UCLouvain

École polytechnique de Louvain

Amélioration des zones d'adresses chez Bpost

Auteur: **Simon Teugels**
Promoteur: **Siegfried Nijssen**
Lecteurs: **Gaël Aglin, Guy Henrard**
Année académique 2018–2019
Master [120] : ingénieur civil en informatique

Table des matières

Remerciements

Je voudrais tout d'abord remercier mon promoteur, le professeur Siegfried Nijssen, pour ses conseils, son écoute et le temps qu'il m'a consacré depuis un peu plus d'un an. Je voudrais également remercier Guy Henrard, qui a été notre contact privilégié chez Bpost, pour le temps qu'il nous a accordé et les conseils judicieux qu'il nous a fournis notamment par sa connaissance du terrain.

Je voudrais enfin remercier Françoise, Benoît, Coline, Marie et Michel, pour leurs relectures et conseils précieux lors de la rédaction de ce travail.

Résumé

Dans un contexte de méfiance grandissante envers les grandes entreprises et leur gestion des données, Bpost souhaite allier respect de la vie privée et performance afin de fournir à ses clients un service de distribution de flyers de qualité.

Dans ce mémoire, il nous était demandé de trouver une solution pour dénombrer les adresses auxquelles distribuer des flyers. Afin de rendre cela possible, il était nécessaire à Bpost de proposer une méthode relativement simple pour sélectionner la zone géographique dans laquelle les flyers seront distribués. Le principal critère à respecter était de ne pas permettre aux utilisateurs de découvrir une zone dans laquelle se trouveraient moins de 80 adresses. Cette contrainte se fonde sur la lecture et l'interprétation du RGPD faite par Bpost.

Nous avons également été tenus de respecter d'autres contraintes d'efficacité concernant les centres de distribution et les segments d'activité auxquels appartiennent les adresses sélectionnées afin de faciliter les procédures liées à ce service.

Le plan a, dans un premier temps, été recouvert de polygones grâce à un diagramme de Voronoï en utilisant les adresses comme germes. Nous avons ensuite fusionné ces polygones jusqu'à atteindre le seuil de 80 adresses préalablement fixé, et ce, tout en respectant les critères imposés. L'algorithme que nous avons construit, codé en Python, s'inspire de celui de Casper et de sa structure pyramidale pour former plusieurs niveaux ainsi que de son fonctionnement bottom-up pour la sélection des adresses à proprement parler. L'utilisation de ces niveaux étant sujette à de potentielles attaques, nous avons également envisagé la possibilité de n'utiliser que le dernier niveau.

Les résultats que nous avons obtenus sont dans les deux cas satisfaisants. Bien entendu, la précision se détériore quelque peu lorsque nous n'utilisons que le dernier niveau. Ceci étant, nous pouvons tout de même envisager l'utilisation de cette version si le risque d'attaque est jugé trop important.

L'objectif de fournir une solution respectueuse de l'anonymat et aussi précise que possible à Bpost afin de dénombrer le nombre d'adresses se trouvant dans un périmètre donné autour d'un point est atteint.

Chapitre 1

Introduction

1.1 Contexte

Parmi les problèmes auxquels se trouve confronté l'informaticien d'aujourd'hui, la création et la maîtrise des grosses bases de données constituent assurément une préoccupation majeure. Celles-ci rassemblent une multitude de données de tout type et de diverse qualité. Concernant notamment l'ensemble des citoyens.

Depuis quelques années, le citoyen lambda découvre avec surprise et inquiétude qu'il se retrouve ainsi fiché dans un nombre toujours croissant de bases de données connues. Le plus souvent, ces bases de données sont cachées et il n'a pas conscience de la quantité de données que certains géants du monde numérique d'aujourd'hui peuvent récolter à son sujet.

Surfant sur cette inquiétude, des groupes de réflexion et de défense des individus se sont constitués afin de réclamer une meilleure protection des données des individus. En 2012, la Commission européenne propose de réviser le règlement [1] adopté en 1995 en matière de protection des données. En 2014, le Parlement européen adopte le *Règlement général pour la protection des données (RGPD)* [2] et le 25 mai 2018, le RGPD entre en vigueur.

Un petit peu plus d'un an après l'entrée en vigueur de ce Règlement général pour la protection des données, les défis pour l'appliquer sont plus grands que jamais. Pendant les deux années qui ont précédé sa mise en vigueur, les entreprises se sont efforcées de changer leurs pratiques afin d'atteindre les exigences du nouveau règlement. Certaines de ces pratiques ont donc été adaptées mais, n'offrent pas toujours de bonnes performances. En effet, ne pouvant plus utiliser certaines don-

1. Conseil de l'Europe et Parlement européen. (1995).
2. Conseil de l'Europe et Parlement européen. (2016).

nées à caractère personnel, les algorithmes ont dû être adaptés. En ce qui concerne les nouveaux projets, il n'est plus question de reprendre les mauvaises pratiques passées, il faut directement inscrire les nouvelles solutions dans le cadre du RGPD.

C'est dans ce contexte que se développent les nouveaux projets informatiques au sein des grandes entreprises. Pour tout nouveau projet, l'informaticien doit, lors de la création de bases de données, allier les contraintes de performance et les contraintes légales.

1.2 Objectif

Bpost, société anonyme belge de droit public, un des leaders en matière de distribution de courrier sur le territoire national n'échappe pas à la problématique de la sécurisation de ses bases de données.

Bpost propose un service de distribution de flyers publicitaires à ses clients. Dans le cadre de ce service, Bpost laisse au client le soin de choisir la zone dans laquelle les flyers seront distribués. Afin de faciliter le choix des adresses auxquelles les flyers seront distribués, Bpost souhaite fournir à ses clients la possibilité de sélectionner une zone circulaire définie par un point central et un rayon. Cette zone une fois définie permet de visualiser la surface et le nombre d'habitations qu'elle touchera. Dès lors si l'on connaît le nombre d'habitations dans la zone, il est possible d'en déduire le prix du service de distribution.

Tel est le contexte dans lequel s'inscrit ce mémoire. Notre but est donc de créer un algorithme permettant de dénombrer les adresses incluses dans un périmètre circulaire donné. Le dénombrement doit être le plus précis possible, mais ne doit en aucun cas permettre de délimiter une zone qui comporterait moins de 80 adresses.

1.3 Motivations

Le *Règlement général pour la protection de données*[3] incite à la plus grande prudence lorsque les données utilisées sont dites "à caractère personnel".

Les données à caractère personnel sont définies comme suit : "toute information se rapportant à une personne physique identifiée ou identifiable (ci-après dénommée "personne concernée") ; est réputée être une «personne physique identifiable» une

3. Conseil de l'Europe et Parlement européen. (2016).

personne physique qui peut être identifiée, directement ou indirectement, notamment par référence à un identifiant, tel qu'un nom, un numéro d'identification, des données de localisation, un identifiant en ligne, ou à un ou plusieurs éléments spécifiques propres à son identité physique, physiologique, génétique, psychique, économique, culturelle ou sociale ;" [4]

L'algorithme actuel de Bpost respecte le caractère privé des données, mais n'est pas très précis. Il se base essentiellement sur les zones NIS9. Les zones NIS correspondent à un découpage de la Belgique selon des critères socio-démographiques. À ce niveau (NIS9), la Belgique est divisée en 19782 zones. Ceci implique des zones contenant beaucoup d'adresses (jusqu'à 9371). L'algorithme retourne donc un nombre trop large par rapport à la requête du client. Cela s'explique par le fait que Bpost ne veut pas révéler les zones comportant moins de 80 adresses.

Bpost a procédé à une analyse juridique du RGPD. Cette réglementation manque de précision et laisse place à une grande part d'interprétation. Ceci est vrai notamment en ce qui concerne la problématique dont il est question dans ce travail. De cette analyse, il ressort qu'il devrait être impossible de délimiter, à l'aide de l'outil de Bpost, une zone rassemblant moins de 80 adresses.

Un autre problème réside dans l'absence de rapport entre les zones NIS et les tournées des facteurs. Il arrive donc parfois qu'une adresse soit la seule sélectionnée dans la tournée d'un facteur. Ceci complique le travail de ce dernier. Bpost cherche donc à remédier à cette incohérence dans le processus de distribution.

Bpost a donc décidé de ne pas divulguer d'informations à propos de ses clients et ce même lorsque ces informations peuvent être recueillies par d'autres moyens tels que par exemple Google Map. Bpost ne souhaite pas faciliter la tâche à des opérateurs désirant collecter des informations à caractère personnel telles que des adresses. Bpost a choisi d'adopter une position prudente en la matière.

1.4 Roadmap

Dans ce mémoire, vous trouverez tout d'abord un chapitre consacré à l'explication détaillée du problème et des contraintes qui nous étaient imposés ainsi que les souhaits formulés par Bpost.

Vous trouverez ensuite un bref descriptif de quelques algorithmes issus de la littérature : Nearest neighbor cloak, Hilbert cloak et Casper. Bien que nous n'ayons utilisé en pratique aucun de ces algorithmes, l'un d'entre eux (Casper) nous a largement inspiré pour créer l'algorithme que nous souhaitions.

4. Conseil de l'Europe et Parlement européen. (2016) p. 33.

Après une explication détaillée du problème et des algorithmes issus de la littérature, vous pourrez découvrir en détail les différentes phases de l'algorithme ainsi que les résultats qu'il a produit. Enfin, nous terminerons en évoquant les pistes d'améliorations possibles.

Chapitre 2

Spécifications de la problématique

2.1 But recherché

Bpost souhaite offrir un service de distribution de flyers publicitaires à ses clients. Pour ce faire, il est indispensable pour les clients de pouvoir choisir une zone dans laquelle les flyers seront distribués. Pour ce service, Bpost envisage notamment de proposer à ses clients de sélectionner une zone circulaire à partir d'une adresse et d'un rayon. Par exemple, une petite boulangerie pourrait envoyer un flyer à toutes les adresses autour d'elle dans un rayon de 3 km. Il est également nécessaire pour Bpost de pouvoir fournir le nombre d'adresses sélectionnées afin d'établir le prix de ce service.

Ceci pourrait être relativement simple si nous ne nous préoccupions pas du respect de la vie privée. En effet, il suffirait d'interroger la base de données afin d'en tirer le nombre d'adresses présentes dans un périmètre donné. Ceci aurait l'avantage d'identifier le nombre exact d'adresses dans le périmètre et donc de calculer correctement le prix pour le client.

Toutefois, pour respecter au mieux le RGPD, Bpost considère qu'il est indispensable de ne pas révéler les zones contenant moins de 80 adresses. En effet, ceci peut être considéré comme une donnée à caractère personnel et fait donc l'objet de la plus grande prudence de la part de l'entreprise.

Il est donc demandé de développer un algorithme permettant de dénombrer les adresses présentes dans le périmètre donné, et ce, sans révéler les endroits comprenant moins de 80 adresses.

2.2 Contraintes

Bpost souhaite répondre aux critères d'anonymat dans le but de respecter strictement la réglementation RGPD. Ceci débouche sur la principale contrainte qui consiste à imposer un minimum de 80 adresses à communiquer quelle que soit la taille de la zone sélectionnée. D'autres contraintes relatives à la qualité du service et à l'efficience des procédures viennent compléter la définition du problème. Ces contraintes sont présentées dans cette section.

2.2.1 Anonymat

Premièrement, toujours dans l'idée de ne pas divulguer d'information sensible, il ne doit pas être possible de trouver une zone contenant moins de 80 adresses, et cela, malgré la possibilité d'effectuer plusieurs requêtes.

Pour cela, il est important de ne jamais renvoyer un nombre en dessous de 80 adresses, mais également de ne pas pouvoir isoler grâce à plusieurs requêtes une zone dans laquelle se trouveraient moins de 80 adresses.

2.2.2 Précision

Deuxièmement, le dénombrement doit être le plus précis possible. L'un des atouts de Bpost face à ses concurrents réside dans la précision qu'il offre dans la sélection des zones qui seront couvertes par la campagne du client. Il n'est donc pas envisageable de créer des groupes d'adresses fortement supérieurs à 80, ce qui aurait pour conséquence d'augmenter très rapidement le nombre d'adresses concernées par une requête et donc introduirait de l'imprécision. Ceci serait très efficace en matière de respect de la vie privée, mais ne conviendrait pas vraiment aux clients de Bpost souhaitant un dénombrement précis.

