Agoulif Youssef [youssef.agoulif@epita.fr]
Ferroni Sandro [sandro.ferroni@epita.fr]
Cardi Julien [julien.cardi@epita.fr]
Melanger Alexandre [alexandre.melanger@epita.fr]

ERO1

1) Résumé des données utilisées et contraintes :

Pour notre projet, nous avons plusieurs données, notamment les caractéristiques du drone, c'est-à-dire son prix d'utilisation quotidien de 100 €, et son prix à la consommation de 0,01 €/km, afin de calculer le prix de son utilisation. On a également pris en compte ces caractéristiques pour les 2 déneigeuses 500 € ou 800 € par jour, 1,1 €/km ou 1,3 €/km, 1,1 €/h les 8 premières heures puis 1,3 €/h ou 1,3 €/h puis 1,5 €/h, mais également la vitesse de 10km/h et 20 km/h. À cela, s'ajoute la longueur de chaque rue de la ville.

Pour le drone, nous avons choisi un point de départ de manière aléatoire, car le technicien n'est pas obligé de se rendre à la mairie par exemple pour lancer le drone, et parcourt toute la ville. Cependant, le programme est très long, il y a donc une implémentation du drone disponible pour chaque quartier à déneiger.

Pour les déneigeuses, chaque déneigeuse démarre de l'entrepôt le plus proche du quartier, ses kilomètres parcourus et le temps de trajet sont compris dans nos variables affichées, ils démarrent leur déneigement dès l'entrée du quartier et retournent à l'entrepôt à la fin.

2) Hypothèses et choix de la modélisation :

Pour le drone, on a donc fait l'hypothèse que le technicien pourrait le lancer de n'importe où, que celui-ci parcourt la ville sans difficulté (ex: pas d'obstacle, oiseaux, panneaux, ...), qu'il était pilotable à une distance quasi-infinie (pas la peine de se trouver à 200 mètres) et de durée infinie (le technicien ne fait pas de pause).

Pour l'implémenter, on a utilisé dans la bibliothèque NetworkX la fonction eulerize et eulerian_circuit, sans prendre en compte l'aspect orienté. Ces fonctions vont rendre le graphe eulérien et ensuite donnent le circuit

eulérien présent. Par la suite, on calcule la distance parcourue et le prix, puis on affiche le chemin effectué par une liste de rues, sous cette forme : “(noeud_depart, noeud_arrive)”. Ainsi que le graphe du quartier avec les rues empruntées en rouge et en bleu celles non empruntées (on ne voit pas de bleu = toute la ville est bien parcouru).

Pour les déneigeuses, on a supposé qu’il y avait toujours des machines à disposition dans les entrepôts et que celles-ci roulaient à vitesse constante en permanence (aucun obstacle, essence illimitée, pas de pause, 24 h/24, pas d'arrêt : ni feu rouge ni stop …).

Pour l'implémenter, on a utilisé le même code que le drone, à l'exception que le graphe est orienté donc à chaque fois que le circuit devait passer par une route dans un sens interdit, on utilisait shortest_path, qui nous donnait le chemin le plus rapide pour aller au prochain sommet, on passait l'arrêt dans le bon sens puis on y retournait de la même façon, cette solution n’est pas la plus optimale, mais fonctionne (on en reparle partie 4).

Nous avons également implémenté le déneigement avec n déneigeuses pour un même quartier, on garde le même code à l’exception que le circuit eulérien est coupé en n listes de routes, toutes les déneigeuses démarrent du même point et vont effectuer leur partie de circuit et reviennent au point initiales pour retourner à l’entrepôt.

