UVSQ

Université de Versailles Saint-Quentin-en-Yvelines

A Microservices-Based Architecture for Implementing and Automating Analytics Data Pipelines for Mobile Crowdsensing Applications

Projet conception

Élève :
TRILLO Baptiste
TOPILKO Yann

Professeur :
Yehia TAHER
Karine Zeitouni

Projet réalisé pour le 24 mars 2025

Table des matières

1 Introduction

Le projet **MobiSpace** s'inscrit dans le cadre du traitement et de l'analyse de données issues de la collecte mobile participative (Mobile Crowd Sensing). Ce projet repose sur une architecture orientée microservices permettant la mise en place de pipelines de données analytiques robustes et automatisés. Il est conçu pour gérer de bout en bout le cycle de vie des données environnementales récoltées via des capteurs portables connectés à des appareils mobiles.
Le cadre d'application principal de MobiSpace est le suivi de la qualité de l'air dans le cadre de projets comme *Polluscope*, où des participants volontaires recueillent des mesures telles que les particules fines, le NO_2, le carbone noir, la température et l'humidité. Ces données sont ensuite enrichies, segmentées, analysées, stockées et visualisées via une chaîne de traitement modulaire.
Le projet est divisé en plusieurs groupes, chacun chargé d'un aspect particulier du pipeline de traitement :

- Groupe 1 : Ingestion et prétraitement des données ;
- Groupe 2 : Segmentation et enrichissement des trajectoires ;
- Groupe 3 : Visualisation des données avec Grafana ;
- **Groupe 4 : Extraction de données anonymisées en environnement extérieur**.

Dans le cadre du **Groupe 4**, notre objectif est de garantir la préservation de la vie privée des participants, en supprimant ou en masquant les portions sensibles des trajectoires GPS — typiquement les lieux de vie ou de travail. C'est dans ce contexte que nous utilisons le projet **my-home-is-my-secret**, un outil développé pour anonymiser des trajectoires de mobilité. Ce projet repose sur des techniques de clustering spatial et de troncature géométrique pour identifier et exclure les points jugés sensibles, comme les zones d'habitation.

2 Étude bibliographique

2.1 My Home is My Secret

L'article "My Home is My Secret" aborde le problème de la protection des données de trajectoires issues des systèmes de positionnement GNSS. Les trajectoires humaines peuvent révéler des informations sensibles, comme les domiciles ou les lieux de travail. Pour résoudre ce problème, les auteurs proposent un algorithme innovant, le Site-dependent Trajectory Truncation (S-TT), qui préserve la confidentialité en supprimant stratégiquement des segments de trajectoires.

2.2 Problématique

Les données de trajectoires, collectées par les applications mobiles et les services basés sur la localisation, sont souvent exploitées à des fins d'analyse urbaine, de planification de transport ou de ciblage publicitaire. Cependant, ces données peuvent être réidentifiées et permettre de localiser des lieux sensibles, comme des domiciles ou des établissements médicaux.

2.2.1 Algorithmes d'anonymisation

Plusieurs approches d'anonymisation sont détaillées :

- La `k-anonymité` : concept de protection de la vie privée initialement pour les bases de données relationnelles. Chaque trajectoire doit être indistinguable d'au moins k1 autres trajectoires par rapport à un ensemble d'attributs appelés quasi-identifiants. Cela signifie qu'un attaquant ne peut pas identifier une trajectoire spécifique parmi un groupe d'au moins k trajectoires.
- La `confidentialité différentielle` : un concept garantissant que l'ajout ou la suppression d'un seul élément dans une base de données ne modifie pas significativement

les résultats d'une requête, ce qui empêche les attaquants d'exploiter le mécanisme pour obtenir des informations sur un individu.
— Des mécanismes atténuants risques de violation de la vie privée. Il existe des techniques d'offuscation traditionnelles qui déplacent les points de trajectoire individuellement, la segmentation des trajectoires où chaque segment est associé à un identifiant différent ou où les identifiants sont échangés entre trajectoires et le cloaking faisant la réduction de la granularité des trajectoires dans l'espace ou le temps.
— Des mécanismes se concentrant uniquement sur la protection de lieux sensibles, laissant le reste de la trajectoire intact. Il existe des techniques de substitution qui remplace les lieux sensibles par des lieux proches et sémantiquement similaires, du cloaking qui remplace les points de trajectoire proches des lieux sensibles par des zones généralisées contenant plusieurs lieux, la suppression des points de trajectoire proches des lieux sensibles et la distorsion temporelle masquant les points de séjour en recréant les trajectoires avec une vitesse constante, ce qui supprime les arrêts.

