Insper

Instituto de Ensino e Pesquisa

Curso de Engenharia de Computação

Programa Institucional de Bolsas de Iniciação Científica (PIBIC)

RELATÓRIO FINAL

ESTUDO COMPARATIVO DE MODELOS DE OTIMIZAÇÃO DE AGENTES AUTÔNOMOS BASEADOS EM APRENDIZADO POR REFORÇO

JOÃO GABRIEL VALENTIM ROCHA

ORIENTADOR: Prof. Dr. Fábio José Ayres

São Paulo

2022, Fevereiro

João Gabriel Valentim Rocha

RELATÓRIO FINAL

Estudo comparativo de modelos de otimização de agentes autônomos baseados em aprendizado por reforço

Relatório de Projeto de Iniciação Científica para cumprir com os requisitos estabelecidos no Edital do Programa Institucional de Bolsas de Iniciação Científica do Insper.

São Paulo

2022, Fevereiro

RESUMO

O estudo comparativo de modelos de optimização de agente autônomos tem influência direta nas circunstâncias tecnológicas atuais. O crescimento exponencial do número de aplicações da área de Reinforcement Learning tem trazido a necessidade de estudos comparativos, haja vista a importância de selecionar e utilizar algoritmos cada vez mais atuais, mais rápidos e mais eficientes para aquela tarefa. Logo, nosso objetivo é trazer um estudo detalhado a respeito do desempenho, sob as métricas adequadas, que os principais algoritmos de optimização de agentes autônomos têm e compará-los. Nesse sentido, utilizamos um ambiente de simulação para entender cada algoritmo na essência enquanto fazemos cada estudo comparativo. Sob essa perspectiva, é de suma importância tomar como base os estudos mais recentes de algoritmos de optimização que são utilizados hoje. A utilização das métricas para avaliar um algoritmo vai variar de algoritmo para algoritmo, caberá a nós escolher e utilizar as mais adequadas para aquele tipo de processo de aprendizagem do agente. Além disso, alguns métodos podem ser inseridos junto ao algoritmo, com o intuito de melhorar a performance da aprendizagem do agente. Nesse sentido, utilizamos os métodos de Curriculum Learning e Transfer Learning, que consiste na ideia de transferência de aprendizado por meio de um curriculum. Em outras palavras, é possível que um agente possa aprender a realizar uma tarefa complexa pela secção dessa tarefa em tarefas menores que podem transferir o aprendizado de uma tarefa (menos complexa) para a próxima (mais complexa). Dessa forma, é possível que a curva de aprendizagem (ou a de recompensa acumulativa) venha a convergir em um tempo menor e com menos esforço. O processo de estabelecer um comparativo entre modelos de algoritmos de optimização de agente autônomos é, portanto, crucial para o embasamento necessário que permite que o número de aplicações e solução de problemas cresçam cada vez mais, sobretudo, aquelas que envolvem as técnicas de Reinforcement Learning.

Palavras-chaves: Reinforcement Learning. Genetic Algorithms. Neural networks policies. Steering Behaviors. Policy gradients. REINFORCE. Proximal Policy Optimization. Curriculum Learning.

SUMÁRIO

1 INTRODUÇÃO

Grande parte das técnicas utilizadas para traçar a caminhada de robôs ou carros que dirigem sem a necessidade de motorista são pautadas na otimização de agente autônomos. A área do conhecimento denominada "Otimização de agentes autônomos" (em inglês: *Autonomous Agent Optimization*) refere-se ao conjunto de técnicas utilizadas para aprimorar uma atividade feita por um agente autônomo, inserido em um ambiente, por meio de ferramentas computacionais, com o objetivo de buscar a solução ótima de uma tarefa determinística ou não (GéRON, 2019). Exemplos do uso de otimização de agentes autônomos incluem a otimização da caminhada de um robô, a trajetória de carros que dirigem sem a necessidade de um motorista, sistemas que podem aprender a jogar jogos eletrônicos (MNIH et al., 2013), otimização de um termostato para economia de energia e negociações financeiras automáticas (GéRON, 2019). Nesse sentido, o mundo está cheio de situações em que um agente inteligente pode ser útil.

Dentre as formulações do problema de otimização de agentes autônomos temos o "aprendizado por reforço" (*Reinforcement Learning*). Nesta formulação temos que o agente é o responsável por interagir com o ambiente e recebe recompensas para realizar alguma tarefa. O processo de aprendizado objetiva a construção de uma política de ação do agente (*policy*) que maximize o valor esperado da recompensa em uma realização de tarefa. Nesse aspecto, se tomarmos como exemplo um sistema em que um drone precisa fazer uma entrega para uma empresa de *delivery*, o drone seria o agente, o ambiente seria o cenário em que ele se encontra além de seu próprio estado (posição, velocidade, orientação, estado da bateria, entre outros) e a recompensa seria algum escore positivo ao realizar o objetivo de levar a entrega para o endereço designado no menor tempo possível. Dessa forma, é possível visualizar que as aplicações do *Reinforcement Learning* podem ser ampliadas para âmbitos diversos, como o exemplo da entrega do drone e o exemplo de sistemas que aprendem a jogar jogos eletrônicos, citados anteriormente.

Existem diversas técnicas de *reinforcement learning*, e entender suas vantagens e limitações comparativamente, e em relação à características das tarefas a serem executadas, é importante para o efetivo projeto de sistemas autônomos inteligentes. A realização de tarefas quer sejam complexas, quer sejam simples, exigem que exista uma avaliação de como o agente performará para aprender e chegar no seu objetivo com a máxima recompensa. No presente trabalho procura-se estabelecer critérios comparativos para a determinação de qual modelo será mais viável para uma determinada tarefa.

2 REINFORCEMENT LEARNING

Aprendizado por reforço (*Reinforcement Learning*, ou RL) refere-se ao conjunto de técnicas para o treinamento de agentes autônomos através de incentivos que chamamos de recompensa. Um agente pode ser qualquer entidade que se encontra em um determinado ambiente, com o intuito de realizar alguma tarefa. Nesse sentido, o agente interage com o ambiente e isso caracteriza a essência do sistema que é o foco do estudo do *Reinforcement Learning*, tal como visto na Figura 1.

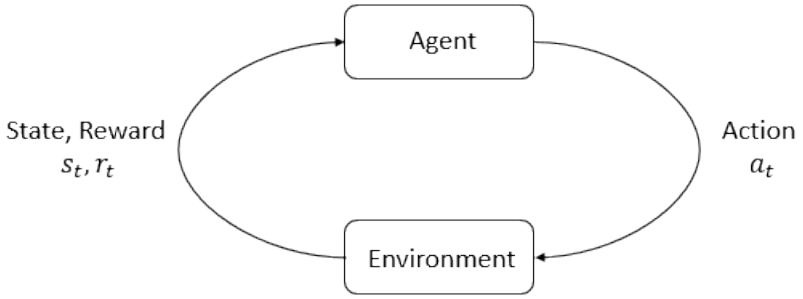

Figura 1 – Interação agente-ambiente.

Na Figura 1 é possível descrever o comportamento e o andamento do processo com a descrição de algumas variáveis que são pertinentes à condição do Agente: a ação a_t a ser tomada no tempo $t \in \{0, 1, 2, \cdots\}$, o estado s_t do tempo t, o próximo estado s_{t+1}, alcançado após a tomada da ação a_t e a recompensa r_t devido a troca de estado.

A exemplo disso, podemos pensar num robô aspirador de pó que traça trajetórias com o intuito de limpar uma sala. Nesse contexto, o agente é o robô e o ambiente é sala. Nosso sistema buscará resolver o problema central do *Reinforcement Learning*, que é maximizar a recompensa acumulada, que nesse caso, é fazer com que o robô limpe a sala no menor tempo possível, traçando trajetórias que são optimizadas para que isso ocorra.

Dessa forma, vamos definir alguns aspectos importantes e necessários para que possamos descrever e entender os sistemas de *Reinforcement Learning* de uma maneira mais consistente e estruturada, sobretudo, fazendo uso das ferramentas matemáticas para descrever como funcionam alguns conceitos e abordagens.

2.1 *Policy*

A *policy* é a regra utilizada pelo agente para decidir quais ações tomar. Uma *policy* pode ser determinística, na qual as ações são uma função do estado do ambiente, ou estocástica, onde as ações são escolhidas a partir de uma distribuição estocástica condicionada ao estado

do ambiente. Neste trabalho vamos estudar *policies* estocásticas. As *policies* estocásticas são usualmente denotadas por π:

$$a_t \sim \pi(\cdot|s_t). \tag{2.1}$$

A Equação 2.1 indica que a ação a_t a ser tomada pelo agente é obtida como uma amostra aleatória da distribuição condicional $\pi(\cdot|s_t)$. No caso limite em que a distribuição condicional converge para uma distribuição delta de Dirac, a *policy* torna-se determinística.

É comum que a *policy* seja mencionada de forma intercambiável com o agente, pelo fato da policy poder ser interpretada como o cérebro do agente. Comumente lidamos com *policies* parametrizadas, onde denotamos os parâmetros de tal *policy* por θ ou ϕ e, em seguida escrevemos isso como um subscrito no símbolo da *policy* (Equação 2.2):

$$a_t \sim \pi_\theta(\cdot|s_t). \tag{2.2}$$

2.2 Trajetória

Uma trajetória é uma sequência de estados e ações de um agente que atua em um determinado ambiente, para realizar uma determinada tarefa. Lembre-se que utilizamos as notações de s_t e a_t para representar, respectivamente, o estado e a ação no tempo t.

$$\tau = (s_0, a_0, s_1, a_1, ...) \tag{2.3}$$

O primeiro estado é aleatoriamente inserido seguindo uma distribuição de estado inicial denotada por ρ_0:

$$s_0 \sim \rho_0(\cdot).$$

A transição de estado do ambiente de s_t para s_{t+1} depende de modo probabilístico do estado atual do ambiente (s_t) e da ação tomada pelo agente (a_t), conforme ilustrado na Equação 2.4:

$$s_{t+1} \sim P(\cdot|s_t, a_t). \tag{2.4}$$

2.3 Recompensa e Retorno

A função de recompensa $\mathcal{R}$ é criticamente importante para o *Reinforcement Learning*. Isso depende do estado atual (s_t), da ação tomada (a_t), e o próximo estado (s_{t+1}):

$$r_t = \mathcal{R}(s_t, a_t, s_{t+1}). \tag{2.5}$$

Em conceitos mais palpáveis, a função de recompensa nos traz a influência da transição entre os estados (tomada a ação a_t) de forma quantitativa. Sobretudo, isso tem forte relação com a tarefa que deve ser realizada pelo agente, naquele ambiente. Nesse sentido, vale ressaltar que existe alguns tipos de cálculo do retorno.

Dada uma sequência de recompensas $(r_0, r_1, \cdots, r_T)$ em uma trajetória τ, podemos definir o retorno completo $R(\tau)$ sobre a trajetória de diferentes formas, conforme visto abaixo. O objetivo do *Reinforcement Learning* é a construção de uma *policy* que maximize o retorno médio sobre todas as trajetórias.

Um tipo de retorno é o *finite-horizon undiscounted return*, que é a soma das recompensas obtidas (fixas) para cada passo, sobre toda a trajetória:

$$R(\tau) = \sum_{t=0}^{T} r_t. \tag{2.6}$$

Um outro tipo de retorno é o *infinite-horizon discounted return*, que é a soma das recompensas obtidas para cada passo, multiplicado por um fator de desconto $\gamma \in (0, 1)$:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{2.7}$$

Note que $\gamma \in (0, 1)$ é muito conveniente pois, além de garantir a convergência da soma, demonstra que a recompensa maior será atribuída para ação tomada no menor tempo, o que faz sentido para o treinamento do agente.

Por fim, é possível utilizar também o *reward to-go*. Nesse caso, levaremos em conta somente as recompensas acumuladas do instante t em diante, ou seja, até o fim da trajetória.

$$G_t = \sum_{t'=t}^{T} \mathcal{R}(s_{t'}, a_{t'}, s_{t'+1}), \tag{2.8}$$

O que faz todo sentido, haja vista o fato de que a atualização da *policy* deve ser tomada de acordo com a recompensa tomada do instante t atual até o fim da trajetória. Mais adiante vamos utilizar essa função de recompensa para a construção do algoritmo *REINFORCE* na seção 2 e vamos mostrar que a utilização dela pode ser feita no teorema do *policy gradient* sem problemas.

2.4 Função de vantagem

A função de vantagem, denotada simplificadamente por $A(s, a)$, nos permite estimar o quanto uma ação a pode ser vantajosa (ou não) dado um estado s. Para tanto, é de suma importância que exploremos as funções de valor associadas ao *Reinforcement Learning*. As funções de valor servem para atribuir valores em um dado par estado-ação em que o agente se encontrar, sob uma *policy* π. Existem quatro principais funções de valores que serão abordadas aqui:

1. **Função de valor na *policy***, $V^{\pi}(s)$, que nos dá o retorno esperado se o agente começa num estado s e sempre atua de acordo com uma *policy* π:

$$V^{\pi}(s) = E_{\tau\sim\pi}[R(\tau)|s_0 = s] \tag{2.9}$$

2. **Função de ação-valor na *policy***, $Q^{\pi}(s, a)$, que nos dá o retorno esperado se o agente começa num estado s, toma uma ação arbitrária a (que pode ou não seguir a *policy*), e então sempre atua de acordo com a *policy* π:

$$Q^{\pi}(s, a) = E_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a] \tag{2.10}$$

   As definições de função de valor e de ação-valor nas Equações 2.9 e 2.10 estão relacionadas conforme a Equação 2.11:

$$V^{\pi}(s) = E_{a\sim\pi}[Q^{\pi}(s, a)] \tag{2.11}$$

3. **Função de valor ótima**, $V^{*}(s)$, que nos dá o retorno esperado se o agente começa num estado s e então sempre atua de acordo com a *policy* ótima.

$$V^{*}(s) = \max_{\pi} E_{\tau\sim\pi}[R(\tau)|s_0 = s] \tag{2.12}$$

4. **Função de ação-valor ótima**, $Q^{\pi}(s, a)$, que nos dá o retorno esperado se o agente começa num estado s, toma uma ação arbitrária a (que pode ou não seguir a *policy*), e então sempre atua de acordo com a *policy* ótima:

$$Q^{*}(s, a) = \max_{\pi} E_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a] \tag{2.13}$$

Dessa forma, sabendo o comportamento das funções de valores, é possível definir a função de vantagem dada como $A^{\pi}(s, a)$ correspondendo a uma *policy* π, que irá descrever quanto será melhor para o agente tomar uma ação a específica. Matematicamente, podemos definir a função como sendo:

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s). \tag{2.14}$$

Intuitivamente, podemos entender o valor da função de vantagem como sendo o quão bom é o valor da ação a no estado s em relação ao valor médio das ações neste estado. A função de vantagem será utilizada na construção do algoritmo PPO, descrito na seção 3.2.

2.5 O Problema do *Reinforcement Learning*

Para qualquer que seja o retorno medido, a meta do RL é selecionar a *policy* que maximiza o retorno esperado quando o agente atua numa trajetória τ. Para falar sobre retorno esperado, vamos falar sobre a probabilidade de uma trajetória. Vamos supor que os ambientes de transição e as *policies* são estocásticos. Nesse caso, queremos calcular a probabilidade de ocorrência de uma trajetória τ seguindo uma *policy* π, que denotaremos por $P(\tau|\pi)$:

$$P(\tau|\pi) = P(s_0, a_0, s_1, a_1, ...|\pi)$$

A relação das probabilidades condicionais nos diz que:

$$P(A, B) = P(B) \cdot P(A|B) \tag{2.15}$$

Portando, aplicando a relação, temos:

$$P(\tau|\pi) = P(s_0, a_0, s_1, a_1, ...|\pi)$$

$$= P(a_0, s_1, a_1, ...|\pi, s_0) \cdot P(s_0|\pi)$$

$$= P(s_1, a_1, ...|\pi, s_0, a_0) \cdot \underbrace{P(s_0|\pi)}_{\rho_0(s_0)} \cdot \overbrace{P(a_0|\pi, s_0)}^{\pi(a_0|s_0)}$$

$$= \rho_0(s_0) \cdot P(s_1, a_1, ...|\pi, s_0, a_0) \cdot \pi(a_0|s_0)$$

$$= \rho_0(s_0) \cdot P(a_1, s_2, a_2, ...|\pi, s_0, a_0, s_1) \cdot \underbrace{P(s_1|\pi, s_0, a_0)}_{P(s_1|a_0, s_0)} \cdot \pi(a_0|s_0)$$

$$= \rho_0(s_0) \cdot P(s_2, a_2, ...|\pi, s_0, a_0, s_1, a_1) \cdot \overbrace{P(a_1|\pi, s_0, a_0, s_1)}^{\pi(a_1|s_1)} \cdot P(s_1|a_0, s_0) \cdot \pi(a_0|s_0)$$

$$= \rho_0(s_0) \cdot P(s_2, a_2, ...|\pi, s_0, a_0, s_1, a_1) \cdot P(s_1|a_0, s_0) \cdot \pi(a_0|s_0) \cdot \pi(a_1|s_1).$$

$$= \dots$$

Realizando esta operação repetidas vezes, teremos que para o T-ésimo passo a probabilidade da trajetória dado uma policy será:

$$P(\tau|\pi) = \rho_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t), \tag{2.16}$$

Dessa forma, podemos calcular o retorno esperado de um agente ao longo de um conjunto de trajetórias. O retorno esperado denotado por $J(\pi)$ é então:

$$J(\pi) = E_{\tau\sim\pi}[R(\tau)] = \int_{\tau} P(\tau|\pi)R(\tau)d\tau, \tag{2.17}$$

Com base nas equações que encontramos e definimos anteriormente, podemos, portanto, definir o problema central em RL, que pode então ser expresso por:

$$\pi^* = \arg\max_{\pi} J(\pi), \tag{2.18}$$

com π^* sendo a *policy* ótima. Ou seja, a política de tomada de ações que maximiza a recompensa acumulada ao longo de uma trajetória que o agente pode tomar. Se tomarmos como exemplo um sistema que o agente é um drone de entregas, a *policy* ótima π^* vai conduzir o agente a tomar as melhores tomadas de ações para que ele chegue ao destino designado no menor tempo possível e desviando dos possíveis obstáculos do ambiente. Isso se deve, sobretudo, ao treinamento por aprendizado por reforço, que confere estímulos positivos e negativos ao agente, no intuito de tornar o agente mais apto a tomar as decisões necessárias para a tarefa ser feita com o máximo de recompensa acumulada.

3 ALGORITMOS DE POLICY GRADIENT

Para atingir o objetivo central do *Reinforcement Learning*, utilizamos algoritmos de treinamento que podem ser por meio de métodos que utilizam o gradiente da policy (*Policy Gradient*). Nesse sentido, vejamos a taxonomia dos métodos de treinamento de policy na figura 2.

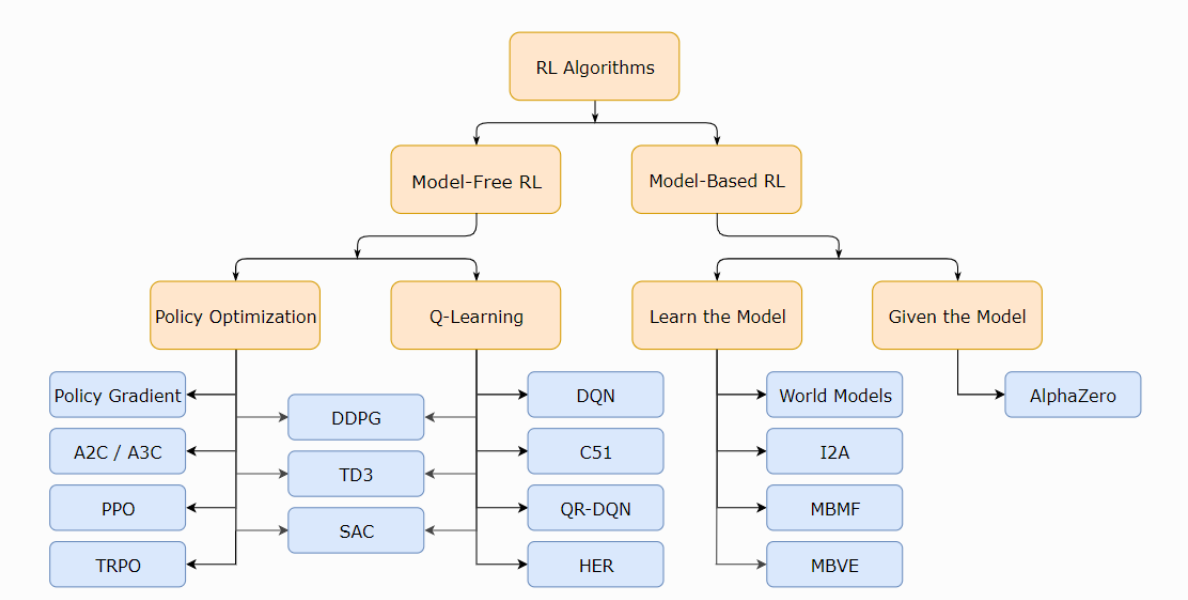

Figura 2 – Taxonomia dos algoritmos de RL.

