Epita Lyon
Élément de Recherche Opérationnelle

Rapport de Projet

Déneiger Montréal

Membres du groupe :

Pauline MARSAUDON
Nathan BONELLI
Yoan BIGGIO
Enzo SOLER
Thomas FLORION

Table des matières

1 Introduction **3**

2 Données **3**

3 Drone **4**
3.1 Modélisation 4
3.2 Les itérations 4
3.2.1 Itération 1 4
3.2.2 Itération 2 4

4 Déneigeuses **5**
4.1 Modélisation 5
4.2 Itération 1 5
4.3 Itération 2 5
4.4 Itération 3 6

5 Conclusion **7**

1 Introduction

Ce projet est une simulation de déneigement de Montréal. Le principe est simple. Tout d'abord, nous devons utiliser un drone qui va déterminer quelles routes il faudrait vraiment déneiger, en parcourant tout Montréal. Le résultat du drone nous dit qu'il y a de la neige dans les cinq quartiers de Montréal : Outremont, Verdun, Saint-Léonard, Rivière-des-prairies-pointe-aux-trembles et Le Plateau-Mont-Royal.
À ce moment là, nous devons déterminer le nombre de déneigeuses à utiliser pour déneiger ces quartiers. Nous devons aussi avoir leur itinéraire afin que l'on puisse utiliser ces données en temps réel.
Enfin, nous devons avoir un modèle qui va minimiser le coût des déneigeuses ainsi que le temps qu'elles passent à déneiger.
Pour simuler le parcours de Montréal et de ses quartiers, nous avons utilisé une librairie python s'appelant "OSMnx". Le principe est que cette librairie va transformer une carte de Montréal en graphe. Les arcs du graphe symbolisent les rues et les noeuds représentent les intersections. Nous allons donc présenter le processus qui nous a permis de trouver une solution plus ou moins optimale suivant les demandes de la mairie de Montréal.

2 Données

Données utilisées :

- Super Drone :
  - Coût fixe : 100 €/jour
  - Coût kilométrique : 0.01 €/km
  - Vitesse : 60 km/h (vitesse des drones entre 20 et 80 km/h)
- Déneigeuse type 1 :
  - Coût fixe : 500 €/jour
  - Coût kilométrique : 1.1 €/km
  - Coût horaire les 8 premières heures : 1.1 €/h
  - Coût horaire au delà des 8 premières heures : 1.3 €/h
  - Vitesse : 10 km/h
- Déneigeuse type 2 :
  - Coût fixe : 800 €/jour
  - Coût kilométrique : 1.3 €/km
  - Coût horaire les 8 premières heures : 1.3 €/h
  - Coût horaire au delà des 8 premières heures : 1.5 €/h
  - Vitesse : 20 km/h

3 Drone

3.1 Modélisation

Pour le drone, nous avons modélisé cela comme un problème du postier chinois : il faut passer par toutes les arêtes. Nous avons alors à résoudre un problème linéaire. Notre fonction à minimiser est la suivante : min Z = 2 * coût + temps de compilation
Nos variables sont :
temps = distance parcourue / 60
coût = 100 * roof(temps // 24) + 0.01 * temps
Le signe // correspond à la division entière et roof est la fonction d'arondi au supérieur : roof(1) = 1, roof(0,4) = 1. Nous avons aucune contraintes dans ce modèle.

3.2 Les itérations

Il faut alors passer une fois par chaque arête individuellement. Le plus simple pour cela est de rendre le graphe eulérien et y trouver un cycle eulérien. Nos deux itérations sont alors orientées selon ce principe.
La différence entre les deux itérations est la façon de rendre le graphe eulérien.

3.2.1 Itération 1

Lors de la première itération, nous avons pensé à la librairie "networkx" et sa fonction "eulerize". Cela fonctionnait très bien sur des petits quartiers mais prenait beaucoup de temps sur les plus gros.
Après 7h de compilation sur Montréal, sans résultats, cette solution fut abandonnée. De plus, la fonction n'était pas du tout optimale. En effet, elle rendait le graphe eulérien en ajoutant le moins d'arêtes possible. Cela n'est pas toujours le mieux. Par exemple, ajouter 2 arêtes de 100m est mieux qu'une seule de 1 km.

3.2.2 Itération 2

Lors de la seconde itération, nous avons donc choisi de coder notre propre fonction pour rendre le graphe eulérien en fonction de la distance à parcourir entre les points en vol d'oiseau (car un drone, ça vole). Pour connaître cette distance, nous avons utilisé la librairie "geopy" et sa fonction "geodesic" qui prend des coordonnées GPS (longitude, latitude) et renvoie une distance (en kilomètres) car nous avons accès à de telles coordonnées dans un graphe osmnx.
Voici alors le résultat obtenu avec cette solution :

Parcours du drone sur tout Montréal
Rouge foncé => chemins ajoutés afin de rendre le graph eulérien

La distance totale parcourue 4924.91 km.
Le prix de ce parcours est de 449.25 €.
Le temps du parcours est alors de 3 jours, 10 heures, 48 minutes et 36 secondes.
Le temps de compilation est de 1 heure, 15 minutes et 36 secondes.

