ULB UNIVERSITÉ LIBRE DE BRUXELLES

INFO-F311 - Projet d'IA 1

Recherche

Auteur:
Manuel Rocca - 000596086

Professeurs:
Tom Lenaerts
Assistants:
Axel Abels
Martin Colot
Yannick Molinghen
Pascal Tribel

Année académique 2025-2026

Contents

1 Introduction

Durant cette année, au cours d'Intelligence Artificielle, IA pour les intimes, nous serons amenés à réaliser une série de projet permettant, comme chaque année, l'application de la matière vue. Nous commençons fort avec ce premier projet de "Recherche" ayant pour but d'appliquer les algorithmes de recherche de plus court chemin, à savoir *BFS*, *DFS* et finalement, *A** à l'aide de diverses structures de données comme le *Stack* ou la *PriorityQueue*.

Afin d'appliquer ces connaissances, il nous faut bien entendu un cadre. Celui-ci nous est proposé directement sous forme d'une librairie nommées sobrement "*Laser Learning Environment*", LLE pour les personnes friandes de termes succints. Elle nous permet d'avoir une grille en deux dimensions peuplée d'éléments divers comme des agents, qui font le déplacement et des objectifs, c'est-à-dire un état à atteindre pour résoudre le problème donné. Ces objectifs se manifestent sous forme de gemmes à collecter, de coins à visiter et de sorties à trouver (chaque objectif consiste en un problème à part entière).

2 Mode opératoire

2.1 Création des cartes

Lors de la création de nos cartes personnalisées, notre but principal était de créer des environnements permettant des scénario d'exécution des algorithmes uniques et distincts les un des autres. Ceci nous permet de bien nous rendre compte du comportement de chaque algorihme de recherche de chemin dans chaque cas de manière claire.

- *EASY MAP*: Une carte totalement vide avec une gemme pour se rendre compte du cas le plus direct.
- *ONE PATH MAP*: Cette carte ne possède qu'un seul chemin valide, permettant de se rendre compte de la robustesse des algorithmes, de leur temps d'exécution.
- *MANY GEMS MAP*: Ici nous testons tous les paramètres sur une carte ouverte.
- *COMPLEX MAP*: Pareil que pour la carte précédente mais avec plus d'obstacles pour mieux se rendre compte de certains aller-retours inutiles (en particulier dans le cas du DFS).
- *IMPOSSIBLE MAP*: Cette dernière carte permet de vérifier la justesse des algorithmes de recherche dans le cas où il n'existe pas de chemin valide (voir s'il n'existe pas d'abberation).

2.2 Heuristiques

Il nous était demandé d'implémenter une heuristique par problème mais nous ne l'avons pas fait. Afin d'expliquer cette décision, commençons par définir l'heuristique pour l'algorihme A*.

Définition 1 (Heuristique A*) *La fonction heuristique $h(n)$ fournit une estimation du coût entre le nœud actuel et le nœud cible, agissant comme une "supposition éclairée" de l'algorithme sur le chemin restant à parcourir.*[1]

[1]Source: https://www.datacamp.com/fr/tutorial/a-star-algorithm

En d'autres termes, l'algorithme A* est l'équivalent de l'algorihme de Dijkstra avec une heuristique qui "dirige" la trajectoire, elle oriente la recherche pour être plus efficace.

Nous avons trois problèmes différents. L'*Exit Problem*, le *Gem Problem* et le *Corner Problem*. L'heuristique que nous avons implémenté ne s'adresse initialement qu'au problème de sortie (la distance de Manhattan est calculée par rapport à la sortie/aux sorties). Nous voulions d'abord bêtement implémenter deux heuristiques supplémentaires pour chacun des problèmes mais, après réflexion, nous sommes arrivés à la conclusion suivante: Le *Gem Problem* et le *Corner Problem* ne font qu'ajouter des conditions à l'état de sortie valide. En d'autres termes, ils ajoutent des étapes avant d'atteindre la sortie. Or, le but d'A* est de minimiser le coût total pour arriver au but final. Donc, A* trouvera quand même le meilleur chemin en rammassant toutes les gemmes/en passant par tous les coins.

