



UNIVERSIDADE DO ESTADO DO AMAZONAS - UEA
ESCOLA SUPERIOR DE TECNOLOGIA - EST
CURSO DE ENGENHARIA DE COMPUTAÇÃO



ANÁLISE COMPARATIVA DE TÉCNICAS DE NEUROEVOLUÇÃO NO
APRENDIZADO DO JOGO *CHROME DINO*

LUIZ PEDRO GADELHA DA SILVA

Manaus – AM

Junho – 2026

LUIZ PEDRO GADELHA DA SILVA

**ANÁLISE COMPARATIVA DE TÉCNICAS DE NEUROEVOLUÇÃO NO
APRENDIZADO DO JOGO *CHROME DINO***

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Computação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Computação.

Orientadora: Profa. Dra. Áurea Hileia da Silva Melo

Manaus – AM

Junho – 2026

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).
Sistema Integrado de Bibliotecas da Universidade do Estado do Amazonas.

S586a	Silva, Luiz Pedro Gadelha da Análise Comparativa de Técnicas de Neuroevolução no Aprendizado do Jogo Chrome Dino / Luiz Pedro Gadelha da Silva. Manaus : [s.n], 2026. 86 f.: color.; 21.0 cm. TCC - Graduação em Engenharia de Computação- Universidade do Estado do Amazonas, Manaus, 2026. Inclui Bibliografia. Orientador: Melo, Áurea Hileia da Silva. 1. Neuroevolução. 2. Algoritmos Genéticos. 3. Estratégias Evolutivas. 4. Redes Neurais Artificiais. 5. Chrome Dino. I. Melo, Áurea Hileia da Silva (Orient.) II. Universidade do Estado do Amazonas. III. Título CDU(1997)62:004
-------	--



FOLHA DE APROVAÇÃO

ANÁLISE COMPARATIVA DE TÉCNICAS DE NEUROEVOLUÇÃO NO APRENDIZADO DO JOGO CHROME DINO

LUIZ PEDRO GADELHA DA SILVA

Trabalho de Conclusão de Curso defendido e aprovado pela banca avaliadora constituída pelos professores:

Profª Dra. Áurea Hiléia da S. Melo
Presidente

Prof. MSc. Fabrizio Honda Franzoia
Avaliador

Prof. Dr. Jucimar Maia da Silva Junior
Avaliador

Manaus, 24 de junho de 2026

Dedicatória

Dedico este trabalho à minha mãe, Geise, e ao meu pai, Jorge, que sempre se esforçaram na minha criação, marcando as pedras no caminho e mostrando o caminho certo a seguir, como na música do Boi-Bumbá Caprichoso, assim como, encorajando-me a enfrentar os desafios que surgiram nessa jornada.

À Paula, minha parceira, aos meus irmãos, minha família, meus amigos da graduação, colégio, jogos online e todos que me apoiaram a seguir, estiveram presente ao meu lado, ofereceram ajuda, auxiliaram-me ou inspiraram-me a escrever este trabalho.

Aos remixes de discoteca, funk e música eletrônica nos canais do Youtube “@bookclubradio”, “@iloveyoueze”, “@booch” e tantos outros que tornaram a escrita deste trabalho surpreendentemente mais leve.

À professora Áurea Melo, que sacrificou momentos de lazer para que eu pudesse receber as orientações necessárias, compartilhando conhecimentos valiosos e contribuindo de forma inestimável com a minha formação.

Esta é uma conquista conjunta de todos esses, nada teria sido possível sozinho.

Resumo

A Neuroevolução (NE), campo da Ciência da Computação que utiliza técnicas da Computação Evolucionária (CE) para otimizar e treinar Redes Neurais Artificiais (RNA), apresenta-se como uma abordagem robusta para a criação de agentes autônomos para jogos. A definição dos operadores genéticos utilizados é uma importante etapa na elaboração de um método de NE, devido à influência deles na habilidade de evitar a convergência prematura do modelo de RNA. Nesse contexto, este trabalho apresenta os resultados de uma análise comparativa de desempenho entre o Algoritmo Genético (AG) canônico, aliado à técnica de Elitismo, e a Estratégia Evolutiva (EE) de Rechenberg (μ, λ) , aplicados no treinamento de uma rede neural para o aprendizado do jogo *Chrome Dino*. Para isso, estendeu-se a base de código *Dinossauro-Google*, criando classes que implementam tanto os métodos de CE comparados quanto o padrão de projeto *Strategy*, de forma a flexibilizar a alternância de operadores genéticos durante os testes realizados. Os resultados dos experimentos finais indicaram que a abordagem de AG alcançou os maiores picos de aptidão e a maior média final, enquanto a abordagem de EE apresentou um desempenho mais regular entre as neuroevoluções experimentadas. Um teste U de Mann-Whitney bicaudal, com nível de significância $\alpha = 0,05$, no entanto, não encontrou evidência suficiente para afirmar que as distribuições de aptidão final das duas abordagens são estatisticamente diferentes. Como contribuições, este trabalho oferece evidências quantitativas sobre o impacto da escolha do algoritmo evolutivo e do hiperparâmetro de mutação nas distribuições de aptidão final de neuroevoluções aplicadas ao jogo *Chrome Dino*, além de disponibilizar uma extensão reutilizável da base de código *Dinossauro-Google*, na qual o uso do padrão de projeto *Strategy* facilita a investigação de novos operadores genéticos em pesquisas futuras.

Palavras-Chave: Neuroevolução; Algoritmos Genéticos; Estratégias Evolutivas; Redes Neurais Artificiais; Chrome Dino.

Abstract

Neuroevolution (NE), a field of Computer Science that uses Evolutionary Computation (EC) techniques to optimize and train Artificial Neural Networks (ANNs), presents itself as a robust approach for creating autonomous agents for games. Defining the genetic operators used is an important step in developing a NE method, due to their influence on the ability to avoid premature convergence of the ANN model. In this context, this work presents the results of a comparative performance analysis between the canonical Genetic Algorithm (GA) with elitism and Rechenberg’s Evolution Strategy (ES) (μ, λ) , applied to the training of a neural network for learning the game Chrome Dino. To this end, the *Dinossau-ro-Google* codebase was extended, creating classes that implement both the compared EC methods and the Strategy design pattern, in order to allow for flexible switching of genetic operators during the tests performed. The results of the final experiments indicated that the GA approach achieved the highest fitness peaks and the highest final mean, while the ES approach showed more consistent performance among the neuroevolutions tested. A two-tailed Mann-Whitney U test, with a significance level of $\alpha = 0.05$, however, found insufficient evidence to claim that the final fitness distributions of the two approaches are statistically different. As contributions, this work offers quantitative evidence regarding the impact of the choice of evolutionary algorithm and the mutation hyperparameter on the final fitness distributions of neuroevolution applied to the Chrome Dino game, in addition to providing a reusable extension of the *Dinossau-ro-Google* codebase, where the use of the Strategy design pattern facilitates the investigation of new genetic operators in future research.

Keywords: Neuroevolution; Genetic Algorithms; Evolution Strategies; Artificial Neural

Networks; Chrome Dino.

Sumário

Lista de Tabelas	xi
Lista de Figuras	xiii
1 INTRODUÇÃO	1
1.1 Justificativa	3
1.2 Objetivos	3
1.2.1 Objetivo Geral	3
1.2.2 Objetivos Específicos	4
1.3 Metodologia	4
1.4 Tecnologias	6
1.5 Organização do Trabalho	8
2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS CORRELATOS	9
2.1 Computação Evolucionária	9
2.1.1 Algoritmo Evolucionário	10
2.2 Algoritmo Genético	14
2.2.1 Seleção de progenitores	16
2.2.2 Operadores genéticos	17
2.2.3 Recombinação (Cruzamento)	17
2.2.4 Mutação no Algoritmo Genético	18
2.2.5 Seleção de sobreviventes	18

2.3	Estratégia Evolutiva	19
2.3.1	Seleção de progenitores	20
2.3.2	Mutação na Estratégia Evolutiva	21
2.3.3	Seleção de sobreviventes	21
2.4	Redes Neurais Artificiais	22
2.4.1	Limiar de ativação	25
2.4.2	Função de Ativação	25
2.4.3	Algoritmo de aprendizado	28
2.5	Neuroevolução	30
2.5.1	Neuroevolução em jogos	30
2.6	O Jogo <i>Chrome Dino</i>	31
2.7	Trabalhos Relacionados	33
2.7.1	Avaliação da eficácia das Estratégias Evolutivas no treinamento de Redes Neurais Perceptron aplicadas a Jogos Digitais	34
2.7.2	<i>Learning Dark Souls With Neuroevolution</i>	35
2.7.3	Comparação de Algoritmos no aprendizado do jogo 2048	36
2.7.4	Neuroevolução Aplicada na Conclusão de Jogos de Plataforma	37
2.7.5	Aprendizado Evolucionário de Máquina em Jogos	39
2.7.6	Base de código <i>Dinossauro-Google</i> para Neuroevolução	39
2.7.7	Conclusão	42
3	Desenvolvimento	44
3.1	Definição do Modelo e Configuração Evolutiva	44
3.2	Preparação do Ambiente de Execução	49
3.2.1	Preparação do ambiente de desenvolvimento	49
3.2.2	Extensão da base de código	50
3.2.3	Execução automatizada da Neuroevolução	52
4	Resultados e Discussões	55

4.1	Execução dos processos de neuroevolução	55
4.1.1	Ciclo 1 de neuroevoluções	57
4.1.2	Ciclo 2 de neuroevoluções	64
4.1.3	Ciclo 3 de neuroevoluções	72
4.2	Análise dos Resultados	78
4.2.1	Limitações	82
5	Considerações Finais	83

Lista de Tabelas

2.1	Comparação entre trabalhos relacionados e o presente estudo	43
4.1	Organização dos experimentos por ciclo de neuroevolução	56
4.2	Condições de execução do Ciclo 3 de neuroevolução (confronto final)	72
4.3	Resultado do teste de Shapiro-Wilk de normalidade para a melhor aptidão das NEs de AG-F e EE-F no Ciclo 3 ($n = 24$ por experimento, $\alpha = 0,05$).	80
4.4	Resultado do teste U de Mann-Whitney (bicaudal) para a comparação das dis- tribuições da melhor aptidão obtida por AG-F e EE-F no Ciclo 3 ($n = 24$ por experimento, $\alpha = 0,05$).	80

Lista de Figuras

1.1	Metodologia do trabalho	5
2.1	Esquema geral de um Algoritmo Genético. Fonte: Cunha, Takahashi e Antunes (2012)	15
2.2	Esquema geral de uma Estratégia Evolutiva (μ, λ).	20
2.3	Algoritmo do Neurônio Perceptron (adaptado de Millington (2006))	24
2.4	Captura de tela exibindo a tela de início do jogo <i>Chrome Dino</i> em uma janela do navegador <i>Google Chrome</i>	32
2.5	Captura de tela exibindo a tela do jogo <i>Chrome Dino</i> durante uma jogatina em uma janela do navegador <i>Google Chrome</i>	32
2.6	Captura de tela exibindo a tela de fim do jogo <i>Chrome Dino</i> em uma janela do navegador <i>Google Chrome</i>	33
2.7	Ilustração da rede neural do agente inteligente criado. Fonte: Chen (2022). . . .	38
2.8	Métricas encontradas para a partida concluída de forma autônoma. Fonte: Chen (2022).	38
2.9	Interface de usuário exibida ao executar a base de código <i>Dinossauro-Google</i> . . .	40
3.1	Captura de tela exibindo a arquitetura escolhida para o modelo de RNA	46
3.2	Interface gráfica pós-extensão.	53
3.3	<i>Script</i> em <i>Bash</i> para automatizar a execução dos processos de neuroevolução. .	54
4.1	Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com algoritmo genético e mutação uniforme de 1% (AG-1).	58

4.2	Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs do AG-1 no Ciclo 1.	59
4.3	Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com algoritmo genético e mutação uniforme de 2% (AG-2).	60
4.4	Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs do AG-2 no Ciclo 1.	61
4.5	Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com algoritmo genético e mutação uniforme de 3% (AG-3).	62
4.6	Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs do AG-3 no Ciclo 1.	63
4.7	Comparação das curvas de convergência do melhor <i>fitness</i> por geração entre AG-1, AG-2 e AG-3 (média entre os 12 treinamentos).	64
4.8	Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com estratégia evolutiva e mutação gaussiana de $\sigma = 0,01$ (EE-010). . . .	65
4.9	Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs da EE-010 no Ciclo 2.	66
4.10	Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com estratégia evolutiva e mutação gaussiana de $\sigma = 0,05$ (EE-050). . . .	67
4.11	Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs da EE-050 no Ciclo 2.	68
4.12	Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com estratégia evolutiva e mutação gaussiana de $\sigma = 0,10$ (EE-100). . . .	69
4.13	Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs da EE-100 no Ciclo 2.	70
4.14	Comparação das curvas de convergência do melhor <i>fitness</i> por geração entre EE-010, EE-050 e EE-100 (média entre os 12 treinamentos).	71

4.15	Gráficos de linha da melhor aptidão por geração (com 50 partidas/geração) para as 24 neuroevoluções realizadas no Ciclo 3 com o AG-F, a combinação campeã do Ciclo 1 (AG e mutação uniforme de 3%).	73
4.16	Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs do AG-F no Ciclo 3.	75
4.17	Gráficos de linha da melhor aptidão por geração (com 50 partidas/geração) para as 24 neuroevoluções realizadas no Ciclo 3 com a EE-F, a combinação campeã do Ciclo 2 (EE e mutação gaussiana de $\sigma = 0,05$).	76
4.18	Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs da EE-F no Ciclo 3.	77
4.19	Comparação das curvas da média da melhor aptidão por geração (com ± 1 desvio padrão) entre AG-F e EE-F no Ciclo 3 ($n = 24$ neuroevoluções, 50 partidas/geração).	78

INTRODUÇÃO

A Otimização é o campo da ciência cujas técnicas visam determinar os extremos (máximos ou mínimos) de funções (CUNHA; TAKAHASHI; ANTUNES, 2012). De forma prática, a função a ser otimizada, muitas vezes chamada de função-objetivo, representa um fator que nos interessa em um sistema. Nesse caso, encontrar seu extremo significa identificar a melhor maneira possível de projetar ou operar esse sistema para maximizar ou minimizar o fator de interesse. Por exemplo, encontrar as melhores formas de minimizar o tempo gasto diariamente no trânsito, maximizar a produção de peças em uma fábrica ou minimizar os gastos mensais de uma família são tarefas comuns de otimização.

Existem métodos clássicos de otimização que, seguramente, encontram a solução para maximizar ou minimizar uma função-objetivo. Esses métodos, no entanto, apoiam-se em premissas fortes como a unimodalidade e a linearidade. Essa última, por exemplo, é a premissa do método SIMPLEX (DANTZIG, 1963), que assegura a solução de problemas cuja função-objetivo é uma função linear, ou seja, a saída é proporcional à entrada.

Contudo, quando solucionamos problemas do mundo real é muito comum que a função-objetivo não atenda às premissas necessárias para o uso dos métodos clássicos. Essa limitação motivou o desenvolvimento de heurísticas: métodos focados em obter boas soluções sem a necessidade de garantias teóricas de se atingir o ótimo global exato (CUNHA; TAKAHASHI; ANTUNES, 2012).

Nesse contexto, a Computação Evolutiva (CE), campo da Ciência da Computação, destaca-se como um conjunto de heurísticas inspiradas em metáforas biológicas (EIBEN; SMITH, 2015), como a teoria darwiniana da evolução natural. Historicamente, o desenvolvimento dessas técnicas aconteceu nos meados da década de 1970: nos Estados Unidos, John Holland formalizou os

Algoritmos Genéticos (AGs) (HOLLAND, 1975) e, na Alemanha, Ingo Rechenberg e Hans-Paul Schwefel inventaram as Estratégias Evolutivas (EEs) (RECHENBERG, 1973).

O uso de técnicas de Computação Evolutiva no treinamento de Redes Neurais Artificiais (RNAs) dá origem ao campo da Neuroevolução (NE) (RISI; TOGELIUS, 2015). Nessa abordagem, os algoritmos evolutivos, como AGs e EEs, podem participar da otimização de um grande número de parâmetros presentes em um modelo de RNA, como os pesos das conexões entre neurônios e a topologia do modelo. Em determinados casos, é possível substituir completamente métodos tradicionais de treinamento, como o *backpropagation*, por um algoritmo evolutivo (RISI; TOGELIUS, 2015).

Ainda segundo Risi e Togelius (2015), desde sua origem, a Neuroevolução tem sido amplamente investigada como uma abordagem robusta para o desenvolvimento de agentes inteligentes em jogos digitais. Os jogos oferecem ambientes dinâmicos, complexos e controlados, ideais para testar a capacidade adaptativa de algoritmos de otimização. Estudos demonstram, por exemplo, que agentes treinados via Neuroevolução podem obter desempenhos competitivos ou até sobre-humanos em jogos como Quake II (PARKER; BRYANT, 2008) e títulos do console Atari (MNIH et al., 2015).

A literatura, no entanto, aponta divergências históricas e práticas sobre o uso de certos operadores genéticos: enquanto a escola dos Algoritmos Genéticos de Holland (1975) tradicionalmente prioriza a recombinação como motor de busca, a escola das Estratégias Evolutivas de Rechenberg (1973) defende a mutação (e sua auto-adaptação) como o mecanismo primário de inovação.

Alguns trabalhos recentes (PEREIRA, 2023; CHEN, 2022) exploram a aplicação dessas técnicas em jogos específicos, mas a determinação da configuração ótima desses operadores para garantir a convergência em jogos de reação rápida, como o *Chrome Dino*, ainda carece de análises comparativas aprofundadas.

Dada a necessidade de equilibrar a exploração do espaço de busca, o refinamento de soluções e considerando as diferentes filosofias de projeto entre Algoritmos Genéticos e Estratégias Evolutivas, este trabalho investiga a maneira com que tanto a escolha de diferentes algoritmos

evolutivos quanto o uso de diferentes valores para o hiperparâmetro de mutação influenciam a eficácia de uma Rede Neural Artificial treinada via Neuroevolução para jogar o *Chrome Dino*.

1.1 Justificativa

Apesar do potencial da neuroevolução, a definição e configuração dos operadores genéticos permanecem uma importante decisão de projeto para garantir a eficácia do algoritmo. A literatura clássica apresenta divergências teóricas sobre o motor primário nos algoritmos evolutivos: enquanto a escola dos Algoritmos Genéticos enfatiza a recombinação (cruzamento) como meio de trocar informações úteis entre soluções avaliadas a cada geração (HOLLAND, 1975), as Estratégias Evolutivas priorizam a mutação como mecanismo essencial de inovação e prevenção contra a estagnação em ótimos locais (RECHENBERG, 1973). A escolha inadequada desses operadores pode levar a convergências prematuras ocasionadas pela perda de diversidade da população devido à técnica tornar-se incapaz de acessar áreas promissoras do espaço de soluções, onde se encontram indivíduos de melhor ou igual valor de aptidão (CUNHA; TAKAHASHI; ANTUNES, 2012).

Dessa forma, realizar uma análise comparativa sobre qual das abordagens, dentre os Algoritmos Genéticos acrescidos de Elitismo e as Estratégias Evolutivas, oferece o melhor desempenho final, usando-se como estudo de caso o jogo *Chrome Dino*, representa uma relevante contribuição acadêmica.

1.2 Objetivos

A seguir, são apresentados os objetivos geral e específicos deste trabalho.

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é realizar um estudo comparativo entre o Algoritmo Genético de Holland (1975) com Elitismo e a Estratégia Evolutiva de Rechenberg (1973) no treinamento de uma rede neural para o aprendizado do jogo *Chrome Dino*, fornecendo evidências quantitativas

tanto sobre o impacto desses métodos no treinamento de redes neurais do tipo Perceptron Multicamadas quanto sobre qual dos modelos algorítmicos obtém melhor desempenho nos critérios de comparação definidos.

1.2.2 Objetivos Específicos

- Realizar estudos exploratórios para identificar bases de código aberto para Neuroevolução;
- Estabelecer critérios de comparação entre os algoritmos evolutivos e seus operadores genéticos;
- Determinar a arquitetura da Rede Neural Artificial empregada;
- Automatizar o processo de Neuroevolução para avaliar resultados de diferentes hiperparâmetros;
- Avaliar comparativamente o impacto das variações de algoritmos evolutivos e de hiperparâmetros de mutação por meio de análise gráfica e de testes estatísticos sobre os resultados obtidos nos treinamentos das redes neurais.

1.3 Metodologia

A metodologia deste trabalho foi estruturada em 5 etapas contemplando as disciplinas de Trabalho de Conclusão de Curso I (TCC I) e II (TCC II), quais sejam: Estudos exploratórios, Definição do Modelo e Configuração Evolutiva, Preparação do ambiente de execução, Execução dos Processos de Neuroevolução, e, por fim, a Análise dos resultados.

Quanto à sua natureza, esta pesquisa caracteriza-se como um Experimento Controlado, no qual se manipulam variáveis de interesse (o algoritmo evolutivo e o valor do hiperparâmetro de mutação) enquanto os demais hiperparâmetros do processo de Neuroevolução permanecem fixos, de modo a isolar o efeito dos fatores comparados sobre a aptidão final das redes neurais. Essa característica confere maior robustez à comparação e justifica o uso de testes estatísticos para a análise dos resultados.

A Figura 1.1 apresenta graficamente a referida metodologia com suas respectivas etapas em cada disciplina, as quais são descritas em seguida.

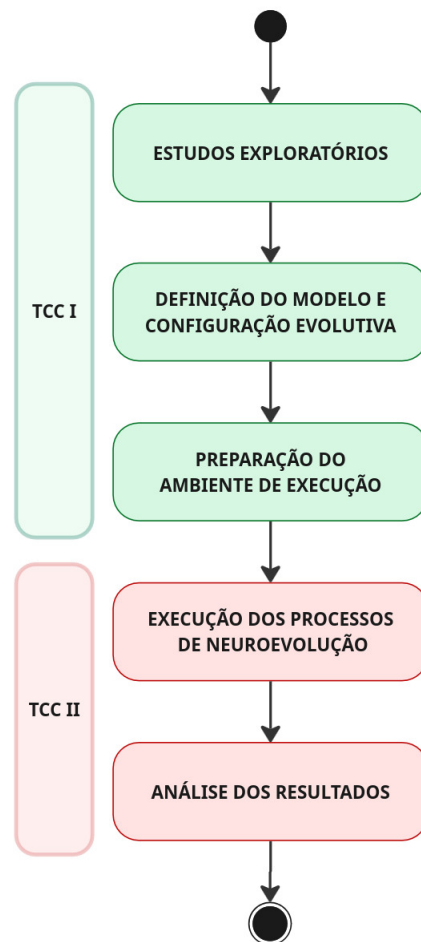


Figura 1.1: Metodologia do trabalho

1. **Estudos Exploratórios:** Responsável pelos diversos estudos da literatura para identificação e seleção de valores para o hiperparâmetro do operador genético de mutação dos Algoritmos Evolutivos e da arquitetura do modelo de Rede Neural Artificial, bem como o levantamento bibliográfico de trabalhos correlatos. Nessa etapa, também se realiza a procura de bases de código aberto capazes de executar processos de Neuroevolução.

O resultado dessa etapa, desenvolvida durante a disciplina de TCC I, é contemplado no capítulo de Fundamentação Teórica e Trabalhos Correlatos.

2. **Definição do Modelo e Configuração Evolutiva:** Essa etapa é responsável, a partir dos estudos da etapa anterior, pela definição dos hiperparâmetros do operador genético de mutação para os AEs, além da arquitetura para o modelo de RNA, a base de código aberto de NE e os critérios de comparação entre os AEs implementados.

Essa etapa foi realizada durante a disciplina de TCC I e está contemplada no capítulo de Desenvolvimento.

3. **Preparação do Ambiente de Execução:** Contemplada na disciplina de TCC I e no capítulo de Desenvolvimento, essa etapa foi dividida em 3 subetapas que são descritas a seguir:

- a) Subetapa onde são realizadas atividades de preparação do ambiente de desenvolvimento com a base de código aberto.
- b) Subetapa para realizar atividades de extensão da base com novas classes de código.
- c) Nessa subetapa, criam-se *scripts* de código para automatizar a execução da base de código.

4. **Execução dos Processos de Neuroevolução:** Desenvolvida durante a disciplina de TCC II e contemplada no capítulo de Resultados e Discussões, essa etapa é responsável pela execução dos processos de neuroevolução do modelo de RNA a partir de diversas combinações de AEs e hiperparâmetros para aprender a jogar o jogo *Chrome Dino*.

5. **Análise dos Resultados:** Também contemplada pela disciplina de TCC II e apresentada no capítulo de Resultados e Discussões, essa etapa é responsável por avaliar os resultados obtidos por cada combinação de AE e hiperparâmetros experimentados com base nos critérios de comparação, por meio de gráficos de linha e análise descritiva do desempenho relativo entre os AEs testados.

1.4 Tecnologias

Para a realização deste trabalho, foram utilizados ambientes de desenvolvimento, materiais de base e diversas linguagens de programação. Essas tecnologias são descritas a seguir.

- **Código-base, Repositório “Brccabral/Dinossauro-Google”:** Este repositório contém uma implementação do jogo Chrome Dino juntamente com classes de Rede Neural aplicadas à execução do jogo e uma estratégia evolutiva que utiliza um método de mutação original. O repositório do GitHub “Dinossauro-Google” do usuário “Brccabral” foi utilizado como código-base para o desenvolvimento das classes de Algoritmo Genético (**GenAlgorithm**) e de Estratégia Evolutiva de Rechenberg (**RechenbergMuLambdaES**). Foram realizadas algumas alterações neste código-base para habilitar a sua correta execução em ambientes Linux. A escolha por essa base justifica-se por se tratar de um repositório de código aberto que já implementa por completo o jogo *Chrome Dino*, uma RNA e um método de Neuroevolução, além de ser facilmente adaptável e extensível, conforme detalhado na Seção 2.7.6.
- **Linguagem de programação, C++:** Foi a linguagem utilizada na implementação do código-base e, portanto, também utilizada para a escrita das novas classes de algoritmos evolutivos e de seus operadores genéticos. Seu uso justifica-se pela necessidade de conformidade com a linguagem já adotada pela base de código.
- **Linguagem de programação, Shell Script:** Foi a linguagem utilizada para a escrita do *Script* que executa os experimentos de teste. Sua escolha justifica-se pela possibilidade de automação de tarefas repetitivas no ambiente Linux utilizado e pela experiência prévia do autor com a linguagem. Detalhes sobre o comportamento deste *Script* estão disponíveis na Seção 3.2.3 do capítulo de Desenvolvimento.
- **Linguagem de programação, Python:** Foi a linguagem utilizada para a análise dos resultados obtidos através dos experimentos de teste e para a geração dos gráficos apresentados no capítulo de Resultados e Discussões. Sua escolha justifica-se pela ampla adoção da linguagem em análise de dados, pela disponibilidade de bibliotecas para manipulação de dados, cálculos estatísticos e visualização, além da experiência prévia do autor.
- **Sistema Operacional, Ubuntu:** O sistema operacional Ubuntu foi escolhido como ambiente de execução dos experimentos de teste pois os recursos computacionais disponíveis

eram limitados e este sistema mostrou-se o mais rápido dentro das alternativas disponíveis - *Windows*, *Windows Subsystem for Linux* (WSL) e *Linux* (Ubuntu).

