Devoir Programmation Logique

Algorithmes de recherche pour le problème du voyageur de commerce

I) Le problème du voyageur de commerce

Le problème du voyageur de commerce est un problème d'optimisation qui, étant donné une liste de villes, et des distances entre toutes les paires de villes, détermine un plus court chemin qui visite chaque ville une et une seule fois et qui termine dans la ville de départ[1].

Malgré la simplicité de son énoncé, il s'agit d'un problème d'optimisation pour lequel on ne connaît pas d'algorithme permettant de trouver une solution exacte rapidement dans tous les cas. Diverses approches ont été proposées pour le résoudre: des approches exactes (l'algorithme retourne effectivement le plus court trajet) et des algorithmes d'approximation dans le cas où les arêtes respectent l'inégalité triangulaire.

Une façon de représenter l'ensemble des villes placées sur un plan est par la donnée d'une matrice de coût M_{ij}. Cette matrice stocke la distance entre deux villes i et j, $i \neq j$. Dans ce projet, nous considérerons que la matrice de coût est symétrique, i.e. pour tout i et j tels que $i \neq j$, $M_{ij} = M_{ji}$. La matrice de coût peut représenter une distance euclidienne, reflétant la propriété $M_{ij} + M_{jk} \geq M_{ik}$, il s'agit donc de résoudre le problème du *voyageur de commerce euclidien*. On peut également utiliser une autre représentation du problème, où l'on donne pour chaque ville du problème ses 2 coordonnées dans le plan; il est ensuite facile de calculer la distance "à vol d'oiseau" entre deux villes i et j quelconques.

II) Algorithme de recherche A*

a) Préalables

Dans le cas du problème du voyageur de commerce à n villes, nous vous proposons le codage suivant. Chaque ville est représentée par un entier strictement positif et chaque cycle hamiltonien est une permutation de $[1, 2, \cdots, n]$. Par exemple, pour $n = 8$, un chemin solution est:

$$[4,\ 8,\ 1,\ 5,\ 2,\ 3,\ 6,\ 7].$$

Le coût total d'une permutation σ :

$$f(\sigma) = \sum_{i=1}^{n-1} M_{\sigma(i)\sigma(i+1)} + M_{\sigma(n)\sigma(1)}.$$

Le dernier terme de la somme permettant de revenir au sommet initial. C'est cette fonction que nous cherchons à minimiser.

III) A faire

Vous implanterez un algorithme A^* avec deux fonctions heuristiques de votre choix (vous établirez si vos fonctions heuristiques sont minorantes ou non) et un algorithme heuristique de votre choix (une recherche gloutonne, par exemple) pour résoudre le problème du voyageur de commerce. Vous évaluerez les performances des différents algorithmes implémentés pour un ensemble d'instances avec un nombre croissant de villes. Vous testerez et comparerez vos algorithmes sur des problèmes décrits dans le format TSP (voir

[1] Source : Wikipedia.

par exemple `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/` pour la description du format TSPLIB, et `http://www.math.uwaterloo.ca/tsp/` ou `http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html` pour avoir un ensemble d'instances de problèmes de voyageurs de commerces avec leur solution optimale.

Le devoir se fait en binôme. Vous rendrez avant le **21 décembre minuit** sur l'ENT :

1. un fichier pdf décrivant vos choix d'implémentation pour chacun des algorithmes implémentés et les résultats de vos comparaisons expérimentales. Vous évaluerez à la fois le temps pour obtenir une solution que la qualité de la solution obtenue (i.e. la distance entre le coût de la solution optimale et la solution que vos algorithmes retournent)

2. votre code prolog commenté et avec des requêtes de test pour les principaux prédicats

Une soutenance aura lieu le 19 janvier 2018 où vous présenterez votre code sur une sélection d'instances que vous choisirez ou qui vous seront proposées.