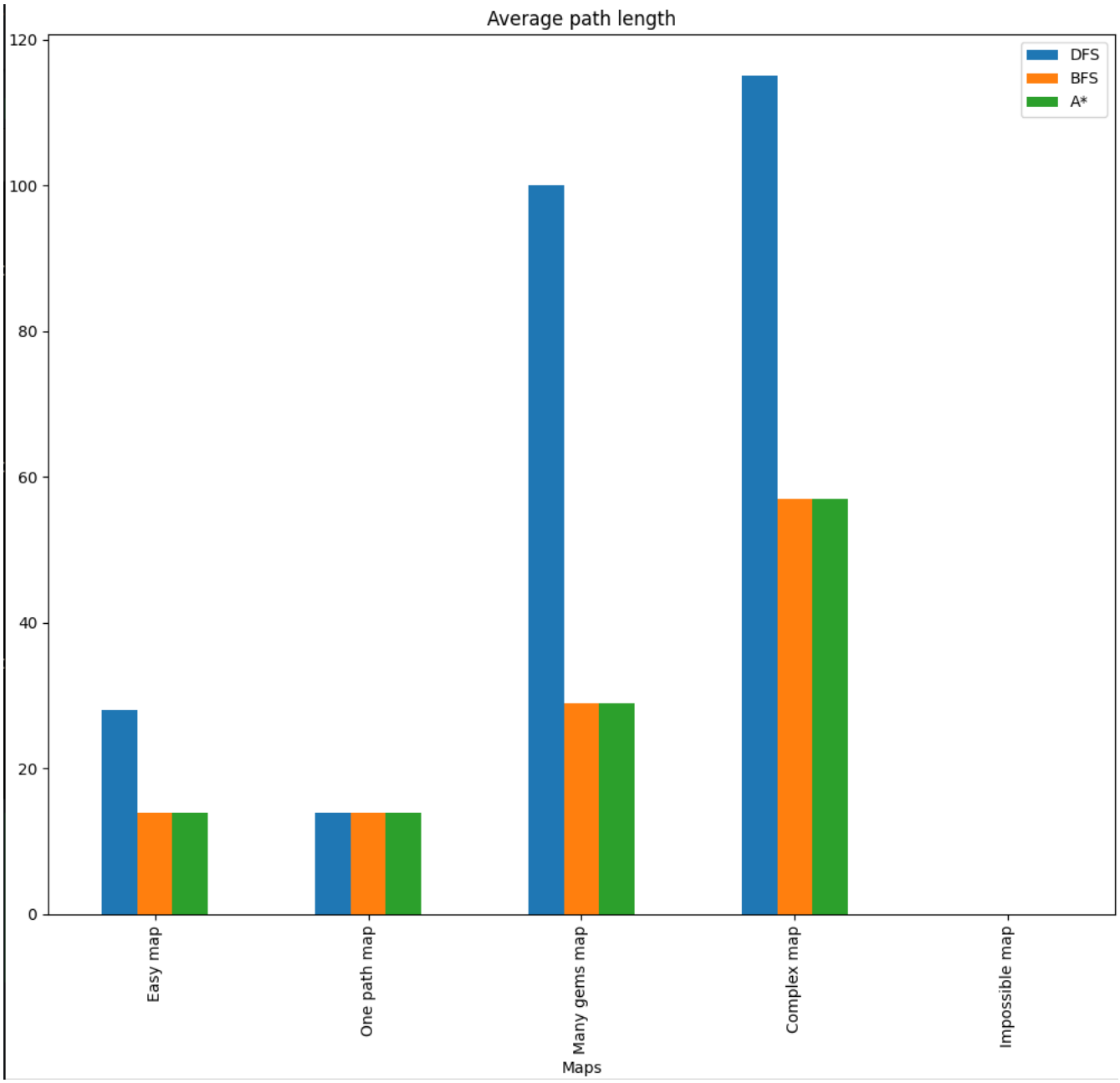
Un défaut de notre implémentation que nous souhaitons tout de même relever est le fait que A* visitera plus de nœuds pour trouver ce chemin optimal. En effet, l'heuristique dirige toujours vers la sortie. Le scénario d'exécution logique de l'algorithme dans notre cas est le suivant: il va d'abord chercher la sortie, la trouver et se rendre compte que ce n'est pas un état final valide et donc continuer l'exploration. Notre défaut vient donc du fait que, si nous avions une heuristique adaptée aux gemmes/coins, A* se dirigerait d'abord vers les gemmes/coins à la place de sortie et visiterait moins de nœuds mais nous n'avons pas implémenté de telles heuristiques.

