10.18605/2175-7275/cereus.v16n3p182-196

ARTIGO ORIGINAL

<< Recebido em: 15/06/2024 Aceito em: 18/08/2024. >>

REVISTA CEREUS ISSN: 2175-7275

Controle de um Robô Móvel com Pêndulo Invertido por Meio de Aprendizado por Reforço

Control of a Mobile Inverted Pendulum Robot Using Reinforcement Learning

Miguel Lopes de Moraes[1], Isaac Jesus da Silva[2], Danilo Hernani Perico[3]

RESUMO

Este trabalho apresenta o conceito de Aprendizado por Reforço aplicado ao problema do pêndulo invertido preso a um robô móvel (problema conhecido como *Cart Pole* em inglês). Nesse problema, o robô deve aprender as melhores ações para manter o pêndulo em equilíbrio. Os algoritmos de Aprendizado por Reforço utilizados nesse projeto foram *o Q-Learning*, *Q-Learning* com heurística (HAQL) e o *Deep Q Network* (DQN). Os resultados experimentais demonstraram que os três algoritmos analisados puderam resolver com sucesso o problema de controle proposto, contudo o HAQL e o DQN alcançaram resultados mais promissores.

Palavras-chave: Aprendizado por Reforço. Controle de Robôs Móveis. Pêndulo Invertido.

ABSTRACT

This paper introduces the concept of Reinforcement Learning applied to the problem of an inverted pendulum attached to a mobile robot (also known as CartPole). In this problem, the robot must learn the best actions to maintain the pendulum in balance. The Reinforcement Learning algorithms used in this project were Q-Learning, Q-Learning with heuristics (HAQL), and Deep Q Network (DQN). The experimental results demonstrate that the application of all algorithms successfully addressed the proposed control problem, with HAQL and DQN yielding the best outcomes.

Keywords: Reinforcement Learning. Mobile Robots Control. Inverted Pendulum.

[1] Engenheiro de Robôs, Centro Universitário FEI, São Bernardo do Campo, São Paulo, Brasil. ORCID: https://orcid.org/0009-0006-0054-7788
E-mail: miguellopes54954@gmail.com

[2] Doutor em Engenharia Elétrica. Centro Universitário FEI, São Bernardo do Campo, São Paulo, Brasil.
ORCID: https://orcid.org/0000-0002-5662-6593

[3] Doutor em Engenharia Elétrica. Centro Universitário FEI, São Bernardo do Campo, São Paulo, Brasil.
ORCID: https://orcid.org/0000-0001-6887-9240
E-mail: dperico@fei.edu.br

1. INTRODUÇÃO

A Inteligência Artificial (IA) é um campo de estudo que busca desenvolver sistemas capazes de simular a inteligência humana. Desde os primórdios da humanidade, os seres humanos buscam criar artefatos que possam imitar a inteligência humana. Com o avanço da tecnologia, a IA se tornou uma área de pesquisa cada vez mais importante e promissora.

Considerando a área de Aprendizado de Máquina, pode-se citar, por exemplo, o Aprendizado por Reforço (*Reinforcement Learning* – RL) como uma das técnicas mais usuais. O RL é uma técnica de aprendizado bastante usual, pois um agente pode aprender por meio de interação direta com o ambiente e seu algoritmo sempre converge para uma situação de equilíbrio (SUTTON, 1988). No RL, um agente pode aprender em um ambiente não conhecido previamente, por intermédio de experimentações. Dependendo de sua atuação, o agente recebe uma recompensa ou uma penalização e, desta forma, o algoritmo encontra um conjunto de ações que levam o agente a percorrer o caminho ótimo. A este conjunto, formado pelas melhores ações, dá se o nome de política ótima (π^*).

O *Q-Learning* é um algoritmo de RL que permite a um agente aprender a tomar decisões em um ambiente desconhecido, maximizando a recompensa obtida ao longo do tempo. O algoritmo é baseado na ideia de que o agente pode aprender uma função-valor chamada *Q*-valor, que estima a recompensa que o agente pode esperar ao tomar uma ação em um determinado estado do ambiente (WATKINS, 1989).

No *Q-Learning*, heurísticas podem ser utilizadas para melhorar o desempenho do agente em situações em que a quantidade de estados e ações possíveis é grande demais para ser tratada de forma exaustiva. Heurística é um termo utilizado para se referir a técnicas de resolução de problemas que não são baseadas em cálculos exatos ou algoritmos precisos, mas sim em experiência, julgamento e intuição. Assim, o *algoritmo Q-Learning* Acelerado por Heurísticas (*Heuristically Accelerated Q-Learning* - HAQL) utiliza alguma informação conhecida a priori para acelerar o aprendizado de um agente (BIANCHI, 2004).

Por sua vez, o algoritmo *Deep Q Network* (DQN) utiliza uma rede neural artificial como base para extrair mais conhecimento do ambiente, permitindo que o agente aprenda a tomar decisões em ambientes mais complexos e com maior dimensionalidade de estados (SEWAK, 2019). O DQN é um método de Aprendizado por Reforço Profundo (*Deep Reinforcement Learning* - DRL) (MNIH et al., 2013). Diferentemente dos métodos clássicos de *Q-Learning*, em que a função-valor é representada por uma tabela, o DRL usa redes neurais profundas para aproximar a função-valor. Isso permite que o agente lide com problemas

mais complexos, onde a dimensionalidade do espaço de estados pode ser muito grande para ser representada por uma tabela.

