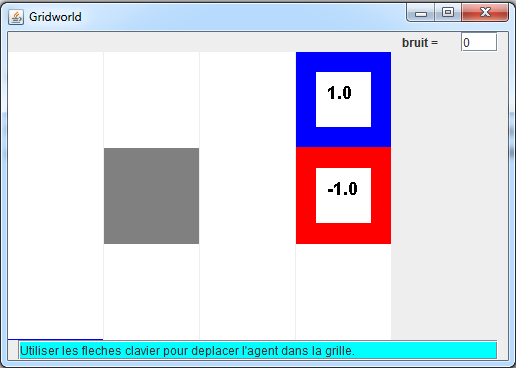
Intelligence artificielle

Planification stochastique & MDP

## Mise en place du TP



*Figure 1 : GridworldMDP avec bruit = 0*

Le bruit est un chiffre, compris entre 0 et 1, qui correspond à la probabilité, pour notre robot, d'emprunter le mauvais chemin (par exemple aller à droite alors que l'utilisateur a appuyé sur la flèche du haut).

Ici notre bruit est réglé sur la valeur 0, c’est-à-dire que dans tous les cas le robot ira dans la direction entrée par l'utilisateur, nous nous trouvons ici dans un environnement déterministe.

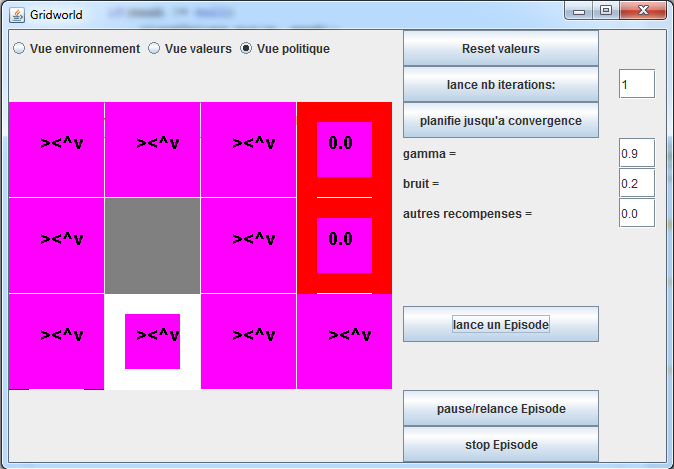
Plus le bruit est positionné sur une valeur élevée, plus il sera difficile pour l'utilisateur de contrôler efficacement le robot.

## Agent Aléatoire

Cet agent renvoie de manière aléatoire la prochaine action que le robot doit effectuer pour cela elle s'aide de 2 principales fonctions :

* getPolitique : qui renvoie, dans le cas d'un agent aléatoire, toutes les actions possibles à partir d'un état donné.
* - getAction : à partir d'un état, renvoie une action aléatoirement sélectionnée parmi les actions renvoyées par getPolitique.

Dans le cas d'un agent aléatoire la fonction getValeur n'a pas lieu d'être, celle-ci va donc renvoyer 0 pour chaque état.



*Figure 2 : TestRandomAgent lancé avec les valeurs par défaut*

Lorsqu'on lance un épisode notre robot se déplace aléatoirement jusqu'à qu'il arrive à la récompense ou qu'il entre dans un état absorbant ayant une récompense négative.

## Algorithme value iteration

### Agent Value Iteration

Cet agent planifie sa politique en utilisation l'algorithme value iteration, cet algorithme est basé sur une valeur optimale V\* qui est calculée à chaque étape de l'algorithme.

On retrouve ici les mêmes fonctions qui étaient disponibles dans l'agent aléatoire, cependant leur comportement a changé :

* getValeur : Cette action renvoie cette fois la valeur calculée par l'algorithme, pour l'état donné, à l'itération précédente, si aucune valeur n'a été calculée précédemment, elle renvoie 0.
* getPolitique : Cette fois getPolitique ne va pas renvoyée toutes les actions possibles à partir d'un état, elle va uniquement renvoyée les actions dont la somme calculée par la formule de la politique maximale est la plus haute, getPolitique peut donc être amenée à renvoyer plusieurs actions.
* getAction : Similairement à l'agent aléatoire, getAction va renvoyer une action choisie aléatoirement parmi les actions renvoyées par getPolitique (toujours pour un état donné).

De plus cet agent dispose d'une nouvelle fonction, la fonction updateV, qui va itérer à travers chacun des états afin de mettre à jour leur valeur V. UpdateV mettra à jour cette valeur tant que la distance entre la valeur à l'itération k et la valeur à l'itération k – 1 est supérieure à une valeur de convergence δ.

## Influence des paramètres

Le programme que nous utilisons nous permet de modifier 3 paramètres :

* Le bruit : Ce paramètre est compris entre 0 et 1, et représente la probabilité pour le robot de ne pas aller dans la bonne direction.
* Le gamma γ : Cette valeur est comprise entre 0 et 1, avec un γ faible, la politique priorisera les récompenses proches, un γ élevé forcera la politique a regardé les récompenses plus éloignés et éventuellement essayé de récupérer ces récompenses si celle-ci sont plus avantageuses que les récompenses proches.
* Les autres récompenses rother : Cette valeur représente la récompense obtenue lorsque le robot passe sur un des autres états.

