Rapport de Projet ERO1

Membres du groupe 123 :

Tan Thibault
Régnier-Vigouroux Nicolas
Buat Charlotte
Kieraga Florine
Carret Sami

Sommaire

1 Introduction

Dans ce rapport, nous détaillerons les différentes tâches que nous avons dû réaliser pour implémenter un algorithme de déneigement de la ville de Montréal.
L'objectif final de ce projet est la mise en place d'un algorithme qui va d'abord parcourir Montréal avec un drone pour déterminer quelles zones de Montréal doivent êtres déneiges.
Ensuite nous devons mettre en place un autre algorithme permettant de déterminer le trajets des déneigeuses de 2 types différents en fonction de plusieurs contraintes tels que le coût et le temps de déneigement.
Le tout devant suivre un temps d'exécution le plus bas possible.

2 Trajet minimal du drone

2.1 Objectif

L'objectif de cette partie est de parcourir la totalité des rues de Montréal le plus efficacement possible à l'aide d'un drone possédant les caractéristiques suivantes

- Coût fixe : 100 €/jour
- Coût kilométrique : 0.01 €/km
- Le drone peut emprunter des routes sans se soucier du sens de circulation des voitures, il parcourt donc un graphe non orienté

2.2 Notre solution

Pour parcourir Montréal de la manière la plus efficace possible, nous avons commencé par Euleriser le graphe obtenu avec la bibliothèque NetworkX pour obtenir un graphe que nous pouvons parcourir en une seule fois.

Puis nous avons suivi un algorithme de Euleurian circuit pour parcourir l'entièreté des arrêtes de ce graphe à partir d'un noeud pris au hasard de la manière la plus efficace possible ce qui nous renvoie un cycle Eulerien contenant le parcours du drone pour traverser chaque arrête avec les distances entre chaque noeud que nous pourrons ensuite utiliser pour parcourir le graphe avec des déneigeuses. Ce fut notre premier tour de boucle sur le parcours de drone.

Pour notre deuxième tour de boucle nous avons décidé d'améliorer la fonction qui permettait d'eulerize notre graphe trouvée dans la librairie. On a utilisé l'algorithme du postier chinois pour implémenter l'ajout d'arêtes qui fonctionne sur plusieurs étapes :

- On trouve tous les sommets de dégrée impair et on les met dans une liste
- On énumère toutes les combinaisons de couples de sommets possibles dans cette liste
- Pour chaque combinaison de couples, on trouve le chemin le plus court les connectant
- On trouve la combinaison de couples avec le chemin le plus court le moins coûteux
- On ajoute au graphe les arrêtes représentant les couples trouvés dans l'étape précédente

Cette méthode nous a permis d'économiser 2 km sur le parcours par rapport à la première méthode.

2.3 Difficultés

Pour cette partie nous avons commencé par utiliser la fonction eulerize de la bibliothèque NetworkX. Malheureusement cette fonction n'utilisait pas les poids de chaque arrête de la bonne manière.

Nous avons donc décidés de coder notre propre fonction eulerize adaptée à notre problème ce qui nous à permis de compléter cette étape en prenant en compte les contraintes qui nous étaient fixées.

3 Déneigement

3.1 Objectif

L'objectif de cette partie est de trouver un moyen de parcourir toute les arrêtes du graphe représentant Montréal en minimisant le coût et le temps de travail. Nous avons les données suivantes:

- Coût fixe : type I : 500€/jour, type II : 800 €/jour
- Coût kilométrique : type I : 1.1 €/km, type II : 1.3 €/km
- Coût horaire les 8 premières heures : type I : 1.1 €/h, type II : 1.3 €/h
- Coût horaire au delà des 8 premières heures : type I : 1.3 €/h, type II : 1.5 €/h
- Vitesse moyenne : type I : 10 km/h, type II : 20 km/h
- Les déneigeuses doivent suivre le sens de circulation des routes

Pour se faire, nous avons commencé par modéliser notre problème avec une programmation linéaire:

