CENTRO UNIVERSITÁRIO FEI

DOUGLAS DE RIZZO MENEGHETTI

ESPECIALIZAÇÃO DE COMUNICAÇÃO E POLÍTICAS EM APRENDIZADO POR REFORÇO COM MÚLTIPLOS AGENTES HETEROGÊNEOS UTILIZANDO REDES NEURAIS DE GRAFOS

São Bernardo do Campo

2021

DOUGLAS DE RIZZO MENEGHETTI

ESPECIALIZAÇÃO DE COMUNICAÇÃO E POLÍTICAS EM APRENDIZADO POR REFORÇO COM MÚLTIPLOS AGENTES HETEROGÊNEOS UTILIZANDO REDES NEURAIS DE GRAFOS

Tese de Doutorado, apresentada ao Centro Universitário da FEI para obtenção do título de Doutor em Engenharia Elétrica. Orientado pelo Prof. Dr. Reinaldo Augusto da Costa Bianchi.

São Bernardo do Campo

2021

De Rizzo Meneghetti, Douglas.
Especialização de comunicação e políticas em aprendizado por reforço com múltiplos agentes heterogêneos utilizando redes neurais de grafos / Douglas De Rizzo Meneghetti. São Bernardo do Campo, 2021.
136 p. : il.

Tese - Centro Universitário FEI.
Orientador: Prof. Dr. Reinaldo Augusto da Costa Bianchi.

1. Sistemas multi-agentes. 2. Aprendizado por reforço. 3. Agentes heterogêneos. 4. Redes neurais de grafos. I. da Costa Bianchi, Reinaldo Augusto, orient. II. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

centro universitário FEI

APRESENTAÇÃO DE TESE
ATA DA BANCA EXAMINADORA

Programa de Pós-Graduação Stricto Sensu em Engenharia Elétrica

Doutorado

PGE-10

Aluno: Douglas de Rizzo Meneghetti **Matrícula:** 517102-0

Título do Trabalho: ESPECIALIZAÇÃO DE COMUNICAÇÃO E POLÍTICAS EM APRENDIZADO POR REFORÇO COM MÚLTIPLOS AGENTES HETEROGÊNEOS UTILIZANDO REDES NEURAIS DE GRAFOS.

Área de Concentração: Inteligência Artificial Aplicada à Automação e Robótica

Orientador: Prof. Dr. Reinaldo Augusto da Costa Bianchi

Data da realização da defesa: 12/08/2021

ORIGINAL ASSINADA

Avaliação da Banca Examinadora

A banca foi realizada no dia 12 de agosto de 2021 às 9:00 horas, e se iniciou com a apresentação do aluno, que foi muito boa, e seguiu para a arguição, onde o aluno respondeu a todas as questões de forma adequada demonstrando conhecimento pleno do tema. Foram sugeridas melhorias em relação ao texto para a versão final.

A aprovação foi por unanimidade.

São Bernardo do Campo, / / .

MEMBROS DA BANCA EXAMINADORA	
Prof. Dr. Reinaldo Augusto da Costa Bianchi	Ass.: _____
Prof. Dr. Flávio Tonidandel	Ass.: _____
Prof. Dr. Plinio Thomaz Aquino Júnior	Ass.: _____
Prof. Dr. Paulo Drews Junior	Ass.: _____
Prof. Dr. Carlos Henrique Costa Ribeiro	Ass.: _____

A Banca Examinadora acima-assinada atribuiu ao aluno o seguinte:

APROVADO ☒ REPROVADO ☐

VERSÃO FINAL DA TESE

ENDOSSO DO ORIENTADOR APÓS A INCLUSÃO DAS RECOMENDAÇÕES DA BANCA EXAMINADORA

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Carlos Eduardo Thomaz

Dedico este trabalho aos meus pais.

AGRADECIMENTOS

Ao meu orientador Reinaldo Augusto da Costa Bianchi, pela confiança depositada em um orientando extremamente perdido.

Aos professores Plinio Thomaz Aquino Junior, Carlos Eduardo Thomaz, Paulo Eduardo Santos, Flavio Tonidandel e Paulo Sérgio Silva Rodrigues pelas discussões relacionadas a meu trabalho que possibilitaram seu progresso.

Aos professores Paulo Jorge Lilles Drews Junior e Carlos Henrique Costa Ribeiro, por tomarem o tempo de avaliarem meu trabalho, e aos revisores de meus artigos, cujas contribuições persistiram nesta tese.

Aos meus colegas da FEI, que me viram trilhar este caminho, ou o trilharam comigo. Uma lista não-exaustiva e em ordem pseudo-aleatória daqueles que me acompanharam todos esses anos inclui Danilo, Aislan, Isaac, Buzuti, Vilão, Bruno, Jeff, Amanda, Fagner, Laércio, Jonas, Andrey, Rodrigo, Rodrigo, Renato, Ricardo, Contador, Thiago, Marina e Lanterna. Agradeço especialmente à Lígia, André e Anjoletto, pela amizade que se estendeu para além da instituição, e à Kimberlin, que me apontou a eficácia do apoio mútuo em situações desafiadoras.

Como resultado de um trabalho realizado em tempos tão reclusos, reconheço as contribuições que os grupos *online* de aprendizado de máquina trouxeram para meu trabalho. Em especial os membros do grupo *Reinforcement Learning Discussion* no Discord, que me permitiram contribuir nas discussões desta área à qual me dediquei nos últimos anos.

Aos membros da comunidade de software científico aberto que me permitiram contribuir com seus trabalhos e que inadvertidamente contribuíram com o meu: Mikayel Samvelyan (SMAC), Daniel Marques (*biblatex-abnt*), Mathias Fey (*PyTorch Geometric*) e professor Santiago Ontañón (microRTS). Agradecimentos especiais ao criador da classe LaTeX da FEI.

Aos meus amigos, Saulo e André, que se mantiveram companheiros enquanto eu executava um trabalho exótico demais para ser tópico de conversas normais.

Em penúltimo, agradeço aos meus pais por me acompanharem diariamente nessa jornada e me apoiarem em todas as decisões que tomei pelo caminho, assim como por nutrirem um nível saudável de preocupação ao testemunharem as exteriorizações de minhas inseguranças inerentes do mundo acadêmico.

E, por fim: o presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

“A situação mais simpática é aquela em que as pessoas não se envergonham umas das outras, mas agem franca e abertamente. E para quê enganar-se? É a mais vã e imprudente das ocupações.”

Fiódor Dostoiévski

“Ora et labora.”

Bento de Núrsia

“Eu sou eu e minha circunstância, e se não salvo a ela, não me salvo a mim.”

José Ortega y Gasset

RESUMO

Esta tese apresenta uma arquitetura de rede neural voltada ao aprendizado de políticas em sistemas multi-agentes totalmente cooperativos, compostos de agentes heterogêneos e comunicativos. O ambiente é formalizado como um Processo de Decisão de Markov Parcialmente Observável Descentralizado e os estados transformados em grafos direcionados rotulados atribuídos de agentes e entidades. Vértices representam agentes e entidades; os rótulos dos vértices, suas classes, sendo todos os agentes dentro de uma mesma classe considerados homogêneos entre si; arcos direcionados representam a capacidade dos agentes de adquirir informação de outras entidades; e vetores armazenados nos vértices representam as características que descrevem agentes e entidades, ou as observações dos agentes.

A topologia de rede neural proposta usa camadas totalmente conectadas para codificar as observações dos agentes; convoluções relacionais em grafos para aprender mecanismos de comunicação específicos para diferentes pares de classes; e diferentes redes neurais treinadas utilizando aprendizado por reforço para modelar as políticas das classes de agentes.

A tese apresenta dois métodos. No primeiro, os módulos de codificação e aprendizado de funções valor-ação são modelados como redes neurais distintas para cada classe de entidade e agente, e o treinamento do modelo é feito utilizando uma memória de repetição de transições. O segundo método usa compartilhamento de parâmetros entre as classes de agentes para obter uma rede neural com menos parâmetros, assim como emprega camadas recorrentes e treinamento com amostras de uma memória de repetição de episódios.

A comunicação relacional é comparada à comunicação realizada através de mecanismos de atenção e à ausência de comunicação entre os agentes. Também é testada a compatibilidade do método com outras contribuições disponíveis na literatura, como a regularização por relações temporais e o *mixing* aditivo.

Testes realizados no ambiente do *StarCraft Multi-Agent Challenge* demonstram que o emprego de camadas de convolução relacionais para a especialização da comunicação entre agentes viabiliza desempenho comparável ou superior aos outros métodos em todos os cenários testados, principalmente naqueles com maior número de classes de agentes. Já a combinação da comunicação relacional com o *mixing* aditivo apresentou, geralmente, os melhores resultados.

Palavras-chave: Sistemas multi-agentes. Aprendizado por reforço. Agentes heterogêneos. Redes neurais de grafos.

ABSTRACT

This thesis presents a neural network architecture specialized in learning policies for agents in fully cooperative multi-agent systems, composed by heterogeneous, communicative agents. The environment is formalized as a Decentralized Partially Observable Markov Decision Process and states are transformed into directed labeled attributed graphs of agents and entities, in which vertex labels represent agent/entity classes. Agents that share a single class are considered homogeneous among themselves; directed arcs represent an agent's capacity of acquiring information from other entities; and vectors stored in vertices represent the features that describe the agents and entities, or agents' observations.

The proposed neural network topology uses fully connected layers to encode agent features and observations; relational graph convolutions to learn specific communication protocols for different pairs of agent classes; and different neural networks, trained using reinforcement learning, to model agent class policies.

The thesis presents two methods. The first one uses separate neural networks to encode and learn policies for each agent/entity class and training is realized through a replay buffer of state transitions. The second version employs parameter sharing between agent classes to attain a neural network with fewer parameters, as well as recurrent layers and training via a replay buffer of complete episodes.

Relational communication is compared to communication via an attention mechanism and no communication. Compatibility with other contributions provided in the literature is also tested, such as temporal relation regularization and additive mixing.

Tests performed in the *StarCraft Multi-Agent Challenge* environment demonstrate that employing relational graph convolutions with specialization of communication protocols attains comparable or superior performance in all the tested scenarios, specially in the ones with higher number of agent classes. Furthermore, the combination of relational communication with additive mixing achieved, in general, the best results.

Keywords: Multi-agent systems. Reinforcement learning. Heterogeneous agents. Graph neural networks.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ANOVA	Análise de Variância (*Analysis Of Variance*)
APM	ações por minuto (*actions per minute*)
BPTT	*Retropropagação através do Tempo* (*Backpropagation Through Time*)
BWAPI	*Brood War Application Programming Interface*
CRT	Campo Receptivo Total
CTA	Comunicação Total Entre Agentes
DDQN	Rede Q Profunda Dupla (*Double Deep Q-Network*)
Dec–MDP	Processo de Decisão de Markov Descentralizado (*Decentralized Markov Decision Process*)
Dec–POMDP	Processo de Decisão de Markov Parcialmente Observável Descentralizado (*Decentralized Partially Observable Markov Decision Process*)
DQN	Rede Q Profunda (*Deep Q-Network*)
DRL	Aprendizado por Reforço Profundo (*Deep Reinforcement Learning*)
DRQN	Rede Q Profunda Recorrente (*Deep Recurrent Q-Network*)
EPM	ações efetivas por minuto (*effective actions per minute*)
GAT	Atenção em Grafos (*Graph Attention*)
GNN	Rede Neural de Grafos (*Graph Neural Network*)
GPL	Aprendizado de Política baseado em Grafos (*Graph-based Policy Learning*)
HCA-Net	Rede Neural de Agentes Heterogêneos e Comunicativos (*Neural Network for Heterogeneous Communicative Agents*)
HCA-Net(AO)	Rede Neural de Agentes Heterogêneos e Comunicativos – Observações de Agentes (*Neural Network for Heterogeneous Communicative Agents – Agent Observations*)
HCA-Net(EF)	Rede Neural de Agentes Heterogêneos e Comunicativos – Características de Entidades (*Neural Network for Heterogeneous Communicative Agents – Entity Features*)
HSD	Diferença Honestamente Significativa (*Honestly Significant Difference*)
IQL	*-Learning* Independente (*Independent -Learning*)
LSTM	Memória a curto prazo longa (*Long short-term memory*)
MADRL	Aprendizado por Reforço Profundo Multi-Agentes (*Multi-Agent Deep Reinforcement Learning*)

MARL	Aprendizado por Reforço Multi-Agentes (*Multi-Agent Reinforcement Learning*)
MDP	Processo de Decisão de Markov (*Markov Decision Process*)
MLP	Perceptron Multi-Camadas (*Multi-Layer Perceptron*)
MMDP	Processo de Decisão de Markov Multi-Agentes (*Multi-Agent Markov Decision Process*)
MPNN	Rede Neural de Passagem de Mensagens (*Message-Passing Neural Network*)
MTDP	Processo de Decisão de Time Multi-Agentes (*Multi-Agent Team Decision Process*)
ORTS	*Open RTS*
PER	Repetição de Experiência Priorizada (*Prioritized Experience Replay*)
POMDP	Processo de Decisão de Markov Parcialmente Observável (*Partially Observable Markov Decision Process*)
POSG	Jogo Estocástico Parcialmente Observável (*Partially Observable Stochastic Game*)
PPO	*Proximal Policy Optimization*
RGCN	Convolução Relacional em Grafos (*Relational Graph Convolution*)
RL	Aprendizado por Reforço (*Reinforcement Learning*)
RMDP	Processo de Decisão de Markov Relacional (*Relational Markov Decision Process*)
RODE	*Roles to Decompose*
ROMA	Aprendizado por Reforço Multi-Agentes Orientado a Papéis (*Role-Oriented Multi-Agent Reinforcement Learning*)
ROS	*Robot Operating System*
RTS	Estratégia em Tempo Real (*Real-Time Strategy*)
SC2LE	*StarCraft II Learning Environment*
SG	Jogo Estocástico (*Stochastic Game*)
SMAC	*StarCraft Multi-Agent Challenge*
TRR	Regularização por Relações Temporais (*Temporal Relation Regularization*)
VDN	Rede de Decomposição de Valor (*Value Decomposition Network*)

LISTA DE SÍMBOLOS

$\varnothing$	conjunto vazio
A	conjunto de ações/ações conjuntas
a	ação
$\mathcal{A}$	função vantagem
$\boldsymbol{a}$	ações de todos os agentes em um mesmo instante
$A(u, s)$	ações legais do agente u no estado s
A_c	conjunto de ações para classe de agentes c
b	um lote de treinamento
C	conjunto de classes
$\mathcal{D}$	domínio de aplicação de um grafo heterogêneo de agentes e entidades
D	memória de repetição
e	uma aresta ou arco
$\mathcal{E}$	conjunto de arestas de um grafo
$\mathbb{E}$	expectativa
$\mathcal{G}$	grafo
G	retorno esperado
H	modelo de observações
HCANet	chamada à HCA-Net nos algoritmos
J	função de erro
$\mathcal{L}$	função de erro total
$\mathcal{M}$	uma instância de um Dec–POMDP
$\mathcal{N}^-$	vizinhança de entrada de um vértice
$\mathcal{N}^+$	vizinhança de saída de um vértice
O	conjunto de observações
o	observação
$\boldsymbol{o}$	observações de todos os agentes em um mesmo instante
P	função de probabilidade de transição
Q	função valor
Q_*	função valor ótima
Q_c	função valor para classe de agentes c

$\mathbb{R}$	domínio dos números reais
r	recompensa
R	função de recompensa
S	conjunto de estados
s	estado
T	conjunto de instantes de tomada de decisão
t	instante de tomada de decisão
u	agente
U	conjunto de agentes
V	função valor-estado
V_*	função valor-estado ótima
v	vértice
$\mathcal{V}$	conjunto de vértices de um grafo
Z	conjunto de classes de agentes
Ψ	variáveis de observação conjuntas
Ψ_c	variáveis de observação para classe de agentes c
Ω	conjunto de relações entre vértices
α	taxa de aprendizado da rede neural
γ	fator de desconto
δ	erro de diferenças temporais
η	coeficiente de auto-atenção
$\boldsymbol{\theta}$	conjunto de parâmetros ajustáveis de um modelo
κ	intervalo de atualização da rede alvo
π	política
π_*	política ótima
π_c	política da classe de agentes c
ρ	um episódio
ω	relação
ϵ	probabilidade de escolha de ação aleatória
ϕ	função de pré-processamento da entrada

SUMÁRIO

1 INTRODUÇÃO

Existem sistemas multi-agentes nos quais os agentes têm acesso a observações parciais dos estados de um sistema, mas precisam selecionar a ação mais apropriada para alcançar seu objetivo, enquanto coordenam seu comportamento com o dos outros agentes. Uma forma de alcançar a coordenação é através da comunicação.

Exemplos desses sistemas incluem equipes de robôs, como *drones* que mapeiam um local para que robôs terrestres possam navegá-lo; ou um time de robôs humanoides jogadores de futebol, em que robôs diferentes são responsáveis por ataque e defesa ao longo do jogo (KITANO et al., 1997). Redes de tráfego de veículos também podem ser modeladas como sistemas multi-agentes, nos quais veículos devem negociar uns com os outros e com o próprio sistema de tráfego, composto pelos semáforos nas intersecções da rede, para alcançarem seus destinos com o mínimo de atraso (YANG, S. et al., 2021). Na área de jogos eletrônicos, times com números variados de agentes, cada um com capacidades distintas devem se coordenar para vencer um time adversário em um cenário competitivo (RASHID et al., 2018; WANG, Tonghan et al., 2020a).

Todos os sistemas listados anteriormente possuem uma característica em comum: eles podem ser modelados como sistemas com agentes heterogêneos, *i.e.* agentes que percebem o ambiente e são capazes de afetá-lo de maneiras distintas ou que possuem o incentivo para interagir com ele de maneiras distintas, quando presentes em situações semelhantes. No entanto, nos sistemas nos quais os agentes cooperam para realizar uma tarefa em comum (chamados neste trabalho de cooperativos ou totalmente cooperativos) ou nos quais suas percepções, ações ou políticas finais possuam alguma similaridade, é interessante acelerar o aprendizado das políticas de todos os agentes usando um método que permita unificar este aprendizado, mesmo quando os agentes são heterogêneos.

Nos últimos anos, diversas pesquisas visaram modelar o problema de Aprendizado por Reforço Multi-Agentes (MARL, *Multi-Agent Reinforcement Learning*) utilizando redes neurais, motivados pela ascensão da área hoje conhecida como Aprendizado Profundo (LECUN; BENGIO; HINTON, 2015). O recente interesse em Aprendizado Profundo também promoveu a expansão e exploração de uma arquitetura de rede neural especializada no processamento de dados representados na forma de grafos (GORI; MONFARDINI; SCARSELLI, 2005), denominada Redes Neurais de Grafos (GNNs, *Graph Neural Network*).

Essas redes neurais foram aplicadas em diversos novos problemas, como: descoberta de novas moléculas (DUVENAUD et al., 2015) e antibióticos (STOKES et al., 2020); simulações

das interações entre objetos (BATTAGLIA, P. et al., 2016) e da transição de estados físicos de certos materiais (BAPST et al., 2020); otimização de *hardware* (MIRHOSEINI et al., 2020); geração de imagens (ASHUAL; WOLF, 2019); classificação semi-supervisionada em redes de citações e grafos do conhecimento (KIPF; WELLING, 2017) e recomendação de produtos em grafos de compradores e suas compras (FAN et al., 2019).

Diversos modelos de GNNs foram categorizados como metodologias de passagens de mensagens entre vértices de grafos (GILMER et al., 2017), sendo alguns desses modelos especializados em grafos heterogêneos (WANG, X. et al., 2020), *i.e.* grafos cujos vértices representam entidades de categorias distintas, ou cujas arestas modelam relações distintas entre entidades.

As GNNs possuem diversas propriedades desejáveis para sistemas de aprendizado por reforço multi-agentes heterogêneos, como capacidade de trabalhar com grafos de tamanhos variados, para sistemas com número variado de agentes, passagem de mensagens entre vértices, análogo à comunicação entre agentes, e processamento de grafos heterogêneos, para sistemas com agentes heterogêneos.

Sob essa perspectiva, esta tese contribui com a área de MARL, abordando um sistema com múltiplos agentes heterogêneos de forma análoga à que redes neurais para grafos heterogêneos são empregadas em trabalhos de aprendizado supervisionado e semi-supervisionado, assim como à forma como redes neurais são empregadas problemas de MARL totalmente cooperativos.

O trabalho se iniciou com a exploração de uma categoria de redes neurais especializadas no processamento de dados representados como grafos, as GNNs (GORI; MONFARDINI; SCARSELLI, 2005; SCARSELLI et al., 2009b,a), assim como suas aplicações em tarefas de aprendizado supervisionado e semi-supervisionado. Desta pesquisa, um primeiro método foi proposto, implementado e testado, o qual se fundamenta na representação dos estados de um sistema multi-agentes como grafos heterogêneos direcionados rotulados atribuídos de agentes e entidades, podendo ambos os grupos serem heterogêneos ou não. Na tentativa de se adaptar o uso das GNNs de suas aplicações originais para um problema de aprendizado por reforço, o grafo armazena em seus vértices os vetores de características dos agentes e entidades. Este método foi denominado de Rede Neural de Agentes Heterogêneos e Comunicativos – Características de Entidades (HCA-Net(EF), *Neural Network for Heterogeneous Communicative Agents – Entity Features*)[1].

[1]Ele foi originalmente nomeado HMAGQ-Net em Meneghetti et al. (2020). A renomeação ocorreu a fim de evitar confusões com um trabalho denominado MAGNet (MALYSHEVA; KUDENKO; SHPILMAN, 2019).

Os grafos são então processados por uma rede neural, também proposta neste trabalho, a qual codifica os vetores de características dos vértices num módulo de codificação; realiza a comunicação entre agentes e entidades num módulo de comunicação composto por camadas de Convolução Relacional em Grafos (SCHLICHTKRULL et al., 2018); e aprende as funções valoração ótimas para todos os agentes. Um algoritmo para o treinamento desta rede neural também é apresentado. Este conjunto de artefatos, assim como os resultados dos experimentos no ambiente *StarCraft Multi-Agent Challenge* são as primeiras contribuições deste trabalho. Elas foram publicadas no XVI Encontro Nacional de Inteligência Artificial e Computacional (MENEGHETTI; BIANCHI, 2020a).

Os resultados dos experimentos expuseram algumas desvantagens deste método, atribuídas às características herdadas das aplicações originais das GNNs que não se transpõem para o Aprendizado por Reforço Profundo Multi-Agentes (MADRL, *Multi-Agent Deep Reinforcement Learning*). Essas limitações motivaram a criação de um novo método, o qual incorporou ideias provenientes de trabalhos recentes na área de MADRL totalmente cooperativo. Este método, denominado Rede Neural de Agentes Heterogêneos e Comunicativos – Observações de Agentes (HCA-Net(AO), *Neural Network for Heterogeneous Communicative Agents – Agent Observations*), modela os estados como grafos apenas de agentes, cujos vetores de dados armazenados nos vértices agora contêm as observações dos agentes. A nova representação dos estados como grafos, assim como a topologia modificada da rede neural e um novo algoritmo de treinamento caracterizam mais um conjunto de contribuições alcançadas nesta tese, as quais foram publicadas no *AAAI-21 Workshop on Reinforcement Learning in Games* (MENEGHETTI; BIANCHI, 2021).

1.1 OBJETIVO

Este trabalho tem por objetivo propôr arquiteturas de redes neurais especializadas no aprendizado de políticas em tarefas de aprendizado por reforço multi-agentes totalmente cooperativas compostas por times de agentes heterogêneos e com capacidade de comunicação.

1.1.1 Contribuições

Esta tese apresenta as seguintes contribuições:

a) duas arquiteturas de redes neurais para aprendizado de políticas em sistemas com múltiplos agentes heterogêneos em cenários totalmente cooperativos, fundamen-

tadas na representação dos estados de um sistema multi-agentes como grafos heterogêneos direcionados rotulados atribuídos:

- HCA-Net(EF) (seção 5), uma arquitetura baseada principalmente em técnicas de aprendizado em redes neurais de grafos, a qual opera em grafos cujos vértices representam agentes e entidades sem agência;
- HCA-Net(AO) (seção 6), que incorpora avanços recentes na área de aprendizado por reforço profundo multi-agentes em tarefas totalmente cooperativas;

b) descrições formais das representações dos estados do ambiente como grafos heterogêneos direcionados rotulados atribuídos e dos algoritmos de treinamento para ambos os métodos;

c) resultados da aplicação de ambos os métodos em uma seleção de cenários do *StarCraft Multi-Agent Challenge*, um ambiente de aprendizado por reforço multi-agentes totalmente cooperativo.

1.2 JUSTIFICATIVA

A classe de problemas que este trabalho aborda é aquela dos sistemas multi-agentes parcialmente observáveis, totalmente cooperativos, com agentes heterogêneos e capacidade de comunicação. Essa classe de problemas possui diversos desafios provenientes da área de Aprendizado por Reforço Multi-Agentes, como coordenação, cooperação, comunicação e atribuição de crédito, os quais são exacerbados quando investigados sob a perspectiva de técnicas da área de Aprendizado Profundo.

Alguns exemplos de desafios adicionais provenientes de operar nesta classe de problemas utilizando redes neurais incluem: a necessidade da rede operar em dados multi-modais, a variação na dimensão dos dados de entrada e no número de parâmetros da rede devido à quantidade de agentes e as diferentes percepções que agentes de tipos distintos podem possuir do ambiente. Este trabalho aborda estes desafios ao propor uma arquitetura de rede neural que acomoda agentes que efetuem sensoriamento e atuação de formas distintas.

Os métodos apresentados nesta tese unem duas áreas de relevância contemporânea: o uso de redes neurais para a solução de problemas de aprendizado por reforço em sistemas multi-agentes heterogêneos (SUNEHAG et al., 2018; FOERSTER, Jakob et al., 2016, 2017a,b; FOERSTER, J. N. et al., 2018b; FOERSTER, J. et al., 2018; FOERSTER, J. N. et al., 2018a; RASHID et al., 2018, 2020a; VINYALS et al., 2019; SAMVELYAN et al., 2019; WANG,

Tonghan et al., 2020b,a; RAHMAN et al., 2020; HU, J. et al., 2021) e o aprendizado de máquina em grafos heterogêneos (SCHLICHTKRULL et al., 2018; ZHANG, C. et al., 2019; CEN et al., 2019; WANG, X. et al., 2019; YUN et al., 2019; FU et al., 2020; HONG et al., 2020; HU, Z. et al., 2020), aplicado à comunicação entre equipes heterogêneas de agentes utilizando apenas redes neurais.

1.3 ORGANIZAÇÃO DO TRABALHO

O restante do trabalho está organizado nas seguintes seções:

a) A seção 2 apresenta a teoria por trás das áreas que embasam a proposta do trabalho;
b) o domínio de testes do trabalho, o *StarCraft Multi-Agent Challenge* (SMAC), é descrito na seção 3;
c) a seção 4 descreve os trabalhos relacionados em diferentes níveis de detalhe;
d) a seção 5 descreve a HCA-Net(EF), um método baseado principalmente em técnicas de aprendizado em redes neurais de grafos, assim como experimentos e resultados;
e) a seção 6 descreve a HCA-Net(AO), um método que incorpora avanços recentes na área de aprendizado por reforço profundo multi-agentes em tarefas totalmente cooperativas, assim como experimentos e resultados;
f) a seção 7 conclui a tese e apresenta propostas de trabalhos futuros;
g) os apêndices A e B apresentam as publicações científicas e as contribuições de *software* livre realizadas durante o doutorado, respectivamente.

2 CONCEITOS FUNDAMENTAIS

A criação de uma rede neural que aprende politicas para múltiplos agentes heterogêneos através de aprendizado por reforço combina múltiplas áreas do conhecimento. Esta seção apresenta tais áreas, a saber: os processos de decisão de Markov totalmente observáveis, parcialmente observáveis e parcialmente observáveis descentralizados, utilizados como primeira etapa da formalização dos domínios nos quais o modelo proposto pode ser aplicado; as redes neurais profundas e os métodos de treinamento *off-policy* dessas redes para que aprendam a resolver tarefas através de aprendizado por reforço; e as redes neurais de grafos e seus métodos de passagem de mensagem.

2.1 PROCESSOS DE DECISÃO MARKOVIANOS E APRENDIZADO POR REFORÇO

Diversos problemas reais podem ser considerados como situações de tomada de decisão sequencial nas quais a escolha informada de decisões leva a resultados ótimos. Para se estudar problemas com essas características, é comum formalizá-los como Processos de Decisão de Markov (MDPs, *Markov Decision Process*) (BELLMAN, 1957).

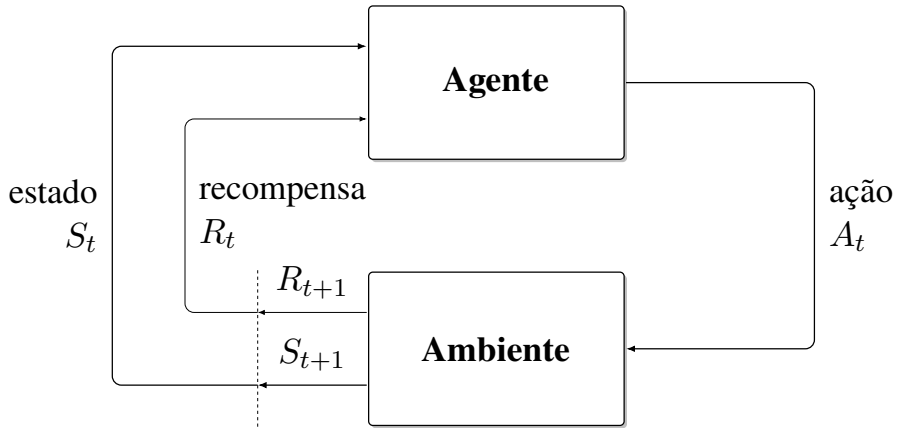
Um MDP é um processo de controle estocástico no qual um agente pode influenciar as dinâmicas de um sistema através da tomada de decisões sequenciais (PUTERMAN, 2005). Mais formalmente:

Definição 1 – MDP. Um MDP é uma tupla (S, A, R, P, γ), onde

a) S: conjunto de estados;
b) A: conjunto de ações;
c) $R \rightarrow S \times A$: função de recompensa;
d) $P(s'|s,a) \rightarrow S \times A \times S$: função de transição;
e) $\gamma \in [0; 1)$: fator de desconto.

A cada instante de tomada de decisão $t \in T$, o sistema se encontra em um dos estados $s \in S$. Um agente, então, observa s, executa uma ação $a \in A$ e observa uma recompensa $r(s, a)$ por executar a em s e o próximo estado $s' \in S$ com probabilidade $P(s'|s, a)$. A figura 1 demonstra diagramaticamente o processo interativo entre agente e ambiente. Para simplificar a tratabilidade dos MDP, pode-se assumir que S e A são discretos e finitos ou contáveis infinitos (PUTERMAN, 2005).

Figura 1 – Representação da interação entre agente e ambiente em um MDP



Fonte: Autor “adaptado de” Sutton e Barto, 2018

A solução de um MDP no instante t consiste em selecionar as ações em todos os instantes de decisão posteriores a t, de forma a maximizar a soma de todas as recompensas futuras, até o fim de um episódio no instante T ou até um horizonte infinito, em problemas não-episódicos. A essa quantia dá-se o nome de retorno e ela é descrita da seguinte forma (SUTTON; BARTO, 2018)

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

onde um fator de desconto $\gamma \in [0; 1)$ é utilizado para se limitar o horizonte de observações. Uma vez que diferentes valores de γ alteram a solução do MDP, ele é comumente considerado parte da definição do MDP (PUTERMAN, 2005).

À função que dita quais ações um agente toma, dada a observação do estado atual dá-se o nome de política. Ela é representada por π. $\pi(s|a)$ indica a probabilidade do agente escolher a quando se encontra em s. À política que maximiza o retorno esperado de um MDP dá-se o nome de política ótima. Ela é representada por π_*. Um MDP pode ter mais de uma política ótima, sendo todas representadas por π_*. Todo MDP possui uma política ótima determinística (SUTTON; BARTO, 2018).

Quando todas as dinâmicas do sistema são conhecidas, é possível encontrar a política ótima de um MDP utilizando programação dinâmica, uma série de técnicas voltadas à execução de planejamento em MDPs. Em outras palavras, programação dinâmica é aplicável a um MDP quando os elementos (S, A, R, P) são conhecidos (SILVER, 2015).

No entanto, em diversos problemas não há conhecimento completo do MDP. Nestes casos, é possível aplicar técnicas que aprendem (direta ou indiretamente) as dinâmicas do sistema

e também encontram a política ótima de um MDP. Este aprendizado é realizado através de experiência e interações com o MDP. A este conjunto de técnicas dá-se o nome de Aprendizado por Reforço (RL, *Reinforcement Learning*) (SUTTON; BARTO, 2018).

Tanto os métodos advindos da programação dinâmica como de RL encontram π_* aproximando funções que indicam preferências por estados ou pela execução de ações em estados. Estas funções denominam-se função valor-estado V e função valor-ação Q. Mais formalmente, a função-valor $V_\pi(s)$ de um estado s, dada uma política π indica o retorno cumulativo esperado de um agente ao se encontrar em s e seguir π até um estado terminal.

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{2}$$

De maneira similar, $Q_\pi(s, a)$ indica o retorno cumulativo esperado de se encontrar s, tomar a ação a e seguir π até um estado terminal.

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{3}$$

A política ótima π_* é aquela que maximiza os valores de V e Q, ou seja, $\pi_* = \arg\max_\pi V_\pi(s)$ e $\pi_* = \arg\max_\pi Q_\pi(s, \pi(s))$.

O método de aprendizado por diferenças temporais é um algoritmo de RL que encontra a função valor-estado ótima V_π de uma política π. Algoritmos que expandem o método de aprendizado por diferenças temporais para os casos de controle (o aprimoramento de uma política, dada sua função valor) aprendendo a função valor-ação ótima Q_* e consequentemente π_*, incluem Sarsa (RUMMERY; NIRANJAN, 1994), Q-*Learning* (WATKINS; DAYAN, 1992) e suas variações.

Métodos que não aproximam V_* e Q_* explicitamente, mas ainda assim buscam por π_*, incluem os métodos de gradiente de políticas. Neles, a política é parametrizada por um vetor de parâmetros $\boldsymbol{\theta}$ e otimizada calculando-se o gradiente de uma função de desempenho $J(\boldsymbol{\theta})$ com relação a $\boldsymbol{\theta}$ e maximizando-a. Para mais informações referentes aos algoritmos supracitados, o leitor é guiado a Sutton e Barto (2018) e Szepesvári (2010).

2.1.1 Ambientes com observações parciais

Em diversas situações práticas, é irreal esperar que o agente possua informação total sobre o estado no qual o ambiente se encontra. Seja devido à alta complexidade do ambiente ou a

limitações no sensoriamento do agente, este último deve trabalhar com informações incompletas, representadas através de observações parciais dos estados.

Os MDPs com informações incompletas foram propostos pela primeira vez por Åström (1965) e introduzidos à área de inteligência artificial e robótica por Kaelbling, Littman e Cassandra (1998), sob o nome de Processos de Decisão de Markov Parcialmente Observáveis (POMDPs, *Partially Observable Markov Decision Process*).

Definição 2 – POMDP. Um POMDP é uma tupla $(S, A, R, P, \gamma, O, H)$, onde:

a) S: conjunto de estados;
b) A: conjunto de ações;
c) $R \rightarrow S \times A$: função de recompensa;
d) $P(s'|s,a) \rightarrow S \times A \times S$: função de transição;
e) $\gamma \in [0; 1)$: fator de desconto;
f) O: conjunto finito de observações;
g) $H \rightarrow S \times A$: função de observação, a qual dá a a distribuição de probabilidades sobre todas as possíveis observações, dada uma ação executa a e um estado resultante s'. $H(o|s', a)$ representa a probabilidade específica para a observação o.

Dado que o agente não sabe ao certo em qual estado o ambiente se encontra, ele deve planejar suas ações considerando a distribuição de probabilidades de todos os possíveis estados. A esta distribuição dá-se o nome de estado de crença (do inglês *belief state*). Para aumentar a certeza sob o estado no qual o ambiente está, uma estratégia viável nos POMDPs é a execução de ações para fins de coleta de informação (KAELBLING; LITTMAN; CASSANDRA, 1998). Uma categoria de ação viável para coleta de informação em sistemas multi-agentes é a realização de comunicação entre os agentes, como será discutido na seção 2.2.2

2.1.2 *Deep Q-Networks*

Apesar da eficiência comprovada em diversas aplicações, muitos algoritmos clássicos de RL, como Sarsa e Q-*Learning*, costumam aproximar Q_* pelo método tabular, isto é, aproximando valores individuais para cada par (s, a), $\forall s \in S$ e $\forall a \in A$. No entanto, diversos domínios são compostos por um conjunto de estados de alta cardinalidade, devido ao número de características que compõem cada estado, assim como os possíveis valores que cada característica pode assumir.

Para estes domínios, o uso de métodos capazes de modelar funções com entradas de alta dimensão apresenta-se como uma alternativa viável para a aproximação de funções como Q_*, V_* e π_*.

Com os recentes avanços teóricos e tecnológicos que possibilitaram o treinamento de redes neurais cada vez mais complexas e expressivas, deu-se origem a um campo de pesquisa no qual essas redes neurais profundas são utilizadas para a aproximação de Q_* (MNIH et al., 2015), ou para realizar a aproximação de políticas através de gradiente de política utilizando uma rede neural profunda (SCHULMAN et al., 2015, 2017). Também foram implementadas variantes dos algoritmos ator-crítico (KONDA; TSITSIKLIS, 2003–0001) utilizando redes neurais profundas (MNIH et al., 2016). A esta área que utiliza redes neurais profundas na implementação de algoritmos de aprendizado por reforço para utilização em domínios de alta dimensão, deu-se o nome de Aprendizado por Reforço Profundo (DRL, *Deep Reinforcement Learning*).

Os métodos implementados neste trabalho possuem similaridades com as Redes Q Profundas (DQNs, *Deep Q-Network*) (MNIH et al., 2015). O restante desta seção é dedicado a descrevê-las, assim como às melhorias propostas ao algoritmo que também foram incorporadas no trabalho.

O algoritmo DQN utiliza uma rede neural profunda parametrizada por um conjunto de parâmetros ajustáveis $\boldsymbol{\theta}$ para aproximar a função Q_* e aprende a jogar 49 jogos de Atari 2600 com os mesmos hiperparâmetros (MNIH et al., 2015).

O algoritmo 1 descreve como treinar uma rede neural profunda para aproximar a função Q_* utilizando uma rede neural profunda. Um estado s é composto da concatenação de 4 capturas sucessivas da tela do jogo. s é pré-processado por uma função ϕ, a qual recorta a captura da tela de forma a manter apenas uma região de 84×84 píxeis onde a maioria da informação visual dos jogos se concentra. O valor máximo do brilho de duas capturas de tela consecutivas é utilizado para formar uma nova imagem (balanceando um artefato presente nos jogos de Atari 2600, o *flickering*) e uma sequência de quatro dessas imagens consecutivas é utilizada como entrada para a rede neural, preservando um breve histórico de eventos passados em cada estado.

Visto que os estados gerados sequencialmente pelos jogos de Atari possuem alta similaridade entre si, os autores utilizaram uma técnica denominada repetição de experiências (LIN, 1992), na qual uma memória de tuplas (s, a, r, s') é armazenada e, a cada instante de treinamento, um lote de tuplas aleatórias é selecionado para o aprendizado da rede. Foi utilizado uma memória (representada no algoritmo 1 por D) com 1 milhão de tuplas. O uso da repetição de experiências

Algoritmo 1 – Algoritmo DQN. Fonte: Autor “adaptado de” Mnih et al., 2015

Inicializa memória de repetição $D \rightarrow \{\}$
Inicializa pesos $\boldsymbol{\theta}$ da rede Q aleatoriamente
Inicializa pesos $\boldsymbol{\theta}^{-} \leftarrow \boldsymbol{\theta}$ da rede alvo
para cada *episódio* **faça**
    Inicializa $s_1 \leftarrow$ estado inicial
    $\phi_1 \leftarrow \phi(s_1)$
    para cada *instante* t **faça**
        $a_t \leftarrow \begin{cases} \text{aleatória} & \text{com probabilidade } \epsilon \\ \max_a Q(\phi_t,a;\boldsymbol{\theta}) & \text{caso contrário} \end{cases}$
        Executa a_t, observa r_t e estado s_{t+1}
        $s_{t+1} \leftarrow s_t$
        $\phi_{t+1} \leftarrow \phi(s_{t+1})$
        Armazena transição $(\phi_t,a_t,\omega_t,\phi_{t+1})$ em D
        Amostra um lote aleatório de transições $(\phi_j,a_j,\omega_j,\phi_{j+1})$ de D
        $y_j \leftarrow \begin{cases} r_j & \text{se } \phi_{j+1} \text{ for terminal} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1},a';\boldsymbol{\theta}^{-}) & \text{caso contrário} \end{cases}$
        $\mathcal{L}(\boldsymbol{\theta}) = (y_j - Q(\phi_t, a_t; \boldsymbol{\theta}))^2$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}}(\mathcal{L})$
        $\boldsymbol{\theta}^{-} \leftarrow \boldsymbol{\theta}$ a cada κ passos *// atualiza parâmetros da rede alvo*
    fim
fim

mostrou-se crucial para o treinamento bem-sucedido da DQN, impedindo que a alta correlação dos dados de entrada atrapalhasse na atualização dos parâmetros da rede. Como consequência, é possível perceber que a DQN não aprende diretamente de transições geradas pela política atual, representada pelos pesos da rede neural, tornando-a um método de aprendizado *off-policy* (MNIH et al., 2015).

Uma segunda modificação realizada na técnica foi o uso de uma segunda rede, denominada “rede alvo”, a qual mantém uma cópia de $\boldsymbol{\theta}$, denominada $\boldsymbol{\theta}^{-}$. A rede alvo é utilizada na geração das transições a serem armazenadas na memória de repetição, sendo que $\boldsymbol{\theta}^{-} \leftarrow \boldsymbol{\theta}$ somente em intervalos de tempo pré-determinados. O uso da rede alvo permitiu estabilizar o treinamento da rede, visto que as transições não eram geradas por uma rede cujos parâmetros estavam em constante mudança (MNIH et al., 2015).

Devido ao desejo de se utilizar os mesmos hiperparâmetros para se aprender a jogar 49 jogos diferentes, as recompensas retornadas pelo ambiente, que podem possuir escalas diferentes (*e.g.* valores iguais a 0 ou 1 para Pong, mas de 15 ou até 200 no caso de Ms. Pac-Man) eram normalizadas para o intervalo $[-1; 1]$ (MNIH et al., 2015).

Utilizando este *setup*, a DQN obteve pontuações maiores ou iguais a 75% de um jogador experiente em 29 dos 49 jogos testados. O desempenho do algoritmo era evidente em jogos reativos com recompensas imediatas, como *Pong* e *Breakout*, porém se mostrava inferior em jogos que necessitavam de planejamento a longo prazo, como *Montezuma's Revenge*. Adicionalmente, a restrição de r (originalmente tomando magnitudes diferentes para cada ambiente do Atari) ao intervalo $[-1; 1]$ resultou em perda de informação para o agente, fazendo-o considerar toda $r \leq -1$ como equivalente, assim como toda $r \geq 1$. Os autores teorizaram que, havendo uma forma de permitir que o agente possua acesso a recompensas de diferentes magnitudes, este pode aprender a priorizar objetivos ou ações que resultem em recompensas maiores, não se limitando à binarização presença/ausência de recompensa (MNIH et al., 2015).