2.2.3 Déterminisme

Troisièmement, le dénombrement ne doit pas être aléatoire. Cette contrainte va de pair avec la première. On ajoute ici l'idée d'être attaché à la réalité. En effet, si nous décidons de manière aléatoire d'augmenter le nombre d'adresses, cela ne représentera pas forcément la réalité du terrain. Or, si nous cherchons par exemple à étendre quelque peu la zone souhaitée afin d'atteindre un nombre d'adresses suffisant, nous obtiendrons un dénombrement bien plus proche de la réalité du terrain. De plus, il faut également garder à l'esprit que si un nombre d'adresses a

été donné au client, il faudra distribuer autant de flyers et donc avoir déterminé quelles étaient ces adresses.

De plus, avec un nombre aléatoire à chaque requête, nous nous exposons à un conflit avec la première contrainte. Effectivement, si les résultats sont issus d'une loi de probabilité, nous risquons de dévoiler une zone comportant moins de 80 adresses. Nous serions en contradiction avec le critère suivant.

2.2.4 Cohérence

Quatrièmement, le dénombrement pour un même rayon et un même point central doit toujours être le même. En effet, si un client souhaite refaire exactement la même campagne, il n'y a pas de raison qu'il obtienne un nombre d'adresses différent. Si ce n'était pas le cas, l'utilisateur pourrait devenir méfiant à l'égard du service.

2.2.5 Sélection des adresses

Enfin, Bpost a également souhaité ajouter quelques contraintes concernant la priorité de sélection des adresses. Ces contraintes sont plutôt des préférences. Elles nous guideront lors de l'élaboration de l'algorithme.

En effet, l'idée étant d'associer plusieurs adresses, nous devons décider quelles adresses nous allons regrouper. Les deux contraintes présentées ci-après nous serviront de guide lorsque nous devrons faire ce choix.

Continuité dans une tournée

Lors de sa tournée, un facteur devra distribuer les flyers des clients aux adresses sélectionnées. Si les adresses pouvaient être sélectionnées de manière arbitraire, il se pourrait qu'un facteur doive distribuer ces flyers à une maison sur deux, ou à trois endroits différents sur sa tournée. Ces situations sont relativement problématiques, car elles complexifient beaucoup la distribution. Elles allongent le temps nécessaire pour les préparer ainsi que le temps nécessaire pour les effectuer. Bpost souhaiterait donc concentrer les adresses sélectionnées par ses clients sur un nombre de tournées le plus limité possible.

Ceci permettrait d'avoir une distribution de flyers plus homogène. Un facteur pourrait par exemple devoir distribuer un type de flyers sur la première moitié de sa tournée et un autre type de flyers sur la seconde moitié. Ceci lui facilitera le travail, tant durant la préparation que durant la distribution.

Distinction des centres de distribution

Une autre difficulté se présente lorsque toutes les adresses sélectionnées ne se situent pas dans le même centre de distribution. En effet, ces centres étant relativement indépendants les uns des autres, la procédure qui s'applique est beaucoup plus complexe lorsqu'une requête concerne plusieurs centres de distribution.

Bpost souhaite donc limiter ces cas. Il doit notamment être possible de facilement désélectionner toutes les adresses ne se situant pas dans le centre de distribution comprenant la majorité des adresses. Nous nous attacherons donc à limiter le nombre de centres de distribution touchés par une requête.

2.3 Formalisation du problème

Le champ de mise en œuvre de notre problème est un plan composé de points aléatoirement répartis sur ce plan. Nous cherchons à y dénombrer le nombre de points se situant dans un cercle donné. Ce dénombrement doit répondre à quelques critères.

Tout d'abord, nous devons toujours renvoyer un nombre supérieur à une borne B. Ceci doit toujours être vrai, quels que soient le centre et le rayon du cercle donnés. Ceci reste d'application même lorsqu'un grand nombre de requêtes est autorisé. En effet, un utilisateur ne doit pas non plus pouvoir découvrir une telle zone à l'aide d'un ensemble de requêtes.

Ensuite, le dénombrement doit toujours retourner le même résultat pour un même cercle de centre C et de rayon R. Il n'est pas question de renvoyer un dénombrement généré aléatoirement et changeant à chaque requête.

Le dénombrement doit être le plus proche possible de la réalité. Il n'est donc pas question d'ajouter une valeur X supérieure à B à toutes les requêtes afin de s'assurer que le dénombrement est bien supérieure à B.

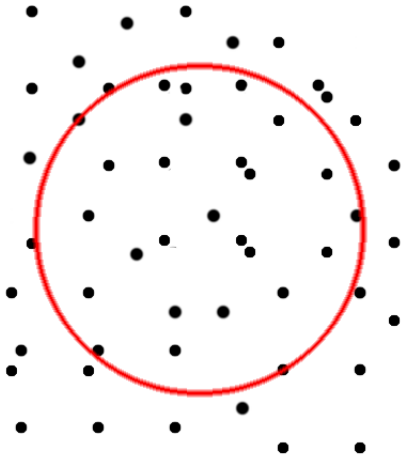

FIGURE 2.1 – représentation du problème

Afin de répondre à ce problème, nous avons décidé de créer des polygones comprenant chacun une adresse. Ceci est fait grâce à l'algorithme de Voronoï. Le nombre d'adresses figurant dans un polygone est appelé poids dans la suite du travail. À l'issue de cette première étape, le poids de chaque polygone est donc de 1.

Nous avons ensuite décidé de rassembler ces polygones en nous inspirant des travaux de Casper afin d'augmenter leur poids. Ceci a pour but d'arriver à un ensemble de polygones ayant un poids supérieur ou égal à 80.

Il est ensuite nécessaire de dénombrer les adresses présentes dans le cercle fourni par l'utilisateur. Pour cela, nous faisons la somme des poids de tous les polygones dont au moins une partie de la surface est recouverte par le cercle. Ceci permet dans une certaine mesure d'élargir le nombre d'adresses que nous dénombrerons pour cette requête. Effectivement, certaines adresses ne se trouvant pas dans le cercle seront comptabilisées si le polygone qui leur est associé possède au moins une partie de sa surface recouverte par le cercle. Un exemple de polygones sélectionnés lors d'une requête est disponible sur la figure 2.3.

2.3.1 Diagramme de Voronoï

Le diagramme de Voronoï tient une place centrale dans notre algorithme. En effet, la technique de découpage du plan en une constellation de polygones est inspirée de la décomposition de Voronoï, partition d'un espace métrique déterminée par

les distances à un ensemble discret de points. Évoquons brièvement le formalisme mathématique de ce diagramme.

Définition : "Le diagramme de Voronoï d'un ensemble S de n points de R^m est une partition de l'espace en n cellules représentant les zones d'influence des points de S : la cellule de Voronoï d'un point x de S est constituée de l'ensemble des points plus proches de x que de tout autre point de S." [5]

Problème Soit S un ensemble de n points du plan E, n≥3 , Il s'agit de décomposer l'espace en régions autour de chaque point p de S, telles que tous les points dans la région contenant p soient plus près de p que de n'importe quel autre point de S.

Il s'agit donc de s'intéresser aux médiatrices de points voisins de S.

Définition mathématique
La médiatrice des points p et q sépare le plan en deux demi-plans ; on note D(p,q) le demi-plan contenant p .

$$D(p,q) = \{M \in E, d(p,M) < d(q,M)\}$$

où d représente la distance euclidienne.

La région (ou cellule) de Voronoï du point p est

$$R(p) = \bigcap_{q \in S} D(p,q)$$

on a alors

$$R(p) = \{M \in E, \forall q \in S, d(M,p) < d(M,q)\}$$

Le diagramme de Voronoï est

$$D = \bigcup_{p,q \in S} \bar{R(p)} \cap \bar{R(q)}$$

La frontière commune à deux régions de Voronoï s'appelle une arête de Voronoï. Il s'agit d'une portion de la médiatrice de deux points de S. Les extrémités des arêtes sont les sommets de Voronoï.

Source : Auclair-Semere S., Jacquin M., Lakritz E et Sanchez A-M.

5. Pilaud V. (2006).

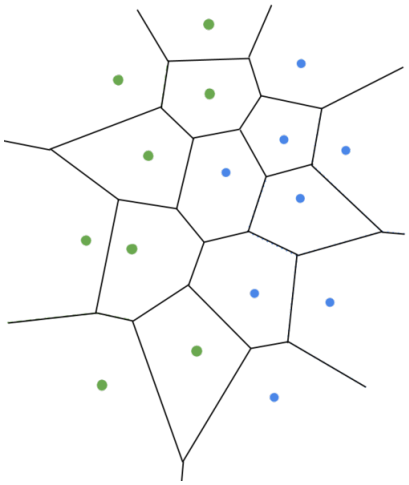

FIGURE 2.2 – Diagramme de Voronoï

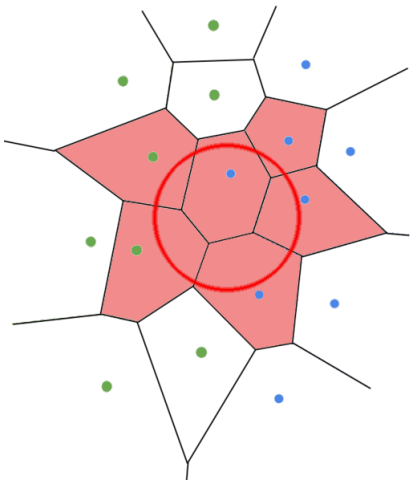

FIGURE 2.3 – Polygones sélectionnés

Afin de créer le diagramme, l'algorithme de Voronoï prend en entrée une série de points appelés germes. Dans notre cas, il s'agira des adresses. Il va ensuite délimiter un polygone par germe. Ce polygone couvre les points du plan pour lesquels le germe est le plus proche. Nous avons donc autant de polygones que de germes fournis en entrée.

2.3.2 Découpage en niveaux

Pour rappel, nous avons pour le moment un ensemble de polygones recouvrant tout le plan. Cet ensemble de polygones constitue le niveau 0 de notre algorithme. Il y a pour l'instant une seule adresse par polygone, ils ont donc tous un poids de 1.

Nous avons ensuite décidé de fusionner les polygones entre eux en nous inspirant du modèle de Casper et de son fonctionnement bottom-up. En effet, ce niveau 0 a l'avantage d'être précis, mais ne nous permet pas à lui seul d'atteindre dans tous les cas le minimum de 80 adresses par requête. C'est pourquoi nous avons décidé de créer plusieurs niveaux afin de garder la précision et ajouter de l'anonymat lorsque cela est nécessaire.

Chaque niveau correspondant à des contraintes sur le poids des polygones. Ce poids doit être de plus en plus grand. Le premier niveau n'a pas de contrainte spécifique. Ensuite, à partir du deuxième niveau, le poids minimum requis est de plus en plus élevé. Pour le dernier niveau, le poids minimum est fixé à 80. Ce poids correspond, comme déjà mentionné, à la borne minimale imposée par Bpost.

Lors de la construction de ces niveaux, nous rassemblons donc entre eux certains polygones. Comme expliqué précédemment, Bpost souhaite limiter le nombre de centres de distribution affecté par une requête. Dans ce but, nous avons décidé de ne jamais fusionner deux polygones qui appartiendraient à des centres de distribution différents.

Bpost souhaite également limiter le nombre de tournées affectées par une requête. Afin de répondre à cette exigence, nous avons décidé de fusionner en priorité les polygones appartenant à une même tournée.

Une fois que nous avons construit ces différents niveaux, nous pouvons réaliser les requêtes désirées. L'algorithme va tout d'abord regarder au premier niveau (là où les polygones ont les poids les plus petits). Si la somme des poids des polygones ayant au moins une partie de leur surface recouverte par le cercle dépasse 80, ce nombre est retourné. Dans le cas contraire, l'algorithme va passer au niveau supérieur. Les polygones étant plus grands, leur poids sera plus grand et par conséquent, la somme obtenue sera plus élevée ou égale à celle obtenue précédemment. L'algorithme répétera l'opération jusqu'à trouver un nombre supérieur ou égal à 80.

C'est donc grâce à ces 5 niveaux, que nous pourrons à la fois être précis et atteindre les objectifs d'anonymat.

