École polytechnique
Promotion X2013
BRICHLER Nicolas

RAPPORT DE STAGE DE RECHERCHE

Représentation de la voirie d'un territoire par un modèle de Manhattan

.....

NON CONFIDENTIEL

Département de Mathématiques Appliquées
MAP 593 : Automatique et Recherche Opérationnelle
Enseignants référents : Stéphane Gaubert & Vincent Bansaye
Maître de stage : Elie Cali
Dates du stage : 14 mars 2016 - 26 août 2016
Orange Gardens
44 Avenue de la République,
92320 Châtillon
France

Déclaration d'intégrité relative au plagiat

Je soussigné Brichler Nicolas certifie sur l'honneur :
1. Que les résultats décrits dans ce rapport sont l'aboutissement de mon travail.
2. Que je suis l'auteur de ce rapport.
3. Que je n'ai pas utilisé des sources ou résultats tiers sans clairement les citer et les référencer selon les règles bibliographiques préconisées.

Je déclare que ce travail ne peut être suspecté de plagiat.

Date ..... Signature

Sommaire

1 Abstract

The widespread development of high-speed internet in all of France (by 2022) and in most of the major cities around the world motivates Orange to seek new ways of geometrically modelling street networks in an effort to better assess the fiber optic wiring and maintenance costs. When a city has been centrally planned, it is filled mostly with a series of parallel and perpendicular streets, forming rectangular city blocks, very much like the city grid of Manhattan in New York. Therefore the aim of this internship is to study "Manhattan" street networks through the lens of stochastic geometry in order to evaluate the probability distribution of the distance between network nodes and eligibility for potential consumers.

Résumé

La généralisation de l'internet très haut débit partout en France (d'ici 2022), et même dans la plupart des grandes villes du monde entier, amène Orange à proposer des modèles géométriques de représentation de la voirie des grandes villes dans le but d'estimer les coûts nécessaires au câblage en fibre optique et à la maintenance du réseau. Il apparaît notamment que plus une ville est récente et de conception centralisée, plus ses quartiers seront constitués de pâtés de maison de forme rectangulaire, comportant des rues parallèles et perpendiculaires à l'exemple des rues du quartier de Manhattan à New York. Aussi, l'objet de ce stage est l'étude des types de voirie "Manhattan" par des méthodes de géométrie stochastique afin d'y estimer les distributions de distance entre les noeuds du réseau ainsi que l'éligibilité des clients potentiels aux services proposés par Orange.

2 Introduction

2.1 Contexte

Orange est l'une des plus grands groupes de télécommunications dans le monde. Le groupe emploie plus de 155 000 personnes dont plus d'un tiers à l'étranger. Présent dans 29 pays différents, Orange compte près de 263 millions de client pour un chiffre d'affaire de plus de 40 milliards d'euros en 2015. Orange est particulièrement présent en Afrique et au Moyen-Orient, où l'entreprise est en forte croissance grâce à son implémentation dans 20 pays et à ses 110 millions de client. Par ailleurs, en France, l'entreprise s'est fixée l'objectif d'apporter la fibre optique à l'ensemble du territoire d'ici 2022.

Ainsi, Orange doit fortement investir dans le déploiement de fibres optiques dans de nombreux territoires, ce qui apporte de nouvelles problématiques, notamment car les villes d'Afrique ne sont pas construites de la même manière que les villes européennes. Beaucoup de villes Africaines sont en effet récentes et possèdent par conséquent une structure beaucoup plus planifiée que les villes européennes qui se sont construites sur de très longues périodes historiques.

Aussi observe-t-on dans ces villes des quartiers aux rues très régulières, presque toutes parallèles ou perpendiculaires les unes aux autres, formant des pâtés de maison rectangulaires (fig 2.1). On nommera par la suite cette structure "Manhattan" en référence au quartier de New York aux Etats-Unis, construit de manière similaire.

Le réseau internet de l'opérateur est organisé suivant une hiérarchie de nœuds de tailles et de fonctions variables. Plus le niveau du nœud est bas, plus ils est proche du domicile du client. Comme les fibres optiques sont majoritairement installées suivant le tracé de la voirie entre les nœuds et les résidences, la structure de celle-ci est un élément important

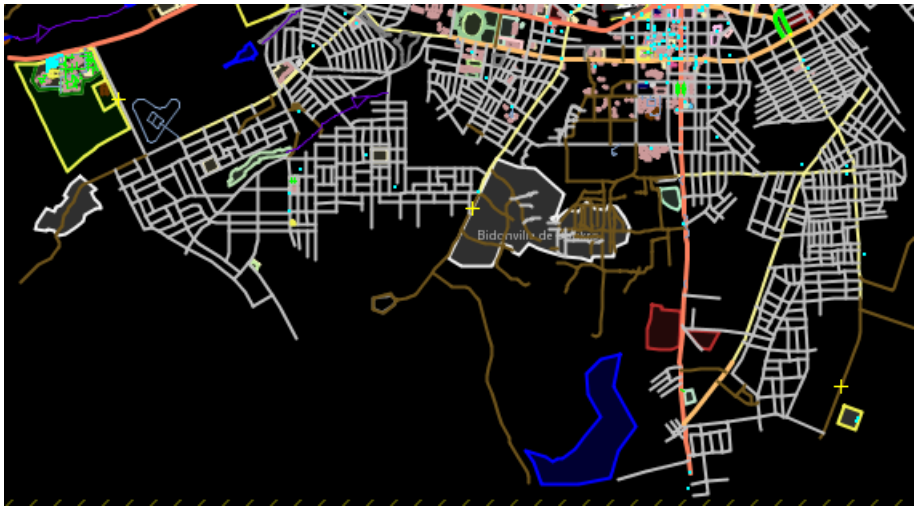

FIGURE 2.1 – Ville de Bouake, Côte d'Ivoire, source : Open Street Map

dans l'estimation des coûts liés au câblage et à la maintenance du réseau de fibre optique. Connaissant, la distribution des distances entre les domiciles des clients et le nœud du réseau, on peut en déduire les coûts réels de câblage ainsi que l'éligibilité des citadins aux services proposés par Orange.

2.2 Orange Labs Network

Orange Labs Network (OLN) est l'entité dédiée aux réseaux. Ses principales missions sont :

1. définir la stratégie réseau du Groupe, en anticipant les ruptures technologiques, l'impact des nouveaux produits et services et les éventuels changements de modèles opérationnels sur les réseaux ;
2. alimenter la stratégie de recherche et piloter les projets relatifs à Orange Labs Networks, sous la résponsabilité d'Orange Labs Recherche ;
3. définir, lancer et accompagner les grands programmes de transformation des réseaux du Groupe ;
4. Organiser et coordonner les travaux d'architecture ;

Au sein de la division Network Modelling & Planning, j'ai rejoint l'équipe Modelling & Statistical Analysis (MSA) qui effectue notamment des activités de recherche sur l'optimisation des déploiements réseaux, des modélisations de la consommation énergétique et des études sur le trafic.

L'équipe travaille depuis plusieurs années sur des modèles macroscopiques de représentation de la voirie de la ville basés sur des principes de géométrie stochastique[4],exploitant la géographie du terrain et les caractéristiques du réseau qui y sera déployé.

Les rues sont ainsi remplacées par le modèle de tessellation aléatoire dont la morphologie se rapproche le plus de la voirie réelle[2]. Les noeuds du réseau sont répartis selon un processus ponctuel de Poisson le long des rues. Chaque noeud du réseau possède une zone d'action qu'on identifie à une cellule typique du modèle retenu. Les caractéristiques de ces modèles aléatoires étant connues, il est possible d'obtenir rapidement une expression paramètrique de la distribution de distance entre le noeud de la cellule typique et les noeuds de niveau inférieur.

L'équipe a de plus mis au point deux logiciels :

1. OsmMiner, qui réalise la segmentation de la ville en quartiers homogènes et, pour chaque quartier, l'identification du modèle aléatoire le plus proche de la voirie réelle. OsmMiner utilise la base de données géographiques libre OpenStreetMap pour récupérer les données relatives à la voirie des villes.
2. NTStools, qui renvoie, à partir de la segmentation réalisée par OsmMiner, la distribution de distance entre les noeuds de niveau supérieur et les noeuds de niveau inférieur ainsi que la distribution de distance entre les noeuds de niveau inférieur et les domiciles des clients dans chaque partie segmentée de la ville. On peut déduire de ces distributions une estimation des coûts de câblage, l'éligibilité des clients aux services d'Orange et l'atténuation du signal dans chaque quartier.

L'intérêt de mes travaux, basés en grande partie sur les résultats de Thomas Courtat[1], est d'apporter un nouveau modèle de représentation de la voirie, le modèle de Manhattan, qui présente la particularité d'être anisotrope. Il pourra être utilisé pour modéliser des voiries dans les villes où Orange cherche à s'implanter, particulièrement en Afrique.

3 Géométrie stochastique

3.1 Processus ponctuel

Un **processus ponctuel** sur $\mathbb{R}^d$ est une variable aléatoire à valeur dans $(\mathbb{T}, \mathscr{F})$ où $\mathbb{T}$ est la famille de toutes les suites ϕ de points de $\mathbb{R}^d$ vérifiant la condition suivante :

1. la suite ϕ est localement finie (i.e. toute sous-partie bornée de $\mathbb{R}^d$ contient un nombre fini de points de ϕ) ;
2. $\forall i \neq j$, $x_i \neq x_j$ (on dit que le processus est simple, mais cette condition n'est pas nécessaire).

On note aussi $\Phi = (X_i)_{i\in\mathbb{N}}$ où les X_i sont des variables aléatoires dans $\mathbb{R}^d$.

Un processus ponctuel Φ induit donc une distribution sur $(\mathbb{T}, \mathscr{F})$, la distribution P de Φ.
Pour B borélien fixé, on note $\Phi(\omega, B)$ le nombre de points de la réalisation $\Phi(\omega)$ dans le borélien B et on appelle variable de comptage la variable aléatoire $\Phi(B) : \omega \to \Phi(\omega, B)$.

D'après le **théorème de Kolmogorov**, la loi de Φ est entièrement déterminée par les lois des $(\Phi(B_1), ..., \Phi(B_n))$ pour $(B_1, ..., B_n)$ boréliens bornés.

Soit Λ une mesure sur $\mathbb{R}^d$ telle que pour tout ensemble mesurable borné A, $\Lambda(A)$ soit fini. Λ est dite de Radon.
On appelle processus ponctuel de Poisson d'intensité Λ sur $\mathbb{R}^d$ le processus ponctuel Φ tel que pour tout k-uplet de boréliens bornés mutuellement disjoints $(B_1, ..., B_k)$ et tous $(n_1, ..., n_k) \in \mathbb{N}^k$:

$$\mathbb{P}(\Phi(B_1) = n_1, ..., \Phi(B_k) = n_k) = \prod_{i=1}^{k} e^{-\Lambda(B_i)} \frac{(\Lambda(B_i))^{n_i}}{n_i!}$$

On dit que Φ est homogène si et seulement si $\Lambda(B) = \lambda|B|$ avec $\lambda \in \mathbb{R}^+$

Un processus ponctuel Φ est dit **stationnaire** si pour tout $y \in \mathbb{R}^d$, $\Phi = (X_i)$ et $\Phi_y = (X_i + y)$ possèdent la même distribution.
Un processus est **isotrope** si ses caractéristiques sont invariantes par rotation, c'est à dire si pour toute rotation **r** autour de l'origine Φ et $\mathbf{r}\Phi$ ont la même distribution.

Dans notre étude, nous nous limiterons aux processus stationnaires sur $\mathbb{R}^2$ On peut ainsi utiliser un processus ponctuel de Poisson sur $\mathbb{R}^2$ pour répartir les noeuds du réseau sur le territoire mais aussi un processus ponctuel sur $\mathbb{R}$ pour répartir les noeuds sur la voirie. C'est cette deuxième méthode qui sera le plus souvent utilisée.

3.2 Diagramme de Voronoï

Soient N un ensemble de points de $\mathbb{R}^n$, d une distance sur l'espace $\mathbb{A}$ et $p \in N$. On appelle région de Voronoï de p l'ensemble des points qui sont plus proches de p que de tout autre point de N :
$Vor_N(p) = \{x \in \mathbb{A} \text{ tq } \forall q \in N \ d(x,p) \leq d(x,q)\}$

Il faut noter que le diagramme de Voronoï peut être construit pour n'importe quelle distance. Même si nous utiliserons la distance euclidienne la majorité du temps, il est aussi intéressant d'utiliser la "distance de Manhattan ", c'est à dire la distance $d : (a,b) = \sum_{i=1}^{n} |a_i - b_i|$.

3.3 Tessellations aléatoires simples sur $\mathbb{R}^2$

Une tessellation (ou mosaïque) est une suite $(C_1, C_2, ...)$ de polygones compacts(ou cellules) tels que $\cup(C_i) = \mathbb{R}^2$ et $\forall i \neq j$, $Int(C_i) \cap Int(C_j) = \emptyset$.

