APPRENTISSAGE CONDITIONNÉ PAR DES BUTS EN BBRL

UE de projet M1

25 mai 2023

Roxane Cellier – Yi Qin – Zhenyue Fu

Objectif

Objectif

L'objectif de ce projet était d'étudier, de comprendre, puis d'implémenter et de fusionner différents algorithmes de l'apprentissage par renforcement :

- Q-Learning (états discrets)
- Hindsight Experience Replay
- Deep Q-Network (états continus)

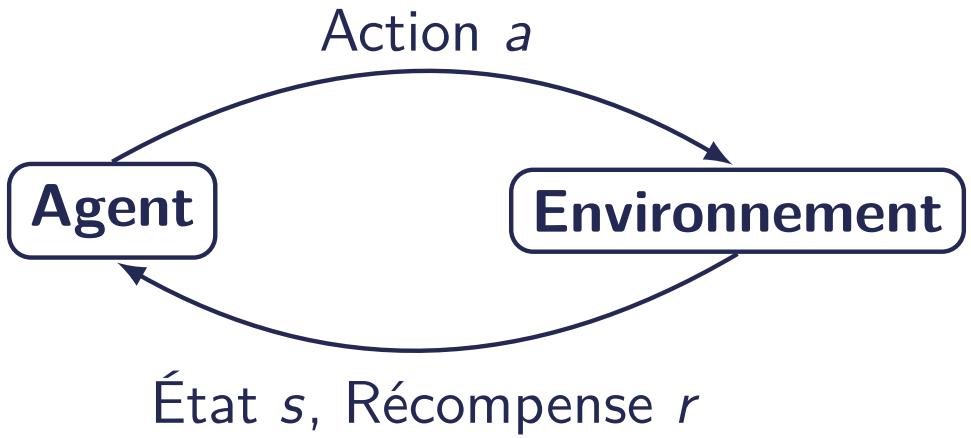
L'application et les tests de nos implémentations se sont faites sur deux environnements différents, pour des buts précis :

- La recherche d'une politique optimale de déplacement dans un labyrinthe
- Le maintien à l'équilibre d'un pendule inversé à une coordonnée x

Principes

Apprentissage par Renforcement

- L'agent observe l'état actuel, et choisit une action
- L'action influe sur l'environnement, qui change d'état
- L'environnement renvoie alors à l'agent une récompense, ainsi que le nouvel état
- L'agent lit la récompense et détermine l'utilité de l'action par rapport à l'état



Objectif : Trouver une politique de choix d'actions maximisant les récompenses reçues

Q-Learning

Algorithme off-policy mono-objectif

- Deux stratégies considérées : $\epsilon - greedy$ et *softmax*
- Fonction Q : mesure la qualité d'une action exécutée dans un état donné
- Q-Table (2D) : stocke les gains potentiels

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

Goal-Conditioned Reinforcement Learning

En apprentissage par renforcement, un agent peut apprendre à résoudre un ensemble de tâches en apprenant des politiques conditionnées par des buts.

- L'espace des états ($\mathcal{S}$)
- L'espace des actions ($\mathcal{A}$)
- La fonction de transition d'état ($\mathcal{P}$)
- L'espace des objectifs ($\mathcal{G}$)
- L'application (ϕ) de l'état actuel à l'espace des objectifs

Hindsight Experience Replay

Principe : À chaque fin de marche, considérer l'état final comme s'il était notre objectif initial

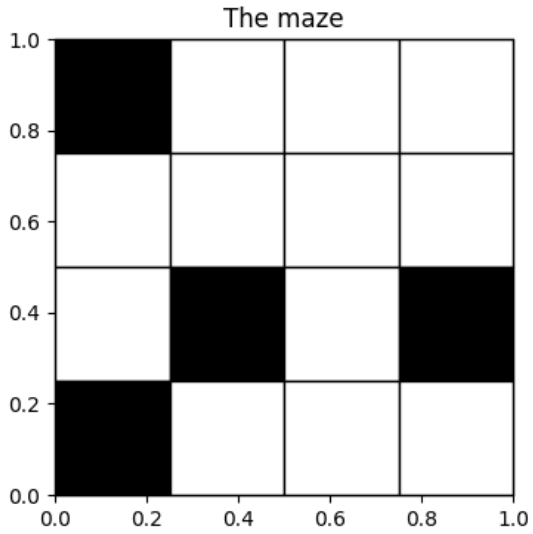
- Re-traitement des transitions effectuées
- Calcul des nouvelles récompenses à chaque pas
- Accélère l'apprentissage