3 Comparaison des algorithmes sur le GemProblem

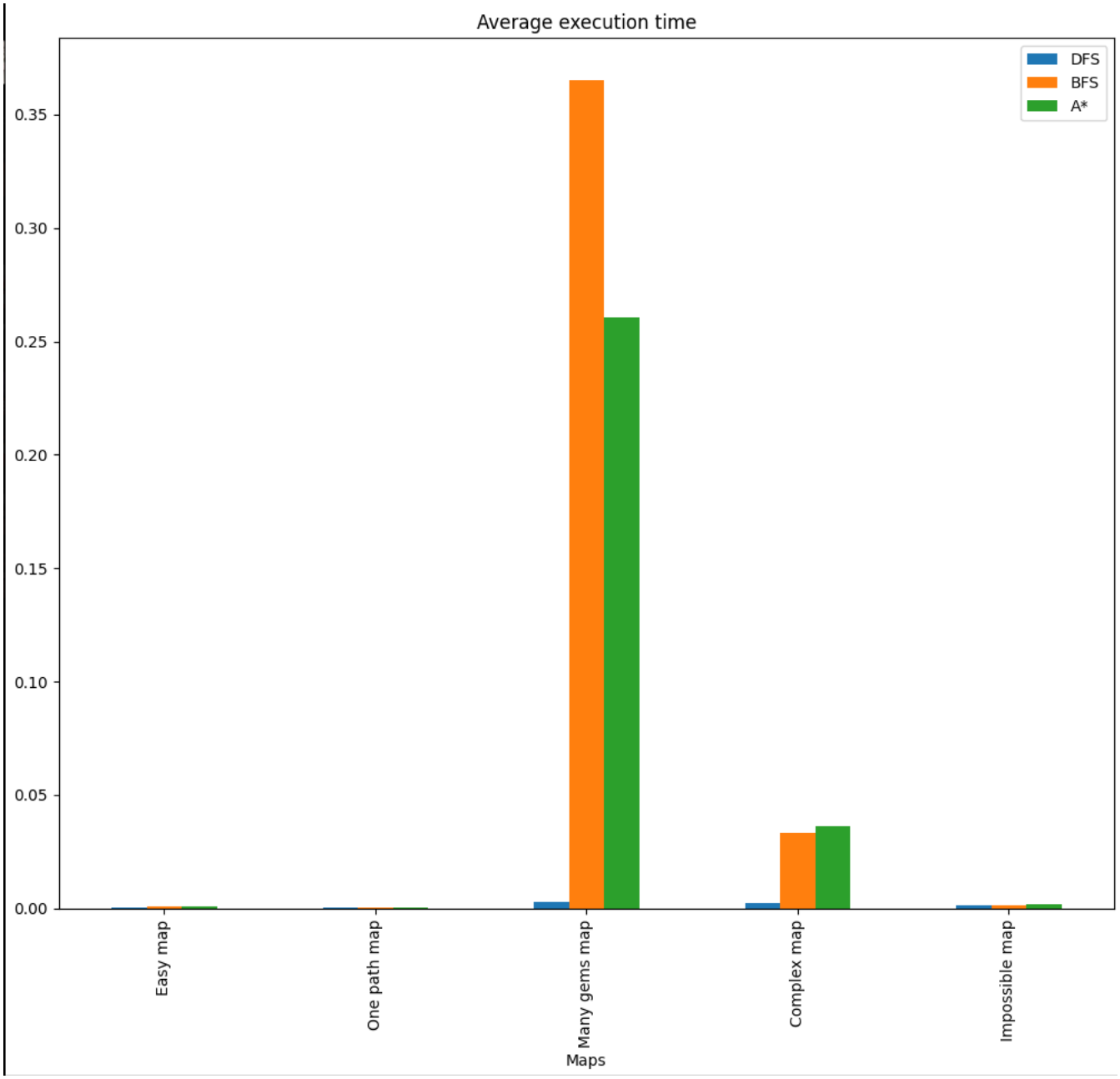
Afin de tester l'implémentation de nos algorithmes, il nous était demandé d'implémenter quatres cartes personnalisées et pertinentes pour l'objet de nos tests. Nous avons pris la liberté de reprendre celles de notre collègue étudiant Ethan Van Ruykenveelde et d'en modifier certaines. Pour chaque expérience, nous avons effectué cinquante itérations pour, d'une part, vérifier l'absence d'aberrations et, d'autre part, avoir une moyenne pertinente (surtout dans le cas de la durée moyenne d'exécution car, normalement, pour un problème donné constant, les algorithmes de recherches trouvent toujours la même solution).

