Algorithmique de graphes appliquée à la mesure passive de la mobilité dans les réseaux routiers

Antoine Huchet

La Rochelle Université

LA ROCHELLE UNIVERSITÉ

ÉCOLE DOCTORALE EUCLIDE

LABORATOIRE L3i

THÈSE présentée par :

Antoine HUCHET

soutenue le : **13 décembre 2023**
pour obtenir le grade de : **Docteur de La Rochelle Université**
Discipline : **Informatique et Applications**

Algorithmique de graphes appliquée à la mesure passive de la mobilité dans les réseaux routiers

---

Rapporteurs	Christophe Crespelle	Professeur des universités	Université Côte d'Azur
	Marcelo Dias de Amorim	Directeur de recherches	CNRS, Sorbonne Université
Examinateurs	Karell Bertet	Professeure des universités	La Rochelle Université
	Anne Fladenmuller	Professeure des universités	Sorbonne Université
Direction	Jean-Loup Guillaume	Professeur des universités	La Rochelle Université
	Yacine Ghamri-Doudane	Professeur des universités	La Rochelle Université

Thèse réalisée au Laboratoire L3i
La Rochelle Université - Institut LUDI
Avenue Michel Crépeau
17042 La Rochelle cedex 01

Tel : +33 5 46 45 82 62
Web : `http://l3i.univ-larochelle.fr`

Sous la direction de Jean-Loup Guillaume Professeur des universités
Yacine Ghamri-Doudane Professeur des universités

Financement Allocation de l'Agence Nationale de la Recherche

Résumé

L'étude de la mobilité humaine en milieu urbain présente de nombreux intérêts : l'aménagement urbain, l'étude de la diffusion de l'information ou la propagation de maladies entre individus. Dans cette thèse, nous étudions la mobilité au travers de *graphes* : des outils mathématiques permettant de modéliser des réseaux, notamment routiers.

L'étude des graphes permet d'analyser ces réseaux, de comprendre leur organisation. Grâce à l'étude des graphes, il est par exemple possible de trouver le chemin le plus court entre deux endroits d'un réseau routier, ce qui rend possible l'utilisation de systèmes de navigation. Il devient aussi possible d'identifier des zones d'intérêt dans un réseau routier, des groupes d'amis dans un réseau social, ou encore de recommander du contenu à des utilisateurs.

Nous choisissons de modéliser le milieu urbain sur lequel nous travaillons par des graphes routiers. Nous étudions donc la mobilité d'individus se déplaçant dans un graphe routier.

Les jeux de données de mobilité, incluant un réseau routier et des trajectoires dans ce réseau, ne sont pas disponibles en quantité suffisante. Une première étape dans l'étude de la mobilité consiste donc à créer des jeux de données. Nous développons une méthodologie permettant de mesurer la mobilité humaine à grande échelle, tout en limitant les biais liés à la sélection des participants à notre étude. Notre approche est basée sur des capteurs qui peuvent voir et différencier, les individus. Nous développons une heuristique permettant de décider où placer ces capteurs pour observer un maximum de déplacements, tout en limitant le nombre de capteurs à déployer.

Par la suite, nous étudions le concept de communautés dans les graphes représentant des réseaux routiers. Les trajectoires d'individus ont tendance à rester dans les mêmes zones géographiques, appelées zones fonctionnelles. Il a été montré que ces zones fonctionnelles peuvent être identifiées grâce aux outils de détection de communautés. Nous présentons une méthode permettant d'identifier des communautés de meilleure qualité. Cette méthode consiste en une étape de précalcul, permettant de filtrer une partie du bruit présent dans les données avec lesquelles certains algorithmes fonctionnent.

Enfin, nous présentons une nouvelle définition de communauté qui est plus pertinente dans le cas de réseaux routiers comme dans d'autres types de réseaux. Cette définition considère des groupes de sommets proches les uns des autres, plutôt que des groupes de sommets où il y a beaucoup de connexions. Nous explorons les différences entre cette nouvelle définition et la définition classique puis explorons des pistes permettant d'identifier ces communautés efficacement.

Cette thèse s'inscrit dans le cadre du projet MITIK (Mesures de mobilité et inférence de contact à partir de mesures passives et non-intrusives, ou *Mobility and contact traces from non-intrusive passive measurements* en anglais), financé par l'Agence Nationale de la Recherche (ANR). Ce projet propose d'étudier la mobilité de personnes en milieu urbain en s'appuyant sur des mesures passives de mobilité.

Graph algorithms applied to passive mobility measurement in road networks

Abstract

The study of human mobility in an urban environment has many purposes : urban planning, the study of information dissemination or the spread of diseases. In this thesis, we study mobility through *graphs* : mathematical tools for modeling networks, including road networks.

The study of graphs allows analyzing those networks and understand how they are organized. The study of graphs makes it possible, for example, to find the shortest path between two points on a road network, enabling the use of navigation systems. It is also possible to identify areas of interest in a road network, groups of friends in a social network, or to recommend content to users.

We have chosen to model the urban environment we are working on using road graphs. We therefore study the mobility of individuals moving in a road graph.

Mobility datasets, including a road network and trajectories within this network, are not available in sufficient quantity. A first step in the study of mobility is therefore to create such datasets. We develop a methodology to measure human mobility on a large scale, while limiting the biases associated with the selection of participants in our study. Our approach is based on sensors that can see and differentiate individuals. We develop a heuristic to decide where to place those sensors to see as many trajectories as possible, while limiting the number of sensors to be deployed.

We then study the concept of communities in graphs representing road networks. The trajectories of individuals tend to stay within the same geographical areas, called functional urban areas. It has been shown that these functional areas can be identified using community detection tools. We present a method for identifying higher-quality communities. This method consists in a pre-calculation step, which filters out some of the noise present in the data with which some algorithms work.

Finally, we present a new definition of community that is more relevant to road networks. This definition considers groups of vertices close to each other, rather than groups of vertices with many connections. We explore the differences between this new definition and the classic one, and then explore ways of identifying these communities effectively.

This thesis is part of the MITIK project (Mobility and contact traces from non-intrusive passive measurements), funded by the French National Research Agency (ANR). The aim of the project is to study the mobility of people in an urban environment using passive mobility measurements.

Remerciements

Si ma thèse a pu se dérouler dans de bonnes conditions, c'est grâce à de nombreuses personnes, que je souhaite remercier.

Tout d'abord mes directeurs de thèse. Jean-Loup, qui a su me guider du début à la fin de ma thèse, et même après. Merci d'avoir été toujours très arrangeant et sérieux. Merci de m'avoir accompagné dans l'étude des réseaux complexes. Yacine, sans qui cette thèse n'aurait pas pu exister. Merci pour tes conseils, et tout ce que tu m'as appris.

Merci à Marcelo Dias de Amorim et Christophe Crespelle, d'avoir accepté d'être rapporteurs. Je sais que cela représente un travail conséquent, je vous en suis reconnaissant. Merci à Karell Bertet et Anne Fladenmuller d'avoir accepté d'être examinatrices.

Merci à Jean-Loup, Damien, Karell, Muriel et tous les autres enseignants qui m'ont permis d'enseigner à leurs côtés.

J'ai eu la chance d'effectuer une visite à l'Indian Institute of Technology de Kharagpur en Inde. Merci Bivas de m'avoir encadré pendant mon séjour. Merci aux membres de Complex Networks Research Group (CNERG) de m'avoir accueilli et accompagné durant mon séjour. Enfin, merci à Jean-Loup de m'avoir mis en contact avec nos collègues indiens, et de m'avoir accompagné jusqu'en Inde !

Merci à Mélanie, Laëtita et Kathy, les secrétaires du laboratoire, pour leur patience et leur aide dans diverses démarches administratives.

Merci à Michel Habib de m'avoir fait découvrir la recherche pendant un stage de master, et merci à Sophie Laplante d'avoir cru en moi au moment des admissions en master.

Merci à mes amis du laboratoire pour leur aide et leurs conseils. Merci aux membres de ma famille pour leur soutien, leurs relectures, et tout le reste.

Enfin, si j'ai pu mener ces études universitaires, c'est grâce à l'université gratuite (ou presque) et ouverte à tous. J'espère qu'elle le restera.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Sommaire

1.1 Contexte

Les graphes permettent de modéliser un grand nombre d'informations issues du monde réel. Ces objets peuvent représenter des choses très diverses, comme le réseau de l'Internet, le réseau du web, mais aussi des réseaux routiers, ferroviaires, ou encore le réseau de distribution de l'électricité. De manière plus générale, dès lors que plusieurs entités interagissent, il est possible de représenter ces interactions sous forme de réseau, donc de graphe. Des domaines très variés étudient des données représentées par des graphes. Les biologistes étudient les réseaux d'interactions entre protéines, les sociologues étudient les réseaux sociaux, formés des liens d'amitiés, professionnels, ou familiaux.

L'étude des graphes permet d'analyser ces réseaux, de comprendre leur organisation, leur structure. Grâce à l'étude des graphes, il est par exemple possible de trouver le chemin le plus court entre deux endroits d'un réseau routier, ce qui rend possible l'utilisation de systèmes de navigation. Il devient aussi possible d'identifier des groupes d'amis dans un réseau social, ou de recommander du contenu à des utilisateurs. La création d'outils permettant d'analyser ces graphes est un domaine de recherche très actif, car l'impact de ces outils peut être important.

Nombre de propriétés sur les graphes s'avèrent difficiles à calculer en un temps raisonnable. Par exemple, il n'existe aucune manière connue de décider efficacement si un

graphe G est contenu dans un autre graphe H [137]. Il n'est pas non plus possible de trouver en un temps raisonnable un chemin qui parcourerait exactement une fois tous les croisement de rue d'un réseau routier [123]. Pour ces raisons, nous travaillons souvent avec des solutions approchées, plutôt que des solutions exactes, car il est souvent possible de trouver de bonnes approximations en un temps raisonnable.

Dans cette thèse, nous nous intéressons plus précisément à l'étude de la mobilité des personnes dans un milieu urbain au travers de l'étude de l'impact de la structure du réseau sur les déplacements humains. En effet, le milieu urbain que nous étudions est composé d'un réseau de rues, et qui peut être représenté par un graphe. Les déplacements de personnes correspondent alors à des trajectoires dans le graphe représentant le réseau de rues en question. Pour se déplacer d'un point à un autre, nous avons tendance à emprunter le chemin le plus court. Peut-être que nous favorisons aussi les grandes avenues ou les grands carrefours. De plus, nous avons tendance à visiter fréquemment les mêmes endroits, qui peuvent être identifiées uniquement grâce à la structure du réseau routier [60]. L'étude du lien entre mobilité et structure du réseau routier soulève donc d'importantes questions d'aménagement du territoire. Nous étudions dans cette thèse diverses propriétés des graphes routiers et développons des outils permettant de calculer ces propriétés en un temps raisonnable.

Cette thèse s'inscrit dans le cadre du projet MITIK (Mesures de mobilité et inférence de contact à partir de mesures passives et non-intrusives, ou *Mobility and contact traces from non-intrusive passive measurements* en anglais), financé par l'Agence Nationale de la Recherche (ANR). Ce projet propose d'étudier la mobilité de personnes en milieu urbain en s'appuyant sur des mesures passives de mobilité. Ceci passe par une première phase de mesure de la mobilité d'individus dans un milieu urbain bien défini. Pour ce faire, nous avons besoin, à partir de la structure du réseau, de décider d'endroits stratégiques où se placer pour observer les trajectoires d'individus. Les données de mobilité issues de ces mesures permettent ensuite d'étudier les contacts entre personnes. Si deux trajectoires se croisent en un point, nous pouvons ainsi en déduire que ces deux personnes ont été en contact. Ces trajectoires ont tendance à rester dans les mêmes zones géographiques, qu'il est possible d'identifier grâce à la structure du réseau routier [60]. Il est par conséquent intéressant d'identifier ces zones de la manière la plus précise possible.

1.2 Contributions

Nous souhaitons mesurer efficacement la mobilité dans une zone urbaine, composée d'un réseau de rues, et étudier les trajectoires humaine via la notion de communautés.

Pour ce faire, nous présentons une heuristique de placement de capteurs, suivi d'algorithmes de détection de communautés, pour finir par l'exploration d'une nouvelle définition de communautés.

Notons que les problèmes que nous étudions sont difficiles à calculer, nous cherchons donc des solutions approchées. Ainsi, nous parlons d'heuristiques plutôt que d'algorithmes, car nos méthodes fournissent des solutions approchées pour lesquelles aucune

garantie théorique n'est donnée.

1.2.1 Heuristique de placement

Pour notre première contribution, consacrée à la mesure de la mobilité dans un milieu urbain, nous souhaitons pouvoir réaliser des mesures à grande échelle et en ciblant un échantillon de population aussi représentatif que possible.

Pour nous assurer de la précision des données nous choisissons de ne pas utiliser de technologies de positionnement par satellites, car il arrive que ce genre de mesures soit trop peu précis pour que nous puissions les exploiter. Nous choisissons d'utiliser des capteurs qui peuvent voir les personnes passer et les différencier. Ces capteurs permettent de savoir qu'une certaine personne est passée à un certain endroit à un certain moment. L'utilisation de capteurs permet de ne pas avoir à recruter de participants et de capturer la mobilité de toute une zone en mesurant la population de manière non discriminée. Ceci nous assure de l'absence de biais lié à l'échantillon de population dans nos mesures. Enfin, l'utilisation de capteurs nous permet d'étudier une zone potentiellement très vaste, pourvu que nous ayons suffisamment de capteurs.

En ce sens, nous développons une heuristique permettant de décider sur quels croisements de rues il est judicieux de placer des capteurs, dans le but de capturer un maximum de trajectoires possibles.

Cette heuristique utilise des notions de théorie des graphes, comme le problème de la couverture par sommets, ou des mesures de centralité grâce auxquelles nous identifions des points de passage sur lesquels il est judicieux de placer un capteur. Notre heuristique détermine ensuite un compromis entre la précision de la mesure et la quantité de capteurs à notre disposition. Un nombre important de capteurs fournira un nombre de points de données plus important et permettra ainsi une reconstruction plus précise des trajectoires mais sera bien évidemment plus coûteuse.

Nous étudions ensuite la pertinence de nos résultats grâce à des jeux de données composés d'un réseau routier, modélisé sous forme de graphe, ainsi que de trajectoires, sous forme de chemins dans ce graphe. Ceci nous permet de simuler un déploiement de capteurs puis de mesurer la quantité de trajectoires capturées par ce déploiement.

1.2.2 Communautés consensuelles filtrées

La deuxième partie, qui constitue le cœur de cette thèse, s'intéresse à la détection de communautés. Ce problème consiste à identifier les zones densément connectées d'un graphe. Dans le cas d'un graphe routier, ceci correspond aux groupes de croisements de rues qui sont reliés par un nombre important de portions de rues.

Les trajectoires humaines ont tendances à rester dans les mêmes zones géographiques. Ces zones, appelées *zones fonctionnelles*, peuvent être identifiées grâce à la notion de communautés [60]. Ainsi, il nous apparait pertinent d'étudier la structure communautaire des graphes routiers dans le cadre de notre étude de la mobilité. De plus, nous pensons que la structure communautaire des graphes représentant des réseaux routiers peut être pertinente dans le cadre de notre problème de placement de capteurs.

L'étude de la structure communautaire de nos graphes nous a mené a étudier le problème du non-déterminisme des algorithmes de détection de communautés. Pour rendre le calcul déterministe, des algorithmes de détection de communautés *consensuelles* sont utilisés.

Nous avons remarqué que ces algorithmes utilisent des données bruitées, et qu'une partie de ce bruit peut être filtré, menant à des communautés de meilleure qualité.

Nous avons ainsi défini un filtre, qui représente une étape de précalcul à la détection de communautés consensuelles, et rajoute donc nécessairement du temps de calcul. Ce filtre peut être utilisé en complément d'un algorithme de détection de communauté consensuelle, et permet donc d'améliorer la qualité de nombreux algorithmes.

1.2.3 Communautés de fermeture transitive

La troisième et dernière partie de cette thèse s'intéresse à la détection de communautés avec une définition différente des communautés, qui est pertinente dans ce cadre de l'étude de la mobilité. Nous considérons dans cette dernière partie que les communautés sont des groupes de sommets proches les uns des autres, plutôt que densément connectés.

En effet, dans un réseau routier il est possible de se déplacer d'un croisement de rue à un autre même s'ils ne sont pas directement reliés par une rue, simplement en passant par un troisième croisement de rue. Dans ce contexte, deux croisements de rues peuvent être proches, sans toutefois être connectés. Ainsi, pour notre étude de mobilité humaine, et notre problème de placement de capteurs, il nous apparait pertinent de considérer des communautés dans lesquelles les sommets sont proches les uns des autres.

Après avoir vérifié que les communautés, selon l'une ou l'autre définition que nous choisissons sont effectivement différentes, nous nous intéressons au calcul de ces communautés dont les sommets sont proches.

Encore une fois, le calcul d'une solution exacte serait trop coûteux en temps. Nous explorons donc une méthode approchée permettant d'obtenir un bon compromis entre le temps de calcul et la qualité de la solution.

1.3 Organisation de cette thèse

Cette thèse s'intéresse à la mobilité humaine, en étudiant le lien entre la structure du réseau routier dans lequel les individus évoluent, et leurs déplacements. Cette thèse est organisée en plusieurs chapitres, présentés comme suit :

En chapitre 2, nous présentons rapidement quelques notions de bases concernant les algorithmes et les graphes, suivis de notions qui seront utilisées tout au long de ce document, comme le problème de la couverture par sommets, des mesures de centralités et de similarités, ainsi que des notions de détection de communautés.

En chapitre 3, notre première contribution est décrite. Il s'agit d'une heuristique de placement de capteurs dans le but de réaliser des mesures passives de mobilité dans un

réseau routier. Nous étudions son efficacité sur des jeux de données réels, et la comparons à des méthodes de référence.

Le chapitre 4 est consacré à notre deuxième contribution, qui porte sur la détection de communautés dans les graphes. Nous présentons une étape de précalcul, qui permet d'améliorer la qualité des communautés produites par de nombreux algorithmes. Nous vérifions la performance de notre méthode de calcul sur des jeux de données réels et synthétiques.

Le chapitre 5 est une étude exploratoire d'une nouvelles définitions de communautés. Après avoir présenté des éléments montrant que cette nouvelle définition est effectivement différente de la définition classique, nous présentons de premiers résultats, ouvrant la voie à une manière efficace de calculer ces communautés.

Enfin, le chapitre 6 conclus ces travaux de thèse, et présente quelques perspectives de recherches ouvertes par ces travaux.

Chapitre 2

État de l'art

Sommaire

2.1 Introduction

Ce chapitre présente des concepts qui seront utiles dans cette thèse. Une première section présente d'abord des notions de base sur les algorithmes et leur complexité, puis présente ensuite des notions de base sur les graphes. Une deuxième section présente le problème de la couverture par sommets, qui sera utilisé dans le chapitre 3. Nous présentons ensuite des mesures permettant d'identifier les sommets et arêtes les plus importantes d'un graphe, ainsi que des mesures de similarité entre sommets, qui seront utilisées dans les chapitres 3, 4 et 5. Enfin, nous présentons la notion de communautés dans les graphes utilisée dans les chapitres 4 et 5. Nous passons en revue quelques mesures permettant d'évaluer leur qualité, puis plusieurs algorithmes de détection de communautés.

2.2 Notions de base

2.2.1 Sur les algorithmes et les problèmes

Certains problèmes peuvent être résolus en *temps polynomial*. C'est-à-dire qu'il existe un algorithme qui permet de le résoudre, et dont le temps d'exécution varie en suivant une fonction polynomiale en la taille des données en entrée du problème. De tels problèmes appartiennent à la classe de problèmes P.

Pour certains problèmes, il est possible de vérifier qu'une solution proposée est effectivement solution du problème, en un temps polynomial. Ces problèmes appartiennent à la classe NP. Les problèmes les plus difficiles à résoudre de la classe NP sont appelés NP-complets. Les problèmes au moins aussi difficiles que les problèmes NP-complets s'appellent les problèmes NP-difficiles.

Il n'existe aucun moyen connu de résoudre les problèmes NP-complets (ni NP-difficiles) en temps polynomial. Pour cette raison, quand nous devons résoudre de tels problèmes, nous utilisons des *algorithmes d'approximation*. Ces algorithmes fournissent une solution *approchée*, qui ne correspond pas forcément à la meilleure solution possible. Nous parlerons d'algorithmes de k-approximation lorsque la solution fournie par l'algorithme est, dans le pire des cas, k fois pire que la solution optimale. Notons que lorsqu'une telle borne n'est pas connue, un algorithme d'approximation s'appelle alors, une *heuristique*.

2.2.2 Sur les graphes

Un graphe G est une représentation mathématique d'un réseau. Il est composé de sommets, représentés par des points noirs dans la figure 2.1, et d'arêtes, représentées par des segments noirs dans la figure 2.1. Une arête permet de relier deux sommets entre eux. L'ensemble des sommets est appelé V et l'ensemble des arêtes est appelé E. Un graphe peut être défini par son ensemble de sommets et d'arêtes, soit $G = (V, E)$. Le nombre de sommets d'un graphe est noté n $(= |V|)$, et son nombre d'arêtes m $(= |E|)$. Un graphe peut être *pondéré* ou *non-pondéré* selon qu'on assigne ou non un poids à ses arêtes.

Lorsqu'il représente des données du monde réel, un graphe est appelé un graphe *de terrain* (*complex network* en anglais). Dans un graphe de terrain représentant un réseau social par exemple, les sommets seraient des personnes, et deux personnes seraient reliées par une arête si elles se connaissent. Dans un réseau routier, les sommets pourraient être des croisements de rue et les arêtes correspondraient à des rues. Comme nous le verrons plus tard, le graphe de la figure 2.1 représente un réseau routier, c'est donc un graphe de terrain.

Quand deux sommets sont reliés par une arête, on dit qu'ils sont *adjacents*. Dans le graphe de la figure 2.2a, les sommets 1 et 3 sont adjacents car ils sont reliés par l'arête $\{1, 3\}$. Une suite d'arêtes consécutives permettant de relier un sommet à un autre est appelé une *chaîne*. Les sommets 3 et 2 du graphe de la figure 2.2a ne sont pas adjacents, mais ils sont reliés par la chaîne $[\{3, 1\}, \{1, 2\}]$. La longueur d'une chaîne correspond au nombre d'arêtes qui le composent. Une chaîne de longueur k s'écrit P_k. La chaîne qui relie les sommets 3 et 2 est de longueur 2. De manière générale, une chaîne P_k peut s'écrire

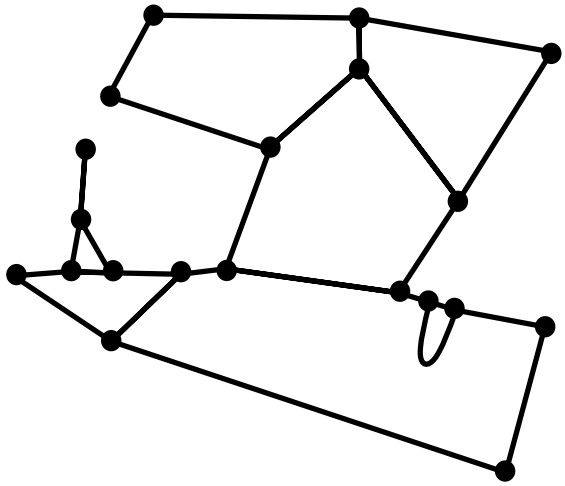


FIGURE 2.1 – Un graphe de terrain G

de la forme suivante :

$$P_k = [\{u_0, u_1\}, \{u_1, u_2\}, \ldots, \{u_{k-1}, u_k\}], \forall i, \{u_{i-1}, u_i\} \in E$$

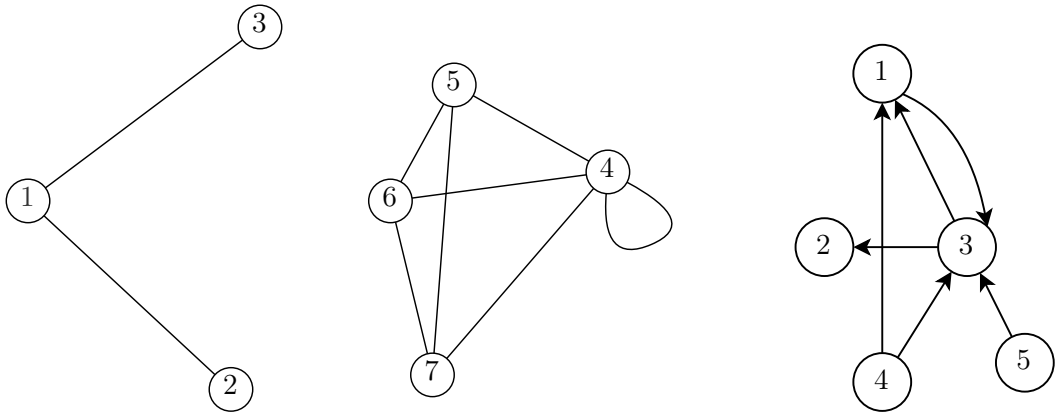
Une chaîne dans lequel le premier et le dernier sommets sont identiques s'appelle un *cycle*. Quand les deux extrémités d'une arête correspondent au même sommet, cette arête s'appelle une *boucle*. Dans le graphe de la figure 2.2a, l'arête $\{4, 4\}$ est une boucle.

Un graphe peut être *orienté*, auquel cas, une arête relie deux sommets depuis un sommet *source* i vers un sommet *destination* j, mais pas depuis j vers i. Ceci n'empêche pas la présence de deux arêtes (i, j) et (j, i). Un graphe orienté est représenté en figure 2.2b. Dans le cas d'un graphe orienté, une arête s'appelle un *arc*, une chaîne s'appelle un *chemin* et un cycle s'appelle un *circuit*.

Le *degré* d'un sommet i, noté d_i, correspond au nombre d'arêtes incidentes à ce sommet. Le degré du sommet 1 de la figure 2.2a est 2, noté $d_1 = 2$, car le sommet 1 a 2 arêtes incidentes. Si le sommet a une boucle, celle-ci doit être comptée deux fois ($d_4 = 5$).

Dans un groupe de sommets, quand il existe un chemin depuis n'importe quel sommet vers n'importe quel autre sommet, et si ce groupe est de taille maximale, on dit que ce groupe de sommets forme une *composante connexe*. Le graphe de la figure 2.2a a deux composantes connexes : $\{1, 2, 3\}$ et $\{4, 5, 6, 7\}$.

Un graphe peut être représenté dans l'espace, on parle alors de *plongement*. Quand un graphe est représenté dans un espace à deux dimensions, on parle de plongement dans le plan. Dans ce document, les graphes des figures 2.1 et 2.2a sont plongés dans le plan. Quand un graphe peut être plongé dans le plan sans qu'aucune de ses arêtes ne se croisent, on dit qu'il est *planaire*. Le graphe de la figure 2.1 est planaire, car un exemple de plongement dans le plan, sans qu'aucune arêtes ne se croisent, est dessiné en figure 2.1. Le graphe de la figure 2.2a est planaire, car il existe une (autre) manière de



(a) Un graphe H (b) Un graphe orienté

FIGURE 2.2 – Deux exemples de graphes

le dessiner sans qu'aucune arêtes ne se croisent. Il suffit par exemple de dessiner l'arête $\{5, 7\}$ de manière courbe à gauche du sommet 6.

Nous étudierons par la suite des graphes représentant des réseaux routiers. Notons que les graphes routiers sont proches d'être *planaires*. Il peut y avoir des ponts et des tunnels qui pourraient les rendre non planaires, mais un graphe routier ne peut pas être arbitrairement complexe en raison de contraintes géographiques. Étant donné que certains problèmes algorithmiques sont plus faciles à résoudre sur les graphes planaires, cette caractéristique pourrait s'avérer précieuse.

Une manière courante de représenter un graphe est la *matrice d'adjacence* A, qui est une matrice de taille $n \times n$, où n est le nombre de sommets du graphe. Dans cette matrice, la valeur à la ligne i, colonne j, noté A_{ij}, est égale à 1 quand il existe une arête entre le sommet i et j, sinon 0. Dans la figure 2.2a, $A_{21} = 1$ car les sommets 2 et 1 sont reliés dans la figure 2.2a, et $A_{44} = 1$ car le sommet 4 possède une boucle. Notons que dans le cas d'un graphe pondéré, les valeurs non-nulles de la matrice d'adjacence représenteraient le poids de l'arête en question, et ne seraient donc pas forcément égales à 1. De plus, si la matrice est symétrique pour les graphes non orientés, elle ne l'est pas forcément pour les graphes orientés.

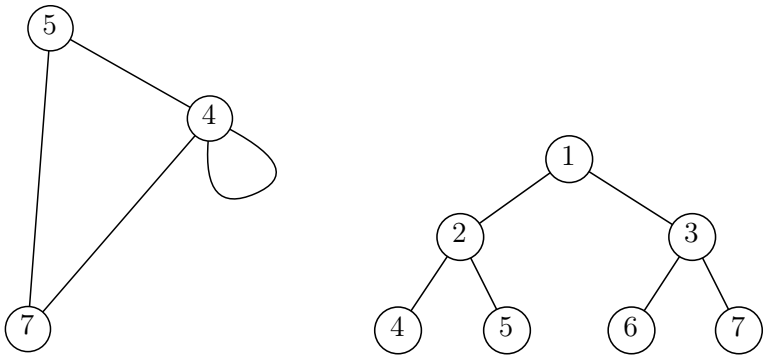
Il est possible de construire un *sous-graphe* induit en ne conservant qu'une partie des sommets d'un graphe. Par exemple, le graphe de la figure 2.4a est un sous-graphe induit du graphe de la figure 2.2a, dans lequel seuls les sommets de l'ensemble $U = \{4, 5, 7\}$ et les arêtes dont les deux extrémités sont dans U ont été conservés. Ce graphe est noté $\tilde{G}(U)$.

Les *arbres* sont une classe de graphes non orientés, connexes, et sans cycles, comme dans la figure 2.4b.

Dans un graphe, un cycle de longueur 3 est appelé un *triangle*, comme illustré en figure 2.5a. Il existe de nombreuses classes de graphes. Par exemple, si tous les cycles de 4 sommets ou plus ont une corde, c'est-à-dire une arête reliant deux sommets non adjacents

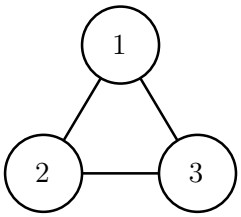
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

FIGURE 2.3 – Matrice d'adjacence du graphe de la figure 2.2a

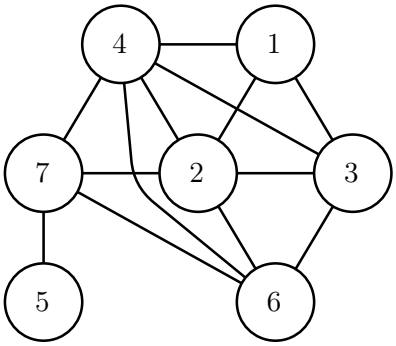


(a) Un sous-graphe induit $\tilde{G}(U)$ (b) Un arbre

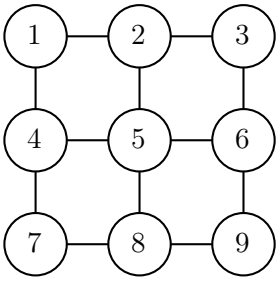
FIGURE 2.4 – Un sous-graphe induit (gauche) et un arbre (droite)



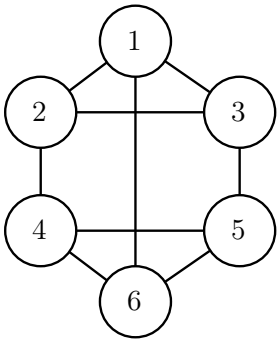
(a) Un triangle



(b) Un graphe triangulé



(c) Une grille



(d) Un graphe 3-régulier

FIGURE 2.5 – Quatre exemples de graphes

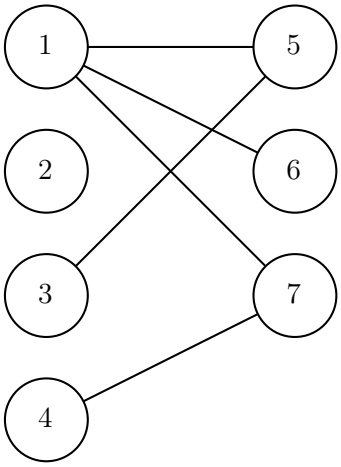


FIGURE 2.6 – Un graphe biparti

du cycle, alors ce graphe est un graphe *triangulé*, ou *cordal*. C'est le cas du graphe de la figure 2.5b. Une *grille* est un graphe qui peut être plongé dans le plan, en dessinant une grille, comme dans la figure 2.5c. Un graphe d-régulier est un graphe dans lequel tous les sommets sont de degré d, comme dans la figure 2.5d. Enfin, la figure 2.6 montre un graphe *biparti*, c'est-à-dire un graphe $G = (V, E)$ dans lequel l'ensemble de sommets V peut être scindé en deux sous-ensembles V_1 et V_2 de manière à ce que toutes les arêtes $e \in E$ aient une extrémité dans V_1 et l'autre dans V_2. Dans cette figure, $V_1 = \{1, 2, 3, 4\}$, et $V_2 = \{5, 6, 7\}$. Notons qu'il existe beaucoup d'autres classes de graphes et que la liste présentée dans ce document est loin d'être exhaustive.

2.2.3 Largeur d'arbre des graphes routiers

La *largeur d'arbre* [11], ou largeur arborescente, d'un graphe est une mesure qui représente à quel point ce graphe est proche d'être un arbre, les arbres ayant une largeur d'arbre de 1. Elle est calculée en construisant la *décomposition en arbre* T du graphe G de la manière suivante. Les sommets de T sont les ensembles $X_1 \dots X_n$, avec $X_i \subset V$. L'union de tous les ensembles X_i est égale à V. De plus, pour chaque arête $\{i, j\}$ dans G, il existe un ensemble X_i qui contient à la fois i et j. Si deux ensembles X_i et X_j contiennent tous deux un sommet v, alors ils sont reliés par un chemin unique, le long duquel tous les ensembles contiennent également v. La largeur d'une décomposition en arbre est la taille de son plus grand ensemble X_i moins un. La largeur d'arbre d'un graphe G est la largeur minimale parmi toutes les décompositions en arbre possibles de G. Cette propriété est importante à connaître puisqu'il est parfois plus simple de résoudre des problèmes sur des graphes avec une largeur d'arbre est bornée [17].

Un certain nombre de problèmes NP-difficiles, comme le problème du stable maximum [138], ou certaines variantes du problème de la couverture par sommet [5], dont nous parlerons par la suite, sont simplifiés pour des classes de graphes à *largeur d'arbre*

Graphe	borne inf.	borne sup.	n	$3,28n^{0,32}$	$\sqrt[3]{n}$
Paris GANG	3	298	5 248 992	463	174
Paris LRI	55	521	4 325 486	435	163
Paris OSM FR	2	120	140 876	146	52
La Rochelle	2	15	6 157	54	18

TABLE 2.1 – Graphes routiers, avec des bornes sur leur largeur d'arbre

bornée [133].

Le calcul de la largeur d'arbre d'un graphe est un problème NP-complet [3]. Notons que les graphes planaires ont une largeur d'arbre non bornée parce qu'une grille est planaire, mais la largeur d'arbre d'une grille de $n \times n$ est de n [17]. Le calcul de la largeur d'arbre d'un graphe étant un problème difficile, certains auteurs ont étudié des bornes sur la largeur d'arbre pour différentes classes de graphes, y compris les graphes routiers.

Dibbelt *et al.* ont conjecturé que les réseaux routiers du monde réel dont le nombre de sommets est de l'ordre de 10^7 ont une largeur d'arbre de 10^2 [40]. Ils ont également fait l'hypothèse qu'une largeur d'arbre de $O(\sqrt[3]{n})$ pour les graphes routiers de n sommets pourrait être proche de la réalité [40]. Maniu *et al.* ont fourni une approximation de la largeur d'arbre d'un graphe routier à n sommets de $\approx 3,28n^{0,32}$ [96], ce qui est cohérent avec la conjecture de Dibbelt *et al.*.

Le tableau 2.1 présente des bornes sur les largeurs d'arbre de différents graphes, ainsi que les estimations données par différentes formules. Ces bornes ont été calculées à l'aide d'un outil [95], développé par Bodlaender *et al.* [19, 18]. Tous ces graphes représentent des réseaux routiers, et ont été récupérés depuis OpenStreetMap. Le graphe Paris GANG a été récupéré par l'équipe GANG d'Inria en 2015 [50], le graphe Paris LRI a été récupéré au LRI dans le cadre d'une étude sur la largeur d'arbre [96]. Les graphes Paris OSM et La Rochelle OSM ont été extraits fin 2020 dans le cadre de cette thèse.

Les bornes connues de la largeur d'arbre des graphes routiers ne sont pas suffisamment précises, et la largeur d'arbre semble croître trop rapidement pour que les algorithmes tirant parti de cette mesure puissent fonctionner en un temps raisonnable [96].

2.2.4 Discussion

Les graphes peuvent modéliser des objets très variés, il existe par conséquent une grande variété de graphes, comme nous l'avons vu dans cette section. Nous avons vu que les problèmes ne sont pas tous aussi difficiles à résoudre, qu'il est possible de quantifier le temps nécessaire à la résolution des problèmes, et qu'en fonction du temps nécessaire à leur résolution, il existe différentes stratégies. Par la suite, nous étudions un problème en particulier, celui de la couverture par sommets.

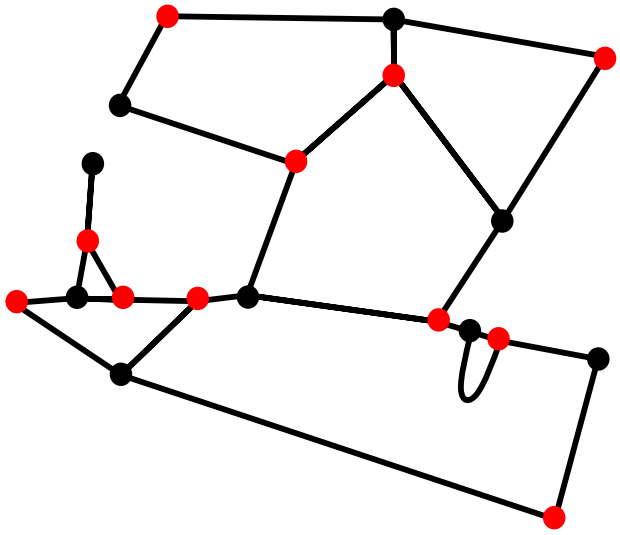


FIGURE 2.7 – Une couverture par sommets

2.3 Couverture par sommets

Dans le cadre de notre étude de la mobilité humaine, nous cherchons à observer des trajectoires d'individus (correspondant à des chemins dans un graphe routier), en plaçant des capteurs sur des sommets d'un graphe routier, comme mentionné dans l'introduction de cette thèse.

Nous présentons le problème de la *couverture par sommets*, ainsi que quelques algorithmes permettant de le résoudre. Ce problème permet d'identifier un ensemble de sommets uniformément répartis dans un graphe, ce qui est un élément central dans la solution que nous proposons au chapitre 3.

2.3.1 Problème de la couverture par sommets

Une couverture par sommets est un sous-ensemble de sommets d'un graphe, qui inclut au moins l'une des deux extrémités de chaque arête du graphe. Dans la figure 2.7, les sommets rouges constituent une couverture par sommets. Formellement, le problème de la couverture minimale par sommets prend en entrée un graphe $G = (V, E)$ et produit un sous-ensemble $S \subset V$ tel que pour chaque arête de E, au moins une de ses extrémités appartient à S, tout en minimisant $|S|$. Ce problème fait partie des 21 problèmes qui ont été prouvés NP-complets par Karp en 1972 [74].

2.3.1.1 Inapproximabilité du problème de la couverture par sommets

En 2005, Dinur *et al.* ont prouvé que le problème de la COUVERTURE MINIMUM PAR SOMMETS est NP-difficile à approximer à moins d'un facteur $1,3606$ [41].

Khot *et al.* ont montré que sous la conjecture du jeu *2-prover-1-round* [79], le problème de la couverture minimum par sommets est difficile à approximer à moins d'un facteur 2 [80]. Ils précisent que l'on peut fortement soupçonner que le problème de la couverture par sommets est NP-difficile à approximer à moins d'un facteur $2-\epsilon$ pour n'importe quel $\epsilon > 0$.

2.3.2 Algorithmes d'approximation pour le problème de la couverture par sommet

2.3.2.1 2-approximation

Puisque ce problème est NP-complet, nous utilisons des algorithmes d'approximation. Il existe une 2-approximation pour le problème de la couverture par sommets, présenté dans l'algorithme 1, qui a été découvert indépendamment par Fanica Gavril et Mihalis Yannakakis [107].

Algorithme 1 Algorithme de Gavril et Yannakakis

1: $S = \emptyset$
2: $E' = E$
3: **tant que** $E' \neq \emptyset$ **faire**
4: Retirer une arête arbitraire $\{u, v\}$ de E', ainsi que toutes les arêtes adjacentes à u ou v
5: $S = S \cup \{u, v\}$
6: **fin tant que**
7: **retourne** S

L'algorithme 1 construit un *couplage maximal*. Un *couplage* est un ensemble $E' \subseteq E$, tel qu'aucune arête n'a d'extrémité en commun. Un couplage E' est *maximal* lorsque chaque arête de $E - E'$ a une extrémité en commun avec un élément de E'. L'ensemble des extrémités des arêtes du couplage maximal E' est un recouvrement par sommets (sinon, E' ne serait pas maximal). Un tel recouvrement de sommet a une cardinalité de $2\,|E'|$. Toutes les couvertures par sommets doivent contenir au moins $|E'|$ sommets puisqu'elles doivent inclure au moins une des extrémités de chaque arête. Ainsi, une telle couverture de sommets n'est jamais plus de deux fois plus grande que la couverture par sommets minimale [51].

