UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MARIA EDUARDA NOTHEN RUHE

Avaliação da navegação sem mapeamento para exploração de ambientes internos desconhecidos por múltiplos robôs móveis autônomos usando Aprendizagem por Reforço

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Edison Pignaton de Freitas

Porto Alegre
2023

RESUMO

Em aplicações de exploração de regiões desconhecidas com utilização de robôs, existe uma grande preocupação de que o método utilizado nos projetos não seja apenas eficiente, mas também demande o menor custo computacional para realizar a tarefa. A maneira mais tradicional de resolver esse problema seria primeiro mapear a região utilizando os robôs e, conforme o mapa da região é criado, a partir do conhecimento adquirido pelos robôs decidir qual direção deve seguir. Porém, esse tipo de método conforme o tamanho da região e número de robôs aumentam significamente o custo de memória e computacional. Desta forma, algoritmos de exploração com multi-robôs de forma que não necessite de um mapa ganharam notoriedade nos últimos anos. Neste trabalho, temos como objetivo avaliar se as técnicas de Aprendizado por Reforço permitem estratégias de exploração eficientes para aplicações com múltiplos robôs em ambientes não mapeados. Para isso, utilizamos o método otimização de política proximal, que é um algoritmos de aprendizado profundo, na qual resultou em bom resultados para o problema de exploração sem mapa. Também focamos de como o número de robôs modifica a qualidade dos resultados utilizando o mesmo método de aprendizado. Para isso, foram criados estruturas de comunicações entre os robôs de modo que o aumento do número de robôs, não cause um aumento significativo no custo de memória ou processamento computacional.

Palavras-chave: Robótica Móvel. Aprendizagem por Reforço Profundo. Sistemas Multi-Rôbos. Exploração de Ambientes Desconhecidos. Exploração Cooperativa.

Maria Eduarda's TCC Title

ABSTRACT

In applications for exploring unknown regions using robots, there is a great concern that the method used in the projects is not only efficient, but also requires the lowest computational cost possible to carry out the task. The most traditional way to solve this problem would be to first map the region using the robots and, as the map of the region is created, based on the knowledge acquired by the robots, decide which direction to follow. However, this type of method, depending on the size of the region and number of robots, significantly increases memory and computational costs. In this way, multi-robot exploration algorithms that use mapless navigation have gained notoriety in recent years.In this work, we aim to evaluate whether Reinforcement Learning techniques allow efficient exploration strategies for applications with multiple robots in unmapped environments. For this, we use the proximal policy optimization method, which is a deep learning algorithm, which has been studied with good results for the mapless exploration problem. We also focus on how the number of robots modifies the quality of the results using the same learning method. To achieve this, communication structures were created between the robots so that the increase in the number of robots does not cause a significant increase in the cost of memory or computational processing.

Keywords: Mobile Robotics, Deep Reinforcement Learning, Multi-Robot Systems, Unknown environment exploration, Cooperative exploration .

LISTA DE FIGURAS

2.1 A interação agente-ambiente (SUTTON, 2018) em um processo de decisão de Markov. 11

3.1 Diagrama simplificado do processo de exploração para N Robôs. 19
3.2 a)Esquema representando como as medições do laser 360 espaçadas por um grau são igualmente dividido em oito zonas de proximidade. b) Escala que ilustra o estado de proximidade de cada zona pode assumir dependendo das distâncias medidas. 20
3.3 Representação de comunicação de trajetórias de múltiplos robôs, pelo ponto de vista do robô R1 no tempo t. 24
3.4 Representação vetorial de estado do robô R1 no tempo t, considerando que a comunicação foi estabelecido com o robô R2 e o número total de robôs é igual a dois. 25
3.5 Esquema simplificado de como os dados são tratados durante treinamento PPO 26
3.6 Esquema simplificado da estrutura de inferência após a conclusão do treinamento. Política de Exploração. 26

4.1 Mapas utilizados para validação da política de exploração. 28
4.2 Métricas de treinamento de dois agentes no Mapa Simples aprendendo a evitar colisões 31
4.3 Métricas de treinamento de dois agentes no Mapa Simples aprendendo exploração eficiente do mapa com reforço 32
4.4 Métricas de treinamento de dois agentes no Mapa Simples aprendendo exploração eficiente do mapa sem reforço 33
4.5 Métricas de treinamento de dois agentes no Mapa Intermediário aprendendo a evitar colisões 34
4.6 Métricas de treinamento de dois agentes no Mapa Intermediário aprendendo exploração eficiente do mapa com reforço 35
4.7 Métricas de treinamento de dois agentes no Mapa Intermediário aprendendo exploração eficiente do mapa sem reforço 36
4.8 Métricas de treinamento de dois agentes no Mapa Intermediário aprendendo exploração cooperativa, ambos com reforço 37
4.9 Métricas de treinamento de dois agentes no Mapa Intermediário aprendendo exploração cooperativa, um agente com reforço 38
4.10 Aplicação executando dois casos de exploração de mapas com dois conjuntos diferentes de agentes 39
4.11 Treinamento cooperação no Mapa Intermediários para diversos conjuntos de robôs 40
4.12 Treinamento cooperação para diversos conjuntos de robôs com Mapa Intermediário proporcional ao número de robôs 42
4.13 Treinamento cooperação no Mapa Teste 1 com 2 robôs 44
4.14 Treinamento cooperação no Mapa Teste 1 com múltiplos robôs 45

LISTA DE TABELAS

4.1 Mapas e suas Dimensões 27
4.2 Tabela dos Hiperparâmetros 29
4.3 Comparação do teste para o Mapa Intermediário com múltiplos robôs 41
4.4 Comparação do segundo teste para o Mapa Intermediário com múltiplos robôs 43
4.5 Comparação do teste para o Mapa Teste1 com múltiplos robôs 45
4.6 Comparação do segundo teste para o Mapa Teste1 com múltiplos robôs 46
4.7 Comparação do teste para o Mapa Teste2 com múltiplos robôs 46
4.8 Comparação do segundo teste para o Mapa Teste2 com múltiplos robôs 47

LISTA DE ABREVIATURAS E SIGLAS

RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
MRS	Multi-Robots System
PPO	Proximal Policy Optimization
MARL	Multi-Agent Reinforcement Learning
LR	Learning Rate

SUMÁRIO

1 Introdução **9**

2 Aprendizagem por Reforço **11**
2.1 O que é Aprendizagem por Reforço? 11
2.2 Otimização de Política Proximal 13
2.2.1 Método de *Policy Gradient* 14
2.2.2 Algoritmo PPO 14
2.3 Utilização de algoritmos RL em sistema Multi-Robôs 16

3 Sistemas de Múltiplos Robôs em Exploração de Mapas **18**
3.1 Robô e seus componentes 18
3.1.1 Estruturas 18
3.1.2 Movimentação 20
3.1.2.1 Sensores 20
3.1.2.2 Recompensa 21
3.2 Sistema de Múltiplos Robôs 22
3.2.1 Compartilhamento de trajetórias entre Robôs 23
3.2.2 Tomada de decisões 24

4 Experimentos e Resultados **27**
4.1 Mapas de Teste 27
4.2 Hiperparâmetros 29
4.3 Método de Validação Utilizando Dois Robôs 29
4.3.1 Mapa Simples 30
4.3.2 Mapa Intermediário 33
4.4 Exploração com n Robôs 38
4.4.1 Mapa Intermediário 39
4.4.2 Mapa Teste 1 43
4.4.3 Mapa Teste 2 46

5 Conclusão **48**

Referências **49**

1 INTRODUÇÃO

Com o passar dos anos, a perspectiva do que pode ser alcançado em aplicações com robôs móveis vem crescendo em múltiplas áreas. Entre elas pode-se citar diversos empregos em empresas, indústrias, hospitais, segurança pública, agricultura e residências como forma de melhorar os serviços e atividades diárias. Algumas destas áreas envolvem aplicações com ênfase em exploração de uma área de maneira rápida e eficiente, como serviços de busca e salvamento e resgate (SAR), missões de reconhecimento e explorações planetárias (ALATISE, 2020).

O método mais simples de abordar problemas que envolvem exploração de ambientes desconhecidos utilizando robôs autônomos é criar um mapa conforme os robôs vasculham a região, para após a exploração realizar a tomada de decisão de qual é a melhor forma de prosseguir utilizando o mapa formado. Isso causa um problema muito grave, já que quanto maior a área de exploração, maior deve ser a memória e poder computacional para conter e processar toda informação adquirida pelo robô. O problema pode ser ainda maior se for múltiplos robôs, que devem repetidamente mandar e receber informações uns para os outros, para se manterem sincronizados.

Por isso, uma forma que se possa explorar uma região tomando boas decisões em cada etapa, porém sem ter que se preocupar com o crescimento da demanda computacional, é algo de grande importância para o avanço na área de exploração com robôs autônomos. Contudo isso não é uma tarefa fácil, principalmente em ambientes onde o mapa da região é obscuro. Conforme descrito em trabalhos passados,focados em aplicações reais com robôs móveis, “O emprego de robôs móveis em contextos tão complicados depende de uma abordagem robusta e estratégia de exploração eficiente.”(GARAFFA, 2021)

Atualmente, a utilização de *deep neural networks* se tornou uma grande aliada para algoritmos RL (*Reinforcement Learning* ou Aprendizagem por Reforço) para resoluções e otimização de inúmeros problemas complexos e com numerosas entradas. Não é coincidência que o número de pesquisas e propostas de soluções utilizando algoritmos RL tenha aumentado significativamente nos últimos anos e algoritmos de aprendizagem por reforço profundo (DRL) em trabalhos utilizando exploração de ambientes desconhecidos. Porém, vale lembrar que, apesar do aumento do número de pesquisa, utilização de DRL como solução eficiente para exploraçãoo de ambientes desconhecidos sem uma meta de posição pré-determinada ainda é uma área que pode ser mais explorada (GARAFFA, 2021).

Este trabalho tem como objetivo avaliar se as técnicas de Aprendizado por Reforço permitem estratégias de exploração eficientes para aplicações com múltiplos robôs em ambientes não mapeados. Também será visto como e o quanto o número de robôs influência na robustez e eficácia do algoritmo de aprendizagem escolhido no projeto.

Este texto está organizada da seguinte forma: Capítulo 2 contém uma breve explicação sobre Aprendizagem por Reforço junto com a situação atual de como está sendo utilizada em projetos com robôs móveis. Também entraremos mais a fundo no algoritmo de aprendizagem Otimização de Política Proximal que é o principal método utilizado neste trabalho. Capítulo 3 apresenta os dois componetes do ambiente que será utilizado o método: os mapas utilizados nos testes e os robôs que terão como objetivo aprender como explorá-los. Capítulo 4 apresenta os resultados dos testes utilizando diferentes mapas e números de robôs e uma análise sobre esses resultados. Por fim, o Capítulo 5 contém a conclusão do trabalho e sugestões para próximas pesquisas.

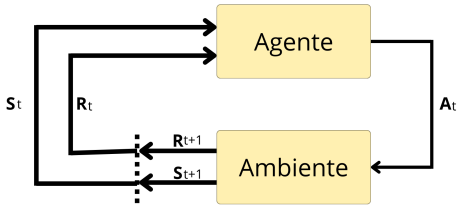
2 APRENDIZAGEM POR REFORÇO

2.1 O que é Aprendizagem por Reforço?

Aprendizagem por Reforço (RL) é, assim como outras formas de aprendizado de máquina, se considerarmos que é simultaneamente um problema, uma classe de soluções que resolvem o problema e um campo que estuda o problema e seus métodos. Desta maneira um problema de RL envolve o que e como fazer o mapeamento das possíveis soluções com a finalidade de maximizar o valor considerado a recompensa para a situação presente no momento. Porém, em vez de ser informado de quais ações devem ser tomadas, a maneira como o sistema descobre quais ações geram a maior recompensa é ao experimentá-las, ou seja, por tentativa e erro. Além disso, as recompensas não são apenas imediatas, mas também afetam o ambiente e, consequentemente, as recompensas futuras da forma que ele pode ser representado por um Processo de Decisão de Markov.

Descrevemos o modelo sequencial de *Processo de Decisão de Markov* (MDP) representa, como o nome sugere, uma forma de tomada de decisão de um agente em um ambiente. No trabalho de (PUTERMAN, 1994), o modelo é descrito da maneira a seguir. Em um determinado momento, um tomador de decisão, agente ou controlador observa o estado de um sistema. Com base neste estado, o tomador de decisão escolhe um Ação. A escolha da ação produz dois resultados: o tomador de decisão recebe uma recompensa imediata (ou incorre em um custo imediato), e o sistema evolui para um novo estado em um momento subsequente de acordo com uma distribuição de probabilidade determinada pela escolha da ação, onde:

Figura 2.1: A interação agente-ambiente (SUTTON, 2018) em um processo de decisão de Markov.



- S representa um conjunto finito de estados $(s_t \in S)$.
- A representa um conjunto finito de ações $(a_t \in A)$.
- T representa o tamanho da trajetória $t \in T$.
- R representa a função de recompensa, onde $R(s_t, a_t, s_{t+1})$ é a recompensa imediata por transitar do estado s_t para o estado s_{t+1}, após a ação a_t.

