Implementação e Análise de Algoritmos de Aprendizado por Reforço Monte Carlo e Q-Learning

Gabriel Romio[1]

[1]Universidade do Vale do Rio dos Sinos (UNISINOS)
Av. Unisinos, 950 - Cristo Rei – 93022-750 – São Leopoldo – RS – Brazil

Resumo. *Este trabalho consiste na implementação e análise de algoritmos de Aprendizado por Reforço baseados nos métodos Monte Carlo e Q-Learning aplicados ao ambiente Gymnasium FrozenLake. Os algoritmos foram desenvolvidos e empregados em linguagem Python. Assim, avaliou-se o desempenho desses métodos ao serem aplicados à diferentes configurações de ambiente, bem como com variações nos parâmetros de aprendizado. Ambos os métodos foram capazes de encontrar políticas ótimas para o ambiente selecionado, porém, com base nos resultados obtidos, constatou-se a necessidade de número de interações significativamente maior no método Monte Carlo em relação ao Q-Learning para a solução de um mesmo problema. O método Monte Carlo também gerou resultados e políticas mais instáveis ao aumentar a complexidade do ambiente. Por fim, notou-se que os algoritmos apresentaram, no geral, melhores resultados em treinamentos com valores de Taxa de Aprendizado (α) e Fator de Desconto (γ) intermediários (0,5) ou altos (0,9).*

1. Introdução

O Aprendizado por Reforço (RL) é o segmento da Inteligência Artificial (IA) onde um agente aprende através de tentativa e erro por meio de interações com o ambiente no qual se encontra. Para isso, o agente recebe *feedbacks* numéricos na forma de recompensas ou penalidades, de modo que buscará sempre maximizar a recompensa acumulada ao longo do tempo. Em situações mais complexas, cada ação do agente não afeta apenas a recompensa imediata, mas pode afetar também a próxima situação bem como todas as subsequentes. [Sutton and Barto 2018]

Além do agente e do ambiente, alguns outros elementos também devem ser considerados em um sistema de Aprendizado por Reforço. Entre eles, pode-se citar a *política*, responsável por determinar como um agente se comporta em cada situação dado um determinado momento e ambiente, podendo ser representada através de uma função, uma tabela de estados, entre outros. Outro parâmetro importante é o *sinal de recompensa*, encarregado de definir o objetivo de um sistema. Assim, como a meta do agente é maximizar a recompensa total, este elemento indica o quão bom ou ruim pode ser uma determinada escolha realizada. Por outro lado, cada escolha pode ter consequências futuras. Assim, a *função de valor* é o elemento responsável por estimar possíveis recompensas a longo prazo de uma decisão realizada pelo agente, sempre avaliando e levando em consideração os estados que se seguirão e a recompensa disponível nesses estados. Por fim, o *modelo de ambiente* simula o comportamento de um local com o qual o agente interage, sendo utilizados para testar políticas e comportamentos, possibilitando que o agente descubra os melhores caminhos por meio de planejamento, tentativa e erro. [Sutton and Barto 2018]

Em aplicações que utilizam ambientes práticos, dificilmente tem-se conhecimento completo do ambiente. Para isso, aplicam-se métodos baseados em experiência, adquiridas através de sequências de interações reais ou simuladas com o meio, sem que seja necessário qualquer conhecimento prévio. Um dos métodos com essa proposta é o método Monte Carlo, que explora o ambiente partindo de um pressuposto de tarefas divididas em episódios finitos. Como resultado, é gerado um modelo com as médias de retornos estimados para cada par estado-ação a ser selecionado pelo agente. Uma característica importante do método Monte Carlo é que os valores da política é atualizada a cada episódio completo, e não a cada passo. Outros métodos com abordagem semelhante são os baseados em Diferença Temporal (TD), no qual se enquadra o método Q-Learning. A sua principal diferença em relação ao método Monte Carlo é que algoritmos TD atualizam a sua política a cada passo do agente, o que geralmente faz com que se encontre uma solução mais rapidamente. [Sutton and Barto 2018]

Dessa forma, este trabalho se propõe a implementar e avaliar os resultados obtidos por meio de dois métodos, Monte Carlo e Q-Learning. Os dois algoritmos foram aplicados com base no ambiente FrozenLake, disponível na biblioteca Gymnasium [Towers et al. 2023]. Devido à sua composição e à possibilidade de se realizar alterações em sua estrutura, esse ambiente possibilitou a análise e a avaliação do comportamento do agente em cada algoritmo. Durante os testes também buscou-se alterar os parâmetros de aprendizado utilizados por cada método, possibilitando uma avaliação completa do impacto de cada um deles bem como em eventuais diferenças no processo para cada alteração.

2. Métodos

Esse trabalho apresenta uma análise e implementação dos métodos Monte Carlo e Q-Learning para o ambiente FrozenLake, do Gymnasium. Para isso, são avaliados os conceitos e parâmetros envolvidos em cada um desses dois métodos, bem como em um estudo sobre políticas geradas para o ambiente FrozenLake, considerando variações em sua estrutura e valores de recompensas e penalidades para cada ação. Os próximos tópicos apresentam os métodos e definições empregados nessa análise.

