par :*

$$r_{\mathcal{C}_1\mathcal{C}_2} = \left\| D^{-\frac{1}{2}} P^t_{\mathcal{C}_1\bullet} - D^{-\frac{1}{2}} P^t_{\mathcal{C}_2\bullet} \right\|$$

Ceci permet aussi de définir le cas particulier d'une distance entre un sommet i et une communauté $\mathcal{C}$:

$$r_{i\mathcal{C}} = \left\| D^{-\frac{1}{2}} P^t_{\mathcal{C}\bullet} - D^{-\frac{1}{2}} P^t_{i\bullet} \right\|$$

Cette distance est bien une généralisation de la distance r_{ij} car lorsque nous prenons le cas particulier de communautés contenant un seul sommet, $\mathcal{C}_1 = \{i\}$ et $\mathcal{C}_1 = \{j\}$, nous retrouvons bien r_{ij}.

Autres distances spectrales

Nous avons vu que notre distance r est directement reliée aux propriétés spectrales de la matrice de transition P (Théorème 2). Nous proposons dans ce paragraphe une approche basée sur des marches aléatoires qui permet de créer d'autres distances utilisant des pondérations différentes des vecteurs propres. Nous allons pour cela définir des vecteurs modifiés $\widehat{P}_{i\bullet}$ qui seront utilisés de manière similaire pour définir une distance $\widehat{r}_{ij}$.

Théorème 3 *Soit f une fonction définie par une série entière : $f(x) = \sum_{k=0}^{\infty} c_k x^k$. Nous définissons le vecteur $\widehat{P}_{i\bullet} = \sum_{k=0}^{\infty} c_k P^k_{i\bullet}$ associé qui permet de définir la distance $\widehat{r}_{ij}$:*

$$\widehat{r}_{ij} = \left\| D^{-\frac{1}{2}} \widehat{P}_{i\bullet} - D^{-\frac{1}{2}} \widehat{P}_{j\bullet} \right\|$$

Cette distance est reliée aux propriétés spectrales de P par :

$$\widehat{r}_{ij}^2 = \sum_{\alpha=2}^{n} f^2(\lambda_\alpha)(v_\alpha(i) - v_\alpha(j))^2$$

Preuve : Utilisons l'expression de $P_{i\bullet}^k = D^{\frac{1}{2}} \sum_{\alpha=1}^{n} \lambda_\alpha^k v_\alpha(i) s_\alpha$ découlant du Lemme 1, nous avons :

$$\widehat{P}_{i\bullet} = \sum_{k=0}^{\infty} c_k P_{i\bullet}^k = D^{\frac{1}{2}} \sum_{k=0}^{\infty} \sum_{\alpha=1}^{n} c_k \lambda_\alpha^k v_\alpha(i) s_\alpha$$

En utilisant cette expression dans la définition de $\widehat{r}_{ij}$, nous obtenons :

$$\widehat{r}_{ij} = \left\| D^{-\frac{1}{2}} \widehat{P}_{i\bullet} - D^{-\frac{1}{2}} \widehat{P}_{j\bullet} \right\| = \left\| \sum_{k=0}^{\infty} \sum_{\alpha=2}^{n} c_k \lambda_\alpha^k (v_\alpha(i) - v_\alpha(j)) s_\alpha \right\|$$

Notons que le terme correspondant à $\alpha = 1$ disparaît car le vecteur v_1 est un vecteur constant. Nous concluons en utilisant le théorème de Pythagore sur la famille de vecteurs orthonormés $(s_\alpha)_{1 \leq \alpha \leq n}$:

$$\widehat{r}_{ij}^2 = \sum_{\alpha=2}^{n} \left\| \sum_{k=0}^{\infty} c_k \lambda_\alpha^k (v_\alpha(i) - v_\alpha(j)) s_\alpha \right\|^2 = \sum_{\alpha=2}^{n} f^2(\lambda_\alpha)(v_\alpha(i) - v_\alpha(j))^2$$

□

Ce théorème permet de considérer d'autres types de pondération des vecteurs propres grâce au choix de la fonction f. Pour calculer la distance, nous estimons les vecteurs $\widehat{P}_{i\bullet} = \sum_{k=0}^{\infty} c_k P_{i\bullet}^k$ en calculant la somme jusqu'à un certain rang r par : $\widehat{P}_{i\bullet} \simeq \sum_{k=0}^{r} c_k P_{i\bullet}^k$. L'erreur commise sur chaque coordonnée est alors inférieure à $\varepsilon_r = \frac{1}{f(1)} \sum_{k=r+1}^{\infty} c_k$. En effet chaque coordonnée des vecteurs $P_{i\bullet}^k$ est inférieure à 1.

La complexité du calcul de distance est donc en $\mathcal{O}(rm)$ car il faut calculer successivement les r premiers vecteurs $P_{i\bullet}^k$, chacun en un temps $\mathcal{O}(m)$ (voir 2.3.2).

Cette analyse fait le lien avec d'autres distances proposées qui utilisent les vecteurs propres de la matrice P. Nous définissons ainsi une large famille de distances, mais nous devons cependant noter que la distance r_{ij} de base possède l'avantage de la simplicité. Nous n'avons pas trouvé d'autre distance de cette famille qui apporte un avantage suffisant par rapport à la distance r pour justifier le surcroît de complexité.

Nous définissons une distance $r_{\mathcal{C}_1\mathcal{C}_2}$ entre communautés généralisant la distance r_{ij} :

$$r_{\mathcal{C}_1\mathcal{C}_2} = \left\| D^{-\frac{1}{2}} P^t_{\mathcal{C}_1 \cdot} - D^{-\frac{1}{2}} P^t_{\mathcal{C}_2 \cdot} \right\| \text{ où } P^t_{\mathcal{C}\cdot} = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} P^t_{i\cdot}$$

Il est aussi possible de définir à l'aide de marches aléatoires des distances $\widehat{r}_{ij}$ vérifiant :

$$\widehat{r}^2_{ij} = \sum_{\alpha=2}^{n} f^2(\lambda_\alpha)(v_\alpha(i) - v_\alpha(j))^2$$

pour toute fonction $f(x) = \sum_{k=0}^{\infty} c_k x^k$ définie par une série entière.

3.2 L'algorithme de clustering hiérarchique

La section précédente a permis de définir une distance r entre sommets et entre communautés qui mesure leurs similarités structurelles dans le graphe. Sachant quels sont les sommets proches grâce à la distance r, le problème de détection de communautés est donc maintenant un problème de clustering pour regrouper les sommets similaires. Nous allons proposer une approche agglomérative basée sur une adaptation de la méthode de Ward [81] de clustering hiérarchique.

3.2.1 L'algorithme de clustering hiérarchique

Nous cherchons à regrouper les sommets similaires selon la distance r en communautés. Nous utilisons une approche de clustering hiérarchique agglomérative dans laquelle nous partons d'une partition initiale $\mathcal{P}^0 = \{\{v\}, v \in V\}$ contenant n communautés composées d'un seul sommet. Nous réalisons alors n étapes successives de fusion de communautés qui créent une structure hiérarchique arborescente (aussi appelée dendrogramme). Le choix des communautés à fusionner se base sur les distances r_{ij} entre sommets que nous avons définies précédemment.

Principe général

Nous considérons une partition initiale en communauté $\mathcal{P}^0 = \{\{v\}, v \in V\}$ formée des n communautés composées d'un seul sommet. Nous allons générer une suite de partitions $(\mathcal{P}^k)_{0 \leq k \leq n}$ en fusionnant successivement des communautés deux à deux. Le choix des communautés à fusionner repose sur les distance r entre les communautés adjacentes de la partition courante. Nous effectuons donc un calcul initial des distances r_{ij} entre chaque paire de sommets adjacents, ceci correspond donc à m distances calculées. Un processus itératif est alors répété jusqu'à l'obtention d'une partition finale $\mathcal{P}^n = \{V\}$ ne possédant qu'une seule communauté regroupant tous les sommets du graphe. Chaque étape consiste à faire évoluer la partition courante $\mathcal{P}^k$ vers la partition $\mathcal{P}^{k+1}$ selon la démarche :

- Choisir deux communautés $\mathcal{C}_1$ et $\mathcal{C}_2$ de $\mathcal{P}^k$. Ce choix est détaillé au paragraphe suivant.
- Fusionner ces deux communautés en une nouvelle communauté $\mathcal{C}_3 = \mathcal{C}_1 \cup \mathcal{C}_2$ et créer la nouvelle partition $\mathcal{P}^{k+1} = (\mathcal{P}^k \backslash \{\mathcal{C}_1, \mathcal{C}_2\}) \cup \{\mathcal{C}_3\})$.

– Mettre à jour les distances r qui ont changé, c'est à dire les distances $r_{\mathcal{C}_3\mathcal{C}}$ pour toutes les communautés $\mathcal{C}$ adjacentes de la nouvelle communauté $\mathcal{C}_3$.

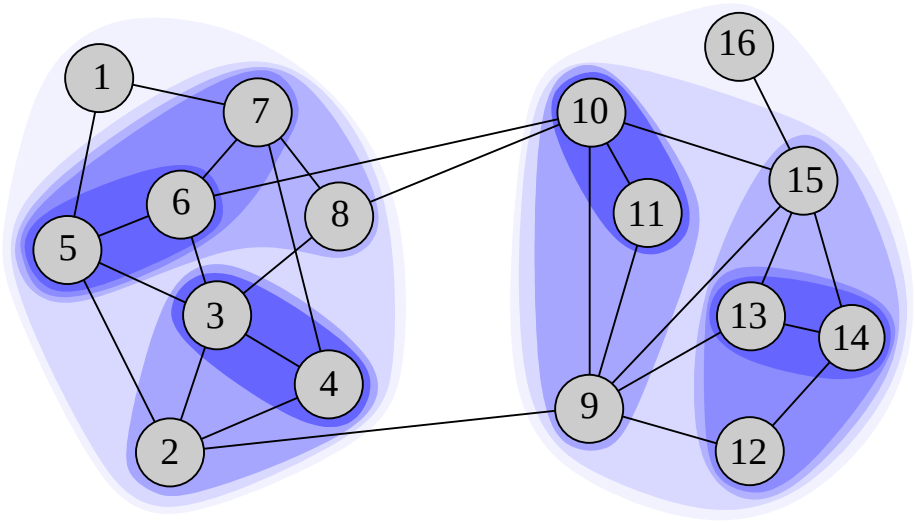

FIG. 3.1 – Exemple de structure hiérarchique de communautés touvée par notre algorithme. Le dendrogramme correspondant est dessiné en Figure 3.2.

Chaque étape fusionne deux communautés pour en créer une nouvelle. Ceci construit une hiérarchie entre communautés qui peut être représentée par un arbre binaire. La racine de l'arbre correspond à la plus grande communauté V et les feuilles de l'arbre correspondent à tous les sommets $v \in V$. Chaque fusion crée un noeud de l'arbre avec deux sous-communautés filles inclues dans une plus grosse communauté. Nous donnons une représentation graphique de cette structure (aussi appelée dendrogramme) dans la Figure 3.2.

L'approche classique pour déterminer la partition en communautés sortie par l'algorithme consiste à choisir la partition $\mathcal{P}^k$ parmi les $n+1$ partitions $(\mathcal{P}^k)_{0\leq k\leq n}$ générées qui maximise la fonction de qualité choisie. Nous proposerons au Chapitre 4 une meilleure approche permettant d'obtenir des partitions de meilleure qualité à partir du dendrogramme.

Choix des communautés à fusionner

Le point clé de l'algorithme est la manière de choisir les communautés à fusionner. Le nombre de fusions envisageables à chaque étape correspond au nombre de paires de communautés dans la partition courante $\mathcal{P}^k$. Pour limiter le nombre de cas à considérer, nous n'allons considérer que des fusions entre communautés adjacentes (c'est à dire des communautés liées par au moins une arête). Cette heuristique raisonnable (déjà utilisée dans [58] et [19]) limite le nombre possible de fusions à m pour chaque étape. Ceci évite avantageusement de considérer de nombreux cas, il est en effet fortement improbable que deux communautés non reliées génèrent des marches aléatoires similaires.

Nous utilisons la méthode de Ward pour choisir les communautés à fusionner. Cette méthode minimise à chaque étape k les distances internes des communautés. Cette distance est

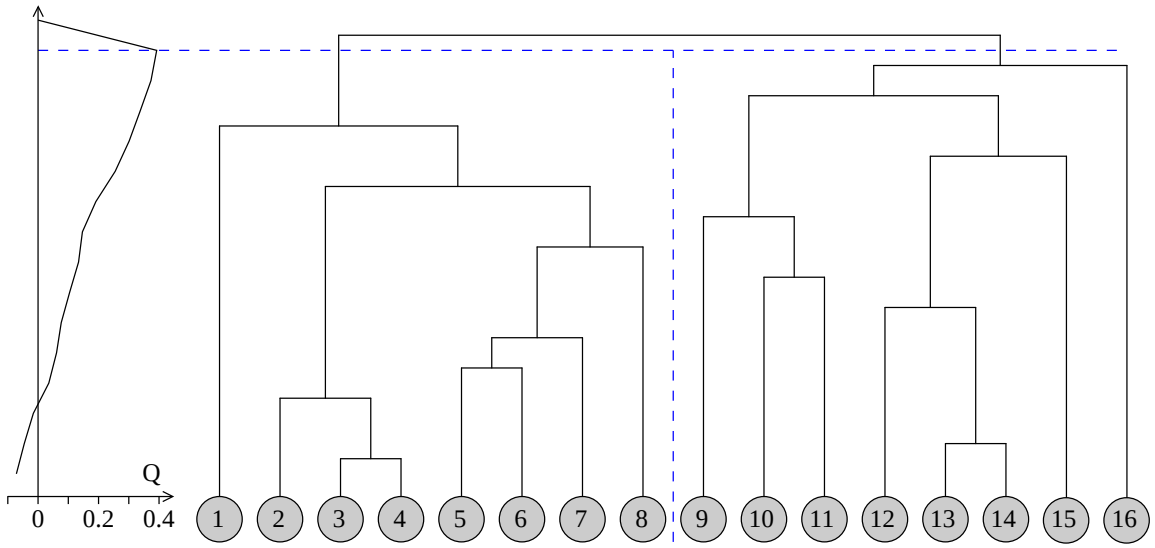

FIG. 3.2 – Dendrogramme associé aux communautés de la Figure 3.1. L'évolution de la fonction de qualité "modularité" Q (Sec. 4.2.1) donne la partition $\mathcal{P}^k$ maximisant Q comme une coupe horizontale du dendrogramme.

mesurée par la moyenne σ_k des distances au carré de chaque sommet à sa communauté :

$$\sigma_k = \frac{1}{n} \sum_{\mathcal{C} \in \mathcal{P}_k} \sum_{i \in \mathcal{C}} r_{i\mathcal{C}}^2$$

Ainsi σ_k mesure l'éloignement moyen des sommets à leur communauté : plus les communautés sont hétérogènes, plus σ_k sera élevé. Nous voulons donc obtenir des communautés homogènes en minimisant σ_k.

La méthode de Ward est une approche gloutonne qui choisit à chaque étape k la paire de communautés produisant la plus faible variation $\Delta\sigma_k = \sigma_{k+1} - \sigma_k$. Cette variation ne dépend que des communautés $\mathcal{C}_1$ et $\mathcal{C}_2$ choisies. Il suffit donc de calculer les variations $\Delta\sigma(\mathcal{C}_1, \mathcal{C}_2)$ correspondant aux fusions des paires $\{\mathcal{C}_1, \mathcal{C}_2\}$ de communautés adjacentes en une nouvelle communauté $\mathcal{C}_3 = \mathcal{C}_1 \cup \mathcal{C}_2$:

$$\Delta\sigma(\mathcal{C}_1, \mathcal{C}_2) = \frac{1}{n} \Big(\sum_{i \in \mathcal{C}_3} r_{i\mathcal{C}_3}^2 - \sum_{i \in \mathcal{C}_1} r_{i\mathcal{C}_1}^2 - \sum_{i \in \mathcal{C}_2} r_{i\mathcal{C}_2}^2 \Big)$$

Nous fusionnerons à chaque étape les communautés possédant le plus faible $\Delta\sigma$. Le paragraphe suivant montre comment calculer efficacement les $\Delta\sigma$.

Calcul des $\Delta\sigma$ et mise à jour des distances

L'efficacité de la méthode proposée repose sur la capacité à calculer efficacement les variations $\Delta\sigma(\mathcal{C}_1, \mathcal{C}_2)$. Un calcul direct de la variation serait bien sûr trop lourd. La distance r étant une distance Euclidienne, nous pouvons simplifier grandement l'expression de $\Delta\sigma(\mathcal{C}_1, \mathcal{C}_2)$ grâce au théorème suivant :

Théorème 4 [44] *La variation de σ_k après la fusion de deux communautés $\mathcal{C}_1$ et $\mathcal{C}_2$ est directement reliée à la distance $r_{\mathcal{C}_1\mathcal{C}_2}$ par :*

$$\Delta\sigma(\mathcal{C}_1,\mathcal{C}_2) = \frac{1}{n}\frac{|\mathcal{C}_1||\mathcal{C}_2|}{|\mathcal{C}_1|+|\mathcal{C}_2|}r^2_{\mathcal{C}_1\mathcal{C}_2}$$

Preuve : Rappelons tout d'abord la définition de $\Delta\sigma(\mathcal{C}_1,\mathcal{C}_2)$:

$$\Delta\sigma(\mathcal{C}_1,C_2) = \frac{1}{n}(\sum_{i\in C_3} r^2_{iC_3} - \sum_{i\in C_1} r^2_{iC_1} - \sum_{i\in C_2} r^2_{iC_2})$$

Commençons par réorganiser la première somme en utilisant le fait que $\mathcal{C}_3 = \mathcal{C}_1 \cup \mathcal{C}_2$ est une union disjointe :

$$\sum_{i\in\mathcal{C}_3} r^2_{i\mathcal{C}_3} = \sum_{i\in\mathcal{C}_1} r^2_{i\mathcal{C}_3} + \sum_{i\in\mathcal{C}_2} r^2_{i\mathcal{C}_3}$$

Notons $< .|. >$ le produit scalaire associé à la norme Euclidienne r. Ainsi par exemple $r^2_{i\mathcal{C}} =< P^t_{\mathcal{C}\bullet} - P^t_{i\bullet}|P^t_{\mathcal{C}\bullet} - P^t_{i\bullet} >$. Pour simplifier les notations, nous allons utiliser des notations vectorielles : posons pour toute communauté $\mathcal{C}$ et tout sommet i : $\overrightarrow{i\mathcal{C}} = P^t_{\mathcal{C}\bullet} - P^t_{i\bullet}$ et de même pour deux communautés $\overrightarrow{\mathcal{C}_1\mathcal{C}_2} = P^t_{\mathcal{C}_2\bullet} - P^t_{\mathcal{C}_1\bullet}$. Nous pouvons alors récrire :

$$\sum_{i\in\mathcal{C}_1} r^2_{i\mathcal{C}_3} = \sum_{i\in\mathcal{C}_1} < \overrightarrow{i\mathcal{C}_3}|\overrightarrow{i\mathcal{C}_3} >$$

Faisons intervenir le barycentre $P^t_{\mathcal{C}_1\bullet} = \frac{1}{|\mathcal{C}_1|}\sum_{i\in\mathcal{C}_1} P^t_{i\bullet}$ de l'ensemble de vecteurs $\{P^t_{i\bullet}|i\in\mathcal{C}_1\}$:

$$\sum_{i\in\mathcal{C}_1} r^2_{i\mathcal{C}_3} = \sum_{i\in\mathcal{C}_1} < \overrightarrow{i\mathcal{C}_1} + \overrightarrow{\mathcal{C}_1\mathcal{C}_3}|\overrightarrow{i\mathcal{C}_1} + \overrightarrow{\mathcal{C}_1\mathcal{C}_3} >$$

$$\sum_{i\in\mathcal{C}_1} r^2_{i\mathcal{C}_3} = \sum_{i\in\mathcal{C}_1}(< \overrightarrow{i\mathcal{C}_1}|\overrightarrow{i\mathcal{C}_1} > +2 < \overrightarrow{i\mathcal{C}_1}|\overrightarrow{\mathcal{C}_1\mathcal{C}_3} > + < \overrightarrow{\mathcal{C}_1\mathcal{C}_3}|\overrightarrow{\mathcal{C}_1\mathcal{C}_3} >)$$

$$\sum_{i\in\mathcal{C}_1} r^2_{i\mathcal{C}_3} = \sum_{i\in\mathcal{C}_1} r^2_{i\mathcal{C}_1} + 2 < \sum_{i\in\mathcal{C}_1} \overrightarrow{i\mathcal{C}_1}|\overrightarrow{\mathcal{C}_1\mathcal{C}_3} > +|\mathcal{C}_1| < \overrightarrow{\mathcal{C}_1\mathcal{C}_3}|\overrightarrow{\mathcal{C}_1\mathcal{C}_3} >$$

comme $P^t_{\mathcal{C}_1\bullet}$ est le barycentre de $\{P^t_{i\bullet}|i\in\mathcal{C}_1\}$ nous avons :

$$\sum_{i\in\mathcal{C}_1} \overrightarrow{i\mathcal{C}_1} = \overrightarrow{0}$$

De plus, $P^t_{\mathcal{C}_3\bullet}$ est le barycentre de $P_{\mathcal{C}_1\bullet}$ pondéré par $|\mathcal{C}_1|$ et de $P_{\mathcal{C}_2\bullet}$ pondéré par $|\mathcal{C}_2|$. Nous avons donc $\overrightarrow{\mathcal{C}_1\mathcal{C}_3} = \frac{|\mathcal{C}_2|}{|\mathcal{C}_1|+|\mathcal{C}_2|}\overrightarrow{\mathcal{C}_1\mathcal{C}_2}$, ce qui donne alors :

$$\sum_{i\in\mathcal{C}_1} r^2_{i\mathcal{C}_3} = \sum_{i\in\mathcal{C}_1} r^2_{i\mathcal{C}_1} + \frac{|\mathcal{C}_1||\mathcal{C}_2|^2}{(|\mathcal{C}_1|+|\mathcal{C}_2|)^2} < \overrightarrow{\mathcal{C}_1\mathcal{C}_2}|\overrightarrow{\mathcal{C}_1\mathcal{C}_2} >= \sum_{i\in\mathcal{C}_1} r^2_{i\mathcal{C}_1} + \frac{|\mathcal{C}_1||\mathcal{C}_2|^2}{(|\mathcal{C}_1|+|\mathcal{C}_2|)^2} r^2_{\mathcal{C}_1\mathcal{C}_2}$$

De manière symétrique nous obtenons pour $\mathcal{C}_2$:

$$\sum_{i\in\mathcal{C}_2} r^2_{i\mathcal{C}_3} = \sum_{i\in\mathcal{C}_2} r^2_{i\mathcal{C}_2} + \frac{|\mathcal{C}_2||\mathcal{C}_1|^2}{(|\mathcal{C}_1|+|\mathcal{C}_2|)^2} r^2_{\mathcal{C}_1\mathcal{C}_2}$$

Et finalement :

$$\sum_{i\in\mathcal{C}_3} r^2_{i\mathcal{C}_3} = \sum_{i\in\mathcal{C}_1} r^2_{i\mathcal{C}_3} + \sum_{i\in\mathcal{C}_2} r^2_{i\mathcal{C}_3} = \sum_{i\in\mathcal{C}_1} r^2_{i\mathcal{C}_1} + \sum_{i\in\mathcal{C}_2} r^2_{i\mathcal{C}_2} + \frac{|\mathcal{C}_1||\mathcal{C}_2|}{|\mathcal{C}_1|+|\mathcal{C}_2|} r^2_{\mathcal{C}_1\mathcal{C}_2}$$

Nous en déduisons donc une expression simple de $\Delta\sigma$:

$$\Delta\sigma(\mathcal{C}_1,\mathcal{C}_2) = \frac{1}{n}\frac{|\mathcal{C}_1||\mathcal{C}_2|}{|\mathcal{C}_1|+|\mathcal{C}_2|} r^2_{\mathcal{C}_1\mathcal{C}_2}$$

□

Ce théorème permet de calculer efficacement les variations $\Delta\sigma(\mathcal{C}_1,\mathcal{C}_2)$ en effectuant un seul calcul de distance r. Nous avons donc uniquement besoin de conserver et de garder à jour l'ensemble des distances entre communautés adjacentes. Ces distances $r_{\mathcal{C}_1\mathcal{C}_2}$ ne changent que lorsque l'une des deux communautés concernées est fusionnée. Nous n'avons donc qu'à calculer les distances $r_{\mathcal{C}_3\mathcal{C}}$ entre la nouvelle communauté crée $\mathcal{C}_3 = \mathcal{C}_1 \cup \mathcal{C}_2$ et toutes ses communautés voisines $\mathcal{C}$. Ces mises à jour de distances peuvent de plus être facilitées par le théorème suivant :

Théorème 5 (Lance-Williams-Jambu formula [44]) .
Si les communautés $\mathcal{C}_1$ et $\mathcal{C}_2$ sont fusionnées en $\mathcal{C}_3 = \mathcal{C}_1 \cup \mathcal{C}_2$ alors pour toute autre communauté $\mathcal{C}$:

$$\Delta\sigma(\mathcal{C}_3,\mathcal{C}) = \frac{(|\mathcal{C}_1|+|\mathcal{C}|)\Delta\sigma(\mathcal{C}_1,\mathcal{C}) + (|\mathcal{C}_2|+|\mathcal{C}|)\Delta\sigma(\mathcal{C}_2,\mathcal{C}) - |\mathcal{C}|\Delta\sigma(\mathcal{C}_1,\mathcal{C}_2)}{|\mathcal{C}_1|+|\mathcal{C}_2|+|\mathcal{C}|}$$

Preuve : Nous allons remplacer les quatre $\Delta\sigma$ de l'égalité à montrer par leur expression donnée par le Théorème 4. Ceci donne l'expression équivalente :

$$\frac{1}{n}\frac{|\mathcal{C}_3||\mathcal{C}|}{|\mathcal{C}_3|+|\mathcal{C}|} r^2_{\mathcal{C}_3\mathcal{C}} = \frac{1}{n}\frac{|\mathcal{C}_1||\mathcal{C}| r^2_{\mathcal{C}_1\mathcal{C}} + |\mathcal{C}_2||\mathcal{C}| r^2_{\mathcal{C}_2\mathcal{C}} - \frac{|\mathcal{C}||\mathcal{C}_1||\mathcal{C}_2|}{|\mathcal{C}_1|+|\mathcal{C}_2|} r^2_{\mathcal{C}_1\mathcal{C}_2}}{|\mathcal{C}_1|+|\mathcal{C}_2|+|\mathcal{C}|}$$

Multiplions chaque côté de l'égalité par $n\frac{|\mathcal{C}_1|+|\mathcal{C}_2|+|\mathcal{C}|}{|\mathcal{C}|}$ et utilisons le fait que $|\mathcal{C}_1|+|\mathcal{C}_2| = |\mathcal{C}_3|$ pour simplifier cette expression :

$$(|\mathcal{C}_1|+|\mathcal{C}_2|) r^2_{\mathcal{C}_3\mathcal{C}} = |\mathcal{C}_1| r^2_{\mathcal{C}_1\mathcal{C}} + |\mathcal{C}_2| r^2_{\mathcal{C}_2\mathcal{C}} - \frac{|\mathcal{C}_1||\mathcal{C}_2|}{|\mathcal{C}_1|+|\mathcal{C}_2|} r^2_{\mathcal{C}_1\mathcal{C}_2}$$

Utilisons le fait que $P_{\mathcal{C}_3\cdot}$ est le barycentre de $P_{\mathcal{C}_1\cdot}$ pondéré par $|\mathcal{C}_1|$ et de $P_{\mathcal{C}_2\cdot}$ pondéré par $|\mathcal{C}_2|$. Nous obtenons :

$$|\mathcal{C}_1| r^2_{\mathcal{C}_1\mathcal{C}} + |\mathcal{C}_2| r^2_{\mathcal{C}_2\mathcal{C}} = (|\mathcal{C}_1|+|\mathcal{C}_2|) r^2_{\mathcal{C}_3\mathcal{C}} + |\mathcal{C}_1| r^2_{\mathcal{C}_1\mathcal{C}_3} + |\mathcal{C}_2| r^2_{\mathcal{C}_2\mathcal{C}_3}$$

Et nous concluons en utilisant :

$$|\mathcal{C}_1| r^2_{\mathcal{C}_1\mathcal{C}_3} + |\mathcal{C}_2| r^2_{\mathcal{C}_2\mathcal{C}_3} = \frac{|\mathcal{C}_1||\mathcal{C}_2|}{|\mathcal{C}_1|+|\mathcal{C}_2|} r^2_{\mathcal{C}_1\mathcal{C}_2}$$

□

Ce second théorème permet de mettre à jour efficacement les quantités $\Delta\sigma(\mathcal{C}_3, \mathcal{C})$ lorsque nous connaissons déjà les deux valeurs $\Delta\sigma(\mathcal{C}_2, \mathcal{C})$ et $\Delta\sigma(\mathcal{C}_1, \mathcal{C})$. La valeur de $\Delta\sigma(\mathcal{C}_1, \mathcal{C}_2)$ est forcément connue car nous fusionnons $\mathcal{C}_1$ et $\mathcal{C}_2$ correspondrant à la plus faible valeur de $\Delta\sigma$. Nous pouvons ainsi, dans ce cas particulier, mettre à jour la nouvelle valeur de $\Delta\sigma$ en temps constant $\mathcal{O}(1)$ au lieu du temps $\mathcal{O}(n)$ demandé par la formule du Théorème 4.

