Compte Rendu TP 01: Algorithmes de recherche

Sorbonne Université 2018/2019

Master sciences pour l'ingenieur

ZICHI Djamel *3803757*

SOUALHI Takieddine 3872967

Abstrait—Ce Compte rendu présente une étude des algorithmes de recherche informés et non informé et leurs performances, dans la suite de ce document nous allons présenter et étudier les algorithmes Breadth-First-Search, Dpeth-First-Search et l'algorithme A*,

Keywords—Breadth First, Depth First, A*(Star).

I. INTRODUCTION

Dans le domaine de l'intelligence Artificielle, il existent de nombreux types d'Agents, nous allons nous interessé par un certain type appellé Agent Orienté but, cet agent utlise la représentation atomique pour modéliser son environnement.

Avant de d'aborder sur les algorithmes de recherche, nous devons definir ce qui est un probleme, un probleme est défini par: son état initiale, une fonction successeur definissant les états succédant à un état donné, test d'atteint de but ou un état but, et le cout des actions exécutés, l'ensemble des états et l'ensemble des actions définissent ce qu'on appelle un graphe d'états, les élement décrites ci dessus définissement un probleme

II. ALGORITHMES DE RECHERCHE

Un algorithme de recherche c'est l'ensemble des actions qui vont explorer un graphe d'état pour trouver un chemin entre un état initiale et un état but, géneralement il existe deux categories d'algorithmes de recherches informés et non informés.

```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
  initialize the search tree using the initial state of problem
  loop do
    if there are no candidates for expansion then return failure
    choose a leaf node for expansion according to strategy
    if the node contains a goal state then return the corresponding solution
    else expand the node and add the resulting nodes to the search tree
```

Figure.1 forme générale d'un algorithme de recherche

A. Algorithmes non-informés:

c'est les algorithmes qui ne possèdent aucune connaissance sur le problème a part le graphe d'états et le cout des de dévelopement, Dans notre cas on va étudier les algorithmes de recherche en largeur (Breadth First Search) et celui de recherche en profondeur (Depth First Search).

le premier fonctionne comme une pile Fifo, il dévélope toujours le premier neoud dans la queue, donc il effectue un balayage par largeur.

le deuxieme fonctionne comme une pile Lifo, il dévelope toujours le dernier neouds dans la queue, donc il effectue un balayage par profondeur.

B. Algorithmes informés:

Cette catégorie possède une fonction d'évaluation pour choisir le nœud a développer.

C. Propriétés d'un algorithme de recherche:

Un algorithme de recherche est caractérisé par :

- Complexité en espace: Nombre maximale des nœuds stockés en mémoire.
- Complexité en temps : c'est le temps de calcul.
- Complétude: Si une solution existe, est elle toujours trouvée ?.
- Optimalité: la solution trouvé est-elle de moindre cout ? .

III. PARITE THEORIQUE DU TP :

QUESTION 1: Complexité en espace

Pour choisir l'algorithme non informés le plus adéquat pour chaque arbre, on on compare les complexités des deux algorithmes.

Comme on a vue en cours l'algorithme de recherche en profondeur est plus adéquant en terme de complexité(moin complexe en espace et en mémoire)

Mais ça dépend toujours de la structure de l'arbre et les facteurs qui la caractérise et notamment : la profondeur de la solution, la profondeur maximale de l'espace de recherche et le facteur de branchement maximale de l'arbre.

Donc visuellement on peut estimer que l'algorithme de recherche en profondeur est plus adéquat .

QUESTION 02: Représentations des données

A. *Matrice d'incidence du graphe (figure 3)*

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

B. *Structures:*

Ce graphe peut etre représenté par des listes (listes d'incidences) .

le temps d'accés pour les deux structures est presque le même et il égale a O(1). Mais si le graphe est complexe en espace, il serait mieux d'utiliser une matrice pour le représenter

QUESTION 04&05: algorithmes BFS et DFS

Un exemple du résultat obtenu est représenté dans la figure ci-dessous: (Voir code)

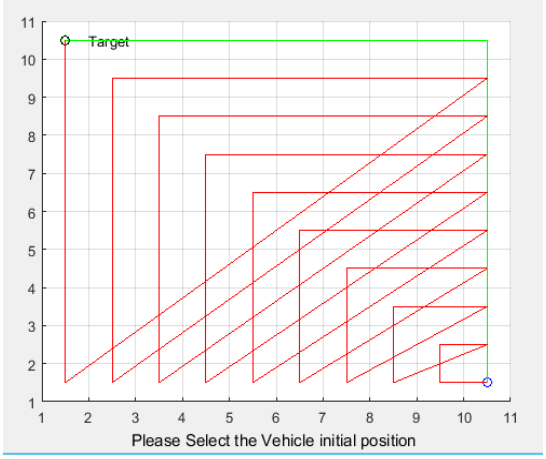

Figure.2 Chemins trouvé par BFS et DFS

Dans cet exemple on Remarque que le chemin trouvé par l'algorthime de recherché en profondeur (en vert) était plus court que celui trouvé par l'algorithme de recherché en largeur (en rouge). Dans autres essayes on a eu le cas inverse. Donc l'efficacité de ces algorithmes depend du point initial et point but de l'arbre et sa complexité en espace.

On note aussi que aucun de des deux algorithms a pu trouvé le chemin le plus court.

QUESTION 06: algorithme A Star:*

la figure ci-dessous représente un exemple du résultat obtenu:

Figure.3 Chemin trouvé par l'Algorithme A*

Par rapport au deux algorithms precedents, on Remarque que l'algorithme A* était plus efficace en terme du chemin, il a pu trouvé le chemin le plus court.