3.1 Graphes

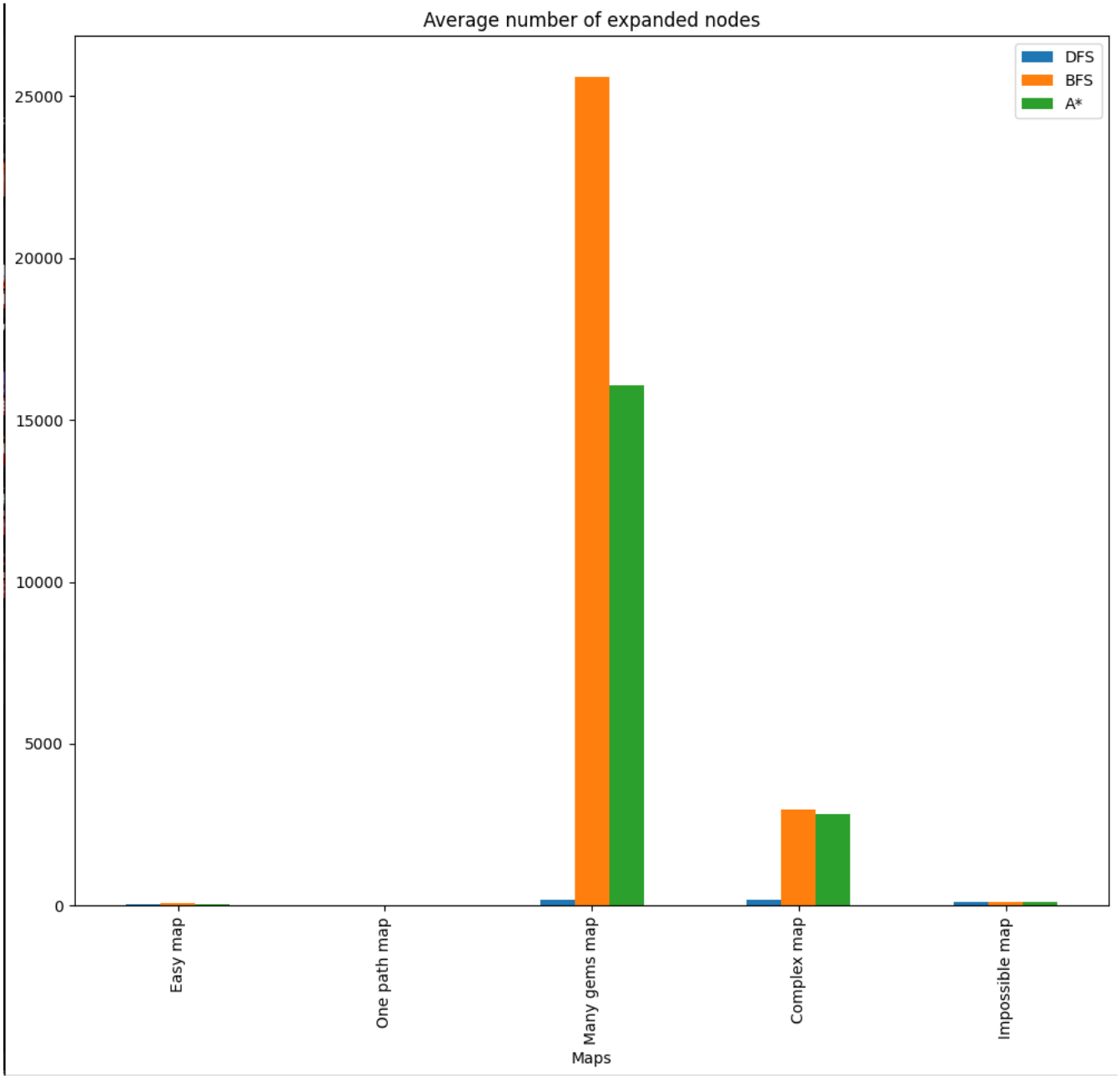
3.1.1 Longueur du chemin



3.1.2 Durée moyenne d'exécution



3.1.3 Nombre de nœuds visités



3.2 Discussion des résultats

Sur base des résultats, il est clair qu'à part au niveau du temps d'exécution, l'algorithme *Depth First Search* (DFS) est moins efficace. Comme son nom l'indique, cet algorithme explore d'abord en profondeur et renvoie la première solution de chemin valide, sans étendre beaucoup de nœuds (d'où le faible temps d'exécution).