A tabela 3 apresenta a topologia da rede neural utilizada no trabalho. Três camadas de convolução são utilizadas para aprender representações visuais dos estados dos jogos, e duas camadas totalmente conectadas transformam os filtros em valores Q para todas as possíveis ações. A primeira camada totalmente conectada possui 512 neurônios, enquanto a camada totalmente conectada de saída possui número de neurônios igual à quantidade de ações disponíveis por jogo, (variando de 4 a 18 ações).

Dada a seguinte equação de Bellman para atualização iterativa dos valores Q em busca de Q_* (SUTTON; BARTO, 2018)

$$Q_*(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q(s',a') | s, a\right], \tag{4}$$

o valor-alvo a ser alcançado pela rede neural substitui $Q(s',a')$ por $Q(s',a';\boldsymbol{\theta}^-)$, a aproximação realizada pela rede alvo (MNIH et al., 2015),

$$Y \equiv r_{t+1} + \gamma \max_{a} Q(s_{t+1},a;\boldsymbol{\theta}^-). \tag{5}$$

A função de erro $\mathcal{L}$ a ser minimizada se torna o erro entre o alvo Y e o valor atual de saída da rede (MNIH et al., 2015),

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}\left[(Y - Q(s_t, a_t; \boldsymbol{\theta}))^2\right]. \tag{6}$$

O valor $Y - Q(s, a; \boldsymbol{\theta})$ na equação 6 é o erro de diferenças temporais da DQN, comumente representado na literatura por δ.

Schaul et al. (2016) modificaram a técnica de repetição de experiências (LIN, 1992) de forma a priorizar transições que possuam maior δ, *i.e.* transições que realizem atualizações mais significativas nos valores Q. A priorização foi feita de duas formas diferentes, uma

Tabela 3 – Arquitetura da DQN apresentada em Mnih et al. (2015)

Camada	Entrada	Filtro	Nº de filtros	Stride	Ativação
Convolucional	$84 \times 84 \times 4$	8×8	32	4	ReLU
Convolucional	$20 \times 20 \times 32$	4×4	64	2	ReLU
Convolucional	$9 \times 9 \times 64$	3×3	64	1	ReLU
MLP	$7 \times 7 \times 64$	—	512	—	ReLU
MLP	512	—	4 a 18	—	—

Fonte: Autor

atribuindo probabilidade de repetição às transições proporcionais ao erro TD destas e outra classificando as transições de acordo com seus erros TD e dando probabilidade de escolha de repetição proporcional à posição da transição na classificação. Ambas as estratégias se mostraram superiores à estratégia de repetição uniforme utilizada na DQN original, melhorando o desempenho da rede em 41 de 49 jogos de Atari. A técnica foi nomeada Repetição de Experiência Priorizada (PER, *Prioritized Experience Replay*).

Para mitigar o efeito de observações parciais em POMDPs e permitir que agentes armazenem informação referente a seus históricos de observações, Hausknecht e Stone (2015) apresentam as Redes Q Profundas Recorrentes (DRQNs, *Deep Recurrent Q-Network*), DQNs que utilizam uma camada Memória a curto prazo longa (LSTM, *Long short-term memory*) após as camadas convolucionais, mas antes da camada densa que aproxima Q_*. As LSTM (HOCHREITER; SCHMIDHUBER, 1997) são redes recorrentes que armazenam informação relacionada a uma sequência de dados em um vetor de memória e é utilizada comumente no tratamento de séries temporais e outros dados sequenciais.

van Hasselt et al. (2016) apontam que, devido a uma tendência do *Q-Learning* em estimar o valor do próximo estado a ser visitado, o que é feito através de $\max_a Q_t(s_{t+1}, a)$, o algoritmo pode superestimar os valores de estados futuros (VAN HASSELT, 2010). Eles então adaptam o método apresentado por van Hasselt (2010), denominado *Q-Learning* Duplo, para o âmbito das DQN, criando a Rede Q Profunda Dupla (DDQN, *Double Deep Q-Network*). Enquanto o *Q-Learning* Duplo aproxima duas tabelas Q^A e Q^B simultaneamente em um mesmo domínio, utilizando amostragens dos valores de uma função para realizar as atualizações da outra, na DDQN, a função utilizada como alvo a ser estimado pela rede neural é dada por

$$Y_t^{DDQN} \equiv r_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t^-), \tag{7}$$

onde $\arg\max_a Q(s_{t+1},a;\boldsymbol{\theta}_t)$ é responsável por selecionar a ação a ser utilizada na atualização dos parâmetros da rede, porém a ação selecionada é avaliada usando os parâmetros $\boldsymbol{\theta}_t^-$ da rede alvo (conforme descrito na página 27).

Aplicando a DDQN no domínio dos jogos de Atari 2600, os autores demonstram o aprendizado da função Q com menos oscilações, assim como desempenho mediano maior em 49 jogos de Atari 2600, quando comparada à DQN. Foi apontado que aumentar o intervalo em que $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$ melhora o desempenho da DDQN, uma vez que, quando $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$, as redes de escolha e de avaliação da ação são idênticas, revertendo ao comportamento da DQN. Não foi reportado desempenho melhor da DDQN em jogos nos quais a DQN já demonstrava baixo desempenho.

Ziyu Wang et al. (2016) apresentam a DDQN duelista, uma rede que aproxima tanto a função $V(s)$, a qual deve aprender um valor para cada estado, como a função vantagem $\mathcal{A}(s,a)$, que aprende um valor para cada par estado-ação. Lembrando que a função Q representa o valor ou a preferência por cada ação em cada estado, V representa a preferência por cada estado e $\mathcal{A}(s,a) = Q(s,a) - V(s)$ representa a importância de cada ação em um determinado estado (WANG, Z. et al., 2016).

A DQN duelista é uma única rede que utiliza um mesmo conjunto de camadas convolucionais para extrair características visuais dos jogos de Atari 2600, possuindo caminhos divergentes para as estimativas de V e $\mathcal{A}$, os quais são posteriormente combinados para gerar os valores Q, os quais são a saída da rede.

Os autores relatam especial melhora de desempenho em domínios que possuem grande quantidade de ações, visto que a DQN duelista aproxima apenas um único valor de V para cada estado s, o que se mostra vantajoso caso várias ações em s possuam valores parecidos.

2.2 APRENDIZADO POR REFORÇO MULTI-AGENTES

Os MDPs são estendidos para sistemas multi-agentes de diversas formas. Nos Jogos Estocásticos (SGs, *Stochastic Game*) (SHAPLEY, 1953), múltiplos agentes observam o estado completo, selecionam ações individualmente e recebem sinais de recompensa distintos, devendo aprender a maximizar recompensas individuais. Quando os agentes possuem acesso a observações ao invés de estados, tem-se um Jogo Estocástico Parcialmente Observável (POSG, *Partially Observable Stochastic Game*) (WIGGERS; OLIEHOEK; ROIJERS, 2015).

Um sistema multi-agentes totalmente observável e totalmente cooperativo pode ser formalizado como um Processo de Decisão de Markov Multi-Agentes (MMDP, *Multi-Agent*

Markov Decision Process) (BOUTILIER, 1999), enquanto um sistema multi-agentes totalmente cooperativo e parcialmente observável, porém no qual as observações dos agentes formam o estado completo pode ser considerado um Processo de Decisão de Markov Descentralizado (Dec–MDP, *Decentralized Markov Decision Process*) (AMATO, 2015).

Uma última classe de problemas, de especial interesse para este trabalho, é a de sistemas multi-agentes totalmente cooperativos e parcialmente observáveis, nos quais a observação conjunta dos agentes não necessariamente reconstrói o estado original. Estes problemas foram descritos de maneira equivalente por Bernstein et al. (2002) como Processos de Decisão de Markov Parcialmente Observáveis Descentralizados (Dec–POMDPs, *Decentralized Partially Observable Markov Decision Process*) e por Pynadath e Tambe (2002) como Processos de Decisão de Times Multi-Agentes (MTDPs, *Multi-Agent Team Decision Process*).

Este trabalho utiliza a definição de um Dec–POMDP como dada por Amato (2015):

Definição 3 – Dec–POMDP. Um Dec–POMDP é uma tupla $\mathcal{M} = (U, S, A, O, P, R, H)$, onde:

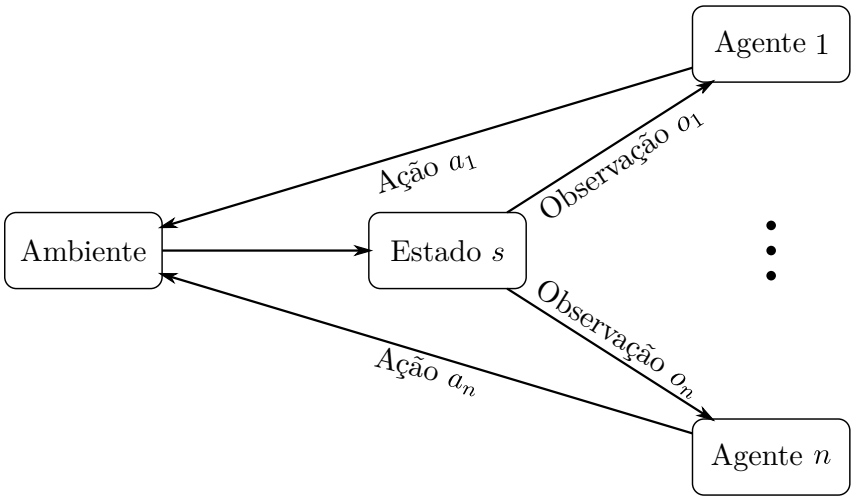
a) U: conjunto finito de agentes;
b) S: conjunto finito de estados;
c) A_u: conjunto finito de ações para o agente u;
d) O_u: conjunto finito de observações para o agente u;
e) P: função de probabilidade de transição $P(s'|s, \boldsymbol{a})$ mapeando as ações conjuntas de todos os agentes ($\boldsymbol{a}$) no estado s à probabilidade de transição do ambiente para o estado s';
f) R: uma função de recompensa compartilhada $R(s,\boldsymbol{a})$;
g) H: um modelo de observações $H(\boldsymbol{o}|s',\boldsymbol{a})$, o qual dita as probabilidades dos agentes observarem $\boldsymbol{o}$ em s' após executarem $\boldsymbol{a}$.

A figura 2 apresenta graficamente um Dec–POMDP, no qual o ambiente provê um estado s, os agentes derivam suas observações $\boldsymbol{o}$ de s e selecionam suas ações $\boldsymbol{a}$, as quais são aplicadas conjuntamente.

2.2.1 Agentes homogêneos e heterogêneos

Ao se trabalhar com sistemas multi-agentes, não existe a expectativa de que todos os agentes sejam semelhantes. A agentes que são implementados utilizando equipamentos físicos distintos ou demonstrando comportamentos diferentes, Vlassis (2007) dá o nome de

Figura 2 – Representação da interação entre agente e ambiente em um Dec–POMDP



Fonte: Autor “adaptado de” Amato, 2015

heterogêneos, enquanto agentes que possuem capacidades e comportamento semelhantes são chamados de homogêneos. Terry et al. (2020) descrevem agentes homogêneos como aqueles que possuem espaços de observações, ações e função de recompensa idênticos e cuja função de transição é simétrica caso hajam permutações entre as ações ou, informalmente, agentes cujas políticas podem ser intercambiadas entre si sem alterar o resultado. Todos os outros agentes são considerados heterogêneos.

Definição 4 – Agentes homogêneos e heterogêneos. Dois agentes u_1, u_2 são considerados homogêneos se $A_{u_1} = A_{u_2}, O_{u_1} = O_{u_2}, R_{u_1} = R_{u_2}$ e $P(s'|s, \{a_{u_1}, a_{u_2}\}) = P(s'|s, \{a_{u_2}, a_{u_1}\})$. Caso contrário, são heterogêneos.

A homogeneidade entre agentes é conveniente, visto que agentes homogêneos podem compartilhar artefatos aprendidos entre si, como a política ou funções-valor, assim como compartilhar experiências que permitam que esses artefatos sejam aprendidos.

2.2.2 Comunicação entre agentes

Ao lidar com sistemas multi-agentes cooperativos parcialmente observáveis, a existência de algum mecanismo de comunicação entre os agentes permite o compartilhamento de informações referentes a suas observações parciais individuais, diminuindo a incerteza presente no estado de crença de cada agente. Em alguns casos, a comunicação entre agentes pode ser considerada uma

classe distinta de ações (TAN, 1993; PYNADATH; TAMBE, 2002; FOERSTER, Jakob et al., 2016), incorrendo um custo aos agentes que enviam ou requisitam mensagens.

Há evidências de que a comunicação entre agentes só é benéfica se as informações transmitidas forem relevantes. Em especial Tan (1993) aponta, em um experimento empírico que, ao aumentar o campo de visão de um agente usando o sensoriamento de outro agente (algo análogo ao que é feito na seção de experimentos deste trabalho), o aprendizado do agente foi acelerado. Porém, essa expansão do campo de visão do primeiro agente exibiu efeitos adversos caso o campo de visão do segundo agente fosse pouco informativo, aumentando o espaço de estados do primeiro agente principal sem adicionar informações relevantes para a tomada de decisão.

Balch e Arkin (1994) apontam que a comunicação é mais eficaz em tarefas nas quais o agente não pode adquirir as mesmas informações de maneira implícita (*i.e.* através das próprias observações) e que, nas tarefas em que um agente se beneficia da recepção de mensagens de outros agentes, mensagens com conteúdo simples podem ser tão eficazes quanto mensagens complexas.

Ao se utilizar uma rede neural capaz de aprender as transformações a serem aplicadas nas observações dos agentes que geram as mensagens a serem transmitidas, pode-se dizer que a relevância das informações contidas nas mensagens é guiada pelo próprio treinamento da rede, influenciada pela função de erro a ser minimizada. Esta observação pode fazer do uso de uma técnica de aprendizado de máquina uma alternativa viável, comparada à seleção manual do conteúdo das mensagens.

Em sistemas multi-agentes totalmente cooperativos, o uso de comunicação pode auxiliar na emergência de comportamento coordenado, beneficiando os agentes no alcance de um objetivo em comum (MATIGNON; LAURENT; LE FORT-PIAT, 2012).

2.2.3 Aprendizado por Reforço Profundo Multi-Agentes

Praticamente todo o progresso na área que hoje é conhecida como MADRL ocorreu nos últimos cinco anos. Uma revisão feita por Hernandez-Leal, Kartal e Taylor (2019) aponta quatro grandes áreas nas quais os trabalhos em MADRL se enquadram:

Na categoria de **análise de comportamento emergente**, o objetivo é analisar as interações entre agentes independentes controlados por redes neurais profundas, interagindo em um mesmo ambiente. Exemplos de trabalhos nesta categoria modelam agentes em cenários competitivos

como redes neurais profundas distintas (TAMPUU et al., 2017), ou realizam o treinamento de múltiplos agentes utilizando uma única rede neural, generalizando observações do ambiente de forma que sejam intercambiáveis entre agentes homogêneos (EGOROV, 2016).

Em **aprendizado de comunicação**, agentes aprendem protocolos de comunicação para resolver tarefas cooperativas. Estratégias incluem a criação de mecanismos de passagem de mensagem entre agentes aprendidos através do *backpropagation* (SUKHBAATAR; SZLAM; FERGUS, 2016; PENG et al., 2017); aprendizado de linguagens compostas de símbolos discretos (MORDATCH; ABBEEL, 2018); e compartilhamento de informação entre agentes em memória compartilhada (PESCE; MONTANA, 2020). Ademais, o ato de se comunicar pode ser considerado uma ação separada (FOERSTER, Jakob et al., 2016) cujo valor é aprendido como parte da política, ou uma ação gratuita.

No **aprendizado de cooperação**, grupos de agentes aprendem a cooperar, tanto em cenários totalmente cooperativos ou mistos, sem o compartilhamento explícito de mensagens entre si. Uma estratégia direta se baseia no uso de métodos voltados para sistemas de um único agente no aprendizado de políticas ou valores-ação de múltiplos agentes, garantindo que cada agente acesse suas próprias políticas simplesmente através das observações distintas ou do uso de um identificador de agente como entrada do modelo (GUPTA; EGOROV; KOCHENDERFER, 2017).

Outra classe de trabalhos ainda categorizados como aprendizado de cooperação inclui as técnicas que assumem que as funções valor-ação individuais Q_u de cada agente u em um cenário cooperativo podem ser interpretadas como resultado da fatoração (KOLLER; PARR, 1999; GUESTRIN et al., 2003a) ou decomposição de uma função valor-ação hipotética Q_U para o time de agentes. Nessa classe de trabalhos, denominada de *mixing*, uma rede neural centralizada aproxima Q_U enquanto disponibiliza Q_u para todo agente u interagir com o ambiente.

O primeiro exemplo de *mixing* foi a Rede de Decomposição de Valor (VDN, *Value Decomposition Network*) (SUNEHAG et al., 2018), na qual os autores fazem a suposição de que as funções valor-ação do time de agentes são a decomposição de uma função valor-ação aditiva para todos os agentes,

$$Q_U(\boldsymbol{o}, \boldsymbol{a}) = \sum_{u \in U} Q_u(o_u, a_u). \tag{8}$$

Já no QMIX (RASHID et al., 2018, 2020a), a suposição de decomposição aditiva é relaxada para uma de monotonicidade entre as funções individuais de cada agente, permitindo que o QMIX aproxime uma família maior de funções valor-ação conjuntas. Monotonicidade, neste contexto, refere-se à relação entre Q_U e cada Q_u apresentada na inequação 9, indicando

que mudanças positivas na função valor-ação de um agente repercutem em nenhuma mudança, ou uma mudança positiva, na função valor-ação conjunta.

$$\frac{\partial Q_U(\boldsymbol{o}, \boldsymbol{a})}{\partial Q_u(o_u, a_u)} \geq 0, \quad \forall u \in U. \tag{9}$$

Tanto a suposição aditiva como monotônica preservam a ordem dos valores de cada ação tanto nas funções valor-ação individuais como na conjunta, garantindo que as ações selecionadas pela política ótima gulosa e determinística aproximada por Q_U são as mesmas que seriam selecionadas seguindo-se as funções Q_u:

$$\arg\max_{\boldsymbol{a}} Q_U(\boldsymbol{o}, \boldsymbol{a}) = \begin{pmatrix} \arg\max_{u_1} Q_1(o_1, a_1) \\ \vdots \\ \arg\max_{u_n} Q_n(o_n, a_n) \end{pmatrix}. \tag{10}$$

Diversas extensões do QMIX foram propostas, na tentativa de relaxar ainda mais a restrição de monotonicidade das funções valor-ação dos agentes. No entanto, trabalhos recentes (HU, S. et al., 2021) demonstram que, empiricamente, todos os métodos de *mixing* possuem desempenho comparável quando seus hiperparâmetros são selecionados de maneira informada e truques de implementação são considerados. Ademais, para aplicações no ambiente estudo neste trabalho, um método de *mixing* restritivo como a VDN é considerado suficiente (WHITESON, 2020).

Por último, na categoria **agentes modelando agentes**, um agente incorpora a modelagem do comportamento de outros agentes em seu aprendizado, tanto em sistemas cooperativos (LOWE et al., 2017) como competitivos (HE et al., 2016; RABINOWITZ et al., 2018).

Na literatura, é possível perceber a prevalência de algoritmos de MADRL baseados em função-valor, em detrimento de métodos baseados em gradiente de políticas. Alguns autores evidenciam o desempenho superior da primeira categoria de métodos em comparação com a segunda (LOWE et al., 2017; RASHID et al., 2020a), desempenho esse atribuído à instabilidade do treinamento de métodos de gradiente de política em cenários multi-agentes.

2.3 GRAFOS

Um grafo é uma estrutura de dados abstrata capaz de representar conceitos e suas relações. Devido ao extenso uso e menção de diferentes tipos de grafos ao longo do texto, as seguintes definições podem ser de interesse do leitor.

Definição 5 – Grafo. Um grafo é um par ordenado $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, no qual $\mathcal{V}$ é um conjunto não-vazio de vértices e $\mathcal{E}$ é um conjunto de arestas. Cada aresta $e \in \mathcal{E}$ conecta um par de vértices não necessariamente distintos (adaptada de Bondy e Murty, 2008).

Definição 6 – Grafo direcionado. Um grafo direcionado ou orientado é um par ordenado $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, no qual $\mathcal{V}$ é um conjunto não-vazio de vértices e $\mathcal{E}$ é um conjunto de arcos. Cada arco $e \in \mathcal{E}$ associa um par ordenado de vértices (não necessariamente distintos) (v_1, v_2) (adaptada de Bondy e Murty, 2008).

Definição 7 – Grafo rotulado. Um grafo é rotulado se seus vértices ou arestas possuem atribuições numéricas, ou categóricas (adaptada de Goldbarg e Goldbarg, 2012).

2.4 REDES NEURAIS DE GRAFOS

É comum que novas classes de redes neurais sejam propostas, incorporando novas operações em suas arquiteturas de forma a se utilizar de características inerentes aos dados sobre os quais vão operar. A essa personalização do funcionamento de um modelo ou algoritmo, seja ela aplicada à escolha de valores de hiperparâmetros ou na alteração das próprias operações que compõem um modelo, dá-se o nome de viés (MITCHELL, 1980) ou viés indutivo (BATTAGLIA, P. W. et al., 2018).

Um exemplo de viés está presente nas redes convolucionais (LECUN, Y. et al., 1989; LECUN, Yann et al., 1998), as quais se aproveitam da informação espacial presente na vizinhança de cada píxel, utilizando filtros de convolução com pesos treináveis que operam sobre regiões fisicamente próximas na imagem.

Já as redes recorrentes (HOPFIELD, 1982; ELMAN, 1990; JORDAN, 1997; HOCHREITER; SCHMIDHUBER, 1997; CHO et al., 2014) se aproveitam da natureza sequencial dos dados sobre os quais operam (*e.g.* palavras em uma sentença) e empregam vetores de memória para armazenar informações relacionadas a unidades de informação processadas anteriormente.

Seguindo este paradigma, uma classe de redes neurais foi proposta, a qual opera sobre dados estruturados na forma de grafos, estruturas de dados que capturam relações entre entidades discretas (GORI; MONFARDINI; SCARSELLI, 2005; SCARSELLI et al., 2009a,b).

No contexto de aprendizado de máquina, um grafo pode armazenar vetores de dados em cada um de seus vértices e arestas, além de possuir um vetor de dados global, representativo do próprio grafo (BATTAGLIA, P. W. et al., 2018). Exemplos de dados neste formato incluem: artigos acadêmicos e as citações realizadas entre si (KIPF; WELLING, 2017), grafos

moleculares (DUVENAUD et al., 2015) ou objetos e suas interações físicas (BATTAGLIA, P. et al., 2016). Os grafos que carregam vetores de características costumam ser nomeados grafos atribuídos (ZHANG, Y. et al., 2019b; ZHANG, C. et al., 2019; ZHANG, Y. et al., 2019a; CEN et al., 2019).

Definição 8 – Grafo atribuído (CEN et al., 2019). Um grafo $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ é atribuído quando possui uma matriz de características $\boldsymbol{X}$, de tal forma que $\forall v \in \mathcal{V}_{\mathcal{G}}, \exists \vec{x}_v \in \boldsymbol{X}$, onde $\vec{x}_v$ é o vetor de características do vértice v.

Os modelos de redes neurais que operam sobre grafos são denominados GNNs (GORI; MONFARDINI; SCARSELLI, 2005; SCARSELLI et al., 2009a,b). As GNNs são modelos versáteis, podendo, por exemplo, ser definidas como funções que mapeiam grafos a vetores, $f(\mathcal{G}) \rightarrow \mathbb{R}^d$, sendo treinadas em bases de dados formadas de tuplas $(\mathcal{G}, y)$, onde y é a saída esperada para a função $f(\mathcal{G})$. Uma GNN também pode ser considerada uma função que mapeia vértices individuais de $\mathcal{G}$ a saídas, $f(\mathcal{G}, v) \rightarrow \mathbb{R}^d$, sendo treinada, neste caso, com tuplas $(\mathcal{G}, v, y)$, na qual v é um vértice de $\mathcal{G}$ e y, a saída esperada para $f(\mathcal{G}, v)$.

De forma geral, uma GNN realiza a propagação dos vetores de informação de vértices ao longo de suas vizinhanças de saída. Cada vértice recebe os vetores dos vértices de sua vizinhança de entrada, agrega-os em um único vetor e aplica uma função de atualização ao vetor agregado. Este mecanismo de propagação, agregação e atualização, o qual é descrito de diversas formas na literatura, foi unificado em Gilmer et al. (2017) sob o nome de Redes Neurais de Passagem de Mensagens (MPNNs, *Message-Passing Neural Network*) e é apresentado a seguir.

Na **propagação**, um vértice v observa os vetores de características de toda a sua vizinhança de entrada e os transforma através da aplicação de uma função M, a qual transforma os vetores de características de cada vértice da vizinhança de entrada individualmente e independentemente dos vetores de outros vértices na mesma vizinhança.

A transformação realizada por M no vetor de características $\vec{x}_j^{(l-1)}$ de um vértice j da vizinhança de entrada de v também pode ser realizada em função do próprio vetor de características de v, $\vec{x}_v^{(l-1)}$, além de um vetor de características armazenado na aresta que une v a j, representado por $\vec{e}_{(j,v)}$.

No contexto das GNN, a função M é atrelada a uma camada l da rede neural, transformando vetores de características provenientes da camada $l-1$, além de ser parametrizada por pesos ajustáveis durante o treinamento da rede. Ela é representada na equação 11.

$$M^{(l)}\left(\vec{x}_v^{(l-1)}, \vec{x}_j^{(l-1)}, \vec{e}_{(j,v)}\right) \tag{11}$$

Já no passo de **agregação**, que sucede a agregação, os valores resultantes da aplicação de M a toda a vizinhança de entrada de v são agregados utilizando a função de agregação Agg, a qual deve ser equivariante a permutações dos vértices. Exemplos de funções que podem ser utilizados no lugar de Agg são a somatória, média e máximo.

A equação 12 une os passos de propagação e agregação, os quais geram um vetor intermediário $\vec{m}$ para cada vértice no grafo. Na equação, $\mathcal{N}^-{}_v$ denota a vizinhança de entrada de um vértice v.

$$\vec{m}_v = Agg_{j \in \mathcal{N}^-{}_v}\left(M^{(l)}\left(\vec{x}_v^{(l-1)}, \vec{x}_j^{(l-1)}, \vec{e}_{(j,v)}\right)\right) \tag{12}$$

Por fim, na fase de atualização, o vetor de características de v no instante l é dado por

$$\vec{x}_v^{(l)} = U^{(l)}\left(\vec{x}_v^{(l-1)}, \vec{m}_v\right), \tag{13}$$

onde u é chamada de função de atualização.

É importante notar que, para fins de generalização, o índice l representa um instante de atualização de todos os vetores dos vértices em $\mathcal{V}(\mathcal{G})$. Na prática, l pode ser interpretada como o índice de uma camada numa rede neural, sendo que cada camada possui um par de funções $M^{(l)}$ e $U^{(l)}$ distinto, ambas parametrizadas por pesos que são aprendidos durante o treinamento.

2.4.1 Equivariância e invariância a permutações

Duas propriedades de interesse que uma GNN deve possuir são a equivariância e a invariância a permutações. Dada uma matriz qualquer $\boldsymbol{X}$ e uma matriz de permutação $\boldsymbol{P}$, a transformação realizada em $\boldsymbol{X}$ por $\boldsymbol{P}$ permuta as linhas de $\boldsymbol{X}$, como exemplificado na equação 14 (VELIČKOVIĆ, 2021).

$$\boldsymbol{P}_{(2;4;1;3)}\boldsymbol{X} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} — & \vec{x}_1 & — \\ — & \vec{x}_2 & — \\ — & \vec{x}_3 & — \\ — & \vec{x}_4 & — \end{bmatrix} = \begin{bmatrix} — & \vec{x}_2 & — \\ — & \vec{x}_1 & — \\ — & \vec{x}_4 & — \\ — & \vec{x}_3 & — \end{bmatrix} \tag{14}$$

Suponha que exista, nas equações 11 e 12, uma versão da função de propagação M aplicada à matriz de vetores da vizinhança de entrada de um vértice v, matriz esta que pode ser denotada por $\boldsymbol{X}_{\mathcal{N}^-{}_v}$. Cada linha de $\boldsymbol{X}_{\mathcal{N}^-{}_v}$ representa um vetor de características de um vértice de $\mathcal{N}^-{}_v$. É desejável que a transformação que M realiza em cada linha de $\boldsymbol{X}_{\mathcal{N}^-{}_v}$

seja independente das transformações realizadas nas outras linhas. Este é um exemplo de *equivariância a permutações*.

Definição 9 – Equivariância a permutações (HAMILTON, 2020). Uma função f é considerada equivariante a permutações quando, ao ser aplicada a uma matriz $\boldsymbol{X}$, ela alterar cada linha de $\boldsymbol{X}$ individualmente. Esse é o equivalente da equação abaixo, na qual $\boldsymbol{P}$ é uma matriz de permutações.

$$f(\boldsymbol{P}\boldsymbol{X}\boldsymbol{P}^{\top}) = \boldsymbol{P}f(\boldsymbol{X}) \tag{15}$$

De forma semelhante, suponha que as vizinhanças de entrada de dois vértices v e j sejam compostas pelo mesmo conjunto de vértices, isto é, $\mathcal{N}^{-}{}_{v} = \mathcal{N}^{-}{}_{j}$. Uma vez que $\mathcal{N}^{-}{}_{v}$ e $\mathcal{N}^{-}{}_{j}$ são conjuntos, seus elementos não são ordenados. Independente disso, é desejável que a função Agg, apresentada na equação 12, possua a mesma saída para ambos os conjuntos de vértices, i.e. $\vec{m}_v = \vec{m}_j$. À propriedade que garante que o resultado de uma função independa da ordem de seus parâmetros de entrada dá-se o nome de *invariância a permutações*.

Definição 10 – Invariância a permutações (HAMILTON, 2020). Uma função f é considerada invariante a permutações quando, ao ser aplicada a uma matriz $\boldsymbol{X}$, ela retornar o mesmo resultado independente da ordem das linhas de $\boldsymbol{X}$. Esse é o equivalente da equação abaixo, na qual $\boldsymbol{P}$ é uma matriz de permutações.

$$f(\boldsymbol{P}\boldsymbol{X}\boldsymbol{P}^{\top}) = f(\boldsymbol{X}) \tag{16}$$

As propriedades de equivariância e invariância a permutações permitem que uma rede neural que opera sobre grafos tenha o mesmo resultado em dois grafos isomorfos (grafos com a mesma topologia).

2.4.2 Aprendizado de máquina em grafos heterogêneos

Algumas espécies de dados podem ser melhor representadas através de grafos caso estes possam especializar vértices ou arestas em tipos específicos. A este tipo específico de grafo costuma-se dar o nome na literatura de grafos heterogêneos.

Definição 11 – Grafo heterogêneo (SUN et al., 2011; CEN et al., 2019). Dados um grafo $\mathcal{G}$, um conjunto de tipos de vértices C e um conjunto de tipos de arestas Ω, $\forall v \in \mathcal{V}_{\mathcal{G}}, \exists c \in C | C(v) = c$

e $\forall e \in \mathcal{E}_{\mathcal{G}}, \exists \omega \in \Omega | \Omega(e) = \omega$. Um grafo $\mathcal{G}$ é considerado heterogêneo quando $|C| + |\Omega| > 2$. Caso contrário, $\mathcal{G}$ é homogêneo.

Exemplos de bases de dados definidas como grafos individuais ou múltiplos grafos heterogêneos incluem:

a) redes de relações semânticas entre palavras (sinônimos, hiperônimos, hipônimos, merônimos e holônimos): WordNet, (FELLBAUM, 2010) e FB15k-237, Toutanova e Chen, 2015;
b) relações espaciais e de interações entre objetos em imagens: Visual Genome, Krishna et al., 2017;
c) redes de trabalhos acadêmicos, autores, orientadores e citações: OAG, Fanjin Zhang et al., 2019, DBLP[1] e Aminer, (TANG, 2016);
d) estruturas de proteínas e compostos químicos: PROTEINS (BORGWARDT et al., 2005), e MUTAG (DEBNATH et al., 1991);
e) bases de vendedores, produtos, clientes e revisões de produtos: Yelp[2];
f) bancos e de filmes, seus gêneros, atores, diretores e locais de filmagem: IMDB[3] e Douban (MA et al., 2011).

Os problemas a serem solucionados nesses tipos de bases de dados são similares àqueles presentes em grafos homogêneos, com a peculiaridade da existência de informações adicionais embutidas nos dados. Esses problemas incluem classificação, agrupamento e regressão de caraterísticas, tanto de nós como de grafos, e predição de conexões entre vértices.

Uma revisão de trabalhos que apresentam métodos de aprendizado supervisionado e semi-supervisionado em redes neurais de grafos para grafos heterogêneos é apresentada na seção 4.3.

2.5 RESUMO

A seção 2 apresentou os conceitos fundamentais por trás dos métodos apresentados nesta tese, os quais aplicam técnicas de aprendizado de máquina voltadas para grafos para aprender políticas e protocolos de comunicação em tarefas totalmente cooperativas envolvendo agentes heterogêneos. Foram apresentados os fundamentos de aprendizado por reforço com um único

[1] `https://dblp.org/`
[2] `https://www.yelp.com/dataset`
[3] `https://www.imdb.com/interfaces/`

e múltiplos agentes, os recentes avanços nas áreas conhecidas como Aprendizado por Reforço Profundo e Aprendizado por Reforço Profundo Multi-Agentes, e as redes neurais de grafos como mecanismos de passagem de mensagem entre vértices de grafos tanto homogêneos como heterogêneos.

3 DOMÍNIO DE ESTUDO

Neste trabalho, os métodos propostos foram testados no *StarCraft Multi-Agent Challenge* (SMAC), um ambiente multi-agentes baseado em um jogo de computador chamado *StarCraft II*, o qual, por sua vez, faz parte de um gênero específico de jogos denominado Estratégia em Tempo Real (RTS, *Real-Time Strategy*). Esta seção apresenta: o lugar dos jogos de RTS no mundo acadêmico, as características que definem o *StarCraft II*, e o SMAC como um Dec–POMDP.

Os jogos de RTS são jogos eletrônicos nos quais dois ou mais jogadores controlam bases e múltiplas unidades em um cenário de combate. Apesar de existir variação na jogabilidade dentre os jogos do gênero, o objetivo de um jogador em uma partida é destruir todas as bases e unidades dos jogadores adversários. Aspectos secundários, porém igualmente importantes em assegurar a vitória em um jogo de RTS incluem o correto gerenciamento de bases, assim como a coleta e gerenciamento de recursos disponíveis no mapa, representados em diferentes jogos como minério, gás, ouro, madeira ou pedra, os quais permitem a expansão das próprias forças.

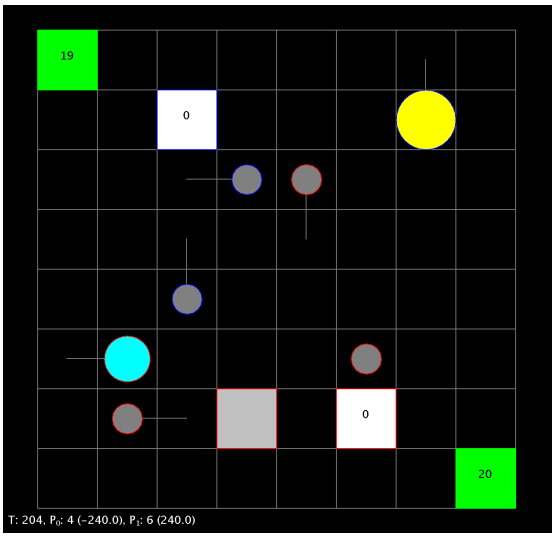
Diversos jogos de RTS foram criados ou instrumentados com o intuito de estudar as peculiaridades que o gênero de jogo apresenta ou se aproveitar dessas peculiaridades para avançar o estado da arte em áreas correlatas. Alguns jogos do gênero que marcaram presença na área acadêmica incluem o *Open RTS* (ORTS) (BURO, 2003), *Wargus*[1], μRTS (ONTAÑÓN, 2014), ELF (TIAN et al., 2017) e StarCraft: Brood War (CERTICKY; SARNOVSKY; VARGA, 2018–0008). Em conjunto, todos esses jogos foram utilizados como plataformas de pesquisa, estimulando a pesquisa em algoritmos e aprendizado de máquina (CERTICKY et al., 2018) por quase duas décadas.

Os diferentes jogos de RTS diferem em complexidade. A figura 3 exibe capturas de tela de três RTS distintos, os quais exibem níveis de complexidade visivelmente distintos. A figura 3(a) apresenta o μRTS (ONTAÑÓN, 2014), criado explicitamente para fins de pesquisa e competições universitárias. A figura 3(b) apresenta o *Wargus*, uma versão de código aberto do *WarCraft*, um RTS comercial datado de 1996. Por último, a figura 3(c) apresenta o *StarCraft II*, um RTS da atual geração, com gráficos tridimensionais.

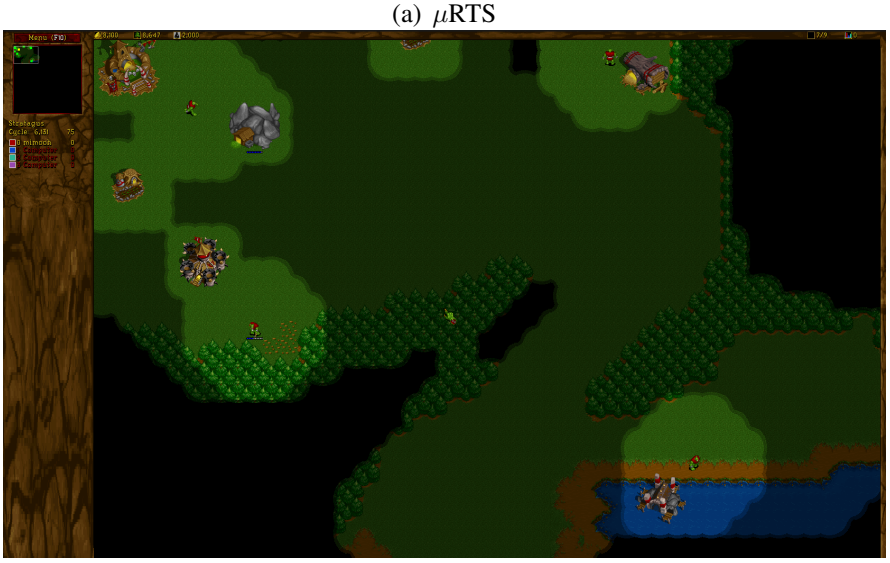
Essa categoria de jogos continua sendo de interesse para a comunidade acadêmica, principalmente pelo desafio que a alta complexidade do domínio apresenta aos algoritmos de DRL (VINYALS et al., 2017; USUNIER et al., 2017; TIAN et al., 2017; FOERSTER, Jakob et al., 2017a,b; NAIR; CHERNOVA, 2018; HU, Y. et al., 2018; FOERSTER, J. N. et al., 2018b;

[1] https://wargus.github.io/

Figura 3 – Capturas de tela de diferentes jogos de RTS



(a) μRTS



(b) *Wargus*

(c) *StarCraft II*

Fontes: 3(a) autor “adaptado de” Ontañón, 2014, 3(b) autor “adaptado de” Wargus, 2018, 3(c) autor

ZAMBALDI et al., 2019; VINYALS et al., 2019; JUSTESEN et al., 2019). Portanto, este trabalho considera os jogos de RTS um domínio válido para os estudos aqui propostos.

3.1 *STARCRAFT II*

StarCraft II é um jogo de RTS desenvolvido pela *Blizzard Entertainment* e lançado em 27 de junho de 2010. Em seu modo competitivo ou multi-jogadores, cada partida é composta de dois ou mais jogadores, tanto humanos como controlados autonomamente pelo jogo. Cada jogador escolhe entre uma de três raças disponíveis, Terran, Zerg ou Protoss e devem explorar um mapa controlando uma ou mais unidades, enquanto administram bases e recursos naturais e atacam outros jogadores com suas unidades. O objetivo do jogo é derrotar todos os outros jogadores, destruindo suas bases e unidades.

As três raças possuem diversas características semelhantes, porém as estratégias para se jogar com cada uma diferem consideravelmente, ao ponto em que jogadores tendem a se especializar. Adicionalmente, existem diversas estratégias criadas e validadas pela comunidade, especializadas em cada uma das raças ou em duelos entre duas raças específicas.

Cada raça possui aproximadamente 20 tipos de unidades diferentes. Cada unidade possui uma quantidade de pontos de vida. Quando uma unidade recebe dano de outra unidade, seus pontos de vida são subtraídos de acordo com o dano do ataque infligido pela unidade atacante. Quando os pontos de vida de uma unidade chegam a zero, ela morre e é removida do jogo. As unidades da raça Protoss possuem uma propriedade chamada “escudo de plasma”, o qual age semelhante aos pontos de vida, mas se recupera ao longo de um período de tempo caso a unidade não esteja sendo atacada.

As unidades diferem entre si em diversas propriedades. Elas podem ser categorizadas como Leves ou Blindadas, Mecânicas ou Biológicas, Terrestres ou Voadoras e de Ataque à Distância ou Corpo-a-Corpo. Unidades muito grandes são categorizadas como Massivas. Unidades de ataque corpo-a-corpo devem se aproximar de unidades adversárias para atacá-las, enquanto unidades de ataque à distância podem fazê-lo sem se aproximar. Unidades voadoras podem voar sobre obstáculos ou vãos, enquanto as terrestres devem se desviar deles. É comum que unidades voadoras não possam ser atacadas por unidades de ataque corpo-a-corpo.

Algumas unidades recebem um bônus de dano quando atacam unidades adversárias com propriedades específicas, *e.g.* ataques com temática elétrica causam mais dano a unidades mecânicas, ataques de unidades massivas afetam múltiplas unidades leves simultaneamente,

ataques explosivos causam mais dano a unidades blindadas e ataques com temática tóxica são mais eficazes em unidades biológicas.

Desde o lançamento do primeiro jogo da franquia, *StarCraft* é jogado profissionalmente, sendo particularmente famoso na Coreia do Sul. Jogadores profissionais competem todos os anos por prêmios de dezenas de milhares de dólares, sendo que o prêmio pela vitória no campeonato mais famoso em 2018 igual a 700 mil dólares para o campeão.

Jogadores também jogam *online* uns contra os outros, sendo que as habilidades de cada jogador podem ser mensuradas através de indicadores como o *Matchmaking Ranking* ou os *ladder points*, ambos utilizados na classificação dos jogadores e escolha de adversários.

Uma partida comum de *StarCraft II* competitivo dura entre 20 minutos e 1 hora, tempo no qual os jogadores devem gerenciar suas bases e recursos e criar novas unidades, compondo suas forças para atacar o jogador adversário. Não existe limite teórico máximo para o número de unidades que um jogador pode ter sob seu controle, sendo comum ter-se centenas de unidades em jogo em momentos avançados de uma partida. Todas as decisões e comandos às unidades são tomados em tempo real, uma vez que o jogo não possui pausas ou turnos.