Une tessellation aléatoire est une tessellation où les C_i sont des fermés aléatoires. Elle est stationnaire si sa distribution est invariante par translation et isotrope si sa distribution est invariante par rotation. Voici quelques exemples de tessellations aléatoires et stationnaires classiques sur $\mathbb{R}^2$(fig 3.1) :

1. **Tessellation Poisson Ligne (PLT)** : On tire des points selon un processus de poisson sur droite. Puis, on choisit pour chaque point une direction uniformément

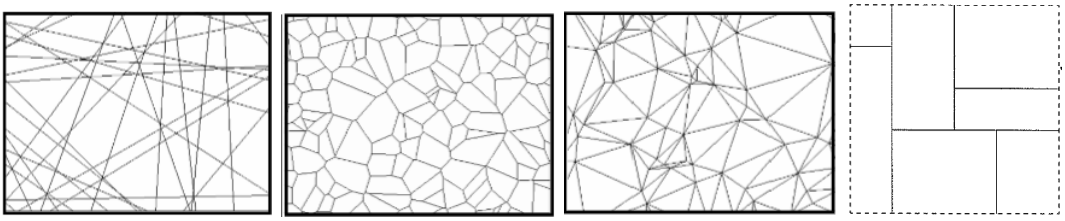

FIGURE 3.1 – De gauche à droite, exemples de tessellations PLT, PVT, PDT [1] et STIT anisotrope composé uniquement de segments horizontaux et verticaux[5].

dans l'intervalle [0,2π] et on trace la droite passant par le point avec la direction obtenue ;

2. **Tessallation Poisson Voronoi (PVT)** : On tire des points selon un processus de Poisson ponctuel sur $\mathbb{R}^2$ et on trace le diagramme de Voronoi associé à cet ensemble de points.
3. **Tessellation Poisson Delaunay (PDT)** : On tire des points selon un processus de Poisson ponctuel sur $\mathbb{R}^2$ et on trace la triangulation de Delaunay associée à cet ensemble de points.
4. **Stable with respect to iteration (STIT)** : C'est une mosaïque construite par itération successive d'un processus de division[5]. Une mesure sur des compacts de $\mathbb{R}^2$ est nécessaire. A une étape donnée, chaque cellule de la tessellation actuelle est divisée par un segment aléatoire (l'angle de ce segment étant choisi par une fonction d'isotropie ϕ auxiliaire) avec une probabilité proportionnelle à sa mesure. L'itération s'arrête après un certain fixé initialement.

Les 3 premières tessellations (ainsi que la mosaïque STIT dans le cas isotrope) ont la particularité d'être décrites par un seul paramètre (l'intensité du processus), ce qui facilite la modélisation.

3.4 Probabilité de Palm et objet typique

La notion d'objet typique essaie de donner un sens à l'idée de l'objet moyen. L'objet typique est une variable aléatoire sur l'ensemble des objets qui a en moyenne toutes les propriétés de l'ensemble ainsi que les corrélations entre ces propriétés. Formellement, on peut le définir ainsi :
Soient $C = (c_i)$ un processus aléatoire d'objet stationnaire (par exemple une cellule) sur un

espace mesurable A $\subset \mathbb{R}^2$, et μ_2 la mesure de Lebesgue en dimension 2. Alors il existe une variable aléatoire typique c^* tel que, pour toute surface W bornée et pour toute fonction f mesurable sur A et invariante par translation, on ait :

$$\mathbb{E}(\frac{\sum_{c_i \in C \cap W} f(c_i)}{card(X \cap W)}) = \mathbb{E}(f(c^*)) \tag{3.1}$$

avec $\lambda_0 = \mathbb{E}(\frac{card(C \cap W)}{\mu_2(W)})$, on peut aussi écrire cela sous la forme :

$$\frac{1}{\lambda_0 \mu_2(W)} \mathbb{E}(\sum_{c_i \in C \cap W} f(c_i)) = \mathbb{E}(f(c^*)) \tag{3.2}$$

Ainsi si l'on choisit une réalisation au hasard ω parmi les objets de $C \cap W$, alors :

$$\mathbb{E}(f(\omega)) \rightarrow_{W \rightarrow \mathbb{R}^2} \mathbb{E}(f(c^*))$$

La théorie de Palm permet de définir une famille de mesures $(\mathscr{P}_x)_{x \in \mathbb{R}^2}$ sur $(\mathbb{T}, \mathscr{F})$, qui permet d'évaluer le processus lui-même et pas seulement ses réalisations.
Cette notion est notamment importante pour évaluer les propriétés de la cellule typique. En effet si l'on se contentait de choisir la cellule contenant par exemple l'origine du repère, on introduirait un biais dû au fait que les cellules plus grosses ont plus de chance de contenir l'origine.

Dans les cas des processus de Poisson, **la formule de Slivnyak (ou de Mecke)** nous permet d'obtenir la cellule typique sans générer le processus total. En effet, si Φ est un processus de Poisson d'intensité λ, alors :

$$\mathbb{E}(\sum_{x \in X} f(x, X)) = \lambda \mathbb{E}(f(0, X + x)) \tag{3.3}$$

ou encore

$$\mathscr{P}_x(.) = P_{\Phi + \delta_x}(.), \tag{3.4}$$

où $P_{\Phi+\delta_x}(.)$ est la loi du processus ponctuel $\Phi + \delta_x$

Ce théorème formalise l'idée qu'ajouter un seul point à un processus de Poisson ne modifie pas les caractéristiques globales du processus.

3.5 Paramètres des tessellations

Une tessellation produit d'autres processus aléatoires. On nomme centroïde d'une face F d'une tessellation T un unique point c(F) de F tel que si T' est la tessellation translatée de T par un vecteur $y \in \mathbb{R}^2$, c(F+y)=c(F)+y est le centroïde de la face F+y de T'. Les différents paramètres à étudier sont :

1. N_{02} nombre moyen de cellules contiguës sur le sommet typique ;
2. N_{20} nombre moyen de sommets que possède la cellule typique ;
3. l_0 longueur totale des arêtes issus d'un sommet typique ;
4. l_1 longueur moyenne d'une arête ;
5. λ_0 densité surfacique de sommets ;
6. λ_1 densité surfacique d'arêtes ;
7. λ_2 densité surfacique des centroïdes ;
8. $\lambda_3 = L_A$ intensité de longueur (somme des longueurs d'arêtes par surface unitaire) ;
9. U_2 périmètre moyen d'une cellule ;
10. A_2 surface moyenne d'une cellule ;

On introduit par ailleurs ξ le paramètre d'isotropie défini de la manière suivante :

$$\xi = \iint_0^{2\pi} |sin(u_1 - u_2)| \mu(d_{u_1}) \mu(d_{u_2}) \tag{3.5}$$

avec μ la mesure angulaire de la tessellation.

Remarque : dans le cas isotrope $\xi = \frac{2}{\pi}$ et dans le cas Manhattan $\xi = \frac{1}{2}$

On dispose de plusieurs relations intuitives entre ces différents paramètres dans le cas de tessellations stationnaires :

$$\lambda_2 A_2 = 1 \tag{3.6}$$

$$l_0 \lambda_0 = 2 L_A \tag{3.7}$$

	PVT	PDT	PLT	STIT
λ_0	$\frac{1}{2}$	$\frac{9\pi^2}{1024}$ (0.0867)	$\frac{1}{\pi}$ (0.3183)	ξ
λ_1	$\frac{3}{4}$	$\frac{27\pi^2}{1024}$ (0.2602)	$\frac{2}{\pi}$ (0.6366)	$\frac{3}{2}\xi$
λ_2	$\frac{1}{4}$	$\frac{18\pi^2}{1024}$ (0.1735)	$\frac{1}{\pi}$ (0.3183)	$\frac{1}{2}\xi$
$\lambda_3 = L_A$	1	1	1	1

TABLE 3.1 – Valeurs du vecteur $(\lambda_0, \lambda_1, \lambda_2, \lambda_3^2)$ pour les tessellations stationnaires précédemment décrites. λ_3^2 est normalisé à 1, ce qui revient à comparer les modèles pour une même densité de voirie.

$$U_2\lambda_2 = 2L_A \tag{3.8}$$

$$l_1\lambda_1 = 1 \tag{3.9}$$

$$N_{20}\lambda_2 = 2\lambda_1 \tag{3.10}$$

$$N_{02}\lambda_0 = 2\lambda_1 \tag{3.11}$$

Enfin la **formule d'Euler-Poincaré** relie le nombre de sommets n_s, le nombre d'arêtes n_a et le nombre de faces n_f d'un polyèdre à 3 dimensions :

$$n_s + n_f - n_a = 2 \tag{3.12}$$

En considérant une tessellation comme la surface limite d'un polyèdre dépliée, on obtient :

$$\lambda_1 = \lambda_0 + \lambda_2 \tag{3.13}$$

Il en résulte qu'une tessellation stationnaire est caractérisée par 3 de ces valeurs seulement. En pratique, on utilisera $\lambda_0, \lambda_1, \lambda_2$ ainsi que λ_3 pour des raisons historiques : avant l'utilisation d'ordinateurs, il fallait faire tous ces calculs manuellement à partir de cartes. Les paramètres les plus simples à compter ont donc été retenus, ainsi qu'un quatrième qui permettait de vérifier la présence éventuelle d'erreurs.

3.6 Processus de Cox

Un processus de Cox (ou de Poisson-Cox) est un processus ponctuel généralisant le processus ponctuel de Poisson dans lequel l'intensité du processus stochastique est elle-même

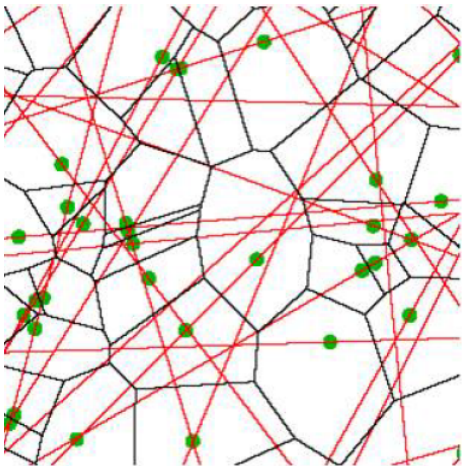

FIGURE 3.2 – Le réseau routier est remplacé par un PLT (en rouge). Les noeuds du réseau sont ensuite simulés par un processus de Poisson linéaire sur les segments du PLT (en vert), qui est donc un processus de Cox. En noir est représentée la région de Voronoï de chaque noeud.[7]

une variable aléatoire.
Par exemple soient Φ une tessellation aléatoire stationnaire et Ψ un processus ponctuel de Poisson sur les segments de Φ. Ψ est alors un processus ponctuel de Poisson linéaire. La variable aléatoire décrivant les réalisations possibles de ce processus est un processus de Cox, dépendant donc des paramètres des deux processus Φ et Ψ.

3.7 Paramètres de la tessellation de Manhattan

La tessellation de Manhattan sera générée par un processus de Cox. On notera L_1 et L_2 la hauteur et la longueur d'un rectangle de Manhattan, c'est à dire la moyenne des écarts entre les rues verticales et horizontales. L_1 et L_2 jouent des rôles symétriques, on peut échanger leur rôle sans modifier la tessellation. Sur cette voirie de type Manhattan seront ensuite tirés des points selon un processus ponctuel de Poisson linéaire d'intensité λ. La tessellation finalement étudiée correspondra à la voirie de Manhattan contenue dans la région de Voronoï d'un de ces point.

Il découle de ces notations que l'aire d'un rectangle élémentaire de Manhattan est L_1L_2 et son périmètre $2(L_1+L_2)$. Ainsi dans une surface de taille L_1L_2, on trouvera en moyenne

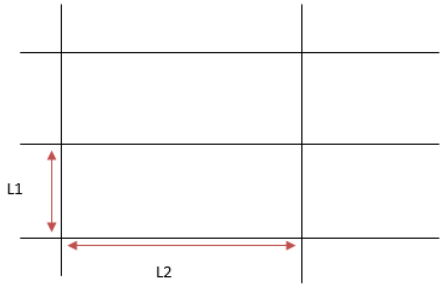

FIGURE 3.3 – illustration d'une petite mosaïque de Manhattan

une face, deux milieux d'arête, un sommet ainsi qu'une longueur totale de $(L_1 + L_2)$. D'où les paramètres suivant :

$$\lambda_0 = \frac{1}{L_1 L_2} \tag{3.14}$$

$$\lambda_1 = \frac{2}{L_1 L_2} \tag{3.15}$$

$$\lambda_2 = \frac{1}{L_1 L_2} \tag{3.16}$$

$$\lambda_3 = \frac{L_1 + L_2}{L_1 L_2} \tag{3.17}$$

En normalisant $\lambda_3^2 = L_A^2 = 1$, on fait apparaître le vecteur d'intensité suivant :

$$\Lambda_{Man} = \begin{pmatrix} \frac{L_1 L_2}{(L_1+L_2)^2} \\ \frac{2L_1 L_2}{(L_1+L_2)^2} \\ \frac{L_1 L_2}{(L_1+L_2)^2} \\ 1 \end{pmatrix}$$

Une rapide étude de la fonction $(x, y) :\rightarrow \frac{xy}{(x+y)^2}$ montre qu'elle atteint un maximum de 0.25 pour x=y. Pour un Manhattan homogène, on trouvera donc le vecteur $\begin{pmatrix} 0.25 \\ 0.5 \\ 0.25 \\ 1 \end{pmatrix}$.