A tupla $< S, A, T, R >$ é o problema de RL formalizado matematicamente, porém ainda existem algumas variáveis que merecem ser realçadas, para melhor compreensão do problema. Uma delas é P, que representa a probabilidade de uma transição associado a um estado específico $P(s_{t+1}||s_t, a)$. Como o sistema atende as demandas do MDP, se soubermos os estados passados ao estado atual (S_t), podemos descobrir matematicamente o valor de P com a Equação 2.1.

$$P(s_{t+1} \mid s_t) = P(s_{t+1} \mid s_0, s_1, ..., s_t) \tag{2.1}$$

Outra variável, definida na Equação 2.2, é a trajetória, também chamada de episódio (tau), que é a sequencia de estado (s), ação (a) e recompensa (r) que resulta da ação entre o agente e ambiente conforme representado na 2.1. A Policy (π) é uma função que mapeia o espaço de estado no espaço de ação. Ela pode ser *determinística* ($\pi(s_t)$) ou *estocástica* ($\pi(a_t|s_t)$), sendo determinística retorna uma ação específica para um estado específico, enquanto estocástico gera uma distribuição de probabilidade de ação.

$$\tau_t = (s_0, a_0, r_1, s_1, a_1, r_2, ..., s_t, a_t, r_{t+1}) \tag{2.2}$$

Com todas variáveis apresentadas, o agente tem como objetivo maximizar a expectativa do retorno acumulativo de episódios ($R(\tau)$), ou a recompensa acomulada durante a trajetória. Se considerarmos T o comprimento da trajetória, $\gamma \in [0, 1)$ um fator de desconto e r_t a recompensa na etapa do tempo t, $R(\tau)$ pode ser representado pela Equação 2.3. Se considerarmos a tanto a probabilidade de transição (P), quanto a policy ($\pi(a_t)$) são estocásticas, a probabilidade de uma específica trajetória com T etapas representada pela Equação 2.4.

$$R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t \tag{2.3}$$

$$P_\pi(\tau) = \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t) \tag{2.4}$$

Assim, a **expectativa de retorno** ($J(\pi)$) é definida pela Equação 2.5, que é definida pela integral do produto da probabilidade de transição e retorno acumulativo de episódios. E, por fim, a Equação 2.6 apresenta a política ideal que maximiza a expectativa do retorno acumulativo.

$$J(\pi) = \int_{\tau} P_{\pi}(\tau) R(\tau) = \mathbb{E}_{\pi}[R(\tau)] \tag{2.5}$$

$$\pi^* = argmax_{\pi}(J(\pi)) \tag{2.6}$$

Sabendo a política π realizada num estado s, temos como definir a o retorno esperado de uma trajetória que inicia no estado s, também conhecida como **valor do estado**. Uma função que consegue medir a qualidade de um estado valor, ou **função de valor do estado** ($V^{\pi}(S)$), pode ser representada conforme a Equação 2.7. Da mesma forma, uma função que calcula o retorno esperado depois de tomar a ação a no estado s enquanto segue a política π, ou a **função de valor da ação-estado** ($Q^{\pi}(a, s)$), é representada pela Equação 2.8. Assim, a diferença entre essas duas funções geram a **função de vantagem** que, como o nome implica, estima o quanto vantajosos é tomar a específica ação, num especifico estado, comparando com a ação/estado atual, definida de acordo com a Equação 2.9. Essas três funções são úteis, no processo de encontrar a política ideal.

$$V^{\pi}(S) = \mathbb{E}_{\tau\pi}[R(\tau)|s_0 = s] \tag{2.7}$$

$$Q^{\pi}(a, s) = \mathbb{E}_{\tau\pi}[R(\tau)|s_0 = s, a_0 = a] \tag{2.8}$$

$$A^{\pi}(s, a) = Q^{\pi}(a, s) - V^{\pi}(S) \tag{2.9}$$

2.2 Otimização de Política Proximal

Otimização de Política Proximal(PPO) é um método de *Policy Gradient*, proposto em 2017 por Schulman,John et al. (SCHULMAN FLIP WOLSKI, 2017). O algoritmo PPO foi escolhido neste trabalho pois tende alcançar boas performances em trabalhos modernos de inúmeras diferentes áreas de aplicação, com a vantagem de ser mais robusto a perturbações do que outros algoritmos de aprendizagem por reforço. Uma das perturbações tratadas é as grandes variações que ocorrem no espaço político do método ator-crítico, por pequenas mudanças nos parâmetros subjacentes da rede neural (NN), o algoritmo PPO resolve esse problema limitando as atualizações da rede de políticas. Assim a atualização se baseia na proporção da nova política com a política antiga, a restrin-

gindo a um determinado intervalo. Além disso, para evitar que a função de perda cresça demais, o valor superior é cortado e apenas consideramos o valor inferior. Também foi provado que o PPO fornece um bom equilíbrio entre facilidade de implementação, ajuste dos hiperparâmetros e complexidade de amostra.

2.2.1 Método de *Policy Gradient*

Policy Gradient é um método, na qual, um estimador do gradiente de política é calculado e conectado a um algoritmo estocástico. A Equação 2.10 é a mais comum usada para estimar o gradiente e Equação 2.11 como a função que calcula a redução do gradiente de política, conforme o trabalho de (SCHULMAN FLIP WOLSKI, 2017).

$$\hat{g} = \hat{\mathbb{E}}_t[\nabla_\theta log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{2.10}$$

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t[log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{2.11}$$

Onde:

- π_θ representa a política estocástica.
- $\hat{A}_t$ representa a função de vantagem no tempo t.
- $\hat{\mathbb{E}}_t$ representa a média empírica sobre um lote finito de amostra.

Porém, a Equação 2.11, apresenta o problema de com o tempo criar novas políticas destrutivas, por terem uma grande variação com as passadas. Por isso, o algoritmo PPO, neste trabalho, utiliza Perda da Iteração da Política Conservadora Limitada ($L_{CLIP}(\theta)$) em seu lugar.

2.2.2 Algoritmo PPO

Durante o treinamento, um tamanho fixo de trajetória é salvo na memória. Nesse estado ocorre o *rollout*, também chamado de *game*. Após o término do *rollout*, é atualizado a politica que adota uma ascensão gradiente estocástica do minibatch- dados de treinamento divididos em pequenos lotes, cada lote é um minibatch. O método utilizado para atualizar a política é, como apontado antes, a Perda da Iteração da Política Conser-

vadora ($L_{CPI}(\theta)$), que é descrita pela Equação 2.12

$$L_{CPI}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)} A_t \right] = \hat{\mathbb{E}}_t[r_t(\theta A_t] \tag{2.12}$$

$$L_{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t) \right] \tag{2.13}$$

Onde:

- π_θ representa a politica sob os parâmetros atuais
- $\pi_{\theta old}$ representa a politica sob os parâmetros anteriores, com as vantagens incluídas.
- $r_t(\theta)$ representa a relação da politica entre os dois primeiros itens.
- a_t representa a ação realizada no tempo t
- s_t representa o estado no tempo t
- A_t representa o valor da vantagem no tempo t

A Equação 2.13 é o resultado da adição do hiperparâmetro ϵ para limitar a função de redução.Por fim, a Equação 2.14 representa a função de perda crítica ($L_C(\theta)$) e a Equação 2.15 é a função de perda total ($L_t(\theta)$).

$$L_C(\theta) = MSE(A_t + V_{old} - V_t) \tag{2.14}$$

$$L_t(\theta) = L_{CPI}(\theta) + c_1 L_C(\theta) - c_2 S_{\pi\theta} \tag{2.15}$$

Onde:

- MSE representa o erro quadrático médio.
- V_{old} representa o valor crítico da memória
- V_t representa o valor crítico atual da rede.
- c_1 representa uma constante positiva
- c_2 representa uma segunda constante positiva
- $S_{\pi\theta}$ representa a entropia da política de exploração no estado atual.

Abaixo está a visão geral processo de aprendizagem PPO-Clip, ou PPO com limite, que é o algoritmo implementado neste trabalho.

Algorithm 1 PPO-Clip

Imput: Parâmetros de política inicial θ_0 e parâmetros de função de valor ϕ_0

1: **for** `k = 0,1,2,...` **do**

2: Coletar conjuntos de trajetórias $T_k = \tau_i$, executando política $\pi_k = \pi(\theta_k)$

3: Computar estimadas de vantagens A_t^{π} a partir do valor da função $V_{\phi k}$

4: Usando a subida gradinte estocástica com o o otimizador Adam, atualize a política, maximizando o objetivo do PPO-Clip:

$$\theta_{k+1} =_{\theta} \frac{1}{|T_k|N} \sum_{\tau \in T_k} \sum_{t=0}^{N} min(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), clip(A^{\pi_{\theta_k}}(s_t, a_t), i + \epsilon, i - \epsilon)) \tag{2.16}$$

5: Usando retro-propagação, ajuste a função de valor por regressão no erro quadrático médio:

$$\phi_{k+1} =_{\phi} \frac{1}{|T_k|N} \sum_{\tau \in T_k} \sum_{t=0}^{N} (V_{\phi}(s_t) - R_t)^2 \tag{2.17}$$

6: **end for**= 0

2.3 Utilização de algoritmos RL em sistema Multi-Robôs

O trabalho de Gautam e Mohan (GAUTAM, 2012) define um sistemas multi-robôs (MRS) como um grupo de dois ou mais robôs que são capazes de cooperar e competir entre si para atingir um objetivo específico.Conjunto de características como alto níveis de tolerância a falhas, aumento de eficiência na realização das tarefas, consciência situacional de vários locais e flexibilidade nas operações fez com que pesquisas sobre MRS ganhasse força nos últimos anos. Pois esta se adéqua para diversas aplicações complexas, incluindo exploração de ambientes desconhecidos. Porém, o sucesso de aplicações com múltiplos-robôs depende da concepção de uma estratégia de cooperação adequada, o que é uma tarefa desafiadora.

Desta forma, os algoritmos de Aprendizagem por Reforço se tornam uma solução promissora.Em trabalhos como (HU SHIJI SONG, 2019), cada robô recebe um nome diferente localização de destino baseada em partições Voronoi dinâmicas. Como o DDPG, um algoritmo de RL, pode lidar com espaços de ação, é utilizado para decidir as velocidades lineares e angulares dos robôs, aumentando possibilidades de comportamento e possibilitando movimentos mais suaves. Em (WALKER FERNANDO VANEGAS AL-

VAREZ, 2020), tem o foco na busca de alvos, onde o solucionador POMDP define se o agente deve explorar seu atual nó ou para qual nó ele deve se mover em seguida, e em (CAI; YANG; XU, 2013) o método Hungaro é usado para obter o melhor arranjo de subtarefas cooperativas e distribuí-lo entre os robôs. Em ambos os trabalhos, o algoritmo RL recebe a próxima localização do objetivo e um conjunto de medições do sensor e retorna o movimento dos agentes. Ao contrário de (HU SHIJI SONG, 2019), os espaços de ação são discretos e compostos por 3 e 4 possibilidades de movimentos, respectivamente.

3 SISTEMAS DE MÚLTIPLOS ROBÔS EM EXPLORAÇÃO DE MAPAS

3.1 Robô e seus componentes

No contexto de navegação robótica exploratória, os robôs representam agentes do sistema que têm o objeto de aprender como navegar por um mapa, do qual não se tem nenhum conhecimento prévio, utilizando o algoritmo PPO. Neste trabalho, estamos lidando com uma simulação controlada que é a exploração de um ambiente fechado, sem sinais externos como GPS, e 2D. Em outras palavras, temos como objetos encontrar um conjunto de pontos de referência (x, y) que resulta na cobertura do ambiente mais rápida e de maneira cooperativa. Por isso, um conjunto de estruturas foram projetadas para que o robô possa encontrar as melhores soluções para as situações que ele possa enfrentar.

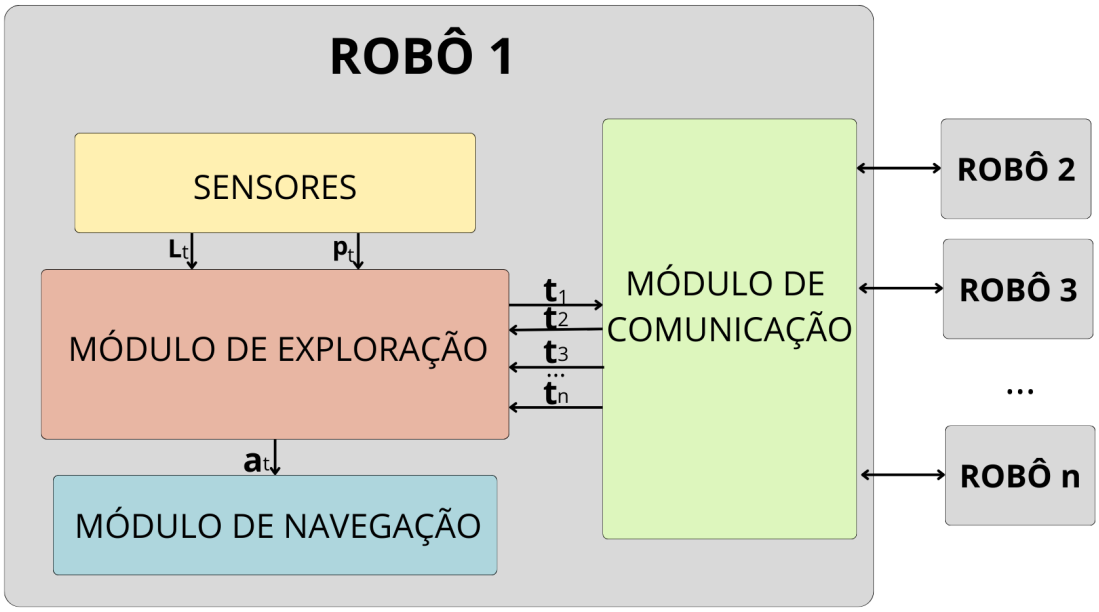
3.1.1 Estruturas

O sistema foi planejado com o intuito de solucionar o problema de exploração colaborativa. A Figura 3.1 é a representação visual do sistema de exploração para **N** robôs. Em sistema de multi-robôs, onde N_R é o número de robôs e $N_R \in \mathbb{N}^*$, todos os robôs do sistema, $R_1, R_2, .., R_n$, são considerados. Para cada agente que compõe a rede de robôs possui as mesmas estruturas: Sensores, Módulo de Exploração e Módulo de Navegação.

A estratégia de exploração aplicada no trabalho é decentralizada, isso significa que cada robô roda o sistema de determina a politica de exploração de maneira individual. Desta forma, os agentes conseguem explorar uma área de maneira independente, porém pode otimizar o processo de sua tomada de decisão ao receber informações colhidas por outros agentes. Além disso, essa estratégia também ocorre independente de qualquer processo de mapeamento.

O fluxo do sistema ocorre nesta ordem. Primeiro os Sensores, no tempo t, manda a leitura dos lasers L_t e a posição atual do robô P_t para o Módulo de Exploração. O Módulo de exploração, atualiza a trajetória do robô com a nova posição t_1 e manda para o Módulo de Comunicação, que tentará transmitir para os outros robôs. O Módulo de Comunicação também pode transmitir as trajetórias dos outros robôs, $t2, t3, ..., tn$ para o Modulo de Exploração, se houver disponibilidade. Com as medidas do Sensores e as trajetórias recebidas, tando dos outros robôs quanto do próprio, o Modulo de Exploração calcula qual

Figura 3.1: Diagrama simplificado do processo de exploração para N Robôs.



a próxima ação do robô a_t e a manda para o Módulo de Navegação. Finalmente, o Modulo de Navegação recebe a ação traduz o abstração de alto nível em variáveis elétricas e mecânicas que controlam o robô para executar a ação selecionado.

O sistema também foi projetado para ser modular, isso significa que os blocos conectados à exploração módulo não estão limitados a um método específico. Desde que algumas suposições determinadas sejam respeitados, os Sensores, o Módulo de Comunicação e o Módulo de Navegação podem usar diferentes implementações para diferentes projetos. Da maneira que foi planejado a cada passo de tempo, os agentes tentam se comunicar uns com os outros. Os dados podem ser transmitidos e recebidos entre os agentes quando a comunicação é estabelecido. Os mecanismos que o Módulo de Comunicação emprega para estabelecer a comunicação pode variar. Por outro lado, o Módulo de Navegação é responsável por controlar o robô para executar a ação selecionada pela política de exploração. O robô o espaço de ação é discreto e o robô se move em etapas fixas. Portanto, o Módulo de Navegação pode usar diferentes técnicas para controlar o robô, como métodos de planejamento de caminho, ou simplesmente usando dados de odometria para girar o robô na direção alvo e ativar sua hélices para percorrer uma distância específica, em casos como exploração marítima. É válido notar que quanto melhor o desempenho dos blocos periféricos, melhor será o desempenho da estratégia de exploração.