Na comunidade de *StarCraft*, o número de ações que um jogador faz por minuto é medida através de duas quantidades. A primeira, denominada ações por minuto (APM, *actions per minute*), diz efetivamente quantas ações um jogador executa. Cada clique do mouse, tecla ou combinação de teclas pressionada (por exemplo Ctrl + 1 ou Shift + 1) é considerada uma ação. Ações que possuem um alvo (por exemplo, ordenar a unidade A a atacar a unidade B) contam como duas ações. O APM é a média da soma desses valores durante o jogo. Já a segunda medida, denominada ações efetivas por minuto (EPM, *effective actions per minute*), foi criada como uma medida mais precisa que o APM, ignorando ações redundantes comumente executadas pelos jogadores (por exemplo, pressionar a mesma tecla várias vezes para se enviar um comando).

Jogadores casuais costumam ter APM ≤ 50, enquanto no meio profissional, os jogadores costumam apresentar APM ≥ 200. Uma vez que *StarCraft* é jogado em tempo real, jogadores que possuem valores maiores de APM e EPM podem ter vantagem contra seus adversários, por possuírem maior frequência na passagem de comandos para suas unidades. No caso de agentes autônomos, o APM costuma ser limitado artificialmente, a fim de impedir que um possível desempenho superior da máquina sobre humanos aconteça devido à velocidade com a qual esta consegue enviar comandos ao jogo.

Por último, *replays* dos jogos podem ser salvos em um formato próprio do jogo para serem assistidos e analisados posteriormente, o que prontificou a criação de diversas bases de *replays* disponíveis *online*, tanto de jogos amadores como de campeonatos profissionais.

A figura 3(c) apresenta uma captura de tela de uma partida de *StarCraft II*. Nela, o exército controlado pelo jogador (à esquerda) invade uma base adversária (à direita) utilizando diversas unidades. A captura de tela contém toda a informação visual à disposição de um jogador humano em um instante de tempo. O mini-mapa no canto inferior esquerdo exibe a extensão do mapa onde uma partida acontece, sendo o local focado pela câmera demarcado por trapézio na extremidade inferior direita do mini-mapa. Os retratos visíveis na parte inferior-central da figura indicam as unidades selecionadas atualmente pelo jogador, possibilitando enviar comandos para múltiplas unidades de uma vez.

Ao se jogar contra um adversário controlado pelo computador, *StarCraft II* possui dez opções de níveis de dificuldade, sendo os primeiros sete crescentes no quão agressivamente o computador busca atacar e pressionar os adversários e os três últimos considerados trapaceiros pela comunidade, pelo fato de terem visão completa do mapa, 40% a mais de recursos coletados ou as duas vantagens simultaneamente.

A taxa de atualização visual do jogo é limitada a 60 quadros por segundo. Já a taxa de atualização lógica, na qual qualquer processamento é de fato feito, é realizada 16 vezes por segundo na velocidade normal de jogo e 22,4 vezes na velocidade mais rápida, velocidade esta utilizada nos jogos *online* e competições. Ao utilizar uma API, é possível modificar esta taxa para a velocidade ou taxa desejadas, permitindo tomar mais tempo em possíveis fases de aprendizado ou tomada de decisão.

Com relação às propriedades determinísticas do jogo, fatores aleatórios são introduzidos ao jogo para fins cosméticos e de balanceamento. Por exemplo, no caso de duas unidades executando a mesma ação em simultâneo, uma diferença de -1 a +2 passos lógicos é adicionada ao repetir a ação para que as animações das unidades pareçam menos robóticas e para que, no caso de um embate entre unidades semelhantes, o resultado seja decidido aleatoriamente[2]. No mais, o jogo age de forma completamente determinística.

StarCraft II também é caracterizado por observações parciais. A visibilidade de um jogador é limitada de duas formas. Primeiramente, a escolha relacionada a qual região do mapa a visão deve ser focada pode ser considerada uma ação, passível de mudança pelo próprio agente.

[2] `https://github.com/deepmind/pysc2/blob/master/docs/environment.md#determinism-and-randomness`

Em segundo lugar, a visibilidade do mapa que um jogador possui é limitada à união de todas as regiões observadas por suas unidades. Uma região do mapa que nunca foi visualizada antes pode ser representada como completamente desconhecida. Após descoberta, a disposição atual de estruturas na região é atualizada enquanto unidades do jogador possuírem visibilidade da região. Quando um jogador não possui mais visibilidade de uma região do mapa, ela permanece assim como foi vista da última vez. A figura 4 demonstra a observabilidade parcial que um jogador possui do mapa.

Com relação ao número de agentes, jogos de RTS podem ser abordados de diferentes formas. Pode-se considerar o jogo como adversarial, onde dois ou mais agentes (os jogadores) buscam a vitória na partida, ou pode-se adotar uma abordagem na qual todas as unidades e estruturas de um jogador são consideradas múltiplos agentes e cada um aprende políticas independentemente. Diversas unidades de um mesmo tipo podem ser tratadas como classes de agentes que podem dividir a tarefa de aprender políticas (JAIDEE; MUNOZ-AVILA, 2013) ou como objetos de mesmo tipo cujas relações com outros objetos independe apenas do tipo de objeto (WASSER; COHEN; LITTMAN, 2008). Pode-se ainda ignorar completamente a presença do adversário e tratá-lo como parte da representação do estado do jogo.

Relatos de StarCraft sendo utilizado formalmente como plataforma de estudos para inteligência artificial remetem a competições realizadas em 2010 (ONTAÑÓN, 2014), nas quais uma API não-oficial para *StarCraft: Brood War*, denominada *Brood War Application Programming Interface* (BWAPI), foi utilizada para acessar dados e enviar comandos ao jogo. Tais competições têm por objetivo testar agentes criados pela comunidade em partidas adversariais. As competições continuam ocorrendo em conferências de ciência da computação e inteligência artificial (CERTICKY; SARNOVSKY; VARGA, 2018–0008).

Mais recentemente, *StarCraft II* foi introduzida à comunidade acadêmica como uma plataforma padronizada de testes de algoritmos de aprendizado por reforço profundo, através de um projeto chamado *StarCraft II Learning Environment* (SC2LE) (VINYALS et al., 2017).

3.2 *STARCRAFT MULTI-AGENT CHALLENGE*

O *StarCraft Multi-Agent Challenge* (SMAC) (SAMVELYAN et al., 2019) é uma plataforma para testes de algoritmos de aprendizado por reforço multi-agentes implementada dentro do *StarCraft II*, criado como resposta à existência de plataformas especializadas na disponibilização de múltiplos ambientes com interfaces padronizadas para a experimentação de algoritmos de

Figura 4 – Exemplo de observabilidade parcial em *StarCraft II*

(a) Região do mapa (à direita) não observada pelas unidades do jogador (à esquerda)

(b) Mesma região, agora observada

Fonte: Autor

Tabela 4 – Características das unidades presentes em cenários do SMAC usados neste trabalho. † indica capacidade de atacar unidades aéreas. ‡ indica que o ataque danifica diversas unidades simultaneamente.

Nome	Vida Escudo Armadura	Tipos	Dano/s	Velocidade de Movimento	Dano Extra	
					Contra	Dano/s
Marine	45/0/0	Terrestre, Leve	9,76 †	3,15		
Marauder	125/0/1	Terrestre, Blindado	9,33	3,15	Blindado	9,33
Medivac	150/0/1	Aéreo, Blindado	12,6	3,5		
Stalker	80/80/1	Terrestre, Blindado	9,73 †	4,13	Blindado	3,74
Zealot	100/50/1	Terrestre, Leve	18,67	3,15		
Colossus	200/150/1	Terrestre, Blindado	18,67 ‡	3,15	Leve	9,33

Fonte: Autor

aprendizado por reforço com um agente (como o OpenAI Gym, Brockman et al., 2016) e a comparativa escassez de artefatos semelhantes para o uso na pesquisa com sistemas multi-agentes.

No SMAC, assim como no jogo original, dois times adversários se enfrentam em uma situação de combate. O objetivo para as unidades de um time é derrotar todas as unidades do time adversário. O dano sofrido pelo agente ou seus aliados durante uma partida, assim como a perda de unidades aliadas não é um fator determinante para o resultado. Ao contrário do SC2LE, no qual o controle de todas as unidades é delegada a um agente central (VINYALS et al., 2017, 2019), no SMAC, cada unidade possui observações parciais individuais do ambiente e também devem selecionar ações individualmente.

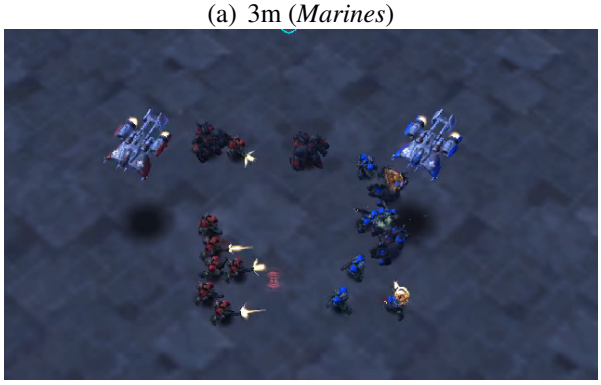
A figura 5 apresenta capturas de tela de uma seleção dos cenários disponibilizados pelo SMAC, os quais serão utilizados neste trabalho. Os cenários possuem diversas características que os diferenciam, como cenários com agentes homogêneos, contendo uma única classe de unidades (figura 5(a)) e heterogêneos, contendo unidades de múltiplas classes (figuras 5(c), 5(b) e 5(e)) e cenários simétricos, nos quais o time de agentes e o time adversário possuem a mesma quantidade de unidades (figuras 5(a), 5(b), 5(b) e 5(e)) e assimétricos, no qual os agentes possuem uma desvantagem no número de unidades (figura 5(d)).

A tabela 4 apresenta as características das unidades do SMAC presentes nos mapas apresentados na figura 5. Os pontos de vida e escudo indicam o dano necessário para que a única seja removida do jogo, como explicado na seção 3.1. A armadura indica um redutor adicional de dano nativo da unidade, cujo funcionamento específico não é de conhecimento dos jogadores. A coluna “Tipos” indica os tipos de cada unidade. “Dano/s” indica o dano de uma ação de ataque

Figura 5 – Capturas de tela de alguns cenários do SMAC

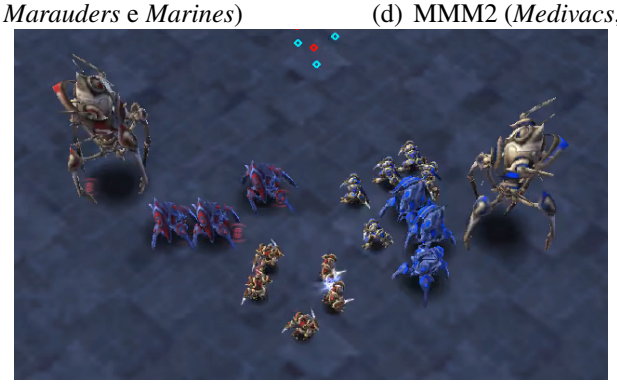
(a) 3m (*Marines*)

(b) 3s5z (*Stalkers* e *Zealots*)



(c) MMM (*Medivacs*, *Marauders* e *Marines*)

(d) MMM2 (*Medivacs*, *Marauders* e *Marines*)



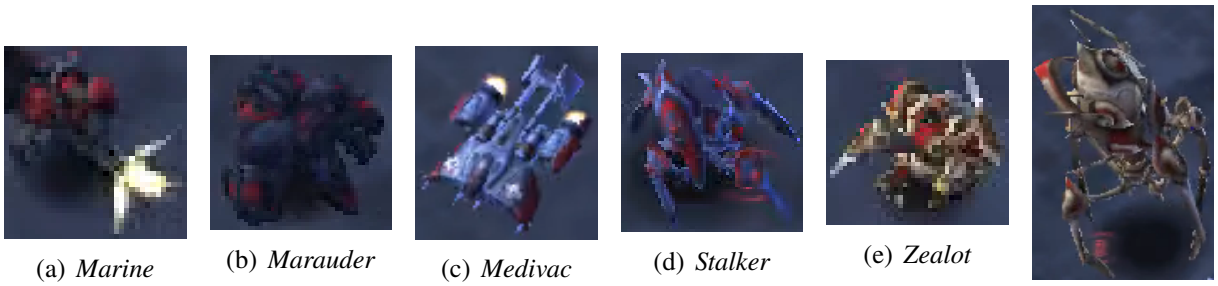
(e) 1c3s5z (*Colossus*, *Stalkers* e *Zealots*)

Fonte: Autor

da unidade dividido pelo intervalo entre animações de ataque. "Velocidade de Movimento" representa a velocidade que unidades se movimentam no mapa, mensurada em um métrica cuja unidade é próxima do tamanho de uma unidade de pequena/médio porte no jogo.

Como mencionado na seção 3.1, os ataques de algumas unidades recebem um bônus de dado, de acordo com as características da unidade sendo atacada. A coluna "Contra" indica o tipo de unidade que recebe maior dano contra o ataque daquela unidade, e a última coluna indica o dano adicional. A figura 6 lista as representações gráficas das unidades listadas na tabela 4.

Figura 6 – Unidades do SMAC presentes nos cenários usados no trabalho



(a) *Marine* (b) *Marauder* (c) *Medivac* (d) *Stalker* (e) *Zealot* (f) *Colossus*

Fonte: Autor

Dadas todas as características listadas de cada uma das unidades presentes nos cenários exemplificados, é possível perceber que cada unidade se beneficia da execução de estratégias específicas, *e.g.* atacar adversários contra os quais seus ataques são mais eficazes, evitar adversários cujos ataques são mais perigosos para si ou se manter a uma distância segura de unidades que atacam a distância. Unidades que atacam a distância podem aprender uma técnica chamada *kiting*, na qual ataques a distância são intercalados com movimentos na direção oposto do adversário, impedindo que este se aproxime para realizar seu ataque enquanto é lentamento danificado. Unidades com escudo têm a opção de recuar e esperar a regeneração do escudo, enquanto unidades massivas podem danificar múltiplas unidades com um único ataque, caso este seja bem posicionado.

3.2.1 Representação do SMAC como um Dec–POMDP

Enquanto alguns autores modelam jogos de RTS (como o *StarCraft*) como SGs (FOERSTER, Jakob et al., 2017a,b), o SMAC é mais comumente formalizado como um Dec–POMDP (FOERSTER, J. N. et al., 2018b; RASHID et al., 2018; SAMVELYAN et al., 2019; RASHID et al., 2020a; LI et al., 2020; HU, J. et al., 2021; HU, S. et al., 2021), visto que a união das observações de todos os agentes não permite necessariamente a reconstrução do estado final, além do ambiente disponibilizar uma recompensa compartilhada entre todos os agentes. Ademais, não existe a suposição da observabilidade conjunta total (*i.e.* a união das observações dos agentes não garante a reconstrução do estado), além das transições e observações não serem independentes, o que significa que a presença de um agente no ambiente afeta outro agente (transições dependentes) e que agentes podem se observar (observações dependentes) (OLIEHOEK; AMATO, 2016).

3.2.1.1 Estados e observações

Originalmente proposto para a pesquisa de algoritmos de MADRL sob um regime de treinamento centralizado e execução decentralizada, o SMAC prevê que os agentes possuam acesso aos estados do ambiente durante o treinamento e a apenas as próprias observações durante a execução dos algoritmos. Estados e observações são compostos das mesmas variáveis, sendo que as observações não contêm os valores das características de unidades fora do campo de visão do agente. As variáveis que descrevem estados e observações para cada agente são listadas a seguir.

Características do agente:

a) ações de movimentação disponíveis: indica quais das 4 ações de movimento são válidas no estado atual;
b) pontos de saúde: normalizados pela quantidade de pontos de saúde máximo que a unidade pode possuir
c) pontos de escudo: normalizados pela quantidade de pontos de escudo máximo que a unidade pode possuir;
d) tipo de unidade: codificado como um vetor *one-hot*. A codificação é exclusiva para cada mapa, englobando apenas os tipos de unidade presentes no mapa e não todos os tipos de unidades presentes no SMAC;

As seguintes características são incluídas na observação do agente, para cada unidade. Caso uma unidade não esteja no campo de visão do agente, as características da unidade em questão são apresentadas como um vetor de zeros.

a) visibilidade: *flag* binária que indica se a unidade é visível;
b) sob alcance: *flag* binária que indica se uma ação de ataque pode ser executada contra a unidade no instante atual (apenas para unidades adversárias);
c) distância euclideana normalizada: a normalização é realizada dividindo-se a distância pelo alcance da visibilidade do agente;
d) posição horizontal relativa ao agente;
e) posição vertical relativa ao agente;
f) pontos de saúde normalizados;
g) pontos de escudo normalizados;
h) tipo de unidade codificado como um vetor *one-hot*.

Outras características opcionais incluem um mapa de altura do terreno e a última ação executada pelas unidades observáveis no último instante de tomada de decisão.

3.2.1.2 Ações

As unidades possuem 4 ações de movimento (norte, sul, leste e oeste), uma ação parametrizada de ataque na forma $atacar(ID)$, onde ID é um identificador único da unidade adversária ou, no caso da unidade *Medivac*, uma ação parametrizada de cura na forma $curar(ID)$, dessa vez um identificador único da unidade aliada; a ação de parar; e a ação *no-op*. O SMAC disponibiliza uma máscara que informa as ações inválidas para cada agente em cada estado, não permitindo a execução de tais ações. Unidades derrotadas possuem apenas a ação *no-op* como válida.

As ações parametrizadas de atacar e curar são discretizadas, tornando-se múltiplas ações discretas de número igual ao número de aliados ou adversários. Isso significa que unidades de tipos equivalentes em mapas distintos podem ter número diferente de ações, condicionado à quantidade de unidades adversárias ou aliadas presentes no mapa.

3.2.1.3 Recompensas

O SMAC disponibiliza recompensas conjuntas para todas as unidades do mesmo time, *i.e.* todos os agentes de um mesmo time observam a mesma recompensa no mesmo estado, independente de quais ações individuais cada um tomou. A recompensa pode ser esparsa (1 para um episódio vitorioso, -1 para uma derrota no episódio, observada no estado terminal) ou densa. Na recompensa densa, o time de agentes é recompensado proporcionalmente a cada instante de tomada de decisão pelo dano infligido às unidades adversárias, por derrotar uma unidade inimiga e por vencer um episódio. A recompensa negativa provém do dano tomado por unidades adversárias, por ter uma unidade aliada derrotada e pela derrota.

A tabela 5 apresenta as recompensas para eventos específicos. Quando múltiplos eventos ocorrem em um mesmo estado, a recompensa resultante é a soma das recompensas dos eventos individuais. Quando um agente incorre dano a uma unidade adversária, a recompensa resultante é proporcional aos pontos de saúde da unidade atacada, sendo 1 equivalente a reduzir os pontos de saúde de uma unidade adversária do máximo para 0.

Uma estratégia comum para guiar o time de agentes à vitória é ignorar todas as recompensas negativas, visto que o dano sofrido e unidades perdidas são irrelevantes, contanto que o time

Tabela 5 – Valores padrão das recompensas no ambiente do *StarCraft Multi-Agent Challenge*

Evento	**Recompensa**
Infligir dano na unidade adversária	$(0; 1]$
Sofrer dano	$[-1; 0)$
Derrotar uma unidade adversária	10
Perder uma unidade	-10
Vitória	200
Derrota	0

Fonte: Autor

vença o episódio. Isso elimina a necessidade de preservar as próprias unidades, facilitando o desafio para os agentes.

3.3 RESUMO

Esta seção apresentou os jogos de RTS e seus principais desafios como um domínio de estudo e aplicação de técnicas de aprendizado por reforço, a saber: espaços de estados e ações consideravelmente grandes, observações parciais e, em alguns casos, não-determinismo. Dentro do gênero, o jogo *StarCraft II* recebeu considerável atenção da comunidade acadêmica nos últimos anos, devido à sua contemporaneidade, complexidade e apelo comercial.

Adaptando o ambiente do *StarCraft II* para a pesquisa em Aprendizado por Reforço Multi-Agentes, o *StarCraft Multi-Agent Challenge* permite que cada unidade seja controlada como um agente individual, com observações parciais do ambiente baseadas no campo de visão do agente. A heterogeneidade nas características dos agentes, implementadas no *StarCraft II* para trazer uma esperiência mais diversificada, complexa e desafiadora para jogadores humanos, se traduz, dentro do SMAC, em um desafio e uma oportunidade de pesquisa para a criação de métodos que sejam capazes de usar as distinções entre os agentes no aprendizado de políticas melhores para o time de agentes como um todo.

4 TRABALHOS RELACIONADOS

Os métodos propostos neste trabalho foram influenciados por trabalhos de duas áreas distintas. O primeiro grupo de trabalhos são aqueles da área de Aprendizado por Reforço Profundo Multi-Agentes que incorporam grafos em suas representações e utilizaram GNNs para realizar passagem de mensagens, principalmente entre agentes.

No segundo grupo estão os trabalhos que apresentam avanços recentes na área de MADRL em ambientes totalmente cooperativos, onde outros desafios, além da comunicação, foram enfrentados, como o aprendizado em ambientes com observações parciais, a aceleração do aprendizado através do compartilhamento de parâmetros entre agentes e o aprendizado de funções-valor para times de agentes sobre suposições de decomposição.

No entanto, os trabalhos presentes nessas categorias, cujo número cresceu consideravelmente nos últimos anos, são precedidos por ideias publicadas anteriormente, as quais também possuem relações com esta tese. Para modelar a coordenação em sistemas multi-agentes nos quais a coordenação ocorre entre subconjuntos de agentes responsáveis por partes de um sistema, Guestrin, Lagoudakis e Parr (2002) adaptam os MDPs fatorados (apresentados em Boutilier, Dean e Hanks, 1999 para sistemas com um único agente) de forma a simplificar a descoberta de soluções ótimas para sistemas multi-agentes sem tratá-los como MDPs monolíticos. Guestrin, Koller e Parr (2002) e Guestrin, Venkataraman e Koller (2002) expandem a fatoração de MDPs para sistemas multi-agentes nos quais os grupos de agentes que devem se coordenar variam com o tempo. Cada estado tem um *grafo de coordenação* associado a si, no qual os vértices são agentes a função valor-ação de um agente é uma combinação aditiva das funções valor-ação dos agentes vizinhos. Os MDPs fatorados são implementados utilizando redes neurais por Castellini et al. (2019) e os grafos de coordenação por Boehmer, Kurin e Whiteson (2020).

Guestrin et al. (2003b) apresentam os Processos de Decisão de Markov Relacionais (RMDPs, *Relational Markov Decision Process*), nos quais os estados são representados por conjuntos de objetos, os quais são associados a classes. Cada classe de objetos aprende uma função valor independente, sendo a função valor do RMDP composta pela soma das funções valor-ação de todos os objetos, assim como a recompensa num dado estado. Os RMDPs acomodam a seleção de ações por um único agente central, um único objeto, ou múltiplos objetos simultaneamente, constituindo um sistema multi-agentes heterogêneo.

O trabalho de Kapetanakis e Kudenko (2005) é um dos primeiros a abordar a tarefa de cooperação entre agentes heterogêneos sem comunicação, através de um algoritmo heurístico em

que cada agente extrai informação sobre as ações de outros agentes enquanto interage diretamente com o ambiente.

Uma ideia comparável à comunicação entre agentes heterogêneos explorada neste trabalho é a cooperação *ad hoc* (STONE et al., 2010), um paradigma no qual agentes sem contato prévio (e, provavelmente, heterogêneos) devem ser capazes de cooperação entre si ao se encontrarem no ambiente. A descoberta de novos tipos de agentes no ambiente implica na cooperação sem coordenação prévia.

Essa seção apresentou os trabalhos que são tanto relacionados a esta tese como constituem os fundamentos para os próximos trabalhos relacionados apresentados, os quais são consideravelmente mais recentes e serão apresentados a seguir.

4.1 COMUNICAÇÃO ENTRE AGENTES COM REDES NEURAIS DE GRAFOS

No trabalho que apresenta a NerveNet (WANG, Tingwu et al., 2018), o esqueleto de uma criatura tridimensional é representado como um grafo heterogêneo, no qual articulações que desempenham funções similares são categorizadas como vértices de mesma classe. Cada vértice aprende uma única ação contínua e é treinado usando o algoritmo *Proximal Policy Optimization* (PPO). Apesar de ser o primeiro trabalho que modela grupos de agentes heterogêneos em uma tarefa cooperativa, uma limitação do método está na realização do aprendizado e aplicação da política aprendida em um grafo mesmo grafo, cuja topologia não se modifica (*i.e.* a topologia do grafo é condicionada ao domínio e não ao estado, como neste trabalho). Uma diferença entre os trabalhos é o aprendizado por parte da NerveNet de uma única ação contínua utilizando gradiente de políticas.

Agarwal, Kumar e Sycara (2019) representam estados de sistema multi-agentes como grafos de agentes (homogêneos) e entidades (também homogêneas) e utilizam um mecanismo de atenção Transformer (VASWANI et al., 2017) para passagem de mensagens entre agentes e entidades, treinando agentes homogêneos em um domínio multi-agentes de partículas em tarefas relacionadas a manter formações de agentes. A modelagem de entidades como parte do grafo de passagem de mensagens foi abordada na primeira versão do método apresentado neste trabalho, mas foi toada como uma limitação do método, visto que a inclusão de entidades sem agência ou capacidade de comunicação não pode ser considerada uma tarefa viável em cenários reais.

Jiang et al. (2020) propõem um trabalho, denominado DGN, no qual times de agentes homogêneos em cenários totalmente cooperativos são modelados como um grafo de agentes e

suas observações são compartilhadas através de convoluções de grafos. Após múltiplas camadas de convoluções em grafos, cada agente acumulou informação relacionada às observações de todos os agentes no grafo e utilizam o vetor resultante de múltiplas etapas de comunicação na seleção de suas ações. Para acelerar o aprendizado, os autores realizam o compartilhamento de parâmetros das camadas de codificação e função valor-ação da rede neural, facilitado pela presença de apenas um tipo de agentes e aprendizado de uma única política para todos os agentes.

A técnica Aprendizado de Política baseado em Grafos (GPL, *Graph-based Policy Learning*) (RAHMAN et al., 2020) se apresenta como um passo rumo à cooperação *ad hoc*, utilizando grafos de coordenação para modelar grupos de agentes em busca de coordenação e aplicando convoluções em grafos como uma forma de decompor a função valor-ação de times de agentes formados de maneira *ad hoc*. Algumas limitações do método incluem o uso de tipos de agentes em cenários com ambientes homogêneos, a suposição de observação total do estado por todos os agentes e o fato do método não lidar com ações parametrizadas (MASSON; RANCHOD; KONIDARIS, 2016), limitações estas enfrentadas pelo método proposto neste trabalho.

Shantian Yang et al. (2021) abordam o problema de controle de semáforos em redes de tráfego de veículos, transformando as redes em grafos heterogêneos, nos quais os vértices representam veículos, pistas, cruzamentos e sistemas de controle de semáforos, e as arestas indicam a localização dos veículos nas pistas e das pistas e semáforos nos cruzamentos. As características de diferentes tipos de entidades são codificadas usando redes específicas e um algoritmo ator-crítico utiliza estados representados na forma destes grafos heterogêneos para aprender a controlar os semáforos, de forma a diminuir o tempo de viagem dos veículos.

A tabela 6 apresenta os trabalhos descritos nesta seção e suas principais características.

Por fim, é importante mencionar que também há trabalhos que utilizam comunicação entre agentes utilizando GNNs, porém em cenários competitivos, nos quais agentes aprendem políticas e observam recompensas distintas uns dos outros (BLUMENKAMP; PROROK, 2020).

Tabela 6 – Trabalhos que realizam comunicação em sistemas multi-agentes, em tarefas de aprendizado por reforço, através de redes neurais de grafos

Método	**Função aproximada**	**Algoritmo de aprendizado**	**Tipos de agentes**	**Grafos mutáveis**	**Ações**
NerveNet	Política	PPO	Heterogêneos	Não	1, contínua
Agarwal	Política	PPO	Homogêneos	Não	Discretizadas
DGN	Função valor-ação	DQN	Homogêneos	Sim	Discretas
GPL	Função valor-ação conjunta	GPL	Homogêneos, múltiplos tipos	Sim	Discretas
IHG-MA	Política e função valor-ação	Ator-crítico	Homogêneos	Sim	Discretas
Este trabalho	Função valor-ação (individual ou conjunta)	DDRQN Duelista	Heterogêneos	Sim	Discretas parametrizadas

Fonte: Autor

4.2 APRENDIZADO POR REFORÇO PROFUNDO MULTI-AGENTES COOPERATIVO

Assim como descrito na seção 2.2.3, a fatoração ou decomposição da função valor-ação para um grupo de agentes num cenário cooperativo (KOLLER; PARR, 1999; GUESTRIN et al., 2003a) foi explorada no ambiente das redes neurais profundas (SUNEHAG et al., 2018; RASHID et al., 2018; SON et al., 2019; MAHAJAN et al., 2019; RASHID et al., 2020a,b; WANG, Y. et al., 2020; YANG, Y. et al., 2020; DE WITT et al., 2020; BOEHMER; KURIN; WHITESON, 2020). Essas técnicas, denominadas de *mixing*, também foram estendidas para redes neurais que aproximam políticas (WANG, J. et al., 2020; ZHOU et al., 2020). Neste trabalho, uma destas técnicas, a VDN (SUNEHAG et al., 2018), é utilizada para se aferir o desempenho do *mixing* em conjunto com a comunicação heterogênea.

Sun, Li e Belta (2020) alcançam coordenação entre agentes utilizando máquinas de estados finitos e uma linguagem comum para compartilhamento de informação entre agentes. No trabalho, a heterogeneidade entre agentes é definida pela capacidade de cada agente em executar diferentes tarefas em um sistema multi-tarefas definido como um em Jogo Estocástico, não um Dec–POMDP, diferenciando-o consideravelmente deste trabalho.

Zheng et al. (2020) propõem um método de DRL distribuído para treinamento de um único agente, no qual múltiplos agentes propositalmente exploram diferentes locais no espaço de soluções de um MDP para formar uma memória de repetição mais abrangente, fazendo deste um método de treinamento para um único agente composto de múltiplos agentes heterogêneos.

Em uma aplicação de controle de tráfego, Calvo e Dusparic (2018) treinam agentes heterogêneos usando DDQNs duelistas independentes (TAN, 1993), na qual os agentes não realizam comunicação entre si. Huang e Liu (2021) implementa comunicação por mecanismo de atenção entre *drones* homogêneos que executam tarefas distintas.

Um trabalho contemporâneo a este que trabalha com a heterogeneidade de agentes no SMAC é o de Tonghan Wang et al. (2020b). Neste trabalho, um número pré-determinado de papéis de agentes é disponibilizado e agentes se associam a papéis de acordo com a similaridade de seus históricos de observações. Os autores descrevem que, apesar de as observações de todos os agentes provirem de um mesmo conjunto, agentes associados a papéis distintos consideram mais relevantes os mesmos subconjuntos de variáveis que compõem suas observações. O método, denominado Aprendizado por Reforço Multi-Agentes Orientado a Papéis (ROMA, *Role-Oriented Multi-Agent Reinforcement Learning*), alcançou desempenho similar ou superior ao QMIX (RASHID et al., 2018, 2020a) em uma amostra de mapas do SMAC.

Os resultados ablativos do ROMA validam alguns resultados apresentados na seção 5, assim como as melhorias apresentadas na seção 6, a saber, não realizar o compartilhamento de parâmetros nas redes de aprendizado de função valor-ação tende a prejudicar o aprendizado dos agentes. O método apresentado nesta tese e o ROMA diferem em suas definições de classes e papéis. Aqui, classes são características inerentes aos agentes (suas observações, ações e política ótima), enquanto no ROMA, papéis diferenciam as políticas que agentes aprendem, mas são condicionados às similaridades dos históricos de observações dos agentes. Ademais, o ROMA não estuda a comunicação entre agentes de forma alguma.

Uma estratégia diferente de criação de papéis e associação de agentes a papéis é proposta por Tonghan Wang et al. (2020a). A nova técnica, denominada *Roles to Decompose* (RODE), cria *embeddings* para ações discretas, construídos por uma rede neural de codificação para prever a observação futura e recompensa, dada uma ação. Essa rede de codificação deve ser pré-treinada por um número de instantes de tempo antes das próximas etapas.

As ações são então agrupadas utilizando k*-means*, medindo-se a distância Euclidiana entre seus *embeddings*, e cada grupo de ações é associado a um de K papéis de agentes. Os agentes são então associados a papéis utilizando uma rede QMIX (RASHID et al., 2018, 2020a) que associa cada agente ao papel que maximize o valor das ações para o time de agentes e, por fim, cada agente seleciona uma das ações associadas a seu papel utilizando uma segunda rede QMIX. Os agentes permanecem em seus papéis por uma quantidade pré-determinada de instantes de tempo até poderem trocar de papel novamente.

Tonghan Wang et al. (2020a) testam o RODE no SMAC, fixando o número de papéis para 2 em cenários com um único agente; 3 em cenários com múltiplos agentes homogêneos e 5 em cenários com múltiplos agentes heterogêneos, alcançando o estado da arte em 10 de 14 cenários. Diferenças com este trabalho incluem a limitação de agentes à exploração de um subconjunto de ações para cada papel de agente, além da suposição de que todos os agentes compartilham um mesmo espaço de ações, o que não é feito aqui. Naderializadeh et al. (2021) combinam o RODE com um módulo de comunicação por mecanismo de atenção homogêneo, reportando desempenho igual ou superior ao RODE sem comunicação.

No recente trabalho de Jian Hu et al. (2021), o desempenho de diversas técnicas de *mixing* foi analisado em múltiplos cenários do SMAC; dentre essas técnicas, a VDN (SUNEHAG et al., 2018), o QMIX (RASHID et al., 2018, 2020a) e suas variantes. Os autores concluem que, em discordância com a tendência de trabalhos publicados posteriormente afirmarem alcançar resultados superiores a trabalhos predecessores, quase todos os trabalhos que assumem algum

nível de monotonicidade da função valor-ação podem alcançar resultados similares e próximos a 100% de vitórias em diversos cenários do SMAC caso uma busca pelos hiperparâmetros corretos seja feita e a topologia adequada de rede neural seja utilizada.

4.3 PASSAGEM DE MENSAGEM EM GRAFOS HETEROGÊNEOS

Esta seção revisa os trabalhos que apresentam métodos de passagem de mensagem entre vértices de grafos heterogêneos atribuídos com treinamento realizado de maneira supervisionada ou semi-supervisionada.

Sun et al. (2011) foram os primeiros a aplicar um algoritmo de aprendizado supervisionado a grafos heterogêneos, o qual foi pioneiro no uso de meta-caminhos para calcular similaridades entre vértices. Um meta-caminho é uma sucessão de classes de C e Ω, da seguinte forma: $c_1 \xrightarrow{\omega_1} c_2 \xrightarrow{\omega_2} \cdots \xrightarrow{\omega_n} c_{n+1}$. Meta-caminhos possibilitam descobrir informações em grafos heterogêneos, como relações de co-autores em bases de obras acadêmicas, utilizando a seguinte expressão: $autor \xrightarrow{escreveu} obra \xrightarrow{escrita\ por} autor$.

Apesar de seu pioneirismo, os meta-caminhos são mais úteis em aplicações que preveem a agregação de informações através de longos percursos pelas arestas de um grafo, além de precisarem ser definidos por especialistas no domínio da base de dados (como no exemplo de co-autores acima).

Xiao Wang et al. (2020) apresentam uma revisão de métodos de aprendizado de máquina aplicados a grafos heterogêneos, em tarefas de aprendizado de *embeddings* para grafos não-atribuídos ou para vértices individuais nestes grafos; predição de conexões em vértices de novos grafos, dada uma base de dados de grafos com conexões pré-existentes; passagem de mensagens entre vértices de grafos; e identificação de vértices anômalos ou de interesse. A tabela 7 apresenta os trabalhos encontrados na literatura cujas aplicações se assemelham às realizadas nesta tese, isto é, a passagem de mensagem entre vértices de grafos heterogêneos atribuídos com treinamento realizado de maneira supervisionada ou semi-supervisionada.

Alguns trabalhos realizam o aprendizado de *embeddings* para vetores de características de grafos atribuídos heterogêneos de maneira não-supervisionada (ZHANG, C. et al., 2019; CEN et al., 2019), visando encontrar representações compactas para bases de grafos heterogêneos, representações essas que podem ser aplicadas posteriormente em outras tarefas, como predição de relações ou classificações de vértices. Esses métodos usam passeios aleatórios na vizinhança

Tabela 7 – Trabalhos que propõem métodos de passagem de mensagem entre vértices de grafos heterogêneos atribuídos com treinamento realizado de maneira supervisionada ou semi-supervisionada. Trabalhos de aprendizado não-supervisionado foram incluídos devido à pequena quantidade existente. GAT se refere ao trabalho de Veličković et al. (2018).

Referência	**Meta-caminhos**	**Mecanismo de atenção**	**Supervisão**
Chuxu Zhang et al. (2019)	N	GAT	Não
Cen et al. (2019)	S	(LIN et al., 2017)	Não
Xiao Wang et al. (2019)	S	GAT semântica	Semi
Fu et al. (2020)	S	GAT	Semi
Hong et al. (2020)	N	GAT heterogênea	Semi
Ziniu Hu et al. (2020)	N	Transformer heterogêneo	Semi
Yun et al. (2019)	S (aprendidos)	N	Semi
Schlichtkrull et al. (2018)	N	N	Semi

Fonte: Autor

de cada vértice para coletar informações sobre sua vizinhança de um vértice na geração de seu *embedding*.

Métodos que encontram *embeddings* para grafos heterogêneos atribuídos de maneira semi-supervisionada (*embeddings* otimizados para tarefas específicas) utilizando meta-caminhos incluem os trabalhos de Xiao Wang et al. (2019) e Fu et al. (2020). Yun et al. (2019) propõe um método de geração automática de meta-caminhos, os quais são processados por convoluções em grafos comuns, utilizadas em grafos homogêneos (KIPF; WELLING, 2017).

Já Schlichtkrull et al. (2018), Hong et al. (2020) e Ziniu Hu et al. (2020) abandonam o uso de meta-caminhos em prol de modelar as relações entre vértices distantes nos grafos empregando múltiplas camadas na rede neural. Cada camada emprega operações individuais de propagação, agregação e atualização nos vizinhos imediatos de cada vértice, como as descritas nas equações 12 e 13, sendo que camadas sucessivas agregam informações de vértices gradualmente mais distantes do vértice em que a operação foi originada. Eles também são os primeiros trabalhos que transformam os vetores de características de vértices e/ou arestas de tipos diferentes usando diferentes conjuntos de parâmetros, eliminando possíveis dependências entre os tipos, ao custo de maior complexidade de treinamento da rede.

Hong et al. (2020) emprega um conjunto de funções específicas para a codificação dos vetores de características de diferentes tipos de vértices, seguido de funções de atenção específicas para ponderar a significância dos vetores de características de vértices conectados por diferentes tipos de arcos, terminado por uma somatória de agregação. O uso de diferentes

funções para diferentes tipos de vértices e arestas também é empregado por Ziniu Hu et al. (2020), o qual expande a arquitetura dos Transformers (VASWANI et al., 2017) para processar grafos heterogêneos. Por último, Schlichtkrull et al. (2018) modelam apenas relações entre vértices, representadas pelos tipos das arestas, na tarefa de classificação de vértices.

4.4 RESUMO

Esta seção descreveu os trabalhos relacionados aos métodos apresentados nesta tese. A seção 4.1 mapeou trabalhos que utilizam redes neurais de grafos para o controle de agentes heterogêneos em cenários com uma unica ação e grafos de topologia fixa, ou que trabalham apenas com agentes homogêneos, ou que assumem observabilidade total do ambiente, ou aplicados a cenários competitivos.

Já na seção 4.2 foram mapeados os trabalhos recentes que visaram resolver o problema MADRL cooperativo sem explorar estratégias de comunicação. Esses trabalhos evoluíram os conceitos clássicos de fatoração e decomposição da função valor-ação, utilizando redes neurais profundas. Também foram exploradas ideias de associação de agentes a papéis, com resultados promissores.

Alguns dos trabalhos da seção 4.2 foram aplicados no SMAC, cada um demonstrando resultados melhores que seus antecessores. No entanto, quase todos os trabalhos foram demonstrados como possuindo desempenhos similares quando uma busca correta por hiperparâmetros e alguns truques na implementação dos algoritmos são levados em consideração. Além disso, nenhum dos trabalhos nessa seção abordou a possibilidade de comunicação entre agentes, muito menos de comunicação especializada entre classes de agentes.

Por último, a seção 4.3 apresentou trabalhos que de fato operam sobre grafos de vértices ou arestas heterogêneas. Estes trabalhos se concentram nas áreas de aprendizado não-supervisionado e semi-supervisionado, necessitando de adaptações caso sejam aplicados em tarefas de Aprendizado por Reforço Multi-Agentes.

5 APRENDIZADO POR REFORÇO PROFUNDO MULTI-AGENTES ATRAVÉS DE GRAFOS HETEROGÊNEOS DE AGENTES E ENTIDADES

Este trabalho tem por objetivo propôr arquiteturas de redes neurais especializadas no aprendizado de políticas em tarefas de aprendizado por reforço multi-agentes totalmente cooperativas compostas por times de agentes heterogêneos e com capacidade de comunicação.

Esta seção apresenta o primeiro de dois modelos contidos na tese, o qual se inspira em trabalhos contemporâneos de duas vertentes distintas. A primeira é a vertente de aprendizado supervisionado e semi-supervisionado aplicado a grafos relacionais, cujos vértices representam entidades de tipos distintos e as arestas, relações também possivelmente distintas entre as entidades. Esses trabalhos dofram descritos na seção 4.3.

A segunda vertente de trabalhos correlatos realiza o aprendizado por reforço multi-agentes utilizando convoluções em grafos para comunicação entre agentes homogêneos, descritos na seção 4.1.

O método apresentado nesta tese foi coletivamente denominado de Rede Neural de Agentes Heterogêneos e Comunicativos (HCA-Net, *Neural Network for Heterogeneous Communicative Agents*). O primeiro modelo se chama Rede Neural de Agentes Heterogêneos e Comunicativos – Características de Entidades (HCA-Net(EF), *Neural Network for Heterogeneous Communicative Agents – Entity Features*), por trabalhar com grafos de agentes e entidades, nos quais os vértices armazenam as *características* desses agentes e entidades.

Aqui, agentes são definidos como o são em RL, enquanto o termo "entidade" é utilizado para descrever quaisquer elementos adicionais do ambiente cuja representação em um estado seja necessária para adicionar informação o suficiente no estado de forma a permitir que os agentes possam aprender suas políticas.

Nesta seção, são descritos os dois elementos que compõem a HCA-Net(EF). O primeiro é uma representação dos estados do ambiente como grafos heterogêneos de agentes e entidades. O segundo é a rede neural que processa tais grafos para aprender políticas para os agentes. Os resultados, aqui apresentados, foram publicados em Meneghetti et al. (2020).

5.1 REPRESENTAÇÃO DE ESTADOS COMO GRAFOS DE AGENTES E ENTIDADES

Esta seção apresenta o conceito de um domínio de agentes e entidades heterogêneas, o qual dita um conjunto de classes às quais agentes e entidades se associam, e um conjunto de relações que associam pares de classes de agentes ou pares de classes de agentes e entidades.

