La Planification de Trajectoire d'un Robot Mobile dans un Environnement Inconnu avec des Obstacles en utilisant l'Apprentissage par Renforcement

Elkamri Hajar ,
Université Abdelmalek Essaadi, ENSAH, département de mathématiques et d'informatique

Résumé:

Ce travail présente une approche d'apprentissage par renforcement visant à optimiser le comportement d'un agent intelligent au sein d'un environnement interactif. Cette méthode repose sur l'apprentissage à partir de l'expérience, où l'agent cherche à maximiser le cumul des récompenses au fil du temps. L'idée fondamentale de l'apprentissage par renforcement est d'ajuster une politique d'action après chaque interaction avec l'environnement, en s'appuyant sur le principe d'essais-erreurs.

Dans ce projet, des algorithmes d'apprentissage par renforcement, tels que SARSA et Q-Learning, ont été mis en œuvre pour permettre à l'agent de développer une politique optimale.

Cette expérimentation permet non seulement d'optimiser la performance de l'agent, mais aussi d'explorer les défis techniques liés à la convergence des algorithmes de renforcement et à la gestion de l'équilibre entre exploration et exploitation. Ce type d'approche trouve des applications dans des domaines variés tels que les jeux vidéo, la robotique, la finance et les systèmes autonomes, où la capacité d'adaptation et d'auto-optimisation est cruciale.

Mots clés : Apprentissage par renforcement, Q-learning, Sarsa , robot mobile, navigation .

1.Introduction:

La navigation d'un robot mobile peut être considérée comme la tache de déterminer une trajectoire permettant au robot de se déplacer d'une position initiale vers une autre finale désirée en évitant les obstacles, tout en respectant les contraintes cinématiques du robot et sans intervention humaine. Le problème de déterminer une telle trajectoire est connu aussi sous le nom du problème de planification de trajectoire [1].

Contrairement aux méthodes de navigation traditionnelles qui nécessitent une planification complète de l'environnement, les stratégies de commande réactive offrent des solutions intéressantes qui utilisent directement l'information issue du capteur du robot pour accomplir ses objectifs [1][2].

Initialement, la destination finale du robot est exprimée dans un environnement 2D par les coordonnées (x_c, y_c). Malheureusement, ce n'est pas toujours le cas dans les applications en extérieur comme le transport, l'agriculture, la surveillance, etc, ou la position finale est souvent requise avec une bonne précision. De ce fait, ces taches se traduisent alors par une navigation autonome de la position initiale (x_i, y_i) vers un but orienté (x_c, y_c) .

La programmation d'une stratégie de navigation autonome passe par l'emploi d'une commande intelligente, nécessitant des connaissances précises d'un expert. L'acquisition des pairs état-action est une tache importante dans la commande des robots mobiles [3]. L'application de l'apprentissage par renforcement est une solution possible pour le problème de navigation d'un robot mobile puisqu'il ne nécessite pas des exemples d'apprentissage. Le signal renforcement permet au navigateur d'ajuster sa stratégie pour améliorer ses performances. C'est une modification automatique du comportement du robot dans son environnement de navigation comme réalisé dans [4][5][6]

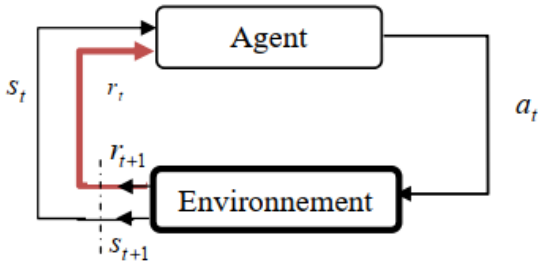
2. L'apprentissage par renforcement :

