
UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Renilson da Silva Albuquerque

# Um currículo de aprendizagem por reforço para recompensas modeladas no Lunar Lander

Recife

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

R413c Albuquerque, Renilson
Um currículo de aprendizagem por reforço para recompensas modeladas no Lunar Lander / Renilson Albuquerque. - 2021.
43 f. : il.

Orientador: Pablo Azevedo Sampaio.
Inclui referências.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Ciência da Computação, Recife, 2021.

1. Aprendizagem por reforço. 2. Modelagem de recompensas. 3. Aprendizagem por currículo. I. Sampaio, Pablo Azevedo, orient. II. Título

CDD 004

**MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO**
**UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)**
**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

http://www.bcc.ufrpe.br

**FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO**

Trabalho defendido por Renilson da Silva Albuquerque às 15 horas do dia 19 de julho de 2021, no link https://meet.google.com/hrf-hxka-dnz, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado Um currículo de aprendizagem por reforço para recompensas modeladas no Lunar Lander, orientado por Pablo Azevedo Sampaio e aprovado pela seguinte banca examinadora:

---

Pablo Azevedo Sampaio
DC/UFRPE

---

Filipe Rolim Cordeiro
DC/UFRPE

Renilson da Silva Albuquerque

# Um currículo de aprendizagem por reforço para recompensas modeladas no Lunar Lander

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Computação

Curso de Bacharelado em Ciência da Computação

Orientador: Pablo Azevedo Sampaio

Recife

2021

*Dedico este trabalho à todas as pessoas que me ajudaram a completar esta etapa importante da minha vida.*

# Agradecimentos

Agradeço à minha família, em especial a minha mãe que sempre me ajudou de todas as formas possíveis, me disponibilizando sempre o melhor que poderia oferecer.

Aos meus amigos de longa data que continuaram ao meu lado ao longo dessa jornada, assim como aos novos amigos que fiz durante a graduação, que me ensinaram coisas incríveis e me ajudaram a conquistar grandes feitos que irei levar para o resto da vida.

Aos professores que sempre me ensinaram da melhor forma possível, em especial o meu orientador, que sempre foi paciente e didático. Agradeço a Universidade Federal Rural de Pernambuco em geral por me oferecer ensino e estrutura de qualidade.

*“Não aceito mais as coisas que não posso mudar. Estou mudando as coisas que não posso aceitar.”*

*(Angela Davis)*

# Resumo

A aprendizagem por reforço é um paradigma de aprendizagem de máquina onde o agente aprende a resolver problemas interagindo com um ambiente através de ações executadas em uma lógica de tentativa e erro. A cada ação executada, o agente recebe uma recompensa do ambiente indicando o quão efetiva foi em relação a resolução do problema, de forma que o objetivo do agente consiste em maximizar a recompensa total recebida. Porém, em alguns sistemas de aprendizagem por reforço o agente precisa aprender tarefas muito complexas que atribuem recompensas não muito informativas, gerando assim o *problema de atribuição de crédito* que torna a aprendizagem do agente muito lenta. A *modelagem de recompensas* e a *aprendizagem por currículo*, são técnicas que podem acelerar o tempo de treinamento do agente ao separar o problema em tarefas menores a serem resolvidas sequencialmente, atribuindo recompensas menores e mais informativas por ação executada. O *Lunar lander* é um simulador 2D simplificado, utilizado como referencial para a aplicação de soluções de aprendizagem por reforço para o problema de otimização do controle de pouso de um módulo lunar. Porém o seu sistema de recompensas padrão atribui muito mais recompensas punitivas pelo uso dos motores, não sendo muito construtivo para o agente, o que pode levar ao *problema de atribuição de crédito*. Neste sentido, este trabalho propôs um currículo utilizando dois novos modelos de recompensas, onde foram realizados experimentos a fim de minimizar o tempo de aprendizado do *Lunar Lander*. Foi constatado neste trabalho que ambos os novos modelos e o currículo, foram mais efetivos em treinar o agente do *Lunar Lander*, em comparação ao modelo de recompensas padrão.

**Palavras-chave**: Aprendizagem por reforço, Modelagem de recompensas, Aprendizagem por currículo.

# Abstract

Reinforcement learning is a machine learning paradigm where the agent learns to solve problems interacting with an environment, executing actions in a trial and error sequence. For each action performed, the agent receives a reward from the environment indicating how effective it was in solving the whole problem. The agent's objective is to maximize the total reward received. However, in some reinforcement learning problems, the agent needs to learn complex tasks receiving uninformative rewards, leading to the *credit assignment problem* that slows the agent's training process. Reward shaping and curriculum learning are techniques that can speed up agent training time by separating the problem into smaller tasks to be solved sequentially, applying smaller and informative rewards for each action performed. *Lunar Lander* is a simplified 2D simulator used as a benchmark for reinforcement learning solutions to the optimization problem on landing control of a lunar module. However, its standard rewards system assigns much more punitive rewards for the use of the engines, not being very constructive for the agent, which can lead to the credit assignment problem. Hence, this work proposes a curriculum using two additional shaped reward models and runs experiments that aim to minimize the *Lunar Lander* learning time. This work found that both the new models and the curriculum were more effective in training the Lunar Lander agent compared to the standard rewards model.

**Keywords**: Reinforcement Learning, Reward shaping, Curriculum learning.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| AC | Aprendizagem por currículo |
| AR | Aprendizagem por Reforço |
| LL | Lunar Lander |
| MR | Modelagem de recompensa |
| PDM | Processo de Decisão de Markov |
| PG | Policy Gradient |
| PPO | Proximal Policy Optimization |

# Sumário

# 1 Introdução

Aprendizagem por reforço é um paradigma de aprendizagem de máquina onde um agente aprende interagindo com um ambiente, obtendo recompensas como resposta (LAUD, 2004). Aprendizagem por reforço tornou-se um tópico relevante em inteligência artificial tendo aplicações em diversas áreas do conhecimento tais como: otimização de reações químicas (ZHOU; LI; ZARE, 2017), controle de tráfico de trânsito (AREL et al., 2010), recomendações de notícias (LIU; DOLAN; PEDERSEN, 2010) e jogos eletrônicos (MNIH et al., 2013).

Na aprendizagem por reforço o agente aprende o que fazer e como mapear situações em ações (SUTTON; BARTO, 2018; LAUD, 2004). Dada a incerteza acerca do ambiente, o agente tem a tarefa de achar as melhores sequências de ações a fim de maximizar a recompensa total recebida (LAUD, 2004). Além de agente e ambiente, um sistema de aprendizagem por reforço possui quatro subelementos: Uma política, um sistema de recompensa, uma função de valor e um modelo do ambiente (estados) (SUTTON; BARTO, 2018).

A política é uma função que, em casos de sistemas determinísticos, mapeia os estados em ações que podem ser executadas pelo agente (LAUD, 2004), logo, o agente tem como objetivo achar uma política que leve a alcançar a melhor recompensa média; o sistema de recompensas define o objetivo a ser alcançado em um problema de aprendizagem por reforço, a cada ação executada pelo agente, o ambiente o envia uma recompensa (SUTTON; BARTO, 2018)

Nos últimos anos, a aprendizagem por reforço profunda recebeu bastante atenção da comunidade científica, contando com avanços empíricos consideráveis em várias áreas como jogos e robótica (AYOUB et al., 2020). As aplicações recentes em aprendizagem por reforço variam em muitas áreas do conhecimento, como podemos observar nestes trabalhos:

Em *Learning to Simulate Vehicle Trajectories from Demonstrations* (Zheng et al., 2020) é proposto um sistema de aprendizagem por reforço para simular tráfego de trânsito.

Em *Delivery Route Optimization Based on Deep Reinforcement Learning* (XING; CAI, 2020) é proposto um algoritmo de aprendizagem por reforço profunda para encontrar melhores rotas em sistemas de delivery.

Em *Single Image Dehazing via Reinforcement Learning* (ZHANG; DONG, 2020) é proposto um algoritmo para o descoloramento de imagem, com o objetivo de restaurar

a imagem limpa de uma imagem turva.

Em *Energy efficient transmission in underlay CR-NOMA networks enabled by reinforcement learning* (LIANG et al., 2020) é proposto um algoritmo de aprendizagem por reforço para melhorar a eficiência energética nas redes de rádio cognitivo subjacentes.

Em *5G Resource Scheduling for Low-latency Communication: A Reinforcement Learning Approach* (HUANG; KADOCH, 2020) é proposto uma abordagem de aprendizagem por reforço utilizando um espectro de rádio que garante a restrição de baixa latência quando os recursos de espectro são insuficientes, a fim de melhorar os serviços de rede sem fio em tempo real.

## 1.1 Dificuldades para treinar Aprendizagem por reforço

Em sistemas com tarefas complexas ou tarefas com recompensas esparsas, o aprendizado pode ser extremamente lento, a principal razão disto é a alta quantidade de passos a serem executados para obter alguma recompensa, logo o agente passa um tempo considerável realizando ações aleatórias até receber alguma indicação mais efetiva de que está progredindo ou não no aprendizado (BRYS et al., 2015). Tal situação é um desafio para a aprendizagem por reforço pois se torna complexo para o sistema avaliar quais ações contribuíram mais para a conquista do objetivo, caracterizando o *problema de atribuição de crédito* que dificulta na tomada de decisão de qual ação executar em seguida, tendo em vista o problema de *exploration* e *exploitation* (Matiisen et al., 2019).

## 1.2 Motivações

Tendo em vista o problema das *recompensas esparsas* e *atribuição de crédito*, uma possível solução é a *modelagem de recompensa (MR)*, que consiste em aplicar uma recompensa $F$ à recompensa $R$ natural do sistema, fazendo o agente receber uma composição das duas (BRYS et al., 2015). É esperado que a recompensa resultante seja mais informativa para o agente durante o processo de aprendizagem. Alguns trabalhos foram realizados utilizando modelagem de recompensas em problemas diversos, como em *Policy Transfer using Reward Shaping*, de *Tim Brys, Anna Harutyunyan* e *Matthew E. Taylor* (BRYS et al., 2015) onde é aplicada uma recompensa parcial ao treinamento de um agente para a solução do *Heavy Cart Pole problem*; assim como em *Learning to drive a bicycle using reinforcement learning and shaping* (RANDLØV; ALSTRØM, 1998) onde é proposto um modelo de recompensas parciais para treinar um agente inteligente a se equilibrar em uma bicicleta.