Múltiplos grafos provenientes de um mesmo domínio compartilham as mesmas classes de vértices. Um grafo que pertence a um domínio de agentes e entidades heterogêneas é chamado de um grafo heterogêneo de agentes e entidades, uma representação de um estado num sistema multi-agentes em forma gráfica que codifica agentes, entidades sem agência, as classes de ambos, sua capacidade de comunicação e as relações entre si, em um único objeto.

Definição 12 – Domínio de agentes e entidades heterogêneas. Um domínio de agentes e entidades heterogêneas é uma tupla $\mathcal{D} = (C, \Omega)$, onde:

a) C: conjunto de classes de vértices. Cada vértice de um grafo que pertence a $\mathcal{D}$ pertence a uma e apenas uma classe c de C, onde c é armazenada como o rótulo de v;

b) Ω: conjunto de possíveis relações entre vértices. Dado um arco $e = (v_1, u)$ que liga um vértice qualquer v_1 a um agente u e uma função $C(v)$ que retorna a classe de um vértice v, a relação entre v_1 e u é dada por $(C(v_1), C(u))$ e é armazenada no rótulo de e.

Definição 13 – Grafo heterogêneo de agentes e entidades. Um grafo heterogêneo de agentes e entidades é um grafo heterogêneo direcionado rotulado atribuído $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ que pertence a um domínio $\mathcal{D}$, no qual:

a) $\mathcal{V}$: conjunto de vértices, no qual cada vértice $v \in \mathcal{V}$ representa um agente ou uma entidade sem agência no estado original. As características de v são armazenadas em seu vetor de características, representado por $\vec{x}_v$;

b) $\mathcal{E}$: conjunto de arcos direcionados. Um arco $e = (v, u)$ tem como origem um vértice v qualquer e destino um agente u, indicando a capacidade de u observar $\vec{x}_v$.

Dado um estado s disponibilizado pelo ambiente, um grafo $\mathcal{G}$ é gerado, no qual cada vértice $v \in \mathcal{V}(\mathcal{G})$ representa uma entidade no estado original. A cada vértice é associada uma classe c de um conjunto de classes C. A classe de um vértice pode ser considerada o rótulo daquele vértice, o que torna $\mathcal{G}$ um grafo rotulado. A classe de um vértice v pode ser recuperada utilizando uma função $C(v)$.

Um vértice v armazena em si um vetor de características $\vec{x}_v$, o qual descreve v. A dimensão de $\vec{x}_v$ é ditada por $C(v)$, de forma que todos os vértices de uma mesma classe são representados por vetores de características de mesma dimensão, denotada por $\mathbb{R}^{d_{C(v)}}$.

Um subconjunto de classes $Z \subseteq C$ representa as classes de agentes, que são associadas apenas a entidades do estado original para as quais uma política deve ser aprendida. Para uma

determinada classe de agentes $c \in Z$, espera-se que todos os agentes $u \in c$ compartilhem o mesmo conjunto de ações A_c e sejam controlados por uma mesma política π_c.

Quando um agente u possui a capacidade de observar as características $\vec{x}_v$ de outra entidade v, um arco (v, u) é adicionado ao conjunto de arcos de $\mathcal{G}$, $\mathcal{E}(\mathcal{G})$. O arco (v, u) é direcionado, tendo como origem v e destino u, o que inclui v no conjunto de vizinhos de entrada de u, denotado por $\mathcal{N}^-{}_{(u)}$; enquanto u é adicionado ao conjunto de vizinhos de saída de v, denotado por $\mathcal{N}^+{}_{(v)}$.

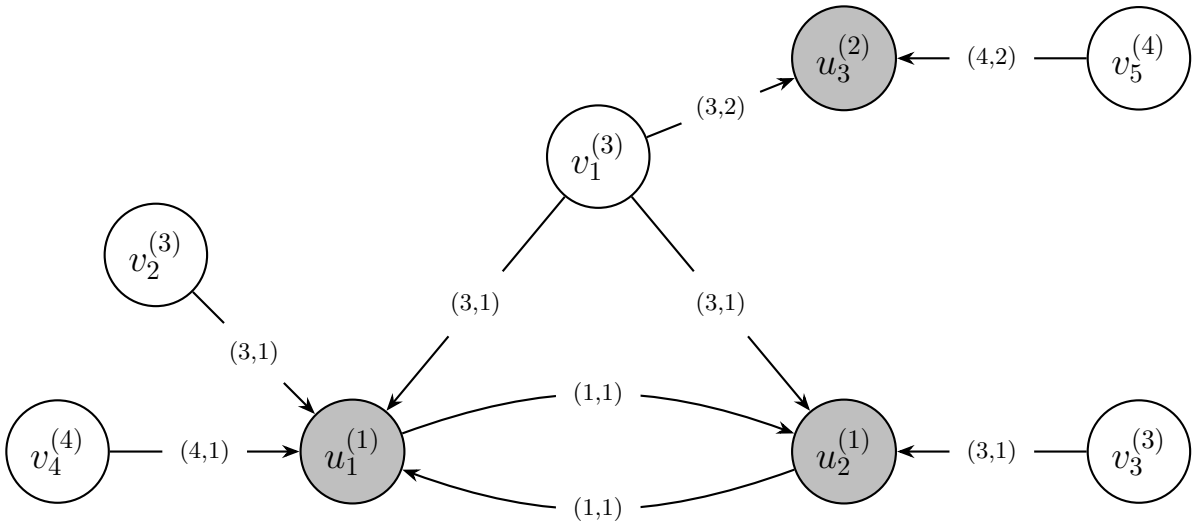
A todo arco é associado um rótulo $(C(v), C(u))$, o qual identifica as classes de seus vértices de entrada e saída. Da mesma forma que diversos vértices podem compartilhar a mesma classe, diversos arcos podem compartilhar o mesmo rótulo, indicando um canal de passagem de mensagens em comum entre vértices de classes semelhantes.

Ao conjunto de todos os possíveis rótulos entre vértices é dado o nome de conjunto de relações Ω. Ele é definido por todos os pares ordenados possíveis de classes $(c_1, c_2), c_1 \in C, c_2 \in Z$, totalizando $|C| \cdot |Z|$ possíveis relações.

A figura 7 exemplifica um grafo com três agentes e outras cinco entidades. Na figura, $v_i^{(j)}$ representa o vértice v com índice i e classe j. Vértices escuros representam agentes, enquanto vértices claros representam outras entidades presentes no estado, codificadas no grafo. É importante reparar que todo arco sempre possui um agente como destino, indicando a capacidade do agente observar o vértice-fonte do arco. A capacidade hipotética de entidades sem agência observarem outros vértices não é de interesse para o método, uma vez que a HCA-Net(EF) não aprende políticas para entidades, apenas para agentes. A presença de entidades nos grafos é feita para que os agentes possam observar mais informações sobre o estado em que o ambiente se encontra.

Utilizando esta representação de estados como grafos, é possível representar estados com diferentes números de agentes e entidades: a ausência ou inclusão de um agente ou entidade em um estado s_2, em comparação com um estado s_1, é denotada pela remoção ou adição do vértice que representa tal agente ou entidade no grafo $\mathcal{G}_2$, em comparação com o grafo $\mathcal{G}_1$. Essa flexibilidade na representação do estado pode ser útil em determinados domínios, como o apresentado na seção 3.

Figura 7 – Exemplo de um grafo heterogêneo de agentes e entidades



Fonte: Autor

5.2 TOPOLOGIA DA REDE NEURAL

A rede neural empregada neste trabalho se utiliza das informações codificadas no grafo apresentado na seção 5.1 para operar com entidades representadas por vetores de características de diferentes dimensões e aprender as políticas para agentes de classes distintas individualmente.

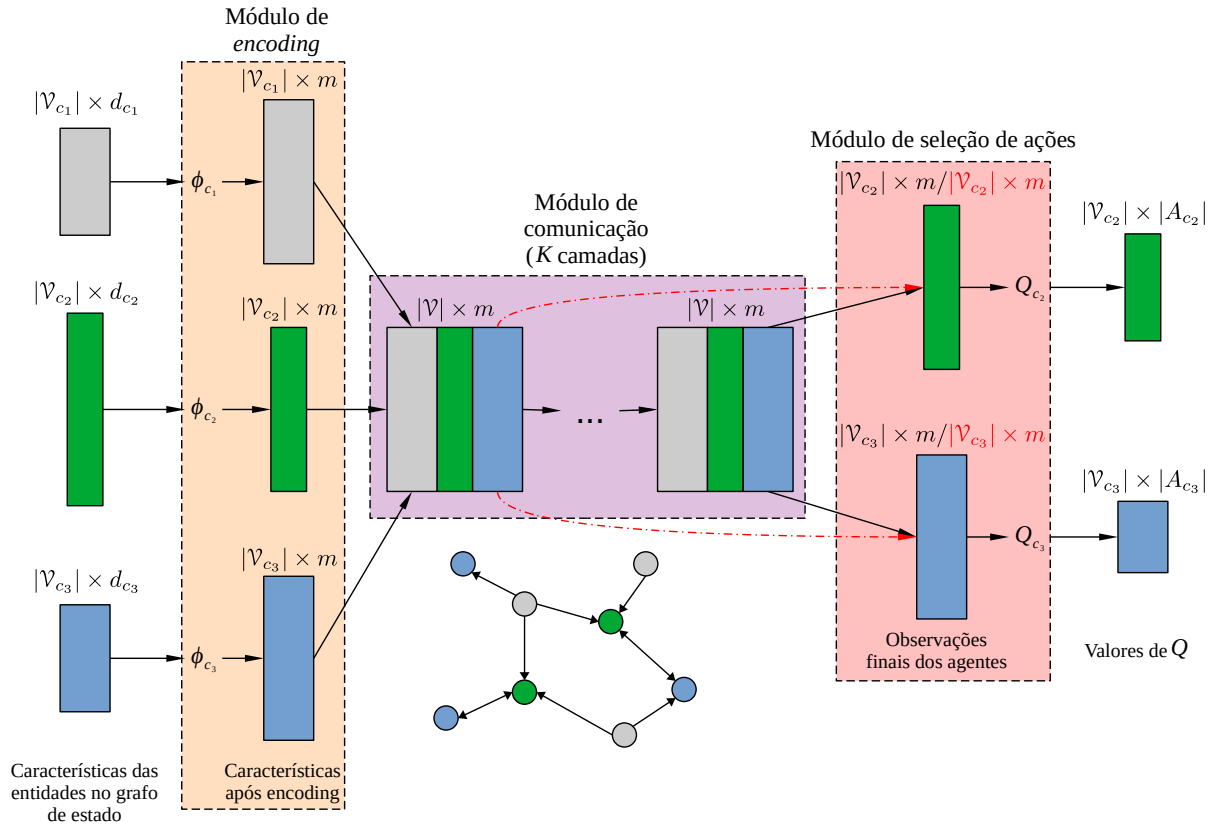
A figura 8 apresenta a topologia da rede neural. Ela recebe como entrada um grafo heterogêneo de agentes e entidades, realiza a codificação das características de todos os vértices, a subsequente passagem de mensagens entre agentes e suas respectivas vizinhanças de entrada e, por último, a aproximação da função valor-ação Q para todos os agentes, utilizando funções individuais para cada classe de agentes.

Para organizar a explicação da topologia, ela foi organizada em três módulos distintos, descritos a seguir.

5.2.1 Módulo de codificação

Para lidar com a variabilidade na dimensão dos vetores de características de diferentes classes de entidades, uma função de codificação ϕ_c para cada classe $c \in C$, a qual recebe um vetor de características $\vec{x}_v \in \mathbb{R}^{d_{C(v)}}$ como entrada e tem como saída um novo vetor $\phi_c(\vec{x}_v) \in \mathbb{R}^m$, onde m é uma dimensão de saída em comum para todas as funções de codificação de todas as classes.

Figura 8 – Topologia de rede neural proposta



Fonte: Autor

5.2.2 Módulo de comunicação

O objetivo do módulo de comunicação é permitir a um grupo de agentes compartilharem as informações armazenadas em seus vetores de características. Enquanto os primeiros trabalhos que abordaram a tarefa de comunicação entre agentes implementaram-na como um mecanismo de passagem de mensagens utilizando um protocolo pré-definido (BALCH; ARKIN, 1994) ou transmitindo as observações do agente diretamente (TAN, 1993), trabalhos contemporâneos (listados em detalhes na seção 4.1) exploraram o uso de operações de GNNs para o aprendizado de protocolos de comunicação inter-agentes.

Seguindo essa tendência, este trabalho propõe um mecanismo de comunicação relacional, no qual pares de classes de agentes aprendem protocolos de comunicação unidirecionais distintos entre si. Para fins de comparação, uma variação de módulo de comunicação também é apresentada, a qual utiliza um mecanismo de atenção para grafos para realização da comunicação inter-agentes.

5.2.2.1 Comunicação por convoluções relacionais em grafos

O mecanismo de comunicação implementado neste trabalho visa utilizar a informação proveniente às classes de agentes e entidades presentes no grafo heterogêneo descrito na seção 5.1 na criação de um método de comunicação relacional entre classes de agentes. Seguindo a classificação dos métodos de aprendizado de máquina aplicados a grafos heterogêneos proveniente de Xiao Wang et al. (2020), cuja revisão é realizada na seção 4.3, o método aplicado deve:

a) considerar as informações das classes dos vértices e arestas de um grafo heterogêneo;
b) considerar os vetores de características contidos nos vértices do grafo;
c) realizar aprendizado supervisionado ou semi-supervisionado, através da função de erro empregada para o treinamento da rede neural na tarefa de MADRL. Essa característica se contrasta com a de aprendizado não-supervisionado, na qual uma GNN aproxima *embeddings* para vértices, arestas ou classes destes através de meta-caminhos ou funções de distância e similaridade;
d) ser aplicável a tarefas genéricas de passagem de mensagem, seguindo *e.g.* a estrutura de Redes Neurais de Passagem de Mensagens (MPNNs, *Message-Passing Neural Network*) proposta por Gilmer et al. (2017), na qual uma camada de uma GNN é dividida em operações de propagação, agregação e atualização (como apresentadas na seção 2.4) que podem ser treinadas em e aplicadas a múltiplos grafos de topologias arbitrárias de um mesmo domínio.

Dadas as características supracitadas, decidiu-se pelo uso de operações de Convolução Relacional em Grafos (RGCN, *Relational Graph Convolution*) (SCHLICHTKRULL et al., 2018), as quais permitem a especialização da passagem de mensagens entre vértices de diferentes classes. As Convoluções Relacionais em Grafos (RGCNs, *Relational Graph Convolution*) foram originalmente propostas para possibilitar que a rede neural realize transformações distintas nos vetores de vértices ligados por arestas de tipos de relação diferentes, permitindo a especialização dessas transformações através do uso de diferentes matrizes de parâmetros para diferentes tipos de relação.

Em uma camada RGCN, o vetor de características $\vec{x}_v^{(l+1)}$ do vértice v na camada $l+1$ é dado por

$$\vec{x}_v^{(l+1)} = \sigma\left(\sum_{\omega\in\Omega}\sum_{j\in\mathcal{N}_v^\omega}\frac{1}{c_{v,\omega}}\boldsymbol{W}_\omega^{(l)}\vec{x}_j^{(l)} + \boldsymbol{W}_0^{(l)}\vec{x}_v^{(l)}\right), \tag{17}$$

onde $\mathcal{N}_v^\omega$ representa os vizinhos de entrada do vértice v, conectados a v por um arco com relação $\omega \in \Omega$; $\boldsymbol{W}_\omega^{(l)}$ é a matriz de parâmetros para a relação ω na camada relacional l da rede neural; e $c_{v,\omega}$ é uma constante de normalização para modular a influência dos parâmetros do próprio vértice v na operação.

A regularização nas camadas RGCN é alcançada através da decomposição da matriz de parâmetros $\boldsymbol{W}_\omega^{(l)}$ em B transformações de base $\boldsymbol{V}$ e vetores de coeficientes $\vec{g}$,

$$\boldsymbol{W}_\omega^{(l)} = \sum_{b=1}^{B} \vec{g}_{\omega b}^{(l)} \boldsymbol{V}_b^{(l)}. \tag{18}$$

Dessa forma, todas as relações $\omega \in \Omega$ de uma mesma camada l dividem as mesmas matrizes de base, enquanto os vetores de coeficientes dependem de tanto de ω como l.

Este trabalho define o conjunto Ω como descrito na definição 13, na página 65, e utiliza camadas RGCN para que os agentes possam agregar informações de suas vizinhanças de entrada.

Essa estratégia de modelagem das relações permite a especialização da forma como cada classe de agente transforma os vetores de características de vértices de classes diferentes. Não por coincidência, essas relações são as mesmas armazenadas nos arcos dos grafos representativos dos estados do ambiente, como apresentados na figura 7.

5.2.2.2 Comunicação por mecanismo de atenção para grafos

Trabalhos contemporâneos que utilizaram redes neurais de grafos para aprendizado por reforço multi-agentes (JIANG et al., 2020; AGARWAL; KUMAR; SYCARA, 2019; WANG, Tingwu et al., 2018) se utilizaram de alguma espécie de mecanismo de atenção na arquitetura da rede neural, tipicamente baseados na arquitetura Transformer (VASWANI et al., 2017). Com o objetivo de realizar comparações com o mecanismo de comunicação relacional sem repetir trabalhos passados, este trabalho implementa um módulo de comunicação utilizando camadas de Atenção em Grafos (GAT, *Graph Attention*) (VELIČKOVIĆ et al., 2018).

Em uma camada Atenção em Grafos (GAT, *Graph Attention*), o vetor de saída do vértice v na camada $l+1$ é dado pela equação 19, na qual η_{vj} é um coeficiente de "auto-atenção" (do inglês *self-attention*), dado pela equação 20. Esse coeficiente é calculado como o resultado de uma função *softmax* aplicada à saída de um perceptron de uma única camada, o qual recebe como entrada a concatenação dos vetores $\vec{x}_v$ e $\vec{x}_j$. Os coeficientes de auto-atenção de todos os vizinhos de entrada de v são normalizados para que sua soma seja igual a 1.

Nas equações 19 e 20, M se refere ao número de "cabeças de auto-atenção" (do inglês *self-attention heads*), cada uma produzida por um perceptron de única camada distinto, potencialmente

melhorando a expressividade do mecanismo de atenção; e $\|$ é o operador de concatenação de vetores.

$$\vec{x}_v^{(l+1)} = \|_m^M \sigma\left(\sum_{j \in \mathcal{N}_v^-} \eta_{vj}^m W^m \vec{x}_j^{(l)}\right) \tag{19}$$

$$\eta_{vj} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{g}^\top [\boldsymbol{W}\vec{x}_v \,\|\, \boldsymbol{W}\vec{x}_j]\right)\right)}{\sum_{k \in \mathcal{N}(v)\cup\{v\}} \exp\left(\text{LeakyReLU}\left(\vec{g}^\top [\boldsymbol{W}\vec{x}_v \,\|\, \boldsymbol{W}\vec{x}_k]\right)\right)} \tag{20}$$

LeakyReLU (MAAS; HANNUN; NG, 2013) é uma função de ativação não-linear dada por

$$\text{LeakyRELU}(x) = \begin{cases} x & \text{se } x \geq 0 \\ 0{,}01x & \text{caso contrário} \end{cases}. \tag{21}$$

As camadas GAT não especializam o mecanismo de passagem de mensagens explicitamente da forma como convoluções relacionais o fazem.

5.2.3 Módulo de seleção de ações

Os vetores de características armazenados nos vértices do grafo relacionados a agentes, resultantes da aplicação de K camadas de convoluções em grafos, são utilizados como as observações finais dos agentes.

Neste ponto, para cada classe de agentes $c \in Z$, uma função Q_c recebe como entrada as observações finais de todos os agentes da classe c e tem como saída uma matriz de dimensões $|\mathcal{V}(\mathcal{G}, c)| \times |A_c|$, onde $\mathcal{V}(\mathcal{G}, c)$ representa os vértices em $\mathcal{G}$ da classe c e A_c, o conjunto de ações dos agentes da classe c.

Em suma, cada função Q_c é responsável por aproximar os valores estado-ação para uma classe específica de agentes.

5.2.4 Método de treinamento

O algoritmo 2 lista o processo de treinamento da rede neural. Seguindo a estratégia apresentada inicialmente por Mnih et al. (2015), duas redes neurais de topologias idênticas são utilizadas no processo de treinamento. A primeira, denominada rede de política (do inglês *policy network*) é composta pelos parâmetros $\boldsymbol{\theta}$ e treinada utilizando lotes de transições amostrados da memória de repetição. A segunda rede neural, denominada rede alvo (do inglês *target network*),

possui a mesma topologia que a rede de política, é composta pelos parâmetros $\boldsymbol{\theta}^{-}$ e é utilizada na geração de valores estáveis para a função de erro.

A rede de política é treinada de forma *off-policy*. Uma memória de repetição priorizada proporcionalmente (SCHAUL et al., 2016) armazena transições na forma $\mathcal{G}, \boldsymbol{a}, r, \mathcal{G}'$, onde $\mathcal{G}$ e $\mathcal{G}'$ representam s e s' já modelados como grafos; $\boldsymbol{a}$ é o conjunto de ações escolhidas por todos os agentes e r, a recompensa compartilhada observada pelos agentes.

A função $\hat{Q} \leftarrow \mathrm{HCANet}(\mathcal{G}_t; \boldsymbol{\theta})$ na linha 10 retorna os valores Q estimados de todas as ações para todos os agentes no grafo $\mathcal{G}_t$. Esses valores são calculados pelas múltiplas redes do módulo de seleção de ações, como exibido na figura 8, e agrupados por conveniência, de forma que $\hat{Q}(u,a)$, na linha 12 do algoritmo, indica uma busca em $\hat{Q}$ pelo valor da ação a do agente u.

Para acelerar o aprendizado da rede neural, tanto a seleção de ações dos agentes na linha 12 como a função de erro $\mathcal{L}(\boldsymbol{\theta})$ consideram apenas as ações válidas para cada agente em cada transição em ρ. As ações válidas para um agente u num estado s são definidas como o subconjunto de ações $A(u, s) \subseteq A_{C(u)}$ que u pode executar em s. Nos ambientes que disponibilizam $A(u, s)$, essa informação pode ser utilizada para que os agentes explorem apenas ações relevantes.

O conceito de ações válidas utilizado neste trabalho é semelhante operacionalmente ao conceito de ações legais descrito em Lanctot et al. (2020) para jogos que possuem regras bem definidas. O SMAC impede que agentes executem ações inválidas em um determinado estado, *e.g.* atacar um adversário fora do alcance ou executar qualquer ação que não a *no-op* após ter sido derrotado, mesmo que essas restrições não façam necessariamente parte do *StarCraft II*.

A linha 23 apresenta como os valores esperados $\boldsymbol{y}$ são calculados para um lote de transições e múltiplos agentes. A função $\mathrm{terminal}(\mathcal{G}'_b)$ retorna um vetor em $\mathbb{R}^{|b|}$ com valores 0 onde $\mathcal{G}'_b$ possui estados terminais e 1, caso contrário. O operador $\odot$ indica a multiplicação elemento-a-elemento, utilizada para cancelar os retornos provenientes do estado terminal (os quais, na prática, não são calculados).

Em sua forma mais simples, o erro a ser minimizado é o quadrado do erro de diferenças temporais entre as predições da rede de política e os valores esperados provenientes da rede alvo, apresentado na linha 24. Devido à especialização do módulo de seleção de ações da HCA-Net(EF), esse erro pode ser expresso como a soma dos erros individuais de cada rede do

Algoritmo 2 – Algoritmo de treinamento para a rede neural

Inicializa memória de repetição $D \leftarrow \{\}$
Inicializa pesos $\boldsymbol{\theta}$ da rede de política aleatoriamente
$\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$, os pesos da rede alvo
para $step = 0, \ldots, step_{max}$ **faça**
    Inicializa ambiente
    $t \leftarrow 0$
    $s_t =$ estado inicial
    enquanto $s_t \neq terminal$ **faça** *// executa um episódio*
        $\mathcal{G}_t \leftarrow$ grafo do estado s_t
        $\hat{Q} \leftarrow \text{HCANet}(\mathcal{G}_t; \boldsymbol{\theta})$ *// estimativas de Q_* para os agentes, de acordo com suas classes*
        para cada *agente* u **faça**
            $a_{u,t} \leftarrow \begin{cases} \arg\max_{a \in A(u,s)} \hat{Q}(u,a), & \text{com prob. } 1-\epsilon \\ \text{amostra aleatoriamente de } A(u,s) & \text{caso contrário} \end{cases}$
        fim
        realiza $\boldsymbol{a}_t$, observa r_{t+1}, s_{t+1}
        $\mathcal{G}_{t+1} \leftarrow$ grafo do estado s_{t+1}
        $D \leftarrow D + \{(\mathcal{G}_t, \boldsymbol{a}_t, r_{t+1}, \mathcal{G}_{t+1})\}$ *// armazena transição na memória de repetição*
        $t \leftarrow t + 1$
    fim
    se $|D| \geq$ *tamanho de um lote* **então**
        $b \leftarrow$ lote de transições amostradas de D
        $\mathcal{G}_b, \boldsymbol{a}_b, \boldsymbol{r}_b, \mathcal{G}'_b \leftarrow b$
        $\hat{\boldsymbol{Q}} \leftarrow \text{HCANet}(\mathcal{G}_b, \boldsymbol{a}_b; \boldsymbol{\theta})$
        $\boldsymbol{y} \leftarrow \boldsymbol{r}_b + \text{terminal}(\mathcal{G}'_b) \odot \gamma \max_{\boldsymbol{a}'} \text{HCANet}(\mathcal{G}'_b, \boldsymbol{a}'; \boldsymbol{\theta}^-)$
        $\mathcal{L} \leftarrow (\boldsymbol{y} - \hat{\boldsymbol{Q}})^2$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}}(\mathcal{L})$
    fim
    $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$ a cada κ passos *// atualiza parâmetros da rede alvo*
fim

módulo de seleção ações. Para uma única transição, esse erro é

$$J(\boldsymbol{\theta}) = \sum_{c \in Z} \frac{1}{|Z|} \sum_{u \in c} (y - Q_c(\vec{x}_u^{(K)}, a_u; \boldsymbol{\theta}))^2$$

$$y = \begin{cases} r + \gamma \max_{a'} Q_c(\vec{x}_u'^{(K)}, a'; \boldsymbol{\theta}^-) & \text{se } s' \neq \text{terminal} \\ r & \text{caso contrário} \end{cases}$$

onde u representa um único agente, $\vec{x}_u^{(K)}$ é o vetor de características de u após as K camadas de comunicação e $\vec{x}_u'^{(K)}$ é o vetor de características de u no estado futuro após as K camadas de comunicação. O erro para um lote é a média dos erros das transições no lote.

5.2.5 Variações

A topologia da rede e o algoritmo de treinamento permitem que a HCA-Net(EF) seja testada com algumas variações. A primeira variação, denominada Comunicação Total Entre Agentes (CTA), ignora restrições na comunicação entre os agentes impostas pelo ambiente (*e.g.* por motivos de distanciamento físico) e cria todos os arcos possíveis entre os agentes nos grafos de estados $\mathcal{G}$.

A segunda variação, proposta por Jiang et al. (2020) e implementada neste trabalho para fins de comparação, chama-se Regularização por Relações Temporais (TRR, *Temporal Relation Regularization*). A TRR é um termo extra na função de erro que tenta aproximar os pesos de cada cabeça de auto-atenção da última camada de atenção quando a camada a rede é aplicada a dois estados subsequentes, s e s'. Esse termo é composto pela divergência de Kullback-Leibler entre as duas distribuições de pesos de atenção.

Seja $\mathcal{G}_m^{(l)}(s;\boldsymbol{\theta})$ a distribuição dos pesos da cabeça de auto-atenção m da camada l quando a rede neural observa o estado s. A função de erro final utilizada para treinar a rede neural é dada por

$$\mathcal{L}(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \lambda_{\text{trr}} \frac{1}{M} \sum_{m \in M} D_{KL}(\mathcal{G}_m^{(l)}(s;\boldsymbol{\theta}) \| \mathcal{G}_m^{(l)}(s';\boldsymbol{\theta}^-)), \tag{22}$$

na qual λ_{trr} é um coeficiente usado para alterar a relevância do termo da TRR no erro da rede neural.

A última variação advém de Jiang et al. (2020) e consiste em utilizar a concatenação das saídas de todas as camadas de convolução em grafos como o vetor de características final $\vec{x}^{(K)}$ de um agente, a entrada do módulo de seleção de ações. A intuição por trás desta prática, aqui denominada de Campo Receptivo Total (CRT), está em fornecer aos agentes informação coletada de diferentes distâncias no grafo.

5.3 EXPERIMENTOS

A HCA-Net(EF) descrito foi implementado na linguagem Python, versão 3.8, utilizando a biblioteca de redes neurais *PyTorch* (PASZKE et al., 2019), versão 1.6 e a biblioteca de GNNs *PyTorch Geometric* (FEY; LENSSEN, 2019), versão 1.6.1. As contribuições realizadas a estes e outros projetos de código aberto durante a produção dessa tese são listadas no apêndice B. O código usado nos experimentos realizados nesta seção estão disponíveis em `https://github.com/douglasrizzo/hcanet/releases/tag/v8`.

O método e suas variações foram testadas no cenário `2s3z` do SMAC, o qual possui 5 agentes divididos em 2 classes, 5 unidades adversárias e um número máximo de 120 instantes de tomada de decisão, após os quais o cenário é reiniciado sem penalidade para os agentes. O cenário foi selecionado por sua baixa complexidade e pela presença de agentes heterogêneos, contendo dois agentes de uma classe e três de outra. O grafo de agentes e entidades é composto das unidades controladas pelo time do jogador como agentes e pelas unidades adversárias, controladas pelo jogo, como entidades.

Para que os estados do SMAC possam aderir à representação em forma de grafos de agentes e entidades descrita na seção 5.1, o mapeamento de estados nativos do SMAC para grafos é feita da forma descrita a seguir.

Dado um estado s_t no instante t em um cenário do SMAC, cada unidade viva no instante t se torna um vértice no grafo $\mathcal{G}_t$. Unidades que são controladas pelo código controlador de agentes se tornam tanto agentes no conjunto de agentes U como vértices correspondentes a agentes em $\mathcal{G}_t$. Unidades do time adversário se tornam vértices correspondentes a entidades em $\mathcal{G}_t$. Sua presença em $\mathcal{G}_t$ se justifica devido a suas características serem necessárias na composição das informações necessárias em um estado para que o time de agentes consiga aprender as políticas necessárias para vencer episódios.

O conjunto de classes de agentes e entidades C é definido como os tipos de unidades aos quais as próprias unidades no cenário do SMAC fazem parte. No caso do cenário `2s3z`, as classes são correspondentes aos tipos de unidade *Stalker* e *Zealot*. Caso uma unidade (aliada ou adversária) esteja dentro do campo de visão do agente u (definido como um círculo ao redor de u), um vértice $e(j, u)$ conecta o vértice j correspondente à unidade visível por u ao próprio u.

Vértices que representam agentes guardam em seus vetores as seguintes características:

a) pontos de saúde;
b) posição x relativa ao centro do mapa;
c) posição y relativa ao centro do mapa;
d) escudo (apenas unidades que os possuem);
e) tempo de espera para o próximo ataque.

Vértices que representam adversários possuem as 4 primeiras das características supracitadas. As ações dos agentes são aquelas descritas na seção 3.2.1.2. O otimizador utilizado no treinamento da rede neural foi o RMSProp (HINTON; SRIVASTAVA; SWERSKY, 2012). A tabela 8 apresenta os hiperparâmetros utilizados em todos os experimentos.

Tabela 8 – Hiperparâmetros usados no treinamento dos modelos

Passos de treinamento	10^6
Intervalo κ de atualização de $\boldsymbol{\theta}^-$	250
Taxa de aprendizado da rede	$2{,}5 * 10^{-4}$
Coeficiente de regularização L2	10^{-5}
Coeficiente da TRR	$0{,}01$
Fator de desconto γ	$0{,}99$
ϵ_{max}	$0{,}95$
ϵ_{min}	$0{,}1$
Coeficiente α da PER proporcional	$0{,}6$
Coeficiente β da PER proporcional	$0{,}4$
Fator de suavização do RMSProp	$0{,}99$
Constante de estabilização do RMSProp	10^{-5}

Em todos os testes, cada rede ϕ no módulo de codificação é um Perceptron Multi-Camadas (MLP, *Multi-Layer Perceptron*) com duas camadas escondidas de 128 neurônios e um vetor de saída de 64 valores. O módulo de comunicação é composto de 4 camadas relacionais com dimensões de saída de 64, 128, 128 e 64 valores, respectivamente, e 3 matrizes de base em cada camada. O módulo relacional foi comparado com um módulo de comunicação composto por camadas GAT (VELIČKOVIĆ et al., 2018) de mesma dimensão, com quatro cabeças de auto-atenção cujas saídas são concatenadas ao fim de cada camada. Por último, as redes Q no módulo de seleção de ações possuem duas camadas com 128 neurônios e os vetores de saída de tamanhos correspondentes ao número de ações de cada classe de agentes. Em todas as camadas aplicáveis, a função de ativação utilizada foi a sigmoide.

Foram realizados experimentos com todas as combinações de variações apresentadas na seção 5.2.5. Os experimentos foram realizados numa combinação de equipamentos físicos: um computador equipado com uma Nvidia GTX 1070 e um servidor, com uma Nvidia Tesla V100. Cada execução levou cerca de 70 horas em tempo real para concluir.

Durante o treinamento, a exploração foi realizada através de uma política ϵ-*greedy* conjunta, na qual todos os agentes selecionam suas melhores ações ou uma ação aleatória simultaneamente, no mesmo instante de tomada de decisão. Esta política sincronizada faz com que os agentes não sofram do problema de explorações alternadas (BRAFMAN; TENNENHOLTZ, 2003; MATIGNON; LAURENT; LE FORT-PIAT, 2012), no qual a convergência no aprendizado da política de um agente é afetada pela aleatoriedade das recompensas observadas, provenientes das decisões aleatórias dos demais agentes. A política conjunta também torna o algoritmo menos sensível ao decaimento de ϵ durante o treinamento (WHITESON, 2020).

Tabela 9 – Resultados da aplicação do modelo apresentado e suas variações no cenário 2s3z do SMAC

Módulo de comunicação	CRT	CTA	TRR	Número de episódios médios		Recompensa acumulada média	
				Todos os episódios	Últimos 10%	Todos os episódios	Últimos 10%
RGCN	✓	✓		77.66	82.18	**4.69**	**4.79**
RGCN	✓			78.65	83.62	3.82	3.77
RGCN				76.85	81.93	4.24	4.30
GAT	✓	✓	✓	70.63	75.10	3.85	3.86
GAT	✓		✓	77.20	82.13	3.53	3.47
GAT		✓	✓	**79.41**	**84.59**	3.97	3.99
GAT				77.21	82.29	3.98	3.99
Agente aleatório				52.155		2.222	

5.4 RESULTADOS

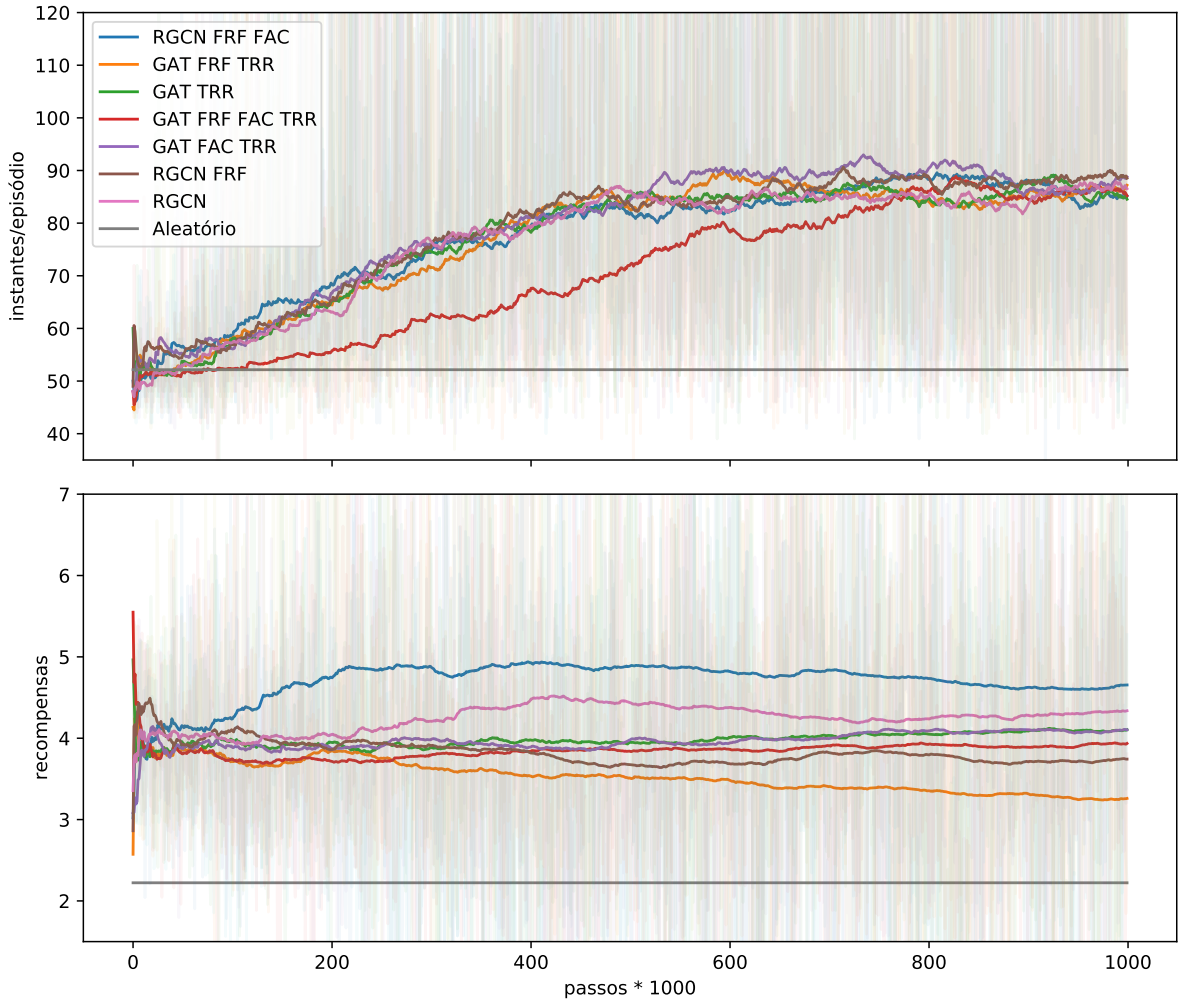
A tabela 9 exibe os resultados de todos os modelos treinados. Dois valores são utilizados como medida de desempenho dos modelos: média das recompensas acumuladas ao final dos últimos 1000 episódios e média de instantes de tomada de decisão até o time de agentes ser derrotado, também para os últimos 1000 episódios. Nos cenários do SMAC, modelos que acumulam mais recompensa foram capazes de causar mais danos e baixas ao time oponente, enquanto episódios mais longos indicam que o time de agentes foi capaz de sobreviver por mais tempo.

Quando comparados a um grupo de agentes que seleciona ações aleatoriamente, todos os modelos treinados tiveram desempenho superior em ambas as medidas. Os dois modelos que acumularem mais recompensa por episódio utilizaram módulos de comunicação relacional, já o modelo cujo time de agentes sobreviveu por mais tempo utilizou o módulo de comunicação por atenção.

Geralmente, as redes que utilizaram a comunicação total entre agentes (CTA) acumularam mais recompensa por episódio do que aquelas que não o fizeram. No entanto, não é possível observar melhor desempenho das redes que utilizaram o campo receptivo total (CRT) como entrada para as camadas de comunicação ou utilizaram a TRR em sua função de erro. Isso pode se dever ao fato dessas técnicas terem sido originalmente propostas para resolver o ambiente mais simples de predador-presa, contendo apenas agentes homogêneos (JIANG et al., 2020).

A figura 9 apresenta as mesmas medidas supracitadas ao longo de todos os passos de treinamento. Dado o ruído presente nas medidas e o longo tempo de treinamento, decidiu-se por

Figura 9 – Média de passos por episódio (topo) e recompensa nos últimos 1000 episódios de treinamento



Fonte: Autor

utilizar uma média móvel exponencial para visualizar melhor as linhas de tendência. Pode-se observar pelo gráfico de cima que todos os modelos tendem a convergir para a mesma quantidade de instantes de tempo máximo até o fim de um episódio, enquanto atingem retornos finais diferentes. Os dois modelos que utilizaram comunicação relacional foram os que acumularam mais recompensa de todos os métodos, durante todo o processo de treinamento.

5.5 DISCUSSÃO

Essa seção apresentou a HCA-Net(EF), o primeiro dos dois modelos apresentados nesta tese, o qual é composto da representação de estados em um sistema multi-agentes através de

grafos direcionados rotulados (denominados *grafos heterogêneos de agentes e entidades*) e do uso de uma rede neural especializada no aprendizado tanto de mecanismos de comunicação como de políticas especializadas para agentes heterogêneos (organizados em classes homogêneas de agentes).

Experimentos foram conduzidos para aferir a viabilidade do treinamento da rede neural, assim como comparar o método de passagem de mensagens relacional com um baseado em um mecanismo de atenção. Também foram testadas variações da HCA-Net(EF), envolvendo métodos de estabilização de treinamento disponíveis na literatura e diferentes restrições relacionadas à capacidade de comunicação entre agentes.

Os resultados desta seção demonstraram indícios de que a especialização dos canais de passagem de mensagens entre classes de agentes e entidades alcança os melhores resultados dentre os métodos testados de passagem de mensagens, quando consideradas as métricas empregadas. No entanto, o emprego dos métodos de estabilização do treinamento da rede multi-agentes disponíveis nos trabalhos relacionados (JIANG et al., 2020) e aqui re-implementados não exibiram mudanças significativas no desempenho dos modelos. O desempenho sub-ótimo desses métodos também foi observado por outros trabalhos posteriores na literatura (RAHMAN et al., 2020).

No entanto, algumas limitações, tanto do método proposto quanto do procedimento experimental apresentados nesta seção, foram observadas. As limitações referentes à HCA-Net(EF) podem ter sido a causa da incapacidade das redes neurais propostas em aprender políticas que permitissem ao time de agentes controlados pela rede neural de vencer uma partida contra o time adversário.

A primeira limitação reside na suposição de que as características de todos os agentes e entidades possam ser acessadas separadamente de forma a gerar um grafo no qual cada entidade é representada separadamente. Apesar dessa ser uma abordagem vista em outros trabalhos (AGARWAL; KUMAR; SYCARA, 2019), nem sempre essa suposição é realista, sendo mais plausível que o estado atual seja resumido apenas às observações dos agentes, de forma condizente à forma com que estados são representados em um Dec–POMDP (OLIEHOEK; AMATO, 2016).

A segunda limitação vem da forma como os agentes são guiados a tomar decisões consistentes ao longo de diferentes instantes de tempo. O único método puramente baseado em GNNs que analisou este problema foi a TRR de Jiang et al. (2020). No entanto, ela não é aplicável a camadas de convolução em grafos que não possuam pesos de atenção, como a RGCN, utilizada neste trabalho. A TRR e a DGN em geral Jiang et al. (2020) também não

obtiveram resultados promissores em outros trabalhos que a aplicaram em domínios diferentes do original (RAHMAN et al., 2020).