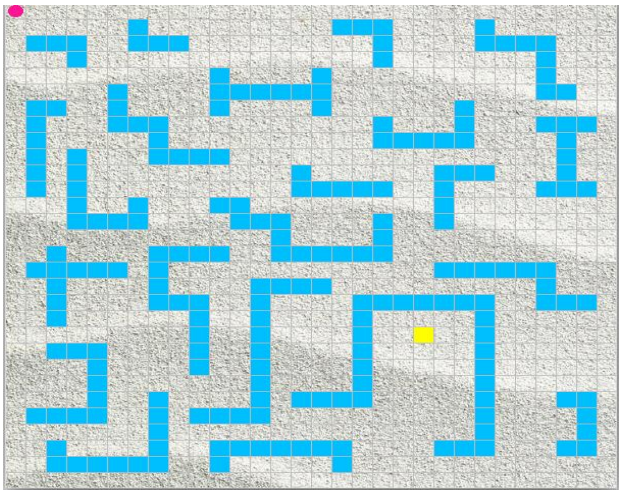
L'apprentissage par renforcement est une approche de l'intelligence artificielle qui permet l'apprentissage d'un agent par l'interaction avec son environnement; afin de trouver, par un processus essais-erreurs, l'action optimale à effectuer pour chacune des

situations que l'agent va percevoir pour maximiser ses récompenses [7][10]. C'est une méthode de programmation ne nécessitant que de spécifier les moments auxquels punir ou récompenser l'agent. Il n'est nul besoin de lui indiquer que faire dans telle ou telle situation, l'agent se charge de l'apprendre par lui même en renforçant les actions qui s'avèrent les meilleures [8]. Si on considère un agent situé dans un environnement et avec lequel il peut interagir. L'agent à

l'instant t perçoit d'une part cet environnement ($situation\ s\_t$), et peut agir ($action\ a\_t$), et d'autre part ; reçoit une récompense (r_t), comme illustré sur la figure ci dessous , ou s_t+1 et r_t+1 sont la situation et la récompense suivante.

L'agent ne connaît pas la situation dans laquelle il se trouve que par l'intermédiaire de l'historique de ses perceptions. Son but, dans le cadre de l'apprentissage par renforcement, est de trouver le comportement le plus efficace, c'est à dire savoir, dans chaque situation possible, quelle action accomplir pour maximiser de ses gains [7][9] .





3. Définition/modélisation des tâches :

État s : Nous considérons l'état à tout instant donné comme étant la configuration du système. Dans le cas d'un robot mobile, l'état pourrait être représenté par des informations telles que la position actuelle du robot, l'orientation et la présence d'obstacles dans son environnement. L'état reflète toutes les informations nécessaires pour que l'agent prenne une décision optimale à cet instant.

Action a : L'agent peut choisir parmi plusieurs actions possibles en fonction de l'état actuel. Par exemple, dans le cas d'un robot, les actions pourraient inclure "avancer", "tourner à gauche", "tourner à droite" ou "s'arrêter". L'ensemble des actions dépend de l'état dans lequel l'agent se trouve, et chaque action a des conséquences sur l'état futur.

Récompense r : La récompense est renvoyée par l'environnement après que l'agent a effectué une action. Elle mesure la qualité de l'action choisie en fonction de l'objectif de l'agent. Dans le cas de la navigation d'un robot, la récompense peut être positive lorsque l'agent se rapproche de l'objectif ou négative lorsqu'il percute un obstacle. La récompense est souvent normalisée dans une plage spécifiée, par exemple, entre [-1, 1], pour faciliter l'apprentissage de l'agent et encourager la maximisation de son score global.

Exploration vs. Exploitation et Stratégie Epsilon-Greedy

Dans le cadre des algorithmes de reinforcement learning, comme Q-Learning et SARSA, l'agent apprend une politique optimale en interagissant avec un environnement. Ces interactions consistent à choisir des actions $a \in A$ dans un état donné $s \in S$, et à mettre à jour les valeurs de fonction $Q(s,a)$ en fonction des récompenses reçues.

Exploitation :

L'exploitation repose sur le choix de l'action a_t qui maximise la valeur actuelle estimée :

$$a_t = \arg\max_{a \in \mathcal{A}} Q(s_t, a)$$

Elle favorise les décisions basées sur les connaissances accumulées, en visant directement à maximiser la récompense immédiate ou future.

