UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Comportamento Cooperativo Emergente em Rede Neural Aplicada a Jogo de Combate por Turno

Ítalo Tobler Silva

São Carlos – SP

Comportamento Cooperativo Emergente em Rede Neural Aplicada a Jogo de Combate por Turno

Ítalo Tobler Silva

Orientador: **Ms. Leonardo Tórtoro Pereira**

Monografia final de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como requisito parcial para obtenção do título de Bacharel em Computação.
Área de Concentração: Inteligência Computacional

USP – São Carlos
Novembro de 2019

Silva, Ítalo Tobler
Comportamento Cooperativo Emergente em Rede Neural Aplicada a Jogo de Combate por Turno / Ítalo Tobler Silva. - São Carlos - SP, 2019.
65 p.; 29,7 cm.

Orientador: Leonardo Tórtoro Pereira.
Monografia (Graduação) - Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos - SP, 2019.

1. Aprendizado de Máquina. 2. Jogos. 3. Comportamento Cooperativo. I. Pereira, Leonardo Tórtoro. II. Instituto de Ciências Matemáticas e de Computação (ICMC/USP). III. Título.

Dedico esse trabalho à Amanda, pelas horas de conversa ao longo do curso.

E à minha mãe, cujo aniversário esqueci por conta da monografia,
e ainda assim apenas se preocupou se eu conseguiria entregar o trabalho.

AGRADECIMENTOS

Agradecimentos ao meu orientador Leonardo, cuja ajuda foi indispensável nesse semestre que não parecia tão corrido inicialmente.

Agradecimentos ao Cláudio por, junto com o Leonardo, ter se prontificado a ajudar tantos alunos com interesse em pesquisa na área de jogos.

Agradecimentos também ao grupo Fellowship of the Game por despertar meu interesse na área e disponibilizar um espaço tão confortável de aprendizado e prática.

Por fim um agradecimento a todos os professores com quem tiver o prazer de aprender ao longo dos meus anos de curso, assim como aos funcionários de apoio do ICMC que se mostraram dispostos a ajudar um aluno perdido em mais de uma ocasião.

“Memento Mori”
(desconhecido)

RESUMO

SILVA, Í. T.. **Comportamento Cooperativo Emergente em Rede Neural Aplicada a Jogo de Combate por Turno**. 2019. 65 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Este trabalho propõe avaliar se agentes independentes treinando uma rede neural com treinamento por reforço para jogar um jogo de times conseguem desenvolver comportamentos cooperativos mesmo quando são forçados a ignorar a existência de seus respectivos aliados. Um jogo de batalha por turno é utilizada como ambiente de treino para os modelos, e tanto o treino quanto a coleta de dados são feitas com a ajuda de uma heurística. Após analisar as batalhas dos modelos treinados, foi possível observar que a rede neural resultante do agente individualista desenvolveu sim comportamentos cooperativos, mas de maneira bem menos acentuada que sua contraparte cooperativa.

Palavras-chave: Aprendizado de Máquina, Jogos, Comportamento Cooperativo.

ABSTRACT

SILVA, Í. T.. **Comportamento Cooperativo Emergente em Rede Neural Aplicada a Jogo de Combate por Turno**. 2019. 65 f. Monografia (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

This paper proposes evaluating if independent agents training a neural network through reinforcement learning to play a team game can develop cooperative behaviours even when forced to ignore the existence of their respective allies. A turn-based battle game is used as environment for training the models, and a heuristic is used for both training and data collection. After analyzing the battle results of the trained models, we could indeed notice cooperative behaviour from the neural network trained with the individualistic agent, but it was on a much smaller scale than it's cooperative counterpart.

Key-words: Machine Learning, Games, Cooperative Bevahiour.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API ..... Application Programming Interface
BtH ..... Bonus to Hit
HP ..... Health Points
IA ..... Inteligência Artificial
ML ..... Machine Learning
PPO ..... Proximal Policy Optmization
RPG ..... Role Playing Game

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Motivação e Contextualização

Machine Learning (ML) ou Aprendizado de Máquina é um campo de estudos de Inteligência Artificial (IA) cada vez mais discutido atualmente, que gira em torno do uso de aprendizado por inferência para criar modelos de comportamento que se baseiam mais nos dados e menos nas ideias dos desenvolvedores. A utilidade desta técnica para análise de dados faz com que esteja presente em pesquisas de diversas áreas do conhecimento (RAJKOMAR; DEAN; KOHANE, 2019) e tenha até mesmo sido conectada ao cenário político de grandes países (GRANVILLE, 2018).

Outra utilidade comum dessa técnica pode ser vista no treinamento de redes neurais capazes de realizar atividades mais complexas, que se assemelhem mais ao comportamento humano ou tenham maior capacidade de adaptação para novas situações (SONI; HINGSTON, 2008). Este tipo de aplicação é muito interessante para a indústria de jogos, que vê a criação de IAs desafiantes e interessantes como um atrativo para seu produto. A relação entre ML e jogos, entretanto, se extende além da indústria e pode ser percebida também no meio acadêmico, pois jogos são muito úteis como ambientes para treinamento e observação de redes neurais.

Um exemplo recente do uso de jogos para pesquisa em machine learning é a pesquisa realizada pelo time da OpenAI Five (OPENAI, 2018), na qual foi utilizado DOTA 2[1], um jogo de times complexo, que exige cooperação, planejamento e tomada de decisões em tempo real para vencer. Nessa pesquisa a equipe conseguiu fazer com que agentes independentes e sem comunicação entre si desenvolvessem comportamento cooperativo e um entendimento bom o suficiente do jogo para derrotar os melhores times de humanos profissionais do mundo[2].

Na pesquisa mencionada acima os agentes independentes aprenderam a cooperar sem se comunicar, mas durante suas observações eles ainda podiam ver os aliados. Caso estivessem em uma situação em que o comportamento cooperativo se faz necessário para ter um melhor desempenho, seriam eles capazes de aprender a cooperar da mesma maneira tendo acesso limitado a informações sobre seus aliados?

Jogos do gênero Role Playing Game (RPG) com batalhas de turno e múltiplas unidades

[1] <http://br.dota2.com/>
[2] <https://openai.com/blog/how-to-train-your-openai-five/>

são exemplos de jogos com menor complexidade mas que também exigem um nível de cooperação entre as unidades para atingir a vitória. Normalmente essa cooperação existe por conta do jogador controlar todas as unidades sozinho, mas é possível fazer com que cada unidade seja controlada por uma IA independente, simulando uma situação parecida com a proposta pela pesquisa do time da OpenAI Five num escopo factível para um trabalho de graduação, tanto em termos de tempo de desenvolvimento como de *hardware*, uma vez que a OpenAI utilizava-se de 256 núcleos de GPU e 128 mil de CPU (OPENAI, 2018) para o treino de redes neurais.

1.2 Objetivos

Utilizando o *plugin* de ML da *engine* de jogos Unity, treinar dois modelos em um jogo com batalha de turnos e de times desenvolvido especificamente para isso. Cada modelo teria um tipo de agente diferente: Um deles **Individualista** nas observações e cálculo de recompensas e o outro **Cooperativo**. Após o treino, analisar seus comportamentos, principalmente buscando saber se o agente **Individualista** foi capaz de aprender comportamentos cooperativos.

1.3 Organização

No capítulo 2, "Métodos, Técnicas e Tecnologias Utilizadas", temos uma explicação de conceitos importantes para o bom entendimento dos demais capítulos, além de uma breve descrição das atividades realizadas no trabalho. No capítulo 3, "Desenvolvimento", essas atividades são descritas de maneira mais detalhada e os resultados obtidos são apresentados e analisados. No capítulo 4, "Conclusão", é feito um resumo das observações mais importantes levando em conta os objetivos definidos anteriormente, bem como um planejamento de projetos futuros que podem surgir a partir deste.

Capítulo 2

MÉTODOS, TÉCNICAS E TECNOLOGIAS UTILIZADAS

2.1 Tecnologias

Para melhor compreender o desenvolvimento deste trabalho, apresentamos informações fundamentais sobre algumas tecnologias utilizadas, listadas abaixo.

2.1.1 Unity Game Engine

A Unity[1] é uma *Game Engine* (ou Motor de Jogos, em português) gratuita que possui vários módulos relevantes já implementados, como cálculos de física e configuração de interface gráfica. Ela também permite ao desenvolvedor criar comportamentos customizados usando scripts nas linguagens C# ou javascript.

2.1.2 Tensorflow

O Tensorflow[2] é uma biblioteca *open-source*[3] de ML escrita na linguagem Python. Ela permite o uso de uma *Application Programming Interface (API)* (Interface para Programação de Aplicativos, em português) para definir parâmetros de treino e outras configurações necessárias, gerando um modelo e sumários de treino ao final do treinamento.

2.1.3 Unity ML-Agents Toolkit

O Unity ML-Agents[4] é um *plugin open-source* para a Unity, que permite ao desenvolvedor configurar agentes na cena do seu jogo, possibilitando uma comunicação com a API do Tensorflow e transformando a cena em um ambiente de treino.

[1] <https://unity.com/>
[2] <https://www.tensorflow.org/>
[3] <https://github.com/tensorflow/>
[4] <https://github.com/Unity-Technologies/ml-agents>

2.1.4 Tensorboard

O Tensorboard[5] é uma ferramenta de visualização que, a partir dos sumários gerados durante o treinamento, fornece gráficos legíveis ou tabelas para análise.

2.1.5 Overleaf

Foi utilizado o software de edição e compilação de texto LaTeX Overleaf para a escrita da monografia [6].

2.1.6 Ferramentas estatísticas

Para o cálculo das estatísticas e criação dos gráficos apresentados nessa monografia foi utilizado o Apache OpenOffice Calc[7].

2.2 Conceitos e Técnicas

Alguns dos conceitos mencionados e técnicas utilizadas exigem uma explicação mais detalhada para o entendimento, como visto abaixo.

2.2.1 RPG de Turno

Numa batalha de turnos clássica encontrada em jogos de RPG, existem dois times com quantidade variável de unidades. O jogo é dividido em turnos sequenciais e a cada turno uma personagem (ou unidade) tem a chance de agir uma vez. Cada unidade possui pontos de vida, também chamados de *Health Points (HP)* e a batalha se encerra quando todas as unidades de um time não possuem mais HP, causando a vitória do time oposto.

Uma unidade costuma ter outros valores relevantes, comumente chamados de *status*. Alguns dos status relevantes para o jogo criado neste trabalho são:

- **Ataque:** Valor associado a quanto dano a unidade consegue causar nos inimigos.
- **Defesa:** Valor asssociado à redução de dano recebido pela unidade, seja reduzindo o valor diretamente ou fazendo com que o golpe não conecte.
- **Bonus to Hit (BtH):** Valor associado à chance de um ataque conectar.

Alguns outros conceitos relevantes:

[5] <https://www.tensorflow.org/tensorboard>
[6] <https://www.overleaf.com/about>
[7] <https://www.openoffice.org/pt/product/calc.html>

Figura 1 – Exemplo de uma cena de batalha do RPG Final Fantasy VI. Os personagens do jogador localizam-se à direita, enquanto o inimigo está à esquerda.

- ***Skill*(Habilidade):** Uma ação com efeito diferente da ação padrão usual, o ataque simples que apenas causa dano no alvo desejado.
- ***Cooldown*:** Tempo necessário para que uma habilidade possa ser usada novamente, pode ser contado em tempo real ou relativo, como por exemplo em número de turnos.
- ***Heal*(Cura):** Ação que ao invés de causar dano e diminuir o HP, o recupera.
- ***Status Effect*:** Condição especial aplicada à unidade, que causa um efeito todo turno, ou por toda a sua duração.
- ***Stun*:** *Status effect* que deixa a unidade afetada impossibilitada de realizar uma ação no seu turno.
- ***Blind*(Cegueira):** *Status effect* que diminui a chance dos golpes da unidade afetada conectarem enquanto durar.
- ***Poison*(Veneno):** *Status effect* que faz a unidade afetada tomar dano uma vez por turno enquanto durar.

2.2.2 Rede Neural

Redes neurais são sistemas com múltiplas camadas de processamento que, ao receber uma entrada, fazem com que ela passe por uma ou mais dessas camadas para gerar uma saída. As camadas de processamento são formadas a partir de um treinamento para serem capazes de prever qual é o valor de saída desejado com base apenas na entrada recebida.

[8] <https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg>

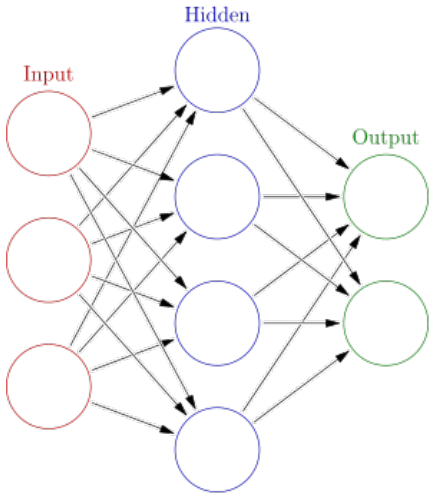

Figura 2 – Representação Gráfica de uma Rede Neural[8]

2.2.3 Machine Learning(Aprendizado de máquina)

Um treinamento de rede neural com ML pode usar vários métodos diferentes: treino supervisionado, treino por reforço, treino por imitação etc. Neste trabalho foi usado treino por reforço, que consiste em aleatorizar valores da rede neural, fazer com que ela tente prever valores (o que equivale a tomar uma decisão, como será explicado mais abaixo) e distribuir recompensas e punições de acordo com o desempenho dela para que ela detecte quais escolhas são corretas.

Após identificar se as decisões atuais são melhores ou piores que as anteriores, um algoritmo é utilizado para ajustar os valores da rede neural e tentar otimizá-la, e inicia-se um novo ciclo (ou episódio) de avaliação. A repetição contínua desse processo eventualmente resulta em uma rede neural satisfatória. Neste trabalho o algoritmo utilizado para otimização é o *Proximal Policy Optmization (PPO)*, desenvolvido por membros da equipe do OpenAI em 2017 (SCHULMAN *et al.*, 2017) e adotado por desenvolvedores da Unity como o algoritmo padrão utilizado pelo ML-Agents no modo de treino por reforço (JULIANI *et al.*, 2018)

2.2.4 Unity ML-Agents

Para utilizar o *plugin* do ML-Agents em um jogo, é preciso estar usando uma versão da Unity compatível com a versão desejada do *plugin* e seguir os guias de instalação disponíveis na página do projeto no GitHub [9]. Após a instalação dos pacotes necessários e *download* dos arquivos do *plugin*, é necessário alterar o jogo para que ele seja compatível com um ciclo de treinamento.

Os requisitos para preparar o jogo podem ser sumarizados em configurar as observações e ações do(s) agente(s), configurar uma "Academia"responsável por reajustar a cena a cada ciclo de treino, e montar um sistema de recompensas e punições de acordo com as ações dos agentes.

[9] <https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Installation.md>

O *plugin* já contém classes base para a academia e agentes, cabe ao desenvolvedor sobrescrever os métodos de acordo com o que diz a documentação e definir como a cena do seu jogo deve ser reiniciada para garantir um treino consistente.

Agentes precisam ter um cérebro, responsável por definir como sua decisão é tomada. Esse cérebro pode fazer com que eles esperem instrução de um jogador humano para agir, utilizem um modelo pré-treinado para tomada de decisões, utilizem uma heurística pré-programada ou usem a API para interagir com o Tensorflow e treinar um modelo.

O *plugin* permite também que seja implementado um sistema de currículo para o treinamento de reforço. Quando o treinamento atinge níveis de recompensa ou número de episódios desejados, um ou mais parâmetros podem ser alterados como desejado e o desenvolvedor pode fazer com que esses parâmetros alterem o ambiente de treinamento, aumentando a dificuldade dos episódios.

2.3 Metodologia

Neste trabalho foi usado como base um jogo simples, que consiste apenas de uma batalha em turnos com 2 times de duas unidades se enfrentando. Cada unidade tem os mesmos *status*, o mesmo conjunto de habilidades, e uma vez por turno escolhe uma ação e um alvo para essa ação. Todas as ações possuem *cooldown* e tipos de alvos possíveis, e ações inválidas quanto a esses requisitos não podem ser tomadas, sendo ocultadas. Cada unidade possui um agente independente dos demais e cada batalha corresponde a um episódio para efeitos de treino.

Foram implementados dois tipos de agentes. O agente **Individualista**, ao observar o estado atual da batalha e ao calcular suas recompensas, ignora por completo seu aliado. Já o agente **Cooperativo** leva essas informações em conta nas observações e chega até mesmo a usar a média entre o seu desempenho e o desempenho de seu aliado como recompensa.

Quanto ao treino dos modelos, foi implementada uma heurística para servir de time adversário aos agentes sendo treinados, e a dificuldade dela foi dividida em 3 níveis. Foi configurado um currículo de treino onde, para cada nível de dificuldade da heurística, ao atingir um valor suficientemente alto de recompensa o nível aumentaria. Foram então treinados 3 modelos para cada tipo de agente, e os modelos com melhor recompensa final de cada trio foram selecionados para fazer a comparação de dados.

Após a seleção, foi feita uma sequência de batalhas entre esses modelos e cada nível de dificuldade da heurística para coleta de dados, assim como batalhas de um modelo contra o outro. Os resultados de cada batalha e outros resultados relevantes foram armazenados em tabelas, a análise estatística e de visualização desses dados coletados serão apresentados no capítulo seguinte da monografia.

Capítulo 3

DESENVOLVIMENTO

3.1 O Problema

Dados dois agentes: um **Individualista** que não tem acesso às informações do aliado durante a batalha e não leva o aliado em consideração ao calcular as recompensas de fim de batalha, e um **Cooperativo**, que tem acesso a essas informações e faz uma média entre o seu desempenho e o desempenho do aliado para calcular recompensas; Avalia-se o desempenho dos modelos treinados com esses agentes lutando contra uma heurística pré-determinada em diferentes níveis de dificuldade e lutando entre si, tanto para comparar seu desempenho quanto para determinar se o agente **Individualista** foi capaz de aprender a cooperar com o aliado.

3.2 Atividades Realizadas

3.2.1 Revisão Bibliográfica

A revisão bibliográfica deste trabalho foi feita usando em grande parte o banco de dados da ferramenta Google Scholar[1]. Palavras-chaves utilizadas nas pesquisas incluem "Machine Learning", "Cooperative Behaviour", "Review", "Games", "Unity", "Turn-based".

Como foi mencionado na introdução, pesquisas na área de ML estão em alta nos anos recentes como visto em (AL-JARRAH *et al.*, 2015), onde o autor faz uma revisão da literatura de modelagem de dados para análise com ML, focando na eficiência computacional e de uso de memória de cada modelo. Ou em (MAXWELL; WARNER; FANG, 2018), onde o autor faz uma revisão de aplicações da técnica de ML para classificação de imagens, comparando esses casos quanto ao algoritmo usado, estruturas de dados e outros fatores relevantes para a otimização. Muitas vezes vemos menção a ML em áreas aparentemente distantes de computação, como em (MOSAVI; OZTURK; CHAU, 2018), onde o autor faz uma revisão da literatura de aplicações de ML para previsão de enchentes, demonstrando o estado da arte dessa aplicação e identificando os modelos mais apropriados. Como também em (LIAKOS *et al.*, 2018), onde o autor faz uma revisão da literatura de aplicações de ML em sistemas de produção agrícola, classificadas de acordo com o tipo de sistema. Essa alta flexibilidade de aplicações da área,

[1] <https://scholar.google.com>

bem como a acessibilidade relativamente alta quando se trata de apenas aplicar os métodos já estabelecidos podem contribuir para a popularidade de pesquisas na área.

