Element de Recherche Opérationnel

Etienne CHANUDET
Hugo GENDARME
Sacha ATHIAS
Massil FERHANI
Vincent MUSCEDERE

3 juin 2021

Table des matières

1 Introduction

Lors de ce projet nous devions aider la ville de Montréal à optimiser le déneigement des rues. Nous avons traité le problème sous deux angles. Le premier est le repérage des rues à déneiger grâce à un drone et le second aspect qui est d'optimiser le trajet des déneigeuses. Pour se faire nous avons donc dû transformer la ville de Montréal en un graph non orienté pour le drone et en graph orienté pour le trajet des déneigeuses.

2 Solution trouvée

2.1 Solution pour le drone

Le drone permet d'analyser la quantité de neige dans les rues de la ville. Il parcourt la ville sans prendre en compte le sens de circulation et peut analyser toutes les voies d'une rue en un seul passage, peu importe le sens de circulation.

Afin de trouver un chemin optimal permettant au drone d'analyser toutes les rues de la ville le plus efficacement possible, nous avons choisi de représenter la ville sous forme de graphe non orienté, où les arêtes représentent les rues et les sommets représentent les intersections.

La première étape pour trouver un chemin est de rendre notre graphe eulérien. En le rendant eulérien, nous autorisons au drone de passer plusieurs fois par une rue, mais cela lui permet de ne pas rester bloqué à une intersection faute de route non utilisée.

Un graphe est eulérien quand tous ses sommets sont de degré pair. Afin de rendre le graphe eulérien, nous avons commencé par trouver tous ses sommets impairs. Ensuite, à partir de la liste des sommets impairs, nous avons utilisé l'algorithme hongrois, permettant de résoudre le problème d'affectation, afin de créer des couples de sommets. Ces couples de sommets représentent les nouveaux chemins à ajouter au graphe afin de le rendre eulérien.

Dans notre cas, nous avons modifié l'algorithme afin de pouvoir récupérer les sommets intermédiaires et ainsi créer une route avec une liste d'intersections contiguës.

Une fois le graphe rendu eulérien, il nous suffit de trouver un cycle eulérien ou un chemin eulérien. Dans notre cas, nous avons décidé de chercher systématiquement un cycle eulérien, car contrairement au chemin eulérien, le point de départ et le point d'arrivée sont les mêmes. Cela nous permet de prendre en compte le trajet de retour à l'entrepôt et ainsi diminuer les coûts.

2.2 Solution pour la déneigeuse

La déneigeuse va circuler dans les rues de la ville afin de déblayer la neige. Contrairement au drone, elle doit suivre la signalisation et le sens de circulation. De plus, lorsqu'une rue est à double sens, la déneigeuse doit passer dans les deux sens afin de déblayer les deux sens de circulation.

Pour trouver un chemin permettant à la déneigeuse de déblayer toutes les rues, nous avons décidé de représenter la ville sous forme de graphe orienté. Seulement, lorsque nous obtenons le graphe, celui-ci est mixte, c'est-à-dire que les arêtes en double sens ne sont pas orientées. Afin de simplifier la recherche du cycle eulérien, nous avons transformé ce graphe mixte en graphe orienté en remplaçant les arêtes non orientées par deux arêtes, une dans chaque sens.

Afin que la déneigeuse ne se retrouve pas bloquée dans une rue à sens unique sans issue, nous avons décidé de garder uniquement la plus grande composante connexe du graphe.

Ensuite, comme pour le cas du drone, nous devons rendre ce graphe eulérien afin que la déneigeuse ne reste pas bloquer à une intersection. Pour cela, nous devons trouver les sommets ayant un nombre d'arêtes entrantes différent du nombre d'arêtes sortantes, et créer des chemins entre celles où il manque des connexions entrantes et celles où il manque des connexions sortantes. Comme pour le drone, nous avons utilisé l'algorithme hongrois afin de résoudre ce problème d'affectation.

Enfin, une fois le graphe rendu eulérien, nous cherchons un cycle eulérien passant par toutes les arêtes du graphe, qui représentera le trajet de la déneigeuse pour déblayer la ville.

3 Choix des algorithmes

Pour rendre le graphe eulérien, nous cherchons les sommets impairs, et créons des couples entre ces sommets afin de créer de nouvelles arêtes. Afin de réaliser cela, nous utilisons deux algorithmes : dijkstra et l'algorithme hongrois.

