TP : Introduction à l'Intelligence Artificielle
2020/2021

Objectifs

L'objectif de ces premiers travaux pratiques est de générer et de résoudre de manière optimale tous les taquins 3x3.

Résoudre un taquin (dit initial) revient à trouver le nombre de déplacements *minimal*, de la case vide, nécessaire pour passer du taquin initial vers un taquin but (décrit ci-dessous). Chaque déplacement, de la case vide, coûte une unité.

Le but de ces travaux pratique est aussi de comparer les performances de plusieurs algorithmes et tester expérimentalement l'admissibilité de fonctions heuristiques.

La réalisation de ces premier travaux pratiques est une étape nécessaire pour résoudre efficacement et pleinement le problème du taquin 4x4 en utilisant l'algorithme IDA* avec des heuristiques avancées.

Langages de programmation

Il est préférable d'utiliser le langage C (le langage Python est accepté).

Taquin but

Nous utiliserons le taquin but suivant :

Table 1: Example d'un taquin 3x3 but

	1	2
3	4	5
6	7	8

Pour ce TP, encore un peu de pré-traitements et implémentation des algorithmes de recherches aveugles

Je rappelle que vous avez déjà traité (en plus d'autres fonctions) des fonctions qui vérifient i) si un taquin 3x3 est un taquin but ou non et ii) si un taquin 3x3 est solvable ou non.

Voici d'autres fonctions de pré-traitements à programmer :

1. Ecrire une fonction qui prend un taquin initial 3x3 en paramètre et calcule ses taquins successeurs. Un taquin successeur est obtenu depuis le taquin initial en déplaçant (quand cela est possible) la case vide vers la gauche, le haut, la droite, le bas respectivement.

2. Ecrire une fonction, appelée h_malplacees, qui prend un taquin initial 3x3 en paramètre et calcule le nombre de cases mal placées (qui ne sont pas à la bonne position dans le taquin final).

Pour les algorithmes de recherche non-informés, voici le travail de programmation qui est demandé :

1. Ecrire une fonction qui prend en paramètre un taquin 3x3 "initial" et qui implémente l'algorithme de recherche en coût uniforme. On suppose qu'un déplacement de la case "blanc" vous coûte une unité.
2. Ecrire une fonction qui prend en paramètre un taquin 3x3 "initial" et un entier positif l et qui implémente l'algorithme de recherche en profondeur limitée.

Pour l'évaluation de vos algorithmes, générer un échantillon de taquins initiaux. La sortie de chaque algorithme est un fichier où chaque ligne est composée de cinq champs (une ligne = résultats obtenus sur un taquin initial de l'échantillon généré) :

- le premier champs contient le taquin initial (les 9 chiffres du taquins, ligne par ligne. Le vide sera codé par le chiffre 0)
- le deuxième champs contient le nombre de déplacements nécessaires pour la résolution du taquin, et
- le troisième champs contient une chaîne de caractères, composée de l'un des quatre caractères : "G" (Gauche), "D" (Droite), "H" (Haut) et "B" (Bas). Cette chaîne de caractères représente la séquence des déplacements de la case vide de l'état initial vers l'état but. Cette séquence permet de vérifier que votre solution est correcte (voir TP précédent).

Remarques :

- Il est important de réfléchir à une bonne structure de données et à l'algorithmique avant de se lancer dans la programmation.
- Les travaux pratiques doivent-être réalisés par vous-même (il n'est pas autorisé de reprendre un code existant; peu importe sa provenance).