Intelligence Artificielle : projet 2021/2022

Description du problème

On se propose dans ce projet de résoudre des instances du problème du *voyageur de commerce*.

Dans ce problème, étant donnés un ensemble de villes et des distances entre toutes les paires de villes (on peut se représenter un graphe complet non-orienté et avec des poids sur les arêtes), on cherche à trouver un plus court chemin qui visite chaque ville exactement une fois et qui se termine dans la ville initiale. On parlera de *circuit Hamiltonien*. Les villes sont représentées par des points dans un espace à deux dimensions. On utilisera à titre illustratif l'exemple de la Figure 1.

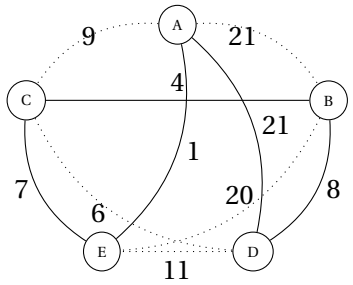

Figure 1: Un exemple d'une instance du voyageur de commerce. Les poids affectés aux arêtes sont indiqués le long de celles ci. Un exemple de circuit Hamiltonien (A,E,C,B,D) est donné par les arêtes qui ne sont pas en pointillés.

Recherche informée

Dans la première partie du projet, vous modéliserez le problème comme une recherche dans un espace d'état.

- L'état correspond à la ville actuelle et à un ensemble de villes déjà visitées. L'état initiale correspond à la situation où l'agent se trouve dans la ville

de départ et aucune ville n'a été visitée. Dans l'état final, l'agent se trouve dans la ville de départ et toutes les villes ont été visitées.

- Dans un état, les actions possibles sont de visiter une ville parmi celles pas encore visitées ; le coût correspond à la distance entre la dernière ville visitée et celle du nouvel état.

Tout d'abord, formaliser le problème avec les notations utilisées dans le cours (fonctions ACTIONS, RESULT, etc). Ensuite vous allez utiliser l'algorithme A^* pour trouver le parcours optimal.

Dans l'arbre de recherche, pour chaque noeud n :

- $g(n)$ est le coût total du trajet effectué à partir de la racine.
- $h(n)$ est le valeur d'une heuristique.

  On vous conseille d'utiliser comme heuristique la valeur de l'arbre couvrant minimal, *Minimum Spanning Tree (MST)* en anglais, en considerant les villes non encore visitées et la ville associée au noeud n.

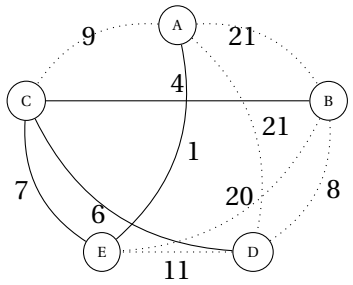

Figure 2: L'arbre couvrant minimale du graphe de l'exemple.

Minimum Spanning Tree (MST) On appelle *arbre couvrant minimal* dans un graphe valué, un sous-ensemble d'arêtes formant un arbre (un graphe acyclic et connexe) sur l'ensemble des sommets du graphe initial, et tel que la somme des poids de ces arêtes soit minimale. L'*algorithme de Prim* est un algorithme glouton qui calcule un arbre couvrant minimal dans un graphe connexe valué et non orienté. L'algorithme de Prim consiste à faire croître un arbre $(\mathcal{V},\mathcal{A})$, où $\mathcal{V}$ et $\mathcal{A}$ sont les ensembles des sommets et des arêtes de l'arbre. L'algorithme part d'un arbre contenant un seul sommet, et agrandit cet arbre à chaque itération en prenant l'arête de coût minimum reliant un sommet de l'arbre à un sommet hors de l'arbre. Dans l'exemple de la Figure 1, l'es arbres construits aux différentes

itérations sont $(\{A\},\emptyset)$, $(\{A,E\},\{\{A,E\}\})$, $(\{A,E,C\},\{\{A,E\},\{E,C\}\})$, $(\{A,E,C,B\},\{\{A,E\},\{E,C\},\{C,B\}\})$, et $(\{A,E,C,B,D\},\{\{A,E\},\{E,C\},\{C,B\},\{C,D\}\})$. L'arbre ansi trouvé est montré en Figure 2.

