Classification spectrale pour la gestion de la congestion routière

Pamela Al Alam

Thèse de Doctorat

Mention : Sciences et Technologies de l'Information et de la communication
Spécialité : Traitement du Signal et des Images

présentée à *l'École Doctorale en Sciences Technologie et Santé (ED 585)*

de l'Université du Littoral Côte d'Opale

et

à *l'École Doctorale des Sciences et Technologie*

de l'Université Libanaise

par

AL ALAM PAMELA

pour obtenir le grade de Docteur de l'Université du Littoral Côte d'Opale

Classification spectrale pour la gestion de la congestion routière

Soutenue le 06 Septembre 2021, après avis des rapporteurs, devant le jury d'examen :

M. Kifah TOUT, Professeur, Université Libanaise	**Président**
M. Yassine RUICHEK, Professeur, Université UTBM	**Rapporteur**
M. Louahdi KHOUDOUR, Directeur de recherches, CEREMA à Toulouse	**Rapporteur**
M. Ghaleb FAOUR, Directeur de recherches, CNRS-Liban	**Examinateur**
M^{me} Marwa EL BOUZ, Enseignante-Chercheuse, Yncréa-Ouest	**Examinateur**
M. Denis HAMAD, Professeur, Université du Littoral Côte d'Opale	**Directeur de thèse**
M. Youssef ZAATAR, Professeur, Université Libanaise	**Directeur de thèse**
M. Joseph CONSTANTIN, Professeur, Université Libanaise	**Co-encadrant**

Table des matières

Abstract

Graph representation offers a comprehensive and efficient concept to study the relationships between different types of data. It has natural applications in various fields of science engineering, in particular, the management of congestion in urban networks. Indeed, studying graph properties gives valuable information about the state of road traffic. In general, there are various approaches to graph-based clustering. These are distinguished by implicit assumptions on the data represented by the graphs, in particular the notions of densities and similarities. Spectral clustering allows to discover the properties and structure of a graph by studying the spectrum of its associated Laplacian matrix. Depending on the temporal evolution of graphs, spectral clustering can be either static, incremental or dynamic.

After a brief review of the graph concepts and properties used to characterize a simulated urban network, we present a clustering approach using the search of density peaks on the graph. The decision graph is used for the automatic selection of the number of clusters in the simulated urban network. Moreover, the static spectral clustering is presented and applied to the urban network. Then, an incremental approach is presented which is useful when the weights of the edges of the graph evolve over time. Finally, the evolutionary spectral clustering is developed to study the dynamic state of a real urban traffic. Two concepts are applied to respect the temporal regularity, preserving cluster quality and preserving cluster membership.

Keywords : Graph-based clustering ; Spectral clustering ; Density peaks ; Dynamic learning ; Traffic congestion management

Résumé

La représentation par graphes offre un concept polyvalent et efficace pour étudier les relations entre différents types de données. Elle trouve des applications naturelles dans divers domaines des sciences de l'ingénieur, en particulier la gestion de la congestion des réseaux urbains. En effet, l'étude des propriétés des graphes donne des informations précieuses sur l'état du trafic routier. D'une manière générale, il existe diverses approches de classification basées graphes. Celles-ci se distinguent par des hypothèses implicites sur les données représentées par les graphes, en particulier les notions de densités et de similarités. La classification spectrale permet de découvrir les propriétés et la structure d'un graphe à partir de l'étude du spectre de la matrice Laplacienne associée. Selon l'évolution temporelle des graphes, la classification spectrale peut être statique, incrémentale ou dynamique.

Après un rappel de quelques notions et propriétés de graphes pour la caractérisation d'un réseau urbain simulé, nous présentons une approche de classification par recherche de pics de densités s'appuyant sur le graphe ou plan de décision pour la recherche automatique du nombre de classes dans le réseau urbain simulé. La classification spectrale statique est exposée et appliquée sur le réseau urbain et une version incrémentale est présentée, utile lorsque les poids des arêtes du graphe évoluent au cours du temps. Enfin, la classification spectrale évolutive est développée pour étudier l'état dynamique d'un trafic urbain réel. Deux concepts sont appliqués pour respecter la régularité temporelle, préservation de la qualité de la classification et préservation de l'appartenance aux classes.

Mots clés : Classification basée graphes ; Classification spectrale ; Pics de densités ; Apprentissage dynamique ; Gestion de la congestion urbaine.

Remerciements

Je remercie tout particulièrement l'Université du Littoral Côte d'Opale (ULCO), l'Agence Universitaire de la Francophonie (AUF) et le Conseil National de la Recherche Scientifique (CNRS-L) pour avoir soutenu ce travail par une bourse d'études dans le cadre du projet ARCUS E2D2.

Je tiens à remercier tout d'abord mes directeurs de thèse : Professeur Denis HAMAD et Professeur Youssef ZAATAR et mon encadrant de thèse, Docteur Joseph CONSTANTIN. Je leur suis très reconnaissante, pour tout le temps qu'ils ont consacré à la réussite de ce travail. Merci pour votre patience, votre disponibilité et surtout vos conseils judicieux. J'ai beaucoup appris à vos côtés et je vous adresse toute ma gratitude pour tout cela. Ce fut un grand privilège et un honneur de travailler sous leur direction. Je tiens à remercier tout particulièrement Dr. Ibtissam CONSTANTIN et Dr. Clélia LOPEZ pour leurs implications et contributions dans mon travail.

Je tiens également à exprimer mes remerciements particuliers au Professeur Yassine RUICHEK et au Dr Louahdi KHOUDOUR pour le temps précieux et les efforts qu'ils ont consacrés à la révision de mon travail. Je remercie également le Professeur Kifah TOUT, le Professeur Ghaleb FAOUR, et le Dr. Marwa El BOUZ pour avoir examiné ma thèse.

À mes collègues de recherche, je tiens à vous remercier un par un pour tous les moments inoubliables que vous m'avez offerts. Ce fut un grand plaisir de travailler avec une équipe aussi sympathique et dynamique. Je tiens à vous remercier pour votre soutien qui m'a permis de réussir.

Je remercie tout particulièrement tous mes amis qui ont toujours été un réconfort quand j'en avais le plus besoin. Les mots ne pourront jamais exprimer à quel point je suis reconnaissante que vous soyez toujours là pour moi. À mes parents, aucun mot ne peut exprimer mes

sentiments de gratitude. Merci d'être là à chaque instant, de me couvrir d'amour, de soutien, d'encouragement, de prières sincères et d'attention. Je vous suis extrêmement reconnaissante pour tous les sacrifices que vous avez faits pour rendre mon avenir plus brillant. À ma sœur, simplement merci d'être toi. Je suis extrêmement chanceuse d'être ta petite sœur. À mes frères, mes soutiens dans cette vie, un grand merci à vous.
Un dernier mot pour mon partenaire... Merci pour tous tes encouragements, tes soins, ton amour et ton soutien. C'est à toi que je dois tout.

Liste des notations

G	Graphe de données
$\mathcal{V}$	Ensemble des noeuds d'un graphe
E	Ensemble des arrêtes d'un graphe
v_i	Nœud du graphe correspondant au point de données $\boldsymbol{x}_i$
N	Nombre total de points de données
m	Nombre total des arrêtes dans un graphe
$\boldsymbol{A}$	Matrice d'adjacence
$\boldsymbol{W}$	Matrice de similarité
w_{ij}	Poids entre le noeud v_i et v_j
$\boldsymbol{D}$	Matrice de degré diagonale
d_i	Degré du noeud v_i
C_D	Centralité de degré
C_P	Centralité de proximité
C_I	Centralité d'intermediarité
C_{ex}	Centralité d'excentricité
$\aleph$	Réseau urbain
$\mathcal{C}$	Ensemble des croisements du réseau urbain $\aleph$
R	Ensemble des routes du réseau urbain $\aleph$
vit	Vitesse des véhicules
Q	Débit des véhicules
K_c	Concentration des véhicules
K	Nombre de classe
X	Ensemble de données
Δ	Matrice de distance entre les points de données
$\boldsymbol{L}$	Matrice Laplacienne $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$
$\boldsymbol{L}_{sym}$	Matrice Laplacienne normalisé symétrique
$\boldsymbol{L}_a$	Matrice Laplacienne normalisé asymétrique

$\boldsymbol{I}$	Matrice identité
C	Ensemble de K classes
Cut	Coupe du graphe
φ_i	Densité de la route r_i
$\mathcal{D}$	Matrice des distances
$\rho(x_i)$	Densité d'un point x_i
d_c	Distance de coupure
$\delta(x_i)$	Distance séparant x_i du point le plus proche ayant une haute densité
$\rho^{frontmax}$	Vecteur de densité moyenne maximale à la frontière
X_i	Vecteur caractéristiques d'une route
B_i	Vecteur d'adjacence d'une route
α	Paramètre accordé au coût instantané
$TotalCost$	Coût total
L_{pcq}	Matrice Laplacienne évolutive associée au cadre PCQ
L_{pcm}	Matrice Laplacienne évolutive associée au cadre PCQ
ϕ	Coefficient de pondération
T	Séquence des routes
L	Longueur du snake
NC	Coupe normalisée

Liste des abréviations

SUMO	Simulation of Urban MObility
OSM	Open Street Map
DUAROUTER	Dynamic User Assignment
NETEDIT	Network Editor
XML	Extensible Markup Language
DPC	Denisty Peaks Clustering
PCQ	Preserving Cluster Quality
PCM	Preserving Cluster Membership
TC	Temporal Cost
SC	Snapshot Cost
IND	Normalized Spectral Clustering
CCD	Connected Cluster Dissimilarity
NMI	Normalized Mutual Information
DP	Density Peaks
ISC	Incremental Spectral Clustering

Introduction générale

La congestion du trafic routier est un problème majeur qui affecte la vie quotidienne entraînant des retards et des perturbations avec ses impacts négatifs aux niveaux économiques, sociaux et environnementaux. Les problèmes de congestion routière ont augmenté dans les pays à cause de la croissance démographique et les changements dans la densité de la population. Il existe différentes approches dans la littérature qui ont été créées pour décrire le trafic de manière similaire à la dynamique des flux.

En raison de la régularité de la mobilité humaine, la congestion du trafic présente de fortes corrélations dans un contexte spatial et temporel. La propagation du trafic d'un endroit à un autre cause des embouteillages : des voitures provenant de routes sans trafic peuvent s'accumuler à un certain point de la route (Zhang et al., 2012). Par conséquent, le volume et la vitesse des véhicules sur une route spécifique sont affectés par ceux des routes adjacentes. De ce fait, il est primordial de détecter toutes les zones de congestion afin d'organiser au mieux le trafic, en particulier dans les centres-villes. Pour ces derniers, il est préconisé d'identifier les routes de la ville les plus traversées par les automobilistes afin d'adapter les infrastructures du réseau de transport en conséquence, et ainsi optimiser les flux de circulation.

La congestion du trafic dépend de plusieurs facteurs, dont le plus influent est le système de la circulation routière. Ce dernier est très complexe car il est basé sur les changements de circulation, qui peuvent être incertains (Pojani and Stead, 2015). De ce fait, il est difficile de trouver une caractérisation de haute précision en utilisant un modèle de connaissance standard. Par conséquent, les modèles d'apprentissage peuvent être utilisés pour la prédiction du flux de trafic. Ce type de modèle fournira plus de précision en terme d'homogénéité et de connexité que les modèles de connaissance standard.

De nombreuses approches conventionnelles ont été développées pour résoudre le pro-

blème de la congestion du trafic : Farhi et al. (2015) ont travaillé sur une méthode de décentralisation urbaine qui présente une difficulté pour la mise en pratique ; Ognjenović et al. (2015) ont développé une approche basée sur la planification urbaine, qui consiste à prendre en compte le nombre croissant des voitures dans les villes avant de construire l'infrastructure de cette dernière. Pour cela, il faut tenir compte du fait que chaque ville et son système de circulation ont des caractéristiques différentes. Pour cette raison, il est impossible de généraliser une approche de planification pour les villes. En outre, ces méthodes deviennent coûteuses et plus difficiles à maîtriser pour atteindre des objectifs de haut niveau (i.e. à l'échelle du système). Néanmoins, il est nécessaire d'utiliser des formes d'apprentissage alternatif qui prennent en compte les situations de congestion actuelles sur les routes proposées pour naviguer dans les villes. De ce fait, des systèmes automatiques doivent donc être conçus pour détecter la congestion du trafic dans le réseau routier. Des recherches plus récentes qui sont essentiellement basées sur des stratégies de classification automatiques, ont prouvé une amélioration dans la détection de la congestion du trafic dans les villes (Saeedmanesh and Geroliminis, 2017). Ces recherches consistent à appliquer les méthodes de classification afin de partitionner les routes en fonction de la congestion du trafic (Ji and Geroliminis, 2012a), (Saeedmanesh and Geroliminis, 2016), (Lopez et al., 2017). Le partitionnement en groupes homogènes peut être extrêmement utile pour le contrôle du trafic, sachant que la congestion est spatialement corrélée dans les routes adjacentes et qu'elle se propage à des vitesses différentes (Lopez et al., 2017). De plus, Anwar et al. (2017) ont proposé de regrouper un réseau de transport en appliquant la méthode de classification basée sur les pics de densités. Les auteurs ont démontré l'efficacité de cette méthode en partitionnant un réseau de transport en considérant le voisinage entre les segments de routes.

La modélisation de la congestion du trafic dans un cadre automatique peut être approchée par les réseaux déterminés par la théorie des graphes. Cette théorie offre un concept polyvalent et efficace permettant de modéliser facilement les connexions entre différents types de données. Elle est appliquée dans divers domaines notamment les réseaux urbains (Ji and Geroliminis, 2012b), sociaux (White and Smyth, 2005), biologiques, chimiques, etc. Dans le cadre du trafic, la représentation par graphes du réseau urbain offre différentes propriétés pour caractériser le trafic routier. Néanmoins, le choix d'un modèle de réseau approprié pour la représentation du réseau routier est une étape très importante et a un impact sur l'analyse et le traitement effectués ultérieurement. Pour que le modèle soit approprié, la représentation du réseau routier doit être précise en termes de connectivité et de direction des segments de route afin d'appliquer les algorithmes de classifications (Saeedmanesh and Geroliminis, 2016).

Le problème du partitionnement d'un réseau de transport a été étudié dans un cadre statique en considérant les conditions de trafic à un moment donné (Ji and Geroliminis, 2012a; Lopez et al., 2017; Saeedmanesh and Geroliminis, 2016). Or, le trafic routier est considéré comme un système dynamique basé sur les flux du trafic. La prise en compte de cette dynamique représente un avantage significatif pour les acteurs du transport afin d'éviter les zones congestionnées et d'optimiser les itinéraires à réaliser. Par conséquent, cela va induire des changements aussi bien dans la structure du graphe de données que dans les poids des connexions. De plus, les données spatiales à classifier évoluent dans le temps, ce qui va permettre d'avoir un résultat de classification à chaque étape. D'où, l'importance d'étudier la congestion du trafic dans une dimension spatio-temporelle. La classification spectrale basée sur l'utilisation d'une résolution de système propre n'est pas adaptée pour les données qui évoluent dans le temps. En raison de sa complexité temporelle élevée, elle ne convient pas aux données dynamiques.

Dans ce contexte, ce travail de thèse a pour but de caractériser les réseaux de transport dans un contexte spatio-temporel en appliquant des méthodes de classifications évolutives et itératives adaptées aux données dynamiques. Pour faire cela, des algorithmes de calcul en temps réel ont été adaptés pour gérer les changements de similarité entre les points de données. De plus, nous avons développé une méthode qui permet d'obtenir des clusters qui évoluent de manière fluide dans le temps et préservent la connexité entre eux. L'intégration de cette dynamique du trafic dans ces algorithmes de classification évolutive permet d'améliorer les performances de classification à long terme.

Le manuscrit est structuré en cinq chapitres principaux représentant le travail de la thèse. Ci-après, un aperçu général des chapitres du manuscrit :

Dans le premier chapitre, nous présentons les approches de la représentation d'un réseau de transport en un graphe routier. Ces approches reposent sur la théorie des graphes. Dans un premier temps, nous présentons une étude sur les notions des graphes et les notions de centralité d'un graphe qui permettent de mettre en évidence la structure du réseau routier. Ces propriétés sont illustrées par un exemple simulé au moyen du logiciel de simulation microscopique. Ce modèle de simulation permet de reproduire le trafic routier sur un réseau urbain. Le réseau choisi est extrait de la ville de Calais, qui correspond au quartier des Cailloux et Fontinettes. De plus, nous présentons les variables d'analyse qui permettent la pondération des liens du graphe. La représentation des données sous forme de graphe pondéré, conduit à considérer l'application des algorithmes de classification qui sont étudiés dans le cadre de cette thèse.

La classification, utilisée principalement comme méthode d'apprentissage non supervisée, est une technique fondamentale pour l'exploration de données, la segmentation d'images et la classification des réseaux urbains. L'objectif principal de cette technique est de diviser un ensemble de données en groupes ou en classes ayant des caractéristiques communes, tandis que les objets dissemblables appartiennent à des classes différentes. Dans un premier temps, dans le chapitre 2, nous nous intéressons à la méthode de classification basée sur la densité, plus particulièrement, la méthode Density Peaks Clustering (DPC). Cet algorithme qui a été présenté par Rodriguez and Laio (2014) a soulevé un grand intérêt dans la communauté scientifique pour son aspect innovant et son efficacité à classer les données non linéairement séparable (Anwar et al., 2017), (Shi et al., 2017), (Liu et al., 2015). Afin de mettre en évidence les performances de ce modèle à classifier les données d'un réseau routier, nous appliquons cet algorithme sur les données recueillies au niveau de la simulation microscopique. Les routes du réseau urbain sont représentées par des points dans l'espace, ayant chacun 4 caractéristiques : la densité, la vitesse, le taux d'occupation et le temps d'attente moyenne. En plus, l'adjacence des routes est aussi considérée. Ces attributs permettent de réaliser une classification plus précise et donnent des résultats plus cohérents.

Parmi les méthodes de classification, nous distinguons une autre méthode basée sur les graphes, la classification spectrale. Nous présentons dans le chapitre 3 les différents algorithmes de la classification spectrale. Les données recueillies au niveau de la simulation microscopique seront utilisées comme bases de données. Le réseau est représenté, en premier lieu, par un graphe primal où les nœuds du graphe sont les croisements du réseau et les routes sont les arêtes. Le poids entre les nœuds est basé sur la variable de la vitesse. Ensuite, le réseau est représenté par un graphe dual, où les nœuds sont les routes du réseau et les arêtes sont les croisements communs entre deux nœuds. Nous appliquons la classification spectrale sur les deux modèles du graphe afin de mettre en évidence les caractéristiques des méthodes présentées sur les différentes représentations.

Le problème du partitionnement spectral dans les réseaux de trafic urbain a été principalement étudié dans un cadre statique en considérant les conditions de circulation à un moment donné. Néanmoins, il est important de noter que le trafic routier est un processus dynamique ce qui conduit à des changements dans les poids des connexions. Malgré l'importance des algorithmes de classification spectrale, ils ne peuvent pas être adaptés directement pour les problèmes de classification des données dynamiques. En effet, ces algorithmes sont très coûteux. Pour réduire la complexité du temps de calcul, il est nécessaire de développer des algorithmes de calcul incrémentale. Dans ce contexte, nous présentons dans

le chapitre 4 une méthode de classification spectrale incrémentale qui permet d'étudier la congestion du trafic dans la dimension spatio-temporelle. L'idée de base de cette méthode consiste à employer la théorie de perturbation des matrices afin de mettre à jour d'une manière dynamique les valeurs et les vecteurs propres de la matrice du graphe.

Dans le dernier chapitre, nous présentons une approche de classification évolutive pour partitionner un graphe routier issu des données réelles en zones homogènes. Afin d'améliorer les performances des résultats de la classification, la matrice de similarité est calculée de manière à donner plus de poids aux segments de route voisins afin de simplifier la connectivité du graphe. Ensuite, cet algorithme est appliqué pour partitionner un réseau de trafic routier qui varie dans le temps. Deux cadres sont utilisés pour caractériser la classification évolutive : le concept de préservation de la qualité du clustering (Preserving Clustering Quality) et le concept de préservation de l'appartenance (Preserving Clustering Membership). Ces deux cadres sont appliqués aux graphes de données afin de partitionner le réseau routier. Pour optimiser les résultats attendus, il faut que ce dernier soit bien adapté aux données actuelles, tout en considérant l'historique des données les plus récentes.

Chapitre 1

Notions de graphes et application au réseau urbain

Sommaire

1.1 Introduction

Le trafic routier, en particulier le trafic urbain, est au cœur de nombreux problèmes et est devenu un aspect important qui affecte la vie quotidienne. Ce dernier, entraîne des retards et des perturbations avec ses impacts négatifs aux niveaux économiques, sociaux et environnementaux. Il est donc indispensable de connaître les points de congestion afin d'organiser au mieux les routes, en particulier dans les centres-villes. Il faut donc rechercher les points de la ville les plus traversés par les automobilistes afin d'adapter les infrastructures du réseau de transport en conséquence, et ainsi optimiser les flux de circulation.

Le choix d'un modèle approprié pour la représentation d'un réseau routier est une étape très importante et a un impact sur l'analyse et le traitement effectués ultérieurement. La représentation du réseau routier doit être précise en termes de connectivité et de direction des segments de routes afin d'appliquer les algorithmes de la planification d'itinéraire, les algorithmes de classifications, etc. En effet, la représentation par graphes du réseau urbain est naturelle et offre différentes propriétés pour caractériser le trafic routier. La théorie des graphes offre un concept polyvalent et efficace permettant de modéliser facilement les relations entre différents types de données. Elle trouve des applications dans divers domaines notamment les réseaux urbains, sociaux, biologiques, chimiques, etc.

Dans ce chapitre, nous présentons dans la section 1.2, quelques notions des graphes et leurs propriétés en particulier la notion de centralité. Nous montrons dans la section 1.4 les différentes méthodes de représentation d'un réseau urbain en un graphe routier sur un exemple pédagogique. Ensuite, dans la section 1.5 nous présentons le modèle de simulation du trafic utilisé dans notre étude qui permet de reproduire le trafic routier. Enfin, nous expliquons les étapes de la réalisation d'une simulation de trafic routier sur le réseau urbain par un modèle microscopique et nous présentons les différentes variables qui peuvent être adoptées pour la pondération d'un graphe routier. La représentation des données sous forme de graphe pondéré, conduit à considérer l'application des algorithmes de classification qui sont étudiés dans les prochains chapitres.

1.2 Notions de graphes

Un graphe fini $G = (\mathcal{V}, E)$ est une structure mathématique comprenant un ensemble non vide $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ de nœuds ou sommets (vertices), et un ensemble d'arêtes (edges) $E \subset \mathcal{V} \times \mathcal{V}$, $E = \{e_1, e_2, \ldots, e_m\}$. Une arête $e = (v_i, v_j)$ est un arc de connexion reliant les nœuds v_i et v_j qui sont dits adjacents ou voisins. On appelle ordre d'un graphe G le nombre

des nœuds $|\mathcal{V}| = N$ de G, et la taille d'un graphe G correspond au nombre des arêtes de G, $|E| = m$.

Les graphes peuvent être, selon le contexte, orientés ou non-orientés, pondérés ou non-pondérés. Un graphe est dit non orienté si l'ensemble E est composé de couples non ordonnés $e = (v_j, v_i)$. Il est dit, orienté ou dirigé si l'ensemble E est composé de couples de nœuds ordonnés $e = (v_i, v_j)$ avec e est un arc fléché de v_i vers v_j. Un graphe est dit pondéré si à chaque arête $e = (v_i, v_j) \in E$ un poids non nul est affecté. Il existe aussi d'autres types de graphes, comme les graphes simple, complet ou connexe. Un graphe simple est un graphe sans boucle c'est-à-dire sans nœuds rebouclés sur eux-mêmes. Un graphe est complet si chaque nœud du graphe est relié à tous les autres nœuds. Un graphe connexe est un graphe dans lequel chaque paire de nœuds est reliée par une chaîne, c.à.d. s'il est possible, à partir de n'importe quel nœud, de rejoindre tous les autres en suivant les arêtes. La figure 1.1 présente un exemple pédagogique d'un réseau routier représenté par un graphe $G = (\mathcal{V}, E)$ connexe contenant $\mathcal{V} = 12$ nœuds libellés de A à L et reliés par un total de $|E| = 24$ arêtes.

Un graphe non orienté et non pondéré $G = (\mathcal{V}, E)$ composé de $|\mathcal{V}| = N$ nœuds, peut être

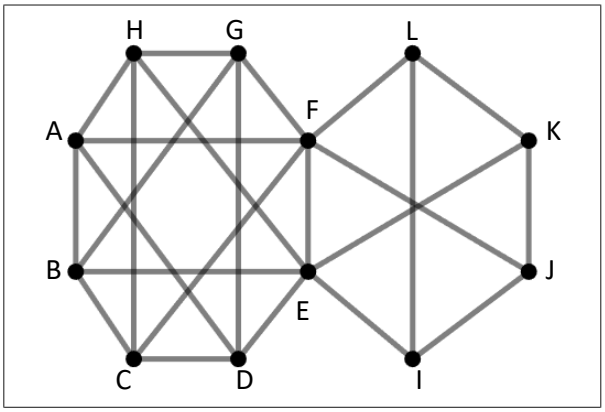


FIGURE 1.1 – Exemple d'un graphe $G = (\mathcal{V}, E)$ d'ordre $\mathcal{V} = 12$ nœuds et de taille $E = 24$ arêtes.

représenté par une matrice carré symétrique binaire appelée matrice d'adjacence, notée par $\boldsymbol{A}$. La matrice d'adjacence permet de définir les arêtes entre chaque sommet du graphe. La matrice d'adjacence peut être vue comme un tableau dont les lignes sont les sommets de départ, et les colonnes sont les sommets d'arrivée de chaque arête. Elle est définie par :

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{pmatrix} \tag{1.1}$$

où,

$$\begin{cases} a(i,j) = 1, & \text{si } v_i \text{ est adjacent à } v_j \\ a(i,j) = 0, & \text{sinon.} \end{cases}$$

Pour un graphe non orienté et pondéré, la matrice d'adjacence est pondérée, c'est-à-dire, pour chaque arête reliant deux nœuds, un poids non nul est associé. La matrice d'adjacence pondérée est aussi appelée matrice de similarité. La matrice de similarité notée par $\boldsymbol{W}$ associée au graphe G est une matrice carrée de dimension $N \times N$ symétrique de poids $(w_{ij} = w_{ji})$. On considère que le graphe est simple et sans boucle $(\boldsymbol{W}(i,i) = w_{ii} = 0)$. La matrice est donnée par :

$$\boldsymbol{W} = \begin{pmatrix} 0 & w_{12} & \dots & w_{1N} \\ w_{21} & 0 & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & 0 \end{pmatrix} \tag{1.2}$$

avec, w_{ij} est le poids de l'arête (v_i, v_j). On peut pondérer le graphe de la figure 1.1 en associant un poids w_{ij} à chaque arête reliant deux nœuds. La matrice de similarité $\boldsymbol{W}$ associée au graphe G de la figure 1.1 est définie comme suit :

$$\boldsymbol{W} = \begin{pmatrix} 0 & 0.07 & 0 & 0.07 & 0 & 0.90 & 0 & 0.02 & 0 & 0 & 0 & 0 \\ 0.07 & 0 & 0.65 & 0 & 0.38 & 0 & 0.90 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.65 & 0 & 0.65 & 0 & 0.18 & 0 & 0.38 & 0 & 0 & 0 & 0 \\ 0.07 & 0 & 0.65 & 0 & 0.38 & 0 & 0.90 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.38 & 0 & 0.38 & 0 & 0.38 & 0 & 0.18 & 0.90 & 0 & 0.99 & 0 \\ 0.90 & 0 & 0.18 & 0 & 0.38 & 0 & 0.07 & 0 & 0 & 0.07 & 0 & 0.90 \\ 0 & 0.90 & 0 & 0.90 & 0 & 0.07 & 0 & 0.65 & 0 & 0 & 0 & 0 \\ 0.02 & 0 & 0.38 & 0 & 0.18 & 0 & 0.65 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.90 & 0 & 0 & 0 & 0 & 0.90 & 0 & 0.38 \\ 0 & 0 & 0 & 0 & 0 & 0.07 & 0 & 0 & 0.90 & 0 & 0.65 & 0 \\ 0 & 0 & 0 & 0 & 0.99 & 0 & 0 & 0 & 0 & 0.65 & 0 & 0.65 \\ 0 & 0 & 0 & 0 & 0 & 0.90 & 0 & 0 & 0.38 & 0 & 0.65 & 0 \end{pmatrix} \tag{1.3}$$

Notons qu'une matrice de similarité peut toujours se convertir en une matrice binaire en substituant $w_{ij} = 1$ pour chacune des variables représentant les lignes de $\boldsymbol{W}$.

1.2.1 Attributs topologiques de graphes

Pour un graphe G non pondéré, on définit le degré d'un nœud $v_i \in \mathcal{V}$ par le nombre d'arêtes incidentes à ce nœud. Alors, que pour un graphe pondéré G, le degré d'un nœud v_i

est défini par la somme des éléments de la ligne de la matrice de similarités $\boldsymbol{W}$ correspondant au noeud. On note le degré d'un nœud v_i par d_i, et il est défini par (Zaki et al., 2014) :

$$d_i = \sum_{j=1}^{N} w_{ij} \tag{1.4}$$

Dans l'exemple donné par la figure 1.1 et la matrice de similarité $\boldsymbol{W}$ associée (Eq. 1.3), le degré du nœud A est égal à la somme des poids des arêtes reliant le nœud A à ses 4 nœuds adjacents, c'est à dire $d_A = 1.06$.

Dans le cas où le graphe G est orienté, nous distinguons deux types de degrés pour un nœud $v_i \in \mathcal{V}$: le degré sortant noté par $d^+(v_i)$ et le degré entrant noté par $d^-(v_i)$. Le degré sortant est calculé par rapport aux arêtes sortants du v_i et il est donné par :

$$d^+(v_i) = \sum_{j=1}^{N} w_{ij} \tag{1.5}$$

Alors que le degré entrant est calculé par rapport aux arêtes entrantes vers v_i, il est donné par :

$$d^-(v_i) = \sum_{j=1}^{N} w_{ji} \tag{1.6}$$

Le degré d'un graphe est représenté par une matrice carré, appelée matrice de degré $\boldsymbol{D}$ de dimensions $|\mathcal{V}| \times |\mathcal{V}|$ telle que les éléments sur la diagonale sont les valeurs des degrés des nœuds d_i et les éléments hors diagonale sont nuls, elle est représentée par :

$$\boldsymbol{D} = diag\{d_1, d_2, \ldots, d_{N-1}, d_N\}. \tag{1.7}$$

La matrice de degré $\boldsymbol{D}$ associée au graphe de l'exemple donné par la figure 1.1 est la suivante :

$$\boldsymbol{D} = diag\{1.06, 2, 1.86, 2, 3.21, 2.50, 2.52, 1.23, 2.18, 1.62, 2.29, 1.93\}. \tag{1.8}$$

Dans les graphes connexes, il est toujours possible de définir la distance entre deux nœuds d'un graphe par la longueur du plus court chemin entre ces deux nœuds, notée par $d(v_i, v_j)$. Pour un graphe non pondéré, la longueur d'un chemin est égale au nombre des arêtes qui le composent alors que, pour un graphe pondéré la longueur d'un chemin est égale à la somme des poids des arêtes qui le composent. L'excentricité notée par $ex(v_i)$ d'un nœud v_i est la distance maximale de ce nœud à tous les autres nœuds dans le graphe, l'excentricité est définie par :

$$ex(v_i) = max_j \left\{ d(v_i, v_j) \right\} \tag{1.9}$$

Le tableau 1.1 montre les résultats de l'excentricité de chaque noeud dans le graphe de notre exemple pédagogique.

TABLEAU 1.1 – Tableau des résultats de calcul de l'excentricité de chaque noeud dans le graphe.

	A	B	C	D	E	F	G	H	I	J	K	L
Excentricité	1.48	1.55	1.15	1.55	1.28	0.97	1.04	1.46	1.17	0.97	1.17	1.55

Le rayon d'un graphe G noté $rad(G)$ est l'excentricité minimale de tout nœud dans le graphe G :

$$\begin{aligned} rad(G) &= min_i \{ex(v_i)\} \\ &= min_i\{max_j \{d(v_i,v_j)\}\} \end{aligned} \tag{1.10}$$

Le diamètre d'un graphe G noté $diam(G)$ est l'excentricité maximale de ses nœuds, il est donné par :

$$\begin{aligned} diam(G) &= max_i \{ex(v_i)\} \\ &= max_i\{max_j \{d(v_i,v_j)\}\} \end{aligned} \tag{1.11}$$

La valeur du rayon dans notre exemple, représenté dans la figure 1.1, est donc égale à 0.97 et la valeur de son diamètre est égale à 1.55.
Dans la suite, nous présentons quelques mesures de centralité proposées dans le cadre de l'analyse des graphes. Celles-ci incluent les centralités de degré, de proximité, d'intermédiarité ainsi que la centralité d'excentricité.

1.3 Mesures de centralités

Dans l'analyse des réseaux sociaux, certains acteurs jouent un rôle très important dans la diffusion des informations, alors que d'autres acteurs ont moins d'effets sur le réseau. Ces acteurs occupent en fait une position avantageuse dans les réseaux sociaux (Wasserman et al., 1994). Ce phénomène n'est pas propre aux réseaux sociaux, comme on le retrouve dans de nombreux autres types de graphes. Par exemple, dans un réseau de communication, un réseau de transport (Crucitti et al., 2006), etc. Afin de quantifier le concept de l'importance des nœuds dans le graphe, les chercheurs ont proposé plusieurs définitions dans la littérature connues sous le nom de mesures de centralité (Koschützki et al., 2005). La centralité est une mesure décrivant l'importance ou la pertinence des nœuds dans un graphe (Chikhi, 2010). Dans la suite, nous décrivons quelques mesures de centralité des graphes. Notons qu'un graphe peut être un graphe pondéré ou non pondéré.

— **Centralité de degré** : La centralité de degré (Freeman, 1978) d'un nœud $v_i \in \mathcal{V}$ est son degré d_i (i.e. la somme de ses arêtes). Cette mesure est basée sur l'idée que l'importance d'un nœud dans un graphe dépend du nombre total de nœuds avec lesquels il interagit directement. Ainsi, plus le nœud a un degré élevé, plus sa centralité est importante dans le graphe. La centralité de degré normalisée est définie par :

$$C_D(v_i) = \frac{d_i}{|\mathcal{V}| - 1} = \frac{d_i}{N - 1} \tag{1.12}$$

— **Centralité de proximité** : La centralité de proximité (Freeman, 1978) d'un nœud $v_i \in \mathcal{V}$, notée $C_p(v_i)$, indique si le nœud v_i du graphe est proche des autres nœuds de ce graphe et s'il peut rapidement interagir avec ces nœuds. Cette mesure de centralité est calculée par l'inverse de la somme des distances de v_i aux autres nœuds d'un graphe :

$$C_P(v_i) = \frac{1}{\sum_j d(v_i, v_j)} \tag{1.13}$$

avec, $v_j \in \mathcal{V}$ et $d(v_i, v_j)$ est la distance entre les deux noeuds v_i et v_j. Cette distance est définie par la longueur du chemin le plus court entre les deux noeuds v_i et v_j.

— **Centralité d'intermediarité** : La centralité d'intermédiarité (Freeman, 1978) d'un nœud $v_i \in \mathcal{V}$ est mesurée par le nombre des plus courts chemins entre toutes les paires de nœuds y compris v_i. Cette centralité mesure l'importance de traverser le nœud v_i pour aller d'un nœud v_j à un autre nœud v_p. La centralité d'intermédiarité est définie par :

$$C_I(v_i) = \sum_{j \neq i} \sum_{p \neq i, p > j} \frac{\eta_{j,p}(v_i)}{\eta_{j,p}} \tag{1.14}$$

avec, $v_j, v_p \in \mathcal{V}$ et $\eta_{j,p}$ est le nombre de chemins les plus courts entre les sommets v_j et v_p et $\eta_{j,p}(v_i)$ est le nombre total de chemins entre les nœuds v_j et v_p qui passent par le nœud v_i. Notons, la longueur du plus court chemin entre deux nœuds (dite distance géodésique) correspond au nombre minimum d'arêtes qu'il faut traverser pour relier les deux nœuds ; pour un graphe pondéré, c'est la somme minimale des poids des arêtes pour joindre les deux nœuds.

— **Centralité d'excentricité** : La centralité d'excentricité d'un nœud $v_i \in \mathcal{V}$ est l'inverse de l'excentricité, plus l'excentricité est importante, moins le nœud est central. Cette mesure est définie par :

$$C_{ex}(v_i) = \frac{1}{ex(v_i)} = \frac{1}{max_j\{d(v_i, v_j)\}} \tag{1.15}$$

Pour illustrer les différentes mesures de centralité, nous appliquons ces mesures sur l'exemple routier pédagogique de la figure 1.1. La figure 1.2 présente un exemple d'analyse de centralité du graphe : la centralité de degré (1.2a), la centralité de proximité (1.2b), la centralité d'intermédiarité (1.2c) et la centralité d'excentricité (1.2d). Dans chaque graphe, les nœuds sont colorés du ton clair au ton foncé, les tons clairs indiquent une faible valeur de centralité, alors qu'un ton plus foncé indique une valeur plus élevée. Le tableau 1.3 présente les résultats de calcul des centralités de chaque nœud dans le graphe. D'après les résultats représentés dans la figure 1.2 et dans le tableau 1.2, on constate que les résultats fournis par l'analyse de la centralité de degré montrent que le nœud E possède la plus forte centralité par rapport aux autres nœuds du graphe, et les deux noeuds F et G jouent aussi un rôle important dans le graphe. Cette mesure est basée sur l'idée que l'importance d'un nœud dans un graphe dépend du nombre total des noeuds adjacents avec lesquels il interagit directement. Nous remarquons que les nœuds E et F sont les plus importants dans le graphe du point de vue proximité. Cela est dû au fait que les noeuds E et F possèdent la plus forte centralité par rapport aux autres noeuds. Pour la centralité d'intermédiarité le nœud H est le plus important dans le graphe, ce nœud est en effet le plus traversé par les chemins entre les nœuds du graphe. On peut aussi remarquer que les nœuds B, D et L sont les nœuds les plus excentriques dans le graphe.

TABLEAU 1.2 – Tableau des résultats des mesures de la centralité du graphe

	A	B	C	D	E	F	G	H	I	J	K	L
C_D	1.06	2	1.86	2	**3.12**	2.50	2.52	1.23	2.18	1.62	2.20	1.93
C_P	0.15	0.14	0.16	0.14	**0.17**	**0.17**	0.15	0.16	0.09	0.16	0.09	0.08
C_I	19	0	6.08	0	17.91	22.91	0	**24.5**	2.58	6	0	1
C_{ex}	0.67	0.64	0.86	0.64	0.78	**1.03**	0.96	0.68	0.85	**1.03**	0.85	0.64

1.3.1 Mesures de centralité de différents types de graphes

Lors de l'étude du concept de centralité, il est important de faire la distinction entre les graphes orientés et non orientés. Dans un graphe orienté, les nœuds ont deux types d'arêtes, à savoir les arêtes entrantes et les arêtes sortantes. Pour une définition donnée du concept de centralité, chaque nœud aura deux métriques importantes : des métriques sur ses arêtes sortantes, appelées métriques de centralité d'influence ou de centralité sortante et une autre métriques relative à ses arêtes entrantes, appelée mesure de prestige ou de centralité entrante. Alors que, dans un graphe non orienté, chaque nœud a un seul type d'arête, ensuite, chaque nœud a une mesure d'importance, appelée mesure de centralité. Les méthodes de calcul des mesures de la centralité se diffèrent suivant les types de graphes (Ahmadzai et al.,

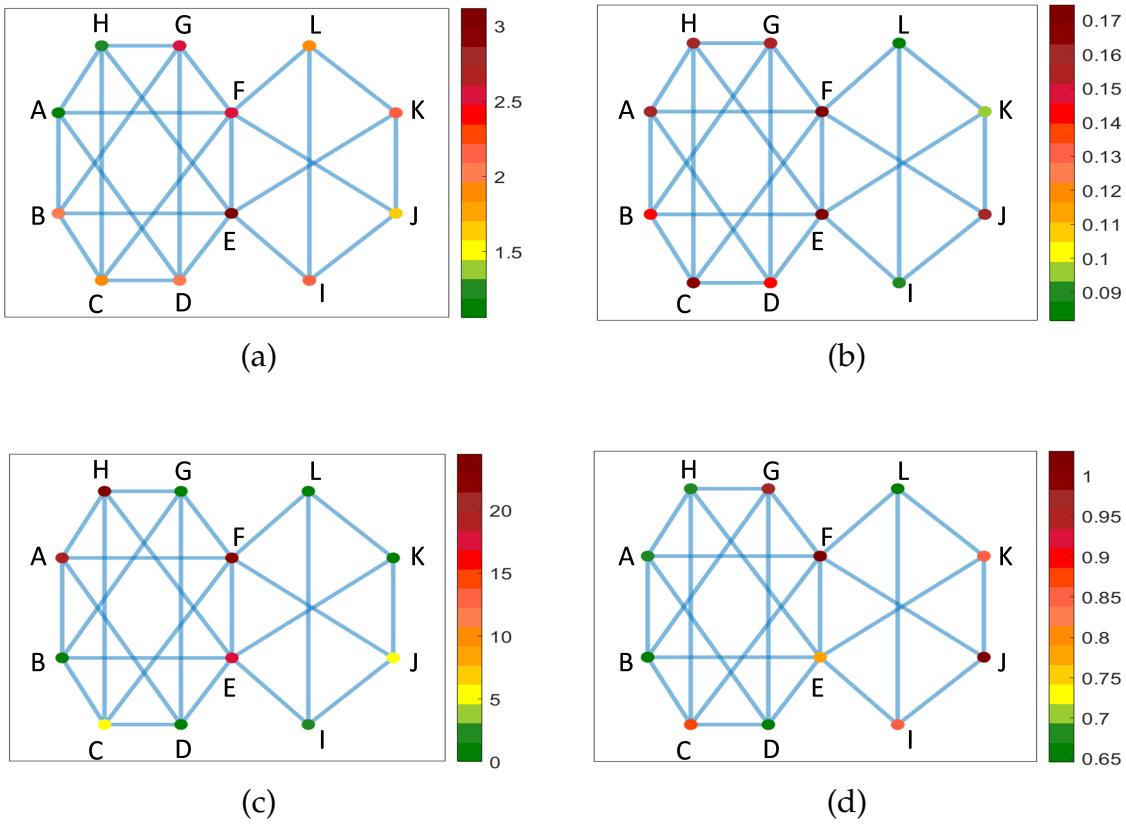


FIGURE 1.2 – Mesures de centralité d'un graphe : (a) centralité de degré, (b) centralité de proximité, (c) centralité d'intermédiarité et (d) centralité d'excentricité.

2019; Opsahl et al., 2010). Le tableau 1.3, introduit les différentes méthodes de calcul des mesures de centralité correspondante aux différents types de graphes G : "non pondéré, non orienté", "pondéré, non orienté", "non pondéré, orienté", "pondéré, orienté".

1.4 Représentation d'un réseau urbain par graphe

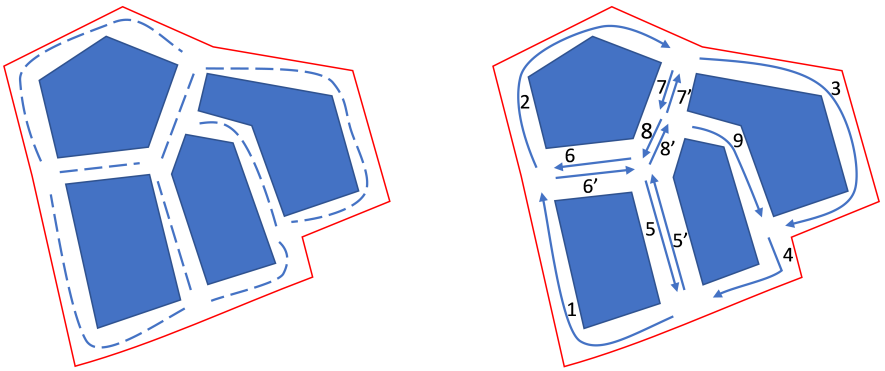
Un réseau routier urbain est composé d'un ensemble de routes représentés afin de permettre aux usagers de se déplacer d'un endroit à un autre. Les croisements entre deux routes ou plus sont appelés intersections. Au niveau des intersections, des règles de circulation comme les panneaux de signalisation et les feux de circulation s'appliquent. Le réseau routier urbain peut donc être défini comme étant un ensemble d'intersections, liées les unes aux autres par des routes, auxquelles des règles opérationnelles peuvent être introduites (Gundaliya et al., 2008; Krajzewicz et al., 2002). Pour en faire un réseau routier interprétable, nous

TABLEAU 1.3 – Tableau des mesures de centralité de différents types de graphes

	Centralité de degré	**Centralité de proximité**	**Centralité d'intermédiarité**
G **non pondéré, non orienté**	$\frac{1}{N-1}\sum_{j=1}^{N} a_{ij}$	$\frac{N-1}{\sum_{j=1,j\neq i}^{N} d(v_i,v_j)}$	$\frac{2}{(N-1)(N-2)}\sum_{j\neq i}\sum_{p\neq i,p>j}\frac{\eta_{j,p}(v_i)}{\eta_{j,p}}$
G **pondéré, non orienté**	$\frac{1}{N-1}\sum_{j=1}^{N} w_{ij}$	$\frac{\sum_{j=1,j\neq i}^{N}[w_{ij}-1]}{\sum_{j=1,j\neq i}^{N}[d(v_i,v_j)\times w_{ij}]}$	$\frac{2}{(\sum_{j=1,j\neq i}^{N} w_{ij}-1)(\sum_{j=1,j\neq i}^{N} w_{ij}-2)}\sum_{j\neq i}\sum_{p\neq i,p>j}\left[\frac{\eta_{j,p}(v_i)}{\eta_{j,p}}\times w_{ij}\right]$
G **non pondéré, orienté**	$C_D^{out}(v_i) = \frac{1}{N-1}\sum_{j=1}^{N} a_{ij}$ $C_D^{in}(v_i) = \frac{1}{N-1}\sum_{j=1}^{N} a_{ji}$	$C_p^{out}(v_i) = \frac{N-1}{\sum_{j=1,j\neq i}^{N} d(v_i,v_j)}$ $C_p^{in}(v_i) = \frac{N-1}{\sum_{j=1,j\neq i}^{N} d(v_j,v_i)}$	$\frac{1}{(N-1)(N-2)}\sum_{j\neq i}\sum_{p\neq i,p>j}\frac{\eta_{j,p}(v_i)}{\eta_{j,p}}$
G **pondéré, orienté**	$C_D^{out}(v_i) = \frac{1}{N-1}\sum_{j=1}^{N} w_{ij}$ $C_D^{in}(v_i) = \frac{1}{N-1}\sum_{j=1}^{N} w_{ji}$	$C_p^{out}(v_i) = \frac{\sum_{j=1,j\neq i}^{N}[w_{ij}-1]}{\sum_{j=1,j\neq i}^{N}[d(v_i,v_j)\times w_{ij}]}$ $C_p^{in}(v_i) = \frac{\sum_{j=1,j\neq i}^{N}[w_{ji}-1]}{\sum_{j=1,j\neq i}^{N}[d(v_j,v_i)\times w_{ji}]}$	$\frac{1}{(\sum_{j=1,j\neq i}^{N} w_{ij}-1)(\sum_{j=1,j\neq i}^{N} w_{ij}-2)}\sum_{j\neq i}\sum_{p\neq i,p>j}\left[\frac{\eta_{j,p}(v_i)}{\eta_{j,p}}\times w_{ij}\right]$

devons lui donner une représentation mathématique sous la forme d'un graphe.