Dentre os inúmeros métodos e modelos existentes, vamos focar nos modelos livres (*Model-free RL*), em específico nos algoritmos de *Policy Gradient*, *PPO* (*Proximal Policy Opmization*) e *Genetic Algorithm*.

3.1 Derivando um *Policy Gradient* simplista

Como visto anteriormente, queremos maximizar o nosso return esperado que é definido por $J(\pi_\theta) = E_{\tau\sim\pi_\theta}[R(\tau)]$. Iremos optimizar nossa policy através do gradiente ascendente definido por:

$$\theta_{k+1} = \theta_k + \alpha\nabla_\theta J(\pi_{\theta_k}) \tag{3.1}$$

O gradiente da performance da *policy*, $\nabla_\theta J(\pi_\theta)$, é chamado de ***policy gradient***, os algoritmos que optimizam a *policy* são chamados de ***Policy Gradient Algorithms*** (como os citados aqui, *Policy Gradient simplista* e *Proximal Policy Optimization*). Nesse caso, para chegarmos no valor do *policy gradient*, vamos seguir alguns passos algébricos que vão nos direcionar para uma expressão que pode ser computada.

1. **Probabilidade de uma trajetória**. Dada uma trajetória $\tau = (s_0, a_0, ..., s_{T+1})$ e tomadas ações que seguem uma *policy* π_θ, teremos que:

$$P(\tau|\theta) = \rho_0(s_0) \prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t). \tag{3.2}$$

2. **Propriedade do log derivativo**. Tomando como base a regra da cadeia, teremos:

$$\nabla_\theta P(\tau|\theta) = P(\tau|\theta)\nabla_\theta \log P(\tau|\theta). \tag{3.3}$$

3. **Log-Prob de uma trajetória**. O log-prob de uma trajetória é:

$$\log P(\tau|\theta) = \log \rho_0(s_0) + \sum_{t=0}^{T} \Big(\log P(s_{t+1}|s_t, a_t) + \log \pi_\theta(a_t|s_t) \Big). \tag{3.4}$$

4. **Gradiente de funções de ambiente**: Como estamos derivando para variáveis de θ, funções de ambiente (que não dependem de θ) têm gradiente nulo.

5. **Grad-Log-Prob da trajetória**: O gradiente do log-prob de uma trajetória é:

$$\begin{aligned} \nabla_\theta \log P(\tau|\theta) &= \underbrace{\nabla_\theta \log \rho_0(s_0)}_{=0} + \sum_{t=0}^{T} \Big(\underbrace{\nabla_\theta \log P(s_{t+1}|s_t, a_t)}_{=0} + \nabla_\theta \log \pi_\theta(a_t|s_t) \Big) \\ &= \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t). \end{aligned} \tag{3.5}$$

Utilizando as construções acima, podemos concluir que:

$$\begin{aligned} \nabla_\theta J(\pi_\theta) &= \nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] \\ &= \nabla_\theta \int_\tau P(\tau|\theta)R(\tau)d\tau && \text{Expande esperança} \\ &= \int_\tau \nabla_\theta P(\tau|\theta)R(\tau)d\tau && \text{Insere o gradiente na integral} \\ &= \int_\tau P(\tau|\theta)\nabla_\theta \log P(\tau|\theta)R(\tau)d\tau && \text{Log-derivativo} \\ &= E_{\tau\sim\pi_\theta}[\nabla_\theta \log P(\tau|\theta)R(\tau)] && \text{Esperança do retorno} \\ \nabla_\theta J(\pi_\theta) &= E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau)] && \text{Expressão para o grad-log-prob} \end{aligned}$$

O valor encontrado é uma esperança, portanto, podemos estimar esse valor. Se coletarmos um conjunto de trajetórias $\mathcal{D} = \{\tau_i\}_{i=1,...,N}$, onde cada trajetória é feita pelo agente seguindo uma *policy* π_θ, o *policy gradient* pode ser estimado com:

$$\hat{g} = \frac{1}{|\mathcal{D}|} \sum_{\tau\in\mathcal{D}} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau), \tag{3.6}$$

Onde $|\mathcal{D}|$ é o número de trajetórias no conjunto $\mathcal{D}$ (aqui, N). Essa última expressão é a versão mais simples computável que desejamos. Assumindo que tenhamos representado nossa *policy* de maneira que podemos calcular $\nabla_\theta \log \pi_\theta(a|s)$, e se somos capazes de executar a *policy* em um ambiente para coletar um conjunto de dados de trajetórias, nós podemos computar o *policy gradient* e tomar a atualização dos passos (*update step*).

3.1.1 Lemma do *Grad-log-prob*

Vamos provar um lema que será de fundamental importância para construirmos o algoritmo *REINFORCE* e para mostrar a redução da variância por meio da causalidade. Inicialmente, dada uma *policy* parametrizada π_θ, teremos que, por definição (distribuições de probabilidade são normalizadas):

$$\int_x \pi_\theta(x)dx = 1.$$

Tomando o gradiente dos dois lados:

$$\nabla_\theta \int_x \pi_\theta(x)dx = \nabla_\theta 1 = 0.$$

Utilizando o "truque" do log derivativo citado na expressão 3.3:

$$\begin{aligned} 0 &= \nabla_\theta \int_x \pi_\theta(x)dx \\ &= \int_x \nabla_\theta \pi_\theta(x)dx \\ &= \int_x \pi_\theta(x)\nabla_\theta \log \pi_\theta(x)dx \\ &= E_{x\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(x)]. \end{aligned}$$

Logo, teremos que:

$$E_{x\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(x)] = 0. \tag{3.7}$$

3.1.2 Redução da Variância

É de suma importância que a variância do modelo procure ser minimizada, para tanto, utilizaremos a expressão 3.6 com uma nova abordagem. Faz sentido que nós utilizemos uma recompensa mais parecida que com a citada em 2.8. Isso se deve, sobretudo, ao fato de que o passado não influência o futuro em nosso modelo. Para termos melhor precisão em qual tomada de ação tomar para maximizar a recompensa ao fim da trajetória. Portanto, faz sentido utilizar a seguinte expressão:

$$\nabla_\theta J(\pi_\theta) = E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)G_t(\tau)]$$

Para provar que podemos fazer essa substituição, vamos desenvolver um pouco de álgebra a partir da expressão conhecemos citada em 3.6.

Temos que a expressão original é dada por:

$$\nabla_\theta J(\pi_\theta) = E_{\tau\sim\pi_\theta}[(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t))(\sum_{t=0}^{T} r_t(s_t, a_t))]$$

Desenvolvendo a expressão, teremos que:

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= E_{\tau\sim\pi_\theta}[(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t))(\sum_{t=0}^{T} r(s_t, a_t))] \\
&= \sum_{t=0}^{T} E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)(\sum_{t'=0}^{T} r(s_{t'}, a_{t'}))] \\
&= \sum_{t=0}^{T} E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)(\sum_{t'=0}^{t-1} r(s_{t'}, a_{t'})) + \nabla_\theta \log \pi_\theta(a_t|s_t)(\sum_{t'=t}^{T} r(s_{t'}, a_{t'}))] \\
&= \sum_{t=0}^{T} (E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)(\sum_{t'=0}^{t-1} r(s_{t'}, a_{t'}))] + E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)(\sum_{t'=t}^{T} r(s_{t'}, a_{t'}))])
\end{aligned}$$

Note que, a ação que o agente toma no instante t só depende do estado mais recente. Portanto, podemos considerar que as variáveis aleatórias do primeiro termo da soma das esperanças sejam independentes, portanto, podemos utilizar que:

$$\nabla_\theta J(\pi_\theta) = \sum_{t=0}^{T} (E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)(\sum_{t'=0}^{t-1} r(s_{t'}, a_{t'}))] + E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)(\sum_{t'=t}^{T} r(s_{t'}, a_{t'}))])$$

$$= \sum_{t=0}^{T} (E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)] \cdot E_{\tau\sim\pi_\theta}[\sum_{t'=0}^{t-1} r(s_{t'}, a_{t'})] + E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)(\sum_{t'=t}^{T} r(s_{t'}, a_{t'}))])$$

Porém, é possível utilizar o lema citado em 3.1.1, e zerar o primeiro termo da expressão dentro do somatório, daí teremos que:

$$\nabla_\theta J(\pi_\theta) = \sum_{t=0}^{T} (E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)(\sum_{t'=t}^{T} r(s_{t'}, a_{t'}))])$$

Por fim, com mais um pouco de álgebra, teremos que a expressão para o gradiente da recompensa será dada pela seguinte expressão:

$$\nabla_\theta J(\pi_\theta) = E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t] \tag{3.8}$$

Portanto, podemos concluir que, o efeito da causalidade promove a redução da variância: A ação da *policy* tomada no tempo t' não afeta a recompensa da *policy* no tempo t quando $t < t'$.

3.1.2.1 Introdução de uma *Baseline* para redução da variância

Tomando ainda como base a seção 3.1.2, é de suma importância que busquemos cada vez mais amenizar os problemas referentes à variância. Isso se deve, sobretudo, aos aspectos associados à instabilidade do treinamento e à destrução da *policy*. Tendo isso em vista, vamos introduzir uma *baseline* na nossa expressão do *grad-log-prob* citada em 3.8. Acompanhe a expressão 3.9

$$\nabla_\theta J(\pi_\theta) = E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t))] \tag{3.9}$$

Em primeiro lugar, vamos provar que a expressão 3.9 é válida. Tendo como base o lemma que provamos em 3.1.1, é possível mostrar o corolário expresso 3.10.

$$E_{a_t\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)] = 0. \tag{3.10}$$

Agora, desenvolvendo a nossa suposição citada em 3.9, vamos obter o seguinte:

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - b(s_t))] \\
&= E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)G_t - \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t))] \\
&= E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)G_t] - E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)] \\
&= \sum_{t=0}^{T}(E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)G_t] - E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)]) \qquad \text{Usando 3.10} \\
&= \sum_{t=0}^{T}(E_{\tau\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)G_t]) \\
&= E_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)G_t])
\end{aligned}$$

Portanto, demonstramos que o fator $b(s_t)$ não altera o valor da esperança, conservando o valor do gradiente, porém, diminuindo a variância. Agora, vamos determinar o valor de $b(s_t)$ para minimização da variância. Queremos o valor mínimo, para tanto, vamos derivar a variância em relação a b e igualar a zero. Acompanhe os cálculos abaixo para uma trajetória completa do agente (vamos simplificar um pouco a notação e escrever o *grad-log-prob* para uma trajetória completa):

$$\begin{aligned} Var(\nabla_\theta J(\pi_\theta)) &= E[\nabla_\theta J(\pi_\theta)^2] - E[\nabla_\theta J(\pi_\theta)]^2 \\ &= E[\nabla_\theta \log \pi_\theta(\tau)(G_\tau - b)^2] - E[\nabla_\theta \log \pi_\theta(\tau)(G_\tau - b)]^2 \quad \text{usando 3.10} \\ &= E[\nabla_\theta \log \pi_\theta(\tau)(G_\tau - b)^2] - E[\nabla_\theta \log \pi_\theta(\tau) G_\tau]^2 \end{aligned}$$

Agora vamos derivar a expressão que encontramos, por simplificação, denote $g(\tau) = \nabla_\theta \log \pi_\theta(\tau)$:

$$\begin{aligned} \frac{dVar}{db} &= \frac{d(E[(g(\tau)(G_\tau - b))^2] - E[g(\tau)G_\tau]^2)}{db} \\ &= \frac{d(E[(g(\tau)(G_\tau - b))^2])}{db} \\ &= \frac{d(E[g(\tau)^2 G_\tau^2 - 2g(\tau)^2 G_\tau b + g(\tau)^2 b^2])}{db} \\ &= -2E[g(\tau)^2 G_\tau] + 2bE[g(\tau)^2] \\ &= 0 \end{aligned}$$

Por fim, isolando o valor de b, teremos que:

$$b = \frac{E[g(\tau)^2 G_\tau]}{E[g(\tau)^2]} \tag{3.11}$$

Note que nesse caso o b assume o valor de G com fatores determinando os pesos. Nesse sentido, podemos utilizar sem problemas o valor de b como sendo a média de recompensa ao longo da trajetória. Existem outras funções que podemos utilizar como uma *baseline* como as citadas em (WU et al., 2018), inclusive algumas delas podem ser as funções de valores que citamos em seções anteriores, porém, vamos trabalhar com essa (recompensa média) que funciona relativamente bem.

3.1.3 *REINFORCE algorithm*

Com os resultados obtidos da expressão 3.6, vamos desenvolver a ideia do algoritmo *REINFORCE* (WILLIAMS, 1992). Tal como na expressão 3.1, nós vamos atualizar os parâmetros pelo método do gradiente ascendente estocástico, vamos usar o teorema do *policy gradient*, ou seja, o termo temporal da expressão 3.6 e vamos utilizar o valor de $G_t(\tau)$ como sendo o valor de retorno *reward to-go*, citado na expressão 2.8. Logo poderemos definir que o passo de atualização dos parâmetros θ será definido pela expressão 3.12.

$$\Delta\theta_t = \alpha \nabla_\theta \log \pi_\theta(a_t|s_t) G_t(\tau) \tag{3.12}$$

Portanto, vamos criar um *loop* para um episódio e um *loop* temporal para cada instante t de uma trajetória do episódio. Portanto, seguiremos o algoritmo 2. Dessa forma, a atualização dos parâmetros será dada em direção à maximização do retorno dado pela *policy* π_θ, tal como na expressão 2.18.

Algorithm 1 REINFORCE

Input: Parâmetros iniciais da policy θ_0.
for cada episodio $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
  for $t = 1$ até $T - 1$ **do**
    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t) G_t(\tau)$
  end for
end for

Além disso, podemos citar também o algoritmo *REINFORCE* utilizando uma baseline (recompensa média).

Algorithm 2 REINFORCE With Baseline

Input: Parâmetros iniciais da policy θ_0.
for cada episodio $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
  for $t = 1$ até $T - 1$ **do**
    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t)(G_t(\tau) - b)$
  end for
end for

3.2 *Proximal Policy Optimization*

Antes de falarmos a principal meta e motivação do algoritmo *PPO* (SCHULMAN et al., 2017), é imprescindível lembrar que, diante dos vários métodos de otimização de *policy*, é comum se utilizar o estimador do gradiente dado por:

$$\hat{g} = \hat{E}_t[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{3.13}$$

Onde π_θ é uma policy estocástica de parâmetro θ e $\hat{A}_t$ é um estimador da função de vantagem no passo t. Aqui, a esperança indica a média empírica sobre um lote finito de amostras, em um algoritmo que alterna entre amostragem e otimização. O estimador $\hat{g}$ é obtido diferenciando o objetivo:

$$L^{PG}(\theta) = \hat{E}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{3.14}$$

É comum que, em primeiro momento, seja pensado em submeter vários passos para uma mesma trajetória, porém isso pode oferecer prejuízos para *policy*. Isso se deve, sobretudo, ao fato de que as funções computadas apresentarem um alto valor de variância.

Nesse sentido, é preciso que o problema da variância seja resolvido ou amortecido. Para tanto, se faz necessário o uso de outras funções computáveis de tal forma que possamos minimizar o prejuízo da *policy*. Dessa forma, vamos construir uma nova função de perda que utilizaremos no lugar da Função 3.14. Essa função trará a ideia de uma *policy* mais conservadora, tornando a atualização para novas *policies* mais restritas e controladas por um hiper-parâmetro. Algumas outras técnicas famosas para redução de variância são introduzidas, tais como a utilização da *generalized advantage estimation* (SCHULMAN PHILIPP MORITZ, 2015) ou *finite-horizon estimators* (MNIH et al., 2016).

3.2.1 Key equations

Nesse sentido, a função objetivo que chamaremos de L será dada pela Equação 3.15.

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a),\ \ \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right)A^{\pi_{\theta_k}}(s,a)\right). \tag{3.15}$$

A atualização da *policy* se dará de forma típica ao tomar múltiplos passos de algum tipo de otimizador como *Stochastic Gradient Descent* (AMARI, 1993) ou *Adam* (KINGMA, 2015) por meio da Equação 3.16.

$$\theta_{k+1} = \arg\max_\theta \underset{s,a\sim\pi_{\theta_k}}{\mathrm{E}}\left[L(s,a,\theta_k,\theta)\right]. \tag{3.16}$$

Note que introduzimos um valor ϵ na equação que desempenhará um papel importante como hiper-parâmetro do modelo. Ele será um valor pequeno que será responsável pelo limite aceitável de divergência entre a *policy* atual e a *policy* anterior. Isso trás um caráter conservador para a *policy*, haja vista o fato de que os novos conjuntos de parâmetros atualizados são muito parecidos com os anteriores.

Vamos abordar um pouco mais a expressão 3.15 e simular o que ocorre em cada situação possível, isto é, para os casos em que $A > 0$ e $A < 0$.

A expressão 3.15 é um tanto complexa e é difícil dizer a primeira vista o que está fazendo ou como ajuda a manter a nova *policy* próxima da anterior. Nesse sentido, vamos destrinchar os aspectos mais importantes de como essa expressão se comporta:

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s,a))\right), \tag{3.17}$$

Onde,

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0. \end{cases} \tag{3.18}$$

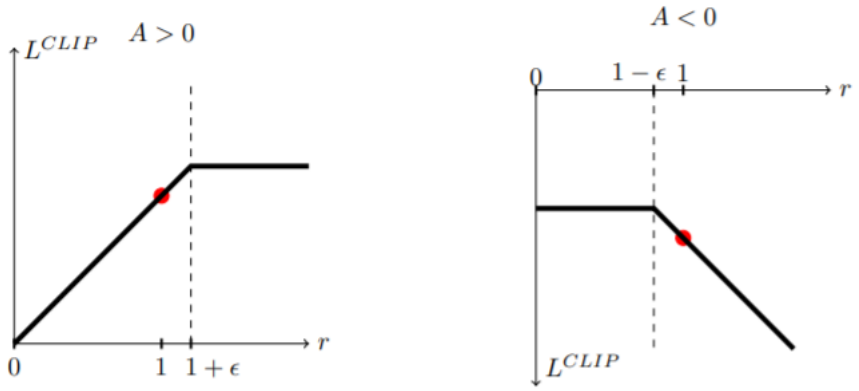

Figura 3 – Nesse caso, temos o valor de L x r (ratio), onde r é definido como $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$. Note que para $\theta = \theta_k$, teremos $r(\theta_k) = 1$, simbolizado pela bola vermelha no gráfico.

Para exemplificar melhor como isso irá funcionar, vamos tomar um par estado-ação (s, a) fixos, e pensar nos casos,

- **Vantagem** (A) **positiva**: Suponha que a vantagem para o par (s, a) é positiva. Caso em que sua contribuição para a função objetivo é:

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1+\epsilon)\right) A^{\pi_{\theta_k}}(s, a). \tag{3.19}$$

  Nesse caso, como a vantagem é positiva, a função objetiva aumentará se a ação se tornar mais provável, ou seja, se $\pi_\theta(a|s)$ aumentar. Mas a função min nesse termo limita o quanto o objetivo pode aumentar. Uma vez que $\pi_\theta(a|s) > (1+\epsilon)\pi_{\theta_k}(a|s)$, o min entra em ação e este termo atinge um teto de $(1+\epsilon) \cdot A^{\pi_{\theta_k}}(s, a)$. Portanto: a nova policy não se afasta muito da policy anterior. É possível analisar isso no gráfico em 3.

- **Vantagem** (A) **negativa**: Suponha que a vantagem para o par (s, a) é negativa. Caso em que sua contribuição para a função objetivo é:

$$L(s, a, \theta_k, \theta) = \max\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1-\epsilon)\right) A^{\pi_{\theta_k}}(s, a). \tag{3.20}$$

  Como a vantagem é negativa, a função objetiva aumentará se a ação se tornar menos provável, ou seja, se $\pi_\theta(a|s)$ diminuir. Mas o max nesse termo limita o quanto o objetivo pode aumentar. Uma vez que $\pi_\theta(a|s) < (1-\epsilon)\pi_{\theta_k}(a|s)$, o max atua nesse momento e o termo atinge um piso de $(1-\epsilon)A^{\pi_{\theta_k}}(s, a)$. Portanto, de novo: a nova policy não se afasta muito da policy anterior. É possível analisar isso no gráfico em 3.

Portanto, o que vimos até agora é que o *Clip* funciona com um regularizador para que a *policy* não mude drasticamente de uma atualização para outra. Por fim, o hiper-parâmetro ϵ funciona como o limitador que dita o quão a nova *policy* pode mudar em relação à antiga.