Assim, o objetivo desse projeto é o estudo, a implementação e a comparação de algoritmos de Aprendizado por Reforço para a realização do controle de posição de um pêndulo invertido posicionado em cima de um robô móvel com rodas; dessa forma, nesse trabalho o Aprendizado por Reforço foi explorado no domínio do controle autônomo de robôs móveis. Esse domínio é constantemente explorado para estudo e avaliação de técnicas de controle, como controlador Nebuloso, por exemplo (LIMA, A.L., SOUZA, P.H.M., MOURA Jr, J.R.V, 2022). Os algoritmos de RL estudados foram o clássico *Q-Learning* (WATKINS, 1992), o *Q-Learning* Acelerado por Heurísticas (*Heuristically Accelerated Q-Learning* – HAQL) (BIANCHI, 2004) e o *Deep Q Network* (DQN) (SEWAK, 2019).

O domínio aplicado a essa proposta é um ambiente criado no simulador Webots (WEBOTS, 2022) em que o pêndulo é posicionado em cima de um robô móvel com tração nas quatro rodas.

2. APRENDIZADO POR REFORÇO

Essa seção apresenta os fundamentos teóricos dos algoritmos de Aprendizado por Reforço estudados nesse trabalho: *Q-Learning* (WATKINS, 1992), *Q-Learning* Acelerado por Heurísticas (*Heuristically Accelerated Q-Learning* – HAQL) (BIANCHI, 2004) e *Deep Q Network* (DQN) (SEWAK, 2019).

O *Q-Learning* é um algoritmo de Aprendizado por Reforço que tem como objetivo permitir que um agente aprenda a tomar decisões em um ambiente dinâmico, sem a necessidade de ter um conhecimento prévio do ambiente. Ele foi proposto por Watkins (1989) e utiliza uma tabela de valores Q para armazenar a utilidade de cada estado s para cada ação a. Com o RL, os agentes podem aprender por meio da interação direta com o ambiente, ajustando suas ações com base nas recompensas e penalizações recebidas. Esse processo permite que os algoritmos encontrem uma política ótima.

É possível descrever o processo de atuação do *Q–Learning* como um agente que busca maximizar uma função valor-ação (Q) por meio de recompensas recebidas conforme ações e estados escolhidos, tudo em decorrência do instante de tempo t em que foi executado. Assim, a função Q pode ser descrita pela Equação 1, que tem seu funcionamento baseado na soma do reforço recebido pelo agente realizando a ação (a_t), no estado (s_t), em um momento t.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r(s_t, a_t) + \gamma maxQ(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (1)$$

Onde:

- s_t é o estado atual
- a_t é a ação realizada em s_t
- $r(s_t, a_t)$ é o reforço recebido após realizar a_t em s_t
- s_{t+1} é o próximo estado
- a_{t+1} é a ação realizada em s_{t+1}
- γ é o fator desconto, admitindo um intervalo de $0 \leq \gamma \leq 1$
- α é a taxa de aprendizado, admitindo um intervalo de $0 \leq \alpha \leq 1$

É interessante notar uma propriedade desse algoritmo: as ações para cada instante de tempo podem ser escolhidas por meio de exploração aleatória ou seguindo uma estratégia gulosa (ε-*Greedy*). É possível descrever essa função como uma forma do agente executar a ação que tenha o maior valor de Q com uma probabilidade $1 - \varepsilon$. Assim, o código pode tirar proveito das ações anteriores, atuando com uma política ótima π. Dessa forma, o modelo de transição de estados é descrito na Equação 2.

$$\pi(st) = \begin{cases} a_{random} & se\ q < \varepsilon \\ arg\ max\ a_t Q_t(st, a_t) & caso\ contrário \end{cases} \quad (2)$$

Onde:

- q é um valor aleatório com distribuição uniforme entre $[0, 1]$
- ε é o parâmetro que define a taxa de exploração/explotação $(0 \leq \varepsilon \leq 1)$: quanto menor o valor de ε, menor a probabilidade da escolha aleatória
- a_{random} é a ação aleatória selecionada dentre as ações possíveis em s_t

É possível descrever o código do *Q–Learning* conforme Algoritmo 1. No geral, ele percorrerá toda a tabela Q e explorará as possibilidades em busca das melhores recompensas, atuando com devidas ações em determinados estados. Então, o mesmo processo será repetido até que algum critério de parada seja atendido, atualizando os valores dos estados e ações na Tabela Q (WATKINS, 1989).

Algoritmo 1: Pseudocódigo *Q–Learning*

início
  Inicialize $Q(s,a)$ arbitrariamente
  repita
    visite o estado s
    Selecione uma ação a de acordo com a regra de transição
    Execute a ação a
    Receba o reforço $r(s,a)$ e observe o próximo estado s'
    Atualize os valores $Q(s,a)$ de acordo com a regra:
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a) + \gamma max_{a'} Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$
  até *algum critério de parada seja atingido*;
fim

O *Q-Learning* Acelerado por Heurísticas (*Heuristically Accelerated Q-Learning* – HAQL) é um algoritmo de aprendizado por reforço que usa uma função heurística para acelerar a convergência do algoritmo *Q-Learning* tradicional. O algoritmo HAQL é usado para encontrar a política ótima para um agente que está tentando tomar decisões em um ambiente dinâmico e incerto (BIANCHI, 2004).

