APPERO

- Projet ERO-

LEMONNIER Alexandre

MORIN Jeanne

SIMONIN Victor

ANTON-LUDWIG Adrien

1 Représentation en graphe et Problème Eulérien

Le problème du déneigement à Montréal est un problème eulérien : on cherche à minimiser le trajet des véhicules déblayeurs à travers la ville en passant par toutes ses routes. Ainsi, on peut modéliser les rues et avenues de la ville par des arêtes d'un graphe qui relient une intersection à une autre et voir le chemin qui parcourt toutes les rues comme un cycle eulérien.

Dans le cas du survol de l'ensemble des rues de la ville par un drone, le parcours recherché n'est pas limité aux rues : le drone peut voler d'une intersection à l'autre. La seule condition est de parcourir au moins une fois toutes les rues. Conséquences, tous les sommets de la représentation en graphe seront reliés car ils peuvent être ralliés rapidement par les airs. De plus, le graphe étudié sera non-orienté car le sens de circulation n'a pas à être respecté lors du survol des rues. Enfin, on pourra se contenter d'un chemin eulérien et pas d'un cycle car de n'importe où en ville, le drone est capable de retourner à son entrepôt sans encombre.

Pour la minimisation du parcours des déneigeuses, la représentation graphique se complexifie. Les véhicules doivent respecter le sens de circulation des rues, le graphe sera ainsi orienté mais les rues à double sens vont poser problème. Il ne faut y passer qu'une fois : lors de la recherche de parcours il faudra donc les orienter.

2 Eulérisation d'un graphe

Pour chercher un cycle eulérien, il nous faut un graphe eulérien : or dans la majorité des cas, les graphes représentants des villes ne seront pas eulériens. Il faut donc trouver un moyen de rendre notre graphe eulérien.

Pour l'itinéraire du drone, il est possible de relier tous les sommets ensemble car le drone peut tous les rallier. Mais, nous ne pouvons pas calculer la notion de distance entre les sommets reliés par les airs et pas par la route.

Une autre solution s'offre à nous, il suffirait de trouver tous les sommets ayant un degré impair pour les relier deux à eux. Ainsi, tous les sommets du graphe non-orienté auraient un degré pair, le graphe deviendrait donc eulérien. D'après le lemme des poignées de mains, on sait que le nombre de noeuds de degré impair sera toujours pair.

Là encore, un problème se pose. Dans le cas de la déneigeuse, nous ne pouvons pas rajouter une arête entre 2 nœuds non reliés. Il faut donc améliorer la création des arêtes pour vérifier qu'il existe déjà un lien entre les nœuds de degré impair que l'on relie. Parfois, il n'existera pas d'arêtes existantes entre 2 noeuds de degré impairs, c'est donc une première limite importante à cette solution.

Nous avons finalement choisi une autre solution en deux étapes, qui consistent a orienter toutes les arêtes a doubles sens pour obtenir un graphe orienté, puis on ajoute des arêtes de sorte à ce que le graphe devienne eulérien.

2.1 Orientation du graphe : problèmes de flots maximal

Nous avons donc besoin d'orienter les arêtes à double sens de façon à transférer les surcharges sur des noeuds sous chargés. Bien sûr dans l'objectif d'avoir des noeuds les plus équilibrés possible. Cela s'apparente à trouver la solution à un problème de flot maximal. Nous utilisons donc l'algorithme d'Edmond-Karp pour orienter de la meilleure façon nos rues à double sens.

Cet algorithme est efficace mais ne permet pas d'orienter toutes les rues à double sens car certaines ont un flot identique. Il nous reste donc à les orienter de façon arbitraire pour finir l'orientation de nos doubles sens.

2.2 Ajout de nouvelles arêtes

Maintenant que nous avons obtenu un graphe orienté, nous souhaiterions le rendre eulérien. Pour cela, on va chercher à ajouter des arêtes. Le problème, c'est qu'ajouter des arêtes signifie que les déneigeuses vont devoir passer par des routes qu'elles auront déjà visitées par ailleurs. Or cela va rallonger le temps de parcours et ainsi le coût du déneigement. Pour réduire cette perte, on commence par calculer les plus courts chemins entre toutes les paires de sommets du graphe original, et cela grâce à l'algorithme de Bellman-Ford.