- **Editor de código, Visual Studio Code:** O editor de código *Visual Studio Code* foi escolhido como ambiente de desenvolvimento pela sua versatilidade e capacidade de edição de códigos em todas as linguagens utilizadas na implementação deste trabalho.

1.5 Organização do Trabalho

Além deste capítulo introdutório, o presente trabalho possui outros quatro, organizados da seguinte forma:

- **Fundamentação Teórica e Trabalhos correlatos:** Apresenta a base conceitual para compreensão do trabalho, tais como os conceitos de Algoritmo Genético, Rede Neural Artificial, Computação Evolutiva, entre outros, bem como as dissertações e artigos com temas relacionados.
- **Desenvolvimento:** Detalha a arquitetura da Rede Neural adotada, a configuração dos algoritmos evolutivos comparados, a preparação do ambiente, a extensão da base de código com as classes de AG e EE, a aplicação do padrão de projeto *Strategy* e a criação de *scripts* para execução automatizada dos experimentos.
- **Resultados e Discussões:** Apresenta os experimentos realizados e a análise comparativa de desempenho entre o AG e a EE, utilizando gráficos da melhor aptidão por geração para validar os resultados.
- **Considerações Finais:** Apresenta as conclusões obtidas a partir dos experimentos e as relaciona com os objetivos definidos para a pesquisa. O capítulo traz também direções para pesquisas subsequentes.

2. FUNDAMENTAÇÃO TEÓRICA E TRABALHOS CORRELATOS

Esse capítulo apresenta uma revisão literária necessária para o desenvolvimento deste trabalho. As seções a seguir tratam dos conceitos de Computação Evolucionária, Algoritmo Genético, Estratégia Evolutiva, Redes Neurais Artificiais e Neuroevolução, além de uma seção dedicada ao jogo *Chrome Dino*, os quais representam a base de conhecimento para a realização de todas as etapas da metodologia definida para alcançar os objetivos propostos. O capítulo é finalizado com uma discussão acerca dos trabalhos correlatos e da base de código aberto para Neuroevolução encontrada na literatura.

2.1 Computação Evolucionária

Segundo Eiben e Smith (2015), a Computação Evolucionária (CE), ou Computação Evolutiva, é uma área de pesquisa dentro da ciência da computação que consiste em um conjunto de técnicas amplamente inspiradas na teoria da seleção natural das espécies. Estas técnicas utilizam conceitos emprestados da Biologia como, por exemplo, população, indivíduo, cálculo de aptidão, seleção, cruzamento e mutação.

O argumento central da CE apoia-se principalmente no evidente poder demonstrado pela seleção natural ao ser capaz de originar, em cada espécie do nosso mundo, adaptações bastante específicas que auxiliam sua sobrevivência na natureza (BäCK; SCHWEFEL, 1993). No entanto, a área compreende também técnicas inspiradas em outros mecanismos de adaptação dos seres vivos além da seleção natural. Nesse contexto, é possível citar o Algoritmo Colônia

de Formigas, por exemplo, fundamentado no modo empregado por colônias de formigas para encontrar caminhos ótimos até uma fonte de alimento (CUNHA; TAKAHASHI; ANTUNES, 2012).

Historicamente, as origens da Computação Evolucionária podem ser traçadas para até os anos de 1940, décadas antes da revolução dos computadores, segundo Eiben e Smith (2015). De acordo com Fogel (1998), os conceitos basais das técnicas de CE figuram em múltiplas linhas independentes de pesquisa da década de 1950. Ainda segundo Eiben e Smith (2015), foi apenas na década de 1960, porém, que houve o surgimento de técnicas hoje conhecidas como Algoritmos Evolucionários (AE).

A subseção a seguir apresenta a definição de Algoritmo Evolucionário, assim como os conceitos compartilhados pelas técnicas de AE utilizadas neste trabalho.

2.1.1 Algoritmo Evolucionário

Os Algoritmos Evolucionários, assim denominadas as técnicas inspiradas na teoria evolutiva, baseiam-se no conhecimento coletivo desenvolvido dentro de uma população de indivíduos em que cada indivíduo corresponde a um ponto no espaço de possíveis soluções do problema que se deseja resolver (BäCK; SCHWEFEL, 1993).

Dentre os AE, é possível citar os Algoritmos Genéticos (AGs), desenvolvidos por Holland (1975), e as Estratégias Evolutivas (EEs), concebidas por Rechenberg (1973).

Alguns conceitos importantes definem as entidades que compõem um AE. Estes conceitos são:

- **População:** um grupo de possíveis soluções para resolver o problema.
- **Indivíduo:** elemento que faz parte de uma determinada população. Também chamado de cromossomo.
- **Gene:** característica elementar da possível solução que um indivíduo representa. Indivíduos são formados por um conjunto de genes.

- **Aptidão:** valor que representa o potencial de um indivíduo para solucionar o problema. Também chamado pelo termo em inglês *fitness*. Esse valor é calculado através de uma função denominada função-objetivo.
- **Progenitor:** indivíduo de uma população selecionado para originar descendentes em uma determinada geração.
- **Descendente:** indivíduo resultante da aplicação dos operadores genéticos sobre um ou mais progenitores.
- **Geração:** uma iteração do processo de otimização. Uma geração compreende, por exemplo, as etapas de cálculo de aptidão, seleção e aplicação dos operadores genéticos.

A seguir, são definidos os conceitos compartilhados pelos AEs empregados neste trabalho, como **representação cromossômica**, **avaliação**, **mutação** e **condição de parada**. Mais à frente, são detalhados como o Algoritmo Genético e a Estratégia Evolutiva empregam esses elementos, destacando os operadores e hiperparâmetros específicos de cada técnica.

Representação cromossômica

A representação cromossômica é uma escolha relacionada à forma com que representaremos os genes de um indivíduo. Outras definições, como o mecanismo de avaliação e os operadores genéticos, dependem desta escolha.

A **Representação Real** consiste em utilizar números reais para representar os genes. Esta representação soa mais natural quando tratamos de problemas com variáveis de decisão de distribuição contínua, pois cada gene representa diretamente uma variável de decisão. Neste trabalho, os genes assumem valores no intervalo $[-1, 1]$ e codificam, por meio de codificação direta, os pesos sinápticos e os limiares de ativação da Rede Neural Artificial empregada na Neuroevolução.

Em Holland (1975), foi proposta a representação cromossômica utilizando código binário, na qual os genes são codificados por *bits* (0's e 1's). A **Representação Binária** pode trazer desvantagens em problemas com restrições (HINTERDING, 1999) ou cujas variáveis são valores

reais. A **Representação em Permutação**, por sua vez, exige que a ordem dos genes importe e que não haja valores repetidos, sendo adequada a problemas de sequenciamento como o caixeiro viajante (EIBEN; SMITH, 2015).

Neste trabalho, escolheu-se representar os genes de um indivíduo a partir da Representação Real em ambos os AEs utilizados.

Avaliação

O mecanismo de avaliação consiste em uma função, comumente chamada de função-objetivo, que calcula o valor de aptidão que representa a qualidade de um determinado indivíduo para solucionar o problema tratado. Esta função é essencial para as etapas de seleção e de aplicação dos operadores genéticos, de modo que sua definição constitui uma importante escolha de projeto.

No Problema da Mochila, por exemplo, considerando uma solução potencial S qualquer, é possível utilizar a função-objetivo definida na equação a seguir:

$$Aptidao(S) = \sum_{i=1}^N x_i \times V_i - Pen(S)$$

em que $x_i \in \{0, 1\}$, V_i se refere ao valor do objeto Obj_i e $Pen(S)$ penaliza soluções que violam as restrições do problema.

Neste trabalho, a função-objetivo utilizada calcula o valor de aptidão de cada indivíduo a partir da distância percorrida, em pixels, pelo agente controlado no jogo *Chrome Dino* até o momento de sua morte em cada partida. Como cada geração compreende múltiplas partidas, a aptidão final do indivíduo corresponde à média aritmética das distâncias obtidas nessas partidas.

Mutação

A mutação é um operador genético que introduz modificações aleatórias em um ou mais genes de um indivíduo (EIBEN; SMITH, 2015). Seu principal objetivo é inserir e manter a diversidade genética da população ao longo das gerações. Sem a aplicação da mutação, o algoritmo estaria limitado unicamente ao conjunto de genes presentes na população inicial, o que frequentemente

leva à convergência prematura (CUNHA; TAKAHASHI; ANTUNES, 2012).

A intensidade da mutação é controlada por hiperparâmetros distintos em cada AE implementado neste trabalho. No AG, utiliza-se a **taxa de mutação**, que define a probabilidade de cada gene ser alterado. Na EE, a magnitude das mutações é controlada pelo desvio padrão da distribuição normal utilizada para gerar os valores das mutações. Segundo Eiben e Smith (2015), valores muito elevados para esses parâmetros transformam o AE em uma busca essencialmente aleatória, enquanto valores muito baixos aumentam o risco de estagnação do algoritmo. Os valores empregados em cada AE foram alvo de experimentos realizados no capítulo de Resultados e Discussões.

Existem, na literatura, diversos métodos de mutação para representação real de cromossomos (EIBEN; SMITH, 2015). Na **Mutação Uniforme**, os genes selecionados são substituídos por valores completamente novos, sorteados de forma aleatória e uniforme dentro do domínio do gene (CUNHA; TAKAHASHI; ANTUNES, 2012; EIBEN; SMITH, 2015). Na **Mutação Gaussiana**, soma-se ao valor atual do gene um incremento extraído de uma distribuição normal centrada em zero, sendo o desvio padrão σ o principal hiperparâmetro de controle (EIBEN; SMITH, 2015; RECHENBERG, 1973). Na **Mutação por Multiplicação**, aplica-se um fator de escala ao valor original do gene, sendo particularmente útil quando os parâmetros do problema operam em ordens de magnitude distintas (CUNHA; TAKAHASHI; ANTUNES, 2012).

Além dos métodos descritos, há também um método denominado **Mutações Parciais** (PEREIRA, 2023). Nesse método, todos os indivíduos em uma população sofrem mutações. Mais especificamente, para cada indivíduo da população, uma quantidade de mutações é sorteada aleatoriamente dentre 1 e a quantidade máxima de mutações m . Com a quantidade de mutações definida, para cada mutação, sorteia-se um gene do indivíduo para mutar. Para cada gene selecionado, executa-se o método de Mutação Uniforme, isto é, sorteia-se um novo valor dentro do domínio definido para os genes.

Neste trabalho, o AG implementado emprega o método de **Mutação Uniforme** e a EE adotada emprega o método de **Mutação Gaussiana**, ambos descritos nessa subseção.

Condição de parada

A condição de parada, também chamada de critério de terminação, é a condição determinante do fim do algoritmo evolucionário (EIBEN; SMITH, 2015). Este critério é verificado ao final de cada iteração (ou geração).

Diversas condições de parada são utilizadas pela literatura. Em Yannakakis e Togelius (2018), é sugerido o uso de um **Número máximo de gerações** ou de um **Valor de aptidão alvo**. Em Eiben e Smith (2015), são propostas ainda as condições de **Limite de tempo** e de **Estagnação**, onde o processo é interrompido caso a aptidão não apresente melhorias significativas ao longo de um certo número de gerações.

Neste trabalho, são verificadas duas condições de parada nos AEs desenvolvidos. São elas: o número máximo de gerações e o valor de aptidão alvo.

2.2 Algoritmo Genético

Os Algoritmos Genéticos (AGs) são uma técnica de Computação Evolutiva desenvolvida por Holland (1975) que avalia simultaneamente um grupo de possíveis soluções para um dado problema. Esse grupo é submetido a repetidas transformações com o intuito de obter uma solução de alta qualidade. São estas transformações que dão o caráter evolutivo ao processo, uma vez que são as principais responsáveis pelo aprimoramento das soluções. A busca nos AGs é tradicionalmente conduzida pela **recombinação** entre soluções. Neste trabalho, o AG é empregado como algoritmo de aprendizado de uma Rede Neural Artificial via Neuroevolução.

Com os conceitos essenciais de um AE definidos na seção anterior, conceitos esses válidos também ao trabalhar com AGs, é possível definir um esquema geral do funcionamento de um AG. Esse esquema é visualizado pelo algoritmo apresentado na Figura 2.1 e no passo a passo a seguir:

- (i) No passo inicial, define-se um valor t que representará o número da geração em cada etapa do processo. Isso é necessário pois o processo é composto por gerações e cada geração possui uma população relacionada.

- (ii) Em seguida, é criada uma população inicial de indivíduos, chamada de $P(t)$, de forma aleatória.
- (iii) No passo 3, avalia-se a população $P(t)$ relacionando, a cada indivíduo desta população, um valor de aptidão.
- (iv) No 4º passo, uma nova população, chamada $P'(t)$, é selecionada a partir da população $P(t)$. Essa seleção é feita com base nos valores de aptidão dos indivíduos. Um indivíduo pertencente a $P'(t)$ é denominado um Progenitor.
- (v) No passo 5, a população da geração seguinte denominada $P(t+1)$ é criada a partir da população $P'(t)$ através dos operadores genéticos de recombinação e mutação.
- (vi) E então, no passo 6, a população $P(t+1)$ é avaliada relacionando um valor de aptidão a cada indivíduo desta população.
- (vii) No passo 7, o valor de t é incrementado para indicar o começo de uma nova geração.
- (viii) No 8º passo, um critério de terminação, ou condição de parada, é verificado. Caso a condição tenha sido atingida, encerra-se o processo e tem-se o fim da evolução. Caso contrário, o processo volta ao passo 4.

Algoritmo 1 Esquema Geral de um Algoritmo Genético

- 1: $t \leftarrow 0$
 - 2: Gera a população inicial $P(t)$ de forma aleatória
 - 3: Avaliar os indivíduos de $P(t)$
 - 4: **repita**
 - 5: Selecionar progenitores $P'(t)$ a partir de $P(t)$
 - 6: Aplicar operadores genéticos a $P'(t)$ obtendo a nova população $P(t+1)$
 - 7: Avaliar $P(t+1)$
 - 8: $t \leftarrow t + 1$
 - 9: **até** critério de terminação atingido
 - 10: Devolver resultado final da otimização
-

Figura 2.1: Esquema geral de um Algoritmo Genético. Fonte: Cunha, Takahashi e Antunes (2012)

Analisando o esquema geral descrito no passo a passo anterior e na Figura 2.1, é possível identificar alguns conceitos não definidos até então: o mecanismo de **seleção de progenitores**, os **operadores genéticos** de **recombinação** e **mutação** específicos do AG, bem como o mecanismo de **seleção de sobreviventes**. Nas subseções a seguir, são definidos esses conceitos e a forma com que são empregados neste trabalho.

2.2.1 Seleção de progenitores

A seleção de progenitores consiste em escolher, em uma dada geração, indivíduos com bons valores de aptidão para originar descendentes na geração seguinte. Esse é um dos mecanismos mais importantes dos AGs, pois direciona a busca por soluções potenciais para regiões promissoras do espaço de soluções.

Neste mecanismo, há um conceito importante conhecido como **pressão seletiva**. A pressão seletiva representa a diferença entre a probabilidade que os melhores indivíduos têm de serem selecionados e a probabilidade de seleção dos piores. Quando elevada, pode ocasionar convergência prematura e deterioração da diversidade entre as soluções potenciais.

Na 1ª edição do livro *"Adaptation in Natural and Artificial Systems"* de Holland (1975) é proposta a **seleção por roleta** (ou proporcional). Nesse método, a probabilidade de um indivíduo ser selecionado é diretamente proporcional à razão entre sua aptidão e a soma das aptidões de todos os indivíduos de sua população.

Outro método de seleção muito utilizado é a **seleção por torneio** (EIBEN; SMITH, 2015). Nessa seleção, cada progenitor é selecionado a partir de um torneio entre K indivíduos escolhidos aleatoriamente dentro da população, sendo K um parâmetro chamado de tamanho do torneio. O vencedor é o indivíduo de maior aptidão dentre os K escolhidos; ao variar K, é possível controlar a pressão seletiva do algoritmo.

Neste trabalho, o AG implementado utiliza o mecanismo de seleção de progenitores chamado seleção por roleta.

2.2.2 Operadores genéticos

Os operadores genéticos são técnicas inspiradas nos mecanismos de variação genética biológica. Em um AG, esses operadores atuam sobre os indivíduos escolhidos como progenitores e sobre os descendentes desses progenitores com o objetivo de originar novas soluções potenciais e assegurar um certo nível de diversidade entre os indivíduos da população.

Existem dois operadores genéticos fundamentais para um AG: a **recombinação** (ou **cruzamento**) e a **mutação**.

2.2.3 Recombinação (Cruzamento)

A recombinação, também chamada de cruzamento, é um operador binário que toma dois progenitores e tem como resultado um novo indivíduo. Esses indivíduos, aqui chamados de descendentes, herdarão uma combinação dos genes dos seus progenitores. A ideia por trás desse mecanismo é de que os descendentes poderão herdar boas características de ambos os progenitores, o que poderá resultar em um valor de aptidão maior que os de seus pais.

A recombinação é também definida como um mecanismo estocástico, pois depende de outros parâmetros probabilísticos: a probabilidade de aplicação da recombinação e o ponto de corte.

A probabilidade de aplicação da recombinação determina com que frequência a recombinação será efetivamente aplicada entre os indivíduos do conjunto de progenitores. Devido à importância da recombinação na criação de novos indivíduos, esse parâmetro é comumente um valor alto (muitas vezes entre 0.7 e 1.0).

O ponto de corte é um ponto escolhido aleatoriamente entre os genes dos progenitores, determinando quais segmentos serão combinados para formar os descendentes.

Em uma dada recombinação de dois progenitores de 100 genes cada, por exemplo, imaginemos que o gene de nº 65 foi escolhido aleatoriamente como o ponto de corte. Nesse caso, a partir desse ponto de corte, poderíamos dividir cada progenitor em 2 segmentos denominados: 1.1, compreendendo os genes de nº 1 (primeiro gene) ao nº 65 (ponto de corte) do 1º progenitor; e o segmento 1.2, compreendendo os genes de nº 66 (gene sucessor do ponto de corte) ao nº

100 (último gene) do 1º progenitor; e, de forma análoga, os segmentos 2.1, com genes de nº 1 ao nº 65, e 2.2, com genes nº 66 ao nº 100 do 2º progenitor. Como resultado da recombinação, o descendente é formado pela concatenação do segmento 1.1 com o segmento 2.2, herdando em parte genes de ambos os progenitores. Esse método, proposto por Holland (1975), é um dos operadores de cruzamento mais utilizados em AGs com representação binária ou real (EIBEN; SMITH, 2015; CUNHA; TAKAHASHI; ANTUNES, 2012).

Em casos em que a recombinação não acontece, devido à probabilidade de aplicação menor que 100%, o descendente é gerado por **clonagem**, isto é, por cópia integral de um dos progenitores selecionados. A clonagem complementa a recombinação ao manter, na população, cópias de indivíduos que foram selecionados como progenitor e que, por isso, são boas candidatas à solução do problema.

Neste trabalho, o AG implementado utiliza o mecanismo de recombinação definido por Holland (1975).

2.2.4 Mutação no Algoritmo Genético

Nos AGs, a mutação é aplicada aos descendentes gerados após a etapa de recombinação. Diferente do cruzamento, que combina informações genéticas já existentes nos progenitores, a mutação introduz modificações aleatórias em um ou mais genes.

Neste trabalho, o AG implementado emprega o método de **Mutação Uniforme**, definido na Subseção 2.1.1.

2.2.5 Seleção de sobreviventes

O mecanismo de Seleção dos Sobreviventes, também chamado de Substituição da População, é a fase na qual ocorre uma competição entre progenitores e descendentes para decidir quais indivíduos formarão a população da geração seguinte (EIBEN; SMITH, 2015).

Em um AG, utiliza-se com frequência o método **Geracional**, no qual a população da geração seguinte é inteiramente composta pelo conjunto de descendentes (CUNHA; TAKAHASHI; ANTUNES, 2012).

O método de **Elitismo** estende o método geracional ao reservar uma pequena parcela da população seguinte para os melhores progenitores da geração anterior (EIBEN; SMITH, 2015; CUNHA; TAKAHASHI; ANTUNES, 2012). O método de **Estado Estacionário**, ou *steady state*, é uma alternativa em que apenas uma pequena parcela da população é substituída a cada geração, possibilitando que progenitores e descendentes coexistam por várias iterações (EIBEN; SMITH, 2015).

Neste trabalho, o AG implementado utiliza o método de Elitismo como seleção de sobreviventes, estendendo o método geracional anteriormente.

2.3 Estratégia Evolutiva

Desenvolvidas por Rechenberg (1973) na década de 1970, as Estratégias Evolutivas (EEs) constituem uma técnica da Computação Evolucionária que aplica o princípio biológico de mutação e seleção para evoluir soluções para um problema. Nessa técnica, não há recombinação; progenitores são selecionados e descendentes formados a partir dos operadores genéticos de clonagem e mutação. Isso acontece, pois, em contraste com os AGs de Holland (1975), a escola das Estratégias Evolutivas defende a mutação como o mecanismo primário de inovação no espaço de soluções (CUNHA; TAKAHASHI; ANTUNES, 2012).

Na literatura, destacam-se 3 abordagens de EE. São elas: $EE-(1+1)$, $EE-(\mu, \lambda)$ e $EE-(\mu + \lambda)$ (EIBEN; SMITH, 2015; CUNHA; TAKAHASHI; ANTUNES, 2012).

Neste trabalho, adota-se a abordagem denominada $EE-(\mu, \lambda)$. O funcionamento geral dessa abordagem de EE pode ser visualizado no algoritmo apresentado na Figura 2.2 e também descrito pelo passo a passo a seguir:

- (i) A partir de uma boa solução conhecida para o problema, ou de forma aleatória, inicializa-se uma população inicial $P(t)$ com μ indivíduos e avaliam-se seus valores de aptidão.
- (ii) Geram-se λ descendentes dos μ progenitores através de clonagem, seguida de mutação. Para cada um dos λ descendentes, um progenitor é selecionado aleatoriamente do conjunto

de progenitores. De forma específica, cada progenitor tem a probabilidade de $1/\mu$ de ser sorteado para cada descendente gerado.

- (iii) Avaliam-se os descendentes por meio da função de aptidão.
- (iv) Selecionam-se os μ melhores descendentes entre os λ gerados. Forma-se a população da geração seguinte $P(t + 1)$ com o conjunto de μ descendentes selecionados.
- (v) Verifica-se o critério de parada. Caso não tenha sido atingido, o processo retorna para o passo 2.

Algoritmo 2 Esquema Geral de uma Estratégia Evolutiva (μ, λ)

```

1:  $t \leftarrow 0$ 
2: Gera população inicial  $P(t)$  com  $\mu$  indivíduos
3: Avalia  $P(t)$ 
4: repita
5:   Seleciona  $\mu$  progenitores a partir de  $P(t)$ 
6:   Gera  $\lambda$  descendentes por clonagem e mutação
7:   Avalia descendentes
8:    $P(t + 1) \leftarrow$  os  $\mu$  melhores descendentes
9:    $t \leftarrow t + 1$ 
10: até critério de terminação atingido
11: Devolver melhor indivíduo de  $P(t)$ 

```

Figura 2.2: Esquema geral de uma Estratégia Evolutiva (μ, λ) .

Analisando o esquema geral descrito no passo a passo anterior e na Figura 2.2, é possível identificar o mecanismo de **seleção de progenitores** na linha 5 do algoritmo, o operador genético de **mutação** na linha 6 e, ainda, o mecanismo de **seleção de sobreviventes** executado na linha 8. Nas subseções a seguir, são definidos esses conceitos no contexto específico das EEs e a forma com que são empregados neste trabalho.

2.3.1 Seleção de progenitores

A seleção de progenitores consiste em escolher, em uma dada geração, indivíduos da população atual para originar descendentes por clonagem e mutação. Na EE canônica, esse mecanismo

de seleção costuma ser realizado de forma aleatória: cada um dos μ indivíduos da população atual tem probabilidade igual a $1/\mu$ de ser sorteado como progenitor (EIBEN; SMITH, 2015; CUNHA; TAKAHASHI; ANTUNES, 2012).

Neste trabalho, a EE- (μ, λ) implementada utiliza o método definido anteriormente de seleção aleatória de progenitores da EE canônica.

2.3.2 Mutação na Estratégia Evolutiva

Em uma EE, a mutação apresenta-se como o operador principal de exploração do espaço de soluções. Esse operador genético é o responsável por introduzir novos genes na população a cada geração e, dessa forma, mover o conjunto de soluções através do espaço de busca (RECHENBERG, 1973; CUNHA; TAKAHASHI; ANTUNES, 2012).

Neste trabalho, a mutação da EE é realizada pelo método de **Mutação Gaussiana**, definido na Subseção 2.1.1: a cada gene sorteado para mutação, soma-se um incremento extraído de uma distribuição normal com desvio padrão σ . Esse hiperparâmetro foi alvo de experimentos realizados no capítulo de Resultados e Discussões.

2.3.3 Seleção de sobreviventes

O mecanismo de seleção de sobreviventes, também chamado de substituição da população, define quais indivíduos formarão a geração seguinte (CUNHA; TAKAHASHI; ANTUNES, 2012). Na EE- (μ, λ) , é aplicada a seleção de sobreviventes **geracional**, em que todos os progenitores são descartados e os μ melhores descendentes entre os λ avaliados constituem uma nova população (EIBEN; SMITH, 2015).

Esse esquema difere da seleção aplicada pela abordagem $(\mu + \lambda)$, na qual progenitores e descendentes competem entre si para formar a população da geração subsequente (CUNHA; TAKAHASHI; ANTUNES, 2012).

Neste trabalho, a EE implementada realiza a seleção de sobreviventes clássica da EE- (μ, λ) definida anteriormente nessa subseção.

2.4 Redes Neurais Artificiais

Redes Neurais Artificiais (RNA), ou somente redes neurais, é um método bioinspirado pertencente ao campo do Aprendizado de Máquina. Neste método, unidades de processamento, chamadas de neurônios artificiais, são agrupadas, interconectadas e utilizadas para processar, aprender e realizar diversas tarefas (YANNAKAKIS; TOGELIUS, 2018).

Segundo Millington (2006), os neurônios artificiais são chamados desta forma pois inicialmente pretendiam simular o funcionamento de um neurônio biológico.

Quando agrupados em conjuntos, denominados camadas, cada neurônio artificial se comunica com um subconjunto de outros neurônios artificiais através de conexões características do modelo de neurônio adotado e da arquitetura de rede neural empregada (MILLINGTON, 2006). Estas conexões, também chamadas de sinapses, estão associadas a um valor correspondente, chamado peso sináptico.

O modelo de neurônio artificial denominado Perceptron (ROSENBLATT, 1957), por exemplo, recebe um ou mais valores como entrada, processa estes valores e, por fim, emite o resultado do processamento como valor de saída.

Quando um neurônio Perceptron é organizado em uma rede neural artificial de apenas uma camada, este constitui a arquitetura chamada de Rede Perceptron, considerada a mais simples arquitetura de rede neural possível (SILVA; SPATTI; FLAUZINO, 2016).

