INFO-F311 - Project - Search

Luca Palmisano - 000516721

October 2023

1 Niveau 3 - Comparaisons DFS, BFS et A*

L'algorithme **DFS** a trouvé une solution dans laquelle le chemin est de longueur 276. Les algorithme **BFS** et **A*** ont tous deux trouvé une solution dans laquelle le chemin est de longueur 10.

L'algorithme **DFS** a trouvé une solution mais qui est loin d'être idéale. Le niveau 3 est relativement grand et ne possède pas beaucoup d'obstacles, il y donc beaucoup de possibilités différentes. La solution donnée par **BFS** et **A*** est identique et meilleure que celle de **DFS**. La différence entre ces 2 algorithmes réside dans le nombre de nœuds étendues au cours de la recherche.

Algorithme	Taille chemin	Nœuds etendu
DFS	276	280
BFS	10	7149
A*	10	953

On remarque que **DFS** a trouvé un chemin de taille 276, mais il n'a pas étendu beaucoup de nœuds par rapport aux autres algorithmes. On voit aussi directement la différence entre **BFS** et **A***: malgré le fait que leurs solutions soit identiques, **A*** a été beaucoup plus efficace pour la trouver.

Cette différence entre **BFS** et **A*** est dû au fait que **A*** va utiliser une heuristique. Cette heuristique estime à quelle point on se rapproche du but. Dans le cas du `SimpleSearchProblem`, l'heuristique est la somme des distance (*manhattan distance*) entre chaque agent et la sortie disponible la plus proche. Contrairement a **BFS**, **A*** va étendre le nœud qui minimise cette heuristique. **BFS** est "bête" par rapport à **A***, il va simplement tester toutes les possibilités jusqu'à trouver la solution.

2 Heuristiques

CornerSearchProblem:
Pour ce problème de recherche, j'ai comparé 3 heuristiques. La première sert de référence:

- Calculer la distance (*manhattan*) entre l'agent et la sortie
- Calculer la distance (*manhattan*) entre:
  - l'agent et la sortie : si l'agent est pas passé par tous les coins

– l'agent et le coin le plus proche : sinon

- Calculer la distance (*manhattan*) entre l'agent et la sortie multiplier par le nombre de coins non visité

Heuristique	Chemin	Nœuds etendu
Distance avec sortie	51	1341
Distance avec coins et/ou sortie	51	1198
Distance avec sortie * coins non visité	53	568

Dans le cas de la première heuristique, celle-ci n'est pas très efficace car elle ne prend pas en compte les coins à visiter. Elle se base uniquement sur la distance entre l'agent et la sortie.

On remarque que dans la deuxième heuristique, celle-ci est légèrement plus efficace. Le comportement de l'agent que le peut observer grâce à cette heuristique est qu'il va s'approcher du coin le plus proche et ainsi de suite, jusqu'à ce qu'il n'ait plus de coins a visiter et dans ce cas la il se dirigera vers la sortie. C'est comme si on déplaçait la sortie a chaque fois qu'il se rendait dans un coin.

La troisième heuristique trouve un chemin légèrement plus long mais est beaucoup plus efficace, elle a divisé par 2 le nombre de nœuds étendus. Le comportement de l'agent que l'on peut observer grâce à cette heuristique est qu'il va se débarrasser des coins le plus rapidement possible tout en restant proche de la sortie.

GemSearchProblem:
Pour ce problème de recherche, j'ai comparé 2 heuristiques:

- Le nombre de gemmes qui n'ont pas encore été ramassé
- Calculer la distance (*manhattan*) entre l'agent et la sortie multiplier par le nombre de gemmes qui n'ont pas encore été ramassé

Heuristique	Chemin	Nœuds etendu
Gemmes non ramassé	22	666290
Distance avec sortie * gemmes non-ramassé	27	1711185

On remarque que la distance dans ce problème-ci a un effet négatif sur les résultats. Je n'ai pas trouvé d'heuristique intéressantes mais j'ai tout de même remarqué qu'en jouant avec le coût associé à des actions on peut légèrement optimiser la première heuristique. Par exemple en incrémentant le coût si un agent s'est déplacé en en le décrémentant si une gemme a été ramassée.