Outra técnica utilizada para contornar tais problemas é a *aprendizagem por currículo*. Nesta abordagem, ao invés do agente tentar aprender todo o problema de uma vez, primeiro aprende a resolver tarefas menores, que são organizadas em uma sequência a fim de acelerar ou melhorar o aprendizado (NARVEKAR, 2017). A aprendizagem por currículo foi utilizada para resolver problemas de AR em alguns trabalhos, como em *Transfer of Samples in Batch Reinforcement Learning* (LAZARIC; RESTELLI; BONARINI, 2008), onde foi proposto um currículo para a resolução do problema do barco, que consiste em ensinar um agente a pilotar um barco do banco esquerdo até o direito através de um rio com uma correnteza não linear, assim como em *Autonomous Task Sequencing for Customized Curriculum Design in Reinforcement Learning* (NARVEKAR; SINAPOV; STONE, 2017), onde é proposto um currículo para resolver o problema do *grid world domain*.

O *Lunar lander* do *GYM* é um referencial para a estudos e aplicação de soluções de AR para o problema de otimização de pouso de um módulo lunar. Porém, suas recompensas padrões podem não ser muito informativas para o agente, pois atribui muitas recompensas negativas a cada uso das ignições, sem aplicar mais recompensas positivas para indicar objetivamente o comportamento ideal. Estas recompensas negativas que quase geram uma estrutura próxima a de *penalidade por viver*, podem gerar o *problema de atribuição de crédito*, dificultando o aprendizado do agente.

Os estudos de otimização realizados no *Lunar lander*, em sua maioria, são alterações na arquitetura da rede neural ou no algoritmo de AR utilizado. A aplicação de *modelagem de recompensas* e *aprendizagem por currículo* no *Lunar Lander* é encontrada com mais frequência em blogs e repositórios de código. Na literatura é restrita a poucos trabalhos como em *Powered Landing Guidance Algorithms Using Reinforcement Learning Methods for Lunar Lander Case* (NUGROHO, 2021) onde é aplicada a modelagem de recompensas juntamente com aprimoramentos na rede neural no ambiente do *Lunar lander*. Não foram encontradas, na literatura, aplicações simultâneas das duas técnicas no *Lunar Lander* do *GYM*.

Neste sentido, este trabalho consiste de experimentos onde serão propostos novos modelos de recompensas, estruturados sequencialmente em tarefas de currículos para treinar um agente do ambiente do *Lunar Lander*, a fim de contornar os possíveis problemas da *atribuição de crédito* e minimizar o tempo de aprendizagem comparado ao agente treinado com as recompensas padrões do ambiente.

# 2 Literatura sobre Aprendizagem por Reforço

## 2.1 Processo de Decisão de Markov

O *processo de decisão de Markov* (PDM) é uma formalização do problema de tomada de decisão sequencial, onde as ações influenciam não só as recompensas imediatas, mas as situações subsequentes, estados e recompensas futuras. (SUTTON; BARTO, 2018). O PDM tem sido utilizado como um framework geral para formalização de problemas de AR.

Intuitivamente, o processo de decisão de Markov consiste do aprendiz e tomador de decisões, o chamado *agente*, interagir continuamente com o ambiente, em uma sequência de passos $t$, onde $t = 1, 2, 3, 4...T$, sendo $T$ a quantidade total de passos a serem executados (SUTTON; BARTO, 2018). Alguns problemas de AR possuem a noção natural de passo final, onde a interação entre agente e ambiente é interrompida naturalmente e dividida em subsequências chamadas de *episódios*. A cada passo da interação, o agente recebe uma percepção $o_t$ do ambiente, indicando o atual estado do sistema no passo $t$ (ARULKUMARAN et al., 2017). O agente então toma uma decisão e executa uma determinada ação neste ambiente, mudando assim o estado de ambos, agente e ambiente (ARULKUMARAN et al., 2017).

Formalmente, o Processo de Decisão de Markov (*PDM*) consiste em uma 5-tupla $M = (S, A, P, R, \gamma)$ onde temos que:

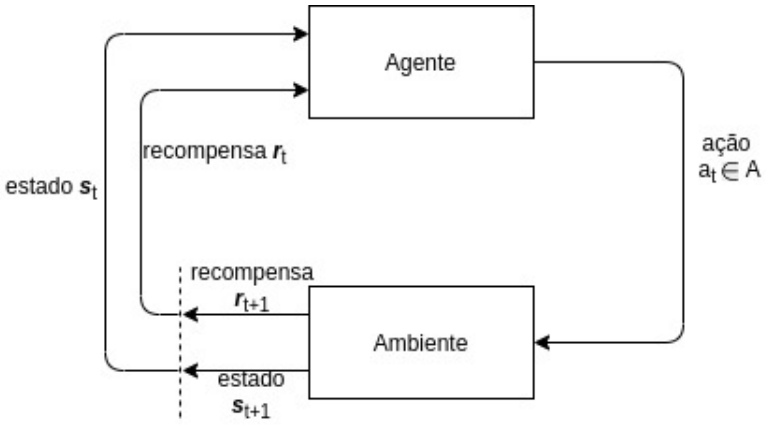


Figura 1 – Descrição de uma iteração de um PDM. Elaborado pelo autor (2021)

- $S$ é um conjuno de estados $s$ sendo $s_t$ o estado $s$ do agente em um tempo $t$.
- $A$ é um conjunto de ações $a$ onde $a_t$ é a ação $a$ executada pelo agente no tempo $t$.

- Uma função de transição $P$, tal que $P(s, a, s') : S \times A \times S \to [0, 1]$. Que retorna a probabilidade de acontecer a transição do estado $s$ para $s'$, após executar uma ação $a$.
- Uma função de recompensa $R$, tal que $R : S \times A \times S \to R$ que retorna para o agente uma recompensa imediata $R(s, a, s')$ após o agente executar uma ação $a$.
- Um fator de desconto $\gamma \in [0, 1]$. Enquanto $\gamma$ for mais próximo de zero $\gamma = 0$, o agente se torna "viciado" em maximizar apenas a recompensa imediata $R_{t+1}$; enquanto $\gamma$ mais próximo de um $\gamma = 1$, leva o agente a se preocupar mais com as recompensas futuras. (YU; LIU; NEMATI, 2019; SUTTON; BARTO, 2018)

Um estado $s_t$ contém a propriedade de Markov se apenas este afeta o próximo estado $s_{t+1}$. (ARULKUMARAN et al., 2017).

Uma política $\pi$ é um mapeamento de estados em probabilidades de escolher uma possível ação, descrito como $\pi : S \times A \to [0, 1]$. (YU; LIU; NEMATI, 2019).

O valor esperado de um estado $s$ sob uma política $\pi$ é formalmente definido por $v_\pi$ descrito na equação 2.1, que de forma mais simples, é uma função que indica o quão bom é estar em um estado $s$ seguindo a política $\pi$, calculando a recompensa acumulada a partir do estado $s$ seguindo a política $\pi$.

$$v_\pi = E_\pi[G_t|S_t = s], \forall s \in S \tag{2.1}$$

$G_t$ é a recompensa acumulada que o agente obtém a partir do passo $t$, formalmente definido na equação 2.2

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.2}$$

Também podemos definir o valor de uma ação $a$ em um estado $s$, seguindo uma política $\pi$, como $q_\pi(s, a)$ onde temos:

$$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a]$$

A função de valor de ação que obtem a maior recompensa acumulada sobre todas as políticas é chamada de *função de valor de ação ótima*, denotada por $q^*(s, a)$. Em outras palavras, ao encontrar $q^*$, significa que se sabe a melhor ação a ser tomada em cada estado, resolvendo assim o PDM.

## 2.2 Aprendizagem por reforço

Aprendizagem por reforço (AR) é aprender o que fazer e como mapear estados em ações de forma a maximizar a recompensa acumulada recebida (SUTTON; BARTO, 2018). Em um problema de AR, um agente autônomo controlado por um algoritmo, interage com o ambiente e pode aprender a alterar seu comportamento observando as consequências das suas ações em resposta às recompensas recebidas (ARULKUMARAN et al., 2017). Esse paradigma de aprendizado por tentativa e erro vem da psicologia behaviorista e é um dos principais fundamentos da AR (ARULKUMARAN et al., 2017).

Na aprendizagem por reforço, a distinção entre as atribuições do agente e do ambiente pode ser resumida em: tudo o que o agente não pode controlar é considerado parte do ambiente (ARULKUMARAN et al., 2017). De forma mais descritiva e sequencial, em uma interação de um problema de AR, o agente está em um estado, recebe informações sobre o ambiente, toma uma decisão e executa uma ação, levando o sistema para um próximo estado (ARULKUMARAN et al., 2017). A cada transição de estados, o ambiente atribui uma recompensa para o agente, que por sua vez, deve se comportar e interagir com o ambiente de forma a maximizar a recompensa recebida (KAELBLING; LITTMAN; MOORE, 1996; SUTTON; BARTO, 2018). A lógica que guia o comportamento do agente sobre quais decisões tomar em cada estado é chamada de *política* (SUTTON; BARTO, 2018), deste modo, a resolução de um sistema de AR se resume em encontrar uma política ótima que maximize a recompensa acumulada esperada(ARULKUMARAN et al., 2017).

Ao contrário de outros problemas de aprendizagem de máquina, de forma geral, os problemas de AR não possuem a dinâmica do seu modelo explícita, tornando necessário ao agente explorar o ambiente para aprender a política ótima por tentativa e erro (SUTTON; BARTO, 2018; WIERING; OTTERLO, 2012). Porém, esta definição implica em um dos maiores desafios de AR, o equilíbrio entre *exploration* vs *exploitation* (SUTTON; BARTO, 2018).

