Monde des bloc (fil rouge)

Introduction:

Le Monde des Blocs est un exemple classique en intelligence artificielle utilisé pour illustrer les algorithmes de planification. Il s'agit d'un monde composé de blocs empilés dans des piles, où l'objectif est de trouver une séquence de déplacements pour transformer une configuration initiale en une configuration.

Dans le cadre de notre projet (fil rouge), l'objectif est de modéliser le monde des blocs à l'aide de variables et de contraintes, de concevoir des algorithmes capables de planifier des séquences d'actions pour atteindre un état final donné, et d'extraire des connaissances à partir de bases de données générées à partir des configurations d'états.

Ce projet se structure en quatre grandes parties :

1/Variables et contraintes
2/Planification
3/Problèmes de satisfaction de contraintes
4/Extraction de connaissances

1/Variables et contraintes :

Modélisation du monde des blocs à travers des variables et des contraintes permettant de décrire ses configurations de manière précise.

Dans cette partie on a utilisé 6 classe:

- **BlocsWorld.java** :générer les variables pour les blocs et les piles en fonction des paramètres (nombre de blocs et de piles).

- **BlocsWorldConstraint.java:**construire les contraintes binaires assurant la validité des configurations
- **RegularityConstraints.java :**Cette classe a pour but de créer une configuration régulière en maintenant un écart constant entre chaque deux blocs pour éviter l' effet circulaire .
- **CroissanteConstraint.java:** Cette methode cree des contraintes pour s'assurer qu'un bloc ne peut pas être posé que sur des blocs de numéro plus petit ou directement sur la table
- **TestRegulier.java:** classe executable.
- **TestCroissante.java**:classe executable.
- **TestBlocsWorldConstraint.java:**classe executable

2/Planification:

- **ActionBlocsWorld.java:**Génère les actions possibles pour déplacer des blocs dans un monde structuré. Fournit un ensemble d'actions avec préconditions et effets.
- **HeuristicBlocs.java:**Le but de cette heuristique est d'estimer la distance entre l'état actuel et l'état cible. Elle évalue cette distance en comptant le nombre de blocs qui ne sont pas encore positionnés à leur emplacement final.
- **PlannerTest.java**:classe executable.

3 /Problèmes de satisfaction de contraintes:

- **SolverTestRegulier.java**:dans cette classe on teste les solveurs déjà implémentés dans le tp3 avec des contraintes régulières.
- SolverTestCroissant.java :dans cette classe on teste les solveurs déjà implémentés dans le tp3 avec des contraintes croissantes.

4 /Extraction de connaissances:

- **BlocsDataBase.java:**crée et gère des variables booléennes pour représenter les états et les relations des blocs dans un monde de blocs.