UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

LUCAS DA SILVA ALVES

OTIMIZAÇÃO DE PARÂMETROS DE REDES NEURAIS DO TIPO FEEDFORWARD COM ALGORITMOS META-HEURÍSTICOS

Porto Alegre
2024

LUCAS DA SILVA ALVES

OTIMIZAÇÃO DE PARÂMETROS DE REDES NEURAIS DO TIPO FEEDFORWARD COM ALGORITMOS META-HEURÍSTICOS

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Engenheiro Eletricista.

ORIENTADOR: Prof. Dr. Tiago Oliveira Weber

Porto Alegre
2024

LUCAS DA SILVA ALVES

OTIMIZAÇÃO DE PARÂMETROS DE REDES NEURAIS DO TIPO FEEDFORWARD COM ALGORITMOS META-HEURÍSTICOS

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Engenheiro Eletricista.

Orientador: _____
Prof. Dr. Tiago Oliveira Weber, UFRGS
Doutor pela Universidade de São Paulo – São Paulo, Brasil

Banca Examinadora:

Prof. Dr. Carlos Eduardo Pereira, UFRGS
Doutor pela Stuttgart University – Stuttgart, ALEMANHA.

Prof. Dr. Ivan Müller, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul

Prof. Dr. Tiago Oliveira Weber, UFRGS
Doutor pela Universidade de São Paulo – São Paulo, Brasil

Chefe do DELET: _____
Prof. Dr. Alexandre Balbinot

Porto Alegre, agosto de 2024.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por me conceder força, saúde e sabedoria para enfrentar os desafios ao longo dessa jornada acadêmica.

Agradeço à Universidade Federal do Rio Grande do Sul (UFRGS) pela oportunidade de realizar este curso e crescer tanto pessoal quanto profissionalmente.

Ao meu professor orientador, Tiago Oliveira Weber, expresso minha gratidão pela orientação, paciência e conhecimento compartilhado, que foram essenciais para a conclusão deste trabalho.

Minha profunda gratidão à minha família, especialmente aos meus pais e à minha irmã, pelo apoio incondicional, amor e incentivo constante.

A minha namorada, Milene, agradeço pelo carinho, compreensão e por estar ao meu lado em todos os momentos, me apoiando, me inspirando e sempre me motivando a seguir em frente.

Aos demais professores do curso de Engenharia Elétrica, obrigado por todo o conhecimento transmitido e pelas valiosas contribuições ao longo dessa caminhada.

Aos meus colegas de curso Leomar, Gabriel, Vinícius, Morgan e Gus, minha gratidão pelas noites de estudo, amizade, parceria e apoio mútuo que tornaram essa jornada mais leve e especial.

Um agradecimento especial à Atlética da Engenharia, pela experiência inesquecível e por ter me proporcionado momentos únicos. Em especial, agradeço aos amigos Juan e Corazza, pela amizade, parceria e pelas memórias que levarei para toda a vida.

E, por fim, agradeço a todos os amigos que, de alguma forma, contribuíram para a minha formação e para a realização deste trabalho. Cada um de vocês tem um lugar especial na minha trajetória.

RESUMO

O campo das meta-heurísticas tem experimentado um crescimento notável tanto em termos de interesse acadêmico quanto de produção científica. As meta-heurísticas, uma classe de métodos de otimização inspirados por processos naturais e heurísticas adaptativas, têm desempenhado um papel crucial na solução de problemas complexos em diversos domínios, como, por exemplo, na otimização de redes neurais, design de sistemas e análise de dados. Este trabalho teve como objetivo implementar e analisar combinações de algoritmos convencionais e meta-heurísticos para otimização de parâmetros em Redes Neurais Feedforward (FNNs), utilizando tanto técnicas de busca local quanto global. Utilizou-se o Método Experimental para comparar quatro métodos de otimização: aleatório (RAND), Algoritmo Genético (GA), Otimização por Enxame de Partículas (PSO) e um híbrido GA-PSO (HGAPSO), aplicados a três datasets distintos. Os testes mostraram que o PSO foi o mais eficaz, seguido pelo GA, enquanto o HGAPSO apresentou desempenho intermediário e o RAND teve maior variabilidade e desempenho inferior. A análise estatística confirmou a eficácia das meta-heurísticas em comparação com abordagens aleatórias. Os resultados destacam o PSO como a técnica mais eficaz para otimização de FNNs nos problemas de regressão escolhidos, de acordo com o comparativo da métrica RMSE e o tempo de execução dos algoritmos como um limitante.

Palavras-chave: Inteligencia Artificial, Aprendizado de Máquina, Otimização de Parâmetros, Redes Neurais Feedforward, Meta-Heurísticas, Backpropagation, Algoritmos Híbridos, Algorítmos Genéticos, Otimização por Enxame de Partículas.

ABSTRACT

The field of metaheuristics has experienced remarkable growth in both academic interest and scientific output. Metaheuristics, a class of optimization methods inspired by natural processes and adaptive heuristics, have played a crucial role in solving complex problems across various domains, such as neural network optimization, system design, and data analysis. This study aimed to implement and analyze combinations of conventional and metaheuristic algorithms for parameter optimization in Feedforward Neural Networks (FNNs), utilizing both local and global search techniques. The Experimental Method was employed to compare four optimization methods: random (RAND), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and a hybrid GA-PSO (HGAPSO), applied to three different datasets. The tests showed that PSO was the most effective, followed by GA, while HGAPSO performed intermediately, and RAND exhibited greater variability and inferior performance. Statistical analysis confirmed the effectiveness of metaheuristics compared to random approaches. The results highlight PSO as the most effective technique for optimizing FNNs in the selected regression problems, considering the RMSE metric and the execution time of the algorithms as a limiting factor.

Keywords: Keywords: Artificial Intelligence, Machine Learning, Parameter Optimization, Feedforward Neural Networks, Metaheuristics, Backpropagation, Hybrid Algorithms, Genetic Algorithms, Particle Swarm Optimization..

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

BP	*Backpropagation*
IA	*Inteligência Artificial*
ML	*Machine Learning*
ANN	*Artificial Neural Network*
FNN	*Feedforward Neural Network*
MLP	*Multi Layer Perceptron*
GA	*Genetic Algorithm*
PSO	*Particle Swarm Optimization*
$HGAPSO$	*Hybrid Genetic Algorithm and Particle Swarm Optimization*
LHS	*Latin Hypercube Sampling*

SUMÁRIO

1 INTRODUÇÃO

Segundo LUKE (2013), meta-heurísticas são uma classe de algoritmos utilizados para resolver problemas complexos de otimização que não possuem uma solução clara ou métodos sistemáticos para encontrá-la. Elas pertencem ao campo da otimização estocástica, que é caracterizado pelo uso de elementos de aleatoriedade para buscar soluções que sejam o mais próximo possível da ideal. A ideia fundamental por trás das meta-heurísticas é que, para problemas em que não é possível determinar uma solução ótima de maneira direta ou onde uma busca exaustiva seria inviável devido ao tamanho do espaço de soluções, essas técnicas permitem explorar o espaço de soluções de maneira inteligente.

Nos últimos 20 anos, o campo das meta-heurísticas tem experimentado um crescimento notável tanto em termos de interesse acadêmico quanto de produção científica. As meta-heurísticas, uma classe de métodos de otimização inspirados por processos naturais e heurísticas adaptativas, têm desempenhado um papel crucial na solução de problemas complexos em diversos domínios, como por exemplo na otimização de redes neurais, design de sistemas e análise de dados (OJHA; ABRAHAM; SNÁŠEL, 2017).

O crescimento significativo do número de publicações sobre meta-heurísticas reflete a crescente importância desse campo. Na Figura 1, temos o resultado da pesquisa do termo *metaheuristic* no site do IEEE Xplore, onde podemos ver que em 2004, foram registrados 17 artigos em conferências e 5 em *journals*. Desde então, o número de publicações tem aumentado de maneira exponencial. Em 2023, por exemplo, foram publicados 1192 artigos em conferências e 313 em *journals*, destacando uma tendência contínua de crescimento e expansão do interesse na área.

O impacto das meta-heurísticas é amplamente reconhecido na academia e na indústria, proporcionando soluções eficazes para problemas de otimização que são difíceis de resolver por métodos tradicionais. A aplicação de meta-heurísticas tem se mostrado fundamental para

Figura 1 – Resultado da pesquisa do termo *metaheuristic*.

Fonte: Autor com dados do site (IEEE, 2024).

avançar em áreas como inteligência artificial, aprendizado de máquina e engenharia de sistemas complexos. A crescente quantidade de pesquisas e publicações sobre meta-heurísticas ressalta não apenas a sua relevância, mas também o dinamismo e a inovação contínua neste campo, que continua a atrair atenção e investimentos significativos da comunidade científica e tecnológica (OJHA; ABRAHAM; SNÁŠEL, 2017).

1.1 Contextualização

Assim como o tema sobre meta-heurísticas vem ganhando popularidade, a otimização de redes neurais feedforward (FNN) tem sido um foco de intenso interesse tanto para pesquisadores quanto para profissionais em diversas áreas. A otimização de FNNs pode ser abordada de várias maneiras, incluindo a otimização dos pesos, da estrutura da rede, das funções de ativação e dos parâmetros de aprendizado. Conforme OJHA; ABRAHAM; SNÁŠEL (2017) essas abordagens visam principalmente melhorar a capacidade de generalização das redes neurais. Tradicionalmente, o algoritmo de gradiente descendente, incluindo o método de retropropagação (*backpropagation*), tem sido amplamente utilizado para essa tarefa, demonstrando sucesso em vários problemas do mundo real. Contudo, devido às limitações dos métodos baseados em gradientes, como a sensibilidade aos parâmetros e a tendência a cair em mínimos locais, algoritmos meta-heurísticos, como algoritmos evolutivos e inteligência de enxame, estão sendo explorados para obter FNNs mais generalizadas e eficazes para problemas específicos.

Uma FNN é composta por múltiplos neurônios organizados em camadas, onde cada neurônio de uma camada se conecta aos neurônios da camada anterior por meio de pesos. A otimização ou treinamento de uma FNN envolve a busca pela estrutura de rede apropriada e a determinação dos pesos ideais. Isso inclui definir o número de neurônios, suas funções de ativação e a configuração da rede. Além disso, a otimização dos pesos é crucial e envolve ajustar um vetor que representa os pesos da rede (HAYKIN, 2009).

1.1.1 Abordagens Convencionais de Otimização

Os algoritmos convencionais baseados em gradiente, como por exemplo o **backpropagation**, trabalham com uma única solução (vetor de pesos) durante a otimização, tornando-os mais rápidos do que os algoritmos que utilizam múltiplos vetores de solução. Esses métodos são aplicáveis tanto ao treinamento estocástico quanto ao modo em lote das Redes Neurais Feedforward (FNNs), proporcionando um desempenho computacional eficiente (RUMELHART; HINTON; WILLIAMS, 1986).

O treinamento estocástico oferece vantagens, como a capacidade de lidar com padrões redundantes e incluir dados não presentes no conjunto de treinamento atual, permitindo uma aprendizagem mais dinâmica e rápida em comparação com o modo em lote. Embora o treinamento em lote possa garantir um mínimo local e ser mais rápido com datasets maiores, ele é menos flexível. Encontrar um algoritmo eficiente para otimização de FNNs tem sido um desafio contínuo. Métodos tradicionais baseados em gradientes, tratam a otimização de FNNs como um problema sem restrições e tentam minimizar uma função de custo. No entanto, esses métodos têm limitações, como a sensibilidade à taxa de aprendizado e ao fator de impulso, e a tendência de cair em mínimos locais. Além disso, os métodos convencionais têm dificuldade em otimizar a estrutura da rede e podem ser limitados a problemas de otimização contínuos (GORI; TESI *et al.*, 1992).

Portanto, a necessidade de métodos mais robustos levou ao surgimento de algoritmos meta-heurísticos. Métodos meta-heurísticos, como o algoritmo genético e o de otimização por enxame de partículas, oferecem uma abordagem alternativa, otimizando simultaneamente todos os componentes da FNN e reduzindo sua complexidade estrutural, o que pode melhorar a generalização da rede OJHA; ABRAHAM; SNÁŠEL (2017).

1.1.2 Algoritmos Meta-heurísticos e Abordagens Híbridas

Os algoritmos meta-heurísticos, inspirados na natureza e no comportamento coletivo, como o Algoritmo Genético (GA) (HOLLAND, 1992) e a Otimização por Enxame de Partículas (PSO) (EBERHART; KENNEDY, 1995), oferecem uma abordagem alternativa para otimização de FNNs. Esses algoritmos são capazes de lidar com problemas complexos, não lineares e não diferenciáveis, e oferecem uma melhor capacidade de exploração global do espaço de busca.

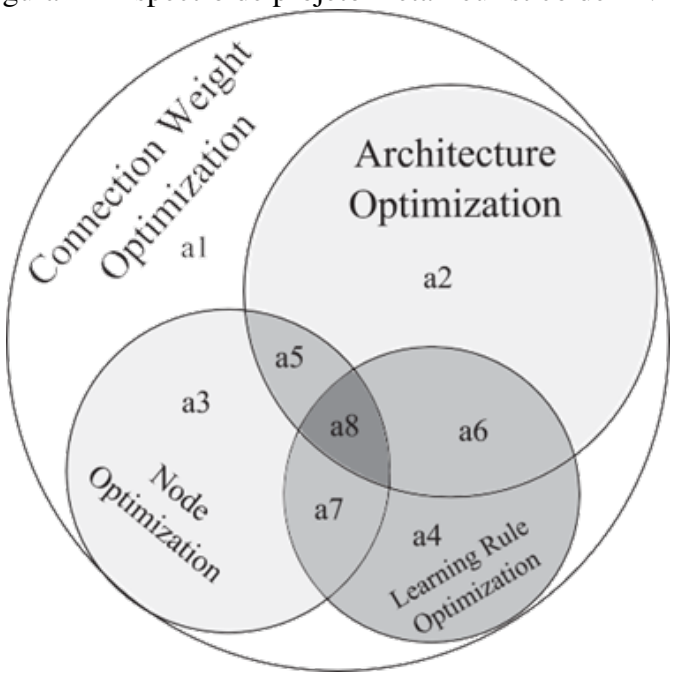
A combinação de diferentes algoritmos meta-heurísticos, conhecida como abordagem híbrida, pode superar as limitações dos métodos convencionais e dos algoritmos meta-heurísticos individuais. Por exemplo, algoritmos híbridos podem combinar a exploração global de meta-heurísticas com a exploração local de métodos baseados em gradiente, proporcionando uma busca mais eficiente por soluções ótimas (MOSCATO *et al.*, 1989).

Segundo JUANG (2004), vemos uma abordagem inovadora para empregar um algoritmo híbrido que combina o Algoritmo Genético (GA) com a Otimização por Enxame de Partículas (PSO). Enquanto o GA evolui indivíduos ao longo das gerações, o PSO aprimora o desempenho dos melhores indivíduos por meio da troca de conhecimento social. O HGAPSO integra o PSO para otimizar os indivíduos de alto desempenho no GA antes de sua reprodução, com o objetivo de gerar soluções mais eficazes e eliminar as menos eficientes.

1.2 Problema de Pesquisa

O presente trabalho tem como principal problema de pesquisa o estudo de métodos modernos de inteligência artificial para auxílio na otimização dos componentes das FNNs. Na Figura 2, temos uma ilustração do espectro de otimização dos componentes das FNNs: pesos, arquitetura, função de ativação e parâmetros da regra de aprendizado. A área "a1"indica a otimização dos pesos; a área "a2"indica a otimização dos pesos e da arquitetura, área "a3"adiciona a otimização da função de ativação; e assim todas as outras combinações possíveis são representadas até a "a8". Pode-se dizer que a força e a complexidade da otimização aumentam da área denotada "a1"para "a8", em que "a1"é a abordagem mais simples e "a8"é a abordagem mais sofisticada.

Figura 2 – Espectro de projeto meta-heurístico de FNNs.



Fonte: OJHA; ABRAHAM; SNÁŠEL (2017)

O escopo deste trabalho será restrito à área "a2", onde abordaremos a otimização das Redes Neurais Feedforward (FNNs) em dois aspectos: ajustando os pesos por meio do método convencionais de otimização, *backpropagation* e otimizando a arquitetura quanto ao número de camadas ocultas e número de neurônios por camada, utilizando algoritmos meta-heurísticos.

1.3 Justificativa

A escolha deste tema é motivada pela crescente necessidade de melhorar a capacidade de generalização e o desempenho de FNNs em problemas complexos. A otimização eficiente de FNNs é crucial para a aplicação bem-sucedida dessas redes em áreas como reconhecimento de padrões, previsão e controle. Métodos tradicionais têm mostrado limitações, principalmente em problemas com grandes espaços de busca e muitas variáveis. Portanto, a exploração de algoritmos meta-heurísticos oferece uma abordagem promissora para superar essas limitações e obter soluções mais eficazes e robustas (OJHA; ABRAHAM; SNÁŠEL, 2017).

1.4 Objetivos

1.4.1 Objetivo Geral

Implementar e analisar combinações entre algoritmos convencionais e meta-heurísticos para otimização de parâmetros em Redes Neurais Feedforward (FNNs), aproveitando as vantagens tanto das técnicas de busca local quanto das de busca global.

1.4.2 Objetivos Específicos

1. Apresentar conceitos básicos de otimização;
2. Implementar e analisar uma abordagem híbrida entre métodos convencionais como o **backpropagation**, com meta-heurísticas para otimização de problemas de regressão;
3. Implementar e analisar uma combinação híbrida de dois ou mais algoritmos meta-heurísticos para fazer uso de uma influência combinada de duas ou mais heurísticas de busca global;
4. Realizar uma análise estatística dos resultados para os diferentes métodos.

1.5 Formulação de Hipóteses

São estabelecidas as hipóteses - **($\mathbf{H}_0$ e $\mathbf{H}_1$)** - para a dedução lógica, por meio da comparação de resultados experimentais e avaliação da melhoria quantitativa de acurácia do algoritmo.

- $\mathbf{H}_0$: Não há diferença significativa no RMSE e no tempo de treinamento entre uma FNN com parâmetros de arquitetura definidos aleatoriamente e uma FNN otimizada com métodos meta-heurísticos (GA, PSO, HGAPSO). Uma FNN com parâmetros aleatórios e uma FNN com parâmetros otimizados têm desempenho equivalente.
- $\mathbf{H}_1$: Existe uma diferença significativa no RMSE e no tempo de treinamento entre uma FNN com parâmetros de arquitetura definidos aleatoriamente e uma FNN otimizada com métodos meta-heurísticos (GA, PSO, HGAPSO). Uma FNN com parâmetros aleatórios e uma FNN com parâmetros otimizados têm desempenhos diferentes.

1.6 Motivação

Os projetos modernos enfrentam crescente complexidade e exigências rigorosas, tornando essencial a adoção de técnicas avançadas de otimização. Embora as técnicas clássicas ainda

sejam úteis, os métodos modernos oferecem soluções mais eficientes e adaptáveis, melhorando a produtividade e a qualidade dos resultados. Além disso, os avanços em inteligência artificial têm transformado a otimização de projetos, com modelos inteligentes, como algoritmos de aprendizado de máquina e redes neurais, aprimorando a eficiência e a eficácia tanto no desenvolvimento de projetos quanto em diversos setores de negócios.

Como consultor de negócios em uma empresa multinacional, o autor demonstra grande interesse na área de ciência de dados e análises preditivas, com ênfase na construção de modelos que auxiliam na tomada de decisões estratégicas em diversos setores de mercado.

1.7 Trabalhos Relacionados

Os estudos selecionados fornecem uma visão abrangente sobre a aplicação de algoritmos meta-heurísticos na otimização de redes neurais, abordando objetivos semelhantes e temas correlacionados ao nosso trabalho.

O presente trabalho foi inspirado pelos desafios e direções propostas no artigo *Metaheuristic design of feedforward neural networks: A review of two decades of research*, de OJHA; ABRAHAM; SNÁŠEL (2017) que revisa duas décadas de pesquisa sobre o design de redes neurais feedforward utilizando técnicas metaheurísticas. Esse artigo destaca as oportunidades e os desafios na aplicação de algoritmos meta-heurísticos para a otimização de redes neurais, fornecendo uma base teórica sólida que orientou nossa investigação sobre a eficácia comparativa de diferentes algoritmos de otimização.

Além disso, o principal artigo norteador do método híbrido abordado em nosso estudo é *A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design*. Este artigo de JUANG (2004) é fundamental para entender a combinação dos algoritmos GA e PSO no contexto de redes neurais recorrentes, fornecendo uma base essencial para a aplicação e adaptação desses métodos na nossa pesquisa.

Outro ponto de referência importante foi o estudo de MENDES *et al.* (2021), intitulado "Abordagens Evolutivas para Otimização de Redes Neurais Convolucionais Baseadas em Algoritmos Genéticos". Este trabalho destacou como os algoritmos genéticos podem ser aplicados para otimizar tanto a arquitetura das redes neurais convolucionais (CNNs) quanto os processos de aprendizado por transferência.

O estudo "Busca por Arquiteturas de Redes Neurais Artificiais Profundas Utilizando Algoritmos Genéticos" de RIBEIRO *et al.* (2022) abordou a aplicação de algoritmos genéticos para

automatizar a construção de arquiteturas de redes neurais profundas. A proposta do algoritmo ENAS (Evolutionary Neural Architecture Search) mostrou-se eficaz na busca por arquiteturas otimizadas para a classificação de imagens, evidenciando a capacidade dos algoritmos genéticos de gerar soluções de alta qualidade em tarefas complexas.

O trabalho de MARQUES (2012), “Predição de Séries Temporais Utilizando Algoritmos Genéticos”, também contribuiu para nossa compreensão da aplicação dos algoritmos genéticos na previsão de séries temporais. A abordagem apresentada reforça a utilidade desses algoritmos para ajustar hiperparâmetros e melhorar a precisão dos modelos preditivos.

Esses estudos compartilham o objetivo comum de otimizar redes neurais utilizando diferentes abordagens e técnicas, oferecendo uma base valiosa para a comparação com nossos resultados. A investigação sobre a eficácia de diferentes algoritmos de otimização, como o GA e o PSO, e suas implicações na performance das redes neurais, é um tema central que conecta todos esses trabalhos com a nossa pesquisa.

1.8 Método de Pesquisa

Conforme (WAZLAWICK, 2009), em geral as monografias têm um capítulo ou seção designados como “Metodologia”. Entretanto, linguisticamente, seria mais correto afirmar que um trabalho científico tem um método de pesquisa e não uma metodologia. Apesar do uso corrente, a seção de "Metodologia"deve ser entendida como uma descrição do método de pesquisa adotado.

1.8.1 Método Experimental

Para o desenvolvimento deste trabalho, foi escolhido o Método Experimental. Segundo BIAGI (2009) este método é utilizado para comprovar e medir variações ou efeitos que se reproduzem em uma situação específica ao introduzir uma causa. Com base no método experimental, será considerado o desenho de "Experimento Puro", que é conhecido por alcançar um maior rigor e nível explicativo dos fenômenos em questão. Veremos mais detalhes sobre este método no próximo capítulo.

1.9 Estrutura do Trabalho

Esta pesquisa organiza-se pelas normas da Associação Brasileira de Normas Técnicas (ABNT) e está dividida em 5 capítulos e referências.

O Capítulo 1 - Introdução - contextualiza e introduz o problema de pesquisa, aponta a justificativa à problemática, fixa as hipóteses, estabelece o objetivo geral e específicos, descreve a motivação, apresenta os trabalhos relacionados e especifica o método científico que será utilizado no desenvolvimento.

O Capítulo 2 - Referencial Teórico - observando a literatura científica referente ao tema de pesquisa, buscando teorias úteis e aplicáveis, sendo a base sólida para construção da investigação em foco. Descreve especificamente a teoria usada para desenvolvimento e implementação dos experimentos

O Capítulo 3 - Desenvolvimento - Apresenta através de fluxogramas a estrutura gral dos experimentos, define os critérios para realização dos testes, descreve os materiais utilizados, detalha sobre as bases de dados escolhidas, mostra o funcionamento da função objetivo, que é o objeto de estudo do trabalho, explica como foi feita a implementação de cada um dos testes, define os dados que serão coletados e seus formatos, e por fim, apresenta como será feita a análise estatística, quais os testes e a ordem, para que em seguida possamos avaliar as hipóteses propostas.

O Capítulo 4 - Resultados e Discussões - este capítulo inicia com a tabela de resultados, mostrando as métricas de avalização obtidas em todos os experimentos, em seguida mostra os resultados dos testes estatísticos, juntamente com a validação das hipóteses propostas. É realizado também uma análise empírica dos resultados e das limitações do trabalho. Por fim, realizamos um comparativo dos resultados com outros trabalhos relacionados.

