UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA QUÍMICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

ASSOCIAÇÃO DE INTELIGÊNCIA DE ENXAME COM APRENDIZADO DE MÁQUINA: UMA NOVA ABORGAGEM PARA MÉTODOS DE OTIMIZAÇÃO

Raiana Roland Seixas

Uberlândia – MG

2021

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA QUÍMICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA

ASSOCIAÇÃO DE INTELIGÊNCIA DE ENXAME COM REDE NEURAL ARTIFICIAL: UMA NOVA CLASSE DE MÉTODOS DE OTIMIZAÇÃO

Raiana Roland Seixas

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Química da Universidade Federal de Uberlândia como parte dos requisitos necessários à obtenção do título de Doutora em Engenharia Química.

Uberlândia – MG

2021

TESE DE DOUTORADO SUBMETIDA AO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA QUÍMICA DA UNIVERSIDADE FEDERAL DE UBERLÂNDIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO TÍTULO DE DOUTOR EM ENGENHARIA QUÍMICA, EM 29 DE OUTUBRO DE 2021.

Na prática a teoria é outra!

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

NOMENCLATURA

Sigla	Definição
ABC	*Artificial Bee Colony*
ACO	*Ant Colony Optimization*
Avg	Média
CR	Taxa de Crossover do método DE
D	Dimensão
DE	*Differential Evolution*
Ens	Ensaio
F	Fator de Mutação do método DE
FA	*Firefly Algorithm*
FRANNK	*Full Referenced Artificial Neural Network Knowledge*
freq	Frequência
Gen	Gerações ou iterações
IA	Inteligência Artificial
JADE	DE/current-to-pbest com arquivo externo facultativo
jDE	*Differential Evolution* Auto-adaptativo
M	Multimodal
MC	Método Colaborativo
Mét	Método
ML	*Machine Learning*
NS	Não Separável
PCA	Análise de Componentes Principais
Pop	População
Prop	Proporção
PSO	*Particle Swarm Optimization*
RF	*Random Forest*
RL	Regressão Logística
RNA	Rede Neural Artificial
RNA-CR	Rede Neural Artificial associada ao ajuste do parâmetro CR
RNA-F	Rede Neural Artificial associada ao ajuste do parâmetro F
S	Separável
Sdt	Desvio padrão

SI	*Swarm Intelligence*
SVM	*Support Vector Machine*
TAC	Custo anual total
U	Unimodal

RESUMO

Este trabalho apresenta o desenvolvimento do otimizador FRANNK (*Full Referenced Artificial Neural Network Knowledge*), um método de solução para problemas de otimização baseado em Inteligência de Enxame (SI) acoplado a uma Rede Neural Artificial (RNA) pré-treinada para realizar um ajuste dinâmico dos parâmetros do método. Desenvolveu-se nessa pesquisa, uma versão do Otimizador FRANNK baseada em Evolução Diferencial (DE), a qual ajusta o Fator de Mutação (*F*) e a Taxa de Crossover (*CR*) ao longo das iterações usando uma RNA de classificação para definir a direção da alteração do parâmetro, somada a um conjunto de equações para definir a magnitude do ajuste. Em termos de resultados, observou-se que a diversidade da população e a fração total das iterações impactaram fortemente o ajuste dos parâmetros, assim como os valores de *F* e *CR* da iteração anterior. A comparação de desempenho entre o otimizador FRANNK e outros otimizadores (DE, jDE e JADE) foi realizada através da otimização de diversas funções benchmark e indicou que a versão proposta do otimizador FRANNK é eficiente e tem uma boa capacidade de busca global na resolução de problemas. A eficiência do otimizador FRANNK foi igualmente avaliada na solução de um problema de integração energética, no qual buscou-se minimizar o gasto anual total (relativo a investimentos e custos com utilidades) associado a uma rede de trocadores de calor. Adicionalmente, desenvolveu-se um novo tipo de estratégia de solução para problemas de otimização denominada Método Colaborativo, no qual uma parcela da população trabalha com ajuste dos parâmetros *CR* e *F* seguindo a estratégia de RNA e a outra parcela trabalha com os parâmetros *CR* e *F* fixos em diferentes valores para diferentes subpopulações. Nesta estratégia, as melhores soluções são compartilhadas entre as subpopulações a cada iteração, o que promove um efeito sinergético durante a otimização. A utilização de subpopulações constituída de 1/3 dos indivíduos realizando o ajuste dinâmico dos parâmetros *CR*-*F* pelo método FRANNK e 2/3 da população subdividida em outras 3 partes, as quais usaram valores para o par *CR*- *F* fixados em, respectivamente, 0,5-0,5, 0,25-0,75 e 0,25-0,25, mostrou ótimos resultados, sendo que o método resultante, denominado C-FRANNK obteve desempenho superior aos métodos DE, jDE, JADE, FRANNK e, igualmente, foi superior aos únicos métodos citados até então na literatura que realizam ajuste de parâmetros com técnicas de inteligência artificial (métodos PSO e *Gravitacional Search* combinados com o ajuste de parâmetros por lógica *fuzzy*). Sendo assim, as estratégias de solução que combinam SI com RNA resultantes deste trabalho, tanto em sua forma pura, como na forma de subpopulações, apesar de ainda não amplamente exploradas, demostraram grande potencial na otimização de problemas.

Palavras-chave: otimização, inteligência de enxame, rede neural artificial, evolução diferencial, controle adaptativo de parâmetros.

ABSTRACT

This paper introduces the Full Referenced Artificial Neural Network Knowledge (FRANNK) optimizer based on swarm intelligence (SI), in which a pretrained ANN linked to a solution method is capable of efficiently make the dynamic tuning of the method's parameters. We developed a Differential Evolution version of FRANNK optimizer, that adjusts scaling factor and crossover rate, respectively *F* and *CR*, over the iterations using a classification ANN to define the direction of the parameter changing and a set of equations to define the magnitude of the adjustment. It was observed that the diversity of the population and the total fraction of iterations have impact in the tuning of the parameters, as well as the values of *F* and *CR* of the previous iteration. A performance comparison among FRANNK, DE, jDE and JADE made in benchmark functions indicated that the proposed version of FRANNK optimizer is efficient and has a strong global search ability in problem solving. The efficiency of the FRANNK optimizer was also evaluated in the solution of an energy integration problem, in which we sought to minimize the operational and investment annual cost associated with a network of heat exchangers. Additionally, a new class of solving strategy for optimization problems, called Collaborative Method, was developed, in which a portion of the population works with adjustment of the parameters *CR* and *F* following the ANN strategy and the other portion works with these parameters fixed at different values for different subpopulations. In this strategy, the best solutions are shared among subpopulations at each iteration, which promotes a synergistic effect during optimization. The use of subpopulations consisting of 1/3 of the individuals performing the dynamic adjustment of *CR* and *F* parameters by the FRANNK method and 2/3 of the population subdivided into other 3 parts, in which the values for the pair *CR*-*F* are set at, respectively, 0.5-0.5, 0.25-0.75 and 0.25-0.25, showed excellent results, and the resulting method, called C- FRANNK obtained better performance than other methods such as DE, jDE, JADE, FRANNK and, equally, it was superior to the only methods mentioned so far in the literature that perform parameter adjustment with artificial intelligence techniques (PSO and Gravitational Search methods combined with parameter tunning by fuzzy logic). Thus, the solution strategies that combine SI with RNA resulting from this work, both in its pure form and in the form of subpopulations, although not yet widely explored, have shown great potential for optimizing problems.

Keywords: optimization**,** swarm intelligence, artificial neural network, differential evolution, adaptive parameter control.

INTRODUÇÃO

Inteligência Artificial (IA) é um ramo de pesquisa e conhecimento prático baseado no desenvolvimento de *softwares* e dispositivos que simulem a capacidade do ser humano e outras formas de vida de resolver problemas, aprender, identificar e interpretar padrões, tomar decisões e agir. Muitas das aplicações da IA já têm um impacto significativo na vida das pessoas, proporcionando tanto a automatização de processos como a transformação de dados em informações úteis à tomada de decisão. Entre as mais diversas aplicações da Inteligência Artificial estão o reconhecimento de voz, o processamento de imagens, a detecção de falhas, o diagnóstico de doenças, a detecção de fraudes em transações financeiras, os sistemas de recomendações de filmes e músicas, a automatização da manutenção preditiva, os carros autônomos e as estratégias de soluções de diversos problemas de otimização.

Dentre as subáreas da Inteligência Artificial, *Machine Learning* (ML) e *Swarm Intelligence* (SI) merecem destaque. Aprendizado de Máquinas ou *Machine Learning* refere-se a um conjunto de estratégias expressas em códigos computacionais que conferem a computadores, celulares e outras máquinas a capacidade de aprender com os dados, de forma que possam identificar padrões, fazer predições e executar tarefas sem serem explicitamente para isso programados (Simon, 2013). Já a Inteligência de Enxame ou *Swarm Intelligence* (SI) refere-se a métodos de solução para problemas de otimização inspirados no comportamento coletivo de colônias de insetos sociais e outras sociedades de animais, e podem ser definidos como sistemas compostos por agentes com capacidade individual limitada, porém capazes de apresentar comportamentos coletivos inteligentes (White et al., 1998).

A maioria das estratégias de solução baseadas em SI possuem parâmetros, de modo a poder calibrar o método aos diferentes tipos de problemas a serem resolvidos. Tipicamente, os valores desses parâmetros são definidos pelo usuário antes do início da otimização e permanecem fixos no decorrer das iterações. Vários trabalhos, no entanto, mostram que o ajuste dinâmico dos parâmetros ao longo das iterações favorece o equilíbrio entre busca global e refinamento local das soluções, tendo a possibilidade de aumentar substancialmente a eficiência de um método SI (Wang et al., 2020).

Embora *Swarm Intelligence* já seja bastante usada na otimização de problemas com altas complexidades e não linearidades, nas mais diversas áreas, entre elas, engenharia, economia, biologia, ciência e negócios, até o presente momento, não há relatos bem-sucedidos do uso conjunto de *Swarm Intelligence* e *Machine Learning* no contexto de solução de problemas de otimização. Assim, a proposta do presente trabalho é acoplar um modelo de ML pré-treinado e validado para ser usado no ajuste dinâmico dos parâmetros de um método SI ao longo da otimização. Este tipo de ajuste dinâmico

que associa SI e ML se caracteriza como uma contribuição no ramo da otimização que ainda não foi estudado pela literatura e por esta razão foi escolhido como assunto da presente tese.

Neste contexto, foi escolhido o método Evolução Diferencial como referência para associação de ML aos métodos de solução de otimização. O método Evolução Diferencial ou *Differential Evolution* (DE) é um método populacional inspirado na evolução de espécies que minimiza ou maximiza uma função objetivo, dependendo da finalidade da otimização, ao longo de gerações sucessivas. Nesse método, os valores dos parâmetros *F* e *CR* chamados, respectivamente, fator de mutação e taxa de crossover, são particularmente importantes, afetando consideravelmente a velocidade de convergência da otimização.

Dentre as diversas técnicas de ML testadas na presente tese, escolheu-se as Redes Neurais Artificiais (RNA) como método de interesse principal, após a realização de um estudo comparativo com três outros métodos conhecidos como *Randon Forest* (RF), Máquina de Vetor Suporte (SVM, *Support Vector Machine*) e Regressão Logística (RL). Historicamente, as Redes Neurais Artificiais são rápidas como resultado do processamento paralelo, uma característica inspirada no sistema cerebral. Assim, o ajuste de parâmetros pôde tirar proveito do processamento rápido e da descrição simples de sistemas complexos fornecidos pelas RNAs.

Assim, no presente trabalho, uma nova estratégia de ajuste de parâmetros utilizando Redes Neurais Artificiais é apresentada e uma versão melhorada de um algoritmo DE denominado otimizador FRANNK (*Full Referenced Artificial Neural Network Knowledge*) é proposta. As principais contribuições do estudo em questão podem ser resumidas da seguinte forma: a) criou-se uma nova classe de metodologias em que a inteligência inspirada no funcionamento do cérebro e a inteligência de enxame trabalham juntas em problemas de otimização; b) a avaliação da RNA de FRANNK forneceu informações, ainda não disponíveis na literatura, em termos de compreensão dos fatores que impactam no ajuste dinâmico dos parâmetros do DE; c) a presente metodologia mostrou-se eficiente quando comparada a outros otimizadores.

Esta tese está organizada da seguinte maneira: no Capítulo 1, apresenta-se uma revisão bibliográfica, que reúne os fundamentos necessários para o desenvolvimento do modelo de otimização proposto. No Capítulo 2, detalha-se a metodologia usada para geração de dados de treinamento e teste da RNA, a metodologia usada no desenvolvimento do otimizador FRANNK bem como os resultados relativos à eficiência do novo *solver* proposto. No Capítulo 3 testa-se a eficiência do otimizador FRANNK na solução de um problema de aplicação prática (otimização de uma rede de trocadores de calor), e apresenta-se uma nova classe de estratégia de solução para problemas de otimização denominada Método Colaborativo. No Capítulo 4, desenvolve-se uma nova versão do método

colaborativo, denominado C-FRANNK, e faz-se um estudo comparativo entre C-FRANNK, metodologias adaptativas tradicionais, e metodologias citadas na literatura que usam AI para o ajuste de parâmetros. Finalmente, no Capítulo 5, são apresentadas as conclusões gerais deste trabalho.

CAPÍTULO 1 - REVISÃO BIBLIOGRÁFICA

OTIMIZAÇÃO

A área de otimização é bastante estratégica para indústrias e empresas que desejam ter um diferencial competitivo. Os objetivos podem ser diversos: maximizar os lucros, reduzir os tempos de desenvolvimento de produto e de processo, reduzir os custos, otimizar a performance de equipamentos pela escolha dos parâmetros de processo mais adequados, definir o melhor sequenciamento de tarefas e a melhor alocação de recursos, entre outros.

