



CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

ABORDAGENS EVOLUCIONÁRIAS PARA APRENDIZADO AUTÔNOMO EM REDES NEURO-*Fuzzy* PREDITIVAS

GLENDER BRÁS DE MEDEIROS

Orientador: Alisson Marques da Silva
CEFET-MG

Coorientadora: Elizabeth Fialho Wanner
CEFET-MG

BELO HORIZONTE
AGOSTO DE 2025

GLENDER BRÁS DE MEDEIROS

**ABORDAGENS EVOLUCIONÁRIAS PARA
APRENDIZADO AUTÔNOMO EM REDES
NEURO-*Fuzzy* PREDITIVAS**

Tese apresentada ao Programa de Pós-graduação em Modelagem Matemática e Computacional do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito parcial para a obtenção do título de Doutor em Modelagem Matemática e Computacional.

Área de concentração: Modelagem Matemática e Computacional

Linha de pesquisa: Sistemas Inteligentes

Orientador: Alisson Marques da Silva
CEFET-MG

Coorientadora: Elizabeth Fialho Wanner
CEFET-MG

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL
BELO HORIZONTE
AGOSTO DE 2025

M488a Medeiros, Glender Brás
Abordagens evolucionárias para aprendizado autônomo em redes neuro-fuzzy
preditivas / Glender Brás de Medeiros. – 2025.
92 f.

Tese (doutorado) apresentada ao Programa de Pós-Graduação em Modelagem
Matemática e Computacional.

Orientador: Alisson Marques da Silva.

Coorientadora: Elizabeth Fialho Wanner.

Tese (doutorado) – Centro Federal de Educação Tecnológica de Minas Gerais.

1. Redes neurais (Ciência da computação) – Teses. 2. Sistemas difusos –
Teses. 3. Aprendizagem de máquina – Teses. 4. Computação autônoma – Teses.
5. Computação evolutiva – Teses. I. Silva, Alisson Marques da. II. Warner,
Elizabeth Fialho. III. Centro Federal de Educação Tecnológica de Minas Gerais.
IV. Título.

CDD 006.3



SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

“Abordagens Evolucionárias para Aprendizado Autônomo em Redes Neuro-Fuzzy Preditivas”.

Tese de Doutorado apresentada por **Glender Brás de Medeiros**, em 26 de agosto de 2025, ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, e aprovada pela banca examinadora constituída pelos professores:

Prof. Dr. Alisson Marques da Silva (Orientador)
Centro Federal de Educação Tecnológica de Minas Gerais



Documento assinado digitalmente

MARCOS FLAVIO SILVEIRA VASCONCELOS D'ANGELO
Data: 08/12/2025 20:46:55-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. André Paim Lemos
Mercado Livre



Documento assinado digitalmente

ANDRE PAIM LEMOS
Data: 03/12/2025 08:51:41-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Marcos Flávio Silveira Vasconcelos D'Angelo
Universidade Estadual de Montes Claros



Documento assinado digitalmente

FLAVIO VINICIUS CRUZEIRO MARTINS
Data: 28/11/2025 11:03:32-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Flávio Vinicius Cruzeiro Martins
Centro Federal de Educação Tecnológica de Minas Gerais



Documento assinado digitalmente

DENIS EMANUEL DA COSTA VARGAS
Data: 27/11/2025 19:10:26-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Dênis Emanuel da Costa Vargas
Centro Federal de Educação Tecnológica de Minas Gerais

Visto e permitida à impressão,

Prof. Dr. Alisson Marques da Silva
Coordenador do Programa de Pós-Graduação Stricto Sensu em
Modelagem Matemática e Computacional

Agradecimentos

Agradeço primeiramente a Deus, pelo dom da vida e por todas as graças concedidas, das quais, sem dúvidas, a conclusão desta Tese é uma das maiores. Por ter me sustentado, me amado e tornado possível a realização e conclusão de mais esta etapa.

À minha família por sempre me apoiar e acreditar em mim.

A todos os meus amigos que torceram por mim e acompanharam essa jornada que não foi nada fácil.

Aos colegas e amigos do MMC por dividirem essa jornada comigo, por toda partilha de conhecimento e por também terem sido meu apoio quando necessário.

À família EJC, que me proporcionou momentos incríveis e que certamente me serviram de suporte nestes últimos anos.

Muito especialmente, agradeço aos meus orientadores, Alisson e Beth, pela condução deste trabalho, por terem me orientado do início ao fim mesmo com muitas dificuldades no caminho, por me apoiarem quando precisei, me entenderem quando precisei e por me cobrarem à medida quando foi necessário. Se não fosse por vocês, provavelmente eu não teria chegado até aqui.

Ao CEFET-MG por toda a formação acadêmica e profissional, deste o curso técnico, e por todas as oportunidades.

À CAPES e ao CEFET-MG pelo apoio financeiro.

"Os desafios não existem para nos paralisar, mas para nos ajudar a descobrir quem realmente somos." (Bernice Johnson Reagon)

"Conheça todas as teorias, domine todas as técnicas, mas ao tocar uma alma humana, seja apenas outra alma humana." (Carl Jung)

Resumo

Este trabalho propõe novas abordagens evolucionárias para o aprendizado autônomo em redes neuro-fuzzy. São apresentados três novos modelos neuro-fuzzy evolucionários: um linguístico (FALCON-GA) e dois funcionais (EvoNFuzz e DGP-NF). O FALCON-GA utiliza um algoritmo genético para a extração de regras e um método baseado em gradiente para o ajuste dos parâmetros. Já o EvoNFuzz e o DGP-NF empregam programação genética multi-gene para a modelagem das regras funcionais. Esses modelos apresentam uma estrutura mais próxima das redes neurais, incorporando pesos sinápticos aos consequentes das regras. Para o ajuste de parâmetros, o EvoNFuzz utiliza um método baseado em gradiente, enquanto o DGP-NF adota um método de aprendizagem incremental baseado em mínimos quadrados ponderados. Além disso, o DGP-NF introduz mecanismos voltados à melhoria da exploração do espaço de busca, como população dinâmica, controle de diversidade populacional e um operador de poda baseado na análise de diversidade dos genes, com o objetivo de eliminar regras redundantes. Tais mecanismos são fundamentados na diversidade fenotípica, uma vez que o principal objetivo é aumentar a qualidade das soluções geradas pela programação genética e aprimorar as respostas fornecidas pelo modelo final. Experimentos computacionais em tarefas de regressão, conduzidos em 21 conjuntos de dados, sugerem que as abordagens propostas são promissoras e que os modelos desenvolvidos são eficientes para esse tipo de problema. Destaca-se o DGP-NF, que obteve os melhores resultados em 16 conjuntos e apresentou desempenho competitivo nos demais. Os resultados também indicam que o método de aprendizagem incremental supera o método baseado em gradiente no ajuste de parâmetros em redes neuro-fuzzy evolucionárias, além de apontarem que redes funcionais são mais eficazes do que as linguísticas em problemas de regressão. O uso de mecanismos como população dinâmica e controle de diversidade demonstrou ser eficiente para evitar a convergência prematura e melhorar a acurácia dos modelos. Além disso, a poda baseada na diversidade fenotípica mostrou-se uma estratégia simples e eficaz para eliminar regras redundantes, por meio da poda de indivíduos multi-gene.

Palavras-chave: Redes neuro-*fuzzy*, aprendizado autônomo, sistemas *fuzzy* evolucionários, computação evolucionária.

Abstract

This work proposes new evolutionary approaches for autonomous learning in neuro-fuzzy networks. Three new neuro-fuzzy models are presented: one linguistic (FALCON-GA) and two functional (EvoNFuzz and DGP-NF). In FALCON-GA, a genetic algorithm is used for rule extraction, and a gradient-based method is used for parameter adjustment. The EvoNFuzz and DGP-NF models use multi-gene genetic programming to model functional rules. They also present a structure closer to neural networks, incorporating synaptic weights into the rule consequents. For parameter adjustment, EvoNFuzz uses a gradient-based method, while DGP-NF implements an incremental learning method based on weighted least squares. Furthermore, DGP-NF incorporates mechanisms to improve the search space exploration, such as dynamic population and population diversity control, and a pruning operator based on gene diversity analysis to eliminate redundant rules. The proposed mechanisms are based on phenotypic diversity, aiming to improve the quality of solutions generated by genetic programming and the responses provided by the final model. Experimental results on 21 nonlinear regression datasets suggest that the proposed approaches are promising and that the proposed models are efficient for this type of problem, with DGP-NF standing out as the best performer on 16 datasets and competitive on the remaining datasets. The results also suggest that the incremental learning method outperforms the gradient method for parameter tuning in evolutionary neuro-fuzzy networks and that functional networks are more efficient than linguistic networks for regression problems. The use of dynamic population and diversity control mechanisms proved effective in avoiding premature convergence and improving model accuracy. Furthermore, phenotypic diversity-based pruning has demonstrated potential as a simple and effective strategy for pruning multi-gene individuals and eliminating redundant rules.

Keywords: *Neuro-fuzzy networks, autonomous learning, evolutionary fuzzy systems, evolutionary computation.*

Lista de Figuras

Figura 1 – Fluxo de execução de um algoritmo evolucionário.	7
Figura 2 – Exemplo de indivíduo no algoritmo genético.	8
Figura 3 – Exemplo de indivíduos representados por árvores de sintaxe.	9
Figura 4 – Exemplo de operação de cruzamento no algoritmo genético.	11
Figura 5 – Exemplo de operação de cruzamento na programação genética.	12
Figura 6 – Exemplo de operação de mutação no algoritmo genético.	12
Figura 7 – Exemplo de operação de mutação na programação genética.	13
Figura 8 – Sistema <i>fuzzy</i> genético baseado em dados.	20
Figura 9 – Estrutura da rede FALCON.	26
Figura 10 – Função de pertinência gaussiana.	29
Figura 11 – Representação dos indivíduos no FALCON-GA.	31
Figura 12 – Cruzamento no FALCON-GA.	33
Figura 13 – Mutação no FALCON-GA.	34
Figura 14 – Fluxograma do algoritmo FALCON-GA.	35
Figura 15 – Estrutura da rede EvoNFuzz.	39
Figura 16 – Indivíduo multi-gene no EvoNFuzz.	41
Figura 17 – Cruzamento no EvoNFuzz.	43
Figura 18 – Mutação no EvoNFuzz.	43
Figura 19 – Fluxograma do algoritmo de aprendizagem da rede EvoNFuzz.	48
Figura 20 – Estrutura da rede DGP-NF.	52
Figura 21 – Indivíduo multi-gene no DGP-NF.	55
Figura 22 – Cruzamento no DGP-NF.	56
Figura 23 – Mutação no DGP-NF.	56
Figura 24 – Fluxograma do algoritmo de aprendizagem da rede DGP-NF.	64
Figura 25 – Impacto da população dinâmica.	73
Figura 26 – Impacto do controle de diversidade.	73
Figura 27 – Diferença percentual em relação ao tamanho da população = 50 sem controle de diversidade.	74
Figura 28 – Diferença percentual em relação ao tamanho da população = 50 com controle de diversidade.	75
Figura 29 – Análise de convergência.	76
Figura 30 – Tamanho da base de regras com e sem poda.	78

Lista de Tabelas

Tabela 1 – Conjuntos de dados utilizados. k = Número de amostras; n = número de atributos; X_{rg} = faixa dos valores de entrada; Y_{rg} = faixa dos valores de saída; X_{sd} = Desvio padrão da entrada; Y_{sd} = Desvio padrão da saída. .	66
Tabela 2 – Valores testados no <i>Grid Search</i>	67
Tabela 3 – Comparação dos 8 preditores em 21 conjuntos de dados: RMSE médio das 21 execuções, e resultados do Teste-T pareado.	69
Tabela 4 – Comparação entre o DGP-NF, versão sem população dinâmica (sem pop. din.) e versão sem diversidade (sem div.).	72
Tabela 5 – Comparação entre DGP-NF e a versão sem poda.	77

Lista de Algoritmos

Algoritmo 1 – Geração das funções de pertinência.	30
Algoritmo 2 – Geração da população inicial	32
Algoritmo 3 – Cálculo da saída da rede.	32
Algoritmo 4 – Algoritmo de aprendizado do FALCON-GA.	36
Algoritmo 5 – Geração das funções de pertinência - EvoNFuzz.	45
Algoritmo 6 – Geração da população inicial - EvoNFuzz.	46
Algoritmo 7 – Cálculo da saída da rede - EvoNFuzz.	46
Algoritmo 8 – Algoritmo de aprendizado do EvoNFuzz.	49
Algoritmo 9 – Ajuste do tamanho da população	59
Algoritmo 10 – Poda do indivíduo	60
Algoritmo 11 – Geração das funções de pertinência - DGP-NF.	61
Algoritmo 12 – Geração da população inicial - DGP-NF.	62
Algoritmo 13 – Cálculo da saída da rede - DGP-NF.	62
Algoritmo 14 – Algoritmo de aprendizado do DGP-NF.	63

Lista de Abreviaturas e Siglas

AB	<i>AdaBoost</i>
AG	Algoritmo Genético
ANFIS	<i>Adaptive Neuro-Fuzzy Inference System</i>
ART	<i>Adaptive Resonance Theory</i>
DGP-NF	<i>Dynamic Genetic Programming-based Neuro-Fuzzy Network</i>
DT	<i>Decision Tree</i>
EvoNFuzz	<i>Evolutionary Neuro-Fuzzy Network</i>
FALCON	<i>Fuzzy Adaptive Learning Control Network</i>
FALCON-GA	<i>Fuzzy Adaptive Learning Control Network with Genetic Algorithm</i>
GB	<i>Gradient Boosting</i>
IA	Inteligência Artificial
IRL	<i>Iterative Rule Learning</i>
MLP	<i>Multilayer Perceptron</i>
MSE	<i>Mean Square Error</i>
NFN	<i>Neo-Fuzzy-Neuron</i>
NFN-PG-MG	<i>Neo-Fuzzy-Neuron com Programação Genética Multi-Gene</i>
NSI	<i>Neural-symbolic Integration</i>
PG	Programação Genética
PGMG	Programação Genética Multi-Gene
RF	<i>Random Forest</i>
RMSE	<i>Root Mean Square Error</i>
TS	Takagi-Sugeno

Lista de Símbolos

A	conjunto <i>fuzzy</i> de entrada
c	centro das funções de pertinência
$\bar{c}$	centroide dos grupos
D	grau de disparo do antecedente k de uma regra
E	distância euclidiana
g	número de gerações
H	matriz de atualização dos parâmetros
i	índice das variáveis de entrada
j	índice das funções de pertinência de entrada
k	número final de clusters do k-means
k_{max}	máximo de clusters do k-means
k_{min}	mínimo de clusters do k-means
L^1	saída do componente de fuzzificação
L^2	saída do componente de inferência
L^3	saída do componente de agregação
m	número de funções de pertinência de uma variável de entrada
MR	matriz de regras
n	número de variáveis de entrada
N	número de amostras
n_{out}	número de variáveis de saída
n_r	número total de regras
O	conjunto <i>fuzzy</i> de saída
q	número de regras ativadas para um consequente

r	índice das regras <i>fuzzy</i>
s	espalhamento das funções de pertinência
T	tamanho da população
TP	número de testes pareados
tx_c	taxa de cruzamento
tx_m	taxa de mutação
u_{ij}	saída da camada 2
w	peso sináptico
W	parâmetro ajustável das camadas ou neurônios
x	vetor de entrada do modelo
Y_k	consequente estimado para a regra k
$\hat{y}$	saída do modelo
z	número de regras
α	nível de significância do teste estatístico
Δ	taxa de variação da população
ϵ	tamanho mínimo da população
η	taxa de aprendizado
λ	limiar inferior de diversidade
$\mu_{A_{ij}}$	grau de associação à i -ésima variável de entrada na j -ésima função de pertinência para o conjunto <i>fuzzy</i> A
σ	limiar superior de diversidade
ω	tamanho máximo da população

Sumário

1 – Introdução	1
1.1 Motivação e Relevância	1
1.2 Objetivos: Geral e Específicos	3
1.3 Contribuições	4
1.3.1 Publicações	4
1.4 Organização do Trabalho	4
2 – Revisão Bibliográfica	6
2.1 Computação Evolucionária	6
2.1.1 Algoritmos Genéticos	8
2.1.2 Programação Genética	9
2.1.3 Métodos de Seleção	10
2.1.4 Operadores Genéticos	11
2.2 Redes Neuro-Fuzzy	13
2.2.1 Sistemas Fuzzy	15
2.2.2 Aprendizado e Extração de Regras	16
2.2.3 Sistemas Fuzzy Evolucionários	19
3 – FALCON-GA - Fuzzy Adaptive Learning Control Network with Genetic Algorithm	24
3.1 Revisitando a Rede FALCON	25
3.2 Estrutura da Rede FALCON-GA	28
3.3 Funções de Pertinência	29
3.4 Algoritmo Genético para Extração de Regras	30
3.4.1 Representação dos indivíduos	31
3.4.2 Geração da população inicial	31
3.4.3 Avaliação da população	32
3.4.4 Geração da nova população	33
3.5 Atualização dos Parâmetros	34
3.6 Algoritmo de Aprendizado do FALCON-GA	34
4 – EvoNFuzz - Evolutionary Neuro-Fuzzy Network	37
4.1 Estrutura da Rede EvoNFuzz	38
4.2 Modelagem da Base de Regras com Programação Genética	41
4.2.1 Representação do indivíduo e operadores genéticos	41
4.3 Processo de Aprendizagem do EvoNFuzz	44

5 – DGP-NF - <i>Dynamic Genetic Programming-based Neuro-Fuzzy Network</i>	50
5.1 Estrutura da Rede DGP-NF	51
5.2 Modelagem da Base de Regras com Programação Genética	54
5.2.1 Representação do indivíduo e operadores genéticos	54
5.3 População Dinâmica e Controle de Diversidade	56
5.4 Poda Baseada em Diversidade Fenotípica	58
5.5 Processo de Aprendizagem do DGP-NF	60
6 – Experimentos Computacionais	65
6.1 Metodologia	65
6.2 Resultados Experimentais	68
6.3 Ablação - População Dinâmica e Controle de Diversidade	70
6.4 Ablação - Estratégia de Poda	75
7 – Considerações Finais	79
7.1 Conclusão	79
7.2 Trabalhos Futuros	81
Referências	82

Capítulo 1

Introdução

Neste capítulo inicial, apresenta-se uma contextualização sobre redes neuro-*fuzzy*, destacando alguns dos desafios enfrentados em seu desenvolvimento, bem como as principais técnicas de aprendizagem utilizadas. A motivação e a relevância deste trabalho para a área são discutidas na Seção 1.1. A Seção 1.2 expõe o objetivo geral e os objetivos específicos a serem alcançados. As principais contribuições e as publicações resultantes deste estudo são descritas na Seção 1.3. Por fim, a Seção 1.4 apresenta a estrutura do trabalho e o conteúdo abordado em cada um dos capítulos seguintes.

1.1 Motivação e Relevância

Em um cenário de constante evolução tecnológica, que lida com sistemas e dados cada dia mais dinâmicos, o estudo de técnicas eficientes para a construção de sistemas autônomos mostra-se cada vez mais importante e desafiador para a área de Inteligência Artificial (IA) (GU et al., 2023). Esses sistemas devem ser capazes de simular ações, respostas ou até raciocínio humano para a resolução de diversos problemas do mundo real e para o auxílio na tomada de decisão, extraíndo seu conhecimento a partir de dados e sem interferência humana em seu aprendizado (WATSON; SCHEIDT, 2005; HAGOS; RAWAT, 2022). Desta forma, as redes neuro-*fuzzy* tornam-se uma ferramenta poderosa e bastante popular na área de IA, devido à sua propriedade inerente de representação de conhecimento incerto e à capacidade de adaptação a contextos dinâmicos (SOUZA, 2020a). As redes neuro-*fuzzy* integram o processo de raciocínio e inferência, capaz de tratar imprecisões e incertezas, dos sistemas *fuzzy* dentro de uma estrutura conexionista com alta habilidade de aprendizado e generalização das redes neurais (SHIHABUDHEEN; PILLAI, 2018).

A combinação entre essas duas técnicas reduz efetivamente o espaço de busca e o tempo necessário para se chegar a uma solução ótima, o que faz com que esses modelos sejam ferramentas poderosas na solução de diversos tipos de problemas, como classificação (XUE et al., 2022; VIVEKANANDHAN; SAKTHIVEL; MANIKANDAN, 2022; RAVICHANDRAN;

KEIKHOSROKIANI, 2023; ABRO et al., 2025), previsão (AHMADIANFAR et al., 2022; GAO et al., 2024; SALAMEH et al., 2025; GÜLMEZ, 2025), controle (RAO et al., 2022; KHANAFARI et al., 2024; ELBORLSY et al., 2025; GIL et al., 2025), tomada de decisão (DEVECI et al., 2022; TEŠIĆ et al., 2023; CHANDER; DAS, 2024; MEHMOOD et al., 2025), entre outros.

Uma das vantagens das redes neuro-*fuzzy* é o aprendizado adaptativo, que permite que os parâmetros da rede sejam ajustados dinamicamente a partir de mudanças nas condições de operação, sem a necessidade de alterar a base de regras (COX, 2002). A modelagem dessas redes envolve essencialmente três etapas: definição dos conjuntos *fuzzy* por meio do particionamento do universo de discurso; determinação da estrutura do modelo (que envolve também a estrutura das regras *fuzzy*); e ajuste dos parâmetros da rede (MAMAGHANI; PEDRYCZ, 2020). Assim, a extração de regras a partir de dados é uma tarefa essencial, permitindo obter sistemas adaptativos, autônomos e com boa capacidade de generalização (TOWNSEND; CHATON; MONTEIRO, 2019). Esta etapa consiste em identificar as regras mais adequadas para compor a base de regras de um sistema *fuzzy*, por meio da união da estrutura das redes neurais com modelos relacionais simbólicos, utilizando diversas técnicas para extrair e mapear relações entre dados de entrada e saída (GARCEZ et al., 2022).

Devido à importância dessa tarefa, diversas técnicas têm sido exploradas para definição da base de regras e para o ajuste dos parâmetros em redes neuro-*fuzzy*, incluindo métodos de *backpropagation* (ASHOKKUMAR; MOHANBABU, 2020; CHEN et al., 2021) e agrupamento (PUSTOKHINA et al., 2021; RAJAN et al., 2023). Porém os métodos convencionais de aprendizado enfrentam dificuldades como falta de estabilidade, dificuldade de lidar com dependências de longo prazo em séries temporais, dificuldade de escapar de ótimos locais, além de tornar a busca pela solução ótima exaustiva, devido ao fato de serem técnicas baseadas em busca local (SHI; MIZUMOTO, 2000; Min Han et al., 2004). Com isso, a utilização de técnicas híbridas para aprendizado nesse tipo de sistema, especialmente de métodos baseados em busca global, tem despertado um crescente interesse nas últimas décadas. Dentre esses métodos, destacam-se as técnicas evolucionárias, que se distinguem de outras heurísticas principalmente pela sua capacidade de escapar de ótimos locais sem a necessidade de explorar todo o espaço de busca (MOGHADDAM, 2024).

Técnicas evolucionárias são técnicas populacionais, ou seja, consideram uma população de soluções (indivíduos) que possuem seu aprendizado baseado em processos análogos aos da evolução biológica dos seres vivos, como cruzamento, mutação e seleção natural (BABU B. V.; GUJARATHI, 2017). Essas técnicas são interessantes pelo fato de serem livres de derivadas, não necessitando de suposição de diferenciabilidade. Ou seja, requerem apenas a disponibilidade de valores da função objetivo, mas nenhuma informação derivada,

tornando a modelagem matemática dos algoritmos muito mais simples. As principais abordagens evolucionárias encontradas na literatura são algoritmos genéticos (HOLLAND, 1975), evolução diferencial (STORN; PRICE, 1997), estratégias evolutivas (BEYER; SCHWEFEL, 2002) e programação genética (KOZA, 1992). Dessa forma, as redes *fuzzy* evolucionárias combinam algoritmos evolucionários com lógica *fuzzy* para melhorar sua adaptabilidade e desempenho, particularmente para problemas com amplo espaço de busca ou em cenários que exigem atualizações contínuas de regras (FERNANDEZ et al., 2019).

Neste contexto, este trabalho propõe o uso de abordagens evolucionárias para o desenvolvimento de novos algoritmos voltados ao aprendizado autônomo em redes neuro-*fuzzy*, com aplicação em tarefas de previsão e identificação de sistemas. Tais técnicas podem ser combinadas com outros métodos de aprendizado, visando a construção de sistemas mais autônomos e genéricos, que dispensam a necessidade de modelagem por especialistas. Dessa forma, tornam-se soluções totalmente orientadas a dados, capazes de lidar com uma ampla variedade de problemas.

1.2 Objetivos: Geral e Específicos

O objetivo geral deste trabalho é propor e implementar novas abordagens evolucionárias para a modelagem autônoma de redes neuro-*fuzzy* com boa acurácia e capacidade de generalização.

Os objetivos específicos são:

- Propor e implementar algoritmos de aprendizado para redes neuro-*fuzzy* autônomas baseados em técnicas evolucionárias, integradas a técnicas de *backpropagation* e de aprendizagem incremental;
- Analisar a diferença de aplicabilidade das técnicas evolucionárias em modelos linguísticos e funcionais, bem como a aplicação desses modelos a problemas de regressão;
- Propor e implementar técnicas para controle de diversidade, populacional e melhor exploração do espaço de busca em algoritmos evolucionários, especialmente em programação genética, baseadas em fenótipo;
- Propor e implementar um novo operador de poda para bases de regras *fuzzy*, utilizando análise de diversidade fenotípica;
- Validar as abordagens propostas em problemas de previsão e identificação de sistemas não lineares.

1.3 Contribuições

A principal contribuição deste trabalho está no desenvolvimento de novos modelos neuro-fuzzy evolucionários autônomos, capazes de obter toda a estrutura da rede a partir de dados e com menor necessidade de estruturas pré-definidas, além do desenvolvimento e da exploração de técnicas para uma maior exploração do espaço de busca em algoritmos evolucionários, que também podem ser aplicadas em outros contextos. A utilização de técnicas evolucionárias visa favorecer o aprendizado nesses sistemas por meio de uma exploração ampla e global do espaço de busca. Por serem métodos populacionais, pode-se obter algoritmos de treinamento eficientes, com menor número de iterações e capazes de escapar de ótimos locais. As contribuições específicas de cada modelo desenvolvido serão discutidas nos capítulos referentes a eles.

1.3.1 Publicações

Durante a realização deste trabalho, foram desenvolvidos/publicados os seguintes artigos:

- Brás, G.; Silva, A. M.; Wanner, E. F. **A Genetic Algorithm for Rule Extraction in Fuzzy Adaptive Learning Control Networks**. *Genetic Programming and Evolvable Machines* (periódico). 2024. **Publicado**.
- Brás, G.; Wanner, E. F.; Silva, A. M.; **Uma abordagem híbrida com aprendizagem incremental para modelagem de redes neuro-fuzzy evolucionárias**. XVII Simpósio Brasileiro de Automação Inteligente (SBAI). 2025. **Publicado**.
- Brás, G.; Silva, A. M.; Wanner, E. F. **EvoNFuzz: A New Evolutionary Neuro-Fuzzy Network with Genetic Programming-based Learning**. The EPIA 2025 International Conference. 2025. **Publicado**.
- Brás, G.; Martins, F.V.C.; Wanner, E. F.; Silva, A. M. **Incremental Evolutionary Neuro-Fuzzy System with Dynamic Population and Diversity Control**. *Evolutionary Intelligence* (periódico). 2025. **Submetido**.

1.4 Organização do Trabalho

O restante deste trabalho está dividido da seguinte maneira:

- No Capítulo 2 é apresentada uma revisão bibliográfica, percorrendo sobre os conceitos básicos tratados neste trabalho e apresentando uma revisão de trabalhos relacionados.
- O Capítulo 3 apresenta a rede FALCON-GA, composta por regras linguísticas e que utiliza um algoritmo genético para extração de regras e um método baseado no gradiente para ajuste dos parâmetros;

- O Capítulo 4 apresenta a rede EvoNFuzz, uma rede neuro-*fuzzy* funcional composta por regras polinomiais, cujo aprendizado é baseado em programação genética para modelagem da base de regras e em um método baseado no gradiente para ajuste dos parâmetros.
- O Capítulo 5 apresenta a rede DGP-NF, uma rede neuro-*fuzzy* funcional composta por regras polinomiais que emprega um método de programação genética com população dinâmica e controle de diversidade para modelagem da base de regras, além de um método de aprendizagem incremental baseado em mínimos quadrados ponderados para ajuste dos parâmetros.
- O Capítulo 6 descreve os experimentos realizados para validação dos modelos propostos e discute os resultados alcançados.
- O Capítulo 7 traz as considerações finais e perspectivas de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Este capítulo apresenta uma revisão bibliográfica sobre os principais temas tratados neste trabalho. Na Seção 2.1, os principais conceitos da área de computação evolucionária são discutidos, com foco em algoritmo genético, que foi utilizado para extração de regras no modelo apresentado no Capítulo 3, e programação genética, que foi empregada para modelagem de regras funcionais nos modelos apresentados nos capítulos 4 e 5. A Seção 2.2 apresenta os fundamentos de sistemas *fuzzy* e neuro-*fuzzy*, além de uma revisão da literatura recente sobre redes neuro-*fuzzy* e sistemas *fuzzy* evolucionários, com ênfase nas técnicas de aprendizado aplicadas nesses contextos.

2.1 Computação Evolucionária

Computação evolucionária é a área da computação inspirada nos processos de evolução natural dos seres vivos, onde o indivíduo mais forte tende a sobreviver. A computação evolucionária utiliza analogias ao processo de evolução, como seleção natural, cruzamento e mutação, para gerar e evoluir uma população de soluções, em que cada indivíduo (solução) compete pela sobrevivência (KELLER DERONG LIU, 2016). As definições e princípios básicos de um algoritmo evolucionário são:

- **Indivíduo:** é a representação de uma solução para o problema. Pode variar de acordo com o problema, e é de extrema importância escolher a representação adequada para o que está sendo tratado.
- **Gene:** são as informações que compõem um indivíduo. Por exemplo, se um indivíduo é representado por um vetor binário, cada índice do vetor representa um gene.
- **População:** é um conjunto de soluções para o problema. O tamanho da população (por quantas soluções será composta) é pré-definido, e é interessante que existam soluções boas e ruins em uma mesma população para favorecer o processo evolutivo.
- **Fitness:** é a aptidão de uma solução; é uma função que determina o quão boa uma solução é. Quanto maior a *fitness*, melhor é a solução.

- **Cruzamento:** combinação de duas soluções (soluções pai) para formar uma nova solução (solução filha). Essencial para o processo de melhoria das soluções, já que os pais com melhor aptidão (*fitness*) têm maior probabilidade de se reproduzir.
- **Mutação:** é o processo que altera algum gene do indivíduo aleatoriamente. Esse processo tem como objetivo inserir diversidade na população. Quanto maior a diversidade, maior é o espaço de busca que está sendo explorado.
- **Seleção natural:** é o processo de escolha dos indivíduos que passarão para a próxima população. Esse processo é probabilístico, e os indivíduos com maior *fitness* possuem maior chance de sobreviver para próxima população. Acontece após o cruzamento (geração de filhos) e a mutação.

Na computação evolucionária o processo inicia com a geração de uma população inicial, normalmente aleatória, em que será gerada uma quantidade determinada de indivíduos. A cada geração, os indivíduos são avaliados conforme a *fitness* e os melhores têm mais chances de serem selecionados pelo operador de seleção natural, que será aplicado em dois momentos: na seleção de pais que serão submetidos às operações genéticas e na seleção dos indivíduos que sobreviverão para a próxima geração. Como todo esse processo é probabilístico, corre-se o risco de perder o melhor indivíduo entre uma geração e outra. Por isso, a maioria das abordagens atualmente utiliza o mecanismo de elitismo, que consiste em sempre salvar o melhor indivíduo para a próxima população antes de executar o operador de seleção natural. Todos esses passos são repetidos até atender a um critério de parada, que pode ser um valor pré-definido de *fitness* para o melhor indivíduo ou um número determinado de gerações (BABU B. V.; GUJARATHI, 2017). O fluxo geral de um algoritmo evolucionário pode ser visto na Figura 1.

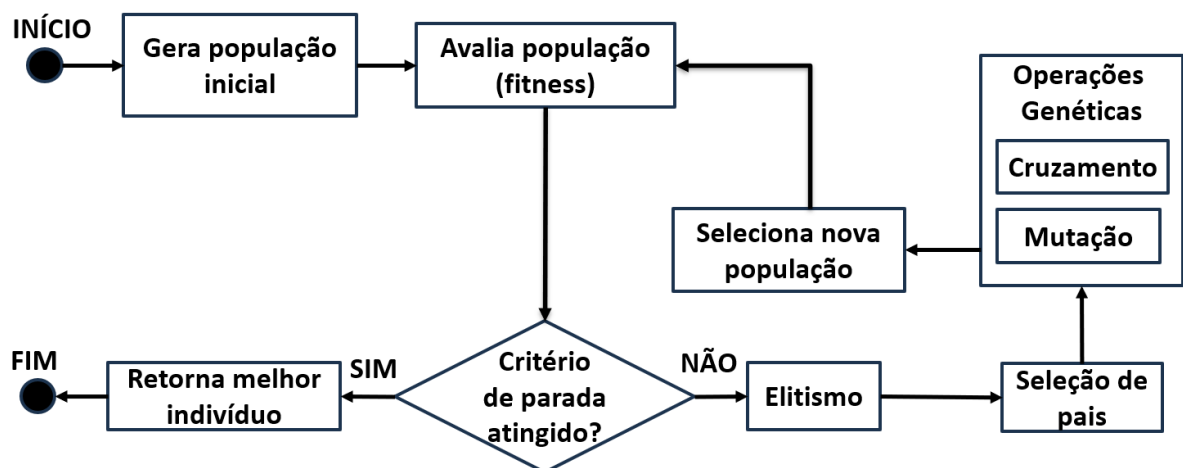


Figura 1 – Fluxo de execução de um algoritmo evolucionário.

