Rapport Deep Reinforement Learning

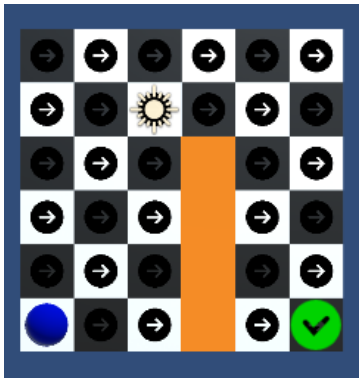
Dans ce rapport, nous allons passer en revue les résultats obtenus avec les différents algorithmes étudiés en cours et donner un avis sur lequel de ces algorithmes est, selon nous, le plus performant.

Les cas d'utilisations des algorithmes seront dans un modèle GridWorld et dans un modèle Sokoban.

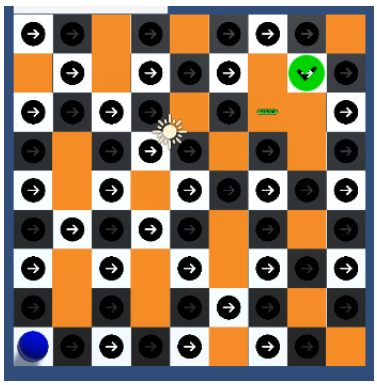
Policy Iteration :

Résultat dans GridWorld : Cet algorithme combine l'évaluation de la politique (policy evaluation) et l'amélioration de la politique (policy improvement) en une seule étape itérative. Il est souvent efficace pour trouver la meilleure politique dans des environnements de GridWorld relativement simples. Cependant, si l'environnement est très complexe, il peut être coûteux en temps de calcul.

0.001s



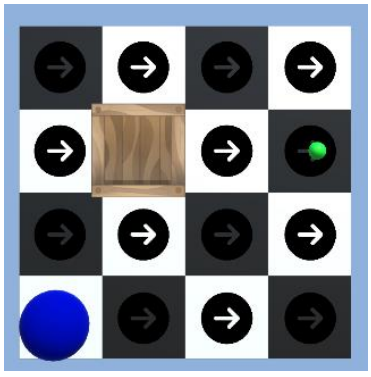
0.009s



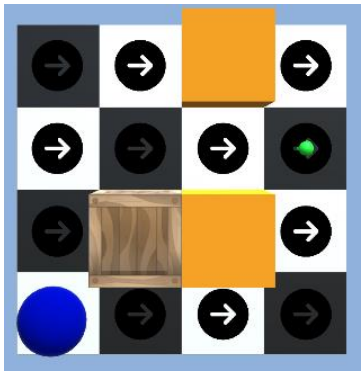
0.038s

Temps d'execution moyen

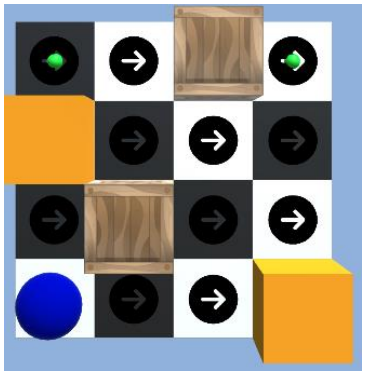
Résultat dans Sokoban : Cet algorithme peut être efficace dans des environnements Sokoban relativement simples avec un petit nombre d'états et d'actions. Cependant, il peut devenir inefficace dans des environnements plus complexes en raison de la complexité de la recherche de la meilleure politique.