2.1. Exploration e Exploitation

Um item a ser considerado em ambientes de Aprendizado por Reforço é o equilíbrio entre *exploration* e *exploitation*. A *exploitation* refere-se a ação do agente de buscar e dar preferência para ações com a maior recompensa estimada. Por outro lado, a *exploration* refere-se a exploração do ambiente pelo agente, buscando caminhos e rotas alternativas a fim de buscar políticas melhores do que as já avaliadas anteriormente. Dessa forma, um agente ideal deve estar balanceado entre selecionar os estados já conhecidos para obter recompensa e explorar rotas alternativas em busca da possibilidade de maiores recompensas futuras. Assim diferentes métodos podem ser implementados, como, por exemplo, a utilização de fatores que fazem com que o agente alterne aleatoriamente entre *exploration* e *exploitation* a cada passo. [Sutton and Barto 2018]

2.2. Algoritmo Monte Carlo

O algoritmo Monte Carlo é um método de aprendizado para estimar funções de valor e encontrar políticas apenas baseada na experiência adquirida pela exploração de um

determinado ambiente, sendo empregado em tarefas episódicas. Dessa forma, o modelo é construído a partir de seleção de estados baseados em probabilidade, sem a necessidade de computar todas as possibilidades ou ter uma mapa completo do ambiente. A cada passo do agente, o método Monte Carlo avalia a recompensa esperada de cada estado para determinar a próxima ação, podendo, de acordo com configurações de parâmetros de aprendizado, alterná-las entre *exploration* e *exploitation*. Uma sequência de passos, desde o estado inicial do agente até uma ocorrência de interrupção ou conclusão, constitui um episódio de aprendizagem. Assim, apenas ao térmico de cada episódio, a política do modelo é atualizada com os valores estimados a partir das recompensas obtidas ao longo do trajeto. [Sutton and Barto 2018]

A implementação do Monte Carlo pode ser realizada através dos métodos *on-policy* e *off-policy*. Na abordagem *on-policy* o agente se concentra apenas na política em que está seguindo, buscando atualizá-la e aprimorá-la ao longo do treinamento. Em métodos *off-policy* o agente aprimora a sua política de destino a partir de uma segunda política com foco em explorar o ambiente e coletar dados, o que geralmente apresenta melhores resultados. Neste trabalho os algoritmos foram implementados utilizado apenas a abordagem *off-policy*. [Sutton and Barto 2018]

Para otimizar o processo de aprendizado, ajustando-o para melhor interação do agente com o ambiente, alguns parâmetros podem ser empregados. Neste trabalho, os parâmetros utilizados no método Monte Carlo são a Taxa de Exploração (ε), fator a partir do qual define-se a probabilidade entre o agente escolher entre um estado que gera a maior estimativa de recompensa e um estado aleatório, visando explorar o ambiente. Isso significa que com ε=0,1, por exemplo, o método selecionará um estado que não gera a maior recompensa a cada 10% das ações. O Fator de Desconto (γ) define a importância das recompensas futuras em relação às imediatas. Valores próximos de 1 consideram recompensas futuras mais fortemente. Por fim, a Taxa de Decaimento pode ser utilizada para descontar as taxas de exploração e de aprendizado ao longo do tempo. É ajustado para se buscar sistemas que iniciem com valores de ε e γ altos que vão reduzindo a cada episódio. [Sutton and Barto 2018]

2.3. Algoritmo Q-Learning

O algoritmo Q-Learning é um método de aprendizado de RL baseado em Diferença Temporal (TD). Como no método Monte Carlo, métodos TD também aprendem apenas com a experiência da exploração do ambiente, porém a diferença é que algoritmos TD não tem a necessidade de esperar até o final do episódio para atualização da política atual. Dessa forma, a cada transição de estado realizada pelo agente, os valores da política são incrementados com base nas estimativas de recompensas atuais. Assim, métodos TD tendem a convergir para uma política ótima muito mais rapidamente do que o método Monte Carlo. [Sutton and Barto 2018]

O Q-Learning é uma implementação *off-policy* do aprendizado TD. Assim, o algoritmo atualiza uma política atual com base na exploração de diferentes ações. Para a implementação do algoritmo, além dos fatores empregados no método Monte Carlo, pode-se incluir também a Taxa de Aprendizado(α), a qual determina a magnitude das atualizações dos valores de uma função. Desse modo, um valor mais baixo tornará o aprendizado mais lento, mas mais estável, enquanto um valor mais alto pode levar a um

aprendizado mais rápido, mas menos estável. Também pode-se aplicar a α uma Taxa de Decaimento, fazendo com que o agente seja mais suscetível ao aprendizado no inicio do processo e menos suscetível no final. [Sutton and Barto 2018]

2.4. Gymnasium

O Gymnasium é uma biblioteca em Python *open source* desenvolvida e mantida pela OpenAI, tendo sido projetada para fornecer uma API com um conjunto de ambientes em comum para o desenvolvimento e análise de algoritmos de RL. Cada um dos ambientes possui funções relacionadas a diferentes ações de um agente, configurações da estrutura do modelo, além de *reset* e renderização desse ambiente, proporcionando uma visualização gráfica das interações. [Towers et al. 2023]

