Projet d'Apprentissage par Renforcement

Jeu *Corridor* (Quoridor simplifié)

Master 2 – Parcours Intelligence Artificielle
Université de Bordeaux

1. Objectif du projet

Le but de ce projet est de concevoir un agent intelligent capable de jouer au jeu **Corridor** (version simplifiée du jeu *Quoridor*) en utilisant les méthodes vues en cours d'Apprentissage par Renforcement (RL).

Vous travaillerez à partir d'une **implémentation fournie du jeu** (fichier `corridor.py`) et devrez développer vos propres agents.L'accent sera mis sur :

- la formulation du problème comme un **Markov Decision Process (MDP)**. Attetion à bien définir les états, actions, et récompenses,et à réduire la taille de l'espace d'état si nécessaire, pour éviter l'explosion combinatoire.
- l'implémentation et la comparaison de différentes stratégies d'apprentissage (tabulaires et approximées),
- l'analyse des performances et des comportements émergents.

2. Le jeu Corridor

Le jeu se joue sur un plateau de taille $N \times N$ (par défaut $N = 9$ mais ce chiffre devra être modifié à la baisse si vos agents n'arrivent pas à jouer dans un tel plateau). Chaque joueur dispose :

- d'un pion, placé initialement au centre de sa première ligne ;
- d'un certain nombre de murs à placer (10 par joueur par défaut).

Le but est d'atteindre la ligne opposée avant l'adversaire. Les joueurs alternent leurs tours, et à chaque coup, ils peuvent :

1. déplacer leur pion d'une case (en avant, arrière, gauche ou droite, avec gestion des sauts/diagonales) ;
2. placer un mur horizontal ou vertical, à condition de ne pas bloquer complètement l'accès à la ligne adverse.

3. Matériel fourni

Deux fichiers sont fournis :

— `corridor.py` : contient la classe `Corridor`, qui implémente toutes les règles du jeu. Ce fichier **ne doit pas être modifié**.

— `starter_corridor.py` : contient une boucle principale et un `RandomAgent` de base. C'est le point d'entrée de votre code.

La classe `Corridor` fournit les principales méthodes suivantes :

— `reset()` : initialise une nouvelle partie ;

— `step(action)` : applique une action et renvoie la nouvelle observation, la récompense, et un indicateur de fin ;

— `legal_actions()` : renvoie la liste des actions légales pour le joueur courant ;

— `render()` : affiche l'état actuel du plateau en ASCII ;

— `shortest_path_length(player)` : calcule la distance minimale d'un joueur à sa ligne d'arrivée (utile pour des heuristiques).

4. Travail demandé

4.1. Agent de base (obligatoire)

Dans un premier temps, implémentez un agent simple en complétant la classe :

```
class MyAgent(BaseAgent):
    def select_action(self, env, obs):
        ...
```

Votre agent doit choisir une action parmi celles renvoyées par `env.legal_actions()` et interagir avec l'environnement via la boucle `play_game()` du fichier `starter_corridor.py`.

Plusieurs approches sont possibles :

— stratégie aléatoire (Random Agent),

— stratégie heuristique (minimiser la distance à la ligne d'arrivée),

— apprentissage par renforcement tabulaire (Q-Learning, SARSA, etc.),

— apprentissage avec approximation de fonction (DQN, réseau de neurones simple, etc.).

4.2. Entraînement et évaluation

Vous devrez concevoir une fonction d'entraînement adaptée à votre méthode :

— définir la représentation des états et actions ;
— choisir la politique d'exploration (ex : ε-greedy) ;
— ajuster les hyperparamètres (facteur γ, taux d'apprentissage, etc.).

L'évaluation se fera en faisant jouer votre agent contre :
— un agent aléatoire ;
— un agent heuristique simple (fourni dans le starter).

Mesurez :
— le taux de victoire,
— la longueur moyenne des parties,
— la stabilité de l'apprentissage.

4.3. Rapport à rendre

Vous devrez fournir un court rapport (2 à 3 pages maximum) contenant :
— la description de votre méthode d'apprentissage,
— les choix d'implémentation (représentation d'état, fonctions de récompense, etc.),
— les résultats expérimentaux (tableaux, graphiques, observations),
— une discussion critique des limites et perspectives.

5. Contraintes et consignes de rendu

— Le fichier `corridor.py` **ne doit pas être modifié**.
— Le fichier `starter_corridor.py` peut être modifié ou étendu pour inclure vos agents et vos fonctions d'apprentissage.
— Vous pouvez ajouter d'autres fichiers Python (par ex. `my_agent.py`) à condition qu'ils soient importés dans `starter_corridor.py`.
— Le rendu final devra contenir :
  — le code source de votre agent,
  — le script de test/démonstration,
  — un rapport PDF.

6. Extensions optionnelles

Pour aller plus loin, vous pouvez explorer :
— une **fonction de valeur approchée** par un réseau de neurones (DQN) ;
— une visualisation graphique des trajectoires et politiques apprises.

7. Critères d'évaluation

Critère	**Barème indicatif**
Implémentation correcte de l'agent	30%
Méthode d'apprentissage / heuristique	25%
Analyse expérimentale et discussion	25%
Qualité du code et respect des consignes	10%
Présentation du rapport	10%

Le projet est à réaliser en binôme ou individuellement. La soutenance (optionnelle) consistera à une courte démonstration du fonctionnement de l'agent.

Bon courage et amusez-vous à entraîner vos agents !