A associação entre machine learning e aplicações em jogos também pode ser percebida sem muita dificuldade, como visto em (OPENAI, 2018), onde a equipe de pesquisadores treinou agentes independentes para que fossem capazes de tomar decisões em tempo real, prever recompensas futuras e cooperar sem comunicação em um jogo mais complexo que Go, ou em (SILVEY; TOLK; TRACY, 2018), onde o autor apresenta o uso de ML e um ambiente de treino criado em uma game engine como alternativa para a definição manual de políticas complexas. Em alguns desses casos os jogos em questão são jogos do tipo RPG de turno, como em (Nam; Ikeda, 2019), onde o autor usa uma combinação de ML e Geração Procedural de Conteúdo para gerar níveis de um jogo de RPG de turno.

3.2.2 Desenvolvimento do Jogo

O jogo já foi descrito em termos gerais nos capítulos anteriores, mas alguns pontos não mencionados merecem um pouco mais de atenção. Ao implementar a lógica do jogo e as heurísticas, as escolhas de design foram feitas levando em consideração que o agente precisava ser incentivado a cooperar com o aliado para obter uma recompensa mais alta.

Para adicionar um pouco de aleatoriedade e deixar o jogo menos determinístico, a ordem das unidades dentro de cada turno foi decidida de maneira aleatória a cada novo turno, com a única restrição sendo que uma unidade pode agir apenas uma vez por turno. Além disso, foi adicionada uma pequena variação no ataque de cada unidade (inteiro aleatório entre 15 e 17) e todo tipo de habilidade ofensiva possuia, em situação comum, uma chance de 5% de não conectar.

Todas as habilidades, com exceção do ataque básico, tem um tempo de *cooldown* mais alto que a duração de qualquer efeito que ela tenha. Isso foi feito tanto para forçar a batalha a ter uma duração máxima menor quanto para forçar a unidade a usar a habilidade da maneira mais eficiente possível, de preferência coordenando seu uso com as habilidades do aliado. As habilidades se dividem entre habilidades puramente de dano (*Attack, Double e Poison*), habilidades que diminuem a chance de ataques inimigos conectarem (*Blind e Block*), uma habilidade de cura(*Heal*), que pode ser usada em si mesmo ou no aliado, e uma habilidade de *Stun*, que imobiliza um inimigo por dois turnos, o que pode ser considerado o *status effect* mais forte, já que nega 100% do dano que aquela unidade poderia causar durante esse tempo e atrasa qualquer reação que ela possa ter às ações tomadas nesse período.

3.2.3 Configuração dos Agentes

A API do ML-Agents espera que as observações e as ações sejam repassadas ao cérebro como vetores de números de ponto flutuante, e é responsabilidade do agente converter os dados

observados para repassar ao cérebro e analisar os números recebidos do cérebro para poder interagir com o ambiente. Para ambos os tipos de agente, ao fazer a observação da situação atual, as ações impossíveis de serem realizadas devem ser ocultas da árvore de decisões utilizando um método disponibilizado pela própria API.

As observações do agente Cooperativo incluem os possíveis *status effects* afetando a unidade do agente, seu valor de HP atual, o número do turno atual, o *cooldown* restante de suas habilidades, bem como o valor atual de HP e a situação dos *status effects* de seus inimigos e aliados. Para o agente **Individualista** os valores relativos ao seu aliado serão sempre vistos como 0, mesmo com o valor correto sendo avaliado para determinar as ações possíveis.

Para evitar comportamentos inesperados, as recompensas dos agentes são calculadas apenas no final de cada batalha, quando o vencedor já foi decidido. Caso as recompensas fossem dadas, por exemplo, ao causar dano no inimigo os modelos poderiam dar prioridade demais às habilidades ofensivas. Os valores de recompensas escolhidos foram 0 para o perdedor e 50 para o vencedor, com um bônus para o perdedor diretamente proporcional à porcentagem de HP combinado que os vencedores perderam, sendo 0 caso os vencedores não tenham perdido HP e 25 caso não tenham HP sobrando. Esse mesmo valor é aplicado como uma punição aos vencedores, mas enquanto para o agente **Cooperativo** é usado o HP combinado da própria unidade e de seu aliado, para o agente **Individualista** o HP restante do seu aliado é ignorado, e apenas o seu valor é considerado.

3.2.4 Heurística

A heurística implementada define uma ordem de prioridade para os tipos de habilidades disponíveis e a situação atual e, de acordo com a disponibilidade dessas habilidades, usa a melhor habilidade para a situação de acordo com uma análise empírica do comportamento de IAs e jogadores em jogos semelhantes. A escolha de habilidade e alvo leva em consideração também a chance do ataque conectar com cada alvo possível. Infelizmente a escolha de alvo não ficou tão aleatória quanto deveria e as consequências disso serão discutidas mais adiante.

3.2.5 Treino

A heurística descrita acima possui um parâmetro de dificuldade que, à medida que é incrementado, permite que a heurística utilize mais habilidades e se torne mais forte. O currículo de treino foi configurado de modo que, quando a recompensa atingisse um valor satisfatório, a dificuldade da heurística fosse incrementada. Como o aumento de dificuldade forçava a recompensa máxima possível a ser menor, o valor considerado satisfatório precisou ser ajustado após testes.

Tendo sido definido o treino, foi necessário definir os parâmetros de treino para o PPO. Essa etapa também dependeu muito de testes empíricos devido à falta de conhecimento profundo

Tabela 1 – Parâmetros de treino usados para os modelos finais

Parâmetro	**Valor**
trainer	ppo
batch_size	512
beta	0.01
buffer_size	5120
epsilon	0.2
gamma	0.995
hidden_units	256
lambd	0.9
learning_rate	0.001
max_steps	5.0e6
memory_size	256
normalize	False
num_epoch	3
num_layers	6
time_horizon	1024
sequence_length	64
summary_freq	2000
use_recurrent	False
use_curiosity	False
curiosity_strength	0.01
curiosity_enc_size	128

da matemática por trás dos cálculos. Os parâmetros foram considerados satisfatórios quando um modelo capaz de derrotar o nível mais forte da heurística foi obtido em tempo de treino aceitável para a execução do trabalho.

Uma vez que os parâmetros estavam satisfatórios, foram treinados um total de 6 modelos: 3 modelos com o agente **Cooperativo** e 3 modelos com o agente **Individualista**. De cada grupo de 3 modelos, foi selecionado o que tinha maior recompensa média final para fazer a coleta de dados. Tanto nos treinos quanto na heurística cada time sempre tinha o mesmo tipo de comportamento nos agentes.

3.2.6 Coleta e Análise de Dados

Tendo os modelos treinados em mãos, o jogo foi ajustado para permitir um grande número de batalhas consecutivas mesmo que não estivesse em modo de treino, e, para cada modelo, foram feitas sequências de batalhas para nível fraco, médio e forte da heurística, 2000 em cada. Por fim, foram feitas batalhas(também 2000) entre os dois modelos para avaliar seu comportamento quando lutavam contra um oponente anteriormente desconhecido e mais forte do que os inimigos com os quais eles estavam acostumados.

Em todas as batalhas, os dados coletados de cada agente foram um valor indicativo de

vitoria ou derrota, HP restante ao final da batalha, recompensa calculada ao final da batalha, quantos *Stuns* ou *Blinds* foram utilizados ao final da duração de outro *status effect* semelhante previamente aplicado e quantas curas foram aplicadas na unidade aliada. Os 3 últimos parâmetros foram observados como indicadores de comportamento cooperativo, e o uso das habilidades *Block*, *Poison* e *Double* não foram analisados pois foram consideradas muito simples, seja por ter caráter exclusivamente ofensivo no caso de *Poison* e *Double* ou por não permitir uma escolha de alvo no caso de *Block*. Após cada batalha, todos os dados foram salvos em tabelas no formato de arquivo .csv para análise posterior.

3.3 Resultados

3.3.1 Treino dos Agentes

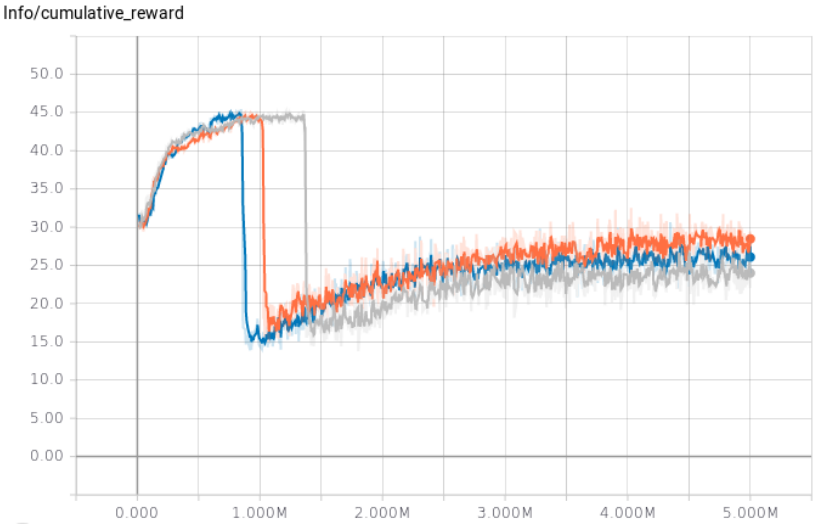

Figura 3 – Recompensas dos 3 agentes Individualistas ao longo do treino

Durante os treinos dos modelos finais os valores se comportaram da maneira esperada, mas é importante ressaltar que, como observado nas Figuras 3 e 4, mesmo quando o treino acabou por ter atingido o numero máximo de episódios é possível notar que o crescimento das recompensas não havia estabilizado por completo. Gráficos com todas as estatísticas de treino dos dois tipos de rede neural podem ser encontrados no apêndice A.

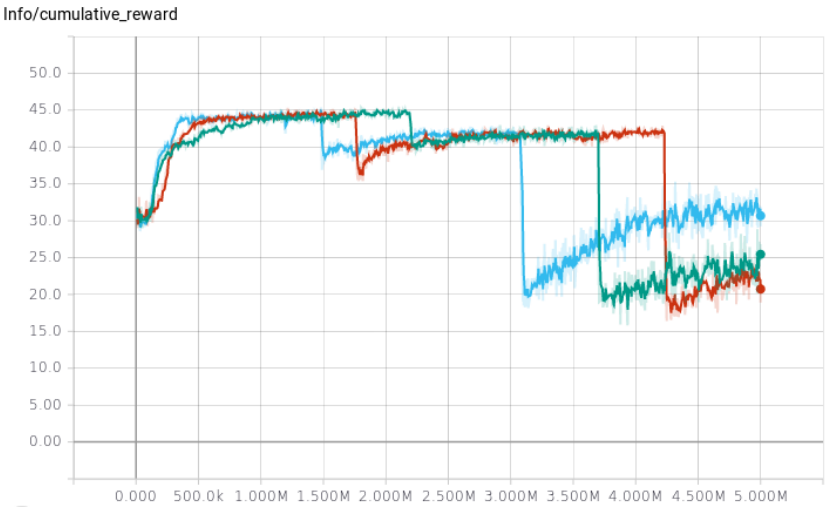

Figura 4 – Recompensas dos 3 agentes Cooperativos ao longo do treino

3.3.2 Taxa de Vitórias

A primeira análise importante é quanto à vitoria dos agentes contra as IAs de cada dificuldade e também na disputa entre os agentes. A Figura 5 mostra os resultados da soma das vitórias dos agentes contra cada tipo de inimigo. É possível notar uma tendência do agente **Cooperativo** em ganhar mais do que o **Individualista** contra todas as IAs. Também é clara a dominância do **Cooperativo** em suas lutas contra o **Individualista**. Porém, para comprovar tais tendências, foi realizada uma análise estatística dos dados.

Ao realizarmos uma análise estatística sobre os dados de vitória de cada agente contra

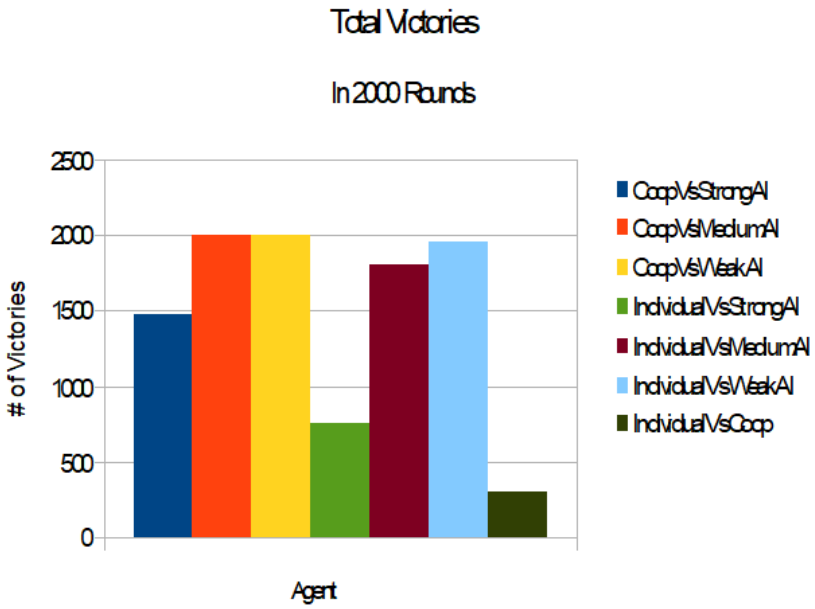

Figura 5 – Total de vitórias de cada agente contra cada IA e contra sua contraparte

cada IA, com um teste-Z de duas proporções foi possível comprovar que o número de vitórias do agente **Cooperativo** é significativamente maior que o do **Individualista** contra cada uma das IAs, considerando um alfa de 5%. A Tabela 2 mostra os resultados de z e p para cada par.

Tabela 2 – Resultados do teste-Z de duas proporções entre as vitórias do agente Cooperativo e o Individualista contra cada IA, considerando um alfa de 5%

IA	Valor de Z	Valor de p
Strong	22.5187	<.00001
Medium	12.7582	<.00001
Weak	5.3246	<.00001

Observando a Tabela 2 podemos ver que para todas as IAs as vitórias do agente **Cooperativo** foram estatísticamente maiores que as do agente *Individualista*. Mostrando que ele adaptou-se melhor contra cada um dos níveis de dificuldades das heurística.

3.3.3 Análise dos Combates

Para fazer uma análise de visualização dos resultados das outras variáveis coletadas, foram criados *boxplots* para os conjuntos de dados de cada agente. Esses gráficos estão distribuídos em imagens e agrupados de acordo com o tipo de inimigos contra quem aquele agente lutou para facilitar a comparação. Nesta seção será feita uma avaliação sobre os resultados mais relevantes encontrados na análise dos gráficos. Alguns dos gráficos foram omitidos para facilitar a leitura, mas todos podem ser encontrados no apêndice B, incluindo aqueles que foram considerados irrelevantes.

3.3.3.1 Recompensas

Analisando os *boxplots* de recompensas como os das Figuras 6 e 7 podemos ver o comportamento esperado do valor de recompensas ficando mais baixo à medida que os inimigos se tornam mais fortes.

Para verificar se houve ou não uma diferença estatisticamente significativa entre as recompensas de cada agente contra cada IA, foi realizada uma análise com o teste-T entre as populações de recompensas. Porém, antes, foi necessário verificar se a variância das populações poderiam ser consideradas iguais ou não (verificação da homoscedasticidade). Para tal, foi feito um teste-F de duas caudas em cada população. Se o *p-valor do teste-F* na Tabela 3 é menor que o alfa utilizado, de 0,05, pode-se rejeitar a hipótese nula de que as variâncias são iguais. Caso contrário, conclui-se que a hipótese de que são iguais não pode ser descartada. Sendo assim, foi aplicado um teste-T homoscedástico ou heteroscedástico para o conjunto baseado na resposta do teste-F.

Para o teste-T foi aplicado o teste de uma cauda, com a hipótese nula de que a média de ambas as populações era igual, e a alternativa de que as médias eram diferentes. Se o

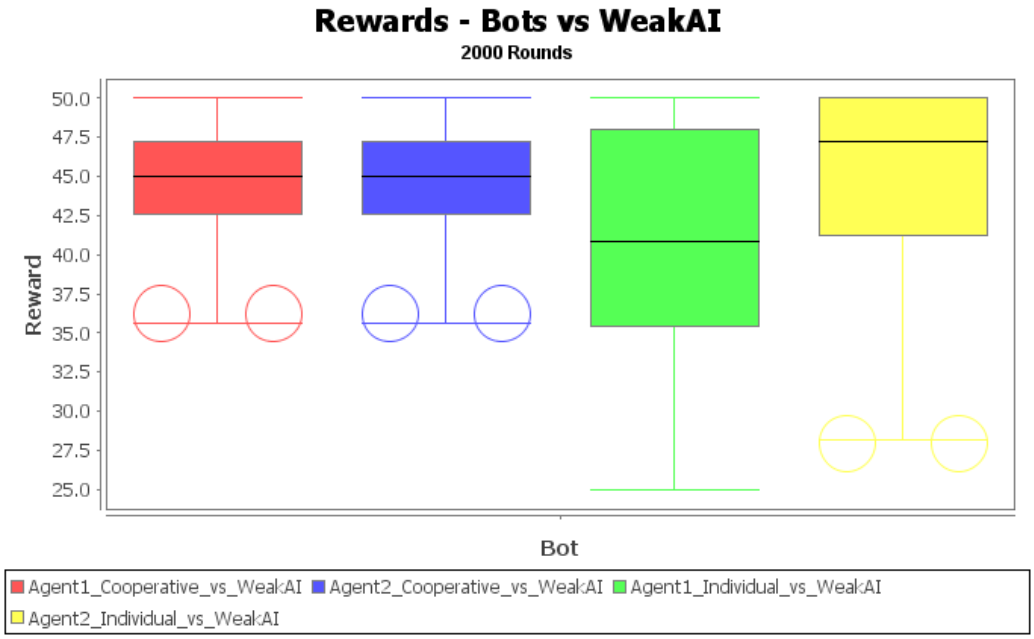

Figura 6 – Boxplot das recompensas de todos os agentes ao lutar contra a IA fraca

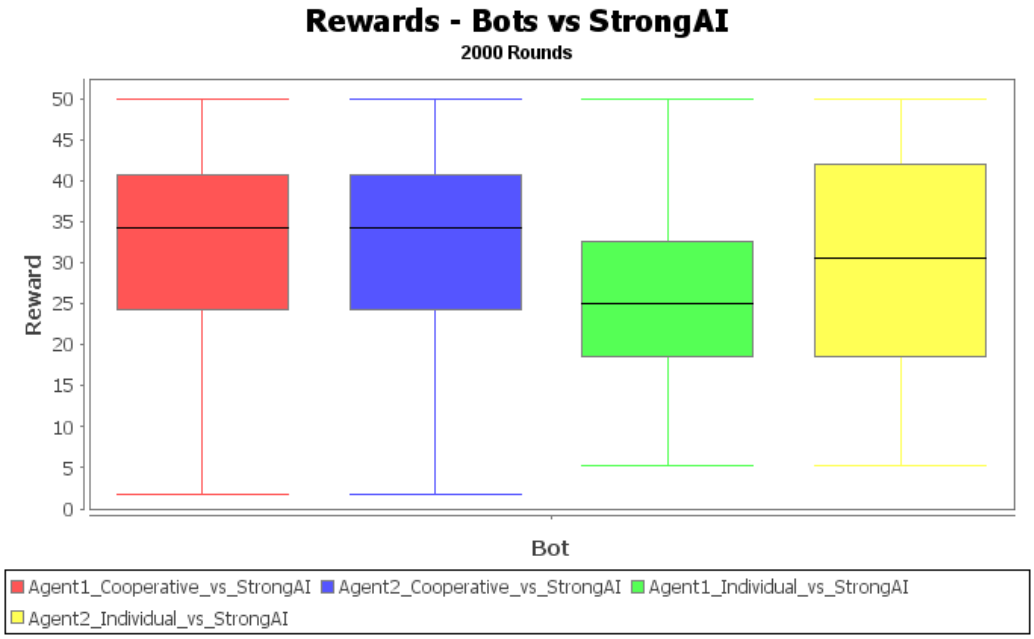

Figura 7 – Boxplot das recompensas de todos os agentes ao lutar contra a IA forte

p-valor do teste-T fosse menor que 0.05, conclui-se que a recompensa do **Cooperativo** era estatisticamente maior que a do **Individual** contra determinada IA. Caso contrário, elas não podem ser consideradas diferentes.