### Question 1

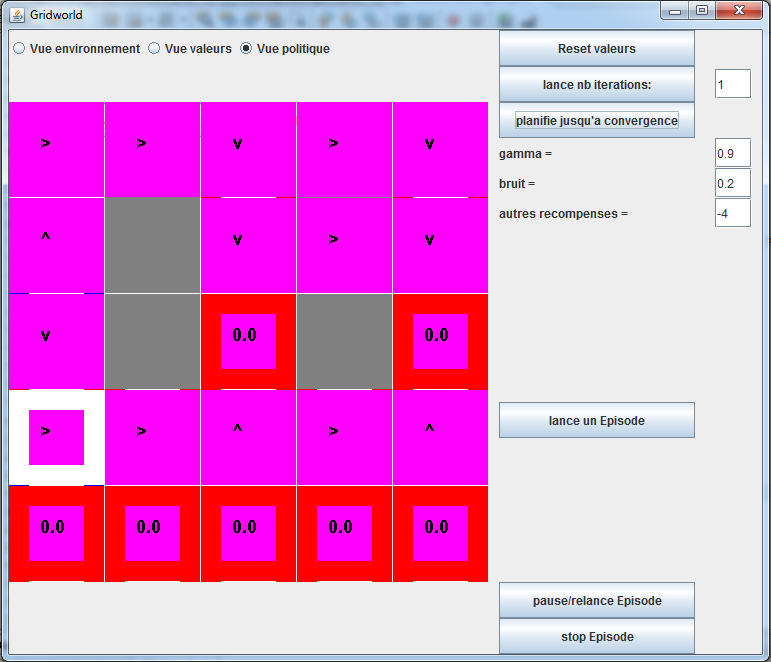
Il faut diminuer le bruit à 0 pour avoir une politique optimale qui permet de traverser le pont.

### Question 2

Par défaut, les valeurs sont fixées de la façon suivante :

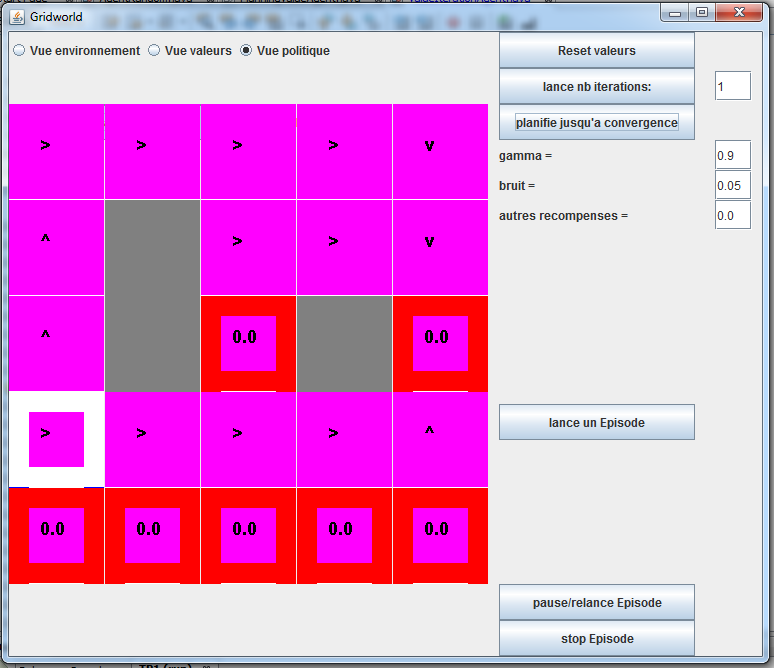
γ = 0.9 bruit = 0.2 rother = 0

1. rother = -2, -3 ou -4 permet d'obtenir une politique optimale qui suit un chemin risqué pour atteindre l'état absorbant de récompense +1. Les récompenses des autres états étant négatives, la politique fera tout pour arriver le plus vite possible sur un état ayant une récompense positive afin de maximiser le plus possible sa récompense globale.