O algoritmo HAQL usa uma função heurística para guiar o processo de aprendizagem e reduzir o número de iterações necessárias para convergir para a política ótima. A função heurística é projetada para fornecer informações adicionais ao algoritmo *Q-Learning*, de forma que o agente possa aprender mais rapidamente (PERICO, 2013).

O HAQL é uma extensão do *Q-Learning*, um algoritmo de aprendizado por reforço que usa uma tabela de valores para armazenar as estimativas de valor de cada ação em cada estado. No entanto, o *Q-Learning* pode ser lento para convergir em ambientes complexos com muitos estados e ações. Assim o HAQL supera essa limitação, fornecendo uma maneira mais rápida de encontrar a política ótima, reduzindo o número de iterações necessárias para convergir. A função heurística usada no HAQL pode ser projetada de várias maneiras, dependendo do ambiente específico em que o agente está operando (BIANCHI, 2004).

A implementação do HAQL em código possui algumas semelhanças com o pseudocódigo do *Q-Learning* (Algoritmo 1), mas há diferenças importantes no processo de escolha das ações pelo agente. Uma das principais diferenças é que o HAQL usa a função heurística para escolher a ação que o agente deve tomar em cada estado. A função heurística fornece informações adicionais sobre o ambiente, permitindo que o agente tome decisões mais informadas e eficazes. O Algoritmo 2 apresenta o pseudocódigo desse método.

Algoritmo 2: Pseudocódigo do HAQL

início
  Inicialize $Q(s,a)$ arbitrariamente
  repita
    visite o estado s
    Selecione uma ação a de acordo com a regra de transição de estado do HAQL
    Execute a ação a
    Receba o reforço $r(s,a)$ e observe o próximo estado s'
    Atualize os valores $Q(s,a)$ de acordo com a regra:
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a) + \gamma max_{a'} Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$
  até *algum critério de parada seja atingido*;
fim

A transição de estados do HAQL pode ser representada conforme Equação 3.

$$\pi\,(st) = \begin{cases} a_{random} & se\ q < \varepsilon \\ arg\ max\ a_t[Q_t(st,a_t) + H_t(st,a_t)] & caso\ contrário \end{cases} \quad (3)$$

Por sua vez, o *Deep Q Network* (DQN) combina o aprendizado por reforço *Q-Learning* com as Redes Neurais Convolucionais, sendo o primeiro algoritmo de Aprendizado por Reforço Profundo (DRL). Mnih et al. (2013) desenvolveu a DQN e implementou para os jogos de Atari, utilizando os quadros das imagens do jogo como entrada da rede e a saída da rede corresponde às diversas ações disponíveis no jogo. A arquitetura da rede possui três camadas de redes neurais convolucionais seguidas por duas camadas de neurônios totalmente conectadas, sem o uso de camadas de *pooling*.

DQN adotou algumas técnicas de estabilização do processo de aprendizado, como o *experience replay* e o *target network*. Durante o jogo as experiências $e_t = <s_t, a_t, r_t, s_{t+1}>$ são armazenadas em uma variável chamada *replay memory* $\mathcal{D}$ (Yu et al. 2021). O uso do *experience replay* armazenado na *replay memory* $\mathcal{D}$ permite que os agentes lembrem e reutilizem as experiências do passado. Na DQN, a *experience replay* pode reduzir a quantidade de experiência necessária para o agente aprender.

Como é possível ver no Algoritmo 3, a variável x é o quadro da imagem do jogo. Já o estado s_t é considerando como uma sequência de ações e observações (onde nesse domínio cada ação e observação é um passo) $s_t = x_1, a_1, x_2, a_2, x_3, a_3, \ldots, x_t$, considerando t como um tempo finito dessas sequências. N_r é a capacidade máxima de armazenamento de experiências na *replay memory* ($|\mathcal{D}| = N_r$). N_b é a quantidade de experiências que será

utilizado da *replay memory* ($|\mathcal{E}| = N_b$) sendo que N_b não pode ser maior *que* N_r $(1 \leq N_b \leq N_r)$. $\mathcal{E}$ é um conjunto de experiências ($\mathcal{E} = \{e_1, e_2, \ldots, e_{Nb}\}$) extraído de forma aleatória da memória $\mathcal{D}$ ($\mathcal{E} \sim \mathrm{U}(\mathcal{D})$).

Algoritmo 3: Algoritmo Deep Q-Network.

