2024-2025

INFO-F311

INFO-F311 - Intelligence artificielle - Recherche

BERTHION Antoine

Tom Leanaerts

Table des matières

1 Rapport

1.1 Introduction

L'objectif de ce rapport est d'analyser et de comparer l'efficacité des algorithmes DFS (Depth-First Search), BFS (Breadth-First Search) et A* (A-star) à travers une série de tests conçus pour évaluer leurs forces et faiblesses respectives. Ces algorithmes seront évalués dans différents contextes de recherche sur des cartes simulées, mettant en lumière leurs comportements distincts dans des scénarios variés.

1.2 Méthodologie

Afin d'étudier les performances de chaque algorithme, plusieurs cartes complexes ont été générées, chacune représentant des défis spécifiques en termes de résolution de problème de recherche. Les algorithmes seront appliqués à ces cartes afin d'identifier leurs capacités respectives à atteindre un objectif donné, tout en prenant en compte plusieurs critères de performance.

1.3 Expériences

Nous commencerons cette section par une brève présentation des 3 cartes proposées ;

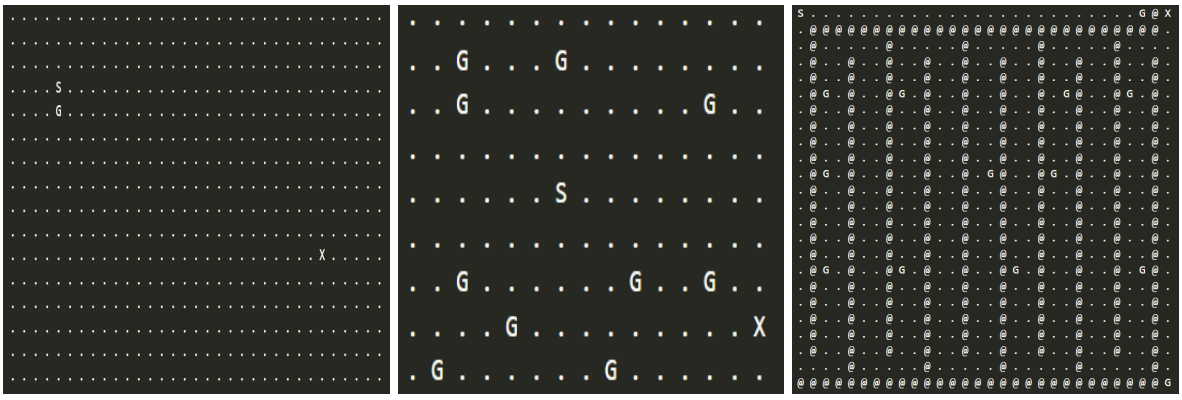

FIGURE 1.1 : EMPTYROOM, FULLGEMS et ZIGZAG.

La carte "EMPTYROOM" est une carte ne contenant quasiment aucun obstacle, la carte "FULLGEMS" est une carte contenant un grand nombre de gemmes et pour finir la carte "ZIGZAG" est une carte contenant un grand "zigzag" ainsi que 2 culs-de-sac, à droite et en haut.
Dans un second temps, nous créeons un GemProblem basé sur ces 3 cartes, que nous résolvons à l'aide de : DFS, BFS ainsi que A*. Voici les graphiques associés à ces résolutions des problèmes de gemme créés ;