Cependant, cette approche risque de piéger l'agent dans un optimum local, en négligeant des actions potentiellement meilleures mais encore inexplorées.

Exploration :

L'exploration incite l'agent à choisir une action aléatoire $a \sim U(A)$, même si elle ne semble pas optimale selon les valeurs actuelles de $Q(s,a)$. Bien que ces choix puissent temporairement diminuer les récompenses, ils permettent de collecter des informations sur l'environnement, essentielles pour évaluer les valeurs Q et découvrir des stratégies meilleures à long terme.

Stratégie Epsilon-Greedy :

Le dilemme central consiste à équilibrer entre l'exploration et l'exploitation . une solution couramment utilisée est la stratégie ϵ-greedy . Elle introduit un paramètre $\epsilon \in [0,1]$, qui contrôle la probabilité d'exploration :

• Avec une probabilité ϵ , l'agent choisit une action $a \sim U(A)$ (exploration).

• Avec une probabilité $1-\epsilon$, l'agent choisit l'action optimale actuelle

Décroissance d'Epsilon (ϵ - Decay) :

Pour optimiser l'apprentissage, ϵ est souvent initialisé à une valeur élevée ($\epsilon \approx 1$) pour encourager une exploration intense au début. Il décroît ensuite progressivement au fil des épisodes selon une fonction décroissante comme :

$$\epsilon_t = \epsilon_0 \cdot e^{-\lambda t}$$

Où $\lambda > 0$ est un facteur de décroissance. Cela permet à l'agent d'explorer largement au début, puis de se concentrer sur l'exploitation à mesure que la politique s'améliore.

Matrice de Valeurs (Table Q) :

La Table Q, ou Matrice de Valeurs Q, est un élément central des algorithmes de reinforcement learning. Elle stocke les valeurs $Q(s,a)$, qui représentent la récompense attendue si l'agent choisit une action a dans un état s, puis suit une politique donnée. Cette table est structurée en lignes (états) et colonnes (actions), permettant à l'agent d'identifier l'action optimale (celle avec la plus grande Q-value) dans chaque état. L'objectif de l'apprentissage est de mettre à jour ces valeurs pour guider l'agent vers des décisions maximisant ses récompenses à long terme.

.

$$Q(s,a) = \begin{bmatrix} Q(0,\text{haut}) & Q(0,\text{bas}) & Q(0,\text{gauche}) & Q(0,\text{droite}) \\ Q(1,\text{haut}) & Q(1,\text{bas}) & Q(1,\text{gauche}) & Q(1,\text{droite}) \\ \vdots & \vdots & \vdots & \vdots \\ Q(8,\text{haut}) & Q(8,\text{bas}) & Q(8,\text{gauche}) & Q(8,\text{droite}) \end{bmatrix}$$

4. La programmation dynamique :

Elle permet de calculer une politique optimale dans un ***PDM*** connu, en utilisant les propriétés des équations de Bellman, ou le modèle de l'environnement est connu. Ces algorithmes permettent d'estimer la fonction de valeur optimale V afin d'en déduire une politique optimale [7]

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S}, \end{aligned}$$

The bellman equation for the state value

le problème avec la programmation dynamique est que nous avons besoin d'une connaissance complète du jeu. En d'autres termes nous devons imaginer toutes les combinaisons possible d'état/action

5. Les méthodes à modèles libres

Ce sont les méthodes les plus utilisées en pratique et ne nécessitent pas un modèle de l'environnement. Il existe de nombreux algorithmes basés sur la différence temporelle comme TD(0), SARSA, l'acteur critique (ACRL) et le Q-learning. Ce dernier est l'algorithme le plus connu et le plus utilisé. Toutes ces méthodes sont basées sur les processus de décision markoviens et demandent de discrétiser l'environnement en états et les consignes sur les actionneurs en commandes échantillonnées [7]

6. Le Q-learning :

Il s'agit sans doute de l'algorithme d'apprentissage par renforcement le plus connu et le plus utilisé en raison des preuves formelles de convergence [11]12].

