UNIVERSIDADE FEDERAL DE ALAGOAS - UFAL
INSTITUTO DE COMPUTAÇÃO - IC
CIÊNCIA DA COMPUTAÇÃO

LUCAS A. LISBOA

LEVANTAMENTO E AVALIAÇÃO DE FUNÇÕES SCORE PARA ALGORITMO MINIMAX EM BATALHAS POKÉMON

Maceió - AL
2022

LUCAS A. LISBOA

ESTUDO E AVALIAÇÃO DE FUNÇÃO SCORE PARA ALGORITMO MINIMAX EM BATALHAS POKÉMON

Trabalho de Conclusão de Curso submetido ao corpo docente do Instituto de Computação da Universidade Federal de Alagoas. Orientado pela Profa. Dra. Roberta Vilhena Vieira Lopes

Maceió - AL
2022

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Taciana Sousa dos Santos – CRB-4 – 2062

L769i Lisboa, Lucas A..
Levantamento e avaliação de funções score para algoritmo minimax em batalhas de pokémon / Lucas A. Lisboa. – 2022.
38 f. : il. color.

Orientadora: Roberta Vilhena Vieira Lopes.
Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2022.

Bibliografia: f. 29-32.
Apêndices: f. 33-36.
Anexos: f. 37-38.

1. Games. 2. Pokémon (Jogo). 3. Inteligência artificial. 4. Minimax. I. Título.

CDU: 004.8: 794

Folha de Aprovação

AUTOR: LUCAS A. LISBOA

ESTUDO E AVALIAÇÃO DE FUNÇÃO SCORE PARA ALGORITMO MINIMAX EM BATALHAS POKÉMON

Trabalho de Conclusão de Curso submetido ao corpo docente do Instituto de Computação da Universidade Federal de Alagoas e aprovado em 26 de maio de 2022

Orientadora:

Roberta V. V. Lopes

Profa. Dra. Roberta Vilhena Vieira Lopes (IC - UFAL)

Banca Examinadora:

Prof. Dr. Evandro de Barros Costa (IC - UFAL)

Prof. Dr. George Darmiton da Cunha Cavalcanti (CIn - UFPE)

Agradecimentos

O processo da Graduação pode ser, por vezes, bastante intenso e tormentoso. Infelizmente, cursos de exatas possuem alta taxa de evasão, e não é diferente com Ciência da Computação. Presenciei diversos colegas e amigos abandonarem o processo da graduação pelas mais diversas razões. Dessa forma, isso nos faz refletir sobre nosso processo educacional, nossa valorização da ciência e tecnologia e do ritmo de produção em massa que vivemos. Assim, não posso deixar de ser grato por todos que contribuíram de alguma forma nessa trajetória, pois se cheguei aqui foi *porque me apoiei em ombros de gigantes*.

Em primeiro lugar, agradeço a meus pais, Maria da Piedade Albuquerque Lima e Samuel Lisboa Santos Neto, por todo apoio, suporte e incentivo ao longo dos anos. Juntamente deles, agradeço também à minha avó, Maria de Lourdes Albuquerque Lima, e à mulher que me criou, Zenilda Felix da Silva (Patrícia) pelo amor e carinho recebidos.

Agradeço à minha companheira, Edvania Fernandes, que está junto comigo há sete anos, acompanhando meu trajeto desde o início do Ensino Médio até aqui. Com ela, dividi meus anseios, frustrações e esperanças com o futuro.

Agradeço à minha orientadora, Roberta Lopes, a qual acreditou no meu potencial desde o início do curso, propondo-se a me ensinar e auxiliar nos mais diversos dias e horários. Em diversos momentos, ela se prontificou a resolver problemas de ordem institucional e educacional que eu precisava, não medindo esforços para ver o sucesso em seu alunado, além de todo suporte emocional prestado.

Agradeço aos professores que também aceitaram me apoiar nos diversos projetos e situações que apresentei, em especial ao professor Rafael de Amorim Silva, Thiago Cordeiro, Bruno Pimentel, Fábio Coutinho, Fábio Paraguaçu, André Aquino e Marcus de Melo Braga. Agradeço aos técnicos Marcelo de Gusmão e Ana Ferreira pelo apoio e agilidade nas necessidades burocráticas.

Agradeço aos colegas que percorreram comigo a graduação e aceitaram contribuir com alguns dos meus projetos, dentre eles José Rubens da Silva Brito, João Victor Ribeiro Ferro, Rodrigo Santos, Maria Fernanda Herculano e Jadson Costa.

Agradeço aos professores que aceitaram compor a banca examinadora, Evandro Costa e George Darmiton.

Agradeço à ajuda indireta de Jeff Atwood e Joel Spolsky, criadores do Stack Overflow, e Alexandra Elbakyan, criadora do Sci-hub.

Por fim, agradeço a você leitor, que se interessou em ler esta monografia.

RESUMO

O mercado de games tem ganhado cada vez mais destaque no mundo, sendo Pokémon uma de suas principais franquias. Nesse sentido, a concepção de estratégias para tomada de decisão nos agentes de Inteligência Artificial é uma preocupação constante dos desenvolvedores. Assim, este trabalho visa avaliar funções *score* para o algoritmo *Minimax* aplicado a batalhas pokémon, com o intuito de verificar quais abordagens obtêm melhores resultados. Com o uso de ferramentas, como *Pokemon Showdown* e *Poke-env*, diversas simulações foram realizadas para testar o desempenho das propostas apresentadas na literatura. Ao final dos experimentos, apesar do baixo número de propostas, foi possível constatar que funções de evolução as quais tinham como base a diferença percentual dos Pontos de Vida entre jogadores obtiveram melhores resultados nos experimentos, bem como as que continham um sistema de pontuação para status e efeitos.

Palavras–chave: Games; Pokémon; Minimax; Teoria dos Jogos; Inteligência Artificial; Tomada de Decisão.

ABSTRACT

The gaming market has gained more and more prominence in the world, with Pokémon being one of its main franchises. This way, the design of strategies for decision making in Artificial Intelligence agents is a constant concern of developers. So, this work aims to evaluate score functions for the *Minimax* algorithm applied to pokémon battles, in order to verify which approaches obtain better results. Using tools such as Pokemon Showdown and Poke-env, several simulations were performed to test the performance of the proposals presented in the literature. At the end of the experiments, despite the low number of proposals, it was possible to verify that evolution functions which were based on the percentage difference of Life Points between players obtained better results in the experiments, as well as those that contained a scoring system for status and effects.

Key-words: Games; Pokemon; Minimax; Game Theory; Artificial Intelligence; Decision Making.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

IA - Inteligência Artificial

RPG - Role-playing Game

HP - Health Points

ATK - Attack

DEF - Deffense

SPA - Special Attack

SPD - Special Deffense

SPE - Speed

PP - Power Points

SUMÁRIO

1 - Introdução

1.1 - Motivação

O mercado de games digitais tem crescido bastante nos últimos anos, tendo em 2021 superado a indústria da música e do cinema juntos (WAKKA, 2021). Dentro do cenário nacional, um dos indícios desse fato é que, entre final de outubro de 2019 e setembro de 2020, o número de transações financeiras nas principais plataformas e consoles aumentou em cerca de 140% no país (LARGHI, 2021).

Dentre os jogos mais significativos, está a franquia Pokémon, a qual, em 2021, tornou-se a maior franquia do mundo, superando as de outros segmentos, como Vingadores e *Star Wars* (AGRELA, 2021). Dada sua relevância, é interessante observar os aspectos que fizeram com que a franquia fosse tão apreciada, sendo um dos conceitos centrais dentro da proposta do game o sistema de batalhas.

Dessa forma, o estudo e análise das batalhas pokémon pode ser uma importante ferramenta para pensar no desenvolvimento de jogos, além de possibilitar colocar em prática algoritmos de tomada de decisão num novo contexto de simulação. Nesse sentido, competições pokémons entre agentes criados mediante o uso de Inteligência Artificial (IA) têm crescido nos últimos anos (LEE; TOGELIUS, 2021), voltadas, especialmente, para a ferramenta *Pokemon Showdown* (LUO, 2021), a qual é um simulador open-source de batalhas pokémon, no qual o usuário pode utilizar servidor oficial ou criar seu próprio servidor local.

Um dos algoritmos mais conhecidos e utilizados na literatura envolvendo tomada de decisão em jogos é o *Minimax*. Dessa forma, este trabalho visa identificar uma função *score* vencedora para tal algoritmo, isto é, que apresente resultado satisfatório no contexto de batalhas pokémon e se mostre mais eficiente (do ponto de vista de vitórias) que as demais analisadas.

1.2 - Objetivos

O objetivo geral deste trabalho consiste em identificar funções *score* que apresentem bons resultados para batalhas pokémon com algoritmo *Minimax* e implementar um agente com esta função. Enquanto os objetivos específicos podem ser descritos como:

- Utilizar a ferramenta Pokemon Showdown para realização das simulações de batalhas;
- Implementar algoritmos para tomada de decisão na linguagem *Python*, mediante uso da API *Poke-env*;
- Montar agentes com funções *score* diferentes para o algoritmo *Minimax*;
- Avaliar resultados.

Assim, algumas perguntas nortearam os estudos e experimentos aqui realizados, dentre elas:

- Questão 1: Quais propostas de função *score* para algoritmo *Minimax* para batalhas pokémon já foram apresentadas?
- Questão 2: Quais dessas propostas apresentam melhor resultado?