Observando a Tabela 3 é possível concluir que para todos os casos o agente **Cooperativo** conseguiu uma recompensa significativamente maior que o **Individualista**. Isso mostra que não só ele adaptou-se melhor para ganhar de seus inimigos, como para ganhar com melhores condições. É importante notar que um resultado similar era esperado, já que o ato de vencer

Tabela 3 – Resultados do teste-F e teste-T entre as recompensas do agente Cooperativo e o Individualista contra cada IA, e entre eles mesmos (*Versus*) considerando um alfa de 5%

IA	Valor de p para teste-F	Valor de p para teste-T
Strong	0.33503	<.00001
Medium	0.00057	<.00001
Weak	<.00001	<.00001
Versus	0.99999	<.00001

as batalhas por si só gera um bônus de pontos. Porém, vencer batalhas com pouca vida faria esse bônus ser quase irrelevante, o que não foi o caso, como observado nos gráficos da subseçao seguinte.

3.3.3.2 HP Restante

Um comportamento inesperado surge ao observarmos os *boxplots* de HP individual restante como na Figura 8. O HP restante(e consequentemente a recompensa, no caso do **Individualista**) do agente 2 é consistentemente maior do que o HP restante do agente 1. Ao revisar o código à procura de uma explicação para isso, foi notado que a heurística tinha preferência a escolher sempre o agente 1 ao invés do 2 quando não há indicativo claro de qual deles é o melhor alvo, portanto qualquer análise que poderia ser feita utilizando as diferenças de valores de HP restante dos agentes de um mesmo time não é tão confiável para esse conjunto de dados.

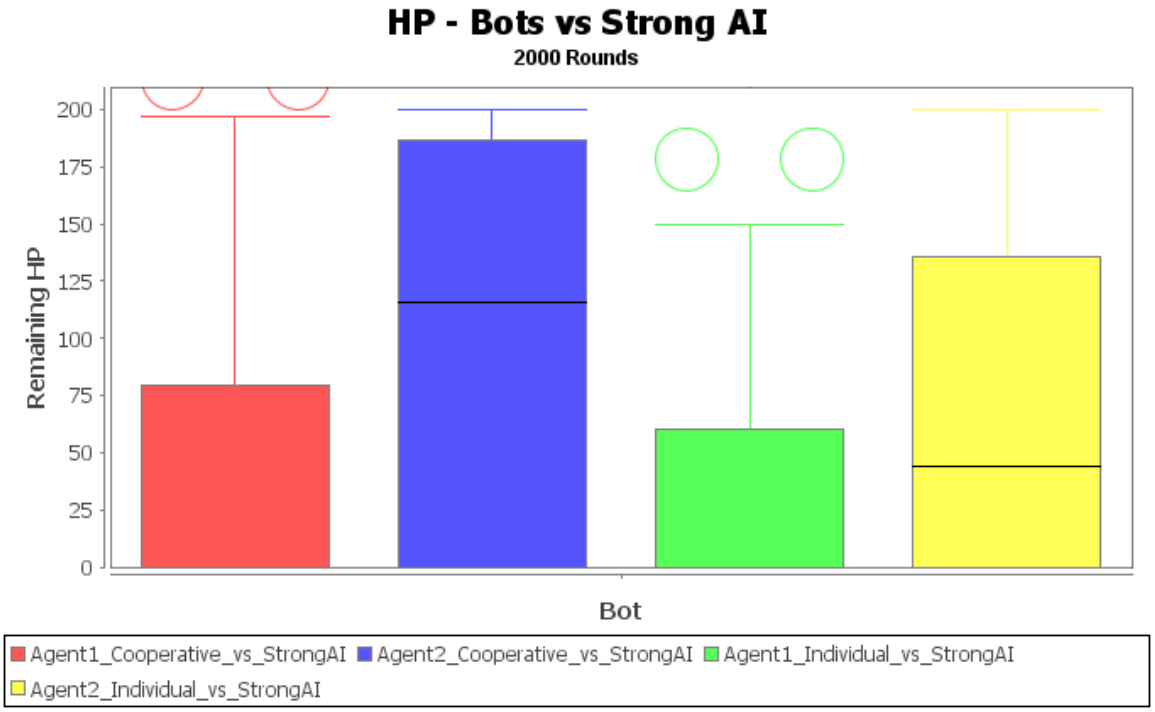

Figura 8 – Boxplot do HP restante de todos os agentes ao lutar contra a IA forte

3.3.3.3 Número de curas usadas no aliado

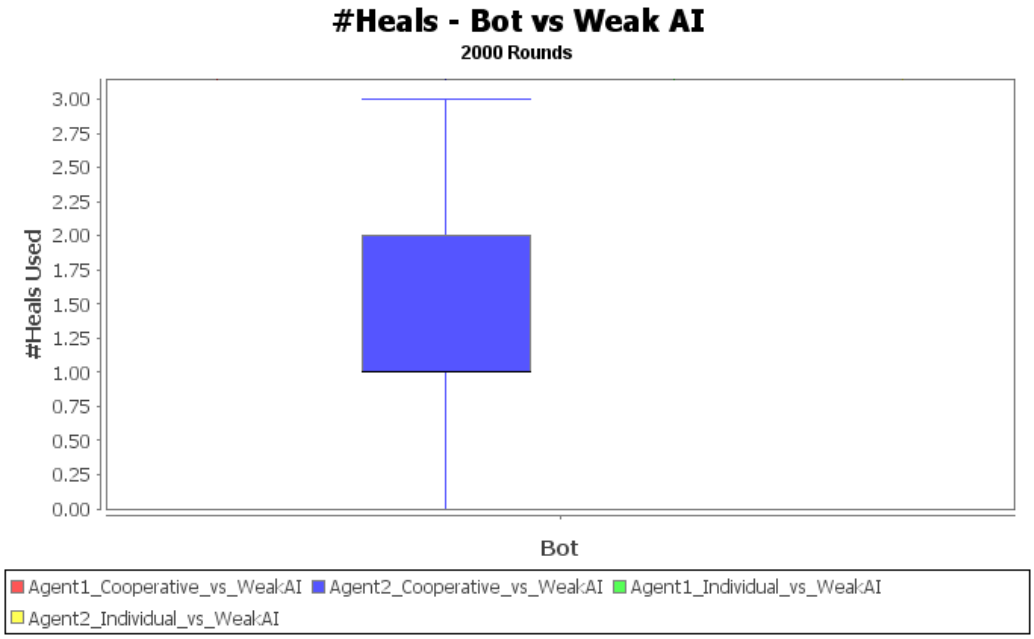

Figura 9 – Boxplot de quantas vezes cada unidade usou a cura no aliado na luta contra a IA fraca

Analisando os *boxplots* de número de curas usadas no aliado, exibidos nas Figuras 9 e 10 podemos observar que os agentes **Cooperativos** são muito mais propensos a curar seus aliados, enquanto esse comportamento só é perceptível no **Individualista** quando ele luta contra oponentes mais fortes. Os valores do agente 2 estão significativamente mais altos que os do agente 1 em todos os casos, de novo por conta da escolha de alvos da heurística, mas nesse caso a análise não parece ter sido prejudicada de maneira muito significativa.

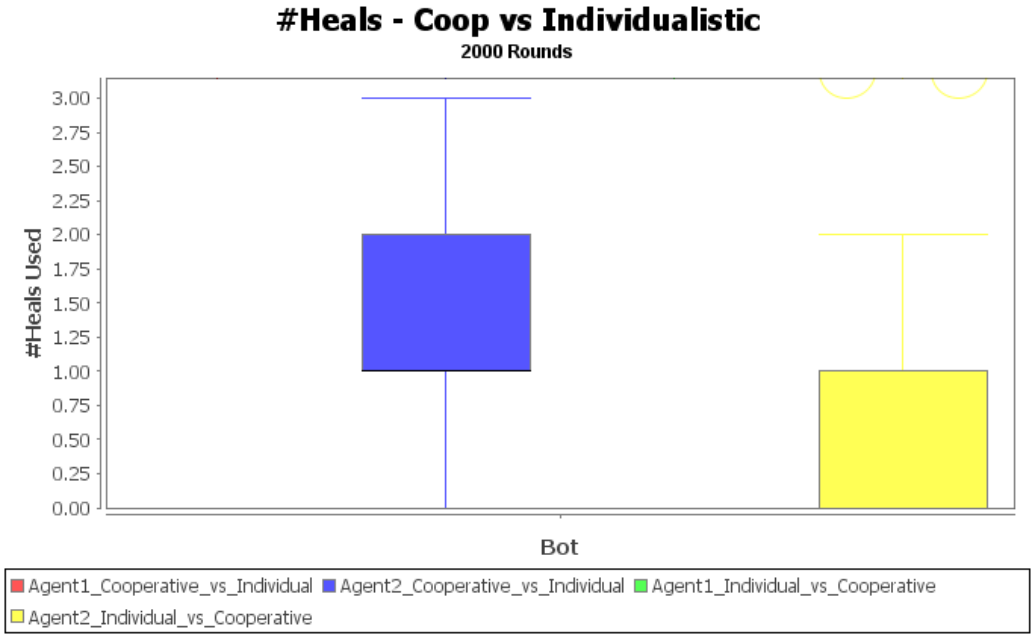

Figura 10 – Boxplot de quantas vezes cada unidade usou a cura no aliado na luta de Individualistas versus Cooperativos

3.3.3.4 Número de Stuns coordenados com o aliado

A análise da quantidade de *Stuns* coordenados, detalhada nos gráficos do apêndice B, mostra que o agente Individualista tem uma propensão muito maior a exibir esse tipo de comportamento em comparação com o Cooperativo, mas também mostra que o agente 2 do time Cooperativo tem uma tendência a exibir esse comportamento com menor frequência contra inimigos mais fortes, o que vai contra as expectativas. Vale questionar aqui se não existem mais parâmetros relevantes que, se coletados, poderiam explicar tal comportamento, como por exemplo o uso de *Stun* em uma unidade quando o aliado dela já está sendo afetado por um *Stun*, o que também poderia ser considerado um tipo de comportamento cooperativo.

3.4 Dificuldades e Limitações

Este trabalho começou como um projeto independente de intercâmbio e por conta disso foi desenvolvido usando uma versão relativamente antiga tanto da Unity (2017.4.32f) quanto do *plugin* ML-Agents(0.5.0a) e por conta de projetos de disciplinas da gradução e grupos de extensão não houve tempo hábil durante o semestre para atualizar o projeto sem arriscar passar do limite de prazo determinado para entrega. Ter tido que usar essas versões antigas fez ser necessário também usar versões antigas de python e Tensorflow, o que dificultou fazer a implementação usando o sistema operacional Windows. Felizmente foi possível configurar a máquina sob o sistema operacional Arch Linux após algumas tentativas.

Durante o intercâmbio a máquina que seria usada para desenvolver o projeto teve problemas e foi preciso usar uma máquina emprestada para o desenvolvimento. Foi quase impossível fazer o projeto executar nela de maneira correta e a única solução que realmente resolveu esse problema foi a utilização de imagens Docker[2], que apesar de iniciar o treino com uma *build* completa do jogo, não mostravam uma interface gráfica correspondente nem uma saída de *debug* de fácil acesso. Por conta desses contratempos e do pouco tempo livre no semestre seguinte ao intercâmbio, quando este trabalho foi iniciado(1 semestre após o fim do intercâmbio) muito tempo foi perdido encontrando os problemas já existentes antes que os treinos e análise pudessem de fato começar.

A escolha dos parâmetros de treino para o algoritmo PPO foi feita de maneira empírica e, por conta da falta de conhecimentos matemáticos profundos pode ser descrita analogamente a tatear em uma sala mal iluminada. Felizmente, o guia de documentação do ML-Agents no github[3] possui explicações para cada um dos parâmetros que devem ser configurados, bem como sugestões de valores para cada um deles. Isso foi o suficiente para ter um treinamento funcional, mas a falta de conhecimentos ainda impossibilita afirmar com convicção que essa escolha foi ótima.

[2] <https://www.docker.com/>
[3] <https://github.com/Unity-Technologies/ml-agents/blob/0.5.0a/docs/Training-PPO.md>

Como mencionado anteriormente, a quantidade de tempo disponível durante o semestre era limitado devido a outros elementos da graduação e após obter valores considerados satisfatórios para os parâmetros de treino, o número de episódios foi incrementado para garantir um resultado melhor, resultando em treinos de 8 horas mesmo utilizando uma GPU para que eles acelerassem. Por conta disso, só foi possível no tempo fornecido obter 3 modelos de cada tipo, quando o ideal seria aumentar esse número para ter garantia de qualidade ou ainda extender o tempo de treino, já que como pôde ser visto nos gráficos apresentados nos Resultados(3.3) o valor da recompensa ainda não estava completamente estabilizado e apontava sinais de crescimento.

A coleta de dados poderia ter levado em conta alguns outros paramêtros que também podem ser indicadores de comportamento cooperativo, como por exemplo o uso do *Stun* em uma unidade quando a outra unidade do mesmo time já estava sendo afetada pela habilidade.

Por fim, durante a análise dos dados coletados foi possível perceber que a heurística estava configurada para sempre priorizar a unidade 1 ao escolher seu alvo, e isso afetou de maneira inesperada os valores obtidos na coleta. Por exemplo, não foi possível avaliar se o comportamento dos agentes resultava em uma diferença maior ou menor entre o hp restante no final da batalha dentro de um mesmo time pois o comportamento da heurística causava uma discrepância por padrão, tornando os dados não confiaveis para essa avaliação.

Capítulo 4

CONCLUSÃO

O treino dos dois tipos de modelos foi realizado com sucesso, ainda que não esteja necessariamente no melhor valor possível. Foi possível, através da análise dos dados, observar que o agente Individualista desenvolveu sim comportamentos cooperativos, mas esses comportamentos não são explícitos tão claramente quanto no agente Cooperativo, sendo perceptível apenas nas situações em que ele mais tem dificuldade e, consequentemente, necessidade deles.

4.1 Trabalhos Futuros

Como foi descrito na seção de dificuldades(3.4), o projeto apresenta problemas de fácil correção ou possibilidade de melhora com apenas mais tempo livre para os treinos e nova coleta de dados. Idealmente, esses erros seriam corrigidos e uma avaliação do comportamento dos modelos gerados seria feito com a utilização de uma pesquisa com seres humanos jogando contra os modelos visando publicação de artigo.

Posteriormente uma revisão do código para deixá-lo mais legível e comentado ao invés de apenas funcional, bem como uma atualização do projeto para utilizar versões mais atuais da Unity e do ML-Agents beneficiariam qualquer pessoa que deseje reutilizá-lo a partir do github. Com o mesmo intuito pode ser feito um guia detalhado na wiki do projeto no github, listando problemas de uso mais comuns.

4.2 Feedback do Curso

Por ser um aluno de transferência, minha evolução no curso não pode ser considerada como padrão e minha perspectiva é diferente da de alunos que entraram diretamente através da FUVEST ou do ENEM. No meu curso anterior(Engenharia Elétrica, Enfase em Eletrônica), já havia cursado as disciplinas de cálculo e outras disciplinas de matemática. Por ter pego essas equivalências meus primeiros semestres ficaram com vários espaços livres na grade, que foram usados para cursar optativas de semestres mais avançados.

Percebi que meus semestres costumavam sem bem menos cansativos do que os de meus colegas pela ausência das disciplinas de matemática, e isso permitiu que me dedicasse mais ao grupo de extensão que fazia parte, uma experiência que acredito ter sido muito benéfica. Além disso consegui me planejar para fazer um ano de intercâmbio através de um edital da AUCANI,

sendo que o processo de inscrição só foi completado com sucesso pela ajuda da Comissão de Relações Internacionais do ICMC.

Ao longo do curso tive aulas com professores bons e ruins, mas a possibilidade de escolher a turma na matrícula e a disponibilidade de tempo extra que tive fez com que a grande maioria das disciplinas tenham sido experiências agradáveis. Não me acho capaz de fazer críticas ou elogios mais relevantes do que aqueles comentados durante a mesa redonda promovida pela Semcomp 22 e acredito que mais eventos como aquele gerariam um ambiente de ensino e aprendizado mais saudáveis para alunos e professores.

REFERÊNCIAS

AL-JARRAH, O. Y.; YOO, P. D.; MUHAIDAT, S.; KARAGIANNIDIS, G. K.; TAHA, K. Efficient machine learning for big data: A review. **Big Data Research**, Elsevier, v. 2, n. 3, p. 87–93, 2015. Citado na página 29.

GRANVILLE, K. Facebook and cambridge analytica: What you need to know as fallout widens. **The New York Times**, v. 19, p. 18, 2018. Citado na página 21.

JULIANI, A.; BERGES, V.; VCKAY, E.; GAO, Y.; HENRY, H.; MATTAR, M.; LANGE, D. Unity: A general platform for intelligent agents. **CoRR**, abs/1809.02627, 2018. Disponível em: <http://arxiv.org/abs/1809.02627>. Citado na página 26.

LIAKOS, K. G.; BUSATO, P.; MOSHOU, D.; PEARSON, S.; BOCHTIS, D. Machine learning in agriculture: A review. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 18, n. 8, p. 2674, 2018. Citado na página 29.

MAXWELL, A. E.; WARNER, T. A.; FANG, F. Implementation of machine-learning classification in remote sensing: An applied review. **International Journal of Remote Sensing**, Taylor & Francis, v. 39, n. 9, p. 2784–2817, 2018. Citado na página 29.

MOSAVI, A.; OZTURK, P.; CHAU, K.-w. Flood prediction using machine learning models: Literature review. **Water**, Multidisciplinary Digital Publishing Institute, v. 10, n. 11, p. 1536, 2018. Citado na página 29.

Nam, S.; Ikeda, K. Generation of diverse stages in turn-based role-playing game using reinforcement learning. In: **2019 IEEE Conference on Games (CoG)**. [S.l.: s.n.], 2019. p. 1–8. Citado na página 30.

OPENAI. **OpenAI Five**. 2018. <https://blog.openai.com/openai-five/>. Citado 3 vezes nas páginas 21, 22 e 30.

RAJKOMAR, A.; DEAN, J.; KOHANE, I. Machine learning in medicine. **New England Journal of Medicine**, Mass Medical Soc, v. 380, n. 14, p. 1347–1358, 2019. Citado na página 21.

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. **CoRR**, abs/1707.06347, 2017. Disponível em: <http://arxiv.org/abs/1707.06347>. Citado na página 26.

SILVEY, P. E.; TOLK, A.; TRACY, B. A. Applying complexity science with machine learning, agent-based models, and game engines: Towards embodied complex systems engineering. In: SPRINGER. **Unifying Themes in Complex Systems IX: Proceedings of the Ninth International Conference on Complex Systems**. [S.l.], 2018. p. 173. Citado na página 30.

SONI, B.; HINGSTON, P. Bots trained to play like a human are more fun. In: IEEE. **2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)**. [S.l.], 2008. p. 363–369. Citado na página 21.

APÊNDICE A

ESTATÍSTICAS DE TREINO

Estatísticas de Treinos das Redes Neurais

Neste apêndice estão todos os gráficos contendo as estatísticas de treino das redes neurais.

