Rapport TME2 : Apprentissage par Renforcement Tabulaire

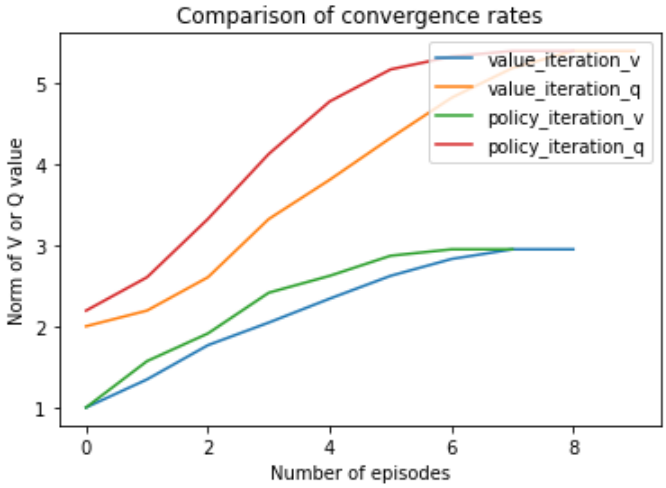
CELLIER Roxane - MOULIN-ROUSSEL Lou

1

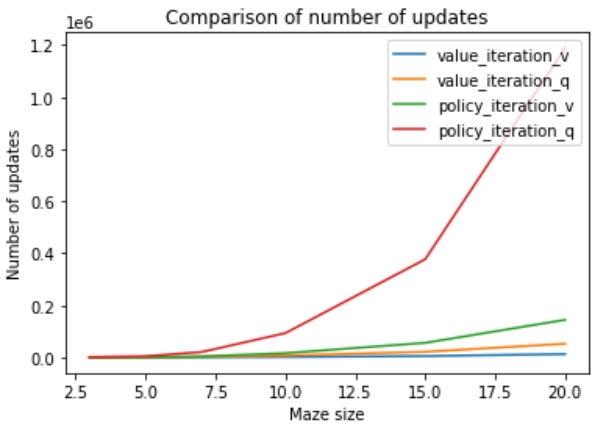
1. Dynamic programming

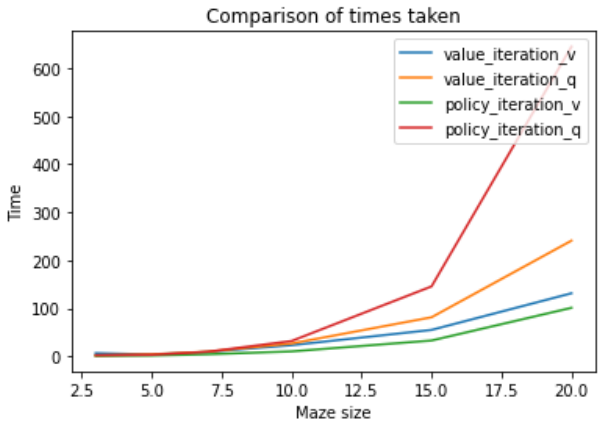
Dans cette première partie, notre but est de comparer l'efficacité de différentes méthodes de programmation dynamique utilisées par les fonctions V et Q. Pour ce faire, nous allons visualiser le temps de convergence des différentes méthodes, en considérant le nombre d'épisodes et la norme des matrices résultantes. Nous allons également chercher à compter le nombre de modifications apportées aux matrices V et Q, ainsi que le temps d'exécution de chacun des algorithmes, pour différentes topologies du labyrinthe. Nous allons donc séparément modifier la taille de l'environnement, et la proportion de cases représentant des murs.

La première étude nous renvoit donc le graphique présenté ci-dessous. Nous pouvons voir assez vite que la norme du retour de la fonction Q converge vers une valeur plus élevée que la fonction V. Nous pouvons également observer que la méthode policy iteration semble converger légèrement plus rapidement que value iteration.