Na arquitetura chamada de Perceptron Multicamadas (PMC), por sua vez, temos redes neurais artificiais compostas de várias camadas de neurônios. Nesta arquitetura, os neurônios de uma determinada camada recebem, como valores de entrada, os valores de saída de todos os neurônios da camada anterior e o seu limiar de ativação específico. A saída de cada neurônio, conseqüentemente, é emitida à entrada de cada neurônio presente na camada seguinte.

As camadas em uma RNA recebem nomes diferentes de acordo com sua posição na arquitetura. A seguir, definimos os tipos possíveis de camada em uma RNA com a arquitetura PMC:

- **Camada de entrada:** A camada de entrada é responsável por receber os dados proveni-

entes do ambiente externo e disponibilizá-los para processamento pela rede neural. Cada neurônio dessa camada corresponde, em geral, a uma variável de entrada do problema, repassando seus valores diretamente para a camada subsequente sem realizar transformações no valor de entrada. Dessa forma, essa camada atua como interface entre os dados brutos e o modelo computacional, sendo essencial para a correta representação do espaço de entrada (HAYKIN, 2021; SILVA; SPATTI; FLAUZINO, 2016).

- **Camada de saída:** A camada de saída é responsável por produzir o resultado final do processamento realizado pela rede neural. Seus neurônios representam diretamente as respostas do modelo, podendo assumir diferentes interpretações dependendo da aplicação, como valores contínuos em problemas de regressão ou probabilidades associadas a classes em problemas de classificação. A definição da estrutura dessa camada está diretamente relacionada ao tipo de tarefa que a rede se propõe a resolver (RUSSELL; NORVIG, 2020; SILVA; SPATTI; FLAUZINO, 2016) .
- **Camadas ocultas:** As camadas ocultas são aquelas posicionadas entre a camada de entrada e a camada de saída, sendo responsáveis pela extração e transformação progressiva das características dos dados de entrada. Nessas camadas, os neurônios realizam combinações ponderadas das entradas seguidas da aplicação de funções de ativação, permitindo que a rede aprenda representações internas cada vez mais abstratas (HAYKIN, 2021; RUSSELL; NORVIG, 2020).

A organização em camadas define o fluxo de informação em uma rede neural artificial, no qual os dados são propagados da camada de entrada até a camada de saída por meio das sinapses entre os neurônios. Nesse contexto, o processamento realizado por cada neurônio, como o Perceptron, ocorre localmente em cada unidade, sendo responsável por transformar os sinais recebidos da camada anterior e encaminhar o resultado para a camada seguinte (HAYKIN, 2021; SILVA; SPATTI; FLAUZINO, 2016).

Considerando essa dinâmica de propagação entre camadas, o processamento executado por um neurônio Perceptron k pode ser descrito pelo passo a passo a seguir:

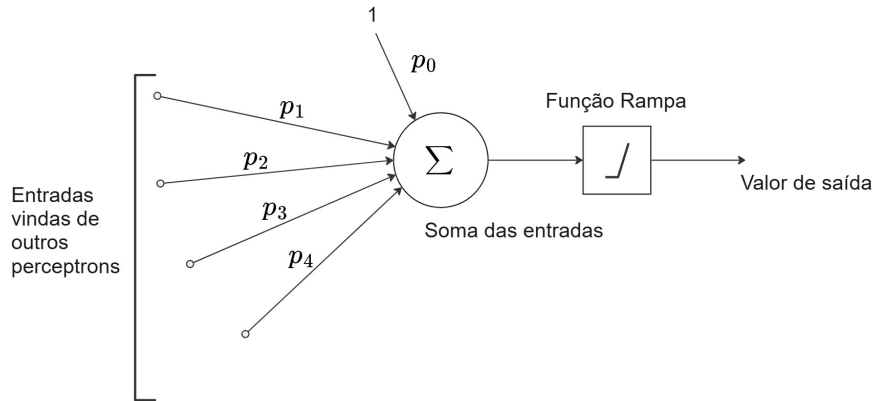


Figura 2.3: Algoritmo do Neurônio Perceptron (adaptado de Millington (2006))

1. Inicialmente, iteramos através do conjunto de entradas x . Neste conjunto, cada entrada x_i tem seu valor multiplicado por seu peso sináptico $p_{k,i}$ correspondente.
2. Em seguida, calculamos o chamado potencial de ativação, que é equivalente à soma da multiplicação de todas as entradas por seus pesos, obtidos no passo anterior, juntamente do valor do limiar de ativação θ_k . Em termos matemáticos, considerando o número de entradas igual a n , o cálculo do potencial de ativação u_k é descrito na equação a seguir.

$$u_k = \theta_k + \sum_{i=1}^n p_{k,i} * x_i$$

3. O potencial de ativação u_k é então submetido a uma função de ativação, como a função degrau ou a função rampa.
4. O resultado da função de ativação é o valor de saída do neurônio.

A figura 2.3 ilustra o processamento descrito anteriormente no contexto de um neurônio pertencente a uma RNA utilizando a arquitetura PMC e a função de rampa como função de ativação.

As subseções a seguir explicam conceitos importantes relacionados ao funcionamento das Redes Neurais Artificiais.

2.4.1 Limiar de ativação

Um parâmetro importante no algoritmo executado pelo neurônio Perceptron é o chamado limiar de ativação, ou viés (do inglês, *bias*). O limiar de ativação de um neurônio é um parâmetro cujo valor é capaz de transladar o domínio da função de ativação utilizada por este neurônio (SILVA; SPATTI; FLAUZINO, 2016). Em outras palavras, o valor do viés tem o efeito de aplicar uma transformação afim à entrada da função de ativação (HAYKIN, 2021).

É interessante notar que o limiar de ativação θ de um neurônio pode ser representado também como o peso sináptico p_0 de uma entrada extra x_0 cujo valor é sempre igual a 1. A Figura 2.3, por exemplo, ilustra um neurônio que possui uma entrada de valor 1 e cujo peso p_0 representa o valor de viés para este neurônio. Para chegar nesta representação, é preciso primeiramente definir o limiar de ativação θ_k de um dado neurônio k segundo a equação a seguir.

$$\theta_k = p_{k,0} * x_0$$

Em Russell e Norvig (2020), o limiar de ativação é representado desta forma com o intuito de substituir a equação anterior na equação que define o potencial de ativação e transformar o cálculo do potencial de ativação u em um simples somatório do produto entre pesos e valores de entrada do neurônio k , como podemos ver abaixo.

$$\begin{aligned} u &= \theta_k + \sum_{i=1}^n p_{k,i} * x_i \\ u &= p_0 * x_0 + \sum_{i=1}^n p_{k,i} * x_i \\ u &= \sum_{i=0}^n p_{k,i} * x_i \end{aligned}$$

2.4.2 Função de Ativação

O neurônio artificial aplica uma regra matemática para determinar qual será o seu sinal de saída. Esta regra é definida pela função de ativação (SILVA; SPATTI; FLAUZINO, 2016). O papel

fundamental desta função é decidir se o sinal processado pelo neurônio é relevante o suficiente para ser propagado para as camadas seguintes da rede (RUSSELL; NORVIG, 2020). Diz-se que um neurônio está ativado quando a saída de sua função de ativação é positiva. Caso a saída seja nula ou negativa, o neurônio é classificado como desativado (SILVA; SPATTI; FLAUZINO, 2016).

A Necessidade da Não Linearidade

As funções de ativação podem ser categorizadas em lineares e não lineares. As funções lineares seguem o formato definido:

$$f(u) = m * u$$

onde m é uma constante. Qualquer função que não siga esse formato é uma função não-linear (RUSSELL; NORVIG, 2020).

Embora sejam úteis em casos específicos, como na camada de saída de modelos de regressão, o uso de funções lineares em uma RNA com a arquitetura PMC apresenta uma limitação matemática severa, pois a composição de múltiplas operações lineares resulta em apenas mais uma transformação linear. Sendo assim, independentemente da quantidade de camadas ocultas que a rede possua, ela se comportaria matematicamente como um modelo de regressão linear simples, incapaz de resolver problemas não triviais (HAYKIN, 2021).

Para que a rede neural consiga aprender representações complexas, criar fronteiras de decisão intrincadas e mapear padrões contextuais, é fundamental a introdução de funções de ativação não-lineares. São estas funções que conferem às redes neurais a capacidade descrita pelo Teorema da Aproximação Universal, permitindo que a arquitetura atue como um aproximador de funções genéricas (RUSSELL; NORVIG, 2020).

Exemplos de Funções de Ativação

Diversas funções de ativação são empregadas na literatura, cada qual com características adequadas a diferentes tipos de problemas ou arquiteturas de rede. A seguir, destacam-se algumas das principais funções:

- **Função Degrau:** A função degrau é uma função cuja imagem pode assumir apenas dois valores: 0 ou 1. Esta função é definida na equação a seguir.

$$f(u) = \begin{cases} 1, & \text{se } u \geq 0 \\ 0, & \text{se } u < 0 \end{cases}$$

É chamada desta forma devido a mudança brusca na imagem da função a partir de um determinado ponto do domínio, denominado ponto de limiar, o que no gráfico assemelha-se visualmente a um degrau físico de escada. Usualmente, este ponto de limiar é definido como zero, embora outros valores possam ser adotados dependendo da modelagem do problema (HAYKIN, 2021). A função degrau é uma das primeiras funções de ativação utilizadas em redes neurais artificiais, sendo empregada no modelo Perceptron original (SILVA; SPATTI; FLAUZINO, 2016).

- **Função Identidade:** Retorna o próprio valor do potencial de ativação sem alterações. A função identidade é definida a seguir.

$$f(u) = u$$

Dado que é uma função linear, a função identidade não é recomendada como função de ativação para camadas ocultas. Seu uso é comumente restrito à camada de saída em tarefas cuja saída desejada para a RNA é um valor contínuo real (SILVA; SPATTI; FLAUZINO, 2016).

- **Função Sigmoid (Logística):** Uma função não-linear que mapeia qualquer valor real de entrada para um intervalo contínuo e suave entre 0 e 1. É amplamente utilizada como função de ativação em redes neurais devido a ser uma função totalmente diferenciável, uma característica importante para o treinamento de uma RNA (HAYKIN, 2021). Um

exemplo de função sigmóide é a função logística definida na equação abaixo.

$$f(u) = \frac{1}{1 + e^{-u}}$$

onde e representa o número de Euler.

- **Unidade Linear Retificada (ReLU - *Rectified Linear Unit*):** Também chamada de função rampa, é a função de ativação padrão para as camadas ocultas na grande maioria das arquiteturas modernas que utilizam múltiplas camadas. A ReLU introduz não linearidade na rede ao zerar os valores negativos, enquanto mantém os valores positivos inalterados. Um dos motivos que justifica sua adoção generalizada é sua simplicidade computacional: o cálculo da função ReLU envolve apenas uma operação de comparação (RUSSELL; NORVIG, 2020). A função ReLU é definida na equação definida a seguir.

$$f(u) = \max(0, u)$$

2.4.3 Algoritmo de aprendizado

O aprendizado de uma RNA é definido como o processo pelo qual os parâmetros livres de uma rede (pesos sinápticos e limiar de ativação de um neurônio) são ajustados através da análise dos estímulos do ambiente no qual a rede está inserida. Não há um algoritmo de aprendizagem único para o projeto de uma RNA. Em vez de um algoritmo universal, a literatura oferece um verdadeiro arsenal de algoritmos de aprendizagem, cada qual com vantagens específicas em diferentes cenários (HAYKIN, 2021).

Parâmetros do aprendizado

O sucesso do treinamento de uma rede depende da configuração de certos parâmetros que controlam a dinâmica do ajuste de pesos sinápticos. Dentre os principais, destacam-se:

- **Taxa de Aprendizado (η):** É o parâmetro que determina a magnitude das alterações feitas nos pesos a cada iteração de ajuste. Uma taxa de aprendizado muito elevada

pode acelerar o treinamento, mas também pode impedir a convergência à valores que representam a solução ótima do ajuste. Por outro lado, uma taxa de aprendizado muito reduzida torna o processo bastante lento e suscetível a ficar retido em ótimos locais, de acordo com Silva, Spatti e Flauzino (2016).

- **Épocas:** Uma época representa um ciclo completo em que um conjunto de dados é processado e utilizado para ajustar os parâmetros de uma rede. Este é um conceito importante na abordagem do aprendizado preditivo, pois um grande número de épocas de treinamento pode ocasionar o problema de superajuste da rede, também chamado de *overfitting* (SILVA; SPATTI; FLAUZINO, 2016).
- **Inicialização dos pesos sinápticos e limiares:** A inicialização dos pesos sinápticos e do limiar de ativação de cada neurônio é uma tarefa crucial. É recomendado que os pesos iniciem com valores aleatórios pequenos, garantindo que neurônios comecem a aprender diferentes características do problema desde o início do processo, segundo Haykin (2021).

Neste trabalho, os parâmetros de aprendizado empregados são a taxa de aprendizado e inicialização dos pesos sinápticos e limiares.

Exemplos de Algoritmo de Aprendizado

Em arquiteturas PMC que utilizem o paradigma do Aprendizado Preditivo, o algoritmo de Retropropagação do erro, ou *Backpropagation*, estabeleceu-se como o mais popular dos algoritmos de aprendizagem (HAYKIN, 2021).

No *backpropagation*, o ajuste dos pesos acontece da seguinte forma: a cada entrada processada, calcula-se a diferença entre a saída obtida e a desejada. Esta diferença, também chamada de erro, é então propagada de volta pela rede e utilizada como o principal critério de ajuste dos pesos (HAYKIN, 2021).

Por outro lado, há cenários em que a saída desejada não é conhecida para todo conjunto de entradas possíveis de um problema, o que impede o cálculo do erro na saída e torna inviável a aplicação do *backpropagation*. Nestes cenários, o uso de Algoritmos Evolutivos como

algoritmo de aprendizado, a chamada Neuroevolução, surge como uma alternativa promissora (YANNAKAKIS; TOGELIUS, 2018).

Neste trabalho, utilizamos um Algoritmo Genético e uma Estratégia Evolutiva, ambos pertencentes à classe dos Algoritmos Evolutivos, como algoritmos de aprendizado em uma rede PMC.

2.5 Neuroevolução

A Neuroevolução (NE) consiste na aplicação de Algoritmos Evolutivos para o treino ou projeto de arquiteturas de Redes Neurais Artificiais (YANNAKAKIS; TOGELIUS, 2018). Esta abordagem surge como uma alternativa robusta aos métodos baseados em gradiente, como o *backpropagation*, para problemas onde não é possível calcular um sinal de erro para cada valor de entrada processado pela rede (MICHALEWICZ, 1994; RISI; TOGELIUS, 2015).

Segundo Yannakakis e Togelius (2018), existem normalmente dois tipos de abordagens em NE. No primeiro tipo, são ajustados os pesos sinápticos de uma RNA e cada peso sináptico é codificado na forma de um gene do AE (LUCAS, 2005). Esta forma de codificar os parâmetros da RNA é também chamada de **codificação direta**. A segunda abordagem consiste em ajustar não mais apenas os pesos sinápticos, mas também a topologia da RNA. Nesta abordagem, apesar de possível o uso da codificação direta, é mais comum a opção por **codificação indireta**, onde um gene do AE pode ser traduzido em mais de um dos elementos ajustados na RNA (RISI; TOGELIUS, 2015).

2.5.1 Neuroevolução em jogos

O domínio dos jogos digitais é atualmente reconhecido como um dos ambientes mais férteis e desafiantes para a aplicação da neuroevolução (RISI; TOGELIUS, 2015). Isto deve-se ao fato de um agente num jogo raramente poder ser avaliado através de uma única decisão isolada, dependendo, em vez disso, de uma sequência complexa de ações tomadas ao longo do tempo para atingir um objetivo, como completar um nível (MILLINGTON, 2006).

Trabalhos recentes (CHEN, 2022; PEREIRA, 2023), demonstram a viabilidade empírica destas técnicas. Diferente do aprendizado preditivo (ou supervisionado) tradicional, a neuro-evolução em jogos utiliza a aptidão (*fitness*) acumulada no final de uma partida para guiar a otimização. Assim, o algoritmo evolutivo avalia o comportamento global gerado por diferentes configurações de pesos sinápticos e seleciona as estratégias mais eficientes, revelando-se uma ferramenta poderosa para lidar com ambientes dinâmicos e contínuos (RISI; TOGELIUS, 2015; YANNAKAKIS; TOGELIUS, 2018).

Neste trabalho, a NE empregada ajusta os pesos sinápticos de uma RNA responsável pelo controle de um personagem do jogo *Chrome Dino*. Quanto à forma de representar estes pesos em genes, opta-se pela codificação direta.

2.6 O Jogo *Chrome Dino*

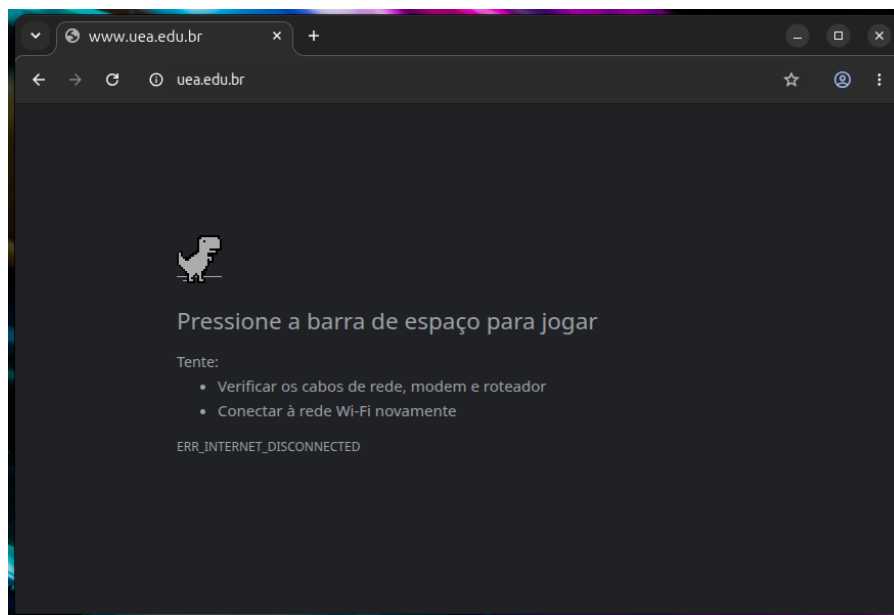
O *Chrome Dino* é um jogo digital desenvolvido pela Google e embutido no navegador *Google Chrome* (Google, 2008). Lançado em 2014, o jogo é acessível por meio do navegador e sem a necessidade de conexão com a internet. Mais especificamente, ao tentar acessar uma página da *web* no navegador *Google Chrome* sem conexão com a internet, o navegador exibe uma tela de erro interativa. Nessa tela, além de instruções úteis para solucionar o problema de falta de conexão, é exibida também a tela inicial do jogo. A Figura 2.4 apresenta uma captura de tela exibindo a tela de início do jogo.

No *Chrome Dino*, o jogador controla um personagem, representado por um dinossauro da espécie *Tyrannosaurus rex*, que avança automaticamente por um cenário desértico. O objetivo do jogo é percorrer a maior distância possível, desviando-se dos obstáculos que surgem ao longo do percurso. À medida que o jogo avança, a velocidade de deslocamento do cenário é aumentada, o que eleva a dificuldade e exige reações cada vez mais rápidas do jogador.

Para desviar dos obstáculos, os jogadores podem executar duas ações: saltar e abaixar. Essas ações são executadas através de teclas do teclado: a tecla Espaço para saltar e a tecla Seta para baixo para abaixar. A colisão do dinossauro com qualquer obstáculo encerra a partida.

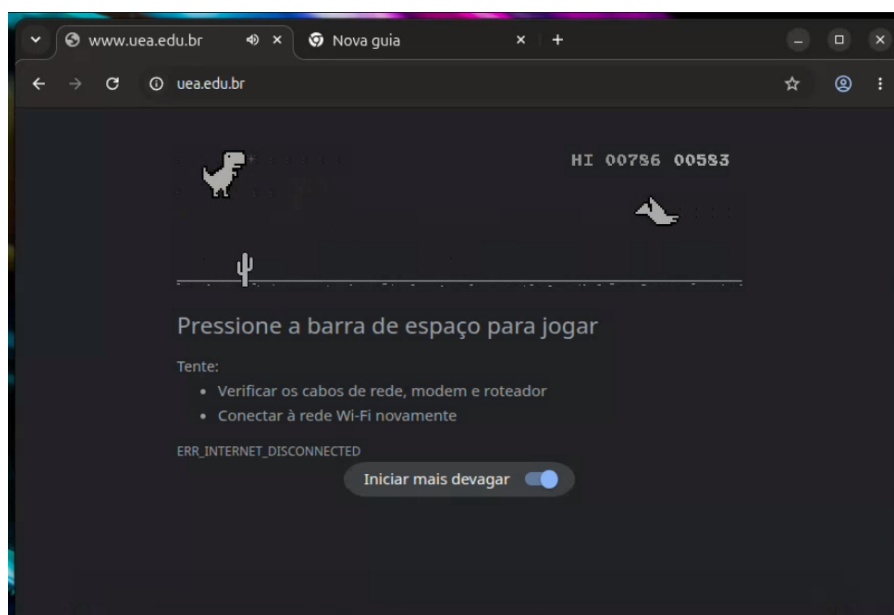
Os obstáculos do jogo são cactos e pássaros (pterodáctilos), como é possível ver na Figura

Figura 2.4: Captura de tela exibindo a tela de início do jogo *Chrome Dino* em uma janela do navegador *Google Chrome*.



2.5. Os cactos são posicionados no chão e devem ser ultrapassados por meio de um salto. Os pássaros, por sua vez, surgem em diferentes alturas e podem exigir tanto o salto quanto a ação de abaixar do personagem para serem evitados.

Figura 2.5: Captura de tela exibindo a tela do jogo *Chrome Dino* durante uma jogatina em uma janela do navegador *Google Chrome*.

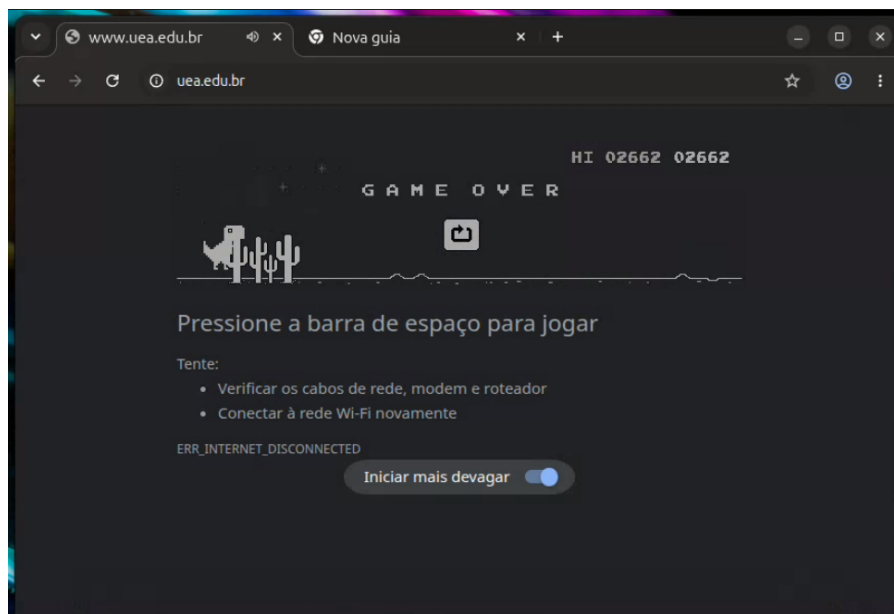


O jogo conta também como uma mecânica de gravidade, que atua sobre o dinossauro *Rex*

sempre que esse pula e faz com que ele retorne ao solo. É possível controlar a velocidade com a qual o retorno do dinossauro ao solo acontece, o que permite aos jogadores realizar saltos mais longos ou mais curtos. Ao segurar a tecla Espaço durante o salto, por exemplo, a velocidade de retorno ao solo é reduzida e, conseqüentemente, o salto do dinossauro perdura por mais tempo. Por sua vez, ao pressionar a tecla Seta para baixo durante um salto, a velocidade de retorno ao solo é aumentada e, conseqüentemente, o dinossauro retorna ao solo mais rapidamente.

A pontuação do jogo é determinada pela distância percorrida pelo personagem até o momento da colisão. A Figura 2.6 apresenta uma captura de tela exibindo a tela de fim de jogo.

Figura 2.6: Captura de tela exibindo a tela de fim do jogo *Chrome Dino* em uma janela do navegador *Google Chrome*.



Neste trabalho, o jogo Chrome Dino foi adotado como estudo de caso para aplicação de técnicas de Neuroevolução com Redes Neurais Artificiais e dois algoritmos do campo da Computação Evolucionária.

2.7 Trabalhos Relacionados

Nesta seção, são analisados artigos e outras obras relacionadas com os temas desenvolvidos neste trabalho. De forma específica, foram selecionados cinco trabalhos (e uma base de código)

que aplicam métodos de Neuroevolução em Jogos com o objetivo principal de evoluir agentes inteligentes para jogar um jogo. Nas subseções a seguir, cada obra é analisada em particular.

2.7.1 Avaliação da eficácia das Estratégias Evolutivas no treinamento de Redes Neurais Perceptron aplicadas a Jogos Digitais

O trabalho de Pereira (2023) aborda o desenvolvimento de Estratégias Evolutivas para a Neuroevolução de Redes Neurais do tipo Perceptron Multicamadas. Para tanto, o autor implementou 4 diferentes jogos que serviram de ambiente de treinamento para as redes neurais. Os jogos implementados foram: *Flappy Bird*, Dinossauro do Google Chrome, Simulador de Foguete e Simulador de Corrida.

Neste trabalho, o autor optou por evoluir somente os pesos sinápticos das redes neurais utilizadas, deixando fixados outros aspectos da arquitetura das redes, como por exemplo o número de camadas ocultas que foi definido como apenas uma.

Ademais, o autor escolheu uma abordagem empírica quanto às entradas fornecidas às redes evoluídas: para cada jogo, foi definido um conjunto de variáveis que representassem um momento no tempo do jogo. Essa abordagem possibilitou a redução da complexidade das redes neurais evoluídas quando comparadas à abordagem alternativa, que consistia em fornecer capturas de telas para representar cada momento no tempo dos jogos.

Adicionalmente, o cálculo da aptidão de indivíduos foi realizado a partir da média de pontuação de múltiplas partidas. Isto foi feito como forma de mitigar a aleatoriedade na configuração inicial das partidas dos jogos e conferir robustez às aptidões calculadas.

O autor realizou ainda experimentos entre dois tipos de método de seleção, dois tipos de método de mutação (incluindo uma proposta própria denominada método de Mutação Parcial onde genes são sorteados para serem mutados) e um experimento quanto ao uso do mecanismo de autoadaptação de Rechenberg (1973).

Os resultados dos experimentos realizados demonstraram que as redes neurais alcançaram uma acurácia média em torno de 94%, o que confirma a eficácia do uso das Estratégias Evolutivas na Neuroevolução de agentes para os jogos analisados. A investigação de Pereira, no

entanto, restringiu-se a utilizar apenas Estratégias Evolutivas, deixando em aberto o espaço para explorar outros algoritmos evolutivos para Neuroevolução em jogos, como os Algoritmos Genéticos.