3.2.2 *Exploration* vs. *Exploitation*

PPO treina uma *policy* estocástica de forma online. Isso significa que ele explora (*exploration*) por ações de amostragens de acordo com a versão mais recente de sua *policy* estocástica. A aleatoriedade na seleção de ações depende das condições iniciais e do procedimento de treinamento. Ao decorrer do treinamento, a *policy* se torna progressivamente menos aleatória, incrementando o conjunto de regras de incentivo a explorar (*exploitation*) as recompensas que já foram encontradas. Isso pode fazer com que a *policy* fique presa em soluções locais.

Para incentivar que a *policy* não fique presa em soluções locais, é atribuído bônus de entropia (WILLIAMS, 1992) (MNIH et al., 2016). O ganho pela desordem do sistema se mostra eficaz ao incentivar a *policy* a tomar atitudes mais aleatórias, fazendo com que o agente explore (*exploration*) mais do seu ambiente e em contra partida aproveite menos das recompensas já encontradas (*exploitation*).

3.2.3 Algoritmo

Concatenando os termos devidos, podemos chegar na seguinte função objetivo, que pode ser maximizada (aproximadamente) a cada iteração:

$$L_t^{CLIP+VF+S}(\theta) = E_t[L_t^{CLIP}(\theta) - c_1 \cdot L_t^{VF}(\theta) + c_2 \cdot S[\pi_\theta](s_t)] \tag{3.21}$$

Onde c_1 e c_2 são coeficientes, S denota o bônus por entropia, e L_t^{VF} é o erro quadrado da perda $(V_\theta(s_t) - V_t^{target})^2$.

3.2.3.1 Pseudocódigo

Algorithm 3 PPO-Clip

1: Input: Parâmetros iniciais da policy θ_0, valor inicial da função de parâmetros ϕ_0
2: **for** $k = 0, 1, 2, ...$ **do**
3: Colete o conjunto de trajetórias $\mathcal{D}_k = \{\tau_i\}$ executadas pela policy $\pi_k = \pi(\theta_k)$ no ambiente.
4: Contabilize rewards-to-go $\hat{R}_t$.
5: Contabilize estimativas da vantagem, $\hat{A}_t$ (Usando algum método de estimativa de vantagem) baseado na atual função de valor V_{ϕ_k}.
6: Atualize a policy maximizando o PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

Tipicamente via stochastic gradient ascent com Adam (KINGMA, 2015).
7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

Tipicamente via algum gradient descent algorithm.
8: **end for**

3.3 Algoritmos Genéticos

O fluxograma para algoritmos genéticos é relativamente simples, se comparados com algoritmos de *policy gradient*. Para uma ampla gama de problemas de otimização, os algoritmos genéticos são soluções excelentes tais como os clássicos problemas de controle (SUTTON; BARTO, 2018), *Cart Pole* por exemplo (Ilustrado na Figura 9). Isso se deve, sobretudo, à facilidade de implementação e uma boa tendência de encontrar soluções globais para problemas não tão complexos. Dessa forma, acompanhe a sequência de passos que define os tipos de algoritmos genéticos na figura 4.

Para tanto, vamos discutir um pouco a respeito do funcionamento dos operadores genéticos. No presente trabalho, focaremos no operador de seleção e de mutação.

3.3.1 Seleção

De acordo com os passos iniciais registrados na Figura 4, temos uma população de N agentes. Para cada agente, criaremos uma *policy* arbitrária π_θ, onde $\theta \in \mathbb{R}^d$ é o vetor de parâmetros do modelo. Portanto, teremos um conjunto de *policies* que denotaremos por $\mathcal{P} = \{\pi_{\theta_i}\}_{i=1,...,N}$. No processo de seleção, é necessário que exista um método de

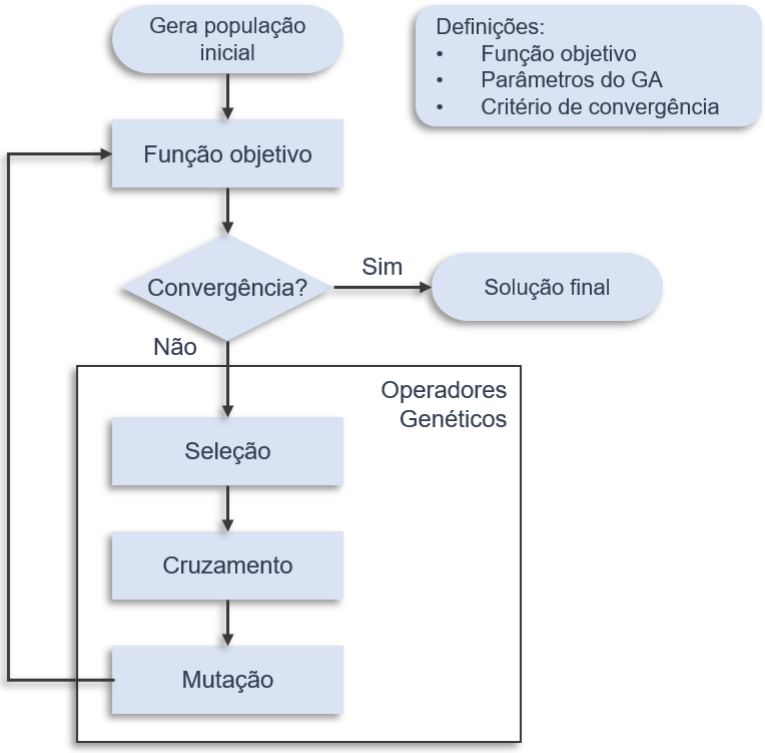

Figura 4 – Algoritmo Genético.

avaliação dos indivíduos. Para tanto, definiremos uma função de *fitness* denotada por $f(\theta)$. Matematicamente, podemos definir que:

$$f(\theta) = E_{\tau\sim\pi_\theta}[R(\tau)] = J(\pi_\theta) \tag{3.22}$$

Tendo em vista que temos um método objetivo de avaliação, agora é possível selecionar os indivíduos mais aptos pelos maiores valores de $f(\theta)$. Adotaremos um valor $\lambda \in (0, 1)$ para representar o percentual de indivíduos selecionados. Portanto, após a aplicação da seleção, teremos a seguinte desigualdade:

$$f(\theta_{x_1}) \geq f(\theta_{x_2}) \geq \dots f(\theta_{x_N}) \xrightarrow{Equação\ 3.22} J(\pi_{\theta_{x_1}}) \geq J(\pi_{\theta_{x_2}}) \geq \dots J(\pi_{\theta_{x_N}}) \tag{3.23}$$

Como citado anteriormente, selecionaremos os $\lfloor \lambda N \rfloor$ agentes com maior valor da função *fitness*. Logo, teremos um novo conjunto que denotaremos por $\mathcal{P}^*$, no qual teremos somente os agentes com maiores valores de $f(\theta)$:

$$\mathcal{P}^* = \{\pi_{\theta_{x_i}}\}_{i=1,\dots,\lfloor \lambda N \rfloor} \tag{3.24}$$

Dessa forma, podemos registrar os agentes que obtiveram os maiores valores de recompensa daquela geração. Além disso, para construir a próxima geração, utilizaremos o conjunto definido em 3.24.

3.3.2 Cruzamento

Apesar de não estarmos utilizando o operador de cruzamento no presente trabalho, vamos definir de forma objetiva. Como o próprio nome sugere, a técnica de cruzamento consiste em coletar características de duas instâncias (ou mais) e criar um novo indivíduo com base nos anteriores. Portanto, dados dois vetores de parâmetros $\theta_X = (\theta_{X,1}, \theta_{X,2}, \ldots, \theta_{X,n})$ e $\theta_Y = (\theta_{Y,1}, \theta_{Y,2}, \ldots, \theta_{Y,n})$, vamos definir o novo indivíduo como sendo θ_{new} e será matematicamente definido por:

$$\theta_{new} = \alpha \cdot \theta_X + \beta \cdot \theta_Y \tag{3.25}$$

Onde $\alpha = \begin{bmatrix} \alpha_1 & 0 & 0 & \ldots & 0 \\ 0 & \alpha_2 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \alpha_n \end{bmatrix}$, $\beta = \begin{bmatrix} \beta_1 & 0 & 0 & \ldots & 0 \\ 0 & \beta_2 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \beta_n \end{bmatrix}$.

Além disso, $\forall i \in [0, n] \rightarrow \alpha_i + \beta_i = 1$ com α_i e β_i reais positivos. Logo, teremos que:

$$\theta_{new} = (\alpha_1 \cdot \theta_{X,1} + \beta_1 \cdot \theta_{Y,1} \ ; \ \alpha_2 \cdot \theta_{X,2} + \beta_2 \cdot \theta_{Y,2} \ , \ \ldots \ , \ \alpha_n \cdot \theta_{X,n} + \beta_n \cdot \theta_{Y,n}) \tag{3.26}$$

Para um caso geral, podemos pensar que cada novo parâmetro do novo indivíduo será uma média ponderada dos parâmetros dos indivíduos que o geraram. Em casos particulares, podemos apenas trocar os parâmetros sem promover alterações nos valores com médias ponderadas.

3.3.3 Mutação

A operação de mutação consiste em alterar parâmetros de um determinado conjunto de informações. Nesse caso, nosso conjunto de informações é a nossa *policy* parametrizada. Para alterar parâmetros utilizaremos o seguinte procedimento:

- Dado um agente cuja *policy* $\pi_\theta \in \mathcal{P}^*$, criaremos outra *policy* $\pi_{\theta'}/\theta' = \theta + \frac{Z}{k}$ onde $k \in \mathbb{R}$ e $Z \sim U(-1, 1)^d$ é um vetor de números aleatórios uniformemente distribuídos entre -1 e 1 e com as mesmas dimensões d do vetor θ.

Logo, faremos um novo conjunto populacional de agentes $\mathcal{P}_{new} = \{\pi_{\theta'_i}\}_{i=1,\ldots,N}$, de tal forma que $\forall \pi_{\theta'_i}$, onde $i > \lfloor \lambda N \rfloor$, teremos que:

$$\theta'_{i \ > \ \lfloor \lambda N \rfloor} = \theta_{x_i} + \frac{Z}{k}, \tag{3.27}$$

Onde $x_i = i \mod \lfloor \lambda N \rfloor$. Portanto, utilizaremos todo o conjunto $\mathcal{P}^*$ para construir a população da próxima geração, ou seja, o conjunto $\mathcal{P}_{new}$.

É válido citar que existem outras maneiras de se implementar a mutação. Por exemplo, é possível mudar apenas alguns pesos do vetor de parâmetro θ' e atribuir ao evento uma taxa $m \in (0, 1)$.

3.3.4 Domínio do tempo

Note que o modelo genético aqui descrito leva em conta que cada passo no *loop* de simulação é feito e aplicado para uma geração de *policies*. Para traçarmos o padrão de comparação com os outros modelos, será necessário transferirmos isso para passos de simulação, haja vista que os outros algoritmos utilizam o números de *steps* como o domínio do tempo. Logo, apesar da simulação ser feita por número de geração, os gráficos serão esboçados utilizando a soma de cada passo de simulação de cada modelo individual das gerações.

4 METODOLOGIA E RESULTADOS

Tendo em vista os métodos citados na seção 3, além dos diversos existentes e citados na taxonomia do *RL* na figura 2, é necessário que existam avaliações para seus respectivos desempenhos de aprendizado. Isso se deve, sobretudo, ao fato de que, para uma mesma tarefa, algoritmos diferentes podem desempenhar de formas diferentes. Nesse sentido, é de suma importância que critérios comparativos entre modelos sejam abordados para que possamos julgar em quais situações cada método performará melhor. Além disso, é possível que uma visão mais precisa dos pontos fortes e fracos de cada algoritmo nos permita propor possíveis melhorias e adaptações em algum dos métodos.

Nesse aspecto, cada um dos algoritmos citados (3.1, 3.2 e 3.3) precisam ser explorados de tal forma que possamos julgar, com as métricas adequadas, qual deles pode performar e apresentar melhores resultados para uma tarefa fixa.

4.1 Soluções locais e soluções globais

Inseridos nos nossos objetivos gerais, também vamos abordar situações em que o agente pode ter resistência no processo de aprendizagem. Isso pode ocorrer em casos que o algoritmo tende a retornar soluções locais e não globais de uma tarefa.

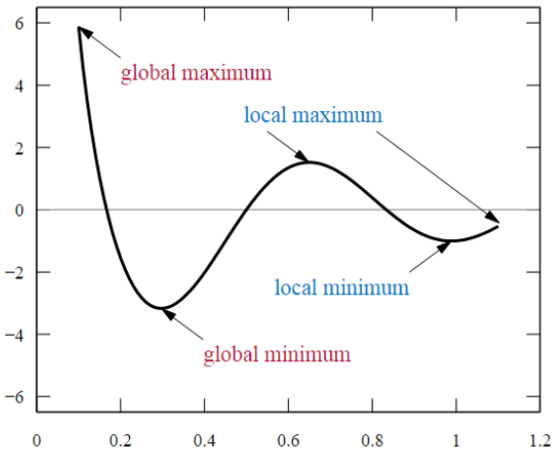

Figura 5 – Exemplos de soluções locais e globais.

Para realizar uma tarefa e alcançar o nosso objetivo, queremos encontrar uma solução de mínimo global. Porém, existem casos em que o algoritmo pode tender a fornecer soluções de mínimo local. A exemplo disso, podemos citar um caso que pode ser evidenciado utilizando os algoritmos citados aqui.

4.1.1 Lunar Lander (*gym - OPENAI*)

Lunar Lander é um ambiente disponibilizado pela biblioteca *Gym*, criada pela *Open-AI*, no qual o nosso objetivo é pousar uma espaço-nave em um ambiente demarcado. Porém, o pouso precisa ser suave, sob pena de, caso o pouso seja brusco, destruir a nave e receber penalizações (ou recompensas negativas).

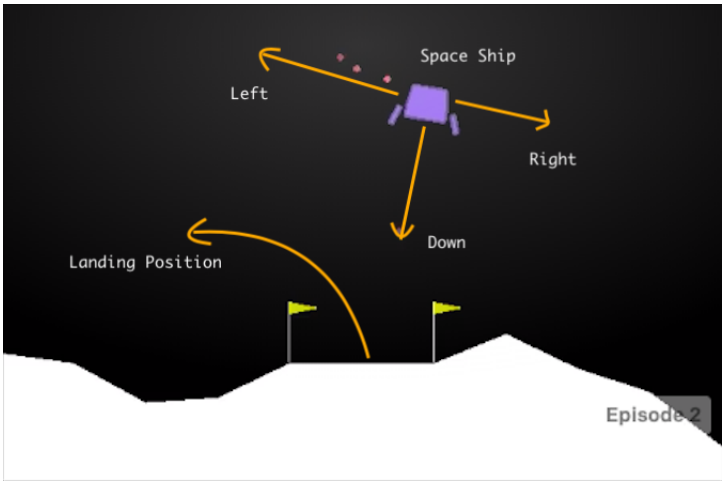

Figura 6 – Lunar Lander environment - Gym.

A solução ideal que buscamos é aquela em que a nave pousa suavemente no solo (dentro das marcações), maximizando sua recompensa. Porém, a penalidade por cair e se destruir pode ser tão alta, que o sistema pode acabar nos fornecendo uma solução local: A nave tentar permanecer no ar pela maior quantidade de tempo possível. Nesse aspecto, é preciso contornar esse problema de alguma forma. Por exemplo, inserir entropia no sistema para que a *policy* tome ações mais aleatórias e dessa forma consiga alcançar o objetivo central de maximizar o retorno. Outra opção é utilizar algoritmos que possuem uma mudança um tanto maior nos parâmetros da *policy*, o que nesse caso, pode ser fornecido pela diversidade oferecida pelos algoritmos genéticos. Portanto, nosso objetivo é avaliar quais algoritmos utilizar para nos fornecer maiores chances de alcançar o mínimo global.

4.2 Avaliação de recompensa e tempo de simulação

Além do citado na seção 4.1, dois aspectos que são de fundamental importância na avaliação dos métodos de aprendizado no *Reinforcement Learning* são a recompensa acumulada e o tempo de simulação. Nesse sentido, a maximização da recompensa do agente, por definição, sendo o problema central do *Reinforcement Learning* tal como em 2.18, tem importância significativa na resolução de uma determinada tarefa. Além disso, o tempo de simulação é um fator que devemos sempre levar em conta para avaliar o desempenho do algoritmo, sobretudo, devido a performance estar associada ao custo que o algoritmo exigiu para atingir o objetivo.

Portanto, esses dois aspectos são cruciais para a avaliação performática dos algoritmos citados nesse referido trabalho. Dessa forma, utilizaremos eles como métrica desempenho para estabelecer um padrão comparativo entre os modelos.

4.2.1 Procedimento comparativo entre modelos

Para fazermos análises individuais de modelos, basta avaliar o desempenho do algoritmo sem traçar padrões de comparações. Porém, nesse caso, é necessário que façamos um comparativo entre os modelos, portanto, surge a necessidade de estabelecer procedimentos para tal.

1. **Comparativo por meio da recompensa média**: Para fazermos essa comparação, fixamos uma tarefa (para um mesmo ambiente) e treinamos a *policy* por meio dos modelos que vamos comparar. Depois, fixamos um instante t (final de simulação) igual para ambos os modelos e daí fazemos um comparativo dos retornos esperados para ambos os modelos. Para o exemplo elaborado na Figura 7, os algoritmos 1 e 2

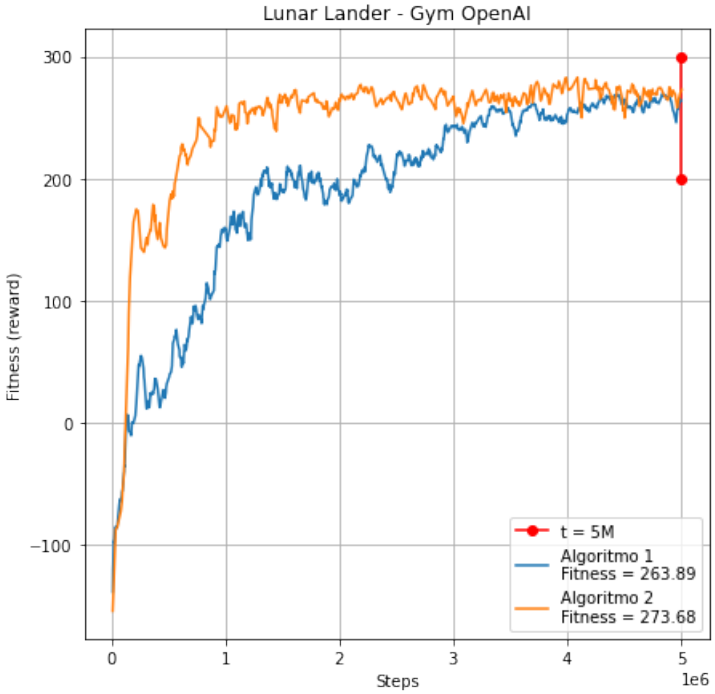

Figura 7 – Comparar recompensa média (fitness) para um valor fixo de t.

(a título de exemplificação) têm desempenho parecido, haja vista suas recompensas serem próximas.

2. **Comparativo por meio de tempo de simulação**: Para fazemos esse tipo de comparação, análogo ao comparativo anterior, vamos fixar uma tarefa (para um mesmo ambiente) e treinar a *policy* por meio dos modelos que vamos comparar. Desta vez, vamos fixar um valor δ de recompensa e vamos ver o tempo em que

cada algoritmo leva para atingir o valor δ. Para o exemplo elaborado na Figura 8 é possível notar que o algoritmo 2 atinge mais rápido a marcação de recompensa δ, apesar de ambos os algoritmos convergirem para soluções muito próximas.

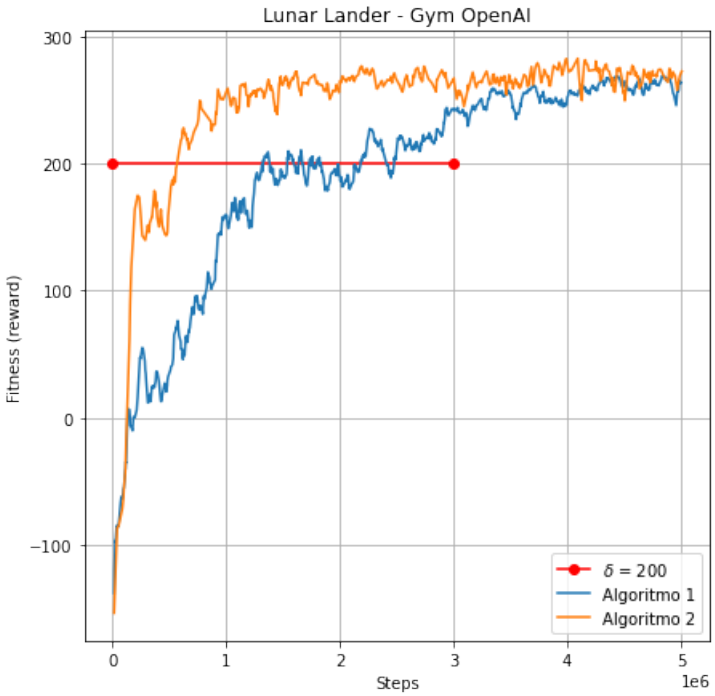

Figura 8 – Comparar tempo para atingir um valor fixo de recompensa (fitness) δ.