O ambiente selecionado para as análises apresentadas neste trabalho é o FrozenLake, representando um lago congelado. A Figura 1 apresenta a sua configuração padrão, onde o agente parte do estado [0,0] e o último bloco [3,3], que contém a representação de um presente, é o seu objetivo final. O agente deve se deslocar pelo meio evitando os buracos [1,1], [1,3], [2,3] e [3,0], que representam a ”morte”do agente e encerram o episódio. [Towers et al. 2023]

As ações possíveis pelo agente são os movimento para a esquerda, direita, cima e baixo e, nas configurações padrões do ambiente, a recompensa se dá apenas ao chegar no estado [3,3], sendo 1, de modo que qualquer outro movimento possui recompensa 0. Ao cair em um buraco não há nenhuma penalidade, apenas encerra-se o ensaio imediatamente sem a possibilidade de obter uma recompensa. [Towers et al. 2023]

Além disso, podem-se realizar alterações no ambiente para melhor adequá-lo a cada ensaio. Assim, é possível alterar o seu tamanho, aumentando ou diminuindo a quantidade de estados possíveis, bem como a sua estrutura, acrescentando ou diminuindo a quantidade de buracos, as suas localizações e a posição do presente. Também podem-se alterar as recompensas distribuídas ao agente em cada estado.

Figura 1. Ambiente FrozenLake [Towers et al. 2023]

2.5. Implementação e análise dos algoritmos

Os algoritmos referentes aos métodos Monte Carlo e Q-Learning foram implementados através de funções em Python. Durante o treinamento, o agente explora o ambiente FrozenLake em busca de recompensas de acordo com o ajuste dos parâmetros de aprendizado. Uma segunda função faz com que o agente percorra o ambiente ”gananciosoamente”, ou seja, sempre escolhendo a ação que resulta na estimativa de maior

recompensa. Dessa forma, a partir dessa atuação gananciosa do agente, busca-se avaliar o desempenho de um determinado treinamento.

Para avaliar e comparar a implementação de diferentes métodos e configurações de aprendizagem, foram definidas algumas variações para o ambiente FrozenLake. No Ensaio 1 foi utilizado o modelo padrão em um espaço de estados 4x4, como apresentado na Figura 1. O Ensaio 2 foi realizado utilizando a configuração padrão 8x8 do ambiente, conforme Figura 2. Para o Ensaio 3 empregou-se um ambiente 16x16. Nesse ensaio, conforme apresentado na Figura 3, a composição do espaço foi disposta manualmente de modo a dificultar a exploração dos trajetos para o agente atingir o objetivo final. Dessa forma, pode-se avaliar com maior aprofundamento cada um dos métodos e políticas.

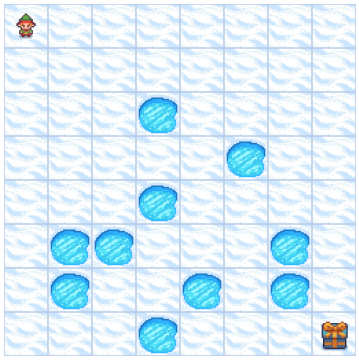

Figura 2. Ambiente FrozenLake 8x8 [Towers et al. 2023]

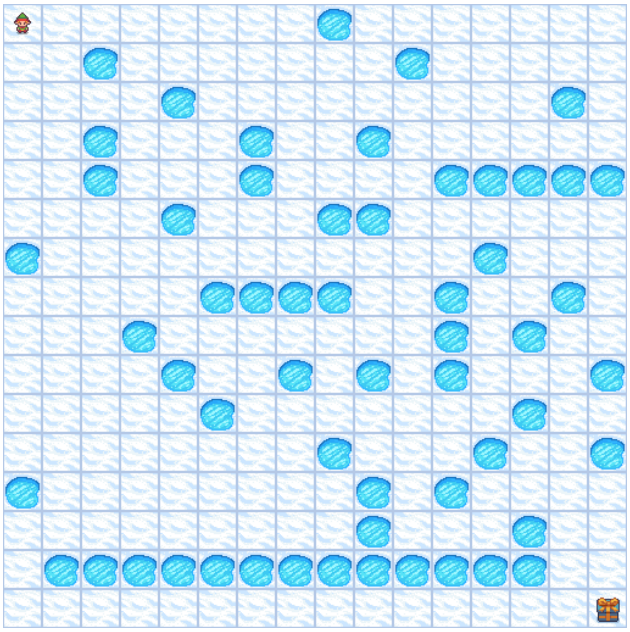

Figura 3. Ambiente FrozenLake 16x16 personalizado

Dentro desses espaços de ensaios também foram avaliados os comportamentos dos agentes para diferentes configurações dos parâmetros de aprendizagem e diferentes valores de recompensa para cada estado. Assim, buscou-se realizar um estudo comparativo do impacto desses parâmetros tanto no tempo de aprendizado, como na avaliação

das políticas resultantes de cada treinamento. As recompensas e penalidades, bem como a quantidade de interações de cada treinamento, foram ajustadas de acordo com o número de estados de cada ambiente.

3. Resultados e discussões

Nessa seção são apresentados os resultados obtidos experimentalmente a partir dos métodos descritos.