En effet, pour chaque requête, quel que soit le point central et quel que soit le rayon fourni par le client, l'algorithme retournera au minimum un polygone regroupant lui-même au minimum 80 adresses. Ceci se vérifie, car dans le pire des cas, l'algorithme atteindra le dernier niveau composé de polygones dont le poids dépasse 80. Quel que soit le cercle donné par l'utilisateur, nous aurons donc au moins un polygone sélectionné. Ce polygone aura donc bien un poids supérieur à 80 et le critère d'anonymat sera respecté

En ce qui concerne les 4 premiers niveaux, ils permettent d'obtenir un nombre plus proche de la réalité que si nous n'avions que le dernier niveau. En effet, le cas exprimé précédemment est le cas critique, dans une majorité des cas, lorsque le rayon a une taille minimum (quelques dizaines de mètres), l'algorithme trouve une somme de 80 adresses avant d'atteindre le dernier niveau.

2.3.3 Risques d'attaques

À l'instar de l'attaque par le centre dont est victime l'algorithme Center Cloak, notre algorithme risque également d'être victime de certaines attaques. En réalité, le fait d'utiliser les différentes couches que nous avons construites ne nous permet pas de nous assurer complètement qu'il soit impossible de déterminer avec précision une zone contenant moins de 80 adresses.

À l'aide de requêtes suffisamment petites et nombreuses, il doit être possible dans certains cas de trouver de telles zones. En effet, dans l'exemple présenté à la figure 2.4, nous pouvons voir un polygone comportant 10 adresses entouré par des polygones contenant plus de 80 adresses. Lorsque nous faisons une requête dans l'un des polygones comportant plus de 80 adresses, nous obtenons en retour son poids. Lorsque la requête chevauche à la fois un polygone de poids supérieur à 80 et le polygone de poids 10, nous obtenons en retour la somme des deux. Ainsi, en multipliant les requêtes tout au tour du polygone de taille 10, il sera possible de déterminer où il se trouve précisément. Ceci peut se faire même si nous ne connaissons la position d'aucun polygone.

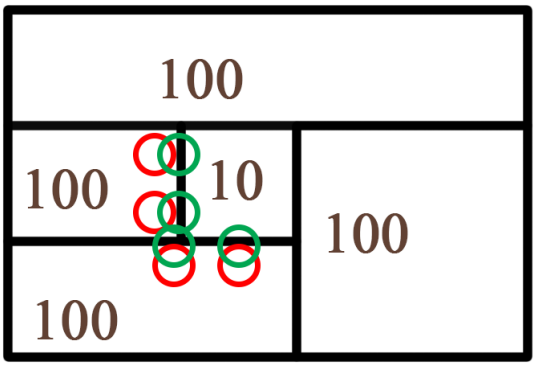

FIGURE 2.4 – Exemple de situation d'attaque

Afin d'éviter cette attaque, nous pourrions utiliser uniquement le dernier niveau que nous avons créé. En effet, ce niveau ne possède que des polygones dont le poids dépasse 80. Toutes les variations observables seraient donc supérieures à 80 et l'attaque ne pourrait donc plus révéler des polygones trop petits.

Nous analyserons donc les résultats que produit notre algorithme en utilisant uniquement cette dernière couche afin de pouvoir envisager cette solution.

Nous tenons tout de même à souligner que cette attaque est possible seulement dans certains types de configurations très particulières dont fait partie l'exemple cité précédemment. Il est également important de souligner que les moyens à mettre en œuvre pour réaliser ce type d'attaque sont relativement lourd. Il serait probablement plus simple et efficace d'aller simplement sur Google Map afin de repérer des bâtiments isolés.

Chapitre 3

Les algorithmes de protection de la vie privée dans la littérature

Afin de créer notre algorithme, nous nous sommes intéressés à plusieurs algorithmes. Ces algorithmes nous ont permis de découvrir différents principes et solutions permettant de protéger la vie privée des utilisateurs. La plupart des algorithmes auxquels nous nous sommes intéressés abordent un cas un petit peu différent que celui traité dans notre travail.

L'exemple type de la situation abordée dans ces algorithmes serait celle d'un individu qui utilise une application nécessitant sa localisation. Prenons l'exemple d'une application lui permettant de trouver les meilleurs restaurants à proximité de sa localisation à un moment donné. Lorsqu'il effectue sa demande auprès du serveur, il sera bien obligé de lui dévoiler sa localisation afin que celui-ci lui retourne la liste adéquate de restaurants.

Le dispositif mis en place est illustré à la figure 3.1. L'utilisateur va donc envoyer sa localisation à un intermédiaire de confiance qui se chargera d'anonymiser la requête afin de l'envoyer au serveur fournissant le service désiré. C'est dans cette phase d'anonymisation qu'interviennent les algorithmes auxquels nous nous sommes intéressés.

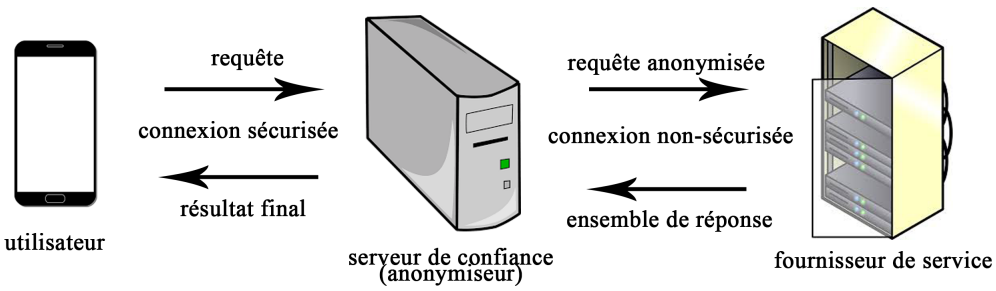

FIGURE 3.1 – Modèle de préservation de la vie privée

L'enjeu de l'intermédiaire chargé de l'anonymisation est donc de pouvoir à la fois transmettre l'information au fournisseur de service afin d'obtenir la réponse souhaitée, mais également d'empêcher ce fournisseur de déterminer la position exacte d'un utilisateur.

Un concept clé dans ce domaine est le k-anonymat. On désigne par ce terme le fait d'être anonyme à un degré k. Plus précisément, le serveur dispose de k utilisateur(s) associé(s) à k localisation(s) mais il ne sait pas quel utilisateur est associé à quelle localisation.

Il existe de nombreuses façons de choisir les k-1 utilisateurs que l'on va associer à l'individu titulaire de la requête. Nous nous sommes intéressés à trois algorithmes en particulier qui sont décrits dans les sections suivantes.

Il s'agit de :

— Nearest Neighbor Cloak
— Hilbert Cloak
— Casper

3.1 Nearest Neighbor Cloak

Cet algorithme [6] est une version améliorée de l'algorithme Center Cloak [7]. Center cloak est, quant à lui, un algorithme relativement simple qui, lorsqu'on cherche à avoir une k-anonymité, sélectionne les k-1 plus proches utilisateurs afin de les envoyer ensemble dans la même requête. Cet algorithme naïf présente un point faible important. En effet, il est facilement possible de retrouver l'utilisateur à l'origine de la requête, car il se trouve statistiquement au centre des k utilisateurs envoyés. Ceci est plus communément appelé "attaque par le centre".

6. Kalnis, P., Ghinita, G., Mouratidis, K., & Papadias, D. (2007), p. 1724.
7. Gkoulalas-Divanis, A., Kalnis, P., & Verykios, V. S. (2010), p. 6.

Nearest Neighbor Cloak débute tout d'abord comme Center cloak, il sélectionne les k-1 plus proches utilisateurs de notre utilisateur à l'origine de la requête (U_0). Dans l'exemple, présent à la figure 3.2, ces utilisateurs sont U_1 et U_2. Ensuite, il choisit un de ces derniers utilisateurs de manière aléatoire (dans notre exemple U_2). Il construit l'ensemble S_1 qui comprend les k-1 utilisateurs les plus proches de l'utilisateur choisi aléatoirement (U_2). Dans notre exemple, l'ensemble S_1 reprend les utilisateurs U_2, U_3 et U_4. Enfin, il retourne l'ensemble $S_2 = S_1 \bigcup U_0$.

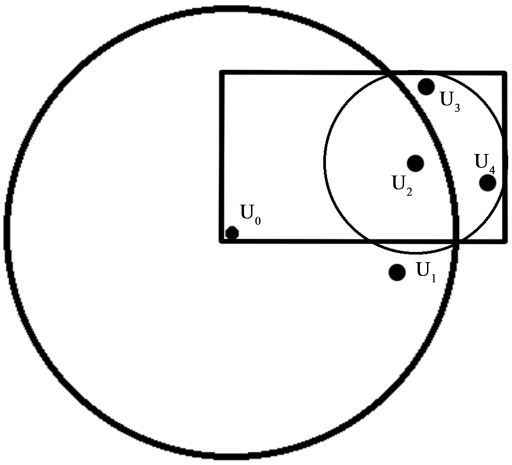

FIGURE 3.2 – Exemple de l'algorithme NNC avec k=3

Cet algorithme n'est plus sensible aux attaques par le centre, mais il reste possible de déterminer la position de l'utilisateur U_0 lorsque des valeurs aberrantes sont présentes. Sur notre exemple, nous pouvons voir qu'il sera facile de déterminer que l'utilisateur à l'origine de la requête est U_0. En effet, s'il s'agissait de l'un des trois autres, nous aurions comme ensemble de sortie U_1, U_2, U_3 et U_4.

3.2 Hilbert Cloak

L'algorithme [8] commence par créer une courbe transformant les coordonnées à deux dimensions en données à une dimension grâce à la fonction H(U). Lorsqu'il reçoit une requête de l'utilisateur U_0 avec un degré d'anonymité k, il trie les valeurs H(U) et les répartit en ensembles de taille k. Chaque ensemble a exactement k utilisateurs sauf le dernier qui peut en contenir jusqu'à 2k-1. Ensuite, l'algorithme retourne l'ensemble de k utilisateurs contenant l'utilisateur U_0.

8. Kalnis, P., Ghinita, G., Mouratidis, K., & Papadias, D. (2007). p. 1724

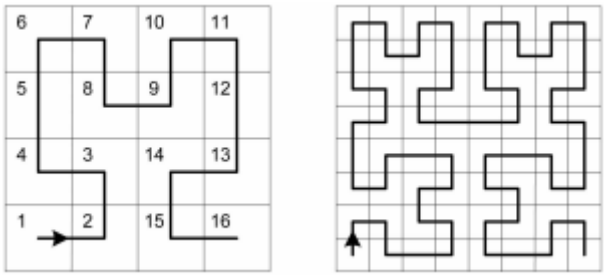

FIGURE 3.3 – Courbe d'Hiblert (a) 4x4, (b) 8x8

Sur la figure 3.3, nous avons une représentation de cette courbe pour un plan divisé en 16 et 64 carrés. Si nous avions par exemple un utilisateur dans chaque case, que nous désirions un degré d'anonymité de 4 et que notre utilisateur U_0 se trouvait dans la case 3, l'algorithme renverrait les utilisateurs des cases 1, 2, 3 et 4. Il diviserait la courbe en 4 afin d'avoir 4 utilisateurs dans chaque segment.

Cet algorithme possède une propriété très intéressante. En effet, il respecte le principe de réciprocité, c'est-à-dire qu'il renvoie un ensemble S de taille k (incluant l'utilisateur U_0) ou plus. Mais pour chaque utilisateur U_i présent dans cet ensemble S, il renverra également l'ensemble S. Dans notre exemple, si notre utilisateur U_0 était dans la case 1, 2 ou 4, nous aurions donc toujours le même ensemble de retour.

3.3 Casper

"L'idée principale est d'employer une structure de données pyramidale complète basée sur une grille qui décompose hiérarchiquement l'espace spatial en H niveaux où un niveau de hauteur h a 4h cellules sur sa grille. La racine de la pyramide est de hauteur zéro et n'a qu'une seule cellule sur sa grille qui couvre tout l'espace. Chaque cellule de la pyramide est représentée par (cid, N) où cid est l'identificateur de la cellule tandis que N est le nombre d'utilisateurs mobiles dans les limites de la cellule" [9].