1.3 - Justificativa

A tomada de decisão para IA é um dos pontos de grande relevância e interesse por pesquisadores e desenvolvedores. A análise de estratégias e, consequentemente, funções *score* pode contribuir para o avanço das soluções desse segmento, em especial nos games. Assim, a escolha da franquia pokémon se justifica também pela tamanha influência que possui em outros games, além de seu sucesso e projeção internacional dela contribuir para a existência de propostas de estudo sobre ela na literatura.

1.4 - Estrutura do Trabalho

- **Capítulo 2 - Fundamentação Teórica:** Apresenta e explica dos conceitos utilizados neste trabalho, bem como relata os trabalhos relacionados que serviram de base para este;
- **Capítulo 3 - Materiais e Métodos:** Descreve a metodologia e tecnologia utilizadas nos análises e experimentos realizados;
- **Capítulo 4 - Resultados e Discussão:** Apresenta os resultados dos experimentos realizados e compara as informações presentes na literatura;
- **Capítulo 5 - Conclusão:** Apresenta as conclusões obtidas neste trabalho, bem como as considerações finais e trabalhos futuros.

2 - Fundamentação Teórica

2.1 - Pokémon

Pokémon é uma franquia de jogos desenvolvida pela *Game Freak* e publicada pela *Nintendo*. Criada por Satoshi Tajiri e Ken Sugimori, a franquia surgiu em 1996, com o lançamento de Pokémon Red/Green para *Game Boy*. O sucesso foi tamanho que, no ano seguinte (1997), a obra foi adaptada para animação. Ao todo foram 122 jogos lançados, totalizando 380 milhões de unidades vendidas (THE POKEMON COMPANY, 2022).

A franquia Pokémon é tão significativa para o Japão que, em 11/03/2022, o ministro de Estado para Assuntos Exteriores do Japão, Kiyoshi Odawara, presenteou o presidente chileno, Gabriel Boric, com um boneco do pokémon Squirtle (PACHECO, 2022). Nesse sentido, outro evento marcante foi, em 2014, a inserção do pokémon Pikachu no uniforme oficial da seleção japonesa de futebol (VICTORINO, 2014). Tais registros podem ser verificados nos Anexos A e B.

Os jogos, em sua linha principal, compõem o gênero de *Role-playing Game* (*RPG*), no qual o jogador monta uma equipe de pokémons e realiza batalhas com ela. Tais batalhas seguem o modelo de turnos, isto é, a ação do jogador ocorre e, logo depois, a do adversário e vice-versa. Apesar da tomada de decisão ocorrer simultaneamente, sempre um pokémon realiza o movimento primeiro, nunca ao mesmo tempo. Dessa forma, um turno da batalha é composto de três momentos:

1. Os jogadores tomam a decisão de qual ação realizar (simultaneamente);
2. Um dos jogadores executa a ação decidida;
3. O outro jogador executa a ação decidida.

Nos jogos, cada jogador possui 4 tipos de movimento:

- **Ataque**: O jogador ordena que seu pokémon realize um dos 4 ataques disponíveis;
- **Troca**: O jogador troca o seu pokémon atual por algum outro que ainda esteja habilitado para o combate;
- **Item**: O jogador utiliza algum item de sua mochila para gerar algum efeito (muito utilizado para cura, com uso de poções);
- **Fugir**: O jogador abandona a batalha;

Para efeito de estudo de estratégias e implementação do algoritmo, os movimentos do tipo Item e Fugir não foram habilitados nas simulações. O motivo de limitar o movimento de Fugir é bastante intuitivo: ao fugir, o jogador adversário automaticamente vence a batalha. Já

a limitação de Item consiste em dois pontos: analisar o desempenho das equipes sem intervenção de itens externos, apenas os ataques disponíveis de cada pokémon; implementar o algoritmo de busca em árvore[1] seria muito custoso se os itens da mochila fossem levados em consideração, pois não há limite previsto para o número de itens que o jogador pode carregar.

Cada pokémon é definido por, pelo menos, um tipo[2], o qual determina se ele possuirá vantagem, desvantagem, imunidade ou neutralidade em relação a outro pokémon. A relação de vantagem e desvantagem é inversa, isto é, se um pokémon X possui vantagem em relação a um pokémon Y, o pokémon Y possui desvantagem em relação ao pokémon X. A neutralidade é uma relação comutativa, isto é, se pokémon X é neutro em relação ao pokémon Y, pokémon Y também será neutro em relação ao pokémon X. Já a neutralidade é a relação que mais destoa das demais, pois se um pokémon X possui neutralidade em relação a pokémon Y, não há nenhuma condição consequente de pokémon Y para pokémon X.

Tabela 1 - Tabela de Relações entre Tipos

Attacker \ Defender	Normal	Fire	Water	Grass	Electric	Ice	Fighting	Poison	Ground	Flying	Psychic	Bug	Rock	Ghost	Dragon	Dark	Steel	Fairy
Normal													½	0			½	
Fire		½	½	2		2						2	½		½		2	
Water		2	½	½					2				2		½			
Grass		½	2	½				½	2	½		½	2		½		½	
Electric			2	½	½				0	2					½			
Ice		½	½	2		½			2	2					2		½	
Fighting	2					2		½		½	½	½	2	0		2	2	½
Poison				2				½	½				½	½			0	2
Ground		2		½	2			2		0		½	2				2	
Flying				2	½		2					2	½				½	
Psychic							2	2			½					0	½	
Bug		½		2			½	½		½	2			½		2	½	½
Rock		2				2	½		½	2		2					½	
Ghost	0										2			2		½		
Dragon															2		½	0
Dark							½				2			2		½		½
Steel		½	½		½	2							2				½	2
Fairy		½					2	½							2	2	½	

Fonte: Evil (2012)

[1] Estrutura de Dados que dá base ao algoritmo Minimax
[2] Em alguns casos, o pokémon possui mais de um tipo, como, por exemplo, Dragão e Fogo (Charizard)

A Tabela 1 ilustra as relações entre tipos pokémon, sendo as células da tabela preenchidas com o valor ½ e com a cor vermelha a relação de desvantagem; as células preenchidas com o valor 0 e com a cor preta a relação de imunidade; as células preenchidas com o valor 2 e com a cor verde a relação de vantagem; e as células vazias a relação de neutralidade. Tais valores representam o fator de multiplicação do dano durante a batalha (MONTES, 2013). Dessa forma, por exemplo, se um pokémon X possui vantagem em relação ao seu oponente, seu dano será multiplicado por 2.

Além do tipo do pokémon, há alguns outras características que dão base para o desempenho de um pokémon (MONTES,2013):

- **Pontos de Vida Base (HP):** Pode ser definido como a "vida" do pokémon, isto é, a quantidade de dano que ele pode receber antes de ficar inativo;
- **Ataque Base (ATK):** Determina a quantidade de dano infligido em um ataque físico;
- **Defesa Base (DEF):** Determina a resistência a um ataque físico;
- **Ataque Especial Base (SpAtk):** Determina a quantidade de dano infligido em um ataque especial;
- **Defesa Especial Base (SpDef):** Determina a resistência a um ataque especial;
- **Velocidade Base (SPE):** Determina a velocidade de um pokémon. Pokémons mais rápidos realizam seu movimento antes do oponente.
- **Level**: Nível de um pokémon

Cada movimento também possui uma série de características que influenciam no desempenho (MONTES, 2013):

- **Poder**: Dano base do ataque;
- **PP (Pontos de Poder):** Define a quantidade de vezes que aquele movimento pode ser usado na batalha;
- **Acurácia**: Probabilidade de acertar o ataque;

Alguns movimentos especiais podem infringir certos *status* ao oponente (MONTES, 2013), dentre eles:

- **Queimado**: Infligido por ataques do tipo fogo e causa perda de ⅛ do HP Base ao final de cada turno;
- **Paralisado:** Faz com o pokémon tenha 25% de chance de não conseguir atacar em seu turno, além de reduzir em 25% a velocidade dele;

- **Envenenado:** Infligido por ataques do tipo Veneno e causa perda de ⅛ do HP Base ao final de cada turno;
- **Dormindo**: Faz com que o pokémon não consiga atacar entre 1 a 3 turnos;
- **Congelado**: Faz com que o pokémon não consiga atacar, mas ele possui 20% de chance de se curar a cada turno;

De vista dessas informações, é possível analisar a fórmula de Dano Base. Ela é definida por (MONTES, 2013):

$$DanoBase = (((2 * Level)/5) + 2) * Poder * SpAtk) * (SpDef * 50 + 2)$$

Esse valor pode ser multiplicado por um fator a depender da relação vantagem-desvantagem entre os pokémons que estão se enfrentando.

2.2 - *Minimax*

O Teorema *Minimax* é um algoritmo proposto por John von Neumann (SARTINI et al, 2004) , sendo um dos mais importantes para a área de Inteligência Artificial e Teoria dos Jogos. Ele consiste em buscar minimizar o ganho do jogador adversário enquanto busca maximizar seu próprio resultado, sendo bastante utilizado em jogos de soma zero, isto é, jogos em que vetor de resultado possui valores sempre opostos (RUSSEL; NORVING, 2013), indicando que, quando um jogador ganha, o outro perde. Para isso, é preciso mapear as possíveis combinações de ações em uma tabela, denominada *payoff*, conforme pode ser visualizada na Tabela 2.