O Capítulo 5 - Conclusões - Faz a observação do cumprimento dos objetivos específicos e alcance do objeto geral, apresenta aprendizados pessoais e contribuições para o campo de pesquisa e sugestão de trabalhos futuros.

Em seguida, temos as Referências, que lista as literaturas utilizadas, fazendo referência a livros, artigos científicos e sites, sendo a base teórica para a execução da pesquisa.

Por último, temos nos apêndices, os pseudocódigos desenvolvidos e implementados nos experimentos.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção explora os conceitos fundamentais para a compreensão da otimização e do treinamento de redes neurais, com ênfase em técnicas meta-heurísticas como o Algoritmo Genético (GA), a Otimização por Enxame de Partículas (PSO) e uma abordagem híbrida que combina os dois algoritmos (HGAPSO). Também falaremos sobre redes neurais do tipo feedforward (FNN), com destaque para a variante Perceptron Multicamadas (MLP). Além disso, temos o desenho experimental que vamos seguir no capítulo de desenvolvimento. Veremos também os conceitos dos métodos estatísticos que serão utilizados, abordagens para o pré-processamento dos dados e por fim, o modelo de comparação que será utilizado nos resultados.

2.1 Desenho Experimental

Segundo BIAGI (2009) o métodos de Experimento "puro", são os que alcançam um melhor nível explicativo sobre os fenômenos que estão sendo estudados. Temos que, os experimentos devem ser iguais em tudo, exceto no ftor experimental, que para o caso deste trabalho será o uso de diferentes métodos de otimização.Para garantir uma avaliação abrangente e controlada dos métodos de otimização, a seguir, são apresentadas as principais características e limitações do desenho experimental:

1. **Estudo de Caso**: Serão realizados experimentos em 3 *datasets* distintos para verificar a consistência dos resultados em diferentes conjuntos de dados. Os *datasets* selecionados são: *Auto MPG Dataset*, *Energy Efficiency Dataset* e um *dataset* gerado artificialmente que chamaremos de *Rastrigin Benchmark*, pelo uso da função Rastrigin na sua construção. Cada *dataset* possui características específicas que influenciam o comportamento dos métodos de otimização e a performance das Redes Neurais.

2. **Critério de Execução**: O número de execuções da função objetivo, a qual veremos

com mais detalhes nas próximas seções, será limitado a 350 por teste. Este critério visa controlar o uso de hardware e o tempo de execução do experimento, garantindo uma utilização eficiente dos recursos e permitindo uma comparação justa entre os diferentes métodos de otimização.

3. **Experimentos**: Os testes serão mantidos iguais em todos os aspectos, exceto pelo fator experimental em questão, que é a técnica de otimização aplicada para ajustar dois dos parâmetros de arquitetura da MLP. As técnicas de otimização consideradas incluem um método aleatório (RAND) e os métodos meta-heurísticos, Algoritmo Genético (GA), Otimização por Enxame de Partículas (PSO) e a hibridização de GA e PSO (HGAPSO).

4. **Coleta de Dados**: As técnicas de otimização serão aplicadas para ajustar os parâmetros da arquitetura da MLP, limitando-se ao número de camadas ocultas e o número de neurônios por camada oculta. Os Resultados das abordagens experimentais serão coletados e comparados para avaliar a eficácia no desempenho da rede neural com base na métrica RMSE.

5. **Análise Estatística**: Após a realização dos experimentos e coleta dos resultados, será aplicada uma análise estatística para avaliar a aceitação ou rejeição das hipóteses propostas.

2.2 Redes Neurais Feedforward

Conforme GOODFELLOW (2016) Redes Neurais Feedforward (FNN) são um tipo fundamental de rede neural onde as conexões entre os neurônios não formam ciclos. Em uma FNN, as informações fluem apenas em uma direção: da camada de entrada para a camada de saída, passando pelas camadas ocultas. Isso contrasta com redes neurais recorrentes, onde as conexões podem formar ciclos e a informação pode fluir em ambas as direções. FNNs são amplamente utilizadas para tarefas de classificação e regressão devido à sua simplicidade e eficiência.

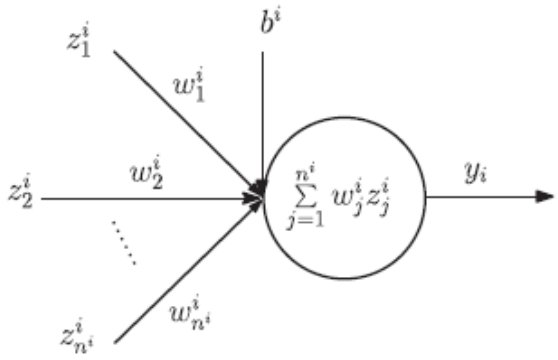
KUBAT (2017) e **?** nos apresentam mais sobre a estrutura das FNNs.

1. **Neurônios**

   Na Figura 3 vemos a estrutura de um neurônio, a unidade básica de uma MLP, que realiza uma operação simples: calcula uma soma ponderada dos sinais de entrada e aplica uma

função de ativação.

Figura 3 – Estrutura do Neurônio



Fonte: OJHA; ABRAHAM; SNÁŠEL (2017)

A função de ativação escolhida comumente para os MLPs é a função sigmoide, que é definida pela Equação 1:

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

Onde ξ é a soma ponderada das entradas. A função sigmoide tem a característica de limitar a saída ao intervalo $(0, 1)$, o que a torna útil para problemas de classificação. A função cresce monotonicamente com ξ e nunca ultrapassa o intervalo $(0, 1)$, com um ponto médio em $f(0) = 0.5$.

Outra função de ativação é a ReLU (Rectified Linear Unit) assim como a sigmoide, uma das funções de ativação mais amplamente utilizadas em redes neurais devido à sua simplicidade e eficiência computacional (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). A função ReLU é definida pela Equação 2:

$$\text{ReLU}(x) = \max(0, x) \tag{2}$$

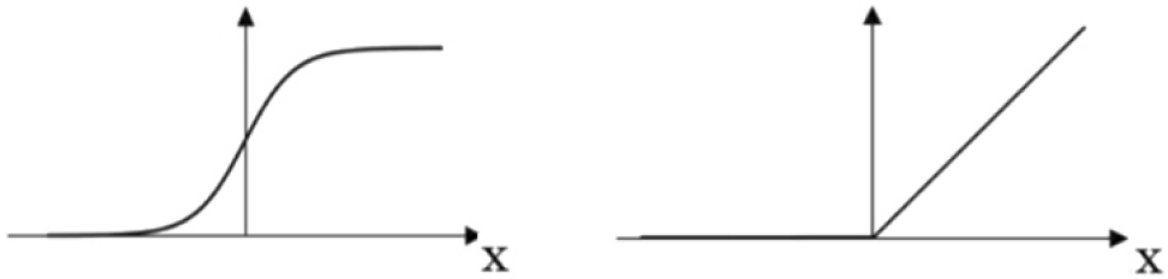
onde x é a entrada da função de ativação. Em outras palavras, a ReLU ativa a saída para valores positivos de x e é zero para valores negativos de x. A ReLU possui várias vantagens, incluindo:

- **Simplicidade Computacional:** A ReLU é simples de calcular e aplicar, tornando-a eficiente para treinamento e inferência.

- **Sparsity:** A ReLU produz uma saída esparsa (muitos valores são zero), o que pode ajudar a reduzir a complexidade do modelo e melhorar a eficiência computacional.
- **Mitigação do Problema do Gradiente Desvanecente:** A ReLU ajuda a mitigar o problema do gradiente desvanecente, que é comum em funções de ativação como a sigmoide e a tangente hiperbólica.

Figura 4 – Funções de Ativação.

(a) Função sigmoide. (b) Função Relu.



Fonte: Autor

2.2.0.1 Arquitetura das FNNs

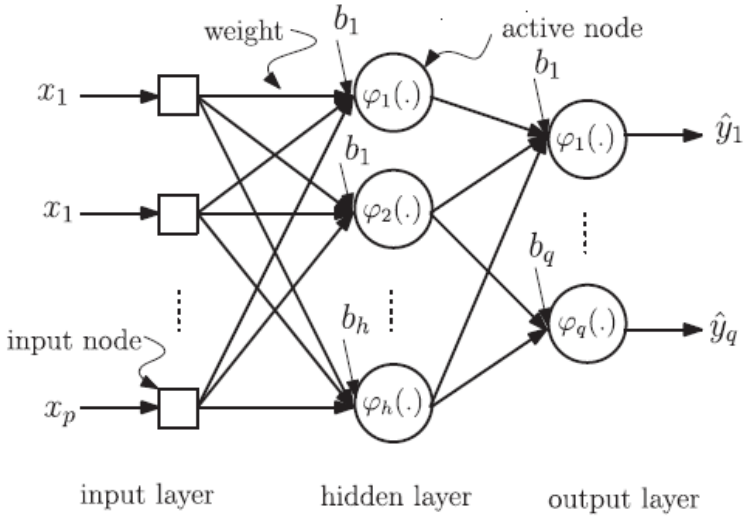
Uma FNN é composta por três tipos de camadas: camada de entrada, camada oculta e camada de saída.

- **Camada de Entrada:** Recebe as características do exemplo de entrada.
- **Camada Oculta:** Composta por neurônios que aplicam a função de ativação às somas ponderadas das entradas.
- **Camada de Saída:** Fornece as previsões finais da rede, com cada neurônio representando uma saída potencial.

2.2.1 Redes Neurais Perceptron Multicamadas

Segundo KUBAT (2017), Perceptron Multicamadas (MLP) é um tipo de FNN que inclui uma ou mais camadas ocultas entre a camada de entrada e a camada de saída. Cada camada oculta é composta por neurônios que aplicam funções de ativação não-lineares às combinações ponderadas das entradas. A MLP é capaz de aprender representações complexas e modelar superfícies de decisão não-lineares, o que a torna adequada para uma ampla gama de problemas

Figura 5 – Feedforward Neural Network.

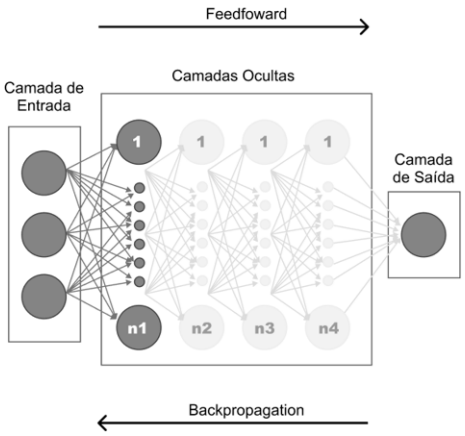


Fonte: OJHA; ABRAHAM; SNÁŠEL (2017)

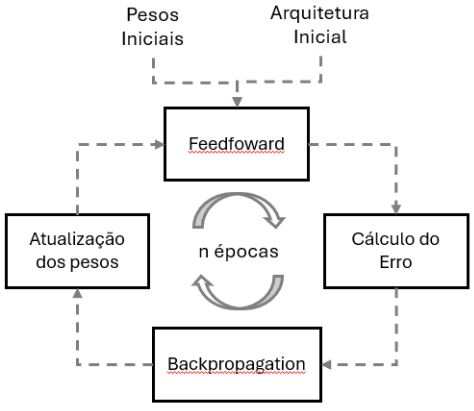
de aprendizado de máquina. A principal característica das MLPs é a capacidade de aprender a partir de dados, nos problemas de aprendizado supervisionado, ajustando os pesos das conexões durante o treinamento usando algoritmos como o *backpropagation*. Na Figura 6 podemos observar a estrutura de uma MLP. Na Figura 6b vemos o diagrama de blocos com as etapas do funcionamento da Rede Neural MLP.

Figura 6 – Redes Neurais Perceptron Multicamadas.

(a) Estrutura da Rede Neural MLP.



(b) Diagrama de Blocos da MLP.



Fonte: Autor

2.2.1.1 Propagação para Frente

Quando um exemplo é apresentado à rede, seus valores de atributos são propagados através da rede de entrada até a saída. Em cada camada, os valores são multiplicados pelos pesos associados às conexões, e a soma resultante é passada pela função de ativação. No caso de regressão, o neurônio da camada de saída gera um valor contínuo que representa a predição da rede para o exemplo dado (KUBAT, 2017). Por exemplo, se o valor de saída é $y = 3.5$, a rede estima que o valor da variável alvo para esse exemplo é 3.5.

A Equação 3 nos mostrar como calcular a saída de um neurônio da camada de saída em um modelo de regressão:

$$y = f\left(\sum_j w_{ji}^{(1)} f\left(\sum_k w_{kj}^{(2)} x_k\right)\right) \tag{3}$$

Onde $w_{kj}^{(2)}$ são os pesos da camada oculta e $w_{ji}^{(1)}$ são os pesos da camada de saída. O valor y resultante é a predição contínua do modelo para o exemplo de entrada.

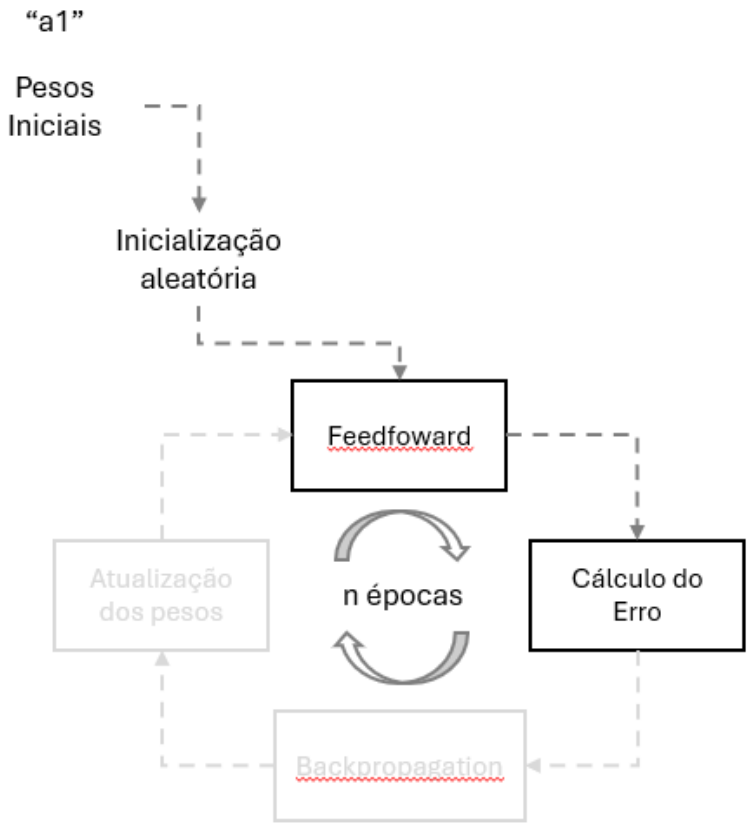
2.2.1.2 Teorema da Universalidade

Um resultado significativo na teoria das redes neurais é o Teorema da Universalidade, que afirma que uma MLP com a quantidade adequada de neurônios ocultos pode aproximar qualquer função contínua com precisão arbitrária. Isso sugere que, em teoria, MLPs podem ser utilizadas para resolver uma ampla gama de problemas de classificação e regressão. No entanto, o teorema não fornece uma solução prática para determinar o número exato de neurônios ocultos ou os valores dos pesos, o que significa que a tarefa de encontrar a configuração ideal permanece um desafio (KUBAT, 2017).

2.2.2 Erro da Rede Neural

Para avaliar o desempenho de um MLP, calculamos a taxa de erro e o erro quadrático médio (MSE). A taxa de erro é a proporção de exemplos classificados incorretamente. No entanto, para uma avaliação mais precisa, utilizamos o MSE, que leva em consideração a confiança das previsões da rede (KUBAT, 2017). Na Figura 7 temos no diagrama de blocos a etapa onde ocorre o cálculo do erro.

Figura 7 – Cálculo do Erro RMSE



Fonte: Autor

2.2.2.1 Erro Quadrático Médio (RMSE)

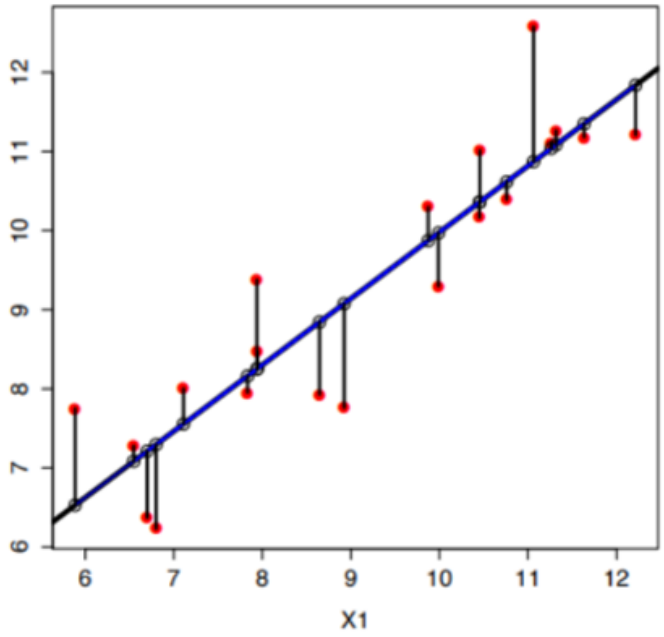
O Erro Quadrático Médio (RMSE) é utilizado para avaliar o desempenho da rede neural. O RMSE é a raiz quadrada do MSE e fornece uma medida da magnitude média dos erros (KUBAT, 2017). É definido pela Equação 4:

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{4}$$

O RMSE é útil para interpretar a precisão da rede, pois está na mesma unidade dos dados originais, facilitando a comparação com a escala dos valores previstos. Na Figura 8 temos o gráfico esperado para o erro da previsão da rede em comparação com os valores reais.

No perceptron multicamada, os parâmetros que afetam o comportamento da rede são os conjuntos de pesos, $w_{ji}^{(1)}$ e $w_{kj}^{(2)}$. A tarefa do aprendizado de máquina é encontrar valores para esses pesos que otimizem o desempenho de classificação da rede. Assim como no caso dos classificadores lineares, isso é alcançado através do treinamento. Esta seção é dedicada a uma

Figura 8 – Gráfico espera para o Erro RMSE



Fonte: Autor

técnica popular capaz de realizar essa tarefa.

2.2.3 Retropropagação do Erro

Segundo KUBAT (2017), para a retropropagação do erro, o *backpropagation* é uma técnica amplamente utilizada no treinamento de redes neurais do tipo perceptron multicamada (MLP). Essa técnica visa ajustar os pesos da rede para minimizar o erro na classificação dos exemplos de treinamento. Os pesos, que são os parâmetros da rede, determinam o comportamento da mesma, e o objetivo do aprendizado é encontrar valores ótimos para esses pesos.

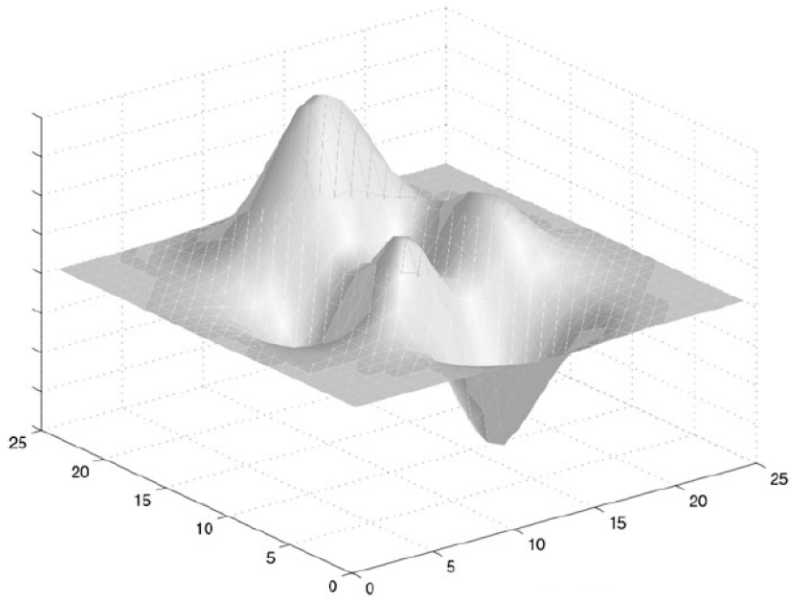
2.2.3.1 Inicialização e Treinamento

O processo de treinamento de uma MLP começa com a inicialização dos pesos com valores aleatórios pequenos, geralmente entre -0.1 e 0.1. Em seguida, os exemplos de treinamento são apresentados à rede um por vez. Cada exemplo é propagado para frente, desde a camada de entrada até a camada de saída, gerando uma previsão. A diferença entre a previsão da rede e o valor alvo do exemplo é usada para ajustar os pesos, de modo a reduzir o erro. Este processo é repetido para todos os exemplos de treinamento, completando uma época. No caso das MLPs, o número de épocas necessárias para um treinamento bem-sucedido pode ser de milhares ou até dezenas de milhares KUBAT (2017).

2.2.3.2 Descida do Gradiente

Para ajustar os pesos da rede, utilizamos o método *Gradient Descent*, onde buscamos encontrar o mínimo global da função de erro, representado graficamente como um "vale"em um "terreno"de erros, na Figura 9 temos uma ilustração. Cada par de pesos define uma localização específica neste "terreno", e qualquer alteração nos pesos resultará em uma nova posição, seja em uma subida ou descida do "terreno". O objetivo é realizar alterações nos pesos que resultem na maior redução possível do erro, seguindo o gradiente da função de erro.

Figura 9 – Representação gráfica da busca pelo mínimo global da função de erro.



Fonte: Adaptado de KUBAT (2017)

2.2.3.3 Propagação do Erro

O *backpropagation* ajusta os pesos de maneira eficiente ao calcular o gradiente da função de erro em relação a cada peso. Para isso, ele considera a contribuição de cada neurônio ao erro total. Os neurônios da camada de saída, que geram as previsões da rede, têm suas responsabilidades pelo erro calculadas pela Equação 5:

$$\delta_i^{(1)} = y_i(1 - y_i)(t_i - y_i) \tag{5}$$

Onde t_i é o valor alvo e y_i é a saída do neurônio. Este termo mede o quanto cada neurônio "concorda"ou "discorda"com a classificação correta. O sinal de $\delta_i^{(1)}$ depende de $t_i - y_i$, indicando se a correção deve aumentar ou diminuir o peso.

Para os neurônios da camada escondida, a responsabilidade pelo erro é propagada a partir da camada de saída, seguindo o princípio da retropropagação, Equação 6:

$$\delta_j^{(2)} = h_j(1 - h_j) \sum_i \delta_i^{(1)} w_{ji} \tag{6}$$

Aqui, a responsabilidade dos neurônios escondidos é calculada pela soma ponderada das responsabilidades dos neurônios da camada de saída, ajustada pela ativação dos neurônios escondidos.

2.2.3.4 Atualização dos Pesos

Com as responsabilidades dos neurônios conhecidas, os pesos da rede são atualizados utilizando uma regra aditiva:

- Para os neurônios da camada de saída: $w_{ji}^{(1)} = w_{ji}^{(1)} + \eta \delta_i^{(1)} h_j$
- Para os neurônios da camada escondida: $w_{kj}^{(2)} = w_{kj}^{(2)} + \eta \delta_j^{(2)} x_k$

Aqui, η é a *taxa de aprendizado*, que controla a magnitude das atualizações dos pesos. Pesos menores são mais afetados pelas atualizações, enquanto pesos maiores sofrem ajustes menores.

Esse processo se repete para todos os exemplos de treinamento, em várias épocas, até que o erro da rede seja minimizado ou atinja um valor aceitável, ou um critério de parada seja atingido. O *backpropagation* é uma técnica poderosa que permite às redes neurais ajustarem seus pesos de maneira eficiente, resultando em melhores desempenhos.