Les LPPMs basés sur la k-anonymité et la confidentialité différentielle offrent des garanties formelles de confidentialité, mais peuvent réduire l'utilité des données. Les mécanismes qui atténuent les risques ou protègent spécifiquement les lieux sensibles sont souvent plus simples et préservent mieux l'utilité des données, mais ne fournissent pas toujours des garanties de confidentialité aussi solides.
L'article souligne également qu'il existe un manque dans la littérature concernant les mécanismes capables de protéger des lieux qui ne sont pas universellement sensibles, mais qui le sont pour certains individus, comme leur domicile ou leur lieu de travail. Cela motive l'introduction du mécanisme proposé dans l'article, qui est conçu pour protéger ces lieux sensibles de manière contextuelle et en préservant l'utilité des données.

2.2.2 Modèle d'Attaque et Modèle de Confidentialité

Nous allons introduire un modèle d'attaquant et un modèle de confidentialité. Enfin, nous formulons le problème que l'algorithme proposé (S-TT) vise à résoudre.
L'attaquant cherche à identifier le site de destination $\tilde{s}$ d'un ensemble de trajectoires $\Theta = \{T_1, \ldots, T_l\}$. On suppose que l'attaquant connaît un ensemble de sites S contenant $\tilde{s}$ et utilise des fonctions d'attaque $f_1, \ldots, f_a$. Chaque fonction f prend une trajectoire T et retourne un ensemble de sites candidats $f(S, T) \subseteq S$. Par exemple, une fonction pourrait retourner les sites les plus proches du point final de T. L'attaquant utilise ces ensembles pour réduire S et deviner $\tilde{s}$.
Pour limiter la capacité de l'attaquant à identifier $\tilde{s}$, l'article introduit un concept de confidentialité appelé **k-site-unidentifiability** (k-site-indifférenciabilité), qui est une adaptation du principe de **k-anonymité** pour les sites. La **k-site-unidentifiability** est garantie si la destination $\tilde{s}$ est cachée parmi au moins $k-1$ autres sites dont les enregistrements partagent les mêmes valeurs de quasi-identifiants. Formellement, D_S satisfait la **k-site-unidentifiability** par rapport à Θ et un quasi-identifiant Q s'il existe un ensemble d'enregistrements $R \subseteq (D_S \setminus \{\tilde{r}\})$ tel que :

1. $|R| \geq k-1$ (il y a au moins $k-1$ enregistrements dans R),
2. Pour tout attribut $a \in Q$ et pour tout enregistrement $r \in R$, $\tilde{r}(a) = r(a)$ (les enregistrements partagent les mêmes valeurs de quasi-identifiants).

L'ensemble des sites correspondant à ces enregistrements est appelé **ensemble de protection** de $\tilde{s}$ et est noté $C(\tilde{s})$.
Le problème consiste à transformer l'ensemble de trajectoires Θ en un nouvel ensemble Θ' qui préserve la **k-site-unidentifiability** sous un ensemble d'attaques $F = \{f_1, \ldots, f_a\}$. La condition à satisfaire est :

$$\forall T \in \Theta, \forall f \in F : f(S,T) \cap C(\tilde{s}) \in \{\emptyset, C(\tilde{s})\}.$$

Cela signifie que pour chaque trajectoire T et chaque fonction d'attaque f, soit aucun site de $C(\tilde{s})$ n'est retourné, soit tous les sites de $C(\tilde{s})$ sont retournés. Ainsi, $\tilde{s}$ reste caché parmi k sites.

2.3 Algorithme S-TT (Site-Dependent Trajectory Truncation)

L'algorithme S-TT (Site-dependent Trajectory Truncation) a pour objectif de protéger les lieux sensibles dans les trajectoires en supprimant les points de trajectoire qui pourraient révéler la destination. L'idée principale est de tronquer les trajectoires de manière à ce que la destination ne puisse pas être identifiées parmi un ensemble de sites candidats.
L'algorithme prend en entrée :

- Un ensemble de trajectoires Θ ayant une destination commune $\tilde{s}$.
- Un ensemble de sites S contenant $\tilde{s}$.
- Un ensemble de protection $C(\tilde{s}) \subseteq S$ contenant au moins k sites, dont $\tilde{s}$.
- Un ensemble de fonctions d'attaque $F = \{f_1, \ldots, f_a\}$.