Après chaque fusion le calcul des nouvelles valeurs $\Delta\sigma$ est prioritairement effectué en $\mathcal{O}(1)$ avec la formule du Théorème 5 lorsque celle-ci peut s'appliquer. Dans le cas contraire, elle sont calculées en $\mathcal{O}(n)$ avec la formule du Théorème 4.

Étant donné que nous ne fusionnons que des communautés adjacentes, nous n'avons besoin de calculer et conserver que les distances correspondantes. Ceci implique que le nombre maximal de distances stockées est au plus m (le nombre d'arêtes du graphe). Nous pouvons stocker ces valeurs dans un arbre équilibré ou dans une structure de tas indexée. Ces deux structures permettent de retrouver efficacement la valeur minimale de $\Delta\sigma$ (en $\mathcal{O}(\log(m))$ pour l'arbre et en $\mathcal{O}(1)$ pour le tas). Elles permettent de plus l'ajout et la suppression d'éléments en $\mathcal{O}(\log(m))$.

En partant d'une partition initiale en communautés $\mathcal{P}^0 = \{\{v\}, v \in V\}$, nous fusionnons successivement des paires de communautés $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{P}^k$ pour créer une série de partitions définies par : $\mathcal{P}^{k+1} = (\mathcal{P}^k \backslash \{\mathcal{C}_1, \mathcal{C}_2\}) \cup \{\mathcal{C}_3\}$. Le choix des communautés $\mathcal{C}_1$ et $\mathcal{C}_2$ à fusionner suit la méthode de Ward de clustering hiérarchique qui minimise à chaque étape la quantité suivante :

$$\sigma_k = \frac{1}{n} \sum_{C \in \mathcal{P}_k} \sum_{i \in C} r_{iC}^2$$

Ceci correspond à fusionner les communautés minimisant la variation $\Delta\sigma_k = \sigma_{k+1} - \sigma_k$. Cette variation ne dépend que des communautés choisies pour la fusion $\mathcal{C}_3 = \mathcal{C}_1 \cup \mathcal{C}_2$:

$$\Delta\sigma(C_1, C_2) = \frac{1}{n}\Big(\sum_{i \in C_3} r_{iC_3}^2 - \sum_{i \in C_1} r_{iC_1}^2 - \sum_{i \in C_2} r_{iC_2}^2 \Big) = \frac{1}{n} \frac{|C_1||C_2|}{|C_1| + |C_2|} r_{C_1C_2}^2$$

Nous obtenons en sortie une structure hiérarchique de communautés (appelée dendrogramme) qui sera utilisée dans le Chapitre 4 pour déterminer la meilleur partition maximisant la fonction de qualité choisie.

3.2.2 Complexité

Nous allons évaluer la complexité totale de notre algorithme. Tout d'abord considérons l'étape de pré-calcul des vecteurs $P_{i\bullet}^t$ pour tous les sommets $i \in V$. Ceci correspond à n calculs de vecteurs de probabilité $P_{i\bullet}^t$. Chaque vecteur est calculé en $\mathcal{O}(mt)$. La complexité totale en temps de cette étape d'initialisation est donc en $\mathcal{O}(mnt)$. La complexité en espace est en $\mathcal{O}(n^2)$ pour stocker les vecteurs $P_{i\bullet}^t$. Nous verrons en Section 3.2.3 une approche dynamique pour réduire cet espace mémoire qui peut dans certains cas devenir trop important.

Nous avons besoin à chaque étape k de l'algorithme de conserver en mémoire les vecteurs $P^t_{\mathcal{C}\bullet}$ pour les communautés courantes $\mathcal{C} \in \mathcal{P}^k$. Pour les communautés précédemment fusionnées, nous conservons uniquement l'information indiquant avec quelle autre communauté elle a été fusionnée. Nous pouvons ainsi à la fin de l'algorithme obtenir la structure hiérarchique de communautés codée par le dendrogramme. Pour maintenir les vecteurs $P^t_{\mathcal{C}\bullet}$ des communautés courantes en mémoire, il suffit à chaque étape de calculer le nouveau vecteur $P^t_{(\mathcal{C}_1\cup\mathcal{C}_2)\bullet} = \frac{|\mathcal{C}_1|P^t_{\mathcal{C}_1\bullet}+|\mathcal{C}_2|P^t_{\mathcal{C}_2\bullet}}{|\mathcal{C}_1|+|\mathcal{C}_2|}$ et de supprimer les deux vecteurs $P^t_{\mathcal{C}_1\bullet}$ et $P^t_{\mathcal{C}_2\bullet}$. Ceci demande à chaque étape un temps en $\mathcal{O}(n)$ et permet de limiter à n le nombre de vecteurs en mémoire à tout instant.

Le facteur prépondérant dans la complexité totale de l'algorithme est le nombre de distances r calculées. Chaque calcul demande un temps $\mathcal{O}(n)$ à partir des vecteurs de probabilité stockés en mémoire. Ces calculs de distances sont effectués lors du calcul initial des distances entre sommets et lors des mises à jour des $\Delta\sigma$ qui utilisent le Théorème 4. Le Théorème 6 établit une borne supérieure au nombre de distances calculées en fonction de la hauteur H du dendrogramme. La hauteur $h(\mathcal{C})$ de chaque noeud $\mathcal{C}$ du dendrogramme est définie récursivement par $h(\mathcal{C}_1 \cup \mathcal{C}_2) = 1 + \max(h(\mathcal{C}_1), h(\mathcal{C}_2))$. La hauteur des feuilles de l'arbre correspondant aux sommets est fixée à 0 ($\forall v \in V, h(\{v\}) = 0$) et la hauteur globale H du dengrogramme correspond à la hauteur de sa racine : $H = h(V)$.

Théorème 6 *Une borne supérieure du nombre de distances r calculées par notre algorithme est $2mH$. Ceci conduit à une complexité totale de notre approche en $\mathcal{O}(mn(H+t))$.*

Preuve : Soit M le nombre de distances r calculées par notre algorithme. M est égal au nombre de distances initiales calculées entre sommets plus, pour chaque étape k, le nombre de communautés voisines de la communauté créée. Le nombre de distances initiales correspond à m, le nombre d'arêtes du graphe. Et à chaque étape, le nombre de communautés adjacentes est une borne supérieure du nombre de distances calculées en supposant que nous utilisons à chaque fois le Théorème 4 et jamais le Théorème 5.

Pour chaque hauteur $1 \leq h < H$, les communautés de même hauteur sont deux à deux disjointes. Ainsi la somme de leur nombre de voisins est majorée par $2m$. En effet, chaque arête peut intervenir au plus deux fois dans une relation d'adjacence. La hauteur $h = 0$ est assimilée à l'étape de pré-calcul. La somme sur toutes les hauteurs permet alors de conclure : $M \leq 2mH$.

Chacune des M distances nécessite un temps de calcul $\mathcal{O}(n)$, le pré-calcul des vecteurs nécessite $\mathcal{O}(mnt)$. Nous obtenons donc une borne supérieure de complexité pour ces deux étapes en $\mathcal{O}(mn(H+t))$. Les autres étapes de l'algorithme demandent des temps de calcul négligeables par rapport à cette complexité. En effet le maintien des vecteurs de probabilité nécessite $\mathcal{O}(n^2)$ et le calcul des $\Delta\sigma$ minimaux grâce à une structure de tas nécessite $\mathcal{O}(M\log(m)) = \mathcal{O}(Hm\log(m))$. La complexité en temps totale de l'algorithme est donc $\mathcal{O}(mn(H+t))$. □

En pratique la longueur t des marches doit être courte (Section 3.1.4). Vu la vitesse exponentielle de convergence du processus de marche aléatoire, nous pouvons considérer que $t = \mathcal{O}(\log(n))$ pour éviter de tomber dans le régime stationnaire limite des marches aléatoires. La complexité globale devient dans ce cas $\mathcal{O}(mnH)$.

Le pire des cas $H = n-1$ correspondant à un dendrogramme déséquilibré en "peigne" peut arriver lorsqu'à chaque étape un unique sommet est fusionné avec la plus grosse communauté. Ce cas extrême peut arriver lorsque nous considérons un graphe en étoile composé d'un sommet central relié à $n-1$ sommets identiques de degré 1. Cependant la méthode de Ward est connue pour favoriser les clusters de tailles petites et équilibrées. En effet la formule donnée par le Théorème 4 montre que pour des distances $r^2_{C_1C_2}$ égales, la pondération en $\frac{|\mathcal{C}_1||\mathcal{C}_2|}{|\mathcal{C}_1|+|\mathcal{C}_2|}$ va donner l'avantage à la fusion des plus petites communautés et favoriser l'équilibrage du dendrogramme.

Le cas le plus favorable correspond à une structure hiérarchique équilibrée. Dans ce cas, la hauteur du dendrogramme est $H = \mathcal{O}(\log n)$ qui conduit à une complexité totale en $\mathcal{O}(mn\log(n))$. Dans les cas rencontrés en pratique, il est souvent raisonnable de considérer que les structures de communautés présentes dans ces graphes sont équilibrées. Ceci peut par exemple découler d'une structure de communautés hiérarchique à plusieurs échelles. Cette constatation a aussi été faite par [58, 16] qui va même jusqu'à considérer que $H = \log(n)$ dans tous les cas pratiques. Nous ne prendrons pas cette hypothèse qui semble trop optimiste, mais nous constatons qu'en pratique nous nous approchons souvent de ce cas favorable et que les cas totalement déséquilibrés n'apparaissent jamais.

Méthode		Distances calculées
Bornes supérieures	$2m(n-1)$	282 000 000
	$2mH$	2 970 000
Tests pratiques	Sans le Théorème 5	321 000
	Avec le Théorème 5	277 000
	Avec les optimisations proposées en 3.2.3	103 000

TAB. 3.1 – Nombre de distances calculées par notre algorithme sur des graphes de $n = 3\,000$ sommets et $m = 47\,000$ arêtes. Les bornes supérieures théoriques correspondent au pire des cas en $2m(n-1)$ et à la borne du Théorème 6 en $2mH$ (pour un hauteur moyenne mesurée $H = 31,6$). Elles sont comparées au nombre réel de distances calculées sur les tests expérimentaux.

Notons que la complexité prouvée au Théorème 6 est une borne supérieure qui est, sauf cas pathologiques, très loin d'être atteinte. En effet, pour chaque hauteur h de communautés dans le dendrogramme, le nombre de distances calculées est majoré par deux fois le nombre total d'arêtes. Le nombre réel de distances est bien inférieur car les nombreuses arêtes internes des communautés et les nombreuses arêtes multiples entre communautés ne génèrent pas de calcul de distances. De plus nous avons considéré que nous n'utilisons jamais le Théorème 5 qui permet dans certain cas de mettre à jour les $\Delta\sigma$ sans avoir recours à un calcul de distance r. Nous avons effectué une étude expérimentale pour évaluer le nombre réel de distances calculées par notre approche. Nous avons considéré des graphes de $n = 3\,000$ sommets comportant un nombre moyen de $m = 47\,000$ arêtes, la hauteur moyenne des dendrogrammes générés est $H = 31,6$. Les résultats (Tableau 3.1) montrent qu'en pratique la borne supérieure de complexité est bien au dessus de la complexité réelle.

Le facteur prépondérant de complexité de notre algorithme est le nombre de distances r calculées. Nous avons montré une borne supérieure conduisant à une complexité en temps en $\mathcal{O}(mn(H+t))$ (où H est la hauteur du dendrogramme produit). Cette complexité donne un pire des cas en $\mathcal{O}(mn^2)$, mais en pratique la méthode de Ward favorise le cas le plus favorable en $\mathcal{O}(mn\log(n))$. La complexité en espace est en $\mathcal{O}(n^2)$.

3.2.3 Améliorations et optimisations

Nous allons proposer deux optimisations qui ne changent pas les complexités asymptotiques mais améliorent grandement les performances de notre approche.

Minoration heuristique des distances

Lorsque nous fusionnons deux communautés $\mathcal{C}_1 \cup \mathcal{C}_2 = \mathcal{C}_3$, nous pouvons mettre à jour les distances $r_{\mathcal{C}_3\mathcal{C}}$ en utilisant le Théorème 4 en $\mathcal{O}(n)$ ou le Théorème 5 en $\mathcal{O}(1)$. La seconde méthode plus efficace nécessite la connaissance préalable des distances $r_{\mathcal{C}_1\mathcal{C}}$ et $r_{\mathcal{C}_2\mathcal{C}}$. Nous allons tout de même utiliser ce théorème lorsque nous ne connaissons pas ces deux distances (notons que nous devons forcément en connaître au moins une).

Lors de la fusion des communautés $\mathcal{C}_1$ et $\mathcal{C}_2$, la distance $r_{\mathcal{C}_1\mathcal{C}_2}$ correspond à la distance minimale entre les communautés (ceci correspond au critère de choix de communautés à fusionner). Nous faisons l'hypothèse que la distance inconnue $r_{\mathcal{C}_1\mathcal{C}}$ ou $r_{\mathcal{C}_2\mathcal{C}}$ est elle aussi supérieure à cette distance minimale connue $r_{\mathcal{C}_1\mathcal{C}_2}$. Ceci permet de calculer une borne inférieure de $r_{\mathcal{C}_3\mathcal{C}}$ en utilisant la formule du Théorème 5.

Nous assimilons cette borne inférieure à la vraie valeur pour la suite de l'algorithme. Si cette distance devient la distance minimale que l'on doit choisir pour fusionner deux communautés, nous effectuons alors le calcul exact de la distance pour vérifier que nous devons bien fusionner ces communautés. Si par contre une des deux communautés venait à se faire fusionner avant que la distance minimale n'atteigne la borne inférieure calculée, alors le calcul de distance correspondant est définitivement économisé. Ceci permet de réduire le nombre de distances à calculer, comme le montre le Tableau 3.1.

Cependant, cette méthode peut entraîner des erreurs lorsque la distance inconnue $r_{\mathcal{C}_1\mathcal{C}}$ (ou $r_{\mathcal{C}_2\mathcal{C}}$) est en réalité inférieure à la distance minimale connue. La fusion des communautés $\mathcal{C}_3$ et $\mathcal{C}$ est alors envisagée plus tard que prévu car la borne inférieure de $r_{\mathcal{C}_3\mathcal{C}}$ peut ne pas être correcte. Il est cependant rare de trouver des distances aussi faibles entre deux communautés qui ne sont pas reliées. Nous avons mesuré sur l'exemple pratique utilisé pour le Tableau 3.1 un taux d'erreur de $0,05\%$. En pratique, cette optimisation heuristique engendre des gains de performance d'un facteur 2 à 3, tout en conservant des résultats équivalents.

Gestion dynamique de la mémoire

La complexité en espace de notre approche est en $\mathcal{O}(n^2)$ car il est nécessaire de stocker n vecteurs de probabilité de taille n. Cette espace mémoire requis peut s'avérer un facteur limitant lorsque nous considérons de très grands graphes. Il est cependant possible de ne pas calculer ces vecteurs et de calculer chaque distance directement en un temps $\mathcal{O}(mt)$. Dans

ces conditions, la complexité en espace devient linéaire mais la complexité en temps passe à $\mathcal{O}(m^2Ht)$.

Nous proposons une solution intermédiaire gérant dynamiquement un espace mémoire réservé pour le stockage des vecteurs de probabilité. Nous supposons que l'utilisateur indique un espace mémoire maximal utilisable pour le stockage de vecteurs. Nous n'effectuons alors les calculs des vecteurs $P^t_{\mathcal{C}\bullet}$ que lorsque nous avons besoin de calculer une distance les utilisant. A chaque instant nous conservons en mémoire (dans la limite de la capacité allouée) les vecteurs de probabilité qui ont le plus de chance d'être utilisés dans les prochains calculs de distances. Ceci permet lorsqu'un vecteur est stocké en mémoire de faire un calcul de distance en $\mathcal{O}(n)$. Sinon, nous calculons les vecteurs nécessaires en $\mathcal{O}(mt)$ puis la distance en $\mathcal{O}(n)$, nous pouvons alors stocker ce vecteur en remplacement d'autres vecteurs en mémoire selon le critère suivant : Nous stockons en mémoire les vecteurs pouvant être utilisés dans les plus proches calculs de distances. Un vecteur est utilisé lorsque une distance est mise à jour avec une communauté voisine qui vient de fusionner. Pour chaque communauté, nous connaissons la distance minimale $r^k_{min}(\mathcal{C})$ qui la relie à une autre communauté de $\mathcal{P}^k$, cette distance minimale indique l'ordre dans lequel les communautés vont être fusionnées. Les vecteurs ayant le plus de chance d'être utilisés rapidement correspondent aux communautés voisines des communautés possédant les plus faibles distances minimales. Nous conservons donc cette valeur que nous tenons à jour après chaque nouveau calcul de distance. Ainsi, nous conservons prioritairement les vecteurs des communautés liées aux communautés possédant les plus petites distances minimales $r^k_{min}(\mathcal{C})$.

Nous proposons deux améliorations de notre approche :
- Un calcul heuristique d'une borne inférieure des distances permettant de limiter grandement le nombre de distances à calculer.
- Un procédé de gestion dynamique de mémoire pour faire fonctionner l'algorithme avec un espace maximal fixé de mémoire. Ceci permet de traiter les très grands graphes pour lesquels la quantité $\mathcal{O}(n^2)$ de mémoire n'est pas disponible.

Chapitre 4

Partitions en communautés et coupes multi-échelles

Étant donné un dendrogramme, il existe une multitude de partitions en communautés possibles. Le choix parmi ces partitions se fait généralement par une *fonction de qualité* capturant les propriétés souhaitées. La partition retenue est celle qui maximise cette fonction. Nous allons voir dans ce chapitre qu'il est possible de définir un classe très générale de fonctions de qualité, englobant les plus usuelles, pour lesquelles des méthodes d'optimisation efficaces existent. Nous verrons ensuite comment on peut utiliser ces concepts pour détecter des communautés à plusieurs échelles, et même identifier les échelles d'observation les plus pertinentes.

4.1 Problématique

Considérons une structure hiérarchique de communautés (dendrogramme) produite en sortie de tout algorithme agglomératif ou divisif de détection de communautés. Elle peut donc être issue de l'algorithme décrit au Chapitre 3 mais aussi de n'importe quel autre algorithme produisant un dendrogramme [33, 68, 29, 16, 20, 88].

Nous considérons que le dendrogramme est codé par une série $\mathcal{P}^k$ de partitions correspondant aux étapes successives de l'algorithme. Nous fixons la première partition $\mathcal{P}^0 = \{\{v\}, v \in V\}$ comme la partition la plus fine composée de communautés de taille 1. Ensuite chaque étape k regroupe plusieurs communautés conduisant à la partition $\mathcal{P}^{k+1} = (\mathcal{P}^k \backslash \{\mathcal{C}'_1, \ldots, \mathcal{C}'_j\}) \cup \{\mathcal{C}_k\}$ où les sous-communautés $\mathcal{C}'_i$ sont disjointes et vérifient $\mathcal{C}_k = \cup_{i=1}^{j} \mathcal{C}'_i$. Chacune de ces étapes correspond à un noeud du dendrogramme. Nous considérons ainsi une structure arborescente générale, dans laquelle une communauté peut avoir plus de deux sous-communautés. Le cas particulier et fréquent d'une structure hiérarchique binaire dans laquelle chaque communauté possède exactement deux sous-communautés (dont notre approche fait partie) rentre dans ce cadre plus général. Nous avons choisi des notations cohérentes avec une approche agglomérative (dont notre approche fait partie) qui fusionne successivement des communautés, les dendrogrammes produits par des approches divisives peuvent bien sûr être pris en compte par le même formalisme.

Les approches classiques cherchent à maximiser une fonction de qualité $Q(\mathcal{P})$ sur l'ensemble des partitions successives $\mathcal{P}^k$ produites par l'algorithme. Cette approche ne considère qu'un nombre restreint de partitions correspondant au nombre d'étapes de l'algorithme utilisé. Ce nombre d'étapes est au plus $n-1$ (borne atteinte pour une structure hiérarchique

binaire). L'ensemble $\mathcal{S} = \cup_k \mathcal{P}^k$ de toutes les communautés présentes dans le dendrogramme permet cependant d'envisager d'autres partitions. Le nombre de communautés présentes dans le dendrogramme est aux plus $2n-1$ ($|\mathcal{S}| < 2n$) : n communautés d'un seul sommet plus une communauté par étape de l'algorithme. Nous pouvons alors construire des partitions de sommets en choisissant des communautés de $\mathcal{S}$ qui n'apparaissent pas forcément dans une même partition $\mathcal{P}^k$. Nous créons ainsi des nouvelles partitions qui ne correspondent pas uniquement aux étapes de l'algorithme.

Nous définissons l'ensemble Π des partitions de V que nous pouvons créer à partir des communautés de $\mathcal{S}$. Ces partitions sont composées de communautés disjointes de $\mathcal{S}$ dont l'union recouvre l'ensemble des sommets V.

$$\Pi = \left\{ \mathcal{P} \text{ tel que } \forall \mathcal{C} \in \mathcal{P}, \mathcal{C} \in \mathcal{S} \text{ et } \forall \mathcal{C}_1, \mathcal{C}_2 \in \mathcal{P}, \mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset \text{ et } \cup_{\mathcal{C} \in \mathcal{P}} \mathcal{C} = V \right\}$$

Intuitivement, ces partitions sont données par des coupes horizontales, mais pas nécessairement droites, du dendrogramme (Figure 4.1b page 73). Le nombre de ces partitions est dans le cas général exponentiel. Nous allons dans ce chapitre proposer une méthode pour maximiser une classe de fonctions de qualité, dites additives (Section 4.2.2), sur l'ensemble Π. Cette méthode donne nécessairement des résultats au moins aussi bons que la méthode classique car elle considère un ensemble de partitions plus important.

Nous aurons aussi l'occasion de considérer des partitions des sommets d'une communauté $\mathcal{C} \in \mathcal{S}$ donnée. Nous définissons dans cette optique la restriction $\Pi^{\mathcal{C}}$ de Π regroupant les partitions possibles d'une communauté $\mathcal{C} \in \mathcal{S}$:

$$\Pi^{\mathcal{C}} = \left\{ \mathcal{P}^{\mathcal{C}} \text{ tel que } \forall \mathcal{C}' \in \mathcal{P}^{\mathcal{C}}, \mathcal{C}' \in \mathcal{S} \text{ et } \forall \mathcal{C}'_1, \mathcal{C}'_2 \in \mathcal{P}^{\mathcal{C}}, \mathcal{C}'_1 \cap \mathcal{C}'_2 = \emptyset \text{ et } \cup_{\mathcal{C}' \in \mathcal{P}^{\mathcal{C}}} \mathcal{C}' = \mathcal{C} \right\}$$

Ces ensembles de partitions $\Pi^{\mathcal{C}}$ peuvent être vus comme des restrictions de Π à une communauté $\mathcal{C} \in \mathcal{S}$. En effet, toute partition $\mathcal{P} \in \Pi$ des sommets V induit une partition $\mathcal{P}^{\mathcal{C}} \in \Pi^{\mathcal{C}}$ des sommets de la communauté $\mathcal{C}$.

> Nous cherchons à déterminer une partition en communautés à partir d'un dendrogramme produit par une méthode quelconque de détection de communautés. Nous considérons l'ensemble $\mathcal{S}$ des communautés présentes dans le dendrogramme et nous définissons l'ensemble Π des partitions de V composées de communautés de $\mathcal{S}$ (cet ensemble Π contient en général un nombre exponentiel de partitions). Nous définissons de même les ensembles restreints $\Pi^{\mathcal{C}}$ des partitions des sommets d'une communauté $\mathcal{C} \in \mathcal{S}$.

4.2 Fonctions de qualité

Nous allons dans cette section donner plusieurs fonctions qui permettent de mesurer la qualité d'une partition en communautés. Ces fonctions de qualité font intervenir des critères de densité ou d'homogénéité à l'intérieur et entre les communautés. L'objectif est de trouver une partition $\mathcal{P}$ qui optimise ces fonctions. Sans perte de généralité nous considérerons des fonctions de qualité que l'on souhaite maximiser. Nous proposerons ensuite une classe de fonctions additives qui regroupe les fonctions de qualité les plus courantes et qui permettra de proposer une méthode de maximisation efficace sur l'ensemble des partitions Π.