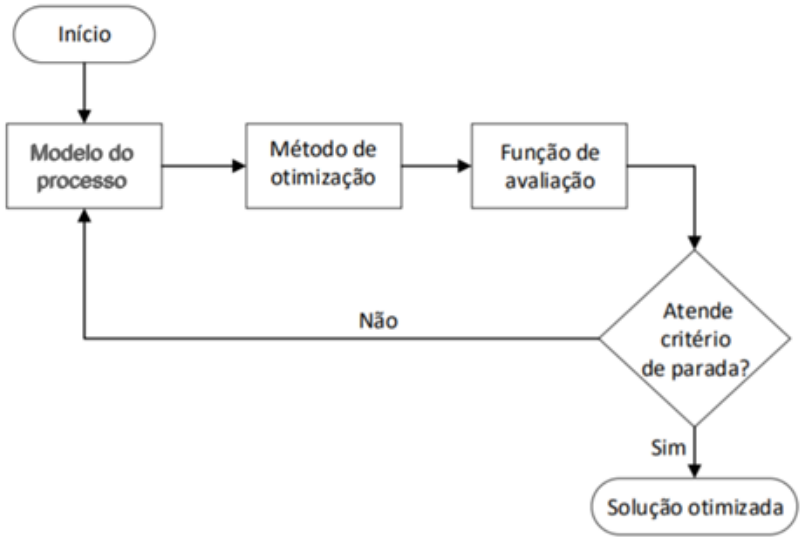

Figura 1: Fluxograma geral da otimização de um processo.

A Figura 1 mostra um fluxograma geral e simplificado da otimização de um processo. Do ponto de vista matemático, o termo otimização refere-se a problemas de maximização ou minimização de uma função objetivo, através da definição da melhor combinação de um conjunto de variáveis, sejam elas contínuas, inteiras ou composta por uma mistura de ambas. Os métodos de solução para problemas de otimização podem ser divididos em dois grandes grupos: determinísticos e probabilísticos. Em algoritmos determinísticos, sempre que uma mesma entrada for repetida, o resultado obtido será o mesmo. No caso dos algoritmos probabilísticos, uma mesma entrada pode produzir resultados diferentes, em decorrência de alguma etapa do algoritmo que contenha eventos baseados em aleatoriedade, como por exemplo a geração de um número aleatório.

Os métodos de solução determinísticos são baseados nos cálculos de derivadas de primeira ou segunda ordem ou de uma aproximação destas e garantem o alcance do ótimo global no caso de problemas que possuem apenas uma solução ótima (ditos unimodais). A maioria desses métodos podem apresentar, porém, algumas limitações em outros casos, entre elas a dificuldade de trabalhar com variáveis discretas e com funções objetivo não diferenciáveis, bem como a dificuldade em encontrar soluções ótimas globais no caso de problemas contendo diversos ótimos locais (ditos multimodais) (Albuquerque et al., 2010; Rainer Storn & Price, 1997). Além disso, para a maioria dos métodos de programação determinística, tanto a solução encontrada como o tempo de otimização são fortemente dependentes do ponto de partida e mesmo quando se trata de um algoritmo de otimização global, para muitos problemas os tempos computacionais necessários para se chegar à solução são extremamente altos.

Os métodos de solução probabilísticos, por outro lado, não empregam o cálculo de derivadas atuando na busca direta de soluções no espaço viável. Essa metodologia, porém, exige um grande número de avaliações do valor da função objetivo e das restrições, o que pode gerar um custo computacional elevado. Uma subclasse dos métodos de solução probabilísticos são as heurísticas, termo usado para descrever métodos que, baseado na experiência ou julgamento, parece conduzir a uma boa solução de um problema, mas que não garante necessariamente o alcance do ótimo global, apesar de muitos casos práticos mostrarem grande proximidade a ele (Foulds, 1984). Dentre os métodos heurísticos, os algoritmos conhecidos como *Swarm Intelligence* (SI) ou Inteligência de Enxame merecem destaque por serem geralmente capazes de encontrar boas soluções para problemas difíceis. Esse tipo de método será mais bem detalhado na seção a seguir.

CONCEITOS FUNDAMENTAIS DE OTIMIZAÇÃO

Um dos aspectos importantes na otimização relaciona-se ao esforço computacional relativo à busca das soluções de interesse. Nos métodos heurísticos, não somente o tamanho da população e os critérios de evolução desta influem na busca da solução, mas também fatores como o espaço de busca, o número máximo de iterações e as condições de término da otimização.

O espaço de busca pode ser caracterizado como um conjunto que contém todas as soluções viáveis do problema, ou seja, aquelas que obedecem às restrições do problema. Já as iterações representam um processo de resolução de operações lógico-matemáticas em que sucessivamente o objeto de cada uma é o resultado da que a precede.

Em um método de solução populacional, como é o caso dos algoritmos baseados em *Swarm Intelligence,* cada indivíduo representa uma possível solução para o problema. Uma população bem

dimensionada, composta por *N* indivíduos (também chamados vetores) deve cobrir todo espaço de busca bem como garantir que o esforço computacional necessário para que se tenha uma solução satisfatória seja aceitável. Além disso, existe uma relação diretamente proporcional entre o tamanho da população necessário e dimensão do problema (número de variáveis de projeto do problema).

De modo geral, os problemas de otimização podem ser classificados em duas categorias: funções *benchmark* e problemas de aplicação prática. As funções *benchmark* são problemas artificiais comumente usados para avaliar o comportamento de um algoritmo em situações diversas e muitas vezes de difícil resolução. Essas funções *benchmark* são comumente classificadas em termos de separabilidade e modalidade.

Uma função de *p* variáveis é chamada separável se puder ser escrita como a soma de *p* funções de apenas uma variável. Por outro lado, uma função é chamada de não separável se suas variáveis apresentam inter-relação entre si. A separabilidade é considerada uma medida de dificuldade das funções de benchmark. Em geral, funções separáveis são relativamente fáceis de resolver, quando comparadas com sua contraparte indissociável, porque se todos os parâmetros ou variáveis forem independentes, uma sequência de *p* processos de otimização independentes pode ser executada (Jamil & Yang, 2013; Ortiz-Boyer et al., 2005).

A modalidade, por outro lado, refere-se ao número de picos, ou seja, ótimos locais presentes em uma função. Uma função com um único mínimo ou máximo local é denominada unimodal, enquanto uma função com mais de um mínimo ou máximo local é chamada multimodal. As funções multimodais, especialmente aquelas com muitos mínimos locais, estão entre as classes de problemas mais difíceis para muitos algoritmos, sendo usadas para testar a capacidade destes de escapar de qualquer mínimo local. Se o processo de exploração de um algoritmo for mal projetado, ele não poderá pesquisar a região de busca de certas funções de maneira eficaz (Jamil & Yang, 2013).

Além das classificações quanto à modalidade e separabilidade de uma função, há outras características que possuem impacto na velocidade de convergência de uma otimização. Por exemplo, funções com superfícies planas representam uma dificuldade para os algoritmos, uma vez que essa característica não dá ao algoritmo nenhuma informação para direcionar o processo de busca. Regiões de vales (área estreita com pouca variabilidade cercada por regiões de descida íngreme) e bacias (declínio relativamente íngreme em torno de uma grande área) podem igualmente apresentar dificuldades para os algoritmos de otimização, pois estes algoritmos podem ser facilmente atraídos para regiões íngremes, e uma vez nessas regiões de pouca variabilidade, o processo de busca é severamente prejudicado (Jamil & Yang, 2013).

SWARM INTELLIGENCE (SI)

No contexto de algoritmos computacionais, um enxame ou *swarm* é uma população de indivíduos simples interagindo entre si que é capaz de otimizar algum objetivo através de uma busca cooperativa em determinado espaço de soluções. Esses indivíduos podem ser insetos, pássaros, seres humanos, vetores, robôs etc. Assim, a família de algoritmos probabilísticos conhecida como *Swarm Intelligence* (SI) baseia-se em uma população de indivíduos que propõem soluções inicialmente aleatórias e geralmente uniformemente distribuídas em toda a região de busca de um problema. Ao longo das iterações, essa população de soluções é refinada através da interação de cada indivíduo com os outros elementos da população, sejam eles os vizinhos próximos, indivíduos escolhidos aleatoriamente, ou os indivíduos que encontraram a melhor solução até o momento, a depender da estratégia adotada pelo método de solução. Dessa forma, ao longo das iterações, os indivíduos ajustam suas variáveis e começam a encontrar boas soluções para os problemas, mesmo quando se trata de problemas difíceis, por exemplo, multimodais, não separáveis, não convexos, ou não diferenciáveis (Zomaia, 2006).

Assim, para certos tipos de problemas, com altas complexidades e não linearidades, geralmente é mais interessante usar um algoritmo de inteligência de enxame do que um método determinístico, especialmente se o tempo computacional for um fator importante (Aslan et al., 2019; Bansal et al., 2018; H. Liu et al., 2010; H. Wang et al., 2017). Além disso, algoritmos SI são normalmente fáceis de implementar, não requerem o gradiente de uma função para otimizar um problema e, em muitos casos, podem contornar ótimos locais, sendo adequados para encontrar ótimos globais em um tempo razoável. Otimização por Enxame de Partículas ou *Particle Swarm Optimization* (PSO), Colônia de Abelhas artificiais ou *Artificial Bee Colony* (ABC), Otimização por colônia de formigas ou *Ant Colony Optimization* (ACO), Algoritmo de Vagalumes ou *Firefly Algorithm* (FA), Algoritmo Genético ou *Genetic Algorithm* (GA) e Evolução Diferencial ou *Differential Evolution* (DE) estão entre os algoritmos SI mais populares presentes na literatura (Dorigo et al., 2006; Holland, 1975; Karaboga & Basturk, 2007; Kennedy & Eberhart, 1995; Rainer Storn & Price, 1997; Yang, 2009)

DIFFERENTIAL EVOLUTION (DE)

A Evolução Diferencial (DE, *Differential Evolution*), proposta por Storn e Price em 1997, é um método estocástico, populacional, inspirado na evolução das espécies. O algoritmo DE tradicional utilizado (DE / *rand* / 1/bin) nesta pesquisa é composto por 4 etapas, descritas a seguir pelas Equações (1) a (4). A primeira etapa é a geração de uma população aleatória, o que ocorre apenas na primeira

iteração; as outras 3 etapas (mutação, cruzamento e seleção) são repetidas em todas as iterações subsequentes, até que o critério de parada seja atingido.

I- Inicialização: geração de uma população aleatória, uniformemente distribuída, de possíveis soluções (indivíduos).

$$x_i^0 = x^{min} + rand(0,1).(x^{max} - x^{min}) \quad (1)$$

onde x_i representa cada vetor (de dimensão *d*) ou indivíduo da população, o índice 0 representa a primeira geração *g*, o índice *i* refere-se a cada indivíduo da população (*i*=1,2...*n*), *rand* (0,1) é um número aleatório uniforme entre 0 e 1 sorteado para cada posição de cada vetor; x^{min} e x^{max} são, respectivamente, os limites inferior e superior do espaço de busca, definidos para cada coordenada do vetor x_i .

II- Mutação: combinação linear de 3 indivíduos da população, escolhidos aleatoriamente, gerando uma segunda população, denominada população mutante, de mesmo tamanho e dimensão da população original.

$$v_i^g = x_{R1i}^g + F.(x_{R2i}^g - x_{R3i}^g) \quad (2)$$

onde *g* é a geração atual; v_i^g é um vetor mutante criado em cada geração; x_{R1i}^g , x_{R2i}^g e x_{R3i}^g são vetores da população atual, escolhidos aleatoriamente, que se diferem do vetor atual x_i^g; e *F* é um parâmetro denominado fator de mutação, que é um número positivo, geralmente definido no intervalo [0, 1].

III- Cruzamento ou *Crossover*: o vetor atual x_i^g (população original) mescla suas coordenadas com o vetor v_i^g da população mutante, gerando um outro vetor $u_i^g = (u_{i,1}^g, u_{i,2}^g, \ldots, u_{i,d}^g)$ que comporá uma terceira população, denominada população de cruzamento. A intenção desta etapa, representada na Figura 2, é cruzar os genes (coordenadas) de dois indivíduos (pais) de duas populações para gerar um indivíduo novo (descendente). Para decidir qual gene é transmitido, sorteia-se um número aleatório entre 0 e 1 e compara-se esse valor com a taxa de crossover *CR*. Se *rand* (0,1) for menor do que *CR*, o gene mutante é passado adiante, caso contrário, o gene da população corrente é transmitido.

$$u_{i,j}^{g} = \begin{cases} v_{i,j}^{g} \ \ se\ j = k\ ou\ rand_j \leq \ CR \\ x_{i,j}^{g} \ \ caso\ contrário \end{cases} \tag{3}$$

onde os índices *k* e *j* representam coordenadas do vetor, *k* é escolhido aleatoriamente em {1,2,3,…, dimensão}, $rand_j$ é um número aleatório no intervalo [0, 1] e *CR* é um parâmetro denominado de taxa de crossover pertencente ao intervalo [0, 1]. A expressão *j* = *k* assegura que as novas soluções não dupliquem a solução original.