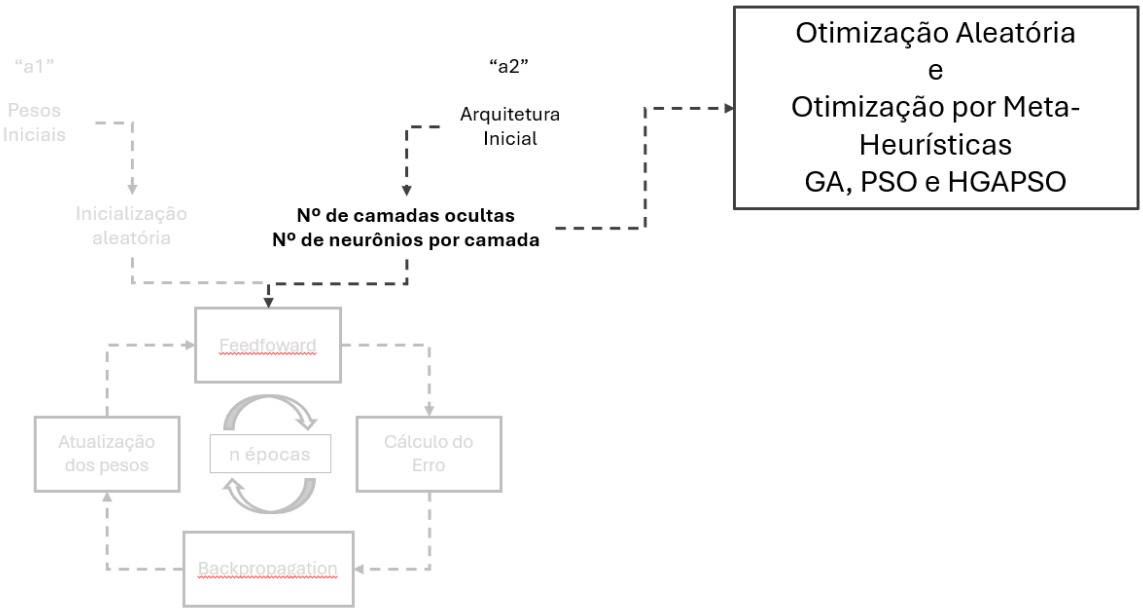
2.3 Otimização da Arquitetura de FNNs

(KUBAT, 2017) nos trás uma questão importante na construção de redes neurais, que é determinar o número adequado de neurônios ocultos. Se o número de neurônios ocultos for muito pequeno, a rede pode não ter flexibilidade suficiente para modelar superfícies de decisão complexas, o que pode resultar em um treinamento que fica preso em mínimos locais. Por outro lado, usar um número excessivo de neurônios ocultos aumenta os custos computacionais e pode levar a um ajuste excessivo dos dados, tornando a rede mais flexível do que o necessário. Portanto, é necessário encontrar um equilíbrio apropriado.

O texto de OJHA; ABRAHAM; SNÁŠEL (2017) discute a otimização de arquitetura e pesos em Redes Neurais Feedforward (FNN). A abordagem básica para otimização da arquitetura

é a *cascade correlation learning*, que adiciona iterativamente nós à camada oculta para construir a arquitetura ideal. Outros métodos incluem o construtivo (adicionar nós) e o destrutivo (podar nós), semelhantes ao método manual de tentativa e erro. Para otimizar a arquitetura de forma mais eficaz, pode-se usar uma representação genética da arquitetura da FNN, que permite buscar a arquitetura ideal dentro de um espaço compacto de topologias de FNN. Na Figura 10, temos a representação pelo diagrama de blocos, da etapa onde ocorre o processo de otimização da arquitetura da MLP.

Figura 10 – Representação da Otimização da área "a2", para arquitetura da MLP.



Fonte: Autor

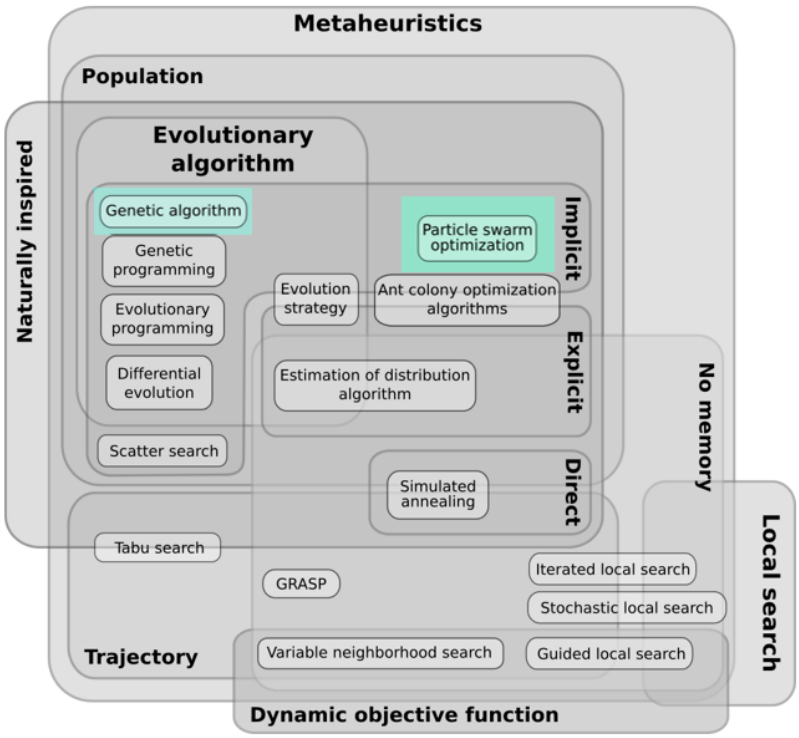
Segundo OJHA; ABRAHAM; SNÁŠEL (2017), vemos também que cada componente de uma Rede Neural Feedforward (FNN) pode ser otimizado separadamente, como pesos, arquitetura e função de ativação, ou todos juntos. A otimização pode ser feita mantendo certos componentes fixos enquanto ajusta outros. Alternativamente, pode-se otimizar todos os componentes simultaneamente usando uma representação vetorizada. Após projetar essa representação, um algoritmo meta-heurístico pode ser aplicado para encontrar a FNN ótima.

2.4 Métodos baseados em população

Segundo LUKE (2013), os métodos baseados em população diferenciam-se dos métodos tradicionais por manterem um conjunto de soluções candidatas em vez de uma única solução. Cada uma dessas soluções é avaliada e ajustada, mas o que distingue esses métodos de uma

simples busca local paralela é que as soluções candidatas influenciam a forma como outras soluções serão otimizadas na função de qualidade. Isso pode ocorrer através da rejeição de soluções fracas em favor de novas soluções ou pelo ajuste das soluções em direção às melhores opções. O Algoritmo Genético (GA) e o Particle Swarm Optimization (PSO) são dois exemplos notáveis de métodos baseados em população que se inspiram na biologia e em comportamentos naturais para resolver problemas complexos de otimização. Na Figura **??**, é apresentada uma representação geral das meta-heurísticas, destacando como elas se distribuem em suas respectivas classes. Nessa figura, também estão em evidência as posições ocupadas pelos algoritmos GA e PSO .

Figura 11 – Representação das classes de Meta-Heurísticas.



Fonte: Adaptado de Wikipedia

O GA se baseia em conceitos de genética e evolução, onde um conjunto de soluções candidatas, representadas como cromossomos, passa por operações semelhantes à seleção natural, cruzamento e mutação para gerar novas soluções em cada geração. A ideia é que, ao longo de várias gerações, a população evolua, produzindo soluções cada vez melhores.

Por outro lado, o PSO é uma técnica de otimização estocástica que, embora compartilhe algumas semelhanças com os algoritmos evolutivos, difere deles de maneira significativa. Em vez de se basear na evolução, o PSO é inspirado nos comportamentos de enxames e bandos

observados em animais. Ao contrário de outros métodos baseados em população, o PSO não gera novas populações, ele não envolve qualquer tipo de nova seleção. Em vez disso, o PSO mantém uma população estática cujos membros são ajustados em resposta a novas descobertas no espaço de busca, funcionando essencialmente como uma forma de mutação direcionada.

Ambos os métodos, GA e PSO, são amplamente utilizados em diversas áreas devido à sua capacidade de encontrar soluções de alta qualidade em problemas de otimização complexos.

2.4.1 Algoritmo Genético (GA)

O Algoritmo Genético (GA), desenvolvido por John Holland na década de 1970, é uma técnica de otimização inspirada na evolução natural. Semelhante às Estratégias Evolutivas, o GA também realiza a seleção, reprodução e substituição da população, mas difere na forma como gera novos indivíduos.

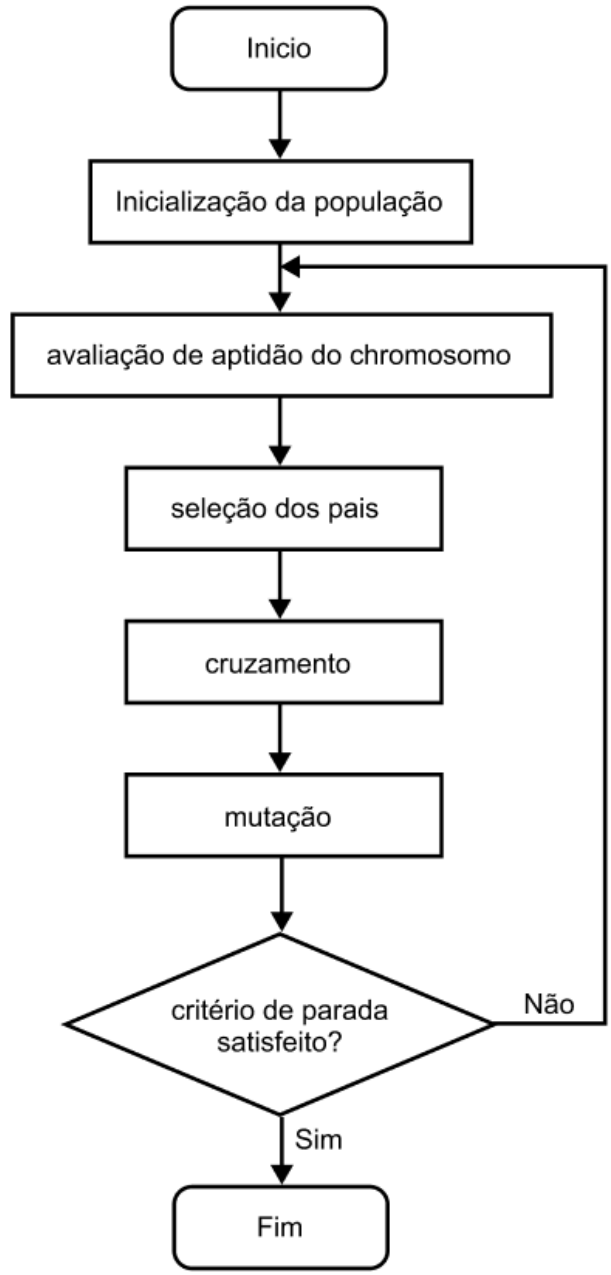
Em um GA, começamos com uma população inicial e, em seguida, selecionamos pares de pais por torneio, realizando assim, a aptidão entre os escolhidos. Esses pais são cruzados e mutados para gerar novos filhos, que são adicionados a uma população de filhos. Esse processo de seleção, cruzamento e mutação é repetido até que a população de filhos esteja completa (LUKE, 2013).

A seguir, detalharemos o funcionamento do Algoritmo Genético. A Figura 12 ilustra o funcionamento do GA.

1. **Inicialização da População**: Normalmente iniciada de forma aleatória, para este trabalho, apresentaremos a técnica de Latin Hypercube Sampling (LHS) foi introduzida por McKay, Conover e Beckman em 1979. Segundo MCKAY; BECKMAN; CONOVER (2000) o LHS é uma técnica de amostragem estratificada que divide o espaço de variáveis contínuas em intervalos iguais e seleciona amostras de cada intervalo de forma a garantir que todas as regiões do espaço de busca sejam representadas. Em vez de gerar amostras aleatórias em todo o espaço, o LHS garante que cada intervalo do espaço de busca seja amostrado, evitando lacunas ou sobreposições. As vantagens do Latin Hypercube Sampling são:

    - Cobertura Uniforme: O LHS garante uma cobertura mais uniforme do espaço de soluções comparado com a amostragem aleatória, reduzindo a probabilidade de regiões não serem exploradas;

Figura 12 – Fluxograma do funcionamento do GA.



Fonte: Adaptado de LAMBORA; GUPTA; CHOPRA (2019)

- Redução de Amostras Necessárias: Comparado com a amostragem aleatória pura, o LHS pode exigir menos amostras para cobrir o espaço de variáveis de forma eficaz;
- Eficiência: O LHS pode ser mais eficiente em termos de qualidade das amostras geradas, especialmente em espaços de alta dimensão, ao evitar a redundância e melhorar a diversidade das soluções iniciais.

A biblioteca pyDOE é uma ferramenta amplamente utilizada para design de experimentos

e amostragem em Python, particularmente em contextos de otimização e análise estatística. Dentre as principais funcionalidades está o Latin Hypercube Sampling (LHS), que permite gerar amostras uniformemente distribuídas ao longo do espaço de variáveis, ideal para inicialização de populações em algoritmos evolutivos ou para experimentos em que a cobertura uniforme do espaço é desejada WANG; DOWLING (2022).

2. **Seleção de Pais por Torneio**

   A Seleção por Torneio é uma técnica popular utilizada em algoritmos genéticos para selecionar indivíduos para a próxima geração. Esta abordagem se destaca por sua simplicidade e flexibilidade, além de não depender diretamente dos valores absolutos de fitness dos indivíduos, mas sim de seu desempenho relativo. A Seleção por Torneio segue um processo direto e eficiente, descrito no Algoritmo 1. Neste algoritmo, um número fixo de indivíduos (tamanho do torneio) é escolhido aleatoriamente da população. Entre esses indivíduos, o mais apto (com maior valor de fitness) é selecionado para a próxima fase do algoritmo genético.

Algorithm 1 Seleção por Torneio

$P \leftarrow$ população
$t \leftarrow$ tamanho do torneio, $t \geq 1$
$Melhor \leftarrow$ indivíduo escolhido aleatoriamente de P com reposição
for i de 3 até t **do**
  $Prximo \leftarrow$ indivíduo escolhido aleatoriamente de P com reposição
  if $Fitness(Prximo) > Fitness(Melhor)$ **then**
    $Melhor \leftarrow Prximo$
  end if
end for
return $Melhor$ =0

3. **Vantagens da Seleção por Torneio**

   - **Independência do Valor Absoluto de *Fitness*:** Ao contrário da Seleção Proporcional ao *Fitness*, que pode enfrentar problemas quando a diferença entre os valores de *fitness* é pequena, a Seleção por Torneio não depende dos valores absolutos, apenas da ordenação relativa. Isso ajuda a evitar a seleção aleatória quando todos os indivíduos têm valores de *fitness* semelhantes.

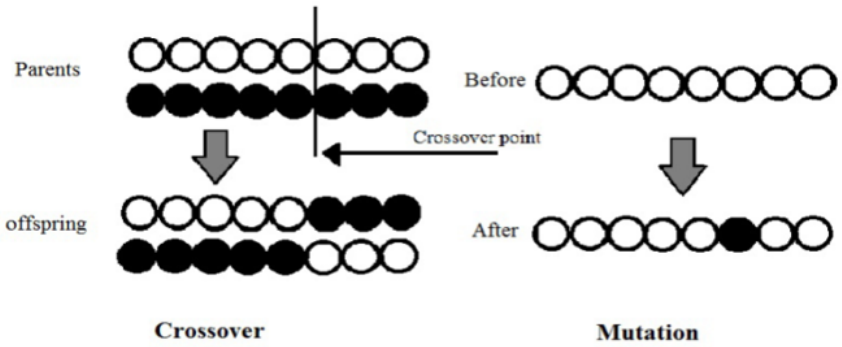
- **Simplicidade e Eficiência:** O algoritmo é fácil de implementar e não requer pré-processamento complexo. Além disso, é eficiente em termos de tempo de execução, o que é vantajoso em grandes populações.
- **Flexibilidade:** O tamanho do torneio t pode ser ajustado para controlar a seletividade do processo. Um tamanho pequeno (por exemplo, $t = 3$) faz com que a seleção seja menos seletiva e mais aleatória, enquanto um tamanho maior (por exemplo, $t > 7$) aumenta a probabilidade de selecionar o indivíduo mais apto da população.

A Seleção por Torneio é uma técnica amplamente adotada em algoritmos genéticos devido à sua eficácia e facilidade de ajuste, tornando-a uma escolha popular para muitas aplicações práticas em otimização e aprendizado de máquina (LUKE, 2013).

4. **Cruzamento e Mutação**

No contexto dos Algoritmos Genéticos, cruzamento e mutação são operações essenciais que permitem a evolução das soluções através da combinação e modificação dos indivíduos em uma população (LUKE, 2013). Na Figura 13 temos a representação dessas operações.

Figura 13 – Operações de Cruzamento e Mutação.



Fonte: Adaptado de LUKE (2013)

O *crossover* é uma operação fundamental no Algoritmo Genético que mistura partes de dois vetores pais para gerar novos indivíduos. Existem várias formas de realizar o

crossover, dependendo da representação dos indivíduos. As técnicas clássicas incluem *Crossover* de Um Ponto, *Crossover* de Dois Pontos e *Crossover* Uniforme.

5. **Crossover Uniforme** O *Crossover* Uniforme trata cada ponto do vetor individualmente e decide, com uma probabilidade p, se deve trocar o valor de cada posição entre dois vetores pais. O Algoritmo 2 é apresentado a seguir:

Algorithm 2 Crossover Uniforme

1: **Entrada:** Vetores $\tilde{v} = \langle v_1, v_2, \ldots, v_l \rangle$ e $\tilde{w} = \langle w_1, w_2, \ldots, w_l \rangle$
2: Probabilidade de troca p
3: **for** $i = 1$ **to** l **do**
4: **if** uma variável aleatória é menor ou igual a p **then**
5: Troque os valores de v_i e w_i
6: **end if**
7: **end for**
8: **Saída:** Vetores $\tilde{v}$ e $\tilde{w}$ modificados =0

No Crossover Uniforme, cada posição do vetor tem uma chance independente de ser trocada, o que permite uma maior diversidade de combinações entre os vetores pais. Embora o *crossover* e a mutação sejam métodos distintos, ambos são vitais para a diversidade e a capacidade de exploração dos Algoritmos Genéticos. O *crossover* combina informações de pais para gerar novos indivíduos, enquanto a mutação introduz variação aleatória para evitar a convergência prematura e explorar novas regiões do espaço de soluções.

6. **Mutação** A mutação é um dos operadores cruciais do algoritmo genético e tem um papel vital na preservação da diversidade genética da população, o que ajuda a evitar a convergência prematura para soluções sub ótimas. Uma taxa muito alta pode levar a uma busca aleatória, enquanto uma taxa muito baixa pode não fornecer diversidade suficiente. É importante ajustar essa taxa conforme o problema específico. Dependendo do problema, a mutação pode ser feita de diferentes maneiras, como alteração de valores contínuos ou binários, troca de genes em permutações, entre outros LUKE (2013).

7. **Atualização da População** Substitui parte ou toda a população antiga pela nova população gerada. Isso pode ser feito de várias maneiras, como substituição total, onde toda a população é substituída, ou substituição parcial, onde apenas alguns indivíduos são trocados.

8. **Condição de Parada** O algoritmo para quando uma condição de parada é atingida. Isso pode ser um número máximo de gerações, uma solução suficientemente boa ou a convergência da população.

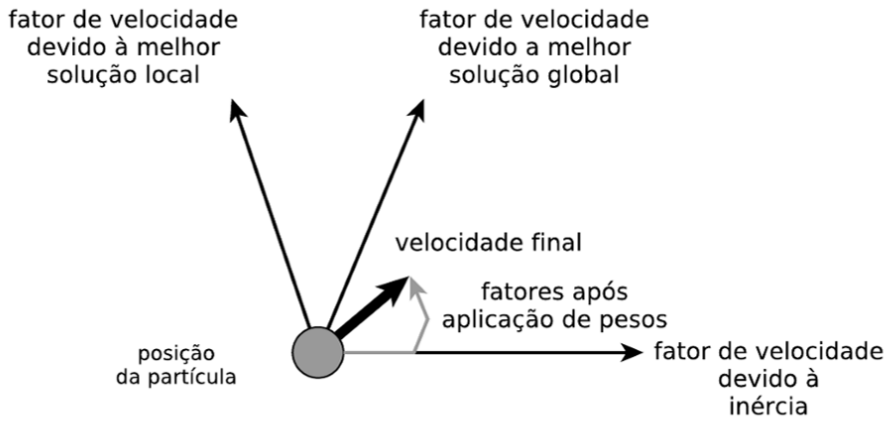
2.4.2 Otimização por Enxame de Partículas (PSO)

O algoritmo *Particle Swarm Optimization* (PSO) tem sido amplamente utilizado em diversas áreas, incluindo a otimização de redes neurais artificiais. De acordo com KENNEDY; EBERHART (1997), o PSO oferece algumas vantagens, tais como:

- fácil implementação computacional, exigindo poucas linhas de código;
- baixo consumo de memória e reduzida demanda de processamento;
- um processo de busca otimizado pelo aprendizado contínuo das partículas.

Introduzido em 1995, o PSO é uma técnica que se baseia no movimento coletivo de um grupo de partículas, denominado enxame. Cada partícula é movida no espaço de busca do problema por duas forças: uma que a atrai em direção à melhor posição que ela própria já encontrou (pbest) e outra que a direciona para a melhor posição encontrada por qualquer partícula do enxame (gbest). A cada iteração, a posição e a velocidade de cada partícula são atualizadas até que o enxame convirja para o melhor resultado KENNEDY; EBERHART (1997). Na Figura 14 apresentamos como esses vetores influenciam na otimização da partícula.

Figura 14 – Vetores de Otimização de Partículas no PSO.

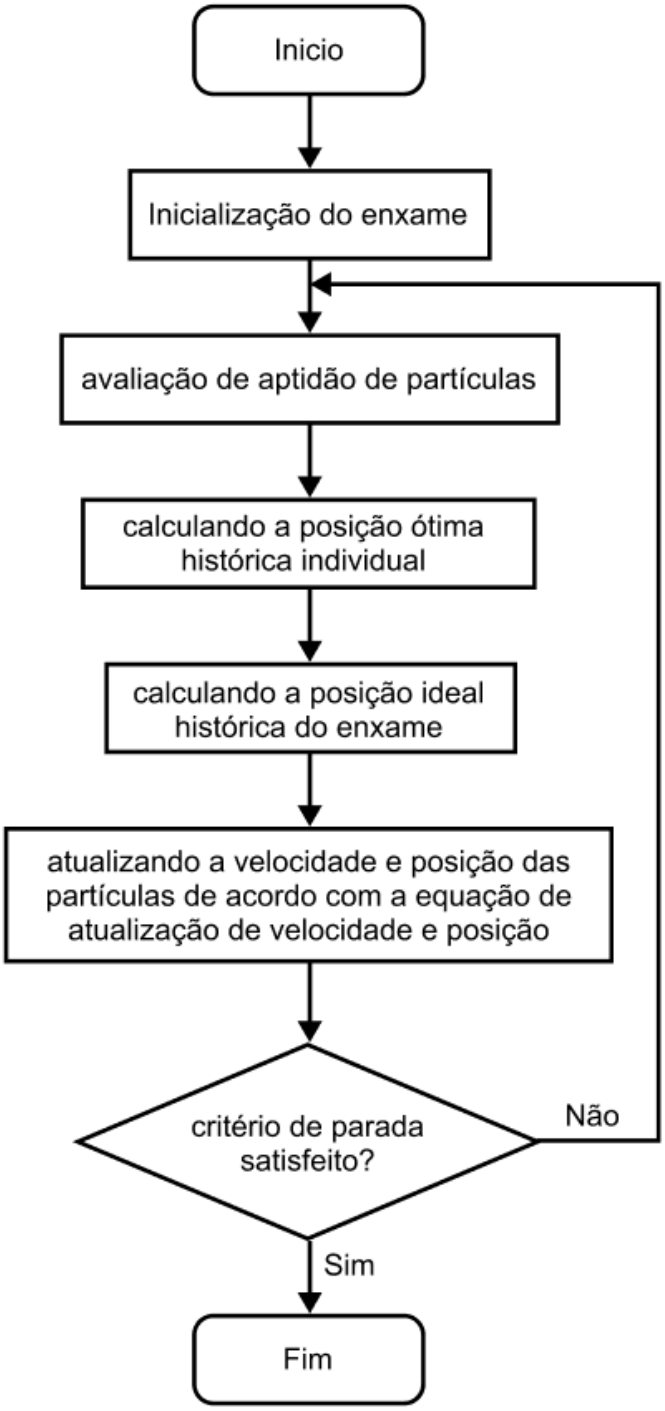


Fonte: WEBER (2023)

O PSO é um algoritmo de busca e otimização que simula o comportamento de um enxame de partículas em um espaço de busca para encontrar a solução ótima para um problema específico. Cada partícula representa uma solução candidata e ajusta sua posição com base em sua própria experiência e na experiência do grupo (**?**).

A Figura 15, ilustra o funcionamento do PSO e cada etapa será detalhada a seguir.