L'algorithme produit en sortie un ensemble de trajectoires tronquées Θ', où chaque trajectoire $T \in \Theta$ est remplacée par une version tronquée T'. La troncation est effectuée de manière itérative en supprimant le point final de la trajectoire jusqu'à ce que pour chaque fonction d'attaque f, soit aucun site de l'ensemble de protection $C(\tilde{s})$ n'est retourné, soit tous les sites de $C(\tilde{s})$ sont retournés. Ainsi, les sites dans $C(\tilde{s})$ sont indistinguables par rapport à f.
L'article propose une version concrète de l'algorithme S-TT qui se concentre sur deux types d'attaques géométriques courantes :

- **Attaque de proximité** : L'attaquant suppose que la destination $\tilde{s}$ est le site le plus proche du point final de la trajectoire.
- **Attaque de direction** : L'attaquant utilise la direction du dernier segment de la trajectoire pour deviner la destination.

Pour contrer ces attaques, l'algorithme utilise deux fonctions d'attaque spécifiques :

- **Fonction de proximité** f_p : retourne le site le plus proche du point final de la trajectoire.
- **Fonction de direction** f_d : Retourne sur les sites situés dans une région en forme de V alignée avec le dernier segment de la trajectoire.

La condition de troncation devient alors :

$$\forall f \in \{f_p, f_d\} : f(S,T') \cap C(\tilde{s}) \in \{\emptyset, C(\tilde{s})\}.$$

Cela signifie que la trajectoire est tronquée jusqu'à ce que la région en forme de V ne contienne soit aucun site de $C(\tilde{s})$, soit tous les sites de $C(\tilde{s})$.