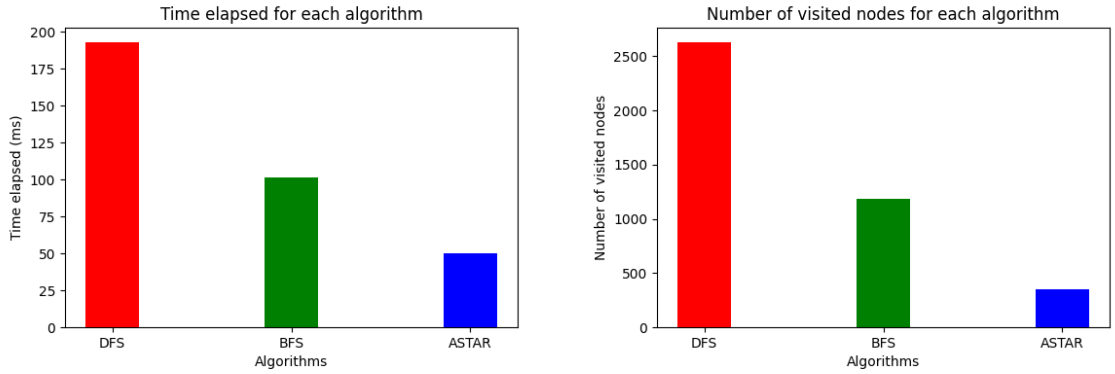

FIGURE 1.2 : Solution pour EMPTYROOM

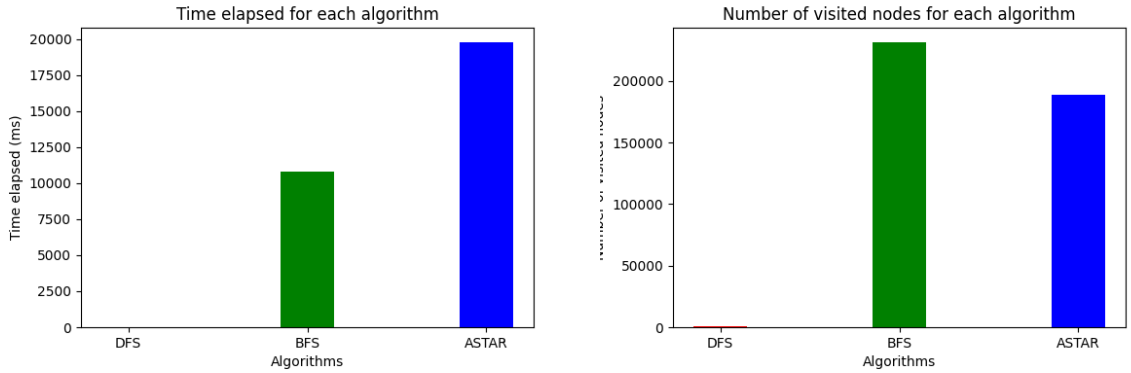

FIGURE 1.3 : Solution pour FULLGEMS

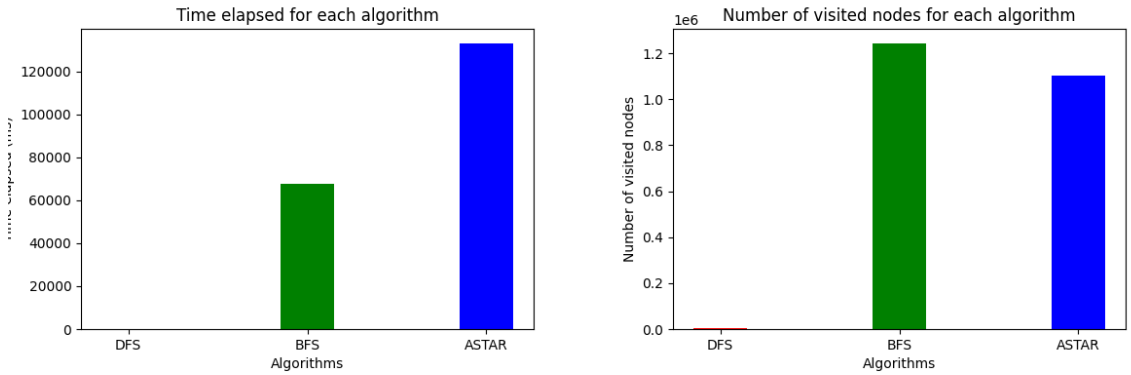

FIGURE 1.4 : Solution pour ZIGZAG

1.4 Discussion

Nous allons maintenant procéder à une analyse comparative des forces et faiblesses des algorithmes étudiés en nous appuyant sur les résultats obtenus pour différentes configurations de cartes.

1.4.1 EMPTYROOM

Cette carte met en évidence les limitations de DFS par rapport à BFS et A*. DFS a nécessité un total de 2622 étapes, contre seulement 36 étapes pour BFS et A*. La raison en est que DFS explore souvent des branches

dans leur intégralité avant de revenir en arrière, ce qui peut entraîner un parcours inefficace. En revanche, A* optimise le nombre de nœuds visités grâce à son heuristique, tandis que BFS explore uniformément jusqu'à trouver la sortie.
Sur cette carte, DFS procède de manière peu efficiente, zigzaguant à travers le graphe, tandis que BFS et A* trouvent rapidement la sortie en parcourant les nœuds les plus proches de manière plus systématique.

