Algorithmes pour la Résolution de Problèmes L3

TP 4 & 5 — *Recherche informée*

Exercice 0. Finir le TP précédent si ce n'est pas déjà fait !

Si vous avez terminé le TP précédent, bravo ! Vous êtes un pro du problème du parcours (ouvert) du cavalier ! Vous remarquerez donc que ce problème est un cas particulier d'un problème plus général : le **problème du chemin Hamiltonien**.

Définition 1. *Soit un graphe connexe $G = (V, E)$, un* ***chemin Hamiltonien*** *est un chemin passant par chaque sommet de V exactement une fois. Le problème de décision du chemin Hamiltonien consiste à déterminer si un tel chemin existe dans G.*

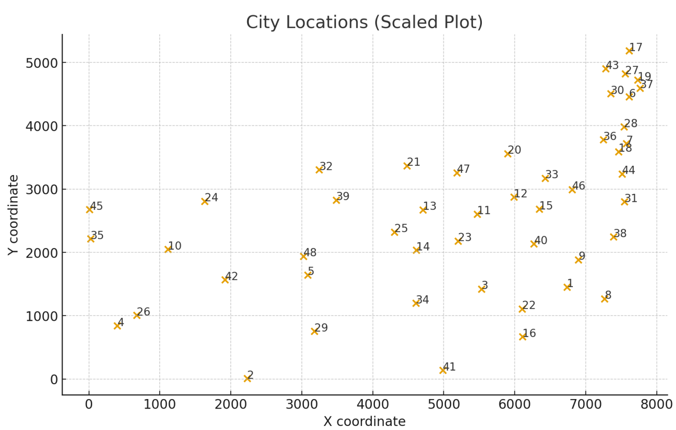
Le problème d'optimisation associé consiste à trouver un tel chemin de coût minimum sur un graphe pondéré. Le problème du chemin Hamiltonien est un problème NP-complet, ce qui signifie qu'aucun algorithme polynomial n'est connu pour le résoudre dans le cas général.

Exercice 1. Un sacré voyage

Vous souhaitez réaliser un roadtrip aux USA en visitant toutes les capitales des états américains. Vous pouvez choisir votre ville de départ et votre ville d'arrivée, mais vous ne pouvez visiter chaque ville qu'une seule fois.

Vous trouverez sur Moodle un fichier `us_capitals.txt` contenant les coordonnées géographiques (x, y) de chaque capitale d'état américain. Ce fichier est structuré de la manière suivante : chaque ligne contient l'identifiant de la ville, suivi de sa latitude et de sa longitude, séparées par des espaces.

Si l'on trace ces coordonnées sur une carte, voici le résultat obtenu :



Par simplicité, nous allons considérer que la distance entre deux villes est la distance euclidienne entre leurs coordonnées géographiques, c'est-à-dire

$$d(u, v) = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}.$$

Question 1. Proposez une modélisation constructive pour ce problème en utilisant les 5 éléments vus en cours.

Question 2. Un algorithme de recherche non-informée sera difficilement efficace pour résoudre ce problème. Le parcours en largeur demandera beaucoup trop de mémoire, et le parcours en profondeur risque de s'égarer dans des solutions non-optimales. De plus, le parcours en profondeur itératif n'aucun intêret ici, car (comme pour le problème du cavalier) $d = D$. Utilisez le parcours en profondeur pour avoir une première solution réalisable. Quel est votre solution et son coût ?

Question 3. Implémentez maintenant l'algorithme A* pour résoudre ce problème. Vous utiliserez une fonction heuristique basée sur l'arbre couvrant de poids minimum (Minimum Spanning Tree) des villes non encore visitées. Un code permettant de calculer le MST est fourni sur Moodle. (Si votre code traine un peu, essayez de réduire la taille du problème en supprimant une partie de villes à visiter).

Le théorème suivant montre qu'en effet, le coût du MST des villes non visitées est une *borne inférieure* du coût pour visiter toutes les villes restantes.

Théorème 1. *Soit P^* une solution optimale du problème du chemin Hamiltonien, et soit T^* une solution optimale du problème du MST. Alors, le coût de T^* est une borne inférieure du coût de P^*.*

Démonstration. Soit $f(P^*)$ le coût de la solution optimale du problème du chemin Hamiltonien, et soit $f(T^*)$ le coût de la solution optimale du problème du MST. Montrons que $f(T^*) \leq f(P^*)$.

Supposons le contraire : $f(T^*) > f(P^*)$. Notons que tout chemin Hamiltonien est aussi un arbre couvrant car tous les sommets du graphe sont visités et un chemin est toujours acyclique. P^* est donc un arbre couvrant. Si jamais $f(T^*) > f(P^*)$, alors T^* n'est pas optimal, ce qui contredit la définition de T^*. Donc, notre supposition est fausse, et on a bien $f(T^*) \leq f(P^*)$. □

Question 4. En utilisant l'arbre couvrant de poids minimum comme borne inférieure, implémentez un algorithme de branch-and-bound pour résoudre ce problème. Proposez une façon de claculer une borne supérieure pour chaque sous-problème. Comparez les performances de cet algorithme avec celles de l'algorithme A*. Lequel des deux algorithmes obtient des meilleures performances ? Justifiez votre réponse.