Para obter uma maior recompensa, o agente intuitivamente deve preferir *explorar*, no sentido de tirar proveito (*exploitation*), ações executadas em passos anteriores que provaram ser efetivas (SUTTON; BARTO, 2018). Porém, para descobrir tais ações e a sua efetividade, deve *explorar*, no sentido de desbravar (*exploration*), novas ações e estados, com o objetivo de fazer melhores escolhas no futuro (WIERING; OTTERLO, 2012; SUTTON; BARTO, 2018). O dilema reside em que o agente deve equilibrar a *exploration* e *exploitation* sem falhar na resolução do problema (SUTTON; BARTO, 2018).

Um outro aspecto importante da tomada de decisão sequencial é a decisão

de qual ação é mais efetiva em cada passo do aprendizado, pois a adequação das ações é completamente determinada pelo objetivo que o agente está perseguindo (WIERING; OTTERLO, 2012). Existem problemas de AR com tarefas complexas ou com recompensas esparsas, que tornam o aprendizado do agente extremamente lento (BRYS et al., 2015), por exemplo, em um jogo de xadrez, os movimentos iniciais tem uma grande influência sobre quem vai vencer o jogo, cada movimento contribui mais ou menos para o sucesso do ultimo movimento, porém a recompensa do jogo (vencer, empatar ou peder) só é definida no final (WIERING; OTTERLO, 2012). O *problema de atribuição de crédito* consiste em decidir como atribuir recompensas aos primeiros passos de forma que não atribua a recompensa imediata por vencer, e de forma geral, distribuir as recompensas ao longo dos passos de forma a representar eficientemente a política do agente, a fim de evitar ações aleatórias sem receber recompensa alguma (WIERING; OTTERLO, 2012; Matiisen et al., 2019).

## 2.3 Algoritmos de AR

Dado o objetivo da aprendizagem por reforço em encontrar o comportamento do agente que o leve a receber a recompensa máxima, existem algumas estratégias algorítmicas para alcançar tal objetivo. Esses algoritmos são divididos em classes de acordo com sua estratégia em comum, tais como os algoritmos de *função de valor*, que focam em estimar o valor esperado de retorno de estar em um determinado estado, e os algorítmos de *Policy search* que, não necessariamente, precisam focar em um modelo de função de valor, mas procuram diretamente por uma política ótima (ARULKUMARAN et al., 2017).

*Policy search* é uma abordagem de AR que foca em encontrar diretamente uma política otima $\pi^*$. A política normalmente é modelada como uma função parametrizada da forma $\pi_\theta(a|s)$ cujos parâmetros de entrada são escolhidos de uma forma a maximizar o retorno esperado $E[R|\theta]$ (ARULKUMARAN et al., 2017).

Para maximizar o retorno esperado de uma função, a estratégia mais comum em algoritmos de *Policy Search* é a aplicação de gradiente, criando assim uma subcategoria de algoritmos de *Policy Search*, os algoritmos de *Policy Gradient* (PG).

Existem vários algoritmos de PG mas para fins didáticos, utilizaremos o *REINFORCE*, um dos pioneiros e mais simples desta catgoria. O *REINFORCE* tem o seu pseudocodigo descrito em 1.

Considerando que $\theta$ seja os parâmetros da política (os pesos da rede neural), o algoritmo primeiramente inicializa $\theta$ arbitrariamente, executa um episódio inteiro e em sequência calcula o valor da função de erro $L(\theta)$ descrita na equação 2.3, que intuitivamente permite aumentar os pesos para ações que retornaram mais recompensas

**Algorithm 1** Reinforce

1: Inicialize $\theta_0$ arbitrariamente
2: **for** $t = 1, 2, \ldots$ **do**
3: Rode 1 episódio, utilizando $\pi_\theta$
4: Calcule $L(\theta)$
5: Atualize $\pi_\theta$ aplicando gradiente de $L(\theta)$
6: **end for**

positivas e diminui para as ações que trouxeram recompensas mais negativas, utilizando o $G_t$, que como explicado na seção 2.1 deste trabalho, é a recompensa acumulada em cada estado ao seguir a política atual.

$$L(\theta_t) = -\frac{1}{T}\sum_{k=1}^{T} G_t * log\pi(a_t|s_t) \tag{2.3}$$

Por fim, sendo $\theta$ os parâmetros da política e a política $\pi_\theta(a|s)$, o *REINFORCE* aplica a regra de atualização para ajustar os próximos valores de pesos $\theta_{t+1}$ a partir dos valores atuais $\theta_t$, para mover o vetor $\theta$ na direção que minimiza $L$. Temos a regra de atualização da política descrita na equação 2.4.

$$\theta_{t+1} = \theta_t + \alpha\nabla L(\theta_t) \tag{2.4}$$

Onde, informalmente, tem se que o gradiente $\nabla\pi\theta_t(a|s)$ indica a direção para onde mover $\theta$ de modo que diminua o valor de $L$ rapidamente.

## 2.4 Métodos de ator-crítico

Comparando com o *REINFORCE*, no modelo *Ator-Crítico*, as alterações na rede são feitas a cada passo, eliminando, para isto, a necessidade de finalizar um episódio inteiro. Logo, não é possível utilizar o $G_t$ para calcular o valor de recompensa acumulada.

Realizando uma alteração na função de erro do *REINFORCE*, podemos trocar o $G_t$ por uma função de vantagem $A_t$, obtendo a equação 2.5.

$$L(\theta_t) = -\frac{1}{T}\sum_{k=1}^{T} A_t * log\pi_\theta(a_t|s_t) \tag{2.5}$$

Onde $A_t = G_t - V(s)$, sendo $V(s)$ o valor médio do estado, chamado de baseline. Esta alteração cria um subgrupo de algoritmos de PG chamados de *Policy Gradient* com *baselines*.

Um algoritmo com estrutura de *Ator-Crítico* é, em resumo, um algoritmo *Policy Gradient* com *baselines*, que executa apenas *K* passos por vez para atualizar os pesos da rede. Logo, podemos utilizar um valor de estimativa da recompensa esperada, alterando a função de vantagem $A_t$ para a descrita na equação 2.6.

$$A_t = \sum_{k=0}^{K-1} (\gamma^k r_{t+k}) + \gamma^K V(s_{t+K}) - V(s_t) \tag{2.6}$$

A estrutura de *Ator-Crítico* propõe uma alteração da arquitetura de um sistema de AR proposto na figura 1, resultando na forma:

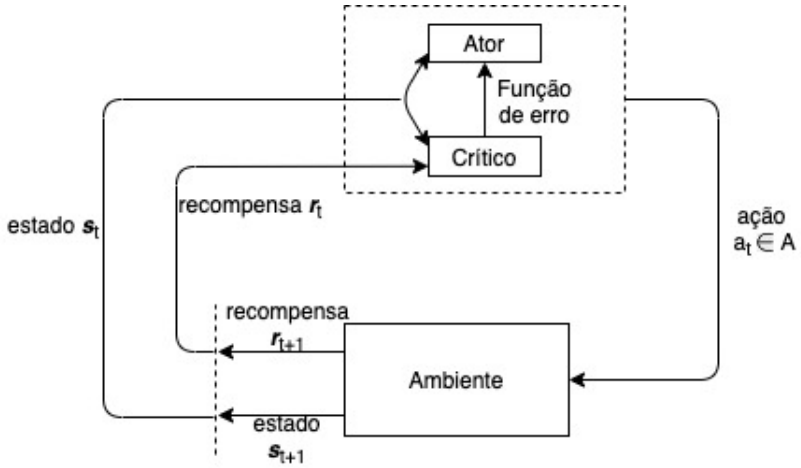


Figura 2 – Descrição de uma arquitetura *Ator-Crítico*. Elaborado pelo autor (2021)

O *ator* é uma rede neural responsável por escolher as ações a serem tomadas. O *crítico*, como o próprio nome remete, critica as ações tomadas pelo ator ao calcular o quão bom foi executar tal ação. Em termos mais práticos, o ator aplica a política, enquanto o crítico calcula a função de valor.

## 2.5 Proximal Policy Optimization - PPO

Um dos maiores desafios em algumas abordagens de resolução de problemas de AR é que uma vez que o modelo adota uma política ruim, ele apenas tomará decisões ruins, levando a um caminho de treino irrecuperável. Em métodos de PG, esse problema pode ocorrer após o algoritmo fazer atualizações do pesos, utilizando o gradiente, provocando uma grande alteração das probabilidades das ações, levando a um colapso na qualidade das novas decisões.

Um dos algoritmos recentes de PG mais popular é o *Proximal Policy Optimization* (PPO) que, em linhas gerais, resolve o problema do gradiente apenas realizando pequenas alterações nos pesos da rede, estabilizando o processo de treinamento.

O PPO é um algoritmo *on-policy*, ou seja, ele não armazena e considera experiências antigas ou produzidas por outras políticas para definir os valores da próxima iteração. Tendo isso em vista, o PPO define uma razão $r$ entre a política atual $\pi_\theta(a_t|s_t)$ e a política mais antiga $\pi_{\theta_k}(a_t|s_t)$ sendo:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \tag{2.7}$$

Para que tenhamos a função de erro descrita em 2.8:

$$L^{CLIP}(\theta) = E_t[-min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)] \tag{2.8}$$

Onde $\epsilon$ é um hiperparâmetro geralmente $\epsilon = 0.2$, e a função $clip$ limita a razão $r_t(\theta)$ (2.7) a ser menor que $1+\epsilon$ e maior que $1-\epsilon$. Assim, a função de erro do PPO considera o valor mínimo entre o valor original e o valor da função $clip$. Desta forma, a rede perde a motivação em atualizar os pesos para valores maiores que $\epsilon$ que alteram excessivamente a política $\pi$, a procura de recompensas maiores.

