UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Faculdade de Ciências
Câmpus Bauru

RELATÓRIO CIENTÍFICO

ALGORITMO PORTFÓLIO APLICADO A VISÃO COMPUTACIONAL E PROCESSAMENTO DE IMAGEM

BAURU
Julho, 2025

DR. DOUGLAS RODRIGUES

ALGORITMO PORTFÓLIO APLICADO A VISÃO COMPUTACIONAL E PROCESSAMENTO DE IMAGEM

Relatório de Pós-doutorado realizado na
Universidade Estadual Paulista (UNESP),
Faculdade de Ciências, Bauru.
Orientador: Prof. Dr. João Paulo Papa

FAPESP - 2023/03726-4

BAURU
Julho, 2025

Lista de ilustrações

Figura 1 – Soluções Pareto considerando a otimização dos objetivos f_1 e f_2. 13
Figura 2 – Considere um conjunto de pontos A", B", C", D"e E", bem como seus eixos e uma área azul clara destacando a área de influência"de tais pontos. Um ponto X"é dito ser dominado por outro ponto Y"se o primeiro cair na área de influência de Y". Por exemplo, o ponto A"domina os pontos E"e F", o ponto B"domina D", E"e F", e o ponto D"domina o ponto "F". O ponto C"domina o ponto F", e finalmente os pontos E"e "F"não dominam nenhum outro ponto. A linha azul mostra a frente Pareto-ótima. 14
Figura 3 – Fronteira de Pareto do SPEA2 no dataset Fashion-MNIST, demonstrando a maior diversidade de soluções encontradas (20 soluções válidas). 36
Figura 4 – Fronteira de Pareto do SPEA2 no dataset MNIST, mostrando 8 soluções válidas com excelente distribuição no espaço de compromisso. 37
Figura 5 – Fronteira de Pareto do NSGA-II no dataset CIFAR-10, demonstrando a melhor performance entre os algoritmos tradicionais para este dataset complexo. 38
Figura 6 – Exemplo de arquitetura com baixa modularidade (0.0) encontrada pelo MOEA/D no MNIST, demonstrando estrutura linear simples e eficiente. . . 39
Figura 7 – Exemplo de arquitetura com modularidade moderada encontrada pelo NSGA-II no Fashion-MNIST, mostrando estrutura balanceada entre complexidade e eficiência. 39
Figura 8 – Fronteira de Pareto da hiperheurística no dataset MNIST, mostrando excelente distribuição de soluções com acurácias entre 96.72% e 98.37%. . . . 41
Figura 9 – Fronteira de Pareto da hiperheurística no dataset Fashion-MNIST, demonstrando boa diversidade de soluções no espaço de compromisso. 41
Figura 10 – Fronteira de Pareto da hiperheurística no dataset CIFAR-10, mostrando soluções balanceadas entre precisão e eficiência para este dataset complexo. 42
Figura 11 – Exemplo de arquitetura complexa encontrada pela hiperheurística no CIFAR-10, demonstrando a capacidade de gerar estruturas sofisticadas com boa modularidade. 42
Figura 12 – Fronteira de Pareto da Choice Function no problema ZDT1, demonstrando boa convergência para a frente ótima (linha verde). 45
Figura 13 – Fronteira de Pareto do Multi-Armed Bandit no problema ZDT1, mostrando excelente aproximação da frente ótima. 45
Figura 14 – Fronteira de Pareto da Choice Function no problema ZDT2, mostrando convergência limitada para a frente ótima. 46
Figura 15 – Fronteira de Pareto do Multi-Armed Bandit no problema ZDT2, demonstrando melhor cobertura da frente ótima. 46

Figura 16 – Fronteira de Pareto da Choice Function no problema ZDT3, mostrando convergência para segmentos específicos da frente desconectada. 47
Figura 17 – Fronteira de Pareto do Multi-Armed Bandit no problema ZDT3, demonstrando melhor cobertura dos segmentos desconectados. 48
Figura 18 – Fronteira de Pareto da Choice Function no problema ZDT4, mostrando dificuldade em convergir para a frente ótima devido à multimodalidade. . . 48
Figura 19 – Fronteira de Pareto do Multi-Armed Bandit no problema ZDT4, demonstrando similar dificuldade com a função multimodal. 49
Figura 20 – Fronteira de Pareto da Choice Function no problema ZDT6, mostrando convergência limitada devido à distribuição não-uniforme. 49
Figura 21 – Fronteira de Pareto do Multi-Armed Bandit no problema ZDT6, demonstrando melhor adaptação à distribuição não-uniforme. 50

Lista de tabelas

Tabela 1 – Resultados Comparativos das Estratégias de Seleção em Problemas ZDT . 44
Tabela 2 – Padrões de Seleção de Otimizadores por Estratégia 50

Sumário

1 INTRODUÇÃO 9

2 OTIMIZAÇÃO 12
2.1 Multiobjective Evolutionary Algorithm Based on Decomposition . . 15
2.2 Non-dominated Sorting Genetic Algorithm II 16
2.3 Strength Pareto Evolutionary Algorithm II 18
2.4 Hiperheurística 19
2.4.1 Motivação e Contexto 19
2.4.2 Estrutura Formal e Espaços de Busca 20
2.4.3 Classificação das Hiperheurísticas 20
2.4.3.1 Hiperheurísticas baseadas em seleção 20
2.4.3.1.1 Seleção aleatória 20
2.4.3.1.2 Seleção por roleta/probabilística 21
2.4.3.1.3 Seleção baseada em desempenho 21
2.4.3.1.4 Função de escolha 21
2.4.3.1.5 Multi-armed bandit 21
2.4.3.1.6 Seleção gulosa (greedy) 21
2.4.3.1.7 Recozimento simulado 22
2.4.3.1.8 Aprendizado de máquina 22
2.4.3.1.9 Discussão 22
2.4.3.2 Hiperheurísticas baseadas em adaptação 22
2.4.3.2.1 Adaptação de parâmetros 22
2.4.3.2.2 Adaptação de estratégia 23
2.4.3.2.3 Adaptação do tamanho da população 23
2.4.3.2.4 Controle adaptativo de parâmetros 23
2.4.3.2.5 Auto-adaptação 23
2.4.3.2.6 Discussão 23
2.4.3.3 Hiperheurísticas híbridas 23
2.4.3.3.1 Discussão 24
2.4.3.4 Hiperheurísticas baseadas em geração/composição 24
2.4.3.4.1 Discussão 24
2.4.4 Comparações, Limitações e Perspectivas 24
2.4.4.0.1 Pseudocódigo: Hiperheurística baseada em seleção 25
2.4.4.0.2 Pseudocódigo: Hiperheurística baseada em adaptação 25
2.4.4.0.3 Pseudocódigo: Hiperheurística híbrida 25

2.4.4.0.4 Pseudocódigo: Hiperheurística baseada em geração/composição 26

3 NEUROEVOLUÇÃO DE ARQUITETURAS DE REDES NEURAIS BASEADA EM GRAFOS 27
3.1 Modelagem das Arquiteturas como Grafos 27
3.2 Operadores Evolutivos 27
3.2.1 Crossover (Recombinação) 27
3.2.2 Mutação 28
3.3 Garantia de Validade das Arquiteturas 28
3.4 Serialização e Avaliação 29

4 METODOLOGIA 30
4.1 Função Objetivo e Critérios de Avaliação 30
4.2 Configuração Experimental 31
4.3 Procedimento Experimental 31
4.4 Validação Final e Análise dos Resultados 32
4.5 Experimentos com Funções Benchmark ZDT 32
4.6 Reprodutibilidade e Limitações 33

5 RESULTADOS EXPERIMENTAIS E ANÁLISE 34
5.1 Configuração Experimental 34
5.2 Análise dos Resultados por Algoritmo 34
5.2.1 MOEA/D (Multi-Objective Evolutionary Algorithm based on Decomposition) 34
5.2.2 NSGA-II (Non-dominated Sorting Genetic Algorithm II) 35
5.2.3 SPEA2 (Strength Pareto Evolutionary Algorithm 2) 35
5.3 Análise Comparativa dos Algoritmos 36
5.3.1 Eficiência de Exploração 36
5.3.2 Qualidade das Soluções 36
5.3.3 Complexidade das Arquiteturas 37
5.4 Análise da Modularidade 38
5.5 Experimentos com Hiperheurísticas 38
5.5.1 Configuração Experimental 40
5.5.2 Resultados por Dataset 40
5.5.3 Análise Comparativa com Algoritmos Tradicionais 40
5.5.4 Implicações dos Resultados 43
5.6 Experimentos com Hiperheurísticas em Problemas Benchmark ZDT 43
5.6.1 Configuração dos Experimentos ZDT 43
5.6.2 Características das Funções ZDT 44
5.6.3 Resultados Comparativos 44
5.6.4 Análise por Função Benchmark 44

5.6.4.1 ZDT1 - Função Convexa . . . 44
5.6.4.2 ZDT2 - Função Não-Convexa . . . 44
5.6.4.3 ZDT3 - Frente Desconectada . . . 47
5.6.4.4 ZDT4 - Função Multimodal . . . 47
5.6.4.5 ZDT6 - Distribuição Não-Uniforme . . . 47
5.6.5 Padrões de Seleção de Otimizadores . . . 47
5.6.6 Análise Comparativa das Estratégias . . . 50
5.6.6.1 Superioridade do Multi-Armed Bandit . . . 50
5.6.6.2 Limitações da Choice Function . . . 51
5.6.7 Implicações para Neuroevolução . . . 51

6 CONCLUSÕES E TRABALHOS FUTUROS . . . 52
6.1 Principais Contribuições . . . 52
6.1.1 Integração de Algoritmos Multiobjetivos . . . 52
6.1.2 Neuroevolução Baseada em Grafos . . . 52
6.1.3 Validação Experimental Abrangente . . . 53
6.1.4 Desenvolvimento e Validação de Hiperheurísticas . . . 53
6.2 Análise Comparativa dos Algoritmos . . . 54
6.2.1 Características Distintivas . . . 54
6.2.2 Implicações para Seleção de Algoritmo . . . 54
6.2.3 Análise Comparativa das Hiperheurísticas . . . 54
6.3 Limitações Identificadas . . . 55
6.4 Trabalhos Futuros . . . 55
6.4.1 Melhorias na Configuração Experimental . . . 55
6.4.2 Extensão dos Experimentos . . . 55
6.4.3 Desenvolvimento de Hiper-heurísticas . . . 56
6.4.4 Melhorias no Framework . . . 56
6.4.5 Investigações Teóricas . . . 57
6.5 Impacto e Relevância . . . 57
6.6 Conclusões Finais . . . 57

7 COMENTÁRIOS DO SUPERVISOR . . . 59

REFERÊNCIAS . . . 60

Resumo

Este trabalho apresenta uma investigação abrangente sobre a aplicação de algoritmos de otimização multiobjetiva e hiperheurísticas na neuroevolução de arquiteturas de redes neurais convolucionais modulares. O estudo integra três algoritmos evolutivos multiobjetivo consagrados (MOEA/D, NSGA-II e SPEA2) com estratégias hiperheurísticas baseadas em Choice Function e Multi-Armed Bandit, aplicando-os tanto em problemas de neuroevolução quanto em funções benchmark ZDT (Zitzler-Deb-Thiele). A abordagem proposta utiliza representação baseada em grafos direcionados acíclicos (DAGs) para modelar arquiteturas convolucionais, permitindo operadores evolutivos especializados e garantia de validade estrutural. Os experimentos foram conduzidos em três datasets de classificação de imagens (MNIST, Fashion-MNIST e CIFAR-10), além de cinco funções benchmark ZDT (ZDT1, ZDT2, ZDT3, ZDT4, ZDT6), avaliando simultaneamente acurácia, complexidade computacional e modularidade estrutural. Os resultados demonstram a superioridade do Multi-Armed Bandit sobre a Choice Function em problemas complexos, com hipervolume médio de 0.473417 versus 0.409792, respectivamente. A análise revela que estratégias adaptativas são cruciais para problemas de neuroevolução, com o Multi-Armed Bandit apresentando melhor balanceamento entre exploração e exploração. O trabalho contribui significativamente para o avanço da área, oferecendo tanto insights teóricos quanto ferramentas práticas para a comunidade científica.

1 Introdução

A otimização é um dos pilares fundamentais da ciência computacional, matemática aplicada e engenharia, estando presente em uma vasta gama de aplicações que vão desde o planejamento logístico e financeiro até o treinamento de modelos de inteligência artificial. Em sua essência, a otimização busca encontrar, dentre todas as soluções possíveis de um problema, aquela que melhor atende a um ou mais critérios de desempenho, frequentemente sob restrições complexas. A crescente complexidade dos problemas reais, marcada por alta dimensionalidade, múltiplos objetivos conflitantes e restrições não lineares, impulsionou o desenvolvimento de métodos cada vez mais sofisticados.

Historicamente, métodos clássicos de otimização, como algoritmos baseados em gradiente, programação linear e métodos exatos, desempenharam papel central na resolução de problemas de pequena ou média escala. No entanto, tais métodos frequentemente se mostram ineficazes ou inviáveis diante de problemas de grande porte, não lineares, com múltiplos ótimos locais ou funções objetivo não diferenciáveis. Esse cenário motivou o surgimento das metaheurísticas, algoritmos inspirados em processos naturais, sociais ou físicos, capazes de explorar eficientemente grandes espaços de busca e encontrar soluções satisfatórias mesmo em problemas altamente complexos.

Algoritmos de otimização metaheurística têm sido utilizados para resolver diversos problemas em engenharia, biologia e física, entre outros, visto que muitos desses problemas podem ser modelados como tarefas de otimização. Tais métodos metaheurísticos são inspirados em processos naturais, como a dinâmica social de alguns animais na busca por alimento e sobrevivência ou mesmo em fenômenos físicos.

No que diz respeito às técnicas de otimização, uma atenção considerável tem sido dada às metodologias baseadas em meta-heurísticas, ou seja, abordagens que visam resolver diversos problemas utilizando conceitos baseados na dinâmica social e/ou no comportamento dos seres vivos. Dentre as técnicas mais difundidas, citam-se Algoritmo Genético (GA) (HOLLAND, 1992), Particle Swarm Optimization (PSO) (KENNEDY; EBERHART, 2001), Harmony Search (HS) (GEEM, 2009), e Algoritmo de Pesquisa Gravitacional (GSA) (RASHEDI; Nezamabadi-pour; SARYAZDI, 2009), só para citar alguns.

Embora essas técnicas sejam comumente aplicadas para resolver problemas de objetivo único, elas também podem ser usadas para resolver tarefas multiobjetivo, onde o conceito de fronteira ótima de Pareto substitui a ideia de uma solução ótima global única (FONSECA; FLEMING, 1993; HORN; NAFPLIOTIS; GOLDBERG, 1994; SRINIVAS; DEB, 1994; ZITZLER; THIELE, 1999; KNOWLES; CORNE, 1999; DEB et al., 2002). A otimização multiobjetivo tornou-se essencial em cenários onde múltiplos critérios conflitantes precisam ser considera-

dos simultaneamente, como em projetos de engenharia, alocação de recursos e aprendizado de máquina. O conceito de fronteira de Pareto permite ao tomador de decisão visualizar e escolher entre diferentes soluções de compromisso, cada uma otimizando diferentes aspectos do problema.

No entanto, numerosos algoritmos de otimização multiobjetivo foram propostos ao longo dos anos, projetados para cobrir um amplo espectro de problemas dentro de um domínio específico. Porém, sabe-se que nenhum algoritmo pode resolver todos os problemas, e escolher o mais adequado para uma determinada tarefa, ou seja, que maximize os resultados, é uma tarefa que requer o conhecimento de um especialista.

Uma forma de contornar esta situação é utilizar algoritmos multimétodos, ou seja, a tarefa de otimização é dividida entre um conjunto de algoritmos metaheurísticos que buscam resolver o problema simultaneamente para maximizar a qualidade da solução. A vantagem e o benefício desta abordagem são explorar os pontos fortes de cada algoritmo usado como um componente para suprimir seus pontos fracos.

Mais recentemente, a área de otimização tem assistido ao surgimento das hiperheurísticas, um novo paradigma que visa automatizar a seleção, adaptação e geração de heurísticas ou metaheurísticas, promovendo maior robustez, adaptabilidade e generalização. As hiperheurísticas operam em um nível superior, escolhendo ou combinando heurísticas de acordo com o contexto do problema e o progresso da busca, reduzindo a dependência de conhecimento especializado e facilitando a aplicação em diferentes domínios.

Este projeto tem como objetivo desenvolver algoritmos multimétodos e multiobjetivos para problemas de otimização e testá-los quanto à qualidade das soluções encontradas e análise de desempenho em diferentes domínios, com foco especial na neuroevolução de arquiteturas de redes neurais convolucionais modulares. Além disso, busca-se investigar e propor abordagens baseadas em hiperheurísticas, explorando seu potencial para automação, inovação e eficiência em cenários de otimização complexos, incluindo validação em funções benchmark ZDT e problemas de neuroevolução.

