Résumé des méthodes et chemins employés

3 juillet 2022

Le projet que nous avions à réaliser consistait en la conception d'une carte des flux logistiques dans Paris dans le cadre d'un nouveau type de système de livraison, plus décentralisé, inspiré des concepts de l'Internet Physique.

Dans notre démarche, nous avons tout d'abord cherché à la meilleure manière de représenter les flux de véhicules dans les rues de Paris. Nous avons rapidement trouvé la bibliothèque Folium, permettant de manipuler des cartes sous Python. Un outil disponible était alors la fonction heatmap, qui permet comme son nom l'indique de réaliser des cartes de chaleur des flux, en plaçant des points ayant chacun une intensité, et ainsi avoir un affichage assez dynamique. Cependant, nous nous sommes vites rendus compte que cet outil présentait de nombreux défauts. Il avait tendance à demander de très longs temps de calcul dès que l'on plaçait de nombreux points (ce qui allait être le cas avec notre projet), ce qui entrainait une déplacement extrêmement fastidieux sur la carte, le temps que tout charge à chaque fois. De plus, il avait tendance à beaucoup gommer et uniformiser les différences d'intensité de flux, ce qui ne permettait pas bien de se rendre compte des zones encombrées ou non.

Notre idée a alors été de coder par nous même une heatmap, sous la forme d'une grande grille composée de petits carrés, que l'on coloriait à notre convenance en fonction du flux présent. Chacun des points de flux que l'on plaçait sur un carré particulier allait alors "rayonner" sur les carrés d'à côté.. Mais cette idée s'est aussi révélée assez insuffisante, car avait aussi tendance à effacer les différences de flux entre des routes, pouvait parfois entrainer de très longs temps de calcul (il fallait en effet parcourir l'entièreté de la grille plusieurs fois. . .). Ces deux premières idées nous auront globalement occupé les mois d'avril et de mi mai.

La troisième étape, qui a cette fois été fructueuse, a été de représenter les flux simplement par des traits de couleur entre des points (fonction polyline sur folium). En réfléchissant en amont à la meilleure méthode de placer ces traits,

nous avons décidé de nous lancer dans la méthode dite des 'noeuds'. Il s'agit de placer un noeud à chaque intersection entre des routes dans Paris, et relier tous les noeuds adjacents par des segments polyline, sous la forme d'une classe route. Chaque route pouvait ainsi accumuler du flux passant dessus, et le représenter en conséquence par des couleurs différentes, son intensité étant normalisée par le flux maximal absorbé par une route, et des poids étant aussi mis en fonction du type de route sur lequel on se trouve (petite rue, avenue, boulevard). La méthode pour détecter informatiquement les noeuds a pris beaucoup de temps à être réalisée, notamment par de nombreux tâtonnements. L'idée générale a rapidement été de générer de nombreux itinéraires aléatoires sur une portion de quartier, et pour chaque itinéraire détecter des changements brusques de direction le long d'une route, tendant à indiquer une intersection. La manière de générer ces itinéraires a été pensée pour couvrir le mieux le quartier, notamment en faisant des itinéraires entre les bords de la portion étudiée. Cette fonction n'est toujours pas totalement au point, car manque parfois quelques noeuds, mais présente déjà des résultats très satisfaisants. Une amélioration a notamment été de faire retravailler le programme sur les zones où il manque des noeuds, pour générer des itinéraires plus resserrés sur cette zone, et ainsi augmenter le taux de détection.

Une fois cette méthode plus ou moins aboutie, il a fallu coder la fonction pour détecter le type de route étudié entre deux noeuds, selon qu'il s'agisse d'une rue, avenue, boulevard, et ce grâce à la couleur sur la carte Openstreetmap, qui représente ces différences par plusieurs couleurs. Notre méthode a été de prendre une capture d'écran d'une carte du quartier, puis faire correspondre les pixels de la capture aux coordonnées réelles, pour détecter la couleur des pixels correspondant à chaque route. Nous avions également pensé à trouver une API pour réaliser cela, même toutes celles trouvées offraient des temps de réponse beaucoup trop faibles pour être utilisables. Réaliser ces deux étapes a pris près d'un mois, entre mi mai et mi juin.

Une des dernières étapes a été de faire tourner le tout en condition réelle, en reliant chacun des noeuds aux demandes des magasins environnants, pour injecter le tout dans vrpy (bibliothèque Python pour optimiser les coûts, ici temps, de transports de marchandises) pour obtenir les tournées de livraison à représenter. Cependant, nous avons aussi vite vu que les temps de calcul résultants pour vrpy étaient colossaux, et pas forcément envisageables pour notre utilisation. L'idée a alors été de diviser les grands quartiers en sous-quartiers, découpés de manières rectangulaires, et en simulant à chaque fois les tournées à faire sur chacun des sous quartiers séparément (l'idée étant qu'a priori, un camion faisant une tournée n'allait pas aller d'un bout à l'autre du quartier pour la réaliser, mais visiter des noeuds proches les uns des autres). Cette méthode s'est montrée assez fructueuse, et a encore pu être améliorée

grâce à des techniques de parallélisation (grâce au module multiprocessing) afin de faire fonctionner les différents coeurs du processeur sur cette tâche, sous-quartier par sous-quartier. Toutes ces méthodes ont ainsi été réalisées entre mi juin et début juillet.

Un des problèmes rencontrés pour travailler sur ce projet a aussi été le fait que l'API Géoportail, permettant de simuler les itinéraires entre deux points, avait tendance à offrir des temps de réponse très faibles en journée, sans doute de par le fait qu'elle était très sollicitée, et nous obligeait ainsi à travailler sur le projet en horaires parfois un peu décalés, pour pouvoir avoir des temps de réponses décents. Le programme finalement obtenu donne ainsi des résultats intéressants, toutes ses fonctions réalisant bien ce qu'elle a à faire. Des améliorations sont cependant toujours possibles sur la détection des noeuds, la représentation plus fidèle des flux, ainsi que la diminution du temps total de calcul, même si sur ce point l'utilisation d'un processeur de bonne qualité résoudrait sans doute beaucoup de problèmes.