Dans un réseau urbain, les routes peuvent être unidirectionnelles sur certaines voies tandis que bidirectionnelles sur d'autres, c'est-à-dire, le graphe représentant le réseau est un graphe orienté (figure 1.3b).



(a) (b)

FIGURE 1.3 – Les 2 types de réseaux (a) sans sens de routes et (b) avec sens de routes.

Dans la suite, nous présentons sur un exemple pédagogique, la construction de graphes associés à un réseau routier, avec ou sans sens de circulation. Ensuite, on utilise la simulation microscopique du trafic Simulation of Urban MObility (SUMO) pour représenter une carte graphique en réseau routier et construire le graphe des routes. Après la construction du graphe, on s'intéresse aux études de mesures de centralité pour analyser l'interaction entre les nœuds du graphe et appuyer sur l'importance de chaque route dans le réseau.

1.4.1 Graphe d'un réseau urbain non orienté

Considérons un réseau routier défini par $\aleph = (\mathcal{C}, R)$ où $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$ est l'ensemble des croisements (ou intersections) reliés par $R = \{r_1, r_2, \ldots, r_m\}$, l'ensemble des routes (ou lien du réseau) non orientées. Afin de construire le graphe routier $G = (\mathcal{V}, E)$ correspondant au réseau $\aleph$, nous distinguons deux représentations possibles d'un réseau de transport : en graphe primal ou en graphe dual.

La première approche, le graphe primal, consiste intuitivement à représenter les croisements par des nœuds et les routes par des arêtes. Le graphe routier G est construit en considérant chaque croisement $c_i \in \mathcal{C}$ dans le réseau comme étant un nœud v_i, et en ajoutant une arête non orienté e_i entre chaque paire de noeuds (v_i, v_j) s'il existe au moins une route $r_i \in R$ reliant les deux croisements (c_i, c_j) correspondants entre eux. Dans ce type, la

topologie du réseau est prise en compte, c'est-à-dire, on conserve la même forme du réseau, les arêtes dans le graphe G suivent les chemins de vraies routes du réseau routier $\aleph$. Cette représentation est surtout adaptée pour l'analyse de la centralité du graphe (Crucitti et al., 2006), ou aux applications sur le contrôle des feux de signalisation rattaché aux intersections des routes afin d'attribuer le droit de passage aux flux des véhicules (Ma et al., 2009). La figure 1.4 présente un exemple de la représentation d'un réseau routier en un graphe primal. La première étape consiste à identifier les intersections dans le réseau, ensuite, il faut identifier les routes reliant chaque deux intersections, et finalement, construire le graphe G où les intersections sont représentées par des noeuds et les routes par des arêtes.

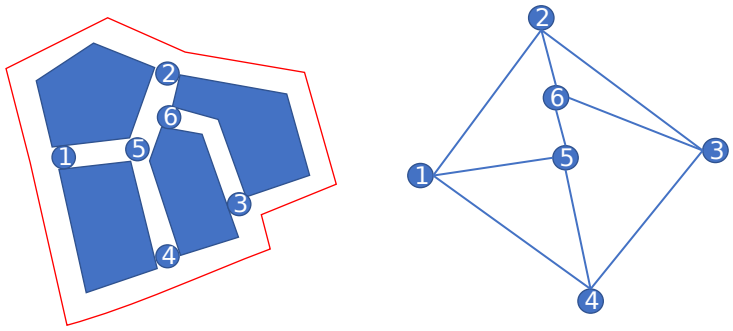


FIGURE 1.4 – Les étapes de construction d'un graphe routier primal.

La seconde approche, le graphe dual, consiste à considérer les routes comme les nœuds de graphe et les intersections comme les arêtes. Le graphe G est construit en ajoutant chaque segment de route (ou lien) $r_i \in R$ dans le réseau comme étant un nœud v_i, et en établissant une arête non orientée e_i entre chaque paire de nœuds possible (v_i, v_j) s'il existe au moins un croisement $c_i \in \mathcal{C}$ qui est un croisement commun entre les routes r_i et r_j. La figure 1.5 illustre un exemple de la représentation d'un réseau de transport en un graphe dual. La première étape consiste à identifier les routes dans un réseau, à noter qu'une route est considérée comme un seul nœud quelle que soit sa longueur réelle. Ensuite, on construit le réseau routier en reliant les routes où il existe au moins une intersection entre elles. La dernière étape consiste à construire le graphe G où les segments de la route sont représentées par des nœuds et les intersections reliant les routes par des arêtes.

1.4.2 Graphe d'un réseau urbain orienté

La plupart des routes urbaines existent en tant que routes à double sens, en effet, ces routes peuvent être considérées comme deux segments à sens unique dirigés de façons opposées et séparés l'un de l'autre. Chacune des deux directions subit différents types de mo-

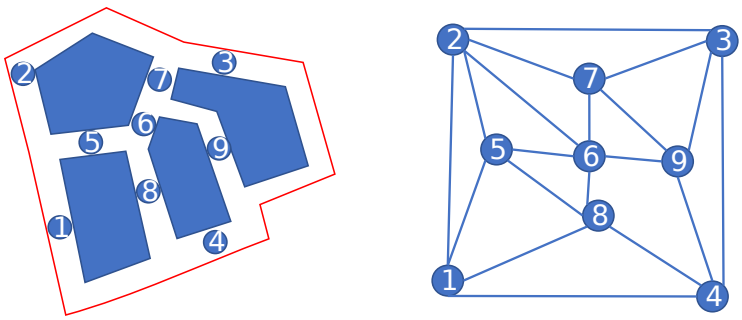
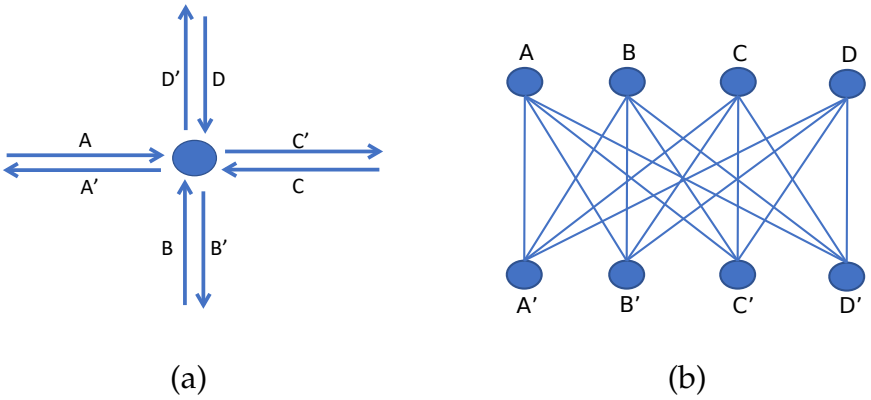


FIGURE 1.5 – Les étapes de construction d'un graphe routier dual

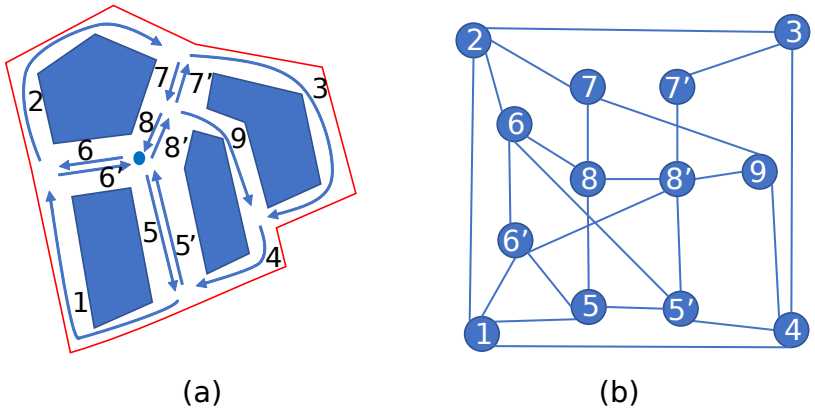
dèles de flux de trafic. Par exemple, à une période donnée de la journée, une route qui relie la périphérie au centre-ville trouverait plus ou moins de trafic en direction du centre-ville que dans la direction opposée. Pour préserver cette caractéristique des réseaux urbains il faut considérer les deux directions de trafic comme des segments de route séparés. Le graphe routier G est donc construit en représentant chaque segment de route $r_i \in R$ du réseau routier $\aleph$ par un nœud v_i du graphe, et en établissant une arête non orientée e_i entre chaque paire de nœuds possible (v_i, v_j) s'il existe au moins un point de croisement $c_i \in \mathcal{C}$ commun entre les routes de r_i vers r_j et que le trafic peut circuler de r_i vers r_j ou vice versa (Anwar et al., 2017). Dans un réseau routier, il existe des points d'intersection circulaires appelés ronds-points composés d'un ensemble de quatre routes, où chaque route peut être décomposée en une direction entrante et une direction sortante (exemple, figure 1.6-a), dans laquelle le trafic routier entre par une direction et ressort par une autre. Un "rond-point" permet de faire des demi-tours, ce qui est souvent impossible à d'autres formes d'intersections. La figure 1.6 explique la méthode de représentation d'un rond-point ayant des segments de routes orientés entrant et sortant de ce point commun en un graphe. Un segment de route (A par exemple) entrant vers une intersection est connecté avec tous les autres segments sortant de ce même point (B',C' et D'), en plus, il est connecté à son segment opposé de l'autre sens (A').

La figure 1.7 introduit un exemple de la représentation d'un réseau routier orienté en un graphe routier. La figure 1.7-a illustre le réseau routier avec des intersections et des routes orientées bien définies. Dans le réseau routier de la figure 1.7-b, on peut voir que chaque segment de routes est représenté sous forme d'un nœud et les arêtes du graphe sont représentées par les croisements entre les pairs des routes.



(a) (b)

FIGURE 1.6 – Représentation d'un rond-point par graphe.



(a) (b)

FIGURE 1.7 – Représentation d'un réseau de transport avec sens de routes en un graphe routier

1.5 Modèles microscopiques du trafic routier

Les modèles de simulation microscopiques permettent de décrire le comportement individuel de chaque véhicule sur le réseau routier. Ces modèles permettent de réaliser des simulations numériques réalistes. Les interactions entre les véhicules sont déterminées par la façon dont les véhicules influencent le mouvement les uns des autres. Chaque véhicule est simulé individuellement, défini au moins par un identifiant (nom), l'heure de départ, l'itinéraire du véhicule sur le réseau routier, la vitesse et la position peuvent aussi être définies (Behrisch et al., 2011; Krajzewicz, 2010). Le choix du simulateur de mobilité dépend du type d'études à effectuer. Dans le cas d'études de communication véhiculaire, la mobilité

doit être couplée à un simulateur de réseau routier (Codeca et al., 2015) tel que le simulateur microscopique appelé Simulation of Urban MObility (SUMO) qui permet d'accéder à une simulation de trafic routier en cours, de récupérer les valeurs des objets simulés et de gérer leur comportement. SUMO est aussi capable de simuler un trafic multimodal, des feux de circulation, des boucles inductives, etc, donc, le simulateur microscopique Simulation of Urban MObility convient à nos cas d'étude. SUMO est un cadriciel de simulation de trafic basé sur un modèle d'évolution temporelle discrète. Il utilise un modèle microscopique de flux de trafic. SUMO est conçu comme une plateforme d'évaluation générale pour différents modèles et algorithmes liés au trafic ou à sa gestion. Notons quelques fonctionnalités de cet outil :

— La simulation de trafic est basée sur un modèle d'évolution en temps discret, mais les véhicules se déplacent de manière continue.
— L'exécution est relativement rapide : des données jusqu'à 100000 véhicules peuvent être traitées chaque seconde à l'aide d'un processeur à 1 GHz.
— La simulation peut prendre en charge les rues à plusieurs voies ainsi que le déplacement des véhicules entre les voies.
— Des feux de circulation peuvent également être introduits dans le réseau afin de gérer les mouvements des véhicules aux intersections ainsi que certaines autres règles de priorité spécifiées.
— La simulation peut être effectuée en ligne de commande ou à l'aide d'une interface utilisateur graphique.
— Les réseaux routiers peuvent être créés manuellement ou importés à partir d'OpenStreetMap.

Dans le but d'avoir une simulation fonctionnelle et réaliste, une simulation doit prendre en charge plusieurs propriétés afin de (Codeca et al., 2015) :

— Supporter les différents types de trafic routier, tels que les circulations dense ou fluide.
— Supporter des scénarios de différents tailles (un scénario à petite échelle ou à grande échelle).
— Inclure différents types de routes, par exemple les autoroutes, et les routes artérielles.
— Permettre des évaluations multimodales du trafic, par exemple, véhicules de particuliers, transports publics et piétons.
— Décrire un scénario de trafic réaliste, c'est-à-dire éviter les blocages et les schémas de mobilité irréalistes. Elle doit prendre en compte la congestion du trafic durant les heures de pointes (forte densité), durant la journée (densité modérée) et durant la nuit (faible densité).

Nous avons choisi d'utiliser le réseau routier de la ville de Calais comme base de notre scénario afin de reproduire une simulation de trafic proche du réel et les schémas de mobilité. Le réseau routier choisi est une partie de la ville correspondant aux quartiers des Cailloux et des Fontinettes. Cette zone a été choisie parce qu'elle constitue un échantillon représentatif de la ville de Calais : elle comprend des routes de différentes catégories, des routes locales, des routes principales, des routes rejoignant l'autoroute, elle comprend à la fois des zones résidentielles et des centres d'activités diverses. Nous avons utilisé OpenStreetMap (OSM), un service de cartographie en ligne Open Source, qui permet d'exporter facilement des cartes au format *.osm*. Les données sont donc enregistrées sous un fichier qui contient toutes les informations nécessaires sur les intersections, les types de routes, le sens de circulation, les pistes cyclables, les passerelles, etc. La figure 1.8, montre la partie de la carte utilisée dans notre étude.

FIGURE 1.8 – Exportation de la carte avec Open Street Map

La création d'une simulation sur un réseau routier consiste à générer un réseau du type SUMO, le réseau peut être créé ou importé, dans notre cas, le réseau est basé sur la carte graphique que nous avons extraite d'OSM. Ceci peut être réalisé à partir de la commande NETCONVERT mise à disposition par SUMO (Krajzewicz et al., 2012). NETCONVERT est un outil utilisé en ligne de commande, pour importer des réseaux routiers à partir de différentes sources de données, par exemple, OpenStreetMap (OSM), OpenDRIVE, Shapefile ou à partir d'autres simulateurs tels que MATSim et Vissim (Krajzewicz et al., 2012). Notons que, parfois les informations importantes pour réaliser une simulation microscopique correcte sont souvent manquantes, telles que, la voie qui doit être utilisée par un véhicule pour se rendre sur une route, les règles de passage au niveau des intersections, et même des

informations sur les positions des feux de circulation. Dans ce cas, NETCONVERT introduit une heuristique pour calculer les valeurs manquantes, c'est-à-dire les plans de gestion des feux de circulation, les règles de priorité et les géométries d'intersection pour les réseaux OSM.
En plus, NETCONVERT est capable d'exporter des réseaux routiers au format xml appelés "XML plain". Cette représentation XML est divisée en cinq types de fichiers, contenant des informations sur : (1) les intersections, (2) les routes, (3) les types des routes, (4) les connexions et (5) les plans de feux de circulation. Ces fichiers permettent le plus haut degré de contrôle pour décrire un réseau routier par SUMO. Dans notre cas, les fichiers XML ont été utilisés pour obtenir les coordonnées des intersections ainsi que les routes reliant ces intersections, afin de pouvoir tracer le graphe routier en respectant la topologie. Nous avons analysé ces fichiers dans Matlab en utilisant des outils d'analyse XML pour les convertir en tableaux de structure et accéder aux attributs nécessaires pour la construction du graphe.

Dans le but d'avoir une simulation fonctionnelle, toutes les intersections ont été vérifiées manuellement en utilisant un processus itératif avec NETEDIT, NETCONVERT et SUMO pour améliorer la qualité du réseau. NETEDIT est un éditeur de réseau graphique qui peut être utilisé pour créer, analyser et modifier les fichiers du réseau. Cet éditeur sert à compléter l'heuristique de génération de réseau avec des raffinements manuels.
Certaines structures d'intersections sont souvent représentées par une suite de plusieurs intersections adjacentes très rapprochées les unes des autres, et aussi de fausses intersections peuvent être générées lorsque, par exemple, la largeur d'une route est augmentée ou réduite d'une voie. Afin de ne pas surcharger le réseau, nous avons effectué quelques traitements sur le réseau routier en utilisant NETCONVERT en premier lieu, pour supprimer les éléments qui ne sont pas utiles à notre cas d'étude, par exemple, les routes isolées, les voies de chemin de fer, les pistes cyclables et les rues piétonnes, etc. Ensuite, nous avons utilisé l'éditeur NETEDIT pour supprimer les fausses intersections et les routes inutiles pour la simulation, par exemple les routes qui ne sont pas accessibles par des véhicules motorisés et certaines routes des quartiers résidentiels. L'utilisation de ce processus itératif est nécessaire pour construire un réseau routier en respectant la géométrie des intersections et des segments appropriés requis par SUMO et pour éviter les goulots d'étranglement. Ces changements ont entraîné des différences mineures dans la topologie du réseau, c'est-à-dire que, l'angle entre deux routes peut être différent et cela apporte un changement léger dans la forme des routes.

1.6 Simulation du trafic sur le réseau urbain

La simulation de trafic routier est une catégorie d'applications utilisées pour modéliser et reproduire la dynamique des systèmes de transport (Krajzewicz, 2010). Elle est généralement utilisée pour la conception, l'aide à la planification et l'exploitation des systèmes de transport. Elle permet l'étude de modèles complexes pour l'analyse ou le traitement numérique. Après la construction du réseau routier de la ville de Calais sous SUMO, le prochain objectif est de générer une simulation réaliste de trafic routier sur le réseau. Pour chaque simulation, une demande de trafic est fixée, ce qui génère une certaine condition de trafic sur le réseau. La demande de trafic est définie par le nombre de véhicules souhaitant circuler sur le réseau et les informations concernant leurs itinéraires. Pour chaque véhicule, on peut définir le point de départ, le point de destination, l'heure de départ et l'itinéraire du véhicule. Définir la demande de trafic revient à définir des trajets, donc une série de rues à traverser, et à choisir le nombre d'entrées de véhicules sur ces trajets. Afin d'estimer une demande réaliste, il est nécessaire de connaître l'offre du réseau et ses paramètres qui peuvent agir sur la capacité de l'infrastructure. La capacité est définie par le nombre maximal de véhicules pouvant être écoulés dans un temps donné. Cela dépend de paramètre tel que le nombre de voies, la vitesse maximale dans la zone, etc. Il existe différentes approches pour caler une demande de trafic sur le réseau. Une première approche nécessite à définir une demande manuellement dans NetEdit, il faut définir des trajets (on sélectionne les rues qui constituent un trajet), puis on ajoute des véhicules qui vont emprunter ce trajet à différents temps de la simulation. Par exemple 5 véhicules peuvent commencer leur trajet à un temps t=0, et 5 autres à t=10, etc. Une autre approche est le modèle de génération directe de la matrice origine et destination (O-D). Une matrice O-D suppose que la région étudiée est divisée en un ensemble fini de zones ne se chevauchant pas. Cette matrice enregistre le nombre total de voyages des zones d'origine vers les zones de destination qui ont eu lieu dans un jour et un intervalle de temps bien déterminé. La troisième approche est de générer des trajets aléatoires avec le programme Python randomTrips.py. On choisit ici un temps de début et un temps de fin (par exemple de t=0 à t=300) et une fréquence d'apparition de véhicules (par exemple 0.6). Le programme, dans sa définition des routes, fait apparaître aléatoirement un véhicule et une route associée entre les temps 0 et 300, tous les 0.6 s.

Dans notre étude, nous définissons la demande par les matrices O-D avec un trafic important sur les axes principaux et secondaires (figure 1.9). Les données de cette matrice comprennent 10000 voyages pour une duré temporelle de 24 heures. Nous avons décidé d'insérer les véhicules à des intervalles de temps uniformes pendant la simulation. Les données de cette matrice ne peuvent pas être utilisées directement dans SUMO. L'outil OD2TRIPS permet de définir les points source et destination du trajet des véhicules (Maiorov et al.,

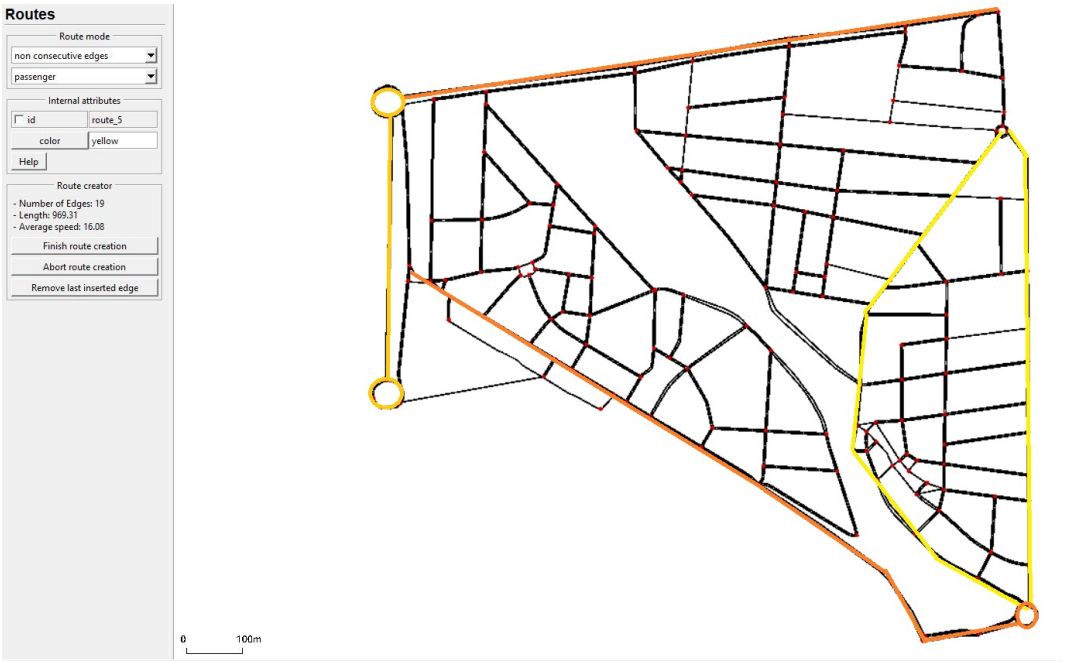


FIGURE 1.9 – Définition manuelle des routes avec NetEdit

2019). Une fois que la demande de trafic est générée, l'affectation du trafic dans SUMO peut être exécutée. Plusieurs outils peuvent être utilisés pour l'affectation de la demande du trafic routier, comme par exemple, DUAROUTER, DFROUTER et JTRROUTER (Lopez et al., 2018). Dans le cadre de nos travaux, nous utilisons DUAROUTER. L'outil DUAROUTER est nécessaire pour générer l'itinéraire complet de chaque voiture et pour valider qu'il existe un chemin connecté entre chaque source et destination, si aucun itinéraire n'est trouvé entre la source et la destination le trajet sera rejeté lors de la simulation. Les données issues d'une simulation sont stockées dans un fichier de format XML facile à interpréter sous Matlab. Il contient les détails sur le nombre des voitures, les longueurs de voyages, le temps d'attente, les vitesses pratiquées, etc. D'après ce rapport, on peut estimer les vitesses, la densité et le débit des véhicules sur chaque route dans le réseau durant une période donnée.

Le trafic routier suit des lois de la physique qui expliquent la majorité des phénomènes aperçut sur les réseaux routiers. Depuis plusieurs années, ces lois sont étayées par l'observation de ces phénomènes à travers des dispositifs de comptage : débit, vitesse, occupation, type de véhicule, etc. Ces comptages permettent de comprendre ces phénomènes physiques, de les mesurer, de les expliquer et de les relier à la connaissance théorique de l'écoulement routier. Afin d'étudier les états du trafic routier dans un réseau de transport, il faut intégrer la notion de pondération des graphes. Donc, dans ce contexte, il faut considérer un graphe pondéré. Nous présentons par la suite les variables de pondération associées à un réseau de transport. Pondérer les arêtes d'un graphe, permet d'appliquer plusieurs études et réaliser

des analyses sur le réseau de transport, comme le partitionnement de graphe (Ji and Geroliminis, 2012b; Saeedmanesh and Geroliminis, 2016). Les poids des arêtes dans le graphe dépendent de la variable d'analyse utilisée dans ce réseau. Dans le contexte de partitionnement, nous distinguons trois variables pour la pondération des arêtes d'un graphe : le débit, la concentration (densité) et la vitesse. Le débit noté Q correspond au nombre de véhicules passant sur une route du réseau par unité de temps. L'interaction entre le nombre de véhicules et la capacité des routes disponible détermine le niveau de congestion de trafic routier (Çolak et al., 2016). Un faible débit correspond à un état fluide induit par une faible demande, et un haut débit correspond à un état congestionné. La concentration notée K_c est le nombre de véhicules présents dans un espace donné. La concentration est un critère d'affectation de pondération des arêtes adopté par de nombreux travaux (Ji and Geroliminis, 2012b; Pascale et al., 2015; Zhou et al., 2012). L'avantage de cette méthode est son efficacité à distinguer les différentes états de trafic : fluides et congestionnés. Un état est considéré fluide lorsque la demande de trafic routier est inférieure à la capacité de la zone d'étude et un état est congestionné lorsque la demande de trafic est supérieure à la capacité d'écoulement de la zone d'étude. Toutefois, un inconvénient identifié est la difficulté à mesurer expérimentalement les concentrations effectives. Ces deux quantités peuvent être calculées de manière discrète, par le débit en une position fixée x, pendant une période Δt. Et par la concentration à une période défini t, entre les positions x et Δx, respectivement. La vitesse est au croisement des deux variables : le débit et la concentration (Saeedmanesh and Geroliminis, 2016). La vitesse notée ν peut être calculée par le rapport de la distance traversé au temps de parcours. Les états fluides présentent des vitesses libres et les états congestionnés présentent des vitesses réduites. Dans notre cas, le coût d'une arête e_i entre deux nœuds v_i et v_j est donné par la vitesse moyenne spatiale des véhicules situés sur la route correspondante.

1.7 Description des données réelles

La carte sélectionnée est celle du quartier des Cailloux et des Fontinettes à Calais, ces quartiers sont principalement des quartiers d'habitations, ils permettent de rejoindre le centre-ville et l'autoroute et sont reliés à d'autres quartiers. Le réseau est composé de plusieurs axes principaux en rouge sur la figure 1.10 : la rue de Verdun à fort trafic qui permet de rejoindre l'autoroute et la plage, le boulevard Gambetta qui amène au Théâtre de Calais et au Channel. Il y aussi des axes importants mais secondaires (en vert sur la figure) comme la rue de Toul, la rue des Fontinettes et la rue de Valenciennes, qui sont des axes entre quartiers. Les principales intersections sujettes à congestions sont représentées par des étoiles sur la figure 1.10. L'axe principal est celui formé par la suite rue de Verdun - boulevard Gambetta qui présente un trafic continu important en heure de pointe. Cette figure sert de référence pour

les noms des points et rues évoqués.

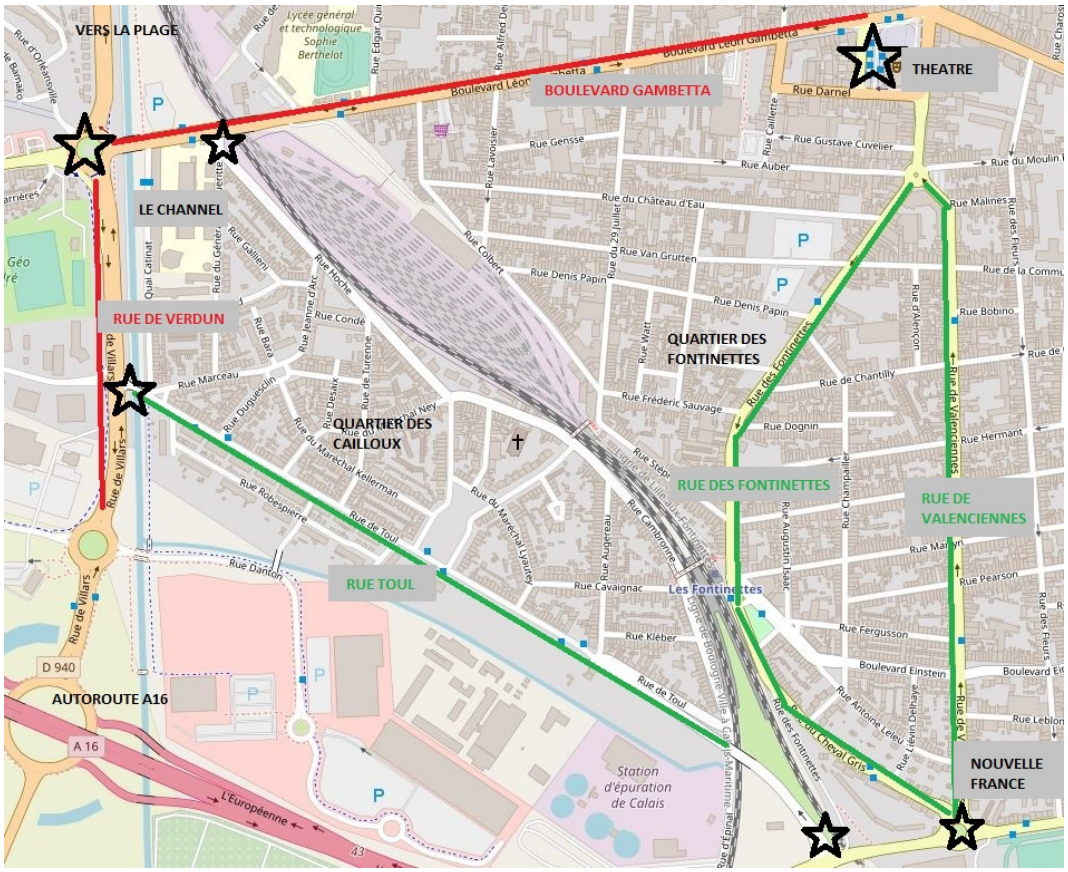


FIGURE 1.10 – Carte annotée de la zone étudiée montrant les quartiers des cailloux et des fontinettes.

1.7.1 Exploitation des résultats

Le réseau de transport $\aleph = (\mathcal{C}, R)$ de la ville de Calais est extrait à partir de la base de données d'OSM. Le résultat de la simulation est un fichier au format XML contenant les données de chaque rue, comme la densité en véhicules, la vitesse moyenne, le nombre de véhicules entrés et sortis de la rue, le temps moyen passé, le temps moyen d'arêt.

En première étape, on lit les fichiers XML contenant les résultats de la simulation pour les convertir en structure Matlab. Les résultats obtenues concernent chaque sens de circulation sur une route, par exemple pour une rue à double sens "classique", on dispose des vitesses moyennes, densités, taux d'occupation et le temps d'attente moyen pour les deux sens de circulation. Ce qui nous conduit très naturellement à réaliser un graphe orienté. Ici, par souci de simplification, nous allons travailler avec un graphe non orienté. Nous allons donc simplifier la liste obtenue en ignorant les sens de routes, c-à-d, pour les routes à double

sens, nous ajoutons une arête non orienté entre les pairs des noeuds. Le graphe $G = (\mathcal{V}, E)$ est donc construit en suivant la méthode du graphe primal introduite dans la section 1.4.1. L'ensemble des noeuds $\mathcal{V}$ est représenté par l'ensemble des croisements $\mathcal{C}$ du réseau $\aleph$, et l'ensemble des arêtes E entre les noeuds est représenté par l'ensemble des routes R.

1.7.2 Analyse simple des données de la simulation

Nous avons proposé un modèle capable de représenter les données du trafic routier afin de générer des stratégies permettant de contrôler un flux perçu par l'intermédiaire de la simulation. Dans la suite, nous étudions par les moyens des graphes les données du trafic. Cette étude permet de caractériser le réseau routier et son système de circulation. Le graphe $G(\mathcal{V}, E)$ illustré dans la figure 1.11 est le graphe routier résultant du réseau routier de la ville de Calais. Il est décrit par sa matrice d'adjacence $\boldsymbol{A}$ de dimension $|\mathcal{V}| \times |\mathcal{V}|$ avec $|\mathcal{V}| = 162$ où $a_{ij} = 1$ si v_i est adjacent à v_j, c'est-à-dire lorsqu'il y a une arête reliant deux noeuds v_i et v_j (Eq. 1.1) et $|E| = 260$ arêtes. Notons que le graphe ici est non pondéré et symétrique. Ensuite, nous avons calculé les degrés de chaque nœud comme dans l'équation 1.7 et construit la matrice de degré diagonale $\boldsymbol{D}$ du graphe.

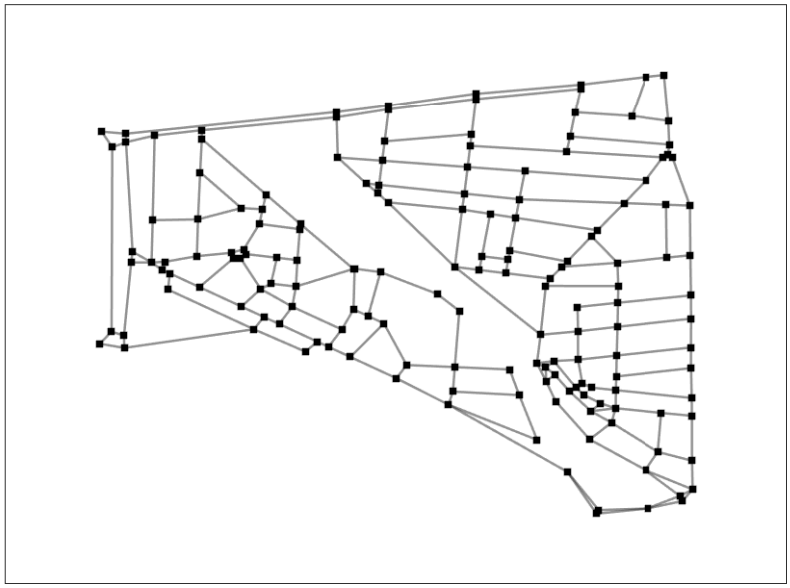


FIGURE 1.11 – Graphe primal correspondant à la carte étudiée

Dans un premier temps, nous observons un exemple des données résultantes de la simulation. On s'intéresse en particulier aux données du nombre de véhicules qui sont entrées sur chaque route, aux vitesses moyennes et aux densités en véhicules de chaque route. La

figure 1.12 montre que le nombre d'entrées est important sur les axes principaux et secondaires définis précédemment et moins important ailleurs. On observe également une forte densité et une faible vitesse au niveau de certaines intersections, c'est le cas au niveau du Boulevard Gambetta et rue de Verdun en direction du Nord. On voit que la vitesse est relativement élevée dans les zones les moins fréquentées, ce qui s'explique par la proximité des rues. Dans les réseaux de transport les mesures de la centralité aident à caractériser une ville

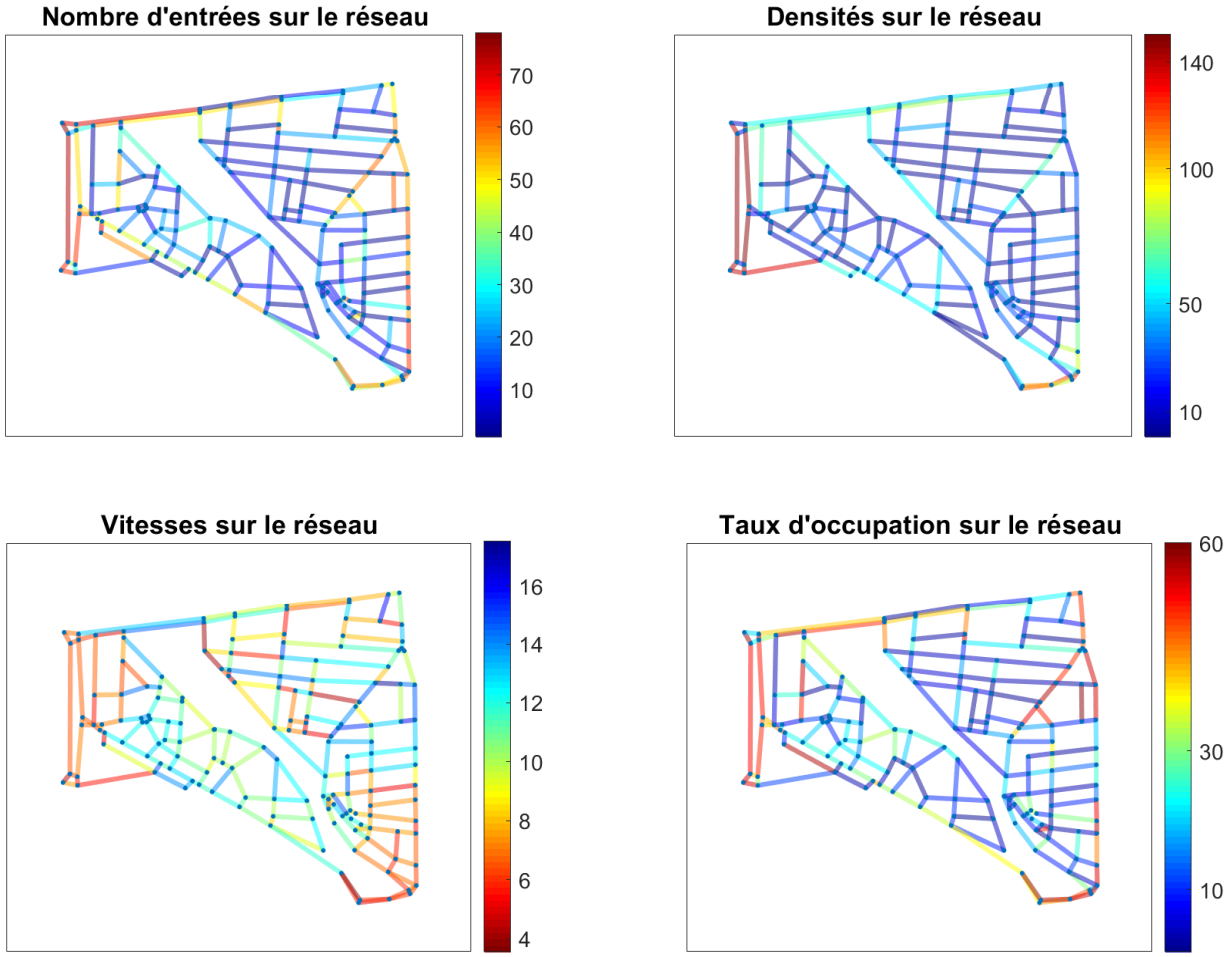


FIGURE 1.12 – Données issues de la simulation

et à comprendre sa topologie, c'est-à-dire, à repérer les endroits caractéristiques d'une ville, par exemple, identifier les lieux les plus centraux, les lieux "ponts" les plus traversés, etc. Dans le but de quantifier la notion d'importance des routes dans un réseau nous appliquons ces mesures sur le graphe routier. De ce fait, on pondère le graphe G en affectant à chaque arêtes $e = (v_i, v_j)$ reliant les deux noeuds v_i et v_j la valeur de la vitesse moyenne de la route correspondante r_i.

La figure 1.13 illustre les différentes mesures de centralité caractérisant le graphe routier.

En effet, sur la figure 1.13 on voit que la centralité de degré est plus importante en dehors des axes principaux, on voit également qu'en terme de centralité de proximité pour les vitesses, on a une forte centralité au niveau de l'intérieur des quartiers, ce qui confirme l'existence de ces zones où la vitesse est faible "par construction". L'analyse de centralité d'intermédiarité permet de mettre en évidence les axes qui sont responsables dans la diminution de la vitesse pour un trajet donné. Cette mesure détermine les noeuds nommés "ponts" qui sont les plus souvent traverser pour aller d'un lieu quelconque à un autre. On voit que l'intermédiarité en vitesse est forte au niveau des axes principaux comme le Channel, rue de valenciennes, rue de toul et au niveau de la salle Nouvelle-France et donc que ces points sont potentiellement des points de ralentissements importants.

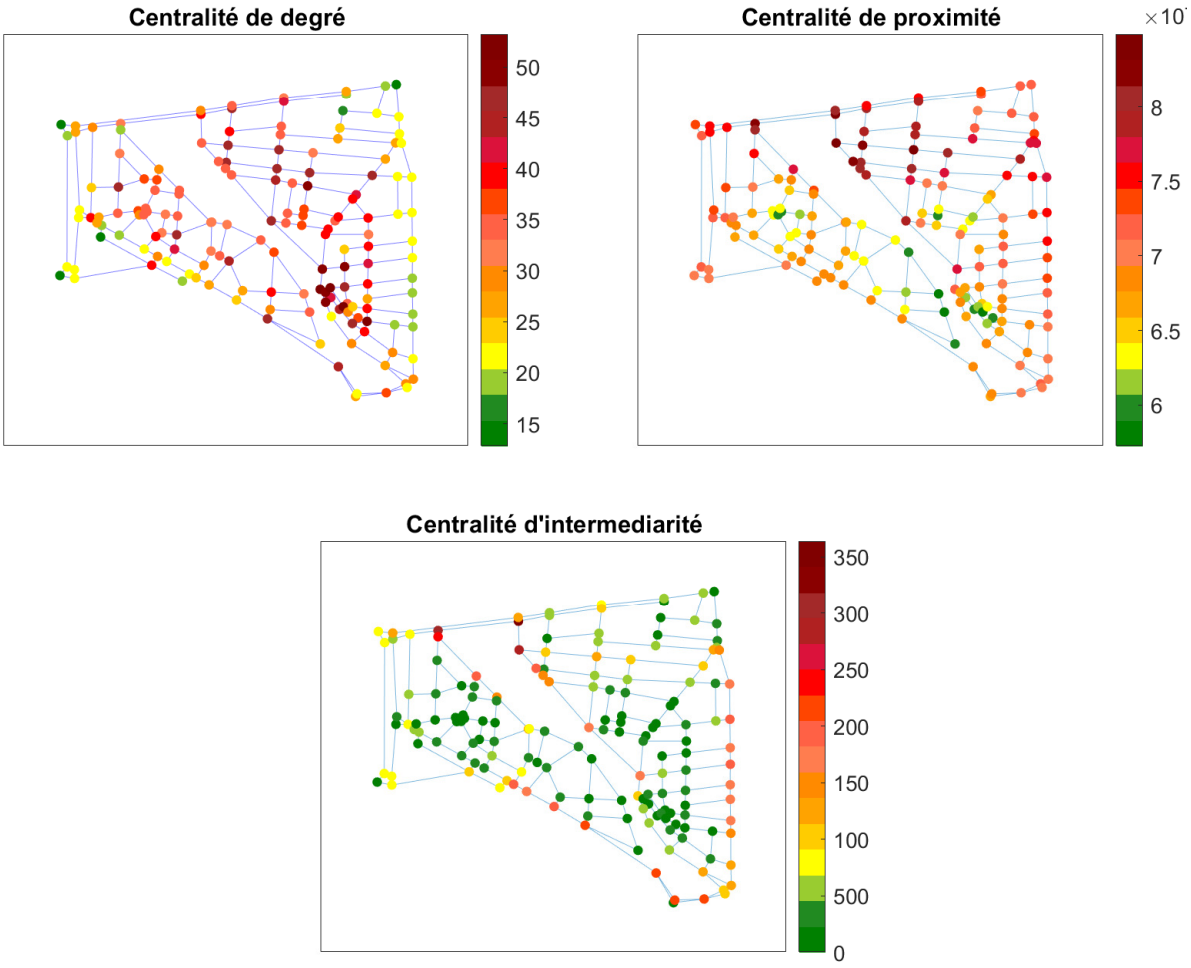


FIGURE 1.13 – Calculs de centralité basés sur la vitesse

1.8 Conclusion

La modélisation de la congestion du trafic peut être approchée par les réseaux déterminés par la théorie des graphes. De ce fait, nous avons présenté dans ce chapitre, différentes méthodes de construction d'un réseau de transport en un graphe routier. Ce qui nous a ramené à introduire quelques concepts et définitions des graphes, notions, les méthodes de calcul et de construction des matrices associées au graphe, le calcul du degré d'un nœud et la construction de la matrice de degré diagonale associée au graphe.

Nous avons présenté une modélisation microscopique du réseau de transport que nous avons extrait à partir d'OpenStreetMaps, en utilisant le simulateur de réseau routier SUMO. Les données issues de la simulation apportent plusieurs informations sur le comportement des véhicules sur le réseau de transport, en plus, des informations sur les dispositifs de comptage du trafic. Dans l'analyse des réseaux routiers, certaines routes ont plus d'effets sur le réseau que d'autres. Cette importance peut être quantifiée par une mesure de centralité. Dans ce chapitre, nous avons présenté plusieurs mesures de centralités qui permettent de décrire l'importance de chaque nœud dans le graphe.

Pour améliorer la gestion de la congestion du trafic, il semble pertinent d'appliquer des stratégies de classification automatique dans l'objectif est d'identifier des zones de congestions dans le réseau de transport. Le chapitre suivant introduit une méthode de classification automatique via un partitionnement basé sur la recherche de pics de densité utilisant la densité du trafic et la connectivité d'adjacence.

Chapitre 2

Classification par recherche de pics de densité

Sommaire

2.1 Introduction

La classification non-supervisée consiste à rechercher dans un ensemble de points, des groupes naturels (appelées classes) tels que les points d'un même groupe sont les plus similaires entre eux alors que les points appartenant à des groupes différents sont les moins similaires les uns des autres. Il existe un grand nombre d'approches de classification non supervisée, comme la classification basée sur les densités.