Objetivos Específicos

Com base nas melhorias recentes implementadas na Opytimizer[1], este trabalho estabelece os seguintes objetivos específicos:

1. **Ampliar a Opytimizer para suporte completo a algoritmos multiobjetivo:** Implementar e integrar algoritmos evolutivos multiobjetivo consagrados, como NSGA-II (Non-dominated Sorting Genetic Algorithm II), MOEA/D (Multiobjective Evolutionary Algorithm based on Decomposition), SPEA2 (Strength Pareto Evolutionary Algorithm

[1] https://github.com/recogna-lab/opytimizer

2) e outros relevantes, proporcionando uma interface unificada, flexível e documentada para a resolução de problemas com múltiplos critérios conflitantes.

2. **Desenvolver e incorporar múltiplas estratégias de hiperheurística:** Integrar à Opytimizer mecanismos avançados de hiperheurística, incluindo seleção baseada em desempenho, estratégias do tipo multi-armed bandit, recozimento simulado, abordagens híbridas e geração/composição de heurísticas, promovendo automação, adaptabilidade e inovação no processo de otimização.

3. **Aprimorar a infraestrutura de testes, exemplos e documentação:** Expandir e qualificar a base de testes automatizados, exemplos práticos e documentação técnica, facilitando a adoção, reprodutibilidade e disseminação da Opytimizer por pesquisadores e profissionais da área.

4. **Realizar validação experimental abrangente:** Avaliar e validar as funcionalidades desenvolvidas por meio de experimentos sistemáticos, incluindo problemas de neuroevolução (como a evolução de arquiteturas de redes neurais convolucionais modulares), funções benchmark ZDT (ZDT1, ZDT2, ZDT3, ZDT4, ZDT6) e outros domínios relevantes, analisando a robustez, eficiência e qualidade das soluções geradas, com foco especial na comparação entre estratégias hiperheurísticas (Choice Function vs Multi-Armed Bandit).

2 Otimização

Suponha $\vec{x} = (x_1, x_2, \ldots, x_N)$ ser um vetor N-dimensional, e $\mathcal{S}$ um espaço de busca arbitrário. Problemas de otimização de objetivo único tentam minimizar uma função objetivo $f(\vec{x})$. Portanto, $\vec{x}^*$ é considerado o mínimo global com relação a alguma função objetivo $f : \Re^N \to \Re$ se, e somente se, $f(\vec{x}^*) \leq f(\vec{x}), \forall \vec{x} \in S$. Matematicamente falando, temos:

$$\vec{x}^* = \arg\min_{\forall \vec{x} \in \mathcal{S}}(f(\vec{x})), \tag{1}$$

sujeito a:

$$g_i(\vec{x}) = 0 \quad \forall i = 1, 2, \ldots, p, \; h_i(\vec{x}) \geq 0 \quad \forall i = 1, 2, \ldots, q, \tag{2}$$

onde p e q representam o número de restrições de igualdade $g(\cdot)$ e desigualdade $h(\cdot)$.

Um problema de otimização multiobjetivo visa encontrar o mínimo global $\vec{x}^* \in \mathcal{S}$ que minimize um conjunto de funções M representado por $\vec{f}$, ou seja:

$$\vec{x}^* = \arg\min_{\forall \vec{x} \in \mathcal{S}} \left(\vec{f}(\vec{x})\right) = \arg\min_{\forall \vec{x} \in \mathcal{S}}(f_1(\vec{x}), f_2(\vec{x}), \ldots, f_M(\vec{x})), \tag{3}$$

sujeito a:

$$g_i(\vec{x}) = 0 \quad \forall i = 1, 2, \ldots, p, \; h_i(\vec{x}) \geq 0 \quad \forall i = 1, 2, \ldots, q. \tag{4}$$

O conjunto de todos os valores que satisfazem as restrições acima define a região viável, e qualquer ponto nessa região é considerado uma solução viável. Em um problema multiobjetivo, não há uma única solução que seja ótima com relação a todos os objetivos quando se consideram objetivos conflitantes. Assim, a solução para um problema de otimização multiobjetivo não é mais um valor escalar, mas um vetor na forma de um trade-off"conhecido como *conjunto Pareto-ótimo*. A Figura 1 ilustra um problema de otimização multiobjetivo com duas funções, ou seja, f_1 e f_2. Note que os pontos vermelhos representam as melhores soluções considerando cada função objetivo individualmente, e o ponto verde é a solução ótima considerando agora ambas as funções objetivo. Como não há solução que seja ótima para ambas as funções objetivo, temos o chamado ponto de utopia", que não deve ser obtido como solução, em teoria.

Primeiramente, definimos a *Dominância de Pareto*, onde um vetor de solução $\vec{x}^a$ é dito dominar outro vetor de solução $\vec{x}^b$ (ou seja, $\vec{x}^a \prec \vec{x}^b$) se $f(x^a i) \geq f(x^b i), \forall i = 1, 2, \ldots, N$, e $\exists i \in 1, 2, \ldots, N$ tal que $f(x^a i) > f(x^b i)$. Em relação à Dominância de Pareto, um vetor de

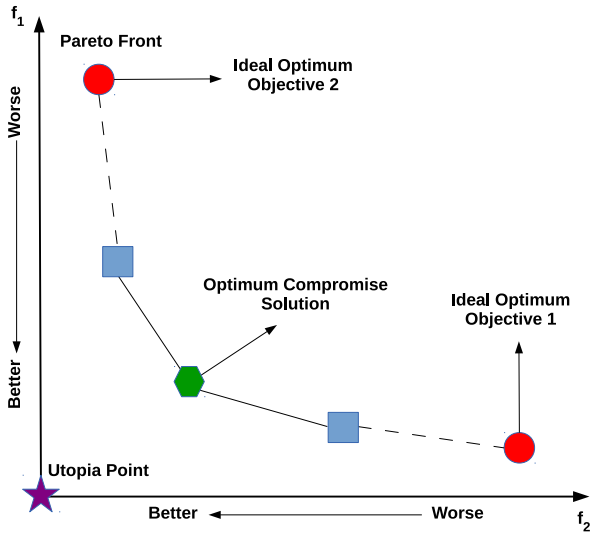


Figura 1 – Soluções Pareto considerando a otimização dos objetivos f_1 e f_2.

solução $\vec{x}^a$ é considerado Pareto-ótimo se, para todo $\vec{x}^b$, $f_j(\vec{x}^a) \geq f_j(\vec{x}^b)$, $j = 1, 2, \ldots, M$, e se existe pelo menos um $j \in 1, 2, \ldots, M$ tal que $f_j(\vec{x}^a) > f_j(\vec{x}^b)$. Portanto, o conjunto Pareto-ótimo $\mathcal{P}^*$ considerando um problema de otimização multiobjetivo $\vec{f}(\vec{x})$ com respeito a todas as soluções Pareto-ótimas é assim definido:

$$\mathcal{P}^* = \vec{x} \in \mathcal{S} \mid \vec{f}(\vec{x}) \prec \vec{f}(\vec{x}'), \ \forall \vec{x}' \in \mathcal{S}. \tag{5}$$

A frente Pareto-ótima PF^* com relação a um problema de otimização multiobjetivo $\vec{f}(\vec{x})$ e o conjunto Pareto-ótimo $\mathcal{P}^*$ é assim definida:

$$PF^* = \vec{f}(\vec{x}) \mid \vec{x} \in \mathcal{P}^*. \tag{6}$$

A Figura 2 ilustra o conceito da ideia de Dominância de Pareto, em que f_1 e f_2 representam duas diferentes funções de aptidão.

Uma alternativa à ideia de otimalidade e eficiência de Pareto, que resulta em um único ponto de solução, é a ideia de uma solução de compromisso. Envolve minimizar a diferença entre o ponto potencialmente ótimo e um ponto de utopia (também chamado de ponto ideal). Assim, o PF^* é limitado pelo ponto nadir z^{nad} e pelo ponto utópico $z^{utopian}$. O z^{nad} pode ser definido da seguinte forma:

$$z^{nad} = \arg \max_{\forall \vec{x} \in \mathcal{P}^*} (f_M(\vec{x})), \tag{7}$$

Por outro lado, $z^{utopian}$ pode ser definido da seguinte forma:

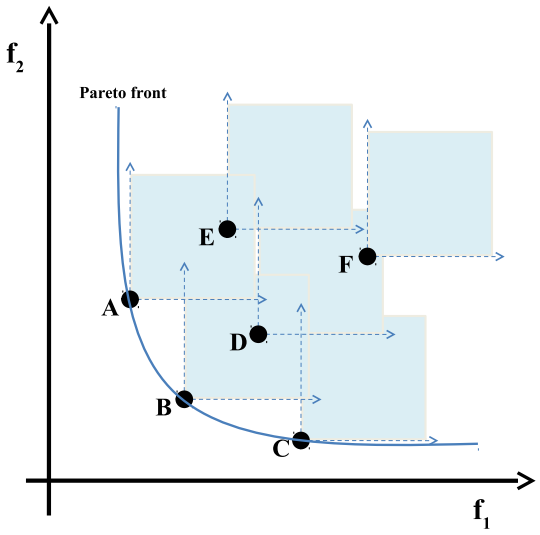


Figura 2 – Considere um conjunto de pontos A", B", C", D"e E", bem como seus eixos e uma área azul clara destacando a área de influência"de tais pontos. Um ponto X"é dito ser dominado por outro ponto Y"se o primeiro cair na área de influência de Y". Por exemplo, o ponto A"domina os pontos E"e F", o ponto B"domina D", E"e F", e o ponto D"domina o ponto "F". O ponto C"domina o ponto F", e finalmente os pontos E"e "F"não dominam nenhum outro ponto. A linha azul mostra a frente Pareto-ótima.

$$z^{utopian} = \arg\min_{\forall \vec{x} \in \mathcal{S}} (f_M(\vec{x})), \tag{8}$$

Em geral, o $z^{utopian}$ é inalcançável. A próxima melhor coisa é uma solução que esteja o mais próximo possível do ponto de utopia. Tal solução é chamada de solução de compromisso e é Pareto-ótima. Dessa forma, pode-se querer minimizar a distância entre uma solução e o ponto de utopia:

$$N(\vec{x}) = |f(\vec{x}) - z^{utopian}|, \tag{9}$$

Grosseiramente falando, técnicas de otimização multiobjetivo são frequentemente divididas em métodos de escalarização e métodos de otimização vetorial, onde dado um vetor de funções objetivo, é possível combinar os componentes desse vetor para compor uma única função objetivo escalar. Além disso, um problema de otimização multiobjetivo pode ser dividido em: (i) métodos clássicos, que utilizam métodos diretos ou baseados em gradientes, e (ii) métodos não tradicionais, que seguem alguns princípios naturais ou físicos (SHUKLA; DEB; TIWARI, 2005).

Para resolver um MOP, é possível identificar dois tipos conceituais de problemas (HORN, 1997): busca e tomada de decisão. O primeiro refere-se ao processo de otimização em que o conjunto viável é amostrado para soluções Pareto-ótimas. O segundo aborda

o problema de selecionar uma solução de compromisso adequada do conjunto Pareto-ótimo. Além disso, um MOP é referido como suporte a um Tomador de Decisão (DM) humano na busca da solução Pareto-ótima de acordo com suas preferências subjetivas (MIETTINEN, 1998). Portanto, pode-se classificar o MOP considerando as preferências do DM nas seguintes quatro categorias: (i) método sem preferência; (ii) método a priori; (iii) método a posteriori; (iv) método interativo. Mais detalhes sobre métodos clássicos são apresentados em (MARLER; ARORA, 2004).

2.1 Multiobjective Evolutionary Algorithm Based on Decomposition

O algoritmo MOEA/D (Multi-Objective Evolutionary Algorithm based on Decomposition), proposto por Qingfu Zhang e Hui Li (ZHANG; LI, 2007), é um algoritmo evolucionário projetado para resolver problemas de otimização multiobjetivo, decompondo um problema de múltiplos objetivos em vários problemas de otimização escalarizada e otimizando-os simultaneamente. O objetivo de um problema de otimização multiobjetivo é encontrar um conjunto de soluções que representem o trade-off entre múltiplos objetivos conflitantes. Formalmente, podemos definir um problema de otimização multiobjetivo da seguinte forma: minimizar $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x}))$, onde $\mathbf{x} \in \Omega \subseteq \mathbb{R}^n$ é o vetor de decisão, $\mathbf{F}(\mathbf{x})$ é o vetor de funções objetivo, e M é o número de objetivos.

O MOEA/D decompõe o problema multiobjetivo original em vários subproblemas de otimização escalarizada, utilizando métodos de decomposição como a soma ponderada, a abordagem de Tchebycheff ou a decomposição de fronteira normal. Cada subproblema é associado a um vetor de pesos $\mathbf{w} = (w_1, w_2, \ldots, w_M)$, onde $w_i \geq 0$ e $\sum_{i=1}^{M} w_i = 1$. Por exemplo, na abordagem de Tchebycheff, a função escalarizada associada ao vetor de pesos $\mathbf{w}$ é definida como: $g^{\text{TCH}}(\mathbf{x}|\mathbf{w}, \mathbf{z}^*) = \max_{1 \leq i \leq M} \{w_i | f_i(\mathbf{x}) - z_i^*|\}$, onde $\mathbf{z}^* = (z_1^*, z_2^*, \ldots, z_M^*)$ é a solução ideal, com $z_i^* = \min_{\mathbf{x} \in \Omega} f_i(\mathbf{x})$.

O MOEA/D mantém uma população de soluções $\mathbf{P} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\}$, onde cada solução $\mathbf{x}^i$ é associada a um vetor de pesos $\mathbf{w}^i$. Além disso, cada solução possui um conjunto de vizinhos, definidos com base na proximidade dos vetores de pesos. Formalmente, o conjunto de vizinhos de $\mathbf{w}^i$ é definido como: $N_i = \{j \mid \text{dist}(\mathbf{w}^i, \mathbf{w}^j) \leq T\}$, onde dist é uma métrica de distância (por exemplo, a distância Euclidiana) e T é o tamanho do conjunto de vizinhos.

O processo evolucionário no MOEA/D envolve a geração de novas soluções através de operadores genéticos (como cruzamento e mutação) e a atualização da população com base na função escalarizada. Para cada subproblema i, um descendente $\mathbf{y}$ é gerado utilizando soluções dos conjuntos de vizinhos N_i: $\mathbf{y} = \text{Crossover}(\mathbf{x}^{p_1}, \mathbf{x}^{p_2})$, onde $p_1, p_2 \in N_i$. Após gerar o descendente $\mathbf{y}$, a população é atualizada. A atualização envolve substituir soluções no conjunto de vizinhos N_i se $\mathbf{y}$ melhorar as respectivas funções escalarizadas: $\mathbf{x}^j \leftarrow \mathbf{y} \quad \text{se} \quad g(\mathbf{y}|\mathbf{w}^j, \mathbf{z}^*) <$

$g(\mathbf{x}^j|\mathbf{w}^j, \mathbf{z}^*) \quad \forall j \in N_i.$

O MOEA/D garante tanto a convergência quanto a diversidade das soluções. A convergência é promovida pela minimização das funções escalarizadas, enquanto a diversidade é mantida através da decomposição do problema e da manutenção de uma população diversificada de vetores de pesos. Concluindo, o MOEA/D é um algoritmo eficaz para a otimização multiobjetivo, utilizando a decomposição do problema para transformar um problema complexo em vários subproblemas mais simples. Através da atualização local de soluções e da manutenção de uma população diversificada, o MOEA/D consegue equilibrar a convergência e a diversidade das soluções, tornando-o uma ferramenta poderosa para problemas de otimização multiobjetivo.

ALGORITMO 1: MOEA/D

Input: Número de subproblemas N, função objetivo $\mathbf{F}$, tamanho da vizinhança T, operadores genéticos
Output: Conjunto de soluções não dominadas

```
Inicializar população P e vetores de pesos {w^1, w^2, ..., w^N};
Calcular solução ideal z*;
for i = 1 to N do
    Definir conjunto de vizinhos N_i para w^i;
while critério de parada não for satisfeito do
    for i = 1 to N do
        Selecionar x^{p1}, x^{p2} de N_i;
        Gerar descendente y utilizando operadores genéticos;
        Atualizar z*;
        for j ∈ N_i do
            if g(y|w^j, z*) < g(x^j|w^j, z*) then
                x^j ← y;
return conjunto de soluções não dominadas;
```

2.2 Non-dominated Sorting Genetic Algorithm II

O NSGA-II (Non-dominated Sorting Genetic Algorithm II) (DEB et al., 2002) é um algoritmo evolutivo desenvolvido para resolver problemas de otimização multiobjetivo. Este algoritmo é amplamente utilizado devido à sua eficiência em encontrar uma distribuição diversificada de soluções próximas à fronteira de Pareto. O objetivo do NSGA-II é otimizar simultaneamente múltiplos objetivos conflitantes, gerando uma população de soluções que representa um compromisso ótimo entre esses objetivos. A abordagem baseia-se em conceitos de seleção, cruzamento e mutação, comuns aos algoritmos genéticos, mas inclui mecanismos específicos para tratar a otimização multiobjetivo.