3) La solution retenue :

Que ce soit pour le drone ou pour la déneigeuse, nous avons décidé de garder cette solution, dans un premier temps, car d’autres essais n’ont pas abouti, mais aussi par un manque de temps. Cependant, nous sommes convaincus que nos solutions sont fonctionnelles, on constate avec l’affichage, que chaque parcours de déneigeuse fait obligatoirement plus de kilomètres que la somme de chaque longueur de routes du quartier. De plus, les points de départ et d'arrivé sont les mêmes. On constate que le chemin est cohérent, en effet, le nœud d’arrivé d’une rue et toujours le nœud de départ de la prochaine. L’affichage route par route nous prouve que le véhicule ne prend pas de sens interdit, et l’affichage des chemins pris en rouge montre bien que toutes les routes du quartier, on était emprunté. Si on compare avec nos différents scénarios implémentés, celui-ci fonctionne à 100 %, les anciens délaissaient parfois certaines rues ou prenaient des sens interdits notamment, un parcours profondeur ou de l'algorithme de Fleury. De plus avec notre nouveau code, nous pouvons démarrer du nœud que l’on souhaite ce qui nous permet de commencer le programme au point le plus

cohérent lors du démarrage du déneigement du quartier dans la vraie vie, c'est-à-dire la route la plus proche d'un entrepôt de déneigement.

On a constaté qu'avec plusieurs déneigeuses, le temps de déneigement sera plus rapide, cependant le nombre de kilomètres parcouru au total et le prix sera beaucoup plus important.

4) Les limites :

Notre code comporte certaines limites, notamment les hypothèses faites ne colle pas à la réalité. Les techniciens sont dans l'obligation d'effectuer des pauses, les machines ne peuvent pas rouler 24 h/24 et constamment à la même vitesse.

Niveau code, le programme est très long, cela est dû à la fonction eulerize, celle-ci possède une complexité temporelle non négligeable, elle va dans un premier temps chercher tous les nœuds de degré impaires, en se basant sur différent critères pour relier ces noeuds de façon à les rendrent paire en comparant avec tous les autres noeuds (distance entre les nœuds, le coût des arêtes supplémentaires, la structure du graphe), ce qui rend le processus très long, on aurait pu par exemple seulement relier avec le noeud le plus proche, en implémentant notre propre fonction, cependant le manque de temps nous a poussé à passer à autre choses et passer aux déneigeuses.

Pour les déneigeuses, on peut aussi remettre en cause l'algorithme, déjà pour les mêmes raisons, étant donné que la base est la même que celle du drone, mais aussi, car à chaque fois que l'on croise un sens interdit, dû à l'orientation du graphe, on se rend à l'autre bout de la rue par un autre chemin, on passe la rue et de nouveau le même processus sans prendre en compte que l'on a déneiger le chemin pris pour y arriver, et cela rajoute énormément de kilomètre. De plus, lorsqu'il y a plusieurs déneigeuses, chaque véhicule effectue une partie du circuit. Peut-être qu'avec un autre algorithme où chacune a un circuit, les kilomètres parcourus et ainsi l'argent dépensé pourrait être diminué.

5) Résultats

Lieux :	action :	prix (€) :	km :	durée :	temps exécution :
Montréal	drone	plus le	temps		+48h
Outremont	drone	100.57	56km		5sec
Verdun	drone	100.82	82km		5sec
Saint-Léonard	drone	102.32	231km		30sec
Rivière	drone	105.45	545km		13min
Plateau	drone	101.56	156km		10sec
Outremont	deneige	651.36	12,4km	13h	5sec
Verdun	deneige	720.13	18km	18h	6sec
Saint-Léonard	deneige	1152.14	53km	53h	45sec
Rivière	deneige	2353.11	150km	150h	14min
Plateau	deneige	1176.17	55km	55h	10sec

Exemple pour 4 déneigeuses sur Outremont : (2173€, 143km, 4h)

Les valeurs ont été prises ici uniquement avec la déneigeuse 1, la comparaison avec la deuxième sera proposé lors de l'oral (le km et le prix comportent également le trajet entre le quartier et l'entrepôt)

6) Conclusion :

Pour conclure, nous avons réussi notre but initial qui était de déneiger les quartiers donnés de Montréal, et d'effectuer un parcours de drone. Nous avons également réussi à mettre en place plusieurs déneigeuses sur un même quartier. Cependant, nous aurions aimé avoir plus de temps pour implémenter cela plus proprement dans le script shell et pour chaque quartier (sans modifier les données dans le fichier à chaque nouveau test ou nouveau quartier lorsque l'on utilise plusieurs déneigeuses).

Cependant, ce projet fut intéressant et nous a permis de réfléchir à de vrais problèmes en équipe afin de trouver une solution non-évidente qui pourrait le mieux satisfaire notre problème.