La classification basée sur les densités contient une variété importante d'algorithmes appartenant à la famille des "boules". On peut mentionner l'emploi des boules en classification non supervisée dans Flamenbaum et al. (1979) et dans le contexte supervisé dans l'algorithme Radial Basis Function Networks(RBF) (Moody and Darken, 1989). L'idée de base est de couvrir l'espace par des boules de même rayon centrées sur les points de données. Les noyaux des classes sont des boules de fortes densités. Ester et al. (1996) ont proposé un algorithme nommé DBSCAN (density based spatial clustering of applications with noise) qui est un des algorithmes les plus populaires de ce type. L'idée principale est d'identifier les classes en recherchant le voisinage de chaque point de données dans un rayon ϵ prédéfini et de vérifier qu'il contient au plus $MinPts$ de points dans une boule. Cependant, cet algorithme est très sensible aux seuils ϵ et au paramètre $MinPts$, et le choix d'un seuil approprié est très important pour obtenir des classes adéquates (Rodriguez and Laio, 2014). Au-delà des concepts de DBSCAN, Xu et al. (2007) ont développé un algorithme nommé SCAN (structural clustering algorithm for networks) qui partitionne un graphe en fonction de sa structure pour détecter les classes et les points aberrants. Toutefois, la performance de cet algorithme dépend fortement de paramètres d'entrée. Rodriguez and Laio (2014) ont récemment proposé une méthode de clustering basée sur la densité en cherchant les pics de densité au niveau local. L'algorithme de partitionnement par pics de densité (Density peaks clustering) permet de mettre en évidence des groupes non-convexes, donnant ainsi des règles de décision appropriée pour identifier les classes de manière significative (Rodriguez and Laio, 2014).

Ce chapitre porte sur l'analyse des zones de congestion dans le réseau routier urbain (figure 2.1), en particulier la détection de ces zones. L'objectif est donc d'appliquer la méthode de classification basée sur la recherche de pics de densité pour la détection de zones de congestion dans la ville de Calais. Dans un premier temps, nous allons appliquer cette méthode à des distributions gaussiennes de points dans un espace 2D, puis nous allons adapter l'algorithme à notre étude de cas sur la recherche de zones de congestion dans un réseau routier.

FIGURE 2.1 – Zone de congestion à Calais

2.2 Description de l'algorithme pics de densité

L'algorithme pics de densité est basée sur la recherche de points présentant une densité élevée qui serviront de "centres" pour les classes. Les centres des classes doivent vérifier deux propriétés : (1) les centres sont entourés par des points voisins de faible densité locale et (2) ils sont à une distance relativement grande des points de densité locale plus grande. La densité locale d'un point est le nombre des points dans la boule centrée sur ce point. De ce fait, chaque point de données est caractérisé par deux quantités : sa densité locale ρ_i et sa distance δ_i des points de plus fortes densités. Considérons un ensemble de N points $X = \{x_1, \ldots, x_N\}$. La matrice des distances $\mathcal{D}$ est définie, telle que, $\mathcal{D}_{ij} = d(x_i, x_j)$ qui est la distance entre les couples de points x_i et x_j, où $i \neq j$:

$$\mathcal{D} = \begin{pmatrix} 0 & d(x_1,x_2) & d(x_1,x_3) & \ldots & d(x_1,x_N) \\ d(x_2,x_1) & 0 & d(x_2,x_3) & \ldots & d(x_2,x_N) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ d(x_N,x_1) & d(x_N,x_2) & d(x_N,x_3) & \ldots & 0 \end{pmatrix}.$$

— *La densité locale* : Rodriguez and Laio (2014) ont défini deux méthodes possibles pour le calcul de la densité. La définition la plus simple de la densité locale pour le point x_i est le nombre de points présents dans le voisinage de ce point, c'est-à-dire le nombre de points dont la distance à x_i est inférieure à un seuil d_c. Plus formellement, la densité locale est définie par :

$$\rho(x_i) = \sum_{j=1}^{N} \chi(\mathcal{D}_{ij} - d_c) \text{ avec } \chi(x) = \begin{cases} 1 & \text{si } x < 0 \\ 0 & \text{sinon.} \end{cases}. \tag{2.1}$$

Le paramètre d_c est une distance fixée qui permet de compter pour chaque point x_i le nombre de points qui sont à une distance inférieure ou égale à d_c. Le choix de ce paramètre est sensible aux performances de la classification. Une autre approche consiste à estimer la densité par des fonctions noyaux gaussiennes (Lin, 2019) :

$$\rho(x_i) = \sum_{j=1}^{N} \exp(-\frac{\mathcal{D}_{ij}^2}{d_c^2}) \tag{2.2}$$

— *La distance* : $\delta(x_i)$ consiste à rechercher pour chaque x_i le point le plus proche parmi ceux les plus denses. Pour le point dont la densité est maximale, on associe la plus grande distance possible, donc la distance au point le plus éloigné (Lin, 2019) :

$$\delta(x_i) = \begin{cases} \min\limits_{j:\rho(x_j)>\rho(x_i)} \mathcal{D}_{ij} & \text{si } \rho(x_i) < \max\limits_{x_j \in X} \rho(x_j) \\ \max\limits_{x_j \in X} \mathcal{D}_{ij} & \text{sinon.} \end{cases} \tag{2.3}$$

Par la suite, nous définissons par $\sigma(x_i)$ l'indice j du point de données x_j qui est le plus proche du point x_i, tel que $\rho(x_j) > \rho(x_i)$. Pour les points dont les densités sont maximales, $\sigma(x_i)$ est fixé à i :

$$\sigma(x_i) = \begin{cases} \arg\min\limits_{j:\rho(x_j)>\rho(x_i)} \mathcal{D}_{ij} & \text{si } \rho(x_i) < \max\limits_{x_j \in X} \rho(x_j) \\ i & \text{sinon.} \end{cases} \tag{2.4}$$

— *Le graphe de décision* : Les données sont représentés par leurs coordonnées $(\rho(x_i), \delta(x_i))$ dans le plan densité-distance dit graphe de décision. Ce graphe donne à l'utilisateur une idée sur la structure des données et permet la sélection du nombre de classes. Les centres des classes sont caractérisés par des points ayant relativement une densité élevée et assez distants. Ils occupent donc la zone en haut à droite du graphe de décision. Les points aberrants ou "outliers" sont les points ayant une faible densité et une grande distance. Pour les noyaux des classes, on procède comme suit. Pour chaque classe, on considère sa frontière, définie comme l'ensemble de points affectés à la classe mais qui sont à une distance d_c des points appartenant à d'autres classes. Nous cherchons, pour chaque classe, le point de plus forte densité à l'intérieur de la région frontière. Les points de la classe dont la densité est supérieur à celle-ci sont considérés comme noyau de la classe. Une fois les centre des classes sont déterminés, chaque point restant est affecté à la même classe que son plus proche voisin de plus forte densité. L'affectation est faite en une seule étape.

— *La règle gamma* : Une autre approche a été envisagée pour sélectionner le nombre de classe qui est la règle gamma. Cette approche utilise le produit des deux caractéristiques δ et ρ et réalise un seuillage de cette quantité :

$$\gamma(x_i) = \delta(x_i)\rho(x_i) > \gamma_{seuil}. \tag{2.5}$$

Les points "centraux" sont les points x_i tels que $\gamma(x_i) > \gamma_{seuil}$, γ_{seuil} étant un seuil à définir par l'utilisateur (Rodriguez and Laio, 2014).

L'algorithme 1 résume la méthode de partitionnement par pics de densité.

Algorithme 1 : Algorithme DPC (Density Peaks Clustering)

Résultat : Vecteur C tel que $C(x_i)$ est la classe du point x_i
Entrées : un ensemble de points $X = \{x_1,...,x_N\}, x_i \in \mathbb{R}^M$;
Le seuil d_c;
Calculer la densité $\rho(x_i)$ de chaque point x_i;
Calculer $\delta(i)$ et $\sigma(x_i)$ pour chaque point x_i;
Afficher les $\delta(x_i)$ en fonction des densités $\rho(x_i)$;
Sélectionner manuellement les pics de densités sur le graphe;
Pour chaque pic de densité x_i, on assigne $C(x_i) = k$, $k = 1..K$ pour K pics de densité;
Pour chacun des autres points x_i, parcourus par ρ décroissant, on assigne $C(x_i) = C(\sigma(x_i))$;

2.3 Application de DPC sur un exemple simple

Dans cette section, nous appliquons l'algorithme DPC et nous détaillons ses étapes par un exemple pédagogique. La figure 2.2 présente la distribution de l'ensemble de points $X = \{x_1,...,x_N\}$ à analyser. La première étape, consiste à calculer les densités de chaque point en suivant l'équation 2.2, comme indiqué précédemment. Les densités sont résumées dans le tableau 2.1.

TABLEAU 2.1 – Densités des points de l'exemple simple.

Point	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
ρ	3	4	3	3	5	3	3	3	4	1

Par la suite, pour chaque point x_i nous déterminons les points de densité supérieure les plus proches. Sauf pour le point de densité maximale, auquel on associe la distance au point

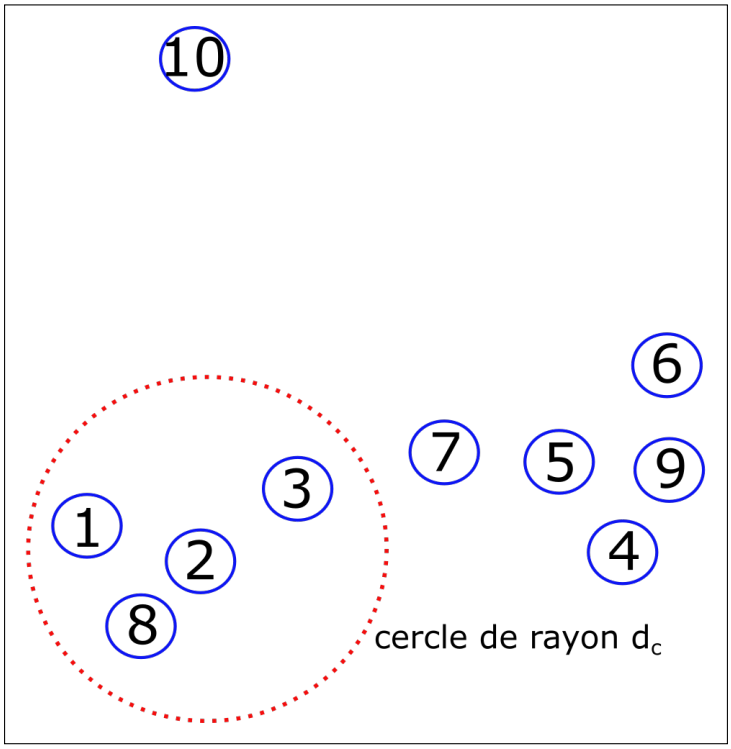


FIGURE 2.2 – Exemple de distribution simple

le plus éloigné, le tableau 2.2 présente les résultats obtenus.

TABLEAU 2.2 – δ et σ des points de l'exemple pédagogique.

Point	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
δ	$d(1,2)$	$d(2,5)$	$d(3,2)$	$d(4,9)$	$d(5,10)$	$d(6,9)$	$d(7,5)$	$d(8,2)$	$d(9,5)$	$d(10,3)$
σ	2	5	2	9	5	9	5	2	5	3

Une fois les valeurs de δ et σ et ρ sont calculées, nous traçons par la suite le graphe de décision, où les densité ρ sont en abscisse et δ sont en ordonnée. La figure 2.3 illustre le graphe de décision résultant. Sur ce graphe, on distingue les trois zones possibles : les centres des classes de δ et ρ élevés ; les valeurs aberrantes de δ élevé et de ρ faible ; et les autres points. Sur ce graphe, nous remarquons que les points 2 et 5 ont les plus hautes valeurs de δ et ρ, alors, ces deux points deviennent centres d'une classe chacun. Ainsi, on associe une classe à chaque centre : $C(x_5) = 1, C(x_2) = 2$.

Par la suite, nous attribuons les autres points d'une manière récursive à la classe la plus proche :

$C(x_9) = C(\sigma(x_9)) = C(x_5) = 1$

$C(x_1) = C(\sigma(x_1)) = C(x_2) = 2$

$C(x_3) = C(\sigma(x_3)) = C(x_2) = 2$

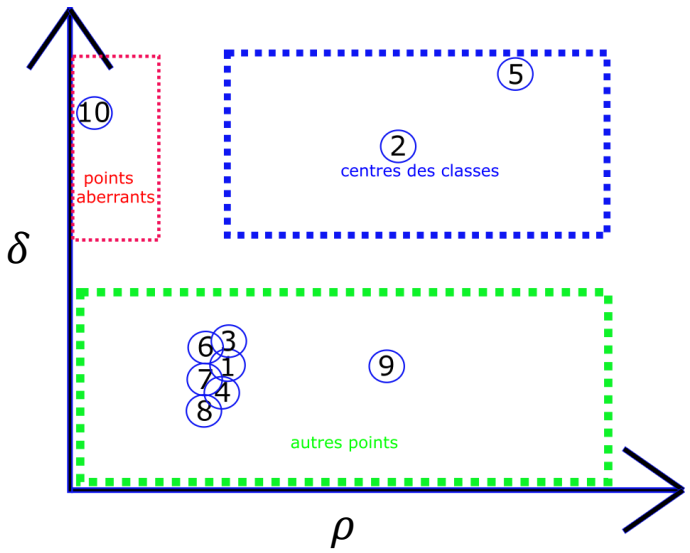


FIGURE 2.3 – Graphe de décision obtenu pour l'exemple pédagogique.

$C(x_4) = C(\sigma(x_4)) = C(x_9) = C(x_5) = 1$
$C(x_6) = C(\sigma(x_6)) = C(x_9) = C(x_5) = 1$
$C(x_7) = C(\sigma(x_7)) = C(x_5) = 1$
$C(x_8) = C(\sigma(x_8)) = C(x_2) = 2$
On a donc : $C_1 = \{4,5,6,7,9\}$ et $C_2 = \{1,2,3,8\}$.

2.4 Elimination des valeurs aberrantes

Nous considérons toujours notre ensemble de points $X = \{x_1,...,x_N\}$ et nous supposons connus le vecteur indicateur de détermination C tel que $C(x_i)$ indique la classe du point x_i. La recherche de valeurs aberrantes va consister à chercher dans chaque classe C_k les points dont la densité est inférieure à la densité moyenne maximale à la frontière de chaque classe (Li and Tang, 2018; Rodriguez and Laio, 2014).
Pour cela, nous définissons un vecteur $\rho^{frontmax} = (\rho_1^{frontmax},...,\rho_K^{frontmax})$ avec $\rho_k^{frontmax}$ la densité moyenne maximale à la frontière de la classe C_k. Pour chaque couple de points (x_i, x_j) avec $C(x_i) \neq C(x_j)$ et voisins, donc formant une frontière, nous calculons $\rho^{moy} = \frac{\rho_i+\rho_j}{2}$ la moyenne des densités des deux points. Par la suite, nous mettons à jour le vecteur $\rho^{frontmax}$ avec les valeurs maximales. Dans le code source fourni par les auteurs Rodriguez and Laio (2014) l'algorithme d'élimination des valeurs aberrantes est résumé par l'algorithme 2.

Reprenons notre exemple pédagogique précédent, nous remarquons une frontière entre

Algorithme 2 : Elimination des valeurs aberrantes

Résultat : Vecteur $\hat{C}$ tel que $\hat{C}(x_i)$ est la classe du point x_i
Entrées : la matrice de distances $\mathcal{D}$, le vecteur de détermination C, le nombre de classes K, le rayon d_c;
On initialise un vecteur $\hat{C}$ égal au départ à C, $\hat{C}(x_i)$ indiquera la classe du point x_i ou un 0 si c'est une valeur aberrante.

```
pour i de 1 à N faire
    pour j de i+1 à N-1 faire
        si C(x_i) ≠ C(x_j) et D_ij < d_c alors
            ρ^moy = (ρ(x_i)+ρ(x_j))/2
            si ρ^moy > ρ^frontmax_C(x_i) alors
                ρ^frontmax_C(x_i) = ρ^moy
            fin
            si ρ^moy > ρ^frontmax_C(x_j) alors
                ρ^frontmax_C(x_j) = ρ^moy
            fin
        fin
    fin
fin
pour i de 1 à N faire
    si ρ(x_i) < ρ^frontmax_C(x_j) alors
        Ĉ(x_i) = 0
    fin
fin
```

les points x_3 et x_7, et donc $\rho_1^{frontmax} = \rho_2^{frontmax} = \frac{\rho(x_3)+\rho(x_7)}{2} = 3$. Dans ce cas, nous pouvons donc éliminer tous les points dont la densité est inférieure à 3. Ce qui est le cas du point aberrant x_{10}. Le partitionnement avec élimination du point aberrant est visible sur la figure 2.4.

2.5 Estimation du rayon d_c

Le paramètre d_c reflète une distance afin de compter pour chaque point de données x_i, le nombre de points dont la distance est plus petite ou égale à d_c. Comme suggéré par Rodriguez and Laio (2014), nous pouvons estimer le rayon d_c tel qu'en moyenne (notée p%) des points se trouvent sous le rayon. Pour cela, on range toutes les distances $d(i,j)$ entre chaque point de données par ordre croissant dans un vecteur d_{sort}. Par la suite, nous estimons le paramètre d_c tel que $d_c = d_{sort}(|d_{sort}| \times \frac{p}{100})$ avec $|d_{sort}|$ le nombre d'éléments de

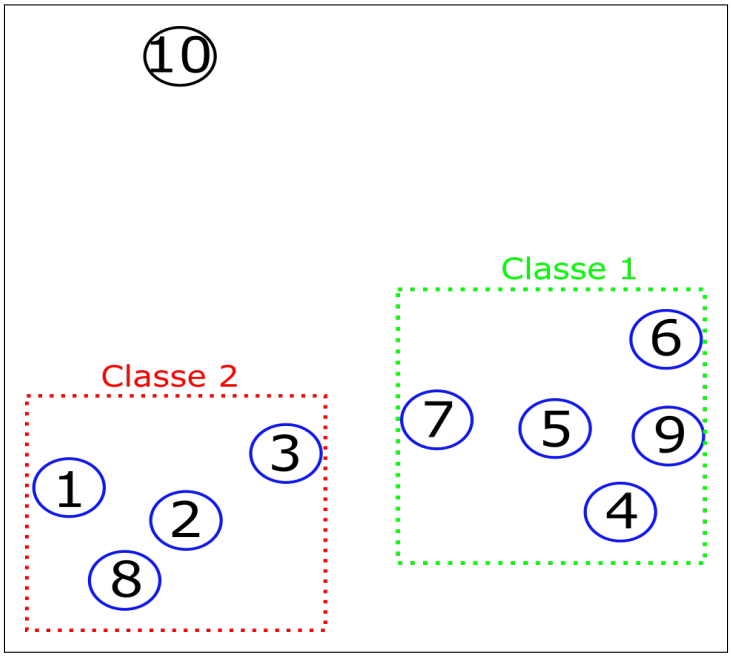


FIGURE 2.4 – Partitionnement de l'exemple simple avec élimination des points aberrants.

$|d_{sort}| = \frac{N\times(N-1)}{2}$, N le nombre de points.
Dans notre exemple, nous allons chercher d_c avec $p = 5\%$, on a $|d_{sort}| = \frac{10\times9}{2} = 45$ et $45 \times 0.05 \approx 2$ donc le d_c estimé est égal à la deuxième plus petite distance de la matrice de distances, qui correspond à la distance entre les points 5 et 9 dans l'exemple, donc $d_c = D_{59}$.

2.6 Test de l'algorithme sur quelques exemples

Nous allons tester cet algorithme sur des exemples de distributions de points, que nous allons comparer à K-means, afin de montrer l'intérêt de la méthode DPC. Dans ces exemples, les centres des classes seront choisis manuellement sur le graphe de décision. Les exemples sont générés à partir d'une distribution gaussienne, que l'on a tronquée afin de dessiner les formes. Pour le calcul du rayon d_c, nous avons choisi un pourcentage de 1% dans les 3 exemples. Le premier ensemble de points testés est réparti en 4 groupes, chacun formant un rectangle. La figure 2.5, illustre le résultat de classification ainsi que le graphe de décision. Sur ce dernier, on a sélectionné les 4 points ayant la distance δ la plus élevée.
Le résultat ne montre pas de différence entre les deux méthodes, en effet, K-means reste performant lorsque les groupes à identifier ne sont pas imbriqués, nous avons donc 0% d'erreurs pour les deux méthodes.
Nous observons en revanche une nette amélioration sur les figures 2.6 et 2.7. Pour ces exemples, nous remarquons bien les limites de K-means lorsqu'il s'agit d'identifier des structures imbriquées, ce que l'algorithme DPC est capable de faire. En effet, nous observons

dans cet exemple un taux de reconnaissance de 100% dans les deux exemples pour DPC, alors qu'avec K-means, il est de l'ordre de 75% dans le premier exemple et de 50% dans le second.

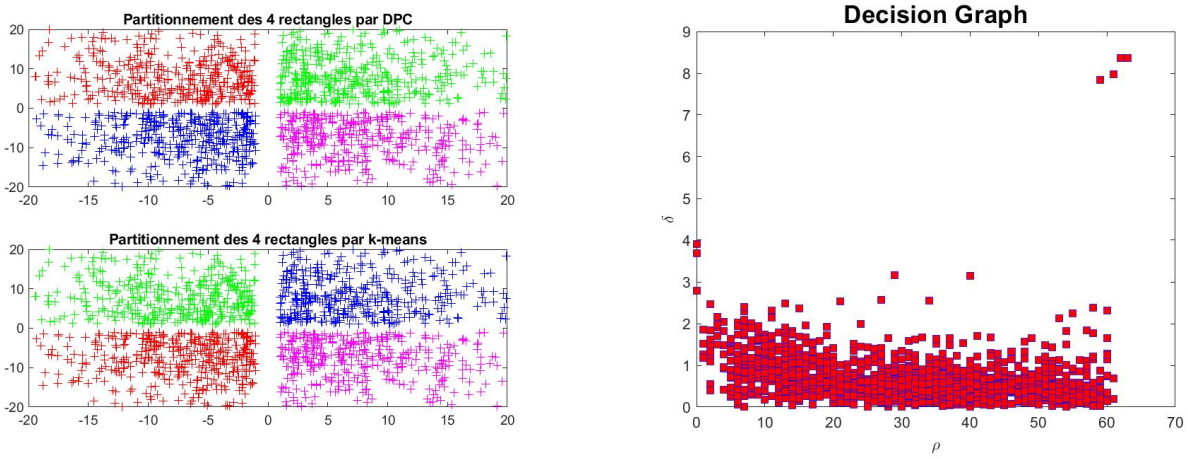


FIGURE 2.5 – DPC et *K*-means sur 4 rectangles.

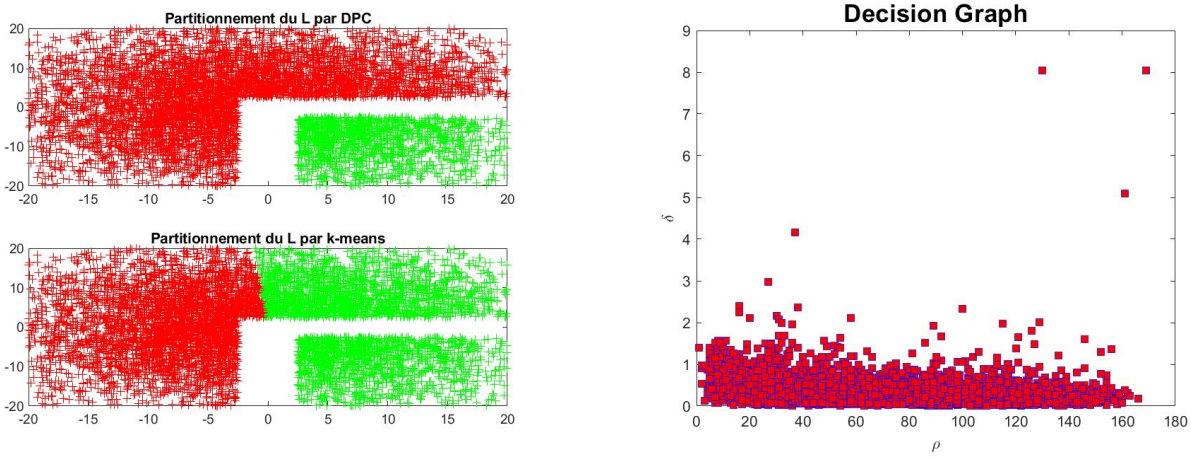


FIGURE 2.6 – DPC et *K*-means sur un exemple de forme L.

Pour conclure cette partie de tests, la méthode DPC présente de nombreux avantages par rapport à un algorithme K-means classique. D'abord elle permet de reconnaître des structures imbriquées là où K-means aura plus de difficultés à le faire avec un nombre de classes limité. La méthode DPC permet également de traiter les valeurs aberrantes, en analysant les densités aux frontières des classes déterminées, ce qui permet d'éliminer "naturellement" des points de la distribution.

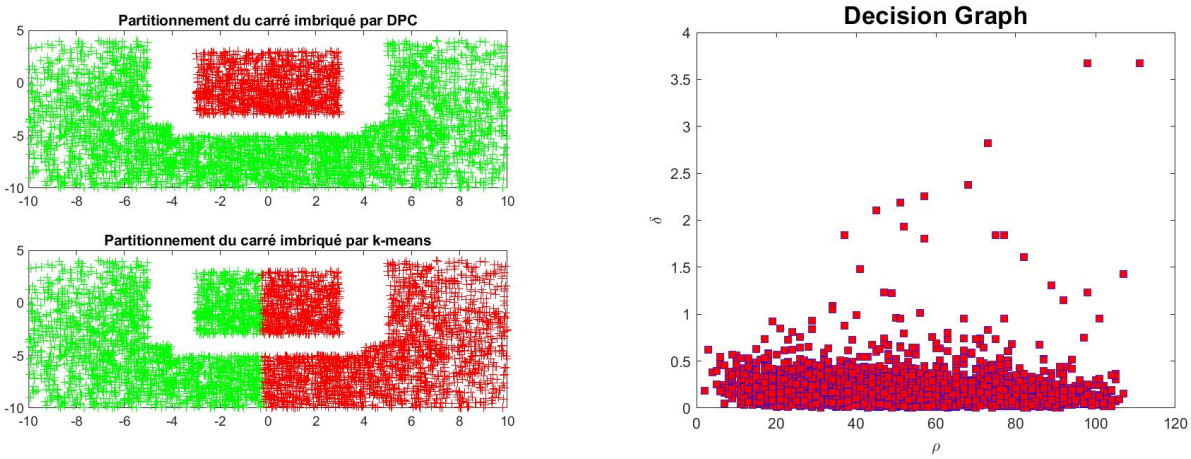


FIGURE 2.7 – DPC et K-means sur un carré imbriqué.

2.7 Sélection automatique du nombre de classes

L'étape de sélection des classes est déterminée par deux méthodes : le graphe de décision et la règle gamma. Ces deux méthodes dépendent des paramètres de seuillage d_c et γ_{seuil} qui sont crucial pour les performances de l'algorithme des pics de densité. En outre, les auteurs Rodriguez and Laio (2014) n'ont pas fourni une méthode efficace pour le choix de ces valeurs et ont laissé ces paramètres à définir par l'utilisateur en se basant sur des expériences. En général, la problématique est de déterminer le seuil optimal des caractéristiques sélectionnées.

En premier lieu, nous introduisons la méthode de calcul du rayon d_c. Wang et al. (2016) ont introduit une méthode de sélection automatique de la valeur de d_c en minimisant une entropie potentielle définie comme suit :

$$H = -\sum_{i=1}^{N} \frac{\varphi_i}{\Phi} log\left(\frac{\varphi_i}{\Phi}\right) \tag{2.6}$$

où $\Phi = \sum_{i=1}^{N} \varphi_i$ est le facteur de normalisation et φ_i est le potentiel scalaire de chaque point, et il est défini par :

$$\varphi_i = \sum_{j=1}^{n} e^{-\frac{\mathcal{D}_{ij}^2}{d_c^2}}. \tag{2.7}$$

Ainsi, l'équation 2.7 est la même que l'équation 2.1 et le potentiel de données distribue les points de la même manière que la densité. Selon cette définition, les points ayant des valeurs élevés de φ_i sont situés dans la région dense. Au final, nous choisissons le rayon d'influence d_c qui minimise l'entropie H selon la règle de distribution gaussienne où un point ne peut

affecter que les points situés dans son rayon d'influence.

Les deux approches de sélection du nombre de classes consiste à fixer un seuil γ_{seuil} pour déterminer le nombre de classes à sélectionner (équation 2.5). L'inconvénient est que ce paramètre doit être déterminé différemment pour chaque ensemble de données. De plus, pour les ensembles de données simples, la valeur du seuil γ_{seuil} peut être sélectionnée facilement par l'utilisateur. Mais pour les ensembles de données complexes et plus grands, il n'est pas facile de sélectionner une valeur de γ_{seuil} optimale. Afin de résoudre ce problème, une méthode d'identification de centres de classes basée sur les statistiques est définie. L'idée de base est que les centres des classes peuvent être considérés comme des valeurs aberrantes avec des distances minimales élevées dans le graphe de décision. Ainsi, une nouvelle méthode de détection statistique des valeurs aberrantes est utilisée pour identifier automatiquement les centres des classes à partir du graphe de décision. Une fonction de densité de probabilité est calculée en incorporant deux valeurs : la probabilité et la variance. La probabilité $\mu_y(\rho_i)$ de la distance y à une valeur spécifique ρ_i est définie par (Yan et al., 2019) :

$$\mu_y(\rho_i) = \frac{\sum_{j=1,j\neq i}^{N} \delta_j exp\left(\frac{-1}{2}\left(\frac{(\rho_j-\rho_i)}{a}\right)^2\right)}{\sum_{z=1,z\neq i}^{N} exp\left(\frac{-1}{2}\left(\frac{(\rho_z-\rho_i)}{a}\right)^2\right)}. \tag{2.8}$$

La variance de la distance y à une valeur spécifique ρ_i est estimée par :

$$\sigma_y^2(\rho_i) = \frac{\sum_{j=1,j\neq i}^{N} \left[b^2 + (\delta_j - \mu_y(\rho_i))^2\right] exp\left(\frac{-1}{2}\left(\frac{(\rho_j-\rho_i)}{a}\right)^2\right)}{\sum_{z=1,z\neq i}^{N} exp\left(\frac{-1}{2}\left(\frac{(\rho_z-\rho_i)}{a}\right)^2\right)}, \tag{2.9}$$

avec a et b sont les largeurs des noyaux 2D. Les valeurs de a et b sont calculées en utilisant les écarts types des ρ_i et des δ_i pour tous les points de données comme suit :

$$\begin{cases} a = \epsilon \times \sigma_\rho & 0 < \epsilon < 1 \\ b = \beta \times \sigma_\delta & 0 < \beta < 1 \end{cases} \tag{2.10}$$

où, ϵ et β sont deux paramètres fixés par l'utilisateur. Ensuite, les centres de classes sont identifiés en utilisant le seuil suivant :

$$\gamma_{seuil}(\rho_i) = \mu_y(\rho_i) + 3 \times \sigma_y^2(\rho_i) \tag{2.11}$$

Tout point de données qui a une distance minimale $\delta_i > \gamma_{seuil}(\rho_i)$ est identifiée comme un centre de classe.

2.8 Classification du graphe routier par pics de densité

Dans la méthode de classification par recherche de pics de densité, nous allons travailler directement sur les caractéristiques de chaque route, et chercher des groupes de routes présentant des caractéristiques similaires. Le nombre de classes sera déterminé automatiquement en sélectionnant des points sur le graphe de décision qui ont une grande valeur de ρ_i et δ_i. L'algorithme DPC présenté précédemment devra être adapté car l'ensemble de points que l'on partitionne représente des routes, et chaque route a des routes adjacentes. Cette dernière contrainte est à prendre en compte dans nos calculs. Nous allons d'abord voir pourquoi la méthode DPC est adaptée à notre étude de cas, puis nous adapterons la méthode aux contraintes d'adjacence.

Une zone de congestion peut être caractérisée ainsi : une forte densité en véhicules et une vitesse moyenne relativement faible, mais également des caractéristiques proches. Par simple observation, on peut dire qu'une zone de congestion est généralement créée sur une seule route, au niveau d'une intersection, puis la congestion se diffuse aux routes voisines. La figure 2.8 illustre un exemple de diffusion de la congestion dans un réseau routier.

Si on considère les routes de notre réseau comme des points possédant chacun des attributs (densité, vitesse moyenne, temps d'attente moyen et taux d'occupation), il y a donc des points pics où ces valeurs sont extrêmes et décroissent dans les rues adjacentes. On retrouve donc bien une distribution en groupes de points, possédant chacun un pic de densité. Ainsi, notre algorithme de classification par pics de densité semble une alternative judicieuse pour la caractérisation de notre réseau.

2.8.1 Description des données

Les points considérés ici vont représenter les routes du réseau de la ville de Calais. Notons, R l'ensemble des N routes étudiées. Chaque route r_i est en fait une structure $< X_i, B_i >$ avec X_i un vecteur de M caractéristiques et B_i le vecteur d'adjacence tel que $B_i(j) = 1$ si r_i et r_j sont adjacentes, $B_i(j) = 0$ sinon. Les B_i forment en fait les lignes de la matrice d'adjacence du graphe dual défini pour la représentation du réseau routier.

Dans notre étude, M=4, les caractéristiques utilisées sont la densité en véhicules, la vitesse moyenne, le taux d'occupation de la route et le temps d'attente moyen sur la route. Le descriptif des variables est dans le tableau 2.3.

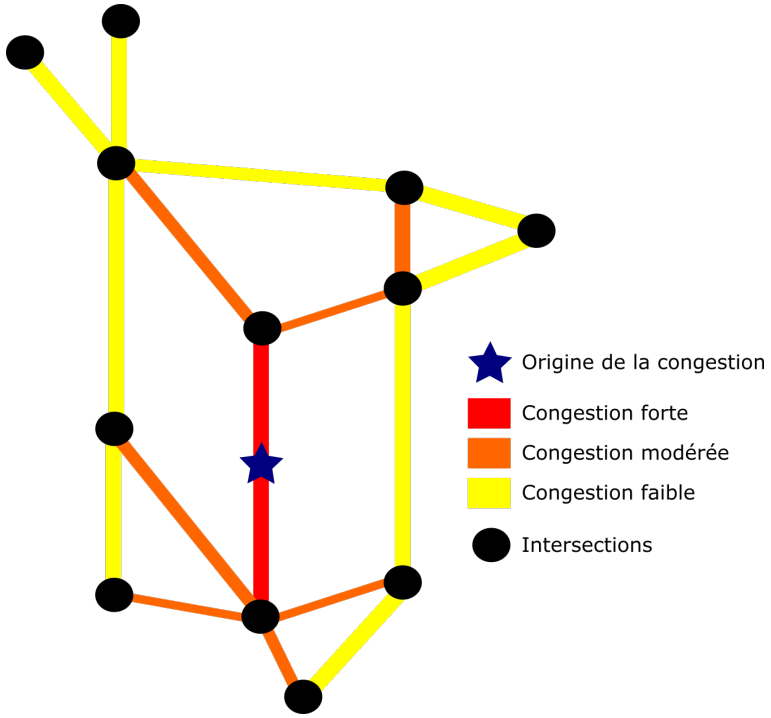


FIGURE 2.8 – Exemple de diffusion de la congestion

TABLEAU 2.3 – Descriptif des données obtenues par simulation avec Sumo

Caractéristique	Descriptif
Densité	Densité en véhicules (en véhicules/km)
Vitesse moyenne	Vitesse moyenne des véhicules sur la route (en m/s)
Taux d'occupation	Pourcentage de la route occupé par des véhicules, une valeur de 100 indiquerait une file de véhicules collés les uns aux autres
Temps d'attente moyen	Temps d'attente total de tous les véhicules à l'arrêt

La figure 2.9 présente un exemple schématique d'un réseau routier composé de 6 routes $r_1,...,r_6$, sa représentation avec les notations précédentes est :

$$r_1 =< X_1, B_1 >, X_1 = [densite(1), occupation(1), vitesse(1), attente(1)], B_1 = [010100]$$

$$r_2 =< X_2, B_2 >, X_2 = [densite(2), occupation(2), vitesse(2), attente(2)], B_2 = [101000]$$

$$r_3 =< X_3, B_3 >, X_3 = [densite(3), occupation(3), vitesse(3), attente(3)], B_3 = [010110]$$

$$r_4 =< X_4, B_4 >, X_4 = [densite(4), occupation(4), vitesse(4), attente(4)], B_4 = [101010]$$

$$r_5 =< X_5, B_5 >, X_5 = [densite(5), occupation(5), vitesse(5), attente(5)], B_5 = [001101]$$

$$r_6 =< X_6, B_6 >, X_6 = [densite(6), occupation(6), vitesse(6), attente(6)], B_6 = [000010]$$

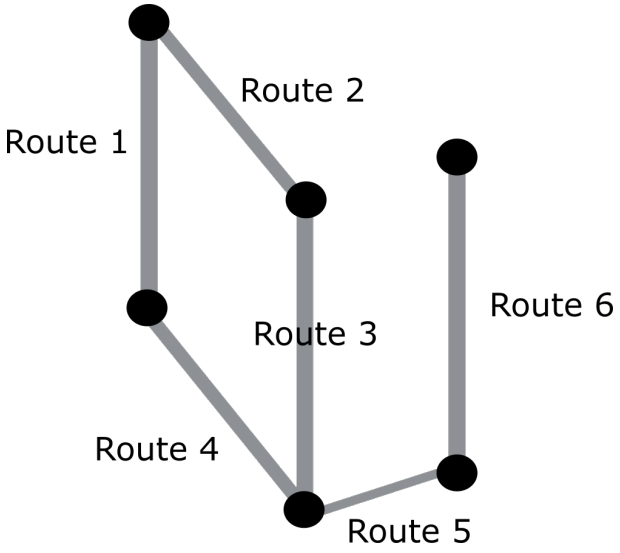


FIGURE 2.9 – Exemple schématique d'un réseau routier

Les X_i forment les lignes de la matrice X des points à partitionner dans un espace de dimension 4.
Nous appliquons l'algorithme DPC dans un espace de 4 dimension sur le graphe routier de Calais. Le graphe $G = (\mathcal{V}, E)$ est composé d'un ensemble de noeuds $\mathcal{V}$ de taille $|\mathcal{V}| = 260$ qui représente les routes du réseau routier et l'ensemble des arêtes représente les intersections entre eux. La figure 2.10 illustre le graphe routier résultant du réseau routier de la ville de Calais.

Nous n'utilisons pas les coordonnées géographiques des routes car leur position ne caractérise pas une zone de congestion, de plus ces données peuvent biaiser le résultat pour les tronçons de routes proches, par exemple les ronds-points définis dans SUMO, sont représentés par plusieurs routes très proches les unes des autres, donc "géographiquement denses". Mais la position des routes a une importance, car une zone de congestion est un ensemble de routes adjacentes. Nous verrons dans la partie suivante comment prendre en compte la position des routes grâce à la notion d'adjacence.

2.8.2 Détection des zones de congestion

Dans cette section, nous introduisons une application de l'algorithme de partitionnement DPC sur le graphe routier afin de détecter les différentes zones de congestion sur le

FIGURE 2.10 – Représentation en graphe du réseau routier à partitionner

réseau. Cette application est composée de 3 étapes : la recherche du voisinage des routes, la recherche automatique du nombre de classes et le partitionnement du réseau en groupes de routes ayant des caractéristiques similaires (algorithme DPC). Par la suite, nous introduisons ces étapes par un exemple d'un graphe routier.

Prise en compte des routes adjacentes

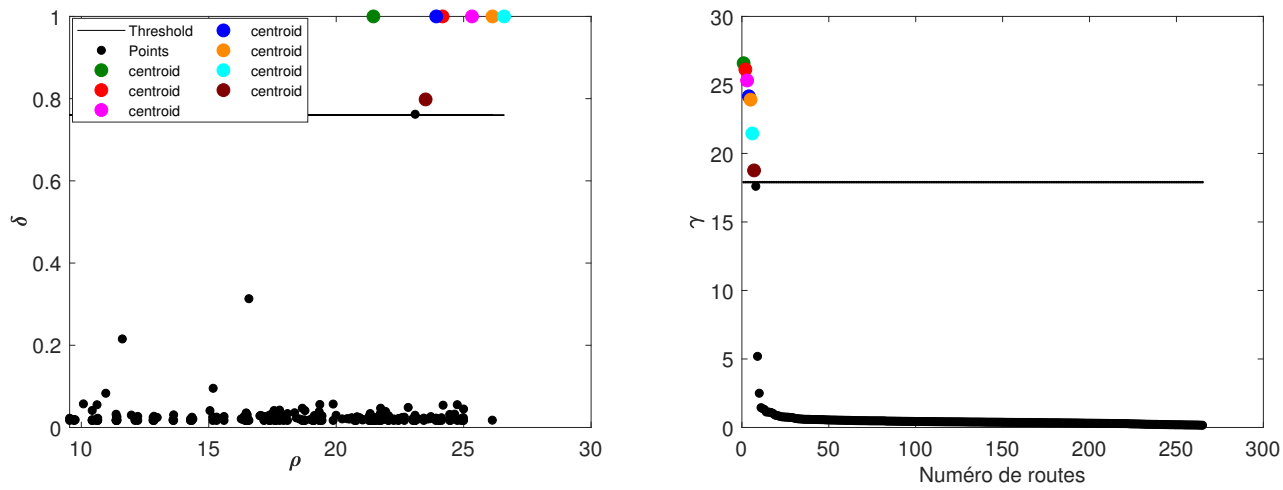
Prendre en compte l'adjacence des routes dans la classification permet d'améliorer les performances de l'algorithme de partitionnement et surtout la connexité des classes obtenues. En effet, l'algorithme DPC recherche des groupes de points denses, donc avec des valeurs similaires, mais dans le cadre d'un réseau routier, 2 routes peuvent avoir des caractéristiques proches, sans pour autant être connectées ensemble. On ajoute donc la prise en compte des routes adjacentes dans le calcul des mesures : δ_i et σ_i. L'objectif est de calculer les σ_i et des δ_i en associant le point le plus proche de densité supérieure, si et seulement s'il représente une route adjacente.
Pour la matrice d'adjacence, nous allons définir $B^{(1)}$ pour une adjacence d'ordre 1 (c'est-à-dire les routes adjacentes à chaque route), $B^{(2)}$ pour un voisinage d'ordre 2 (les routes adjacentes, et les routes adjacentes à ces dernières). Par exemple dans l'exemple de la figure 2.9 le voisinage à l'ordre 1 de la route 3 est composé des routes 2,4 et 5. Le voisinage à l'ordre de 2 de la route 3 contient toutes les routes de la figure. Afin de comparer l'impact de l'ordre d'adjacence sur les résultats de partitionnement, nous choisissons d'introduire une comparaison entre les deux ordres 1 et 2.

La première étape consiste donc à calculer le rayon d_c automatiquement en minimisant l'équation 2.6 et le calcul des densités par la méthode du noyau Gaussien (équation 2.2), tel que suggéré pour un ensemble de petite taille. Par la suite, nous calculons les mesures de la distance selon l'équation 2.3 et les σ_i selon l'équation 2.4. Après avoir calculé ces mesures, nous traçons les graphes de décision avec une adjacence d'ordre 1 et 2 pour comparer les résultats obtenus et en plus, le graphe représentant les valeurs de gamma afin de rechercher le nombre de classes.

La figure 2.11a montre un exemple du graphe de décision où les centres des classes sont identifiés par les points ayant des valeurs élevées à la fois de ρ_i et δ_i. Le seuil est calculé selon l'équation 2.11. Le plan est divisé en deux zones de décision et identifie les 7 classes qui sont situées au-dessus de la valeur de seuil. Les centres de classes détectés sont représentés par des points colorés. Ensuite sur la figure 2.11b nous traçons les valeurs de γ triées par ordre décroissant. Les centres de classes sont détectés par ceux ayant des valeurs élevés de γ. On peut observer l'impact du choix du voisinage sur le graphe de décision. Sur la figure 2.11a on remarque qu'il existe des centres de classes avec des valeurs de δ et de ρ supérieurs aux autres, comme le cas de la classe marron. En plus, on remarque qu'il existe un point à proximité du centre de la classe marron ayant des valeurs de δ et de ρ élevées qui n'est pas considéré un centre de classe, or sur la figure 2.11b ce point possède aussi une valeur de γ élevée. Par la suite, nous traçons les mêmes schémas avec un ordre d'adjacence égal à 2. Les figures 2.12a et 2.12b illustre les résultats obtenus. On remarque alors que le groupe de points représentant les centres de densité sont bien éloignés des autres points. Expérimentalement, nous remarquons que le modèle n'est plus fiable pour une adjacence à l'ordre 1. Par conséquent, nous choisissons la matrice d'adjacence d'ordre 2.

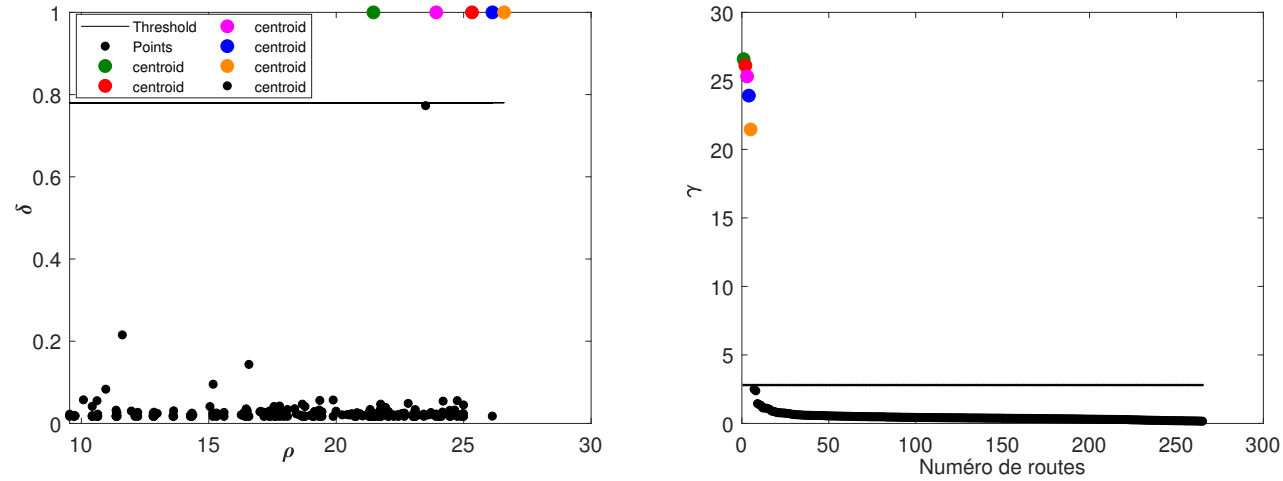
Sélection des zones de congestion

La figure 2.13 montre le résultat de partitionnement obtenu avec une adjacence d'ordre 2. De plus, le tableau 2.4 montre les résultats des vitesses moyennes et écarts-types des classes résultantes. L'algorithme DPC permet de détecter les différentes zones de congestion du réseau routier. Chaque zone correspond à la description du trafic du réel (voir figure 1.10). Les 2 quartiers représentant des zones fluides (classes rouge et orange) sont isolés avec des vitesses moyennes de 11.11 et 11.83 m/s. De plus, les zones à fortes congestions sont regroupées dans 3 classes (classes vert, magenta et bleu) identifiant les axes principales de congestion comme le rue de verdun et les principales intersections les plus congestionnées.