4.3 *Curriculum Learning*

Como os agentes autônomos são chamados para realizar tarefas cada vez mais difíceis, novas técnicas serão necessárias para fazer aprender tais tarefas tratáveis. Aprendizagem de transferência (NARVEKAR et al., 2020) é uma área recente de pesquisa que demonstrou acelerar o aprendizado em uma tarefa complexa por transferência de conhecimento de uma ou mais tarefas de origem mais fáceis.

A maioria dos métodos de aprendizagem por transferência existentes trata esta transferência de conhecimento como um processo de uma etapa, onde o conhecimento de todos as fontes são transferidas diretamente para o destino. Contudo, para tarefas complexas, pode ser mais benéfico (e até necessário) adquirir gradualmente habilidades em várias tarefas em sequência, onde cada tarefa subsequente requer e se baseia conhecimento adquirido em uma tarefa anterior.

Portanto, o curriculum serve para ordenar as experiências do agente em tarefas menores que ele vai adquirir ao longo do tempo, de tal forma que melhore a performance e diminua o tempo de aprendizado. Mais adiante, iremos formalizar o conceito de *curriculum learning* e descrever como avaliar os benefícios e custos do *curriculum*.

Para analisarmos os resultados, vamos dividir em duas etapas: Resultados individuais e resultados comparativos. Para os resultados individuais, vamos avaliar cada algoritmo de maneira individual e analisar sua performance para diferentes ambientes. Já para os resultados comparativos, vamos avaliar de seguindo o procedimento descrito na Seção 4.2.1.

4.4 *CartPole - Gym OpenAI*

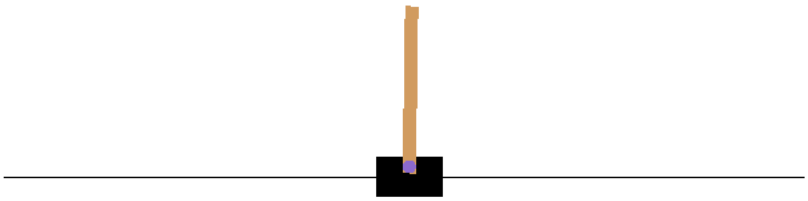

Figura 9 – *CartPole - Gym*

CartPole é um ambiente que pertence à biblioteca *Gym*, cuja classe é um problema clássico de controle. A tarefa associada é que o agente equilibre o bastão verticalmente para cima (Figura 9. Para tanto, o agente possui um espaço de ações discretas, tal que contém as seguintes possibilidades:

- Empurrar o carro para a esquerda (0).
- Empurrar o carro para a direita (1).

Além disso, o agente tem acesso à um espaço de observações que consiste em:

- Posição do carrinho.
- Velocidade do carrinho.
- Ângulo do bastão.
- Velocidade angular do bastão.

Para cada passo de simulação em que o bastão não cai, será atribuída uma recompensa de +1. Note que, nesse caso, **os gráficos de recompensa e tamanho médios serão equivalentes**.

4.4.1 *REINFORCE*

4.4.1.1 Recompensa Média

Utilizando o modelo básico *REINFORCE*, vamos analisar o seu desempenho para o ambiente *CartPole*. Vale ressaltar a alta variância desse método, podemos esperar que o gráfico de aprendizado seja ruidoso, principalmente se comparado ao modelo *PPO*. Acompanhe a Figura 10.

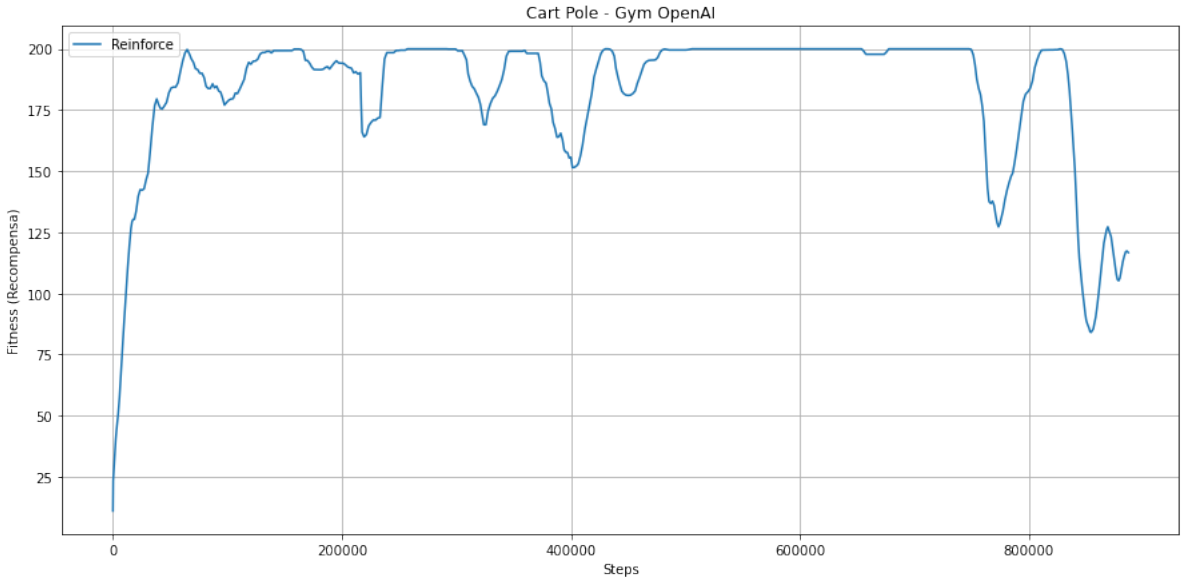

Figura 10 – Recompensa x Steps - *REINFORCE*

4.4.2 *REINFORCE* com *baseline*

4.4.2.1 Recompensa Média

Vamos utilizar o algoritmo *REINFORCE* novamente, porém agora com a utilização de uma *baseline* como descrito na seção 3.1.2 (utilizamos a *baseline* como sendo a recompensa média). Acompanhe a Figura 11.

4.4.3 PPO

4.4.3.1 Recompensa Média

Utilizando o modelo *PPO*, vamos analisar a recompensa média para o ambiente *CartPole*. É possível analisar a subida bastante suave, livre de oscilações. Isso se deve, sobretudo, ao sofisticado mecanismo pertencente ao algoritmo *PPO*, que trás a premissa de que *policies* mais conservadores promove a diminuição da variância. Acompanhe a Figura 12.

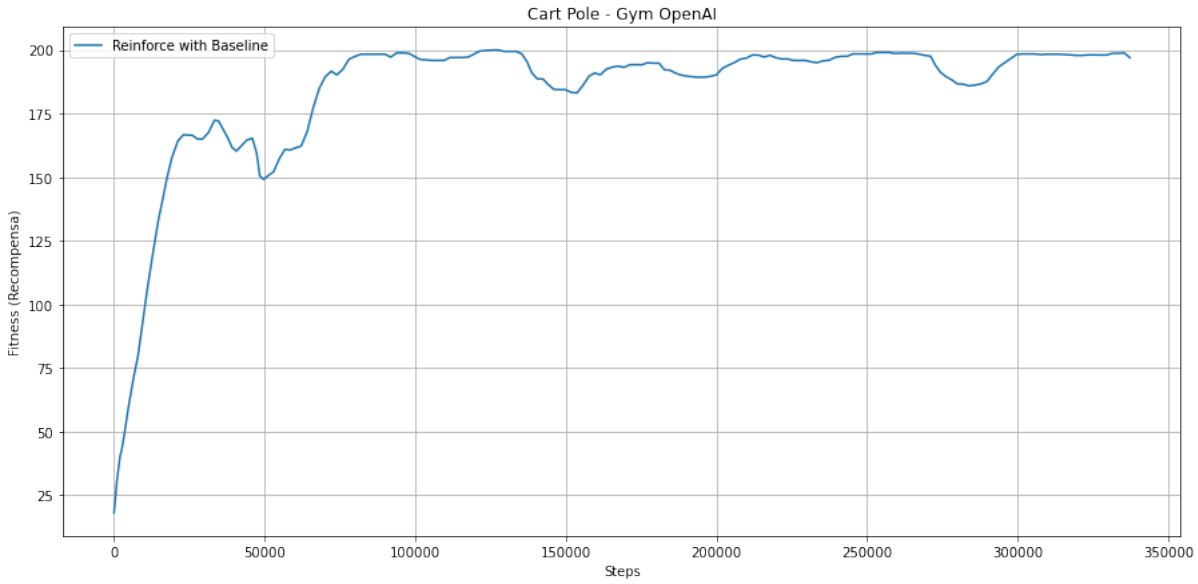

Figura 11 – Recompensa x Steps - *REINFORCE* com *baseline*

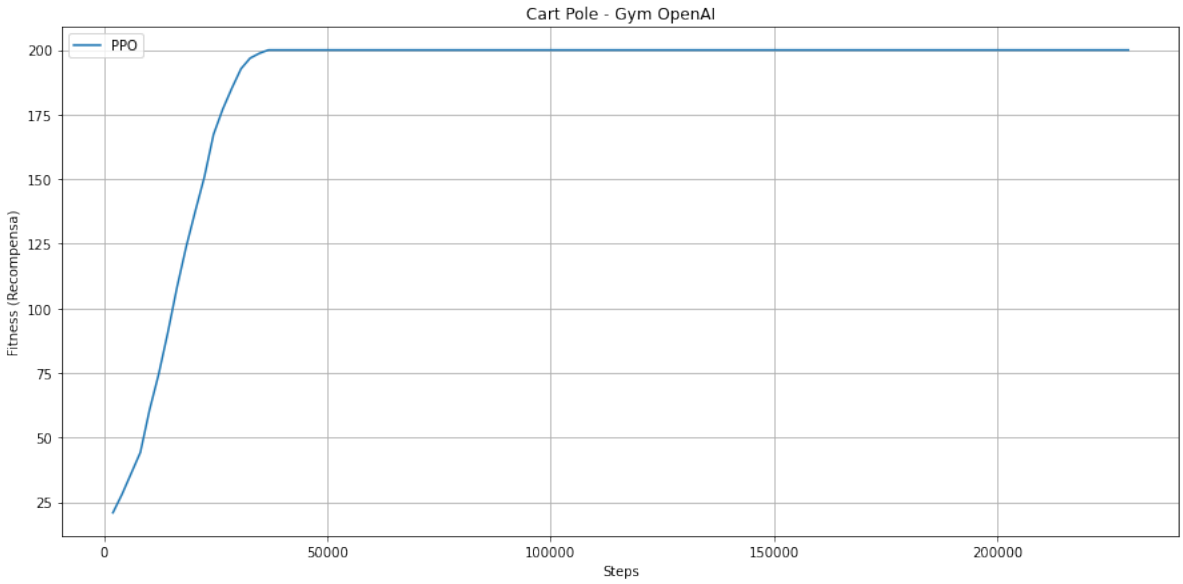

Figura 12 – Recompensa x Steps - *PPO*

4.4.4 Algoritmo Genético

4.4.4.1 Recompensa Média

Vamos analisar o comportamento da recompensa média para o modelo de algoritmos genéticos aplicado ao *CartPole*. Pelo fato de o *CartPole* ser um problema de controle relativamente simples, é de se esperar que o treinamento seja mais direto quando comparado à tarefas mais complexas. Portanto, Acompanhe a Figura 13.

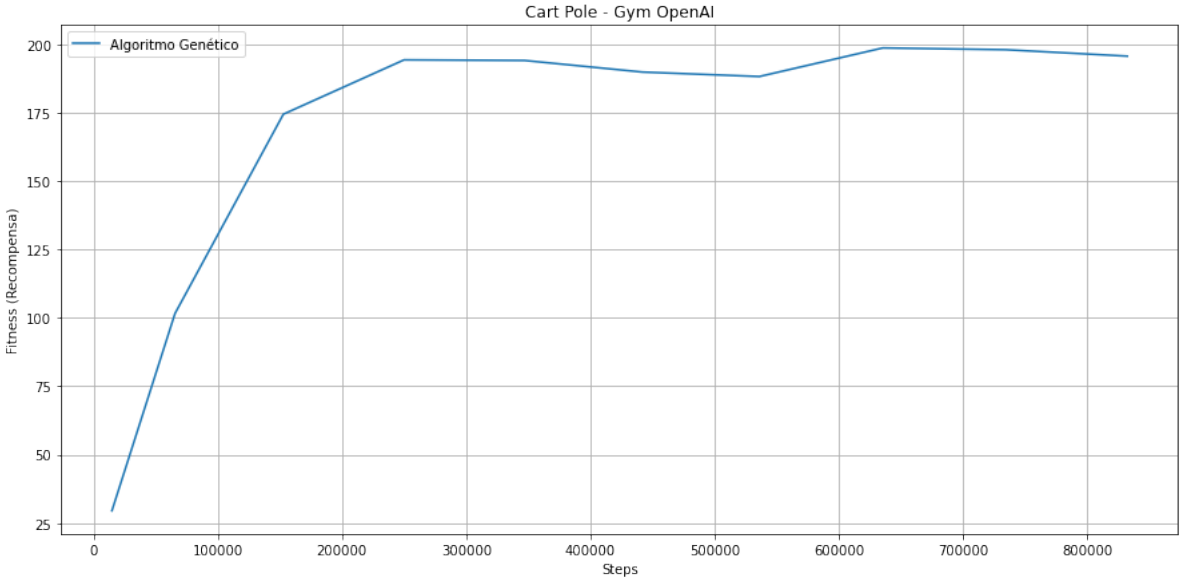

Figura 13 – Recompensa x Steps - Algoritmo Genético

4.4.5 Comparativo entre modelos

Tendo em vista as seções a análise que fizemos para os modelos diferentes, vamos abordar agora o procedimento que foi citado na Seção 4.2.1. Como podemos ver na Figura 14, os três modelos convergem para valores de recompensas médias parecidos para um valor fixo de tempo $t = 800k$. Porém, podemos ver que o modelo *PPO* atinge o valor δ de recompensa em um tempo menor do que o algoritmo genético e o *REINFORCE*, que por sua vez, *REINFORCE* atinge δ antes do algoritmo genético.

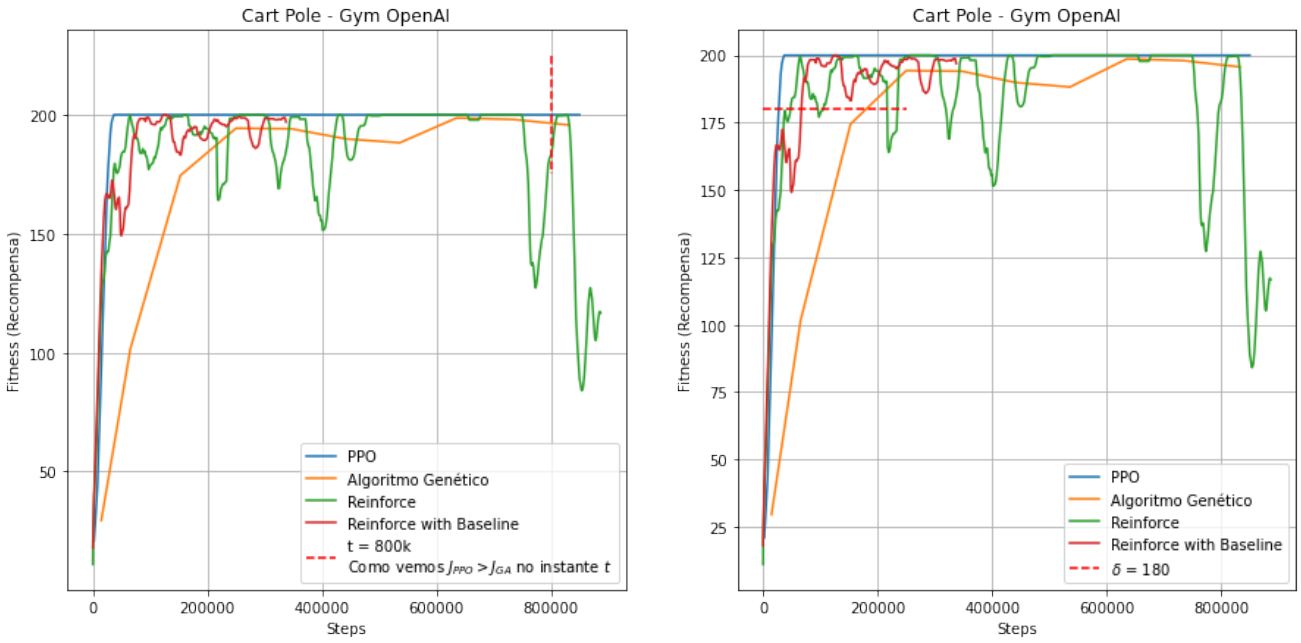

Figura 14 – Procedimento de comparação entre os modelos tal como descrito na Seção 4.2.1.

Analisando em uma faixa de tempo conveniente, é possível observar que de fato a

utilização da *baseline* torna o aprendizado mais estável quando comparado ao algoritmo sem baseline. Acompanhe a Figura 15.

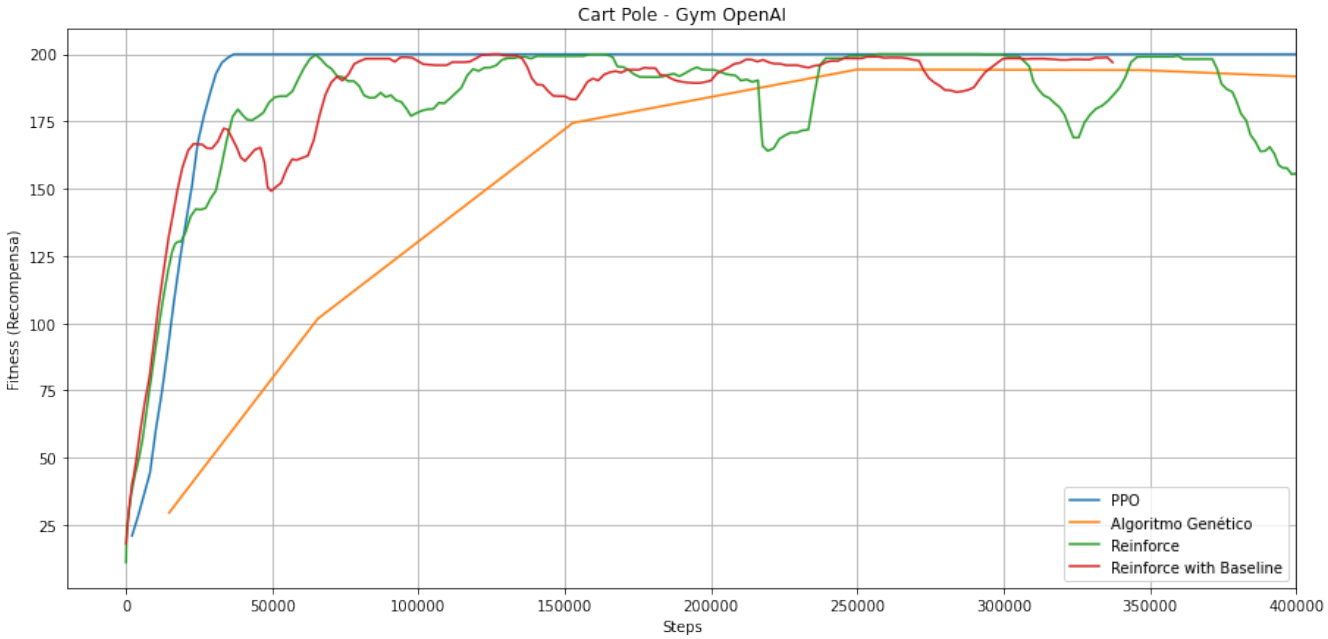

Figura 15 – Comparativo de instabilidade dos modelos para uma faixa de tempo adequada.

4.5 *Lunar Lander - Gym OpenAI*

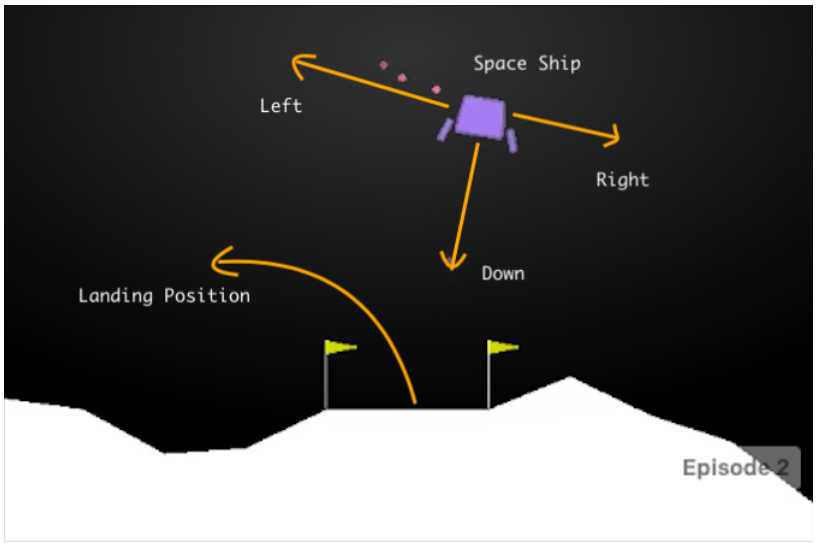

Figura 16 – *Lunar Lande - Gym OpenAI*

Lunar lander é um ambiente que pertence à biblioteca *Gym*, da *OpenAI*. Como mencionado na Seção 4.1.1, a tarefa para que precisamos fazer é pousar a nave sem que ela se quebre (perdendo recompensa). O espaço de ações que o *environment* possui é:

- Não fazer nada.