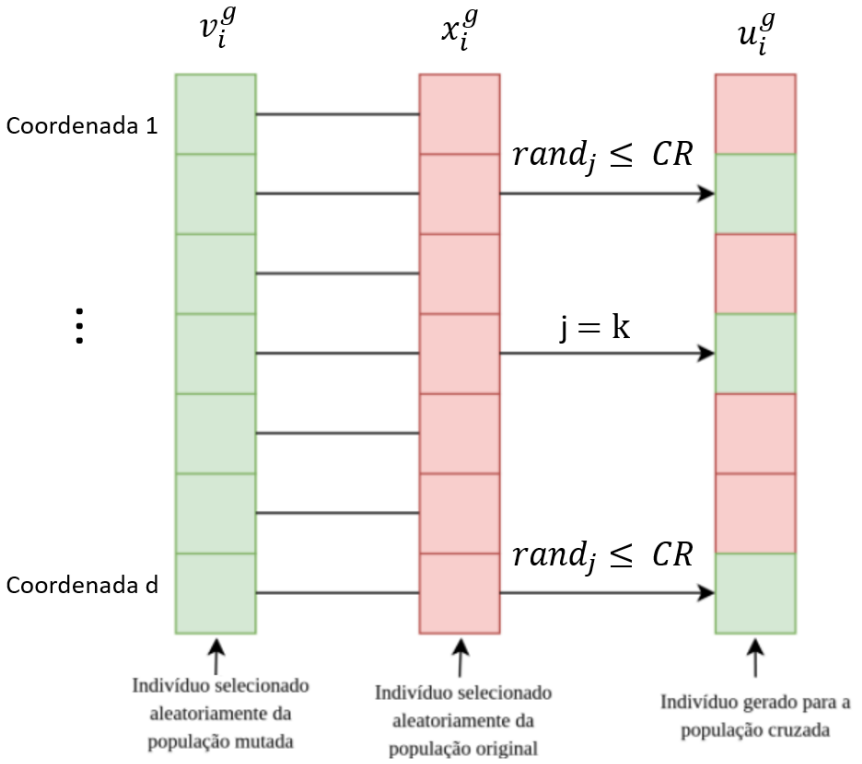

Figura 2: Esquema da etapa de cruzamento do algoritmo DE.

IV- Seleção: É feita a avaliação da função objetivo para cada indivíduo *i* de ambas as populações corrente e cruzada, e o melhor indivíduo entre eles permanece para compor a população corrente da próxima geração (*g*+1).

$$x_i^{g+1} = \begin{cases} u_i^g \ se\ f\left(u_i^g\right) \leq f\left(x_i^g\right) \\ x_i^g \ caso\ contrário \end{cases} \tag{4}$$

DE ADAPTATIVO

O algoritmo DE é considerado um método eficaz na resolução de diversos problemas de otimização teóricos e de aplicação prática, sendo também reconhecido por sua simplicidade e facilidade de implementação. Seu desempenho e velocidade de convergência, no entanto, é geralmente bastante dependente da escolha dos parâmetros *F* (Fator de Mutação) e *CR* (Taxa de Crossover), os quais desempenham um papel importante no balanço entre exploração global e refinamento local das soluções (Moussa & Awotunde, 2018; Zhang & Sanderson, 2009).

No algoritmo de Evolução Diferencial tradicional, ambos os parâmetros *F* e *CR* permanecem fixos durante toda a otimização. Porém, não há configuração de parâmetro fixo que seja adequada para todos os tipos de problemas ou mesmo para diferentes estágios de evolução de um mesmo problema. Por essa razão, no algoritmo DE convencional, esse ajuste de parâmetros de controle deve ser feito por tentativa e erro, o que pode necessitar de um tempo computacional considerável. Motivados por esses fatores, diversos pesquisadores introduziram mecanismos adaptativos a fim de atualizar dinamicamente os parâmetros de controle sem o conhecimento prévio do usuário acerca da relação entre a configuração do parâmetro e as características do problema a ser otimizado (Zhang & Sanderson, 2009). As mais tradicionais metodologias para o ajuste dinâmico de parâmetros estão descritas a seguir.

a) SaDE (*Self-adaptive Differential Evolution* ou Algoritmo de Evolução Diferencial Auto-adaptativo):

Neste método, os fatores de mutação (F_i) são gerados independentemente em cada geração, de acordo com uma distribuição normal com média 0,5, desvio padrão 0,3 e truncamento no intervalo (0, 2]. Os autores afirmam que este esquema é capaz de manter tanto o refinamento local (com pequenos valores de F_i) como a busca global (com grandes valores de F_i). As taxas de *crossover*, por outro lado, são geradas aleatoriamente de acordo com uma distribuição normal independente com média igual a CR_m e desvio padrão igual a 0,1, onde CR_m é a média dos *CR* em gerações anteriores iniciada em 0,5. Além disso, neste *solver* são utilizadas duas estratégias de mutação (DE/*rand*/1 e DE/*current-to-best*/1), escolhidas de acordo com uma probabilidade, a qual é adaptada de acordo com a taxa de sucesso de cada estratégia, considerando as últimas 50 gerações (Qin et al., 2009).

b) jDE:

Método baseado no clássico DE/*rand* /1/ *bin*, o qual adapta os parâmetros de controle F_i e CR_i associados a cada indivíduo. O processo de inicialização define F_i=0,5 e CR_i=0,9 para cada indivíduo e na sequência, novos valores para F_i e CR_i são definidos (com probabilidades τ_1=τ_2=0,1 em cada geração) de acordo com distribuições uniformes em [0,1, 1] e [0, 1], respectivamente. Acredita-se que melhores valores dos parâmetros tendem a gerar indivíduos com maior probabilidade de sobreviver e, portanto, esses valores devem ser propagados para as próximas gerações (Brest et al., 2006).

c) JADE:

Método que utiliza a estratégia de mutação DE /*current-to-pbest*, o qual utiliza os 100 *p*% melhores indivíduos da população, com $p \in (0,1]$, durante o processo de mutação. Além disso, os parâmetros de controle que estão associados a cada indivíduo i da população são regenerados a cada geração, sendo que o fator de mutação F_i é adaptado de acordo com a distribuição de Cauchy, e a taxa de *crossover* CR_i é adaptada de acordo com uma distribuição normal (Zhang & Sanderson, 2009).

Existem também as metodologias híbridas para ajuste de parâmetros, incluindo-se aquelas que empregam a Lógica *Fuzzy* (J. Liu & Lampinen, 2005; Tsafarakis et al., 2020); regressão de dados de parâmetro durante o processo de otimização usando Redes Neurais Artificiais ou projeto ortogonal (Fister et al., 2016; Gong et al., 2008); bem como outras metodologias estocásticas tais como diferentes tipos de PSO-DE (H. Liu et al., 2010; S. Wang et al., 2019).

Embora existam diversos métodos de Evolução Diferencial adaptativos, novos estudos ainda são necessários seja para solucionar problemas como convergência prematura, seja para reduzir a complexidade das regras utilizadas na adaptação de parâmetros.

MACHINE LEARNING

A área de *Machine Learning* (ML) está na interseção entre ciência da computação, estatística e engenharia, explorando a construção de algoritmos com capacidade de aprender através de exemplos. Esses algoritmos realizam análises de um conjunto de dados, de modo a inferir informações sobre as propriedades e os padrões desses dados. A identificação desses padrões permite ao algoritmo criar um modelo que quantifique como cada variável de entrada influencia na obtenção da resposta (saída), e esse processo é denominado treinamento. O modelo desenvolvido é, então, capaz de fazer previsões sobre novos dados, diferentes daqueles usados para treinar o modelo, processo este conhecido como generalização (Segaran, 2007).

Um exemplo do uso de *Machine Learning* presente no dia a dia de grande parte das pessoas é o sistema de recomendações de vídeos do YouTube. A cada minuto, centenas de horas de vídeos são carregadas na plataforma, e a função do algoritmo de ML é selecionar e ranquear um conjunto de vídeos que aparecerá na página inicial de cada usuário, bem como a lista de reprodução automática. Para realizar essa seleção, o algoritmo usa como entradas a) informações do usuário (estilo de vídeos já assistidos, textos utilizados no campo de busca, curtidas e comentários realizados, dados demográficos, faixa etária, sexo, etc.); b) informações do vídeo (número de cliques, número de interações dos usuários com o vídeo, porcentagem de visualizações completas do conteúdo, data de postagem); c) inter-relação entre usuário e vídeo (quando foi a última vez que o usuário assistiu um vídeo do tópico em questão, quantos vídeos do referido canal o usuário já assistiu) (Covington et al., 2016).

O Aprendizado de Máquinas pode ser dividido em diferentes categorias, conforme detalhado a seguir (Raschka, 2015; Segaran, 2007):

a) Aprendizado supervisionado: composto por técnicas que usam exemplos de entradas (variáveis independentes) e saídas (variáveis dependentes) a fim de aprender como fazer previsões. A característica fundamental desse tipo de aprendizado é a presença da resposta desejada nos dados de treinamento. Um exemplo clássico é a classificação de e-mails, no qual o algoritmo é treinado com diversos exemplos de e-mails rotulados em spam ou não spam, de modo a prever a que classe pertencerá um novo e-mail que chegar na caixa de entrada. Além dos problemas de classificação, que categorizam elementos em diferentes classes, há também os problemas de regressão, os quais buscam prever saídas numéricas e contínuas. Um exemplo de problema de regressão é a predição da demanda de um produto de acordo com dados históricos da demanda passada, dados sazonais e dados da concorrência.

b) Aprendizado não supervisionado: composto por técnicas que buscam encontrar padrões em um conjunto de dados não rotulados (nos quais as respostas corretas para determinado conjunto de entradas não estão presentes na fase de treinamento, sendo função do algoritmo buscar padrões para encontrar as saídas). Um exemplo típico desse tipo de aprendizado é o agrupamento de clientes de mesmo perfil. Neste caso, não é informado ao algoritmo a qual grupo pertence o cliente, e este deverá identificar padrões semelhantes entre os clientes para agrupá-los. Outros exemplos de aplicações de aprendizado não supervisionados são sistemas de recomendação de filmes ou músicas.

c) Aprendizado por reforço: neste tipo de aprendizado, o objetivo é desenvolver um sistema capaz de melhorar seu desempenho com base nas interações com o ambiente. O algoritmo recebe recompensas ou penalidades pelas ações que executa, buscando maximizar a recompensa total. Um

exemplo de aplicação do aprendizado por reforço é o treinamento de um modelo para o jogo de xadrez. Neste caso, o agente decide sobre uma série de movimentos dependendo do estado do tabuleiro (ambiente), e a recompensa pode ser definida como vitória ou derrota ao final do jogo.

O problema de ajuste de parâmetros explorado no presente trabalho, enquadra-se como um problema de aprendizado supervisionado do tipo classificação, no qual o modelo precisa decidir se a magnitude do parâmetro deve aumentar ou reduzir. Para solucioná-lo, foram testados quatro métodos de *Machine Learning* distintos (Redes Neurais Artificiais, *Randon Forest*, Máquina de Vetor Suporte e Regressão Logística), os quais serão explanados em seções seguintes.

As métricas de avaliação de um modelo de classificação binário, como o do presente trabalho, são baseadas na comparação entre as classes preditas pelo modelo e as classes verdadeiras do problema, e objetivam mensurar, quão distante da classificação perfeita está o modelo. Uma maneira de representar visualmente a performance de um modelo é através da matriz confusão (Figura 3), que indica quantas ocorrências existem em cada grupo: verdadeiro positivo (*VP*), verdadeiro negativo (*VN*), falso positivo (*FP*), falso negativo (*FN*). A partir dos dados presentes na matriz confusão, uma série de outros indicadores podem ser obtidos, conforme será descrito a seguir.

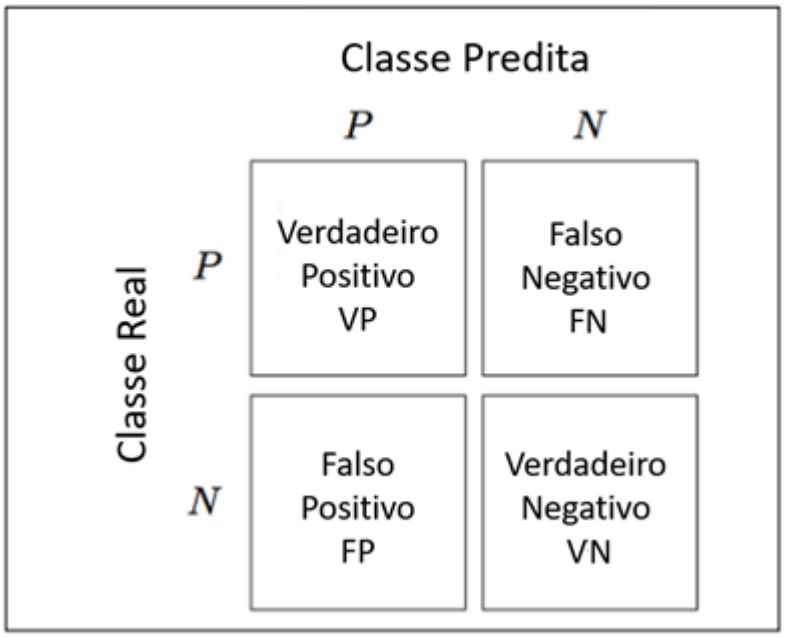

Figura 3: Esquema de uma matriz confusão.

A Acurácia (*A*) informa quantas ocorrências foram classificadas corretamente, independente da classe, conforme Equação (5). Essa métrica é uma das mais utilizadas para medir o desempenho global de problemas com classes balanceadas.

$$A = \frac{VP + VN}{VP + VN + FP + FN} \tag{5}$$

A Precisão (*P*) é definida, conforme Equação (6), pela razão entre a quantidade de ocorrências classificadas corretamente como positivas e o total de ocorrências classificadas como positivas.

$$P = \frac{VP}{VP + FP} \tag{6}$$

A Taxa de Verdadeiro Positivo (R, *Recall*) é definida, conforme Equação (7), pela razão entre a quantidade de ocorrências classificadas corretamente como positivas e a quantidade de ocorrências que são de fato positivas. Já a Taxa de Verdadeiro Negativo ou Especificidade (S, S*pecificity*) é a proporção de ocorrências negativas classificadas corretamente, conforme Equação (8).

$$R = \frac{VP}{VP + FN} \tag{7}$$

$$S = \frac{VN}{VN + FP} \tag{8}$$

O F1 score (*F1*) é definido, conforme Equação (9), pela média harmônica entre a Precisão (P) e a Taxa de Verdadeiro Positivo (R). Essa métrica é especialmente útil para medir o desempenho global de problemas com classes não balanceadas.

$$F1 = \frac{2VP}{2VP + FP + FN} \tag{9}$$

REDE NEURAL ARTIFICIAL (RNA)

As Redes Neurais Artificiais (RNAs) constituem a parte da Inteligência Artificial e *Machine Learning* que se inspira no funcionamento do cérebro, mais precisamente no funcionamento das redes de neurônios que compõe o cérebro, no que diz respeito ao processamento de informações de forma paralela, gerando aprendizado. Assim como o cérebro, as RNAs têm capacidade de generalizar e de reproduzir respostas complexas, na forma de um modelo caixa preta ou na forma de um modelo híbrido que possui equações fenomenológicas associadas ao aprendizado com dados.