0.063s



0.05s



3.836s

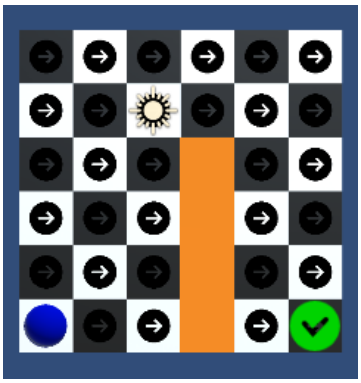
Temps d'execution moyen

Utilité : Etant donné la perte d'efficacité face à des problèmes de plus en plus complexe, l'algorithme de PolicyIteration ne semble pas adapté au domaine du jeu vidéo. En effet, la complexité des features implémentées dans un jeu peuvent faire de l'algorithme un poids et pourrait causer des ralentissements. Il reste malgré tout assez performant pour des exemples simples de GridWorld ou de Sokoban comme montré ci-dessus.

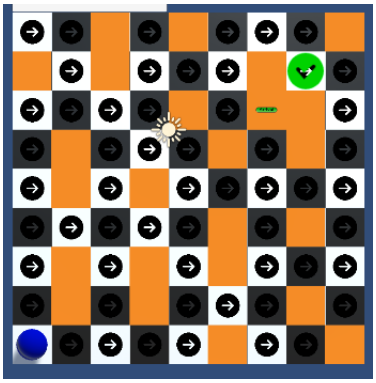
Value Iteration :

Résultat dans GridWorld : Cet algorithme calcule itérativement la fonction de valeur optimale pour chaque état en utilisant une fonction de récompense et une fonction de transition définies pour l'environnement. Bien que cela puisse être coûteux en temps de calcul pour des environnements très complexes, Value Iteration est efficace pour des environnements relativement simples de GridWorld.

0.011s



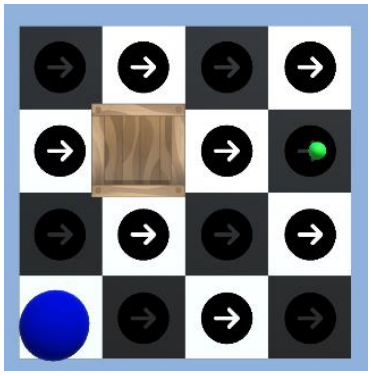
0.028s



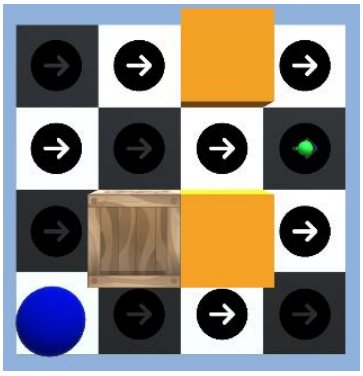
0.06s

Temps d'execution moyen

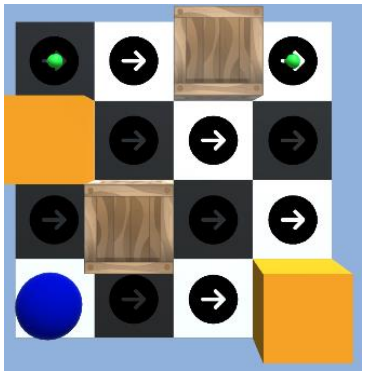
Résultat dans Sokoban : Cet algorithme est également utile pour des environnements Sokoban relativement simples, mais il peut être plus rapide à converger que PolicyIteration, en particulier lorsque la fonction de valeur est initialement bien estimée.



0.03s



0.019s



1.488s

Temps d'execution moyen

Utilité : Même si l'efficacité de l'algorithme est meilleure que celui de PolicyIteration, il reste quelques défauts qui font qu'il n'est pas optimal d'utiliser cet algorithme dans le domaine du jeu vidéo, le coût en temps de calcul étant le plus gros frein selon nous.

Monte-Carlo :

Résultat dans GridWorld : les algorithmes de Monte-Carlo, plus précisément ES, on-policy et off-policy, sont des méthodes d'apprentissage par renforcement qui sont souvent utilisées dans des environnements où il est difficile d'initialiser l'agent dans toutes les situations possibles. L'algorithme explore toutes les possibilités en initialisant l'agent dans des états aléatoires et des actions aléatoires, puis met à jour la valeur de la politique en fonction des résultats de ces expériences. Les algorithmes de Monte-Carlo peuvent être efficace dans des environnements de petite taille, mais ils peuvent être lent à converger dans des environnements plus grands.