Tabela 2 - Tabela Payoff

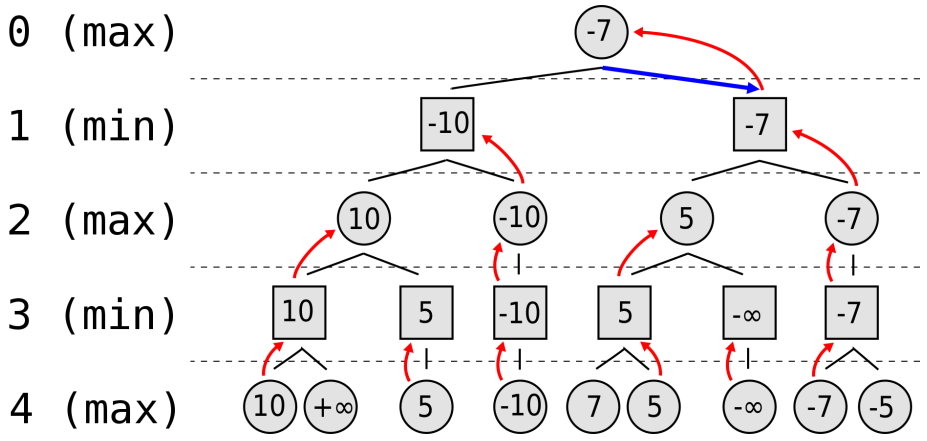
		jogador coluna			
		1	2	$\cdots$	n
jogador linha	1	(a_{11}, b_{11})	(a_{12}, b_{12})	$\cdots$	(a_{1n}, b_{1n})
	2	(a_{21}, b_{21})	(a_{22}, b_{22})	$\cdots$	(a_{2n}, b_{2n})
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	m	(a_{m1}, b_{m1})	(a_{m2}, b_{m2})	$\cdots$	(a_{mn}, b_{mn})

Autor: Sartini et al (2004)

Em cada célula da tabela acima, tem-se (a_{ij}, b_{ij}) representado o resultado da combinação das ações i e j, sendo a_{ij} o peso para o jogador linha e b_{ij} o peso para o jogador coluna. Essa forma de representação, matricial, é denominada forma normal, bastante utilizada em jogos simultâneos.

Além da forma normal, há também a forma extensa, na qual o jogo é representado como uma árvore. Nesta, o algoritmo *Minimax* funciona como uma busca em profundidade no grafo, em que utiliza uma função de evolução ou função *score* para determinar a pontuação do jogador do turno em questão (RUSSEL; NORVING, 2013), alternando entre minimizar e maximizar. Esta forma está representada na Figura 1, bem como no Anexo C, em que há o exemplo da execução do algoritmo no Jogo da Velha.

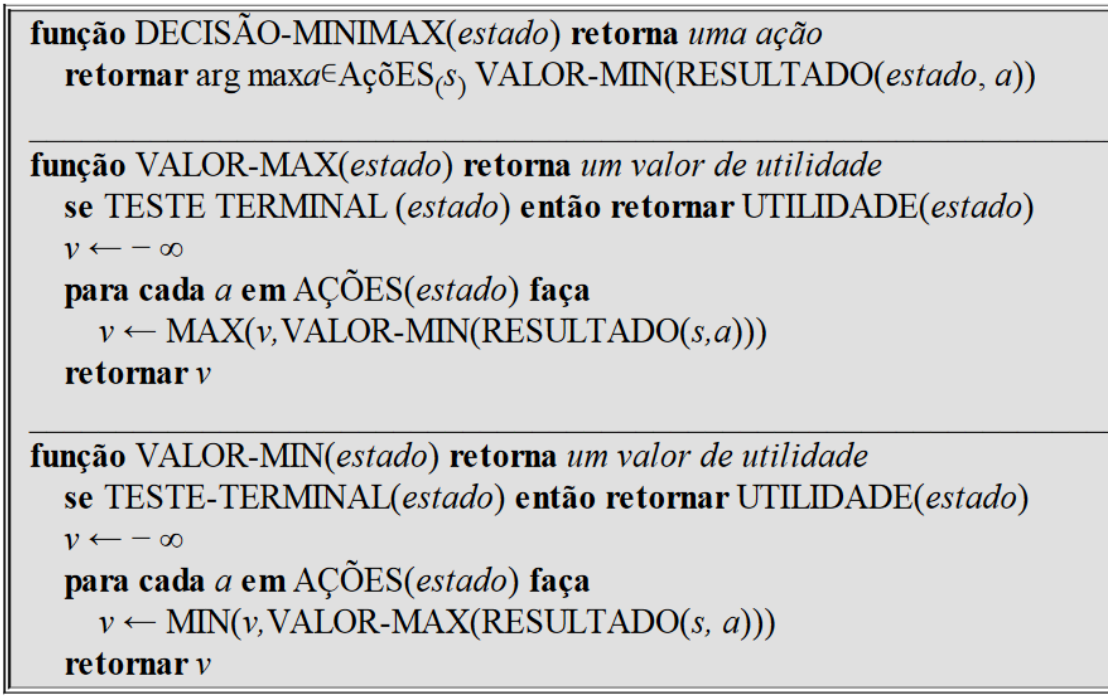
Figura 1 - Árvore *Minimax*



Autor: Nogueira (2006)

O funcionamento desse algoritmo pode ser verificado no pseudo-código elaborado por Russel e Norving (2013), o qual consta na Figura 2. Ressalta-se, também, que, apesar de existirem versões alternativas do algoritmo, tais como *NegaMax* e Alpha-*Beta* (ELNAGGAR et al, 2014), este trabalho focou apenas no algoritmo base original.

Figura 2 - Pseudocódigo *Minimax*



```
função DECISÃO-MINIMAX(estado) retorna uma ação
  retornar arg max a∈AçõES(s) VALOR-MIN(RESULTADO(estado, a))

função VALOR-MAX(estado) retorna um valor de utilidade
  se TESTE TERMINAL (estado) então retornar UTILIDADE(estado)
  v ← −∞
  para cada a em AÇÕES(estado) faça
    v ← MAX(v, VALOR-MIN(RESULTADO(s,a)))
  retornar v

função VALOR-MIN(estado) retorna um valor de utilidade
  se TESTE-TERMINAL(estado) então retornar UTILIDADE(estado)
  v ← −∞
  para cada a em AÇÕES(estado) faça
    v ← MIN(v, VALOR-MAX(RESULTADO(s, a)))
  retornar v
```

Autor: Russel e Norving (2013)

2.3 - Trabalhos Correlatos

O processo de revisão bibliográfica do que foi produzido acerca do tema na literatura iniciou-se com a definição das bases de buscas e quais *strings* seriam usadas. Para a busca de artigos e publicações, foram utilizadas: *IEEE Xplore*, *Scopus*, *ACM*, Periódicos CAPES e *Google Scholar*. Já as strings escolhidas foram: *pokemon AND ((game theory) OR (Nash equilibrium))*; *pokémon AND ((teoria dos jogos) OR (equilíbrio de Nash))*; *pokemon AND minimax.* Ao final das buscas, foram constatados:

- *IEEE Xplore*: 7 resultados;
- *Scopus*: 59 resultados;
- *ACM*: 654 resultados;
- Periódicos CAPES: 988 resultados;
- *Google Scholar*: 9.334 resultados;

Totalizando 11.042 resultados, sendo 96 resultados duplicados. A imensa maioria dos resultados tratava sobre games no geral, citando pokémon de maneira vaga, ou propunha outra abordagem sobre o tema. Apenas 14 publicações trataram de métricas para avaliação de batalhas pokémon.

A primeira publicação foi de Montes (2013) e trata-se de um relatório técnico de seu projeto para desenvolvimento de uma IA para batalhas pokémon 1 vs 1. Ele utilizou dois algoritmos para sua abordagem: *Minimax* e *AlphaBeta*. Para o trabalho aqui apresentado, foi aproveitado a função de evolução de Montes (2013), sendo ela bastante simples:

$$payoff = 100 * (HP_{Player}/MaxHP_{Player}) - 100 * (HP_{Opp}/MaxHP_{Opp})$$

A qual consiste na diferença entre o percentual da vida do jogador e do percentual da vida do oponente, variando entre -100 e 100. Algumas soluções foram adotadas para otimizar a busca na árvore, como um cálculo para poder ramos que não atingissem o dano mínimo, no entanto não entraram na implementação final por estarem fora do escopo de análise deste trabalho, apenas a função *payoff* pura.

Panumate, Xiong e Iida (2015) usam a plataforma do *Pokemon Showdown* (LUO, 2021) para desenvolver uma métrica que avaliava o valor de refinamento do *game*, isto é, o quanto ele é sofisticado e atrativo. Como o foco era na percepção do jogador em relação ao game, tal métrica não foi utilizada neste estudo.

Xu (2015) publica sua tese em que põe melhora na IA para jogos de desgaste, com foco em Pokémon. Apesar de citar brevemente o algoritmo *Minimax*, seu foco foi em heurísticas para movimentos para adormecer e curar. Posteriormente, Xu e Verbrugge (2016) publicam novamente tais heurísticas. Por não se tratarem do algoritmo *Minimax*, tais heurísticas não entraram no escopo deste estudo.