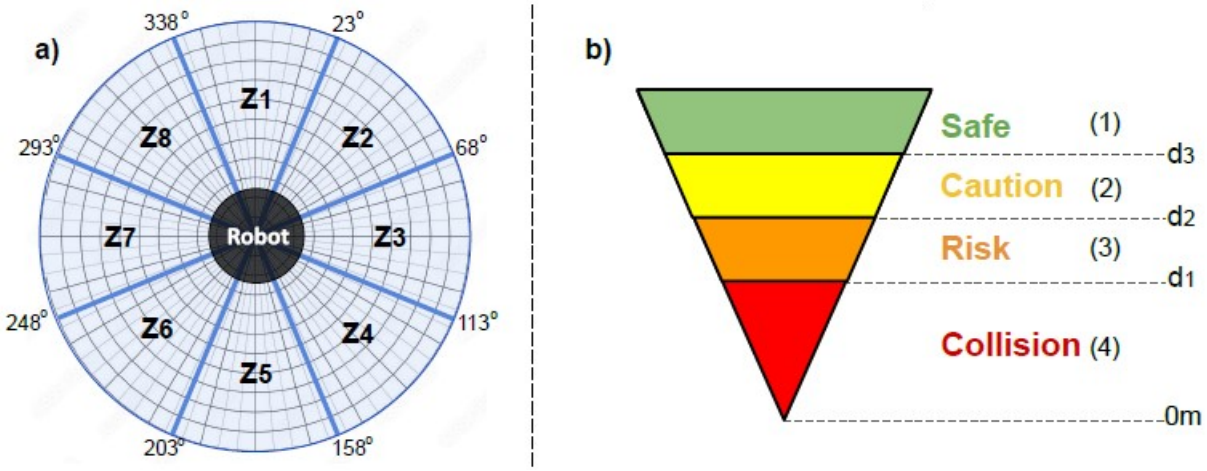
3.1.2 Movimentação

Como a simulação é sobre um ambiente 2D, a movimentação dos robôs também é limitada pela dimensão do sistema. Desta forma, os robôs, a cada ação, podem realizar um dos seguintes quatro movimentos: **esquerda, direita, para frente e para trás**. Para que cada robô possa decidir qual dos movimentos é o melhor a ser tomado dado ao seu estado atual e conhecimento adquirido, é necessário que uma estrutura capaz de perceber o ambiente ao redor do robô e uma logística de como avaliar os dados recebidos.

3.1.2.1 Sensores

Todo robô tem seus próprios sensores que são responsáveis pelo reconhecimento do mundo ao redor do robô e manter o histórico de movimentos tomados previamente. Para este projeto, o agente é equipado com um módulo de odometria e sensor de distância a laser, capaz de detectar todas as direções ao redor do agente. A partir disso, o algoritmo de exploração foi planejado especificamente para receber dados de scanners a laser 2D. Por outro lado, o sistema não é limitado a um tipo de robô (e.g. veículos aéreos, terrestres, aquáticos), também os sensores do módulo de odometria podem ser modificados para outros tipos, como câmeras ou Unidade de Medição Inercial (IMU).

Figura 3.2: a)Esquema representando como as medições do laser 360 espaçadas por um grau são igualmente dividido em oito zonas de proximidade. b) Escala que ilustra o estado de proximidade de cada zona pode assumir dependendo das distâncias medidas.



(GARAFFA, 2022)

Outra responsabilidade importante dos sensores, principalmente em máquinas com navegações autônomas, é evitar que o agente colide com qualquer objeto indesejável. Para evitar as colisões, poder calcular a distância de qualquer obstáculo do agente, nos dá a van-

tagem de saber quais áreas são mais seguras de explorar, e com isso o mais importante: qual ação tomar a seguir. Por isso, os sensores também podem medir distâncias, além de reconhecer se a região que está sendo escaneada é um obstáculo ou não.

Para realizar estas responsabilidades, os lasers de leitura foram divididos em oito zonas ($Z = (Z_1, Z_2, \ldots, Z_8)$). Dessa forma, se consideramos que devemos poder reconhecer a região em torno do robô, ou seja, 360^o, e que as oito regiões serão divididas igualmente, cada região representa 45^o. Além disso, cada região é dividida em quatro regiões que definem o nível de perigo o agente se encontra de possível colisão de algum obstáculo: colisão, risco, cuidado e seguro, como mostrado na Figura 3.2, onde, considerando l_m é a distância de uma região recebida pelo sensores:

- Colisão representa quando $0 \leq l_m < d_1$
- Perigo represente quando $d_1 \leq l_m < d_2$
- Cuidado represente quando $d2 \leq l_m < d_3$
- Seguro representa quando $d_3 \leq l_m$

3.1.2.2 Recompensa

Em projetos de aprendizado, utilizando robôs, as recompensas são incorporadas utilizando estratégias heurísticas, de modo que seja algo mais simples, sem precisar de alto processamento pela parte do robô ou do programa. Também, conforme o trabalho de Montasib et al. (MOHTASIB GERHARD NEUMANN, 2021), em aplicações similares a este trabalho, recompensas densas resultam, na maioria das vezes, em melhores resultados do que recompensas mais esparsas. Sendo que recompensas densas realizam recompensas intermediárias, que permite que o agente seja recompensado em múltiplos estados variados. Neste trabalho, as recompensas foram pensadas para que os robôs fossem altamente penalizados se colidirem com algum objeto e compensado se escolherem explorar novas regiões.O Algoritmo2 abaixo representa a estrutura básica da função de recompensa utilizada no algoritmo.

Algorithm 2 Estrutura básica da função de recompensa

1: **if** $collision$ **then**
2: $\quad R(t) = max\_penalty$
3: **else if** $(explored\_rate \leq max\_explored\_rate)$ **then**
4: $\quad R(t) = max\_reward$
5: **else if** $(explored\_rate(t) - explored\_rate(t-1)) > 0$ **and** $new\_position$ **then**
6: $\quad R(t) = R_1$
7: **else if** $explored\_rate(t) - explored\_rate(t-1)) == 0$ **and** $new\_position$ **then**
8: $\quad R(t) = R_2$
9: **else if** $(explored\_rate(t) - explored\_rate(t-1)) == 0$ **and** $old\_position$ **then**
10: $\quad R(t) = R_3$

Como mencionado, se um robô colidir, a recompensa ser o valor de maior penalidade, ou seja, *max-penalty* que deve ser um valor negativo. Nos outros casos em que não houve colisão, a recompensa depende do novo valor da taxa de exploração, *explore-rate(t)*, comparado com a taxa anterior,*explore-rate(t-1)*. Caso a diferença for maior, o robô recebe a recompensa *R1*, que é proporcional com o tamanho da diferenças entre as taxas. Se a diferença for igual a zero, ou seja, não houve mudança da taxa, porém a posição do robô é um local novo ainda não explorado, o robô recebe a recompensa *R2* que é positiva e fixa. Finalmente, se a diferença for igual a zero, porém a posição é uma já visitada pelo próprio robô ou pelos outros, a recompensa do robô é um valor negativo, porém com uma penalidade muito menor do que a penalidade máxima. Como os robôs tem acesso as trajetórias dos outros, essas recompensas incentivem que os robôs cooperarem entre si.

3.2 Sistema de Múltiplos Robôs

Como descrito anteriormente, o Módulo de Decisão é responsável pela computação dos dados recebidos tanto dos sensores quanto o Módulo de Comunicação para selecionar qual ação deve ser mandada para o módulo de Comunicação. As principais ações compostas por esse módulo estão apresentadas em ordem na lista 3.2, porém nesta seção nós iremos focar no compartilhamento de trajetórias entre robôs e na tomada de decisão de ações.

1. Definir estado das zonas lidas pelos sensores, conforme descrito do Subcapítulo 3.1.2.2.
2. Salvar a trajetória do robô.

3. Compartilhar e salvar as trajetórias dos outros robôs disponíveis.
4. Montar o vetor de estado.
5. Utilizar a politica de exploração para decidir melhor ação a seguir.

3.2.1 Compartilhamento de trajetórias entre Robôs

Um dos problemas de sistemas com múltiplos robôs é o custo de memória necessária para o processamento conforme o números de robôs aumenta. Não apenas atrapalha a eficácia do sistema, mas também retira os benefícios dos processos de exploração sem mapa, como a desse trabalho. Porém, ainda querem que o agente ainda possa ter acesso as trajetórias dos outros robôs para a otimização do processo da politica de exploração. Então, para minimizar o máximo possível esse problema, foi criado um método para lidar com as trajetórias dos outros robôs, ainda mantendo a compartibilidade entre os robôs.

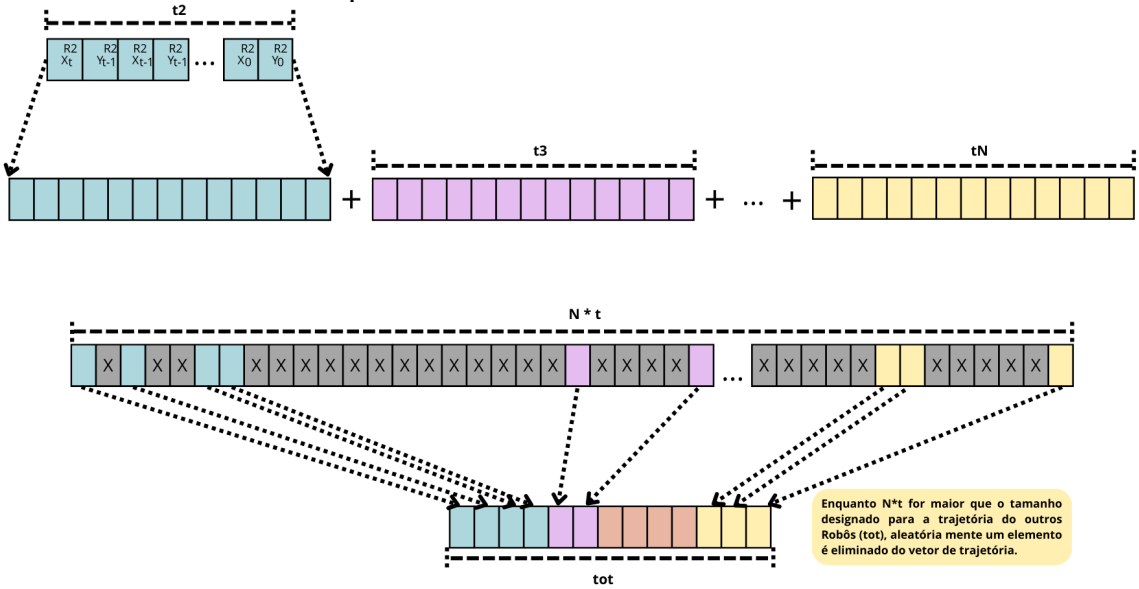
Primeiro vamos imaginar o seguinte cenário, digamos que cada robô salva um número fixo de pontos de trajetória em um vetor a partir do tempo t_0 e que, no total, necessite um espaço T de memória para armazenar esses pontos. Desta forma, um robô ao receber as trajetórias dos outros n robôs, necessitará de um espaço de $n * T$ se for armazenar todos os pontos de trajetória. Se considerarmos que isso ocorre em todos os robôs, o sistema, não apenas irá necessitar de $n^2 * T$ de espaço de memória, mas também terá que processar um vetor de trajetória cada vez maior em todas funções em que o vetor é usado. Com isso, a necessidade de espaço de memória em um robô cresce de maneira linear, $O(n)$, enquanto a necessidade do sistema como um todo cresce de maneira quadrática $O(n^2)$, para cada robô adicionado.

Como um dos objetivos do algorítimo PPO é aprender como explorar uma região desconhecida utilizando um ou mais agentes sem a necessidade de mapear a região, a situação apresentada a cima é inaceitável. A solução criada para resolver o problema foi modelado com o foco de que o custo de memória de um robô não seja relacionado com o número total de robôs no sistema. Para isso o vetor, onde é salvo as trajetórias compartilhadas, se manterá com um valor fixo, que é o mesmo valor do vetor de trajetória do próprio robô, dessa forma, nos caso onde o numero de robô é igual a dois, cada robô terá acesso a toda trajetória enviada pelo outro robô.

Agora que o vetor de trajetória é de tamanho fixo, temos que lidar com a situação onde o tamanho total das trajetórias compartilhadas é maior que o vetor. Primeiro verifi-

camos que todos os pontos são únicos e deletamos os repetidos se houver. Se o tamanho total das trajetórias ainda for maior que o vetor após a triagem, um ponto aleatório é retirado da lista, essa ação é repetida até que a trajetória seja do tamanho máximo que possa ser inserido no vetor. A Figura 3.3 mostra um exemplo dessa transmissão de trajetória, supondo que todas trajetórias são únicas, sem pontos repetidos.

Figura 3.3: Representação de comunicação de trajetórias de múltiplos robôs, pelo ponto de vista do robô R1 no tempo t.



Finalmente, com essas modificações, em vez de um sistema onde a memória para armazenar as trajetórias de crescimento $O(n^2)$, temos um sistema com crescimento linear $O(n)$, e para cada robô não há mudança no tamanho da memória.

3.2.2 Tomada de decisões

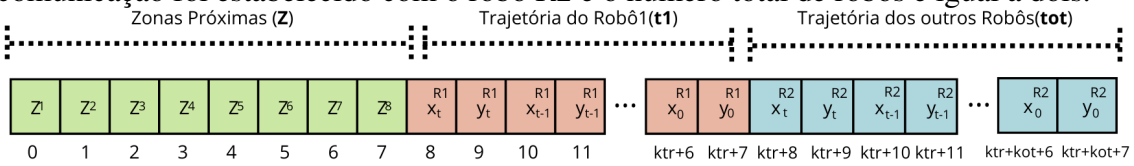
Em termos gerais, o módulo de tomada de decisão combina os dados processados pelo estágio anterior no estado atual, aplica este estado na política de exploração e retorna a próxima ação do robô. O problema de exploração já foi formulado como um MDP, e a estratégia é utilizar o algoritmo de RL para achar a melhor politica possível. Os elementos de o problema RL, que são o espaço de estado, o espaço de ação e a função de recompensa, são definido da seguinte forma:

- Estados: O estado do ambiente é o resultado da concatenação da proximidade zonas (Z), a trajetória do robô (tn) e as trajetórias dos outros robôs (tot), resultando em um vetor com tamanho fixo $8 + k_{tr} + k_{ot}$. A Figura 3.4 ilustra a representação do

vetor de estado de robô R1 na etapa de tempo t. Para facilitar a compreensão da estrutura vetorial, os outros a trajetória do robô contém apenas a trajetória do robô R2. No entanto, como explicado anteriormente, o vetor de estado final estrutura e tamanho permaneceriam os mesmos, independentemente de quantas trajetórias de agentes fossem recebido.

- Ação: Neste caso as ações são compostas pelos possíveis movimentos que o robô pode realizar em um step, são eles: **Frente, Pra trás, Esquerda e Direita**
- Recompensa: Valor designado para uma ação num estado no tempo t. Conforme explicado em 3.1.2.2

Figura 3.4: Representação vetorial de estado do robô R1 no tempo t, considerando que a comunicação foi estabelecido com o robô R2 e o número total de robôs é igual a dois.



O algoritmo utilizado para otimizar os parâmetros da política é o Proximal Policy Optimization, cujas equações e vantagens foram descritas na Seção 2.2. O algoritmo usa um modelo ator-crítico, onde ator e crítico são redes neurais totalmente conectadas com duas camadas ocultas de 64 neurônios. A exploração é descentralizada, ou seja, cada robô tem sua própria rede neural. Uma estrutura de treinamento proposta no trabalho utilizando O modelo ator-crítico é mostrado na Figura 3.5. Consideremos o treinamento de um único robô. Durante a fase de treinamento, o agente realiza diversos lançamentos de políticas, ou em outras palavras, joga vários jogos. Cada implementação tem um número fixo de etapas (steps-roll). O agente e os parâmetros das redes neurais são inicializados quando o treinamento começa.