As principais técnicas utilizadas nessa área são algoritmos genéticos, programação genética, estratégias evolutivas e evolução diferencial. Todas seguem os mesmos princípios, com

algumas diferenças na representação dos indivíduos ou em algum detalhe do processo evolutivo do algoritmo. Enquanto, no algoritmo genético, os indivíduos são normalmente vetores simples, em que cada um representa diretamente uma solução para determinado problema, na programação genética trabalha-se com uma população de programas computacionais que levam à solução final. Por outro lado, as estratégias evolutivas e a evolução diferencial têm sua principal diferença no processo de evolução das soluções. Enquanto, na primeira, os pais sobrevivem até que gerem filhos necessariamente melhores, na segunda, as operações de cruzamento e mutação são realizadas a partir de perturbações nos indivíduos, em que cada casal gera um novo filho em um local diferente do espaço de busca.

2.1.1 Algoritmos Genéticos

Propostos por [Holland \(1975\)](#), os algoritmos genéticos (AG) são algoritmos de busca global baseados na evolução natural dos seres vivos. Por ser um método populacional e evolucionário, o AG pode encontrar boas soluções em um amplo espaço de busca com menos passos que outros algoritmos. Como qualquer outra técnica evolucionária, permite encontrar a solução para um problema sem que o algoritmo seja explicitamente programado para isso ([JONG, 2017](#)). Isso torna o método genérico, flexível e altamente adaptável para resolver problemas distintos.

Os AGs consideram uma população de soluções (indivíduos) que passam por evoluções a cada nova geração (iteração). Os indivíduos nos algoritmos genéticos são normalmente representados por *strings* binárias, nas quais cada *bit* (gene) corresponde a um parâmetro do espaço de busca ([KRAMER, 2017](#)). Assim, cada *string* (indivíduo) representa um ponto no espaço de busca, ou seja, uma possível solução para o problema. Um exemplo de indivíduo no AG pode ser visto na Figura 2. A figura ilustra um indivíduo representado como uma *string* binária composta por 8 genes, na qual os genes 1, 2, 3, 6 e 8 possuem o *bit* 1, enquanto os genes 4, 5 e 7 possuem o *bit* 0. O significado dos valores contidos nos genes depende do problema em questão. Por exemplo, em um sistema *fuzzy*, esse indivíduo poderia representar uma base de regras, em que o *bit* 1 indica que a regra correspondente àquela posição está incluída na base, e o *bit* 0 indica que a regra está excluída.

1 1 1 0 0 1 0 1

Figura 2 – Exemplo de indivíduo no algoritmo genético.

Nos AGs a população inicial é normalmente gerada de forma aleatória, e as próximas gerações são formadas a partir dos operadores de cruzamento (recombinação), mutação e seleção natural ([KELLER DERONG LIU, 2016](#)). A cada geração, os indivíduos são avaliados conforme uma função de aptidão, e os melhores têm mais chances de serem escolhidos pelo método de seleção. A cada iteração, uma nova população é gerada. Na geração

da nova população, pode-se substituir todos os indivíduos da última geração pelos filhos gerados (modo geracional) ou utilizar um método de seleção que considere a *fitness* para determinar quais indivíduos devem sobreviver para a próxima geração. Em ambos os casos, corre-se o risco de perder o melhor indivíduo, já que os processos são probabilísticos. Para resolver esse problema, é possível utilizar o elitismo, mecanismo que garante que o melhor indivíduo seja mantido para a próxima população (LAMBORA; GUPTA; CHOPRA, 2019). Todos esses passos são repetidos até que se atenda a um critério de parada, que pode ser um valor pré-definido de *fitness* para o melhor indivíduo ou um número determinado de gerações.

2.1.2 Programação Genética

Na programação genética (PG) (KOZA, 1992), os indivíduos são considerados como "programas computacionais" e são representados como árvores de sintaxe interpretáveis (LANGDON; POLI, 2013), compostas por funções (nós internos) e terminais (nós folha). A Figura 3 mostra dois exemplos de indivíduos em forma de árvores de representação, onde cada árvore representa uma função matemática. Esta função matemática pode ser uma solução para o problema ou uma função que resolve o problema, por exemplo. Em um sistema *fuzzy* funcional, cada indivíduo poderia representar uma função de consequente.

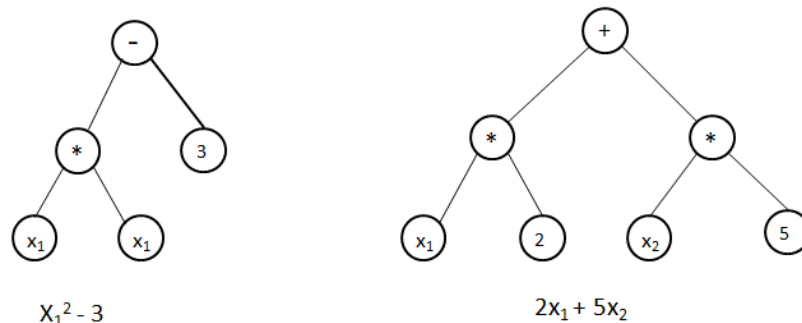


Figura 3 – Exemplo de indivíduos representados por árvores de sintaxe.

A PG surgiu com o intuito de trazer uma abordagem mais ampla aos métodos evolucionários, em que o termo "programa" pode ser entendido de diversas formas, como, por exemplo, expressões matemáticas, nas quais as funções são operações aritméticas e os terminais são as variáveis e constantes do problema, ou como uma sequência de instruções, em que as funções são os operadores e os terminais são os atributos. Dessa forma, a PG se torna um método bastante genérico e sem necessidade de grandes adaptações para trabalhar com problemas distintos.

Devido à facilidade em representar funções matemáticas, conforme citado anteriormente, a PG tem se demonstrado uma poderosa ferramenta para regressão simbólica de funções, como discutido por Oliveira et al. (2015) e Virgolin et al. (2021). Segundo Koza (1994),

em problemas de Inteligência Artificial, a representação mais natural para uma solução é um programa computacional, isto é, uma composição hierárquica de funções primitivas e terminais com tamanho variável, ao contrário de representações em cadeias simples de tamanho fixo, utilizadas por outras abordagens, que tornam a representação mais difícil, não natural e excessivamente restritiva. Uma representação eficiente de PG exige três propriedades básicas para os indivíduos gerados (KOZA, 1992):

- **Suficiência:** a expressão representada pelo indivíduo deve ser suficiente para representar uma solução candidata para o problema em questão. Ou seja, todos os indivíduos devem representar uma solução candidata completa para o problema.
- **Fechamento:** uma função deve aceitar como entrada qualquer saída produzida por qualquer outro elemento do conjunto de funções ou do conjunto de terminais.
- **Parcimônia:** conter apenas funções realmente necessárias para resolver o problema em questão. Esta propriedade é desejável, mas não obrigatória.

2.1.3 Métodos de Seleção

A seleção nos algoritmos evolucionários é o processo análogo à seleção natural na teoria da evolução dos seres vivos e consiste em escolher um indivíduo baseado em sua *fitness* para realizar uma operação (RUSSELL; NORVIG, 2016). Ocorre em dois momentos: seleção de pais e seleção da nova população e, por ser baseada na *fitness*, permite que indivíduos mais aptos tenham mais chances de serem selecionados. Na seleção de pais, são escolhidos pares de indivíduos que podem se reproduzir, ou seja, gerar soluções filhas. Esse processo é probabilístico, isto é, existe uma probabilidade de cada par se reproduzir ou não. Já na seleção natural, são selecionados os indivíduos que sobreviverão para a próxima geração, que podem ser tanto pais quanto filhos. Em ambos os casos, indivíduos com maior *fitness* têm mais chance de serem selecionados, porém também há a possibilidade de indivíduos menos aptos serem escolhidos, o que é importante para o algoritmo, uma vez que ajuda a manter a diversidade da população, o que pode favorecer o processo de evolução do algoritmo como um todo.

Os dois métodos mais comuns são a seleção por torneio e a roleta (EIBEN; SMITH et al., 2003). Na seleção por torneio, uma quantidade determinada de indivíduos é sorteada para o torneio e o mais apto vence, sendo escolhido como primeiro pai. Realiza-se outro torneio, e é escolhido o segundo pai. O sorteio é randômico, mas o pai escolhido será sempre o mais apto entre os sorteados. Já na seleção por roleta, os indivíduos são dispostos no espaço (roleta) proporcional à sua *fitness*, de forma que os que tiverem maior *fitness* ocupem uma fatia maior da roleta. É feito o sorteio do indivíduo selecionado a partir do espaço definido pelas aptidões.

2.1.4 Operadores Genéticos

Para evoluir, ou seja, melhorar a qualidade das soluções geradas, são aplicados os operadores genéticos para modificá-las durante o processo evolucionário. Os operadores normalmente utilizados são o cruzamento e a mutação.

O cruzamento é um operador essencial no processo evolutivo, cujo objetivo é gerar novas soluções, denominadas filhos, a partir da combinação de duas soluções existentes, conhecidas como pais. Parte-se do pressuposto de que, ao combinar informações genéticas de dois indivíduos, há uma elevada probabilidade de que o indivíduo resultante herde características vantajosas de ambos, aumentando assim a chance de obter soluções melhores. Esse operador contribui para a convergência do algoritmo genético (AG) em regiões mais promissoras do espaço de busca. No AG, o cruzamento é realizado por meio da troca de informações genéticas entre os dois indivíduos pais, como ilustrado na Figura 4, que mostra uma operação de cruzamento no ponto 5. Nesse exemplo, são gerados dois filhos com genes recombinados: o primeiro filho herda os genes do primeiro pai até o ponto 5 e os genes restantes do segundo pai, enquanto o segundo filho herda os genes do segundo pai até o ponto 5 e os genes restantes do primeiro pai.

1	1	0	1	1	1	0	1	Pai 1
0	1	0	0	1	0	1	1	Pai 2
1	1	0	1	1	0	1	1	Filho 1
0	1	0	0	1	1	0	1	Filho 2

Figura 4 – Exemplo de operação de cruzamento no algoritmo genético.

Já na programação genética (PG), o cruzamento é realizado por meio da troca de subárvores entre dois indivíduos, como mostrado na Figura 5. Para isso, seleciona-se aleatoriamente um nó em cada uma das árvores dos pais. Em seguida, a subárvore cuja raiz corresponde ao nó escolhido no pai 1 é trocada com a subárvore correspondente do pai 2. Como resultado, são gerados dois novos indivíduos (filhos), cujas estruturas combinam partes das árvores originais dos pais. A figura ilustra esse processo, apresentando duas árvores iniciais (pais) e os filhos resultantes após a substituição de subárvores, evidenciando como a recombinação de estruturas pode gerar expressões matemáticas ou lógicas diferentes.

A mutação, por sua vez, consiste na alteração de alguma informação genética de um indivíduo previamente selecionado, promovendo maior diversidade na população e evitando que os novos indivíduos sejam formados apenas a partir das informações dos pais. Esse mecanismo é fundamental para a exploração mais ampla do espaço de busca. A Figura 6 apresenta um exemplo de mutação em um algoritmo genético clássico, no qual um dos

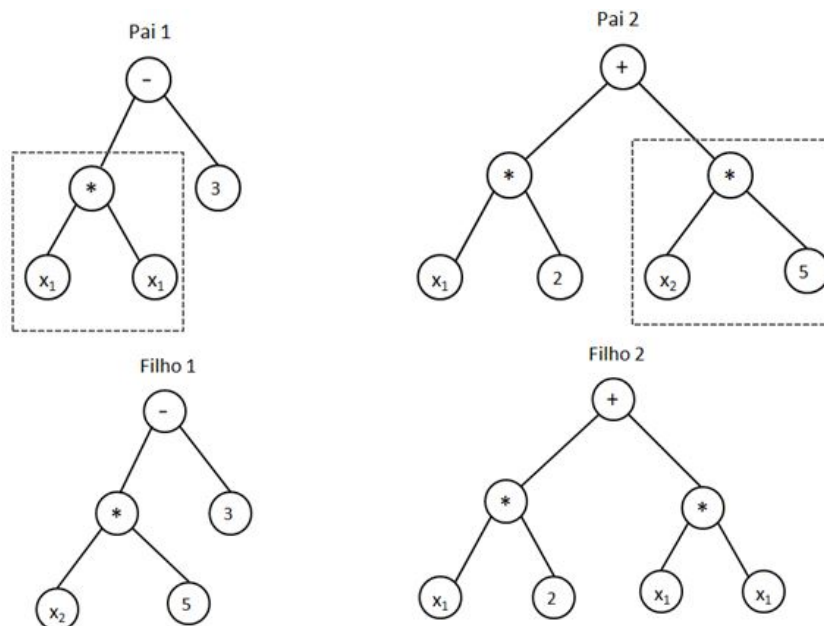


Figura 5 – Exemplo de operação de cruzamento na programação genética.

genes de um indivíduo é alterado. Especificamente, a mutação ocorre no gene 4, onde o *bit* original 0 é substituído por 1, exemplificando a modificação direta de um ponto na estrutura binária do indivíduo.

0	1	0	0	1	1	0	1	Antes da mutação
0	1	0	1	1	1	0	1	Após mutação

Figura 6 – Exemplo de operação de mutação no algoritmo genético.

Na PG, a mutação pode ocorrer de diferentes formas. Pode-se, por exemplo, substituir uma subárvore ou nó selecionado por outra subárvore ou nó gerados aleatoriamente. Também é possível trocar uma função por outra função ou terminal, ou ainda substituir um terminal por outro terminal ou função. A Figura 7 mostra um exemplo prático de mutação na PG: uma subárvore foi selecionada e substituída por um terminal, o que altera de forma significativa a estrutura e o comportamento do indivíduo, destacando o potencial desse operador em transformar as expressões representadas pelas árvores.

Os operadores de cruzamento e mutação são aplicados com base em probabilidades definidas previamente pelo usuário. Em geral, o cruzamento é aplicado com alta frequência, entre 70% e 90%, enquanto a mutação possui taxas menores, em torno de 10% (ENGELBRECHT, 2007; KACPRZYK; PEDRYCZ, 2015). No entanto, na programação genética, a mutação tende a desempenhar um papel ainda mais relevante, contribuindo significativamente para

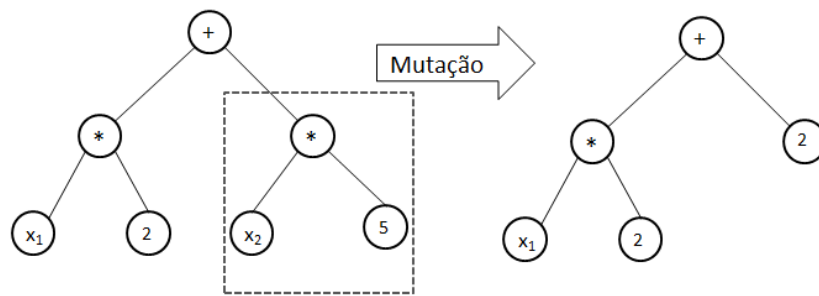


Figura 7 – Exemplo de opera  o de muta  o na programac  o gen tica.

a diversidade da popula  o e para a explora  o do espa o de busca. Por isso,   comum utilizar taxas de muta  o mais elevadas, frequentemente superiores a 10% ou at  mesmo 20% (POLI; KOZA, 2013; BURLACU; YANG; AFFENZELLER, 2024).

2.2 Redes Neuro-Fuzzy

As redes neuro-*fuzzy* combinam caracter sticas das redes neurais e dos sistemas *fuzzy* com o objetivo de integrar a capacidade de lidar com dados lingu sticos e imprecisos, t pica dos sistemas *fuzzy*,   habilidade de aprendizagem e generaliza  o das redes neurais (SHIHABUDHEEN; PILLAI, 2018). Essa integra  o permite superar as limita  es de cada abordagem isoladamente (ABRAHAM, 2001). Os sistemas *fuzzy* dependem da defini  o manual de regras e da divis  o dos conjuntos *fuzzy* para modelar sua base de conhecimento, pois n o possuem mecanismos de aprendizagem aut noma. Por outro lado, as redes neurais s o eficientes na aprendizagem de padr es a partir de dados, por m carecem de interpretabilidade e n o conseguem lidar diretamente com informa  es lingu sticas e imprecisas (CZOGALA; LESKI, 2012). Assim, observa-se que as vantagens e desvantagens das duas t cnicas s o complementares.

O principal desafio no desenvolvimento de abordagens h bridas, em especial nas redes neuro-*fuzzy*, est  na constru  o de um sistema que possua tanto capacidade de generaliza  o quanto um processo de racioc nio bem definido. Para isso, diversas t cnicas de aprendizado de m quina s o empregadas com o intuito de automatizar e tornar esse processo de modelagem mais eficiente. A coopera  o entre redes neurais e sistemas *fuzzy* tem se mostrado bastante eficaz para superar dificuldades relacionadas   constru  o da base de regras e ao ajuste de par metros (ABRAHAM, 2001). Desta forma, espera-se que uma rede neuro-*fuzzy* eficiente apresente caracter sticas como aprendizado r pido, adaptabilidade, autoajuste para minimiza  o do erro global e baixa complexidade computacional.

O funcionamento computacional dessas redes inicia-se com o desenvolvimento de neur nios *fuzzy*, inspirados nos neur nios biol gicos, e posteriormente com a incorpora  o de t cnicas

de aprendizado (FULLÉR, 2000; KRUSE et al., 2016). Esse processo pode ser dividido em três etapas: (i) desenvolvimento de modelos neurais *fuzzy* baseados em neurônios biológicos; (ii) definição de modelos de conexões sinápticas que incorporam incertezas à rede; e (iii) desenvolvimento de algoritmos de aprendizado voltados ao ajuste dos pesos sinápticos.

Existem, basicamente, duas formas de modelagem para redes neuro-*fuzzy*. Na primeira, o módulo *fuzzy* recebe variáveis linguísticas e as converte em um vetor de valores numéricos, que são então processados pelo módulo neural. Este realiza o aprendizado e fornece a saída do sistema. Na segunda abordagem, o módulo neural processa diretamente os dados de entrada e aciona, posteriormente, o mecanismo de inferência *fuzzy*, que contém a base de regras.

Os conjuntos de regras e demais parâmetros dos sistemas neuro-*fuzzy* podem ser obtidos por meio de métodos de aprendizado que extraem conhecimento a partir dos dados. Os principais métodos encontrados na literatura são:

- **Métodos de aprendizado do tipo gradiente:** técnica baseada na descida em direção à região mais íngreme da vizinhança, ou seja, que proporciona melhor resultado.
- **Técnicas híbridas:** combinam duas ou mais técnicas de aprendizado como *backpropagation*, mínimos quadrados, algoritmos de agrupamento, entre outros.
- **Técnicas populacionais:** envolvem algoritmos genéticos, evolução diferencial, algoritmos de enxame, colônias de formigas, entre outros.
- **ELM (*Extreme Learning Machine* - Máquina de Aprendizagem Extrema):** técnica de rede neural onde os parâmetros dos neurônios ocultos são definidos aleatoriamente enquanto os dos neurônios de saída são definidos analiticamente baseado no método dos mínimos quadrados.
- **SVM (*Support Vector Machine* - Máquina de Vetor de Suporte):** algoritmos de classificação que tem sua decisão guiado por pontos vetoriais de suporte. É uma das principais técnicas de classificação na área de aprendizado de máquina.

Essas técnicas e os principais trabalhos que as aplicam no aprendizado de redes neuro-*fuzzy* são discutidos por Shihabudheen e Pillai (2018). Os principais modelos neuro-*fuzzy* descritos na literatura podem ser agrupados em três categorias, conforme apresentado por Abraham (2001): modelos cooperativos, concorrentes e totalmente fundidos. Nos modelos cooperativos, a rede neural é utilizada em uma etapa de pré-processamento, para definir automaticamente as funções de pertinência e as regras *fuzzy* a partir dos dados de treinamento. Após essa etapa, a inferência passa a ser responsabilidade do módulo *fuzzy*, com a rede neural assumindo papel secundário. Nos modelos concorrentes, o módulo neural participa ativamente, auxiliando continuamente na definição dos parâmetros do sistema *fuzzy*. Essa abordagem é útil especialmente quando os dados de entrada não podem

ser diretamente medidos ou quando as saídas *fuzzy* não são diretamente aplicáveis ao processo. Nesse caso, o módulo neural pode funcionar como um pós-processador das saídas *fuzzy*. Já nas arquiteturas totalmente fundidas, os algoritmos de aprendizado da rede neural são utilizados diretamente para definir os parâmetros do sistema de inferência *fuzzy*. Nesses casos, as redes compartilham estrutura de dados e representação do conhecimento. Uma forma comum de implementar essa fusão é representar o sistema *fuzzy* dentro da arquitetura da rede neural, ou seja, criar neurônios com características *fuzzy*, conforme discutido anteriormente.

2.2.1 Sistemas *Fuzzy*

Sistemas baseados em regras têm como objetivo modelar o comportamento e a tomada de decisão humana na resolução de problemas, por meio de um processo de raciocínio construído a partir de regras do tipo *IF-THEN* (VARSHNEY; TORRA, 2023). Essas regras são compostas por antecedentes e consequentes, e baseiam-se na teoria dos conjuntos *fuzzy* introduzida por Zadeh (1965). Quando o antecedente é satisfeito, a regra é ativada e o consequente correspondente é estimado. Esse tipo de sistema é capaz de lidar com informações linguísticas, incertas e imprecisas, aproximando-se da forma como os humanos expressam conhecimento e tomam decisões (KACPRZYK; PEDRYCZ, 2015).

Dois tipos principais de sistemas *fuzzy* baseados em regras são encontrados na literatura: os modelos linguísticos e os modelos funcionais (FULCHER, 2008). Os sistemas linguísticos, baseados no modelo proposto por Mamdani Mamdani e Assilian (1975), representam tanto os antecedentes quanto os consequentes das regras por meio de conjuntos *fuzzy*. Uma das principais vantagens desse tipo de sistema é a sua semelhança com a linguagem humana, o que facilita o processamento de informações imprecisas tanto nas variáveis de entrada quanto nas de saída, permitindo capturar com maior fidelidade as incertezas presentes nos dados (ZADEH, 1975). As regras linguísticas seguem uma estrutura típica, como:

$$\begin{array}{l} \text{SE } x_1 \text{ é } A_{1,1} \text{ E } x_2 \text{ é } A_{1,3} \text{ E } x_3 \text{ é } A_{1,2} \text{ ENTÃO } y_1 \text{ é } O_3 \\ \text{SE } x_1 \text{ é } A_{2,2} \text{ E } x_2 \text{ é } A_{2,4} \text{ E } x_3 \text{ é } A_{2,1} \text{ ENTÃO } y_2 \text{ é } O_2 \end{array}$$

na qual A representa os conjuntos *fuzzy* das variáveis de entrada, e O , os conjuntos *fuzzy* das variáveis de saída.

A saída desses sistemas é obtida por meio do processo de defuzzificação, que transforma os valores *fuzzy* estimados durante a inferência em valores precisos (*crisp*). Diversos métodos de defuzzificação são utilizados na literatura, tais como:

- **Método min-max:** a regra com maior força de disparo é selecionada e determina qual função de consequente é ativada.
- **Método da média:** é calculada a média da força de ativação de cada regra de forma que todas as regras ativadas tenham influência na saída.
- **Método raiz-soma-quadrado:** cada função de pertinência é dimensionada de modo que o pico da função seja igual à força máxima de disparo que corresponde a essa função.
- **Método do centro de gravidade recortado:** cada função de pertinência é recortada nas forças de ativação da regra correspondentes.

Por sua vez, os modelos funcionais, baseados no modelo de Takagi-Sugeno (TS) proposto por [Takagi e Sugeno \(1985\)](#), utilizam conjuntos *fuzzy* nos antecedentes, enquanto os consequentes são definidos por funções matemáticas $f(x)$ das variáveis de entrada. A principal vantagem dessa abordagem é a sua capacidade de representar relações não lineares por meio de modelos locais lineares, permitindo uma modelagem mais precisa de sistemas dinâmicos e complexos ([JOHANSEN; SHORTEN; MURRAY-SMITH, 2000](#)). Devido a essa característica, os modelos Takagi-Sugeno são amplamente utilizados como base em redes neuro-*fuzzy* e são considerados ferramentas eficazes para a modelagem de sistemas não lineares ([MATÍA; MARICHAL; JIMÉNEZ, 2014](#)). As regras funcionais seguem, por exemplo, o seguinte formato:

$$\begin{array}{llllll} \text{SE} & x_1 \text{ é } A_{1,1} & \text{E} & x_2 \text{ é } A_{1,3} & \text{E} & x_3 \text{ é } A_{1,2} & \text{ENTÃO} & y_1 \text{ é } 2x_1 + 3x_2 + x_3 \\ \text{SE} & x_1 \text{ é } A_{2,2} & \text{E} & x_2 \text{ é } A_{2,4} & \text{E} & x_3 \text{ é } A_{2,1} & \text{ENTÃO} & y_2 \text{ é } 5x_1 + x_2 + 3x_3 \end{array}$$

Para expandir as capacidades dos sistemas *fuzzy* funcionais que, como mencionado, utilizam funções lineares locais para determinar o consequente das regras, os modelos Takagi-Sugeno foram aprimorados para incluir consequentes polinomiais ([LAM, 2011](#)), que são estimados por expressões matemáticas mais complexas que podem assumir formatos diversos. As regras polinomiais permitem a modelagem de relações mais complexas entre as variáveis de entrada e saída, aumentando sua precisão, especialmente para sistemas e variáveis com relações não lineares.

2.2.2 Aprendizado e Extração de Regras

Uma das principais vantagens das redes neuro-*fuzzy* é seu aprendizado adaptativo, que permite que os parâmetros de rede sejam ajustados dinamicamente a partir de mudanças nas condições de operação, sem necessidade de alterar a base de regras ([COX, 1993](#)). Ao se reorganizar internamente conforme a necessidade, esses sistemas tornam-se ideais para tratar problemas do mundo real que possuem um dinamismo característico. Na

prática, esses sistemas podem, ao contrário dos sistemas *fuzzy* tradicionais, ajustar seu comportamento de uma execução para outra, sem necessariamente alterar sua base de regras, sendo este seu grande diferencial (PEDRYCZ; GOMIDE, 2007). A cada fluxo de execução, um sistema adaptativo permite alterar uma variedade de estruturas internas da rede, incluindo itens fundamentais como a granularização dos conjuntos *fuzzy*, a interpretação de uma determinada regra, região de ativação e os pesos associados às regras. Desta forma podemos dizer que os sistemas *fuzzy* adaptativos são mais sofisticados e possuem parâmetros adaptáveis, ou seja, são mais flexíveis que os sistemas *fuzzy* tradicionais sem perder a capacidade de possuir um processo de raciocínio explicável.

O processo de aprendizado nas redes neuro-*fuzzy* envolve essencialmente três etapas: definir os conjuntos *fuzzy* por meio do particionamento do universo do discurso; determinar a estrutura do modelo (que também envolve a estrutura das regras *fuzzy*); e ajustar os parâmetros da rede (MAMAGHANI; PEDRYCZ, 2020).

A extração de regras *fuzzy* a partir de dados é uma tarefa essencial nesse tipo de rede, permitindo obter sistemas adaptativos, autônomos e com boa capacidade de generalização (TOWNSEND; CHATON; MONTEIRO, 2019). Esta tarefa consiste em identificar as regras mais adequadas para compor a base de conhecimento de um sistema *fuzzy*. A área conhecida como *Neural-symbolic integration* (NSI) busca unir as redes neurais com modelos relacionais simbólicos, explorando técnicas como grafos, árvores de decisão e programação lógica para extrair e mapear relações entre dados de entrada e saída (GARCEZ et al., 2022). Devido à importância desta tarefa, diversas técnicas são utilizadas tanto para definição das regras quanto para o ajuste dos parâmetros de rede, como métodos de *backpropagation* (CHEN et al., 2021; ASHOKKUMAR; MOHANBABU, 2020) e agrupamento (PUSTOKHINA et al., 2021; RAJAN et al., 2023).

A construção de uma estrutura inicial de rede baseada em conhecimento (base da lógica *fuzzy*), alinhada a uma topologia conexionista (estrutura neural) ajuda reduzir o espaço de busca e o tempo para se chegar a uma solução ótima. Para isso, pode-se utilizar técnicas para extração de regras conexionistas baseado no mapeamento de fatores causais e dependências funcionais no domínio dos dados (TICKLE et al., 1998; BANERJEE; MITRA; PAL, 1998).

Algoritmos para extração de regras conexionistas podem ser classificados considerando algumas dimensões apresentadas em (ANDREWS; DIEDERICH; TICKLE, 1995):

- **Poder expressivo das regras:** foca no tipo de regra utilizado e as relações que permite traçar - lógica booleana, proposicional, *fuzzy*, *crisp*, probabilística, entre outros;
- **Translucidez entre a técnica de aprendizado e as unidades neurais subjacentes:**

trata da interpretabilidade de cada unidade neural, ou seja, se cada regra extraída pode ter seu comportamento interpretado (abordagem decomposicional), ou se funcionará como um método caixa-preta, onde avalia-se apenas entrada e saída (abordagem pedagógica);

- **Medida em que a rede neural incorpora a técnica de treinamento utilizada:** define a portabilidade da técnica de extração de regras para outras arquiteturas conexionistas;
- **Qualidade das regras extraídas:** define quão bem a tarefa de extração de regras foi realizada, baseado em fatores ligados ao conjunto de regras como acurácia - capacidade de generalização; fidelidade - se o conjunto de regras pode imitar o comportamento da rede neural da qual foi extraída; consistência - se o conjunto de regras gera a mesma classificação para instâncias de teste mesmo utilizando diferentes instâncias de treinamento; compreensibilidade - medida pela quantidade de regras extraídas e pela quantidade de antecedentes por regra.

Diversas técnicas são exploradas na literatura para este fim. Em [Townsend, Chaton e Monteiro \(2019\)](#) são apresentadas as principais técnicas para extração de relação explicáveis entre variáveis para construção de regras em redes simbólicas. Essas relações podem ser construídas através do cálculo de importância dos neurônios no processo de inferência da rede. Técnicas de *backpropagation* baseadas no gradiente estão entre as mais utilizadas. Em [Shrikumar, Greenside e Kundaje \(2017\)](#) é apresentada uma técnica de decomposição da saída predita da rede de forma mapear a contribuição dos neurônios da rede para cada entrada. Já [Zintgraf et al. \(2017\)](#) utilizam uma técnica de análise de diferença de predição para visualizar a resposta de uma rede profunda para uma entrada específica, permitindo observar a influência de determinado dado de entrada em favor de determinada classe. No sentido de construir regras eficientes para redes *fuzzy* e *neuro-fuzzy* adaptativas destacam-se a utilização de técnicas híbridas ([SOUZA, 2020b](#)). Técnicas de agrupamento também são bastante utilizadas para definição de funções de pertinência e conjuntos *fuzzy*. [Pantula, Miriyala e Mitra \(2020\)](#) apresentam um algoritmo *fuzzy C-Means* para construir mapas funcionais entre os pontos de dados e os valores das funções de pertinência, evitando o aumento marginal no número de pontos de dados e consequente aumento no tamanho das variáveis de decisão.

Técnicas supervisionadas de *backpropagation* podem ser utilizadas tanto para construção de regras quanto para ajuste dos parâmetros. Em [Chen et al. \(2021\)](#) uma árvore de decisão para extração de regras precisas e interpretáveis é apresentada. A abordagem é capaz de capturar uma estrutura de regras simples para a inicialização de redes adaptativas. Outras abordagens que utilizam árvores de decisão para o mesmo fim são exploradas em [Sabah et al. \(2019\)](#) e [Rejeesh \(2019\)](#). Um modelo híbrido extração de regras em redes adaptativas para problemas de classificação de alta dimensão é apresentado em [Xue et al. \(2022\)](#). A

abordagem é baseada em um algoritmo de *backpropagation* para seleção de características com parâmetros ajustáveis que utiliza o conceito de *gate functions*. Uma rede neuro-fuzzy recorrente multifuncional é proposta em [Nasiri e Ebadzadeh \(2022\)](#). A abordagem consiste em duas redes TS funcionando simultaneamente e capazes de aprender múltiplas funções para previsão de séries temporais caóticas. O aprendizado integra um algoritmo de mínimos quadrados um algoritmo de otimização por enxame de partículas para aprendizados dos pesos da rede.

Um modelo *fuzzy* baseado em *gradient boosting* para aprendizado das regras é apresentado em [Zhang et al. \(2024a\)](#). O processo do algoritmo proposto envolve a granularização dos espaços de entrada e saída do modelo e definição das regras. O trabalho também traz um estudo sobre o impacto da qualidade inicial do modelo no processo do *gradient boosting*. Integrando o *gradient boosting* com árvores de regressão, [Mei, Zhao e Xie \(2024\)](#) propõem um método para modelagem de sistemas *fuzzy* hierárquicos com número reduzido de conjuntos e regras *fuzzy*. A otimização dos parâmetros é realizada por um método de mínimos quadrados regularizados e ponderados. Já o trabalho de [Sajid et al. \(2024\)](#) desenvolve uma rede RVFL (*Random Vector Functional Link*) com seu aprendizado baseado em três algoritmos de agrupamento (*R-Means*, *K-Means* e *C-Means*) para obter regras mais interpretáveis. O modelo é aplicado a problemas de classificação e regressão. Um método baseado em gradiente descendente com regularização e *AdaBound* é apresentado em [\(GAO et al., 2024\)](#). O método é aplicado à modelagem de redes TS, focando no ajuste dos parâmetros de rede, e é particularmente eficiente para grandes conjuntos de dados. Outras técnicas para aprendizado de regras com aprendizado extremo ([ASHOKKUMAR; MOHANBABU, 2020](#)) e mínimos quadrados ([LI; WANG; ZHANG, 2017](#)) também podem ser encontradas na literatura.

2.2.3 Sistemas *Fuzzy* Evolucionários

A combinação entre computação evolucionária e sistemas *fuzzy* vem sendo amplamente explorada desde a década de 1990, com o objetivo de integrar o tratamento de imprecisão e incerteza, bem como a interpretabilidade dos sistemas *fuzzy*, à capacidade de adaptação e aprendizado dos algoritmos evolucionários ([FERNANDEZ et al., 2019](#)). Os chamados sistemas *fuzzy* genéticos, ou *fuzzy* evolucionários, são definidos como sistemas *fuzzy* cujo processo de aprendizado é conduzido por técnicas de computação evolucionária, como algoritmos genéticos, programação genética e estratégias evolutivas. Essas técnicas podem ser empregadas tanto na definição da base de regras quanto no ajuste de parâmetros do sistema ([ESMIN; LAMBERT-TORRES, 2007](#); [MOGHADDAM, 2024](#)). O primeiro passo no projeto de um sistema *fuzzy* genético consiste em determinar quais partes do sistema serão modeladas ou otimizadas por meio de métodos evolucionários, representando-as por cromossomos (indivíduos). Essa abordagem pode ser dividida em dois processos principais:

o aprendizado, responsável por extrair conhecimento, gerar regras e definir componentes do sistema; e o ajuste, voltado à obtenção da melhor combinação de parâmetros visando o aprimoramento do desempenho. A estrutura de aprendizado de um sistema *fuzzy* genético baseado em dados é ilustrada na Figura 8.