4.2.1 Exemples de fonctions de qualité

Nous présentons ici trois fonctions de qualité. La première est largement utilisée dans le contexte de détection de communautés, et la dernière est une nouvelle fonction que nous proposons à partir de la quantité σ introduite dans le chapitre précédent. D'autres fonctions de qualité existent, nous renvoyons par exemple le lecteur à l'ouvrage de synthèse [12] qui en référence plusieurs.

La modularité Q^M

Cette fonction, introduite par [59], est aujourd'hui la plus utilisée [16, 19, 22, 29, 38, 59]. Elle repose sur la proportion d'arêtes internes aux communautés $e(\mathcal{C})$ et la proportion d'arêtes liées aux communautés $a(\mathcal{C})$.

Une arête est dite interne à une communauté $\mathcal{C}$ si ses deux extrémités sont dans la communauté $\mathcal{C}$. Le nombre d'arêtes internes est donc égal à $\frac{1}{2}\sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{C}} A_{ij}$ ou A est la matrice d'adjacence du graphe. La proportion d'arêtes internes $e(\mathcal{C})$ est prise par rapport au nombre total d'arêtes m : $e(\mathcal{C}) = \frac{1}{2m}\sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{C}} A_{ij}$. Cette définition a été introduite pour des graphes non-pondérés, mais nous pouvons l'adapter aux graphes pondérés en remplaçant le nombre d'arêtes internes par le poids total des arêtes internes $w_{int}(\mathcal{C}) = \frac{1}{2}\sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{C}} w_{ij}$ et en remplaçant le nombre total d'arêtes par le poids total du graphe $w(G) = \frac{1}{2}\sum_{i\in V}\sum_{j\in V} w_{ij}$. Nous obtenons l'expression suivante pour la proportion d'arêtes internes d'une communauté $\mathcal{C}$ d'un graphe pondéré :

$$e(\mathcal{C}) = \frac{w_{int}(\mathcal{C})}{w(G)} = \frac{\sum_{i\in\mathcal{C}}\sum_{j\in\mathcal{C}} w_{ij}}{\sum_{i\in V}\sum_{j\in V} w_{ij}}$$

De même une arête est dite liée à une communauté $\mathcal{C}$ si l'une de ses deux extrémités (éventuellement les deux) appartient à la communauté $\mathcal{C}$. Les arêtes ayant une seule extrémité dans $\mathcal{C}$ comptent pour moitié par rapport aux arêtes ayant leurs deux extrémités dans $\mathcal{C}$. Le nombre d'arêtes liées à une communauté $\mathcal{C}$ est donc $\frac{1}{2}\sum_{i\in\mathcal{C}}\sum_{j\in V} A_{ij}$. Nous pouvons remplacer pour un graphe pondéré ce nombre d'arêtes par un poids des arêtes liées à $\mathcal{C}$: $w(\mathcal{C}) = \frac{1}{2}\sum_{i\in\mathcal{C}}\sum_{j\in V} w_{ij}$. Nous obtenons alors l'expression suivante pour la proportion $a(\mathcal{C})$ des arêtes liées à une communauté $\mathcal{C}$:

$$a(\mathcal{C}) = \frac{w(\mathcal{C})}{w(G)} = \frac{\sum_{i\in\mathcal{C}}\sum_{j\in V} w_{ij}}{\sum_{i\in V}\sum_{j\in V} w_{ij}}$$

Nous voulons des communautés de forte densité interne, mesurée par $e(\mathcal{C})$. Les grosses communautés ont cependant mécaniquement une proportion d'arêtes internes plus élevée : si $\mathcal{C}$ est un ensemble aléatoire de sommets et si les liens sont eux mêmes aléatoires, alors la proportion de liens internes à $\mathcal{C}$ attendue est $a(\mathcal{C})^2$. En effet, chacune des deux extrémités d'un lien pris au hasard a, dans cette hypothèse, une probabilité de $a(\mathcal{C})$ d'être dans la communauté $\mathcal{C}$.

La modularité compare la proportion effective d'arêtes internes aux communautés à la proportion attendue selon ce schéma. Une communauté est d'autant plus pertinente que sa proportion d'arêtes internes sera supérieure à sa proportion attendue d'arêtes. Ceci est capturé dans la définition suivante de la modularité que l'on cherchera à maximiser :

$$Q^M(\mathcal{P}) = \sum_{\mathcal{C}\in\mathcal{P}} e(\mathcal{C}) - a(\mathcal{C})^2$$

Notons que la modularité est comprise entre -1 et 1. En effet $0 \leq \sum_{\mathcal{C}\in\mathcal{P}} e(\mathcal{C}) \leq 1$ et $\sum_{\mathcal{C}\in\mathcal{P}} a(\mathcal{C}) = 1$ donc $0 \leq \sum_{\mathcal{C}\in\mathcal{P}} a(\mathcal{C})^2 \leq 1$. Il est possible de rencontrer des partitions possédant une modularité négative. La modularité de la partition contenant une unique communauté regroupant tous les sommets est nulle ($Q^M(\{V\}) = 0$).

Remarquons enfin que le calcul de la modularité d'une partition peut être effectué en temps $\mathcal{O}(m)$.

La performance Q^P

Une définition de cette fonction de qualité est donnée dans [12]. Elle compte le nombre de paires de sommets qui sont en accord avec la partition en communautés. Deux sommets reliés par une arête sont en accord avec la partition s'ils appartiennent à la même communauté. Deux sommets qui ne sont pas liés par une arête sont en accord avec la partition s'ils appartiennent à deux communautés différentes. La performance $Q^P(\mathcal{P})$ est la proportion de paires de sommets en accord avec la partition :

$$Q^P(\mathcal{P}) = \frac{|\{\{u,v\} \in E, \mathcal{C}(u) = \mathcal{C}(v)\}| + |\{\{u,v\} \notin E, \mathcal{C}(u) \neq \mathcal{C}(v)\}|}{\frac{1}{2}n(n-1)}$$

où $\mathcal{C}(u)$ représente la communauté de la partition $\mathcal{P}$ qui contient le sommet u. Nous avons donc $0 \leq Q^P(\mathcal{P}) \leq 1$. Il n'est pas possible de généraliser directement et simplement cette fonction de qualité à des graphes pondérés. Le nombre d'arêtes internes aux communautés peut être transformé par leur poids total, par contre il n'est pas immédiat d'attribuer un poids aux paires de sommets non liés. Plusieurs généralisations possibles sont proposées dans [12].

À nouveau, le calcul de la performance d'une partition peut être effectué en temps $\mathcal{O}(m)$.

Une fonction de qualité basée sur la similarité des sommets Q^S

Nous proposons maintenant une nouvelle fonction de qualité, basée sur la similarité des sommets définie par la quantité σ introduite dans le chapitre précédent. Rappelons que cette quantité se base sur une mesure r de similarité entre sommets et entre communautés pour mesurer l'homogénéité des communautés :

$$\sigma(\mathcal{P}) = \frac{1}{n} \sum_{\mathcal{C}\in\mathcal{P}} \sum_{i\in\mathcal{C}} r_{i\mathcal{C}}^2$$

De plus, la distance r que nous proposons est une distance Euclidienne, ce qui permet de donner une définition équivalente de σ :

$$\sigma(\mathcal{P}) = \frac{1}{n} \sum_{\mathcal{C}\in\mathcal{P}} \frac{1}{|\mathcal{C}|} \sum_{i,j\in\mathcal{C}} r_{ij}^2$$

Nous suggérons l'utilisation de la distance r basée sur les marches aléatoires, cette distance étant Euclidienne et permetant un calcul efficace des quantités $\sigma(\mathcal{P})$. Cependant nous pouvons envisager n'importe quelle autre distance qui mesure la similarité des sommets, le cas des distances Euclidiennes étant plus favorable car il permet de n'effectuer que n calculs de distance pour chaque évaluation de σ alors que le cas général peut en nécessiter jusqu'à n^2.

Nous avons en réalité uniquement besoin de définir une mesure de l'homogénéité des communautés $\sigma(\mathcal{C})$. Dans le cas que nous proposons et avec l'utilisation d'une distance r Euclidienne, nous avons :

$$\sigma(\mathcal{C}) = \frac{1}{n}\sum_{i\in\mathcal{C}} r_{i\mathcal{C}}^2 = \frac{1}{n}\frac{1}{|\mathcal{C}|}\sum_{i,j\in\mathcal{C}} r_{ij}^2$$

Nous souhaitons minimiser la quantité $\sigma(\mathcal{P}) = \sum_{\mathcal{C}\in\mathcal{P}} \sigma(\mathcal{C})$ pour obtenir des communautés homogènes. Cependant cette minimisation conduit à la partition possédant n communautés d'un seul sommet. Pour contrebalancer cet effet nous allons minimiser en même temps le nombre $|\mathcal{P}|$ de communautés de la partition $\mathcal{P}$. Nous obtenons ainsi un compromis entre l'homogénéité des communautés et leur taille. En vue d'une normalisation, la valeur maximale de $\sigma(\mathcal{C})$, obtenue pour la plus grosse communauté $\mathcal{C} = V$, est notée σ_{max} et le nombre maximal de communautés est n ($|\mathcal{P}| \leq n$). Nous cherchons donc à maximiser la quantité Q^S suivante :

$$Q^S = -\frac{|\mathcal{P}|}{n} - \sum_{\mathcal{C}\in\mathcal{P}} \frac{\sigma(\mathcal{C})}{\sigma_{max}}$$

Cette fonction de qualité vérifie $-2 \leq Q^S \leq 0$. Il aurait bien évidement été plus intuitif de définir la fonction positive $-Q^S$ que l'on cherche à minimiser. Nous avons cependant choisi de définir une fonction de qualité que l'on doit maximiser pour garder une cohérence avec les définitions précédentes et avec la définition de fonctions de qualité additives en Section 4.2.2.

Le calcul de la fonction de qualité $Q^S(\mathcal{P})$ nécessite, suivant la définition de $\sigma(\mathcal{C})$, l'évaluation de n distances $r_{i\mathcal{C}}$ ou $\mathcal{O}(n^2)$ distances r_{ij}. Ce calcul coûteux des $\sigma(\mathcal{C})$ est cependant déjà effectué dans l'algorithme que nous proposons au Chapitre 3. Nous pouvons donc utiliser cette fonction de qualité avec notre approche de détection de communautés sans aucune surcharge de calcul.

Nous présentons trois fonctions de qualité que l'on cherche à maximiser
- La modularité Q^M [59]

$$Q^M(\mathcal{P}) = \sum_{\mathcal{C}\in\mathcal{P}} e(\mathcal{C}) - a(\mathcal{C})^2 \text{ où } e(\mathcal{C}) = \frac{w_{int}(\mathcal{C})}{w(G)} \text{ et } a(\mathcal{C}) = \frac{w(\mathcal{C})}{w(G)}$$

  $e(\mathcal{C})$ et $a(\mathcal{C})$ représentent respectivement les proportions d'arêtes internes et d'arêtes liées à la communauté $\mathcal{C}$.
- La performance Q^P [12]

$$Q^P(\mathcal{P}) = \frac{|\{\{u,v\} \in E, \mathcal{C}(u) = \mathcal{C}(v)\}| + |\{\{u,v\} \notin E, \mathcal{C}(u) \neq \mathcal{C}(v)\}|}{\frac{1}{2}n(n-1)}$$

  où $\mathcal{C}(u)$ représente la communauté contenant le sommet u. Cette fonction compte la proportion de paires de sommets en accord avec la partition.
- Une fonction basée sur une distance de similarité Q^S

$$Q^S = -\frac{|\mathcal{P}|}{n} - \sum_{\mathcal{C}\in\mathcal{P}} \frac{\sigma(\mathcal{C})}{\sigma_{max}} \text{ où } \sigma(\mathcal{C}) = \frac{1}{n}\frac{1}{|\mathcal{C}|}\sum_{i,j\in\mathcal{C}} r_{ij}^2$$

  Cette nouvelle fonction se base sur une distance de similarité entre sommets r_{ij}. Elle peut être avantageusement utilisée dans notre algorithme proposé au Chapitre 3 qui calcule déjà les valeurs de $\sigma(\mathcal{C})$.

4.2.2 Fonctions de qualité additives

Nous définissons maintenant une classe générale de fonctions de qualité, que nous appellerons *additives*.

Définition 6 *Une fonction de qualité Q est additive s'il existe une fonction élémentaire $q(\mathcal{C})$ (qui ne dépend que de la communauté $\mathcal{C}$) telle que pour toute partition $\mathcal{P}$:*

$$Q(\mathcal{P}) = \sum_{\mathcal{C}\in\mathcal{P}} q(\mathcal{C})$$

Sans perte de généralité, nous pouvons imposer que l'on cherche à maximiser la fonction de qualité. Il est possible de produire des fonctions de qualité que l'on veut minimiser, dans ce cas il suffit de prendre l'opposé de cette fonction (comme nous l'avons fait pour Q^S).

Cette définition de fonction de qualité s'applique pour toute partition $\mathcal{P} \in \Pi$. Nous pouvons cependant appliquer directement cette définition additive pour toute partition d'un sous ensemble de V. Nous noterons donc aussi $Q(\mathcal{P}^{\mathcal{C}}) = \sum_{\mathcal{C}'\in\mathcal{P}^{\mathcal{C}}} q(\mathcal{C}')$ pour toute partition $\mathcal{P}^{\mathcal{C}} \in \Pi^{\mathcal{C}}$. Ceci est possible car les fonctions de qualité additives dépendent des communautés présentes dans la partition $\mathcal{P}$ mais pas des interactions entre ces communautés.

Nous allons maintenant montrer que cette classe de fonctions n'est pas restrictive en prouvant que les trois fonctions présentées en Section 4.2.1 sont des fonctions additives.

Théorème 7 *Les trois fonctions de qualité présentées en Section 4.2.1 (Q^M, Q^P, Q^S) sont additives.*

Preuve : Pour prouver ce théorème, il suffit de trouver les trois fonctions élémentaires (q^M, q^P, q^S) correspondantes :

- Q^M : $q^M(\mathcal{C}) = e(\mathcal{C}) - a(\mathcal{C})^2$.
- Q^P : $q^P(\mathcal{C}) = \frac{1}{n(n-1)} \sum_{u \in \mathcal{C}} |\{v \in \mathcal{C}, \{u,v\} \in E\}| + |\{v \notin \mathcal{C}, \{u,v\} \notin E\}|$.
- Q^S : $q^S(\mathcal{C}) = -\frac{1}{n} - \frac{\sigma(\mathcal{C})}{\sigma_{max}}$

La seule fonction élémentaire non immédiate est q^P. Elle compte les paires de sommets en accord avec la communauté $\mathcal{C}$. Ces paires sont d'une part les paires internes liées par une arête et d'autre part les paires externes non liées. Chaque paire est comptée deux fois (une fois pour chacun des deux sommets), ce qui explique la disparition du facteur $\frac{1}{2}$. □

En pratique, ces fonctions de qualité additives représentent une classe de fonctions peu restrictive. En particulier, nous venons de voir que les principales fonctions de qualité sont des fonctions additives. Cette propriété simple permettra par la suite de proposer des algorithmes efficaces pour trouver la meilleure partition $\mathcal{P} \in \Pi$ optimisant une fonction de qualité de ce type.

Nous définissons la classe des fonctions de qualité additives : elles s'expriment comme sommes de fonctions élémentaires $q(\mathcal{C})$ qui ne dépendent que des communautés.

$$Q(\mathcal{P}) = \sum_{\mathcal{C} \in \mathcal{P}} q(\mathcal{C})$$

Cette classe de fonctions est peu restrictive en pratique : les trois fonctions de qualité présentées en 4.2.1 (Q^M, Q^P, Q^S) sont toutes additives. Nous utiliserons par la suite les propriétés des fonctions de qualité additives pour proposer des approches efficaces d'optimisation.

4.3 Optimisation d'une fonction de qualité additive sur un dendrogramme

Nous allons montrer dans cette section qu'il est possible d'optimiser une fonction de qualité additive avec une approche récursive simple. Le problème est de trouver la partition $\mathcal{P}$ maximisant une fonction de qualité additive Q. Cette maximisation n'est pas faisable efficacement si l'on considère l'ensemble des partitions possibles des sommets. Nous allons donc nous limiter à l'ensemble Π des partitions que l'on peut construire à partir des communautés données par un dendrogramme. Cet ensemble Π contient tout de même, dans le cas général, un nombre exponentiel de partitions. L'approche que nous proposons se base sur le lemme suivant permettant de calculer la partition de qualité maximale sur l'ensemble restreint $\Pi^{\mathcal{C}}$ des partitions de $\mathcal{C}$.

Lemme 2 *Soit une fonction de qualité additive Q et un dendrogramme dans lequel la communauté $\mathcal{C}$ possède les sous-communautés filles $\mathcal{C}'_1, \ldots, \mathcal{C}'_k$.*

La partition $\mathcal{P}^{\mathcal{C}}_{max} \in \Pi^{\mathcal{C}}$ *qui maximise* Q *est soit* $\mathcal{P}^{\mathcal{C}}_{max} = \{\mathcal{C}\}$, *soit* $\mathcal{P}^{\mathcal{C}}_{max} = \mathcal{P}'_1 \cup \ldots \cup \mathcal{P}'_k$ *où les partitions* $\mathcal{P}'_i \in \Pi^{\mathcal{C}'_i}$ *maximisent* Q *sur* $\Pi^{\mathcal{C}'_i}$. *Nous avons alors :*

$$Q(\mathcal{P}^{\mathcal{C}}_{max}) = \max_{\mathcal{P} \in \Pi^{\mathcal{C}}} Q(\mathcal{P}) = \max \left(q(\{\mathcal{C}\}), \sum_{i=1}^{k} Q(\mathcal{P}'_i) \right)$$

Preuve : Supposons que la partition $\mathcal{P}^{\mathcal{C}}_{max}$ qui maximise Q sur $\Pi^{\mathcal{C}}$ n'est pas la partition $\{\mathcal{C}\}$. Cette partition induit alors des sous-partitions $\mathcal{P}'_i \in \Pi^{\mathcal{C}_i}$ sur chacune des communautés filles $\mathcal{C}_i$ de $\mathcal{C}$. Ces partitions $\mathcal{P}'_i$ vérifient $\cup_{i=1}^{k} \mathcal{P}'_i = \mathcal{P}^{\mathcal{C}}_{max}$.

Supposons maintenant qu'il existe une autre partition $\mathcal{P}''_i \in \Pi^{\mathcal{C}_i}$ qui vérifie $Q(\mathcal{P}''_i) > Q(\mathcal{P}'_i)$. Nous pouvons alors construire une nouvelle partition $\mathcal{P}'_{max} = \mathcal{P}'_1 \cup \ldots \cup \mathcal{P}''_i \cup \ldots \cup \mathcal{P}'_k$ de $\Pi^{\mathcal{C}}$. L'additivité de la fonction de qualité Q permet de conclure que dans ce cas $Q(\mathcal{P}'_{max}) > Q(\mathcal{P}_{max})$. Nous obtenons ainsi une nouvelle partition qui contredit notre hypothèse de départ sur le fait que $\mathcal{P}^{\mathcal{C}}_{max}$ maximise Q sur $\Pi^{\mathcal{C}}$. Nous en déduisons donc que chacune des partitions $\mathcal{P}'_i$ maximise Q sur $\Pi^{\mathcal{C}_i}$. Dans ce cas l'additivité de la fonction Q permet de vérifier que $Q(\mathcal{P}^{\mathcal{C}}_{max}) = \sum_{i=1}^{k} Q(\mathcal{P}'_i)$.

Il suffit alors de prendre en compte le cas particulier où la partition qui maximise Q est la partition $\mathcal{P}^{\mathcal{C}}_{max} = \{\mathcal{C}\}$ pour obtenir le lemme. □

Nous allons calculer récursivement la meilleure partition $\mathcal{P}^{\mathcal{C}}_{max} \in \Pi^{\mathcal{C}}$ qui maximise une fonction additive Q sur une communauté donnée. Le Lemme 2 permet de définir précisément comment construire cette partition et permet de calculer la valeur correspondante de $Q(\mathcal{P}^{\mathcal{C}})$. Nous allons définir une fonction qui retourne cette partition et la valeur de la fonction de qualité Q associée à la partition.

Théorème 8 *Étant donnés une fonction de qualité additive* Q *et un dendrogramme, il est possible de trouver la partition* $\mathcal{P} \in \Pi$ *qui maximise* Q *en effectuant* $\mathcal{O}(n)$ *évaluations de fonctions élémentaires* $q(\mathcal{C})$. *Ceci est accompli par la fonction* `MeilleurePartition`.

Function `MeilleurePartition(`$\mathcal{C}$`)`

---

foreach *sous-communauté fille* $\mathcal{C}_i$ *de* $\mathcal{C}$ **do**
  $(Q_i, \mathcal{P}_i) \leftarrow$ `MeilleurePartition(`$\mathcal{C}_i$`)`
end
if $\mathcal{C}$ *n'a pas de sous-communauté ou* $q(\mathcal{C}) > \sum_i Q_i$ **then**
  return $(q(\mathcal{C}), \{\mathcal{C}\})$
else
  return $(\sum_i Q_i, \cup_i \mathcal{P}_i)$

Preuve : La fonction `MeilleurePartition` retourne la partition $\mathcal{P}^{\mathcal{C}}_{max}$ et la valeur maximale de Q pour toute communauté $\mathcal{C} \in \mathcal{S}$. La correction de l'étape de récursion est donnée par le Lemme 2 : les cas d'initialisation correspondant aux communautés feuilles du dendrogramme n'ont qu'un seul choix de partition, les autres cas font le choix entre la partition composée d'une seule communauté et l'union des meilleures partitions de chaque sous-communauté $\mathcal{C}_i$. Pour trouver la meilleure partition $\mathcal{P} \in \Pi$ il suffit d'appeler la fonction avec $\mathcal{C} = V$.

Pour la complexité, la fonction est appelée une seule fois pour chaque noeud correspondant à une communauté $\mathcal{C} \in \mathcal{S}$ du dendrogramme . Une seule évaluation de fonction élémentaire $q(\mathcal{C})$ est effectuée à chaque appel récursif de la fonction. Le nombre total d'évaluations de $q(\mathcal{C})$ est donc $|\mathcal{S}| < 2n$. □

Notons qu'il est possible, pour certaines fonctions de qualité, d'optimiser le calcul des fonctions élémentaires $q(\mathcal{C})$. En effet certaines fonctions de qualité permettent de calculer efficacement $q(\mathcal{C}_1 \cup \mathcal{C}_2)$ à partir des valeurs de $q(\mathcal{C}_1)$ et $q(\mathcal{C}_2)$. Ceci permet d'améliorer les performances de notre approche pour des fonctions de qualité telles que la modularité Q^M (en utilisant l'analyse de [16]) ou la fonction basée sur la similarité Q^S (en utilisant le Théorème 4).

Nous proposons une approche récursive simple pour calculer la meilleure partition $\mathcal{P} \in \Pi$ qui maximise une fonction de qualité additive Q. Cette approche nécessite un nombre linéaire $\mathcal{O}(n)$ d'évaluations de fonctions élémentaires $q(\mathcal{C})$.
`MeilleurePartition(`V`)` donne cette partition.

4.4 Détection multi-échelles de communautés

Comme nous l'avons vu, de nombreux algorithmes de détection de communautés génèrent des structures hiérarchiques de communautés. Pourtant ils se contentent généralement de fournir en sortie une partition unique. Il est possible qu'il y ait en fait d'autres structures de communautés plus petites ou plus grosses qui soient aussi pertinentes que cette partition unique. En effet, nous pouvons rencontrer des communautés à différentes échelles mais l'approche classique n'en retiendra qu'une. L'approche que nous avons proposée dans la Section 4.3 rentre dans cette vision classique et ne permet de trouver qu'une seule partition correspondant à une seule échelle de taille de communauté.

Pour dépasser cette limitation, nous allons proposer des fonctions de qualité multi-échelles qui permettront de détecter des communautés à différentes échelles. Ces fonctions de qualité sont paramétrées par un facteur d'échelle α ($0 \leq \alpha \leq 1$) qui permet de sélectionner l'échelle à laquelle nous souhaitons chercher des communautés. Ces fonctions paramétrées peuvent directement être utilisées avec l'algorithme récursif proposé à la Section 4.3. Nous proposerons cependant une meilleure approche qui permettra de trouver en une seule étape toutes les partitions pour toutes les échelles possibles données par le paramètre α.

4.4.1 Fonctions de qualité multi-échelles

Nous allons proposer une définition générale pour des familles de fonctions de qualité multi-échelles $(Q_\alpha)_{0\leq\alpha\leq1}$. Ces fonctions de qualité sont paramétrées par un paramètre d'échelle α ($0 \leq \alpha \leq 1$) qui représente le niveau auquel on souhaite chercher des communautés.

Le paramètre $\alpha = 0$ correspond au niveau microscopique pour lequel la meilleure partition sera la partition la plus fine $\mathcal{P} = \{\{v\}, v \in V\}$ composée de n communautés de taille un. À l'opposé, le paramètre $\alpha = 1$ correspond au niveau macroscopique pour lequel la meilleure partition sera la partition la moins fine $\mathcal{P} = V$ composée d'une seule communauté regroupant

tous les sommets. Les valeurs intermédiaires de α correspondent à des partitions intermédiaires qui seront ordonnées par leur facteur d'échelle α. La définition suivante formalise cette notion.

Définition 7 *Soit une famille de fonctions de qualité $(Q_\alpha)_{0\leq\alpha\leq 1}$, nous notons $\mathcal{P}_\alpha$ la partition qui maximise Q_α sur Π :*

$$\forall\alpha\in[0,1], Q_\alpha(\mathcal{P}_\alpha)=\max_{\mathcal{P}\in\Pi} Q_\alpha(\mathcal{P})$$

Nous dirons que $(Q_\alpha)_{0\leq\alpha\leq 1}$ est une famille de fonctions de qualité multi-échelles si les partitions $\mathcal{P}_\alpha$ sont ordonnées en accord avec le facteur d'échelle α i.e. vérifient :

$$\mathcal{P}_0=\{\{v\}|v\in V\},\ \mathcal{P}_1=\{V\}\ et\ \alpha_1\leq\alpha_2\Rightarrow\mathcal{P}_{\alpha_1}\preceq\mathcal{P}_{\alpha_2}$$

où $\mathcal{P}_{\alpha_1}\preceq\mathcal{P}_{\alpha_2}$ si et seulement si les communautés de $\mathcal{P}_{\alpha_1}$ sont incluses dans celles de $\mathcal{P}_{\alpha_2}$; on dit alors que $\mathcal{P}_{\alpha_1}$ est plus fine que $\mathcal{P}_{\alpha_2}$:

$$\mathcal{P}_{\alpha_1}\preceq\mathcal{P}_{\alpha_2}\Leftrightarrow\forall\mathcal{C}_1\in\mathcal{P}_{\alpha_1},\exists\mathcal{C}_2\in\mathcal{P}_{\alpha_2}\ tel\ que\ \mathcal{C}_1\subseteq\mathcal{C}_2$$

Pour tout choix d'un paramètre d'échelle α, nous obtenons donc une fonction de qualité Q_α qui définit une partition $\mathcal{P}_\alpha$. La partition $\mathcal{P}_\alpha$ représente la meilleure partition en communautés pour le facteur d'échelle α donné. La notion de fonction de qualité additive peut s'appliquer aux fonctions de qualité multi-échelles. Nous allons maintenant proposer une classe générale de fonctions de qualité additives et multi-échelles.