Figura 15 – Fluxograma do funcionamento do PSO.



Fonte: Adaptado de **?**

1. **Inicialização**: O algoritmo começa com a inicialização de um enxame de partículas. Cada partícula tem uma posição inicial e uma velocidade inicial aleatórias.

   - **Posição:** $\mathbf{x}_i^t$ representa a posição da partícula i no tempo t.
   - **Velocidade:** $\mathbf{v}_i^t$ representa a velocidade da partícula i no tempo t.

   Além disso, cada partícula mantém informações sobre a melhor posição encontrada por ela mesma ($\mathbf{p}_i^t$) e a melhor posição encontrada pelo enxame ($\mathbf{g}^t$).

2. **Atualização da Velocidade**: A velocidade de cada partícula é atualizada com base em três componentes:

   - A inércia da velocidade anterior.
   - A atração pela melhor posição pessoal.
   - A atração pela melhor posição global.

   Para atualizar a velocidade da partícula temos a Equação 7:

$$\mathbf{v}_i^{t+1} = \omega \cdot \mathbf{v}_i^t + c_1 \cdot r_1 \cdot (\mathbf{p}_i^t - \mathbf{x}_i^t) + c_2 \cdot r_2 \cdot (\mathbf{g}^t - \mathbf{x}_i^t) \tag{7}$$

   onde:

   - $\mathbf{v}_i^t$ é a velocidade da partícula i no tempo t.
   - $\mathbf{x}_i^t$ é a posição da partícula i no tempo t.
   - $\mathbf{p}_i^t$ é a melhor posição pessoal da partícula i até o tempo t.
   - $\mathbf{g}^t$ é a melhor posição global encontrada pelo enxame até o tempo t.
   - ω é o coeficiente de inércia, controlando a influência da velocidade anterior.
   - c_1 e c_2 são coeficientes de aceleração que determinam a influência das melhores posições pessoais e globais.
   - r_1 e r_2 são números aleatórios uniformemente distribuídos entre 0 e 1.

3. **Atualização da Posição**: A posição de cada partícula é atualizada adicionando a nova velocidade, Equação 8:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{8}$$

4. **Critério de Parada**

   O algoritmo PSO é executado até que um critério de parada seja atendido. Esse critério pode ser um número máximo de iterações, uma melhoria mínima na solução encontrada, ou uma convergência em torno de um ótimo.

2.4.3 Abordagem Híbrida (HGAPSO)

Segundo OJHA; ABRAHAM; SNÁŠEL (2017) os algoritmos meta-heurísticos têm proporcionado uma nova dimensão para a otimização das FNNs. Eles permitem simplificar a arquitetura da rede e otimizar vários componentes simultaneamente, o que é essencial para a generalização da rede. A capacidade de evoluir tanto a estrutura quanto os parâmetros da FNN facilita a obtenção de uma arquitetura mais geral e adaptada ao problema em questão.

Uma abordagem híbrida de algoritmos meta-heurístico foi aplicado com sucesso em problemas como o design de redes neurais recorrentes totalmente conectadas (*Fully Connected Recurrent Neural Networks* - FCRNN) e redes nebulosas recorrentes do tipo Takagi-Sugeno-Kang (TRFN) por JUANG (2004). Em testes de produção de sequência temporal e controle dinâmico de plantas, o HGAPSO mostrou superioridade em comparação com o GA e o PSO isoladamente, destacando-se como uma poderosa ferramenta de otimização para redes complexas que envolvem dados temporais e dinâmicos.

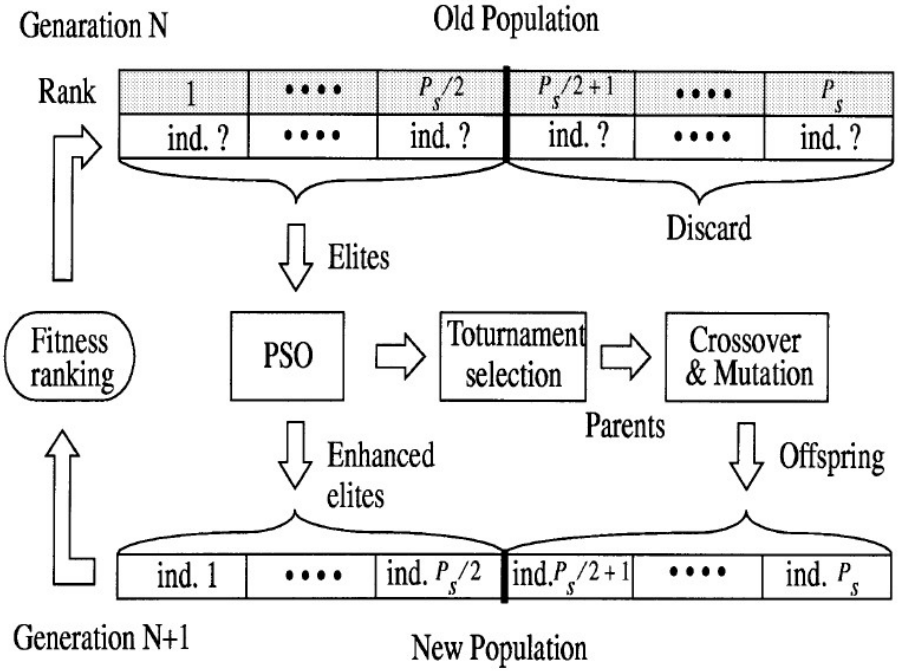
Hybrid Genetic Algorithm and Particle Swarm Optimization - HGAPSO - é um algoritmo de aprendizado evolutivo que combina elementos do Algoritmo Genético (GA) com a Otimização por Enxame de Partículas (PSO) para projetar redes neurais. O objetivo principal do HGAPSO é aproveitar as forças complementares de ambos os métodos para melhorar a eficiência e a precisão do processo de otimização(JUANG, 2004).

Na Figura 16 vemos a representação do algoritmo híbrido e como é feita a sequencia de combinações entre os algoritmos GA e PSO.

1. **População Inicial:** O algoritmo começa com uma população inicial aleatoriamente gerada, denotada como W, composta por m indivíduos. Cada indivíduo nessa população é tratado de forma dual: como um cromossomo no contexto do GA e como uma partícula no contexto do PSO.

2. **Cálculo da Aptidão:** Em cada geração, tanto o GA quanto o PSO avaliam a aptidão (*fitness*) de cada indivíduo. A função de aptidão mede o quão bem um indivíduo resolve o problema de otimização.

Figura 16 – Fluxograma do funcionamento do HGAPSO.



Fonte: (JUANG, 2004)

3. **Seleção dos Elites:** Após o cálculo da aptidão, os indivíduos com melhor desempenho (a metade superior da população) são selecionados como elites. Esses elites representam as melhores soluções encontradas até o momento.

4. **Cópia dos Elites:** Os indivíduos elitizados são copiados para a próxima geração, garantindo que as boas soluções sejam preservadas e aprimoradas. No entanto, ao contrário do GA tradicional, onde os elites são diretamente propagados, aqui ocorre uma divisão.

5. **Aprimoramento com GA e PSO:** A metade dos elites copiados é otimizada usando PSO, enquanto a outra metade é otimizada usando GA.

    - *Otimização com PSO:* Os elites selecionados para PSO atualizam suas posições (pesos da rede neural) com base em suas melhores posições pessoais e na melhor posição global do enxame.
    - *Otimização com GA:* Os elites selecionados para GA passam por operações de seleção por torneio, onde os melhores indivíduos são escolhidos para reprodução. Em seguida, ocorrem operações de cruzamento (*crossover*) e mutação para gerar novos indivíduos.

6. **Iteração:** Esse processo de avaliação, seleção de elites e otimização com GA e PSO é repetido por várias gerações. A cada geração, a população tende a melhorar em termos de aptidão.

2.4.3.1 Benefícios do Algoritmo Híbrido

- **Exploração Balanceada:** O GA é eficaz em explorar novas áreas do espaço de busca através de cruzamento e mutação, enquanto o PSO é bom em explorar localmente em torno de boas soluções. A combinação dos dois permite um equilíbrio entre a exploração e a exploração, melhorando a qualidade das soluções encontradas.

- **Preservação e Aprimoramento de Boas Soluções:** A estratégia de elitismo combinada com o aprimoramento pelos dois métodos garante que as boas soluções sejam preservadas e melhoradas a cada geração, aumentando a probabilidade de convergência para uma solução ótima.

- **Versatilidade:** Essa abordagem pode ser aplicada a diversos problemas de otimização de redes neurais, oferecendo uma solução robusta para o ajuste de pesos e arquitetura de redes complexas.

Segundo JUANG (2004), esse método pode ser usado para otimizar Redes Neurais Feed-forward em tarefas como classificação, regressão, e outras aplicações de aprendizado supervisionado, onde a combinação de GA e PSO ajuda a encontrar uma configuração de rede que minimiza o erro de predição ou maximiza a precisão do modelo.

2.5 Definição dos Estudos de Caso

Na escolha das bases de dados para a aplicação de testes de comparação de algoritmos, é essencial considerarmos uma série de requisitos para garantir a validade e a relevância dos experimentos. Segundo (MANN; WHITNEY, 1947) podemos citar como requisitos:

- Número de Amostras: As bases de dados devem conter um número adequado de amostras para que os testes sejam representativos e as análises estatísticas sejam confiáveis. No contexto deste estudo, foi definido um limite máximo de 1000 amostras para equilibrar a carga computacional e a complexidade dos cálculos, garantindo que os experimentos possam ser realizados de maneira eficiente e prática.

- Não-Linearidade: É importante que as bases de dados apresentem relações não lineares entre as variáveis de entrada e saída. Isso permite avaliar a capacidade dos algoritmos de lidar com a complexidade dos dados e identificar como eles se comportam em cenários onde as relações não são simplesmente lineares, o que é comum em muitos problemas reais.

- Complexidade: As bases de dados devem refletir uma variedade de complexidades, desde problemas com dados reais e aplicáveis até funções teóricas com alta complexidade. Isso garante que os algoritmos sejam testados em diferentes níveis de dificuldade e possam demonstrar sua eficácia em uma gama ampla de cenários.

- Diversidade de Domínios: A escolha de bases de dados de diferentes áreas e contextos ajuda a assegurar que os algoritmos sejam avaliados em uma variedade de situações e tipos de problemas. Isso inclui dados do mundo real, como consumo de combustível e eficiência energética, bem como funções matemáticas complexas como a função Rastrigin.

2.6 Pré-Processamento dos Dados

2.6.1 Codificação de Variáveis Categóricas

A codificação de variáveis categóricas é um processo essencial de pré-processamento de dados que transforma variáveis com valores de categorias distintas em um formato numérico que pode ser usado por algoritmos de aprendizado de máquina. Muitas vezes, algoritmos de aprendizado de máquina não conseguem lidar diretamente com dados categóricos, pois eles exigem entradas numéricas. A técnica mais comum para essa transformação é a codificação *one-hot*.

A *codificação one-hot* converte variáveis categóricas em uma série de colunas binárias, onde cada coluna representa uma categoria única da variável original. Aqui está um passo a passo do processo:

- **Identificação das Categorias Únicas:** Primeiro, identificam-se todas as categorias únicas presentes na variável categórica.

- **Criação de Colunas Dummy:** Para cada categoria única, cria-se uma nova coluna binária (*dummy*).

- **Atribuição de Valores Binários:** Para cada linha no *dataset*, a coluna correspondente à categoria da observação recebe o valor 1, enquanto as outras colunas recebem 0.

A codificação *one-hot* preserva a informação das categorias e é adequada para algoritmos que não podem lidar com variáveis categóricas diretamente.

Essa técnica pode aumentar a dimensionalidade do *dataset*, especialmente se a variável categórica tiver muitas categorias, resultando em uma matriz esparsa (muitas entradas são 0). Além disso, é importante aplicar a mesma transformação aos dados de teste para garantir a consistência.

A codificação de variáveis categóricas é, portanto, uma etapa fundamental para preparar dados para análise, permitindo que modelos de aprendizado de máquina trabalhem com dados que originalmente não estariam em formato numérico.

2.6.2 Normalização

A normalização é uma técnica essencial de pré-processamento que ajusta os valores das características para uma escala comum. Isso é crucial para garantir que todas as variáveis tenham a mesma influência durante o treinamento do modelo, evitando que características com escalas maiores dominem o processo de aprendizado. O objetivo da normalização é transformar os dados para que todas as características estejam dentro de um intervalo específico, geralmente entre 0 e 1. Isso assegura que todas as variáveis contribuam igualmente para o modelo, independentemente de suas escalas originais. Um método comum de normalização é o *Min-Max Scaling*. Para normalizar um valor x temos a Equação 9:

$$x_{norm} = \frac{x - \min(X)}{\max(X) - \min(X)} \tag{9}$$

onde:

- x é o valor original da característica.
- $\min(X)$ é o valor mínimo da característica.
- $\max(X)$ é o valor máximo da característica.
- x_{norm} é o valor normalizado.

A normalização dos dados de entrada oferece alguns benefícios:

- **Consistência:** Garante que todas as variáveis estejam na mesma escala, o que é especialmente importante para algoritmos sensíveis à escala, como redes neurais.
- **Convergência Rápida:** Melhora a velocidade e a eficiência do treinamento, pois os algoritmos convergem mais rapidamente quando as variáveis estão na mesma escala.
- **Estabilidade Numérica:** Reduz o risco de problemas de precisão numérica que podem ocorrer devido a grandes variações na escala das variáveis.

Ao aplicar a normalização:

- A normalização deve ser feita nos dados de treinamento e os parâmetros calculados (mínimo e máximo) devem ser usados para normalizar também os dados de teste e novos dados durante a previsão.
- É importante observar que valores fora do intervalo original após a normalização podem afetar a performance do modelo, pois a normalização não lida com valores extremos.

A normalização é, portanto, um passo fundamental para preparar dados para análise e modelagem, garantindo que todas as características contribuam de maneira equilibrada e melhorando a eficiência do processo de aprendizado de máquina.

2.7 Análise Estatística

A análise estatística é fundamental em pesquisas científicas, pois permite a interpretação precisa dos dados coletados, auxiliando na identificação de padrões, tendências e relações entre variáveis. Esse processo é essencial para validar hipóteses e garantir a confiabilidade dos resultados, o que é crucial para a tomada de decisões baseadas em evidências. Além disso, a análise estatística oferece ferramentas para lidar com incertezas e variabilidades inerentes aos dados, proporcionando uma base sólida para conclusões robustas e generalizáveis. Como destacado por ALTMAN (1990), a falta de uma análise estatística adequada pode comprometer significativamente a qualidade e a credibilidade das conclusões científicas.

2.7.1 Teste de Shapiro-Wilk:

O teste de Shapiro-Wilk segundo (SHAPIRO; WILK, 1965) é utilizado para verificar a normalidade dos dados e determinar se uma amostra segue uma distribuição normal. As hipóteses do teste são:

- **Hipótese Nula (H_0):** Os dados seguem uma distribuição normal.
- **Hipótese Alternativa (H_1):** Os dados não seguem uma distribuição normal.

O teste fornece dois principais valores:

- **Estatística de Teste:** Medida de quão bem os dados se ajustam a uma distribuição normal. Quanto mais próximo de 1 for o valor da estatística, melhor os dados se ajustam a uma distribuição normal.
- **p-valor:** Valor que indica a probabilidade de observar os dados se a hipótese nula for verdadeira. Um p-valor baixo (geralmente < 0,05) sugere que os dados não seguem uma distribuição normal, levando à rejeição da hipótese nula.

2.7.2 Teste de Kruskal-Wallis:

O teste de Kruskal-Wallis é um teste não paramétrico utilizado para comparar as medianas de mais de dois grupos independentes (KRUSKAL; WALLIS, 1952). As hipóteses do teste são:

- **Hipótese Nula (H_0):** As medianas dos grupos são iguais.
- **Hipótese Alternativa (H_1):** Pelo menos uma das medianas dos grupos é diferente.

O resultado inclui:

- **Estatística H:** Valor do teste que reflete a variabilidade das medianas entre os grupos, calculado comparando a variabilidade entre grupos com a variabilidade dentro dos grupos.
- **p-valor:** Valor que indica se as diferenças observadas entre as medianas são estatisticamente significativas. Um p-valor baixo sugere que pelo menos um dos grupos é significativamente diferente dos outros.

2.7.3 Testes U de Mann-Whitney:

Os Testes U de Mann-Whitney são usados para comparar duas amostras independentes e determinar se há diferenças significativas entre suas medianas MANN; WHITNEY (1947). As hipóteses do teste são:

- **Hipótese Nula (H_0):** As duas amostras têm distribuições com a mesma mediana.
- **Hipótese Alternativa (H_1):** As duas amostras têm distribuições com medianas diferentes.

O teste fornece:

- **Estatística U:** Medida que reflete a diferença entre as distribuições dos dois grupos, calculada com base nas classificações dos dados.
- **p-valor:** Valor que indica a significância das diferenças observadas entre as duas amostras. Um p-valor baixo sugere que as medianas dos dois grupos são significativamente diferentes.

2.7.4 Teste ANOVA (Análise de Variância):

O teste ANOVA é usado para comparar as médias de três ou mais grupos independentes e verificar se há diferenças significativas entre eles TABACHNICK; FIDELL (2007). As hipóteses do teste são:

- **Hipótese Nula (H_0):** As médias dos grupos são iguais.
- **Hipótese Alternativa (H_1):** Pelo menos uma das médias dos grupos é diferente.

O resultado inclui:

- **Estatística F:** Valor calculado que reflete a relação entre a variabilidade entre os grupos e a variabilidade dentro dos grupos. Uma Estatística F elevada sugere diferenças significativas entre as médias dos grupos.
- **p-valor:** Valor que indica se a Estatística F é significativa. Um p-valor baixo sugere que há diferenças estatisticamente significativas entre as médias dos grupos.

2.7.5 Teste de Tukey:

O Teste de Tukey segundo TUKEY (1953), também conhecido como Teste de Tukey para Comparações Múltiplas, é um teste post hoc utilizado após a realização de um teste ANOVA para realizar comparações múltiplas entre todos os pares de grupos. As hipóteses do teste são:

- **Hipótese Nula (H_0):** As médias de todos os pares de grupos comparados são iguais.

- **Hipótese Alternativa (H_1):** Pelo menos um dos pares de grupos comparados tem médias diferentes.

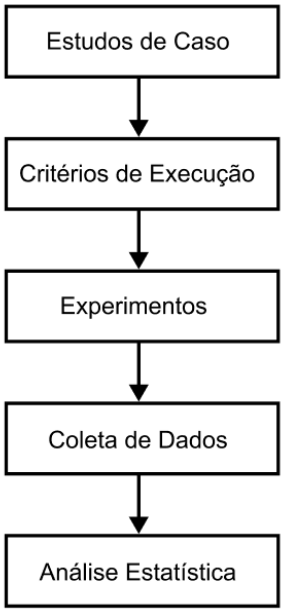
O resultado inclui:

- **Estatística Q (ou Estatística de Tukey):** Medida que reflete a diferença entre as médias de pares de grupos ajustada para o erro padrão das médias das amostras.
- **p-valor ajustado:** Valor que indica a significância das diferenças entre os pares de grupos, ajustado para múltiplas comparações. Um p-valor baixo indica que a diferença entre os grupos é significativa após ajuste.

3 DESENVOLVIMENTO

Neste capítulo, desenvolveremos os passos do método experimental visto no capitulo anterior. O objetivo principal é aplicar diferentes técnicas de otimização da arquitetura, quanto ao número de camadas ocultas e número de neurônios por camada, para a construção de uma rede neural multicamada (MLP), na solução de diferentes problemas de regressão. Na Figura 17 podemos ver uma representação do processo.

Figura 17 – Método Experimental



Fonte: Autor

3.1 Materiais

Para a realização dos experimentos, foram utilizados os seguintes recursos de hardware e software:

3.1.1 Hardware

- **Sistema Operacional:** Microsoft Windows 11 Enterprise
- **Fabricante do Sistema:** LENOVO
- **Tipo de Sistema:** x64-based PC
- **SKU do Sistema:** LENOVO_MT_20S1_BU_Think_FM_ThinkPad T14 Gen 1
- **Processador:** Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz, 2304 Mhz, 4 Core(s), 8 Logical Processor(s)
- **Memória Física Total:** 15,8 GB

3.1.2 Software

- **Ambiente de Desenvolvimento:** Spyder IDE 5.5.1 (conda)
- **Python:** 3.11.7 64-bit

3.2 Estudos de Caso

Nesta seção, são apresentados os conjuntos de dados utilizados nos experimentos, incluindo a descrição de cada um, complexidade e justificativa de escolha. Vemos na Tabela 1, um resumo das principais informações dos *datasets* escolhidos para fins de comparação.

Tabela 1 – Exemplo de tabela com 3 colunas e 4 linhas

Nome	Complexidade	Amostras	Entradas
Auto MPG	Baixa	398	7
Energy Efficiency	Média	768	8
Rastrigin Benchmark	Alta	400	4

Fonte: Autor

3.2.1 Auto MPG Dataset

- **Descrição:** O Auto MPG Dataset de QUINLAN (1993) contém dados sobre a eficiência de combustível de automóveis, incluindo variáveis como o número de cilindros, a potência do motor e o peso do veículo. Este dataset é comumente utilizado para problemas de

regressão devido à sua simplicidade e à sua capacidade de fornecer uma base sólida para experimentos iniciais.

- **Justificativa da Escolha:** A escolha deste dataset permite avaliar a capacidade dos algoritmos de otimização em um cenário menos complexo, servindo como um ponto de partida para testar a eficácia dos métodos de otimização na melhoria da precisão dos modelos de regressão.

3.2.2 Energy Efficiency Dataset

- **Descrição:** O Energy Efficiency Dataset de TSANAS; XIFARA (2012), contém dados sobre a eficiência energética de edifícios, com variáveis relacionadas às características dos edifícios e seus consumos de energia. As variáveis categóricas são convertidas em variáveis dummy para o processamento. Este dataset é mais complexo do que o Auto MPG Dataset e é útil para testar a capacidade dos algoritmos de otimização em cenários de média complexidade.

- **Justificativa da Escolha:** A escolha deste dataset permite avaliar o desempenho dos algoritmos em um problema de regressão com maior complexidade e variabilidade, ajudando a verificar a robustez e a eficácia das técnicas de otimização em condições mais desafiadoras.

3.2.3 Rastrigin Benchmark

- **Descrição:** A função de Rastrigin é amplamente utilizada como benchmark em problemas de otimização, conhecida por sua complexidade devido à presença de muitos mínimos locais. Os dados são gerados aleatoriamente e a função é usada para criar a variável alvo com adição de ruído. Este *dataset* é utilizado para avaliar a capacidade dos algoritmos de otimização em um cenário desafiador com muitas armadilhas locais. Foi adicionado um ruido de 0.01 nas entradas do *dataset*. No Apendice temos o Algoritmo 3 que mostra como foi realizada a implementação.

- **Justificativa da Escolha:** A escolha da função de Rastrigin permite testar a eficácia dos algoritmos de otimização em problemas de alta complexidade e verificar se eles são capazes de encontrar soluções próximas ao ótimo global em meio a muitos mínimos locais.

No Apêndice A temos o Algoritmo 3 com mais detalhes sobre a implementação.

3.2.4 Pré-processamento

Para a base de dados Energy Efficiency Dataset, única que contém dados categóricos foi aplicada a Codificação de Variáveis Categóricas para as entradas "X6"e "X8", conforme o processo visto no capitulo anterior.

Para as bases de dados Auto MPG e Energy Efficiency Dataset tambem foi aplicada uma normalização dos dados de entrada, através do método Min-Max Scaling.

3.3 Critérios de Execução

Para garantir a eficácia e a equidade na comparação dos algoritmos propostos, algumas definições e premissas foram estabelecidas:

- **Número de Execuções da Função Objetivo**: Para garantir uma comparação equitativa entre os algoritmos, o número de execuções da função objetivo será limitado a 328 por teste, com uma margem de ±3%. A função objetivo desempenha o papel crucial de validar as soluções e aprimorar os resultados conforme a técnica de otimização aplicada. Esse limite de execuções será ajustado para cada algoritmo, assegurando que todos operem sob as mesmas condições de comparação. A escolha desse critério visa manter o tempo de execução de cada algoritmo em aproximadamente 30 minutos, considerando o tempo disponível para a coleta de resultados e os recursos computacionais disponíveis. Dada a arquitetura específica de cada algoritmo, essa margem de variação no número de execuções é considerada aceitável para manter a consistência nos resultados.