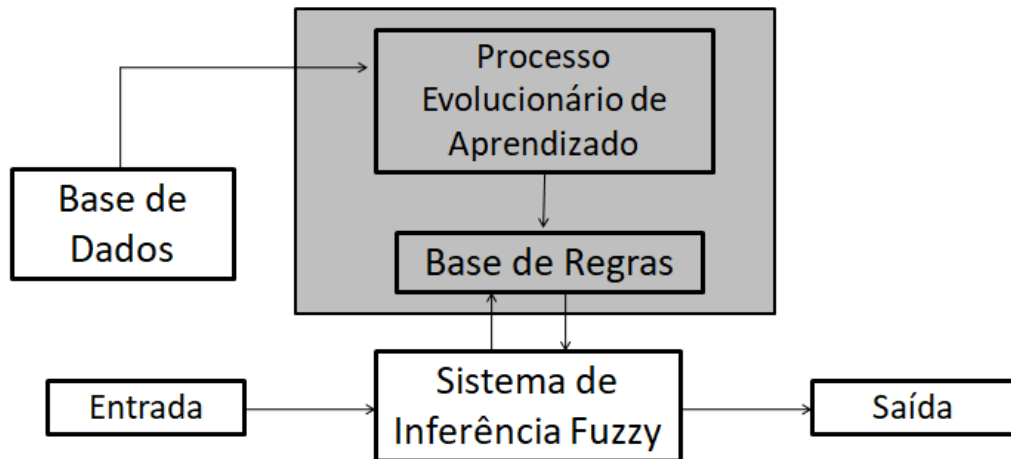


Figura 8 – Sistema *fuzzy* genético baseado em dados.

Considerando a tarefa de aprendizado da base de regras, que é foco deste trabalho, existem quatro possibilidades para realizar o processo de aprendizado genético, conforme descrito em (FERNANDEZ et al., 2015):

- **Seleção evolucionária de regras:** o algoritmo evolucionário atua sobre uma base de regras previamente existente, identificando regras redundantes, irrelevantes, incorretas ou conflitantes, de forma a otimizar o desempenho do sistema por meio de uma base de regras mais eficiente;
- **Aprendizado evolucionário de componentes:** a técnica evolucionária é utilizada para gerar os melhores componentes para a base de conhecimento, como funções de pertinência ou conjuntos *fuzzy*. Nessa abordagem, o espaço de busca pode crescer significativamente, tornando o processo de aprendizado mais lento;
- **Aprendizado evolucionário de regras:** as regras são construídas diretamente pelo algoritmo evolucionário, utilizando, em geral, dados numéricos extraídos da base de dados;
- **Aprendizado evolucionário do banco de dados:** o algoritmo evolucionário é utilizado no processo de geração do banco de dados que permite aprender a forma das funções, a granularidade dos conjuntos *fuzzy* e outros componentes do banco de dados.

A definição automática de um sistema *fuzzy* baseado em regras pode ser vista como um problema de busca ou de otimização, tendo despertado assim o uso de técnicas de computação evolucionária para esta tarefa devido a estas já serem bem conhecidas

como umas técnicas globais de pesquisa capazes de trabalhar em grandes espaços de busca (CORDÓN et al., 2001). Além disso, a estrutura de código genérico e os recursos independentes de desempenho dos algoritmos evolucionários os tornam boas opções para problemas em que se tem um conhecimento *a priori*, que no caso dos sistemas *fuzzy*, pode estar na forma de variáveis linguísticas, parâmetros das funções de pertinência, regras *fuzzy*, número de regras etc. Ou seja, busca-se obter um conjunto adequado de valores para os parâmetros de acordo com o critério de otimização. Esses parâmetros constituem o espaço de busca do algoritmo evolucionário (KELLER DERONG LIU, 2016).

As três principais abordagens genéticas para modelagem da base de regras são: *Michigan*, *Pittsburgh* e IRL (*Iterative Rule Learning*) (CORDÓN et al., 2004). Na abordagem *Michigan* (HOLLAND; REITMAN, 1977), cada indivíduo representa uma única regra, o que reduz o custo computacional e simplifica a codificação, mas dificulta a avaliação da qualidade individual de cada regra. Já na abordagem *Pittsburgh* (SMITH, 1980), cada indivíduo representa uma base completa de regras, facilitando a avaliação do desempenho global, embora com maior custo computacional e tempo de treinamento. A abordagem IRL (VENTURINI, 1993) também representa regras individualmente, mas executa o algoritmo evolucionário repetidamente, retornando uma regra por execução para compor a base final. Critérios como suporte, cobertura e grau de confiança são usados para selecionar as melhores regras (CORDÓN et al., 2001).

Dada a relevância das redes neuro-*fuzzy* e o potencial das técnicas evolucionárias para sua modelagem, diversos trabalhos na literatura têm explorado essa combinação. Em Ashish et al. (2018), algoritmos genéticos (AG) é utilizado para otimizar pesos e eliminar regras redundantes, resultando em uma base de regras mais limpa e eficiente. Já em Soliman et al. (2018), o AG adapta parâmetros do modelo ANFIS (*Adaptive Neuro-Fuzzy Inference System*) em uma arquitetura Takagi-Sugeno, modelando antecedentes e consequentes como variáveis do algoritmo. Utilizando a metodologia IRL, Jozi et al. (2019) propõem a modelagem de regras como indivíduos, compondo a base de regras a partir de um conjunto selecionado. Em Masoumi, Tavakolpour-Saleh e Rahideh (2020), um AG é empregado para ajustar os parâmetros de antecedentes e consequentes de um controlador *fuzzy*, melhorando a qualidade das relações entre entradas e saídas e favorecendo a interpretabilidade.

O desenvolvimento de modelos *fuzzy* para lidar com dados de alta dimensionalidade foi proposto por Mamaghani e Pedrycz (2020). Na arquitetura proposta, a PG é utilizada para formar expressões lógicas *fuzzy* envolvendo operadores lógicos e os conjuntos *fuzzy* do espaço de entrada. Um modelo de regressão *fuzzy* baseado em programação genética multi-gene para análise da satisfação do cliente derivada de avaliações online foi introduzido em Yakubu, Kwong e Lee (2021). Neste caso, a PG é utilizada para extração de regras de um modelo funcional. Em Dirik (2022), são utilizadas abordagens genéticas para aprendizado

e ajuste dos parâmetros em redes de inferência *fuzzy* com arquitetura adaptativa. Os parâmetros de antecedentes e consequentes são modelados e ajustados pelo algoritmo genético em duas redes diferentes. Um sistema *fuzzy* genético é utilizado para estimativa de crescimento urbano em [Foroutan et al. \(2022\)](#). Neste caso, o AG foi utilizado para otimizar as funções de pertinência e os pesos das regras, fornecendo uma modelagem completa do sistema de inferência *fuzzy*. Em [Pietropolli et al. \(2023\)](#), é proposta uma heurística baseada em programação genética linear acoplada a uma função de aptidão autoajustável para construir um modelo *fuzzy* substituto com funções mais suaves que o original. A função substituta é mais fácil de otimizar, mas mantém a propriedade fundamental de localização do ótimo global presente no modelo original.

As técnicas evolucionárias também podem ser associadas a algoritmos de agrupamento visando reduzir ruídos das amostras de dados utilizadas como mostrado em [Moutaouakil, Roudani e Ouissari \(2023\)](#). A abordagem proposta une um algoritmo genético ao C-Means para balancear dados, tornando o modelo menos sensível a ruídos e aumentando sua capacidade de generalização. Em [Moutaouakil, Roudani e Ouissari \(2023\)](#), um AG é integrado ao C-Means para balanceamento e melhoria da generalização. Em [Ran et al. \(2024\)](#), um sistema *fuzzy* adaptativo é proposto combinando AG e colônia de formigas, superando limitações do K-Means e determinando automaticamente o número de *clusters*. O AG é utilizado para determinação autônoma do número ideal de grupos, mitigando a alta variação de parâmetros que impacta na otimização dos *clusters*. Já em [Mazouzi et al. \(2024\)](#) o AG é utilizado para otimizar vários parâmetros rede *fuzzy*, incluindo funções de pertinência, base de regras, pesos as regras e intervalos das variáveis *fuzzy*. Também para otimizar a estrutura de uma rede neuro-*fuzzy* adaptativa, [Guler et al. \(2025\)](#) implementam um algoritmo genético para determinar dinamicamente os parâmetros ideais de um controlador ANFIS. No mesmo sentido, os parâmetros de uma rede MANFIS são otimizados por um AG em [Cheng et al. \(2025\)](#), enquanto [Gülmez \(2025\)](#) utiliza o AG em diversos contextos na rede *fuzzy*, indo desde a otimização de hiperparâmetros até a o ajuste das taxas de aprendizagem e na arquitetura de rede.

Embora algoritmos evolucionários já tenham se mostrado eficientes na modelagem de sistemas *fuzzy*, poucos estudos se aprofundaram em técnicas para maior exploração do espaço de busca nesses modelos, como populações dinâmicas e controle de diversidade, o que se torna essencial especialmente quando se utilizam regras polinomiais com inúmeras possibilidades de composição. Isso pode ser observado nos trabalhos citados acima, especialmente considerando que a maioria deles atinge a “base de regras ótima” com poucas gerações. Algumas abordagens importantes nesse caminho podem ser vistas em [Brindha et al. \(2021\)](#), [Li, He e Cao \(2021\)](#), [Saadatmand e Akbarzadeh-T \(2023\)](#), [Zhang et al. \(2024b\)](#), [Escobar-Cuevas et al. \(2024\)](#), que exploram diferentes técnicas para trazer maior diversidade às soluções geradas por algoritmos evolucionários em modelos *fuzzy*.

No entanto, nenhuma dessas abordagens utiliza o controle de diversidade integrado em populações dinâmicas, como proposto neste trabalho, além de não explorar especificamente a implementação de indivíduos Multi-Gene e diversidade fenotípica.

Capítulo 3

FALCON-GA - *Fuzzy Adaptive Learning Control Network with Genetic Algorithm*

Este capítulo apresenta a abordagem proposta para a construção da base de regras linguísticas do tipo *IF-THEN* em uma rede *fuzzy* adaptativa, empregando um algoritmo genético (AG). A rede proposta é baseada no modelo FALCON (LIN; LEE, 1991) – *Fuzzy Adaptive Learning Control Network*. A rede FALCON consiste em uma rede adaptativa que segue o modelo linguístico de Mamdani (MAMDANI; ASSILIAN, 1975). O modelo proposto, denominado FALCON-GA, utiliza um AG para a extração de regras linguísticas nas quais, como em todo modelo clássico de Mamdani, tanto os antecedentes quanto os consequentes são conjuntos *fuzzy*.

Além disso, o modelo proposto incorpora novos tipos de regras ao modelo original, que utiliza exclusivamente regras do tipo AND, expandindo as possibilidades da rede FALCON por meio de tipos de regras ainda não explorados neste contexto, tornando-a mais flexível e extensível. O FALCON-GA também estabelece uma nova metodologia para a fase de aprendizado supervisionado, utilizando um AG para a extração de regras e um método baseado em gradiente para o ajuste dos parâmetros, em momentos distintos.

A utilização de um AG se justifica pelo fato de que, como discutido na Seção 2.1.1, trata-se de uma técnica evolucionária na qual o indivíduo é classicamente modelado como uma *string* binária, facilitando a associação entre antecedentes e consequentes linguísticos. Apesar de a rede FALCON já ter sido explorada por um número considerável de trabalhos na literatura (como discutido no Capítulo 2), a maioria desses estudos foca em técnicas de agrupamento para a definição dos conjuntos *fuzzy* (QUAH; QUEK; LEEDHAM, 2005; WONG, 2022), não explorando novas abordagens para extração de regras, tampouco sugerindo alterações na composição das regras, como a introdução de novos operadores *fuzzy*.

Dessa forma, este capítulo apresenta não apenas um novo algoritmo de aprendizado evolucionário para a construção de redes FALCON, como também propõe uma estrutura aprimorada para a própria rede. Os principais diferenciais e inovações do modelo FALCON-GA podem ser resumidos como:

- Um novo método de extração de regras linguísticas em redes neuro-fuzzy adaptativas utilizando algoritmo genético;
- Um novo modelo FALCON utilizando múltiplos tipos de regras *fuzzy*;
- Uma arquitetura mais expansível e extensível para o modelo FALCON;

A seguir, a Seção 3.1 revisa a estrutura básica da rede FALCON; a Seção 3.2 apresenta as mudanças estruturais propostas pelo FALCON-GA; a Seção 3.3 descreve o método para definição das funções de pertinência na abordagem proposta; a Seção 3.4 detalha o processo de extração de regras com AG; a Seção 3.5 aborda o processo de atualização dos parâmetros; e, por fim, a Seção 3.6 resume todo o processo de aprendizado do FALCON-GA.

3.1 Revisitando a Rede FALCON

Esta seção apresenta a estrutura básica da rede FALCON (*Fuzzy Adaptive Learning Control Network*), proposta originalmente por Lin e Lee (1991), a qual servirá como base para o algoritmo introduzido neste capítulo. FALCON é uma rede *feedforward* multicamada que integra, de forma eficaz, os principais componentes e mecanismos de um sistema *fuzzy* tradicional em uma arquitetura conexionista.

A estrutura da rede FALCON, representada na Figura 9, é composta por cinco camadas, descritas a seguir:

- **1. Camada de entrada:** esta camada recebe as variáveis de entrada, na qual cada amostra x^t é representada por um vetor $[x_1^t, \dots, x_i^t, \dots, x_n^t]^T$, em que i é o índice da variável, t denota o passo atual e n é a dimensão do espaço de entrada. A camada de entrada apenas repassa esses valores diretamente à próxima camada:

$$f_1 = x^t. \quad (1)$$

- **2. Camada de fuzzificação:** é responsável por transformar as variáveis de entrada x^t em termos linguísticos. Esta camada utiliza funções de pertinência que particionam o universo de discurso de cada uma das n variáveis de entrada. A saída de elemento desta camada, denotada como u_{ij}^t , corresponde ao grau de ativação da função de

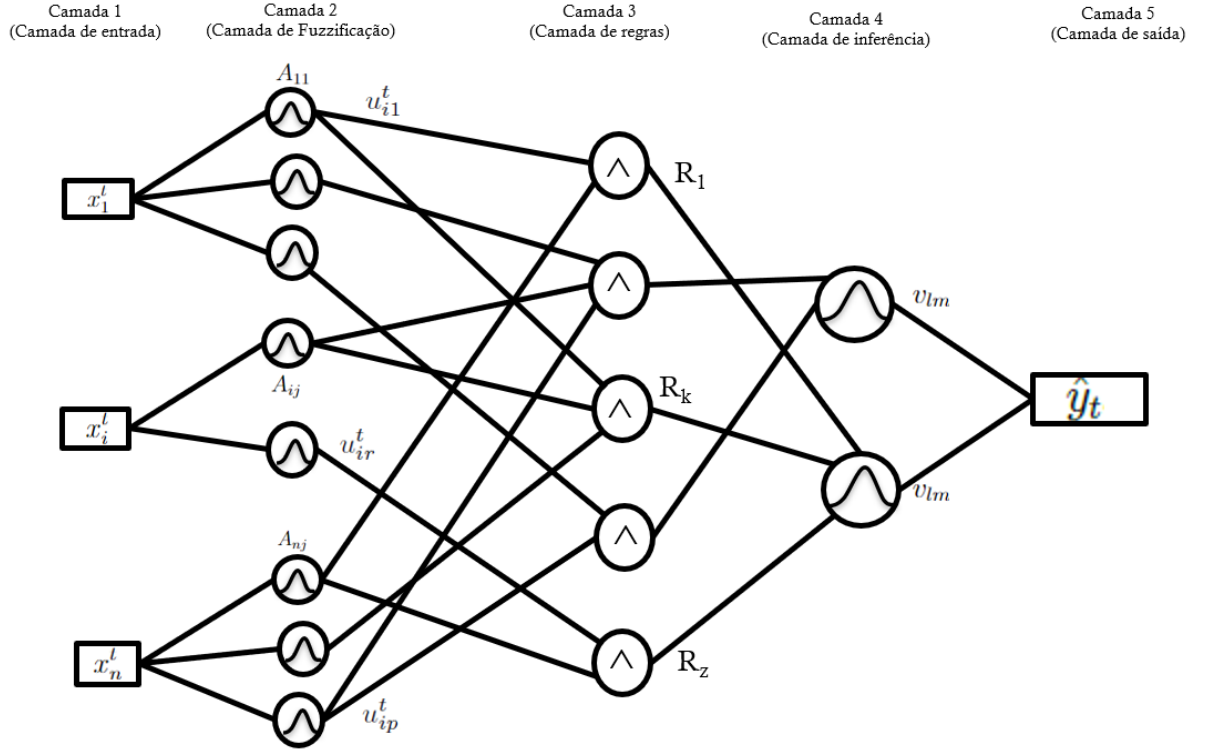


Figura 9 – Estrutura da rede FALCON.

pertinência A_{ij} pela variável de entrada x_i na amostra t , sendo representada por $\mu_{A_{ij}}^t(x_i^t)$. Assim, tem-se:

$$f_2 = u_{ij}^t = \mu_{A_{ij}}^t(x_i^t), \quad (2)$$

em que i indexa a variável de entrada, j é o índice da função de pertinência. No modelo original, são utilizadas funções trapezoidais, no entanto tipos alternativos de funções de pertinência também podem ser utilizados, como os discutido em [Lin e Lee \(1996\)](#) e [Quadrelli, Tanscheit e Vellasco \(2003\)](#).

- **3. Camada de regras:** o papel dessa camada é construir o antecedente das regras, estabelecendo conexões entre os elementos dessa camada e os da Camada 2. Para cada regra, a saída é determinada por uma operação de interseção (AND) que combina os graus de ativação dos conjuntos *fuzzy* associados ao antecedente. O operador *min* é comumente utilizado para essa agregação, por sua simplicidade e eficácia, embora outras *T-normas* também possam ser empregadas, conforme o problema em questão. A saída correspondente à k -ésima regra é dada por:

$$f_3 = D_k = \min(u_{i1}^t, u_{ir}^t, u_{ip}^t), \quad (3)$$

em que u_{ir}^t é o grau de pertinência do r -ésimo conjunto *fuzzy* do antecedente, k é o índice da regra, e p corresponde ao número total de termos linguísticos utilizados para compor o antecedente da r -ésima regra.

- **4. Camada de inferência:** é responsável pela definição dos consequentes das regras. Os neurônios desta camada contêm as funções de pertinência que particionam o universo de discurso da variável de saída. Seu funcionamento ocorre em dois modos distintos que são executados sequencialmente: fase *down-up* e *up-down*. Na fase *down-up*, é realizada a agregação dos antecedentes das regras que foram ativadas:

$$v_{rj} = \sum_{r=1}^z D_r, \quad (4)$$

$$f_4 = \min(1, v_{rj}), \quad (5)$$

em que v_{rj} representa o grau de pertinência da r -ésima regra ao j -ésimo conjunto *fuzzy* consequente, e z representa o número de regras ativadas ligadas a este consequente. A Equação (5), garante que a ativação do consequente não ultrapasse 1.0.

Em seguida, na fase *up-down*, os elementos operam de forma semelhante aos da Camada 2, exceto pelo fato de que apenas 1 neurônio poderá ser ativado nesta camada. O neurônio ativado aplicará a função de pertinência de saída correspondente, capturando assim a informação linguística associada à saída da rede.

É importante ressaltar que os dois modos da camada de inferência são executados sequencialmente. Primeiro, o modo *up-down* é aplicado para obter o grau de ativação dos consequentes de cada regra. Posteriormente, o modo de *down-up* é empregado para agregar as regras, combinando os consequentes e fornecendo a saída resultante para a próxima camada.

- **5. Camada de saída:** esta camada realiza a defuzzificação dos consequentes, que consiste na conversão dos valores de saída *fuzzy* em valores *crisp*. Para isso, é utilizado o método do centro da área (*centroid*), que calcula o valor médio ponderado dos centros das funções de pertinência ativadas, conforme:

$$f_5 = \hat{y}_t = \frac{\sum_j v_j c_j}{\sum_j v_j}, \quad (6)$$

em que v_j representa o grau de ativação do j -ésimo conjunto *fuzzy* consequente (proveniente da Camada 4), e c_j é o centro correspondente da função de pertinência associada. Esse processo gera a saída final da rede para a amostra t como um valor numérico representativo do conjunto de regras ativadas.

3.2 Estrutura da Rede FALCON-GA

A rede FALCON-GA segue a mesma arquitetura da rede FALCON, porém, diferente do original, incorpora três operadores *fuzzy*, a saber, *AND*, *OR* e *SUM*, para compor os antecedentes das regras da Camada 3. Com base nos princípios discutidos na Seção 3.1, o modelo proposto garante que a função do operador *AND* esteja alinhada aos conceitos e mecanismos da arquitetura FALCON, conforme definido na Equação (3). Por outro lado, a regra *OR* realiza uma operação *max* sobre os antecedentes associados:

$$f_3 = D_i = \max(u_{i1}^t, u_{ir}^t, u_{ip}^t). \quad (7)$$

Além disso, o operador *SUM* realiza a soma de antecedentes vinculados:

$$f_3 = D_i = \sum_{i=1}^p u_{ir}^t. \quad (8)$$

Como resultado, a composição das regras *fuzzy* no FALCON-GA é altamente flexível e adaptável. Assim, além dos operadores *AND*, *OR* e *SUM*, é possível incluir outros operadores conforme os requisitos do problema. A estrutura geral das regras no FALCON-GA é:

$$\begin{array}{llllll} R_1 & \mathbf{SE} & x_1 \text{ é } A_{i1} & \mathbf{E} & x_2 \text{ é } A_{i2} & \mathbf{E} & x_3 \text{ é } A_{i2} & \mathbf{ENTÃO} & y_i \text{ é } O_{i1} \\ R_k & \mathbf{SE} & x_1 \text{ é } A_{i2} & \mathbf{OU} & x_2 \text{ é } A_{i3} & \mathbf{OU} & x_p \text{ é } A_{i1} & \mathbf{ENTÃO} & y_i \text{ é } O_{i2} \\ & & \cdot & & & & & & \\ & & \cdot & & & & & & \\ & & \cdot & & & & & & \\ R_z & \mathbf{SE} & x_1 \text{ é } A_{i2} & \mathbf{OU} & x_2 \text{ é } A_{i2} & \mathbf{OU} & x_p \text{ é } A_{i3} & \mathbf{ENTÃO} & y_i \text{ é } O_{iJ} \end{array}$$

em que x^t representa a entrada da rede FALCON, p é o número de conjuntos *fuzzy* usados para construir o antecedente da regra, A representa os conjuntos *fuzzy* de entrada ativados, O representa os conjuntos *fuzzy* de saída ativados e z é o número de regras.

O método de aprendizado empregado na construção da rede FALCON-GA combina três técnicas distintas:

- O algoritmo de agrupamento ART (*Adaptive Resonance Theory*) é empregado para criar as funções de pertinência iniciais. Este algoritmo identifica os grupos nos dados de entrada, que servem de base para a definição das funções de pertinência de entrada e saída.
- O algoritmo genético (AG) otimiza as bases de regras. O AG busca a combinação mais adequada de operadores *fuzzy*, antecedentes e consequentes dentro das bases de regras por meio de um processo evolucionário.
- O método Gradiente é aplicado para ajustar os parâmetros das funções de pertinência.

3.3 Funções de Pertinência

O FALCON-GA utiliza funções de pertinência gaussianas para granularizar o universo de discurso das variáveis de entrada e saída. Uma função de pertinência Gaussiana é representada por seu centro c e espalhamento s . O grau de ativação da função A_{ij} para a variável x_i^t é obtido por:

$$\mu_{A_{ij}}(x_i^t) = \exp\left(-\frac{1}{2}\left(\frac{x_i^t - c_{ij}}{s_{ij}}\right)^2\right). \quad (9)$$

O formato de uma função de pertinência gaussiana pode ser visto na Figura 10.

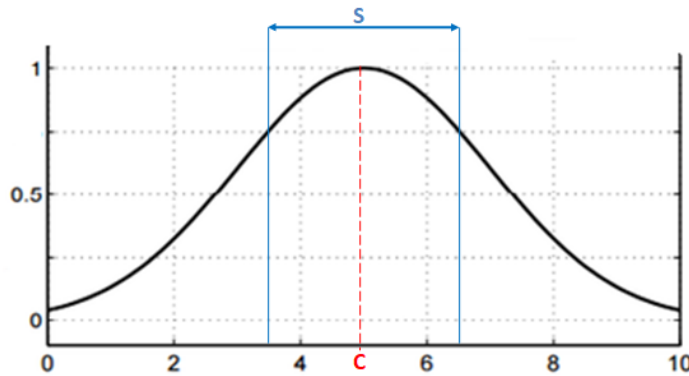


Figura 10 – Função de pertinência gaussiana.

No FALCON-GA, os parâmetros c e s das funções de pertinência são definidos a partir dos grupos gerados pelo algoritmo de agrupamento *Fuzzy ART* (CARPENTER; GROSSBERG; ROSEN, 1991). *Fuzzy ART* é um algoritmo de aprendizado não supervisionado para categorização de dados e tarefas de agrupamento. Ele integra a teoria dos conjuntos *fuzzy* a um componente ART (*Adaptive Resonance Theory*) para processamento cognitivo de informação humana. O modelo ART é muito utilizado para criação de redes neurais a partir de sequências de dados arbitrárias e dados ruidosos, sendo capaz de aprender rapidamente

mapeamentos categóricos estáveis entre vetores de entrada e saída analógicos ou binários (CARPENTER; GROSSBERG; REYNOLDS, 1991).

No FALCON-GA, os centros obtidos para cada grupo são usados como os centros c das respectivas funções de pertinência. O espalhamento s de cada função de pertinência é calculado por:

$$s_{ij} = w_{ij}(x_{max_i} - x_{min_i}), \quad (10)$$

em que s representa o espalhamento da função de pertinência, w denota o peso, i é o índice da variável de entrada, j é o índice do conjunto *fuzzy*, x_{max} representa o valor máximo para a variável de entrada indexada por i , e x_{min} representa o valor mínimo da variável de entrada indexada por i .

O Algoritmo 1 detalha o procedimento para criar as funções de pertinência no FALCON-GA utilizando o algoritmo *Fuzzy ART*.

Algoritmo 1: Geração das funções de pertinência.

```

1 Aplica entradas ao Fuzzy ART
2 Fuzzy ART retorna categorias, vetor_centros, vetor_pesos
3  $m = \text{categorias}$ ;
4  $c2 = \text{vetor\_centros}$ ;
5 for  $j = 1 : m$  do
6   for  $i = 1 : n$  do
7      $s2(i, j) = \text{vetor\_pesos}(i, j) * (x_{max_i} - x_{min_i}) - (10)$ ;
8   end
9 end
10 Aplica saídas ao Fuzzy ART
11 Fuzzy ART retorna categorias, vetor_centros, vetor_pesos
12  $mt = \text{categorias}$ ;
13  $c4 = \text{vetor\_centros}$ ;
14 for  $j = 1 : mt$  do
15   for  $i = 1 : nout$  do
16      $s(i, j) = \text{vetor\_pesos}(i, j) * (y_{max_i} - y_{min_i}) - (10)$ ;
17   end
18 end

```

Mais detalhes sobre o *Fuzzy ART* podem ser encontrados em (CARPENTER; GROSSBERG; ROSEN, 1991) e (LIN; LIN, 1997). Ressalta-se que, nesta etapa do FALCON-GA, é possível empregar quaisquer técnicas de agrupamento para o particionamento do universo de discurso.

3.4 Algoritmo Genético para Extração de Regras

Após a definição das funções de pertinência, um algoritmo genético (AG) é empregado para identificar as regras que comporão a rede FALCON-GA. Essas regras consistem em combinações entre antecedentes, operadores *fuzzy* e consequentes, e o AG tem como

objetivo selecionar, dentre a vasta gama de possibilidades, as combinações mais eficazes para a modelagem do problema.

O processo inicia-se com a geração de uma matriz de regras MR . A matriz MR contém todas as regras possíveis para rede e é criada a partir do mapeamento de todas as combinações possíveis entre, antecedentes, operadores e consequentes. Em seguida, o AG é aplicado para selecionar um subconjunto de regras que formará a base de regras da rede FALCON-GA. Para simplificar e acelerar a execução do algoritmo, todas as saídas associadas às regras são calculadas antecipadamente, resultando em um vetor de saídas completo. Essas saídas correspondem às ativações potenciais da Camada 3 da arquitetura FALCON-GA.

3.4.1 Representação dos indivíduos

Na abordagem proposta, cada indivíduo do AG representa uma base de regras completa. A representação é feita por meio de um vetor binário de comprimento R , onde R corresponde ao número total de regras disponíveis na matriz MR . Cada elemento do vetor assume o valor 1 para indicar a inclusão de uma regra na base, ou 0 para indicar sua exclusão. A Figura 11 ilustra um exemplo de indivíduo considerando uma matriz MR com 10 regras possíveis.

0	0	1	1	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---

Indivíduo que contém as regras 3,4,6,7,8

1	0	1	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---	---	---

Indivíduo que contém as regras 1,3,5,6,8,10

Figura 11 – Representação dos indivíduos no FALCON-GA.

3.4.2 Geração da população inicial

Durante a geração inicial da população, os indivíduos são criados de forma aleatória. Nesse processo, definem-se previamente os limites mínimo e máximo de regras que cada indivíduo pode conter, estabelecendo-se esses valores como parâmetros do algoritmo. Para cada indivíduo gerado, é selecionada uma quantidade específica de regras (n_r), sorteadas aleatoriamente entre as R possibilidades listadas na matriz MR .

As etapas envolvidas nesse processo de geração da população inicial estão descritas no Algoritmo 2.

Algoritmo 2: Geração da população inicial

```
1 for individuo = 1 : T do
2   Sorteia um número (z) de regras;
3   for regra = 1 : z do
4     // pos é o índice da matriz MR que corresponde à regra sorteada para compor a base de regras
5     pos ← sorteia posição da matriz MR ;
6     rb(pos) = 1;
7   end
8 end
```

3.4.3 Avaliação da população

Após a geração da população, a saída da rede FALCON-GA é calculada para cada indivíduo, conforme mostrado no Algoritmo 3.

Algoritmo 3: Cálculo da saída da rede.

```
1 for t = 1 : N do
2   //Fuzzificação das entradas
3   for i = 1 : n do
4     for j = 1 : m do
5       |  $u(i, j) = \mu_{A_{ij}}(x_i)$  - (2);
6     end
7   end
8   //Executa regras
9   //tamMR é tamanho da matriz MR for r = 1 : tamMR do
10    if rb(r) == 1 then
11      if rb(r) == 1 then
12        if regra é AND then
13          |  $R(r) = \min(u_{i1}^t, u_{ir}^t, u_{ip}^t)$  - (3);
14        end
15        if regra é OR then
16          |  $R(r) = \max(u_{i1}^t, u_{ir}^t, u_{ip}^t)$  - (7);
17        end
18        if regra é SUM then
19          |  $R(r) = \sum_{i=1}^p u_{ir}^t$  - (8);
20        end
21      end
22    end
23  end
24  //Agrega a saída das regras for m = 1 : mt do
25    for k = 1 : length_rb do
26      |  $v(m) = v(m) + R_k$  - (4);
27    end
28    f4(m) = min(v(m)) - (5);
29  end
30   $\hat{y}^t = defuzz(mt, f4)$  - (6);
31 end
```

Posteriormente, para ser utilizado como *fitness*, o erro quadrático médio (EQM) é calculado para cada indivíduo:

$$MSE = \frac{\sum_{t=1}^N ((y^t - \hat{y}^t)^2)}{N}, \quad (11)$$

em que y^t representa a saída desejada no tempo t , $\hat{y}^t$ a saída da rede e N o número de amostras.

3.4.4 Geração da nova população

O FALCON-GA segue uma abordagem elitista, onde o melhor indivíduo da população atual é preservado e transferido diretamente para a próxima geração. Os indivíduos restantes são gerados usando uma combinação de operações de cruzamento, mutação e reprodução.

Durante a operação de cruzamento no FALCON-GA, as regras do Pai 1 são combinadas com as do Pai 2. Neste processo são desconsideradas as posições do vetor que contenham 0, garantindo que apenas as regras relevantes sejam consideradas para o cruzamento. A Figura 12 ilustra um exemplo da operação de cruzamento no FALCON-GA, onde o índice 4 é escolhido como o ponto de cruzamento. Neste ponto, as regras além do índice 4 no Pai 1 são trocadas com as regras correspondentes do Pai 2, resultando na geração de novos indivíduos filhos.

0	0	1	1	0	1	1	1	0	0	Pai 1
1	0	1	0	1	1	0	1	0	1	Pai 2
0	0	1	1	1	1	0	1	0	1	Filho 1
1	0	1	0	0	1	1	1	0	0	Filho 2

Figura 12 – Cruzamento no FALCON-GA.

O processo de reprodução no contexto do FALCON-GA envolve a geração de filhos que são cópias idênticas de seus pais, chamados de clones. Se os pais não atenderem à probabilidade de cruzamento, a reprodução é aplicada a ambos os indivíduos, o que significa que sua informação genética permanece inalterada e é transmitida diretamente para a próxima geração.

Já o operador de mutação é responsável por alterar uma regra dentro de um indivíduo. Envolve a substituição de uma regra existente no indivíduo por outra regra da matriz MR que ainda não esteja incluída nela. A Figura 13 mostra um exemplo do processo de mutação. Nesse caso, o indivíduo inicialmente contém a regra 7. Através da mutação, a regra 7 é substituída pela regra 4, que não fazia parte do indivíduo original.

A nova população é gerada substituindo a população atual pela população recém-criada. O elitismo garante que o melhor indivíduo seja sempre copiado para a nova população.

1	0	1	0	0	1	1	1	0	0	Antes da mutação
1	0	1	1	0	1	0	1	0	0	Após mutação

Figura 13 – Mutação no FALCON-GA.