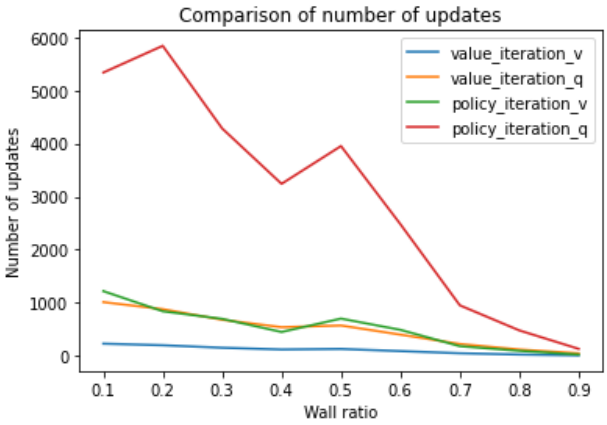
Nous allons maintenant ici comparer le nombre de modifications apportées aux matrices V et Q ainsi que le temps d'exécution des algorithmes en fonction de la taille du labyrinthe, pour des labyrinthes ayant toujours la même proportion de murs : 0.2.





Nous remarquons que dès que la largeur du labyrinthe dépasse 7, le nombre de modifications ainsi que les temps d'exécution de la méthode policy iteration par la fonction Q sont très largement supérieurs à ceux des autres méthodes. Le nombre d'opération semble croître de façon exponentielle, comparée aux autres méthods dont l'augmentation est plus douce. En terme de temps d'éxecution, les valeurs des deux algorithmes utilisant la fonction Q augmentent plus vite que ceux utilisant V.

Puis nous comparons les nombres de modifications et les temps d'exécution pour des labyrinthes de taille 5×5, mais dont les proportions de murs sont différentes.



On remarque que le nombre de modifications diminue lorsque la proportion de mur augmente pour toutes les méthodes. En effet, plus il y a de murs dans le labyrinthe moins il y a de chemins possibles à explorer.