9. Mokbel, M. F., Chow, C. Y., & Aref, W. G. (2006, September). p. 765

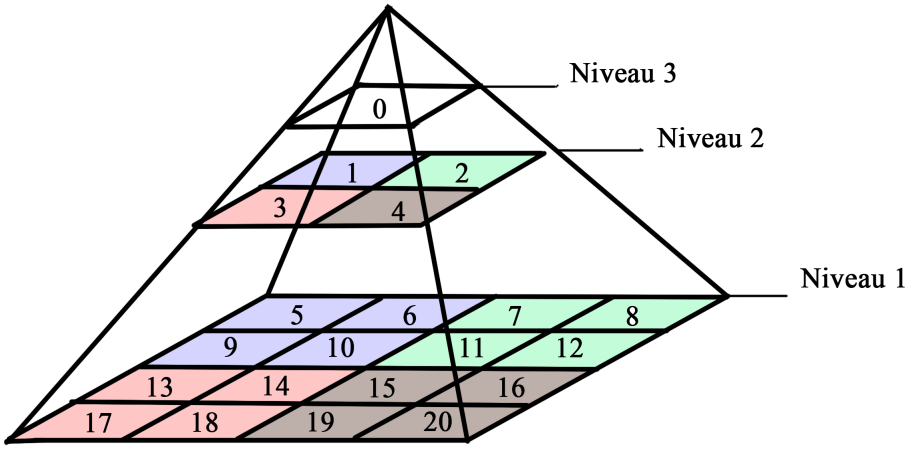

FIGURE 3.4 – Structure de données pyramidale

Vient ensuite l'algorithme de dissimulation. L'algorithme proposé est bottom-up pour la structure de données pyramidale présentée ci-avant. Il prend en entrée :

— l'identifiant de la cellule du niveau le plus élevé où se trouve l'utilisateur
— k, le degré d'anonymité souhaité
— A_{min}, la surface minimale de la zone qui sera renvoyée

L'algorithme commence par vérifier si la cellule dans laquelle se trouve l'identifiant de l'utilisateur U_0 contient un nombre d'utilisateurs supérieurs à k, ainsi que si sa surface est supérieure à A_{min}. Si c'est le cas, il retourne l'ensemble des utilisateurs présents dans cette cellule.

Si ce n'est pas le cas, il cherche s'il peut combiner cette cellule avec une cellule voisine. Une cellule est voisine si elle se trouve dans la même ligne ou colonne et a la même cellule mère au niveau supérieur (par exemple dans la figure 3.4, la cellule 10 a comme voisine la cellule 6 et 9, la cellule 11 et 14 n'ont pas la même cellule mère). Chaque cellule a donc deux voisines (une verticale et une horizontale). Si les conditions sont respectées pour l'une des deux combinaisons, les utilisateurs de la cellule voisine ainsi que ceux de la cellule initiale (où se trouve U_0) sont retournés.

Si ce n'est pas le cas, l'algorithme passe au niveau supérieur. Il reproduit cela jusqu'à respecter les critères imposés (le nombre d'utilisateurs $\geq$ k, l'aire des cellules sélectionnées $\geq A_{min}$).

Algorithme 1 Bottom–up cloaking algorithm

```
1:  Function BottomUp–Cloaking(k, A_min, cid)
2:  if cid.N >= k and cid.Area >= A_min then
3:      return Area(cid);
4:  end if
5:  cid_V = The vertical neighbor cell of cid.
6:  cid_H = The horizontal neighbor cell of cid.
7:  N_V = cid.N + cid_V .N, N_H = cid.N + cid_H.N
8:  if (N_V >= k OR N_H >= k) AND 2cid.Area >= A_min then
9:      if (N_H >= k AND N_V >= k AND N_H <= N_V )
        OR N_V < k then
10:         return Area(cid) U Area(cid_H);
11:     else
12:         return Area(cid) U Area(cid_V );
13:     end if
14: else
15:     BottomUp–Cloaking((k, A_min), Parent(cid));
16: end if
```

Chapitre 4

Algorithme et développement

4.1 Algorithme

Le développement se compose de deux grandes parties. La première consiste à traiter les données et à créer les différents niveaux. Ces niveaux contiennent chacun des polygones ayant un poids correspondant au nombre d'adresses recouvertes par ce polygone. La seconde grande partie consiste à utiliser ces différents niveaux afin de fournir le nombre d'adresses se situant à l'intérieur d'un cercle donné.

4.1.1 Création des différents niveaux

Comme expliqué précédemment, afin de dénombrer les adresses présentes dans le cercle donné, nous avons décidé de construire plusieurs niveaux. Dans ces niveaux, nous retrouverons principalement des polygones couvrant la carte de la Belgique avec une indication du nombre exact d'adresses se trouvant dans ce polygone. Le dernier niveau étant soumis à un poids minimum de 80 et soucieux de limiter le temps de création de ces différents niveaux, nous avons décidé de nous limiter à la création de 5 niveaux en tout.

Nous avons donc un premier niveau qui n'est soumis à aucune contrainte en matière de nombre d'adresses par polygone. Le deuxième niveau comporte au minimum 20 adresses par polygone, 40 pour le troisième, 60 pour le quatrième et 80 pour le dernier.

Création du premier niveau

Nous créons tout d'abord un diagramme de Voronoï grâce à la librairie *scipy.spatial*. Les germes utilisés sont les adresses. Nous obtenons donc un grand nombre de

polygones ayant un poids de 1. Un grand nombre, mais pas tous. En effet, il existe certaines adresses qui comportent plusieurs boites aux lettres. Nous considérons le nombre de boites comme poids. De plus, certaines adresses sont trop proches les unes des autres. Afin d'éviter les problèmes que cela pourrait causer avec la librairie, nous fusionnons les adresses concernées.

Nous pourrions déjà utiliser ce niveau pour notre algorithme, mais comme expliqué plus tôt, il est compliqué pour un facteur de devoir distribuer un grand nombre de flyers différents sur une section limitée. En revanche, comme nous l'avons déjà évoqué plus tôt, le premier niveau n'est pas soumis à une contrainte de taille. Nous avons donc décidé de fusionner les polygones des adresses partageant le même segment d'activité. *Un segment d'activité ou "Activity Segment" se compose de toutes les adresses se trouvant sur un même côté d'une route et est limité par les carrefours.* La figure 4.1 illustre ce concept.

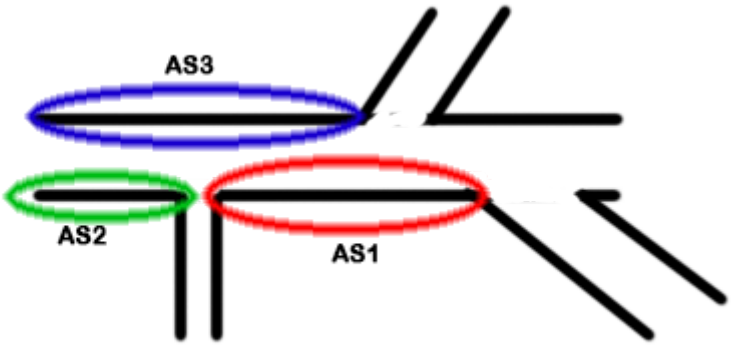

FIGURE 4.1 – Schéma d'exemple d'AS

Ceci permet d'assurer que, quoi qu'il arrive, nous ne devrons jamais distribuer le même flyer uniquement à une maison sur deux, ou un flyer différent à chaque maison dans une même portion de rue.

Création des autres niveaux

Pour les autres niveaux, nous sommes maintenant soumis à une contrainte sur le poids minimum des polygones. Le deuxième niveau étant soumis à un poids minimum de 20, 40 pour le troisième, 60 pour le quatrième et 80 pour le dernier.

L'un des critères imposés par Bpost était de limiter au maximum la sélection d'adresses se trouvant dans différents centres de distribution. Afin de respecter cela,

nous avons décidé de ne jamais créer un polygone reprenant des adresses de différents centres de distribution. Il est malgré tout toujours possible que l'utilisateur sélectionne des adresses appartenant à plusieurs centres de distribution. Dans ce cas, les adresses seront dans différents polygones. Nous pouvons donc l'avertir en temps voulu afin qu'il modifie sa requête ou accepte les modalités imposées par Bpost dans ce cas.

Comme nous ne désirions pas fusionner deux polygones relatifs à des centres de distribution différents, nous avons pu diviser notre calcul en regardant tour à tour les polygones relatifs à chacun des centres de distribution. Ceci nous permet de gagner énormément de temps à l'exécution. Nous avions dans un premier temps envisagé de réaliser le calcul de tous les centres de distribution de manière concurrente, mais nous n'avons pas conservé cette solution. En effet, dans cette version de la solution, nous devions récupérer tous les résultats produits dans une variable spécialement créée pour gérer les accès concurrents. Cette variable nécessitait un espace mémoire très grand. Malgré l'utilisation d'un ordinateur puissant (Intel(R) Xeon(R) CPU E5-2687W v3 @ 3.10GHz pour un total de 20 cœurs et 40 threads, accompagné de 128 Go de RAM) l'initialisation de cette variable nous était refusée.

Un autre critère imposé par Bpost était de regrouper les adresses sélectionnées par l'utilisateur sur un minimum de tournées possibles. Dans ce but, nous commençons par trier les polygones en fonction de la tournée à laquelle ils sont relatifs.

Nous trions ensuite à nouveau les polygones en fonction de leur poids. Ce deuxième tri permet de rencontrer le deuxième critère qui prévoit de créer des polygones avec un poids relativement bas afin de garantir une précision assez élevée.

Ces deux tris consécutifs ont pour effet d'ordonner les polygones par taille et au sein des groupes de polygones de même taille, les polygones sont ordonnés par tournée. Par conséquent, les polygones de même poids qui appartiennent à la même tournée restent ensemble.

Nous prenons ensuite les polygones n'ayant pas la taille requise un à un. Nous tentons de les fusionner avec un autre polygone ayant le poids le plus petit possible tout en respectant deux critères :

— La tournée à laquelle ils sont relatifs est la même.

— Les polygones se touchent.

Si ces deux critères sont respectés, les deux polygones sont extraits de la liste et fusionnés. Le résultat de cette fusion sera réinjecté dans la liste des polygones quand tous les polygones n'atteignant pas le poids minimal auront été parcourus.

Si lors d'un passage, aucun polygone de poids inférieur au poids minimum requis pour ce niveau ne trouve d'autre polygone avec lequel il pourrait fusionner, nous relaxons les contraintes. Dans ce cas, les polygones ne doivent plus être relatifs à la même tournée pour pouvoir fusionner. En revanche, la somme des poids des deux polygones doit dépasser le poids minimum requis pour ce niveau.

Lorsque tous les polygones ont un poids supérieur ou égal au poids minimum, nous passons aux polygones relatifs au centre de distribution suivant.

Lorsque nous avons réalisé ces étapes pour l'ensemble des centres de distribution, nous pouvons créer un Shapefile de ce niveau. Il s'agit d'un fichier reprenant tous les polygones du niveau ainsi que le poids de chaque polygone, le centre de distribution auquel il se rapporte, les tournées qui contiennent des adresses dans la zone couverte par le polygone, ainsi que les identifiants (PDP_ID) de toutes les adresses couvertes par le polygone. Ce Shapefile nous sera utile dans la seconde partie afin de dénombrer les adresses dans un périmètre donné autour d'un point.

Détection et suppression du bruit

La base de données fournie par Bpost ne fait pas exception à la règle, elle contient un certain nombre d'erreurs, on dit qu'elle contient du "bruit". Pour des raisons diverses, la localisation des adresses n'est pas toujours correcte (erreur d'encodage, information non disponible, ...). Une erreur courante est de placer les coordonnées d'une adresse au milieu de la rue dans laquelle elle se situe. Ceci arrive quand Bpost n'a pas trouvé de meilleure information. Ce type de problème est en constante évolution. En effet, Bpost bien conscient de ce problème tente de corriger ces erreurs et d'améliorer la qualité de sa base de données en termes d'exactitude, d'exhaustivité et de précision.

Afin d'obtenir un résultat intéressant, nous avons donc fait le choix d'enlever une partie de ces erreurs. Nous avons en particulier distingué deux cas problématiques. Il s'agit d'une part des erreurs relatives au niveau des adresses et d'autre part des erreurs relatives au niveau des segments d'activité.