A.1 Individualista

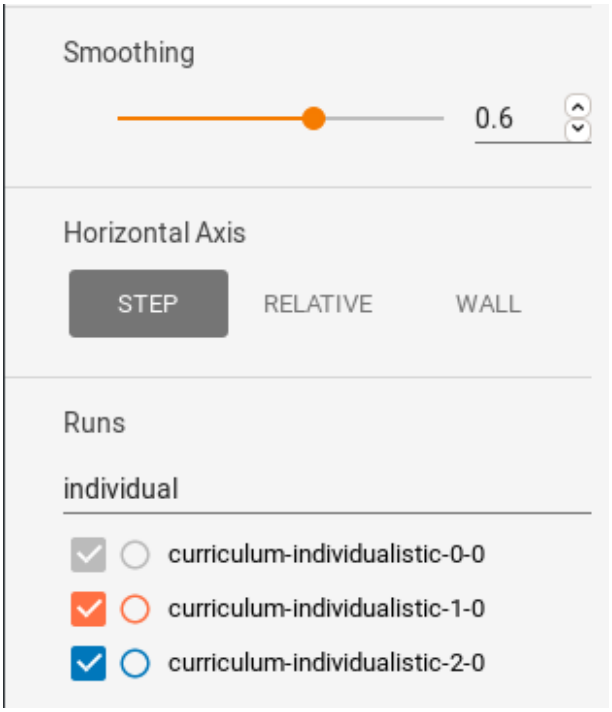

Figura 11 – Código de cores para os gráficos de treino das redes neurais Individualistas

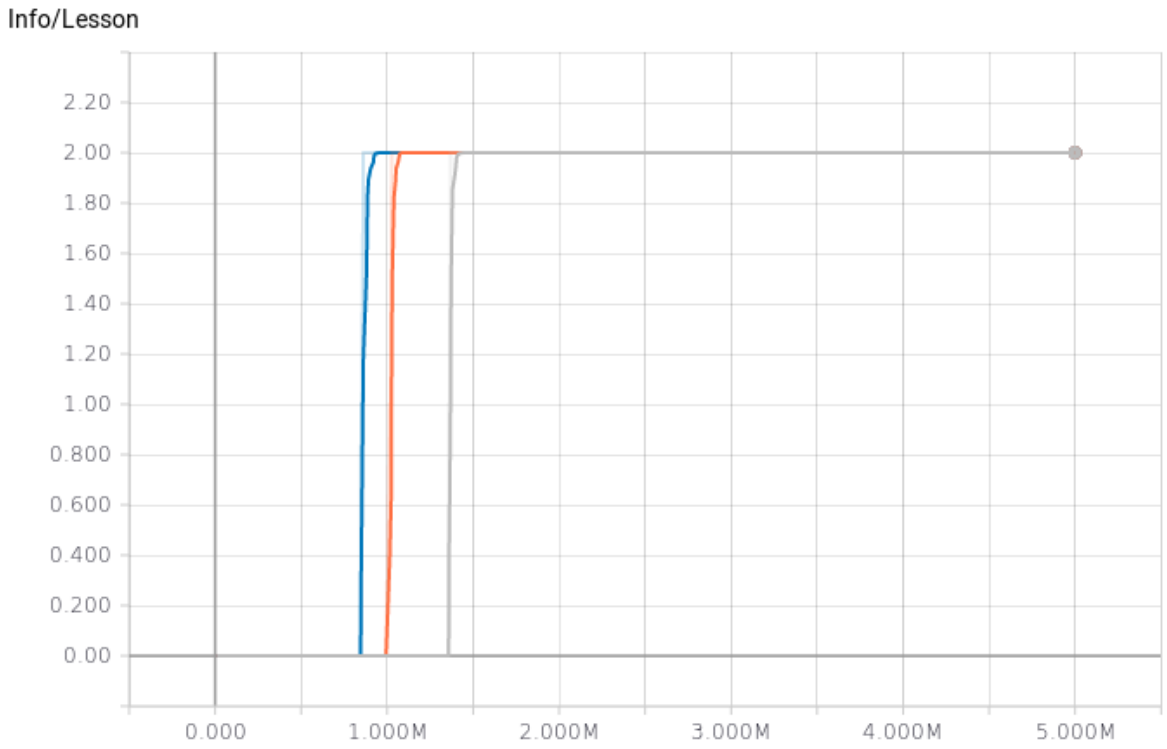

Figura 12 – Progressão de currículo das redes neurais Individualistas

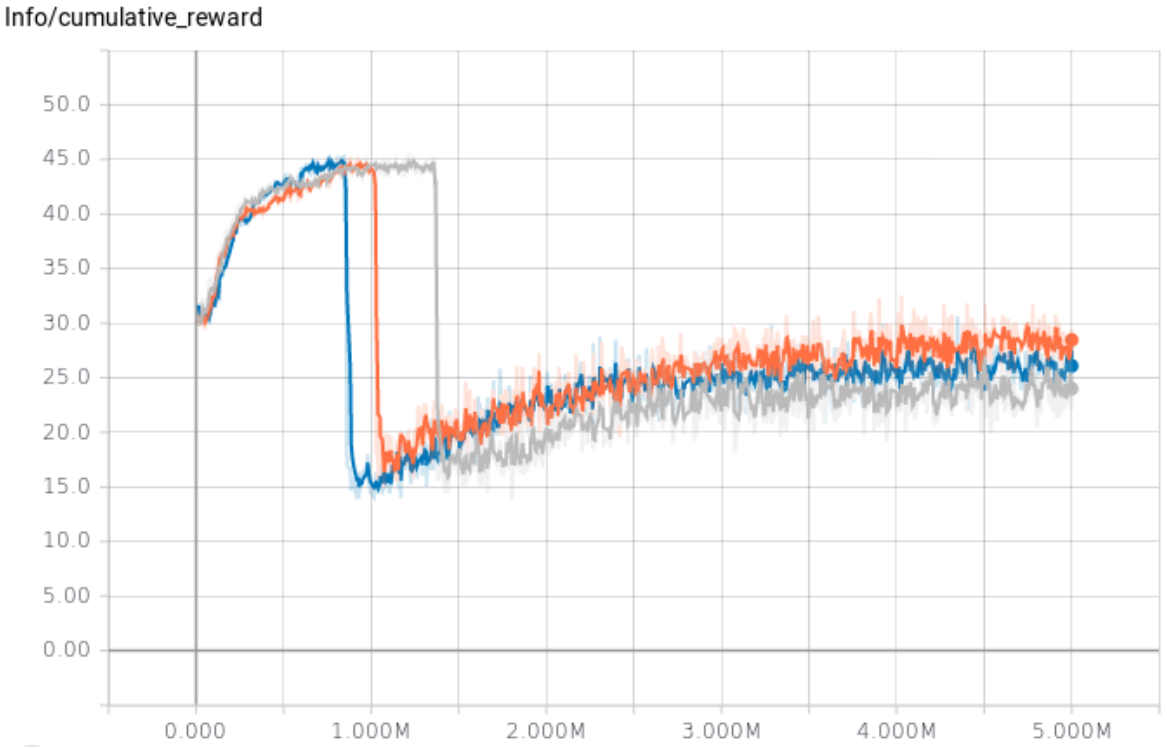

Figura 13 – Progressão de recompensa das redes neurais Individualistas

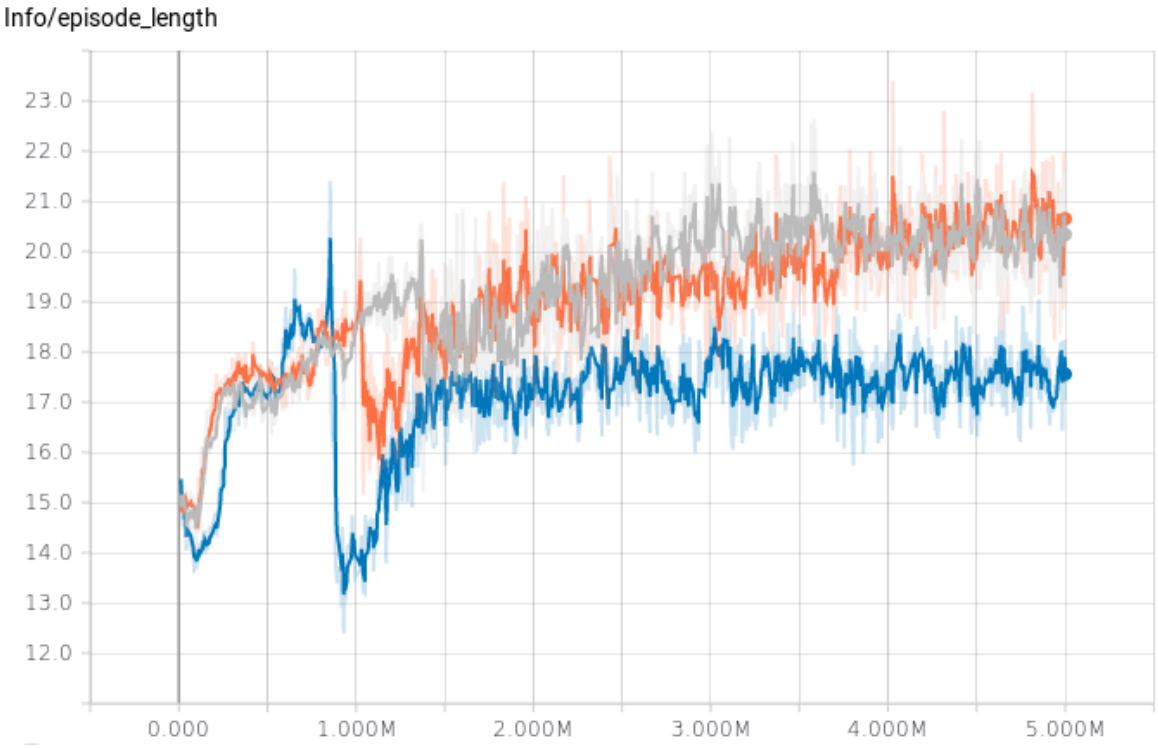

Figura 14 – Duração média de cada batalha em turnos para o treino das redes neurais Individualistas

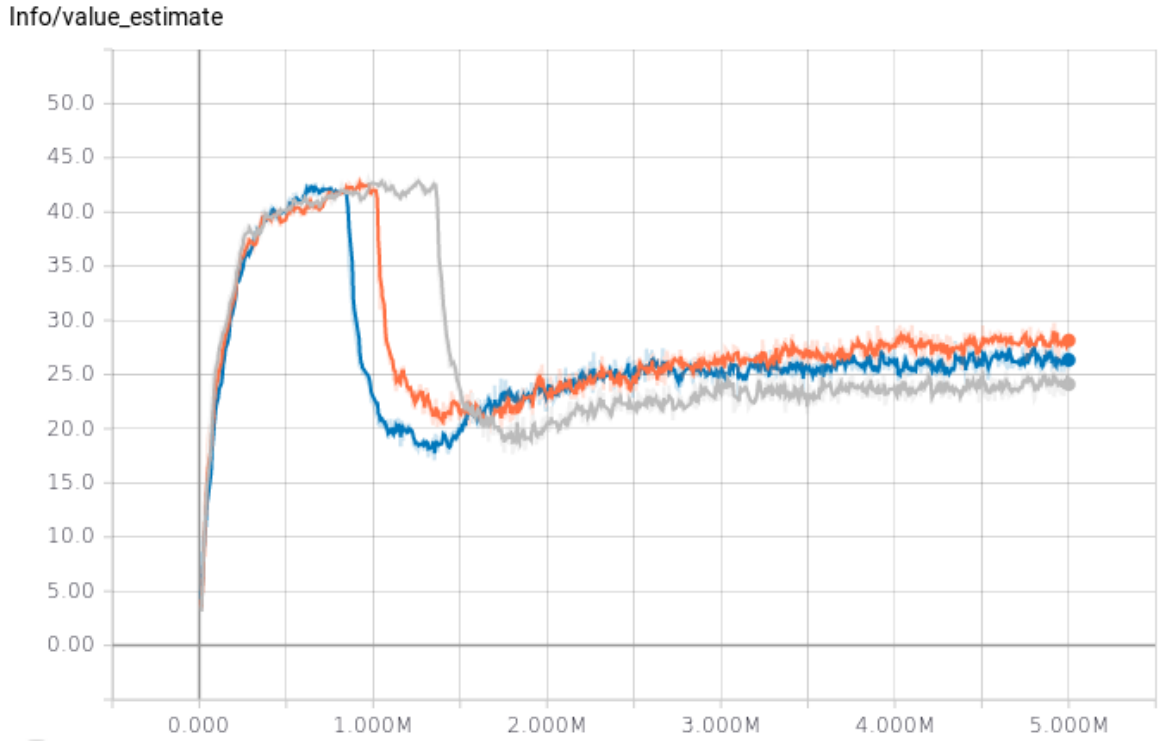

Figura 15 – Média das estimativas de recompensa das redes neurais Individualistas ao longo do treino

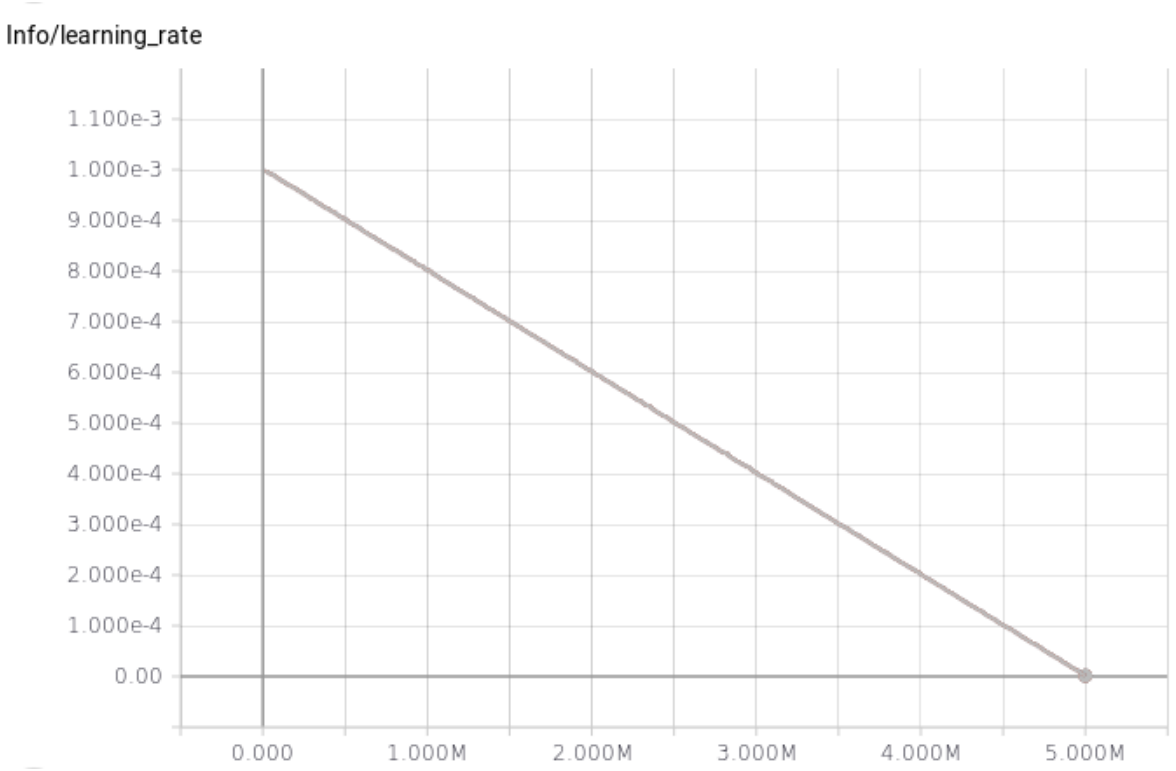

Figura 16 – Taxa de busca para o algoritmo de otimização ao longo do treino das redes neurais Individualistas

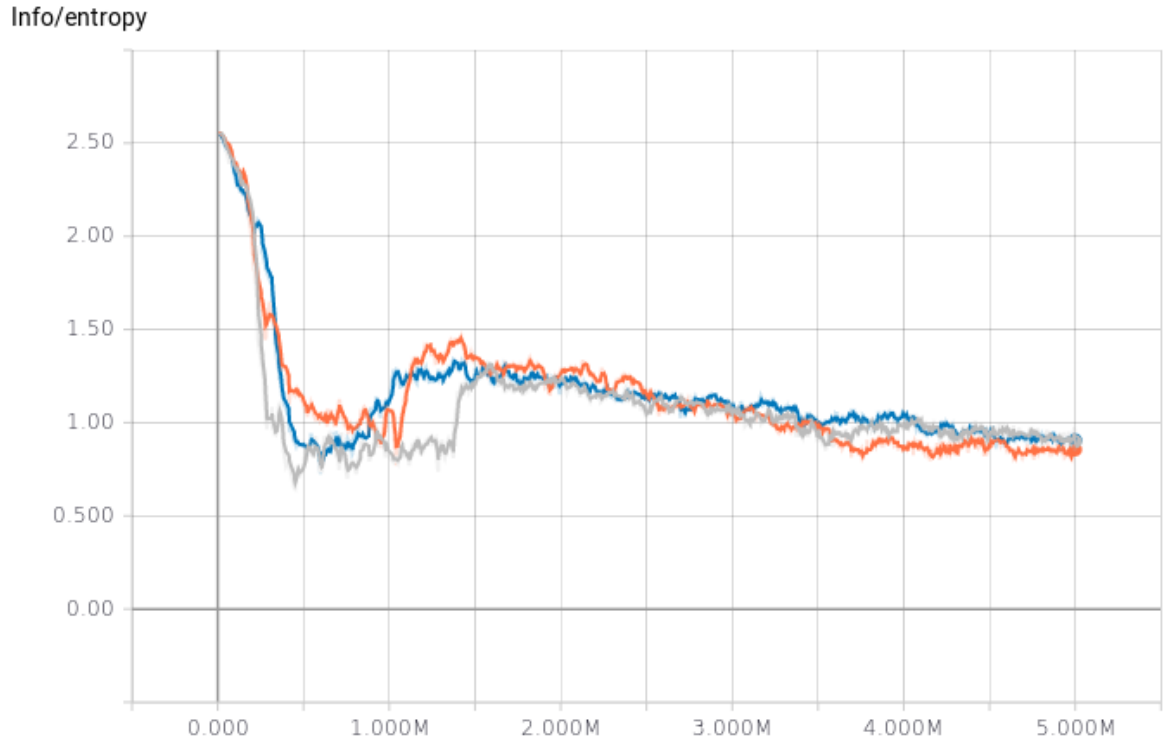

Figura 17 – Aleatoriedade das decisões das redes neurais Individualistas ao longo do treino

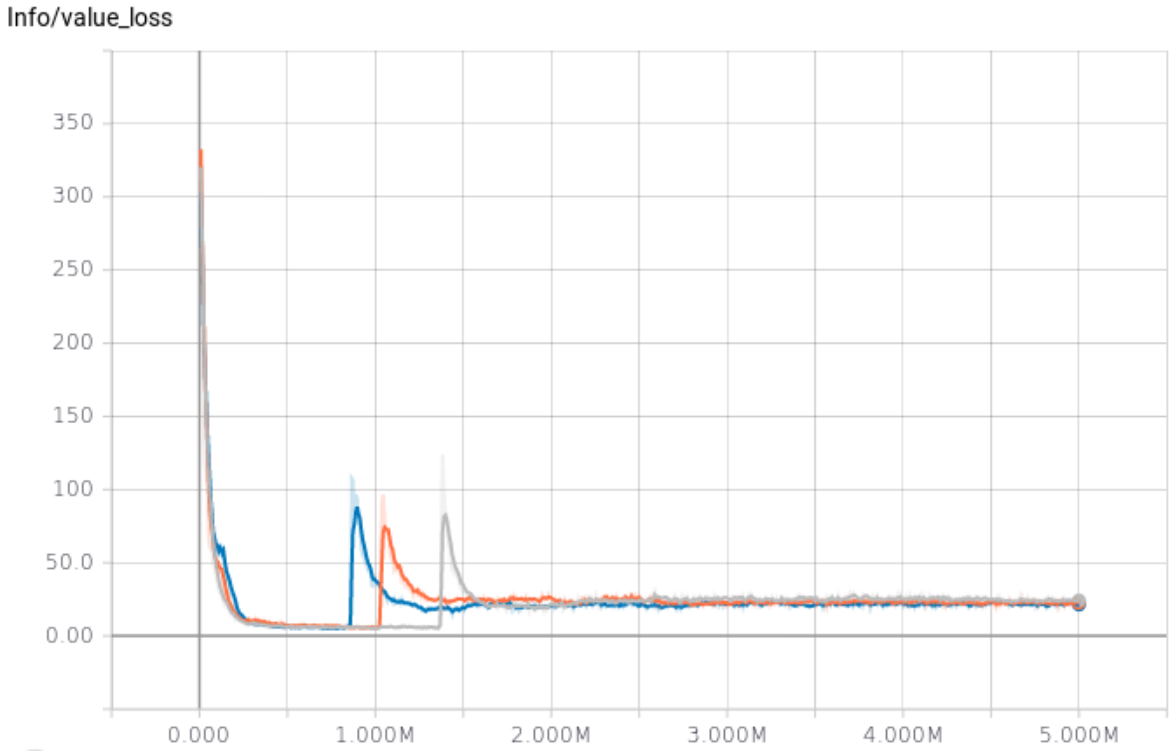

Figura 18 – Erros de estimativa das redes neurais Individualistas ao longo do treino