- **Modelo**: O MLPRegressor, da biblioteca scikit-learn, foi escolhido por ser um modelo de rede neural de múltiplas camadas (Multilayer Perceptron) amplamente utilizado para problemas de regressão, que aprende mapeamentos não lineares entre entradas e saídas por meio de treinamento supervisionado, utilizando retropropagação para ajustar os pesos da rede. Implementado na linguagem Python.

- **Treinamento**: O treinamento escolhido para ser realizado pela MLP foi o de validação cruzada, ou *kfold*, onde a performance do modelo é avaliada calculando a média das métricas de desempenho, erro quadrático médio, obtidas em cada uma das iterações. Não foram realizados testes para validação em dados que não participaram do treinamento. Este é um ponto importante e será explicado com mais detalhes na seção *Limitações do*

Trabalho.

- **Fator de Aleatoriedade**: Na construção dos algoritmos, não foi utilizado nenhum fator como semente para replicar os valores aleatórios. Isso permite que cada uma das técnicas mantenha suas características de aleatoriedade, potencializando assim a capacidade de generalização dos modelos.

Essas definições e premissas são fundamentais para garantir que os experimentos sejam realizados de maneira consistente e que os resultados obtidos sejam comparáveis.

3.4 Experimentos

O processo de desenvolvimento é dividido nas seguintes etapas:

1. Implementação da MLP: Uma MLP é configurada para receber como entrada, parâmetros que definam a sua arquitetura, quanto ao número de camadas ocultas e número de neurônios por camada.

2. Implementação da MLP com otimizador RAND: No primeiro experimento, é implementada a MLP com parâmetros de arquitetura de rede otimizados de forma aleatória por uma função geradora de números aleatórios entre 0 e 1. Dois valores são gerados para representar o número de camadas ocultas e o número de neurônios por camada oculta. Esta etapa estabelece a base para a comparação entre as diferentes técnicas de otimização.

3. Implementação da MLP com otimizador GA: No segundo experimento, é implementada a MLP com parâmetros de arquitetura de rede otimizados pelo algoritmo GA.

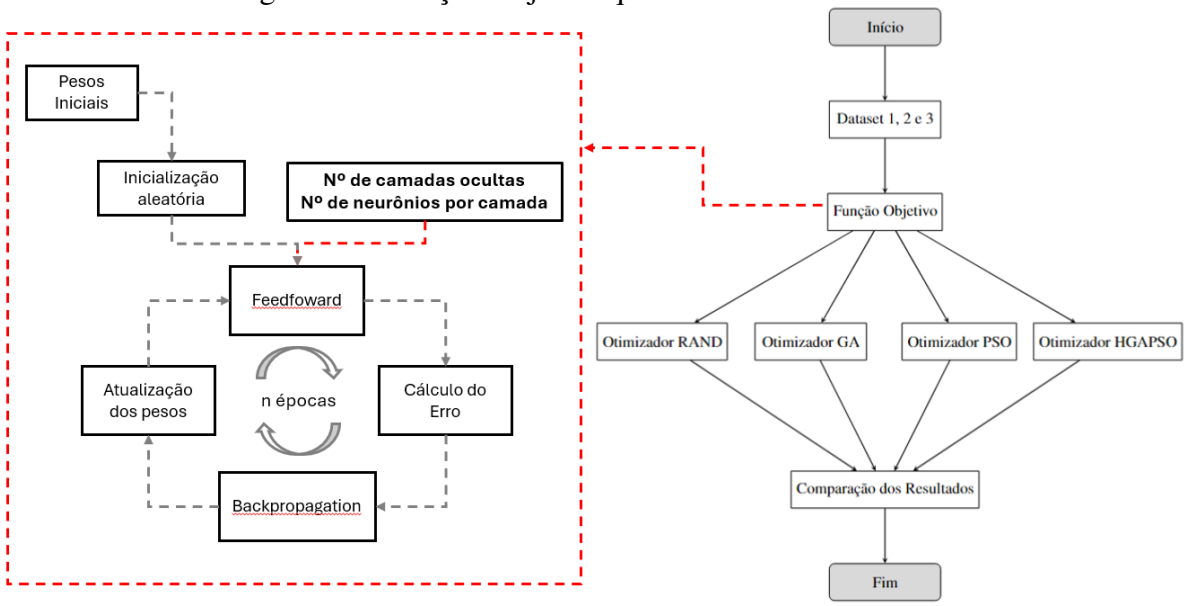
4. Implementação da MLP com otimizador PSO: No terceiro experimento, é implementada a MLP com parâmetros de arquitetura de rede otimizados pelo algoritmo PSO.

5. Implementação da MLP com otimizador HGAPSO:No quarto experimento, é implementada a MLP com parâmetros de arquitetura de rede otimizados pelo algoritmo híbrido HGAPSO, que combina as características do GA e do PSO.

6. Comparativo: Os passos anteriores são realizados para os três *datasets* escolhidos. Após a execução de cada experimento 20 vezes para cada *dataset*, a média do RMSE para cada combinação experimento e *datasets* é calculada, a fim de servir como parâmetro principal para comparação de resultados.

Na Figura 18, temos o fluxo dos experimentos, onde o o objetivo geral é comparar a eficácia das diferentes técnicas de otimização e avaliar se a otimização dos parâmetros por meta-heurísticas proporciona melhorias significativas no desempenho da MLP em comparação com a abordagem com otimização aleatória independente do conjunto de dados utilizado.

Figura 18 – Função Objetivo que contém a Rede Neural.



Fonte: Autor

3.4.1 Função Objetivo

A função objetivo é uma parte crucial do processo de otimização, responsável por avaliar a qualidade das soluções propostas pelos algoritmos de otimização. No contexto deste trabalho, a função objetivo contém a Rede Neural Perceptron Multi-Camadas (MLP), na qual estamos otimizando os paramentos de arquitetura. Dessa forma, quando executando a função objetivo, estamos avaliamos o desempenho da MLP com os parâmetros propostos para a iteração, retornando três resultados: número de camadas ocultas, o número de neurônios por camada e o erro médio quadrático (RMSE).

A função objetivo tem como entrada o vetor das *features* x, o vetor de *target* y, o número de divisões da base de dados para o treinamento $kfold$ para a validação cruzada. Os valores dos parâmetros de arquitetura são inseridos de diferentes formas para cada experimento de otimização. Para o experimento RAND, os parâmetros são inseridos por uma função geradora de números aleatórios entre 0 e 1. Para os casos com otimização por meta-heurística, temos um vetor de duas posições que chamaremos de *chromosomo* - nos casos de GA e HGAPSO -

e *particle* - no caso do PSO - contendo os valores que representam as soluções propostas. Em seguida, a função realiza as etapas abaixo:

1. **Decodificação do Cromossomo/Partícula**: O cromossomo/particula contendo informações sobre o número de camadas ocultas e o número de neurônios por camada, é decodificados utilizando os seguintes limites:

   - **Número de Camadas Ocultas:** Entre 1 e 10.
   - **Número de Neurônios por Camada:** Entre $\frac{n\_features}{2}$ e $n\_features \times 2$, onde $n\_features$ é o tamanho do vetor x que armazena as *features* para cada conjunto de dados.

   A função calcula o número de camadas ocultas e o número de neurônios por camada a partir do cromossomo ou particula e define a arquitetura da rede com essas especificações.

2. **Estratégia de Validação Cruzada**: Para garantir a robustez dos resultados, é utilizada a validação cruzada *k-fold*:

   - **Número de Partições (k)**: 3.
   - **Processo**: Para cada iteração, o conjunto de dados é dividido em 3 *folds*, onde um *fold* é usado para testes e os outros 2 para treinamento. O modelo é avaliado em cada *fold*, e o erro quadrático médio (RMSE) é calculado.

3. **Modelo MLPRegressor:** O modelo *MLPRegressor* da biblioteca *scikit-learn* é configurado com parâmetros padrão, exceto pelo número de iterações (*max_iter* = 4000). Esse número elevado de iterações pode garantir a convergência do modelo em problemas complexos, mas pode aumentar significativamente o tempo de execução. A função de ativação *ReLU* e o otimizador *Adam* foram escolhidas para o treinamento. A lista completa dos parâmetros do `MLPRegressor` é a seguinte:

   - **Função de Ativação**: 'relu'
   - **Otimizador**: 'adam'
   - **Regularização**: `alpha=0.0001`
   - **Tamanho do Lote**: 'auto'

- **Taxa de Aprendizado**: 'constant'
- **Taxa de Aprendizado Inicial**: 0.001
- **Potência de Decaimento**: 0.5
- **Número Máximo de Iterações**: 4000
- **Embaralhamento dos Dados**: `shuffle=True`
- **Estado Aleatório**: `None`
- **Critério de Parada**: `tol=1e-4`
- **Momentum**: 0.9
- **Nesterov's Momentum**: `True`
- **Early Stopping**: `False`
- **Fração de Validação**: 0.1
- **Parâmetros Beta para Adam**: `beta_1=0.9`, `beta_2=0.999`
- **Épsilon para Adam**: 1e-08
- **Número de Iterações Sem Mudança**: 10
- **Número Máximo de Funções de Avaliação**: 15000

4. **Treinamento**: O modelo é treinado usando o conjunto de dados de treinamento (`x_train` e `y_train`).

5. **Predição e Avaliação**: As previsões são geradas para o conjunto de teste (`x_test`), e o RMSE é calculado para medir a performance do modelo. O RMSE é calculado com a função `mean_squared_error` da biblioteca `sklearn.metrics`, com o parâmetro `squared=False` para obter a raiz quadrada do erro quadrático médio.

6. **Armazenamento dos Resultados**: O RMSE médio por iteração é calculado e armazenado, assim como a configuração do modelo que gerou o melhor desempenho.

7. **Resultados e Visualização**:

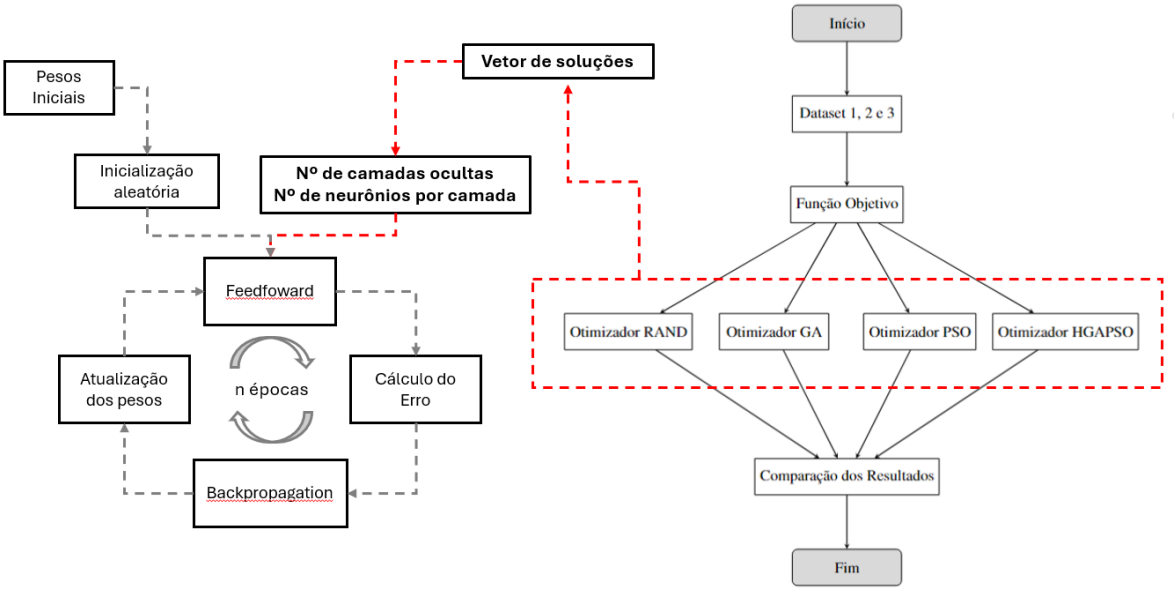
   - **Desempenho do Modelo**: O melhor RMSE e a configuração correspondente são identificados e apresentados. Além disso, a média do RMSE de todas as iterações é calculada.

- **Visualizações**:
  - **Comparação entre Valores Reais e Previsões**: Um gráfico de dispersão é gerado para comparar as previsões do modelo com os valores reais, incluindo uma linha ideal.
  - **RMSE por Iteração**: Um gráfico é criado para mostrar a variação do RMSE ao longo das iterações.

8. **Tempo de Execução**: O tempo total de execução do processo é registrado e convertido para horas, minutos e segundos, fornecendo uma visão sobre a eficiência computacional do experimento. O tempo total de execução para cada amostra não deve exceder 35 minutos.

9. **Funcionalidade**: O objetivo da função é minimizar o erro médio quadrático (RMSE) ao otimizar os parâmetros da arquitetura da rede neural. Os resultados da função objetivo são usados pelos algoritmos de otimização para avaliar e ajustar os parâmetros da rede neural, com a finalidade de encontrar a configuração que proporciona o melhor desempenho preditivo. Na Figura 19 temos a representação da etapa em que os algoritmos realizam o processo de otimização dos parâmetros de arquitetura da Rede Neural.

Figura 19 – Etapa de otimização da Rede Neural.



Fonte: Autor

3.4.2 Otimizador RAND

O algoritmo RAND implementa um processo de busca aleatória para otimizar a arquitetura de uma rede neural Multilayer Perceptron (MLP) usando a função `np.random.rand()` que gera números aleatórios em uma distribuição uniforme no intervalo $[0, 1)$, ou seja, números reais entre 0 (inclusivo) e 1 (exclusivo).

Matematicamente, podemos representar como:

$$x \sim U(0, 1)$$

onde x é o valor gerado pela função, e $U(0, 1)$ indica uma distribuição uniforme no intervalo $[0, 1)$.

Dessa forma, a arquitetura da MLP é definida de forma aleatória, para selecionar o número de camadas ocultas e neurônios por camada, não se tem controle sobre o processo de busca por uma solução, sem aproveitar qualquer estratégia de aprendizado ou otimização dirigida. Isso pode levar a resultados inconsistentes: algumas configurações podem gerar boas soluções, enquanto outras podem ser ineficazes, especialmente em problemas complexos. A falta de direcionamento na busca pode resultar em uma exploração insuficiente do espaço de hiperparâmetros, potencialmente deixando de encontrar uma solução aceitável para o problema.

A seguir, veremos os passos que seguimos para implementar essa abordagem.

1. **Configuração de Parâmetros:** O número de iterações para o método RAND foi definido após a análise dos outros métodos. Como o RAND não possui um sistema de otimização inteligente, foi determinado o número de 328 execuções, que corresponde à média entre o algoritmo com mais execuções (GA, com 338) e o com menos execuções (HGAPSO, com 318). Dessa forma, o método RAND permanece em conformidade com a limitação de tempo de execução estabelecida no experimento. Em cada iteração, o algoritmo configura a arquitetura da rede neural de forma aleatória, onde os limites para o número de camadas ocultas e o número de neurônios por camada são definidos conforme especificado na seção sobre a função objetivo.

2. **Cálculo do RMSE**: O erro médio quadrático (RMSE) foi utilizado como métrica de avaliação, sendo particularmente adequado para problemas de regressão. Como o método RAND não possui uma inteligência de otimização, para realizar uma comparação justa com os outros otimizadores, determinamos a execução de 328 iterações da função objetivo, sorteando novos valores para a configuração da rede a cada execução. Após isso,

foi calculada a média dos valores de RMSE obtidos, a qual representou o valor final da amostra para aquele teste. Assim como nos outros métodos, 20 amostras foram geradas. Os melhores valores também foram armazenados para análise posterior.

3. **Armazenamento das Configurações e Resultados:** A configuração da rede que gerou o menor RMSE é armazenada, juntamente com a média de todos os RMSEs. Isso fornece uma visão clara de qual arquitetura foi mais eficaz durante a busca aleatória. Uma possível melhoria seria armazenar e avaliar outras métricas, como o tempo de treinamento ou a variância do RMSE entre os *folds*.

4. **Visualização:** O algoritmo inclui visualizações dos resultados, como um gráfico de dispersão para comparar os valores reais com as previsões, e um gráfico de linha para visualizar a evolução do RMSE ao longo das iterações. A visualização clara e intuitiva facilita a interpretação dos resultados, ajudando a identificar potenciais problemas no ajuste do modelo.

5. **Tempo de Execução:** O tempo total de execução é calculado e exibido ao final do experimento. Este detalhe é importante em experimentos com múltiplas iterações, sendo útil para otimizar o tempo de computação em experimentos futuros.

6. **Possíveis Melhorias:** Algumas possíveis melhorias incluem:

    - Implementar *early stopping* para reduzir o risco de *overfitting* e diminuir o tempo de treinamento.
    - Utilizar um conjunto de validação separado para garantir que o modelo não esteja superajustado aos dados de treino/teste.

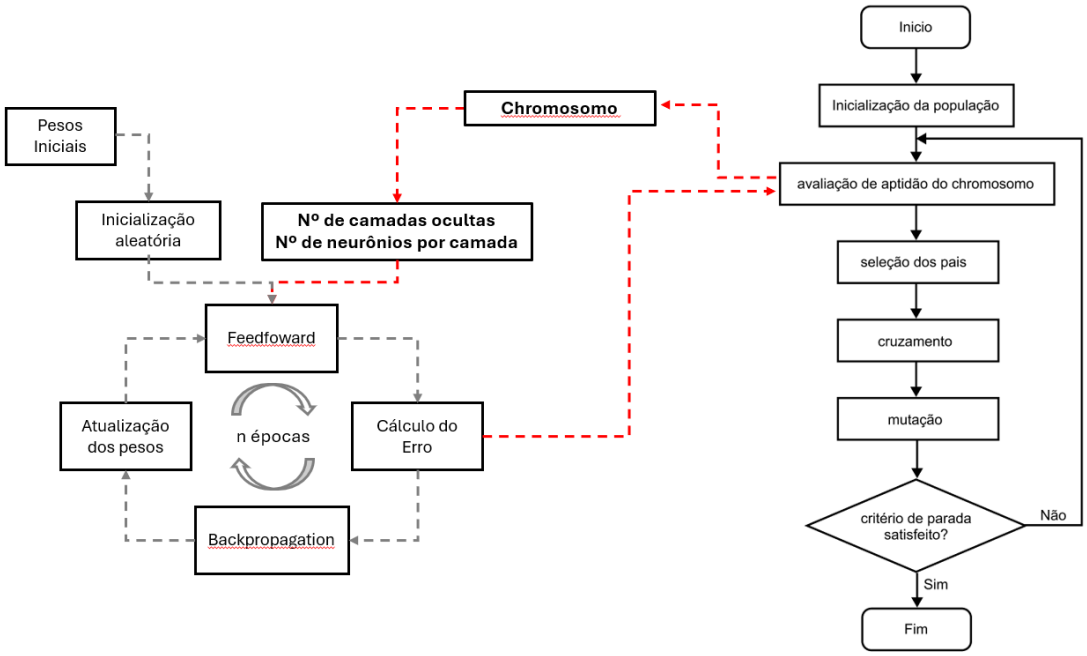
No Apêndice B temos o Algoritmo 4 com mais detalhes sobre a implementação.

3.4.3 Otimizador GA

O Otimizador GA é implementado para otimizar a arquitetura de rede neural MLP (Perceptron Multicamadas) aplicada aos problemas de regressão. O processo começa importando um *dataset* para treinamento e teste, onde as variáveis independentes são armazenadas no vetor de entrada x e a variável alvo no vetor de saída y. Os dados são normalizados antes de serem utilizados na otimização. O objetivo do algoritmo genético é ajustar dois parâmetros principais da MLP: o número de camadas ocultas e o número de neurônios por camada, para minimizar o

erro médio quadrático (RMSE) da previsão da rede neural. Na Figura 20 temos a representação da etapa onde o otimizador GA se relaciona com a Função Objetivo que contém a MLP.

Figura 20 – Otimização da Função Objetivo pelo GA.



Fonte: Autor

O algoritmo genético é executado através de múltiplas gerações, onde em cada geração são realizados processos de seleção, cruzamento e mutação para criar uma nova população de soluções potenciais. A função objetivo avalia o desempenho de cada configuração da MLP usando validação cruzada k-fold para calcular o RMSE médio. Após encontrar a melhor configuração de hiperparâmetros, um modelo MLP final é treinado e avaliado. O experimento gera gráficos que comparam as previsões do modelo com os valores reais e visualizam a evolução do RMSE ao longo das gerações e das execuções da função objetivo, fornecendo insights sobre o desempenho e a eficiência do algoritmo genético na otimização da rede neural.

As definições de parâmetros para a implementação do Otimizador GA inclui:

- **Inicialização da População**: A população inicial é gerada com 14 indivíduos utilizando amostragem com `lhs` (Latin Hypercube Sampling), que garante uma cobertura uniforme do espaço de busca.

- **Gerações**: O número de geração escolhido foi 6, devido a limitação da quantidade de execuções da função objetivo determinada para os experimentos, que para esse caso ficou

com o valor de 338.

- **Mutação**: Introduz aleatoriamente mudanças nos cromossomos com uma probabilidade de mutação de 0.1.
- **Cruzamento**: Gera novos cromossomos a partir da combinação dos pais com uma probabilidade de cruzamento de 0.8.
- **Seleção dos Pais**: Os pais são selecionados por torneio, com o número de selecionados igual a 3, seguindo os métodos vistos no capítulo anterior.

Sobre o funcionamento do Otimizador GA temos a seguir algumas etapas importantes:

1. Resultados da Convergência
    - **Melhores Soluções por Geração**: A evolução das soluções é monitorada e os melhores resultados de cada geração são registrados.
    - **Solução Final**: Após a execução do algoritmo, o modelo final é treinado com os melhores hiperparâmetros encontrados na última geração.
2. Modelo Final

    Após a execução do algoritmo GA, o modelo final é treinado utilizando os melhores hiperparâmetros encontrados na última geração. Os parâmetros do modelo final são:
    - **Número de Camadas Ocultas**: Definido pela solução de convergência encontrada.
    - **Número de Neurônios por Camada**: Definido pela solução de convergência encontrada.
3. Visualização
    - **Comparação de Previsões**: Um gráfico de dispersão é gerado para comparar os valores reais com as previsões do modelo. O gráfico inclui uma linha de referência representando a identidade.
    - **Evolução do RMSE por Geração**: A evolução do erro médio quadrático ao longo das gerações é plotada para análise da eficácia do algoritmo de otimização.
    - **RMSE por Reprodução da Função Objetivo**: A variação do RMSE ao longo das chamadas da função objetivo é plotada para análise adicional.

4. Tempo de Execução

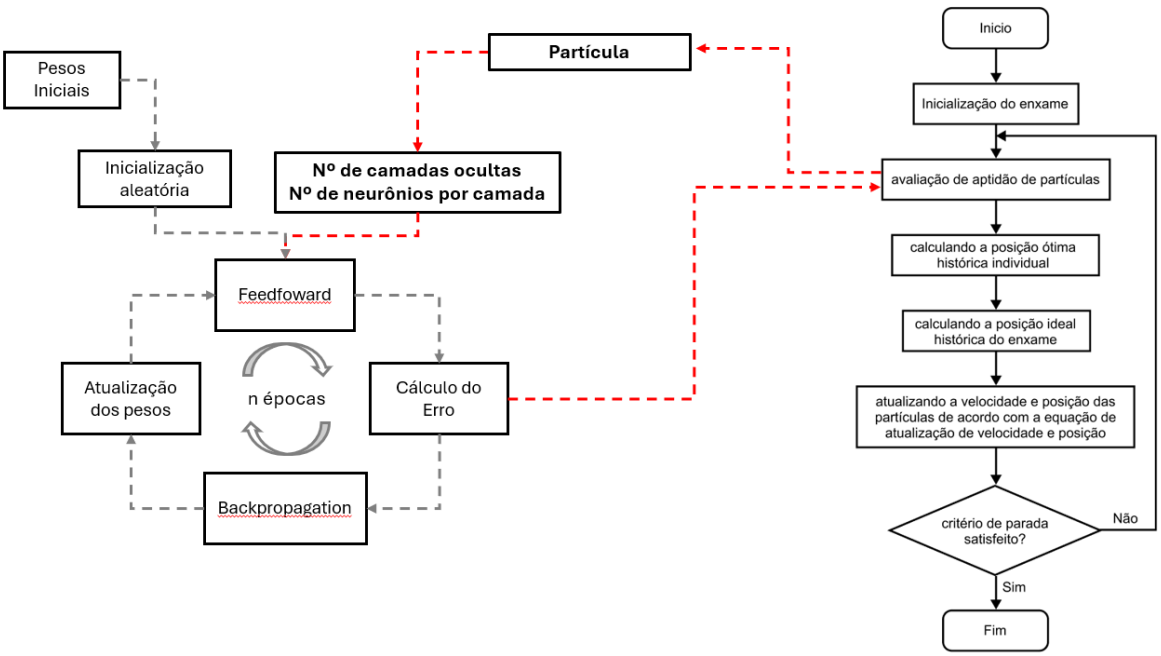
   O tempo total de execução do algoritmo é registrado, e não deve ser superior a 35 minutos, por esse motivo foi limitado o número de execuções da função objetivo.

