Universités d'Aix-Marseille I & III

Licence Informatique – Intelligence Artificielle

TD n°3 : Graphes d'états

1. Les missionnaires et les cannibales

Trois missionnaires et trois cannibales doivent traverser une rivière à l'aide d'une embarcation qui ne contient que 2 places. A aucun moment les cannibales ne doivent être en surnombre sur une rive où se trouvent des missionnaires (sinon ils mangent les missionnaires).

1.1. Modélisez le problème de manière à obtenir le graphe d'états le plus petit possible.

1.2. Représentez le graphe d'états en entier.

1.3. Effectuez une recherche en largeur de la solution : représentez l'arbre de recherche.

1.4. Même question pour une recherche en profondeur d'abord.

2. Le problème des seaux d'eau

On dispose d'une fontaine et de deux seaux de contenance respective 3 litres et 5 litres. On peut soit remplir (entièrement) un seau à la fontaine, soit le vider (entièrement) soit transvaser une partie de (ou tout) son contenu dans l'autre seau de manière à ce que ce dernier soit plein. Au départ, les deux seaux sont vides et le but est d'obtenir 4 litres d'eau (dans le seau de 5 litres).

Modélisez le problème. Déterminez une heuristique d'évaluation de la proximité d'un état par rapport à un état objectif. Résolvez le problème de 3 manières :

- en effectuant une recherche en profondeur d'abord.
- en effectuant une recherche en largeur.
- en effectuant une recherche du meilleur d'abord.

3. Le problème de Langford

Voici une version simplifiée du problème de Langford. Il s'agit de trouver comment aligner six entiers : deux 1, deux 2 et deux 3, de telle manière qu'il y ait un entier entre les deux 1, deux entiers entre les deux 2 et trois entiers entre les deux 3. Une solution est par exemple : 312132.

On représente un état par une suite des 6 entiers. On passe d'un état à un autre en permutant deux entiers qui se jouxtent (on a donc 5 voisins au maximum). Résoudre le problème grâce à l'algorithme de recherche du meilleur d'abord, en partant de l'état $e_0 = 123123$ et en utilisant l'heuristique suivante : choisir l'état e qui minimise $h(e) = |\delta_1 - 1| + |\delta_2 - 2| + |\delta_3 - 3|$, où δ_i représente le nombre d'entiers séparant les deux entiers i.

4. L'algorithme de recherche Iterative Deepening

Iterative Deepening (ID) est un algorithme intermédiaire entre la recherche en profondeur d'abord (Depth First Search, DFS) et la recherche en largeur (Breadth First Search, BFS). Il consiste à

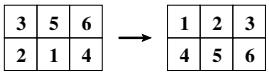
faire une succession de DFS limitée à une profondeur p. Au début, $p = 1$. Tant qu'une solution n'a pas été trouvée avec la profondeur p grâce à DFS, on recommence une DFS en incrémentant p.

On considère un graphe d'états de largeur L dont les sommets ont une arité maximale A.

Comparez DFS, BFS et ID en terme en nombre de nœuds générés et en terme de complexité spatiale. Peut-on en conclure qu'un de ces algorithmes est meilleur que les autres ?

5. Problème

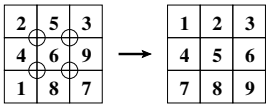
5.1. On considère le problème représenté par la figure ci-dessous. Des chiffres sont placés initialement dans une grille comme indiqué à gauche. Il faut les déplacer de manière à obtenir la position finale indiquée à droite.



Quel que soit l'état courant, il n'y a que deux mouvements possibles. Chacun consiste en une rotation dans le sens horaire des chiffres situés dans un carré 2×2, comme indiqué sur la figure ci-dessous.

Résolvez le problème en utilisant la recherche arborescente en largeur.

5.2. On considère maintenant une version plus grande du problème sur un carré 3×3. Il s'agit cette fois de passer de l'état initial indiqué par la partie gauche de la figure ci-dessous à l'état final indiqué dans la partie droite de cette figure.



Cette fois-ci, les mouvements possibles sont de même nature (une rotation dans le sens horaire dans un carré 2×2) mais au nombre de quatre : rotation dans le carré supérieur gauche, supérieur droit, inférieur gauche ou inférieur droit.

Utilisez la recherche "meilleur d'abord" pour résoudre ce problème. On rappelle que cette recherche consiste à sélectionner l'état qui a la meilleure valeur de fonction heuristique parmi tous les états ouverts (= qui n'ont pas encore généré leurs fils) de l'arbre. *On utilisera la fonction heuristique suivante : nombre de chiffres bien placés.* On représentera l'arbre de recherche en numérotant les états dans l'ordre dans lequel ils ont été générés. On pourra encercler les chiffres bien placés en chaque état afin d'y visualiser la valeur de la fonction heuristique.

5.3. Dans la question précédente, aurait-on généré moins, autant ou plus d'états si on avait utilisé une recherche en largeur ? Expliquez pourquoi.

5.4. Même question si on avait utilisé une recherche en profondeur d'abord avec la même heuristique pour le choix du fils.