Lee e Togelius (2017) apresentaram o trabalho mais relevante para o tema desta pesquisa, tendo montado uma competição de IAs em um servidor do *Pokemon Showdown*. A metodologia apresentada no artigo serviu como base para este trabalho[3]. Dentre os agentes criados, está um que utiliza como estratégia o *Minimax*. Sua função de *payoff* consiste em:

$$payoff = (HP_{Player}/MaxHP_{Player}) - 3 * (HP_{Opp}/MaxHP_{Opp}) - 0,3 * profundidade$$

A variável *profundidade* se refere à altura da árvore. Esta função *score* entrou na implementação final.

Em seu trabalho de conclusão de curso, Sanchez (2018) implementou um agente de Aprendizagem por Reforço no ambiente de *Pokemon Showdown*. O algoritmo *Minimax* só foi

[3] Mais detalhes no Capítulo 3

utilizado como *baseline* para evolução do agente implementado. Outra publicação que enfoca uso de Aprendizagem por Reforço utilizando *Pokemon Showdown*, foi a de Simões et al (2020). Dessa forma, esses trabalhos ficaram de fora do escopo.

Ihara et al (2018) implementou um simulador de batalhas pokémon com uso do algoritmo da Árvore de Monte Carlo. Tal abordagem também foi feita por Norström (2019) em sua tese de mestrado, enquanto Verdear e Visser (2020) desenvolveram um sistema de conhecimento baseado em ontologia para batalhas pokémon. Novamente, tais trabalhos ficaram de fora do escopo.

Alguns trabalhos feitos durante certas disciplinas também foram contabilizados, tais como os de Chen e Lin (2018); Kalose, Kaya e Kim (2018); e Rill-García (2018). Todos acerca de Aprendizagem por Reforço.

Por fim, foi feita uma sondagem em repositórios do Github que utilizam o algoritmo *Minimax* em servidores *Showdown*, sendo três selecionados. Compton (2021) estipula sua função *score* como:

```
def score(self, node):
score = 0
for pokemon in node.opponent_HP.keys():
  if pokemon.current_hp is not None:
      if node.opponent_HP[pokemon] <= 0 and pokemon.current_hp > 0:
            score += 300
      else:
            damage = pokemon.current_hp - node.opponent_HP[pokemon]
            score += 3 * damage
for pokemon in node.current_HP.keys():
      if node.current_HP[pokemon] <= 0 and pokemon.current_hp > 0:
      score -= 100
      else:
      damage = (pokemon.current_hp / pokemon.max_hp) -
(node.current_HP[pokemon] / pokemon.max_hp)
      score -= damage
node.score = score
return score
```

Basicamente, ele estipula alguns condicionais para pontuação: caso o pokémon do oponente fique inativo, são somados 300 pontos; caso não, são somados pontos equivalentes ao dano causado. Já para diminuição do *score*, tem-se: caso o pokémon do jogador fique inativo, são subtraídos 100 pontos; caso não, são subtraídos pontos equivalentes ao dano

sofrido. Esse procedimento é aplicado a todos os pokémons da equipe do jogador e aos da equipe adversária.

Por fim, Ho e Ramesh (2017) e Pmariglia (2022) seguem a mesma abordagem, na qual eles elaboram um sistema de pontuação para cada *Status*, *Effects* e *Boots* que ocorre durante a batalha. Tais pontuações podem ser vistas na Tabela 3, a qual foram retirados elementos que não entraram na implementação, como pontuação por item. Depois da contabilização desses pontos, é somado ao score a diferença entre percentual do HP dos pokémons, seguindo o padrão de Montes (2013)

Tabela 3 - Sistema de Pontuação de Ho e Ramesh (2017) e Pmariglia (2022)

STATUS / EFFECTS / BOOSTS	*Minimax_Ho_Ramesh*	*MiniMax_Pmariglia*
Reflect	64	20
Spikes	-150	-7
Stealth Rock	-200	-10
Sticky Web	-40	-25
Toxic Spikes	-100	-7
Light Screen	64	20
Tailwind	32	7
Substitute	150	40
Confusion	-32	-20
Leech Seed	-80	-30
ATK	50	15
DEF	25	15
SPA	50	15
SPD	50	15
SPE	0	25
ACCURACY	10	3
EVASION	10	3
Poison	-100	-10
Toxic	-100	-30
Sleep	-120	-25
Burn	-100	-25
Frozen	-100	-40
Paralyzed	-100	-25
Aurora Veil	0	40
Safeguard	0	5

Fonte: Autor

Em suma, foram selecionadas cinco funções *score* para testagem e avaliação: Montes (2013); Lee e Togelius (2017); Ho e Ramesh (2017); Compton (2021); e Pmariglia (2022).

Como o foco do trabalho, foi na análise das função evolução do *payoff*, a implementação reutilizou a estrutura de busca em árvore, alterando apenas a função *score* de cada função. A base da implementação foi inspirada na demonstração de Compton (2021), o qual utilizou a estratégia de criar nós temporários para não afetar o estado real das partidas, enquanto realizava a busca. Detalhes dessa implementação podem ser vistos no Apêndice A, bem como a autorização de Compton (2021) para ajuste em seu código no Apêndice B.

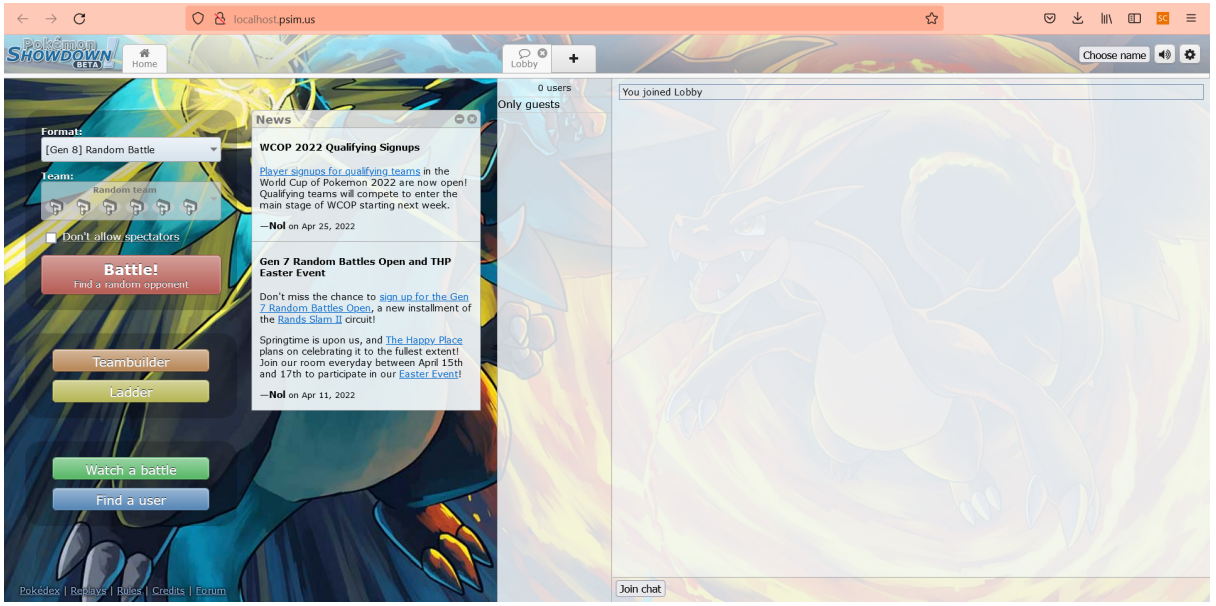
3 - Materiais e Métodos

3.1 - *Pokemon Showdown* e *Poke-env*

Pokemon Showdown é um simulador de batalhas pokémon desenvolvido por Luo (2021). Com ele, é possível realizar batalhas *online*, criadas de maneira randômica ou montada pelo próprio usuário. O sistema está atualizado até a 8ª geração pokémon. Neste estudo, foi utilizado a versão 0.11.7.

Durante os experimentos, foi criado um servidor local. Para isso, é necessário a instalação do *Node.js* a partir da versão 10, sendo a versão utilizada para testes a v16.14.2 para *Windows*. Para isso, foi feita a clonagem do repositório do *Github*, configuração e criação do servidor a partir dos comandos presentes no Anexo D. Após isso, é possível acessar o servidor local criado, como é possível verificar na Figura 3.

