Aide à la décision et intelligence artificielle

Kiyan Gelinaud, 22100872

November 2023

1 Introduction

Ce document PDF a pour but de décrire les démonstrations réalisées ainsi que les choix d'implémentations effectués pour le fil rouge du cours Aide à la décision et intelligence artificielle

2 Implémentations

2.1 BlocksWorld

J'ai tout d'abord choisi d'implémenter les variables comme étant simplement du type Variable ou BooleanVariable créé dans les TPs en mettant un nom du type "onb", du quel je récupére b en utilisant la méthode replace("on","") dans un parseInt(), ce qui me semblais la meilleur solution.

Ensuite j'ai choisi l'implémentations suivante pour les différentes instance de blocksworld.

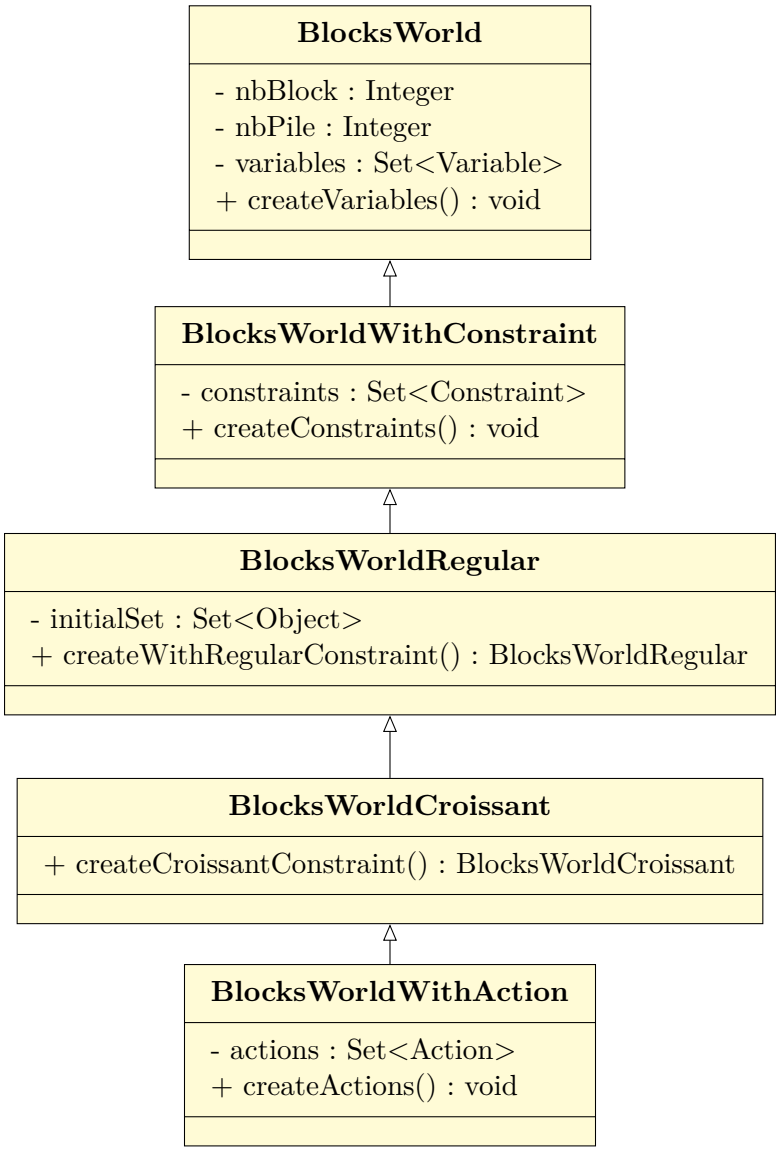

Figure 1 – Diagramme BlocksWorld

Le fait que BlocksWorldRegular et BlocksWorldCroissant aient chacun une fonction renvoyant une instance d'eux même est pour pouvoir créer des blocksworld ayant un seul des types de contraintes ou ayant les deux, je ne pense pas que ce soit le meilleur choix mais c'est celui qui m'ai venu a l'esprit et je la pense plutôt optimal.