O agente inicializado coleta dados e monta o vetor de estado. O estado é usado como entrada para o crítico, que produz o valor do estado $(V(s))$, e para o ator que retorna o próxima ação a_t. O robô atua, resultando em um novo estado ambiental. Tomando o selecionado ação tem consequências no meio ambiente (por exemplo, robô colide, nova região é visitada,etc.). Tais consequências são consideradas para determinar a recompensa associada à ação e ao par. Em cada etapa, o estado, a ação, o valor do estado, a recompensa e as probabilidades de ação são armazenados como dados de implementação. Quando uma implementação termina, essas informações são usadas para atualizar os parâmetros das redes neurais ator e crítica de acordo com as equações apresentadas em 2.2. Um

Figura 3.5: Esquema simplificado de como os dados são tratados durante treinamento PPO

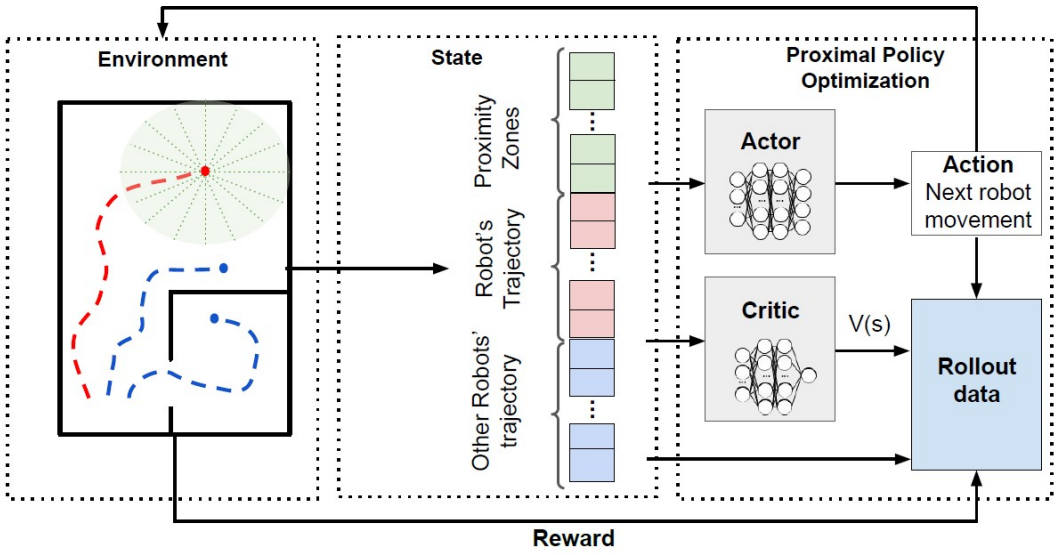
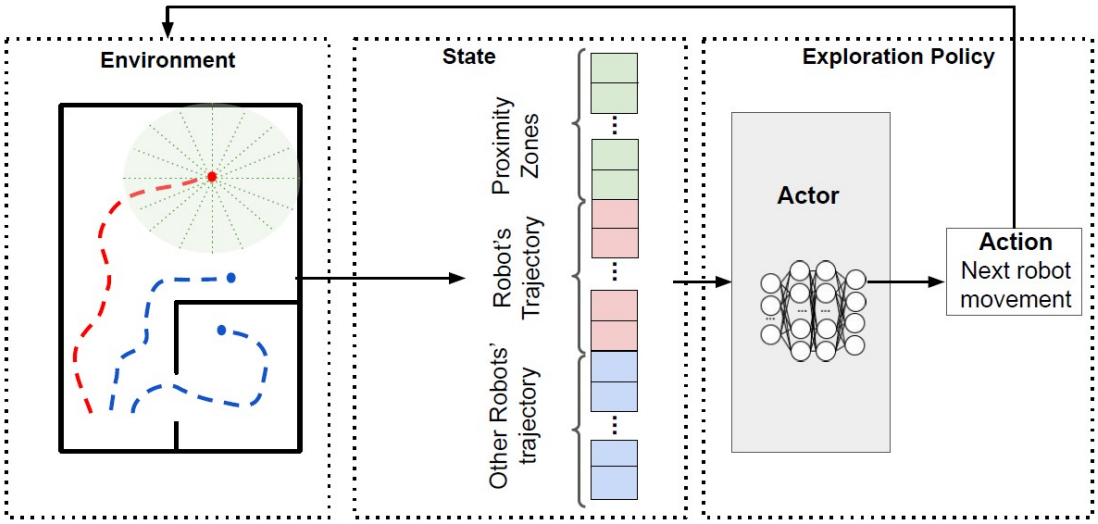


Figura 3.6: Esquema simplificado da estrutura de inferência após a conclusão do treinamento. Política de Exploração.



processo de treinamento possui um número máximo de etapas (max-steps), o que significa que o número de atualizações de NNs é igual ao quociente de max-steps e steps-roll. A Figura 3.6 ilustra uma estrutura de inferência simplificada, onde um robô treinado executa exploração. A rede crítica e o cálculo da recompensa são removidos, e o resultado rede de atores configura a política de exploração.[1]

[1]O código da simulação realizada neste trabalho pode ser encontrado em : <https://github.com/DudaRuhe/multiple_robotic_exploration.git>

4 EXPERIMENTOS E RESULTADOS

Este capítulo apresenta e discute os resultados obtidos nos experimentos propostos no Seção 4.1. A Seção 4.3 define os parâmetros utilizados no treinamento e apresenta os resultados de validação para 2 robôs no labirinto simples e nos ambientes de sala simples. A Seção 4.4 apresenta os resultados para diferentes conjuntos de robôs, variando de 2 à 10, em ambientes mais complexos, comparando a eficiência de exploração alcançada e a capacidade de generalização com diferentes métodos de exploração.

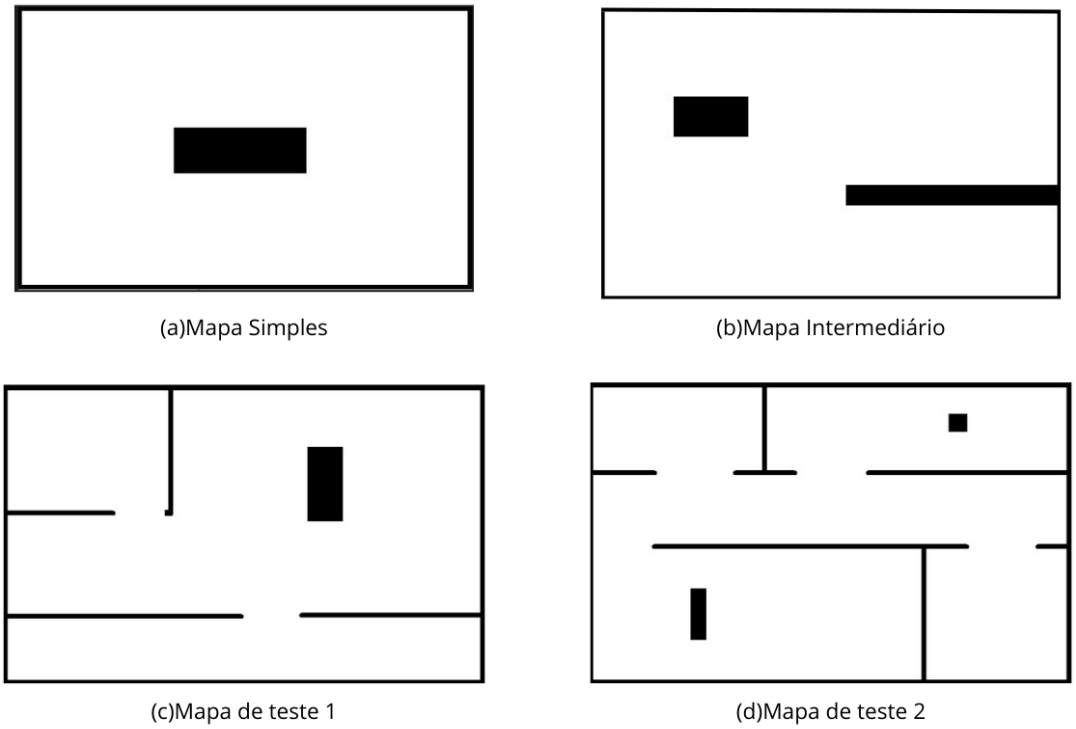
4.1 Mapas de Teste

Para a simulação utilizada nos métodos de validação, foram usados quatro diferentes mapas. Os quatro mapas possuem dimensões diferentes e servem como três níveis de dificuldade para verificar os comportamentos realizados pelos robôs conforme seu treinamento. No caso, um mapa com baixa complexidade possui menos obstáculos e uma área menor a ser explorada, como o Mapa Simples, enquanto um mapa de alta complexidade possui múltiplos obstáculos e uma área maior, criando uma situação na qual, para explorar certas regiões, o agentes devem realizar uma trajetória especifica, como no Mapa de teste 1 e 2. A Figure 4.1 representa cada um dos mapas, enquanto a Tabela 4.1 mostra as dimensões dos mapas e qual seria a escala real em metros, em testes onde o número de robôs é mais alto o mapas foram escalonado para 1.5 do seu valor, se não, pela quantidade de robôs o mapa era explorado completamente com apenas alguns steps.

Tabela 4.1: Mapas e suas Dimensões

Nome do Mapa	**Dimensões**	Escala Real(m)
Mapa Simples	8x9	1.6x1.8
Mapa Intermediário	20x13	4.0x2.6
Mapa de Teste 1	40x25	8.0x5.0
Mapa de Teste 2	40x25	8.0x5.0

Figura 4.1: Mapas utilizados para validação da política de exploração.



O Mapa simples, por seu baixo nível de complexidade, foi escolhido para validação de certos comportamentos dos robôs, sendo o número de robôs nesses testes são dois ou quatro. Desses comportamentos temos conceitos mais simples como evitar obstáculos e explorar o mapa com sucesso. O mesmo será feito para o mapa intermediário, porém nesse também será validado para um número pequeno de robôs conseguem explorar o mapa de uma maneira cooperativa. O que for aprendido será utilizado para os outros testes com grupo maiores de robôs. O Mapa de Teste 1 serve para verifica o quanto os robôs utilizam da cooperação para explorar o mapa de uma maneira eficiente, para estes testes foram utilizado diversos números de robôs, para visualizar como o números de robôs afeta a cooperatividade. Por fim, o Mapa de Teste 2 serve para ver como os robôs se adaptam a um mapa diferente utilizando a politica de exploração aprendida no teste do Mapa de Teste 1, quanto tempo demora para se adaptar, ou se adapta para um ambiente diferente por si só. Os detalhes dos testes serão explicados no próximo capítulo.

4.2 Hiperparâmetros

Conforme descrito na Seção 2.2, uma das vantagens do algoritmo de Otimização de Política Proximal é que ele fornece um bom equilíbrio entre facilidade de implementação e ajuste de parâmetros. Ao mesmo tempo, a inicialização adequada dos parâmetros pode ser decisiva para que o desempenho das políticas tenha os melhores resultados. Para isso, foram consultados diferentes trabalhos que utilizam PPO (SCHULMAN FLIP WOLSKI, 2017) (CHEN, 2019), e os parâmetros mais comumente utilizados foram selecionados conforme descrito na Tabela 4.2. O recozimento da Taxa de Aprendizagem (LR) foi usado conforme descrito pela Equação 4.1, onde U representa o número total de atualizações de parâmetros de política durante o treinamento, u representa quantas atualizações foram realizadas no tempo t e $LR_{t=0}$ representa o aprendizado inicial. valor da taxa. O número total de atualizações (U) pode ser definido como o quociente entre o número total de etapas de treinamento e o número de etapas por implementação. Dessa forma, a LR diminui linearmente à medida que aumenta o número de atualizações de políticas.

Tabela 4.2: Tabela dos Hiperparâmetros

Parâmetros	***Melhor valor***
Clip Coefficient ϵ	0.2
Valor inicial do Learning rate	0.00025
Valor da função de Coefficient (c_1)	0.5
Entropy Coefficient (c_2)	0.01
Número dos mini-batches	4
Step por rollouts	128
Número de ambientes vetorizados	4

$$LR_t = \frac{1.0 - (u_t - 1)}{U} LR_{t=0} \tag{4.1}$$

4.3 Método de Validação Utilizando Dois Robôs

A estrutura de exploração $mapless$ proposta foi validada nos mapas simples ilustrado na Figura 4.1. Diferentes configurações foram empregadas para treinamento dos agentes, nesta seção iremos apresentar quais funções de recompensa, parâmetros do modelo, lógica de trajetória, e o raio de exploração resultaram na exploração mais eficiente para o conjunto de de Dois Robôs. Nas próxima seção serão realizar os testes para dife-

rentes conjuntos de robôs e compararmos com os melhores resultados desta seção

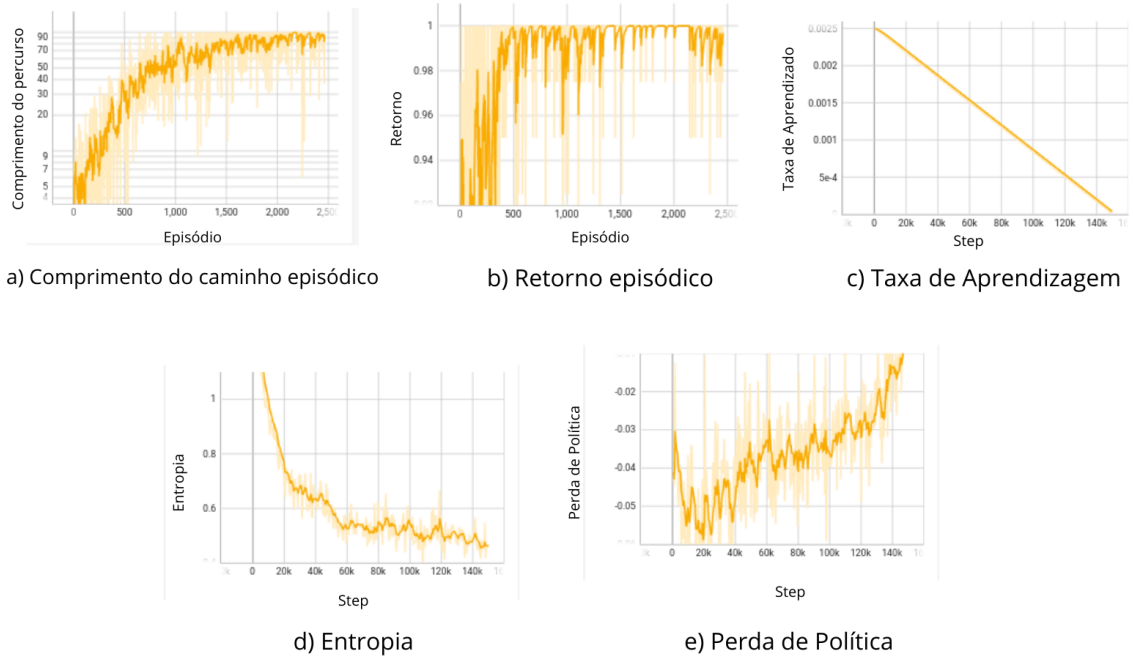
4.3.1 Mapa Simples

Os experimentos utilizando o Mapa Simples com dois robôs são os ambientes mais simples para confirmar a convergência do método de aprendizagem. Para esse mapa, não iremos ainda nos preocupar se os robôs estão se comportando de uma maneiro cooperativa, apenas se eles conseguem realizar a tarefa de explorar o mapa.O primeiro experimento usa aprendizagem curricular para que os robôs aprenda primeiro como evitar colisões e depois aprenda como otimizar o caminho de exploração. Assim, o vetor estado apenas atualiza o status das regiões próximas. O as posições que receberiam a trajetória permanecem preenchidas com zeros. A recompensa é moldada de forma diferente do que para experimentos que visam otimizar a exploração. Considerando o objetivo de evitar colisão, o agente recebe uma punição de -1 quando colide. Se e um episódio um robô atingir 100 passos sem colidir, ele recebe uma recompensa de +1. Uma pequena recompensa de 0,01 é definido toda vez que uma nova posição é visitada para encorajar os agentes a explorarem outras posições diferentes em vez de repetir os mesmos movimentos (por exemplo, para frente e para trás), pois isso o comportamento também evitaria colisões com sucesso. Se nenhuma das condições anteriores for cumprido, a recompensa é 0. Um episódio termina se houver uma colisão ou se um robô avançar 100 passos. O agente1 é inicializado em uma posição livre aleatória, que será considerada a posição (0,0) no mapa, o agente2 também será alocado numa posição aleatória e sua posição será referente ao agente1 no início de cada episódio. Treinamento termina quando o robô completa um total de 150 mil passos.