Cette solution est plus efficace car Z est plus bas ; que cela concerne les coûts ou le temps de compilation (qui a drastiquement baissé).

4 Déneigeuses

4.1 Modélisation

Comme pour le drone, nous avons modélisé le problème comme un problème du postier chinois. Nous avons alors à résoudre un problème linéaire. Notre fonction à minimiser est la suivante : min Z = coût + temps
Nos variables sont :
Temps1 = liste des temps d'utilisation des déneigeuses de type 1. Temps2 = liste des temps d'utilisation des déneigeuses de type 2. temps = Temp1 + Temp2
Dist1 = liste des distances parcourues par les déneigeuses type 1.
Dist2 = liste des distances parcourues par les déneigeuses type 2.
L1 = nombre de déneigeuses type 1
L2 = nombre de déneigeuses type 2
coût1 = $\sum_{i=0}^{L1}$ 500 * (roof(Temps1[i] // 24)) + 1.1 * Dist1[i] + min(8, Temps1[i]) * 1.1 + max(0, Temps1[i] - 8) * 1.3
coût2 = $\sum_{i=0}^{L1}$ 800 * (roof(Temps2[i] // 24)) + 1.3 * Dist1[i] + min(8, Temps1[i]) * 1.3 + max(0, Temps1[i] - 8) * 1.5
coût = coût1 + coût2
Le signe % correspond au modulo, // correspond à la division entière et roof est la fonction d'arondi au supérieur : roof(1) = 1, roof(0,4) = 1.
Nos contraintes sont :
- Le graphe est orienté. - Les valeurs aberrantes de prix ou de temps tel que 2000€ ou 48h pour un quartier.

4.2 Itération 1

L'objectif de cette itération est de découvrir les algorithmes de parcours de graphe et de simplement parcourir tous les arcs. Ainsi, nous ne nous sommes pas imposés de contraintes pour cette version.
À ce moment-là, nous n'avions pas très bien compris le sujet et nous pensions que seulement une partie des routes des quartiers devaient être déneigées.
Nous avons donc pris en argument une liste d'arcs qu'il faut visiter et nous avons fait un algorithme qui fait apparaître une déneigeuse sur une intersection aléatoire et parcourt les arcs allant de noeud en noeud en prenant à chaque fois le noeud le plus proche à vol d'oiseau.

4.3 Itération 2

Pour cette itération, nous avons repris le principe de la première itération mais en ajustant un certain nombre d'améliorations qui s'adonnent mieux au sujet.
Nous nous sommes donnés comme objectif que cet algorithme soit assez minimisé au niveau du temps nécessaire afin de déneiger un quartier. Nous avons géré le fait qu'il puisse y avoir plusieurs voies et des doubles sens dans

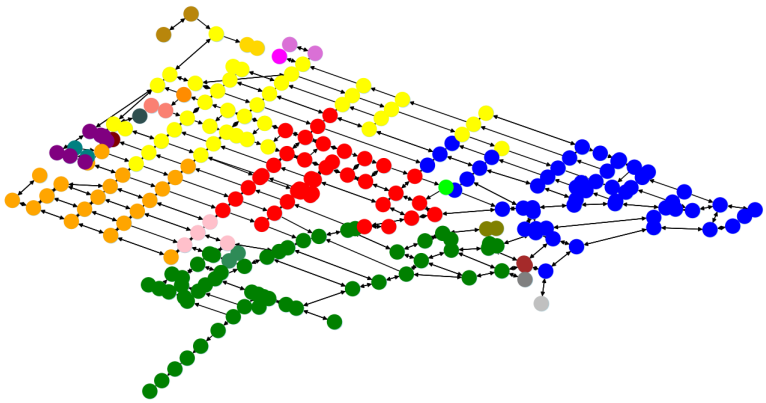
une même rue, ce qui optimise déjà cela. Nous avons aussi pris en compte qu'il fallait déneiger tout le quartier au lieu de quelques routes isolées. Ainsi, nous avons pu parcourir tout le quartier en déneigeant au fur et à mesure. Dans cet algorithme, nous avons fait en sorte qu'une déneigeuse parcourt le quartier jusqu'à ce qu'elle soit bloquée. À ce moment-là, une autre déneigeuse apparaît à un endroit qui n'a pas encore été déneigé et continue comme celle d'avant.