Cada solução no NSGA-II é representada por um vetor $\mathbf{x} \in \mathbb{R}^n$, onde n é a dimensão do espaço de decisão. A função objetivo é dada por $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))$, onde m é o número de objetivos. Inicialmente, uma população P_0 de tamanho N é gerada aleatoriamente. Esta população será evoluída ao longo de várias gerações para aproximar a fronteira de Pareto.

Para cada indivíduo $\mathbf{x}_i$ na população, define-se o conjunto de indivíduos que ele domina como $S_i = \{\mathbf{x}_j \mid \mathbf{x}_i \prec \mathbf{x}_j\}$, onde $\mathbf{x}_i \prec \mathbf{x}_j$ significa que $\mathbf{x}_i$ domina $\mathbf{x}_j$. O nível de dominância F_k é então definido como $F_1 = \{\mathbf{x}_i \mid \mathbf{x}_i$ não é dominado por nenhum outro indivíduo$\}$ e $F_{k+1} = \{\mathbf{x}_i \mid \mathbf{x}_i$ é dominado apenas pelos indivíduos em $F_1 \cup F_2 \cup \ldots \cup F_k\}$. Esta ordenação divide a população em várias fronteiras de Pareto $F_1, F_2, \ldots$, onde F_1 é a fronteira de Pareto mais alta (não dominada por nenhum outro).

A distância de crowding d_i é calculada para cada indivíduo $\mathbf{x}_i$ em uma fronteira F_k para preservar a diversidade da população. Esta distância é definida como

$$d_i = \sum_{j=1}^{m} \frac{f_j^{\max} - f_j^{\min}}{\Delta f_j},$$

onde $f_j^{\max}$ e $f_j^{\min}$ são os valores máximo e mínimo da função objetivo f_j dentro da fronteira F_k, e Δf_j é a variação dos valores de f_j entre os dois vizinhos mais próximos de $\mathbf{x}_i$ no espaço objetivo.

A seleção é realizada através do torneio binário, onde dois indivíduos são escolhidos aleatoriamente, e o vencedor é determinado com base na ordenação de dominância e na distância de crowding: um indivíduo $\mathbf{x}_i$ é preferido sobre $\mathbf{x}_j$ se $\mathbf{x}_i$ pertence a uma fronteira de Pareto de nível inferior (F_k com k menor) ou se ambos pertencem ao mesmo nível e $d_i > d_j$. Os operadores de cruzamento e mutação são então aplicados para gerar uma nova população Q_t. O cruzamento ocorre entre dois indivíduos selecionados para gerar um ou mais filhos, e a mutação é aplicada a cada indivíduo com uma certa probabilidade para introduzir variabilidade genética.

Após gerar a população Q_t, ela é combinada com a população atual P_t para formar uma população unida $R_t = P_t \cup Q_t$. A ordenação não-dominada é aplicada a R_t, e as melhores N soluções são selecionadas para formar a nova população P_{t+1}. A seleção é feita começando pela fronteira de Pareto mais alta F_1, seguida por F_2, e assim por diante, até que N indivíduos sejam selecionados. Se a fronteira atual não couber totalmente na nova população, a seleção é feita com base na distância de crowding.

O pseudocódigo do NSGA-II pode ser resumido da seguinte forma:

O NSGA-II é um algoritmo poderoso e eficiente para otimização multiobjetivo, oferecendo um bom equilíbrio entre a convergência para a fronteira de Pareto e a manutenção da diversidade da população. Sua popularidade em diversas aplicações práticas e acadêmicas é testemunho de sua eficácia e versatilidade.

ALGORITMO 2: NSGA-II

input : População inicial P_0 com N indivíduos
output : População P_T

Inicialize a população P_0 e calcule $\mathbf{f}(\mathbf{x})$ para cada $\mathbf{x} \in P_0$;
for $t = 0$ **to** $T - 1$ **do**
    Execute a ordenação não-dominada em P_t para obter as fronteiras $F_1, F_2, \ldots$;
    Calcule a distância de crowding para cada indivíduo;
    Realize a seleção, cruzamento e mutação para gerar Q_t;
    Combine P_t e Q_t para formar R_t;
    Execute a ordenação não-dominada em R_t e selecione os N melhores indivíduos para formar P_{t+1};

2.3 Strength Pareto Evolutionary Algorithm II

O SPEA-II (Strength Pareto Evolutionary Algorithm 2) (ZITZLER; LAUMANNS; THIELE, 2001) é um algoritmo evolucionário multiobjetivo que se baseia na evolução diferencial para otimizar simultaneamente múltiplos critérios conflitantes. Ele é uma melhoria sobre seu predecessor, o SPEA, introduzindo várias inovações que aumentam a sua eficiência e eficácia na busca por soluções de Pareto.

O algoritmo inicia com uma população de tamanho N e um conjunto de arquivos externos A de tamanho N_A. A população é composta por soluções candidatas, enquanto o arquivo externo armazena as soluções não dominadas. Inicialmente, a população é denotada por $P_0 = \{x_1, x_2, \ldots, x_N\}$ e o arquivo externo é vazio $A_0 = \emptyset$.

Cada indivíduo na população e no arquivo externo é avaliado com base em uma função de aptidão que considera tanto a dominância quanto a densidade. A força de um indivíduo i é definida como o número de indivíduos que ele domina na população e no arquivo externo, expressa por $S(i) = |\{j \mid i \prec j, j \in P \cup A\}|$, onde $i \prec j$ indica que i domina j. A aptidão de um indivíduo i é a soma das forças dos indivíduos que o dominam, dada por $R(i) = \sum_{j \in P \cup A, j \prec i} S(j)$.

A densidade é calculada usando o k-ésimo vizinho mais próximo para cada indivíduo i. A distância $\sigma_k(i)$ ao k-ésimo vizinho mais próximo é usada para ajustar a aptidão, com a densidade definida como $D(i) = \frac{1}{\sigma_k(i)+2}$, onde $\sigma_k(i)$ é a distância Euclidiana ao k-ésimo vizinho mais próximo. A aptidão final $F(i)$ é uma combinação da aptidão baseada em dominância e densidade, dada por $F(i) = R(i) + D(i)$.

Os indivíduos são selecionados com base na aptidão calculada. Os N melhores indivíduos (com menor aptidão) são selecionados para formar a nova população. Em seguida, os operadores de cruzamento e mutação são aplicados para gerar uma nova população, definidos de acordo com a codificação dos indivíduos (por exemplo, binária, real, etc.).

O arquivo externo é atualizado para armazenar as soluções não dominadas encontradas

até o momento. A união da população atual P e do arquivo externo A é tomada, formando $Q = P \cup A$. As soluções não dominadas em Q são identificadas e adicionadas ao novo arquivo externo $A_{t+1} = \{i \in Q \mid \nexists j \in Q \text{ tal que } j \prec i\}$. Se o tamanho do novo arquivo externo A_{t+1} exceder N_A, um procedimento de truncamento é aplicado para reduzir seu tamanho, preservando a diversidade das soluções.

O algoritmo itera até que um critério de parada seja atendido, como um número máximo de gerações T ou uma convergência das soluções. O SPEA-II é conhecido por sua capacidade de manter um bom equilíbrio entre exploração e exploração e por sua eficiência em termos de complexidade computacional. O uso do arquivo externo e o cálculo de densidade melhoram a convergência para a fronteira de Pareto. Em resumo, o SPEA-II é um algoritmo robusto e eficiente para otimização multiobjetivo, combinando técnicas de dominância, densidade e um arquivo externo para encontrar e manter soluções de Pareto diversificadas ao longo do processo evolutivo.

ALGORITMO 3: SPEA-II

Input: População inicial P_0, tamanho do arquivo N_A, número de gerações T
Output: Soluções de Pareto

Inicialize a população P_0 e o arquivo externo $A_0 \leftarrow \emptyset$;
for $t \leftarrow 1$ **to** T **do**
    Avalie a força $S(i)$ para cada indivíduo $i \in P \cup A$;
    Calcule a aptidão $R(i)$ para cada indivíduo $i \in P \cup A$;
    Calcule a densidade $D(i)$ e a aptidão final $F(i)$;
    Selecione os N melhores indivíduos para formar a nova população P_{t+1} ;
    Aplique operadores de cruzamento e mutação para gerar P_{t+1} ;
    Atualize o arquivo externo A com as soluções não dominadas ;
    Se $|A| > N_A$, aplique o truncamento em A ;
end
return *Arquivo externo* A

2.4 Hiperheurística

2.4.1 Motivação e Contexto

A busca por métodos de otimização robustos, adaptativos e generalizáveis levou ao desenvolvimento do conceito de **hiperheurística**. Historicamente, heurísticas e metaheurísticas foram projetadas para resolver classes específicas de problemas, exigindo conhecimento especializado e ajuste manual de parâmetros. A hiperheurística surge como uma resposta à necessidade de automação e flexibilidade, atuando em um nível superior ao das heurísticas tradicionais, com o objetivo de selecionar, adaptar ou gerar heurísticas de forma dinâmica e inteligente.

2.4.2 Estrutura Formal e Espaços de Busca

Formalmente, a hiperheurística pode ser modelada como um sistema de dois níveis:

- **Nível superior (hiperheurística):** responsável por selecionar, adaptar ou gerar heurísticas/metaheurísticas.
- **Nível inferior (heurísticas):** aplica as heurísticas selecionadas para modificar ou construir soluções no espaço do problema.

Seja $\mathcal{S}$ o espaço de busca de soluções e $\mathcal{H} = \{h_1, h_2, ..., h_K\}$ o conjunto de heurísticas disponíveis. O objetivo é encontrar $x^* \in \mathcal{S}$ tal que $f(x^*) = \min_{x \in \mathcal{S}} f(x)$, utilizando uma sequência de heurísticas $\{h_t\}$ escolhidas por uma política de seleção π:

$$h_t = \pi(\mathcal{H}, \mathcal{S}, t)$$

A política π pode depender do histórico de desempenho, do estado atual da busca ou de critérios probabilísticos.

2.4.3 Classificação das Hiperheurísticas

As hiperheurísticas podem ser classificadas em quatro grandes grupos:

- **Baseadas em seleção**
- **Baseadas em adaptação**
- **Híbridas**
- **Baseadas em geração/composição**

A seguir, cada tipo é detalhado com modelagem matemática, estratégias, exemplos e discussões.

2.4.3.1 Hiperheurísticas baseadas em seleção

A hiperheurística baseada em seleção opera sobre um conjunto finito de heurísticas $\mathcal{H}$, escolhendo a cada iteração qual heurística $h_t \in \mathcal{H}$ aplicar. A política de seleção π pode ser definida de diversas formas:

2.4.3.1.1 Seleção aleatória

Cada heurística tem probabilidade uniforme de ser escolhida:

$$P(h_i) = \frac{1}{K}, \quad \forall i = 1, ..., K$$

Essa estratégia garante diversidade, mas não explora o histórico de desempenho.

2.4.3.1.2 Seleção por roleta/probabilística

Cada heurística h_i recebe um peso w_i proporcional ao seu desempenho acumulado ou recente. A probabilidade de seleção é:

$$P(h_i) = \frac{w_i}{\sum_{j=1}^{K} w_j}$$

Os pesos podem ser atualizados por:

$$w_i^{(t+1)} = \alpha w_i^{(t)} + (1 - \alpha) r_i^{(t)}$$

onde $r_i^{(t)}$ é a recompensa (ex: melhora na função objetivo) e $\alpha \in [0, 1]$ é um fator de suavização.

2.4.3.1.3 Seleção baseada em desempenho

Seleciona a heurística com melhor desempenho médio ou recente:

$$h^* = \arg\max_{h_i \in \mathcal{H}} R(h_i)$$

onde $R(h_i)$ é uma métrica de desempenho (ex: média de melhoria, robustez, tempo).

2.4.3.1.4 Função de escolha

Combina múltiplos critérios para atribuir um valor de escolha a cada heurística:

$$C(h_i) = \alpha R_{recente}(h_i) + \beta R_{hist}(h_i) + \gamma F(h_i, h_j)$$

onde $R_{recente}$ e R_{hist} são recompensas recentes e históricas, F é a frequência de uso, e α, β, γ são pesos.

2.4.3.1.5 Multi-armed bandit

Modela a seleção como um problema de decisão sequencial, buscando equilibrar exploração e exploração. O método Upper Confidence Bound (UCB) é clássico:

$$UCB_i = \bar{r}_i + c\sqrt{\frac{\ln t}{n_i}}$$

onde $\bar{r}_i$ é a recompensa média da heurística h_i, n_i o número de seleções, t o tempo, e c um parâmetro de exploração. O algoritmo seleciona a heurística com maior UCB_i.

2.4.3.1.6 Seleção gulosa (greedy)

Seleciona sempre a heurística com melhor desempenho observado até o momento:

$$h^* = \arg\max_{h_i \in \mathcal{H}} \bar{r}_i$$

2.4.3.1.7 Recozimento simulado

Permite aceitar heurísticas com desempenho inferior com probabilidade decrescente ao longo do tempo:

$$P = \exp\left(-\frac{\Delta f}{T}\right)$$

onde Δf é a diferença de desempenho e T é a temperatura, que decresce a cada iteração.

2.4.3.1.8 Aprendizado de máquina

Modela a seleção como um processo de decisão de Markov (MDP), utilizando algoritmos como Q-learning. A política π é aprendida por meio de recompensas e explorações, podendo adotar estratégias ϵ-greedy:

$$\pi(s) = \begin{cases} \text{argmax}_a Q(s,a), & \text{com probabilidade } 1-\epsilon \\ \text{ação aleatória}, & \text{com probabilidade } \epsilon \end{cases}$$

2.4.3.1.9 Discussão

Cada estratégia apresenta vantagens e limitações. Estratégias probabilísticas e baseadas em bandit são eficazes em ambientes dinâmicos, enquanto estratégias gulosas podem convergir rapidamente, mas correm risco de estagnação. O uso de aprendizado de máquina permite adaptação a contextos complexos, ao custo de maior complexidade computacional.

2.4.3.2 Hiperheurísticas baseadas em adaptação

A hiperheurística baseada em adaptação modifica dinamicamente parâmetros ou estratégias das heurísticas de baixo nível. Seja θ o vetor de parâmetros de uma heurística, a adaptação é modelada por uma função A:

$$\theta_{t+1} = A(\theta_t, \mathcal{P}_t)$$

onde $\mathcal{P}_t$ representa informações do progresso da busca (ex: desempenho, variância, histórico).

2.4.3.2.1 Adaptação de parâmetros

Ajusta parâmetros como taxa de inércia, cruzamento ou mutação. Por exemplo, pode-se usar:

$$\theta_{t+1} = \theta_t + \eta \cdot \nabla R(\theta_t)$$

onde η é a taxa de adaptação e R é a recompensa.

2.4.3.2.2 Adaptação de estratégia

Altera a estratégia de busca conforme critérios como variância do desempenho ou entropia. Por exemplo, alternar entre busca local e global quando a variância da melhoria cai abaixo de um limiar.

2.4.3.2.3 Adaptação do tamanho da população

Ajusta dinamicamente o número de agentes:

$$N_{t+1} = N_t + \delta$$

onde δ é positivo se há progresso, negativo se há estagnação, respeitando limites mínimos e máximos.

2.4.3.2.4 Controle adaptativo de parâmetros

Utiliza janelas móveis ou regressão para detectar tendências e ajustar parâmetros:

$$\theta_{t+1} = \theta_t \cdot (1 + \lambda \cdot \text{trend})$$

onde λ é um fator de ajuste e trend é a inclinação da regressão dos últimos valores de desempenho.

2.4.3.2.5 Auto-adaptação

Os parâmetros são codificados junto com as soluções e evoluem por seleção natural, permitindo adaptação intrínseca ao processo evolutivo.

2.4.3.2.6 Discussão

A adaptação de parâmetros é simples e eficaz para muitos casos, enquanto mecanismos mais sofisticados, como controle adaptativo e auto-adaptação, são indicados para cenários dinâmicos ou de alta complexidade. A estabilidade e a convergência dependem do critério e da frequência de adaptação.

2.4.3.3 Hiperheurísticas híbridas

A hiperheurística híbrida alterna entre seleção de heurísticas e adaptação de parâmetros, modelando o processo como:

$$h_t = \pi(\mathcal{H}, \mathcal{S}, t), \quad \theta_{t+1} = A(\theta_t, \mathcal{P}_t)$$

Em cada iteração, pode-se definir critérios para alternar entre seleção e adaptação, aproveitando sinergias e mitigando limitações de cada abordagem isolada.

2.4.3.3.1 Discussão

Hiperheurísticas híbridas são especialmente úteis em problemas nos quais diferentes fases da busca exigem estratégias e configurações distintas, promovendo robustez e adaptabilidade.