Erreur au niveau des adresses

Le premier cas de figure qui s'est présenté à nous est le cas d'adresses très éloignées du reste des adresses appartenant au même segment d'activité (AS [10]). Ceci a pour principale conséquence de créer plusieurs polygones à l'issue de la fusion et non pas un seul et même polygone.

Nous aurions donc pu nous contenter du polygone ayant le plus grand poids

10. Activity Segment

en supposant les autres comme du bruit. Ceci serait une erreur. En effet, il existe un grand nombre de situations où il n'est pas impossible d'obtenir un ensemble de polygones à la place d'un seul. Un exemple montrant une de ces situations est représenté sur la figure 4.2.

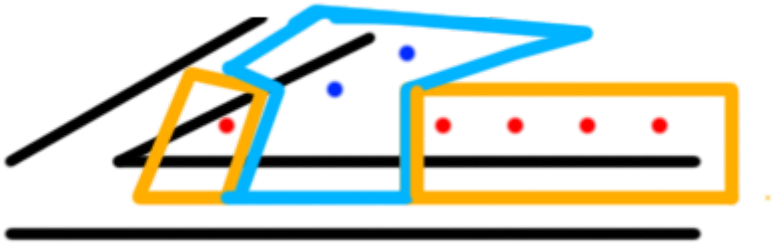

FIGURE 4.2 – Exemple de situation potentielle avec un ensemble de polygones

Nous avons donc décidé de supprimer les adresses se trouvant dans un des polygones résultant de la fusion si et seulement si ce dernier est le même polygone que celui généré par le diagramme de Voronoï et qu'il se trouve à au moins 500 mètres de tout autre polygone du même AS.

Nous avons cependant décidé de considérer ces adresses comme "mal localisées". Nous ajoutons donc leurs identifiants et un poids correspondant au nombre d'adresses "mal localisées" à l'un des autres polygones non supprimés de cet AS.

Erreur au niveau des AS

Le second cas de figure est plus problématique encore pour l'exécution de l'algorithme. Nous avons imposé que pour être fusionnés, deux polygones doivent obligatoirement se toucher. Or, nous tentons de fusionner uniquement les polygones relatifs au même centre de distribution. Nous avons donc un problème lorsqu'un polygone (résultant de la fusion par AS) se retrouve entouré uniquement par d'autres polygones relatifs à un autre centre de distribution. Un exemple est illustré sur la figure 4.3.

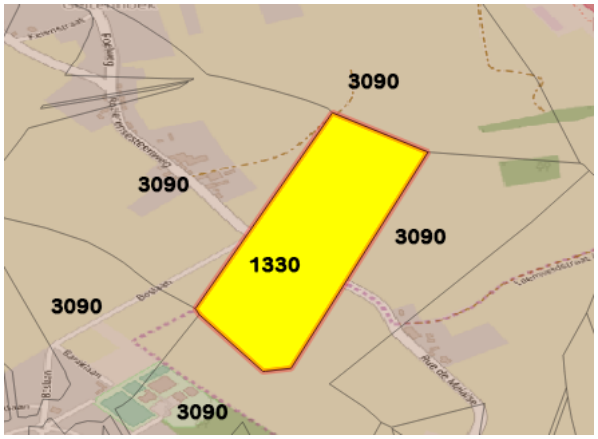

FIGURE 4.3 – Exemple d'un AS entouré de polygones d'autres centres de distribution

Nous avons donc décidé de supprimer complètement toutes les adresses se trouvant dans la zone couverte par le polygone associé à ces AS.

Pseudo code

```
input: Data[id, AS_ID, Longitude, latitude,
            DistributionCenter_ID, round, box_number)

    points = points_collection(input)
    regions = Voronoi(points)
    noise_removal(regions)
    level[1] = first_level_creation(regions)

    for i = 2 to 5 do
        level[i] = level_creation(i,level[i-1])
    end
```

```
level_creation(position,level)
    minimal_bound = (position-1)*20
    relax = 0

    while true do
        tri_par_weight(level)
        polygons = level.polygons()

        for dc_id in level.dc() do
            for i in polygons.size() do
                if polygons[i].weight()>= minimal_bound then
                    break
                else:
                    for j= i+1 to polygons.size() then
                        if polygons[j] not yet merged and
                            check(polygons[i],polygons[j],relax) then
                            polygons.merge(i,j)
                            break
                        endif
                    end
                endif
            end
        end

        if number_of_merge = 0 then
            if polygons[0].weight() >= minimal_bound then
                return level(polygons)
            else
                relax ++
            endif
        endif
    end
```

4.1.2 Dénombrement

Maintenant que nous avons supprimé les erreurs dans les données et que nous avons créé les différents niveaux, nous pouvons passer à la dernière étape. Cette dernière étape consiste à utiliser les différents niveaux créés précédemment afin de dénombrer le nombre d'adresses présentes dans une zone circulaire définie par un point central et un rayon.

L'algorithme de dénombrement commence avec le premier niveau. Ce niveau contient les polygones les plus petits.

L'algorithme effectue une requête sur ce niveau. La requête vise à connaître la somme des poids des polygones dont au moins une partie de la surface est

recouverte par le cercle défini par le point P et le rayon R reçu en entrée. Si ce nombre est plus élevé que 80, il retourne ce nombre et l'algorithme s'arrête là.

Si ce n'est pas le cas, il fait une nouvelle requête sur le niveau supérieur. Il continue jusqu'à obtenir un résultat satisfaisant.

Comme dit précédemment, le dernier niveau étant soumis à un poids minimum de 80, quels que soient le point et le rayon en entrée, nous pourrons communiquer un nombre supérieur ou égal à 80.

4.1.3 Variantes

Version 2

Nous avons expliqué plus tôt la manière dont nous créons les cinq niveaux. Nous avons également envisagé une variante à cet algorithme. L'idée générale reste bien entendu la même. La différence se marque dans les critères nous permettant de fusionner deux polygones. En effet, la première version considère la fusion possible seulement lorsque les polygones se touchent. Cela peut sembler indispensable à priori, mais la réalité du terrain nous complique quelque peu la vie.

Dans cette variante, nous avons donc décidé de permettre dans certains cas de fusionner deux polygones même s'ils ne se touchent pas.

Lorsque nous parcourons les polygones à fusionner (toujours dans le même ordre que dans la première version) nous commençons avec les mêmes contraintes que dans la version initiale. Nous cherchons donc à fusionner deux polygones s'ils font partie de la même tournée et qu'ils se touchent.

Lorsque nous avons parcouru une fois tous les polygones dont le poids est sous le seuil du niveau en cours, et que nous n'avons trouvé aucune possibilité de fusion, nous cherchons à fusionner des polygones appartenant à la même tournée, mais cette fois le second critère n'est plus le contact entre les polygones, mais la distance entre ceux-ci qui doit être inférieur à 100 mètres. La troisième phase reste similaire à la première version. Pour rappel, le critère d'appartenance à la même tournée n'est plus contraignant, mais nous cherchons à fusionner des polygones qui se touchent et dont la somme des poids est supérieure au seuil du niveau considéré.

point les plus proches

Version 3

La troisième version, tout comme la deuxième, reste proche de la première. La différence apportée dans cette version se situe également dans les critères imposés lors de la fusion de deux polygones. Comme nous l'avons expliqué, lorsque nous avons tenté de fusionner les polygones n'ayant pas encore le poids requis et que nous n'avons pas réussi à fusionner un seul de ces polygones restants, nous relaxons

le problème. Lorsque nous relaxons ce problème, nous imposons dans la première version que les polygones ne doivent plus spécialement être dans la même tournée, mais que la somme de leurs poids doit être supérieure au seuil imposé pour le niveau considéré.

Ce critère est hérité d'une succession de versions et semble contre-intuitif. Il permettait notamment de gagner un temps précieux, car une fois fusionnés, les polygones avaient un poids supérieur au seuil et donc ne nécessitaient plus d'être traités par l'algorithme pour le niveau considéré.

Dans cette version, lorsque nous relaxerons le problème, les polygones pourront donc être fusionnés simplement s'ils se touchent. La somme des poids des deux polygones ne devra donc plus spécialement être supérieure au seuil imposé pour le niveau.

Version 4

La quatrième version regroupe les deux variantes précédentes. C'est-à-dire que lorsque nous relaxerons le problème, nous pourrons fusionner deux polygones simplement s'ils sont à moins de 150 mètres.

4.2 Choix du langage de programmation

Le choix du langage utilisé dans le cadre de ce projet a été motivé par plusieurs éléments. Tout d'abord, comme vous avez pu le constater dans les sections précédantes, nous avions besoin de pouvoir générer un diagramme de Voronoï. Nous avons trouvé une librairie répondant à nos critères en Python.

Une autre raison est en lien avec QGIS. En effet, afin de visualiser les fichiers créés lors de la première phase, nous utilisons QGIS. "QGIS est un Système d'Information Géographique (SIG) convivial distribué sous licence publique générale GNU. C'est un projet officiel de la fondation Open Source Geospatial (OSGeo). Il est compatible avec Linux, Unix, Mac OS X, Windows et Android et intègre de nombreux formats vecteur, raster, base de données et fonctionnalités". [11] Les avantages de ce programme sont donc nombreux, c'est un logiciel libre, gratuit, compatible avec la majorité des systèmes d'exploitation. Il nous est également possible de créer nous-mêmes des plug-ins que nous pouvons ensuite intégrer à QGIS. Ces plug-ins utilisent Python.

11. https ://www.QGIS.org/fr/site/about/index.html

Python est également un langage de programmation avec lequel nous sommes familiers. Ce qui nous a permis de ne pas perdre de temps lors du développement de l'algorithme.

Ces trois éléments ont été les principaux facteurs de décision lors du choix du langage de programmation à utiliser. En effet, la facilité d'utilisation de par la connaissance existante et l'existence de librairies indispensables ainsi que la cohérence entre les différentes parties de code étaient à nos yeux les critères les plus importants à prendre en compte.

4.3 Plug-in QGIS

Afin de pouvoir tester l'efficacité de l'algorithme de création de niveaux et de l'algorithme de dénombrement en condition réelle, nous avons décidé d'implémenter un plug-in QGIS. Cette implémentation fournira un outil graphique et visuel relativement simple d'utilisation. Cet outil peut également servir de base pour l'élaboration d'un outil utilisable par les clients de Bpost.

Afin de créer cet outil nous avons modifié un plug-in existant dans la version 2 de QGIS (*Multiple Layer selection*[12]). Nous l'avons tout d'abord rendu compatible avec la version 3 de QGIS. En effet, un certain nombre de librairies ont été modifiées entre la version 2 et la version 3 de QGIS.

Nous nous sommes ensuite consacrés à ne garder que la sélection du niveau qui nous intéressait. En réalité, la fonctionnalité initiale de ce plug-in est de sélectionner tous les polygones étant recouverts au moins en partie par la zone sélectionnée. Nous désirions qu'il garde seulement la sélection pour le premier niveau nous donnant un nombre d'adresses supérieur à 80.

Pour cela, l'algorithme sélectionne tous les polygones ayant au moins un point dans la zone établie par l'utilisateur. Si la somme des poids de tous ces polygones est supérieure à 80, il renvoie ce nombre et cache les autres niveaux. Si ce n'est pas le cas, il cache ce niveau et recommence avec le niveau suivant.

Enfin nous avons encore modifié quelque peu le plug-in initial afin d'obtenir une sélection circulaire et non rectangulaire.

12. https ://github.com/felferrari/MultipleLayerSelection

Chapitre 5

Analyse de l'algorithme

5.1 Évaluation

Il est à présent temps de déterminer l'efficacité de notre algorithme. Dans ce but, nous nous attarderons sur plusieurs mesures effectuées sur les données produites par l'algorithme. Nous comparerons également les résultats obtenus pour les quatre versions introduites précédemment. Enfin, nous regarderons également les résultats fournis par ces mêmes versions mais lorsque nous utilisons uniquement le cinquième et dernier niveau.