Résultat dans Sokoban : Dans le cas de Sokoban, cela pourrait être utile pour l'initialisation de la position des caisses et des objectifs. Monte-Carlo reste efficace dans des environnements de petite taille, mais il peut malgré tout lent à converger dans des environnements plus grands.

Utilité : La convergence assez lente dans de grands environnements fait que l'algorithme de Monte-Carlo ne sera pas très utile dans des jeux vidéos dans lesquels un grands nombres d'événements peuvent se passer dans le même temps dans une vaste zone de jeu. Cependant il peut peut être trouvé son utilité dans des domaines limités ou des zones restreintes tel que le Sokoban.

(Source : https://youtu.be/-YpalutQCKw)

SARSA :

Résultat dans GridWorld : Cet algorithme est un exemple d'apprentissage par renforcement on-policy, c'est-à-dire que la politique apprise est utilisée pour choisir les actions pendant l'apprentissage. SARSA est efficace pour des environnements où les actions de l'agent entraînent des conséquences à court terme. Cependant, il peut être plus lent à converger que d'autres algorithmes.

Résultat dans Sokoban : Cet algorithme est souvent utile pour des environnements Sokoban où les actions de l'agent ont des conséquences à court terme. Par exemple, si l'agent doit pousser une boîte, il peut être important de tenir compte de l'état futur de la boîte lors du choix de l'action. SARSA est un algorithme on-policy, ce qui signifie que l'agent apprend à partir de ses propres expériences plutôt que de données fournies par l'environnement.

Utilité : L'algorithme de SARSA, contrairement aux algorithmes abordés avant, semble être une bonne option dans le développement d'IA dans le jeu vidéo. Il peut permettre à des PNJ d'apprendre des comportements qui les rendraient réaliste dans le jeu. Malgré une convergence assez lente, son utilisation ne semble pas être une mauvaise chose compte tenu de sa fiabilité.

(Source : https://youtu.be/kp4TNUbiqFA)

Q-Learning :

Résultat dans GridWorld : Cet algorithme est un exemple d'apprentissage par renforcement off-policy, c'est-à-dire que la politique apprise est indépendante des actions de l'agent pendant l'apprentissage. Q-Learning est souvent efficace pour les environnements de GridWorld où les actions de l'agent ont des conséquences à long terme. Il peut également être plus rapide à converger que SARSA.

Résultat dans Sokoban : Cet algorithme peut être utile dans des environnements Sokoban où les actions de l'agent ont des conséquences à long terme. Q-Learning peut également être plus rapide à converger que SARSA, car il ne prend pas en compte l'état futur de l'agent lors de la sélection de l'action.

Utilité : L'algorithme de Q-Learning est selon nous le meilleur compromis de tous les algorithmes présentés jusqu'ici, car tout aussi performant que l'algorithme de SARSA sans les problèmes de lenteur lors de la convergence. Il est selon nous le plus approprié dans l'utilisation d'algorithme dans le Jeu Vidéo.

(Source : https://youtu.be/bHeeaXgqVig)

Conclusion :

Le Deep Reinforcement Learning peut être utilisé pour prédire la rétention des joueurs en apprenant à partir des données de jeu et des habitudes de jeu des joueurs pour prédire le moment où ils sont susceptibles d'abandonner le jeu.

Le Deep Reinforcement Learning peut également servir à générer automatiquement des niveaux, des personnages ou des objets de jeu. L'agent apprend à créer du contenu en fonction des préférences des joueurs en maximisant une récompense prédéfinie.

On peut ainsi aider les Game Designers à anticiper au préalable la manière dont les joueurs peuvent réagir face à ce qu'ils auront en main. Cela peut être un outil très puissant qui permettra d'offrir aux joueurs l'expérience idéale.