3.1. Ensaio 1 - FrozenLake 4x4

O primeiro ensaio utilizou o ambiente FrozenLake 4x4 padrão, conforme Figura 1. Para esse ensaio, como o ambiente possui um número reduzido de estados, definiu-se valores baixos para penalidades e recompensas. Caso o agente caia em um buraco, será penalizado em -10, e a recompensa ao encontrar o objetivo será 10. Para fazer com que o agente sempre busque o objetivo no menor caminho possível, foi acrescentada uma penalidade de -1 para cada passo do agente. Desse modo, a recompensa máxima possível, considerando o menor caminho para buscar a recompensa, será 5. A Taxa de Exploração (ε) inicial foi mantida em 1,0, com uma taxa de decaimento de 0,01, para fazer com que o agente explore ao máximo o ambiente no inicio do treinamento e reduza linearmente esse fator ao longo das primeiras 100 interações do treinamento. Assim, avaliou-se o desempenho do agente para diferentes configurações de Fator de Desconto (γ), no método Monte Carlo, durante 300 interações, cada uma com 20 passos do agente. Os valores de γ foram definidos iniciando em 0,1 (baixo), 0,5 (intermediário) e 0,9 (alto), estando apresentados na Figura 4. Para o método Q-Learning, considerou-se os parâmetros Taxa de Aprendizado(α) e Fator de Desconto (γ), onde foram testados diferentes combinações para os valores de 0,1, 0,5 e 0,9 em cada um deles. Devido a capacidade desse método encontrar uma política ótima com número menor de interações do que no Monte Carlo, 100 interações foram o suficiente para se avaliar os resultados, também com 20 passos cada, gerando os resultados demonstrados na Figura 5.

