L3 MIASHS
Projet Graphe 2017-2018

Objectif général

Dans le cadre de l'initiative Open Data, la Communauté Urbaine de Toulouse Métropole rend disponible un catalogue de données ouvertes [1]. Parmi les jeux des données disponibles, vous trouverez les données sur les stations vélo en libre service [2]. L'objectif est de proposer un algorithme pour optimiser le rechargement des stations vélo. Cet algorithme doit calculer le plus court chemin passant pour toutes les stations qui doivent être rechargées.

Partie 1 : modélisation

La première partie du projet consiste à modéliser le scénario sous forme de graphe. Pour cela, vous allez vous appuyer sur deux sources de données :

— données statiques : fournissent des informations comme les coordonnées GPS (latitude et longitude) des 281 stations (Figure 1), leur adresse, la présence d'un terminal de paiement, etc. Ces données sont disponibles en format csv [3] ;

— données dynamiques : indiquent l'état d'une station, le nombre de vélos disponibles, le nombre d'emplacements libres, etc. Ces données sont rafraîchies régulièrement et ne sont accessibles que depuis une API.

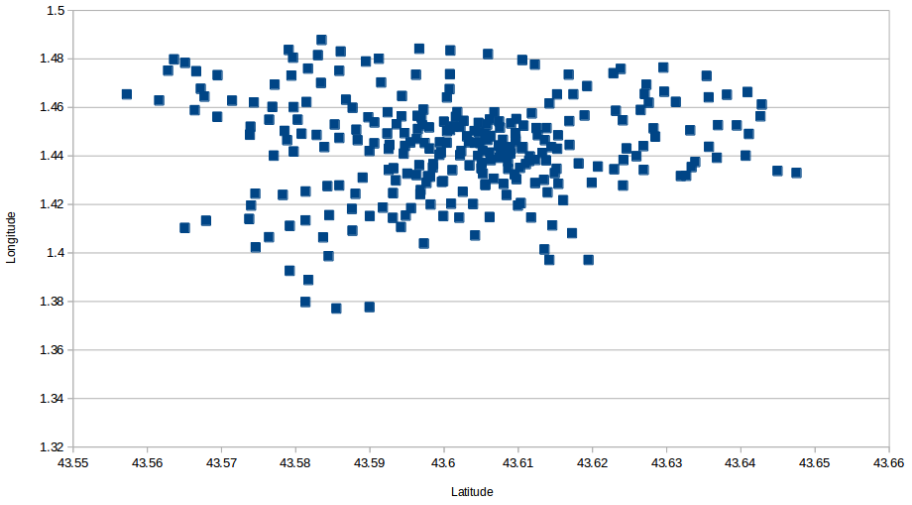

FIGURE 1 – Latitude et longitude des 281 stations vélo de la métropole de Toulouse.

Votre application doit être capable de restituer un graphe connexe à partir de ces deux sources de données. Deux solutions sont envisageables :

1. votre graphe contiendra toutes les stations du réseau et l'algorithme de parcours ne prendra en compte que les stations à recharger (une station vélo doit être rechargée si le nombre de vélos disponibles est inférieur à un seuil donné) ;

1. https://data.toulouse-metropole.fr/
2. https://developer.jcdecaux.com/#/opendata/vls?page=getstarted
3. https://developer.jcdecaux.com/#/opendata/vls?page=static

2. votre graphe ne contiendra que les stations à recharger.

Vous devrez définir une stratégie pour relier les sommets de votre graphe (c'est à dire, pour déterminer si deux sommets sont adjacents). Ne vous basez pas sur l'hypothèse d'un graphe complet. Les poids des arrêtes (ou arcs) de votre graphe correspondent à la distance entre les deux sommets. Pour deux sommets adjacents, vous devrez représenter (toutes) les directions possibles entre eux. Pour cela, vous pouvez utiliser le *Google Directions API*[4] (plus d'informations dans la section suivante).

Dans un premier temps, vous pouvez vous concentrer sur un sous-ensemble des 281 stations (par exemple, en choisissant les n stations autour d'une coordonnée GPS donnée, où n >20).

Partie 2 : développement

Vous devrez développer un algorithme qui prend en entrée un graphe représentant le réseaux des stations vélo et qui retourne le plus court chemin passant par tous les sommets représentant les stations qui doivent être rechargées :

- votre application doit être capable de construire un graphe à partir d'un ou plusieurs fichiers d'entrée (format csv, JSON, ou autre) ;
- choisissez une façon appropriée pour représenter votre graphe (matrice d'adjacence, liste d'adjacence, ou matrice d'incidence sommets-arcs) ;
- identifiez le problème parmi les problèmes connus de la théorie de graphes et implémentez l'algorithme correspondant (vous êtes libres de proposer une adaptation pour l'algorithme en question) ;
- votre application doit renvoyer le plus court chemin sous forme d'une liste de sommets à parcourir (avec la distance entre chaque pair de sommets). Le développement de toute interface graphique (par exemple, l'affichage d'un plan à la Google à partir d'un fichier généré par l'application) sera considéré comme un bonus ;
- vous n'avez pas besoin de prendre en compte les possibles changements, en temps réel, du nombre de vélos disponibles dans une station, au cours de l'exécution de l'algorithme ;

Requêtes HTTP et réponses JSON

Afin de pouvoir calculer (toutes) les directions possibles entre deux coordonnées données, des requêtes HTTP doivent être envoyées sur le service Google Directions API. Une réponse en format JSON est renvoyée. Voici un exemple de requête (notez les paramètres `origin`, `destination` et `alternatives`) : `https://maps.googleapis.com/maps/api/directions/json?origin=43.604134687,1.4454207807&destination=43.6048526522,1.4452858767&alternatives=true&key=AIzaSyAG3bhdapKXj9TpHlic9DgluyQ0Be_Hw5A`. Afin de pouvoir utiliser ce service, vous devrez récupérer une clé d'autorisation. Vous êtes, cependant, libre de choisir une autre façon de calculer les distances entre deux coordonnées données.

A rendre

- un document PDF détaillant vos choix de modélisation et développement
- le code de votre application

4. `https://developers.google.com/maps/documentation/directions/`

Le date limite pour le dépôt du projet sera fixée pour la période 4. Une démonstration sera également faite durant la période 4.