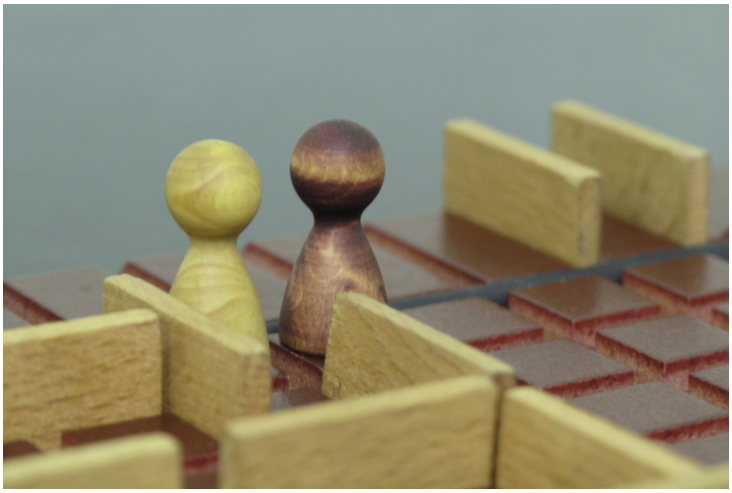
Université de Bordeaux

Master 2 Informatique - Intelligence Artificielle

Apprentissage par Renforcement

Agent Intelligent pour le Jeu de Corridor



Auteurs :
Arthur Macdonald
Niels Roudeau

Date :
2025 – 2026

Table des matières

1 Présentation du projet **2**

2 Architecture du Projet **2**

3 Définition d'un MDP **2**

3.1 Représentation de l'état d'une partie pour Apprentissage tabulaire 2

3.2 Représentation de l'état d'une partie pour Apprentissage profond 3

3.3 Système de Récompenses 4

4 Résultats expérimentaux **5**

1 Présentation du projet

L'objectif de notre projet est d'implémenter un agent intelligent capable de jouer au jeu Corridor, une implémentaion simplifié du jeu Quoridor. Nous avons implémenté plusieurs agents, chacuns utilisant des méthodes d'apprentissages basées sur l'apprentissage par renforcement :

— **Q-Learning / Sarsa** (Apprentissage tabulaire)

— **Deep Q-Network** (Réseau de neurones)

Nos agents comprennent le jeu à l'aide d'un processus de décision Markovien (MDP) la définition et l'implémentation de ce processus font parti de ce rapport.

Nous traiterons dans un premier temps de nos choix d'implémentation pour ces agents.

Enfin, nous présenterons nos résultats expérimentaux obtenus lors de phases d'entrainement et d'évaluation de nos modèles.

2 Architecture du Projet

Pour alléger le rapport, l'arbre détaillé de l'arborescence du dépôt a été retiré. Le code complet et la structure du projet sont disponibles en ligne : https://github.com/Roirtur/CorridorAIRL.

Les composants principaux sont :

— le moteur de jeu : `corridor.py` ;

— des scripts utilitaires : `start_training.py`, `start_evaluation.py` et `corridor_starter.py` ;

— la mise en oeuvre des agents dans le dossier `models/` (agents tabulaires, DQN, heuristiques) ;

— les notebooks d'analyse : `experiments.ipynb` ;

— utilitaires divers dans `utils/` et modèles sauvegardés dans `saved_models/`.

Notre travail comprend :

— Implémentation de deux agents intelligents suivant les méthodes d'apprentissage tabulaires Q-Learning et Sarsa.

— Implémentation d'un agent utilisant un Deep Q-Network.

— Définition d'un MDP autour du jeu de Corridor.

— **start_evaluation.py, start_training.py** : programmes utilisables en CLI pour l'entrainement et l'évaluation des modèles.

— **experiments.ipynb** : Notebook Jupyter sur nos procédés experimentaux et leurs résultats

3 Définition d'un MDP

3.1 Représentation de l'état d'une partie pour Apprentissage tabulaire

La difficulté principale dans l'apprentissage tabulaire du jeu Corridor est d'éviter une explosion combinatoire de l'espace d'états (*Curse of Dimensionality*).