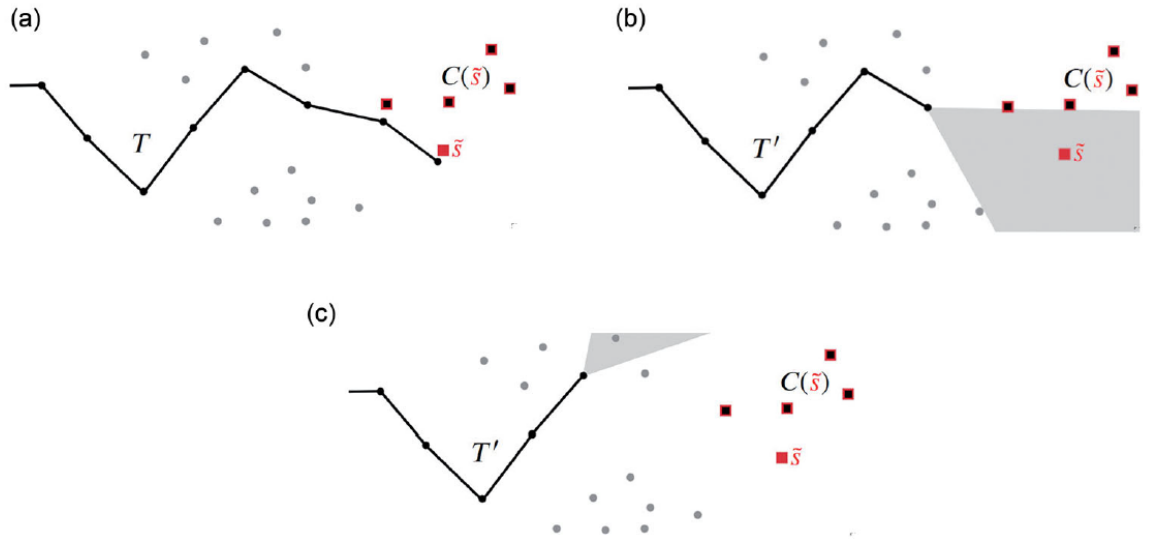

FIGURE 1 – Schéma de l'algorithme STT

3 Nettoyage des données et détection des stops & moves

Le processus débute par une étape de *nettoyage*, dont l'objectif est d'exclure tous les points GPS jugés aberrants afin de conserver uniquement des trajectoires plausibles. Pour cela, on utilise un algorithme de détection d'anomalies (tel que `OutlierCleaner`), qui compare la position et la vitesse instantanée de chaque point à des seuils statistiques définis par un paramètre (`alpha`). Les points identifiés comme trop éloignés du comportement général d'une trajectoire sont alors supprimés. Après cette suppression, on recalcule certaines informations essentielles (notamment la vitesse) pour refléter correctement l'état de la trajectoire nettoyée. On peut ensuite écarter les trajectoires devenues trop courtes (si elles ne possèdent plus un nombre suffisant de points). Les données ainsi nettoyées sont enfin exportées au format `CSV` ou stockées sous forme de `GeoDataFrame`, prêtes à être visualisées ou analysées.
Une fois ce nettoyage achevé, on se concentre sur la distinction entre les périodes d'arrêt (`stops`) et de déplacement (`moves`). On commence par créer une collection de trajectoires nettoyées, sur laquelle on applique un détecteur d'arrêts (`TrajectoryStopDetector`) pour isoler les segments où le déplacement est négligeable durant un certain laps de temps (par exemple, plus de sept minutes dans un rayon de cinquante mètres). Les fragments répondant à ces critères sont identifiés comme `stops`, tandis que le reste de la trajectoire est considéré comme `moves`. Dans certains cas, on sépare davantage la trajectoire à l'aide d'un `StopSplitter`, qui segmente précisément les arrêts et les déplacements en s'appuyant sur la durée et le rayon de mouvement autorisé. L'aboutissement de ce processus est la production de deux ensembles : un premier correspondant aux arrêts (généralement représentés comme des points *centroïdes*), et un second correspondant aux trajets actifs (lignes). Ces deux ensembles peuvent être exportés dans un format `GeoDataFrame` (ou `Shapefile`) afin de faciliter les analyses spatiales ultérieures.

4 Algorithme de privacy

4.1 Analyse de code

4.1.1 GeometricClustering

Le programme `GeometricClustering` a pour objectif de réaliser un regroupement spatial (clustering) de points géographiques, à partir d'un fichier shapefile contenant des centroïdes. Il commence par lire les points géographiques et les transforme en nœuds d'un graphe, chacun identifié par ses coordonnées. Ces nœuds sont stockés dans une structure ordonnée.
Ensuite, une triangulation de Delaunay est effectuée sur l'ensemble des points. Cette triangulation permet de créer automatiquement des arêtes entre les points proches. À partir de cette structure, les cellules de Voronoï sont également calculées pour chaque nœud. Chaque cellule de Voronoï représente la zone d'influence d'un point.
Une fois le graphe construit, avec ses nœuds et ses arêtes (pondérées par la distance entre les points), un algorithme de clustering est appliqué pour regrouper les points en k clusters, où k est un paramètre fourni par l'utilisateur (par défaut à 4).
Le programme affiche le tout dans une interface graphique avec deux couches : une pour les points, et une pour les arêtes de la triangulation. Enfin, plusieurs résultats sont exportés dans des fichiers shapefile :

- `multipoints.shp` : les clusters représentés comme multipoint,
- `clusteredges.shp` : les arêtes internes à chaque cluster,
- `graphedges.shp` : toutes les arêtes du graphe,
- `hulls.shp` : les enveloppes convexes de chaque cluster,
- `cells.shp` : les cellules de Voronoï associées à chaque nœud.

Ces fichiers servent notamment de base pour le traitement dans d'autres programmes, comme `TrajectoryTruncation`.

4.1.2 TrajectoryTruncation

Le programme `TrajectoryTruncation` a pour but de nettoyer des trajectoires GPS en supprimant leurs extrémités (début et fin), susceptibles de révéler des informations personnelles comme l'adresse du domicile. Pour cela, il exploite les résultats du programme précédent, à savoir les cellules de Voronoï (`cells.shp`) et les clusters (`multipoints.shp`), ainsi que les trajectoires GPS à traiter (`synthetic_trajectories_hel.shp`).
Les cellules de Voronoï sont indexées spatialement grâce à une structure `STRtree`, ce qui permet de déterminer rapidement dans quelle région se situe un point donné. Pour chaque trajectoire, le programme identifie les cellules correspondant au point de départ et au point d'arrivée. Ces zones sont considérées comme des zones « sensibles ».
Le programme cherche ensuite à déterminer les premiers et derniers points « sûrs » à conserver dans la trajectoire. Pour cela, il parcourt les points dans l'ordre (pour le début) et à l'envers (pour la fin), en utilisant une méthode géométrique : à chaque point potentiel, un triangle orienté est construit à partir de la direction du déplacement. Si ce triangle contient quelques points du cluster d'origine (mais pas tous), le point est considéré comme encore situé dans une zone sensible, et donc exclu.
Dès qu'un point ne répond plus à ces conditions, il est conservé et marque le début ou la fin de la trajectoire nettoyée. La partie centrale de la trajectoire, située entre ces deux points, est alors conservée.
Enfin, le programme exporte les résultats :