A Figura 4 mostra a representação de uma Rede Neural Artificial (RNA) do tipo *feedfoward* (sem retroalimentação), na qual as informações de entrada passam por camadas ocultas até gerar a saída, que é a resposta desejada. Cada camada oculta é formada de elementos ditos neurônios, que

processam as informações de entrada, multiplicando-as por valores numéricos (pesos sinápticos). Em cada neurônio, é feita, então, a soma das entradas ponderadas pelos pesos e ao resultado dessa soma aplica-se uma função matemática, denominada função de ativação. O aprendizado da RNA consiste em ajustar os pesos sinápticos de modo que as informações recebidas gerem um maior ou menor impacto na resposta final, conforme a relevância de cada entrada.

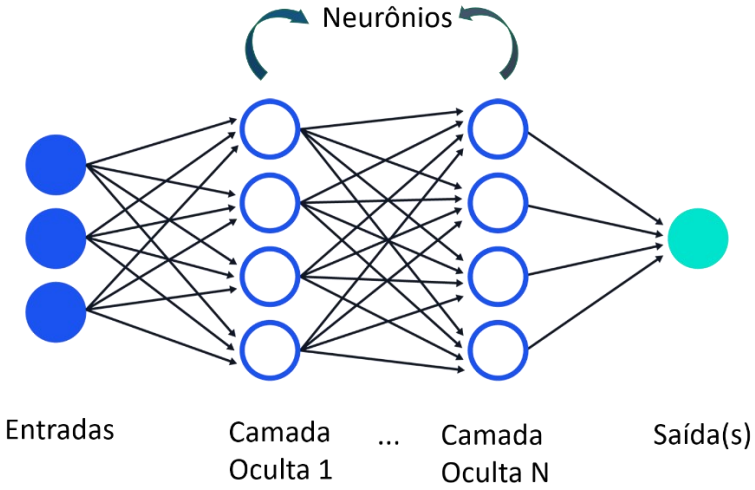

Figura 4: Representação de uma Rede Neural Artificial

A equação geral de uma RNA está representada abaixo.

$$y_i = f\left(\sum_{j=1}^{n} w_{ij}u_i + b_i\right) \tag{10}$$

onde u_i é a entrada ou saída do i-ésimo neurônio, w_{ij} são os pesos na conexão do i-ésimo neurônio na camada anterior com o j-ésimo na camada presente, b_j representa o *bias* no j-ésimo neurônio e f é a função de ativação.

As funções matemáticas, ditas funções de ativação, devem ser escolhidas de forma a proporcionar o processamento paralelo, o que significa dizer que diferentes neurônios contribuem para obtenção da resposta final. As Equações (11) a (13) são as funções de ativação mais comuns denominadas, respectivamente, sigmóide, tansig e purelin.

$$f(u) = \frac{1}{1 + e^{-u}} \tag{11}$$

$$f(u) = \frac{1 - e^{-u}}{1 + e^{-u}} \tag{12}$$

$$f(u) = u \tag{13}$$

Durante o processo de ajuste dos pesos sinápticos, uma das estratégias mais utilizadas trata-se do método chamado gradiente descendente, em um processo de retropropagação ou *backprogation* dos erros. O *backprogation* é realizado através da diferença (erro) entre a resposta final esperada e a resposta calculada pelo modelo seguido do cálculo do gradiente descendente para cada peso que compõe a entrada de cada camada. Esse processo é repetido sequencialmente até que a camada de entrada seja corrigida, conforme mostra a Figura 5. Nesta figura, o processo de *backpropagation* é representado por setas vermelhas para todas as respostas finais, bem como para a primeira linha de dados, de modo a representar a correção dos pesos das outras camadas, que ocorre de traz para frente a partir da resposta.

Neste processo, uma função objetivo de erro, também chamada de função perda ou custo global (Equação (14)) precisa ser minimizada e no método do gradiente descendente, em cada iteração (ou época) de correção, a derivada do erro é utilizada na estimativa dos novos parâmetros, conforme mostra a Figura 6, a qual ilustra esse processo considerando a correção de um peso sináptico *w*. Observa-se nesta figura que, após cada época ou iteração de ajuste dos pesos sinápticos (*w*), a correção dos parâmetros é feita através do erro estimado como um valor associado à derivada. Esse erro diminui ao longo das iterações e por esta razão o método é chamado gradiente descendente.

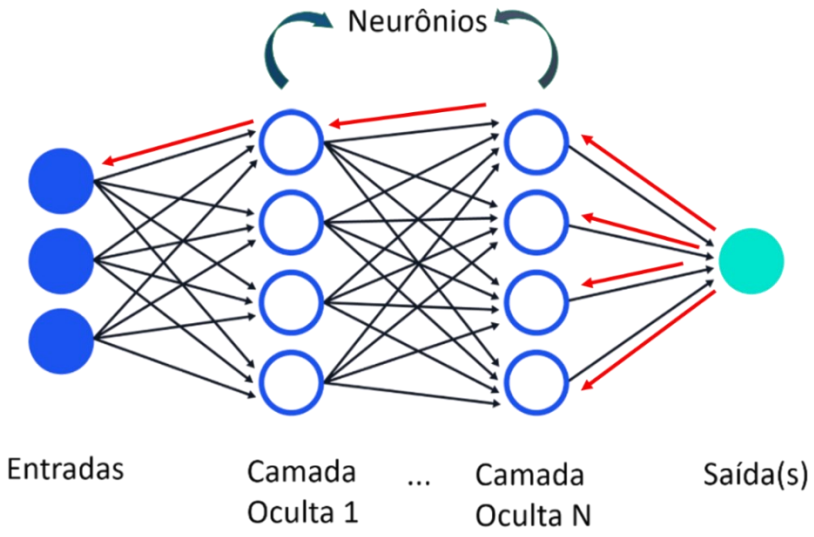

Figura 5: Ajuste dos pesos sinápticos da RNA por *backpropagation*.

$$J(w) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - y_i^{true})^2 = \frac{1}{n}\sum_{i=1}^{n} L(y_i, y_i^{true}) \quad (14)$$

sendo *J(w)* a função custo global, y_i e y_i^{true}, respectivamente, os valores estimados pelo modelo e esperados (reais) e $L(y_i, y_i^{true})$ representa a função custo que, no presente caso, corresponde a diferença entre os valores esperados e calculados elevada ao quadrado e dividida por dois, mas que pode ser descrita por outra função distinta.

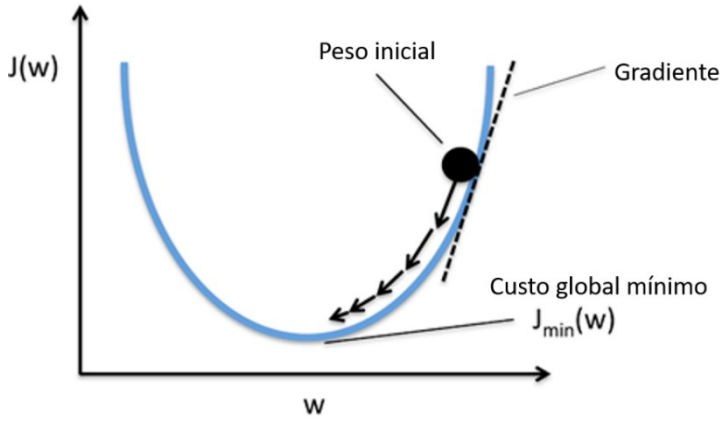

Figura 6: Método do gradiente descendente - adaptado da literatura (Mayo, 2017)

Quando o ajuste dos pesos é realizado segundo a metodologia do gradiente descendente, as diferenças entres os valores gerados como erros são propagadas através de toda rede, de forma a corrigir os pesos sinápticos associados a cada camada, conforme mostra a Figura 5. Em termos matemáticos, este processo é descrito pelas Equação (15), na qual *L* representa a função custo utilizada, $f\left(x^{(i)}, w\right)$ representa o valor estimado da variável associada aos pesos sinápticos e valores de entrada, $y^{(i)}$ representa os valores esperados, λ é uma constante utilizada na regularização e *R(w)* é a função de regularização, que na maioria dos casos é a soma dos quadrados dos pesos sinápticos (regularização L2) ou a soma dos valores absolutos dos pesos sinápticos (regularização L1), função esta usada para possibilitar o controle dos valores máximos dos pesos sinápticos.

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} L\left(f\left(x^{(i)}, w\right), y^{(i)}\right) + \lambda R(w) \quad (15)$$

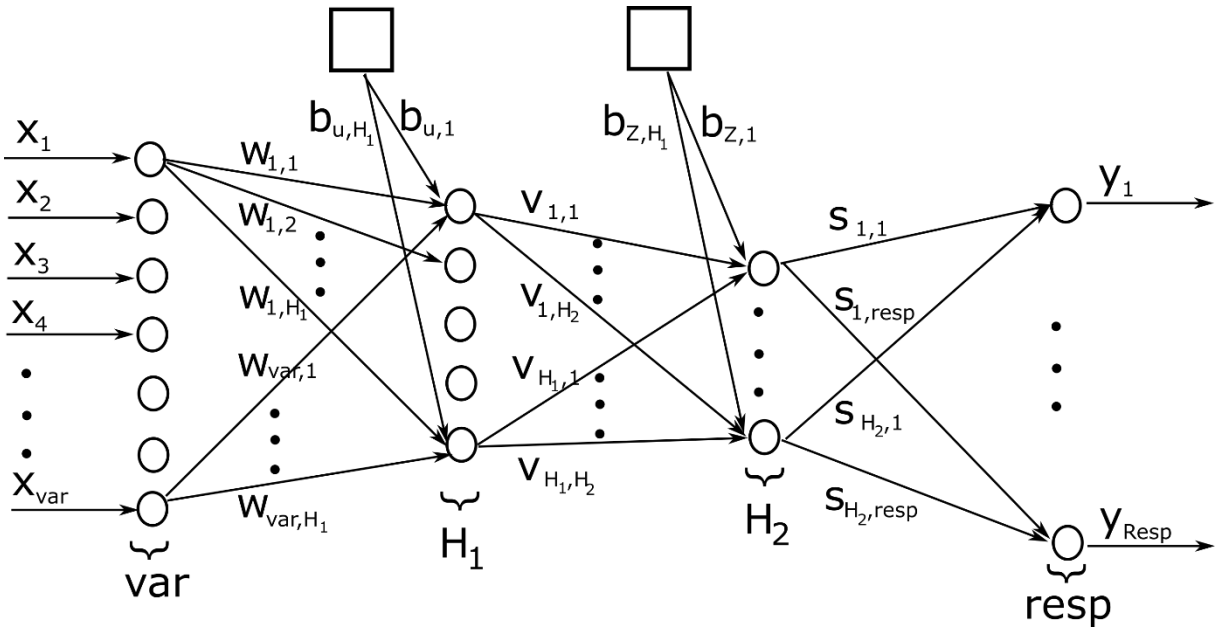

Figura 7: RNA com duas camadas ocultas com apresentação dos pesos sinápticos

Para uma rede neural com duas camadas ocultas denominadas H1 e H2, conforme mostra a Figura 7, tem-se que uma das formas de descrever o método do gradiente descendente é apresentado pelas Equações (16) a (24) nas quais *k* e *k*+1 representam valores da iteração atual e iteração anterior, as matrizes unidimensionais *X*, *U*, *Z* e *Y*, são, respectivamente, as variáveis de entrada, de saída das camadas ocultas (primeira e segunda camada oculta) e de resposta final, as matrizes *W*, *V* e *S* são os pesos sinápticos associadas a rede neural, α e η representam o coeficiente de momento e a taxa de aprendizagem e *B* é o limiar de resposta, que pode ser incorporado às matrizes de pesos quando se considera o valor unitário como um dos elementos de *X*, *U*, *Z* e *Y*.

$$S^{k+1} = S^k + \Delta S \tag{16}$$

$$V^{k+1} = V^k + \Delta V \tag{17}$$

$$W^{k+1} = W^k + \Delta W \tag{18}$$

$$\Delta S_{ij}^{k+1} = \eta \Delta Y_j Z_i + \alpha \Delta S_{ij}^k \tag{19}$$

$$\Delta V_{ji}^{k+1} = \eta \Delta Z_i U_j + \alpha \Delta V_{ji}^k \tag{20}$$

$$\Delta W_{ij}^{k+1} = \eta \Delta U_j X_i + \alpha \Delta W_{ij}^k \tag{21}$$

$$\Delta Y = (Y^{true} - Y).f'(S^t.Z) \tag{22}$$

$$\Delta Z = S.\Delta Y f'(Z^t.U) \quad (23)$$

$$\Delta U = V.\Delta Z f'(W^t.X) \quad (24)$$

O aprendizado da RNA consiste em ajustar os parâmetros denominados pesos sinápticos de forma que a rede neural consiga reproduzir repostas utilizando processamento paralelo da informação. As respostas associadas a neurônios isolados são incapazes de garantir a reprodução de todos valores possíveis, mas a rede neural, com o conjunto de todos neurônios, tem como característica ser um aproximador universal, isto é, ser capaz de representar qualquer tipo de função como resposta (Cybenko, 1989; Hornik, 1991; Rumelhart et al., 1986).