Inicialize a *replay memory* $\mathcal{D}$ vazia mas com capacidade N_r
Inicialize os parâmetros θ da rede
Inicialize a função valor Q com valores aleatórios
enquanto *o número de episódios desejado não for alcançado* **faça**
  Observe o estado s_t do quadro x, $(s_t \leftarrow x)$
  enquanto *O episódio não for finalizado* **faça**
    Selecione a ação a_t do estado s_t utilizando a seleção $\epsilon - greedy$
    Execute a ação a_t no emulador e observe a recompensa r_t e a imagem x
    Observe o estado s_{t+1} do quadro x, $(s_{t+1} \leftarrow x)$
    Armazena a experiência $e_t =< s_t, a_t, r_t, s_{t+1} >$ na memória $\mathcal{D}$
    Selecione de forma aleatória um conjunto de experiências $\mathcal{E}$ da memória $\mathcal{D}$ possuindo uma quantidade N_b de experiências $(\mathcal{E} \sim U(\mathcal{D}))$
    para *cada experiência* $e_j \in \mathcal{E}$ **faça**
      se *episódio termina no passo* $j+1$ **então**
        $y_j \leftarrow r_j$
      fim
      senão
        $y_j \leftarrow r_j + \gamma \max_a \hat{Q}(s_{j+1}, a; \theta^-)$
      fim
    fim
    Treine a rede realizando a descida de gradiente em $(y_j - Q(s_j, a_j; \theta))^2$ para os parâmetros θ da rede
    $\theta^- \leftarrow \theta$
    $s_t \leftarrow s_{t+1}$
    A cada C passos faça: $\hat{Q} = Q$
  fim
fim

Para melhorar a estabilidade da DQN foi usado um Q diferente para atualizar o y, então a cada C passos realiza-se uma cópia dos valores Q para um $\hat{Q}$. Usando esse valor $\hat{Q}$ o algoritmo torna-se mais estável, portanto, as divergências ou oscilações da política torna-se menos comum (MNIH et al., 2015).

3. MATERIAIS E MÉTODOS

Sendo o objetivo desse trabalho o estudo, a implementação e a comparação de algoritmos de Aprendizado por Reforço para realizar o controle de posição de um pêndulo

invertido posicionado em cima de um robô móvel com rodas, foi desenvolvido uma pesquisa quantitativa de natureza aplicada, buscando dados objetivos que permitam o aprofundamento do entendimento sobre como os algoritmos de Aprendizado por Reforço podem ser aplicados de forma eficaz em cenários complexos de controle robótico. A pesquisa visa não apenas implementar esses algoritmos, mas também compará-los e avaliar seu desempenho em uma configuração prática e desafiadora.

Todos os experimentos foram realizados utilizando o simulador Webots, que permite a criação e a simulação de ambientes robóticos virtuais. O simulador possui uma interface gráfica que permite a criação e a programação de robôs e cenários, tornando-se uma ferramenta poderosa para a experimentação em robótica. O simulador contém, por padrão, uma variedade de mapas, sensores e robôs disponíveis e ele pode ser programado em variadas linguagens de programações, dentre elas, a que foi usada nesse projeto, o Python. A versão do Webots utilizada nesse trabalho foi a 2022a.

Além do simulador, foram utilizadas bibliotecas de aprendizagem de máquina, como o TensorFlow (ABADI et al. 2016) e o Keras, que permitem a implementação de algoritmos de aprendizado profundo, como o DQN. Também foi utilizada a biblioteca Gym do OpenAI (BROCKMAN et al. 2016) para implementação dos algoritmos de aprendizado por reforço. A ideia principal das ferramentas da OpenAI é possibilitar o desenvolvimento da IA de maneira mais amigável.

O ambiente de simulação utilizado foi um cenário padrão do Webots, contendo o robô Pioneer 3AT com a inclusão de um pêndulo invertido em sua parte superior. O Pionner 3AT tem tração nas quatro rodas e vários sensores ao redor de seu chassi. Além disso, o robô utilizado conta com um sensor para medir a angulação do pêndulo invertido. A Figura 1 apresenta o robô no simulador conforme utilizado nesse trabalho.

Para realizar a comparação dos métodos mencionados, foram desenvolvidos códigos utilizando os conceitos correspondentes. O código com o *Q-Learning* teve como base o OpenAI para a captura das variáveis e a formulação do *Q-Learning* para o treinamento do agente; o mesmo procedimento foi utilizado para o HAQL. Já o DQN utilizou do *TensorFlow* para a construção de uma Rede Neural e da mesma forma do *Q-Learning* para a obtenção das variáveis.

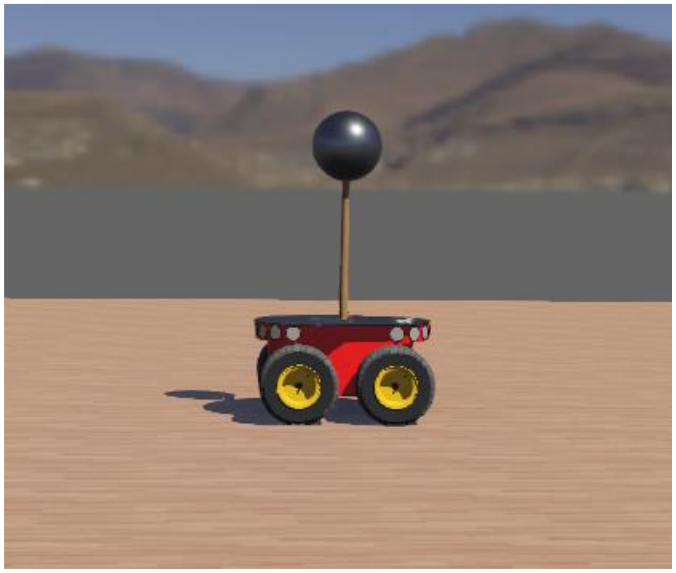

Figura 1. Robô Pioneer com um pêndulo em sua estrutura utilizado no simulador Webots.