Figura 3 - Lobby Pokemon Showdown



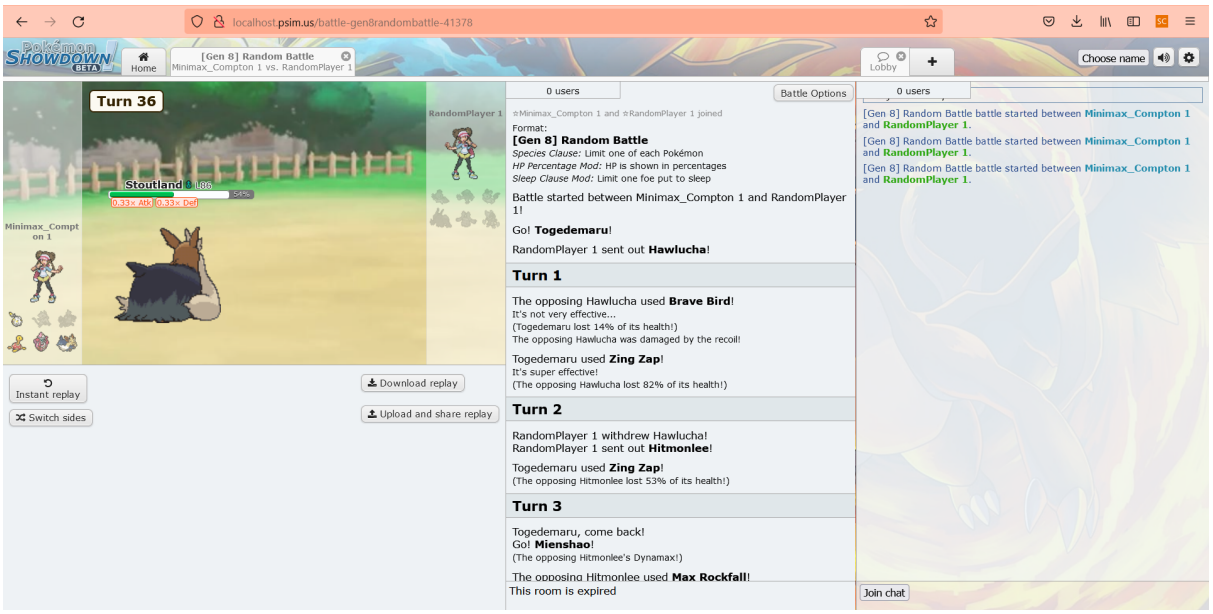
Fonte: Autor

Atrelado a isso, foi utilizada a *The Pokemon Showdown Python Environment*, mais conhecida como como poke-env, uma interface (API) em Python para criação de agentes pokémon para batalhas no *Pokemon Showdown*, tendo sido desenvolvida por Sahovic (2021). A Poke-env permite a construção de IAs baseadas em regras ou em Aprendizagem por Reforço. Neste estudo, foi utilizada a versão 0.4.21.

Para utilizar a *Poke-env*, é necessário a instalação do *Python* 3.6 ou superior, sendo a versão utilizada para testes a 3.8. Tendo todos os requisitos, o comando presente no Anexo E

foi executado para realizar a instalação. É possível assistir cada batalha das simulações pelo servidor local criado, como é possível visualizar na Figura 4.

Figura 4 - Exemplo de Batalha no Servidor Local



Fonte: Autor

3.2 - Ambiente de Teste

Os testes foram realizados na seguinte máquina: *Ideapad* S145, Processador *Intel*(R) *Core*(TM) i7-8565U *CPU* @ 1.80GHz 1.99 GHz, 8 GB de memória RAM, sistema operacional *Windows* 11 64 *bits*.

3.3 - Agentes, Batalhas e Validação

As equipes foram geradas aleatoriamente por pokémons da 8ª geração, no modelo 6vs6, isto é, batalhas entre equipes completas. Foram criados cinco agentes a serem avaliados, um para cada função *score* / *evoluation* do algoritmo *Minimax*[4].

Dois agentes foram criados como grupo controle para verificar se o desempenho dos agentes avaliados poderia ser considerado satisfatório. O primeiro foi o *RandomPlayer*, o qual tomava a decisão do próximo movimento de maneira aleatória; e o segundo foi o *MaxDamagePlayer*, o qual escolhia sempre o ataque com maior dano. Ambos foram baseados na documentação da *Poke-env* (SAHOVIC, 2021), tendo sido escolhidos pela simplicidade de implementação e por serem estratégias pouco desenvolvidas. Dessa forma, para um agente

[4] Já descritas no Capítulo 2.3

avaliado ter seu desempenho tido como satisfatório é preciso ter desempenho superior a estas duas estratégias num confronto direto.

Passando pela etapa de validação pelo grupo controle, o agente avaliado enfrenta outro agente validado anteriormente. Após isso, há inversão na ordem deles e é feito novamente o confronto. Repete-se esse processo, até que todos os agentes avaliados realizem o confronto entre si.

Em cada rodada de teste, 500 partidas são realizadas. Esse valor foi escolhido levando em conta a amostragem necessária para representar a população de equipes pokémons possíveis de serem montadas. Nesse sentido, fez-se a combinação de 905 pokémons para as seis posições da equipe, considerando que a ordem dos pokémons importa e que é possível ter pokémons iguais dentro da mesma equipe. Assim, obteve-se o número de 549.403.567.610.640.640 possibilidades de equipe, sendo 500 o tamanho da amostra para conseguir nível de confiança de 95% (1,96 de z-escore) com margem de erro de 4,385%

Vale destacar que Agente X VS Agente Y representa uma rodada, enquanto Agente Y VS Agente X representa outra rodada. Após cada rodada, é feito o Teste de Hipótese Bilateral, com nível de confiança de 95%, para verificar se o resultado encontrado é significativo ou não (ASSIS; SOUSA; LINHARES, 2020). Para isso, foi assumido como Hipótese Nula que os algoritmos avaliados entre si não possuem vantagem ou desvantagem, tendo assim:

$$H_0: \ resultado\ (X\ vs\ Y)\ =\ 50\%\ de\ vitórias$$

Dessa forma, é assumido que o valor de $resultado\ (X\ vs\ Y)$ deve ser estatisticamente maior /menor que 250. Assim, formula-se a Hipótese Alternativa:

$$H_1: \ resultado\ (X\ vs\ Y)\ \neq\ 50\%\ de\ vitórias$$

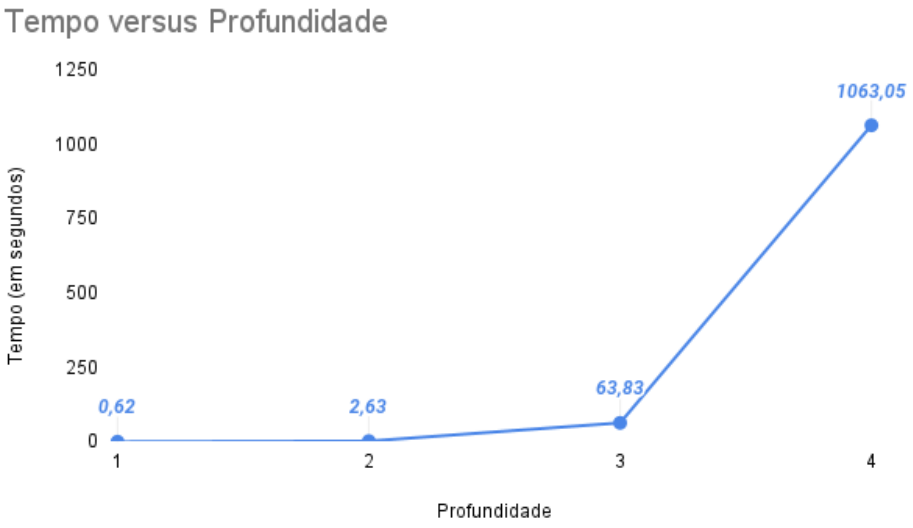
4 - Resultados e Discussão

Antes de adentrar nos resultados em si, é preciso discutir um fator limitante bastante importante: o custo computacional do algoritmo *Minimax*. No capítulo 2.1, foi citado que o uso de itens foi removido tendo como um dos motivos diminuir a busca na árvore de resultados, no entanto, mesmo com essa alteração, ainda existe um custo alto na implementação. Busca em Profundidade tem complexidade em $O(n^p)$, sendo n as opções de ações em um turno e p a profundidade da árvore, ou seja, tem-se um custo exponencial (RUSSEL; NORVING, 2013).

Calculando o custo para batalhas pokémon, temos que cada jogador possui 4 opções de ataque mais as opções de troca. Se todos os demais pokémons da equipe estiverem ativos, o jogador tem 5 opções de troca. Dessa forma, temos que $n = 9$. No entanto, como os dois jogadores jogam dentro do mesmo turno, é preciso elevar esse valor ao quadrado. Logo, $n = 81$. Dessa forma, o custo da busca em batalhas pokémon é $O(81^p)$, por turno.

Isso ficou demonstrado nas primeiras execuções, nas quais foi analisado o tempo que cada nível de profundidade levaria para cada batalha. Na Figura 5, tem-se o gráfico com os valores constatados. Percebe-se que a partir da transição da profundidade máxima 2 para 3 já há um aumento considerável no tempo que uma partida dura. A partir da profundidade 4, por vezes, o tempo entre uma tomada de decisão e outra era tão grande que ultrapassava o tempo máximo previsto no servidor, gerando timeout e interrompendo a execução.

Figura 5 - Tempo versus Profundidade



Fonte: Autor

Dessa forma, foram estipulados dois cenários para análise: *Profundidade Máxima = 1* e *Profundidade Máxima = 2*, pois a partir do nível 3 de profundidade seria bastante custoso computacionalmente. Além disso, Compton (2021) constatou que quanto maior a profundidade máxima, pior o resultado obtido, recomendando o uso de sua função com *Profundidade Máxima = 1*.