No Apêndice C temos o pseudocódigo 5 com mais detalhes sobre a implementação.

3.4.4 Otimizador PSO

O terceiro experimento, tem como objetivo otimizar os hiperparâmetros de arquiterura da rede neural Multilayer Perceptron (MLP) usando o algoritmo de Otimização por Enxame de Partículas (PSO). Assim como no experimento anterior do Otimizador GA, iniciamos importando os *datasets* que serão trabalhas. Esses dados são utilizados para treinar e validar o modelo de rede neural, que é avaliado por meio de validação cruzada K-Fold. O PSO é aplicado para ajustar o número de camadas ocultas e o número de neurônios por camada na MLP, com o objetivo de minimizar o erro médio quadrático (RMSE) das previsões. A função objetivo, que calcula o RMSE médio para uma configuração específica de hiperparâmetros, é chamada repetidamente durante as iterações do PSO para encontrar a melhor configuração de rede. Na Figura 21 temos a representação da etapa onde o otimizador PSO se relaciona com a Função Objetivo que contém a MLP.

Figura 21 – Otimização da Função Objetivo pelo PSO.



Fonte: Autor

O algoritmo PSO é inicializado com partículas e velocidades aleatórias. As partículas representam diferentes configurações de hiperparâmetros da MLP, enquanto as velocidades são usadas para atualizar a posição das partículas em cada iteração. A cada iteração, as partículas são atualizadas com base em seus melhores desempenhos individuais e no melhor desempenho global do grupo. Após o término das iterações, o melhor modelo encontrado é treinado e avaliado com todos os dados disponíveis. Os resultados são visualizados com gráficos que mostram a comparação entre valores reais e previsões, bem como a evolução do RMSE ao longo das iterações. Finalmente, o tempo total de execução do algoritmo é exibido, junto com os melhores hiperparâmetros encontrados e o desempenho final do modelo.

As definições de inicialização para a implementação do Otimizador PSO inclui:

- **Inicialização da População**: A população inicial é gerada com 20 partículas utilizando amostragem com `lhs` (Latin Hypercube Sampling).
- **Velocidades**: As velocidades iniciais são geradas com valores contínuos e multiplicadas por 0.1.

Sobre o funcionamento do Otimizador PSO temos a seguir algumas etapas importantes:

1. Atualização da Velocidade e Posição
    - **Velocidade**: Atualizada com base nos componentes de inércia, cognitivo e social:
        - Peso de inércia (`w`): 0.8
        - Coeficientes de aprendizagem (`c1` e `c2`): 2.0
    - **Posição**: Atualizada pela adição da velocidade à posição atual. As posições são limitadas ao intervalo [0, 1].
2. Parâmetros do PSO
    - **Número de Partículas**: 20
    - **Número de Dimensões**: 2 (cada partícula tem dois valores contínuos)
    - **Número de Iterações**: 15
    - **Número de Execuções da Função Objetivo**: Devido à população de 20 e ao número de iterações definido em 15, este experimento resultou em 320 execuções, mantendo-se dentro dos critérios de limitação de tempo previamente estabelecidos.

- **K-Fold**: 3

3. Execução e Resultados

   O algoritmo PSO é executado por 15 iterações. Durante a execução, o tempo de execução total é registrado, e os melhores parâmetros encontrados são:

   - **Número de Camadas Ocultas**: Determinado pela melhor solução global encontrada.
   - **Número de Neurônios por Camada**: Determinado pela melhor solução global encontrada.
   - **RMSE**: Erro médio quadrático obtido.

4. Modelo Final Após a execução do algoritmo PSO, o modelo final é treinado utilizando os melhores hiperparâmetros encontrados. Os parâmetros do modelo final são:

   - **Número de Camadas Ocultas**: Definido pela solução de convergência encontrada.
   - **Número de Neurônios por Camada**: Definido pela solução de convergência encontrada.

   A performance do modelo é avaliada através de validação cruzada K-Fold (k=3) e os resultados são comparados com os valores reais.

5. Visualização

   - **Comparação de Previsões**: Um gráfico de dispersão é gerado para comparar os valores reais com as previsões do modelo. O gráfico inclui uma linha de referência representando a identidade.
   - **Evolução do RMSE**: A evolução do erro médio quadrático ao longo das iterações é plotada para análise da eficácia do algoritmo de otimização.

6. Tempo de Execução

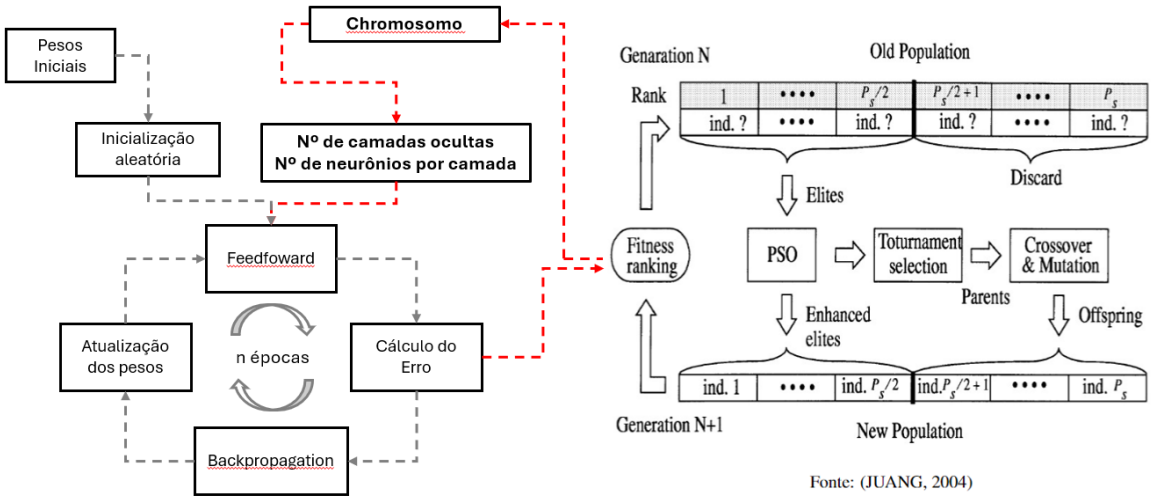
   O tempo total de execução do algoritmo é registrado, e não deve ser superior a 35 minutos, por esse motivo foi limitado o número de execuções da função objetivo.

No Apêndice D temos o Algoritmo 6 com mais detalhes sobre a implementação.

3.4.5 Otimizador HGAPSO

O último experimento é um exemplo de aplicação de um algoritmo híbrido de otimização por GA e PSO denominado HGAPSO, para ajustar os hiperparâmetros da rede neural MLP (Multilayer Perceptron) nos problemas de regressão escolhidos. Os conjunto de dados são importados da mesma forma que nos experimentos anteriores. Após importados, são normalizados e convertidos em um DataFrame para visualização. O algoritmo HGAPSO utiliza duas técnicas: um algoritmo genético para a criação e evolução da população de soluções e o PSO para melhorar a solução dos melhores indivíduos encontrados pelo algoritmo genético. O processo inclui a definição da função objetivo, que avalia o desempenho de uma MLP com parâmetros definidos por um cromossomo (número de camadas ocultas e número de neurônios por camada) usando validação cruzada K-Fold para calcular o erro médio quadrático. Na Figura 22 temos a representação da etapa onde o otimizador HGAPSO se relaciona com a Função Objetivo que contém a MLP.

Figura 22 – Otimização da Função Objetivo pelo HGAPSO.



Fonte: Autor

Durante a execução do algoritmo, é realizada a mutação e o cruzamento para gerar novos indivíduos e selecionar os melhores para a próxima geração. O melhor modelo é treinado com os melhores hiperparâmetros encontrados e avaliado usando o conjunto de dados de teste. Resultados são apresentados em gráficos que comparam valores reais e previstos e mostram a evolução do erro médio ao longo das gerações. O tempo total de execução é calculado e exibido, juntamente com o tamanho da população e o número de gerações. Finalmente, o código exibe

gráficos que mostram a comparação entre valores reais e previsões da MLP final e o erro médio quadrático ao longo das gerações.

1. **Descrição Geral do Algoritmo** O algoritmo utilizado é uma combinação de Algoritmo Genético (GA) e Otimização por Enxame de Partículas (PSO), denominado **HGAPSO**.

2. **Metodologia para o HGAPSO**

   - **Inicialização da População** A população inicial é gerada com 8 indivíduos utilizando amostragem com `lhs` (Latin Hypercube Sampling), que garante uma cobertura uniforme do espaço de busca.
   - **Atualização com PSO Parâmetros**: A atualização das partículas é realizada por um número fixo de 8 iterações, com os seguintes parâmetros:
     - Peso de inércia (`w`): 0.8
     - Coeficientes de aprendizagem (`c1` e `c2`): 2.0

     Velocidades e Posições: As partículas são atualizadas com base nas melhores posições locais e globais. As partículas são ajustadas para garantir que permaneçam no intervalo [0, 1].
   - **Operadores Genéticos Seleção dos Pais**: Os pais são selecionados por torneio, com o número de selecionados igual a 3, seguindo os métodos vistos no capítulo anterior. **Mutação**: Introduz aleatoriamente mudanças nos cromossomos com uma probabilidade de mutação de 0.1. **Cruzamento**: Gera novos cromossomos a partir da combinação dos pais com uma probabilidade de cruzamento de 0.8.
   - **Execução e Resultados** O algoritmo HGAPSO é executado por 4 gerações. Os seguintes parâmetros foram utilizados:
     - **Tamanho da População**: 8 indivíduos
     - **Número de Gerações**: 4
     - **Número de Execuções da Função Objetivo**: Devido à complexidade da arquitetura deste algoritmo, a população foi definida em 8, com 4 gerações e 8 iterações no PSO. Essa configuração de experimento resultou em 318 execuções, mantendo-se dentro dos critérios de limitação de tempo previamente estabelecidos.

Os resultados são monitorados e plotados para observar a convergência do erro médio quadrático ao longo das gerações.

- **Modelo Final** Após a execução do algoritmo HGAPSO, o modelo final é treinado utilizando os melhores hiperparâmetros encontrados. Os parâmetros do modelo final são:
  - **Número de Camadas Ocultas**: Definido pela melhor solução global encontrada.
  - **Número de Neurônios por Camada**: Definido pela melhor solução global encontrada.

  A performance do modelo é avaliada através de validação cruzada K-Fold (k=3) e os resultados são comparados com os valores reais.
- **Visualização Comparação de Previsões**: Um gráfico de dispersão é gerado para comparar os valores reais com as previsões do modelo. O gráfico inclui uma linha de referência representando a identidade. **Evolução do RMSE**: A evolução do erro médio quadrático ao longo das gerações é plotada para análise da eficácia do algoritmo de otimização.

No Apêndice E temos o Algoritmo 7 com mais detalhes sobre a implementação.

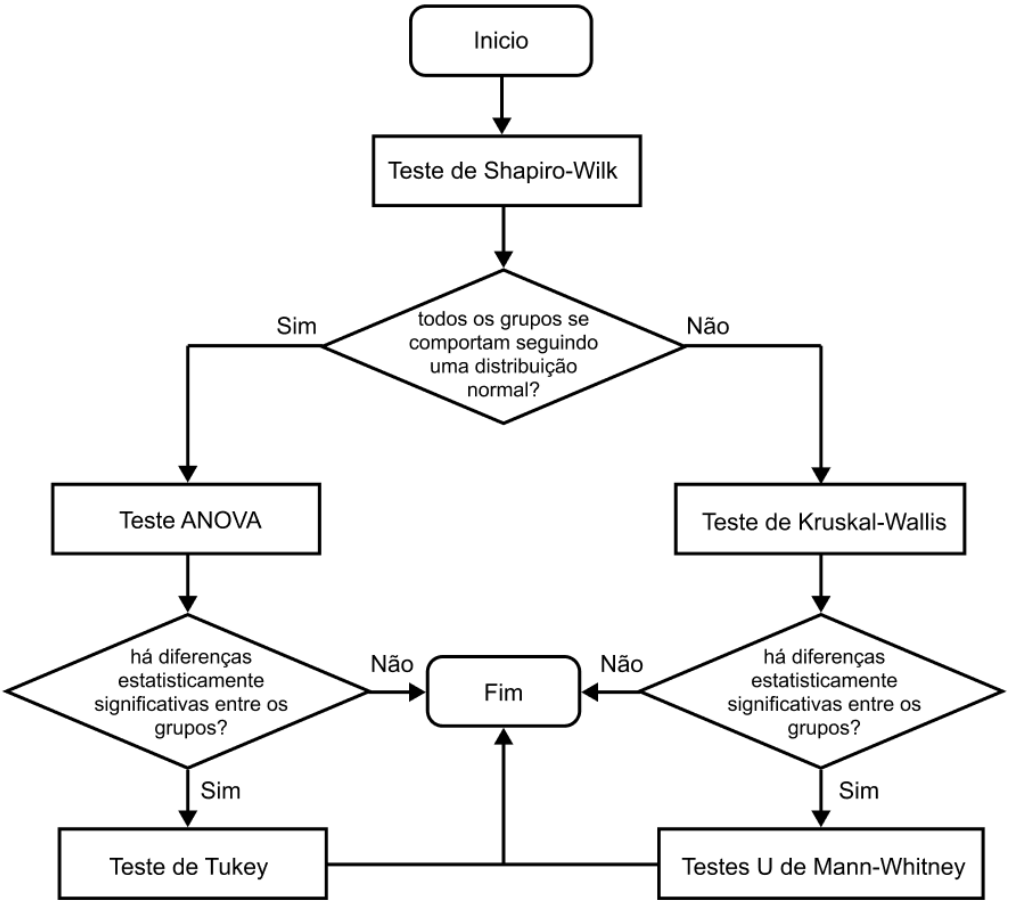
3.5 Análise Estatística

Para melhor visualização do caminho escolhido para a análise estatística, temos na Figura 23 uma representação das decisões que serão seguidas para aplicação dos teste.

3.5.1 Teste de Shapiro-Wilk

O teste de Shapiro-Wilk foi proposto para verificar a normalidade dos dados para cada método de otimização. Este teste será realizado em cada um dos quatro otimizadores, e para cada uma das três bases de dados que estão sendo trabalhadas neste projeto. Para cada base de dados, será avaliada a normalidade para cada um dos grupos, o que definirá qual teste será aplicado posteriormente. Para o caso de haver um dos grupos que não siga os padrões de distribuição normal, o Teste de Kruskal-Wallis, não paramétrico deverá ser aplicado. Caso todos os grupos se comportem seguindo uma distribuição normal, o teste comparativo ANOVA será utilizado.

Figura 23 – Fluxo de decisões para Análise Estatística



Fonte: Autor

3.5.2 Teste de Kruskal-Wallis

O teste de Kruskal-Wallis foi proposto para comparar os erros médios entre todos os métodos de otimização simultaneamente. Este é um teste não paramétrico que avalia se há diferenças significativas entre as medianas de mais de dois grupos independentes. Se o resultado de *p-valor* indica se as diferenças observadas entre as medianas são estatisticamente significativas. Um p-valor baixo sugere que pelo menos um dos métodos difere significativamente dos outros. Caso seja identificado que há diferença significativa entre algum dos grupos, o teste Testes U de Mann-Whitney será realizado a fim de identificar quais grupos possuem diferenças significativas entre si.

3.5.3 Testes U de Mann-Whitney

Foram propostos os testes U de Mann-Whitney para comparações pares entre os métodos de otimização. Este teste não paramétrico é usado para comparar dois grupos independentes

e determinar se há uma diferença significativa nas medianas. Para cada par de métodos, será calculado p *p-valor*, que indica a significância das diferenças observadas. Um p-valor baixo indica diferenças significativas entre os dois grupos comparados.

3.5.4 Teste ANOVA (Análise de Variância)

Para o caso de todos os grupos de um **dataset** segurem a distribuição normal, o teste ANOVA é utilizado para comparar as médias entre os grupos independentes simultaneamente e verificar se há diferenças estatisticamente significativas entre eles. Se o teste ANOVA indicar diferenças significativas, um teste post hoc pode ser realizado para identificar quais grupos são diferentes entre si. Para o caso de idetificarmos diferenças significativas entre os grupos, o Teste de Tukey, um teste post hoc será realizado para múltiplas comparações entre os grupos.

3.5.5 Teste de Tukey

O Teste de Tukey, também conhecido como Teste de Tukey para Comparações Múltiplas, é um teste post hoc utilizado após a realização de um teste ANOVA, quando se deseja identificar quais grupos específicos são significativamente diferentes entre si. Este teste é projetado para realizar comparações múltiplas entre todos os pares de grupos.

3.5.6 Interpretação

Os resultados dos testes de Shapiro-Wilk indicam se os erros médios de cada método seguem uma distribuição normal. O testes de Kruskal-Wallis e ANOVA fornecem uma visão geral se há diferenças significativas entre todos os métodos, enquanto os testes U de Mann-Whitney e Teste de Tukey ajudam a identificar quais pares específicos de métodos apresentam diferenças significativas.

Essas análises estatísticas permitem uma compreensão mais profunda das diferenças de desempenho entre os métodos de otimização, identificando onde os métodos podem ter vantagens ou desvantagens em termos de erros médios.

3.6 Análise de Resultados

falar sobre os novos testes com os melhores valores encontrados

- Boxplots: Foram criados dois boxplots para visualizar a distribuição dos erros médios dos métodos de otimização:

  - **Com Outliers:** Mostra a distribuição dos dados incluindo outliers. O boxplot destaca os quartis e a mediana dos dados, além de identificar os pontos que são considerados outliers.
  - **Sem Outliers:** Exclui os outliers para fornecer uma visão mais focada na distribuição central dos dados. Isso ajuda a identificar a dispersão e a mediana dos dados sem a influência dos valores extremos.

  Os boxplots foram configurados com cores e estilos distintos para facilitar a visualização e comparação.

- Gráficos de Densidade: Os gráficos de densidade foram gerados para cada método de otimização. Estes gráficos mostram a distribuição contínua dos erros médios, ajudando a identificar padrões de distribuição e possíveis sobreposições entre os métodos. Cada método foi representado com uma cor distinta e a área sombreada indica a densidade dos dados.

3.7 Limitações

Algumas limitações foram identificadas, influenciando os resultados e a análise. As principais limitações são descritas a seguir:

1. **Tempo de Treinamento:** O tempo de treinamento foi limitado a aproximadamente 30 minutos para cada experimento. Para manter essa restrição, o número de execuções da função objetivo foi restrito a 328, com uma variação de ±3% devido às diferenças na arquitetura dos algoritmos. Essa limitação também afetou o número de indivíduos na população inicial, o número de gerações no Algoritmo Genético (GA) e o número de iterações no Algoritmo de Otimização por Enxame de Partículas (PSO).

2. **Número de Amostras:** O estudo foi limitado a 20 amostras por experimento devido ao alto tempo de demanda envolvido. Embora esse número tenha sido suficiente para realizar análises estatísticas, uma quantidade maior de amostras poderia proporcionar uma visão mais detalhada e robusta dos resultados.

3. **Limitação das Meta-heurísticas:** A pesquisa focou em duas meta-heurísticas principais, o GA e o PSO, e na combinação dessas técnicas. A comparação foi realizada ape-

nas com a configuração aleatória dos parâmetros, não abrangendo outras meta-heurísticas que poderiam potencialmente oferecer resultados variados.

4. **Número de Parâmetros Otimizados:** Para simplificação e viabilidade do estudo, a otimização foi realizada apenas em dois parâmetros. A inclusão de mais parâmetros poderia fornecer uma compreensão mais completa do impacto da otimização em redes neurais.

Essas limitações devem ser consideradas ao interpretar os resultados da pesquisa, e futuras investigações podem explorar esses aspectos para obter uma compreensão mais abrangente e detalhada dos métodos de otimização aplicados.

4 RESULTADOS E DISCUSSÕES

Neste capítulo, veremos inicialmente os resultados dos testes estatísticos para cada experimento juntamente com a análise individual de hipóteses por estudo de caso. Após uma análise e discussão de resultados geral será apresentada juntamente com um teste adicional.

4.1 Resultados dos Testes Estatísticos

Nos três estudos de caso, os testes estatísticos revelam informações importantes sobre o comportamento dos dados gerados pelos diferentes métodos de otimização (RAND, GA, PSO e HGAPSO) para ajustar os parâmetros de uma rede neural feedforward (FNN). A seguir, destacam-se os principais pontos em comum observados nos experimentos, bem como insights importantes derivados dessas análises.

- **Normalidade dos Dados** Na Tabela 2 temos um resumo dos resultados do teste de Shapiro-Wilk, que foi realizado em todos os datasets para verificar a normalidade dos dados nos grupos RAND, GA, PSO e HGAPSO. Nos três casos, os resultados variaram entre os métodos, mas algumas tendências foram observadas quanto à rejeição da hipótese nula:

  - RAND: Em todos os datasets, os dados gerados pelo método RAND passaram no teste de normalidade, sugerindo que a escolha aleatória de parâmetros de rede neural tende a produzir uma distribuição de erros que é normalmente distribuída.
  - GA e PSO: Nos três estudos de caso, o método GA apresentou, em geral, maior dificuldade para atender à suposição de normalidade. No primeiro estudo (Auto MPG Dataset), GA falhou no teste de Shapiro-Wilk, enquanto PSO e HGAPSO passaram. Isso pode indicar que a natureza do algoritmo GA, que explora regiões específicas

Tabela 2 – Resultados do Teste de Shapiro-Wilk.

Dataset	Métricas	RAND	GA	PSO	HGAPSO
Auto MPG	Estatística W	0.974	0.2871	0.9419	0.9508
	p-valor	0.8368	0.0000	0.2602	0.3788
	Rejeição da Hipótese Nula	Não	Sim	Não	Não
Energy Efficiency	Estatística W	0.9587	0.9631	0.9385	0.9649
	p-valor	0.5176	0.608	0.2245	0.6451
	Rejeição da Hipótese Nula	Não	Não	Não	Não
Rastrigin Benchmark	Estatística W	0.9854	0.8681	0.9869	0.9234
	p-valor	0.984	0.0109	0.9907	0.1155
	Rejeição da Hipótese Nula	Não	Sim	Não	Não

Fonte: Autor

do espaço de busca, pode gerar distribuições de erros não normais, especialmente em certos contextos.

  - HGAPSO: Em todos os experimentos, o HGAPSO mostrou uma performance estável em termos de normalidade, sugerindo que a combinação dos princípios do GA e PSO no algoritmo híbrido pode levar a uma distribuição mais equilibrada dos erros. Isso reflete a eficácia do HGAPSO em capturar as vantagens de ambos os métodos, sem introduzir a variabilidade extrema observada no GA.

- **Comparação das Distribuições** Na Tabela 3 temos os resultados dos testes da segunda etapa da análise estatística. Após verificar a normalidade dos dados, foram realizados testes adicionais para comparar as distribuições dos erros entre os métodos:

  - Teste de Kruskal-Wallis: Em todos os datasets, os resultados do teste de Kruskal-Wallis indicaram diferenças estatisticamente significativas entre os métodos de otimização, confirmando que as distribuições de erros não são as mesmas entre os grupos. Isso sugere que os métodos de otimização meta-heurística (GA, PSO, HGAPSO) afetam de maneira diferente os parâmetros da FNN em comparação com a escolha aleatória (RAND).

Tabela 3 – Resultados dos Testes de Kruskal-Wallis e ANOVA.

		Teste de Kruskal-Wallis	Teste ANOVA
Dataset	**Métricas**	Hipótese Nula: As medianas dos grupos são iguais.	Hipótese Nula: As médias dos grupos são iguais.
Auto MPG	Estatística H	65.0209	
	p-valor	0.0000	
	Rejeição da Hipótese Nula	Sim. Pelo menos uma das distribuições dos grupos é diferente.	
Energy Efficiency	Estatística F		60.32993
	p-valor		4.6440
	Rejeição da Hipótese Nula		Sim. Pelo menos uma das distribuições dos grupos é diferente.
Rastrigin Benchmark	Estatística H	65.801	
	p-valor	0.0000	
	Rejeição da Hipótese Nula	Sim. Pelo menos uma das distribuições dos grupos é diferente.	