De manière équivalente, chaque arête du couplage doit avoir au moins une de ses extrémités dans chaque couverture par sommets. L'algorithme 1 utilise les deux, c'est donc une 2-approximation [6].

2.3.2.2 Approximation du problème du recouvrement par sommets sur les graphes planaires

L'algorithme 2 a été présenté par Bar-Yehuda et Even en 1982 [6]. Il s'agit d'un algorithme d'approximation de facteur $\frac{5}{3}$, son facteur d'approximation donc est légèrement meilleur que la 2-approximation de l'algorithme 1. Il fonctionne en temps linéaire [6].

L'approche est similaire à l'algorithme de Gavril et Yannakakis (algorithme 1), mais au lieu de travailler sur des arêtes individuelles, ils travaillent sur des sous-graphes tels que des triangles ou des graphes bipartis, pour lesquels ils obtiennent une meilleure approximation. Dans l'algorithme 2, $\tilde{G}(U)$ représente le sous-graphe de G, induit par $U \subset V$.

Algorithme 2 Algorithme de Bar-Yehuda et Even

1: $VC \leftarrow \emptyset$ ▷ Sommets de la couverture
2: $U \leftarrow V$ ▷ Reste des sommets
3: **tant que** $\tilde{G}(U)$ contient des triangles **faire** ▷ Phase 1
4: Soit $X \subset U$, t.q. $\tilde{G}(X)$ est un triangle
5: $VC \leftarrow VC \cup X$, $U \leftarrow U - X$
6: **fin tant que**
7: **tant que** $U \neq \emptyset$ **faire** ▷ Phase 2
8: Soit v un sommet de $\tilde{G}(U)$ de degré minimum d_v
9: Soit $X = \{x | \{v, x\} \text{ in } \tilde{G}(U)\} = \{x_1, x_2, \ldots, x_{d_v}\}$ ▷ Voisins à distance 1
10: Soit $Y = \{y | y \in U - \{v\} \text{ et } \exists x \in X, \{x, y\} \in \tilde{G}(U)$ ▷ Voisins à distance 2
11: Soit $E' = \{\{x, y\} | x \in X, y \in Y \text{ et } \{x, y\} \in \tilde{G}(U)\}$
12: Dans le graphe biparti (X, Y, E'), trouver un couplage maximal de $\{x_1, x_2, \ldots, x_{d_v - 1}\}$ dans Y.
13: Soit Y' l'ensemble des $d - 1$ sommets de Y utilisés dans le couplage
14: $VC \leftarrow VC \cup X \cup Y'$, $U \leftarrow U - \{v\} - X - Y'$
15: **fin tant que**

Complexité La détection de triangles de la ligne 3 peut être effectuée en temps $O(n)$ et en espace $O(n)$, avec une variante de l'algorithme d'Itai et Rodeh [6].

Notons qu'aux lignes 11 et 12, le degré de chaque $x \in X$ est d'au moins $d - 1$. Ainsi, trouver un couplage maximal de $\{x_1, \ldots, x_{d-1}\}$ dans Y est trivial.

Bar-Yehuda et Even montrent que la phase 2 de l'algorithme peut être réalisée en temps $O(n)$ en utilisant des listes doublement chaînées pour stocker les sommets, avec une liste pour chaque degré. Ensuite, un tableau est utilisé pour retrouver les sommets et leur degré, afin de récupérer les données associées à un sommet en temps constant.

Correction La phase 1 de l'algorithme utilise 3 sommets par triangle tandis qu'une couverture par sommets en nécessiterait au moins 2. Ainsi, la phase 1 produit une approximation de $\frac{3}{2}$.

Dans la phase 2, pour chaque $\tilde{G}(\{v\} \cup X \cup Y')$, il faut au moins d_v sommets dans toute couverture par sommets, puisque le couplage de $\{x_1, \ldots, x_{d_v - 1}\}$ et Y', plus l'arête $\{v, x_{d_v}\}$ est un couplage de cardinalité d_v. L'algorithme utilise quant à lui $2d_v - 1$ arêtes.

Ainsi, $|VC^*| \geq \sum_{i=1}^{e} d_i$ et $|VC| = \sum_{i=1}^{e} (2 \times d_i - 1)$. Donc :

$$\frac{|VC|}{|VC^*|} \geq \frac{2\sum d_i - e}{\sum d_i} = 2 - \frac{1}{\tilde{d}}$$

Les auteurs montrent que le degré moyen $\tilde{d}$ traité par leur algorithme n'est jamais supérieur à trois. Ceci est trivial lors de la phase 1 car un triangle n'a que des sommets de degré deux. Ensuite, pour la phase 2, les auteurs montrent que dans un graphe planaire simple (sans multi-arêtes, ni boucles) et sans triangle, alors $m < 2n$, grâce à la formule d'Euler [44]. Ainsi, la somme des degrés de $\tilde{G}(U)$ est $\sum_{i \in V} d_i = 2m < 4n$. Par conséquent, le degré moyen traité par l'algorithme 2 $\tilde{d} < \frac{4n}{n}$, donc $\tilde{d} < 4$.

Ainsi, la phase 2 produit une $2 - \frac{1}{3} = \frac{5}{3}$ approximation.

Notons que cette conjecture ne contredit pas l'algorithme 2, dans lequel le facteur d'approximation est de $\frac{5}{3}$, puisque cette borne n'est valable que pour des graphes planaires.

2.3.3 Variantes du problème de la couverture par sommets

Il existe différentes variantes du problème de la couverture par sommets. Nous passons en revue quelques-unes de ces variantes dans le but de trouver celles qui ont du sens dans notre cas. Nous cherchons à mesurer des trajectoires, ainsi nous n'avons pas forcément besoin de placer des capteurs à chaque croisement de rue, ni même sur chaque sommet d'une couverture par sommets.

La couverture par sommets *connectée*, dans laquelle la couverture doit être connexe [52]. Cette variante pourrait avoir du sens si les capteurs doivent communiquer entre eux, et donc être placés assez proches les uns des autres.

La couverture des sommets *pondérée* est une version dans laquelle les sommets du graphe sont pondérés, et le poids (plutôt que le nombre de sommets) de la couverture des sommets doit être minimisé [56].

La couverture par sommets *partielle*, dans lequel il s'agit de décider si un graphe contient au plus k sommets, couvrant au moins t arêtes. Ceci peut être pertinent dans le cas où nous avons une limite stricte sur le nombre de capteurs à déployer, mais où nous souhaitons tout de même couvrir une certaine zone.

Le problème de la couverture par sommets *paramétrée*, pour lequel la taille de la couverture doit être de k, est soluble à paramètre fixé (*fixed parameter tractable* ou FPT). Il existe un algorithme en temps $O(1,2738^k + kn)$ [29].

La couverture par sommets *éternelle* est une version dynamique du problème [82, 45], dans laquelle la couverture est composée de gardes sur les sommets du graphe. Un attaquant choisit une arête. Chaque garde a alors la possibilité de rester où il est, ou de traverser l'une des arêtes qui lui sont adjacentes. Si aucun garde ne peut traverser l'arête attaquée, alors l'attaquant gagne. S'il est possible de se défendre contre une suite infinie d'attaques, alors le défenseur gagne.

Il existe également le problème de la couverture minimum *de k-chemin* par sommets, pour lequel il s'agit de couvrir les chemins de longueur k plutôt que toutes les arêtes. Notre problème de mesure de la mobilité et de reconstruction de trajectoire est proche

de cette variante car nous devons mesurer les trajectoires, qui sont des chemins, pour ensuite les reconstruire.

2.3.4 Couverture de k-chemin par sommets

Nous nous intéressons à la mesure des trajectoires, pas à la mesure de l'ensemble du réseau routier, ainsi notre problème n'est pas exactement celui de la couverture par sommets. Considérons le problème de la couverture minimale de k-chemins par sommets, dans laquelle nous devons trouver un sous-ensemble minimal de sommets S tel que chaque chemin de longueur au moins k passe par S.

Cette variante du problème de couverture par sommets a été présentée par Brešar, Kardoš, Katrenič et Semanišin en 2011, dans le cadre d'un problème lié à la communication sécurisée dans les réseaux de capteurs sans fil [23]. Nous passons en revue quelques propriétés connues de ce problème.

Ce problème est NP-difficile dans le cas général [23] et un certain nombre de propriétés sont connues pour les arbres, les graphes d-réguliers, les grilles ou encore les graphes à largeur d'arbre bornée [23, 22]. Cependant les réseaux routiers ne vérifient pas ces propriétés et il n'est donc pas possible de tirer parti de ces propriétés dans notre cas.

Il existe un schéma d'approximation pour le problème de la couverture de k-chemin par sommets *connectée*, qui est le même problème avec une condition supplémentaire : la couverture doit être connectée. L'idée est d'énumérer des solutions sur un sous-graphe, puis de les fusionner afin d'en faire une solution [30]. Cette solution utilise des résultats de Björklund [14]. Ce schéma d'approximation fonctionne en temps $O\left(n^{O\left(\frac{1}{\epsilon^2}\right)}\right)$, avec un facteur d'approximation de $1+\epsilon$. Dans notre cas, il n'est pas nécessaire que la couverture soit connectée.

2.3.4.1 Couverture maximum de k-chemin par sommets

Le problème de la couverture *maximum* de k-chemins par sommets (ou $MaxP_kVC$) [98] prend en entrée un graphe G, un entier k, et un entier s, et requiert un sous-ensemble $S \subset V$, tel que $|S| \leq s$, tout en maximisant la quantité de chemins de longueur au moins k couverts par S.

Ce problème est NP-difficile pour les graphes triangulés, ainsi que pour les graphes scindés quand $k = 3$ [98]. D'autres propriétés sont également connues lorsque la largeur d'arbre du graphe est connue, ou pour certaines valeurs de k [98].

Ce problème a été présenté dans un article de 2018, qui n'a été cité qu'une fois en 2020. À notre connaissance, c'est tout ce que nous savons du problème de la couverture maximum de k-chemins par sommets.

2.3.4.2 Détection des P_k dans un graphe

Il existe une 2-approximation au problème de la couverture de k-chemin par sommets, pour $k = 3$, pour les sommets pondérés, en temps $O((n+m)m)$ [129]. L'idée est d'enlever

les parties les plus lourdes du graphe dans un certain ordre, jusqu'à ce qu'il ne reste qu'un graphe sans P_3. L'algorithme restaure ensuite les parties supprimées une par une, en ajoutant un sous-ensemble minimal de sommets à la couverture. Cette approche pourrait être adaptée à d'autres valeurs de k, mais elle implique de détecter efficacement les chemins de longueur k (appelés P_k).

Si nous pouvions détecter efficacement les P_k dans un graphe, nous pourrions envisager une approche similaire à l'algorithme 3, inspiré de l'algorithme 1 de Gavril et Yannakakis.

Algorithme 3 Algo. d'approx. pour la couverture minimale des k-chemin par sommets

1: **tant que** Il y a des P_k dans G **faire**
2: Retirer un P_k de G
3: Ajouter ses sommets à VC
4: **fin tant que**
5: **retourne** VC

Il existe un algorithme randomisé en temps $O(1,66^k \times n^{O(1)})$, qui permet de détecter des P_k dans un graphe [14]. L'algorithme détecte tout chemin de longueur k depuis un sommet s dans G avec une probabilité d'au moins $\frac{(1-\epsilon^{-1})}{2}$. Cet algorithme est compliqué et son implémentation n'est pas disponible.

Une autre propriété qui pourrait nous permettre de vérifier la condition en ligne 1 de l'algorithme 3 est qu'un graphe sans P_k satisfait la condition suivante : $m \leq \frac{n(k-2)}{2}$ [42]. Avec n le nombre de sommets, m le nombre d'arêtes et k la longueur des chemins recherchés.

2.3.5 Discussion

Si nous voulions surveiller un graphe routier dans sa totalité, avec le moins de capteurs possible, la couverture minimale de k-chemin par sommets permettrait de répondre à notre problème. En pratique, nous disposons d'un nombre fixe de capteurs, et nous voulons surveiller autant de trajectoires que possible.

Notre problème de placement de capteurs pourrait donc se résoudre par une couverture maximum de k-chemins par sommets, sur des graphes quasi-planaires.

Entrée :

— $G = (V, E)$ = Graphe quasi-planaire (graphe routier)
— k = Longueur minimum des trajectoires
— s = Nombre de capteurs disponibles

Sortie :

— $S \subset V$, tel que $|S| \leq s$, en maximisant le nombre de chemins de longueur au moins k couverts par S.

Puisque nous travaillons sur des graphes représentant des réseaux routiers, et que ces réseaux sont quasiment planaires à cause de contraintes géographiques, nous pouvons tirer parti de cette propriété pour résoudre notre problème de placement de capteurs.

Une autre propriété, qui aurait pu être intéressante, est la largeur d'arbre, puisque certains problèmes de théorie de graphes deviennent plus simples sur des classes de graphes dont la largeur d'arbre est bornée. Hélas, la largeur d'arbre des graphes planaires n'est pas bornée [17], nous ne pourrons pas utiliser cette propriété dans notre cas.

À notre connaissance, il ne semble pas y avoir de bons algorithmes disponibles, ce pourquoi nous nous concentrerons sur le problème de la couverture minimale par sommets et sur l'algorithme 2 de Bar-Yehuda et Evens, qui fournit de bonnes garanties théoriques pour les graphes planaires, ainsi que sur des notions de centralité des graphes, qui sont abordées dans la section suivante.

2.4 Centralités et similarités

Cette section est consacrée aux mesures de *centralité* et de *similarité*, qui permettent d'identifier les sommets et arêtes d'importance dans un graphe, ainsi que de mesurer leur différence.

2.4.1 Centralités de graphe

Nous présentons ici des mesures appelées mesures de *centralité*, qui permettent d'identifier les sommets les plus importants d'un graphe. Il a été montré que les centralités permettent une « compréhension précise de la structure des zones urbaines » [35]. Nous passons ici en revue les principales mesures de centralité de la littérature, qui sont toutes utilisées dans les chapitres 3 et 5.

Les centralités sont des valeurs attribuées à chaque sommet d'un graphe. Intuitivement, elles représentent l'importance d'un sommet dans un graphe.

2.4.1.1 Quelques centralités

La centralité *degré* est l'une des plus anciennes. Elle est définie pour chaque sommet i, comme le nombre d'arêtes incidentes à i. Dans les réseaux urbains, le degré est limité par des contraintes géographiques [9].

La centralité *force* d'un sommet i est la somme des poids des arêtes adjacentes du sommet i. Il s'agit de l'équivalent pondéré de la centralité degré, également appelée centralité de degré pondérée [8].

La centralité *closeness* d'un sommet i est l'inverse de la longueur moyenne du chemin le plus court entre le sommet i et tous les autres sommets du graphe [10].

La centralité *intermédiaire* ou *betweenness* en anglais, d'un sommet i est le nombre de plus courts chemins passant par i, divisé par le nombre de plus courts chemins entre toutes les paires de sommets [48]. Une utilisation bien connue de la centralité intermédiaire est la détection des parties denses d'un graphe. Les sommets ayant une centralité intermédiaire élevée sont considérés comme des points de passage importants, reliant les parties denses. Ainsi, un moyen d'identifier les parties denses consiste à supprimer les arêtes à forte centralité intermédiaire jusqu'à ce que les parties denses soient déconnectées.

La centralité *vecteur propre* additionne les voisins d'un sommet, à la manière de la centralité degré. Elle va plus loin en pondérant les voisins sur la base de leur propre centralité vecteur propre. Ainsi, le fait d'être adjacent à des voisins centraux rend le sommet plus central [103].

Soit la matrice d'adjacence A du graphe $G = (V, E)$, définie par :

$$A_{ij} = \begin{cases} 1 & \text{si les sommets } i \text{ et } j \text{ sont adjacents} \\ 0 & \text{sinon} \end{cases}$$

La centralité vecteur propre d'un sommet i est :

$$c_i = \frac{1}{\lambda} \sum_{j \in V} A_{ij} c_j$$

Avec λ la valeur propre associée à l'unique vecteur propre non-nul [20].

La centralité de *Katz* d'un sommet i représente l'influence relative de ce sommet. Intuitivement, elle compte le nombre de voisins d'un sommet, puis ajoute ceux à distance deux avec un coefficient plus faible, puis ajoute ceux à distance trois avec un coefficient encore plus faible et ainsi de suite.

Plus formellement,

$$c_i = \sum_{k=1}^{\infty} \sum_{j=1}^{n} \alpha^k \left(A^k\right)_{ij}$$

Avec $\alpha < 1$ un facteur d'atténuation et A la matrice d'adjacence du graphe. $\left(A^k\right)_{ij}$ représente le nombre de chemins de longueur k entre le sommet i et le sommet j [75].

La *centralité de l'information* , issue de la théorie de l'estimation (du domaine des statistiques), et non de la théorie de l'information, est basée sur l'information contenue dans tous les chemins possibles entre les paires de sommets [124]. Intuitivement, un chemin entre deux sommets i et j peut être vu comme un signal, transmis depuis le sommet i, jusqu'au sommet j. Si le chemin est de longueur 1, c'est-à-dire qu'il n'y a qu'une seule arête le long de ce chemin, un signal serait émis par le sommet i puis directement capté par le sommet j. À chaque fois qu'un signal est transmis, il peut y avoir du bruit. Supposons ce bruit constant. Un chemin de longueur k, qui serait donc

une suite de k transmissions et réceptions, aurait k fois du bruit lors de ces transmissions. Les chemins courts ayant moins de bruits, ils contribueront plus que des chemins longs à la centralité de l'information entre deux sommets.

Plus formellement, une matrice Dij est définie pour chaque paire de sommet $\{i, j\}$. Cette matrice est de dimension $s \times s$ où s est le nombre de chemins entre les sommets i et j. Les éléments Dij_{pq} représentent le nombre d'arêtes en commun entre les chemins p et q, reliant i et j. La matrice Dij est ensuite inversée pour obtenir Dij^{-1}. Enfin, I_{ij} est la somme des éléments de Dij^{-1}.

Une fois la centralité de l'information I_{ij} calculée pour une paire de sommets $\{i, j\}$, pour trouver la centralité de l'information du sommet i, il suffit de calculer la moyenne harmonique de l'information contenue dans les chemins partant de i, vers tous les sommets j :

$$I_i = \frac{n}{\sum_{j=1}^{n} I_{ij}}$$

L'*accessibilité* d'un sommet i est basée sur l'ensemble des sommets accessibles par des chemins auto-évitants de longueur h, partant de i [132]. C'est-à-dire, des chemins qui ne contiennent pas deux fois le même sommet, et qui sont de longueur h.

Soit l'ensemble P des chemins auto-évitants de longueur h partant d'un sommet i. Soit Q l'ensemble des sommets traversés par les chemins de P. Soit $p_j^{(h)}$ le nombre de fois où le sommet j est traversé par un chemin de P, divisé par la longueur totale de tous les chemins. L'accessibilité est définie par :

$$c_i^{(h)} = e^{\left(\sum_j p_j^{(h)} \log p_j^{(h)}\right)}$$

Notons qu'un sommet est central à l'échelle globale de toute la zone à étudier pour les grandes valeurs de h, et localement central (à l'échelle d'une partie de la zone, par exemple un quartier d'une ville) pour les petites valeurs de h. Pour les grandes valeurs de h, les sommets centraux se trouveraient au centre-ville et peu dans la périphérie, alors que pour des petites valeurs de h, les sommets centraux seraient mieux répartis dans différents quartiers [132].

L' *Expected Force* vise à modéliser la force d' « infection » d'un sommet i [89]. Elle est définie par :

$$c_i = -\sum_{j=1}^{J} d_j \log d_j$$

Où l'ensemble J est composé de triplets de sommets. Il s'agit de i, et deux de ses voisins, ou i, un de ses voisins j, puis un voisin de j. J contient toutes les combinaisons de triplets possibles suivant cette construction. Ainsi, deux triplets contenant les mêmes sommets, mais dans un ordre différent, peuvent cohabiter dans J. L'intuition est de

considérer le sommet i comme infecté, puis de modéliser quels sommets peuvent être contaminés par le sommet i après deux « transmissions ».

d_j correspond au degré d'un triplet $j \in J$, c'est-à-dire au nombre d'arêtes ayant exactement une extrémité dans le triplet. d_j est ensuite normalisé, c'est-à-dire divisé par la somme de tous les $d_j \in J$.

2.4.1.2 Discussion

Il existe un grand nombre de centralités, chacune permettant de mettre en avant différentes propriétés des sommets mesurés. Nous étudierons chaque centralité dans le cadre de notre étude de la mobilité humaine, dans le but de voir si une, ou plusieurs d'entre-elles permettent de capturer le mieux possible les trajectoires des individus. Nous chercherons à savoir si les sommets centraux sont les sommets par lesquels beaucoup d'individus transitent.

2.4.2 Mesures de similarité de sommets

Comme nous le présenterons dans le chapitre 4, la détection de communautés peut tirer parti de la similarité entre deux sommets. Nous passons en revue quelques mesures de similarité entre sommets.

2.4.2.1 Indice de Jaccard

L'indice de Jaccard, appelé le « coefficient de communautés » par Paul Jaccard en 1901, permet de mesurer la distance entre deux ensembles d'éléments [69]. Cette distance consiste à calculer le rapport entre le cardinal de l'intersection et de l'union de deux ensembles :

$$\mathrm{Jac}(A,\, B) = \frac{|A \cap B|}{|A \cup B|}$$

Pour appliquer cette distance à des sommets i et j, il nous suffit de poser $A =$ l'ensemble des voisins de i, et B les voisins de j.

2.4.2.2 Distance de Hamming

La distance de hamming de deux chaînes de caractères a et b de même taille est le nombre de positions où les caractères sont différents [58]. Par exemple, dans les mots de même longueur « Focal » et « Vocal », la distance de Hamming est de 1, car seule la première lettre de ces deux mots est différente.

Cette distance est utilisée en télécommunication pour compter le nombre de bits altérés lors de la transmission d'un message. Dans notre cas, nous considérons deux vecteurs a et b, correspondant respectivement aux lignes i et j de la matrice d'adjacence A d'un graphe G, donc au voisinage des sommets i et j. $a_k = 1$ si les sommets i et k sont voisins dans G, sinon $a_k = 0$.

$$\mathrm{Ham}(a,\, b) = |\{k, a_k \neq b_k\}|$$

2.4.2.3 Similarité cosinus

La similarité cosinus permet de calculer la similarité de deux vecteurs à n dimensions en calculant le cosinus de leur angle. Il s'agit du produit scalaire de ces deux vecteurs divisé par le produit de leur norme. Cette mesure est aussi utilisée en fouille de texte [122]. Le cosinus est pratique pour les mesures de distance puisqu'il est égal à 0 dans le cas de vecteurs orthogonaux et à 1 dans le cas de vecteurs identiques.

$$\cos\theta = \frac{a.b}{||a||||b||}$$

De la même manière que pour la distance de Hamming, nous construisons deux vecteurs a et b représentant respectivement les voisins de deux sommets i et j. La similarité cosinus est alors définie comme suit :

$$\frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

2.4.2.4 Coefficient de clustering d'arête

Le coefficient de clustering d'arête, ou Edge Clustering Coefficient (ECC) [111] est une mesure de la similarité entre deux sommets i et j défini comme suit :

$$ECC(i,j) = \frac{nb\_voisins\_communs(i,j)}{\min(d_i, d_j)},$$

Où $nb\_voisins\_communs(i,j)$ est le nombre de voisins en commun des sommets i et j, tandis que d_i est le degré d'un sommet i.

Cette mesure peut être utilisée pour détecter des paires de sommets dont le voisinage est similaire.

2.4.2.5 Discussion

Nous avons passé en revue plusieurs mesures permettant d'identifier des paires de sommets. Dans la suite de cette thèse nous concentrerons notre attention sur le coefficient de clustering d'arête, pour sa rapidité de calcul, et la pertinence de ses résultats dans le cadre de notre étude de mobilité.

2.5 Détection de communautés

Les trajectoires humaines ont tendance à rester dans les mêmes zones géographiques. Ces zones, appelées *zones fonctionnelles*, peuvent être identifiées en étudiant les graphes routiers à l'aide d'outils de la théorie des graphes, comme les algorithmes de détection

de communautés [60]. Ainsi, notre étude de la mobilité humaine nous a mené à étudier la notion de *communautés* dans les graphes routiers.

Le problème de la *détection de communautés* est un problème de partitionnement de données. Le but est donc de créer des *partitions* de données, c'est-à-dire de scinder notre ensemble de données en *parties* (non vides, et disjointes deux à deux). La détection de communautés est un problème de partitionnement de données sur les graphes, où les parties sont appelées *communautés*. Il s'agit dont de créer des communautés de sommets, dans lesquelles les sommets sont aussi densément connectés que possible. Il doit donc y avoir un maximum d'arêtes à l'intérieur des communautés, et un minimum d'arêtes entre les communautés.

Ces algorithmes utilisent des mesures de qualité permettant de guider leurs choix ainsi que de comparer les partitions produites par différents algorithmes.

2.5.1 Mesures de qualité

La qualité des communautés peut être évaluée à l'aide de différentes mesures. Certaines dépendent uniquement du graphe en question ainsi que des communautés calculées, comme la *modularité*. D'autres comparent les communautés calculées à une vérité terrain comme l'*information mutuelle normalisée*.

2.5.1.1 Modularité et modèle configurationnel

La *modularité* [104] est une fonction utilisée pour mesurer la qualité d'une partition de communautés dans un graphe G. Elle compare la quantité d'arêtes intra-communautaires à la quantité attendue d'arêtes intra-communautaires dans un graphe aléatoire H avec le même nombre d'arêtes et la même distribution de degrés.

Ce graphe aléatoire est construit selon le *modèle configurationnel* [7] comme indiqué dans l'algorithme 4, dans lequel un graphe H avec le même nombre de sommets que dans le graphe G est généré. Des arêtes sont ensuite ajoutées à H jusqu'à ce que les sommets de H aient le même degré que les sommets de G. C'est-à-dire que dans les graphes G et H, il y aura le même nombre de sommets de degré i, pour toutes les valeurs de i possibles. On dit que la liste des degrés est identique dans G et H.

Algorithme 4 Modèle configurationnel (nombre de sommets n, liste de degrés D)

1: Générer un graphe H à n sommets
2: Pour chaque sommet, choisir un degré, appelé degré final f_i dans la liste de degrés D
3: **tant que** Tant qu'il existe un sommet i dont le degré $d_i < f_i$ **faire**
4: Choisir au hasard deux sommets dont le degré est inférieur à leur degré final, puis ajouter une arête entre ces sommets
5: **fin tant que**
6: **retourne** H

Dans un graphe où les arêtes sont disposées de manière aléatoire, considérons deux sommets u et v, de degré d_u et d_v respectivement. La probabilité qu'une arête parte du

sommet u pour atteindre le sommet v est $\frac{d_v}{2m-1}$ car il y a $2m-1$ possibilités de connexion. Parmi ces $2m-1$ possibilités, d_v correspondent au sommet v.

Ainsi, la probabilité que l'une des d_u arêtes du sommet u se connecte au sommet v est $\frac{d_u d_v}{2m}$.

Par conséquent, la différence entre le nombre d'arêtes entre les sommets u et v et la quantité attendue d'arêtes entre les sommets u et v dans un graphe aléatoirement câblé, avec le même nombre d'arêtes et la même distribution des degrés est :

$$A_{uv} - \frac{d_u d_v}{2m}$$

La modularité, qui prend ses valeurs dans l'intervalle $[\frac{-1}{2}, 1]$, est donc définie comme suit :

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta(c_i, c_j)$$

Avec m le nombre d'arêtes du graphe G et A la matrice d'adjacence du graphe G. d_i est le degré du sommet i. c_i est la communauté à laquelle appartient le sommet i. Le delta de Kronecker $\delta(c_i, c_j) = 1$ lorsque $c_i = c_j$, c'est-à-dire lorsque les sommets i et j appartiennent à la même communauté, 0 sinon.

La modularité est une mesure couramment utilisée pour évaluer les algorithmes de détection de communautés, mais le problème de maximisation de la modularité est NP-COMPLET [21]. Par conséquent, tous les algorithmes en temps polynomial qui maximisent la modularité que nous présentons ici sont des heuristiques.

2.5.1.2 Limites de la modularité

Bien que la modularité soit une fonction objectif largement utilisée, elle n'est pas parfaite. Le problème de la *limite de résolution* [47] indique que la modularité ne permet pas de détecter les petites communautés. En d'autres termes, si les communautés sont trop petites, la modularité sera maximale lorsque les petites communautés seront fusionnées pour en former de plus grandes. Pour palier à ce problème, un *paramètre de résolution* a été proposé, de manière à régler la taille attendue des communautés [39].

Il a aussi été montré qu'il est possible de trouver des communautés ayant une bonne modularité dans des graphes aléatoires n'ayant pas de structure communautaire [37, 57].

2.5.1.3 Information mutuelle

L'*information mutuelle* est une mesure issue de la théorie de l'information [120], qui permet d'analyser les performances des algorithmes de détection de communautés, en comparant une partition de communauté à la vérité terrain [97].

L'information mutuelle repose sur la notion d'*entropie*, dont nous nous servons pour mesurer la quantité d'information contenue dans une partition. Intuitivement, l'entropie mesure l'incertitude qu'un élément pris au hasard appartienne à une certaine partie.

Dans le cas d'une partition dans laquelle il n'y aurait qu'une seule partie, l'entropie de la partition serait donc nulle.

Considérons que tous les éléments d'un ensemble A, de taille n, ont la même probabilité d'être choisis. La probabilité qu'un élément a soit dans la partie $C_i \in C$ est $P(i) = \frac{|C_i|}{n}$. L'entropie de la partie C_i est donc :

$$H(C) = -\sum_{i=1}^{n} P(i) \log_2 P(i)$$

L'information mutuelle représente la quantité d'information partagée entre deux partitions, c'est-à-dire à quel point la connaissance d'une partition C_1 réduit l'incertitude concernant la partition C_2. L'information mutuelle est définie comme :

$$I(C_1, C_2) = \sum_{i=1}^{k} \sum_{j=1}^{l} P(i,j) \log_2 \frac{P(i,j)}{P(i)P(j)}$$

Où $P(i,j) = \frac{|C_{i,1} \cap C_{j,2}|}{n}$ est la probabilité qu'un élément appartienne à la partie $C_{i,1}$ de C_1 et $C_{j,2}$ de C_2.

L'information mutuelle maximum dépend de l'entropie des deux partitions étudiées, elle n'est donc pas bornée, ce qui la rend difficile à interpréter.

$$I(C_1, C_2) \leq \min\left(H(C_1), H(C_2)\right)$$

Pour résoudre le problème de l'information mutuelle non bornée, il a été proposé de la normaliser [125]. Nous parlons alors d'*Information Mutuelle Normalisée* (ou *Normalised Mutuel Information* ou NMI en anglais). Cette mesure permet aussi d'analyser une partition de communautés, en la comparant à la vérité terrain.

$$NMI(C_1, C_2) = \frac{I(C_1, C_2)}{\sqrt{H(C_1)H(C_2)}}$$

Notons que la NMI ne fonctionne que dans le cas de *partitions* de communautés. Dans le cas de *communautés recouvrantes*, où les sommets peuvent appartenir à plusieurs communautés, la NMI n'est plus adaptée [43].

Dans notre cas, nous travaillons avec des communautés non-recouvrantes, donc nous pourrons utiliser la NMI.

2.5.1.4 Indice de Rand

L'indice de Rand [113] est une autre mesure de similarité entre deux partitions, dont le principe est de compter le nombre d'éléments se trouvant dans une même partie. Ses valeurs vont de 0 quand aucun des éléments ne se trouve dans la même partie, à 1 quand tous les éléments se trouvent dans la même partie.

Soit A un ensemble, P_1 une partition de l'ensemble A, et P_2 une autre partition du même ensemble A. Définissons :

— a comme le nombre de paires d'éléments de A qui se trouvent dans la même partie de P_1 et dans la même partie de P_2.
— b comme le nombre de paires d'éléments de A qui se trouvent dans une partie différente de P_1 et une partie différente de P_2.
— c comme le nombre de paires d'éléments de A qui se trouvent dans la même partie de P_1 et dans une partie différente de P_2.
— d comme le nombre de paires d'éléments de A qui se trouvent dans une partie différente de P_1 et dans la même partie de P_2.

Intuitivement, a et b représentent le nombre d'éléments pour lesquels les partitions P_1 et P_2 sont indentiques, alors que c et d représentent le nombre d'éléments pour lesquels les partitions P_1 et P_2 diffèrent.

L'indice de Rand est alors défini comme :

$$R = \frac{a+b}{a+b+c+d}$$

Puisque l'espérance de l'indice de Rand entre deux partitions aléatoires n'est pas constant [61], plusieurs auteurs ont cherché à normaliser cette mesure.

Hubert et Arabie ont proposé une extension à l'indice de Rand, qu'ils appellent l'Indice de Rand Ajusté (ou *Adjusted Rand Index* ou ARI en anglais) [61], et qui représente la différence entre l'indice de Rand et son espérance. Ainsi, deux partitions identiques ont un ARI de 1, et 0 quand l'indice de Rand est égal à son espérance.

$$ARI = \frac{\text{Indice} - \text{Espérance}}{\text{Indice\_Max} - \text{Espérance}}$$

Où $\text{Indice} = \sum_{i,j} \binom{n_{ij}}{2}$, $\text{Espérance} = \frac{\sum_i \binom{n_{j\cdot}}{2} \sum_j \binom{n_{\cdot j}}{2}}{\binom{n}{2}}$, et $\text{Indice\_Max} = \frac{\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}}{2}$.

Où n_{ij} représente le nombre d'éléments en commun des parties P_{1i} et P_{2j}. $n_{i\cdot}$ représente la somme des $n_{ij} \forall j$, et $n_{\cdot j} = \sum_i n_{ij}$.

2.5.1.5 Discussion

Notons que lorsqu'il s'agit de se rapprocher d'une vérité terrain, la NMI et l'ARI sont des mesures plus fiables que la modularité, mais nécessitent de connaître cette vérité terrain. Dans la suite de ce document, quand la vérité terrain est connue, seule la NMI est utilisée, car au moment de faire les expériences, nous n'avions pas connaissance de l'ARI. Nous utilisons aussi la modularité dans certains cas où la vérité terrain n'est pas connue. Notons que certains auteurs recommandent de ne pas réduire la distance entre deux partitions à une seule valeur, qui ne permet pas, selon eux, de capturer toute la complexité des communautés [106]. Ils recommandent plutôt d'utiliser une combinaison de plusieurs mesures.

Puisqu'il n'est possible d'utiliser la NMI et l'ARI que lorsque la vérité terrain est connue, il est intéressant de travailler sur des graphes dont la structure communautaire est connue. La section suivante présente quelques-uns de ces graphes, ainsi que plusieurs manières de générer aléatoirement des graphes avec une structure communautaire connue.

2.5.2 Détection de communautés

Nous présentons le problème de la détection de communautés, puis passons en revue plusieurs algorithmes de détection de communautés de manière à déterminer ceux qui ont du sens dans notre contexte.

2.5.2.1 Problème de la détection de communautés

Le problème de la détection de communautés prend donc un graphe en entrée et produit une partition des sommets en communautés. Le nombre de partitions d'un ensemble à n éléments croît de façon super-exponentielle, suivant les nombres de Bell. Cela signifie qu'un graphe à 10 sommets a 115 milliers de partitions possibles, avec 11 sommets, il y aurait 678 milliers de partitions possibles, tandis qu'un graphe à 20 sommets aurait $5,1 \times 10^{13}$ partitions possibles. Dans le cas général, le nombre de partitions d'un ensemble à n éléments est donné par :

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$$

Il n'est donc pas possible d'énumérer toutes les solutions, nous avons besoin d'algorithmes pour la résolution de ce problème. Le reste de cette section passe en revue les principaux algorithmes de détection de communautés. Nous ne présentons ici que les méthodes les plus représentatives du domaine, étant donné le nombre très élevé de méthodes proposées dans la littérature [46]. Cette section est divisée en deux sous-sections, l'une décrivant des algorithmes basés sur la maximisation de la modularité, l'autre décrivant d'autres approches diverses.

2.5.2.2 Heuristique de Kernighan-Lin

L'*heuristique de Kernighan-Lin* [77] est un algorithme glouton publié en 1970, qui divise le graphe $G = (V, E)$ en deux sous-ensembles de sommets A et B dont la taille est la plus proche possible, tout en minimisant le nombre d'arêtes ayant une extrémité dans chacun des deux sous-ensembles. L'algorithme commence par choisir deux sous-ensembles A et B au hasard, puis, pour chaque sommet a du sous ensemble A, il calcule la valeur I_a, qui correspond au nombre d'arêtes du sommet a dont l'autre extrémité se trouve aussi dans A, et E_a le nombre d'arêtes du sommet a dont l'autre extrémité se trouve dans le sous-ensemble B. L'algorithme calcule la même chose pour le sous-ensemble B. Un coût $D_a = E_a - I_a$ est ensuite calculé. L'algorithme optimise alors de manière gloutonne, le coût total $D = \sum_{i \in V} D_i$.

2.5.2.3 Algorithme de Girvan et Newman

L'algorithme de *Girvan et Newman* [53] choisit de supprimer les arêtes qui se trouvent entre les communautés (les arêtes inter-communautés). Ils définissent la mesure *arête intermédiaire* ou *edge betweenness* (qui est basée sur la centralité *intermédiaire* [48] sur

les arêtes, qui est définie plus loin), ce qui leur permet d'ordonner les arêtes, puis de supprimer les arêtes ayant la centralité intermédiaire la plus élevée. L'algorithme s'arrête lorsqu'il n'y a plus d'arêtes à supprimer. Le graphe sera ainsi successivement divisé en composantes connexes, qui sont considérées comme des communautés. Cet algorithme produit un dendrogramme. La centralité intermédiaire des arêtes doit être recalculée après chaque suppression d'arête car quand deux communautés sont reliées par des arêtes, les auteurs expliquent qu'au moins une de ces arêtes aura une centralité intermédiaire élevée. Ainsi, en recalculant la centralité intermédiaire, nous avons la garantie qu'au moins une arête inter-communautés aura une centralité intermédiaire importante. Ce recalcul fait que cet algorithme a un temps d'exécution important. Certains choix arbitraires effectués dans cet algorithme, en cas d'égalité dans la centralité intermédiaire par exemple, le rendent non déterministe.

2.5.2.4 Algorithme de Newman

L'*algorithme de Newman* [102] est un algorithme glouton qui est beaucoup plus rapide que l'algorithme de Girvan et Newman que nous venons de présenter. C'est un algorithme dont l'objectif est le maximiser la modularité. Chaque sommet commence dans sa propre communauté, puis lorsque deux communautés sont reliées par une arête, l'algorithme les fusionne si cela augmente la modularité. L'algorithme renvoie ensuite la partition ayant la modularité la plus élevée. Cet algorithme a une complexité temporelle en $O((m+n)n)$.

L'algorithme de Clauset *et al.* [33] est une approche plus récente qui améliore la complexité temporelle de l'algorithme de Newman, en utilisant une structure de données différente, et quelques optimisations lui permettant de converger plus vite, passant à $O(md \log n)$, où d est la profondeur du dendrogramme généré par l'algorithme. La complexité a été améliorée une nouvelle fois par Wakita et Tsurumi, passant à $O(n \log^2 n)$ [134].

2.5.2.5 Algorithme walktrap

L'algorithme *Walktrap* [109] est basé sur l'idée que les marches aléatoires ont tendance à rester dans la même communauté, puisque les communautés sont densément connectées. Cet algorithme a une complexité temporelle de $O(mn^2)$ et spatiale de $O(n^2)$. Une marche aléatoire partant d'un sommet i visitera un voisin k de i avec probabilité $\frac{1}{d_i}$. À partir de cette simple observation, il est possible de calculer la probabilité P_{ij}^t qu'une marche aléatoire rejoigne le sommet j depuis le sommet i en traversant t arêtes. La valeur t doit être choisie en fonction de la taille du graphe.

Les auteurs définissent ensuite une mesure de distance entre deux sommets i et j :

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{\left(P_{ik}^t - P_{jk}^t\right)^2}{d_k}}$$

Ainsi qu'une distance entre un sommet i et une communauté C : $P^t_{Ci} = \frac{1}{C}\sum_{i\in C} P^t_{ij}$, puis une distance entre communautés :

$$r_{C_1C_2} = \sqrt{\sum_{k=1}^{n} \frac{\left(P^t_{C_1k} - P^t_{C_2k}\right)^2}{d_k}}$$

Les distances r_{ij} et $r_{C_1C_2}$ sont ensuite utilisées pour optimiser la modularité en parcourant le graphe et en fusionnant les communautés de manière gloutonne comme dans l'algorithme de Newman.

2.5.2.6 Algorithme label propagation

Dans cette approche, chaque sommet commence seul dans sa communauté. Ensuite, à chaque étape, un sommet est choisi puis il rejoint la communauté la plus fréquente parmi celles de ses voisins. Une communauté aléatoire (parmi les voisins) est choisie en cas d'égalité. Cet algorithme n'a besoin d'aucun paramètre, il est rapide, mais non déterministe car l'ordre dans lequel les sommets sont visités est arbitraire, et en cas d'égalité, la communauté choisie est aléatoire [112].