1.4.2 FULLGEMS

Sur cette carte riche en gemmes, DFS surpasse A* et BFS. Les algorithmes perfectionnistes comme A* et BFS cherchent des solutions optimales, ce qui augmente leur temps de calcul, surtout en raison de la collecte des gemmes. A* doit constamment évaluer les distances, et BFS explore exhaustivement, ce qui ralentit leur exécution.
DFS, moins préoccupé par l'optimalité, parcourt le graphe sans chercher à minimiser les déplacements, ramassant des gemmes en chemin sans viser une collecte complète. Ce comportement aléatoire permet à DFS de trouver une solution relativement rapidement, bien que non optimale. Il reste cependant important de préciser que la solution de DFS est extrêmement mauvaise, comptabilisant 419 étapes contre 37 pour la solution optimale.

1.4.3 ZIGZAG

Sur la carte 'ZIGZAG', DFS se montre également plus performant que BFS et A*. Les couloirs en zigzag et les culs-de-sac posent problème à BFS, qui explore systématiquement les extrémités de ces culs-de-sac pour récupérer les gemmes, augmentant ainsi le temps d'exécution. A* est aussi affecté par son heuristique, qui le pousse à explorer des chemins non optimaux en raison de la complexité des obstacles, notamment les murs.
En revanche, DFS suit naturellement les chemins en zigzag jusqu'à leur terme avant de revenir en arrière, ce qui s'avère plus efficace dans cette configuration. Néanmoins, bien que plus rapide à exécuter, la solution trouvée par DFS est en moyenne deux fois plus longue que celles obtenues par BFS et A*, ce qui est acceptable compte tenu du temps d'exécution de A* et BFS.

1.5 LLM

Dans cette courte partie, nous discuterons de l'utilisation des LLM dans le cadre du projet. Aucun LLM n'a été utilisé pour l'aspect implémentation ainsi que la compréhension du projet. Cependant, ce rapport à été corrigé du point de vue de la syntaxe et de la grammaire par DeepL ainsi qu'un modèle GPT. Notons tout de même qu'aucune des informations du rapport n'a été produite par une autre personne que l'auteur.

1.6 Conclusions

En résumé, chaque algorithme présente des forces et des faiblesses spécifiques, en fonction des objectifs de la recherche et de la configuration des cartes.

L'algorithme Depth-First Search (DFS) se montre particulièrement efficace dans des contextes où l'optimalité de la solution n'est pas primordiale. Son comportement dit "erratique" peut être avantageux dans des situations où

la recherche d'une solution optimale exigerait un coût computationnel élevé. Cependant, DFS présente des limitations significatives dans des environnements nécessitant une évaluation heuristique pour trouver une solution rapidement. En effet, il a tendance à explorer de manière exhaustive des branches entières sans nécessairement se rapprocher de la solution, ce qui le rend inadapté pour des scénarios où la sortie ne se trouve pas dans les premières branches explorées.

En revanche, les algorithmes Breadth-First Search (BFS) et A* sont bien mieux adaptés à la recherche de solutions optimales. Le BFS explore systématiquement l'espace de recherche, garantissant ainsi une solution optimale, tandis que A* intègre une fonction heuristique qui lui permet d'orienter efficacement son exploration vers l'objectif. Toutefois, ces algorithmes sont mis à l'épreuve lorsqu'une contrainte de temps stricte s'applique. Ceci est particulièrement vrai pour l'algorithme A*, dont la performance dépend de l'efficacité de la fonction heuristique utilisée. Cette dernière, bien qu'elle permette d'éviter l'exploration inutile de certaines branches, peut également introduire un surcoût en temps d'exécution.

Dans des environnements complexes, l'algorithme A* se distingue comme un choix judicieux, car il parvient à éviter des chemins non pertinents grâce à l'utilisation de son heuristique. Le BFS, bien qu'efficace dans la recherche systématique, devient moins pertinent face à A* dans des situations où la complexité du chemin requiert une exploration plus intelligente et ciblée.