Para a compreensão do problema, é necessário considerar algumas variáveis de estado. Primeiro, a observação: todos os estados do robô e do pêndulo, tais como a posição, ângulo e velocidade precisam ser adquiridos. Após isso, as ações: o robô tem apenas duas opções para escolher – esquerda ou direita (entenda como direita e esquerda o robô se movimentando para frente e ré, devido a movimentação na imagem, conforme Figura 1). Dessa forma, a ação esquerda é representada por 0 e a ação direita por 1; em sequência, a recompensa, se o pêndulo está em equilíbrio, a recompensa é 1 senão 0; e, finalmente, o fim de um episódio (percebido pela instrução *done*).

Assim, com o algoritmo do *Q–Learning*, foram feitos três experimentos:

- Duas variáveis para compor o estado: ângulo θ e velocidade angular do pêndulo ω; foi aplicado um decaimento na taxa de exploração (ε).
- Duas variáveis para compor o estado: ângulo θ e velocidade angular do pêndulo ω; não foi aplicado um decaimento na taxa de exploração (ε).
- Quatro variáveis na composição do estado: ângulo θ, velocidade angular do pêndulo ω, posição do robô x, velocidade linear do robô v; foi aplicado um decaimento na taxa de exploração (ε).

Assim, foi possível analisar a interferência da taxa de exploração (ε) nesse problema e, também, a quantidade e qualidade de variáveis para compor os estados.

O Algoritmo HAQL foi implementado de maneira análoga ao *Q-Learning*, porém, foi feita a inclusão da heurística considerando que o robô deveria ter ação ir para esquerda

quando o estado do pêndulo representava esquerda, $H(s_\theta = esquerda, a = esquerda) = +5$, e ir para direita quando o estado do pêndulo representava direita, $H(s_\theta = direita, a = direita) = +5$. Na tabela H foi atribuído valor 0 para todos os outros casos de estado-ação.

Para o Algoritmo DQN, os dados coletados de cada episódio foram usados para treinar o agente, então, ao invés de apenas receber os dados e continuar executando, procurando e atualizando a tabela Q, o robô foi treinado para aprender a melhor ação utilizando uma Rede Neural como base.

4. RESULTADOS E DISCUSSÃO

Para validar os experimentos, foram realizadas 30 repetições de cinco mil episódios cada. Ao concluir esse processo, calculou-se a média móvel dos resultados obtidos, permitindo comparações entre os algoritmos.

Cada experimento realizado gerou um gráfico que contém os episódios no eixo horizontal e as recompensas no eixo vertical. A recompensa apresentada é a média móvel da média das trinta experiências realizadas. Para uma melhor compreensão da aprendizagem, a Figura 2 demonstra um caso em que o robô realiza apenas ações aleatórias, ou seja, sem qualquer forma de aprendizado. No gráfico, observa-se que as recompensas permanecem praticamente constantes, mantendo uma curva estável ao longo do tempo. Por outro lado, para evidenciar a eficácia do aprendizado por reforço no problema, é esperado que a curva de recompensas apresente um crescimento significativo ao longo dos episódios.

O gráfico da Figura 3 exibe o resultado do experimento com o método *Q–Learning* apenas, considerando as três formas de compor as variáveis para representar o estado, conforme apresentado na seção 3. Para os casos com decaimento na taxa de exploração, ε começava com valor igual 1 e decaia de maneira proporcional pelos episódios até chegar no valor mínimo de $0{,}1$. Para o caso sem decaimento na taxa de exploração, ε foi fixado em $0{,}2$. A composição que utilizou apenas as variáveis relacionadas ao pêndulo (ângulo θ e velocidade angular ω) com desconto na taxa de exploração (ε) é representado pela curva em roxo e apresentou os melhores resultados, demonstrando uma solução rápida e mantendo-se estável. Pode-se notar que altos valores de recompensa são alcançados nessa curva; nota-se ainda que a velocidade de estabilização foi maior quando comparada aos outros métodos.

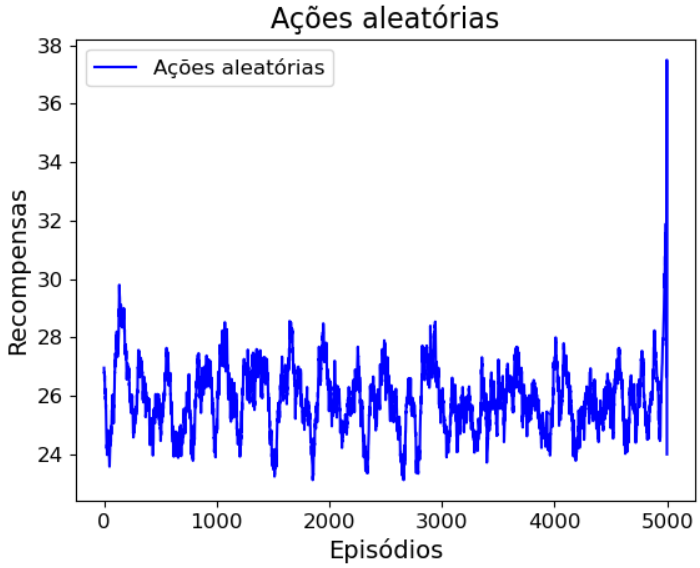

Figura 2. Demonstração de ações aleatórias no problema do pêndulo invertido.