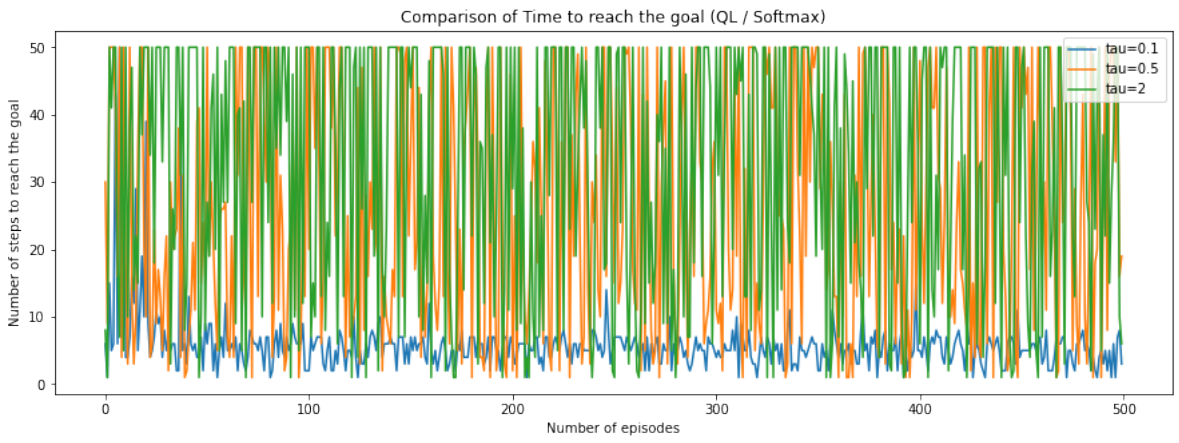
2. Tabular Reinforcement Learning

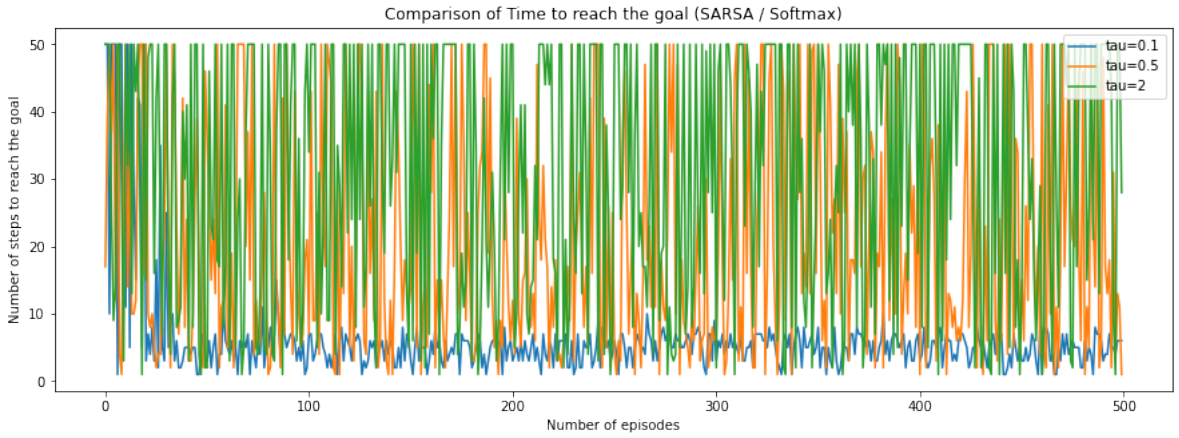
Dans cette partie, nous cherchons à étudier et comparer les algorithmes de Q-Learning et SARSA selon deux différentes stratégies d'évolution, Epsilon-greedy et Softmax. Pour cela, nous allons premièrement considérer le taux de convergence et le nombre d'étapes effectuées, pour des variations des paramètres $epsilon$ et tau des stratégies d'évolution. Les tests ont été effectués sur un environnement de taille 5×5, avec un ratio de murs du labyrinthe de 0.2. Nous avons considéré cette simple boucle pour nous permettre d'étudier

les différences (ici pour l'algorithme Softmax) :

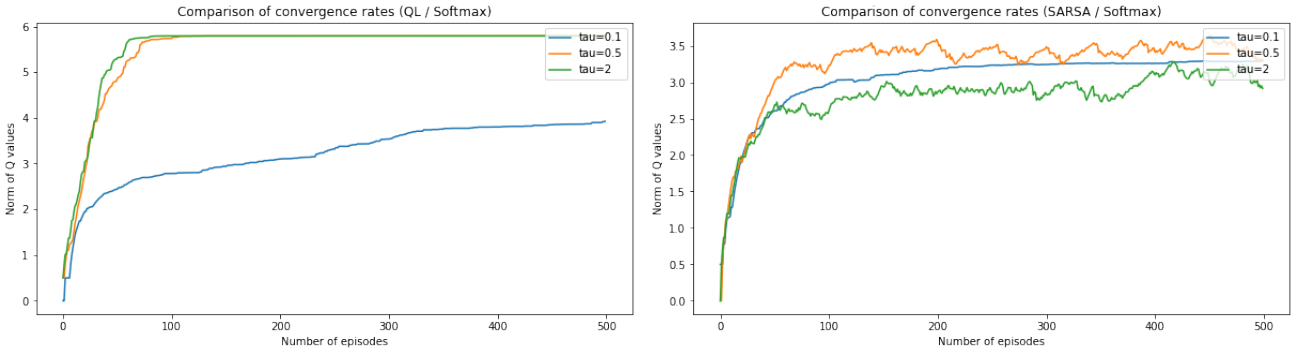
```
def plot_ql_sarsa_soft(env, tau, nb_episodes, timeout, alpha, render):

    plt.figure(figsize=(15,5))

    for t in tau:
        q, _, t_list2 = q_learning_soft(env, t, nb_episodes, timeout, alpha,
    render)
        plt.plot(range(nb_episodes), t_list2, label='tau='+str(t))

    plt.xlabel('Number of episodes')
    plt.ylabel('Number of steps to reach the goal')
    plt.legend(loc='upper right')
    plt.title("Comparison of Time to reach the goal (QL / Softmax)")
    plt.show()

    plt.figure(figsize=(15,5))

    for t in tau:
        q, _, t_list4 = sarsa_soft(env, t, nb_episodes, timeout, alpha, render)
        plt.plot(range(nb_episodes), t_list4, label='tau='+str(t))

    plt.xlabel('Number of episodes')
    plt.ylabel('Number of steps to reach the goal')
    plt.legend(loc='upper right')
    plt.title("Comparison of Time to reach the goal (SARSA / Softmax)")
    plt.show()
```