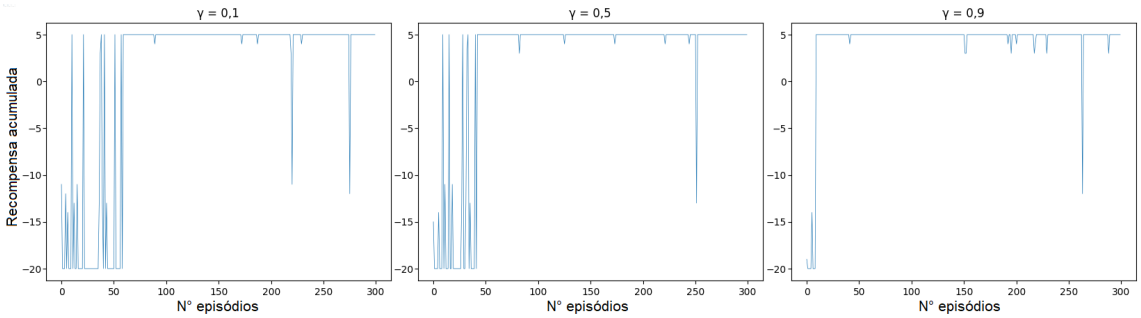

Figura 4. Resultados Monte Carlo FrozenLake 4x4

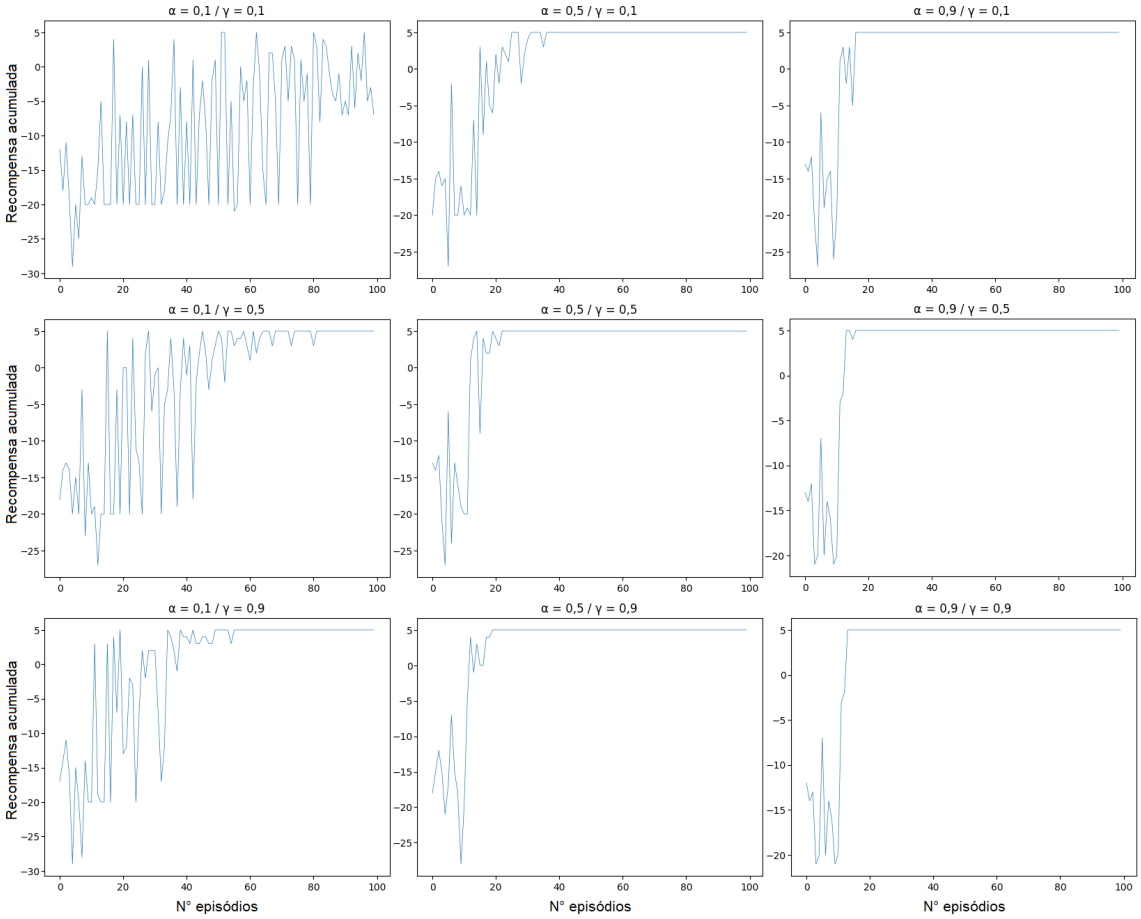

Figura 5. Resultados Q-Learning FrozenLake 4x4

Como o ambiente possui número baixo de estados e políticas possíveis, os dois algoritmos encontraram políticas ótimas após um número significativamente baixo de interações. A exceção foi para valores muito baixos de α e γ no método de Q-Learning, que resultaram em um aprendizado muito mais lento do agente. Assim, ambos os métodos foram capazes de encontrar políticas com a recompensa máxima (5), porém o método Monte Carlo necessitou de número maior de interações. As Figuras 6 e 7 demonstram um exemplo desses trajetos percorridos pelo agente em cada algoritmo.

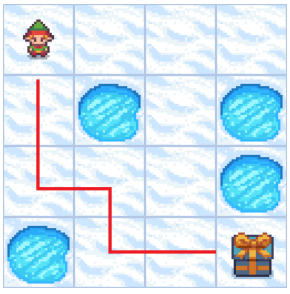

Figura 6. Política Monte Carlo FrozenLake 4x4

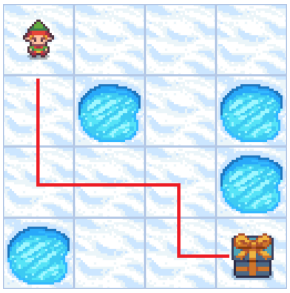

Figura 7. Política Q-Learning FrozenLake 4x4

3.2. Ensaio 2 - FrozenLake 8x8

O segundo ensaio foi realizado no ambiente FrozenLake 8x8 padrão, apresentado na Figura 2. Nesse ensaio definiu-se uma penalidade de -20 se o agente cair em um buraco e uma recompensa de 20 ao encontrar o objetivo. Como no ensaio anterior, utilizou-se uma penalidade de -1 para cada passo do agente. Dessa forma, a recompensa máxima possível será 7. A Taxa de Exploração (ε) inicia em 1,0, com uma taxa de decaimento de 0,0025, encerrando após 400 interações. Como o ambiente é consideravelmente maior do que o do Ensaio 1, no método Monte Carlo avaliou-se o desempenho do agente ao longo de 1.000 interações, cada uma com 50 passos. Assim, os resultados obtidos nessas configurações para os valores de Fator de Desconto (γ) 0,1, 0,5 e 0,9, estão expressos na Figura 8. No método Q-Learning o agente foi treinado durante 400 interações, também com 50 passos cada, e os resultados para diferentes combinações dos valores de Taxa de Aprendizado(α) e Fator de Desconto (γ) estão apresentados na Figura 9.