A composição que considerou as variáveis do pêndulo (ângulo θ e velocidade angular ω) e não considerou desconto na taxa de exploração (ε), representado pela curva em verde, apresentou um índice de aprendizado muito baixo e, portanto, atingiu baixos valores de recompensa. Assim, é possível notar que o robô apresentou dificuldade em aprender a política correta de ações, como pode ser visto ao longo dos episódios no gráfico.

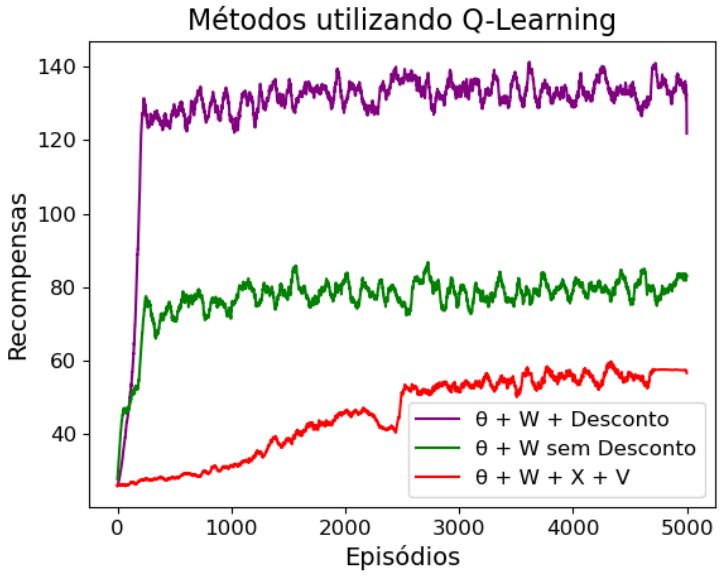

Figura 3. Gráfico demonstrando o resultado da comparação dos métodos apenas com *Q–Learning*.

Dessa forma, é possível perceber a importância da taxa de exploração (ε) e a aplicação de um desconto ou decaimento em seu valor ao longo do tempo.

A composição de estado que considerou as variáveis do pêndulo e do robô (ângulo θ, velocidade angular do pêndulo ω, posição do robô x, velocidade linear do robô v), representado pela curva em vermelho, apresentou um crescimento de recompensa lento, mas constante. De modo geral, com base nesse gráfico da Figura 3, é possível entender as diferenças da quantidade de variáveis na composição do estado e sua influência no aprendizado do agente.

Considerando que o melhor resultado apresentado pelo *Q-Learning* foi aquele em que o estado foi composto por ângulo θ e velocidade angular ω do pêndulo e desconto na taxa de exploração ε, essa estratégia foi a escolhida na implementação dos algoritmos HAQL e DQN. Desa forma, as comparações realizadas entre os algoritmos *Q-Learning*, HAQL e DQN foram realizadas com base nessa composição de estado.

Assim, a Figura 4 apresenta a comparação dos resultados atingidos pelos métodos *Q–Learning* (curva roxa), HAQL (curva ciano) e DQN (curva amarela), todos considerando somente o pêndulo para compor o estado (θ, ω) e desconto em ε.

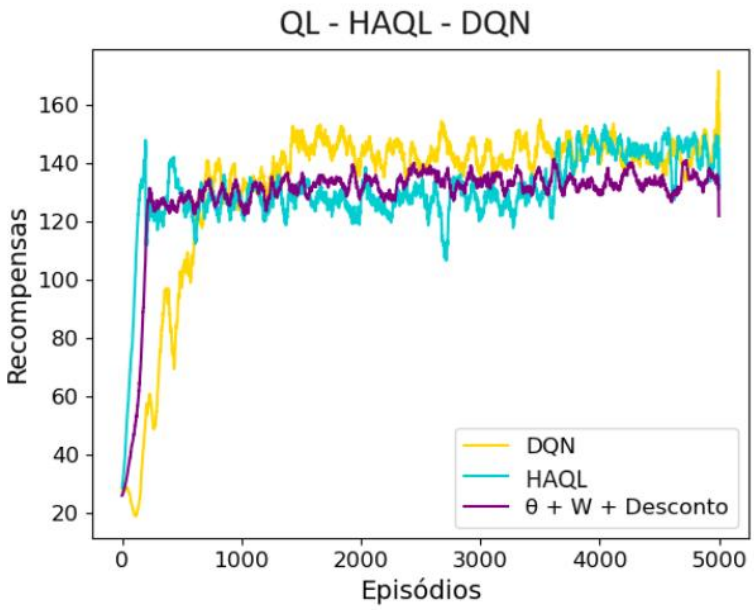

Figura 4. Gráfico demonstrando o resultado da comparação dos métodos *Q–Learning* (curva roxa), HAQL (curva ciano) e DQN (curva amarela).

Na Figura 5 são apresentados os valores médios das recompensas e as barras de erro encontrados dentre as 30 repetições executadas para cada método analisado.