Définition 8 *Considérons deux fonctions $h(\mathcal{C})$ et $l(\mathcal{C})$ définies sur les parties de l'ensemble des sommets V et vérifiant pour toutes communautés $\mathcal{C}_1$ et $\mathcal{C}_2$ disjointes :*

$$h(\mathcal{C}_1\cup\mathcal{C}_2)\geq h(\mathcal{C}_1)+h(\mathcal{C}_2)\ et\ l(\mathcal{C}_1\cup\mathcal{C}_2)\leq l(\mathcal{C}_1)+l(\mathcal{C}_2)$$

h est plus élevé pour les communautés macroscopiques et l est plus élevé pour les communautés microscopiques. Nous définissons à partir de ces deux fonctions la famille de fonctions suivante :

$$Q_\alpha(\mathcal{P})=\sum_{\mathcal{C}\in\mathcal{P}}\alpha h(\mathcal{C})+(1-\alpha)l(\mathcal{C})$$

Théorème 9 *La famille de fonctions Q_α de la Définition 8 est une famille de fonctions multi-échelles et additives.*

Preuve : Tout d'abord les fonctions Q_α sont additives : il suffit de considérer les fonctions élémentaires $q_\alpha(\mathcal{C})=\alpha h(\mathcal{C})+(1-\alpha)l(\mathcal{C})$.

Pour $\alpha=0$ la fonction de qualité $Q_0(\mathcal{P})$ est égale à $\sum_{\mathcal{C}\in\mathcal{P}} l(\mathcal{C})$. L'inégalité $l(\mathcal{C}_1\cup\mathcal{C}_2)\leq l(\mathcal{C}_1)+l(\mathcal{C}_2)$ vérifiée par la fonction l implique que la partition $\mathcal{P}_0=\{\{v\}|v\in V\}$ maximise Q_0. De même, l'inégalité vérifiée par la fonction h implique que la partition $\mathcal{P}_1=\{V\}$ maximise Q_1 (pour $\alpha=1$).

Supposons maintenant que $\alpha_1\leq\alpha_2$ mais $\mathcal{P}_{\alpha_1}\not\preceq\mathcal{P}_{\alpha_2}$. Ceci signifie, comme les partitions sont dans Π, qu'il existe $\mathcal{C}\in\mathcal{P}_{\alpha_1}$ et $\mathcal{C}_1,\ldots,\mathcal{C}_k\in\mathcal{P}_{\alpha_2}$ tels que $\mathcal{C}=\mathcal{C}_1\cup\ldots\cup\mathcal{C}_k$.

Nous avons alors : $q_{\alpha_1}(\mathcal{C})=\alpha_1 h(\mathcal{C})+(1-\alpha_1)l(\mathcal{C})=q_{\alpha_2}(\mathcal{C})+(\alpha_1-\alpha_2)h(\mathcal{C})+(\alpha_2-\alpha_1)l(\mathcal{C})$. Mais $\mathcal{P}_{\alpha_2}$ (contenant $\mathcal{C}_1,\ldots,\mathcal{C}_k$) maximise Q_{α_2}, ainsi : $q_{\alpha_2}(\mathcal{C})\leq q_{\alpha_2}(\mathcal{C}_1)+\ldots+q_{\alpha_2}(\mathcal{C}_k)$.

De plus h et l satisfont : $h(\mathcal{C}) \geq h(\mathcal{C}_1)+\ldots+h(\mathcal{C}_k)$ et $l(\mathcal{C}) \leq l(\mathcal{C}_1)+\ldots+l(\mathcal{C}_k)$. Finalement en utilisant $\alpha_1 \leq \alpha_2$ nous obtenons : $q_{\alpha_1}(\mathcal{C}) \leq q_{\alpha_2}(\mathcal{C}_1)+\ldots+q_{\alpha_2}(\mathcal{C}_k)+(\alpha_1-\alpha_2)(h(\mathcal{C}_1)+\ldots+h(\mathcal{C}_k))+(\alpha_2-\alpha_1)(l(\mathcal{C}_1)+\ldots+l(\mathcal{C}_k))$.

Nous reconnaissons l'inégalité $q_{\alpha_1}(\mathcal{C}) \leq q_{\alpha_1}(\mathcal{C}_1)+\ldots+q_{\alpha_1}(\mathcal{C}_k)$. Ceci est en contradiction avec le fait que la partition $\mathcal{P}_{\alpha_1}$ maximise Q_{α_1}. Ceci prouve la propriété principale de la définition de fonction de qualité multi-échelles. □

Cette définition et ce théorème permettent de construire une famille de fonctions de qualité additives et multi-échelles à partir de deux fonctions élémentaires h et l. Pour construire de bonnes fonctions de qualité, ces deux fonctions doivent capturer des propriétés qui caractérisent les communautés. Elle doivent de plus posséder des comportements de croissance opposés : l'une doit favoriser les grandes communautés et l'autre doit favoriser les petites communautés. Le choix du paramètre α représente alors un compromis entre deux propriétés souhaitables mais incompatibles d'une partition en communautés.

Nous allons proposer des généralisations multi-échelles des trois fonctions de qualité Q^M, Q^P et Q^S présentées en Section 4.2.1. Dans chacun des trois cas, la fonction de qualité originale sera obtenue (à un facteur $\frac{1}{2}$ près qui n'influe pas sur la détermination de la meilleure partition) comme un cas particulier pour le paramètre $\alpha = \frac{1}{2}$.

Définition 9 *Nous définissons (avec les notations introduites en Section 4.2.1) :*

- *La modularité multi-échelles :*

$$Q^M_\alpha(\mathcal{P}) = \sum_{\mathcal{C}\in\mathcal{P}} \alpha e(\mathcal{C}) - (1-\alpha)a(\mathcal{C})^2$$

- *La performance multi-échelles :*

$$Q^P_\alpha(\mathcal{P}) = \frac{\alpha|\{\{u,v\} \in E, \mathcal{C}(u) = \mathcal{C}(v)\}| + (1-\alpha)|\{\{u,v\} \notin E, \mathcal{C}(u) \neq \mathcal{C}(v)\}|}{\frac{1}{2}n(n-1)}$$

- *Une fonction multi-échelles basée sur la similarité des sommets. La quantité $\sigma(\mathcal{C}) = \frac{1}{n}\frac{1}{|\mathcal{C}|}\sum_{i,j\in\mathcal{C}} r_{ij}^2$ doit être définie à partir d'une distance r Euclidienne :*

$$Q^S_\alpha(\mathcal{P}) = -\alpha\frac{|\mathcal{P}|}{n} - (1-\alpha)\sum_{\mathcal{C}\in\mathcal{P}} \frac{\sigma(\mathcal{C})}{\sigma_{max}}$$

Théorème 10 *Les fonctions Q^M_α, Q^P_α et Q^S_α sont des fonctions de qualité multi-échelles vérifiant la Définition 8.*

Preuve : Pour prouver que les fonctions Q^M_α, Q^P_α et Q^S_α sont des fonctions multi-échelles additives, il suffit de montrer dans chaque cas qu'il existe des fonctions élémentaires l et h qui permettent d'appliquer le Théorème 9.

- Q^M_α : Nous prenons $h^M(\mathcal{C}) = e(\mathcal{C})$ la fraction d'arêtes internes à une communauté $\mathcal{C}$ et $l^M(\mathcal{C}) = -a(\mathcal{C})^2$ utilisant la fraction d'arêtes liées à une communauté $\mathcal{C}$. Nous avons $e(\mathcal{C}_1 \cup \mathcal{C}_2) = e(\mathcal{C}_1) + e(\mathcal{C}_2) +$ (fraction d'arêtes entre $\mathcal{C}_1$ et $\mathcal{C}_2$) et donc $h^M(\mathcal{C}_1 \cup \mathcal{C}_2) \geq h^M(\mathcal{C}_1) + h^M(\mathcal{C}_2)$. Nous avons de plus $a(\mathcal{C}_1 \cup \mathcal{C}_2) = a(\mathcal{C}_1) + a(\mathcal{C}_2)$, et donc $l^M(\mathcal{C}_1) + l^M(\mathcal{C}_2) - l^M(\mathcal{C}_1 \cup \mathcal{C}_2) = 2a(\mathcal{C}_1)a(\mathcal{C}_2) \geq 0$.

- Q^P_α : Nous prenons les fonctions élémentaires $h^P(\mathcal{C}) = \frac{1}{n(n-1)} \sum_{u \in \mathcal{C}} |\{v \in \mathcal{C}, \{u,v\} \in E\}|$ et $l^P(\mathcal{C}) = \frac{1}{n(n-1)} \sum_{u \in \mathcal{C}} |\{v \notin \mathcal{C}, \{u,v\} \notin E\}|$. Les inégalités sont faciles à vérifier lorsque nous considérons que $h^P(\mathcal{C})$ compte le nombre d'arêtes internes à $\mathcal{C}$ et $h^P(\mathcal{C})$ compte le nombre de sommets externes à $\mathcal{C}$ qui ne sont pas liés à la communauté.
- Q^S_α : Nous prenons $h^S(\mathcal{C}) = -\frac{1}{n}$ et $l^S(\mathcal{C}) = -\frac{\sigma(\mathcal{C})}{\sigma_{max}}$. h^S vérifie trivialement $h^S(\mathcal{C}_1 \cup \mathcal{C}_2) > h^S(\mathcal{C}_1) + h^S(\mathcal{C}_2)$. Dans le cas d'une quantité $\sigma(\mathcal{C})$ définie à partir d'une distance Eucidienne, le Théorème 4 garantit que la quantité $\Delta\sigma(\mathcal{C}_1, \mathcal{C}_2) = \sigma(\mathcal{C}_1 \cup \mathcal{C}_2) - \sigma(\mathcal{C}_1) - \sigma(\mathcal{C}_2)$ est positive. Ceci permet de conclure $l^S(\mathcal{C}_1 \cup \mathcal{C}_2) \leq l^S(\mathcal{C}_1) + l^S(\mathcal{C}_2)$.

□

Nous avons finalement défini une classe générale de fonctions de qualité additives et multi-échelles ainsi que trois exemples pratiques de telles fonctions qui généralisent des fonctions de qualité connues. Ces fonctions de qualité étant additives, nous pouvons pour tout paramètre α appliquer l'approche proposée dans la Section 4.3 et obtenir une partition correspondant au facteur d'échelle α considéré. Il est cependant difficile de savoir à quelles échelles apparaissent les meilleures structures de communautés. Nous allons donc proposer dans la suite de ce chapitre une approche permettant de résoudre ce problème. Nous allons calculer efficacement toutes les partitions pour tous les paramètres α possibles. Nous allons ensuite déterminer quels sont les α les plus significatifs.

Nous définissons des fonctions de qualité multi-échelles $(Q_\alpha)_{0 \leq \alpha \leq 1}$ paramétrées par un facteur d'échelle α. Les partitions $\mathcal{P}_\alpha$ maximisant Q_α sur Π vérifient un ordre en accord avec le facteur d'échelle : $\alpha_1 \leq \alpha_2 \Rightarrow \mathcal{P}_{\alpha_1} \preceq \mathcal{P}_{\alpha_2}$. Nous pouvons définir à partir de toute paire de fonctions élémentaires h et l vérifiant $h(\mathcal{C}_1 \cup \mathcal{C}_2) \geq h(\mathcal{C}_1) + h(\mathcal{C}_2)$ et $l(\mathcal{C}_1 \cup \mathcal{C}_2) \leq l(\mathcal{C}_1) + l(\mathcal{C}_2)$ une classe générale de fonctions de qualité multi-échelles additives :

$$Q_\alpha(\mathcal{P}) = \sum_{\mathcal{C} \in \mathcal{P}} \alpha h(\mathcal{C}) + (1-\alpha) l(\mathcal{C})$$

Cette classe de fonctions permet de donner une généralisation multi-échelles Q^M_α, Q^P_α et Q^S_α aux fonctions présentées en Section 4.2.1. Ces fonctions seront utilisées dans la section suivante pour calculer efficacement toutes les partitions $(\mathcal{P}_\alpha)_{0 \leq \alpha \leq 1}$.

4.4.2 Meilleure partition pour toutes les échelles

Une fonction de qualité multi-échelles Q_α permet de définir une partition $\mathcal{P}_\alpha$ pour chaque facteur d'échelle $0 \leq \alpha \leq 1$. Nous allons montrer dans cette section comment calculer toutes les partitions $\mathcal{P}_\alpha$ définies par une fonction de qualité additive et multi-échelles (vérifiant la Définition 8).

La Définition 7 des fonctions de qualité multi-échelles impose que $\alpha_1 \leq \alpha_2 \Rightarrow \mathcal{P}_{\alpha_1} \preceq \mathcal{P}_{\alpha_2}$. Cet ordre entre les partitions $\mathcal{P}_{\alpha_1} \preceq \mathcal{P}_{\alpha_2}$ indique que pour toute communauté $\mathcal{C}_1$ de $\mathcal{P}_{\alpha_1}$, il

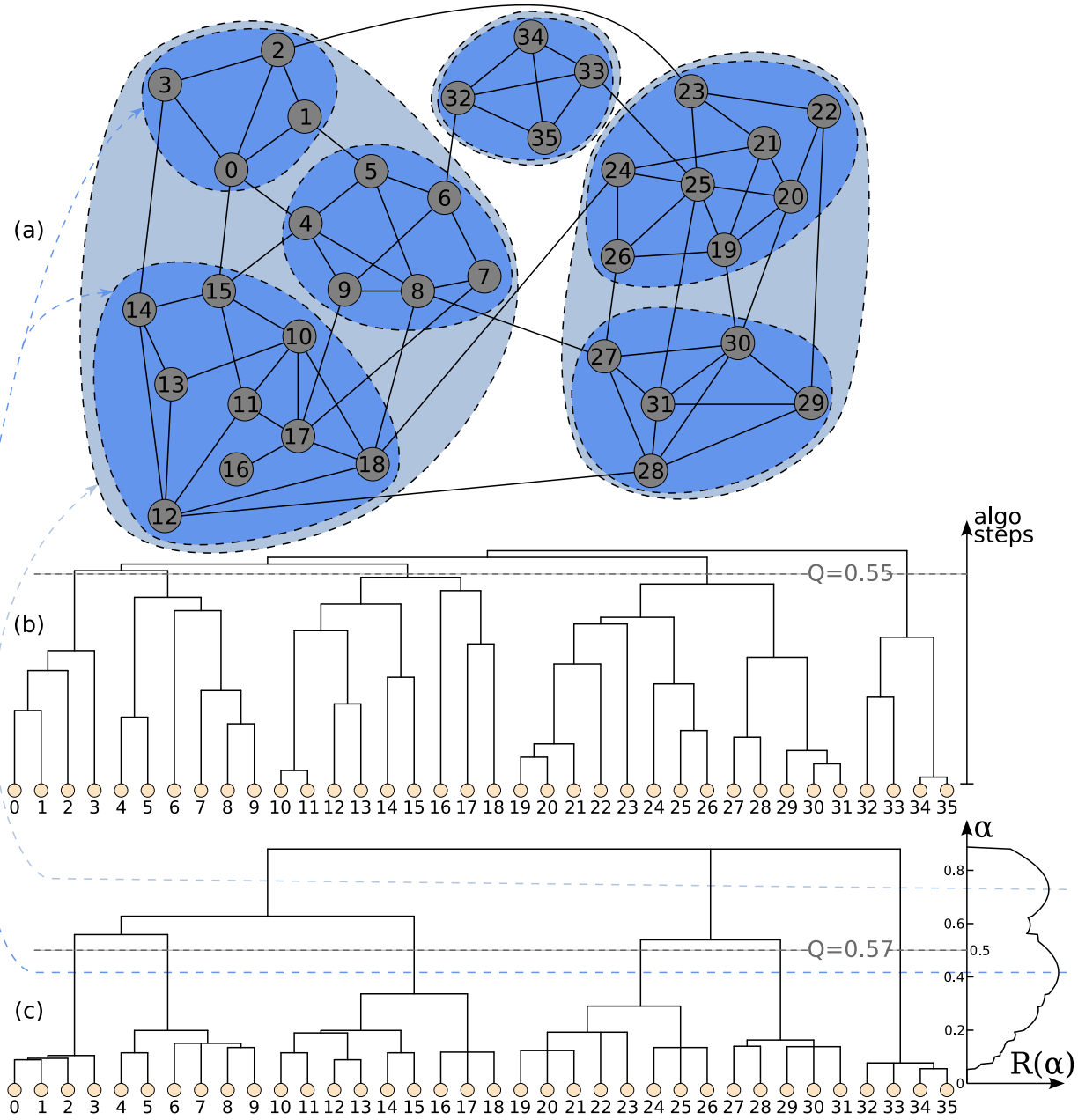

FIG. 4.1 – **(a)** Exemple de graphe avec une structure multi-échelles de communautés. **(b)** Dendrogramme trouvé par l'approche décrite au Chapitre 3. La méthode classique maximise une fonction de qualité en ne considérant que des coupes horizontales de ce dendrogramme, ici une partition en 5 communautés de modularité $Q^M = 0.55$. **(c)** Dendrogramme réordonné en fonction de la modularité multi-échelles Q^M_α. Les coupes horizontales dans ce dendrogramme donnent les meilleures partitions $\mathcal{P}_\alpha$ pour tout facteur d'échelle α. La meilleure modularité $Q^M = 0.57$ (obtenue pour $\alpha = \frac{1}{2}$) améliore la méthode classique. La fonction de pertinence $R(\alpha)$ (définie en Section 4.5) pointe deux facteurs d'échelles significatifs ($\alpha = 0.42$ et $\alpha = 0.73$) correspondant à des partitions de 6 et 3 communautés. Ces deux partitions sont mises en évidence sur le graphe.

existe une communauté $\mathcal{C}_2$ de $\mathcal{P}_{\alpha_2}$ qui est plus grande pour l'inclusion ($\mathcal{C}_1 \subseteq \mathcal{C}_2$). Ceci implique qu'au moins deux communautés sont fusionnées pour passer d'une partition $\mathcal{P}_{\alpha_1}$ à la première partition strictement supérieure différente $\mathcal{P}_{\alpha_2}$. Le nombre de communautés de la seconde partition $\mathcal{P}_{\alpha_2}$ est alors strictement inférieur à celui de $\mathcal{P}_{\alpha_1}$

Le nombre total de partitions $(\mathcal{P}_\alpha)_{0\leq\alpha\leq 1}$ est donc au plus n. En effet, la première partition $\mathcal{P}_0 = \{\{v\}, v \in V\}$ contient n communautés et la dernière partition $\mathcal{P}_1 = \{V\}$ contient une seule communauté. Le nombre de communautés des partitions successives étant strictement décroissant, il y a donc au plus n partitions différentes dans la famille $(\mathcal{P}_\alpha)_{0\leq\alpha\leq 1}$.

Chacune de ces partitions $\mathcal{P}_\alpha$ est la meilleure partition sur une plage $\alpha_i \leq \alpha \leq \alpha_{i+1}$ des facteurs d'échelle. Pour déterminer ces partitions, il suffit de déterminer ces facteurs d'échelle particuliers α_i ainsi que les communautés qui sont fusionnées lors de ces changements de la meilleure partition $\mathcal{P}_\alpha$. Ceci permet de plus de définir un facteur d'échelle de création d'une communauté $\alpha_{min}(\mathcal{C})$, qui correspond au moment α_i auquel la communauté $\mathcal{C}$ apparaît dans la partition $\mathcal{P}_\alpha$. Nous pouvons ainsi réordonner le dendrogramme par ce nouvel ordre entre communautés, et nous obtenons alors de meilleures indications qui permettent de comparer les échelles des communautés. La Figure 4.1 illustre cette transformation de dendrogramme.

Pour toute partition $\mathcal{P} \in \Pi$, la fonction de qualité $Q_\alpha(\mathcal{P}) = l(\mathcal{P}) + (h(\mathcal{P}) - l(\mathcal{P}))\alpha$ peut être vue comme une fonction affine du paramètre α. La valeur maximale de $Q_\alpha(\mathcal{P})$ peut être vue elle aussi comme une une fonction du paramètre alpha :

$$Q^{\Pi}_{max}(\alpha) = \max_{\mathcal{P}\in\Pi} Q_\alpha(\mathcal{P})$$

Chacune des fonctions $Q_\alpha(\mathcal{P})$ étant affine et l'ensemble Π des partitions envisagées étant fini, la fonction $Q^{\Pi}_{max}(\alpha)$ ainsi définie est donc une fonction affine par morceaux. Nous avons vu qu'il y avait au plus n meilleures partitions différentes $(\mathcal{P}_\alpha)_{0\leq\alpha\leq 1}$, ceci implique qu'il existe au plus n points α_i de discontinuité pour cette fonction affine.

La connaissance de cette fonction affine par morceaux $Q^{\Pi}_{max}(\alpha)$ donne les facteurs d'échelle α_i pour lesquels il y a un changement de meilleure partition $\mathcal{P}_\alpha$. Il suffit de plus de savoir quelles sont les communautés qui sont fusionnées pour chaque α_i pour connaître toutes les partitions $(\mathcal{P}_\alpha)_{0\leq\alpha\leq 1}$. Nous allons proposer un algorithme pour calculer $Q^{\Pi}_{max}(\alpha)$ et retenir les fusions de communautés correspondantes. Cet algorithme va calculer récursivement les fonctions affines par morceaux $Q^{\Pi^{\mathcal{C}}}_{max}(\alpha)$ définies comme suit :

Définition 10 *Soit* $Q_\alpha(\mathcal{P}) = \sum_{\mathcal{C}\in\mathcal{P}} \alpha h(\mathcal{C}) + (1-\alpha) l(\mathcal{C})$ *une fonction multi-échelles vérifiant la Définition 8. Nous définissons la fonction affine par morceaux* $Q^{\Pi^{\mathcal{C}}}_{max}(\alpha)$ *qui donne, en fonction du paramètre* α*, le maximum de* Q_α *sur l'ensemble* $\Pi^{\mathcal{C}}$ *des partitions possibles d'une communauté* $\mathcal{C}$*.*

$$Q^{\Pi^{\mathcal{C}}}_{max}(\alpha) = \max_{\mathcal{P}^{\mathcal{C}}\in\Pi^{\mathcal{C}}} Q_\alpha(\mathcal{P}^{\mathcal{C}})$$

Théorème 11 *Soit* $Q_\alpha(\mathcal{P}) = \sum_{\mathcal{C}\in\mathcal{P}} \alpha h(\mathcal{C}) + (1-\alpha) l(\mathcal{C})$ *une fonction multi-échelles vérifiant la Définition 8. Il est possible de calculer la fonction affine par morceaux* $Q^{\Pi}_{max}(\alpha)$ *en faisant au plus* $2n$ *évaluations des fonctions élémentaires* l *et* h*. La complexité additionnelle moyenne est* $\mathcal{O}(n\sqrt{n})$ *pour un dendrogramme arbitraire, elle est de* $\mathcal{O}(n\log(n))$ *pour un dendrogramme équilibré et le pire des cas est* $\mathcal{O}(n^2)$*. La fonction* `PartitionsMultiEchelles` *réalise cela.*

Function `PartitionsMultiEchelles(`$\mathcal{C}$`)`

foreach *sous-communauté* $\mathcal{C}_i$ *de* $\mathcal{C}$ **do**
  $Q_{max}^{\Pi_{\mathcal{C}_i}} \leftarrow$ `PartitionsMultiEchelles(`$\mathcal{C}_i$`)`;
end
if $\mathcal{C}$ *n'a pas de sous-communauté* **then**
  return $\alpha \mapsto q_\alpha(\mathcal{C})$
else
  return $\alpha \mapsto \max(q_\alpha(\mathcal{C}), \sum_i Q_{max}^{\Pi_{\mathcal{C}_i}})$

Preuve : Pour un α fixé, et pour une communauté $\mathcal{C}$ possédant les sous-communautés filles $\mathcal{C}_1, \ldots, \mathcal{C}_k$ dans le dendrogramme, le Lemme 2 indique que $\max_{\mathcal{P} \in \Pi^{\mathcal{C}}} Q_\alpha(\mathcal{P}) = \max\Big(q_\alpha(\mathcal{C}), \sum_i \max_{\mathcal{P} \in \Pi_{\mathcal{C}_i}} Q_\alpha(\mathcal{P})\Big)$. Cette égalité est valable pour tout $\alpha \in [0,1]$, nous en déduisons donc que $Q_{max}^{\Pi^{\mathcal{C}}}(\alpha) = \max(q_\alpha(\mathcal{C}), \sum_i Q_{max}^{\Pi_{\mathcal{C}_i}}(\alpha))$. Ceci prouve la correction de l'approche récursive `PartitionsMultiEchelles` qui calcule $Q_{max}^{\Pi^{\mathcal{C}}}$ en manipulant des fonctions affines par morceaux. La fonction $Q_{max}^{\Pi}(\alpha)$ est obtenue en prenant comme paramètre $\mathcal{C} = V$.

La fonction effectue à chaque appel une évaluation de $q_\alpha(\mathcal{C})$ et une opération d'addition et de maximum sur des fonctions affines par morceaux. Ces fonctions affines par morceaux sont codées par la liste de leurs points de discontinuité $(\alpha_i, Q_{max}^{\Pi_{\mathcal{C}_i}}(\alpha_i))$. Ces opérations d'addition et de maximum sont effectuées en temps linéaire en fonction du nombre de points de discontinuité α_i présents dans la liste. Le nombre de ces points particuliers de la fonction $Q_{max}^{\Pi^{\mathcal{C}}}(\alpha)$ est au plus égal au nombre de feuilles attachées au noeud $\mathcal{C}$ du dendrogramme, c'est à dire au plus $|\mathcal{C}|$. Par conséquent, la complexité d'un appel à la fonction est une évaluation de $q_\alpha(\mathcal{C})$ plus $\mathcal{O}(|\mathcal{C}|)$ opérations supplémentaires pour manipuler les fonctions affines par morceaux.

La fonction récursive est appelée exactement une fois par communauté du dendrogramme, soit $\mathcal{O}(n)$ évaluations des fonctions élémentaires $q_\alpha(\mathcal{C})$ qui nécessitent chacune l'évaluation d'une fonction élémentaire $h(\mathcal{C})$ et $l(\mathcal{C})$. La somme des coûts supplémentaires $\mathcal{O}(|\mathcal{C}|)$ pour toutes les communautés $\mathcal{C}$ du dendrogramme n'est autre que la longueur de cheminement externe du dendrogramme. La longueur de cheminement externe d'un arbre est la somme des longueurs des chemins partant de la racine et allant vers toutes les feuilles de l'arbre. Il passe exactement $|\mathcal{C}|$ chemins par chaque noeud interne du dendrogramme. La somme des longueurs de ces chemins peut être vue comme une somme de 1 pour chaque noeud interne par lequel le chemin passe, soit une somme de $|\mathcal{C}|$ pour chaque noeud interne équivalente à notre complexité.