Para mostrar os resultados dos teste foram gerados gráficos utilizando o TensorBoard, ferramenta de visualização desenvolvida pelo TensorFlow (ABADI ASHISH AGARWAL, 2016). Facilitar visualização, as curvas foram suavizadas com um recurso do Tensorboad que aplica um média móvel exponencial para as métricas. Os valores absolutos são representados pelo cores desbotadas.

Os gráficos da Figura 4.2 reforçam que o processo de aprendizagem foi bemsucedido. Política a perda está correlacionada com o quanto a política muda durante o treinamento. Conforme mencionado na Seção 2.2, os parâmetros de política são atualizados usando a subida gradiente. No gráfico 4.2e é possível observar que a perda da política aumenta ao longo do tempo, convergindo para um valor próximo de zero. A entropia re-

Figura 4.2: Métricas de treinamento de dois agentes no Mapa Simples aprendendo a evitar colisões



a) Comprimento do caminho episódico b) Retorno episódico c) Taxa de Aprendizagem

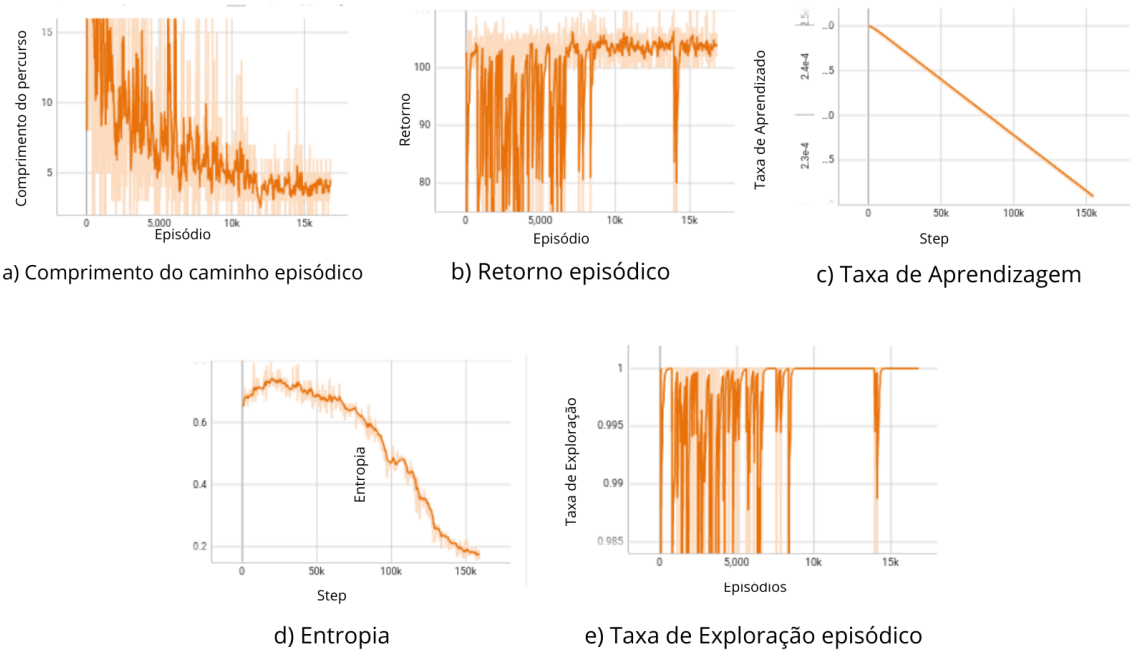
d) Entropia e) Perda de Política

presenta quão aleatórias são as decisões políticas. Deve diminuir lentamente durante um processo de treinamento bem-sucedido, conforme mostrado na Figura 4.2d. A variação no comprimento do caminho observada nos episódios finais mostra que o agente ainda pode colidir em determinadas situações. No entanto, os resultados globais demonstram que a política resultante aprende suficientemente como evitar colisões, sendo adequado para a rede neural base para ajuste fino de exploração.

Agora queremos avaliar se os agentes conseguem explorar o mapa de uma maneira eficiente, ou seja que diminua o numero de passos necessários para explorar o mapa. Primeiro nós vamos inicializar ambos agentes com a política de prevenção a colisão aprendido no teste anterior, cujos parâmetros são atualizados durante o treinamento. Considerando que este experimento é para validar o método e como o labirinto apresenta um tamanho pequeno, considerou-se que os agentes exploram apenas uma área de 20cm em torno de suas posições. A lógica de trajetória de queda de pontos aleatórios é empregado, significando que quando mais de 50 posições são visitadas, os pontos aleatórios são descartados. O formato da recompensa mudou de acordo com o novo objetivo. O a base para a função de recompensa foi apresentada no Subcapítulo 3.1.2.2. Se ocorrer uma colisão, o a recompensa é -1. Se a taxa de exploração for 100%, a recompensa será 100. Se o delta de exploração, ou em outras palavras, o número de novas células exploradas, for maior que zero, a recompensa é o número de novas células exploradas. Se nenhuma nova célula

for explorada, a recompensa será $-0,1$. Um episódio termina se o agente colidir ou a taxa de exploração for 100%. Novamente, o agente é inicializado em uma posição livre aleatória no início de cada episódio. O treinamento termina quando o agente completar 150 mil passos. Um grande número de etapas é escolhido para avaliar por quanto tempo a política leva para otimizar a exploração e comparar com os resultados do aprendizado da exploração de arranhar.

Figura 4.3: Métricas de treinamento de dois agentes no Mapa Simples aprendendo exploração eficiente do mapa com reforço



a) Comprimento do caminho episódico b) Retorno episódico c) Taxa de Aprendizagem

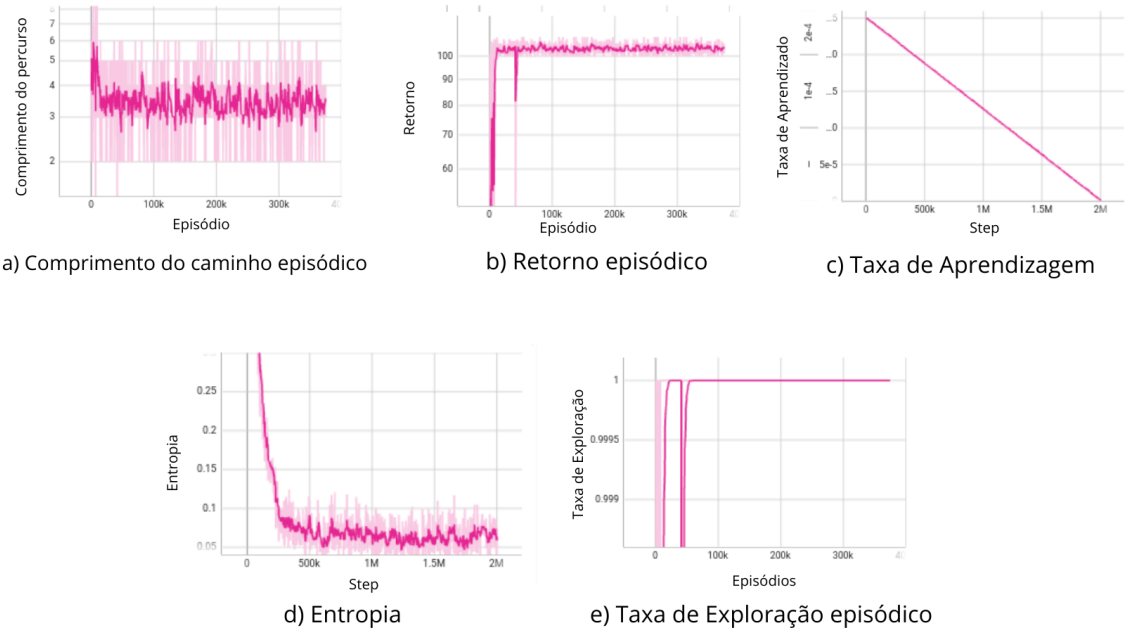
d) Entropia e) Taxa de Exploração episódico

Percebe-se que, no gráfico 4.3a, devido o tamanho pequeno do Mapa Simples, o comprimento da trajetória dos agentes converge para em torno de 5 $steps$ por episódio. Isso ocorre após 10 mil episódios, ao mesmo tempo que o retorno converge para um valor acima de 100, ou seja, a recompensa máxima por explorar completamente o mapa mais a soma de células novas encontradas e a taxa de exploração converge para 1. Vale ressaltar que ainda pode ainda ocorrer algumas colisões ocasionalmente, como no ponto perto do 15 mil episódio.

Agora, como o último teste no Mapa Simples, vamos repetir o teste anterior, porém não iremos inicializar o agente1 com a politica de prevenção a colisão, dessa forma apenas um dos agentes inicia com a rede neural enquanto o outro inicia com os pesos aleatórios, vamos aumentar o numero de passos para 2M. Os resultados do teste, representado pelos gráficos da Figura 4.4, foram bastante parecidos com o teste utilizando a politica de

colisão. Sendo que nos primeiro 100 mil episódios ainda pode se encontrar casos onde colisão ocorre, como mostrado no gráficos 4.4b e 4.4e até os agentes, até convergir para a exploração de 100% do mapa, necessitando de entorno de 5 passos.

Figura 4.4: Métricas de treinamento de dois agentes no Mapa Simples aprendendo exploração eficiente do mapa sem reforço

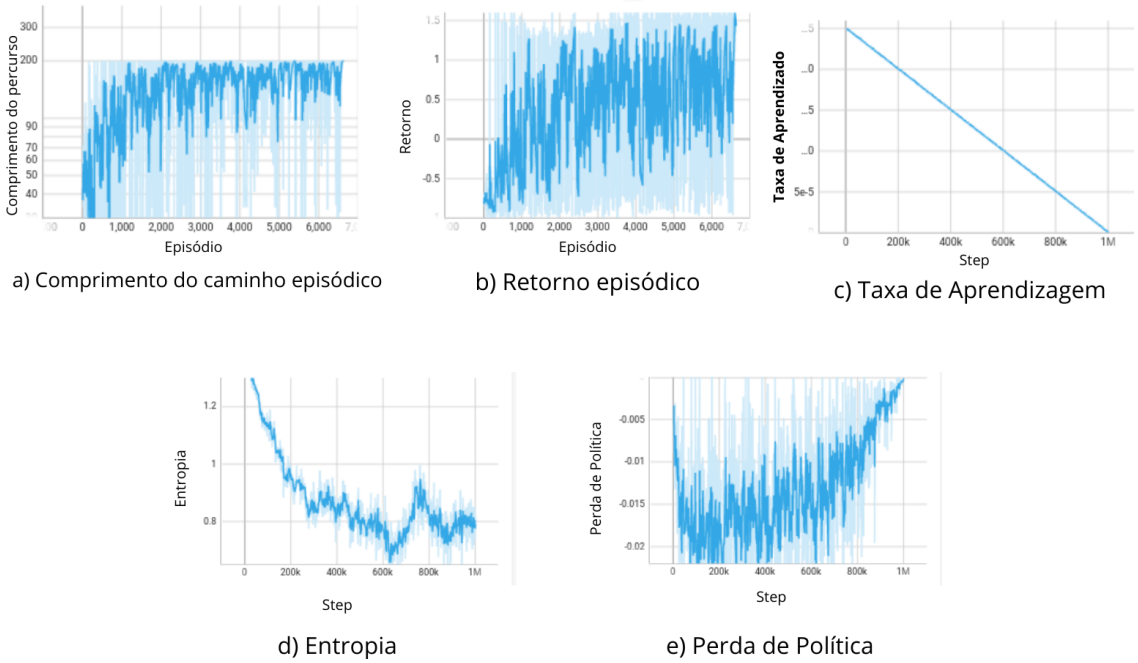


4.3.2 Mapa Intermediário

Semelhante aos experimentos do Mapa Simples, o primeiro conjunto de treinamento no Mapa Intermediário usa aprendizagem curricular. Portanto, o objetivo do primeiro treinamnto é aprender como evitar colisões com base apenas no estatuto das regiões de proximidade. A função de recompensa é a mesma que aquele empregado para o Labirinto Simples, exceto pelo número de etapas utilizadas como sucesso critério. Como a Sala Simples apresenta dimensões superiores ao labirinto, o máximo o número de passos por episódio foi considerado 200 em vez de 100. Assim, a recompensa foi -1 para colisão, 1 para completar 200 passos, 0,01 para descobrir novas posições, e 0 caso contrário. O número de etapas para encerrar o processo de treinamento foi aumentado para 1 milhão.