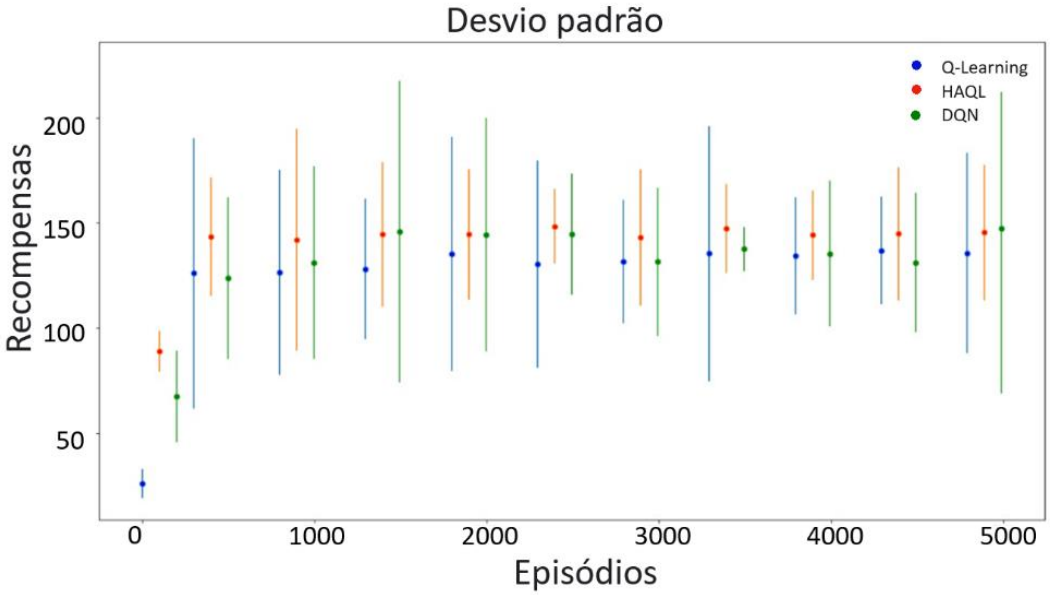

Figura 5. Gráfico demonstrando o valor médio e as barras de erros das recompensas obtidas pelos métodos *Q–Learning*, HAQL e DQN.

Ao decorrer dos episódios é possível notar que, para o caso do *Q-Learning*, o erro apresenta uma variação excessiva ao longo de seus episódios. Para o caso do HAQL, o erro diminui ao longo dos episódios e, em alguns casos, é encontrado uma maior variação do erro, o que significa que mesmo com a heurística o algoritmo segue aprendendo. Por fim, para o caso do DQN, assim como no *Q-Learning*, considerando os 5000 episódios, existe uma grande variação de erro. O gráfico da Figura 6 apresenta todos os experimentos analisados.

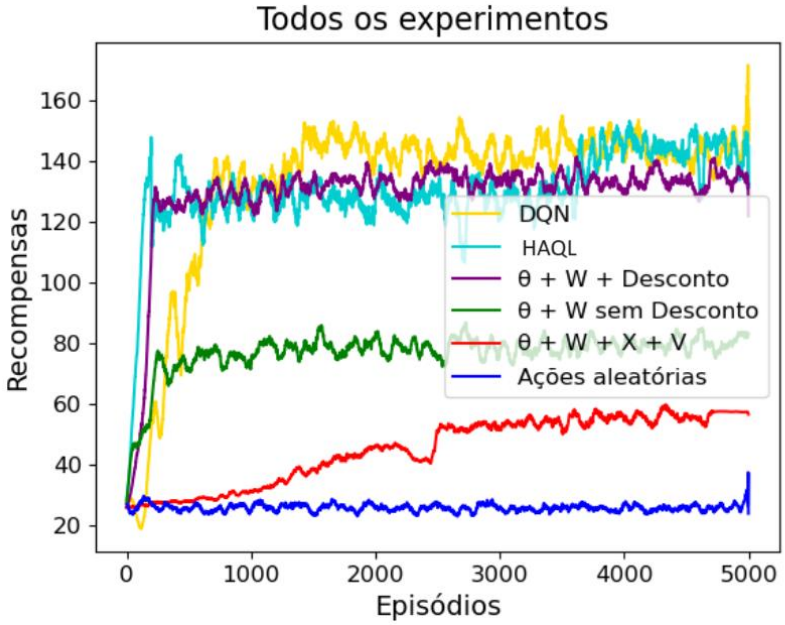

Figura 6. Todos os experimentos realizados.

5. CONSIDERAÇÕES FINAIS

Em conclusão, os experimentos realizados neste artigo proporcionaram resultados valiosos sobre a aplicação de algoritmos de Aprendizado por Reforço no controle de posição de um pêndulo invertido em um robô móvel. A pesquisa adotou uma abordagem quantitativa de natureza aplicada, visando a obtenção de dados objetivos para avaliar e comparar os algoritmos.

Ao analisar os resultados, fica evidente que a escolha do algoritmo desempenha um papel crucial no aprendizado do robô. No gráfico da Figura 2, que representa ações aleatórias, observa-se que o robô não apresenta aprendizado algum, com recompensas praticamente constantes ao longo dos episódios.