5.1.1 Précision

Comme indiqué précédemment, la précision du dénombrement est un critère important dans cet algorithme. Afin de déterminer si sa précision est acceptable, nous avons réalisé un grand nombre de requêtes. Afin d'avoir un ensemble représentatif de requêtes, nous avons sélectionné 1145 adresses,une par code postal. Ceci permet d'avoir des requêtes réparties sur tout le territoire, aussi bien en ville qu'en zone rurale. Une visualisation de la localisation de ces adresses est disponible à la figure 5.1.

utilisée comme centre

Nous avons testé la précision avec des requêtes ayant trois rayons différents, 100 mètres, 500 mètres et 1 km. Dans les tableaux suivants (tableau 5.1 à 5.8) reprenant les résultats, l'erreur est calculée de cette façon :

$$\frac{\text{nombre d'adresses dénombré} - \text{nombre réel d'adresse}}{\text{nombre réel d'adresse}} * 100$$

Nous faisons également la distinction entre l'erreur totale englobant tous les points quel que soit le nombre d'adresses se trouvant réellement dans ce périmètre et l'erreur réelle qui ne prend en compte que les points où il y a en réalité plus de 80 adresses. En effet, notre but étant de ne pas révéler les zones où il n'y a qu'un

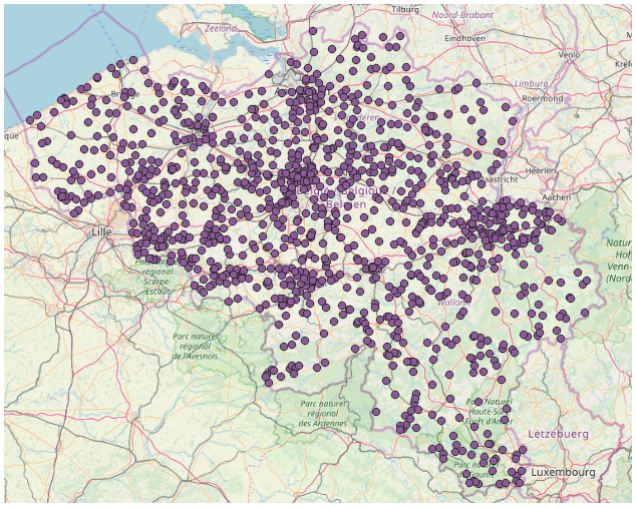

FIGURE 5.1 – Visualisation des adresses utilisées pour le test de précision

petit nombre d'adresses, il est normal qu'elles n'apparaissent pas en sortie.

Vous trouverez ci-après, tout d'abord les résultats de l'algorithme utilisant les 5 niveaux et ensuite les résultats de l'algorithme utilisant uniquement le cinquième et dernier niveau.

Utilisation des 5 niveaux

Rayon (m)	Erreur totale (%)	Erreur nombre réel > 80 (%)
100	16.70	0.69
500	2.89	0.21
1000	1.02	0.13

TABLE 5.1 – Erreur de dénombrement des adresses dans un périmètre donné (version 1, utilisation des 5 niveaux)

Rayon (m)	Erreur totale (%)	Erreur nombre réel > 80 (%)
100	16.76	0.69
500	2.93	0.21
1000	1.02	0.13

TABLE 5.2 – Erreur de dénombrement des adresses dans un périmètre donné (version 2, utilisation des 5 niveaux)

Rayon (m)	Erreur totale (%)	Erreur nombre réel > 80 (%)
100	16.76	0.69
500	2.89	0.21
1000	1.02	0.13

TABLE 5.3 – Erreur de dénombrement des adresses dans un périmètre donné (version 3, utilisation des 5 niveaux)

Rayon (m)	Erreur totale (%)	Erreur nombre réel > 80 (%)
100	16.78	0.69
500	2.93	0.21
1000	1.02	0.13

TABLE 5.4 – Erreur de dénombrement des adresses dans un périmètre donné (version 4, utilisation des 5 niveaux)

Utilisation uniquement du cinquième et dernier niveau

Rayon (m)	Erreur totale (%)	Erreur nombre réel > 80 (%)
100	48.44	5.96
500	13.47	2.22
1000	6.18	1.74

TABLE 5.5 – Erreur de dénombrement des adresses dans un périmètre donné (version 1, utilisation uniquement du cinquième et dernier niveau)

Rayon (m)	Erreur totale (%)	Erreur nombre réel > 80 (%)
100	48.36	5.98
500	13.65	2.24
1000	6.26	1.73

TABLE 5.6 – Erreur de dénombrement des adresses dans un périmètre donné (version 2, utilisation uniquement du cinquième et dernier niveau)

Rayon (m)	Erreur totale (%)	Erreur nombre réel > 80 (%)
100	48.39	5.97
500	13.48	2.22
1000	6.17	1.73

TABLE 5.7 – Erreur de dénombrement des adresses dans un périmètre donné (version 3, utilisation uniquement du cinquième et dernier niveau)

Rayon (m)	Erreur totale (%)	Erreur nombre réel > 80 (%)
100	48.72	5.99
500	13.69	2.24
1000	6.27	1.73

TABLE 5.8 – Erreur de dénombrement des adresses dans un périmètre donné (version 4, utilisation uniquement du cinquième et dernier niveau)

En ce qui concerne la version de l'algorithme utilisant les 5 niveaux, nous remarquons que les résultats sont relativement bons. En effet, en ce qui concerne l'erreur réelle, elle se situe en dessous de 1 %. Nous pouvons également remarquer que cette erreur a tendance à diminuer avec l'augmentation du rayon. Ceci est une bonne chose, car les requêtes autorisées pour le moment par Bpost doivent avoir un rayon d'au moins 1 km.

Nous remarquons que la différence n'est pas très marquée entre les quatre versions de l'algorithme. La deuxième et la quatrième versions sont toutefois légèrement moins satisfaisantes que les deux autres. Il n'est pas surprenant d'observer ces résultats, car ces deux versions augmentent encore le nombre de polygones fusionnés. Ceci a pour conséquence d'augmenter le nombre d'adresses par polygones et donc entraîne un dénombrement plus important.

En revanche, il nous est difficile de ne pas remarquer la franche différence entre les résultats obtenus lorsque l'on utilise les 5 niveaux par rapport à ceux obtenus lorsque l'on utilise uniquement le dernier niveau. Ce n'est évidemment pas surprenant, car les polygones étant beaucoup plus grands et de poids plus élevés, le nombre d'adresses comptabilisées grimpe plus vite. Ces résultats ne sont toute fois pas désastreux et permettent tout de même d'éventuellement envisager cette solution.

5.1.2 Anonymat

Le critère que nous allons évaluer ensuite, est celui de l'anonymat. Avons-nous bien toujours un nombre supérieur à 80 adresses ? Pour cela, nous pouvons regarder les tableaux 5.9 à 5.12 présentant quelques informations sur le nombre d'adresses par polygone pour les quatre versions de l'algorithme.

Pour rappel, nous désirions obtenir un nombre d'adresses minimum au-dessus d'un seuil. Ce seuil augmente en fonction du niveau.

niveau	seuil	minimum	maximum	moyen	sous le seuil
1	1	1	897	6.94	0
2	20	1	897	36.58	84
3	40	1	897	62.75	95
4	60	1	897	125.69	98
5	80	1	897	256.09	98

TABLE 5.9 – Statistiques sur le nombre d'adresses par polygone en fonction du niveau (version 1)

niveau	seuil	minimum	maximum	moyen	sous le seuil
1	1	1	897	6.94	0
2	20	1	897	36.58	31
3	40	1	897	62.84	53
4	60	1	897	125.95	54
5	80	1	665	257.48	52

TABLE 5.10 – Statistiques sur le nombre d'adresses par polygone en fonction du niveau (version 2)

niveau	seuil	minimum	maximum	moyen	sous le seuil
1	1	1	897	6.94	0
2	20	1	897	36.58	81
3	40	1	897	62.75	92
4	60	1	897	125.67	95
5	80	1	897	256.03	95

TABLE 5.11 – Statistiques sur le nombre d'adresses par polygone en fonction du niveau (version 3)

niveau	seuil	minimum	maximum	moyen	sous le seuil
1	1	1	897	6.94	0
2	20	1	897	36.58	38
3	40	1	897	62.82	44
4	60	1	897	125.93	44
5	80	1	897	257.40	44

TABLE 5.12 – Statistiques sur le nombre d'adresses par polygone en fonction du niveau (version 4)

Force est de constater que ce n'est pas tout à fait le cas. Il subsiste à chaque niveau, un nombre limité de polygones dont le poids (nombre d'adresses) est inférieur au seuil fixé pour ce niveau. Nous pouvons trouver deux explications probables.

La première possibilité s'explique par des îlots de quelques AS au milieu d'un autre centre de distribution. Ce cas conduit à des polygones d'un poids entre 2 et 80. Ces îlots ne sont actuellement pas détectés, car, pour rappel, nous avons décidé de supprimer les adresses d'un AS dans le cas où il se retrouve seul au milieu d'un autre centre de distribution.

La seconde possibilité s'explique par la suppression du bruit. En effet, afin de gagner du temps lors de l'exécution, nous repérons les adresses "mal localisées" et les AS îlots en même temps. Ceci implique que lorsque nous supprimons ces adresses, nous pouvons recréer d'autres AS îlots. Cela vaudrait certainement la peine de mesurer l'impact des corrections apportées aux données.

La seconde observation que nous pouvons faire concerne la différence entre les quatre versions de l'algorithme. Si le problème des polygones demeurant sous le seuil requis, évoqué juste avant, reste valable dans tous les cas, la deuxième et la quatrième versions réduisent approximativement de 50% le nombre de polygones ayant un poids inférieur au seuil. Ceci n'est pas négligeable. Ceci signifie que

certains polygones qui ne pouvaient pas être fusionnés auparavant, le peuvent lorsque l'on relaxe le critère de contact des polygones.

5.1.3 Concentration des tournées

Un autre critère était de limiter autant que possible le nombre de tournées lors d'une requête. Pour vérifier cela, nous pouvons analyser le nombre de tournées dans chacun des polygones, et cela, pour chaque niveau.

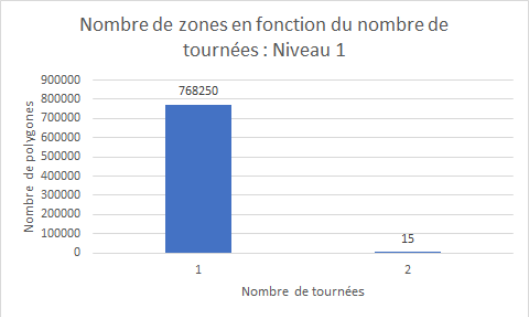

FIGURE 5.2 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 1, niveau 1)

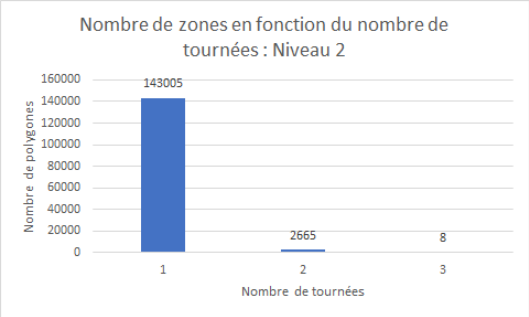

FIGURE 5.3 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 1, niveau 2)

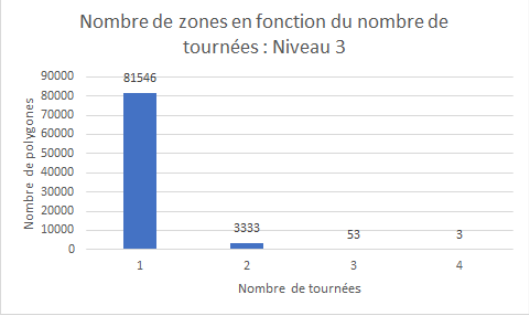

FIGURE 5.4 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 1, niveau 3)

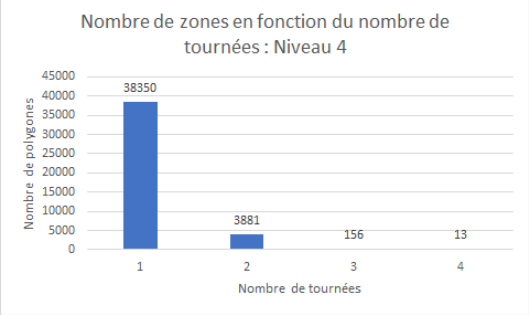

FIGURE 5.5 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 1, niveau 4)