À l'inverse l'algorithme *Breadth First Search*, qui cherche donc en largeur, pratique une recherche exhaustive et renvoie un chemin optimal. Similairement, A* fait usage d'une recherche dite "informée" au travers de la fonction heuristique et permet également de trouver un chemin optimal. Cette optimalité se fait au prix d'une durée d'exécution supérieure et, surtout, un nombre de nœuds étendus nettement plus important (particulièrement visible sur la *MANY GEMS MAP* (3.1.3)).

4 Utilisation de l'IA

En cette fin d'année 2025, un étudiant qui dit qu'il n'a pas utilisé l'IA pour faire son projet est essentiellement l'équivalent d'un développeur qui prétend ne jamais avoir eu de segmentation fault en C. Autrement dit, des foutaises.

En tant qu'étudiant, l'IA peut être un outil très puissant mais également très dangereux pour l'apprentissage. Le but de l'université est d'apprendre. Si l'intelligence artificielle est mal utilisée, dans le sens où l'étudiant lui demande les réponses, de tout faire sans comprendre, autant ne pas étudier à l'université.

Notre point de vue est donc celui-ci. Il faut faire usage l'IA de manière pertinente, c'est-à-dire pour comprendre ce que l'on fait (pas bêtement copier-coller du code[2]), demander des explications (idéalement après avoir lu les supports de cours, recherché en ligne, etc...).

Dans le cadre de ce projet, nous avons donc utilisé l'IA pour les choses suivantes: comprendre des résultats d'exécution plus facilement lorsqu'un test ne passe pas après plusieurs tentatives, essayer de comprendre le code initialement donné (la compréhension des types était particulièrement éprouvante car la librairie utilisée ne précise pas toujours les types dans la documentation et, de manière générale, car python est un langage très peu typé, même si l'on peut les préciser dans certains cas). Nous l'avons également utilisé pour le rapport. Après avoir fait nos recherches et rédigé une explication (concernant l'heuristique et les résultats) nous avons demandé à l'IA d'analyser les conclusions pour lesquelles nous n'étions pas totalement certains (notamment pour la justification de l'exécution sans heuristique).

Avec du recul, nous nous rendu compte que cela reste tout de même beaucoup. Cependant, comme expliqué au début de cette section, nous nous sommes tenus à faire un usage de l'IA dans un but explicatif et non un usage de type esclave.

Pour conclure cette section, nous trouvons que permettre l'usage de l'IA aux étudiants et quelque chose de plutôt pertinent car ce n'est pas un outil qui semble être voué à disparaître. Au contraire, c'est une chose qui se développe sans cesse et apprendre à élever son usage au travers d'outils modernes est une super bonne chose. De plus, les étudiants sont (dans la majorité des cas) des adultes responsables. Ce que nous voulons dire par là est que, si un étudiant fait usage de l'IA de manière irréfléchie et abusive, c'est, en un sens, son problème car il n'apprendra absolument rien.

5 Conclusion

Afin de conclure ce rapport ainsi que ce projet, revenons brièvement sur notre travail. Ce projet de recherche nous a permis de mettre la manière vue en pratique. Nous avons vu le concept d'environnement, d'agent, de recherche de solution optimale (ou non) à l'aide de divers outils (ici les algorihmes DFS, BFS, A*). Nous avons étudié et comparé les résultats afin d'arriver à des conclusions pertinentes.

Ce projet compose donc une bonne introduction aux techniques d'intelligence d'artificielle basées sur la recherche dans les graphes.

[2]Copier-coller du code pouvait déjà être fait avant les LLMs à partir de forums, de sites internet ou n'importe quelles autres sources. Comprendre ce que l'on fait dans le cadre universitaire n'est donc en soit pas une nouvelle chose mais ne pert aucunement en importance avec l'avènement des LLMs.