# 3 Revisão sobre Modelagem de recompensas e Aprendizagem por currículo

## 3.1 Modelagem de recompensas

Tendo em vista o problema de *atribuição de crédito* e *recompensas esparsas* como fatores de dificuldade que desaceleram significativamente o aprendizado de um agente de AR, existe uma estratégia que pode ser aplicada para contornar tal situação, a aplicação de *recompensas modeladas* (MAROM; ROSMAN, 2018).

Agentes de AR tipicamente aprendem como agir no seu ambiente, recebendo unicamente uma recompensa por ação. Uma *recompensa modelada* é um mecanismo para guiar a exploração do agente através da aplicação de uma recompensa adicional à recompensa natural do ambiente (MANNION et al., 2017). De forma geral e formal, temos que uma recompensa modelada consiste em:

$$R' = R + F \tag{3.1}$$

onde $R$ é a recompensa original do ambiente, $F$ é a recompensa adicional a ser adicionada ao agente e $R'$ é a recompensa modelada a ser entregue ao agente (MANNION et al., 2017).

Modelagem de recompensa se provou uma estratégia poderosa no aumento da velocidade de aprendizagem de vários problemas de AR, como por exemplo, jogos de *Atari* (MNIH et al., 2015; GRZES, 2017). Porém, não existe nenhuma garantia de que um PDM com uma recompensa modelada arbitrária terá uma uma política ótima consistente com a original, podendo levar a resultados viciados e totalmente longe da política ótima (MAROM; ROSMAN, 2018; MANNION et al., 2017).

Um exemplo da dificuldade de treinar um agente com uma recompensa modelada pode ser visto em *Learning to drive a bicycle using reinforcement learning and shaping* (RANDLØV; ALSTRØM, 1998), onde foi utilizado uma MR para acelerar o processo de aprendizagem de um agente a andar de bicicleta. Em uma das tentativas de criar um modelo eficiente, o agente descobriu que poderia acumular uma recompensa maior apenas por pedalar em círculos para coletar a recompensa modelada que o encorajava a continuar equilibrado, fazendo-o ignorar o objetivo geral do problema que consistia em se deslocar de um ponto A a um ponto B (RANDLØV; ALSTRØM, 1998).

## 3.2 Aprendizagem por currículo

Uma outra abordagem que pode acelerar o treinamento de agentes de AR é a divisão do aprendizado em tarefas de currículos. Nesta técnica de aprendizagem, os objetivos do sistema são divididos em partes menores para serem treinadas separadamente e sequencialmente. A cada tarefa treinada, a experiência de treino adquirida é passada para a próxima. A ideia básica é começar pequeno, aprender os aspectos mais simples das tarefas e então aumentar o nível de dificuldade (BENGIO et al., 2009).

Para ilustrar melhor como funciona uma *aprendizagem por currículo* (AC), vamos supor um sistema de AR de um jogo de xadrez, onde o objetivo principal é vencer aplicando um xeque-mate no adversário. Porém, para o agente aprender a jogar o jogo completo de uma vez, pode ser bastante complexo, visto que existem diferentes peças com diferentes movimentos, o que faz a quantidade de jogadas possíveis ser muito grande. Então, uma possível abordagem de currículo para acelerar a resolução deste problema seria dividir o jogo total em pequenos jogos dentro do jogo de xadrez completo, como por exemplo, reduzir o tabuleiro para uma versão menor de tamanho 5x5, contendo apenas peões, formando uma tarefa inicial que foca em ensinar o agente a aprender primeiro a jogar com os peões. Em seguida, o aprendizado com peões seria passado para uma próxima tarefa do currículo que além de peões, acrescenta as duas torres no jogo, a fim de aumentar a complexidade e aproveitar o aprendizado anterior. Essa ideia de divisão seria repetida várias vezes até que o agente aprenda a jogar o jogo completo com todas as peças, finalizando o treinamento (NARVEKAR, 2017).

Os desafios de propor e treinar agentes com AC residem na complexidade de dividir as tarefas efetivamente e organizá-las em uma sequência eficiente ao ponto do conhecimento ser cumulativo e não sobreposto com vieses da tarefa aprendida recentemente (NARVEKAR, 2017).

# 4 Modelagem de recompensas no Lunar lander

Ao pesquisar novas estratégias em AR, a comunidade científica precisa de referências para comparar as soluções entre si, e para permitir reproduzir facilmente os resultados obtidos em trabalhos anteriores, gerando assim uma necessidade de padronização nos ambientes onde se aplica AR. Assim, a *OpenAI* desenvolveu o *GYM*, uma ferramenta para pesquisas em AR que inclui uma coleção de ambientes com uma interface em comum (BROCKMAN et al., 2016).

## 4.1 Lunar Lander - Gym

Um desses ambientes inclusos no *GYM* é o *Lunar Lander* (LL), uma simulação episódica e simplificada em 2D do problema de otimização da trajetória de aterrissagem de um módulo lunar. O ambiente consiste de uma nave cujo objetivo é aterrisar com sucesso na área de pouso localizada na coordenada $(0, 0)$ em uma superfície da lua gerada aleatoriamente, como mostrado na figura 3.

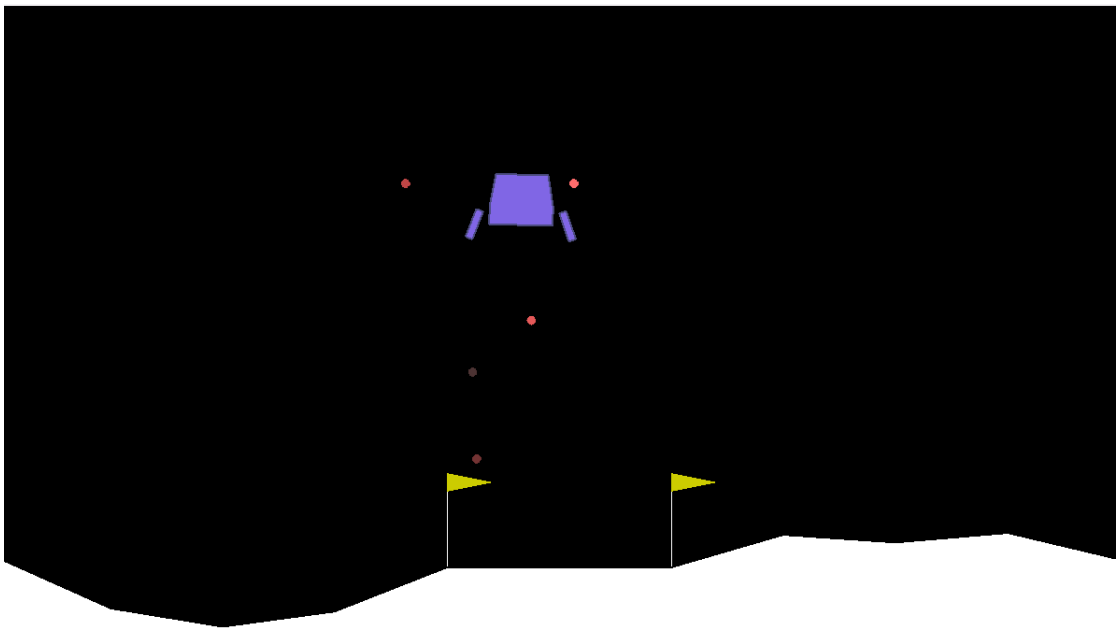

Figura 3 – Ambiente do Lunar lander. Elaborado pelo autor (2021).

Os ambientes do GYM são divididos em discretos e contínuos. Para este trabalho será utilizada a versão discreta do LL, o que significa que as ações executadas pelo

agente estão dentro de um conjunto de números inteiros não negativos que indicam apenas a ação selecionada, sem especificar valores de intensidade como velocidade e força. No ambiente discreto, o agente tem quatro ações possíveis: *(0) fazer nada*, *(1) ativar a ignição da direita*, *(2) ativar a ignição da esquerda* e *(3) ativar a ignição principal*.

No *Lunar lander* as observações padrão consistem de um *array* de 8 números que servem de entrada para a rede neural da política. Os valores do *array* correspondem em sequencia à posição $(x, y)$ da nave, as velocidades horizontal e vertical $(vx, vy)$, o ângulo $\theta$ de inclinação da nave, a velocidade angular $\omega$ e dois valores boleanos representando se cada pé da nave encostou no solo.

O sistema de recompensas padrão do *Lunar lander* define que, ir do topo da tela até a área de pouso atribui ao agente entre 100 a 140 pontos de recompensa. Caso o agente saia da área de pouso, ele perde a recompensa. Cada contato de um pé da nave com o solo, são +10 pontos; ativar a ignição principal custa -0.3 pontos a cada iteração. O episódio acaba se a nave colidir, recebendo -100 pontos de recompensa; ou se a nave pousar com sucesso, recebendo +100 pontos. Não existe limite máximo de pontuação. Considerando que o valor mínimo atribuído por descer em direção a $y = 0$ é de +100 pontos de recompensa e pousar são mais 100, então, segundo a *OpenAI*, um episódio com uma recompensa final de 200 pontos ou mais, é considerado um pouso. O combustível é infinito, portanto, o agente pode aprender a voar e pousar com sucesso na sua primeira tentativa.

Para este trabalho foi utilizado o *Lunar Lander* do *GYM* e a *Stable baselines*, uma biblioteca compatível com a interface do *GYM* que possui implementações melhores dos algoritmos da *Baselines*, a biblioteca padrão da *OpenAI*. Tais melhorias envolvem código mais padronizado, mais documentação e mais testes, o que facilita ainda mais a implementação de novas ideias, inclusive os modelos propostos neste trabalho.

## 4.2 Estrutura do Lunar Lander

A interface em comum que o *GYM* proporciona é composta resumidamente por uma classe *Environment* que possui alguns métodos que indicam o ciclo do treinamento, onde podemos destacar, como exemplo, os mais utilizados neste trabalho:

- *reset()* - executado a cada vez que um episódio é reinicializado.
- *step()* - a cada passo aplica uma ação no ambiente, alterando e retornando a nova observação e recompensa.