Fonte: Autor

  - Teste U de Mann-Whitney: As análises pós-hoc com o Teste U de Mann-Whitney revelaram que, em muitos casos, as distribuições de erros dos métodos meta-heurísticos (GA, PSO, HGAPSO) diferem significativamente da distribuição do RAND. No entanto, um resultado interessante foi observado entre GA e HGAPSO, onde, em alguns estudos, suas distribuições foram semelhantes. Esse achado sugere que, embora o HGAPSO seja uma combinação dos dois métodos, sua performance em termos de erros pode se aproximar do GA em determinadas situações.

- **Considerações Gerais** Na Tabela 4 temos o comparativo final entre cada um dos pares de algoritmos testados. A seguir, algumas considerações sobre os resultados:

  - Impacto dos Métodos Meta-Heurísticos: Os resultados indicam que o uso de métodos de otimização meta-heurísticos, como GA, PSO e HGAPSO, influencia significativamente os erros de previsão da FNN em comparação com a escolha aleatória de parâmetros (RAND). Isso corrobora a hipótese de que técnicas de otimização

Tabela 4 – Resultados dos Testes de Tukeu e Teste U de Mann-Whitney.

		Testes U de Mann-Whitney			Teste Tukey		
		Hipótese Nula: As duas amostras têm distribuições com a mesma mediana.			Hipótese Nula: As médias de todos os pares de grupos comparados são iguais.		
Dataset	**Comparação**	Estatística U	p-valor	Rejeição da Hipótese Nula	Estatística Q	p-valor	Rejeição da Hipótese Nula
Auto MPG	RAND e GA	400	0.0000001	Sim			
	RAND e PSO	400	0.0000001	Sim			
	RAND e HGAPSO	400	0.0000001	Sim			
	GA e PSO	400	0.0000001	Sim			
	GA e HGAPSO	248	0.1988345	Não			
	PSO e HGAPSO	1	0.0000001	Sim			
Energy Efficiency	RAND e GA				0.4331	0.0001	Sim
	RAND e PSO				1.2354	0.0000	Sim
	RAND e HGAPSO				0.3657	0.0013	Sim
	GA e PSO				-0.8023	0.0000	Sim
	GA e HGAPSO				0.0674	0.8924	Não
	PSO e HGAPSO				-0.8697	0.0000	Sim
Rastrigin Benchmark	RAND e GA	380	0.0000012	Sim			
	RAND e PSO	400	0.0000001	Sim			
	RAND e HGAPSO	396	0.0000001	Sim			
	GA e PSO	400	0.0000001	Sim			
	GA e HGAPSO	291	0.0143593	Sim			
	PSO e HGAPSO	1	0.0000001	Sim			

Fonte: Autor

podem melhorar o desempenho de redes neurais ao encontrar configurações de parâmetros mais adequadas.

– Estabilidade do HGAPSO: O algoritmo HGAPSO mostrou-se consistente em gerar distribuições de erros que atendem às suposições de normalidade e, em alguns casos, suas distribuições foram semelhantes às do GA. Isso reforça a eficácia do HGAPSO como uma abordagem híbrida que captura as vantagens de ambos os métodos, GA e PSO.

– Desempenho do GA: Embora o GA seja amplamente utilizado para otimização, os resultados mostraram que ele pode gerar distribuições de erros não normais, especialmente em situações mais complexas. Isso sugere que o GA pode ser mais suscetível a *outliers* ou a explorar regiões específicas do espaço de busca que não produzem bons resultados em termos de distribuição de erros. No entanto, é importante destacar que essa característica de exploração pode ser uma qualidade útil para problemas complexos. A capacidade do GA de explorar amplamente o ambiente das soluções pode ser vantajosa, desde que haja disponibilidade de hardware

suficiente para realizar uma busca mais profunda ao longo de mais gerações. Isso permite que o GA identifique soluções que outros métodos, menos exploratórios, poderiam não encontrar, tornando-o uma opção viável em cenários onde a exploração completa do espaço de busca é necessária.

Os experimentos realizados confirmam que os métodos de otimização meta-heurísticos impactam de maneira significativa a performance das redes neurais, tanto em termos de erros quanto em termos de distribuição dos mesmos. O HGAPSO, em particular, demonstrou um desempenho estável e consistente, o que o torna uma opção promissora para otimização de parâmetros de FNN. Por outro lado, o GA mostrou-se eficaz, mas pode apresentar desafios em termos de normalidade dos dados gerados. Com base nesses *insights*, o uso de técnicas híbridas e a investigação de novas combinações de algoritmos podem levar a melhorias adicionais no desempenho de redes neurais.

4.2 Análise Qualitativa dos Resultados dos Experimentos

Na Figura 24 adaptada da Tabela 4 vemos que para os três *datasets*, o método RAND apresentou uma diferença significativa em relação aos outros testes, confirmando assim a rejeição da hipótese nula proposta na introdução deste trabalho para esses estudos de caso específicos.

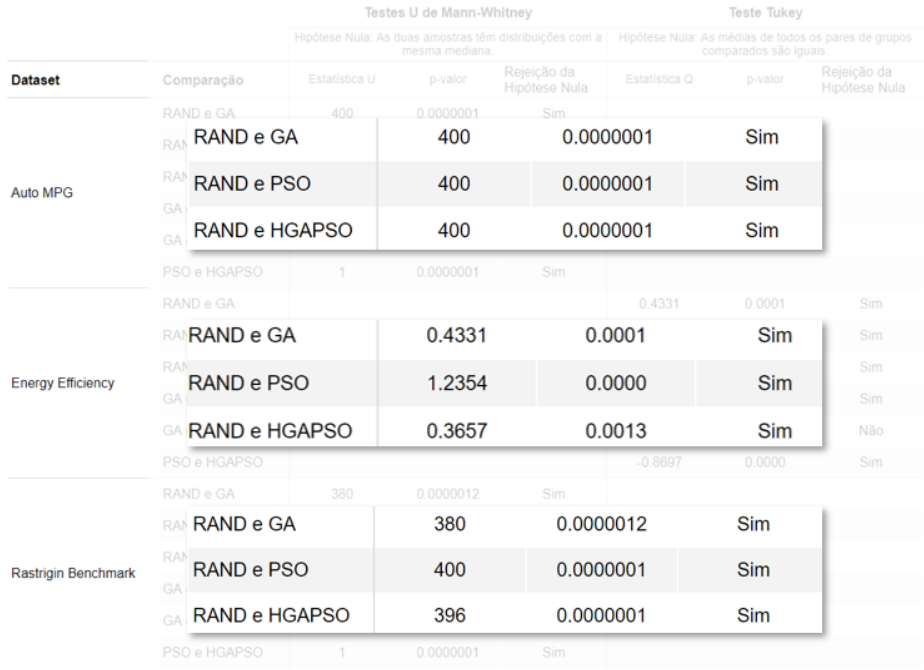
Agora, na Figura 24 também adaptada da Tabela 4, vemos que em dois dos casos, os algoritmos GA e HGAPSO não apresentaram diferenças significativas; no entanto, no caso mais complexo, uma diferença foi observada. Isso ressalta a importância da diversidade de testes para a análise de hipóteses.

4.2.1 Análise Gráfica de Resultados

Nesta subseção, buscamos realizar uma análise gráfica para complementar com elementos qualitativos à análise quantitativa obtida por meio dos testes estatísticos propostos. O objetivo principal é examinar mais a relação e o comportamento dos resultados ao longo dos testes realizados, proporcionando uma visão mais rica dos resultados.

Para alcançar esse objetivo, foram gerados gráficos que permitem observar visualmente as distribuições e comparações dos erros médios obtidos pelos diferentes métodos de otimização (RAND, GA, PSO e HGAPSO) nos três conjuntos de dados distintos: Auto MPG, Energy Efficiency e Rastrigin Benchmark. Os gráficos foram divididos em duas categorias principais:

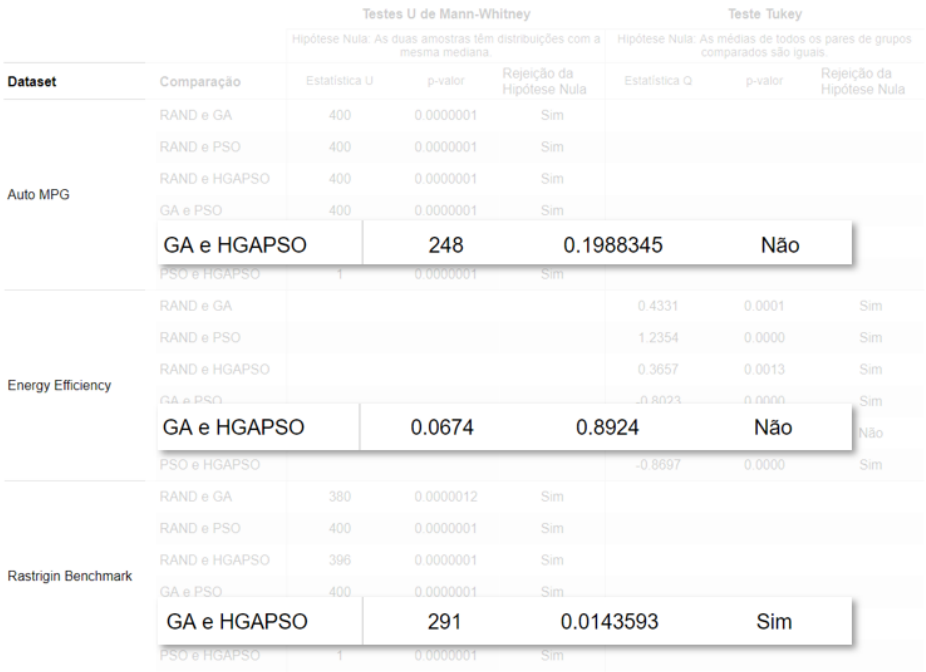
Figura 24 – Diferença significativa entre o RAND e as Meta-Heurísticas.



Dataset	Comparação	Estatística U	p-valor	Rejeição da Hipótese Nula	Estatística Q	p-valor	Rejeição da Hipótese Nula
Auto MPG	RAND e GA	400	0.0000001	Sim			
	RAND e PSO	400	0.0000001	Sim			
	RAND e HGAPSO	400	0.0000001	Sim			
	GA e PSO	400	0.0000001	Sim			
	GA e HGAPSO	248	0.1988345	Não			
	PSO e HGAPSO	1	0.0000001	Sim			
Energy Efficiency	RAND e GA				0.4331	0.0001	Sim
	RAND e PSO				1.2354	0.0000	Sim
	RAND e HGAPSO				0.3657	0.0013	Sim
	GA e PSO				-0.8023	0.0000	Sim
	GA e HGAPSO				0.0674	0.8924	Não
	PSO e HGAPSO				-0.8697	0.0000	Sim
Rastrigin Benchmark	RAND e GA	380	0.0000012	Sim			
	RAND e PSO	400	0.0000001	Sim			
	RAND e HGAPSO	396	0.0000001	Sim			
	GA e PSO	400	0.0000001	Sim			
	GA e HGAPSO	291	0.0143593	Sim			
	PSO e HGAPSO	1	0.0000001	Sim			

Fonte: Autor

Figura 25 – Relação entre algoritmos GA e HGAPSO.

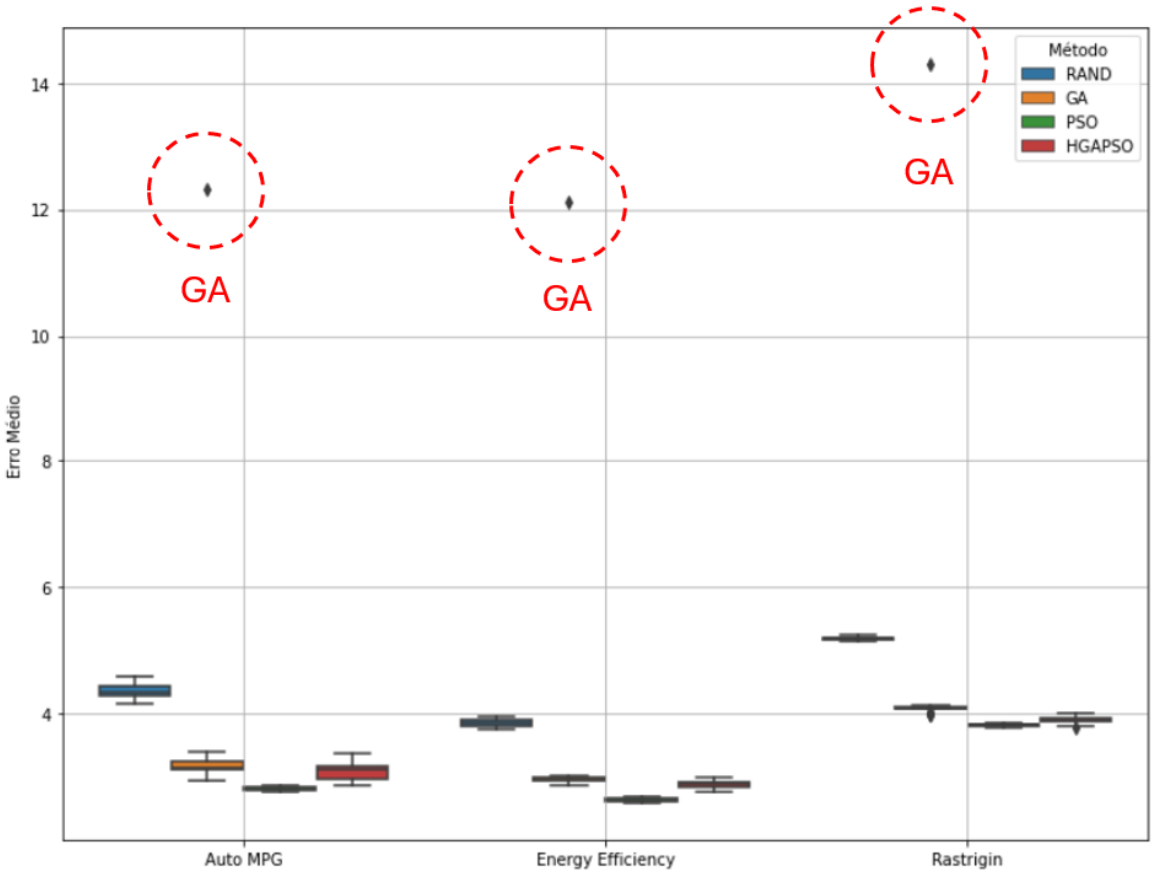


Dataset	Comparação	Estatística U	p-valor	Rejeição da Hipótese Nula	Estatística Q	p-valor	Rejeição da Hipótese Nula
Auto MPG	RAND e GA	400	0.0000001	Sim			
	RAND e PSO	400	0.0000001	Sim			
	RAND e HGAPSO	400	0.0000001	Sim			
	GA e PSO	400	0.0000001	Sim			
	GA e HGAPSO	248	0.1988345	Não			
	PSO e HGAPSO	1	0.0000001	Sim			
Energy Efficiency	RAND e GA				0.4331	0.0001	Sim
	RAND e PSO				1.2354	0.0000	Sim
	RAND e HGAPSO				0.3657	0.0013	Sim
	GA e PSO				-0.8023	0.0000	Sim
	GA e HGAPSO				0.0674	0.8924	Não
	PSO e HGAPSO				-0.8697	0.0000	Sim
Rastrigin Benchmark	RAND e GA	380	0.0000012	Sim			
	RAND e PSO	400	0.0000001	Sim			
	RAND e HGAPSO	396	0.0000001	Sim			
	GA e PSO	400	0.0000001	Sim			
	GA e HGAPSO	291	0.0143593	Sim			
	PSO e HGAPSO	1	0.0000001	Sim			

Fonte: Autor

- Boxplots Com Outliers: Estes gráficos fornecem uma visão geral das distribuições dos erros médios para cada método e *dataset*, sem filtrar os outliers. Essa abordagem permite observar a dispersão geral dos dados e identificar padrões ou anomalias evidentes nas distribuições. Na Figura 26, podemos perceber que o algoritmo otimizador GA, foi que gerou mais pontos de outliers para os diferentes *datasets*.

Figura 26 – Boxplot dos erros médios RMSE com outliers.



Fonte: Autor.

- Boxplots Sem Outliers: Para melhorar a clareza e a visualização das distribuições, os outliers foram removidos utilizando o método de intervalo interquartílico (IQR). Esses gráficos permitem uma análise mais refinada das distribuições dos erros médios, destacando tendências e comparações sem a influência de valores extremos que poderiam distorcer a percepção dos resultados. Na Figura 27, podemos analisar informações interessantes incluindo:

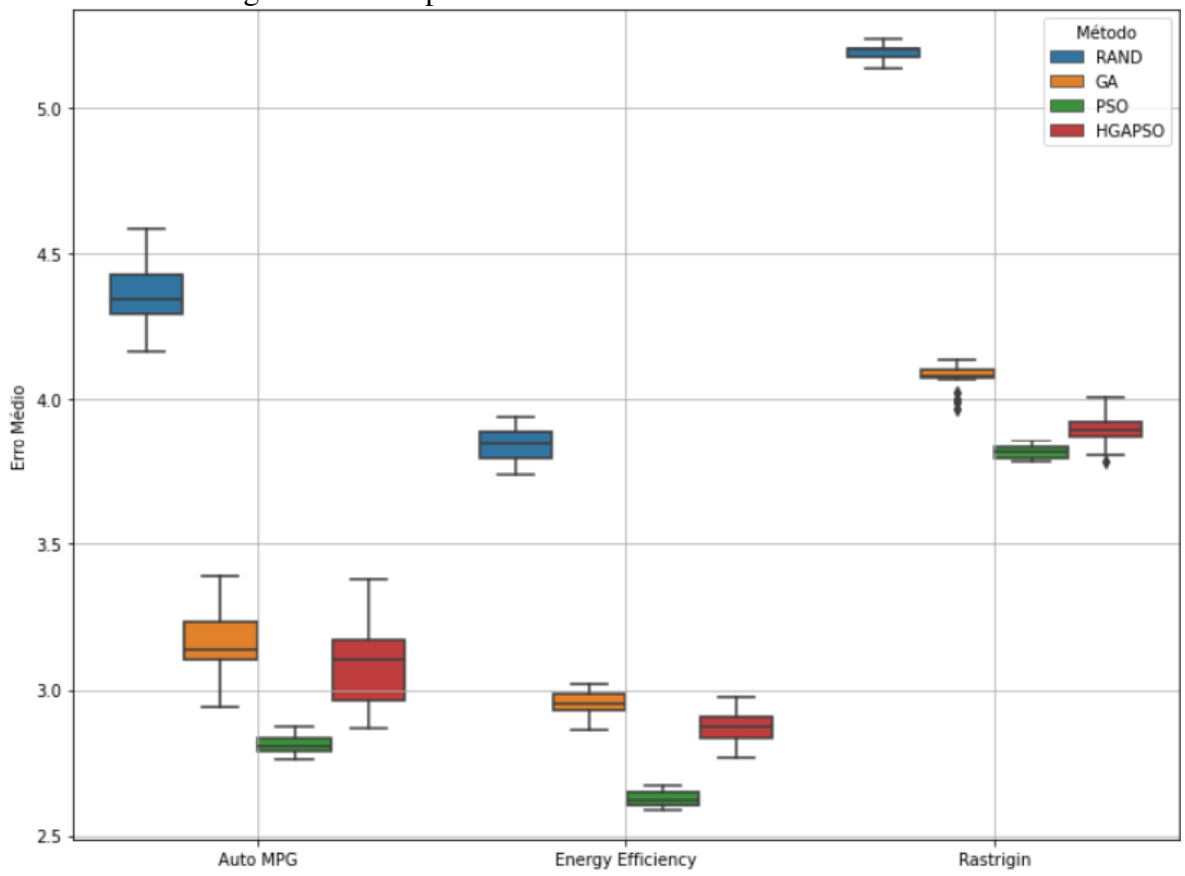
  – Análise dos Resultados de RMSE: Para os três conjuntos de dados analisados, observamos que o otimizador RAND apresentou resultados médios de RMSE signi-

ficativamente piores em comparação com os otimizadores meta-heurísticos. Essa diferença destaca a eficácia superior dos métodos baseados em meta-heurísticas na otimização de redes neurais para esses problemas específicos.

– Desempenho do Otimizador PSO: Entre os métodos de otimização avaliados, o otimizador PSO apresentou consistentemente os melhores resultados em todos os três conjuntos de dados. Isso indica que o PSO foi o mais eficaz na minimização do erro médio, demonstrando sua capacidade superior para encontrar soluções de alta qualidade em comparação com outros métodos.

– Comparação entre GA e HGAPSO: A análise revelou que os algoritmos GA e HGAPSO exibem comportamentos semelhantes nos resultados. Esse comportamento pode ser atribuído à influência significativa do algoritmo GA nas estratégias de hibridização. A semelhança nos resultados sugere que as características do GA desempenham um papel importante no desempenho do HGAPSO.

Figura 27 – Boxplot dos erros médios RMSE sem outliers.



Fonte: Autor.

4.2.2 Resultados Gerais

Na Tabela 5, compilamos as principais informações dos resultados dos testes, apresentando as médias, desvios padrão e melhores resultados obtidos para cada método de otimização aplicado aos diferentes conjuntos de dados. Além disso, a tabela inclui detalhes sobre a configuração de parâmetros que gerou os melhores resultados, embora esses resultados possam não ser predominantes no conjunto geral de amostras.

Tabela 5 – Resultados Médios, Desvios Padrão, Melhores Resultados e Parâmetros das Melhores Configurações.

Dataset	**Método**	**Média**	**Desvio Padrão**	**Melhor Resultado**	**Camadas Ocultas**	**Neurônios por Camada**
Auto MPG Dataset	RAND	4.349	0.133	2.789	9	7
	GA	3.151	0.236	2.944	6	11
	PSO	2.814	0.039	2.762	9	14
	HGAPSO	3.114	0.137	2.870	10	14
Energy Efficiency Dataset	RAND	2.089	0.036	0.77625	5	31
	GA	1.663	0.301	0.912	3	31
	PSO	0.850	0.073	0.704	3	31
	HGAPSO	1.742	0.359	0.997	5	26
Rastrigin Benchmark	RAND	16.207	0.474	3.101	6	7
	GA	11.209	1.225	9.024	7	7
	PSO	4.006	0.716	2.912	10	8
	HGAPSO	9.556	1.437	7.391	7	8

Fonte: Autor.

Os resultados apresentados revelam importantes informações sobre a eficácia dos métodos de otimização em relação à arquitetura das redes neurais e seus parâmetros. No Auto MPG Dataset, o algoritmo PSO se destacou com a menor média de RMSE e o melhor resultado geral. Este desempenho superior pode ser atribuído à sua configuração de parâmetros, que incluiu 9 camadas ocultas e 14 neurônios por camada, sugerindo que uma maior complexidade na arquitetura pode ser benéfica para este tipo de problema. O GA apresentou a segunda menor média e utilizou 6 camadas ocultas e 11 neurônios por camada, o que também demonstra uma configuração relativamente eficaz. O RAND teve o desempenho menos consistente, com a maior média de RMSE e desvio padrão, e uma configuração de 9 camadas ocultas e 7 neurônios por camada.

No Energy Efficiency Dataset, o PSO novamente se destacou com a menor média de RMSE e o melhor resultado, utilizando 3 camadas ocultas e 31 neurônios por camada. Esta configuração sugere que uma arquitetura mais simples com maior número de neurônios pode ser eficaz

para dados de eficiência energética. O GA, com 3 camadas ocultas e 31 neurônios por camada, apresentou resultados competitivos, mas o PSO ainda se mostrou superior. O RAND, com 5 camadas ocultas e 31 neurônios por camada, teve uma média menor que o GA e o HGAPSO, mas ainda inferior ao PSO.

Para o Rastrigin Benchmark, o PSO obteve a menor média de RMSE e o melhor resultado, com 10 camadas ocultas e 8 neurônios por camada, o que indica que uma configuração mais complexa pode ser benéfica. No entanto, o melhor resultado obtido pelo RAND, embora significativamente abaixo da média geral, deve ser interpretado com cautela, pois pode ter sido alcançado por sorte, dada a variabilidade observada nos resultados. As diferenças nas melhores configurações de parâmetros entre os métodos ressaltam a importância de ajustar a arquitetura da rede neural para otimizar o desempenho em problemas específicos.

4.2.3 Teste Adicional

Um teste adicional foi proposto para analisar a consistência dos melhores resultados obtidos nas configurações de parâmetros otimizadas. Utilizando as melhores configurações de camadas ocultas e neurônios por camada identificadas anteriormente, a função objetivo foi executada 20 vezes para cada configuração, a fim de avaliar o comportamento e a estabilidade dos resultados. Na Tabela 6, temos os resultados para o teste.