A.2 Cooperativa

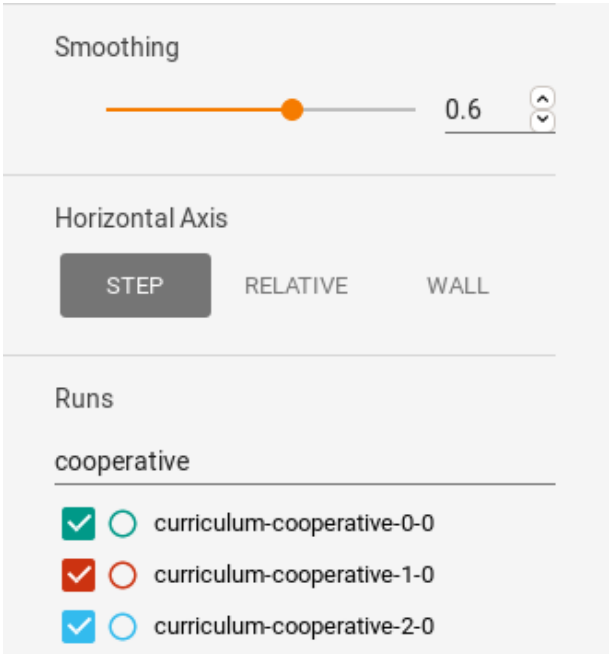

Figura 19 – Código de cores para os gráficos de treino das redes neurais Cooperativas

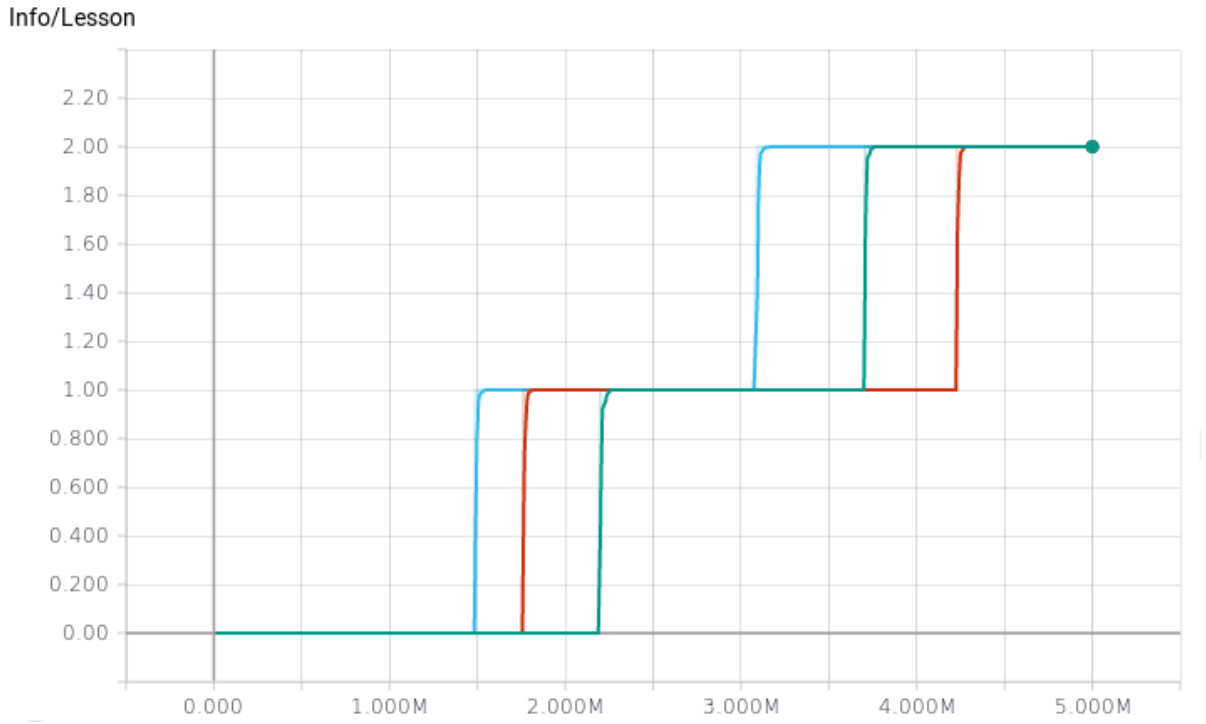

Figura 20 – Progressão de currículo das redes neurais Cooperativas

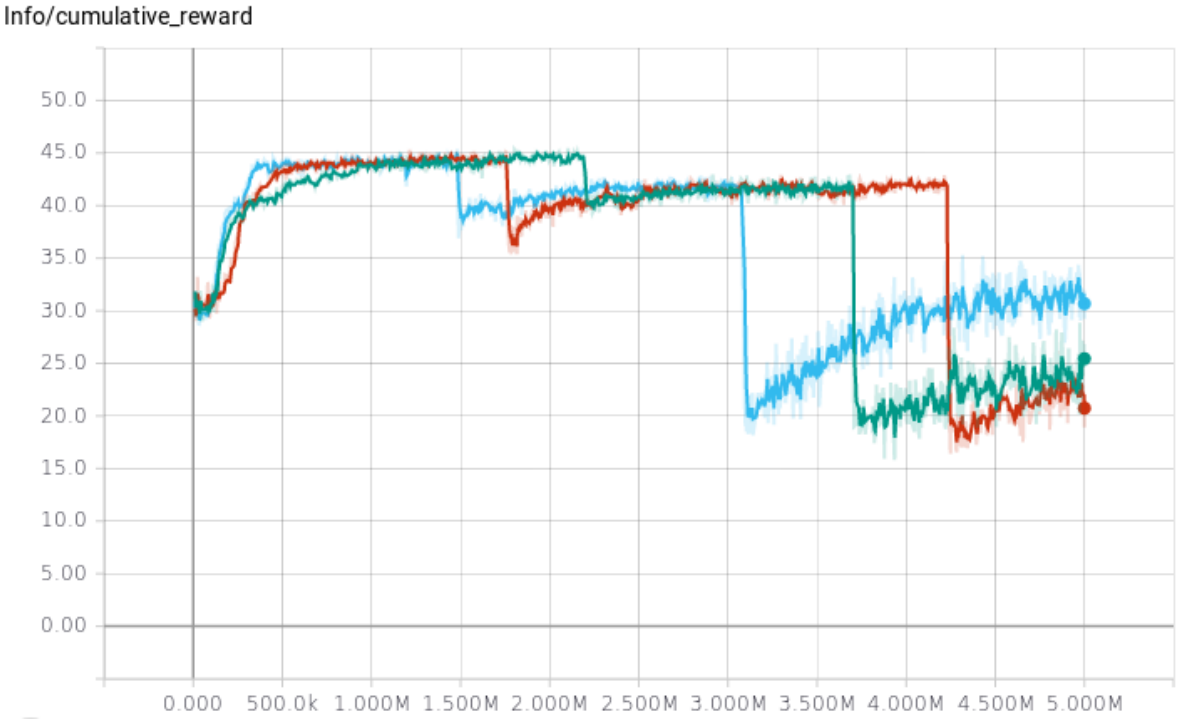

Figura 21 – Progressão de recompensa das redes neurais Cooperativas

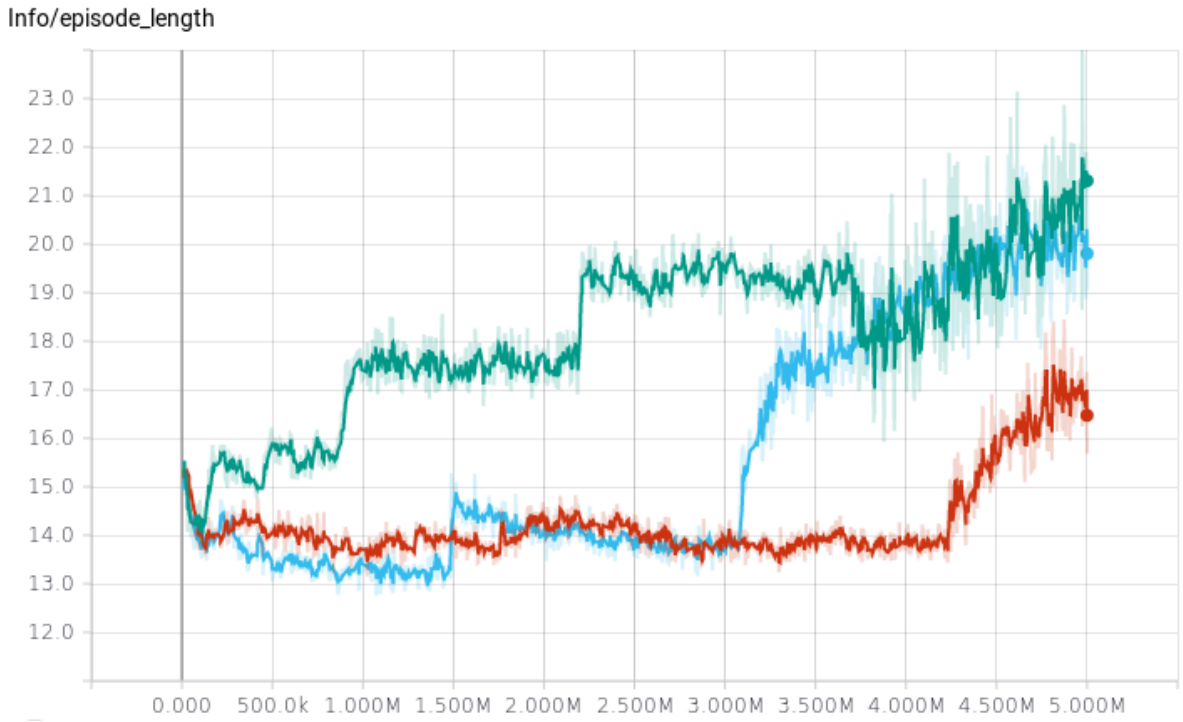

Figura 22 – Duração média de cada batalha em turnos para o treino das redes neurais Cooperativas

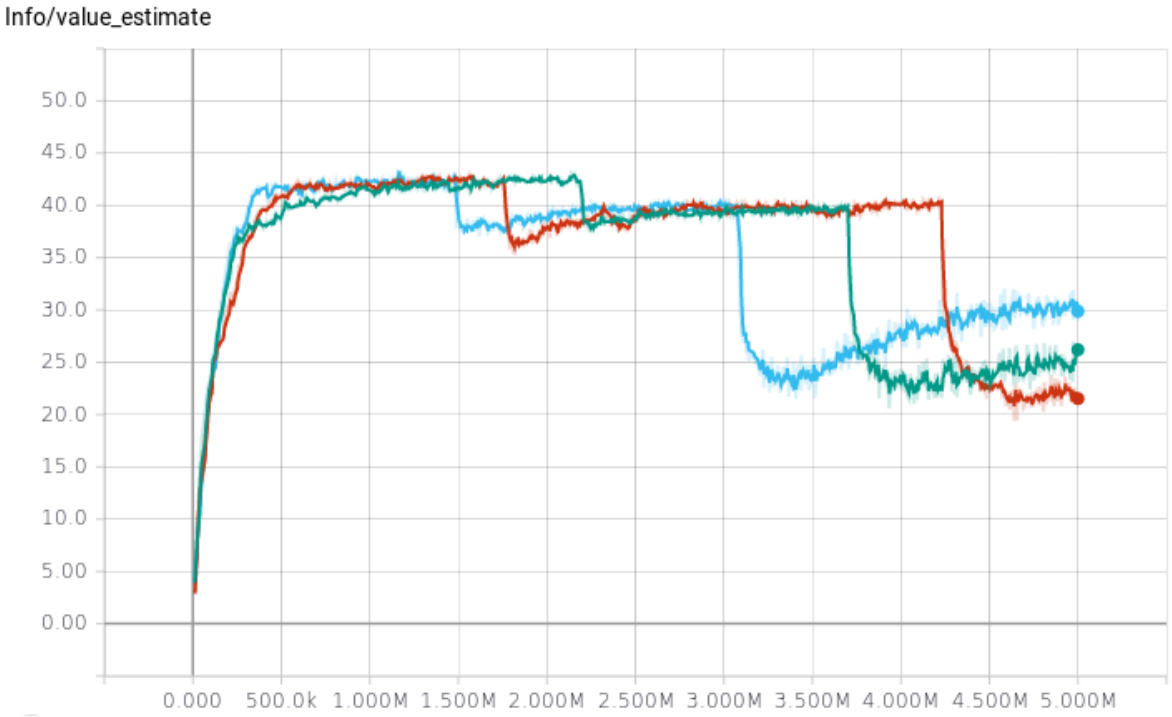

Figura 23 – Média das estimativas de recompensa das redes neurais Cooperativas ao longo do treino

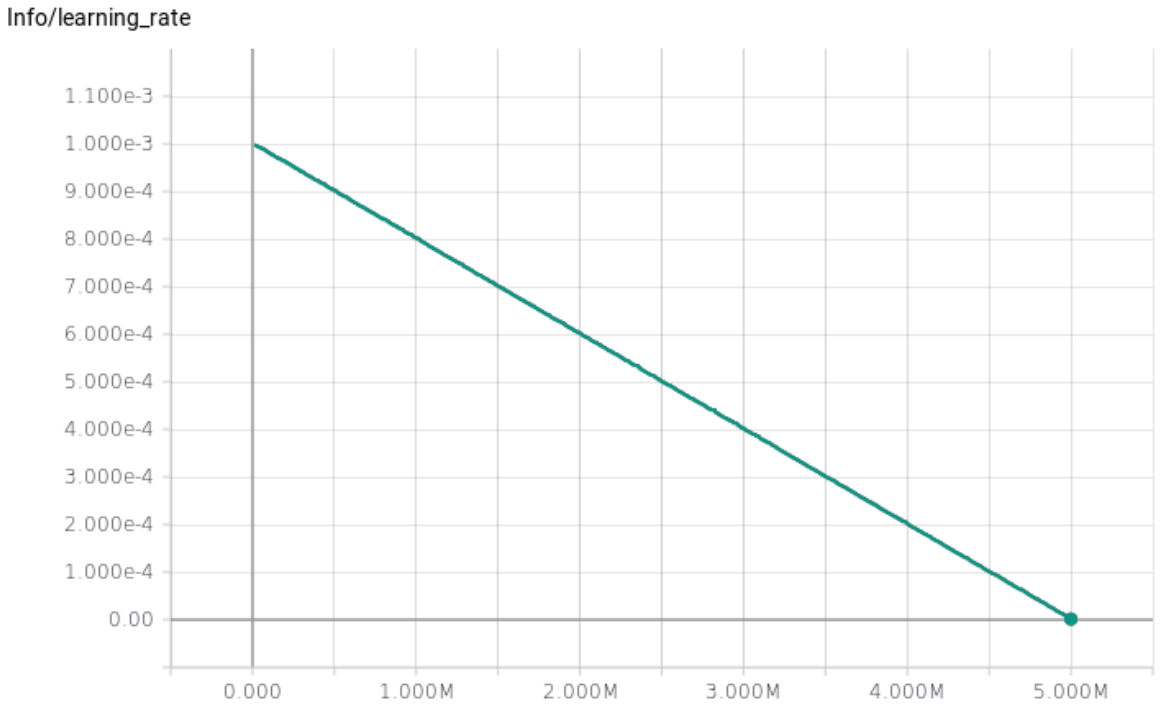

Figura 24 – Taxa de busca para o algoritmo de otimização ao longo do treino das redes neurais Cooperativas

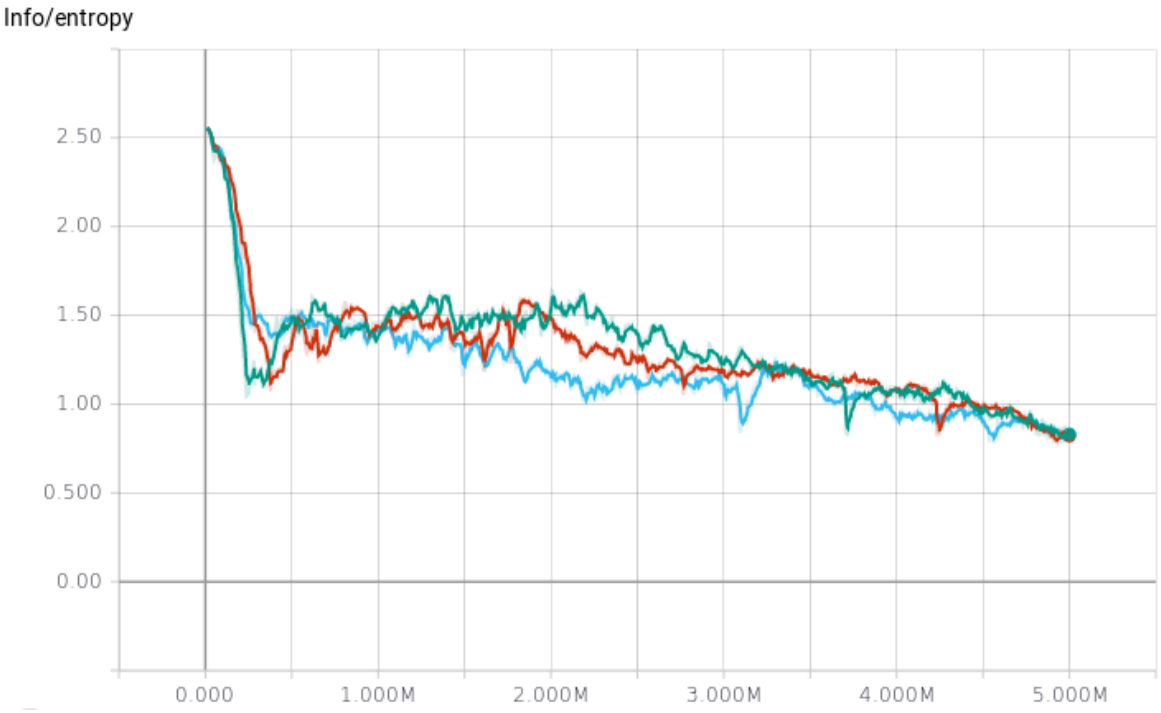

Figura 25 – Aleatoriedade das decisões das redes neurais Cooperativas ao longo do treino

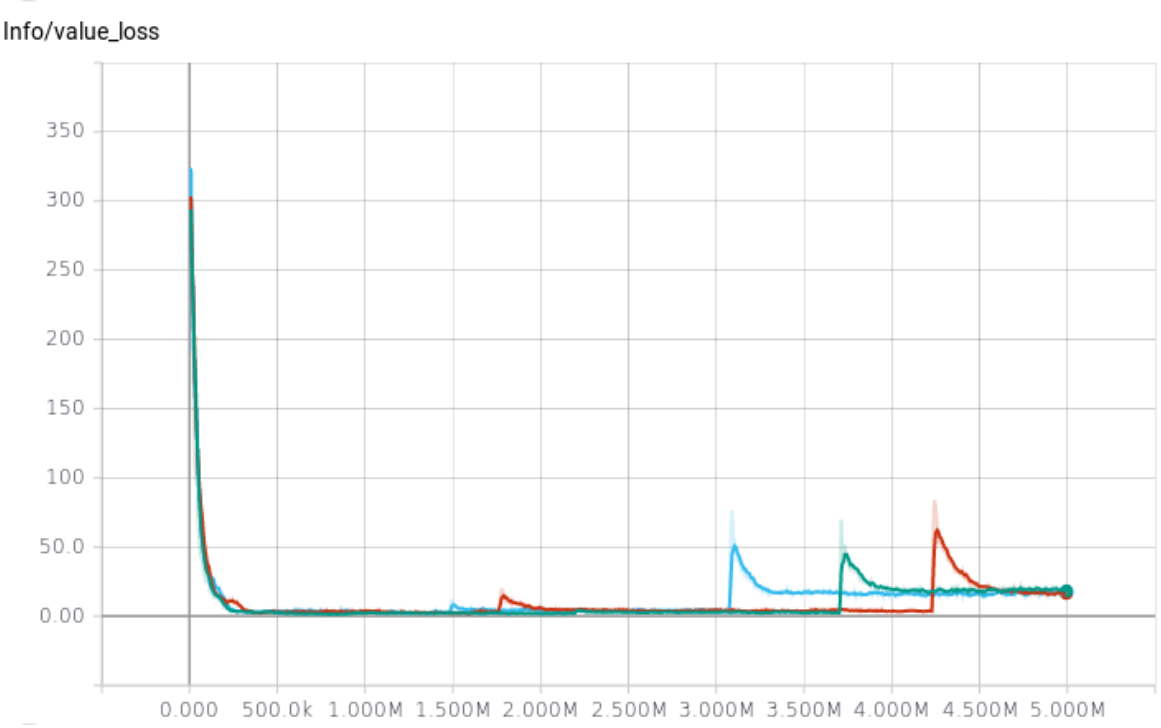

Figura 26 – Erros de estimativa das redes neurais Cooperativas ao longo do treino

APÊNDICE B

RESULTADOS DETALHADOS

Análise dos Combates

Neste apêndice estão todos os *boxplots* gerados com os combates das redes neurais, incluindo os que foram considerados irrelevantes para a pesquisa.