Une analyse classique [74] montre que la longueur de cheminement est comprise entre $\mathcal{O}(n \log(n))$ et $\mathcal{O}(n^2)$ avec une valeur moyenne (la moyenne est considérée par rapport à l'ensemble des arbres de taille n) en $\mathcal{O}(n\sqrt{n})$. □

Nous avons proposé un algorithme qui calcule la fonction affine par morceaux $Q_{max}^{\Pi}(\alpha)$ en appelant la fonction `PartitionsMultiEchelles` avec le paramètre $\mathcal{C} = V$. Ceci nous permet de connaître tous les points particuliers α_i où les partitions $\mathcal{P}_\alpha$ sont modifiées. Nous pouvons facilement lors du calcul des $Q_{max}^{\Pi^{\mathcal{C}}}(\alpha)$ retenir quelles sont les nouvelles communautés créées pour chaque point particulier α_i. Cette information permet au final de reconstituer les

partitions $\mathcal{P}_\alpha$ pour tout facteur d'échelle α.

La sortie de cet algorithme peut aussi être vue de manière plus intuitive comme un dendrogramme réordonné (Figure 4.1c) dans lequel chaque communauté est placée à la hauteur α_i à laquelle elle apparaît dans la partition $\mathcal{P}_\alpha$. Les meilleures partitions $\mathcal{P}_\alpha$ sont alors obtenues par des coupes horizontales de ce dendrogramme réordonné à la hauteur α désirée. Nous pouvons noter que certaines communautés présentes dans le dendrogramme d'origine peuvent disparaître dans le dendrogramme réordonné et ne jamais intervenir dans les meilleures partitions $\mathcal{P}_\alpha$. Ce phénomène peut être expliqué par le fait que des fusions plus importantes que les fusions données par le dendrogramme d'origine peuvent intervenir lorsque l'on considère ce nouveau dendrogramme réordonné.

Nous pouvons donc calculer efficacement toutes les meilleures partitions $\mathcal{P}_\alpha$ correspondant à tous les facteurs d'échelle possibles $\alpha \in [0, 1]$. Toutes ces partitions n'ont bien sûr pas le même intérêt et ne représentent pas forcément toutes des structures de communautés pertinentes. Nous allons utiliser ces informations pour proposer dans la section suivante une méthode pour déterminer les échelles α les plus pertinentes.

Considérons une fonction de qualité additive et multi-échelles $Q_\alpha(\mathcal{P}) = \sum_{\mathcal{C}\in\mathcal{P}} \alpha h(\mathcal{C}) + (1-\alpha) l(\mathcal{C})$ vérifiant $h(\mathcal{C}_1 \cup \mathcal{C}_2) \geq h(\mathcal{C}_1) + h(\mathcal{C}_2)$ et $l(\mathcal{C}_1 \cup \mathcal{C}_2) \leq l(\mathcal{C}_1) + l(\mathcal{C}_2)$. Nous pouvons calculer efficacement toutes les meilleures partitions $(\mathcal{P}_\alpha)_{0\leq\alpha\leq 1}$ (ce qui représente au plus n partitions différentes) avec une complexité moyenne de $\mathcal{O}(n\sqrt{n})$ en plus des $\mathcal{O}(n)$ évaluations des fonctions élémentaires $h(\mathcal{C})$ et $l(\mathcal{C})$.
Les partitions $(\mathcal{P}_\alpha)_{0\leq\alpha\leq 1}$ permettent d'associer à chaque communauté $\mathcal{C}$ du dendrogramme un facteur d'échelle α correspondant à son apparition dans les partitions $\mathcal{P}_\alpha$. Nous pouvons ainsi réordonner le dendrogramme (Figure 4.1c), ce qui permet de comparer les différentes échelles des communautés et d'obtenir les partitions $\mathcal{P}_\alpha$ par des coupes droites horizontales à la hauteur α qui correspond à l'échelle souhaitée.

4.5 Détermination des échelles pertinentes

Nous avons montré comment obtenir efficacement toutes les partitions $(\mathcal{P}_\alpha)_{0\leq\alpha\leq 1}$. Cependant les différentes échelles données par le paramètre α n'ont pas forcément la même pertinence en termes de structures de communautés. Il est en effet possible que certaines échelles ne représentent pas de bonnes partitions en communautés car le graphe d'origine ne possède pas de communautés à ces échelles. Nous allons évaluer la pertinence de ces partitions pour détecter les facteurs d'échelle α les plus significatifs auxquels apparaissent les structures de communautés les plus marquées.

L'algorithme proposé dans la section précédente permet, pour chaque communauté, de savoir à partir de quels facteurs d'échelle α_i la communauté apparaît puis disparaît des meilleures partitions $\mathcal{P}_\alpha$. Ceci permet de définir une plage de paramètres α pour lesquels la communauté $\mathcal{C}$ fait partie des meilleures partitions $\mathcal{P}_\alpha$.

Définition 11 *Pour toute communauté $\mathcal{C}$ apparaissant dans les partitions $\mathcal{P}_\alpha$, nous définissons les bornes $\alpha_{min}(\mathcal{C})$ et $\alpha_{max}(\mathcal{C})$ entre lesquelles la communauté $\mathcal{C}$ fait partie des partitions*

$\mathcal{P}_\alpha$:

$$\mathcal{C} \in \mathcal{P}_\alpha \Leftrightarrow \alpha_{min}(\mathcal{C}) \leq \alpha \leq \alpha_{max}(\mathcal{C})$$

Nous considérons que les communautés les plus marquées et les plus pertinentes vont apparaître sur des plages étendues de facteurs d'échelle. En effet, les communautés qui restent stables lors de l'évolution des partitions $\mathcal{P}_\alpha$ apparaissent à de nombreuses échelles et traduisent la présence de seuils importants dans la densité ou l'homogénéité correspondant à des frontières nettes entre les communautés. La pertinence d'une communauté $\mathcal{C}$ sera donc mesurée par la durée $\alpha_{max}(\mathcal{C}) - \alpha_{min}(\mathcal{C})$ de présence de cette communauté dans les meilleures partitions. Le facteur d'échelle qui représente le mieux la communauté est $\frac{\alpha_{max}(\mathcal{C})-\alpha_{min}(\mathcal{C})}{2}$. Ces considérations nous permettent de définir la fonction de pertinence suivante.

Définition 12 *Nous définissons la pertinence $R_\alpha(\mathcal{C})$ d'une communauté $\mathcal{C}$ en fonction du facteur d'échelle α par :*

$$R_\alpha(\mathcal{C}) = \frac{\alpha_{max}(\mathcal{C}) - \alpha_{min}(\mathcal{C})}{2} + \frac{2(\alpha_{max}(\mathcal{C}) - \alpha)(\alpha - \alpha_{min}(\mathcal{C}))}{\alpha_{max}(\mathcal{C}) - \alpha_{min}(\mathcal{C})}$$

Ceci conduit à définir la fonction de pertinence globale $R(\alpha)$:

$$R(\alpha) = \sum_{\mathcal{C}\in\mathcal{P}_\alpha} \frac{|\mathcal{C}|R_\alpha(\mathcal{C})}{n} = \sum_{\mathcal{C}\in\mathcal{P}_\alpha} \frac{|\mathcal{C}|}{n}\Big(\frac{\alpha_{max}(\mathcal{C}) - \alpha_{min}(\mathcal{C})}{2} + \frac{2(\alpha_{max}(\mathcal{C}) - \alpha)(\alpha - \alpha_{min}(\mathcal{C}))}{\alpha_{max}(\mathcal{C}) - \alpha_{min}(\mathcal{C})}\Big)$$

Les fonctions $R_\alpha(\mathcal{C})$ ainsi définies sont des fonctions quadratiques en α qui admettent leur maximum $R(\frac{\alpha_{max}(\mathcal{C})-\alpha_{min}(\mathcal{C})}{2}) = \alpha_{max}(\mathcal{C}) - \alpha_{min}(\mathcal{C})$ pour le paramètre α qui caractérise le mieux chaque communauté $\mathcal{C}$. Leurs valeurs aux bornes α_{min} et α_{max} valent $R(\alpha_{min}(\mathcal{C})) = R(\alpha_{max}(\mathcal{C})) = \frac{\alpha_{max}(\mathcal{C})-\alpha_{min}(\mathcal{C})}{2}$. La fonction de pertinence globale $R(\alpha)$ est une moyenne des fonctions $R_\alpha(\mathcal{C})$ pondérées par la taille des communautés $\mathcal{C}$ de la partition $\mathcal{P}_\alpha$ correspondante.

Nous pouvons utiliser cette fonction pour trouver l'échelle α la plus pertinente qui maximise $R(\alpha)$. Nous pouvons aussi nous orienter vers les différents maxima locaux qui peuvent correspondre à d'autres partitions intéressantes. La Figure 4.1c montre l'évolution de $R(\alpha)$ en fonction du paramètre α et met en évidence la possibilité de trouver plusieurs échelles qui correspondent à plusieurs partitions envisageables.

Le calcul de $R(\alpha)$ et de ses maxima peut être fait en temps $\mathcal{O}(n)$. $R(\alpha)$ est une fonction quadratique qui peut être mise sous la forme $R(\alpha) = A\alpha^2 + B\alpha + C$ entre chaque α_i qui correspondent à des changements de partition $\mathcal{P}_\alpha$. À chaque α_i nous pouvons modifier les coefficients A, B et C en fonction des coefficients des $R_\alpha(\mathcal{C})$ correspondants aux communautés fusionnées en α_i. Chaque communauté conduit à deux mises à jour des coefficients (une lorsque la communauté est créée en $\alpha_{min}(\mathcal{C})$ et l'autre lorsque la communauté disparaît en $\alpha_{max}(\mathcal{C})$). La complexité totale de ces mises à jour est donc en $\mathcal{O}(n)$. Pour calculer les maxima locaux, il suffit de considérer pour chaque plage $[\alpha_i, \alpha_{i+1}]$ les deux bornes et l'éventuel maximum de la fonction $R(\alpha)$ (qui est une fonction quadratique en α sur chacun des intervalles considérés) si celui-ci est compris entre ces deux bornes. Ces 3 points sont les seuls points pour lesquels il peut y avoir un maximum sur la plage $[\alpha_i, \alpha_{i+1}]$, ce qui conduit à considérer au plus $3n$ points. Nous pouvons donc déterminer les maxima locaux de $R(\alpha)$ en $\mathcal{O}(n)$.

Nous définissons les facteurs d'échelle $\alpha_{min}(\mathcal{C})$ et $\alpha_{max}(\mathcal{C})$ auxquels apparaît puis disparaît chaque communauté $\mathcal{C}$ des partitions $\mathcal{P}_\alpha$. La pertinence d'une communauté peut être mesurée par la durée $\alpha_{max}(\mathcal{C}) - \alpha_{min}(\mathcal{C})$ pendant laquelle elle est présente dans les partitions $\mathcal{P}_\alpha$. Ceci nous amène à définir une fonction $R(\alpha)$ qui mesure la pertinence des facteurs d'échelle α :

$$R(\alpha) = \sum_{\mathcal{C} \in \mathcal{P}_\alpha} \frac{|\mathcal{C}| R_\alpha(\mathcal{C})}{n}$$

$$\text{où } R_\alpha(\mathcal{C}) = \frac{\alpha_{max}(\mathcal{C}) - \alpha_{min}(\mathcal{C})}{2} + \frac{2(\alpha_{max}(\mathcal{C}) - \alpha)(\alpha - \alpha_{min}(\mathcal{C}))}{\alpha_{max}(\mathcal{C}) - \alpha_{min}(\mathcal{C})}$$

Cette fonction permet de déterminer les facteurs d'échelle α auxquels apparaissent les structures de communautés les plus pertinentes en considérant les maxima locaux de $R(\alpha)$ (Figure 4.1c).

Chapitre 5

Analyse expérimentale et comparaison de performances

Les chapitres précédents ont été consacrés à la présentation d'un algorithme pour la détection de communautés dans les grands graphes de terrain et d'une approche permettant l'analyse des communautés multi-échelles. Nous allons ici confronter en pratique notre algorithme aux principales autres approches de détection de communautés. Cette évaluation des performances a été effectuée au travers de comparaisons directes des différents algorithmes sur de nombreux graphes tests, certains artificiels (générés aléatoirement) et d'autres issus directement des applications. Nous mesurerons ensuite les apports des fonctions de qualité multi-échelles et des méthodes de coupe des dendrogrammes proposées. Nous évaluerons finalement l'influence de la longueur des marches aléatoires utilisées dans notre approche.

5.1 Protocole d'évaluation expérimentale

L'évaluation des performances d'un algorithme de détection de communautés est une tâche difficile car nous devons déterminer si les communautés trouvées sont effectivement des communautés pertinentes. L'idéal serait de disposer d'une partition de référence qui indique quelles sont les "bonnes" communautés. Malheureusement ce genre de partition est en pratique très rarement accessible car ceci nécessiterait l'intervention manuelle d'un expert pouvant juger de la qualité des communautés. Il de plus difficile de comparer directement des partitions à une liste préétablie de communautés car il peut exister, pour un même graphe, plusieurs "bonnes" partitions dont les communautés diffèrent légèrement, ou même fortement.

Une des méthodes largement utilisées pour pallier l'impossibilité d'obtenir une partition de référence dans les cas pratiques est de générer aléatoirement des graphes tests présentant une structure de communautés de référence, construite lors de la génération. Ceci permet d'obtenir des données plus nombreuses pour effectuer des statistiques plus précises. Nous allons utiliser ici cette approche pour produire un ensemble de graphes tests qui permettra de comparer de manière intensive les principaux algorithmes proposés.

5.1.1 Génération des graphes de test

Nous allons proposer un modèle de génération de graphes aléatoires non-pondéré qui présente une structure de communautés. Ce modèle basé sur un modèle de graphes aléatoires

du type Erdős-Renyi prendra plusieurs paramètres pour permettre de balayer le plus grand nombre de types de graphes possible. Les paramètres que nous pourront choisir seront les suivants :

- le nombre k de communautés et leurs tailles $|\mathcal{C}_i|$,
- la densité de chaque communauté donnée par leur degré moyen interne $d_{in}(\mathcal{C}_i)$,
- la modularité espérée Q_e^M de la partition ainsi générée.

Notons tout d'abord que le nombre de sommets du graphe est déterminé par le nombre de communautés et leurs tailles, en effet $n = \sum_{i=1}^{k} |\mathcal{C}_i|$. Le degré moyen interne $d_{in}(\mathcal{C}_i)$ permet de déterminer la densité moyenne de la communauté $\frac{d_{in}(\mathcal{C}_i)}{|\mathcal{C}_i|-1}$.

Pour limiter le nombre de paramètres, nous avons considérons que le degré moyen externe de chaque communauté est proportionnel à leur degré interne : $\forall i, d_{out}(\mathcal{C}_i) = \beta d_{in}(\mathcal{C}_i)$. Le choix du paramètre β permet de déterminer la modularité espérée Q_e^M de la partition $\mathcal{P} = \{\mathcal{C}_i, 1 \leq i \leq k\}$ ainsi générée. Q_e^M est calculée à partir des fractions espérées d'arêtes internes $e(\mathcal{C}_i) = \frac{d_{in}(\mathcal{C}_i)|\mathcal{C}_i|}{2m}$ et des fractions espérées d'arêtes liées aux communautés $a(\mathcal{C}_i) = \frac{(d_{in}(\mathcal{C}_i)+d_{out}(\mathcal{C}_i))|\mathcal{C}_i|}{2m}$ où le nombre d'arêtes espéré du graphe est $m = \frac{1}{2}\sum_{i=1}^{k}(d_{in}(\mathcal{C}_i) + d_{out}(\mathcal{C}_i))|\mathcal{C}_i|$. Ceci permet d'obtenir l'expression suivante pour la modularité $Q_e^M = \sum_i e(\mathcal{C}_i) - a(\mathcal{C}_i)^2$:

$$Q_e^M = \sum_{i=1}^{k} \frac{d_{in}(\mathcal{C}_i)|\mathcal{C}_i|}{\sum_{i=1}^{k}(d_{in}(\mathcal{C}_i) + d_{out}(\mathcal{C}_i))|\mathcal{C}_i|} - \frac{((d_{in}(\mathcal{C}_i) + d_{out}(\mathcal{C}_i))|\mathcal{C}_i|)^2}{(\sum_{i=1}^{k}(d_{in}(\mathcal{C}_i) + d_{out}(\mathcal{C}_i))|\mathcal{C}_i|)^2}$$

En utilisant la contrainte $d_{out}(\mathcal{C}_i) = \beta d_{in}(\mathcal{C}_i)$, nous obtenons l'expression simplifiée :

$$Q_e^M = \frac{1}{1+\beta} - \frac{\sum_{i=1}^{k}(d_{in}(\mathcal{C}_i)|\mathcal{C}_i|)^2}{(\sum_{i=1}^{k} d_{in}(\mathcal{C}_i)|\mathcal{C}_i|)^2}$$

Cette expression permet de fixer le facteur β de manière adéquate pour obtenir la modularité Q_e^M souhaitée. Notons que certains choix de Q_e^M peuvent aboutir à des valeurs de β négatives. Dans ce cas le choix de Q_e^M est impossible à réaliser et nous devons donc le rejeter.

Une fois ces paramètres choisis, nous allons générer des graphes aléatoirement en fixant une probabilité d'existence pour chaque arête. Ce modèle de génération de graphe aléatoires est basé sur le modèle de Gilbert [32] qui est souvent dénommé (à tort) comme modèle d'Erdős-Renyi [24]. Pour les arêtes internes la probabilité d'existence de chaque arête correspond à la densité de chaque communauté $\frac{d_{in}(\mathcal{C}_i)}{|\mathcal{C}_i|-1}$. Nous pouvons de la même manière établir une probabilité d'existence des arêtes entre chaque paire de communauté. Cette méthode simple de génération de graphe demande $\mathcal{O}(n^2)$ opérations, ce qui n'est pas suffisamment efficace pour générer de très grands graphes. Nous allons décrire une méthode équivalente plus efficace qui fonctionne de manière linéaire en fonction du nombre d'arêtes.

Plutôt que de tirer aléatoirement chaque arête possible, nous allons d'abord choisir aléatoirement le degré interne et le degré externe de chaque sommet. Nous attribuons ainsi des demi-arêtes à chaque sommet que nous connectons dans un second temps. Pour simuler le processus de génération de graphe selon le modèle de tirage aléatoire de chaque arête, nous commençons par tirer aléatoirement les degrés internes et externes des sommets selon les lois binomiales correspondantes. Nous pouvons traiter séparément l'assortiment des liens internes de chaque communauté et celui des liens externes. Dans chaque cas, nous connectons les demis-arêtes successivement en choisissant aléatoirement deux demi-arêtes non encore liées (en

évitant pour la seconde les boucles et les arêtes doubles). S'il n'existe pas de seconde demi-arête valide libre, nous choisissons aléatoirement une demi-arête valide déjà liée dont nous annulons l'association antérieure. Ceci constitue un échec d'association que nous reportons sur une autre demi-arête. Nous limitons ce nombre d'échecs à un certain seuil à partir duquel nous abandonnons totalement la génération du graphe. En pratique les paramètres choisis permettent de connecter facilement le graphe et il est exceptionnel que le nombre d'échecs dépasse quelques dizaines. Nous arrêtons le processus lorsqu'il reste moins de deux demi-arêtes à lier. Lorsque le nombre de demi-arêtes initial est impair il est bien sûr impossible de toutes les connecter, dans ce cas l'appariement s'arrête lorsqu'il n'en reste plus qu'une.

Nous générons des graphes test présentant une structure de communautés selon les paramètres suivants : le nombre k de communautés et leur taille $|\mathcal{C}_i|$, les degrés internes de chaque communauté $d_{in}(\mathcal{C}_i)$ et la modularité espérée Q_e^M de la partition ainsi générée $\mathcal{P} = \{\mathcal{C}_i, 1 \leq i \leq k\}$. Les arêtes sont alors tirées aléatoirement (à la façon du modèle Erdős-Renyi) selon des probabilités fixées par ces paramètres.

5.1.2 Évaluation de la qualité d'une partition

Nous pouvons générer des graphes aléatoires présentant une structure de communautés connue. Cette partition servira de référence pour évaluer les performances des différents algorithmes sur ces graphes. Pour évaluer la qualité d'une partition trouvée par un algorithme, nous devons mesurer la similarité entre la partition trouvée et la partition de référence.

Cette mesure doit permettre de comparer des partitions de tailles différentes. En effet il est possible de trouver deux partitions proches mais contenant des nombres de communautés différents. Par exemple si deux communautés d'une des deux partitions sont fusionnées dans l'autre partition la structure globale des deux partitions restent assez proche. Il est de plus possible de trouver des partitions proches n'ayant aucune communauté en commun, par exemple si toutes les communautés des deux partitions sont des grandes communautés qui ne diffèrent que sur quelques rares sommets, les deux partitions sont encore une fois assez proches.

Pour prendre en compte ces considérations, nous allons utiliser une version modifiée de l'indice de Rand [69] qui mesure une distance entre les partitions d'un même ensemble. L'indice brut de Rand $I(\mathcal{P}_1, \mathcal{P}_2)$ représente le taux de paires de sommets corrélées par les deux partitions $\mathcal{P}_1$ et $\mathcal{P}_2$. Une paire de sommets $\{i, j\}$ est dite corrélée par $\mathcal{P}_1$ et $\mathcal{P}_2$ si, du point de vue de ces deux partitions, les deux sommets sont tous les deux classés dans une même communauté ou tous les deux classées dans deux communautés différentes. Nous comptabilisons ainsi le nombre de paires de sommets classés de la même manière (ensemble ou séparés) dans les deux partitions. Ceci conduit à la définition suivante :

$$I(\mathcal{P}_1, \mathcal{P}_2) = \frac{|\{i,j\}, \mathcal{C}_1(i) = \mathcal{C}_1(j) \text{ et } \mathcal{C}_2(i) = \mathcal{C}_2(j)| + |\{i,j\}, \mathcal{C}_1(i) \neq \mathcal{C}_1(j) \text{ et } \mathcal{C}_2(i) \neq \mathcal{C}_2(j)|}{\frac{1}{2}n(n-1)}$$

où $\mathcal{C}_1(i)$ et $\mathcal{C}_2(i)$ représentent respectivement les communautés du sommet $i \in V$ dans les partitions $\mathcal{P}_1$ et $\mathcal{P}_2$.

La valeur de cet indice de Rand dépend en partie du nombre de communautés et de leurs tailles : ces deux grandeurs ont une influence statistique sur le nombre moyen de paires de sommets corrélées. En particulier, l'espérance de l'indice de Rand de deux partitions aléatoires dépend des tailles des communautés des deux partitions. Ceci pose un problème important car nous ne pouvons pas comparer de manière absolue deux indices de Rand obtenus entre différentes partitions.

Pour remédier à ce problème, Hubert et Arabie [41] ont proposé de modifier l'indice de Rand de manière à obtenir une espérance nulle pour des partitions aléatoires. L'indice de Rand modifié $I' = \frac{I - I_{esp}}{I_{max} - I_{esp}}$ est une version normalisée de l'indice de Rand brut par rapport à sa valeur maximale I_{max} et sa valeur espérée I_{esp} (lorsque l'on considère deux partitions aléatoires ayant les mêmes tailles de communautés que les partitions que l'on souhaite comparer). La nouvelle valeur est donc inférieure à 1 et a une espérance nulle lorsque les partitions sont tirées aléatoirement. La propriété suivant permet de calculer en temps linéaire l'indice de Rand modifié I' entre deux partitions.

Propriété 3 *L'indice de Rand modifié I' peut se calculer efficacement selon la formule suivante :*

$$I'(\mathcal{P}_1, \mathcal{P}_2) = \frac{\binom{n}{2} \sum_{i,j} \binom{|\mathcal{C}_i^1 \cap \mathcal{C}_j^2|}{2} - \sum_i \binom{|\mathcal{C}_i^1|}{2} \sum_j \binom{|\mathcal{C}_j^2|}{2}}{\frac{1}{2}\binom{n}{2} \left(\sum_i \binom{|\mathcal{C}_i^1|}{2} + \sum_j \binom{|\mathcal{C}_j^2|}{2} \right) - \sum_i \binom{|\mathcal{C}_i^1|}{2} \sum_j \binom{|\mathcal{C}_j^2|}{2}}$$

où les $(\mathcal{C}_i^x)_{1 \leq i \leq k_x}$ représentent les communautés de la partition $\mathcal{P}_x$ ($x \in \{1, 2\}$) et les $\binom{k}{2} = \frac{k(k-1)}{2}$ représentent des facteurs binomiaux.

Preuve : Il est possible de transformer l'expression de l'indice brut de Rand de la manière suivante : $\binom{n}{2} I(\mathcal{P}_1, \mathcal{P}_2) = \binom{n}{2} - \sum_i \binom{|\mathcal{C}_i^1|}{2} - \sum_j \binom{|\mathcal{C}_j^2|}{2} + 2 \sum_{i,j} \binom{|\mathcal{C}_i^1 \cap \mathcal{C}_j^2|}{2}$. Cette expression compte le nombre de paires de sommets corrélées de la façon suivante : on considère que toutes les paires sont corrélées, puis on retire les paires de sommets qui sont dans la même communauté sur $\mathcal{P}_1$ puis sur $\mathcal{P}_2$, on corrige finalement car les paires dans une même communauté à la fois dans $\mathcal{P}_1$ et $\mathcal{P}_2$ ont été comptabilisées à tort deux fois comme négatives.

Cette expression ne fait intervenir que des tailles de communautés ou d'intersections de communautés. Pour calculer l'expression de I_{esp} Hubert et Arabie montrent [41] que le calcul d'espérance se simplifie grâce à :

$$E\left[\sum_{i,j} \binom{|\mathcal{C}_i^1 \cap \mathcal{C}_j^2|}{2} \right] = \frac{\sum_i \binom{|\mathcal{C}_i^1|}{2} \sum_j \binom{|\mathcal{C}_j^2|}{2}}{\binom{n}{2}}$$

La valeur maximale absolue I_{max} est égale à 1. Il suffit alors de reporter les différentes expressions dans la définition $I' = \frac{I - I_{esp}}{I_{max} - I_{esp}}$ puis de simplifier la fraction pour obtenir le résultat. □

Cette expression de $I'(\mathcal{P}_1, \mathcal{P}_2)$ permet de calculer efficacement l'indice de Rand modifié. En effet il suffit de connaître la taille des communautés de chaque partition et des intersections de ces communautés. Le nombre d'intersections non vides de communautés est au plus n (il y a au moins un sommet dans une intersection non-vide). Le nombre total d'opérations est donc en $\mathcal{O}(n)$ en ne sommant que sur les intersections de communautés non-vides.