Com relação ao procedimento experimental, aponta-se a falta de repetições suficientes dos experimentos para se alcançar resultados com significância estatística, devido ao elevado tempo de treinamento dos modelos (cerca de 70 horas para cada repetição). Na seção 6, todas essas limitações serão abordadas, com a criação de um método que apresenta resultados superiores aos apresentados nesta seção, assim como menor tempo de treinamento (entre 8 e 16 horas em tempo real), maior número de repetições dos experimentos e o uso de métricas de avaliação de desempenho utilizadas por outros trabalhos correlatos.

Essas questões são abordadas no método apresentado na seção 6.

6 APRENDIZADO POR REFORÇO PROFUNDO EM DEC-POMDPS COM MÚLTIPLOS AGENTES HETEROGÊNEOS

Na seção 5, duas limitações principais foram levantadas com relação à HCA-Net(EF): a suposição da capacidade de se modelar entidades sem agência no grafo heterogêneo de agentes e entidades e a inexistência de um método que permitisse que o modelo tomasse decisões consistentes ao longo do tempo. A busca pela solução desses dois problemas, assim como melhorias nos procedimentos experimentais para avaliação dos modelos, são apresentadas nesta seção, cujo conteúdo foi publicado parcialmente em Meneghetti e Bianchi (2021).

Para solucionar o primeiro problema, a definição de um Dec–POMDP é estendida de forma a conter as classes de e possíveis relações entre agentes, e a definição do grafo de agentes e entidades é reformulada de forma depender de um Dec–POMDP estendido.

Definição 14 – Dec–POMDP estendido. Um Dec–POMDP estendido é uma tupla $\mathcal{M} = (U, S, A, O, P, R, H, C, \Omega)$ onde U, S, A, O, P, R, H são definidos como na definição 3 e C, Ω são:

a) C: conjunto de classes de agentes. Cada agente pertence a uma e apenas uma classe c de C;

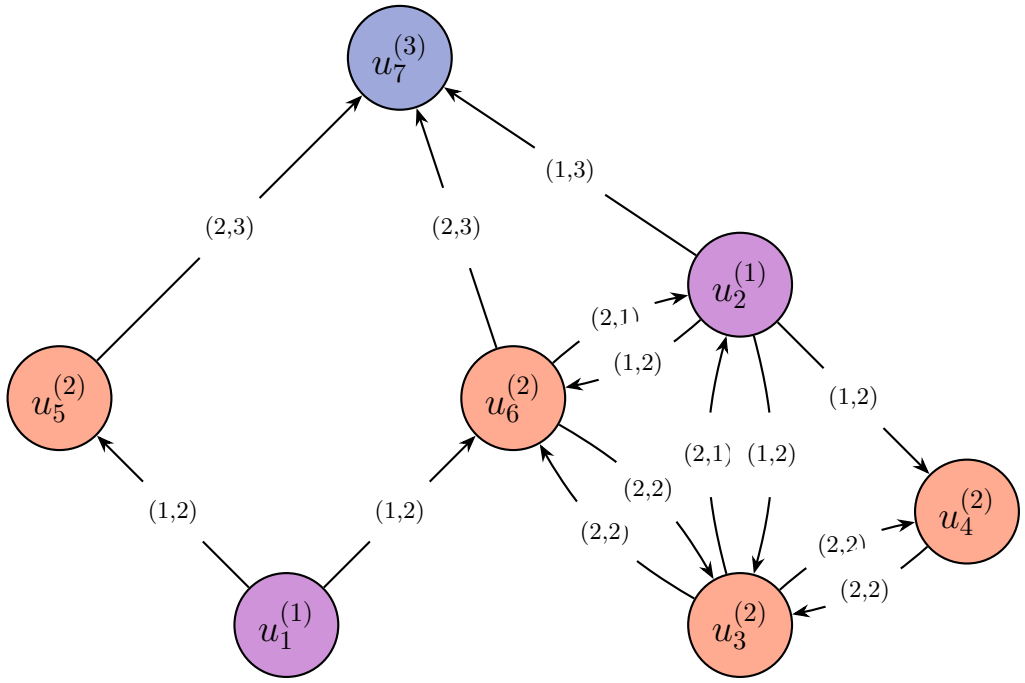
b) Ω: conjunto de possíveis relações entre pares de agentes. Dado um par de agentes relacionados (u_1, u_2) e uma função $C(u)$ que retorna a classe de um agente u, a relação entre u_1 e u_2 é dada por $(C(u_1), C(u_2))$.

Definição 15 – Grafo heterogêneo de agentes. Dado um estado s do conjunto de estados S de um Dec–POMDP estendido $\mathcal{M}$, um grafo heterogêneo de agentes é um grafo heterogêneo direcionado rotulado atribuído $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, no qual:

a) $\mathcal{V}$: conjunto de vértices, no qual cada vértice u representa um agente do conjunto de agentes U. A observação o_u de u no estado s é armazenada no vetor de características de seu vértice correspondente, v, e é representado por $\vec{x}_v$. A classe do agente representado pelo vértice v é armazenada no rótulo de v;

b) $\mathcal{E}$: conjunto de arcos direcionados. Um arco $e = (u_1, u_2)$ tem como origem um agente u_1 qualquer e destino um agente u_2, indicando a capacidade de u_1 transmitir o_{u_1} para u_2. A relação entre u_1 e u_2 é dada por $(C(u_1), C(u_2))$ e é armazenada no rótulo de e.

Figura 10 – O novo grafo de estado, representando apenas agentes. Os números superscritos e as cores dos vértices são indicativos das classes dos agentes. Números subscritos são identificadores únicos dos agentes.



Fonte: Autor

A definição do grafo heterogêneo de agentes se difere da apresentada na seção 5.1, dado que $\mathcal{V}(\mathcal{G})$ contém apenas agentes e o vetor de características armazenado em cada vértice também passa a conter a observação que o agente representado por aquele vértice tem do ambiente. Em outras palavras, o vetor de características $\vec{x}_u$ de um vértice u representa a observação o_u do próprio agente u, não mais contendo apenas as características de u.

Uma consequência dessa reformulação é a ausência de vértices representativos de entidades sem agência em $\mathcal{G}$, assim como a ausência de classes de entidades sem agência em C, fazendo com que $C = Z$, simplificando a topologia da rede neural. A figura 10 exemplifica graficamente o novo grafo de estado, onde os agentes são separados em três classes, discriminadas visualmente pelas cores dos vértices. A metodologia de criação e rotulação dos arcos do grafo se mantém, porém, o número de relações possíveis agora é $|\Omega| = |C|^2$.

Para mitigar o segundo problema, referente à garantia de que os agentes tomem decisões coerentes ao longo de instantes de tempo sucessivos, tanto a topologia da rede neural quanto o algoritmo de treinamento do modelo são modificados, de forma que a rede passe a empregar camadas recorrentes e seja treinada utilizando sequências de transições, ao invés de transições

individuais. Uma forma de fazer isso, implementada neste trabalho, é através do armazenamento de episódios na memória de repetição.

Essa estratégia foi originalmente proposta por Hausknecht e Stone (2015) para que agentes modelados como redes neurais profundas possam aprender políticas em POMDPs. A disponibilização do histórico de observações para os agentes de um Dec–POMDP também é uma estratégia válida para aumentar a crença dos agentes no estado atual (AMATO, 2015; OLIEHOEK; AMATO, 2016). Portanto, espera-se que a adição de camadas recorrentes no modelo e o treinamento com lotes de episódios atenue dois problemas existentes com a HCA-Net(EF): a incerteza sob observações parciais e a tomada de decisões coerentes ao longo de sequências de estados.

Por fim, o módulo de seleção de ações agora foi implementado utilizando DDQNs recorrentes (HAUSKNECHT; STONE, 2015) duelistas (VAN HASSELT et al., 2016; WANG, Z. et al., 2016).

Devido às características supracitadas, o modelo apresentado nesta seção foi nomeado Rede Neural de Agentes Heterogêneos e Comunicativos – Observações de Agentes (HCA-Net(AO), *Neural Network for Heterogeneous Communicative Agents – Agent Observations*), uma vez que, ao contrário da HCA-Net(EF), a HCA-Net(AO) opera sobre grafos de agentes e suas observações e não grafos de agentes, entidades e suas características.

6.1 COMPARTILHAMENTO DE PARÂMETROS

Ao se treinar a rede neural nos ambientes do SMAC, o aprendizado da HCA-Net(EF) era consideravelmente mais rápido em ambientes com uma classe de agentes (*e.g.* o mapa `3m`, que possui 3 agentes), comparado aos ambientes com mais de uma classe de agentes. Anteriormente, o vetor de características $\vec{x}_v$ de um vértice v era composto por 5 a 6 valores. No entanto, as observações dos agentes, utilizadas na segunda parte do trabalho, passaram a possuir a partir de 90 valores (com dimensionalidade dependente do número de agentes e adversários no ambiente).

O uso de redes neurais de codificação e seleção de ações especializadas em classes específicas de agentes multiplica o número de parâmetros a serem aprendidos em tempo de treinamento, também dividindo os dados de treinamento entre as redes especializadas, resultando na lentidão de aprendizado observada, a qual não poderia mais ser ignorada.

Para se solucionar o problema, optou-se por realizar o compartilhamento de parâmetros das camadas de codificação e seleção de ações da rede neural entre todas as classes de agentes.

Para viabilizar tanto o compartilhamento de parâmetros dos módulos de codificação e seleção de ações da rede neural utilizando agentes heterogêneos, a seguinte notação é introduzida. Para denotar o conjunto de variáveis utilizadas para representar as observações de agentes da classe c, utilizaremos Ψ_c. Analogamente, o conjunto de ações de uma classe de agentes c é denotado por A_c.

Essas duas definições permitem a composição dos conjuntos de variáveis de observação conjuntas $\Psi = \bigcup_{c \in Z} \Psi_c$ e de ações conjuntas de todas as classes de agentes $A = \bigcup_{c \in Z} A_c$.

Dessa forma, uma rede neural capaz de compartilhar parâmetros entre os todas as classes de agentes tem como entrada Ψ e, como saída A. O compartilhamento de parâmetros na camada de codificação ocorre quando, para qualquer par de classes de agentes $\{c_1, c_2\}$, $\Psi_{c_1} \cap \Psi_{c_2} \neq \varnothing$. De forma similar, quando $A_{c_1} \cap A_{c_2} \neq \varnothing$, c_1 e c_2 compartilham parâmetros no módulo de seleção de ações.

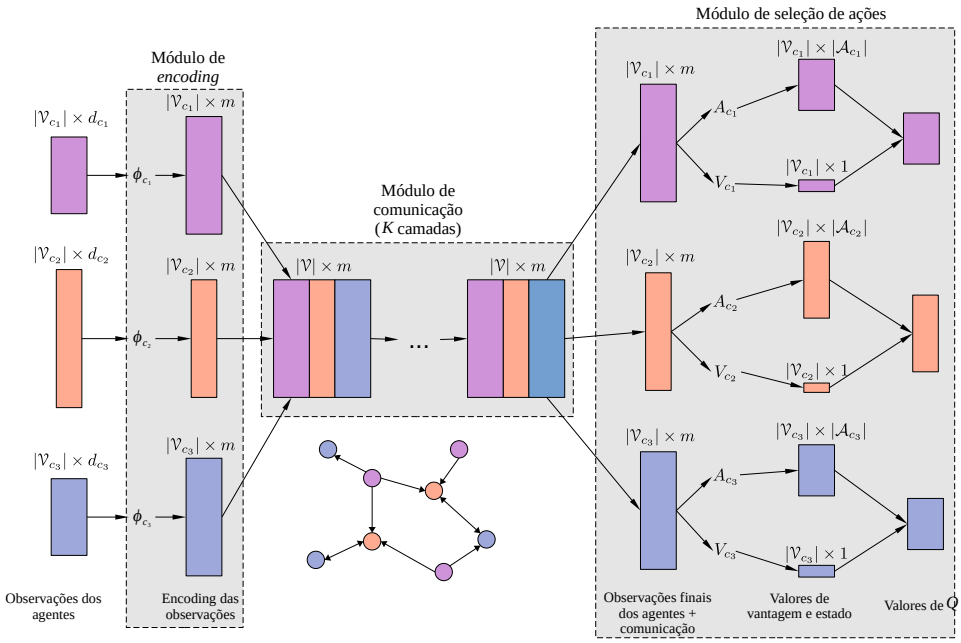
Para se implementar o compartilhamento de parâmetros no módulo de codificação, as variáveis de observação não utilizadas por agentes de uma determinada classe são preenchidas com zeros. Já no módulo de seleção de ações, os valores Q calculados para ações de agentes de uma classe c fora do conjunto A_c são ignorados tanto na seleção de ações como no cálculo do erro de diferenças temporais.

O compartilhamento de parâmetros em redes neurais aplicadas a sistemas multi-agentes não é uma técnica nova. Ele foi utilizado em sistemas com agentes homogêneos em Sunehag et al. (2018), Rashid et al. (2018), Agarwal, Kumar e Sycara (2019) e Jiang et al. (2020) e analisado teoricamente em ambientes com agentes homogêneos e heterogêneos em Terry et al. (2020), um trabalho contemporâneo a este, porém sem resultados experimentais para agentes heterogêneos.

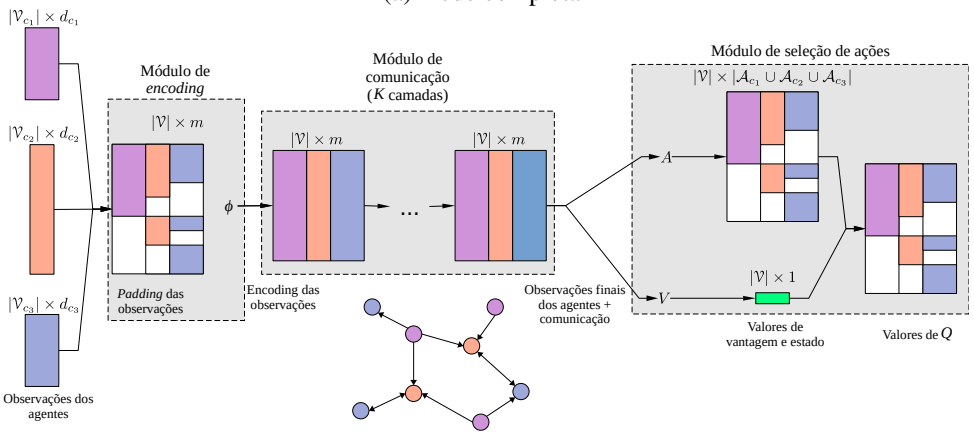
A figura 11 apresenta a nova topologia de rede neural. Em 11(a), é demonstrada a rede sem compartilhamento de parâmetros, especializando as camadas de codificação e seleção de ações para três classes de agentes distintas. Seguindo a estratégia da DDQN, a saída da camada de comunicação agora é utilizada na estimação de $\mathcal{A}$ e V. Já a figura 11(b) apresenta uma topologia análoga de rede neural com compartilhamento de parâmetros.

O número de parâmetros treináveis da HCA-Net(AO) depende tanto do uso de compartilhamento de parâmetros nos módulos de codificação ou seleção de ações, como do número de classes de agentes presentes no ambiente. A seção 6.3.2 analisa o número de parâmetros do modelo no ambiente utilizado nos experimentos, assim como das variações comparativas.

Figura 11 – Topologia da rede neural proposta para processamento de grafos heterogêneos de agentes



(a) Rede completa



(b) Rede com compartilhamento de parâmetros entre classes de agentes

Fonte: Autor

6.2 MÉTODO DE TREINAMENTO

O algoritmo 3 demonstra o procedimento de treinamento da rede neural após as alterações descritas ao longo da seção 6. O laço de repetição iniciado na linha 10 executa um episódio de interação com o ambiente. O episódio é então armazenado na memória de repetição de episódios (linha 20). Um lote de episódios é então amostrado da memória, os quais têm seus tamanhos normalizados através da inserção de transições nulas nos episódios menores. A rede neural é

treinada, visualizando as transições de todos os episódios do lote de treinamento em ordem sequencial.

Algoritmo 3 – Algoritmo de treinamento episódico para a rede neural

1 Inicializa memória de repetição episódica $D \leftarrow \{\}$
2 Inicializa pesos $\boldsymbol{\theta}$ da rede de política aleatoriamente
3 $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$, os pesos da rede alvo
4 **para** $step = 0, \ldots, step_{max}$ **faça**
5 Inicializa ambiente
6 $t \leftarrow 0$
7 $s_t =$ estado inicial
8 Inicializa histórico do episódio atual $\rho = \{\}$
9 Reinicia estado das camadas LSTM de $\boldsymbol{\theta}$ para 1 sequência
10 **enquanto** $s_t \neq terminal$ **faça** *// executa um episódio*
11 $\mathcal{G}_t \leftarrow$ grafo do estado s_t
12 $\hat{Q} \leftarrow \mathrm{HCANet}(\mathcal{G}_t, \boldsymbol{a}_t; \boldsymbol{\theta})$, estimativas de Q_* para os agentes, de acordo com suas classes
13 **para cada** *agente* u **faça**
14 $a_{u,t} \leftarrow \begin{cases} \arg\max_{a \in A(u,s)} \hat{Q}(u,a), & \text{com prob. } 1-\epsilon \\ \text{amostra aleatoriamente de } A(u,s) & \text{caso contrário} \end{cases}$
15 **fim**
16 realiza $\boldsymbol{a}_t$, observa r_t, s_{t+1}
17 $\rho \leftarrow \rho + \{(\mathcal{G}_t, \boldsymbol{a}_t, r_t, terminal(s_{t+1}))\}$
18 $t \leftarrow t + 1$
19 **fim**
20 $D \leftarrow D + \{\rho\}$ *// armazena episódio na memória de repetição*
21 **se** $|D| \geq$ *tamanho de um lote* **então**
22 $b \leftarrow$ lote de episódios amostrados uniformemente de D
23 $\mathcal{G}_b, \boldsymbol{a}_b, \boldsymbol{r}_b \leftarrow b$ *// grafos de estado, ações tomadas e recompensas observadas por todos os agentes em todos os estados de todos os episódios em* b
24 $N \leftarrow$ soma do número de transições de todos os episódios em b
25 $T \leftarrow$ tamanho do episódio mais longo em b
26 realiza *padding* dos episódios em b para que tenham tamanho T
27 Reinicia estado das camadas LSTM de $\boldsymbol{\theta}$ e $\boldsymbol{\theta}^-$ para $|b|$ sequências
28 $\hat{\boldsymbol{Q}} \leftarrow \mathrm{HCANet}(\mathcal{G}_{b,1:T-1}, \boldsymbol{a}_b; \boldsymbol{\theta})$
29 $\boldsymbol{y} \leftarrow \boldsymbol{r}_b + \mathrm{terminal}(\mathcal{G}_{b,2:T}) \odot \gamma \mathrm{HCANet}(\mathcal{G}_{b,2:T}, \arg\max_{\boldsymbol{a}'} \mathrm{HCANet}(\mathcal{G}_{b,2:T}, \boldsymbol{a}'; \boldsymbol{\theta}); \boldsymbol{\theta}^-)$
30 $\mathcal{L} \leftarrow \sum (\boldsymbol{y} - \hat{\boldsymbol{Q}})^2 / N$
31 $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}}(\mathcal{L})$
32 **fim**
33 $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$ a cada κ passos *// atualiza parâmetros da rede alvo*
34 **fim**

Devido à execução do treinamento da rede neural utilizando episódios e à presença de camadas recorrentes na topologia, um truque de implementação é apresentado na linha 26,

permitindo que a rede neural consiga processar um lote de episódios, quando estes possuem tamanhos variados. Realiza-se o *padding* dos episódios menores que o maior episódio do lote, utilizando transições sintéticas que não incorram em mudanças no processo de treinamento da rede. Essas transições sintéticas são implementadas usando observações como vetores zerados, ações *no-op* e garantindo que o erro de diferenças temporais envolvendo essas transições falsas seja sempre zero. Ao término do processamento de ρ, a atualização dos pesos utiliza a *Retropropagação através do Tempo* (BPTT, *Backpropagation Through Time*) (WERBOS, 1988; WILLIAMS, 1992) nas camadas recorrentes.

O valor $\hat{\boldsymbol{Q}}$ apresentado na linha 28 é uma matriz tridimensional com os valores Q das ações selecionadas pelo grupo de agentes, em todos os instantes de todos os episódios de b. Analogamente, $\boldsymbol{y}$ é uma matriz de mesma dimensão contendo os valores futuros ótimos estimados de acordo com o método DDQN, uma adaptação da 7 (VAN HASSELT; GUEZ; SILVER, 2016) para o caso de múltiplos agentes, estados e episódios.

Analogamente ao algoritmo 2, o termo $\mathrm{terminal}(\mathcal{G}_{b,2:T})$ representa uma matriz com valores 0 onde $\mathcal{G}_{b,2:T}$ possui estados terminais e 1, caso contrário. O operador $\odot$ indica a multiplicação elemento-a-elemento, utilizada para cancelar os retornos provenientes do estado terminal (os quais, na prática, não são calculados).

O erro, apresentado na linha 30, é a média do quadrado das diferenças temporais para todos os agentes, em todos os estados e episódios de b. A quantidade N corrige o cálculo da média, ignorando as transições sintéticas adicionadas a b na linha 26.

Assim como descrito na seção 5.2.4, apenas as ações válidas em cada estado são consideradas para exploração. É importante reparar que, no caso em que uma mesma rede neural tem como saída as ações de todas as classes de agentes (como ao compartilhar o módulo de seleção de ações entre todas as classes de agentes), selecionar as ações de cada agente u usando o subconjunto $A(u, s)$ age como um filtro para que u não escolha ações não pertencentes a $C(u)$.

6.3 EXPERIMENTOS

Os experimentos para a nova versão da rede neural foram realizados em uma coleção mais extensa de cenários do SMAC. Os cenários escolhidos foram `3m`, `3s5z`, `1c3s5z`, `MMM` e `MMM2`. Os cenários foram selecionados de forma a avaliar o modelo sob diferentes circunstâncias: trabalhando com apenas uma classe de agentes (`3m`), com um número crescente de classes de agentes (`3s5z` e `1c3s5z`) e cenários com quantidades equivalentes de unidades aliadas e

Tabela 10 – Características dos mapas do SMAC utilizados nos experimentos. Os valores entre parênteses indicam o total de unidades em cada uma das classes

Mapa	Nº de classes de agentes	Nº de agentes	Nº de adversários	Nº máximo de passos
3m	1	3 (3)	3 (3)	120
3s5z	2	8 (3/5)	8 (3/5)	150
1c3s5z	3	9 (1/3/5)	9 (1/3/5)	180
MMM	3	10 (1/2/7)	10 (1/2/7)	180
MMM2	3	10 (1/2/7)	12 (1/3/8)	180

Fonte: Autor

adversárias (MMM), em contraste a um cenário no qual as unidades aliadas possuem desvantagem em números (MMM2). A tabela 10 apresenta as principais características dos mapas.

Como previsto a definição de um Dec–POMDP, todos os agentes observam uma mesma recompensa conjunta a cada instante de tomada de decisão. O SMAC foi configurado para retornar uma recompensa composta da soma dos seguintes valores:

a) 200, caso os agentes vençam o episódio;

b) 10 caso uma unidade adversária seja derrotada;

c) $(\text{pontos de saúde dos adversários em } t-1) - (\text{pontos de saúde dos adversários em } t)$.

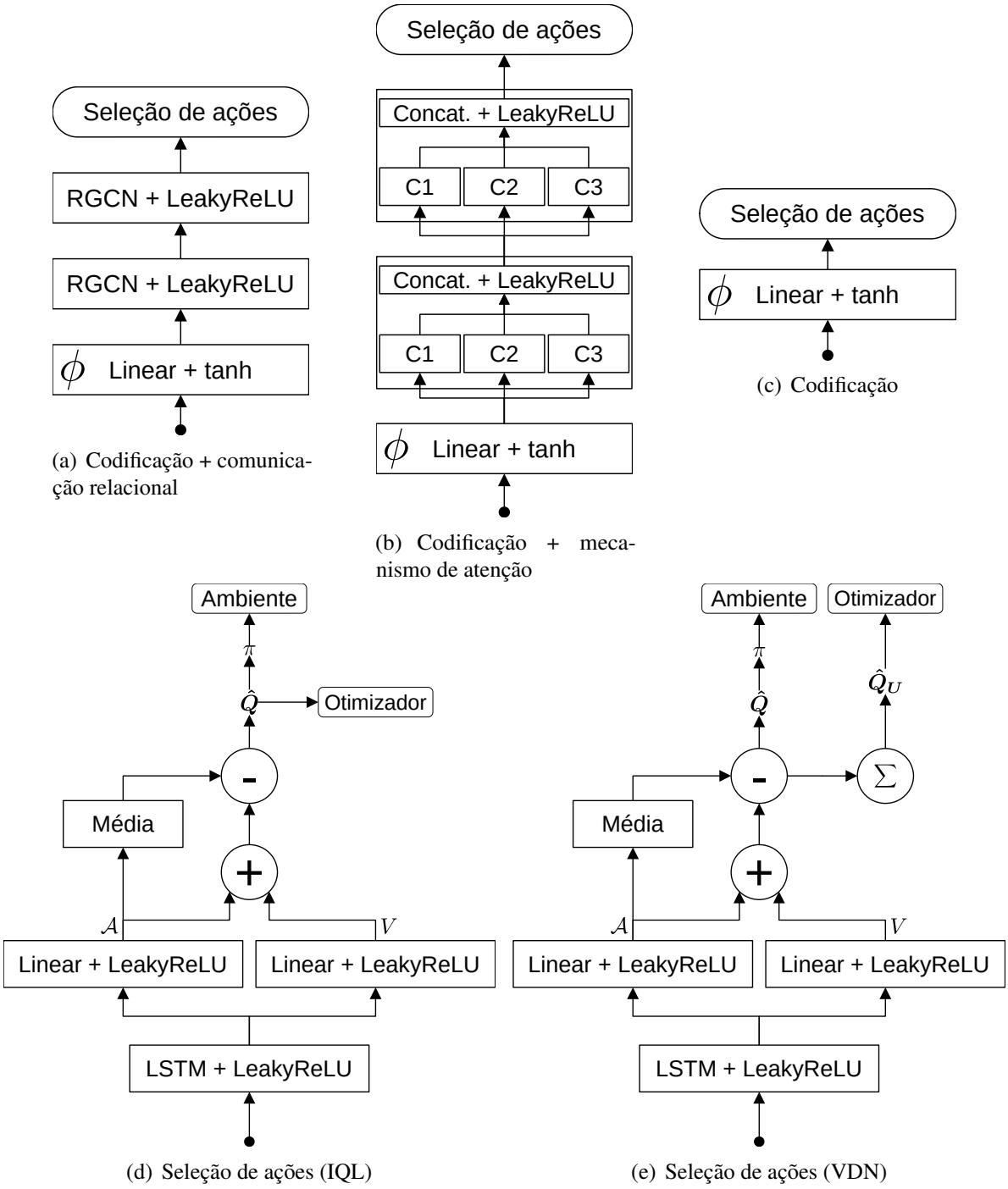
6.3.1 Topologia específica da rede neural

Para realização dos experimentos, uma topologia específica do modelo apresentado na figura 11 é apresentada nesta seção. A figura 12 apresenta os módulos utilizados nos experimentos. A rede de codificação ϕ é um perceptron de única camada, com 96 neurônios. Ela precede cada um dos módulos de comunicação nas figuras 12(a), 12(b) e 12(c).

Assim como na seção 5.2, três variações do módulo de comunicação foram testadas. A primeira variação utiliza o grafo heterogêneo de agentes proposto e duas camadas RGCN (SCHLICHTKRULL et al., 2018) com duas matrizes de base, entradas e saídas de tamanho 96 (figura 12(a)).

A segunda variação emprega duas camadas de atenção para grafos do tipo GAT (VELIČKOVIĆ et al., 2018), também com entradas e saídas de tamanho 96 e três cabeças de auto-atenção (figura 12(b)). No entanto, diferentemente do vetor de características utilizado na seção 5.2, nos novos experimentos os agentes possuem em suas observações seu tipo de unidade, na forma de

Figura 12 – Os módulos de rede neural utilizados nos experimentos



(a) Codificação + comunicação relacional

(b) Codificação + mecanismo de atenção

(c) Codificação

(d) Seleção de ações (IQL)

(e) Seleção de ações (VDN)

Fonte: Autor

um vetor *one-hot*, possibilitando que as camadas GAT utilizem essa informação na especialização da comunicação para diferentes tipos de unidade.

A última variação do módulo de comunicação é a ausência total de passagem de mensagens entre agentes, conectando a saída do módulo de codificação ao módulo de seleção de ações (figura 12(c)).

O módulo de seleção de ações é composto por uma DDQN duelista recorrente. A recorrência é implementada através de uma camada LSTM (HOCHREITER; SCHMIDHUBER, 1997) com vetor escondido de tamanho 64. As camadas que aproximam a função vantagem $\mathcal{A}$ e a função valor-estado V são totalmente conectadas, com 64 neurônios. Essa configuração pode ser observada nas figuras 12(d) e 12(e).

Para determinar a ortogonalidade de técnicas de *mixing* (as quais já foram utilizadas em outros trabalhos no ambiente do SMAC) com os métodos de comunicação implementados, dois métodos de aproximação das funções valor-ação dos agentes foram incluídos nos experimentos. O primeiro método aproxima os valores Q de cada agente individualmente, análogo ao processamento de um lote de observações independentes. Este método é equivalente ao -*Learning* Independente (IQL, *Independent -Learning*) (TAN, 1993) e é apresentado na figura 12(d).

O segundo método é a VDN (SUNEHAG et al., 2018), uma das técnicas de *mixing* apresentadas na seção 2.2.3. Neste método, a rede neural aproxima os valores Q dos agentes indiretamente ao aproximar a soma deles (cf. equação 8). A topologia da rede neural que implementa a VDN é exibida na figura 12(e).

A função de ativação utilizada no módulo de codificação foi a tangente hiperbólica,

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}, \tag{23}$$

enquanto a função utilizada nos módulos de comunicação e seleção de ação foi a Leaky-ReLU (MAAS; HANNUN; NG, 2013), apresentada na página 71 e repetida aqui,

$$\text{LeakyRELU}(x) = \begin{cases} x & \text{se } x \geq 0 \\ 0{,}01x & \text{caso contrário} \end{cases}. \tag{24}$$

A escolha da LeakyReLU se justifica por ela preservar os gradientes da rede neural, mitigando o problema do desaparecimento dos gradientes (HOCHREITER et al., 2001) que impede o treinamento de redes neurais profundas, enquanto a tangente hiperbólica nas primeiras camadas da rede auxilia no aprendizado de representações para os dados de entrada, sendo treinada usando gradientes de múltiplas magnitudes tanto para saídas positivas como negativas das suas respectivas camadas.

Ao comparar a HCA-Net(AO) com variações não-relacionais e não-comunicativas, este trabalho visa isolar as contribuições da comunicação relacional especializada entre classes de agentes no desempenho do time de agentes. Como explicado na seção 4.2, o desempenho de outros trabalhos aplicados ao SMAC pode ser maximizado através de buscas por hiperparâmetros e topologia da rede neural, além de truques de implementação (HU, J. et al., 2021). Essa tese visa isolar este fator ao propor múltiplos modelos com topologias equiparáveis e representações de estados idênticas e algoritmos de treinamento idênticos.

6.3.2 Número de parâmetros treináveis

A tabela 11 apresenta o número de parâmetros treináveis tanto da HCA-Net(AO) como das variações utilizadas nos experimentos, sob diferentes configurações de compartilhamento de parâmetros (apenas no módulo de codificação, seleção de ações ou ambos), nos diferentes cenários do SMAC usados nos experimentos tanto desta seção, assim como no cenário `2s3z` usado na seção 5.3). Os experimentos nesta seção foram efetuados com compartilhamento de parâmetros em ambos os módulos (codificação e seleção de ações) de todos os modelos.

Comparando as três últimas colunas entre si, percebe-se que o compartilhamento do módulo de seleção de ações entre classes de agentes diminui mais a rede do que o compartilhamento do módulo de codificação, pelo fato deste último possuir menos parâmetros. Esses fatores são, obviamente, condicionados à topologia escolhida para realização dos experimentos em primeiro lugar e não são universais.

Em todos os casos, a última coluna é uma soma das duas anteriores, comprovando que o compartilhamento de parâmetros nos módulos de codificação e seleção de ações são independentes e possuem efeito aditivo na diminuição da rede neural.

Ao comparar o número de parâmetros das redes neurais nos mapas que possuem mesmo número de classes de agentes (`2s3z/3s5z` e `MMM/1c3s5z/MMM2`), percebe-se que o total de parâmetros da rede neural é uma função não só do número de classes de agentes, mas também do número de agentes em si, assim como do número de adversários presentes no cenário. Isso se dá pelo fato do número de ações (que ditam o total de saídas da rede) ser condicionado ao número de agentes e adversários pela presença de ações paramétricas (HAUSKNECHT; STONE, 2016) no SMAC.

Por último, comparando os valores das três últimas colunas entre os mapas, percebe-se que o compartilhamento de parâmetros de fato diminui mais o número de parâmetros em cenários

Tabela 11 – Número de parâmetros treináveis das redes neurais sem compartilhamento de parâmetros (Base) e com compartilhamento nos módulos de codificação (C), seleção de ações (A) e em ambos os módulos (C/A).

Mapa	Classes/ Agentes/ Adversários	Nº de parâmetros				Redução (%)		
		Base	C	A	C/A	C	A	C/A
Sem comunicação								
3m	1/3/3	54698	54698	54698	54698	0.00	0.00	0.00
2s3z	2/5/5	119640	102072	77388	59820	14.68	35.32	50.00
3s5z	2/8/8	129822	107358	87375	64911	17.30	32.70	50.00
MMM	3/10/10	204915	153360	119761	68305	25.16	41.56	66.67
1c3s5z	3/9/9	205008	153459	119984	68336	25.14	41.47	66.67
MMM2	3/10/12	209913	155385	124499	69971	25.98	40.69	66.67
RGCN								
3m	1/3/3	110190	110190	110190	110190	0.00	0.00	0.00
2s3z	2/5/5	175144	157576	132892	115324	10.03	24.12	34.15
3s5z	2/8/8	185326	162862	142879	120415	12.12	22.90	35.03
MMM	3/10/10	260439	208884	175285	123829	19.80	32.70	52.45
1c3s5z	3/9/9	260532	208983	175508	123860	19.79	32.63	52.46
MMM2	3/10/12	265437	210909	180023	125495	20.54	32.18	52.72
GAT								
3m	1/3/3	216170	216170	216170	216170	0.00	0.00	0.00
2s3z	2/5/5	330264	312696	238860	221292	5.32	27.68	33.00
3s5z	2/8/8	340446	317982	248847	226383	6.60	26.91	33.50
MMM	3/10/10	464691	413136	281233	229777	11.09	39.48	50.55
1c3s5z	3/9/9	464784	413235	281456	229808	11.09	39.44	50.56
MMM2	3/10/12	469689	415161	285971	231443	11.61	39.11	50.72

Fonte: Autor

com mais classes de agentes. Esse fator indica que, caso compartilhamento de parâmetros não seja implementado em cenários com agentes heterogêneos, o número de parâmetros da rede neural tende a crescer linearmente com o número de classes de agentes, podendo vir a tornar a aplicação dessas arquiteturas de rede neural inviável em cenários com muitas classes de agentes.

6.3.3 Metodologia experimental e hiperparâmetros de treinamento

Pelos mesmos motivos mencionados na seção 5.3, os agentes utilizam uma política ϵ-*greedy* conjunta, na qual, em cada instante de tomada de decisão, todos selecionam suas melhores ações com probabilidade $1 - \epsilon$ ou selecionam uma ação aleatória com probabilidade ϵ.

Tabela 12 – Hiperparâmetros usados no treinamento dos modelos. Quando procedidos por †, seus valores foram replicados de Samvelyan et al. (2019).

Passos de treinamento	10^6
Intervalo κ de atualização de $\boldsymbol{\theta}^-$	250
Taxa de aprendizado da rede†	$5 * 10^{-4}$
Coeficiente de regularização L2	10^{-5}
Fator de desconto $\gamma^\dagger$	$0{,}99$
ϵ_{max}	$0{,}95$
ϵ_{min}	$0{,}1$
Número de passos para decaimento linear de $\epsilon^\dagger$	50 000

Fonte: Autor

Uma estratégia de decaimento linear para ϵ foi implementada, cujos valores relevantes, assim como outros hiperparâmetros de treinamento utilizados nos experimentos, são exibidos na tabela 12.

Os valores de alguns hiperparâmetros cruciais para a execução bem-sucedida dos experimentos foram replicados de Samvelyan et al. (2019), uma vez que os autores foram responsáveis por propor o ambiente do SMAC e executar experimentos bem-sucedidos nele.

Seguindo recomendações da literatura (RASHID et al., 2020a), cada variação da HCA-Net(AO) é treinada 5 vezes em cada um dos mapas selecionados, por uma duração de 1 milhão de passos de treinamento. A cada dez mil passos de treinamento, o modelo passa por uma etapa de avaliação, na qual ele segue uma política gulosa por 32 episódios. Dado que cada modelo é treinado por 1 milhão de episódios, um total de 100 etapas de avaliação são conduzidas.

As seguintes métricas são capturadas durante as avaliações: taxa de vitórias (qual fração dos 32 episódios o time de agentes venceu), número médio de unidades adversárias derrotadas, recompensa acumulada e número médio de unidades aliadas derrotadas pelo adversário.

a) taxa de vitórias (qual fração dos 32 episódios o time de agentes venceu);
b) número médio de unidades adversárias derrotadas;
c) recompensa acumulada;
d) número médio de unidades aliadas derrotadas pelo adversário.

As 3 primeiras métricas influenciam diretamente na recompensa do agente, sendo a terceira uma substituta para a segunda (caso um modelo não derrote nenhuma unidade) e a segunda, uma substituta para a primeira (caso um modelo não vença episódios, mas derrote adversários).

Nos gráficos de linha, o menor e o maior valor de cada métrica em cada etapa de avaliação é descartado (*i.e.* medidas referentes a dois dos cinco modelos treinados em um mesmo mapa), reportando-se a mediana e os valores nos percentis de 25% e 75%. Nos testes de significância estatística, nenhum valor é ignorado, mas apenas os valores coletados nas 5 últimas etapas de avaliação são utilizados nos testes.

Um subconjunto dos modelos também foi treinado por 6 milhões de passos no mapa `3s5z` para se investigar os efeitos do treinamento por tempo prolongado. Cada intervalo de um milhão de passos levou entre 5 e 11 horas em um computador equipado com uma placa de vídeo Nvidia GTX 1070.

6.4 RESULTADOS

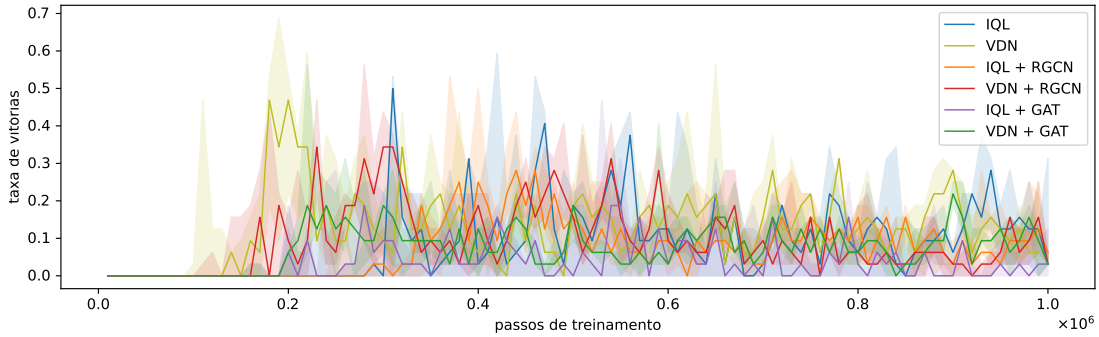
A figura 13 apresenta os resultados dos experimentos no mapa `3m`. Todas as variações convergiram para porcentagens de vitória abaixo de 30% (fig. 13(a)), conseguindo derrotar 2 dos 3 adversários no mapa (fig. 13(b)) e, consequentemente, acumulando a mesma recompensa ao fim dos episódios (fig. 13(c)) e perdendo todas as 3 unidades no próprio time (fig. 13(d)). A equivalência nos valores de todas as métricas para todos os métodos neste mapa (equivalência essa confirmada em testes estatísticos na página 101, a qual não ocorre nos outros mapas) pode ser explicada pela homogeneidade dos agentes, os quais pertencem todos de uma mesma classe.

Mesmo assim, as variações que se utilizam de *mixing* ou de comunicação relacional demonstram menor variação em seus resultados que aquelas que não utilizam *mixing* e não se comunicam, ou se comunicam por camadas GAT, características que afetaram negativamente seus resultados (figs. 13(b) e 13(c)).

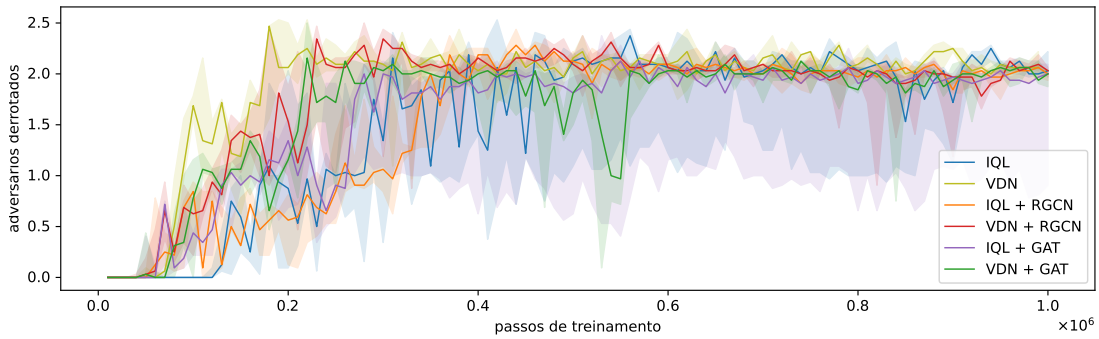
A figura 14 apresenta os resultados dos experimentos no mapa `3s5z`. Neste cenário, nenhuma das variações da HCA-Net(AO) foi capaz de aprender a vencer partidas (fig. 14(a)), conseguindo derrotar entre 1,5 e 3 unidades adversárias, de um total de 8 (fig. 14(b)). Os métodos que conseguiram acumular mais recompensa empregaram apenas *mixing* ou *mixing* com comunicação relacional (fig. 14(c)) e os métodos que empregaram comunicação por camada GAT demonstraram o pior desempenho em termos de recompensa e unidades aliadas perdidas (figs. 14(c) e 14(d)). A incapacidade dos métodos de aprenderem a vencer partidas neste mapa é revisitada em experimentos posteriores.