SUPPORT VECTOR MACHINE (SVM)

Máquina de Vetor Suporte ou *Support Vector Machine* (SVM) é um método de ML que utiliza a teoria estatística do erro, também conhecida como teoria Vapnik-Chervonenkis, na obtenção de respostas para problema de classificação ou regressão. No método SVM, um grupo de dados, também chamados vetores suportes, é selecionado e usado para obter hiperplanos utilizados para separar os demais dados que são ajustados por meio de funções ditas kernels. Os kernels podem ser lineares, polinomiais, radiais ou sigmoides. Esse método pode ser utilizado na regressão ou categorização dos dados, a depender do problema em questão.

Para o núcleo $k(x,x') = [\Phi(x), \Phi(x')]$, com x' representando os dados em um espaço, tem-se que o mapeamento dos dados pode ser descrito como $\Phi: X \rightarrow H$ sendo que Φ representa o mapeamento, H é um espaço de dimensão elevada de mapeamento dos dados e X o espaço original. Nas Equações (25) - (28) são apresentados os núcleos usados em SVM. Neles γ representa o parâmetro de desvio do kernel que deve ser ajustado com amostragem de dados que caracteriza um aprendizado estatístico. Neste processo, um parâmetro de regularização C, que representa uma penalização na fase de treinamento, é utilizado para garantir que o processo de treino não gere perda da capacidade de generalização.

$$k(x,x') = \langle x,x' \rangle \quad (25)$$

$$k(x,x') = (\gamma\langle x,x' \rangle + r)^d \quad (26)$$

$$k(x,x') = exp(-\gamma\|x - x'\|^2) \quad (27)$$

$$k(x,x') = tanh(\gamma\langle x,x' \rangle + r) \quad (28)$$

RANDOM FOREST (RF)

Random Forest (RF) ou Floresta Randômica é uma coleção de preditores organizados em árvore de propósito ou decisão na qual cada nó da árvore representa um conjunto de condições utilizadas para separar os dados e assim atribuir equações descritivas mais precisas para possíveis valores descritos pelas equações a eles associados (Breiman, 2001). O processo de criação dos nós é chamado *bootstrap* ou reamostragem e trabalha com geração de amostras aleatórias para garantir a qualidade do modelo segundo inferências de natureza estatística.

Trata-se de um modelo não paramétrico em que os preditores, quando em conjuntos, geram boas aproximações, tanto para classificação como para regressão, geralmente realizada de maneira linear. Nos problemas de classificação, os preditores são combinados por votação, já nos problemas de regressão, realiza-se esse procedimento por média. O método RF pode ser descrito em três etapas (válidas para regressão e classificação) que se repetem ao longo de diferentes ciclos (Liaw & Wiener, 2002) :

a) Geração uma amostragem de *n* árvores de classificação para o dado original.

b) Para cada árvore de classificação é feita a geração de novas árvores seguindo a seguinte regra: cada nó da árvore de classificação é separado em novos preditores randomicamente segundo um parâmetro chamado *mtry*, o qual representa o valor máximo de ramificações a serem realizadas. Em seguida, a melhor versão (antes ou após as ramificações) de cada árvore da *Randon Forest* é selecionada.

c) As novas versões da RF com suas novas árvores são avaliadas em termos de qualidade e, caso elas gerem uma melhora no método, a nova RF agregando os novos preditores passa a ser a versão atual. Esse processo é repetido até o alcance do número máximo de árvores associadas à RF ou até que a precisão de interesse seja atingida.

Na Figura 20 é mostrada a representação de uma árvore de decisão (esquerda) e uma RF (direita), na qual cada árvore de decisão gera uma resposta e a resposta global é feita por média ou votação, a depender se o modelo for de classificação ou regressão, conforme já explicado.

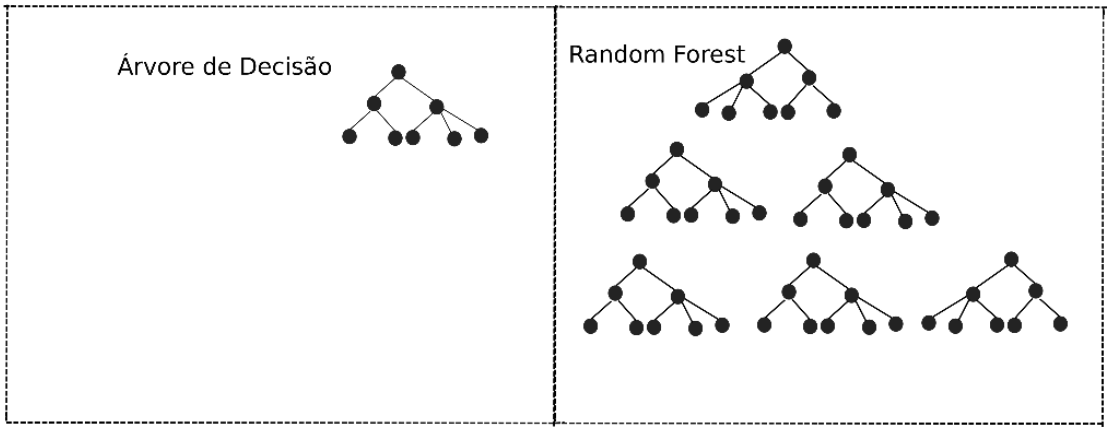

Figura 8: Representação dos métodos Árvore de decisão e *Random Forest*

REGRESSÃO LOGÍSTICA (RL)

Regressão Logística (RL) é um método de classificação ou determinação de uma variável binária no qual esta variável é associada, por regressão, a uma probabilidade dependente de uma ou mais variáveis independentes, sendo que esta probabilidade descreve se a variável binária se encontra ou não em uma dada classe.

Na regressão logística, a variável dependente y é associada a p (probabilidade de a variável categórica pertencer ou não a uma das classes). A Equação (29) representa uma RL sendo que x_k representa cada uma das k variáveis independentes do problema, e $y=\ln(p/(1-p))$ é chamado logaritmo da verossimilhança *(likelihood*). A verossimilhança pode ser descrita como a chance em favor do sucesso, conforme descrito na literatura (Pagano & Gauvreau, 2013). Isso pode ser exemplificado considerando, por exemplo, p=0,9, que fornece $p/(1-p)$ =9, significando que para p=0,9 o evento em questão tem 9 vezes mais chance de ocorrer do que se p fosse igual a 0,1.

$$y(X) = ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k \tag{29}$$

A descrição de p a partir da Equação (29) fornece a Equação (30), na qual a expressão por ela descrita constitui a função logística, que tem entre outras características o fato de gerar valores entre 0 e 1 independente dos coeficientes associados ao modelo de regressão logística.

$$p = \frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_k x_k)}} \tag{30}$$

Embora a RL seja semelhante aos modelos de regressão linear múltipla e ambos possam ser descritos como pertencentes a classe de modelos generalizados de regressão linear, a RL se difere de regressão linear em diversos aspectos. Entre estas diferenças podem ser citados: a) a não necessidade de distribuição normal dos erros, b) o erro associado à qualidade do ajuste não é avaliado pelo erro padrão ou variância do erro utilizado na regressão linear múltipla e sim pelo método da máxima verossimilhança, c) não há a necessidade de a relação entre uma variável independente e a variável dependente ser linear, d) a RL pode ser usada apenas para classificação e não para regressão (Pagano & Gauvreau, 2013; Richert & Coelho, 2013).

Na Figura 9 é mostrada a representação particular de uma RL com uma variável independente definida no intervalo entre -8,0 e 8,0. É possível observar na figura o formato característico de curva em S que caracteriza uma RL. Em termos de classificação (classes 1 e 0), o valor 1 é atribuído para todos os elementos com probabilidade *p* maiores que 0,5 e o valor 0 para valores de *p* menores que 0,5, ou seja, o valor 0,5 representa o valor limite que associado a classificação da resposta na classe 0 ou 1.

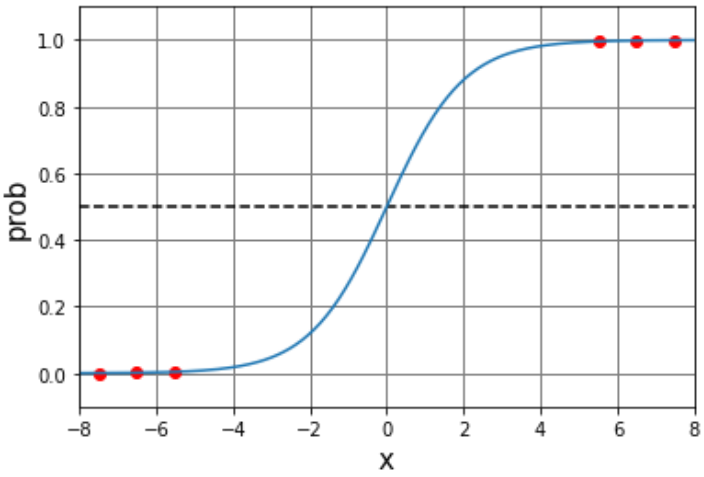

Figura 9: Representação da probabilidade associada ao modelo de Regressão Logística

ANÁLISE DE COMPONENTES PRINCIPAIS (PCA)

A análise de componentes principais (PCA) é uma das técnicas de transformação de variáveis que consiste na transformação linear de variáveis correlacionadas entre si em variáveis não correlacionadas entre si pelo uso dos autovalores e autovetores. As novas variáveis, ditas componentes principais, devem ser capazes de explicar a maior variação possível dos dados e não devem ser correlacionadas entre si, ou seja, devem ser linearmente independentes.

Nesta metodologia, um conjunto de *n* variáveis iniciais é transformado em *n* componentes principais, sendo que o método será tão mais efetivo quanto maior for a correlação entre as variáveis iniciais. Neste método, quanto menor número de componentes que seja capaz de explicar a maior parte da variabilidade dos dados melhor, uma vez que, dessa forma, o conjunto inicial de variáveis poderá ser reduzido para um conjunto menor de componentes sem que ocorra a perda significativa de informação associada aos dados.

Para um conjunto inicial de variáveis expressa pelo vetor *X* tem-se que os componentes principais *Z* a elas associadas podem ser obtidos de acordo com as Equações (60) a (64) conforme cita a literatura (Abdi & Williams, 2010). As equações (32) a (64) representam os três primeiros e o último dos componentes principais associados ao conjunto de *n* variáveis, sendo que a matriz *A* $(a_{11,}a_{12}, \dots, a_{nn})$ que associa os componentes às variáveis é conhecida como matriz de carregamentos ou cargas (*loadings*)

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix} \tag{31}$$

$$Z_1 = a_{11}X_1 + a_{12}X_2 + \dots + a_{1n}X_n \tag{32}$$

$$Z_2 = a_{21}X_1 + a_{22}X_2 + \dots + a_{2n}X_n \tag{33}$$

$$Z_3 = a_{31}X_1 + a_{32}X_2 + \dots + a_{3n}X_n \tag{34}$$

$$Z_n = a_{n1}X_1 + a_{n2}X_2 + \dots + a_{nn}X_n \tag{35}$$

As equações (65) a (39) mostram as condições necessárias para que a variabilidade dos dados se torne a maior possível.

$$a_{11}^2 + a_{12}^2 + a_{13}^2 + \cdots a_{1n}^2 = 1 \tag{36}$$

$$a_{21}^2 + a_{22}^2 + a_{23}^2 + \cdots a_{2n}^2 = 1 \tag{37}$$

$$a_{31}^2 + a_{32}^2 + a_{33}^2 + \cdots a_{3n}^2 = 1 \tag{38}$$

$$a_{n1}^2 + a_{n2}^2 + a_{n3}^2 + \cdots a_{nn}^2 = 1 \tag{39}$$

A garantia de máxima variabilidade e obtenção de variáveis linearmente independentes pode ser realizada através da matriz das covariâncias, *S* ou matriz das correlações *R* (os componentes podem ser calculados em relação a covariâncias ou coeficientes de correlação), e matriz diagonal dos autovetores como mostram as equações (40) a (48) nas quais λ representa os autovalores e v os autovetores e $\bar{v}$ a carga dos componentes principais que corresponde aos autovalores padronizados.

$$X^* = \begin{pmatrix} x_{11}^* & \cdots & x_{1n}^* \\ \vdots & \ddots & \vdots \\ x_{m1}^* & \cdots & x_{mn}^* \end{pmatrix} \tag{40}$$

$$x_{ij}^{*} = x_{ij} - \frac{\sum_{i=1}^{m} x_{ij}}{m} \tag{41}$$

$$A = X^{*t}.X^{*} \tag{42}$$

$$A = X^{*t}.X^{*} \tag{43}$$

$$\det(\mathrm{A} - \lambda \mathrm{I}) = 0 \tag{44}$$

$$Av = \lambda v \tag{45}$$

$$Q = \sum_{i=1}^{m} v_i^2 \tag{46}$$

$$Q = \sum_{i}^{n} v_i^2 \tag{47}$$

$$\bar{v}_i = \lambda_i \frac{v_i}{\sqrt{Q}} \tag{48}$$

CAPÍTULO 2 – DESENVOLVIMENTO DE UM NOVO MÉTODO DE SOLUÇÃO ATRAVÉS DA ASSOCIAÇÃO DE *SWARM INTELLIGENCE* E REDES NEURAIS ARTIFICIAIS

INTRODUÇÃO

A Evolução Diferencial ou *Differential Evolution* é um método de solução para problemas de otimização amplamente explorado, sendo considerado uma metodologia simples e eficiente, capaz de superar muitos algoritmos evolutivos e outros métodos heurísticos na resolução tanto de problemas benchmark e como de problemas de aplicação prática (Goudos et al., 2011; Moussa & Awotunde, 2018; S. Wang et al., 2019).

Este capítulo apresenta o desenvolvimento do otimizador FRANNK (*Full Referenced Artificial Neural Network Knowledge*), uma versão melhorada do *Differential Evolution,* na qual um método de solução baseado em inteligência de enxame (SI) é acoplado a uma Rede Neural Artificial (RNA) pré-treinada para realizar um ajuste dinâmico dos parâmetros do método.