- On a d une liste de la distance totale a déneiger pour chaque déneigeuse
- x1 le nombre de déneigeuses de type 1
- x2 le nombre de déneigeuses de type 2

Programmation linéaire pour les déneigeuses:

$min\ Z = (500 * x1) + (800 * x2) + \Sigma_{k=0}^{x1}(1.1 * d[k] * x1) + \Sigma_{k=0}^{x2}(1.3 * d[k] * x2) + \Sigma_{k=0}^{x1}(1.1 * min(d[k]\,/\,10,\,8) * x1) + \Sigma_{k=0}^{x2}(1.3 * min(d[k]\,/\,20,\,8) * x2) + \Sigma_{k=0}^{x1}(1.3 * max(d[k]\,/\,10 - 8,\,0) * x1) + \Sigma_{k=0}^{x2}(1.5 * max(d[k]\,/\,20 - 8,\,0) * x2)$

sous les contraintes
$x1 \geq 0$
$x2 \geq 0$

3.2 Notre solution

Pour notre premier tour de boucle nous avons comme méthode initiale de découper notre graphe en ses composantes fortement connexes puis d'eulerizer ce graphe avec la méthode ci-dessous:

- On trouve les noeuds avec un demi-degré intérieur et extérieur différent et on les classe si l'un est plus grand que l'autre.
- Pour chaque noeud de demi-degré extérieur supérieur on teste le chemin le plus court avec un noeud de demi-degré intérieur supérieur
- On ajoute ensuite les arrêtes du chemin le plus court le moins coûteux au graphe et on retire les 2 noeuds de nos listes

On pourra alors chercher le chemin eulerien dans ce graphe et d'envoyer une déneigeuse pour s'occuper de ce circuit. Il restera alors des routes à sens-unique où l'on enverra d'autres déneigeuses s'occuper des chemins restants.

Pour le deuxième tour de boucle nous avons décidé d'améliorer l'utilisation des déneigeuses après avoir obtenu les graphs des quartiers eularisés. De base, on envoyait une déneigeuse pour s'occuper du circuit principal mais il est plus rapide d'en envoyer plusieurs et de découper le circuit en plusieurs boucles pour chaque déneigeuse.

Grâce à la programmation fonctionnelle on sait le nombre optimal de déneigeuses des 2 types et donc de la distance idéale à déneiger pour une machine.

On ajoute alors une arête au parcours de graph d'une déneigeuse jusqu'à atteindre la limite fixée puis de passer la suite du parcours à une autre. On a alors une solution plus rapide et plus performant que la précédente au détriment du coût global et du temps de calcul plus long.

3.3 Limite du modèle

Nous avons pu identifier plusieurs limites à notre modèle. Tout d'abord, il ne prend pas en compte le trafic sur les routes. Ce qui fait que le trajet pourrait prendre plus de temps que ce qui était calculé initialement.

Il ne prend pas non plus en compte les conditions météorologiques(ex: tempête de neige), qui pourraient empêcher les déneigeuses de passer ou leur compliquer le travail.

Une autre des limites de notre programme est l'application de l'algorithme de déneigement à tout Montréal. Bien que nous puissions le faire assez rapidement sur les quartiers séparément, le faire sur un regroupement de tous les quartiers prend beaucoup plus de temps.

Aussi, nous n'avons pas pris en compte les employées et leurs durées de travail. ils peuvent donc travailler pendant plus de 8h consécutives, ce qui n'est pas réaliste dans la vrai vie.

3.4 Amélioration

Comme autre amélioration pour notre troisième tour de boucle on pensait à créer le graph réduit de notre graph initial. Cela nous permettait de déplacer une déneigeuse d'une composante fortement connexe à une autre et de prendre une route à sens unique. Cela faisait économiser l'utilisation d'une déneigeuse. Malheureusement, il était trop difficile à trouver le chemin pour passer d'une composante à une autre.

4 Bibliographie

* Problème du postier chinois
* Problème du postier chinois (networkx)
* Eulerization d'un graphe (networkx)
* Trouver un circuit eulerien (networkx)