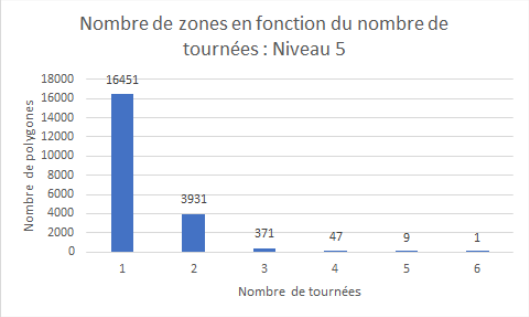

FIGURE 5.6 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 1, niveau 5)

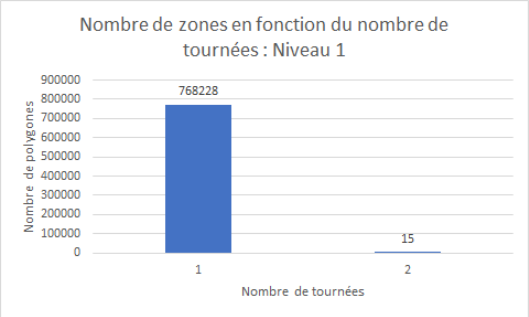

FIGURE 5.7 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 2, niveau 1)

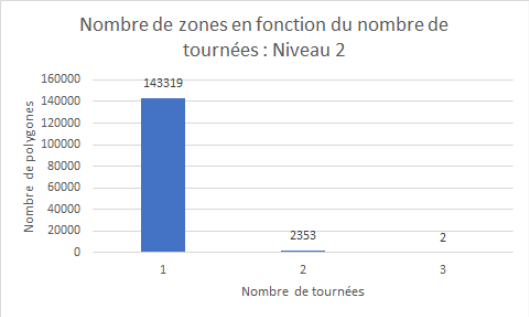

FIGURE 5.8 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 2, niveau 2)

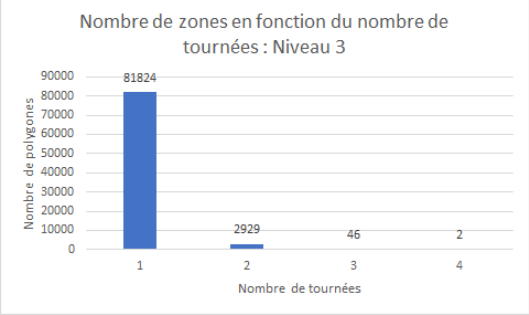

FIGURE 5.9 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 2, niveau 3)

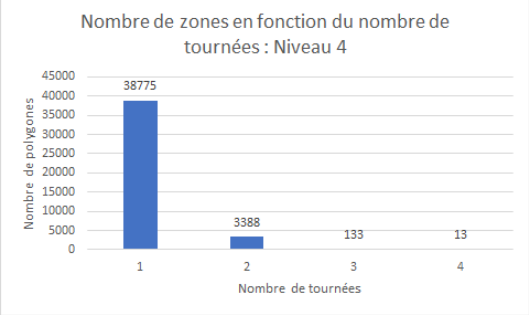

FIGURE 5.10 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 2, niveau 4)

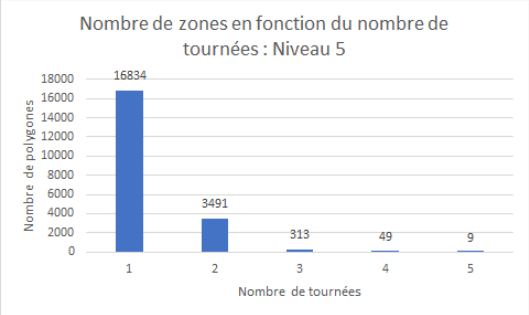

FIGURE 5.11 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 2, niveau 5)

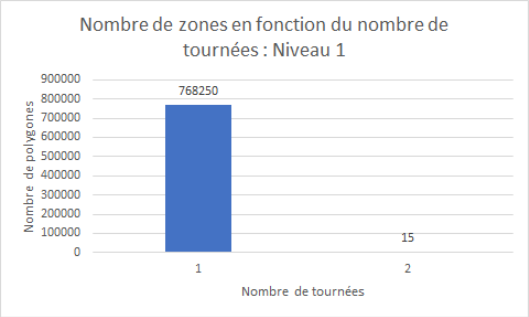

FIGURE 5.12 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 3, niveau 1)

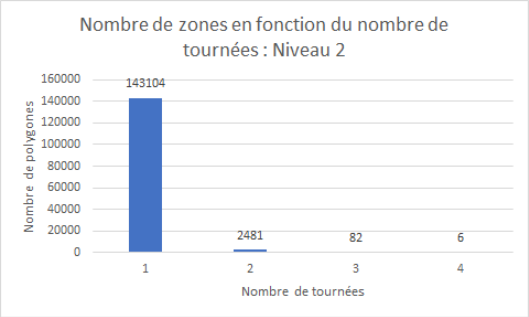

FIGURE 5.13 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 3, niveau 2)

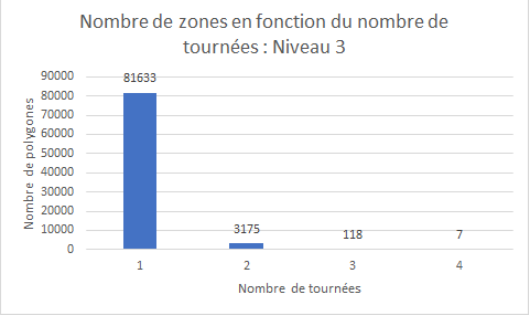

FIGURE 5.14 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 3, niveau 3)

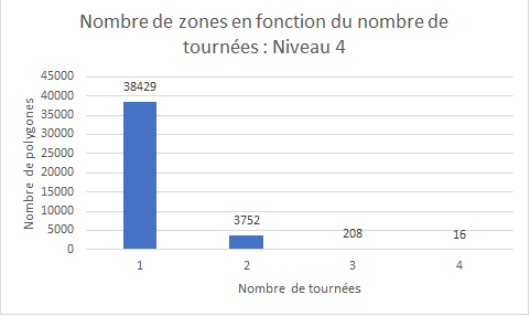

FIGURE 5.15 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 3, niveau 4)

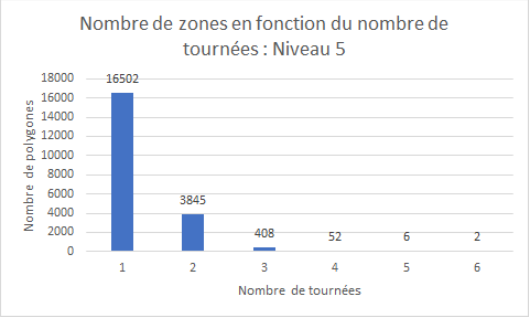

FIGURE 5.16 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 3, niveau 5)

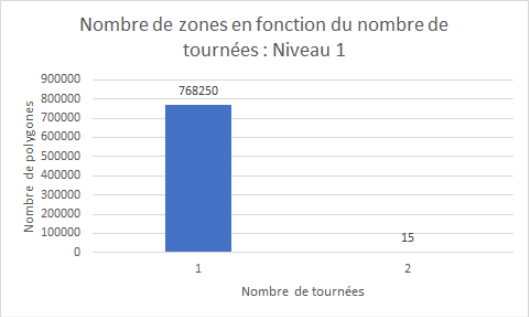

FIGURE 5.17 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 4, niveau 1)

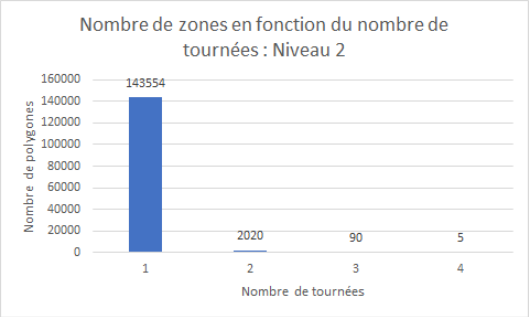

FIGURE 5.18 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 4, niveau 2)

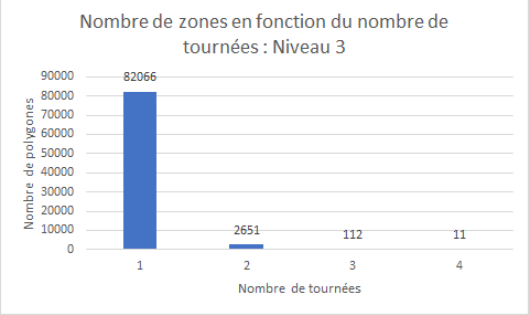

FIGURE 5.19 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 4, niveau 3)

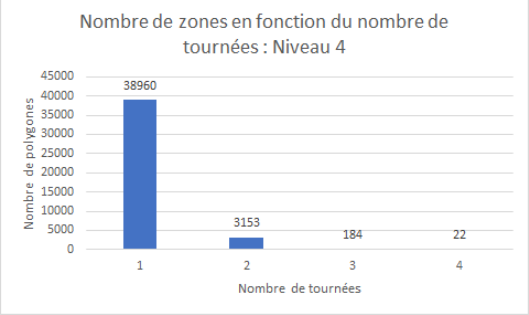

FIGURE 5.20 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 4, niveau 4)

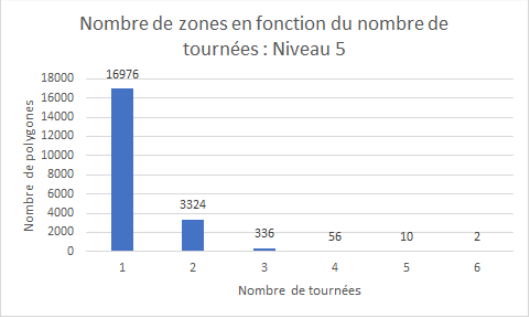

FIGURE 5.21 – Histogramme du nombre de polygones en fonction du nombre de tournées (version 4, niveau 5)

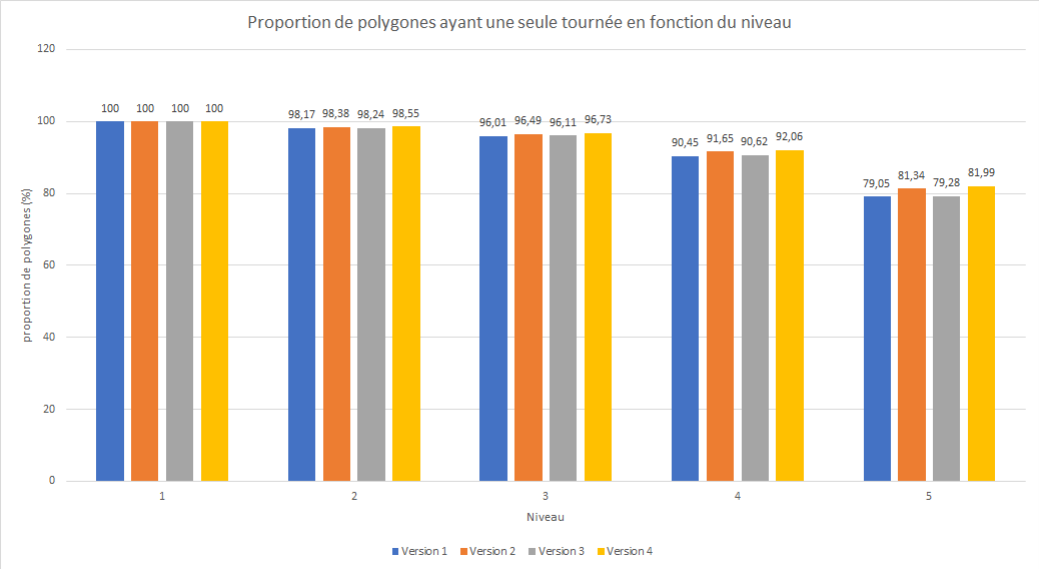

FIGURE 5.22 – Résumé des proportions de polygones ayant une seule tournée en fonction du niveau

Nous observons qu'une très grande majorité de ces polygones se composent uniquement d'adresses se trouvant dans une même tournée. Ceci impliquera forcément un nombre limité de tournées par requête. Ce nombre reste bien sûr relatif à la dimension du périmètre de la requête.