Tabela 6 – Resultados Médios, Desvios Padrão, Melhores Resultados das Melhores Configurações.

Dataset	**Método**	**Média**	**Desvio Padrão**	**Melhor Resultado**	**Camadas Ocultas**	**Neurônios por Camada**
Auto MPG Dataset	RAND	5.296	4.089	3.050	9	7
	GA	3.098	0.768	2.918	6	11
	PSO	3.092	0.809	2.866	9	14
	HGAPSO	3.251	1.457	2.791	10	14
Energy Efficiency Dataset	RAND	1.617	0.884	1.020	5	31
	GA	1.396	0.786	0.822	3	31
	PSO	1.396	0.786	0.822	3	31
	HGAPSO	1.707	1.058	1.100	5	26
Rastrigin Benchmark	RAND	12.992	6.693	7.845	6	7
	GA	11.716	6.424	7.157	7	7
	PSO	5.461	0.862	2.552	10	8
	HGAPSO	10.082	6.307	3.580	7	8

Fonte: Autor.

A análise dos resultados do teste adicional, que avaliou a função objetivo com as melhores configurações de camadas ocultas e neurônios por camada, oferece uma visão detalhada sobre

a estabilidade e eficácia das configurações otimizadas para cada método e dataset.

Auto MPG Dataset:

- **RAND:** A média dos erros aumentou significativamente para 5.296, com um desvio padrão de 4.089, indicando alta variabilidade nos resultados. O melhor resultado obtido foi 3.050, ainda inferior ao desempenho médio dos outros métodos. Comparado com a tabela anterior, onde o RAND apresentava um erro médio de 4.349, a nova média mostra um desempenho relativamente mais baixo, porém com maior variabilidade, sugerindo que o RAND é menos confiável.

- **GA:** A média do erro é agora de 3.098, com um desvio padrão de 0.768, enquanto o melhor resultado foi 2.918. Esses valores são semelhantes aos anteriores, onde a média era 3.151 e o melhor resultado era 2.944, indicando que a configuração do GA continua a ser estável e eficaz, com pequenas variações em relação aos testes anteriores.

- **PSO:** A média dos erros foi de 3.092, com um desvio padrão de 0.809 e o melhor resultado foi 2.866. Esses valores são comparáveis aos encontrados anteriormente (média de 2.814 e melhor resultado de 2.762), indicando que o PSO mantém uma performance consistente e confiável.

- **HGAPSO:** A média do erro foi de 3.251, com um desvio padrão de 1.457 e o melhor resultado foi 2.791. Apesar de uma leve diminuição na média e um aumento no desvio padrão em comparação com os resultados anteriores (média de 3.114 e melhor resultado de 2.870), o HGAPSO ainda apresenta um desempenho competitivo.

Energy Efficiency Dataset:

- **RAND:** A média dos erros caiu para 1.617, com um desvio padrão de 0.884 e o melhor resultado foi 1.020. Comparado com os resultados anteriores, onde a média era 2.089 e o melhor resultado era 0.77625, os resultados do RAND mostram uma melhoria na média, mas ainda com alta variabilidade.

- **GA e PSO:** Ambos apresentaram médias de 1.396 e melhor resultado de 0.822, sendo idênticos para o PSO. Isso mostra que, para este *dataset*, o GA e o PSO são eficazes e mantêm a mesma performance estável observada anteriormente. A média do GA melhorou ligeiramente em comparação com a anterior (1.663), e a estabilidade foi bem mantida.

- **HGAPSO:** A média dos erros foi de 1.707, com um desvio padrão de 1.058 e o melhor resultado foi 1.100. Embora tenha mostrado uma leve deterioração em comparação com os resultados anteriores (média de 1.742 e melhor resultado de 0.997), o desempenho ainda está dentro de uma faixa competitiva.

Rastrigin Benchmark:

- **RAND:** A média dos erros aumentou para 12.992, com um desvio padrão de 6.693 e o melhor resultado foi 7.845. Esses valores indicam um desempenho inferior em comparação com os resultados anteriores, onde a média era 16.207, sugerindo que o RAND pode ocasionalmente alcançar bons resultados, mas com alta variabilidade.

- **GA:** A média dos erros foi de 11.716, com um desvio padrão de 6.424 e o melhor resultado foi 7.157. Comparado com a média anterior (11.209), a nova média é ligeiramente pior, mas ainda mostra uma consistência razoável.

- **PSO:** A média dos erros foi de 5.461, com um desvio padrão de 0.862 e o melhor resultado foi 2.552. Embora a média seja um pouco pior do que os resultados anteriores (4.006), o PSO ainda demonstra um desempenho robusto em relação aos outros métodos.

- **HGAPSO:** A média dos erros foi de 10.082, com um desvio padrão de 6.307 e o melhor resultado foi 3.580. Embora o melhor resultado tenha melhorado significativamente em comparação com os resultados anteriores (7.391), a média indica uma pequena baixa na estabilidade.

Em resumo, os resultados do teste extra reforçam a robustez das configurações ótimas para o PSO e o GA, demonstrando consistência e confiabilidade em diferentes cenários. O RAND mostrou uma alta variabilidade e um desempenho inconsistente, enquanto o HGAPSO apresentou uma performance mista, com variações na média e no melhor resultado dependendo do *dataset*.

4.3 Resultados de Trabalhos Relacionados

Nesta seção, exploraremos trabalhos acadêmicos relevantes que investigam temas correlatos ao nosso estudo sobre a otimização de redes neurais utilizando algoritmos meta-heurísticos,

como Algoritmos Genéticos (GA), Otimização por Enxame de Partículas (PSO) e Híbridos. A análise desses trabalhos oferece um contexto valioso e uma comparação com nossos resultados.

Para o Auto MPG Dataset, podemos comparar os resultados com DORUK; BAYRAM (2023) onde foram utlizados 5 métodos. Os valores de RMSE obtidos e seus respectivos métodos foram: O *Random Forest Regressor* apresenta o menor RMSE (3.714), em seguida temos *Support Vector Regressor*, com o maior RMSE (7.025), depois LSTM com RMSE (4.533), GRU com RMSE (4.324) e *Decission Tree Regressor* com RMSE (5.28). Comparando com os resultados obtidos neste trabalho, vemos que a hibridização da MLP com a otimização da arquitetura por Meta-Heurísticas, obteve resultados médios de RMSE muito próximos a esses, sendo o RAND com (4.349), GA com (3.151), HGAPSO com (3.114) e o melhor método, inclusive entre os resutlados comparados, o PSO com (2.814).

Para o Energy Efficiency Dataset, é possível comparar com os resultados de REDDY; KUMAR (2019), onde foi obtido um RMSE de 0,4664, enquanto, neste trabalho, o melhor valor alcançado foi de 0,850 com o algoritmo PSO. Em AKGUNDOGDU (2020), temos os melhores resultados de estimativa foram obtidos com o modelo de regressão ANFIS, apresentando RMSE de 0,68. Considerando que o objetivo deste trabalho não era resolver um problema específico, mas sim comparar o desempenho de diferentes técnicas, pode-se considerar aceitável não atingir a mesma precisão que o outro estudo, onde o foco estava justamente na otimização para um problema particular.

Esses estudos fornecem uma base sólida para entender e interpretar nossos resultados, mostrando que as técnicas meta-heurísticas e evolutivas são eficazes na otimização de redes neurais e na melhoria de modelos preditivos complexos.

5 CONCLUSÕES

O objetivo geral deste trabalho foi implementar e analisar combinações entre algoritmos convencionais e meta-heurísticos, com foco na otimização de redes neurais feedforward (FNNs) para problemas de regressão. Foram realizadas com sucesso quatro implementações de algoritmos híbridos, incluindo uma abordagem inovadora que combinou o poder de diferentes heurísticas de pesquisa global. Esses experimentos permitiram não apenas explorar a eficácia de cada método, mas também fornecer insights importantes sobre suas aplicações práticas.

Os objetivos secundários do trabalho foram abordados de maneira sistemática. Em primeiro lugar, foram apresentados os conceitos básicos de otimização, destacando a importância dessas técnicas em cenários complexos onde a simples aplicação de métodos convencionais pode ser insuficiente. Em seguida, foi realizada a implementação de uma abordagem híbrida que combinou o tradicional método de *backpropagation* com meta-heurísticas, demonstrando como a combinação dessas técnicas pode aprimorar os resultados de otimização.

Outro objetivo foi a implementação de uma combinação híbrida de dois ou mais algoritmos meta-heurísticos, como o GA e o PSO, para explorar a influência combinada dessas heurísticas. Os resultados mostraram que essa abordagem híbrida pode oferecer vantagens significativas, como maior capacidade de exploração e melhor desempenho em problemas de alta complexidade.

Além disso, uma análise estatística detalhada dos resultados foi realizada, fornecendo uma base quantitativa para avaliar o desempenho dos diferentes métodos. O teste de Shapiro-Wilk revelou diferenças importantes nas distribuições dos erros, especialmente quando comparados os métodos convencionais com os métodos meta-heurísticos. Isso contribuiu para uma compreensão mais profunda das características e limitações de cada técnica.

No que diz respeito às hipóteses formuladas, os resultados indicam que a hipótese nula (H_0) pode ser rejeitada, pois foi observada uma diferença significativa no RMSE entre as FNNs

com parâmetros aleatórios e aquelas otimizadas com meta-heurísticas. Isso confirma a hipótese alternativa (H_1), que sugere que os métodos meta-heurísticos, como o GA, PSO e HGAPSO, são mais eficazes na otimização de redes neurais quando comparados a abordagens aleatórias.

Por fim, a motivação deste trabalho foi fundamentada na necessidade crescente de técnicas avançadas de otimização em projetos complexos. A adoção de métodos modernos, especialmente aqueles que combinam inteligência artificial e aprendizado de máquina, tem se mostrado essencial para melhorar a produtividade e a qualidade dos resultados em diversas áreas de negócios. Como consultor de negócios, a aplicação dessas técnicas em modelos preditivos traz uma contribuição significativa para a tomada de decisões estratégicas, reforçando o valor das abordagens desenvolvidas neste trabalho.

5.1 Trabalhos Futuros

Para expandir e aprofundar os resultados obtidos nesta pesquisa, várias direções podem ser exploradas em trabalhos futuros:

Exploração de Novos Tipos de Problemas: É fundamental testar os métodos de otimização em diferentes tipos de problemas, como classificação e clusterização, para avaliar a eficácia das abordagens desenvolvidas em cenários variados.

Avaliação com Mais *Datasets*: A realização de testes com uma gama mais ampla de *datasets* é crucial para uma comparação mais robusta e abrangente dos métodos de otimização, contribuindo para uma análise mais precisa das vantagens e limitações dos algoritmos em diferentes contextos.

Utilização de Outras Meta-heurísticas: A incorporação de outras meta-heurísticas, como o *simulated annealing*, bem como outras técnicas emergentes, pode proporcionar novas perspectivas e possíveis melhorias no processo de otimização.

Aumento do Tempo de Teste: Ampliar o tempo de teste e permitir que os algoritmos realizem mais iterações pode resultar em soluções mais refinadas e uma melhor exploração do espaço de soluções, permitindo uma avaliação mais completa do desempenho dos métodos de otimização.

Hibridização de Outras Meta-heurísticas: A combinação de diferentes meta-heurísticas, além das já testadas, pode oferecer novos *insights* e melhorias na eficiência e eficácia dos processos de otimização. A criação e análise de híbridos entre meta-heurísticas não exploradas nesta pesquisa podem revelar abordagens inovadoras e potencialmente mais eficazes.

Otimização de Mais Parâmetros das MLPs: A investigação de métodos para otimizar uma gama mais ampla de parâmetros das redes neurais multicamadas (MLPs) pode contribuir para uma melhor configuração e desempenho das redes.

Otimização de Outros Tipos de Redes Neurais: Explorar a aplicação dos métodos de otimização em diferentes arquiteturas de redes neurais, como redes convolucionais e redes neurais recorrentes, pode ampliar o impacto das abordagens estudadas.

Exploração de Outros Algoritmos Preditivos: Além das redes neurais, a otimização e avaliação de outros tipos de algoritmos preditivos, como árvores de decisão, máquinas de vetores de suporte e algoritmos de ensemble, pode oferecer uma visão mais abrangente sobre a eficácia das técnicas de otimização em diversos contextos.

Essas direções para futuros trabalhos têm o potencial de expandir significativamente o impacto e a aplicabilidade das técnicas de otimização estudadas, proporcionando novas oportunidades para pesquisa e desenvolvimento.

REFERÊNCIAS

AKGUNDOGDU, A. Comparative Analysis of Regression Learning Methods for Estimation of Energy Performance of Residential Structures. *In*: 2020. **Proceedings [...]** [S.l.: s.n.], 2020.

ALTMAN, D. G. **Practical statistics for medical research**. [S.l.]: Chapman and Hall/CRC, 1990.

BIAGI, M. C. **Pesquisa científica**: roteiro prático para desenvolver projetos e teses. [S.l.]: Juruá, 2009.

DORUK, A.; BAYRAM, M. A. Predicting Vehicle Fuel Efficiency: a comparative analysis of machine learning models on the auto mpg dataset. , [S.l.], 2023.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. *In*: MHS'95. PROCEEDINGS OF THE SIXTH INTERNATIONAL SYMPOSIUM ON MICRO MACHINE AND HUMAN SCIENCE, 1995. **Proceedings [...]** [S.l.: s.n.], 1995. p. 39–43.

GOODFELLOW, I. **Deep Learning**. [S.l.]: MIT Press, 2016.

GORI, M.; TESI, A. *et al.* On the problem of local minima in backpropagation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v. 14, n. 1, p. 76–86, 1992.

HAYKIN, S. **Neural networks and learning machines, 3/E**. [S.l.]: Pearson Education India, 2009.

HOLLAND, J. H. **Adaptation in natural and artificial systems**: an introductory analysis with applications to biology, control, and artificial intelligence. [S.l.]: MIT press, 1992.

IEEE. **IEEE Xplore Digital Library**. Acesso em: 12 ago. 2024.

JUANG, C.-F. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, [S.l.], v. 34, n. 2, p. 997–1006, 2004.

KENNEDY, J.; EBERHART, R. C. A discrete binary version of the particle swarm algorithm. *In*: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS. COMPUTATIONAL CYBERNETICS AND SIMULATION, 1997., 1997. **Proceedings [...]** [S.l.: s.n.], 1997. v. 5, p. 4104–4108.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, [S.l.], v. 25, 2012.

KRUSKAL, W. H.; WALLIS, W. A. Use of ranks in one-criterion variance analysis. **Journal of the American statistical Association**, [S.l.], v. 47, n. 260, p. 583–621, 1952.

KUBAT, M. **An introduction to machine learning**. [S.l.]: Springer, 2017.

LAMBORA, A.; GUPTA, K.; CHOPRA, K. Genetic algorithm-A literature review. *In*: COMITCON), 2019., 2019. **Proceedings [...]** [S.l.: s.n.], 2019. p. 380–384.

LUKE, S. **Essentials of Metaheuristics**. second. ed. [S.l.]: Lulu, 2013. Available for free at http://cs.gmu.edu/$nsim$sean/book/metaheuristics/.

MANN, H. B.; WHITNEY, D. R. On a test of whether one of two random variables is stochastically larger than the other. **The annals of mathematical statistics**, [S.l.], p. 50–60, 1947.

MARQUES, I. d. S. Prediçao de séries temporais utilizando algoritmos genéticos. , [S.l.], 2012.

MCKAY, M. D.; BECKMAN, R. J.; CONOVER, W. J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. **Technometrics**, [S.l.], v. 42, n. 1, p. 55–61, 2000.

MENDES, R. d. L. *et al.* Abordagens evolutivas para otimização de redes neurais convolucionais baseadas em algoritmos genéticos. , [S.l.], 2021.

MOSCATO, P. *et al.* On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. **Caltech concurrent computation program, C3P Report**, [S.l.], v. 826, n. 1989, p. 37, 1989.

OJHA, V. K.; ABRAHAM, A.; SNÁŠEL, V. Metaheuristic design of feedforward neural networks: a review of two decades of research. **Engineering Applications of Artificial Intelligence**, [S.l.], v. 60, p. 97–116, 2017.

QUINLAN, R. **Auto MPG**. DOI: https://doi.org/10.24432/C5859H, UCI Machine Learning Repository.

REDDY, A. V. R.; KUMAR, M. S. A Comparative Analysis of Regression Algorithms for Energy Estimation in Residential Buildings. **ICICCT 2019 – System Reliability, Quality Control, Safety, Maintenance and Management**, [S.l.], 2019.

RIBEIRO, D. M. F. *et al.* Busca por arquiteturas de redes neurais artificiais profundas utilizando algoritmos genéticos. , [S.l.], 2022.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, [S.l.], v. 323, n. 6088, p. 533–536, 1986.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, [S.l.], v. 52, n. 3-4, p. 591–611, 1965.

TABACHNICK, B. G.; FIDELL, L. S. **Experimental designs using ANOVA**. [S.l.]: Thomson/Brooks/Cole Belmont, CA, 2007. v. 724.

TSANAS, A.; XIFARA, A. **Energy Efficiency**. DOI: https://doi.org/10.24432/C51307, UCI Machine Learning Repository.

TUKEY, J. W. The problem of multiple comparisons. **Multiple comparisons**, [S.l.], 1953.

WANG, J.; DOWLING, A. W. Pyomo. DOE: an open-source package for model-based design of experiments in python. **AIChE Journal**, [S.l.], v. 68, n. 12, p. e17813, 2022.

WAZLAWICK, R. S. **Metodologia de pesquisa para ciência da computação**. [S.l.]: Elsevier Rio de Janeiro, 2009. v. 2.

WEBER, T. O. ENG04471 - Tópicos Especiais De Engenharia Elétrica I. , [S.l.], 2023.

APÊNDICE A RASTRIGIN BENCHMARK

Algorithm 3 Rastrigin Benchmark

Entrada: Número de amostras $n\_samples$, Número de features $n\_features$, Ruído $noise$, Constante A
Saída: Dados normalizados x, Variável alvo y
Inicializar parâmetros:
$n\_samples \leftarrow 400$
$n\_features \leftarrow 4$
$noise \leftarrow 0.01$
$A \leftarrow 10$
Gerar dados aleatórios:
$X \leftarrow$ Dados aleatórios $\in \mathbb{R}^{n\_samples \times n\_features}$
{Usar semente 42 para reprodutibilidade}
Definir função de Rastrigin: rastrigin_function$X$, A
$n \leftarrow$ Número de features
Retornar: $A \times n + \sum_{i=1}^{n}(X_i^2 - A \cdot \cos(2\pi X_i))$
Gerar a variável alvo y:
$y \leftarrow$ rastrigin_function(X, A)
Adicionar ruído:
$y \leftarrow y +$ Ruído Normal$(0, noise)$
Normalizar dados de entrada:
$x \leftarrow$ Normalizar(X)
{Usar MinMaxScaler}
Criar DataFrame para visualização:
$df \leftarrow$ Criar DataFrame com X e y
=0

APÊNDICE B MLP COM OTIMIZADOR RAND

Algorithm 4 Algoritmo MLP com Otimizador RAND

Início
Carregar o arquivo de dados
Separar os dados em variáveis `x` (entradas) e `y` (saídas)
Definir parâmetros gerais:
    Número de folds `kfold` ← 3
    Limites para camadas ocultas: `lb_layers` ← 1, `ub_layers` ← 10
    Limites para neurônios por camada: `lb_neurons` ← `n_features`/2, `ub_neurons` ← `n_features`*2
Inicializar listas `rmse_per_iteration` e `configurations`
`num_iterations` ← 328
for iterações de 1 a `num_iterations` **do**
    `sum_rmse` ← 0
    Gerar número de camadas ocultas `num_hidden_layers` aleatoriamente
    Gerar número de neurônios por camada `neurons_per_layer` aleatoriamente
    Definir `hidden_layer_sizes` como (neurons_per_layer, ..., neurons_per_layer) com `num_hidden_layers`
    for cada divisão de `train_index` e `test_index` usando `KFold` **do**
        Criar conjuntos de treinamento e teste `x_train`, `x_test`, `y_train`, `y_test`
        Criar e configurar modelo `MLPRegressor` com `hidden_layer_sizes`
        Ajustar modelo com `x_train` e `y_train`
        Prever `y_pred` com `x_test`
        Calcular RMSE usando `y_test` e `y_pred`
        Atualizar `sum_rmse` com RMSE
        Armazenar `y_test` e `y_pred` em `all_y_test` e `all_y_pred`
    end for
    Calcular `avg_rmse` como `sum_rmse` / `kfold`
    Adicionar `avg_rmse` e a configuração `(num_hidden_layers, neurons_per_layer)` às listas
end for
Encontrar o índice da melhor RMSE `best_rmse_index`
Determinar `best_rmse` e `best_config` com base no índice
Calcular a RMSE média de todas as iterações `mean_rmse_all_iterations`
Imprimir melhores resultados e RMSE média
Fim =0

APÊNDICE C OTIMIZADOR GA

Algorithm 5 Algoritmo Otimizador GA

Parâmetros:
Tamanho da População $\leftarrow 14$
{Número de indivíduos na população}
Número de Gerações $\leftarrow 6$
Número de Execuções da Função Objetivo por Teste $\leftarrow$ Limitado a 328 com margem de ±3%
Número de Camadas Ocultas $\leftarrow$ Variável de 1 a 10
Número de Neurônios por Camada $\leftarrow$ Variável de $n\_features/2$ a $n\_features \times 2$
Início do Algoritmo Genético:
Inicializar População com tamanho definido
Para cada geração até o número de gerações:
    Avaliar a função objetivo para cada indivíduo
    Selecionar indivíduos para reprodução
    Aplicar crossover para criar novos indivíduos
    Aplicar mutação aos novos indivíduos
    Substituir a população com novos indivíduos
Fim do Algoritmo Genético
=0

APÊNDICE D OTIMIZADOR PSO

Algorithm 6 Algoritmo Otimizador PSO

Parâmetros:
Número de Partículas $\leftarrow 20$
{Número de partículas na população}
Número de Iterações $\leftarrow 15$
Número de Dimensões $\leftarrow 2$
{Número de hiperparâmetros a serem otimizados}
Inércia $\leftarrow 0.8$
Cognitivo $\leftarrow 2.0$
Social $\leftarrow 2.0$
Limite Inferior e Superior para Número de Camadas $\leftarrow 1$ a 10
Limite Inferior e Superior para Número de Neurônios por Camada $\leftarrow \frac{n\_features}{2}$ a $n\_features \times 2$
Início do Algoritmo PSO:
Inicializar partículas e velocidades aleatoriamente
Para cada iteração até o número de iterações:
    Atualizar velocidades das partículas
    Atualizar posições das partículas
    Aplicar limites às posições das partículas
    Avaliar a função objetivo para cada partícula
    Atualizar melhores posições e melhores scores
Fim do Algoritmo PSO
Treinamento do Modelo Final:
Treinar o modelo MLP com os melhores hiperparâmetros encontrados
Avaliar o modelo com validação cruzada K-Fold
Plotar os resultados de predição e RMSE por iteração
=0

APÊNDICE E OTIMIZADOR HGAPSO

Algorithm 7 Algoritmo Otimizador HGAPSO

Parâmetros:
Número de Indivíduos na População $\leftarrow 8$
{Número de indivíduos na população}
Número de Gerações $\leftarrow 4$
Número de Dimensões $\leftarrow 2$
{Número de hiperparâmetros a serem otimizados}
Probabilidade de Cruzamento $\leftarrow 0.8$
Probabilidade de Mutação $\leftarrow 0.1$
Número de Partículas $\leftarrow 8$
{Número de partículas na população de PSO}
Número de Iterações do PSO $\leftarrow 10$
Inércia $\leftarrow 0.8$
Cognitivo $\leftarrow 2.0$
Social $\leftarrow 2.0$
Limite Inferior e Superior para Número de Camadas $\leftarrow 1$ a 10
Limite Inferior e Superior para Número de Neurônios por Camada $\leftarrow \frac{n\_features}{2}$ a $n\_features \times 2$
Início do Algoritmo HGAPSO:
Inicializar população de indivíduos e partículas
Para cada geração até o número de gerações:
    Avaliar a função objetivo para cada indivíduo da população
    Selecionar a elite da população com base na função objetivo
    Aplicar PSO para melhorar a elite
    Reproduzir novos indivíduos usando cruzamento e mutação
    Atualizar a população com a elite melhorada e novos indivíduos
Fim do Algoritmo HGAPSO
Treinamento do Modelo Final:
Treinar o modelo MLP com os melhores hiperparâmetros encontrados
Avaliar o modelo com validação cruzada K-Fold
Plotar os resultados de predição e RMSE por geração
=0