Nous mesurerons les performances d'un algorithme, sur des graphes tests générés aléatoirement, par l'indice de Rand modifié par Hubert et Arabie [41] $I'(\mathcal{P}_1, \mathcal{P}_2)$ entre la partition de référence du graphe test et la partition trouvée par l'algorithme. Il s'agit d'une version normalisée de l'indice brut de Rand [69] qui compte le nombre de paires de sommets corrélées par les deux partitions. La qualité du résultat est d'autant meilleure que l'indice I' s'approche de sa valeur maximale 1. Cet indice est nul pour des partitions aléatoires.

5.2 Comparaison des différents algorithmes de détection de communautés

Nous présentons dans cette section des résultats expérimentaux qui comparent différentes méthodes de détection de communautés. Nous avons effectué les comparaisons sur un ensemble de graphes tests générés aléatoirement. Afin de garantir un certaine équité des tests, les différentes implémentations des algorithmes testés ont été exécutées sur les mêmes graphes tests et sur la même machine.
Nous allons comparer les approches suivantes (les sigles entre parenthèses seront utilisés sur les graphiques pour repérer chaque approche) :

- notre approche (Walktrap) avec des marches aléatoires de longueur $t = 5$ (WT 5) et $t = 2$ (WT 2),
- l'algorithme de Girvan et Newman (GN) [33, 59] (approche divisive qui retire les arêtes de plus grande centralité à chaque étape),
- l'algorithme rapide qui optimise la modularité (FM) proposé par Newman [58] et amélioré par Clauset *et al* [16] (approche gloutonne de maximisation de la modularité),
- la première approche de Donetti et Muñoz (DML) [19] utilisant ma matrice Laplacienne et sa version améliorée qui utilise une version normalisée de cette matrice (DM) [20] (approches spectrales couplées à des méthodes de clustering hiérarchique),
- l'algorithme Netwalk (NW) [88] (basé sur le temps moyen d'atteinte d'un sommet par des marches aléatoires et sur une méthode de clustering hiérarchique),
- le Markov Cluster Algorithm (MCL) [79] (basé sur des simulations de flots/marches aléatoires),
- l'algorithme de Duch et Arenas (DA) [22] (optimisation par recuit simulé de la modularité),
- l'algorithme Cosmoweb (CW) [9] (approche gravitationnelle).

Nous avons considéré deux versions de notre approche avec des marches aléatoires de longueur $t = 5$ et $t = 2$. Les longueurs $t = 2$ sont très courtes et donneront des résultats de qualités inférieures aux résultats obtenus pour des longueurs $t = 5$. Cependant avec des marches si courtes, il est possible de calculer les probabilités de transition de manière très

efficace en ne prenant en compte que le voisinage à distance 2 de chaque sommet. Cette seconde approche sera donc plus rapide et permettra de traiter des graphes de plus grandes tailles. Nous avons considéré cette longueur de marches pour pouvoir comparer notre approche aux algorithmes (FM) et (CW) qui permettent de traiter de très grands graphes.

La partition retenue en sortie de nos deux approche (WT5 et WT2) est la partition maximisant la modularité Q^M obtenue selon la méthode d'optimisation d'une fonction de qualité décrite au chapitre précédent.

Nous tenons à remercier tous les auteurs ayant fourni ou ayant rendu publique une implémentation de leur approche. A l'exception de l'approche de Girvan et newman (GN), toutes les implémentations des approches testées ont été réalisées par leurs auteurs respectifs. Ceci limite les sources de bugs possibles et permet de garantir un minimum d'optimisation pour chaque implémentation.

5.2.1 Comparaison sur des graphes homogènes

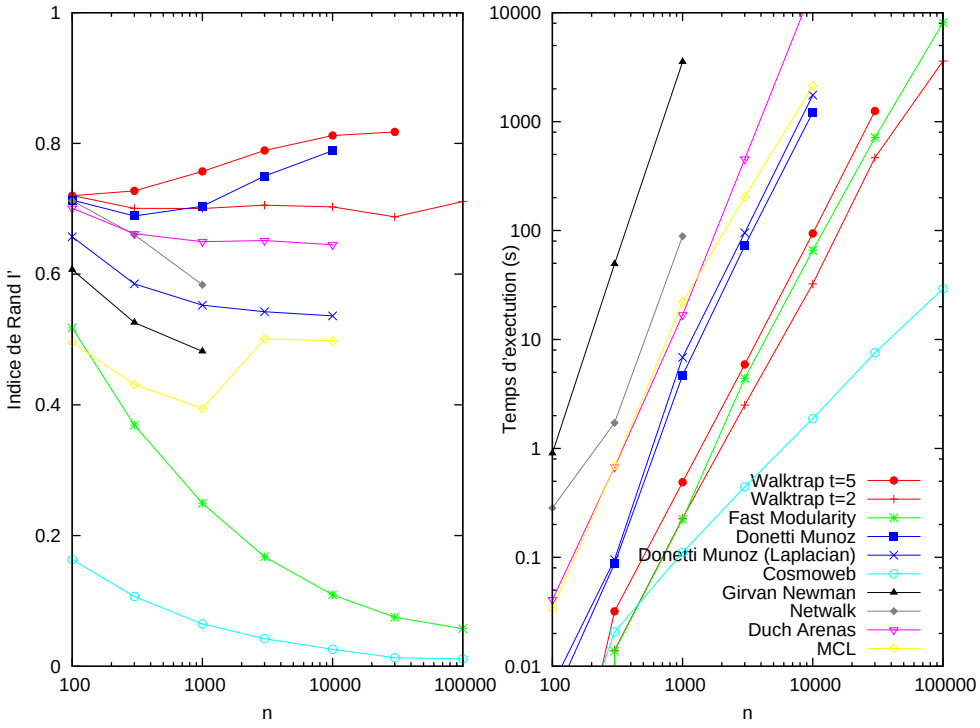

FIG. 5.1 – Performances des principales approches de détection de communautés sur des graphes tests homogènes générés aléatoirement. Les deux graphiques sont donnés en fonction du nombre n de sommets des graphes. Gauche : indice de Rand modifié moyen I' mesurant la qualité des partitions trouvées ; droite : temps moyen d'exécution (en secondes) des algorithmes.

Nous allons d'abord effectuer une comparaison des algorithmes sur des graphes tests simples qui présentent des structures de communautés homogènes. Nous générerons ainsi un premier jeu de graphes tests possédant chacun plusieurs communautés de même taille et de

même densité. Les communautés de chaque graphe sont similaires mais nous allons choisir plusieurs paramètres pour obtenir une grande diversité des communautés à travers l'ensemble des graphes générés.
Ce jeu de graphes tests est caractérisé par le choix des paramètres suivants :

- Nous considérons des tailles de graphes de $n = 100$, 300, 1 000, 3 000, 10 000, 30 000 ou 100 000 sommets.
- Pour chaque taille de graphe, nous considérons 3 nombres de communautés $k = n^{\gamma}$ avec $\gamma = 0.3$, 0.42 ou 0.5. Ceci permet d'obtenir de nombreuses petites communautés avec $k = n^{0.5}$ ou peu de grandes communautés avec $k = n^{0.3}$. Par exemple pour $n = 1\,000$ ces deux paramètres donnent respectivement 32 communautés de 31-32 sommets et 8 communautés de 125 sommets.
- Le degré interne permettant de définir la densité interne est de la forme : $d_{in}(\mathcal{C}_i) = \alpha \ln(|\mathcal{C}_i|)$ où $\alpha \in \{2, 4, 6, 8, 10\}$.
- Nous considérons 6 niveaux possibles pour la modularité moyenne des graphes générés $Q_e^M \in \{0.2, 0.3, 0.4, 0.5, 0.6\}$. Ce choix de modularité permet de fixer les densités entre les communautés.

Pour que le jeu de graphes tests soit le plus diversifié possible, nous allons générer des graphes correspondant à toutes les combinaisons possibles de ces paramètres. Ainsi pour une taille de graphe donnée, il existera 75 types de graphes différents correspondant aux 3 choix de taille de communautés, aux 5 choix des densité interne et aux 5 choix de modularité. Cette diversité des paramètres permet de réduire l'impact des paramètres qui avantageraient ou désavantageraient particulièrement une approche.

Pour chacun de ces 75 jeux de paramètres, nous avons généré aléatoirement 100 graphes pour les tailles $n \leq 1\,000$, 20 graphes de taille $n = 3\,000$, 8 graphes de taille $n = 10\,000$, 2 graphes de taille $n = 30\,000$ et un graphe de taille $n = 100\,000$. Ce qui conduit à un premier jeu de 24825 graphes tests. Notons que nous avons du limiter le nombre de graphes de grande taille pour la double raison des temps de calculs des algorithmes sur ces graphes et de l'espace disque nécessaire au stockage de telles données.

Les approches de détection de communautés ont été testées sur ce premier jeu de graphes tests dans les mêmes conditions. Les résultats généraux sont reportés sur la Figure 5.1 qui regroupe les temps de calculs et les indices de Rand I' moyens réalisés par les différents algorithmes. Notons que chaque point des graphiques représente une moyenne sur l'ensemble des graphes d'une taille donnée. Certains algorithmes n'ont pas pu être testés sur les plus grands graphes pour des raisons de temps de calcul trop importants.

Du point de vue de du temps de calcul, notre approche (WT) s'établit parmi les plus rapide : elle concurrence l'approche rapide d'optimisation de la modularité (FM). Notons que le passage de marches aléatoires de longueur normale $t = 5$ à des marches aléatoires très courtes $t = 2$ permet un gain de temps de calcul d'un facteur de 2 à 3. Nous avons pu avec notre approche traiter des graphes allant jusqu'à un million de sommets, seule l'approche Cosmoweb (CW) a des temps de calculs suffisamment rapides pour pouvoir traiter des graphes de tailles supérieures. Les algorithmes de Girvan et Newman (GN) et Netwalk (NW) sont significativement plus lents que les autres approches et nous n'avons pas pu traiter des graphes de plus de 1000 sommets avec ces deux approches. Les autres approches possèdent des temps de calculs intermédiaires qui permettent de traiter raisonnablement les graphes jusqu'à 10 000 sommets.

Notre approche se comporte de manière comparable à l'algorithme (FM) en ce qui concerne le temps de calcul. Il est cependant à noter que nous avons montré une complexité de notre

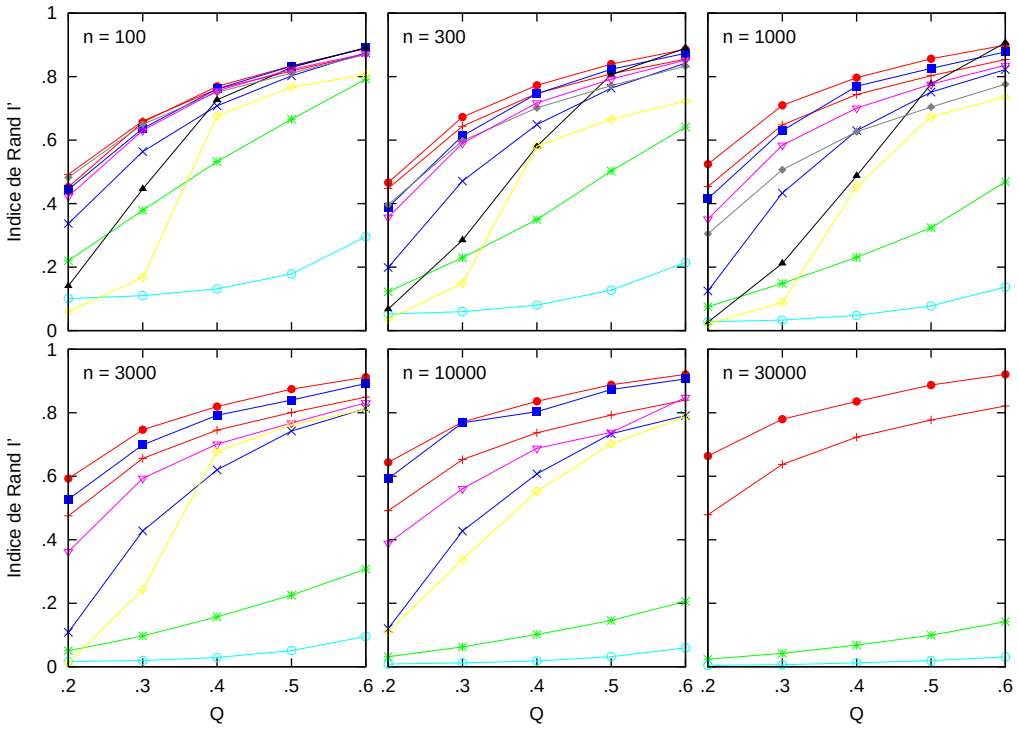

FIG. 5.2 – Qualité des résultats (Indice de Rand modifié I') en fonction de la modularité Q des graphes générés et du nombre n de sommets. La légende des différentes approches est donnée en Figure 5.1.

approche en $\mathcal{O}(mnH)$ (où H est la hauteur du dendrogramme obtenu) alors que la complexité annoncée de l'approche (FM) [16] est en $\mathcal{O}(mH\log(n))$. Ceci peut en partie être expliqué par le fait que nous avons prouvé une borne supérieure de la complexité et que notre approche se comporte en pratique mieux que la borne établie.

Du point de vue de la qualité des partitions trouvées, notre approche avec des marches de longueur $t = 5$ (WT5) obtient les meilleurs résultats quelle que soit la taille des graphes. Le passage à des marches très courtes $t = 2$ (WT2) dégrade logiquement la qualité des résultats qui restent tout de même acceptables. L'approche améliorée de Donetti et Muñoz obtient aussi de très bons résultats, comparables aux nôtres. Ceci s'explique en partie par le fait que ces deux méthodes sont très proches car les vecteurs propres utilisés par l'approche (DM) sont en fait les mêmes vecteurs propres que ceux de la matrice de transition des marches aléatoires que nous utilisons. La qualité des résultats des deux approches rapides (FM) et (CW) décroît fortement avec la taille des graphes : elles obtiennent des résultats en deçà de la moyenne pour les plus grands graphes. Les autres approches obtiennent des résultats intermédiaires et nous pouvons noter qu'à l'exception de notre approche, de celle de Donetti et Muñoz et de MCL, la qualité des résultats des autres approches se déprécie lorsque la taille des graphes augmente (nous n'avons pas d'explication pour la discontinuité observée sur la courbe des résultats de MCL).

Les résultats des différents algorithmes en fonction de la modularité des graphes tests sont

donnés sur la Figure 5.2. On s'aperçoit logiquement que pour toutes les approches la qualité augmente avec la modularité. En effet, plus la modularité du graphe test est élevée, plus les communautés générées sont clairement séparées, avec moins de liens inter-communautaires et plus de liens intra-communautaires. . Ce comportement similaire de tous les algorithmes n'est pas surprenant et il est normal que la détection de communautés soit plus facile lorsque celles-ci sont plus clairement identifiés.

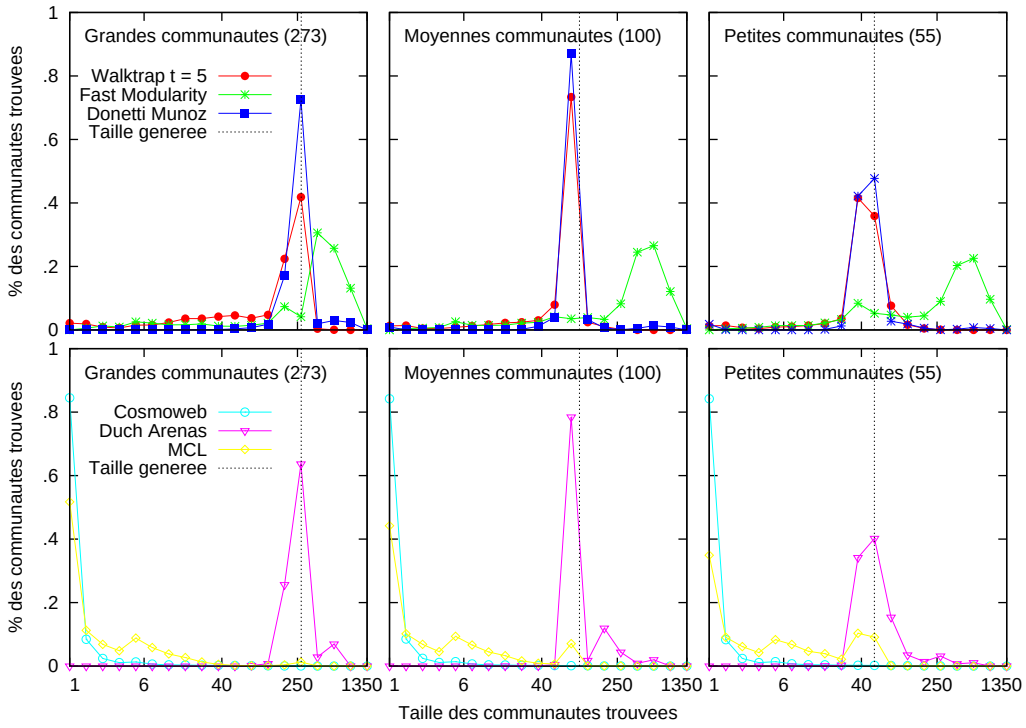

FIG. 5.3 – Distribution des tailles de communautés trouvées sur des graphes de $n = 3000$ sommets. Les trois colonnes correspondent aux différentes tailles de communautés dans la partition de référence (11, 30 et 55 communautés de tailles respectives 273, 100 et 55 sommets).

Si l'on se penche maintenant sur la distribution des tailles des communautés trouvées, nous pouvons observer des comportements différents de la part des approches testées. Étant donné que chaque graphe test généré contient des communautés de tailles identiques, le comportement souhaité d'un algorithme de détection de communautés est de retrouver une distribution des tailles concentrée sur la taille des communautés générées. La Figure 5.3 montre la distribution des tailles des communautés trouvées sur les graphes de $n = 3000$ sommets de notre premier jeu test. Les distributions sont bien sûr séparées suivant le nombre (et donc la taille) des communautés de la partition de référence : rappelons que nous avons choisi, pour chaque taille de graphe, trois tailles de communautés qui correspondent dans les cas de 3000 sommets à des grandes communautés de 273 sommets, des moyennes communautés de 100 sommets ou des petites communautés de 55 sommets.

Les distribution des tailles de graphe montrent clairement que l'approche de Donetti et Muñoz (DM), l'approche de Duch et Arenas (DA) et notre approche (WT5) trouvent bien

des communautés de la taille souhaitée. En effet les distribution des tailles des communautés correspondantes sont bien centrées sur les tailles des communautés qui ont été générées (nous pouvons clairement identifier un pic).

A l'inverse, les trois autres approches testées produisent des partitions dont la distribution des tailles des communauté semble dépendre de manière limitée de la taille réelle de communautés présentes dans le graphe. Nous notons ainsi que l'approche d'optimisation rapide de la modularité (FM) trouve principalement, sur l'ensemble des graphes tests de 3000 sommets, des communautés de grande tailles (400 à 800 sommets) indépendamment du nombre et de la taille des communautés réelles. Les approches Cosmoweb (CW) et dans une moindre mesure (MCL) ont quand à elles tendance à privilégier les très petites communautés, et ce visiblement de manière indépendante des tailles des communautés de la partition de référence. Ces comportements très pénalisant pour la qualité des résultats mesurée par l'indice de Rand, laissent à penser que ces approches possèdent une échelle intrinsèque à laquelle elles détectent des communautés. Ceci explique que ces approches peuvent obtenir de très bons résultats lorsque les communautés à détecter ont des tailles adaptées à leur échelle intrinsèque. A l'opposé lorsque les communautés ne correspondent pas à leur échelle intrinsèque, ces algorithmes risquent de fournir une partition de mauvaise qualité. Ceci peut en partie expliquer les chutes de qualités de approches (FM) et (CW) lorsque la taille des graphes augmente : les tailles de communautés dans le jeu de graphes tests pourraient se trouver plus éloignés des échelles intrinsèques de ces deux approches.

5.2.2 Comparaison sur des graphes hétérogènes

Nous allons maintenant tester les algorithmes sur des graphes plus hétérogènes possédant des communautés de tailles et de densités variables. Cette hétérogénéité des communautés permet de rapprocher le modèle de graphes de conditions plus réalistes et moins favorables que le premier jeu de graphes tests. Ces graphes sont générés selon le même modèle de graphes aléatoires, cependant nous choisissons maintenant des tailles $|\mathcal{C}_i|$ et des degrés internes $d_{in}(\mathcal{C}_i)$ différents pour chaque communauté du graphe. Plutôt que de fixer le nombre de sommet des graphes, nous fixons le nombre de communautés et leur distribution de tailles (la taille moyenne des graphes sera donc égale au nombre de communautés multiplié par la taille moyenne d'une communauté). Nous considérerons encore les 6 mêmes niveaux de modularité $Q_e^M \in \{0.2, 0.3, 0.4, 0.5, 0.6\}$ et les tailles et densités des communautés sont choisies de la façon suivant :

- Le degré interne de chaque partition $d_{in}(\mathcal{C}_i)$ est choisi uniformément dans une plage $[\alpha_{min} \ln(|\mathcal{C}_i|), \alpha_{max} \ln(|\mathcal{C}_i|)]$ avec trois niveau d'hétérogénéité donnés par $(\alpha_{min}, \alpha_{max}) = (5, 7), (4, 8)$ ou $(3, 9)$.
- Les nombres de communautés sont les mêmes ($k = n^\gamma$ avec $\gamma = 0.3$, 0.42 ou 0.5) mais leur taille suit une distribution en loi de puissance selon trois exposants α possibles : $\alpha \in \{2.1, 2.3, 3\}$. Ainsi le nombre de sommets des graphes générés ne sont pas forcément égaux. Les nombres moyens de sommets envisagés sont $n_{moy} = 100$, 300, 1 000, 3 000 (ces nombres sont déterminés par les choix des paramètres des distributions de tailles des communautés).

Les degrés internes sont choisis de telle manière que le degré interne moyen corresponde à $6 \ln(|\mathcal{C}_i|)$ pour chaque cas. Nous pourrons ainsi comparer les résultats sur ce second jeu test aux résultats sur les graphes du premier test homogène ayant cette même densité interne. La largeur de la plage autorisée pour le degré interne $d_{in}(\mathcal{C}_i)$ indique l'hétérogénéité de la densité

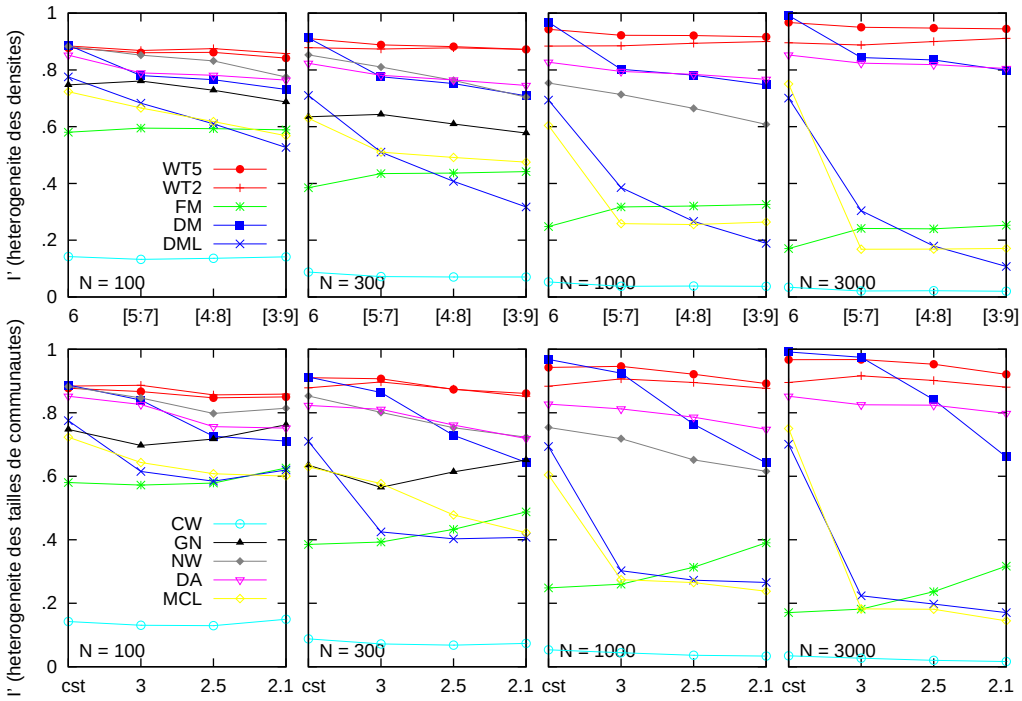

FIG. 5.4 – Influence de l'hétérogénéité des graphes sur la qualité des résultats (mesurée par l'indice de Rand modifié I') pour 4 tailles de graphes ($n = 100, 300, 1000, 3000$). Pour chaque graphique, l'axe des abscisses mesure l'hétérogénéité selon 4 niveaux (décrits dans le texte) allant des cas homogènes (à gauche) aux cas les plus hétérogènes (à droite). **(en haut)** Influence de la densité des communautés donnée par $(\alpha_{min}, \alpha_{max})$. **(en bas)** Influence de de la distribution des tailles des communautés donnée par l'exposant α de la loi de puissance.

des communautés. Ainsi, les paramètres $(\alpha_{min}, \alpha_{max}) = (3, 9)$ produiront des graphes plus hétérogène que les paramètres $(\alpha_{min}, \alpha_{max}) = (5, 7)$.

Une fois le nombre de communautés choisi, les tailles des communautés suivent une loi de puissance dans une plage fixée $[S_{min}, S_{max}]$. Cette plage qui encadre la taille des communautés possibles évite de générer de trop petites ou trop grosses communautés. La probabilité de choisir une communauté de taille $S \in [S_{min}, S_{max}]$ est proportionnelle à $(S + \mu)^{\alpha}$ où μ est déterminé de façon à ce que la taille espérée du graphe généré soit égale à la taille moyenne n_{moy} choisie. Plus l'exposant α de la loi de puissance est proche de 2, plus les tailles des communautés générées seront hétérogènes. Nous avons choisi les bornes S_{min} et S_{max} en fonction de la taille moyenne des communautés $S_{moy} = \frac{n_{moy}}{k}$ (donnée par la taille moyenne n_{moy} du graphe souhaité et le nombre k de communautés). Nous créons alors trois niveaux d'hétérogénéité : pour $\alpha = 3$ nous prenons $S_{min} = 0.75\ S_{moy}$ et $S_{max} = 1.5\ S_{moy}$; pour $\alpha = 2.3$ nous prenons $S_{min} = 0.5\ S_{moy}$ et $S_{max} = 2\ S_{moy}$ et pour $\alpha = 2.1$ non prenons $S_{min} = 0.25\ S_{moy}$ et $S_{max} = 4\ S_{moy}$. Ainsi nous créons des distributions de tailles de communautés modérément hétérogènes pour les paramètres associés à $\alpha = 3$ et des distributions très hétérogènes pour les paramètres associés à $\alpha = 2.1$.