Vemos pelo gráfico da Figura 4.5a, que o treinamento, da mesma forma que o Mapa Simples, os agentes conseguem evitar colisões dado que o gráfico converge para 200 passos por episódio. Devido o mapa possuir um número maior de obstáculos, exite ainda

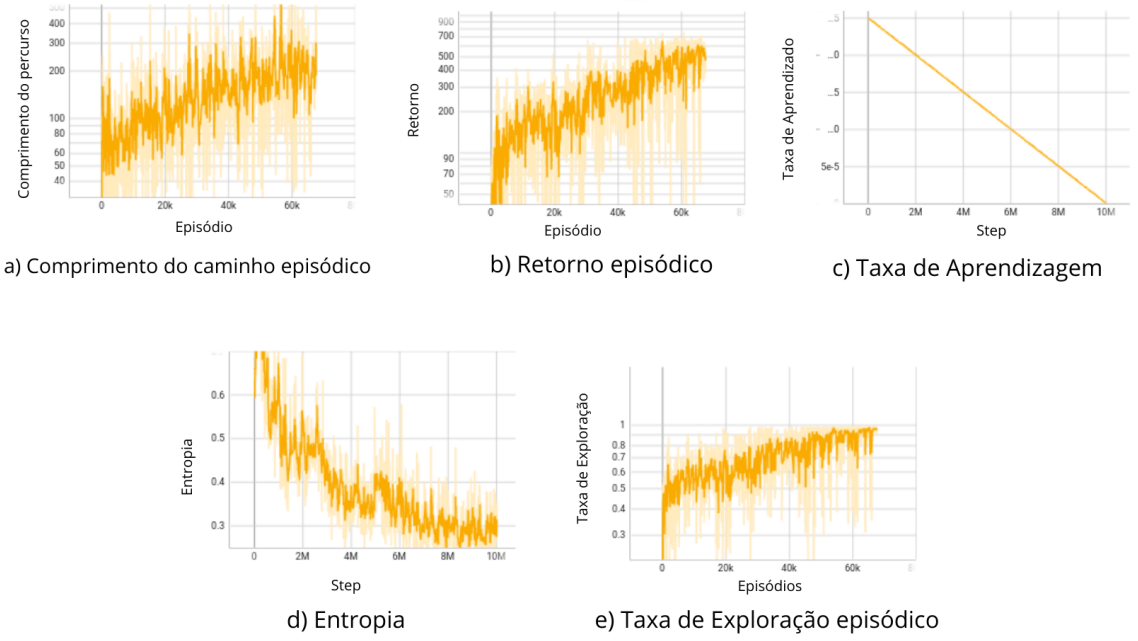
Figura 4.5: Métricas de treinamento de dois agentes no Mapa Intermediário aprendendo a evitar colisões



algumas ocorrências de colisões, mesmo ao final da do treinamento. Por isso, mesmo o retorno episódico mostrado no gráfico 4.5b existe uma variança maior, concentrando a um valor entre 0.5 a 1.0 ao final do treinamento.

A politica de prevenção de colisão resultante desse teste foi utilizado para o próximo teste. Considerando o tamanho do mapa, o raio de exploração foi definido como 80cm. O mesmo raio é empregado em todos os experimentos do Mapa Simples. Para reforçar a prevenção de colisões, uma recompensa de -100 foi determinado para cada colisão. Se a taxa de exploração for 98%, a recompensa será 100. Se o agente 1 explorarem células inéditas, a recompensa passa a ser a quantidade de novas células exploradas. Se nenhuma nova célula for explorada, mas o robô estiver em uma posição que não foi visitado antes, a recompensa é 0,5. Finalmente, se nenhuma nova célula for explorada e o robô tiver já visitou sua posição atual, uma penalidade de $-0,5$ é aplicada. Um episódio termina quando ocorre uma colisão ou quando a taxa de exploração é de 98%. O treinamento termina quando o agente executa 10 milhões de etapas.

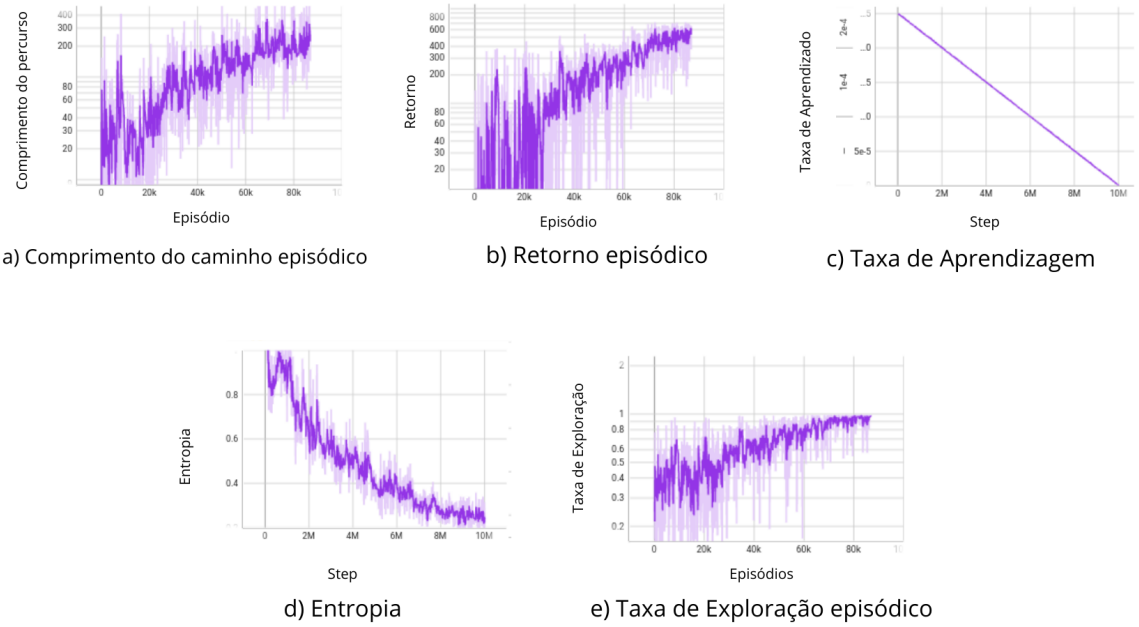
Figura 4.6: Métricas de treinamento de dois agentes no Mapa Intermediário aprendendo exploração eficiente do mapa com reforço



Como de esperado, devido seu maior tamanho, o comprimento de exploração do Mapa Intermediário é bem maior, comparando com o Mapa Simples, ficando entorno de 100 à 200 passos por episódio , Figura 4.6a. Também a taxa de exploração cresce de uma forma mais gradual, sendo que, apenas após 70 mil episódios, converte para entorno de 98% da área explorada conforme o gráfico da Figura 4.6e. O retorno, porém, fica à cima de 100 por uma boa parte dos episódio, pois como vários casos, mesmo não chegando na taxa desejada, devido o tamanho do mapa, a recompensa por células novas descobertas se acumulam resultando num retorno de valor alto.

Para ensinar os robôs a explorar o ambiente sem currículo, aprenda foi empregada a mesma configuração usada para o ajuste fino de exploração, exceto que os pesos das redes neurais são inicializados aleatoriamente. Da mesma forma que foi realizado nos testes para o Mapa Simples.

Figura 4.7: Métricas de treinamento de dois agentes no Mapa Intermediário aprendendo exploração eficiente do mapa sem reforço



a) Comprimento do caminho episódico
b) Retorno episódico
c) Taxa de Aprendizagem
d) Entropia
e) Taxa de Exploração episódico

Os gráficos da Figura 4.7, mostra que os agentes conseguem resultados extremamente parecidos, mesmo iniciando com uma rede neural com pesos aleatórios, em vez de pré-carregados. A maior diferença é que nos primeiros 20 mil episódios, o treinamento apresenta piores resultados do que o treinamento anterior, devido o fato que os agentes necessitam de mais tempo para aprender evitar colisões. Por fim, como foi comprovado que os agentes conseguem aprender a explorar um mapa com sucesso, vamos testar se eles aprendem explorar uma região de maneira cooperativa. Para isso, a função de recompensa, descrita no Subcapítulo 3.1.2.2, foi aplicada de acordo com a parametrização apresentada no Algoritmo3.

Algorithm 3 Algoritmo de Recompensa

if $collision$ **then**
    $R(t) = -100$
else if $(explored\_rate \leq 93\%$ **then**
    $R(t) = +100$
else if $(robot\_1\_new\_cells) > 0$ $(robot\_2\_new\_cells) > 0$ **then**
    $R(t) = robot\_1\_new\_cells + robot\_2\_new\_cells$
else if $(robot\_1\_new\_cells) == 0$ **and** $(robot\_2\_new\_cells) == 0$ **and** $new\_position$ **then**
    $R(t) = -1$
else if $(robot\_1\_new\_cells) == 0$ **and** $(robot\_2\_new\_cells) == 0$ **and** $old\_position$ **then**
    $R(t) = -5$

Antes, apenas se o agente1 descobrisse células novas a recompensa seria igual a soma de células novas descobertas. Agora, se qualquer um dos agentes descobrir novas células, essa recompensa é adquirida, ou seja, uma nova região não explorada foi descoberta. Dessa forma, incentivamos a cooperação entre os agentes, pois um agente não recebe uma recompensa positiva quando a aumento no novo número de células descoberta pelo agente1, mas também quando o mesmo ocorrem com os outros agentes.Comparado com o anterior experimentos, a taxa de exploração foi reduzida para 93% porque o Mapa de Intermediário apresenta maiores dimensões e maior complexidade. Usando um valor de 98%, por exemplo, poderia resultar em muitos episódios antes de atingir a taxa de exploração necessária ou mesmo em trens completos onde o agente nunca obtém a recompensa máxima.

Foi realizado dois testes para validar a cooperação entre 2 robôs. O primeiro ambos os robôs foram inicializados com com a política aprendida no teste de exploração sem reforço representado pela Figura 4.6. O segundo apenas o segundo robô foi iniciado com a politica, enquanto o outro iniciou como pesos aleatórios. A Figura 4.8 e Figura 4.9 representam o primeiro e segundo método respectivamente. Ambos treinamentos acabam ao realizarem 5M de passos.

Figura 4.8: Métricas de treinamento de dois agentes no Mapa Intermediário aprendendo exploração cooperativa, ambos com reforço

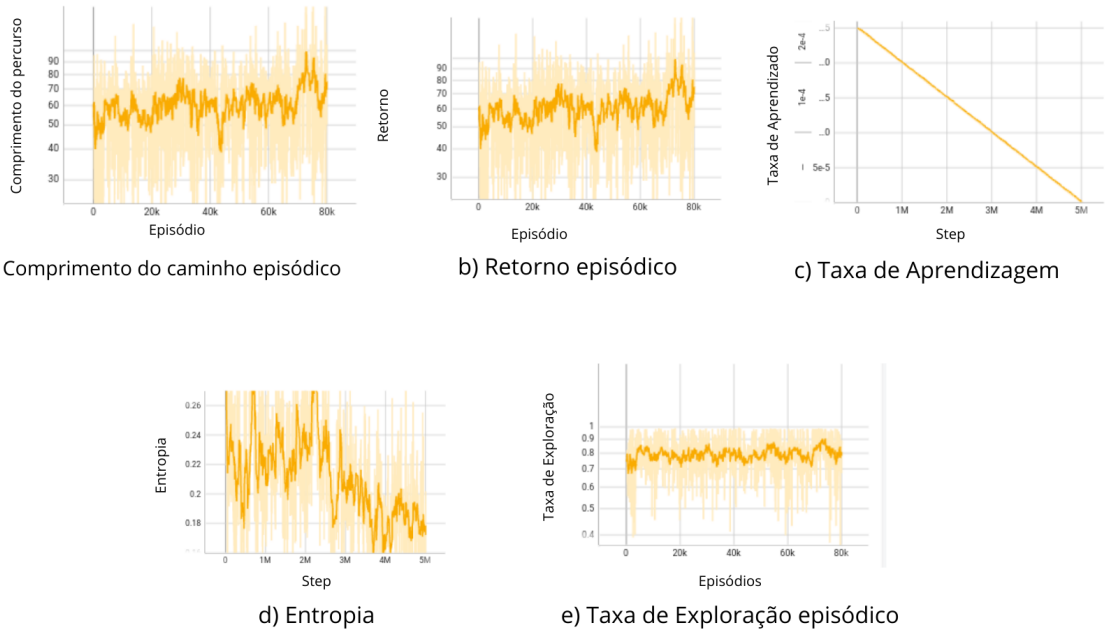
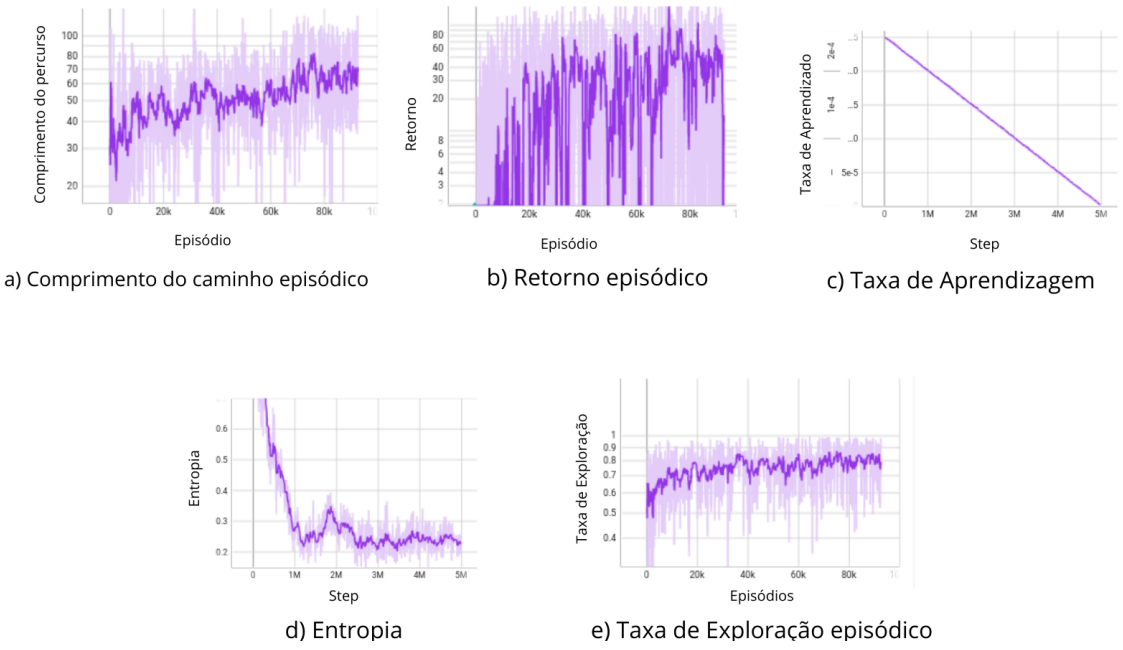


Figura 4.9: Métricas de treinamento de dois agentes no Mapa Intermediário aprendendo exploração cooperativa, um agente com reforço



a) Comprimento do caminho episódico
b) Retorno episódico
c) Taxa de Aprendizagem
d) Entropia
e) Taxa de Exploração episódico

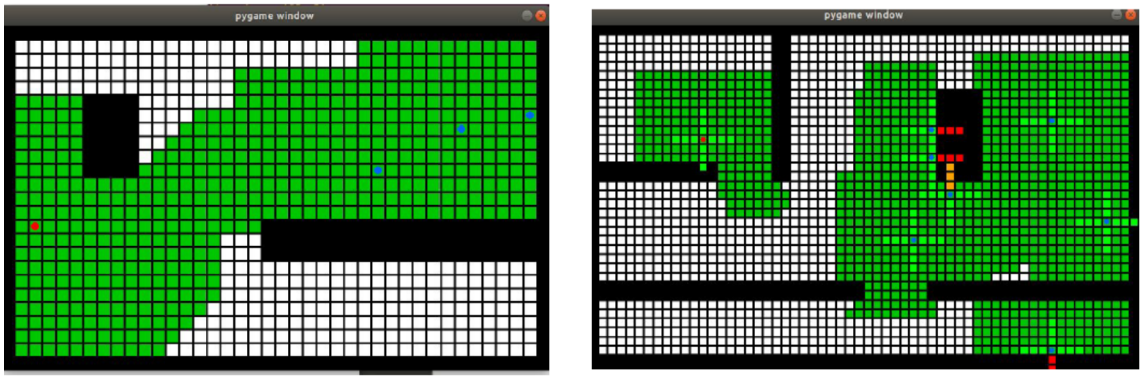
Vemos que em ambos os casos, a taxa de exploração mostrada nos gráficos 4.8b e 4.9b converge para as casas dos 90%, sendo que o treinamento, onde apensa um dos agentes recebe os pesos do treinamento anterior, tem menor variância de valor da taxa de exploração do que o treinamento onde ambos inciam com os pesos pré definidos, principalmente após o 60 mil episódio. Em ambos casos também há uma grande melhora nos números de passos por episódio, conforme os gráficos 4.8a e 4.9a mostram. Em vez de necessitar 100 à 200 passos como ocorreu no último treinamento, agora é necessário de 60 à 80 passos, novamente o treinamento iniciado com pesos aleatórios possui uma variância menor nos resultados que o primeiro método. Combinando com o fato que os agentes exploram uma taxa de exploração alta, conforme definida no Algoritmo3, podemos dizer que os agentes aprenderam com sucesso a explorar o mapa de uma maneira cooperativa.