A figura 15 apresenta os resultados dos experimentos no mapa `1c3s5z`. Neste mapa, o qual possui 3 classes de agentes, a HCA-Net(AO) que emprega *mixing* e comunicação relacional

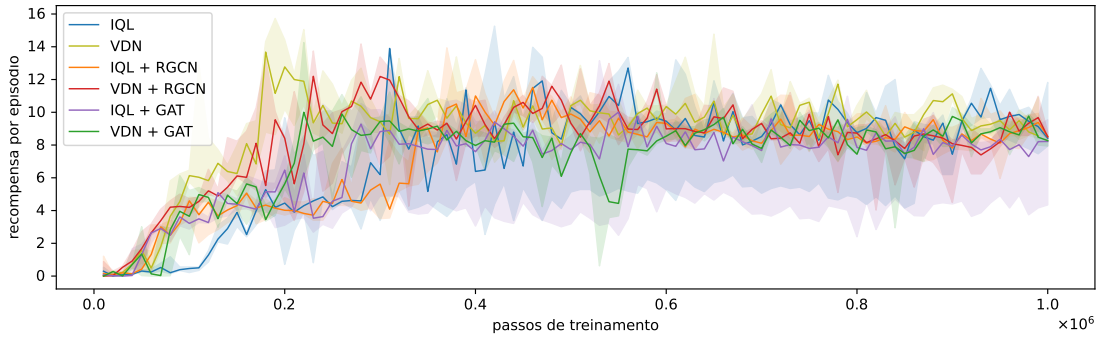
Figura 13 – Resultados das redes neurais no mapa `3m` durante 1 milhão de passos



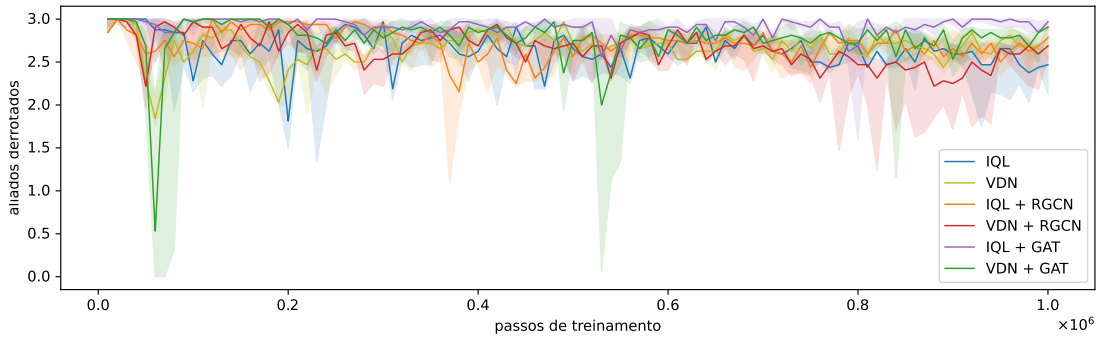
(a) Batalhas vencidas



(b) Adversários derrotados (máximo: 3)



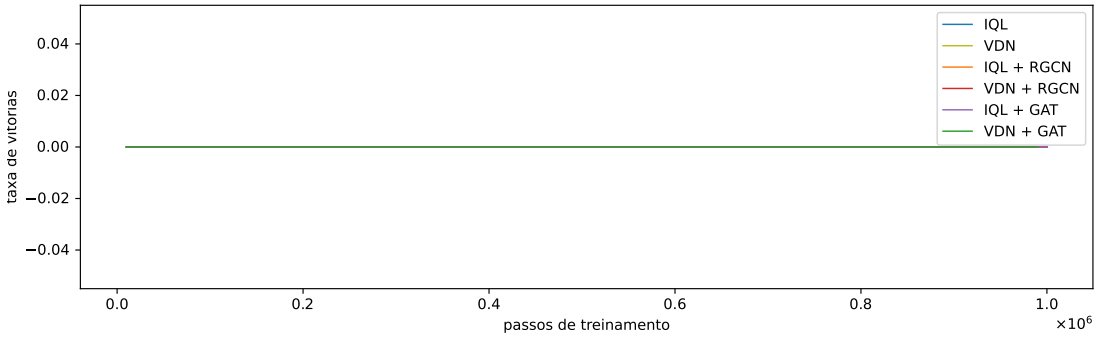
(c) Recompensa por episódio



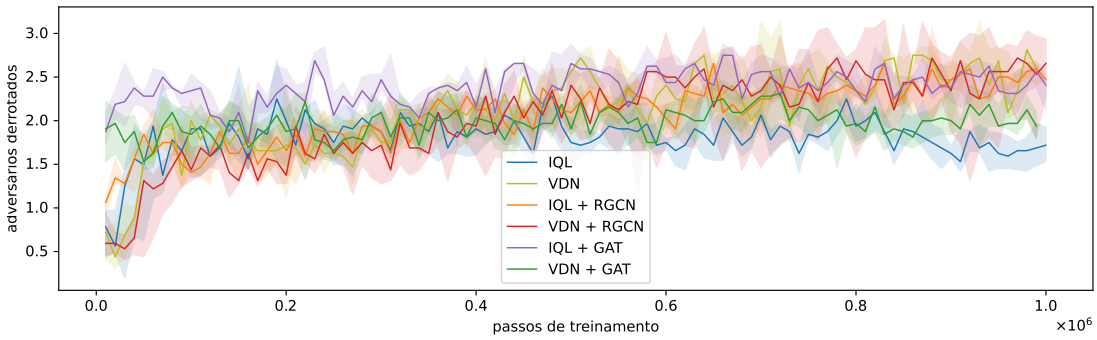
(d) Unidades aliadas perdidas

Fonte: Autor

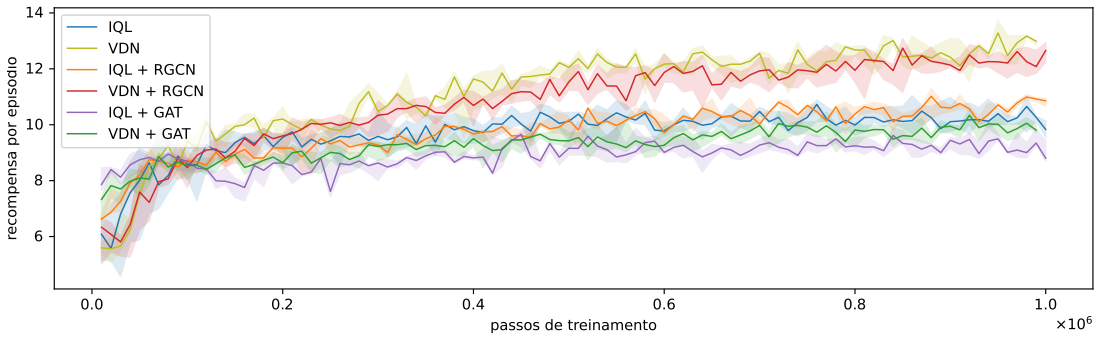
Figura 14 – Resultados das redes neurais no mapa 3s5z durante 1 milhão de passos



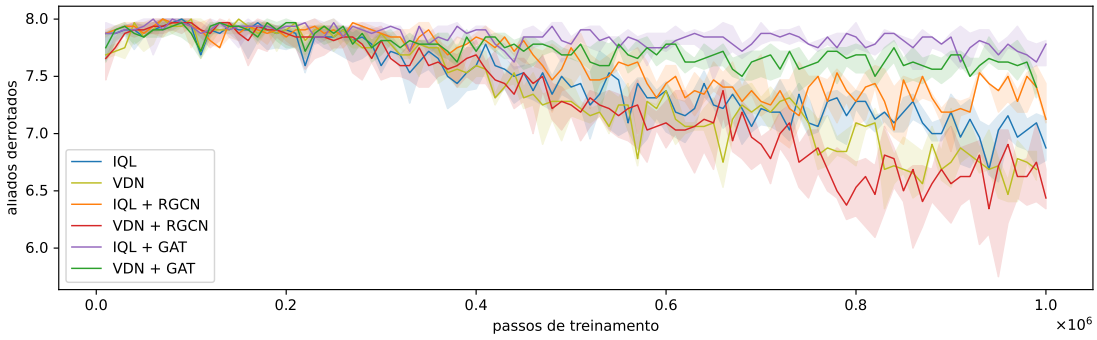
(a) Batalhas vencidas



(b) Adversários derrotados (máximo: 8)



(c) Recompensa por episódio



(d) Unidades aliadas perdidas

Fonte: Autor

alcança o melhor desempenho em todas as métricas, seguida da variação que emprega apenas *mixing*.

A figura 16 apresenta os resultados dos experimentos no mapa `MMM`. Neste mapa, não é possível visualizar diferença de desempenho entre as redes através dos gráficos. Portanto, a avaliação será realizada com os testes estatísticos na página 101.

A figura 17 apresenta os resultados dos experimentos no mapa `MMM2`. Neste mapa, considerado difícil pelos criadores do SMAC (SAMVELYAN et al., 2019), nenhum modelo foi capaz de aprender a vencer partidas. Todos os modelos aprenderam a derrotar mais de 2 adversários (de um total de 12, fig. 16(c)). Alguns modelos demonstraram a capacidade de sobreviver por mais tempo, perdendo menos unidades até o número máximo de instantes de tomada de decisão que reinicia o ambiente. Isso é evidente na figura 17(d), na qual, apesar de não vencer partidas, alguns modelos tiveram menor perda de unidades aliadas que o máximo (10), indicando serem capazes de sobreviver até o final forçado do episódio.

Para avaliar a significância estatística nas diferenças das curvas apresentadas nas figuras 13 a 17, realizou-se a Análise de Variância (ANOVA, *Analysis Of Variance*) unilateral das métricas.

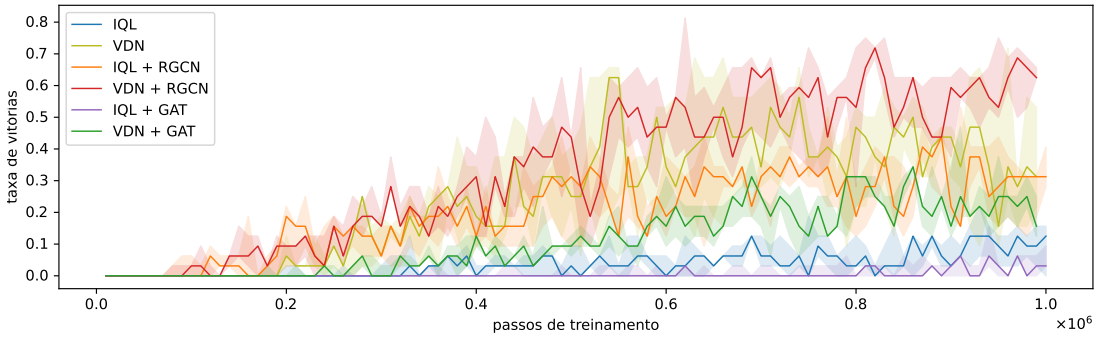
A ANOVA foi considerada adequada para analisar os experimentos pois ela generaliza o teste t de Student na tarefa de aferir se duas ou mais médias de distribuições são distintas. Visto que os experimentos possuem resultados para 6 redes neurais, totalizando 6 distribuições distintas em cada métrica de desempenho, um valor p menor que um determinado limiar indica que pelo menos dois modelos demonstraram diferenças de desempenho estatisticamente significativas (WITTE; WITTE, 2016).

A tabela 13 apresenta a estatística F e o valor-p de cada métrica em cada um dos mapas. Caso um conjunto de medidas apresente $p \leq 0{,}05$ em um determinado mapa, o valor é exibido em negrito na tabela, indicando diferença significativa no desempenho de pelo menos um dos métodos, comparado aos outros. Para cada curva, foram utilizadas as métricas coletadas nas 5 últimas etapas de avaliação, representativas do desempenho de cada modelo em sua fase final de aprendizado.

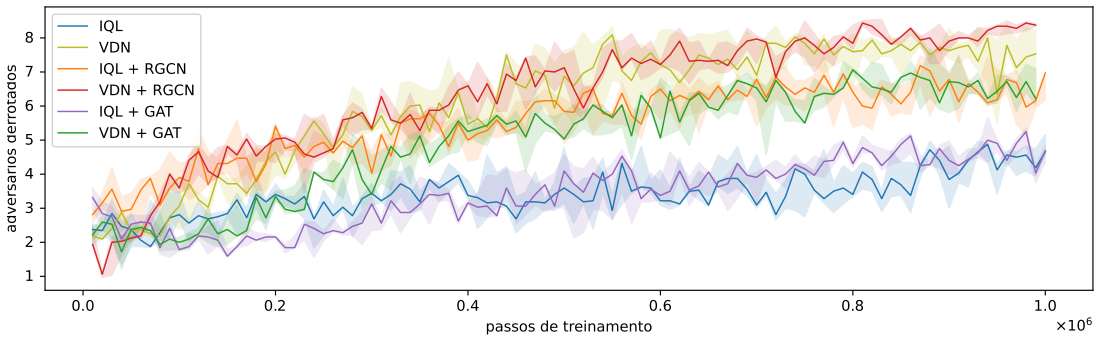
Assim como observado nas figuras, não existe diferença significativa entre as variações da HCA-Net(AO), em nenhuma métrica, no mapa `3m`, composto apenas de agentes homogêneos.

Nos casos em que $p < 0{,}05$ na tabela 13, foram realizados testes da Diferença Honestamente Significativa (HSD, *Honestly Significant Difference*) de Tukey (TUKEY, 1949), um teste *post hoc*, cuja realização é condicionada aos resultados da ANOVA. Sua aplicação é adequada para descobrir quais pares de distribuições possuem as médias diferentes, após concluir-se com a

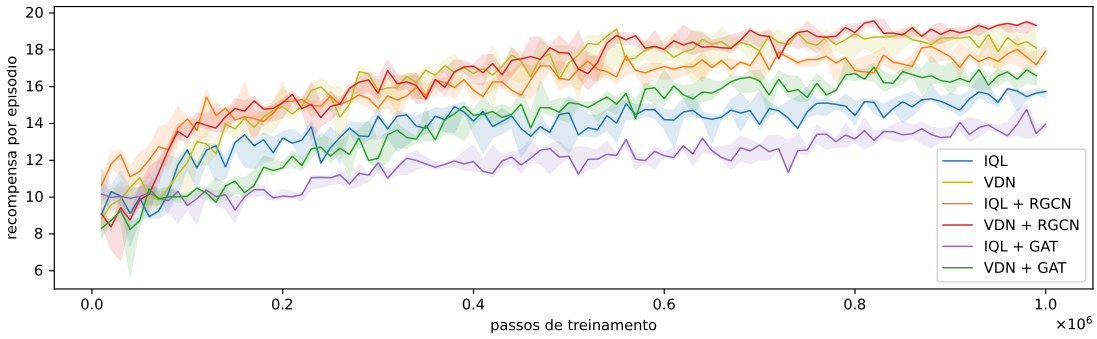
Figura 15 – Resultados das redes neurais no mapa `1c3s5z` durante 1 milhão de passos



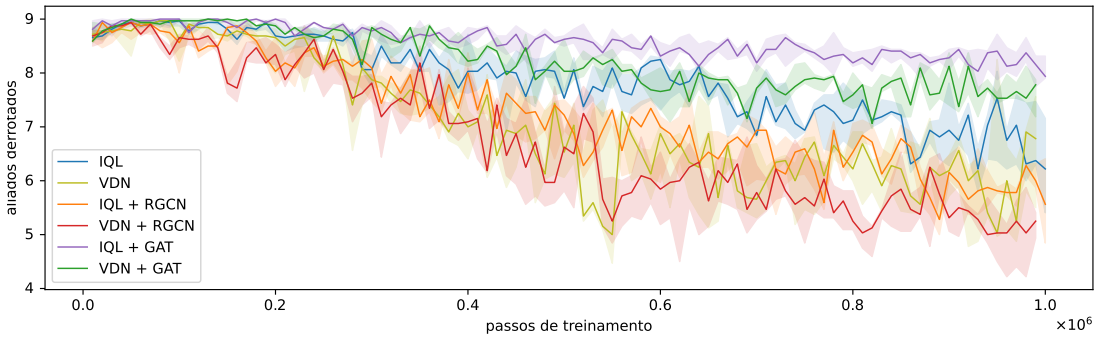
(a) Batalhas vencidas



(b) Adversários derrotados (máximo: 9)



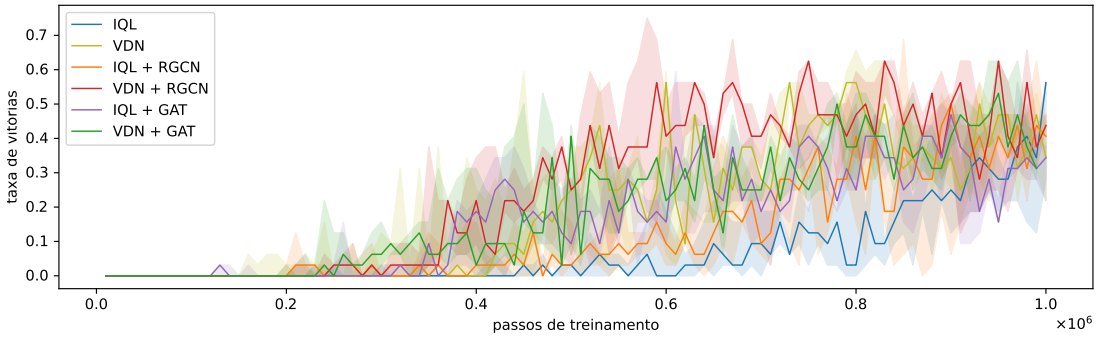
(c) Recompensa por episódio



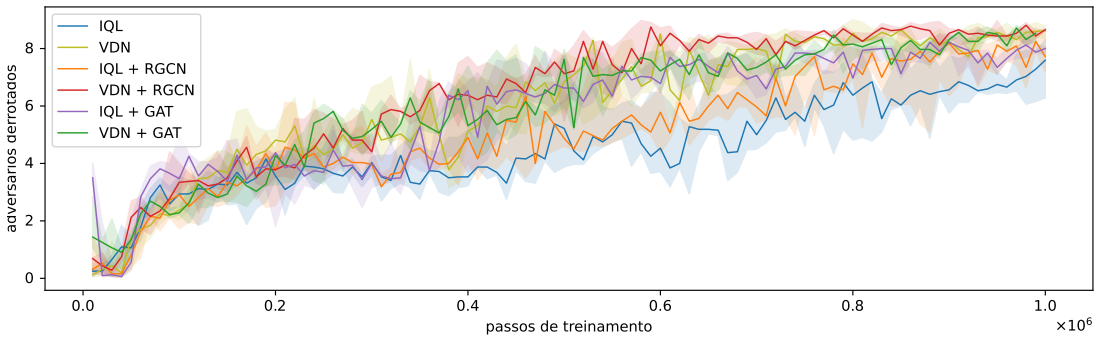
(d) Unidades aliadas perdidas

Fonte: Autor

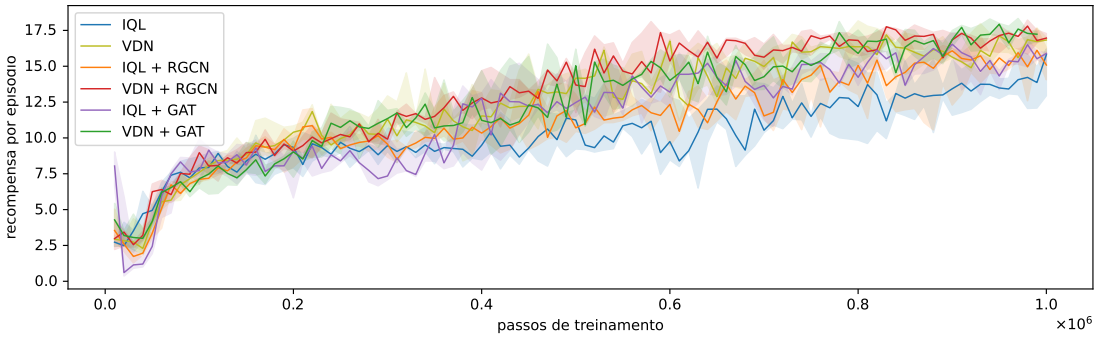
Figura 16 – Resultados das redes neurais no mapa `MMM` durante 1 milhão de passos



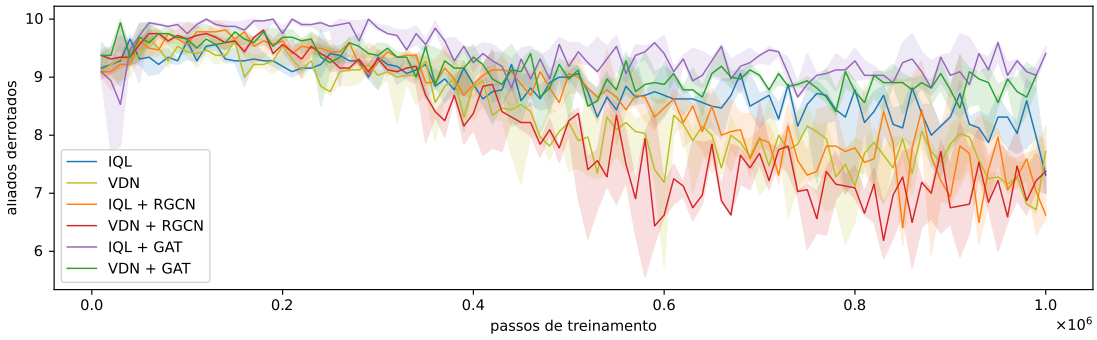
(a) Batalhas vencidas



(b) Adversários derrotados (máximo: 10)



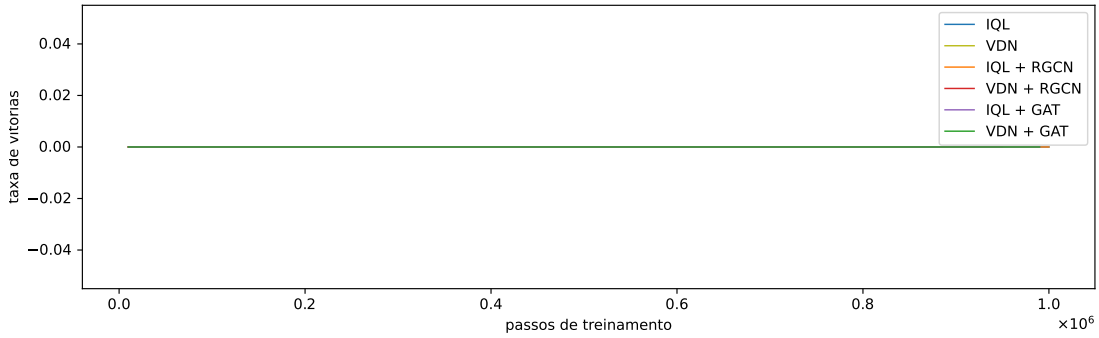
(c) Recompensa por episódio



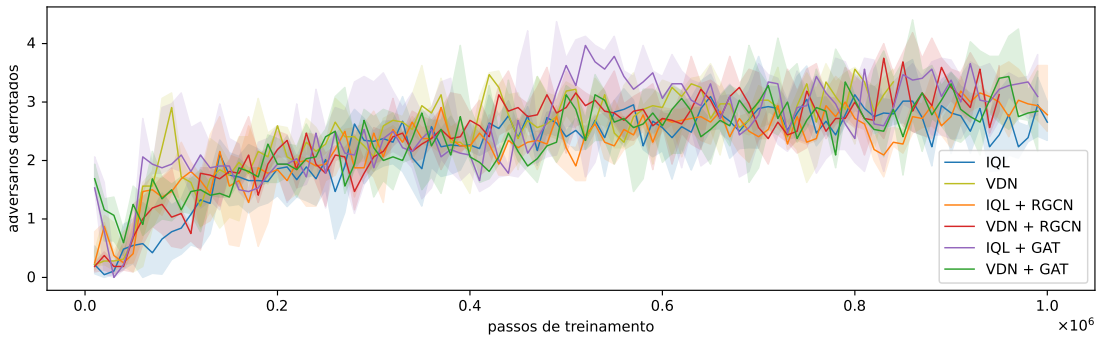
(d) Unidades aliadas perdidas

Fonte: Autor

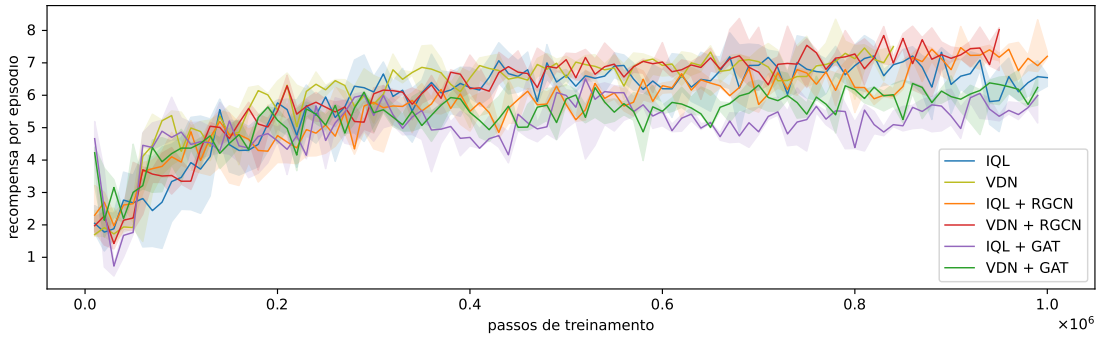
Figura 17 – Resultados das redes neurais no mapa `MMM2` durante 1 milhão de passos



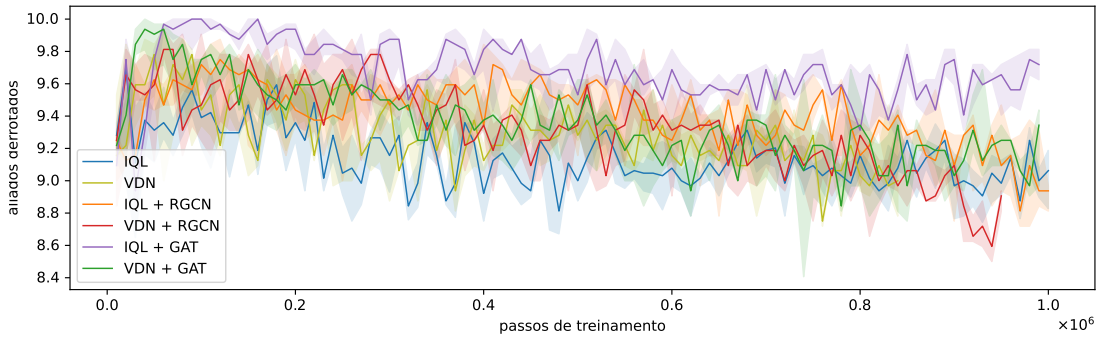
(a) Batalhas vencidas



(b) Adversários derrotados (máximo: 12)



(c) Recompensa por episódio



(d) Unidades aliadas perdidas

Fonte: Autor

Tabela 13 – ANOVA unilateral dos indicadores de desempenho dos diferentes métodos testados. Números em negrito apresentam $p < 0{,}05$

Métrica	Cenário	F	p
Taxa de vitórias	3m	1,0659	0,40355
	3s5z	—	—
	1c3s5z	6,9423	**0,00038**
	MMM	0,7369	0,60308
	MMM2	—	—
Adversários derrotados	3m	1,6778	0,17829
	3s5z	2,6167	**0,04172**
	1c3s5z	9,1779	$\mathbf{5{,}50695 \cdot 10^{-5}}$
	MMM	4,4890	**0,00498**
	MMM2	0,3040	0,90539
Recompensa por episódio	3m	1,4670	0,23733
	3s5z	23,971	$\mathbf{2{,}95059 \cdot 10^{-10}}$
	1c3s5z	17,802	$\mathbf{2{,}20038 \cdot 10^{-7}}$
	MMM	3,9198	**0,00970**
	MMM2	5,3841	**0,00202**
Aliados derrotados	3m	1,9697	0,11983
	3s5z	18,441	$\mathbf{7{,}82732 \cdot 10^{-9}}$
	1c3s5z	14,312	$\mathbf{1{,}53789 \cdot 10^{-6}}$
	MMM	12,936	$\mathbf{3{,}63836 \cdot 10^{-6}}$
	MMM2	2,1609	0,09402

Fonte: Autor

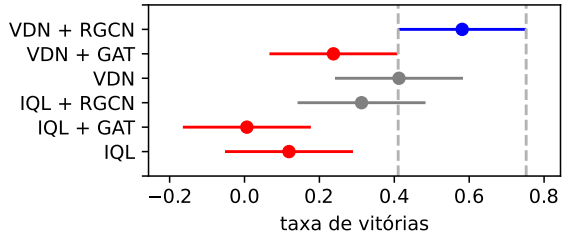
ANOVA que médias diferentes de fato existem (WITTE; WITTE, 2016). O teste HSD de Tukey é mais restritivo que a ANOVA, sendo que ele não permite rejeitar a hipótese nula em situações nas quais a ANOVA permitiria (WITTE; WITTE, 2016).

Os resultados dos testes são apresentados nas figuras 18, 19 e 20. Em todas as figuras, a barra azul indica a distribuição da métrica em questão para a HCA-Net(AO) com comunicação relacional e *mixing* aditivo (o modelo que se espera ter o melhor desempenho). Quando o teste de Tukey não considera a distribuição da métrica para um modelo diferente da distribuição representada em azul, ela é representada na cor cinza. Caso contrário, ela é representada pela cor vermelha.

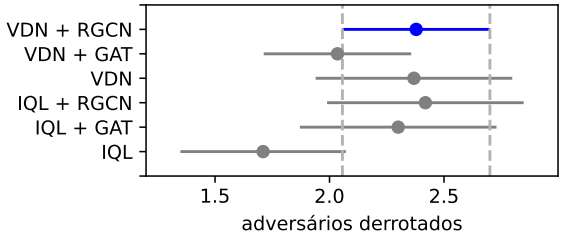
A taxa de vitórias no mapa 1c3s5z apresentou diferenças significativas, de acordo com seu valor-$p <= 0{,}05$. Como é possível observar na figura 18(a), a variação que utiliza comunicação relacional e *mixing* se equipara às que usam apenas *mixing*, e comunicação relacional sem *mixing*.

Com relação ao número de unidades adversárias derrotadas, os modelos demonstraram diferenças significativas nos mapas `1c3s5z` (onde já demonstrou desempenho superior em taxa de vitórias), `3s5z` (onde não venceu nenhum episódio) e `MMM`. No mapa `1c3s5z`, a variação com comunicação relacional, que já exibiu desempenho superior em taxa de vitórias, se mantém no grupo de métodos com melhor desempenho (fig. 18(c)). No mapa `3s5z`, onde nenhum modelo se mostrou capaz de vencer nenhum episódio, aqueles que se utilizam de *mixing* formam o grupo que mais derrotou adversários (fig. 18(b)), mas a significância com a qual o teste de Tukey considerou as duas distribuições distintas foi $p = 0{,}0571$, acima do limiar usado pela ANOVA. O desempenho insatisfatório da HCA-Net(AO) que emprega comunicação relacional no mapa `3s5z` será revisitado nos experimentos em que o treinamento dos modelos é efetuado por 6 milhões de episódios.

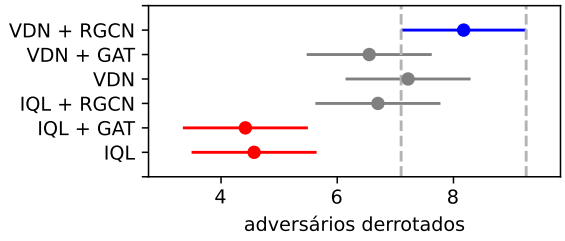
Figura 18 – Visualização do teste de Tukey para taxa de vitórias e número de adversários derrotados



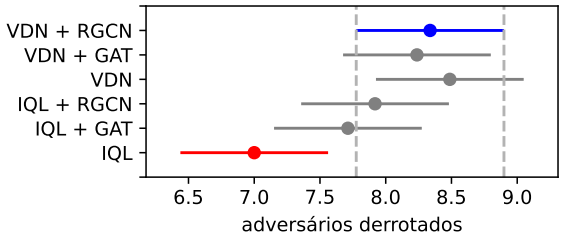
(a) Taxa de vitórias – 1c3s5z ($p = 0{,}00038$)



(b) Adversários derrotados – 3s5z ($p = 0{,}04172$)



(c) Adversários derrotados – 1c3s5z ($p = 5{,}50695 \cdot 10^{-5}$)



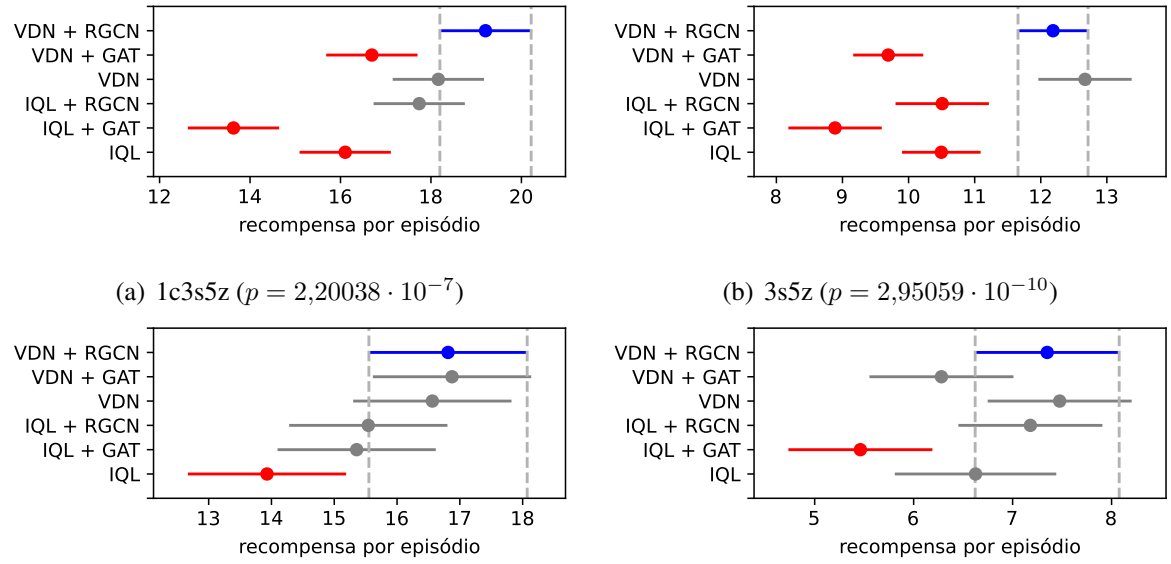
(d) Adversários derrotados – MMM ($p = 0{,}00498$)

Fonte: Autor

Com relação à recompensa acumulada, foram observadas diferenças significativas nos mapas `1c3s5z`, `3s5z`, `MMM` e `MMM2`. Nesta métrica, os resultados nos três primeiros mapas (figuras 19(a), 19(b) e 19(c)) se equipara aos resultados apresentados na figura 18, visto que, como mencionado na página 93, as métricas são relacionadas. Contudo, no mapa `MMM2` (no qual nenhuma variação da HCA-Net(AO) obteve vitórias ou demonstrou desempenho distinto no número de adversários derrotados), as variações que empregaram comunicação relacional (tanto

IQL como VDN) e a que usou apenas *mixing* obtiveram desempenho superior comparado à que utilizou IQL e comunicação por mecanismo de atenção (fig. 19(d)).

Figura 19 – Visualização do teste de Tukey para a recompensa acumulada por episódio



(a) 1c3s5z ($p = 2{,}20038 \cdot 10^{-7}$)

(b) 3s5z ($p = 2{,}95059 \cdot 10^{-10}$)

(c) MMM ($p = 0{,}00970$)

(d) MMM2 ($p = 0{,}00202$)

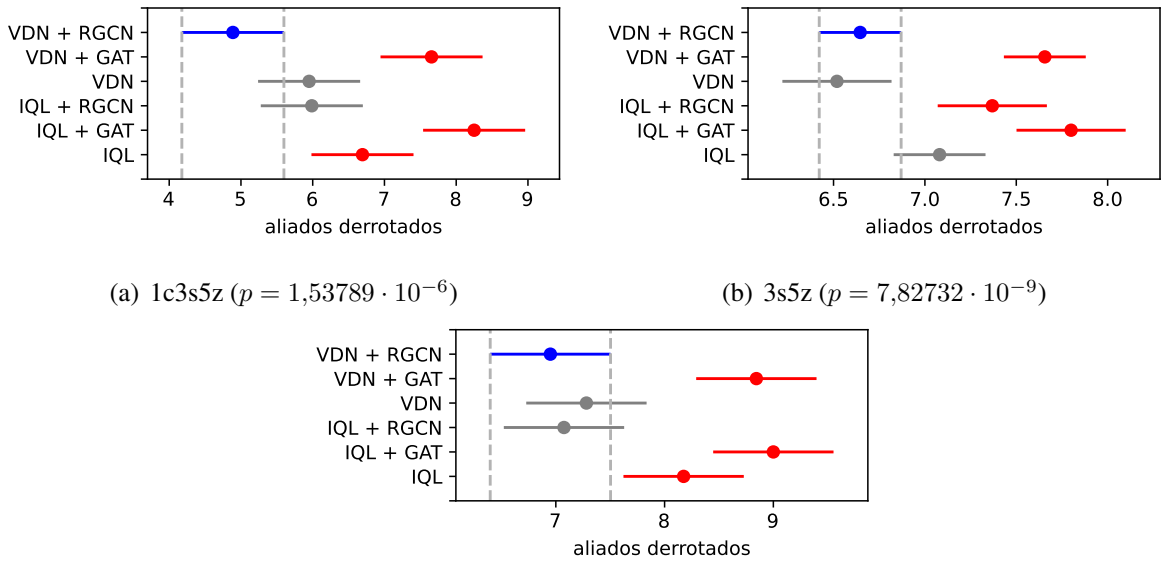
Fonte: Autor

Os mapas que demonstraram diferenças significativas no número de unidades aliadas perdidas (a única métrica não relacionada à recompensa dos agentes) foram `1c3s5z`, `3s5z` e `MMM`. Nos mapas `1c3s5z` e `MMM`, os métodos que empregam comunicação relacional (tanto IQL como VDN) assim como o que se utilizou apenas do *mixing* tiveram o melhor desempenho, apresentando o menor número médio de unidades aliadas perdidas. Já no mapa `3s5z`, o método que usou comunicação relacional e *mixing* demonstrou o mesmo desempenho que aquele que utilizou apenas *mixing* e o IQL, que não utiliza comunicação nem *mixing*.

Devido à ausência de vitórias no cenário `3s5z`, considerado vencível (SAMVELYAN et al., 2019), assim como no cenário `MMM2` (SAMVELYAN et al., 2019), considerado super-difícil, um subconjunto das variações da HCA-Net(AO) foi treinado por 6 milhões de episódios nestes cenários.

Devido ao extenso tempo necessário para realizar 6 milhões de passos de treinamento, os experimentos foram realizados apenas com um subconjunto dos modelos e não foram repetidos múltiplas vezes. As variações foram selecionadas seguindo um critério ablativo, de modo a contrastar a contribuição da presença e ausência de comunicação, assim como a contribuição do

Figura 20 – Visualização do teste de Tukey para o número de aliados derrotados



(a) 1c3s5z ($p = 1{,}53789 \cdot 10^{-6}$)

(b) 3s5z ($p = 7{,}82732 \cdot 10^{-9}$)

(c) MMM ($p = 3{,}63836 \cdot 10^{-6}$)

Fonte: Autor

mixing com as três categorias de comunicação (relacional, por mecanismo de atenção e ausência de comunicação).

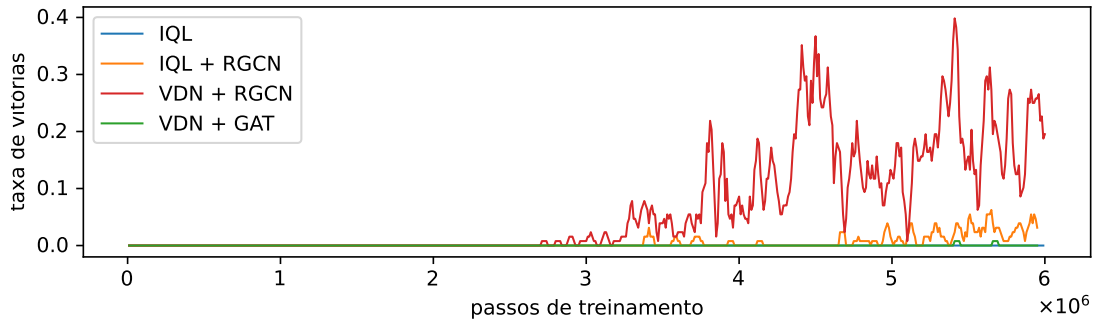
As figuras 21, 22, 23 e 24 apresentam as métricas de desempenho coletadas nos treinamentos realizados por 6 milhões de passos. Foi realizada uma etapa de avaliação a cada 10 mil passos de treinamento, realizando-se 32 episódios com uma política gulosa, de onde as métricas foram coletadas. Os gráficos exibem a média móvel comum (não exponencial) com uma janela de tamanho 3, a fim de reduzir o ruído das métricas.

A figura 21 apresenta a taxa de vitórias nos dois cenários. Em contraste com os resultados apresentados nas figuras 14(a) e 17(a), é possível observar a capacidade de alguns modelos aprenderem a vencer partidas quando treinados por períodos extensos de tempo. No cenário `3s5z`, os modelos que se utilizaram de comunicação relacional foram os únicos capazes de aprender a vencer partidas, quando treinados por cerca de 6 milhões de passos (fig. 21(a)).

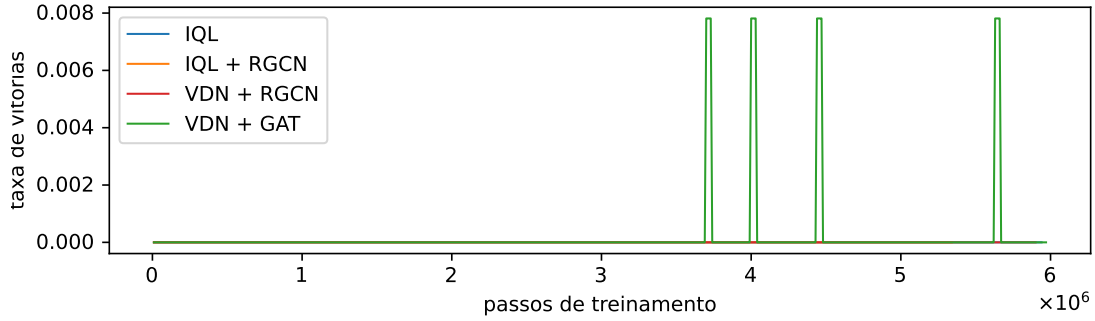
Já no cenário `MMM2`, o modelo que utilizou *mixing* e comunicação por mecanismo de atenção foi capaz de vencer partidas, porém de forma inconsistente. O baixo desempenho de todos os métodos é esperado, dada a dificuldade do cenário (SAMVELYAN et al., 2019).

A figura 22 apresenta a média de adversários derrotados por episódio nos dois cenários. Ao serem treinados por mais iterações, alguns modelos foram capazes de derrotar mais unidades adversárias, indicando indiretamente a capacidade de alcançar mais vitórias.