(a) (b)

FIGURE 2.11 – (a) Graphe de décision avec une adjacence d'ordre 1 et (b) Valeurs de γ triées par ordre décroissant.



(a) (b)

FIGURE 2.12 – (a) Graphe de décision avec une adjacence d'ordre 2 et (b) Valeurs de γ triées par ordre décroissant.

TABLEAU 2.4 – Valeurs de la vitesse moyenne (m/s) et écart-type.

(μ/σ)	Vert	Magenta	Bleu	Rouge	Orange
IND	6.53/ 1.70	7.94/2.30	7.94/2.30	11.11/ 3.12	11.83/ 1.50

FIGURE 2.13 – Partitionnement du réseau par DPC avec une adjacence d'ordre 2.

2.9 Conclusion

L'objectif de ce chapitre est de mettre en œuvre l'algorithme Density Peak Clustering et de montrer son efficacité dans le partitionnement d'un réseau urbain. Cet algorithme est basé sur le calcul des densités locaux et des distances entre les points de données. Il permet de détecter des classes de formes quelconques non-sphériques et des classes de densités différentes. Il est robuste quant au changement de métrique employée. Les classes sont obtenues directement à partir des données sans transformation. Le nombre des classes est obtenu automatiquement par des représentations graphiques du graphe du décision et la règle gamma.

Cet algorithme offre des résultats intéressants dans le contexte d'un réseau de transport. Cependant, on ne peut pas obtenir de résultats cohérents sans prendre en compte l'adjacence des routes. Pour un réseau de grande taille, il faut considérer un ordre de la matrice d'adjacence supérieur à 2 pour préserver la connexité des classes. Dans le chapitre suivant, nous présentons une autre méthode de classification automatique basée sur le concept des graphes. Cette méthode comprend plusieurs algorithmes de partitionnement spectral qui peuvent être appliqués à un réseau urbain.

Chapitre 3

Classification spectrale

Sommaire

3.1 Introduction

À part la méthode de classification introduite dans le chapitre précédent, il existe une autre méthode de classification qui s'appuie sur le concept de graphe et la théorie des matrices. Plus précisément, la méthode de la classification spectrale qui est basée sur la représentation des données sous forme de graphe et la recherche des vecteurs propres de la matrice Laplacienne du graphe qui permet d'offrir un espace de représentation plus simple pour la classification des données. Les avantages de cette approche sont ses bases théoriques solides, ses performances à classer des données de structures diverses et ses applications récentes avec succès dans les domaines tels que les réseaux de transport.

La représentation des données sous forme de graphe et la possibilité d'intégrer les connaissances contextuelles dans celui-ci, conduisent à considérer le problème de la classification comme un problème de coupe de graphe, sans présumer d'hypothèses sur la forme des objets. Les techniques de classification spectrale sont basées sur la minimisation d'un critère de type coupe de graphe. De plus, la classification spectrale nécessite la connaissance a priori du nombre des classes. Ce nombre peut être fixé par l'utilisateur, ou bien recherché automatiquement.

Ce chapitre s'intéresse à la méthode de classification basée sur les graphes, la classification spectrale et de sa mise en oeuvre. En premier lieu, nous donnons, des détails sur les méthodes de calcul de la matrice de similarité et la matrice Laplacienne dans la section 3.2. Ensuite, nous présentons dans la section 3.3 les différents algorithmes de la classification spectrale. Dans la sections 3.5, nous appliquons les méthodes de classification décrites sur les données du réseau routier de la ville de Calais afin de mettre en évidence les caractéristiques des méthodes présentées. Les données acquises au niveau de la simulation microscopique seront utilisées comme bases de données.

3.2 Graphe des données

Les méthodes de classification spectrale sont des méthodes de classification en K groupes (appelés classes) qui utilisent un algorithme de classification non supervisée, basée sur le spectre de la matrice de similarités. L'objectif de la classification non supervisée est de diviser un graphe en groupes naturels tels que les nœuds d'un même groupe sont similaires alors que les nœuds des groupes différents sont dissimilaires.

3.2.1 Représentation de graphe des données

Un graphe peut être utilisé comme un modèle de représentation de données permettant d'exprimer les relations et de révéler les dépendances entre ces données. Une étape fondamentale dans les méthodes basées sur les graphes est la construction des graphes. À partir d'un ensemble de N points de données $X = \{x_1, x_2, \ldots, x_N\}$, il est facile de construire un graphe $G = (\mathcal{V}, E)$ en associant à chaque point de l'ensemble de données x_i un nœud $v_i \in \mathcal{V}$ et on pondère chaque arête $e_i \in E$ reliant deux nœuds v_i et v_j par une similarité w_{ij}.

3.2.2 Matrices de distances et de similarités

Afin de fournir une première analyse des données pour identifier des groupes d'objets naturels, il est souvent utile d'utiliser des représentations géométriques.

3.2.3 Matrice de distance

L'espace des données est un espace à M dimensions dans lequel chaque objet est représenté par un point. Une mesure de distance des paires de points (x_i, x_j) notée δ_{ij}, doit être positive et symétrique ($\delta_{ij} \geq 0$ et $\delta_{ij} = \delta_{ji}$). Elle atteint sa valeur minimale $\delta_{ij} = 0$ dans le cas où $x_i = x_j$. Cette mesure doit satisfaire également la contrainte d'inégalité triangulaire ($\forall x_i, x_j, x_l, \delta_{ij} \leq \delta_{il} + \delta_{lj}$) (Losee, 2012). Les points proches dans l'espace sont suspectibles de représenter des données appartenant à un même groupe où la valeur de mesure de la distance est proche de 0, tandis que les points éloignés appartiennent à des groupes différents où la valeur de mesure de la distance est grande (la distance tend vers l'infini).
A partir de l'ensemble de données X, nous construisons la matrice de distance $\mathbf{\Delta}$ de dimension $(N \times N)$ de terme général δ_{ij}, qui représente la distance séparant les paires de points (x_i, x_j) :

$$\mathbf{\Delta} = \begin{pmatrix} 0 & \ldots & \delta_{1i} & \ldots & \delta_{1N} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \delta_{i1} & \ldots & 0 & \cdots & \delta_{iN} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \delta_{N1} & \ldots & \delta_{Ni} & \cdots & 0 \end{pmatrix}. \tag{3.1}$$

Des algorithmes de classification comme les algorithmes de classification hiérarchique et K-means utilisent le concept de distance pour obtenir les classes de points. En fait, il semble cohérent de regrouper les points dont la distance les uns des autres est faible et de séparer les points les plus éloignés.

3.2.4 Matrices de similarités

Dans la représentation des données par graphe, la mesure de similarité notée par w_{ij} entre paires de nœuds (v_i, v_j) exprime le degré de ressemblance entre les couples de données associés (x_i, x_j). La similarité entre deux nœuds est fonction de la distance des points associés. Une fonction de similarité doit respecter la contrainte de normalisation ($w_{ij} \in [0,1]$), de symétrie ($w_{ij} = w_{ji}$) et vérifie que $w_{ii} \geq w_{ij}$.

Dans la littérature, plusieurs définitions des fonctions de similarité ont été proposées. Parmi elles, nous choisissons de présenter les fonctions suivantes :

— **Similarité inverse de distance euclidienne** : Cette fonction de similarité permet d'obtenir un maximum global borné à 1 pour une distance égale à zéro. Elle est définie comme suit :

$$w_{ij} = \frac{1}{\frac{\|x_i - x_j\|^2}{\sigma^2} + 1} \tag{3.2}$$

Avec $\|x_i - x_j\|$ la distance euclidienne entre le couple de points (x_i, x_j). Le paramètre de dispersion σ permet de tenir compte de la dispersion locale des données (Kunegis et al., 2008).

— **Similarité cosinus** : La fonction cosinus permet de mesurer la similarité entre deux vecteurs (x_i, x_j), en calculant l'angle entre eux. Elle est définie comme suit (Salton and Buckley, 1988) :

$$w_{ij} = |cos(x_i, x_j)| = \frac{|{x_i}^T x_j|}{\|x_i\| \cdot \|x_j\|} \tag{3.3}$$

Plus l'angle entre deux vecteurs est faible, plus les points associés sont similaires. Par conséquent, cette fonction de similarité est sensible à la direction des données projetées dans l'espace d'attributs.

— **Similarité gaussienne** : Cette fonction de similarité prend en compte les relations de voisinage entre les points de données. Par conséquent, une fonction gaussienne basée sur la distance euclidienne entre les points est souvent utilisée. Elle est définie par (Von Luxburg, 2007) :

$$w_{ij} = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{3.4}$$

où, $\|x_i - x_j\|$ est la distance euclidienne entre les points de données (x_i, x_j), et σ est un paramètre de dispersion, choisi par l'utilisateur de telle sorte qu'il soit adapté à la dispersion locale des données. La fonction de similarité gaussienne diminue de manière exponentielle lorsque les distances sont plus importantes, ce qui permet d'obtenir un

poids plus proche de 1 pour les points très similaires (Kunegis et al., 2008). Cette fonction est couramment utilisée dans le domaine de classification (Von Luxburg, 2007).

Le choix du paramètre de dispersion σ est crucial pour une bonne représentation des données. L'ajustement du paramètre σ est souvent négligé lors de la description des algorithmes de classification spectrale. Cependant, des valeurs de σ différentes peuvent conduire à des résultats de partitionnement très différents. Nous présentons trois méthodes de choix de ce paramètre :

— *Auto-adaptif (ou self-tuning)* : Cette méthode consiste à adapter le paramètre de dispersion à la structure locale de chaque couple de données (x_i, x_j) (Zelnik-Manor and Perona, 2005) :

$$\sigma_{ij}^2 = \sigma_i \sigma_j$$

Il s'agit d'approximer la dispersion au voisinage d'un point x_i par la distance de ce point à son r plus proche voisin x_{r_i} :

$$\sigma_i = \|x_i - x_{r_i}\|$$

et d'approximer la dispersion au voisinage d'un point x_j par la distance de ce point à son r plus proche voisin x_{r_j} :

$$\sigma_j = \|x_j - x_{r_j}\|$$

La similarité gaussienne est alors exprimée par :

$$w_{ij} = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i \sigma_j}\right) \tag{3.5}$$

— *Méthode basée sur un seuil de similarité* : Le paramètre de dispersion est déduit à partir d'un seuil de similarité minimale ϵ de telle sorte que la similarité entre chaque point et son r plus proche voisin est $\geq \epsilon$. Le choix de σ est déduit de l'équation (Xu et al., 2005) :

$$\epsilon = e^{-\frac{\delta^2}{2\sigma^2}} \tag{3.6}$$

avec, $\delta = max_i \left\{\delta_i^r\right\}$, où δ_i^r est la distance euclidienne du point x_i à son r plus proche voisin.

— *Méthode basée sur la moyenne des distances* : Cette méthode consiste à calculer les distances $\|x_i - x_j\|$ entre les couples des points (x_i, x_j). Le paramètre σ^2 est choisi égal à la moyenne des distances des paires des points (Zhao et al., 2018) :

$$\sigma^2 = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \|x_i - x_j\|^2 \tag{3.7}$$

3.2.5 Matrices Laplaciennes

Le principal outil d'analyse spectrale des graphes associés aux données est la matrice Laplacienne. À partir de la matrice de similarité $\boldsymbol{W}$ d'un graphe et la matrice de degré $\boldsymbol{D}$ associée au graphe, plusieurs matrices Laplaciennes peuvent être définies.

— La matrice Laplacienne non normalisée est donnée par :

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} = \begin{pmatrix} d_1 & -w_{12} & \dots & -w_{1N} \\ -w_{21} & d_2 & \cdots & -w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -w_{N1} & -w_{N2} & \cdots & d_N \end{pmatrix}. \tag{3.8}$$

La matrice Laplacienne $\boldsymbol{L}$ est toujours positive semi-définie et symétrique. La matrice Laplacienne non normalisée ne dépend pas des éléments de la diagonale de la matrice de similarité $\boldsymbol{W}$ (Ng et al., 2002; Von Luxburg, 2007). Les valeurs sur la diagonale appelées auto-similarité, définies la similarité d'un noeuds avec lui-même. En effet, ces valeurs ne modifient pas la matrice Laplacienne $\boldsymbol{L}$. Par conséquent, toute autre matrice de similarités composée de valeurs hors diagonale, égales à celles de $\boldsymbol{W}$, donne la même solution que la matrice $\boldsymbol{L}$.

Dans la littérature, il existe d'autres formes de la matrice Laplacienne, qui sont de la forme normalisée (Meila and Shi, 2001; Von Luxburg, 2007). Ces matrices Laplaciennes sont définies de deux manières différentes :

— La matrice Laplacienne normalisée symétrique, notée par $\boldsymbol{L_s}$, d'un graphe est définie par (Shi and Malik, 2000) :

$$\boldsymbol{L_s} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}, \tag{3.9}$$

avec, $\boldsymbol{D}^{1/2}$ est la matrice diagonale définie par $\boldsymbol{D}(i,i)^{1/2} = \sqrt{d_i}$. Alors, $\boldsymbol{D}^{-1/2}$ est la matrice diagonale définie par $\boldsymbol{D}(i,i)^{-1/2} = \frac{1}{\sqrt{d_i}}$, sachant que $d_i \neq 0$. Alors, la matrice

Laplacienne symétrique normalisée s'écrit de la forme suivante :

$$\boldsymbol{L_s} = \begin{pmatrix} 1 & \frac{-w_{12}}{\sqrt{d_1 d_2}} & \cdots & \frac{-w_{1N}}{\sqrt{d_1 d_N}} \\ \frac{-w_{21}}{\sqrt{d_2 d_1}} & 1 & \cdots & \frac{-w_{2N}}{\sqrt{d_2 d_N}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{-w_{N1}}{\sqrt{d_N d_1}} & \frac{-w_{N2}}{\sqrt{d_N d_2}} & \cdots & 1 \end{pmatrix}. \tag{3.10}$$

— La matrice Laplacienne normalisée asymétrique, notée par $\boldsymbol{L_a}$, d'un graphe est définie par (Meila and Shi, 2001) :

$$\boldsymbol{L_a} = \boldsymbol{L}\boldsymbol{D}^{-1} = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{W}, \tag{3.11}$$

avec, $\boldsymbol{D}^{-1}$ est la matrice diagonale définie par $\boldsymbol{D}(i,i)^{-1} = \frac{1}{d_i}$ sachant que $d_i \neq 0$. Alors, la matrice Laplacienne normalisée asymétrique s'écrit de la forme suivante :

$$\boldsymbol{L_a} = \begin{pmatrix} 1 & \frac{-w_{12}}{d_1} & \cdots & \frac{-w_{1N}}{d_1} \\ \frac{-w_{21}}{d_2} & 1 & \cdots & \frac{-w_{2N}}{d_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{-w_{N1}}{d_N} & \frac{-w_{N2}}{d_N} & \cdots & 1 \end{pmatrix}. \tag{3.12}$$

3.3 Classification spectrale

La classification spectrale consiste à partitionner un graphe en groupes (ou classes) non vides et disjoints deux à deux. Soit $C = \{C_1, C_2, \ldots, C_K\}$ l'ensemble de K classes. Les éléments dans une même classe C_k se ressemblent. Les classes doivent respecter les contraintes suivantes :

— Une classe C_k contient au moins un élément : $card(C_k) = |C_k| \neq 0$.
— L'intersection entre deux classes différentes doit être un ensemble vide : $C_k \cap C_l = \varnothing$ pour tout $k \neq l$.
— L'union de toutes les classes constitue le graphe : $G = \cup_{k=1}^{K} C_k$.

On considère un graphe $G = (\mathcal{V}, E)$ composé de N nœuds interconnectés $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ représentant un ensemble de données $X = \{x_1, x_2, \ldots, x_N\}$. On se propose de partitionner les nœuds $\mathcal{V}$ de cardinal $|\mathcal{V}| = N$ du graphe G en K classes $\{C_1, C_2, \ldots, C_K\}$. La classifiaction spectrale repose sur la minimisation d'un critère de type coupe : coupe simple à $K = 2$ (Shi and Malik, 2000), ou de type coupe multiple à $K > 2$ (Meila and Shi, 2001; Ng et al., 2002). Nous introduisons par la suite ces deux critères de coupe simple et multiple.

3.3.1 Coupe simple K=2

Dans la théorie des graphes, il existe différentes fonctions objectifs : la coupe minimale (MinCut), la coupe ratio (RatioCut), la coupe normalisée (NC), la coupe min-max (MinMaxCut), etc. Le but est de partitionner le graphe $G = (\mathcal{V}, E)$ en deux sous-ensembles disjoints de nœuds, C_1 et C_2, grâce à l'optimisation d'une de ces fonctions. Pour les sous-ensembles de nœuds C_1 et C_2 du graphe G, les deux termes suivants peuvent être définis :

— **La coupe inter-groupes** : est calculée en sommant les poids sur les arêtes entre les deux sous-ensembles C_1 et C_2, ou autrement dit en sommant les similarités inter-groupes, elle est définie par :

$$Cut(C_1, C_2) = \sum_{v_i \in C_1} \sum_{v_j \in C_2} w_{ij} \tag{3.13}$$

— **La coupe intra-groupe** : est calculée en sommant les poids des arêtes entre les nœuds du sous-ensemble C_1 (ou C_2), ou autrement dit en sommant les similarités intra-groupes, elle est définie par :

$$\begin{aligned} Cut(C_1, C_1) &= \sum_{v_i \in C_1} \sum_{v_j \in C_1} w_{ij} \\ Cut(C_2, C_2) &= \sum_{v_i \in C_2} \sum_{v_j \in C_2} w_{ij} \end{aligned} \tag{3.14}$$

La coupe minimale

L'objectif de la méthode de la coupe minimale est de trouver deux sous-graphes et, par conséquent, deux sous-ensembles C_1 et C_2 de nœuds pour lesquels la somme des poids des arêtes entre ces sous-ensembles est minimale. Le critère d'optimisation est défini par :

$$MinCut(C_1, C_2) = Cut(C_1, C_2) \tag{3.15}$$

L'inconvénient principal de cette méthode est le partitionnement du graphe en sous-ensembles déséquilibrés. Cela s'explique notamment par le fait que les sous-ensembles ou les nœuds isolés ont un degré faible. On reprend l'exemple du chapitre précédent de la figure 1.1 et l'équation 1.3. La figure 3.1 illustre un exemple du partitionnement des données en deux classes. La coupe obtenue est représentée par une ligne pointillée qui représente une valeur minimale égale à 1.06, mais les groupes de nœuds obtenues sont déséquilibrés. La minimisation de ce type de coupe est donc très sensible aux nœuds isolés. Une solution consiste à normaliser la valeur de la coupe en fonction du nombre des éléments dans le sous-ensemble (Chen et al., 2005).

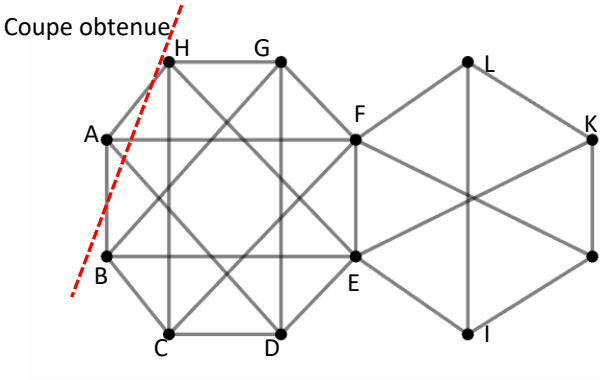


FIGURE 3.1 – exemple illustratif du critère de la coupe minimale.

La coupe normalisée

L'objectif du critère de la coupe normalisée est de chercher une partition des nœuds telle que les connexions intra-groupes sont maximisées et les connexions inter-groupes sont minimisées. Ce qui revient à maximiser les termes $Cut(C_1,C_1)$ et $Cut(C_2,C_2)$ et à minimiser le terme $Cut(C_1,C_2)$. Dans ce type de coupe, la notion de volume d'un sous-ensemble est introduite (Shi and Malik, 2000). Le critère de coupe normalisée est défini comme suit :

$$\begin{aligned} NC(C_1,C_2) &= \frac{Cut(C_1,C_2)}{vol(C_1)} + \frac{Cut(C_1,C_2)}{vol(C_2)} \\ &= Cut(C_1,C_2) \times \left(\frac{1}{Cut(C_1,\mathcal{V})} + \frac{1}{Cut(C_2,\mathcal{V})} \right) \end{aligned} \tag{3.16}$$

Le volume d'un sous-ensemble $vol(C_1)$ (ou $vol(C_2)$) mesure la taille de C_1 en additionnant les poids de tous les arêtes attachées aux noeuds associés au sous-ensemble C_1. Il est donné par :

$$\begin{aligned} vol(C_1) &= \sum_{v_i \in C_1} d_i = \sum_{v_i \in C_1} \sum_{v_j \in \mathcal{V}} w_{ij} = Cut(C_1,\mathcal{V}) \\ vol(C_2) &= Cut(C_2,\mathcal{V}) \end{aligned} \tag{3.17}$$

Il est possible de normaliser ce critère de coupe en multipliant par $\frac{1}{2}$, afin d'obtenir des valeurs comprises dans l'intervalle $[0,1]$. Nous reprenons l'exemple pédagogique du graphe de la figure 1.1. Les deux classes correspondantes aux critères de la coupe normalisée sont : $C_1 = \{B, C, D, G, H\}$ et $C_2 = \{A, E, F, I, J, K, L\}$. Les volumes des sous-ensemble C_1 et C_2 sont $vol(C_1) = 9.61$ et $vol(C_2) = 14.78$ avec une valeur pour la coupe normalisée $NC(C_1,C_2) = 0.23$.

3.3.2 La coupe multiple K>2

Lorsque le nombre de classes est $K > 2$, il est possible d'utiliser deux approches. Une approche récursive, dont le principe est d'appliquer récursivement un algorithme de bi-partitionnement d'une manière hiérarchique. Après avoir divisé le graphe en deux sous-ensembles de nœuds, il est nécessaire de réappliquer le même processus sur ces deux sous-ensembles. On suppose donc que le nombre de groupes à trouver est connu ou directement contrôlé par un seuil (fixé par l'utilisateur) associé à la valeur de la fonction objectif. Une seconde approche est une généralisation des fonctions objectifs définies pour $K = 2$, afin de prendre en compte un nombre de classes K plus élevé. Il faut connaître "a priori" le nombre de ces classes. Les groupes recherchés $\{C_1, C_2, \ldots, C_K\}$ sont définis de la même façon que pour le cas $K = 2$.
Soit C_k le k-ème groupe de nœuds et $\overline{C}_k$ le complément du groupe C_k. En étendant le standard prédéfini à $K = 2$, il est possible d'obtenir les fonctions objectifs suivantes :

— La coupe minimale est définie par :

$$MinCut(C_1, C_2, \ldots, C_K) = \sum_{k=1}^{K} Cut(C_k, \overline{C}_k) \tag{3.18}$$

— La coupe normalisée est définie par :

$$NC(C_1, C_2, \ldots, C_K) = \sum_{k=1}^{K} \frac{Cut(C_k, \overline{C}_k)}{vol(C_k)} \tag{3.19}$$

avec, $vol(C_k) = Cut(C_k, V)$.

Il est possible de normaliser les critères de coupe en multipliant par $\frac{1}{K}$, pour obtenir des valeurs comprises dans l'intervalle $[0,1]$.

3.3.3 Minimisation du critère de la coupe normalisée

Il existe une relation entre le critère de la coupe normalisée et la matrice Laplacienne du graphe. Considérons le vecteur $u = (u_1, u_2, \ldots, u_N)^T$ indicateur d'appartenance des noeuds aux classes C_1 et C_2 avec :

$$u_i = \begin{cases} \frac{1}{vol(C_1)} & si\, v_i \in C_1 \\ -\frac{1}{vol(C_2)} & si\, v_i \in C_2 \end{cases} \tag{3.20}$$

La relation reliant le critère de coupe de graphe et le Laplacien est donnée par :

$$\frac{u^T \boldsymbol{L} u}{u^T \boldsymbol{D} u} = \frac{1}{2} Cut(C_1, C_2) \left(\frac{1}{vol(C_1)} + \frac{1}{vol(C_2)} \right) \tag{3.21}$$

En effet : $u^T \boldsymbol{D} u = \frac{1}{vol(C_1)} + \frac{1}{vol(C_2)}$

D'après von Luxburg (Von Luxburg, 2007), $u^T \boldsymbol{L} u$ peut être définie par :

$$\begin{aligned} u^T \boldsymbol{L} u &= u^T (\boldsymbol{D} - \boldsymbol{W}) u \\ &= u^T \boldsymbol{D} u - u^T \boldsymbol{W} u \\ &= \frac{1}{2} \left(\sum_i d_i u_i^2 - 2 \sum_{i,j} w_{ij} u_i u_j + \sum_i d_i u_i^2 \right) \\ &= \frac{1}{2} \sum_{i,j} w_{ij} (u_i - u_j)^2 \\ &= \frac{1}{2} \sum_{i \in C_1, j \in C_2} w_{ij} \left(\frac{1}{vol(C_1)} + \frac{1}{vol(C_2)} \right)^2 \\ &= \frac{1}{2} Cut(C_1, C_2) \left(\frac{1}{vol(C_1)} + \frac{1}{vol(C_2)} \right)^2 \end{aligned} \tag{3.22}$$

Le critère de la coupe normalisée NC peut alors être écrit de la forme suivante :

$$\frac{u^T \boldsymbol{L} u}{u^T \boldsymbol{D} u} = \frac{1}{2} NC(C_1, C_2)$$

et la minimisation du critère de la coupe normalisée NC est donnée par :

$$\min_{C_1, C_2} NC(C1, C_2) = \min_u \frac{u^T \boldsymbol{L} u}{u^T \boldsymbol{D} u}$$

La recherche de deux groupes C_1 et C_2 qui minimisent NC est résout en relaxant les contraintes sur le vecteur indicateur u sous la forme :

$$\min_u \frac{u^T \boldsymbol{L} u}{u^T \boldsymbol{D} u}$$

La solution triviale est le vecteur $\mathbf{1}$ dont les éléments sont des 1 et correspond à la valeur propre nulle. Les vecteurs indicateurs u doivent vérifier la contrainte $u^T \boldsymbol{D} \mathbf{1} = 0$. Ce problème est équivalent à celui obtenu en normalisant le dénominateur :

$$\min u^T \boldsymbol{L} u, \tag{3.23}$$

sous la contrainte $u^T \boldsymbol{D} u = 1$.
La minimisation est obtenue en résolvant le système de valeurs propres généralisées :

$$\boldsymbol{L} u = \lambda \boldsymbol{D} u \tag{3.24}$$

Pour affecter les points aux classes C_1 et C_2, il suffit de seuiller le vecteur indicateur u par rapport à zéro de la manière suivante : $v_i \in C_1$ si $u_i \geq 0$ et $v_i \in C_2$ si $u_i < 0$. En gros, partitionner un graphe en K classes, $C_1, \ldots, C_K$ revient donc, à minimiser le critère de la coupe normalisée donné par :

$$NC(C_1, \ldots, C_K) = \sum_{k=1}^{K} \frac{Cut(C_k, \overline{C}_k)}{vol(C_k)}$$

où, $\overline{C}_k$ est l'ensemble de nœuds complément de C_k dans $\mathcal{V}$. La solution est un problème d'optimisation de la forme :

$$\min_{U} Tr(U^T \boldsymbol{L} U) \tag{3.25}$$

avec $U^T \boldsymbol{D} U = I$ et Tr est la trace d'une matrice. Ce qui conduit à rechercher les K premiers vecteurs propres associés aux K plus petites valeurs propres non nulles. Par la suite les points des données sont affectés à leurs classes correspondantes par un algorithme de partitionnement comme les K-means.

D'une façon similaire, le problème d'optimisation de la coupe normalisée a été résolu par une maximisation de la trace, où la solution est la matrice U dont les colonnes sont les K vecteurs propres associés aux K premières valeurs propres de la matrice Laplacienne définie par Ng et al. (2002) :

$$\boldsymbol{L} = \boldsymbol{D}^{-1/2} \boldsymbol{W} \boldsymbol{D}^{-1/2}. \tag{3.26}$$

On peut donc écrire la coupe normalisée de manière équivalente par l'équation suivante (Bach and Jordan, 2006) :

$$NC = K - Tr\left[U^T \left(\boldsymbol{D}^{-1/2} \boldsymbol{W} \boldsymbol{D}^{-1/2}\right) U\right]. \tag{3.27}$$

Le tableau 3.1 illustre quelques exemples de fonctions objectives utilisées comme critères de coupes pour la partition de graphes.

TABLEAU 3.1 – Tableau récapitulatif de critères de coupes de graphe.

Fonction	Représentation graphe	Représentation compacte
Degré d'un noeud	$d_i = \sum_{j=1}^{N} w_{ij}$	$vol(v_i)$
Volume d'une classe C_k	$vol(C_k) = \sum_{v_i \in C_k} d_i = \sum_{v_i \in C_k} \sum_{v_j \in \mathcal{V}} w_{ij} = Cut(C_k, \mathcal{V})$	$vol(C_k) = u_k^T \boldsymbol{D} u_k$
Volume du graphe	$vol(\mathcal{V}) = Cut(\mathcal{V}, \mathcal{V}) = \sum_{i=1}^{N} d_i$	$vol(\mathcal{V}) = tr(\boldsymbol{D})$
Somme poids intra-classe C_k	$Cut(C_k, C_k) = \sum_{v_i \in C_k} \sum_{v_j \in C_k} w_{ij}$	$Cut(C_k, C_k) = u_k^T \boldsymbol{W} u_k$
Somme poids inter-classe $(C_k, \overline{C_k})$	$Cut(C_k, \overline{C_k}) = \sum_{v_i \in C_k} \sum_{v_j \in \overline{C_k}} w_{ij}$	$Cut(C_k, \overline{C_k}) = u_k^T \boldsymbol{L} u_k$
Coupe minimale	$MinCut = \sum_{k=1}^{K} Cut(C_k, \overline{C_k})$	$MinCut = \sum_{k=1}^{K} u_k^T \boldsymbol{L} u_k$
Coupe ratio	$RCut = \sum_{k=1}^{K} \frac{Cut(C_k, \overline{C_k})}{\lvert C_k \rvert}$	$RCut = \sum_{k=1}^{K} \frac{u_k^T \boldsymbol{L} u_k}{u_k^T u_k}$
Coupe normalisée	$NC = \sum_{k=1}^{K} \frac{Cut(C_k, \overline{C_k})}{vol(C_k)}$	$NC = \sum_{k=1}^{K} \frac{u_k^T \boldsymbol{L} u_k}{u_k^T \boldsymbol{D} u_k}$
Poids moyenne	$AW = \sum_{k=1}^{K} \frac{Cut(C_k, C_k)}{\lvert C_k \rvert}$	$AW = \sum_{k=1}^{K} \frac{u_k^T \boldsymbol{W} u_k}{u_k^T u_k}$
Modularité *mod*	$\sum_{k=1}^{K} \frac{1}{\lvert C_k \rvert} \left(\frac{Cut(C_k, C_k)}{Cut(\mathcal{V}, \mathcal{V})} - \left(\frac{Cut(C_k, \mathcal{V})}{Cut(\mathcal{V}, \mathcal{V})} \right)^2 \right)$	$mod = \sum_{k=1}^{K} u_k^T \boldsymbol{Q} u_k$ $\mathbf{Q} = \frac{1}{tr(\boldsymbol{D})} \left(\boldsymbol{W} - \frac{dd^T}{tr(\boldsymbol{D})} \right)$
Modularité normalisée *nmod*	$\sum_{k=1}^{K} \frac{1}{Cut(C_k, \mathcal{V})} \left(\frac{Cut(C_k, C_k)}{Cut(\mathcal{V}, \mathcal{V})} - \left(\frac{Cut(C_k, \mathcal{V})}{Cut(\mathcal{V}, \mathcal{V})} \right)^2 \right)$	$nmod = \frac{1}{Cut(\mathcal{V}, \mathcal{V})} (k - 1 - NC)$

3.4 Algorithmes de classification spectrale

Les algorithmes de classification spectrale sont des méthodes de résolution du problème de la coupe du graphe. Ils sont basés sur les trois étapes suivantes :

1. Pré-traitement des données : la construction du graphe et sa matrice de similarité $\boldsymbol{W}$.
2. La représentation spectrale : la construction de la matrice Laplacienne, le calcul des valeurs et vecteurs propres de la matrice et la projection des points dans l'espace spectral basé sur les vecteurs propres retenus.
3. Partitionnement : la recherche des classes dans l'espace spectral par application d'un algorithme de partitionnement comme K-means sur les points de données, et l'affectation des points dans l'espace d'origine aux classes.

Une première approche consiste à appliquer l'algorithme de K-means sur la matrice la-

placienne du graphe, avec l'algorithme 3.

Algorithme 3 : Algorithme de classification spectrale non normalisé

Résultat : Vecteur indiquant la classe de chaque nœud
Entrée : graphe G à N nœuds, nombre de classes K;
Calculer la matrice Laplacienne $\boldsymbol{L}$;
Calculer les K premiers vecteurs propres de $\boldsymbol{L}$;
Choisir les K premiers vecteurs propres correspondant aux K plus petites valeurs propres. ;
Utiliser K-means sur les N lignes de U;
La classe de chaque ligne correspond à la classe de chaque nœud;

Nous représentons par la suite les deux algorithmes de classification spectrale normalisée.

Algorithme de Von Luxburg.

Von Luxburg (Von Luxburg, 2007) généralise le critère coupe normalisé NC au critère de la coupe multiple, comme montré dans l'équation 3.19. Cette méthode est une méthode directe de K-partition. Ce problème est résolu, en définissant K vecteurs indicateurs u_k tel que $u_k = (u_{1k}, \ldots, u_{Nk})$ avec $u_{ik} = \frac{1}{\sqrt{vol(C_k)}}$. Ces vecteurs indicateurs sont rangés en colonne dans la matrice U. Le problème est exprimé sous la forme de la généralisation suivante :

$$\min_{Z} NC(C_1, \ldots, C_K) = \min_{Z} \sum_{k=1}^{K} z_k^T \boldsymbol{L_s} z_k \quad s.c. \quad z_k^T z_k = 1, \tag{3.28}$$

avec une condition supplémentaire : $U = \boldsymbol{D}^{-1/2} Z$ et $U^T \boldsymbol{D} U = I$.

Par conséquent, les K premiers vecteurs propres correspondant aux K plus petites valeurs propres de la matrice Laplacienne minimisent ce critère. Ils permettent aussi d'estimer les K vecteurs indicateurs. Le but est de trouver des valeurs discrètes pour ces vecteurs indicateurs, pour cela, l'extraction du spectre est suivie d'une étape d'application d'un algorithme de classification non supervisée comme le K-means sur les lignes de $U = \boldsymbol{D}^{-1/2} Z$. L'algorithme de Von Luxburg est résumé par l'algorithme 4.

Algorithme de Ng et al.

Ng et al. (2002) proposent un algorithme similaire à celui de Meila et Shi (Meila and Shi, 2001) qui résout le problème de l'équation 3.28. Ils proposent d'utiliser les K plus petits vec-

Algorithme 4 : Algorithme de Von Luxburg

Résultat : Vecteur indiquant la classe de chaque noeud
Entrée : graphe G à N noeuds, nombre de classes K;
Calculer la matrice Laplacienne normalisée symétrique $\boldsymbol{L_s}$;
Calculer les K plus petits vecteurs propres de $\boldsymbol{L_s}$;
Regrouper les vecteurs propres dans une matrice Z;
Utiliser K-means sur les N lignes de $U = D^{\frac{-1}{2}} Z$;
La classe de chaque ligne correspond à la classe de chaque noeud;

teurs propres correspondant aux K plus petites valeurs propres de la matrice Laplacienne normalisée symétrique $\boldsymbol{L_s}$ (Eq. 3.9), afin de projeter les données. Ces vecteurs propres sont les K plus grands vecteurs propres z_k de la matrice Laplacienne proposée par les auteurs $\boldsymbol{L_{Ng}} = \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$. Notons que la matrice $\boldsymbol{L_s} = \boldsymbol{I} - \boldsymbol{L_{Ng}}$, avec $\boldsymbol{I}$ la matrice identité. Ensuite, au lieu de calculer la matrice $U = \boldsymbol{D}^{-1/2}Z$ à partir de la matrice Z contenant les K vecteurs propres, ils projettent les points dans l'espace spectral, en normalisant les lignes de Z et ils forment une nouvelle matrice U avec $U_{ij} = \frac{Z_{ij}}{\sqrt{\sum_j Z_{ij}^2}}$. Enfin, pour partitionner les noeuds, les auteurs utilisent l'algorithme de partitionnement K-means sur les lignes de U. L'algorithme de Ng et al. est résumé par l'algorithme 5.

Algorithme 5 : Algorithme de Ng et al.

Résultat : Vecteur indiquant la classe de chaque noeud
Entrée : graphe G à N noeuds, nombre de classes K;
Calculer la matrice Laplacienne $\boldsymbol{L_{Ng}} = \boldsymbol{D}^{\frac{-1}{2}}\boldsymbol{W}\boldsymbol{D}^{\frac{-1}{2}}$;
Calculer les K plus grands vecteurs propres de $\boldsymbol{L_N g}$;
Regrouper les vecteurs propres dans une matrice Z;
Normaliser les lignes de Z et construire U, avec $U_{ij} = \frac{z_{ij}}{\sqrt{\sum_j z_{ij}^2}}$;
Utiliser K-menas sur les N lignes de U;
La classe de chaque ligne correspond à la classe de chaque nœud;

Algorithme des K-means

Les algorithmes de classification spectrale utilisent l'algorithme de partitionnement K-means afin d'affecter les points à leurs classes correspondantes. Le K-means est un algorithme de classification automatique non supervisée qui résout le problème du partitionnement en calculant la distance entre les points de données et les centres des classes les plus proches. La distance de chaque point au centre de cette classe doit être la plus petite possible et les classes $C = \{C_1, \ldots, C_K\}$ obtenues doivent être fortement séparées. Notons que, le

nombre de classes K doit être connu. Les étapes de l'algorithme K-means sont les suivantes :

Algorithme 6 : Algorithme des K-means

Résultat : Vecteur indiquant la classe de chaque point
Entrée : un ensemble de points;
Fixer le nombre de classes K;
Générer K centres aléatoires;
tant que *au moins un centre change* **faire**
    Associer chaque point au centre le plus proche;
    Calculer le centre de chaque groupe de points créé;
fin

Complexité

La complexité du calcul de l'algorithme de classification spectrale est $O(N^3)$ où $N = |\mathcal{V}|$ est le nombre des nœuds. Ceci est essentiellement dû au calcul des valeurs propres et des vecteurs propres de la matrice laplacienne. L'application de l'algorithme K-means (ou K-moyennes) nécessite $O(dNK^2)$, avec d est le nombre d'itérations pour la convergence de l'algorithme K-means et K est le nombre de clusters finaux.

3.4.1 Estimation automatique du nombre des classes

Les algorithmes de classification spectrale nécessitent de fixer, en entrée, le nombre de classes K désirées. Une des plus importantes difficultés de ces méthodes est donc l'estimation de ce nombre. Généralement, ce paramètre est défini manuellement, mais il existe des méthodes automatiques pour estimer le nombre de classes.

Critère Eigengap

L'approche la plus simple pour estimer le nombre de classes souhaitées consiste à analyser les valeurs propres de la matrice Laplacienne. Ng et al. (2002) ont montré que, dans le cas idéal où les points appartenant à des classes différentes sont très distants les uns des autres, il est possible d'obtenir la matrice de similarités sous forme de matrice en blocs. La plus petite valeur propre de la matrice Laplacienne bloc-diagonale est nulle. Cette valeur se répète avec une multiplicité égale au nombre de classes. Cela signifie que le nombre de classes K est estimé, en comptant le nombre de valeurs propres nulles.

Toutefois, ce processus d'estimation n'est possible que dans la mesure où il s'agit d'un cas idéal, i.e. si les classes sont compactes et séparées les unes des autres. En effet, si des

classes se chevauchent, les K premières valeurs propres ne sont nécessairement plus nulles et le comptage devient moins pratique.

Pour résoudre cette problématique, une alternative est proposée par Shortreed and Meila (2005) et reprise par Von Luxburg (2007), qui consiste à organiser par ordre croissant les valeurs propres et à rechercher un "saut" (ou gap) important entre les valeurs propres successives. Le calcul du gap est défini par :

$$gap(i) = |\lambda_i - \lambda_{i+1}| \tag{3.29}$$

L'objectif est de choisir le nombre de classes tel que les amplitudes des K premières valeurs propres $\lambda_1, \ldots, \lambda_K$ soient très faibles et que celle de λ_{K+1} soit plus élevée :

$$K = arg\{max_i\{gap(i)\}\} \tag{3.30}$$

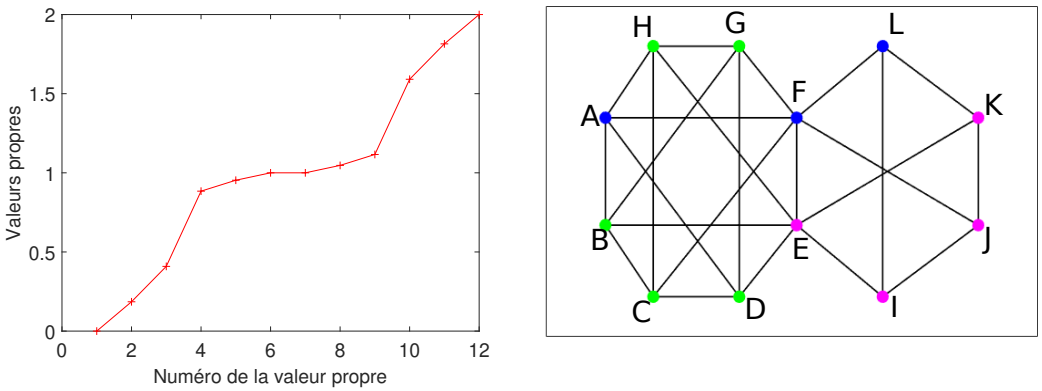
En effet, si la dispersion des valeurs propres ne présente pas de "saut", l'estimation du nombre de classes s'avère difficile et non pertinente.
La figure 3.2a illustre un exemple de la représentation des valeurs propres associées à la matrice Laplacienne normalisée symétrique du graphe de la figure 1.1. On remarque qu'entre la 3^{e} et la 4^{e} valeurs propres il y a un saut de gap remarquable. C'est-à-dire l'écart $|\lambda_3 - \lambda_4|$ est relativement grand. Cet écart indique que le nombre de classes K est égal à 3. Notons que, le choix $K = 9$ donne un résultat absurde, car plusieurs classes sont à 1 élément. La figure 3.3b montre les résultats de classification obtenus en appliquant le critère eigengap pour la détection du nombre de classes K.

Critère de modularité

White and Smyth (2005) proposent une méthode qui permet d'estimer automatiquement le nombre de classes. Cette méthode est basée sur la notion de modularité introduite par Newman and Girvan (2004). Leur fonction de modularité est définie par :

$$mod = \sum_{k=1}^{K} \frac{1}{|C_k|} \left(\frac{Cut(C_k, C_k)}{Cut(\mathcal{V}, \mathcal{V})} - \left(\frac{Cut(C_k, \mathcal{V})}{Cut(\mathcal{V}, \mathcal{V})} \right)^2 \right).$$

Rappelons, $|C_k|$ est le cardinal de C_k, $Cut(C_k, C_k)$ mesure la somme des poids des arêtes intra-classes, $Cut(C_k, \mathcal{V})$ mesure la somme des poids de tous les arêtes de liaison attachés à la classe C_k et $Cut(\mathcal{V}, \mathcal{V})$ représente la somme de tous les poids des arêtes du graphe. La fonction de modularité est calculée en fonction du nombre de classes. Afin d'estimer le nombre final de classes K, il faut calculer la valeur de la fonction de modularité pour chaque valeur de k où $2 \leq k \leq K$ et ensuite sélectionner la valeur maximisant la modularité.



(a) (b)

FIGURE 3.2 – (a) Valeurs propres correspondantes à la matrice Laplacienne normalisée et (b) les résultats de classification.

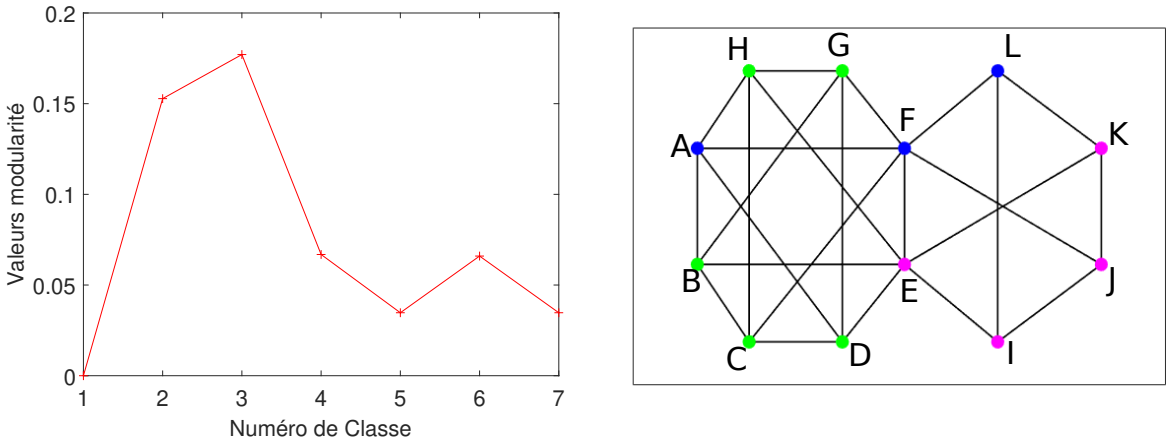
La fonction de modularité *mod* mesure l'écart entre la probabilité des connexions intra-classes du graphe, i.e. les connexions dont les nœuds sont dans les mêmes classes et la probabilité que les connexions soient aléatoires entre les nœuds du même graphe.
Yu and Ding (2010) ont introduit une autre manière de résoudre le regroupement de modularité, ils ont utilisé une fonction objective :

$$nmod = \sum_{k=1}^{K} \frac{1}{Cut(C_k, \mathcal{V})} \left(\frac{Cut(C_k, C_k)}{Cut(\mathcal{V}, \mathcal{V})} - \left(\frac{Cut(C_k, \mathcal{V})}{Cut(\mathcal{V}, \mathcal{V})} \right)^2 \right) \tag{3.31}$$

Yu and Ding (2010) ont démontré la relation entre le critère de coupe normalisée et le critère de modularité normalisé par :

$$nmod = \frac{1}{Cut(\mathcal{V}, \mathcal{V})}(k - 1 - NC) \tag{3.32}$$

Nous avons appliqué cette méthode sur le même exemple du graphe utilisé précédemment dans la méthode eigengap 3.4.1. La figure 3.3a illustre la représentation des valeurs de la modularité en fonction du nombre de classes. Nous remarquons que la valeur de la modularité est au maximum pour $k = 3$. C'est-à-dire que le nombre de classes finales K est égal à 3. La figure 3.3b montre les résultats de classification obtenus en appliquant le critère de modularité pour la détection du nombre de classes K.