Para manipular as observações, acrescentar recompensas adicionais à padrão

do ambiente ou propor alterações em geral, o *GYM* possui uma classe *Wrapper* que contém os mesmos métodos da *Environment* e que pode ser estendida, facilitando a manipulação do ciclo de treinamento sem alterar o código base. Neste trabalho foram elaboradas e testadas várias classes *Wrappers* de acordo com cada modelo proposto, entre eles os modelos 1 e 2 descritos nas seções 4.3 e 4.4 deste trabalho.

## 4.3 Modelo de recompensa 1: Centralizar a nave

A área de pouso do *Lunar lander* corresponde à coordenada $(0, 0)$ e os limites laterais do ambiente, ou seja, os valores da coordenada $x$ se estendem de -1 a +1. Logo, um dos aprendizados do agente ao longo do treinamento é que enquanto estiver pousando, é interessante se manter mais perto de $x = 0$, onde está a area de pouso. O sistema de recompensas padrão do Lunar Lander não atribui nenhuma recompensa positiva ou negativa para o agente, enquanto ainda está no ar, para sinalizar que a sua coordenada $x$ está perto ou longe do ponto de aterrissagem, apenas aplica uma recompensa negativa por utilização das ignições laterais e principal.

Durante o desenvolvimento deste trabalho, alguns modelos de recompensas foram propostos visando esta estratégia de estabilizar o agente no centro. Inicialmente foi testado um modelo de recompensa que a cada passo $t$, atribui o valor absoluto de $|x_t|$ como recompensa negativa ao agente. Este modelo apresentou resultados melhores em comparação ao modelo de recompensas padrão, porém, não é muito educativo para o agente apenas receber recompensas negativas indicando o que não fazer, sendo preferível dizer também qual o caminho ideal a seguir, por meio de recompensas positivas.

Tendo isso em vista, foi desenvolvido um novo modelo que aplica uma recompensa negativa ou positiva de acordo com o deslocamento horizontal do agente no estado $t$ em relação ao passo anterior $t-1$. A aplicação da recompensa é feita de acordo com uma divisão da coordenada horizontal espelhada a partir do zero. Formalmente temos uma substituição do valor de $F$ na equação 3.1 onde:

$$F = a * \begin{cases} x_{t-1} - x_t & \text{se} \quad x_t \geq 0 \\ |x_{t-1}| + x_t & \text{se} \quad x_t < 0 \end{cases} . \tag{4.1}$$

Sendo $a$ um fator de multiplicação arbitrário a fim de intensificar ou diminuir o valor da recompensa aplicada ao agente. A figura 4 ilustra o modelo.

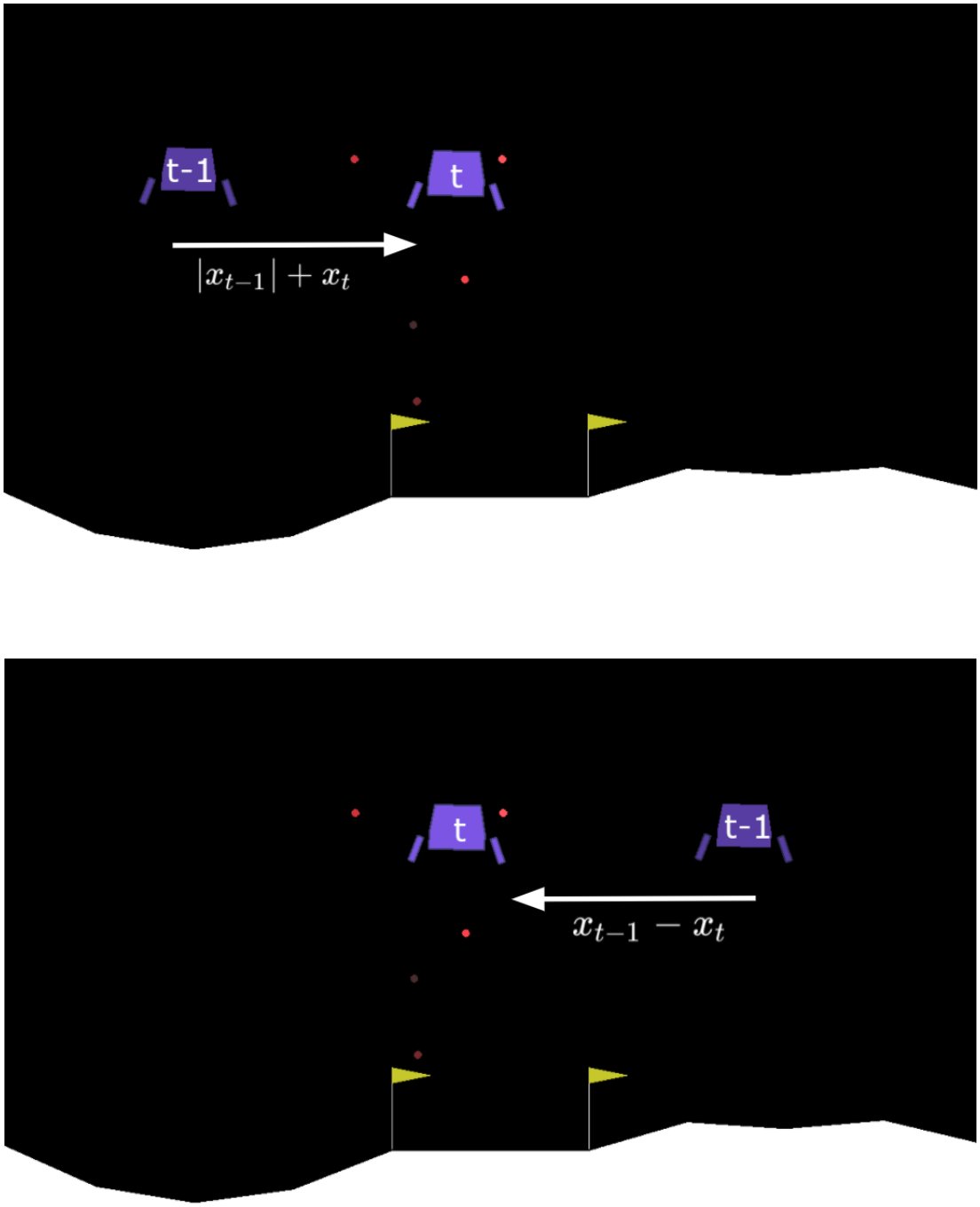


Figura 4 – Ilustração do funcionamento do modelo 1 de centralizar a nave. Elaborado pelo autor (2021).

## 4.4 Modelo de recompensa 2: Descer a nave

O objetivo principal da nave no *Lunar Lander* é pousar na área designada, e para que isso ocorra, intuitivamente precisa-se que a nave seja encorajada a descer. Portanto, alguns modelos foram implementados neste trabalho aplicando recompensas

ao agente de acordo com a sua velocidade vertical e o valor da sua coordenada $y$.

Tendo em vista que deseja-se aproximar o agente do solo, ou seja, de que $y = 0$, então a ideia mais intuitiva é de aplicar uma recompensa negativa ao agente de acordo com o valor da sua coordenada $y$, assim, quanto mais alto e distante do chão o agente estiver, mais recompensas negativas ele receberá. Este modelo ensina o agente a descer mais rapido que o modelo padrão do LL, considerando ainda que este encoraja o agente a ”cair”, ou seja, utiliza com bem menos frequência a ignição principal para desacelerar a descida e com isso, reduz significativamente as recompensas negativas de -0.3 a cada iteração. Porém, se o valor da recompensa for alto o suficiente ao ponto do somatório das recompensas ao final do episódio ser mais atrativo do que pousar com sucesso ou cair, o agente se torna viciado em simplesmente cair e destruir a nave.

Analogamente ao modelo inicial apresentado na seção 4.3 deste capítulo, existe a alternativa de aplicar o valor proporcional à variação do deslocamento do agente em relação a área de pouso, ou seja, aplicar uma recompensa positiva caso $y_t$ for menor que $y_{t-1}$, porém esta ideia na prática mostrou-se mais destrutiva, incentivando ainda mais o agente a cair e destruir a nave.

Outros modelos foram implementados visando esta estratégia de descer a nave, porém a maioria deles cai em dois cenários de falha possíveis : caso seja aplicada uma recompensa de valor baixo, o modelo não afeta ou acelera significativamente a curva de aprendizado natural do agente no problema do LL; e caso seja aplicada uma recompensa de valor alto, o agente aprende a simplesmente cair e quebrar a nave.

A solução que apresentou melhores resultados neste sentido foi o modelo que aplica recompensas positivas ao agente por cada marco alcançado ao descer cada vez mais em direção à área de pouso. Supondo que existam quatro marcos $m$ distribuidos sequencialmente no caminho do agente até o ponto $(0, 0)$, então, a cada vez que o agente alcançar o um marco $m_n$ ele recebe uma recompensa r, como mostrado na figura 5.

Desta forma, tem-se um incentivo para que o agente desça de uma forma mais controlada, ativando a ignição principal de forma mais restrita.

## 4.5 Outros modelos implementados

Várias ideias foram implementadas para resolver o problema, mas a maioria foi avaliada como muito complexa ou não muito instrutiva para o agente.