Le principe du Q-learning consiste à apprendre la fonction de qualité en interagissant avec l'environnement en mettant à jour itérativement la fonction courante $Q(s_t, a_t)$ à la suite de chaque transition

(s_t, a_t, r_t, s_{t+1}) [7][8].

Cette mise à jour se fait sur la base de l'observation des transitions instantanées et de leur récompense associée par l'équation suivante :

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

• s : état actuel

• a : action prise à l'état s

• s′ : nouvel état après avoir pris l'action a

• r : récompense obtenue en passant de s à s′

• α : taux d'apprentissage (learning rate), contrôle la vitesse de mise à jour de Q

• γ : facteur d'actualisation (discount factor), pondère l'importance des

récompenses futures par rapport aux immédiates

L'algorithme peut être résumé comme suit :

$Q(s,a) \leftarrow 0, \ \forall (s,a) \in (S,A)$

Pour un grand nombre ∞ faire

Initialiser l'état initial s_0

$t \leftarrow 0$

Répéter

Choisir l'action à émettre a_t en fonction de la politique de

Q et l'émettre. Observer r_t et S_{t+1}.

$Q(s_t,a_t) \leftarrow Q^{\pi}(s_t,a_t) + \alpha[r_t + \gamma \max_{a \in A(s)} Q(s_{t+1},a_t) - Q(s_t,a_t)$

$t \leftarrow t+1$

Jusqu'à $s_t \in F$

Fin pour

L'algorithme *Q-learning*

6.1 Application de Q-learning pour la navigation d'un robot mobile :

On a utilisé l'algorithme précédent pour une tache de recherche d'une cible par un robot mobile. Le robot construit une fonction Q qui définit la qualité de chaque action appliquée dans un état donné. Dans le cas discret c'est un tableau, qui comporte des coefficients qui représentent la qualité de telle ou telle action. Après l'exécution d'une action, les coefficients doivent être ajustés en fonction du résultat obtenu. Cela se fait par le biais de la formule (9). Afin de généraliser la navigation du robot mobile pour toute les situations, l'apprentissage se fait avec une position initiale aléatoire du robot et de la cible dans chaque épisode.

Plusieurs implémentations de l'algorithme Qlearning ont été réalisées en variant le nombre d'états et d'actions proposées. Le tableau suivant décrit le nombre des épisodes en fonction des espaces utilisés

Nombre d'états	Nombre d'actions	Nombre d'épisode
09	03	3260
11	03	4100
13	03	6200
19	05	8950

Tableau I

Une discrétisation plus fine permet d'améliorer le comportement du robot, mais nécessite un volume mémoire plus important et un temps d'apprentissage plus long.

Dans une tache de navigation d'un robot mobile, l'espace d'états et d'actions sont continus. Dans ce cas, l'utilisation d'autres approches pour approximer les fonctions de qualités est une solution possible. Dans la section suivante, on va utiliser les systèmes d'inférences floue pour introduire des connaissances à priori pour que le comportement initial soit acceptable et afin d'accélérer l'apprentissage

7. SARSA :

Il s'agit d'un algorithme d'apprentissage par renforcement basé sur l'interaction avec l'environnement. Contrairement au Q-learning, SARSA suit une mise à jour de la fonction de qualité *Q(s,a)* en

utilisant une séquence complète de transitions *(s_t , a_t , r_t, s_{t+1}, a_{t+1})*.

Le principe de SARSA consiste à apprendre la fonction de qualité en mettant à jour ses valeurs itérativement à chaque transition observée. La mise à jour se base sur l'équation suivante :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right]$$

L'algorithme peut être résumé comme suit :

Q(s, a) ← 0, ∀ (s, a) ∈ (S, A)

Pour un grand nombre ∞ faire
  Initialiser l'état initial s_0
  Choisir une action a_0 en fonction de la politique de Q
  t ← 0

  Répéter
    Exécuter a_t, observer r_t et s_{t+1}
    Choisir a_{t+1} en fonction de la politique de Q
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
    t ← t + 1
  Jusqu'à $s_t \in F$