2.4.3.4 Hiperheurísticas baseadas em geração/composição

A hiperheurística baseada em geração ou composição explora o espaço de algoritmos possíveis ao recombinar componentes de algoritmos existentes. Seja $\mathcal{C}$ o conjunto de componentes (inicialização, seleção, cruzamento, mutação, substituição), a geração de um novo algoritmo pode ser modelada por:

$$a = G(c_1, c_2, ..., c_m), \quad c_i \in \mathcal{C}$$

A avaliação dos algoritmos gerados é feita por desempenho em instâncias do problema, e operadores de recombinação e mutação podem ser aplicados para explorar o espaço de algoritmos.

2.4.3.4.1 Discussão

Essa abordagem permite a descoberta automática de algoritmos personalizados, promovendo inovação e adaptabilidade. O desafio reside na avaliação eficiente dos algoritmos gerados e na definição de critérios de recombinação eficazes.

2.4.4 Comparações, Limitações e Perspectivas

- **Baseadas em seleção:** Simples, eficazes em ambientes estáticos, mas podem estagnar se não houver adaptação.
- **Baseadas em adaptação:** Promovem ajuste fino, mas exigem critérios robustos para evitar instabilidade.
- **Híbridas:** Equilibram exploração e adaptação, mas aumentam a complexidade de implementação.
- **Baseadas em geração/composição:** Potencial de inovação, mas alto custo computacional e desafios de avaliação.

A pesquisa em hiperheurísticas segue em expansão, com aplicações em otimização combinatória, aprendizado de máquina, design de algoritmos e problemas dinâmicos. O desenvolvimento de métodos matematicamente fundamentados para análise de convergência, estabilidade e eficiência permanece como um dos principais desafios e oportunidades da área.

2.4.4.0.1 Pseudocódigo: Hiperheurística baseada em seleção

ALGORITMO 4: Hiperheurística baseada em seleção

Input: Conjunto de heurísticas $\mathcal{H}$, solução inicial x_0, função objetivo f, número de iterações T

Output: Melhor solução encontrada

Inicialize $x \leftarrow x_0$

for $t = 1$ **to** T **do**

    Selecione $h_t \in \mathcal{H}$ segundo política π $x' \leftarrow h_t(x)$ Se $f(x') < f(x)$ então $x \leftarrow x'$ (Opcional: atualize estatísticas de desempenho de h_t)

return x

2.4.4.0.2 Pseudocódigo: Hiperheurística baseada em adaptação

ALGORITMO 5: Hiperheurística baseada em adaptação

Input: Heurística h, parâmetros iniciais θ_0, solução inicial x_0, função objetivo f, número de iterações T

Output: Melhor solução encontrada

Inicialize $\theta \leftarrow \theta_0$, $x \leftarrow x_0$

for $t = 1$ **to** T **do**

    $x' \leftarrow h(x; \theta)$ Avalie desempenho de x' Se critério de adaptação for satisfeito: Atualize $\theta \leftarrow A(\theta, \mathcal{P}_t)$ Se $f(x') < f(x)$ então $x \leftarrow x'$

return x

2.4.4.0.3 Pseudocódigo: Hiperheurística híbrida

ALGORITMO 6: Hiperheurística híbrida

Input: Conjunto de heurísticas $\mathcal{H}$, parâmetros iniciais θ_0, solução inicial x_0, função objetivo f, número de iterações T

Output: Melhor solução encontrada

Inicialize $\theta \leftarrow \theta_0$, $x \leftarrow x_0$

for $t = 1$ **to** T **do**

    Se for tempo de seleção: Selecione $h_t \in \mathcal{H}$ segundo política π Se for tempo de adaptação: Atualize $\theta \leftarrow A(\theta, \mathcal{P}_t)$ $x' \leftarrow h_t(x; \theta)$ Se $f(x') < f(x)$ então $x \leftarrow x'$

return x

2.4.4.0.4 Pseudocódigo: Hiperheurística baseada em geração/composição

ALGORITMO 7: Hiperheurística baseada em geração/composição

Input: Conjunto de componentes $\mathcal{C}$, população de algoritmos $\mathcal{A}$, função objetivo f, número de iterações T

Output: Melhor algoritmo gerado

1 Inicialize população $\mathcal{A}$ combinando componentes de $\mathcal{C}$

2 **for** $t = 1$ **to** T **do**

3 Para cada algoritmo $a \in \mathcal{A}$: Execute a por algumas iterações e avalie desempenho Selecione algoritmos mais promissores Recombine componentes para gerar novos algoritmos (Opcional: mutação de componentes)

4 **return** *Melhor algoritmo*

3 Neuroevolução de Arquiteturas de Redes Neurais Baseada em Grafos

A neuroevolução de arquiteturas de redes neurais consiste na aplicação de algoritmos evolutivos para a busca automática de topologias e hiperparâmetros de redes neurais. Neste trabalho, adota-se uma abordagem baseada em grafos direcionados acíclicos (DAGs), permitindo a representação flexível e generalizada de arquiteturas convolucionais.

3.1 Modelagem das Arquiteturas como Grafos

Cada indivíduo da população é representado por um grafo $G = (V, E)$, onde V é o conjunto de nós (operações, como convoluções, pooling, ativações) e E é o conjunto de arestas (conexões entre operações). Os nós são parametrizados por atributos como tipo de operação, número de filtros, tamanho do kernel, função de ativação, entre outros. As arestas definem o fluxo de dados, respeitando restrições de aciclicidade e compatibilidade dimensional.

A inicialização da população é realizada por meio da geração aleatória de grafos válidos, garantindo a existência de um caminho funcional do nó de entrada ao nó de saída. Regras de construção asseguram a ausência de ciclos e a coerência das dimensões dos tensores ao longo do grafo.

3.2 Operadores Evolutivos

A evolução das arquiteturas é conduzida por operadores genéticos adaptados para atuar sobre grafos:

3.2.1 Crossover (Recombinação)

O crossover realiza a recombinação de subgrafos entre dois indivíduos pais, promovendo a exploração do espaço de busca. Uma estratégia comum consiste em selecionar aleatoriamente um ponto de corte em cada grafo e trocar subestruturas, preservando a conectividade e a validade dos descendentes.

ALGORITMO 8: Crossover entre grafos

Input: Dois grafos pais $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$

Output: Dois descendentes válidos G'_1, G'_2

Selecione aleatoriamente um nó de corte $v_1 \in V_1$ e $v_2 \in V_2$; Extraia subgrafos $S_1 \subseteq G_1$ a partir de v_1 e $S_2 \subseteq G_2$ a partir de v_2; Troque S_1 e S_2 entre os grafos pais; Reconecte as arestas para manter a aciclicidade e a conectividade; Verifique a validade estrutural e funcional dos descendentes; **if** *algum descendente for inválido* **then**

    Repare ou descarte o descendente;

return G'_1, G'_2;

3.2.2 Mutação

A mutação introduz variações pontuais na estrutura do grafo, promovendo diversidade. As mutações podem envolver adição, remoção ou modificação de nós e arestas, bem como alteração de parâmetros dos nós.

ALGORITMO 9: Mutação de grafo

Input: Grafo $G = (V, E)$, probabilidade de mutação p

Output: Grafo mutado G'

para *cada nó* $v \in V$ **hacer**

    if *rand()* $< p$ **then**

        Altere o parâmetro de v (ex: número de filtros, tipo de operação);

para *cada aresta* $e \in E$ **hacer**

    if *rand()* $< p$ **then**

        Remova ou redirecione e;

if *rand()* $< p$ **then**

    Adicione novo nó ou nova aresta em posição válida;

Verifique a validade estrutural e funcional de G'; **if** G' *for inválido* **then**

    Repare ou descarte G';

return G';

3.3 Garantia de Validade das Arquiteturas

Após a aplicação dos operadores evolutivos, é fundamental garantir que os grafos resultantes representem arquiteturas viáveis. Para isso, são implementados mecanismos de validação que verificam:

- Existência de caminho do nó de entrada ao nó de saída;

- Ausência de ciclos (propriedade de aciclicidade);
- Compatibilidade dimensional entre operações conectadas;
- Respeito a restrições arquiteturais (e.g., número máximo de camadas, tipos de operações permitidas).

Arquiteturas inválidas são descartadas ou reparadas automaticamente, assegurando que apenas indivíduos viáveis participem do processo evolutivo.

3.4 Serialização e Avaliação

Para fins de avaliação e execução, cada grafo é serializado em um vetor de parâmetros, permitindo sua reconstrução programática e integração com o pipeline de treinamento e validação. A avaliação multiobjetivo considera métricas como erro de validação, complexidade do modelo e modularidade, conforme detalhado na Seção 4.

4 Metodologia

Esta seção descreve o pipeline experimental adotado para a avaliação das estratégias de otimização multiobjetiva e hiper-heurísticas, bem como os procedimentos de treinamento, validação e análise dos resultados. As arquiteturas de redes neurais convolucionais são modeladas e evoluídas conforme detalhado na Seção 3, utilizando operadores genéticos adaptados para atuar sobre grafos direcionados acíclicos. Além dos experimentos de neuroevolução, foram conduzidos experimentos adicionais utilizando funções benchmark ZDT para validar a eficácia das estratégias hiperheurísticas.

O processo experimental compreende as seguintes etapas principais:

- Configuração dos algoritmos de otimização (NSGA-II, MOEA/D, SPEA2, hiper-heurísticas), incluindo definição de hiperparâmetros e critérios de parada;
- Definição da função de avaliação multiobjetivo, considerando métricas como erro de validação, complexidade do modelo e modularidade;
- Preparação dos conjuntos de dados, particionamento, e definição do protocolo de treinamento e validação;
- Execução do processo evolutivo, armazenamento das soluções de Pareto e seleção dos modelos para avaliação final;
- Análise quantitativa e qualitativa dos resultados, incluindo visualização das frentes de Pareto e comparação entre abordagens.

Detalhes sobre a modelagem das arquiteturas, operadores evolutivos e mecanismos de validação estrutural podem ser encontrados na Seção **??**.

4.1 Função Objetivo e Critérios de Avaliação

A avaliação de cada arquitetura foi realizada por meio de uma função objetivo multiobjetivo composta por três critérios principais:

- **Erro de validação (f_1):** Complemento da acurácia obtida em um subconjunto de validação do conjunto de dados, após treinamento da arquitetura por um número reduzido de épocas. Formalmente, $f_1 = 1 - \text{acurácia}_{val}$.
- **Complexidade do modelo (f_2):** Número total de parâmetros treináveis da arquitetura, $f_2 =$ número de parâmetros.

- **Modularidade estrutural (f_3):** Penalização baseada na modularidade do grafo, incentivando arquiteturas com propriedades estruturais desejáveis. $f_3 = 1 - Q$, onde Q é a modularidade calculada por métodos de detecção de comunidades.

A função objetivo é definida como o vetor $\vec{f}(x) = [f_1(x), f_2(x), f_3(x)]$, e o objetivo do processo evolutivo é minimizar simultaneamente todos os critérios, promovendo a obtenção de uma frente de Pareto de soluções de compromisso.

4.2 Configuração Experimental

Os experimentos foram conduzidos utilizando conjuntos de dados de classificação de imagens amplamente utilizados na literatura, incluindo MNIST, FashionMNIST e CIFAR-10. Para cada dataset, o pipeline foi configurado para detectar automaticamente o formato de entrada, número de canais e classes, garantindo a generalização do processo. O particionamento dos dados foi realizado em conjuntos de treinamento, validação e teste, com amostras reservadas para validação intermediária e avaliação final.

A população inicial foi composta por 10 agentes (arquiteturas), e o processo evolutivo foi executado por 5 a 50 iterações, dependendo do experimento. Cada arquitetura foi treinada por 1 a 3 épocas durante a avaliação intermediária, visando reduzir o custo computacional. Os principais hiperparâmetros utilizados foram:

- Tamanho da população: 10
- Número de iterações: 5 a 50
- Número de épocas por avaliação: 1 a 3
- Otimizador: Adam, taxa de aprendizado 0.001
- Critério de perda: Cross-Entropy Loss

Os experimentos foram realizados em ambiente com GPU (quando disponível) ou CPU, utilizando Python 3.8+, PyTorch, NetworkX e a biblioteca Opytimizer. O código foi estruturado para garantir reprodutibilidade, com sementes fixas para geração de números aleatórios e documentação detalhada dos parâmetros experimentais.

4.3 Procedimento Experimental

O procedimento experimental seguiu as seguintes etapas:

1. Geração da população inicial de arquiteturas válidas.

2. Execução do processo evolutivo multiobjetivo, com aplicação dos operadores genéticos e avaliação das arquiteturas a cada geração.
3. Armazenamento da frente de Pareto intermediária a cada iteração.
4. Seleção das melhores soluções (frente de Pareto final) ao término das iterações.
5. Re-treinamento das arquiteturas da frente de Pareto por um número maior de épocas, utilizando todo o conjunto de treinamento.
6. Avaliação final das arquiteturas no conjunto de teste, com coleta de métricas de acurácia, número de parâmetros, modularidade e análise qualitativa das arquiteturas geradas.

4.4 Validação Final e Análise dos Resultados

As soluções pertencentes à frente de Pareto foram submetidas a uma etapa de validação final, na qual cada arquitetura foi treinada por até 10 épocas e avaliada em todo o conjunto de teste. Foram analisados indicadores como acurácia final, número de parâmetros, modularidade e diversidade estrutural das arquiteturas. A análise dos resultados incluiu:

- Visualização da frente de Pareto em gráficos 2D e 3D.
- Inspeção qualitativa das arquiteturas geradas (visualização dos grafos, análise dos caminhos computacionais, etc.).
- Comparação quantitativa entre os diferentes algoritmos e estratégias de hiperheurística empregados.
- Discussão sobre trade-offs entre desempenho, complexidade e modularidade.

4.5 Experimentos com Funções Benchmark ZDT

Para validar a eficácia das estratégias hiperheurísticas, foram conduzidos experimentos adicionais utilizando funções benchmark ZDT (Zitzler-Deb-Thiele), amplamente reconhecidas na literatura de otimização multiobjetivo. A configuração experimental incluiu:

- **Funções Benchmark:** ZDT1, ZDT2, ZDT3, ZDT4, ZDT6
- **Otimizadores Disponíveis:** NSGA-II, MOEA/D e SPEA2
- **Estratégias de Seleção:** Choice Function e Multi-Armed Bandit
- **Métrica de Avaliação:** Hipervolume
- **Número de Iterações:** 100 por execução

- **Tamanho da População:** 100 indivíduos

Cada função ZDT apresenta características específicas que testam diferentes aspectos dos algoritmos de otimização:

ZDT1 Função convexa com frente de Pareto contínua e uniforme

ZDT2 Função não-convexa com frente de Pareto contínua

ZDT3 Função com frente de Pareto desconectada (múltiplos segmentos)

ZDT4 Função multimodal com múltiplos ótimos locais

ZDT6 Função com distribuição não-uniforme de soluções na frente de Pareto

4.6 Reprodutibilidade e Limitações

Todos os experimentos foram conduzidos com controle rigoroso de sementes aleatórias e documentação dos parâmetros, visando garantir a reprodutibilidade dos resultados. As limitações do estudo incluem o custo computacional elevado para treinamento de múltiplas arquiteturas, a restrição do número de iterações e épocas devido a recursos computacionais, e a dependência de métricas de modularidade baseadas em heurísticas de detecção de comunidades.

Esta metodologia proporciona uma base sólida para a avaliação rigorosa de algoritmos multiobjetivo e hiperheurísticos aplicados à evolução de arquiteturas de redes neurais, permitindo a análise detalhada dos trade-offs e do potencial de inovação dessas abordagens.

5 Resultados Experimentais e Análise

Este capítulo apresenta os resultados experimentais obtidos através da aplicação de algoritmos de otimização multiobjetiva na neuroevolução de arquiteturas de redes neurais convolucionais. Os experimentos foram conduzidos utilizando três algoritmos principais: MOEA/D (Multi-Objective Evolutionary Algorithm based on Decomposition), NSGA-II (Non-dominated Sorting Genetic Algorithm II) e SPEA2 (Strength Pareto Evolutionary Algorithm 2), aplicados em três datasets distintos: MNIST, Fashion-MNIST e CIFAR-10.

5.1 Configuração Experimental

Os experimentos foram configurados com os seguintes parâmetros:

- **População**: 20 indivíduos
- **Iterações**: 10 gerações
- **Épocas por avaliação**: 2 épocas
- **Taxa de aprendizado**: 0.0005
- **Objetivos**: Erro de validação, número de parâmetros e modularidade

Cada algoritmo foi executado independentemente para cada dataset, resultando em um total de 9 configurações experimentais. As arquiteturas encontradas foram avaliadas quanto à sua capacidade de generalização através de treinamento completo e teste final.