- `truncated.shp` : les trajectoires nettoyées,
- `triangles.shp` : les triangles utilisés pour la détection des zones sensibles, accompagnés de leur direction et d'un indicateur signalant s'il s'agit du début ou de la fin.

Ce programme est un bon exemple d'utilisation combinée de techniques géométriques et d'analyses de graphes pour la préservation de la vie privée dans les données de mobilité.

4.2 Tests du code

Pour utiliser ce code, nous avons compris qu'il nécessitait des fichiers au format *Shapefile*, un format de données géospatiales largement utilisé dans les systèmes d'information géographique (SIG). Un Shapefile est en réalité composé de plusieurs fichiers complémentaires, dont les trois principaux sont `.shp` qui contient la géométrie des objets (points, lignes ou polygones), `.shx` qui contient un index spatial des objets pour un accès rapide et `.dbf` qui est une vase de données associée contenant des attributs (informations descriptives) liés à chaque objet géographique.
Nous avons d'abord testé le code avec les fichiers shapefile fournis dans le projet **my-home-is-my-secret**, avant de l'adapter aux données de trajectoires collectées dans le cadre du projet **Mobispace**.
Concernant les données de Finlande, elles ont été extraites depuis le site *Geolife*, utilisé dans la recherche pour analyser les déplacements humains (à pied, en voiture, en transport en commun, etc.). Ce jeu de données est directement fourni sous forme de Shapefile, avec une entête contenant les champs `osm_id, code, fclass, name, type, geometry`. Toutefois, l'encodage initial du fichier posait des problèmes lors de la lecture dans le code Java ; il a donc fallu le nettoyer pour supprimer les caractères spéciaux.
Pour les données de Versailles, l'extraction a été réalisée à partir d'une API référençant les géopoints des bâtiments. Ces points ont ensuite été convertis en fichier Shapefile afin de les rendre compatibles avec le code du projet **my-home-is-my-secret**.
Après nettoyage et conversion, nous avons pu utiliser la première partie du code, **GeometricClustering**, qui permet de construire des clusters à partir des points centraux des bâtiments. Ces clusters serviront ensuite à sécuriser les trajectoires personnelles. Voici les résultats obtenus :

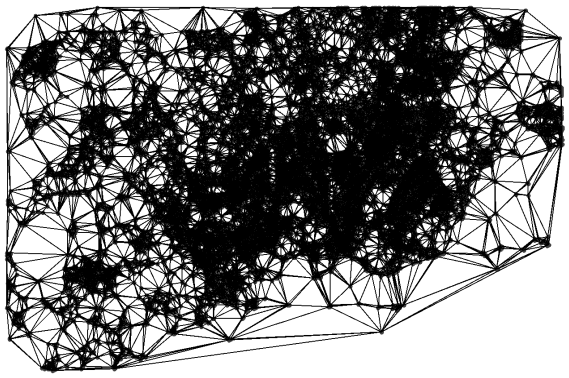

FIGURE 2 – Résultat du code Java **GeometricClustering** pour les données de Finlande

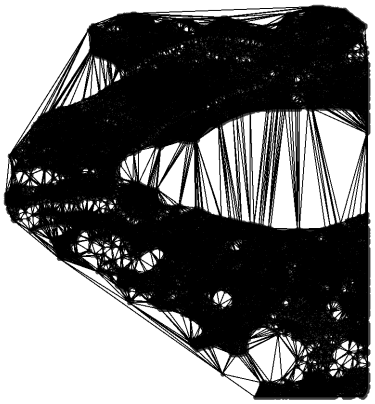

FIGURE 3 – Résultat du code Java **GeometricClustering** pour les données de Versailles

Nous avons ensuite utilisé la seconde partie du projet, **TrajectoryTruncation**, qui exploite les résultats produits par **GeometricClustering** (notamment les fichiers `cells.shp` et `multipoints.shp`) ainsi que les données de trajectoires GPS pour anonymiser les trajets.
Dans le cas de la Finlande, les trajectoires étaient déjà disponibles sous format Shapefile, ce qui a permis une intégration directe. En revanche, pour les données des participants à Versailles, il a fallu appliquer un algorithme de détection des phases de déplacement et d'arrêt (*stop and*

move detection) pour reconstituer les trajectoires, avant de les convertir à leur tour en format Shapefile.
Au final, nous obtenons un nouveau fichier shapefile contenant les trajectoires nettoyées : certaines ont été partiellement tronquées pour anonymisation, tandis que d'autres ont été totalement supprimées lorsqu'elles restaient intégralement dans des zones sensibles.