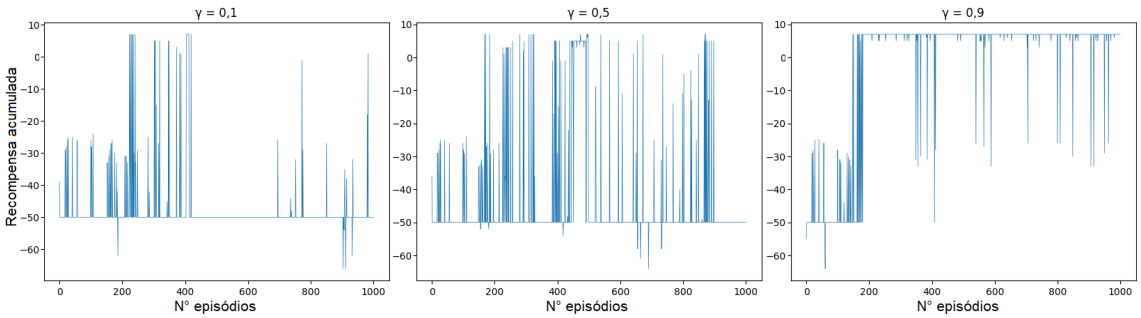

Figura 8. Resultados Monte Carlo FrozenLake 8x8

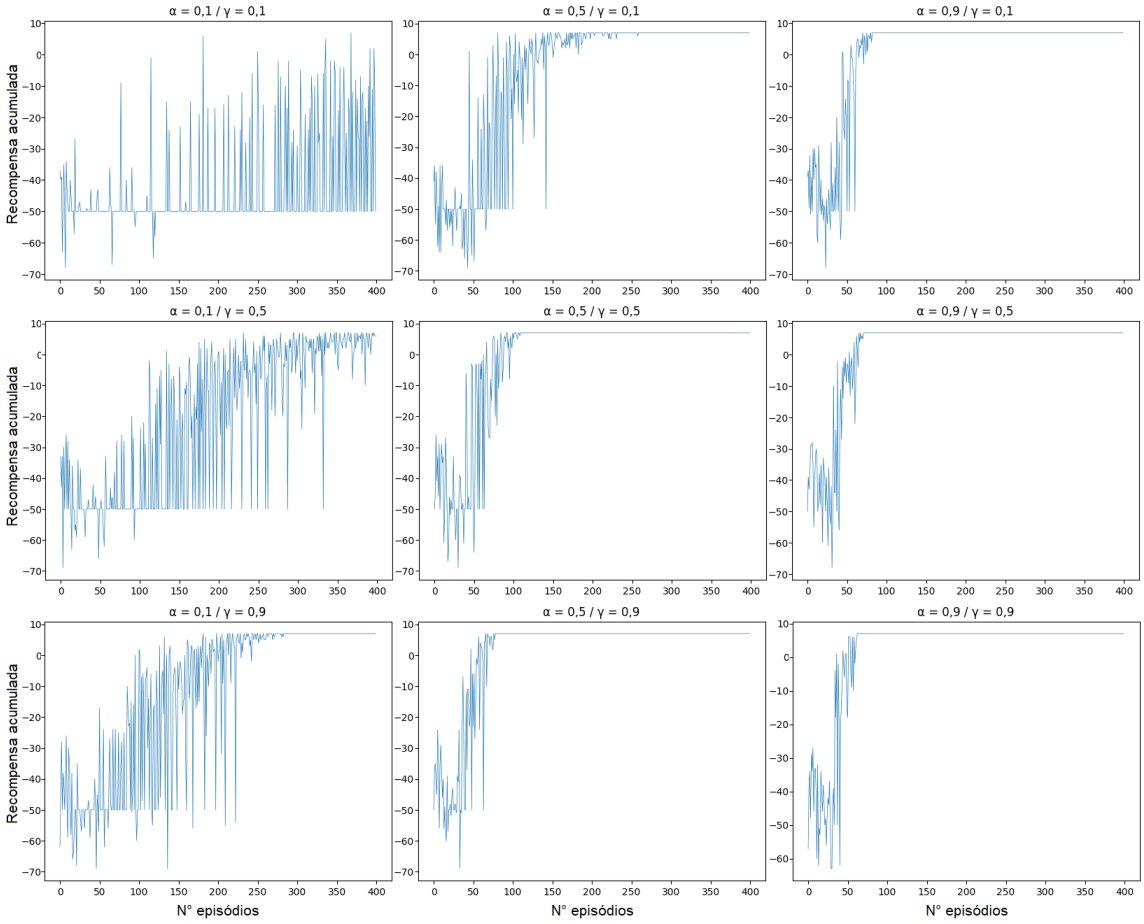

Figura 9. Resultados Q-Learning FrozenLake 8x8

Nesse ensaio, ao empregar o método Monte Carlo, o agente apenas encontrou políticas ótimas para valores de γ=0,9. Além disso, o número de interações aumentou significativamente em relação ao primeiro ensaio e no método Q-Learning foi necessário aumentar o valor de α para se obter melhores resultados. Assim como no ensaio anterior, o método Monte Carlo utilizou um número consideravelmente maior de interações do que Q-Learning para encontrar uma política ideal. Dessa forma, no ambiente 8x8, ambos os métodos foram capazes de encontrar soluções ótimas, conforme trajetos apresentados nas Figuras 10 e 11.