QUESTION 07: Influence de l'heuristique :

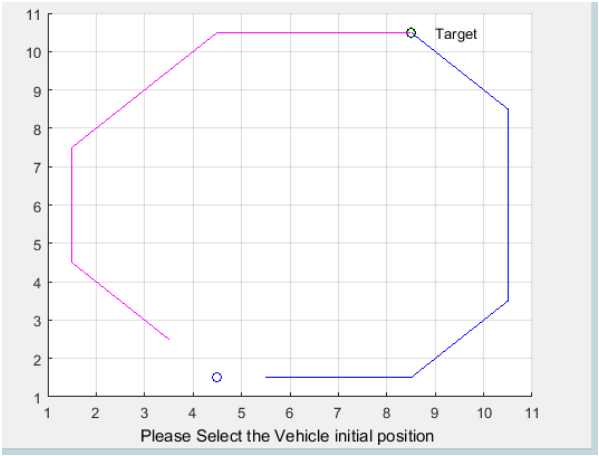

Figure.4 h(n) de Manhattan(Bleu) vs h(n) euclidienne(Rose)

Dans ce cas lorsque on compare les deux heuristiques utlisés, on trouve que : l'heuhistique de Manhattan n'est pas optimale en termes du longeur du chemin. Le chemin trouvé par l'heuristique euclidienne est plus court. Donc on conclut que l'optimalité de l'algorithme A* depend sur le choix de l'heuristique.

QUESTION 08: Etude d'un cout variable

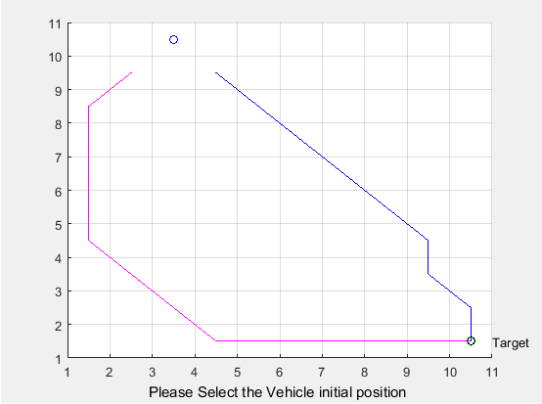

Figure.5 Chemins trouvés pour un cout variable.

Pour un cout variable le chemin trouvé par la function A* n'est plus le plus court sur le plan comme on a déja trouvé dans la QUESTION 6. Donc la variation du cout est un facteur important qui caractérise l'optimalité de l'algorithme A*.

QUESTION 09: Etude de la complexitée en temps

```
>> T_Astar_Euc
T_Astar_Euc =
    0.4592
>> T_Astar_Man
T_Astar_Man =
    0.4272
```

Figure.6 Temps estimé pour chaque algorithme A*.

En utilisant les fonctions fournit pour estimer le temps pris par chaque fonction pour faire les calculs, on a obtenu les résultats presents sur la Figure.5:

On Conclut que en terme de complexité en Temps, A.Star dote de l'heurisitique de Manhattan est le plus rapide, donc c'est l'algorithme le plus efficace,

Note : si le déplacement sur le plan s'effectue uniquement selons les direction (UP , DOWN , LEFT , RIGHT), dans ce cas il trouvera pas toujours le chemin le plus court (Vue en cours)

QUESTION 10: Etude de la complexité en espace,

```
>> T1paroourusD
T1paroourusD =
    10
>> T1paroourusB
T1paroourusB =
    36
```

```
>> T2paroourusB
T2paroourusB =
    36
>> T2paroourusD
T2paroourusD =
     9
```

[4] Figure.7 Complexité en espace estimé pour BFS et DFS

En pratique il n'existe pas de methods directes pour determiner la complexité en epsace d'un algorithme, dans notre cas, on s'appuie sur le nombre des noeuds parcourus par chaque algorithme. La Figure.7 montre les résultart obtenus :

- Pour Tree1 l'algorithme de recherche en Profondeur est plus efficace
- Pour Tree2 l'algorithme de recherche en Profondeur est plus efficace

On Remarque que dans les deux cas, BFS parcourt tous les noeuds des graphe, donc il est utilize beaucoup plus de mémoire , c-à-d il est plus complexe en espace que DFS. Et cela confirme notre hypothese dans la question 1.
Pour la complexité en espace

IV. CONCLUSION

Lors de ce TP, on a pu mettre en pratique les connaissances acquises en cours sur la résolution des problemes et les algorithmes de recherche. Et on était capable de noté plusieurs points importantes :

- Quoi que ce soit le probleme, l'algorithme A* est toujours celui qui fournit la solution optimale en prennant en considération la minimalité de la complexité en espace et en temps.

- On peut utiliser l'heuristique pour controler le comportement de l'algorithme A*.

- un Algorithme A* doté d'une heuristic constante il est equivalent a un algorithme de Djikstra.

- un Algorithme A* doté d'une h(n) qui est égale au cout exacte de passer d'un état n a l'état n+1, rendre A* Parfait, il va directement aller au état but sans avoir tester les autres états

- un Algorithme A* avec h(n)>>g(n) est equivalent à un algorithme Best-First.

- l'algorithme A* est un algorithme complet, c-à-d si une solution est existe, elle sera trouvée

- Pour les algorithmes non-informés (BFS et DFS), principalement DFS est le meilleur mais le choix dépends toujours de la complexité en espace du graphe.

REFERENCES

[1] http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.htmlJ. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] P. Norvig and S. J. Russel, "Artificial Intelligence : A modern Aproach".

[3] Cours de M. Raja Chatil