Apesar da grande quantidade de métodos de solução presentes na literatura, vale ressaltar que não existe um algoritmo mais adequado para todos os tipos de situações. As respostas fornecidas por uma técnica específica podem ser adequadas para alguns problemas, no entanto, podem ser pouco eficientes para outro conjunto de aplicações. Nesse contexto, implementar novos algoritmos e aprimorar os existentes é sempre necessário para lidar com os complexos problemas de otimização com interesse prático (Wolpert & Macready, 1997).

METODOLOGIA

Nesta seção são descritas todas as etapas de desenvolvimento do otimizador FRANNK, incluindo o processo de seleção do algoritmo de ML mais adequado para ajustar os parâmetros *F* e *CR*; o algoritmo de geração de dados para treinamento e teste dos modelos (o qual necessitou de dois otimizadores DE trabalhando simultaneamente para encontrar um conjunto de valores otimizados dos parâmetros *F* e *CR* para determinado problema); a maneira de acoplar o modelos de ajuste de *F* e *CR* ao otimizador DE; bem como a maneira de avaliar a eficácia do otimizador FRANNK.

ESCOLHA DO ALGORITMO DE ML PARA AJUSTE DOS PARÂMETROS *F* E *CR*

A fim de gerar os modelos de ajuste dos parâmetros *F* e *CR*, realizou-se de um estudo comparativo entre quatro métodos de *Machine Learnig*: Redes Neurais Artificiais (RNA), *Randon*

Forest (RF), Máquina de Vetor Suporte (SVM, *Support Vector Machine*) e Regressão Logística (RL). Esses modelos foram escritos em python com auxílio da biblioteca Scikit-Learn (Den et al., 2020), usando parâmetros otimizados pela biblioteca GridSearchCV (Den et al., 2020). Foram avaliadas diversas métricas de desempenho, entre elas, acurácia, precisão, *recall, e*specificidade e F1 score, e baseado no desempenho desses indicadores, selecionou-se as Redes Neurais Artificiais (RNA) como método de ML a ser acoplado ao otimizador FRANNK, método que será mais bem descrito a seguir. Após a seleção da RNA como método de ML, optou-se, para melhor flexibilidade e integração dos códigos, pela substituição da RNA da biblioteca Sckit-Learn por uma RNA implementada em Python com uma única camada oculta, a qual foi treinada utilizando uma taxa de aprendizagem de 10^{-3} em 20000 épocas, sendo que os detalhes desta RNA são apresentados no próximo tópico.

REDE NEURAL ARTIFICIAL

O ajuste dos parâmetros, *F* e *CR*, foi descrito por RNAs independentes (RNA- F e RNA- CR) para cada um dos parâmetros. Os modelos de classificação propostos usaram RNAs do tipo *feedforward* contendo três camadas (camadas de entrada, oculta e de saída) que avaliam se os parâmetros do DE devem ser aumentados ou diminuídos ao longo das iterações. A presente topologia de RNAs composta por 3 camadas foi selecionada em decorrência do número limitado de pontos para treino e teste das Redes Neurais Artificiais (126 pontos).

Foram testadas, em um processo de *grid search,* diferentes quantidades de neurônios na camada oculta (3 a 20 neurônios), diferentes funções de ativação (purelin, sigmóide e tansig), bem como foram selecionadas as variáveis de entrada (entre as Equações (50) a (56)) que melhor representam a variação dos parâmetros. No processo de seleção, o número de neurônios na camada oculta e as funções de ativação ideais foram encontrados usando o algoritmo descrito na Figura 10 em combinação com o valor da Acurácia do modelo, descrita pela Equação(5). O número ótimo de neurônios da camada oculta obtido foi igual a 9 para ambas RNA-F e RNA-CR, enquanto para a camada de saída foi selecionada a função de ativação purelin, e na camada oculta foi usada a função tansig. As RNAs para ajuste de parâmetros foram implementadas em Python usando a Equação (49) para normalizar os valores das variáveis entre -1 e +1. O treinamento foi realizado usando algoritmo *backpropagation* e seguindo o método do gradiente descendente para ajuste de pesos.

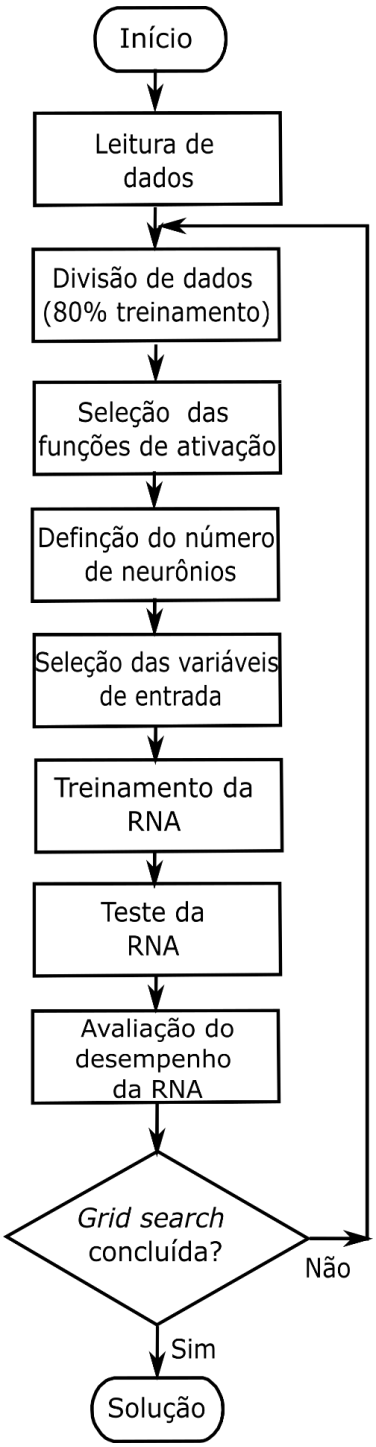

Figura 10: Fluxograma de treinamento e teste da Rede Neural Artificial

$$u_i = 2\left(\frac{z_i - \frac{b-a}{2}}{b-a}\right) \tag{49}$$

onde *i* é o número de série do conjunto de dados, *zi* é o valor atual da variável, *ui* é a variável codificada no intervalo -1 a +1, *b* e a representam, respectivamente, os valores máximo e mínimo de cada variável no conjunto de dados.

GERAÇÃO DE DADOS PARA TREINAMENTO E TESTE DAS RNAS

Os dados usados para o desenvolvimento das RNAs foram obtidos usando um conjunto de variáveis de entrada definidas pelas Equações (50) a (56) e algumas funções *benchmark* descritas na Tabela 1. Nas Equações (50) e (51), os valores *DI* e DI_R são medidas da diversidade populacional, com *DI* e DI_R representando, respectivamente, a diversidade em sentido absoluto e relativo; *RV* e *Delta* são valores qualitativos que descrevem a rapidez com que o valor da função objetivo muda, considerando os melhores valores (f_i e f_{i+1}) encontrados em duas interações consecutivas; FI_R está relacionado à fração do número total de iterações já concluídas; F_0 e CR_0 representam, respectivamente, os valores do fator de mutação e da taxa de crossover da iteração anterior.

$$DI = \left[\frac{\sum_{j=1}^{D} \sum_{i=1}^{NP} \left(X_{ij} - \frac{\sum_{i=1}^{NP} X_{ij}}{NP} \right)^2}{D} \right]^{\frac{1}{2}} \tag{50}$$

$$DI_R = \frac{DI}{DI_0} \tag{51}$$

$$RV = \begin{cases} 0\ if & |f_{g+1}| = |f_g| \\ 1\ if & |f_{g+1}| < |f_g| \leq 2\,.\,|f| \\ 2\ if & 2\,.\,|f_{g+1}| < |f_g| \end{cases} \tag{52}$$

$$Delta = RV_{g+1} - RV_g \tag{53}$$

$$FI_R = \frac{g}{Total} \tag{54}$$

$$CR_0 = CR_{g-1} \tag{55}$$

$$F_0 = F_{g-1} \tag{56}$$

onde D representa o número de variáveis ou dimensão da função objetivo; *g* é a iteração atual; *Total* representa o número total de gerações; *NP* é o tamanho da população.

Tabela 1: Funções *Benchmark* usadas para obtenção de dados de treinamento e teste dos modelos de ajustes de parâmetros acoplados ao otimizador FRANNK.

Função	**Equação**	**Mínimo**	**Tipo**	**Limites**
SeiCo*	$f_1 = 100.(x_D - x_{D-1}^2)^2 + (x_{D-1} - 1)^2$	0	U,NS	$[-10,10]^D$
Sphere	$f_2 = \sum_{i=1}^{D} {x_i}^2$	0	U,S	$[-5.12,5.12]^D$
Rastrigin	$f_3 = 10D + \sum_{i=1}^{D} [{x_i}^2 - 10\ \cos(2\pi x_i)]$	0	M,S	$[-5.12,5.12]^D$
Schwefel 2.26 **	$f_4 = a\ D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lceil x_i \rceil})$	0	M,S	$[-500,500]^D$
Sum of diff. powers	$f_5 = \sum_{i=1}^{D} \lvert x_i \rvert^{i+1}$	0	U,S	$[-1,1]^D$
Ackley	$f_6 = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + \exp(1)$	0	M,S	$[-32,32]^D$
Michalewicz	$f_7 = -\sum_{i=1}^{D} \sin(x_i)\sin^{20}\left(\frac{i{x_i}^2}{\pi}\right)$	b***	M,S	$[0,\pi]^D$

Nota: *nova função proposta no presente trabalho; ***a*=418.982887272433799807913601398; ****b*= - 9.66 para D=10 e *b*=-29.63 para D=30.

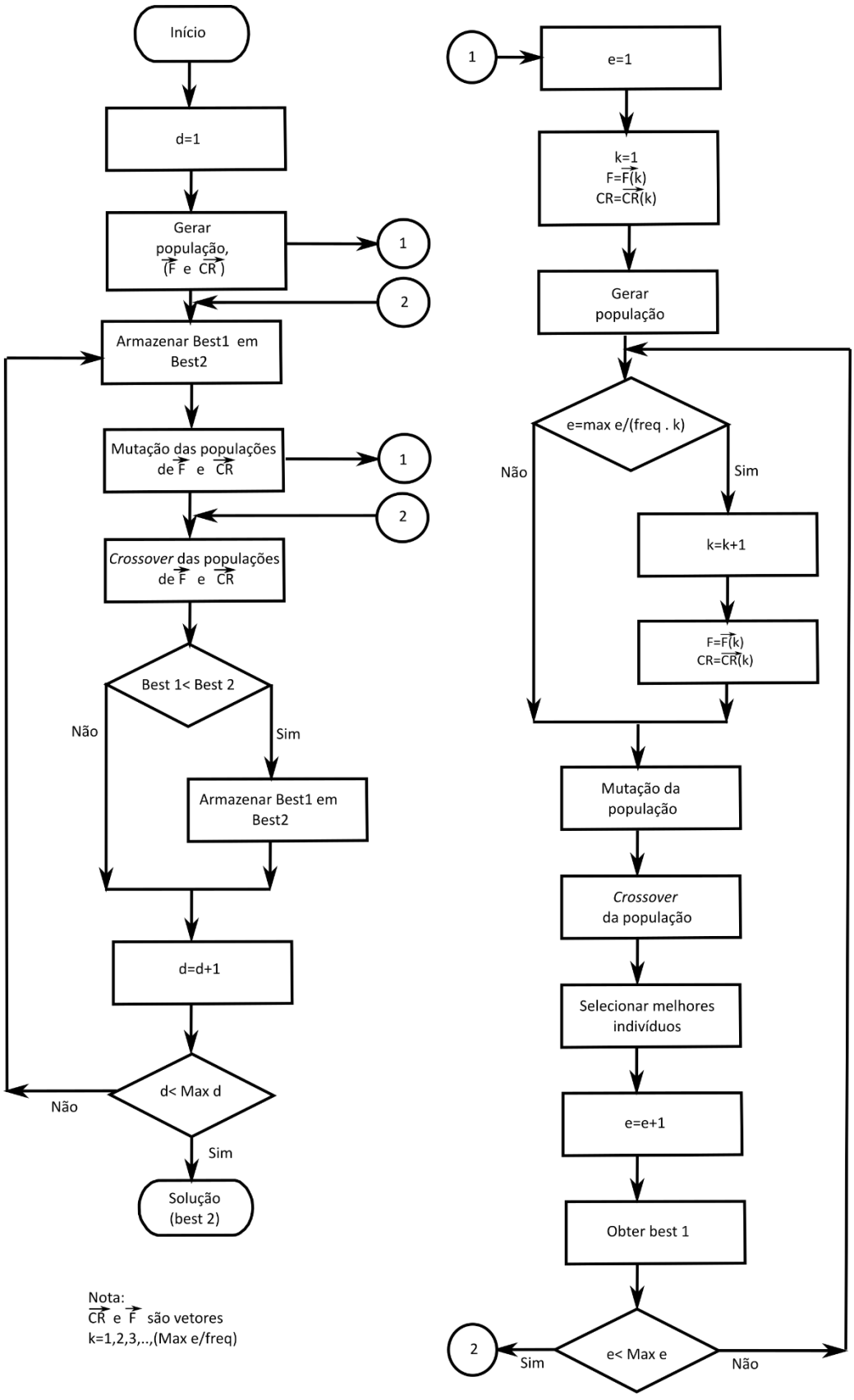

Figura 11: Fluxograma do Meta-Otimizador usado para obter as entradas e saídas necessárias para o desenvolvimento das RNAs de ajuste de parâmetros.

A Figura 11 descreve a estrutura do algoritmo que foi usado na geração de dados para o treinamento e teste das RNAs. Pode-se observar que dois otimizadores DE trabalhando simultaneamente foram necessários para encontrar um conjunto de valores otimizados de *F* e *CR* para determinado problema. Cada indivíduo no primeiro otimizador (Figura 11 à esquerda) está armazenando um conjunto de valores *F* e *CR* ($\vec{F}$ e $\overrightarrow{CR}$) que serão usados pelo segundo otimizador (Figura 11 à direita) para otimizar as funções *benchmark*. Durante o processo de otimização, o segundo otimizador fornece uma solução otimizada ao selecionar progressivamente os valores de *F* e *CR* fornecidos pelo primeiro otimizador, seguindo uma frequência (*Freq*) de ajuste de *F* e *CR*. O número máximo de iterações em cada *solver* DE ("*Max d*" e "*Max e*") deve ser selecionado de acordo com a dimensão, os limites e a complexidade de cada problema.