3.5 Atualização dos Parâmetros

A atualização dos parâmetros das funções de pertinência do FALCON-GA é realizada usando um método baseado no gradiente. Esse processo ocorre após a fase de construção da base de regras, realizada pelo algoritmo genético, e não envolve modificações na estrutura das demais camadas. Consequentemente, as próprias funções de pertinência permanecem inalteradas durante a execução do algoritmo genético. Na fase anterior, o foco é apenas ajustar a composição das regras dentro da rede FALCON-GA.

Durante o aprendizado desta etapa, que é supervisionada, o objetivo é atualizar os parâmetros c e s das funções de pertinência de entrada e saída. O método Gradiente, nesta fase, atualizará a estrutura dos neurônios nas Camadas 2 e 4 da rede FALCON-GA. O método é executado por l épocas, sendo os parâmetros atualizados da seguinte forma em cada época:

$$c_{ij} = c_{ij} - \eta (y_t - \hat{y}_t) dW_{ti} dc_{ij}, \quad (12)$$

$$s_{ij} = s_{ij} - \eta (y_t - \hat{y}_t) dW_{ti} ds_{ij}, \quad (13)$$

em que dW_{ij} , dc_{ij} e ds_{ij} são obtidos respectivamente por:

$$dW_{ij} = \frac{\mu_{A_{ij}}}{\sum_{j=1}^{m_i} \mu_{A_{ij}}}, \quad (14)$$

$$dc_{ij} = \mu_{A_{ij}} \frac{x_i^t - c_{ij}}{c_{ij}^2}, \quad (15)$$

$$ds_{ij} = \mu_{A_{ij}} \frac{x_i^t - s_{ij}}{s_{ij}^2}. \quad (16)$$

3.6 Algoritmo de Aprendizado do FALCON-GA

Com base nas discussões apresentadas nas Seções 3.3, 3.4 e 3.5, a Figura 14 ilustra o fluxo de execução do FALCON-GA, evidenciando a integração entre os algoritmos ART, o

Algoritmo Genético (AG) e o método baseado em gradiente. Já o Algoritmo 4 resume todo o procedimento de aprendizado adotado no modelo.

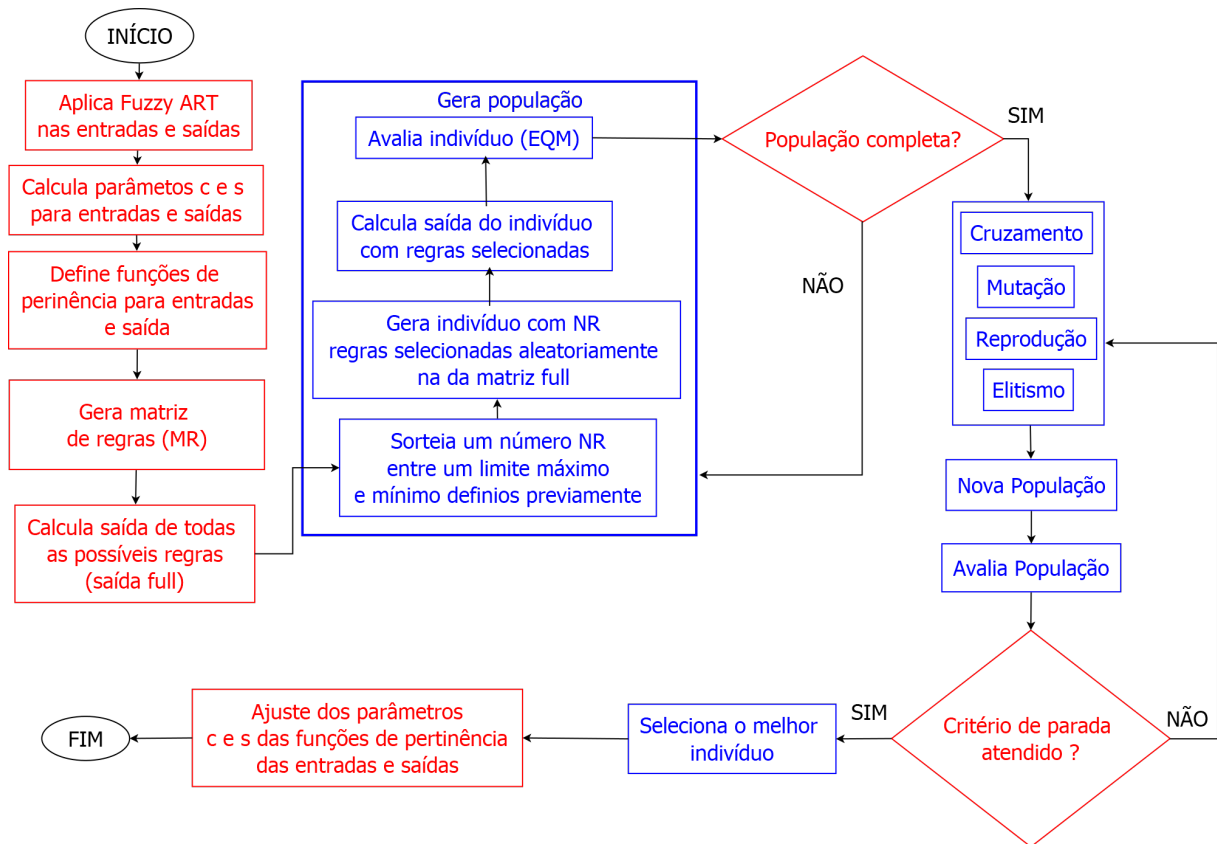


Figura 14 – Fluxograma do algoritmo FALCON-GA.

Algoritmo 4: Algoritmo de aprendizado do FALCON-GA.

```
1 Entrada  $x_t, y_t, g, T, tx\_c, tx\_m$  ;
2 Saída  $\hat{y}_t$  ;
3 Gera funções de pertinência - Algoritmo 1;
4 Gera população inicial - Algoritmo 2;
5 for  $geracao = 1 : g$  do
6   for  $individo = 1 : T$  do
7     Processa saída da rede - Algoritmo 3;
8     Calcula Fitness (11) ;
9   end
10  Seleciona melhor indivíduo;
11  Atribui base de regras do melhor indivíduo à rede;
12  //Gera nova população
13  Aplica elitismo;
14  for  $individo = 2 : T$  do
15    Pai1  $\leftarrow$  Selecaotorneio(PopulacaoAtual,Trn);
16    Pai2  $\leftarrow$  Selecaotorneio(PopulacaoAtual,Trn);
17    //Checa probabilidade de cruzamento
18    if  $rand < tx\_c$  then
19      Filho1  $\leftarrow$  Cruzamento(Pai1,Pai2);
20      Filho2  $\leftarrow$  Cruzamento(Pai2,Pai1);
21    else
22      //Reprodução
23      Filho1  $\leftarrow$  Pai1;
24      Filho2  $\leftarrow$  Pai2;
25    end
26    //Checa probabilidade de mutação
27    if  $rand < tx\_m$  then
28      Filho1  $\leftarrow$  Mutação(Filho1);
29    end
30    if  $rand < tx\_m$  then
31      Filho2  $\leftarrow$  Mutação(Filho2);
32    end
33  end
34  Substitui população antiga pela nova;
35 end
36 Ajusta parâmetros das funções de pertinência pelo método de gradiente (12-16);
```

Capítulo 4

EvoNFuzz - *Evolutionary Neuro-Fuzzy Network*

Este capítulo apresenta a rede EvoNFuzz - *Evolutionary Neuro-Fuzzy Network* -, proposta neste trabalho. EvoNFuzz é um modelo polinomial, isto é, um modelo funcional que emprega funções polinomiais representadas por expressões matemáticas no formato $f(\bullet)$, que são modeladas e evoluídas por um método de programação genética multi-gene (PGMG), para determinar os consequentes das regras. A modelagem por PG permite obter regras com diversos formatos e evoluí-las dinamicamente, ao contrário de modelos tradicionais, que são baseados no modelo clássico de Takagi-Sugeno (TAKAGI; SUGENO, 1985), já possuem um formato pré-definido de regras funcionais e restringem seu aprendizado a ajustar os parâmetros numéricos destas funções como, por exemplo, o ANFIS (Jang, 1993) e o NFN (YAMAKAWA et al., 1992). As expressões matemáticas modeladas pela PG permitem capturar melhor relacionamentos complexos entre variáveis, tornando-o mais flexível e adaptável em diversas aplicações. A abordagem com PGMG permite modelar uma base de regras completa em um único indivíduo, visto que cada indivíduo é composto por um conjunto árvores de representação, como será detalhado na Seção 4.2.

Além disso, o modelo proposto incorpora pesos sinápticos à sua estrutura, aproximando-se da arquitetura das redes neurais, o que permite definir graus de influência ou importância diferentes para cada regra. Esta abordagem traz maior robustez à modelos neuro-*fuzzy* e favorece o processo de aprendizagem e generalização. Tanto os pesos sinápticos quanto as funções de pertinência são ajustados por um método baseado no gradiente, que se integra ao método de programação genética ajustando os parâmetros de rede a cada geração. Uma vez que cada indivíduo possui seu próprio conjunto de parâmetros, o método do gradiente permite melhorar a performance dos indivíduos a cada geração, guiando a PG para áreas mais promissoras do espaço de busca.

As principais novidades trazidas pelo modelo EvoNFuzz são:

- Um novo modelo com regras polinomiais integradas a pesos sinápticos, aproximando sua estrutura das arquiteturas conexionistas das redes neurais;
- Modelagem das funções de consequente através da PGMG integrada a um algoritmo e *backpropagation* baseado no gradiente para ajustar os parâmetros de rede;
- Estimativa dos consequentes através de expressões matemáticas no formato $f(\bullet)$, permitindo capturar relações mais complexas entre variáveis, especialmente relações não-lineares, e obter regras em diferentes formatos;
- Saída da rede calculada de forma mais simples, através da ponderação pelos pesos sinápticos como nas redes neurais.

A seguir, a Seção 4.1 apresenta a estrutura de rede proposta neste capítulo; a Seção 4.2 explica o processo de modelagem da base de regras com PGMG e a Seção 4.3 detalha todo o processo de aprendizado do EvoNFuzz.

4.1 Estrutura da Rede EvoNFuzz

EvoNFuzz é uma rede neuro-fuzzy evolucionária preditiva com regras polinomiais, formada por três componentes principais: Componente de Fuzzificação, Componente de Inferência e Componente de Agregação. Cada um deles desempenha uma etapa diferente e importante no processo de predição. Os componentes são detalhados a seguir e ilustrados na Figura 15.

Componente de Fuzzificação (L^1): O primeiro componente da rede EvoNFuzz contém neurônios fuzzy compostos por funções de pertinência gaussianas, responsáveis por fuzzificar as variáveis de entrada. Cada neurônio representa um conjunto *fuzzy* A_{ij} relacionado à variável x_i que lhe está vinculada. A saída de cada neurônio (L_{ij}^1) recebe o grau de pertinência μ da variável x_i no respectivo conjunto fuzzy A_{ij} :

$$L_{ij}^1 = \mu_{A_{ij}}(x_i), \quad (17)$$

em que i indexa as variáveis de entrada, j indexa a função de pertinência e x é a variável de entrada descrita como um vetor $[x_1^t \cdots x_i^t \cdots x_m^t]^T$, em que t é o passo atual e n é a dimensão do espaço de entrada. Como mencionado, o grau de ativação $\mu_{A_{ij}}(x_i)$ é dado pela função de pertinência gaussiana e é calculado por:

$$\mu_{A_{ij}}(x_i) = \exp\left(-\frac{1}{2}\left(\frac{x_i - c_{ij}}{s_{ij}}\right)^2\right), \quad (18)$$

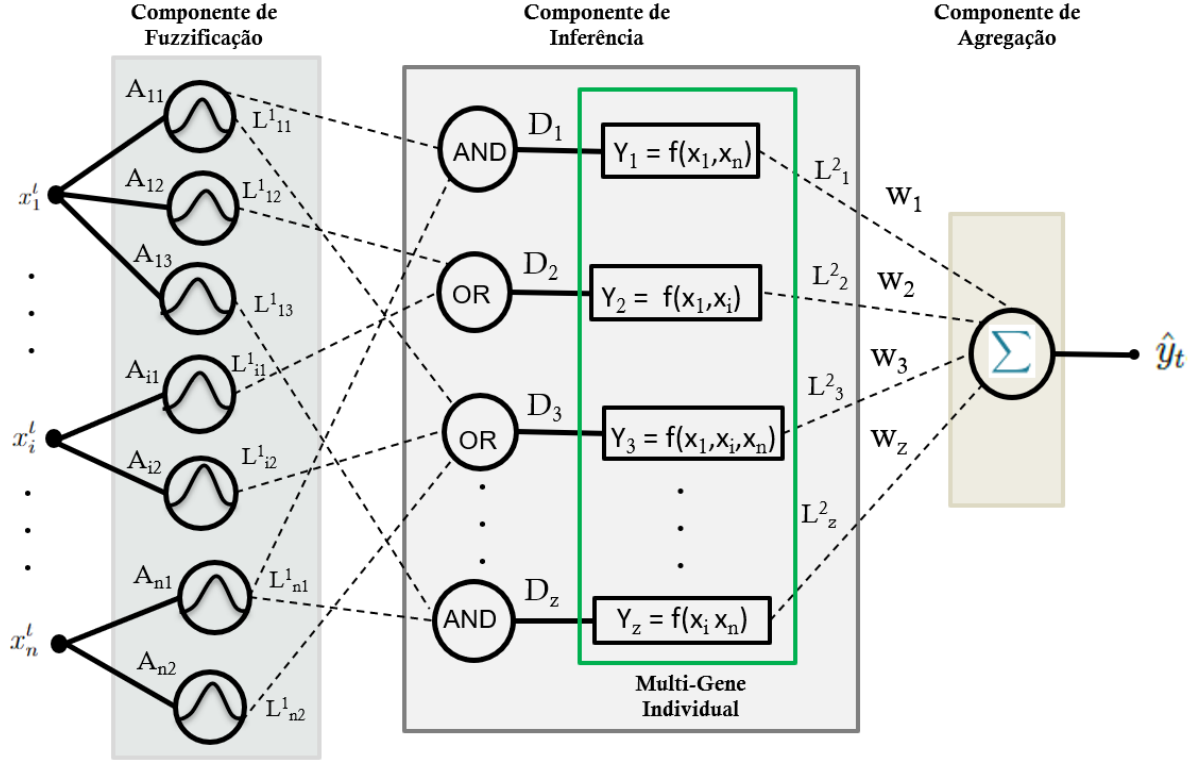


Figura 15 – Estrutura da rede EvoNFuzz.

em que c_{ij} e s_{ij} são o centro e a dispersão da função de pertinência j vinculada à variável de entrada i , respectivamente.

Componente de Inferência (L^2): O componente de Inferência compreende a base de regras do EvoNFuzz, formada por regras polinomiais *IF-THEN*, divididas em duas camadas. A primeira contém os neurônios que encapsulam os operadores lógicos composicionais AND e OR para combinar os antecedentes das regras. Os antecedentes são formados pelas variáveis de entrada fuzzificadas recebidas do componente anterior e o grau de disparo D_r de cada antecedente é dado pelo operador *min* para regras AND, conforme (19) e pelo operador *max* para regras OR, conforme (20).

$$D_r = \min(L_{11}^1, L_{n1}^1), \quad (19)$$

$$D_r = \max(L_{12}^1, L_{i1}^1, L_{n2}^1). \quad (20)$$

em que r indexa a regra.

Já os consequentes Y_r são representados por funções matemáticas $f(x)$ modeladas por indivíduos genéticos multi-gene. Tomando como exemplo os antecedentes mostrados em (19) e (20), dois exemplos da regra podem ser:

R_1 :

$$\text{SE } \overbrace{x_1 \text{ é } A_{11}}^{L_{11}^1 = \mu_{A_{11}}(x_1)} \text{ E } \overbrace{x_n \text{ é } A_{n1}}^{L_{n1}^1 = \mu_{A_{n1}}(x_n)} \\ \text{ENTÃO } Y_1 = (4x_1 + 3x_n)/(2x_n).$$

R_2 :

$$\text{SE } \overbrace{x_1 \text{ é } A_{12}}^{L_{12}^1 = \mu_{A_{12}}(x_1)} \text{ OU } \overbrace{x_i \text{ é } A_{i1}}^{L_{i1}^1 = \mu_{A_{i1}}(x_i)} \text{ OU } \overbrace{x_n \text{ é } A_{n2}}^{L_{n2}^1 = \mu_{A_{n2}}(x_n)} \\ \text{ENTÃO } Y_r = x_i(2x_1 - 3) + ((x_n - x_i)/5x_1).$$

em que R representa uma regra *fuzzy*, r indexa a regra e Y denota o consequente polinomial $f(x)$. Como pode ser visto no exemplo, uma regra *fuzzy* no EvoNFuzz pode ser composta por uma ou mais variáveis de entrada e apenas uma função de pertinência por variável associada.

Como a função do consequente é uma função direta das variáveis de entrada $f(x)$, o consequente Y_r de uma regra r deve ser multiplicado pelo grau de ativação do seu antecedente D_r . Assim, a saída do Componente de Inferência (L^2) é dada por:

$$L_r^2 = D_r Y_r, \quad (21)$$

em que r indexa a regra.

Componente de Agregação (L^3): O terceiro componente é responsável por agregar as regras ativadas e calcular a saída da rede. Este componente se conecta ao anterior por meio dos pesos sinápticos, que determinam o nível de influência e a importância de cada regra na base de regras. Assim, a saída do modelo $\hat{y}$ é dada pela soma das saídas do componente de inferência (L^2) multiplicada pelos pesos sinápticos correspondentes (w), dividida pela soma dos pesos, conforme:

$$\hat{y} = L^3 = \frac{\sum_{r=1}^z L_r^2 w_r}{\sum_{r=1}^z w_r}. \quad (22)$$

em que z é o número de regras ativadas e r indexa a regra.

4.2 Modelagem da Base de Regras com Programação Genética

A rede proposta utiliza programação genética multi-gene (PGMG) (FERREIRA, 2001) para modelar a base de regras que compõe o componente de inferência da rede. PGMG é uma variação da programação genética (PG), onde os indivíduos são representados por um conjunto de árvores conectados por uma estrutura linear de genes, chamados genes superiores, ao invés de apenas uma árvore como na abordagem original (NIAZKAR, 2023). A saída do indivíduo será a agregação da saída de todas as sub-árvores que o compõe. Esta representação colabora para a modelagem de bases de regras *fuzzy*, visto que, muitas vezes, uma regra isolada pode não ter um significado muito expressivo ou uma grande relevância para a saída da rede. Por este motivo, a abordagem de Pittsburg (SMITH, 1980) propõe uma modelagem evolucionária em que cada indivíduo representa uma base de regras completa, facilitando assim sua avaliação durante o processo evolucionário. Assim, a PGMG torna-se uma ferramenta interessante para facilitar a modelagem da base de regras em redes *fuzzy* evolucionárias.

4.2.1 Representação do indivíduo e operadores genéticos

Na abordagem proposta, cada indivíduo representa um conjunto de funções, modeladas como um conjunto de árvores de representação, que estimam os consequentes polinomiais das regras. Ou seja, a PG atua na criação e ajuste dos consequentes das regras, onde cada indivíduo representa um conjunto de funções de consequente, que estão ligadas aos seus respectivos antecedentes e pesos sinápticos. A Figura 16 mostra um exemplo de indivíduo multi-gene composto por 3 regras polinomiais. Conforme apresentado na Seção 2.1.2, pode ser visto na figura que as funções do indivíduo são compostas por operações matemáticas básicas e (+, -, *, /), os terminais, por variáveis e constantes.

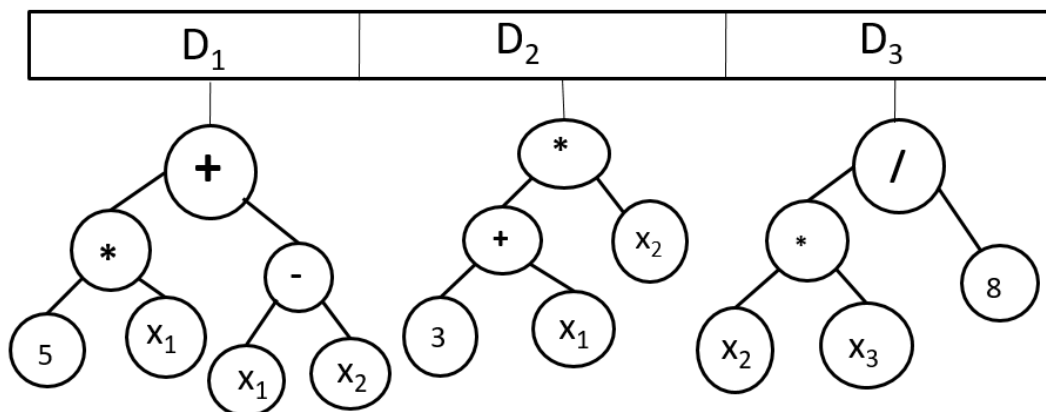


Figura 16 – Indivíduo multi-gene no EvoNFuzz.

Como pode ser visto no exemplo, o consequente de cada uma das regras é definido respectivamente por:

$$R_1: Y_1 = 5x_1 + (x_1 - x_2).$$

$$R_2: Y_2 = (3 + x_1) * x_2.$$

$$R_3: Y_3 = (x_2 * x_3)/8).$$

Neste exemplo, as regras 1 e 2 estão associadas as variáveis x_1 e x_2 e a regra 3 está associada às variáveis x_2 e x_3 . Vale ressaltar que a PGMG, como já citado, cria e ajusta os consequentes das regras, sendo que seus antecedentes já foram formados e calculados na camada anterior, conforme Figura 15.

As operações genéticas na PGMG podem ser realizadas em alto nível - manipulando os genes superiores e, consequentemente sub-árvores inteiras -, ou em baixo nível - manipulando partes das sub-árvores. No caso do EvoNFuzz, as operações tanto de cruzamento quanto de mutação, são realizadas em alto nível. Visto que cada sub-árvore representa uma regra, uma operação em alto nível indica que os dois operadores alterarão regras inteiras na base de regras e não a composição destas regras. Ou seja, no cruzamento as regras do Pai 1 serão recombinadas com as regras do Pai 2 para gerar os filhos. Já na mutação, uma regra escolhida será trocada por uma nova regra gerada aleatoriamente. Desta forma, operador de cruzamento é responsável por recombinar regras já existentes em indivíduos da população, podendo também alterar o número de regras ao gerar os filhos. Já a mutação é responsável por inserir informação nova, isto é, novas regras na população. As operações em alto nível evitam o aumento excessivo do tamanho das funções polinomiais representadas pelas sub-árvores do indivíduo, ou seja, evita a obtenção de regras excessivamente grandes, como será discutido mais adiante no Capítulo 6. As figuras 17 e 18 ilustram, respectivamente, as operações de cruzamento e mutação no modelo proposto. No cruzamento, as regras que compõem os indivíduos pais são recombinadas para formar os filhos, de modo que cada filho herda parte das regras de cada pai, sem que haja inclusão ou exclusão de regras. Por outro lado, na mutação, uma das regras é selecionada para ser modificada: ela é removida da base de regras e substituída por uma nova.

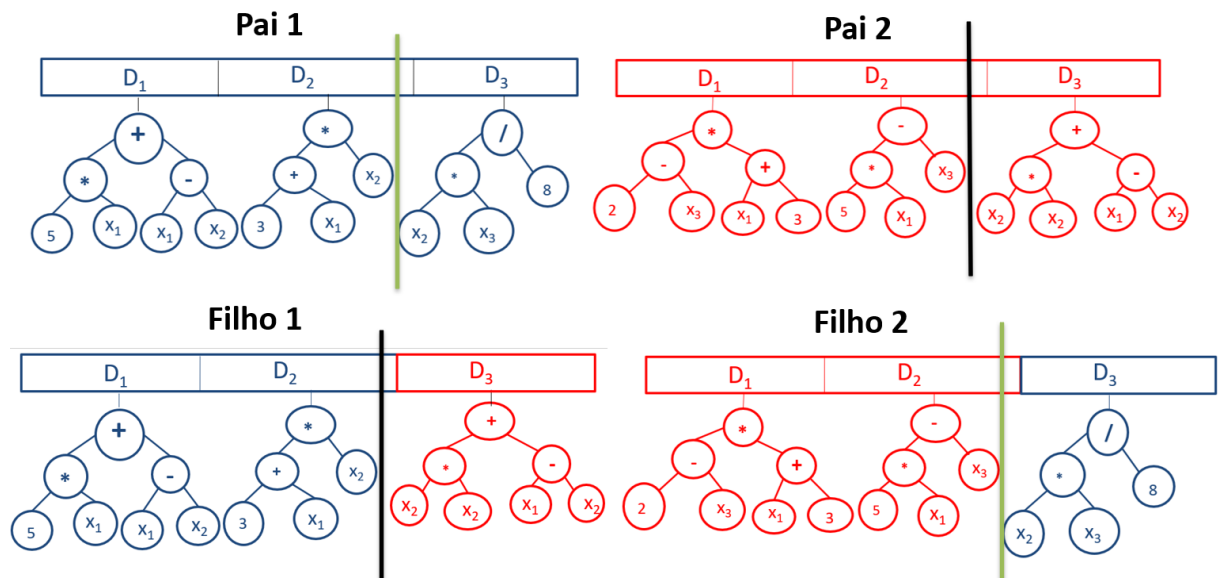


Figura 17 – Cruzamento no EvoNFuzz.

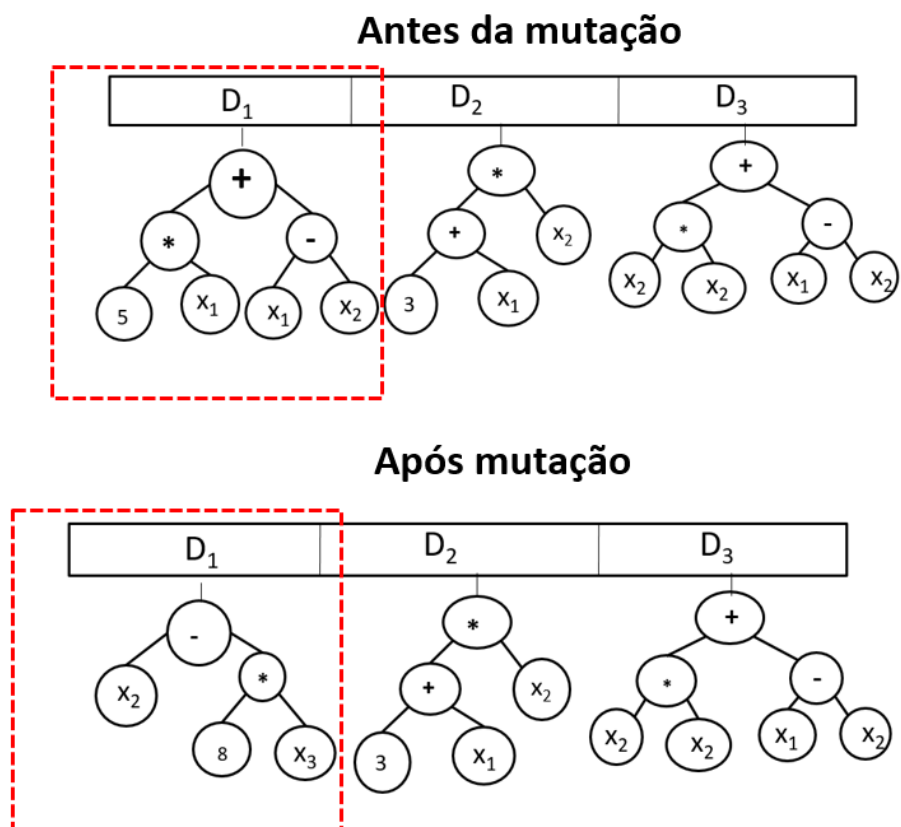


Figura 18 – Mutação no EvoNFuzz.

4.3 Processo de Aprendizagem do EvoNFuzz

O EvoNFuzz emprega um algoritmo evolucionário simples baseado em PG, utilizando as operações detalhadas na Seção 4.2.1 em conjunto com um algoritmo baseado no gradiente para ajuste dos parâmetros de rede: funções de pertinência e pesos sinápticos. Este processo ocorre de forma integrada, ou seja, para cada geração de programação genética, são realizados l épocas do gradiente para cada indivíduo, uma vez que cada indivíduo possui seus próprios parâmetros. A quantidade l de épocas por geração da PG é um parâmetro de entrada do algoritmo. Dessa forma, o método do gradiente auxilia na melhora do desempenho dos indivíduos a cada geração.

O modelo proposto possui duas fases de aprendizado: uma não supervisionada e outra supervisionada. Na fase não supervisionada, são determinadas as funções de pertinência que compõem os neurônios do componente de fuzzificação, sendo responsáveis pelo particionamento do universo de discurso das variáveis de entrada e pela definição dos conjuntos fuzzy. Nesta etapa, podem ser utilizados algoritmos de agrupamento, permitindo que a configuração inicial da rede já esteja adaptada aos dados de entrada.

A fase supervisionada, por sua vez, é responsável pela modelagem da base de regras e pelo ajuste dos parâmetros. A modelagem das regras que compõem o componente de inferência é realizada por meio de um método de PGMG, no qual cada indivíduo é denotado por um conjunto de árvores de representação. Para o ajuste dos parâmetros, é aplicado um método baseado no gradiente que atualiza os parâmetros das funções de pertinência e os pesos sinápticos a cada nova geração da PGMG, conforme mencionado anteriormente.

As funções de pertinência são definidas pelo algoritmo *K-Means* (IKOTUN et al., 2023). O *K-Means* é um algoritmo iterativo que particiona os dados em k grupos, minimizando a soma das distâncias quadradas entre cada ponto e o centróide do grupo correspondente. O método *elbow* (HUMAIRA; RASYIDAH, 2020) é empregado para determinar o número ideal de grupos com base nos dados de entrada. Esse método consiste em plotar a soma dos erros quadrados para diferentes valores de k e analisar a curva gerada. O “cotovelo” desta curva, representando o ponto de inflexão, é então interpretado como o valor mais apropriado para k .

Após a definição dos grupos pelo *K-Means*, eles são utilizados para a criação dos conjuntos fuzzy. O centróide $\bar{c}_{ij}$ de cada grupo é atribuído como o centro c_{ij} da função de pertinência j associada à variável de entrada i . Já o espalhamento s_{ij} da função de pertinência é proporcional à densidade do grupo, ou seja, ao número de elementos classificados dentro desse grupo, conforme indicado pelo vetor de rótulos (*labels*), que associa cada entrada a um grupo específico. Assim, s_{ij} assume um valor entre 0 e 1, representando a proporção de elementos pertencentes ao grupo, conforme:

$$s_{ij} = \frac{d_j}{N}, \quad (23)$$

na qual d_j representa o número de elementos classificados com o rótulo j e N corresponde ao total de amostras. Observe que, se todos os elementos pertencem a um único grupo, a dispersão será 1.0; se apenas metade dos elementos pertencer a esse grupo, a dispersão será 0.5. Essa abordagem simplifica a definição inicial dos parâmetros das funções de pertinência, permitindo o ajuste durante a fase supervisionada. O procedimento para criação das funções de pertinência é mostrado no Algoritmo 5.

Algoritmo 5: Geração das funções de pertinência - EvoNFuzz.

- 1 Aplica método *elbow* para definição do k ótimo;
 - 2 Aplica k-means com k ótimo encontrado;
 - 3 k-means retorna $\bar{c}$, *labels*;
 - 4 $c \leftarrow \bar{c}$;
 - 5 Calcula espalhamento das funções (23);
-

A construção da base de regras é realizada por um método de PGMG, no qual cada indivíduo é definido como um conjunto de árvores de representação, permitindo que um único indivíduo represente uma base de regras completa. No preditor proposto, cada indivíduo é estruturado como um conjunto de árvores, em que cada árvore corresponde a uma função consequente $f(\bullet)$ associada a um antecedente. Conforme discutido anteriormente nesta seção, cada função consequente é multiplicada pelo grau de ativação de seu antecedente. As funções são construídas a partir de operações básicas, como adição, subtração, multiplicação e divisão, além das variáveis de entrada e constantes numéricas. O número mínimo e máximo de regras por indivíduo na população inicial é um hiperparâmetro a ser definido pelo usuário.

A população inicial é gerada aleatoriamente. Primeiramente, são escolhidas as variáveis que compõem a regra e, para cada variável selecionada, um conjunto *fuzzy* é atribuído para formar o antecedente da regra. Em seguida, um operador lógico (*AND* ou *OR*) é escolhido aleatoriamente para completar o antecedente. Para definir o consequente, é gerada uma árvore aleatória (representando uma função matemática para uma regra *fuzzy* polinomial), e um peso aleatório entre 0 e 1 é atribuído. Vale destacar que todas as decisões na população inicial são tomadas de maneira aleatória. A geração da população inicial é resumida no Algoritmo 6.

Após a geração dos indivíduos, cada um deles deve ser avaliado de acordo com uma medida de desempenho (*fitness*). Para isso, é necessário estimar a saída da rede para cada indivíduo, conforme Algoritmo 7.

Após o processamento da saída de cada indivíduo, a função de desempenho pode ser aplicada, calculando-se uma medida de erro baseada na saída esperada. Neste trabalho, o

Algoritmo 6: Geração da população inicial - EvoNFuzz.

```
1 for  $i = 1 : T$  do
2   Definir aleatoriamente o número de regras  $n_r$ 
3   for  $k = 1 : z$  do
4     Seleciona aleatoriamente as variáveis que irão compor a regra
5     Seleciona aleatoriamente um conjunto fuzzy por variável escolhida
6     Selecionar aleatoriamente um operador AND ou OR
7      $Y(r) \leftarrow$  gera uma árvore aleatória para o conseqüente
8      $w(r) \leftarrow$  peso aleatório
9   end
10 end
```

Algoritmo 7: Cálculo da saída da rede - EvoNFuzz.

```
1 for  $t = 1 : N$  do
2   Calcula  $L_{ij}^1$  (17) com  $i = 1..n, j = 1..m$ ;
3   Calcula  $D_r$  (19) e (20) com  $r = 1..z$ ;
4   Calcula  $L_r^2$  (21) com  $r = 1..z$ ;
5   Calcula  $\hat{y}^t$  (22)
6 end
```

desempenho do indivíduo é avaliado por meio do MSE (*Mean Square Error*), definido por:

$$MSE = \frac{\sum_{t=1}^N (y^t - \hat{y}^t)^2}{N}, \quad (24)$$

em que y^t representa o valor real da saída para a t -ésima amostra, $\hat{y}^t$ é o valor predito pelo modelo e N é o número total de amostras.