On introduit enfin le paramètre κ permettant de décrire la densité de la voirie à l'intérieure de la cellule typique, indépendemment de la taille réelle de celle-ci. Il est défini comme le rapport entre la somme des longueurs d'arêtes par unité de surface L_A et l'intensité linéaire du processus ponctuel λ. C'est donc un paramètre adimensionné, qui sert de valeur d'étalonnage dans tous les modèles employés.

$$\kappa = \frac{L_A}{\lambda} = \frac{(L_1 + L_2)}{\lambda L_1 L_2} \tag{3.18}$$

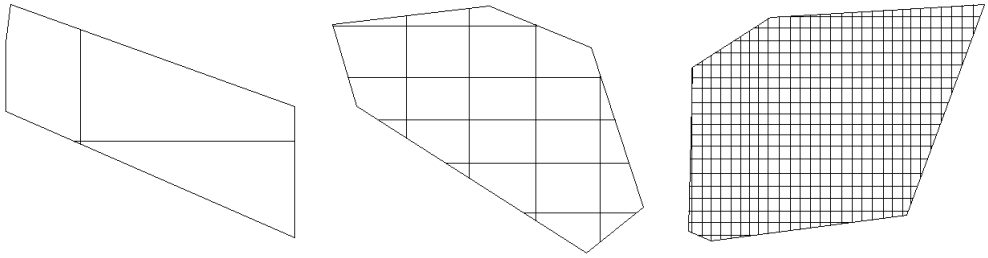

FIGURE 3.4 – Trois différentes voiries de Manhattan pour des valeurs respectives de κ de 6,50 et 2000. Kappa permet de décrire l'intensité de la voirie dans la cellule, indépendamment de l'aire de la cellule.

4 Modélisation de la ville

Dans ce chapitre, nous allons montrer comment on utilise les principes de géométrie stochastiques précédemment exposés pour modéliser la voirie d'un territoire urbain.

4.1 Structure de graphe

Une représentation intuitive de la voirie d'une ville consiste à la décrire par un graphe où les sommets représentent les intersections et les arrêtes les rues.
Soient $\mathbb{A}$ l'espace total disponible (pour une ville par exemple) un sous-ensemble compact de $\mathbb{R}^2$, V une sous-partie finie de $\mathbb{A}$ et E un ensemble de chemins dérivables presque partout inclus dans A reliant deux éléments de V. Alors G=(V,E) est un graphe. À un graphe G, on peut associer sa projection géométrique dans le plan $\mathbb{A}$ π_G :
$\pi_{\mathbb{G}} = \{x \in \mathbb{A}, \exists e \in E \; x \in e\}$

Ainsi, on peut se servir des distances usuelles sur $\mathbb{R}^2$ pour décrire les distances entre les points du graphe. Comme une même rue être représentée par plusieurs segments, on y rajoute une structure d'hypergraphe pour montrer l'appartenance des segments à une même rue.

4.2 Segmentation

L'étape suivante consiste à segmenter la ville (et le graphe associé) en quartiers homogènes qui seront ensuite "remplacés" par un modèle de tessellation stationnaire.

On commence par générer un nombre important de points sur la voirie par un processus ponctuel de Poisson. Ce grand nombre permet d'éviter des segmentations trop inégales

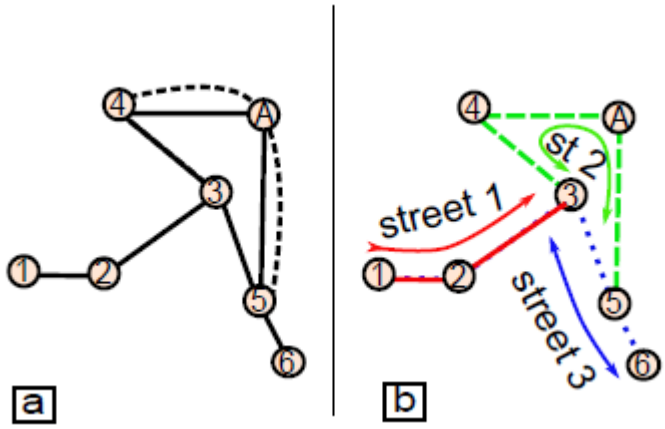

FIGURE 4.1 – Dans l'exemple ci-dessus, le graphe a est un graphe normal tandis que le graphe b possède une structure d'hypergraphe. Ainsi on montre que le graphe b possède 3 rues mais 7 segments de rue.[1]

de la ville (avec par exemple plusieurs points les uns à côté des autres qui créeraient des quartiers de tailles trop différentes). Ainsi, en notant λ l'intensité du processus, r une rue de la ville, N_r le nombre de noeuds sur cette rue et l_r sa longueur, on a :

$$\mathbb{P}(N_r = k) = e^{-\lambda l_r} \frac{(\lambda l_r)^k}{k!}$$

Pour la simulation, on utilisera le fait que la distance entre deux points d'un processus de Poisson linéaire suit une loi exponentielle de paramètre λ.
Il reste à segmenter la ville. Pour ce faire, nous allons construire le diagramme de Voronoï associé aux points générés (fig 4.2). La ville est alors découpée en de nombreux petits quartiers qu'il faut maintenant rassembler.

On crée ensuite une matrice de "distances" entre ces quartiers. Celle-ci prend en compte la distance réelle entre les quartiers, le nombre moyen de segments de rue par intersections, la densité de la voirie ainsi que des paramètres d'isotropie et de longueur. Cela permet d'évaluer quels sont les quartiers qui se ressemblent le plus et enfin, de les fusionner par une méthode de spectral clustering (pour plus de détails, la méthode est décrite de manière approfondie dans le chapitre 10 de la thèse de Thomas Courtat[1]). La segmentation finale permet de révéler des quartiers homogènes dans la ville(fig 4.2).

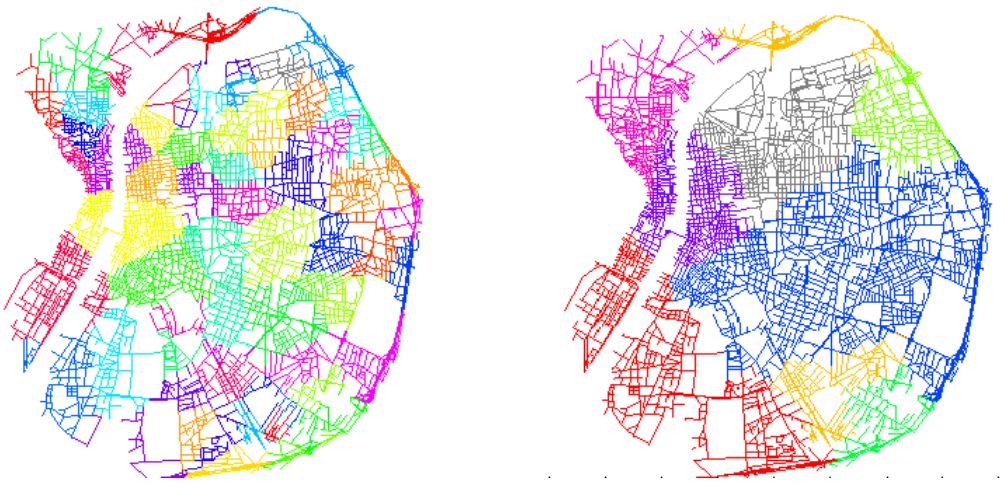

FIGURE 4.2 – A gauche, segmentation initiale de la ville de Lyon ; à droite, segmentation finale de la ville de Lyon en 8 quartiers.

4.3 Choix du modèle de tessellation

Le découpage de la ville réalisé, il faut maintenant associer à chaque quartier le modèle de tessellation aléatoire se rapprochant le plus de la voirie réelle. Nous avons vu précédemment que le vecteur $\Lambda = (\lambda_0, \lambda_1, \lambda_2, lambda_3^2)$ suffit à décrire une tessellation stationnaire, c'est donc en comparant le vecteur de la voirie réelle aux vecteurs des modèles connus que nous choisirons le modèle qui représentera le quartier. On compare ainsi la grandeur suivante pour chaque tessellation i :

$$C_i = \min_{t \in \mathbb{R}_+} \|t.\overrightarrow{\Lambda_{modele}} - \overrightarrow{\Lambda_{reel}}\|^2 \tag{4.1}$$

Le modèle retenu est celui qui minimise C. Le paramètre t est conservé et sert à caractériser l'intensité de la voirie dans le quartier. Finalement, la ville est segmenté en un nombre de quartiers homogènes et un modèle leur a été attribué.

5 Mosaïque de Manhattan

Afin d'obtenir des résultats empiriques sur les cellules typiques de tessellations Manhattan et de modéliser statistiquement les distances à l'intérieur de ces tessellations, j'ai conçu un algorithme de génération de cellules typiques de Manhattan sur Matlab.

5.1 Simulation de cellules typiques

5.1.1 Dans le cas général

La difficulté est de pouvoir générer une cellule typique sans avoir à produire une tessellation complète sur $\mathbb{R}^2$ puis de savoir comment choisir une cellule parmi toutes celles que l'on génère. Pour cela, nous allons nous servir de la formule de Slivnyak. Ainsi, l' idée générale consiste à placer initialement un point O en (0,0) et à simuler radialement de nouveaux points par le processus de Cox[2] en s'éloignant de O. A chaque étape de la simulation, on recalcule la région de Voronoï de ce point. A partir d'une certaine distance par rapport à O, tout point supplémentaire du processus ne peut plus modifier la région de Voronoï et on peut alors arrêter la simulation.

5.1.2 Simulation de Cox-Manhattan

En résumé, il s'agit de générer une voirie de Manhattan(la direction des droites appartenant à $\{0, \pi/2\}$, puis de simuler des points avec un processus ponctuel de Poisson linéaire de paramètre λ sur la voirie créée. La formule de Slivnyak nous assure que la cellule générée par le point (0,0) est une cellule typique.

Les pavages de type Manhattan sont assez réguliers car résultats d'une planification humaine. Or, dans le cas où les rues sont tirées au sort selon un processus de Poisson, on observe une variance importante de la longueur des rues, ce qui n'est pas le cas dans la

réalité. J'ai donc choisi de créer un simulateur où on l'on peut aussi générer la longueur des segments de rues selon une loi normale. Dans ce cas, la variance de la loi est choisie égale à un dixième de son espérance. Cela permet de générer des rues beaucoup plus régulières.

L'algorithme prend en **entrée** :

1. L' espérance des écarts horizontaux entre les intersections L1 ;
2. L' espérance des écarts verticaux entre les intersections L2 ;
3. L' intensité des centroïdes sur la voirie λ ;
4. p facteur de « remplissage » valeur comprise entre 0 et 1. Lors de la création du graphe de la voirie, avec probabilité p on trace l'arrête reliant deux intersections. Ainsi p=1 donne un graphe régulier où les sommets sont tous de degré 4 ;
5. le choix de la distance (euclidienne ou Manhattan) pour le diagramme de Voronoï.

En **sortie**, il renvoie :

1. Le contour de la cellule typique sous forme d'un graphe
2. un vecteur contenant les ordonnées de toutes les rues horizontales passant par la cellule.
3. un vecteur contenant les abscisses de toutes les rues verticales passant par la cellule.
4. Le graphe de la voirie à l'intérieur de la cellule sous forme d'un graphe

Remarque : L'intérêt de stocker les abscisses et ordonnées à part est de pouvoir recréer rapidement la voirie de la cellule typique sans avoir à recommencer l'algorithme.

Algorithme détaillé : L'origine (0,0) servira de centre pour la cellule typique. On commence par tracer la première droite passant par l'origine. Celle-ci est aléatoirement verticale ou horizontale (ie l'angle θ initial est congru à 0 $[\pi/2]$. Sur cette droite, de chaque côté en partant de l'origine un centroïde est placé suivant une loi exponentielle de paramètre λ.

Premièrement, on choisit la loi probabiliste qui doit décrire les écarts entre les rues. On peut utiliser une loi exponentielle ou une loi normale. Puis, $\forall j \in \{1,2,3,4\}$, on initialise $R_0^j = 0$. Ensuite, on répète les étapes suivantes :

1. $\forall j \in \{1,2,3,4\}$ on calcule $R_i^j = R_{i-1}^j + T_i^j$ où T_i^j suit une loi normale ou exponentielle (selon le choix de l'utilisateur au début de la simulation) d'espérance L1 ou L2 selon qu'il s'agisse de la longueur associée à une droite verticale ou horizontale.