Considérons un état consistant en un chemin partiel. Pour estimer le coût minimal pour terminer le tour à partir de ce chemin partiel, on considere le sous-graphe contenant les villes non encore visitées, la ville actuelle, et la ville initiale (où se terminera le cycle, et on calcule le poids de l'arbre couvrant minimal. Est que cette heuristique est admissible? Est-il cohérente?

Recherche locale

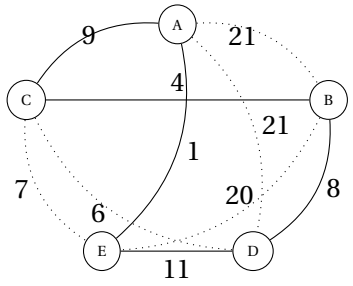

Figure 3: Resultat de l'application de 2-opt à partir du circuit Hamiltonien de l'exemple de la Figure 1, en enlevant les arêtes (D,A) et (E,C) et rajoutant les arêtes (D,E) et (A,C).

Dans cette deuxième approche on utilisera une représentation du problème où chaque état consiste en un circuit Hamiltonien, i.e., une solution réalisable (cf Figure 1).

Étant donné un circuit Hamiltonien C, contenant deux arêtes (a,b) et (c,d) et dans lequel les sommets apparaissent dans l'ordre a,b,c, et d. On obtient alors le circuit Hamiltonien C' à partir de C en enlevant les arêtes (a,b) et (c,d) et en rajoutant les arêtes (a,c) et (b,d); voire en Figure 3 un exemple.

Cette opération nommée 2-opt définit le voisinage de chaque état. Dans l'optique de la recherche locale, on désire uniquement une "bonne" solution, pas necessairement la solution optimale.

Travail demandé

Le travail s'effectue *en binôme ou en trinôme*. Il s'agit de réaliser un projet informatique pour tester les méthodes de recherche pour résoudre ces instances :

- **Recherche informé**: Developpez l'algorithme A^* pour ce problème comme suggéré ci-dessus.
- **Recherche locale**: au moins un algorithme entre *Hill-climbing search* et ses variantes (*Stochastic hill climbing*, *First Choice hill climbing*, *Random-restart hill climbing*), *Simulated annealing*, *Local beam search*, *Genetic algorithms*.

💣 Si vous etes en trinôme, vous devez delopper au minimum trois algorithmes au total (pour exemple, vous pouvez choisir d'implementer l'algorithme IDA^*).

Quelques remarques :

- On ne demande pas la réalisation d'une interface graphique. Cependant, on doit pouvoir tester votre programme facilement et de manière intuitive.
- Vous pouvez utiliser les outils que vous voulez et le langage de programmation que vous désirez. Cependant, il est préférable de demander l'autorisation si vous souhaitez utiliser un langage "exotique".

Vous devez proposer un protocole expérimental pour tester ces méthodes (generer aleatoirement des istances et/ou tester avec des graphes représentant des réseaux routiers), mettre en place ces expérimentations, puis réaliser un rapport détaillant votre travail. Vous devrez y détailler l'analyse du problème, la mise en place des algorithmes et les résultats de vos expérimentations numériques : génération des instances, calibrage des algorithmes, comparaison entre les deux approches (temps de calcul, cout total des solutions trouvées).

Vous montrerez en détail l'exécution des algorithmes sur des exemples d'instances.

- Pour A^* montrez la taille maximale de l'ensemble *frontier* pendant l'execution.
- Pour les algorithmes de recherche local vous allez montrer le taux d'amélioration par rapport à la solution initiale, et l'écart à l'optimum ou à des points de références.

Pour les deux approches vous detaillez la taille de graphe que vous pouvez gérer.

💣 Si vous utilisez des ressources (e.g., mémoires, papiers de recherche, autres cours, code), vous devez y faire référence dans votre rapport.