La qualité des résultats des différents algorithmes en fonction des choix de ces paramètres est donnée en Figure 5.4. Nous pouvons clairement distinguer plusieurs comportements différents de ces approches, qui s'adaptent plus ou moins bien aux graphes hétérogènes.

On observe en particulier que notre approche (WT5) et (WT2) n'est pratiquement pas influencée par l'hétérogénéité des communautés. En effet, la qualité des partition obtenues n'est que très peu modifiée par les différents paramètres de densité et de taille de communautés. L'approche de Girvan et Newman (GN) se comporte elle aussi très bien face à des graphes hétérogènes et l'on peut observer une légère dégradation des résultats pour les densités hétérogènes couplée à une légère amélioration des résultats pour les tailles de communautés hétérogènes. L'approche Netwalk (NW) et celle de Duch et Arenas (DA) montrent de légères dégradations des résultats pour les graphes possédant des communautés hétérogènes, mais elles conservent des résultats qui sont globalement équivalents dans tous les cas envisagés. Contrairement à cela, les approches de Donetti et Muñoz (DM) et (DML) ainsi que (MCL) obtiennent par contre des résultats dont la qualité est nettement diminuée pour les graphes présentant des structures de communautés hétérogènes. Et à l'opposé, seule l'approche (FM) obtient des résultats de meilleure qualité pour les graphes hétérogènes.

Nous avons comparé notre approche aux principaux algorithmes de détection de communautés (Girvan et Newman [33, 59], Fast modualrity [16], Donetti et Muñoz [19, 20], Netwalk [88], MCL [79], Duch Arenas [22] et Cosmoweb [9]) sur deux jeux de graphes tests générés aléatoirement. Le premier jeu est composé de graphes possédants des structures de communautés homogènes pour lesquels nous avons fait varier le nombre, la taille et la densité des communautés ainsi que la modularité espérée du graphe. Le second jeu de graphes tests introduit des communautés de densités et de tailles hétérogènes rendant le modèle plus proche de la réalité mais souvent moins favorable pour les algorithmes de détection de communautés.
Notre approche obtient des résultats qui se placent parmi les meilleurs tant du point de vue du temps de calcul (nous avons pu traiter des graphes possédant jusqu'à un million de sommets) que du point de vue de la qualité des partitions obtenues qui ont été comparées aux partition de référence des structures de communautés générées. Nous avons de plus observé que notre approche, contrairement à certaines autres, conserve ses bonnes performances lorsqu'elle est confrontée à des structures de communautés hétérogènes.

5.2.3 Comparaison sur des graphes réels

Nous allons maintenant comparer les différentes approches sur des graphes réels. Il est difficile de juger de la qualité des résultats sur ces graphes car nous ne possédons pas de partition de référence. Nous ne pourrons donc pas utiliser l'indice de Rand pour évaluer la qualité des partitions trouvées. Nous nous contenterons de comparer leur modularité. Comme nous allons le voir dans la section suivante, cette méthode de comparaison est limitée car il est possible que les partitions de meilleures modularités ne correspondent pas aux partitions en communautés les plus pertinentes. Dans le cas de graphes réels, toutefois, il n'y a pas d'alternative satisfaisante.

Nous allons utiliser les graphes réels suivants :
- le réseau du club de karaté étudié par Zachary [87], un petit réseau social qui a largement été utilisé pour tester la plupart des algorithmes de détection de communautés,
- un réseau de rencontres d'équipes de football américain pris en exemple dans [33],
- un réseau d'interactions protéiques étudié dans [46],
- un réseau de collaborations scientifiques calculé sur la base de données d'arXiv [91],
- une carte d'internet fournie par Damien Magoni [40], et
- un graphe du web étudié par [3].

graphe	karate	foot	protein	arxiv	internet	www
n/degré	33/4.55	115/10.7	594/3.64	9377/5.14	67882/8.12	159683/11.6
WT5	0.38/0s	0.60/0s	0.67/0.02s	0.76/4.61s	0.76/1030s	0.91/5770s
WT2	0.38/0s	0.60/0s	0.64/0.01s	0.71/1.08s	0.69/273s	0.84/468s
FM	0.39/0s	0.57/0s	0.71/0s	0.77/1.65s	0.72/483s	0.92/1410s
DM	0.41/0s	0.60/0s	0.59/0.34s	0.66/1460s	–	–
DML	0.41/0s	0.60/0s	0.60/1.37s	0.62/1780s	–	–
CW	-0.05/0s	0.33/0s	0.50/0.02s	0.60/0.65	0.47/6.82s	0.79/21s
GN	0.40/0s	0.60/0.39s	0.70/6.93s	>40000s	–	–
NW	0.40/0.02s	0.60/0.07s	0.60/5.2s	>40000s	–	–
DA	0.41/0s	0.60/0.05s	0.69/1.9s	0.77/14000s	–	–
MCL	0.36/0s	0.60/0.05s	0.66/0.58s	0.73/61.3s	–	–

TAB. 5.1 – Performances sur des graphes réels (modularité / temps d'exécution (en s)). La seconde ligne indique la taille des graphes donnée par leur nombre de sommets n et leur degré moyen.

Nous avons réduit la taille de ces graphes en ne considérant à chaque fois que la plus grande composante connexe et en retirant itérativement tous les sommets de degré 1, qui n'apportent pas d'information importante sur les structures de communautés. Ceci nous a permis d'augmenter le nombre de graphes tests accessibles aux algorithmes les moins rapides.

Les résultats des testes sont reportés dans la Table 5.1. Le graphe du club de karaté a été découpé en 4 communautés par presque toutes les approches. Les différentes partitions trouvées sont très proches et représentent des divisions compatibles avec les deux groupes identifiés par Zachary [87]. De même les résultats des différentes approches pour le graphe des compétitions de football sont globalement les mêmes pour toutes les approches. Les partitions trouvées correspondent aux différentes ligues dans lesquelles les équipes se rencontrent plus fréquemment. Il est cependant difficile de fournir des interprétations pour les autres graphes à cause de leur taille et du fait que les données brutes étaient souvent anonymisées.

Nous pouvons relever un premier point préoccupant sur la validité de la comparaison des performance des algorithmes en utilisant la modularité. En effet, nous observons que les partitions trouvées par les différents algorithmes peuvent obtenir des niveaux de modularité équivalents tout en n'étant que très peu corrélées par l'indice de Rand. Par exemple les trois algorithmes WT5, FM et DA obtiennent des modularités très proches pour le graphe d'arxiv (entre 0.76 et 0.77) mais leurs indices de Rand modifiées I' relatifs indiquent que ces trois partitions sont très différentes : leurs indices sont de 0.33 pour WT5/FM, de 0.25 pour WT5/DA et de 0.18 pour FM/DA. Ceci signifie que plusieurs partitions peuvent avoir des valeurs équivalentes (et élevées) de modularité et représenter des structures de communautés très

différentes. Nous pouvons nous demander si ces partitions différentes sont toutes des partition en communautés pertinentes (ce qui pourrait par exemple s'expliquer par des communautés qui se chevauchent), ou si certaines structures de communautés non-pertinentes peuvent tout de même atteindre des niveaux élevés de modularité. Nous pensons qu'il s'agit d'une problématique importante qui doit être considérée avec attention par les différentes approches de détection de communautés, surtout celles qui ce basent sur une optimisation directe de la modularité [16, 22, 38, 58]. Nous aborderons de nouveau ce problème dans la section suivante.

5.3 Évaluation de la détection multi-échelles

Nous allons évaluer dans cette section les gains apportés par la détection multi-échelles ainsi que par les différentes fonctions de qualité proposées au Chapitre 4. Nous nous baserons d'abord sur le premier jeu de graphes tests introduit à la section précédente pour évaluer les différentes fonctions de qualité. Nous proposerons ensuite d'autres tests pour évaluer la capacité de notre approche à détecter des structures de communautés à différentes échelles. Les comparaisons menées dans cette section se rapportent principalement aux fonctions multi-échelles. Tous les calculs de communautés seront donc effectués avec la même approche de détection de communautés (WT5) et seuls les traitements du dendrogramme pour choisir la partition de sortie changent.

5.3.1 Évaluation des différentes fonctions de qualité

Nous allons baser notre étude sur le premier jeu de graphes tests homogènes que nous avons utilisé pour comparer les différentes approches, cependant nous allons maintenant comparer les performance des différentes fonctions de qualité. Nous ne considérerons que les graphes de 100 à 10 000 sommets mais nous conserverons tous les autres choix de paramètres, c'est à dire 3 tailles de communautés, 5 niveaux de densité des communautés et 5 niveaux de modularité de la partition générée.

Nous allons comparer les différentes façons d'obtenir une partition en communautés à partir de l'approche hiérarchique que nous avons proposée au Chapitre 3. Pour chaque graphe test, les différentes méthodes testées se baseront donc sur le même dendrogramme obtenu en sortie de notre algorithme. La qualité des partitions obtenues sera mesurée par l'indice de Rand modifié I' présenté en début de chapitre.

Nous allons considérer les trois fonctions de qualité introduites en Section 4.2 (Q^M, Q^P et Q^S). Pour chacune d'elles, nous allons envisager une maximisation sur l'ensemble Π des partitions que l'on peut construire à partir des communautés du dendrogramme fourni par l'algorithme de détection de communautés (ces méthodes font intervenir l'algorithme décrit en Section 4.3). Nous allons aussi envisager la maximisation des fonctions de qualités multi-échelles correspondantes (Q^M_α, Q^P_α et Q^S_α) sur l'ensemble des partitions Π (méthode proposée en Section 4.4). Les paramètres d'échelle α retenus seront donnés par la fonction de pertinence $R(\alpha)$ proposée à la Section 4.5. Nous allons de plus intégrer dans la comparaison la méthode classique qui consiste à choisir la partition maximisant la modularité Q^M sur l'ensemble restreint des n partitions obtenue après chaque fusion lors du déroulement de l'algorithme agglomératif. Nous allons donc comparer les méthodes suivantes :

- (CM) Maximisation classique de la modularité (Q^M) sur l'ensemble restreint des n partitions obtenues après chaque fusion de communauté.

- (BM) Maximisation de la modularité (Q^M) sur l'ensemble Π des partitions. Il s'agit de l'approche (WT5) utilisée dans la section précédente.
- (MM) Maximisation de la modularité multi-échelle (Q^M_α) sur l'ensemble Π pour le facteur d'échelle α maximisant $R(\alpha)$.
- (BP) Maximisation de la performance (Q^P) sur l'ensemble Π des partitions.
- (MP) Maximisation de la performance multi-échelle (Q^P_α) sur l'ensemble Π pour le facteur d'échelle α maximisant $R(\alpha)$.
- (BS) Maximisation de la fonction de qualité basée sur la similarité (Q^S) sur l'ensemble Π des partitions.
- (MS) Maximisation de la fonction de qualité multi-échelle basée sur la similarité (Q^S_α) sur l'ensemble Π pour le facteur d'échelle α maximisant $R(\alpha)$.

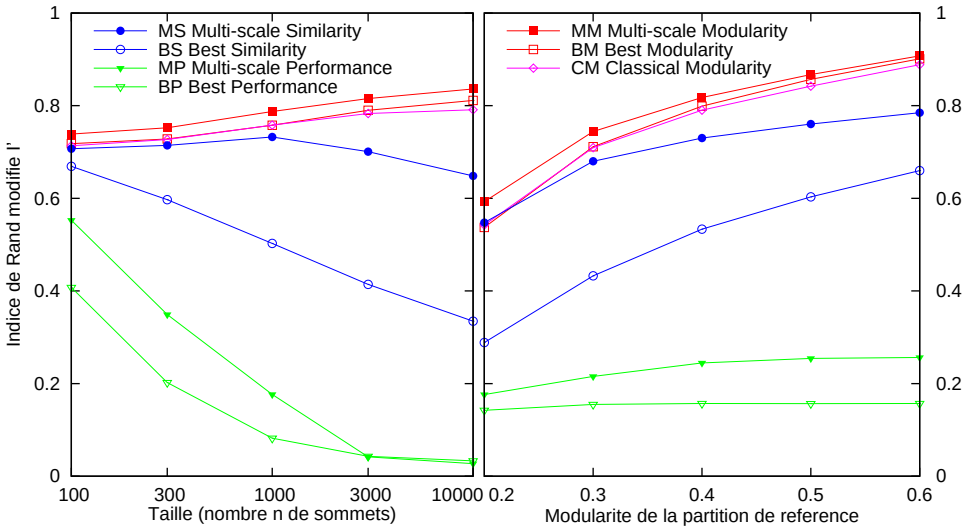

FIG. 5.5 – Performance des différentes méthodes de détermination d'une partition en communautés à partir d'un dendrogramme. L'indice de Rand modifié (I') de la partition obtenue est donné en fonction de la taille des graphes tests (à gauche) et de la modularité Q^M espérée de la partition de référence (à droite).

Les résultats de ces tests, en fonction de la taille des graphes et de la modularité des partitions de référence, sont présentés sur la Figure 5.5. Nous remarquons tout d'abord que, dans le contexte de ce premier jeu de graphes tests, nous pouvons établir un classement sans ambiguïté des fonctions de qualité testées. La modularité permet d'obtenir les meilleures partitions en communautés : en effet les trois méthodes (CM, BM et MM) produisent les partitions les plus proches des partitions de référence. La performance (méthodes BP et MP) est une fonction de qualité peu adaptée à ce type de graphe et conduit à des partitions de moins bonne qualité que les deux autres fonctions de qualité. La performance accorde beaucoup d'importance aux arêtes qui n'existent pas et ceci pénalise probablement cette fonction dans le cas des graphes peu denses que nous avons générés. La fonction de qualité basée sur la similarité (méthodes BS et MS) se place, du point de vue de la qualité des partitions obtenues, entre les deux autres fonctions de qualité. Nous notons que l'utilisation de la version multi-échelles

(MS) permet d'obtenir des résultats de bonne qualité, proches de ceux obtenus grâce à la modularité. Cependant l'utilisation directe de cette fonction (BS) ne produit pas de résultats satisfaisants. Dans le cadre de ces premiers tests, la méthode classique de maximisation de la modularité (CM) n'est que très légèrement améliorée par l'approche (BM) d'optimisation de la modularité sur l'ensemble des partitions Π. Ceci montre que le déroulement de notre algorithme pour ces graphes tests est compatible avec la notion de communautés définie par la modularité.

Nous observons que l'utilisation des fonctions de qualité multi-échelles couplées à la détermination du meilleur facteur d'échelle α par la fonction de pertinence $R(\alpha)$ produit pour les trois fonctions de qualité testées une amélioration des résultats. En effet les qualités des partitions trouvées par les méthodes MM, MP et MS sont respectivement supérieures à celle obtenue par les méthodes mono-échelle correspondantes BM, BP et BS. Ceci montre l'utilité de l'approche multi-échelles que nous avons proposée. Les prochains tests de cette section seront plus ciblés et appuieront cette première constatation.

> Les comparaisons et tests effectués avec les 3 fonctions de qualité présentés en Section 4.2 montrent que la modularité Q^M obtient les meilleurs résultats et que la performance Q^P semble inadaptée aux faibles densités des grands graphes de terrain. La fonction de qualité basée sur la similarité Q^S obtient, lorsqu'elle est utilisée dans sa version multi-échelle (Q^S_α), de bon résultats, comparables à ceux de la modularité.

5.3.2 L'apport des fonctions de qualité multi-échelles

Nous allons maintenant nous concentrer sur l'apport des fonctions multi-échelles par rapport aux fonctions de qualité classiques. Nous allons tout d'abord montrer que ces nouvelles fonctions permettent de s'adapter à différentes tailles de communautés alors que les fonctions de qualité classiques ne détectent souvent les communautés que pour une échelle intrinsèque. Nous considérons en premier lieu un ensemble de graphes simples présentant des structures de communautés de tailles variées. Ce troisième jeu de graphes tests est composé de graphes homogènes de $n = 1000$ sommets possédant k communautés de même taille et de degré interne moyen $d_{in} = 3$. La modularité moyenne de la partition de référence est fixée à $Q^M_e = 0.3$. Les graphes diffèrent par leur nombre k de communautés allant de $k = 2$ (grandes communautés, échelle macroscopique) à $k = 100$ (petites communautés, échelle microscopique). Nous avons comparé les deux approches multi-échelles basées sur la modularité (MM) et sur la similarité (MS) aux autres approches utilisant la modularité (BM et CM) et nous n'avons pas intégré les résultats moins significatifs des approches (MP, BP et BS). La Figure 5.6 donne l'indice de Rand modifié I' et la modularité Q^M des partitions obtenues en fonction du nombre de communautés dans le graphe.

Les résultats montrent que les deux approches utilisant des fonctions multi-échelles (MM et MS) permettent de trouver une partition en communautés très proche de la structure de référence. Nous notons au passage que la fonction de qualité basée sur la similarité (MS) n'est visiblement pas adaptée pour détecter les très grosses communautés, comme en témoigne le faible résultat dans les cas où le graphe comporte deux ou trois communautés. Les approches basées sur la modularité non multi-échelle (BM et CM) obtiennent de bons résultats pour des

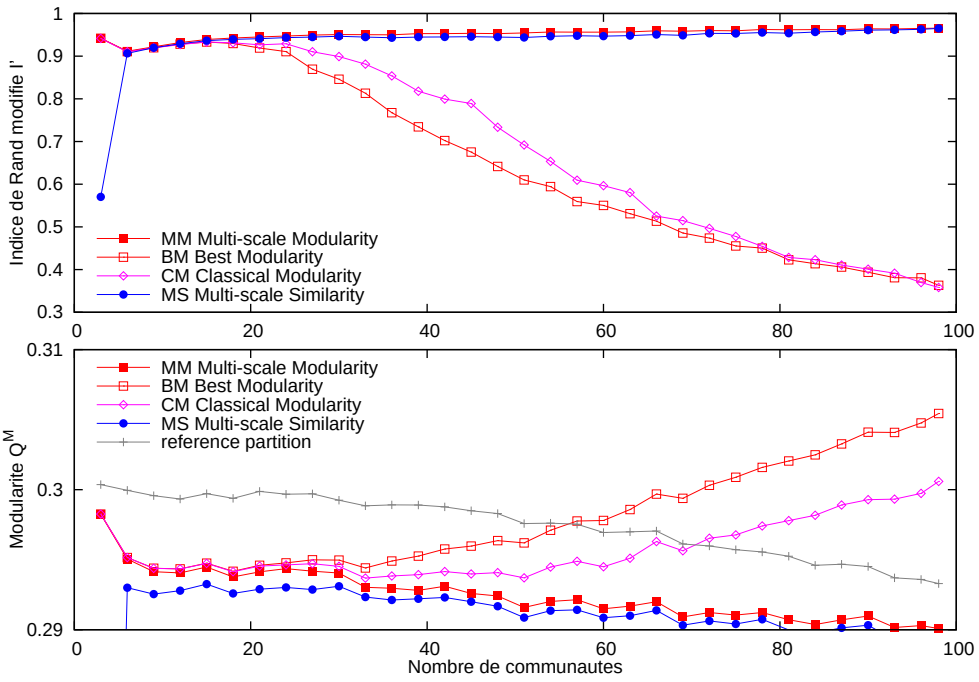

FIG. 5.6 – Influence de l'échelle des structures de communautés donnée par le nombre de communautés (de 2 à 100 communautés en abscisse) sur des graphes de $n = 1000$ sommets. Haut : qualité des partitions trouvées mesurée par l'indice de Rand modifié. Bas : modularité des partitions trouvées et de la partition de référence.

nombres de communautés entre 2 et 20, puis la qualité diminue. Ceci montre que la modularité est une fonction de qualité qui possède une échelle intrinsèque : elle permet de détecter les communautés à cette échelle mais a plus de mal à détecter les plus petites communautés. Les fonctions de qualité multi-échelles que nous avons proposées, et en particulier la modularité multi-échelles, permettent de répondre à cette limitation et s'adaptent naturellement à toutes les tailles de communautés.

Si l'on observe la modularité des partitions trouvées par les différentes approches on constate naturellement que les approches qui maximisent la modularité (BM et CM) obtiennent une meilleure modularité. Nous constatons par ailleurs que cette modularité obtenue dépasse la modularité de la partition de référence pour les partition ayant des petites communautés. Ceci montre que l'optimisation en soi d'une fonction de qualité ne conduit pas forcément à la partition souhaitée. Dans le cas de nos graphes tests, la partition de référence ne possède pas forcément la meilleure modularité et des partitions à des échelles plus macroscopiques peuvent obtenir des niveaux de modularité plus élevés. Ceci soulève un problème important de méthodologie dans la détection de communautés. En effet de nombreuses approches se contentent d'optimiser la modularité et comparent les performances de deux algorithmes grâce aux valeurs de la modularité obtenues. Nous montrons par cette analyse expérimentale que cette problématique est loin d'être aussi simple et nous pensons que les fonctions de qualité multi-échelles que nous proposons permettent de mieux capturer la notion de communauté et d'améliorer la méthodologie de détection de communautés.

Nous venons de voir que les fonctions de qualité multi-échelles permettent de s'adapter à la taille des communautés présentes dans le graphe. Nous allons maintenant étudier leurs comportement vis à vis de structures de communautés à différentes échelles dans un même graphe. Nous avons pour cela généré un dernier jeu de graphes tests possédant $n = 1000$ sommets et deux niveaux de communautés imbriquées : l'ensemble des sommets est d'abord divisé en 10 grandes communautés qui sont à leur tour divisées en 10 petites communautés. Ceci définit une partition à l'échelle microscopique de 100 communautés de 10 sommets et une partition à l'échelle macroscopique de 10 communautés de 100 sommets. Nous définissons 3 densités données par le choix de 3 degrés moyens : d_{in}^{micro} le degré à l'intérieur des petites communautés, d_{in}^{macro} le degré entre les sommets d'un même communauté de 100 sommets et d_{out} le degré externe entre les grandes communautés. Nous avons généré des graphes selon toutes les combinaisons de ces trois paramètres en choisissant chaque degré entre 2 et 6 (soit 5 possibilités pour chacun des 3 degrés considérés).

Nous avons testé les deux approches multi-échelles (MM et MS) sur ces graphes en considérant en sortie les deux meilleures partitions selon la fonction de pertinence $R(\alpha)$. En effet cette fonction de pertinence montre dans ces cas plusieurs maxima locaux et nous avons retenu les deux facteurs d'échelles α correspondant aux deux meilleurs maxima locaux de $R(\alpha)$. La partition obtenue pour le facteur d'échelle le plus élevé est comparée à la partition de référence macroscopique de 10 communautés et la partition obtenue pour le plus faible facteur d'échelle est comparée à la partition de référence microscopique de 100 communautés. Nous obtenons donc pour chaque graphe deux indices de Rand modifiés I' donnant la qualité des partitions selon les deux échelles de structures de communautés générées. Nous avons de plus intégré dans le test la méthode (BM) qui ne permet d'obtenir qu'une seule partition à une seule échelle. Notons qu'il est donc normal que cette méthode ne puisse pas obtenir à la fois de bon résultats pour les échelles microscopiques et macroscopiques.

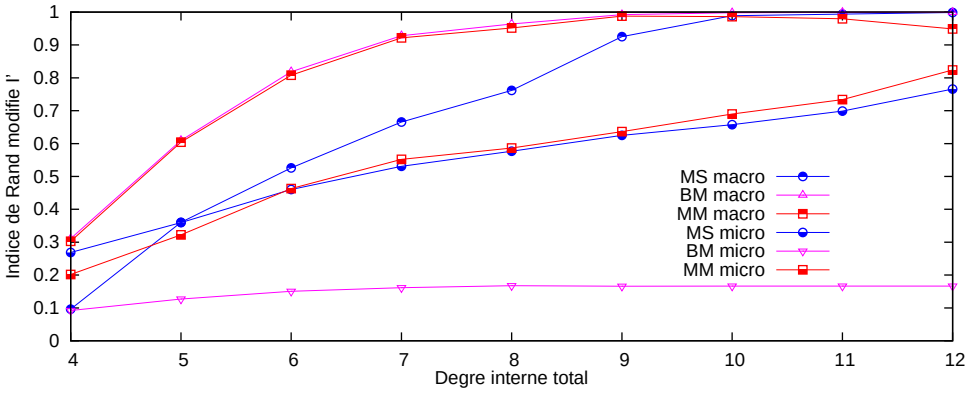

FIG. 5.7 – Performance des approches multi-échelles sur des graphes possédant deux échelles des structures de communautés composées de $n = 1000$ sommets divisés en 10 macro communautés elles mêmes divisées en 10 micro communautés. Les deux indices de Rand modifiés I' par rapport aux deux partitions de références (communautés macro et micro) sont donnés en fonction de degré interne total.

Les résultats de ce test sont présentés en Figure 5.7. Ils donnent les deux indices de Rand modifiés I' (échelle micro et macro) en fonction du degré interne total $d_{in} = d_{in}^{micro} + d_{in}^{macro}$.

Nous observons d'abord que l'approche (BM) d'optimisation mono-échelle de la modularité détecte uniquement la structure de communauté macroscopique au détriment de la structure microscopique. Les deux approches multi-échelles permettent bien de détecter deux partitions différentes qui correspondent aux deux échelles de communautés générées. Dans les deux cas, la structure macroscopique est mieux détectée que la structure microscopique. Les deux approches multi-échelles (MM et MS) obtiennent des résultats similaires pour la détection des petites communautés et l'approche MM obtient de meilleurs résultats que l'approche MS pour la détection des grandes communautés. Ceci illustre bien la capacité des fonctions de qualité multi-échelles à détecter dans un même graphe des structures de communautés à plusieurs échelles.

L'utilisation de fonctions de qualité multi-échelles (Q_α^M et Q_α^S), couplée à la fonction de pertinence $R(\alpha)$, améliore la qualité des résultats. Ces fonctions permettent de détecter des structures de communautés à toutes les échelles, contrairement aux versions non multi-échelles. Nous avons par exemple constaté que la modularité (Q^M) possède une échelle intrinsèque. Ceci soulève un problème important de méthodologie vis à vis des approches qui se contentent de maximiser la modularité. Nous avons en effet exhibé des cas où la meilleure modularité ne correspond pas aux structures de communautés générées. Ce problème peut être résolu par la modularité multi-échelles Q_α^M que nous avons proposée.
Nous avons finalement validé la capacité des fonctions de qualité multi-échelles à détecter plusieurs échelles de structures de communautés dans un même graphe. Nous nous concentrons pour cela sur les facteurs d'échelles α correspondants aux différents maxima locaux de la fonction de pertinence $R(\alpha)$.

5.4 Influence de la longueur des marches aléatoires

Nous allons maintenant étudier l'effet du choix de la longueur t des marches aléatoires. Cette analyse expérimentale illustre et complète les discussions sur le choix de la longueur des marches de la Section 3.1.4. Nous allons étudier le comportement de notre algorithme de détection de communautés lorsque la longueur des marches aléatoires s'allonge et sa capacité à détecter les différentes échelles de communautés en fonction de la longueur des marches.

