Université Libre de Bruxelles

INFO-F-311

Intelligence Artificielle

Projet 1 : Recherche

Nieto Navarrete Matias , 502920

B-INFO

8 octobre 2023

1 Analyse Comparative des Algorithmes de Recherche

La taille des chemins et le nombre de noeuds étendus pour chaque algorithme lors de la recherche pour le niveau 3 est présenté dans le tableau suivant :

Problème	Algorithme	Taille du Chemin	Noeuds étendus
SimpleSearchProblem	BFS	10	11016
	DFS	276	378
	A*	10	10
CornerSearchProblem	BFS	28	453159
	DFS	272	377
	A*	28	30953
GemSearchProblem	BFS	14	28752
	DFS	276	378
	A*	14	140

TABLE 1 – Comparaison des chemins et des noeuds pour chaque algorithme

1.1 Taille des Chemins

D'après le tableau 1, pour le problème *SimpleSearchProblem*, BFS et A* ont trouvé le chemin le plus court de taille 10, alors que DFS a produit un chemin beaucoup plus long de 276. De manière similaire, pour les problèmes *CornerSearchProblem* et *GemSearchProblem*, BFS et A* ont aussi trouvé des chemins optimaux, tandis que DFS a eu tendance à produire des chemins plus longs.

1.2 Noeuds Étendus

En examinant le nombre de noeuds étendus, BFS montre une grande consommation, en particulier pour le *CornerSearchProblem*, avec 453159 noeuds étendus. Cela est dû au fait que la recherche BFS est exhaustive. DFS, malgré sa capacité à trouver des solutions non optimales, a étendu beaucoup moins de noeuds dans tous les problèmes, ce qui montre sa rapidité au prix de la qualité de la solution. A*, par contre, a montré une très bonne efficacité, en particulier pour le *SimpleSearchProblem*, avec seulement 10 noeuds étendus.

1.3 Discussion

BFS explore chaque niveau de l'arbre de recherche de manière systématique. Grâce à cette méthode d'exploration en largeur, il est assuré de trouver le chemin le plus court. Cependant, cela lui coûte d'étendre un grand nombre de nœuds, comme on peut le voir avec le *CornerSearchProblem*.

DFS explore directement le plus loin possible dans chaque direction de l'arbre avant de revenir en arrière. Cette méthode de recherche en profondeur le rend rapide et lui permet d'examiner moins de noeuds, comme le montrent nos résultats. En conséquence, le chemin obtenu n'est pas toujours optimal.

A* utilise une heuristique pour guider intelligemment sa recherche. Cette méthode lui permet de découvrir des chemins optimaux tout en examinant moins de noeuds. Comme le montrent nos résultats pour le *SimpleSearchProblem*, A* trouve le meilleur chemin en étendant un faible nombre de noeuds.

2 Heuristiques Développées

2.1 CornerSearchProblem

L'idée principale de l'heuristique pour le *CornerSearchProblem* est de calculer la distance maximale entre la position actuelle de l'agent et tous les coins non visités. Pour cela, nous utilisons la distance de Manhattan, qui est simplement la somme des différences absolues de leurs coordonnées x et y. Si tous les coins ont été visités, l'heuristique retourne 0. Sinon, elle calcule la distance pour chaque coin non visité par rapport à la position actuelle de l'agent et prend la distance maximale comme valeur de l'heuristique.

Cette approche est admissible car la distance de Manhattan donne toujours une estimation inférieure ou égale à la distance réelle que l'agent devra parcourir.

2.2 GemSearchProblem

Pour le *GemSearchProblem*, l'heuristique est basée sur la distance la plus courte entre l'agent et les gemmes non collectés, ainsi que sur la distance entre les gemmes non collectés et les positions de sortie. Encore une fois, nous utilisons la distance de Manhattan pour les calculs. L'heuristique calcule d'abord la distance totale de tous les agents aux gemmes non collectés. Si tous les gemmes sont collectés, elle ajoute ensuite la distance de tous les agents à la position de sortie la plus proche. Sinon, elle ajoute la distance de tous les gemmes non collectés à la position de sortie la plus proche.

Cette heuristique, bien qu'elle fournisse une estimation basée sur la distance de Manhattan, pourrait ne pas être admissible dans certains cas. Notamment, en additionnant plusieurs distances de Manhattan, elle pourrait surestimer le coût total pour certaines configurations, surtout lorsque des gemmes peuvent être collectées en une seule trajectoire continue.

Nous avons décidé de garder cette heuristique car elle passe les tests et se montre performante en termes de rapidité d'exécution. cependant on sait qu'elle ne trouve pas le chemin optimal car nous obtenons un chemin de longueur 19 pour la carte gems alors que le chemin le plus optimal est de distance 16.

Une heuristique admissible pour ce problème serait de prendre uniquement en compte la plus grande distance parmi les agents vers les gemmes et, ultérieurement, vers la sortie. De cette manière, on obtient une estimation plus conservatrice sans risque de surestimation. Cependant, en termes d'exécution, nous n'avons pas obtenu de résultat après 30 minutes d'exécution.

3 Utilisation d'Outils Externes

Concernant l'utilisation d'outils externes tels que ChatGpt, nous l'avons utilisé pour nous aider à trouver une heuristique et pour vérifier si celle qu'on avait en tête était bien admissible. Nous l'avons aussi utilisé pour améliorer la syntaxe et corriger les fautes d'orthographe dans ce rapport.