Figura 21 – Porcentagem de partidas vencidas por etapa de avaliação em 6 milhões de passos de treinamento



(a) 3s5z



(b) MMM2

Fonte: Autor

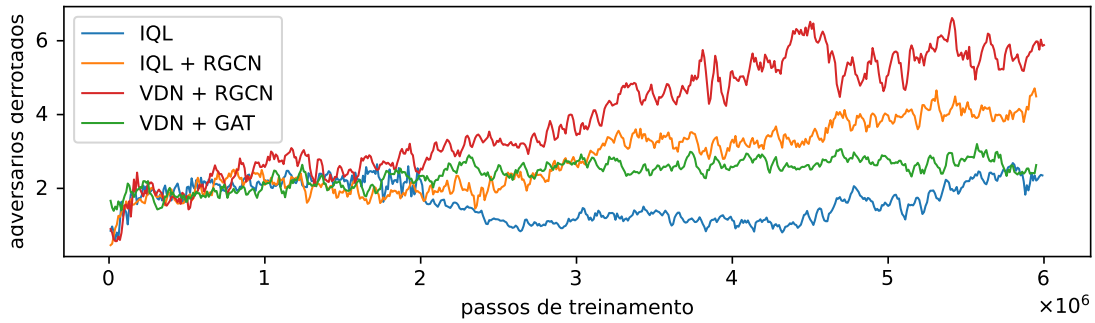
No cenário `3s5z`, os modelos treinados com comunicação relacional conseguiram derrotar entre 4 a 8 adversários nas etapas finais de avaliação (fig. 22(a)), enquanto os outros modelos não superaram a marca de 3 adversários derrotados, de um máximo de 8 (cf. figura 14(b) na qual todos os métodos também não ultrapassaram esta marca).

No cenário `MMM2`, alguns modelos foram capazes de ultrapassar os resultados antes alcançados e exibidos na figura 17(b), porém não se aproximaram do valor necessário para alcançar a vitória de 12 adversários, indicando que o tempo prolongado de treinamento pode não ser o suficiente para vencer este cenário super-difícil.

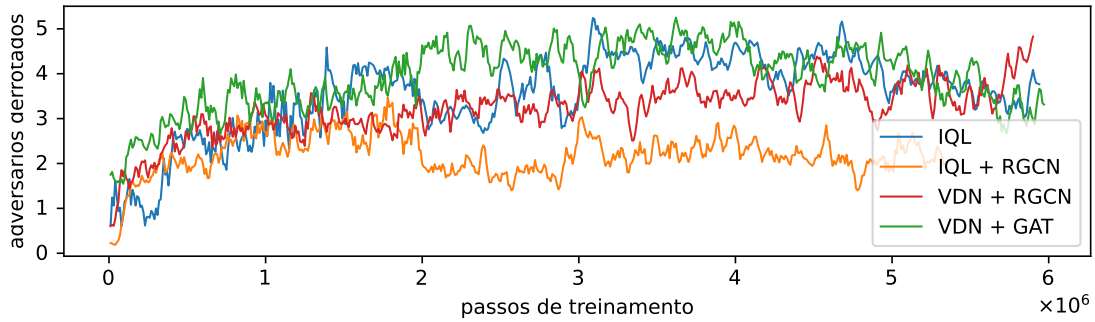
A figura 23 apresenta a recompensa acumulada média nos dois cenários. Os valores se assemelham aos da figura 22, não possibilitando conclusões adicionais.

A figura 24 apresenta a média de unidades perdidas nos dois cenários. Mesmo não compondo a recompensa observada pelos agentes, é possível perceber que o método que utiliza comunicação relacional e *mixing* tendeu a perder menos unidades por episódios no cenário `3s5z`

Figura 22 – Média de adversários derrotados por episódio, por etapa de avaliação em 6 milhões de passos de treinamento



(a) 3s5z (8 adversários)



(b) MMM2 (12 adversários)

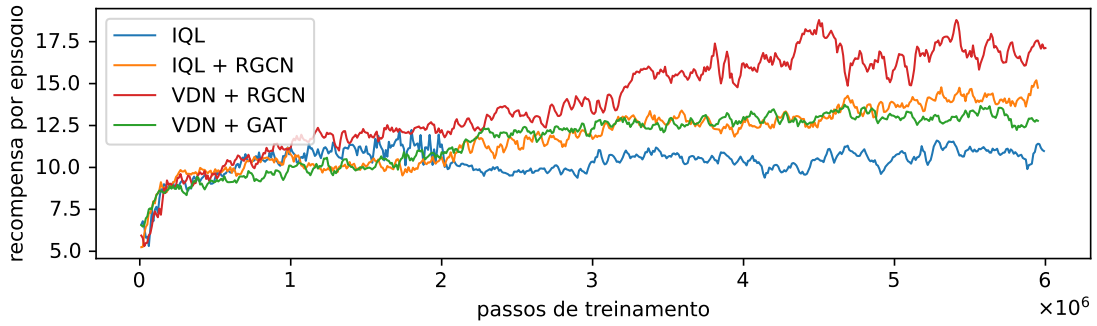
Fonte: Autor

(fig. 23(a)). Já o método que empregou comunicação por mecanismo de atenção foi o que perdeu mais unidades em ambos os cenários.

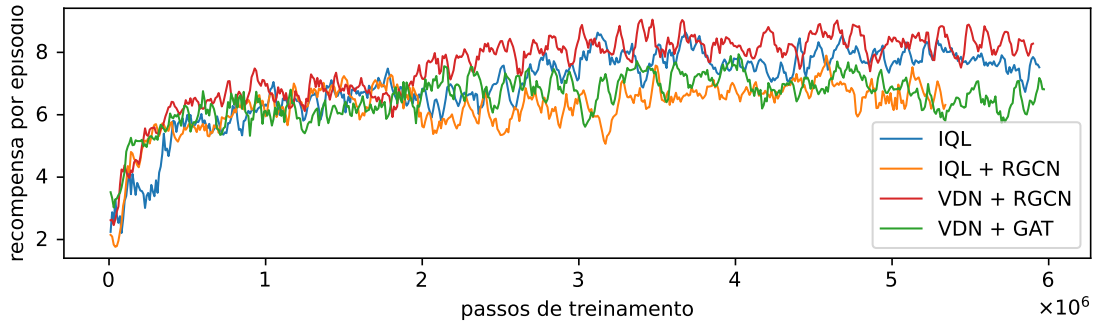
6.5 DISCUSSÃO

Esta seção apresentou a HCA-Net(AO), a versão final do método apresentado na tese. Ao combinar os conceitos do grafo heterogêneo de agentes e entidades, e a comunicação relacional da HCA-Net(EF), apresentada na seção 5; incorporar novas informações nos vetores de características armazenados em cada vértice dos grafos e aprimorar o modelo e seu algoritmo de treinamento com métodos advindos de trabalhos recentes em MADRL cooperativo (ao invés de trabalhos puramente baseados em MADRL com redes neurais de grafos), foi observada uma melhora significativa no desempenho do método. Também foi possível comparar a HCA-Net(AO) com variações igualmente relevantes, como uma versão do IQL implementada utilizando redes

Figura 23 – Média da recompensa acumulada por episódio, por etapa de avaliação, em 6 milhões de passos de treinamento



(a) 3s5z



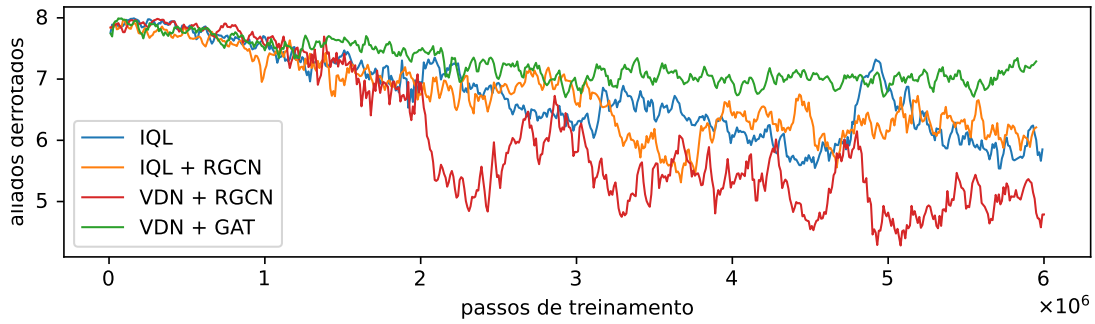
(b) MMM2

Fonte: Autor

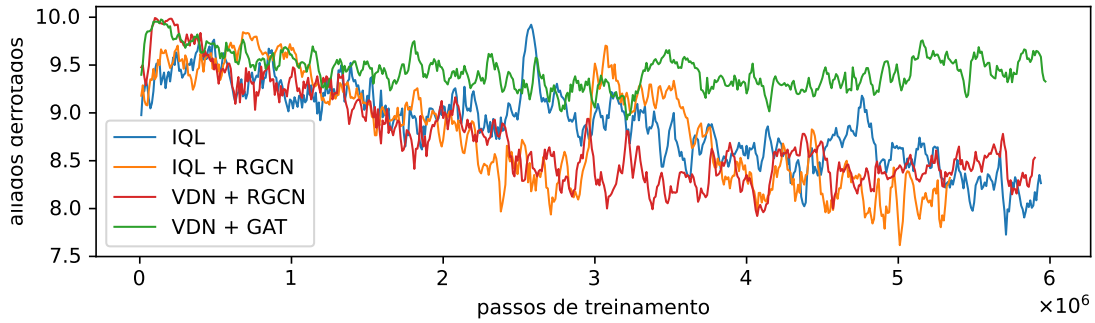
neurais, métodos que utilizam comunicação por mecanismo de atenção em grafos e métodos que utilizam o *mixing* aditivo através da VDN.

Os resultados demonstraram que existem situações nas quais a comunicação relacional implementada da maneira descrita neste trabalho aprimora o desempenho da rede neural em tarefas cooperativas compostas de times de agentes heterogêneos. Nos cenários do SMAC utilizados nos experimentos, a HCA-Net(AO) com comunicação relacional demonstra o melhor desempenho dentre todos os métodos testados, ou posiciona-se no grupo dos métodos que alcançaram o melhor desempenho. A comunicação relacional foi consistentemente melhor que a comunicação por mecanismo de atenção, também sendo capaz de posicionar o IQL no mesmo patamar de desempenho que os métodos que utilizaram *mixing* através da VDN. Contudo, em todos os experimentos, a rede que utilizou a VDN e comunicação relacional não alcançou desempenho superior ao método que utilizou apenas a VDN, indicando que não existe vantagem em combinar os dois métodos. Esse é um forte indicativo de que o aprendizado centralizado e execução descentralizada dos métodos de *mixing* podem ser suficientes para

Figura 24 – Média de unidades aliadas perdidas por etapa de avaliação em 6 milhões de passos de treinamento



(a) 3s5z



(b) MMM2

Fonte: Autor

alcançar a coordenação dos agentes em alguns domínios, como é o caso do SMAC, de acordo com Whiteson (2020).

Apesar disso, nos mapas nos quais os agentes demoram extensos períodos de exploração para aprender políticas capazes de vencer partidas, os métodos que utilizaram comunicação relacional foram os únicos que demonstraram a capacidade de aprender tais políticas. Também foram os métodos que derrotaram mais adversários, acumularam mais recompensa e perderam menos unidades aliadas em um dos mapas (`3s5z`), indicando que, nos cenários que necessitam de extensos períodos de aprendizado, o uso de comunicação relacional nos times de agentes heterogêneos permite o aprendizado de políticas aproveitáveis.

7 CONCLUSÕES

A presença de agentes heterogêneos em um problema de aprendizado por reforço multi-agentes adiciona desafios a um problema que, em sua forma original, já se mostra desafiador. Em situações nas quais soluções baseadas em redes neurais são implementadas, questões como a modelagem de entradas de tamanho variado, aprendizado de múltiplas políticas e implementação de comunicação inter-agentes também se apresentam como passos para a criação de uma metodologia adequada para o controle de múltiplos agentes.

Motivado por esses desafios, este trabalho apresentou as HCA-Nets, arquiteturas de rede neural voltadas ao aprendizado de políticas e protocolos de comunicação em sistemas totalmente cooperativos com múltiplos agentes heterogêneos.

Em conjunto com uma nova representação de estados baseada em grafos heterogêneos direcionados rotulados atribuídos, utilizada para representar a passagem de mensagens entre agentes e entidades heterogêneas, e um algoritmo de treinamento adequado, a rede neural com comunicação relacional proposta na seção 5, denominada HCA-Net(EF), foi capaz de alcançar desempenho superior a métodos com comunicação baseada em mecanismos de atenção para grafos e métodos sem comunicação. Além disso, todos os métodos implementados alcançaram desempenho superior a um agente aleatório.

Na seção 6, a HCA-Net(EF) foi aprimorada de forma a codificar apenas agentes heterogêneos e suas observações; compartilhar parâmetros dos módulos de codificação e seleção de ações da rede neural entre todos os agentes, não apenas classes de agentes homogêneos; e realizar o treinamento da rede neural usando lotes de episódios e não de transições, utilizando camadas LSTM para processar as sequências de transições dos episódios. Essas mudanças deram origem ao segundo método, denominado HCA-Net(AO).

Em primeiro lugar, a melhoria de desempenho da HCA-Net(AO) após a substituição dos vetores de características armazenados nos vértices dos grafos por vetores contendo as observações dos agentes demonstra que, ao contrário da suposição feita no início do trabalho, os agentes não são capazes de reconstituir observações suficientemente informativas apenas transmitindo suas próprias características. A inclusão das observações completas dos agentes nos vetores dos vértices do grafo torna a representação do estado mais complexa (no caso particular do SMAC), porém pode ser considerada uma representação adequada e plausível de ser utilizada na maioria dos ambientes.

Em segundo lugar, os resultados apresentados na seção 6 demonstram que a HCA-Net(AO) que emprega compartilhamento de parâmetros entre agentes heterogêneos e comunicação através de Convoluções Relacionais em Grafos, apresentou desempenho similar ou superior a redes neurais alternativas que não utilizavam comunicação relacional. Em algumas métricas, o uso de comunicação relacional melhorou o desempenho de redes neurais que utilizavam o *mixing* aditivo ou faziam a rede neural sem *mixing* alcançar desempenho comparável à rede com *mixing*.

Uma terceira observação foi a de que, nos cenários nos quais nenhum método foi capaz de aprender políticas capazes de vencer partidas do SMAC dentro de 1 milhão de passos de treinamento, o aumento do tempo de treinamento para 6 milhões de passos demonstrou que apenas os modelos que utilizaram comunicação relacional foram capazes de aprender tais políticas. A sensibilidade dos métodos de MADRL a hiperparâmetros e decisões gerais do processo de treinamento é sabida (HU, S. et al., 2021), porém a busca pelas condições ótimas de treinamento foi considerada fora do escopo deste trabalho, devido à quantidade de tempo real necessária para realizar essas buscas.

Como contribuições para a comunidade científica, a versão do método descrita na seção 5 foi publicada e apresentada em Meneghetti e Bianchi (2020a), enquanto a versão descrita na seção 6 foi publicada e apresentada em Meneghetti e Bianchi (2021). Outras publicações realizadas durante o doutorado são listadas no apêndice A. Os programas de *software* criados neste período, assim como as contribuições feitas a trabalhos de terceiros são listadas no apêndice B.

7.1 TRABALHOS FUTUROS

Apesar de o *StarCraft Multi-Agent Challenge* (SMAC) ter se tornado um ambiente padrão para testes de novos métodos de controle de times de agentes em sistemas totalmente cooperativos, características como a presença de ações parametrizadas (*e.g.* ações para atacar adversários específicos) e o número fixo de agentes por cenário limitaram os testes que puderam ser feitos nos métodos propostos. Em vista disso, um trabalho futuro proposto consiste na criação de um novo ambiente para aprendizado por reforço multi-agentes heterogêneo sem essas limitações, nos quais os métodos apresentados nesta tese possam ser aferidos em situações com número variado de agentes, características essas que as GNNs já são capazes de realizar.

Outros ambientes de interesse para experimentações com os métodos aqui apresentados são aqueles que possuem agentes com capacidades homogêneas, porém com a expectativa de executar papéis diferentes (políticas ótimas distintas) como, por exemplo, o futebol de robôs

humanoides, onde robôs homogêneos de um mesmo time possam se beneficiar ao aprender políticas que os especializem em posições de ataque ou defesa.

Outro trabalho futuro proposto consiste na criação de técnicas de aprendizado por reforço profundo multi-agentes que tratem números variados de classes de agentes, desafio este que este trabalho e os trabalhos relacionados não enfrentaram diretamente. A solução deste problema aproximaria as técnicas de MARL da solução do problema de coordenação *ad hoc* proposto por Stone et al. (2010).

Uma última melhoria possível nos métodos aqui apresentados consiste no uso do Adam (KINGMA; BA, 2017) como otimizador da rede neural, no lugar do RMSProp (HINTON; SRIVASTAVA; SWERSKY, 2012), visto que trabalhos contemporâneos (VIEILLARD; PIETQUIN; GEIST, 2020; HU, S. et al., 2021) apontam que o primeiro método é mais estável no treinamento de redes neurais profundas em tarefas de aprendizado por reforço que o segundo.

REFERÊNCIAS

AGARWAL, Akshat; KUMAR, Sumit; SYCARA, Katia. Learning Transferable Cooperative Behavior in Multi-Agent Teams. In: ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Representations. [S.l.: s.n.], 4 jun. 2019. Citado nas pp. 56, 70, 79, 84.

AMATO, Christopher. Cooperative Decision Making. In: KOCHENDERFER, Mykel J. **Decision Making Under Uncertainty**. [S.l.]: The MIT Press, 2015. ISBN 978-0-262-33170-8. DOI: 10.7551/mitpress/10187.003.0011. Disponível em: https://direct.mit.edu/books/book/4074/chapter/168950/cooperative-decision-making. Acesso em: 20 set. 2020. Citado nas pp. 31, 32, 83.

AQUINO JUNIOR, Plinio Thomaz et al. HERA: Home Environment Robot Assistant. In: II BRAZILIAN HUMANOID ROBOT WORKSHOP (BRAHUR) and III Brazilian Workshop on Service Robotics (BRASERO). [S.l.: s.n.], abr. 2019. Citado na p. 133.

ASHUAL, Oron; WOLF, Lior. Specifying Object Attributes and Relations in Interactive Scene Generation. In: PROCEEDINGS OF THE IEEE/CVF International Conference ON Computer Vision, p. 4561–4569. Disponível em: https://openaccess.thecvf.com/content_ICCV_2019/html/Ashual_Specifying_Object_Attributes_and_Relations_in_Interactive_Scene_Generation_ICCV_2019_paper.html. Acesso em: 5 mai. 2021. Citado na p. 18.

ÅSTRÖM, K.J. Optimal Control of Markov Processes with Incomplete State Information. **Journal of Mathematical Analysis and Applications**, v. 10, n. 1, p. 174–205, fev. 1965. ISSN 0022247X. DOI: 10/bwg3n7. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/0022247X6590154X. Acesso em: 20 nov. 2020. Citado na p. 25.

BALCH, Tucker; ARKIN, Ronald C. Communication in Reactive Multiagent Robotic Systems. **Autonomous Robots**, v. 1, n. 1, p. 27–52, 1994. ISSN 0929-5593, 1573-7527. DOI: 10/cz7kzg. Disponível em: http://link.springer.com/10.1007/BF00735341. Acesso em: 2 nov. 2020. Citado nas pp. 33, 68.

BAPST, V. et al. Unveiling the Predictive Power of Static Structure in Glassy Systems. **Nature Physics**, Nature Publishing Group, v. 16, n. 4, p. 448–454, 4 abr. 2020. ISSN 1745-2481. DOI: 10/drd5. Disponível em: https://www.nature.com/articles/s41567-020-0842-8. Acesso em: 5 mai. 2021. Citado na p. 18.

BATTAGLIA, Peter et al. Interaction Networks for Learning about Objects, Relations and Physics. In: LEE, D. et al. (Ed.). **Advances in Neural Information Processing Systems**. [S.l.]: Curran Associates, Inc., 2016. v. 29, p. 4502–4510. Disponível em: https://proceedings.neurips.cc/paper/2016/file/3147da8ab4a0437c15ef51a5cc7f2dc4-Paper.pdf. Citado nas pp. 18, 37.

BATTAGLIA, Peter W. et al. Relational Inductive Biases, Deep Learning, and Graph Networks. **arXiv e-prints**, 4 jun. 2018. arXiv: 1806.01261. Disponível em: https://arxiv.org/abs/1806.01261. Citado na p. 36.

BELLMAN, Richard. **Dynamic Programming**. Princeton, New Jersey: Princeton University Press, 1957. DOI: 10.2307/2310708. Citado na p. 22.

BERNSTEIN, Daniel S. et al. The Complexity of Decentralized Control of Markov Decision Processes. **Mathematics of Operations Research**, v. 27, n. 4, p. 819–840, nov. 2002. ISSN 0364-765X, 1526-5471. DOI: 10/d2d7f5. Disponível em: http://pubsonline.informs.org/doi/abs/10.1287/moor.27.4.819.297. Acesso em: 18 nov. 2020. Citado na p. 31.

BIANCHI, Reinaldo Augusto da Costa et al. **Open Soccer Ball Dataset**. [S.l.]: IEEE, 21 set. 2020. DOI: 10/ghcfxn. Disponível em: https://ieee-dataport.org/open-access/open-soccer-ball-dataset. Citado na p. 133.

BLUMENKAMP, Jan; PROROK, Amanda. **The Emergence of Adversarial Communication in Multi-Agent Reinforcement Learning**. 4 nov. 2020. arXiv: 2008.02616 [cs]. Disponível em: http://arxiv.org/abs/2008.02616. Acesso em: 16 abr. 2021. Citado na p. 57.

BOEHMER, Wendelin; KURIN, Vitaly; WHITESON, Shimon. Deep Coordination Graphs. In: III, Hal Daumé; SINGH, Aarti (Ed.). **Proceedings of the 37th International Conference on Machine Learning**. [S.l.]: PMLR, 13–18 jul. 2020. v. 119. (Proceedings of Machine Learning Research), p. 980–991. Disponível em: http://proceedings.mlr.press/v119/boehmer20a.html. Citado nas pp. 55, 59.

BONDY, J. A.; MURTY, U. S. R. **Graph Theory**. [S.l.]: Springer London, 2008. DOI: 10.1007/978-1-84628-970-5. Citado na p. 36.

BORGWARDT, K. M. et al. Protein Function Prediction via Graph Kernels. **Bioinformatics**, v. 21, p. i47–i56, Suppl 1 1 jun. 2005. ISSN 1367-4803, 1460-2059. DOI: 10/c6s8jt. Disponível em: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bti1007. Acesso em: 26 mai. 2021. Citado na p. 40.

BOUTILIER, C.; DEAN, T.; HANKS, S. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. **Journal of Artificial Intelligence Research**, v. 11, p. 1–94, 1 jul. 1999. ISSN 1076-9757. DOI: 10/gjtqdm. Disponível em: https://www.jair.org/index.php/jair/article/view/10237. Acesso em: 29 abr. 2021. Citado na p. 55.

BOUTILIER, Craig. Sequential Optimality and Coordination in Multiagent Systems. In: DEAN, Thomas (Ed.). **Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 Pages**. [S.l.]: Morgan Kaufmann, 1999. P. 478–485. Disponível em: http://ijcai.org/Proceedings/99-1/Papers/069.pdf. Citado na p. 31.

BRAFMAN, Ronen I.; TENNENHOLTZ, Moshe. Learning to Coordinate Efficiently: A Model-Based Approach. **Journal of Artificial Intelligence Research**, v. 19, p. 11–23, 2003. DOI: 10/ghhpnc. Disponível em: https://doi.org/10.1613/jair.1154. Citado na p. 76.

BROCKMAN, Greg et al. OpenAI Gym. **CoRR**, abs/1606.01540, 2016. Disponível em: http://arxiv.org/abs/1606.01540. Citado na p. 49.

BURO, Michael. ORTS: A Hack-Free RTS Game Environment. In: COMPUTERS and Games. [S.l.]: Springer Berlin Heidelberg, 2003. P. 280–291. DOI: 10.1007/978-3-540-40031-8_19. Citado na p. 42.

CALVO, J.A.; DUSPARIC, I. Heterogeneous Multi-Agent Deep Reinforcement Learning for Traffic Lights Control. In: CEUR Workshop Proceedings. v. 2259, p. 2–13. Disponível em: http://www.tara.tcd.ie/handle/2262/89897. Citado na p. 59.

CASTELLINI, Jacopo et al. The Representational Capacity of Action-Value Networks for Multi-Agent Reinforcement Learning. In: PROCEEDINGS of the 18th International Conference on Autonomous Agents and MultiAgent Systems. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 8 mai. 2019. (AAMAS '19). ISBN 978-1-4503-6309-9. arXiv: 1902.07497. Citado na p. 55.

CEN, Yukuo et al. Representation Learning for Attributed Multiplex Heterogeneous Network. In: PROCEEDINGS of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York, NY, USA: Association for Computing Machinery, 25 jul. 2019. (KDD '19), p. 1358–1368. ISBN 978-1-4503-6201-6. DOI: 10/gf7m2j. Disponível em: https://doi.org/10.1145/3292500.3330964. Acesso em: 2 abr. 2021. Citado nas pp. 21, 37, 39, 61, 62.

CERTICKY, Martin; SARNOVSKY, Martin; VARGA, Tomas. Use of Machine Learning Techniques in Real-Time Strategy Games. In: 2018 World Symposium ON Digital Intelligence FOR Systems AND Machines (DISA), Kosice. **2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)**. [S.l.]: IEEE, ago. 2018–8. (DISA 2018 - IEEE World Symposium on Digital Intelligence for Systems and Machines, Proceedings), p. 159–164. ISBN 978-1-5386-5102-5. DOI: 10/gj2zgx. Disponível em: https://ieeexplore.ieee.org/document/8490528/. Citado nas pp. 42, 47.

CERTICKY, Michal et al. StarCraft AI Competitions, Bots and Tournament Manager Software. **IEEE Transactions on Games**, p. 1–1, 2018. ISSN 2475-1502. DOI: 10/gj2zgw. Disponível em: https://ieeexplore.ieee.org/document/8544023/. Citado na p. 42.

CHO, Kyunghyun et al. Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation. In: PROCEEDINGS of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). [S.l.]: Association for Computational Linguistics, 3 jun. 2014. DOI: 10/gddmvq. arXiv: 1406.1078. Citado na p. 36.

DEBNATH, Asim Kumar et al. Structure-Activity Relationship of Mutagenic Aromatic and Heteroaromatic Nitro Compounds. Correlation with Molecular Orbital Energies and Hydrophobicity. **Journal of Medicinal Chemistry**, v. 34, n. 2, p. 786–797, fev. 1991. ISSN 0022-2623, 1520-4804. DOI: 10/d79bzx. Disponível em: https://pubs.acs.org/doi/abs/10.1021/jm00106a046. Acesso em: 26 mai. 2021. Citado na p. 40.

DE OLIVEIRA, Jonas Henrique Renolfi et al. Object Detection under Constrained Hardware Scenarios: A Comparative Study of Reduced Convolutional Network Architectures. In: 2019

Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium ON Robotics (SBR) AND 2019 Workshop ON Robotics IN Education (WRE). **2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)**. Rio Grande, Brazil: IEEE, out. 2019. P. 25–30. ISBN 978-1-72814-268-5. DOI: 10/ghpz8g. Disponível em: https://ieeexplore.ieee.org/document/9018560/. Citado na p. 132.

DE WITT, Christian Schroeder et al. **Deep Multi-Agent Reinforcement Learning for Decentralized Continuous Cooperative Control**. 6 jun. 2020. arXiv: 2003.06709 [cs, stat]. Disponível em: http://arxiv.org/abs/2003.06709. Acesso em: 21 set. 2020. Citado na p. 59.

DOMINGUES, Pedro Henrique Silva; MENEGHETTI, Douglas De Rizzo; AQUINO JUNIOR, Plinio Thomaz. Comparação Entre Métodos de Detecção de Objetos Por Pontos-Chave e Redes Neurais. In: VIII Simpósio DE Iniciação Científica, Didática E DE Ações Sociais DA FEI. Disponível em: https://fei.edu.br/sites/sicfei/2018/cc/SICFEI_2018_paper_106.pdf. Citado na p. 133.

DOSTOIEVSKI, F. **O Jogador**. [S.l.]: Martin Claret, 2006. (Coleção a Obra-Prima de Cada Autor). ISBN 978-85-7232-297-3. Disponível em: https://books.google.com.br/books?id=UEnmMAAACAAJ.

DUVENAUD, David et al. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In: PROCEEDINGS of the 28th International Conference on Neural Information Processing Systems - Volume 2. Cambridge, MA, USA: MIT Press, 7 dez. 2015. (NIPS'15), p. 2224–2232. arXiv: 1509.09292v2. Citado nas pp. 17, 37.

EGOROV, Maxim. **Multi-Agent Deep Reinforcement Learning**. [S.l.], 2016. Disponível em: http://cs231n.stanford.edu/reports/2016/pdfs/122_Report.pdf. Citado na p. 34.

ELMAN, Jeffrey L. Finding Structure in Time. **Cognitive Science**, v. 14, n. 2, p. 179–211, mar. 1990. ISSN 03640213. DOI: 10/fwxk7w. Disponível em: http://doi.wiley.com/10.1207/s15516709cog1402_1. Acesso em: 22 nov. 2020. Citado na p. 36.

FAN, Wenqi et al. Graph Neural Networks for Social Recommendation. In: THE World Wide Web Conference. New York, NY, USA: Association for Computing Machinery, 13 mai. 2019. (WWW '19), p. 417–426. ISBN 978-1-4503-6674-8. DOI: 10/ghcqmw. Disponível em: https://dl.acm.org/doi/10.1145/3308558.3313488. Acesso em: 5 mai. 2021. Citado na p. 18.

FELLBAUM, Christiane. WordNet. In: POLI, Roberto; HEALY, Michael; KAMEAS, Achilles (Ed.). **Theory and Applications of Ontology: Computer Applications**. Dordrecht: Springer Netherlands, 2010. P. 231–243. ISBN 978-90-481-8847-5. DOI: 10.1007/978-90-481-8847-5_10. Disponível em: https://doi.org/10.1007/978-90-481-8847-5_10. Acesso em: 18 abr. 2021. Citado na p. 40.

FERREIRA, Leonardo Anjoletto; MENEGHETTI, Douglas De Rizzo; SANTOS, Paulo Eduardo. CAPTION: Correction by Analyses, POS -Tagging and Interpretation of Objects Using Only Nouns. In: FIRST Annual International Workshop ON Interpretability: METHODOLOGIES AND Algorithms (IMA 2019). Citado na p. 133.

FERREIRA, Leonardo Anjoletto et al. CAPTION: Caption Analysis with Proposed Terms, Image's Objects and NLP. **Information Processing and Management**, p. 22, abr. 2021. ISSN 0306-4573. Citado na p. 132.

FEY, Matthias; LENSSEN, Jan Eric. Fast Graph Representation Learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds. [S.l.: s.n.], 6 mar. 2019. arXiv: 1903.02428v3. Citado na p. 74.

FOERSTER, J. et al. Learning with Opponent-Learning Awareness. In: PROCEEDINGS of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS. [S.l.]: International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2018. v. 1, p. 122–130. ISBN 978-1-5108-6808-3. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85055312121&partnerID=40&md5=c0ae874f64b2694b4e74077397821c7d. Citado na p. 20.

FOERSTER, Jakob et al. **Counterfactual Multi-Agent Policy Gradients**. 24 mai. 2017. arXiv: 1705.08926. Citado nas pp. 20, 42, 51.

FOERSTER, Jakob et al. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In: ADVANCES in Neural Information Processing Systems. [S.l.: s.n.], 24 mai. 2016. P. 2137–2145. Disponível em: http://arxiv.org/abs/1605.06676. Citado nas pp. 20, 33, 34.

FOERSTER, Jakob et al. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In: PROCEEDINGS of the 34th International Conference on Machine Learning - Volume 70. Sydney, NSW, Australia: JMLR.org, 6 ago. 2017. (ICML'17), p. 1146–1155. Citado nas pp. 20, 42, 51.

FOERSTER, Jakob N. et al. Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning. **CoRR**, abs/1811.01458, 4 nov. 2018. arXiv: 1811.01458v1. Citado na p. 20.

FOERSTER, Jakob N. et al. **Multi-Agent Common Knowledge Reinforcement Learning**. 27 out. 2018. arXiv: 1810.11702. Citado nas pp. 20, 42, 51.

FU, Xinyu et al. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In: PROCEEDINGS of The Web Conference 2020. New York, NY, USA: Association for Computing Machinery, 20 abr. 2020. (WWW '20), p. 2331–2341. ISBN 978-1-4503-7023-3. DOI: 10/gg6rb7. Disponível em: https://doi.org/10.1145/3366423.3380297. Acesso em: 5 mar. 2021. Citado nas pp. 21, 62.

GILMER, Justin et al. Neural Message Passing for Quantum Chemistry. In: PROCEEDINGS of the 34th International Conference on Machine Learning - Volume 70. Sydney, NSW, Australia: JMLR.org, 6 ago. 2017. (ICML'17), p. 1263–1272. Citado nas pp. 18, 37, 69.

GOLDBARG, Marco; GOLDBARG, Elizabeth. **Grafos: Conceitos, Algoritmos e Aplicações**. [S.l.]: Elsevier, 2012. Citado na p. 36.

GORI, M.; MONFARDINI, G.; SCARSELLI, F. A New Model for Learning in Graph Domains. In: PROCEEDINGS of the International Joint Conference on Neural Networks. [S.l.]: IEEE, jul. 2005. v. 2, p. 729–734. DOI: 10/cr2f33. Disponível em:

https://www.scopus.com/inward/record.uri?eid=2-s2.0-33745967023&doi=10.1109%2fIJCNN.2005.1555942&partnerID=40&md5=580a04c2dd0d808087bb56f0c6090a1a. Citado nas pp. 17, 18, 36, 37.

GRUN, Anselm. **Benedict of Nursia: His Message for Today**. [S.l.]: LITURGICAL PR, 11 fev. 2006. 63 p. ISBN 0-8146-2910-5. Disponível em: https://www.ebook.de/de/product/4908694/anselm_grun_benedict_of_nursia_his_message_for_today.html.

GUESTRIN, Carlos; KOLLER, Daphne; PARR, Ronald. Multiagent Planning with Factored MDPs. In: DIETTERICH, T.; BECKER, S.; GHAHRAMANI, Z. (Ed.). **Advances in Neural Information Processing Systems**. [S.l.]: MIT Press, 2002. v. 14. Disponível em: https://proceedings.neurips.cc/paper/2001/file/7af6266cc52234b5aa339b16695f7fc4-Paper.pdf. Citado na p. 55.

GUESTRIN, Carlos; LAGOUDAKIS, Michail; PARR, Ronald. Coordinated Reinforcement Learning. In: ICML. [S.l.]: Citeseer, 2002. v. 2, p. 227–234. Citado na p. 55.

GUESTRIN, Carlos; VENKATARAMAN, Shobha; KOLLER, Daphne. Context-Specific Multiagent Coordination and Planning with Factored MDPs. In: AAAI/IAAI. [S.l.: s.n.], 2002. P. 253–259. Citado na p. 55.

GUESTRIN, Carlos et al. Efficient Solution Algorithms for Factored MDPs. **Journal of Artificial Intelligence Research**, v. 19, p. 399–468, 1 out. 2003. ISSN 1076-9757. DOI: 10/ggb7c5. Disponível em: https://www.jair.org/index.php/jair/article/view/10341. Acesso em: 3 mar. 2021. Citado nas pp. 34, 59.

GUESTRIN, Carlos et al. Generalizing Plans to New Environments in Relational MDPs. In: Acapulco, Mexico. PROCEEDINGS of the 18th International Joint Conference on Artificial Intelligence. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003. (IJCAI'03), p. 1003–1010. Disponível em: http://dl.acm.org/citation.cfm?id=1630659.1630803. Citado na p. 55.

GUPTA, Jayesh K.; EGOROV, Maxim; KOCHENDERFER, Mykel J. Cooperative Multi-Agent Control Using Deep Reinforcement Learning. In: AUTONOMOUS Agents and Multiagent Systems. [S.l.]: Springer International Publishing, 2017. P. 66–83. DOI: 10.1007/978-3-319-71682-4_5. Citado na p. 34.

HAMILTON, William L. Graph Representation Learning. **Synthesis Lectures on Artificial Intelligence and Machine Learning**, Morgan and Claypool, v. 14, n. 3, p. 1–159, 2020. DOI: 10/ghfgfg. Disponível em: https://www.cs.mcgill.ca/~wlh/grl_book/files/GRL_Book.pdf. Citado na p. 39.

HAUSKNECHT, Matthew; STONE, Peter. Deep Recurrent Q-Learning for Partially Observable MDPs. In: AAAI FALL SYMPOSIUM - Technical Report. [S.l.]: AI Access Foundation, 2015. FS-15-06, p. 29–37. ISBN 978-1-57735-752-0. Disponível em: https://arxiv.org/abs/1507.06527. Citado nas pp. 29, 83.

HAUSKNECHT, Matthew; STONE, Peter. Deep Reinforcement Learning in Parameterized Action Space. In: PROCEEDINGS of the 4th International Conference on Learning Representations. [S.l.: s.n.], 2016. arXiv: 1511.04143v4. Citado na p. 91.

HE, He et al. Opponent Modeling in Deep Reinforcement Learning. In: INTERNATIONAL Conference ON Machine Learning. **33rd International Conference on Machine Learning, ICML 2016**. [S.l.]: PMLR, 11 jun. 2016. P. 1804–1813. ISBN 978-1-5108-2900-8. Disponível em: http://proceedings.mlr.press/v48/he16.html. Acesso em: 4 mar. 2021. Citado na p. 35.

HERNANDEZ-LEAL, Pablo; KARTAL, Bilal; TAYLOR, Matthew E. A Survey and Critique of Multiagent Deep Reinforcement Learning. **Autonomous Agents and Multi-Agent Systems**, v. 33, n. 6, p. 750–797, 1 nov. 2019. ISSN 1387-2532, 1573-7454. DOI: 10/ggbm5g. arXiv: 1810.05587. Disponível em: http://arxiv.org/abs/1810.05587. Acesso em: 11 mai. 2020. Citado na p. 33.

HINTON, Geoffrey E.; SRIVASTAVA, Nitish; SWERSKY, Kevin. Overview of Mini-Batch Gradient Descent. Neural Networks for Machine Learning. [S.l.], 2012. Disponível em: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. Citado nas pp. 75, 111.

HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. DOI: 10/bxd65w. Citado nas pp. 29, 36, 90.

HOCHREITER, Sepp et al. Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies. In: KOLEN, John F.; KREMER, Stefan C. (Ed.). **A Field Guide to Dynamical Recurrent Networks**. [S.l.]: IEEE, 2001. P. 237–243. ISBN 978-0-470-54403-7. DOI: 10.1109/9780470544037.ch14. Disponível em: https://ieeexplore.ieee.org/document/5264952. Acesso em: 9 mai. 2021. Citado na p. 90.

HONG, Huiting et al. An Attention-Based Graph Neural Network for Heterogeneous Structural Learning. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 34, n. 04, p. 4132–4139, 3 abr. 2020. ISSN 2374-3468, 2159-5399. DOI: 10/gjjct6. Disponível em: https://aaai.org/ojs/index.php/AAAI/article/view/5833. Acesso em: 5 mar. 2021. Citado nas pp. 21, 62.

HOPFIELD, John J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. **Proceedings of the national academy of sciences**, National Acad Sciences, v. 79, n. 8, p. 2554–2558, 1982. DOI: 10/c2b78f. Citado na p. 36.

HU, Jian et al. **Revisiting the Monotonicity Constraint in Cooperative Multi-Agent Reinforcement Learning**. Versão 15. 30 set. 2021. arXiv: 2102.03479 [cs]. Disponível em: http://arxiv.org/abs/2102.03479. Acesso em: 7 out. 2021. Citado nas pp. 21, 51, 60, 91.

HU, Siyi et al. **UPDeT: Universal Multi-Agent Reinforcement Learning via Policy Decoupling with Transformers**. 7 fev. 2021. arXiv: 2101.08001 [cs]. Disponível em: http://arxiv.org/abs/2101.08001. Acesso em: 12 fev. 2021. Citado nas pp. 35, 51, 110, 111.

HU, Yue et al. Knowledge-Guided Agent-Tactic-Aware Learning for StarCraft Micromanagement. In: LANG, J. (Ed.). **Proceedings of the Twenty-Seventh International**

Joint Conference on Artificial Intelligence. [S.l.]: International Joint Conferences on Artificial Intelligence Organization, jul. 2018. 2018-July, p. 1471–1477. DOI: 10/ghmjbb. Disponível em: https://www.ijcai.org/proceedings/2018/204. Citado na p. 42.

HU, Ziniu et al. Heterogeneous Graph Transformer. In: WWW '20: THE Web Conference 2020. **Proceedings of The Web Conference 2020**. Taipei Taiwan: ACM, 20 abr. 2020. P. 2704–2710. ISBN 978-1-4503-7023-3. DOI: 10/gg8f23. Disponível em: https://dl.acm.org/doi/10.1145/3366423.3380027. Acesso em: 5 mar. 2021. Citado nas pp. 21, 62, 63.

HUANG, Chao; LIU, Rui. **Robot Inner Attention Modeling for Task-Adaptive Teaming of Heterogeneous Multi Robots**. 14 mar. 2021. arXiv: 2006.15482 [cs]. Disponível em: http://arxiv.org/abs/2006.15482. Acesso em: 16 abr. 2021. Citado na p. 59.

JAIDEE, Ulit; MUNOZ-AVILA, Hector. Modeling Unit Classes as Agents in Real-Time Strategy Games. **Proceedings of the 9th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment**, v. 9, n. 1, p. 149–155, nov. 2013. Disponível em: https://ojs.aaai.org/index.php/AIIDE/article/view/12667. Citado na p. 47.

JIANG, Jiechuan et al. Graph Convolutional Reinforcement Learning. In: INTERNATIONAL Conference ON Learning Representations. **Proceedings of the 8th International Conference on Learning Representations, ICLR 2020**. [S.l.: s.n.], 2020. arXiv: 1810.09202v3. Disponível em: https://arxiv.org/abs/1810.09202. Citado nas pp. 56, 70, 74, 77, 79, 84.

JORDAN, Michael I. Serial Order: A Parallel Distributed Processing Approach. In: ADVANCES in Psychology. [S.l.]: Elsevier, 1997. v. 121. P. 471–495. ISBN 978-0-444-81931-4. DOI: 10.1016/S0166-4115(97)80111-2. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S0166411597801112. Acesso em: 22 nov. 2020. Citado na p. 36.

JUSTESEN, Niels et al. Deep Learning for Video Game Playing. **IEEE Transactions on Games**, p. 17, 2019. DOI: 10/ghm4rc. arXiv: 1708.07902. Citado na p. 44.

KAELBLING, Leslie Pack; LITTMAN, Michael L.; CASSANDRA, Anthony R. Planning and Acting in Partially Observable Stochastic Domains. **Artificial Intelligence**, v. 101, n. 1-2, p. 99–134, mai. 1998. ISSN 00043702. DOI: 10/drssvh. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S000437029800023X. Acesso em: 20 nov. 2020. Citado na p. 25.

KAPETANAKIS, Spiros; KUDENKO, Daniel. Reinforcement Learning of Coordination in Heterogeneous Cooperative Multi-Agent Systems. In: KUDENKO, Daniel; KAZAKOV, Dimitar; ALONSO, Eduardo (Ed.). **Adaptive Agents and Multi-Agent Systems II**. Berlin, Heidelberg: Springer, 2005. (Lecture Notes in Computer Science), p. 119–131. ISBN 978-3-540-32274-0. DOI: 10/dg5g7s. Citado na p. 55.

KINGMA, Diederik P.; BA, Jimmy. Adam: A Method for Stochastic Optimization. In: 3RD International Conference FOR Learning Representations, p. 15. arXiv: 1412.6980. Disponível em: http://arxiv.org/abs/1412.6980. Acesso em: 12 abr. 2021. Citado na p. 111.

