Multi-agents - Markov Decision Process / Reinforcement Learning

Valérian ACIER : *valerian.acier@etu.univ-lyon1.fr*

1 MDP - Markov Decision Process

1.1 Le sujet

Le but de ce Travail Pratique est de mettre en place un processus markovien pour de l'aide à la décision.

Pour ce faire on utilise l'algorithme de Value Iteration qui consiste à passer sur les états et à modifier leurs valeurs selon l'équation de bellman jusqu'à convergence.

1.2 Question 1 :

Pour permettre la traverser du pont une solution consiste à mettre un bruit à 0 cela permettant de passer dans un environnement déterministe et ainsi annuler le risque d'aller sur un état -10 alors que l'agent souhaite simplement se rapprocher de l'état + 10.

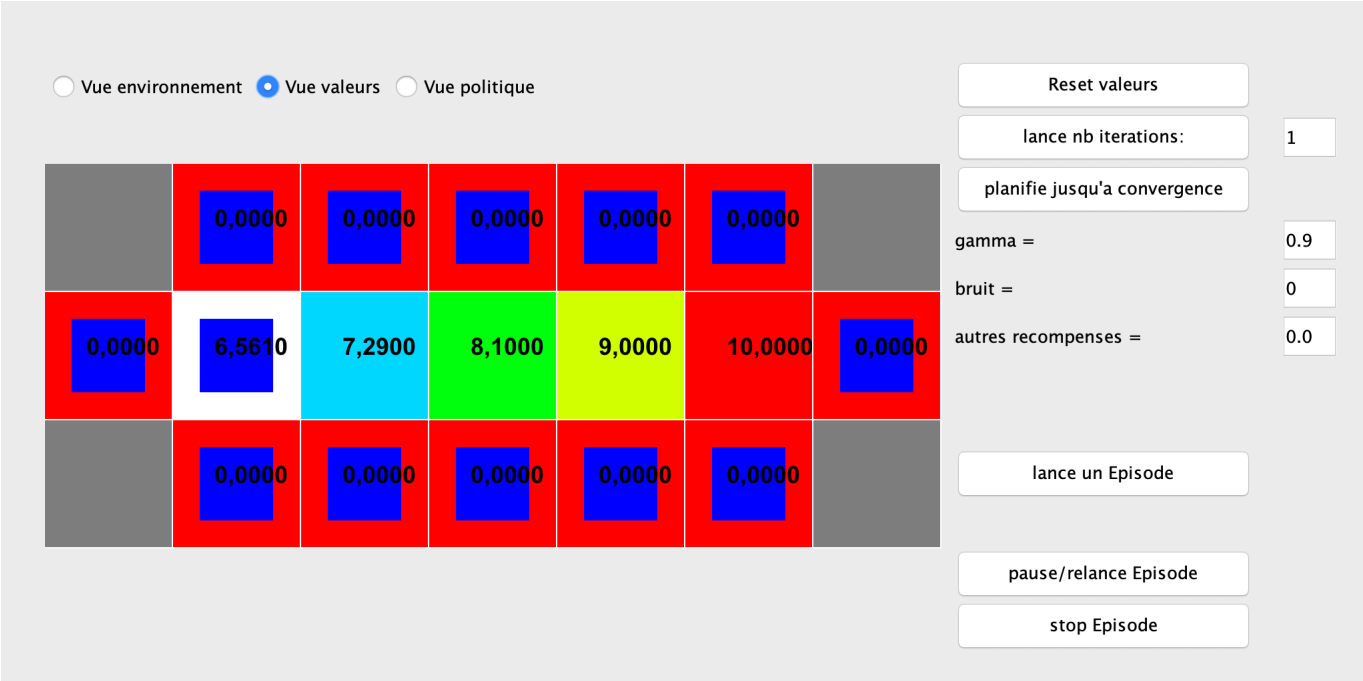