Replay Buffer

- Stocke les transitions effectuées par l'agent
- $(s_t, a_t, r_t, s_{t+1}, but)$

Processus Expérimentaux

Environnement Labyrinthe



- Chaque case représente un état : espace d'états fini
- Quatre actions possibles : haut, bas, droite, gauche
- Murs infranchissables

Q-Learning conditionné par les buts

Objectif : déterminer la meilleure politique de déplacement pour chacun des buts possibles

- Choix d'un but à chaque début de marche
- Ajout d'une limite M du nombre de pas
- Ajout d'une dimension à la Q-Table

$$Q(s, but, a) \leftarrow Q(s, but, a) + \alpha[r + \gamma max_{a'} Q(s', but, a') - Q(s, but, a)]$$

Implémentation : modification de l'environnement

```
r""" modifie les paramètres de la MDP du MazeMDPEnv en fonction du
but à atteindre """
    def maj_goal(mdp: MazeMDPEnv, but, r_list, P_list):
        # modification de la récompense
        mdp.mdp.r = r_list[but]

        # modification des transisitons
        mdp.P = P_list[but]
        mdp.mdp.P = P_list[but]

        # modifie le plotter
        mdp.mdp.plotter.terminal_states = [but]
```

Résultats - Q-Learning 3D I

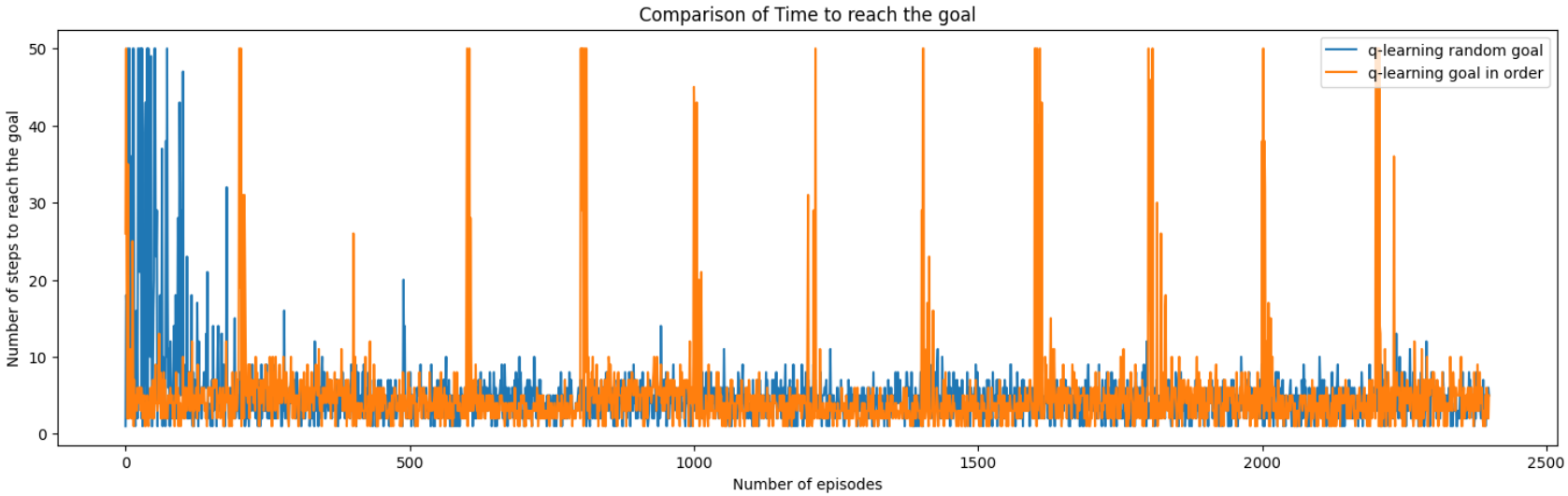


Figure 1 – Comparatif entre tirage aléatoire et apprentissage dans l'ordre

Résultats - Q-Learning 3D II

