TP Noté Intelligence Embarquée

Vincent Alves – INEM ING3

I) Environnement

Mon implémentation du Q-Learning est faite en Perl, avec quelques modules basiques. Pour une exécution sur Linux :

```
apt-get install cpanminus
cpanm Term::Screen
perl QLearning.pl
```

L'implémentation est paramétrable en éditant directement le fichier QLearning.pl :

- La carte est un tableau 2D de caractères, chaque caractère correspondant à un élément (mur, récompense, zone dangereuse)
- La position initiale du robot et les coefficients λ et ϵ sont des variables éditables juste en dessous
- La politique d'exploration est techniquement modifiable dans les fonctions vers le milieu du code, mais ce n'est pas très optimal.

Un fichier csv est écrit par le script, afin de pouvoir transcrire facilement les résultats en graphe.

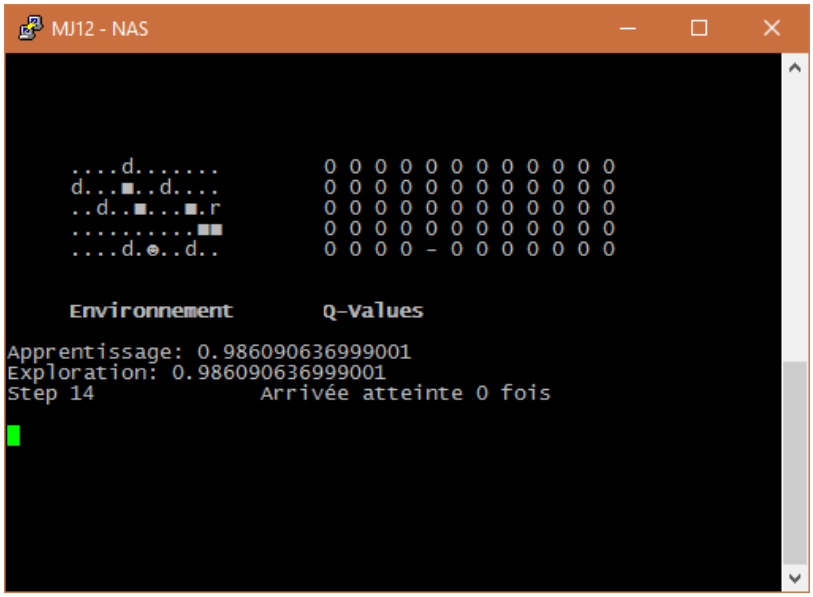

Figure 1 - Interface de l'IA une fois lancée

II) Résultats

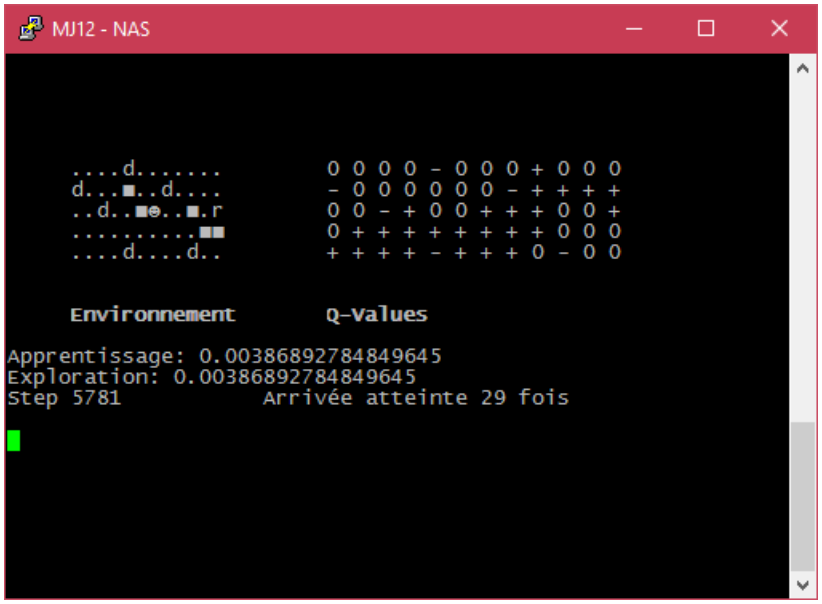

Figure 2 - 5000 itérations plus tard...

Au fur et à mesure de l'exploration par le robot, la carte des Q-Values est remplie (indiquée sur l'interface par un + pour une Q-Value positive et un – pour une Q-Value négative).

a) Test de base

L'exemple de la figure 2 est fait avec λ et ϵ = 1 et une décrémentation de 0.999*previous_value. La politique d'exploration est un banal `if (random(1) < ϵ)`.