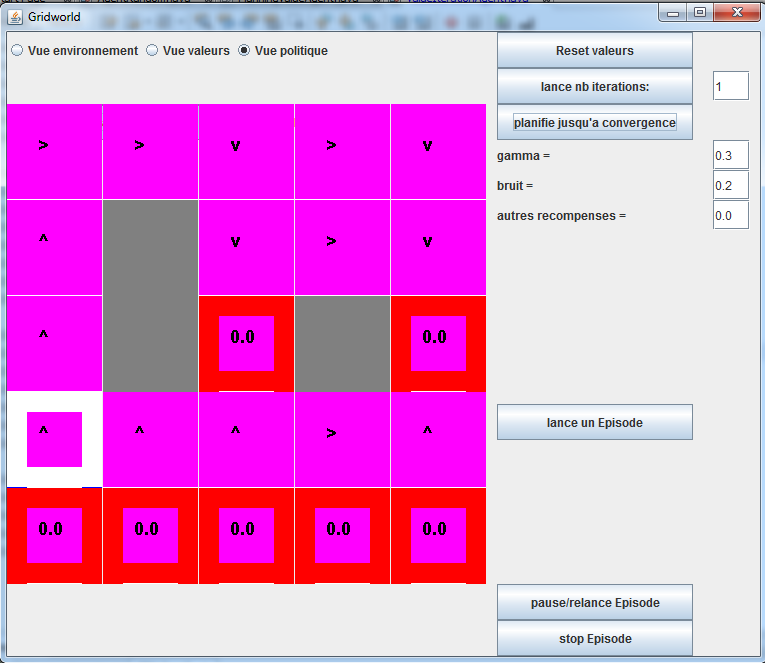
*Figure 3 : Suivre un chemin risqué pour atteindre l'état absorbant de récompense + 1*

2. bruit = 0.05, on diminue le bruit afin que notre robot puisse atteindre l'état de récompense + 10 en suivant un chemin risqué. Le bruit étant faible il y a peu de chance que le robot se "trompe" de chemin et récupère une récompense négative, il est donc assez sûr de prendre un chemin dit "risqué".



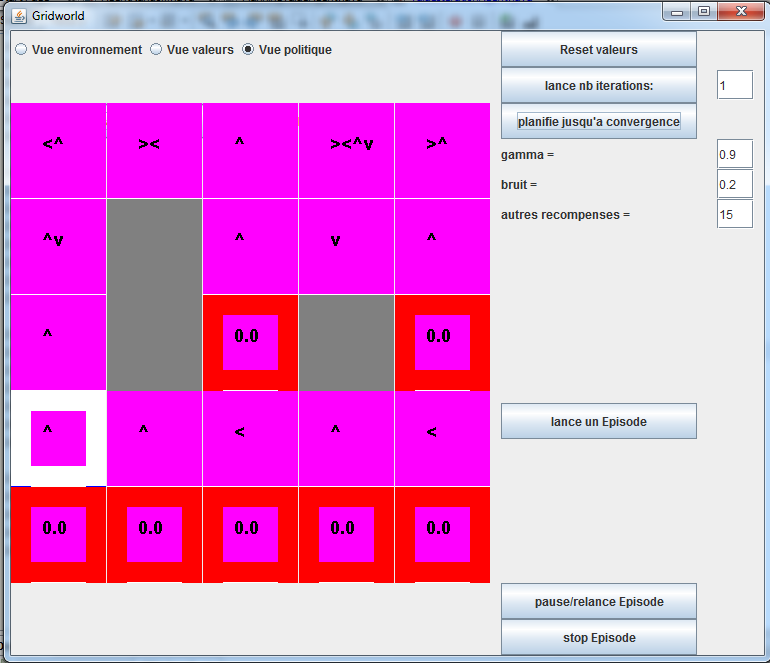
*Figure 4 : Suivre un chemin risqué pour atteindre l'état absorbant de récompense + 10*

3. γ = 0.3, notre politique privilégiera ainsi un chemin sûr pour atteindre l'état absorbant + 1. γ étant faible la politique se dirigera vers une récompense proche d'elle, de plus elle essayera d'éviter les chemins risqués qui pourraient lui faire diminuer sa récompense.



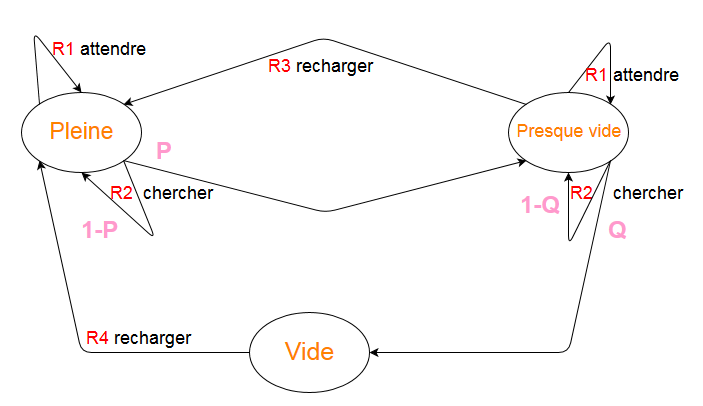
*Figure 5 : Suivre un chemin sûr pour atteindre l'état absorbant de récompense + 1*

4. Fixer rother à 15 permet de faire en sorte que notre politique évite les états absorbants, en effet une récompense de 15 est plus élevée que la récompense de n'importe quelle état absorbant, la politique n'a donc aucune raison de se diriger vers un de ces états.



*Figure 6 : Politique qui permet d'éviter les états absorbants*

## Modélisation d'un problème sous forme de MDP



*Figure 7 : Graphe du MDP*

R1 : Récompense attente R2 : Récompense chercher

R3 : Pénalité de rechargement R4 : Pénalité de rechargement opérateur

P : Probabilité de passer en batterie presque vide

Q : Probabilité de passer en batterie vide