DESCRIÇÃO E AVALIAÇÃO DO OTIMIZADOR FRANNK

A Figura 12 mostra a estrutura do otimizador FRANNK usado para aumentar ou diminuir *F* e *CR* ao longo das iterações. As Equações (57) e (58) mostram como o otimizador FRANNK ajusta cada um dos parâmetros, seguindo a direção fornecida pela avaliação das RNAs.

$$P = \begin{cases} P_0(1 - Fd) \text{ para ajuste de incremento} \\ P_0(1 + Fd) \text{ para ajuste de decremento} \end{cases} \quad (57)$$

$$P = P_0\,(1 - m) + P\,.m \quad (58)$$

onde P é o parâmetro ajustado (representando ou *F* ou *CR*), P_0 é o valor de *F* e *CR* antes de cada correção, *Fd* é a constante de atualização e m é uma constante positiva entre 0 e 1, usada como um filtro para evitar mudanças bruscas. Nos testes apresentados nesta tese, foram usados *Fd* = 0,10 (que representa uma alteração de 10% da amplitude do domínio do parâmetro) e *m* = 0,25 (valor que faz com que o valor do parâmetro da geração atual seja uma média ponderada cujos coeficientes são 0,75 e 0,25 relativos respectivamente ao valor do parâmetro na iteração anterior, e ao novo valor de parâmetro obtido pela Equação (57).

Nos resultados de otimização obtidos usando o método FRANNK, os ajustes nos parâmetros foram feitos a cada geração seguindo a seleção do tipo de ajuste (incremento ou decremento) fornecida pelas RNAs em conjunto com as Equações (57) e (58), com a possibilidade de aumentar os parâmetros até o limite de 1 e diminuir até o limite de 0,05. Os valores definidos para os parâmetros no início da otimização foram $F_0 = CR_0 = 0{,}50$, que representa o ponto médio do domínio dos parâmetros.

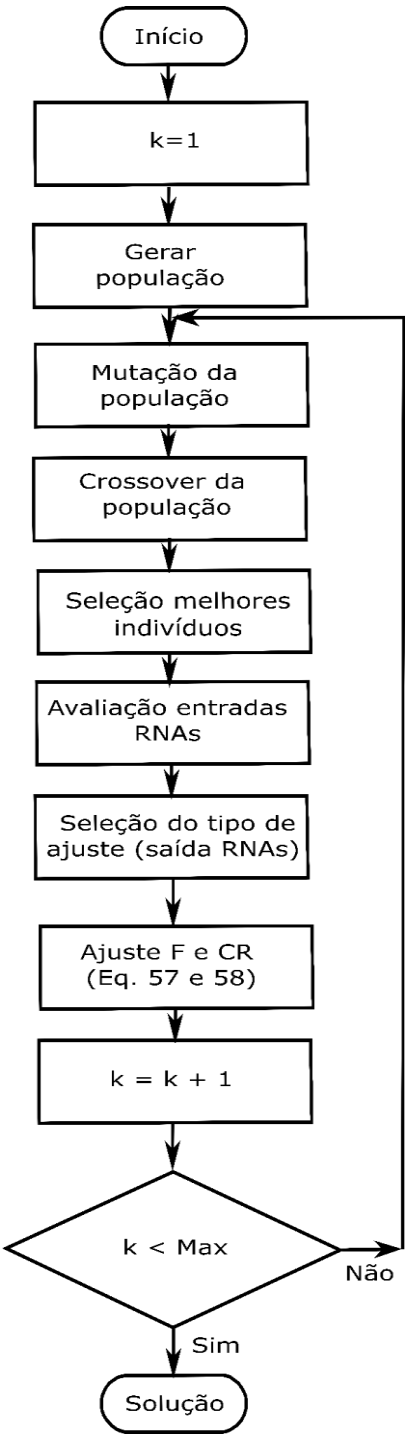

Figura 12: Fluxograma que descreve o otimizador FRANNK.

A Tabela 2 mostra as funções *benchmark* (diversas daquelas usadas para treinamento e teste da RNA) que foram usadas para avaliar o otimizador FRANNK.

Tabela 2: Funções *Benchmark* empregadas para avaliar o otimizador FRANNK.

Função	Equação	Mínimo	Tipo	Limites
Schwefel 2.22	$f_8 = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	0	U,NS	$[-10,10]^D$
Schwefel 2.21	$f_9 = \max \lvert x_i \rvert \quad i = 1,2,\ldots,D$	0	U, S	$[-100,100]^D$
Alpine n1	$f_{10} = \sum_{i=1}^{D} \lvert x_i . \sin(x_i) + 0.1 . x_i \rvert$	0	M,S	$[-10,10]^D$
Styblinski-Tang	$f_{11} = \frac{1}{2} \sum_{i=1}^{D} (x_i^4 - 16 . x_i^2 + 5 . x_i)$	a.D*	M,S	$[-5,5]^D$
Griewank	$f_{12} = 1 + \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos\left(\frac{x_i}{\sqrt{i}}\right)$	0	M,NS	$[-600,600]^D$
Rosenbrock	$f_{13} = \sum_{i=1}^{D} [100 . (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	0	U,NS	$[-10,10]^D$
Step	$f_{14} = \sum_{i=1}^{D} \lvert x_i + 0.5 \rvert^2$	0	U,S	$[-100,100]^D$
Sum Squares	$f_{15} = \sum_{i=2}^{D} i . x_i^2$	0	M,S	$[-10,10]^D$
Bent Cigar	$f_{16} = x_1^2 + 10^6 \sum_{i=1}^{D} x_i^2$	0	U,S	$[-100,100]^D$
Shubert 3	$f_{17} = \sum_{i=1}^{D} \sum_{j=1}^{5} j . sin((j+1) . x_i + j)$	-186.7389	M,S	$[-10, 10]^D$
Noisy Quartic	$f_{18} = \sum_{i=1}^{D} i . x_i^4 + random[0,1)$	0	M,S	$[-1.28, 1.28]^D$

Nota: * a=-39.16599

O impacto das variáveis de entrada apresentadas no otimizador FRANNK foi avaliado em termos de efeito relativo usando o método de particionamento, conforme descrito na Equação (59):

$$V_i = \frac{\sum_{j=1}^{h} \left[\left(\frac{\lvert IW_{ij} \rvert}{\sum_{k=1}^{m} \lvert IW_{kj} \rvert} \right) \lvert LW_j \rvert \right]}{\sum_{i=1}^{m} \left\{ \sum_{j=1}^{h} \left[\left(\frac{\lvert IW_{ij} \rvert}{\sum_{k=1}^{m} \lvert IW_{kj} \rvert} \right) \lvert LW_j \rvert \right] \right\}} \tag{59}$$

onde V_i é o efeito relativo da variável de entrada i, m e h são os números de neurônios na camada de entrada e camada oculta, respectivamente, e IW e LW são os pesos da conexão da camada oculta e da camada de saída, respectivamente.

PROBLEMAS TESTE

Testes de performance e confiabilidade de algoritmos de otimização são frequentemente realizados usando um conjunto funções *benchmark* presentes na literatura. Conforme está descrito na Tabela 1 e Tabela 2, nesta pesquisa foram usadas 18 funções benchmark para avaliar tanto a RNA, como o desempenho do otimizador FRANNK. Nesta seção serão fornecidos os gráficos representando a região de busca para cada um destes problemas teste avaliados com D=2. Conforme pode ser observado da Figura 9 a Figura 30 representadas a seguir, foram selecionados problemas com diferentes características entre si, a fim de obter uma visão global do desempenho da estratégia de solução desenvolvida.