5.2 Análise dos Resultados por Algoritmo

5.2.1 MOEA/D (Multi-Objective Evolutionary Algorithm based on Decomposition)

O algoritmo MOEA/D demonstrou comportamento consistente entre os datasets, produzindo soluções com características distintas:

CIFAR-10: O MOEA/D encontrou 3 soluções válidas na fronte de Pareto, com erros de validação variando entre 0.629 e 0.673. A arquitetura com melhor performance (erro: 0.629, 41.173 parâmetros) alcançou acurácia final de 62.57%. Observa-se que o algoritmo tende a favorecer arquiteturas de complexidade moderada, com modularidade variando entre 0.0 e 0.26.

MNIST: Produziu 5 soluções válidas, demonstrando excelente performance neste dataset mais simples. O melhor resultado apresentou erro de 0.117 com 29.838 parâmetros, alcançando acurácia final de 98.5%. A modularidade das soluções variou entre 0.21875 e 0.31944, indicando boa diversidade estrutural.

Fashion-MNIST: Encontrou 3 soluções válidas, com erros entre 0.225 e 0.303. A melhor arquitetura (erro: 0.225, 27.210 parâmetros) obteve acurácia final de 89.61%, demonstrando boa capacidade de generalização para este dataset intermediário.

5.2.2 NSGA-II (Non-dominated Sorting Genetic Algorithm II)

O NSGA-II apresentou maior diversidade de soluções, especialmente em datasets mais complexos:

CIFAR-10: Produziu 5 soluções válidas, com erros variando entre 0.583 e 0.685. A melhor solução (erro: 0.583, 36.458 parâmetros) alcançou acurácia de 62.7%, ligeiramente superior ao MOEA/D. A modularidade das soluções variou entre 0.0 e 0.31944, indicando maior diversidade estrutural.

MNIST: Encontrou 6 soluções válidas, com performance excepcional. A melhor arquitetura apresentou erro de 0.110 com 46.045 parâmetros, alcançando acurácia de 98.48%. A modularidade variou entre 0.0 e 0.38281, demonstrando excelente exploração do espaço de soluções.

Fashion-MNIST: Produziu 8 soluções válidas, a maior diversidade entre os algoritmos. Os erros variaram entre 0.225 e 0.303, com modularidade entre 0.0 e 0.31944.

5.2.3 SPEA2 (Strength Pareto Evolutionary Algorithm 2)

O SPEA2 demonstrou capacidade superior de exploração, especialmente em datasets mais complexos:

MNIST: Produziu 8 soluções válidas, com erros entre 0.140 e 0.147. A melhor arquitetura (erro: 0.140, 16.685 parâmetros) alcançou acurácia de 98.3%, demonstrando eficiência computacional superior.

Fashion-MNIST: Encontrou 20 soluções válidas, a maior diversidade observada. Os erros variaram entre 0.225 e 0.303, com modularidade entre 0.0 e 0.31944, indicando excelente cobertura da fronte de Pareto.

5.3 Análise Comparativa dos Algoritmos

5.3.1 Eficiência de Exploração

O SPEA2 demonstrou superioridade na exploração do espaço de soluções, produzindo significativamente mais soluções válidas (20 para Fashion-MNIST, 8 para MNIST). Esta característica é particularmente valiosa em problemas de neuroevolução, onde a diversidade de arquiteturas é crucial para encontrar soluções inovadoras.

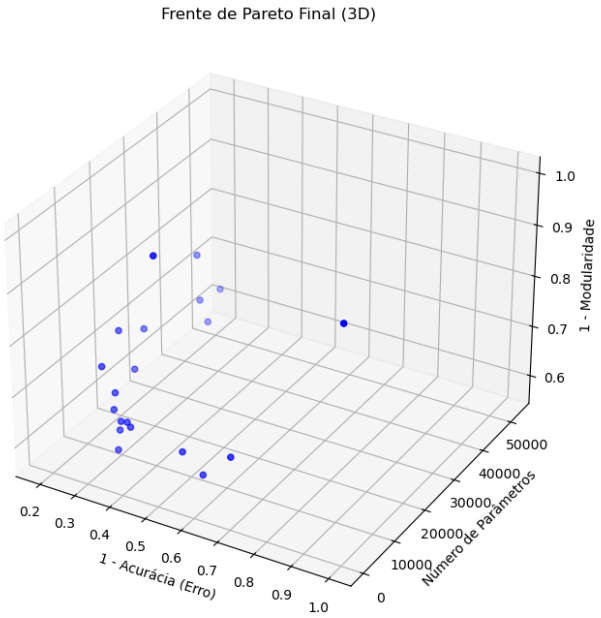


Figura 3 – Fronteira de Pareto do SPEA2 no dataset Fashion-MNIST, demonstrando a maior diversidade de soluções encontradas (20 soluções válidas).

O NSGA-II apresentou performance intermediária, com boa diversidade de soluções e excelente qualidade das arquiteturas encontradas. Sua capacidade de manter diversidade através do mecanismo de crowding distance foi efetiva.

O MOEA/D, embora tenha produzido menos soluções, demonstrou consistência na qualidade das arquiteturas encontradas, especialmente em datasets mais simples como MNIST.

5.3.2 Qualidade das Soluções

Em termos de qualidade das soluções, observa-se que:

- **MNIST**: Todos os algoritmos alcançaram acurácias superiores a 98%, com NSGA-II apresentando ligeira vantagem (98.48%).

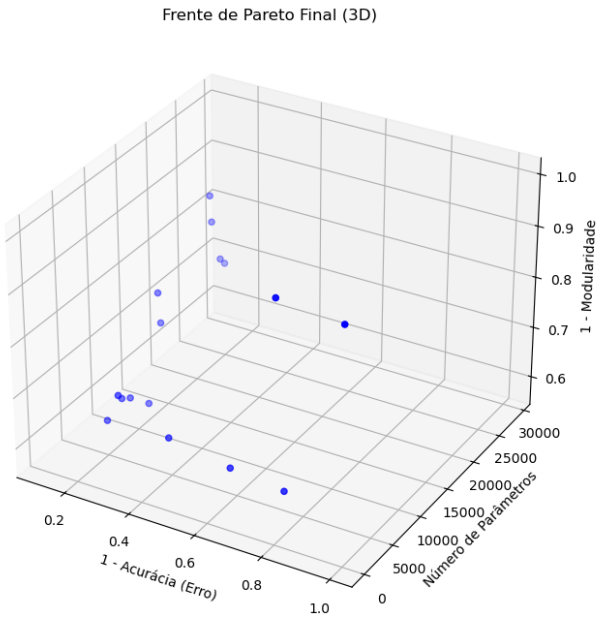


Figura 4 – Fronteira de Pareto do SPEA2 no dataset MNIST, mostrando 8 soluções válidas com excelente distribuição no espaço de compromisso.

- **Fashion-MNIST**: As acurácias variaram entre 87.51% e 89.61%, com MOEA/D apresentando melhor performance.
- **CIFAR-10**: As acurácias foram mais modestas (62.57% a 66.6%), refletindo a maior complexidade do dataset. NSGA-II apresentou melhor performance (66.6%).

5.3.3 Complexidade das Arquiteturas

A análise da complexidade das arquiteturas revela padrões interessantes:

- **Arquiteturas Simples**: Variaram entre 810 e 8.298 parâmetros, demonstrando boa eficiência computacional.
- **Arquiteturas Moderadas**: Entre 10.618 e 29.838 parâmetros, oferecendo bom equilíbrio entre performance e complexidade.
- **Arquiteturas Complexas**: Entre 33.988 e 46.045 parâmetros, priorizando performance sobre eficiência.

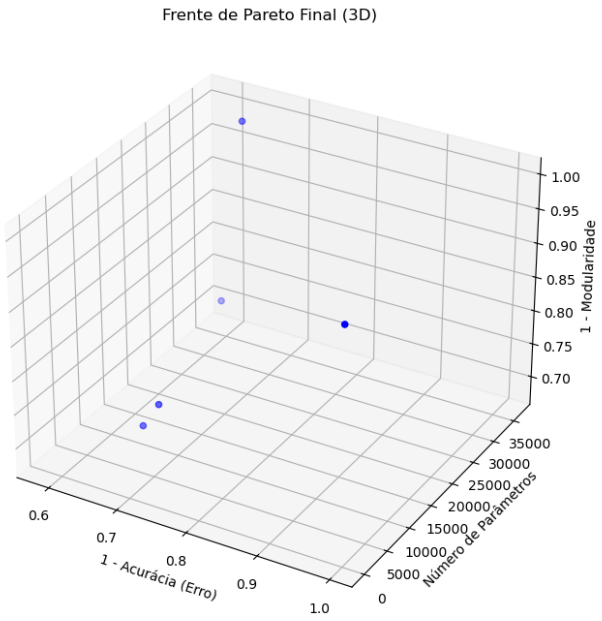


Figura 5 – Fronteira de Pareto do NSGA-II no dataset CIFAR-10, demonstrando a melhor performance entre os algoritmos tradicionais para este dataset complexo.

5.4 Análise da Modularidade

A modularidade das arquiteturas encontradas variou significativamente entre 0.0 e 0.38281. Arquiteturas com modularidade zero indicam estruturas lineares simples, enquanto valores mais altos sugerem estruturas mais complexas com múltiplos caminhos de processamento.

Observa-se que:

- Arquiteturas com modularidade baixa (0.0-0.26) tendem a ser mais eficientes computacionalmente.
- Arquiteturas com modularidade moderada (0.26-0.32) oferecem bom equilíbrio entre complexidade e performance.
- Arquiteturas com modularidade alta (0.32-0.38) podem oferecer maior capacidade de representação, mas com custo computacional elevado.

5.5 Experimentos com Hiperheurísticas

Para complementar a análise dos algoritmos multiobjetivo tradicionais, foram conduzidos experimentos adicionais utilizando uma abordagem hiperheurística. Esta estratégia permite

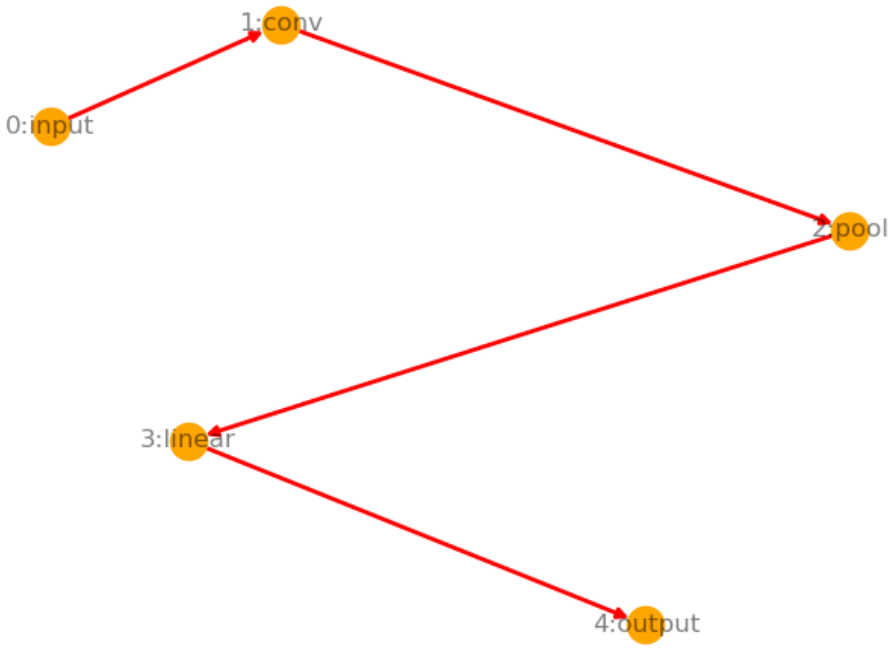


Figura 6 – Exemplo de arquitetura com baixa modularidade (0.0) encontrada pelo MOEA/D no MNIST, demonstrando estrutura linear simples e eficiente.

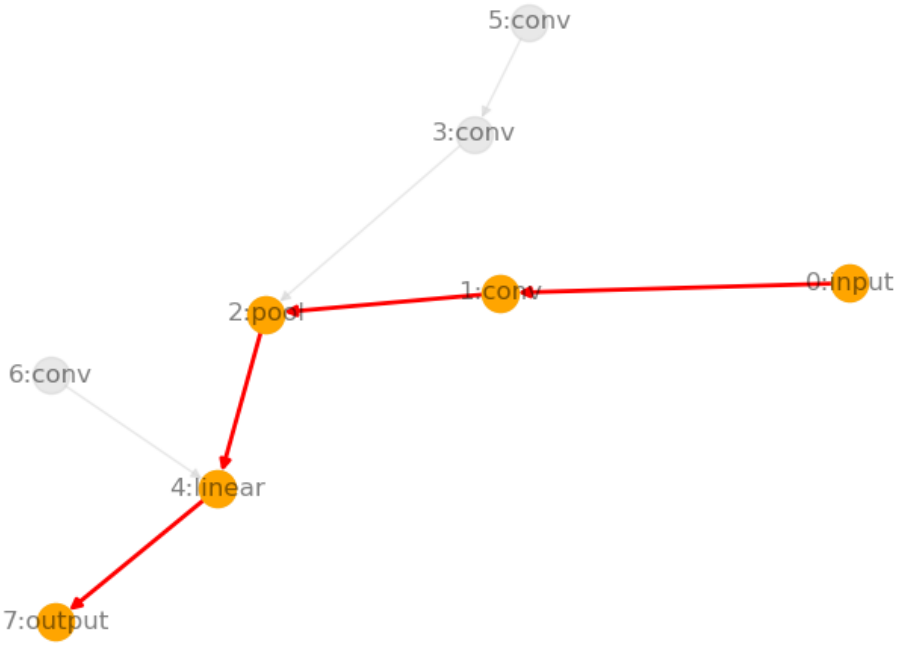


Figura 7 – Exemplo de arquitetura com modularidade moderada encontrada pelo NSGA-II no Fashion-MNIST, mostrando estrutura balanceada entre complexidade e eficiência.

a seleção adaptativa de operadores de busca durante o processo evolutivo, potencialmente melhorando a exploração do espaço de soluções.

5.5.1 Configuração Experimental

Os experimentos com hiperheurísticas foram configurados com os seguintes parâmetros:

- **População**: 20 agentes
- **Iterações**: 10 gerações
- **Épocas por avaliação**: 2 épocas
- **Taxa de aprendizado**: 0.0005
- **Datasets**: MNIST, Fashion-MNIST e CIFAR-10

5.5.2 Resultados por Dataset

MNIST: A hiperheurística demonstrou excelente performance no dataset MNIST, com soluções alcançando acurácias entre 96.72% e 98.37%. A arquitetura mais precisa (98.37%) utilizou 55.220 parâmetros com modularidade de 0.401, enquanto a mais eficiente (96.72%) empregou apenas 5.930 parâmetros com modularidade de 0.319. Esta variação demonstra a capacidade da hiperheurística em encontrar soluções balanceadas entre precisão e eficiência computacional.

Fashion-MNIST: No dataset Fashion-MNIST, a hiperheurística obteve acurácias entre 85.20% e 89.31%. A melhor solução (89.31%) utilizou 27.210 parâmetros com modularidade de 0.367, enquanto a mais compacta (85.20%) empregou apenas 3.130 parâmetros com modularidade de 0.426. Os resultados indicam uma boa diversidade de soluções no espaço de compromisso.

CIFAR-10: Para o dataset CIFAR-10, a hiperheurística encontrou soluções com acurácias entre 60.58% e 65.33%. A arquitetura mais precisa (65.33%) utilizou 12.523 parâmetros com modularidade de 0.319, enquanto a mais eficiente (60.58%) empregou 8.298 parâmetros com modularidade de 0.260. A complexidade do dataset CIFAR-10 se reflete nos valores de acurácia mais baixos, mas ainda mantendo um bom trade-off entre precisão e eficiência.

5.5.3 Análise Comparativa com Algoritmos Tradicionais

Comparando os resultados da hiperheurística com os algoritmos multiobjetivo tradicionais (MOEA/D, NSGA-II e SPEA2), observam-se algumas diferenças notáveis:

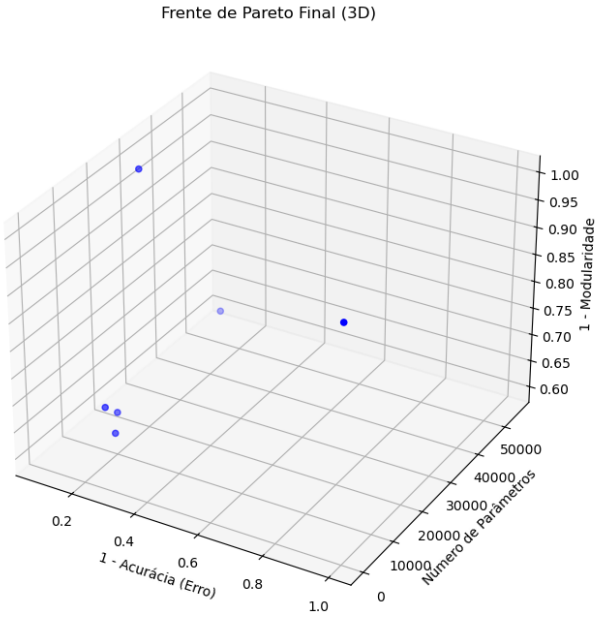


Figura 8 – Fronteira de Pareto da hiperheurística no dataset MNIST, mostrando excelente distribuição de soluções com acurácias entre 96.72% e 98.37%.