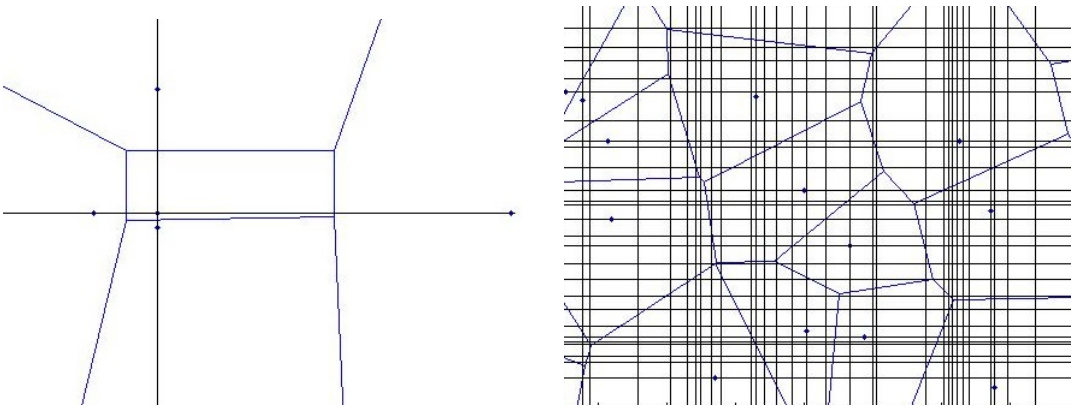

FIGURE 5.1 – A gauche, le diagramme de Voronoï formé par les 4 premiers centroïdes. A droite, diagramme de Voronoi final superposé à la voirie

2. On trace ensuite les droites horizontales passant par les points (0,$-R_i^1$) et (0,R_i^2) et les droites verticales passant par les points ($-R_i^3$,0) et (R_i^4,0) Ainsi on trace 4 droites à cette étape, une en haut,une en bas, une à gauche et une à droite de l'origine.
3. on stocke l'abscisse des droites verticales (respectivement l'ordonnée des droites horizontales) dans un vecteur ver (resp. hor)
4. Sur chaque nouvelle droite est tiré un nouveau centroïde selon une loi exp(λ)
5. on construit le nouveau diagramme de Voronoi des centroïdes.
6. On compare la distance Ri avec la taille de la cellule d'origine C0. Si dans chaque direction $R_i^j \geq 2 \max_{x \in C0} d(O, x)$, alors la cellule typique ne peut plus être modifiée par le tirage de nouveaux centroïdes et l'algorithme termine (fig 5.1).

On récupère ensuite la cellule typique et on la stocke sous la forme d'un graphe. Puis, la voirie est "nettoyée" : on ne conserve que les sommets à l'intérieur de la cellule. Puis on ré-indexe les arrêtes et celles-ci sont tronquées au bord de la cellule typique (fig 5.2). Le résultat apparaît sous la forme de deux graphes : l'un contenant le contour de la cellule et l'autre contenant la voirie. Le choix de ne conserver que la voirie située à l'intérieur de la cellule typique peut légèrement augmenter la distance entre le centroïde et des des points de la voirie situés près du bord de la cellule typique.

Éventuellement, si le facteur p est inférieur à 1, alors certaines arrêtes ne seront pas tracées, ce qui donne un type de voirie plus irrégulier, lacunaire.

5.2 Exemples de cellules typiques obtenues par simulation

Voici quelques exemples de cellules typiques obtenues par ce procédé ainsi que la voirie contenue à l'intérieur de la cellule dans différents cas (rues simulées selon une loi exponentielle ou normale, distance euclidienne ou "distance Manhattan" retenue pour le diagramme de Voronoi)

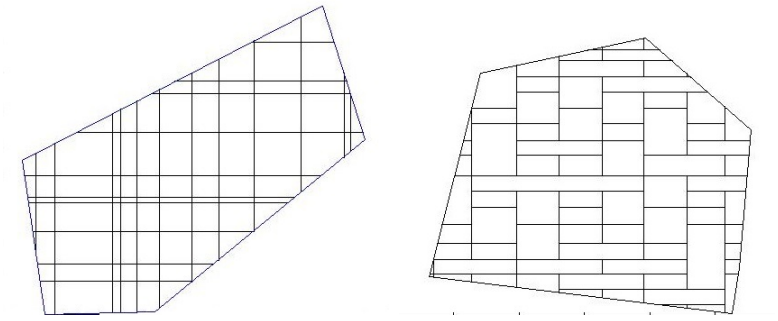

FIGURE 5.2 – Exemples de cellules typiques. A gauche, l'écart entre les rues suit une loi exponentielle tandis qu'à droite, les écarts suivent une loi normale. p=1 à gauche et p=0.5 à droite, la voirie est donc plus lacunaire dans la figure de droite.

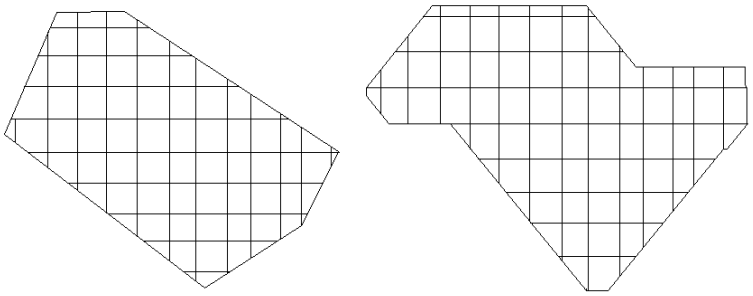

FIGURE 5.3 – Cellules typique simulées à gauche avec la distance euclidienne, à droite avec avec la "distance de Manhattan", pour le même processus ponctuel de Poisson. $L_1 = L_2 = 1, 1/\lambda = 100$

5.3 Validation de la simulation

En raison de la ressemblance de la tessellation de Manhattan avec la tessellation de type STIT, il est intéressant de comparer les valeurs des paramètres habituels (densité surfacique de sommets, d'arrêtes,etc. . .) obtenues par simulation avec celles, connues, d'une mosaïque de type STIT[5]. On observe de très bonnes correspondances.

Dans le cas du STIT, on dispose des relations suivantes :

$$N_{01} = N_{02} = 3 \tag{5.1}$$

$$N_{20} = N_{21} = 6 \tag{5.2}$$

$$\lambda_2 = \frac{1}{2}\xi L_A^2 \tag{5.3}$$

$$\lambda_1 = \frac{3}{2}\xi L_A^2 \tag{5.4}$$

$$\lambda_0 = \xi L_A^2 \tag{5.5}$$

$$U_2 = \frac{4}{\xi L_A} \tag{5.6}$$

$$A_2 = \frac{2}{\xi L_A^2} \tag{5.7}$$

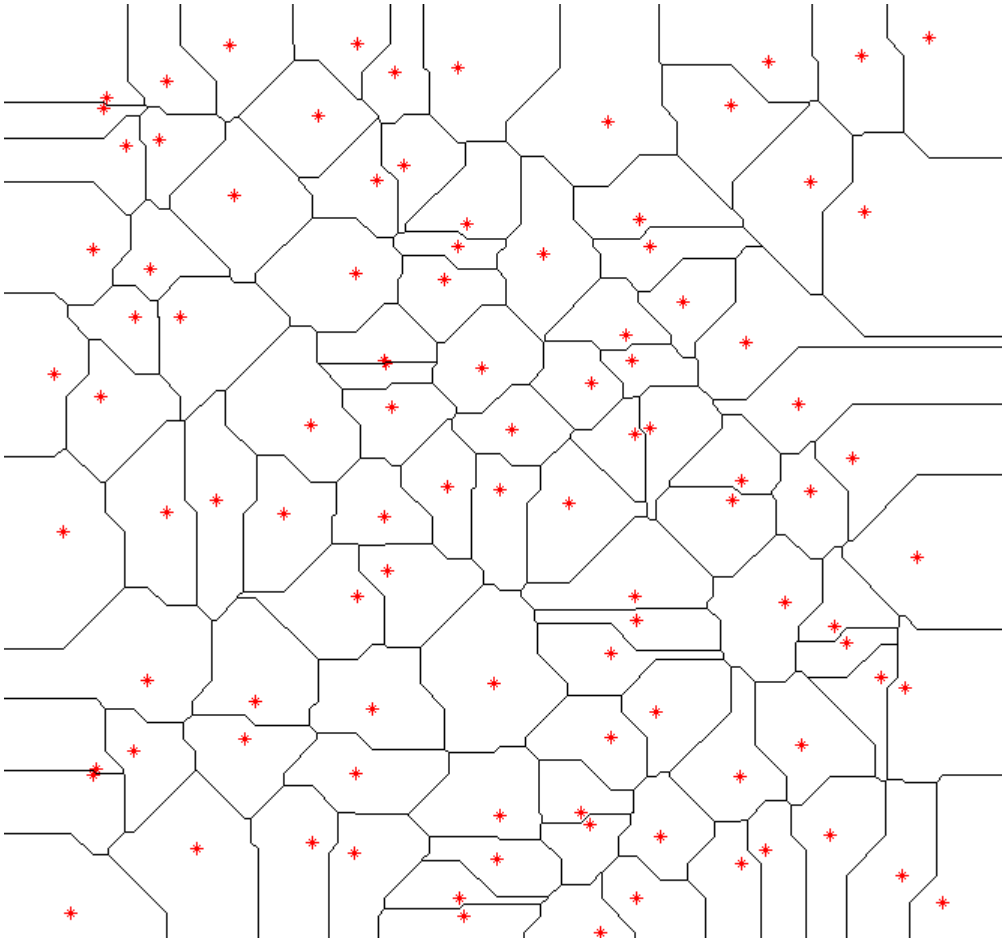

FIGURE 5.4 – diagramme de Voronoi avec la "distance de Manhattan" pour un processus de Poisson ponctuel. On remarque que la forme de la cellule typique change beaucoup. L'aire moyenne est la même (puisqu'elle dépend des centroïdes seulement). Le périmètre augmente de 10% en moyenne. Les arêtes sont beaucoup plus nombreuses et on remarque que leur direction est bien particulière, horizontale, verticale ou faisant un angle de 45 degrés avec ces directions. Le nombre moyen de côté passe d'un peu moins de 6 à plus de 13.

Dans le cas du modèle Manhattan, on trouve facilement

$$\lambda_2 = \frac{\lambda}{L_2} + \frac{\lambda}{L_1} = \frac{\lambda(L_1 + L_2)}{L_1 L_2} \tag{5.8}$$

D'où, en se basant sur les relations du STIT, les relations à vérifier suivantes :

$$L_A = \sqrt{\frac{\lambda(L_1 + L_2)}{(L_1 L_2)}} \tag{5.9}$$

$$U_2 = 2\sqrt{\frac{L_1 L_2}{\lambda(L_1 + L_2)}} \tag{5.10}$$

$$A_2 = \frac{L_1 L_2}{\lambda(L_1 + L_2)} \tag{5.11}$$

Par ailleurs, on obtient empiriquement $N_{20} = N_{21} \simeq 5.9$.

Pour vérifier si ces relations restaient vraies pour les cellules des mosaïques de Manhattan, on a effectué des simulations (10 000 cellules typiques générées par simulation) en faisant varier les paramètres λ et L_2, tandis que L_1 reste constant.

Les résultats obtenus sont encourageants même s'il faut noter un décrochage lorsque L1 et L2 sont de tailles très différentes (ce qui ne correspond pas à ce qu'on observe en réalité car L1 et L2 sont souvent proches) ou que la longueur de rue devient trop petite (i.e. $\kappa << 10$).Ces résultats tendent à montrer que les propriétés de la cellule typique d'une mosaïque de Manhattan suivent les mêmes relations que celles issues d'une tessellation STIT composée uniquement de segments verticaux et horizontaux.