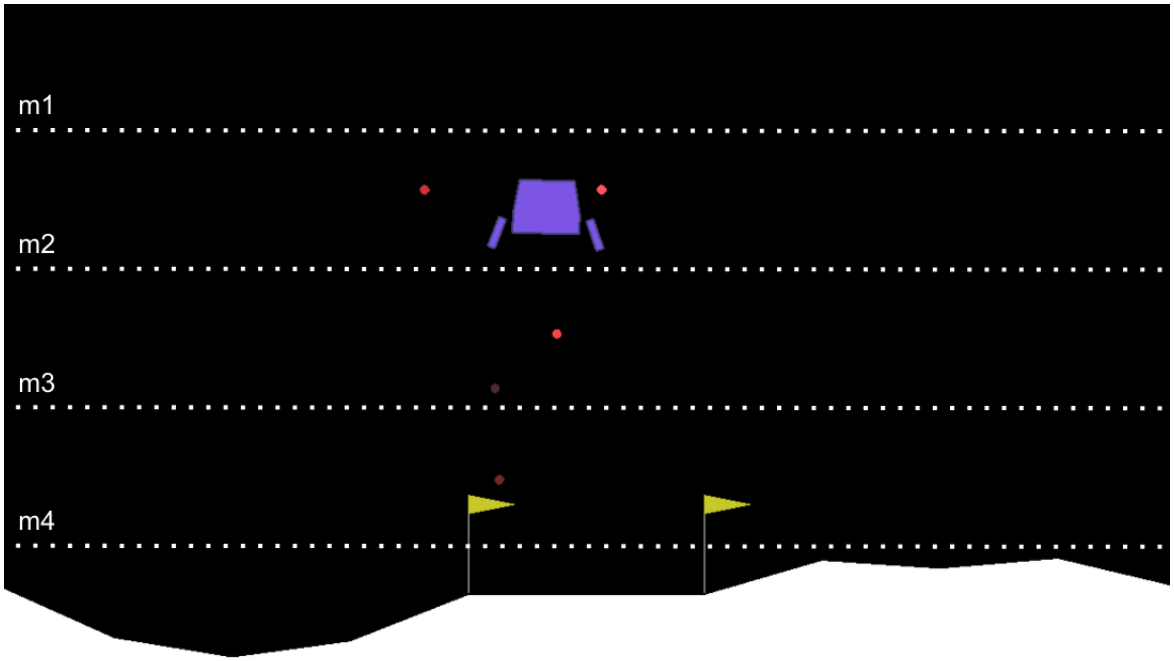


Figura 5 – Ilustração do funcionamento do modelo 2 de descer a nave com 4 marcos. Elaborado pelo autor (2021).

### 4.5.1 Modelo 3: Distância entre coordenadas

Visto a importância de incentivar o agente a descer e centralizar, descrito nos modelos 1 e 2, intuitivamente tem-se a ideia de que a distância entre a posição do agente e a coordenada $(0, 0)$, é uma reta. Logo, foi construído um modelo que aplica a distância euclidiana da posição $(x_t, y_t)$ da nave até a área de pouso $(0, 0)$ como recompensa negativa para o agente, com o objetivo de atraí-lo, buscando livrar-se da recompensa negativa. Porém, a resolução deste problema depende de duas variáveis simultaneamente, o que tornou, em alguns casos, confuso para o agente.

### 4.5.2 Modelo 4: Controle de velocidade

Um dos problemas recorrentes do treinamento no *Lunar lander* é o equilíbrio da utilização dos motores, pois ao passo que são fundamentais para controlar a nave, trazem recompensas negativas por uso. Em vários casos, o agente aprende a desligar o motor, cair e ligá-lo quando estiver muito perto de $y = 0$. Porém nestes casos, manobrar a nave é quase impossível, o que acarreta em muitos episódios onde a nave é destruída. Portanto, foram implementados modelos de controle de velocidade. A maioria destes foram rapidamente descartados pois necessitavam explicitamente de uma velocidade ideal, descaracterizando os aspectos de AR. O modelo decorrente desta ideia que obteve mais destaque foi o que atrela a velocidade de descida à coordenada vertical da

nave, atribuindo uma recompensa $r' = y - vy$. Desta forma, a velocidade $vy$ precisa ser sempre menor ou igual à altura $y$, aplicando uma recompensa negativa em situações contrárias. Este modelo é eficiente na ideia de controle e redução da velocidade mas não resolve o problema de pouso si só, pois em vários casos, o agente aprende a descer controladamente mas perde muito tempo pairando perto da coordenada $y = 0$, necessitando de outras recompensas como complemento, sendo mais aplicável em testes com currículos.

### 4.5.3 Modelo 5: Controle de inclinação

Também foram implementadas ideias de controle de inclinação da nave, porém, todas elas sofrem do problema de explicitar um valor ideal ou restringir muito as manobras da nave, originando em mais tempo de manobra utilizando as ignições, resultando em mais recompensas negativas.

### 4.5.4 Combinações de modelos

Por fim, foram testadas combinações de modelos de recompensas para serem executados de uma vez, no lugar de separá-los sequencialmente em tarefas de currículos. Uma das combinações que a priori parecia ser mais promissora é do modelo 1 com o modelo 5, pois um dos problemas iniciais do modelo 1 é que em determinados casos, a manobra feita para centralizar a nave, acaba por inclina-lá demais, fazendo o agente perder o controle. Porém, esta combinação também não se mostrou efetiva.

Outras combinações foram testadas, principalmente dos outros modelos com o modelo 3. Porém, isto só aumentou ainda mais a complexidade do *problema de atribuição de crédito* já presente no modelo 3.

# 5 Experimentos

Este capítulo se destina a descrever como foram executados os experimentos deste trabalho, tendo uma breve descrição das configurações do ambiente de desenvolvimento, os parâmetros do algoritmo utilizado, as dificuldades e os resultados exibidos mais detalhadamente.

## 5.1 Configurações do ambiente

A máquina utilizada nestes experimentos foi um notebook Asus X556U, com processador core i7-7500u e 8gb de memória ram.

O ambiente utilizado foi o Lunar Lander v2 discreto. O algoritmo de aprendizagem utilizado foi o PPO2 da *Stable baselines* na versão 2.10.2, com a política *MlpPolicy*, que consiste de uma arquitetura *ator-crítico* de duas redes perceptron multi camada. As configurações da *MlpPolicy* foram mantidas em seu padrão, utilizando assim duas redes de duas camadas de 64 nós, uma para a política (o ator) e outra para a função de valor (o crítico).

Os parâmetros de configuração do PPO2 utilizados neste trabalho foram, em sua maioria, os valores definidos por padrão da *stable baselines*, descritos na tabela 1.

| **Parâmetro** | **Valor** |
|---|---|
| Fator de desconto gama | 0.99 |
| Coefiente de entropia | 0.01 |
| Taxa de aprendizado | 0.00025 |
| Coeficiente da função de valor | 0.5 |
| Valor maximo para o *clip* do gradiente | 0.5 |
| Valor de *clip* da política | 0.2 |
| Quantidade de Threads de execução | 4 |

Tabela 1 – Parâmetros de configuração do PPO2 utilizados nos experimentos deste trabalho.

Os valores dos fatores de multiplicação dos modelos foram otimizados utilizando o framework *Optuna*. Os gráficos dos resultados deste trabalho foram construídos utilizando a biblioteca *Plotly*.

## 5.2 Organização dos experimentos

O passo inicial deste trabalho consistiu em idealizar os modelos. Depois do primeiro momento de entendimento do código, os wrappers foram implementados e testados. Foi construído um wrapper básico de qual todos os outros herdam, cuja função é registrar os *logs* de cada execução. Para além dos *wrappers* construidos por cada modelo, como descrito no capítulo 4 deste trabalho, foi desenvolvido um *wrapper* identidade, que serviu para registrar os *logs* do Lunar Lander padrão, sem recompensas modeladas.

O Lunar Lander é um ambiente relativamente simples, portanto um treinamento de 1 milhão de passos durou por volta de 10 minutos. Os gráficos de todos os resultados descritos neste trabalho são uma média de 5 execuções, incluindo uma barra de erro com o desvio padrão.

## 5.3 Resultados

Inicialmente foi executado o Lunar Lander com o sistema de recompensas padrão, a fim de estimar em quantos passos o agente consegue aprender a pousar completamente.

A execução dos modelos foi limitada empiricamente pela quantidade de passos necessários para que o algoritmo convergisse. Portanto, o modelo padrão foi executado com 5 Milhões de passos, ainda sim, não foi o suficiente para o algoritmo atingir uma média acima dos 200 pontos de recompensa, como mostrado na figura 6.

Os modelos 1 e 2 também foram executados com 5 milhões de passos com o objetivo de compará-los com o modelo de recompensas padrão, como mostrado nas figuras 7 e 8 respectivamente. O modelo 1 obteve o melhor resultado ao ser executado com o valor de $a$, descrito na equação 4.1, sendo $a = 1000$. O modelo 2 obteve o melhor resultado ao ser executado com 5 marcos, atribuindo +10 pontos por cada marco alcançado.

O modelo 1 mostrou-se mais eficiente, pois a partir dos 2.5 milhões de passos, a média de recompensas se concentrou acima da linha dos 200 pontos, o que segundo a *OpenAI*, significa que o agente a prendeu a pousar corretamente.

O modelo 2, mesmo sendo mais eficiente em relação ao modelo padrão, não conseguiu atingir um resultado satisfatório como o modelo 1, pois a média de recompensas em nenhum momento cruzou ou superou a marca de 200 pontos. Porém, ao observar o início da curva, temos que nos primeiros 1.5 milhões de passos o modelo 2 foi mais eficiente do que os modelos 1 e o padrão. Portanto, foi observada a possibilidade de separar o aprendizado do agente em tarefas de currículos de forma que o modelo 2

pudesse acelerar ainda mais o aprendizado do modelo 1.

Como descrito na seção 3.2 deste trabalho, é um desafio definir manualmente a quantidade de passos ideal em cada currículo. Logo, foram feitos testes empíricos utilizando tamanhos de currículo seguindo a ideia trazida nos resultados individuais dos modelos.

Foi constatado empiricamente que nos cenários onde o modelo 1 é posto como a primeira tarefa do currículo, e o modelo 2 é posto em sequência (currículo 1-2) o agente aprende unicamente a centralizar a nave e, em seguida, a descer, não alterando muito o tempo de aprendizado do próprio modelo 1.