- Ir para a esquerda.
- Ir para a direita.
- Subir.

Além disso, o espaço para observações é:

- Posição em x.
- Posição em y.
- Velocidade em x.
- Velocidade em y.
- Ângulo de inclinação.
- Velocidade angular.
- *Booleano* da base esquerda.
- *Booleano* da base direita.

4.5.1 *REINFORCE*

4.5.1.1 Recompensa média

Ao modelo *REINFORCE*, como vimos até aqui, é de se esperar aspectos expressivos da alta variância. Nesse sentido, é possível ver que a curva de aprendizado cresce, porém com oscilações significativas. Acompanhe a Figura 17.

4.5.1.2 Tempo médio de episódio

Tal como descrito na análise de recompensa, é possível notar as oscilações do gráfico. Porém, para o tamanho médio do episódio é possível notar oscilações ainda mais bruscas. Acompanhe a Figura 18.

4.5.2 *REINFORCE* com baseline

4.5.2.1 Recompensa média

Tal como descrito em seções anteriores, vamos analisar agora o gráfico gerados pela recompensa média do algoritmo *REINFORCE* com baseline (recompensa média da trajetória). Para tanto, acompanhe a Figura 19.

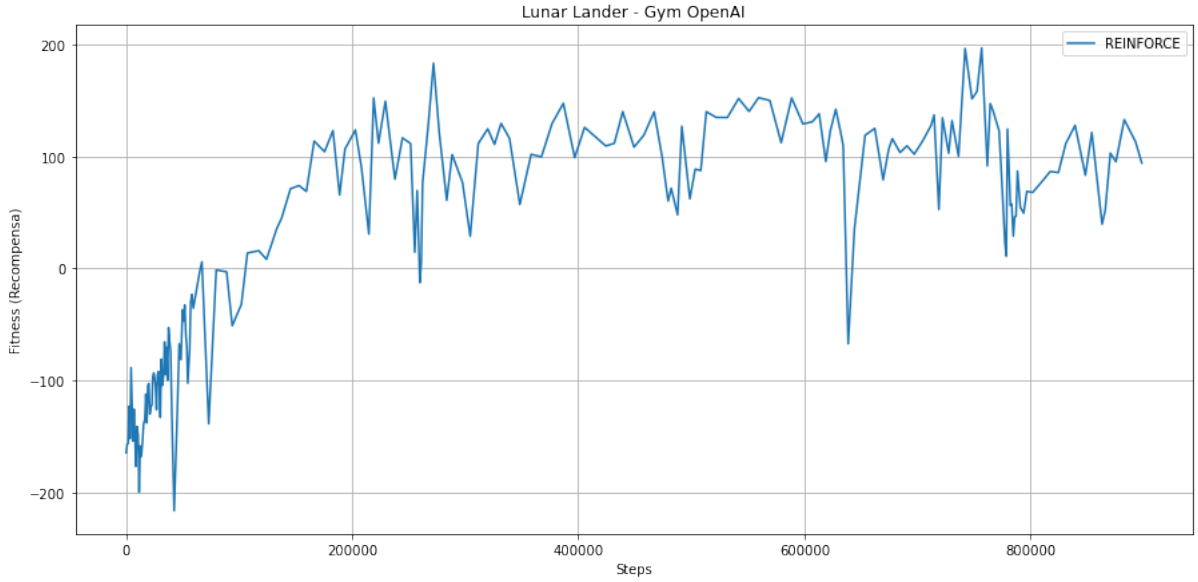

Figura 17 – Recompensa x Steps.

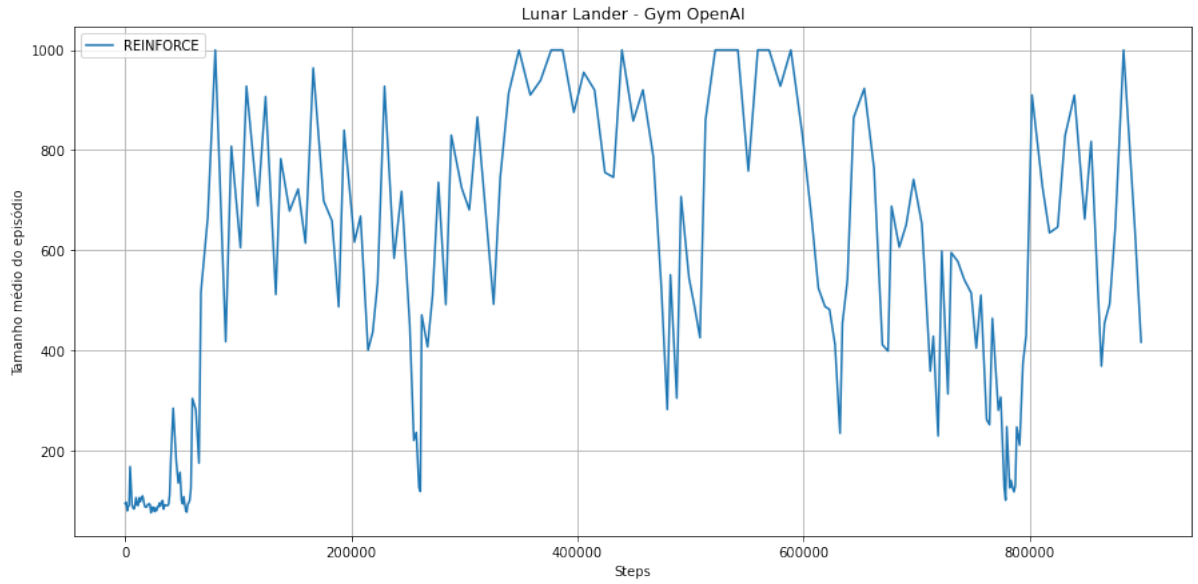

Figura 18 – Tamanho médio do episódio x Steps.

4.5.3 PPO

4.5.3.1 Recompensa média

Utilizando o modelo *PPO*, podemos avaliar como foi o processo de aprendizado através da análise da recompensa média do agente durante uma simulação de 10M de passos. Acompanhe na Figura 20.

4.5.3.2 Tamanho médio de episódio

Além da nossa avaliação da recompensa média de retorno, podemos avaliar também o tempo médio de um episódio. É possível notar que a medida que o processo de aprendizado

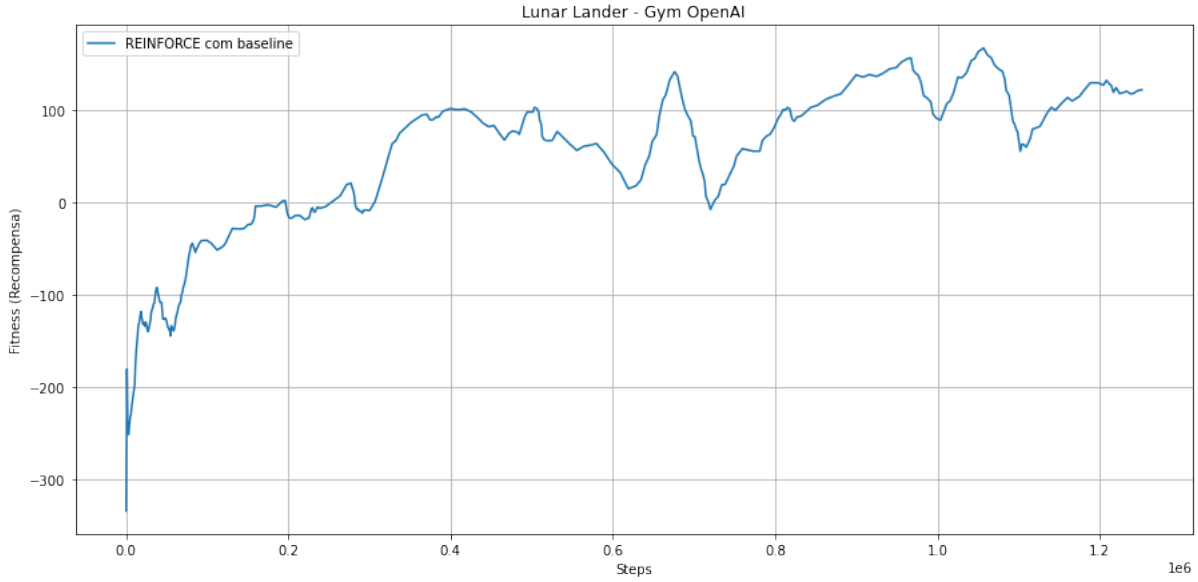

Figura 19 – Recompensa x Steps.

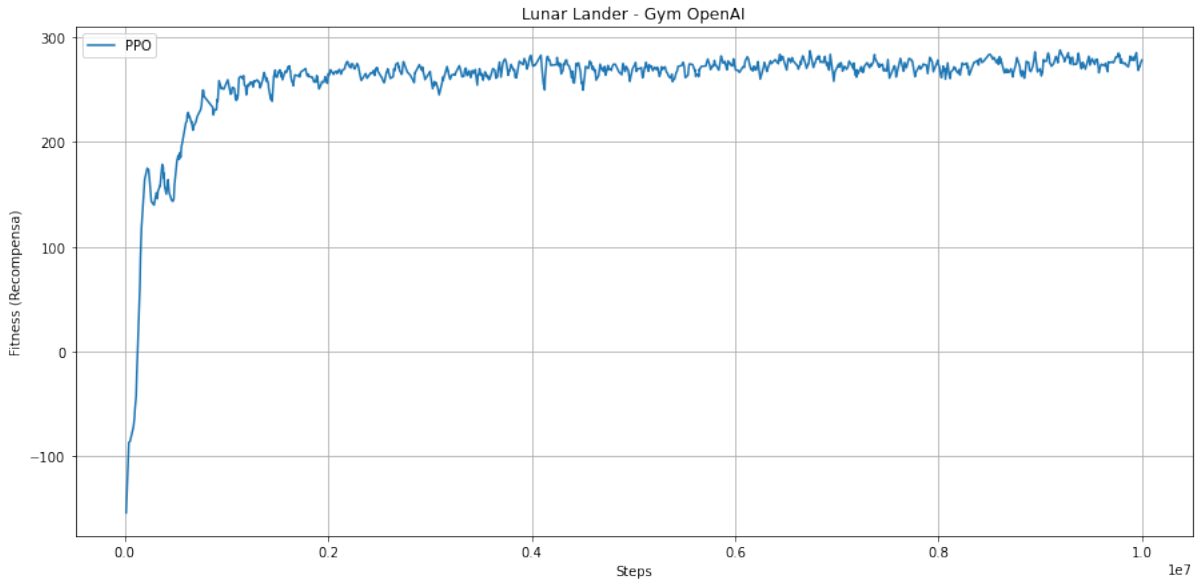

Figura 20 – Recompensa x Steps.

foi sendo instaurado, houve uma queda significativa no tempo de um episódio. Isso se deve ao fato de que a medida com que a *policy* vai sendo treinada, ela passa, gradativamente, a tomar decisões mais diretas para maximizar o retorno. Dessa forma, o tamanho do episódio vai diminuindo. Acompanhe a Figura 21.

4.5.4 Algoritmo Genético

4.5.4.1 Recompensa média

Tal como a análise feita para o *PPO*, vamos analisar a performance do algoritmo genético do ponto de vista da recompensa média. Como podemos ver na Figura 22, o valor para o qual

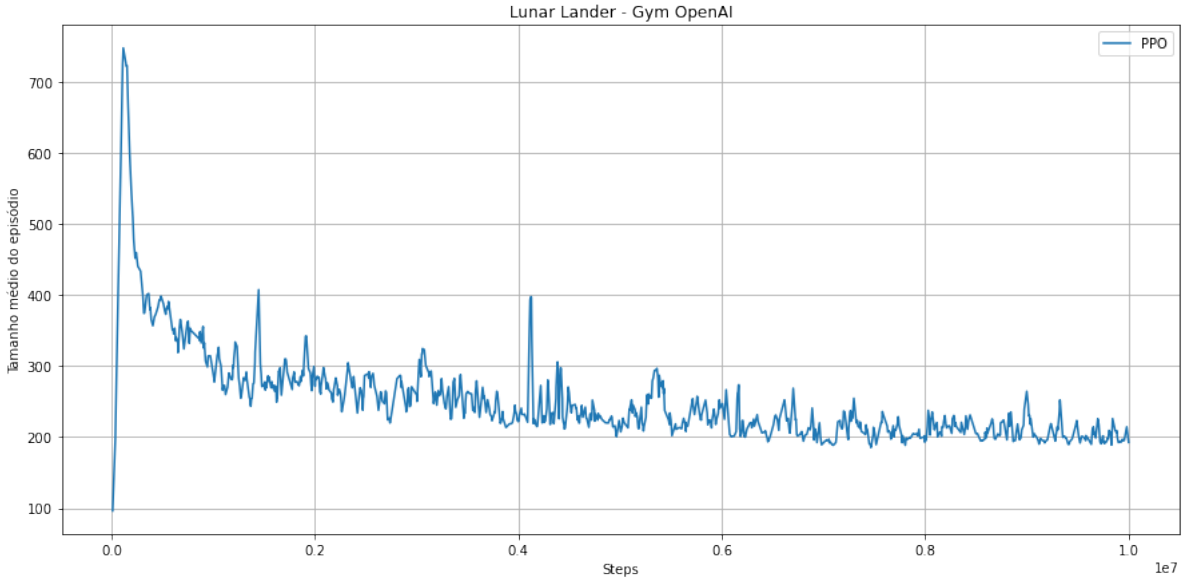

Figura 21 – Queda significativa no tempo do episódio após a nave "aprender" a pousar da forma correta.

convergiu é próximo do valor que convergiu com o *PPO*, porém o tempo foi drasticamente diferente. Nesse sentido, o algoritmo genético necessitou de mais tempo de simulação para treinar a *policy*, portanto, maior esforço computacional.

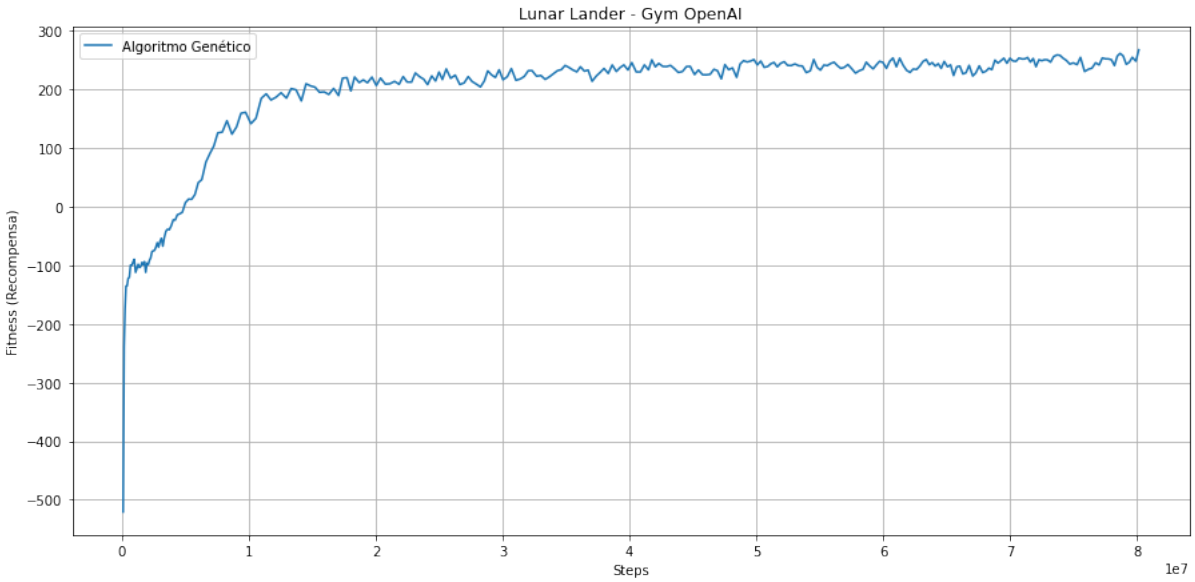

Figura 22 – Recompensa x Steps

4.5.4.2 Tamanho médio de episódio

Na análise do tamanho médio do episódio, é possível ver que encontramos, também, uma maior resistência para estacionar o tamanho médio do episódio em relação ao *PPO*. Isso era de se esperar, dado que o principio do algoritmo genético exige um grande esforço computacional. Acompanhe a Figura 23.

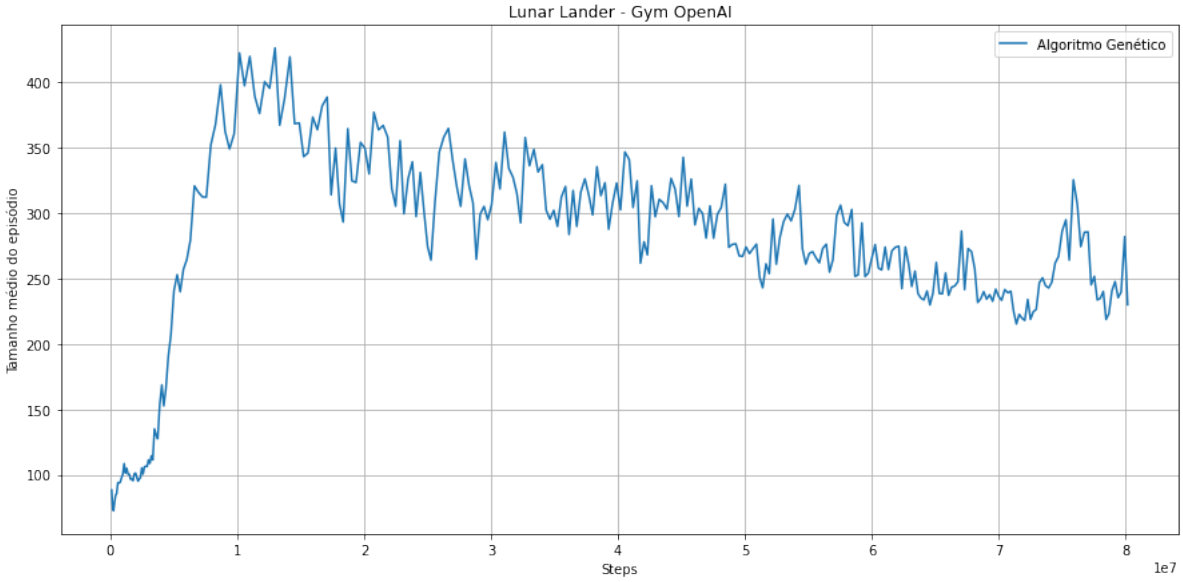

Figura 23 – Tempo médio do episódio x Steps

4.5.5 Comparativo entre modelos

Tal como descrito na Seção 4.2.1, vamos avaliar o tempo em que os modelos levam para atingir o marco de um δ de recompensa e em qual será o maior valor de recompensa para um instante t adotado. Isso está registrado devidamente na Figura 24, onde é possível notar, agora com uma perspectiva mais direta, que o algoritmo genético leva um custo superior para treinar a *policy* em relação à quantidade de passos de simulação. Além disso, é válido notar que o modelo *REINFORCE* se posicionou como um intermediário entre os modelos *PPO* e genético, porém possui variância muito superior e tempo real de simulação muito alto.