A nova população é gerada por meio dos operadores genéticos: cruzamento, mutação e seleção natural. Para garantir que o melhor indivíduo seja preservado de uma geração para a outra, é utilizada uma abordagem elitista. Na PGMG, as operações de cruzamento e mutação podem ocorrer em dois níveis: alto ou baixo. Neste trabalho, ambas as operações foram realizadas em alto nível, o que significa que regras inteiras são alteradas tanto no cruzamento quanto na mutação. Como método de seleção natural, foi empregado o torneio binário, utilizado tanto para a seleção dos pais quanto para a escolha dos indivíduos que compõem a nova população. Além disso, conforme mencionado anteriormente, o operador de elitismo é aplicado antes do processo de seleção, garantindo que o melhor indivíduo da geração anterior seja replicado para a nova geração.

A cada geração, os parâmetros da rede devem ser ajustados para cada indivíduo. Esse ajuste é realizado usando um método baseado em gradiente, em que um número especificado de épocas de gradiente é executado para cada geração no processo de GP. Neste estudo, três épocas de gradiente foram usadas por geração para acelerar a convergência e evitar a convergência prematura do algoritmo. Os ajustes dos parâmetros são feitos de acordo com as seguintes equações:

$$c_{ij} = c_{ij} - \eta (y - \hat{y}) dw_r dc_{ij}, \quad (25)$$

$$s_{ij} = s_{ij} - \eta (y - \hat{y}) dw_r ds_{ij}, \quad (26)$$

$$w_r = w_r - \eta (y - \hat{y}) x_i, \quad (27)$$

em que η é a taxa de aprendizado, r indexa a regra, y é a saída desejada, $\hat{y}$ é a saída da rede e, dw_{ij} , dc_{ij} e ds_{ij} são obtidos respectivamente por:

$$dw_{ij} = \frac{\mu_{A_{ij}}}{\sum_{j=1}^{m_i} \mu_{A_{ij}}}, \quad (28)$$

$$dc_{ij} = \mu_{A_{ij}} \frac{x_i - c_{ij}}{(c_{ij})^2}, \quad (29)$$

$$ds_{ij} = \mu_{A_{ij}} \frac{x_i - s_{ij}}{(s_{ij})^2}. \quad (30)$$

O procedimento de aprendizagem completo do EvoNFuzz é ilustrado na Figura 19 e detalhado no Algoritmo 8.

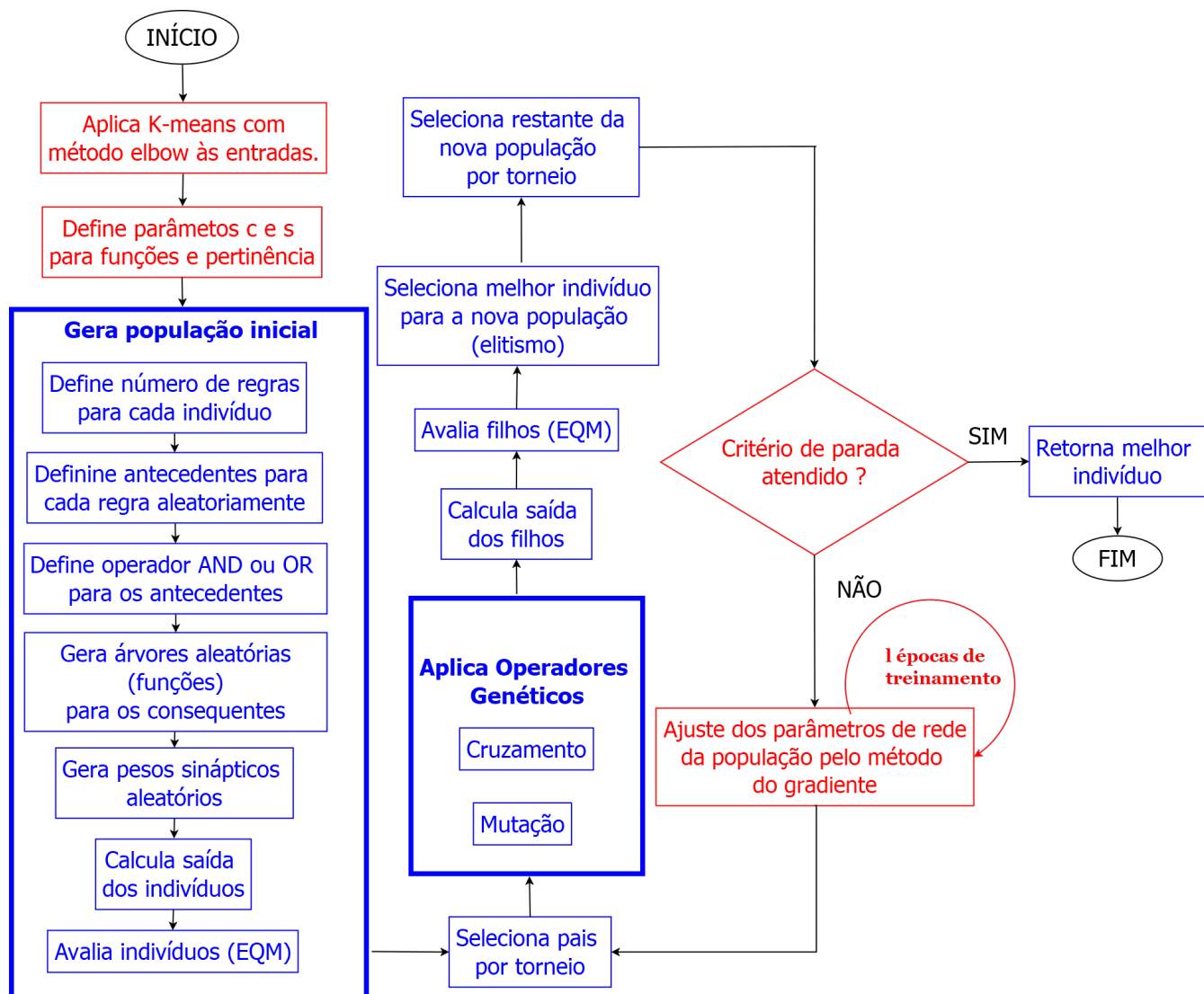


Figura 19 – Fluxograma do algoritmo de aprendizagem da rede EvoNFuzz.

Algoritmo 8: Algoritmo de aprendizado do EvoNFuzz.

```
1 Entrada:  $x^t, y^t, g, T, tx\_c, tx\_m,$   
2 Saída:  $\hat{y}^t$   
3 Gera funções de pertinência (Algoritmo 5);  
4 Gera população inicial (Algoritmo 6);  
5 for  $individual = 1 : T$  do  
6   | Calcula fitness (24);  
7   | Processa saída da rede (Algoritmo 7);  
8 end  
9 for  $geracao = 1 : g$  do  
10  | for  $individual = 1 : T$  do  
11  |   | for  $epoch = 1 : l$  do  
12  |   |   | Calcula  $dw_{ij}$  (28),  $dc_{ij}$  (29), e  $ds_{ij}$  (30);  
13  |   |   | Atualiza  $c_{ij}$  (25) e  $s_{ij}$  (26);  
14  |   |   | Atualiza  $w(r)$  (27) com  $r = 1...z, i = 1..n$ ;  
15  |   | end  
16  |   | Processa saída da rede (Algoritmo 7);  
17  |   | Calcula fitness (24) ;  
18  | end  
19  | Aplicar elitismo;  
20  | for  $individual = 1 : T/2$  do  
21  |   | Pai1  $\leftarrow$  SelecaoTorneio(PopulaçãoAtual,Trn)  
22  |   | Pai2  $\leftarrow$  SelecaoTorneio(PopulaçãoAtual,Trn)  
23  |   | Filho1  $\leftarrow$  Crossover(Pai1,Pai2)  
24  |   | Filho2  $\leftarrow$  Crossover(Pai2,Pai1)  
25  |   | Filho1  $\leftarrow$  Mutation(Filho1)  
26  |   | Filho2  $\leftarrow$  Mutation(Filho2)  
27  | end  
28  | Calcular fitness (24) ;  
29  | Seleciona nova população por torneio;  
30 end
```

Capítulo 5

DGP-NF - *Dynamic Genetic Programming-based Neuro-Fuzzy Network*

Este capítulo apresenta a rede DGP-NF - *Dynamic Genetic Programming-based Neuro-Fuzzy Network*-, proposta neste trabalho. DGP-NF é uma rede neuro-fuzzy preditiva com regras polinomiais e aprendizado baseado em programação genética multi-gene (PGMG) com população dinâmica e controle de diversidade, integrado a um algoritmo de aprendizagem incremental baseado em mínimos quadrados ponderados. A estrutura de rede, modelagem do indivíduo e operações genéticas são as mesmas propostas e empregadas na EvoNFuzz (Capítulo 4), diferenciando-se pelo algoritmo de ajuste de parâmetros de rede e pelas estratégias evolucionárias para exploração do espaço de busca que são implementadas na DGP-NF. Contudo, para maior entendimento e reprodutibilidade deste trabalho, a estrutura e conceitos envolvidos serão explicados novamente neste capítulo.

Diferente da EvoNFuzz, a DGP-NF emprega um algoritmo de aprendizagem incremental baseado em mínimos quadrados ponderados para ajuste dos parâmetros de rede, evitando assim a execução de várias épocas de treinamento por geração da PG, já que o algoritmo de aprendizagem incremental realiza apenas um passo por iteração. Além disso a DGP-NF utiliza algumas técnicas para favorecer a exploração do espaço de busca e evitar a convergência prematura, que serão apresentadas na Seção 5.3, e eliminação de regra redundantes, que será apresentada na Seção 5.4. Assim, pode-se citar com principais novidades e contribuições do modelo DGP-NF:

- Um novo modelo neuro-fuzzy evolucionário com regras polinomiais e pesos sinápticos para os consequentes, visando aprimorar a identificação de relações complexas e não lineares entre as variáveis de entrada e a precisão do modelo;
- Um novo algoritmo de aprendizado híbrido baseado em Programação Genética Multi-

Gene para a criação das regras e um algoritmo de aprendizado incremental baseado em mínimos quadrados ponderados para o ajuste dos parâmetros da rede;

- Um mecanismo para maior exploração do espaço de busca em sistemas *fuzzy* evolucionários, evitando também a convergência prematura, baseado em população dinâmica e controle de diversidade populacional.
- Um novo operador de mutação que promove e garante a diversidade fenotípica entre os indivíduos gerados;
- Um operador de poda mais simples baseado na análise da diversidade fenotípica do indivíduo;

A seguir, a Seção 5.1 descreve a estrutura de rede proposta neste capítulo; a Seção 5.2 explica o processo de modelagem da base de regras com programação genética multi-gene; já na Seção 5.3 são apresentadas as técnicas para aumento e diminuição dinâmica do tamanho da população e controle de diversidade populacional propostas; a Seção 5.4 apresenta o operador de poda baseado em diversidade fenotípica proposto e; na Seção 5.5 o processo de aprendizagem do DGP-NF é detalhado.

5.1 Estrutura da Rede DGP-NF

Composta por três componentes - Fuzzificação, Inferência e Agregação - a DGP-NF é uma rede evolucionária, da qual integra a estrutura de indivíduos genéticos à sua arquitetura conexonista para modelagem da base de regras. Além disso, inspirada nas redes neurais, ela também incorpora os conceitos de pesos sinápticos para definir diferentes graus de importância para cada regra e favorecer o processo de generalização da rede. A Figura 20 detalha a estrutura da rede DGP-NF.

Componente de Fuzzificação (L^1): Este componente é composto por uma única camada de neurônios *fuzzy* responsáveis por particionar o universo de discurso das variáveis de entrada através de funções de pertinência gaussianas. Cada neurônio representa um conjunto *fuzzy* A_{ij} referente à variável x_i que está ligada a ele. Assim, este componente realiza fuzzificação das variáveis de entrada e a saída de cada neurônio (L_{ij}^1) será o grau de pertinência μ da variável x_i no respectivo conjunto fuzzy A_{ij} :

$$L_{ij}^1 = \mu_{A_{ij}}(x_i), \quad (31)$$

em que x é o vetor de entrada $[x_1^t \cdots x_i^t \cdots x_n^t]^T$, sendo t é o passo atual e n a dimensão

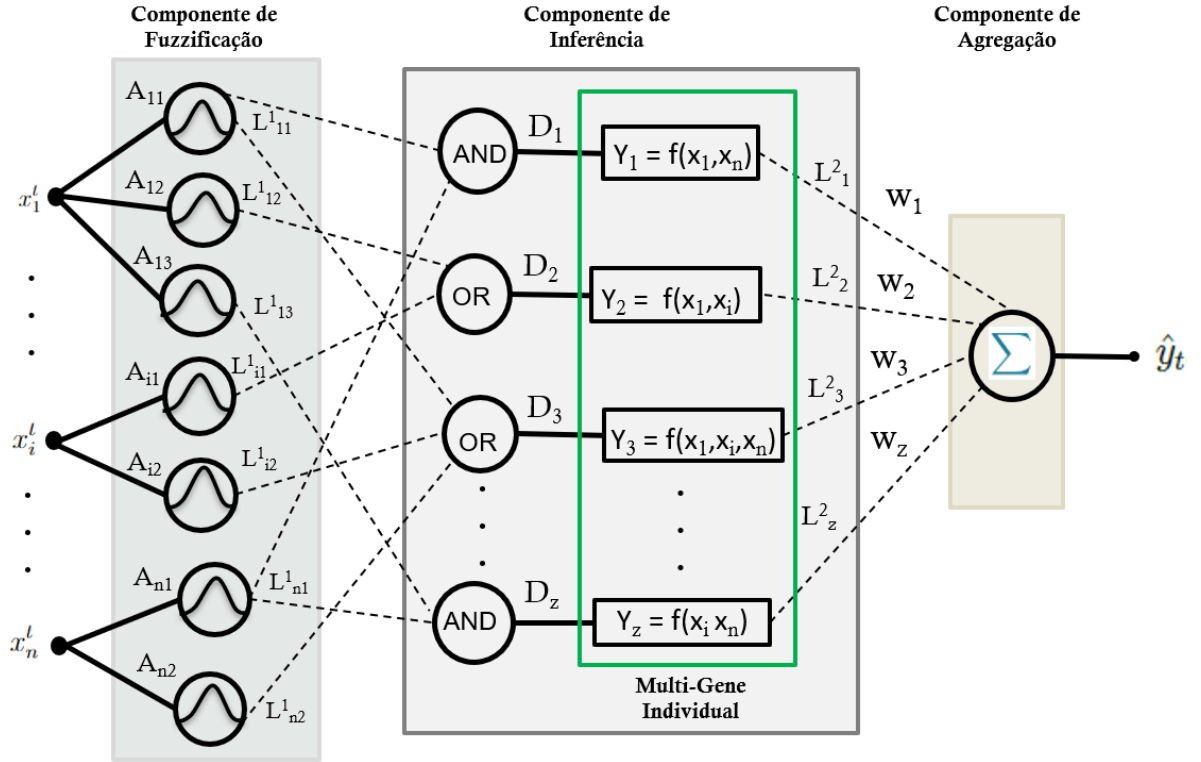


Figura 20 – Estrutura da rede DGP-NF.

de x , i indexa as variáveis de entrada e j indexa a função de pertinência. Como as funções de pertinência são gaussianas, o grau de ativação $\mu_{A_{ij}}(x_i)$ é calculado por:

$$\mu_{A_{ij}}(x_i) = \exp\left(-\frac{1}{2}\left(\frac{x_i - c_{ij}}{s_{ij}}\right)^2\right), \quad (32)$$

em que c_{ij} e s_{ij} são, respectivamente, o centro e o espalhamento da função de pertinência j para a variável i .

Componente de Inferência (L^2): O componente de Inferência encapsula a base de regras da rede DGP-NF. É nele que a estrutura evolucionária da PGMG é integrada à estrutura conexionista da rede neuro-fuzzy. As regras SE-ENTÃO, modeladas neste componente, possuem antecedentes *fuzzy* e consequentes polinomiais. Este componente é composto por duas camadas, sendo a primeira formada por neurônios que contém operadores lógicos AND e OR que recebem as entradas fuzzificadas pelo componente anterior e definem os antecedentes das regras. A saída de cada neurônio é o grau de disparo D_r de cada antecedente, que é dado pela operação *min* no caso das regras AND, e pela operação *max* no caso das regras OR, conforme equações 33 e 34.

$$D_r = \min(L_{11}^1, L_{n1}^1), \quad (33)$$

$$D_r = \max(L_{12}^1, L_{i1}^1, L_{n2}^1). \quad (34)$$

em que r indexa a regra.

Já a segunda camada é composta por um indivíduo multi-gene que modela as funções $f(\bullet)$ que estimam os consequentes Y_r de cada regra. Na estrutura final da rede, este indivíduo será o melhor indivíduo retornado após a última geração da PG. Desta forma, unindo-se as duas camadas deste componente, as regras na DPG-NF ficam no seguinte formato:

R_1 :

$$\begin{array}{c} \text{SE } \overbrace{x_1 \text{ é } A_{13}}^{L_{13}^1 = \mu_{A_{11}}(x_1)} \text{ E } \overbrace{x_n \text{ é } A_{32}}^{L_{32}^1 = \mu_{A_{32}}(x_3)} \\ \text{ENTÃO } Y_1 = 3x_1 + 2x_3. \end{array}$$

R_2 :

$$\begin{array}{c} \text{SE } \overbrace{x_1 \text{ é } A_{11}}^{L_{11}^1 = \mu_{A_{11}}(x_1)} \text{ OU } \overbrace{x_2 \text{ é } A_{23}}^{L_{23}^1 = \mu_{A_{23}}(x_2)} \text{ OU } \overbrace{x_3 \text{ é } A_{32}}^{L_{32}^1 = \mu_{A_{32}}(x_3)} \\ \text{ENTÃO } Y_r = 2(x_2 - 3x_3) + 2x_3/(x_1 * 5x_2). \end{array}$$

em que R representa uma regra *fuzzy*, r indexa a regra e Y denota o consequente polinomial $f(x)$. Cada regra pode ter uma ou mais variáveis de entrada envolvidas, sendo que apenas 1 conjunto *fuzzy* de cada variável pode estar associado a esta regra.

Após estimar o consequente Y_r de uma regra, ele deve ser multiplicado pelo grau de disparo D_r do seu antecedente. Assim, a saída do Componente de Inferência (L^2) é dada por:

$$L_r^2 = D_r Y_r, \quad (35)$$

em que r indexa a regra.

Componente de Agregação (L^3): O terceiro e último componente realiza a agregação das regras ativadas e estima a saída da rede. É composto por um único neurônio, que se

conecta ao componente anterior por meio de pesos sinápticos. Os pesos sinápticos w estão relacionados a cada saída do componente de inferência e determina o grau de importância de cada regra na base de regras e, conseqüentemente, seu grau de influência na saída da rede. Desta forma, a saída final do modelo é calculada por média ponderada das regras ativadas e seus respectivos pesos sinápticos, como mostrado na Equação (36).

$$\hat{y} = L^3 = \frac{\sum_{r=1}^z L_r^2 w_r}{\sum_{r=1}^z w_r}. \quad (36)$$

em que z é o número de regras ativadas e r indexa a regra.

5.2 Modelagem da Base de Regras com Programação Genética

A base de regras na DGP-NF, como já mencionado, é modelada por um método de programação genética multi-gene (PGMG) (FERREIRA, 2001), que define as regras presentes no componente de inferência. Diferente da PG tradicional, onde cada indivíduo é representado por uma única árvore, a PGMG modela cada indivíduo como um conjunto de árvores e representação (subárvores) ligadas a uma estrutura linear superior (genes superiores) (NIAZKAR, 2023). A saída de um indivíduo é obtida pela agregação das saídas de todas as subárvores que o compõem.

Esta representação é particularmente adequada para modelar bases de regras *fuzzy*, uma vez que regras isoladas podem não ter tanta relevância para serem avaliadas individualmente e nem fornecer informações suficientemente relevantes para a saída. Para contornar essa limitação, a abordagem de Pittsburgh (SMITH, 1980) é adotada, na qual cada indivíduo representa uma base de regras completa. Essa estrutura facilita a avaliação do desempenho de cada base de regras durante o processo evolucionário. Dessa forma, a PGMG pode ser uma estratégia eficaz para realizar a modelagem automática de bases de regras em redes *fuzzy* evolucionárias.

5.2.1 Representação do indivíduo e operadores genéticos

No DGP-NF cada indivíduo representa uma base de regras completa, modelado através de um conjunto de árvores, onde cada árvore representa uma função de conseqüente para uma regra, como mostrado anteriormente na Figura 20. Desta forma, o método de PGMG é responsável por criar e evoluir os conseqüentes das regras. É importante ressaltar que as funções contidas no indivíduo estão também relacionadas aos seus respectivos antecedentes e pesos sinápticos, que são atualizados pelo algoritmo de aprendizagem incremental e não pela PG. Um exemplo de indivíduo multi-gene na DGP-NF pode ser visto

na Figura 21 que exemplifica um indivíduo composto por 3 regras *fuzzy*, onde a função de consequente de cada uma delas, respectivamente, é:

$$R_1: Y_1 = (x_3 - 8) * x_2.$$

$$R_2: Y_2 = (3x_1)/(x_3 + x_2).$$

$$R_3: Y_3 = (x_2 * x_2)/3).$$

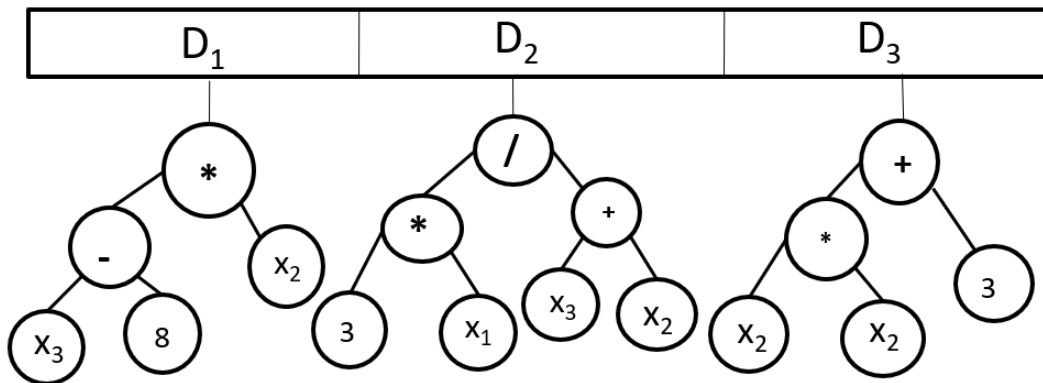


Figura 21 – Indivíduo multi-gene no DGP-NF.

No exemplo, a regra 1 está associada às variáveis x_2 e x_3 , a regra 2 está ligada às 3 variáveis e a regra 3, apenas à variável x_2 . Também é importante observar que, apesar de usar apenas operações matemáticas básicas nas funções do indivíduo, é possível, com esta estrutura, obter fatores de outros tipos, por exemplo fatores exponenciais, como visto na R_3 onde o termo $x_2 * x_2$ é equivalente à $(x_2)^2$.

Na PGMG as operações de cruzamento e mutação podem ser realizadas em alto nível - afetando os genes superiores e, consequentemente sub-árvores inteiras -, ou em baixo nível - manipulando partes das sub-árvores. Na rede DGP-NF as duas operações são realizadas em alto nível, sendo responsável por alterar regras inteiras contidas na base de regras daquele indivíduo, mas não a estrutura destas regras. Ou seja, no cruzamento os filhos herdam as regras já existentes nos pais, recombinaando-as, e na mutação uma regra escolhida será substituída por uma nova regra gerada. Desta forma, o cruzamento é responsável por melhorar a base de regras a partir de regras já conhecidas e já existentes na população, enquanto a mutação é responsável por inserir novas regras ainda não existentes na população. As figuras 22 e 23 mostram a operações de cruzamento e mutação na DGP-NF.

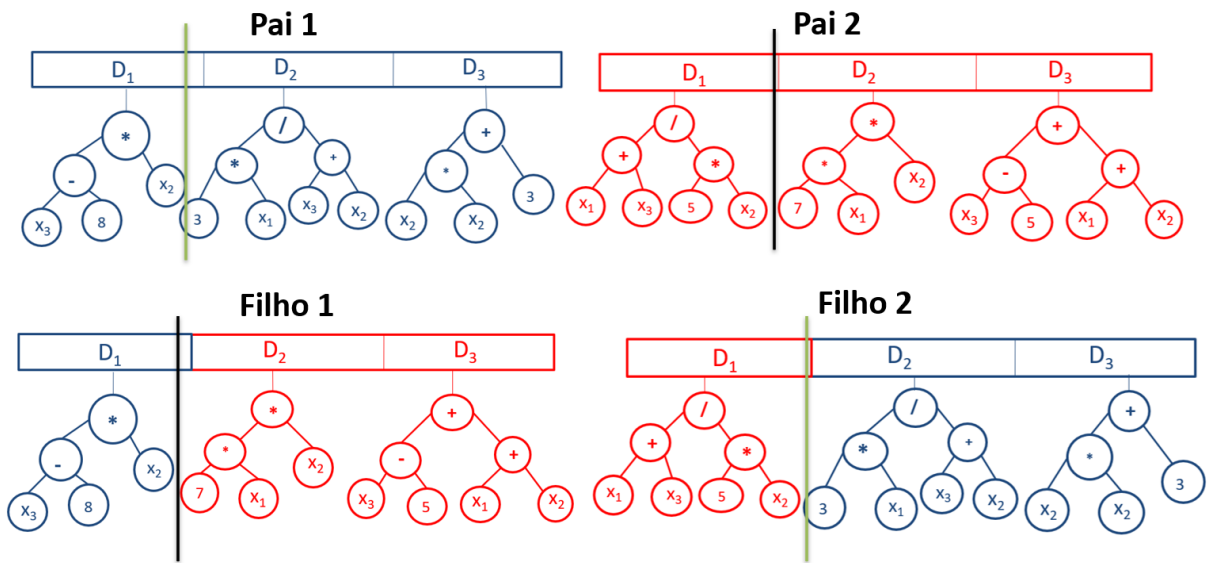


Figura 22 – Cruzamento no DGP-NF.

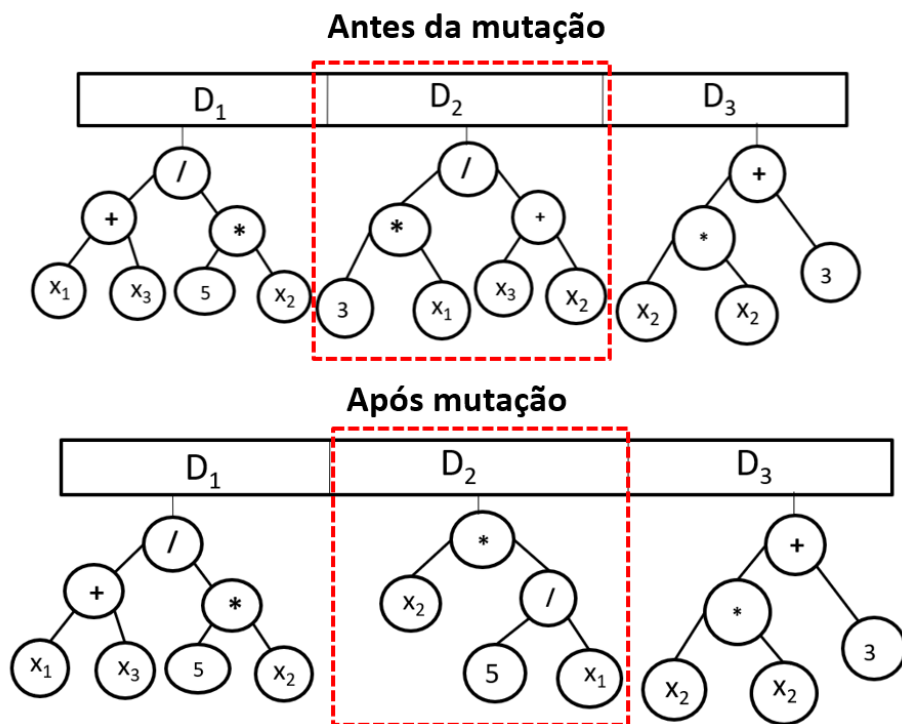


Figura 23 – Mutação no DGP-NF.

5.3 População Dinâmica e Controle de Diversidade

Com o objetivo de melhorar a exploração do espaço de busca e evitar a convergência prematura, o DGP-NF adota dois mecanismos complementares: populações dinâmicas e controle de diversidade. Neste trabalho, optou-se pelo uso da diversidade fenotípica, que já

demonstrou ser mais relevante do que a diversidade genotípica na melhoria da acurácia e da qualidade da solução final em algoritmos evolucionários, especialmente em problemas de regressão (BURKE; GUSTAFSON; KENDALL, 2002; FARINATI; BAKUROV; VANNESCHI, 2023).

Essa escolha se justifica pelo fato de que indivíduos com estruturas internas distintas podem gerar respostas muito semelhantes. Assim, o foco do modelo está em promover a diversidade das respostas produzidas pelos indivíduos, uma vez que essa característica tem maior impacto no desempenho final da rede.

A diversidade entre indivíduos é mensurada por meio da distância euclidiana média entre as saídas que eles produzem para todas as amostras do conjunto de treinamento. A distância euclidiana E entre dois indivíduos i e j é dada por:

$$E_{ij} = \frac{\sqrt{\sum_{t=1}^N (y_i^t - y_j^t)^2}}{N}, \quad (37)$$

em que N é o número de amostras. A taxa de diversidade de cada indivíduo expressa o quão diferente ele é dos demais indivíduos da população e é definida como:

$$E_i = \frac{\sum_{j=1}^{T-1} \sum_{k=j+1}^T E_{ik}}{T}, \quad (38)$$

na qual T é o tamanho da população.

A taxa de diversidade populacional é dada pela média da taxa de diversidade de todos os indivíduos:

$$E = \frac{\sum_{i=1}^T E_i}{T}, \quad (39)$$

Para o ajuste do tamanho da população, este trabalho adota uma abordagem baseada na variação da diversidade entre gerações consecutivas. A escolha dessa estratégia se fundamenta no comportamento típico observado ao final do processo evolucionário, quando a diversidade tende naturalmente a diminuir em razão da convergência do algoritmo. Nesses casos, mesmo que o tamanho da população seja aumentado, a diversidade pode não se elevar de forma significativa, pois os novos indivíduos gerados tendem a ser similares aos já existentes.

Dessa forma, utilizar a diferença de diversidade como critério de ajuste permite ao algoritmo identificar com mais precisão o ponto de convergência e evita expansões desnecessárias

da população, contribuindo para maior estabilidade e eficiência do processo evolutivo. Para isso, o algoritmo aumenta o tamanho da população sempre que a diferença de diversidade entre a população atual e a anterior estiver abaixo de um limiar inferior λ (indicando possível estagnação) e reduz o tamanho da população quando essa diferença estiver acima de um limiar superior σ (indicando excesso de variação desnecessária). Esses dois limiares (λ e σ), assim como a taxa de variação da população (Δ), que define a porcentagem de indivíduos a serem adicionados ou removidos, são parâmetros de entrada do algoritmo. Além disso, é necessário definir valores mínimos (ϵ) e máximos (ω) para o tamanho da população, a fim de garantir limites operacionais adequados ao processo evolucionário.

Para manter a diversidade da população, é fundamental que os novos indivíduos gerados não sejam excessivamente semelhantes aos já existentes. Com esse objetivo, este trabalho propõe um mecanismo de controle de diversidade baseado no operador de mutação. Esse operador realiza a mutação apenas se a similaridade entre o indivíduo gerado e o original estiver acima de um limiar inferior de diversidade, evitando assim a criação de indivíduos praticamente idênticos aos anteriores. O operador de mutação é aplicado tanto na modificação de indivíduos já existentes na população quanto na geração de novos indivíduos, nos casos em que o algoritmo determina o aumento do tamanho populacional.

Quando há um aumento na população, metade dos novos indivíduos é gerada por meio da mutação de indivíduos com boa aptidão, previamente selecionados por um operador de seleção, sem excluir os indivíduos selecionados. A outra metade é composta por indivíduos gerados aleatoriamente. Essa estratégia busca equilibrar a exploração do espaço de busca, ao inserir soluções totalmente novas, com a exploração localizada em torno de indivíduos promissores.

De maneira análoga, na redução da população, metade dos indivíduos excluídos é selecionada aleatoriamente, desde que estejam abaixo do limiar inferior de diversidade (indicando baixa contribuição para a diversidade populacional). A outra metade é excluída com base na aptidão, utilizando o mesmo operador de seleção, porém removendo o indivíduo perdedor do torneio, em vez do vencedor. Essa abordagem combina a eliminação de indivíduos redundantes com a exclusão de indivíduos de baixo desempenho, promovendo uma população mais enxuta e eficiente.

O processo completo de ajuste do tamanho da população é apresentado no Algoritmo 9.

5.4 Poda Baseada em Diversidade Fenotípica

Com o objetivo de eliminar regras redundantes e, ao mesmo tempo, obter uma base de regras mais enxuta, este trabalho propõe um operador de poda baseado na análise de diversidade intra-indivíduo. Considerando que um indivíduo multi-gene é composto por

Algoritmo 9: Ajuste do tamanho da população

```
1  Calcular  $E_{ij}$  (37) com  $i = 1...N$ ,  $j = i + 1...N$ 
2  Calcular  $E_i$  (38) com  $i = 1...T - 1$ ,  $j = i + 1...T$ 
3  Calcular  $E$  (39)
4  if  $E > \sigma$  then
5      reduzpop =  $T * \Delta$ ;
6      size_pop =  $T - \text{reduzpop}$ ;
7      while  $i < \text{reduzpop}$  do
8          if  $i < (\text{reduzpop}/2)$  then
9              del1  $\leftarrow$  Posição aleatória na população;
10             if  $E_{del1} < \lambda$  then
11                 Deletar pop(del1);
12                 i++;
13             end
14         end
15     else
16         del1  $\leftarrow$  SelecaoTorneio(PopulacaoAtual,Trn);
17         del2  $\leftarrow$  SelecaoTorneio(PopulacaoAtual,Trn);
18         if pop(del1).fitness > pop(del2).fitness then
19             if  $E_{del1} < \lambda$  then
20                 Deletar pop(del1);
21                 i++;
22             end
23         end
24     else
25         if  $E_{del1} < \lambda$  then
26             Deletar pop(del2);
27             i++;
28         end
29     end
30 end
31 end
32 end
33 if  $E < \lambda$  then
34     aumentapop =  $T * \Delta$ ;
35     size_pop =  $T + \text{aumentapop}$ ; for  $i = 1 : \text{aumentapop}$  do
36         if  $i < (\text{reduzpop}/2)$  then
37             Gerar um Indivíduo Aleatório;
38         end
39     else
40         pai  $\leftarrow$  SelecaoTorneio(PopulacaoAtual,Trn);
41         novoIndividuo  $\leftarrow$  Mutacao(pop(pai));
42     end
43 end
44 end
```

um conjunto de árvores de representação, no qual cada árvore corresponde a uma regra, é possível tratar cada indivíduo como uma subpopulação de regras. Para simplificar o processo de poda e análise de redundâncias, o operador de poda proposto neste trabalho utiliza o mesmo processo de análise de diversidade apresentado na Seção 5.3 para avaliar a diversidade das regras contidas em um indivíduo.