(a) (b)

FIGURE 3.3 – (a) Valeur de la modularité en fonction de nombre de classes K et (b) les résultats de classification.

3.5 Classification spectrale d'un réseau urbain

Dans cette section, nous appliquons les méthodes de classification spectrale décrites précédemment sur un réseau routier pour déterminer les différentes zones de congestion. Les résultats des méthodes de partitionnement sont analysés et comparés. Le réseau routier choisi est celui de la ville de Calais correspondant aux quartiers des Cailloux et des Fontinettes extrait à partir de la base de données d'OSM (figure 1.8). Le réseau comprend des routes résidentielles et des routes tertiaires. Les vitesses pratiquées ont été simulées par un modèle microscopique (section 1.6). Dans cette section, nous appliquons la classification en construisons les deux modèles du graphe routier, le graphe primal et le graphe dual. Les étapes de construction des graphes routiers sont détaillées dans le chapitre précédent dans la section 1.4.1. La figure 3.4 montre le graphe routier des quartiers des Cailloux et des Fontinettes.

3.5.1 Classification d'un graphe primal

Dans un premier temps, nous appliquons la classification sur le graphe primal, où les intersections sont les noeuds du graphe et les routes sont les arêtes entre eux. Rappelons que le réseau a été simplifié et les routes à double sens sont représentées par des arêtes non orienté dans le graphe. Le coût d'une arête entre deux noeuds v_i et v_j est la vitesse moyenne

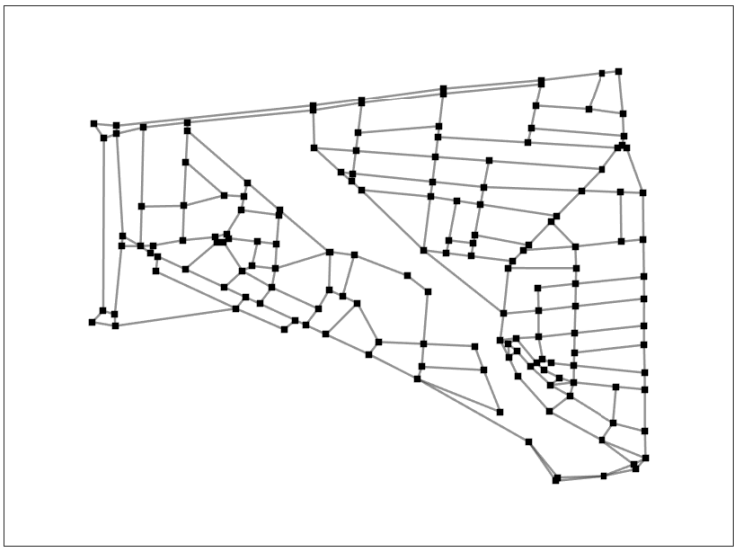


FIGURE 3.4 – Illustration du graphe routier des quartiers des Cailloux et des Fontinettes.

ν issue de la route correspondante entre les deux intersections. On estime le nombre de classes avec les différentes méthodes proposées, à savoir la méthode eigengap, et le critère de modularité.

3.5.1.1 Estimation du nombre de classes par critère eigengap

Tout d'abord, nous avons appliqué la méthode d'estimation de nombre de classes par eigengap. Cette méthode consiste à calculer les valeurs propres de la matrice Laplacienne afin de rechercher un saut entre ces derniers. Le nombre de classe choisi correspond au maximum gap (équation 3.30). D'après les résultats illustrés par la figure 3.5 nous pouvons voir nettement un saut entre les valeurs propres λ_3 et λ_4, selon cette méthode il y a donc $K = 3$ classes dans le réseau. Par la suite, nous avons appliqué l'algorithme de partitionnement spectral avec un nombre de classes K égal à 3 pour vérifier les zones de congestion dans le réseau routier. La figure 3.6 illustre les résultats obtenus et la moyenne de vitesse pour chaque groupe, en plus de leurs écart-types. Nous remarquons d'après les résultats que le réseau est regroupé en deux zones correspondent aux quartiers décrits dans le réel : Cailloux, Fontinettes (classe magenta et vert) et une zone regroupant les axes secondaires de la Nouvelle-France (classe rouge). Les vitesses moyennes des deux zones magenta et vert sont plutôt similaires et les zones à forte congestion comme la rue de Verdun et le Théâtre ne sont pas correctement isolés (voir la figure 1.10 pour les lieux). Par conséquent, le nombre de classes $K = 3$ détecté par la méthode eigengap ne permet pas de bien distinguer les poches de congestion.

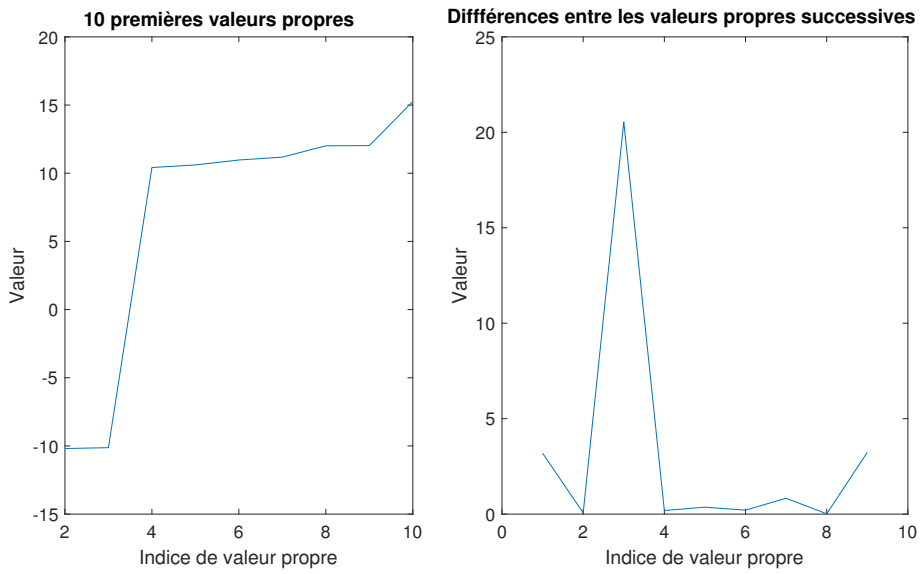


FIGURE 3.5 – Valeurs propres par ordre croissant et leurs différences.

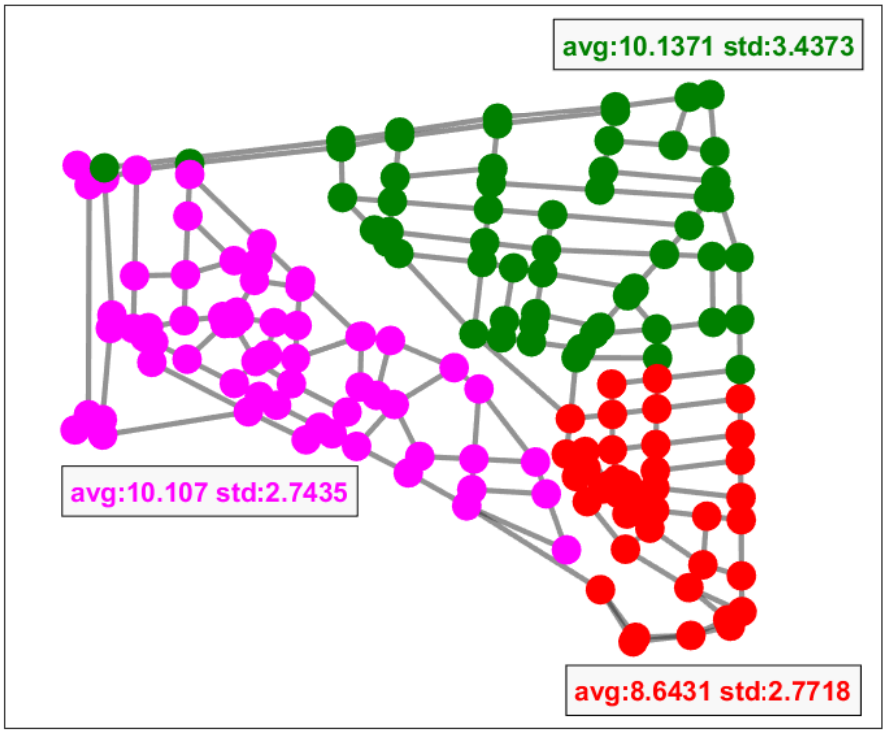


FIGURE 3.6 – Partitionnement en $K = 3$ classes.

3.5.1.2 Estimation du nombre de classes par critère de modularité

Dans un second temps, nous avons appliqué la deuxième méthode pour la recherche du nombre de classes, celle de la modularité. La fonction de la modularité utilisée est définie par l'équation 3.31. La figure 3.7 montre le résultats obtenus par le critère de modularité. Le nombre de classes choisi est celui qui maximise la fonction de modularité. Par exemple, dans la figure 3.7 le nombre de classe maximisant la fonction de modularité est égal à 5. Alors le nombre de classes choisi est $K = 5$.

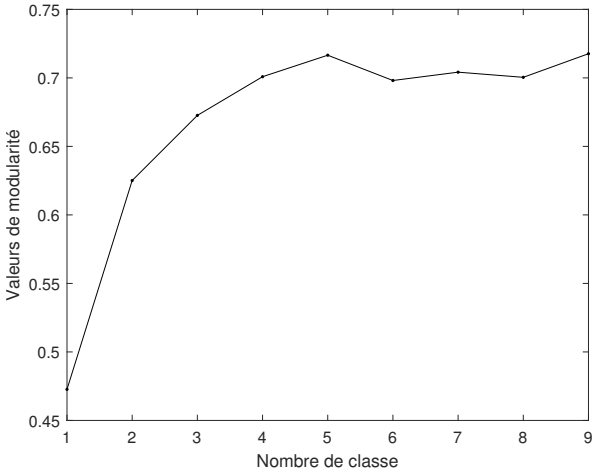


FIGURE 3.7 – Résultat du critère de modularité normalisée.

Par la suite, nous avons réalisé les partitionnements avec les valeurs de K obtenues, pour les deux algorithmes 5 et 4. La figure 3.8 montre les résultats du partitionnement obtenu en appliquant l'algorithme de Von Luxburg (algorithme 4). Ce dernier est donc capable de regrouper les noeuds ayant des vitesses similaires et de séparer les noeuds appartenant à des classes différentes. Les zones obtenues découpées représentent différentes zones de congestion et permettent d'identifier les zones à forte congestion, en particulier à proximité de la Nouvelle-France et du Théâtre sont isolées du reste. En plus, nous montrons sur la figure 3.8 les moyennes de la vitesse de chaque classe et leurs écart-types. Les vitesses moyennes des zones semblent dissimilaires entre les zones adjacentes. La figure 3.9 montre les résultats obtenus par l'algorithme de Ng et al. (algorithme 5). De la même manière que l'algorithme de Von Luxburg, cette méthode est capable de regrouper les noeuds les plus proches et de séparer les noeuds de différents vitesses. Ces résultats montrent que ces deux algorithmes permettent d'obtenir des résultats similaires en terme de partitionnement. Notons que ces algorithmes diffèrent par les étapes de normalisation des vecteurs propres.

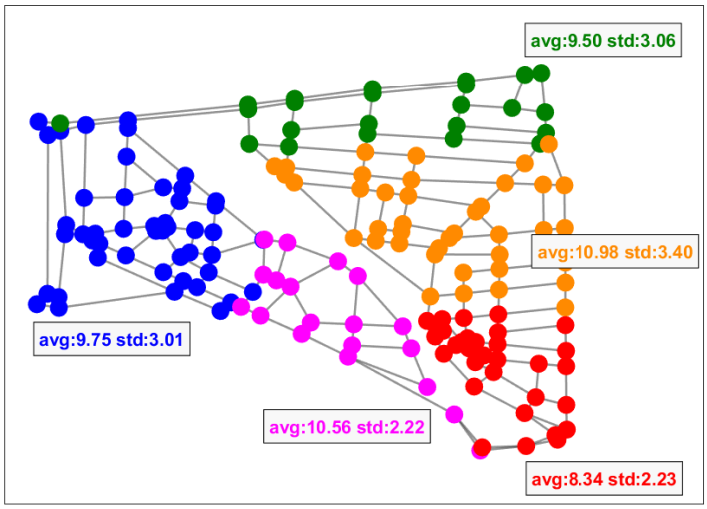


FIGURE 3.8 – Résultats de la classification spectrale avec l'algorithme de Von Luxburg

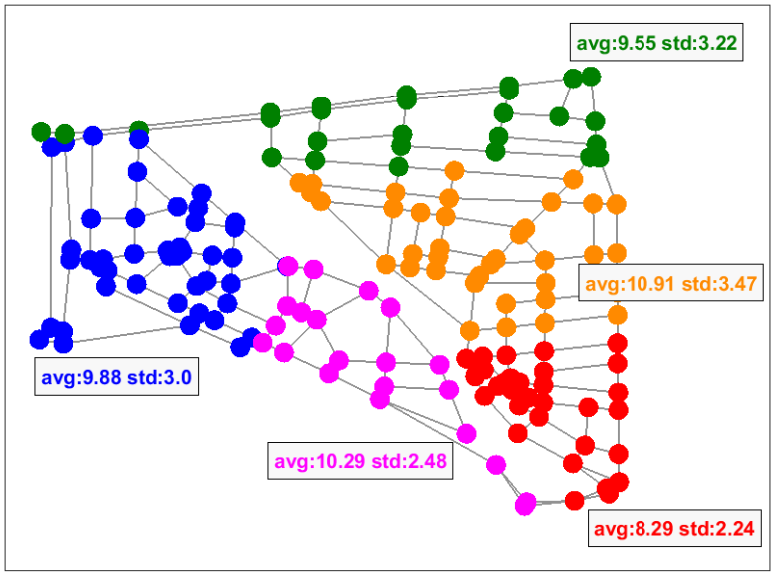


FIGURE 3.9 – Résultats de la classification spectrale avec l'algorithme de Ng et Al

3.5.2 Classification du graphe dual

Dans cette section, nous adaptons la représentation d'un réseau routier par un graphe dual. De ce fait, nous construisons le graphe $G(\mathcal{V}, E)$ associé au réseau $\aleph$, en considérant l'ensemble des routes R du réseau comme étant l'ensemble des nœuds $\mathcal{V}$ dans le graphe et les arêtes E traduisent les intersections $\mathcal{C}$ que les routes ont en commun. La figure 3.10 montre le graphe dual associé au réseau. Le graphe dual traduit donc les similarités entre les routes du réseau par l'équation suivante :

$$w_{ij} = \exp(-\frac{(v_i - v_j)^2}{2\sigma^2}), \tag{3.33}$$

avec, v_i la vitesse de la route r_i et v_j la vitesse de la route adjacente r_j. Le paramètre σ permet de prendre en compte la dispersion locale des données. Ce paramètre est calculer automatiquement par la méthode basée sur la moyenne des distances selon l'équation 3.7.

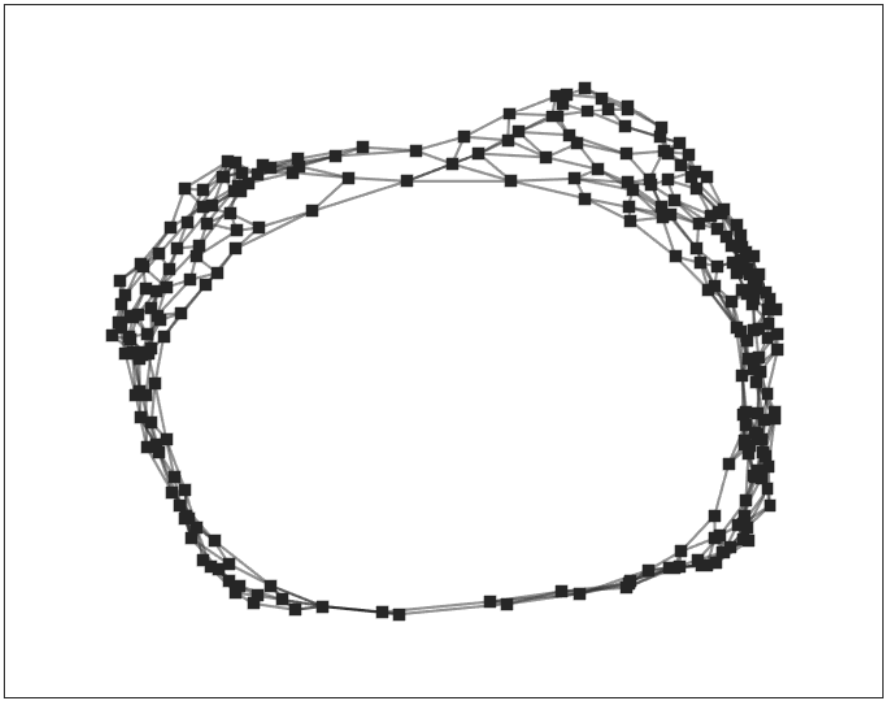


FIGURE 3.10 – Graphe dual associé au réseau

Après avoir calculer la matrice de similarité, nous appliquons les algorithmes de partitionnement spectral sur le graphe dual du réseau routier.

3.5.2.1 Estimation du nombre de classes par critère eigengap

Tout d'abord, nous recherchons le nombre de classes d'une manière automatique en appliquant le critère eigengap. La figure 3.11 illustre les résultats des valeurs propres et la différence entre les valeurs propres successives. Le gap maximale est entre la 3ème et la 4ème valeur, donc, le nombre de classes est égal à 3. Les résultats de partitionnement sont illustrés dans la figure 3.12. Ces résultats montrent un découpage simple en 3 zones qui permet d'isolé une seule zone à forte congestion (classe magenta) et l'isolation de chaque quartier : cailloux et Fontinettes ayant des valeurs de vitesses moyennes très similaire.

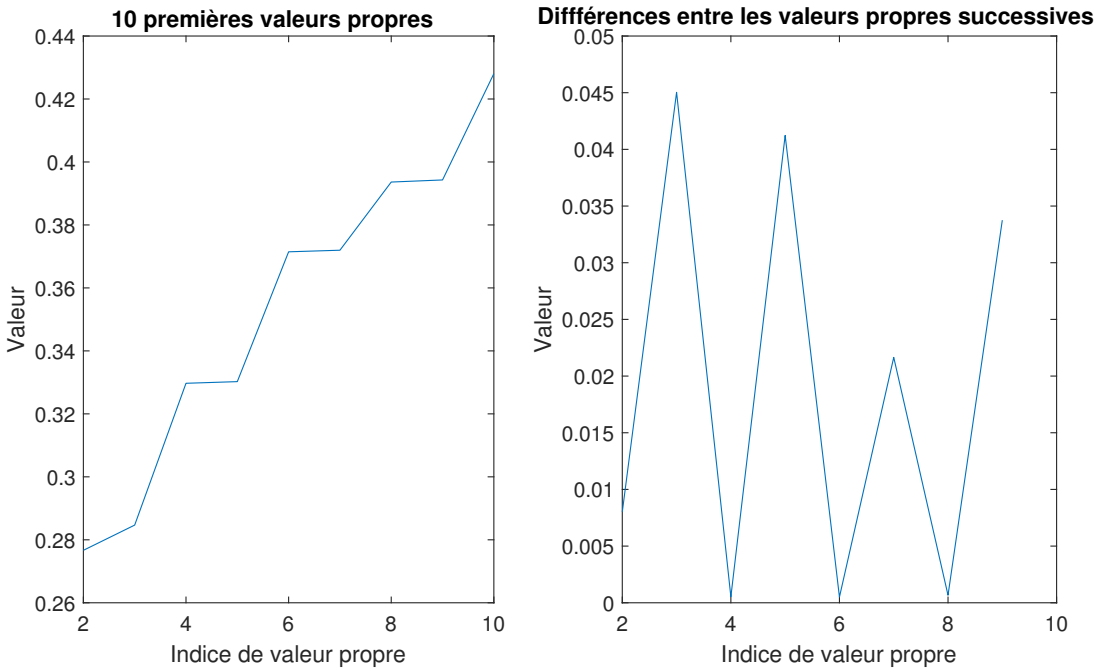


FIGURE 3.11 – Valeurs propres par ordre croissant et leurs différences.

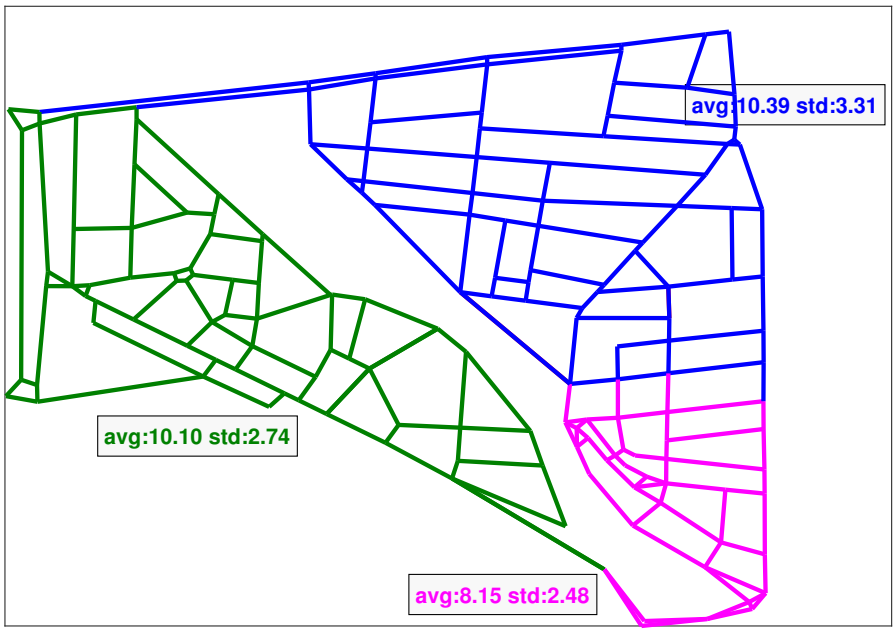


FIGURE 3.12 – Résultats de classification avec $K = 3$.

3.5.2.2 Estimation du nombre de classes par critère de modularité

Dans cette section, nous étudions les performances des algorithmes de classification on appliquons le critère de modularité pour la recherche du nombre de classe. Les résultats

de la section 3.5.1.2 montrent que les deux algorithmes de partitionnement avec la matrice Laplacienne normalisée fournissent des résultats identiques en terme de partitionnement. De ce fait, dans cette section nous étudions les résultats sur l'algorithme de Ng et al. La figure 3.13 montre la variation des valeurs de fonction de la modularité par rapport au nombre de classes. D'après les résultats, nous remarquons qu'un nombre de classe $K = 5$ maximise la fonction de la modularité.

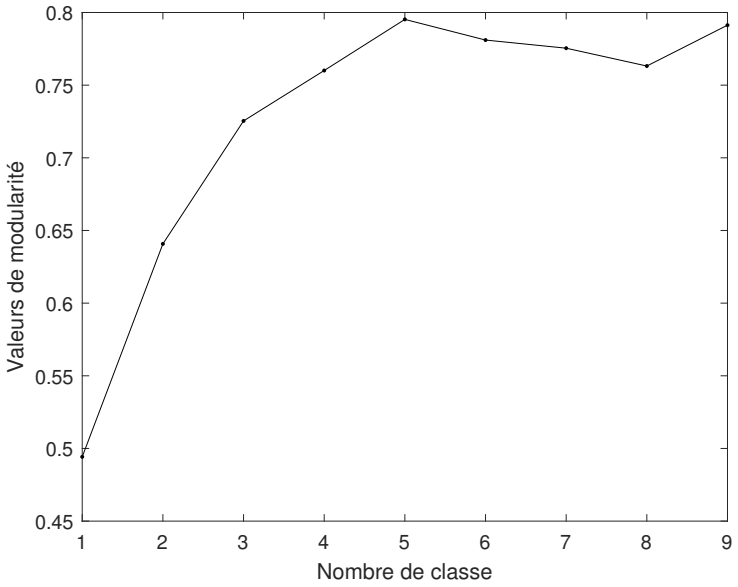


FIGURE 3.13 – Variation de la fonction de modularité.

Ensuite, nous appliquons le partitionnement spectral en appliquant l'algorithme 5. Afin de visualiser des résultats plus significatifs, les noeuds du graphe (les routes du réseau) sont représentés par les coordonnées (x,y). La figure 3.14 illustre les résultats du partitionnement, en plus, de la moyenne des vitesses de chaque classe et leur écart-type. D'après les résultats obtenus, nous observons en particulier une bonne homogénéité au niveau des quartiers et une isolation de la zone à fort trafic au niveau du Channel, du Théâtre et de la Nouvelle-France.

D'après les résultats montrés en appliquant les différents algorithmes de classification spectrale, nous déduisons, que l'approche par le graphe primal n'est pas la meilleure solution pour traiter un problème de partitionnement d'un réseau routier. Ceci s'explique par le fait que la notion de similarité des routes du réseau routier n'entre pas en compte dans cette approche. Autrement dit, en suivant la représentation d'un graphe primal, nous regroupons les croisements du réseau routier. en plus, les expériences apportées montrent que l'approche par graphe dual permet une meilleure classification, avec une isolation cohérente des zones remarquables.

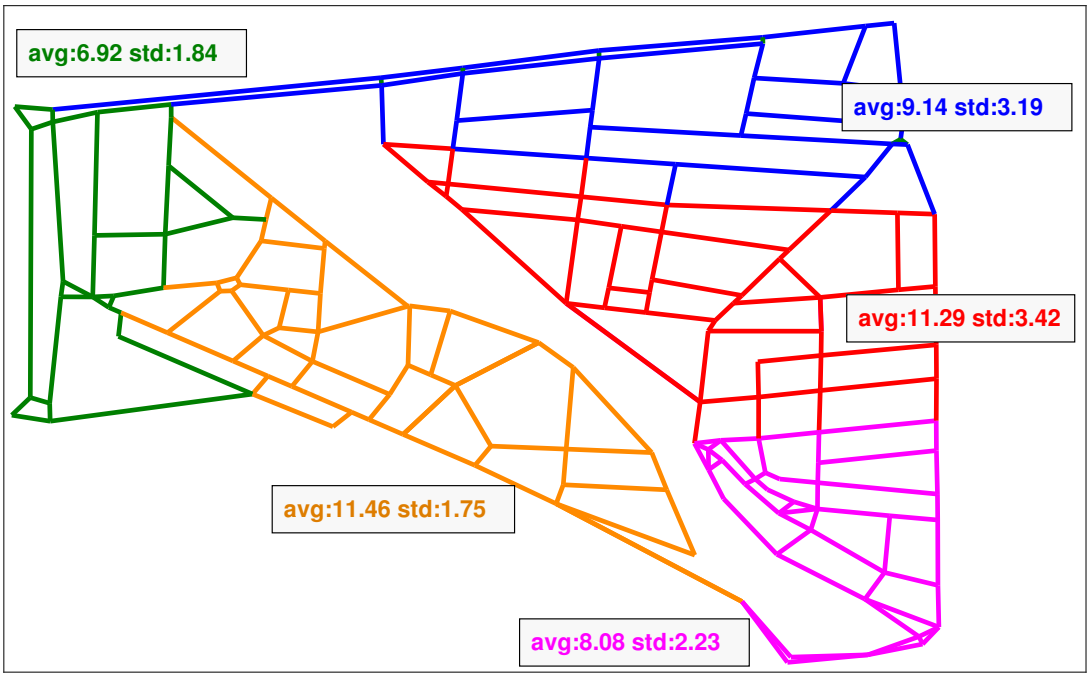


FIGURE 3.14 – Résultats de la classification avec Ng et al.

3.6 Conclusion

L'objectif de ce chapitre est d'introduire les méthodes de la classification spectrale qui permettent de regrouper un réseau de transport en des zones de congestion. L'application de la méthode de classification spectrale sur un réseau urbain a été prouvée performante dans la littérature. Ces méthodes s'appuient sur le concept de graphe et la théorie des matrices. Plus précisément, la représentation des données sous forme de graphe et la recherche des vecteurs propres de la matrice Laplacienne du graphe permettent d'offrir un espace de représentation dans lequel il est plus aisé de classer les données. La représentation en graphe primal du réseau permet de respecter la topologie du réseau réel, mais ne permet pas de réaliser une classification efficace. C'est le graphe dual du réseau qui permet de travailler sur les similarités des routes et donc d'obtenir une classification plus adéquate.

La classification spectrale nécessite la connaissance a priori du nombre des classes. Ce nombre peut soit supposé fourni par l'utilisateur, soit recherché automatiquement. Deux méthodes sont présentées dans ce but, détection à partir du saut maximum entre les valeurs propres ordonnées de la matrice Laplacienne soit à partir du maximum du critère de modularité en fonction du nombre des classes.

Malgré l'importance des algorithmes de classification spectrale, ils ne peuvent pas être adaptés pour les problèmes de classification des données dynamiques qui évoluent au cours du temps. En effet, la résolution du système propre est en général faite hors ligne et coûteuse. Pour réduire la complexité du temps de calcul, il est nécessaire de développer des algorithmes de calcul incrémentale. Dans le chapitre suivant, nous introduisons les méthodes de classification spectrale incrémentale pour la classification d'un réseau urbain dynamique.

Chapitre 4

Classification spectrale incrémentale

Sommaire

4.1 Introduction

Le problème du partitionnement spectral dans les réseaux de trafic urbain a été principalement étudié dans un cadre statique en considérant les conditions de circulation à un moment donné. Néanmoins, il est important de noter que le trafic routier est un processus qui varie dans le temps ce qui conduit à de changements aussi bien dans la structure du graphe de données que dans les poids des connexions. Pour cela il est important d'étudié la congestion du trafic dans la dimension spatio-temporelle. Pour analyser d'une manière efficace les données évolutives, il est nécessaire de développer des algorithmes de calcul incrémental : (1) pour gérer les changements de similarité entre les points de données et (2) afin d'éviter un calcul coûteux des valeurs et vecteurs propres à nouveau. L'idée de base du partitionnement spectral incrémental consiste à employer la théorie de perturbation des matrices afin de mettre à jour d'une manière dynamique les valeurs et les vecteurs propres de la matrice du graphe.

Dans ce chapitre, nous appliquons la méthode de classification spectrale incrémentale sur le réseau routier simulé. En premier lieu, nous introduisons les définitions de vecteur et matrices d'incidence pour ensuite étudier l'effet de changement de poids d'une connexion sur la matrice Laplacienne. Au final, nous appliquons l'algorithme de partitionnement spectral incrémental sur le réseau routier de la ville de Calais à différentes périodes de la journée. L'étude du problème de partitionnement incrémental est essentielle pour mieux comprendre l'évolution des données et les interprété par des méthodes de calcul en diminuant la complexité.

4.2 Positionnement du problème

Plusieurs méthodes de classification sont introduites afin de partitionner les graphes dynamiques. Valgren et al. (2007) ont appliqué la classification spectrale après l'ajout d'un nouveau point à la matrice des données. C'est-à-dire, après l'ajout d'une ligne ou colonne à la matrice de similarité. Ils ont rendus possible d'appliquer un partitionnement du graphe et la détection des classes pour les données dynamiques. Mais cette méthode ne peut pas gérer le cas de changements de similarité.

Récemment, dans la littérature, les auteurs s'intéressent à sauvegarder les flux de données afin que l'utilisateur puisse consulter les données historiques de manière interactive. Aggarwal et al. (2003) ont introduit une méthode pour effectuer cette tâche. Cette méthode comprend un composant en ligne qui calcule et stocke périodiquement des statistiques réca-

pitulatives détaillées et un composant hors ligne qui répond à une grande variété d'entrées et renvoie des informations sur les classes. Cependant, ces méthodes ne peuvent pas être directement appliquées au scénario avec des changements de similarité, comme l'évolution des données du trafic routier.

La méthode de classification spectrale incrémentale présentée par Ning et al. (2010) peut à la fois gérer l'insertion et la suppression d'un nœud du graphe et également les changements de similarité. Tout changement d'un élément de la matrice de similarité nécessite la résolution du système propre en tenant compte de l'ensemble des données. Une solution efficace est de calculer d'une manière incrémentale les valeurs et vecteurs propres en se basant sur les résultats historiques et en mettant à jour de manière dynamique le système généralisé de valeurs propres. Cette approche est utile pour les applications où la matrice de similarité est creuse et où les points de données et leurs similarités sont mis à jour dynamiquement. Afin de pouvoir modéliser la nature dynamique de la matrice des degrés et la matrice Laplacienne ainsi que les valeurs et vecteurs propres, les vecteurs d'incidence et les matrices d'incidence doivent être introduits. La matrice Laplacienne peut être décomposée en produit de deux matrices d'incidence. Un changement de similarité peut être considéré comme un vecteur d'incidence ajouté à la matrice d'incidence originale. Une insertion ou suppression d'un nœud du graphe est décomposée en une séquence de changements de similarité. De plus, chaque vecteur d'incidence nouvellement ajouté, c'est-à-dire chaque changement de similarité peut induire un changement des matrices Laplacienne et des degrés. De cette manière, le système propre et les vecteurs indicateurs des classes sont mis à jour de manière incrémentale lorsque des nouvelles données sont insérées, supprimées ou si des changements de similarité se produisent.

On considère un graphe $G = (\mathcal{V}, E)$ pondéré comprenant l'ensemble $\mathcal{V}$ de nœuds et l'ensemble E des arêtes et la matrice de similarité symétrique $\boldsymbol{W}$ qui vérifie $w_{ii} = 1$ et $0 \leq w_{ij} \leq 1$ si $i \neq j$. Soit $\boldsymbol{D} = diag\{d_1, d_2, \ldots, d_N\}$ la matrice de degré associée à $\boldsymbol{W}$ et $\boldsymbol{L}$ la matrice Laplacienne $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$. Supposons que l'on souhaite partitionner l'ensemble $\mathcal{V}$ des noeuds du graphe G en deux ensembles C_1 et C_2. Le critère d'optimisation coupe normalisée appliqué sur C_1 et C_2 s'écrit selon l'équation 3.3.3 et le vecteur indicateur u satisfait : $u^T \boldsymbol{D} u = 1$ et $u^T \boldsymbol{D} 1 = 0$. Le vecteur indicateur u est choisi de la forme :

$$u_i = \begin{cases} +\sqrt{\frac{d_{C_2}}{(d_{C_1} \times d)}} & \text{si } v_i \in C_1 \\ -\sqrt{\frac{d_{C_1}}{(d_{C_2} \times d)}} & \text{si } v_i \in C_2 \end{cases} \tag{4.1}$$

avec $d = vol(G)$, et d_{C_1}, d_{C_2} sont égaux au volume des groupes C_1 et C_2 respectivement défi-

nie par l'équation 3.17.

4.3 Vecteur d'incidence et matrice d'incidence

Un vecteur d'incidence $f_{ij}(w)$ est un vecteur colonne de dimension N qui comprend seulement deux éléments non nuls : le ième élément égal à $\sqrt{w}$ et le jème élément égal à $-\sqrt{w}$. La dimension du vecteur est égale au nombre de nœuds du graphe G (Ning et al., 2007a, 2010). Un vecteur d'incidence peut être réécrit comme : $f_{ij}(w) = \sqrt{w}.c_{ij}$ où c_{ij} est un vecteur colonne qui comprend seulement deux éléments non nuls : le ième élément égal à 1 et le jième élément égal à -1. Alors, c_{ij} s'écrit de la forme suivante : $c_{ij} = (0,\ldots,1,\ldots,0,-1,\ldots,0)^T$ Une matrice d'incidence $\mathbf{R}$ est une matrice composée des vecteurs d'incidence disposés en colonnes. La matrice $\mathbf{R}$ est composée de $|\mathcal{V}| = N$ lignes et de $|E|$ colonnes ($|E| \leq N(N-1)/2$. La matrice d'incidence peut être considérée comme une autre représentation de la matrice de similarité.

4.4 Modification des matrices

Dans ce qui suit, $\Delta w = \Delta w_{ij} = \Delta w_{ji}$ désigne un changement dans la similarité w_{ij} entre les nœuds v_i et v_j dans un graphe déjà partitionné en utilisant l'algorithme de classification spectrale. Afin d'appliquer l'algorithme de classification spectrale incrémentale, l'incrément des valeurs propres $\Delta\lambda$ et des vecteurs propres Δu résultant de Δw doivent être calculés. Mais pour y parvenir, il faut calculer la variation de la matrice de degrés $\Delta \boldsymbol{D}$ et la variation de la matrice Laplacienne $\Delta \boldsymbol{L}$. Ces formulations sont représentées dans les sous-sections suivantes.

4.4.1 Incrément de la matrice de degrés

L'incrément de la matrice de degrés $\Delta \boldsymbol{D}$ est simple, ainsi $\Delta \boldsymbol{D}$ peut être obtenue en utilisant l'expression suivante :

$$\Delta \boldsymbol{D} = \tilde{\boldsymbol{D}} - \boldsymbol{D} = \Delta w_{ij}\, diag\{b_{ij}\} \tag{4.2}$$

avec $\tilde{\boldsymbol{D}}$ est la nouvelle matrice de degrés résultante de la modification de Δw et b_{ij} un vecteur colonne dont le ième élément et le jième élément sont égaux à 1, alors, $b_{ij} = (0,\ldots,1,\ldots,0,1,\ldots,0)$. $diag\{b_{ij}\}$ est la matrice diagonale dont les éléments sont les composantes du vecteur b_{ij}. En

d'autres termes :

$$diag\{b_{ij}\} = \begin{pmatrix} \ddots & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & \ddots \end{pmatrix} \tag{4.3}$$

4.4.2 Incrément de la matrice laplacienne

Après avoir calculé la variation de la matrice de degrés $\Delta \boldsymbol{D}$, il faut aussi calculer la variation de la matrice laplacienne $\Delta \boldsymbol{L}$ en utilisant l'équation suivante :

$$\Delta \boldsymbol{L} = \tilde{\boldsymbol{L}} - \boldsymbol{L} = \Delta w_{ij}\, c_{ij} c_{ij}^T \tag{4.4}$$

où, $\tilde{\boldsymbol{L}}$ est la nouvelle matrice Laplacienne résultante de la modification de Δw. Soit $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} \in \mathbb{R}^{N \times N}$ la matrice Laplacienne, il existe une matrice d'incidence $\boldsymbol{R}$ telle que : $\boldsymbol{L} = \boldsymbol{R}\boldsymbol{R}^T$. La matrice $\boldsymbol{R}$ contient les vecteurs d'incidences $f_{ij}(w_{ij})$ avec $1 \leq i < j \leq N$ (Ning et al., 2007a, 2010) :

$$\begin{aligned} \boldsymbol{L} = \boldsymbol{R}\boldsymbol{R}^T &= \sum_{(1 \leq i < j \leq N)} f_{ij}(w_{ij}) f_{ij}(w_{ij})^T \\ &= \sum_{(1 \leq i < j \leq N)} w_{ij} c_{ij} c_{ij}^T \end{aligned} \tag{4.5}$$

avec $c_{ij}c_{ij}^T$ est une matrice $N \times N$:

$$c_{ij}c_{ij}^T = \begin{pmatrix} & \vdots & & \vdots & \\ \cdots & +1 & \cdots & -1 & \cdots \\ & \vdots & & \vdots & \\ \cdots & -1 & \cdots & +1 & \cdots \\ & \vdots & & \vdots & \end{pmatrix} \tag{4.6}$$

À partir de cette proposition, on cherche à étudier l'influence de changement de poids sur la matrice $\boldsymbol{L}$. Supposons que la similarité entre deux noeuds (v_i, v_j) a changé. Le changement de similarité Δw_{ij} peut être incorporé dans la matrice d'incidence $\boldsymbol{R}$ en ajoutant le vecteur d'incidence $f_{ij}(w_{ij})$ à $\boldsymbol{R}$:

$$\tilde{\boldsymbol{R}} = \left[\mathbf{R}, f_{ij}(\Delta w_{ij})\right] \tag{4.7}$$

On définit la nouvelle matrice Laplacienne $\tilde{L} = \tilde{R}\tilde{R}^T$. Selon le fait que : $f_{ij}(w) = \sqrt{w}c_{ij}$, la matrice $\tilde{L}$ est définie par :

$$\begin{aligned}\tilde{L} &= \left[\mathbf{R}, f_{ij}(\Delta w_{ij})\right] \begin{bmatrix} \mathbf{R}^T \\ f_{ij}(\Delta w_{ij})^T \end{bmatrix} \\ &= \mathbf{R}\mathbf{R}^T + f_{ij}(\Delta w_{ij}).f_{ij}(\Delta w_{ij})^T \\ &= \mathbf{R}\mathbf{R}^T + \Delta w_{ij}.c_{ij}\,c_{ij}^T \\ &= \mathbf{L} + \Delta w_{ij}.c_{ij}\,c_{ij}^T\end{aligned} \tag{4.8}$$

Alors, l'incrément de la matrice Laplacienne $\boldsymbol{L}$ par rapport à Δw_{ij} est défini par :

$$\begin{aligned}\Delta \boldsymbol{L} &= \tilde{\boldsymbol{L}} - \boldsymbol{L} \\ &= \Delta w_{ij} c_{ij}\,c_{ij}^T\end{aligned} \tag{4.9}$$

4.5 Incrément des valeurs propres

Les changements des similarités entre les noeuds du graphe, induisent à un nouveau calcul des valeurs et vecteurs propres associés. L'algorithme de partitionnement incrémental propose une nouvelle méthode qui consiste à calculer ce changement sans la nécessité de calculer ces valeurs à nouveau. Considérons le système propre généralisé $\mathbf{L}u = \lambda \mathbf{D}u$ où $\boldsymbol{L} \in \mathbb{R}^{N\times N}$ et $\boldsymbol{D} \in \mathbb{R}^{N\times N}$ sont des matrices symétriques, alors la perturbation de λ en fonction de $\boldsymbol{L}$ et $\boldsymbol{D}$ est définie par :

$$\Delta\lambda = \frac{u^T(\Delta \boldsymbol{L} - \lambda \Delta \boldsymbol{D})\,u}{u^T \boldsymbol{D} u}. \tag{4.10}$$

En effet, différencier les deux côtés du système propre généralisé : $\boldsymbol{L}u = \lambda \mathbf{D}u$. On obtient :

$$(\Delta \boldsymbol{L})u + \boldsymbol{L}(\Delta u) = (\Delta\lambda)\mathbf{D}u + \lambda(\Delta \boldsymbol{D})u + \lambda \boldsymbol{D}(\Delta u) \tag{4.11}$$

Si on multiplie par u^T à gauche, on obtient :

$$u^T(\Delta \boldsymbol{L})u + u^T\boldsymbol{L}(\Delta u) = (\Delta\lambda)u^T\mathbf{D}u + \lambda u^T(\Delta \boldsymbol{D})u + \lambda u^T\boldsymbol{D}(\Delta u)$$

Or, $\boldsymbol{L}u = \lambda \boldsymbol{D}u$ et $u^T\boldsymbol{L} = \lambda \boldsymbol{D}u^T$. Puisque les deux matrices $\boldsymbol{L}$ et $\boldsymbol{D}$ sont symétriques l'équation 4.11 peut être réécrite par :

$$u^T(\Delta \boldsymbol{L})u = (\Delta\lambda)u^T\boldsymbol{D}u + \lambda u^T(\Delta \boldsymbol{D})u$$

Après quelques manipulations, l'équation 4.10 est vérifiée. Selon le fait que $f_{ij}(\Delta w_{ij}) = \sqrt{\Delta w_{ij}}c_{ij}$, et en remplaçons les valeurs de $\Delta\boldsymbol{L}$ et $\Delta\boldsymbol{D}$ par leurs valeurs. L'incrément de $\Delta\lambda$

est représenté par :

$$\Delta\lambda = \Delta w_{ij} \frac{u^T \left(c_{ij} c_{ij}^T - \lambda \, diag\{b_{ij}\} \right) u}{u^T \boldsymbol{D} u} \tag{4.12}$$

On peut remarquer que :

$u^T c_{ij} c_{ij}^T u = (u_i - u_j)^2$,
$u^T \, diag\{b_{ij}\} u = u_i^2 + u_j^2$, avec $u^T \boldsymbol{D} u = 1$

Et par la suite, puisque u est normalisé, l'incrément de $\Delta\lambda$ est exprimé par :

$$\Delta\lambda = \Delta w_{ij} \left((u_i - u_j)^2 - \lambda (u_i^2 + u_j^2) \right) \tag{4.13}$$

$\Delta\lambda$ peut encore être simplifié en considérant que les deux classes ont à peu près les mêmes degrés et $\lambda \ll 1$. Alors, l'incrément de $\Delta\lambda$ est simplifié par Ning et al. (2007a) :

$$\Delta\lambda \approx \begin{cases} -2\lambda \frac{\Delta w_{ij}}{d}, & \text{si } v_i \text{ et } v_j \text{ sont dans la même classe.} \\ 4\frac{\Delta w_{ij}}{d}, & \text{si } v_i \text{ et } v_j \text{ sont dans de classes différentes.} \end{cases} \tag{4.14}$$

Le changement de la similarité entre deux noeuds v_i et v_j du graphe influence la valeur de λ, i.e., si la valeur de la similarité augmente, la valeur de λ est plus élevée si ces noeuds sont dans des classes différentes.

4.6 Incrément des vecteurs propres

Après avoir calculé les variations des valeurs propres, on introduit dans cette section la méthode de calcul des variations des vecteurs propres, noté Δu afin de procéder à la classification spectrale incrémentale sur le graphe modifié.