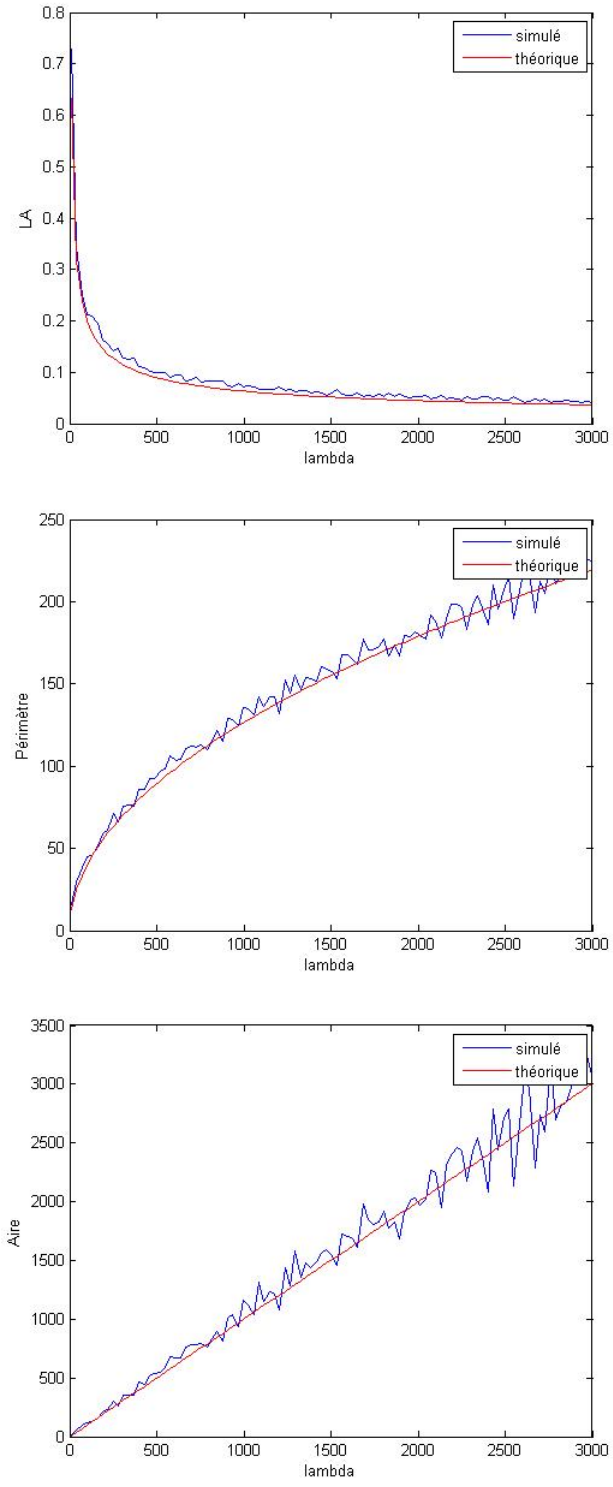

FIGURE 5.5 – Comparaisons de L_A intensité de longueur d'arêtes, du périmètre et de l'aire obtenus par la simulation et par les relations (5.9), (5.10) et (5.11) pour $1/\lambda$ variant de 10 à 3000, $L_1 = L_2 = 2$

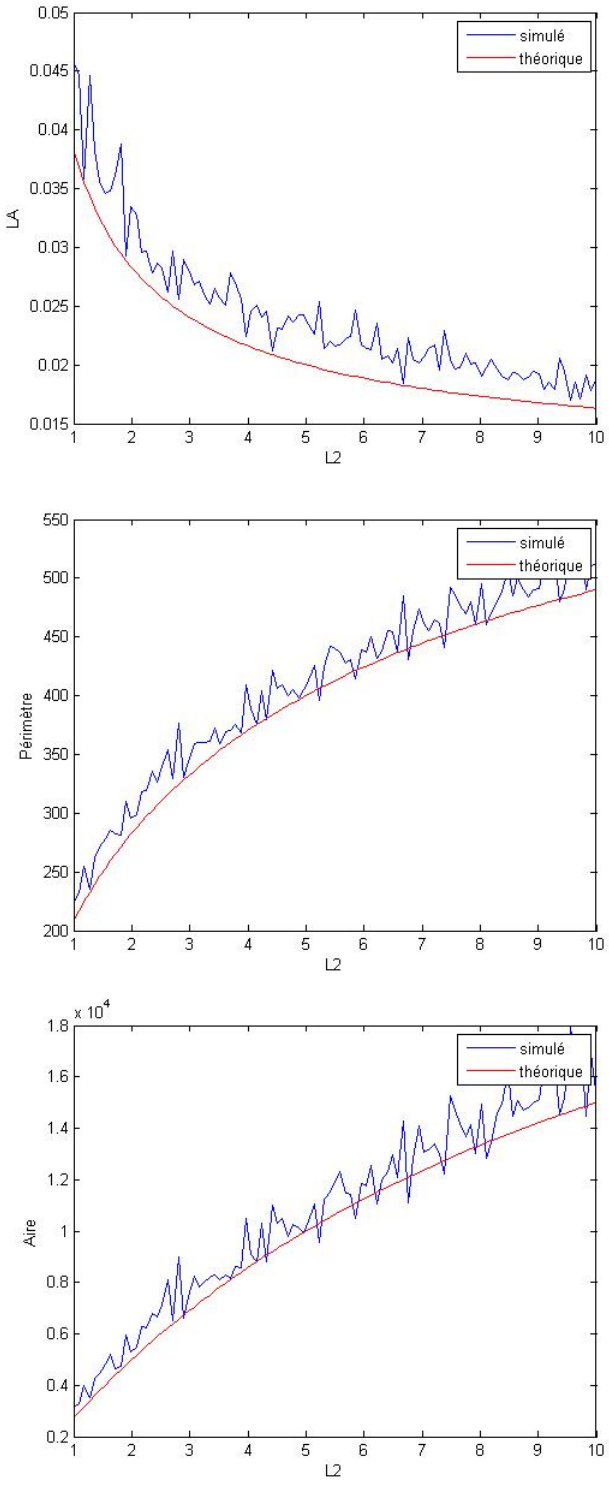

FIGURE 5.6 – Comparaison de L_A intensité de longueur d'arêtes, du périmètre et de l'aire obtenus par la simulation et par les relations (5.9), (5.10) et (5.11) pour $1/\lambda = 1000$, $L_1 = 10$,et L_2 variant de 1 à 10

6 Analyse statistique de la distribution des distances dans la cellule typique

Le but final du projet est d'obtenir la distribution de distance totale entre les noeuds de plus haut niveau et les domiciles des utilisateurs. Connaissant l'arborescence choisie dans une ville donnée (exemple en figure 6.1), on peut retrouver la distribution de distance totale en calculant le produit de convolution de plusieurs distributions : d'une part, la distribution de distance entre le noeud de plus bas niveau et les utilisateurs, d'autre part, les distributions de distance entre les noeuds de niveau k et k+1.

A partir de cette distribution totale, comme on connaît aussi les coûts fixes liés à chaque noeud et le prix linéique des câbles à poser, on peut en déduire des coûts de câblage pour toute la ville. De même, en connaissant les pertes linéiques des fibres optiques, et les pertes fixes de chaque noeud, on peut calculer la distribution totale de l'atténuation du signal sur une ville, d'où la détermination de l'éligibilité des clients potentiels.

Or, les utilisateurs sont à la fois très nombreux et les facteurs qui déterminent là où ils décident d'habiter sont aussi variés. Ainsi, il est logique de considérer statistiquement et non pas de façon déterministe la localisation des clients. On rend bien compte de la réalité en supposant que les clients sont répartis par un processus ponctuel de Poisson le long de la voirie.

Même si on connaît mieux les facteurs qui influencent le positionnement des noeuds sur la voirie, il existe néanmoins des contraintes variées et difficiles à prévoir qui font qu'en pratique, il n'est pas incohérent de considérer que les noeuds du réseau sont eux aussi répartis le long de la voirie selon un processus ponctuel de Poisson. En conséquence, les

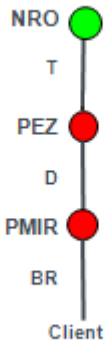

FIGURE 6.1 – Réseau hiérarchique comportant 3 niveaux de noeuds (source : orange)

différents noeuds et les utilisateurs peuvent être modélisés par un processus ponctuel de Poisson avec une intensité propre à chaque niveau.

Ainsi, connaître la distribution de distance entre le centroïde d'une cellule typique et des points générés uniformément sur la voirie nous permet en fait d'accéder à la distribution de distance entre le noeud de niveau k et les noeuds de niveau k-1 pour tout k $\leq n$ où n est le niveau maximal de l'arborescence. Néanmoins, pour obtenir la dernière distribution manquante, à savoir celle entre les noeuds de plus bas niveau et les clients, il y a un travail supplémentaire à effectuer car les clients sont beaucoup plus nombreux, ce qui permet de regrouper des câbles ensemble. Cette partie n'est pas étudiée dans le cadre de mon stage.

6.1 Première approche sur des cas simples (cellules rectangulaires)

Mes premières simulations ne portaient pas sur des cellules typiques mais sur des cellules rectangulaires donc non aléatoires, afin de me familiariser avec le sujet et d'obtenir des premiers résultats. Lors de ces premières simulations, les centroïde C_i sont tirés uniformément sur la voirie (de la même manière que les points (P_i)). On étudie la distribution des distances $d^{voirie}(C_i, P_i)$ c'est à dire la distance entre ces deux points en suivant la voirie.

Cette première simplification montre déjà une distribution de distance asymétrique. Avec L et l longueur et largeur totale de la tessellation, $L_1 = L_2 = 1$ longueur et largeur

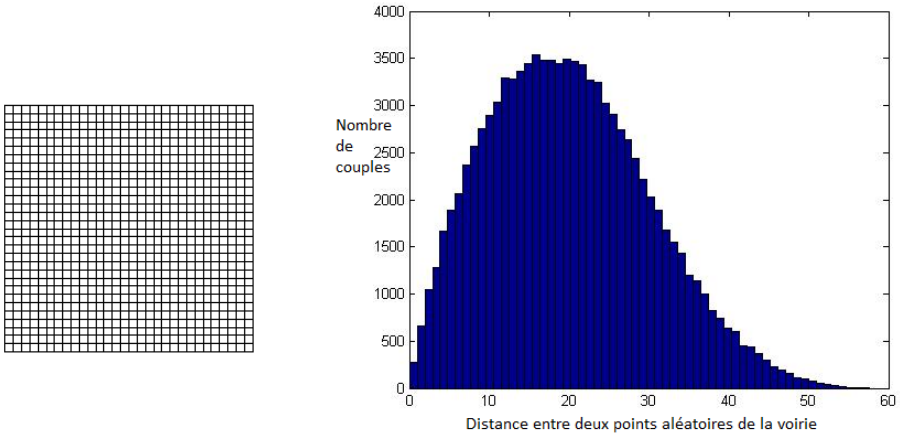

FIGURE 6.2 – Tessellation rectangulaire de taille 30x30 d'un Manhattan uniforme et histogramme joint des distances entre deux points tirés uniformément sur la voirie

des segments de rue, on observe de plus que $\mathbb{E}(X) = \frac{L+l}{3}$, ce qu'on peut vérifier par le calcul (annexe 8.2.1) et que le mode de la distribution est $x_m = \min(L, l)$. Quelques exemples de distribution obtenues sur ces cas simples sont proposées dans les figures 6.2,6.3 et 6.4.

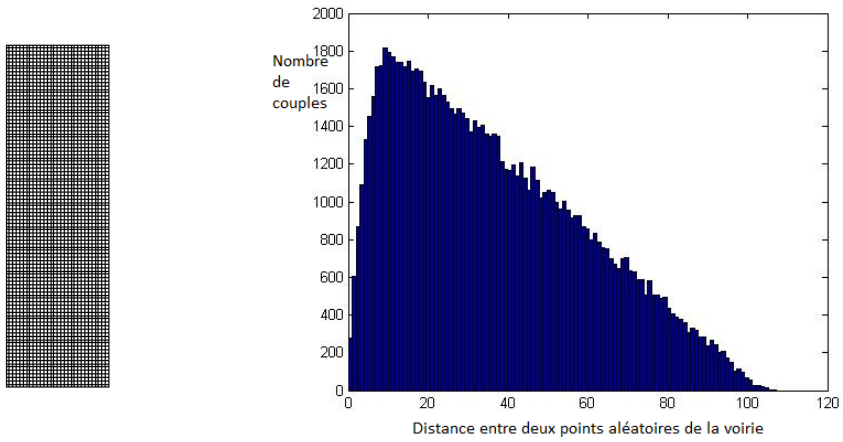

FIGURE 6.3 – Tessellation rectangulaire de taille 100x30 d'un Manhattan uniforme et histogramme joint des distances entre deux points tirés uniformément sur la voirie

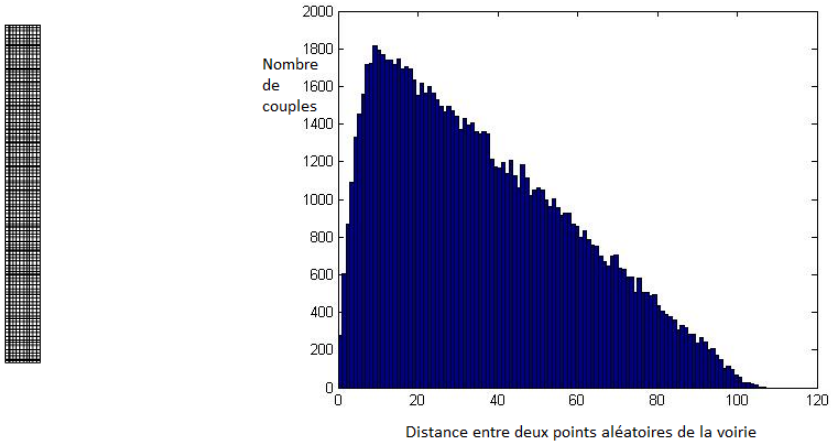

FIGURE 6.4 – Tessellation rectangulaire de taille 100x10 d'un Manhattan uniforme et histogramme joint des distances entre deux points tirés uniformément sur la voirie

6.2 Distribution de distance dans la cellule typique

Nous avons ensuite réalisé des simulations plus poussées avec les cellules typiques et les voiries obtenues par l'algorithme de génération de cellules typiques présenté en section 5.1. Nous avons procédé au tirage de 5000 cellules typiques pour une même valeur de κ. Ensuite, pour chaque cellule, 5000 points sont choisis au hasard uniformément le long de la voirie et les distances entre le centroïde et les points sont calculés grâce à l'algorithme de Dijkstra[8].