2.5.2.7 Méthode de Louvain

Cet méthode est un algorithme glouton d'optimisation de la modularité, dans lequel chaque sommet commence dans sa propre communauté. L'algorithme visite ensuite chaque sommet dans un ordre aléatoire, compare la modularité actuelle avec la modularité qui peut être obtenue en déplaçant ce sommet dans la communauté d'un de ses sommets voisins. Le déplacement qui maximise le gain de modularité est appliqué. Après quelques itérations, lorsque la modularité n'augmente plus, un nouveau graphe est construit. Les sommets de ce nouveau graphe sont les communautés du graphe précédent. Deux sommets sont reliés par une arête si les communautés correspondantes partageaient une arête dans le graphe précédent. La première phase de l'algorithme est ensuite exécutée sur ce nouveau graphe. La méthode crée ainsi plusieurs *niveaux* de communautés : un niveau à chaque fois qu'un nouveau graphe est construit. Pour finir, le niveau ayant la modularité la plus élevée est retourné, ce qui en pratique correspond au dernier niveau. Cet algorithme est très rapide en pratique et produit des partitions de bonne qualité, c'est pourquoi il est couramment utilisé. Il est non déterministe en raison de l'ordre aléatoire dans lequel les sommets sont visités [15].

La méthode de Louvain a été largement utilisée [16], et de nombreux travaux ont cherché à l'améliorer, notamment la méthode de Leiden [128].

2.5.2.8 Algorithme infomap

L'algorithme Infomap est un algorithme glouton qui n'optimise pas la modularité mais l'équation *map* [116]. L'optimisation est réalisée de la même manière que la méthode de Louvain. L'équation map est basée sur le codage de Huffman et l'entropie de Shanon.

Selon ses auteurs, l'optimisation de la modularité est pertinente pour les réseaux dans lesquels les arêtes représentent une relation entre les sommets, alors que l'équation map est plus pertinente pour les réseaux dans lesquels les arêtes représentent un flux entre deux sommets.

2.5.2.9 Discussion

Il existe de nombreux autres algorithmes de détection de communautés, qui ont été comparés par Lancichinetti [85]. Pour un examen plus approfondi de la détection des communautés, voir l'étude de Fortunato [46].

Notons que les algorithmes non déterministes mentionnés ci-dessus peuvent être rendus déterministes en fixant l'ordre dans lequel les sommets sont visités, et éventuellement des règles déterministes en cas d'égalité à certaines étapes des algorithmes. Ce n'est toutefois pas une manière satisfaisante de rendre le calcul déterministe, car différentes partitions de qualité comparable peuvent être obtenues grâce à ce non-déterminisme. L'une ne doit pas être choisie arbitrairement au profit des autres car chacune d'entre elles peut apporter de l'information pertinente concernant la structure communautaire recherchée.

Pour notre part, comme recommandé par Fortunato [46], nous utiliserons principalement les méthodes de Louvain et Infomap qui fournissent toutes deux rapidement des résultats de bonne qualité.

2.5.3 Graphes avec une structure communautaire connue

Les graphes dont la structure communautaire est connue ne sont pas toujours disponibles en quantité suffisante. Ils sont pourtant nécessaires pour pouvoir mesurer précisément la qualité des communautés calculées par les algorithmes de détection de communautés. Aussi, il peut être intéressant de générer des graphes aléatoires avec une structure communautaire connue. Nous passons en revue plusieurs modèles permettant de générer de tels graphes. Enfin, nous présentons quelques graphes de terrain dont la structure communautaire est connue.

2.5.3.1 Modèle de Girvan et Newman

Ce modèle simple permet de générer des graphes avec une structure de 4 communautés connues. Dans ce modèle, les graphes ont 128 sommets, qui sont divisés en 4 communautés de 32 sommets. Ces sommets sont ensuite reliés deux à deux avec une probabilité p_{in} pour les sommets qui appartiennent à une même communauté, et avec une probabilité p_{out} pour ceux qui appartiennent à une communauté différente, avec $p_{out} < p_{in}$. Les probabilités sont choisies de manière à ce que le degré moyen des sommets soit égal à 16 [53].

2.5.3.2 Modèle de Lancichinetti, Fortunato, et Radicci

Les graphes de Lancichinetti, Fortunato et Radicchi (*LFR*) [87] sont des graphes générés, avec une structure communautaire connue, servant de vérité terrain. Ces graphes

sont fréquemment utilisés pour mesurer la performance d'algorithmes.

Ce modèle génère des graphes avec une distribution de degrés en loi de puissance, un nombre choisi de sommets et un paramètre de mélange μ qui représente le rapport moyen entre les arêtes inter-communautaires et les arêtes intra-communautaires de chaque sommet. En d'autres termes, plus μ est élevé, plus il y aura d'arêtes entre les communautés, moins la structure communautaire du graphe sera marquée, et plus les algorithmes auront du mal à retrouver ces communautés.

Le modèle choisit d'abord un degré pour chaque sommet, à partir de la distribution en loi de puissance d'exposant γ qui lui a été donnée, puis il utilise le modèle configurationnel [99] pour relier les sommets (voir algorithme 4) entre eux. En d'autres termes, deux sommets sont choisis uniformément aléatoirement et reliés, jusqu'à ce que le degré de chaque sommet corresponde au degré choisi à partir de la distribution.

La taille des communautés est choisie à partir d'une autre distribution en loi de puissance, d'exposant β. Les sommets sont ensuite affectés aux communautés. Enfin, un recâblage est effectué afin que le degré des sommets reste le même, mais que le rapport moyen entre les arêtes inter-communautaires et intra-communautaires corresponde au paramètre de mélange μ.

De cette manière, il est possible de générer un graphe dans lequel les communautés sont connues, ainsi que la difficulté de les retrouver. Le modèle LFR constitue un outil précieux pour l'analyse des performances des algorithmes de détection de communautés.

2.5.3.3 Modèle ABCD

Le modèle *Artificial Benchmark for Community Detection* (ABCD) est un modèle proche du modèle LFR [73]. C'est une version plus simple du modèle LFR, avec moins de paramètres, ce qui permet de l'analyser d'un point de vue théorique, et de prouver certaines propriétés, comme nous le verrons dans le chapitre 4. Dans ce modèle, quand le paramètre de mélange s'approche de 1, le graphe généré se rapproche d'un graphe aléatoire, sans structure communautaire, contrairement au modèle LFR, dans lequel il n'y aurait plus d'arêtes intra-communautaires, ce qui a pour effet de créer une structure particulière, qui n'est pas tout à fait aléatoire. Le modèle ABCD est aussi plus rapide pour générer des graphes que le modèle LFR, d'un facteur 100 environ [73].

2.5.3.4 Modèles à blocs stochastiques

Le modèle *à blocs stochastiques* [59] génère des graphes dont la structure communautaire est connue. Il prend en entrée le nombre de sommets n, la structure communautaire, et une matrice de probabilité P où $P_{r,s}$ est la probabilité d'avoir des arêtes entre les sommets de la communauté r et les sommets de la communauté s.

Un graphe avec n sommets et aucune arête est d'abord généré. Puis, pour chaque paire de sommets u et v, appartenant respectivement aux communautés r et s, l'arête $\{u, v\}$ est rajoutée avec probabilité $P_{r,s}$.

Soit p_{in} la moyenne des $P_{s,s}$ pour toutes les communautés s. C'est-à-dire la moyenne des éléments diagonaux de P, qui représentent la probabilité d'avoir des arêtes intra-communautaires. Soit p_{out} la moyenne des éléments restants de P, c'est-à-dire la probabilité d'avoir des arêtes inter-communautaires. Soit $c_{in} = n \times p_{in}$ et $c_{out} = n \times p_{out}$ respectivement le nombre attendu d'arêtes intra-communautaires et d'arêtes inter-communautaires. Dans le cadre d'un modèle à blocs stochastiques avec q communautés, les communautés deviennent impossibles à trouver lorsque :

$$c_{in} - cout = \sqrt{q\left(c_{in} + (q-1)c_{out}\right)}$$

Même si, tant que $c_{in} - c_{out} > 0$, la structure communautaire existe [101].

Nadakuditi et Newman [101] ont montré que, dans le cadre du modèle à blocs stochastiques, toutes les méthodes de maximisation de la modularité présentent une transition entre une phase où les communautés sont détectables et une phase où les communautés ne peuvent pas être détectées. Lorsque $c_{in} - c_{out}$ augmente, il y a une transition de phase dans la fraction de sommets correctement classifiés [38].

2.5.3.5 Graphes de terrain avec vérité terrain

Les graphes de terrain sont des graphes qui représentent des données du monde réel. Peu d'entre-eux ont une structure communautaire connue.

Le jeu de données du *Club de karaté de Zachary* [142] a été collecté en 1977 par Wayne Zachary. Il s'agit d'un célèbre réseau social représentant un club de karaté universitaire. Il comporte 34 sommets, représentant les membres d'un club de karaté, et 78 arêtes, représentant une certaine forme d'interaction entre les membres. Le club s'est séparé à la suite d'une dispute entre le directeur et le professeur du club, créant ainsi deux communautés.

Le jeu de données *Football* représente un réseau de matchs de football universitaire américain au cours de la saison d'automne 2000. Les sommets sont des équipes de football. Il y a une arête lorsque deux équipes ont joué l'une contre l'autre. Les sommets sont identifiés par la division dans laquelle les équipes jouent. Les conférences sont les communautés [53]. Ce graphe contient 115 sommets, 613 arêtes et 12 communautés.

Dans le jeu de données *Polbook*, les sommets sont des livres sur la politique vendus sur Amazon.com, et les arêtes représentent le co-achat fréquent de deux livres [83]. Les livres sont étiquetés comme *conservateur*, *libéral* ou *neutre*, formant ainsi des communautés. Cet ensemble de données compte 105 sommets et 441 arêtes.

Des graphes plus grands ont été publiés par Yang et Leskovec [140] sur la base de données *SNAP* [90]. Ces graphes vont de 1 000 sommets à 65 millions de sommets, ce qui permet de tester les algorithmes sur des tailles de graphes très variées.

2.5.3.6 Discussion

Parmi ces modèles, le modèle LFR semble le plus pertinent dans notre cas grâce à son paramètre de mélange, et ses bonnes performances. Le modèle ABCD est aussi intéressant, de par sa simplicité et son plus petit nombre de paramètres, ce qui est un atout dans le cadre d'études plus théoriques.

2.6 Conclusion

Notre étude de la mobilité humaine s'articule autour de trois axes. En chapitre 3, nous nous intéressons au placement de capteurs dans le but de mesurer la mobilité, à l'aide des concepts de couverture par sommets et de mesures de centralité. Ensuite le chapitre 4 est consacré à la détection de communautés dans le but de comprendre comment s'organise le réseau routier et ses zones fonctionnelles, grâce aux concepts de détection de communautés, de mesures de qualité, et de modèles de graphes aléatoires. Enfin, en chapitre 5, nous explorons une autre définition de communautés que nous pensons plus adaptée à notre contexte, et l'étudions grâce aux mesures de similarité de sommets.

Chapitre 3

Placement de capteurs

Sommaire

Dans le cadre de notre étude de la mobilité humaine, nous souhaitons mesurer des trajectoires humaines dans une zone urbaine.

Nous cherchons à réaliser une collecte de données passive, et non-intrusive, de manière à éviter tout biais lié à la sélection des participants. Nous choisissons de déployer des capteurs permettant de détecter (et de différencier) les individus. Ces capteurs devront être placés à certaines intersections bien choisies de la zone que nous étudierons. Pour ce faire, nous avons besoin de techniques d'optimisation pour placer nos capteurs de manière efficace. Nous présentons donc une heuristique de placement de capteurs, basée sur des notions de théorie des graphes, comme le problème de la couverture par sommets, ainsi que des mesures de centralité.

Ce chapitre commence par une introduction qui présente les enjeux de la mesure de trajectoires, suivi d'un rapide tour d'horizon des différentes méthodes de mesure de la mobilité humaine. Ensuite nous présenterons la manière dont nous modélisons notre problème, puis nous proposerons une manière de le résoudre. Enfin, nous présenterons les résultats de nos expériences sur des jeux de données réels.

3.1 Introduction

Les trajectoires d'individus dans une zone urbaine représentent une source d'information précieuse pour l'aménagement urbain. De plus, la connaissance des trajectoires permet de déduire les contacts entre les individus. La transmission d'information, ou de maladie, a lieu quand des individus sont en contact. Ainsi, l'étude de ces contacts peut permettre une meilleure compréhension de domaines tels que la propagation de maladies, d'information, ou encore une meilleure compréhension des réseaux opportunistes, qui sont des réseaux dans lesquels l'information transite directement d'appareil à appareil.

Le manque de jeux de données concernant la mobilité rend l'étude de la transmission d'information difficile. Des jeux de données *actifs* sont disponibles [25]. Dans ces jeux de données, les participants, choisissent de signaler activement leur localisation. Néanmoins, la précision des technologies comme le Géo-Positionnement par Satellite (GPS) est insuffisante pour déduire des contacts de manière fiable. À cause d'imprécisions de mesures, un individu peut se trouver dans une rue différente de la rue mesurée par GPS. D'autres jeux de données, plus précis, ne sont disponibles qu'à une échelle réduite, comme à l'intérieur de bâtiments [2]. D'autres jeux de données utilisent une approche *intrusive*, où seuls les participants ayant fait la démarche de participer à l'étude sont inclus dans le jeu de données. Ce peut être au travers d'une application installée sur leur téléphone, ou à l'aide d'une balise GPS que les participants portent sur eux. Il s'agit en pratique d'étudiants d'un département d'informatique, ou de participants à une conférence [66]. Ceci crée nécessairement un biais dans les données car les participants ne représentent pas un échantillon représentatif de la population, ce qui ne permet pas de tirer des conclusions valides pour l'ensemble de la population. De plus, aucune de ces solutions n'est utilisable à grande échelle, ce pourquoi nous ne disposons que de jeux de données à petite échelle.

Nous choisissons d'étudier la mobilité de manière *passive* et *non-intrusive*, c'est-à-dire que les participants n'ont pas besoin de signaler leur localisation eux-mêmes. Ceci permettra de collecter des données à plus grande échelle, de cibler un échantillon représentatif de la population, tout en étant plus précis. Nous choisissons de travailler avec

des capteurs, qui peuvent détecter quand un participant passe à proximité. Ces capteurs pourront être déployés dans une zone urbaine, pour y étudier la mobilité. Nous ne pourrons pas entièrement couvrir la zone à étudier, car le coût de déploiement serait prohibitif, ou la zone trop petite pour être pertinente. Nous devons donc développer des techniques d'optimisation pour placer nos capteurs. Pour ce faire, nous présentons ici une technique basée sur des notions de théorie des graphes, comme le *problème de la couverture par sommets*, ainsi que des mesures de *centralité*. Le problème de la couverture par sommets permet de garantir un déploiement des capteurs uniforme sur toute la zone à étudier en sélectionnant un ensemble d'intersections candidates. Les mesures de centralité permettront ensuite de sélectionner, parmi ces intersections candidates, celles sur lesquelles des capteurs seront déployés.

Pour évaluer notre solution, nous pouvons récupérer la carte d'une zone pour laquelle certaines trajectoires d'individus sont connues. Sur cette carte, nous simulerons un déploiement de capteurs, puis nous pourrons mesurer la performance de notre heuristique de placement de capteurs en calculant la proportion de trajectoires « observées » par nos capteurs. Nos expériences montrent que sur un jeu de données réaliste, en déployant des capteurs sur 20% des intersections de notre zone urbaine à étudier, nous sommes capables d'« observer » environ 95% des trajectoires en suffisamment d'endroits pour pouvoir par la suite les reconstruire avec une précision suffisante. Pour illustrer la taille d'une zone que nous pourrions étudier, un exemple de déploiement de capteurs est illustré en figure 3.1. Dans cette figure, les points rouges représentent des intersections où des capteurs sont déployés.

Nous commençons par passer en revue quelques manières de mesurer la mobilité humaine dans la Section 3.2, puis nous présentons notre modèle en Section 3.3. Nous présentons ensuite notre heuristique de placement de capteurs en Section 3.4, suivie de nos résultats en Section 3.5.

3.2 Vers une compréhension de la mobilité humaine

Dans la littérature, plusieurs études ont utilisé des appareils portables pour mesurer la mobilité humaine. Nous en présentons quelques-unes, ainsi que leurs limites.

La mobilité a été mesurée au travers d'applications Android installées sur des téléphones portables [25], ou des traceurs GPS [66], dont le but est de signaler la localisation du participant. Ces jeux de données sont nécessairement restreints par le nombre de personnes participant à l'étude, et ne permettent pas de tirer de conclusions générales sur la population à cause de l'échantillon de participants non représentatif de la population générale.

K. Akkaya *et al.* ont utilisé les réseaux Wi-Fi existants à l'intérieur de bâtiments pour déterminer quelles salles étaient occupées, en se basant sur la force du signal reçu, et les adresses physiques (adresses MAC) [2]. Cette approche n'est pas simple à généraliser à l'extérieur des bâtiments dans des zones urbaines car les réseaux Wi-Fi publics ne sont pas disponibles en quantité suffisante. Un autre problème est que cette approche permet de mesurer uniquement la mobilité des individus connectés au réseau Wi-Fi de l'étude.

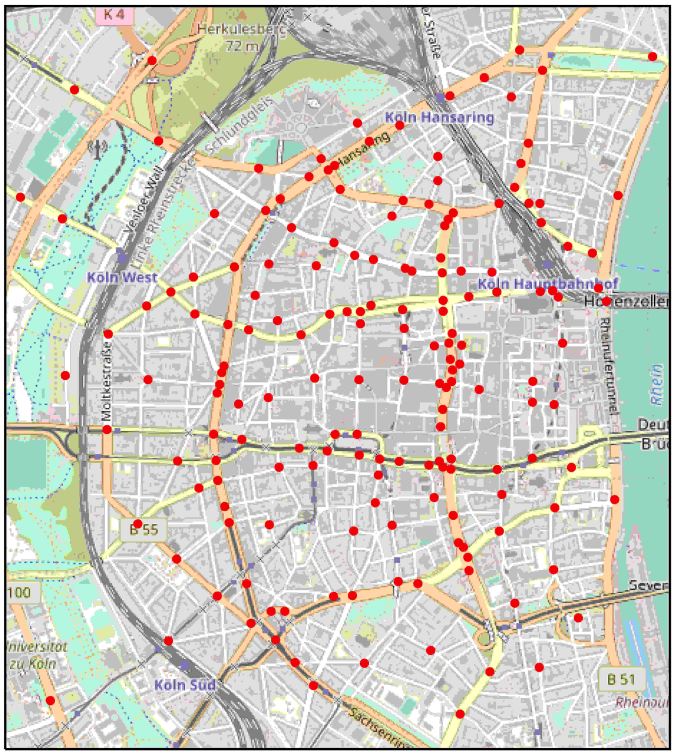


FIGURE 3.1 – Carte du centre-ville de Cologne, avec un exemple de déploiement de capteurs

Akhter *et. al* ont réalisé une étude comparative des systèmes de comptage des piétons [1]. La plupart sont attachés à des poteaux électriques et communiquent au travers de technologies sans fils. Dans notre cas, nous n'avons pas besoin d'établir une connexion entre nos capteurs puisque nous ne souhaitons pas mesurer la mobilité en temps réel. Nous nous contentons de stocker les données, puis de les récupérer manuellement à la fin de la mesure. Ces mêmes auteurs ont aussi présenté une méthode pour compter les piétons en utilisant des lentilles de Fresnel. Ces lentilles utilisent la technologie infrarouge pour compter les passants. La technologie utilisée dans cette étude ne permet pas de différencier les piétons, ce qui rend la reconstruction de trajectoires ainsi que l'inférence de contacts impossibles. De plus, la stratégie de placement de ces capteurs n'est pas discutée.

Le placement de capteurs a été étudié dans plusieurs études [70, 12], mais à de petites échelles seulement. Ces articles étudient l'impact de la position des capteurs sur la précision de la mesure, laquelle est calculée grâce à la force du signal reçu. D'autres études [32, 55] utilisent la technologie Wi-Fi pour étudier la densité et la vitesse du trafic routier. Ces études sont conduites à grande échelle, mais la stratégie de placement de capteurs n'est pas non plus discutée.

Les systèmes de localisation par Wi-Fi (Wi-Fi Positioning Systems, ou WPS), sont des systèmes de géolocalisation basés sur la technologie Wi-Fi. Ils tirent parti de caractéristiques comme la force du signal reçu, l'identification des appareils connectés au réseau, l'angle d'arrivée, ou encore la latence [92]. Des algorithmes basés sur ces critères permettent de localiser les appareils connectés à ces réseaux [139].

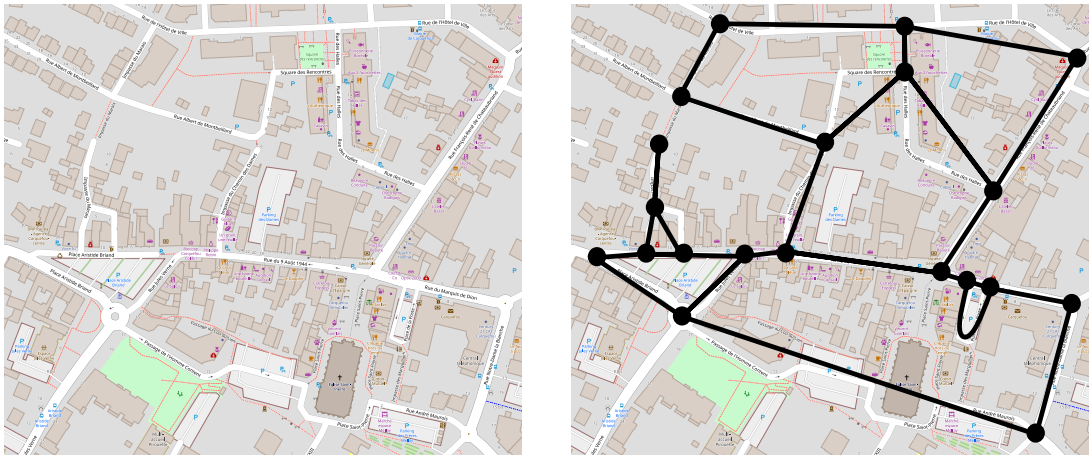
Nous souhaitons donc mesurer la mobilité humaine dans des zones urbaines, de manière à reconstruire les trajectoires des participants, à partir de mesures passives, pour ensuite faire de l'inférence de contacts. Notre but étant aussi d'éviter tout biais social, nous proposons d'utiliser des capteurs pour réaliser ces mesures passives. Nous étudions donc une technique d'optimisation du placement de ces capteurs.

3.3 Modélisation

3.3.1 Réseau routier

Un graphe $G = (V, E)$ est constitué d'un ensemble de sommets V, et d'un ensemble d'arêtes $E \subseteq V \times V$. Un graphe peut représenter des données variées, comme des réseaux routiers, dans lesquels les arêtes sont les tronçons de rue et les sommets sont les croisements de rues. Déployer des capteurs à des croisements de rues revient donc à sélectionner certains sommets de notre graphe représentant un réseau routier. La figure 3.2 montre un exemple de graphe routier à côté de la carte correspondante.

L'utilisation d'arêtes non pondérées ne permet pas toujours d'appréhender la mobilité humaine avec précision, car une arête peut représenter des tronçons de rues de différentes longueurs. Puisque notre graphe routier représente des données réelles, il est possible de l'enrichir d'informations supplémentaires telles que les coordonnées GPS de nos sommets. Cela nous permet de retrouver le plongement réel de nos graphes, et donc de calculer les



(a) Exemple de carte (b) Un graphe routier correspondant

FIGURE 3.2 – Une partie du réseau routier de Cologne (à gauche), ainsi qu'un graphe routier qui le modélise (à droite).

distances à vol d'oiseau entre les sommets

La carte de la zone à étudier peut être récupérée sur OpenStreetMap (OSM) et facilement convertie en un graphe comme sur la figure 3.2. Il s'agit ici d'un exemple simplifié, d'autres sommets pourraient être ajoutés, ainsi que des boucles. Cette représentation n'est pas unique. Travailler avec un graphe nous permet de tirer parti d'un large éventail d'algorithmes et de propriétés déjà connues sur les graphes.

3.3.2 Problème

Notre objectif est de sélectionner le moins de sommets possible du graphe routier, tout en maximisant le nombre de trajectoires vues par les capteurs, ainsi que le nombre de fois qu'elles sont vues. Nos capteurs ont une portée connue, exprimée en mètres. Nous pouvons supposer que l'activité sans fil des appareils se trouvant à une distance inférieure à la portée du capteur sera capturée. Pour cette raison, nous choisissons de pondérer les arêtes de notre graphe routier par la distance à vol d'oiseau entre les deux extrémités de chaque arête.

Nous parlons ici de sommets d'un graphe routier plutôt que de capteurs, car bien que nous ayons l'intention de déployer des capteurs, l'emplacement des capteurs dans une intersection de rues est un autre problème, car il faut tenir compte de la topologie de l'intersection. Un carrefour peut nécessiter plusieurs capteurs s'il est trop grand ou s'il y a trop de trafic. Ces contraintes doivent être prises en compte pour le déploiement physique des capteurs. Ici, nous nous limitons à la vue macroscopique du déploiement des capteurs en sélectionnant uniquement des sommets dans un graphe.

3.4 Placement de capteurs

Comme mentionné précédemment, la stratégie que nous adoptons pour le placement de capteurs consiste à d'abord calculer une couverture par sommets, puis à utiliser des mesures de centralité pour sélectionner les sommets les plus pertinents. Notons que par définition de la couverture par sommets, toutes les arêtes ont au moins une de leurs extrémités dans la couverture, par conséquent, toutes les trajectoires (de longueur au moins un) doivent passer par la couverture. Nous essayons ensuite d'obtenir un bon compromis entre le nombre de capteurs et la quantité de trajectoires observées, en sélectionnant les sommets pertinents en fonction de leur centralité.

Notre heuristique est basée sur la notion de couverture par sommets, que nous calculons avec l'algorithme Bar-Yehuda, comme expliqué dans l'heuristique 5. Plus précisément, Bar-Yehuda renvoie une couverture de sommets VC, puis le sommet le plus central u de VC reçoit un capteur, tandis que les sommets proches de u sont retirés de VC. Ce processus est répété jusqu'à ce que VC soit vide.

Dans cette heuristique, centralité renvoie un vecteur de centralités pour tous les sommets du graphe. En d'autres termes, chaque sommet se voit attribuer une valeur allant de 0 à 1. Différentes centralités peuvent être utilisées, comme celles présentées dans la Sous-Section 2.4.1. Nous verrons par la suite, en Sous-Section 3.5.7, une manière de nous abstraire de ce choix.

À chaque fois qu'un capteur est placé sur un sommet u, tous les sommets de la couverture VC à distance inférieure ou égale à k de u sont supprimés de VC. Cette heuristique prend un entier k comme paramètre, qui correspond donc au *rayon de suppression*. Des valeurs de k élevées signifient qu'un plus grand nombre de sommets seront retirés à chaque fois qu'un capteur est déployé, ce qui signifie que moins d'intersections de rues seront surveillées, donc moins de capteurs nécessaires, mais plus de trajectoires non observées. Un équilibre doit être trouvé entre le ratio de trajectoires non observées, et la proportion d'intersections où des capteurs sont déployés.

La distance à laquelle les sommets sont retirés dépend non seulement de k, mais aussi de la valeur de centralité du sommet u (voir les lignes 7 et 8 de l'heuristique). Les sommets les plus centraux nécessiteront moins de suppressions car ils correspondent probablement à des endroits plus denses de notre zone à étudier, où de nombreuses trajectoires sont susceptibles d'être trouvées. Nos expériences montrent que le fait de surveiller plus étroitement les zones où il y a des sommets centraux nous permet de voir plus de trajectoires.

Enfin, le plongement réel du graphe est récupéré grâce à OSM, puis la distance métrique, à vol d'oiseau, entre deux sommets peut être calculée à l'aide de la distance du grand cercle (qui est la distance entre deux points sur une sphère). Ceci nous permet de ne pas placer les capteurs trop près les uns des autres : chaque fois qu'un capteur s est placé, nous évitons de placer un autre capteur à moins de 25 mètres en supprimant tous les sommets situés à moins de 25 mètres de s dans VC. Nous avons choisi 25 mètres car il s'agit d'une portée facilement atteignable dans un environnement ouvert, mais cette

valeur pourrait être facilement modifiée.

Lors d'un déploiement réel, il suffit de choisir la valeur de k en fonction du nombre de capteurs disponibles. k doit nous permettre d'utiliser tous nos capteurs, sans dépasser le nombre de capteurs disponibles. Tous les paramètres de cette heuristique peuvent donc être choisis sans problème par l'utilisateur.

Algorithme 5 Heuristique de placement de capteurs (G, k)

1: $VC \leftarrow$ BAR-YEHUDA(G)
2: $VC_k \leftarrow \emptyset$
3: vecteur_centr $\leftarrow$ CENTRALITÉ(G)
4: **tant que** $VC \neq \emptyset$ **faire**
5: Choisir le sommet de plus central u dans VC
6: $VC_k \leftarrow u$
7: $c \leftarrow 1 - \left(\frac{\text{index de } u \text{ dans } vecteur\_centr}{|V|}\right)$
8: Retirer de VC, u et tous les sommets à distance inférieure à ARRONDI(kc) de u
9: **fin tant que**
10: **retourne** VC_k

L'implémentation des algorithmes présentés dans ce document est disponible sur Software Heritage [65].

La figure 3.3 montre comment notre heuristique se comporte sur un exemple simple. Nous récupérons d'abord la carte depuis OpenStreetMap (figure 3.3a), puis nous construisons un graphe à partir de cette carte (figure 3.3b). Nous construisons ensuite une couverture par sommets avec l'algorithme de Bar-Yehuda et Even (algorithme 2), illustré en rouge en figure 3.3d. Le sommet le plus central est choisi (sommet en bleu en figure 3.3e, choisi arbitrairement en cas d'égalité), puis tous les sommets à distance inférieure à $k = 2$ sont retirés de la couverture (figure 3.3f). Nous répétons l'opération jusqu'à ce que la couverture soit vide (jusqu'à ce qu'il n'y ait plus de sommets rouges, comme en figure 3.3i).

Notons que dans cet exemple, les sommets situés à distance inférieure à $k = 2$ sont supprimés, alors qu'en réalité cette distance varie en fonction de la valeur de centralité et de k (voir la ligne 7 de l'algorithme 5). Nous n'avons pas pu montrer cette opération sur un exemple de taille réduite car ces variations sont visibles sur des graphes de grande taille uniquement.

Pour finir, à partir de l'ensemble des sommets sur lesquels se trouve un capteur, nous calculons l'ensemble des sommets qui se trouvent à moins de 25 mètres d'un capteur. Nous supposons ensuite que toute trajectoire passant par cet ensemble de sommets peut être observée par un capteur.

3.5 Expériences sur des jeux de données réels

Pour mesurer la qualité de notre heuristique, nous choisissons de l'exécuter sur quelques jeux de données.

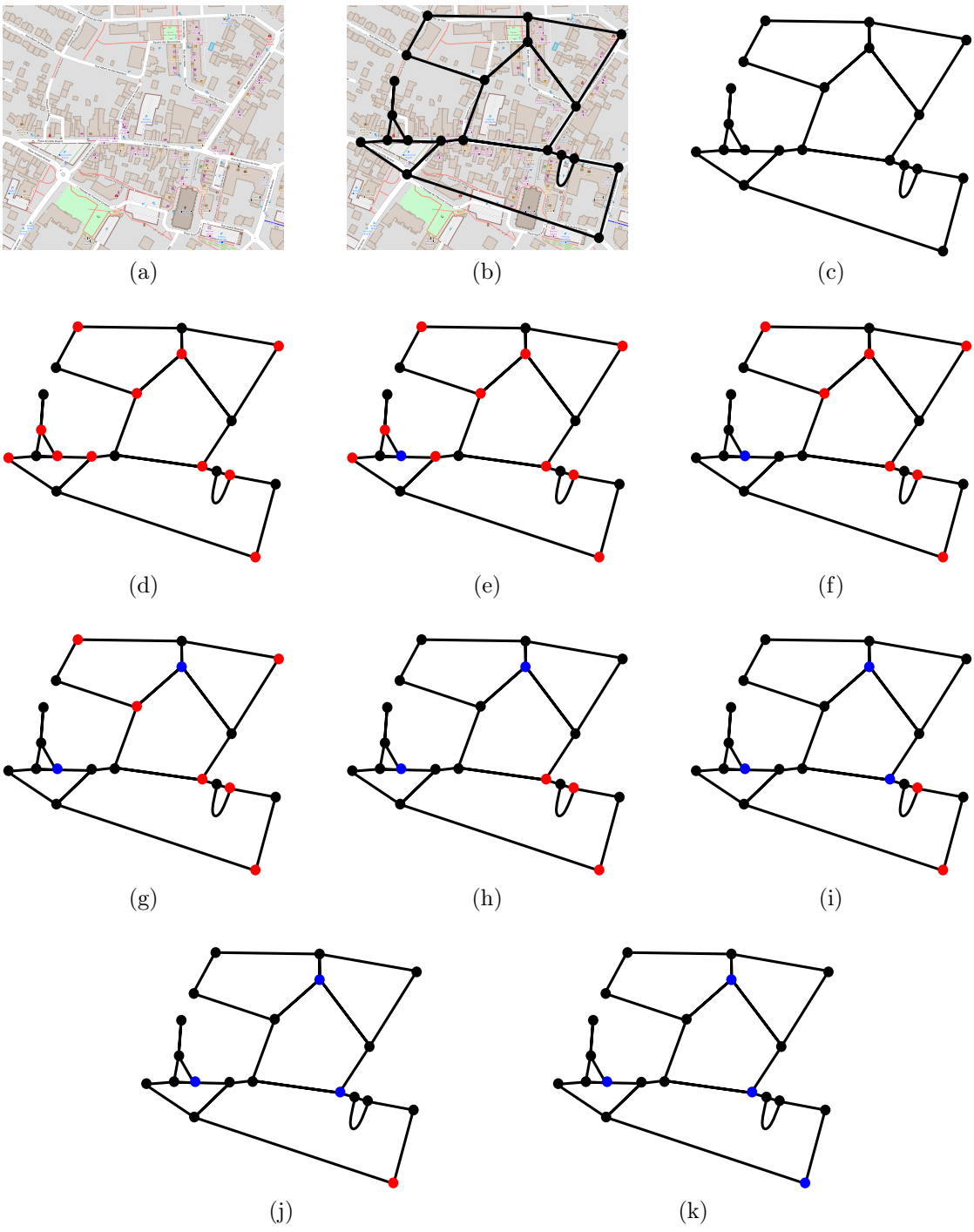


FIGURE 3.3 – Heuristique 5 sur un exemple de petite taille

3.5.1 Jeux de données

Les tableaux 3.1 et 3.2 présentent une vue d'ensemble de jeux de données sur la mobilité. Ces jeux de données contiennent les trajectoires d'individus dans des zones urbaines. Certains ne couvrent qu'une seule journée, quand d'autres peuvent couvrir plusieurs années. Certains n'incluent que quelques personnes, tandis que le plus grand contient presque 200 000 personnes. La plupart des jeux de données reposent sur la technologie GPS, ce qui fait que les mesures ne sont pas toujours suffisamment précises. Nous avons choisi de ne pas utiliser les jeux de données qui reposent sur des mesures, car nous devrions faire correspondre les points de données aux rues réelles. Cette tâche, appelée *cartospondance* (ou *map matching* en anglais), est loin d'être triviale. Notre objectif n'est pas d'effectuer une telle tâche, c'est pourquoi nous avons choisi d'utiliser des jeux de données simulés tels que TAPAS Cologne ou Bologne. La plupart des jeux de données sont répertoriés sur la documentation d'Accio [67].

L'un des objectifs finaux étant de déployer des capteurs à La Rochelle, nous avons également récupéré la carte de La Rochelle sur OpenStreetMap. La zone correspond à ce qui est affiché dans la figure 3.4. Notons que nous n'avons pas de trajectoires pour ce jeu de données, nous n'effectuerons donc que quelques expériences préliminaires sur celui-ci.

Nom	Collecté par	Nombre de personnes	Durée	Quantité de coordonnées	Zone	Notes
Geolife [144]	Microsoft Research Asia	182 personnes	Avril 2007 à août 2012	25 000 000	Pékin	
UCI Go !Track [36]	Application Android Go !Track	40 trajectoires	Septembre 2014 à janvier 2016	5 317		Trajectoires de bus et voitures
Taxi Service Trajectory [100]	Chercheurs de l'université de Porto	442 taxis	Juin 2013 à juin 2014	1 710 671	Porto	
Cab-spotting [34]	Chercheurs à l'EPFL	536 taxis	17 mai 2008 au 10 juin 2008	11 000 000	San Francisco	Inscription nécessaire au téléchargement
MDC [81, 88]	Lausanne Data Collection Campaign	182 personnes	2009 – 2001	11 000 000	Lac Léman	Coordonnées GPS de téléphones

TABLE 3.1 – Liste de jeux de données de mobilité

Nom	Collecté par	Nombre de personnes	Durée	Quantité de coordonnées	Zone	Notes
T-Drive [143]	Micrisoft Research	10 357 taxis	Une semaine	15 000 000	Pékin	
Brightkite [31]	Brightkite (réseau social)	58 228 personnes	Un an et demi	4 000 000	Mondial	Données check-in
Gowalla [31]	Gowalla (réseau social)	196 591 personnes	Un an et demi	6 000 000	Mondial	Données check-in
TAPAS Cologne [131, 130, 117]	TAPAS Cologne project		24h	1 538 464	Cologne, voir figure 3.6a	Données simulées
Bologna [68]	iTETRIS project			11 000	Quelques rues, voir figure 3.10	Données simulées

TABLE 3.2 – Liste de jeux de données de mobilité

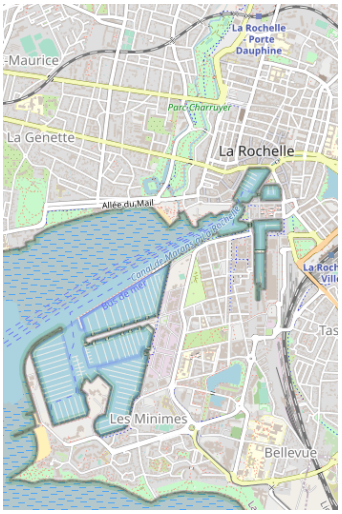


FIGURE 3.4 – Carte de la zone de La Rochelle

3.5.1.1 TAPAS Cologne

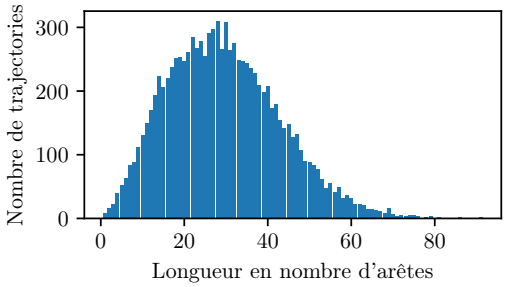
Nous avons choisi de travailler avec les données fournies par le projet TAPAS Cologne, qui est une initiative de l'Institut des Systèmes de Transport du Centre Allemand pour l'Aéronautique et l'Astronautique (Institute of Transportation System au Deutsches Zentrum für Luft- und Raumfahrt, ou ITS-DLR), visant à reproduire, avec le plus haut niveau de réalisme possible, le trafic automobile dans l'agglomération de la ville de Cologne, en Allemagne.

Ces données couvrent une région de 400 kilomètres carrés pendant une période de 24 heures, représentant une journée de travail classique, et incluent 1 538 464 trajectoires [131, 130]. Le jeu de données a été simulé avec le logiciel Simulation of Urban Mobility (SUMO) [27] et est disponible en ligne [117].

La carte de l'agglomération de Cologne, utilisée par TAPAS Cologne, provient de la base de données OpenStreetMap. Seule la plus grande composante connexe a été retenue (31 542 sommets au lieu de 31 614 sommets pour l'ensemble du graphe), car nous avons estimé qu'il n'était pas logique qu'une ville comme Cologne dispose d'un réseau routier déconnecté. Il s'est avéré qu'aucune trajectoire ne passe par les 72 sommets qui ont été retirés, ce qui n'a donc pas entraîné une grande perte.

La carte et les trajectoires ont été extraites à l'aide d'un outil qui convertit la carte en un graphe et les trajectoires en une liste ordonnée d'arêtes. Cet outil est disponible en ligne [62].

Étant donné que la zone incluse dans le projet TAPAS Cologne (illustrée dans la figure 3.6a) est bien plus grande que les déploiements que nous pourrions raisonnablement effectuer, et qu'elle n'est pas homogène en raison de ses différentes zones (urbaines, non urbaines, résidentielles), nous choisissons de nous limiter au centre-ville de Cologne. Nous réduisons donc les données à une zone de quatorze kilomètres carrés, correspondant à ce



(a) Distribution des longueurs, en nombre de tronçons de rues, des trajectoires du centre-ville de Cologne

(b) Carte de chaleur représentant les endroits où les trajectoires se trouvent sur la carte de Cologne

FIGURE 3.5 – Données relatives aux trajectoires du jeu de données de Cologne

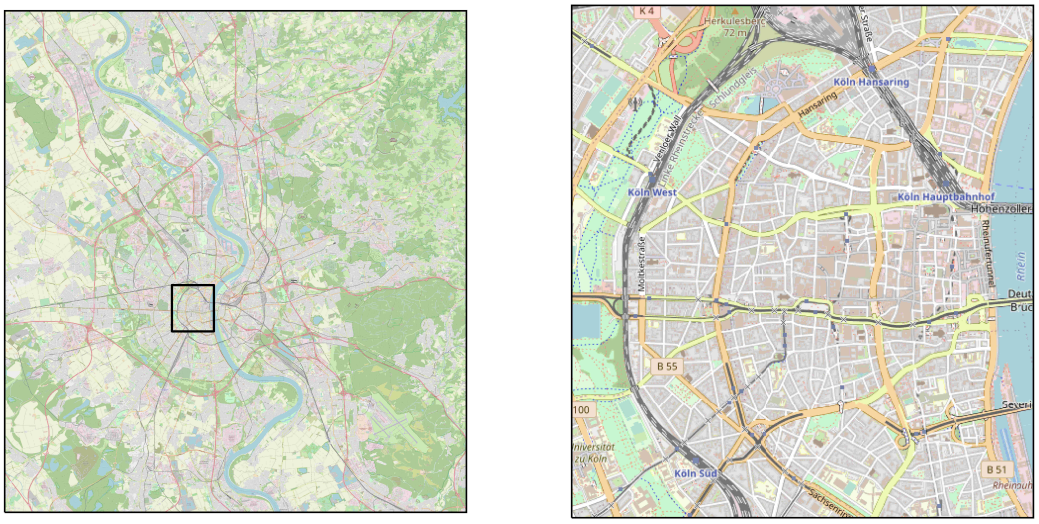
qui est montré dans la figure 3.6b.

