Éléments de recherche opérationnelle

I] Le trajet du drone

Afin de déterminer le trajet minimal du drone au-dessus du trafic routier, nous avons considéré un graphe non orienté, mettant ainsi le booléen de la fonction solve *is_oriented* à False.
Un membre de notre groupe a eu l'idée de considérer l'algorithme du postier chinois pour résoudre ce problème et nous l'avons découpé en plusieurs parties :

- trouver tous les nœuds impairs

- réaliser toutes les combinaisons possibles de ces nœuds impairs entre eux

- calculer la distance minimale entre ces nœuds

- générer un graphe complet à partir du dernier calcul

- trouver un couplage maximum

- ajouter le chemin augmentant au graphe de départ

Ces différentes étapes permettent de rendre le graphe eulérien afin de trouver par la suite un cycle eulérien de ce graphe qui sera la solution au problème du drone.

II] Le trajet de la déneigeuse

Pour la déneigeuse, nous avons considéré un graphe orienté (*is_oriented* est mis à True).
Le cas le plus simple est le cas où le graphe donné est déjà eulérien car il suffit de trouver un chemin eulérien.
Dans tous les autres cas, la solution est plus complexe car pour qu'un graphe orienté soit eulérien, il faut que pour chaque nœud, le nombre d'arêtes entrantes soit le même que celles sortantes.

Nous avons décidé de nous ramener au problème du flux maximal/minimal. On considère que les nœuds avec une demande positive (respectivement négative) sont ceux avec un nombre d'arêtes sortantes supérieur (respectivement inférieur) au nombre d'arêtes entrant.

A partir du résultat on rajoute les arêtes manquantes pour former un graphe eulérien et il ne nous reste plus qu'à trouver un chemin eulérien.

Cette solution n'est malheureusement pas adaptée avec les graphes qui ne sont pas « strongly connected », c'est-à-dire un graphe pour lequel chaque pair de nœuds est liée par un chemin allé et un retour, et constitue donc une piste d'amélioration. En effet, nous avons choisi de nous concentrer sur la zone « strongly connected » la plus grande possible et cela néglige les zones qui ne sont pas accessibles par la déneigeuse depuis notre point de départ (ce qui est arbitraire). Afin de régler ce problème, une solution aurait été de découper le graphe en plusieurs sous-graphes « strongly connected ». Ainsi, non pourrions exécuter notre algorithme sur chacun des sous-graphes.

Le temps de calcul pour une ville comme Montréal est de quelques minutes. Nous pensons que c'est largement acceptable et ne constitue donc pas une réelle amélioration possible.