2.2 Heuristiques

J'ai décidé de créer une classe abstraite BlocksWorldHeuristic qui serait implémenté par mes deux heuristique cette dernière ayant juste un constructeur demandant un blocksWorld de type BlocksWorld et son accesseur. comme ceci :

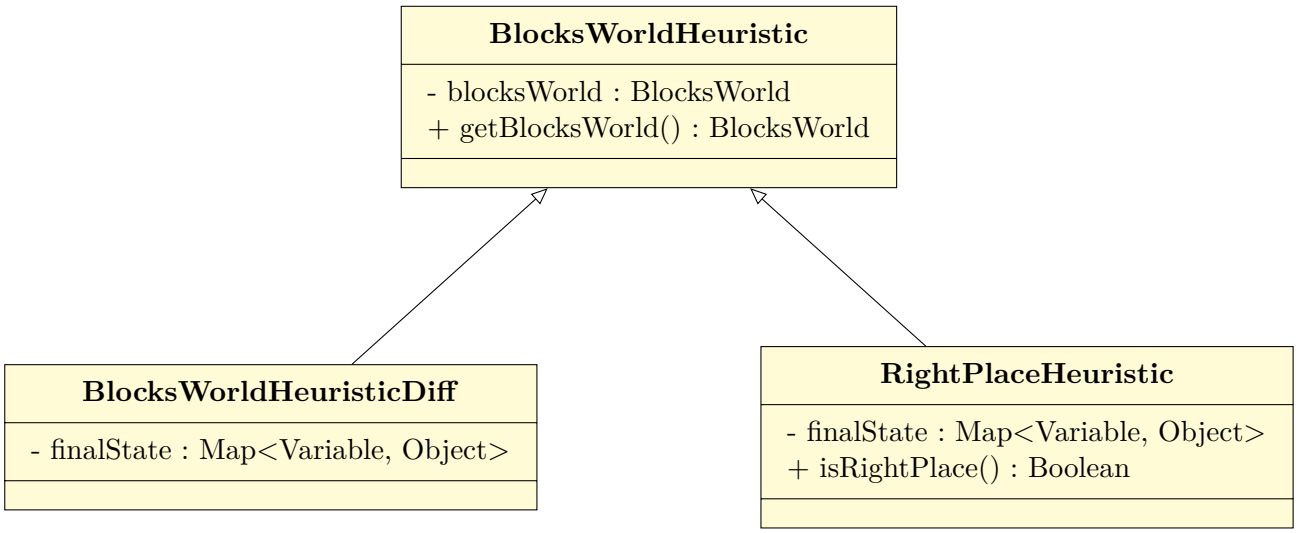

FIGURE 2 – Diagramme Heuristic

L'heuristique Diff, de différence, calcule le nombre de variables de l'état actuel différèrent de l'état final. Tandis que RightPlace elle calcule le nombre de blocks mal placé en utilisant isRightPlace qui est un méthode récursive permettant de dire qu'un block est mal placé si il est sur un block mal placé.

2.3 Goal

Une nouvelle classe de type Goal été pour moi nécessaire pour permettre des instance partielles ce qui était impossible avec BasicGoal créé en TP, j'ai donc créé une classe BlocksworldGoal qui ne regarde que si un block est sur le bon block ou la bonne pile pour que celui ci soit satisfait par un état.

2.4 Database

J'ai aussi choisi de créé une nouvelle classe appelé BlocksWorldDatabase qui étend BooleanDatabase me permettant d'initialisé dedans les variables et une methode statique getInstance(List<List<Integer> > state) permettant de changer un état en List<List<Integer> > en un Set<BooleanVariables>.

3 Les Exécutables

3.1 ModellingExecutable

ModellingExecutable est l'exécutable permettant de tester la modélisation des variables aux contraintes croissantes, trois configuration sont créé pour tester quand le blocksworld n'est que croissant, quand il n'est que régulier et quand il satisfait ces deux derniers.

3.2 PlanningExecutable

PlanningExecutable est l'exécutable permettant de tester la generation de plan par les algorithme tel que dijkstra ou A* avec une configuration créé manuellement et partiellement pour le but, en commentaire est le code permettant de tester des génération aléatoire d'état initial et final. Sont en commentaire les parties permettant de tester dfs, bfs et dijkstra car prenant trop de temps et n'étant pas intéressant mais libre a vous de les dé-commenter, seul A* avec les deux heuristiques vues précédemment sont lancé originalement.

3.3 CPExecutable

CPExecutable est l'exécutable permettant de lancer la résolution avec BacktrackSolver, MACSolver et MACSolver heuristique fait durant les TPs pour les configurations demandé cet à dire, régulière, croissante et les deux.

3.4 DataminingExecutable

DatamingExecutable est l'éxecutable permettant de lancer une extraction de données sur dix mille instance de 100 blocks et 35 piles le temps pris pour executer l'extraction de Apriori est de environ 3min dans mon cas.

3.5 Executable

Executable est un executable permettant de tester à peu prés tout en utilisant des Scanner permettant de choisir par exemple la configuration en plaçant les blocks nous même dans les piles à partir de l'invite de commandes, tout est expliqué a chaque étape.