B.1 Recompensas

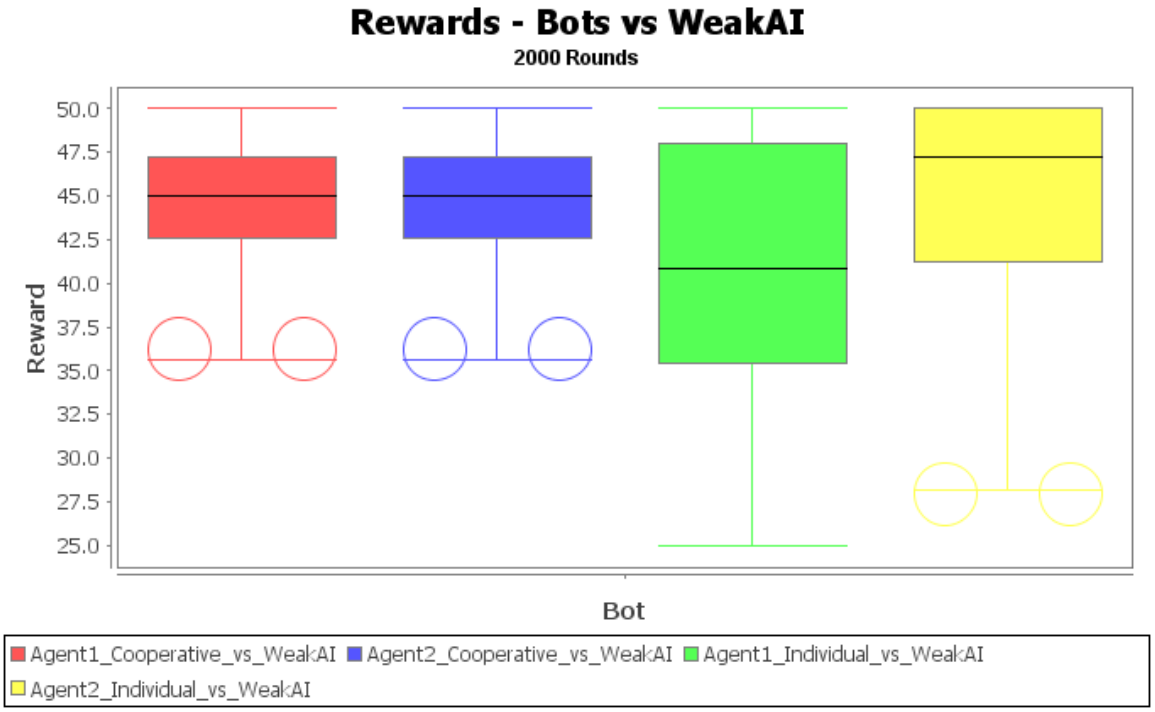

Figura 27 – Boxplot das recompensas de todos os agentes ao lutar contra a IA fraca(1)

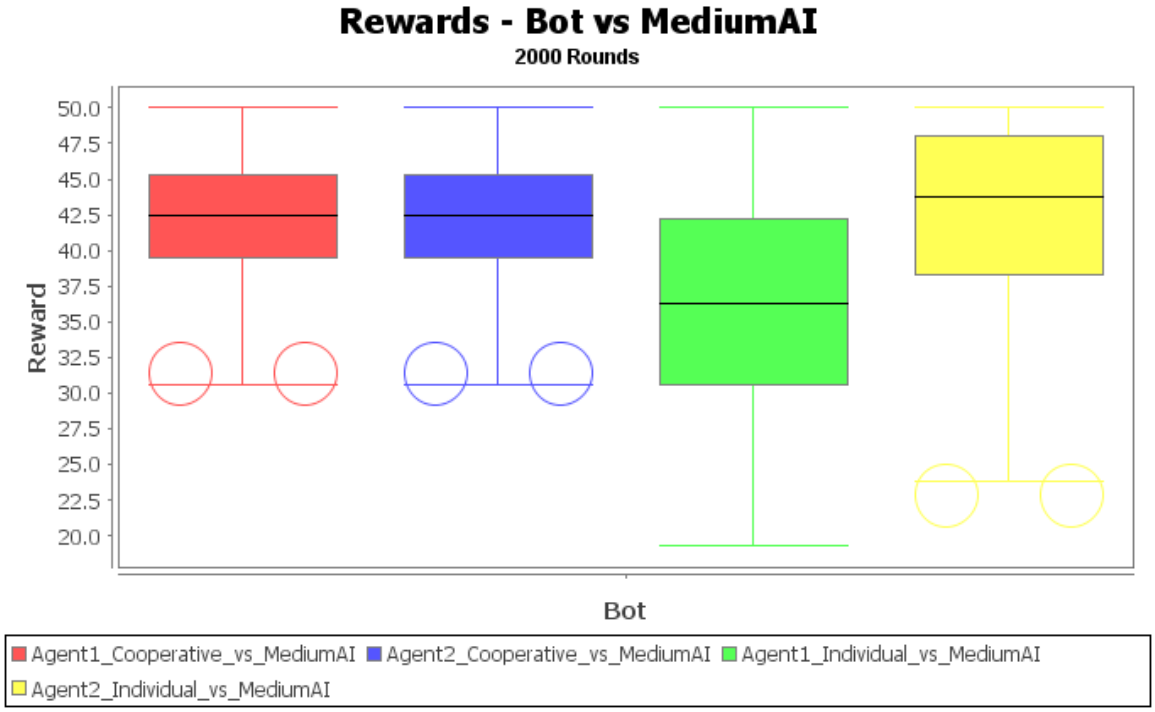

Figura 28 – Boxplot das recompensas de todos os agentes ao lutar contra a IA média

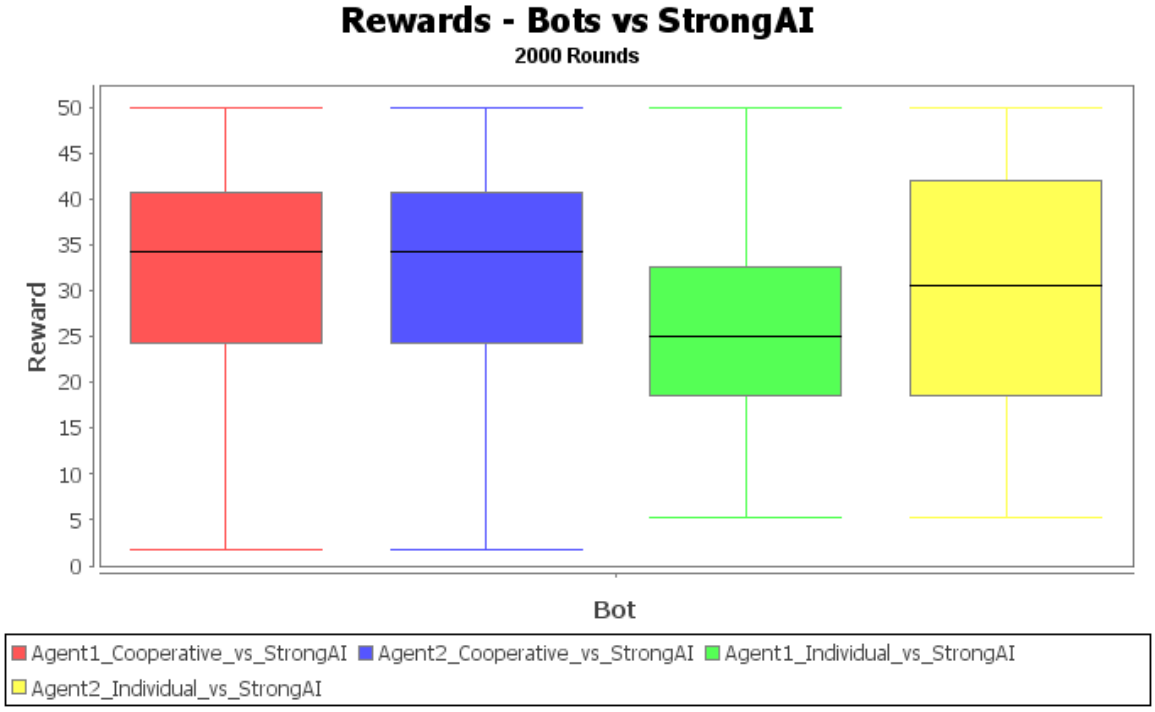

Figura 29 – Boxplot das recompensas de todos os agentes ao lutar contra a IA forte(1)

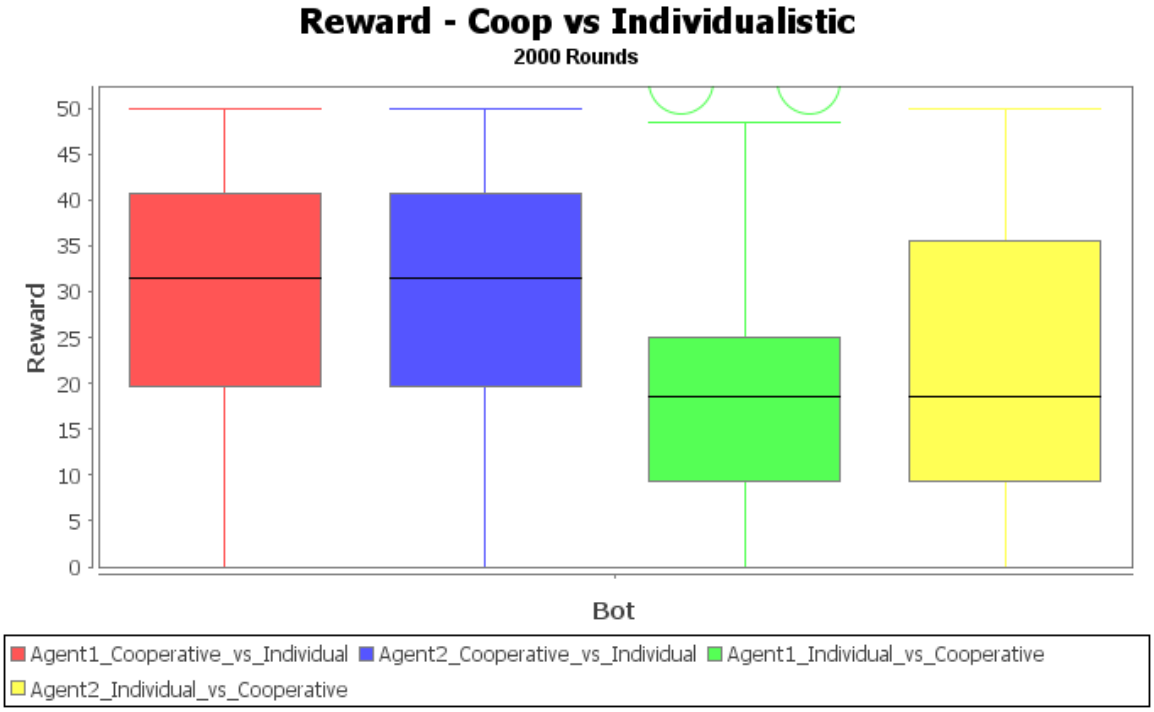

Figura 30 – Boxplot das recompensas de todos os agentes na luta de Individualistas versus Cooperativos(1)

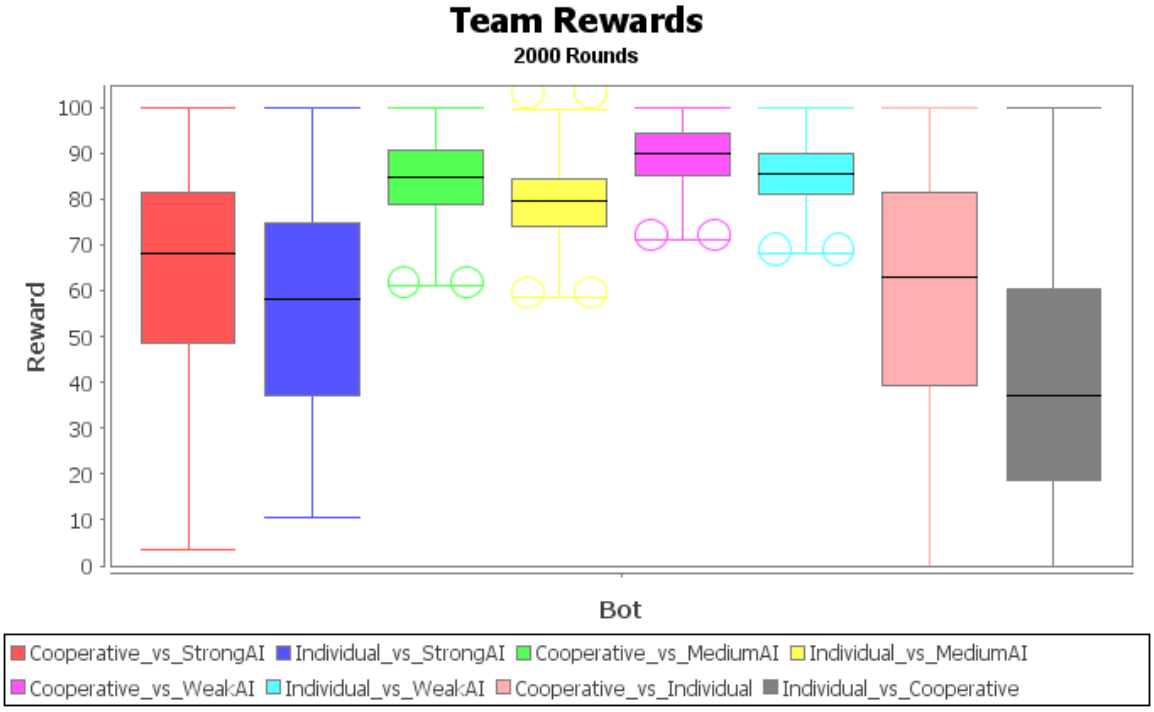

Figura 31 – Boxplot da soma de recompensas de cada time em todas as lutas realizadas

B.2 HP Restante

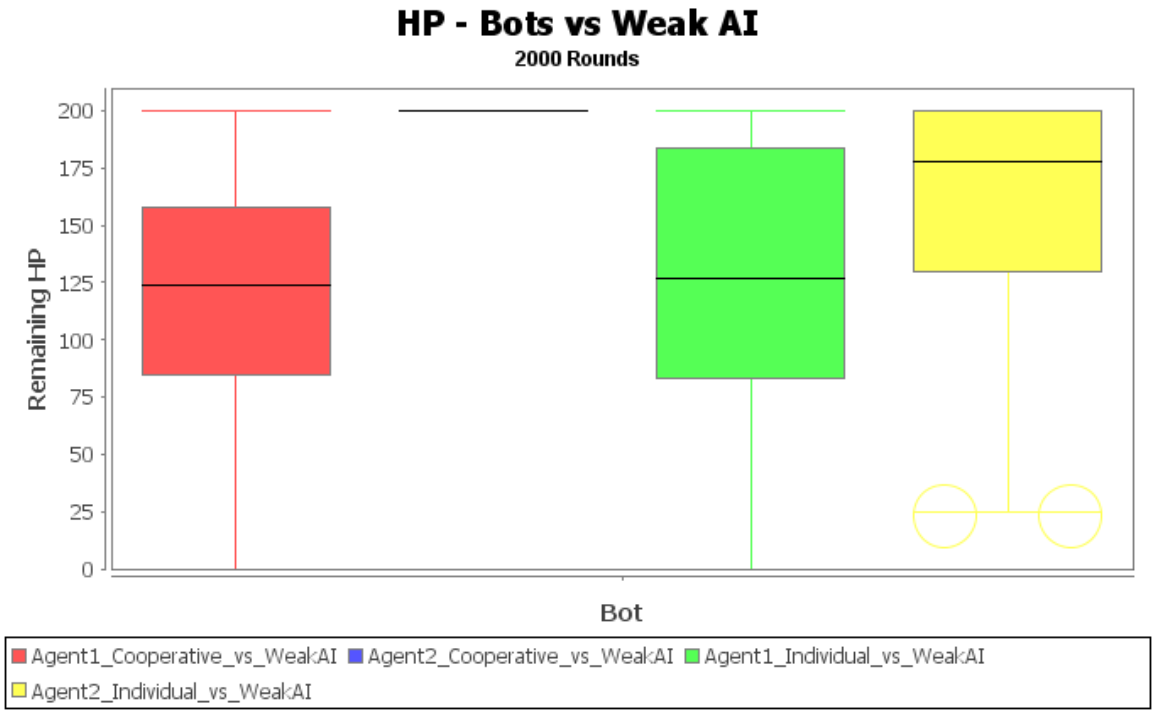

Figura 32 – Boxplot do HP restante de todos os agentes ao lutar contra a IA fraca