2.7.2 *Learning Dark Souls With Neuroevolution*

No trabalho “*Learning Dark Souls Combat Through Pixel Input With Neuroevolution*”, O’Connor, Parker e Bugti (2025) abordam o treinamento de agentes inteligentes utilizando neuroevolução no desafiador jogo de ação *Dark Souls* (FromSoftware, 2018). O objetivo central dos autores nesse estudo foi investigar a capacidade de uma rede neural aprender a derrotar o primeiro chefe do jogo, chamado *Asylum Demon*.

Para tanto, os autores escolheram o algoritmo evolutivo NEAT (*Neuroevolution of Augmenting Topologies*) (STANLEY; MIIKKULAINEN, 2002) para comandar o processo de evolução. O algoritmo NEAT se destaca por evoluir, simultaneamente, os pesos sinápticos e a topologia das redes neurais empregadas.

Para contornar a falta de uma interface de programação (API) oficial para o jogo, pela qual seria possível consultar dados sobre a jogatina e executar ações de forma programada com o personagem do jogador, os autores optaram por desenvolver sua própria interface de programação, chamada de DSAPI (*Dark Souls API*). Essa interface utiliza visão computacional para processar dados do jogo e foi utilizada para extrair as duas métricas necessárias para o cálculo da aptidão dos indivíduos evoluídos: as porcentagens de vida do jogador e do chefe inicial. A DSAPI foi utilizada ainda para controlar o personagem e executar as ações calculadas pelas redes neurais.

Nesse estudo, o conjunto de dados de entrada fornecido às redes neurais foi composto de capturas de tela do jogo redimensionadas para a resolução 40x30 pixels. Como saída, as redes neurais retornavam a distribuição probabilística de um conjunto composto por 14 ações possíveis de se executar, como rolar, atacar, curar, entre outras ações. A maior probabilidade dentro do conjunto era aquela que era enviada à interface DSAPI para execução.

Os resultados quantitativos obtidos revelaram que os agentes desenvolvidos atingiram uma

taxa máxima de 35% de vitória contra o chefe inicial do jogo em mais de uma das 50 gerações processadas. Esses achados comprovam a viabilidade da Neuroevolução com o algoritmo evolutivo NEAT no treinamento de redes neurais em cenários 3D desprovidos de representações explícitas sobre os estados do jogo. No entanto, o estudo não realiza experimentos com outros algoritmos evolutivos ou mesmo outros hiperparâmetros, limitando-se apenas à avaliação do desempenho do algoritmo NEAT em uma única configuração de valores para seus hiperparâmetros, e.g. taxa de mutação e taxa de cruzamento.

2.7.3 Comparação de Algoritmos no aprendizado do jogo 2048

No trabalho “*Komparasi Algoritma WOA, MFO dan Genetic pada Optimasi Evolutionary Neural Network dalam Menyelesaikan Permainan 2048*”, Armanto, Setiabudi e Pickerling (2021) abordam a aplicação de algoritmos de otimização para desenvolver agentes inteligentes capazes de jogar o jogo de quebra-cabeça “2048”. O objetivo central da pesquisa consistiu em comparar o desempenho de três métodos de otimização diferentes: O Algoritmo Genético canônico, o WOA (*Whale Optimization Algorithm*) e o MFO (*Moth Flame Optimization*). Esses algoritmos foram utilizados no treinamento de duas técnicas distintas de Inteligência Artificial: Redes Neurais Artificiais e Árvores de Decisão.

Nesse estudo, o treinamento das IAs aconteceu em 3 variações do tabuleiro do jogo 2048: 4x4 (o tabuleiro padrão), 5x5 e 6x6. A arquitetura empregada pela RNA evoluída foi composta de uma camada de entrada de 6 neurônios, uma camada oculta com 10 neurônios e uma camada de saída com 2 neurônios de saída binária. Os neurônios da camada de entrada recebem o conjunto de variáveis que representam o estado do jogo em um dado momento. Esse conjunto é definido por 6 variáveis, compostas pelo valor da maior peça no tabuleiro, um indicativo (inteiro booleano) se a peça de maior valor está em um dos cantos do tabuleiro, a quantidade de peças no tabuleiro cujo valor é menor que 32, entre outras variáveis.

Uma vez que a saída da rede neural indica a ação a executar dentro do jogo em um dado momento, as 4 possíveis ações de se executar no jogo, i.e., os movimentos de esquerda, cima, direita e abaixo, são representadas pela tuplas formadas pela saída binária dos dois neurônios

presentes na camada de saída. Mais especificamente, as possíveis ações a se executar no jogo descritas anteriormente são, respectivamente, representadas pelas saídas “00”, “01”, “10” e “11” da rede neural. Durante o treinamento das RNAs, a aptidão de cada indivíduo evoluído foi calculada a partir do valor da maior peça alcançada e a pontuação final da partida.

Os resultados obtidos pelo estudo revelam que a arquitetura de RNA teve um desempenho consistentemente superior à arquitetura de Árvores de Decisão utilizada. Na comparação entre os métodos de otimização, por sua vez, tanto o WOA como o MFO demonstraram maior eficácia no treinamento de agentes para o jogo 2048, alcançando pontuações superiores às aquelas obtidas pelo Algoritmo Genético.

O estudo de Armanto, Setiabudi e Pickerling (2021) apresenta uma valiosa comparação entre diferentes algoritmos, comprovando a eficácia no uso de Algoritmos Genéticos e dos demais métodos de otimização testados no aprendizado do jogo digital “2048”. No entanto, o trabalho não se aprofunda em experimentos relacionados a diferentes hiperparâmetros específicos de cada algoritmo, uma lacuna que poderia alterar consideravelmente o desfecho da comparação, visto que o desempenho observado dentre os algoritmos é muito similar.

2.7.4 Neuroevolução Aplicada na Conclusão de Jogos de Plataforma

No estudo “Neuroevolução Aplicada na Conclusão de Jogos de Plataforma” de Chen (2022), o autor desenvolve técnicas de Neuroevolução para automatizar a conclusão da primeira fase do clássico jogo de plataforma *Super Mario Bros*. O objetivo deste estudo foi propor uma ferramenta capaz de evoluir um agente inteligente para jogar o jogo e, ao concluir a fase, extrair métricas sobre a partida, como a quantidade de botões pressionados, entre outros parâmetros. Estas métricas auxiliam profissionais de saúde a classificar quantitativamente o nível de dificuldade do jogo e possibilita-os decidir o quão adequada a fase jogada se mostra ao tratamento cognitivo e motor de pacientes.

Nesse estudo, o algoritmo evolutivo escolhido para dirigir a evolução do agente foi o NEAT, de Stanley e Miikkulainen (2002). O autor utilizou também a ferramenta de emulação *Bizhawk* (TASVIDEOS, 2017) para executar o jogo, processar os quadros e controlar o personagem. A

Rede neural empregada neste trabalho, por sua vez, é constituída de uma camada de entrada com 170 neurônios, sendo 169 oriundos de uma matriz de 13x13 pixels processada para representar o estado do jogo a cada momento e um neurônio representante do viés. A camada de saída é composta de 6 neurônios, cada um representando uma das seguintes ações dentro do jogo: andar nas 4 direções cardeais, correr e pular.

O estudo aponta que o algoritmo evolutivo conseguiu, após um período de treinamento de 70 horas, criar uma rede neural apta a percorrer o mapa e completar o nível inicial do jogo de forma inteiramente autônoma. A Figura 2.7 exhibe uma ilustração representando a topologia encontrada para a rede neural do agente inteligente. A Figura 2.8, por sua vez, apresenta as métricas extraídas da partida e os valores encontrados para a partida concluída pelo agente.



Figura 2.7: Ilustração da rede neural do agente inteligente criado. Fonte: Chen (2022).

Informação	Quantidade
Número de botões	408
Tempo para concluir	23
Número de inimigos por segundo	105
Dificuldade	268.1

Figura 2.8: Métricas encontradas para a partida concluída de forma autônoma. Fonte: Chen (2022).

Os resultados do autor confirmam a viabilidade do uso de Neuroevolução com o algoritmo NEAT para o aprendizado e controle de personagens em jogos digitais.

2.7.5 Aprendizado Evolucionário de Máquina em Jogos

O trabalho “*Evolutionary Machine Learning in Games*” (TOGELIUS et al., 2024) apresenta uma revisão de estudos pertencentes à área do Aprendizado Evolucionário de Máquina (AEM), do inglês *Evolutionary Machine Learning*, no ambiente de jogos. O AEM é uma área da Computação situada na interseção das áreas de Computação Evolucionária e do Aprendizado de Máquina e que engloba o campo da Neuroevolução. O objetivo central desta obra consiste em mapear o estado da arte e documentar as formas com que técnicas de CE e de AM podem ser combinadas.

Nesta revisão, os autores analisam diversas aplicações de Neuroevolução em jogos, não se atendo somente à criação de agentes para jogar de forma autônoma um jogo digital, mas cobrindo também aplicações onde algoritmos evolutivos originam redes neurais responsáveis por gerar novos níveis, regras e texturas em jogos. Outra aplicação discutida no trabalho de Togelius et al. (2024) é a modelagem de jogadores, em que NE é utilizada para mimetizar diferentes estilos de jogo através da análise de dados coletados de jogadores. Além de possibilitar a conclusão de níveis inéditos, a criação de agentes com a habilidade de mimetizar um estilo de jogo possibilita também a navegação de níveis e a reprodução, com boa fidelidade, de táticas e da linha de pensamento do jogador utilizado como inspiração.

A revisão analisada reafirma a utilidade do campo de Neuroevolução na tarefa de criar robôs inteligentes e autônomos para jogar jogos. Além disso, embasa o emprego da NE na criação de agentes com diversas outras habilidades úteis no desenvolvimento de jogos.

2.7.6 Base de código *Dinossauro-Google* para Neuroevolução

A base de código de Neuroevolução *Dinossauro-Google* (DIAS, 2019) é um repositório de código aberto disponível no site *Github*. Essa base de código, desenvolvida na linguagem *C++*, apresenta uma implementação do jogo *Chrome Dino*, juntamente de classes de Rede Neural Artificial e Estratégia Evolutiva responsáveis pela Neuroevolução de agentes inteligentes para aprender o jogo.

Ao executar a base, uma interface de usuário é carregada e exibe os agentes inteligentes, representados na forma de dinossauros, jogando o jogo. Essa interface exibe também diversas outras informações sobre o processo de NE desses agentes, a citar o número de gerações evoluídas, a quantidade de dinossauros vivos, a arquitetura da rede neural que comanda o dinossauro de melhor aptidão, bem como um gráfico contendo os valores da melhor aptidão e da aptidão média através das gerações. A Figura 2.9 apresenta essa interface.

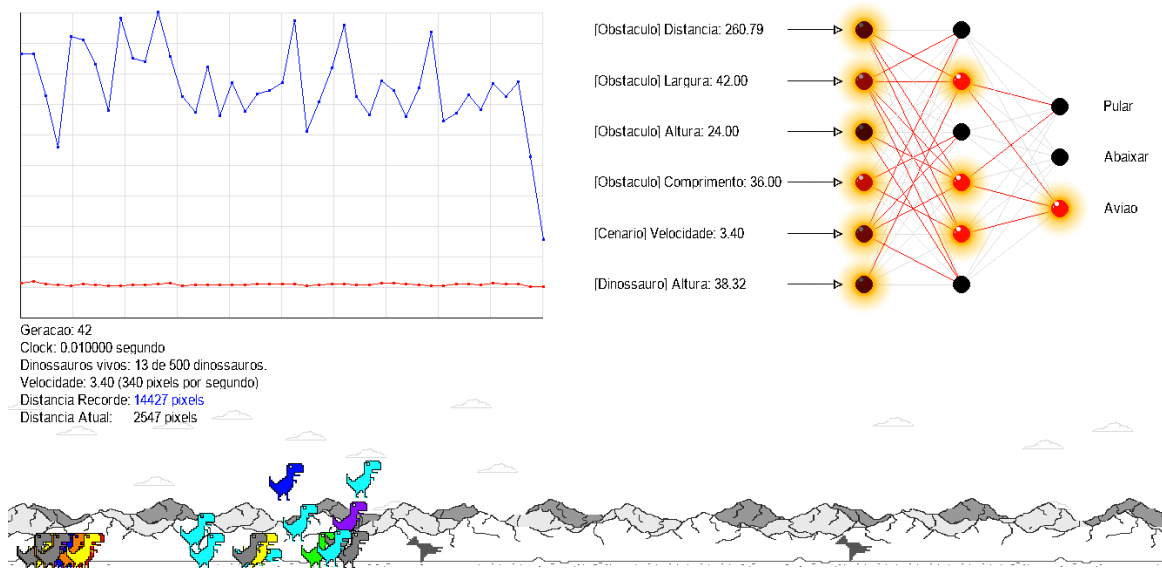


Figura 2.9: Interface de usuário exibida ao executar a base de código *Dinossauro-Google*.

Nessa base de código, a cada período definido do jogo, é fornecido um conjunto de dados de entrada aos modelos de RNA dos agentes inteligentes. Os estados do jogo que compõem esse conjunto de entrada são descritos a seguir:

- a distância entre o agente inteligente e o próximo obstáculo;
- a largura do próximo obstáculo;
- a altura em que o obstáculo está posicionado;
- o comprimento do próximo obstáculo;
- o módulo da velocidade com que o cenário avança em direção ao agente; e
- a altura em que o dinossauro se encontra.

Além disso, os demais parâmetros do jogo e hiperparâmetros da estratégia evolutiva, da rede neural artificial e da neuroevolução estão definidos da seguinte forma:

- Para o jogo:
 - Sequência dos obstáculos: predefinida (não-aleatória).
 - Quantidade de ações dentro do jogo: 3 ações.
 - Período de processamento dos estados: 5 milissegundos.
- Para a rede neural:
 - Tipo de arquitetura: Perceptron Multicamadas.
 - Arquitetura: 1 camada de entrada com 6 neurônios, 1 camada oculta com 6 neurônios e 1 camada de saída com 3 neurônios.
- Para a estratégia evolutiva:
 - Tamanho da população: 2000.
 - Operador genético de mutação: sorteio aleatório entre mutações uniforme, por soma e por multiplicação.
 - Seleção de descendentes: (μ, λ) .

É interessante citar também que a base adiciona um novo obstáculo, chamado de “espinho”, e uma nova ação, denominada “Avião”, à implementação do jogo. Por consequência, uma vez que a quantidade de neurônios na camada de saída está diretamente ligada à quantidade de possíveis ações dentro do jogo, a camada de saída da RNA definida pela base é composta de 3 neurônios.

Por se tratar de uma base de código aberto onde o jogo e as classes de RNA e EE são desenvolvidos de forma integral, diversos valores de parâmetros definidos pela base estão sujeitos a modificação, como, por exemplo, o conjunto de dados de entrada das RNA descrito anteriormente e a ordem dos obstáculos no jogo. É possível também modificar quaisquer outros valores de hiperparâmetros utilizados pela neuroevolução, como o método de avaliação da aptidão dos

agentes, ou ainda estender a base com novos métodos de Computação Evolucionária. Devido a essa capacidade de fácil adaptação e extensão, a base de código *Dinossauro-Google* mostra-se bastante adequada para o uso no desenvolvimento deste trabalho.

2.7.7 Conclusão

Apesar de existirem, na literatura, diversos trabalhos que empregam Neuroevolução para o treinamento de agentes inteligentes em jogos digitais, observa-se que a maior parte deles concentra-se em um único algoritmo evolutivo ou em configurações fixas de hiperparâmetros, sem a experimentação de configurações diferentes para os operadores genéticos da técnica de CE utilizada.

O trabalho de Pereira (2023), por exemplo, investiga a eficácia das Estratégias Evolutivas na Neuroevolução de Perceptrons Multicamadas em quatro jogos, entre eles o *Chrome Dino*. Esse trabalho modela entradas de forma empírica e opta por um método de avaliação que calcula a aptidão de indivíduo por meio da média de múltiplas partidas. Entretanto, o estudo restringe-se às Estratégias Evolutivas, deixando em aberto a comparação direta com outros algoritmos evolutivos, como os Algoritmos Genéticos, nos jogos ambientados.

No trabalho de Armanto, Setiabudi e Pickerling (2021), por sua vez, realiza-se uma comparação entre um Algoritmo Genético e dois outros algoritmos de CE, denominados *Whale Optimization Algorithm* e *Moth Flame Optimization*, no treinamento de redes neurais para o jogo 2048. Contudo, não se explora a variação dos valores para os hiperparâmetro de operadores genéticos dos algoritmos comparados.

Estudos como os de O'Connor, Parker e Bugti (2025) e Chen (2022) empregam o NEAT, evoluindo simultaneamente pesos e topologia da rede, mas limitam-se a uma única combinação de configuração evolutiva e conjunto de hiperparâmetros. Na revisão de Togelius et al. (2024), são discutidas várias aplicações de Neuroevolução em jogos. No entanto, nessa revisão literária, não são apresentados comparativos quantitativos entre algoritmos evolucionários distintos.

Este trabalho busca reduzir a lacuna observada na literatura revisada, oferecendo evidências quantitativas sobre qual combinação de algoritmo evolucionário e valor de hiperparâmetro de mutação, dentre as combinações experimentadas, mostra-se mais promissora para a Neuroevo-

lução de agentes no *Chrome Dino*.

Por fim, a Tabela 2.1 apresenta um comparativo baseado nos diferenciais estabelecidos entre este trabalho e os demais estudos analisados nessa seção de Trabalhos Relacionados.

Tabela 2.1: Comparação entre trabalhos relacionados e o presente estudo

Trabalho	Jogo	AE utilizado	Compara algoritmos de CE?	Varia hiperparâmetros de operador genético?
Pereira (2023)	<i>Chrome Dino</i> , <i>Flappy Bird</i> e outros 2	Estratégia Evolutiva	Não	Sim
O'Connor, Parker e Bugti (2025)	<i>Dark Souls</i>	NEAT	Não	Não
Armanto, Setiabudi e Pickerling (2021)	2048	AG, WOA e MFO	Sim	Não
Chen (2022)	<i>Super Mario Bros</i>	NEAT	Não	Não
Togelius et al. (2024)	Diversos	Diversos (revisão)	Não	Não
Este trabalho	<i>Chrome Dino</i>	AG com Elitismo e EE (μ, λ)	Sim	Sim

3. Desenvolvimento

A partir da metodologia definida, as seções a seguir apresentam o detalhamento das etapas definidas para a condução desse trabalho.

3.1 Definição do Modelo e Configuração Evolutiva

Para essa etapa, primeiramente, se fez necessária a escolha da base de código de Neuroevolução para o desenvolvimento do trabalho. Com a base escolhida, em seguida, definiram-se o tipo de neurônio que compõe as redes treinadas, os hiperparâmetros e a arquitetura do modelo de RNA, bem como a função de ativação dos neurônios. Por fim, para a Configuração Evolutiva, houve a definição dos hiperparâmetros essenciais para implementação dos algoritmos evolutivos e seus operadores genéticos.

As subseções a seguir descrevem com detalhes as atividades desempenhadas nessa etapa.

Escolha da base de código aberto de Neuroevolução

Durante os estudos exploratórios, foi identificada apenas uma base de código aberto de Neuroevolução: a base *Dinossauro-Google* (DIAS, 2019). Dessa forma, essa base foi escolhida para adaptação, extensão e desenvolvimento do trabalho.

Determinação do Tipo de Neurônio e Função de ativação

O tipo de neurônio definido para compor as camadas ocultas e de saída das redes neurais desse trabalho é o neurônio Perceptron de Rosenblatt (1957), capaz de resolver problemas linear-

mente separáveis. A função de ativação utilizada pelos neurônios é a função rampa, ou ReLU, devido ao seu comportamento de anular potenciais de ativação negativos e à sua simplicidade computacional, o que a torna a escolha padrão para camadas ocultas em arquiteturas modernas (RUSSELL; NORVIG, 2020).

Definição da Arquitetura de RNA

A arquitetura definida para o modelo de RNA é a Perceptron Multicamadas constituída de 1 camada de entrada, 1 camada oculta e 1 camada de saída.

Optou-se pela configuração de apenas 1 camada oculta nessa arquitetura, pois é a configuração mínima para solucionar problemas cujo espaço de busca é convexo e não-linearmente separável (SILVA; SPATTI; FLAUZINO, 2016). Além disso, essa é a configuração padrão encontrada na base de Neuroevolução utilizada e, com a arquitetura definida dessa forma, não foram necessárias alterações no código da base *Dinossauro-Google* para possibilitar o uso de um número maior de camadas ocultas.

Para a camada de entrada, definiu-se um conjunto de 6 neurônios correspondentes aos 6 estados do jogo fornecidos por padrão pela base de código às redes neurais treinadas. Na camada oculta, foi escolhido um conjunto de 13 neurônios a partir do método de Kolmogorov para especificação topológica, onde, para uma camada de entrada com n neurônios, a regra $2n + 1$ indica o número mínimo de neurônios ocultos capaz de aproximar funções contínuas em domínios limitados (SILVA; SPATTI; FLAUZINO, 2016). Com $n = 6$, obtém-se 13 neurônios ocultos.

A camada de saída, por sua vez, é composta de apenas 2 neurônios, cada um responsável por uma possível ação dentro do jogo. É interessante dizer aqui que, na implementação do jogo encontrada na base de código adotada, definiam-se 3 ações possíveis aos dinossauros, como explicado na seção 2.7.6. No entanto, uma vez que a ação “Avião” foi uma adição exclusiva da base ao jogo, isto é, não fazia parte do conjunto de ações do jogo verdadeiro, essa ação foi removida. Com a remoção, o conjunto de ações dos dinossauros passou para 2 e, como consequência, definiu-se que a camada de saída seria composta por 2 neurônios.

A Figura 3.1 ilustra como a arquitetura escolhida para o modelo de RNA é exibida durante o processo de neuroevolução.

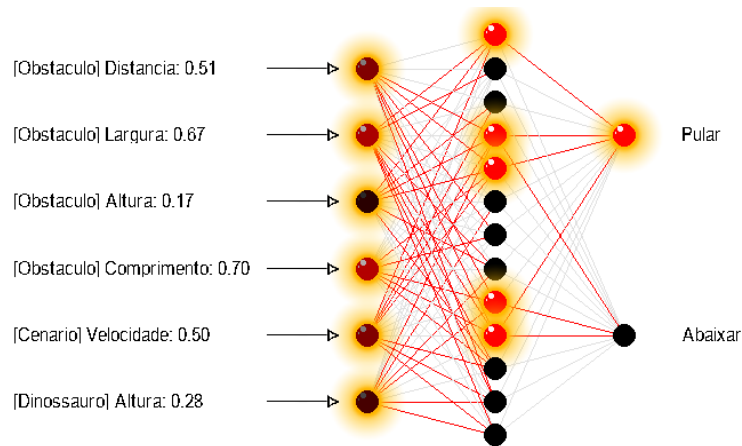


Figura 3.1: Captura de tela exibindo a arquitetura escolhida para o modelo de RNA

Definição de Hiperparâmetros e Operadores Genéticos dos AEs

Nesse trabalho, os algoritmos evolucionários implementados foram o Algoritmo Genético de Holland (1975) com Elitismo e a Estratégia Evolutiva de Rechenberg (1973). Para cada AE, foram utilizados os seguintes operadores genéticos e métodos de seleção de progenitores e de sobreviventes:

- Para o algoritmo genético:
 - Seleção de progenitores: Seleção por roleta;
 - Pontos de cruzamento: 1.
 - Método de mutação: Mutação Uniforme;
 - Seleção de sobreviventes: Geracional com Elitismo.
- Para a estratégia evolutiva:
 - Inicialização da população: Inicialização aleatória e uniforme;
 - Seleção de progenitores: Seleção aleatória e uniforme;

- Método de mutação: Mutação Gaussiana;
- Seleção de sobreviventes: Seleção (μ, λ) ;

Foram definidos, ainda, valores para os hiperparâmetros dos AEs. Além da definição de valores para os hiperparâmetros de uso comum por ambos os AEs desenvolvidos, foi necessária a definição de valores para hiperparâmetros específicos de cada um. Para os hiperparâmetros de uso comum, os valores definidos são apresentados a seguir:

- Tamanho do conjunto de genes: 119 genes;
- Representação Cromossômica: Representação Real;
- Domínio dos genes: $[-1, 1]$;
- Método de Avaliação: Distância média percorrida durante as partidas de uma geração; e
- Condição de Parada: As condições de parada utilizadas variam de acordo com o ciclo de experimentos do capítulo de Resultados e Discussões.
 - Para os Ciclos 1 e 2: Número máximo de gerações igual a 90.
 - Para o Ciclo 3: Número máximo de gerações igual a 160 ou aptidão alvo maior ou igual a 550 mil pixels.

Quanto aos hiperparâmetros específicos, foram definidos os seguintes valores:

- Para o algoritmo genético:
 - Tamanho da População: 1000 indivíduos;
 - Chance de Cruzamento: 85%;
 - Taxa de Mutação: alvo de experimentos no capítulo de Resultados e Discussões; e
 - Taxa de Elitismo: 5%.
- Para a estratégia evolutiva:

- Tamanho da População de Progenitores: 100 indivíduos;
- Tamanho da População de Descendentes: 1000 indivíduos; e
- Desvio padrão da Mutação (σ): alvo de experimentos no capítulo de Resultados e Discussões.

O tamanho da população de 1000 indivíduos foi decidido dessa forma com o intuito de equilibrar quantidade de soluções testadas por geração e custo computacional. A base original *Dinossauro-Google* utiliza 2000 indivíduos por padrão, no entanto, essa quantidade mostrou-se demasiadamente custosa e, portanto, optou-se por reduzir esse valor pela metade.

A população de descendentes $\lambda = 1000$ foi definida igual ao tamanho da população do AG, de modo a permitir comparação entre os dois algoritmos evolutivos sob o mesmo número de avaliações por geração. A população de progenitores $\mu = 100$ corresponde a $\lambda/10$, relação que atende à restrição de divisibilidade imposta pela implementação. Os valores definidos para o tamanho das populações de progenitores e descendentes nesse trabalho também são utilizados em igual quantidade no trabalho de Pereira (2023).

Definição dos critérios de comparação

Para possibilitar a avaliação objetiva e reprodutível dos AEs implementados, definiram-se os seguintes critérios de comparação de desempenho:

- **Melhor aptidão média por geração:** valor da melhor aptidão média registrada a cada geração de neuroevolução, utilizado nas curvas de convergência.
- **Estabilidade entre treinamentos:** média aritmética da melhor aptidão por geração calculada dentre todos os treinamentos independentes de cada condição experimental.
- **Seleção de hiperparâmetro campeão:** o valor de mutação com a melhor aptidão média por geração dentro de cada algoritmo evolutivo é declarado campeão.
- **Desempenho no confronto final:** comparação entre os campeões do AG e da EE com base na melhor aptidão média alcançada, na geração de parada e na forma das curvas de

convergência sob protocolo ampliado de avaliação.

3.2 Preparação do Ambiente de Execução

Nessa seção, são detalhadas as atividades realizadas para possibilitar a compilação e execução sem erros da base de código e sua posterior extensão para o desenvolvimento do trabalho. A base de código resultante dessas atividades está disponível em um repositório público do Github¹.