Il en résulte un jeu de données contenant un graphe routier avec 1080 sommets et 1615 arêtes, ainsi que 51 695 trajectoires, qui ne quittent jamais ce graphe correspondant au centre-ville. La distribution des longueurs est présentée dans la figure 3.5a. La longueur moyenne des trajectoires est d'environ 30 tronçons de rues.

Un autre outil de visualisation a été développé pour voir où vont les trajectoires. Il produit une carte de chaleur, où la zone est divisée en carrés, qui sont colorés en fonction du nombre de trajectoires passant par ce carré. Une telle carte de chaleur est présentée dans la figure 3.5b, où les zones bleues correspondent à des zones où il y a peu de trajectoires, et les zones jaunes puis orange, à des zones où il y a plus de trajectoires.

3.5.1.2 Distribution des centralités dans Cologne

Les figures 3.7 et 3.8 montrent la distribution de chaque centralité, sur le jeu de données de Cologne. Nous calculons la centralité pour chaque sommet du graphe routier et traçons cette distribution. Toutes les centralités sont ensuite normalisées afin d'obtenir des valeurs comprises dans l'intervalle $[0, 1]$, pour pouvoir mieux les comparer. La centralité dans la figure 3.8a est la distribution des trajectoires, c'est-à-dire le nombre de trajectoires passant par chaque sommet. Intuitivement, nous nous attendons à obtenir



(a) Carte de la métropole de Cologne (b) Carte du centre-ville de Cologne

FIGURE 3.6 – Carte de la métropole de Cologne (gauche), ainsi que la carte du centre-ville de Cologne, sur lequel nous travaillons (droite)

de bons résultats avec des centralités aussi proches que possible de la figure 3.8a.

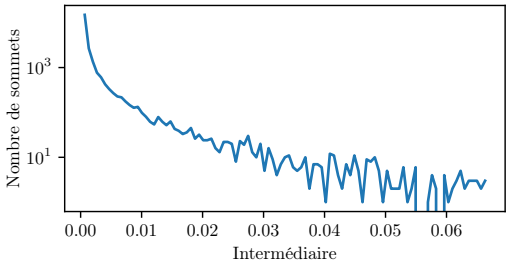
3.5.1.3 Bologne

Le jeu de données Bologne a été développé dans le cadre du projet iTRETRIS [68]. Il s'agit d'un jeu de données plus petit que celui concernant Cologne, qui ne concerne que deux rues principales de la ville de Bologne. Il simule les mouvements piétonniers denses observés autour de stades lors de grands événements tels que des matchs de football ou des concerts [13]. Le graphe routier de ce jeu de données contient 159 sommets, 215 arêtes et 11000 trajectoires. La figure 3.9 présente la distribution des longueurs des trajectoires du jeu de données Bologne, ainsi qu'une carte de chaleur présentée de la même manière que pour le jeu de données TAPAS Cologne. La figure 3.10 présente quant à elle la carte correspondant au jeu de données Bologne.

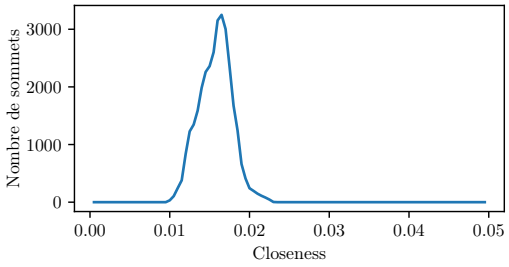
3.5.1.4 Méthodologie d'évaluation

Pour évaluer la qualité de notre heuristique sur ces jeux de données, nous avons besoin d'une mesure de qualité.

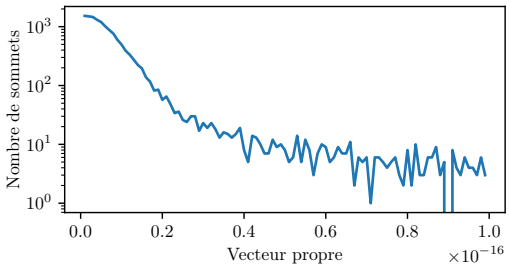
Nous devons déployer nos capteurs de manière à maximiser le nombre de trajectoires qu'il sera possible de reconstruire avec une précision suffisante. Nous considérons qu'une trajectoire peut être reconstruite si elle est observée suffisamment souvent. Dans ce qui suit, nous considérons qu'une trajectoire est « perdue » et ne peut pas être reconstruite,



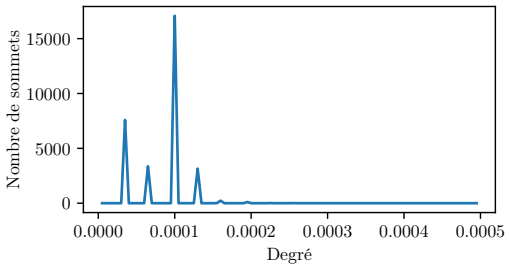
(a) Distribution des sommets en fonction de la centralité intermédiaire



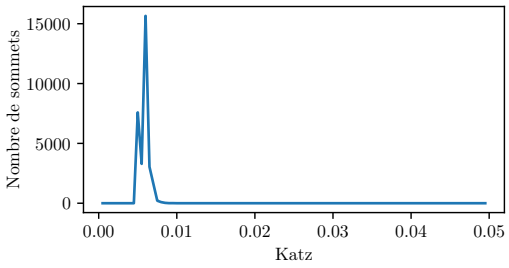
(b) Distribution des sommets en fonction de leur centralité closeness



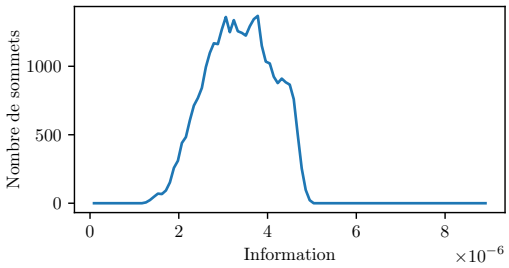
(c) Distribution des sommets en fonction de leur centralité vecteur propre



(d) Distribution des sommets en fonction de leur centralité degré

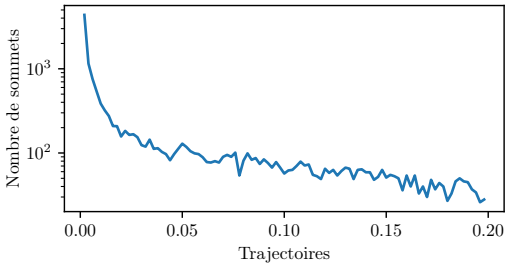


(e) Distribution des sommets en fonction de leur centralité Katz

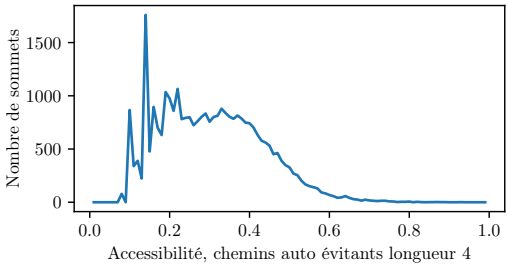


(f) Distribution des sommets en fonction de leur centralité information

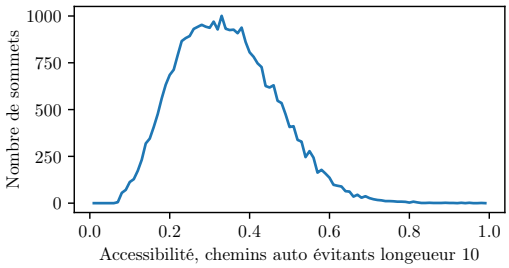
FIGURE 3.7 – Différentes distributions de centralité dans le centre-ville de Cologne



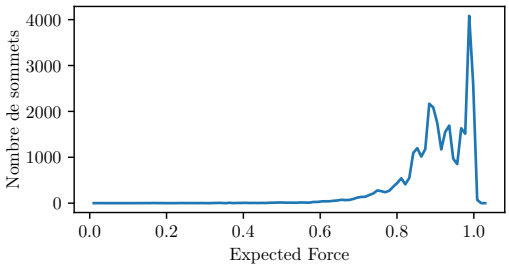
(a) Distribution des sommets en fonction de leur centralité trajectoires (nombre de trajectoire à passer par un sommet)



(b) Distribution des sommets en fonction de leur centralité accessibilité, avec longueur des chemins auto-évitants $h = 4$

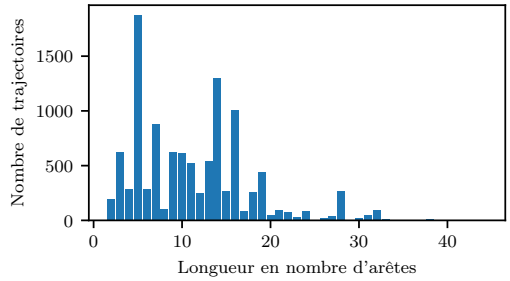


(c) Distribution des sommets en fonction de leur centralité accessibilité, avec $h = 10$



(d) Distribution des sommets en fonction de leur expected force

FIGURE 3.8 – Différentes distributions de centralité dans le centre-ville de Cologne



(a) Distribution des longueurs des trajectoires de Bologne, en nombre de tronçons de rues

(b) Carte de chaleur représentant où les trajectoires se trouvent sur la carte de Bologne

FIGURE 3.9 – Données relatives au jeu de données Bologne

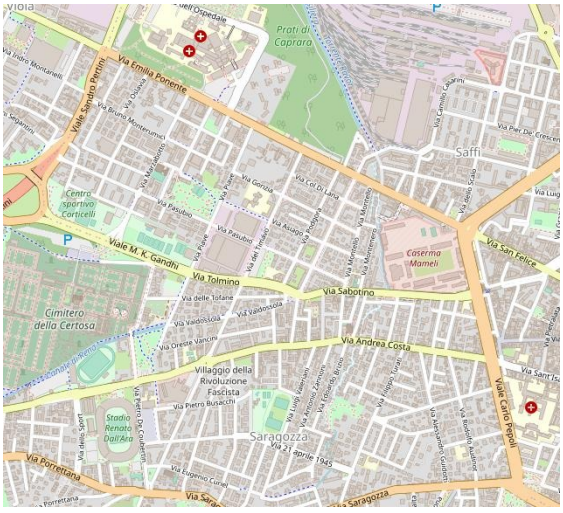


FIGURE 3.10 – Carte du jeu de données Bologne

en utilisant deux définitions : (i) lorsque moins de $x\%$ des sommets par lesquelles elle passe sont observés par des capteurs, ou (ii) lorsqu'elle traverse moins de y sommets où des capteurs ont été déployés. Dans la suite, nous utilisons les valeurs suivantes : $x = 20\%$, $x = 10\%$, $y = 4$. Nous supposons que de tels seuils pourraient permettre de reconstruire les trajectoires avec une bonne précision. D'autres études seraient nécessaires pour affiner ces valeurs.

3.5.1.5 Efficacité

Définissons maintenant une mesure d'efficacité. L'objectif est de minimiser le nombre d'intersections de rues que nous surveillons, c'est-à-dire d'utiliser le moins de capteurs possible, tout en minimisant le nombre de trajectoires perdues. Nous utilisons l'équation 3.1, où $|S|$ est le nombre de capteurs, $|V|$ le nombre de sommets de notre graphe, $|L|$ le nombre de trajectoires *perdues*, et T le nombre de trajectoires dans notre jeu de données. Cette fonction doit être maximisée. Cette fonction est comprise entre 0 et 1.

$$\text{Efficacité} = \frac{|V| - |S|}{|V|} \times \frac{|T| - |L|}{|T|} \tag{3.1}$$

La valeur k influe sur la distance en dessous de laquelle les sommets sont éliminés de la couverture. Lorsque k augmente, le nombre de capteurs diminue, donc le nombre de trajectoires perdues augmente. k doit être choisi de manière à maximiser l'efficacité, comme le montre la figure 3.11. On peut y voir que pour de faibles valeurs de k, il y a beaucoup de capteurs, puisque la totalité des sommets de la couverture possède un capteur. Le pourcentage de trajectoires perdues est par conséquent très faible, mais le coût de déploiement serait prohibitif. Plus k augmente, plus les capteurs seront éloignés les uns des autres, donc moins il y en aura. Par conséquent, le coût de déploiement diminuera mais le pourcentage de trajectoires perdues augmentera.

Avec un tel jeu de données, choisir la valeur de k qui maximise l'efficacité est facile à faire puisque notre jeu de données inclut les trajectoires. Dans un déploiement réel, la courbe de la figure 3.11 serait inconnue. Nous pourrions toutefois choisir k en fonction des ressources disponibles : k doit être le plus petit possible afin de reconstruire autant de trajectoires que possible, sans toutefois dépasser le nombre de capteurs disponibles.

3.5.1.6 Méthodes de référence

Pour mettre nos résultats en perspective, puisque nous n'avons pas trouvé d'autres méthodes de placement de capteurs dans la littérature, nous avons défini plusieurs autres méthodes pour évaluer la qualité de nos résultats.

La première méthode consiste à sélectionner des sommets au hasard. Nous choisissons le nombre de sommets qui nous donne les meilleurs résultats en moyenne sur 10 exécutions. Comme nous choisissons les sommets au hasard, ils doivent être répartis uniformément sur notre graphe. Nous appelons cette stratégie « Aléatoire ». Nous nous attendons à ce que cette méthode donne de mauvais résultats.

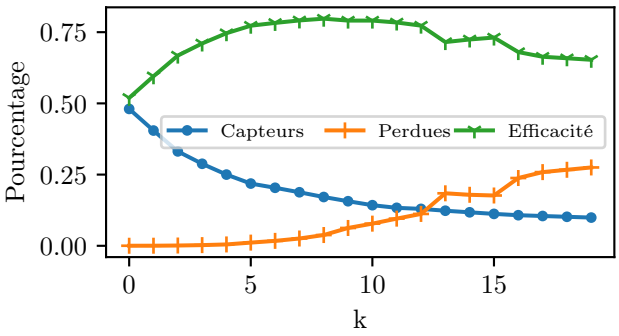


FIGURE 3.11 – Efficacité en fonction de k, sur le jeu de données Cologne, avec la centralité expected force

Nous avons utilisé la centralité *force* en pondérant le graphe de la manière suivante : le poids de chaque arête est le nombre de trajectoires qui passent par cette arête. La centralité *force* d'un sommet u est la somme du poids de chacune des arêtes adjacentes à u. Cette centralité nécessite donc une connaissance *a priori* des trajectoires et devrait être plus performante que les autres centralités.

Nous avons également implémenté quatre stratégies gloutonnes qui utilisent également la connaissance *a priori* des trajectoires. « Glouton perdues » choisit, à chaque étape, le sommet sans capteur par lequel passe le plus grand nombre de trajectoires *perdues*, jusqu'à ce que l sommets soient choisis. Nous choisissons la valeur de l qui maximise notre fonction d'efficacité. « Glouton aucune perdue » fait la même chose mais s'arrête lorsqu'il n'y a plus de trajectoires *perdues*. « Glouton traj » choisit, à chaque étape, le sommet sans capteurs par lequel passe le plus grand nombre de trajectoires, jusqu'à ce que h sommets soient choisis. Enfin, « Glouton aucune trajectoire » est identique mais s'arrête lorsque plus aucune trajectoire n'est *perdue*.

3.5.2 Résultats

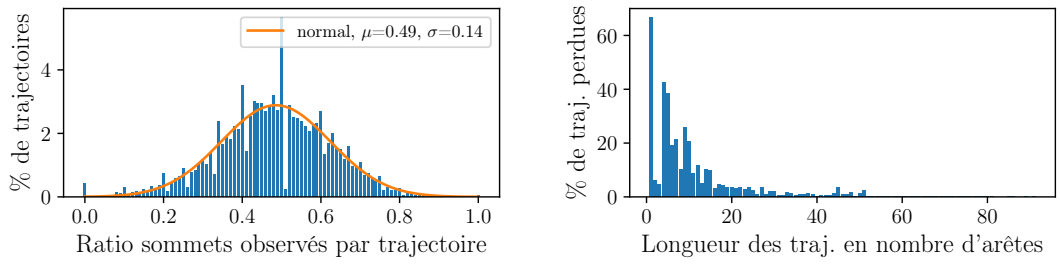
3.5.2.1 Jeu de données centre-ville Cologne

Nous exécutons notre heuristique sur le graphe routier du centre-ville de Cologne, avec différentes mesures de centralité afin de comparer leur efficacité.

Nos résultats sont résumés dans le tableau 3.3. La valeur k est choisie de manière à maximiser la fonction d'efficacité, comme expliqué dans la Section 3.5.1.5.

Perdues	20%				10%				4			
Centralité	k	Captr.	Perd.	Effi.	k	Captr.	Perd.	Effi.	k	Captr.	Perd.	Effi.
Force	9	16,14%	4,10%	0,804	17	10,07%	4,14%	0,862	9	16,14%	8,84%	0,764
Glout. traj.		17,13%	7,77%	0,764		13,89%	5,54%	0,814		17,13%	10,03%	0,746
Glout. auc. traj.		86,02%	0%	0,140		82,50%	0%	0,172		86,67%	0%	0,133
Glout. perd.		16,76%	8,54%	0,761		10,83%	8,84%	0,813		17,41%	10,06%	0,743
Glout. auc. perd.		85,46%	0%	0,144		82,13%	0%	0,177		34,07%	0%	0,659
Degrée	**5**	**16,48%**	**4,54%**	**0,797**	7	13,80%	1,31%	0,851	4	20,56%	7,29%	0,736
Katz	8	18,15%	3,00%	0,794	**20**	**9,54%**	**4,39%**	**0,865**	8	18,15%	9,21%	0,743
Expected Force	9	17,13%	4,14%	0,794	**23**	**8,98%**	**5,00%**	**0,865**	9	17,13%	10,70%	0,740
Accessibilité	10	15,93%	5,88%	0,791	14	12,31%	2,50%	0,856	**9**	**17,31%**	**9,82%**	**0,756**
Intermédiaire	7	20,56%	3,88%	0,764	13	12,96%	3,26%	0,842	7	20,56%	9,02%	0,723
Information	6	20,37%	4,21%	0,763	14	11,02%	4,79%	0,847	10	14,63%	13,52%	0,738
Closeness	4	23,33%	7,10%	0,712	9	13,61%	8,35%	0,792	4	23,33%	10,96%	0,683
Vecteur propre	3	27,41%	3,39%	0,701	5	19,72%	3,19%	0,777	4	22,96%	11,32%	0,683
Aléatoire	286	26,48%	5,60%	0,696	204	18,89%	4,10%	0,780	258	23,89%	14,12%	0,654

TABLE 3.3 – Résultats pour chaque centralité, sur le centre-ville de Cologne, en considérant différentes définitions de *perdues* (moins de 10% des sommets, moins de 20% des sommets, et moins de 4 sommets). Les résultats en gras correspondent aux cas où l'efficacité est maximale.



(a) Pourcentage de trajectoires, en fonction du ratio de leurs sommets qui sont observés par des capteurs. Ex : 5% des trajectoires ont 50% de leurs sommets observés.

(b) Pourcentage de trajectoires perdues, en fonction de la longueur des trajectoires, jeu de données Cologne, centralité Force. Ex : 42% des trajectoires de longueur 4 sont perdues.

FIGURE 3.12 – Centre-ville de Cologne, *perdue* = 20%, centralité force, $k = 9$

La première colonne du tableau 3.3 considère qu'une trajectoire est perdue lorsque moins de 20% des intersections de rues qu'elle traverse ont des capteurs. Dans cette colonne, avec la plupart des centralités, moins de 20% des sommets doivent avoir un capteur pour pouvoir observer plus de 95% des trajectoires. Ceci est également visible sur la figure 3.12a, qui montre la proportion des trajectoires observées.

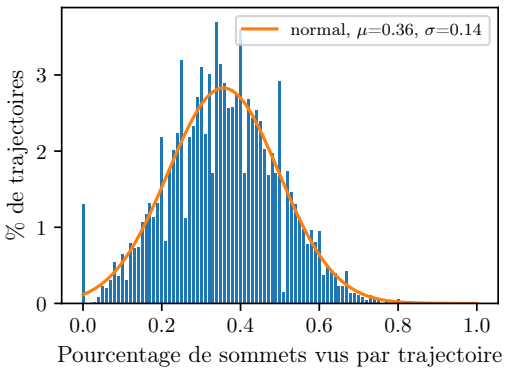
Une inspection plus approfondie montre que les trajectoires *perdues* sont principalement des trajectoires courtes. Ceci est confirmé par la figure 3.12b qui représente la quantité de trajectoires *perdues* en fonction de leur longueur. Nous pouvons voir que près de 40% des trajectoires d'une longueur de 4 sont perdues et que, plus généralement, plus les trajectoires sont courtes, plus il y a de pertes. Cette figure est à comparer avec la figure 3.5a, qui représente la même distribution pour toutes les trajectoires. Nous constatons une diminution de la longueur moyenne des trajectoires.

3.5.2.2 Impact de la définition d'une trajectoire *perdue*

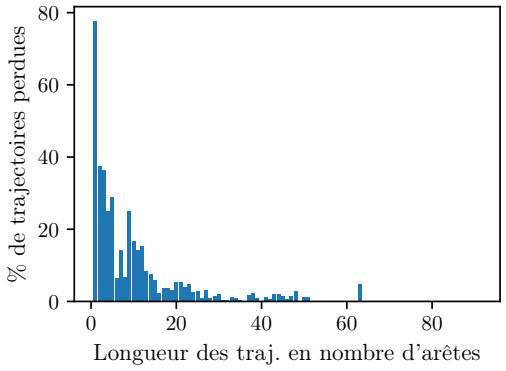
Pour mieux comprendre l'impact de la définition de trajectoires *perdues*, nous considérons également une trajectoire *perdue* lorsque moins de 10% des intersections de rues qu'elle traverse possèdent un capteur. La deuxième colonne du tableau 3.3 montre les résultats obtenus en tenant compte de cette nouvelle définition.

La valeur de k qui maximise notre fonction d'efficacité est plus élevée que dans la colonne 1 du tableau 3.3. Ceci n'est pas surprenant, car nous avons besoin de moins de sommets pour considérer une trajectoire « observée ». Une valeur plus élevée de k signifie qu'il y aura moins de capteurs. Notons que le pourcentage de trajectoires *perdues* reste du même ordre de grandeur. Avec moins de capteurs et le même nombre de trajectoires *perdues*, la fonction d'efficacité donne des valeurs plus élevées.

La figure 3.13b montre qu'encore une fois, les trajectoires courtes ont tendance à être plus perdues que les trajectoires plus longues. La figure 3.13a quant à elle montre qu'en



(a) Pourcentage de trajectoires perdues, en fonction de la longueur des trajectoires, jeu de données Cologne, centralité Force. Ex : 26% des trajectoires de longueur 9 sont perdues.



(b) Pourcentage de trajectoires, en fonction du ratio de leurs sommets qui sont observés par des capteurs. Ex : 3, 6% des trajectoires ont 40% de leurs sommets observés.

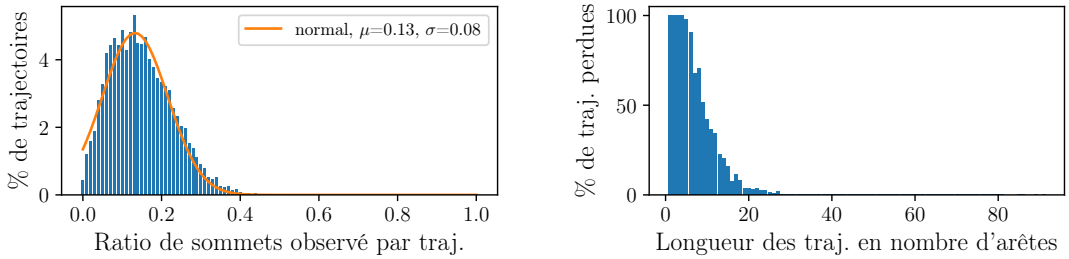
FIGURE 3.13 – Centre-ville de Cologne, *perdue* = 10, centralité force, $k = 17$

moyenne, avec cette valeur de k plus élevée, les trajectoires sont vues moins souvent : 36% de leurs sommets sont vus en moyenne, ce qui est inférieur à la figure 3.12a, dans laquelle environ 49% des sommets des trajectoires sont vues par un capteur.

Ceci s'explique par une valeur de k moins élevée dans la figure 3.12a, ce qui se traduit par plus de capteurs, donc moins de perte. Notons que k est moins élevé dans la figure 3.12a car nous demandons à voir plus de sommets par trajectoire pour les considérer *observées* (le seuil *perdue* est plus élevé).

Considérons maintenant qu'une trajectoire est *perdue* lorsque nous avons observé moins de 4 de ses intersections. La troisième colonne du tableau 3.3 montre comment nos résultats diffèrent selon cette définition. Comme le montre la figure 3.14b, les trajectoires courtes sont inévitablement *perdues*. Les trajectoires courtes sont *perdues* car il est difficile de voir 4 fois une trajectoire de longueur proche de 4. Nous constatons une forte diminution du pourcentage de trajectoires *perdues* lorsque la longueur de la trajectoire augmente. La valeur de k qui maximise notre fonction d'efficacité reste dans la même fourchette, avec un pourcentage plus élevé de trajectoires *perdues*, ce qui se traduit par des valeurs plus faibles pour notre fonction d'efficacité.

Notons qu'avec cette définition de trajectoires *perdues*, il est impossible d'observer une trajectoire qui comporte moins de 4 sommets. Dans le cas de nos approches « Glouton aucune trajectoire » et « Glouton aucune perdue », nous omettons les trajectoires qui comportent moins de 4 sommets.



(a) Pourcentage de trajectoires, en fonction du ratio de leurs sommets qui sont observés par des capteurs. Ex : $3,1\%$ des trajectoires sont observées dans 20% de leurs sommets.

(b) Pourcentage de trajectoires perdues, en fonction de la longueur des trajectoires, jeu de données Cologne, centralité Force. Ex : 100% des trajectoires de longueur 4 sont perdues.

FIGURE 3.14 – Centre-ville de Cologne, *perdue* $= 4$, centralité force, $k = 9$

3.5.2.3 Jeu de données Bologne

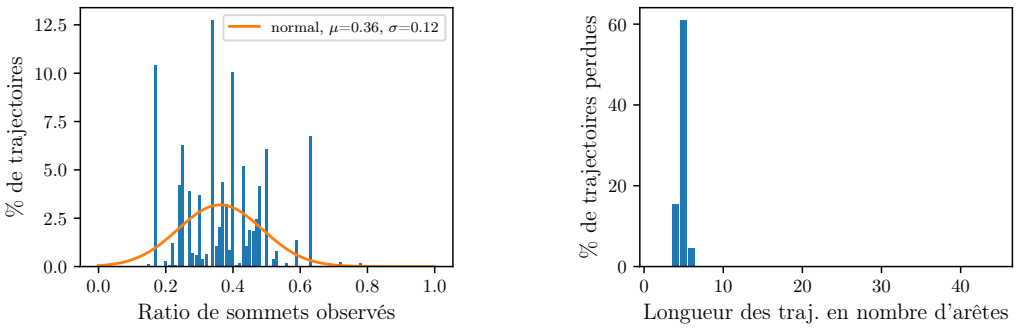
Le tableau 3.4 montre les résultats que nous avons obtenus pour le jeu de données Bologne, où des résultats comparables sont obtenus.

Les deux premières colonnes correspondent aux contextes où les trajectoires sont considérées *perdues* lorsque moins de 20% et 10% des croisements de rues qu'elles traversent ont un capteur. Dans ces colonnes, nous pouvons voir que l'efficacité est similaire à celle de Cologne, mais que les valeurs optimales de k sont plus petites. Cela signifie que nous retirons moins de sommets à chaque étape, donc que nous utilisons proportionnellement plus de capteurs. Cela est probablement dû à la taille du jeu de données : la zone est considérablement plus petite, d'un facteur 10 environ, mais elle comporte seulement 5 fois moins de trajectoires. Par conséquent, le fait de laisser plus de sommets non surveillés augmente le nombre de trajectoires *perdues*.

Dans la troisième colonne, nous observons de moins bons résultats, qui sont dus à la taille du jeu de données. La longueur moyenne des trajectoires à Cologne est de 30 (voir figure 3.5a) alors qu'elle est de 11 à Bologne. Il est donc plus difficile de voir 4 de leurs sommets.

Perdues	20%				10%				4			
Centralité	k	Captr.	Perd.	Effi.	k	Captr.	Perd.	Effi.	k	Captr.	Perd.	Effi.
Force	6	21,90%	2,40%	0,762	19	12,41%	1,50%	0,863	10	16,79%	43,28%	0,472
Glout. perd.		18,87%	9,05%	0,738		11,32%	8,03%	0,816		23,90%	27,84%	0,549
Glout. auc. perd.		45,28%	0%	0,547		40,25%	0%	0,597		28,93%	0%	0,711
Glout. traj.		18,87%	9,05%	0,738		11,32%	8,03%	0,816		23,90%	27,84%	0,549
Glout. auc. traj.		45,28%	0%	0,516		40,25%	0%	0,579		69,81%	0%	0,302
Degré	**4**	**26,75%**	**0,56%**	**0,728**	12	8,92%	3,82%	0,876	**3**	**29,30%**	**34,31%**	**0,464**
Katz	4	27,39%	0,48%	0,723	**19**	**9,55%**	**0,90%**	**0,896**	3	29,94%	42,54%	0,403
Expected Force	4	27,39%	0,56%	0,722	23	8,91%	6,04%	0,856	6	19,11%	57,53%	0,344
Intermédiaire	3	29,30%	0,75%	0,702	13	13,38%	2,22%	0,847	8	17,20%	60,68%	0,326
Accessibilité	3	30,57%	4,61%	0,662	5	22,93%	1,07%	0,762	3	30,57%	47,32%	0,366
Vecteur propre	3	28,66%	10,48%	0,639	6	18,47%	6,31%	0,764	1	40,13%	36,62%	0,379
Closeness	3	27,39%	12,87%	0,633	5	22,29%	5,11%	0,737	2	31,21%	48,15%	0,357
Information	3	28,03%	18,74%	0,585	6	20,38%	8,05%	0,732	2	35,67%	45,14%	0,353
Aléatoire	50	31,45%	10,61%	0,613	42	26,42%	6,42%	0,691	71	44,65%	33,30%	0,369

TABLE 3.4 – Résultats pour chaque centralité, sur Bologne, pour différentes définitions de trajectoires *perdues* (moins de 10% des sommets, moins de 20% des sommets, et moins de 4 sommets). Les résultats en gras correspondent aux cas où l'efficacité est maximale.



(a) Pourcentage de trajectoires, en fonction du ratio de leurs sommets qui sont observés par des capteurs. Ex : 10% des trajectoires ont 40% de leurs sommets observés.

(b) Pourcentage de trajectoires perdues, en fonction de la longueur des trajectoires, jeu de données Bologne, centralité Katz. Ex : 68,9% des trajectoires de longueur 10 sont perdues.

FIGURE 3.15 – Bologne, *perdue* = 20%, centralité degré, $k = 4$

La figure 3.15 montre plus de détails sur nos résultats. La figure 3.15a montre le pourcentage de trajectoires en fonction du ratio de leurs sommets observés par des capteurs. En moyenne, environ 30 à 40% des trajectoires sont observées par les capteurs. Nous avons fait une régression avec la même fonction que sur le jeu de données précédent (figure 3.12a), mais le jeu de données est beaucoup plus petit, donc il y a plus de variabilité dans les données. La figure 3.15b montre le pourcentage de trajectoires *perdues* en fonction de leur longueur. Une fois de plus, en la comparant à la figure 3.9a, nous constatons que les trajectoires courtes ont tendance a être perdues.

3.5.3 Impact de k

Pour voir l'impact de k sur le nombre de capteurs, nous traçons le nombre de capteurs en fonction de k, représenté dans la figure 3.16. Pour cette expérience, nous prenons comme exemple un graphe routier récupéré sur OSM, qui couvre quelques quartiers de La Rochelle (voir figure 3.4). Nous observons une forte diminution du nombre de capteurs déployés pour des valeurs faibles de k, et une diminution plus lente pour des valeurs élevées de k. Cela indique qu'un compromis peut être trouvé parmi les faibles valeurs de k.

3.5.4 Recalcul des centralités

Nous avons essayé de recalculer les centralités après chaque placement de capteur. Pour ce faire, à chaque fois que nous plaçons un capteur, nous supprimons le sommet associé dans le graphe routier, ainsi que tous les sommets situés dans un rayon de 25 mètres, puis nous recalculons les centralités. Cette méthode n'a pas permis d'améliorer l'efficacité.

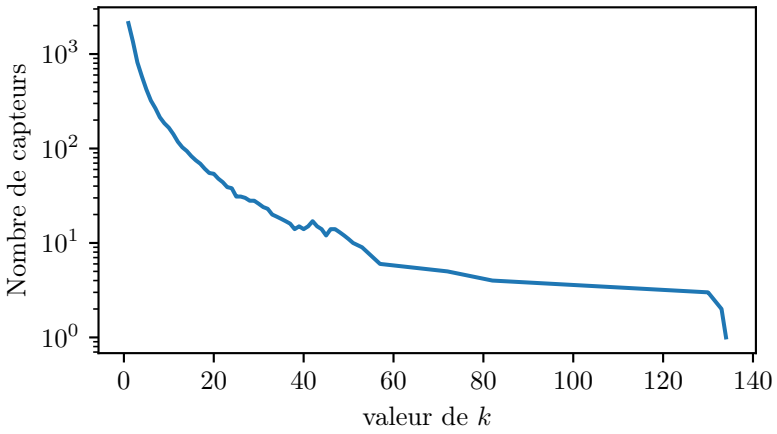


FIGURE 3.16 – Taille de la couverture, en fonction de k, sur la plus grande composante connexe de La Rochelle (3 671 sommets)

3.5.5 Impact d'un rayon de suppression variant en fonction de la valeur de centralité

Le tableau 3.5 montre la différence entre la suppression de sommets à une distance k fixe lorsqu'un capteur est déployé, et la suppression à une distance variable, comme expliqué dans l'algorithme 5. Pour chaque centralité, nous avons choisi la valeur de k qui maximise l'efficacité. Nous observons que pour chaque centralité, l'efficacité est plus élevée lorsque le rayon de suppression est variable.

Les images 3.17a et 3.17b montrent respectivement l'emplacement des capteurs lorsque le rayon de suppression varie et lorsque le rayon de suppression est fixe. Ces images donnent l'intuition que les capteurs sont uniformément répartis lorsque le rayon de suppression est fixe, alors qu'ils semblent plus denses dans les zones centrales lorsque le rayon de suppression est variable, ce qui permet d'observer plus de trajectoires, comme le montre le tableau 3.5.

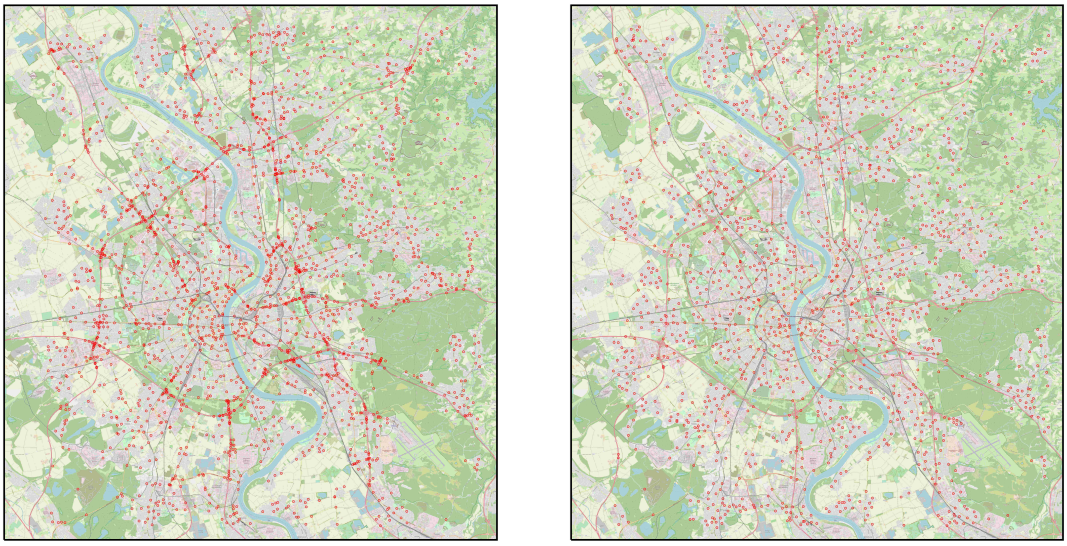
3.5.6 Impact de la couverture sur l'efficacité

Notre heuristique de placement de capteurs calcule une couverture, puis place des capteurs sur les sommets les plus centraux de la couverture. Nous étudions ici l'impact de la couverture sur la qualité du résultat. Nous souhaitons savoir si placer des capteurs parmi une couverture restreinte, ou au contraire contenant plus de sommets, permet d'obtenir une meilleur efficacité.

La première colonne du tableau 3.6 ($|VC|$) montre la taille de la couverture par rapport au nombre de sommets du graphe. La deuxième colonne représente le pourcentage de sommets qui recevront un capteur, tandis que les colonnes suivantes (sauf la dernière) représentent respectivement la proportion de trajectoires vues moins d'une fois, moins

	Variable		Fixé	
	k	Efficacité	k	Efficacité
Force	9	0,804	2	0,756
Degré	5	0,797	1	0,712
Katz	8	0,794	2	0,707
Expected Force	9	0,794	1	0,711
Accessibilité	10	0,791	2	0,707
Intermédiaire	7	0,764	1	0,696
Information	6	0,763	2	0,715
Closeness	4	0,712	1	0,684
Vecteur propre	3	0,701	2	0,721

TABLE 3.5 – Comparaison de l'efficacité pour un rayon de suppression fixe, et un rayon de suppression variable



(a) Rayon de suppression variable (b) Rayon de suppression fixe

FIGURE 3.17 – Comparaison du déploiement de capteurs en fonction du rayon de suppression, fixe ou variable, sur le jeu de données du centre-ville de Cologne

$\|VC\|$	$\|kVC\|$	< 1	< 2	< 4	Note
100%	7,17%	0%	1,40%	11,27%	Graphe complet
81,22%	6,27%	0,21%	2,41%	15,00%	Couverture Gavril et Yanna.
56,48%	5,13%	0,76%	4,33%	20,65%	Couverture Bar-Yehuda
50%	5,17%	1,37%	6,20%	26,00%	$\frac{n}{2}$ sommets aléatoires

TABLE 3.6 – Comparaison de la proportion de trajectoires observées en fonction de la couverture utilisée dans l'heuristique 5, essais réalisés avec la centralité accessibilité ($h = 4$), sur Cologne, et $k = 4$

de deux fois et moins de quatre fois.

La première ligne du tableau 3.6 correspond à l'exécution de notre heuristique, sans calculer de couverture, c'est-à-dire que la ligne 1 de notre heuristique 5 serait $VC \leftarrow V$, la variable VC recevrait tous les sommets du graphe routier.

La deuxième ligne du tableau correspond à une exécution de notre algorithme dans laquelle la couverture est calculée par l'algorithme 1 plutôt que par l'algorithme de Bar-Yehuda (algorithme 2).

La ligne 3 correspond à notre heuristique, sans modification, puisque par défaut, la couverture est calculée par l'algorithme de Bar-Yehuda.

La dernière ligne quant à elle revient à remplir la variable VC avec $\frac{n}{2}$ sommets aléatoires.

Nous observons que choisir $\frac{n}{2}$ sommets aléatoires consomme plus de capteurs, et a plus de trajectoires perdues, choisir le graphe complet (ligne 1) consomme plus de capteurs, mais permet de voir plus de trajectoires. Pour finir, utiliser l'algorithme de Gavril et Yannakakis consomme aussi plus de capteurs, mais permet d'observer plus de trajectoires.

Il s'agit de trouver un compromis entre le nombre de capteurs et la quantité de trajectoires observées. Dans notre heuristique, nous avons choisi d'utiliser l'algorithme de Bar-Yehuda, mais notre heuristique peut être modifiée facilement pour utiliser une autre approche.

3.5.7 Combinaison linéaire de centralités

Jusqu'ici, notre heuristique de placement de capteurs utilise une certaine centralité, passée en paramètre, pour décider quels sommets de la couverture auront un capteur. Choisir la bonne centralité n'est pas une tâche facile. Sur nos exemples, nous pouvions le faire car nous connaissions les trajectoires que nous cherchions, ce qui nous a permis d'étudier la pertinence de chaque centralité. Dans le cadre d'un déploiement réel, les trajectoires ne sont pas connues, il est donc difficile de décider *a priori* quelle centralité utiliser.

De plus, peut-être qu'une seule centralité n'est pas suffisante pour capturer toute

la complexité des déplacements humains : peut-être que les individus ont tendance à emprunter le plus court chemin pour rejoindre leur destination, mais peut-être qu'ils ont tendance à en dévier un peu pour emprunter de grands carrefours, par lesquels beaucoup de rues passent, ou au contraire à les éviter.

Pour ces raisons, nous étudions dans cette dernière section, l'utilisation d'une combinaison linéaire des centralités susmentionnées. Nous montrons que cette approche donne des résultats comparable à nos meilleurs centralités, sans avoir à choisir *a priori* une centralité.