Tabela 4 - Resultados para *Profundidade Máxima = 1*

	Random	MaxDamage	Minimax_Lee_Togelius	Mimimax_Montes	Minimax_Compton	Minimax_Ho_Ramesh	MiniMax_Pmariglia
Minimax_Lee_Togelius	425/500 (85%)	439/500 (87,8%)	-	227/500 (45,4%)	252/500 (50,4%)	221/500 (44,2%)	241/500 (48,20%)
Mimimax_Montes	446/500 (89,2%)	426/500 (85,2%)	273/500 (54,6%)	-	263/500 (52,6%)	259/500 (51,8%)	230/500 (46,00%)
Minimax_Compton	410/500 (82%)	427/500 (85,4%)	249/500 (49,8%)	214/500 (42,8%)	-	228/500 (45,6%)	236/500 (47,20%)
Minimax_Ho_Ramesh	439/500 (87,8%)	429/500 (85,8%)	267/500 (53,4%)	265/500 (53%)	309/500 (61,8%)	-	274/500 (54,80%)
MiniMax_Pmariglia	439/500 (87,8%)	448/500 (89,6%)	287/500 (57,40%)	256/500 (51,20%)	271/500 (54,20%)	241/500 (48,20%)	-

LEGENDA	
VERDE	Significativamente Melhor
AZUL	Diferença Não Significativa
VERMELHO	Significativamente Pior

Fonte: Autor

Na Tabela 4, pode-se observar os resultados obtidos estipulando a Profundidade Máxima como 1. Percebe-se que todos passaram pelo teste de controle, obtendo taxa de sucesso de pelo menos 82%. As propostas de Lee e Togelius (2017) e Montes (2013) não apresentaram resultados significativamente melhores em relação aos demais algoritmos. Já Compton (2021) apresentou-se significativamente pior contra Montes (2013), enquanto Ho e Ramesh (2017) mostrou-se significativamente melhor que Compton (2013) e Pmariglia (2022) que Lee e Togelius (2013).

Tabela 5 - Resultado para *Profundidade Máxima = 2*

	Random	MaxDamage	Minimax-Lee_Togelius	Mimimax_Montes	Minimax_Compton	Minimax_Ho_Ramesh	MiniMax_Pmariglia
Minimax-Lee_Togelius	411/500 (82,2%)	426/500 (85,2%)	-	233/500 (46,6%)	265/500 (53%)	219/500 (43,8%)	240/500 (48,00%)
Mimimax_Montes	430 / 500 (86%)	432/500 (86,4%)	293 / 500 (58.6%)	-	304/500 (60,8%)	246/500 (49,2%)	247/500 (49,40%)
Minimax_Compton	392/500 (78,4%)	414/500 (82,8%)	250/500 (50%)	230/500 (46%)	-	200/500 (40%)	230/500 (46,00%)
Minimax_Ho_Ramesh	414/500 (82,8%)	438/500 (87,6%)	275/500 (55%)	254/500 (50,8%)	284/500 (56,8%)	-	245/500 (49,00%)
MiniMax_Pmariglia	418/500 (83,60%)	428/500 (85,60%)	287/500 (57,40%)	244/500 (48,80%)	267/500 (53,40%)	244/500 (49,20%)	-

LEGENDA	
VERDE	Significativamente Melhor
AZUL	Diferença Não Significativa
VERMELHO	Significativamente Pior

Fonte: Autor

Na Tabela 5, vê-se os resultados constatados para a Profundidade Máxima estipulada como 2. Novamente, todos passaram no teste de controle, caindo um pouco o resultado mínimo da taxa de sucesso, indo para 78,4%. Apesar de não ser significativa essa queda, já é um indício de confirmação das constatações de Compton acerca da perda de desempenho.

As propostas de Lee e Togelius (2017) e Compton (2021) obtiveram resultados significativamente piores em relação ao *Minimax* de Ho e Ramesh (2017). Já Montes (2013) foi significativamente melhor que Lee e Togelius (2017) e que Compton (2021), bem como Ho e Ramesh (2017) demonstraram superioridade em relação a Compton (2021) e Pmariglia (2022) em relação a Lee e Togelius (2017).

Considerando os dois cenários, pode-se considerar que a proposta que apresentou pior resultado foi a de Lee e Togelius (2017), tendo em vista os resultados contra Ho e Ramesh (2017), Montes (2013) e Pmariglia (2022); seguido por Compton (2013), pois foi inferior em combate direto contra Montes (2013) e Ho Hamesh (2017).

Por sua vez, Montes (2013), Ho e Ramesh (2021) e Pmariglia (2022) apresentaram melhores resultados, não tendo nenhum resultado significativamente pior e possuindo resultados significativos em relação a Lee e Togelius (2017) e Compton (2013). Todos têm em comum o uso da diferença percentual entre o HP do jogador e o HP do adversário, mostrando que, apesar de simples, tal função é extremamente funcional. Nesse sentido, o uso do sistema de pontuação também mostra-se eficaz.

Todas as implementações realizadas podem ser verificadas no Github de Lisboa (2022), conforme demonstrado no Apêndice C.

5 - Conclusão

5.1 - Considerações Finais

Apesar do estudo de algoritmos para tomada de decisão ser um tema bastante importante e consolidado na literatura, sua aplicação em games, em especial para pokémon, ainda carece de maiores pesquisas. Poucas soluções foram apresentadas até o momento, e menos propostas ainda focadas no algoritmo *Minimax*. Dessa forma, respondendo à Questão 1 proposta neste estudo, temos apenas cinco funções.

Esse fato pode ser explicado pelo fato das empresas que produzem os jogos não tornarem seus codigo-fontes abertos, mesmos os mais antigos, limitando a análise das estratégias adotadas. Isso faz com que a comunidade de jogadores tenha que criar soluções próprias. Assim, só nos últimos anos, foram surgindo soluções viáveis para simular tais ambientes. *Pokemon Showdown* (LUO, 2021), por exemplo, teve sua primeira versão lançada apenas ao final de 2011, 15 anos após o primeiro jogo de Pokémon. Por sua vez, a primeira versão da *API Poke-env* (SAHOVIC, 2021) foi lançada em 2019, oito anos depois de *Showdown*.

Isso explica o baixo número de publicações no setor e, ainda, o escasso número de artigos acadêmicos. Algumas das soluções encontradas não passaram pelo processo de revisão por pares, estando em repositórios do Github e trabalhos de disciplinas cursadas. Isso também indica a necessidade do avanço e amadurecimento acadêmico com relação à área de *games*, em especial no campo de simulação e estudo para tomada de decisão.

Partindo para a Questão 2 deste estudo, o baixo número de propostas impede uma resposta conclusiva, mas os experimentos já apontam indícios da efetividade da abordagem de Montes (2013) e a criação de um sistema de pontuação envolvendo efeitos e status. Isso evidencia que jogos de soma-zero envolvendo HP tem como base estratégias que visam a diferença percentual entre os HPs momento a momento, mas, também, indica que buscar formas de otimizar efeitos e status no oponente e em si mesmo pode ser um diferencial durante as batalhas.

Pokemon Showdown (LUO, 2021) mostrou-se uma ferramenta muito útil e prática para realizar os tipos de simulação presentes aqui, bem como a Poke-env que funcionou como um ótimo *middleware* entre aplicação e servidor local. Nesse sentido, o potencial de novos estudos utilizando tais recursos é muito grande, pois as constatações presentes aqui ajudam a pensar e planejar a criação de IAs para games, bem como validá-las posteriormente.

5.2 - Trabalhos Futuros

Como trabalhos futuros, destaca-se avançar para avaliação de métricas de outras estratégias para tomada de decisão. Como exemplo, tem-se o uso de algoritmos de Aprendizagem por Reforço, já constando algumas publicações neste segmento.

Além disso, outra possibilidade de trabalho futuro é a criação e avaliação de agentes com estratégias múltiplas para comparação com o desempenho de agentes com estratégias únicas. Esse tipo de trabalho seria interessante por avançar ainda mais na proposição de Lee e Togelius (2017), já que focaram seu estudo no desempenho das estratégias únicas.

Por fim, há, também, a possibilidade de realizar avaliações de funções *score* para *Minimax* em outros games.

Referências Bibliográficas

AGRELA, Lucas. Pokémon supera Star Wars e Marvel como maior franquia do mundo. **Exame**, [*S. l.*], 24 mar. 2021. Disponível em: https://exame.com/tecnologia/pokemon-supera-star-wars-e-marvel-como-maior-franquia-do-mundo/. Acesso em: 2 abr. 2022.

ASSIS, Janilson Pinheiro de; SOUSA, Roberto Pequeno de; LINHARES, Paulo César Ferreira. **TESTES DE HIPÓTESES ESTATÍSTICAS**. [*S. l.*]: Edufersa, 2020.

BEKCER, Lauro. Algoritmo Minimax - Introdução à Inteligência Artificial. **Orgânica Digital**, [*S. l.*], 5 set. 2019. Disponível em: https://www.organicadigital.com/blog/algoritmo-minimax-introducao-a-inteligencia-artificial/ . Acesso em: 7 maio 2022.

CHEN, Kevin; LIN, Elbert. Gotta Train 'Em All: Learning to Play Pokemon Showdown with Reinforcement Learning. 2018. Disponível em: https://web.stanford.edu/class/aa228/reports/2018/final151.pdf. Acesso em: 5 mai. 2022.

COMPTON, Caleb. **POKEMON SHOWDOWN AI BATTLEBOT ****. Github, 27 jun. 2021. Disponível em: https://github.com/RemptonGames/Pokemon-Showdown-Agent. Acesso em: 5 mai. 2022.