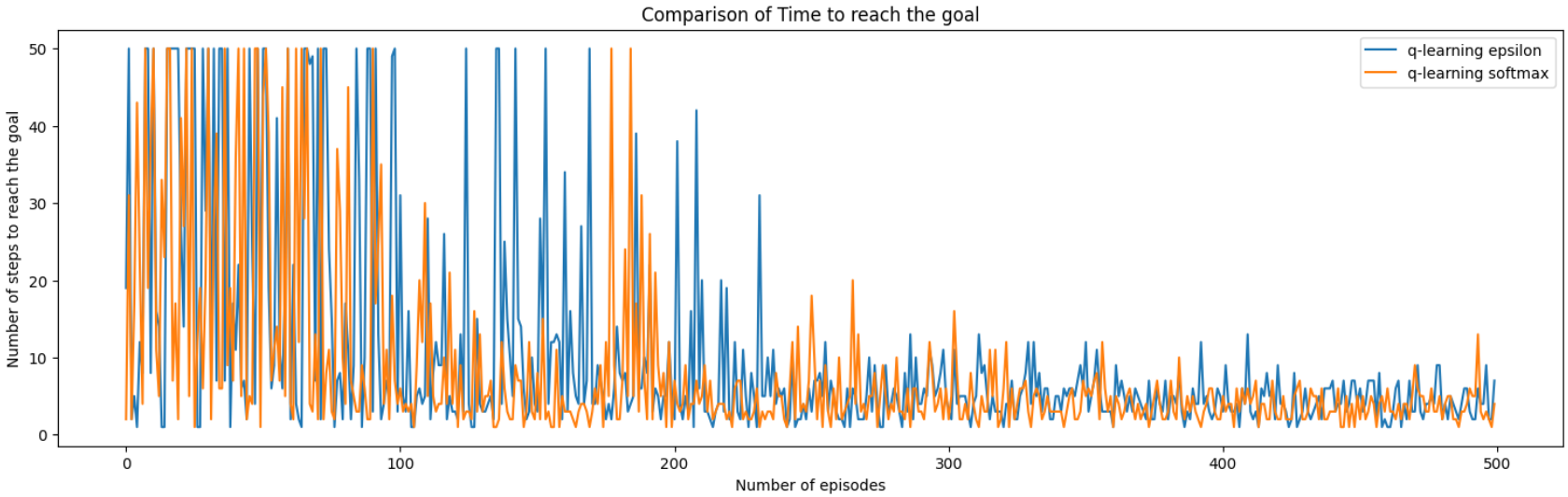


Figure 2 – Comparatif entre ϵ – *greedy* et *softmax*

Résultats - Q-Learning 3D III

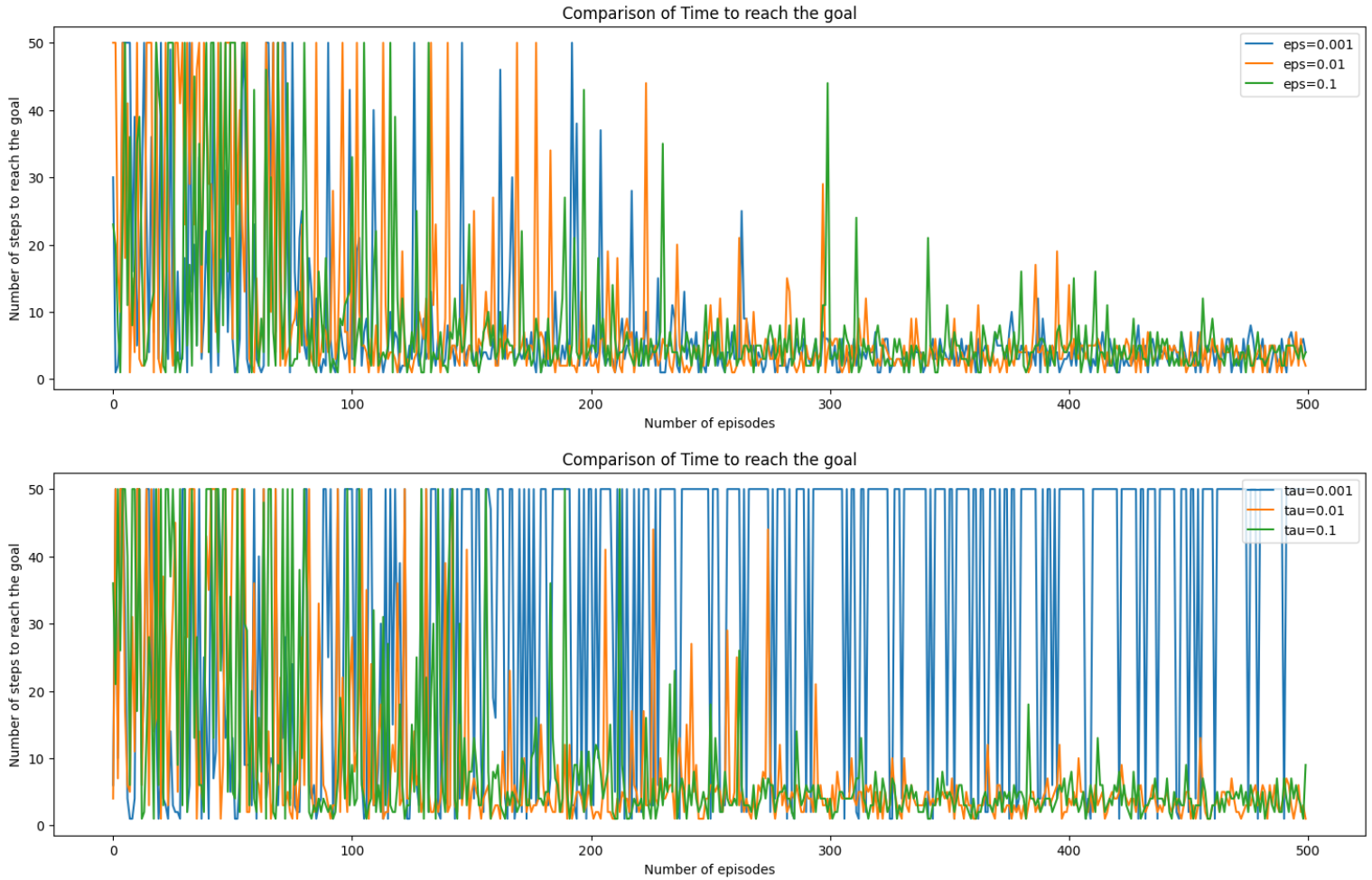


Figure 3 – Comparatifs des différentes valeurs pour ϵ et τ

HER tabulaire

L'implémentation de Hindsight Experience Replay sur l'algorithme précédent s'effectue en trois temps :

- L'utilisation du Replay Buffer pour enregistrer chaque pas pendant le déroulé de l'épisode
- L'échantillonnage et la mise à jour de nouveaux objectifs sur chaque transition du Replay Buffer
- L'extraction d'un sous-ensemble de transitions pour mettre à jour les valeurs de la Q-Table

Implémentations I

```
### boucle de l'épisode déroulée de manière normale
while not done:
    # ajout de l'état actuel au Buffer du chemin
    sb.append(s)
    # Draw an action using an epsilon-greedy policy
    a = egreedy(Q[:, but, :], s, epsilon)
    # ajout de l'action effectuée au Buffer des actions
    ab.append(a)

    # Perform a step of the MDP
    [s_prime, r, done, _] = mdp.step(a)

    # Replay Buffer pour enregistrer les transitions :
    # tuple (s, a, r , s', but)
    rb.append( (s, a, r, s_prime, but) )

    # Update the agent position
    s = s_prime
```

Implémentations II

```