Enfin, j'ai comparé les distances obtenues par simulation avec des lois de probabilité classiques (normale, log-normale, Gumbel, Weibull). Ce procédé a été répété pour quelques valeurs de κ (50,100,1000) et dans tous les cas, la distribution de Weibull correspondait le mieux à la distribution empirique(cf. figures 6.5 et 6.6).

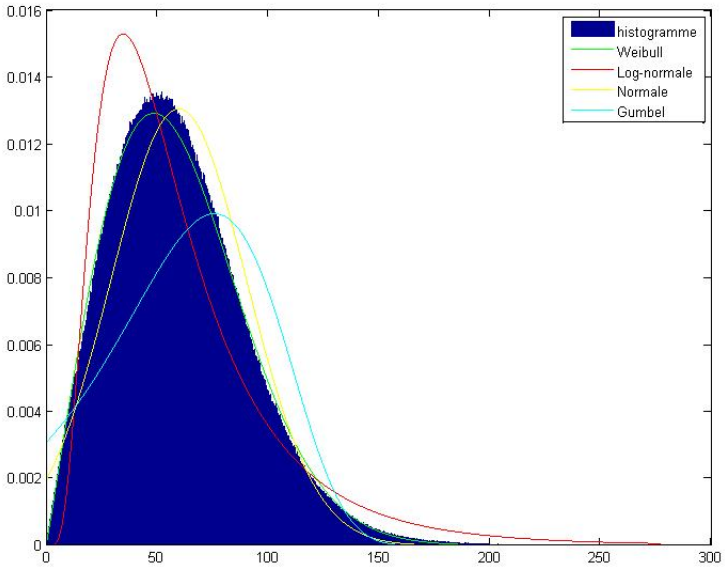

FIGURE 6.5 – Comparaison de la distribution simulée avec les distributions classiques

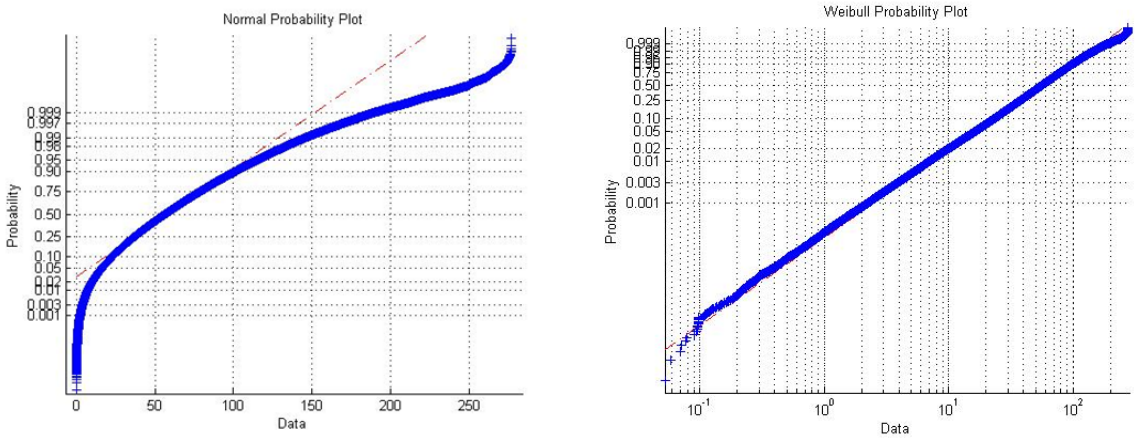

FIGURE 6.6 – Comparaison des quantiles des données simulées aux quantiles d'une loi Normale puis aux quantiles d'une loi Weibull.

La distribution de Weibull possède un **paramètre d'échelle** γ et un **paramètre de forme k**. Le paramètre d'échelle correspond approximativement à la moyenne de la distribution tandis que le paramètre de forme traduit plutôt la dispersion de la distribution. Sa fonction de densité est la suivante :

$$f(x, \gamma, k) = \frac{k}{\gamma} \left(\frac{x}{\gamma} \right)^{k-1} e^{-(x/\gamma)^k} \tag{6.1}$$

Son espérance μ et sa variance σ^2 sont données par :

$$\mu = \gamma \Gamma(1 + \frac{1}{k})$$

$$\sigma^2 = \gamma^2 \Gamma(1 + \frac{2}{k})$$

Pour inférer les paramètres γ et k des distributions obtenues par simulation, on utilisera la méthode du maximum de vraisemblance (directement implémentée dans Matlab).

On s'est tout d'abord penché sur le pramètre d'échelle γ. Pour l'évaluer, on a simulé 2000 cellules typiques de surfaces variables. Pour chacune de ces cellules, 5000 points sont tirés uniformément le long de la voirie, puis les distances entre ces points et le centroïde sont calculées de la même façon que précédemment. Cela donne alors une distribution

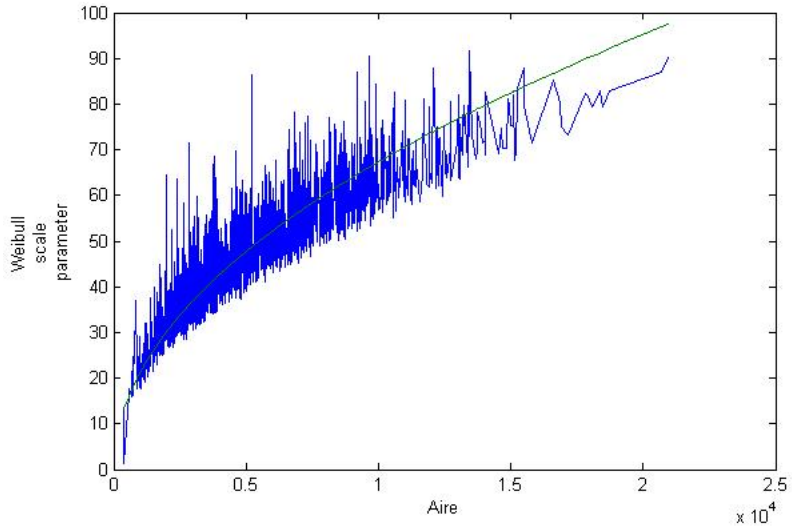

FIGURE 6.7 – Evolution du paramètre d'échelle γ de la loi de Weibull en fonction de l'aire de la cellule

de distance pour chaque cellule. L'identification numérique du paramètre d'échelle pour chaque cellule permet de visualiser les corrélations entre différentes caractéristiques de la cellule et ce paramètre. Le paramètre d'échelle est globalement proportionnel à la racine de l'aire de la cellule (cf. figure 6.7).

Ainsi pour estimer le paramètre d'échelle qui sera utilisé par la suite, j'ai simulé 1000 cellules typiques dont l'aire était maintenue constante à une valeur de référence (100) par homothétie pour des valeurs de κ variant de 10 à 1000 et j'ai retenu la valeur moyenne de γ sur ces simulations. Ainsi pour une cellule typique de Manhattan d'aire A, on a :

$$\gamma = 0.6746\sqrt{A} \tag{6.2}$$

Pour le paramètre de forme k, il n'y a malheureusement pas d'aussi bonne corrélation avec une caractéristique de la cellule. On a donc simulé 5000 cellules typiques ainsi que les distributions de distance correspondantes. On observe que pour des valeurs de κ supérieures à 10, la moyenne de k reste la même au cours des simulations et vaut environ 2.25. Pour des valeurs de κ inférieures à 10, on observe que k décroit légèrement avec κ et tend vers

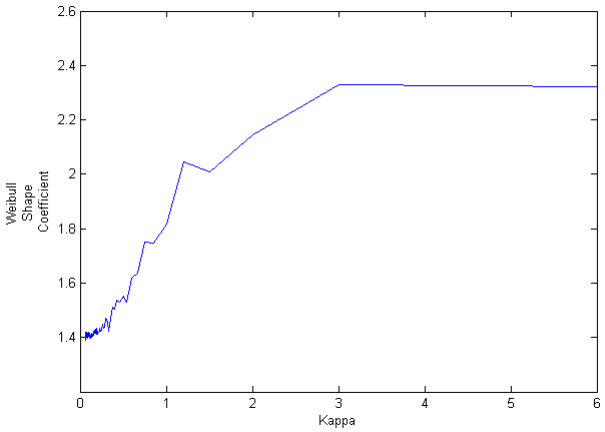

FIGURE 6.8 – Evolution du paramètre de forme k de la loi de Weibull pour des petites valeurs de κ

1.4 (figure 6.8).Finalement, on retient :

$$k = \mathbf{1}_{\kappa \geq 3} 2.25 + \mathbf{1}_{\kappa < 3}(1.4 + 0.3\kappa) \tag{6.3}$$

6.3 Validation par les données réelles avec OsmMiner

Après avoir ajouté le modèle de Manhattan au logiciel OsmMiner (et corrigé quelques bugs lors de l'implémentation), j'ai pu vérifier la validité de mes simulations à partir de la voirie réelle de la ville de New York, et de Manhattan en particulier. Après la segmentation réalisée par le logiciel, on récupère des quartiers homogènes de voirie de type Manhattan. Pour obtenir des cellules types, un nouveau tirage de points est réalisé grâce à un processus ponctuel de Poisson d'intensité t. On choisit t=$\frac{n}{\text{longueur de voirie}}$ de telle sorte à avoir environ n centroïdes dans chaque zone segmentée. On a réalisé ces tests avec plusieurs valeurs de n (des exemples avec n=15 sont proposés en figure 6.9).

Ensuite on calcule les régions de Voronoï associées à chaque centroïde. Dans chaque région de Voronoï, 5000 points sont tirés uniformément sur la voirie et la distance les

séparant du centroïde est calculée. Finalement, on réunit toutes les distances obtenues dans une seule distribution que l'on compare à la distribution théorique établie.

Les résultats sont satisfaisants même lorsque la voirie ne ressemble pas parfaitement à un modèle de Manhattan(fig 6.9). Les différences avec la distribution théorique viennent du fait qu'en pratique, les quartiers sont rarement aussi parfaits et homogènes. Souvent, des facteurs naturels (rivières, colline,...) obligent les urbanistes à courber les rues pour éviter des obstacles. Il existe aussi en général des voies rapides en diagonale ce qui a tendance à "réduire" les distances par rapport à un modèle de Manhattan parfait.

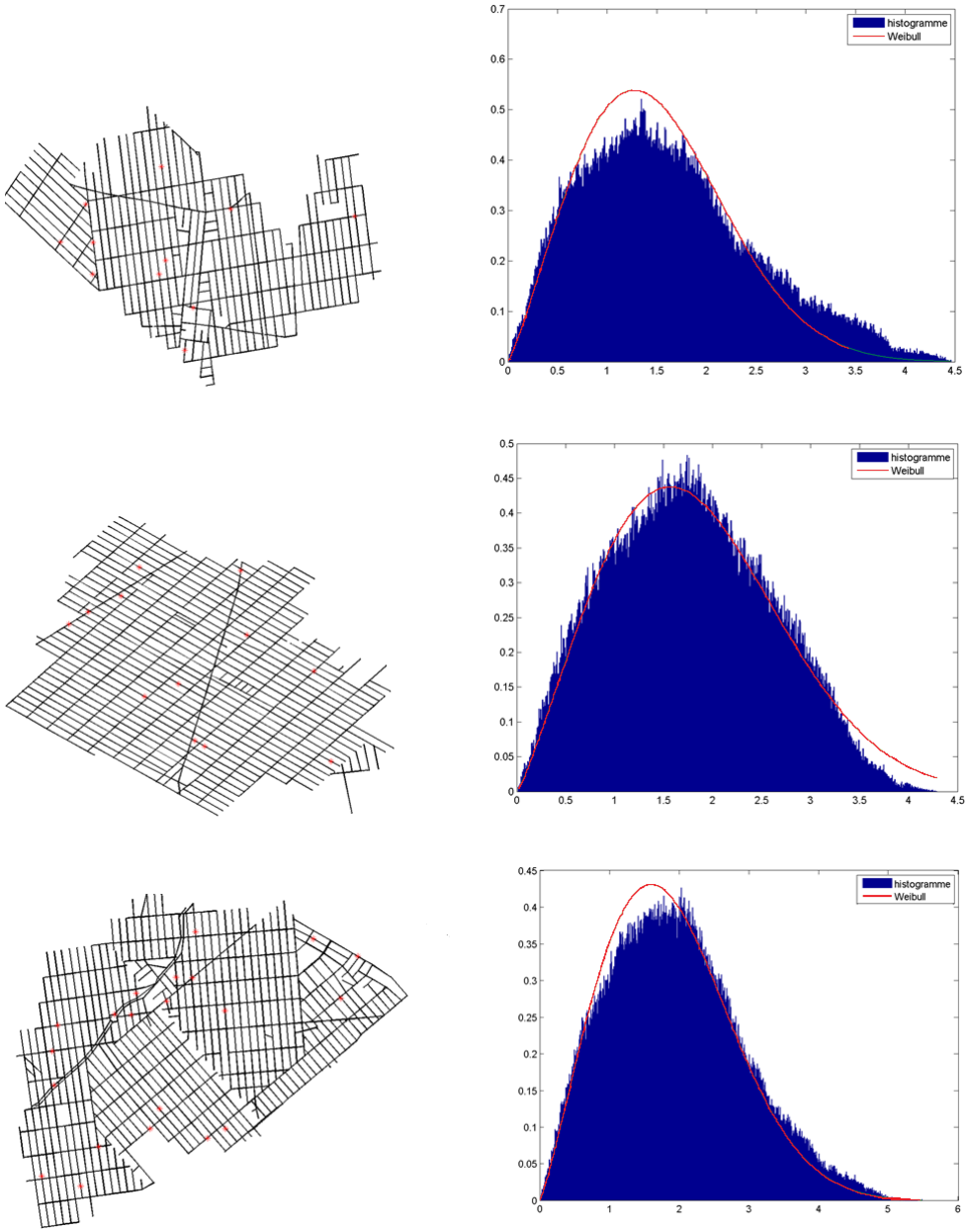

FIGURE 6.9 – Quartiers de New york retenus pour la validation. Les centroïdes générés sont en rouge. Les distributions de distance obtenues sont représentées par l'histogramme en bleu. La courbe rouge représente la distribution théorique.