Nesse contexto, em vez de se analisar a saída final do indivíduo como um todo, avaliam-se as saídas individuais de cada gene superior, ou seja, de cada regra que compõe o indivíduo. Regras que apresentam respostas muito semelhantes são consideradas redundantes e são eliminadas. Este operador é aplicado apenas ao melhor indivíduo da população, uma vez que é ele quem será selecionado ao final da última geração.

Para essa análise, utiliza-se uma adaptação da Equação (37), pois o interesse está na similaridade entre regras, e não na taxa global de diversidade de um indivíduo. Nessa adaptação, o indivíduo é tratado como se fosse uma população, e cada uma de suas regras é considerada como um "indivíduo" dentro dessa subpopulação. A medida de similaridade entre duas regras i e j de um mesmo indivíduo é, portanto, calculada conforme a Equação (40).

$$E_{ij} = \frac{\sqrt{\sum_{r=1}^z (L_{ir}^2 - L_{jr}^2)^2}}{z}, \quad (40)$$

em que z é o número de regras do indivíduo, e L_r^2 é a saída da regra, conforme mostrado anteriormente na Equação (35).

O processo de poda do indivíduo é descrito no Algoritmo 10

Algoritmo 10: Poda do indivíduo

```

1 Calcular  $E_{ij}$  (37) com  $i = 1...z$ ,  $j = 1..z$ 
2 if  $E_{ij} \leq \lambda$  then
3   | Deletar regra  $j$ ;
4 end

```

5.5 Processo de Aprendizagem do DGP-NF

O aprendizado na rede DGP-NF inicia-se com um algoritmo de agrupamento para particionamento do universo de discurso e definição das funções de pertinência que comporão os neurônios do primeiro componente da rede (componente de fuzzificação). Qualquer algoritmo de agrupamento pode ser utilizado nesta fase para permitir que a configuração inicial da rede já esteja adaptada aos dados de entrada.

Neste trabalho foi empregado *K-Means* (IKOTUN et al., 2023) para este propósito, em conjunto com o método *elbow* (HUMAIRA; RASYIDAH, 2020) para definição da quantidade ideal de grupos para cada variável, sem que este precise ser um parâmetro de entrada arbitrário. O *K-Means* particiona os dados em k grupos, minimizando a distância quadrada entre os pontos e o centróide do seu grupo. Já o método *elbow* plota a soma dos erros quadrados para diferentes valores de k e identifica o ponto de inflexão (cotovelo). O ponto de inflexão é considerado o K -ótimo para ser utilizado no algoritmo.

Após aplicação do *K-Means*, os grupos encontrados são utilizados para definição dos conjuntos *fuzzy* e modelagem das funções de pertinência. Desta forma, o centro c_{ij} de cada função de pertinência j associada à variável i equivalerá ao centróide $\bar{c}_{ij}$ definido pelo *K-Means* para aquele grupo. Já o espalhamento s_{ij} será proporcional à densidade do grupo definido pelo *K-Means*, isto é, a quantidade de elementos que compõe aquele

grupo, considerando o vetor de rótulos (*labels*) retornado pelo algoritmo, que determina o grupo designado para cada elemento. Desta forma s_{ij} será um valor entre 0 e 1 que reflete a densidade daquele grupo e será dado por:

$$s_{ij} = \frac{d_j}{N}, \quad (41)$$

em que d_j representa o número de elementos pertencentes ao grupo j e N representa o número de amostras. Pode-se perceber que, quanto mais elementos em um determinado grupo, maior será a dispersão desta função de pertinência e, conseqüentemente, maior a abrangência deste conjunto *fuzzy*. Por exemplo, se todos os elementos pertencerem a um único grupo, o espalhamento será 1.0; se apenas metade dos elementos pertencerem a esse grupo, o espalhamento será 0.5. O objetivo central desta abordagem é simplificar a definição inicial das funções de pertinência, que ainda serão ajustadas e otimizadas na fase supervisionada. Algoritmo 11 resume a etapa de definição das funções de pertinência.

Algoritmo 11: Geração das funções de pertinência - DGP-NF.

- 1 Aplica método *elbow* para definição do k ótimo;
 - 2 Aplica k-means com k ótimo encontrado;
 - 3 k-means retorna $\bar{c}$, *labels*;
 - 4 $c \leftarrow \bar{c}$;
 - 5 Calcula espalhamento das funções (41);
-

Após definição das funções de pertinência, inicia-se a fase supervisionada do aprendizado da DGP-NF. Esta fase integra o aprendizado evolucionário, através da PGMG para definição da base de regras, com um método de aprendizagem incremental baseado em mínimos quadrados ponderados para ajuste dos parâmetros de rede, isto é, funções de pertinência e pesos sinápticos.

O processo evolucionário para modelagem da base de regras inicia-se com a geração da população inicial, onde cada indivíduo da população representará uma possível base de regra para a rede, conforme já detalhado na Seção 5.2. Cada indivíduo na população inicial é gerado de forma aleatória. Define-se o número de regras que aquele indivíduo terá e para cada regra: são escolhidas as variáveis que estarão associadas a ela e então, para cada variável, um conjunto *fuzzy* será escolhido para compor o antecedente da regra; um operador lógico (*AND* ou *OR*) é escolhido aleatoriamente para conectar os conjuntos *fuzzy* selecionados e completar o antecedente da regra; gerada uma árvore aleatória que modela a função de conseqüente daquela regra; por fim um peso aleatório entre 0 e 1 é atribuído. Vale destacar que todas essas decisões na população inicial são aleatórias. A procedimento de geração da população inicial é apresentado no Algoritmo 12.

Cada indivíduo deve ser avaliado através de uma medida de desempenho (*fitness*). O desempenho está relacionado a saída de rede estimada por ele e ao erro obtido. Neste

Algoritmo 12: Geração da população inicial - DGP-NF.

```
1 for  $i = 1 : T$  do
2   Definir aleatoriamente o número de regras  $z$ 
3   for  $k = 1 : z$  do
4     Seleciona aleatoriamente as variáveis que irão compor a regra
5     Seleciona aleatoriamente um conjunto fuzzy por variável escolhida
6     Seleciona aleatoriamente um operador AND ou OR
7      $Y(r) \leftarrow$  gera uma árvore aleatória para o consequente
8      $w(r) \leftarrow$  peso aleatório
9   end
10 end
```

trabalho o desempenho do indivíduo é dado pelo MSE (*Mean Square Error*):

$$MSE = \frac{\sum_{t=1}^N (y^t - \hat{y}^t)^2}{N}, \quad (42)$$

em que y^t representa o valor esperado para a saída da t -ésima amostra, $\hat{y}^t$ é o valor estimado pelo modelo e N é o número total de amostras.

Para calcular o erro obtido por cada indivíduo é necessário estimar a saída de rede através das regras modeladas por ele. A cálculo da saída é realizada conforme o Algoritmo 13.

Algoritmo 13: Cálculo da saída da rede - DGP-NF.

```
1 for  $t = 1 : N$  do
2   Calcula  $L_{ij}^1$  (31) com  $i = 1...n$ ,  $j = 1...m$ ;
3   Calcula  $D_r$  (33) e (34) com  $r = 1...z$ ;
4   Calcula  $L_r^2$  (35) com  $r = 1...z$ ;
5   Calcula  $\hat{y}^t$  (36)
6 end
```

Após geração da população inicial, são executadas g gerações para evolução da estrutura de rede. A cada geração é gerada uma nova população através dos operadores genéticos: cruzamento, mutação e seleção natural já apresentados anteriormente. Para seleção, o método utilizado foi o do torneio binário, discutido na Seção 2.1.3. As operações de cruzamento e mutação são realizadas conforme mostrado na Seção 5.2.1. Além disso a abordagem proposta emprega um operador de elitismo para garantir que o melhor indivíduo não será perdido entre uma geração e outra, já que todo este processo é probabilístico.

O ajuste dos parâmetros da rede é realizado, para cada indivíduo a cada geração, por meio de um método de aprendizagem incremental baseado em mínimos quadrados ponderados (LUGHOFFER, 2016). O método dos mínimos quadrados ponderados estima a influência de cada amostra na modelagem do sistema e ajusta os parâmetros com base na amostra atual, considerando as informações das amostras anteriores. O uso dessa abordagem elimina-se a necessidade da execução de múltiplas épocas de treinamento, como ocorre em métodos tradicionais baseados no gradiente. Dessa forma, o ajuste dos parâmetros c , s e w é realizado por:

$$\theta_i^t = \theta_i^{t-1} + H_i^t x^{t-1} \bar{\mu}_x^{t-1} (y_i^t - (x^{t-1}) h_i^{t-1}), \quad (43)$$

em que θ_i^t representa o parâmetro ajustável (c , s ou w), i indexa a variável de entrada, t denota o instante de tempo, $\bar{\mu}_x^{t-1}$ é a função de pertinência normalizada para a variável x , e H é a matriz de atualização, que representa a influência relativa de cada amostra na estimativa. A matriz H é obtida por:

$$H_i^t = H_i^t - \frac{\bar{\mu}_x^{t-1} H_i^t x^{t-1}}{1 + x^{t-1} H_i^t}. \quad (44)$$

As derivações e adaptações completas destas fórmulas para atender as especificidades do problema de modelagem fuzzy, podem ser vistos em (LUGHOFFER, 2008) e (LUGHOFFER, 2020).

O Algoritmo 14 e a Figura 24 mostram, passo a passo, o processo de aprendizagem do DGP-NF.

Algoritmo 14: Algoritmo de aprendizado do DGP-NF.

```

1  Entrada  $x^t, y^t, g, T, tx\_c, tx\_m, min\_nr, max\_nr$ 
2  Saída  $\hat{y}^t$ 
3  Gera funções de pertinência (Algoritmo 11);
4  Gera população inicial (Algoritmo 12);
5  for  $individual = 1 : T$  do
6      | Calcula  $fitness$  (42);
7      | Processa saída da rede (Algoritmo 13);
8  end
9  //Gerações
10 for  $ger = 1 : g$  do
11     for  $p = 1 : T$  do
12         | Calcula  $H_i^t$  (44)
13         | Atualiza  $c_{ij}, s_{ij}, w(k)$  (43)
14         | Calcula  $L_{ij}^1$  (31),  $D_k$  (33-34),  $L_k^2$  (35)
15         | Calcula a saída  $\hat{y}^t$  (36)
16         | Calcula  $fitness$  (42);
17     end
18     Aplicar elitismo;
19     for  $individual = 2 : (T/2)$  do
20         | Aplicar cruzamento e mutação nos pares selecionados;
21     end
22     Selecionar nova população por torneio;
23     Ajustar tamanho da população (Algoritmo 9);
24     Podar melhor indivíduo (Algoritmo 10);
25 end

```

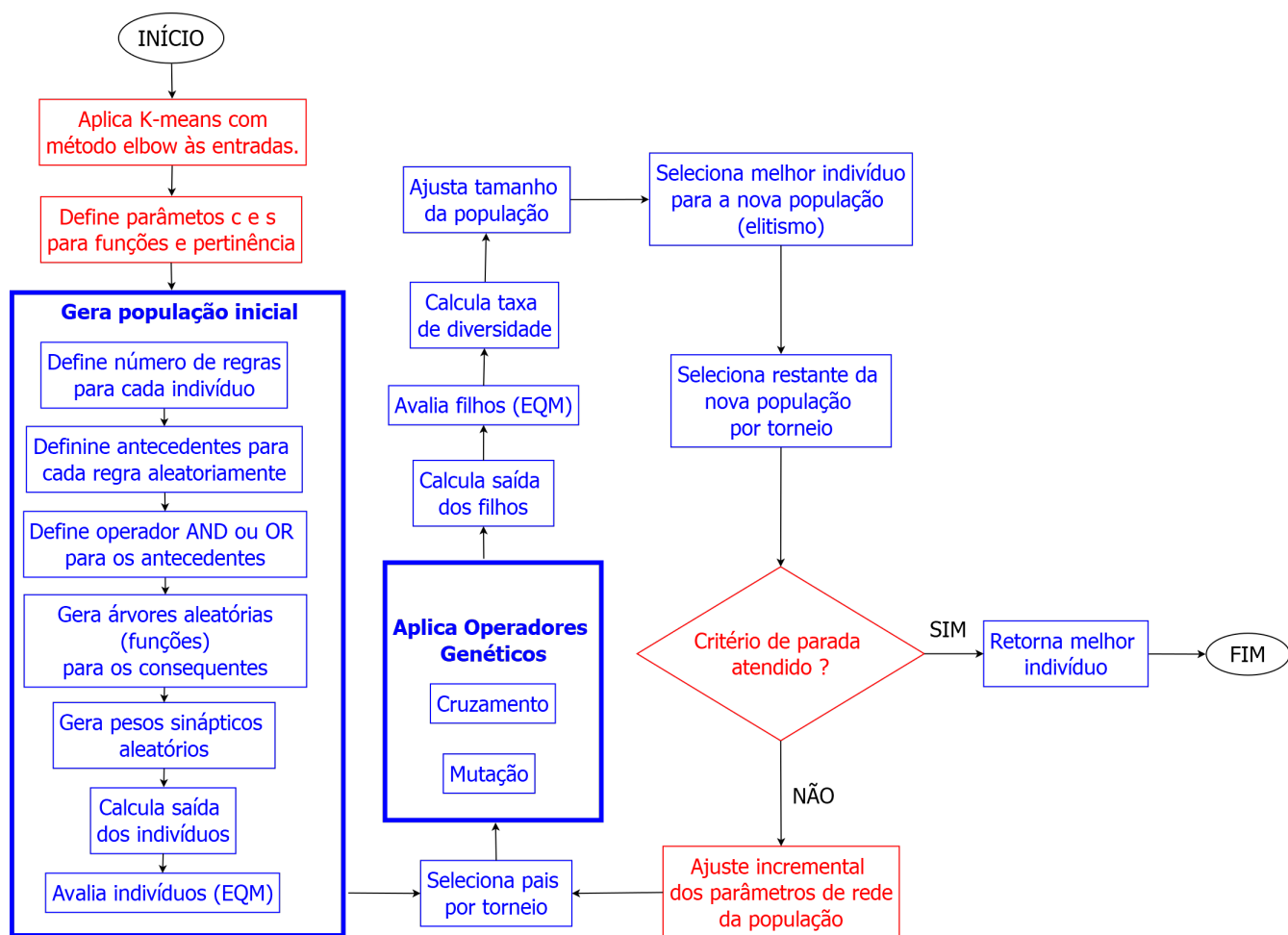


Figura 24 – Fluxograma do algoritmo de aprendizagem da rede DGP-NF.

Capítulo 6

Experimentos Computacionais

Este capítulo apresenta os experimentos computacionais realizados para avaliar os modelos propostos e compará-los com abordagens alternativas em problemas de previsão e identificação de sistemas. A Seção 6.1 descreve a metodologia adotada nos experimentos, incluindo as medidas de desempenho e os testes estatísticos utilizados. Na Seção 6.2, os resultados obtidos são apresentados e os modelos são comparados. A Seção 6.3 analisa a influência dos mecanismos de população dinâmica e controle de diversidade no desempenho do modelo DGP-NF. Por fim, a Seção 6.4 discute o impacto da estratégia de poda proposta neste trabalho.

6.1 Metodologia

Para avaliar as abordagens propostas, os modelos desenvolvidos (FALCON-GA, EvoNFuzz e DGP-NF) foram aplicados a 21 conjuntos de dados selecionados aleatoriamente nos repositórios *UCI Machine Learning*¹ e *Kaggle*², envolvendo tarefas de previsão e identificação de sistemas. Os conjuntos de dados utilizados apresentam diferentes tamanhos e domínios, permitindo avaliar a robustez e a capacidade de generalização dos modelos. As principais características de cada conjunto — número de amostras (k), número de atributos (n), intervalo de valores de entrada (X_{rg}), intervalo de valores de saída (Y_{rg}), desvio padrão das entradas (X_{sd}) e desvio padrão da saída (Y_{sd}) — são apresentadas na Tabela 1.

Os resultados obtidos pelos modelos propostos foram comparados entre si e com modelos alternativos, tais como NFN-MG-GP (BRAS; SILVA; WANNER, 2021), *Decision Tree* (DT), *Random Forest* (RF), *AdaBoost* (AB), *Gradient Boosting* (GB) e *Multi-Layer Perceptron* (MLP). Os hiperparâmetros dos modelos clássicos (DT, RF, AB, GB e MLP) foram ajustados por meio do método de *Grid Search*, que divide o espaço de busca em grades, realizando uma busca exaustiva sobre combinações pré-definidas e avaliando-as via validação cruzada

¹<https://archive.ics.uci.edu/>

²<https://www.kaggle.com/>

Tabela 1 – Conjuntos de dados utilizados. k = Número de amostras; n = número de atributos; X_{rg} = faixa dos valores de entrada; Y_{rg} = faixa dos valores de saída; X_{sd} = Desvio padrão da entrada; Y_{sd} = Desvio padrão da saída.

Dataset	k	n	X_{rg}	Y_{rg}	X_{sd}	Y_{sd}
Mackey-Glass (MC)	1100	4	(0.0, 1.3400)	(0.0, 1.3400)	0.2694	0.2881
Airfoil (AIR)	1502	5	(0.0, 20000.0)	(103.38, 140.987)	634.9512	6.9009
Yacht (YCT)	307	6	(-5.0, 5.35)	(0.01, 62.42)	0.4484	15.1736
Real Estate (RE)	413	6	(0.0, 6488.021)	(7.6, 117.5)	212.8847	13.6230
Carbon V (CV)	10720	3	(0.0451, 12.0)	(0.0389, 0.9611)	1.3695	0.2910
Carbon U (CU)	10720	3	(6.1e-05, 12.0)	(0.0, 1.0)	1.3702	0.2891
Carbon W (CW)	10720	3	(6.1e-05, 12.0)	(0.0, 1.0)	1.3702	0.2891
Auto-MPG (AMG)	391	7	(1.0, 5140.0)	(9.0, 46.6)	143.0794	7.8033
CCPP (CP)	9567	4	(1.81, 1033.3)	(420.26, 495.76)	10.1747	17.0658
CASP (CA)	45729	9	(0.0, 5472011.4075)	(15.228, 55.3009)	63526.8990	5.9797
Concrete (CT)	1029	8	(0.0, 1145.0)	(2.33, 82.6)	62.8739	16.6569
Abalone (ABL)	4176	8	(0.0, 3.0)	(1, 29)	0.2563	3.2236
Turbine (TUB)	35039	9	(-32.7274, 43799.0)	(0.4365, 29.1365)	310.3299	3.5728
Wine Red (WR)	1598	11	(0.0, 289.0)	(3, 8)	4.3937	0.8077
Wine White (WW)	4897	11	(0.0, 440.0)	(3, 9)	6.1057	0.8857
Energy Heating (EH)	767	8	(0.0, 808.5)	(6.01, 43.1)	22.6743	10.0938
Energy Cooling (EC)	767	8	(0.0, 808.5)	(10.9, 48.03)	22.6743	9.5188
QCM (QCM)	978	6	(-200.0, 2775.0)	(-200.0, 2.231)	221.3317	38.9787
Power Z1 (PZ1)	52415	5	(0.004, 1163.0)	(13895.6962, 52204.3951)	82.4659	7130.6267
Power Z2 (PZ2)	52415	5	(0.004, 1163.0)	(8560.0815, 37408.8608)	82.4659	5201.4712
Power Z3 (PZ3)	52415	5	(0.004, 1163.0)	(5935.1741, 47598.3264)	82.4659	6622.2199

(BISCHL et al., 2023). Os valores testados para cada hiperparâmetro estão apresentados na Tabela 2. Devido a diversidade de modelos com características distintas, a escolha dos parâmetros e seus respectivos intervalos baseou-se em estudos que destacaram sua relevância e indicaram faixas promissoras (AGRAWAL, 2021; PROBST; WRIGHT; BOULESTEIX, 2019; PROBST; BOULESTEIX; BISCHL, 2019; KHALID; JAVAID, 2020; MARKOVICS; MAYER, 2022; BENTÉJAC; CSÖRGŐ; MARTÍNEZ-MUÑOZ, 2021). Visto que quantidade de parâmetros e valores testados influenciam diretamente na performance dos modelos, esta escolha visa trazer maior equidade na comparação com as técnicas evolucionárias propostas.

Para os modelos NFN-MG-GP, FALCON-GA, EvoNFuzz e DGP-NF, foram realizados testes específicos para definir as taxas de cruzamento e mutação. Inicialmente, valores entre 0 e 1 foram avaliados em incrementos de 0,10 para ambos os parâmetros. Essa etapa indicou que a programação genética (GP) alcança melhores resultados com taxas de cruzamento entre 0,60 e 0,80 e taxas de mutação entre 0,20 e 0,40. Posteriormente, para cada conjunto de dados, foram testados valores de cruzamento entre 0,60 e 0,90 e de mutação entre 0,20 e 0,50, variando de 0,02 em 0,02, a fim de definir a configuração final dos modelos. Os demais parâmetros foram definidos ou ajustados conforme a Tabela 2.

Cada experimento foi executado 21 vezes, com as amostras divididas em 80% para treinamento e 20% para teste. Os dados foram normalizados durante a fase de treinamento e, posteriormente, desnormalizados para a medição do erro, permitindo uma avaliação mais precisa em valores reais. O desempenho dos modelos foi avaliado pelo RMSE médio calculado ao longo das 21 execuções. O RMSE é definido por:

Tabela 2 – Valores testados no *Grid Search*

Modelo	Hiperparâmetro	Valores
DGP-NF	<i>population_size</i> <i>generations</i> <i>min_rules_individual</i> <i>max_rules_individual</i> <i>selection_method</i> <i>functionsetsofGP</i> <i>constantsrangeofGP</i>	20 – 100 100 5 na população inicial 50 na população inicial <i>binary_tournament</i> {+, -, *, /} 1 – 9
FALCON-GA	<i>population_size</i> <i>generations</i> <i>gradient_epochs</i> <i>selection_method</i> <i>learning_rate</i>	50 100 100 <i>binary_tournament</i> [0.01, 0.1, 0.001]
NFN-MG-GP	<i>population_size</i> <i>generations</i> <i>gradient_epochs</i> <i>selection_method</i> <i>membership_function_number</i> <i>learning_rate</i>	50 100 5 por geração <i>binary_tournament</i> 3 – 10 por entrada [0.01, 0.1, 0.001]
Decison Tree	<i>max_depth</i> <i>min_samples_split</i>	[None, 5, 10, 15, 20] [2, 5, 10, 15, 20]
Random Forest	<i>n_estimators</i> <i>max_depth</i> <i>min_samples_split</i> <i>max_features</i>	[25, 50, 100, 150, 200] [None, 5, 10, 15, 20] [2, 5, 10] ["sqrt", "auto"]
AdaBoost	<i>n_estimators</i> <i>learning_rate</i>	[50, 100, 200, 250] [0.01, 0.1, 0.5, 1]
Gradient Boosting	<i>n_estimators</i> <i>learning_rate</i> <i>max_depth</i>	[25, 50, 70, 100, 120] [0.01, 0.05, 0.1, 0.2] [3, 5, 10]
MLP	<i>hidden_layer_sizes</i> <i>activation</i> <i>solver</i> <i>alpha</i> <i>learning_rate</i>	[(10, 10), (20, 20), (50, 50), (100, 100), (50), (100)] ['tanh', 'relu'] ['adam'] [0.0001, 0.001, 0.01, 0.1] ['constant', 'adaptive']

$$RMSE = \frac{1}{N} \left(\sum_{t=1}^N (y^t - \hat{y}^t) \right)^{\frac{1}{2}}, \quad (45)$$

em que N é o número de amostras de teste, $\hat{y}^t$ é a saída estimada pelo modelo e y^t é a saída desejada.

Para a comparação estatística entre os modelos, foi realizado um teste t pareado unilateral. Como o DGP-NF apresentou a melhor desempenho, todos os testes pareados foram realizados com base nesse modelo. O nível de significância adotado foi $\alpha = 0,05$. Contudo, devido à comparação inicial em formato um contra todos, foram testadas múltiplas hipóteses, o que requer uma correção do nível de significância para reduzir o risco de rejeitar falsamente hipóteses nulas (CAMPELO; WANNER, 2020). Uma das correções mais utilizadas é a correção de Bonferroni (SHAFFER, 1995; SEDGWICK, 2014), que foi aplicada neste trabalho. Essa correção consiste em dividir o nível de significância original pelo número de testes realizados — neste caso, apenas os testes pareados entre o DGP-NF e os demais modelos. Assim, o novo nível de significância para cada teste é dado por:

$$\alpha_a = \frac{\alpha_i}{TP}, \quad (46)$$

em que α_a representa o nível de significância ajustado após a correção, α_i é o nível de significância inicial (0,05), e TP é o número de testes pareados (comparações) realizados. Neste estudo, como o modelo DGP-NF foi comparado com outros oito modelos, temos $TP = 8$, resultando em um novo valor de α igual a 0,0062, conforme definido pela Eq. (46).

6.2 Resultados Experimentais

O RMSE dos modelos, bem como os valores da estatística do teste t e os p-valores são apresentados na Tabela 3. O menor valor de RMSE para cada conjunto de dados está destacado em azul. Quanto ao teste estatístico, os casos em que o DGP-NF apresentou superioridade estatística estão destacados em azul; os casos de inferioridade estatística do DGP-NF estão em vermelho. Os resultados em que não se pode afirmar superioridade ou inferioridade estatística do DGP-NF estão destacados em preto.

Os resultados indicam que o DGP-NF supera os demais modelos em termos de RMSE em 17 dos 21 conjuntos de dados - MC, AIR, RE, CV, CU, CW, AMG, CP, CA, CT, ABL, EH, EC, QCM, PZ1, PZ2 e PZ3. Em YCT e TUB apenas o *Gradient Boosting* obteve melhor RMSE do que DGP-NF e, para WR, apenas o *Random Forest*, sendo o DGP-NF o segundo modelo com menor RMSE para estes 3 conjuntos de dados. Apenas para a base WW o DGP-NF foi superado, considerando-se o RMSE, por 2 modelos, sendo eles *Random Forest* e *Gradient Boosting*.

A partir dos resultados dos testes estatísticos, pode-se observar que, dos 17 conjuntos de dados em que o DGP-NF obteve menor RMSE, ele se demonstra estatisticamente superior em 16 deles - MC, AIR, RE, CV, CU, CW, AMG, CP, CA, ABL, EH, EC, QCM, PZ, PZ2 e PZ3 -, sendo que para a base CT não se observou diferença estatística entre DGP-NF e *Gradient Boosting*, apesar do RMSE mais baixo nos testes conduzidos. Da mesma forma, apesar de obter RMSE mais alto que *Gradient Boosting* para o conjunto de dados YCT, o teste estatístico também não confirmou haver diferença entre eles. Sendo assim, o DGP-NF se mostrou estatisticamente comparável ao *Gradient Boosting* para os conjuntos de dados CT e YCT e superior aos demais modelos. Na base TUB, o teste mostra que há diferença estatística entre DGP-NF e *Gradient Boosting*, sendo o *Gradient Boosting* superior para esta base e, o DGP-NF, inferior ao *Gradient Boosting* e superior aos demais modelos. Já em WR o *Gradient Boosting*, embora tenha obtido maior RMSE que DGP-NF demonstra-se também estatisticamente comparável a ele, enquanto *Random Forest* mostra-se superior a todos os modelos testados para esta base. Por fim, no conjunto WW, DGP-NF se mostra

Tabela 3 – Comparação dos 8 preditores em 21 conjuntos de dados: RMSE médio das 21 execuções, e resultados do Teste-T pareado.

Dataset	Métrica	DGP-NF	EvoNFuzz	FALCON-GA	NFN-MG-GP	DT	RF	AB	GB	MLP
MC	RMSE	0.0231	0.0290	0.0454	0.0535	0.0530	0.0387	0.0444	0.0687	0.0929
	Teste-T	-	-10.6366	-24.5139	-38.4721	-6.092	-4.0647	-24.0123	-3.3792	-9.1156
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0004	0.0000	0.0000	0.0000
AIR	RMSE	1.1131	2.4758	2.2205	2.5445	2.5426	1.7745	3.5823	1.6430	4.6068
	Teste-T	-	-8.1786	-12.8329	-16.3371	-16.5787	-7.8419	-22.9335	-6.8045	-25.2293
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0006	0.0000
YCT	RMSE	0.9916	1.6950	2.3435	3.7518	1.4390	1.1135	1.6348	0.8273	4.5935
	Teste-T	-	-9.4320	-22.2117	-14.0604	-9.1338	-5.7455	-12.9443	-0.783	-22.3907
	p-valor	-	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000	0.4401	0.0000
RE	RMSE	4.3063	5.7517	5.5080	6.7770	8.1954	6.6783	7.0125	7.0086	7.6317
	Teste-T	-	-14.2639	-21.3225	-32.2875	-14.9612	-11.0982	-13.8053	-12.4702	-18.026
	p-valor	-	0.0000	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
CV	RMSE	0.0013	0.0034	0.0043	0.0033	0.0034	0.0031	0.0116	0.0030	0.0048
	Teste-T	-	-23.2609	-96.9352	-20.8509	-55.3697	-49.9195	-102.9129	-48.4085	-37.8728
	p-valor	-	0.0000	0.0003	0.0017	0.0015	0.0047	0.0054	0.0018	0.0011
CU	RMSE	0.0022	0.0030	0.0032	0.0032	0.0036	0.0031	0.0112	0.0031	0.0050
	Teste-T	-	-8.7595	-10.9305	-6.3334	-15.3345	-9.7063	-53.7867	-10.517	-24.1068
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
CW	RMSE	0.0023	0.0075	0.0072	0.0075	0.0177	0.0155	0.0357	0.0137	0.0139
	Teste-T	-	-29.5778	-28.1245	-25.85	-24.1586	-25.0629	-100.9026	-18.7832	-19.8799
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
AMG	RMSE	1.5840	2.4374	2.5380	3.1300	3.0240	2.4570	2.7270	2.5470	3.3840
	Teste-T	-	-9.3903	-18.3016	-15.5205	-20.9876	-13.4258	-18.2677	-15.8122	-22.4151
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
CP	RMSE	2.3580	3.1061	2.9430	3.2013	3.7260	3.0330	4.3200	2.9880	5.4270
	Teste-T	-	-3.4367	-5.7917	-5.7239	-15.8926	-6.7799	-23.0709	-6.3234	-14.0732
	p-valor	-	0.0018	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
CA	RMSE	1.5003	2.0505	1.9620	2.6333	1.8450	1.6200	2.0160	1.6920	2.5830
	Teste-T	-	-7.0220	-8.3581	-6.0036	-17.7806	-5.5075	-12.3299	-5.4431	-19.3729
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
CT	RMSE	3.9888	4.8922	4.7070	5.2100	6.2460	4.5450	6.8220	4.2207	6.4170
	Teste-T	-	-3.5559	-8.2007	-7.3472	-13.1437	-5.5095	-15.9792	-6.3817	-16.0321
	p-valor	-	0.0013	0.0048	0.0001	0.0000	0.0021	0.0000	0.9311	0.0000
ABL	RMSE	1.6020	2.1858	1.9890	2.0520	2.1870	1.9800	2.1510	2.0430	2.2860
	Teste-T	-	-20.0025	-48.8157	-19.6329	-40.5562	-21.5677	-30.3499	-20.8236	-18.5666
	p-valor	-	0.0000	0.0049	0.0024	0.0016	0.0055	0.0029	0.0013	0.0010
TUB	RMSE	0.9427	1.1221	1.3500	2.1300	1.1715	1.0282	1.4237	0.9091	3.1971
	Teste-T	-	-10.8220	-160.514	-77.9242	-39.5218	-3.1547	-132.2561	3.3264	-3.3865
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0037	0.0000	0.0023	0.0020
WR	RMSE	0.5271	0.7580	0.7920	0.7020	0.6541	0.5130	0.5753	0.5568	0.5859
	Teste-T	-	-15.5003	-75.5517	-25.7228	-14.3908	7.2968	-3.5254	0.6707	-5.7600
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0014	0.5081	0.0000
WW	RMSE	0.5967	0.7905	0.7650	0.7920	0.6968	0.5602	0.6673	0.5849	0.6755
	Teste-T	-	-14.1292	-52.906	-28.2845	-22.4291	8.8644	-17.7384	2.4185	-22.8318
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0021	0.0000
EH	RMSE	0.2889	0.4701	0.5580	0.6480	0.5802	0.4605	1.7304	0.3567	1.7127
	Teste-T	-	-10.0362	-31.0321	-40.5123	-15.9548	-11.3327	-67.6597	-2.3323	-16.3582
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0016	0.0000
EC	RMSE	0.8388	1.1671	1.2330	1.5120	1.7788	1.5719	2.1495	0.9390	2.4633
	Teste-T	-	-10.8517	-21.4051	-28.3737	-31.9626	-23.2564	-43.5371	-1.901	-42.2327
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0057	0.0000
QCM	RMSE	4.8447	5.0420	5.3190	6.0700	5.5827	4.4433	7.0965	4.3291	5.9720
	Teste-T	-	-23.2609	-96.9352	-20.8509	-55.3697	-49.9195	-102.9129	-48.4084	-37.8728
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
PZ1	RMSE	11.6352	13.3238	15.8000	15.4050	11.8956	12.1823	15.1557	12.1053	15.3355
	Teste-T	-	-22.6496	-123.6267	-7.8853	-7.0058	-21.2378	-51.0555	-17.8023	-26.8216
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
PZ2	RMSE	13.0186	15.2517	15.9100	16.0748	14.5827	13.4433	16.0965	13.3291	14.9720
	Teste-T	-	-21.4400	-96.9531	-25.1071	-66.4285	-20.9643	-146.7137	-15.0961	-99.1192
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
PZ3	RMSE	12.9685	14.2304	15.5600	16.3917	13.5512	13.3607	13.7213	13.3946	13.9330
	Teste-T	-	-13.4164	-52.1922	-65.3738	-29.8757	-21.2336	-39.2529	-22.6645	-48.0952
	p-valor	-	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

estatisticamente inferior a *Random Forest* e *Gradient Boosting*, mas superior aos outros 6 modelos.