### boucle de l'expérience replay sur chaque step
for t in range(mdp.mdp.timestep -1):
    # choix des buts annexes parmi le chemin parcouru
    nb_buts = min(NB_BUTS_MAX, len(sb))
    goal = random.sample(sb, nb_buts)

    # récupération des états et de l'action de la transition au temps t
    s = sb[t]
    a = ab[t]
    # pour chaque but annexe, ou ajoute la transition au Replay Buffer
    for g in goal:
        maj_goal(mdp, g, r_list, P_list)
        # placement de l'agent dans l'état s
        mdp.mdp.current_state = s
        # calcul de la nouvelle récompense
        [s_prime, r, _, _] = mdp.step(a)
        # ajout au Buffer
        rb.append( (s, a, r, s_prime, g) )
```

Implémentations III

```
### boucle du calcul des nouvelles valeurs de Q par HER
for _ in range(NB_REPLAY):
    # tirage du mini batch parmi les transitions du Replay Buffer
    taille_batch = min(TAILLE_BATCH_MAX, len(rb))
    rb_batch = random.sample(rb, taille_batch)

    # pour chaque transiton du mini batch
    for  i in range(taille_batch):
        # récupération des données de la transition
        s, a, r, s_prime, but = rb_batch[i]

        # calcul des nouvelles valeurs de Q
        delta = r + mdp.gamma * np.max(Q[s_prime, but]) - Q[s, but, a]
        Q[s, but, a] = Q[s, but, a] + alpha * delta
```