7 Conclusion

Ce stage a été très enrichissant pour moi car il m'a permis de découvrir dans le détail le monde de la recherche en entreprise, son mode de fonctionnement et ses contraintes. J'ai aussi eu la chance de me familiariser avec l'univers des télécommunications, riches en nouvelles perspectives et problématiques. Ce domaine, au croisement de multiples technologies met à l'épreuve des ingénieurs de parcours très différents.

J'ai ainsi intégré une équipe travaillant sur des projets à long terme qui exploitent des connaissances poussées de géométrie stochastique dans le but de proposer des modèles représentatifs de la voirie d'un territoire dont certaines applications ont notamment pour objet d'estimer des coûts d'installation ainsi que l'éligibilité de clients potentiels. Mon approche, basée sur des méthodes de Monte-Carlo a permis de donner une distribution paramétrique des distances en suivant la voirie dans un quartier homogène de type Manhattan. Pour valider ce modèle, la distribution a été comparée à la distribution réelle, observée dans les quartiers de New York présentant de très bons exemples de ce type de voirie.

Finalement, j'ai beaucoup apprécié l'utilisation des probabilités et de la simulation dans le domaine de la géométrie, domaine dont je ne soupçonnais pas l'étendue, et cela m'a conforté dans mon choix de continuer à étudier les mathématiques de l'aléatoire.

8 Bibliographie & Annexes

8.1 sources

1. Courtat T., Promenade dans les cartes de villes - phénoménologie mathématiques et physique de la ville - une approche géométrique (2012).
2. C. Gloaguen, F. Fleischer, H. Schmidt and V. Schmidt, Fitting of stochastic telecommunication network models via distance measures and Monte-Carlo tests., Telecommun. Syst. 31 n4 pp.353-377, 2006
3. R. Maier and V. Schmidt, Stationary Iterated Tessellations, Adv. Appl. Prob. (SGSA), 35, pp. 337-353, 2003.
4. Gloaguen, C., Fleischer, F., Schmidt, H., Schmidt, V. (2005). Simulation of typical Cox–Voronoi cells with a special regard to implementation tests. Mathematical Methods of Operations Research, 62(3), 357-373.
5. Nagel, W., Weiss, V. (2008). Mean values for homogeneous STIT tessellations in 3D. Image Analysis and Stereology, 27, 29-38.
6. Chiu, S. N., Stoyan, D., Kendall, W. S., Mecke, J. (2013). Stochastic geometry and its applications. John Wiley Sons.
7. Gloaguen, C., Coupé, P., Maier, R., Schmidt, V. (2002, June). Stochastic modelling of urban access networks. In Proc. 10th Internat. Telecommun. Network Strategy Planning Symp (pp. 99-104).
8. http ://www.mathworks.com/matlabcentral/fileexchange/36140-dijkstra-algorithm.

8.2 annexes

8.2.1 Calcul de la moyenne de la distance entre deux points dans une mosaïque simple

La mosaïque est supposée rectangulaire de longueur L et de hauteur H. On va se placer dans un cas ou κ est suffisamment élevé pour que la distance en suivant la voirie entre deux points tirés uniformément sur la voirie A(x1,y1) et B(x2,y2) puisse être correctement approximée par la distance induite par la norme 1.
on utilisera le fait suivant : si (u1) et (u2) sont deux variables uniformes sur [0,1], alors $\mathbb{E}(|u1 - u2|) = 1/3$.

$$\mathbb{E}(d^{voirie}(A,B)) = \int_{x1=0}^{L}\int_{x2=0}^{L}\int_{y1=0}^{H}\int_{y2=0}^{H} |x1-x2| + |y1-y2| \frac{1}{L^2H^2} d_{x1}d_{x2}d_{y1}d_{y2} \quad (8.1)$$

$$\mathbb{E}(d^{voirie}(A,B)) = \int_{x1=0}^{L}\int_{x2=0}^{L} |x1-x2| \frac{1}{L^2} d_{x1}d_{x2} + \int_{y1=0}^{H}\int_{y2=0}^{H} |y1-y2| \frac{1}{H^2} d_{y1}d_{y2}$$

$$\mathbb{E}(d^{voirie}(A,B)) = \frac{L^3}{3L^2} + \frac{H^3}{3H^2}$$

$$\mathbb{E}(d^{voirie}(A,B)) = \frac{L+H}{3}$$

8.2.2 Coefficient de conversion d'un système à l'autre

Lorsqu'on veut étudier la distribution des noeuds de plus haut niveau, on réalise un processus ponctuel de Poisson sur toute la ville. On ne peut donc plus considérer la ville comme plusieurs quartiers homogènes mais on doit trouver le modèle (PVT,PLT,Manhattan,etc...) qui corresponde le mieux à l'ensemble de la ville. On réalise alors une sorte de moyenne pondérée des modèles de chaque quartier en tenant compte de la superficie de celui-ci). Une fois le modèle final retenu, il faut être capable de déterminer le ou les paramètre(s) correspondant(s) de chaque quartier, si possible sans avoir à refaire les tests d'adéquation et les calculs des vecteurs d'intensité. Par exemple on veut savoir

Conversion d'un Manhattan vers un autre modèle

La conversion d'un Manhattan vers un PVT, un PDT ou un PLT est simple puisqu'on a un seul paramètre à trouver. En notant Λ_{Man}= (α ; 2 α ; α ; 1) le vecteur d'intensité

du modèle Manhattan avec $\alpha \in [0; 0.25]$ connu, t_{Man} le facteur d'intensité connu et Λ_i le vecteur d'intensité du modèle retenu, connu lui aussi, on cherche :

$$\min_{t\in\mathbb{R}} |t_{Man}\Lambda_{Man} - t\Lambda_i|^2$$

Ainsi t est tout simplement le produit scalaire de $t_{Man}\Lambda_{Man}$ et $\frac{\Lambda_i}{\|\Lambda_i\|}$.

•$\Lambda_i = \Lambda_{PVT} = (0.5; 0.75; 0.25; 1)$

$$t = t_{Man}\frac{\frac{9\alpha}{2} + 2}{\frac{7}{4} + 2} \tag{8.2}$$

Pour un Manhattan régulier (i.e. α=0.25), t=0.8333t_{Man}

•$\Lambda_i = \Lambda_{PDT} = (\frac{9\pi^2}{1024}; \frac{27\pi^2}{1024}; \frac{18\pi^2}{1024}; 1)$

$$t = t_{Man}\frac{\frac{162\pi^2}{1024} + 2}{\frac{2268\pi^4}{1024^2} + 2} \tag{8.3}$$

Pour un Manhattan régulier (i.e. α=0.25), t=1.0813t_{Man}

•$\Lambda_i = \Lambda_{PLT} = (\frac{1}{\pi}; \frac{2}{\pi}; \frac{1}{\pi}; 1)$

$$t = t_{Man}\frac{\frac{12\alpha}{\pi} + 2}{\frac{12}{\pi^2} + 2} \tag{8.4}$$

Pour un Manhattan régulier (i.e. α=0.25), t=0.9189t_{Man}

Conversion d'un autre modèle vers Manhattan

La conversion d'une tessellation PVT,PDT,PLT vers du Manhattan est plus compliquée car il y a deux paramètres à trouver, t_{Man} et α. Avec $t_i = t_{PVT}, t_{PDT}, t_{PLT}$ et $\Lambda_i = \Lambda_{PVT}, \Lambda_{PDT}, \Lambda_{PLT}$ selon le modèle retenu, on cherche :

$$\min_{t_{Man}\in\mathbb{R},\alpha\in[0;0.25]} |t_{Man}\Lambda_{Man} - t_i\Lambda_i|^2$$

Heureusement, Matlab permet de trouver ce minimum par une méthode de descente de gradient.

•$\Lambda_i = \Lambda_{PVT} = (0.5; 0.75; 0.25; 1)$

$$t_{Man} = 1.1364 t_i$$
$$\alpha = 0.25$$

Cela correspond à un Manhattan régulier avec L1=L2

•$\Lambda_i = \Lambda_{PDT} = (\frac{9\pi^2}{1024}; \frac{27\pi^2}{1024}; \frac{18\pi^2}{1024}; 1)$