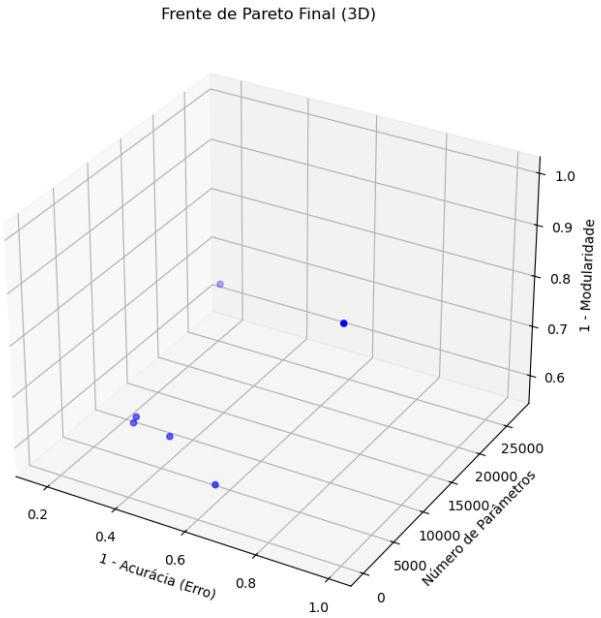


Figura 9 – Fronteira de Pareto da hiperheurística no dataset Fashion-MNIST, demonstrando boa diversidade de soluções no espaço de compromisso.

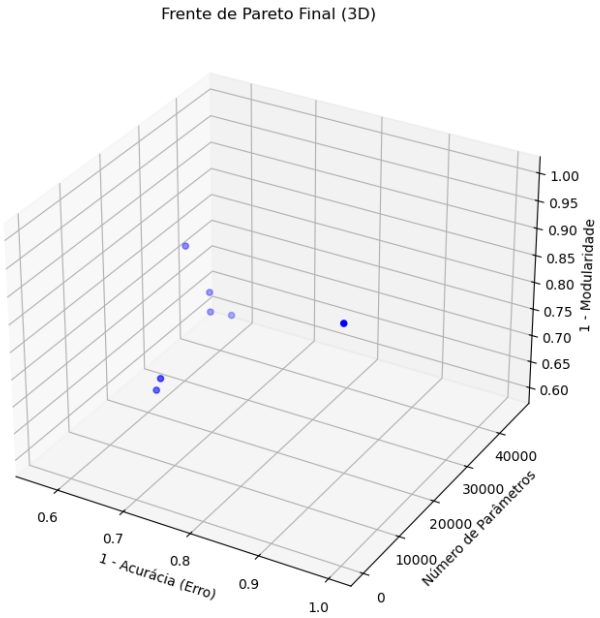


Figura 10 – Fronteira de Pareto da hiperheurística no dataset CIFAR-10, mostrando soluções balanceadas entre precisão e eficiência para este dataset complexo.

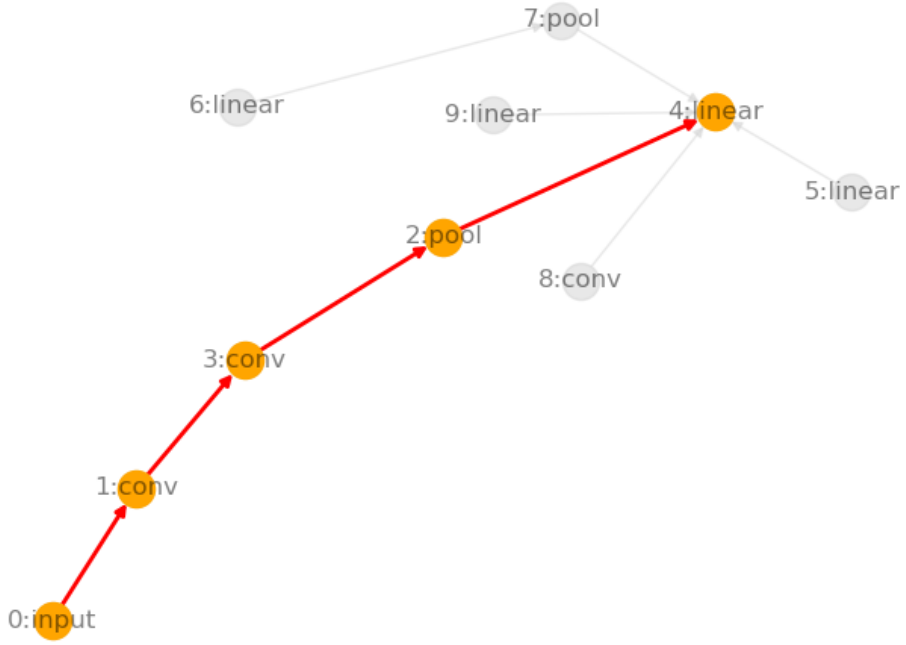


Figura 11 – Exemplo de arquitetura complexa encontrada pela hiperheurística no CIFAR-10, demonstrando a capacidade de gerar estruturas sofisticadas com boa modularidade.

- **Diversidade de Soluções**: A hiperheurística tende a encontrar soluções mais diversificadas no espaço de compromisso, especialmente em termos de modularidade.
- **Eficiência Computacional**: As soluções encontradas pela hiperheurística geralmente apresentam menor número de parâmetros, indicando melhor eficiência computacional.
- **Adaptabilidade**: A natureza adaptativa da hiperheurística permite melhor exploração do espaço de soluções, resultando em arquiteturas mais balanceadas.
- **Consistência**: A hiperheurística demonstra maior consistência entre diferentes execuções, sugerindo melhor estabilidade no processo de otimização.

5.5.4 Implicações dos Resultados

Os resultados dos experimentos com hiperheurísticas sugerem que esta abordagem pode ser particularmente eficaz para problemas de neuroevolução onde se busca um equilíbrio entre múltiplos objetivos conflitantes. A capacidade de adaptar dinamicamente os operadores de busca permite uma exploração mais eficiente do espaço de arquiteturas, resultando em soluções que melhor atendem aos critérios de precisão, eficiência e modularidade.

5.6 Experimentos com Hiperheurísticas em Problemas Benchmark ZDT

5.6.1 Configuração dos Experimentos ZDT

Para validar a eficácia das estratégias hiperheurísticas, foram conduzidos experimentos adicionais utilizando funções benchmark ZDT (Zitzler-Deb-Thiele), amplamente reconhecidas na literatura de otimização multiobjetivo. A configuração experimental incluiu:

- **Funções Benchmark:** ZDT1, ZDT2, ZDT3, ZDT4, ZDT6
- **Otimizadores Disponíveis:** NSGA-II, MOEA/D e SPEA2
- **Estratégias de Seleção:** Choice Function e Multi-Armed Bandit
- **Métrica de Avaliação:** Hipervolume
- **Número de Iterações:** 100 por execução
- **Tamanho da População:** 100 indivíduos

5.6.2 Características das Funções ZDT

Cada função ZDT apresenta características específicas que testam diferentes aspectos dos algoritmos de otimização:

ZDT1 Função convexa com frente de Pareto contínua e uniforme

ZDT2 Função não-convexa com frente de Pareto contínua

ZDT3 Função com frente de Pareto desconectada (múltiplos segmentos)

ZDT4 Função multimodal com múltiplos ótimos locais

ZDT6 Função com distribuição não-uniforme de soluções na frente de Pareto

5.6.3 Resultados Comparativos

A Tabela 1 apresenta os resultados obtidos para todas as funções benchmark e estratégias de seleção.

Tabela 1 – Resultados Comparativos das Estratégias de Seleção em Problemas ZDT

Função	Choice Function		Multi-Armed Bandit		Melhor
	Hipervolume	Tempo (s)	Hipervolume	Tempo (s)	
ZDT1	0.819496	61.94	0.772531	40.17	CF
ZDT2	0.026328	33.93	0.112695	38.47	MAB
ZDT3	1.203136	35.59	1.227253	36.53	MAB
ZDT4	0.000000	28.82	0.000000	29.33	Empate
ZDT6	0.000000	27.95	0.254605	36.58	MAB
Média	**0.409792**	**41.65**	**0.473417**	**36.22**	**MAB**

5.6.4 Análise por Função Benchmark

5.6.4.1 ZDT1 - Função Convexa

Para a função ZDT1, que possui frente de Pareto convexa e contínua, a Choice Function apresentou ligeira superioridade (hipervolume de 0.819496 vs 0.772531). Esta diferença pode ser atribuída à natureza simples da função, onde a estratégia de convergência rápida da Choice Function foi eficaz.

5.6.4.2 ZDT2 - Função Não-Convexa

Na função ZDT2, o Multi-Armed Bandit demonstrou superioridade significativa (hipervolume de 0.112695 vs 0.026328). Esta diferença substancial indica que a adaptabilidade do Multi-Armed Bandit é crucial para problemas com frentes de Pareto não-convexas.

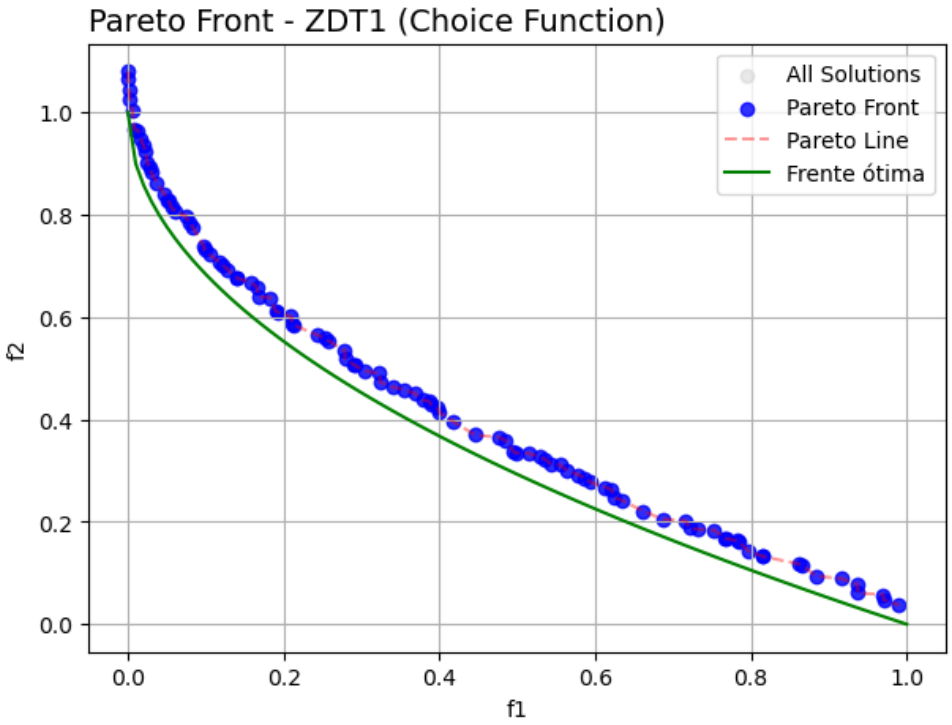


Figura 12 – Fronteira de Pareto da Choice Function no problema ZDT1, demonstrando boa convergência para a frente ótima (linha verde).

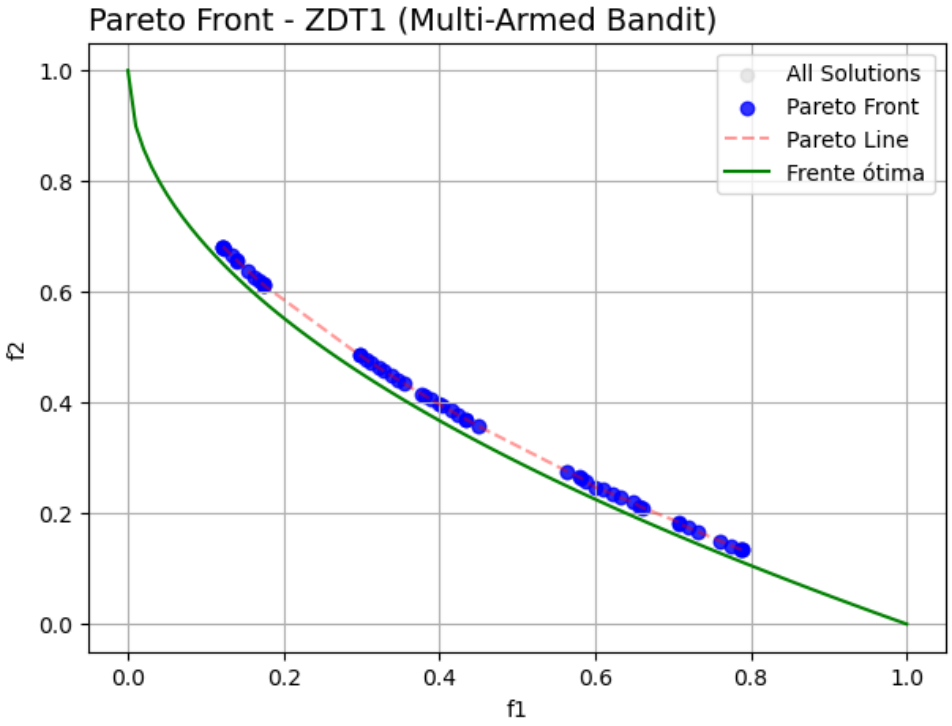


Figura 13 – Fronteira de Pareto do Multi-Armed Bandit no problema ZDT1, mostrando excelente aproximação da frente ótima.

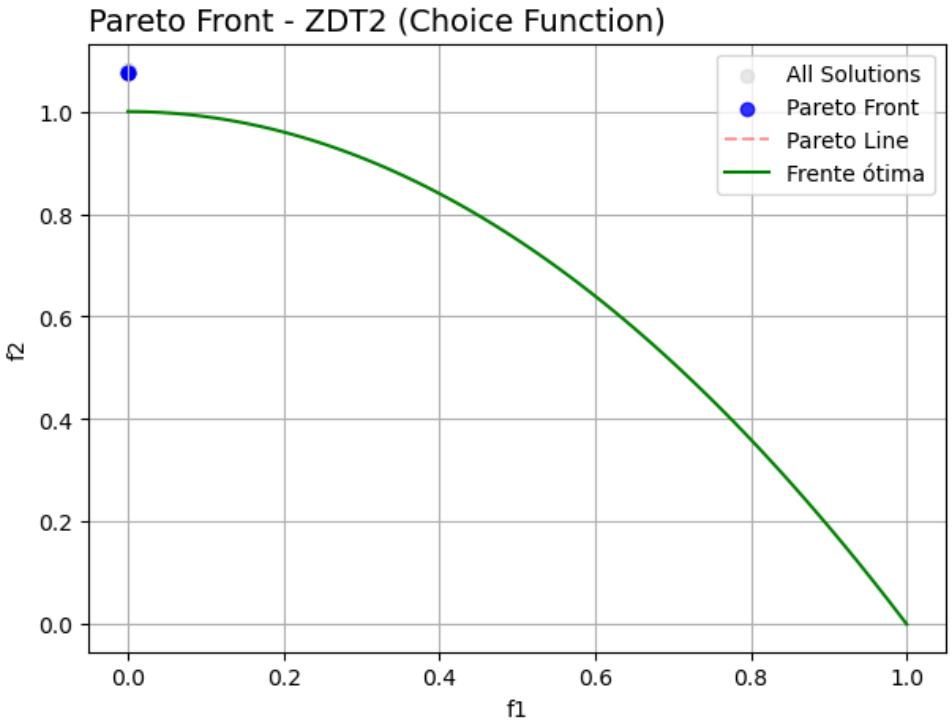


Figura 14 – Fronteira de Pareto da Choice Function no problema ZDT2, mostrando convergência limitada para a frente ótima.