Résultats - HER I

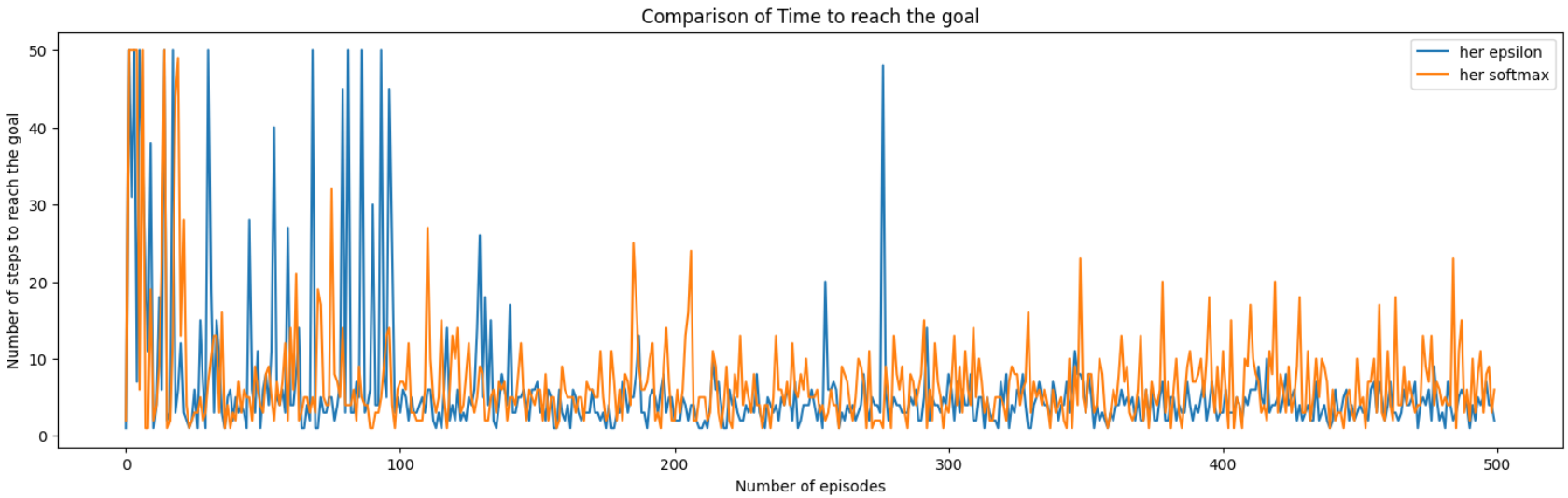


Figure 4 – Comparatif entre ϵ – *greedy* et *softmax*

Résultats - HER II

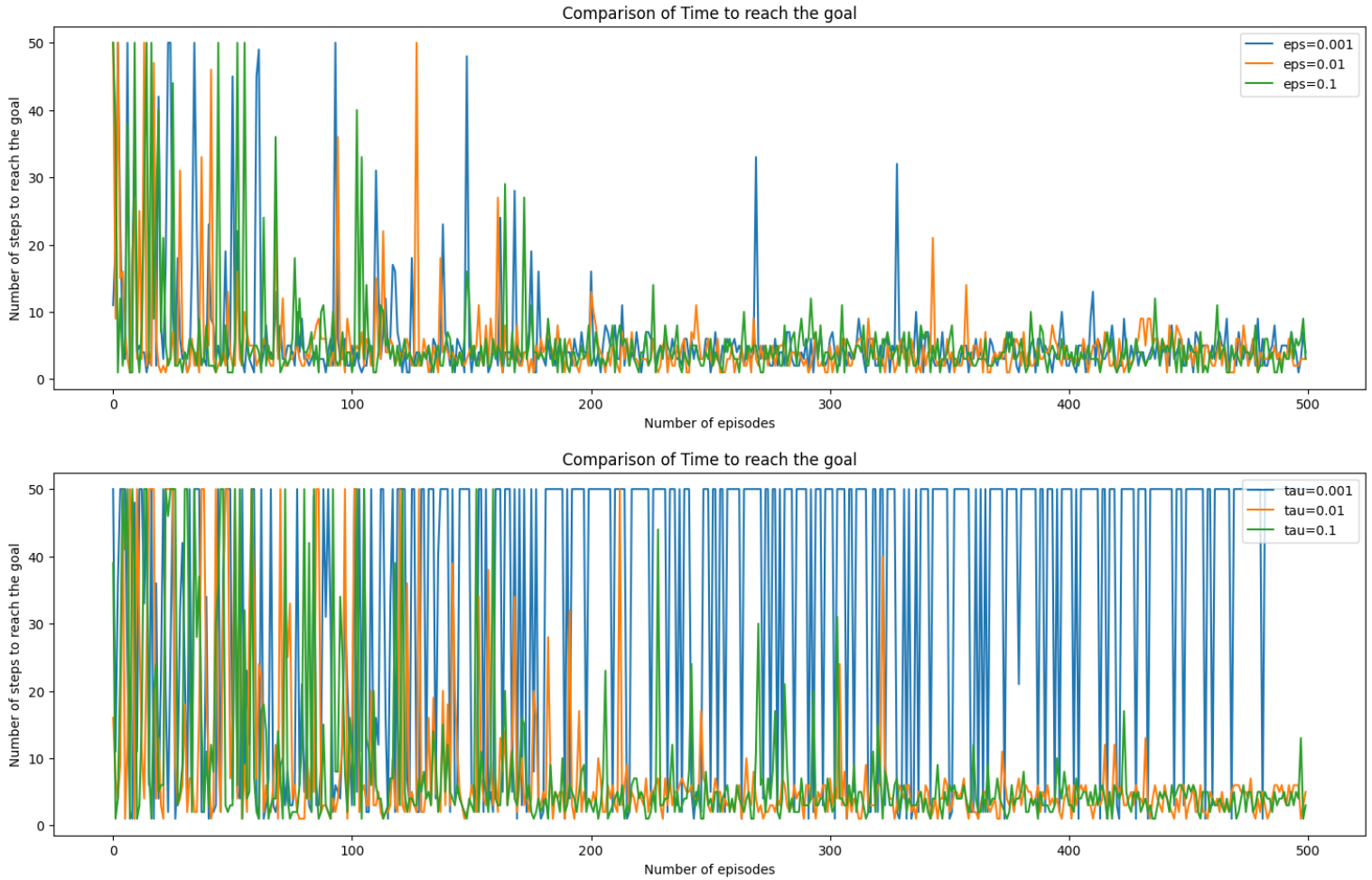


Figure 5 – Comparatifs des différentes valeurs pour ϵ et τ

Comparatif Q-Learning avec/sans HER