Pour pallier à cela, nous avons conçu une représentation compacte, visant à réduire la cardinalité de l'espace tout en conservant les informations stratégiques cruciales. Notre tuple d'état (S) est construit selon les principes suivants :

— **Binning des Murs :** Le nombre exact de murs restants modifie rarement la stratégie optimale. Nous avons donc discrétisé cette valeur en intervalles. L'agent regroupe des états stratégiquement équivalents et accélère donc la convergence de la Q-Table.

— **Abstraction des obstacles :** Plutôt que d'encoder la position de chaque mur, nous capturons l'impact topologique des murs en calculant la distance du plus court chemin à l'arrivée via BFS pour chaque joueur.

— **Encodage de la topologie locale :** Pour faciliter l'apprentissage des mouvements légaux sans avoir à parser la liste globale des murs, nous incluons un *masque d'adjacence*. Il s'agit d'un encodage binaire représentant les obstacles immédiats (murs ou bords du plateau) autour des deux joueurs. Cela offre à l'agent une "vision locale" immédiate.

— **Symétrie de plateau :** Le jeu présentant une symétrie centrale, nous normalisons systématiquement l'observation. L'état est toujours représenté du point de vue d'un joueur devant atteindre le bas du plateau.

L'état final est donc un tuple :

$$S = (Pos_{me}, Pos_{opp}, Walls^{bin}_{me}, Dist_{me}, Dist_{opp}, Mask_{me}, Mask_{opp})$$

Selon nos expérimentations et les documents scientifiques consultés durant nos travaux, le nombre de murs restants pour l'adversaire n'est pas un élément pertinent à utiliser. Il ne fait qu'accélerer l'explosion combinatoire. Selon nos expérimentations et les documents scientifiques consultés durant nos travaux (voir notamment Glendenning [1]), le nombre de murs restants pour l'adversaire n'est pas, dans notre configuration, un facteur décisif et contribue surtout à l'explosion combinatoire si on l'intègre finement.

Notre implémentaion a été dirigée dans l'intêret d'avoir des principes théoriquement "séparés". Par exemple l'ajout d'informations sur les murs feraient perdre sens à notre choix d'utiliser un binning, on se contente d'utiliser des informations simples qui empêchent l'agent de se perdre dans ses décisions.

3.2 Représentation de l'état d'une partie pour Apprentissage profond

Contrairement à l'approche tabulaire où nous avons dû extraire manuellement des caractéristiques de haut niveau (distances, masques), l'approche par Deep Q-Network (DQN) approxime la fonction de valeur d'action à l'aide d'un réseau de neurones. Nous avons choisi d'utiliser un réseau de neurones convolutif (CNN) car le jeu se joue sur une grille et ce type d'architecture est particulièrement adapté pour exploiter la structure spatiale et les motifs locaux (couloirs, impasses). Cette approche nous a semblé cohérente avec la littérature et ressources en ligne consultées durant le projet.

Nous avons donc par découlement opté pour une représentation qui préserve la structure spatiale du jeu. L'état n'est plus un tuple abstrait, mais un tenseur de forme $(C \times N \times N)$.

Notre représentation comporte $C = 6$ canaux, normalisés entre 0 et 1 pour assurer la stabilité numérique du réseau :

1. Glendenning, 2000. Undergraduate Thesis. `https://www.labri.fr/perso/renault/working/teaching/projets/files/glendenning_ugrad_thesis.pdf`

— **Canaux de Position (2 plans) :** Deux grilles binaires encodant respectivement la position du joueur courant ($P1$) et celle de l'adversaire ($P2$). Cela permet au réseau de situer les agents dans l'espace sans ambiguïté.

— **Canaux de Topologie (2 plans) :** Deux grilles binaires représentant les murs posés (un plan pour les murs horizontaux, un pour les verticaux). Contrairement à une liste de coordonnées, cette représentation matricielle permet aux filtres de convolution de détecter des motifs locaux comme des "couloirs", des "impasses" ou des "blocages".

— **Canaux de Contexte (2 plans) :** Le nombre de murs restants est une information scalaire, mais cruciale. Pour l'intégrer dans une architecture convolutionnelle, nous utilisons la technique des "plans scalaires" (utilisée notamment par AlphaGo) : nous remplissons deux plans entiers avec la valeur normalisée du nombre de murs restants (pour soi et pour l'adversaire). Cela fournit un biais constant à chaque neurone de la couche d'entrée, conditionnant l'analyse spatiale aux ressources disponibles.