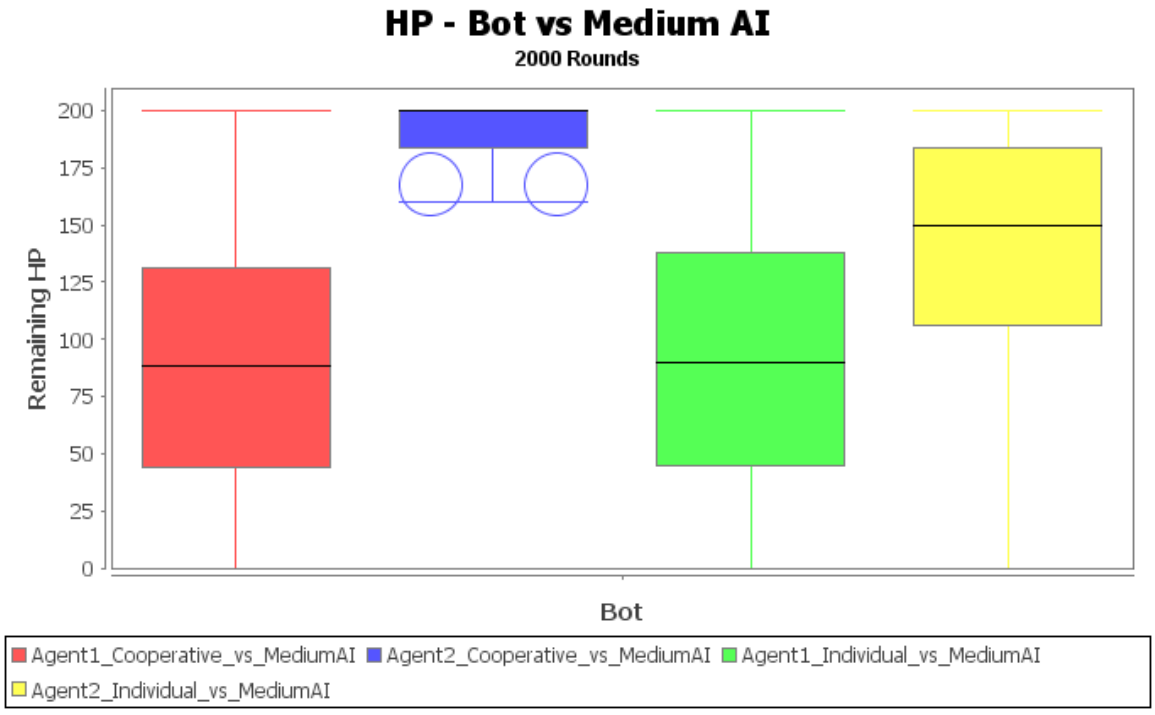

Figura 33 – Boxplot do HP restante de todos os agentes ao lutar contra a IA média

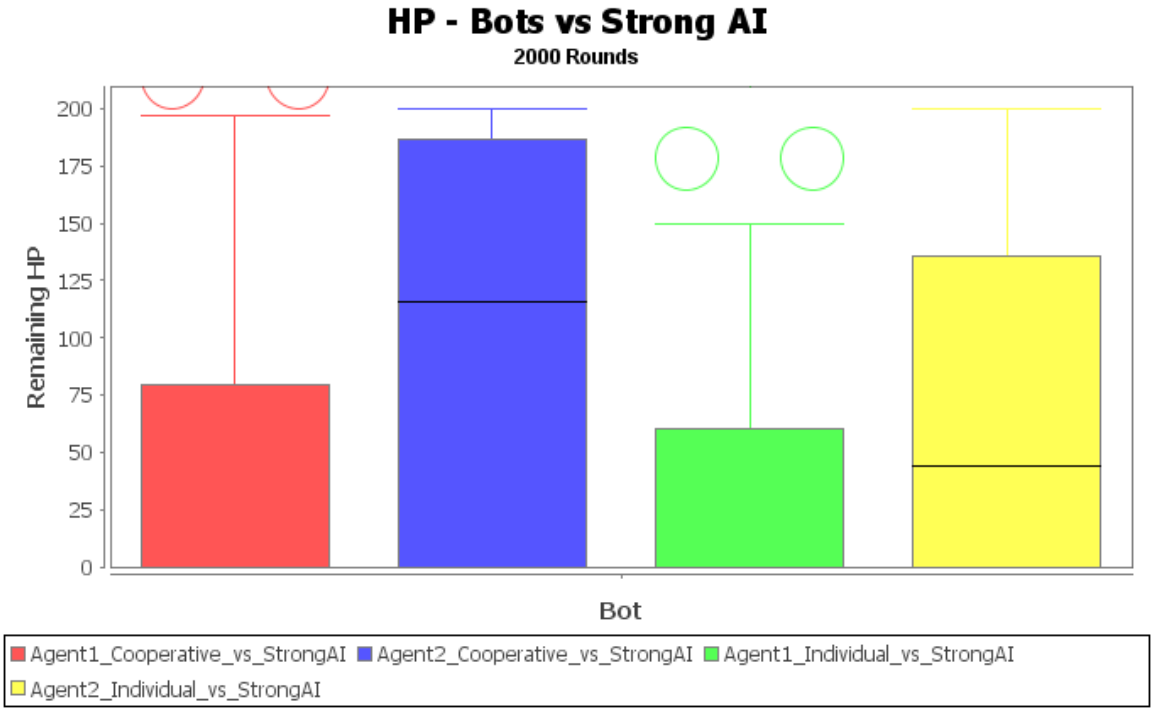

Figura 34 – Boxplot do HP restante de todos os agentes ao lutar contra a IA forte(1)

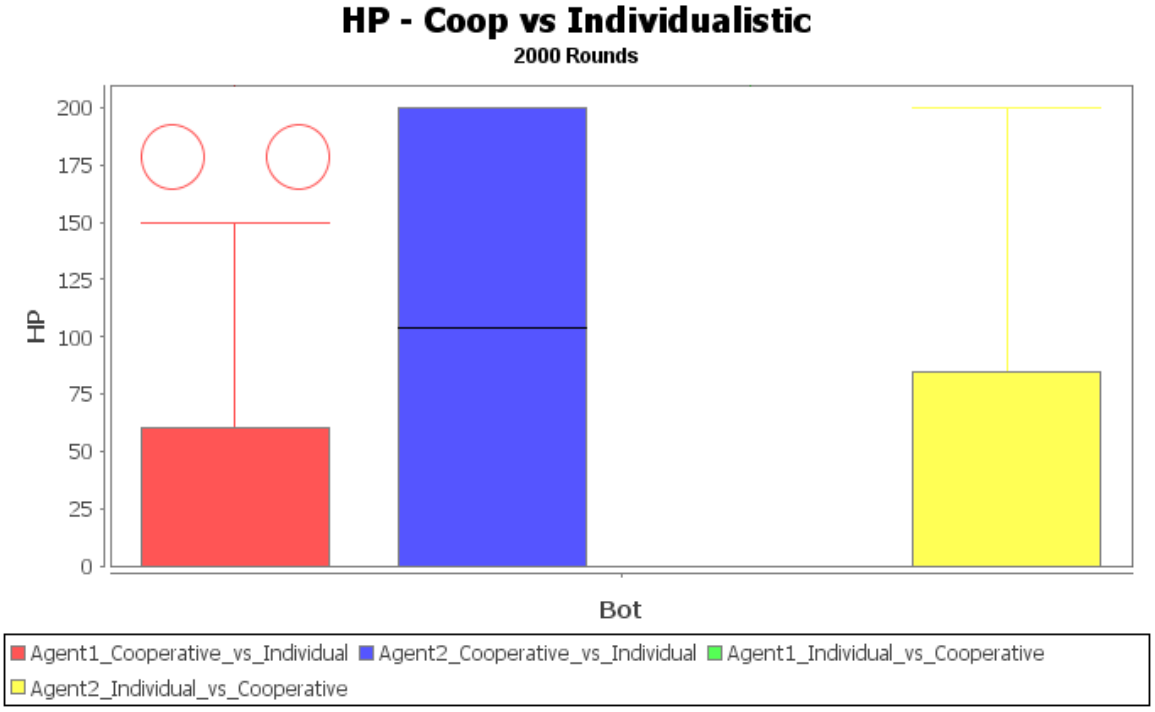

Figura 35 – Boxplot do HP restante de todos os agentes na luta de Individualistas versus Cooperativos

B.3 Número de curas usadas no aliado

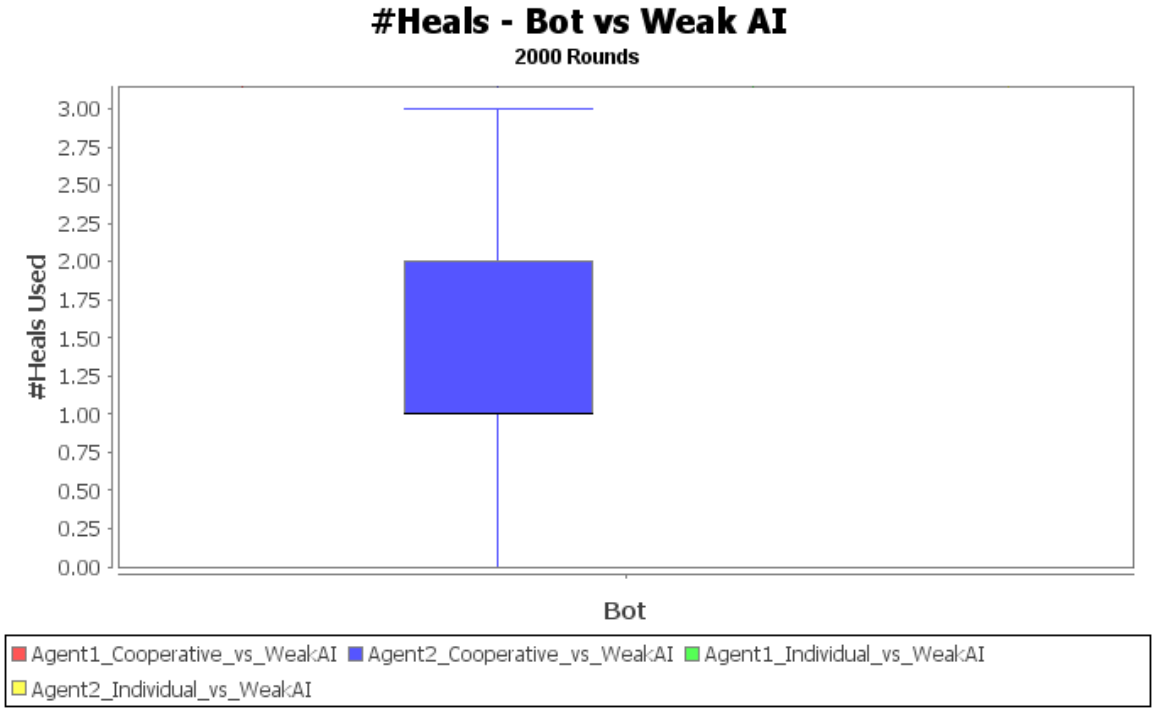

Figura 36 – Boxplot de quantas vezes cada unidade usou a cura no aliado na luta contra a IA fraca(1)

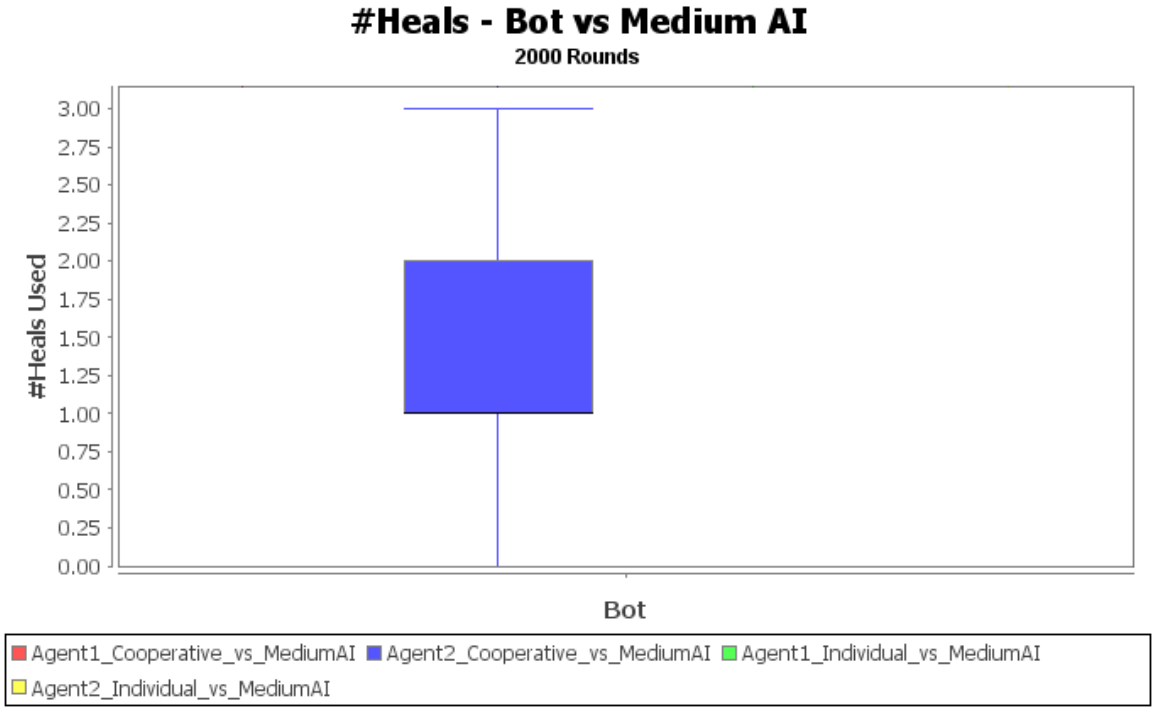

Figura 37 – Boxplot de quantas vezes cada unidade usou a cura no aliado na luta contra a IA média

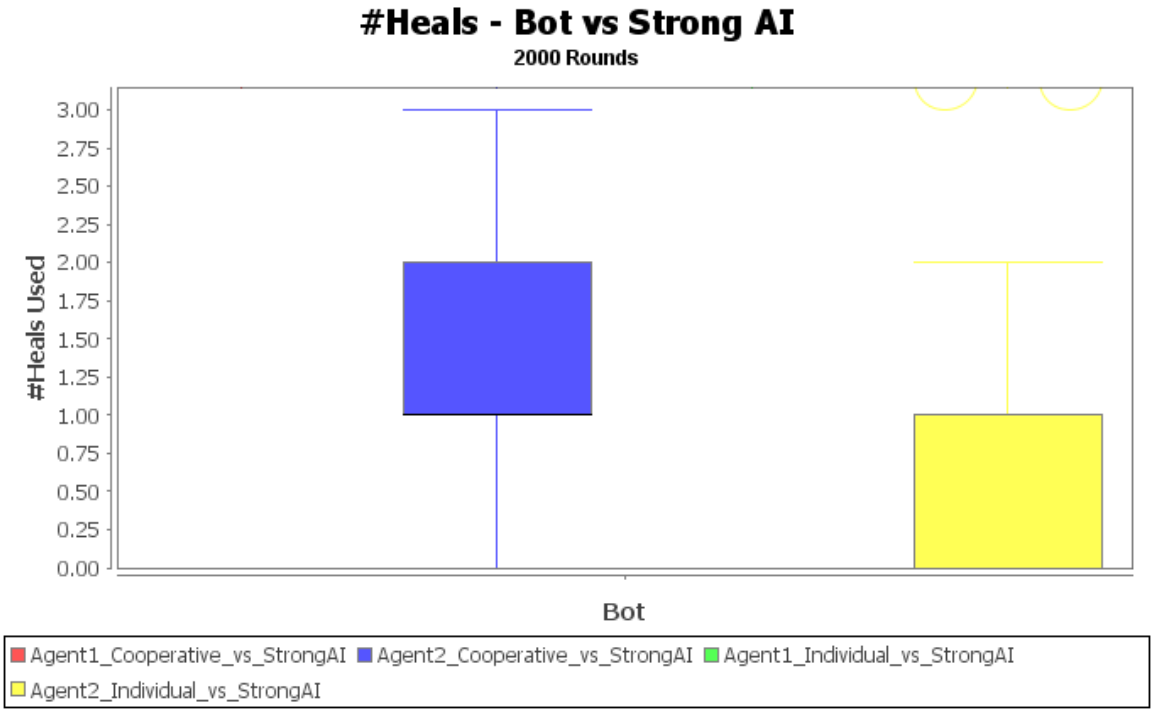

Figura 38 – Boxplot de quantas vezes cada unidade usou a cura no aliado na luta contra a IA forte

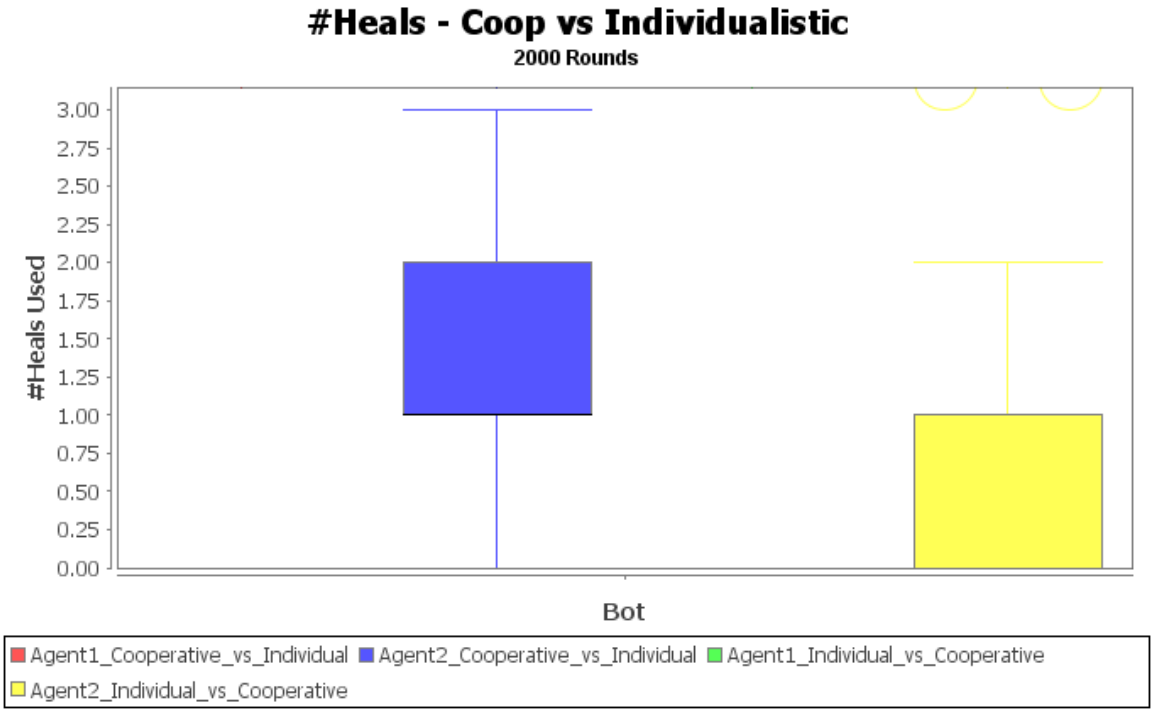

Figura 39 – Boxplot de quantas vezes cada unidade usou a cura no aliado na luta de Individualistas versus Cooperativos(1)

B.4 Número de *Stuns* coordenados com o aliado

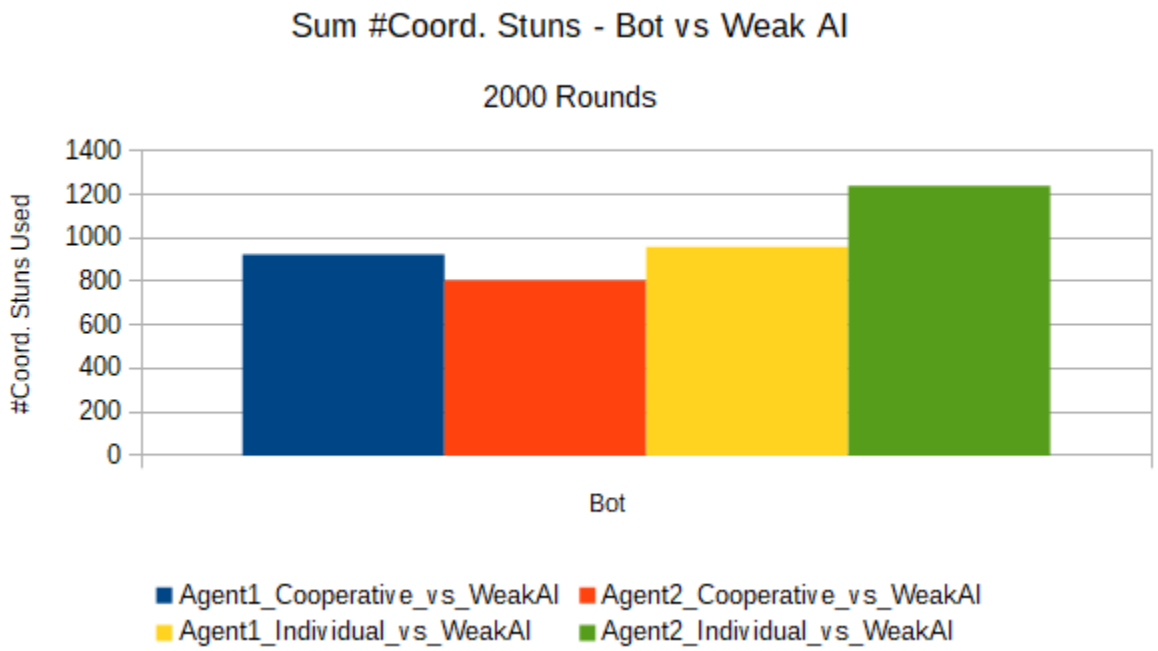

Figura 40 – Barplot de quantas vezes cada unidade usou *Stun* em um inimigo de modo a prolongar a duração durante a luta contra a IA fraca