Pour construire notre combinaison linéaire, nous utilisons des coefficients basés sur la corrélation de chaque centralité avec les trajectoires.

Soit le τ_b *de Kendall* [76], une mesure de corrélation de rang. C'est-à-dire, à quelle point les ordres de deux listes sont corrélés. Cette mesure va de -1 pour les corrélations négatives à 1 pour les corrélations positives. Elle est égale à 0 lorsqu'il n'y a pas de corrélation.

Nous calculons la valeur du $\tau_b^{c,d}$ de Kendall entre la liste de la quantité de trajectoires à chaque sommet et les valeurs d'une centralité c à chaque sommet, sur un jeu de données d. Nous calculons $\tau_b^{c,d}$ pour chaque centralité.

Nous obtenons le coefficient α_c^d de la centralité c, sur le jeu de données d, en normalisant $\tau_b^{c,d}$ par la somme de $\tau_b^{c,d}$ pour toutes nos centralités. Ceci nous permet de passer de valeurs $\tau_b^{c,d}$ comprises dans l'intervalle $[-1, 1]$ à des valeurs α_c^d dont la somme $\sum_c \alpha_c = 1$. α_c représente donc la corrélation entre l'ordre du nombre de trajectoire par sommet, et l'ordre de la valeur de la centralité c à chaque sommet, pour un jeu de données d.

Ensuite, nous calculons la moyenne des α_c^d sur plusieurs jeux de données [121]. Ceci nous permet d'obtenir des coefficients qui marchent en moyenne sur plusieurs jeux de données.

Plus formellement, nous obtenons le coefficient α_c^d, pour une centralité c, et un jeu de données d :

$$\alpha_c^d = \frac{\tau_b^c}{\sum_i \tau_b^i}$$

Nous faisons ensuite la moyenne α_c des α_c^d sur plusieurs jeux de données d. Nous considérons que tous les jeux de données ne sont pas égaux : un jeu de données de grande taille doit influer sur α_c de manière plus importante qu'un jeu de données de petite taille. Pour ce faire, nous faisons la moyenne pondérée sur nos n jeux de données, où d_t est un le nombre de trajectoires dans d divisé par le nombre total de trajectoires sur l'ensemble de nos jeux de données.

$$\alpha_c = \sum_{d=1}^{n} \frac{d_t \times \alpha_c^d}{n}$$

Nous pouvons enfin calculer la combinaison linéaire de centralités pour le sommet v :

$$\text{combinaison}(v) = \sum_c \alpha_c \times \text{cent}_c(v)$$

Centralité	Centre-ville Cologne	Bologne	Moyenne pondérée
Katz	0,1154	0,1651	0,1258
Degré	0,1196	0,1863	0,1196
Information	0,1691	0,1097	0,1691
Intermédiaire	0,1188	0,1603	0,1188
Closeness	0,1461	0,0534	0,1461
Accessibilité	0,1413	0,1022	0,1413
Vecteur propre	0,0750	0,0647	0,0750
Expected Force	0,1144	0,1584	0,1144

TABLE 3.7 – Coefficients $\tau_b^{c,d}$ de Kendall sur différents jeux de données d, pour différentes centralités c

Centre-ville Cologne	Bologne
Information	Degré
Closeness	Katz
Accessibilité	Intermédiaire
Degré	Expected Force
Intermédiaire	Information
Katz	Accessibilité
Expected Force	Vecteur prorpre
Vecteur propre	Closeness

TABLE 3.8 – Centralités ordonnées selon leur valeur α_c^d sur différents jeux de données d

Où α_c est le coefficient de la centralité c et $\text{cent}_c(v)$ est la valeur de la centralité c sur le sommet v.

Le tableau 3.7 montre les valeurs de α_c^d que nous obtenons pour chaque centralité sur les jeux de données de Bologne, et du centre-ville de Cologne, ainsi que la moyenne pondérée α_c sur nos deux jeux de données.

Pour cette expérience, nous avons choisi de diviser notre jeu de données concernant le centre-ville de Cologne en deux parties. Nous retirons d'abord 10 000 trajectoires du jeu de données, nous calculons les coefficients, puis nous utilisons les 10 000 trajectoires restantes pour valider nos coefficients.

Le tableau 3.8 montre nos centralités triées selon leur valeur α_c^d.

Pour finir, le tableau 3.9 résume nos résultats sur le centre-ville de Cologne, à la manière de nos expériences précédentes. Notons que ces résultats portent sur 10 000 trajectoires seulement, et que ces trajectoires n'ont pas été utilisées pour calculer les coefficients. Nous observons que la combinaison linéaire des centralités donne la meilleure efficacité en moyenne, sur l'ensemble de nos définitions des trajectoires perdues.

Malheureusement, la précision des coefficients de notre combinaison linéaire est limi-

tée par le faible nombre de jeux de données disponibles. Grâce à la méthodologie que nous développons, d'autres jeux de donnés pourrons voir le jour et ainsi enrichir nos coefficients.

Nous pensons que cette combinaison linéaire de centralité représente une manière efficace de choisir la bonne centralité pour notre heuristique. Notre heuristique ne dépend donc plus d'aucun paramètre.

3.6 Conclusion et perspectives

Nous avons développé une méthode de placement de capteurs dans un réseau urbain pour la reconstruction de trajectoires et l'inférence de contacts. Cette méthode tire parti du problème de la couverture par sommets et de la notion de centralité dans les graphes pour choisir les sommets les plus importants.

Nous avons montré que sur deux jeux de donnés, les centralités Degree, Katz et Expected Force fournissent les meilleurs résultats. Si l'on considère qu'une trajectoire doit être observée par un capteur dans au moins 20% des croisements de rues qu'elle traverse, sur notre jeu de données du centre-ville de Cologne, nous avons montré qu'en déployant des capteurs sur seulement 16% des intersections de rues, nous observons 95% des trajectoires. Pour mettre nos résultats en perspective, nous avons également comparé notre méthode à d'autres méthodes qui connaissent les trajectoires à l'avance. Celles-ci sont utilisées comme des bornes supérieures. Nous avons également développé une solution naïve pour servir de borne inférieure. Notre solution est en moyenne assez proche de nos bornes supérieures.

Nous pourrions affiner notre objectif : au lieu de maximiser le nombre de fois qu'une trajectoire est vue, nous pourrions la voir juste assez pour la reconstruire avec une précision suffisante. Ceci suppose de faire une étude plus approfondie de la définition de trajectoire *perdue* à retenir, ceci dépend de l'utilisation faite des trajectoires par la suite. Ainsi, nous pourrions éviter de déployer un trop grand nombre de capteurs sur des trajectoires déjà *observées*.

Nous pourrions aussi tenir compte de la topologie de chaque croisement de rue, car il se peut que nous devions placer plusieurs capteurs à certaines intersections afin d'obtenir une couverture complète de l'intersection, ce qui les rendraient plus coûteuses à surveiller. Par exemple, un grand carrefour, de par sa taille, nécessiterait de déployer plus de capteurs pour couvrir l'ensemble de sa surface. Un croisement de rue avec une statue, ou autre monument qui pourrait bloquer les ondes, peut aussi nécessiter plus de capteurs qu'un petit croisement de rues sans obstacle. Notre heuristique pourrait donc prendre en compte le coût du déploiement à chaque intersection.

Les trajectoires ont tendance à rester dans la même zone géographique, appelée *zone fonctionnelle* [60]. Il a été montré qu'il est possible d'identifier ces zones fonctionnelles à l'aide d'outils de théorie des graphes, comme la notion de *communautés*. Dans le cadre de notre étude de la mobilité, nous souhaitons donc identifier ces zones fonctionnelles, ce

qui nous permettrait de répartir nos capteurs uniformément (ou non) dans ces zones. Le chapitre suivant est consacré à la détection de communautés dans les graphes de terrain.

Perdues	20%				10%				4			
Centralité	k	Captr.	Perd.	Effi.	k	Captr.	Perd.	Effi.	k	Captr.	Perd.	Effi.
Force	9	16,14%	4,10%	0,804	17	10,07%	4,14%	0,862	9	16,14%	8,84%	0,764
Glout. traj.	185	17,13%	7,77%	0,764	150	13,89%	5,54%	0,814	185	17,13%	10,03%	0,746
Glout. auc. traj.	939	86,02%	0%	0,140	891	82,50%	0%	0,172	1059	85,93%	0,60%	0,141
Glout. perd.	181	16,76%	8,54%	0,761	117	10,83%	8,84%	0,813	188	17,41%	10,06%	0,743
Glout. auc. perd.	923	85,46%	0%	0,144	887	82,13%	0%	0,177	1058	28,85%	0%	0,731
Degré	**5**	**16,48%**	**4,54%**	**0,797**	7	13,80%	1,31%	0,851	4	20,56%	7,29%	0,736
Combinaison	**9**	**17,04%**	**4,23%**	**0,795**	**16**	**11,39%**	**2,12%**	**0,867**	**11**	**14,91%**	**11,22%**	**0,755**
Katz	8	18,15%	3,00%	0,794	**20**	**9,54%**	**4,39%**	**0,865**	8	18,15%	9,21%	0,743
Expected Force	9	17,13%	4,14%	0,794	**23**	**8,98%**	**5,00%**	**0,865**	9	17,13%	10,70%	0,740
Accessibilité	10	15,93%	5,88%	0,791	14	12,31%	2,50%	0,856	**9**	**17,31%**	**9,82%**	**0,756**
Intermédiaire	7	20,56%	3,88%	0,764	13	12,96%	3,26%	0,842	7	20,56%	9,02%	0,723
Information	6	20,37%	4,21%	0,763	14	11,02%	4,79%	0,847	10	14,63%	13,52%	0,738
Closeness	4	23,33%	7,10%	0,712	9	13,61%	8,35%	0,792	4	23,33%	10,96%	0,683
Vecteur propre	3	27,41%	3,39%	0,701	5	19,72%	3,19%	0,777	4	22,96%	11,32%	0,683
Aléatoire	286	26,48%	5,60%	0,696	204	18,89%	4,10%	0,780	258	23,89%	14,12%	0,654

TABLE 3.9 – Résultats pour chaque centralité, sur 10 000 trajectoires du centre-ville de Cologne, pour plusieurs définitions de trajectoires *perdues* (moins de 10% des sommets, moins de 20% des sommets, et moins de 4 sommets). Les résultats en gras correspondent aux cas où l'efficacité est maximale.

>

Chapitre 4

Communautés

Sommaire

Après avoir proposé une heuristique de placement de capteurs pour mesurer les trajectoires d'individus dans une zone urbaine, nous nous intéressons à l'organisation de ces zones urbaines, que nous étudions au travers de la notion de communautés.

Ce chapitre, qui constitue le cœur de cette thèse, étudie la notion de communautés dans les graphes de terrain. Pour ce faire, nous étudions les algorithmes de détection de communautés, et présentons une méthode permettant d'obtenir des communautés de meilleure qualité.

Ce chapitre commence par une introduction expliquant les enjeux de la détection de communautés, suivi d'une étude des méthodes de détection de communautés *consensuelles*. Nous présentons ensuite une amélioration qu'il est possible d'appliquer à ces méthodes. Enfin, nous présentons les résultats de notre méthode sur des graphes de terrain ainsi que des graphes synthétiques.

4.1 Introduction

Les trajectoires humaines ont tendance à rester dans les mêmes zones géographiques, appelées *zones fonctionnelles*. Dans un réseau routier, les communautés permettent de détecter ces *zones fonctionnelles* [60]. Ainsi, en identifiant ces zones, nous pourrions mieux comprendre l'organisation d'une aire urbaine, et potentiellement améliorer notre méthode de placement de capteurs développée au chapitre précédent.

Les graphes routiers, tout comme les graphes de terrain de manière générale, présentent des caractéristiques particulières qui les distinguent des graphes aléatoires. Un petit nombre de sommets a tendance à avoir beaucoup de voisins, tandis que la majorité des sommets a tendance à avoir peu de voisins. La répartition des arêtes n'est pas homogène : certaines parties du graphe sont densément connectées, tandis qu'entre ces parties denses, il n'y a généralement que quelques arêtes. Cette caractéristique des graphes de terrain est appelée *structure communautaire* [53]. La recherche de ces parties densément connectées d'un graphe est appelée *détection de communautés*.

La détection de communautés permet aussi d'étudier d'autres types de réseaux que les réseaux routiers. Dans un réseau d'achat reliant des clients et des produits, la détection de communautés peut permettre d'identifier des communautés de clients ayant des intérêts similaires et d'améliorer ainsi les systèmes de recommandation. Dans les réseaux sociaux, la détection de communautés peut aider à identifier des groupes de personnes : familles, amis ou collègues.

Il existe de nombreux algorithmes de détection de communautés, tels que Walktrap [109], Infomap [115] ou Louvain [15], que nous avons déjà présentés. Certains algorithmes sont non déterministes, comme Louvain, pour lequel les communautés produites sont déterminées par l'ordre dans lequel les sommets sont visités. Comme les sommets peuvent être visités dans n'importe quel ordre, cet algorithme peut produire différentes partitions. Nous avons donc besoin d'un moyen de choisir une « bonne » partition. Pour ce faire, nous avons besoin d'un critère pour trier les partitions et choisir celle qui est la plus représentative de la structure communautaire réelle du réseau.

En l'absence d'un tel critère, nous combinons les informations de différentes partitions en *communautés consensuelles* [86] (également connues dans la littérature sous le nom *cœurs de communautés* ou de *communautés constantes*). Ces communautés permettent de travailler avec des graphes dynamiques, car il devient plus facile de suivre les

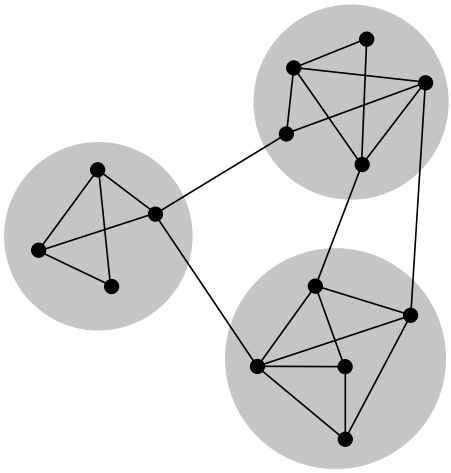


FIGURE 4.1 – Illustration de communautés, j_ham3, CC BY-SA 3.0 via Wikimedia Commons

communautés dans un réseau horodaté lorsque le calcul est déterministe [118].

Plus formellement, une *communauté* est un sous-ensemble de sommets d'un graphe. Dans notre cas, un sommet ne peut appartenir qu'à une seule communauté, et chaque sommet doit appartenir à une communauté. En d'autres termes, les communautés sont une *partition* des sommets d'un graphe.

Notons qu'il existe des *communautés recouvrantes*, dans lesquelles un sommet peut appartenir à plusieurs communautés, mais dans le cadre de ce travail, nous nous limitons aux communautés *non recouvrantes*. La figure 4.1 montre une illustration d'un graphe avec 3 communautés, correspondant aux disques gris.

4.2 Communautés consensuelles

La plupart des algorithmes de détection de communautés ne sont pas déterministes : ils peuvent produire des partitions différentes lorsqu'ils sont exécutés plusieurs fois. Nous avons donc besoin d'un moyen de décider quelle partition choisir. Même si nous pouvons les classer en fonction de leur modularité, nous ne devons pas éliminer les partitions de faible modularité, car elles peuvent encore apporter des informations utiles sur la structure communautaire sous-jacente. De plus, il a été démontré qu'il existe des partitions significativement différentes avec une modularité similaire [54].

Une meilleure solution consisterait donc à fusionner les partitions obtenues lors de plusieurs exécutions d'un algorithme de détection de communautés. Ces partitions fusionnées sont appelées communautés *consensuelles*. Il convient de noter qu'il est préférable d'utiliser des algorithmes non déterministes plutôt que des algorithmes qui n'utilisent que des règles arbitraires de départage d'égalité (*tie-breaking rules*), car ces derniers risquent de ne pas produire suffisamment de diversité pour que l'étape de fusion soit pertinente.

Dans Campigotto *et al.* [26], il a été montré que les graphes aléatoires n'ont pas de communautés consensuelles, même s'il est possible de trouver des partitions avec une modularité élevée. Les communautés consensuelles pourraient donc être plus significatives. De plus, Peixoto *et al.* ont montré que lorsqu'il y a trop de diversité dans les partitions calculées à travers différentes exécutions du même algorithme, il n'est en général pas possible d'obtenir une réponse cohérente [108]. Par conséquent, les communautés consensuelles ne devraient donner de bons résultats que lorsqu'il existe une structure communautaire dans le graphe.

4.2.1 Exemple introductif

Pour prendre en compte les similarités entre plusieurs partitions, la plupart des algorithmes utilisent, d'une manière ou d'une autre, une *matrice de consensus* P. Tout d'abord, un algorithme de détection de communautés (non consensuelles) $\mathscr{A}$ est exécuté n_p fois sur un graphe G. En raison du non-déterminisme de $\mathscr{A}$, les n_p partitions obtenues seront très probablement différentes, sauf dans les cas simples où les communautés sont très faciles à trouver.

La matrice de consensus P est alors construite comme suit : le *coefficient de consensus* P_{ij} est le nombre de fois où les sommets i et j ont été placés dans la même communauté par $\mathscr{A}$ au cours des n_p exécutions. P_{ij} varie donc entre 0 et n_p.

On construit alors le *graphe de consensus* pondéré G_P dont la matrice d'adjacence est P : les sommets i et j sont reliés dans G_P par une arête dont le poids est P_{ij}. Si $P_{ij} = 0$, alors les sommets i et j ne sont pas connectés [86, 118].

Un exemple de construction de matrice de consensus est donné en figure 4.2, où la colonne de gauche correspond à un graphe, dans lequel les sommets d'une même communauté sont entourés. La colonne du milieu représente toutes les partitions déjà détectées, et la colonne de droite montre la matrice de consensus.

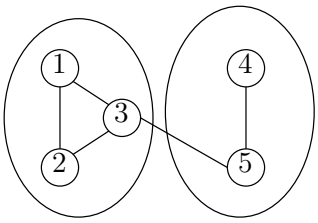
Dans la ligne 1, la matrice de consensus P ne comporte que des 0 ou des 1 car seule une détection de communautés a été faite pour le moment. Nous pouvons voir que les sommets 1 et 2 ont été mis une fois dans la même communauté, donc $P_{12} = 1$, alors que $P_{34} = 0$ car les sommets 3 et 4 n'ont pas été mis dans la même communauté.

Dans la deuxième ligne, correspondant à la deuxième détection de communautés, nous observons que les sommets 4 et 5 n'ont pas été mis dans la même communauté, mais ils ont été mis dans la même communauté une fois par le passé, donc $P_{45} = 1$. $P_{12} = 2$ car les sommets 1 et 2 ont été mis deux fois dans la même communauté jusqu'ici.

4.2.2 Méthode SGL

Le premier type d'approche, développé par Seifi *et al.* [118, 119] fonctionne sur le modèle de l'exemple introductif mentionné précédement.

Cette méthode considère que les faibles valeurs de P_{ij} ne sont pas significatives et ne conserve donc que les valeurs P_{ij} supérieures à un seuil donné τ et fixe toutes les autres à 0. On obtient ainsi une matrice seuillée P_τ et le graphe associé G_{P_τ}.



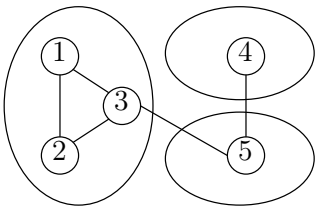
(a) Communautés

— $\{1,2,3\},\{4,5\}$

(b) Partitions

$$P = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

(c) Matrice de consensus



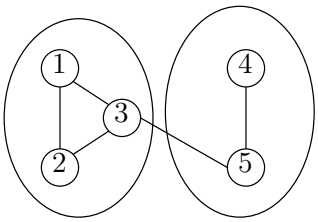
(d) Communautés

— $\{1,2,3\},\{4,5\}$
— $\{1,2,3\}\{4\}\{5\}$

(e) Partitions

$$P = \begin{bmatrix} 2 & 2 & 2 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}$$

(f) Matrice de consensus



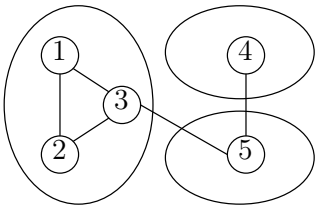
(g) Communautés

— $\{1,2,3\},\{4,5\}$
— $\{1,2,3\}\{4\}\{5\}$
— $\{1,2,3\},\{4,5\}$

(h) Partitions

$$P = \begin{bmatrix} 3 & 3 & 3 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 2 & 3 \end{bmatrix}$$

(i) Matrice de consensus



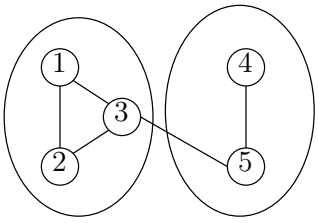
(j) Communautés

— $\{1,2,3\},\{4,5\}$
— $\{1,2,3\}\{4\}\{5\}$
— $\{1,2,3\},\{4,5\}$
— $\{1,2,3\}\{4\}\{5\}$

(k) Partitions

$$P = \begin{bmatrix} 4 & 4 & 4 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 2 & 4 \end{bmatrix}$$

(l) Matrice de consensus



(m) Communautés

— $\{1,2,3\},\{4,5\}$
— $\{1,2,3\}\{4\}\{5\}$
— $\{1,2,3\},\{4,5\}$
— $\{1,2,3\}\{4\}\{5\}$
— $\{1,2,3\},\{4,5\}$

(n) Partitions

$$P = \begin{bmatrix} 5 & 5 & 5 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 5 & 3 \\ 0 & 0 & 0 & 3 & 5 \end{bmatrix}$$

(o) Matrice de consensus

FIGURE 4.2 – Exemple de construction d'une matrice de consensus

Les composantes connexes de G_{P_τ} représentent les communautés consensuelles. Voir l'algorithme 6 pour plus de détails.

Des travaux ont été réalisés pour se débarrasser de τ et décider si l'entrée P_{ij} est statistiquement significative avant de la conserver ou non [94]. Dans le même esprit, Chakraborty *et al.* [28] considèrent que les communautés constantes sont des groupes de sommets qui sont toujours dans la même communauté dans plusieurs exécutions (c'est-à-dire pour lesquels $P_{ij} = 1$).

Algorithme 6 MÉTHODE SGL (G, τ, n_p, $\mathscr{A}$)

1: Appliquer n_p fois l'algorithme $\mathscr{A}$ sur le graphe G
2: Calculer la matrice de consensus P
3: Créer un graphe pondéré complet G_P tel que le poids de l'arête (i, j) est P_{ij}
4: Supprimer de G_P toutes les arêtes (i, j) dont le poids est strictement inférieur à τ pour obtenir G_{P_τ}
5: **retourne** les composantes connexes de G_{P_τ}

4.2.3 Méthode LF

La procédure LF [86] est proche de la méthode SGL. Elle n'utilise pas de paramètre τ, mais construit une matrice de consensus P, puis construit le graphe G_P dont la matrice d'adjacence est P et exécute $\mathscr{A}$ sur G_P n_p fois à nouveau. Si les n_p partitions sont toutes égales, l'algorithme s'arrête, sinon il construit une nouvelle matrice de consensus et itère de cette manière jusqu'à ce que toutes les n_p partitions soient égales. Voir Algorithme 7 pour plus de détails.

Algorithme 7 MÉTHODE LF (G, τ, n_p, $\mathscr{A}$)

1: Appliquer n_p fois l'algorithme $\mathscr{A}$ sur le graphe G
2: Calculer la matrice de consensus P
3: Appliquer $\mathscr{A}$ n_p fois sur P
4: **si** les n_p partitions sont toutes égales **alors**
5: **retourne** la dernière partition obtenue par $\mathscr{A}$
6: **sinon**
7: Retourner à l'étape 2
8: **fin si**

4.2.4 Autres approches

Liang *et al.* [91] utilisent l'Algorithme Label Propagation (LPA) [112] comme algorithme $\mathscr{A}$. Ils l'exécutent n_p fois pour construire une matrice de consensus, puis effectuent une seule exécution d'une version de LPA pour graphes pondérés sur G_P.

Wang et Fleury ont développé un algorithme pour les communautés recouvrantes qui construit un graphe de consensus, puis fusionne les sommets lorsque leur coefficient de consensus est supérieur à un seuil α et répète le processus [136].

Yang et Leskovec ont développé un algorithme pour la détection des communautés consensuelles qui se chevauchent, avec une matrice de consensus [141].

Certains auteurs ont choisi d'utiliser différents algorithmes pour générer les premières n_p partitions (plutôt que d'utiliser le même algorithme $\mathscr{A}$), afin d'avoir plus de diversité dans le consensus. La méthode Azar [78] choisit n_p algorithmes de détection de communautés et chacun d'entre eux est exécuté une seule fois sur G. Ensuite, les informations provenant des n_p partitions sont regroupées dans une matrice de consensus à partir de laquelle ils construisent le graphe de consensus associé. Enfin, ils exécutent un autre algorithme de détection des communautés sur ce graphe de consensus.

Pour éviter de tomber dans un optimum local de modularité, Liu *et al.* [93] ont choisi d'appliquer des perturbations au graphe initial, puis d'utiliser une version modifiée de Louvain [15] qui peut explorer un éventail de solutions plus large. Enfin, ils regroupent les résultats dans une matrice de consensus.

Burgess *et al.* [24] se concentrent sur l'ajout d'arêtes intra-communautaires au graphe de départ G, dans le but d'augmenter l'efficacité de l'algorithme de détection des communautés. Pour ce faire, ils calculent la similarité entre les paires de sommets, en se basant sur des métriques telles que Jaccard [69] ou le nombre de voisins communs. Les arêtes sont ensuite ajoutées de manière non déterministe à G, puis ils exécutent $\mathscr{A}$ sur G. Le processus est répété n_p fois pour obtenir une matrice de consensus.

Rasero *et al.* [114] utilisent une autre version de la matrice de consensus qui n'est pas obtenue par détection de communautés mais à partir des données originales. Ils étudient les graphes de connectivité cérébrale de différentes personnes et agrègent les différents graphes dans une matrice de consensus.

La matrice de consensus P est une matrice $n \times n$. Le calcul et le stockage d'une telle matrice deviennent rapidement infaisables sur de grands graphes. C'est malheureusement régulièrement le cas puisque les réseaux complexes peuvent avoir jusqu'à plusieurs milliards de sommets [46]. Plusieurs études ont donc travaillé à l'amélioration du calcul de la matrice de consensus, comme dans les algorithmes de Tandon [126] et ECG [110] où seules les entrées P_{ij} correspondant à des arêtes (i, j) de G sont considérées. Ainsi, au lieu de calculer n^2 entrées dans P, nous en calculons seulement m (le nombre d'arêtes dans le graphe), toutes les autres entrées étant 0.

4.2.5 Algorithme de Tandon *et al.*

Proposé en 2019, l'algorithme de Tandon [126] se concentre sur l'optimisation du calcul de la matrice de consensus.

Après avoir exécuté n_p fois un algorithme de détection de communauté $\mathscr{A}$ sur un graphe G, les auteurs construisent une matrice de consensus P, où $P_{ij} = 0$ pour les paires de sommets (i, j) qui ne sont pas reliées par une arête dans G. Sinon, le coefficient de consensus P_{ij} est calculé normalement.

Ensuite, le graphe pondéré G_P, dont la matrice d'adjacence est P, est généré. Puis, m sommets sont choisis au hasard. Pour chacun d'entre eux, deux sommets voisins, j et k, sont choisis au hasard, et le coefficient de consensus P_{jk} est calculé, et l'arête correspondante ajoutée à G_P.

L'étape finale consiste à appliquer $\mathscr{A}$ n_p fois de nouveau sur G_P et à mettre à jour la matrice de consensus P. Si moins de 2% des entrées non nulles de P sont inférieures à n_p, $\mathscr{A}(G_P)$ est calculé une dernière fois et les communautés sont retournées. Sinon, le processus est itéré en construisant une nouvelle matrice de consensus P en prenant en compte ces n_p nouvelles partitions. Voir l'algorithme 8 pour plus de détails.

Notons que toutes les arêtes de G_P dont le poids est strictement inférieur à un certain paramètre τ sont alors mises à zéro. Au cas où cela déconnecte le graphe, le lien de poids le plus élevé est conservé.

Algorithme 8 Algorithme de Tandon *et al.* (G, τ, n_p, $\mathscr{A}$, m)

1: Appliquer n_p fois un algorithme de détection de communautés $\mathscr{A}$ sur G
2: Calculer les valeurs P_{ij} de la matrice de consensus P si i et j sont voisins dans G
3: Supprimer les arêtes de G_P dont le poids est $< \tau$. Si un sommet est déconnecté, garder son arête de poids maximal.
4: Sélectionner m sommets au hasard et, pour chacun, sélectionner deux voisins j et k au hasard. Calculer le coefficient de consensus P_{jk} et ajouter l'arête correspondante à G_P
5: Appliquer $\mathscr{A}$ n_p fois sur G_P
6: **si** Moins de 2% des entrées non nulles sont strictement inférieures à n_p **alors**
7: Appliquer une dernière fois $\mathscr{A}$ sur le graphe correspondant à P
8: **retourne** La partition obtenue
9: **sinon**Retourner à l'étape 2
10: **fin si**

Comme l'ont montré Tandon *et al.*, le temps d'exécution de la procédure LF est dominé par le calcul de la matrice de consensus. Ils ont mesuré le temps d'exécution de la procédure LF et de l'algorithme de Tandon et ont observé que le temps d'exécution de la procédure LF semble avoir une complexité de $n^{1,6}$ et celui de l'algorithme de Tandon $n^{1,2}$. Ils ont constaté que, dans leur méthode, le temps de calcul de la matrice de consensus évolue linéairement en fonction du nombre d'arêtes m.

4.2.6 Ensemble Clustering for Graphs (ECG)

Introduite en 2018, la méthode ECG [110] est une autre méthode qui ne calcule pas tous les coefficients de consensus. Elle s'appuie sur la notion de *k-core*, qui est un sous-graphe maximal qui ne contient que des sommets de degré k ou plus. Trouver le *k-core* d'un graphe peut être fait en retirant itérativement chaque sommet de degré inférieur à k. Notons que si la suppression d'un sommet entraîne la création d'un sommet de degré inférieur à k, alors ce dernier devra être retiré aussi.

ECG choisit alors de ne calculer que les coefficients de consensus qui correspondent aux arêtes du 2-core de G. Il prend un paramètre supplémentaire, w_*, qui est le plus petit poids autorisé dans la matrice de consensus. Tout poids inférieur à w_* est remplacé par w_*, comme indiqué par l'équation 4.1. Ils ont choisi d'utiliser le niveau 1 de la méthode de Louvain comme algorithme de détection de communautés $\mathscr{A}$, car il a été démontré que l'utilisation d'un algorithme de clustering faible peut produire des résultats de haute qualité [127], ce qui est le cas du niveau 1 de Louvain [110]. Pour plus de détails, voir Algorithme 9.

Algorithme 9 MÉTHODE_ECG (G, n_p, w_*)

1: Exécuter Louvain n_p fois sur G. Ne garder que les partitions du premier niveau
2: Construire la matrice de consensus P en ne calculant les entrées P_{ij} seulement si i et j sont connectés dans G. Ajouter un poids selon l'équation 4.1
3: Exécuter Louvain sur G_P
4: **retourne** Les communautés calculées à l'étape précédente

$$P_{ij} = \begin{cases} w_* + (1 - w_*) \times \left(\frac{P_{ij}}{n_p}\right) & \text{si } (i,j) \text{ est dans un 2-core de } G \\ w_* & \text{sinon} \end{cases} \tag{4.1}$$

Où P_{ij} est le coefficient de consensus des sommets i et j, c'est-à-dire le nombre de fois ils ont été placés dans la même communauté dans les n_p exécutions de Louvain.

Notons que gâce à leur déterminisme, les communautés consensuelles permettent aussi de suivre l'évolution de la structure communautaire dans les graphes dynamiques, qui sont des graphes dans lesquels les sommets et les arêtes peuvent changer avec le temps [86].

Le critère permettant de décider quels coefficients de consensus calculer peut être amélioré pour réduire encore la complexité en temps et en espace du calcul de la matrice de consensus. La sous-section suivante propose un autre critère.

4.3 Amélioration du filtrage du bruit

La présence d'arêtes entre communautés limite l'efficacité des algorithmes de détection des communautés. S'il n'y avait que des arêtes intra-communautaires, les communautés seraient les composantes connexes du graphe et leur identification serait triviale.

Par conséquent, l'ajout dans la matrice de consensus de toutes les entrées correspondant à des paires de sommets se trouvant dans des communautés différentes ajoute du bruit.

Les deux principales contributions de ce chapitre sont de mettre en évidence la présence de ce bruit et de trouver un critère qui permet de diminuer le nombre d'entrées, et donc de limiter ce bruit, tout en conservant les informations pertinentes. La diminution du nombre d'entrées dans P pourrait également réduire la complexité temporelle et spatiale du calcul de la matrice de consensus.

4.3.1 L'information est bruitée

Cette sous-section montre que la matrice de consensus P est bruitée. Pour ce faire, nous montrons que la qualité du partitionnement est maximale lorsque la matrice de consensus n'est que partiellement remplie.

Pour ce faire, nous générons un graphe LFR G (ainsi que ses communautés, servant de vérité terrain). Nous utilisons également une distance $dist$ entre les sommets qui sera décrite plus en détail dans les sous-sections suivantes.

Ensuite, nous exécutons n_p fois un algorithme de détection de communauté $\mathscr{A}$ sur G. Enfin, nous construisons une matrice de consensus P, mais nous la remplissons en ajoutant les entrées P_{ij} une par une dans l'ordre croissant de $dist(i, j)$. En cas d'égalité, une paire de sommet aléatoire est choisie (parmi les paires à égalité).

Après chaque ajout dans la matrice de consensus partielle P, nous construisons le graphe de consensus partiel associé G_P et exécutons $\mathscr{A}$ sur G_P. Nous calculons ensuite la NMI entre les communautés ainsi calculées et celle de la vérité terrain. Nous itérons de cette manière jusqu'à ce que la matrice P soit complètement remplie. Cette méthode nous permet d'étudier comment la NMI varie en fonction du nombre d'entrées dans P.

Notre méthode repose sur plusieurs paramètres : le graphe généré à l'aide de LFR, en particulier son paramètre de mélange μ qui mesure la proportion de liens entre les communautés, l'algorithme $\mathscr{A}$ utilisé pour calculer les communautés, le nombre n_p d'exécutions de $\mathscr{A}$ et la distance $dist(i, j)$. La méthodologie détaillée est présentée dans l'algorithme 10. Dans notre cas, nous avons choisi $\mathscr{A}$ = Louvain, et $n_p = 12$ car il a été démontré que la matrice de consensus converge rapidement lorsque n_p augmente [110]. Ainsi, fixer n_p à une valeur plus élevée ne devrait pas apporter plus d'information. Nous ferons varier μ et utiliserons plusieurs distances $dist$ dans les sous-sections suivantes.

4.3.2 Distance dans les graphes

Tout d'abord, nous considérons comme $dist$, la distance dans le graphe entre deux sommets (le plus petit nombre d'arêtes consécutives) et nous appliquons l'algorithme 10 pour différentes valeurs de μ. Nous utilisons la distance car dans l'algorithme de Tandon [126] et ECG [110] les auteurs se limitent à des paires de sommets connectés, c'est-à-dire des sommets à distance 1. Une fois toutes les paires de sommets à distance 1 ajoutées nous aurons donc exactement le comportement de ces algorithmes. Nous pourrons ensuite voir ce qui se produit en ajoutant des sommets à distance 2.

Algorithme 10 ORDRE (un graphe LFR G, $\mathscr{A}$, n_p, $dist$)

1: Construire une liste ordonnée L de paires de sommets (i, j) respectant l'ordre donné par $dist(i, j)$
2: Exécuter n_p fois l'algorithme $\mathscr{A}$ sur G
3: **tant que** L n'est pas vide **faire**
4: Choisir la paire (i, j) pour laquelle $dist(i, j)$ est minimale dans L et la supprimer de L
5: Calculer le coefficient de consensus P_{ij} et l'ajouter à P
6: Exécuter $\mathscr{A}(G_P)$
7: Calculer la NMI entre la vérité terrain et la partition trouvée à l'étape précédente
8: **fin tant que**
9: **retourne** Les valeurs de NMI en fonction du nombre d'entrées dans P

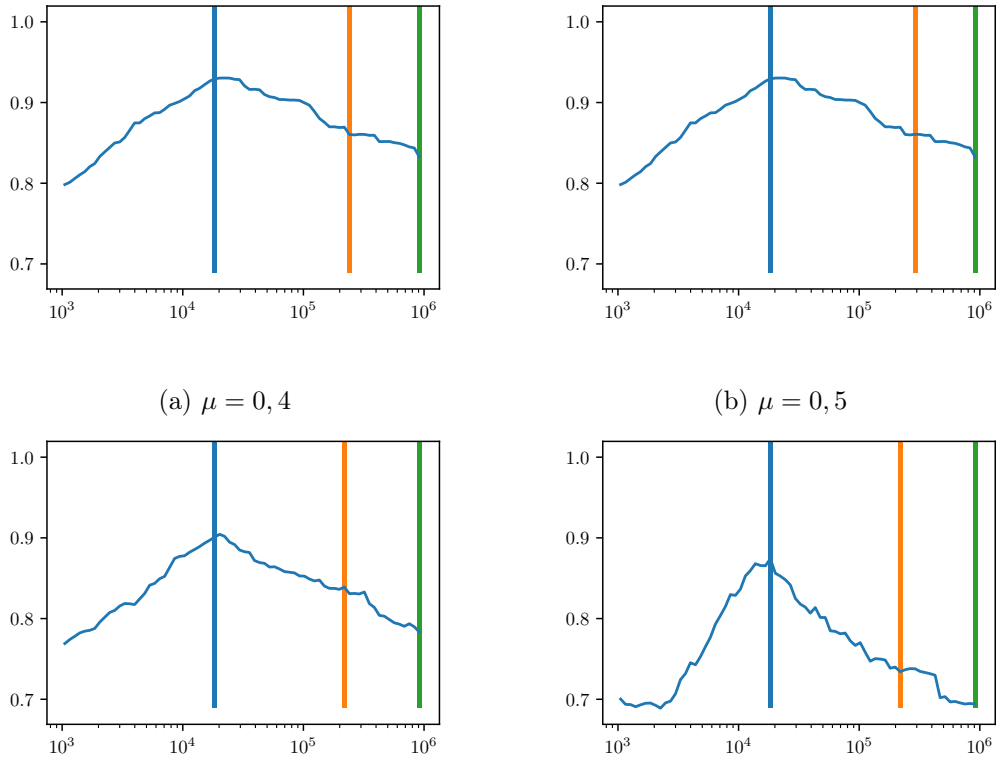
Les figures 4.3a à 4.3d montrent la NMI en fonction du nombre d'entrées dans P, pour des graphes LFR avec 10 000 sommets et 4 valeurs différentes du paramètre de mélange μ. Les barres verticales représentent la variation de la distance entre les sommets. En d'autres termes, à gauche de la barre bleue, seules les paires de sommets (i, j) où la distance$(i, j) \leq 1$ sont présentes, tandis qu'à gauche de la barre orange, les paires (i, j) où la distance$(i, j) \leq 2$ sont présentes et ainsi de suite.

Les figures ont été générées avec une implémentation disponible en ligne [84]. Les paramètres utilisés pour LFR sont les suivants : nombre de sommets $n = 10000$, degré moyen $k = 20$, degré maximal $\max_k = 50$, exposant de la distribution de degrés $t_1 = 2$, exposant de la distribution des tailles des communautés $t_2 = 3$, taille minimale des communautés $\min_c = 10$, taille maximale des communautés $\max_c = 50$, paramètre de mélange des poids $\mu_w = 0,6$. L'algorithme $\mathscr{A}$ est Louvain.

Les figures 4.3 nous fournissent trois informations principales. Tout d'abord, et c'est assez naturel, des valeurs plus élevées du paramètre de mélange μ correspondent à des valeurs plus faibles de NMI. En effet, des valeurs plus élevées de μ créent des communautés moins marquées et donc plus difficiles à identifier. Les valeurs maximales obtenues sont respectivement de $0,93$, $0,93$, $0,904$ et $0,874$ pour μ allant de $0,4$ à $0,7$.

Ensuite, la forme des courbes indique qu'il faut ajouter suffisamment de valeurs dans la matrice de consensus pour pouvoir calculer les communautés mais qu'il ne faut pas la remplir entièrement. En effet, la NMI diminue après un certain nombre d'entrées ajoutées. Intuitivement, les algorithmes de détection de communautés fonctionneraient mieux s'il n'y avait pas d'arêtes inter-communautaires et si toutes les paires de sommets d'une même communauté étaient reliées par une arête.

Le non-déterminisme de $\mathscr{A}$ rend les n_p partitions potentiellement différentes et, par conséquent, certains coefficients de consensus non nuls correspondent à des paires de sommets qui sont dans des communautés différentes dans la vérité terrain. Plus nous ajoutons de telles valeurs, plus il est difficile pour l'algorithme de trouver les communautés réelles, même si ces valeurs sont très proches de 0. En d'autres termes, avoir une matrice de consensus complètement remplie ajoute du bruit et les paires de sommets à grande



(a) $\mu = 0,4$ (b) $\mu = 0,5$

(c) $\mu = 0,6$ (d) $\mu = 0,7$

FIGURE 4.3 – NMI en fonction du nombre d'entrées de P remplie en utilisant l'ordre basé sur la distance dans les graphes. Graphes LFR avec 10 000 sommets et différentes valeurs du paramètre de mélange μ. Les barres verticales représentent le changement de distance : les entrées ajoutées avant la barre bleue correspondent à des paires de sommets à distance 1 (en utilisant un ordre aléatoire pour départager les égalités). Entre les barres bleues et oranges se trouvent les paires à distance 2, et ainsi de suite.

distance semblent plus bruitées que les paires de sommets à distance un ou deux.