3.3 Système de Récompenses

Conformément à l'implémentation du moteur, les récompenses terminales sont :

$$r_T = \begin{cases} +1.0 & \text{si l'agent gagne au coup terminal,} \\ -1.0 & \text{si l'agent perd,} \\ -0.5 & \text{en cas de match nul / timeout (pour certaines expériences).} \end{cases}$$

Pour les agents tabulaires (Q-Learning, SARSA) nous avons ajouté un terme de shaping basé sur la variation de la distance au but calculée par BFS. À chaque pas pertinent de l'agent, la récompense non terminale est :

$$r_t = \alpha \cdot \big(d_{t-1} - d_t\big) + c_{\text{step}},$$

avec les hyper-paramètres utilisés dans le code :

— $\alpha = 0.1$ (gain sur la réduction de la distance) ;

— $c_{\text{step}} = -0.01$ (petite pénalité de pas pour décourager les aller-retour).

Ainsi, chaque action qui rapproche l'agent de son objectif reçoit une récompense positive proportionnelle à la diminution de la distance ; les actions qui l'éloignent reçoivent une récompense négative relative. Ce shaping est appliqué uniquement pendant l'entraînement tabulaire afin d'accélérer la propagation des valeurs dans la Q-table.

Justification et limites Le shaping par différence de distance satisfait l'intuition stratégique (progresser vers la ligne adverse) et réduit la variance des retours en fournissant un signal dense. Néanmoins :

— il peut modifier la politique optimale si mal calibré (risque d'introduire un biais vers des trajectoires “courtes” mais tactiquement faibles) ;

— il dépend de la métrique de distance choisie (ici shortest path via BFS) — lorsque les murs sont posés l'estimation peut varier brutalement ;

— il n'est appliqué qu'aux agents tabulaires : les agents DQN, traitant une représentation spatiale riche, reçoivent par défaut les récompenses fournies par l'environnement (0 pour coups non-terminaux, +1/-1 terminaux). Conformément aux expérimentations présentées dans `experiments.ipynb`, nous n'avons pas utilisé de shaping pour la plupart des entraînements DQN présentés (les courbes et évaluations publiées correspondent à cette configuration). Il reste toutefois possible d'ajouter le même shaping pour un entraînement DQN alternatif.

4 Résultats expérimentaux

L'ensemble des expériences et résultats détaillés sont présentés dans le notebook **experiments.ipynb** (disponible dans le dépôt). Nous résumons ici les principaux enseignements et observations :

Méthodologie expérimentale

1. **Entraînement des agents** :
   — Les courbes de récompense sont enregistrées à chaque épisode et lissées pour une meilleure lisibilité.
   — Plusieurs stratégies d'entraînement sont testées : contre un adversaire aléatoire (Random), contre un adversaire Greedy, ou via un curriculum (Random → Greedy → Mix).
2. **Évaluation** :
   — Chaque agent est évalué sur 500 parties contre `RandomAgent` et `GreedyPathAgent`, en alternant le joueur qui commence.
   — Des tournois agent vs agent sont organisés (200 parties par duel) pour comparer directement les stratégies apprises.

Principaux résultats

— **DQN vs tabulaire** : Les agents DQN surpassent les agents tabulaires (Q-Learning, Sarsa) sur les grands plateaux et montrent une meilleure robustesse face à la diversité des situations.

— **Effet du curriculum** : Pour les agents tabulaires, l'entraînement progressif (curriculum) améliore nettement la capacité à battre différents types d'adversaires. Pour DQN, un entraînement trop long contre Greedy peut nuire à la généralisation.

— **Diversité des stratégies** : Les agents DQN entraînés uniquement contre Random développent parfois des stratégies plus variées et efficaces que ceux sur-entraînés contre Greedy.

— **Classement** : Les tournois montrent que DQN est globalement supérieur, suivi de Sarsa puis Q-Learning, mais la hiérarchie peut varier selon la configuration d'entraînement.

Reproductibilité

Tous les scripts, modèles sauvegardés et logs d'entraînement sont fournis dans le dépôt. Il est possible de reproduire l'ensemble des figures et évaluations à partir des fichiers fournis.

Pour plus de détails, de graphiques et d'exemples de parties, se référer au notebook `experiments.ipynb`.