Dijkstra nous permet de calculer la plus courte distance entre deux sommets et ainsi former une matrice de distance entre chaque sommet impair. Nous aurions pu utiliser l'algorithme de Floyd Warshall ou celui de Bellman Ford afin d'obtenir la matrice des distances avec un seul appel. Cependant, nous n'avons pas besoin de la distance entre chaque sommet du graphe, mais seulement entre les sommets impairs. Donc la complexité est moindre en appelant plusieurs fois Dijkstra entre les sommets impairs que l'un des deux autres algorithmes, sauf dans le cas où la majorité des sommets du graphe sont impairs.

Ensuite, l'algorithme hongrois nous permet, à partir de la matrice créée grace à des appels successifs à Dijkstra, de trouver le meilleur arrangement entre les sommets pour former une liste de couples avec une distance totale optimale. Nous avons choisi cet algorithme car il résout le problème d'affectation en temps polynomial.

Dans le cas de la déneigeuse, nous utilisons l'algorithme de Tarjan afin de trouver les composantes fortement connexes, et ensuite garder seulement la plus grande. Nous avons choisi cet algorithme grâce à sa complexité linéaire, en parcourant une seule fois le graphe, au lieu de deux pour l'algorithme de Kosaraju.

4 Problèmes rencontrés

Le premier problème que nous avons rencontré est lors de l'obtention d'une route en résultat du programme. Ayant ajouté des chemins entre les noeuds impairs afin de permettre de trouver un circuit eulérien dans le graphe, certains des sommets de ces nouveaux chemins n'étaient pas contigües mais mis l'un à la suite de l'autre dans le chemin résultant. Cependant, un employé ne peut pas se téléporter d'un point à un autre s'ils ne sont pas liés directement. Nous avons donc dû retrouver la suite de noeuds avec la plus courte distance permettant de relier ces points et permettre au chauffeur de conduire sereinement.

Un deuxième problème s'est posé lorsque l'on devait trouver une route permettant de parcourir toute la ville pour les déneigeuses. Lors de la récupération du graphe avec OSMNX, certaines voies en sens unique allant d'une ville à une autre étaient coupées. Nous avons donc dû supprimer ces voies sans issue et garder seulement la plus grande composante fortement connexe.

Enfin, nous voulions afficher la route résultante de manière lisible pour tous et non simplement une liste de noeuds difficiles à analyser. Nous avons donc opté pour afficher le résultat avec une animation, où un point se déplace d'intersections en intersections telle une voiture suivant le GPS.

5 Limites de la solution trouvée

Notre solution fonctionne avec n'importe quelle ville. Cependant, pour les grandes villes comme Montréal ou Paris, le temps de calcul du chemin est très long (plus de 30 minutes), que ce soit pour le drone ou la déneigeuse. Nous avons donc décidé de présenter notre solution sur des villes plus petites (Issou pour le trajet du drone et Nice pour le trajet de la déneigeuse). Mais le fait qu'il soit lent à calculer un trajet limitent ses utilisations dans un milieu réel.

Ensuite, sur le graphe généré grâce à OSMNX, il y a des rues à sens unique sans issue qui représente des rues allant d'une ville à une autre. Dans la réalité, les déneigeuses iraient déblayer ces rues et feront demi-tour dans l'autre ville. Dans notre cas, nous avons décidé d'enlever ces impasses dans le cas de la déneigeuse, afin que celle-ci ne reste pas bloquée. Une amélioration serait d'identifier ces rues afin de les intégrer au graphe et ainsi permettre à

la déneigeuse de déblayer toute la ville. (Ce problème ne se présente pas pour le drone qui ne prend pas en compte le sens de circulation).

6 Conclusion

Ce projet nous a permis de nous mettre dans un contexte plus professionnel que les autres projets YAKA/ACU de l'année d'ING1. Nous avons dû développer une solution à partir d'un problème posé, suivant un réel processus de recherche et de développement. Nous avons dû faire face à des problèmes divers qui ont testé nos capacités d'adaptation. Enfin, il nous a permis de mettre en pratique plusieurs notions apprises durant cette année, comme les graphes et les complexités algorithmique afin d'avoir une solution la plus efficace possible.