Figure 1: Les valeurs de l'environnement bridge avec un bruit à 0

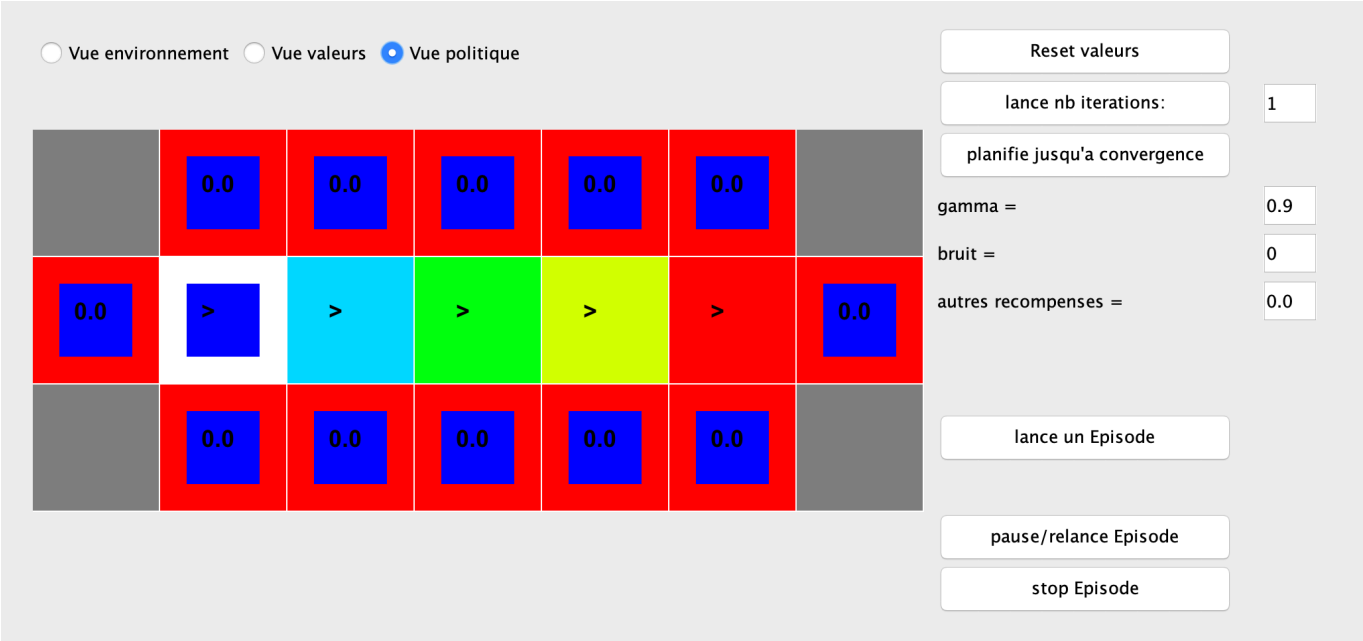

Figure 2: La politique de l'environnement bridge avec un bruit à 0

1.3 Question 2

1 . Chemin risqué à récompense + 1 J'ai mis une récompense par déplacement négatif à -4 ceci dans le but de pousser l'agent à aller vers la récompense la plus rapide, mais pas trop élevée pour ne pas le pousser à aller dans un état négatif.

Figure 3: Politique pour le chemin risqué à récompense + 1

2 . Chemin risqué à récompense + 10 Pour pousser l'agent à aller vers la récompense maximale par le chemin risqué, j'ai simplement mis le bruit à 0, il ira donc vers le chemin le plus rapide qui apporte le plus de récompenses.