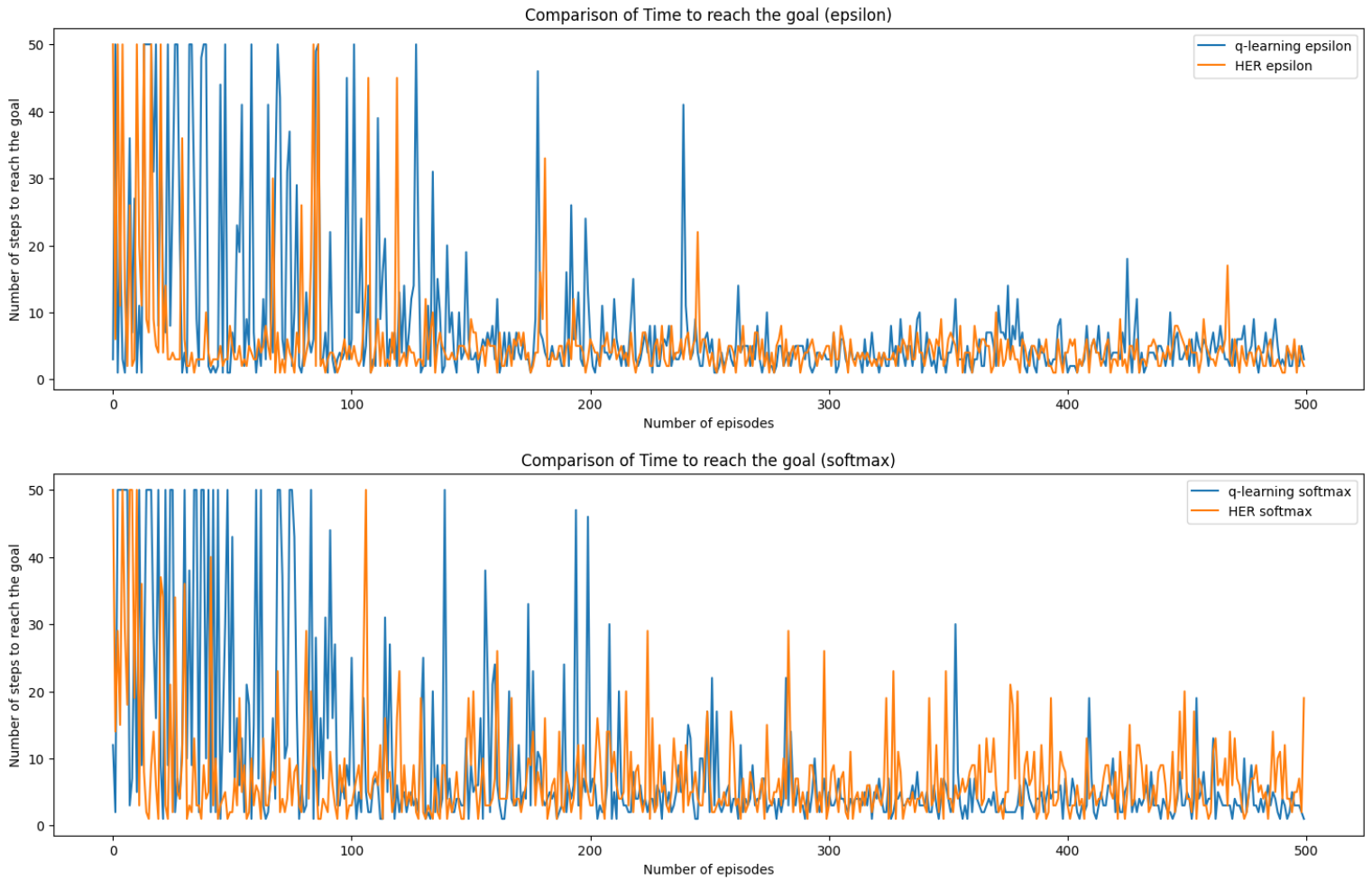


Figure 6 – Comparatifs Q-Learning/HER par stratégie d'exploration

Environnement CartPole

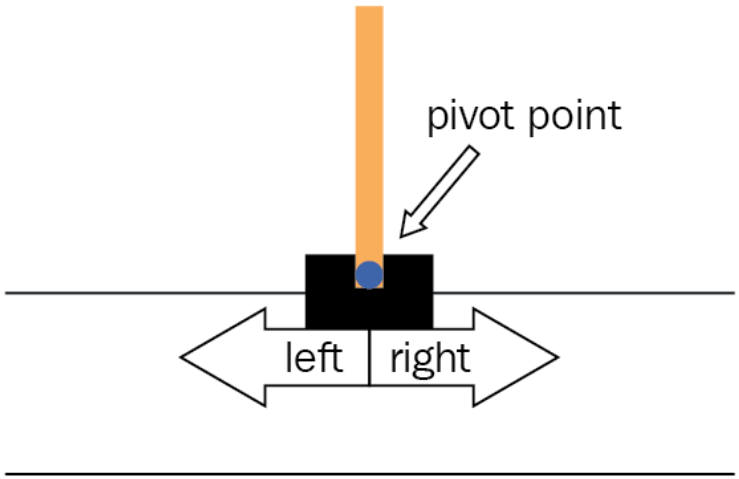


Figure 7 – Cart Pole

- Notre but est une coordonnée x du chariot
- La récompense ne peut être obtenue que lorsque le chariot est près du but