Nous allons cependant ne présenter que les graphes nous permettant une analyse intéressante des résultats. Un premier résultat visible immédiatement est l'impact de la valeur de tau sur les résultats en nombre d'étapes de la stratégie Softmax :



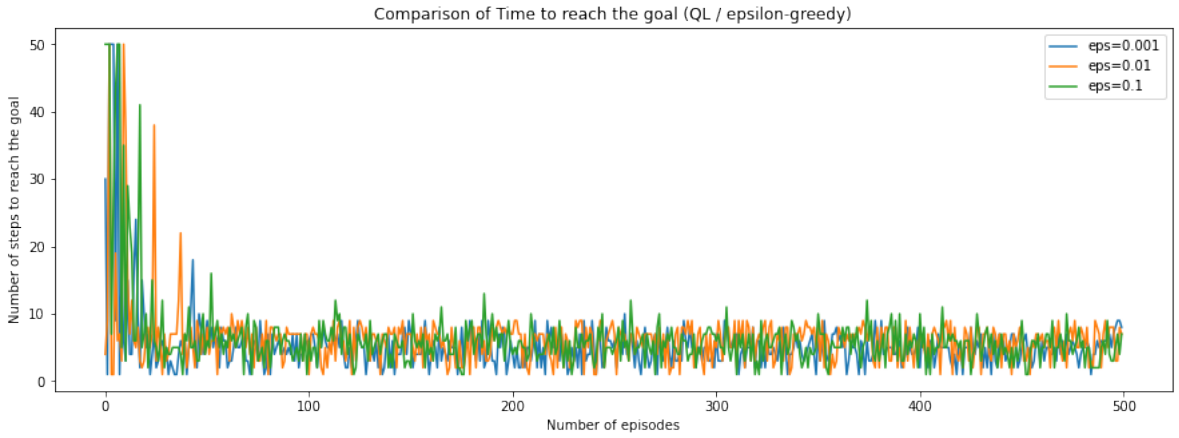


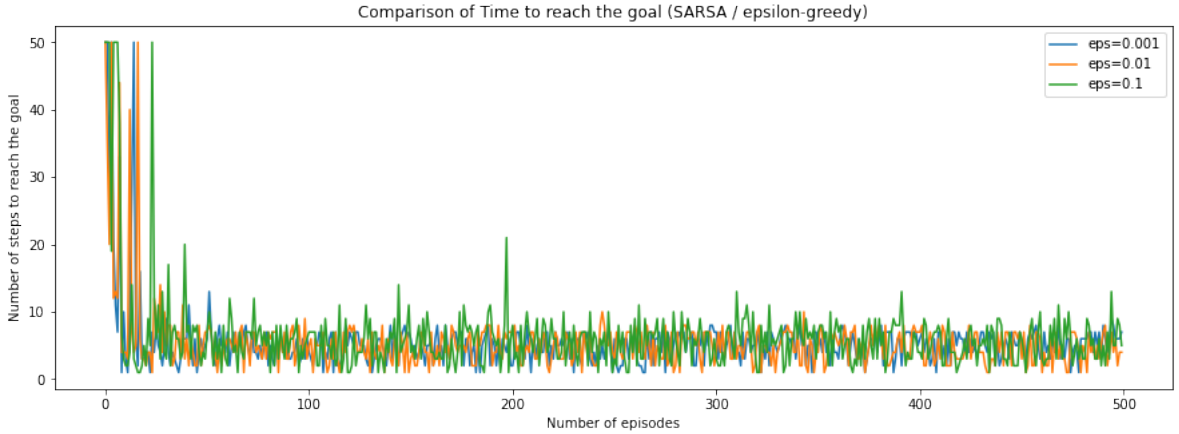
Nous pouvons facilement remarquer qu'une valeur élevée de tau rends l'utilisation de Softmax non optimale, peu importe l'algorithme d'apprentissage utilisé. Avec un tau choisi proche de zéro, le nombre d'étapes par épisode baisse très rapidement. Nous voulons maintenant observer le taux de convergence de la norme de la Q-table résultante de l'application de l'algorithme avec différentes valeurs de tau.