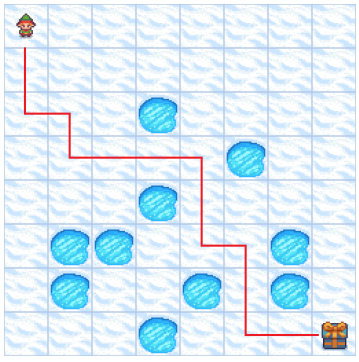

Figura 10. Política Monte Carlo FrozenLake 8x8

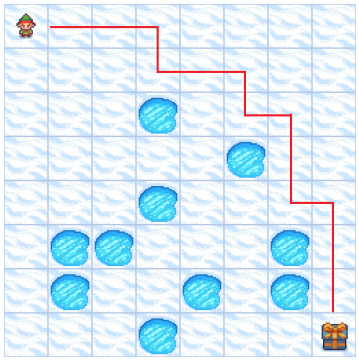

Figura 11. Política Q-Learning FrozenLake 8x8

3.3. Ensaio 3 - FrozenLake 16x16

O terceiro ensaio consistiu num ambiente FrozenLake 16x16, com uma disposição personalizada dos buracos para dificultar a trajetória do agente, conforme apresentado anteriormente na Figura 3. Devido ao seu tamanho, e para compensar as penalidades de -1 de cada passo, definiu-se uma recompensa de 50 caso o agente atinja o seu objetivo e penalidade de -50 caso caia em um buraco, de modo que a máxima recompensa possível seja 19. Foi definido um valor inicial de 1,0 para a Taxa de Exploração (ε) com uma taxa de decaimento de 0,001, encerrando após 1000 interações. Com uma complexidade muito maior do que nos outros ensaio, foi necessário treinar o algoritmo Monte Carlo ao longo de 30.000 episódios com 100 passos cada para encontrar uma política ótima e melhor poder visualizar os resultados. Assim como nos outros ensaios, realizaram-se treinamentos para Fatores de Desconto (γ) 0,1, 0,5 e 0,9, com resultados demonstrados na Figura 12. No método Q-Learning pode-se avaliar o agente em 1.000 episódios com 100 passos cada. Os resultados para diferentes valores de Taxa de Aprendizado(α) e Fator de Desconto (γ) estão apresentados na Figura 13.