3.2.1 Preparação do ambiente de desenvolvimento

O primeiro passo realizado nessa atividade foi a compilação do código-base. Para isto, foi necessário o uso de alguns recursos relacionados à linguagem C++. São eles:

- **O sistema operacional Ubuntu:** Utilizou-se o sistema Ubuntu, pois, apesar de não ser um requisito de desenvolvimento, é um sistema que contém nativamente muitos recursos necessários para o desenvolvimento em C++, como por exemplo o compilador GCC utilizado para compilar a base de código. O arquivo de instalação deste sistema está disponível de forma gratuita na página na *Web* oficial do sistema.
- **O compilador GCC:** Da sigla em inglês, *GNU Compiler Collection*. Disponível nativamente no sistema operacional Ubuntu.
- **A biblioteca gráfica SDL:** Da sigla em inglês, *Simple DirectMedia Layer*, a biblioteca SDL foi utilizada na extensão da Interface de Usuário da base de código. Disponível para instalação através do programa de linha de comando APT (da sigla em inglês, *Advanced Package Tool*) no Ubuntu. As instruções de instalação estão disponíveis na página *Web* oficial da biblioteca.
- **A biblioteca gráfica PIG:** Da sigla em inglês *Programming Interface for Games*, foi utilizada em conjunto com a biblioteca SDL para adicionar itens à IU da base. Disponível

¹<<https://github.com/luizgadel/Dinossauro-Google>>

na *branch* “breccabral_patch_1” do repositório público do GitHub “PIG” do usuário “breccabral”. Para download do repositório, utilizou-se o programa de linha de comando Git, disponível de forma nativa no sistema operacional Ubuntu.

- **Ferramenta GNU Make:** Necessária para a execução do arquivo “Makefile”, responsável pela definição das regras de compilação do arquivo C++ principal, “main.cpp”. As regras de compilação incluem, por exemplo, o endereço da pasta onde estão o executável do compilador GCC e as bibliotecas gráficas citadas anteriormente.

Após a instalação destes recursos, a compilação dos arquivos foi executada com sucesso e a preparação do ambiente de desenvolvimento foi completada.

3.2.2 Extensão da base de código

Com a compilação da base *Dinossauro-Google* ocorrendo sem problemas, foram realizadas atividades de extensão da base com novas classes responsáveis pela execução da neuroevolução e dos AEs comparados neste trabalho. Ademais, na interface gráfica, foram adicionadas algumas informações com o intuito de facilitar o acompanhamento das várias neuroevoluções executadas no desenvolvimento do trabalho.

Classes de algoritmos evolutivos

Por padrão, a estratégia evolutiva original da base encontrava-se acoplada ao fluxo principal do jogo, o que dificultava a inclusão de novos métodos e a manutenção do código. Para resolver esse problema, foram implementadas classes para separar o processo de neuroevolução, os algoritmos evolutivos e os operadores genéticos dos AEs.

O primeiro passo foi a criação da classe abstrata `EvolutionaryAlgorithm`. Essa classe define um método virtual `Evolve()` que é implementado pelas classes de algoritmos evolucionários com seus respectivos métodos de evolução. Nessa classe, há também o registro, em arquivos CSV, dos resultados de cada geração de neuroevolução processada.

Em seguida, herdando da classe abstrata `EvolutionaryAlgorithm`, foram construídas duas classes de AEs: `GenAlgorithm`, implementando o AG canônico; e `RechenbergMuLambdaES`,

responsável pela EE (μ , λ) de Rechenberg. A estratégia evolutiva originalmente contida na base foi preservada em uma terceira classe, porém não foi empregada nas neuroevoluções conduzidas neste trabalho.

Os operadores genéticos dos AEs também foram extraídos para classes próprias como forma de facilitar a manutenção das classes dos algoritmos evolucionários. Foram criadas as classes `MutationBySubstitution` e `MutationByGaussian` para executar os operadores genéticos de mutação de ambos os AEs. Para o AG, foram criadas ainda as classes `TopNElitism` e `NPointCrossover` para seus operadores genéticos de Elitismo e Cruzamento, respectivamente.

O processo de neuroevolução e a execução do jogo foram encapsulados na classe `DinoRN`. Essa classe utiliza o padrão de projeto *Strategy* para definir o algoritmo evolucionário utilizado a cada processo de neuroevolução. Mais especificamente, a classe `DinoRN` possui um parâmetro, denominado `strategy`, do tipo `EvolutionaryAlgorithm` que, por sua vez, pode conter uma instância de `GenAlgorithm` ou `RechenbergMuLambdaES`. Dessa forma, a lógica de simulação do jogo permanece inalterada enquanto o algoritmo evolutivo pode ser substituído a cada nova neuroevolução. O uso desse padrão de projeto viabilizou a execução sistemática dos experimentos conduzidos no capítulo de Resultados e Discussões.

Ainda no contexto da classe `DinoRN`, responsável pelo processo de neuroevolução, o cálculo da aptidão foi levemente modificado para representar a quantidade percorrida em pixels por cada indivíduo até o momento de sua morte na partida jogada. Além disso, cada geração de indivíduos passou a jogar mais de 1 partida antes de ser avaliada. O número de partidas por geração é controlado por um parâmetro denominado `partidasPorGeracao`. A decisão de aumentar o número de partidas por geração visa mitigar os efeitos dos diversos mecanismos aleatórios presentes no processo de neuroevolução. Como consequência, a aptidão de cada indivíduo passou a ser calculada como a média das distâncias percorridas em pixels durante as partidas que compõem a geração. O valor padrão adotado foi de 25 partidas por geração, com base no valor definido para as neuroevoluções realizadas no trabalho de Pereira (2023). Esse valor foi definido na implementação estendida e declarado como um argumento possível de ser repassado ao executável através do argumento `p`: do *script* de automatização.

Extensão da interface gráfica

Com relação à interface gráfica da neuroevolução, foram adicionados elementos para exibir hiperparâmetros utilizados e informações sobre os indivíduos evoluídos.

A primeira adição realizada foi exibir os genes do melhor indivíduo da geração anterior e dos dez melhores indivíduos da população da geração atual. Essa adição visou facilitar a comparação entre os genes dos melhores indivíduos de duas gerações consecutivas. Na interface, os genes foram representados por círculos coloridos com azul e vermelho. Mais especificamente, escolheu-se representar os genes de valor positivo com azul e os de valor negativo com vermelho. A intensidade de azul ou vermelho utilizada para cada gene é proporcional ao módulo de seu valor.

Foram adicionadas, ainda, informações extras sobre o processo em andamento. Essas informações são descritas a seguir:

- Algoritmo evolucionário em uso, e.g., “AG” ou “EE”;
- Método de mutação em uso, e.g., “sub” para a mutação uniforme no AG e “gauss” para a mutação gaussiana na EE;
- Tempo decorrido desde o início da simulação;
- Horário de início da simulação;
- Frames e ticks (*feedforwards*) por segundo da simulação; e
- Partida atual e número de partidas por geração.

A Figura 3.2 ilustra a interface gráfica após as atividades de sua extensão.

3.2.3 Execução automatizada da Neuroevolução

Construiu-se um *script* na linguagem *Bash* responsável pela execução automatizada de neuroevoluções. Para isso, o *script* assume que está na mesma pasta do executável compilado pela base de código e que o nome desse executável é “main”.

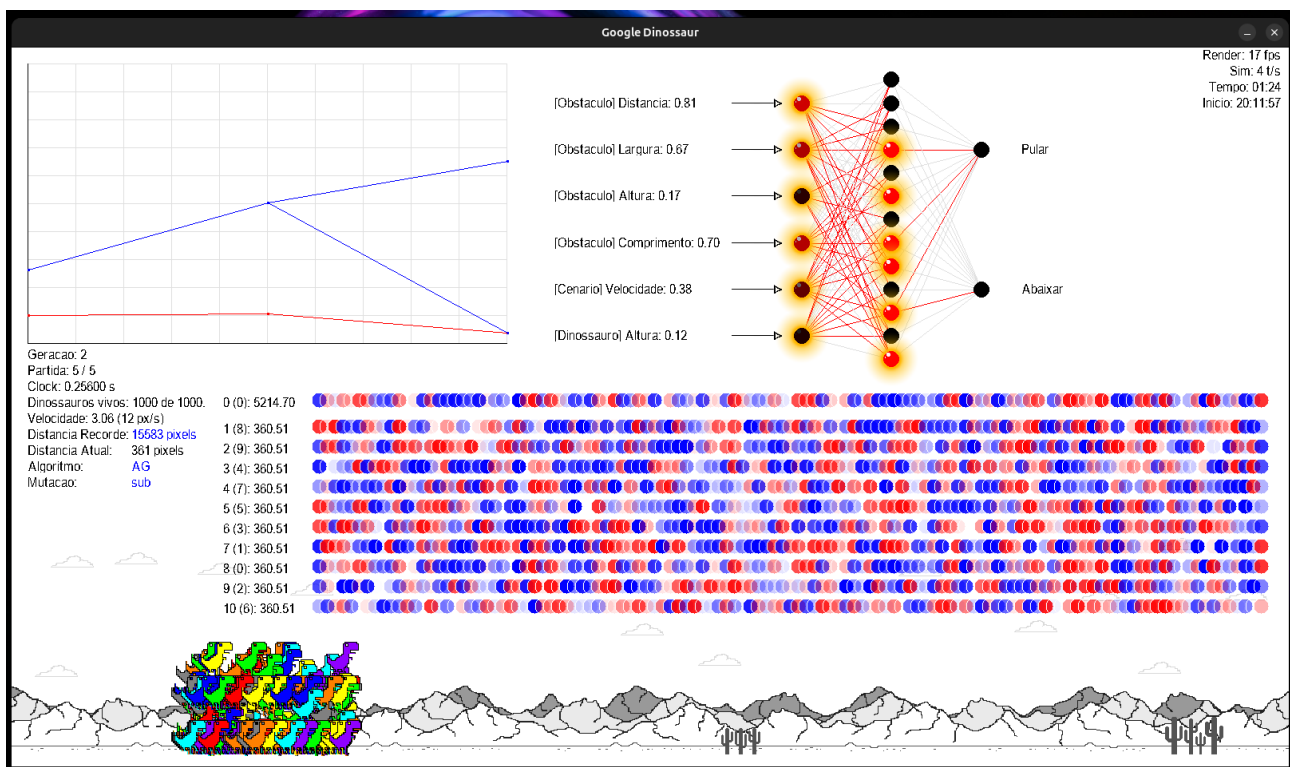


Figura 3.2: Interface gráfica pós-extensão.

No momento de invocação do script, é possível fornecer argumentos de linha de comando para definir parâmetros e hiperparâmetros para as neuroevoluções. Os principais argumentos aceitos pelo *script* são enumerados abaixo:

- I) argumento *ae*: define o algoritmo evolucionário a ser utilizado. Os possíveis valores desse argumento são “AG” para o Algoritmo Genético e “EE” para a Estratégia Evolutiva.
- II) argumento *n*: define a quantidade de neuroevoluções a se executar. Esse argumento aceita apenas números inteiros positivos.
- III) argumento *h*: define o hiperparâmetro utilizado pelo operador genético de mutação. Aceita apenas números reais positivos.
- IV) argumento *p*: define a quantidade de partidas por geração a ser utilizada. Aceita apenas números inteiros positivos.
- V) argumento *g*: define o número de gerações a ser utilizado. Aceita apenas números inteiros positivos.

A Figura 3.3 apresenta o código em *Bash* do *script* criado.

```

$ neuroevolve x
Dinossauro-Google > $ neuroevolve
1  #!/bin/bash
2  # ./neuroevolve ae:AG|ae:EE [n:12] [i:0] [demais tokens repassados ao ./main]
3  set -euo pipefail
4
5  usage() {
6      echo "Uso: $0 ae:AG|ae:EE [n:<int>] [i:<int>] [m:MP] [h:...] [p:<int>] [g:<int>]" >&2
7      echo "  n: execucoes (padrao 12); i: indice fixo repassado ao ./main (padrao 0). Outros tokens: ./main" >&2
8  }
9
10 [[ $# -ge 1 ]] || { usage; exit 1; }
11
12 TOTAL=12 START_INDEX=0 HAS_AE=0 CLI_TOKENS=()
13 for arg in "$@"; do
14     case "$arg" in
15         n:*) TOTAL="${arg#n:}" ;;
16         i:*) START_INDEX="${arg#i:}" ;;
17         ae:AG|ae:EE) HAS_AE=1; CLI_TOKENS+=("$arg") ;;
18         *) CLI_TOKENS+=("$arg") ;;
19     esac
20 done
21
22 [[ "$HAS_AE" -eq 1 ]] || { echo "Erro: ae:AG ou ae:EE e obrigatorio." >&2; usage; exit 1; }
23 [[ "$TOTAL" =~ ^[1-9][0-9]*$ ]] || { echo "Erro: n: deve ser inteiro positivo." >&2; exit 1; }
24 [[ "$START_INDEX" =~ ^[0-9]+$ ]] || { echo "Erro: i: deve ser inteiro >= 0." >&2; exit 1; }
25
26 cd "$(dirname "${BASH_SOURCE[0]}")"
27 [[ -f ./main ]] || { echo "Erro: ./main nao encontrado. Compile com 'make'." >&2; exit 1; }
28 chmod +x ./main 2>/dev/null || true
29
30 desc="${CLI_TOKENS[*]}"
31 for ((j = 0; j < TOTAL; j++)); do
32     echo "== $((j + 1))/$TOTAL ($desc, i:$START_INDEX) =="
33     ./main "${CLI_TOKENS[@]}" "i:$START_INDEX"
34 done
35 echo "Concluido: $TOTAL execucoes ($desc)."
```

Figura 3.3: *Script* em *Bash* para automatizar a execução dos processos de neuroevolução.

Como exemplos de invocação do *script*, tem-se:

- I) `./neuroevolve ae:AG n:10 h:0.01 p:25 g:300`: executa 10 neuroevoluções com o Algoritmo Genético, utilizando a mutação uniforme com chance de 1% e 25 partidas por geração, com 300 gerações.
- II) `./neuroevolve ae:EE n:20 h:0.05 p:50 g:500`: executa 20 neuroevoluções com a Estratégia Evolutiva, utilizando a mutação gaussiana com desvio padrão de 0.05 e 50 partidas por geração, com 500 gerações.

4. Resultados e Discussões

Este capítulo apresenta as atividades de execução e análise dos processos de neuroevolução conduzidos após a implementação do capítulo de Desenvolvimento. Essas atividades contemplam as etapas de *Execução dos Processos de Neuroevolução e Análise dos Resultados* definidas na Metodologia desse trabalho.

Os processos de NE executados visaram comparar o AG canônico de Holland, acrescido de Elitismo, e a EE (μ, λ) de Rechenberg no treinamento de redes neurais para o jogo *Chrome Dino*. Além disso, foram experimentadas variações nas intensidades da Taxa da Mutação uniforme, para o algoritmo genético, e do Desvio padrão (σ) da Mutação gaussiana para a estratégia evolutiva.

As seções seguintes detalham a execução das NEs (Seção 4.1) e a análise comparativa dos resultados obtidos (Seção 4.2).

4.1 Execução dos processos de neuroevolução

Os processos de NEs executados nesse trabalho foram divididos em três ciclos sequenciais de experimentos. Esses ciclos são descritos a seguir:

- I) No **Ciclo 1**, avaliou-se o AG com três taxas de mutação uniforme (1%, 2% e 3%). Para cada taxa, executaram-se 12 neuroevoluções com 25 partidas por geração e condição de parada em 90 gerações completas, identificando-se ao final o hiperparâmetro campeão do ciclo.
- II) No **Ciclo 2**, avaliou-se a EE com três valores de desvio padrão da mutação gaussiana (0,01; 0,05 e 0,10). Para cada valor de σ , repetiu-se o mesmo esquema de 12 neuroevoluções,

25 partidas por geração e 90 gerações completas, declarando-se ao final o hiperparâmetro campeão da EE.

III) No **Ciclo 3**, conduziu-se um confronto entre os hiperparâmetros campeões dos ciclos anteriores, por meio de 24 neuroevoluções para cada algoritmo evolutivo. Nesse ciclo, cada geração passou a ser formada de 50 partidas e adotaram-se duas condições de parada independentes: 160 gerações completas ou aptidão média de um indivíduo igual ou superior a 550 mil.

Com exceção do hiperparâmetro de valor 0,10 do desvio padrão, todos os valores testados para o hiperparâmetro das mutações foram escolhidos de forma empírica. O hiperparâmetro de valor 0,10 do desvio padrão foi baseado no trabalho de Pereira (2023).

Também com base no trabalho de Pereira (2023), definiram-se 12 neuroevoluções por experimento. As 25 partidas por geração herdam o padrão definido no capítulo de Desenvolvimento e são adequadas aos ciclos de experimentos com valores de hiperparâmetro de mutação. O horizonte de 90 gerações, por sua vez, foi o número máximo de gerações definidas para o Ciclo 1 e 2, pois, em testes empíricos, estimou-se que essa quantidade de gerações seria neuroevoluída em 2 horas, em média, e 8 horas de execução no máximo. Dessa forma, com essa quantidade de gerações por neuroevolução, era possível terminar, em média, 12 neuroevoluções de um dado experimento a cada dia ou, de forma prática, um experimento completo por dia.

A Tabela 4.1 resume as condições de execução de cada ciclo definido de NEs.

Tabela 4.1: Organização dos experimentos por ciclo de neuroevolução

ID	Ciclo	Algoritmo evolutivo	Hiperparâmetro variado	Partidas/ger.	Gerações	n
AG-1	1	AG	Mut. Uniforme (1%)	25	90	12
AG-2	1	AG	Mut. Uniforme (2%)	25	90	12
AG-3	1	AG	Mut. Uniforme (3%)	25	90	12
EE-010	2	EE (μ, λ)	Mut. gaussiana ($\sigma = 0,01$)	25	90	12
EE-050	2	EE (μ, λ)	Mut. gaussiana ($\sigma = 0,05$)	25	90	12
EE-100	2	EE (μ, λ)	Mut. gaussiana ($\sigma = 0,10$)	25	90	12

É interessante dizer que tanto as combinações de AG e valores de Taxa de mutação, quanto os conjuntos de EE e valores de desvio padrão receberam codinomes para facilitar discussões sobre seus resultados. O algoritmo genético que utiliza a taxa de mutação igual à 1%, por

exemplo, recebeu o codinome “AG-1”, enquanto a EE que utiliza o desvio padrão $\sigma = 0,05$ tem o codinome de EE-050. Os codinomes de todas as combinações de AEs e hiperparâmetros experimentados nesse trabalho são exibidos na coluna ID da Tabela 4.1.

Durante todos os ciclos de experimentos, os demais hiperparâmetros do processo de NE permaneceram fixos conforme suas definições no Capítulo 3. Ademais, a execução repetida das NEs em cada experimento foi automatizada por meio do *script* em linguagem *Bash* também apresentado no Capítulo 3.

As seções a seguir apresentam os resultados obtidos durante os todos os ciclos de experimentos. Para cada experimento realizado, os resultados obtidos serão apresentados através de figuras e discutidos em detalhe.

4.1.1 Ciclo 1 de neuroevoluções

O Ciclo 1 compreendeu a avaliação do AG canônico com três taxas de mutação uniforme: 1%, 2% e 3%. Nesse Ciclo, as neuroevoluções foram executadas durante 90 gerações. Dessa forma, nas figuras que apresentam os resultados obtidos para cada experimento, o eixo das abscissas exibe valores de 1 a 90 e está dividido em faixas posicionadas a cada geração múltipla de 15. O eixo das ordenadas, no entanto, não tem uma escala fixa, mas variável de acordo com os resultados observados para cada experimento.

A Figura 4.1 apresenta gráficos de linha com a melhor aptidão por geração para cada uma das 12 neuroevoluções executadas com a combinação de codinome AG-1, isto é, o algoritmo genético que utiliza taxa de mutação igual à 1%. Nessa figura, o eixo das ordenadas exibe valores de 0 a 175000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 25000. O eixo das abscissas, por sua vez, exibe valores de 1 a 90, representativo das gerações executadas.

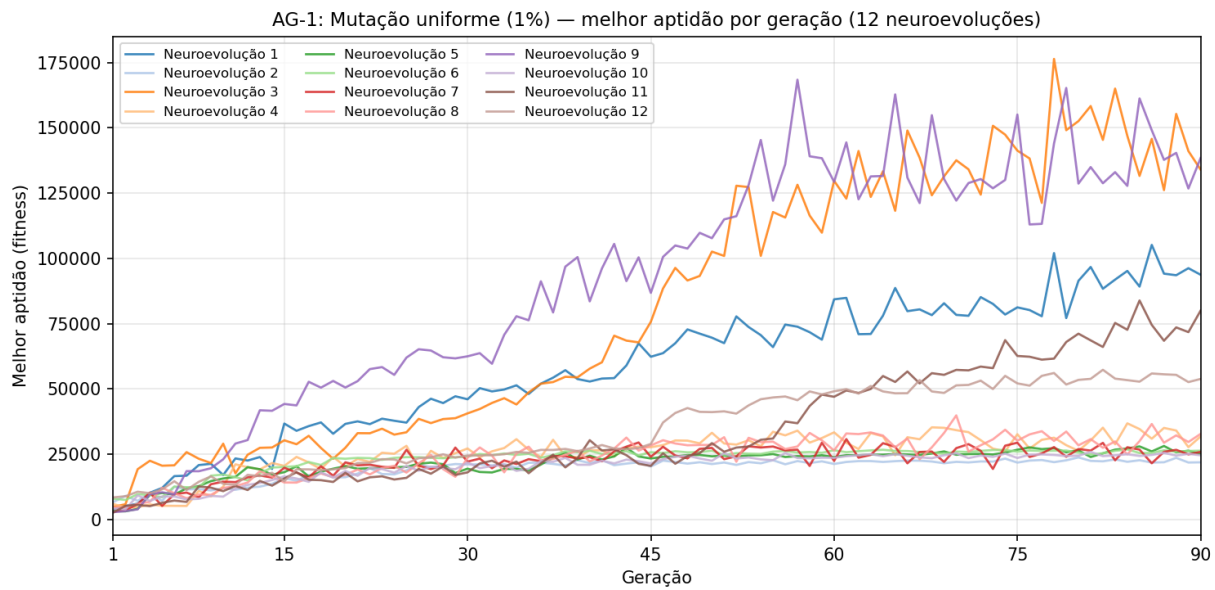


Figura 4.1: Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com algoritmo genético e mutação uniforme de 1% (AG-1).

Pela Figura 4.1, nota-se que, para o AG-1, as curvas mantêm-se próximas nas gerações iniciais. Na geração 15, a maioria das curvas não havia atingido a faixa de 25000 de valor de aptidão, conseqüentemente, o valor da mediana dos resultados de todas as curvas também estava abaixo de 25000.

A partir da 15^a geração, é possível observar que três curvas (representantes das neuroevoluções de índice 1, 3 e 9) foram capazes de superar a faixa de 25000 de valor de aptidão e, portanto, afastar-se do valor apresentado pela mediana dos resultados. Ao longo das 30 gerações seguintes, essas curvas continuam como as únicas a se distanciar da mediana do conjunto de experimentos e, na geração 45, atingem aptidões entre 50000 e 100000. Após a 45^a geração, outras duas curvas (neuroevoluções 11 e 12) também superam a mediana observada, que, nessa geração, continuava próxima de 25000. Na geração 75, essas duas curvas alcançaram valores entre 50000 e 75000.

Na geração 90, a geração final dos experimentos, 5 das 12 curvas foram capazes de atingir valores acima da faixa de 50000. Dessas, 2 curvas superaram 125000 de aptidão.

A Figura 4.2 apresenta um gráfico de linha com a média e o primeiro desvio padrão da melhor aptidão por geração para as 12 neuroevoluções realizadas com mutação uniforme de

1%. Nessa figura, o eixo das ordenadas exibe valores de 0 a 100000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 20000. O eixo das abscissas, por sua vez, exibe valores de 1 a 90, com faixas posicionadas a cada 15 gerações.

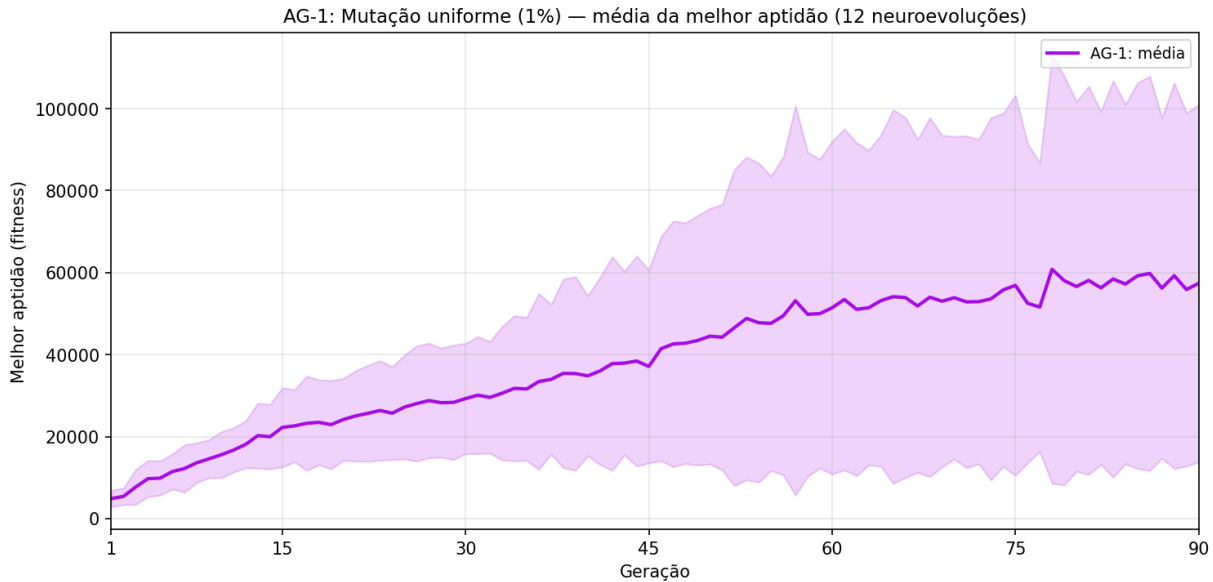


Figura 4.2: Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs do AG-1 no Ciclo 1.

Pela Figura 4.2, observa-se que a média cresce de forma contínua ao longo das 90 gerações, partindo de valores abaixo de 20000 e atingindo valores de, aproximadamente, 58000 de aptidão na geração final.

Após a geração 15, o ritmo de crescimento desacelera e a faixa de um desvio padrão começa a se expandir. Na geração 90, essa faixa abrange aptidões entre aproximadamente 20000 e 100000. Essa discrepância entre os limites do primeiro desvio padrão da média indicam algo já visível na Figura 4.1: certas execuções estagnaram em patamares baixos enquanto outras superaram a mediana e, conseqüentemente, elevaram o teto da distribuição.

A seguir, a Figura 4.3 apresenta gráficos de linha com a melhor aptidão por geração para as 12 neuroevoluções executadas com taxa de mutação uniforme de 2%. Nessa figura, o eixo das ordenadas exibe valores de 0 a 160000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 20000. O eixo das abscissas, por sua vez, exibe valores de 1 a 90, correspondentes as gerações computadas.