La différence entre les quatre versions de l'algorithme est limitée. Le niveau 1 est strictement identique, ceci est tout à fait normal étant donné que nous ne modifions pas la façon de générer ce premier niveau à travers les différentes versions de l'algorithme. À partir du niveau 3, nous pouvons remarquer que la différence entre les deux versions ne relaxant pas le critère de contact entre les polygones (1 et 3) et les versions le relaxant (2 et 4) se marque. Cette différence est en faveur des deux versions relaxant ce critère.

Nous pouvons également noter une légère différence entre les versions 1 et 3 ainsi qu'entre les versions 2 et 4. Il semble donc qu'il ne soit pas très judicieux d'imposer que la somme des poids des deux polygones soit supérieure au seuil lorsque nous relaxons le problème.

Au vu de l'histogramme présenté à la figure 5.22, nous pouvons remarquer que dans tous les cas, la version de l'algorithme n'utilisant que le dernier niveau inclurait des polygones ayant moins souvent une seule tournée. Ceci conduirait certainement à un nombre plus élevé de tournées concernées par une même requête.

5.2 Pistes d'améliorations

5.2.1 Correction de la base de données

L'algorithme n'est pas parfait. Comme nous avons pu nous en rendre compte, il n'atteint pas l'objectif d'un nombre minimal garanti d'adresses par polygone. Pour régler ce problème, il faudrait probablement améliorer la suppression du bruit. Comme nous l'avons déjà évoqué précédemment, lorsqu'un groupe de deux AS se trouve au milieu d'un centre de distribution différent du leur, il n'est pas supprimé. Ceci a pour conséquence qu'une partie des polygones n'atteint pas le seuil requis.

Il n'est pas si simple de supprimer ce bruit. En effet, si à priori la situation décrite paraît facile à repérer, elle n'est pas forcément due à des mauvaises données. La réalité du terrain cause parfois ce type de situation. Il faut donc faire un compromis entre d'un côté respecter les exigences et de l'autre ne pas fausser la réalité du terrain.

Il serait bon d'envisager d'autres techniques afin de repérer ces situations problématiques. Une fois repérés, ces cas doivent être traités. Soit en modifiant la localisation de certaines adresses soit en les supprimant.

Une autre piste consiste à modifier l'algorithme. Comme nous l'avons fait dans les versions 2 et 4, nous pouvons redéfinir les critères de fusion. Ceci peut amener à régler certains problèmes. En effet, si les polygones ne doivent plus être en contact pour être fusionnés, le nombre de cas problématiques d'îlots diminue.

5.2.2 Création d'un graphe

Afin de pouvoir envisager d'autres alternatives d'algorithmes, il serait intéressant de créer un graphe ayant pour nœuds les polygones couvrant les AS. Deux nœuds seraient reliés par une arête lorsque les deux polygones se touchent. En gardant ainsi une liste des voisins de chaque polygone, nous pourrions améliorer l'efficacité de l'algorithme et donc facilement envisager des alternatives.

Le critère principal qui pourrait être amélioré dans ce cas serait l'ordre dans lequel nous tentons de fusionner les polygones. Grâce à un graphe, nous pourrions dans un premier temps regarder uniquement les polygones voisins du polygone à

fusionner. Nous pourrions ensuite par exemple envisager les voisins de ses voisins si aucun d'eux ne respecte les critères imposés pour une fusion.

5.2.3 Choix du niveau initial

Lorsque l'on regarde d'un peu plus près les polygones qui composent les premières couches, on remarque assez vite un effet relativement étrange de prime abord. Cette situation est illustrée sur la figure 5.23.

Le phénomène se produit sur un segment d'activité ayant des adresses situées des deux côtés de cette route.

Pour rappel, chaque polygone sur le premier niveau représente un AS. Chaque polygone devrait donc se trouver d'un côté de la route.

Cependant, si les données nous indiquent les adresses au centre de cette route et non pas précisément du bon côté de la route, nous rencontrons un problème. En effet, lorsque nous créons le diagramme de Voronoï, nous obtenons une multitude de polygones en alternance pour deux AS. Comme nous pouvons l'observer sur la figure 5.23, les polygones rayés sont tous relatifs au même AS, de même, les polygones non rayés sont relatifs à un même AS diffèrent.

FIGURE 5.23 – Situation problématique sur les niveaux inférieurs

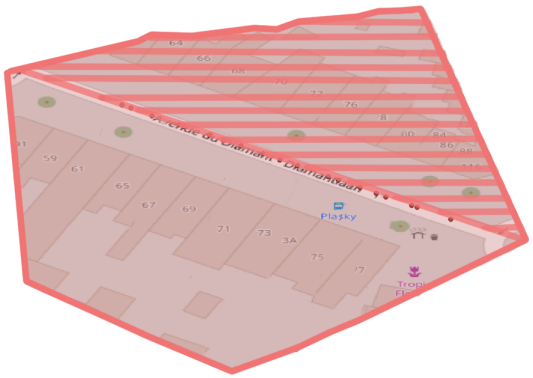

FIGURE 5.24 – Situation souhaitée sur les niveaux inférieurs

Plusieurs solutions pourraient être envisagées.

La première consisterait à prendre non plus les AS comme critère de fusion initial mais les segments de routes postaux (PSS [13]). Ceux-ci reprennent les deux côtés de la route. Ceci aurait pour effet de créer un seul polygone pour notre illustration. L'inconvénient réside dans le fait que les deux côtés d'une route (et donc un PSS) ne font pas toujours partie de la même tournée. Ceci aurait donc comme conséquence d'augmenter fortement le nombre de polygones comportant au moins deux tournées.

Une autre possibilité consiste à modifier les coordonnées géographiques afin de bien distinguer le côté de la route auquel appartient l'adresse. Ceci modifierait considérablement notre exemple, nous aurions deux polygones, un de chaque côté de la route. Cette solution donnerait certainement de bons résultats, mais nécessite un gros travail.

13. Postal Street Segment

Chapitre 6

Conclusion

Proposer à Bpost une solution algorithmique pour la distribution de flyers tout en respectant sa lecture de la réglementation RGPD, tel était l'objet de ce mémoire.

Dans le cadre de son service de distribution de flyers Bpost propose à ses clients de définir une zone répartie autour d'un point central et dans un rayon donné, rassemblant un nombre suffisant d'adresses, en l'occurrence 80, pour ne pas enfreindre la réglementation RGPD. Il s'agissait donc de construire un algorithme respectant cette contrainte en la combinant à des contraintes de précision, de déterminisme, de consistance et de sélection des adresses.

Afin de limiter l'impact des erreurs présentes dans la base de données fournie par Bpost, nous avons fait le choix d'y apporter quelques modifications. La première est de comptabiliser certaines adresses à un endroit nous semblant plus correct lorsque cette adresse était trop éloignée des autres adresses se trouvant sur un même segment d'activité. La seconde est de supprimer les adresses entourées uniquement d'adresses ne se trouvant pas dans le même centre de distribution qu'elles.

Après avoir éliminé les erreurs dans la base de données fournie par Bpost, nous nous sommes inspirés de la structure pyramidale ainsi que du fonctionnement bottom-up introduits dans les travaux de Casper afin de créer un algorithme répondant à notre problème.

L'algorithme ainsi construit a pour point de départ un diagramme de Voronoï. Ce dernier a pour but de couvrir tout le plan de polygones sans laisser de vide. Chaque polygone est généré par un germe qui dans notre cas correspond à une adresse. Afin de constituer le premier niveau nous regroupons tous les polygones appartenant à un même segment d'activité. Ensuite, pour les autres niveaux, nous avons d'abord défini des critères permettant de fusionner petit à petit les polygones initialement créés. Le premier critère consiste à ne pas fusionner deux polygones

n'appartenant pas au même centre de distribution. Le second critère favorise quant à lui la fusion de polygones appartenant à la même tournée. Au fur et à mesure des niveaux, les poids des polygones, représentant le nombre d'adresses, sont de plus en plus grands.

Les niveaux que nous avons créés seront utilisés dans la dernière partie. En effet, dans cette dernière partie nous effectuons la requête du client sur chacun des niveaux en commençant par le niveau le plus bas (ayant les polygones les plus petits) jusqu'à obtenir un dénombrement précis, mais au-delà des 80 adresses imposées par Bpost. Nous avons montré que par construction nous n'avions pas un dénombrement aléatoire. Le dénombrement sera également toujours identique pour un même point et un même rayon.

Cependant, il se pourrait que l'algorithme soit vulnérable à certaines attaques. Nous avons donc décidé de tester également ses performances en utilisant uniquement le dernier niveau.

En ce qui concerne l'anonymat, même si le phénomène reste marginal, nous constatons qu'il reste des polygones ayant un poids sous le seuil fixé.

En définitive, la précision de l'algorithme et le nombre de tournées par requête sont tout à fait satisfaisants. En effet, nous avons pu observer que l'erreur commise lors du dénombrement était légère et qu'elle avait tendance à diminuer lorsque le rayon des requêtes augmente. Nous pouvons donc considérer avoir répondu adéquatement à la demande de Bpost. En réalité, Bpost ne réalise pas de requête en dessous de 1 km. L'erreur pour l'utilisateur (au-delà de 1 km) sera donc minime.

Par ailleurs, les résultats obtenus pour l'algorithme n'utilisant qu'un seul niveau, sont moins bons que ceux de la version utilisant les 5 niveaux, mais permettent tout de même d'envisager cette solution.

Notre travail se termine par la suggestion de quelques points d'amélioration. La création d'un graphe pourrait notamment permettre d'envisager d'autres versions plus performantes de notre algorithme. Nous pourrions également envisager de créer notre premier niveau avec une autre méthode. À titre d'exemple, nous pourrions regrouper les polygones ayant un même PSS[14].

Enfin, nous avons pu remarquer au cours de ce travail que la qualité des données était d'une importance capitale. La suppression des erreurs dans la base de données pourrait donc être poussée plus loin. Étant donné la nature des données présentes dans cette base de données, il serait envisageable de développer un algorithme

14. Postal Street Segment

consacré exclusivement à la détection des erreurs en recourant, par exemple à des techniques d'intelligence artificielle.

Pour conclure, nous pouvons dire que notre algorithme relève un grand nombre de défis lancés par Bpost. S'il ne peut pas garantir à 100% l'anonymat, il a le mérite de pouvoir être utilisé de deux manières différentes. Soit dans sa version la plus précise, en ayant un petit risque d'attaque et donc un critère d'anonymat moins élevé, soit dans une version plus robuste, mais également moins précise.

Bibliographie

[1] Auclair-Semere S., Jacquin M., Lakritz E. et Sanchez A.-M. : Modélisation à l'aide des diagrammes de voronoï : Réseau téléphonique, et autres applications.

[2] Conseil de l'Europe et Parlement européen : Directive 95/46/ce du parlement européen et du conseil, du 24 octobre 1995, relative à la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données. *Journal officiel de l'Union européenne*, (201), 1995.

[3] Conseil de l'Europe et Parlement européen : Règlement (ue) 2016/679 du parlement européen et du conseil du 27 avril 2016 relatif à la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données, et abrogeant la directive 95/46/ce (règlement général sur la protection des données). *Journal officiel de l'Union européenne*, (119), 2016.

[4] Gkoulalas-Divanis A., Kalnis P. et Verykios V.S. : Providing k-anonymity in location based services. *ACM SIGKDD explorations newsletter*, 12(1):3–10, 2010.

[5] Kalnis P., Ghinita G., Mouratidis K. et Papadias D. : Preventing location-based identity inference in anonymous spatial queries. *IEEE transactions on knowledge and data engineering*, 19(12):1719–1733, 2007.

[6] Mokbel M.F., Chow C.-Y. et Aref W.G. : The new casper : Query processing for location services without compromising privacy. *In Proceedings of the 32nd international conference on Very large data bases*, pages 763–774. VLDB Endowment, 2006.

[7] Pilaud V. : Sur le diagramme de voronoï et la triangulation de delaunay d'un ensemble de points dans un revêtement du plan euclidien. 2006.

[8] Wang K., Chen R., Fung BC. et Yu PS. : Privacy-preserving data publishing : A survey on recent developments. *ACM Computing Surveys*, 2010.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**