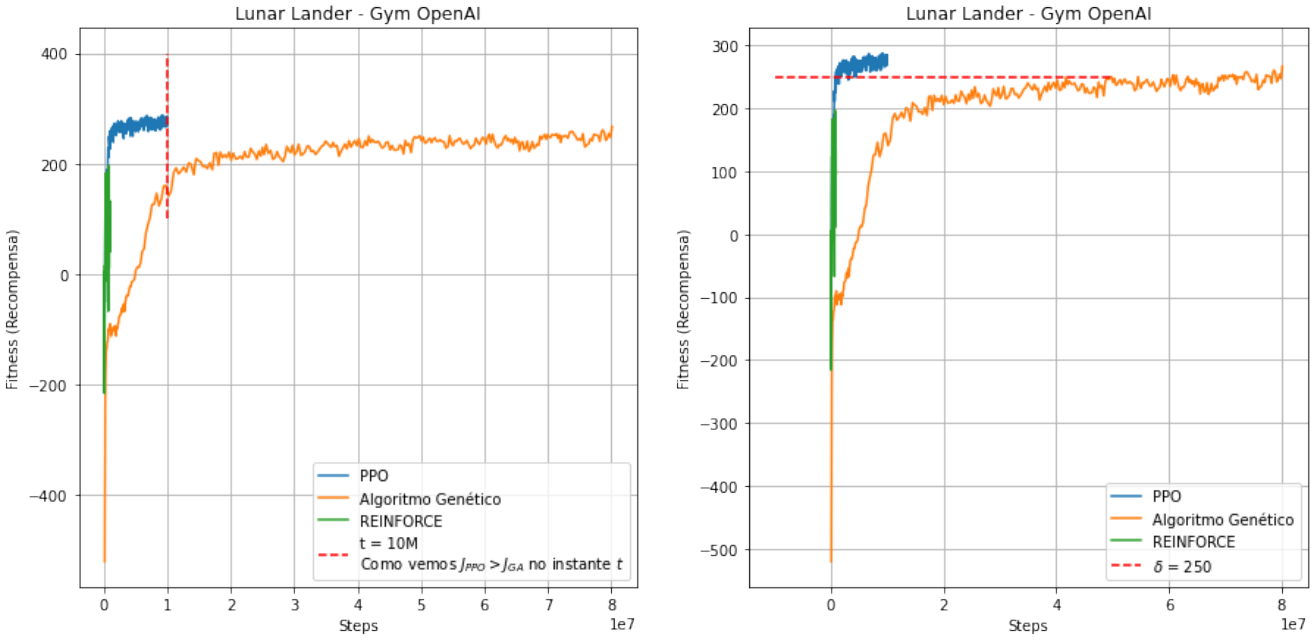

Figura 24 – Comparativo de recompensa x Steps

Além disso, vamos por em perspectiva os gráficos de tamanho médio de um episódio para traçar comparativos mais fidedignos. Para tanto, acompanhe o gráfico da Figura 25.

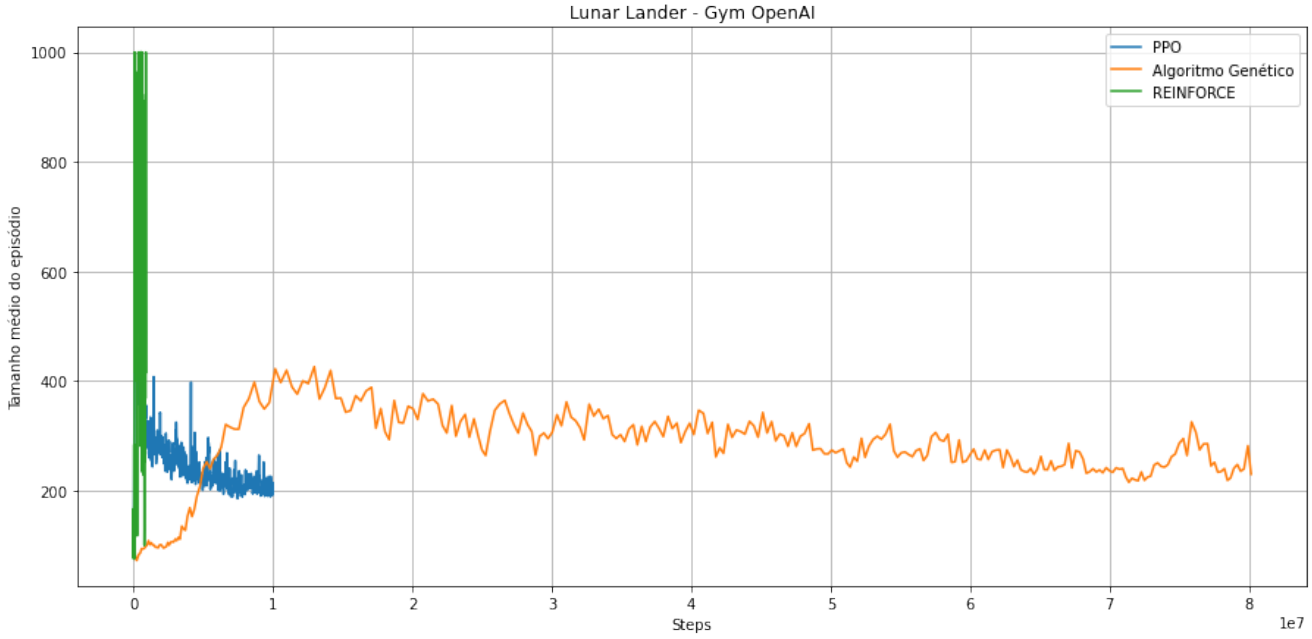

Figura 25 – Comparativo de tempo médio do episódio x Steps

Para colocar em perspectiva a pauta que se iniciou a respeito dos problemas da variância, é de suma importância que coloquemos em perspectiva de uma maneira mais clara os gráficos citados na presente seção. Para tanto, acompanhe os gráficos na Figura 26. É possível observar que a variância para o algoritmo *REINFORCE* é significativamente elevada quando compara com as oscilações dos algoritmos *PPO* e genético.

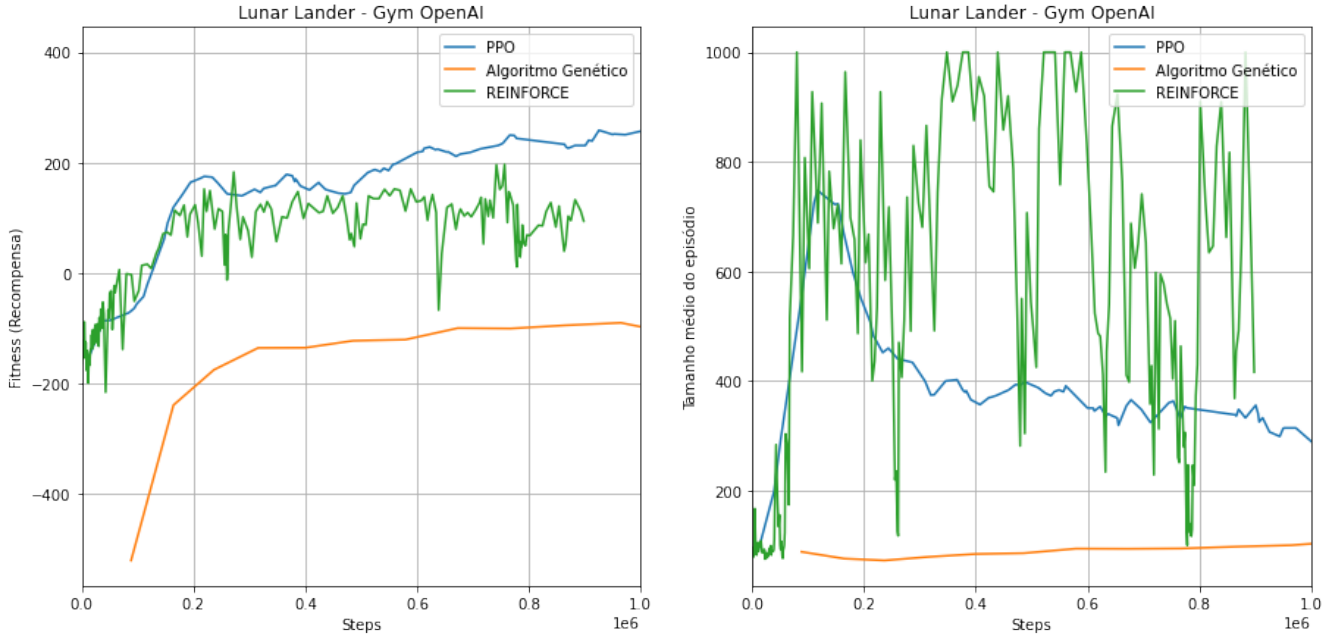

Figura 26 – Perspectiva de análise da variância para os três algoritmos aqui citados.

Vale ressaltar que, em quantidade de passos de simulação, o algoritmo genético se estende muito mais do que os outros dois, mesmo que em contagem de tempo real ele possa ser menor. Isso se deve, sobretudo, ao fato de como realizamos a otimização do agente, ou seja, criando uma "prole" de modelos e aplicando os operadores genéticos como descrito na Seção 3.3.

4.6 *Acrobot - Gym OpenAI*

Esse ambiente faz parte do conjunto de ambientes de controle clássico da biblioteca *Gym* (Figura 27). O objetivo consiste em atingir a marcação com a ponta livre na menor quantidades de passos possível. Para cada passo que a ponta não encontra o objetivo, será atribuída uma recompensa de -1 para o agente. Quando a ponta livre encontrar o objetivo, será atribuída uma recompensa de 0. O limite de recompensa é -100. O espeço de ações é discreto e consiste em três possíveis ações:

- Aplicar torque -1 na junção.
- Aplicar torque 0 na junção.
- Aplicar torque $+1$ na junção.

Além disso, o conjunto de observações fornecidas é composto por 6 valores, onde θ_1 é o ângulo da primeira junção ($\theta_1 = 0$ indica que o primeiro elo está verticalmente para baixo) e θ_2 é relativo ao ângulo do primeiro link ($\theta_2 = 0$ indica que o primeiro elo está alinhado com o segundo):

- $\cos\theta_1$.
- $\sin\theta_1$.
- $\cos\theta_2$.
- $\sin\theta_2$.
- Velocidade angular de θ_1.
- Velocidade angular de θ_2.

4.6.1 REINFORCE

4.6.1.1 Recompensa média

Analisando o gráfico de Recompensa x Steps para o modelo *REINFORCE*, é possível ver, novamente, o ruido provocado pelo algoritmo. Isso se deve, sobretudo, ao fato da

Figura 27 – Acrobot - Gym

variância ser relativamente alta. Além disso, é possível identificar pontos em que houve uma queda significativa no aprendizado da *policy*. Isso pode sinalizar o efeito da *policy* não conservadora. Como citado em (SCHULMAN et al., 2017), atualizações muito bruscas pode acabar destruindo o aprendizado. Acompanhe a Figura 28

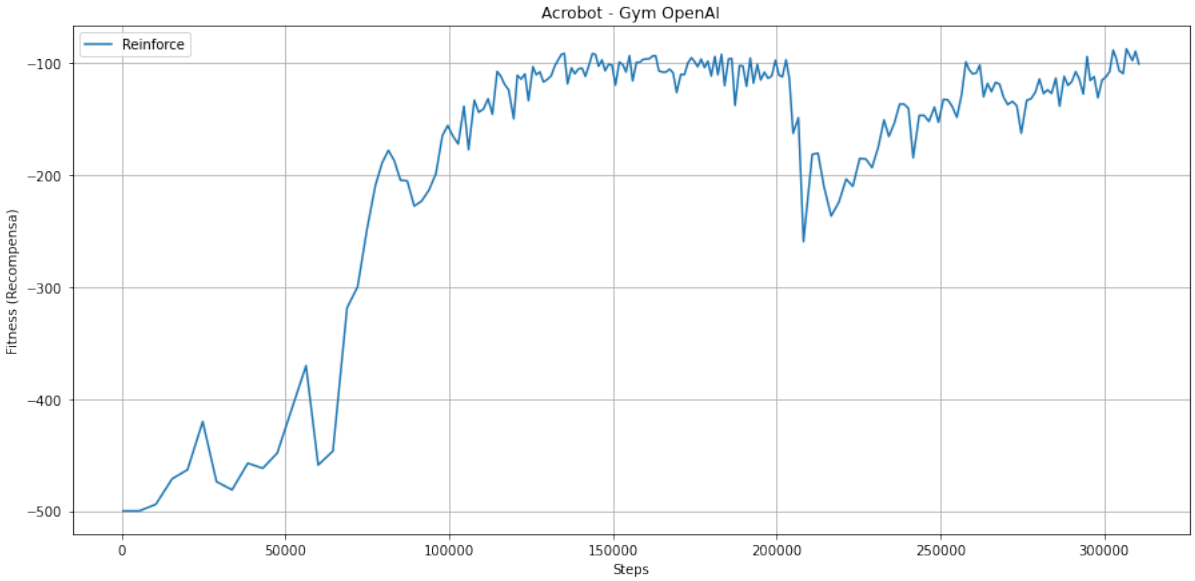

Figura 28 – Recompensa x Steps - *REINFORCE*

4.6.1.2 Tempo médio de episódio

Novamente, os pontos postos em pauta a respeito da recompensa média, se aplicam aqui. Temos um gráfico relativamente ruidoso por conta da variância. Além disso, é possível identificar o momentos em que houve uma queda brusca do aprendizado, aumentando assim o tempo médio do episódio. Acompanhe a Figura 29.

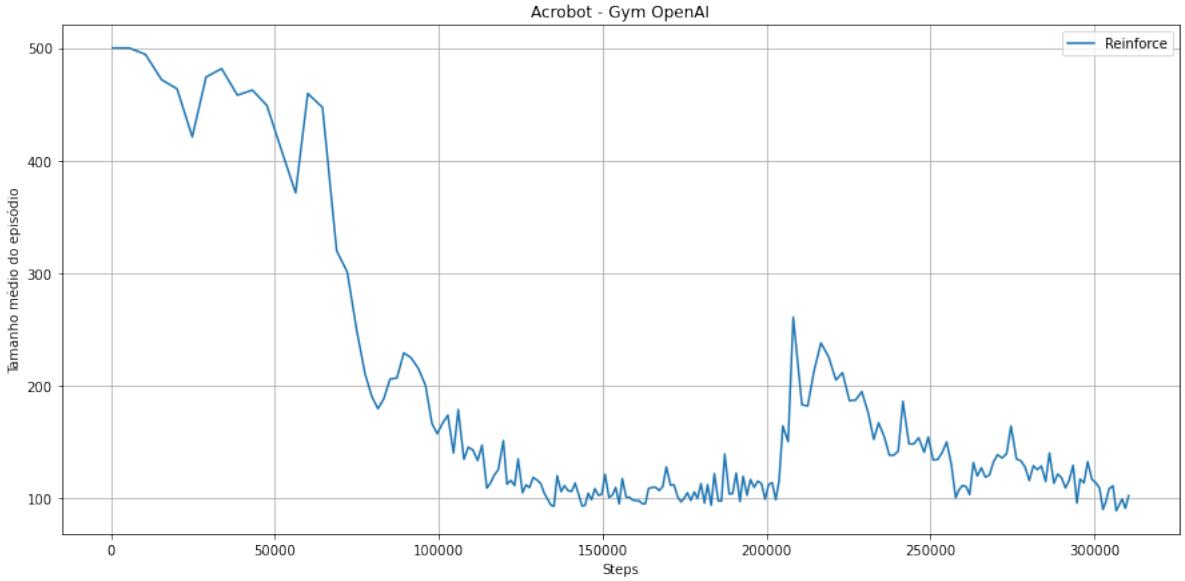

Figura 29 – Tempo médio de episódio x Steps - *REINFORCE*

4.6.2 PPO

4.6.2.1 Recompensa média

Vamos analisar o desempenho do modelo *PPO* ao treinar o agente no ambiente *Acrobot*. Para tanto, acompanhe a Figura 30. Nela é possível ver que a *policy* convergiu relativamente rápido e de forma suave.

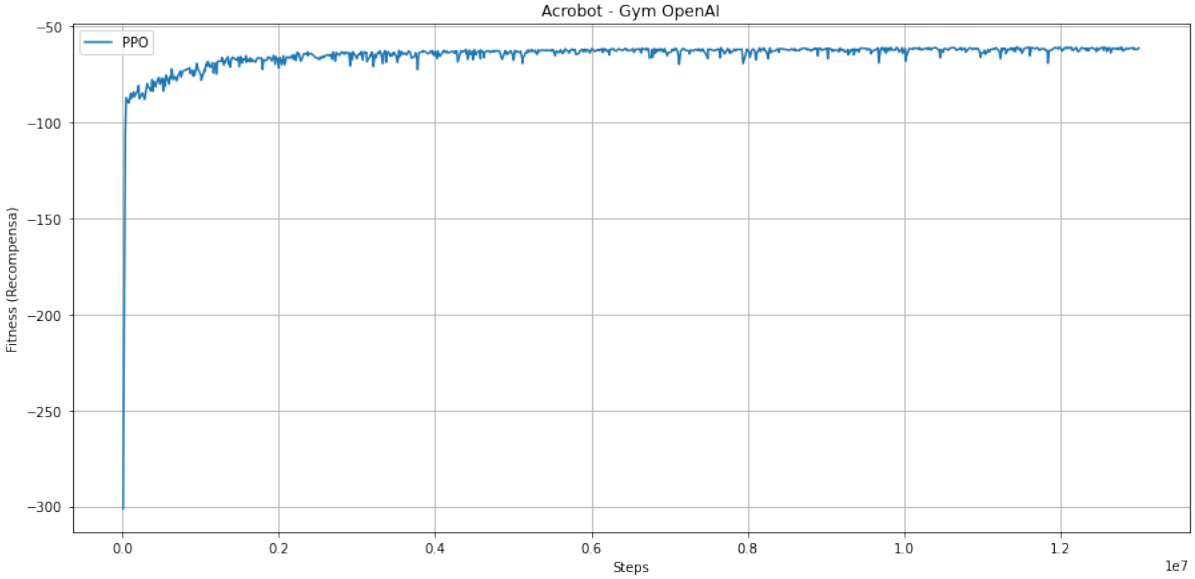

Figura 30 – Recompensa x Steps - *PPO*

4.6.2.2 Tamanho médio de episódio

Analisando o tamanho médio do episódio, faz sentido que exista um rápido decaimento. Isso se deve ao fato de que as articulações tendem a demorar um tempo significativamente

maior quando a *policy* ainda não está treinada. Lembre-se que o objetivo é alcançar a meta com o menor tempo possível. Acompanhe a Figura 31.

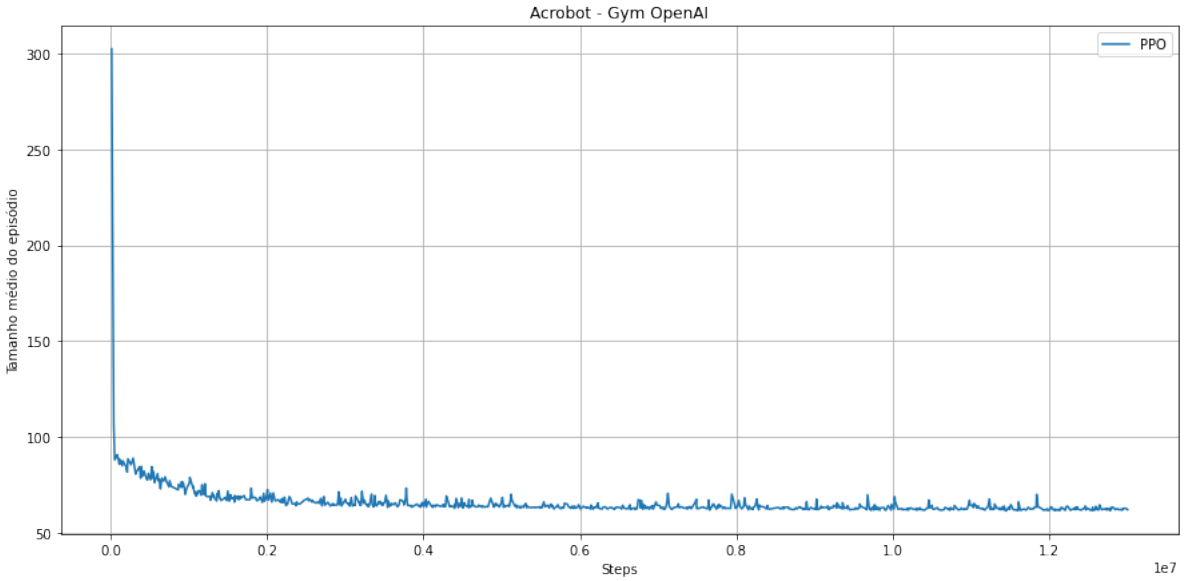

Figura 31 – Tempo médio de episódio x Steps - *PPO*

4.6.3 Algoritmo Genético

4.6.3.1 Recompensa média

Seguindo com a análise para a o modelo de algoritmos genéticos, é possível notar que existe uma oscilação maior em relação ao *PPO*. Acompanhe na Figura 32

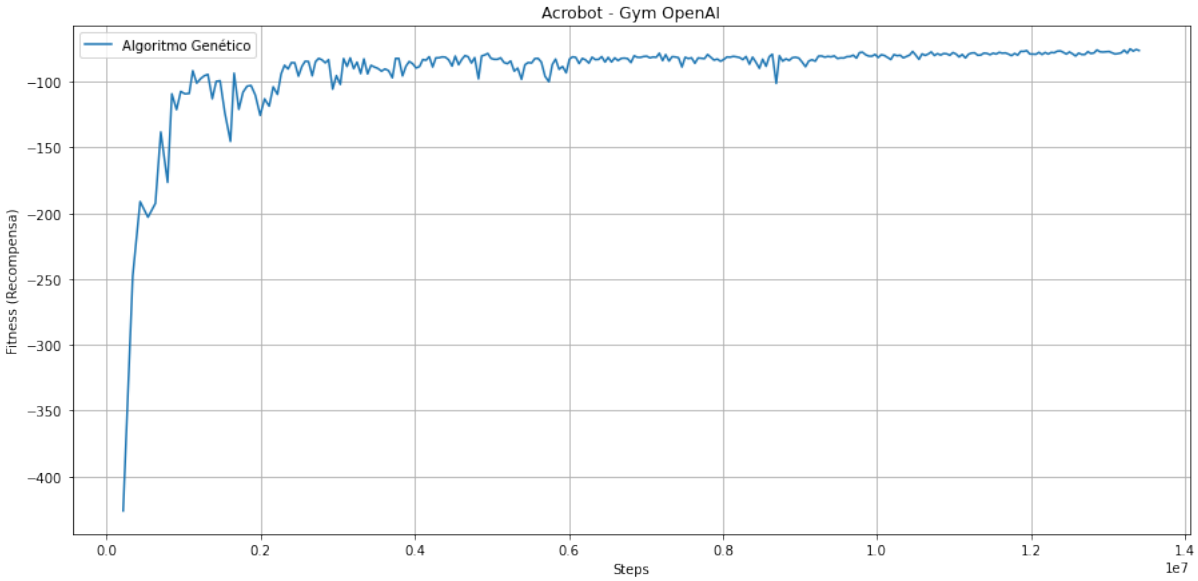

Figura 32 – Recompensa x Steps

4.6.3.2 Tamanho médio de episódio

Em relação ao tamanho médio do episódio, é evidente que o decaimento ocorre também, porém é um pouco mais lento do que quando comparado ao modelo *PPO*. Além disso, é possível notar que o valor em que o gráfico estaciona é um pouco acima do que o valor com o modelo *PPO*. Acompanhe na Figura 33.