Enfin, nous observons qu'en utilisant la distance, la NMI est maximale lorsque toutes les entrées correspondant aux paires de sommets à la distance 1 sont présentes dans la matrice de consensus, ainsi qu'une partie des entrées à distance 2 (le pic est situé un peu après la barre bleue qui correspond à toutes les paires à la distance 1). Même si Tandon [126] et EGC [110] n'utilisent pas exactement la même méthode que nous, il serait possible de faire un peu mieux que ces algorithmes en ajoutant quelques entrées supplémentaires dans la matrice de consensus.

Pour améliorer ces algorithmes, il faudrait donc sélectionner quelques paires de sommets à distance 2, mais pas toutes car la qualité est fortement dégradées si toutes ces paires sont ajoutées à la matrice. Cependant, comme la distance ne peut prendre que des valeurs entières, nous sommes limités par sa précision et nous avons donc besoin d'un autre critère qui nous aiderait à discriminer ces entrées.

4.3.3 Coefficient de clustering d'arête

Intuitivement, nous voulons utiliser une distance qui nous permettra d'ajouter d'abord les entrées correspondant aux paires de sommets intra-communautaires. Plusieurs mesures permettent d'identifier de telles paires de sommets comme le coefficient de Jaccard [69], la similarité Cosinus ou la distance de Hamming [58]. Nous avons testé toutes ces mesures, ainsi que le coefficient de clustering d'arête (ou Edge Clustering Coefficient, ECC) [111] qui mesure la similarité entre deux sommets i et j.

La figure 4.4 montre la distance euclidienne entre une matrice de consensus complètement remplie P_f et une matrice de consensus partiellement remplie P_p. La matrice P_p est remplie en suivant l'ordre donné par l'un des mesures mentionnées ci-dessus. À chaque fois qu'une entrée est ajoutée dans P_p, la distance entre P_p et P_f est recalculée. Nous pouvons ainsi voir la vitesse de convergence entre la matrice P_p et P_f, pour différences mesures. Ceci nous permet de comparer les mesures entre elles.

La mesure « ECCMax » correspond à l'ordre décroissant donné par l'ECC (la paire de sommet dont l'ECC est maximale parmi celles qui ne sont pas encore dans P_p). Les autres mesures dans la légende de la figure 4.4 sont construites sur le même modèle.

La plupart de ces mesures donnent de bons résultats. La similarité Cosinus a été écartée en raison de son temps de calcul plus élevé. La distance de Hamming n'offrait pas de bons résultats. Le coefficient de Jaccard et l'ECC nous permettent d'obtenir de bons résultats. Nous présentons ici les résultats pour l'ECC qui sont légèrement meilleurs.

Nous procédons de la même manière que précédemment : nous générons d'abord un graphe LFR G, nous calculons l'ECC de toutes les paires de sommets qui sont à distance un ou deux. Les sommets situés à une distance de trois ou plus ne peuvent avoir aucun voisin en commun, et auront donc nécessairement une ECC nulle. De toute manière, les paires à distance > 3 doivent être évités pour maximiser la NMI, comme le montrent les figures 4.3. Enfin, nous procédons comme expliqué dans l'algorithme 10 mais en insérant les valeurs par ordre décroissant de leur ECC.

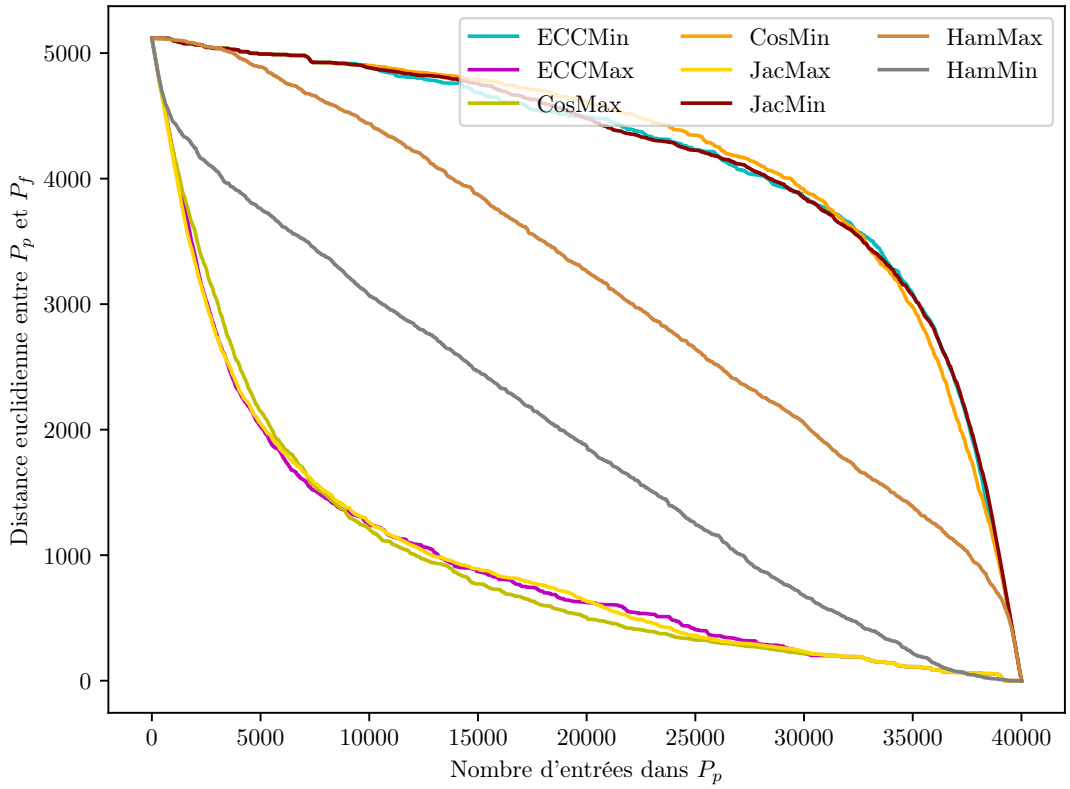
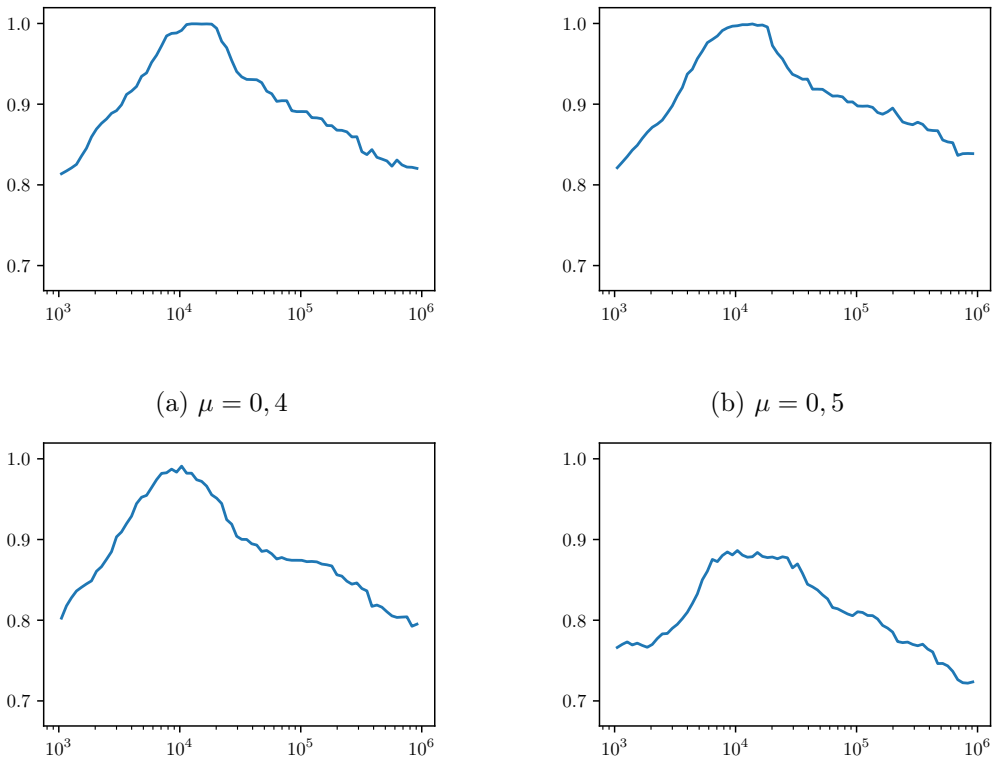


FIGURE 4.4 – Distance entre la matrice de consensus et la matrice partiellement remplie. Le remplissage est effectué dans l'ordre donné par les différentes heuristiques. Graphe LFR avec $n = 200$ et $\mu = 0,5$



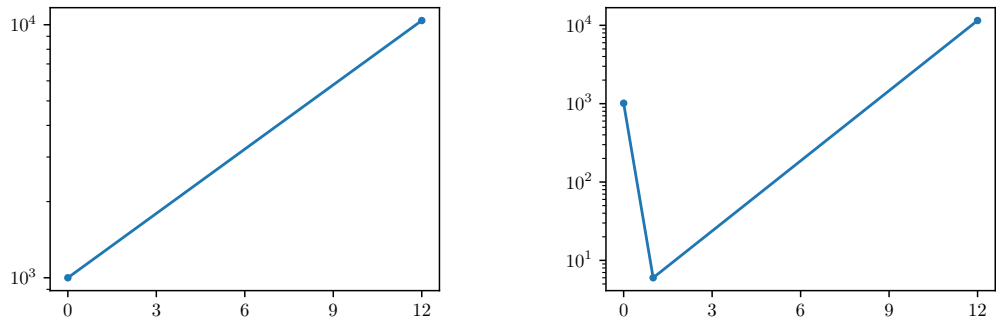
(a) $\mu = 0,4$ (b) $\mu = 0,5$

(c) $\mu = 0,6$ (d) $\mu = 0,7$

FIGURE 4.5 – NMI en fonction du nombre d'entrées dans P remplie par ECC décroissante. Graphes LFR avec 10 000 sommets et différentes valeurs du paramètre de mélange μ.

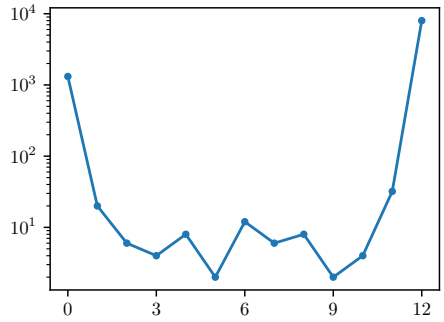
Les figures 4.5 montrent la NMI en fonction du nombre d'entrées dans P, pour 4 graphes LFR différents, avec 10 000 sommets, et différentes valeurs du paramètre de mélange μ. Les graphes ont été générés avec la même implémentation et les mêmes paramètres que les figures 4.3. La valeur de l'ECC étant un nombre réel, nous pouvons maintenant choisir précisément quelles entrées ajouter dans la matrice de consensus, en fonction de leur ECC. Nous remarquons également que la NMI maximale est plus grande pour l'ECC que dans les figures 4.3. Elles valent respectivement $0,9996$, $0,9994$, $0,9910$ et $0,8864$ pour des μ de $0,4$ à $0,7$ (quand les valeurs maximales étaient $0,93$, $0,93$, $0,904$ et $0,874$ avec le critère de distance, en figure 4.3).

Les figures 4.6 montrent la distribution des coefficients de consensus dans la matrice de consensus lorsque la NMI est maximale. En d'autres termes, les entrées de la matrice de consensus sont ajoutées par ordre décroissant de leur ECC jusqu'à ce que la NMI soit maximale. Nous traçons ensuite la distribution des coefficients de consensus dans matrice. Nous constatons qu'à mesure que le paramètre de mélange μ augmente, les coefficients

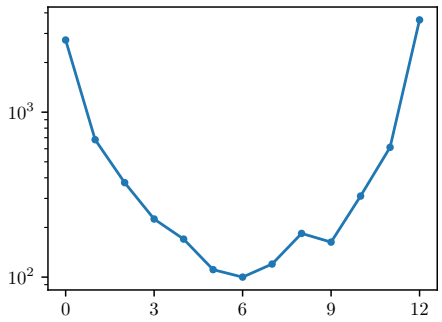


(a) $\mu = 0,4$

(b) $\mu = 0,5$



(c) $\mu = 0,6$



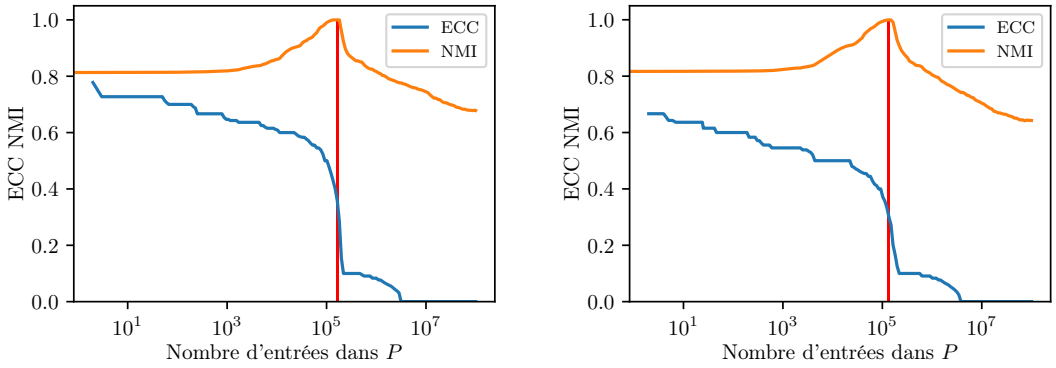
(d) $\mu = 0,7$

FIGURE 4.6 – Distribution des coefficients de consensus dans la matrice de consensus quand la NMI maximale, pour $\mu = 0,4$ à $\mu = 0,7$

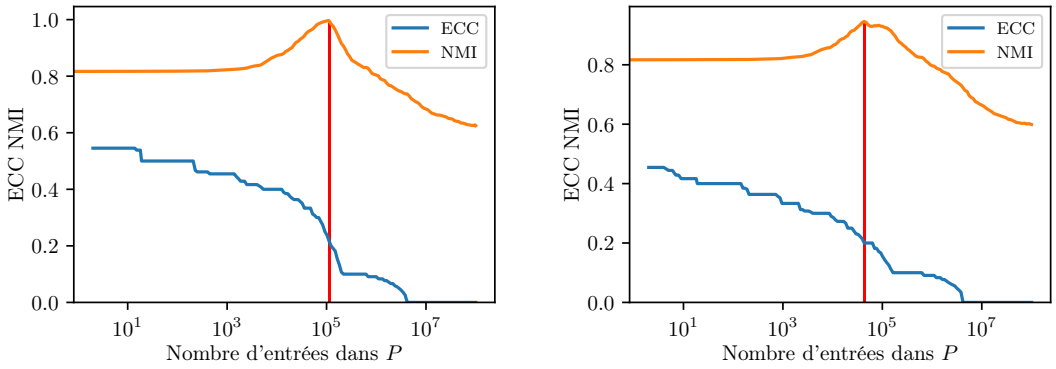
de consensus sont plus faibles lorsque la NMI est maximale. Cela signifie que lorsque μ augmente, la matrice de consensus devient plus bruitée.

Dans les scénarios réels, nous ne connaissons pas les communautés de la vérité terrain, et nous ne pouvons donc pas calculer la NMI. Nous devons donc décider à l'avance du nombre d'entrées de P devant être calculées pour se rapprocher le plus possible de la NMI maximale.

Les figures 4.7a à 4.7d montrent comment la NMI et l'ECC évoluent. Nous remarquons trois phases dans ces figures : une première phase où à mesure que nous remplissons la matrice de consensus P, la NMI augmente, suivi d'un court plateau, suivi d'une phase où plus nous ajoutons d'information dans P, plus la NMI décroit ! Côté ECC, dans la figure 4.7a, nous distinguons aussi trois phases, une première où l'ECC est élevée, ce qui intuitivement pourrait correspondre à des paires de sommets qui se trouvent dans une même communauté, suivi d'une phase de décroissance brutale, suivi d'une phase où l'ECC est faible, ce qui pourrait correspondre à des paires de sommets se trouvant dans des communautés différentes. Il semble que la décroissance brutale de l'ECC corresponde



(a) $\mu = 0,4$ (b) $\mu = 0,5$



(c) $\mu = 0,6$ (d) $\mu = 0,7$

FIGURE 4.7 – NMI et seuil sur l'ECC en fonction du nombre d'entrées dans la matrice de consensus, pour $\mu = 0,4$ à $\mu = 0,7$

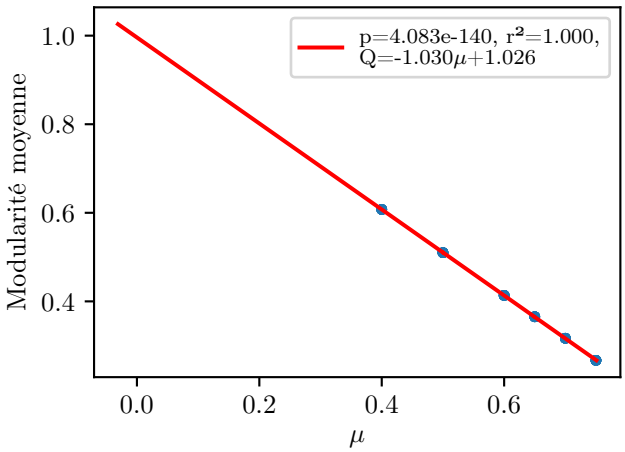


FIGURE 4.8 – Modularité moyenne en fonction du paramètre de mélange μ. Graphes LFR avec 10 000 sommets.

au maximum de la NMI, au moins pour certaines valeurs de μ. La prochaine section se sert de cette observation pour décider quand arrêter de remplir la matrice.

4.4 Trouver le nombre optimal d'entrées – Filtrage du bruit

Cette section a pour objectif de montrer comment, à partir de certaines corrélations, il est possible d'estimer le nombre d'entrées à insérer dans la matrice de consensus.

4.4.1 Vers un critère d'arrêt

La figure 4.8 montre que la modularité moyenne obtenue sur plusieurs exécutions de $\mathscr{A}$ est linéairement corrélée avec le paramètre de mélange μ sur les graphes LFR. Nous utilisons la corrélation :

$$Q = 1 - \mu$$

Kaminski *et al.* ont étudié ce phénomène sur le modèle de graphe ABCD, qui est très proche de LFR [72]. Ils ont montré que la modularité de la vérité terrain d'un graphe ABCD avec un paramètre de mélange μ atteint asymptotiquement $1-\mu$ lorsque le nombre de sommets n augmente. Ils observent également que la modularité est plus faible pour les petits graphes, avant de converger vers $1 - \mu$.

La figure 4.9 montre que le paramètre de mélange μ est corrélé à la valeur d'ECC τ au-delà de laquelle les paires de sommets doivent être ajoutées à la matrice de consensus afin de maximiser la NMI. C'est-à-dire que l'on doit ajouter à la matrice de consensus le

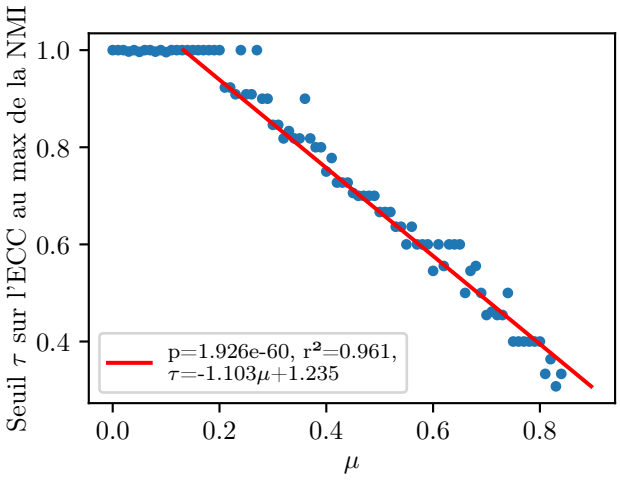


FIGURE 4.9 – Paramètre de mélange μ en fonction du seuil τ sur l'ECC, en dessous duquel les paires de sommets sont ajoutées à la matrice de consensus.

coefficient de consensus de toutes les paires de sommets (i, j) dont l'ECC est supérieur à τ.

$$\tau = -1,103\mu + 1,235 \tag{4.2}$$

La combinaison de ces deux corrélations permet de déduire le nombre optimal d'entrées dans la matrice de consensus à partir des premières n_p exécutions de $\mathscr{A}$. Nous calculons n_p partitions, ce qui n'entraîne pas de surcharge de calcul puisqu'il s'agit de la première étape du calcul de la matrice de consensus. Puis, nous calculons la modularité moyenne et en déduisons la valeur de μ, à partir de laquelle nous déduisons le seuil τ sur l'ECC. Ces corrélations sont effectuées sur des graphes synthétiques LFR et peuvent ne pas être valables pour des graphes réels. Cependant, nous verrons dans la Section 4.5 qu'elles donnent de bons résultats même dans ces situations.

Une limitation potentielle de ces corrélations est que pour un μ fixé, la modularité moyenne sur plusieurs exécutions augmente avec la taille du graphe et se stabilise lorsque le nombre de sommets d'un graphe atteint 20 000 (voir la figure 4.10). Une solution consisterait à augmenter artificiellement la modularité moyenne pour les graphes de moins de 20 000 sommets. La figure 4.10 montre la modularité Q en fonction de n, pour les graphes LFR avec $\mu = 0,4$. Nous répétons l'opération pour différentes valeurs de μ et obtenons un ajustement dans le cas général $\frac{Q_{\max} \times n}{(K+n)}$, où $Q_{\max}$ est la modularité pour un graphe suffisamment grand, et $K = 71,21$. Ainsi :

$$Q_{\max} = \frac{KQ}{n} + Q \tag{4.3}$$

Le seuil τ est ainsi déduit de $Q_{\max}$ si le graphe a moins de 20 000 sommets.

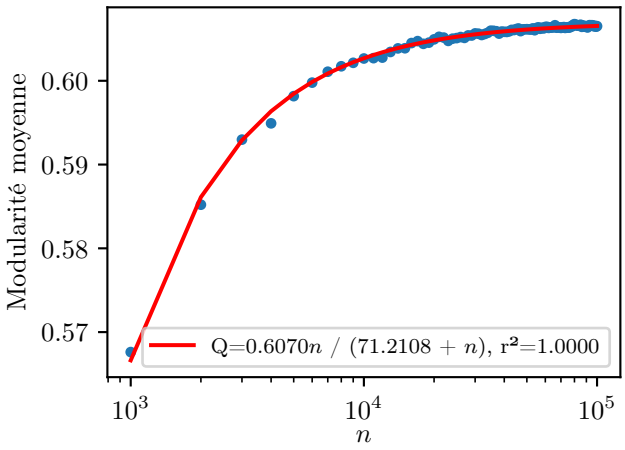


FIGURE 4.10 – Modularité moyenne en fonction du nombre de sommets, graphe LFR avec $\mu = 0,4$.

La figure 4.11 montre le seuil τ sur l'ECC calculé avec les corrélations que nous avons présentées, par rapport au seuil mesuré en remplissant la matrice de consensus pas à pas, en calculant la NMI, et en s'arrêtant lorsqu'elle est maximale. Nous observons une forte corrélation entre les deux courbes. Pour $\mu \leq 0{,}2$, nous mesurons un seuil de 1, ce qui signifie que tous les coefficients de consensus doivent être calculés. Le seuil τ que nous calculons étant supérieur à 1, les coefficients de consensus ajoutés dans P seront donc les mêmes.

4.4.2 Amélioration des algorithmes existants

Dans cette sous-section, nous montrons comment nos observations peuvent être intégrées dans les algorithmes existants pour en améliorer le fonctionnement. Pour ce faire, nous choisissons un algorithme représentatif des algorithmes de détection de communautés consensuelles existants et nous le modifions. Nous construisons également un algorithme générique qui utilise notre méthode de filtrage.

Soit l'algorithme 11, un algorithme générique mettant en œuvre notre filtre. Cet algorithme prend en entrée un graphe G, puis filtre G en utilisant l'ECC et notre corrélation, elle-même basée sur la modularité des n_p exécutions de $\mathscr{A}(G)$. Cet algorithme produit un graphe *filtré* G_L.

Ce graphe filtré G_L peut être utilisé en entrée pour d'autres algorithmes de détection de communautés consensuelles. Nous appelons FILTRE_GÉNÉRIQUE, l'approche qui consiste à exécuter $\mathscr{A}(G_L)$ (où les arêtes de G_L sont pondérées par leur coefficient de consensus).

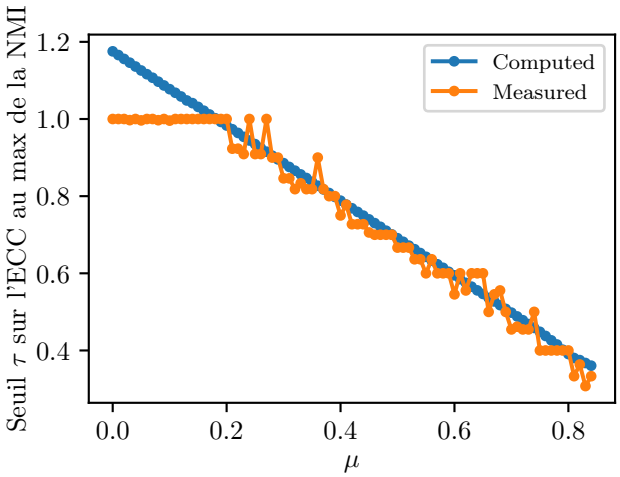


FIGURE 4.11 – Seuil sur l'ECC pour la NMI maximale tel qu'estimé avec le filtre présenté dans la Section 4.4 et seuil mesuré en remplissant la matrice de consensus pas à pas.

Algorithme 11 FILTRE (G, $\mathscr{A}$, n_p)

1: Exécuter n_p fois l'algorithme $\mathscr{A}$ sur G
2: Calculer la modularité moyenne Q des n_p partitions
3: Déduire un seuil τ de Q
4: Calculer l'ECC des paires de sommets (i, j) à distance ≤ 2
5: Construire une liste L de paires (i, j) pour lesquelles $ECC(i, j) > \tau$
6: Construire le graphe G_L dont les arêtes sont L. Le poids de chaque arête est le coefficient de consensus correspondant
7: **retourne** G_L

Considérons également l'algorithme ECG_Filtré qui améliore ECG [110]. Après avoir appliqué l'algorithme FILTRE sur G, nous donnons G_L en entrée à ECG, puis nous renvoyons les communautés consensuelles trouvées par ECG.

Nous étudions également l'algorithme Tandon_Filtré, qui construit G_L de la même manière, puis le donne en entrée à Tandon. Notons que puisque Tandon et ECG travaillent sur des graphes non pondérés, nous ne conservons pas les pondérations pour Tandon_Filtré et ECG_Filtré, puisqu'elles n'auraient aucun effet.

La section qui suit s'intéresse à l'étude de la performance de ces algorithmes.

4.5 Résultats expérimentaux

Tout d'abord, nous exécutons notre algorithme sur plusieurs graphes LFR et comparons la NMI ainsi que le temps d'exécution par rapport à d'autres algorithmes de l'état de l'art, puis nous répétons l'expérience sur des données réelles. Dans cette section, nous utilisons $\mathscr{A}$ = Louvain et $n_p = 64$ afin de paralléliser les exécutions de $\mathscr{A}$, notre machine possédant 64 cœurs.

Les détails de l'implémentation avec les corrélations et le correctif pour les graphes de petite taille se trouvent sur Software Heritage [63]. L'implémentation ayant permis de générer les figures 4.3 et 4.5 se trouve également sur Software Heritage [64].

4.5.1 Données synthétiques

Nous commençons par générer des graphes LFR avec 10 000 sommets et les mêmes paramètres que dans la Section 4.3.2, pour un μ variant de $0,4$ à $0,7$. La figure 4.12a montre la NMI en fonction de μ, tandis que la figure 4.13b montre le temps d'exécution en fonction de la taille du graphe (en nombre de sommets).

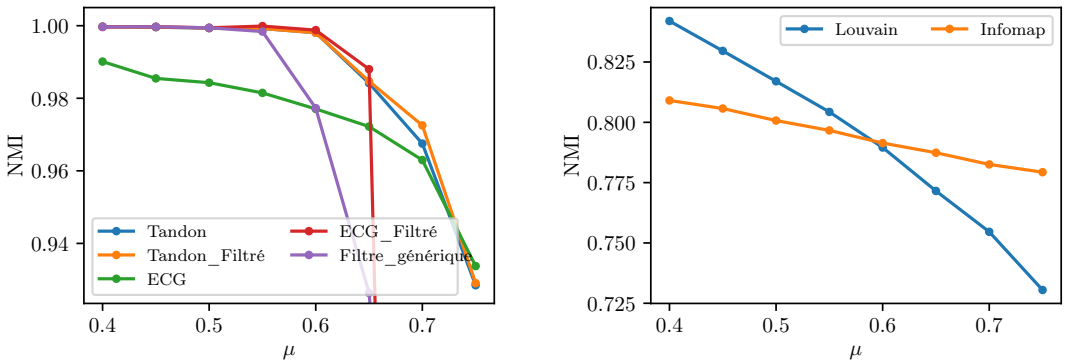
Nous comparons la version filtrée de Tandon et d'ECG à leur version classique. Nous évaluons également les performances de notre filtre générique, ainsi que celles des algorithmes de Louvain et Infomap comme point de référence.

Pour de faibles valeurs de μ, la version filtrée d'ECG fournit une NMI plus élevée, au prix d'un temps d'exécution plus élevé par rapport à l'algorithme ECG classique. Pour des valeurs élevées de μ, la NMI diminue rapidement avec des valeurs de 0,42 et 0,30 pour $\mu = 0,7$ et $0,75$ respectivement, ce qui est beaucoup plus faible que la version classique d'ECG.

Notre filtre générique donne une NMI plus élevée qu'ECG, mais est inférieure à celle de l'algorithme de Tandon. Cependant, il est beaucoup plus rapide, et utilise moins de mémoire, ce qui le rend utilisable sur des graphes plus grands.

Nous observons également une forte diminution de la NMI pour des valeurs élevées de μ, avec une NMI de 0,75 et 0,72 pour $\mu = 0,7$ et $0,75$ respectivement. Comme pour l'algorithme de Tandon, la version filtrée fournit une légère amélioration de la NMI, au prix d'un temps d'exécution légèrement plus élevé.

Il est à noter que, puisqu'une bonne partie du temps d'exécution est consacrée au remplissage de la matrice de consensus, la consommation de mémoire augmente avec le



(a) Différents algorithmes, filtrés ou non

(b) Louvain et Infomap

FIGURE 4.12 – NMI en fonction de μ pour des graphes LFR avec 10 000 sommets, pour différents algorithmes

temps d'exécution.

La figure 4.13a montre la NMI en fonction de la taille du graphe, pour des graphes LFR de 100 à un million de sommets, en fixant $\mu = 0,6$. Nous observons que la NMI a tendance à être plus faible pour les graphes de 100 sommets que pour les graphes plus grands.

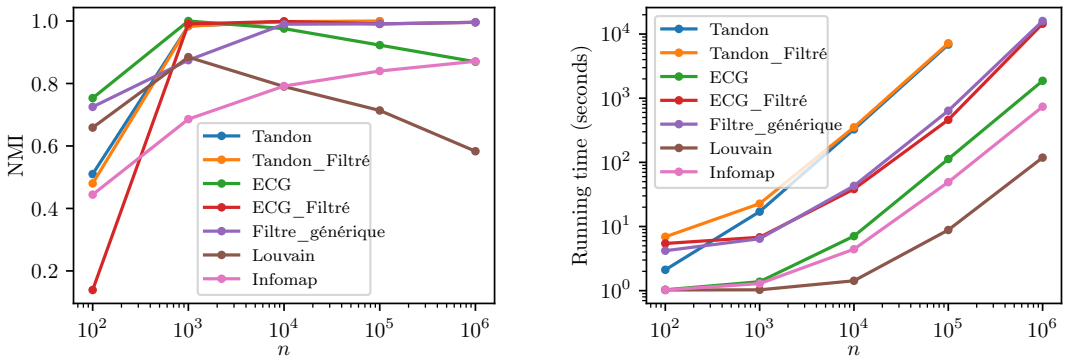
Même si les algorithmes classiques de détection de communautés sont limités par les problèmes de non-déterminisme mentionnés plus haut, nous avons également appliqué le même protocole à Louvain et Infomap. Comme nous pouvions nous y attendre, la NMI est beaucoup plus faible avec ces deux algorithmes, comme illustré sur la figure 4.12b.

4.5.2 Limites

Dans l'ensemble, la figure 4.12a montre que les algorithmes filtrés ont tendance à être plus performants que leur version non filtrée pour de faibles valeurs de μ, alors qu'ils ont tendance à être moins performants pour des valeurs de μ élevées. Nous pensons que cela est dû à une combinaison de plusieurs facteurs.

Au fur et à mesure que μ augmente, le nombre d'arêtes intra-communautaires diminue, tandis que le nombre d'arêtes entre les communautés augmente. Cela signifie que l'ECC des paires de sommets situés dans des communautés différentes aura également tendance à augmenter. Par conséquent, notre filtre est plus susceptible d'ajouter du bruit dans la matrice de consensus, à mesure que μ augmente.

Deuxièmement, toujours quand μ augmente, la modularité de la vérité terrain tend à diminuer, puisqu'il y a de moins en moins d'arêtes intra-communautaires. En conséquence, comme l'ont observé Aynaud *et al.* [4], un algorithme de détection de communautés qui se concentre sur la maximisation de la modularité peut trouver une partition avec



(a) NMI en fonction de n pour des graphes LFR avec $\mu = 0,6$, pour différents algorithmes

(b) Temps d'exécution en fonction de n, graphes LFR, $\mu = 0,6$, pour différents algorithmes

FIGURE 4.13 – NMI et temps d'exécution en fonction de la taille des graphes, pour différents algorithmes

une modularité plus élevée que celle de la vérité terrain. Des algorithmes de détection de communautés qui cherchent à maximiser la modularité ne peuvent donc pas fonctionner. Ce qui n'est pas forcément un problème, puisque pour un μ suffisamment élevé, il n'y a plus de structure communautaire, donc il serait cohérent de ne pas retrouver cette structure communautaire.

Enfin, la figure 4.7a montre la NMI en fonction du nombre d'entrées ajoutées dans la matrice de consensus, ainsi que la valeur de l'ECC de la dernière paire de sommet ajoutée à la matrice.

Pour de faibles valeurs de μ, nous observons une diminution rapide de l'ECC au niveau de la valeur maximale de la NMI. Cela permet une certaine imprécision dans notre corrélation, et notre seuil τ sur la valeur de l'ECC.

Par exemple, dans la figure 4.7a, tout τ compris entre 0,05 et 0,5 produira une NMI élevée. Tant que le seuil se situe dans la zone de forte décroissance de l'ECC, nous devrions être proches du maximum de la NMI. Cependant, à mesure que μ augmente, la décroissance tend à être moins marquée d'une part et à se produire après le pic de NMI, d'autre part. Ceci rend le seuil sur l'ECC plus difficile à déterminer, car une petite imprécision peut entraîner un grand changement dans le nombre d'entrées ajoutées à la matrice de consensus. Ainsi les communautés sont plus difficiles à trouver et le seuil doit être prédit de manière plus précise.

Selon Orman *et al.* [105], dans les graphes LFR avec μ supérieur à 0,5, les communautés sont moins bien définies. Lorsque μ augmente encore, nous obtenons un réseau sans échelle avec peu ou pas de structure communautaire.

Nous pourrions donc considérer que d'un côté, lorsque μ est très faible, la partition est très facile à trouver, et de l'autre côté, si μ est trop élevé, il n'y a plus de structure communautaire dans le graphe. Ainsi, l'important est de pouvoir trouver les communau-

tés lorsqu'elles existent et notre filtre se révèle pertinent pour ces valeurs intermédiaires de μ.

4.5.3 Données réelles

Nous avons vu que notre corrélation permettant d'arrêter le remplissage de la matrice de consensus fonctionne sur des graphes LFR. Cette corrélation fait apparaître le paramètre μ, qui est propre à ces graphes. Dans cette section, nous étudions le comportement de notre filtre sur des jeux de données réels.

Notons que nous n'avons pas pu tester notre méthode sur des jeux de données incluant des réseaux routiers car nous n'avons pas trouvé de réseaux disposant d'une vérité terrain connue. Nous aurions pu utiliser des zones fonctionnelles mais comme elles sont calculées à partir de communautés, la vérité terrain n'aurait pas été pertinente dans notre cas. Nous nous limitons donc à des jeux de données de graphes de terrain au sens large. Notre méthode de détection de communautés consensuelles pouvant s'inscrire dans une problématique plus large d'étude des graphes de terrain, ces jeux de données restent pertinents.

4.5.3.1 Jeu de données Football

Tout d'abord, nous étudions le jeu de données Football [53], où les sommets représentent des équipes de football américain et les arêtes représentent des matchs joués entre ces équipes. Les communautés de la vérité de terrain correspondent à la conférence dans laquelle l'équipe joue. Ce jeu de données compte 115 sommets, 613 arêtes, et 12 communautés. Les résultats sont donnés dans le tableau 4.1.

Nous constatons que notre filtre générique fournit la NMI la plus élevée et le temps d'exécution le plus faible, tandis que notre version filtrée d'ECG fournit une NMI plus élevée que l'algorithme ECG classique, mais avec également un temps d'exécution plus élevé. Nous comparons aussi ces performances à celles de deux algorithmes classiques de détection de communautés : la méthode de Louvain [15], qui fournit une NMI plus faible que les autres algorithmes testés, et l'algorithme Infomap [109], qui donne la deuxième NMI la plus élevée sur ce jeu de données.

La figure 4.14 montre la NMI ainsi que la dernière ECC ajoutée, en fonction du nombre d'entrées de la matrice de consensus P. Comme dans les figures 4.7a et 4.7d, nous observons une forte diminution de l'ECC lorsque la NMI est maximale.

4.5.3.2 Jeu de données DBLP

Le jeu de données du projet « Digital Bibliography & Library Project » (DBLP) [140] est un réseau de coauteurs où les sommets sont des auteurs et deux auteurs sont liés par une arête s'ils ont co-publié un article. Il s'agit d'un graphe plus grand que celui du jeu de données Football, avec 317 000 sommets et environ 1 million d'arêtes. Les communautés de la vérité de terrain sont les lieux de publication. Les résultats obtenus sur ce jeu de données sont synthétisés dans le tableau 4.1.

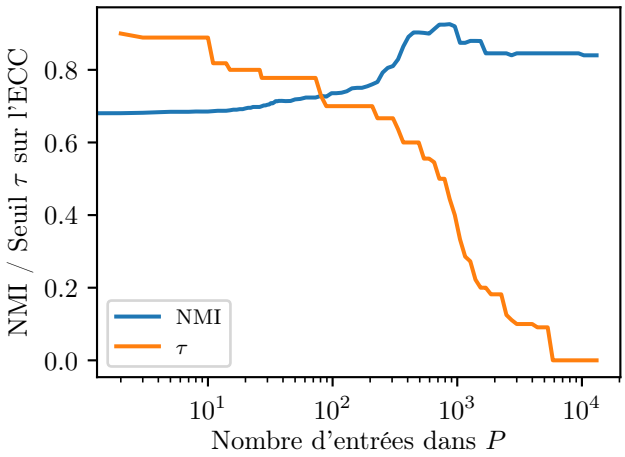


FIGURE 4.14 – NMI et seuil sur l'ECC en fonction du nombre d'entrées dans la matrice de consensus. Jeu de données Football

Tout d'abord, nous constatons une différence significative de temps d'exécution entre les différents algorithmes, allant de 12 secondes à 35 minutes. Les algorithmes *filtrés* sont les plus lents à cause du calcul de l'ECC et de la génération du graphe de consensus. La NMI la plus élevée est fournie par Infomap mais au prix d'un temps d'exécution élevé. La version filtrée d'ECG donne la deuxième NMI la plus élevée, mais a également un temps d'exécution élevé. Notre algorithme de filtrage générique est peu performant sur ces données, tandis que Louvain donne la NMI la plus basse mais est le plus rapide, de loin. Enfin, nous n'avons pas pu exécuter l'algorithme de Tandon car le temps d'exécution était trop élevé.

4.5.4 Discussion

Sur des jeux de données synthétiques, notre filtre permet de gagner en qualité, tant que les valeurs de μ ne sont pas trop élevées, soit pour $\mu < 0,65$ environ. Ceci se fait au prix d'un temps d'exécution plus important car notre filtre rajoute du temps de calcul à des méthodes de détection de communautés existantes.

Notre filtre est contre-productif pour des valeurs de μ élevées, pour plusieurs raisons. Il tend à filtrer les arêtes intra-communautaires, car pour des valeurs de μ élevées, ce sont les arêtes inter-communautaires qui ont une ECC élevée. De plus, la précision du critère d'arrêt doit être plus importante à mesure que μ augmente. Il n'est toutefois pas établi que les graphes ayant un μ élevé aient bien une réelle structure communautaire, donc ces résultats pourraient être cohérents.

Enfin, nos observations sont aussi valables pour les jeux de données réels Football et DBLP.