ELNAGGAR, Ahmed A.; GADALLAH, Mahmoud; AZIEM, Mostafa Abdel; EL-DEEB, Hesham. A Comparative Study of Game Tree Searching Methods. **(IJACSA) International Journal of Advanced Computer Science and Applications,**, [*s. l.*], v. 5, n. 5, p. 68-77, 2014. Disponível em: https://www.researchgate.net/publication/262672371_A_Comparative_Study_of_Game_Tree_Searching_Methods. Acesso em: 4 maio 2022.

EVIL, Aussie. Pokemon Type Chart. **Wikipedia,** [*S. l.*], 11 jan. 2012. Disponível em: <https://pt.m.wikipedia.org/wiki/Ficheiro:Pokemon_Type_Chart.svg>. Acesso em: 28 abr. 2022

HO, Harrison; RAMESH, Varun. **Percymon: A Pokemon Showdown AI**. Github, 24 abr. 2017. Disponível em: https://github.com/rameshvarun/showdownbot. Acesso em: 5 mai. 2022.

IHARA, Hiroyuki; IMAI, Shunsuke; OYAMA, Satoshi; KURIHARA, Masahito. Implementation and evaluation of information set Monte Carlo Tree Search for Pokémon. In: **2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)**. IEEE, 2018. p. 2182-2187.

KALOSE, Akshay; KAYA, Kris; KIM, Alvin. Optimal battle strategy in pokemon using reinforcement learning. Web: https://web. stanford. edu/class/aa228/reports/2018/final151. pdf, 2018.

LARGHI, Nathália. Com pandemia, mercado de games cresce 140% no Brasil, aponta estudo. **Valor investe**, [*S. l.*], 23 jan. 2021. Disponível em: https://valorinveste.globo.com/objetivo/gastar-bem/noticia/2021/01/23/com-pandemia-mercado-de-games-cresce-140percent-no-brasil-aponta-estudo.ghtml. Acesso em: 30 mar. 2022.

LEE, Scott; TOGELIUS, Julian. Showdown AI Competition. **IEEE Conference on Computational Intelligence and Games 2017**, [*s. l.*], 2021.

LISBOA, Lucas A. **Poke_Agents_Minimax**. Github, 08 mai. 2022. Disponível em: https://github.com/lucasalisboa/Poke_Agents_Minimax. Acesso em: 8 mai. 2022.

LUO, Guangcong. **Pokémon Showdown**. v0.11.7. ed. Github, 10 jun. 2021. Disponível em: https://github.com/smogon/pokemon-showdown. Acesso em: 2 abr. 2022.

MONTES, Gustavo. Algorithm for Pokemon battle game using MinMax. **Reporte Tecnico RT-0002-2013**, [*s. l.*], 2013. Disponível em: https://www.researchgate.net/publication/259343975_Sistemas_Inteligentes_Reportes_Finales_Ago-Dic_2013. Acesso em: 28 abr. 2022.

NOGUEIRA, Nuno. A minimax tree example. **Wikipedia,** [*S. l.*], 04 dez. 2006. Disponível em: <https://en.wikipedia.org/wiki/Minimax#/media/File:Minimax.svg>. Acesso em: 28 abr. 2022

NORSTRÖM, Linus. Comparison of artificial intelligence algorithms for pokémon battles. Chalmers University Of Technology (Suécia). 2019.

PANUMATE, Chetprayoon; XIONG, Shuo; IIDA, Hiroyuki. An Approach to Quantifying Pokemon's Entertainment Impact with Focus on Battle. **2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence**. IEEE, 2015. p. 60-66.

PACHECO, Paulo. Fã de Pokémon, presidente eleito do Chile ganha Squirtle de ministro japonês: Gabriel Boric recebeu pelúcia um dia antes da posse. **Omelete**, [*S. l.*], 11 mar. 2022. Disponível em: https://www.omelete.com.br/mangas-animes/presidente-chile-squirtle. Acesso em: 5 abr. 2022.

PMARIGLIA. **Showdown**. Github, 26 mar. 2022. Disponível em: https://github.com/RemptonGames/Pokemon-Showdown-Agent. Acesso em: 5 mai. 2022.

RILL-GARCIA, Rodrigo. Reinforcement Learning for a Turn-Based Small Scale Attrition Game. 2018. Disponível em: https://web.stanford.edu/class/aa228/reports/2018/final151.pdf. Acesso em: 5 mai. 2022.

RUSSELL, Stuart; NORVIG, Peter. Inteligência artificial. **Tradução: Regina Célia Simille de Macedo. Consultoria Editorial e Revisão técnica: Dr. Flávio Soares Corrêa da Silva, Dra. Leliane Nunes de Barros and Dra. Renata Wassermann**, v. 3, p. 13-31, 2013.

SAHOVIC, Haris. **The pokemon showdown Python environment**: poke-env. v0.4.21. ed. Github, 11 nov. 2021. Disponível em: https://github.com/hsahovic/poke-env. Acesso em: 2 abr. 2022.

SANCHEZ, Miquel Llobet. **Learning complex games through self play-Pokémon battles**. 2018. Trabalho de Conclusão de Curso. Universitat Politècnica de Catalunya.

SARTINI, Brígida Alexandre et al. Uma introdução à teoria dos jogos. **Anais da II Bienal da Sociedade Brasileira de Matemática**, p. 25-29, 2004.

SIMÕES, David; LAU, Nuno; REIS, Simão; REIS, Luís Paulo. Competitive deep reinforcement learning over a pokémon battling simulator. In: **2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)**. IEEE, 2020. p. 40-45.

THE POKEMON COMPANY (Japão). **History**. [*S. l.*], 2022. Disponível em: https://corporate.pokemon.co.jp/en/aboutus/history/. Acesso em: 5 abr. 2022.

THE POKEMON COMPANY (Japão). **Pokémon in Figures**. [*S. l.*], 2022. Disponível em: https://corporate.pokemon.co.jp/en/aboutus/figures/. Acesso em: 5 abr. 2022.

WAKKA, Wagner. Mercado de games agora vale mais que indústrias de música e cinema juntas. **Canaltech**, [*S. l.*], 25 fev. 2021. Disponível em: https://canaltech.com.br/games/mercado-de-games-agora-vale-mais-que-industrias-de-musica-e-cinema-juntas-179455/. Acesso em: 30 mar. 2022.

VERDEAR, Daniel; VISSER, Ubbo. Ontology-based Knowledge System and Team Verification Tool for Competitive Pokémon. In: **The International FLAIRS Conference Proceedings**. 2021.

VICTORINO, Vinícius. Uniforme da seleção do Japão ganha estampa do Pikachu: Fornecedora de material esportivo, Adidas tem parceria com a Nintendo e a marca Pokémon. **Época Negócios**, [*S. l.*], 30 maio 2014. Disponível em: https://epocanegocios.globo.com/Essa-E-Nossa/noticia/2014/05/uniforme-da-selecao-do-japao-ganha-estampa-do-pikachu.html. Acesso em: 5 abr. 2022.

XU, Shuo. **Improving companion AI in small-scale attrition games**. McGill University (Canada), 2015.

XU, Shuo; VERBRUGGE, Clark. Heuristics for sleep and heal in combat. **2016 IEEE Conference on Computational Intelligence and Games (CIG)**. IEEE, 2016. p. 1-8.

Apêndices

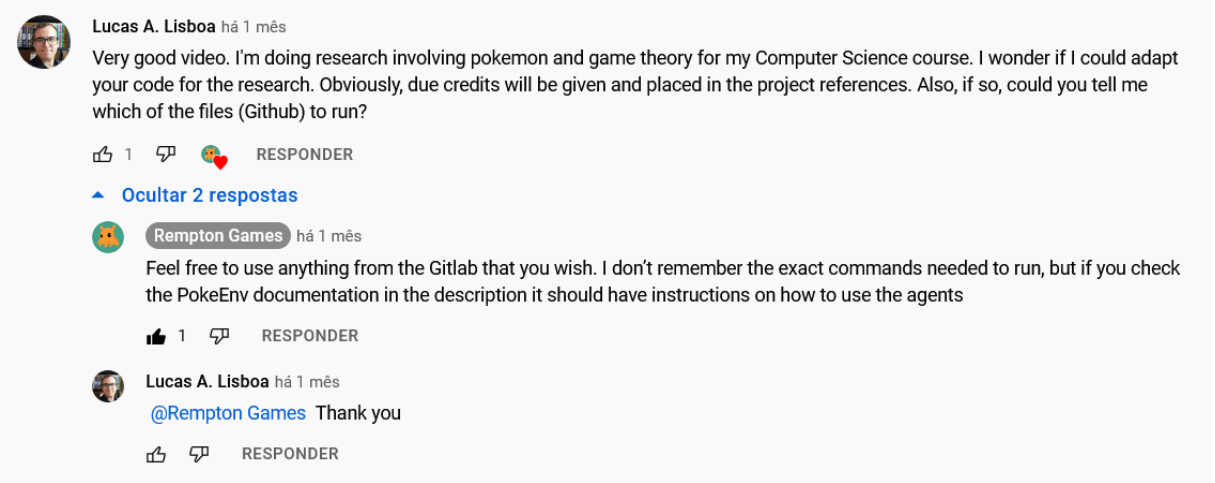
Apêndice A - Estrutura base para Busca em Árvore

```
import asyncio
import time
import sys

sys.path.append("..")

import BattleUtilities
from poke_env.player.random_player import RandomPlayer
from poke_env.player.player import Player
from poke_env.environment.move_category import MoveCategory
from GameNode import GameNode
from poke_env.environment.pokemon import Pokemon

class MiniMax(Player):

	previous_action = None
	maxDepth = 1

	def choose_move(self, battle):
	current_hp = {}
	for pokemon in battle.team.values():
		current_hp.update({pokemon : pokemon.current_hp})
	opponent_hp = {}
	for pokemon in battle.opponent_team.values():
		opponent_hp.update({pokemon : pokemon.current_hp})
	starting_node = GameNode(battle, battle.active_pokemon, current_hp,
battle.opponent_active_pokemon, opponent_hp, None, not battle.can_dynamax,
battle.active_pokemon.is_dynamaxed, not battle.opponent_can_dynamax,
battle.opponent_active_pokemon.is_dynamaxed, float('-inf'), None,
self.previous_action)
	if battle.active_pokemon.current_hp <= 0:
		self.pick_best_switch(starting_node, 0)
	else:
		self.minimax(starting_node, 0, True)
	child_nodes = starting_node.children
	best_score = float('-inf')
	best_node = None
	for child in child_nodes:
		if child.score >= best_score:
		best_score = child.score
		best_node = child
	if best_node == None:
		self.previous_action = None
		return self.choose_default_move(battle)
	self.previous_action = best_node.action
	return self.create_order(best_node.action)
```