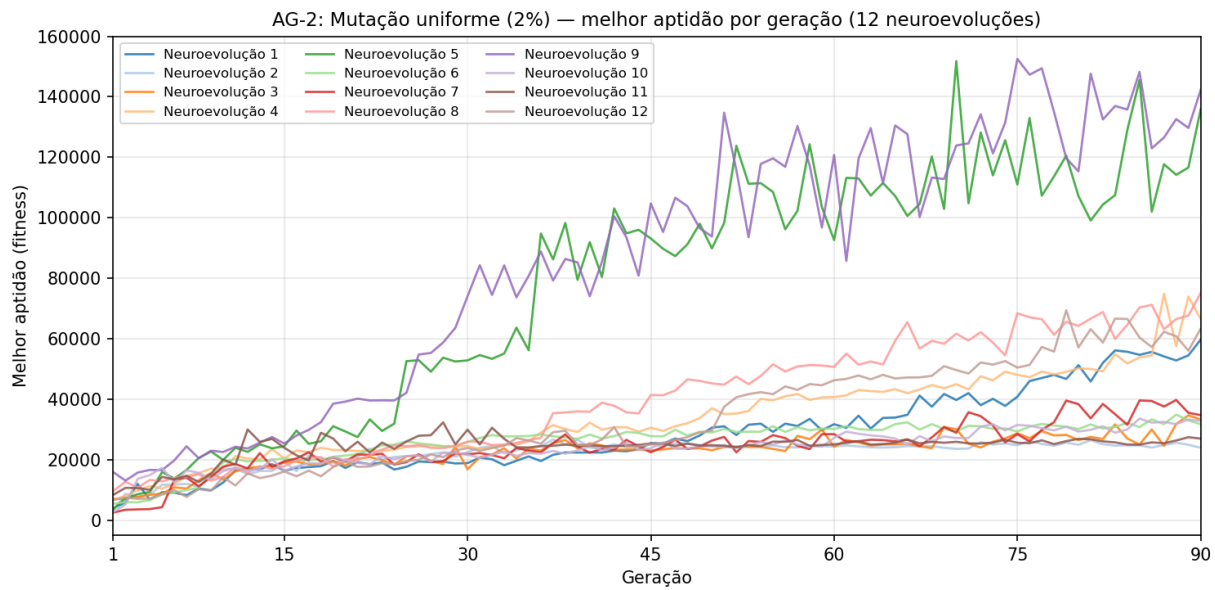


Figura 4.3: Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com algoritmo genético e mutação uniforme de 2% (AG-2).

Pela Figura 4.3, nota-se que, para o AG-2, as curvas desenvolvem aptidões próximas durante as gerações iniciais, o que repete o padrão avistado no Figura 4.1 para o AG-1. Na geração 15, nenhuma curva havia se afastado das demais e o valor da mediana estava próximo da faixa de 20000.

A partir da 15ª geração, 2 curvas (neuroevoluções 5 e 9) se destacam do conjunto de curvas. Nas gerações seguintes, essas curvas distanciam-se acentuadamente e, na geração 45, atingem valores entre 80000 e 100000 de aptidão. Além dessas, outras 4 curvas (NEs 1, 4, 8, 12) são capazes de se afastar do conjunto de curvas estagnadas e, por volta da geração 75, evoluem valores de aptidão entre 40000 e 80000.

Na última geração, 6 das 12 curvas apresentaram aptidões muito próximas ou acima de 60000, enquanto as demais curvas obtiveram aptidões entre as faixas de 20000 e 40000. Além disso, 2 curvas superaram a faixa de 120000 de aptidão.

A Figura 4.4 apresenta um gráfico de linha com a média e o primeiro desvio padrão da melhor aptidão por geração para as 12 neuroevoluções realizadas com mutação uniforme de 2%. Nessa figura, o eixo das ordenadas exibe valores de 0 a 100000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 20000. No eixo das abscissas, a escala varia de 1 a 90,

com faixas posicionadas a cada 15 gerações.

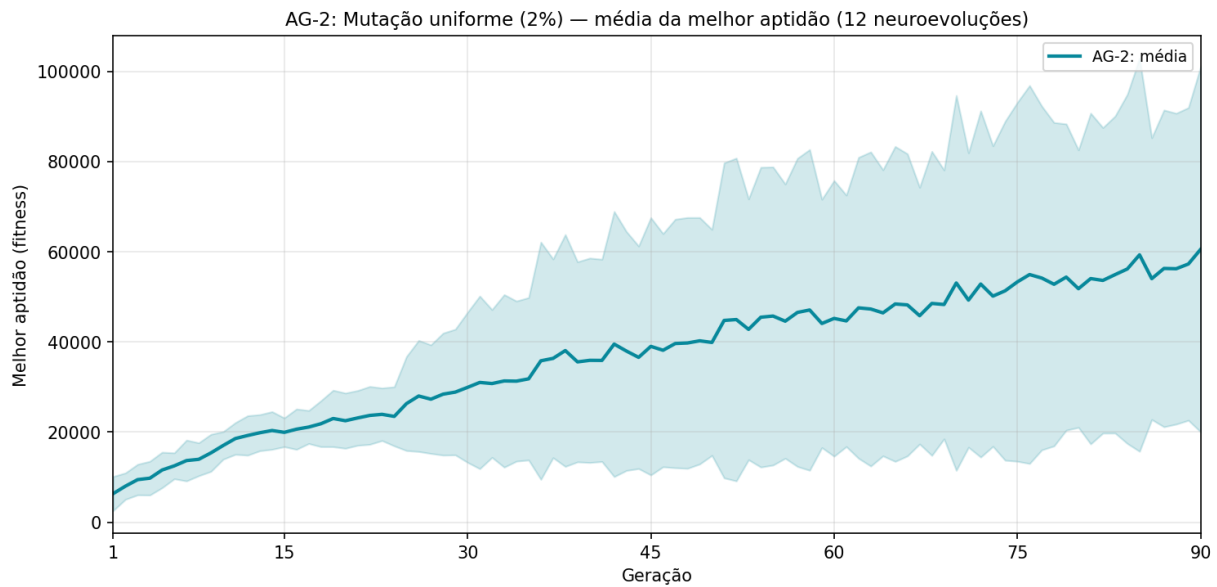


Figura 4.4: Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs do AG-2 no Ciclo 1.

Pela Figura 4.4, é possível avaliar que a média das aptidões cresce de forma progressiva ao longo das 90 gerações. Na geração final, a média atinge cerca de 60000 de valor de aptidão.

A faixa do primeiro desvio padrão a partir da média, por sua vez, expande-se a partir da geração 15, o que acompanha a separação das curvas, possível de observar-se na Figura 4.3. Na geração 90, o intervalo do primeiro desvio padrão abrange aptidões entre aproximadamente 20000 e 100000.

Em seguida, a Figura 4.5 apresenta gráficos de linha com a melhor aptidão por geração para as 12 neuroevoluções executadas com taxa de mutação uniforme de 3%. Nessa figura, o eixo das ordenadas exibe valores de 0 a 350000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 50000. O eixo das abscissas, por sua vez, exibe valores de 1 a 90 e está dividido em faixas posicionadas em cada geração múltipla de 15.

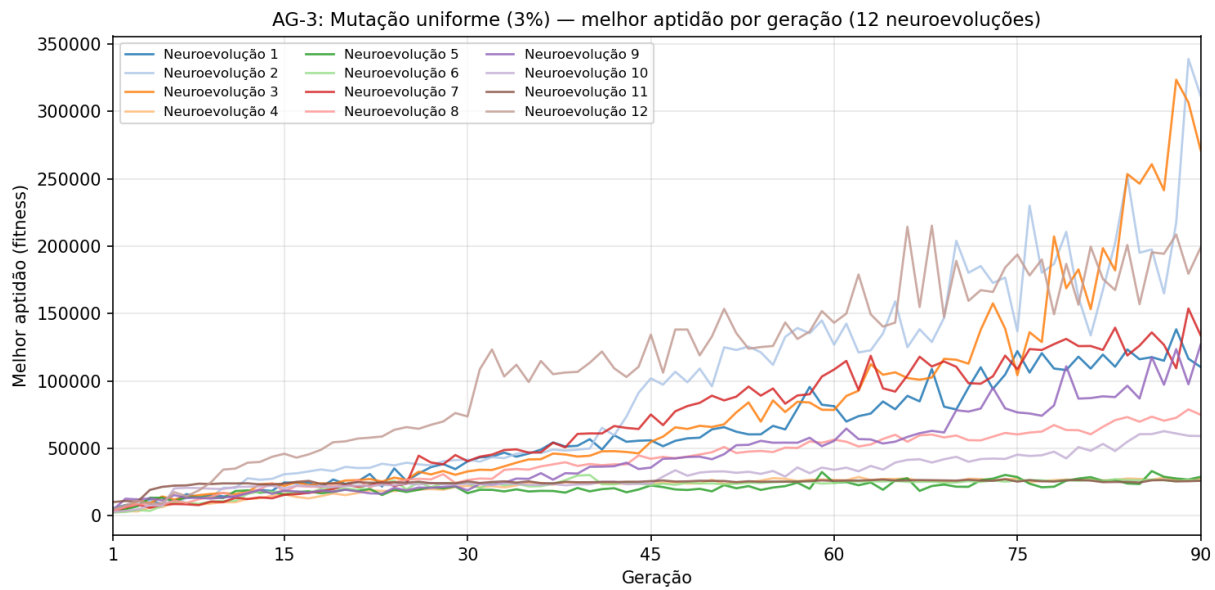


Figura 4.5: Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com algoritmo genético e mutação uniforme de 3% (AG-3).

Através da Figura 4.5, constata-se que, na geração 15 dos experimentos com o AG-3, cerca de 10 das 12 curvas encontram-se próximas da metade da faixa de 50000 do gráfico, isto é, próximas do valor de 25000 de aptidão. Além disso, até essa geração, apenas 1 curva havia sido capaz de se separar das demais e atingir valores muito próximos de 50000. Esses dados apontam para um comportamento já observado nas figuras dos experimentos anteriores, em que ao menos metade das curvas estagna em uma faixa de valor de aptidão atingida até a geração 15 e, ao longo das 75 gerações restantes, não consegue superar a faixa seguinte.

No entanto, para o AG-3, o comportamento observado é diferente: após a geração 15, a maioria das curvas é capaz de se separar e, até a geração 45, é possível apontar que 5 das 12 curvas (neuroevoluções 1, 2, 3, 7 e 12) haviam superado a faixa de 50000 de aptidão. Além dessas 5, outras 2 curvas (NEs 8 e 9) apresentavam um crescimento além da faixa de estagnação das outras curvas, mas não atingem o valor de 50000. Após a geração 60, a NE de índice 10 também foi capaz de evoluir aptidões além do ponto de estagnação, atingindo a faixa de 60000 ao se aproximar da 90ª geração.

No fim, 8 das 12 curvas exibiram aptidões acima do ponto de estagnação, que ficou posicionado na metade da faixa de 50000 de aptidão. Dessas 8 curvas, 6 foram capazes de superar a

faixa de 100000 de aptidão. Destaca-se que a melhor curva do AG-3 atingiu valores acima da faixa de 300000 de aptidão, uma faixa nunca superada até então.

A Figura 4.6 apresenta um gráfico de linha com a média e o primeiro desvio padrão da melhor aptidão por geração para as 12 neuroevoluções realizadas com mutação uniforme de 3%. Nessa figura, o eixo das ordenadas exibe valores de 0 a 200000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 50000. No eixo das abscissas, a escala varia de 1 a 90, com faixas desenhadas a cada 15 gerações.

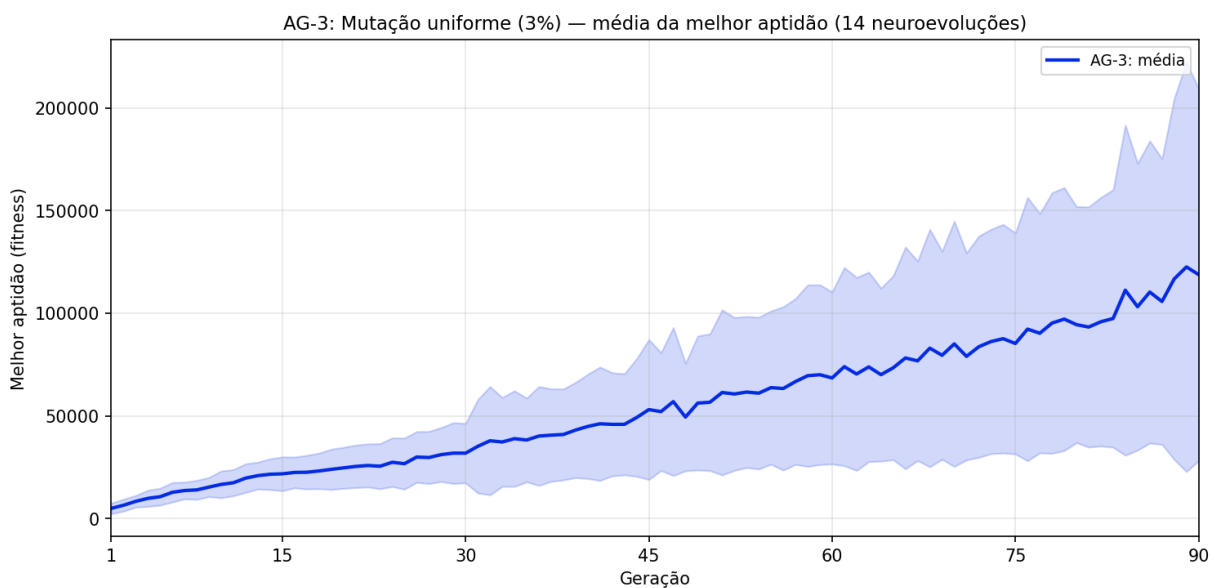


Figura 4.6: Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs do AG-3 no Ciclo 1.

Pela Figura 4.6, observa-se que a média cresce progressivamente ao longo das 90 gerações e, no fim, atinge um valor entre 100000 e 150000 de aptidão. A faixa de um desvio padrão expande-se de forma acentuada a partir da geração 30, acompanhando o crescimento observado nos treinamentos individuais após essa geração. Na geração 90, a faixa do primeiro desvio padrão abrange aptidões entre aproximadamente 30000 e 200000.

A Figura 4.7, por sua vez, apresenta a comparação dos resultados observados nesse ciclo. Nessa figura, o eixo das ordenadas exibe valores de 0 a 120000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 20000. No eixo das abscissas, a escala varia de 1 a 90 e faixas posicionadas a cada 15 gerações, como nos experimentos realizados anteriormente nesse ciclo.

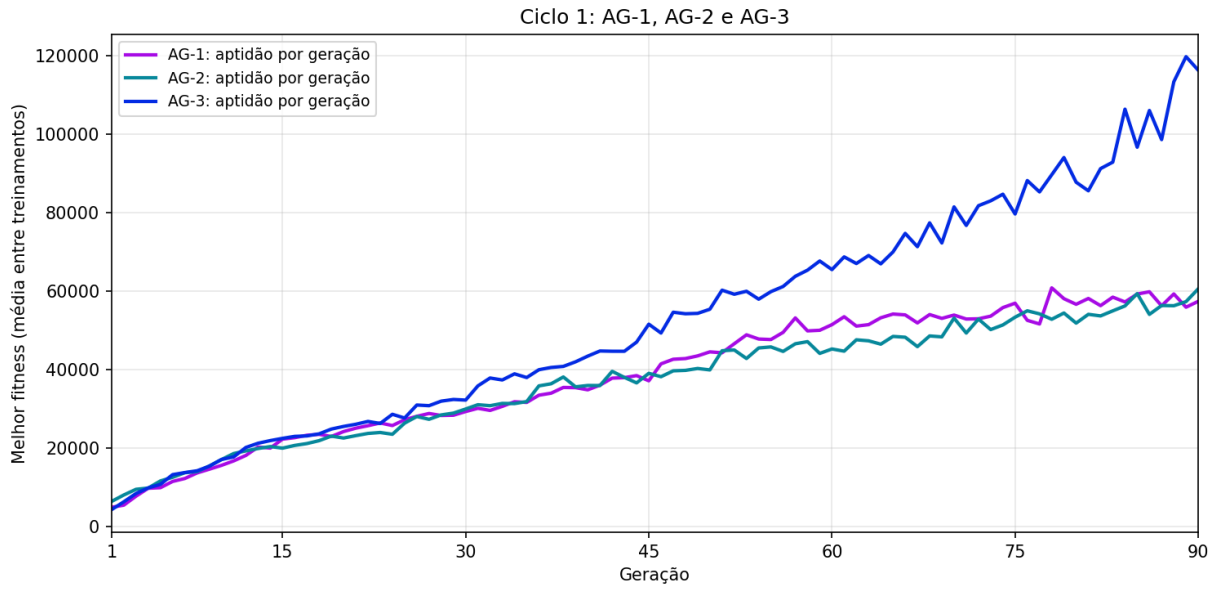


Figura 4.7: Comparação das curvas de convergência do melhor *fitness* por geração entre AG-1, AG-2 e AG-3 (média entre os 12 treinamentos).

Na comparação direta exibida nessa Figura, as curvas apresentam aptidões médias similares no início das neuroevoluções. A partir da geração 20, o AG-3 assume a liderança. Na geração 80, o AG-3 já havia superado a marca de 80000 de aptidão. Nos demais concorrentes, a melhor aptidão atingida estava próxima de atingir 60000. Na geração 90, a condição de parada dessas neuroevoluções, o AG-3 alcança um valor de aptidão entre 100000 e 120000. Para o AG-1 e o AG-2, no entanto, as aptidões atingidas nessa geração final são de aproximadamente 60000, tendo o AG-2 superado o AG-1 por uma pequena margem.

Dessa forma, a combinação do algoritmo genético com a taxa de mutação uniforme de 3%, de codinome AG-3, foi declarada a campeã do Ciclo 1.

4.1.2 Ciclo 2 de neuroevoluções

O Ciclo 2 compreendeu a avaliação da EE de Rechenberg (μ, λ) com três valores de desvio padrão (σ) da mutação gaussiana: 0,01; 0,05 e 0,10. Nesse ciclo, as neuroevoluções também foram executadas durante 90 gerações, de forma que o eixo das abscissas nas figuras subsequentes varia de 1 a 90, com faixas posicionadas a cada geração múltipla de 15. O eixo das ordenadas, por sua vez, permanece variável conforme os resultados observados em cada experimento.

A Figura 4.8 apresenta gráficos de linha com a melhor aptidão por geração para cada uma das 12 neuroevoluções executadas com a combinação de codinome EE-010, isto é, a estratégia evolutiva que utiliza mutação gaussiana com desvio padrão $\sigma = 0,01$. Nessa figura, o eixo das ordenadas exibe valores de 0 a 140000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 20000. O eixo das abscissas, por sua vez, exibe valores de 1 a 90, representativos das gerações executadas.

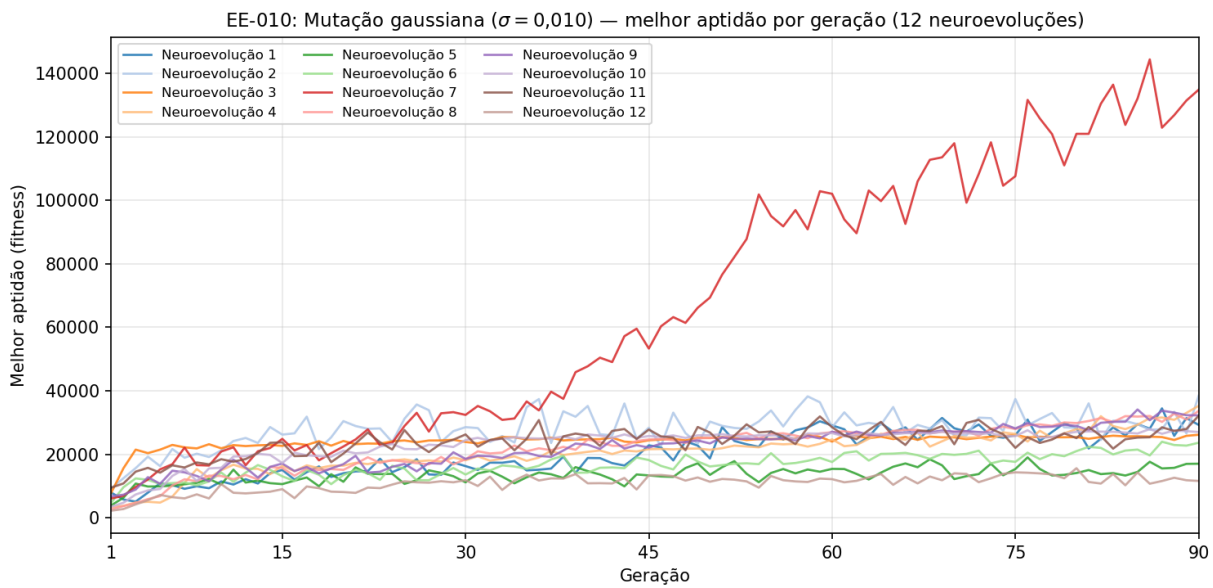


Figura 4.8: Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com estratégia evolutiva e mutação gaussiana de $\sigma = 0,01$ (EE-010).

Pela Figura 4.8, nota-se que, durante as 15 primeiras gerações das neuroevoluções para a EE-010, as 12 curvas dispersam-se em um intervalo compreendido entre 0 e 40000 de aptidão. Na geração 15, no entanto, a grande maioria das curvas ainda se encontrava abaixo da faixa de 20000 de aptidão e o valor da mediana também permanecia nesse patamar.

A partir da 30ª geração, uma curva (neuroevolução 7) consegue afastar-se do conjunto e passa a crescer de maneira acentuada. Até o fim das neuroevoluções, essa curva permanece como a única a fugir da faixa de estagnação definida pelos resultados das outras NEs. Na geração 90, a NE 7 atinge valores acima da faixa de 125000. As demais 11 curvas progridem lentamente, mas, ao fim dos experimentos, nenhuma foi capaz de superar o patamar de 40000 de aptidão.

A Figura 4.9 apresenta um gráfico de linha com a média e o primeiro desvio padrão da melhor aptidão por geração para as 12 neuroevoluções realizadas com mutação gaussiana de $\sigma = 0,01$. Nessa figura, o eixo das ordenadas exibe valores de 0 a 70000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 10000.

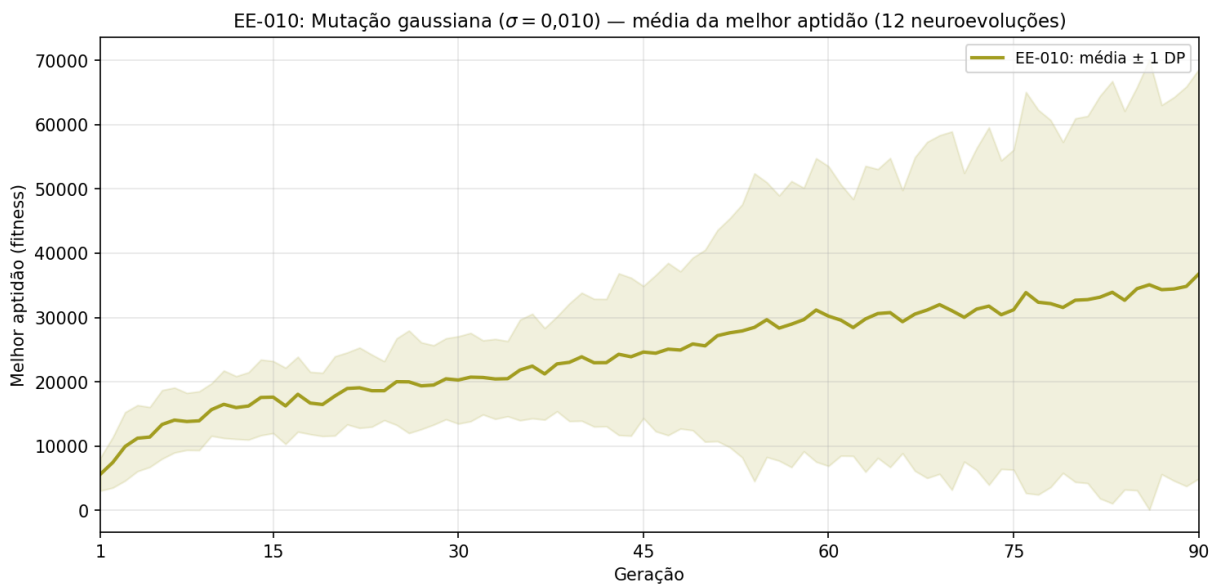


Figura 4.9: Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs da EE-010 no Ciclo 2.

Pela Figura 4.9, observa-se que a média das aptidões para a EE-010 cresce de forma contínua, porém moderada, ao longo das 90 gerações, partindo de valores abaixo de 10000 e atingindo valores entre 30000 e 40000 na geração final.

Após a geração 45, a faixa de um desvio padrão expande-se de forma acentuada, refletindo a distância crescente entre as curvas de destaque, que superaram a faixa de estagnação, e o grupo estagnado visível na Figura 4.8. Na geração 90, essa faixa abrange aptidões entre aproximadamente 5000 e 70000.

A seguir, a Figura 4.10 apresenta gráficos de linha com a melhor aptidão por geração para as 12 neuroevoluções executadas com desvio padrão da mutação gaussiana de 0,05. Nessa figura, o eixo das ordenadas exibe valores de 0 a 175000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 25000. O eixo das abscissas, por sua vez, exibe valores de 1 a 90, relacionados às gerações executadas.

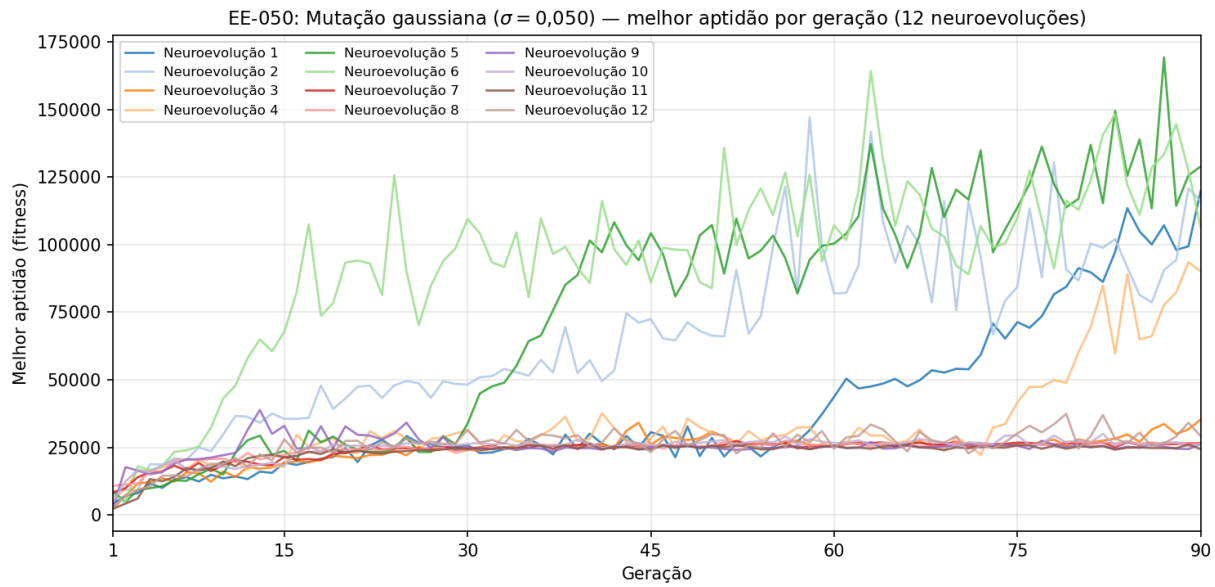


Figura 4.10: Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com estratégia evolutiva e mutação gaussiana de $\sigma = 0,05$ (EE-050).