$$t_{Man} = 0.8692 t_i$$
$$\alpha = 0.1301 (i.e. L1 = 5.5 L2)$$

Cela correspond à $L1 \approx 5.5 L2$

•$\Lambda_i = \Lambda_{PLT} = (\frac{1}{\pi}; \frac{2}{\pi}; \frac{1}{\pi}; 1)$

$$t_{Man} = 1.0745 t_i$$
$$\alpha = 0.25 (i.e. L1 = L2)$$

Cela correspond à un Manhattan régulier avec L1=L2

8.2.3 génération de la cellule typique de Manhattan

```
function [ G,hor,ver,G2,L ] = MGT(mu1,mu2,lambda,p)
%crée une cellule typique sous forme d'un graphe, à partir d'une mosaïque
%de rues de type manhattan, les écarts entre les rues suivent une loi
%gaussienne
inter1=zeros(1,2);
inter2=zeros(1,2);
sigma1=mu1/10;%variance fixée arbitrairement
sigma2=mu2/10;
%stockage des coordonnées
X=zeros(5,1);
Y=zeros(5,1);
L=0;%longueur totale de rue
u0=floor(4*rand)*pi/2;%choix de l'orientation de la droite d'origine
v0=exprnd(lambda,1,2);%tirage des deux centroides sur la droite d'origine
v1=exprnd(lambda/2,1,2);%tirage d'un centroide par droite supplémentaire
```

```
X(2)=v0(1)*cos(u0);
Y(2)=v0(1)*sin(u0);
X(3)=v0(2)*cos(pi+u0);
Y(3)=v0(2)*sin(pi+u0);

%construction des deux premières rues, perpendiculaires à la droite
%d'origine
if abs(mod(u0,pi)) >= 0.01
    r1=max(sigma2/10,normrnd(mu2,sigma2));
    r=r1*rand; %distance avec la droite d'origine
    r=[r r1-r 0 0 ];
    inter1(2)=0;
    inter2(2)=0;
    dir=2*floor(2*rand)-1;
    inter1(1)=r(1);
    inter2(1)=-r(2);
    X(4)=inter1(1);
    X(5)=inter2(1);
    Y(4)=dir*v1(1)+inter1(2);
    Y(5)=-dir*v1(2)+inter2(2);
    hor=[r(1) -r(2)];% stock les ordonnées des droites horizontales
    ver=0 ;% stock les abscisses des droites verticales
else
    r2=max(normrnd(mu1,sigma1),sigma1/10);
    r=r2*rand;
    r=[r r2-r 0 0];
    inter1(1)=0;
    inter2(1)=0;
    dir=2*floor(2*rand)-1;
    inter1(2)=r(1);
    inter2(2)=-r(2);
    X(4)=inter1(1)+dir*v1(1);
    X(5)=inter1(1)-dir*v1(2);
    Y(4)=inter1(2);
    Y(5)=inter2(2);
    hor=0;
    ver=[r(1) -r(2)];
end
```

```
%calcul de la cellule de voronoi d'origine, par construction elle ne peut
%être infinie
V=[X,Y];
[vertic,cell]=voronoin(V);
c0=cell{1};
convhull(vertic(c0,:));
ddd=vertic(c0(convhull(vertic(c0,:))),:);
R=max(sqrt(ddd(1,:).*ddd(1,:)+ddd(2,:).*ddd(2,:)));
%sert de limite à la génération de droites supplémentaires
k=6;

%boucle de générations des droites supplémentaires
while min(r) < R% on génère les rues tant que la cellule originale
peut encore être modifiée
    v=exprnd(lambda/2,1,4);
    dir=2*floor(2*rand(1,2))-1;
    if abs(mod(u0,pi)) >= 0.01
        r= r+ [normrnd(mu2,sigma2,1,2),normrnd(mu1,sigma1,1,2)];
        Y=[Y;r(1);-r(2)];
        if rand >0.5
            x1=X(k-2);
            x2=X(k-1);
        else
            x2=X(k-2);
            x1=X(k-1);
        end
        if rand >0.5
            y1=Y(k-4);
            y2=Y(k-3);
        else
            y2=Y(k-4);
            y1=Y(k-3);
        end
        X=[X; dir(1)*v(1)+x1; -dir(1)*v(2)+x2];
        hor=[hor,r(1),-r(2)];
        Y=[Y;y1+dir(2)*v(3);y2-dir(2)*v(4)];
        X=[X;r(3);-r(4)];
        ver=[ver,r(3),-r(4)];
```

```
    else
        if rand >0.5
            x1=X(k-4);
            x2=X(k-3);
        else
            x2=X(k-4);
            x1=X(k-3);
        end
        if rand >0.5
            y1=Y(k-2);
            y2=Y(k-1);
        else
            y2=Y(k-2);
            y1=Y(k-1);
        end
        r= r+ [normrnd(mu1,sigma1,1,2),normrnd(mu2,sigma2,1,2)];
        X=[X;r(1);-r(2)];
        Y=[Y; dir(1)*v(1)+y1; -dir(1)*v(2)+y2];
        ver=[ver,r(1),-r(2)];
        X=[X;x1+dir(2)*v(3);x2-dir(2)*v(4)];
        Y=[Y;r(3);-r(4)];
        hor=[hor,r(3),-r(4)];
    end
    k=k+4;
    V=[X,Y];
    [vertic,cell]=voronoin(V);
    c0=cell{1};
    convhull(vertic(c0,:));
    ddd=vertic(c0(convhull(vertic(c0,:))),:);
    R=2*max(sqrt(ddd(:,1).*ddd(:,1)+ddd(:,2).*ddd(:,2)));
end

%stockage de l'enveloppe convexe de la cellule typique dans un graphe
V=[X,Y];
[vertic,cell]=voronoin(V);
c0=cell{1};
conv=convhull(vertic(c0,:));
```

```
ddd=vertic(c0(convhull(vertic(c0,:))),:);
G.verticles = ddd;
M=zeros(length(ddd)-1,2);
for i = 1:length(ddd)-1
    M(i,1)=conv(i);
    M(i,2)=conv(i+1);
end
G.edges = M;
%affichage des cellules de voronoi
% figure;
% hold on
% axis equal
% voronoi(X,Y);

%création du graphe de la voirie contenue dans la cellule d'origine
hor=sort(hor);
ver=sort(ver);
N=size(ver,2);
M=size(hor,2);
sommets =zeros(N*M,2);
iter=1;

% on stocke d'abord toutes les intersections crées par l'ensemble des
% droites
for i=1:N
    for j = 1:M
        sommets(iter,1)=ver(i);
        sommets(iter,2)=hor(j);
        iter=iter+1;
    end
end
in = inhull(sommets,G.verticles);
% on vérifie quels sommets sont dans la cellule d'origine
sommets_check=ones(size(sommets,1),1);
if abs(mod(u0,pi)) >= 0.01
    j2=find(ver==0);
    j1=find(hor > 0, 1 );
    sommets_check((j2-1)*M+j1)=0;
```

```
%   sommets((j2-1)*M+j1,:)
else
    j1=find(hor==0);
    j2=find(ver > 0 , 1);
    sommets_check((j2-1)*M+j1)=0;
end
if sum(in)==0
    if abs(mod(u0,pi)) >= 0.01
        inter1=dist_arrete2([0,0],sommets( (j2-1)*M+j1 , :),G.verticles);
        inter2=dist_arrete2([0,0],sommets( (j2-1)*M+j1 - 1, :),G.verticles);
        sommets2=[0,0;inter1;inter2];
        arretes2=[1,2;1,3];
        L=sqrt(inter1*inter1')+sqrt(inter2*inter2');
    else
        inter1=dist_arrete2([0,0],sommets( (j2-1)*M+j1 , :),G.verticles);
        inter2=dist_arrete2([0,0],sommets( (j2-2)*M+j1 , :),G.verticles);
        sommets2=[0,0;inter1;inter2];
        arretes2=[1,2;1,3];
        L=sqrt(inter1*inter1')+sqrt(inter2*inter2');
    end
else
arretes2=zeros(1,2);
sommets2=sommets(in==1,:);
iter=1;

for i=1:N-1
    for j =1:M-1
        %pour chaque sommet on tente de crée l'arrête vers le haut et vers
        %la droite
        if  (in((i-1)*(M)+j)==1 ||in((i-1)*(M)+1+j)==1)
            if  rand <p || sommets_check((i-1)*M +j+1)*sommets_check((i-1)*M +j)==0
            %avec probabilité p on crée l'arrête
                if in( (i-1) * M + j ) == 0
                    inter1=dist_arrete2(sommets( (i-1) * M + j + 1, :),
                    sommets( (i-1) * M + j , :),G.verticles);
                    sommets2=[sommets2 ; inter1 ] ;
                    index1=size(sommets2,1);
                    arretes2(iter,1)=index1;
```

```
        else
             index1=find(sommets2==sommets((i-1) * M + j ));
             for k1=1:length(index1)
                 if sommets2(index1(k1),2) == sommets((i-1) * M + j ,2)
                     break;
                 end
             end
             arretes2(iter,1)=index1(k1);
         end
         if in((i-1) * M + j + 1) == 0
            % in( (i-1) * M + j + 1 ) = 2;
             inter1=dist_arrete2(sommets( (i-1) * M + j, :),
             sommets( (i-1) * M + j + 1, :),G.verticles);
             sommets2=[sommets2 ; inter1 ] ;
             index2=size(sommets2,1);
             arretes2(iter,2)=index2;
         else
             index2=find(sommets2==sommets((i-1) * M + j + 1));
             for k2=1:length(index2)
                 if sommets2(index2(k2),2) == sommets((i-1) * M + j + 1 , 2 )
                     break;
                 end
             end
             arretes2(iter,2)=index2(k2);
         end
    L=L+sqrt((sommets2(arretes2(iter,1),:)-sommets2(arretes2(iter,2),:))
    *(sommets2(arretes2(iter,1),:)-sommets2(arretes2(iter,2),:))') ;
    iter=iter+1;
    else
        sommets_check((i-1) * M +j +1) =0;
        sommets_check((i-1) * M +j )= 0;
    end
end
if (in((i-1)* M + j)==1 || in( i * M +j)==1)
    if rand <p || sommets_check(i * M +j) * sommets_check((i-1) * M +j )== 0
        if in( (i-1) * M + j ) == 0
            inter1=dist_arrete2(sommets( i * M + j, :),
            sommets( (i-1) * M + j , :),G.verticles);
```

```
                    sommets2=[sommets2 ; inter1 ] ;
                    index3=size(sommets2,1);
                    arretes2(iter,1)=index3;
                else
                    index3=find(sommets2==sommets((i-1)*M+j));
                    for k3=1:length(index3)
                        if sommets2(index3(k3),2) == sommets((i-1)*M+j,2)
                            break;
                        end
                    end
                    arretes2(iter,1)=index3(k3);
                end
                if in(i * M + j) == 0
                   % in( i * M + j ) = 2;
                    inter1=dist_arrete2(sommets( (i-1) * M + j, :),
                    sommets( i * M + j , :),G.verticles);
                    sommets2=[sommets2 ; inter1 ] ;
                    index4=size(sommets2,1);
                    arretes2(iter,2)=index4;
                else
                    index4=find(sommets2==sommets(i* M + j));
                    for k4=1:length(index4)
                        if sommets2(index4(k4),2) == sommets(i* M + j,2)
                            break;
                        end
                    end
                    arretes2(iter,2)=index4(k4);
                end
                L=L+sqrt((sommets2(arretes2(iter,1),:)-sommets2(arretes2(iter,2),:))
                *(sommets2(arretes2(iter,1),:)-sommets2(arretes2(iter,2),:))');
                iter=iter+1;
            else
                sommets_check(i * M +j)=0;
                sommets_check((i-1) * M +j )=0;
            end
        end
    end
end
```

```
end
G2.verticles=sommets2;
G2.edges=arretes2;
 end
```

8.2.4 Code du diagramme de Voronoï pour la distance de Manhattan

```
function [ G,L ] = vor_man(X,Y,bord)
%cellule 0 de voronoi pour la distance de Manhattan (norme 1)
if X(1)==0 && Y(1)==0
    X=X(2:end);Y=Y(2:end);
end
V=abs(X)+abs(Y);
[~,o]=sort(V);
X=X(o);Y=Y(o);
grand=100*max(abs(X)+abs(Y));
if nargin==2
    bord=[0.1*grand,0.1*grand;-0.1*grand,0.1*grand;
    -0.1*grand,-0.1*grand;0.1*grand,-0.1*grand];
end
C=[0,0];
n=length(X);
sommets=zeros(4*n,2);
for i=1:length(X)
    x=abs(X(i));
    y=abs(Y(i));
    e=sign(X(i));
    f=sign(Y(i));
    if x < y
        sommets((i-1)*4+1,:)=[e*(grand+x),f*(y-x)/2];
        sommets((i-1)*4+2,:)=[X(i),f*(y-x)/2];
        sommets((i-1)*4+3,:)=[0,f*(y+x)/2];
        sommets((i-1)*4+4,:)=[-e*grand,f*(y+x)/2];
    else
        sommets((i-1)*4+1,:)=[e*(x-y)/2,f*(grand+y)];
```

```
    sommets((i-1)*4+2,:)=[e*(x-y)/2,Y(i)];
    sommets((i-1)*4+3,:)=[e*(y+x)/2,0];
    sommets((i-1)*4+4,:)=[e*(y+x)/2,-f*grand];
end
    out=lineSegmentIntersect([bord(1:end,:),[bord(2:end,:);bord(1,:)]],
    [sommets((i-1)*4+1:(i-1)*4+3,:),sommets((i-1)*4+2:(i-1)*4+4,:)]);
    [j1,j2]=find(out.intAdjacencyMatrix==1);
    j=find(out.intAdjacencyMatrix==1) ;
    if not(isempty(j))
        xx=out.intMatrixX(j);
        yy=out.intMatrixY(j);
        if j1(1) < j1(2)
            newe=[xx(1),yy(1);sommets((i-1)*4+j2(1)+1:(i-1)*4+j2(2),:);
            xx(2),yy(2)];
        elseif j1(1) > j1(2)
            newe=[xx(2),yy(2);sommets((i-1)*4+j2(2):-1:(i-1)*4+j2(1)+1,:);
            xx(1),yy(1)];
        end
            j1=sort(j1);
        if j1(1) > 1 && j1(2) < length(bord)
            conv1 = [bord(1:j1(1),:);newe;bord(j1(2)+1:end,:)];
            conv2 = [bord(j1(1)+1:j1(2),:);newe(end:-1:1,:)];
        elseif j1(1) > 1 && j1(2) == length(bord)
            conv1 = [bord(1:j1(1),:);newe];
            conv2 = [bord(j1(1)+1:j1(2),:);newe(end:-1:1,:)];
        elseif j1(1) == 1 && j1(2) == length(bord)
            conv1 = [bord(1,:);newe];
            conv2 = [bord(2:j1(2),:);newe(end:-1:1,:)];
        elseif j1(1) == 1 && j1(2) < length(bord)
            conv1 = [bord(1,:);newe;bord(j1(2)+1:end,:)];
            conv2 = [bord(2:j1(2),:);newe(end:-1:1,:)];
        end
        %plot(conv1(1:end,1),conv1(1:end,2),'c');
        if inhull(C,conv1)
            bord=conv1;
        elseif inhull(C,conv2)
            bord=conv2;
        else
```

```
                figure;plot(0,0,'*r');hold on;
                plot(bord(1:end,1),bord(1:end,2),'r');
            end
        %plot(sommets((i-1)*4+1:(i-1)*4+4,1),sommets((i-1)*4+1:(i-1)*4+4,2));
         %plot(sommets(1:end,1),sommets(1:end,2));
         %plot(new(1:end,1),new(1:end,2),'g');
         end
end
G.verticles=[bord(1:end,:);bord(1,:)];
G.edges=[1,length(G.verticles);[(1:length(G.verticles)-1)',
(2:length(G.verticles))']];
L=max(abs(G.verticles(:,1))+abs(G.verticles(:,2)));
end
```