En général, l'incrément d'un vecteur propre Δu peut être résolu par la méthode de Lanczos. Ces méthodes fonctionnent efficacement sur des matrices creuses, mais elles sont moins pratiques pour les matrices de grande taille qui ne sont pas nécessairement creuses. Nous adoptons donc une approche de calcul rapide et efficace de Δu afin de pouvoir l'appliquer à des données d'un graphe évolutives. Reprenons l'équation 4.11, en remplaçant $\Delta \boldsymbol{L}$ par l'équation 4.9 et $\Delta \boldsymbol{D}$ par l'équation 4.2 et après quelques manipulations, on obtient :

$$\mathbf{K} \Delta u = h \tag{4.15}$$

avec $\mathbf{K} = \mathbf{L} - \lambda \boldsymbol{D}$ et $h = ((\Delta\lambda)\boldsymbol{D} + \lambda(\Delta \boldsymbol{D}) - \Delta \boldsymbol{L})\, u$. Alors, l'incrément des vecteurs propres

Δu peut être résolu par :

$$\Delta u = \left(\boldsymbol{K}^T\boldsymbol{K}\right)^{-1}\boldsymbol{K}^T h \tag{4.16}$$

La résolution de cette équation nécessite un coût de calcul très élevé du fait que les matrices $\boldsymbol{K}$ et $\boldsymbol{K}^T\boldsymbol{K}$ sont des matrices singulières. En effet, $\boldsymbol{L}u = \lambda \boldsymbol{D}u$, c'est-à-dire, $(\boldsymbol{L} - \lambda \boldsymbol{D})u = \boldsymbol{K}u$. En effet, un grand ensemble de données produit des matrices de grande taille. De ce fait, la taille de $\boldsymbol{K}$ et $\boldsymbol{K}^T\boldsymbol{K}$ devient très grande, ce qui signifie que la résolution de l'équation nécessite un coût de calcul très élevé.
Pour cela, on propose de considérer qu'un changement du poids d'un couple de noeuds n'aura d'influence que sur les K-plus proches voisins. Le voisinage de deux nœuds v_i et v_j est défini comme l'ensemble des nœuds qui sont connectés à v_i avec un poids $w_{ik} > \tau$ ou qui sont connectés à v_j avec un poids $w_{jk} > \tau$. Le paramètre τ est un seuil prédéfini qui peut être nul si l'ensemble de données est bien dispersé. En d'autres termes, étant donné un changement de similarité $f_{ij}(\Delta w_{ij})$, le voisinage des nœuds v_i et v_j est défini comme suit :

$$N_{ij} = \{k / w_{ik} > \tau \text{ ou } w_{jk} > \tau\}, \tag{4.17}$$

avec, N_{ij} le voisinage spatial de v_i et v_j.
De plus, on suppose que les éléments $\Delta u_k = 0$ si $k \notin N_{ij}$ et on les élimine de Δu, et par suite les colonnes correspondantes dans la matrice $(\boldsymbol{K} = \boldsymbol{L} - \lambda \boldsymbol{D})$ sont également éliminées. Ainsi, après élimination des éléments dans u et Δu qui ne sont pas dans le voisinage spatial N_{ij}, les éléments qui restent forment Δu_{ij} et $\boldsymbol{K}_{ij}$. Ainsi Δu calculé dans l'équation 4.16 peut être remplacé par l'équation suivante sans aucune perte d'information :

$$\Delta u_{N_{ij}} = \left(\boldsymbol{K}^T_{\boldsymbol{N_{ij}}}\boldsymbol{K}_{\boldsymbol{N_{ij}}}\right)^{-1}\boldsymbol{K}^T_{\boldsymbol{N_{ij}}} h_{N_{ij}} \tag{4.18}$$

avec $h_{N_{ij}}$ est un vecteur colonne calculé comme suit :

$$h_{N_{ij}} = \left(\Delta\lambda\, \boldsymbol{D}_{\boldsymbol{N_{ij}}} + \lambda\, \Delta \boldsymbol{D}_{\boldsymbol{N_{ij}}} - \Delta \boldsymbol{L}_{\boldsymbol{N_{ij}}}\right) u_{N_{ij}}. \tag{4.19}$$

Après la réduction du nombre de colonnes de $\boldsymbol{K}_{\boldsymbol{N_{ij}}}$ par rapport à la taille de l'ensemble de données, le coût de calcul de l'équation est optimisé.

4.7 Raffinement des valeurs et vecteurs propres

Dans la section 4.5, nous avons introduit l'approximation du premier ordre de l'incrément des valeurs propres. Les équations 4.10 et 4.13 représentes l'approximation du premier ordre des valeurs propres et ignore l'erreur du second ordre. Mais dans certains cas,

l'erreur du second ordre n'est pas négligeable, nous présentons donc, dans cette section, une approximation plus précise de l'incrément des valeurs propres en considérant l'erreur du second et troisième ordre et en utilisant l'incrément des vecteurs propres existant. L'incrément des vecteurs propres peut être affiné d'une manière itérative, en utilisant l'incrément des valeurs propres plus précis, c'est-à-dire, celle du second ordre. En répétant alternativement cette procédure, nous présentons les étapes itératives pour affiner $\Delta\lambda$ et Δu.

Considérons un système propre généralisé $\boldsymbol{L}u = \lambda \boldsymbol{D}u$, alors la perturbation de $\Delta\lambda$ du second ordre en fonction de $\boldsymbol{L}$ et $\boldsymbol{D}$ est donnée par :

$$\Delta\lambda = \frac{u^T\left(\Delta\boldsymbol{L} - \lambda\,\Delta\boldsymbol{D}\right)(u + \Delta u)}{u^T\left(\boldsymbol{D} + \Delta\boldsymbol{D}\right)(u + \Delta u)} \tag{4.20}$$

Si on différencie les deux côtés du système de valeurs propres généralisé et on conserve les éléments d'approximation d'erreur du second et troisième ordre, on obtient :

$$\begin{aligned}\Delta\boldsymbol{L}u + \boldsymbol{L}\Delta u + \Delta\boldsymbol{L}\Delta u = \Delta\lambda\boldsymbol{D}u + \lambda\Delta\boldsymbol{D}u + \lambda\boldsymbol{D}\Delta u + \Delta\lambda\,\Delta\boldsymbol{D}\,u + \lambda\Delta\boldsymbol{D}\,\Delta u \\ + \Delta\lambda\,\boldsymbol{D}\Delta u + \Delta\lambda\,\Delta\boldsymbol{D}\,\Delta u\end{aligned} \tag{4.21}$$

Par la suite, on multiplie à gauche les deux côtés par u^T, et nous considérons que $u^T\boldsymbol{L} = \lambda\boldsymbol{D}u^T$ et puisque $\boldsymbol{L}$ et $\boldsymbol{D}$ sont symétrique, nous obtenons :

$$\begin{aligned}u^T\Delta\boldsymbol{L}u + u^T\Delta\boldsymbol{L}\,\Delta u =& \Delta\lambda\,u^T\boldsymbol{D}u + \lambda u^T\Delta\boldsymbol{D}\,u + \Delta\lambda\,u^T\Delta\boldsymbol{D}u \\ &+ \lambda u^T\Delta\boldsymbol{D}\,\Delta u + \Delta\lambda\,u^T\boldsymbol{D}\Delta u + \Delta\lambda\,u^T\Delta\boldsymbol{D}\,\Delta u\end{aligned} \tag{4.22}$$

Après quelques manipulations, l'équation 4.20 est vérifiée. Selon le fait que $f_{ij}(\Delta w_{ij}) = \sqrt{\Delta w_{ij}}c_{ij}$ et en remplaçons les valeurs de $\Delta\boldsymbol{L}$ et $\Delta\boldsymbol{D}$ par leurs valeurs par les équations 4.9 et 4.2 respectivement. Alors, $\Delta\lambda$ du système propre généralisé $\boldsymbol{L}u = \lambda\boldsymbol{D}u$ est donnée par :

$$\Delta\lambda = \Delta w_{ij}\,\frac{u^T(c_{ij}\,c_{ij}^T - \lambda\,diag\{b_{ij}\})(u + \Delta u)}{u^T(\boldsymbol{D} + \Delta w_{ij}\,diag\{b_{ij}\})(u + \Delta u)} \tag{4.23}$$

Donc, l'incrément du second ordre de $\Delta\lambda$ est donnée par :

$$\Delta\lambda = \Delta w_{ij}\frac{a + b}{1 + c + d + e}, \tag{4.24}$$

avec,
$a = (u_i - u_j)^2 - \lambda(u_i^2 + u_j^2)$.
$b = (u_i - u_j)(\Delta u_i - \Delta u_j) - \lambda(u_i\Delta u_i + u_j\Delta u_j)$.
$c = \Delta w_{ij}(u_i^2 + u_j^2)$.

$d = \sum_{k \in N_{ij}} u_k d_k \Delta u_k$.
$e = \Delta w_{ij}(u_i \Delta u_i + u_j \Delta u_j)$.
Ici, $\sum_{k \in N_{ij}} u_i d_i \Delta u_i$ est une approximation de $u^T \boldsymbol{D} \Delta u$ avec l'hypothèse que l'impact du changement de similarité Δw_{ij} se situe dans le voisinage spatial N_{ij}. Donc, e peut être ignoré puisque $e \ll 1$ et $e \ll c$, donc $\Delta\lambda$ est simplifié suivant l'équation suivante :

$$\Delta\lambda = \Delta w_{ij} \frac{a+b}{1+c+d}. \tag{4.25}$$

Cependant, l'approximation du second ordre de $\Delta\lambda$ dans l'équation 4.25 dépend de l'incrément des vecteurs propres Δu dans l'équation 4.16, et Δu dépend également de $\Delta\lambda$. Nous pouvons initialiser $\Delta\lambda$ par l'équation 4.13 puis affiner $\Delta\lambda$ et Δu alternativement. Les étapes de raffinement itératif sont résumées dans la section suivante.

4.8 Mise à jour des valeurs et vecteurs propres

Afin d'appliquer une classification incrémentale des données évolutives, il faut tout d'abord introduire les étapes de mise à jour des valeurs et vecteurs propres, pour ensuite introduire l'algorithme de classification spectrale incrémentale. Les étapes de mise à jour des valeurs et vecteurs propres noté par $\Delta\lambda$ et Δu respectivement, sont les suivantes :

— initialiser $\Delta u = 0$
— Calculer $\Delta\lambda$ en utilisant l'équation 4.25.
— Calculer Δu en utilisant l'équation 4.18.
— Répéter les étapes 2 et 3 jusqu'à ce que les valeurs de $\Delta\lambda$ et Δu changent peu ou jusqu'à un nombre donné d'itérations.

La propriété de convergence des étapes de mise à jour des valeurs et vecteurs propres est encore inconnue. Plusieurs expériences montrent que l'erreur d'approximation des valeurs propres et des vecteurs propres diminue fondamentalement lorsque les temps d'itération $n \geq 2$. Nous reprenons l'exemple de la figure 1.1 pour illustrer cette propriété. Cela consiste à supprimer une arête du graphe et ensuite à calculer les nouvelles valeurs et vecteurs propres en appliquant les étapes de mise à jour. La figure 4.1 illustre les erreurs d'approximation de la deuxième plus petite valeur propre et du vecteur propre associé après avoir supprimé une seule arête. L'erreur d'approximation entre deux valeurs propres est calculée par la différence entre elles. Alors que, l'erreur d'approximation entre deux vecteurs propres est

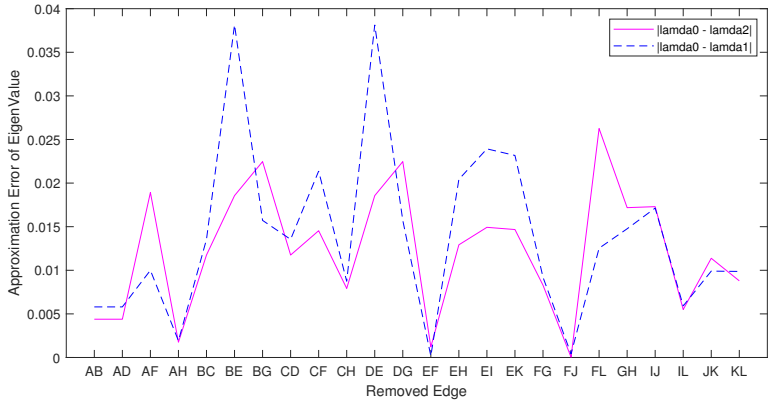
calculée par l'équation suivante (Ning et al., 2007a) :

$$E_{diff}(u_0, u_1) = \min_y ||y\frac{u_0}{||u_0||} - \frac{u_1}{||u_1||}||^2. \tag{4.26}$$

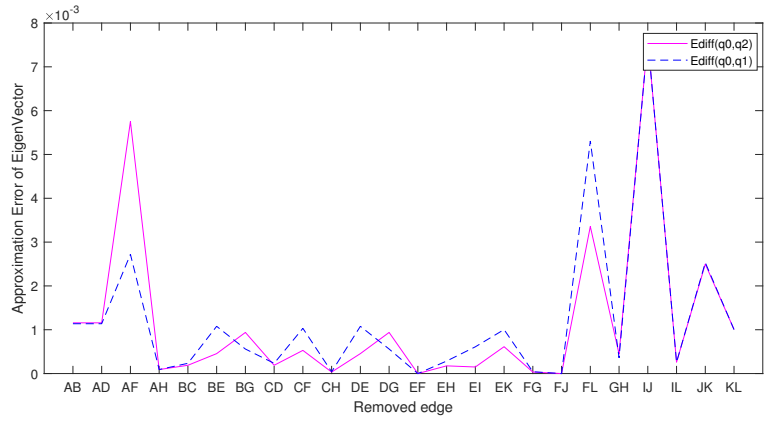
avec,

$$y = \frac{u_0^T u_1}{||u_0||\,||u_1||} \tag{4.27}$$

La figure 4.1a donne l'erreur d'approximation de valeur propre après une itération $|\lambda_0 - \lambda_1|$ et après deux itérations $|\lambda_0 - \lambda_2|$, respectivement. La figure 4.1b donne l'erreur d'approximation du vecteur propre après une itération $E_{diff}(u_0, u_1)$, et après deux itérations $E_{diff}(u_0, u_2)$, respectivement. En fait, d'après ces résultats on peut voir que l'erreur diminue après la deuxième itération.



(a)



(b)

FIGURE 4.1 – Erreur d'approximation de la deuxième plus petite valeur propre et du vecteur propre associé après suppression d'une arête.

4.9 Algorithme de classification spectrale incrémentale

l'algorithme de classification spectral incremental est résumé dans l'algorithme 7. Notons que l'algorithme de la classification spectrale incrémentale s'appuie sur des approximations pour la mise à jour des valeurs et vecteurs propres, il doit être réinitialisé après un certain nombre d'itérations.

Algorithme 7 : Algorithme de classification spectrale incrémentale (ISC)

1: Acquérir à l'instant t suffisamment de données et construire le graphe $G(\mathcal{V}, E)$
2: Pour le graphe $G(\mathcal{V}, E)$ calculer les matrices $\boldsymbol{W}$, $\boldsymbol{D}$ et $\boldsymbol{L}$.
3: Calculer les K vecteurs propres de $\boldsymbol{L}$ relatifs aux K-plus petites valeurs propres de $\boldsymbol{L}$.
4: Pour un changement de similarité, calculer $\Delta\lambda$ et Δu en utilisant les étapes de mise à jour de $\Delta\lambda$ et Δu détaillées dans la section 4.8, et mettre à jour les matrices de degré et Laplacienne.

4.10 Complexité

Par la suite la complexité temporelle est discutée. Le système de valeurs propres généralisé peut être transformé en un problème de valeurs propres standard $\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}u = \lambda u$. La résolution d'un problème de valeurs propres standard prend $O(N^3)$ où N est la taille du graphe à partitionner.

L'avantage apporté par l'algorithme de la classification spectrale incrémentale est son exécution rapide (Ning et al., 2007a) par rapport à l'algorithme de la classification spectrale. La différence de temps d'exécution entre l'algorithme de la classification spectrale et l'algorithme de la classification spectrale incrémentale se situe dans l'étape de calcul de la valeur propre et du vecteur propre. Les complexités temporelles respectives du calcul de $\Delta\lambda$ et Δu sont $O(N)$ qui est le temps de calcul de $\sum_{i=1}^{N} u_i d_i^2$ et $O(\bar{N}^2 N) + O(\bar{N}^3) + O(\bar{N}N) + O(\bar{N}^2)$ où, $\bar{N}$ est le nombre moyen de voisins pour chaque paire de nœuds du graphe dans le contexte de la définition du voisinage dans ce chapitre. Le temps de calcul Δu combine les complexités suivantes :

— $O(\bar{N}^2 N)$: le temps nécessaire pour calculer $\boldsymbol{K}_{\boldsymbol{N}_{ij}}^T \times \boldsymbol{K}_{\boldsymbol{N}_{ij}}$ qui est le temps de calcul du produit d'une matrice de taille $(\bar{N} \times N)$ par une autre de taille $(N \times \bar{N})$.

— $O(\bar{N}^3)$: le temps nécessaire pour inverser le produit des deux matrices $\boldsymbol{K}_{\boldsymbol{N}_{ij}}^T \times \boldsymbol{K}_{\boldsymbol{N}_{ij}}$ qui est de taille $(\bar{N} \times \bar{N})$.

— $O(\bar{N}N)$: le temps nécessaire pour calculer $\boldsymbol{K}_{\boldsymbol{N}_{ij}}^T \times h_{N_{ij}}$ qui est le produit d'une matrice de taille $(\bar{N} \times N)$ par un vecteur colonne de dimension N.

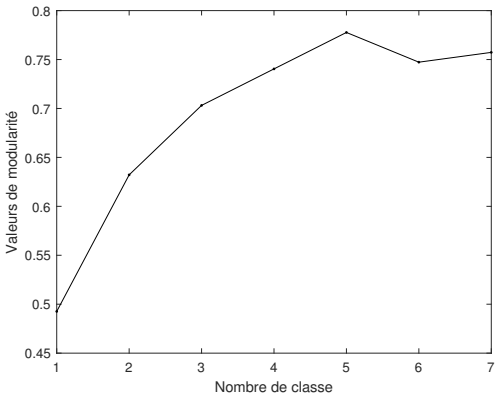
— $O(\bar{N}^2)$: le temps nécessaire pour calculer $\left(\boldsymbol{K}_{\boldsymbol{N}_{ij}}^T \boldsymbol{K}_{\boldsymbol{N}_{ij}}\right)^{-1} \times \left(\boldsymbol{K}_{\boldsymbol{N}_{ij}}^T h_{N_{ij}}\right)$ qui est le produit d'une matrice de taille $(\bar{N} \times \bar{N})$ par un vecteur colonne de dimension N.

Par conséquent, la complexité temporelle de l'algorithme de classification spectrale incrémentale est $O(\bar{N}^2 N) + O(\bar{N}^3) + O(\bar{N}N) + O(\bar{N}^2) + O(N)$ qui est la somme des temps d'exécution pour calculer $\Delta\lambda$ et Δu. Lorsque la matrice de similarité est creuse, le nombre moyen de voisins $\bar{N}$ est petit et dans ce cas la complexité temporelle de l'algorithme de classification spectrale incrémentale devient $O(N)$.

4.11 Classification spectrale incrementale d'un réseau urbain

Dans cette section, nous appliquons l'algorithme de partitionnement spectral incremental sur un réseau routier pour évaluer ses performances à déterminer les zones de congestion et à réduire la complexité en temps de calcul. Cet algorithme est conçu pour traiter les ensembles de données dynamiques. Le trafic routier est un tel type de données. Au cours du temps, les liens congestionnés peuvent devenir fluides et vice-versa, ce qui conduit à un changement dans les vitesses pratiquées sur les liens du réseau. Nous reprenons l'exemple du réseau routier de la ville de Calais correspondant aux quartiers des Cailloux et des Fontinettes (figure 3.4). Les vitesses pratiquées sont estimées par des périodes de temps de 10 minutes. Dans notre expérience, nous considérons deux périodes $t = 1$ et $t = 2$. Nous procédons de la manière suivante : le partitionnement spectral est appliqué au réseau à une période de temps $t = 1$. Ensuite, à la période $t = 2$, le partitionnement spectral est également appliqué ainsi que l'algorithme du partitionnement spectral incrémental. Enfin, les résultats des deux méthodes sont évalués.

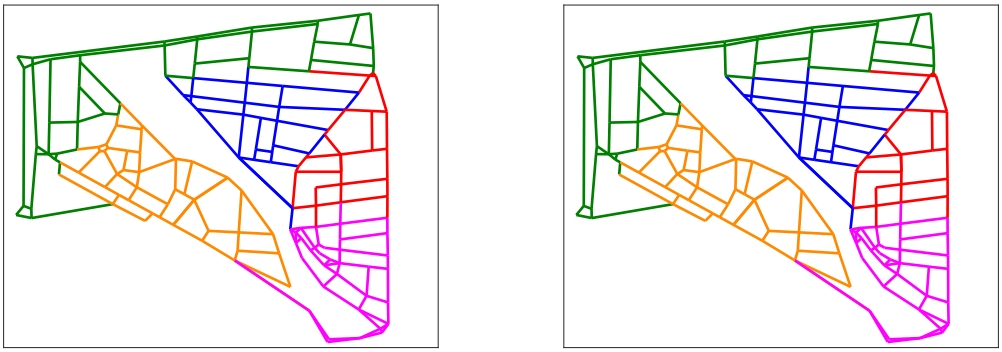
Nous utilisons la coupe normalisée standard comme base de référence qui est mise en œuvre en utilisant l'algorithme de Ng et al (algorithme 5). La matrice de similarité $\boldsymbol{W}$, la matrice de degré $\boldsymbol{D}$ et la matrice Laplacienne $\boldsymbol{L}$ sont construites sur le graphe à la période $t = 1$. Ensuite, l'algorithme de Ng et al. est appliqué sur le graphe et les résultats de partitionnement sont illustrés dans la figure 3.14. Pour la période $t = 2$, en premier lieu, nous recherchons le nombre de classes d'une manière automatique en appliquant le critère de modularité. La figure 4.2 montre que le nombre de classes est égal à 5. Ensuite, les deux algorithmes de partitionnement standard (IND) et incrémental (ISC) résumés par les algorithmes 5 et 7 respectivement, sont appliqués sur le graphe modifié à la période $t = 2$. Lors-



(a)

FIGURE 4.2 – Variation de la fonction de modularité.

qu'un changement se produit, IND recalcule les matrices de similarité **W**, de degrés **D** et Laplacienne **L** afin d'appliquer la décomposition du système de valeurs et vecteurs propres. Par contre, ISC requiert de mettre a jour les deux matrices de degrés **D** et Laplacienne **L** selon les équations 4.2 et 4.9, respectivement. Ensuite, les nouvelles valeurs et vecteurs propres sont calculés en considérant seulement les nouveaux changements selon les équations 4.25 et 4.18 respectivement. Les valeurs et les vecteurs propres sont affinés pour réduire l'erreur sur ces derniers.



(a) (b)

FIGURE 4.3 – Résultats de partitionnement en appliquant l'algorithme Ng et al. et ISC.

La figure 4.3 illustre les résultats obtenus par les deux méthodes. En comparant les résultats illustrés par les figures 4.3a et 4.3b, nous remarquons que les deux algorithmes IND et ISC donne les mêmes résultats de partitionnement. Ceci est expliqué par le fait que les

conditions de trafic changent légèrement pour des périodes de 10 minutes. De plus, l'algorithme de partitionnement spectral incrémental est basé sur une approximation du second ordre, où les erreurs diminuent après raffinement des valeurs et vecteurs propres.

Afin d'évaluer les résultats, nous calculons les vitesses moyennes et l'écart-type de chaque classe dans le but de valider l'efficacité de l'approche incremental dans la séparation des classes homogènes connectées. Les résultats des mesures d'évaluation sont présentés en détail dans le tableau 4.1. En comparant les valeurs moyennes des vitesses routières et les écarts-types dans les différentes classes, nous remarquons que les vitesses routières moyennes entre les partitions sont dissimilaires. De plus, la faible valeur de l'écart-type pour chaque classe définit une bonne homogénéité intra-clusters.

TABLEAU 4.1 – Valeurs de la vitesse moyenne (m/s) et écarts-types dans le cas des approches IND, ISC.

(μ/σ)	Rouge	Bleu	Vert	Orange	Magenta
SC	11.49/ 1.40	9.35/3.20	8.10/ 2.24	12.37/ 2.80	8.35/ 2.98
ISC	11.49/ 1.40	9.35/3.20	8.10/ 2.24	12.37/ 2.80	8.35/ 2.98

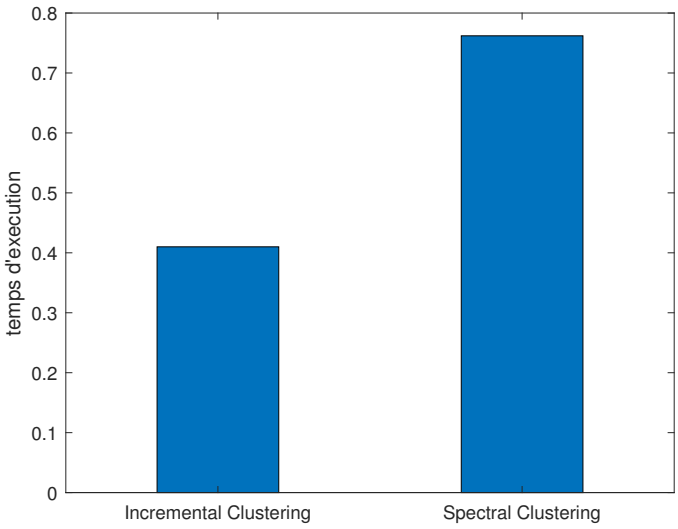


FIGURE 4.4 – Temps d'exécution des deux algorithmes de partitionnement spectral.

En outre, le temps d'exécution pour les deux méthodes est aussi comparé. La Figure 4.4 montre le temps d'exécution pour les deux méthodes de classifications par des bâtonnets. Ce temps quantifie le coût de calcul en seconde (s) appliqué à notre étude. L'algorithme

de partitionnement spectral apparaît comme la méthode la plus coûteuse. En effet, le calcul des valeurs et vecteurs propre est coûteux. À l'opposé, l'algorithme de partitionnement incremental apparaît moins coûteux. L'analyse des résultats expérimentaux montre que l'algorithme ISC est considéré plus adéquat à appliquer pour les données dynamiques. En effet, il réduit le temps d'exécution tout en conservant des bons résultats de partitionnement.

4.12 Conclusion

Ce chapitre présente une approche du partitionnement spectral incrémental pour traiter les graphes routiers dynamiques. En effet, les poids des arêtes du graphe peuvent évoluer. Le calcul des valeurs et vecteurs propres à chaque changement de similarité est très coûteux d'où la nécessité d'introduire l'algorithme ISC. Les changements des similarités des arêtes sont incorporés en les représentant par un vecteur/matrice d'incidence. L'algorithme ISC met à jour de manière efficace le système de valeurs propres lorsque l'ensemble de données évolue.

L'algorithme ISC est appliqué aux données du réseau routier de la ville de Calais. Afin de tester la performance des résultats obtenus, nous appliquons également l'algorithme de Ng et al. au graphe routier. Nous avons comparé les résultats de partitionnement des deux méthodes ainsi que le coût de calcul. L'algorithme ISC atteint une précision similaire dans le partitionnement, mais avec un coût de calcul plus faible.

Chapitre 5

Classification spectrale évolutive d'un réseau de transport

Sommaire

5.1 Introduction

Les algorithmes de classification spectrale évolutive sont introduits pour le partitionnement de graphes dynamiques i.e. évoluant dans le temps. Ces algorithmes ont été appliqués avec succès dans la littérature (Chi et al., 2007). Leurs performances surpassent celle des algorithmes de partitionnement statique en produisant des partitions qui peuvent s'adapter aux variations des graphes tout en étant robustes. Dans le cadre de la classification spectrale évolutive, un bon résultat de classification doit tenir compte du trafic actuel sans trop s'écarter de son évolution passée (Chi et al., 2007). Notamment préserver la qualité des classes obtenues aussi bien l'appartenance aux classes.

Le partitionnement dynamique du graphe consiste à calculer une matrice Laplacienne évolutive, à calculer l'espace propre engendré et à appliquer un simple algorithme de K-means pour le partitionnement du graphe. Dans ce chapitre, nous proposons une méthodologie basée sur la classification spectrale évolutive pour partitionner un réseau urbain comprenant les étapes suivante :

— Nous appliquons les algorithmes de classification spectrale évolutive pour partitionner le réseau de transport urbain en régions homogènes qui évoluent en douceur dans le temps. Le nombre de classes pour chaque période est détecté automatiquement par la méthode de la modularité qui est comparée à l'heuristique Eigengap dans le but de prouver son efficacité.

— Nous testons ensuite notre approche sur des ensembles de données réelles de trafic recueillies sur le réseau de transport de la ville d'Amsterdam (Lopez et al., 2017). L'évolution du graphe sur différentes périodes de la journée est étudiée à l'aide des méthodes évolutives : Preserving Cluster Quality (PCQ) et Preserving Cluster Membership (PCM). Nous montrons la supériorité des algorithmes proposés en termes de robustesse et d'efficacité par rapport à l'algorithme de classification statique.

— Dans une dernière expérience, nous appliquons l'algorithme de classification par recherche des pics de densités sur les graphes de trafic (Rodriguez and Laio, 2014) afin de détecter le nombre de classe. Ensuite, nous comparons ces résultats avec ceux obtenus par la modularité et Eigengap.

5.2 Classification spectrale évolutive

Les algorithmes de classification spectrale sont généralement appliqués sur des données statiques. Or, les données du trafic routier sont considérées comme un système dynamique.

Par conséquent, les données spatiales à classifier évoluent dans le temps, ce qui amènera à avoir un résultat de classification à chaque étape. Pour la prise en compte de cette dynamique les algorithmes statiques doivent être ajustés. Dans un premier temps, nous avons appliqué les algorithmes de classification spectrale incrémentale (Ning et al., 2010). Cette méthode réduit le temps de calcul et se concentre sur des techniques de bas niveau pour trouver un compromis entre précision et efficacité de partitionnement. En outre, lorsque les graphes de données évoluent de manière plus importante, les erreurs s'accumulent. Pour cela, à un certain moment, il est nécessaire de réinitialiser ces algorithmes pour éviter l'accumulation des erreurs. Une autre méthode est proposée dans la littérature pour la classification des données dynamiques est la classification évolutive qui a été proposée par Chakrabarti et al. (2006). La classification évolutive optimise simultanément deux critères : la classification doit s'adapter aux données actuelles autant que possible, tout en ne s'écartant pas trop du contexte historique. Chakrabarti et al. (2006) ont présenté un cadre générique pour ce problème. Ils ont proposé un algorithme de classification hiérarchique évolutive et un algorithme de K-menas évolutive. Chi et al. (2007) sont allés plus loin et ont proposé une méthode de classification spectrale évolutive en incorporant le lissage temporel. Les classes actuelles doivent dépendre des caractéristiques des données du trafic actuels en ne s'écartant pas trop des historiques des données précédentes (Chi et al., 2007). Dans ce contexte, nous proposons d'appliquer deux approches de classification spectrale évolutive pour le partitionnement des réseaux de transport dynamiques : la préservation de la qualité des classes (PCQ) et la préservation de l'appartenance aux classes (PCM). Dans ces deux approches, une fonction du coût total est définie comme une combinaison linéaire entre deux coûts : le coût instantané (SC) et le coût temporel (TC). Le coût total est exprimé par :

$$TotalCost = \alpha\, SC + (1 - \alpha)\, TC. \tag{5.1}$$

où $0 \leq \alpha \leq 1$ est un paramètre fixé par l'utilisateur. Le coût instantané (snapshot cost), mesure la qualité instantanée des résultats du partitionnement actuel par rapport aux caractéristiques du graphe actuelles. Ce coût est exprimé d'une manière similaire pour les deux approches évolutives. Par contre, le coût temporel (temporal cost), est exprimé différemment. La première approche PCQ, mesure la régularité temporelle en terme de compatibilité des résultats de partitionnement actuel par rapport aux caractéristiques historiques du graphe. La deuxième approche PCM, consiste à exprimer le coût temporel comme la différence entre la partition actuelle et la partition historique. Cette fonction de coût est définie afin d'adapter la régularité temporelle.

5.2.1 Préservation de la qualité des classes

Considérons un graphe $G_t = (\mathcal{V}_t, E_t)$ au temps t. Soient Z_t la partition du graphe au temps t et $NC_t|_{Z_t}$ la coupe normalisée au temps t évaluée par la partition Z_t d'une part et $NC_{(t-1)}|_{Z_t}$ la coupe normalisée au temps $(t-1)$ évaluée par la partition Z_t. Le coût total de la coupe normalisée évolutive peut être exprimé par (Chi et al., 2009) :

$$TotalCost = \alpha NC_t|_{Z_t} + (1-\alpha) NC_{(t-1)}|_{Z_t}, \tag{5.2}$$

notons que $Z \in \mathbb{R}^{\mathbb{N}\times\mathbb{K}}$ est une valeur continue relaxée avec $Z^T Z = I_K$ et I_K la matrice identité.

L'approche PCQ consiste à trouver la solution optimale qui minimise le coût total. Après quelques manipulations, on obtient le coût total suivant :

$$TotalCost = K - Tr\left[Z_t^T \left(\alpha \boldsymbol{L}_t + (1-\alpha)\boldsymbol{L}_{(t-1)}\right) Z_t\right], \tag{5.3}$$

avec K le nombre de classes et $\boldsymbol{L}_t = \boldsymbol{D}_t^{-1/2} \boldsymbol{W}_t \boldsymbol{D}_t^{-1/2}$.
On définit la matrice Laplacienne $\mathbf{L}_{pcq}$ par :

$$\boldsymbol{L}_{pcq} = \alpha \boldsymbol{L}_t + (1-\alpha)\boldsymbol{L}_{t-1}. \tag{5.4}$$

Ce problème peut être considéré comme un problème de maximisation de la trace de la Laplacienne $\mathbf{L}_{pcq}$ où la solution est la matrice Z_t. Les colonnes de cette matrice sont les K-vecteurs propres associés aux K-valeurs propres de cette matrice Laplacienne. Après avoir obtenu Z_t, il suffit d'appliquer l'algorithme K-means pour obtenir la partition du graphe.

5.2.2 Préservation de l'appartenance aux classes

Considérons un graphe $G_t = (\mathcal{V}_t, E_t)$ au temps t. Soient Z_t la partition du graphe au temps t et $NC_t|_{Z_t}$ la coupe normalisée au temps t évaluée par la partition Z_t d'une part et Z_{t-1} la partition du graphe $G_{t-1} = (\mathcal{V}_{t-1}, E_{t-1})$ au temps $(t-1)$. La méthode PCM consiste à trouver la solution optimale qui minimise le coût total (Chi et al., 2009) :

$$TotalCost = \alpha NC_t|_{Z_t} + (1-\alpha) TC, \tag{5.5}$$

Le coût temporel TC exprime une certaine distance entre la partition actuelle Z_t au temps t et la partition Z_{t-1} au temps $(t-1)$. De ce fait, le coût temporel est défini par :

$$TC = \left\| Z_t Z_t^T - Z_{t-1} Z_{t-1}^T \right\|^2, \tag{5.6}$$

où, Z_t et Z_{t-1} sont les K-vecteurs propres aux instants t et $(t-1)$ respectivement. Dans ce cas, le coût total évolutif s'exprime par :

$$\begin{aligned} TotalCost &= \alpha K - \alpha Tr\left[Z_t^T \left(\boldsymbol{D}_t^{-1/2} \boldsymbol{W}_t \boldsymbol{D}_t^{-1/2} \right) Z_t \right] \\ &\quad + (1-\alpha) \left\| Z_t Z_t^T - Z_{t-1} Z_{t-1}^T \right\|^2 \\ &= K - Tr\left[Z_t^T \left(\alpha \boldsymbol{D}_t^{-1/2} \boldsymbol{W}_t \boldsymbol{D}_t^{-1/2} + (1-\alpha) Z_{t-1} Z_{t-1}^T \right) Z_t \right]. \end{aligned} \tag{5.7}$$

Ce problème peut être considéré comme un problème de maximisation de la trace de la matrice Laplacienne évolutive $\boldsymbol{L}_{pcm}$ (Chi et al., 2009) :

$$\boldsymbol{L}_{pcm} = \alpha \boldsymbol{L}_t + (1-\alpha)(Z_{t-1} Z_{t-1}^T). \tag{5.8}$$

La solution est la matrice Z_t dont les colonnes sont les K-vecteurs propres associés aux K-valeurs propres de la matrice Laplacienne $\boldsymbol{L}_{pcm}$. Après avoir obtenu Z_t, il suffit d'appliquer l'algorithme K-means pour obtenir la partition du graphe.
Dans la suite nous allons appliqué la classification spectrale évolutive au réseau urbain.

5.3 Résultats expérimentaux

Dans cette section, nous analysons les données expérimentales traitées dans Lopez et al. (2017). La méthodologie choisie est de comparer les méthodes de classification statiques étudiées dans la section 3.4 du chapitre 3, plus particulièrement l'algorithme de Ng et al. 5 et les méthodes de classification évolutives présentées précédemment dans la section 5.2 et leur adaptation au graphe dynamique par la similarité des snakes. Tout d'abord, nous définissons la façon de convertir le réseau en un graphe de liens. Ensuite, nous convertissons les données du graphe en un graphe de snakes, par la suite, nous calculons la matrice de similarité entre les snakes afin d'appliquer les algorithmes de classification. Enfin, nous détectons automatiquement le nombre de classes et nous présentons une étude comparative entre les méthodes de classification afin de montrer la performance et l'efficacité de la méthode évolutive sur le réseau de transport.

5.3.1 Réseau de transport

Les données réelles ont été recueillies dans la ville d'Amsterdam, et le réseau exploitable comprend 7 512 liens (Lopez, 2017). La grande taille du réseau augmente la complexité en temps de calcul des algorithmes de partitionnement. Pour cela, le réseau a été réduit à 208 liens comme illustrer dans la figure 5.1. Notre étude se base sur un seul jour de données, un jour ouvrable, et les vitesses routières sont estimées à partir des temps de parcours individuels. En outre, l'information sur la vitesse moyenne est disponible durant des périodes de 10 minutes de 7 heures du matin à 15 heures de l'après-midi. Au final, la base de données est constituée des vitesses effectives d'une journée du réseau d'Amsterdam de 208 liens sur 48 périodes de 10 minutes. La vitesse maximale pratiquée est de 40 m/s.

5.3.2 Modèles de graphes

À partir de ces données, nous construisons le réseau routier (réseau des liens) et expliquons les représentations possibles du réseau en un graphe des liens et un graphe des snakes afin d'appliquer les algorithmes de classification spectrale.

FIGURE 5.1 – Réseau d'Amsterdam avec 208 liens (Lopez, 2017).

5.3.3 Réseau Routier

Le réseau routier est défini par $\aleph = (\mathcal{C}, R)$ comprenant un ensemble de points d'intersection $\mathcal{C} = \{c_1, c_2, \ldots c_{n_I}\}$ qui sont reliés entre eux par un ensemble des liens (ou routes)

$R = \{r_1, r_2, \ldots r_{n_R}\}$, où pour chaque lien r_i, une valeur de vitesse lui est attribué. Après avoir établi le réseau routier, nous pouvons construire le graphe des liens.

5.3.4 Graphe des liens

Dans le but de construire le graphe des liens associé au réseau routier, nous distinguons deux représentations possibles : le graphe primal et le graphe dual. Nous avons détaillé ces approches dans le chapitre 1 section 1.4.1 par deux exemples pédagogiques illustrés par les figures 1.4 et 1.5. Dans la première représentation, le graphe primal est construit en considérant chaque intersection (ou croisement) comme étant un nœud et en ajoutant une arête entre chaque paire de nœuds s'il existe au moins un lien les reliant. Dans cette représentation, nous conservons la même structure du réseau routier ; les arêtes dans le graphe suivent les chemins du réseau réel. Dans la deuxième représentation, le graphe dual est construit en considérant chaque lien du réseau comme un nœud et en établissant une arête entre chaque paire de nœuds s'il existe au moins un croisement commun entre eux.
Dans ce travail, nous utilisons le graphe dual, car il s'agit d'une représentation plus adapté pour partitionner un réseau de transport (Lopez et al., 2017).

Nous modélisons les données du réseau par un graphe $G = (\mathcal{V}, E)$ dans lequel $\mathcal{V}$ est l'ensemble des noeuds représentant les liens du réseau et E l'ensemble des arêtes représentant les croisements. Deux liens du réseau sont connectés dans l'espace si leur extrémité ou leur début sont reliés à la même intersection. Sur la base de cette définition, tous les liens qui entrent ou sortent de la même intersection sont supposés être connectés. De plus, les liens se connectant à la même intersection sont considérés adjacents. Afin de partitionner le graphe des liens en états de trafic homogènes connectés, nous construisons le graphe des snakes.

5.3.5 Graphe des snakes

Le graphe des snakes est représenté par un graphe non orienté $G = (\Gamma, E)$. L'ensemble des noeuds $\Gamma = \{T_1, T_2, \ldots, T_N\}$ où $N = 208$, représente l'ensemble des snakes du réseau et l'ensemble des arêtes E pondérés par la similarité entre les noeuds-snakes.
Saeedmanesh and Geroliminis (2016), Lopez et al. (2017) ont utilisés la notion de "snakes" pour caractériser la densité du trafic routier. Un snake est une concaténation de L liens adjacents ayant des vitesses proches. Dans un graphe, les snakes sont des chemins ayant le même nombre de liens. Chaque snake est déterminé par un processus itératif démarrant à partir d'un lien individuel du réseau. Un snake est défini par $T_i = \{t_{i1}, \ldots, t_{il}, \ldots, t_{iL}\}$, où $1 \leq l \leq L$, avec $L \leq N$ la taille du snake. Chaque lien noté $t_{il} = \{id_l, v_l\}$ est composé à partir d'un identifiant id_l et d'une vitesse v_l. Un snake est initialisé sur un lien unique du ré-

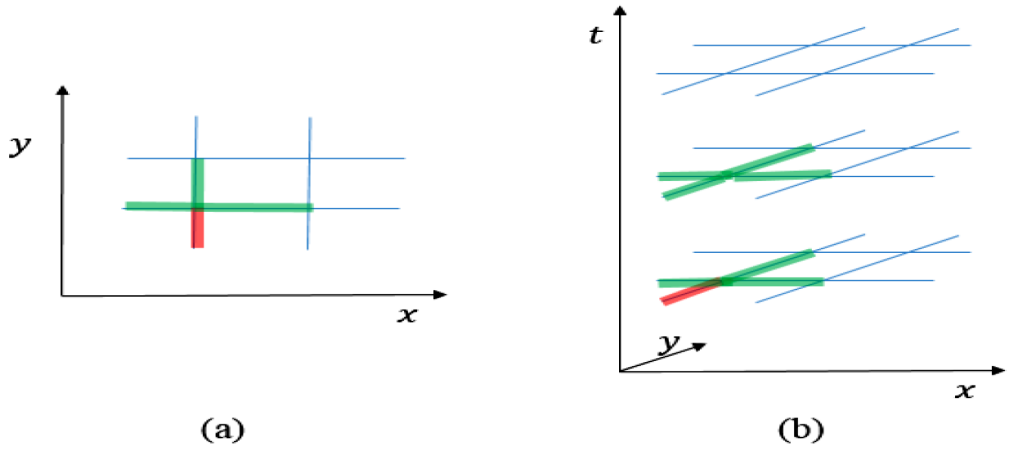


FIGURE 5.2 – Exemple d'un snake, initialisé par un lien en rouge. Les liens verts représentent le voisinage, (a) selon une approche 2D et (b) une approche 3D (Lopez, 2017).

seau, et à chaque étape le lien adjacent est ajouté en commençant par celui ayant les vitesses les plus proches du sien, et ce jusqu'à contenir tous les liens du réseau. À chaque étape, le lien adjacent minimisant la variance du snake est ajouté. La figure 5.2 montre un exemple de snake, initialisé par le lien rouge. Les liens verts représentent le voisinage, selon des approches à deux et trois dimensions respectivement. Le nombre de snakes à créer dépend du nombre de liens dans un réseau. Au total, il y a autant de snakes que les liens dans le réseau routier. Pour chaque étape l, les valeurs de la vitesse moyenne et la variance du snake sont données par :

$$\sigma_l^2 = \sigma_{l-1}^2 + \frac{1}{l}\left[(v_l - \overline{v}_l)(v_l - \overline{v}_{l-1}) - \sigma_{l-1}^2\right], \tag{5.9}$$

où,

$$\overline{v}_l = \overline{v}_{l-1} + \frac{1}{l}(v_l - \overline{v}_{l-1}). \tag{5.10}$$

$\overline{v}_l$ représente la vitesse moyenne du snake de taille l, et v_l la vitesse du lien ajouté lors de la $l$ème étape.

Partitionner les liens du réseau requiert la construction d'un graphe pondéré. Ces graphes sont représentés par une matrice de similarité entre les points de données, dans notre cas, les snakes. Dans la section suivante, nous présentons la méthode de construction de la matrice de similarité basée sur les snakes aussi appelée "snake similarity".

5.3.6 Similarité des snakes

L'adaptation de la coupe normalisée au cas d'un réseau de transport repose sur la matrice de similarité d'entrée qui est nécessaire au calcul de la matrice Laplacienne normalisée. Une discussion sur les différentes méthodes de construction des matrices de similarités a été

réalisée dans la section 3.2.4. Dans notre travail, nous nous intéressons à définir une similarité entre chaque paire de liens dans le réseau de transport qui prend en compte à la fois l'homogénéité et la connectivité spatiale. Pour cela, nous adoptons la matrice de similarité des snakes. La matrice de similarité entre chaque paire de noeuds-snakes est calculée par :

$$\begin{cases} w_{ij} = \frac{1}{\sum_{l=1}^{L} l\phi^l} \sum_{l=1}^{L} \phi^l card(T_i[l] \cap T_j[l], & i,j = 1,\ldots,N. \\ w_{ii} = 0. \end{cases} \tag{5.11}$$

Notons que :

$$\sum_{l=1}^{L} l\phi^l = \frac{\phi(L\phi^{(L+1)} - (L+1)\phi^L + 1)}{(1-\phi)^2} \approx \frac{\phi}{(1-\phi)^2}, \tag{5.12}$$

où $T_i[l]$ et $T_j[l]$ sont deux snakes de taille l correspondant aux liens de départ r_i et r_j, et $card(T_i[l] \cap T_j[l])$ est le nombre d'identifiants de liens communs entre les deux snakes de taille l. Le coefficient de pondération ϕ est fixé par l'utilisateur, avec $0 < \phi \leq 1$. Cette formulation de la matrice de similarité, dépend de la taille des snakes et du coefficient de pondération ϕ.

Cette mesure de similarité accorde moins de poids aux liens ajoutés lorsque la taille du snake augmente et les éléments à la fin des snakes ne sont pas considérés. Après un certain nombre d'itérations, le snake concatène des segments de vitesses trop différentes entraînant une variance plus élevée. Pour cela, il est important d'optimiser la taille des snakes pour réduire le coût de calcul. Pour approfondir ce point, nous traçons la variance du premier et du second snake du graphe des snakes. La figure 5.3 représente deux évolutions de la variance par rapport à la taille des snakes où la croissance de ses valeurs est influencée par l'ajout des liens non similaires. Sur la base de ses propriétés, nous proposons une pondération plus importante pour les liens qui sont spatialement proches les uns des autres (Lopez et al., 2017).