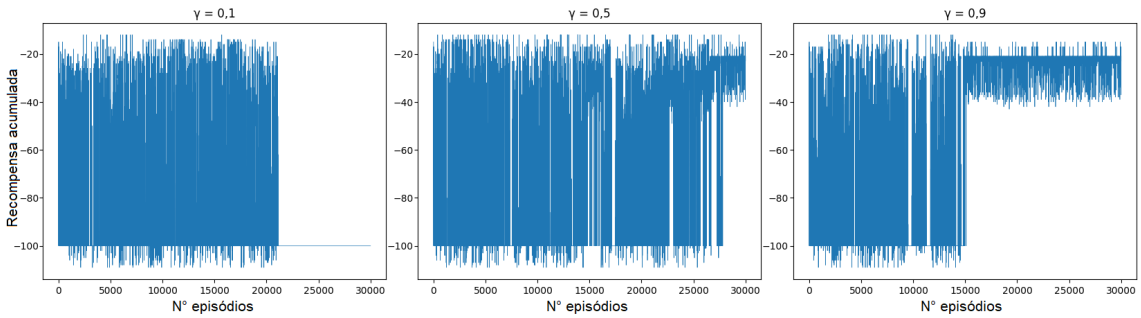

Figura 12. Resultados Monte Carlo FrozenLake 16x16

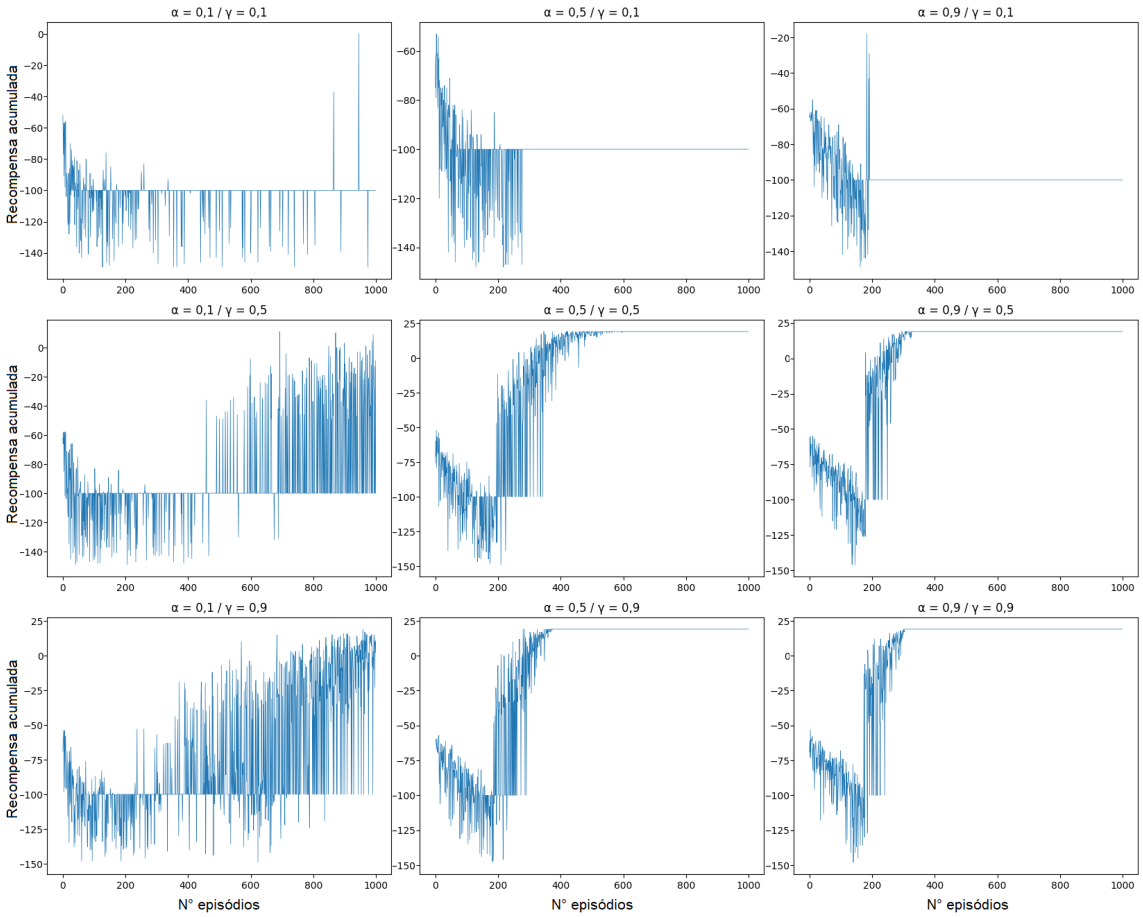

Figura 13. Resultados Q-Learning FrozenLake 16x16

Nesse ensaio o método Monte Carlo apresentou resultados pouco estáveis, além de grande número de interações para encontrar uma política ótima, mesmo para γ=0,9. No método Q-Learning apenas situações onde α e γ foram maior ou igual a 0,5 foram capazes de obter a recompensa máxima. Devido à Taxa de Exploração inicial, as primeiras interações apresentaram resultados instáveis. Dessa forma, apesar de ser necessário treinos significativamente mais longos, bem como maior necessidade de ajuste de parâmetros, ambos os métodos foram capazes de encontrar políticas satisfatórias com máxima recompensa para o ambiente 16x16. Esses trajetos são demonstrados nas Figuras 14 e 15.

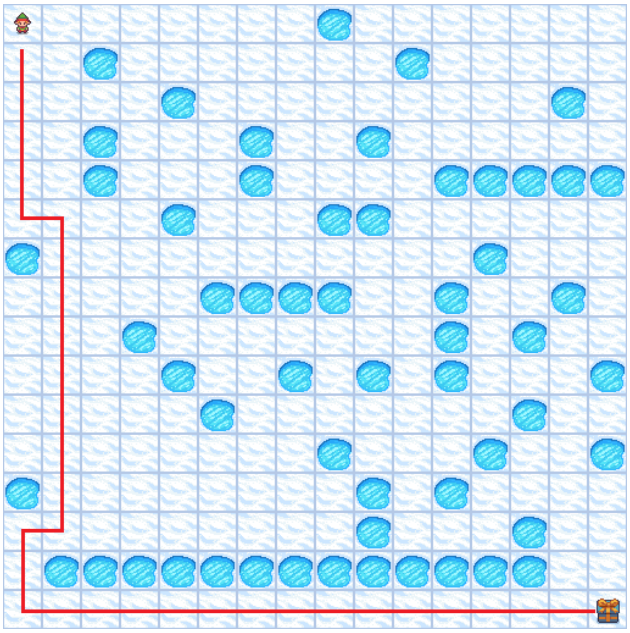

Figura 14. Política Monte Carlo FrozenLake 16x16

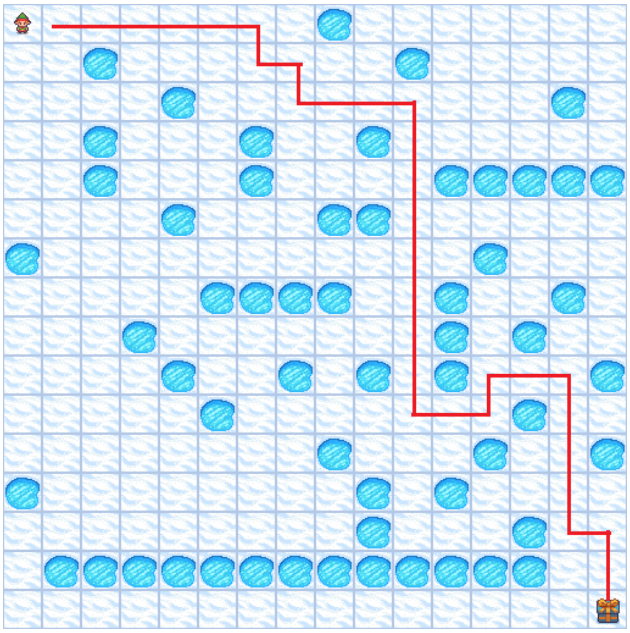

Figura 15. Política Q-Learning FrozenLake 16x16

4. Conclusão

Com os resultados apresentados, percebe-se que o número de interações necessários para se obter uma política ótima é muito maior no método Monte Carlo do que no método Q-Learning. Isso ocorre principalmente em problemas mais complexos como, por exemplo, no ambiente de tamanho 16x16, o que demonstra a importância da atualização da política a cada passo, característica de métodos TD como Q-Learning. Apesar disso, a implementação do Monte Carlo mostrou-se mais simples, principalmente

decorrente do menor número de parâmetros envolvidos, o que facilita a sua configuração. Assim, conclui-se que em problemas com número reduzido de estados o método Monte Carlo é capaz de encontrar soluções ótimas rapidamente, porém, ao aumentar a complexidade do ambiente, o Q-Learning apresenta resultados significativamente melhores.

Referências

Sutton, R. S. and Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). The MIT Press.

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Shen, A. T. J., and Younis, O. G. (2023). Gymnasium. Zenodo.