Pela Figura 4.10, nota-se que as NEs para a EE-050 parecem seguir o padrão observado nos experimentos anteriores, em que, no começo das neuroevoluções, todas as curvas desenvolvem aptidões similares. No entanto, antes mesmo da 15ª geração, já é possível identificar uma curva de destaque (neuroevolução 6) que, nesse ponto, é capaz de atingir valores acima da faixa de 50000 de aptidão, enquanto a maioria das outras curvas encontram-se abaixo da faixa de 25000.

Entre as gerações 30 e 75, outras curvas (NEs 4, 5, 11 e 12) superam a mediana dos experimentos. Por volta da geração 90, as curvas capazes de ultrapassar o valor da mediana evoluem aptidões entre 75000 e 170000. Entretanto, cerca de 7 das 12 curvas permanecem estagnadas e, na geração final, terminam suas neuroevoluções com valores de aptidão entre as faixas de 25000 e 50000.

A Figura 4.11 apresenta um gráfico de linha com a média e o primeiro desvio padrão da melhor aptidão por geração para as 12 neuroevoluções realizadas com mutação gaussiana de $\sigma = 0,05$. Nessa figura, o eixo das ordenadas exibe valores de 0 a 120000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 20000. O eixo das abscissas, como nos outros experimentos, apresenta uma escala que varia de 1 a 90 e faixas posicionadas a cada 15 gerações.

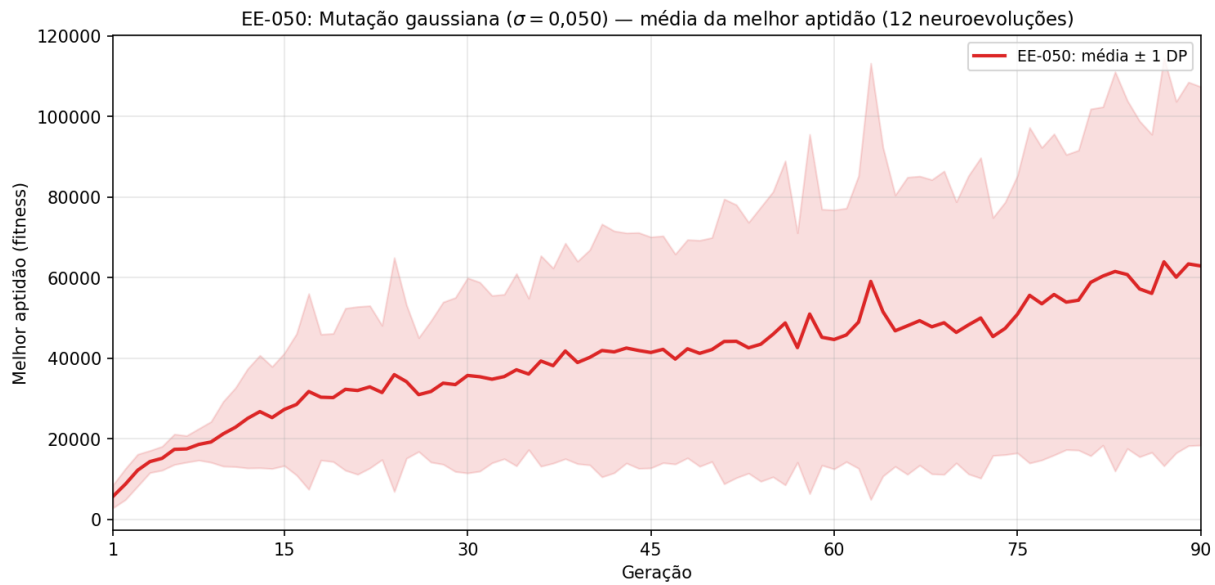


Figura 4.11: Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs da EE-050 no Ciclo 2.

Pela Figura 4.11, é possível avaliar que o crescimento da média das aptidões sustenta-se ao longo das 90 gerações. Na geração final, a média supera a faixa de 60000 de aptidão e atinge um valor de, aproximadamente, 65000 de aptidão.

A faixa do primeiro desvio padrão a partir da média expande-se a partir da geração 15, consequência da separação das curvas nessa geração observada na Figura 4.10. Na geração 90, o intervalo do primeiro desvio padrão abrange aptidões entre aproximadamente 20000 e 110000.

Pelo critério de comparação definido, i.e., a média das aptidões por geração obtidas pelas neuroevoluções, a EE-050 superou a EE-010 e é, até o momento, a campeã do Ciclo 2.

Em seguida, a Figura 4.12 apresenta gráficos de linha com a melhor aptidão por geração para as 12 neuroevoluções executadas com desvio padrão da mutação gaussiana de 0,10. Nessa figura, o eixo das ordenadas exibe valores de 0 a 120000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 20000. O eixo das abscissas, por sua vez, exibe valores de 1 a 90 e está dividido em faixas posicionadas em cada geração múltipla de 15.

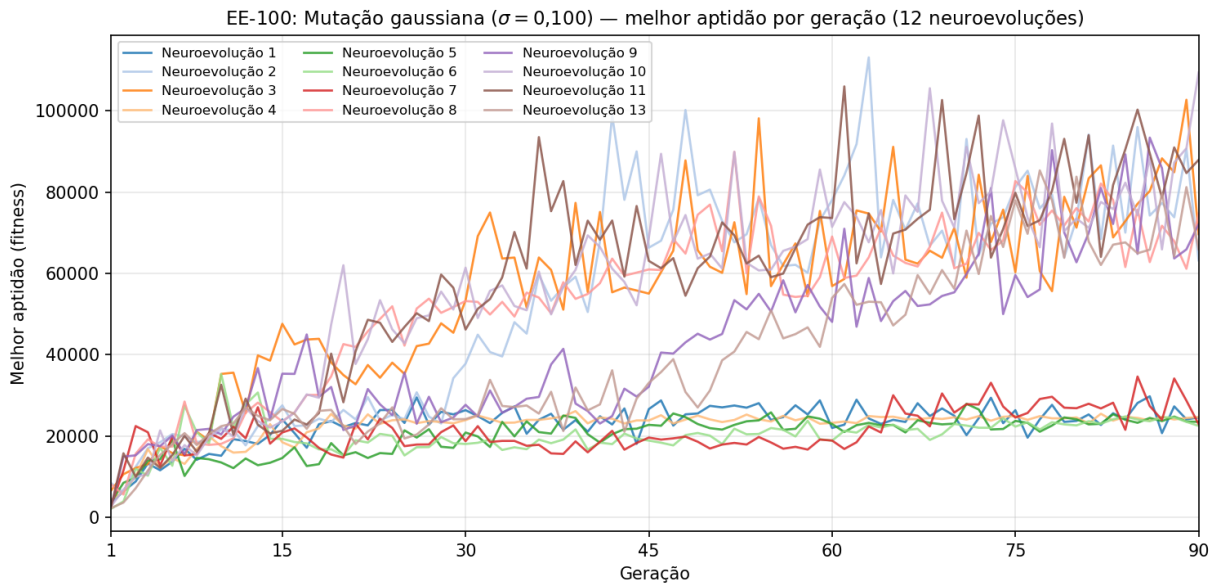


Figura 4.12: Gráficos de linha da melhor aptidão por geração para as 12 neuroevoluções realizadas com estratégia evolutiva e mutação gaussiana de $\sigma = 0,10$ (EE-100).

Através da Figura 4.12, é possível constatar que, até a geração 15, as aptidões desenvolvidas pelas neuroevoluções para a EE-100 estavam na faixa de 0 a 40000, com apenas uma curva superando esse patamar, a curva da NE n° 3.

No entanto, após a geração 15, o conjunto divide-se em dois grupos. O primeiro grupo é formado por 5 curvas (NEs 1, 4, 5, 6 e 7) que acabam suas neuroevoluções na faixa de estagnação. Para essas curvas, as aptidões finais ficaram entre os valores de 20000 e 40000. No segundo grupo estão 7 curvas (NEs 2, 3, 8, 9, 10, 11 e 12) que foram capazes de evoluir além da faixa de estagnação. Essas, por sua vez, finalizaram seus processos de NE com aptidões entre 60000 e 120000. Destaca-se que a melhor curva da EE-100 superou a faixa de 100000 de aptidão.

A Figura 4.13 apresenta um gráfico de linha com a média e o primeiro desvio padrão da melhor aptidão por geração para as 12 neuroevoluções realizadas com mutação gaussiana de $\sigma = 0,10$. Nessa figura, o eixo das ordenadas exibe valores de 0 a 80000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 20000. No eixo das abscissas, a escala varia de 1 a 90, com faixas desenhadas a cada 15 gerações.

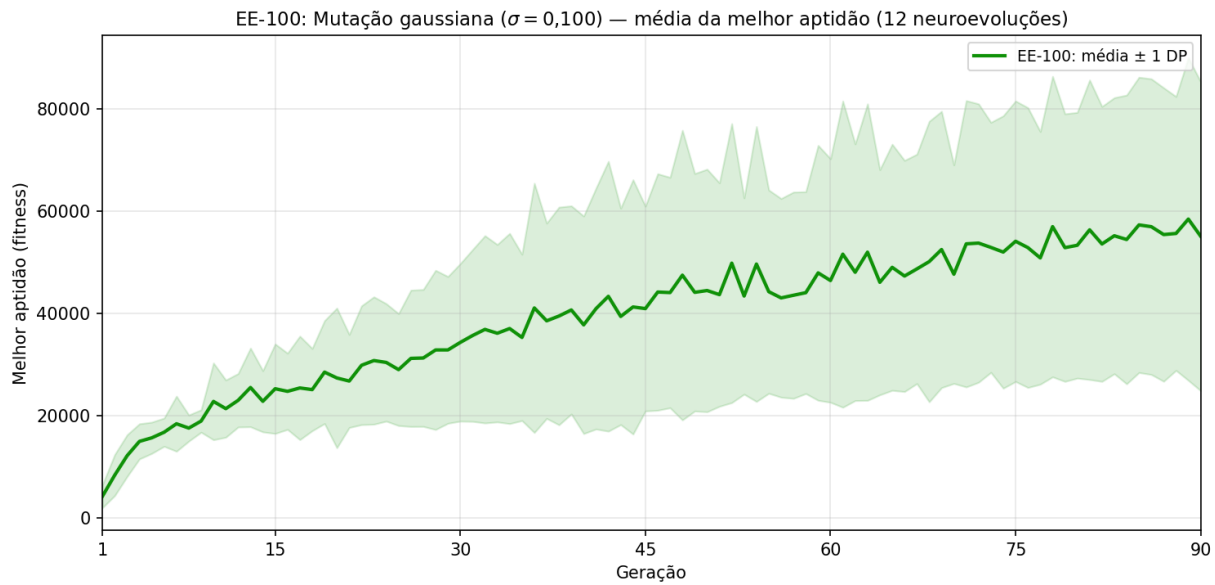


Figura 4.13: Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs da EE-100 no Ciclo 2.

Pela Figura 4.13, pode-se observar que a média cresce progressivamente ao longo das 90 gerações de neuroevolução. No fim das gerações, a média das aptidões atinge um valor de, aproximadamente, 57000 de aptidão.

Nota-se, ainda, que a faixa corresponde ao primeiro desvio padrão a partir da média expande-se gradualmente a partir da geração 20, o que reflete a bifurcação presenciada na Figura 4.12 que exhibe as curvas individuais por neuroevolução. Na geração 90, a faixa do primeiro desvio padrão abrange aptidões entre aproximadamente 25000 e 90000.

Por fim, a Figura 4.14 apresenta a comparação dos resultados observados nesse ciclo de experimentos com as três variações de estratégia evolutiva EE-010, EE-050 e EE-100. Nessa figura, o eixo das ordenadas exhibe valores de 0 a 60000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 10000 de aptidão. No eixo das abscissas, como nos demais experimentos realizados até então, a escala varia de 1 a 90, com faixas desenhadas a cada 15 gerações.

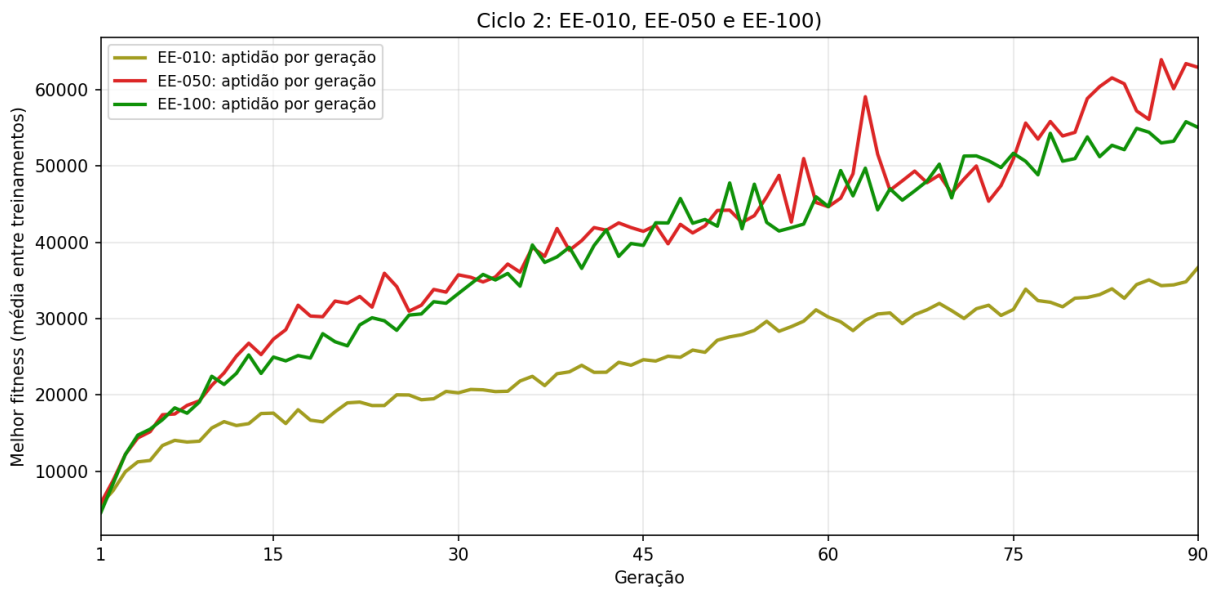


Figura 4.14: Comparação das curvas de convergência do melhor *fitness* por geração entre EE-010, EE-050 e EE-100 (média entre os 12 treinamentos).

Na comparação direta exibida nessa Figura, as três curvas apresentam aptidões médias distintas desde o fim da primeira faixa posicionada no eixo das abscissas, na geração de n° 15. Nessa geração, nota-se a aptidão média da EE-010 abaixo da faixa de 20000, enquanto as aptidões média da EE-050 e da EE-100 encontram-se, aproximadamente, acima do valor de 30000 de aptidão. A partir da geração 15, a EE-050 e a EE-100 distanciam-se da EE-010, que mantém o ritmo de crescimento mais lento ao longo de todo o horizonte de gerações definido.

Embora a EE-050 e a EE-100 permaneçam próximas durante todo o gráfico, a EE-050 consolida vantagem nas gerações finais. Na geração 90, condição de parada dessas neuroevoluções, a EE-050 lidera com aptidão média acima de 60000, seguida pela EE-100, com aptidão de entre 55000 e 60000. A EE-010 finaliza a comparação com uma aptidão entre 30000 e 40000.

Considerando o critério de comparação estabelecido, isto é, a média da aptidão final das neuroevoluções, a combinação da estratégia evolutiva com o desvio padrão de 0,05 da mutação gaussiana, de codinome EE-050, foi declarada campeã da EE no Ciclo 2 devido a ter apresentado a maior média da aptidão final.

4.1.3 Ciclo 3 de neuroevoluções

O Ciclo 3 compreendeu o confronto final entre as combinações campeãs declaradas ao término dos Ciclos 1 e 2. São elas: a combinação de AG e mutação uniforme de 3% (AG-3) e a combinação de EE e mutação gaussiana com desvio padrão $\sigma = 0,05$ (EE-050). Por se tratar do confronto final, essas combinações recebem, neste ciclo, os codinomes AG-F e EE-F, respectivamente. As condições de execução desse confronto são visualizadas na Tabela 4.2.

Tabela 4.2: Condições de execução do Ciclo 3 de neuroevolução (confronto final)

ID	Algoritmo evolutivo	Hiperparâmetro de mutação	Partidas/ger.	Gerações	n
AG-F	AG (campeão)	Mutação uniforme (3%)	50	≤ 160	24
EE-F	EE (campeã)	Mutação gaussiana ($\sigma = 0,05$)	50	≤ 160	24

Diferentemente dos ciclos anteriores, executaram-se 24 neuroevoluções para cada algoritmo evolutivo ($n = 24$, na Tabela 4.2), o dobro da quantidade executada nos Ciclos 1 e 2. Esse incremento visa ampliar a quantidade de resultados por experimento, o que aumentará a qualidade da análise estatística dos resultados na Seção 4.2.

Além disso, neste ciclo, cada geração passou a ser formada de 50 partidas consecutivas, em contraste com as 25 partidas executadas por geração nos Ciclos 1 e 2. A duplicação do número de partidas reduz a variância no cálculo da aptidão dos indivíduos gerados nesse Ciclo.

A condição de parada também foi alterada. Adotaram-se duas condições de parada independentes, das quais basta o atendimento de uma para o encerramento da neuroevolução: completar 160 gerações ou alcançar aptidão média de um indivíduo igual ou superior a 550 mil. O valor de 550 mil de aptidão foi escolhido por representar uma quantidade de 1000 objetos superados durante a partida. Quanto ao número de gerações, escolheu-se 160 como o Número máximo de gerações desse ciclo, pois, em testes empíricos, estimou-se que essa quantidade de gerações seria neuroevoluída em, no máximo, 24 horas de execução.

Nas figuras deste ciclo, o eixo das abscissas exhibe valores de 1 a 160, com faixas posicionadas a cada geração múltipla de 15. O eixo das ordenadas, por sua vez, não tem escala fixa, mas variável de acordo com os resultados observados em cada experimento, como nos Ciclos 1 e 2.

O uso de faixas a cada 15 gerações no eixo das abscissas e a cada 50 mil de aptidão no eixo

das ordenadas facilita a leitura dos gráficos de linha, pois, com uma faixa vertical posicionada a cada 15 gerações e outra horizontal posicionada a cada 50 mil de aptidão, é possível determinar valores de aptidão atingidos por cada combinação de forma mais aproximada.

A Figura 4.15 apresenta gráficos de linha com a melhor aptidão por geração para cada uma das 24 neuroevoluções executadas com o AG campeão (AG-F). Nessa figura, o eixo das ordenadas exibe valores de 0 a 900000 de aptidão. O eixo das abscissas, por sua vez, exibe valores de 1 a 160, representativos das gerações executadas.

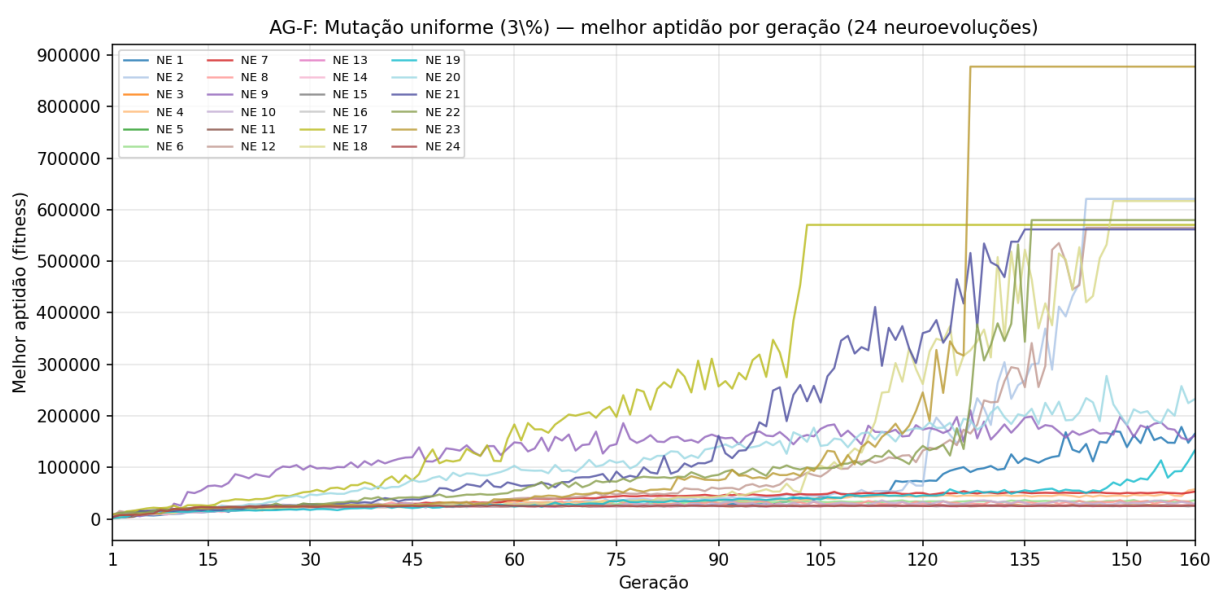


Figura 4.15: Gráficos de linha da melhor aptidão por geração (com 50 partidas/geração) para as 24 neuroevoluções realizadas no Ciclo 3 com o AG-F, a combinação campeã do Ciclo 1 (AG e mutação uniforme de 3%).

Pela Figura 4.15, nota-se que, do início até a geração de nº 30, apenas uma das curvas (NE 9) do AG-F foi capaz de atingir o patamar de 100000 de aptidão. As demais curvas, na 30ª geração, encontravam-se, em sua maioria, estagnadas próximas do valor de 25000 de aptidão.

Na 90ª geração, é possível identificar outras três curvas (NEs 17, 20 e 21) com valores de aptidão acima ou iguais a 100000. Até a geração 160 desse experimento de neuroevoluções, é possível identificar mais 7 curvas (NE 1, 2, 12, 18, 19, 22, 23) que superaram a faixa de estagnação e atingem valores acima de 100000, totalizando 11 curvas que foram capazes de superar a área de estagnação.

Durante as gerações 90 e 105, uma das curvas citadas anteriormente (NE 17) atinge o valor

de, aproximadamente 560000 de aptidão e supera a aptidão alvo definida como condição de parada. Como consequência, a criação de gerações novas para essa neuroevolução em específico é paralisada e, visualmente, a aptidão definida nas gerações restantes é a aptidão atingida na geração que superou a condição de parada. Até o fim desse experimento de neuroevoluções, outras 6 curvas também atingem a condição de parada por aptidão alvo. Essas curvas, suas aptidões finais aproximadas e a faixa de gerações que atingiram a condição de parada são exibidas na lista a seguir:

- Curva da neuroevolução 23 com aptidão final de 870000 entre as gerações 120 e 135.
- Curva da neuroevolução 21 com aptidão final de 555000 na geração 135.
- Curva da neuroevolução 22 com aptidão final de 570000 entre as gerações 135 e 150.
- Curva da neuroevolução 2 com aptidão final de 620000 entre as gerações 135 e 150.
- Curva da neuroevolução 12 com aptidão final de 555000 entre as gerações 135 e 150.
- Curva da neuroevolução 18 com aptidão final de 610000 entre as gerações 135 e 150.

A Figura 4.16 apresenta um gráfico de linha com a média e o primeiro desvio padrão da melhor aptidão por geração para as 24 neuroevoluções realizadas com o AG-F. Nessa figura, o eixo das ordenadas exibe valores de 0 a, aproximadamente, 500000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 100000.

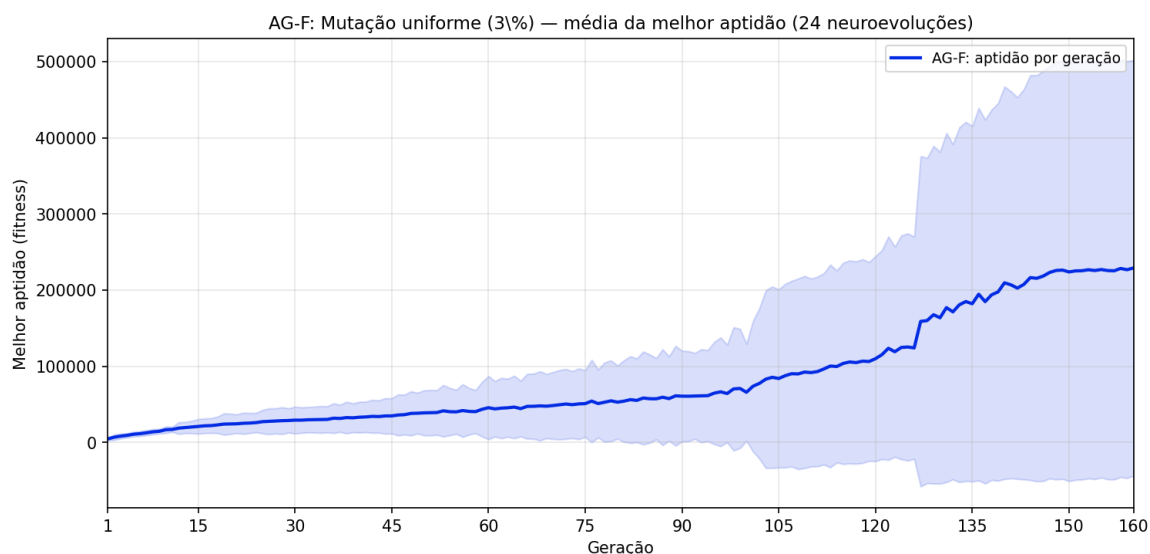


Figura 4.16: Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs do AG-F no Ciclo 3.

Pela Figura 4.16, observa-se que a média cresce de forma lenta até a geração 90. Próximo da geração 105, é notável uma aceleração no ritmo de crescimento e, na geração 120, a média da melhor aptidão das neuroevoluções para o AG-F atinge valores acima da faixa de 100000. Após a 120^a geração, é possível identificar outros pulos de aptidão que levam a média dos resultados do experimento com o AG-F superar a faixa seguinte de aptidão. Na geração 160, a geração final, a média dos resultados para o AG-F atinge o valor de, aproximadamente, 220000 de aptidão.

Quanto à faixa de um desvio padrão, essa exibe um crescimento moderado na primeira metade das gerações, mas, após a geração 100, expande-se de maneira acentuada. Na 160^a geração, a faixa abrange valores acima de 500000 em seu limite superior e valores negativos em seu limite inferior. Essa amplitude reflete a forte assimetria já visível na Figura 4.15, em que 11 NEs elevam a média muito acima dos resultados obtidos pelas demais curvas.

Em seguida, a Figura 4.17 apresenta gráficos de linha com a melhor aptidão por geração para cada uma das 24 neuroevoluções executadas com a EE campeã (EE-F). Nessa figura, o eixo das ordenadas exibe valores de 0 a 200000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 25000. O eixo das abscissas, como no experimento anterior, varia de 1 a 160 e apresenta faixas posicionadas a cada 15 gerações.