Tableau des résultats obtenus avec l'itération 2

Statistiques / Quartiers	Verdun	Outremont	Rivière-des-prairies-pointe-aux-trembles	Le Plateau-Mont-Royal	Saint-Léonard
Distances parcourues	0,37 km	700 km	3000 km	30000 km	5000 km
Nombre déneigeuses	3	24	38	25	9
Coût total type 1	2000 €	13500 €	70300 €	18100 €	13300 €
Coût total type 2	3200 €	23500 €	82300 €	24000 €	15900 €
Temps de déneigement type 1	0 jour, 0 heure	2 jours, 22 heures	21 jours, 4 heures	128 jours, 22 heures	13 jours, 2 heures
Temps de déneigement type 2	0 jour, 0 heure	1 jour, 12 heures	10 jours, 10 heures	64 jours, 3 heures	6 jours, 9 heures

4.4 Itération 3

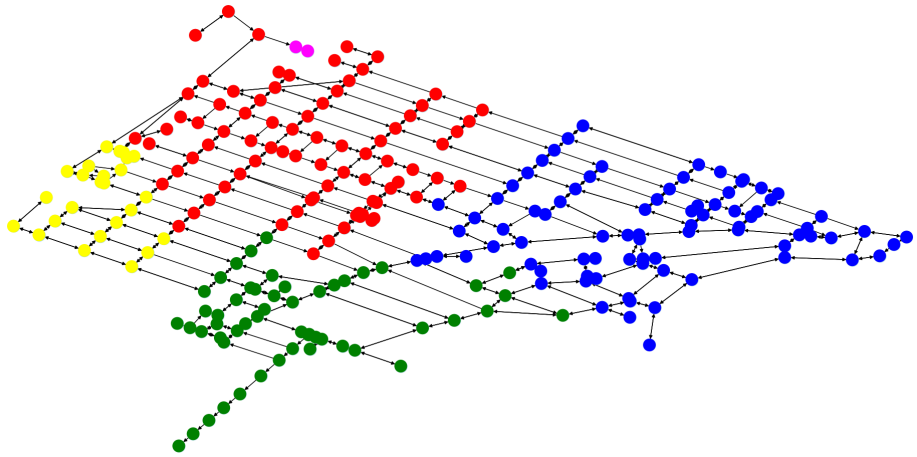
Dans le cadre de ce projet, nous avons entrepris d'explorer des méthodes d'optimisation pour la répartition des déneigeuses dans un quartier. L'objectif était de développer un algorithme capable de diviser le quartier en sous-quartiers de tailles équivalentes, afin d'optimiser l'efficacité du travail de déneigement. Nous nous sommes également intéressés à la minimisation des sous-quartiers trop petits, qui ne justifieraient pas l'utilisation d'une déneigeuse.
Nous nous sommes inspirés du concept de bruit de Worley pour développer notre algorithme de division équitable du quartier. L'idée était de subdiviser le quartier en sous-quartiers de tailles approximativement équivalentes, en utilisant des techniques de parcours récursif de graphe. Cette approche permettrait de distribuer équitablement le travail entre les déneigeuses et d'optimiser l'efficacité globale du déneigement.



Répartition des déneigeuses dans le graph v1

Après avoir mis en place notre algorithme initial, nous avons constaté que certains nœuds du graphe étaient divisés en sous-quartiers extrêmement petits. Il était évident que l'utilisation d'une déneigeuse pour de tels espaces serait inefficace et non rentable. Afin de remédier à ce problème, nous avons eu l'idée d'identifier les plus petites divisions du quartier et de les considérer comme faisant partie du sous-quartier valide le plus proche. Cette approche a permis d'optimiser davantage la répartition des déneigeuses en éliminant les sous-quartiers

non viables.
Après avoir réalisé ces améliorations, nous disposons maintenant d'un algorithme fonctionnel qui divise efficacement un quartier en plusieurs sous-quartiers. Une caractéristique clé de notre algorithme est la possibilité d'ajuster la taille des divisions en fonction des besoins spécifiques du client. Cela permet de s'adapter à différentes configurations de quartiers et de répondre de manière optimale à la demande de déneigement.



Répartition des déneigeuses dans le graphe v2

Ainsi, grâce à cet algorithme, il nous est possible d'ajuster le nombre de déneigeuses par quartier, nous pouvons aussi avoir des statistiques sur quel type de déneigeuse est le meilleur suivant la modélisation du problème que nous choisissons.
Néanmoins, nous n'avons pas réussi à faire un parcours de graphe correct donc nous ne pouvons pas donner les résultats de cette itération même s'ils seraient bien plus optimisés que ceux de la deuxième itération.

5 Conclusion

Voici donc, comment nous avons réalisé ce projet.
Pour tester notre travail, il suffit de lancer l'éxecutable "./main" dans un terminal.
Aussi, le script "install_libraries.py" peut être utilisé pour installer les bibliothèques utilisées dans ce projet.
Ce projet nous aura appris ce qu'est un projet de grande envergure, avec une grosse majorité des éléments qui se rapportent à des données réelles. Il nous a aussi appris l'importance d'une formalisation des modèles. Il est important d'avoir en tête les objectifs d'un projet. Cependant pour un projet de cette envergure, il faut aussi prendre le temps de bien modéliser le problème.