No gráfico da Figura 3, que se concentra no método *Q-Learning*, fica claro que a escolha das variáveis e a taxa exploração com ou sem desconto têm um impacto substancial no desempenho do algoritmo. O método que utiliza apenas duas variáveis (ângulo θ e velocidade angular ω do pêndulo) com desconto na exploração demonstra aprendizado rápido e estabilidade. Em contrapartida, o método sem desconto apresenta um aprendizado mais lento e menos eficaz. O gráfico da Figura 4 amplia a análise, comparando o *Q-Learning* com DQN (*Deep Q-Network*) e um método que incorpora heurística (HAQL). O HAQL demonstrou bom desempenho, principalmente nos primeiros episódios. A heurística ajuda o algoritmo a acelerar o processo de aprendizado, pois insere algum conhecimento do domínio. O DQN alcança recompensas significativas por volta do episódio 1500 e busca estabilidade ao longo do aprendizado.

Conforme o objetivo desse trabalho, três modelos de aprendizado foram implementados, incluindo variações em seus parâmetros, o que permitiu a comparação desses métodos dentro do problema proposto. Os resultados alcançados forneceram uma visão detalhada das complexidades envolvidas na aplicação de algoritmos de Aprendizado por Reforço em um problema de controle no domínio da robótica móvel.

REFERÊNCIAS

ABADI, M. et al. TensorFlow: A System for Large-Scale Machine Learning. *In:* 12th USENIX symposium on operating systems design and implementation (OSDI 16). p. 265-283. 2016. Savannah, USA. **Anais[...]**. Disponível em: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf. Acesso em: 20 dez. 2023.

BIANCHI, R. A. C. **Uso de heurísticas para a aceleração do aprendizado por reforço.** Tese (Doutorado) – Universidade de São Paulo, São Paulo, 2004. Disponível em: https://www.teses.usp.br/teses/disponiveis/3/3141/tde-28062005-191041/pt-br.php. Acesso em: 02 nov. 2023.

BROCKMAN, G., CHEUNG, V., PETTERSON, L., SCHNEIDER, J., SCHULMAN, J., TANG, J., ZAREMBA, W. Openai gym. **arXiv preprint arXiv:1606.01540**. 2016. *Preprint*. Disponível em: https://arxiv.org/pdf/1606.01540.pdf. Acesso em: 15 out. 2023.

LIMA, A. L.; SOUZA, P. H. M; REIS, J. Controlador Nebuloso Aplicado ao Equilíbrio de um Pêndulo Invertido Simples. **Revista Cereus**, v. 14, n. 2, p. 72-86, 6 jul. 2022.

MNIH, Volodymyr et al. Human-level control through deep reinforcement learning. **Nature**, Nature Publishing Group, v. 518, n. 7540, p. 529–533, 2015. Disponível em: https://www.nature.com/articles/nature14236. Acesso em: 17 jan. 2024.

MNIH, Volodymyr et al. Playing atari with deep reinforcement learning. **arXiv preprint arXiv:1312.5602**, 2013. *Preprint.* Disponível em: https://arxiv.org/pdf/1312.5602.pdf. Aceso em: 11 jan. 2024.

PERICO, D. H.; BIANCHI R. A. C. Use of Heuristics from Demonstrations to Speed Up Reinforcement Learning. *In:* XI Simpósio Brasileiro de Automação Inteligente (SBAI). 2013. Fortaleza. **Anais [...]**. Disponível em: http://www.sbai2013.ufc.br/pdfs/4908.pdf. Acesso em: 08 nov. 2023.

SEWAK, Mohit. Deep Q Network (DQN), Double DQN, and Dueling DQN. *In*: SEWAK, Mohit. **Deep Reinforcement Learning: Frontiers of Artificial Intelligence**. Springer Singapore, 2019. P. 95–108. ISBN 978-981-13-8285-7. DOI: 10.1007/978-981-13-8285-7_8. Disponível em: https://link.springer.com/chapter/10.1007/978-981-13-8285-7_8. Acesso em: 12 dez. 2023.

SUTTON, Richard S. **Learning to predict by the methods of temporal differences.** Machine learning, Springer, v. 3, n. 1, p. 9–44, 1988. Disponível em: https://link.springer.com/article/10.1007/BF00115009. Acesso em: 08 out. 2023.

WEBOTS. **Open-source Mobile Robot Simulation Software**. Cyberbotics Ltd. 2022. Disponível em: http://www.cyberbotics.com. Acesso em: 10 jun. 2022.

WATKINS, C. J. C. H.; DAYAN, Peter. **Q-learning.** Machine learning, Springer, v. 8, n. 3-4, p. 279–292, 1992. Disponível em: https://link.springer.com/article/10.1007/BF00992698. Acesso em: 10 ago. 2023.

WATKINS, C. J. C. H. **Learning from Delayed Rewards.** Tese (Doutorado) – Cambridge University. 1989. Disponível em: https://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf. Acesso em: 05 set. 2023.

YU, Liang et al. A review of deep reinforcement learning for smart building energy management. **IEEE Internet of Things Journal**, v. 8, n. 15, p. 12046-12063, 2021. Disponível em: https://ieeexplore.ieee.org/document/9426901. Acesso em: 12 jan. 2024.