BBRL

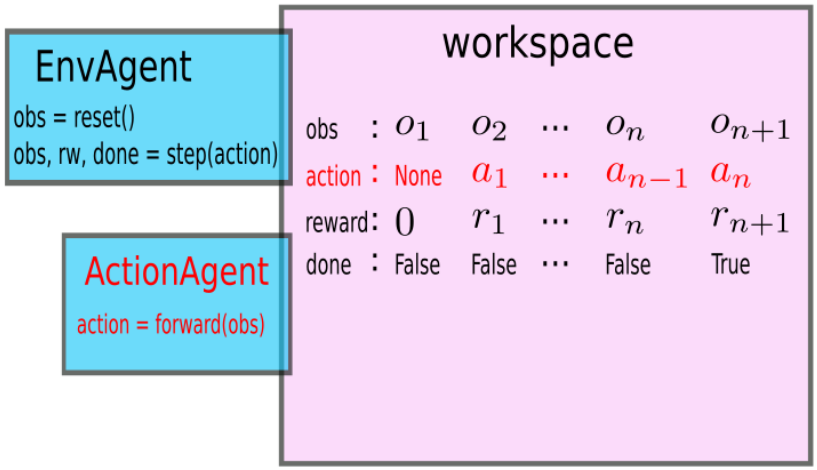


Figure 8 – WorkSpace

Dans BBRL(BlackBoard Reinforcement Learning), le modèle est mis en œuvre sous la forme d'un "WorkSpace" où plusieurs agents lisent et écrivent des informations. Cela est représenté dans la figure 8.