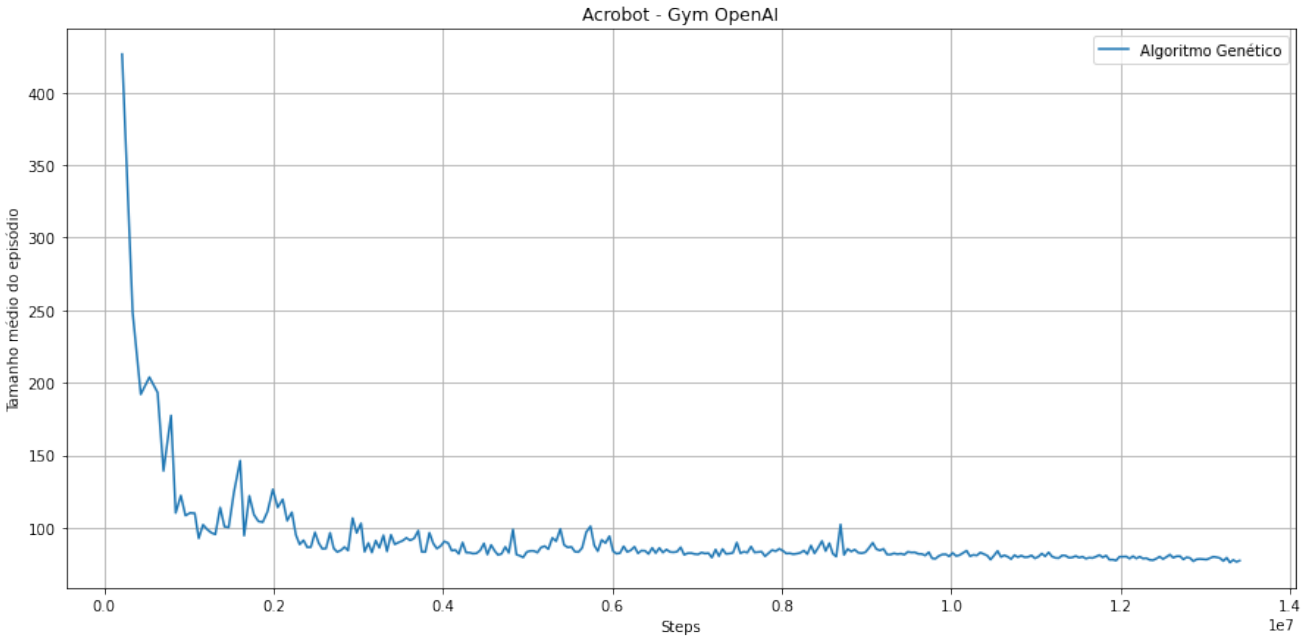

Figura 33 – Tempo médio do episódio x Steps

4.6.4 Comparativo entre modelos

Tal como descrito na Seção 4.2.1, vamos analisar de forma comparativa o os modelos. É possível notar que a convergência para um mesmo tempo $t = 13M$ do modelo *PPO*, forneceu maior valor de recompensa em relação ao algoritmo genético. Além disso, é possível notar que, fixando um valor $\delta = -85$ de recompensa, o modelo *PPO* atinge δ mais rápido que o modelo de algoritmo genético e que o modelo *REINFORCE*. Acompanhe a Figura 34.

Dando continuidade à pauta com respeito a variância, acompanhe os gráficos de recompensa em perspectiva na Figura 35. Novamente é possível notar as oscilações mais abruptas no modelO *REINFORCE*.

4.7 *Bipedal Walker - Gym OpenAI*

Bipedal Walker pertence à classe Box2D da API do *Gym*. Basicamente consiste em um agente que tem como objetivo aprender a caminhar e ir o mais longe e mais rápido possível (Figura 36). Para tanto, possui um espaço de ações e observações contínuos. O conjunto de

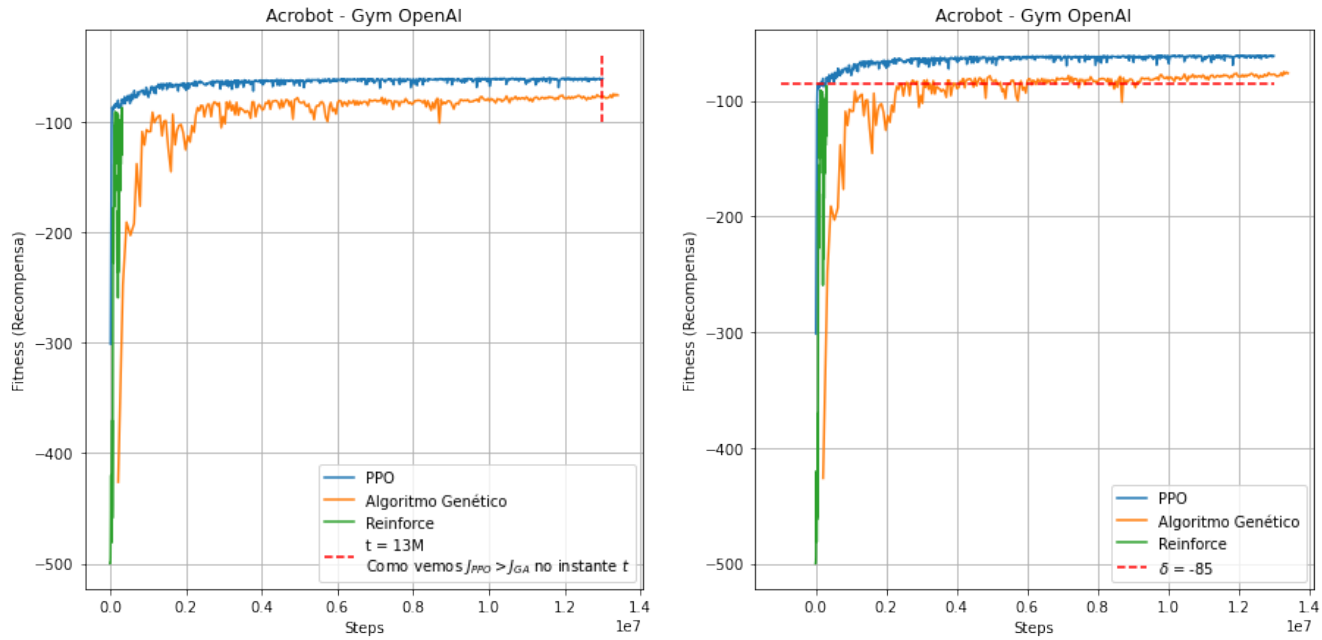

Figura 34 – Comparativo da recompensa x Steps

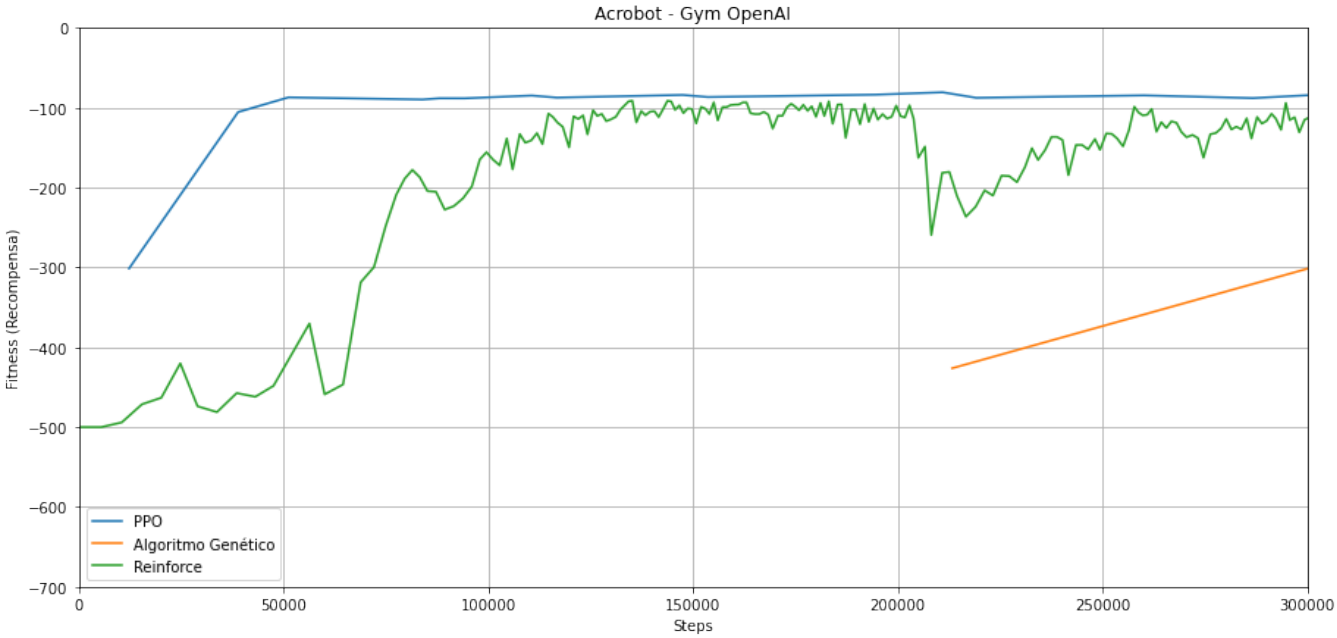

Figura 35 – Comparativo da variância nos gráficos de recompensa para o ambiente *Acrobot*.

ações consiste em velocidade contínua para os quatro motores em cada uma das junções das "pernas" do agente, o valor varia de -1 a 1:

- Velocidade da junção 1: $[-1, 1]$.
- Velocidade da junção 2: $[-1, 1]$.
- Velocidade da junção 3: $[-1, 1]$.
- Velocidade da junção 4: $[-1, 1]$.

O conjunto de observações consiste em 24 valores:

- Velocidade angular do casco.
- Velocidade angular do corpo.
- Velocidade vertical.
- Velocidade horizontal.
- Posição das juntas.
- Velocidade angular das juntas.
- Contato das pernas com o solo.
- 10 medições de um Lidar.

O agente recebe uma recompensa de +300 caso chegue à linha de chega, se ele cair recebe −100 e se aplicar toque recebe pequenos descontos de recompensa.

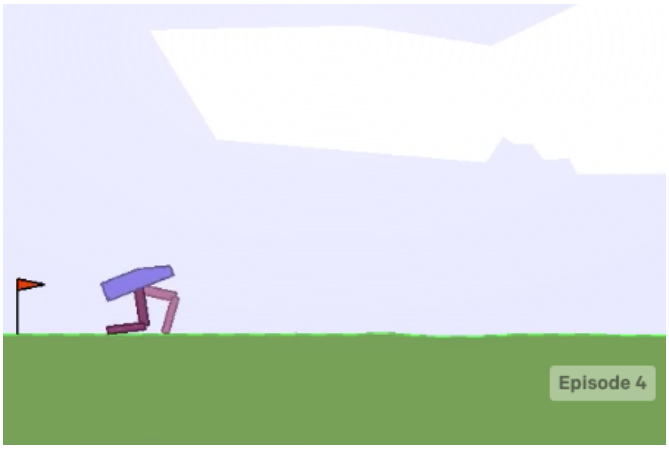

Figura 36 – *Bipedal Walker - Gym OpenAI*

4.7.1 PPO

4.7.1.1 Recompensa Média

Apesar do tempo simulação exigir um esforço considerável quando comparado com modelos mais simples, é possível treinar de forma bastante prática utilizando o modelo *PPO*. Acompanhe a Figura 37.

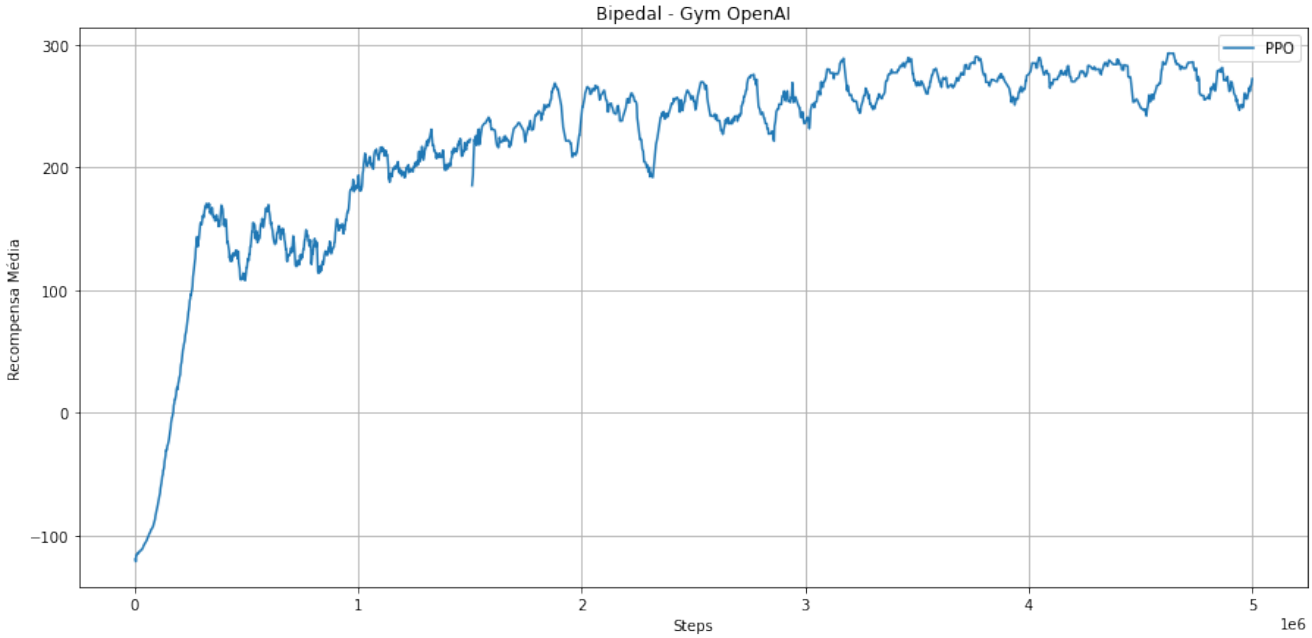

Figura 37 – Recompensa média do treinamento.

4.7.2 Tamanho médio de episódio

Para o tamanho médio do episódio, o início da simulação marca que o agente, por ainda estar aprendendo a tomar decisões no ambiente, possui o tamanho do episódio maior. Aos poucos, o agente passa a aprender a percorrer todo o trajeto em tempos cada vez menores. Acompanhe a Figura 38.

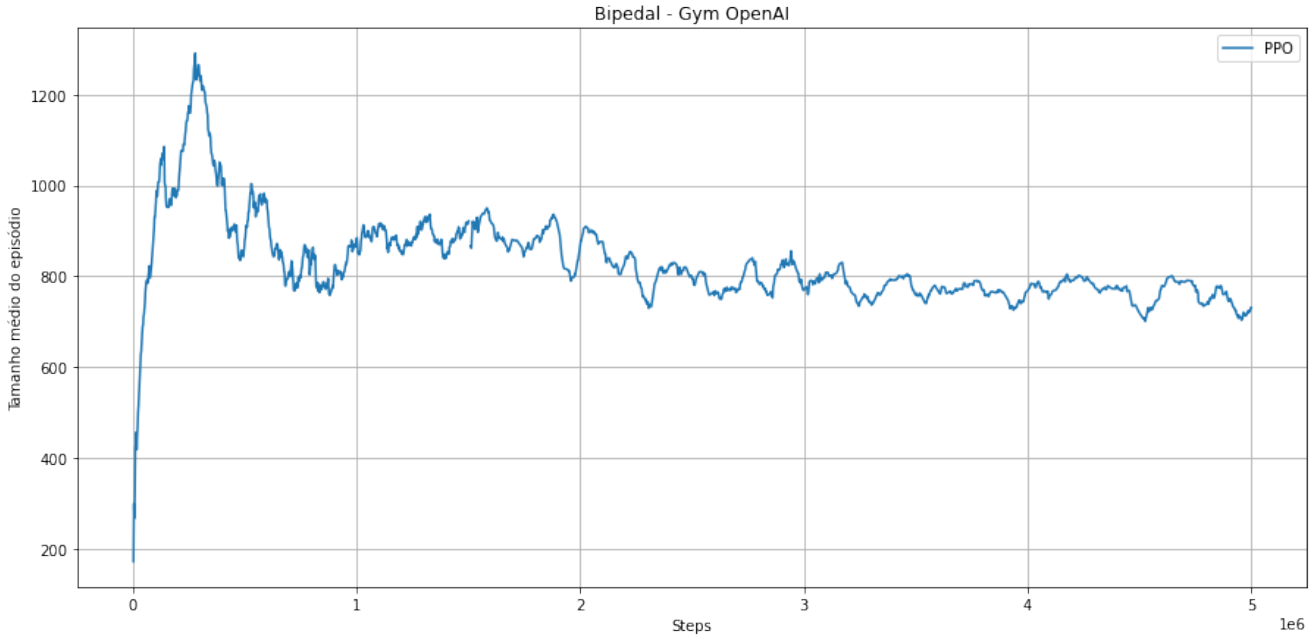

Figura 38 – Tamanho médio do episódio no treinamento.

4.7.3 Algoritmo Genético

4.7.3.1 Recompensa Média

Muitas vezes utilizar modelos evolucionários pode não ser o suficiente para treinar uma *policy*. Isso ocorre, sobretudo, em casos cuja complexidade é razoavelmente significativa, que é o caso do presente ambiente. Nesse sentido, não é possível concluir a tarefa de treinamento da *policy*, porém, é possível notar que o algoritmo genético encontra uma solução local. Como mencionamos na descrição do ambiente, o agente perde muitos pontos caso ele caia com o casco no chão. A solução encontrada pelo algoritmo é que o agente abra as pernas e permaneça com seu centro de massa o mais próximo do chão possível, para assim, evitar o tombamento e a consequente perda exagerada de pontos. Portanto, estamos diante de um caso cujo a solução por meio do algoritmo genético é local e pouco vantajosa. Acompanhe a Figura 39.

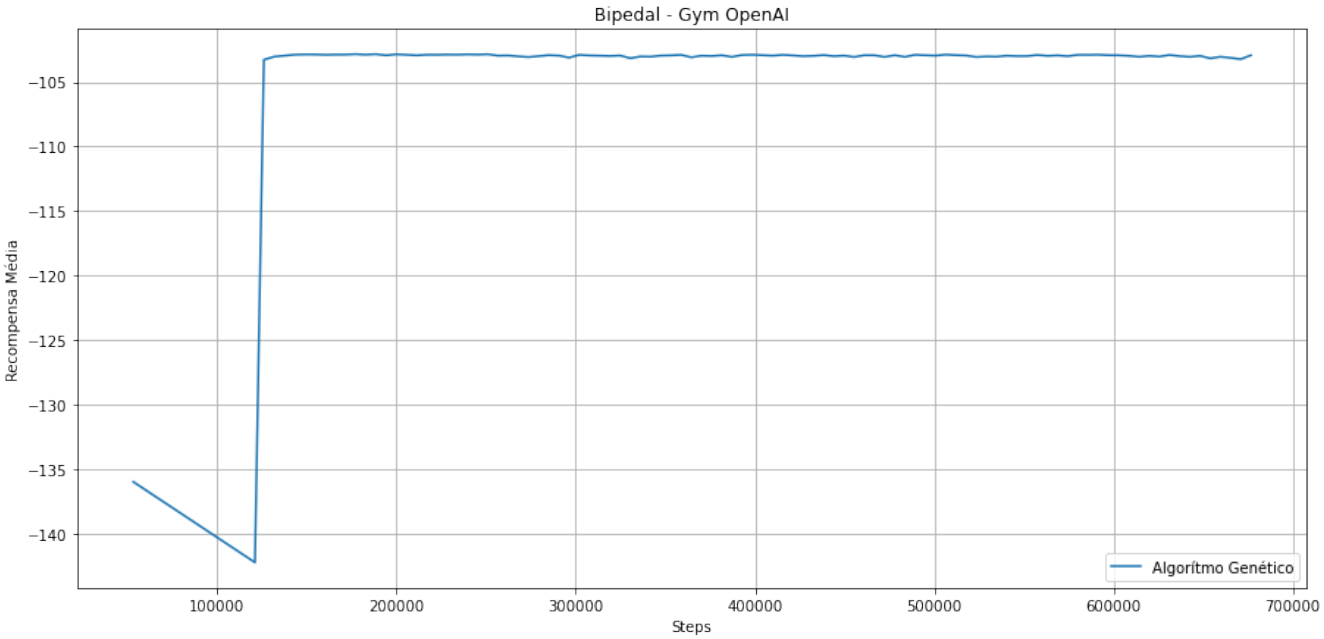

Figura 39 – Recompensa média do treinamento.