O modelo de currículo 2-1 foi testado, otimizado e teve alterações nos parâmetros de cada modelo, sendo realizados testes com quantidades diferentes de passos, com o objetivo de encontrar o currículo ideal. Porém, nas execuções onde o modelo 2 é posto como a primeira tarefa, o agente, logo de início, recebe incentivos a mais para descer, sendo condicionado a receber os 100-140 pontos de recompensas padrão atribuídos naturalmente por descer a nave, complicando o caminho de treinamento quando passado para o modelo 1, necessitando de uma otimização nos parâmetros. A melhor configuração encontrada neste trabalho foi a execução de 1 milhão de passos do modelo 2, com os mesmos valores utilizados na execução individual (5 marcos atribuindo +10 de recompensa por cada marco alcançado); e em sequência a execução do modelo 1 com o valor de $a = 500$. O resultado da execução é mostrado na figura 10.

Após os treinamentos, os modelos foram executados 100 vezes cada, a fim de comparar o desempenho do agente treinado, como pode ser visto na figura 11 e na tabela 2.

A tabela 2 exibe os resultados mais detalhados das execuções dos modelos, acrescentando a *porcentagem de pouso* calculada considerando a indicação do *GYM* de que um episódio com 200 pontos ou mais de recompensa, é considerado um pouso. A proporcionalidade de resultados manteve-se condizente com os gráficos obtidos no treinamento. O modelo 1 não é 100% efetivo, onde em 84% das vezes, o agente treinado pousa corretamente na área de pouso, mas é o suficiente para aprender mais rápido a tarefa em relação ao modelo padrão, como mostrado no gráfico 7. Os modelos padrão e 2 obtiveram um resultado muito similar, o que pode ser interpretado pelo fato de que o modelo 2 é, de certo ponto, uma redundância do modelo padrão, que busca unicamente acelerar a descida do agente.

O currículo 2-1 não obteve um resultado satisfatório, sendo superior ao modelo padrão, porém inferior ao modelo 1. A razão disto pode ser explicada no fato de que, ao treinar a primeira tarefa no modelo 2, a política obtém um viés de descer a nave e ao treinar a segunda tarefa no modelo 1, o agente é induzido a ir para o centro em uma

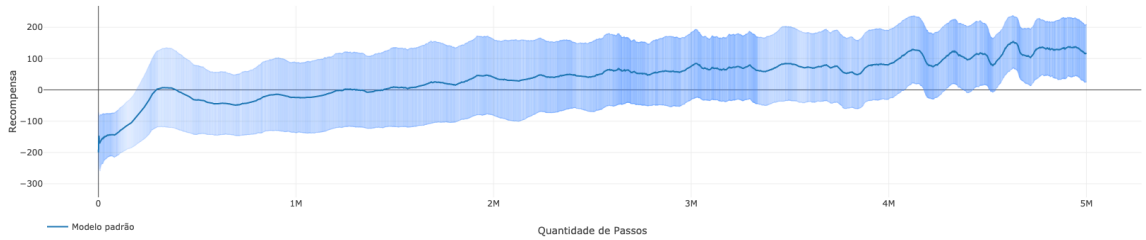


Figura 6 – Gráfico da média de 5 treinamentos de 5 milhões de passos utilizando o modelo padrão de recompensas do Lunar Lander. Elaborado pelo autor (2021).

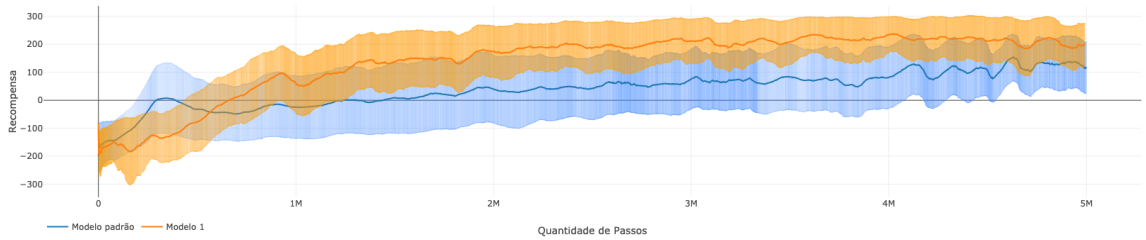


Figura 7 – Gráfico da média de recompensas de 5 treinamentos de 5 milhões de passos utilizando o modelo 1 (laranja) em comparação ao modelo padrão (azul). Elaborado pelo autor (2021).

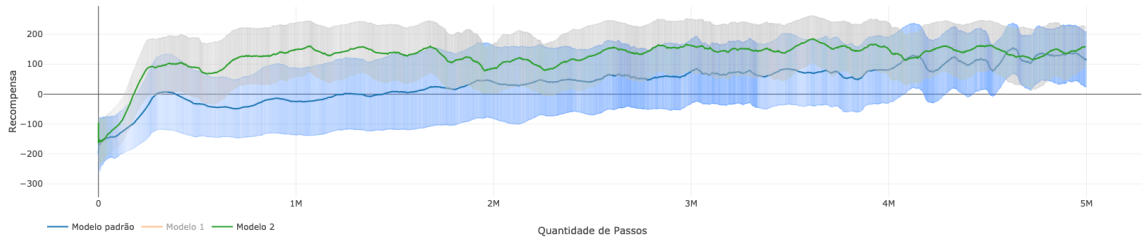


Figura 8 – Gráfico da média de recompensas de 5 treinamentos de 5 milhões de passos utilizando o modelo 2 (verde) em comparação ao modelo padrão (azul). Elaborado pelo autor (2021).

alta velocidade vertical, ocasionando em alguns episódios onde a nave é destruída. Este comportamento ocorreu mesmo com a otimização dos parâmetros dos modelos no currículo.

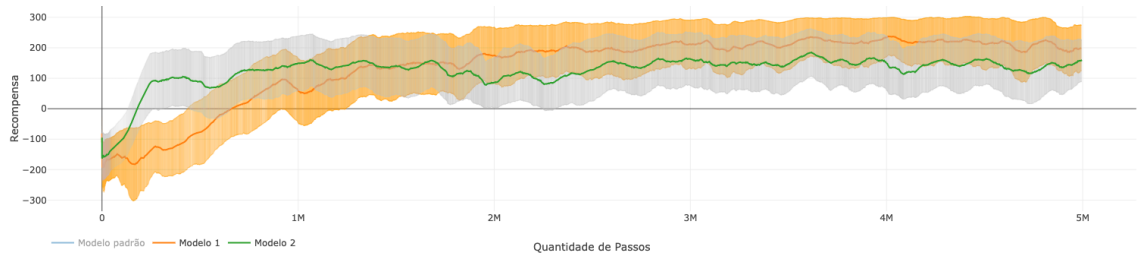


Figura 9 – Gráfico da média de recompensas de 5 treinamentos de 5 milhões de passos utilizando o modelo 1 (laranja) em comparação ao modelo 2 (verde). Elaborado pelo autor (2021).

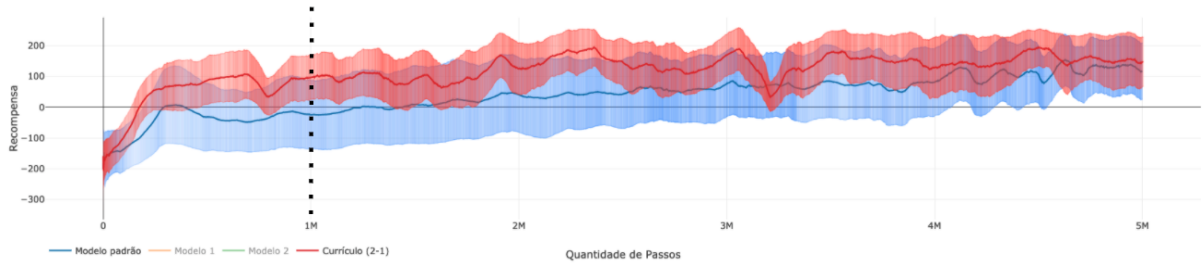


Figura 10 – Gráfico da média de recompensas de 5 treinamentos de 5 milhões de passos utilizando o currículo 2-1 (vermelho) em comparação ao modelo padrão (azul). Elaborado pelo autor (2021).

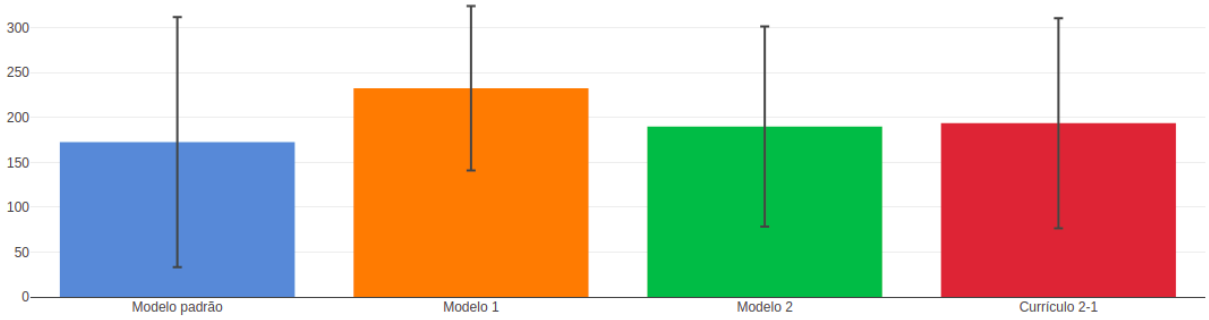


Figura 11 – Gráfico da média de recompensas da execução de 100 passos dos modelos treinados. Elaborado pelo autor (2021).

| Modelo | Recompensa média | Desvio padrão | Porcentagem de pouso |
|---|---|---|---|
| Modelo padrão | 174.47 | 139.31 | 63% |
| Modelo 1 | 232.51 | 91.63 | 84% |
| Modelo 2 | 189.88 | 111.57 | 68% |
| Currículo 2-1 | 193.58 | 116.78 | 69% |

Tabela 2 – Resultados de 100 execuções dos modelos treinados.