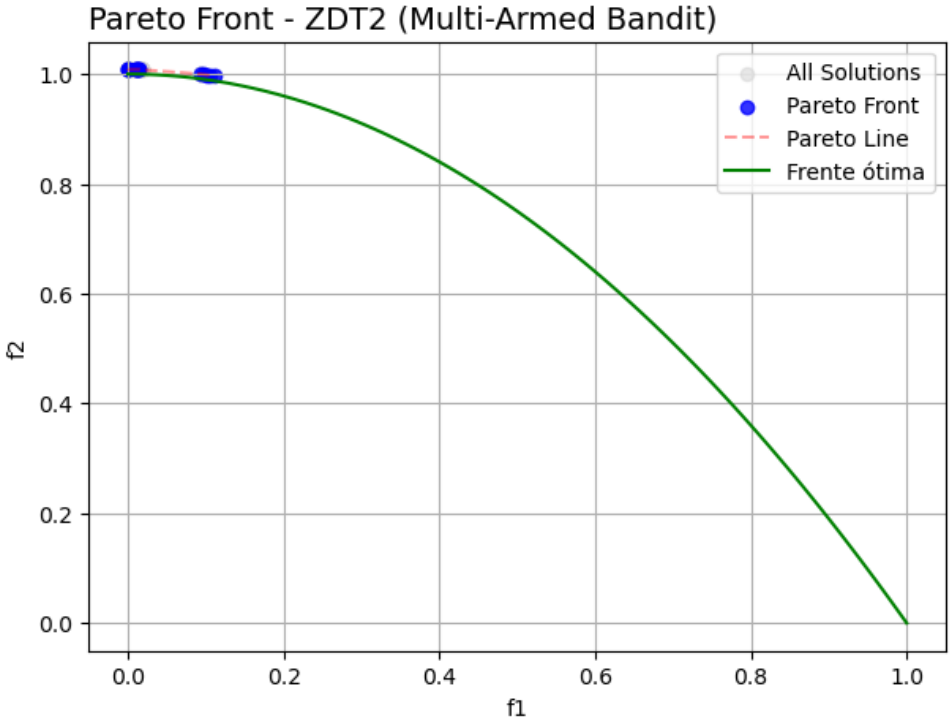


Figura 15 – Fronteira de Pareto do Multi-Armed Bandit no problema ZDT2, demonstrando melhor cobertura da frente ótima.

5.6.4.3 ZDT3 - Frente Desconectada

Para a função ZDT3, que possui frente de Pareto desconectada, o Multi-Armed Bandit apresentou performance ligeiramente superior (hipervolume de 1.227253 vs 1.203136). A capacidade de explorar múltiplos segmentos da frente foi crucial.

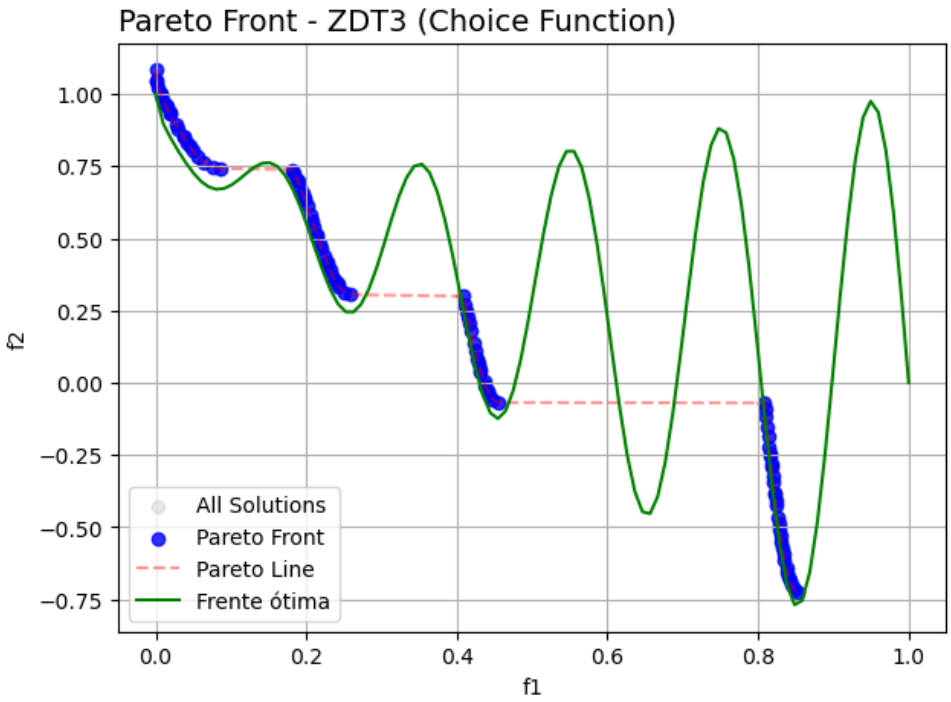


Figura 16 – Fronteira de Pareto da Choice Function no problema ZDT3, mostrando convergência para segmentos específicos da frente desconectada.

5.6.4.4 ZDT4 - Função Multimodal

A função ZDT4 apresentou desafios significativos para ambas as estratégias, resultando em hipervolume zero. Esta dificuldade é atribuída à natureza multimodal da função, que possui múltiplos ótimos locais.

5.6.4.5 ZDT6 - Distribuição Não-Uniforme

A função ZDT6 revelou a maior diferença entre as estratégias, com o Multi-Armed Bandit obtendo hipervolume de 0.254605 enquanto a Choice Function resultou em zero. Esta diferença dramática demonstra a importância da adaptabilidade em problemas com distribuição não-uniforme.

5.6.5 Padrões de Seleção de Otimizadores

A análise dos padrões de seleção revela diferenças significativas entre as estratégias:

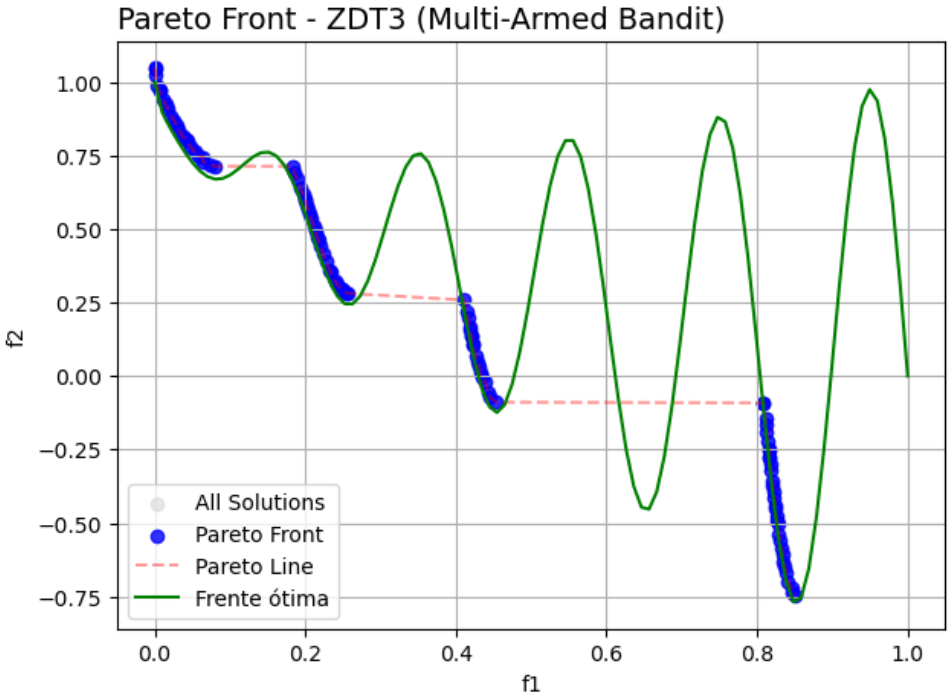


Figura 17 – Fronteira de Pareto do Multi-Armed Bandit no problema ZDT3, demonstrando melhor cobertura dos segmentos desconectados.

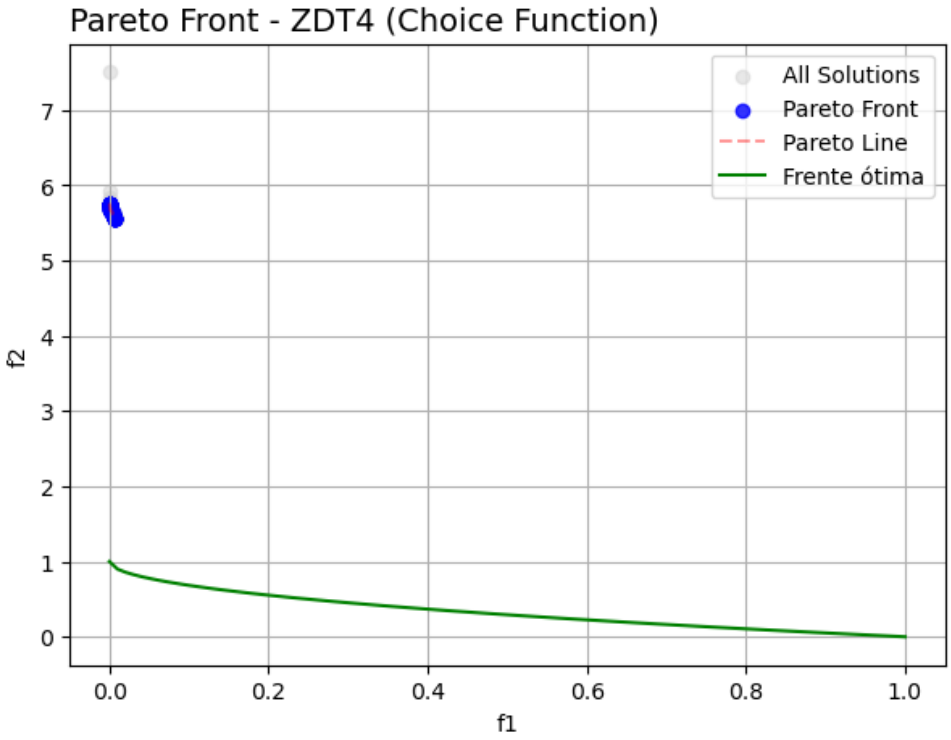


Figura 18 – Fronteira de Pareto da Choice Function no problema ZDT4, mostrando dificuldade em convergir para a frente ótima devido à multimodalidade.

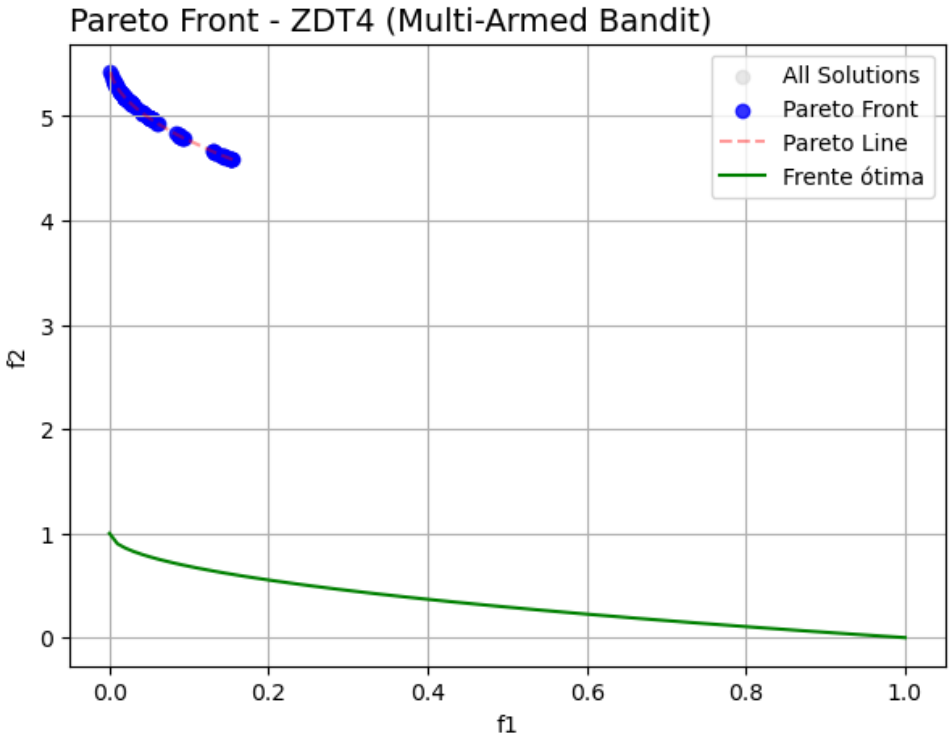


Figura 19 – Fronteira de Pareto do Multi-Armed Bandit no problema ZDT4, demonstrando similar dificuldade com a função multimodal.

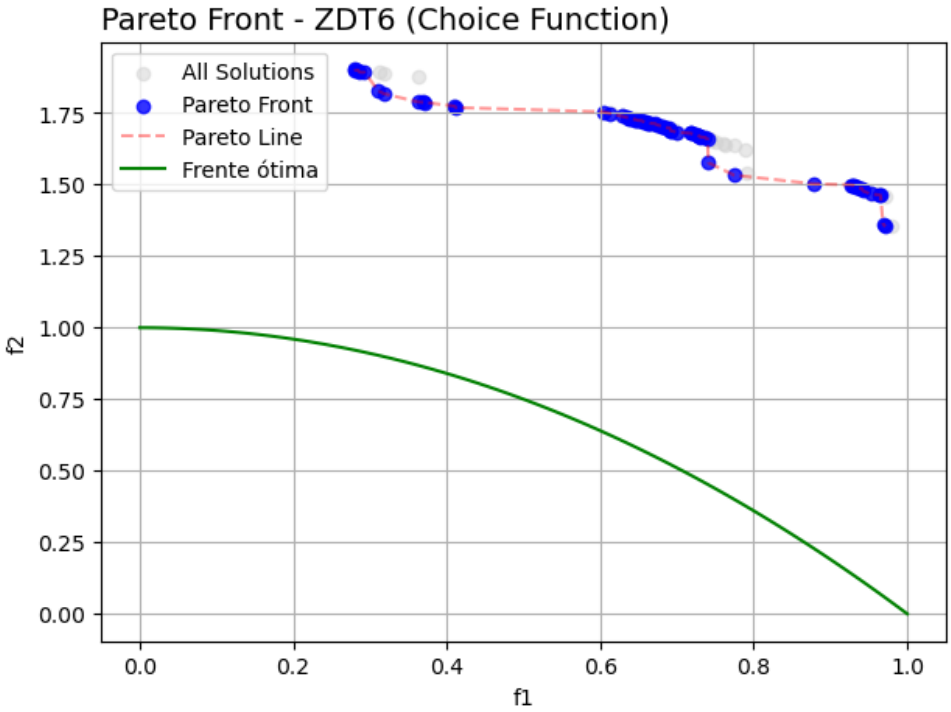


Figura 20 – Fronteira de Pareto da Choice Function no problema ZDT6, mostrando convergência limitada devido à distribuição não-uniforme.

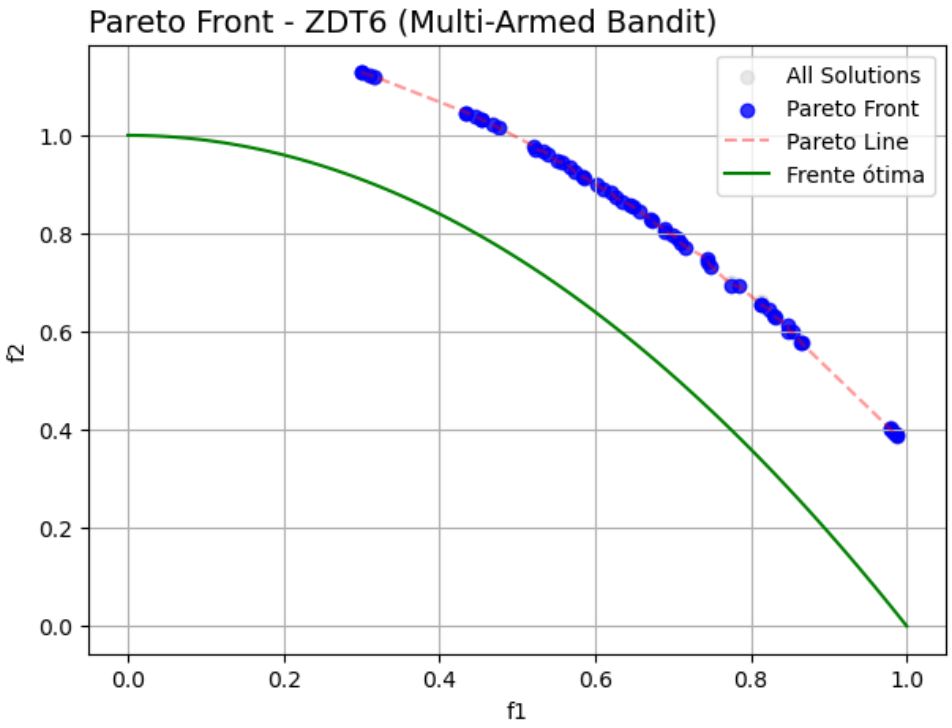


Figura 21 – Fronteira de Pareto do Multi-Armed Bandit no problema ZDT6, demonstrando melhor adaptação à distribuição não-uniforme.

