Problème du Postier Chinois

Le problème du postier chinois est une problématique concernant les graphes appliquée dans la planification des trajets, et l'optimisation des réseaux de transport. Ce problème pensé en 1962 par Meigu Guan consiste à trouver la boucle la plus courte pour un facteur devant parcourir toutes les rues d'un quartier ou d'une ville au moins une fois avant de retourner à son point de départ.

Présentation du Problème

Considérons un quartier avec plusieurs rues interconnectées représentées sous forme de graphe. Chaque rue est une arête du graphe, et chaque intersection est un nœud. Pour optimiser sa tournée, le facteur doit parcourir toutes les rues au moins une fois, et idéalement, il souhaite minimiser la distance totale parcourue (donc passer exactement une fois par chaque arête), avec les distances des rues représentées par le poids des arêtes du graphe.

Pour la partie technique du problème, le point clé est le concept de cycle eulérien. Un cycle eulérien est un cycle (=une boucle) qui passe par chaque arête d'un graphe exactement une fois et revient au point de départ. Un graphe est dit eulérien s'il possède un cycle eulérien. Pour qu'un cycle eulérien soit présent dans un graphe, cela signifie que le graphe est entièrement connexe que tous les sommets sont de degré pair (= ils ont un nombre de sommets pair)

Méthodes et algorithmes Utilisés

Graphe eulérien ou semi-eulérien ?

Pour commencer le programme, il est nécessaire de savoir à quel type de graphe nous avons affaire : un graphe eulérien, semi-eulérien ou aucun des deux ? (un graphe semi-eulérien contient exactement 2 sommets de degré impair).

En effet, notre résolution du problème ne sera pas la même selon cette réponse :

- Avec un graphe eulérien, nous pourrons directement passer à la recherche du cycle
- Si un graphe semi-eulérien est entré, il faut lui ajouter une arête, en reliant les deux sommets de degré impair. La distance de la nouvelle arête sera donc calculée par l'algorithme de Dijkstra, qui va chercher le plus court chemin entre les sommets à lier.

- Si le graphe n'est pas du tout eulérien, nous ne pouvons pas chercher de cycle pour le postier, car rajouter un grand nombre d'arêtes aurait peu de sens dans ce cadre.

Pour développer au sujet de l'algorithme de Dijkstra, c'est une solution optimisée dû au fait qu'il est efficace sur les graphes pondérés et qu'il nous garantit de trouver le chemin le plus court pour relier les sommets, donc optimise la tournée du postier. Les seuls inconvénients de cet algorithme n'ont pas d'impact sur ce problème : sa complexité ne nous dérange que peu car nous ne l'appelons qu'une fois en tout dans le programme, et le fait qu'il ne traite pas d'arêtes de poids négatif n'est pas un problème non plus car un postier n'emprunte pas de rue de distance négative.

Recherche d’un cycle avec Fleury

Afin de chercher le cycle eulérien présent dans le graphe donné (dont on est maintenant sur du type, car les graphes semi-eulériens sont transformés et les graphes non-eulériens ne sont pas traités), nous avons d’abord choisi d’utiliser l’algorithme de Fleury. Ce programme, qui est une méthode classique pour trouver un cycle eulérien dans un graphe, nous permet donc à première vue de résoudre le problème.

En effet, l’algorithme de Fleury est assez simple à implémenter pour des graphes non pondérés, et demande un peu plus de réflexion dans notre cas. Son point clé est sa prise de décision rapide, car ses seuls critères pour choisir les arêtes qu’ils visitent sont d’éviter de passer par un isthme ou de retourner au point de départ. Ces actions ne sont effectuées que si les arêtes correspondantes sont les seules possibilités. Son point faible est relié à son point clé, car sa prise de décisions ne prédit pas les n coups suivants, et il ne peut revenir en arrière, ce qui fait que cet algorithme a du mal à trouver le cycle eulérien présent dans les graphes assez complexes.

Recherche d’un cycle avec Hierholzer

L'algorithme de Hierholzer comme le précédent, recherche un cycle eulérien en parcourant toutes les arêtes d'un graphe tout en revenant au point de départ.

Dans notre version de l'algorithme, on parcourt les chemins possibles depuis le premier sommet du graphe. Dès que l’on tombe sur un sommet qui n'est pas le sommet de départ qu’il n'a aucune arête disponible et non visitée, alors on revient sur nos pas tant que l’on en est contraint. La manœuvre se répète jusqu'à avoir testé tous les chemins possibles ou avoir trouvé un cycle eulérien. En termes de résultats, cette recherche est efficace, dans le sens où elle à la capacité de trouver un cycle eulérien dans des graphes complexes. Cependant, dû au fait qu’il teste toutes les possibilités jusqu’à trouver la bonne, cet algorithme est plus lent à effectuer sa recherche, tout comme il est plus difficile à implémenter.

Comparaison des recherches de cycle

En termes d’implémentation, l’algorithme de Fleury a un concept bien plus simple, et est donc plus facile à réaliser. Pour ce qui est de l’efficacité des recherches, l’algorithme de Hierholzer garantit une solution optimale, là où celui de Fleury peut se perdre en chemin, notamment sur des graphes plus grands et complexes. L’algorithme de Fleury ne peut donc être une solution que pour des graphes assez basiques, et n’offre pas des capacités suffisantes pour suffire à notre projet. L’algorithme de Hierholzer est donc largement préférable, même si plus difficile à mettre en place, car il nous offre l’assurance de trouver un cycle dans chaque graphe ou il en existe un.

Interface graphique

Les graphes sont des structures complexes et abstraites, mais peuvent être graphiquement représentés. Dans notre cas, nous souhaitons apporter une solution concrète et visuelle au problème du postier chinois.

Afin de rendre notre projet plus accessible, plus engageant et plus concret, la création d'une interface graphique nous offre une solution efficace pour visualiser le graphe, et donc le parcours du postier.

Tout d'abord, nous avons naturellement choisi, dans la continuité du projet, de créer notre interface avec le langage Python. Cependant, le langage de base est très limité concernant les interfaces graphiques, ainsi, nous avons eu à trouver de nouvelles fonctionnalités à l'aide de librairies ou de modules.
Nous avons sélectionné plusieurs librairies / modules, premièrement, Tkinter, cet ajout nous permet de créer et gérer des fenêtres, ainsi que tous les widgets utiles comme des boutons, des labels, un champ de saisie, ou une scrollbar.
Deuxièmement, nous avons fait appelle à NetworkX, cette librairie nous a permis de gérer le graphe à l'intérieur de la fenêtre tkinter, NetworkX s'occupe notamment de la création du graphe , et partiellement de l'affichage de ce dernier, en étant par exemple responsable de la couleur des sommets ou des arêtes, ou de leur taille.