4.4 Exploração com n Robôs

Acabamos de demonstrar que o programa utilizando o algoritmo PPO consegue não apenas ensinar dois agentes a explorarem uma região desconhecida, mas também de uma maneira cooperativa entre os agentes, tornando a exploração mais eficaz. Nesta seção iremos ver, não apenas se o mesmo ocorre ao modificarmos o número de agentes,

mas também comparar como o conjunto de agentes se adaptam ao utilizarem redes neurais treinadas para o dois agentes. Para todos os testes serão comparados os conjuntos de dois, quatro, seis, oito e dez agentes. A Figura 4.10 mostra dois casos de exploração, o da esquerda é um conjunto de quatro agentes explorando o mapa Intermediário, enquanto o da direita são oito agentes explorando o mapa Teste 1. A área verde representa a área já explorada, enquanto a branca representa área ainda não descoberta.

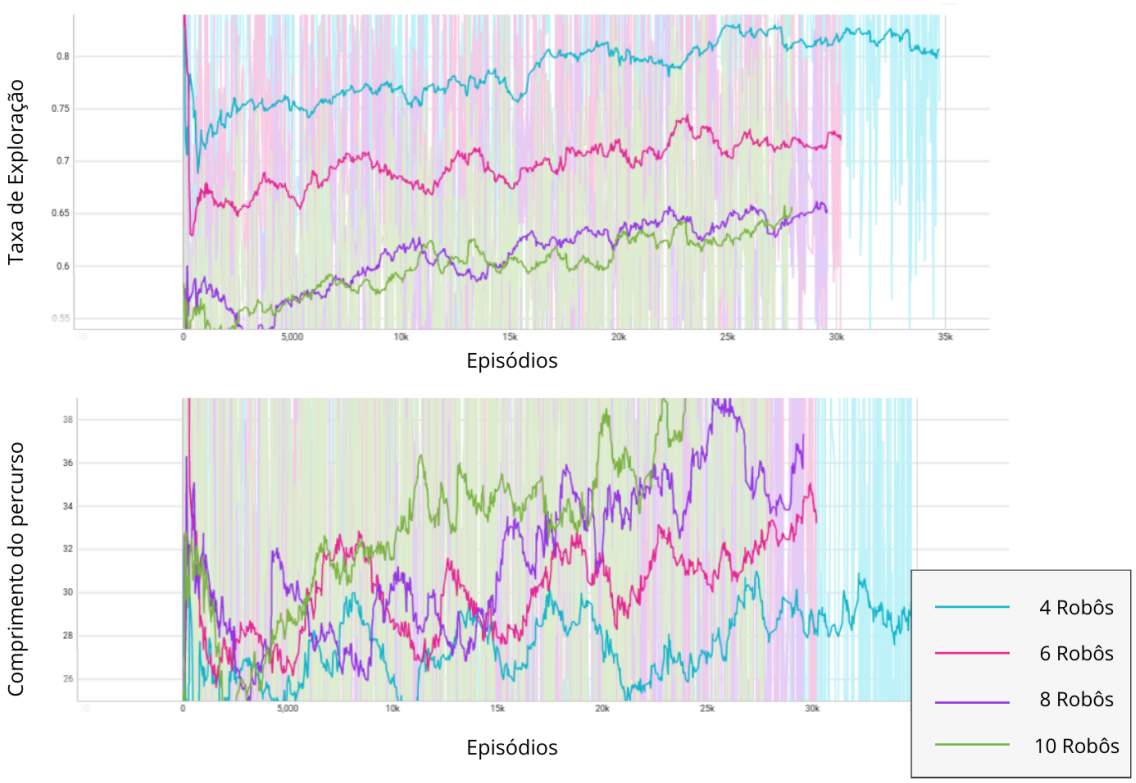
Figura 4.10: Aplicação executando dois casos de exploração de mapas com dois conjuntos diferentes de agentes



4.4.1 Mapa Intermediário

Para o Mapa Intermediário, foi utilizado como base comparativa o treinamento de Dois agentes no Mapa Intermediário com cooperação realizado no Subcapítulo 4.3.2 que obteve o melhor resultado. O resultado pode ser visto da Figura 4.9. Para a comparação, cada conjunto de agente, de quatro à dez, serão treinados para o Mapa Intermediário. Utilizando os mesmos hiperparâmetros e recompensas que foram utilizadas para o treinamento de dois agentes, ou seja, -100 se o agente colidir, +100 se for explorado 93% do mapa, o número de células descobertas acumulada, caso haja, -1 para nova posição e -5 se for uma posição já visitada. O treinamento de todos acaba ao atingirem 1M passos. O resultado do treinamento pode ser visto na Figura 4.11

Figura 4.11: Treinamento cooperação no Mapa Intermediários para diversos conjuntos de robôs



A Tabela 4.3, contém os dados extraídos nos testes de inferência, ou seja, a utilização das redes neurais num mapa, porém a rede não está mais sendo treinada pelo algoritmo PPO, apenas reagindo ao ambiente, conforme explicado no Subcapítulo 3.2.2.

Vemos que apesar do número maior de agentes, os conjuntos de seis, oito e dez tiveram uma pequena perda na média da taxa de exploração. Também percebe-se que a Tabela 4.3 se comporta da mesma forma de ambos gráficos na Figura 4.11, ou seja, a taxa de exploração tende a diminuir conforme aumenta os números de agentes, enquanto o comprimento do percurso aumenta. Desta forma, a cooperatividade entre os agentes é influenciada de forma negativa, conforme o número de agentes aumenta. Isso fica mais evidente, ao se notar que o melhor conjunto de agentes neste caso é o conjunto de quatro agentes, onde houve uma perda de apenas 0,02% na taxa de exploração em relação ao conjunto de dois agentes, porém a média de comprimento caiu quase 50%.

É possível que o resultado do teste anterior esteja relacionado, não apenas o número de robôs testados, mas também o tamanho do mapa que ele se encontram. Com um mapa de tamanho menor e uma quantidade grande de robôs, uma exploração cooperativa possa ser reduzida, como um aumento na possibilidade de colisões entre os robôs

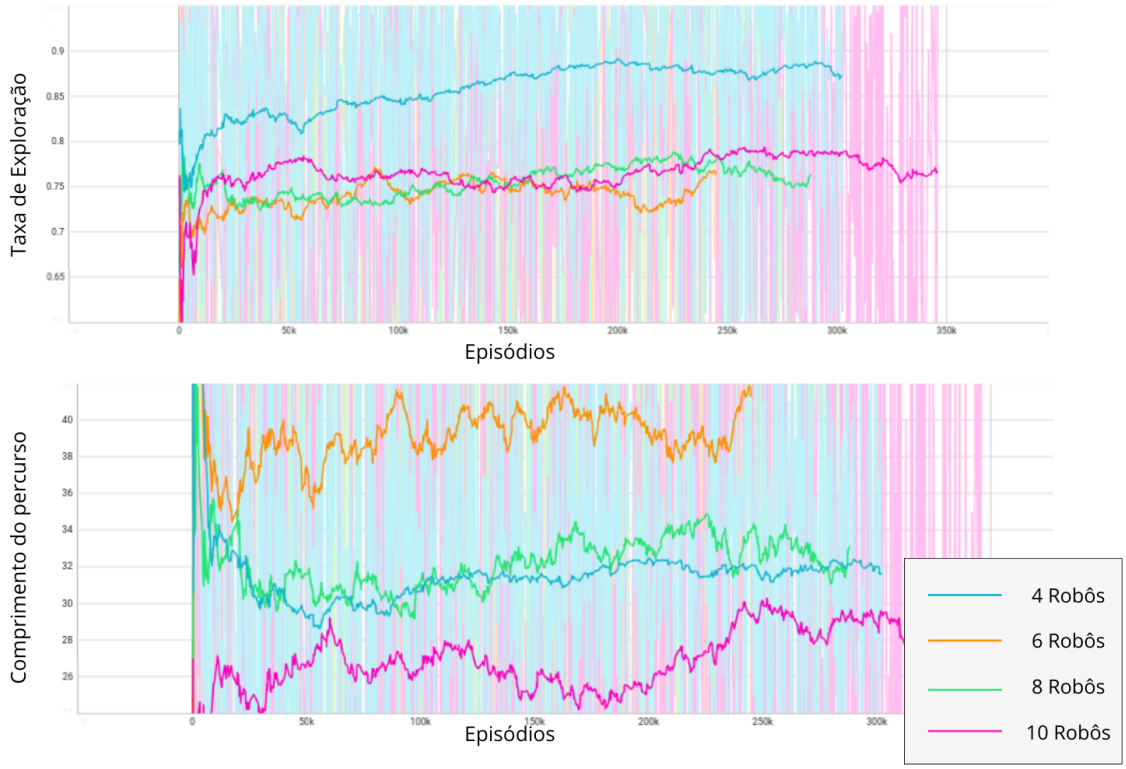
Tabela 4.3: Comparação do teste para o Mapa Intermediário com múltiplos robôs

Mapa	Nª Robôs	Politica	Taxa de Exploração		Comprimento m	
			Média	Std	Média	Std
Mapa Intermediário	2	Dois agentes treinados no Mapa Intermediário	0.85	0.13	9.4	3.60
	4	Quatro agentes treinados no Mapa Intermediário	0.83	0.09	5.01	2.48
	6	Seis agentes treinados no Mapa Intermediário	0.72	0.14	4.0	2.35
	8	Oito agentes treinados no Mapa Intermediário	0.70	0.16	4.42	3.44
	10	Dez agentes treinados no Mapa Intermediário	0.69	0.17	5.3	3.16

ou robôs rodearem uns aos outros incentivando um movimento repetitivo. Para testar está hipótese, iremos realizar outro teste com o Mapa Intermediário, iremos utilizar o mesmo metódo de recompensa do primeiro teste com algumas modificações. Primeiro, iremos aumentar o número de passos para 10M. Segundo, para o conjunto de robôs seis, oito e dez, iremos aumentar o tamanho do Mapa Intermediário para ser 1.5 do tamanho original, porém mantendo suas proporções para que o nível de complexidade não seja modificado. A Escolha de aumentar o mapa 1.5 vezes foi escolhida para que o espaço entre os conjuntos de robôs e áreas não descobertas no inicio de cada turno, fosse proporcionalmente parecida com o conjunto de 4 robôs ao mapa original, já que esse foi o conjunto com o melhor resultados. O teste foi realizado para os conjuntos de quatro a dez robôs, e o resultado pode ser visto na Figura 4.12

Com os resultados dos Gráficos 4.12 e a Tabela 4.4, podemos ver que os resultados para os conjuntos com um número maiores de robôs foi consideravelmente melhor do que o primeiro teste utilizando o Mapa Intermediário. Como o conjunto de oito e dez robôs, que antes a taxa de exploração ficou entorno dos 65% agora ficam à cima dos 75%, um ganho de 10%. Não apenas isso, mas também com um comprimento de passos próximos ao original, mesmo com um mapa maior. O conjunto de robôs seis foi o que teve

Figura 4.12: Treinamento cooperação para diversos conjuntos de robôs com Mapa Intermediário proporcional ao número de robôs



menor melhora, com um aumento de 5% da taxa de exploração, mantendo o comprimento próximo ao teste anterior. Enquanto o conjunto de quatro robôs, continua sendo o melhor conjunto, com a taxa de exploração acima dos 85% e comprimento do percurso entorno dos 30 passos. Vemos que o modificar o tamanho do mapa teve um grande impacto positivo para os casos de teste.

Tabela 4.4: Comparação do segundo teste para o Mapa Intermediário com múltiplos robôs

Mapa	Nª Robôs	Politica	Taxa de Exploração		Comprimento m	
			Média	Std	Média	Std
Mapa Intermediário	4	Quatro agentes treinados no Mapa Intermediário 10M	0.87	0.10	5.26	2.27
Mapa Intermediário x1.5	6	Seis agentes treinados no Mapa Intermediário x1.5 10M	0.79	0.10	5.76	3.54
	8	Oito agentes treinados no Mapa Intermediário x1.5 10M	0.75	0.16	4.15	2.87
	10	Dez agentes treinados no Mapa Intermediário x1.5 10M	0.77	0.17	4.04	2.86

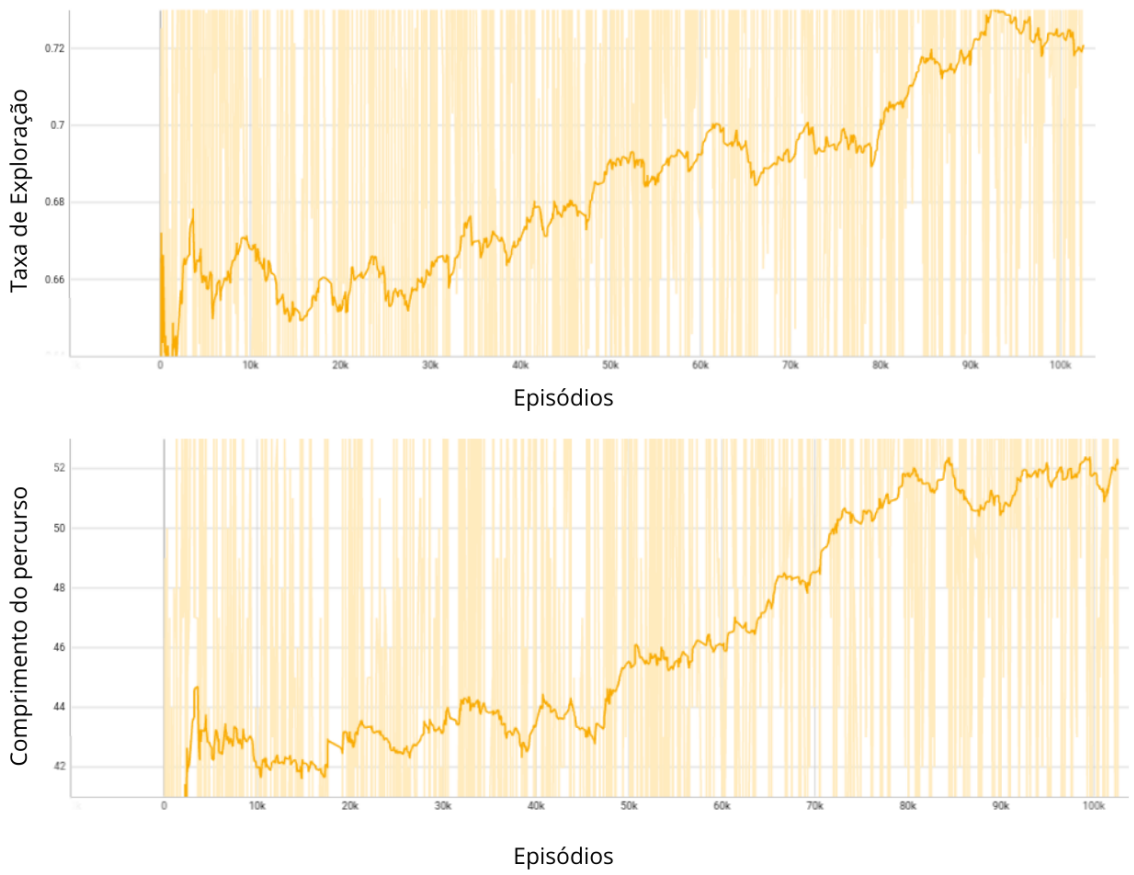
4.4.2 Mapa Teste 1

Para o Mapa de Teste 1, será realizado um teste um pouco diferente, apenas o conjunto de dois agentes será treinado no Mapa de Teste 1. Da mesmo forma que o Mapa Intermediário, será utilizado o mesmos hiperparâmetros e método de recompensa. Como o mapa é mais complexo, o treinamento irá durar até os agentes completarem 5M passos. O resultado do treinamento pode ser visto na Figura 4.13.