Nous allons pour cela utiliser des graphes possédant plusieurs échelles de structures de communautés, similaires aux graphes utilisés dans le test précédent. Les graphes utilisés ont 3 niveaux de communautés : les graphes sont divisés en deux grandes communautés qui sont chacune composées de 2 moyennes communautés et sont à leur tour divisées en 2 petites communautés. La Figure 5.8 illustre cette structure. Les graphes considérées comportent $n = 256$ sommets et sont donc divisés en 8 communautés de 32 sommets à l'échelle la plus microscopique. Un ensemble de 1000 graphes est généré selon les 4 degrés moyens suivants $d_{in}^{micro} = 12, d_{in}^{moy} = 6, d_{in}^{macro} = 8, d_{out} = 12$. Nous comparons alors les partitions trouvées pour les différentes échelles de communautés aux partitions de référence correspondant aux trois échelles de communautés générées.

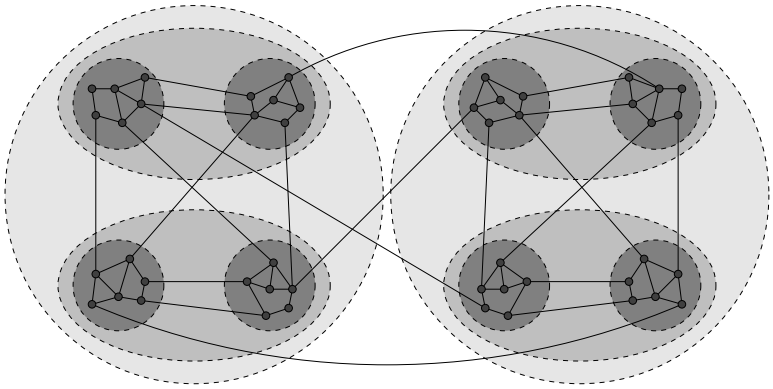

FIG. 5.8 – Illustration de la structure de communautés à plusieurs échelles utilisée. Les graphes générés sont composés de $n = 256$ sommets, 2 grandes communautés de taille 128, 4 moyennes communautés de taille 64 et 8 petites communautés de taille 32.

Les résultats sont donnés en Figure 5.9. Ils montrent la capacité de notre algorithme à détecter les trois différentes échelles de communautés en fonction de la longueur t des marches aléatoires. Ces tests ayant été effectués antérieurement aux autres tests, nous n'avons pas utilisé la métrique de l'indice de Rand mais le taux de sommets correctement identifiés pour comparer les partitions trouvées aux partitions de références. Cette mesure, un peu moins précise que l'indice de Rand donne cependant des résultats suffisants pour identifier les comportement de notre algorithme.

Nous observons que dans tous les cas lorsque la longueur des marches devient importante, la qualité des résultats diminue. Ceci est expliqué par le fait que les marches aléatoires atteignent rapidement leur état stationnaire limite. De même les très courtes marches (de longueur $t \leq 3$) ne donnent pas les meilleurs résultats car elles n'ont pas le temps de recueillir suffisamment d'information autour de chaque sommet. Nous remarquons que les meilleurs résultats sont, pour les trois échelles de communautés, obtenus sur un palier (plus ou moins large) des longueurs de marches. La largeur des paliers est d'autant plus étroite que les communautés à détecter sont petites. Ceci confirme l'intuition que pour détecter des petites communautés, il faut utiliser des marches de longueurs plus courtes car elles atteignent plus rapidement des distributions stationnaires locales correspondant à une diffusion dans une communauté.

La longueur optimale des marches dépend donc de la taille de communautés à détecter. Cependant nous observons que les palier sur lesquels les performances sont optimales sont larges et se chevauchent grandement. Nous estimons qu'en pratique une choix de longueur des marches aléatoires compris entre 4 et 10 permet dans la plupart des cas d'obtenir des performances proches des performances optimales. L'implémentation de notre approche utilise d'ailleurs par défaut une longueur $t = 5$ qui se révèle être un bon compromis entre la qualité des résultats et le temps de calcul pour la majorité des graphes.

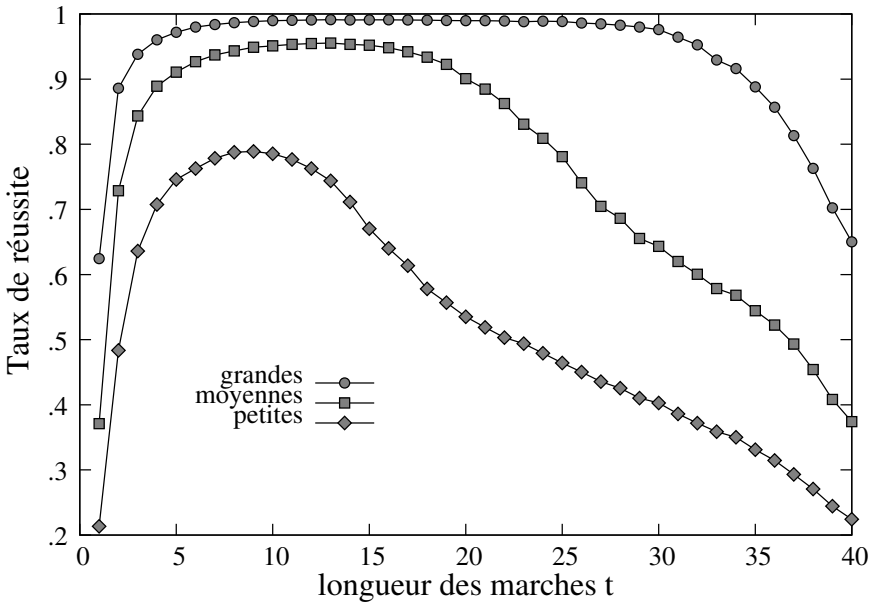

FIG. 5.9 – Qualité des partitions trouvées à différentes échelles en fonction de la longueur t des marches aléatoire. Les graphes tests présentent trois niveaux de communautés (petites, moyennes et grandes) comme illustré en Figure 5.8.

La meilleure longueur des marches aléatoires dépend de la taille des communautés à détecter. Plus les communautés à détecter sont petites et plus les longueurs t à choisir doivent être faibles.
Expérimentalement, un choix de longueur t compris entre 4 et 10 conduit dans la plupart des cas à de très bons résultats. L'implémentation de notre approche utilise par défaut des longueurs $t = 5$ qui se révèlent être un bon compromis entre la qualité des résultats et le temps de calcul pour la majorité des graphes.

Chapitre 6

Conclusion

Nous nous sommes concentrés dans cette thèse sur la problématique de détection de communautés dans les grands graphes de terrain, c'est à dire de zones localement denses et faiblement interconnectées dans des graphes globalement peu denses. La contribution principale de cette thèse réside dans la proposition de deux algorithmes complémentaires : un algorithme de détection de communautés basé sur les marches aléatoires et un algorithme de détermination multi-échelles des partitions les plus pertinentes à partir d'un dendrogramme. Ces algorithmes et leur étude formelle sont complétés par une étude expérimentale approfondie permettant de les comparer avec l'existant.

Nous avons proposé au Chapitre 3 un nouvel algorithme de détection de communautés basé sur les marches aléatoires dans un graphe (une étude préliminaire des marches aléatoires a été proposée au Chapitre 2). Cet algorithme utilise une distance r qui mesure la similarité structurelle entre deux sommets ou deux communautés. Cette distance compare les probabilités de transition par des marches aléatoires courtes : elle se base sur l'intuition que les marches aléatoires tendent à être piégées dans les zones denses du graphe (les communautés). Nous avons relié notre distance r aux propriétés spectrales de la matrice de transition : ainsi nous tirons profit de propriétés spectrales reconnues en termes de clustering et de détection de communautés tout en s'affranchissant du calcul pénalisant des valeurs et vecteurs propres.

Nous utilisons alors une approche de clustering hiérarchique (selon la méthode de Ward) pour fusionner successivement les sommets en communautés. Nous produisons ainsi une structure hiérarchique de communautés appelée dendrogramme. La complexité en temps de notre approche est en $\mathcal{O}(mnH)$ (ou H est la hauteur du dendrogramme trouvé) et nécessite un espace en $\mathcal{O}(n^2)$ mais une gestion dynamique de la mémoire permet de réduire l'espace utilisé en augmentant modérément le temps d'exécution.

Nous avons, au Chapitre 5, comparé expérimentalement notre algorithme aux principales autres approches de détection de communautés. Cette comparaison est la première comparaison directe de plusieurs algorithmes de détection de communautés sur un ensemble important de graphes tests. Nous avons constaté que les performances de notre approche se trouvent parmi les meilleures tant du point de vue de la qualité des résultats que du point de vue de la rapidité d'exécution. Par exemple, il a pu traiter des graphes allant jusqu'à un million de sommets. L'algorithme que nous proposons ainsi que les comparaisons expérimentales effectuées ont donné lieu aux publications [64, 65, 66].

Le second algorithme, proposé au Chapitre 4, est complémentaire du premier. Il permet de

déterminer les partitions en communautés les plus pertinentes à partir d'un dendrogramme. Pour cela il optimise une fonction de qualité sur l'ensemble des partitions que l'on peut créer à partir de ce dendogramme. Cet algorithme s'adapte à n'importe quelle approche de détection de communautés qui produit une structure hiérarchique de communautés, dont la nôtre.

Nous proposons d'abord une méthode efficace d'optimisation pour la classe des fonctions de qualités additives, qui regroupe les fonctions les plus courantes, dont la modularité. Nous généralisons ensuite ces fonctions de qualité en proposant des versions multi-échelles qui permettent de détecter des communautés à plusieurs échelles. Nous proposons un algorithme d'optimisation de ces fonctions de qualité multi-échelles couplé à une méthode qui permet de déterminer les échelles les plus pertinente auxquelles apparaissent les structures de communautés. La recherche des partitions s'adapte ainsi à la taille des communautés et permet de détecter, dans un même graphe, plusieurs structures présentes à plusieurs échelles. Ces méthodes d'optimisation possèdent une faible complexité et peuvent donc être intégrées à la suite de la plupart des méthodes de détection de communautés sans augmenter leur complexité globale.

Nous avons comparé, au Chapitre 5, les différentes fonctions de qualité et les fonction multi-échelles introduites. Ces expériences montrent que l'approche multi-échelle améliore dans tous les cas la qualité des partitions trouvées. Elles permettent de plus de détecter des communautés à toutes les échelles alors que les fonctions de qualité classiques opèrent souvent à des échelles intrinsèques. Nous avons en particulier remarqué que la modularité privilégie de manière intrinsèque les grandes communautés et détecte difficilement les petites communautés. La modularité multi-échelle que nous avons proposée en généralisation permet de résoudre ce problème. L'algorithme d'optimisation ainsi que les études expérimentales effectuées ont fait l'objet d'une publication [63].

L'introduction des fonctions de qualité dans la problématique de la détection de communautés a permis de produire les meilleurs algorithmes actuels qui traitent cette problématique. Nous avons cependant constaté au cours de cette thèse que l'optimisation en soi d'une fonction de qualité, et notamment de la modularité, ne conduit pas forcément au découpage en communautés le plus pertinent. Nous avons proposé dans cette thèse des fonctions de qualité multi-échelles qui permettent de résoudre un problème de certaines fonctions de qualité qui privilégient intrinsèquement certaines tailles de communautés. Il reste cependant de nombreux points sur lesquels le comportement des différentes fonctions de qualités peuvent être critiquables suivant le contexte d'utilisation. Nous pensons donc que les prochaines avancées majeures dans le domaine de la détection de communautés passeront soit par l'introduction de meilleures fonctions de qualité, soit par une nouvelle formalisation de la notion de communauté différente de la formalisation apportée par les fonctions de qualité. Des verrous difficiles semblent toutefois présents dans ces deux directions.

Nous souhaitons aussi rappeler aux utilisateurs des méthodes de détection de communautés que très souvent la qualité des résultats dépend moins de la méthode de détection employée que de la manière de construire le graphe. En effet il est souvent possible de modéliser de plusieurs façons différentes un même phénomène, la liberté résidant principalement dans le choix des sommets liés et du poids de chaque lien. Deux graphes sur le même ensemble de sommets peuvent, suivant le poids et la signification des liens, modéliser des phénomènes totalement différents et conduire à des communautés totalement différentes. Dans ce contexte, nous n'avons traité que des graphes non-orientés et pondérés par des poids positifs mais il existe d'autres types de graphes qui permettent de modéliser d'autres comportements. La

prise en compte d'autres modèles de graphes permettrait de traiter de nouveaux phénomènes jusque là mal gérés par les algorithmes actuels de détection de communautés. Nous pensons en particulier aux pistes de recherche concernant les graphes orientés, les graphes ayant plusieurs types de liens (par exemple des liens sociaux peuvent être familiaux, professionnels ou amicaux) et les graphes possédant des poids négatifs (pouvant ainsi coder une attraction ou une répulsion entre sommets). L'essentiel reste à faire dans ces directions.

Une perspective importante de la détection de communautés est aussi la prise en compte des possibilités de chevauchement des communautés. Il n'existe pour le moment que très peu d'approches qui traitent ce sujet [71, 61]. Il n'existe pas non plus de formalisme qui permette de définir clairement la notion de communautés qui se recouvrent, et la notion de fonction de qualité ne peut pas s'appliquer directement. Les communautés qui se chevauchent correspondent cependant à une réalité des grands graphes de terrain qui n'est absolument pas traitée par la majorité des approches, dont la nôtre. Il s'agit donc d'un axe de recherche quasiment vierge qui correspond à une attente importante.

La visualisation des graphes recueille, pour sa part, une attention beaucoup plus importante mais n'a toujours pas (et n'aura sans doute jamais) de solution générale satisfaisante. Nous pensons que les structures hiérarchiques de communautés pourraient être utilisées, comme proposé dans [6], dans des methodes de visualisation multi-échelles des grands graphes de terrain. Nous visualiserions ainsi la structure du graphe à différentes échelles lorsque le nombre de sommets est trop important pour être dessiné directement. Notons de plus que la distance r qui mesure une similarité entre sommets pourrait aussi être utilisée pour le placement des sommets et des communautés. La visualisation des grands graphes de terrain est selon nous une application directe de nos travaux méritant concrétisation.

Afin d'encourager l'utilisation des algorithmes développés dans cette thèse, nous mettons à disposition des implémentations optimisées qui sont disponibles sur le Web [89]. Notre approche de détection de communautés à déjà donné lieu à des applications, notamment dans le cadre du projet Autograph [90]. Elle est à la base d'un outil de visualisation et de navigation dans les groupes thématiques du site de partage de photos flickr.com. Les sommets du graphe utilisé sont les groupes flickr et les liens entre eux sont pondérés par le nombre de membres communs aux deux groupes. Cet outil, qui est encore à l'état de prototype expérimental, a été initié dans le cadre du projet Autograph lors d'un stage d'ingénieur à France Télécom R&D. Les premiers résultats sont prometteurs et nous laissent penser que l'utilisation de la détection de communautés pour la navigation et la visualisation de données complexes est un axe de recherche important ayant de nombreuses retombée applicatives potentielles. Notre algorithme a aussi été utilisé pour analyser la structure des collaborations entre contributeurs de l'encyclopédie en ligne Wikipédia dans le cadre du projet Autograph.

Bibliographie

[1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1) :47, 2002.

[2] R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance in complex networks. *Nature*, 406 :378–382, 2000.

[3] Réka Albert, Jeong Hawoong, and Barabási Albert-László. Diameter of the world wide web. *Nature*, 401 :130, 1999.

[4] M. S. Aldenderfer and R. K. Blashfield. *Cluster Analysis*. Number 07-044 in Sage University Paper Series on Quantitative Applications in the Social Sciences. Sage, Beverly Hills, 1984.

[5] D. Aldous and J. A. Fill. *Reversible Markov Chains and Random Walks on Graphs*. Ouvrage à paraître, http ://www.stat.berkeley.edu/users/aldous/RWG/book.html.

[6] David Auber, Yves Chiricota, Fabien Jourdan, and Guy Melançon. Multiscale visualization of small world networks. In *Proceedings of the 9th IEEE Symposium on Information Visualization (InfoVis 2003)*, page 10, Seattle, USA, 2003. IEEE Computer Society.

[7] James P. Bagrow and Erik M. Bollt. Local method for detecting communities. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 72(4) :046108, 2005.

[8] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286 :509, 1999.

[9] T. Bennouas, M. Bouklit, and F. de Montgolfier. Un modèle gravitationnel du web. In *5ème Rencontres Francophones sur les aspects Algorithmiques des Télécommunications (Algotel)*, Banyuls (France), 2003.

[10] Vincent Blondel and Pierre Sennelart. Automatic extraction of synonyms in a dictionary. In *Prococeeding of the SIAM Workshop on Text Mining*, 2002.

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

[12] Ulrik Brandes and Thomas Erlebach, editors. *Network Analysis : Methodological Foundations*, volume 3418 of *Lecture Notes in Computer Science*. Springer, 2005.

[13] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7) :107–117, 1998.

[14] D.S. Callaway, M.E.J Newman, S.H. Strogatz, and D.J. Watts. Network robustness and fragility : percolation on random graphs. *Physics Review Letters*, 85 :5468–5471, 2000.

[15] Aaron Clauset. Finding local community structure in networks. *Physical Review E*, 72 :026132, 2005.

[16] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6) :066111, 2004.

[17] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9 :251–280, 1990.

[18] Luciano da Fontoura Costa. Hub-based community finding. *arXiv :cond-mat/0405022*, 2004.

[19] L. Donetti and M. A. Muñoz. Detecting network communities : a new systematic and efficient algorithm. *Journal of Statistical Mechanics*, 2004(10) :10012, 2004.

[20] L. Donetti and M. A. Muñoz. Improved spectral algorithm for the detection of network communities. In *Modeling cooperative behavior in the social sciences*, volume 779, pages 104–107, 2005.

[21] S.N. Dorogovtsev and J.F.F. Mendes. *Evolution of Networks : From Biological Nets to the Internet and WWW*. Oxford University Press, Oxford, 2003.

[22] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 72(2) :027104, 2005.

[23] Holger Ebel, Lutz-Ingo Mielsch, and Stefan Bornholdt. Scale-free topology of e-mail networks. *Physical Review E*, 66, 2002.

[24] Paul Erdős and Alfréd Renyi. On random graphs i. *Publicationes Mathematicae*, 6 :290, 1959.

[25] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. Hodder Arnold, London, 4^{th} edition, 2001.

[26] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Proceedings og the internatonal ACM conference SIGCOMM*, pages 251–262, 1999.

[27] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23 :298–305, 1973.

[28] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-organization and identification of web communities. *Computer*, 35(3) :66–71, 2002.

[29] Santo Fortunato, Vito Latora, and Massimo Marchiori. Method to find community structures based on information centrality. *Physical Review E*, 70(5) :056104, 2004.

[30] B. Gaume. Balades aléatoires dans les petits mondes lexicaux. *I3 Information Interaction Intelligence*, 4(2), 2004.

[31] B. Gaveau, A. Lesne, and L. S. Schulman. Spectral signatures of hierarchical relaxation. *Physics Letters A*, 258(4-6) :222–228, July 1999.

[32] Edgar N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4) :1141–1144, 1959.

[33] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99(12) :7821–7826, 2002.

[34] Jean-Loup Guillaume, Stevens Le Blond, and Matthieu Latapy. Statistical analysis of a p2p query graph based on degrees and their time-evolution. In *Lecture Notes in Computer Sciences (LNCS), proceedings of the 6-th International Workshop on Distributed Computing (IWDC)*, 2004.

[35] Jean-Loup Guillaume and Matthieu Latapy. Complex network metrology. *Complex Systems*, 16 :83–94, 2005.

[36] Jean-Loup Guillaume and Matthieu Latapy. Relevance of massively distributed explorations of the internet topology : Simulation results. In *Proceedings of the 24-th IEEE international conference INFOCOM*, 2005.

[37] Jean-Loup Guillaume and Matthieu Latapy. Bipartite graphs as models of complex networks. *Physica A*, 371 :795–813, 2006.

[38] Roger Guimera and Luis A. Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433 :895–900, 2005.

[39] David Harel and Yehuda Koren. On clustering using random walks. In *Proceedings of the 21st Foundations of Software Technology and Theoretical Computer Science (FSTTCS'01), LNCS 2245*, pages 18–41, 2001.

[40] Mickaël Hoerdt and Damien Magoni. Completeness of the internet core topology collected by a fast mapping software. In *Proceedings of the 11th International Conference on Software, Telecommunications and Computer Networks*, pages 257–261, Split, Croatia, October 2003.

[41] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2 :193–218, 1985.

[42] Ramon Ferrer i Cancho and Ricard V. Solé. The small-world of human language. Technical report, Santa Fe Working paper 01-03-016, 2001.

[43] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering : a review. *ACM Computing Surveys*, 31(3) :264–323, 1999.

[44] M. Jambu and M.-O. Lebeaux. *Cluster analysis and data analysis*. North Holland Publishing, 1983.

[45] H. Jeong, B. Tombor, R. Albert, Z. Oltvai, and A. Barabasi. The large-scale organization of metabolic networks. *Nature, 407, 651*, 2000.

[46] Hawoong Jeong, Sean Mason, Albert-László Barabási, and Zoltán N. Oltvai. Centrality and lethality of protein networks. *Nature*, 411 :41–42, 2001.

[47] R. Kannan, S. Vempala, and A. Veta. On clusterings : good, bad and spectral. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS'00)*, page 367, Washington, DC, USA, 2000. IEEE Computer Society.

[48] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2) :291–308, 1970.

[49] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The Web as a graph : Measurements, models, and methods. In T. Asano, H. Imai, D. T. Lee, S. Nakano, and T. Tokuyama, editors, *Proc. 5th Annual Int. Conf. Computing and Combinatorics, COCOON*, number 1627. Springer-Verlag, 1999.

[50] Jon Kleinberg. Navigation in a small world. *Nature*, 406 :845, 2000.

[51] Stéphane Lafon and Ann B. Lee. Diffusion maps and coarse-graining : A unified framework for dimensionality reduction, graph partitioning and data set parameterization. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 28(9) :1393–1403, 2006.

[52] Matthieu Latapy. Theory and practice of triangle problems in very large (sparse (power-law)) graphs. *article soumis à pubication*.

[53] L. Lovász. Random walks on graphs : a survey. In *Combinatorics, Paul Erdős is eighty, Vol. 2 (Keszthely, 1993)*, volume 2 of *Bolyai Soc. Math. Stud.*, pages 353–397. János Bolyai Math. Soc., Budapest, 1996.

[54] A. Mahdian, H. Khalili, E. Nourbakhsh, and M. Ghodsi. Web graph compression by edge elimination. In *Proceedings of the Data Compression Conference (DCC'06)*, page 459. IEEE Computer Society, 2006.

[55] Mike Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 6 :161–180, 1995.

[56] Boaz Nadler, Stéphane Lafon, Ronald R. Coifman, and Ioannis G. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems (NIPS'05)*, volume 18, 2005.

[57] M. E. J. Newman. The structure and function of complex networks. *SIAM REVIEW*, 45 :167, 2003.

[58] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6) :066133, 2004.

[59] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2) :026113, 2004.

[60] M.E.J. Newman. Scientific collaboration networks : Ii. shortest paths, weighted networks, and centrality. *Phys. Rev. E*, 64, 2001.

[61] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435 :814–818, 2005.

[62] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physcs Review Letters*, 86 :3200–3203, 2001.

[63] Pascal Pons. Post-processing hierarchical community structures : Quality improvements and multi-scale view. *article soumis à publication.*

[64] Pascal Pons. Détection de structures de communautés dans les grands réseaux d'interactions. In *actes des septièmes Rencontres Francophones sur les aspects Algorithmiques des Télécommunications (AlgoTel'05)*, Giens, France, 2005.

[65] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Proceedings of the 20th International Symposium on Computer and Information Sciences (ISCIS'05)*, volume 3733 of *Lecture Notes in Computer Science*, pages 284–293, Istanbul, Turkey, October 2005. Springer.

[66] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2) :191–218, 2006.

[67] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.*, 11(3) :430–452, 1990.

[68] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *PNAS*, 101(9) :2658–2663, 2004.

[69] W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66 :846–850, 1971.

[70] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási. Hierarchical Organization of Modularity in Metabolic Networks. *Science*, 297(5586) :1551–1555, 2002.

[71] Jörg Reichardt and Stefan Bornholdt. Detecting fuzzy community structures in complex networks with a Potts model. *Physical Review Letters*, 93 :218701, 2004.

[72] Mauricio G. C. Resende. Detecting dense subgraphs in massive graphs. In *17th International Symposium on Mathematical Programming*, 2000.

[73] L. S. Schulman and B. Gaveau. Coarse grains : The emergence of space and order. *Foundations of Physics*, 31(4) :713–731, April 2001.

[74] Robert Sedgewick and Philippe Flajolet. *An Introduction to the Analysis of Algorithms.* Addison-Wesley Publishing Company, 1996.

[75] Roger B. Sidje and William J. Stewart. A numerical study of large sparse matrix exponentials arising in markov chains. *Computational Statistics & Data Analysis*, 29(3) :354–368, January 1999.

[76] I. Simonsen, K. Astrup Eriksen, S. Maslov, and K. Sneppen. Diffusion on complex networks : a way to probe their large-scale topological structures. *Physica A : Statistical Mechanics and its Applications*, 336(1-2) :163–173, 2004.

[77] S. H. Strogatz. Exploring complex networks. *Nature*, 410 :268–276, March 2001.

[78] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32(4) :425–443, 1969.

[79] Stijn van Dongen. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, May 2000.

[80] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks. *preprint arXiv :cs/0702048.*

[81] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301) :236–244, 1963.

[82] S. Wasserman and K. Faust. *Social network analysis.* Cambridge University Press, Cambridge, 1994.

[83] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684) :440–442, June 1998.

[84] Richard J. Williams and Neo D. Martinez. Simple rules yield complex food webs. *Nature*, 404 :180–183, 2000.

[85] Fang Wu and Bernardo A. Huberman. Finding communities in linear time : A physics approach. *The European Physical Journal B*, 38 :331–338, 2004.

[86] Luh Yen, Fabien Wouters, François Fouss, Michel Verleysen, and Marco Saerens. Clustering using a random walk based distance measure. In *Proceedings of the 13th Symposium on Artificial Neural Networks (ESANN 2005)*, pages 317–324, 2005.

[87] Wayne W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33 :452–473, 1977.

[88] Haijun Zhou and Reinhard Lipowsky. Network Brownian motion : A new method to measure vertex-vertex proximity and to identify communities and subcommunities. In *International Conference on Computational Science*, pages 1062–1069, 2004.

[89] `http://liafa.jussieu.fr/~pons/`.

[90] http://overcrowded.anoptique.org/projetautograph/.

[91] http://www.cs.cornell.edu/projects/kddcup/datasets.html.