Em resumo, o DGP-NF supera todos os outros modelos testados em 16 conjuntos de dados; apresenta desempenho competitivo ao *Gradient Boosting* e melhor que os demais em 2 conjuntos; e é estatisticamente inferior a 1 ou 2 modelos em 3 conjuntos de dados.

Ao analisar os resultados com base nas características dos conjuntos de dados, é possível notar que o desempenho do DGP-NF tende a diminuir à medida que se aumenta o número de variáveis de entrada. Isso sugere que o modelo proposto é mais adequado para problemas com um número menor de variáveis de entrada, preferencialmente abaixo de 10. Isso pode ser explicado pelo fato de que o espaço de busca cresce exponencialmente com o aumento do número de variáveis de entrada, visto que cada uma ainda é dividida em m conjuntos *fuzzy*. Além disso, o processo de agrupamento realizado na fase não-supervisionada para definir os conjuntos *fuzzy* pode guiar o aprendizado evolucionário da PG para um subespaço de busca específico já identificado pelo previamente por ele. Este fato, combinado com o aumento exponencial do espaço de busca, pode dificultar o processo evolucionário. Este mesmo comportamento é observado no EvoNFuzz.

Já em relação aos modelos propostos neste trabalho, observa-se que o DGP-NF é sempre estatisticamente superior ao EvoNFuzz e ao FALCON-GA, mesmo quando apresenta desempenho inferior ou competitivo em comparação com outros modelos da literatura. Isso indica que a abordagem baseada em aprendizagem incremental, aliada aos mecanismos de maior exploração do espaço de busca, como população dinâmica e controle de diversidade, proporciona ganhos significativos para modelos neuro-*fuzzy*.

Uma vez que o método do gradiente requer mais de uma época de treinamento para atingir resultados relevantes, os resultados sugerem que essa técnica pode, de certa forma, enviesar a evolução da programação genética, conduzindo a população prematuramente a ótimos locais. Nota-se também que o FALCON-GA apresentou o pior desempenho, o que sugere que modelos funcionais baseados em PG são mais eficientes para problemas de regressão do que modelos linguísticos baseados em algoritmos genéticos. Ainda assim, é importante destacar que experimentos anteriores mostraram que o FALCON-GA pode superar modelos mais simples da literatura, especialmente em bases de dados menos complexas, além de sempre superar o modelo FALCON original (BRÁS; SILVA; WANNER, 2024).

6.3 Ablação - População Dinâmica e Controle de Diversidade

Esta seção apresenta uma análise do impacto dos mecanismos de população dinâmica e controle de diversidade no desempenho do DGP-NF. Inicialmente, o algoritmo completo foi

comparado com uma versão alternativa que utiliza uma população fixa de 50 indivíduos. Nessa versão, os operadores de dispersão são mantidos, porém o tamanho da população permanece constante ao longo das gerações.

Em um segundo experimento, comparou-se o algoritmo completo com uma versão que desativa os operadores de controle de diversidade. Ou seja, nessa configuração, o tamanho da população é ajustado de acordo com os critérios definidos, mas não há nenhum mecanismo que assegure que os novos indivíduos gerados contribuam efetivamente para aumentar a diversidade populacional.

Para ambos os testes, as versões alternativas do algoritmo foram comparadas com a versão completa em termos de RMSE, visando analisar a acurácia, e também foram submetidas a um teste t pareado para verificar se as diferenças observadas representam ganhos estatisticamente significativos. A Tabela 4 apresenta os valores de RMSE, número médio de regras, desvio padrão e os resultados do teste t (estatística t e p-valor) obtidos na comparação com o algoritmo completo, considerando um nível de significância de 0,05. Casos de superioridade estatística do DGP-NF estão destacados em azul, enquanto aqueles sem diferença estatística significativa estão indicados em preto.

Observa-se que o algoritmo completo apresentou superioridade estatística em 18 dos 21 conjuntos de dados em relação à versão sem população dinâmica, e em 16 dos 21 conjuntos de dados quando comparado à versão sem controle de diversidade. Esses resultados indicam que os mecanismos de população dinâmica e de controle de diversidade contribuem para o aprimoramento do desempenho do algoritmo, sendo o ajuste dinâmico da população mais impactante que o controle de diversidade. Isso sugere que o tamanho da população exerce um papel mais relevante no processo evolutivo do que a diversidade, ao menos nos contextos considerados neste trabalho.

Além disso, os valores de RMSE reforçam essa conclusão: a versão sem população dinâmica apresenta erros consideravelmente maiores na maioria dos casos. Importante destacar que, em nenhuma situação, o algoritmo completo apresentou inferioridade estatística frente às versões alternativas, o que evidencia que os mecanismos de população dinâmica e controle de diversidade não prejudicam o processo evolutivo, mesmo quando não resultam em ganhos estatisticamente significativos.

As figuras 25 e 26 mostram a diferença percentual entre as versões e o algoritmo completo em termos de acurácia. Os resultados confirmam que a implementação da população dinâmica no algoritmo proporciona ganhos mais significativos do que o uso exclusivo de operadores para controle de diversidade, quando o número de indivíduos permanece fixo. Estudos anteriores, como os apresentados em [Burke, Gustafson e Kendall \(2002\)](#), [Burlacu, Yang e Affenzeller \(2024\)](#), já discutem que a diversidade, por si só, não é um fator decisivo

Tabela 4 – Comparação entre o DGP-NF, versão sem população dinâmica (sem pop. din.) e versão sem diversidade (sem div.).

Dataset	Modelo	RMSE	n. regras	Desv. Pad	Teste-t	p-valor
MC	DGP-NF	0.0091	14	0.0045	-	-
	sem pop. din.	0.0132	15	0.0042	-10.6366	0.0069
	sem div.	0.0110	15	0.0039	-8.0754	0.0369
AIR	DGP-NF	1.1680	9	0.5450	-	-
	sem pop. din.	1.4532	10	0.7424	-8.1786	0.0000
	sem div.	1.1747	9	0.5172	-3.7532	0.0072
YCT	DGP-NF	0.9224	15	0.3696	-	-
	sem pop. din.	1.1436	13	0.5435	-9.4320	0.0000
	sem div.	0.9244	11	0.5226	-4.1445	0.0288
RE	DGP-NF	4.7848	17	0.3696	-	-
	sem pop. din.	5.8080	15	0.3075	-14.2639	0.0005
	sem div.	4.8948	19	0.3219	-5.3859	0.0324
CV	DGP-NF	0.0013	18	0.0002	-	-
	sem pop. din.	0.0016	16	0.0005	-3.2609	0.012
	sem div.	0.0013	17	0.0001	-0.7407	0.0441
CU	DGP-NF	0.0023	24	0.0005	-	-
	sem pop. din.	0.0027	23	0.0002	-8.7595	0.023
	sem div.	0.0025	20	0.0005	-3.6492	0.0070
CW	DGP-NF	0.0023	22	0.0010	-	-
	sem pop. din.	0.0278	21	0.0006	-29.5778	0.0025
	sem div.	0.0220	24	0.0011	-23.2336	0.0104
AMG	DGP-NF	1.7600	20	0.2488	-	-
	sem pop. din.	2.3760	18	0.1672	-9.3903	0.09
	sem div.	1.9600	17	0.4511	-1.6451	0.0608
CP	DGP-NF	2.6200	33	0.5044	-	-
	sem pop. din.	3.2520	31	0.3166	-2.3523	0.0005
	sem div.	2.8200	27	0.4451	-0.7344	0.0063
CA	DGP-NF	1.6670	22	0.1207	-	-
	sem pop. din.	1.9200	21	0.2941	-7.0220	0.0021
	sem div.	1.7700	22	0.2230	-2.1091	0.0910
CT	DGP-NF	4.4320	21	0.1904	-	-
	sem pop. din.	5.6160	17	0.2096	-3.5559	0.0670
	sem div.	4.7300	18	0.2111	-2.2401	0.0092
ABL	DGP-NF	1.7800	18	0.0471	-	-
	sem pop. din.	2.2920	17	0.0328	-48.8157	0.0039
	sem div.	1.9700	18	0.0492	-21.16	0.0116
TUB	DGP-NF	0.8570	27	0.0146	-	-
	sem pop. din.	1.1040	25	0.0223	-1.514	0.11
	sem div.	0.9100	22	0.0262	-7.9242	0.0165
WR	DGP-NF	0.5700	11	0.0187	-	-
	sem pop. din.	0.7440	10	0.0328	-5.5517	0.041
	sem div.	0.6500	11	0.0211	-2.7228	0.0945
WW	DGP-NF	0.6030	12	0.0194	-	-
	sem pop. din.	0.8040	9	0.0255	-12.906	0.032
	sem div.	0.4702	10	0.0214	-8.2845	0.0549
EH	DGP-NF	0.3210	18	0.0451	-	-
	sem pop. din.	0.4560	15	0.0392	-11.0321	0.0000
	sem div.	0.3400	17	0.0498	-8.5123	0.0063
EC	DGP-NF	0.9320	15	0.1090	-	-
	sem pop. din.	1.1760	12	0.4192	-5.0069	0.0000
	sem div.	0.9900	11	0.3659	-2.0369	0.0068
QCM	DGP-NF	0.0166	11	0.0025	-	-
	sem pop. din.	0.0233	10	0.0029	-6.6930	0.001
	sem div.	0.0188	12	0.0036	-2.3167	0.0070
PZ1	DGP-NF	11.6352	23	0.3144	-	-
	sem pop. din.	15.0372	21	0.4826	-10.2362	0.046
	sem div.	12.0350	20	0.4621	-5.3047	0.0174
PZ2	DGP-NF	13.0186	36	0.7800	-	-
	sem pop. din.	16.5520	32	0.9011	8.2810	0.38
	sem div.	13.4200	32	0.8339	-2.5520	0.0335
PZ3	DGP-NF	12.9685	42	0.2190	-	-
	sem pop. din.	15.0970	44	0.4692	-7.9371	0.0000
	sem div.	13.5800	45	0.3169	3.5922	0.0684

para o desempenho da programação genética. Por outro lado, diversas pesquisas, como [Farinati, Bakurov e Vanneschi \(2023\)](#), [Eiben, Marchiori e Valko \(2004\)](#), evidenciam que o tamanho da população exerce um papel crucial nesse desempenho. Os resultados obtidos neste trabalho sugerem que a combinação dos dois mecanismos, população dinâmica e controle de diversidade, pode trazer benefícios ainda mais expressivos para a PG.

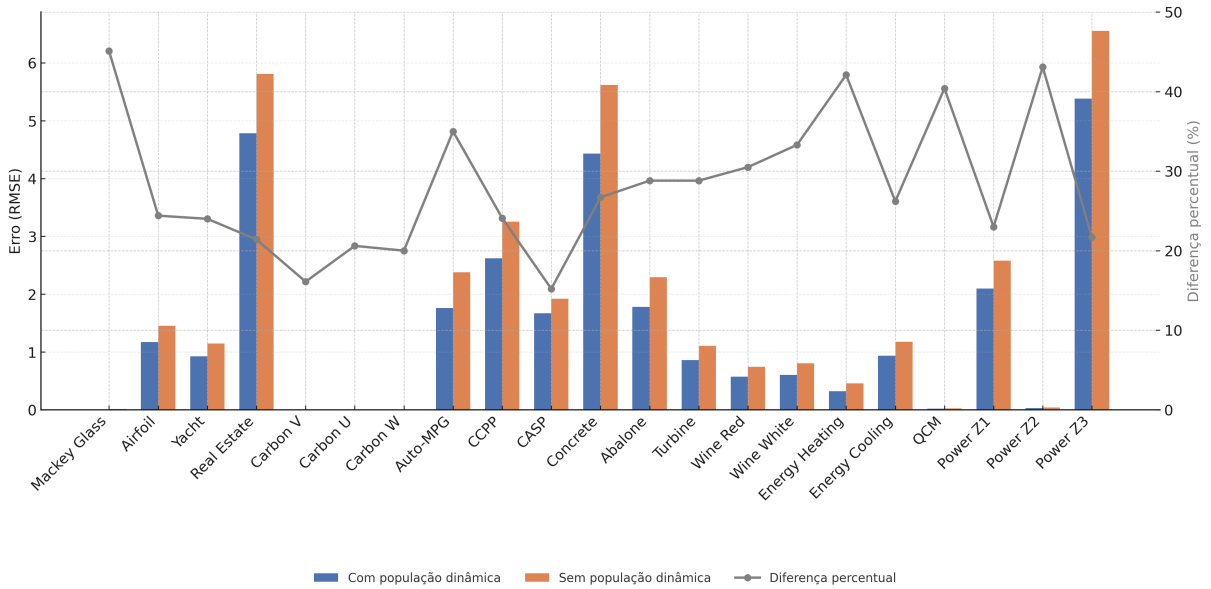


Figura 25 – Impacto da população dinâmica.

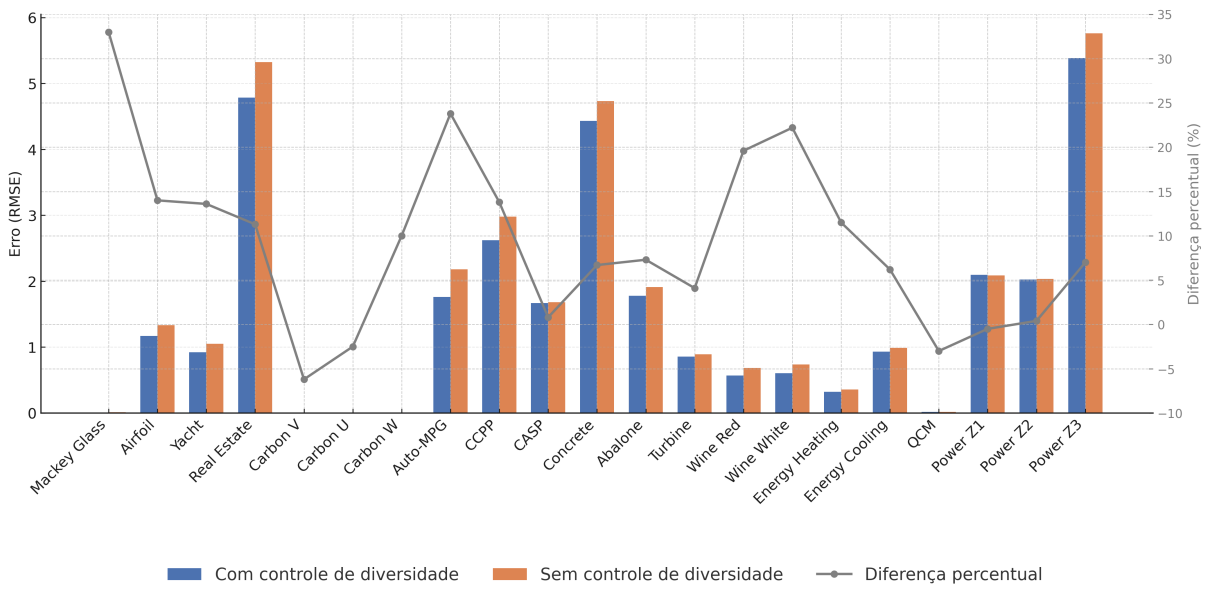


Figura 26 – Impacto do controle de diversidade.

As figuras [27](#) e [28](#) ilustram o ganho percentual das versões com população fixa, com e sem controle de diversidade, para diferentes tamanhos populacionais. O objetivo deste

teste é verificar se os ganhos observados decorrem efetivamente da variação adaptativa do tamanho da população durante o processo evolutivo (população dinâmica), ou se a simples utilização de populações maiores já seria suficiente para alcançar resultados semelhantes.

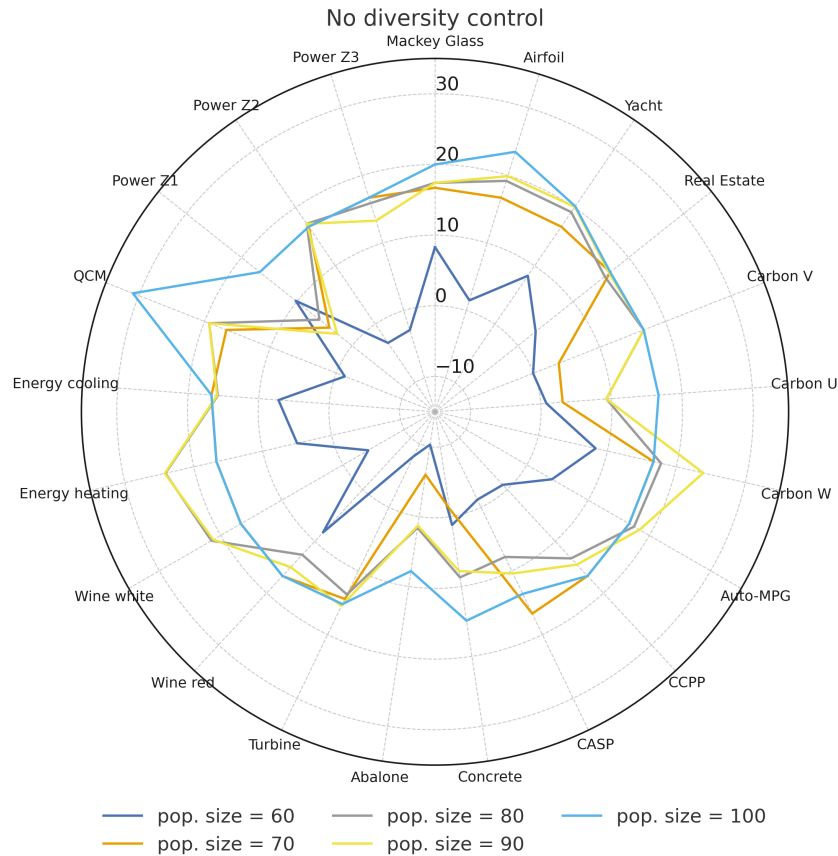


Figura 27 – Diferença percentual em relação ao tamanho da população = 50 sem controle de diversidade.

Pode-se observar que o algoritmo com controle de diversidade consegue obter melhor desempenho em populações menores do que o algoritmo sem esses operadores. Por outro lado, em populações maiores a diferença de desempenho começa a diminuir. Isso sugere que, embora o controle da diversidade dentro de uma população fixa seja importante, a introdução de novos indivíduos pode facilitar ainda mais a evolução das soluções e a exploração do espaço de busca. Assim, esses resultados sugerem que a combinação dos dois mecanismos é o melhor cenário para aprimorar o aprendizado da rede. O uso de população dinâmica e controle da diversidade não se mostrou relevante em termos do número de regras na solução final.

Analisando este resultado em conjunto com o apresentado anteriormente na Tabela 4, fica claro que a combinação das duas estratégias pode ser uma maneira promissora de aumentar o desempenho de um algoritmo baseado em GP, especialmente para redes *neuro-fuzzy*. Isso pode ocorrer porque, embora o tamanho da população seja um fator relevante, a estrutura do modelo *neuro-fuzzy*, por já estar inicialmente ajustada aos dados,

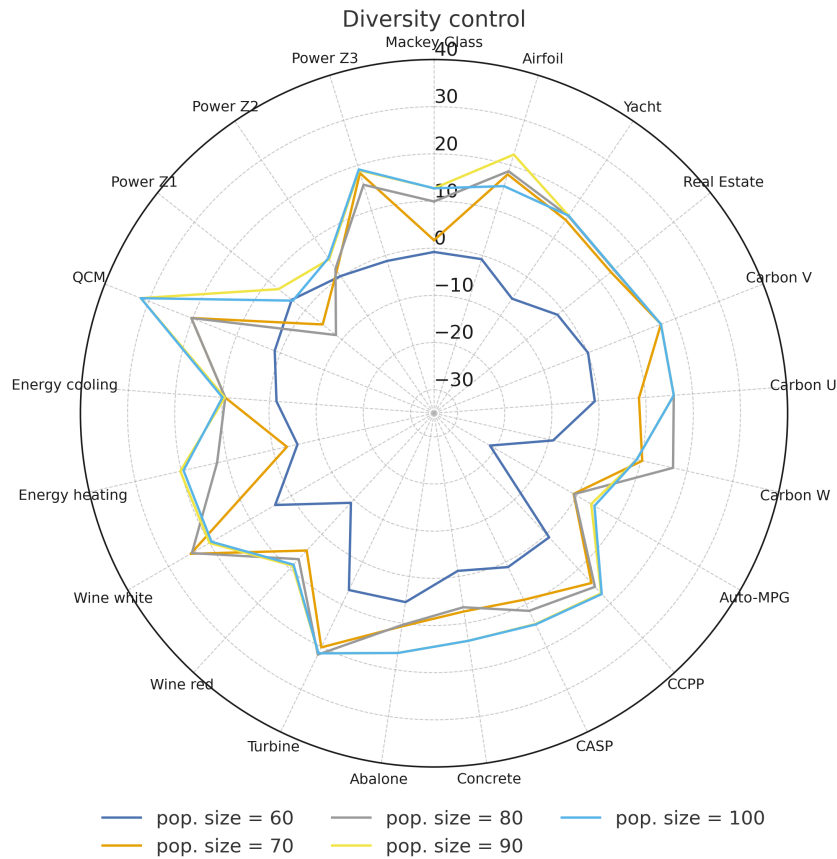


Figura 28 – Diferença percentual em relação ao tamanho da população = 50 com controle de diversidade.

pode guiar o algoritmo evolutivo para áreas específicas do espaço de busca, conforme discutido em diversos trabalhos e apresentado em Moghaddam (2024). Isso também pode ser demonstrado pela análise da convergência do algoritmo. A Figura 29 mostra que o algoritmo sem controle de diversidade populacional tende a sofrer convergência prematura, enquanto o algoritmo com controle de diversidade, mas com população dinâmica, consegue evoluir por mais gerações. Por fim, corroborando observações anteriores, o algoritmo completo demonstra capacidade de evoluir até as últimas gerações.

6.4 Ablação - Estratégia de Poda

Nesta seção, é avaliada a utilização da análise de diversidade como estratégia de poda. Como discutido na Seção 5.4, o objetivo do operador de poda é eliminar regras redundantes a fim de reduzir o tamanho e a complexidade da base de regras, sem comprometer significativamente o desempenho do algoritmo. Dessa forma, a execução do modelo *fuzzy* torna-se mais ágil e eficiente.

Embora existam diversas abordagens para a poda de regras em sistemas *fuzzy* (GU et al., 2023; HIKMAWATI; MAULIDEVI; SURENDRO, 2022; MADBOULY; REHEEM; GUIRGUIS,

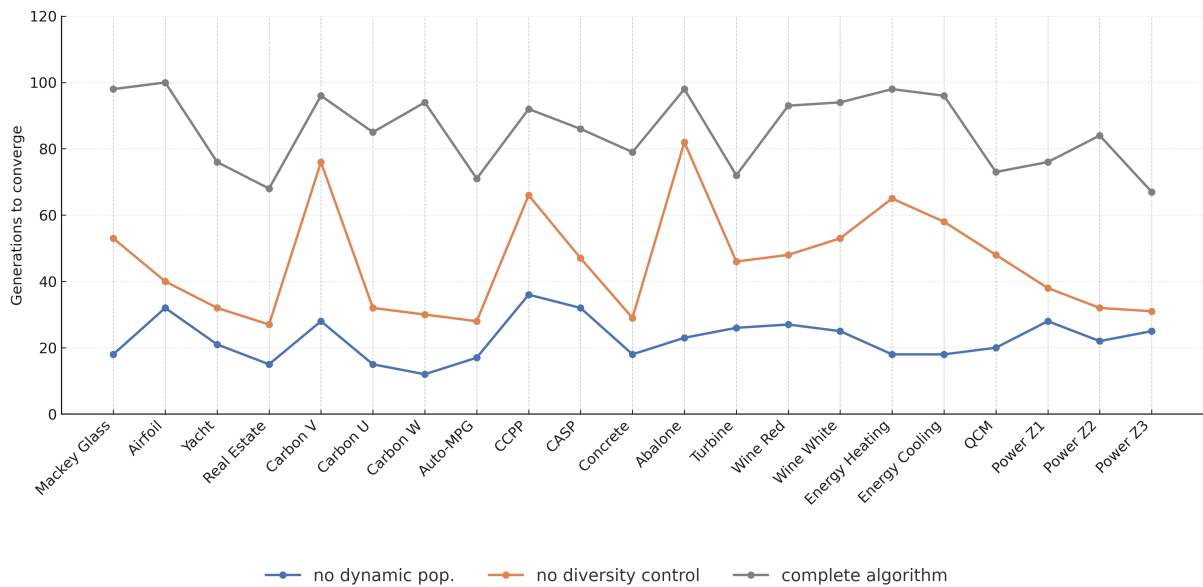


Figura 29 – Análise de convergência.

2021), neste trabalho propõe-se o uso da mesma medida de diversidade populacional para avaliar a similaridade entre regras. Essa proposta se justifica pelo fato de que um indivíduo em programação genética multi-gene pode ser visto como uma mini população, pois é composto por diversas árvores. Assim, torna-se desnecessária a incorporação de cálculos adicionais para identificar regras redundantes, contribuindo para a simplicidade e eficiência do sistema *fuzzy* evolucionário.

A versão completa do algoritmo foi comparada com uma versão alternativa que não utiliza o operador de poda, considerando o RMSE e o teste t pareado, com nível de significância de 0,05. A Tabela 5 mostra que, apesar de pequenas variações no erro médio, não há diferenças estatísticas significativas entre as duas versões, indicando que ambas podem ser utilizadas sem perda de desempenho.

A Figura 30 apresenta a diferença percentual no tamanho da base de regras gerada pelo algoritmo completo em comparação à versão sem o operador de poda. Observa-se que o operador de poda é capaz de reduzir entre 10% e 25% a quantidade de regras no modelo final. Esses resultados sugerem que a análise de diversidade populacional fenotípica pode ser uma abordagem eficaz também para a identificação de similaridade entre os genes superiores de um indivíduo multi-gene e, conseqüentemente, entre regras *fuzzy* na abordagem proposta neste trabalho.

Tabela 5 – Comparação entre DGP-NF e a versão sem poda.

Dataset	Modelo	RMSE	n. regras	Desv. Pad	Teste-t	p-valor
MC	DGP-NF	0.0091	14	0.0045	-	-
	sem poda	0.0110	17	0.0042	-0.6366	0.410
AIR	DGP-NF	1.1680	9	0.5450	-	-
	sem poda	1.1747	12	0.7424	-1.7856	0.080
YCT	DGP-NF	0.9224	15	0.3696	-	-
	sem poda	0.9144	16	0.5435	4.8320	0.320
RE	DGP-NF	4.7848	17	0.3696	-	-
	sem poda	4.5948	23	0.3075	1.3926	0.360
CV	DGP-NF	0.0013	18	0.0002	-	-
	sem poda	0.0013	20	0.0005	2.3269	0.120
CU	DGP-NF	0.0023	24	0.0005	-	-
	sem poda	0.0025	30	0.0002	-2.8892	0.058
CW	DGP-NF	0.0023	22	0.0010	-	-
	sem poda	0.0022	28	0.0006	2.5112	0.087
AMG	DGP-NF	1.7600	20	0.2488	-	-
	sem poda	1.8872	26	0.1672	-3.4330	0.090
CP	DGP-NF	2.6200	31	0.5044	-	-
	sem poda	2.8200	37	0.3166	-1.5723	0.042
CA	DGP-NF	1.6670	22	0.1207	-	-
	sem poda	1.7700	28	0.2941	-4.2307	0.230
CT	DGP-NF	4.4320	21	0.1904	-	-
	sem poda	5.1480	25	0.2096	-0.5559	0.061
ABL	DGP-NF	1.7800	18	0.0471	-	-
	sem poda	1.9700	21	0.0328	-2.8157	0.077
TUB	DGP-NF	0.8570	27	0.0146	-	-
	sem poda	0.8527	31	0.0223	0.5144	0.110
WR	DGP-NF	0.5700	11	0.0187	-	-
	sem poda	0.6500	12	0.0328	-2.3587	0.630
WW	DGP-NF	0.6030	12	0.0194	-	-
	sem poda	0.6300	15	0.0255	-1.9756	0.410
EH	DGP-NF	0.3210	18	0.0451	-	-
	sem poda	0.3400	23	0.0392	-1.2156	0.047
EC	DGP-NF	0.9320	15	0.1090	-	-
	sem poda	0.9900	17	0.4192	-1.6392	0.051
QCM	DGP-NF	0.0166	11	0.0025	-	-
	sem poda	0.0188	15	0.0029	-1.6930	0.900
PZ1	DGP-NF	11.6352	23	0.3144	-	-
	sem poda	12.1700	27	0.4826	-1.2362	0.130
PZ2	DGP-NF	13.0188	36	0.7800	-	-
	sem poda	14.0350	43	0.9011	-2.8510	0.250
PZ3	DGP-NF	12.9685	42	0.2190	-	-
	sem poda	10.3200	50	0.4692	3.9371	0.570

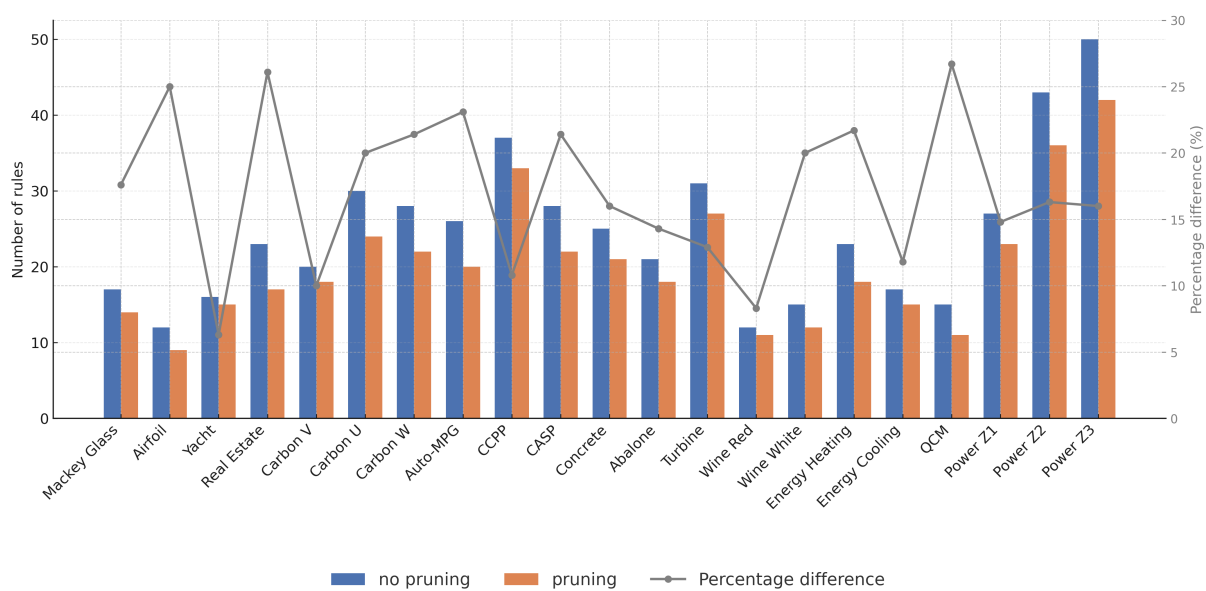


Figura 30 – Tamanho da base de regras com e sem poda.

Capítulo 7

Considerações Finais

Este capítulo apresenta as considerações finais do trabalho. A Seção 7.1 discute as conclusões a partir dos resultados alcançados e a Seção 7.2 apresenta as perspectivas de continuidade da pesquisa.

7.1 Conclusão

Este trabalho apresentou novas abordagens evolucionárias para a modelagem autônoma de redes neuro-*fuzzy* preditivas. Foram introduzidos três novos modelos construídos a partir dessas abordagens: FALCON-GA, EvoNFuzz e DGP-NF. O FALCON-GA utiliza um algoritmo genético (AG) para extração de regras linguísticas e um método baseado no gradiente para ajuste dos parâmetros da rede. Já os modelos EvoNFuzz e DGP-NF são sistemas funcionais que empregam regras polinomiais, modeladas por meio de uma abordagem baseada em programação genética multi-gene (PGMG), o que amplia as possibilidades dos modelos funcionais tradicionais. O EvoNFuzz adota um método de *backpropagation* baseado no gradiente para o ajuste dos parâmetros, enquanto o DGP-NF implementa um algoritmo de aprendizado incremental baseado em mínimos quadrados ponderados, além de incorporar mecanismos de exploração do espaço de busca, como população dinâmica e controle de diversidade. Tanto EvoNFuzz quanto DGP-NF também introduzem pesos sinápticos nos consequentes das regras, ponderando a saída da rede de acordo com a importância relativa de cada regra. Isso permite que diferentes regras tenham graus variados de influência no comportamento do modelo.

Experimentos computacionais realizados em 21 conjuntos de dados demonstraram a eficácia das abordagens propostas para tarefas de previsão e identificação de sistemas. O DGP-NF obteve o melhor desempenho, superando modelos alternativos em 16 dos 21 conjuntos de dados e apresentando resultados competitivos nos demais. Os resultados indicam que o uso de AG é eficiente para a extração de regras linguísticas, enquanto a PGMG se mostra uma técnica ideal para a modelagem de regras funcionais.

Adicionalmente, verificou-se que modelos funcionais apresentam melhor desempenho em problemas de regressão não linear, sendo as regras polinomiais particularmente eficazes para capturar relações complexas entre variáveis. A PGMG demonstrou ser uma abordagem adequada para a modelagem de bases de regras *fuzzy*, graças à sua estrutura baseada em múltiplas árvores, permitindo representar uma base de regras completa em um único indivíduo.