Percebe-se que devido a complexidade do mapa, os melhores resultados da taxa de exploração foram entorno dos 72%, percorrendo um pouco menos de 50 passos. Agora utilizaremos está rede neural para os outros conjuntos de agentes nos testes de inferência, que também será testada 50 vezes. A ideia é ver o quanto um diferente números de agentes afeta a qualidade da rede treinada.

Vemos pela Tabela 4.5, que o conjunto de quatro agentes foi o que obteve o melhor resultado entre os demais, com a média de exploração em 0.71 e 6 m de comprimento de percurso, uma diminuição de aproximadamente de 40% comparada com dois agentes. Podemos ver quanto maior o número de agentes, pior fica o resultado, ou seja, os agentes não se adaptaram bem a rede neural e não reagiram de maneira mais cooperativa.Isso

Figura 4.13: Treinamento cooperação no Mapa Teste 1 com 2 robôs



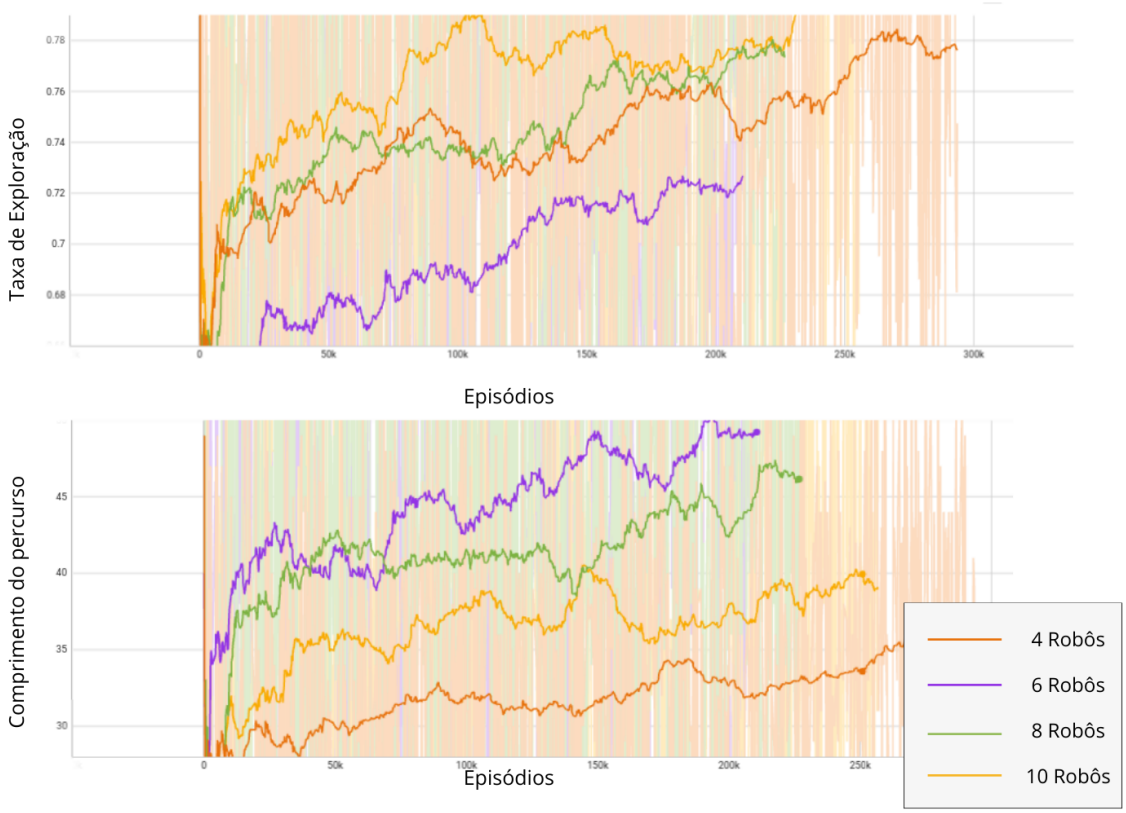
deve ocorrer, pois o algoritmo de PPO incentiva mais competitividade entre os agentes em vez de cooperação quando o treinamento não é realizado com o número de agentes designado, consequentemente o agente ao tomar a decisão que aumenta a recompensa para ele mesmo, pode não ser muitas vezes a melhor decisão para o conjunto como o todo. O mapa também pode acarretar pior resultados para o grupo maior de agentes.

Para verificar isso, vamos comparar agora, o quanto os conjuntos se saem melhor no Mapa Teste 1, se utilizarmos nos testes de inferência redes neurais treinadas para cada conjunto conforme foi feito o segundo teste no Mapa Intermediário. Ou seja, com 10M de passos e mapas proporcionais ao número de robôs. A Figura 4.14 mostra os resultados do treinamento enquanto a Tabela 4.6 mostra o resultado do teste de inferência. Vemos pela Tabela 4.6, que os resultados são bem melhores se utilizarmos redes neurais treinadas não apenas com o conjunto certo de robôs, mas também, como o caso do Mapa Intermediário, com um mapa de proporções adequadas ao número de robôs. Por exemplo, os conjuntos de robôs oito e dez que estavam abaixo de 60% a taxa de exploração passaram para a taxa de exploração entorno dos 70%, mantendo o número médio de passos bem próximo,

Tabela 4.5: Comparação do teste para o Mapa Teste1 com múltiplos robôs

Mapa	Nª Robôs	Politica	Taxa de Exploração		Comprimento m	
			Média	Std	Média	Std
Mapa Teste 1	2	Dois agentes treinados no Mapa Teste 1	0.72	0.12	10.01	4.31
	4	Dois agentes treinados no Mapa Teste 1	0.71	0.12	6.0	2.24
	6	Dois agentes treinados no Mapa Teste 1	0.69	0.11	5.5	2.11
	8	Dois agentes treinados no Mapa Teste 1	0.54	0.12	5.4	2.41
	10	Dois agentes treinados no Mapa Teste 1	0.53	0.12	5.3	2.41

Figura 4.14: Treinamento cooperação no Mapa Teste 1 com múltiplos robôs



mesmo sendo um mapa maior.

Tabela 4.6: Comparação do segundo teste para o Mapa Teste1 com múltiplos robôs

Mapa	Nª Robôs	Politica	Taxa de Exploração		Comprimento m	
			Média	Std	Média	Std
Mapa Teste 1	4	Quatro agentes treinados no Mapa Teste 1 10M	0.77	0.12	9.9	4.4
Mapa Teste 1 x 1.5	6	Seis agentes treinados no Mapa Teste 1 1 x1.5 10M	0.73	0.12	9.08	3.86
	8	Oito agentes treinados no Mapa Teste 1 x.15 10M	0.7	0.10	6.4	3.45
	10	Dez agentes treinados no Mapa Teste 1 x1.5 10M	0.71	0.10	5.9	3.23

4.4.3 Mapa Teste 2

Por fim, o último teste, será utilizar a rede treinada para o Mapa Teste 1 e utilizar para o teste de inferência do Mapa Teste 2. Dessa forma, também veremos como diferentes de conjuntos de agentes reagem a um mapa na qual a rede não foi treinada e se conseguem se adaptar as diferenças.

Tabela 4.7: Comparação do teste para o Mapa Teste2 com múltiplos robôs

Mapa	Nª Robôs	Politica	Taxa de Exploração		Comprimento m	
			Média	Std	Média	Std
Mapa Teste 2	2	Dois agentes treinados no Mapa Teste 1	0.7	0.15	6.7	2.04
	4	Dois agentes treinados no Mapa Teste 1	0.64	0.14	4.4	1.83
	6	Dois agentes treinados no Mapa Teste 1	0.66	0.12	4.62	1.17
	8	Dois agentes treinados no Mapa Teste 1	0.57	0.1	4.3	1.1
	10	Dois agentes treinados no Mapa Teste 1	0.5	0.1	4.09	1.38

Diferente do Mapa Teste 1, o melhor resultado é o conjunto de dois robôs que, mesmo sendo um mapa diferente do que a rede foi treinada. Como no caso anterior, conforme o número de agentes aumenta, pior o resultado comparativo, principalmente se compararmos as médias da taxa de exploração em relação com a médias de comprimento de trajetória entre os diferentes conjuntos. Novamente, a qualidade de cooperação entre os agentes, diminui conforme o número de agentes aumentam. Comprovando, novamente, a tendência onde ao agente toma a melhor decisão para si,porém ela não é a melhor decisão para o todo do conjunto.

Tabela 4.8: Comparação do segundo teste para o Mapa Teste2 com múltiplos robôs

Mapa	Nª Robôs	Politica	Taxa de Exploração		Comprimento m	
			Média	Std	Média	Std
Mapa Teste 2	4	Quatro agentes treinados no Mapa Teste 1 10M	0.66	0.14	4.7	1.87
Mapa Teste 2 x1.5	6	Seis agentes treinados no Mapa Teste 1 1 x1.5 10M	0.64	0.12	4.4	1.85
	8	Oito agentes treinados no Mapa Teste 1 x.15 10M	0.63	0.12	4.8	2.2
	10	Dez agentes treinados no Mapa Teste 1 x1.5 10M	0.62	0.09	4.4	2.1

Na Tabela 4.8, fizemos o mesmo teste de inferência, porém utilizamos as políticas de cada conjunto do robô para o Mapa Teste 1 e também o mapa com tamanho adequado para os conjuntos maiores dos robôs. Vemos pela tabela, que o resutlado foi o que houve menos melhora como um todo. Nenhum dos conjuntos dos robôs ficou acima dos 70% na taxa de exploração, com isso vemos que mesmo utilizando o melhor método para o conjunto de robôs o resultado para um mapa diferente, neste caso o Mapa Teste 2, tende a ter uma grande perda de eficiência na exploração.

5 CONCLUSÃO

Esta dissertação teve como objetivo investigar se o algoritmo de Otimização de Política Proximal pode permitir estratégias de exploração sem mapa eficientes e robustas para múltiplos conjuntos de robôs. Após pesquisar na literatura trabalhos relacionadas ao PPO, observou-se que o algoritmo PPO, como outras formas de aprendizado por reforço, tem aumentado nos últimos anos em aplicações de exploração robótica móvel, mostrando que é uma área com bastante oportunidades atualmente.

Com base nas informações coletadas, propusemos um framework de exploração sem mapa de ponta a ponta baseado em DRL e adequado para robôs em equipes com diferentes tamanhos. A arquitetura foi treinada e testada em ambientes de simulados utilizando diferentes mapas. Nossa solução possibilitou a exploração com eficiência com diversos conjuntos de robôs. Vimos que o tamanho do mapa utilizado e politicas escolhidas foram a chave para um resultado satisfatório, onde mesmo se houvesse uma pequena queda de taxa de exploração, era recompensada por um comprimento de trajetória menor.

Em trabalhos futuros, exploraremos as limitações do método de comunicações entre robôs para analisar em quais casos o método se torna desfavorável para a aprendizagem e quais alternativas podem ser tomadas para melhor resultados. Também, pretendemos avaliar novas técnicas de aprendizado por reforço, principalmente aquelas com foco em aumentar cooperação entre os agentes para melhorar os casos com um número de agentes mais altos. Um exemplo desses aprendizado é o modelo MA-POCA (COHEN ERVIN TENG, 2022), um método relativamente novo, com trabalho lançado em 2022. Assim, as experiências acima abrem o caminho para o objetivo mais importante, que é testar a estratégia de exploração em robôs reais.

REFERÊNCIAS

ABADI ASHISH AGARWAL, P. B. E. B.-Z. C. C. C. G. S. C. A. D. J. D. M. D. e. a. M. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. **arXiv preprint arXiv:1603.04467**, 2016.

ALATISE, G. P. H. M. A review on challenges of autonomous mobile robot and sensor fusion methods. **IEEE Access, IEEE, v. 8, p. 39830–39846**, 2020.

CAI, Y.; YANG, S. X.; XU, X. A combined hierarchical reinforcement learning based approach for multi-robot cooperative target searching in complex unknown environments. **2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)**, 2013.

CHEN, B. S. e. A.-H. T. Z. End-to-end deep reinforcement learning for multi-agent collaborative exploration. **2019 IEEE International Conference on Agents (ICA)**, 2019.

COHEN ERVIN TENG, V.-P. B. R.-P. D. H. H. M. M. A. Z. S. G. A. On the use and misuse of absorbing states in multi-agent reinforcement learning. **arXiv:2111.05992**, 2022.

GARAFFA, L. C. Reinforcement learning for mobile robotics exploration: A survey. **IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, p.1**, 2021.

GARAFFA, L. C. Evaluation of mapless navigation for unknown indoor environment exploration by single and multiple autonomous mobile robots using deep reinforcement learning. **Porto Alegre: PPGC da UFRGS**, 2022.

GAUTAM, S. M. A. A review of research in multi-robot systems. **ICIIS Conference**, 2012.

HU SHIJI SONG, C. L. P. C. H. Plume tracing via model-free reinforcement learning method. **IEEE transactions on neural networks and learning systems, IEEE, v. 30, n. 8, p. 2515–2527**, 2019.

MOHTASIB GERHARD NEUMANN, H. C. A. A study on dense and sparse (visual) rewards in robot policy learning. **Annual Conference Towards Autonomous Robotic Systems.**, 2021.

PUTERMAN, M. L. Markov decision processes, discrete stochastic dynamic programming. **University of British Columbia, WILEYINTERSCIENCE A JOHN WILEY SONS, INC., PUBLICATION, p.1-3**, 1994.

SCHULMAN FLIP WOLSKI, P. D. A. R.-O. K. J. Proximal policy optimization algorithms. **arXiv preprint arXiv:1707.06347**, 2017.

SUTTON, A. G. B. R. S. Reinforcement learning: An introduction. **[S.l.]: MIT press**, 2018.

WALKER FERNANDO VANEGAS ALVAREZ, L. F. G.-S. K. O. Multi-uav target-finding in simulated indoor environments using deep reinforcement learning. **IEEE Aerospace Conference. [S.l.] p.1-9**, 2020.