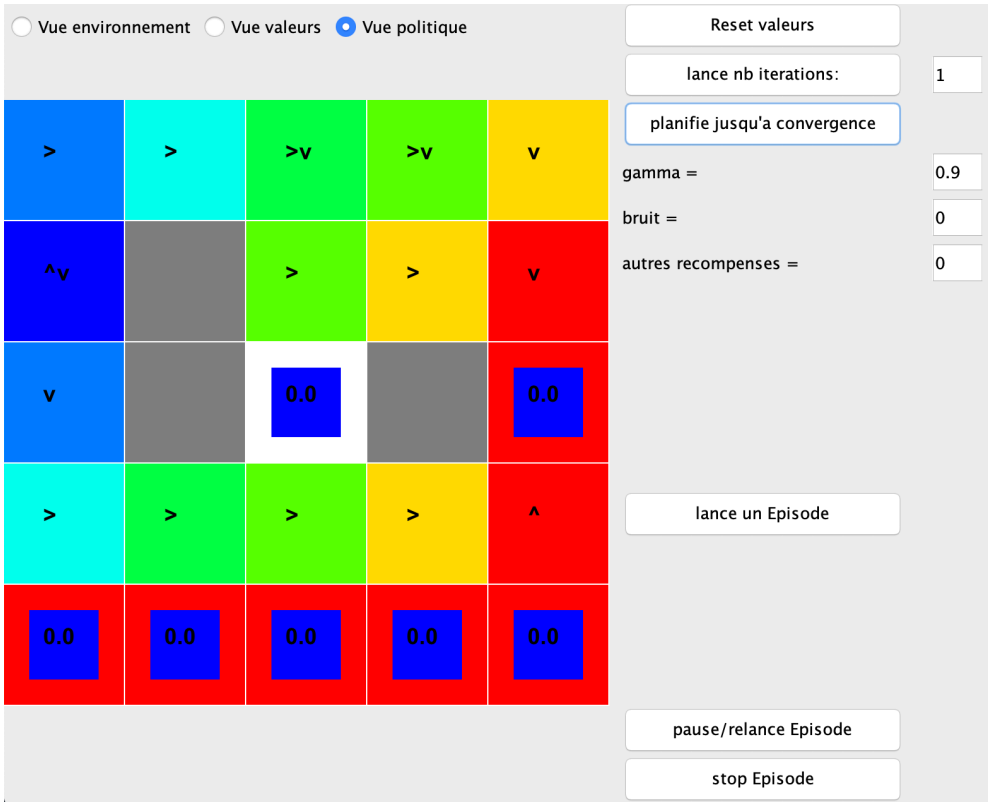

Figure 4: Politique pour le chemin risqué à récompense + 10

3. Chemin sûr à récompense + 1 Pour pousser l'agent à aller chercher la récompense de manière sûre j'ai mis un gamma très faible permettant de diminuer les récompenses des états éloignés ceci poussant ainsi l'agent à prendre le chemin sûr vers la récompense la plus proche.

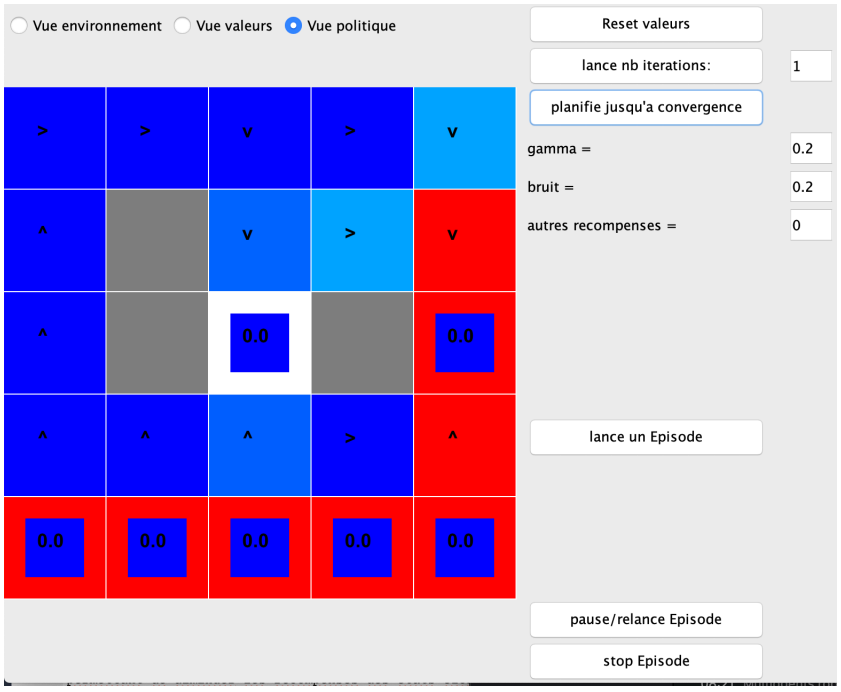

Figure 5: La politique pour le chemin sûr à récompense + 1

4. Éviter les états absorbants Pour éviter que l'agent aille dans les états absorbants, j'ai mis la valeur des récompenses de déplacements standards à 10 ce qui a pour effet de pousser l'agent à éviter les états absorbants, car ils apportent moins de récompenses.