4.3 Analyse sur des données

Après toutes les modifications des données et de l'application des algorithmes, nous pouvons désormais observer comment les trajectoires ont été modifiées en fonction des bâtiments fournis. Nous allons chercher à déterminer le nombre de trajectoires supprimées car elles ne respectaient pas la condition d'arrêt décrite à la figure 1. Nous allons également calculer le nombre total de points supprimés, ainsi que le nombre moyen de points supprimés par trajectoire.
Pour cela, il est nécessaire de charger à la fois les trajectoires originales (avant la troncature) et les trajectoires après traitement. Nous appliquons ensuite des opérations de transformation et de comparaison sur ces données. L'idée est de retrouver la trajectoire d'origine correspondant à chaque trajectoire tronquée, en partant du principe que toute trajectoire tronquée est nécessairement contenue dans l'une des trajectoires originales. Cela permet d'établir une correspondance et de produire les statistiques souhaitées.

4.3.1 Finlande

Pour les données de Finlande, fournies par les doctorants du projet **my-home-is-my-secret**, nous avons d'abord nettoyé les fichiers afin d'assurer leur compatibilité avec le code Java.
Nous disposons d'un total initial de 10 927 trajectoires. Après l'application de l'algorithme d'anonymisation, ce nombre passe à 10 921, ce qui signifie que seulement 6 trajectoires ont été supprimées.
Le nombre moyen de points par trajectoire est de 2 770.87 avant troncature, contre 2 696.56 après-troncatures, soit un total de 755 549 points supprimés. Cela correspond à un taux de réduction moyen de 4.26 %.

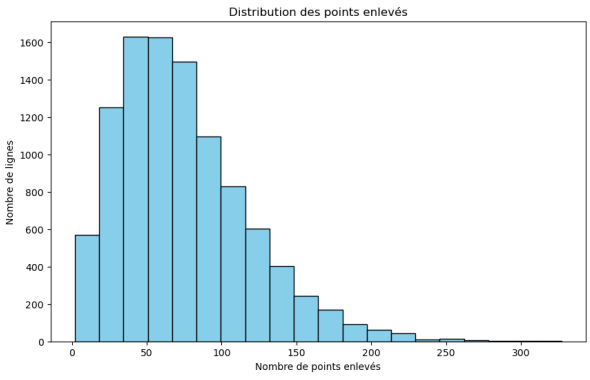

FIGURE 4 – Distribution du nombre de points enlevés par trajectoire pour les données de la Finlande

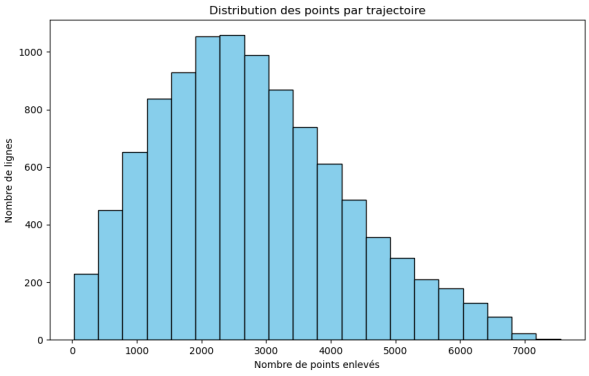

FIGURE 5 – Distribution du nombre de points par trajectoire pour les données de la Finlande

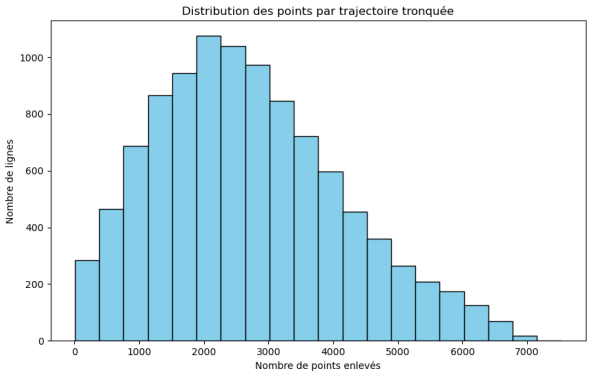

FIGURE 6 – Distribution du nombre de points par trajectoire tronquée pour les données de la Finlande