Nous pouvons constater que pour une valeur de paramètre plus élevée, la convergence de norme est beaucoup plus rapide sur l'algorithme de Q-Learning, mais pour une norme des valeurs beaucoup plus élevée qu'avec une valeur proche de 0. Cependant les courbes sont moins brusques pour SARSA, et semblent correspondre à une convergence vers une norme de même ordre pour les différentes valeurs du paramètre. SARSA semble tout de même tendre en norme vers une valeur plus faible que l'algorithme QL.

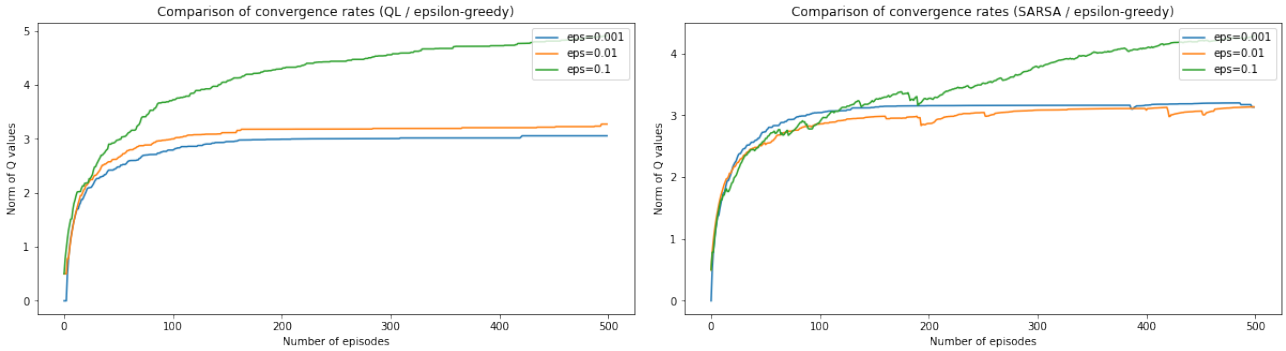
Concernant la stratégie Epsilon-greedy, les résultats sur les valeurs étudiées sont moins évidents à interpréter. Une fonction similaire à celle présentée précédemment nous renvoit les graphiques suivants :





Nous pouvons observer cependant que l'algorithme de Q-Learning semble se stabiliser de manière plus uniforme. En effet, il survient parfois pour SARSA des pics du nombre d'étapes pour un $epsilon$ plus élevé. Ce n'est cependant pas à considérer comme une généralité.

Nous observons maintenant les vitesses de convergences :

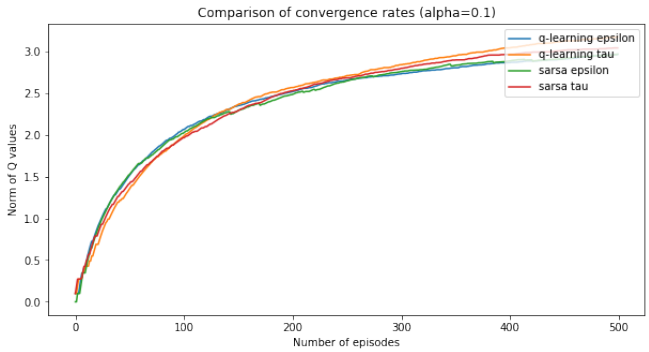


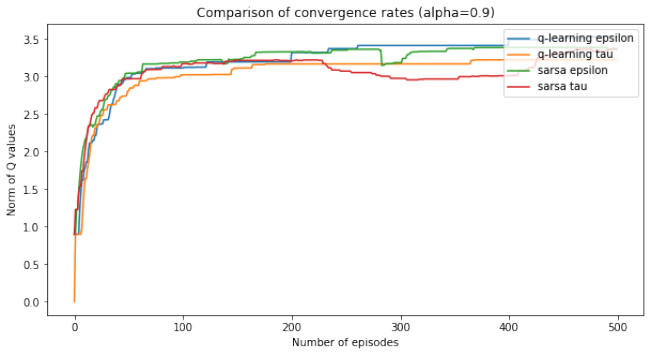
Pour l'algorithme QL, la convergence s'effectue plus rapidement et sur une valeur plus faible pour des petites valeurs de $epsilon$. Et contrairement à précedemment avec Softmax, les algorithmes semblent tendre approximativement vers les mêmes normes pour les mêmes valeurs du paramètres. La meilleure valeur pour epsilon semblerait être 0.001, ce-

pendant prendre une valeur trop faible risque de ne pas permettre suffisemment à l'agent d'ajuster sa politique si la première marche aléatoire n'est pas optimale.