Fin pour

L'algorithme SARSA

8. Comparaison des deux algorithmes :

Q-Learning et SARSA sont deux algorithmes d'apprentissage par renforcement qui permettent à un agent de maximiser ses récompenses, mais ils diffèrent dans leur approche. Q-Learning est un algorithme ***off-policy***, ce qui signifie qu'il apprend une politique optimale indépendamment des actions réellement choisies durant l'entraînement. Il met à jour les valeurs Q en utilisant la récompense attendue de la meilleure action possible. En revanche, SARSA est ***on-policy*** , ce qui signifie qu'il adapte la politique en fonction des actions réellement prises par l'agent, rendant le processus plus conservateur et sensible aux risques. En général, Q-Learning a tendance à explorer de manière plus agressive, ce qui peut le rendre plus performant dans des environnements stables. SARSA, de son côté, privilégie des politiques plus sûres, ce qui le rend mieux adapté aux environnements où un comportement plus prudent est souhaité.

Critère	Q-Learning	SARSA
Type de politique	Off-policy	On-policy
Mise à jour de Q	Basée sur la récompense de l'action optimale	Basée sur la récompense de l'action choisie
Comportement	Plus agressif, favorise l'exploration	Plus conservateur, sensible aux risques
Environnement adapté	Environnements stables	Environnements où un comportement prudent est requis
Formule de mise à jour	$Q(s,a) \leftarrow Q(s,a) + \alpha\,(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$	$Q(s,a) \leftarrow Q(s,a) + \alpha\,(r + \gamma Q(s',a') - Q(s,a))$

9. Résultats :

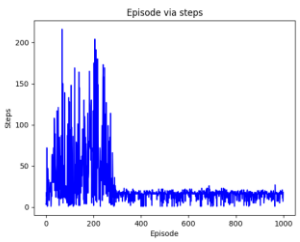
1) Q-Learning :

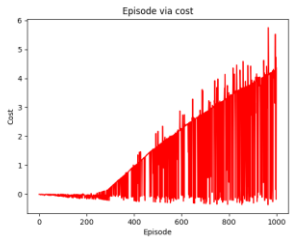
Graphique "Episode via Steps" (bleu) :

- Ce graphique montre le nombre de pas nécessaires pour atteindre l'objectif dans chaque épisode.
- Au début, le nombre de pas est élevé car le robot explore l'environnement.
- Avec le temps, le nombre de pas diminue rapidement (vers environ 50 épisodes), indiquant que le robot apprend une politique optimale pour minimiser les pas.
- Le résultat converge avec une variation limitée, ce qui montre une bonne stabilisation.

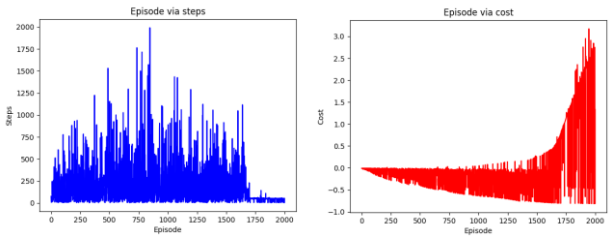
Graphique "Episode via Cost" (rouge) :

- Ce graphique représente le coût (lié à la somme des récompenses reçues) par épisode.
- Initialement, le coût augmente avec des fluctuations importantes. Cela est dû à l'exploration agressive, qui conduit à des choix sous-optimaux au début.
- Avec plus d'épisodes, le coût continue de croître avec une certaine variabilité, ce qui peut refléter une tendance à maximiser les récompenses mais avec une politique moins conservatrice.





2) Sarsa :



Graphique "Episode via Steps" (bleu) :

- SARSA, en tant qu'algorithme "on-policy", choisit des actions en suivant strictement la politique d'exploration actuelle.

- Le nombre de pas diminue plus lentement par rapport à Q-learning.
- SARSA prend plus de temps pour converger, mais une fois stabilisé, le nombre de pas reste légèrement plus élevé. Cela reflète un apprentissage plus conservateur qui privilégie les actions sûres.