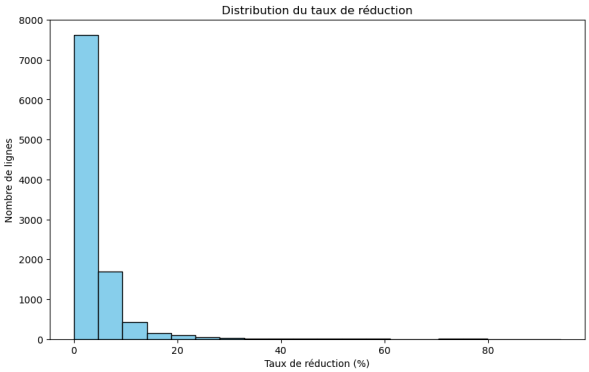

FIGURE 7 – Distribution du taux de réduction par trajectoire pour les données de la Finlande

4.3.2 Versailles

Pour les données de Versailles, les trajectoires ont été extraites à l'aide de l'algorithme **stop and move**, que nous avons adapté afin de produire un format compatible avec le projet **my-home-is-my-secret**.
Nous avons ainsi obtenu 137 trajectoires provenant du participant `Participant9999915`, qui ont été réduites à seulement 23 après-troncatures : 114 trajectoires ont donc été totalement supprimées.
Le nombre moyen de points par trajectoire était de 100 avant traitement, contre 49.65 après-troncatures. Cela représente un total de 1 007 points supprimés, pour un taux de réduction moyen de 50.35 %.

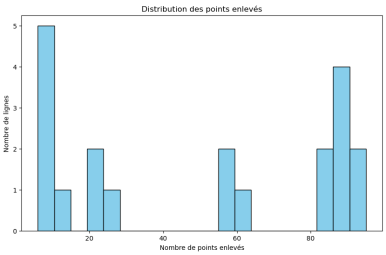

FIGURE 8 – Distribution du nombre de points enlevés par trajectoire pour les données de Versailles

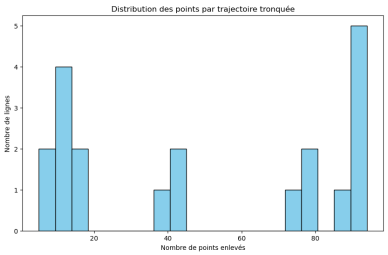

FIGURE 9 – Distribution du nombre de points par trajectoire pour les données de Versailles

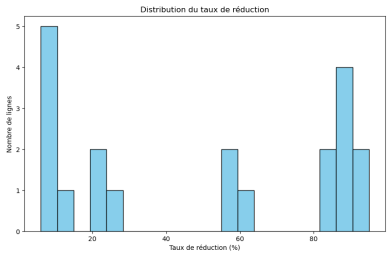

FIGURE 10 – Distribution du taux de réduction par trajectoire pour les données de Versailles

4.4 Problèmes rencontrés

L'objectif initial était d'utiliser le code **my-home-is-my-secret** sans avoir d'informations détaillées sur son fonctionnement interne ni sur les fichiers nécessaires à son exécution. Heureusement, des fichiers de test étaient fournis, permettant d'observer à quoi ressemblaient les fichiers d'entrée attendus ainsi que les formats de sortie générés.
Cependant, lors de l'utilisation de ces fichiers fournis, aucun traitement ne fonctionnait : des messages d'erreur indiquaient que les fichiers ne pouvaient pas être lus correctement. Il a donc été nécessaire d'analyser les fichiers de test pour identifier l'origine des dysfonctionnements.
Ne maîtrisant pas encore le format *Shapefile*, nous avons entrepris d'explorer leur contenu afin de comprendre ce qui pouvait poser problème. Pour cela, nous avons utilisé la bibliothèque