Dans un dernier temps, nous allons observer les taux de convergence selon d'autres hypers paramètres. Nous allons pour ce faire fixer les valeurs d'$epsilon$ et de tau à 0.1 chacun, et étudier les taux de convergence pour différentes valeurs d'$alpha$ (learning rate ou facteur d'apprentissage) et de $gamma$ (discount factor ou facteur d'actualisation).

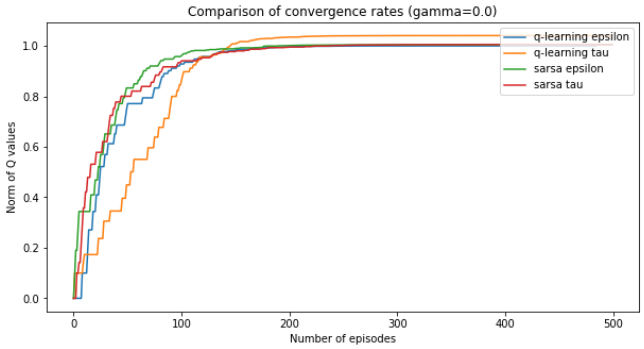
Nous allons tout d'abord faire varier les valeurs d'$alpha$ pour un $gamma$ fixé. Il s'agit du paramètre permettant de définir quelle importance est accordée à la nouvelle information par rapport à l'ancienne. Si $alpha = 0$, l'agent n'apprend rien et si $alpha = 1$, l'agent ne considère que la dernière information.

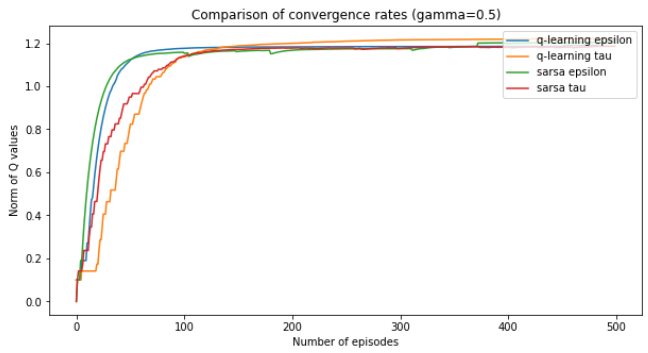


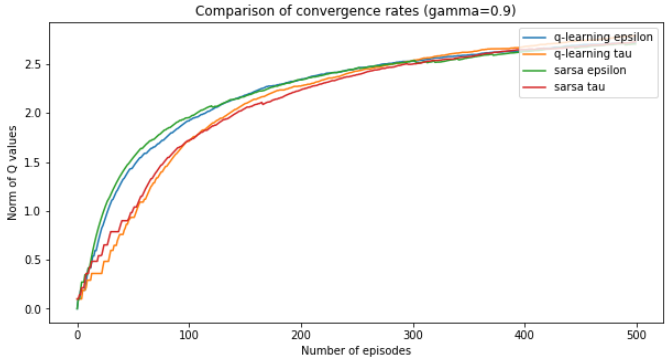


Nous remarquons que lorsque nous augmentons la valeur du paramètre $alpha$, le nombre d'épisodes nécessaires pour atteindre la convergence diminue, et ce pour toutes les méthodes. La convergence est cependant beaucoup plus saccadée pour une valeur proche de 1. Les quatres méthodes évoluent de manière assez uniforme, et tendent sensiblement pour n'importe quelle valeur de $alpha$ vers la même norme limite.

Nous allons maintenant faire varier les valeurs de $gamma$, cette fois pour une valeur $alpha$ fixe. Il s'agit du paramètre permettant de définir l'importance des récompenses futures.







Nous remarquons que la convergence nécessite moins d'épisodes lorsque $gamma$ est petit, et que la valeur limite de la norme est plus basse. Nous pouvons également observer que les algorithmes utilisant la stratégie Softmax semblent converger légèrement plus lentement qu'avec l'utilisation de Epsilon-greedy. Les quatres méthodes tendent cependant chaque fois vers la même valeur limite.