Après avoir créé un graphe biparti entre les noeuds surchargés et souschargés du graphe, copiés un nombre de fois égales à leur charge et reliés entre eux par le plus court chemin pour aller d'un noeud souschargé à un surchargé, on va essayé de déterminer le couplage parfait dans ce graphe biparti. Cela permettrait d'obtenir un graphe avec les meilleures arêtes ajoutées, c'est à dire celles équilibrant le graphe tout en impliquant le minimum de distance supplémentaire à parcourir.

Pour cela, on utilise l'algorithme Hongrois qui permet de résoudre le problème d'affectation, qui consiste à attribuer au mieux des tâches à des agents. Cet algorithme correspond exactement à la solution de notre problème, et une fois les meilleures affectations récupérées, il suffit de créer les arêtes correspondantes pour obtenir notre graphe eulérien.

3 Cycle eulérien : Construction et découpage

Pour trouver ce cycle eulérien c'est assez simple. On commence à notre point de départ, l'entrepôt idéalement ou un point arbitraire du graphe, puis l'on passe d'un sommet à l'autre en utilisant des arêtes qui n'ont pas été visitées jusqu'à retomber sur notre point d'origine. S'il existe encore un sommet ayant des arêtes non visitées, on réitère à partir de sa position et ajoute cette boucle à notre cycle eulérien.

Bien sûr, dans notre cas, nous avons des arêtes ajoutées par l'étape précédente au graphe de la ville et qui ne correspondent pas à une rue que nous pourrions emprunter dans la vraie vie. Après avoir calculé le cycle eulérien, nous allons remplacer les arêtes qui n'existent pas réellement par le plus court chemin qu'elles représentent et les ajouter au parcours. Nous obtenons alors notre cycle pseudo-eulérien de la ville.

Il nous faut maintenant trouver un moyen de découper ce trajet pour un nombre de déneigeuses variant. Nous avons plusieurs choix pour réaliser cela.

Une solution simple et naïve consiste a diviser la longueur totale du cycle eulérien par le nombre de véhicule. On obtiens un trajet par déneigeuse. On ajoute ensuite à ce trajet le plus court chemin entre l'entrepôt et le départ et l'arrivée, afin de fournir un itinéraire précis pour chaque véhicule. Une limitation importante de cette solution est que les véhicules ne vont pas tous parcourir la même distance et que les routes parcourues entre l'entrepôt et trajet de déneigement ne sont pas décomptées des routes à déneiger.

4 Modèle de coût pour les opérations de déblaiement

Une fois les trajets des différentes déneigeuses calculés. Nous en déduisons le temps nécessaire en estimant la vitesse des déneigeuses. Avec celui-ci on permet au client d'estimer le coût en fonction du prix de location des machines. Nous utilisons aussi la longueur du trajet pour estimer le coût en carburant de l'opération de déneigement.

Références

[1] a3nm, *Google Hash Code 2014*,
https ://a3nm.net/blog/google_hashcode_2014.html, 2015

[2] Wikipedia, *Eulerian path*,
https ://en.wikipedia.org/wiki/Eulerian_path

[3] Wikipedia, *Maximum flow problem*,
https ://en.wikipedia.org/wiki/Maximum_flow_problemIntegral_flow_theorem

[4] Wikipedia, *Floyd–Warshall algorithm*,
https ://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm

[5] Wikipedia, *Assignment problem*,
https ://en.wikipedia.org/wiki/Assignment_problem

[6] Wikipedia, *Matching (graph theory)*,
https ://en.wikipedia.org/wiki/Matching_%28graph_theory%29

[7] Wikipedia, *Hungarian algorithm*,
https ://en.wikipedia.org/wiki/Matching_%28graph_theory%29

[8] Wikipedia, *Matching (graph theory)*,
https ://en.wikipedia.org/wiki/Hungarian_algorithm

[9] Wikipedia, *Handshaking lemma*,
https ://en.wikipedia.org/wiki/Handshaking_lemma