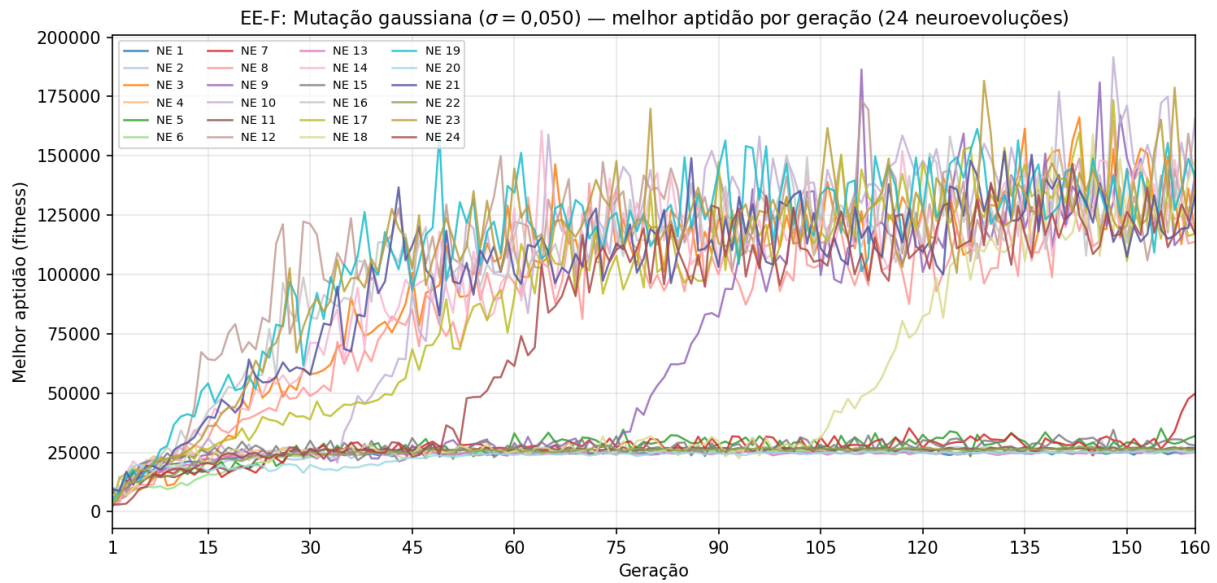


Figura 4.17: Gráficos de linha da melhor aptidão por geração (com 50 partidas/geração) para as 24 neuroevoluções realizadas no Ciclo 3 com a EE-F, a combinação campeã do Ciclo 2 (EE e mutação gaussiana de $\sigma = 0,05$).

Pela Figura 4.17, observa-se que, logo nas primeiras 30 gerações, o conjunto de curvas divide-se em dois grupos. Esses grupos se consolidam ao fim da faixa seguinte de gerações. O primeiro grupo (NEs 3, 8, 10, 12, 14, 17, 16, 19, 21, 23) eleva-se rapidamente nas gerações iniciais e, na geração 45, alcança valores entre 50000 e 125000 de aptidão. O segundo grupo, formado pelas curvas restantes, acompanha o crescimento universal das curvas nas primeiras 15 gerações, mas, em seguida, estagna próximo da faixa de 25000 de aptidão.

Até a geração final, outras três curvas (NEs 9, 18 e 24) se unem ao grupo 1, que na geração 160 apresenta valores de aptidão entre as faixas de 100000 e 175000. No grupo 2, as curvas finalizam suas neuroevoluções com aptidões, em sua maioria, na faixa de 25000.

A Figura 4.18 apresenta um gráfico de linha com a média e o primeiro desvio padrão da melhor aptidão por geração para as 24 neuroevoluções realizadas com a EE-F. Nessa figura, o eixo das ordenadas exibe valores de 0 a, aproximadamente, 150000 de aptidão e é dividido em faixas posicionadas a cada intervalo de 20000.

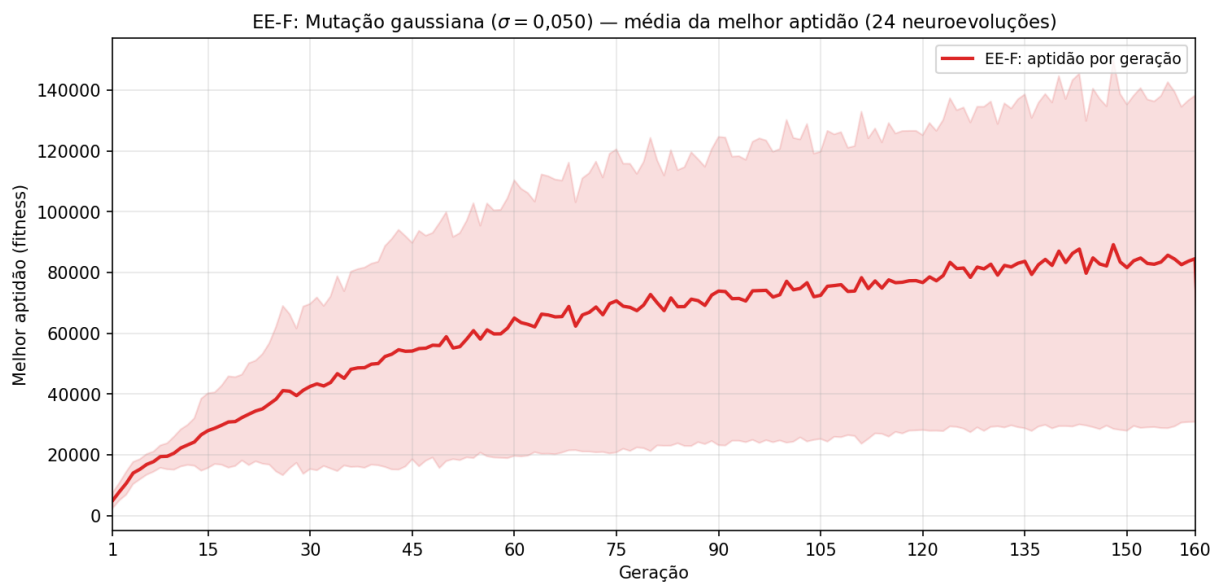


Figura 4.18: Gráfico de linha com a média e ± 1 desvio padrão do resultado da melhor aptidão por geração para as NEs da EE-F no Ciclo 3.

Pela Figura 4.18, observa-se que a média cresce de forma contínua ao longo das 160 gerações, atingindo cerca de 84000 de aptidão na geração final. A faixa de um desvio padrão expande-se de forma gradual e, na geração 160, abrange aptidões entre aproximadamente 30000 e 140000.

Finalmente, a Figura 4.19 apresenta a comparação direta entre as médias da melhor aptidão por geração obtidas pelo AG-F e pela EE-F. Nessa figura, o eixo das ordenadas exibe valores de 0 a 600000 de aptidão, com faixas posicionadas a cada 100000. O eixo das abscissas segue a mesma escala dos demais experimentos desse ciclo.

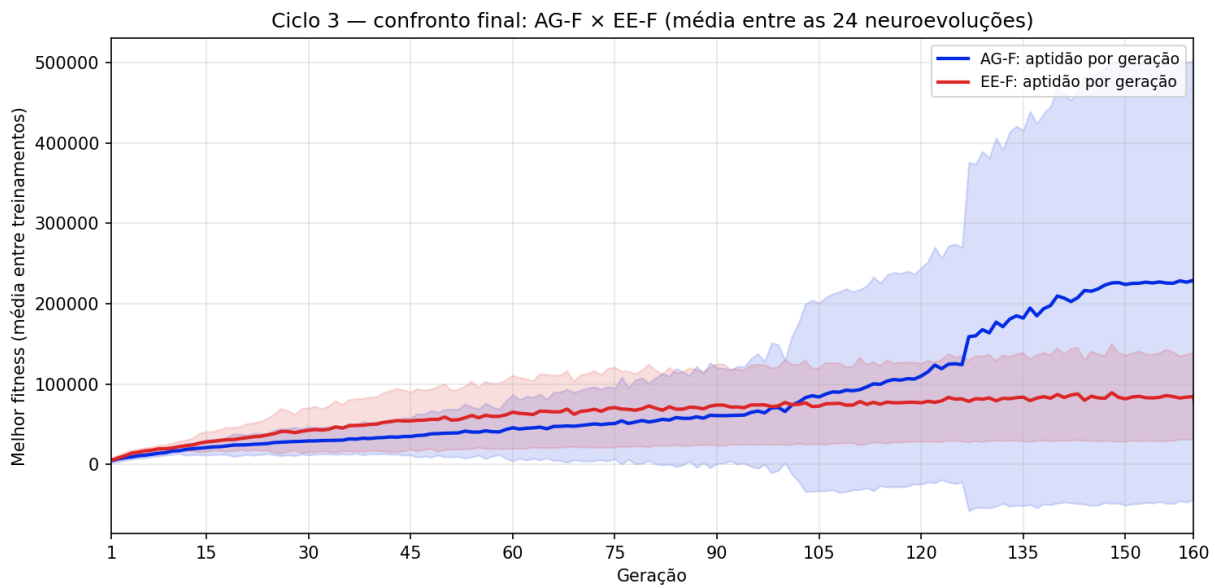


Figura 4.19: Comparação das curvas da média da melhor aptidão por geração (com ± 1 desvio padrão) entre AG-F e EE-F no Ciclo 3 ($n = 24$ neuroevoluções, 50 partidas/geração).

Na comparação direta exibida nessa Figura, nota-se que as médias do AG-F e da EE-F permanecem próximas até cerca da geração 15. Entre as gerações 15 e 95, a média da EE-F mantém-se ligeiramente superior à do AG-F, um efeito do ganho consistente de aptidão apresentado pelo grupo 1 de curvas da estratégia evolutiva. Por volta da geração 100, no entanto, a média do AG-F ultrapassa a da EE-F, impulsionada pelas NEs do algoritmo genético que atingiram aptidões acima da condição de parada por aptidão alvo. O AG-F encerra as gerações com uma média de aptidão em cerca de 220000 de aptidão, enquanto a EE-F exibe aproximadamente 80000.

4.2 Análise dos Resultados

Nessa seção, interpreta-se o confronto final do Ciclo 3 (Subseção 4.1.3), comparando as combinações AG-F e EE-F com base nas Figuras 4.16, 4.18 e 4.19. A análise considera também a forma das curvas de convergência, a dispersão entre as 24 neuroevoluções de cada combinação e as diferenças de protocolo em relação aos Ciclos 1 e 2.

Nos Ciclos 1 e 2, em que identificaram-se AG-3 e EE-050 como combinações campeãs, foram executadas 25 partidas por geração, 90 gerações e 12 neuroevoluções por experimento. O Ciclo 3

ampliou o protocolo para 50 partidas por geração, até 160 gerações, parada por aptidão alvo de 550 mil e 24 neuroevoluções por experimento.

Considerando o critério de comparação adotado nos ciclos anteriores, isto é, a média da melhor aptidão, o AG-F superou a EE-F na geração final (220 000 contra 80 000). No entanto, para verificar se essa diferença é estatisticamente significativa, foram aplicados testes de hipótese. Um teste de hipótese é um procedimento da estatística que avalia, a partir de uma amostra, se há evidência suficiente para rejeitar uma afirmação inicial, denominada hipótese nula (H_0), em favor de uma hipótese alternativa (H_1) (BUSSAB; MORETTIN, 2010). A decisão baseia-se no valor- p , que corresponde à probabilidade de se observar um resultado tão ou mais extremo que o obtido, supondo verdadeira a hipótese nula. Quando o valor- p é inferior ao nível de significância adotado (α), rejeita-se H_0 . Caso contrário, H_0 não é rejeitada (BUSSAB; MORETTIN, 2010).

Os testes de hipótese costumam ser classificados em paramétricos e não-paramétricos. Os testes paramétricos pressupõem que os dados seguem uma distribuição específica, comumente a distribuição normal, enquanto os testes não-paramétricos não exigem esse pressuposto, sendo utilizados quando a normalidade dos dados não pode ser assumida (BUSSAB; MORETTIN, 2010). Por esse motivo, antes de comparar as distribuições da melhor aptidão de AG-F e EE-F, verificou-se a hipótese de normalidade das distribuições dos resultados das NEs de cada experimento.

Para tanto, aplicou-se o teste de *Shapiro-Wilk*, cuja hipótese nula (H_0) afirma que a amostra provém de uma população com distribuição normal, enquanto a hipótese alternativa (H_1) indica o contrário. O teste produz a estatística W , a partir da qual se calcula o valor- p e se avalia a significância do resultado. A estatística W é limitada superiormente por 1 e mede o quanto os dados se aproximam de uma distribuição normal. Mais especificamente, valores próximos de 1 indicam maior aderência à normalidade, enquanto valores menores sugerem o distanciamento da normal (SHAPIRO; WILK, 1965). O teste foi aplicado, com nível de significância $\alpha = 0,05$, às aptidões finais das NEs de cada experimento. Os resultados desse teste são apresentados na Tabela 4.3.

Tabela 4.3: Resultado do teste de Shapiro-Wilk de normalidade para a melhor aptidão das NEs de AG-F e EE-F no Ciclo 3 ($n = 24$ por experimento, $\alpha = 0,05$).

Experimento	Estatística W	Valor- p	Normal? ($\alpha = 0,05$)
AG-F	0,7392	$< 0,0001$	não
EE-F	0,7918	0,0002	não

Para as neuroevoluções do AG-F, mais especificamente, encontrou-se a estatística W igual a 0,7392, o que resulta em valor- $p < 0,0001$. Para as NEs da EE-F, por sua vez, obteve-se $W = 0,7918$, cujo valor- $p = 0,0002$. Dessa forma, rejeita-se a hipótese da normalidade para as distribuições de ambos os experimentos, uma vez que o valor- p obtido para a distribuição dos resultados de ambos AG-F e EE-F foi inferior ao nível de significância adotado $\alpha = 0,05$.

Após constatada a ausência de normalidade nas distribuições, buscou-se averiguar se a diferença observada entre o AG-F e a EE-F é estatisticamente significativa. Para isso, optou-se por um teste não-paramétrico, aplicando-se o teste U de *Mann-Whitney*. Esse teste avalia se duas amostras independentes provêm de distribuições iguais (BUSSAB; MORETTIN, 2010). Ao invés de comparar diretamente as médias, esse teste ordena de forma conjunta todos os valores das duas distribuições, atribui a cada valor uma posição (também chamado de *rank* ou posto) e utiliza a soma dos postos de um dos grupos, a partir da qual se obtém a estatística U . A partir da estatística U , calcula-se o valor- p . O teste foi aplicado sobre os 24 resultados de melhor aptidão de cada experimento. Formularam-se, conseqüentemente, as hipóteses H_0 , de que as distribuições da melhor aptidão de AG-F e EE-F são iguais, e H_1 , de que essas distribuições diferem. Adotou-se o nível de significância $\alpha = 0,05$ e a abordagem bicaudal. Os valores obtidos nesse teste são apresentados na Tabela 4.4.

Tabela 4.4: Resultado do teste U de Mann-Whitney (bicaudal) para a comparação das distribuições da melhor aptidão obtida por AG-F e EE-F no Ciclo 3 ($n = 24$ por experimento, $\alpha = 0,05$).

Teste	Estatística U	Valor- p	Decisão ($\alpha = 0,05$)
Mann-Whitney U (bicaudal)	330,0	0,3922	não se rejeita H_0

Como o valor- p obtido (0,3922) é superior ao nível de significância adotado, não se rejeita H_0 , isto é, não há evidência estatística de que as distribuições dos resultados de AG-F e EE-F sejam

diferentes. Portanto, apesar dos resultados obtidos pelo experimento AG-F possuírem uma média cerca de 2,75 vezes maior que a média dos resultados do experimento EE-F, essa diferença não é estatisticamente significativa. É importante ressaltar, contudo, que a não-rejeição de H_0 não permite concluir que as duas abordagens sejam equivalentes (BUSSAB; MORETTIN, 2010). A ausência de evidência de diferença pode decorrer, por exemplo, do tamanho reduzido das amostras e da elevada dispersão observada nos resultados, e não necessariamente de uma real igualdade de desempenho entre AG-F e EE-F.

Os resultados sugerem, no entanto, que a combinação representada por AG-F, ou AG-3, favorece saltos de aptidão mais expressivos no ambiente experimentado por esse trabalho, atingindo os maiores valores individuais observados. A combinação representada por EE-F, ou EE-050, não alcançou esses picos, mas mostrou-se mais consistente entre as repetições, com um menor desvio padrão entre os resultados individuais de neuroevolução. A forte assimetria observada no AG-F, evidenciada pela amplitude do desvio padrão, indica que a escolha entre as abordagens depende do objetivo: o AG-F é preferível quando se busca o melhor resultado possível em alguma das execuções, enquanto a EE-F é mais adequada quando se prioriza a regularidade do desempenho.

Os achados deste trabalho podem ser posicionados em relação à literatura apresentada no Capítulo 2. O estudo de Pereira (2023), que também utilizou o *Chrome Dino* como ambiente para as neuroevoluções, restringiu-se apenas às Estratégias Evolutivas. O estudo também relatou alta eficácia dessa abordagem na Neuroevolução de Perceptrons Multicamadas (PEREIRA, 2023). Os resultados obtidos no presente trabalho complementam esse estudo ao comparar diretamente a EE com um AG sob o mesmo ambiente de experimentos. Além disso, demonstram que não houve diferença estatisticamente significativa entre os algoritmos evolutivos experimentados, embora o AG tenha conseguido atingir, em diversas neuroevoluções, picos de aptidão mais expressivos do que a EE.

De forma semelhante, Armanto, Setiabudi e Pickerling (2021) compararam o AG a outros algoritmos de otimização no jogo 2048 e observaram desempenhos próximos entre os métodos, sem explorar a variação de hiperparâmetros dos operadores genéticos. O presente trabalho

avança nesse ponto ao demonstrar que diferentes valores do hiperparâmetro de mutação alteram de forma relevante o desempenho de cada algoritmo. Esses resultados reforçam a importância do ajuste desses hiperparâmetros no projeto de métodos de Neuroevolução, posicionando a contribuição da presente pesquisa frente aos trabalhos correlacionados.

4.2.1 Limitações

Os resultados apresentados nesse capítulo devem ser interpretados considerando-se algumas limitações.

Do ponto de vista técnico, destacam-se a forte dispersão observada entre os resultados do Ciclo 3, o tamanho reduzido da amostra (24 neuroevoluções por experimento) e a adoção de uma arquitetura fixa de Rede Neural Artificial, o que impede generalizar as conclusões para outras topologias. Além disso, o conjunto de valores do hiperparâmetro de mutação testado foi limitado e definido majoritariamente de forma empírica. A comparação, por sua vez, restringiu-se a duas abordagens de algoritmos evolutivos (AG com Elitismo e EE (μ, λ)), não contemplando outras técnicas de Computação Evolucionária.

Há ainda limitações relacionadas ao ambiente experimental e aos recursos da pesquisa. Como estudo de caso, restringiu-se a apenas um único jogo, o *Chrome Dino*, cujas características, como a quantidade de ações possíveis de se executar a cada instante, podem não refletir o comportamento dos algoritmos em ambientes diferentes e mais complexos. Por fim, os recursos computacionais disponíveis também foram um fator limitante, uma vez que condicionaram o tempo total de experimentação e, consequentemente, restringiram o número de neuroevoluções, de partidas por geração e de gerações máximas executadas.

Considerações Finais

Este trabalho investigou qual abordagem evolutiva oferece melhor desempenho no treinamento de uma RNA para o jogo *Chrome Dino*: o AG canônico com Elitismo, que enfatiza a recombinação (HOLLAND, 1975), ou a EE (μ, λ) de Rechenberg, que prioriza a mutação (RECHENBERG, 1973). O estudo comparou também o impacto de diferentes valores do hiperparâmetro de mutação em cada algoritmo.

Cumpriram-se os objetivos específicos da Introdução: adotou-se a base *Dinossauro-Google*, estendida com as classes `GenAlgorithm` e `RechenbergMuLambdaES` e operadores modulares via padrão *Strategy* na classe `DinoRN`; automatizou-se a execução por *script Bash*; e geraram-se gráficos de convergência para análise comparativa.

Os experimentos organizaram-se em três ciclos. Nos Ciclos 1 e 2, com 12 neuroevoluções cada, identificaram-se as combinações do AG com mutação uniforme de 3% como campeã do AG e a EE com mutação gaussiana com desvio padrão $\sigma = 0,05$ como campeã da EE (Tabela 4.1; Figuras 4.7 e 4.14). No Ciclo 3, os campeões AG-3 e EE-050 foram reavaliados com 50 partidas por geração, até 160 gerações ou aptidão alvo de 550 mil (Tabela 4.2), com 24 neuroevoluções cada. O AG-F obteve a maior média (220 mil), porém com elevada dispersão; a EE-F apresentou média inferior (80 mil), no entanto, exibiu um desempenho mais regular entre as neuroevoluções (Figura 4.19). Os resultados evidenciaram que o AG campeão do Ciclo 1 alcança os maiores picos de aptidão, enquanto a EE campeã neuroevolui resultados mais consistentes. O teste U de Mann-Whitney bicaudal, com $\alpha = 0,05$, no entanto, não encontrou evidência significativa para afirmar que as distribuições de aptidão final dos dois experimentos sejam estatisticamente diferentes ($U = 330,0$, $p = 0,3922$). Esse resultado, entretanto, também não implica que as distribuições sejam equivalentes.

Por fim, o estudo evidencia que diferentes valores de mutação alteram o desempenho de cada algoritmo e que a combinação de AG, elitismo e mutação uniforme de 3% favoreceu ganhos de aptidão mais expressivos no *Chrome Dino* do que a combinação de EE e mutação gaussiana com desvio padrão de 0,05.

Como continuidade, sugere-se ampliar o número de repetições e o conjunto de hiperparâmetros testados, além de repetir a comparação em outros jogos e arquiteturas de rede.

Referências Bibliográficas

- ARMANTO, H.; SETIABUDI, K.; PICKERLING, C. Komparasi algoritma woa, mfo dan genetic pada optimasi evolutionary neural network dalam menyelesaikan permainan 2048. *Jurnal Inovasi Teknologi dan Edukasi Teknik*, v. 1, p. 1–9, 2021.
- BUSSAB, W. de O.; MORETTIN, P. A. *Estatística Básica*. 6. ed. São Paulo: Saraiva, 2010.
- BäCK, T.; SCHWEFEL, H.-P. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, v. 1, n. 1, p. 1–23, 03 1993. ISSN 1063-6560. Disponível em: <<https://doi.org/10.1162/evco.1993.1.1.1>>.
- CHEN, D. A. Neuro-evolução aplicada na conclusão de jogos de plataforma. 2022. Disponível em: <<http://repositorioinstitucional.uea.edu.br/handle/riuea/4015>>.
- CUNHA, A. G.; TAKAHASHI, R.; ANTUNES, C. H. *Manual de computação evolutiva e metaheurística*. [S.l.]: Imprensa da Universidade de Coimbra, and Editora da Universidade Federal de Minas Gerais, 2012.
- DANTZIG, G. B. *Linear Programming and Extensions*. Princeton University Press, 1963. Disponível em: <<https://archive.org/details/linearprogrammin0000dant/page/n5/mode/2up>>.
- DIAS, V. *GitHub - JVictorDias/Dinossauro-Google: Rede Neural aplicada ao jogo do Dinossauro do Google Chrome!* 2019. Github. Disponível em: <<https://github.com/JVictorDias/Dinossauro-Google>>. Acesso em: 29.03.2023.
- EIBEN, A. E.; SMITH, J. E. *Introduction to Evolutionary Computing*. 2nd. ed. [S.l.]: Springer, 2015.
- FOGEL, D. B. (Ed.). *Evolutionary Computation: the Fossil Record*. [S.l.]: IEEE Press, 1998.
- FromSoftware. *Dark Souls: Remastered*. 2018. Bandai Namco Entertainment. Video game, available on PlayStation 3, Playstation 4, Xbox 360, Xbox One, Nintendo Switch, and Windows.
- Google. *Navegador da web Google Chrome*. 2008. Disponível em: <<https://www.google.com/intl/pt-BR/chrome/>>. Acesso em: 30.06.2026.
- HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Prentice Hall, 2021.
- HINTERDING, R. Representation, constraint satisfaction and the knapsack problem. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. [S.l.: s.n.], 1999. v. 2, p. 1286–1292 Vol. 2.

- HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. 1. ed. [S.l.]: MIT Press, 1975.
- LUCAS, S. M. Evolving a neural network location evaluator to play ms. pac-man. *Proceedings of the 2005 IEEE Symposium on Computational Intelligence and Games (CIG05)*, 2005.
- MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. [S.l.]: Springer Berlin, Heidelberg, 1994.
- MILLINGTON, I. *ARTIFICIAL INTELLIGENCE FOR GAMES*. [S.l.]: Elsevier, 2006.
- MNIH, V. et al. Human-level control through deep reinforcement learning. *Nature*, 2015. Disponível em: <<https://doi.org/10.1038/nature14236>>.
- O'CONNOR, J.; PARKER, G. B.; BUGTI, M. *Learning Dark Souls Combat Through Pixel Input With Neuroevolution*. 2025. Disponível em: <<https://arxiv.org/abs/2507.03793>>.
- PARKER, M.; BRYANT, B. D. Neuro-visual control in the quake ii game engine. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. [S.l.: s.n.], 2008. p. 3828–3833.
- PEREIRA, J. V. D. Avaliação da eficácia das estratégias evolutivas no treinamento de redes neurais perceptron aplicadas a jogos digitais. 2023. Disponível em: <<https://www.universoprogramado.com/tcc.pdf>>. Acesso em: 22.02.2026.
- RECHENBERG, I. *Evolutionssstrategie: Optimierung technischer Systeme nach Prinzipien des biologischen Evolution*. [S.l.]: Friedrich Frommann Verlag, 1973.
- RISI, S.; TOGELIUS, J. *Neuroevolution in Games: State of the Art and Open Challenges*. 2015. Disponível em: <<https://arxiv.org/abs/1410.7326>>.
- ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Reviews*, 1957.
- RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 4th. ed. [S.l.]: Pearson, 2020. ISBN 0134610997, 9780134610993.
- SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, v. 52, n. 3/4, p. 591–611, 1965.
- SILVA, I. N. d.; SPATTI, D. H.; FLAUZINO, R. A. *Redes neurais artificiais para engenharia e ciências aplicadas*. [S.l.]: Artliber Editora, 2016.
- STANLEY, K. O.; MIIKKULAINEN, R. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, v. 10, n. 2, p. 99–127, 2002.
- TASVIDEOS. *Bizhawk*. 2017. Disponível em: <<https://tasvideos.org/Bizhawk>>. Acesso em: 26.05.2026.
- TOGELIUS, J. et al. *Evolutionary Machine Learning and Games*. Springer Nature Singapore, 2024. 715–737 p. ISBN 978-981-99-3814-8. Disponível em: <https://doi.org/10.1007/978-981-99-3814-8_25>.
- YANNAKAKIS, G. N.; TOGELIUS, J. *Artificial Intelligence and Games*. Springer, 2018. Disponível em: <<http://gameaibook.org>>.