Agents - BBRL

DQN

- env_agent
- critic_agent
- explorer_agent

avec le but

- goal_agent
- env_agent
- critic_agent
- explorer_agent
- reward_agent

avec HER

- her_agent
- reward_agent

Résultats - BBRL

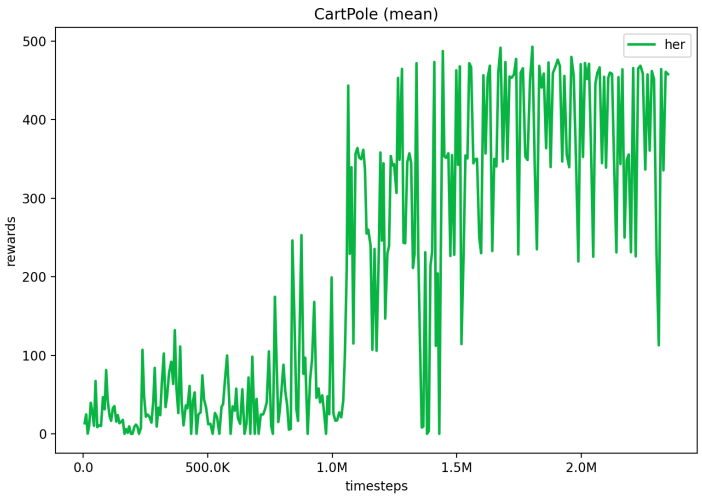


Figure 9 – HER

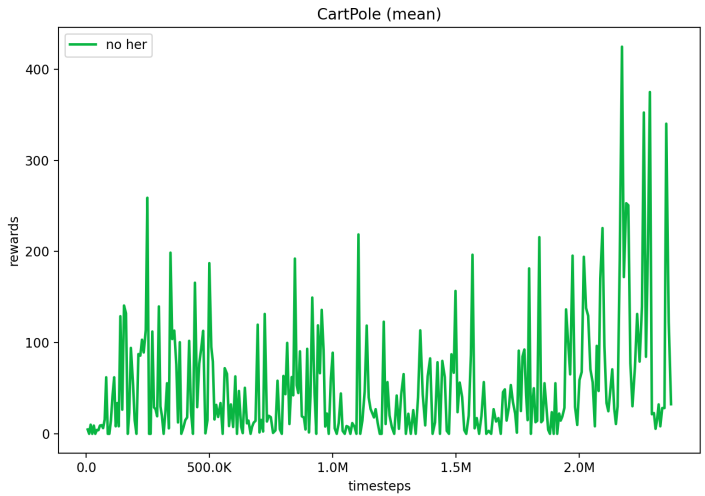


Figure 10 – Sans HER

Conclusions

Conclusions

Nos études nous ont permises d'évaluer et de comparer différentes implémentations, pour en tirer certaines hypothèses :

- L'apprentissage semble plus efficace en déterminant aléatoirement le but à chaque épisode
- Les valeurs ϵ et τ impactent fortement la vitesse d'apprentissage
- Dans le labyrinthe, HER semble plus efficace avec $\epsilon - \textit{greedy}$ qu'avec *softmax*
- Pour un problème complexe (CartPole), l'utilisation de HER semble permettre d'obtenir de meilleurs résultats

Conclusions

Sur le plan personnel, ce projet nous a permis de :

- Prendre en main des algorithmes et concepts complexes et récents de l'apprentissage par renforcement
- Étudier en profondeur les tenants et aboutissants de l'apprentissage conditionné par des buts
- Implémenter des algorithmes d'apprentissage sur des environnements et objectifs concrets
- Comprendre la difficulté à étudier en détail et à modifier une librairie complète déjà existante

Problématiques

Difficultés

- Étude en profondeur du code de la librairie BBRL
- Implémentation de HER dans BBRL pour un environnement continu de type labyrinthe

Limitations

- Environnements et paramètres très restreints
- Temps d'entraînement plus long
- Ressources de calculs plus importantes

Ouvertures

- Ajustement dynamique des paramètres ϵ et τ
- Implémentation de d'autres stratégies d'exploration
- Implémentation de HER à d'autres algorithmes d'apprentissage par renforcement (ex : Actor-Critic)
- Étude de performances sur d'autres types d'environnements et de problèmes