	Football		DBLP	
Algorithm	NMI	Temps	NMI	Temps
ECG	0,9079	1,10 s	0,6030	171 s
Tandon	0,8976	1,29 s	–	–
Tandon_Filter	0,8981	2,54 s	–	–
Generic_Filter	0,9354	1,49 s	0,4613	2236 s
ECG_Filter	0,9349	1,99 s	0,6288	1278 s
Louvain	0,8879	1,02 s	0,4769	12 s
Infomap	0,9242	1,12 s	0,7738	1305 s

TABLE 4.1 – NMI et temps d'exécution de différents algorithmes pour les jeux de données football et DBLP

4.6 Conclusion

La plupart des algorithmes de détection de communautés ne sont pas déterministes et il est souvent nécessaire d'agréger les résultats de plusieurs exécutions pour trouver un consensus, que l'on peut stocker dans une matrice de consensus. Nous avons montré que les informations contenues dans cette matrice peuvent être bruitées mais qu'il est possible de filtrer une partie de ce bruit. Nous avons ensuite utilisé ces observations pour améliorer les algorithmes de l'état de l'art et nous avons vérifié l'efficacité de notre approche sur des graphes synthétiques et réels. L'approche ECG_FILTRÉ tendance à produire une NMI plus élevée qu'ECG, au prix d'un temps d'exécution plus élevé. Notre FILTRE_GÉNÉRIQUE, fournit également une NMI élevée dans la plupart des cas, tout en passant mieux à l'échelle que l'algorithme de Tandon.

Nous pensons que notre filtre peut fournir des résultats encore meilleurs si notre corrélation entre le paramètre de mélange μ et le seuil sur l'ECC τ (équation 4.2) était plus précise. Pour une précision parfaite, il serait même envisageable de prouver cette corrélation pour le modèle ABCD (présenté dans le chapitre 2). De la même manière, notre équation permettant de corriger la modularité pour les graphes de petite taille (équation 4.3) repose sur un paramètre K dont la précision pourrait être encore améliorée en étudiant quel paramètre du modèle LFR influe sur cette valeur, puis en prouvant cette corrélation pour le modèle ABCD.

Enfin, notre filtre produit un graphe filtré dont les arêtes sont pondérées. Les algorithmes ECG et Tandon peuvent être améliorés pour prendre en compte cette pondération, et ainsi bénéficier de l'information contenue dans cette pondération pour produire des communautés de meilleure qualité.

Pour finir, comme mentionné dans la Section 4.5.2, nous observons que la performance de notre filtre chute pour des valeurs de μ élevées. Si cette chute apparaît pour des valeurs de μ pour lesquelles la structure communautaire n'existe plus, ce n'est pas forcément un problème, et pourrait même nous permettre de conclure quant à l'absence

de communautés.

La détection de communautés cherche les zones densément connectées d'un graphe. Cependant, dans le cadre d'un réseau routier, il est possible de se déplacer d'un sommet u vers un sommet v en empruntant un chemin reliant u à v, sans que les sommets u et v ne soient voisins. Ainsi, il peut être pertinent de considérer des communautés de sommets proches géographiquement plutôt que densément connectés. Le prochain chapitre explore la détection de ce type de communautés.

Chapitre 5

Communautés et graphes fermés

Sommaire

La détection de communautés permet d'identifier les zones densément connectées d'un graphe. Il a été montré que dans un graphe routier, ceci permet d'identifier les zones fonctionnelles, qui sont des zones géographiques dans lesquelles les trajectoires humaines ont tendance à rester [60]. Nous souhaitons explorer une définition légèrement différente des communautés en identifiant des groupes de sommets proches les uns des autres, plutôt que densément connectés. Nous pensons en effet que cette notion est plus adaptée à notre étude de la mobilité humaine.

Si deux sommets u et v ne sont pas connectés, il est toutefois possible de transiter de u vers v s'il existe un chemin de u vers v. Ainsi dans un graphe routier, il peut être pertinent de considérer des groupes de sommets géographiquement proches, plutôt que densément connectés.

Ce chapitre explore cette variante des communautés, ce qui à notre connaissance est un nouveau concept, qui n'a pas encore été étudié. Ce chapitre est donc exploratoire, et constitue un nouvel axe de recherche.

Après une introduction expliquant les enjeux de ces communautés, nous détaillons notre modèle, puis étudions les différences entre les communautés au sens classique, et cette nouvelle définition, puis étudions une manière efficace de les calculer.

5.1 Introduction

La structure communautaire d'un graphe représente des groupes de sommets entre lesquels il y a plus d'arêtes que vers le reste du graphe. Ceci sous-entend une interaction plus forte entre les sommes connectés qu'entre les sommets qui ne sont pas directement connectés par une arête.

Or, dans de nombreux cas, comme dans un graphe représentant un réseau routier, il est possible de se déplacer d'un croisement de rue à un autre, même si ces deux croisements ne sont pas directement reliés par une rue. L'étude de ce type de communautés dans les graphes routiers pourrait se révéler pertinente dans le cadre de la compréhension de la mobilité humaine, ainsi que pour notre problème de placement de capteurs évoqué au chapitre 3. En effet, il est probablement pertinent de prendre en compte les zones fonctionnelles dans le cadre de notre déploiement de capteurs.

Un graphe dans lequel deux sommets sont reliés par une arête si et seulement s'il existe un chemin entre ces deux sommets s'appelle un graphe *fermé transitivement*. Les arêtes sont pondérées pour indiquer la proximité entre deux sommets. La proximité est exprimée en terme de distance de graphe, ce qui peut représenter une distance géographique dans un graphe routier, ou encore une latence, une bande passante, ou autre, en fonction du graphe étudié.

Notons que dans le cas d'une rue d'une longueur importante, il est tout à fait possible que deux croisements de rues u et v, pourtant reliés par une arête, aient une distance plus élevée que deux croisements x et z qui ne seraient pas reliés par une arête, mais joignable au travers d'un troisième croisement y, pourvu que la longueur de (u, v) soit supérieure à celle de $(x, y) + (y, z)$.

Dans ce chapitre, nous étudions la structure communautaire d'un tel graphe fermé, ainsi que la possibilité de la trouver de manière efficace.

Une première section explique le modèle avec lequel nous travaillons. La Section 5.3, étudie les communautés de graphes fermés, d'abord en observant la différence entre les communautés du graphe fermé, et les communauté du graphe de départ, puis nous étudions la possibilité de détecter les communautés du graphe fermé, sans fermer totalement le graphe, en évitant ainsi une opération coûteuse.

5.2 Modèle

Considérons un graphe $G = (V, E)$, dans lequel il n'y a qu'une composante connexe, c'est-à-dire qu'il existe un chemin depuis n'importe quel sommet vers n'importe quel autre sommet.

Nous construisons la *fermeture transitive* de G, appelée G_f, en rajoutant toutes les arêtes possibles $(u, v) \in V^2$. Chaque arête (u, v) est pondérée en fonction de la longueur

du plus court chemin allant de u à v. Ainsi, la fermeture transitive pondérée conserve l'information relative à la distance entre deux sommets. Nous parlons par la suite de graphe *fermé*.

Notons que nous choisissons ici de pondérer notre graphe en fonction de la longueur du plus court chemin, mais n'importe quelle propriété sur les chemins pourrait convenir. Nous pourrions pondérer en fonction du nombre de plus courts chemins, ou s'il y a des débits sur les arêtes, nous pourrions pondérer par le minimum des débits le long d'un chemin.

Remarquons qu'un graphe G quelconque ne satisfait pas nécessairement l'inégalité triangulaire. C'est-à-dire qu'il peut exister un triangle (u, v, w) dans lequel le poids de (u, v) et (v, w) est 1 alors que le poids de l'arête (u, w) est 5. Par conséquent, le graphe fermé transitivement ne satisfait pas non plus cette inégalité, puisque nous ne faisons que rajouter des arêtes. En d'autres termes, nous ne sommes pas nécessairement dans un espace euclidien. Par conséquent, l'étape de calcul de la fermeture transitive est nécessaire, nous ne pouvons donc pas tirer parti de cette propriété.

5.3 Communautés et fermetures

Dans le graphe G_f fermé transitivement, les communautés représenteraient des groupes de croisements de rues proches les uns des autres. Nous nous intéressons au calcul des communautés sur ce graphe fermé transitivement.

Le calcul de la fermeture transitive est analogue à la résolution du problème du PLUS COURT CHEMIN ENTRE TOUTES LES PAIRES, qui peut être résolu en $O(n^2 \log n + nm)$ [71]. Étant donné la complexité d'un tel calcul, nous souhaitons calculer ces communautés sans passer par la fermeture transitive de notre graphe.

5.3.1 La modularité seule n'est pas adaptée

La modularité n'est pas adaptée car elle favorise les communautés dont les sommets sont densément connectés, alors que nous cherchons des communautés dont les sommets sont proches, au sens de la distance de graphes. Notons que dans le cas d'un graphe fermé transitivement, la modularité redevient pertinente puisqu'une zone densément connectée est une zone dans laquelle les sommets sont proches les uns des autres.

Des auteurs ont développé une extension à la méthode de Louvain, appelée LouvainSC (pour Louvain *spatially constrained*) [135]. Cette extension rajoute une opération permettant aux communautés de contenir uniquement des sommets contigus dans l'espace. Cette approche suppose de connaître les coordonnées des sommets dans l'espace et n'est pas tout à fait adaptée à notre cas. En effet, tous les sommets d'une même composante connexe seront connectés dans notre graphe fermé transitivement. De plus, nous souhaitons une approche générique, ne nécessitant pas de connaître les coordonnées des sommets dans l'espace.

Comparaison	NMI
Louvain(G) vs Louvain(G_f)	0,6862
Louvain(G) vs Vérité terrain	0,9363
Louvain(G_f) vs Vérité terrain	0,7199

TABLE 5.1 – NMI moyenne sur 100 exécutions de Louvain, pour des graphes LFR avec $\mu = 0,5$, et $n = 500$, arêtes pondérées par 1/spl

5.3.2 Comparaison des communautés du graphe de départ et du graphe fermé

Cette sous-section étudie la différence entre les communauté du graphe de départ G, et du graphe fermé G_f. Pour ce faire, nous générons un graphe LFR, avec $\mu = 0,5$, et $n = 500$, appelé G. Notons que puisque G est un graphe LFR, nous connaissons la vérité terrain de ses communautés. Nous calculons ensuite la fermeture transitive de G, G_f, comme expliqué en Section 5.2, puis nous effectuons des détections de communautés sur G et G_f, à l'aide de deux algorithmes : Louvain et Label Propagation. Nous comparons les communautés produites à l'aide de la NMI. Nous choisissons de répéter le calcul 100 fois et d'afficher une NMI moyenne pour corriger le problème du non-déterminisme des algorithmes utilisés.

Dans le tableau 5.1, les détections de communautés sont effectuées avec la méthode de Louvain, et les arêtes du graphe fermé G_f sont pondérées par l'inverse de la longueur du plus court chemin dans le graphe de départ G, pour que des sommets éloignés soient reliés par une arête de poids faible. Nous observons que la NMI entre Louvain(G) et Louvain(G_f) est plutôt faible, et en tout cas bien inférieur à la NMI entre Louvain(G) et la vérité terrain. Il semblerait que les communautés de G et de G_f soient différentes, et que les communautés issues de Louvain(G) soient les plus proches de la vérité terrain.

Dans le tableau 5.2, les détections de communautés sont de nouveau effectuées avec la méthode de Louvain. Nous générons une fermeture transitive de G, G_{f2}, sur le même modèle que G_f, à la différence que cette fois-ci, les arêtes sont pondérées par :

$$\frac{1}{2 \times \text{longueur du plus court chemin}}$$

Nous observons que la NMI entre Louvain(G) et la fermeture Louvain(G_{f2}) est supérieure à la même NMI dans le tableau 5.1, mais reste inférieure à la NMI entre Louvain(G) et la vérité terrain, qui reste comparable aux valeurs du tableau 5.1. À nouveau, la NMI entre Louvain(G_{f2}) et la vérité terrain est supérieure à la valeur correspondante dans le tableau 5.1.

Les tableaux 5.3 et 5.4 reprennent les mêmes expériences que les tableaux précédents, avec l'algorithme Label Propagation. À nouveau, nous observons une NMI moyenne faible entre Label propagation(G) et Label propagation(G_f), ce qui indique que les communautés présentent des différences non négligeables. De plus, pour les deux graphes G et G_f,

Comparaison	NMI
Louvain(G) vs Louvain(G_{f2})	0,8333
Louvain(G) vs Vérité terrain	0,9355
Louvain(G_{f2}) vs Vérité terrain	0,8514

TABLE 5.2 – NMI moyenne sur 100 exécutions de Louvain, pour des graphes LFR avec $\mu = 0,5$, et $n = 500$, arêtes pondérées par 1/2spl

Comparaison	NMI
Label propagation(G) vs Label propagation(G_f)	0,3499
Label propagation(G) vs Vérité terrain	0,5390
Label propagation(G_f) vs Vérité terrain	0,6781

TABLE 5.3 – NMI moyenne sur 100 exécutions de Label Propagation, pour des graphes LFR avec $\mu = 0,5$, et $n = 500$, arêtes pondérées par 1/spl

la NMI entre les communautés calculées et la vérité terrain est faible. Dans ce cas, l'algorithme Label Propagation ne présente pas de bonnes performances.

Puisque les communautés des graphes fermés sont différentes de celles des graphes de départ, nous souhaitons maintenant étudier la possibilité de les calculer en évitant l'opération couteuse de fermeture du graphe.

5.4 Convergence des communautés du graphe de départ

Dans cette sous-section, nous générons un graphe LFR G, que nous fermons transitivement, pour obtenir G_f. Nous effectuons ensuite ECG(G_f), et considérons le résultat de cette opération comme la vérité terrain, que nous cherchons à retrouver. Ensuite, nous fermons transitivement le graphe G petit à petit. Nous ne rajoutons qu'une partie des arêtes, à G, pour créer une fermeture partielle G_p, puis nous calculons ECG(G_p). Nous calculons ensuite la NMI entre ECG(G_p) et la vérité terrain. Nous poursuivons l'opération en ajoutant une autre partie des arêtes à G_p, puis calculons de nouveau ECG(G_p),

Comparaison	NMI
Label propagation(G) vs Label propagation(G_{f2})	0,4098
Label propagation(G) vs Vérité terrain	0,6274
Label propagation(G_{f2}) vs Vérité terrain	0,6830

TABLE 5.4 – NMI moyenne sur 100 exécutions de Label Propagation, pour des graphes LFR avec $\mu = 0,5$, et $n = 500$, arêtes pondérées par 1/2spl

puis la NMI. Nous itérons ainsi, jusqu'à ce que $G_p = G_f$, c'est-à-dire que G_p soit fermé transitivement. Nous étudions ainsi la possibilité de fermer partiellement un graphe pour calculer les communautés du graphe fermé.

Nous choisissons de rajouter les arêtes dans un ordre aléatoire, mais aussi selon des ordres bien définis, de manière à trouver un ordre qui offrirait un bon compromis entre le nombre d'arêtes à ajouter, et la qualité des communautés calculées.

5.4.1 ECC et distance

La figure 5.1a, représente la NMI en fonction du nombre d'arêtes dans la fermeture transitive partielle G_p. Le graphe de départ G est un graphe LFR, avec 200 sommets, $\mu = 0,3$. Nous avons choisi $n_p = 250$, c'est-à-dire que l'algorithme ECG effectue 250 itérations de Louvain, dans le but de limiter le non-déterminisme du résultat. Dans cette figure, les arêtes (u, v) sont rajoutées par ordre croissant de distance(u, v) dans le graphe de départ G. La NMI obtenue avec un ordre aléatoire sur les arêtes est aussi indiquée, et sert de point de comparaison. Nous observons qu'en ajoutant 5700 arêtes, soit 29% des arêtes que contient la fermeture transitive, il est possible d'obtenir une NMI de 0,78, quand la NMI obtenue en ajoutant les arêtes dans un ordre aléatoire est de 0,28, pour le même nombre d'arêtes. Pour atteindre une NMI de 1, il faut rajouter les 71% d'arêtes manquants à la fermeture.

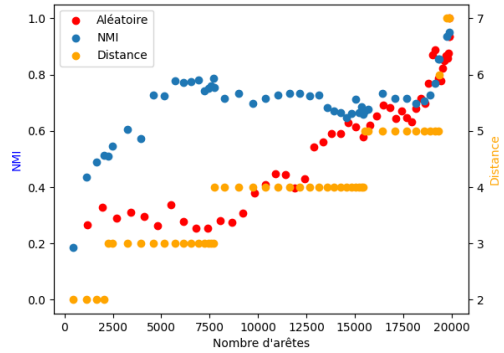
La figure 5.1b quant à elle représente la NMI en fonction du nombre d'arêtes dans la fermeture partielle G_p, quand les arêtes sont ajoutées dans l'ordre décroissant de distance(u, v) dans le graphe de départ. Dans ce cas, la NMI est très mauvaise, en tout cas inférieure à une fermeture partielle dans laquelle les arêtes sont ajoutées dans un ordre aléatoire.

Les figures 5.2a et 5.2b représentent (respectivement) la NMI en fonction du nombre d'arêtes dans la couverture partielle G_p, où les arêtes (u, v) sont ajoutées par ordre croissant et décroissant de l'ECC(u, v) dans le graphe de départ G. Nous observons qu'ajouter les arêtes par ordre décroissant de l'ECC est légèrement bénéfique du point de vue de la NMI. Une fois que la fermeture partielle atteint 8300 arêtes, soit 42% du nombre d'arête dans la fermeture totale, la NMI est de 0,71 contre 0,30 quand les arêtes sont ajoutées dans une ordre aléatoire.

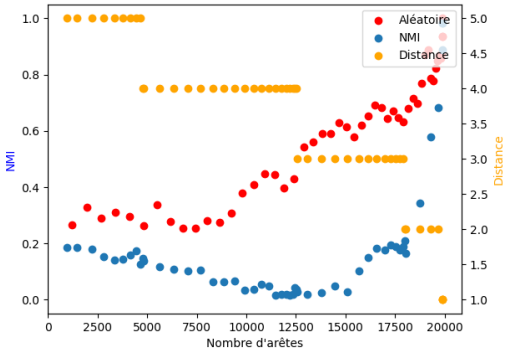
Nous observons qu'en ajoutant les arêtes par ordre croissant de la distance, ou par ordre décroissant de l'ECC, nous obtenons une NMI supérieure à celle que nous obtenons en ajoutant les arêtes dans un ordre aléatoire. La prochaine sous-section présente d'autres ordres d'ajout d'arêtes.

5.4.2 Autres mesures

Dans cette section, nous calculons des propriétés de sommets, comme le degré, ou des centralités comme la closeness ou la betweenness. Nous ordonnons ensuite les sommets en fonction de ces propriétés. Enfin, pour chaque sommet, nous rajoutons des arêtes vers tous les autres sommets du graphe. Une fois tous les sommets passés en revue, soit après n étapes, le graphe sera fermé transitivement.

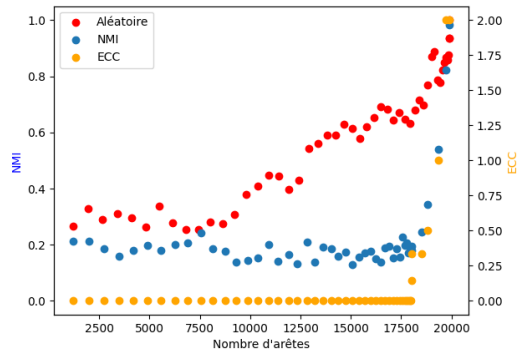


(a) Arêtes ajoutées par ordre croissant de leur distance dans G

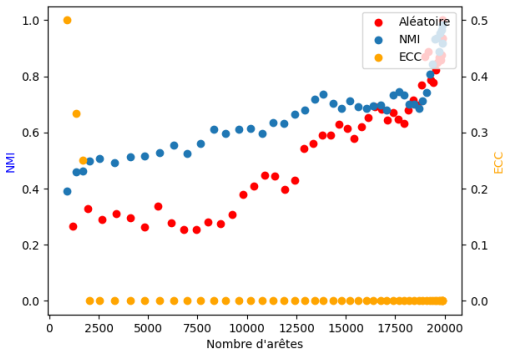


(b) Arêtes ajoutées par ordre décroissant de leur distance dans G

FIGURE 5.1 – NMI en fonction du nombre d'arêtes dans la fermeture transitive, où les arêtes sont ajoutées en fonction de la distance entre leurs extrémités dans le graphe de départ G

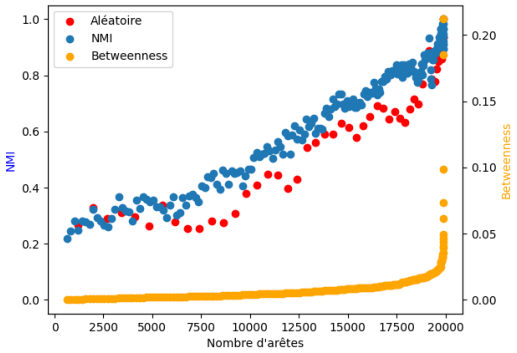


(a) Arêtes ajoutées par ordre croissant de leur ECC dans G

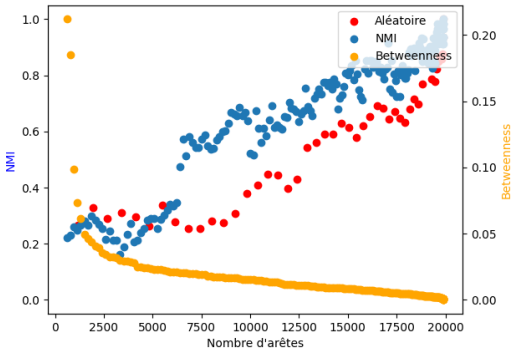


(b) Arêtes ajoutées par ordre décroissant de leur ECC dans G

FIGURE 5.2 – NMI en fonction du nombre d'arêtes dans la fermeture transitive, où les arêtes sont ajoutées en fonction de l'ECC leurs extrémités dans le graphe de départ G



(a) Arêtes ajoutées par ordre croissant de leur betweenness dans G



(b) Arêtes ajoutées par ordre décroissant de leur betweenness dans G

FIGURE 5.3 – NMI en fonction du nombre d'arêtes dans la fermeture transitive, arêtes ajoutées en fonction de leur betweenness dans le graphe de départ G

Les figures 5.3a et 5.3b ajoutent les sommets respectivement par ordre croissant, et décroissant de la betweenness. Nous n'observons pas de différence significative de NMI entre ces ordres et un ordre aléatoire.

Les figures 5.4 et 5.5 ajoutent les sommets respectivement par ordre de closeness, et par ordre de degré, sans que nous n'observions une différence de NMI notable par rapport à un ordre aléatoire.

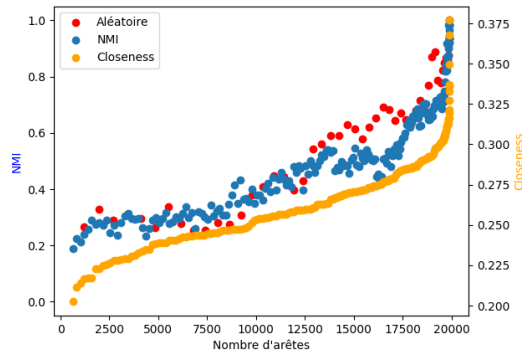
Nous observons que seules la distance et l'ECC fournissent des ordres permettant d'obtenir une bonne NMI sans calculer la fermeture transitive complète du graphe.

Jusqu'ici, notre graphe de départ G est non-pondéré, c'est-à-dire que toutes ses arêtes ont le même poids. Dans la réalité, un graphe représentant un réseau routier aurait ses arêtes pondérées par la longueur du tronçon routier, de même qu'un graphe représentant un réseau d'ordinateur aurait ses arêtes pondérées par la latence ou la bande passante disponible entre deux ordinateurs. La prochaine sous-section reprend notre étude, avec un graphe de départ G pondéré.

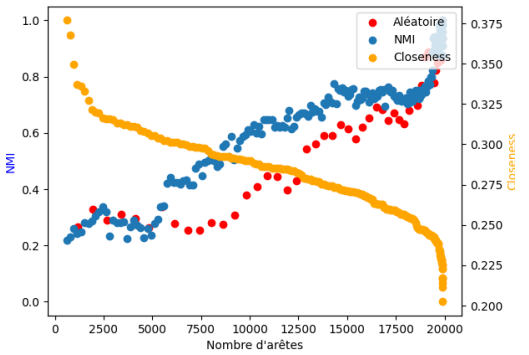
5.5 Pondération du graphe de départ

Nous choisissons ici de pondérer les arêtes de G pour mieux coller à la réalité. En effet, dans certains cas, comme par exemple dans le cas un réseau routier, et où la pondération des arêtes représenterait la longueur de chaque portion de rue, il peut être pertinent de considérer un graphe de départ pondéré. Ceci peut aussi être intéressant dans le cas d'un graphe représentant un réseau d'ordinateurs, où la pondération des arêtes correspondrait par exemple à la latence entre deux ordinateurs.

Pour pondérer nos arêtes, nous choisissons de calculer un plongement de notre graphe

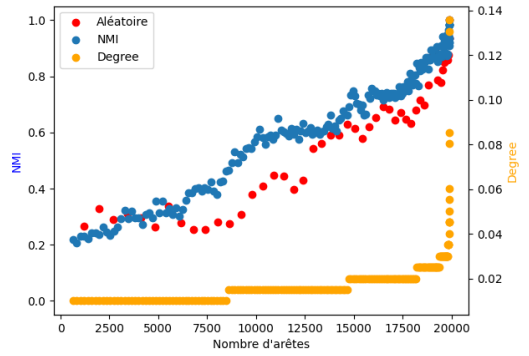


(a) Arêtes ajoutées par ordre croissant de leur closeness dans G

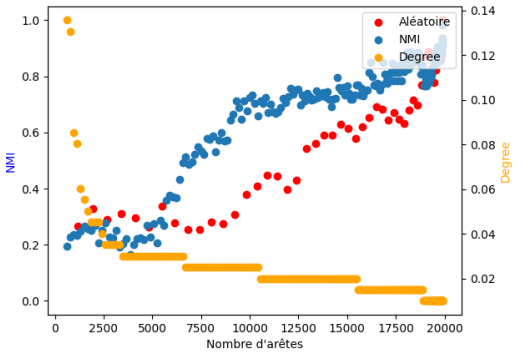


(b) Arêtes ajoutées par ordre décroissant de leur closeness dans G

FIGURE 5.4 – NMI en fonction du nombre d'arêtes dans la fermeture transitive, arêtes ajoutées en fonction de leur closeness dans le graphe de départ G



(a) Arêtes ajoutées par ordre croissant de leur degré dans G



(b) Arêtes ajoutées par ordre décroissant de leur degré dans G

FIGURE 5.5 – NMI en fonction du nombre d'arêtes dans la fermeture transitive, où les arêtes sont ajoutées en fonction de leur degré dans le graphe de départ G

dans un espace à deux dimensions. C'est-à-dire, nous affectons des coordonnées (x, y) à chaque sommet. Par la suite, nous calculons la distance euclidienne entre chaque extrémité d'arêtes, puis nous pondérons chaque arête par cette distance.

Pour calculer le plongement, nous utilisons l'algorithme de Fruchterman-Reingold [49], qui permet de positionner les sommets dans un espace à deux dimensions, en essayant de répartir uniformément les sommets dans l'espace, ainsi qu'en minimisant le nombre de croisements d'arêtes, et en essayant de garder la longueur des arêtes uniforme.

L'algorithme affecte ensuite une force d'attraction à chaque arête, rapprochant les sommets les uns des autres. L'algorithme affecte aussi une force de répulsion entre toutes paires de sommets, les empêchant d'être au même endroit. La force des arêtes est analogue à un ressort, tandis que la force des sommets est analogue à une charge électrique. L'algorithme essaie ensuite de minimiser l'énergie du système en déplaçant les sommets. Pour plus de détails, voir la publication originale [49].

De manière surprenante, dans ce contexte, tant que le graphe n'est pas complètement fermé, la NMI reste très faible, de l'ordre de 0,1, avant de croître très rapidement une fois le graphe quasiment fermé.

Ceci ne nous empêche pas de trouver des ordres permettant d'obtenir une meilleure NMI qu'avec un ordre d'ajout des arêtes aléatoire. La NMI reste toutefois trop faible pour qu'il soit intéressant de ne pas fermer le graphe complètement.

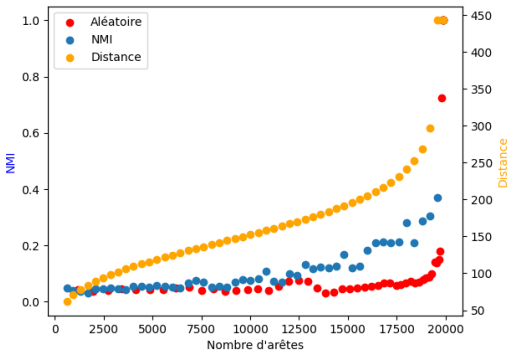
La figure 5.10a montre par exemple qu'il faut ajouter environ 15 000 arêtes, soit 75% des arêtes de la fermeture pour que la NMI soit supérieure à celle obtenue avec un ordre aléatoire. De plus, il suffit de rajouter 25% des arêtes pour passer d'une NMI de 0,2 à quasiment 1. Nous ne pouvons donc pas parler de bon compromis entre une fermeture partielle (plus rapide à calculer), et une bonne NMI. Des observations similaires peuvent être faites sur la figure 5.6a. Pour l'ECC, ainsi que les centralités betweenness et closeness (figures 5.7, 5.8, et 5.9), nous n'observons pas de différences de NMI significatives par rapport à l'ajout des arêtes dans un ordre aléatoire.

5.6 Conclusion

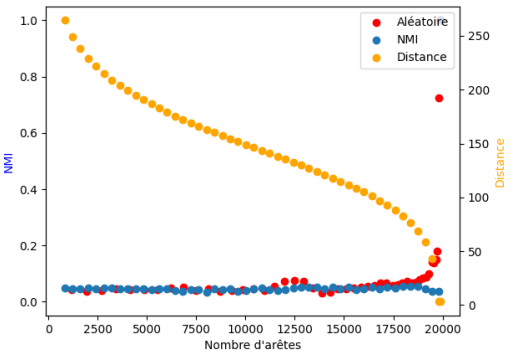
Dans ce chapitre, nous avons choisi d'explorer les communautés de graphes fermés transitivement car cette notion nous semble pertinente dans le cadre de l'identification de zones fonctionnelles dans les graphes routiers.

Nous avons observé que les communautés d'un graphe LFR G, et de sa fermeture transitive G_f sont différentes. Nous avons ensuite observé qu'une fermeture partielle G_p permet d'obtenir une bonne NMI. C'est le cas lorsque les arêtes (u, v) sont ajoutées par ordre croissant de distance(u, v), dans G, où il est possible d'ajouter 29% des arêtes de la fermeture, pour obtenir une NMI de 0,78, contre 0,28 lorsque les arêtes sont ajoutées dans un ordre aléatoire. Lorsque les arêtes sont ajoutées par ordre décroissant de l'ECC(u, v) dans G, il suffit d'ajouter 42% des arêtes de la fermeture pour obtenir une NMI de 0,71, contre 0,30 lorsque les arêtes sont ajoutées dans un ordre aléatoire.

Cette dernière étude ouvre un champ de perspectives nouvelles, autour de la possibilité d'avoir une méthode précise de calcul de ce type de communautés. Ces questions

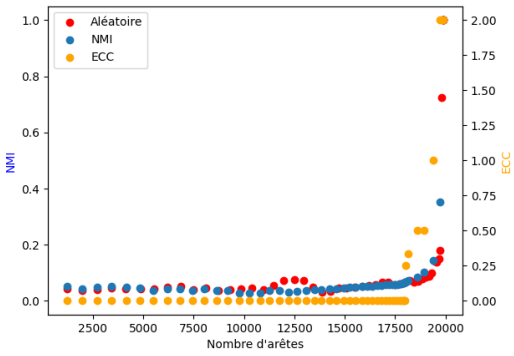


(a) Arêtes ajoutées par ordre croissant de leur distance dans G

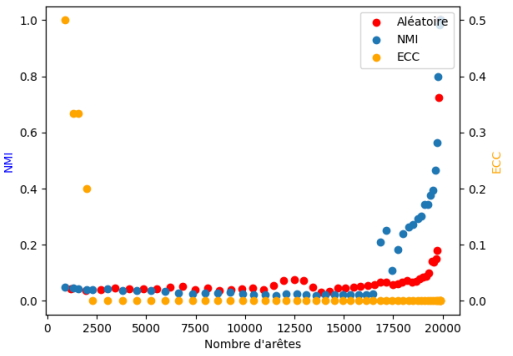


(b) Arêtes ajoutées par ordre décroissant de leur distance dans G

FIGURE 5.6 – NMI en fonction du nombre d'arêtes dans la fermeture transitive, où les arêtes sont ajoutées en fonction de la distance entre leurs extrémités dans le graphe de départ G, « Aléatoire » correspond à la NMI obtenue en ajoutant les arêtes dans un ordre aléatoire.

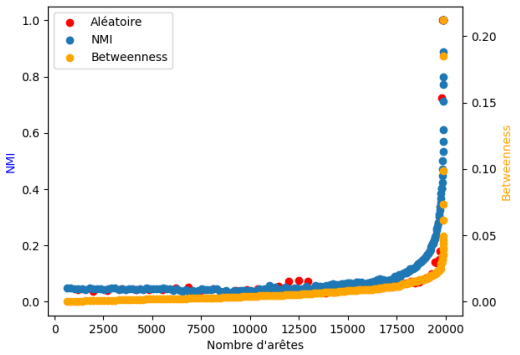


(a) Arêtes ajoutées par ordre croissant de leur ECC dans G

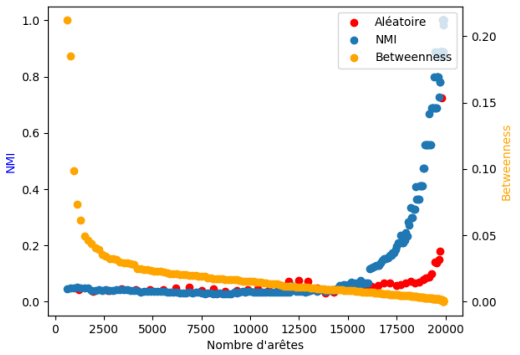


(b) Arêtes ajoutées par ordre décroissant de leur ECC dans G

FIGURE 5.7 – NMI en fonction du nombre d'arêtes dans la fermeture transitive, où les arêtes sont ajoutées en fonction de l'ECC leurs extrémités dans le graphe de départ G, « Aléatoire » correspond à la NMI obtenue en ajoutant les arêtes dans un ordre aléatoire.

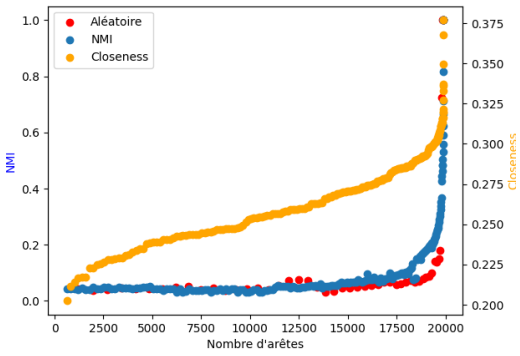


(a) Arêtes ajoutées par ordre croissant de leur betweenness dans G

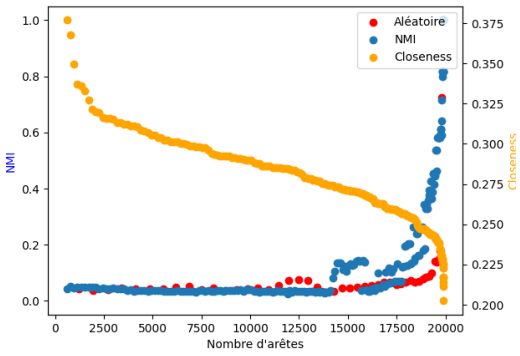


(b) Arêtes ajoutées par ordre décroissant de leur betweenness dans G

FIGURE 5.8 – NMI en fonction du nombre d'arêtes dans la fermeture transitive, où les arêtes sont ajoutées en fonction de leur betweenness dans le graphe de départ G, « Aléatoire » correspond à la NMI obtenue en ajoutant les arêtes dans un ordre aléatoire.

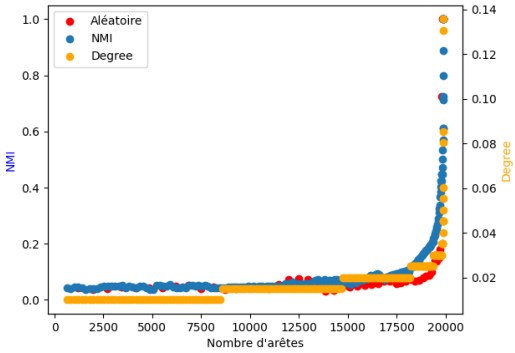


(a) Arêtes ajoutées par ordre croissant de leur closeness dans G

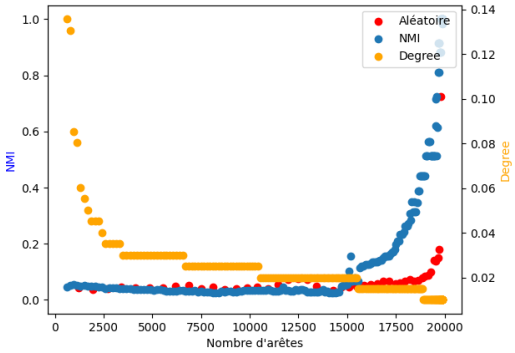


(b) Arêtes ajoutées par ordre décroissant de leur closeness dans G

FIGURE 5.9 – NMI en fonction du nombre d'arêtes dans la fermeture transitive, où les arêtes sont ajoutées en fonction de leur closeness dans le graphe de départ pondéré G, « Aléatoire » correspond à la NMI obtenue en ajoutant les arêtes dans un ordre aléatoire.



(a) Arêtes ajoutées par ordre croissant de leur degré dans G



(b) Arêtes ajoutées par ordre décroissant de leur degré dans G

FIGURE 5.10 – NMI en fonction du nombre d'arêtes dans la fermeture transitive, où les arêtes sont ajoutées en fonction de leur degré dans le graphe de départ G, « Aléatoire » correspond à la NMI obtenue en ajoutant les arêtes dans un ordre aléatoire.

étant des perspectives à moyen terme, elles seront abordées dans le chapitre suivant, consacré aux conclusions et perspectives de ces travaux de thèse.

Chapitre 6

Conclusion et perspectives

Sommaire

6.1 Conclusion

Dans cette thèse, nous nous sommes intéressés à la mobilité humaine dans un milieu urbain. Ce milieu est composé de rues et de croisements de rues, qui peuvent être représentées par un graphe routier. Nous avons étudié le lien entre la structure de ce graphe et les déplacements d'individus.

Nous avons d'abord cherché à mesurer cette mobilité pour créer des jeux de données réels. Nous avons ensuite étudié la structure des réseaux routiers au travers de la notion de communautés. Enfin, nous avons terminé en explorant une nouvelle définition de communautés qui semble mieux adaptée à l'étude de la mobilité dans les réseaux routiers.

6.1.1 Placement de capteurs

Dans un premier temps, nous nous sommes intéressés à la problématique du placement de capteurs dans le but de mesurer la mobilité d'individus dans une zone urbaine. Les données de ces capteurs permettent par la suite de reconstruire les trajectoires des individus, puis d'inférer des contacts entre ces individus.

Plutôt que de demander à des participants d'emporter avec eux une balise GPS, nous avons choisi d'utiliser des capteurs qui peuvent voir (et différencier) les individus. Cette

approche nous permet d'avoir un ensemble de participants représentatifs de la population, puisque nous n'avons pas besoin de sélectionner individuellement chaque personne, ce qui risquerait d'introduire un biais.

Nous avons cherché à positionner un nombre restreint de capteurs tout en essayant de récupérer un maximum de données. Pour ce faire, nous avons créé une heuristique basée sur le problème de la couverture par sommets, ainsi que des mesures de centralité, qui sont des notions issues de la théorie des graphes. Cette heuristique permet de décider où placer des capteurs sur un réseau routier, ainsi que de trouver un compromis entre le nombre de capteurs utilisés et la précision de la mesure des trajectoires.

Nous avons évalué la qualité des placements proposés par notre heuristique grâce à des jeux de données comme TAPASCologne qui simule le trafic routier dans la ville de Cologne sur une période de 24 heures. Nous avons aussi travaillé avec le jeu de données Bologne qui simule des déplacements denses de piétons à l'échelle de quelques rues et de quelques heures, comme il peut y en avoir à l'occasion de grands évènements comme des matchs de football.

Nous avons émis l'hypothèse qu'il est possible de reconstruire une trajectoire si au moins 20% des croisements de rues par lesquelles elle passe sont vus par un capteur. Dans ce contexte, sur le jeu de données TAPASCologne, nous avons observé qu'il est possible de reconstruire 95% des trajectoires en mettant des capteurs sur uniquement 16% des croisements de rues. Sur le jeu de données Bologne, il serait possible de reconstruire 99% des trajectoires en plaçant des capteurs sur 27% des croisements de rues.

Nous avons comparé nos résultats avec d'autres stratégies de placement de référence, avec ou sans connaissance préalable des trajectoires. Ceci nous a permis de comparer la performance de notre heuristique et d'observer qu'elle est plutôt proche des stratégies de placement ayant une connaissance préalable des trajectoires.

6.1.2 Communautés consensuelles

Les zones fonctionnelles d'une zone géographique correspondent aux zones dans lesquelles les trajectoires ont tendance à rester. Ces zones peuvent être détectées à l'aide de la notion de communautés. De plus, nous pensons que la connaissance de ces communautés nous permettra d'améliorer le placement de nos capteurs, par exemple en répartissant uniformément nos capteurs sur ces différentes zones fonctionnelles.

Nous avons donc étudié le problème de la détection de communautés qui consiste à retrouver les zones densément connectées d'un graphe. Les algorithmes actuels sont non déterministes, ce qui signifie qu'ils retournent une partition de communauté de qualité, mais arbitraire. Il n'est pas garanti que cette partition soit la plus représentative de la structure communautaire du graphe. Pour résoudre ce problème, il existe des algorithmes permettant de calculer des communautés de manière quasi déterministe. Ces communautés s'appellent des communautés consensuelles.