4.7.4 Tamanho médio de episódio

Da mesma forma, para o tamanho médio do episódio, encontraremos uma solução local que não treina o nosso agente para a realização da tarefa, tal e qual o modelo *PPO*. Acompanhe a Figura 40.

4.8 *Curriculum Learning*

Tal como discutimos a respeito do *Curriculum Learning* na Seção 4.3, vamos avaliar a performance do agente em ambientes de tal forma que uma tarefa principal seja dividida

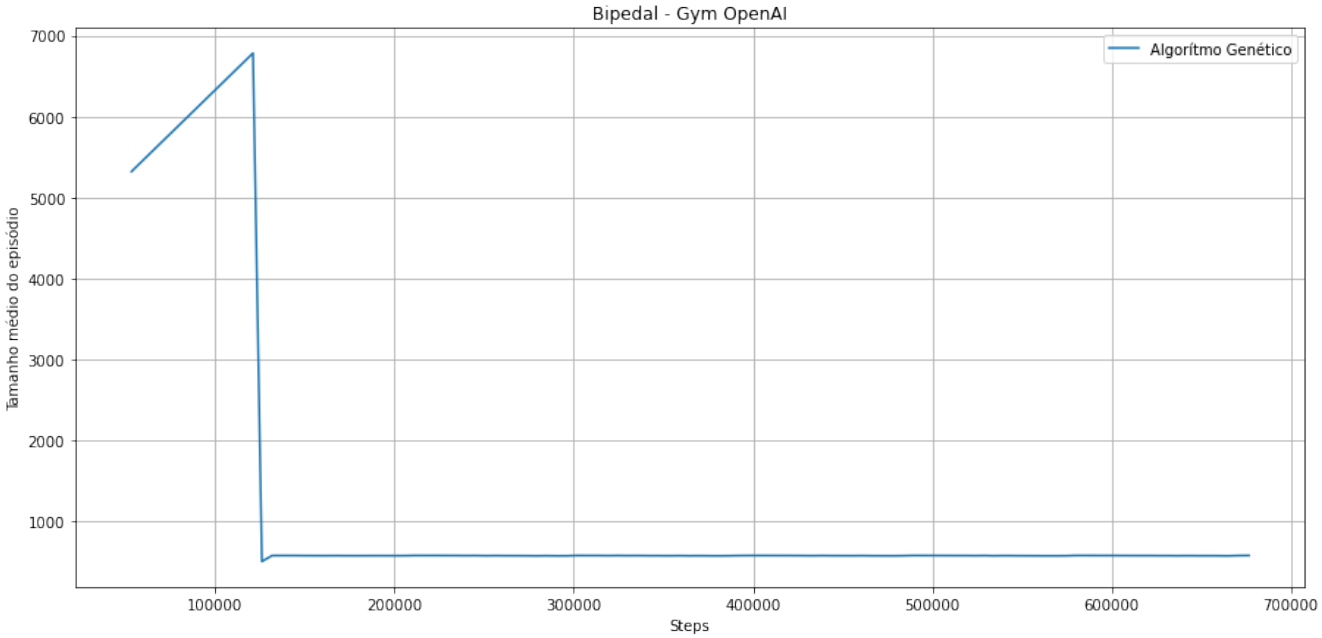

Figura 40 – Tamanho médio do episódio no treinamento.

em sub-tarefas menores. Dessa forma, vamos avaliar como isso impacta no aprendizado do agente. Vamos utilizar, inicialmente, o ambiente do *LunarLander*, citado na Seção 4.5. No referido ambiente, existe a possibilidade de adicionarmos força do vento no agente, assim, dificultando o pouso da espaço nave. Nesse sentido, vamos variar o poder de força do vento e o poder de turbulência, com o intuito de ir cada vez mais dificultando o pouso. Vamos elaborar o seguinte:

1. **Sem *curriculum***: Tarefa única, força do vento em 15 (varia de 0 a 20) e força de turbulência em 1.5 (varia de 0 a 2).
2. **Com *curriculum***:
    a) Task 1: Força do vento em 5 e turbulência me 0.5
    b) Task 2: Força do vento em 10 e turbulência me 1
    c) Task 3: Força do vento em 15 e turbulência me 1.5

Se tomarmos como análise a Figura 41, é possível observar a queda natural de recompensa no momento em que trocamos a tarefa. Isso se deve ao fato de que a *policy* precisa se adaptar às novas condições do ambiente, que nesse caso é ao aumento da força do vento e da turbulência da aeronave. É possível notar, também, que o aprendizado foi mais gradativo do que para o caso sem *curriculum*. Isso se deve, sobretudo, à divisão gradativa das tarefas que colocamos para a *policy* realizar. Por fim, é possível ver que o valor convergiu para um patamar parecido, do ponto de vista da recompensa média. Porém, podemos ver que do ponto de vista do tamanho médio do episódio, na Figura 42, o tempo de pouso da aeronave se tornou menor para o caso em que inserimos o *curriculum*.

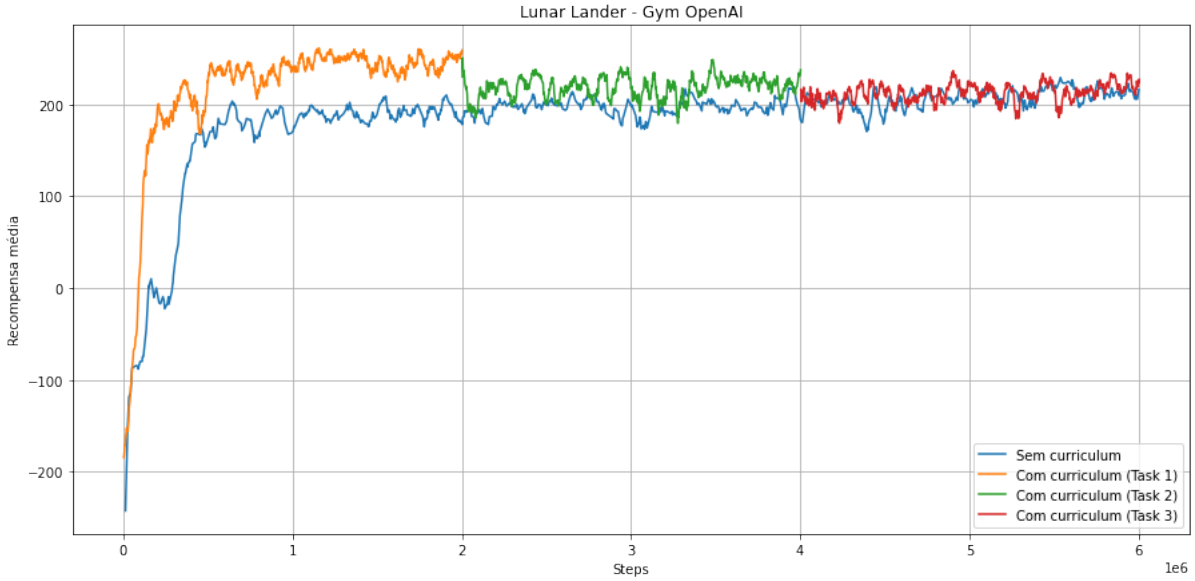

Figura 41 – Comparativo de uso do *Curriculum Learning* com respeito à recompensa média.

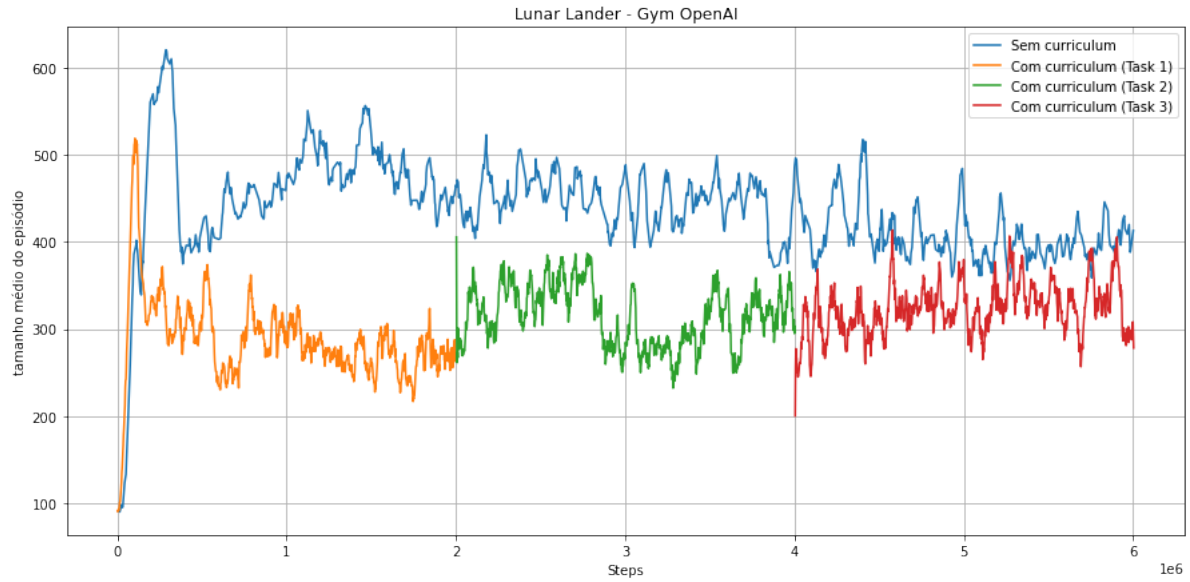

Figura 42 – Comparativo de uso do *Curriculum Learning* com respeito ao tamanho médio do episódio.

Por critérios comparativos, vamos avaliar, a mesma situação anterior, porém com apenas duas tarefas. Sobretudo, é importante fazer esse tipo de análise pois torna o aprendizado menos gradativo do que poderia ser, haja vista o fato de a nova tarefa ter um grau de dificuldade significativamente maior que a anterior. Nesse sentido, vamos variar mais a força do vento e a turbulência.

1. **Sem *curriculum***: Tarefa única, força do vento em 20 (varia de 0 a 20) e força de turbulência em 2.0 (varia de 0 a 2).

2. **Com *curriculum***:

a) Task 1: Força do vento em 5 e turbulência me 0.5

b) Task 2: Força do vento em 20 e turbulência me 2.0

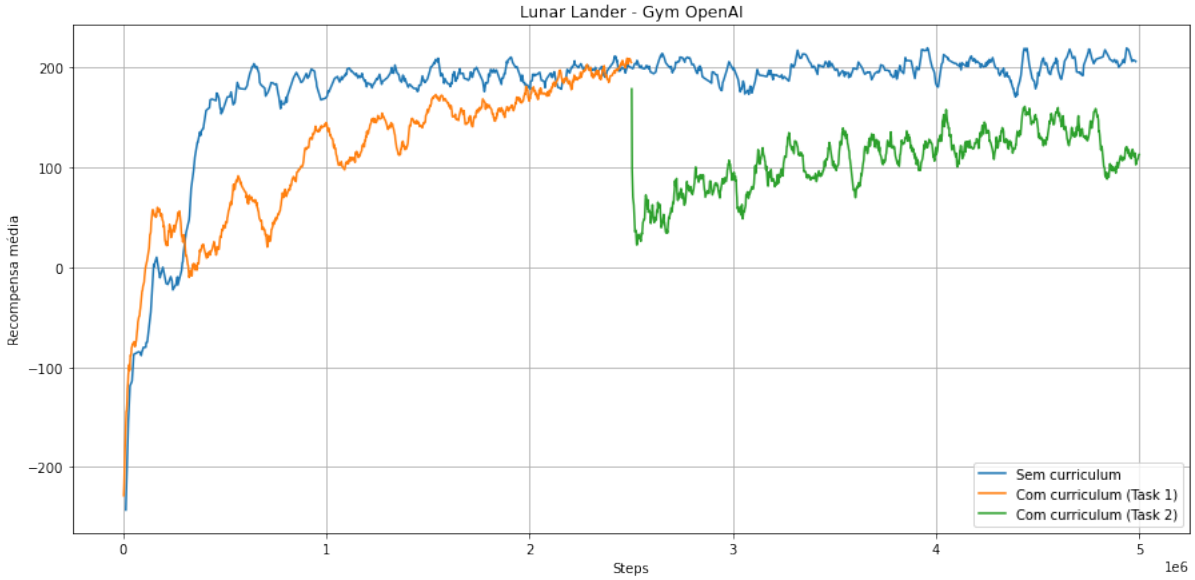

Figura 43 – Comparativo de uso do *Curriculum Learning* com respeito à recompensa média com 2 tasks.

Figura 44 – Comparativo de uso do *Curriculum Learning* com respeito ao tamanho médio do episódio com 2 tasks.

Acompanhando a Figura 43, é possível nota que o aprendizado por meio do *Curriculum*, além de ter tido uma queda significativa ao trocar de tarefa, não houve sinais de vantagens no treinamento, do ponto de vista de passos de simulação. Isso se deve, sobretudo, ao fato de que o agente que passou pelo *Curriculum* não alcançou o agente que não passou pelo *Curriculum*.

5 CONCLUSÕES E TRABALHO FUTURO

Diante de toda análise citada no trabalho presente, é possível traçar boas conclusões com respeito à comparação entre os modelos citados na Seção 3. As estratégias presentes em cada um dos algoritmos podem ser evidenciadas e avaliadas de tal forma a expor suas vantagens e desvantagens. Como visto na construção do modelo *PPO* em 3.2, treinar *policies* impondo limites de divergência em suas atualizações pode trazer grandes benefícios do ponto de vista do desvio padrão e instabilidade de treinamento quando comparados com os modelos genético e *REINFORCE*.

Em todas as nossas análises, registradas na seção 4, o desempenho se mostrou maior quando utilizado o modelo *PPO*. Nesse sentido, as desvantagens dos modelos genético e *REINFORCE* se tornam mais evidentes. Utilizar o algoritmo genético pode necessitar de muito custo do ponto de vista computacional devido à grande utilização de memória para que cada geração seja criada. Porém, é possível que utilizando métodos de processamento paralelo exista uma compensação de desempenho quando comparado com outros modelos em relação ao tempo real de processo.

Ademais, o modelo *REINFORCE*, apesar de ser relativamente fácil de se implementar, tal e qual o genético, apresenta uma grande desvantagem do ponto de vista do desvio padrão. Isso pode ser evidenciado pelas análises feitas na seção 4. Em alguns casos, além da grande instabilidade da *policy*, é possível ver que mesmo depois de treinada pode ocorrer fatores que destroem o aprendizado da *policy* como evidenciado em 4.6 e em 4.4.

Além disso, a investigação feita a respeita da utilização do *Curriculum Learning* (Seção 4.3) trás bons resultados quanto a efetividade da formação do *Curriculum*. Tomando como base o *environment LunarLander*, podemos ver que o aumento gradativo das *tasks* pode contribuir pouco ou até prejudicar o aprendizado da *policy*. Diante dos dois casos, podemos ver que a baixa gradação da dificuldade do *environment* pode fazer com que o aprendizado não se consolide quando comparado ao caso sem utilização do *Curriculum*. Já o caso em que houve uma maior gradação de dificuldades, é possível analisar pouca contribuição do ponto de vista da recompensa acumulada. Nesse sentido, os agentes atingiram praticamente o mesmo patamar de recompensa em praticamente o mesmo tempo de simulação.

5.1 Trabalhos Futuros

Nesse sentido, é possível observar uma enorme quantidade de trabalhos que fazem sentido serem desenvolvidos. Em primeiro lugar, a análise de tipos diferentes de algoritmos e

modelos, que consta no presente trabalho, nos trás a capacidade de entender quais são os pontos fortes e fracos de cada um deles. Nesse aspecto, tendo ciência de todas essas características, temos respaldo para indicar e propor soluções para os algoritmos. A exemplo disso, podemos propor a adaptação de algoritmos que utilizam o gradiente da *policy* com aspectos genéticos. Ou seja, utilizar o gradiente para sabermos em qual direção a maior recompensa pode estar e aplicar atualizações dos parâmetros da *policy* com operadores genéticos. Essa pode ser uma solução viável tanto para o tempo de longa duração do algoritmo genético puro, quanto para o problema da grande variância do modelo *REINFORCE* com ou sem baseline.

Além disso, é possível notar que a utilização do *Curriculum Learning* só é efetiva quando condicionada à distribuição gradativa das *tasks*. Isso pôde ser notado com os gráficos que analisamos na seção 4.3. Sob esse aspecto, vale ressaltar a necessidade de maiores estudos quanto a técnica que precisa ser utilizada para distribuir de maneira efetiva as *tasks*.

Ademais, é de essencial importância que seja explorada a viabilidade de aplicar alguns desses modelos em ambientes reais. No presente trabalho, utilizamos ambientes simulados provenientes da API *Gym* da *OpenAI* e do *Unity*, porém é possível utilizar o modelo simulado para treinar a *policy* e aplicar em um modelo real, tal como o pêndulo invertido.

REFERÊNCIAS

AMARI, S. ichi. Backpropagation and stochastic gradient descent method. *Neurocomputing*, v. 5, n. 4, p. 185–196, 1993. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/092523129390006O>. Citado na página 20.

GéRON, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. [S.l.]: Oreilly, 2019. v. 2nd. Citado na página 6.

KINGMA, J. B. D. P. Adam: A method for stochastic optimization. *Arxiv*, v. 1, n. arXiv:1506.02438, 2015. Disponível em: <https://arxiv.org/abs/1506.02438>. Citado 2 vezes nas páginas 20 e 23.

MNIH, V.; BADIA, A. P.; MIRZA, A. G. M.; LILLICRAP, T. P.; HARLEY, D. S. T.; KAVUKCUOGLU, K. Asynchronous methods for deep reinforcement learning. *arXiv*, v. 1, n. 1602.01783, p. 0–19, 2016. Disponível em: <https://arxiv.org/pdf/1602.01783.pdf>. Citado 2 vezes nas páginas 20 e 22.

MNIH, V.; SILVER, K. K. nad D.; GRAVES, A.; ANTONOGLOU, I.; WIERSTRA, D.; RIEDMILLER, M. Playing atari with deep reinforcement learning. *arXiv*, v. 1, n. 1602.01783, p. 0–19, 2013. Disponível em: <https://arxiv.org/pdf/1312.5602v1.pdf>. Citado na página 6.

NARVEKAR, S.; PENG, B.; LEONETTI, M.; SINAPOV, J.; TAYLOR, M. E.; STONE, P. Curriculum learning for reinforcement learning domains: A framework and survey. *arXiv*, v. 1, n. 2003.04960, p. 0–50, 2020. Disponível em: <https://www.jmlr.org/papers/volume21/20-212/20-212.pdf>. Citado na página 30.

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. *arXiv*, v. 1, n. 1707.06347, p. 0–12, 2017. Disponível em: <https://arxiv.org/pdf/1707.06347.pdf>. Citado 2 vezes nas páginas 19 e 43.

SCHULMAN PHILIPP MORITZ, S. L. M. J. P. A. J. High-dimensional continuous control using generalized advantage estimation. *Arxiv*, v. 1, n. arXiv:1506.02438, 2015. Disponível em: <https://arxiv.org/abs/1506.02438>. Citado na página 20.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018. Citado na página 23.

WILLIAMS, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Neurocomputing*, v. 8, n. 3, 1992. ISSN 0925-2312. Disponível em: <https://doi.org/10.1007/BF00992696>. Citado 2 vezes nas páginas 18 e 22.

WU, C.; RAJESWARAN, A.; DUAN, Y.; KUMAR, V.; BAYEN, A. M.; KAKADE, S.; MORDATCH, I.; ABBEEL, P. Variance reduction for policy gradient with action-dependent factorized baselines. *arXiv*, v. 1, n. 1803.07246, p. 0–16, 2018. Disponível em: <https://arxiv.org/pdf/1803.07246.pdf>. Citado na página 18.