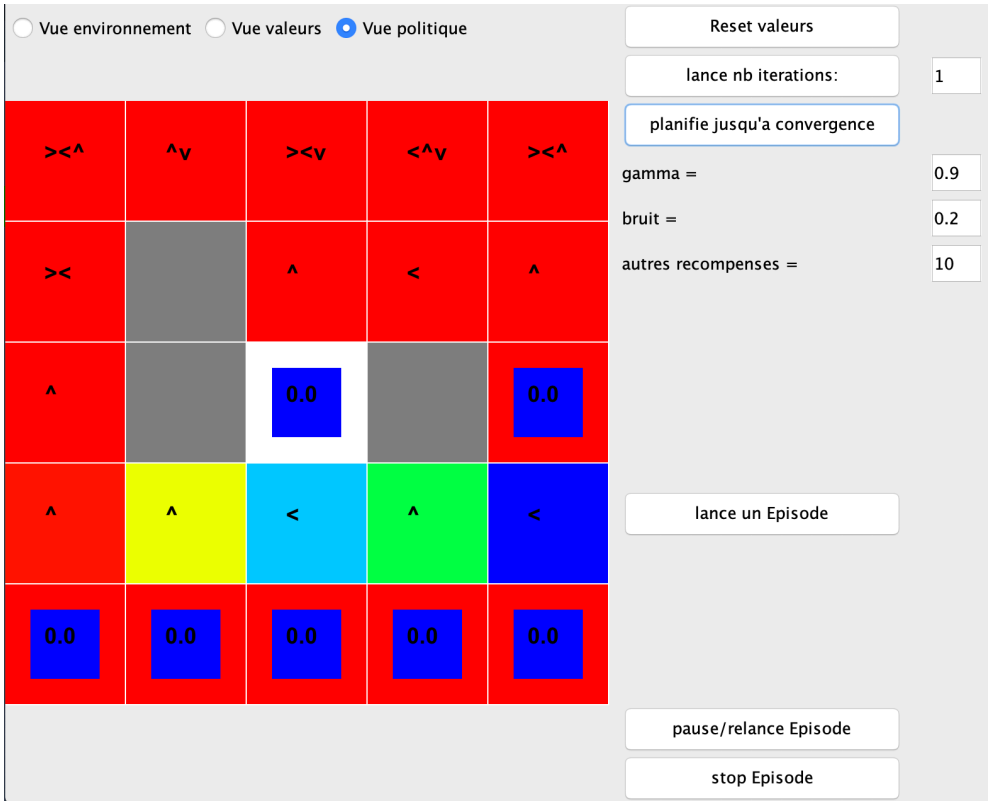

Figure 6: La politique pour éviter les états absorbants

2 Reinforcement Learning - QLearning

2.1 Le sujet

Le but de ce Travail Pratique est de mettre en place un algorithme de Q-Learning à la différence des MDP le QLearning ne nécessite pas de connaitre à l' avance les effets des actions de l'agent et les états récompensés.

2.2 QLearning Tabulaire

Lors de ce TP la première partie a été la mise en place d'un Q-Learning Tabulaire dans les mêmes environnements que pour les MDP (voir figure 7), ensuite une fois fonctionnel, le tester sur le robot crawler puis sur un Pacman.

Figure 7: Résultats du QLearning sur l'état gridworld après 1000 itérations en stratégie greedy avec epsilon à 0.1

Pour la mise en place du pacman, il a fallu extraire des informations de l'état du jeu, car le jeu en lui-même comprend trop de données (positions des murs, de la nourriture, des fantômes ...) et cela aurait pour effet une trop grande QTable et donc beaucoup d'états différents et un apprentissage très long.

J'ai donc choisit de récupérer seulement les cellules voisines au pacman (droite, bas, gauche, haut ainsi que les diagonales) et quatre autre données représentant le premier élément non vide se trouvant dans chaque direction du pacman (voir figure 8 pour un exemple) dans le but de permettre de repérer les fantômes ou nourriture en ligne droite du pacman.

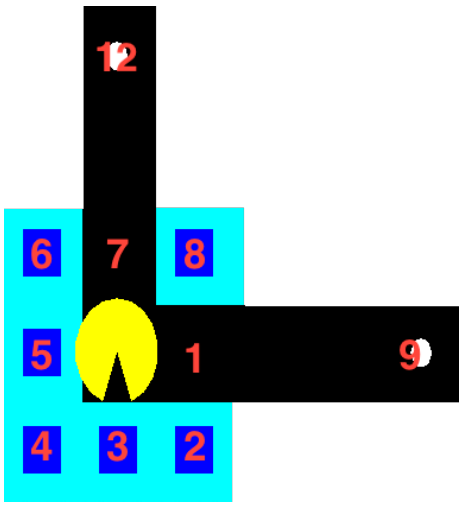

Figure 8: Représentation d'un état dans la QTable. Les numéros en rouge correspondent aux variables, la variable 10 a le même état que la variable 3 et la variable 11 la même que la 5, car l'élément non vide le plus proche dans leur direction est le voisin direct

2.3 QLearning Généralisation

Une fois cela fait, il a fallu mettre en place une approximation linéaire de la fonction Q pour permettre une meilleure gestion des cas jamais rencontrés précédemment.

Je n'ai malheureusement pas pu mettre en place la solution de test de l'approximation, car il fallait réaliser une matrice pour chaque état possible. Dans mon cas cela fait 12 variables avec 5 valeurs différentes ce qui représente une matrice de taille 5^{12} ce qui n'est pas réalisable. J'ai donc passé cette partie pour réaliser directement l'approximation sur les features donnés dans l'exercice.

Une fois l'approximation mise en place j'ai obtenu un taux de 80% de réussite sur le niveau "mediumGrid" du pacman mais en modifiant les features de la fonction Q en mettant la vraie distance à la place de la fonction de distance Manhattan le taux de réussite est monté à plus de 90% pour 500 épisodes d'entrainement et 300 greedy pour gamma à 0.8, alpha à 0.1, epsilon à 0.05 et la moyenne sur 3 entrainements.

Les features extraites de Pacman pour la généralisation :

- Le biais (1)
- Le nombre de fantômes qui peuvent atteindre le pacman sur sa nouvelle position si il se déplace
- La vrai distance (avec pathfinding en BFS=Djikstra) la plus proche de la nourriture si le pacman se déplace diviser par la vraie distance la plus proche de la nourriture par rapport à la position actuelle du pacman.