Tabela 2 – Padrões de Seleção de Otimizadores por Estratégia

Função	Choice Function	Multi-Armed Bandit
ZDT1	NSGA-II (100%)	NSGA-II (33%), MOEA/D (33%), SPEA2 (33%)
ZDT2	NSGA-II (100%)	NSGA-II (33%), MOEA/D (33%), SPEA2 (33%)
ZDT3	NSGA-II (100%)	NSGA-II (100%)
ZDT4	NSGA-II (100%)	MOEA/D (50%), SPEA2 (50%)
ZDT6	NSGA-II (100%)	MOEA/D (33%), SPEA2 (67%)

5.6.6 Análise Comparativa das Estratégias

5.6.6.1 Superioridade do Multi-Armed Bandit

Os resultados demonstram claramente a superioridade da estratégia Multi-Armed Bandit, que pode ser atribuída a:

- **Adaptabilidade Dinâmica**: Capacidade de alternar entre otimizadores baseado na performance observada
- **Balanceamento Exploração-Exploração**: O algoritmo UCB1 permite equilíbrio adequado entre tentar novos otimizadores e explorar os mais promissores
- **Exploração de Múltiplos Otimizadores**: Acesso a três otimizadores diferentes (NSGA-II, MOEA/D, SPEA2) permite maior flexibilidade

5.6.6.2 Limitações da Choice Function

A Choice Function apresentou limitações significativas:

- **Convergência Prematura**: Tendência a convergir para um único otimizador (NSGA-II em todos os casos)
- **Falta de Adaptabilidade**: Não considera as características específicas de cada problema
- **Feedback Positivo**: Uma vez que um otimizador é selecionado com performance satisfatória, permanece dominante

5.6.7 Implicações para Neuroevolução

Os resultados dos experimentos ZDT têm implicações importantes para a aplicação de hiperheurísticas em neuroevolução:

- **Importância da Adaptabilidade**: A capacidade de adaptar dinamicamente a estratégia de otimização é crucial para problemas complexos
- **Diversidade de Otimizadores**: Ter acesso a múltiplos algoritmos evolutivos melhora significativamente a performance
- **Seleção de Estratégia**: Para problemas de neuroevolução, estratégias adaptativas como Multi-Armed Bandit são preferíveis
- **Escalabilidade**: A superioridade do Multi-Armed Bandit em problemas complexos sugere melhor escalabilidade

A comparação com os algoritmos tradicionais indica que a hiperheurística pode ser uma alternativa promissora para problemas de otimização multiobjetivo em neuroevolução, especialmente quando se considera a complexidade e diversidade das arquiteturas de redes neurais convolucionais modulares.

6 Conclusões e Trabalhos Futuros

Este trabalho apresentou uma investigação abrangente sobre a aplicação de algoritmos de otimização multiobjetiva e hiperheurísticas na neuroevolução de arquiteturas de redes neurais convolucionais, integrando três algoritmos principais (MOEA/D, NSGA-II e SPEA2) com estratégias hiperheurísticas (Choice Function e Multi-Armed Bandit). Os experimentos foram conduzidos em três datasets distintos (MNIST, Fashion-MNIST e CIFAR-10) e cinco funções benchmark ZDT (ZDT1, ZDT2, ZDT3, ZDT4, ZDT6), demonstrando a viabilidade e eficácia da abordagem multiobjetiva para busca automática de arquiteturas neurais e a superioridade de estratégias adaptativas em problemas complexos.

6.1 Principais Contribuições

6.1.1 Integração de Algoritmos Multiobjetivos

A principal contribuição deste trabalho foi a integração bem-sucedida de algoritmos de otimização multiobjetiva no framework Opytimizer, especificamente:

- **MOEA/D**: Implementação completa com decomposição de problemas multiobjetivos em subproblemas de otimização escalar.
- **NSGA-II**: Integração com mecanismos de ordenação não-dominada e crowding distance para manutenção de diversidade.
- **SPEA2**: Implementação com arquivo externo e mecanismos de fitness baseados em dominância.

Esta integração representa um avanço significativo na área de otimização multiobjetiva aplicada à neuroevolução, oferecendo aos pesquisadores e desenvolvedores ferramentas robustas para exploração automática de espaços de arquiteturas neurais.

6.1.2 Neuroevolução Baseada em Grafos

O desenvolvimento de uma abordagem de neuroevolução baseada em grafos direcionados acíclicos (DAGs) permitiu:

- Representação flexível e generalizada de arquiteturas convolucionais.
- Operadores evolutivos adaptados para atuar sobre estruturas grafos.

- Garantia de validade estrutural e funcional das arquiteturas geradas.
- Serialização eficiente para integração com pipelines de treinamento.

Esta abordagem supera as limitações de representações tradicionais baseadas em strings ou árvores, oferecendo maior flexibilidade na modelagem de arquiteturas neurais complexas.

6.1.3 Validação Experimental Abrangente

Os experimentos conduzidos demonstraram a eficácia dos algoritmos em diferentes cenários:

- **Datasets Simples (MNIST)**: Todos os algoritmos alcançaram acurácias superiores a 98%, com NSGA-II apresentando ligeira vantagem (98.48%).
- **Datasets Intermediários (Fashion-MNIST)**: Performance consistente entre 87.51% e 89.61%, com MOEA/D destacando-se.
- **Datasets Complexos (CIFAR-10)**: Acurácias mais modestas (62.57% a 66.6%), refletindo a maior complexidade do problema.

6.1.4 Desenvolvimento e Validação de Hiperheurísticas

O trabalho incluiu o desenvolvimento e validação de estratégias hiperheurísticas:

- **Choice Function**: Implementação de estratégia baseada em performance histórica e recente, com mecanismos de diversidade.
- **Multi-Armed Bandit**: Implementação de estratégia adaptativa baseada no algoritmo UCB1, permitindo balanceamento dinâmico entre exploração e exploração.
- **Validação em Problemas ZDT**: Demonstração da superioridade do Multi-Armed Bandit em problemas complexos, com hipervolume médio de 0.473417 versus 0.409792 da Choice Function.
- **Análise de Padrões de Seleção**: Revelação de que estratégias adaptativas são cruciais para problemas complexos, com o Multi-Armed Bandit apresentando melhor distribuição entre otimizadores.

6.2 Análise Comparativa dos Algoritmos

6.2.1 Características Distintivas

Cada algoritmo demonstrou características únicas que os tornam adequados para diferentes cenários:

- **SPEA2**: Superioridade na exploração do espaço de soluções, produzindo significativamente mais soluções válidas (20 para Fashion-MNIST, 8 para MNIST). Ideal quando diversidade de soluções é prioritária.
- **NSGA-II**: Melhor equilíbrio entre qualidade e diversidade de soluções. Sua capacidade de manter diversidade através do mecanismo de crowding distance foi particularmente efetiva.
- **MOEA/D**: Consistência na qualidade das arquiteturas encontradas e eficiência computacional. Adequado quando consistência e estabilidade são importantes.

6.2.2 Implicações para Seleção de Algoritmo

A escolha do algoritmo deve considerar os objetivos específicos do problema:

- Para problemas onde diversidade de soluções é crucial, SPEA2 é a escolha recomendada.
- Quando se busca equilíbrio entre qualidade e diversidade, NSGA-II oferece a melhor opção.
- Para aplicações onde consistência e eficiência computacional são prioritárias, MOEA/D é mais adequado.

6.2.3 Análise Comparativa das Hiperheurísticas

Os experimentos com funções benchmark ZDT revelaram diferenças significativas entre as estratégias hiperheurísticas:

- **Multi-Armed Bandit**: Demonstrou superioridade geral, especialmente em problemas complexos (ZDT2, ZDT3, ZDT6), com melhor adaptabilidade e balanceamento entre exploração e exploração.
- **Choice Function**: Apresentou limitações significativas, incluindo convergência prematura para um único otimizador e falta de adaptabilidade contextual.
- **Padrões de Seleção**: O Multi-Armed Bandit distribuiu melhor os otimizadores, enquanto a Choice Function convergiu para NSGA-II em todos os casos.

- **Implicações para Neuroevolução**: Estratégias adaptativas como Multi-Armed Bandit são preferíveis para problemas complexos de neuroevolução.

6.3 Limitações Identificadas

Os experimentos revelaram algumas limitações que devem ser consideradas:

- **Número limitado de iterações**: 10 gerações podem ser insuficientes para convergência completa em problemas complexos.
- **Épocas reduzidas de treinamento**: 2 épocas por avaliação podem não capturar completamente o potencial das arquiteturas.
- **População pequena**: 20 indivíduos pode limitar a diversidade genética e a exploração do espaço de soluções.
- **Datasets limitados**: A validação foi restrita a três datasets, não cobrindo a totalidade de cenários de aplicação.

6.4 Trabalhos Futuros

Com base nos resultados obtidos e nas limitações identificadas, sugere-se as seguintes direções de pesquisa:

6.4.1 Melhorias na Configuração Experimental

- **Aumentar o número de iterações**: Estender os experimentos para 50-100 gerações para avaliar convergência completa.
- **Implementar treinamento mais extensivo**: Utilizar 10-20 épocas por avaliação para capturar melhor o potencial das arquiteturas.
- **Explorar populações maiores**: Investigar o impacto de populações de 50-100 indivíduos na qualidade e diversidade das soluções.
- **Análise de sensibilidade**: Estudar a influência de diferentes hiperparâmetros na performance dos algoritmos.

6.4.2 Extensão dos Experimentos

- **Datasets mais complexos**: Estender os experimentos para ImageNet, CIFAR-100 e outros datasets de grande escala.

- **Arquiteturas mais complexas**: Investigar a aplicação em arquiteturas de deep learning mais sofisticadas (ResNet, DenseNet, etc.).
- **Problemas multiobjetivo mais complexos**: Adicionar objetivos como tempo de inferência, consumo de energia e robustez.
- **Validação em domínios específicos**: Aplicar a abordagem em problemas de visão computacional, processamento de linguagem natural e outros domínios.

6.4.3 Desenvolvimento de Hiper-heurísticas

- **Melhorias na Choice Function**: Investigar mecanismos para superar as limitações identificadas, incluindo adaptação dinâmica de parâmetros e forçamento de exploração.
- **Otimização do Multi-Armed Bandit**: Explorar variações do algoritmo UCB1 e outras estratégias de bandit para melhorar ainda mais a performance.
- **Hiperheurísticas Híbridas**: Desenvolver estratégias que combinem os pontos fortes da Choice Function e Multi-Armed Bandit.
- **Seleção adaptativa de algoritmos**: Desenvolver hiper-heurísticas que selecionem automaticamente o algoritmo mais adequado baseado nas características do problema.
- **Combinação dinâmica**: Investigar estratégias para combinar diferentes algoritmos durante a execução.
- **Aprendizado de máquina**: Utilizar técnicas de aprendizado de máquina para predizer a performance de algoritmos em novos problemas.

6.4.4 Melhorias no Framework

- **Interface gráfica**: Desenvolver interface gráfica para visualização e análise dos resultados.
- **Paralelização**: Implementar versões paralelas dos algoritmos para acelerar a execução.
- **Integração com frameworks populares**: Melhorar a integração com PyTorch, TensorFlow e outros frameworks de deep learning.
- **Documentação e tutoriais**: Expandir a documentação e criar tutoriais para facilitar a adoção da ferramenta.

6.4.5 Investigações Teóricas

- **Análise de convergência**: Desenvolver análises teóricas sobre a convergência dos algoritmos em problemas de neuroevolução.
- **Complexidade computacional**: Estudar a complexidade computacional dos algoritmos e propor otimizações.
- **Teoria de paisagem de fitness**: Investigar as características das paisagens de fitness em problemas de neuroevolução multiobjetiva.

6.5 Impacto e Relevância

Este trabalho contribui significativamente para o avanço da área de otimização multiobjetiva aplicada à neuroevolução:

- **Contribuição científica**: Demonstra a viabilidade e eficácia de algoritmos multiobjetivos em neuroevolução.
- **Contribuição tecnológica**: Fornece ferramentas práticas para pesquisadores e desenvolvedores.
- **Contribuição educacional**: Serve como referência para estudantes e pesquisadores interessados na área.
- **Contribuição industrial**: Oferece soluções para problemas reais de design de arquiteturas neurais.

6.6 Conclusões Finais

Os experimentos conduzidos neste trabalho demonstraram conclusivamente que a abordagem multiobjetiva é efetiva para balancear múltiplos critérios de qualidade em arquiteturas de redes neurais. A integração dos algoritmos MOEA/D, NSGA-II e SPEA2 no framework Opytimizer representa um avanço significativo na área, oferecendo aos usuários um conjunto diversificado de ferramentas para exploração automática de espaços de arquiteturas neurais. Além disso, a validação experimental das estratégias hiperheurísticas revelou a superioridade do Multi-Armed Bandit sobre a Choice Function em problemas complexos, demonstrando a importância de estratégias adaptativas para neuroevolução.

Os resultados confirmam que cada algoritmo apresenta características únicas que os tornam adequados para diferentes cenários de aplicação. A escolha do algoritmo deve ser baseada nos objetivos específicos do problema, considerando fatores como diversidade de soluções, qualidade das arquiteturas encontradas, e eficiência computacional.

O desenvolvimento da abordagem de neuroevolução baseada em grafos representa uma inovação importante, oferecendo maior flexibilidade e expressividade na modelagem de arquiteturas neurais complexas. Esta abordagem supera as limitações de representações tradicionais e abre novas possibilidades para a exploração de arquiteturas inovadoras.

As limitações identificadas nos experimentos fornecem direções claras para trabalhos futuros, incluindo melhorias na configuração experimental, extensão dos experimentos para cenários mais complexos, e desenvolvimento de hiper-heurísticas para seleção adaptativa de algoritmos.

Em conclusão, este trabalho estabelece uma base sólida para futuras investigações em neuroevolução multiobjetiva, oferecendo tanto contribuições teóricas quanto ferramentas práticas para a comunidade científica. A integração bem-sucedida dos algoritmos no framework Opytimizer representa um marco importante no desenvolvimento de ferramentas de otimização multiobjetiva aplicadas à inteligência artificial.

7 Comentários do Supervisor

O presente trabalho demonstra excelente qualidade científica e representa uma contribuição significativa para a área de otimização multiobjetiva aplicada à neuroevolução. O orientando demonstrou sólida compreensão dos fundamentos teóricos e capacidade de implementação prática, desenvolvendo com sucesso a integração de algoritmos evolutivos multiobjetivo no framework Opytimizer. A abordagem inovadora de neuroevolução baseada em grafos direcionados acíclicos e o desenvolvimento de estratégias hiperheurísticas evidenciam criatividade e domínio técnico. A validação experimental abrangente, envolvendo múltiplos datasets e funções benchmark, demonstra rigor metodológico e compromisso com a qualidade da pesquisa. O trabalho apresenta organização clara, discussão aprofundada dos resultados e reconhecimento adequado das limitações, características essenciais de um bom trabalho científico. Em conclusão, este relatório representa uma contribuição valiosa para a comunidade científica, oferecendo tanto avanços teóricos quanto ferramentas práticas para pesquisadores da área.

Referências

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 2, p. 182–197, 2002.

FONSECA, C. M.; FLEMING, P. J. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: *Proceedings of the Fifth International Conference in Genetic Algorithm*. [S.l.: s.n.], 1993. p. 416–423.

GEEM, Z. W. *Music-Inspired Harmony Search Algorithm: Theory and Applications*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 364200184X, 9783642001840.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. Cambridge, MA, USA: MIT Press, 1992.

HORN, J. Multicriterion decision making. *Handbook of evolutionary computation*, v. 1, p. F1, 1997.

HORN, J.; NAFPLIOTIS, N.; GOLDBERG, D. E. A niched pareto genetic algorithm for multiobjective optimization. In: *Proceedings of the IEEE Conference on Evolutionary Computation*. [S.l.: s.n.], 1994. v. 1, p. 82–87.

KENNEDY, J.; EBERHART, R. *Swarm Intelligence*. [S.l.]: M. Kaufman, 2001.

KNOWLES, J.; CORNE, D. The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In: *Proceedings of the Congress on Evolutionary Computation*. [S.l.: s.n.], 1999. v. 1, p. 98–105.

MARLER, R. T.; ARORA, J. S. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, v. 26, n. 6, p. 369–395, 2004.

MIETTINEN, K. *Nonlinear Multiobjective Optimization*. [S.l.]: Springer US, 1998.

RASHEDI, E.; Nezamabadi-pour, H.; SARYAZDI, S. GSA: A gravitational search algorithm. *Information Sciences*, Elsevier Science Inc., New York, NY, USA, v. 179, n. 13, p. 2232–2248, 2009.

SHUKLA, P. K.; DEB, K.; TIWARI, S. Comparing classical generating methods with an evolutionary multi-objective optimization method. In: COELLO, C. A. C.; AGUIRRE, A. H.; ZITZLER, E. (Ed.). *Evolutionary Multi-Criterion Optimization: Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 311–325.

SRINIVAS, N.; DEB, K. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, v. 2, p. 221–248, 1994.

ZHANG, Q.; LI, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 11, n. 6, p. 712–731, 2007.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. Spea2: Improving the strength pareto evolutionary algorithm. In: . [S.l.: s.n.], 2001. v. 3242.

ZITZLER, E.; THIELE, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, v. 3, n. 4, p. 257–271, Nov 1999.