KIPF, Thomas N.; WELLING, Max. Semi-Supervised Classification with Graph Convolutional Networks. In: 5TH International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings. [S.l.]: International Conference on Learning Representations, ICLR, 22 fev. 2017. arXiv: 1609.02907. Disponível em: http://arxiv.org/abs/1609.02907. Acesso em: 10 fev. 2020. Citado nas pp. 18, 36, 62.

KITANO, Hiroaki et al. RoboCup: A Challenge Problem for AI. **AI Magazine**, v. 18, n. 1, p. 73–73, 1 15 mar. 1997. ISSN 2371-9621. DOI: 10/gj2zgf. Disponível em: https://ojs.aaai.org/index.php/aimagazine/article/view/1276. Acesso em: 5 mai. 2021. Citado na p. 17.

KOLLER, Daphne; PARR, Ronald. Computing Factored Value Functions for Policies in Structured MDPs. In: INTERNATIONAL Joint Conference ON Artificial Intelligence. **Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)**. Stockholm, Sweden: [s.n.], 1999. v. 2, p. 1332–1339. Citado nas pp. 34, 59.

KONDA, Vijay R.; TSITSIKLIS, John N. Actor-Critic Algorithms. **SIAM Journal on Control and Optimization**, v. 42, n. 4, p. 1143–1166, jan. 2003–1. DOI: 10/fvtk5m. Citado na p. 26.

KRISHNA, Ranjay et al. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. **International Journal of Computer Vision**, v. 123, n. 1, p. 32–73, 1 mai. 2017. ISSN 1573-1405. DOI: 10/f96kc4. Disponível em: https://doi.org/10.1007/s11263-016-0981-7. Acesso em: 2 mai. 2021. Citado na p. 40.

LANCTOT, Marc et al. **OpenSpiel: A Framework for Reinforcement Learning in Games**. 26 set. 2020. arXiv: 1908.09453 [cs]. Disponível em: http://arxiv.org/abs/1908.09453. Acesso em: 29 set. 2020. Citado na p. 72.

LECUN, Y. et al. Backpropagation Applied to Handwritten Zip Code Recognition. **Neural Computation**, v. 1, n. 4, p. 541–551, dez. 1989. ISSN 0899-7667, 1530-888X. DOI: 10/bknd8g. Disponível em: https://www.mitpressjournals.org/doi/abs/10.1162/neco.1989.1.4.541. Acesso em: 22 nov. 2020. Citado na p. 36.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep Learning. **Nature**, v. 521, n. 7553, p. 436–444, 27 mai. 2015. ISSN 0028-0836. DOI: 10/bmqp. Citado na p. 17.

LECUN, Yann et al. Gradient-Based Learning Applied to Document Recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, nov. 1998. ISSN 0018-9219. DOI: 10/d89c25. Disponível em: http://ieeexplore.ieee.org/document/726791/. Acesso em: 1 fev. 2017. Citado na p. 36.

LI, Sheng et al. **Deep Implicit Coordination Graphs for Multi-Agent Reinforcement Learning**. 19 jun. 2020. arXiv: 2006.11438 [cs]. Disponível em: http://arxiv.org/abs/2006.11438. Acesso em: 13 out. 2020. Citado na p. 51.

LIN, Long-Ji. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. **Machine Learning**, v. 8, n. 3-4, p. 293–321, 1992. ISSN 0885-6125. DOI: 10/dgbqdc. Citado nas pp. 26, 28.

LIN, Zhouhan et al. A Structured Self-Attentive Sentence Embedding. In: PROCEEDINGS of the 5th International Conference on Learning Representations (ICLR 2017). [S.l.: s.n.], 9 mar. 2017. arXiv: 1703.03130v1. Citado na p. 62.

LOWE, Ryan et al. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In: GUYON, I. et al. (Ed.). **Advances in Neural Information Processing Systems 30**. [S.l.]: Curran Associates, Inc., 2017. P. 6379–6390. Disponível em: http://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments.pdf. Citado na p. 35.

MA, Hao et al. Recommender Systems with Social Regularization. In: PROCEEDINGS of the Fourth ACM International Conference on Web Search and Data Mining. Hong Kong, China: [s.n.], 2011. (WSDM '11), p. 287–296. 10 p. DOI: 10/cj4ntw. Citado na p. 40.

MAAS, Andrew L.; HANNUN, Awni Y.; NG, Andrew Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In: IN ICML Workshop on Deep Learning for Audio, Speech and Language Processing. [S.l.: s.n.], 2013. Citado nas pp. 71, 90.

MAHAJAN, Anuj et al. MAVEN: Multi-Agent Variational Exploration. In: WALLACH, H. et al. (Ed.). **Advances in Neural Information Processing Systems**. [S.l.]: Curran Associates, Inc., 2019. v. 32. Disponível em: https://proceedings.neurips.cc/paper/2019/file/f816dc0acface7498e10496222e9db10-Paper.pdf. Citado na p. 59.

MALYSHEVA, Aleksandra; KUDENKO, Daniel; SHPILMAN, Aleksei. MAGNet: Multi-Agent Graph Network for Deep Multi-Agent Reinforcement Learning. In: ADAPTIVE AND Learning Agents Workshop AT AAMAS (ALA 2019), Montreal, Canada. DOI: 10/gj2zg9. Disponível em: https://ala2019.vub.ac.be/papers/ALA2019_paper_17.pdf. Citado na p. 18.

MASSON, Warwick; RANCHOD, Pravesh; KONIDARIS, George. Reinforcement Learning with Parameterized Actions. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 30, n. 1, 1 21 fev. 2016. ISSN 2374-3468. arXiv: 1509.01644. Disponível em: https://ojs.aaai.org/index.php/AAAI/article/view/10226. Acesso em: 24 mar. 2021. Citado na p. 57.

MATIGNON, Laetitia; LAURENT, Guillaume J.; LE FORT-PIAT, Nadine. Independent Reinforcement Learners in Cooperative Markov Games: A Survey Regarding Coordination Problems. **The Knowledge Engineering Review**, v. 27, n. 1, p. 1–31, 22 fev. 2012. DOI: 10/ggd3xp. Disponível em: https://www.cambridge.org/core/product/identifier/S0269888912000057/type/journal_article. Acesso em: 2 nov. 2020. Citado nas pp. 33, 76.

MENEGHETTI, Douglas De Rizzo; AQUINO JUNIOR, Plinio Thomaz. **Application and Simulation of Computerized Adaptive Tests Through the Package Catsim**. 19 jul. 2018. arXiv: 1707.03012 [stat]. Disponível em: http://arxiv.org/abs/1707.03012. Acesso em: 22 abr. 2021. Citado na p. 132.

MENEGHETTI, Douglas De Rizzo; BIANCHI, Reinaldo Augusto da Costa. Specializing Inter-Agent Communication in Heterogeneous Multi-Agent Reinforcement Learning Using Agent Class Information. In: 35TH AAAI Conference ON Artificial Intelligence (AAAI-21).

AAAI-21 Workshop on Reinforcement Learning in Games. Virtual Conference: [s.n.], 8 fev. 2021. P. 12. Disponível em: https://arxiv.org/abs/2012.07617. Citado nas pp. 19, 81, 110, 133.

MENEGHETTI, Douglas De Rizzo; BIANCHI, Reinaldo Augusto da Costa. Towards Heterogeneous Multi-Agent Reinforcement Learning with Graph Neural Networks. In: XVI Encontro Nacional DE Inteligência Artificial E Computacional (ENIAC). **Anais Do Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)**. Porto Alegre, RS, Brasil: SBC, 20 out. 2020. P. 579–590. DOI: 10.5753/eniac.2020.12161. Disponível em: https://sol.sbc.org.br/index.php/eniac/article/view/12161. Citado nas pp. 19, 110.

MENEGHETTI, Douglas De Rizzo; BIANCHI, Reinaldo Augusto da Costa. Towards Heterogeneous Multi-Agent Reinforcement Learning with Graph Neural Networks. XVII Encontro Nacional de Inteligência Artificial e Computacional (Porto Alegre, RS, Brasil). [S.l.], 21 out. 2020. Disponível em: https://underline.io/events/33/sessions/666/lecture/3261-towards-heterogeneous-multi-agent-reinforcement-learning-with-graph-neural-networks. Citado na p. 132.

MENEGHETTI, Douglas De Rizzo et al. **Annotated Image Dataset of Household Objects from the RoboFEI@Home Team**. [S.l.]: IEEE Dataport, 2020. DOI: 10/ghdk9h. Disponível em: http://dx.doi.org/10.21227/7wxn-n828. Citado nas pp. 18, 64, 133.

MENEGHETTI, Douglas De Rizzo et al. Detecting Soccer Balls with Reduced Neural Networks: A Comparison of Multiple Architectures under Constrained Hardware Scenarios. **Journal of Intelligent & Robotic Systems**, v. 101, n. 3, p. 53, mar. 2021. ISSN 0921-0296, 1573-0409. DOI: 10/gj2zhb. arXiv: 2009.13684. Disponível em: http://link.springer.com/10.1007/s10846-021-01336-y. Citado na p. 132.

MIRHOSEINI, Azalia et al. **Chip Placement with Deep Reinforcement Learning**. 22 abr. 2020. arXiv: 2004.10746 [cs]. Disponível em: http://arxiv.org/abs/2004.10746. Acesso em: 5 mai. 2021. Citado na p. 18.

MITCHELL, Tom M. **The Need for Biases in Learning Generalizations**. [S.l.], 1980. Disponível em: https://core.ac.uk/display/21438353. Acesso em: 22 nov. 2020. Citado na p. 36.

MNIH, Volodymyr et al. Asynchronous Methods for Deep Reinforcement Learning. In: INTERNATIONAL Conference on Machine Learning. [S.l.: s.n.], 2016. Disponível em: http://www.jmlr.org/proceedings/papers/v48/mniha16.pdf. Citado na p. 26.

MNIH, Volodymyr et al. Human-Level Control through Deep Reinforcement Learning. **Nature**, v. 518, n. 7540, p. 529–533, 25 fev. 2015. ISSN 0028-0836. DOI: 10/gc3h75. Disponível em: https://www.nature.com/articles/nature14236. Citado nas pp. 26–29, 71.

MORDATCH, Igor; ABBEEL, Pieter. Emergence of Grounded Compositional Language in Multi-Agent Populations. **Proceedings of the AAAI Conference on Artificial Intelligence**, v. 32, n. 1, 25 abr. 2018. ISSN 2374-3468. Disponível em: https://ojs.aaai.org/index.php/AAAI/article/view/11492. Acesso em: 14 mai. 2021. Citado na p. 34.

NADERIALIZADEH, Navid et al. **Graph Convolutional Value Decomposition in Multi-Agent Reinforcement Learning**. 10 fev. 2021. arXiv: 2010.04740 [cs]. Disponível em: http://arxiv.org/abs/2010.04740. Acesso em: 17 fev. 2021. Citado na p. 60.

NAIR, L.; CHERNOVA, S. Action Categorization for Computationally Improved Task Learning and Planning. In: INTERNATIONAL Joint Conference ON Autonomous Agents AND Multiagent Systems. **Proceedings of the 17th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2018**. Stockholm, Sweden: International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2018. v. 3, p. 2022–2024. ISBN 978-1-5108-6808-3. Disponível em: http://ifaamas.org/Proceedings/aamas2018/pdfs/p2022.pdf. Citado na p. 42.

OLIEHOEK, Frans A.; AMATO, Christopher. **A Concise Introduction to Decentralized POMDPs**. Cham: Springer International Publishing, 2016. (SpringerBriefs in Intelligent Systems). ISBN 978-3-319-28929-8. DOI: 10.1007/978-3-319-28929-8. Disponível em: http://link.springer.com/10.1007/978-3-319-28929-8. Acesso em: 20 set. 2020. Citado nas pp. 51, 79, 83.

ONTAÑÓN, Santiago. The Combinatorial Multi-Armed Bandit Problem and Its Application to Real-Time Strategy Games. In: AAAI Conference ON Artificial Intelligence AND Interactive Digital Entertainment, Boston, MA, USA. **Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment**. [S.l.]: AAAI Press, 2014. P. 58–64. ISBN 978-1-57735-607-3. Disponível em: http://dl.acm.org/citation.cfm?id=3014712.3014722. Citado nas pp. 42, 43, 47.

ORTEGA Y GASSET, José. **Meditaciones Del Quijote: Meditación Preliminar, Meditación Primera**. [S.l.]: Residencia de Estudiantes, 1914. Disponível em: https://books.google.com.br/books?id=ohku8iIR77EC.

PASZKE, Adam et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: WALLACH, H. et al. (Ed.). **Advances in Neural Information Processing Systems 32**. [S.l.]: Curran Associates, Inc., 2019. P. 8024–8035. Disponível em: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. Citado na p. 74.

PENG, Peng et al. **Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-Level Coordination in Learning to Play StarCraft Combat Games**. 29 mar. 2017. arXiv: 1703.10069. Citado na p. 34.

PESCE, Emanuele; MONTANA, Giovanni. Improving Coordination in Small-Scale Multi-Agent Deep Reinforcement Learning through Memory-Driven Communication. **Machine Learning**, v. 109, n. 9, p. 1727–1747, 1 set. 2020. ISSN 1573-0565. DOI: 10/ghdpj4. Disponível em: https://doi.org/10.1007/s10994-019-05864-5. Acesso em: 4 out. 2020. Citado na p. 34.

PUTERMAN, Martin L. **Markov Decision Processes: Discrete Stochastic Dynamic Programming**. 1. ed. New York, NY, USA: Wiley John + Sons, 25 fev. 2005. 680 p. ISBN 0-471-72782-2. DOI: 10.2307/2291177. Disponível em: https://www.ebook.de/de/product/3595119/martin_l_puterman_markov_decision_processes.html. Citado nas pp. 22, 23.

PYNADATH, D. V.; TAMBE, M. The Communicative Multiagent Team Decision Problem: Analyzing Teamwork Theories and Models. **Journal of Artificial Intelligence Research**, v. 16, p. 389–423, 1 jun. 2002. ISSN 1076-9757. DOI: 10/ghkqps. Disponível em: https://www.jair.org/index.php/jair/article/view/10304. Acesso em: 18 nov. 2020. Citado nas pp. 31, 33.

RABINOWITZ, Neil et al. Machine Theory of Mind. In: INTERNATIONAL Conference ON Machine Learning. **International Conference on Machine Learning**. [S.l.]: PMLR, 3 jul. 2018. P. 4218–4227. Disponível em: http://proceedings.mlr.press/v80/rabinowitz18a.html. Acesso em: 4 mar. 2021. Citado na p. 35.

RAHMAN, Arrasy et al. **Open Ad Hoc Teamwork Using Graph-Based Policy Learning**. 16 out. 2020. arXiv: 2006.10412 [cs, stat]. Disponível em: http://arxiv.org/abs/2006.10412. Acesso em: 22 mar. 2021. Citado nas pp. 21, 57, 79, 80.

RASHID, Tabish et al. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. **Journal of Machine Learning Research**, v. 21, n. 178, p. 1–51, 2020. Disponível em: http://jmlr.org/papers/v21/20-081.html. Acesso em: 3 mar. 2021. Citado nas pp. 20, 34, 35, 51, 59, 60, 93.

RASHID, Tabish et al. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In: DY, Jennifer; KRAUSE, Andreas (Ed.). **Proceedings of the 35th International Conference on Machine Learning**. [S.l.]: PMLR, 10–15 jul. 2018. v. 80. (Proceedings of Machine Learning Research), p. 4295–4304. Disponível em: http://proceedings.mlr.press/v80/rashid18a.html. Citado nas pp. 17, 20, 34, 51, 59, 60, 84.

RASHID, Tabish et al. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In: LAROCHELLE, H. et al. (Ed.). **Advances in Neural Information Processing Systems**. [S.l.]: Curran Associates, Inc., 2020. v. 33, p. 10199–10210. Disponível em: https://proceedings.neurips.cc/paper/2020/file/73a427badebe0e32caa2e1fc7530b7f3-Paper.pdf. Citado na p. 59.

RUMMERY, Gavin A.; NIRANJAN, Mahesan. **On-Line Q-Learning Using Connectionist Systems**. Cambridge, England, set. 1994. Citado na p. 24.

SAMVELYAN, Mikayel et al. **The StarCraft Multi-Agent Challenge**. 9 dez. 2019. arXiv: 1902.04043 [cs, stat]. Disponível em: http://arxiv.org/abs/1902.04043. Acesso em: 5 fev. 2020. Citado nas pp. 20, 47, 51, 93, 97, 103, 104.

SCARSELLI, F. et al. Computational Capabilities of Graph Neural Networks. **IEEE Transactions on Neural Networks**, v. 20, n. 1, p. 81–102, jan. 2009. ISSN 1045-9227. DOI: 10/cj9mn9. Disponível em: https://ieeexplore.ieee.org/abstract/document/4703190. Citado nas pp. 18, 36, 37.

SCARSELLI, F. et al. The Graph Neural Network Model. **IEEE Transactions on Neural Networks**, v. 20, n. 1, p. 61–80, jan. 2009. ISSN 1045-9227. DOI: 10/fgf4kh. Disponível em: https://ieeexplore.ieee.org/document/4700287. Citado nas pp. 18, 36, 37.

SCHAUL, T. et al. Prioritized Experience Replay. In: 4TH International Conference ON Learning Representations, ICLR 2016 - Conference Track Proceedings. arXiv: 1511.05952. Disponível em: https://arxiv.org/abs/1511.05952. Citado nas pp. 28, 72.

SCHLICHTKRULL, Michael et al. Modeling Relational Data with Graph Convolutional Networks. In: GANGEMI, Aldo et al. (Ed.). **European Semantic Web Conference**. Cham: Springer International Publishing, 2018. Springer, p. 593–607. ISBN 978-3-319-93417-4. DOI: 10/gfthqc. arXiv: 1703.06103. Citado nas pp. 19, 21, 62, 63, 69, 88.

SCHULMAN, John et al. Proximal Policy Optimization Algorithms, 20 jul. 2017. Disponível em: https://arxiv.org/abs/1707.06347. Citado na p. 26.

SCHULMAN, John et al. Trust Region Policy Optimization. In: BACH, Francis; BLEI, David (Ed.). **Proceedings of the 32nd International Conference on Machine Learning**. Lille, France: PMLR, 7–9 jul. 2015. v. 37. (Proceedings of Machine Learning Research), p. 1889–1897. Disponível em: http://proceedings.mlr.press/v37/schulman15.html. Citado na p. 26.

SHAPLEY, L. S. Stochastic Games. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 39, n. 10, p. 1095–1100, 1 out. 1953. ISSN 0027-8424, 1091-6490. DOI: 10/c9795q. eprint: https://www.pnas.org/content/39/10/1095.full.pdf. Disponível em: https://www.pnas.org/content/39/10/1095. Acesso em: 14 mai. 2021. Citado na p. 30.

SILVER, David. Lecture 3: Planning by Dynamic Programming (London, UK). [S.l.], 2015. Disponível em: http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html. Acesso em: 10 fev. 2019. Citado na p. 23.

SON, Kyunghwan et al. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In: INTERNATIONAL Conference ON Machine Learning. **International Conference on Machine Learning**. [S.l.]: PMLR, 24 mai. 2019. P. 5887–5896. Disponível em: http://proceedings.mlr.press/v97/son19a.html. Acesso em: 5 out. 2020. Citado na p. 59.

STOKES, Jonathan M. et al. A Deep Learning Approach to Antibiotic Discovery. **Cell**, v. 180, n. 4, 688–702.e13, 20 fev. 2020. ISSN 00928674. DOI: 10/ggk84g. Disponível em: https://www.cell.com/cell/abstract/S0092-8674(20)30102-1. Acesso em: 5 mai. 2021. Citado na p. 17.

STONE, Peter et al. Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination. In: 24TH AAAI Conference ON Artificial Intelligence. **Proceedings of the 24th AAAI Conference on Artificial Intelligence**. [S.l.]: AAAI Press, 2010. v. 24. Citado nas pp. 56, 111.

SUKHBAATAR, Sainbayar; SZLAM, Arthur; FERGUS, Rob. Learning Multiagent Communication with Backpropagation. In: LEE, D. D. et al. (Ed.). **Advances in Neural Information Processing Systems 29**. [S.l.]: Curran Associates, Inc., 2016. P. 2244–2252. Disponível em: http://papers.nips.cc/paper/6398-learning-multiagent-communication-with-backpropagation. Citado na p. 34.

SUN, Chuangchuang; LI, Xiao; BELTA, Calin. Automata Guided Semi-Decentralized Multi-Agent Reinforcement Learning. In: 2020 American Control Conference (ACC). **2020 American Control Conference (ACC)**. Denver, CO, USA: IEEE, jul. 2020. P. 3900–3905. ISBN 978-1-5386-8266-1. DOI: 10/gjqkvx. Disponível em: https://ieeexplore.ieee.org/document/9147704/. Acesso em: 14 mai. 2021. Citado na p. 59.

SUN, Yizhou et al. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. **Proceedings of the VLDB Endowment**, v. 4, n. 11, p. 992–1003, ago. 2011. ISSN 2150-8097. DOI: 10/ghmgq5. Disponível em: https://dl.acm.org/doi/10.14778/3402707.3402736. Acesso em: 14 mai. 2021. Citado nas pp. 39, 61.

SUNEHAG, Peter et al. Value-Decomposition Networks for Cooperative Multi-Agent Learning Based on Team Reward. In: PROCEEDINGS of the 17th International Conference on Autonomous Agents and MultiAgent Systems. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018. (AAMAS '18), p. 2085–2087. arXiv: 1706.05296v1. Citado nas pp. 20, 34, 59, 60, 84, 90.

SUTTON, Richard S.; BARTO, Andrew G. **Reinforcement Learning: An Introduction**. 2. ed. Cambridge, Mass: The MIT Press, 13 nov. 2018. 552 p. ISBN 978-0-262-03924-6. Disponível em: https://www.ebook.de/de/product/32966850/richard_s_sutton_andrew_g_barto_reinforcement_learning.html. Citado nas pp. 23, 24, 28.

SZEPESVÁRI, Csaba. Algorithms for Reinforcement Learning. **Synthesis Lectures on Artificial Intelligence and Machine Learning**, v. 4, n. 1, p. 1–103, jan. 2010. ISSN 1939-4608. DOI: 10/dxpsnh. Citado na p. 24.

TAMPUU, Ardi et al. Multiagent Cooperation and Competition with Deep Reinforcement Learning. Edição: Cheng-Yi Xia. **PLOS ONE**, v. 12, n. 4, e0172395, 5 abr. 2017. ISSN 1932-6203. DOI: 10/f9xpcd. Disponível em: https://dx.plos.org/10.1371/journal.pone.0172395. Acesso em: 14 mai. 2021. Citado na p. 34.

TAN, Ming. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In: MACHINE Learning Proceedings 1993. [S.l.]: Elsevier, 1993. P. 330–337. DOI: 10.1016/b978-1-55860-307-3.50049-6. Citado nas pp. 33, 59, 68, 90.

TANG, Jie. AMiner: Toward Understanding Big Scholar Data. In: PROCEEDINGS of the Ninth ACM International Conference on Web Search and Data Mining. New York, NY, USA: Association for Computing Machinery, 8 fev. 2016. (WSDM '16), p. 467. ISBN 978-1-4503-3716-8. DOI: 10/gj7jgh. Disponível em: https://doi.org/10.1145/2835776.2835849. Acesso em: 26 mai. 2021. Citado na p. 40.

TERRY, Justin K. et al. Revisiting Parameter Sharing in Multi-Agent Deep Reinforcement Learning. In: ACCEPTED FOR PRESENTATION AT International Conference ON Learning Representations (ICLR 2021). arXiv: 2005.13625. Disponível em: https://openreview.net/forum?id=MWj_P-Lk3jC. Acesso em: 4 out. 2020. Citado nas pp. 32, 84.

TIAN, Yuandong et al. ELF: An Extensive, Lightweight and Flexible Research Platform for Real-Time Strategy Games. In: ADVANCES in Neural Information Processing Systems. [S.l.]:

Neural information processing systems foundation, 2017. 2017-December. (Advances in Neural Information Processing Systems), p. 2656–2666. Disponível em: http://papers.nips.cc/paper/6859-elf-an-extensive-lightweight-and-flexible-research-platform-for-real-time-strategy-games. Citado na p. 42.

TOUTANOVA, Kristina; CHEN, Danqi. Observed versus Latent Features for Knowledge Base and Text Inference. In: 3RD Workshop on Continuous Vector Space Models and Their Compositionality. [S.l.]: ACL - Association for Computational Linguistics, jul. 2015. Disponível em: https://www.microsoft.com/en-us/research/publication/observed-versus-latent-features-for-knowledge-base-and-text-inference/. Citado na p. 40.

TUKEY, John W. Comparing Individual Means in the Analysis of Variance. **Biometrics**, v. 5, n. 2, p. 99, jun. 1949. ISSN 0006341X. DOI: 10/bb6f3d. JSTOR: 3001913. Citado na p. 97.

USUNIER, Nicolas et al. Episodic Exploration for Deep Deterministic Policies: An Application to StarCraft Micromanagement Tasks. In: INTERNATIONAL Conference on Learning Representations (ICLR) 2017. [S.l.: s.n.], 2017. P. 16. Citado na p. 42.

VAN HASSELT, Hado. Double Q-Learning. In: LAFFERTY, J. D. et al. (Ed.). **Advances in Neural Information Processing Systems 23**. [S.l.]: Curran Associates, Inc., 2010. P. 2613–2621. Disponível em: http://papers.nips.cc/paper/3964-double-q-learning.pdf. Citado na p. 29.

VAN HASSELT, Hado; GUEZ, Arthur; SILVER, David. Deep Reinforcement Learning with Double Q-Learning. In: AAAI. [S.l.: s.n.], 2016. P. 2094–2100. Disponível em: http://www.aaai.org/Conferences/AAAI/2016/Papers/12vanHasselt12389.pdf. Acesso em: 17 fev. 2017. Citado na p. 87.

VAN HASSELT, Hado P. et al. Learning Values across Many Orders of Magnitude. In: ADVANCES in Neural Information Processing Systems. [S.l.: s.n.], 2016. P. 4287–4295. Disponível em: http://papers.nips.cc/paper/6076-learning-values-across-many-orders-of-magnitude. Citado nas pp. 29, 83.

VASWANI, Ashish et al. Attention Is All You Need. In: GUYON, I. et al. (Ed.). **Advances in Neural Information Processing Systems 30**. [S.l.]: Curran Associates, Inc., 2017. P. 5998–6008. Disponível em: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf. Citado nas pp. 56, 63, 70.

VELIČKOVIĆ, Petar (director). **Theoretical Foundations of Graph Neural Networks**. [S.l.: s.n.], 22 fev. 2021. Disponível em: https://www.youtube.com/watch?v=uF53xsT7mjc. Acesso em: 3 out. 2021. Citado na p. 38.

VELIČKOVIĆ, Petar et al. Graph Attention Networks. In: 6TH International Conference ON Learning Representations. **ICLR 2018 - Conference Track Proceedings**. [S.l.: s.n.], 4 fev. 2018. arXiv: 1710.10903. Disponível em: http://arxiv.org/abs/1710.10903. Acesso em: 14 mai. 2021. Citado nas pp. 62, 70, 76, 88.

VIEILLARD, Nino; PIETQUIN, Olivier; GEIST, Matthieu. Munchausen Reinforcement Learning. In: LAROCHELLE, H. et al. (Ed.). **Advances in Neural Information Processing**

Systems. [S.l.]: Curran Associates, Inc., 2020. v. 33, p. 4235–4246. Disponível em: https://proceedings.neurips.cc/paper/2020/file/2c6a0bae0f071cbbf0bb3d5b11d90a82-Paper.pdf. Citado na p. 111.

VINYALS, Oriol et al. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. **Nature**, out. 2019. DOI: 10/ggb9jx. Citado nas pp. 20, 44, 49.

VINYALS, Oriol et al. **StarCraft II: A New Challenge for Reinforcement Learning**. 16 ago. 2017. arXiv: 1708.04782. Disponível em: https://arxiv.org/abs/1708.04782. Citado nas pp. 42, 47, 49.

VLASSIS, Nikos. A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence. **Synthesis Lectures on Artificial Intelligence and Machine Learning**, v. 1, n. 1, p. 1–71, jan. 2007. ISSN 1939-4608. DOI: 10/dc5c4d. Disponível em: http://www.morganclaypool.com/doi/abs/10.2200/S00091ED1V01Y200705AIM002. Acesso em: 14 mai. 2021. Citado na p. 31.

WANG, Jianhao et al. **QPLEX: Duplex Dueling Multi-Agent Q-Learning**. 5 out. 2020. arXiv: 2008.01062 [cs, stat]. Disponível em: http://arxiv.org/abs/2008.01062. Acesso em: 20 abr. 2021. Citado na p. 59.

WANG, Tingwu et al. Nervenet: Learning Structured Policy with Graph Neural Networks. In: INTERNATIONAL Conference ON Learning Representations. **6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings**. [S.l.: s.n.], 2018. Disponível em: https://openreview.net/forum?id=S1sqHMZCb. Citado nas pp. 56, 70.

WANG, Tonghan et al. **RODE: Learning Roles to Decompose Multi-Agent Tasks**. 4 out. 2020. arXiv: 2010.01523 [cs, stat]. Disponível em: http://arxiv.org/abs/2010.01523. Acesso em: 17 fev. 2021. Citado nas pp. 17, 21, 60.

WANG, Tonghan et al. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. In: INTERNATIONAL Conference ON Machine Learning. **International Conference on Machine Learning**. [S.l.]: PMLR, 21 nov. 2020. P. 9876–9886. Disponível em: http://proceedings.mlr.press/v119/wang20f.html. Acesso em: 24 mar. 2021. Citado nas pp. 20, 59.

WANG, Xiao et al. **A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources**. 30 nov. 2020. arXiv: 2011.14867 [cs]. Disponível em: http://arxiv.org/abs/2011.14867. Acesso em: 8 dez. 2020. Citado nas pp. 18, 61, 69.

WANG, Xiao et al. Heterogeneous Graph Attention Network. In: WWW '19: THE Web Conference. **The World Wide Web Conference**. San Francisco CA USA: ACM, 13 mai. 2019. P. 2022–2032. ISBN 978-1-4503-6674-8. DOI: 10/ggnsps. arXiv: 1903.07293. Disponível em: https://dl.acm.org/doi/10.1145/3308558.3313562. Acesso em: 14 mai. 2021. Citado nas pp. 21, 62.

WANG, Yihan et al. **Off-Policy Multi-Agent Decomposed Policy Gradients**. 4 out. 2020. arXiv: 2007.12322 [cs, stat]. Disponível em: http://arxiv.org/abs/2007.12322. Acesso em: 20 abr. 2021. Citado na p. 59.

WANG, Ziyu et al. Dueling Network Architectures for Deep Reinforcement Learning. In: INTERNATIONAL Conference on Machine Learning. [S.l.: s.n.], 2016. P. 1995–2003. Citado nas pp. 30, 83.

WARGUS. **Wargus**. 29 set. 2018. Disponível em: https://wargus.github.io/. Acesso em: 29 abr. 2019. Citado na p. 43.

WASSER, Carlos Gregorio Diuk; COHEN, Andre; LITTMAN, Michael L. An Object-Oriented Representation for Efficient Reinforcement Learning. In: PROCEEDINGS of the 25th International Conference on Machine Learning. [S.l.]: ACM, 2008. P. 240–247. DOI: 10/fhfkgf. Citado na p. 47.

WATKINS, Christopher J. C. H.; DAYAN, Peter. Q-Learning. **Machine Learning**, v. 8, n. 3-4, p. 279–292, 1 mai. 1992. ISSN 1573-0565. DOI: 10/dvm8f4. Citado na p. 24.

WERBOS, Paul J. Generalization of Backpropagation with Application to a Recurrent Gas Market Model, 1 jan. 1988. DOI: 10/dcbws7. Disponível em: https://zenodo.org/record/1258627. Acesso em: 21 out. 2020. Citado na p. 87.

WHITESON, Shimon. Factored Value Functions for Cooperative Multi-Agent Reinforcement Learning. 6th Workshop on Distributed and Multi-Agent Planning (DMAP) (Virtual). [S.l.], 5 nov. 2020. Disponível em: https://www.youtube.com/watch?v=EsCwDYtBZ8M. Acesso em: 3 mar. 2021. Citado nas pp. 35, 76, 108.

WIGGERS, Auke J; OLIEHOEK, Frans A; ROIJERS, Diederik M. Structure in the Value Function of Zero-Sum Games of Incomplete Information. In: 10TH Annual Workshop ON Multiagent Sequential Decision Making Under Uncertainty (MSDM-2015), p. 9. Citado na p. 30.

WILLIAMS, Ronald J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. **Machine Learning**, v. 8, n. 3-4, p. 229–256, mai. 1992. DOI: 10/cwkvww. Citado na p. 87.

WITTE, Robert S; WITTE, John S. **Statistics**. 11. ed. [S.l.]: Wiley, dez. 2016. 496 p. ISBN 978-1-119-29916-5. Citado nas pp. 97, 101.

YANG, Shantian et al. IHG-MA: Inductive Heterogeneous Graph Multi-Agent Reinforcement Learning for Multi-Intersection Traffic Signal Control. **Neural Networks**, v. 139, p. 265–277, 1 jul. 2021. ISSN 0893-6080. DOI: 10/gjqkvv. Disponível em: https://www.sciencedirect.com/science/article/pii/S0893608021000952. Acesso em: 16 abr. 2021. Citado nas pp. 17, 57.

YANG, Yaodong et al. **Qatten: A General Framework for Cooperative Multiagent Reinforcement Learning**. 9 jun. 2020. arXiv: 2002.03939 [cs]. Disponível em: http://arxiv.org/abs/2002.03939. Acesso em: 20 abr. 2021. Citado na p. 59.

YUN, Seongjun et al. Graph Transformer Networks. **Advances in Neural Information Processing Systems**, v. 32, 2019. Disponível em: https://papers.nips.cc/paper/2019/hash/9d63484abb477c97640154d40595a3bb-Abstract.html. Acesso em: 2 abr. 2021. Citado nas pp. 21, 62.

ZAMBALDI, Vinicius et al. Deep Reinforcement Learning with Relational Inductive Biases. In: INTERNATIONAL Conference on Learning Representations. [S.l.: s.n.], 2019. Disponível em: https://openreview.net/forum?id=HkxaFoC9KQ. Citado na p. 42.

ZHANG, Chuxu et al. Heterogeneous Graph Neural Network. In: KDD '19: THE 25TH ACM SIGKDD Conference ON Knowledge Discovery AND Data Mining. **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. Anchorage AK USA: ACM Press, 25 jul. 2019. P. 793–803. ISBN 978-1-4503-6201-6. DOI: 10/gf7ngg. Disponível em: https://dl.acm.org/doi/10.1145/3292500.3330961. Citado nas pp. 21, 37, 61, 62.

ZHANG, Fanjin et al. OAG: Toward Linking Large-Scale Heterogeneous Entity Graphs. In: PROCEEDINGS of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York, NY, USA: Association for Computing Machinery, 25 jul. 2019. (KDD '19), p. 2585–2595. ISBN 978-1-4503-6201-6. DOI: 10/gf7ngf. Disponível em: https://doi.org/10.1145/3292500.3330785. Acesso em: 26 mai. 2021. Citado na p. 40.

ZHANG, Yiming et al. Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework. In: PROCEEDINGS of the 28th ACM International Conference on Information and Knowledge Management. New York, NY, USA: Association for Computing Machinery, 3 nov. 2019. (CIKM '19), p. 549–558. ISBN 978-1-4503-6976-3. DOI: 10/ggnsqt. Disponível em: https://doi.org/10.1145/3357384.3357876. Acesso em: 5 mar. 2021. Citado na p. 37.

ZHANG, Yiming et al. Your Style Your Identity: Leveraging Writing and Photography Styles for Drug Trafficker Identification in Darknet Markets over Attributed Heterogeneous Information Network. In: THE World Wide Web Conference. **The World Wide Web Conference on - WWW '19**. New York, NY, USA: Association for Computing Machinery, 13 mai. 2019. (WWW '19), p. 3448–3454. ISBN 978-1-4503-6674-8. DOI: 10/ghcqzj. Disponível em: http://dl.acm.org/citation.cfm?doid=3308558.3313537. Acesso em: 5 mar. 2021. Citado na p. 37.

ZHENG, Han et al. Competitive and Cooperative Heterogeneous Deep Reinforcement Learning. In: PROCEEDINGS OF THE International Joint Conference ON Autonomous Agents AND Multiagent Systems, AAMAS. 2020-May, p. 1656–1664. Citado na p. 59.

ZHOU, Meng et al. Learning Implicit Credit Assignment for Cooperative Multi-Agent Reinforcement Learning. **Advances in Neural Information Processing Systems**, v. 33, p. 11853–11864, 2020. Disponível em: https://proceedings.neurips.cc/paper/2020/hash/8977ecbb8cb82d77fb091c7a7f186163-Abstract.html. Acesso em: 20 abr. 2021. Citado na p. 59.

APÊNDICE A – PUBLICAÇÕES

Este apêndice apresenta artigos publicados em periódicos e conferências, *pre-prints*, trabalhos sob revisão e outros artefatos tornados públicos no decorrer do programa de doutorado.

REVISTAS E PERIÓDICOS

MENEGHETTI, Douglas De Rizzo et al. Detecting Soccer Balls with Reduced Neural Networks: A Comparison of Multiple Architectures under Constrained Hardware Scenarios. **Journal of Intelligent & Robotic Systems**, v. 101, n. 3, p. 53, mar. 2021. ISSN 0921-0296, 1573-0409. DOI: 10/gj2zhb. arXiv: 2009.13684. Disponível em: http://link.springer.com/10.1007/s10846-021-01336-y.

Sob revisão

MENEGHETTI, Douglas De Rizzo; AQUINO JUNIOR, Plinio Thomaz. **Application and Simulation of Computerized Adaptive Tests Through the Package Catsim**. 19 jul. 2018. arXiv: 1707.03012 [stat]. Disponível em: http://arxiv.org/abs/1707.03012. Acesso em: 22 abr. 2021.

FERREIRA, Leonardo Anjoletto et al. CAPTION: Caption Analysis with Proposed Terms, Image's Objects and NLP. **Information Processing and Management**, p. 22, abr. 2021. ISSN 0306-4573.

CONFERÊNCIAS

MENEGHETTI, Douglas De Rizzo; BIANCHI, Reinaldo Augusto da Costa. Towards Heterogeneous Multi-Agent Reinforcement Learning with Graph Neural Networks. XVII Encontro Nacional de Inteligência Artificial e Computacional (Porto Alegre, RS, Brasil). [S.l.], 21 out. 2020. Disponível em: https://underline.io/events/33/sessions/666/lecture/3261-towards-heterogeneous-multi-agent-reinforcement-learning-with-graph-neural-networks.

DE OLIVEIRA, Jonas Henrique Renolfi et al. Object Detection under Constrained Hardware Scenarios: A Comparative Study of Reduced Convolutional Network Architectures. In: 2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium ON Robotics (SBR) AND 2019 Workshop ON Robotics IN Education (WRE). **2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop**

on Robotics in Education (WRE). Rio Grande, Brazil: IEEE, out. 2019. P. 25–30. ISBN 978-1-72814-268-5. DOI: 10/ghpz8g. Disponível em: https://ieeexplore.ieee.org/document/9018560/.

WORKSHOPS

MENEGHETTI, Douglas De Rizzo; BIANCHI, Reinaldo Augusto da Costa. Specializing Inter-Agent Communication in Heterogeneous Multi-Agent Reinforcement Learning Using Agent Class Information. In: 35TH AAAI Conference ON Artificial Intelligence (AAAI-21). **AAAI-21 Workshop on Reinforcement Learning in Games**. Virtual Conference: [s.n.], 8 fev. 2021. P. 12. Disponível em: https://arxiv.org/abs/2012.07617.

FERREIRA, Leonardo Anjoletto; MENEGHETTI, Douglas De Rizzo; SANTOS, Paulo Eduardo. CAPTION: Correction by Analyses, POS -Tagging and Interpretation of Objects Using Only Nouns. In: FIRST Annual International Workshop ON Interpretability: METHODOLOGIES AND Algorithms (IMA 2019).

AQUINO JUNIOR, Plinio Thomaz et al. HERA: Home Environment Robot Assistant. In: II BRAZILIAN HUMANOID ROBOT WORKSHOP (BRAHUR) and III Brazilian Workshop on Service Robotics (BRASERO). [s.l.: s.n.], abr. 2019.

BASES DE DADOS

MENEGHETTI, Douglas De Rizzo et al. **Annotated Image Dataset of Household Objects from the RoboFEI@Home Team**. [S.l.]: IEEE Dataport, 2020. DOI: 10/ghdk9h. Disponível em: http://dx.doi.org/10.21227/7wxn-n828.

BIANCHI, Reinaldo Augusto da Costa et al. **Open Soccer Ball Dataset**. [S.l.]: IEEE, 21 set. 2020. DOI: 10/ghcfxn. Disponível em: https://ieee-dataport.org/open-access/open-soccer-ball-dataset.

SIMPÓSIOS

DOMINGUES, Pedro Henrique Silva; MENEGHETTI, Douglas De Rizzo; AQUINO JUNIOR, Plinio Thomaz. Comparação Entre Métodos de Detecção de Objetos Por Pontos-Chave e Redes Neurais. In: VIII Simpósio DE Iniciação Científica, Didática E DE Ações Sociais DA

FEI. disponível em: https://fei.edu.br/sites/sicfei/2018/cc/SICFEI_2018_paper_106.pdf.

APÊNDICE B – CONTRIBUIÇÕES DE CÓDIGO ABERTO

Este apêndice apresenta projetos de código aberto próprios e contribuições realizadas a projetos de terceiros no decorrer da realização desta pesquisa. A versão eletrônica desta tese possui *hyperlinks* para as respectivas páginas na *internet* de cada contribuição individual[1].

a) HCA-Net(EF)
b) HCA-Net(AO)
c) PyTorch Geometric: PR #1234
d) microRTS: PRs #52 #53 #55 #56 #58 #60 #61 #62 #63 #64 #65 #66 #67 #69 #70 #72 #73 #74 #75 #76 #78 #82
e) Half-Field Offense: PR #90
f) StarCraft Multi-Agent Challenge (SMAC): PRs #26 #27 #29 #31 #46 #51 #55 #56 #57 #75 #76
g) FacebookResearch/ELF: PR #135
h) DeepMind/pysc2: PR #312
i) abntex/biblatex-abnt: PRs #65 #66 #67 #68 #69 #70 #72

Os seguintes programas de *software* abertos foram criados no decorrer ou como subprodutos da pesquisa realizada durante o programa de doutorado.

a) *python-microRTS*, um *fork* de um projeto de código aberto corrigido e documentado, com a função de comunicar o ambiente μRTS com a linguagem Python;
b) Pacotes de detecção de objetos para a linguagem Python[2], o *framework Robot Operating System* (ROS)[3], assim como *scripts* utilitários[4].

[1] Na data da publicação desta tese, a lista de endereços também está disponível em `https://douglasrizzo.com.br/phd-oss-contributions/`
[2] `https://douglasrizzo.github.io/dodo_detector`
[3] `https://github.com/douglasrizzo/dodo_detector_ros`
[4] `https://github.com/douglasrizzo/detection_util_scripts`