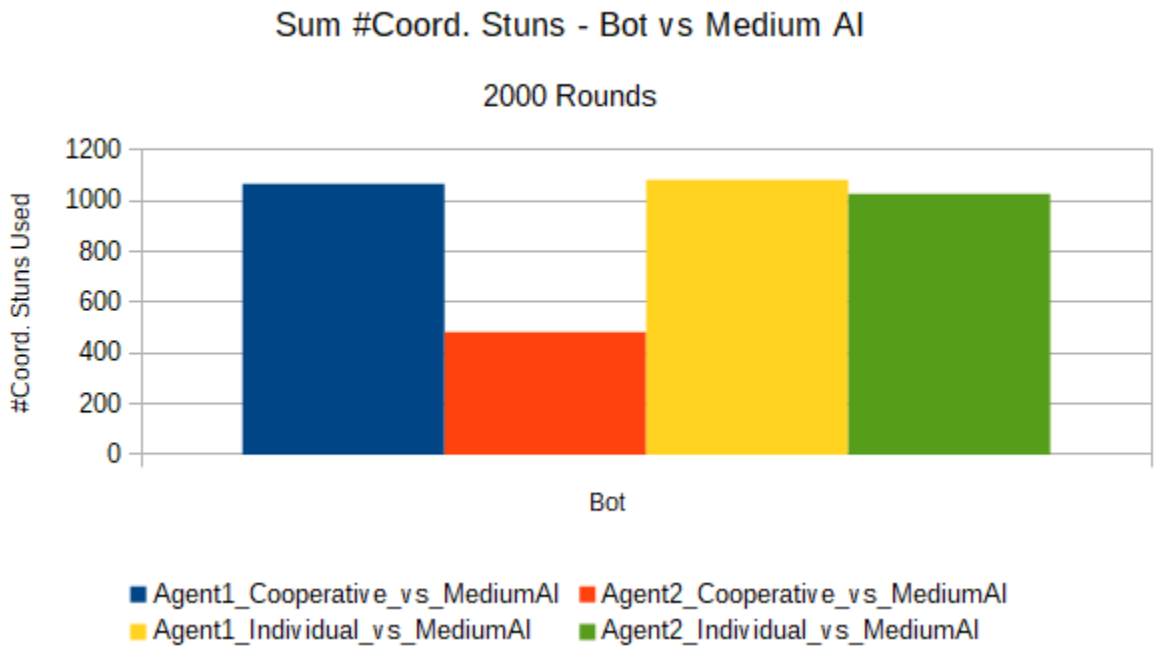

Figura 41 – Barplot de quantas vezes cada unidade usou *Stun* em um inimigo de modo a prolongar a duração durante a luta contra a IA média

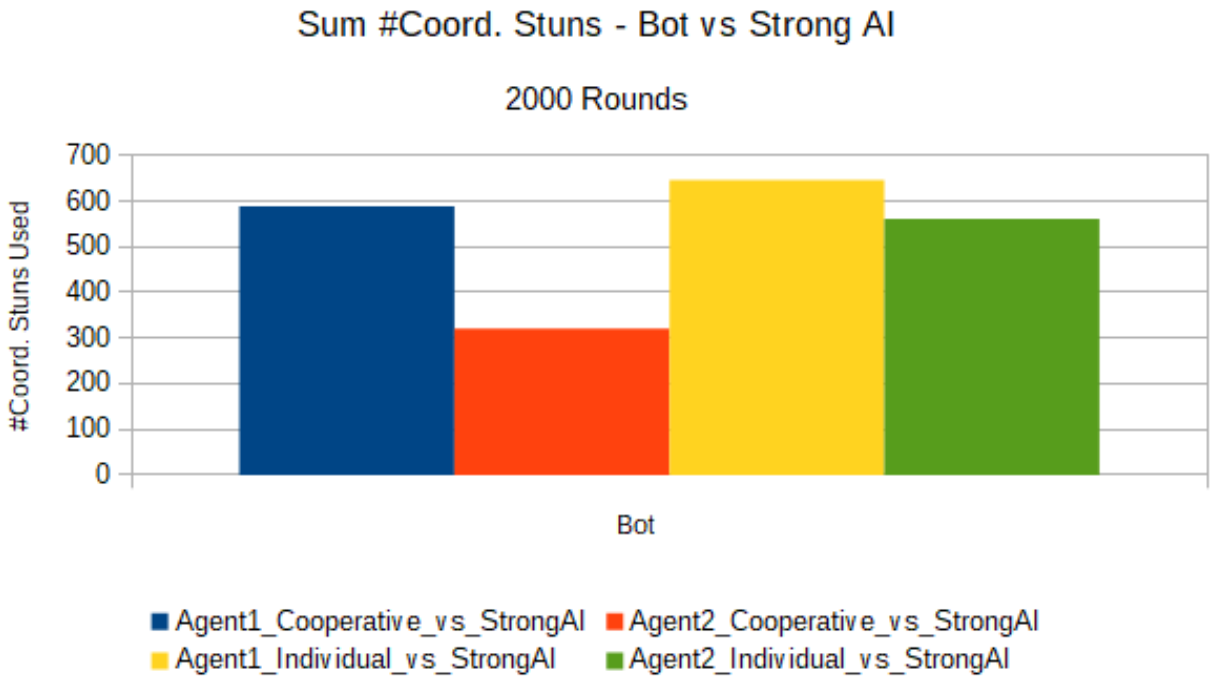

Figura 42 – Barplot de quantas vezes cada unidade usou *Stun* em um inimigo de modo a prolongar a duração durante a luta contra a IA forte

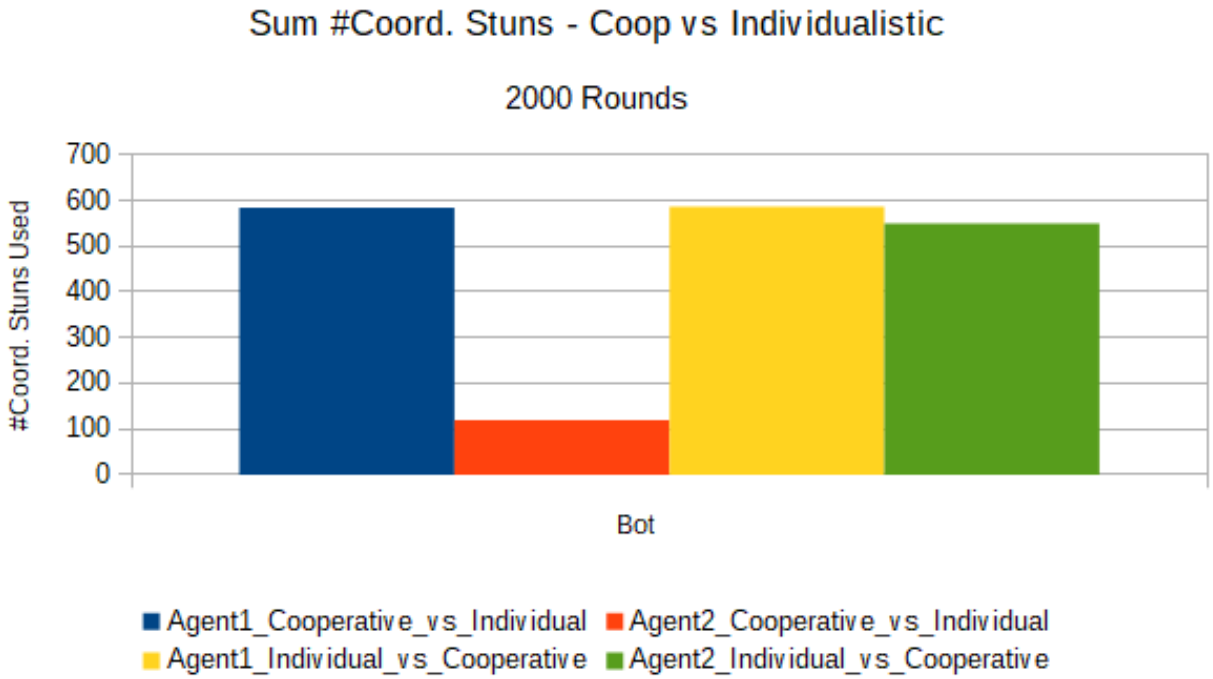

Figura 43 – Barplot de quantas vezes cada unidade usou *Stun* em um inimigo de modo a prolongar a duração na luta de Individualistas versus Cooperativos

B.5 Número de *Blinds* coordenados com o aliado

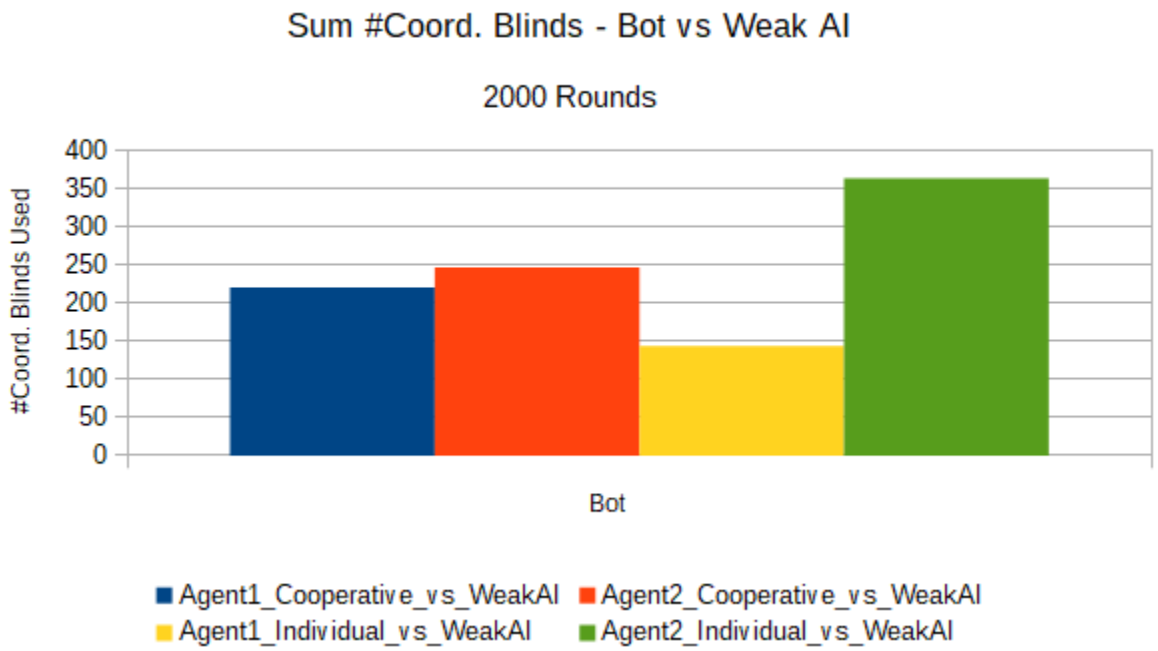

Figura 44 – Barplot de quantas vezes cada unidade usou *Blind* em um inimigo de modo a prolongar a duração durante a luta contra a IA fraca

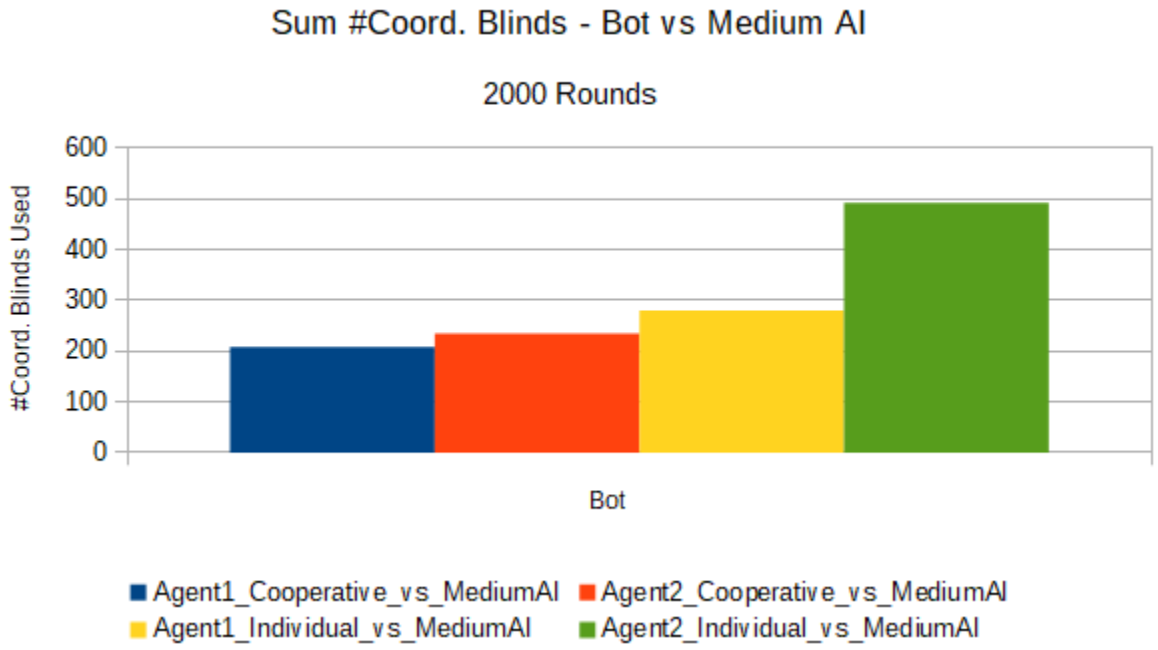

Figura 45 – Barplot de quantas vezes cada unidade usou *Blind* em um inimigo de modo a prolongar a duração durante a luta contra a IA média

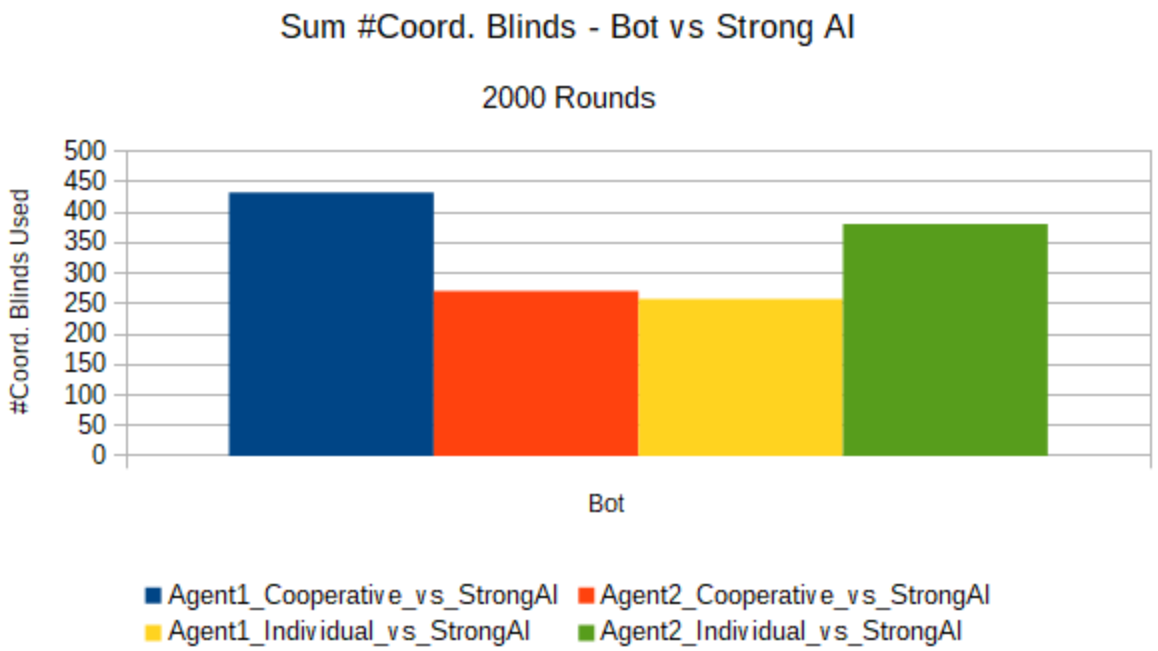

Figura 46 – Barplot de quantas vezes cada unidade usou *Blind* em um inimigo de modo a prolongar a duração durante a luta contra a IA forte

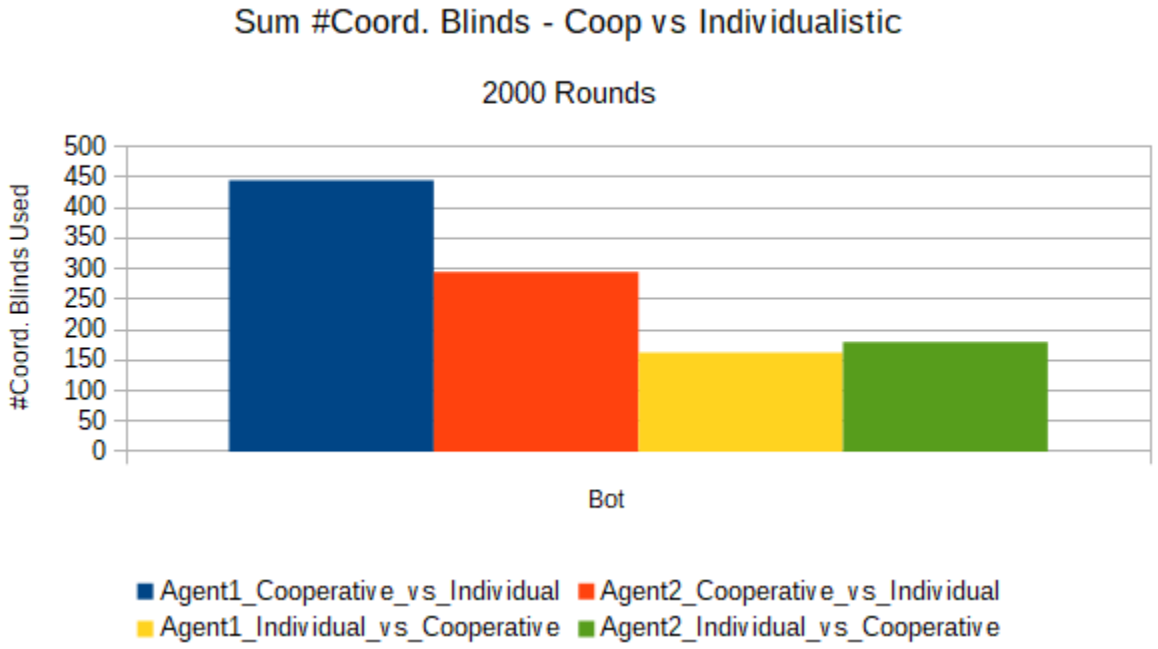

Figura 47 – Barplot de quantas vezes cada unidade usou *Blind* em um inimigo de modo a prolongar a duração na luta de Individualistas versus Cooperativos