`GeoPandas` en Python, qui permet de lire et de manipuler facilement des fichiers géographiques. En analysant les données, nous avons constaté que l'encodage des fichiers était inhabituel, ce qui provoquait l'échec de la lecture par le code Java.
Nous avons alors tenté de corriger l'encodage en passant par **PostGIS**, une extension spatiale de **PostgreSQL** permettant de gérer des données géographiques (points, lignes, polygones, etc.) et d'exécuter des requêtes spatiales. Cette solution aurait pu permettre de charger les fichiers `.shp` et de modifier leur encodage directement en base de données. Toutefois, cela ne fonctionnait pas comme espéré.
Nous nous sommes donc recentrés sur Python avec `GeoPandas`, en testant plusieurs transformations d'encodage, y compris en convertissant les fichiers au format `.csv` pour les régénérer ensuite en `.shp`. Malgré ces essais, les erreurs persistaient.
Finalement, la solution la plus efficace a été la plus simple : supprimer tous les caractères problématiques (accents, caractères chinois, japonais, etc.) présents dans les champs attributaires. Bien que cela réduise légèrement l'information contenue (par exemple, les noms exacts des bâtiments), cela n'avait pas d'impact sur les coordonnées spatiales qui sont les données principales utilisées par le code.
Après ce nettoyage, le programme **my-home-is-my-secret** a pu exécuter correctement les étapes de clustering et de troncature des trajectoires. Ainsi, une solution aussi simple que la suppression de tous les caractères non-numériques ou non-géographiques s'est révélée suffisante pour faire fonctionner l'outil.

4.5 Synthèse des expérimentations avec le code d'anonymisation

L'utilisation du code **my-home-is-my-secret** a permis de constater que les données issues du jeu de données *Geolife* sont bien adaptées à son fonctionnement. En effet, une anonymisation efficace est réalisée sur un grand nombre de trajectoires, avec peu de points supprimés et très peu de trajectoires entièrement éliminées.
À l'inverse, lorsque l'on applique le même traitement à des trajectoires provenant d'autres sources (comme celles collectées à Versailles), les résultats sont beaucoup moins satisfaisants. Le nombre de trajectoires conservées diminue fortement, et le nombre de points par trajectoire est également considérablement réduit.
Plusieurs facteurs peuvent expliquer cette différence :

- Les trajectoires utilisées sont peut-être trop courtes pour permettre une anonymisation efficace ;
- Les points fournis en entrée de l'algorithme **GeometricClustering** ne sont pas sélectionnés manuellement, ce qui conduit à un surpeuplement de points et donc à une suppression excessive des points des trajectoires ;
- Les trajectoires ne sont tout simplement pas adaptées au fonctionnement interne du code **my-home-is-my-secret**, qui a été initialement conçu pour un format ou une structure spécifique (comme celles de Geolife).

Ces constats ouvrent la voie à plusieurs pistes d'amélioration :

- Utiliser des trajectoires plus longues, avec une densité de points suffisante, pour offrir plus de marge à l'algorithme de troncature ;
- Sélectionner manuellement les points représentatifs des bâtiments à fournir à l'entrée de l'algorithme **GeometricClustering**, afin d'obtenir un clustering plus pertinent et donc une anonymisation mieux ciblée.

5 Conclusion

Au terme de ce travail, nous avons mis en œuvre et testé un ensemble de méthodes pour la collecte, la préparation et l'anonymisation de données de trajectoires. La phase de *nettoyage* a d'abord permis de supprimer les points aberrants et d'obtenir des trajectoires fiables, lesquelles ont ensuite été segmentées en périodes d'arrêt (`stops`) et de déplacement (`moves`). Sur ces données nettoyées et segmentées, nous avons appliqué l'approche **my-home-is-my-secret**, qui intègre un mécanisme de troncature des trajectoires afin de masquer les lieux sensibles. Les résultats obtenus montrent que, pour des données comme celles de *Geolife*, le processus d'anonymisation s'avère relativement efficace et peu intrusif sur la structure des trajectoires. En revanche, avec d'autres sources de données (ex. Versailles), nous avons observé une réduction significative du nombre de trajectoires et de points conservés, suggérant un ajustement plus précis des paramètres du clustering et de la troncature. De manière générale, l'ensemble de la chaîne de traitement, du nettoyage à l'anonymisation, constitue une solution modulaire pour la protection de la vie privée dans les analyses de mobilités. Des perspectives d'amélioration incluent une meilleure sélection des points pour le `GeometricClustering` et l'examen d'algorithmes supplémentaires pour ajuster finement le compromis entre anonymisation et utilité des données.