Nous avons remarqué que les algorithmes de détection de communautés consensuelles travaillent sur des données bruitées. Nous avons développé un filtre permettant de réduire le bruit dans le calcul des communautés consensuelles et ainsi de gagner en qualité. Ce

filtre vient en complément d'algorithmes existants. Il s'agit d'une étape de précalcul qui peut être effectuée avant un algorithme de détection de communautés consensuelles.

Nous avons étudié la performance de notre filtre sur des graphes générés, ainsi que sur des graphes de terrain (représentant des données réelles). Puisqu'il s'agit d'une étape de précalcul, notre filtre rajoute nécessairement du temps de calcul aux algorithmes existants.

Sur des données réelles, notre filtre permet d'améliorer la qualité des communautés.

Lorsque nous générons des graphes aléatoires avec le modèle LFR, nous pouvons choisir à quel point la structure communautaire est marquée. Quand celle-ci est bien définie, nous observons que notre filtre permet d'améliorer la qualité des communautés, alors que si la structure communautaire est faible, voire absente, notre filtre dégrade la qualité de la solution.

Nous observons néanmoins que les cas où notre filtre dégrade la solution semblent correspondre à des cas où les graphes présentent une structure communautaire si peu marquée qu'elle pourrait être considérée absente. Notre filtre ouvre donc la possibilité de conclure quant à l'absence de structure communautaire.

6.1.3 Communautés de graphes fermés transitivement

Nous avons par la suite considéré que les communautés qui nous intéressent dans le cadre de l'étude de la mobilité humaine dans une zone urbaine ne sont pas forcément les groupes de sommets densément connectés. Il serait plus pertinent de considérer les groupes de sommets proches les uns des autres, au sens de la distance dans un graphe G.

Nous avons donc étudié le problème de la détection de communautés dans un graphe *fermé transitivement*. La fermeture transitive G_f d'un graphe G correspond au graphe G dans lequel toutes les arêtes (u, v) ont été rajoutées, si et seulement s'il existe un chemin reliant le sommet u au sommet v. Nous avons choisi de pondérer les arêtes ainsi rajoutées par l'inverse de la distance séparant u de v dans G. Ainsi, deux sommets proches dans G sont reliés par une arête de poids important dans G_f et inversement.

Dans ce graphe fermé transitivement, grâce aux nouvelles arêtes et à leur pondération, les communautés correspondent à des groupes de sommets proches les uns des autres.

Dans un premier temps, nous avons étudié la différence entre les communautés d'un graphe G et de son graphe fermé G_f (pour des graphes LFR générés aléatoirement, avec une structure communautaire connue), et nous avons observé que les communautés sont différentes.

L'opération consistant à calculer la fermeture transitive de G est une opération couteuse en temps. Nous avons donc cherché à calculer les communautés du graphe fermé G_f sans toutefois calculer cette fermeture. Pour ce faire nous avons fermé progressivement le graphe G et étudié la qualité des communautés en fonction du nombre d'arêtes dans la fermeture partielle G_p.

Nos résultats montrent qu'en ajoutant les arêtes (u, v) par ordre de distance croissante entre u et v dans G, il suffit d'ajouter 29% des arêtes pour obtenir des communautés de bonne qualité.

Nos résultats permettront de créer une méthode de calcul pour la détection ce type de communautés efficacement en identifiant un critère d'arrêt permettant de décider quand arrêter l'étape de calcul de la fermeture transitive de G.

6.2 Perspectives

Dans le cadre du placement de capteurs pour la mesure de mobilité humaine, nous envisageons de déployer physiquement des capteurs avec lesquels nous pourrions collecter notre propre jeu de données de trajectoires. Nous avons pour l'instant déployé un nombre restreint de capteurs au sein de l'université de La Rochelle uniquement. Nous espérons pouvoir réaliser des déploiements à plus grande échelle par la suite.

Enfin, une fois qu'un premier ensemble de trajectoires est obtenu, nous pourrions les utiliser pour pondérer notre graphe routier et redéployer nos capteurs en prenant en compte ces trajectoires. Chaque arête du jeu de données serait ainsi pondérée par le nombre de trajectoires observées le long de cette arête. Ceci permet d'effectuer un deuxième déploiement avec une centralité pondérée comme la centralité Force. En supposant que les déplacements soient redondants à une certaine échelle de temps, les redéploiements pourraient nous aider à converger vers un meilleur placement de capteurs.

Dans le cadre de notre méthode de détection de communautés, nous pensons qu'une analyse plus approfondie des algorithmes de détection de communautés consensuelles existants serait pertinente. En effet, nous pensons que nos observations générales sont applicables pour la plupart des algorithmes qui utilisent une matrice de consensus et qui pourraient donc bénéficier d'un filtrage du bruit.

De plus, certains algorithmes comme la méthode LF effectuent la détection des communautés en plusieurs étapes. Ces algorithmes effectuent plusieurs détections de communautés (classiques), puis tant que la diversité parmi ces communautés est trop importante, une nouvelle matrice de consensus est générée, puis une nouvelle étape de détection de communautés est effectuée sur le graphe associé à cette matrice. Il pourrait être intéressant de filtrer le graphe à chaque itération puisque chaque matrice de consensus est potentiellement bruitée.

Certains algorithmes comme ECG ou Tandon considèrent des graphes non pondérés, alors que notre filtre génère des graphes pondérés. Une partie de l'information issue de notre filtre est donc perdue. La qualité des communautés pourrait être encore améliorée en modifiant ces algorithmes pour qu'ils prennent en compte cette pondération.

Enfin, notre filtre repose sur la corrélation observée entre la modularité Q et le paramètre de mélange μ des graphes LFR, puis entre μ et un seuil τ sur l'ECC. Nous aimerions prouver la correction de nos corrélations, ce qui permettrait de trouver un critère d'arrêt sur le remplissage de la matrice plus précis, ainsi que d'offrir des garanties théoriques à notre filtre. Pour ceci, le modèle ABCD pourrait être plus adapté que le modèle LFR, par sa simplicité. Nos corrélations reposent sur le paramètre μ, propre aux graphes LFR (et ABCD), cependant nous observons que nos corrélations sont aussi pertinentes dans le

cas de graphes de terrain, pourtant dépourvus de ce paramètre. Il peut être intéressant de mieux comprendre ces corrélations dans le cas des graphes de terrain.

De la même manière, nous avons mis en avant une corrélation permettant de corriger la modularité des graphes de petite taille en la modularité qui aurait pu être obtenue si le graphe avait été d'une taille plus importante. Cette corrélation repose sur un paramètre K. Pour une meilleure compréhension de la modularité, il serait intéressant d'étudier ce paramètre K et de comprendre s'il découle de paramètres du modèle LFR.

Enfin, dans le cadre de notre étude exploratoire de notre nouvelle définition de communautés, nous avons observé qu'il est possible de détecter les communautés d'un graphe fermé transitivement G_f, sans toutefois le fermer complètement. Un critère d'arrêt permettant d'identifier un compromis entre le temps passé à fermer partiellement le graphe de départ G et la qualité des communautés que nous calculons permettrait d'en déduire un algorithme. Nous pourrions envisager un critère ressemblant aux corrélations utilisées dans notre filtre.

Pour finir, il pourrait être pertinent d'étudier l'impact des communautés du graphe fermé sur le placement de capteurs. Nous prévoyons d'utiliser les outils développés dans le chapitre sur les communautés consensuelles. Ainsi, à l'aide de notre filtre, et sur une fermeture partielle, nous prévoyons d'étudier l'impact des communautés sur le placement de capteurs.

Cette thèse a permis de mettre en avant le lien entre structure du réseau routier et déplacement d'individus au sein de ce réseau : seule la structure du réseau permet de déduire où les personnes ont tendance à se déplacer. Certaines mesures permettent d'identifier des croisements de rues par lesquels les individus ont tendance à passer. Cette thèse a aussi permis d'améliorer certains outils nécessaires à l'étude de ces réseaux, comme les algorithmes de détection de communautés, avec la notion de filtre.

Enfin, cette thèse a mis en avant la nécessité de déployer des outils de théorie des graphes pour mieux comprendre les trajectoires dans les graphes de terrain. Poursuivre dans cette voie permettra de récolter des jeux de données de meilleure qualité, comme le propose le projet MITIK, ainsi que d'étudier la diffusion de l'information ou de maladies dans la population, qui sont actuellement deux enjeux majeurs de nos sociétés.

La notion de trajectoire est commune à d'autres types de déplacements comme un paquet dans l'Internet ou un utilisateur naviguant sur un site web. La connaissance de ces trajectoires présente les mêmes intérêts que dans les réseaux routiers comme l'aménagement du milieu dans lequel elles évoluent. Les outils développés dans cette thèse peuvent donc s'appliquer bien au-delà des réseaux routiers.

Publications

2022 Antoine Huchet, Jean-Loup Guillaume, Yacine Ghamri-Doudane
Sniffer deployment in urban area for human trajectory reconstruction and contact tracing

2023 Antoine Huchet, Jean-Loup Guillaume, Yacine Ghamri-Doudane
On filtering the noise in consensual communities

2023 Antoine Huchet, Jean-Loup Guillaume, Yacine Ghamri-Doudane
Filtering the noise in consensual community detection

2023 Antoine Huchet, Jean-Loup Guillaume, Yacine Ghamri-Doudane
Faites du bruit pour la détection de communautés consensuelles (mais pas trop) !

Bibliographie

[1] F. Akhter, S. Khadivizand, H. R. Siddiquei, M. E. E. Alahi et S. Mukhopadhyay. « IoT enabled intelligent sensor node for smart city : pedestrian counting and ambient monitoring ». In : *Sensors* 19.15 (2019), page 3374 (cf. page 53).

[2] K. Akkaya, I. Guvenc, R. Aygun, N. Pala et A. Kadri. « IoT-based occupancy monitoring techniques for energy-efficient smart buildings ». In : *2015 IEEE Wireless communications and networking conference workshops (WCNCW)*. IEEE. 2015, pages 58-63 (cf. pages 50, 51).

[3] S. Arnborg, D. G. Corneil et A. Proskurowski. « Complexity of finding embeddings in ak-tree ». In : *SIAM Journal on Algebraic Discrete Methods* 8.2 (1987), pages 277-284 (cf. page 25).

[4] T. Aynaud, V. D. Blondel, J.-L. Guillaume et R. Lambiotte. « Multilevel local optimization of modularity ». In : *Graph Partitioning* (2013), pages 315-345 (cf. page 107).

[5] Z. Bai, J. Tu et Y. Shi. « An improved algorithm for the vertex cover P_3 problem on graphs of bounded treewidth ». In : *Discrete Mathematics & Theoretical Computer Science* 21.Discrete Algorithms (2019) (cf. page 24).

[6] R. Bar-Yehuda et S. Even. « On approximating a vertex cover for planar graphs ». In : *Proceedings of the fourteenth annual ACM symposium on Theory of computing*. 1982, pages 303-309 (cf. pages 27, 28).

[7] A.-L. Barabási. « Network science ». In : *Philosophical Transactions of the Royal Society A : Mathematical, Physical and Engineering Sciences* 371.1987 (2013), page 20120375 (cf. page 37).

[8] A. Barrat, M. Barthelemy, R. Pastor-Satorras et A. Vespignani. « The architecture of complex weighted networks ». In : *Proceedings of the national academy of sciences* 101.11 (2004), pages 3747-3752 (cf. page 32).

[9] A. Bavelas. « A mathematical model for group structures ». In : *Applied Anthropology* 7.3 (Summer 1948) (1948), pages 16-30 (cf. page 32).

[10] A. Bavelas. « Communication patterns in task-oriented groups ». In : *The journal of the acoustical society of America* 22.6 (1950), pages 725-730 (cf. page 32).

[11] U. Bertele et F. Brioschi. *Nonserial dynamic programming*. Academic Press, Inc., 1972 (cf. page 24).

[12] S. Biaz, Y. Ji et P. Agrawal. « Impact of sniffer deployment on indoor localization ». In : *2005 International Conference on Collaborative Computing : Networking, Applications and Worksharing*. IEEE. 2005, 10-pp (cf. page 53).

[13] L. Bieker, D. Krajzewicz, A. Morra, C. Michelacci et F. Cartolano. « Traffic simulation for all : a real world traffic scenario from the city of Bologna ». In : *Modeling Mobility with Open Data*. Springer, 2015, pages 47-60 (cf. page 63).

[14] A. Björklund, T. Husfeldt, P. Kaski et M. Koivisto. « Narrow sieves for parameterized paths and packings ». In : *arXiv preprint arXiv :1007.1161* (2010) (cf. pages 30, 31).

[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte et E. Lefebvre. « Fast unfolding of communities in large networks ». In : *Journal of statistical mechanics : theory and experiment* 2008.10 (2008), P10008 (cf. pages 43, 86, 91, 109).

[16] V. D. Blondel, J.-L. Guillaume, R. Lambiotte et E. Lefebvre. « Fast unfolding of communities in large networks, 15 years later ». In : *To be published* (2023) (cf. page 43).

[17] H. L. Bodlaender. « A partial k-arboretum of graphs with bounded treewidth ». In : *Theoretical computer science* 209.1-2 (1998), pages 1-45 (cf. pages 24, 25, 32).

[18] H. L. Bodlaender et A. M. Koster. « Treewidth computations I. Upper bounds ». In : *Information and Computation* 208.3 (2010), pages 259-275 (cf. page 25).

[19] H. L. Bodlaender et A. M. Koster. « Treewidth computations II. Lower bounds ». In : *Information and Computation* 209.7 (2011), pages 1103-1119 (cf. page 25).

[20] P. Bonacich. « Technique for analyzing overlapping memberships ». In : *Sociological methodology* 4 (1972), pages 176-185 (cf. page 33).

[21] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski et D. Wagner. « Maximizing modularity is hard ». In : *arXiv preprint physics/0608255* (2006) (cf. page 38).

[22] B. Brešar, M. Jakovac, J. Katrenič, G. Semanišin et A. Taranenko. « On the vertex k-path cover ». In : *Discrete Applied Mathematics* 161.13-14 (2013), pages 1943-1949 (cf. page 30).

[23] B. Brešar, F. Kardoš, J. Katrenič et G. Semanišin. « Minimum k-path vertex cover ». In : *Discrete Applied Mathematics* 159.12 (2011), pages 1189-1195 (cf. page 30).

[24] M. Burgess, E. Adar et M. Cafarella. « Link-prediction enhanced consensus clustering for complex networks ». In : *PloS one* 11.5 (2016), e0153384 (cf. page 91).

[25] L. Calderoni, D. Maio et P. Palmieri. « Location-aware mobile services for a smart city : Design, implementation and deployment ». In : *Journal of theoretical and applied electronic commerce research* 7.3 (2012), pages 74-87 (cf. pages 50, 51).

[26] R. Campigotto, J.-L. Guillaume et M. Seifi. « The power of consensus : Random graphs have no communities ». In : *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 2013, pages 272-276 (cf. page 88).

[27] [Logiciel] G. A. Center, *Simulation of Urban MObility* 2020. url : `https://www.eclipse.org/sumo/`, swhid : ⟨`swh:1:rev:60662f8fcff1191008fc7a7462595c987876ad31`⟩ (cf. page 61).

[28] T. Chakraborty, S. Srinivasan, N. Ganguly, S. Bhowmick et A. Mukherjee. « Constant communities in complex networks ». In : *Scientific reports* 3.1 (2013), pages 1-9 (cf. page 90).

[29] J. Chen, I. A. Kanj et G. Xia. « Improved parameterized upper bounds for vertex cover ». In : *International symposium on mathematical foundations of computer science*. Springer. 2006, pages 238-249 (cf. page 29).

[30] L. Chen, X. Huang et Z. Zhang. « A simpler PTAS for connected k-path vertex cover in homogeneous wireless sensor network ». In : *Journal of Combinatorial Optimization* 36.1 (2018), pages 35-43 (cf. page 30).

[31] E. Cho, S. A. Myers et J. Leskovec. « Friendship and mobility : user movement in location-based social networks ». In : *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. URL : `https://snap.stanford.edu/data/loc-brightkite.html`. 2011, pages 1082-1090. url : `https://snap.stanford.edu/data/loc-gowalla.html` (cf. page 60).

[32] A. Chourasia, B. R. Tamma et A. A. Franklin. « Wi-Fi based Road Traffic Monitoring System with Channel Hopping Functionality ». In : *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE. 2021, pages 680-684 (cf. page 53).

[33] A. Clauset, M. E. Newman et C. Moore. « Finding community structure in very large networks ». In : *Physical review E* 70.6 (2004), page 066111 (cf. page 42).

[34] CRAWDAD. *CRAWDAD dataset epfl/mobility*. 2020. url : `http://crawdad.org/~crawdad/epfl/mobility/20090224/cab/` (visité le 27/10/2020) (cf. page 59).

[35] P. Crucitti, V. Latora et S. Porta. « Centrality in networks of urban streets ». In : *Chaos : an interdisciplinary journal of nonlinear science* 16.1 (2006), page 015113 (cf. page 32).

[36] M. O. Cruz, H. Macedo et A. Guimaraes. « Grouping similar trajectories for carpooling purposes ». In : *2015 Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE. 2015, pages 234-239. url : `https://archive.ics.uci.edu/ml/datasets/GPS+Trajectories#` (cf. page 59).

[37] F. De Montgolfier, M. Soto et L. Viennot. « Asymptotic modularity of some graph classes ». In : *Algorithms and Computation : 22nd International Symposium, ISAAC 2011, Yokohama, Japan, December 5-8, 2011. Proceedings 22*. Springer. 2011, pages 435-444 (cf. page 38).

[38] A. Decelle, F. Krzakala, C. Moore et L. Zdeborová. « Inference and Phase Transitions in the Detection of Modules in Sparse Networks ». In : *Physical Review Letters* 107.6 (août 2011). doi : 10.1103/physrevlett.107.065701. url : https://doi.org/10.1103%2Fphysrevlett.107.065701 (cf. page 46).

[39] J.-C. Delvenne, S. N. Yaliraki et M. Barahona. « Stability of graph communities across time scales ». In : *Proceedings of the national academy of sciences* 107.29 (2010), pages 12755-12760 (cf. page 38).

[40] J. Dibbelt, B. Strasser et D. Wagner. « Customizable contraction hierarchies ». In : *Journal of Experimental Algorithmics (JEA)* 21 (2016), pages 1-49 (cf. page 25).

[41] I. Dinur et S. Safra. « On the hardness of approximating minimum vertex cover ». In : *Annals of mathematics* (2005), pages 439-485 (cf. page 26).

[42] P. Erdős et T. Gallai. « On maximal paths and circuits of graphs ». In : *Acta Mathematica Academiae Scientiarum Hungarica* 10.3-4 (1959), pages 337-356 (cf. page 31).

[43] A. V. Esquivel et M. Rosvall. « Comparing network covers using mutual information ». In : *arXiv preprint arXiv :1202.0425* (2012) (cf. page 39).

[44] L. Euler. « Elementa doctrinae solidorum ». In : *Novi commentarii academiae scientiarum Petropolitanae* (1758), pages 109-140 (cf. page 29).

[45] F. V. Fomin, S. Gaspers, P. A. Golovach, D. Kratsch et S. Saurabh. « Parameterized algorithm for eternal vertex cover ». In : *Information Processing Letters* 110.16 (2010), pages 702-706 (cf. page 29).

[46] S. Fortunato. « Community detection in graphs ». In : *Physics reports* 486.3-5 (2010), pages 75-174 (cf. pages 41, 44, 91).

[47] S. Fortunato et M. Barthelemy. « Resolution limit in community detection ». In : *Proceedings of the national academy of sciences* 104.1 (2007), pages 36-41 (cf. page 38).

[48] L. C. Freeman. « A set of measures of centrality based on betweenness ». In : *Sociometry* (1977), pages 35-41 (cf. pages 33, 41).

[49] T. M. Fruchterman et E. M. Reingold. « Graph drawing by force-directed placement ». In : *Software : Practice and experience* 21.11 (1991), pages 1129-1164 (cf. page 122).

[50] I. GANG. *paris.osm-d.gr.gz*. url : https://files.inria.fr/gang/graphs/osm2015road/paris.osm-d.gr.gz (visité le 2022) (cf. page 25).

[51] M. R. Garey et D. S. Johnson. *Computers and intractability*. Tome 174. freeman San Francisco, 1979 (cf. page 27).

[52] M. R. Garey et D. S. Johnson. « The rectilinear Steiner tree problem is NP-complete ». In : *SIAM Journal on Applied Mathematics* 32.4 (1977), pages 826-834 (cf. page 29).

[53] M. Girvan et M. E. Newman. « Community structure in social and biological networks ». In : *Proceedings of the national academy of sciences* 99.12 (2002), pages 7821-7826 (cf. pages 41, 44, 46, 86, 109).

[54] B. Good, Y.-A. de Montjoye et A. Clauset. « Performance of modularity maximization in practical contexts ». In : *Phys. Rev. E* 81 (4 avr. 2010), page 046106. doi : 10.1103/PhysRevE.81.046106. url : https://link.aps.org/doi/10.1103/PhysRevE.81.046106 (cf. page 87).

[55] E. F. Grumert et A. Tapani. « Traffic state estimation using connected vehicles and stationary detectors ». In : *Journal of advanced transportation* 2018 (2018) (cf. page 53).

[56] S. Guha, R. Hassin, S. Khuller et E. Or. « Capacitated vertex covering ». In : *Journal of Algorithms* 48.1 (2003), pages 257-270 (cf. page 29).

[57] R. Guimera, M. Sales-Pardo et L. A. N. Amaral. « Modularity from fluctuations in random graphs and complex networks ». In : *Physical Review E* 70.2 (2004), page 025101 (cf. page 38).

[58] R. W. Hamming. « Error detecting and error correcting codes ». In : *The Bell system technical journal* 29.2 (1950), pages 147-160 (cf. pages 35, 97).

[59] P. W. Holland, K. B. Laskey et S. Leinhardt. « Stochastic blockmodels : First steps ». In : *Social networks* 5.2 (1983), pages 109-137 (cf. page 45).

[60] N. L. Houssou, J.-l. Guillaume et A. Prigent. « A graph based approach for functional urban areas delineation ». In : *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 2019, pages 652-658 (cf. pages 12, 13, 37, 80, 86, 113).

[61] L. Hubert et P. Arabie. « Comparing partitions ». In : *Journal of classification* 2 (1985), pages 193-218 (cf. page 40).

[62] [Logiciel] A. Huchet, *Tool to convert SUMO data for Networkx* 2021. swhid : ⟨swh:1:rev:2a33ec7013e22b6ee3aca92b5de5b93541e8ef07⟩ (cf. page 61).

[63] [Version de logiciel] A. Huchet, *Filter for consensual community detection* jan. 2023. url : https://gitlab.univ-lr.fr/ahuche01/pij_correlation, swhid : ⟨swh:1:cnt:4b787656631739f059695c3cc7cf7787812b4eae⟩ (cf. page 106).

[64] [Version de logiciel] A. Huchet, *Implementation of figures nmi vs number of entries* jan. 2023. url : https://gitlab.univ-lr.fr/ahuche01/consensual_communities, swhid : ⟨swh:1:snp:179cdb59337e4e6a7793f8f76ea759785cbf4938⟩ (cf. page 106).

[65] [Logiciel] A. Huchet, J.-L. Guillaume et Y. Ghamri-Doudane, *Sniffer placement method* 2021. swhid : ⟨swh:1:rev:7e21f8c59c3496ae8480bf0f6f0d97a92be038b4⟩ (cf. page 56).

[66] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft et C. Diot. « Pocket switched networks and human mobility in conference environments ». In : *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. 2005, pages 244-251 (cf. pages 50, 51).

[67] L. d'InfoRmatique en Image et Systèmes d'information. *Mobility datasets*. url : `https://privamov.github.io/accio/docs/datasets.html` (visité le 2022) (cf. page 58).

[68] iTETRIS. *iTETRIS Platform*. url : `http://www.ict-itetris.eu/` (visité le 22/06/2021) (cf. pages 60, 63).

[69] P. Jaccard. « Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines ». In : *Bull Soc Vaudoise Sci Nat* 37 (1901), pages 241-272 (cf. pages 35, 91, 97).

[70] Y. Ji, S. Biaz, S. Wu et B. Qi. « Optimal sniffers deployment on wireless indoor localization ». In : *2007 16th International Conference on Computer Communications and Networks*. IEEE. 2007, pages 251-256 (cf. page 53).

[71] D. B. Johnson. « Efficient algorithms for shortest paths in sparse networks ». In : *Journal of the ACM (JACM)* 24.1 (1977), pages 1-13 (cf. page 115).

[72] B. Kamiński, B. Pankratz, P. Prałat et F. Théberge. « Modularity of the ABCD random graph model with community structure ». In : *Journal of Complex Networks* 10.6 (2022), cnac050 (cf. page 102).

[73] B. Kamiński, P. Prałat et F. Théberge. « Artificial benchmark for community detection (ABCD)—fast random graph model with community structure ». In : *Network Science* 9.2 (2021), pages 153-178 (cf. page 45).

[74] R. M. Karp. « Reducibility among combinatorial problems ». In : *Complexity of computer computations*. Springer, 1972, pages 85-103 (cf. page 26).

[75] L. Katz. « A new status index derived from sociometric analysis ». In : *Psychometrika* 18.1 (1953), pages 39-43 (cf. page 33).

[76] M. G. Kendall. « The treatment of ties in ranking problems ». In : *Biometrika* 33.3 (1945), pages 239-251 (cf. page 78).

[77] B. W. Kernighan et S. Lin. « An efficient heuristic procedure for partitioning graphs ». In : *The Bell system technical journal* 49.2 (1970), pages 291-307 (cf. page 41).

[78] M. Kheirkhahzadeh et M. Analoui. « A Consensus Clustering Method for Clustering Social Networks ». In : *Statistics, Optimization & Information Computing* 8.1 (2020), pages 254-271 (cf. page 91).

[79] S. Khot. « On the power of unique 2-prover 1-round games ». In : *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 2002, pages 767-775 (cf. page 27).

[80] S. Khot et O. Regev. « Vertex cover might be hard to approximate to within 2- ε ». In : *Journal of Computer and System Sciences* 74.3 (2008), pages 335-349 (cf. page 27).

[81] N. Kiukkonen, J. Blom, O. Dousse, D. Gatica-Perez et J. Laurila. « Towards rich mobile phone datasets : Lausanne data collection campaign ». In : *Proc. ICPS, Berlin* 68 (2010). url : `https://www.idiap.ch/dataset/mdc/download` (cf. page 59).

[82] W. F. Klostermeyer et C. M. Mynhardt. « Edge protection in graphs. » In : *Australas. J Comb.* 45 (2009), pages 235-250 (cf. page 29).

[83] V. Krebs. *Unpublished*. 2004. url : `http://www.orgnet.com/` (cf. page 46).

[84] A. Lancichinetti. *andrealancichinetti - Benchmarks*. url : `https://sites.google.com/site/andrealancichinetti/benchmarks` (visité le 2022) (cf. page 95).

[85] A. Lancichinetti et S. Fortunato. « Community detection algorithms : a comparative analysis ». In : *Physical review E* 80.5 (2009), page 056117 (cf. page 44).

[86] A. Lancichinetti et S. Fortunato. « Consensus clustering in complex networks ». In : *Scientific reports* 2.1 (2012), pages 1-7 (cf. pages 86, 88, 90, 93).

[87] A. Lancichinetti, S. Fortunato et F. Radicchi. « Benchmark graphs for testing community detection algorithms ». In : *Physical review E* 78.4 (2008), page 046110 (cf. page 44).

[88] J. K. Laurila, D. Gatica-Perez, I. Aad, O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, M. Miettinen et al. *The mobile data challenge : Big data for mobile computing research*. Rapport technique. 2012. url : `https://www.idiap.ch/dataset/mdc/download` (cf. page 59).

[89] G. Lawyer. « Understanding the influence of all nodes in a network ». In : *Scientific reports* 5.1 (2015), pages 1-9 (cf. page 34).

[90] J. Leskovec et A. Krevl. *SNAP Datasets : Stanford Large Network Dataset Collection*. `http://snap.stanford.edu/data`. Juin 2014 (cf. page 46).

[91] Z.-W. Liang, J.-P. Li, F. Yang et A. Petropulu. « Detecting community structure using label propagation with consensus weight in complex network ». In : *Chinese Physics B* 23.9 (2014), page 098902 (cf. page 90).

[92] T. Lindner, L. Fritsch, K. Plank et K. Rannenberg. « Exploitation of Public and Private WiFi Coverage for New Business Models ». In : *Building the E-Service Society : E-Commerce, E-Business, and E-Government*. Springer. 2004, pages 131-148 (cf. page 53).

[93] Q. Liu, Z. Hou et J. Yang. « Detecting Spatial Communities in Vehicle Movements by Combining Multi-Level Merging and Consensus Clustering ». In : *Remote Sensing* 14.17 (2022), page 4144 (cf. page 91).

[94] D. Mandaglio, A. Amelio et A. Tagarelli. « Consensus community detection in multilayer networks using parameter-free graph pruning ». In : *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2018, pages 193-205 (cf. page 90).

[95] [Version de logiciel] S. Maniu et P. Senellart, *Treewidth* 2020. URL : `https://github.com/smaniu/treewidth`, SWHID : ⟨`swh:1:rev:7976f32b33189cd6ccdbd1d7b0518aff542fc888`⟩ (cf. page 25).

[96] S. Maniu, P. Senellart et S. Jog. « An Experimental Study of the Treewidth of Real-World Graph Data (Extended Version) ». In : *arXiv preprint arXiv :1901.06862* (2019) (cf. page 25).

[97] M. Meilă. « Comparing clusterings—an information based distance ». In : *Journal of multivariate analysis* 98.5 (2007), pages 873-895 (cf. page 38).

[98] E. Miyano, T. Saitoh, R. Uehara, T. Yagita et T. C. van der Zanden. « Complexity of the maximum k-path vertex cover problem ». In : *International Workshop on Algorithms and Computation*. Springer. 2018, pages 240-251 (cf. page 30).

[99] M. Molloy et B. Reed. « A critical point for random graphs with a given degree sequence ». In : *Random structures & algorithms* 6.2-3 (1995), pages 161-180 (cf. page 45).

[100] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira et L. Damas. « Predicting taxi–passenger demand using streaming data ». In : *IEEE Transactions on Intelligent Transportation Systems* 14.3 (2013), pages 1393-1402. URL : `https://archive.ics.uci.edu/ml/datasets/Taxi+Service+Trajectory+-+Prediction+Challenge,+ECML+PKDD+2015` (cf. page 59).

[101] R. R. Nadakuditi et M. E. J. Newman. « Graph Spectra and the Detectability of Community Structure in Networks ». In : *Physical Review Letters* 108.18 (mai 2012). DOI : `10.1103/physrevlett.108.188701`. URL : `https://doi.org/10.1103%2Fphysrevlett.108.188701` (cf. page 46).

[102] M. E. Newman. « Fast algorithm for detecting community structure in networks ». In : *Physical review E* 69.6 (2004), page 066133 (cf. page 42).

[103] M. E. Newman. « The mathematics of networks ». In : *The new palgrave encyclopedia of economics* 2.2008 (2008), pages 1-12 (cf. page 33).

[104] M. E. Newman et M. Girvan. « Finding and evaluating community structure in networks ». In : *Physical review E* 69.2 (2004), page 026113 (cf. page 37).

[105] G. K. Orman et V. Labatut. « A comparison of community detection algorithms on artificial networks ». In : *Discovery Science : 12th International Conference, DS 2009, Porto, Portugal, October 3-5, 2009 12*. Springer. 2009, pages 242-256 (cf. page 108).

[106] G. K. Orman, V. Labatut et H. Cherifi. « Comparative evaluation of community detection algorithms : a topological approach ». In : *Journal of Statistical Mechanics : Theory and Experiment* 2012.08 (2012), P08001 (cf. page 40).

[107] C. H. Papadimitriou et K. Steiglitz. *Combinatorial optimization : algorithms and complexity*. Courier Corporation, 1998 (cf. page 27).

[108] T. P. Peixoto. « Revealing consensus and dissensus between network partitions ». In : *Physical Review X* 11.2 (2021), page 021003 (cf. page 88).

[109] P. Pons et M. Latapy. « Computing communities in large networks using random walks ». In : *International symposium on computer and information sciences*. Springer. 2005, pages 284-293 (cf. pages 42, 86, 109).

[110] V. Poulin et F. Théberge. « Ensemble clustering for graphs ». In : *International Conference on Complex Networks and their Applications*. Springer. 2018, pages 231-243 (cf. pages 91, 93, 94, 97, 106).

[111] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto et D. Parisi. « Defining and identifying communities in networks ». In : *Proceedings of the national academy of sciences* 101.9 (2004), pages 2658-2663 (cf. pages 36, 97).

[112] U. N. Raghavan, R. Albert et S. Kumara. « Near linear time algorithm to detect community structures in large-scale networks ». In : *Physical review E* 76.3 (2007), page 036106 (cf. pages 43, 90).

[113] W. M. Rand. « Objective criteria for the evaluation of clustering methods ». In : *Journal of the American Statistical association* 66.336 (1971), pages 846-850 (cf. page 39).

[114] J. Rasero, M. Pellicoro, L. Angelini, J. M. Cortes, D. Marinazzo et S. Stramaglia. « Consensus clustering approach to group brain connectivity matrices ». In : *Network Neuroscience* 1.3 (2017), pages 242-253 (cf. page 91).

[115] M. Rosvall, D. Axelsson et C. T. Bergstrom. « The map equation ». In : *The European Physical Journal Special Topics* 178.1 (2009), pages 13-23 (cf. page 86).

[116] M. Rosvall et C. T. Bergstrom. « Maps of random walks on complex networks reveal community structure ». In : *Proceedings of the national academy of sciences* 105.4 (2008), pages 1118-1123 (cf. page 43).

[117] D. N. Sandesh Uppoor et M. Fiore. *TAPAS Cologne project dataset*. `https://sourceforge.net/projects/sumo/files/traffic_data/scenarios/TAPASCologne/`. Accessed on the 3rd of November 2020 (cf. pages 60, 61).

[118] M. Seifi et J.-L. Guillaume. « Community cores in evolving networks ». In : *Proceedings of the 21st International Conference on World Wide Web*. 2012, pages 1173-1180 (cf. pages 87, 88).

[119] M. Seifi, I. Junier, J.-B. Rouquier, S. Iskrov et J.-L. Guillaume. « Stable community cores in complex networks ». In : *Complex Networks*. Springer, 2013, pages 87-98 (cf. page 88).

[120] C. E. Shannon. « A mathematical theory of communication ». In : *ACM SIGMOBILE mobile computing and communications review* 5.1 (2001), pages 3-55 (cf. page 38).

[121] M. Şimşek et H. Meyerhenke. « Combined centrality measures for an improved characterization of influence spread in social networks ». In : *Journal of Complex Networks* 8.1 (2020), cnz048 (cf. page 78).

[122] A. Singhal et al. « Modern information retrieval : A brief overview ». In : *IEEE Data Eng. Bull.* 24.4 (2001), pages 35-43 (cf. page 36).

[123] M. Sipser. « Introduction to the theory of computation. 3-rd edition ». In : *Gengage learning* (2013) (cf. page 12).

[124] K. Stephenson et M. Zelen. « Rethinking centrality : Methods and examples ». In : *Social networks* 11.1 (1989), pages 1-37 (cf. page 33).

[125] A. Strehl et J. Ghosh. « Cluster ensembles—a knowledge reuse framework for combining multiple partitions ». In : *Journal of machine learning research* 3.Dec (2002), pages 583-617 (cf. page 39).

[126] A. Tandon, A. Albeshri, V. Thayananthan, W. Alhalabi et S. Fortunato. « Fast consensus clustering in complex networks ». In : *Physical Review E* 99.4 (2019), page 042301 (cf. pages 91, 92, 94, 97).

[127] A. Topchy, A. K. Jain et W. Punch. « Clustering ensembles : Models of consensus and weak partitions ». In : *IEEE transactions on pattern analysis and machine intelligence* 27.12 (2005), pages 1866-1881 (cf. page 93).

[128] V. A. Traag, L. Waltman et N. J. Van Eck. « From Louvain to Leiden : guaranteeing well-connected communities ». In : *Scientific reports* 9.1 (2019), page 5233 (cf. page 43).

[129] J. Tu et W. Zhou. « A factor 2 approximation algorithm for the vertex cover P3 problem ». In : *Information Processing Letters* 111.14 (2011), pages 683-686 (cf. page 30).

[130] S. Uppoor et M. Fiore. « MobiCom 2011 poster : vehicular mobility in large-scale urban environments ? » In : *ACM SIGMOBILE Mobile Computing and Communications Review* 15.4 (2012), pages 55-57. URL : `http://kolntrace.project.citi-lab.fr/` (cf. pages 60, 61).

[131] S. Uppoor, O. Trullols-Cruces, M. Fiore et J. M. Barcelo-Ordinas. « Generation and analysis of a large-scale urban vehicular mobility dataset ». In : *IEEE Transactions on Mobile Computing* 13.5 (2013), pages 1061-1075. URL : `http://kolntrace.project.citi-lab.fr/` (cf. pages 60, 61).

[132] M. P. Viana, J. L. Batista et L. d. F. Costa. « Effective number of accessed nodes in complex networks ». In : *Physical Review E* 85.3 (2012), page 036105 (cf. page 34).

[133] G. Voigt. « Tree Decompositions, Treewidth, and NP-Hard Problems ». In : () (cf. page 25).

[134] K. Wakita et T. Tsurumi. « Finding community structure in mega-scale social networks ». In : *Proceedings of the 16th international conference on World Wide Web*. 2007, pages 1275-1276 (cf. page 42).

[135] C. Wang, F. Wang et T. Onega. « Network optimization approach to delineating health care service areas : Spatially constrained Louvain and Leiden algorithms ». In : *Transactions in GIS* 25.2 (2021), pages 1065-1081 (cf. page 115).

[136] Q. Wang et E. Fleury. « Detecting overlapping communities in graphs ». In : *European Conference on Complex Systems 2009 (ECCS 2009)*. 2009 (cf. page 91).

[137] I. Wegener. *Complexity theory : exploring the limits of efficient algorithms*. Springer Science & Business Media, 2005 (cf. page 12).

[138] D. P. Williamson et D. B. Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011 (cf. page 24).

[139] W. Yahui et G. Xiaoran. « The study of location technology based on wireless sensor networks in smart city ». In : *2016 12th IEEE International Conference on Control and Automation (ICCA)*. IEEE. 2016, pages 848-853 (cf. page 53).

[140] J. Yang et J. Leskovec. « Defining and evaluating network communities based on ground-truth ». In : *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*. 2012, pages 1-8 (cf. pages 46, 109).

[141] L. Yang, Z. Yu, J. Qian et S. Liu. « Overlapping community detection using weighted consensus clustering ». In : *Pramana* 87.4 (2016), pages 1-6 (cf. page 91).

[142] W. W. Zachary. « An information flow model for conflict and fission in small groups ». In : *Journal of anthropological research* 33.4 (1977), pages 452-473 (cf. page 46).

[143] Y. Zheng. *T-Drive trajectory data sample*. T-Drive sample dataset. Août 2011. URL : `https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/` (cf. page 60).

[144] Y. Zheng, H. Fu, X. Xie, W.-Y. Ma et Q. Li. *Geolife GPS trajectory dataset - User Guide*. Geolife GPS trajectories 1.1. Geolife GPS trajectories 1.1. Juill. 2011. URL : `https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/` (cf. page 59).

Algorithmique de graphes appliquée à la mesure passive de la mobilité dans les réseaux routiers

Résumé : L'étude de la mobilité humaine en milieu urbain présente de nombreux intérêts comme l'aménagement urbain, l'étude de la diffusion de l'information ou la propagation de maladies. Nous choisissons d'étudier cette mobilité au travers d'outils issus de la théorie des graphes.
Nous développons une méthodologie permettant de mesurer la mobilité humaine, basée sur des capteurs qui peuvent voir, et différencier, les individus. Nous développons une heuristique permettant de décider où placer ces capteurs pour observer un maximum de déplacements.
Les trajectoires d'individus ont tendance à rester dans les mêmes zones géographiques, appelées *zones fonctionnelles*. Nous étudions les outils de détection de communautés dans les graphes routiers, dans le but de détecter automatiquement ces zones fonctionnelles. Nous présentons une méthode permettant d'identifier des communautés de meilleurs qualité.
Enfin, nous présentons une nouvelle définition de communauté qui est plus pertinente dans le cas de réseaux routiers comme d'autres types de réseaux. Nous explorons les différences entre cette nouvelle définition et la définitions classique, puis explorons des pistes permettant d'identifier ces communautés efficacement.

Mots clés : Mobilité, trajectoires, graphes, communautés

Graph algorithms applied to passive mobility measurement in road networks

Abstract :
The study of human mobility in an urban environment has many purposes, such as urban planning, the study of information dissemination or the spread of diseases. We have chosen to study this mobility using tools from graph theory.
We develop a methodology to measure human mobility, based on sensors that can see and differentiate individuals. We develop a heuristic to decide where to place those sensors to see as many trajectories as possible, while limiting the number of sensors to be deployed.
The trajectories of individuals tend to stay within the same geographical areas, called *functional urban areas*. We study tools for detecting communities in road graphs, with the aim of automatically detecting these functional areas. We present a method to improve the quality of communities.
Finally, we present a new definition of community that is more relevant in the case of road networks. We explore the differences between this new definition and the classical one, and then explore ways of identifying these communities efficiently.

Keywords : Mobility, trajectories, graphs, communities

Laboratoire Informatique, Image, Interaction
Institut LUDI - La Rochelle Université
Avenue Michel Crépeau

17042 LA ROCHELLE CEDEX 1