5.3.7 Optimisation de la longueur des trajectoires et du coefficient de pondération

Une étude de sensibilité a été réalisée afin d'étudier les effets de la longueur des snakes sur les performances de partitionnement. La longueur des snakes doit être fixée à un seuil minimal pour conserver la connectivité. De ce fait, un snake trop court ne permet pas de détecter l'intégralité de la topologie du réseau dans l'espace et dans le temps. Une longueur du snake courte peut également donner des résultats de partitionnement dans lesquels une classe contient des liens qui ne sont pas tous connectés entre eux. La dissimilarité entre les classes (CCD) est utilisée pour évaluer les classes résultantes du partitionnement par l'algo-

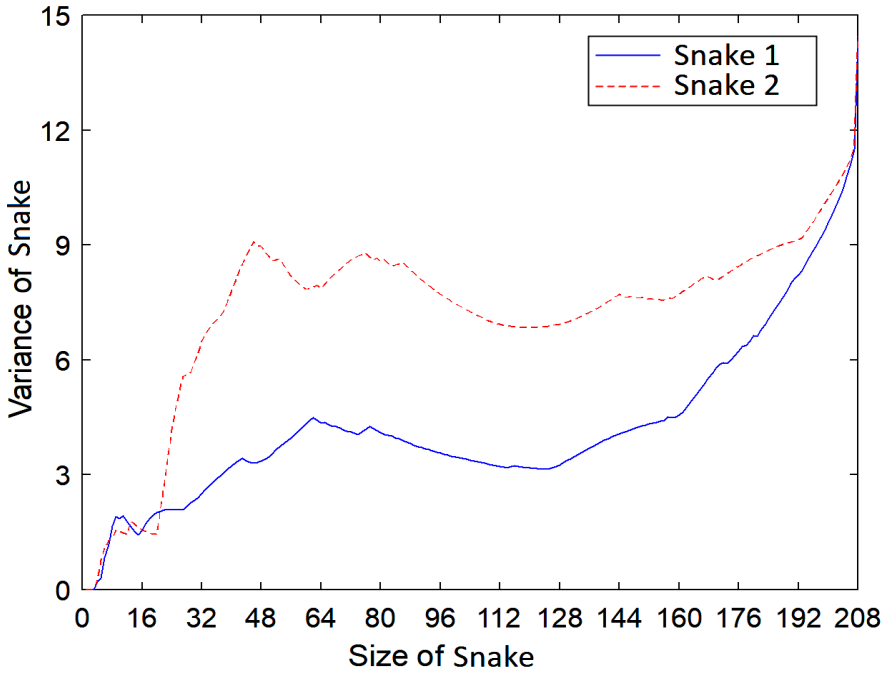


FIGURE 5.3 – Exemple de la croissance de la variance de deux snakes du graphe des liens.

rithme de Ng et al. adopté pour la méthode de classification statique d'un réseau de transport. Nous avons effectué cette étude sur différentes longueurs de snakes afin de choisir la longueur optimal. Le CDD est un indicateur qui mesure la dissimilarité inter-clusters. Cette dissimilarité est mesurée par la différence entre la vitesse moyenne d'une classe donnée et ses classes voisines. Cet indicateur est défini par l'équation suivante (Lopez et al., 2017) :

$$CCD = \frac{\sum_{i=1}^{K}\sum_{k=1+i}^{K}\delta_{ik}|\overline{v_i} - \overline{v_k}|}{\sum_{i=1}^{K}\sum_{k=1+i}^{K}\delta_{ik}}, \tag{5.13}$$

où K est le nombre de classes, $\delta_{ik} = 1$ si i et k sont les indices de classes connectées, et $\delta_{ik} = 0$ sinon. Cette mesure est à maximiser : une valeur élevée de CCD indique une amélioration obtenue par l'algorithme de partitionnement. Les valeurs moyennes de la CCD pour les 48 périodes ont été calculés avec différentes longueurs des snakes, couvrant différentes proportions de la taille du réseau routier. La figure 5.4 montre les résultats de l'évaluation des performances de partitionnement par l'indicateur CCD. L'algorithme produit des résultats similaires pour des longueurs des snakes supérieures à 40% de la taille du réseau. Par conséquent, la longueur des snakes est fixée à 40%. De plus, le coefficient de pondération ϕ a été choisi égal à 0.8 qui permet d'obtenir des valeurs moyennes maximales pour le CCD (figure 5.5).

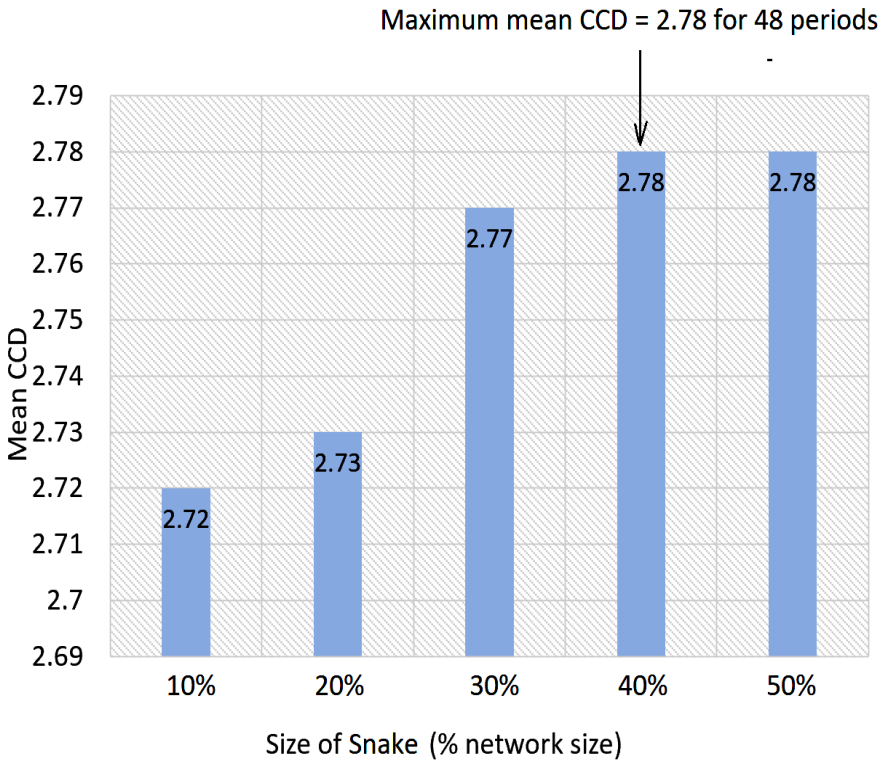


FIGURE 5.4 – Valeurs moyennes de la dissimilarité inter-clusters en fonction de la longueur du snake calculée en pourcentage de la taille du réseau routier.

5.3.8 Sélection automatique du nombre de classes

Le réseau de transport se caractérise par des données dynamiques qui sont obtenues à des périodes temporelles définies. De ce fait, le nombre de classes peut varier selon les niveaux de congestion et d'hétérogénéité à chaque période. Le nombre de classes K au temps t peut être différent de celui de K' au temps $(t-1)$. Dans notre étude, l'approche développée est conçue pour avoir la capacité de détecter le nombre de classes entre les périodes. Le choix du nombre de classes optimales des algorithmes de classification spectrales est un problème sensible. Dans une première expérience, nous avons considéré un nombre de classes fixé à $K = 2$ pour toutes les périodes, afin d'identifier deux états de congestion et d'étudier le comportement des algorithmes évolutifs d'une manière simplifiée (Al Alam et al., 2020). Par conséquent, considérer un nombre de classes fixe pour toutes les périodes entraîne une forte restriction. Dans cette section, nous présentons une étude comparative entre les méthodes de détection automatique : l'heuristique Eigengap (Von Luxburg, 2007) et le critère de modularité (El Mahrsi and Rossi, 2012), présentées précédemment dans le chapitre 3, section 3.4.1. La figure 5.6a montre la variation des valeurs propres pour une période de temps donnée $t = 5$. Les résultats montrent un gap entre la quatrième et la cinquième valeur propre, ce

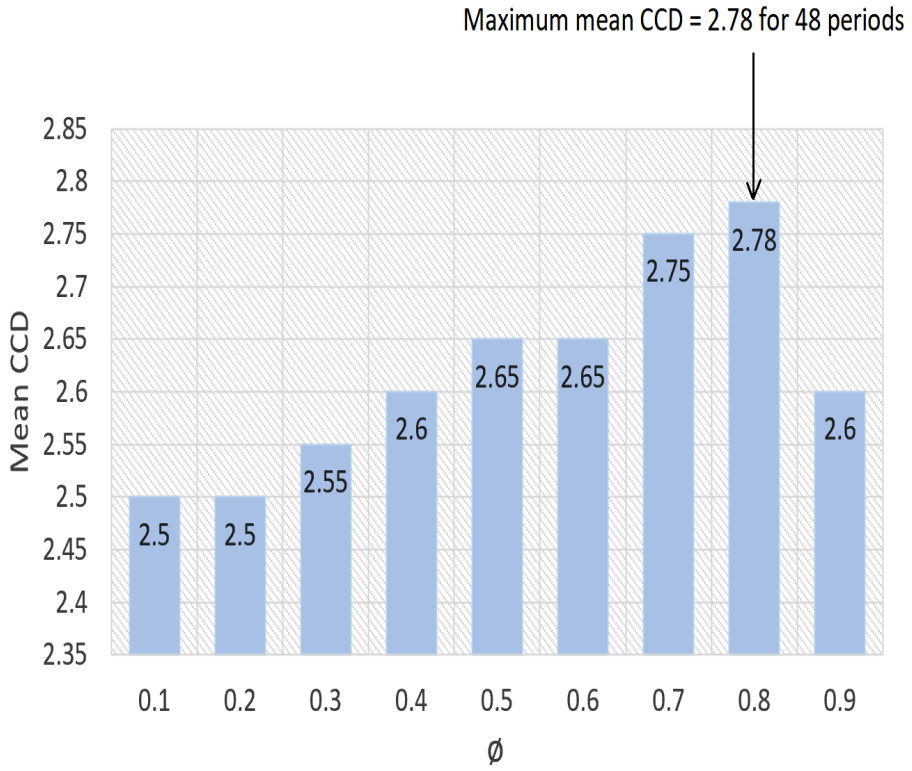
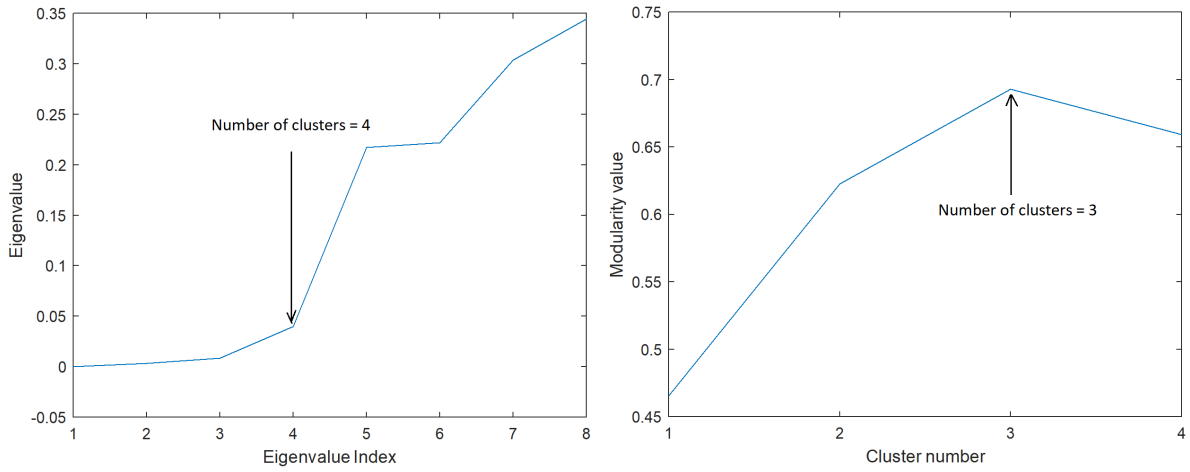


FIGURE 5.5 – Valeurs moyennes de la dissimilarité inter-clusters en fonction du coefficient de pondération ϕ.

qui indique que le nombre de classes K est égal à 4 pour cette période. La figure 5.6b montre la variation des valeurs de la modularité en fonction du nombre de classes pour la même période $t = 5$. Afin d'estimer le nombre de classes finales, nous recherchons la première valeur maximisant la modularité. Dans cette méthode, nous trouvons un nombre de classes $K = 3$.

Par la suite, nous avons appliqué les deux méthodes de sélection du nombre de classes sur les 48 périodes. La figure 5.7 représente le nombre de classes obtenues en appliquant les méthodes de modularité et Eigengap pour chaque période. D'après ces résultats, nous pouvons constater que le nombre de classes obtenues par la méthode de modularité est généralement plus petit que celui de l'Eigengap. Le choix d'un nombre de classes plus petit révèle plus facile à interpréter. De plus, un nombre petit de classes permettent de concevoir des stratégies de contrôle simples (Ji and Geroliminis, 2012b).

En outre, les résultats du partitionnement sont analysés en appliquant la classification spectrale en utilisant les deux méthodes pour la détection du nombre de classes, la modularité et Eigengap. Nous considérons la période $t = 5$ afin de comparer les résultats sur différents nombres de classes. Nous traçons les résultats du partitionnement du réseau d'Amsterdam (figure 5.8) et nous calculons les vitesses moyennes et les écarts-types pour chaque classe pour les deux méthodes. Le tableau 5.1, représente les résultats obtenus pour les valeurs



(a) (b)

FIGURE 5.6 – (a) Variation des valeurs propres pour une période de temps $t = 5$ (b) Variation des valeurs de la modularité en fonction du nombre de classes pour une période de temps $t = 5$.

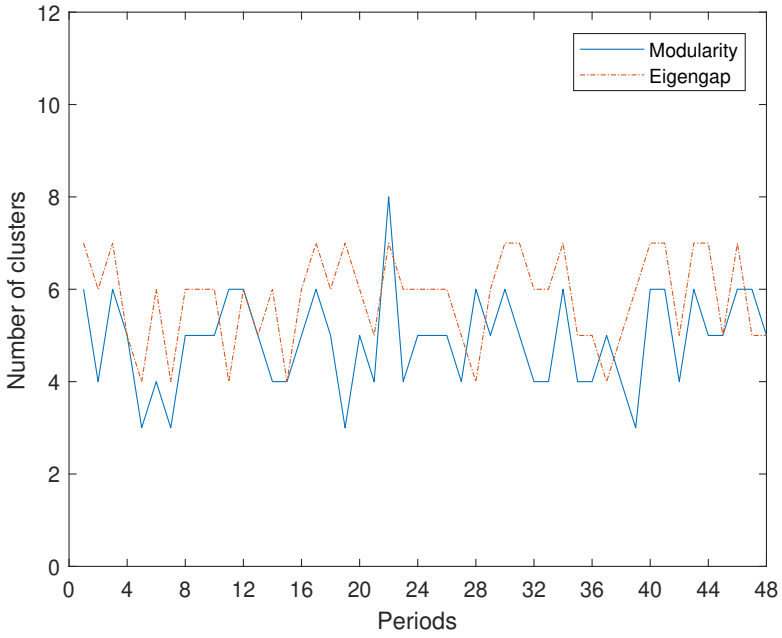


FIGURE 5.7 – Nombre de classes obtenues en appliquant les méthodes de la modularité et Eigengap pour chaque période.

des vitesses moyennes et les écarts-types. D'après ces résultats, on remarque que les classes obtenues ont des valeurs de vitesses moyennes dissimilaires permettant de distinguer les différentes zones de congestion. En plus, les petites valeurs des écarts-types démontrent

une performance efficace dans la séparation des classes homogènes connectées.
Ensuite, nous comparons les résultats du partitionnement des deux méthodes en utilisant

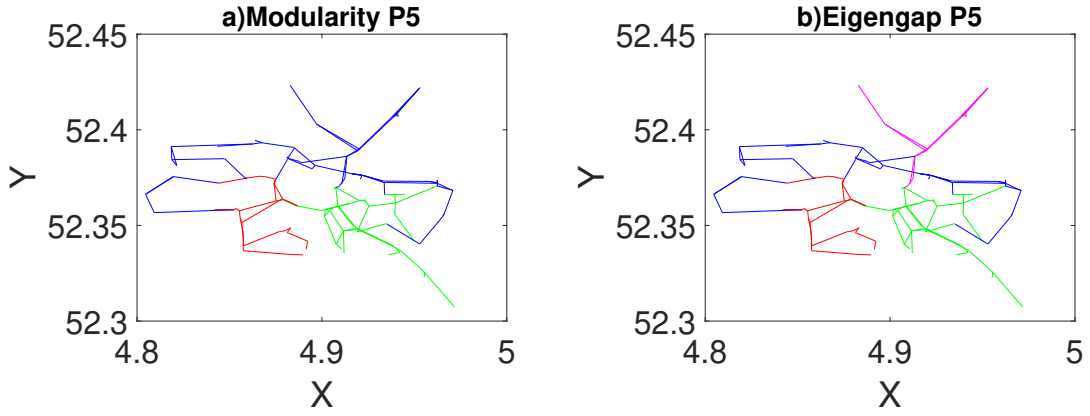


FIGURE 5.8 – Résultats du partitionnement pour la période $t = 5$ dans le cas de la modularité et Eigengap.

TABLEAU 5.1 – Valeurs de vitesse moyenne (m/s) et écart-type pour une période de temps $t = 5$ avec un nombre de classes $K = 3$ dans le cas de la modularité et $K = 4$ dans le cas d'Eigengap.

(μ/σ)	Rouge	Vert	Bleu	Magenta
Modularité	4.95/ 1.07	9.74/ 2.06	11.92/ 3.57	-
Eigengap	4.95/ 1.07	9.74/ 2.06	11.05/ 3.52	14.40/ 2.37

la matrice de confusion pour la période $t = 5$ (Meilă, 2007). Cette matrice permet d'évaluer la qualité de la partition obtenue en appliquant la méthode de l'heuristique Eigengap par comparaison avec celle de la modularité. La matrice de confusion, aussi appelée tableau de contingence est une matrice $K * K'$, dont l'élément kk'^{eme} est le nombre de points à l'intersection de C_k la partition de la modularité et $C'_{k'}$ celle de l'Eigengap. Les résultats sont représentés par le tableau 5.2. Nous remarquons qu'en fusionnant les classes magenta et bleu dans la méthode Eigengap, on obtient la même partition que celle de la modularité.
Afin de tester la ressemblance entre les résultats du partitionnement spectral obtenus avec le nombre de classes fourni par la modularité et l'Eigengap, nous utilisons l'Information Mutuelle Normalisée (NMI). La NMI est une quantité mesurant la dépendance entre deux

TABLEAU 5.2 – Matrice de confusion pour les méthodes de la modularité et Eigengap pour la période $t = 5$.

(Modularité/Eigengap)	Rouge	Vert	Bleu	Magenta
Rouge	39	0	0	0
Vert	0	73	0	0
Bleu	0	0	71	25

classes. Elle est définie par (Meilă, 2007) :

$$NMI(C,C') = \frac{MI(C,C')}{\sqrt{H(C)H(C')}}, \tag{5.14}$$

où, $MI(C,C') = \sum_{i=1}^{K}\sum_{j=1}^{K'} \frac{n_{ij}}{N} log\left(\frac{n_{ij}N}{|C_i||C'_j|}\right)$ est l'information mutuelle entre les partitions C et C', n_{ij} représente le nombre de liens partagés entre les classes C_i et C'_j, K' est le nombre de classe dans la partition C', $H(C) = -\sum_{i=1}^{K} \frac{|C_i|}{N} log\left(\frac{|C_i|}{N}\right)$ est l'entropie de la partition C et $H(C') = -\sum_{j=1}^{K'} \frac{|C'_j|}{N} log\left(\frac{|C'_j|}{N}\right)$ est l'entropie de la partition C'. Une valeur plus élevée de NMI implique que des partitions similaires sont obtenues par les algorithmes de classification. La valeur NMI est comprise dans l'intervalle [0,1], et vaut 1 lorsque les deux partitions sont identiques, et 0 lorsqu'elles sont différentes.

La figure 5.9 montre la variation des valeurs du NMI pour chaque période. Les valeurs se situent entre 0.7 et 1 ce qui implique une ressemblance entre les résultats du partitionnement effectués par la méthode de la modularité et celle d'Eigengap.

Par la suite, nous comparons les performances du partitionnement statique et évolutif en utilisant les deux méthodes. Dans ce but, nous appliquons l'algorithme de classification spectrale normalisée IND, et les deux approches évolutives PCQ et PCM en utilisant les méthodes de la modularité et Eigengap. Les performances sont évaluées en calculant les coûts instantanés, temporels et totaux pour chacune des méthodes. Le tableau 5.3 montre les résultats obtenus : les meilleurs résultats sont représentés en gras. Le partitionnement en utilisant la modularité assure les meilleurs résultats en minimisant les différentes mesures de coûts. Pour vérifier ces résultats sur toutes les périodes, nous traçons les valeurs du coût total pour les 48 périodes. La figure 5.10 montre la variation du coût total pour l'approche PCQ dans le cas des méthodes de la modularité et Eigengap. Il est clair que la modularité assure les meilleurs résultats en minimisant le coût total. À noter que les résultats de l'approche PCM sont similaires à ceux de PCQ.

D'après les résultats apportés dans cette expérience et en considérant le lissage temporel, la

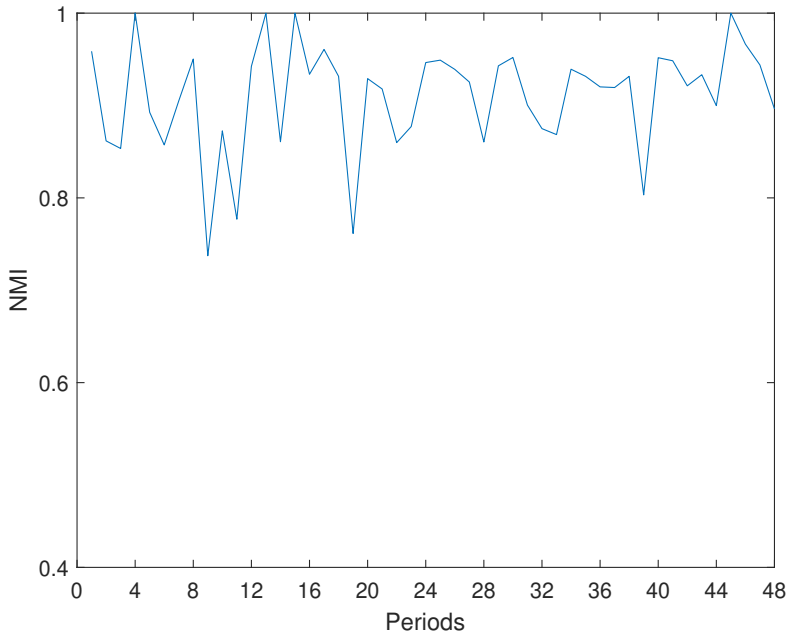


FIGURE 5.9 – Variation de l'NMI pour chaque période.

TABLEAU 5.3 – Performance du partitionnement des méthodes de la modularité et Eigengap.

Methode	Coût	IND	PCQ	PCM
Modularité	SC	**0.16**	0.20	0.21
	TC	0.45	**0.27**	0.30
	Total Cost	0.28	**0.23**	0.25
Eigengap	SC	0.28	0.34	0.37
	TC	0.68	0.46	0.46
	Total Cost	0.44	0.39	0.40

méthode de la modularité est préférée, car elle est plus cohérente avec son évolution passée et minimise le coût total pour les deux approches PCQ et PCM.

5.3.9 Étude comparative entre les algorithmes de classification

Dans cette section, nous comparons les deux approches de la classification spectrale évolutive avec l'algorithme de classification spectrale statique IND Ng et al. (2002). Ce dernier applique le partitionnement sur les snakes à une période de temps t et ignore toutes les données historiques. Le nombre de classes pour les deux approches de classification spectrale évolutive et la méthode IND sont automatiquement détectés en utilisant la méthode de la

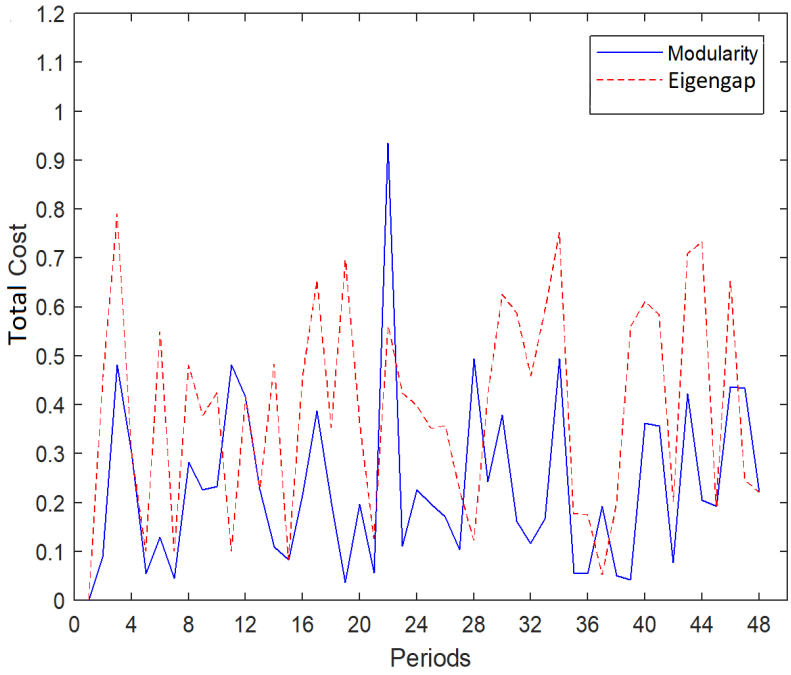
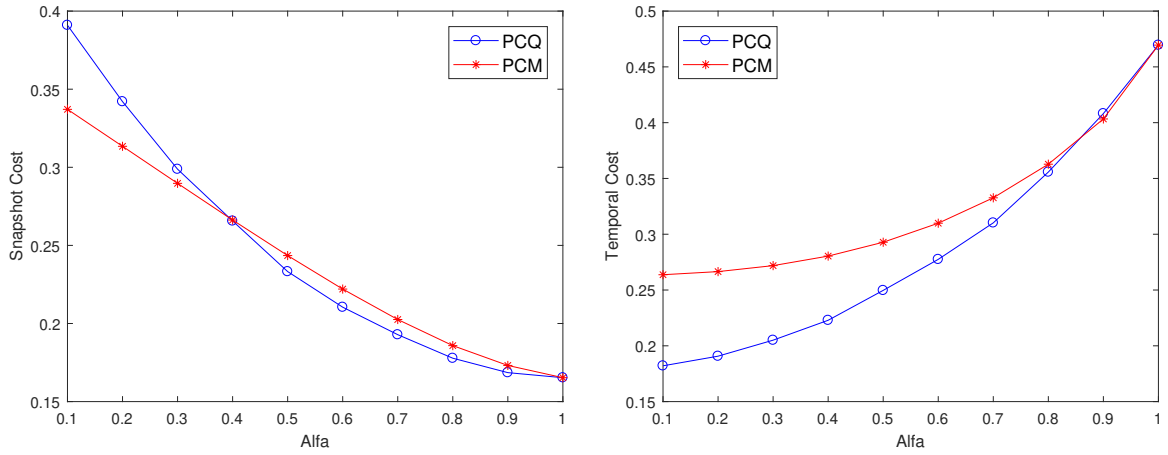


FIGURE 5.10 – Variation du coût total pour l'approche PCQ en utilisant les deux méthodes : modularité et Eigengap.

modularité.

Dans la première expérience, nous traçons les résultats de la variation entre le coût instantané et le coût temporel. Ces résultats montrent que les deux approches évolutives, PCQ et PCM, sont capables de contrôler l'échange entre ces deux coûts. Les figures 5.11a et 5.11b présentent les valeurs moyennes des coûts instantanés et temporels pour toutes les périodes de $t = 1$ à $t = 48$ sous différentes valeurs du paramètre α. Nous modifions la valeur de α de 0.1 à 1 avec un pas de 0.1. Lorsque la valeur de α augmente, nous obtenons une meilleure qualité pour le coût instantané au prix d'une mauvaise régularité temporelle. Dans les prochaines expériences, nous avons choisi une valeur de $\alpha = 0.6$ afin de mettre en évidence le coût instantané tout en considérant en même temps le coût temporel.

Dans la deuxième expérience, nous considérons toutes les périodes de $t = 1$ à $t = 48$ et nous évaluons les valeurs de coûts pour les trois méthodes. Pour tous les coûts, une valeur inférieure implique de meilleurs résultats. Les deux algorithmes de classification spectrale évolutive peuvent gérer la variation du nombre de classes entre les périodes au cours du temps (Chi et al., 2007). Les figures 5.12-5.14 montrent les résultats du coût instantané (SC), du coût temporel (TC) et du coût total pour la méthode IND et les approches PCQ et PCM. D'après ces résultats, nous remarquons que le cout instantané est plus faible pour la méthode IND que celles des approches évolutives PCQ et PCM. Cependant, les PCQ et PCM



(a) (b)

FIGURE 5.11 – Coût moyen (a) instantané et (b) temporel sous différentes valeurs du paramètre α.

assurent un coût temporel plus faible. De plus, ils sont plus performants en terme de minimisation du coût total.

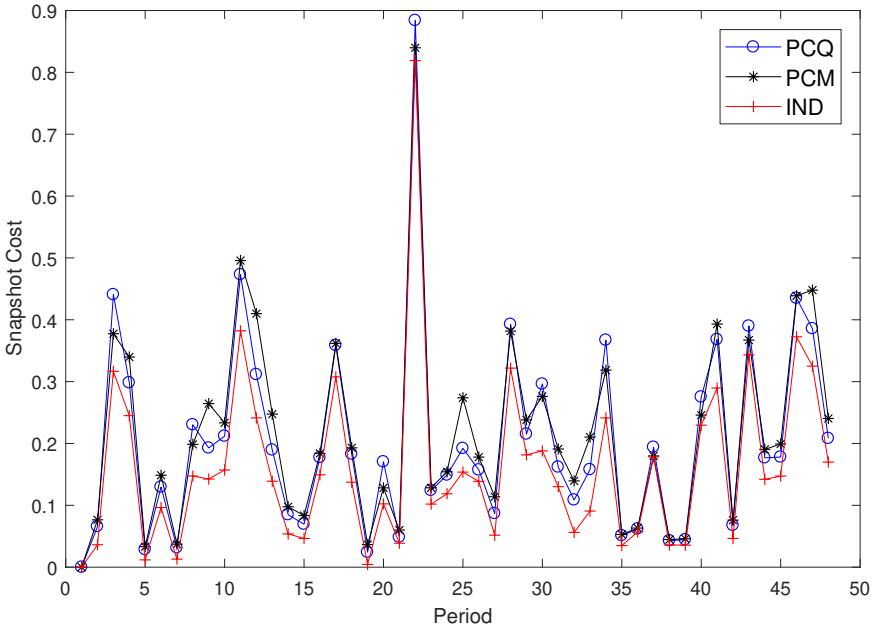


FIGURE 5.12 – Variation du coût instantané pour chaque période de temps.

Dans la suite, nous évaluons les vitesses moyennes et les écarts-types de chaque classe pour la période $t = 4$ dans le but de valider l'efficacité des approches évolutives dans la

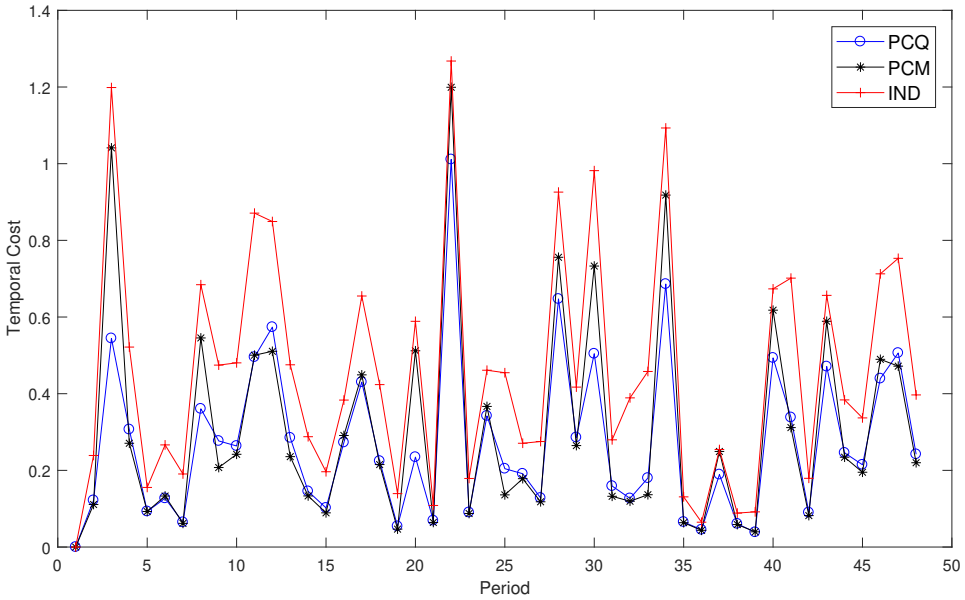


FIGURE 5.13 – Variation du coût temporel pour chaque période de temps.

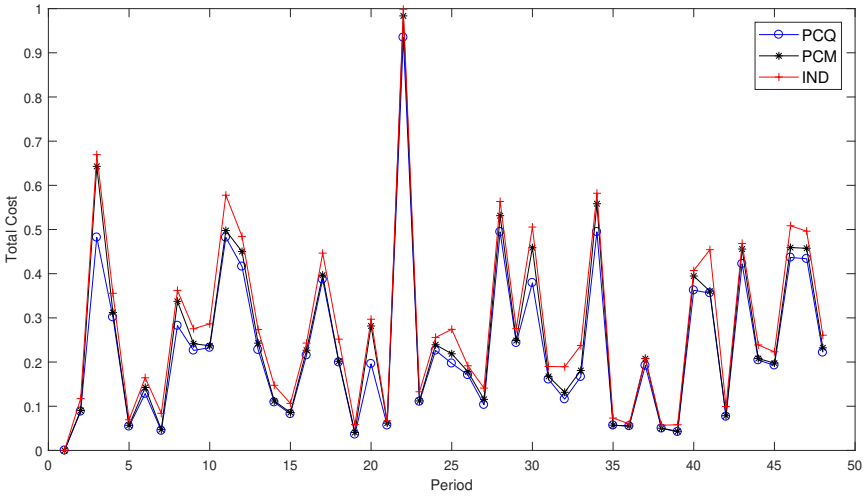


FIGURE 5.14 – Variation du coût total pour chaque période de temps.

séparation des classes homogènes connectées. Les résultats des mesures d'évaluation sont présentés en détail dans le tableau 5.4. En comparant les valeurs moyennes des vitesses des liens et les écarts-types dans les différentes classes, nous remarquons que les vitesses moyennes entre les partitions sont dissimilaires. De plus, la faible valeur de l'écart-type pour chaque classe définit une bonne homogénéité intra-clusters.

En outre, nous nous intéressons à valider les performances des algorithmes de la clas-

sification spectrale évolutive dans la détermination des classes qui correspondent aux données actuelles sans trop s'écarter de son évolution passée. Pour cela, nous présentons dans cette expérience les résultats du partitionnement du réseau d'Amsterdam pour les méthodes PCQ, PCM et IND. Afin de faire correspondre les classes entre les périodes consécutives, nous avons utilisé l'algorithme hongrois qui effectue une correspondance de classes une-à-une basée sur des facteurs de pondération entre les partitionnements consécutifs. Ces pondérations représentent le nombre de liens communs entre les classes (Hong et al., 2016; Xu et al., 2015). Cela permet de suivre avec précision l'évolution des classes dans le temps. La figure 5.15 illustre les résultats du partitionnement avec le nombre de classes détectées automatiquement par la méthode de la modularité. Les périodes analysées varient de $t = 2$ à $t = 5$, ce qui correspond aux heures de la journée de $7:10 - 7:40$ a.m., une des heures de pointe d'un jour ouvrable. La couleur en gras représente les liens pour lesquels un changement d'une classe à l'autre est spécifié. Pour les périodes de $t = 2$ à $t = 5$, nous constatons que pour la méthode IND, la variation du nombre des liens dans une classe change rapidement. Cependant, les approches PCM et PCQ sont plus stables lorsqu'on les compare aux partitions historiques. De cette façon, nous avons pu démontrer que les approches PCQ et PCM assurent une meilleure performance et fournissent des classes plus stables et cohérentes que la méthode IND.

Dans une nouvelle expérience, nous considérons des périodes de temps de 20 minutes, qui est une répétition d'un même réseau à deux couches temporelles. De cette façon, nous dupliquons les liens du réseau aux instants t et $(t-1)$. Le nombre de périodes totales est égal à $t = 24$. Dans ce cas, nous considérons le voisinage d'un lien à la fois dans l'espace et dans le temps. Une étude de sensibilité est aussi réalisée dans le cas d'un réseau spatio-temporel pour étudier le calage de la longueur des snakes. La figure 5.16 montre que l'algorithme du partitionnement produit des résultats similaires pour une longueur de snake supérieure à

TABLEAU 5.4 – Valeurs de la vitesse moyenne (m/s) et écarts-types pour la période $t = 4$ dans le cas des approches IND, PCQ et PCM.

(μ/σ)	Classe 1	Classe 2	Classe 3	Classe 4	Classe 5
Color IND	Rouge 6.02/ 1.35	Vert 8.52/2.02	Cyan 10.97/ 3.82	Jaune 11.58/ 1.59	Olive 14.81/ 2.45
Color PCQ	Rouge 6.21/ 1.39	Vert 8.62/2.60	Cyan 10.78/ 1.91	Jaune 10.97/ 3.82	Marron 14.81/ 2.45
Color PCM	Rouge 6.23/ 1.38	Vert 7.86/1.60	Cyan 10.85/ 1.85	Jaune 11.10/ 3.80	Olive 14.81/ 2.45

30% de la taille spatio-temporel du réseau. Par conséquent, la longueur du snake a été fixée à 30% et nous avons choisi $\phi = 0.8$ afin d'assurer les valeurs moyennes maximales de CCD (figure 5.17).

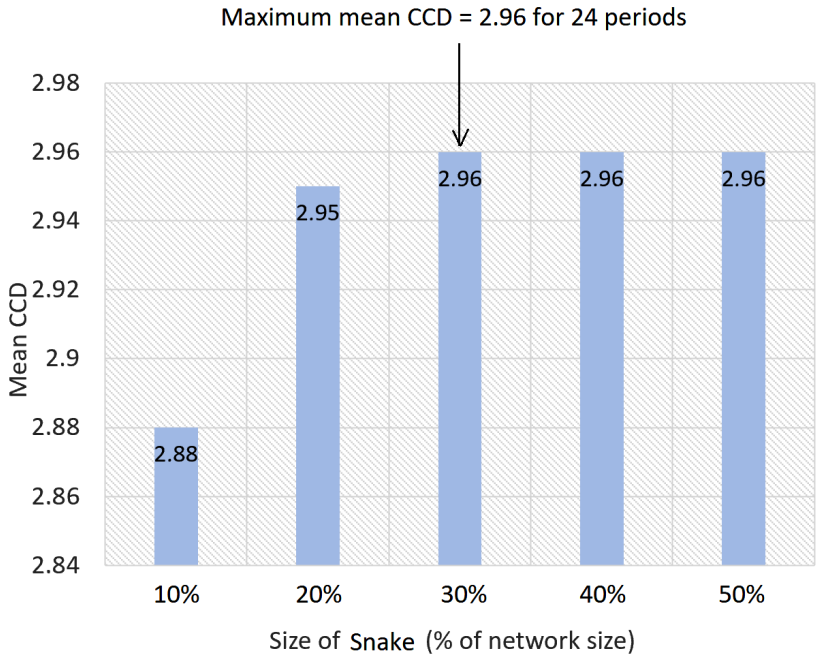


FIGURE 5.16 – Valeurs moyennes de CCD en fonction de la longueur du snake calculée en pourcentage de la taille du réseau des liens dans le cas d'une période de temps = 20 minutes.

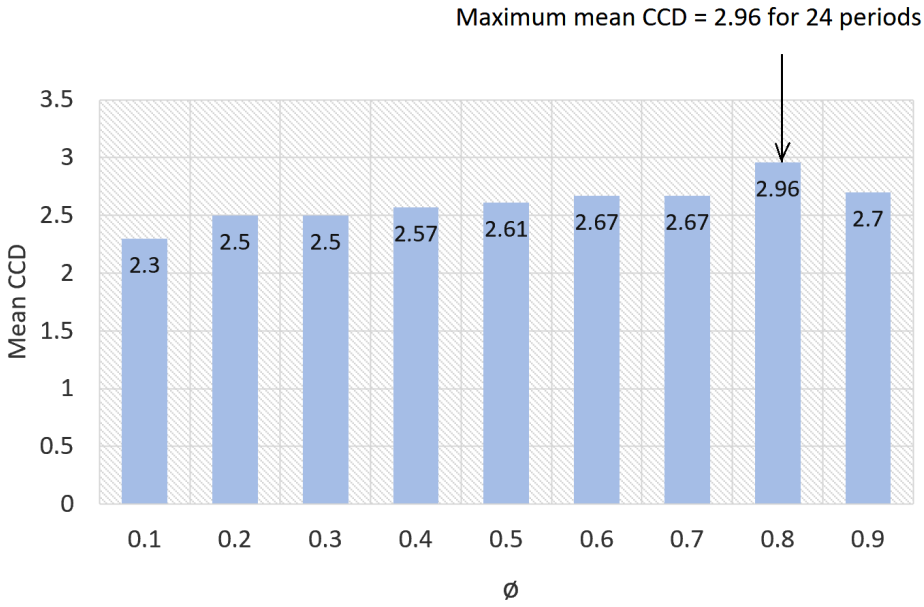


FIGURE 5.17 – Valeurs moyennes de CCD en fonction du coefficient de pondération ϕ dans le cas d'une période de temps = 20 minutes.

La figure 5.18 montre le nombre de classes obtenu en appliquant la méthode de modularité pour les périodes $t = 1$ à $t = 24$. Les figures 5.19-5.21 montrent les résultats du coût instantané, du coût temporel et du coût total pour les méthodes IND, PCQ et PCM, respectivement. Il est clair que les approches de classification évolutive spectrale fournissent les meilleures performances en minimisant le coût temporel et le coût total.

La figure 5.22 montre les résultats du partitionnement pour les périodes de temps $t = 2$ à $t = 4$. On peut constater que l'évolution des partitions entre les périodes de l'algorithme PCQ varie moins que celle de l'algorithme PCM. L'approche PCQ fournit des résultats de classification plus stables et cohérents que les méthodes PCM et IND.

5.4 Sélection du nombre de classes par la recherche de pics de densité

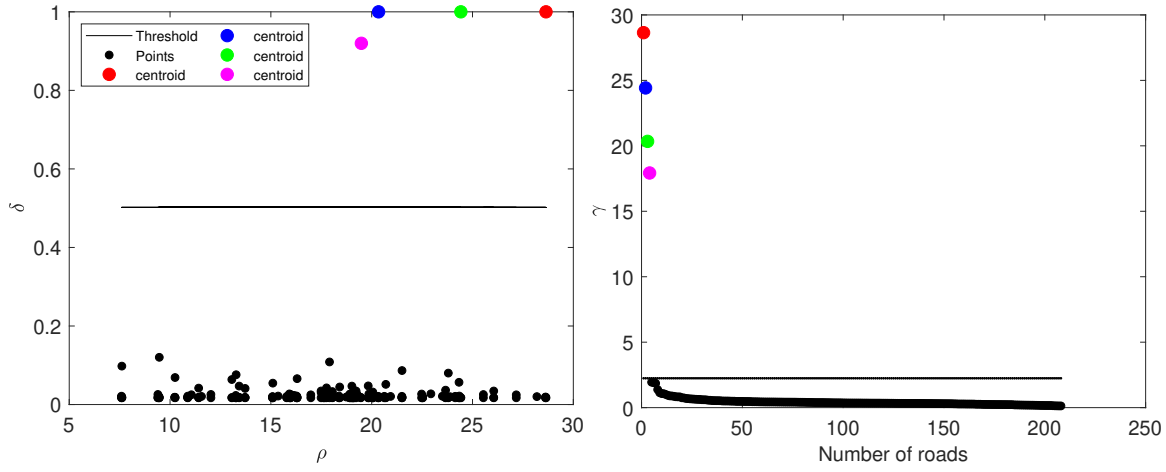
L'algorithme de classification par recherche des pics de densité (DPC) repose sur l'hypothèse que les centres de classes sont dans un voisinage de faible densité locale et qu'ils sont à une distance relativement importante de tout point de densité locale plus élevée. Selon ces hypothèses, deux quantités seront calculées pour chaque snake T_i : la densité locale ρ_i et la distance δ_i. La densité locale ρ_i, mesure le nombre des snakes qui sont similaires à un snake T_i dans un rayon d_c. La valeur de ρ_i est définie en utilisant le noyau gaussien selon l'équation 2.2. La distance δ_i calcul la distance minimale entre un snake T_i et un autre snake ayant une densité plus élevée et qui est la plus proche à T_i. Dans le but de conserver la structure du réseau routier et le voisinage entre les liens de celui-ci, la matrice des distances $\mathcal{D}$ est définie, telle que, $\mathcal{D}_{ij} = d(T_i, T_j)$ qui est la distance entre les couples de snakes T_i et T_j (Ning et al., 2007b) :

$$d(T_i, T_j) = 1 - e^{\left(\frac{-1}{\eta w_{ij}}\right)}, \tag{5.15}$$

où η est une variable qui contrôle l'effet marginal de la dispersion entre les snakes. Dans notre expérience, nous considérons $\eta = 3$ dans le but d'obtenir des snakes qui sont bien distribués dans l'espace. Par la suite δ_i est calculé suivant l'équation 2.3. Cependant, les performances de l'algorithme DPC dépendent fortement du rayon d_c. Pour éviter cette limitation, la valeur de d_c est automatiquement extraite de l'ensemble des snakes en minimisant la fonction d'entropie potentielle définie par l'équation 2.6 (Wang et al., 2016).