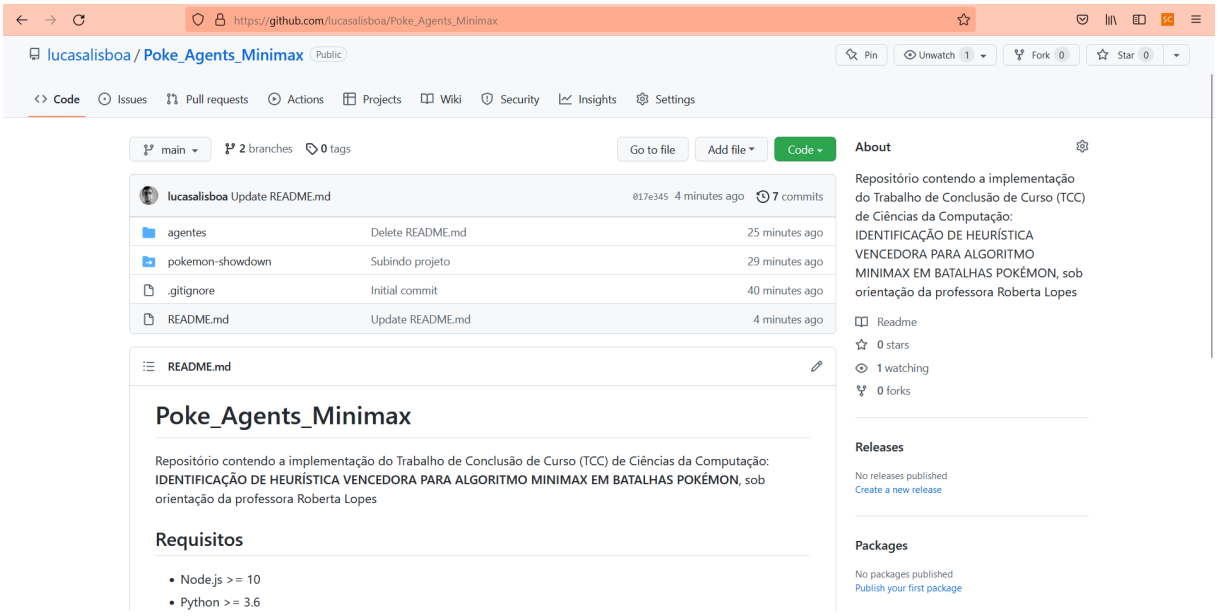
```
def minimax(self, node, depth, is_bot_turn):
if depth == self.maxDepth or self.is_terminal(node):
      self.score(node)
      return node.score
if is_bot_turn:
      score = float('-inf')
      bot_moves = node.generate_bot_moves()
      for move in bot_moves:
      child_score = self.minimax(move, depth, False)
      score = max(score, child_score)
      print
      node.score = score
      return score
else:
      score = float('inf')
      opponent_moves = node.generate_opponent_moves()
      if len(opponent_moves) > 0:
      for move in opponent_moves:
            child_score = self.minimax(move, depth + 1, True)
            score = min(score, child_score)
      else:
      score = float('-inf')
      node.score = score
      return score



def pick_best_switch(self, node, depth):
switches = node.add_bot_switches()
score = float('-inf')
for switch in switches:
      child_score = self.minimax(switch, depth, False)
      score = max(score, child_score)
node.score = score
return score


def is_terminal(self, node):
all_fainted = True
for pokemon in node.current_HP.keys():
      if node.current_HP[pokemon] > 0:
      all_fainted = False
if all_fainted:
      return True
all_fainted = True
for pokemon in node.opponent_HP.keys():
      if node.opponent_HP[pokemon]:
      all_fainted = False
if all_fainted:
      return True
return False
```

```
        def score(self, node):
        #FUNCAO A SER AVALIADA

async def main():
        start = time.time()

        enemy = RandomPlayer(
        battle_format="gen8randombattle",
        )
        minimax_player1 = MiniMax(
        battle_format="gen8randombattle",
        )

        await minimax_player1.battle_against(enemy, n_battles=500)

        print(
        "minimax player won %d / 500 battles against RandomPlayer (this took %f
seconds)"
        % (
                minimax_player1.n_won_battles, time.time() - start
        )
        )

if __name__ == "__main__":
        asyncio.get_event_loop().run_until_complete(main())
```

Apêndice B - Conversa com Compton



Apêndice C - Github do Projeto



Anexos

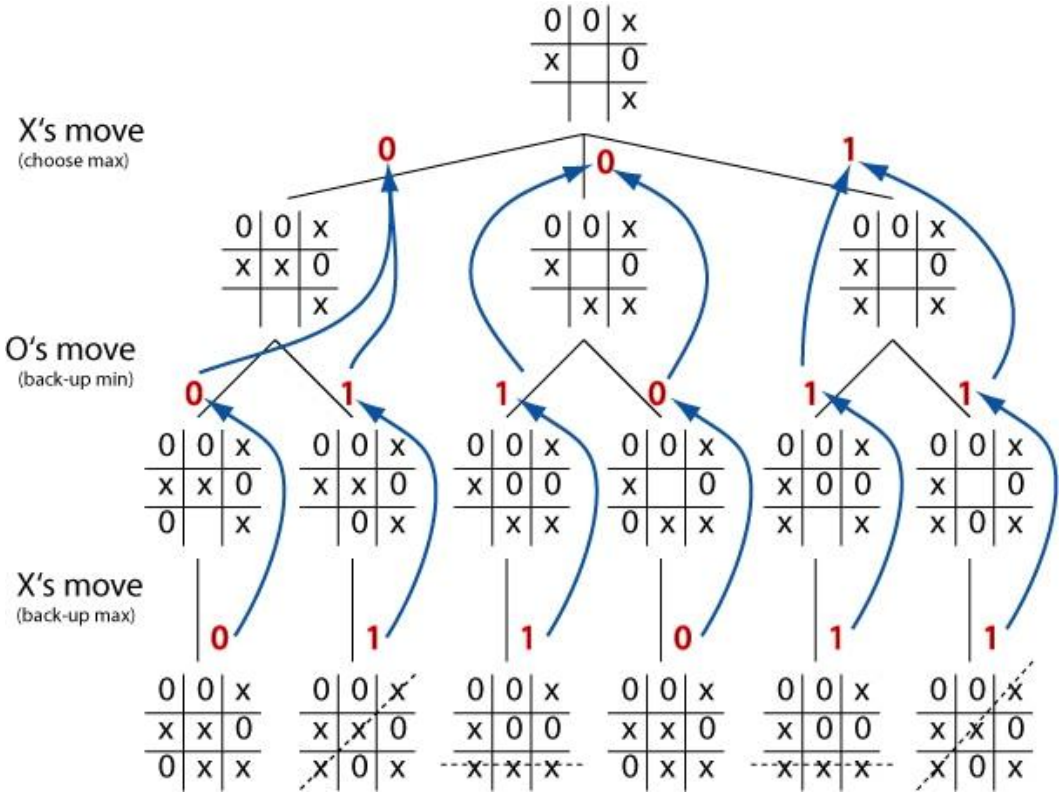
Anexo A - Presidente Chileno Gabriel Boric ganha Squirtle

Fonte: Pacheco (2022)

Anexo B - Camisa da Seleção Japonesa de Futebol

Fonte: Victorino (2014)

Anexo C - Minimax aplicado ao Jogo da Velha



Autor: Becker (2019)

Anexo D

```
git clone https://github.com/smogon/pokemon-showdown.git
cd pokemon-showdown
npm install
cp config/config-example.js config/config.js
node pokemon-showdown start --no-security
```

Autor: Sahovic (2021)

Anexo E

```
pip install poke-env
```

Autor: Sahovic (2021)