# 6 Conclusão

Neste trabalho foram combinadas duas técnicas definidas na literatura, a modelagem de recompensas e a aprendizagem por currículo, com o objetivo de acelerar a aprendizagem no ambiente Lunar Lander.

A modelagem de recompensas, que consiste em adicionar recompensas menores e mais constantes ao agente a fim de guiá-lo com mais eficiência para o seu objetivo, foi utilizada para implementar e testar novos modelos baseando-se em ideias diferentes, onde neste trabalho foram apresentados os dois mais eficientes na diminuição do tempo de aprendizagem. A aprendizagem por currículo, que consiste em dividir o treinamento do agente em tarefas mais simples e tornar o aprendizado mais gradativo, foi aplicada ao definir uma sequência de treinamento dos modelos de recompensa também definidos neste trabalho, a fim de minimizar ainda mais o tempo de aprendizagem.

Após os resultados dos experimentos, ficou evidente que a modelagem de recompensas foi eficiente na resolução do problema, pois o modelo 1 de recompensas parciais, a partir dos 2.5 milhões de execuções, conseguiu uma média de recompensas acima dos 200 pontos, significando um aprendizado da tarefa. Por fim, foram elaborados e executados, currículos utilizando os dois modelos que apresentaram o melhor resultado individual, porém os resultados não foram superiores à execução dos modelos individualmente.

## 6.1 Dificuldades e limitações

Modelagem de recompensas é uma estratégia manual e muitas vezes atrelada à modelagem do problema original. Portanto, entender e modelar uma estratégia efetiva se torna algo não muito trivial. Neste trabalho foram propostos muitos modelos com estratégias diferentes, como descrito no capítulo 4. Algumas estratégias foram efetivas em tentativas iniciais, outras foram descartadas em um outro momento por não serem efetivas, mas reavaliadas em situações posteriores. Os modelos propostos possuem fatores de multiplicação que precisaram ser ajustados em vários testes manuais ou utilizando o *framework Optuna*, demandando um tempo considerável de execução.

A aprendizagem por currículo segue a mesma complexidade manual da modelagem de recompensas. Ao propor os currículos, foi perceptível que a política tomava um viés ao treinar com um modelo, ou seja, assimilava características que eram ótimas para aquele sistema de recompensas específico que não eram realmente necessárias para pousar a nave, o que na maioria das vezes, implicou em dificuldades para treinar

a mesma política na tarefa seguinte, tornando necessária a reparametrização. Ou seja, além de encontrar o tamanho e a sequência ideal das tarefas no currículo, foi necessário reavaliar os parâmetros dos modelos, aumentando a complexidade de entrelaçamento de parâmetros, tomando ainda mais tempo de execução.

Tendo em vista a complexidade de manipular tantos parâmetros, este trabalho foi limitado a parametrizar apenas as variáveis dos modelos, não entrando no escopo de alterar a arquitetura da rede da política ou os parâmetros do *PPO*.

## 6.2 Trabalhos futuros

Este trabalho foi desenvolvido utilizando o algoritmo PPO da família *Policy Gradient* na sua forma discreta. Portanto, outros experimentos podem ser realizados utilizando o PPO contínuo ou mesmo aplicando outros algoritmos, tais como o *Soft Actor-Critic* (SAC), que se mostrou bastante eficiente na resolução de problemas em ambientes da *OpenAI*, como mostrado em (HAARNOJA et al., 2018).

Neste trabalho foi utilizada a técnica de aprendizagem por currículo, que possui a limitação de escolha das tarefas como um processo manual, baseando-se unicamente em conhecimentos empíricos da pessoa que o propôs. Uma abordagem alternativa é utilização do *teacher-student*, uma técnica que automatiza este processo de escolha de tarefas a serem aprendidas (ZIMMER; VIAPPIANI; WENG, 2014). Um trabalho paralelo utilizando esta estratégia foi desenvolvido como trabalho de conclusão de curso pelo aluno *Kenedy Felipe dos Santos da Silva* (BCC-UFRPE).

Neste trabalho foram propostos vários modelos, mas foram avaliados apenas os que apresentaram melhores resultados. Porém, outros modelos de recompensas também foram descritos no capítulo 4 deste trabalho e podem ser utilizados em experimentos com a abordagem de *teacher-student*.

Por fim, possíveis trabalhos podem derivar deste ao mudar os parâmetros de execução, como por exemplo a política, a quantidade de nós e camadas da rede da política ou mesmo alterar os parâmetros do próprio algoritmo, como por exemplo a taxa de aprendizado.

# Referências


AREL, I. et al. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, IET, v. 4, n. 2, p. 128–135, 2010. Citado na página 12.

ARULKUMARAN, K. et al. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, IEEE, v. 34, n. 6, p. 26–38, 2017. Citado 4 vezes nas páginas 15, 16, 17 e 18.

AYOUB, A. et al. Model-based reinforcement learning with value-targeted regression. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2020. p. 463–474. Citado na página 12.

BENGIO, Y. et al. Curriculum learning. In: *Proceedings of the 26th annual international conference on machine learning*. [S.l.: s.n.], 2009. p. 41–48. Citado na página 23.

BROCKMAN, G. et al. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. Citado na página 24.

BRYS, T. et al. Policy transfer using reward shaping. In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2015. (AAMAS '15), p. 181–188. ISBN 978-1-4503-3413-6. Disponível em: <http://dl.acm.org/citation.cfm?id=2772879.2772905>. Citado 2 vezes nas páginas 13 e 18.

GRZES, M. Reward shaping in episodic reinforcement learning. ACM, 2017. Citado na página 22.

HAARNOJA, T. et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: PMLR. *International conference on machine learning*. [S.l.], 2018. p. 1861–1870. Citado na página 37.

HUANG, Q.; KADOCH, M. 5g resource scheduling for low-latency communication: A reinforcement learning approach. In: *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*. [S.l.: s.n.], 2020. p. 1–5. Citado na página 13.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, v. 4, p. 237–285, 1996. Citado na página 17.

LAUD, A. D. *Theory and Application of Reward Shaping in Reinforcement Learning*. Tese (Doutorado), Champaign, IL, USA, 2004. AAI3130966. Citado na página 12.

LAZARIC, A.; RESTELLI, M.; BONARINI, A. Transfer of samples in batch reinforcement learning. In: *Proceedings of the 25th international conference on Machine learning*. [S.l.: s.n.], 2008. p. 544–551. Citado na página 14.

LIANG, W. et al. Energy efficient transmission in underlay cr-noma networks enabled by reinforcement learning. *China Communications*, v. 17, n. 12, p. 66–79, 2020. Citado na página 13.

LIU, J.; DOLAN, P.; PEDERSEN, E. R. Personalized news recommendation based on click behavior. In: ACM. *Proceedings of the 15th international conference on Intelligent user interfaces*. [S.l.], 2010. p. 31–40. Citado na página 12.

MANNION, P. et al. Policy invariance under reward transformations for multi-objective reinforcement learning. *Neurocomputing*, Elsevier, v. 263, p. 60–73, 2017. Citado na página 22.

MAROM, O.; ROSMAN, B. Belief reward shaping in reinforcement learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2018. v. 32, n. 1. Citado na página 22.

Matiisen, T. et al. Teacher-student curriculum learning. *IEEE Transactions on Neural Networks and Learning Systems*, p. 1–9, 2019. Citado 2 vezes nas páginas 13 e 18.

MNIH, V. et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. Citado na página 12.

MNIH, V. et al. Human-level control through deep reinforcement learning. *nature*, Nature Publishing Group, v. 518, n. 7540, p. 529–533, 2015. Citado na página 22.

NARVEKAR, S. Curriculum learning in reinforcement learning. In: *IJCAI*. [S.l.: s.n.], 2017. p. 5195–5196. Citado 2 vezes nas páginas 14 e 23.

NARVEKAR, S.; SINAPOV, J.; STONE, P. Autonomous task sequencing for customized curriculum design in reinforcement learning. In: *IJCAI*. [S.l.: s.n.], 2017. p. 2536–2542. Citado na página 14.

NUGROHO, L. Powered landing guidance algorithms using reinforcement learning methods for lunar lander case. *Jurnal Teknologi Dirgantara*, v. 19, n. 1, p. 43–56, 2021. Citado na página 14.

RANDLØV, J.; ALSTRØM, P. Learning to drive a bicycle using reinforcement learning and shaping. In: *ICML*. [S.l.: s.n.], 1998. v. 98, p. 463–471. Citado 2 vezes nas páginas 13 e 22.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. Disponível em: <http://incompleteideas.net/book/the-book-2nd.html>. Citado 4 vezes nas páginas 12, 15, 16 e 17.

WIERING, M.; OTTERLO, M. V. Reinforcement learning. *Adaptation, learning, and optimization*, Springer, v. 12, n. 3, 2012. Citado 2 vezes nas páginas 17 e 18.

XING, E.; CAI, B. Delivery route optimization based on deep reinforcement learning. In: *2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*. [S.l.: s.n.], 2020. p. 334–338. Citado na página 12.

YU, C.; LIU, J.; NEMATI, S. Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796*, 2019. Citado na página 16.

ZHANG, Y.; DONG, Y. Single image dehazing via reinforcement learning. In: *2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA)*. [S.l.: s.n.], 2020. v. 1, p. 123–126. Citado na página 12.

Zheng, G. et al. Learning to simulate vehicle trajectories from demonstrations. In: *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. [S.l.: s.n.], 2020. p. 1822–1825. Citado na página 12.

ZHOU, Z.; LI, X.; ZARE, R. N. Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, ACS Publications, v. 3, n. 12, p. 1337–1344, 2017. Citado na página 12.

ZIMMER, M.; VIAPPIANI, P.; WENG, P. Teacher-student framework: a reinforcement learning approach. In: *AAMAS Workshop Autonomous Robots and Multirobot Systems*. [S.l.: s.n.], 2014. Citado na página 37.