Os resultados também indicam a superioridade do algoritmo de aprendizado incremental em relação ao método de *backpropagation* baseado em gradiente. O treinamento baseado em gradiente exige múltiplas épocas, o que pode prejudicar o desempenho da programação genética e levar à convergência prematura. A estrutura das redes Neuro-*fuzzy* propostas também contribui para guiar o processo evolucionário, já que parte do treinamento é realizado a priori, a partir do particionamento do universo de discurso com base nos dados.

Este estudo mostrou ainda que a introdução de mecanismos de exploração aprimorada do espaço de busca, como população dinâmica e controle de diversidade, pode gerar ganhos significativos no aprendizado de redes neuro-*fuzzy* evolucionárias e mitigar problemas de convergência prematura. O ajuste no tamanho da população demonstrou ter impacto mais relevante que o controle de diversidade em populações fixas. No entanto, a combinação de ambos os mecanismos produziu os melhores resultados.

Adicionalmente, concluiu-se que, no contexto de redes preditivas, em que se busca priorizar a qualidade das soluções geradas pela PG e a acurácia do modelo final, medidas de diversidade fenotípica são suficientemente eficazes. Essa forma de diversidade também se mostrou útil para a identificação de regras redundantes em indivíduos multi-gene, que, por conterem múltiplas árvores, podem ser vistos como uma mini-população, onde cada gene representa uma regra com saída própria.

Sendo assim, esta pesquisa não apenas contribui significativamente para o avanço da modelagem de redes neuro-*fuzzy* preditivas e para o desenvolvimento de técnicas evolucionárias aplicadas a esse contexto, como também abre caminhos para novas investigações em inteligência computacional, inteligência artificial, aprendizado de máquina e computação evolucionária. Embora este trabalho tenha apresentado novos modelos híbridos, as técnicas e modelos desenvolvidos podem ser utilizadas de forma isolada em outros contextos.

Apesar dos resultados promissores, é importante destacar que os modelos propostos podem não ser adequados para problemas com um número muito elevado de variáveis de entrada.

7.2 Trabalhos Futuros

Com base nos experimentos realizados e conclusões obtidas, alguns trabalhos futuros propostos para esta pesquisa são:

- Utilização de tipos alternativos de operadores *fuzzy* para composição de regras, encapsulando outros tipos de T-normas e S-normas aos indivíduos genéticos;
- Evolução dos antecedentes e operadores *fuzzy* também pela PG;
- Implementação de técnicas para melhoria da interpretabilidade das regras geradas pela PG;
- Implementação de outros operadores para controle de diversidade;
- Avaliação e outras técnicas de exploração do espaço de busca;
- Avaliação e outros critérios e métricas para ajuste no tamanho da população;
- Avaliação dos modelos propostos para problemas de classificação.

Referências

- ABRAHAM, A. Neuro fuzzy systems: State-of-the-art modeling techniques. In: MIRA, J.; PRIETO, A. (Ed.). **Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 269–276. ISBN 978-3-540-45720-6. Citado 2 vezes nas páginas 13 e 14.
- ABRO, I. A. et al. Advanced biosignal-rgb fusion with adaptive neurofuzzy classification for high-precision action recognition. **IEEE Access**, IEEE, 2025. Citado 2 vezes nas páginas 1 e 2.
- AGRAWAL, T. **Hyperparameter optimization in machine learning: make your machine learning and deep learning models more efficient**. [S.l.]: Springer, 2021. Citado na página 66.
- AHMADIANFAR, I. et al. An improved adaptive neuro fuzzy inference system model using conjoined metaheuristic algorithms for electrical conductivity prediction. **Scientific Reports**, Nature Publishing Group UK London, v. 12, n. 1, p. 4934, 2022. Citado na página 2.
- ANDREWS, R.; DIEDERICH, J.; TICKLE, A. B. Survey and critique of techniques for extracting rules from trained artificial neural networks. **Knowledge-based systems**, Elsevier, v. 8, n. 6, p. 373–389, 1995. Citado na página 17.
- ASHISH, K. et al. Genetic-neuro-fuzzy system for grading depression. **Applied computing and informatics**, Elsevier, v. 14, n. 1, p. 98–105, 2018. Citado na página 21.
- ASHOKKUMAR, S.; MOHANBABU, G. Extreme learning adaptive neuro-fuzzy inference system model for classifying the epilepsy using q-tuned wavelet transform. **Journal of Intelligent & Fuzzy Systems**, IOS Press, v. 39, n. 1, p. 233–248, 2020. Citado 3 vezes nas páginas 2, 17 e 19.
- BABU B. V.; GUJARATHI, A. M. **Evolutionary computation : techniques and applications**. [S.l.]: Apple Academic Press Inc, 2017. ISBN 1771883367,978-1-77188-336-8,978-1-315-36638-8,978-1-77188-337-5. Citado 2 vezes nas páginas 2 e 7.
- BANERJEE, M.; MITRA, S.; PAL, S. K. Rough fuzzy mlp: Knowledge encoding and classification. **IEEE Transactions on neural networks**, IEEE, v. 9, n. 6, p. 1203–1216, 1998. Citado na página 17.
- BENTÉJAC, C.; CSÖRGŐ, A.; MARTÍNEZ-MUÑOZ, G. A comparative analysis of gradient boosting algorithms. **Artificial Intelligence Review**, Springer, v. 54, p. 1937–1967, 2021. Citado na página 66.
- BEYER, H.-G.; SCHWEFEL, H.-P. Evolution strategies—a comprehensive introduction. **Natural computing**, Springer, v. 1, n. 1, p. 3–52, 2002. Citado na página 3.
- BISCHL, B. et al. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. **Wiley Int. Reviews: Data Mining and Knowledge Discovery**, 2023. Citado na página 66.

BRAS, G.; SILVA, A. M.; WANNER, E. F. Multi-gene genetic programming to building up fuzzy rule-base in neo-fuzzy-neuron networks. **Journal of Intelligent & Fuzzy Systems**, IOS Press, v. 41, n. 1, p. 499–516, 2021. Citado na página [65](#).

BRÁS, G.; SILVA, A. M.; WANNER, E. F. A genetic algorithm for rule extraction in fuzzy adaptive learning control networks. **Genetic Programming and Evolvable Machines**, Springer, v. 25, n. 1, p. 11, 2024. Citado na página [70](#).

BRINDHA, S. et al. A robust and adaptive fuzzy logic based differential evolution algorithm using population diversity tuning for multi-objective optimization. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 102, p. 104240, 2021. Citado na página [22](#).

BURKE, E.; GUSTAFSON, S.; KENDALL, G. A survey and analysis of diversity measures in genetic programming. In: **Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation**. [S.l.: s.n.], 2002. p. 716–723. Citado 2 vezes nas páginas [57](#) e [71](#).

BURLACU, B.; YANG, K.; AFFENZELLER, M. Population diversity and inheritance in genetic programming for symbolic regression. **Natural Computing**, Springer, v. 23, n. 3, p. 531–566, 2024. Citado 2 vezes nas páginas [13](#) e [71](#).

CAMPELO, F.; WANNER, E. F. Sample size calculations for the experimental comparison of multiple algorithms on multiple problem instances. **Journal of Heuristics**, Springer, v. 26, n. 6, p. 851–883, 2020. Citado na página [67](#).

CARPENTER, G. A.; GROSSBERG, S.; REYNOLDS, J. H. Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. **Neural networks**, Elsevier, v. 4, n. 5, p. 565–588, 1991. Citado na página [30](#).

CARPENTER, G. A.; GROSSBERG, S.; ROSEN, D. B. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. **Neural networks**, Elsevier, v. 4, n. 6, p. 759–771, 1991. Citado 2 vezes nas páginas [29](#) e [30](#).

CHANDER, G. P.; DAS, S. Chaotic rao3 based adaptive neuro-fuzzy inference system to solve global infrastructure project selection problem. **Applied Soft Computing**, Elsevier, v. 165, p. 112046, 2024. Citado na página [2](#).

CHEN, T. et al. A decision tree-initialised neuro-fuzzy approach for clinical decision support. **Artificial Intelligence in Medicine**, Elsevier, v. 111, p. 101986, 2021. Citado 3 vezes nas páginas [2](#), [17](#) e [18](#).

CHENG, Y. et al. An improved multiple adaptive neuro fuzzy inference system based on genetic algorithm for energy management system of island microgrid: Y. cheng et al. **Scientific Reports**, Nature Publishing Group UK London, v. 15, n. 1, p. 17988, 2025. Citado na página [22](#).

CORDÓN, O. et al. Ten years of genetic fuzzy systems: current framework and new trends. **Fuzzy sets and systems**, Elsevier, v. 141, n. 1, p. 5–31, 2004. Citado na página [21](#).

CORDÓN, O. et al. **Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases**. [S.l.]: World Scientific, 2001. (Advances in fuzzy systems - applications and theory). ISBN 9789810240165. Citado na página [21](#).

COX, E. Adaptive fuzzy systems. **IEEE Spectrum**, IEEE, v. 30, n. 2, p. 27–31, 1993. Citado na página [16](#).

COX, E. Adaptive fuzzy systems. **IEEE Spectrum**, IEEE, v. 30, n. 2, p. 27–31, 2002. Citado na página [2](#).

CZOGALA, E.; LESKI, J. **Fuzzy and neuro-fuzzy intelligent systems**. [S.l.]: Physica, 2012. v. 47. Citado na página [13](#).

DEVECI, M. et al. A fuzzy einstein-based decision support system for public transportation management at times of pandemic. **Knowledge-Based Systems**, Elsevier, 2022. Citado na página [2](#).

DIRIK, M. Detection of counterfeit banknotes using genetic fuzzy system. **Journal of fuzzy extension and applications**, Ayandegan Institute of Higher Education, Iran, v. 3, n. 4, p. 302–312, 2022. Citado na página [21](#).

EIBEN, A. E.; MARCHIORI, E.; VALKO, V. Evolutionary algorithms with on-the-fly population size adjustment. In: SPRINGER. **Parallel Problem Solving from Nature-PPSN VIII: 8th International Conference, Birmingham, UK, September 18-22, 2004. Proceedings 8**. [S.l.], 2004. p. 41–50. Citado na página [73](#).

EIBEN, A. E.; SMITH, J. E. et al. **Introduction to evolutionary computing**. [S.l.]: Springer, 2003. v. 53. Citado na página [10](#).

ELBORLSY, M. et al. Neuro-fuzzy controller based adaptive control for enhancing the frequency response of two-area power system. **Heliyon**, Elsevier, v. 11, n. 10, 2025. Citado na página [2](#).

ENGELBRECHT, A. P. **Computational intelligence: an introduction**. [S.l.]: John Wiley & Sons, 2007. Citado na página [12](#).

ESCOBAR-CUEVAS, H. et al. A novel hybrid search strategy for evolutionary fuzzy optimization approach. **Neural Computing and Applications**, Springer, v. 36, n. 6, p. 2633–2652, 2024. Citado na página [22](#).

ESMIN, A. A. A.; LAMBERT-TORRES, G. Evolutionary computation based fuzzy membership functions optimization. In: **2007 IEEE International Conference on Systems, Man and Cybernetics**. [S.l.: s.n.], 2007. p. 823–828. Citado na página [19](#).

FARINATI, D.; BAKUROV, I.; VANNESCHI, L. A study of dynamic populations in geometric semantic genetic programming. **Information Sciences**, Elsevier, v. 648, p. 119513, 2023. Citado 2 vezes nas páginas [57](#) e [73](#).

FERNANDEZ, A. et al. Evolutionary fuzzy systems for explainable artificial intelligence: Why, when, what for, and where to? **IEEE Comp. Int. magazine**, p. 69–81, 2019. Citado 2 vezes nas páginas [3](#) e [19](#).

FERNANDEZ, A. et al. Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges. **Knowledge-Based Systems**, v. 80, p. 109 – 121, 2015. ISSN 0950-7051. Citado na página [20](#).

FERREIRA, C. Gene expression programming: a new adaptive algorithm for solving problems. **Complex Systems**, v. 13, n. 2, p. 87–129, 2001. Citado 2 vezes nas páginas [41](#) e [54](#).

FOROUTAN, E. et al. Urban growth modeling based on cellular automata with transition rules optimized using genetic fuzzy systems. **Transactions in GIS**, Wiley Online Library, v. 26, n. 8, p. 3253–3276, 2022. Citado na página [22](#).

FULCHER, J. Computational intelligence: an introduction. In: **Computational intelligence: a compendium**. [S.l.]: Springer, 2008. p. 3–78. Citado na página [15](#).

FULLÉR, R. **Introduction to neuro-fuzzy systems**. [S.l.]: Springer Science & Business Media, 2000. v. 2. Citado na página [14](#).

GAO, J. et al. A novel machine learning method for multiaxial fatigue life prediction: Improved adaptive neuro-fuzzy inference system. **Int. Journal of Fatigue**, Elsevier, 2024. Citado 2 vezes nas páginas [2](#) e [19](#).

GARCEZ, A. d. et al. Neural-symbolic learning and reasoning: A survey and interpretation. **Neuro-Symbolic Artificial Intelligence: The State of the Art**, IOS Press, v. 342, n. 1, p. 327, 2022. Citado 2 vezes nas páginas [2](#) e [17](#).

GIL, P. et al. Neuro-fuzzy tracking control of discrete-time nonlinear systems under linear matrix inequality region constraints. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 153, p. 110672, 2025. Citado na página [2](#).

GU, X. et al. Autonomous learning for fuzzy systems: a review. **Artificial Intelligence Review**, Springer, v. 56, n. 8, p. 7549–7595, 2023. Citado 3 vezes nas páginas [1](#), [75](#) e [76](#).

GULER, N. et al. Adaptive neuro-fuzzy inference system-genetic algorithm approach for global maximum power point tracking in pv systems under different shading conditions. **Green Technologies and Sustainability**, Elsevier, p. 100239, 2025. Citado na página [22](#).

GÜLMEZ, B. Ga-attention-fuzzy-stock-net: An optimized neuro-fuzzy system for stock market price prediction with genetic algorithm and attention mechanism. **Heliyon**, Elsevier, v. 11, n. 3, 2025. Citado 2 vezes nas páginas [2](#) e [22](#).

HAGOS, D. H.; RAWAT, D. B. Recent advances in artificial intelligence and tactical autonomy: Current status, challenges, and perspectives. **Sensors**, MDPI, v. 22, n. 24, p. 9916, 2022. Citado na página [1](#).

HIKMAWATI, E.; MAULIDEVI, N. U.; SURENDRO, K. Pruning strategy on adaptive rule model by sorting utility items. **IEEE Access**, IEEE, v. 10, p. 91650–91662, 2022. Citado 2 vezes nas páginas [75](#) e [76](#).

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. Ann Arbor, MI: University of Michigan Press, 1975. Second edition, 1992. Citado 2 vezes nas páginas [3](#) e [8](#).

HOLLAND, J. H.; REITMAN, J. S. Cognitive systems based on adaptive algorithms. **SIGART Bull.**, ACM, New York, NY, USA, v. 1, n. 63, p. 49–49, 1977. ISSN 0163-5719. Citado na página [21](#).

HUMAIRA, H.; RASYIDAH, R. Determining the appropriate cluster number using elbow method for k-means algorithm. In: **Proc. of the 2nd Workshop on Multidisciplinary and Applications**. [S.l.: s.n.], 2020. Citado 2 vezes nas páginas 44 e 60.

IKOTUN, A. M. et al. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. **Information Sciences**, 2023. Citado 2 vezes nas páginas 44 e 60.

Jang, J. . R. Anfis: adaptive-network-based fuzzy inference system. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 23, n. 3, 1993. Citado na página 37.

JOHANSEN, T. A.; SHORTEN, R.; MURRAY-SMITH, R. On the interpretation and identification of dynamic takagi-sugeno fuzzy models. **IEEE Transactions on Fuzzy systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 8, n. 3, p. 297–313, 2000. Citado na página 16.

JONG, K. D. Evolutionary computation: a unified approach. In: **Proceedings of the Genetic and Evolutionary Computation Conference Companion**. [S.l.: s.n.], 2017. p. 373–388. Citado na página 8.

JOZI, A. et al. Genetic fuzzy rule-based system using mogul learning methodology for energy consumption forecasting. **ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal**, Ediciones Universidad de Salamanca, v. 8, n. 1, p. 55–64, 2019. Citado na página 21.

KACPRZYK, J.; PEDRYCZ, W. **Springer Handbook of Computational Intelligence**. [S.l.]: Springer Publishing Company, Incorporated, 2015. ISBN 3662435047, 9783662435045. Citado 2 vezes nas páginas 12 e 15.

KELLER DERONG LIU, D. B. F. J. M. **Fundamentals of Computational Intelligence. Neural Networks, Fuzzy Systems and Evolutionary Computation**. [S.l.]: Wiley, 2016. ISBN 978-1-110-21434-2. Citado 3 vezes nas páginas 6, 8 e 21.

KHALID, R.; JAVAID, N. A survey on hyperparameters optimization algorithms of forecasting models in smart grid. **Sustainable Cities and Society**, Elsevier, v. 61, p. 102275, 2020. Citado na página 66.

KHANAFARI, A. et al. Experimental study of water management of a three-cell pem fuel cell stack using adaptive neuro-fuzzy approximation and fuzzy control. **Int. Journal of Hydrogen Energy**, Elsevier, p. 931–944, 2024. Citado na página 2.

KOZA, J. R. **Genetic programming: on the programming of computers by means of natural selection**. [S.l.]: MIT press, 1992. v. 1. Citado 3 vezes nas páginas 3, 9 e 10.

KOZA, J. R. Genetic programming as a means for programming computers by natural selection. **Statistics and computing**, Springer, v. 4, n. 2, p. 87–112, 1994. Citado na página 9.

KRAMER, O. Genetic algorithms. In: **Genetic algorithm essentials**. [S.l.]: Springer, 2017. p. 11–19. Citado na página 8.

KRUSE, R. et al. **Computational intelligence: a methodological introduction**. [S.l.]: Springer, 2016. Citado na página 14.

LAM, H.-K. Polynomial fuzzy-model-based control systems: Stability analysis via piecewise-linear membership functions. **IEEE Trans. on Fuzzy Systems**, IEEE, p. 588–593, 2011. Citado na página [16](#).

LAMBORA, A.; GUPTA, K.; CHOPRA, K. Genetic algorithm-a literature review. In: IEEE. **2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)**. [S.l.], 2019. p. 380–384. Citado na página [9](#).

LANGDON, W. B.; POLI, R. **Foundations of genetic programming**. [S.l.]: Springer Science & Business Media, 2013. Citado na página [9](#).

LI, H.; WANG, Y.; ZHANG, G. Probabilistic fuzzy classification for stochastic data. **IEEE Transactions on Fuzzy Systems**, v. 25, n. 6, p. 1391–1402, Dec 2017. ISSN 1063-6706. Citado na página [19](#).

LI, W.; HE, L.; CAO, Y. Many-objective evolutionary algorithm with reference point-based fuzzy correlation entropy for energy-efficient job shop scheduling with limited workers. **IEEE Transactions on Cybernetics**, IEEE, v. 52, n. 10, p. 10721–10734, 2021. Citado na página [22](#).

LIN, C.-J.; LIN, C.-T. An art-based fuzzy adaptive learning control network. **IEEE Transactions on Fuzzy Systems**, IEEE, v. 5, n. 4, p. 477–496, 1997. Citado na página [30](#).

LIN, C.-T.; LEE, C. G. **Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems**. [S.l.]: Prentice-Hall, Inc., 1996. Citado na página [26](#).

LIN, C.-T.; LEE, C. S. G. Neural-network-based fuzzy logic control and decision system. **IEEE Transactions on computers**, IEEE Computer Society, v. 40, n. 12, p. 1320–1336, 1991. Citado 2 vezes nas páginas [24](#) e [25](#).

LUGHOFER, E. Evolving fuzzy systems—fundamentals, reliability, interpretability, useability, applications. In: **Hand. on comp. intel.: fuzzy logic, systems, art. neural networks, and learn. systems**. [S.l.: s.n.], 2016. Citado na página [62](#).

LUGHOFER, E. Variants of recursive consequent parameters learning in evolving neuro-fuzzy systems. In: IEEE. **2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)**. [S.l.], 2020. p. 1–10. Citado na página [63](#).

LUGHOFER, E. D. Flexfis: A robust incremental learning approach for evolving takagi-sugeno fuzzy models. **IEEE Trans. on fuzzy systems**, 2008. Citado na página [63](#).

MADBOULY, M. M.; REHEEM, E. E.; GUIRGUIS, S. K. Interval type-2 fuzzy logic using genetic algorithm to reduce redundant association rules. **J. Theor. Appl. Inf. Technol**, v. 99, n. 2, p. 316–328, 2021. Citado 2 vezes nas páginas [75](#) e [76](#).

MAMAGHANI, A. S.; PEDRYCZ, W. Genetic-programming-based architecture of fuzzy modeling: towards coping with high-dimensional data. **IEEE Trans. on Fuzzy Systems**, IEEE, p. 2774–2784, 2020. Citado 3 vezes nas páginas [2](#), [17](#) e [21](#).

MAMDANI, E. H.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. **International Journal of Man-Machine Studies**, v. 7, n. 1, p. 1–13, 1975. Citado 2 vezes nas páginas [15](#) e [24](#).

MARKOVICS, D.; MAYER, M. J. Comparison of machine learning methods for photovoltaic power forecasting based on numerical weather prediction. **Renewable and Sustainable Energy Reviews**, Elsevier, v. 161, p. 112364, 2022. Citado na página [66](#).

MASOUMI, A.; TAVAKOLPOUR-SALEH, A.; RAHIDEH, A. Applying a genetic-fuzzy control scheme to an active free piston stirling engine: Design and experiment. **Applied energy**, Elsevier, v. 268, p. 115045, 2020. Citado na página [21](#).

MATÍA, F.; MARICHAL, G. N.; JIMÉNEZ, E. **Fuzzy Modeling and Control: Theory and Applications**. [S.l.]: Springer, 2014. ISBN 978-94-6239-081-2. Citado na página [16](#).

MAZOUZI, A. et al. Comprehensive optimization of fuzzy logic-based energy management system for fuel-cell hybrid electric vehicle using genetic algorithm. **International Journal of Hydrogen Energy**, Elsevier, v. 81, p. 889–905, 2024. Citado na página [22](#).

MEHMOOD, F. et al. Optimizing english language evaluation: A hybrid ai framework using neural networks and fuzzy decision-making models. **The Sciencetech**, p. 187–204, 2025. Citado na página [2](#).

MEI, Z.; ZHAO, T.; XIE, X. Hierarchical fuzzy regression tree: A new gradient boosting approach to design a tsf fuzzy model. **Information Sciences**, Elsevier, v. 652, p. 119740, 2024. Citado na página [19](#).

Min Han et al. Prediction of chaotic time series based on the recurrent predictor neural network. **IEEE Transactions on Signal Processing**, v. 52, n. 12, p. 3409–3416, 2004. Citado na página [2](#).

MOGHADDAM, M. J. A survey on genetic fuzzy systems. **Archives of Computational Methods in Engineering**, Springer, p. 1–46, 2024. Citado 3 vezes nas páginas [2](#), [19](#) e [75](#).

MOUTAOUAKIL, K. E.; ROUDANI, M.; OUISSARI, A. E. Optimal entropy genetic fuzzy-c-means smote (oegfcm-smote). **Knowledge-Based Systems**, Elsevier, v. 262, p. 110235, 2023. Citado na página [22](#).

NASIRI, H.; EBADZADEH, M. M. Mfrfnn: Multi-functional recurrent fuzzy neural network for chaotic time series prediction. **Neurocomputing**, Elsevier, v. 507, p. 292–310, 2022. Citado na página [19](#).

NIAZKAR, M. Multigene genetic programming and its various applications. **Handbook of hydroinformatics**, Elsevier, p. 321–332, 2023. Citado 2 vezes nas páginas [41](#) e [54](#).

OLIVEIRA, L. O. V. et al. Sequential symbolic regression with genetic programming. In: **Genetic Programming Theory and Practice XII**. [S.l.]: Springer, 2015. p. 73–90. Citado na página [9](#).

PANTULA, P. D.; MIRIYALA, S. S.; MITRA, K. An evolutionary neuro-fuzzy c-means clustering technique. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 89, p. 103435, 2020. Citado na página [18](#).

PEDRYCZ, W.; GOMIDE, F. **Fuzzy systems engineering: toward human-centric computing**. [S.l.]: John Wiley & Sons, 2007. Citado na página [17](#).

PIETROPOLLI, G. et al. A genetic programming based heuristic to simplify rugged landscapes exploration. **Emerging Science Journal**, p. 1037–1051, 2023. Citado na página [22](#).

POLI, R.; KOZA, J. Genetic programming. In: **Search methodologies: Introductory tutorials in optimization and decision support techniques**. [S.l.]: Springer, 2013. p. 143–185. Citado na página [13](#).

PROBST, P.; BOULESTEIX, A.-L.; BISCHL, B. Tunability: Importance of hyperparameters of machine learning algorithms. **Journal of Machine Learning Research**, v. 20, n. 53, p. 1–32, 2019. Citado na página [66](#).

PROBST, P.; WRIGHT, M. N.; BOULESTEIX, A.-L. Hyperparameters and tuning strategies for random forest. **Wiley Interdisciplinary Reviews: data mining and knowledge discovery**, Wiley Online Library, v. 9, n. 3, p. e1301, 2019. Citado na página [66](#).

PUSTOKHINA, I. V. et al. Energy efficient neuro-fuzzy cluster based topology construction with metaheuristic route planning algorithm for unmanned aerial vehicles. **Computer Networks**, Elsevier, v. 196, p. 108214, 2021. Citado 2 vezes nas páginas [2](#) e [17](#).

QUADRELLI, G.; TANSCHKEIT, R.; VELLASCO, M. Neuro-fuzzy modelling and control of nonlinear dynamic systems. **Learning and Nonlinear Models—Revista da Sociedade Brasileira de Redes Neurais**, v. 1, n. 1, 2003. Citado na página [26](#).

QUAH, K. H.; QUEK, C.; LEEDHAM, G. Reinforcement learning combined with a fuzzy adaptive learning control network (falcon-r) for pattern classification. **Pattern Recognition**, Elsevier, v. 38, n. 4, p. 513–526, 2005. Citado na página [24](#).

RAJAN, M. S. et al. Diagnosis of fault node in wireless sensor networks using adaptive neuro-fuzzy inference system. **Applied Nanoscience**, Springer, v. 13, n. 2, p. 1007–1015, 2023. Citado 2 vezes nas páginas [2](#) e [17](#).

RAN, X. et al. A hybrid genetic-fuzzy ant colony optimization algorithm for automatic k-means clustering in urban global positioning system. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 137, p. 109237, 2024. Citado na página [22](#).

RAO, S. B. et al. An adaptive neuro-fuzzy control strategy for improved power quality in multi-microgrid clusters. **IEEE Access**, IEEE, 2022. Citado na página [2](#).

RAVICHANDRAN, B. D.; KEIKHOSROKIANI, P. Classification of covid-19 misinformation on social media based on neuro-fuzzy and neural network: A systematic review. **Neural Comp. and Applications**, Springer, p. 699–717, 2023. Citado 2 vezes nas páginas [1](#) e [2](#).

REJEESH, M. Interest point based face recognition using adaptive neuro fuzzy inference system. **Multimedia Tools and Applications**, Springer, v. 78, n. 16, p. 22691–22710, 2019. Citado na página [18](#).

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. [S.l.]: Malaysia; Pearson Education Limited,, 2016. Citado na página [10](#).

SAADATMAND, H.; AKBARZADEH-T, M.-R. Set-based integer-coded fuzzy granular evolutionary algorithms for high-dimensional feature selection. **Applied Soft Computing**, Elsevier, v. 142, p. 110240, 2023. Citado na página [22](#).

SABAH, M. et al. Application of decision tree, artificial neural networks, and adaptive neuro-fuzzy inference system on predicting lost circulation: A case study from marun oil field. **Journal of Petroleum Science and Engineering**, Elsevier, v. 177, p. 236–249, 2019. Citado na página 18.

SAJID, M. et al. Neuro-fuzzy random vector functional link neural network for classification and regression problems. **IEEE Transactions on Fuzzy Systems**, IEEE, v. 32, n. 5, p. 2738–2749, 2024. Citado na página 19.

SALAMEH, T. et al. Adaptive neuro-fuzzy inference system for accurate power forecasting for on-grid photovoltaic systems: A case study in sharjah, uae. **Energy Conversion and Management: X**, Elsevier, v. 26, p. 100958, 2025. Citado na página 2.

SEDGWICK, P. Multiple hypothesis testing and bonferroni's correction. **Bmj**, British Medical Journal Publishing Group, v. 349, 2014. Citado na página 67.

SHAFFER, J. P. Multiple hypothesis testing. **Annual review of psychology**, v. 46, n. 1, p. 561–584, 1995. Citado na página 67.

SHI, Y.; MIZUMOTO, M. Some considerations on conventional neuro-fuzzy learning algorithms by gradient descent method. **Fuzzy Sets and Systems**, v. 112, n. 1, p. 51–63, 2000. ISSN 0165-0114. Citado na página 2.

SHIHABUDHEEN, K.; PILLAI, G. Recent advances in neuro-fuzzy system. **Know.-Based Syst.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 152, n. C, p. 136–162, jul. 2018. ISSN 0950-7051. Citado 3 vezes nas páginas 1, 13 e 14.

SHRIKUMAR, A.; GREENSIDE, P.; KUNDAJE, A. Learning important features through propagating activation differences. In: PMLR. **International conference on machine learning**. [S.l.], 2017. p. 3145–3153. Citado na página 18.

SMITH, S. F. **A Learning System Based on Genetic Adaptive Algorithms**. Tese (Doutorado) — University of Pittsburgh, Pittsburgh, PA, USA, 1980. Citado 3 vezes nas páginas 21, 41 e 54.

SOLIMAN, M. A. et al. Hybrid anfis-ga-based control scheme for performance enhancement of a grid-connected wind generator. **IET Renewable Power Generation**, IET, v. 12, n. 7, p. 832–843, maio 2018. ISSN 1752-1424. Citado na página 21.

SOUZA, P. V. de C. Fuzzy neural networks and neuro-fuzzy networks: A review the main techniques and applications used in the literature. **Applied soft computing**, Elsevier, v. 92, p. 106275, 2020. Citado na página 1.

SOUZA, P. V. de C. Fuzzy neural networks and neuro-fuzzy networks: A review the main techniques and applications used in the literature. **Applied soft computing**, Elsevier, v. 92, 2020. Citado na página 18.

STORN, R.; PRICE, K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. **Journal of global optimization**, Springer, v. 11, n. 4, p. 341–359, 1997. Citado na página 3.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. **IEEE Transactions on Systems, Man, and Cybernetics**, p. 116–132, Jan 1985. Citado 2 vezes nas páginas 16 e 37.

TEŠIĆ, D. et al. Development of the mcdm fuzzy Imaw-grey marcos model for selection of a dump truck. **Reports in Mechanical Engineering**, p. 1–17, 2023. Citado na página 2.

TICKLE, A. B. et al. The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. **IEEE Transactions on Neural Networks**, IEEE, v. 9, n. 6, p. 1057–1068, 1998. Citado na página 17.

TOWNSEND, J.; CHATON, T.; MONTEIRO, J. M. Extracting relational explanations from deep neural networks: A survey from a neural-symbolic perspective. **IEEE transactions on neural networks and learning systems**, IEEE, v. 31, n. 9, p. 3456–3470, 2019. Citado 3 vezes nas páginas 2, 17 e 18.

VARSHNEY, A. K.; TORRA, V. Literature review of the recent trends and applications in various fuzzy rule-based systems. **International Journal of Fuzzy Systems**, Springer, v. 25, n. 6, p. 2163–2186, 2023. Citado na página 15.

VENTURINI, G. Sia: A supervised inductive algorithm with genetic search for learning attributes based concepts. In: BRAZDIL, P. B. (Ed.). **Machine Learning: ECML-93**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993. p. 280–296. ISBN 978-3-540-47597-2. Citado na página 21.

VIRGOLIN, M. et al. Improving model-based genetic programming for symbolic regression of small expressions. **Evolutionary computation**, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 29, n. 2, p. 211–237, 2021. Citado na página 9.

VIVEKANANDHAN, V.; SAKTHIVEL, S.; MANIKANDAN, M. Adaptive neuro fuzzy inference system to enhance the classification performance in smart irrigation system. **Computational Intelligence**, 2022. Citado 2 vezes nas páginas 1 e 2.

WATSON, D. P.; SCHEIDT, D. H. Autonomous systems. **Johns Hopkins APL technical digest**, v. 26, n. 4, p. 368–376, 2005. Citado na página 1.

WONG, Y. Fuzzy adaptive learning control network with policy and another adaptive resonance theory (falcon-paart) embedded deep structure with applications in stock market prediction and analysis. Nanyang Technological University, 2022. Citado na página 24.

XUE, G. et al. An adaptive neuro-fuzzy system with integrated feature selection and rule extraction for high-dimensional classification problems. **IEEE Transactions on Fuzzy Systems**, IEEE, 2022. Citado 3 vezes nas páginas 1, 2 e 18.

YAKUBU, H.; KWONG, C.; LEE, C. K. A multigene genetic programming-based fuzzy regression approach for modelling customer satisfaction based on online reviews. **Soft Computing**, 2021. Citado na página 21.

YAMAKAWA, T. et al. A neo fuzzy neuron and its applications to system identification and predictions to system behavior. In: **Proceedings of the International Conference on Fuzzy Logic and Neural Networks**. [S.l.]: IEEE, 1992. p. 477–484. ISBN 0780311043. Citado na página 37.

ZADEH, L. Fuzzy sets. **Information and Control**, v. 8, n. 3, p. 338 – 353, 1965. ISSN 0019-9958. Citado na página 15.

ZADEH, L. A. The concept of a linguistic variable and its application to approximate reasoning—i. **Information sciences**, Elsevier, v. 8, n. 3, p. 199–249, 1975. Citado na página [15](#).

ZHANG, H. et al. Application of gradient boosting in the design of fuzzy rule-based regression models. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 36, n. 11, p. 5621–5632, 2024. Citado na página [19](#).

ZHANG, X. et al. A decomposition-based evolutionary algorithm with clustering and hierarchical estimation for multi-objective fuzzy flexible jobshop scheduling. **IEEE Transactions on Evolutionary Computation**, IEEE, 2024. Citado na página [22](#).

ZINTGRAF, L. M. et al. Visualizing deep neural network decisions: Prediction difference analysis. **arXiv preprint arXiv:1702.04595**, 2017. Citado na página [18](#).