Graphique "Episode via Cost" (rouge) :

- Le coût augmente avec les épisodes, mais la progression est plus douce et régulière que pour Q-learning.
- SARSA tend à adopter une politique plus stable mais légèrement sous-optimale, car il évite les scénarios risqués qui pourraient maximiser les récompenses.

10. Analyse comparative pour la navigation d'un robot :

- **Q-learning :**

  Converge plus rapidement vers une politique optimale.
  Est plus agressif dans son apprentissage, ce qui peut parfois conduire à des actions risquées dans des environnements dynamiques.
  Convient pour des tâches où la rapidité d'apprentissage et la performance optimale sont cruciales.

- **SARSA :**

Apprend de manière plus conservatrice en prenant en compte les risques liés à la politique actuelle.
Convient pour des environnements incertains où la sécurité et la stabilité sont plus importantes que l'optimalité.

11. Conclusion:

résultats mettent en évidence l'importance de bien choisir l'algorithme en fonction des caractéristiques de l'environnement et des objectifs de l'application. Ainsi, le choix entre Q-Learning et SARSA ne dépend pas uniquement de la recherche d'une politique optimale, mais aussi de la stratégie souhaitée pour atteindre cet objectif dans des contextes spécifiques. Ce projet a ainsi renforcé notre compréhension des mécanismes du reinforcement learning et pose des bases solides pour des recherches futures visant à optimiser et adapter les comportements des agents dans des environnements variés.

Remerciement :

Je tiens à exprimer ma profonde gratitude à mon professeur AZIZ KHAMJANE pour son encadrement, ses conseils avisés et son soutien constant tout au long de ce projet. Ses explications claires et son expertise en machine learning ont été déterminantes pour la réussite de mon travail.

References:

[1] Patrick Reignier, " *Pilotage réactif d'un robot mobile, Etude de lien entre la perception et l'action*". Thèse de Doctorat, institut national polytechnique de Grenoble (INPG), 1994.
[2] Bernard Bayle, " *Robotique Mobile* ", Ecole Nationale Supérieure de Physique de Strasbourg, Université Louis Pasteur, 2007.
[3] Glorennec Pierre Yves, " *Algorithmes d'Apprentissage Pour Systèmes d'inférence Floue*". Editions Hermes, 1999.
[4] Nelson H.C.Yung and Cang Ye, " An Intelligent Mobile Vehicle Navigation Based on Fuzzy Logic and Reinforcement Learning". *IEEE Transactions on Systems, Man and Cybernetics,* vol 29. no.2, pp.314-321, 1999.
[5] Cang Ye, Nelson, H.C,Yung, Danwei Wang, " A Fuzzy Controller with Supervised Leaning Assisted Reinforcement Learning Algorithm For Obstacle Avoidance". *IEEE Transaction on Systems, Man, And Cybernetics-Part B: Cybernetics,* vol.33, no.1,pp.1-11, avril 2003.
[6] Glorennec Pierre Yves, Lionnel Jauffe, " A Reinforcement Learning Method for an Autonomious Robot". *Proceedings of the First Online Workshop on Soft Computing*, 1996.
[7] Richard S. Sutton and Andrew G. Barto, " *Reinforcement Learning: An Introduction* ''. MIT Press, Cambridge, Massachusetts, 2005.
[8] Glorennec Pierre Yves," Reinforcement Learning: an Overview ". *ESIT 2000, Aachen, Germany,* 14-15 septembre 2000.

[9] Richard S. Sutton and Andrew G. Barto,Ronnald J.Williams, ” Reinforcement Learning is direct adaptive Optimal Control ”, *Proc of ACC,* Boston, June 1991.

[10] Richard S. Sutton, ” Learning to predict by the Methods of Temporal Differences” .*Machine Learning,*3,p.9-44*,* 1988.

[11] Watkins C., Dayan P. ” Technical Note, Q-Learning ”**,** *Machine Learning*, 8, p.279-292, 1992.

[12] Kaelbling,L.P.,Littman,M.L., and Moore,A.W, ” Reinforcement Learning: A survey”, *Journal of Artificial Intelligence Research,* vol 4,pp.237-285, 1996.