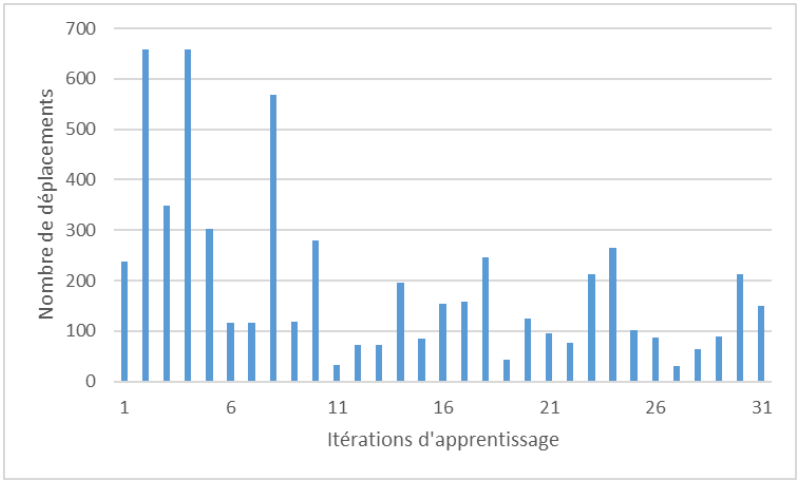

Figure 3 – Résultat de cet exemple après 31 itérations

On voit que les premières itérations d'apprentissage sont assez douloureuses (Pas moins de 100 déplacements par étape, parfois allant jusqu'à 650), mais qu'éventuellement on arrive à des itérations très rapides (moins de 30 déplacements) à partir de la 11ème.

Il y a quelques itérations plus longues/courtes que la moyenne, qu'on peut attribuer aux déplacements au hasard que l'algorithme effectue, même en exploration greedy. (Il y en a très peu, mais ils peuvent arriver)

b) Test avec des coefficients différents

Ce test est effectué avec λ et ϵ = 0.5. Les autres paramètres ne changent pas.

Figure 4 - Bloqué ?

Un souci avec le mode d'exploration greedy (que je n'avais pas vu au test d'avant car il prenait assez longtemps à s'instaurer) est que dans une situation comme celle de la figure 4, le robot a tendance à vouloir éviter la zone dangereuse…sauf qu'il n'a pas vraiment d'autre moyen de sortir.

Une bonne centaine d'étapes ont été perdues à cet endroit avant que le robot décide de prendre l'exploration au hasard et de repasser par la zone.

Quant aux résultats, en ayant arrêté la simulation après le même nombre de déplacements qu'au test précédent (Environ 5800), on peut voir que seulement 23 itérations ont eu lieu, comparé aux 31 itérations avec les coefficients à 1.

Les itérations d'apprentissage ont également plus de déplacements (les premières dépassent les 700 déplacements)

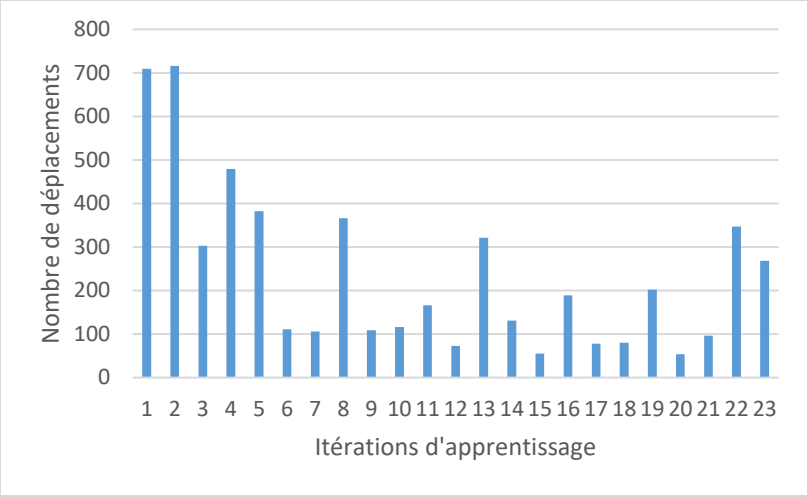

Figure 5- Résultat de ce test après 23 itérations

c) Test avec plusieurs récompenses

Pour ce test, la carte a été changée pour avoir plusieurs récompenses, dont une « cachée » derrière des zones dangereuses. La valeur des zones dangereuses a également été changée de -5 à -100, afin de décourager encore plus le robot à les franchir.