Après avoir calculé ces mesures, le nombre de classes K est recherché en identifiant les centres de classe parmi tous les points représentant les snakes. Dans ce contexte, les centres de classe sont supposés avoir des valeurs élevées à la fois pour ρ_i et δ_i. De ce fait, deux méthodes peuvent être utilisées pour détecter les centres. La première consiste à tracer le

graphe de décision qui est la représentation de δ_i en fonction de ρ_i pour tous les points comme illustré dans la figure 2.3. Les centres de classes sont identifiés par les points ayant des valeurs élevées à la fois pour δ_i et ρ_i. La deuxième méthode consiste à tracer pour toutes les snakes la valeur de γ_i (équation 2.5) triée par ordre décroissant, puis à sélectionner les centres quand cette quantité augmente considérablement. Au final, la sélection des centres de classes sera effectuée selon l'équation 2.5 en définissant un seuil γ_{seuil}. Le seuil γ_{seuil} est déterminé automatiquement en se basant sur le calcul d'une fonction de densité de probabilité (Yan et al., 2018). Cette méthode est détaillée dans la section 2.7 du chapitre 2. Les centres de classes sont alors identifiés en calculant pour chaque snake un seuil dynamique γ_{seuil} définit selon l'équation 2.11.
La figure 5.23a montre le graphe de décision pour la période $t = 5$. Les valeurs de seuils divisent le plan (ρ, δ) en deux zones de décision et identifient quatre centres de classes qui sont situés au-dessus de la valeur de seuil. Ensuite, dans la figure 5.23b, nous traçons les valeurs de γ_i (équation 2.5) triées par ordre décroissant pour la même période $t = 5$. Nous remarquons que les valeurs de γ_i sont très élevées pour ces mêmes centres de classes.



(a) (b)

FIGURE 5.23 – (a) Graphe de décision pour la période $t = 5$, où chaque centre est coloré par la couleur de sa classe. (b) Valeurs de γ triées par ordre décroissant pour toutes les snakes.

Dans ce contexte, l'algorithme des pics de densité est utilisé pour détecter automatiquement le nombre de classes nécessaires pour la classification des différentes liens du réseau de transport en utilisant les algorithmes de classification spectrale statique et évolutive.

5.4.1 Études comparative entre les méthodes de sélection de nombre de partitions

Dans la section 5.3.8, nous avons présenté une étude sur la sélection automatique du nombre de classes en utilisant la modularité et Eigengap. Dans cette expérience, nous reprenons les résultats des deux méthodes et nous les comparons avec ceux obtenus par l'algorithme de recherche par pics de densité.

Dans le but de comparer les performances des trois méthodes, nous traçons les mêmes schémas conçues dans la section 5.3.8. La figure 5.24 montre les variations de nombre de classes pour chacune des méthodes. La méthode par pics de densité (DPC) présente un nombre de classe plus petit et plus stable entre les périodes.

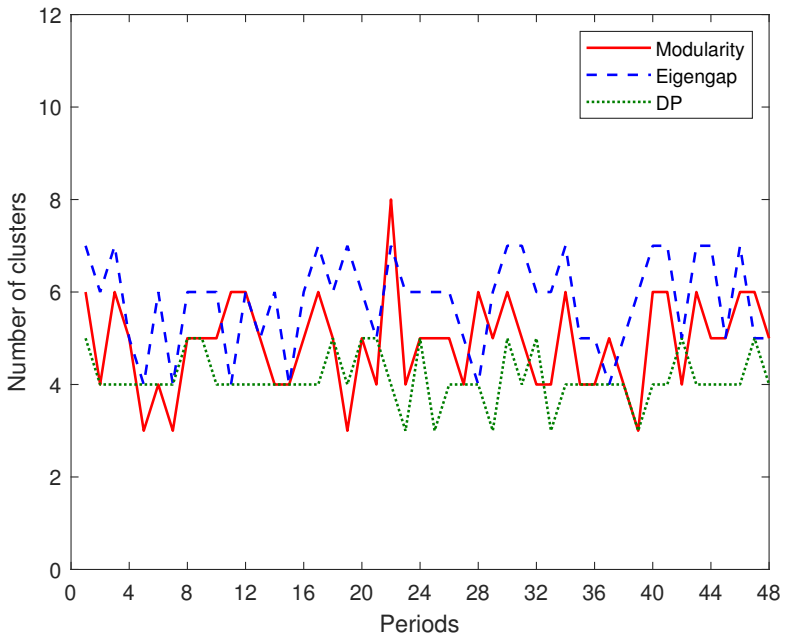


FIGURE 5.24 – Nombre de classes obtenues en appliquant les méthodes de la modularité, eigengap et pics de densité pour chaque période.

Par la suite, nous comparons les performances du partitionnement statique et évolutif pour les trois méthodes. De ce fait, nous appliquons les approches IND, PCQ et PCM en utilisant un nombre de classes défini par la méthode de pics de densité. De plus, nous reprenons les résultats obtenus par celles de la modularité et de la méthode d'Eigengap. Le tableau 5.5 montre les résultats obtenus en calculant les coûts instantanés, temporels et totaux. Les meilleurs résultats sont représentés en gras. Nous remarquons que la méthode des

pics de densité assure les meilleurs résultats en minimisant tous les coûts.

Dans le but de valider les performances du DPC sur toutes les périodes, nous traçons les

TABLEAU 5.5 – Performance du partitionnement des méthodes de la modularité, Eigengap et pics de densité.

Method	Coût	IND	PCQ	PCM
Modularité	SC	0.16	0.20	0.21
	TC	0.45	0.27	0.30
	Total Cost	0.28	0.23	0.25
Eigengap	SC	0.28	0.34	0.37
	TC	0.68	0.46	0.46
	Total Cost	0.44	0.39	0.40
Density Peaks	SC	**0.06**	0.09	0.10
	TC	0.26	**0.13**	0.15
	Total Cost	0.14	**0.10**	0.12

résultats du coût total de l'approche PCQ pour les 3 méthodes sur les 48 périodes. La figure 5.25 montre les résultats obtenus où la méthode de recherche par pics de densités apporte les meilleures résultats en minimisant le coût total.

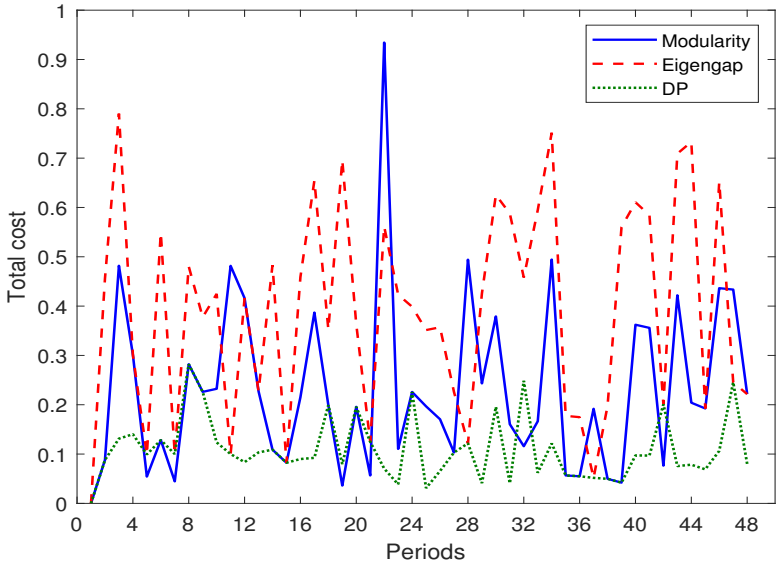


FIGURE 5.25 – Variation du coût total pour l'approche PCQ en utilisant les trois méthodes : modularité, eigengap et pics de densité.

Par la suite, nous présentons les résultats de partitionnement spectral statique et évolutif

sur le réseau d'Amsterdam. Le nombre de classes est déterminé à partir de la méthode de recherche des pics de densités. Dans un premier temps, nous présentons dans la figure 5.26 les résultats obtenus pour les périodes $t = 5$ à $t = 7$ avec les méthodes IND, PCM et PCQ. Nous remarquons que les approches évolutives (PCQ et PCM) permettent de maintenir la régularité du partitionnement en tenant compte des changements des vitesses des liens au temps t, sans trop s'écarter des évolutions historiques des partitions. Dans un deuxième temps, nous comparons les moyennes des vitesses des liens dans les classes et leurs écarts-types pour les trois méthodes. Les résultats de ces mesures sont présentés en détail pour la période t = 6 dans le tableau 5.6. Nous pouvons constater que les vitesses moyennes entre les partitions sont dissimilaires pour les trois méthodes avec une amélioration pour les approches PCQ et PCM par rapport à la méthode IND dans le cas des classes rouge et bleu. De plus, les classes obtenues ont des valeurs des écarts-types plus petites dans le cas du PCM et du PCQ, ce qui démontre une meilleure performance dans la séparation des classes homogènes connectées.

TABLEAU 5.6 – Valeur de la vitesse moyenne (m/s) et écarts-types pour la période $t = 6$ pour les 3 approches : IND, PCM et PCQ.

(μ/σ)	Rouge	Vert	Bleu	Cyan
IND	7.74/2.9	9.99/2.3	10.48/4.1	15.15/2.12
PCM	5.71/1.1	9.89/2.3	11.81/3.8	15.15/2.12
PCQ	5.71/1.1	9.89/2.3	11.81/3.8	15.15/2.12

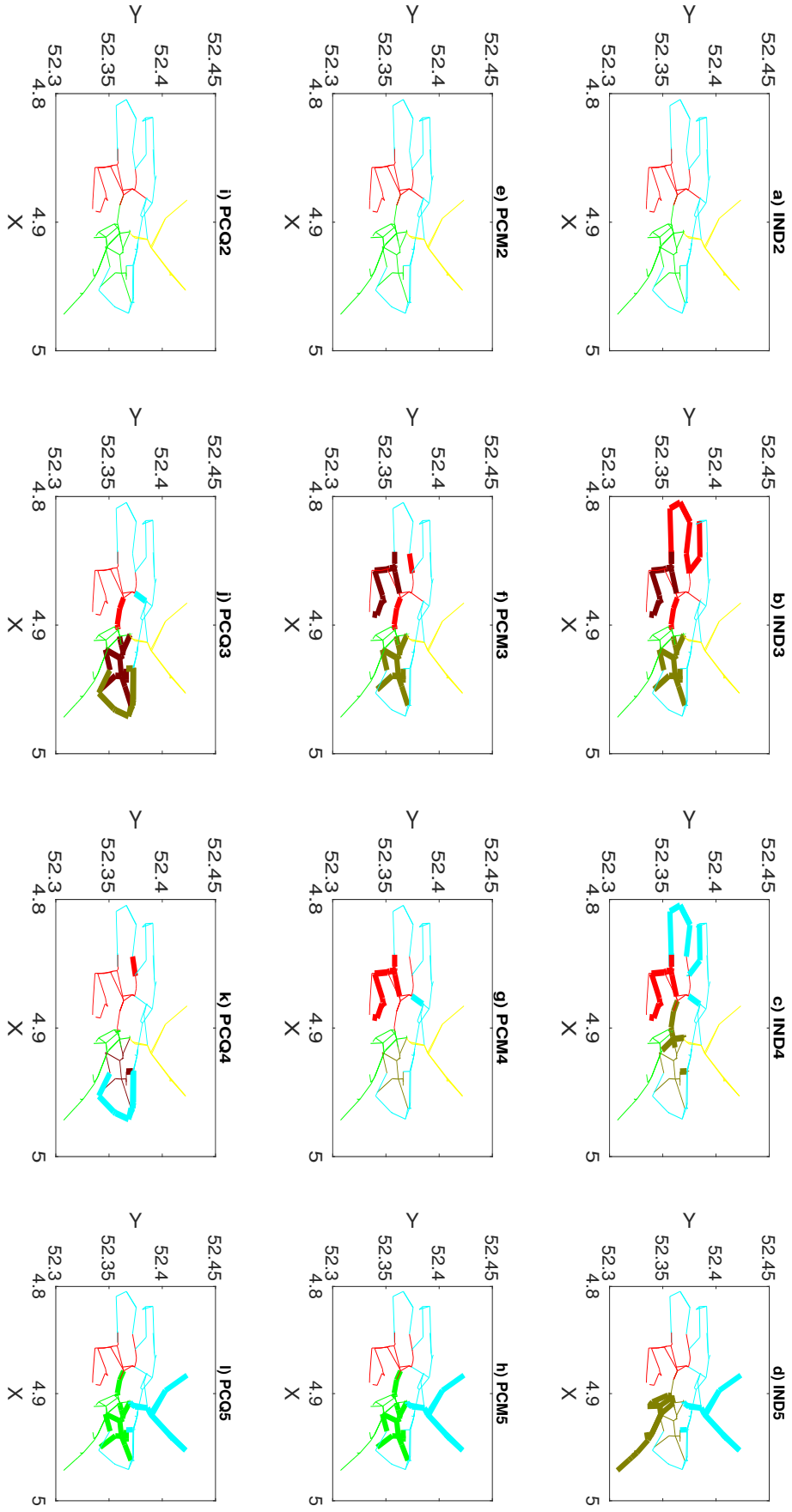


FIGURE 5.15 – Résultats du partitionnement pour des périodes $t = 2$ à $t = 5$ pour les approches IND, PCM et PCQ.

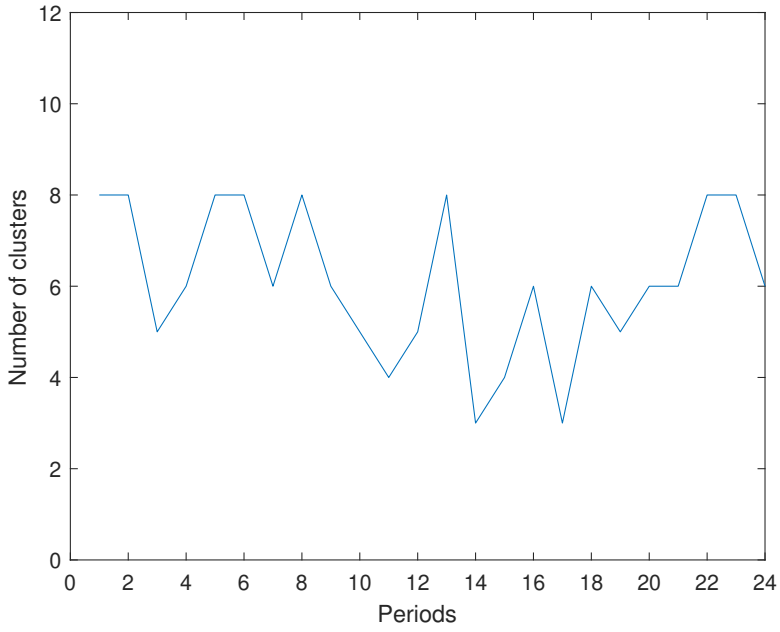


FIGURE 5.18 – Nombre de classes obtenues en appliquant la méthode de modularité pour toutes les périodes en cas de période de temps = 20 minutes.

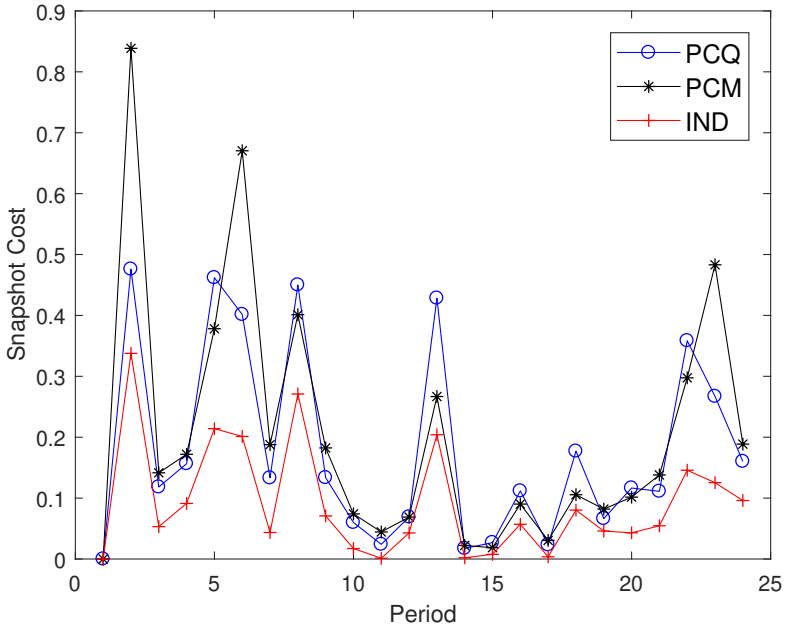


FIGURE 5.19 – Variation du coût instantané pour chaque période en cas de période de temps = 20 minutes.

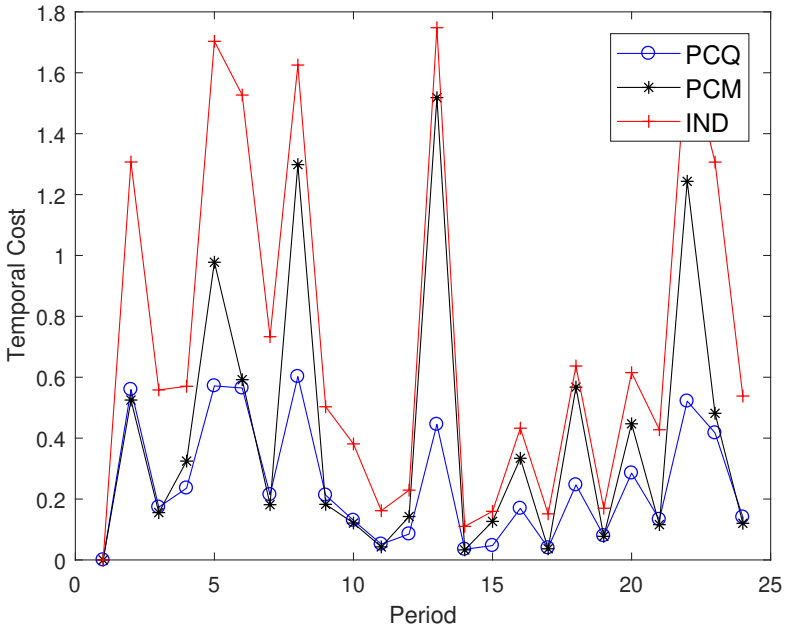


FIGURE 5.20 – Variation du Coût temporel pour chaque période dans le cas d'une période de temps = 20 minutes.

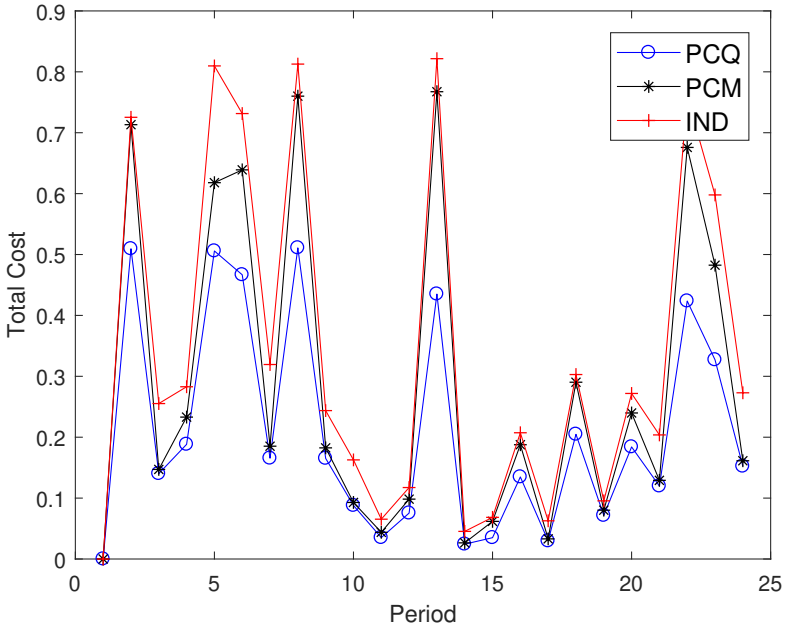


FIGURE 5.21 – Variation du coût total pour chaque période en cas de période de temps = 20 minutes.

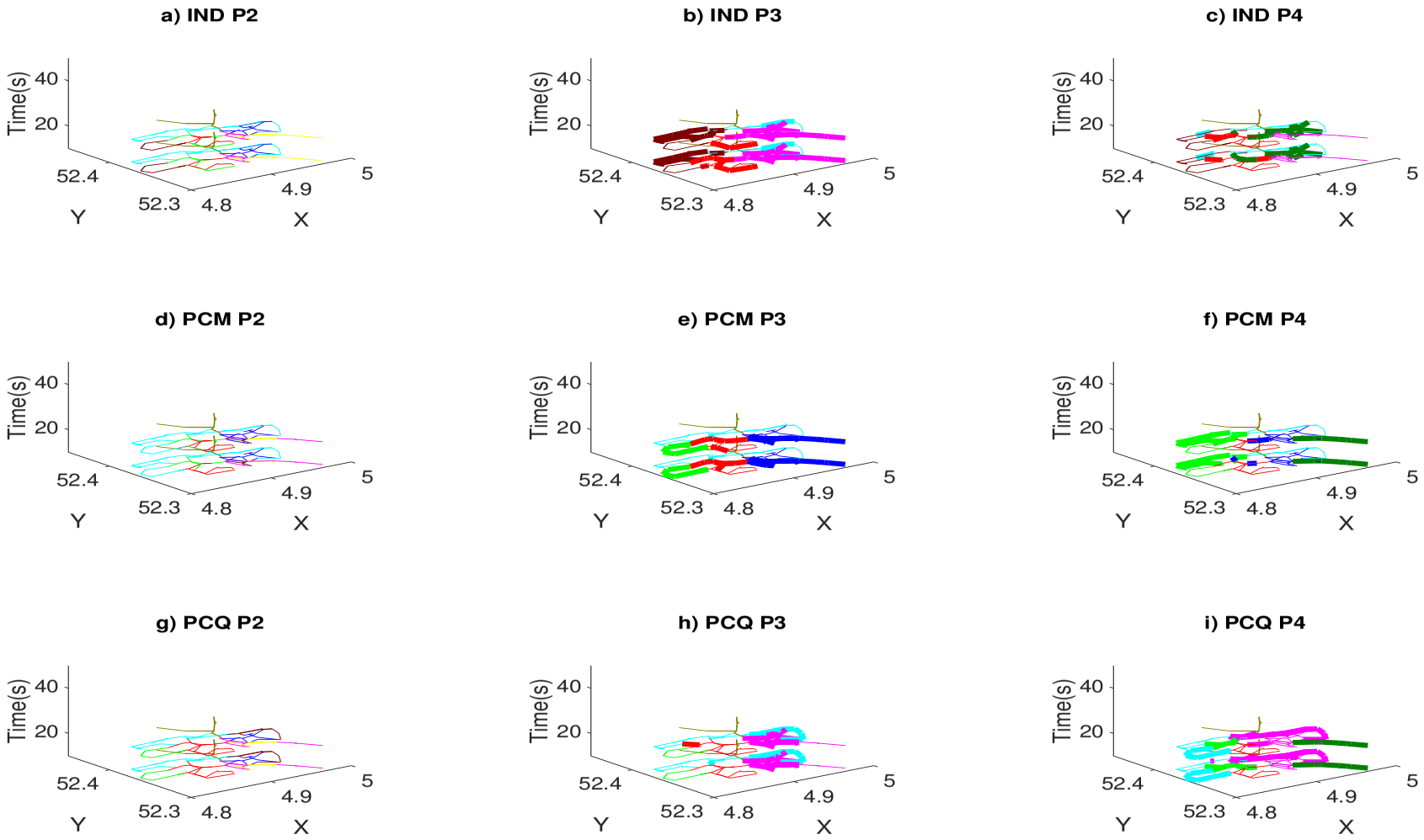
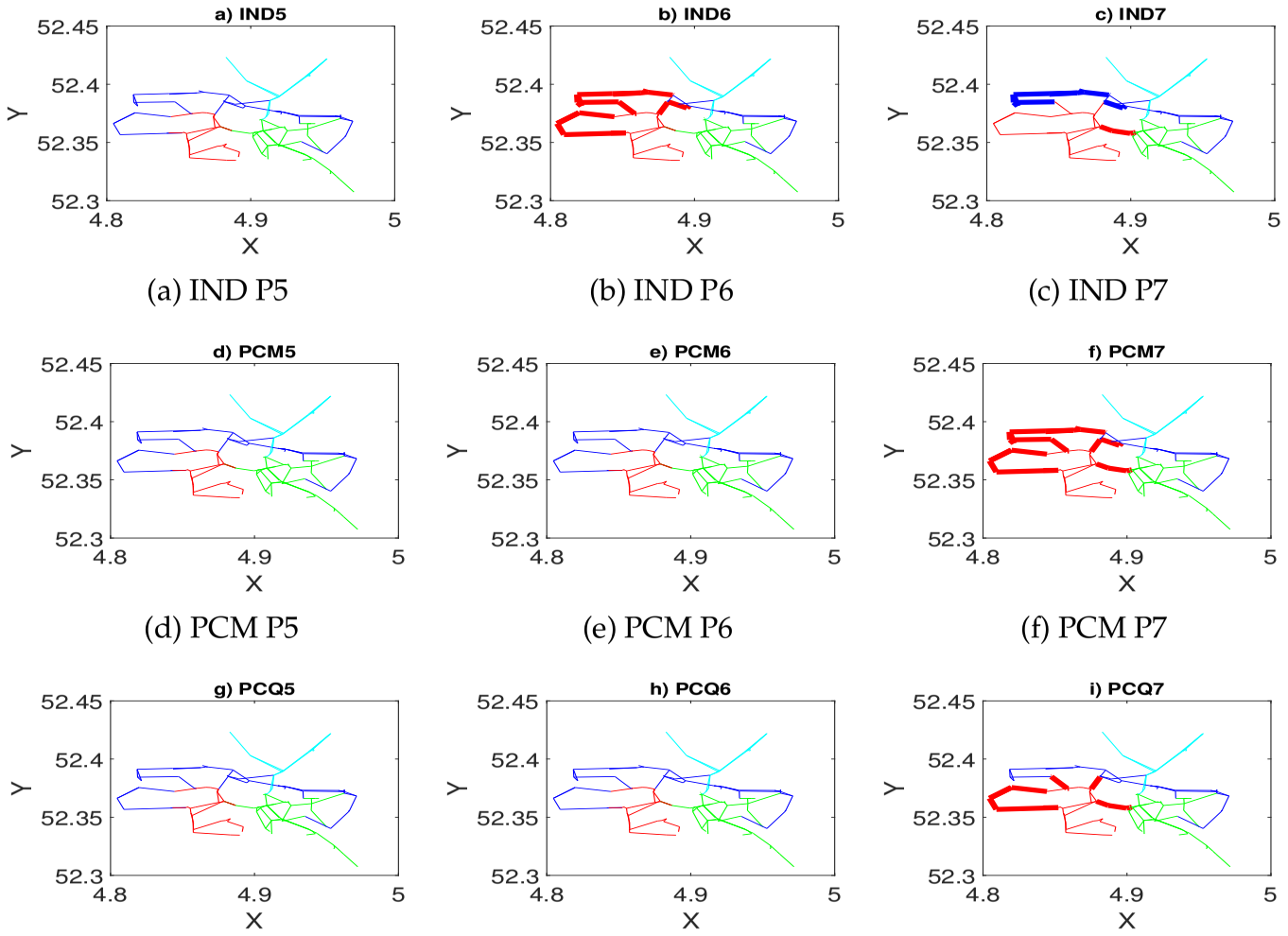


FIGURE 5.22 – Résultats du partitionnement pour les périodes $t = 2$ à $t = 4$ pour les méthodes IND, PCM et PCQ dans le cas d'une période de temps = 20 minutes.



(a) IND P5 (b) IND P6 (c) IND P7

(d) PCM P5 (e) PCM P6 (f) PCM P7

(g) PCQ P5 (h) PCQ P6 (i) PCQ P7

FIGURE 5.26 – Résultats des classifications pour les périodes $t = 5$ à $t = 7$ pour les trois méthodes : IND, PCM et PCQ.

5.5 Conclusion

Dans ce chapitre, notre objectif est de partitionner un réseau de transport en incorporant le lissage temporel dans la classification spectrale évolutive. De ce fait, nous avons présenté deux approches pour partitionner le réseau urbain. Dans la première approche, le coût temporel est exprimé en considérant la partition actuelle qui regroupe la variation historique des vitesses routières. La deuxième approche diffère de la première par la façon dont la régularité temporelle est mesurée. Dans cette approche, le coût temporel est exprimé comme la différence entre la partition actuelle et la partition historique. Pour les deux approches, une fonction de coût est définie pour assurer la régularité temporelle.

Dans le but de déterminer le nombre de classes optimal, nous avons utilisé la méthode de la modularité qui s'est avérée efficace par rapport à l'heuristique d'Eigengap. La méthodologie a été appliquée sur le réseau réel de la ville d'Amsterdam. Les résultats expérimentaux montrent que les deux approches de classification spectrale évolutive proposée fournissent des résultats stables et cohérents dans un environnement où le trafic routier est dynamique au cours du temps.

De plus, nous avons évalué les performances de l'algorithme de recherche par pics de densité pour la détection du nombre de classes que prennent les méthodes de classification spectrale en entrée. Les résultats obtenus ont démontré l'efficacité de cette méthode pour la détection du nombre de classes pour différentes formes de données. Néanmoins, l'algorithme DPC est un algorithme de classification non supervisé qui peut être utilisé indépendamment pour la détection des zones de congestion sur un réseau urbain. De ce fait, développer une méthode évolutive adaptée aux données dynamiques peut être intéressante pour améliorer la connexité et l'homogénéité des classes.

Conclusion générale

La représentation par graphes offre un outil riche et efficace pour l'analyse du trafic urbain. Dans ce sens, la carte routière est transformée en graphe dont les poids des connexions entre paires de nœuds représentent les similarités exprimées en relation avec la vitesse voire la densité du trafic. Deux types de graphes peuvent être déduits de cette carte routière : le graphe primal et le graphe dual. Le premier est simple et intuitif, consiste à représenter les croisements comme nœuds de graphe et les routes les reliant comme arêtes. Le graphe dual, à l'inverse, consiste à associer les nœuds aux routes et les arêtes aux croisements de routes. Si le graphe primal correspond naturellement à la carte routière, le graphe dual est mieux adapté à l'étude de l'état du trafic urbain. En effet, l'objectif est de trouver des zones de routes de vitesses ou densités homogènes. Dans la littérature, des travaux ont étudié le partitionnement spatial de réseau, basé sur une hypothèse statique. Partitionner un réseau en considérant des algorithmes évolutifs et incrémentaux est à notre connaissance une approche nouvelle.

Deux types de méthodes de partitionnement sont comparés : les méthodes de clustering caractérisées par leurs similarités (classification spectrale) et les méthodes de clustering caractérisées par leurs densités (pics de densités).

L'algorithme de classification automatique par recherche de pics de densités consiste à déterminer le graphe ou plan de décision. Le plan a pour abscisses les densités locales des points rangées par ordre croissant et pour ordonnées les distances des points à leurs plus proches centres denses. La densité dépend du paramètre rayon de l'hypersphère centré sur chaque point. Une forme soft de l'algorithme considère une fonction gaussienne centrée sur chaque point, la dispersion de la gaussienne constitue le paramètre à optimiser. Cet algorithme a été appliqué dans nos simulations en utilisant le réseau urbain dynamique (SUMO) et aussi dans la partie expérimentale en considérant le réseau de transport d'Amsterdam.

L'analyse des graphes s'appuie sur les propriétés algébriques des matrices de similarités associées, notamment les matrices Laplaciennes. La classification spectrale permet de découvrir les propriétés et la structure d'un graphe à partir de l'étude du spectre de la matrice Laplacienne. Il existe divers algorithmes de classification spectrale dont l'objet est de partitionner un graphe en groupes homogènes ou classes. L'idée de base consiste à considérer l'espace de projection engendré par les vecteurs propres de la matrice Laplacienne. Dans cet espace, l'analyse des points représentant la structure du graphe est réalisée par de simples algorithmes de classification non supervisée tels que l'algorithme de K-means ou de mélange de densités. La plupart des algorithmes de classification spectrales supposent connu le nombre des classes. Il est possible de le rechercher à partir de l'indice de la distance maximum (eigengap) entre les valeurs propres ordonnées de la matrice Laplacienne et/ou par le calcul du maximum de la modularité du graphe.

Les réseaux de trafic urbain et par suite les graphes qui lui sont associés évoluent au cours du temps. Selon l'état au cours du temps des graphes, la classification spectrale peut être statique, incrémentale ou dynamique. La classification incrémentale de graphe s'applique sur les graphes ayant leurs poids de similarités évoluant au cours du temps. Elle consiste en l'étude de la sensibilité des valeurs et vecteurs propres de la matrice Laplacienne aux changements des poids des arêtes du graphe. Cette sensibilité est étudiée pour aboutir à des relations incrémentales des valeurs et des vecteurs propres. La classification incrémentale a été appliquée sur le réseau simulé urbain dynamique (SUMO).

L'analyse de l'évolution du trafic au cours du temps conduit à partitionner le graphe dynamique associé. Les classes obtenues ont leur nombre et leurs formes évoluant au cours du temps. Il est préférable que l'évolution des classes soit lisse. Deux critères de qualités sont utilisés pour évaluer la qualité de la classification : préservation de la qualité du clustering (PCQ) et préservation de la qualité de l'appartenance aux classes (PCM). Les deux critères ont en commun la fonction coupe normalisée à l'instant t. Le critère PCQ combine la coupe normalisée à l'instant t avec la coupe normalisée à l'instant t-1 en tenant compte la partition à l'instant t. Le critère PCM combine la coupe normalisée à l'instant t avec un terme exprimant un coût temporel qui mesure la "distance" entre la partition à l'instant t et celle à l'instant t-1. Les deux critères sont appliqués sur un réseau urbain réel évoluant sur une période déterminée d'une journée. Nous avons montré que la classification évolutive apporte plus de robustesse et d'efficacité dans un environnement dynamique.

Perspectives

Le travail de thèse ouvre différentes perspectives. On peut en citer quelques-unes.

— L'algorithme de pics de densités est un algorithme de classification non supervisé qui peut être utilisé indépendamment pour la détection des zones de congestion sur un réseau urbain. Il serait intéressant de rendre l'algorithme des pics de densités incrémental voire dynamique.

— Les résultats obtenus par les algorithmes de classification spectrale et de pics de densités ne sont pas toujours en accord. Différents critères de comparaison d'algorithmes extraient de la matrice de contingence peuvent être appliqués. De plus, il serait aussi intéressant d'analyser les nœuds instables pouvant appartenir à plus d'une classe.

— Les cartes routières contiennent des routes à sens uniques et des routes à doubles sens. Dans ce cas, il est plus naturel d'employer des graphes dirigés et par suite des approches de classification spectrale adaptées aux graphes dirigés.

— La classification évolutive a été traitée en considérant uniquement le changement de poids des arêtes. Elle peut être traitée d'une manière plus approfondie en considérant, en plus, la création et la disparition de nœuds du graphe.

— Enfin, l'étude du trafic a été effectuée sur une journée, il serait intéressant d'étudier le trafic sur une plus longue durée et à différentes échelles temporelles afin de découvrir des formes de régularités dans les motifs de congestion voire estimer le temps de parcours.

Bibliographie

Aggarwal, C. C., Philip, S. Y., Han, J., and Wang, J. (2003). A framework for clustering evolving data streams. In *Proceedings 2003 VLDB conference*, pages 81–92. Elsevier.

Ahmadzai, F., Rao, K. L., and Ulfat, S. (2019). Assessment and modelling of urban road networks using integrated graph of natural road network (a gis-based approach). *Journal of Urban Management*, 8(1) :109–125.

Al Alam, P., Hamad, D., Constantin, J., Constantin, I., and Zaatar, Y. ((2020)). Dynamic partitioning of transportation network using evolutionary spectral clustering. In *Smart Applications and Data Analysis,Third International Conference, SADASC 2020, Marrakesh, Morocco, June 25–26*, pages 178–186. Springer Link.

Anwar, T., Liu, C., Vu, H. L., and Leckie, C. (2017). Partitioning road networks using density peak graphs : Efficiency vs. accuracy. *Information Systems*, 64 :22–40.

Bach, F. R. and Jordan, M. I. (2006). Learning spectral clustering, with application to speech separation. *The Journal of Machine Learning Research*, 7 :1963–2001.

Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). Sumo–simulation of urban mobility : an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind.

Chakrabarti, D., Kumar, R., and Tomkins, A. (2006). Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 554–560.

Chen, Y., Zhang, Y., and Ji, X. (2005). Size regularized cut for data clustering. *Advances in Neural Information Processing Systems*, 18 :211–218.

Chi, Y., Song, X., Zhou, D., Hino, K., and Tseng, B. L. ((2007)). Evolutionary Spectral Clustering by Incorporating Temporal Smoothness. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 153–162.

Chi, Y., Song, X., Zhou, D., Hino, K., and Tseng, B. L. (2009). On Evolutionary Spectral Clustering. *ACM Transaction on Knowledge Discovery from Data, Article 17,*, 3(4).

Chikhi, N. F. (2010). *Calcul de centralité et identification de structures de communautés dans les graphes de documents*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier.

Codeca, L., Frank, R., and Engel, T. (2015). Luxembourg sumo traffic (lust) scenario : 24 hours of mobility for vehicular networking research. In *2015 IEEE Vehicular Networking Conference (VNC)*, pages 1–8. IEEE.

Çolak, S., Lima, A., and González, M. C. (2016). Understanding congested travel in urban areas. *Nature communications*, 7(1) :1–8.

Crucitti, P., Latora, V., and Porta, S. (2006). Centrality in networks of urban streets. *Chaos : an interdisciplinary journal of nonlinear science*, 16(1) :015113.

El Mahrsi, M. K. and Rossi, F. (2012). Graph-based approaches to clustering network-constrained trajectory data. *CoRR*, abs/1210.0762.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Farhi, N., Phu, C. N. V., Amir, M., Haj-Salem, H., and Lebacque, J.-P. (2015). A semi-decentralized control strategy for urban traffic. *Transportation Research Procedia*, 10 :41 – 50.

Flamenbaum, G., Thiery, J., and Benzecri, J. (1979). Agrégation en boules de rayon fixe et centres optimisés. *Cahiers de l'analyse des données*, 4(3) :357–364.

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social networks*, 1(3) :215–239.

Gundaliya, P., Mathew, T. V., and Dhingra, S. L. (2008). Heterogeneous traffic flow modelling for an arterial using grid based approach. *Journal of Advanced Transportation*, 42(4) :467–491.

Hong, C., Jingjing, Z., Chunfeng, C., and Qinyu, C. (2016). Solving large-scale assignment problems by Kuhn-Munkres algorithm. In *2nd International Conference on Advances in Mechanical Engineering and Industrial Informatics*, pages 822–827.

Ji, Y. and Geroliminis, N. (2012a). On the spatial partitioning of urban transportation networks. *Transportation Research Part B, Elsevier*, 46(10) :1639–1656.

Ji, Y. and Geroliminis, N. (2012b). On the spatial partitioning of urban transportation networks. *Transportation Research Part B : Methodological*, 46(10) :1639–1656.

Koschützki, D., Lehmann, K. A., Peeters, L., Richter, S., Tenfelde-Podehl, D., and Zlotowski, O. (2005). Centrality indices. In *Network analysis*, pages 16–61. Springer.

Krajzewicz, D. (2010). Traffic simulation with sumo–simulation of urban mobility. In *Fundamentals of traffic simulation*, pages 269–293. Springer.

Krajzewicz, D., Erdmann, J., Behrisch, M., and Bieker, L. (2012). Recent development and applications of sumo-simulation of urban mobility. *International Journal On Advances in Systems and Measurements*, 5(3&4).

Krajzewicz, D., Hertkorn, G., Rössel, C., and Wagner, P. (2002). Sumo (simulation of urban mobility)-an open-source traffic simulation. In *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, pages 183–187.

Kunegis, J., Lommatzsch, A., and Bauckhage, C. (2008). Alternative similarity functions for graph kernels. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE.

Li, Z. and Tang, Y. (2018). Comparative density peaks algorithm. *Expert Systems with applications*, 95 :236 – 247.

Lin, J.-L. (2019). Accelerating density peak clustering algorithm. *Symmetry*, 11(7) :859.

Liu, P., Liu, Y., Hou, X., Li, Q., and Zhu, Z. (2015). A text clustering algorithm based on find of density peaks. In *2015 7th International Conference on Information Technology in Medicine and Education (ITME)*, pages 348–352. IEEE.

Lopez, C. (2017). *Modélisation dynamique du trafic et transport de marchandises en ville : vers une approche combinée*. PhD thesis, Université de Lyon.

Lopez, C., Krishnakumari, P., Leclercq, L., Chiabaut, N., and Lint, H. V. (2017). Spatio-temporal partitioning of transportation network using travel time data. *Transportation Research Record, Journal of the trasportation research*, 2623(1) :98–107.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and WieBner, E. (2018). Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582. IEEE.

Losee, R. M. (2012). *Text retrieval and filtering : analytic models of performance*, volume 3. Springer Science & Business Media.

Ma, Y.-Y., Chiu, Y.-C., and Yang, X.-G. (2009). Urban traffic signal control network automatic partitioning using laplacian eigenvectors. In *2009 12th International IEEE Conference on Intelligent Transportation Systems*, pages 1–5. IEEE.

Maiorov, E., Ludan, I., Motta, J., and Saprykin, O. (2019). Developing a microscopic city model in sumo simulation system. In *Journal of Physics : Conference Series*, volume 1368, page 042081. IOP Publishing.

Meilă, M. (2007). Comparing clusterings-an information based distance. *Journal of multivariate analysis*, 98(5) :873–895.

Meila, M. and Shi, J. (2001). Learning segmentation by random walks. In *Advances in neural information processing systems*, pages 873–879.

Moody, J. and Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural computation*, 1(2) :281–294.

Newman, M. E. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical review E*, 69(2) :026113.

Ng, A. Y., Jordan, M. I., and Weiss, Y. ((2002)). On spectral clustering : Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

Ning, H., Xu, W., Chi, Y., Gong, Y., and Huang, T. (2007a). Incremental spectral clustering with application to monitoring of evolving blog communities. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 261–272. SIAM.

Ning, H., Xu, W., Chi, Y., Gong, Y., and Huang, T. (2007b). Incremental spectral clustering with application to monitoring of evolving blog communities. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 261–272.

Ning, H., Xu, W., Chi, Y., Gong, Y., and Huang, T. S. (2010). Incremental spectral clustering by efficiently updating the eigen-system. *Pattern Recognition*, 43(1) :113–127.

Ognjenović, S., Zafirovski, Z., and Vatin, N. (2015). Planning of the traffic system in urban environments. *Procedia Engineering*, 117 :574–579.

Opsahl, T., Agneessens, F., and Skvoretz, J. (2010). Node centrality in weighted networks : Generalizing degree and shortest paths. *Social networks*, 32(3) :245–251.

Pascale, A., Mavroeidis, D., and Lam, H. T. (2015). Spatiotemporal clustering of urban networks : Real case scenario in london. *Transportation Research Record*, 2491(1) :81–89.

Pojani, D. and Stead, D. (2015). Sustainable Urban Transport in the Developing World : Beyond Megacities. *Sustainability*, 7(6) :7784–7805.

Rodriguez, A. and Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, 344(6191) :1492–1496.

Saeedmanesh, M. and Geroliminis, N. (2016). Clustering of heterogeneous networks with directional flows based on "snake" similarities. *Transportation Research Part B : Methodological*, 91 :250–269.

Saeedmanesh, M. and Geroliminis, N. (2017). Dynamic clustering and propagation of congestion in heterogeneously congested urban traffic networks. *Transportation Research Part B, Elsevier*, 105 :193–211.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5) :513–523.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8) :888–905.

Shi, Y., Chen, Z., Qi, Z., Meng, F., and Cui, L. (2017). A novel clustering-based image segmentation via density peaks algorithm with mid-level feature. *Neural Computing and Applications*, 28(1) :29–39.

Shortreed, S. and Meila, M. (2005). Unsupervised spectral learning. *Proceedings of the Twenty-First Conference Annual on Uncertainty in Artificial Intelligence*, pages 534–541.

Valgren, C., Duckett, T., and Lilienthal, A. (2007). Incremental spectral clustering and its application to topological mapping. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4283–4288. IEEE.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4) :395–416.

Wang, S., Wang, D., Li, C., Li, Y., and Ding, G. (2016). Clustering by fast search and find of density peaks with data field. *Chinese Journal of Electronics*, 25(3) :397–402.

Wasserman, S., Faust, K., et al. (1994). Social network analysis : Methods and applications.

White, S. and Smyth, P. (2005). A spectral clustering approach to finding communities in graphs. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 274–285. SIAM.

Xu, K., Kliger, M., and Hero, A. (2015). Adaptive evolutionary clustering. *Data Mining and Knowledge Discovery*, 28(2) :304–336.

Xu, Q., Desjardins, M., and Wagstaff, K. (2005). Constrained spectral clustering under a local proximity structure assumption. In *In FLAIRS Conference*. Citeseer.

Xu, X., Yuruk, N., Feng, Z., and Schweiger, T. A. (2007). Scan : a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833.

Yan, H., Lu, Y., and Ma, H. (2018). Density-based Clustering using Automatic Density Peak Detection. In *International Conference on Pattern Recognition Applications and Methods*, pages 95–102.

Yan, H., Wang, L., and Lu, Y. (2019). Identifying cluster centroids from decision graph automatically using a statistical outlier detection method. *Neurocomputing*, 329 :348–358.

Yu, L. and Ding, C. ((2010)). Network community discovery : solving modularity clustering via normalized cut. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pages 34–36.

Zaki, M. J., Meira Jr, W., and Meira, W. (2014). *Data mining and analysis : fundamental concepts and algorithms*. Cambridge University Press.

Zelnik-Manor, L. and Perona, P. (2005). Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608, Vancouver, British Columbia, Canada.

Zhang, B., Xing, K., Cheng, X., Huang, L., and Bie, R. (2012). Traffic clustering and online traffic prediction in vehicle networks : A social influence perspective. In *2012 Proceedings IEEE Infocom*, pages 495–503. IEEE.

Zhao, Y., Yuan, Y., Nie, F., and Wang, Q. (2018). Spectral clustering based on iterative optimization for large-scale and high-dimensional data. *Neurocomputing*, 318 :227–235.

Zhou, Z., Lin, S., and Xi, Y. (2012). A dynamic network partition method for heterogenous urban traffic networks. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 820–825. IEEE.

Publications

Au cours de cette thèse de doctorat, deux articles ont été présentés et publiés dans des conférences internationales. En plus, un article a été soumis à une revue internationale.

Conférences internationales :

1. Al Alam, Pamela, Denis Hamad, Joseph Constantin, Ibtissam Constantin, and Youssef Zaatar. "Dynamic Partitioning of Transportation Network Using Evolutionary Spectral Clustering." In International Conference on Smart Applications and Data Analysis, pp. 178-186. Springer, Cham, 2020.
2. P. Al Alam, D. Hamad, J. Constantin, I. Constantin, and Y. Zaatar "Evolutionary clustering for dynamic partitioning of transportation network", Proc. SPIE 11400, Pattern Recognition and Tracking XXXI, 1140005 (22 April 2020)

Article soumis à une revue internationale :

1. Al Alam Pamela, Lopez Clélia, Constantin Ibtissam, and Constantin Joseph "Evolutionary Spectral Clustering for Graph Models with Application to Transportation Network" in Transportmetrica B : Transport Dynamics. 30 pages. Soumis le 17 mai 2021.

Liste des tableaux

Liste des figures