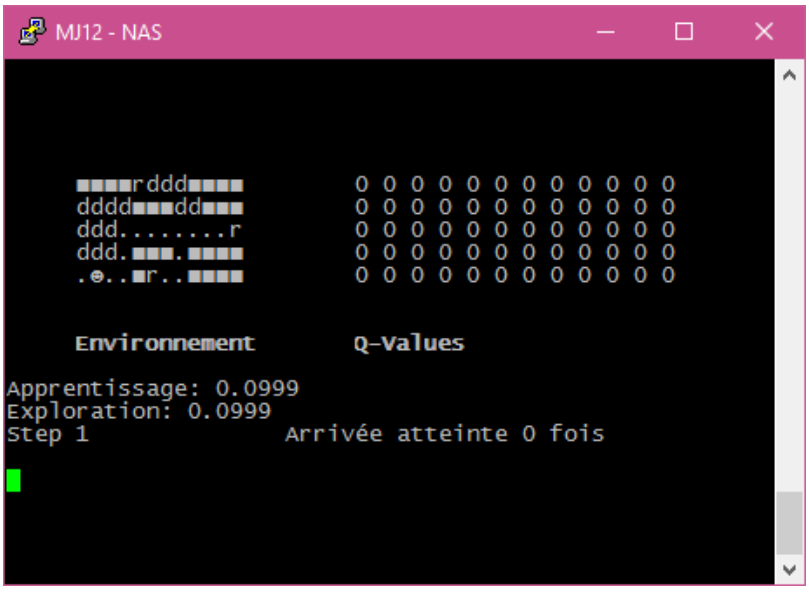

Figure 6 - La carte pour ce test

Le robot atteignait la récompense de droite sans souci, mais ne s'est jamais risqué à aller chercher les autres à cause de la valeur négative trop élevée des zones dangereuses. Les chiffres n'étant pas très intéressants, j'ai remis la valeur des zones à -5 :

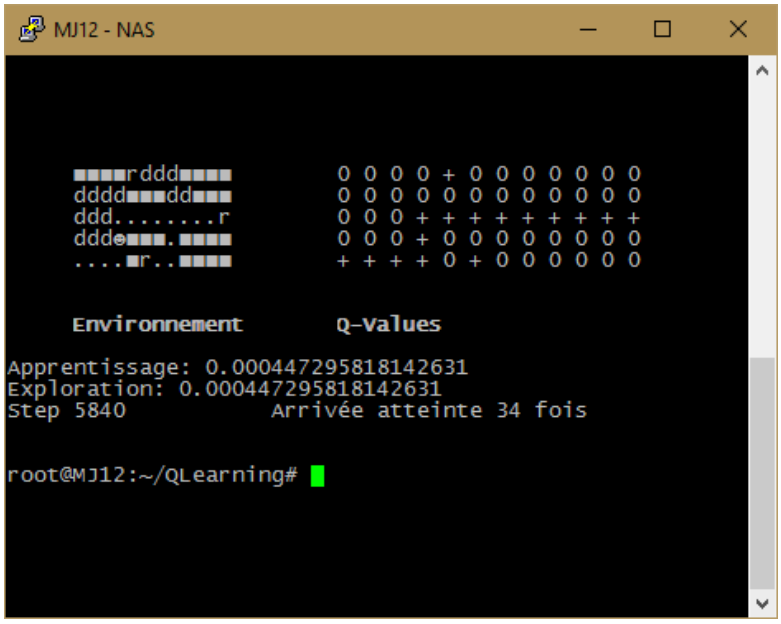

Figure 7 - Le robot atteint les autres récompenses...au moins une fois.

Les résultats ont été plus intéressants : Le robot atteignait les autres récompenses une fois, mais préférait par la suite aller directement vers la récompense de droite, plus facilement atteignable.

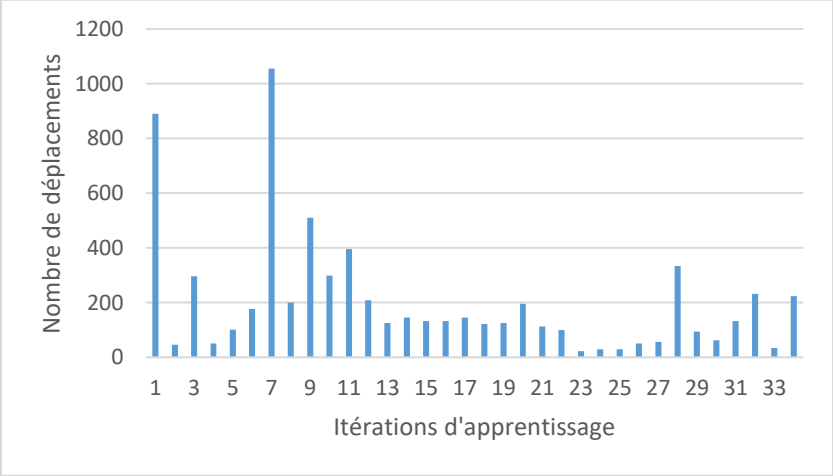

Figure 8 - Résultat du test après 34 itérations

Les itérations 2 et 4 sont relativement intéressantes : Elles étaient dues au fait que le robot a trouvé rapidement par hasard les autres récompenses.

Après cela, il s'est cependant concentré sur la récompense de droite, amenant des itérations relativement stables à partir de la 12ème.

d) Conclusion

Le Q-Learning est un algorithme d'intelligence artificielle relativement efficace, même s'il relie beaucoup sur le hasard pour la phase d'exploration initiale.

Il faut cependant garder un équilibre assez strict entre les deux modes d'exploration (random et greedy) afin d'obtenir les meilleurs résultats : Une exploration trop greedy a tendance à bloquer le robot et à l'empêcher de faire de nouvelles découvertes, tandis qu'une exploration trop hasardeuse demande nettement plus de déplacements.

III) Question Ouverte

Si on n'a pas accès à la position du robot dans son environnement, le robot pourrait construire ses ensembles S et A en se basant sur une carte qu'il construirait lui-même au fur et à mesure de son exploration.

Cependant, cela impliquerait de remettre à jour les ensembles dans leur intégralité à chaque extension de la carte, afin qu'ils restent pertinents.