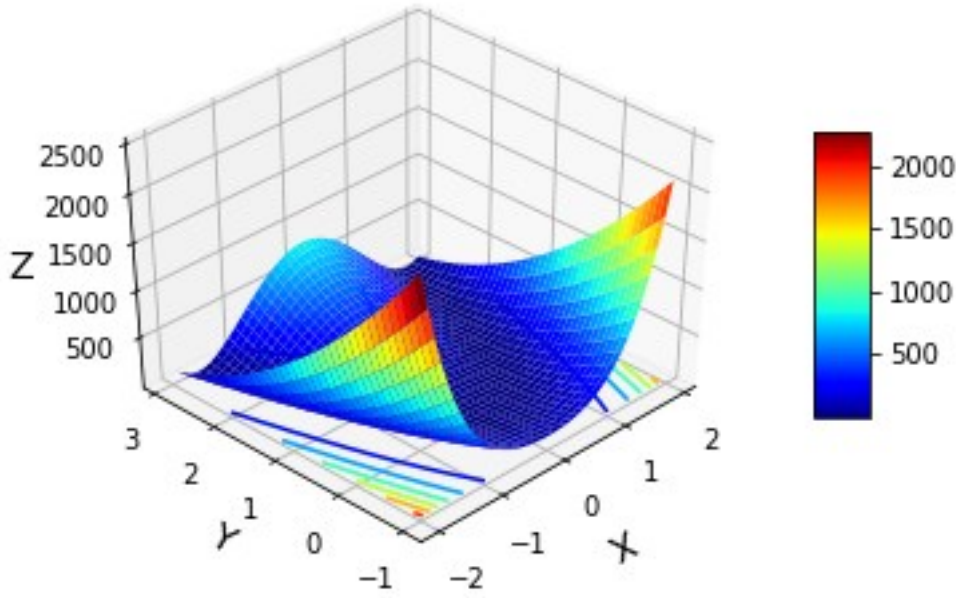

Figura 13: Função Seico

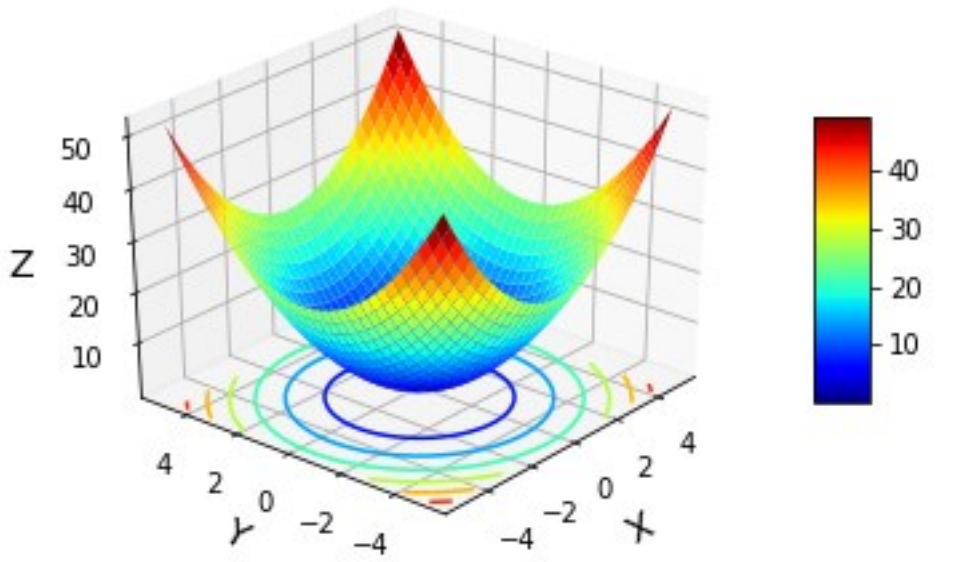

Figura 14: Função Sphere

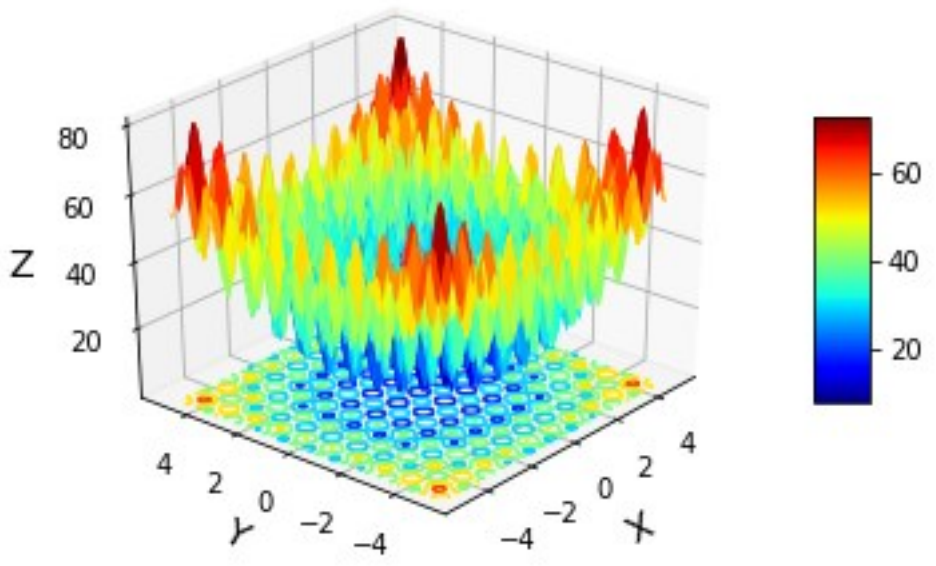

Figura 15: Função Rastrigin

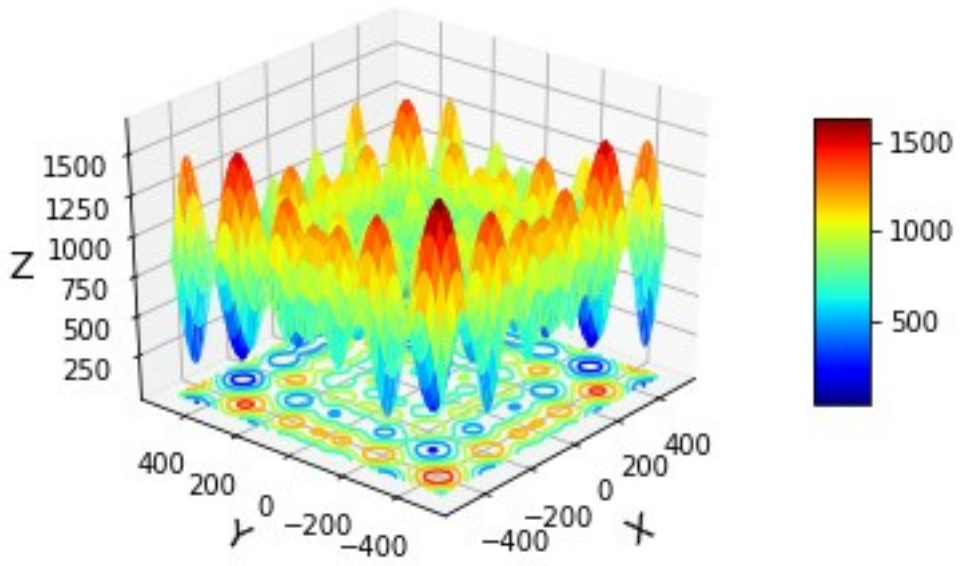

Figura 16: Função Schwefel 2.26

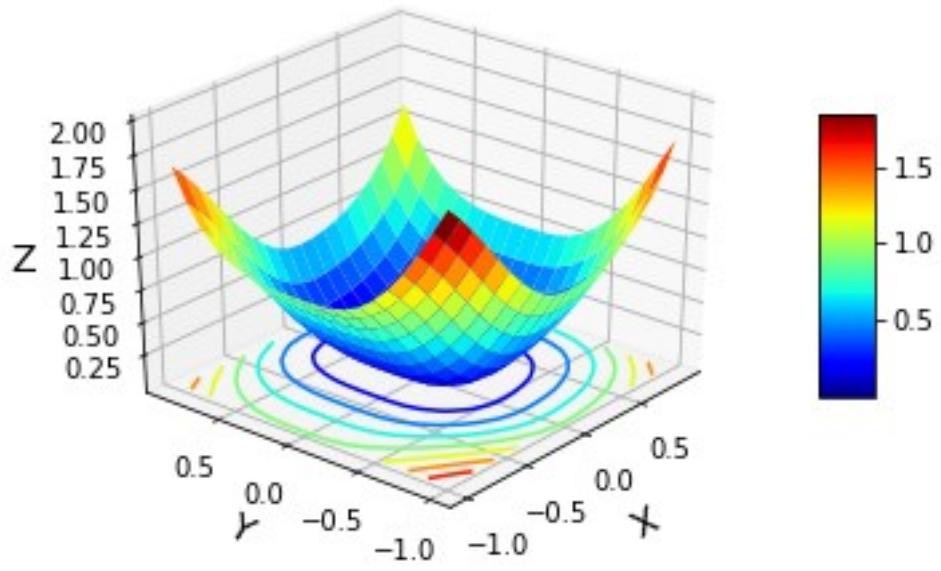

Figura 17: Função Sum of diff. powers

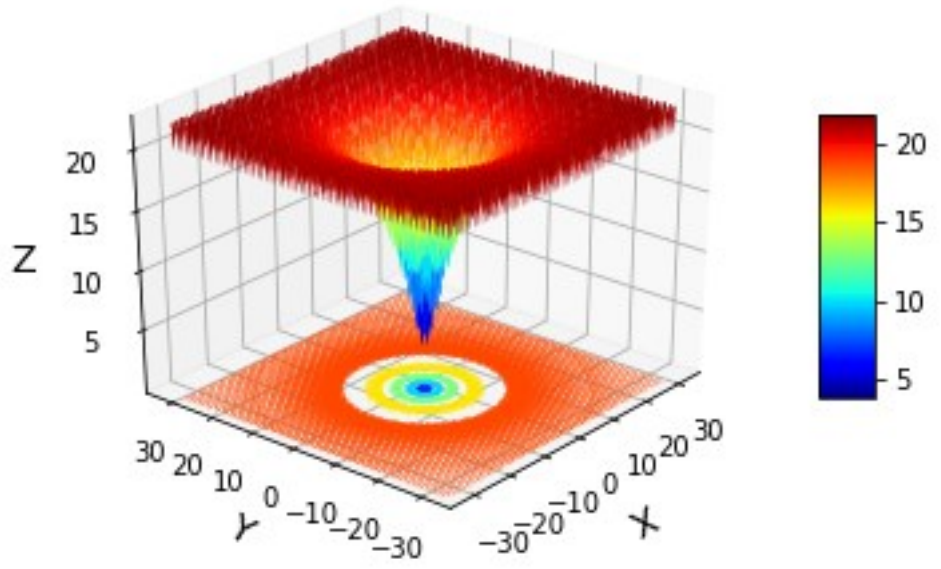

Figura 18: Função Ackley

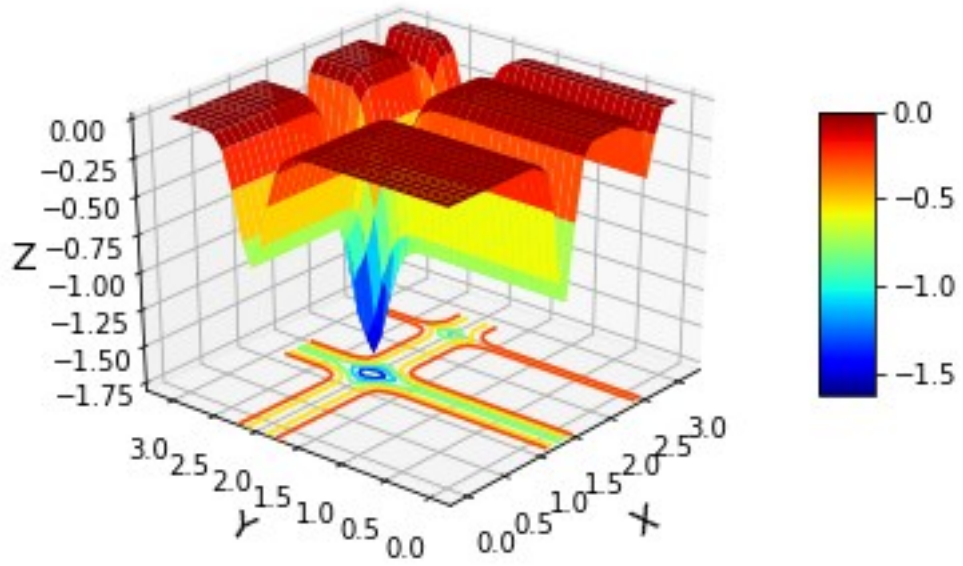

Figura 19: Função Michalewicz

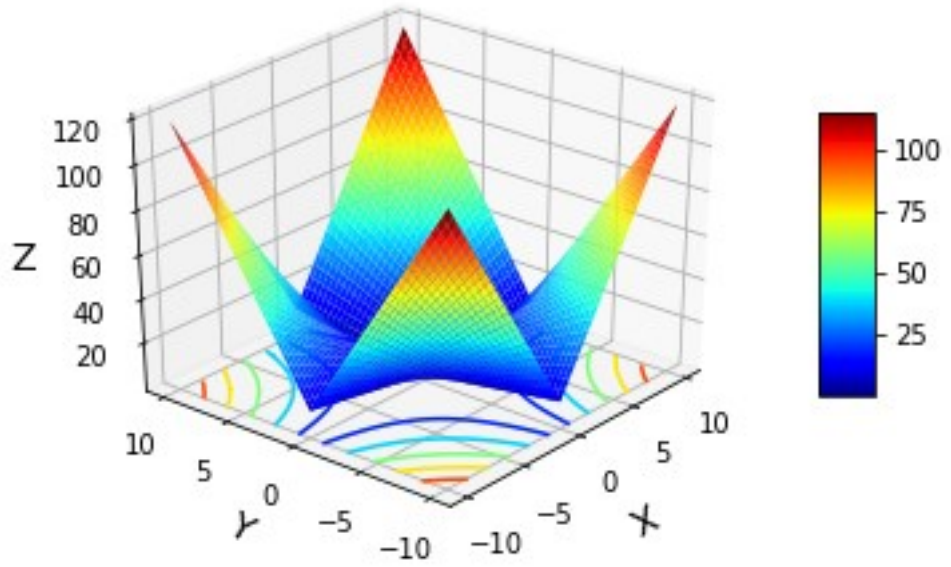

Figura 20: Função Schwefel 2.22

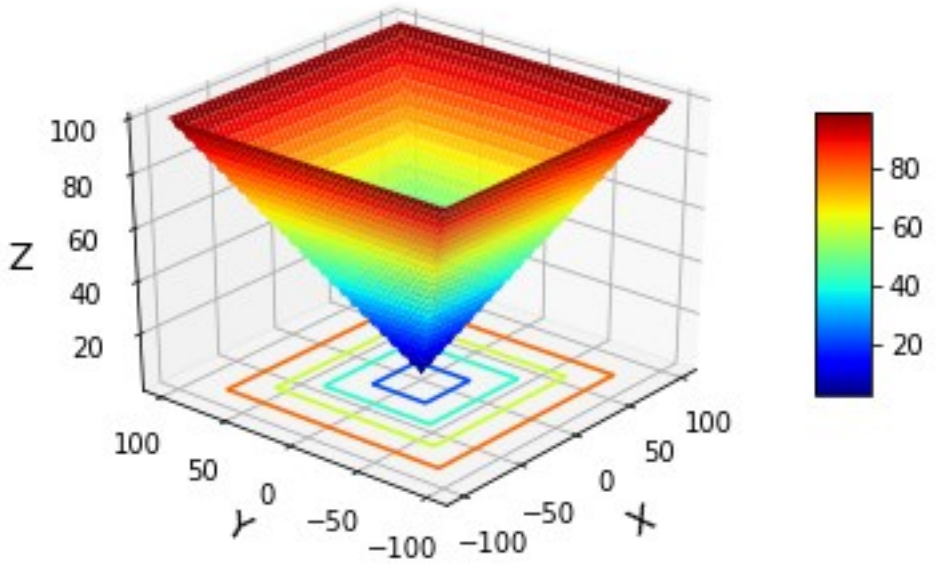

Figura 21: Função Schwefel 2.21

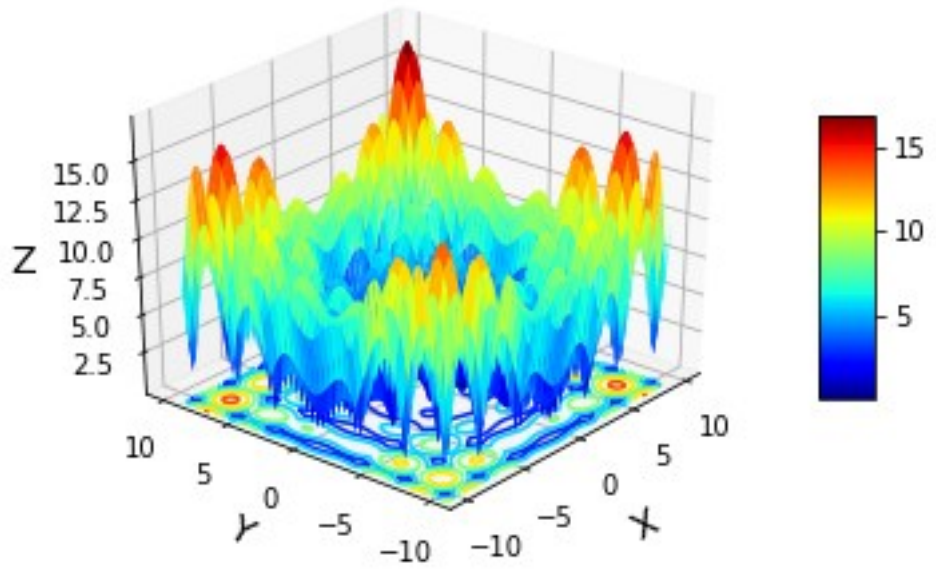

Figura 22: Função Alpine n1

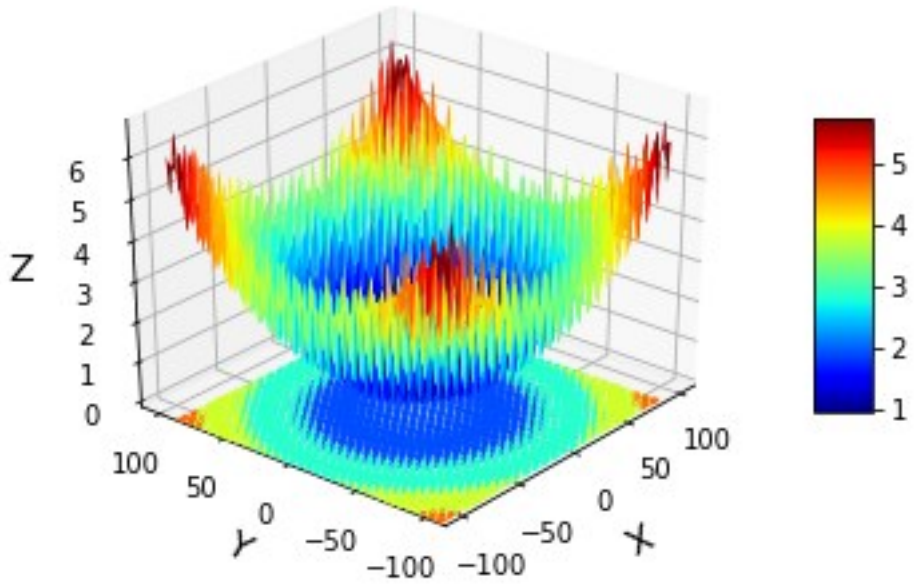

Figura 23: Função Styblinski-Tang

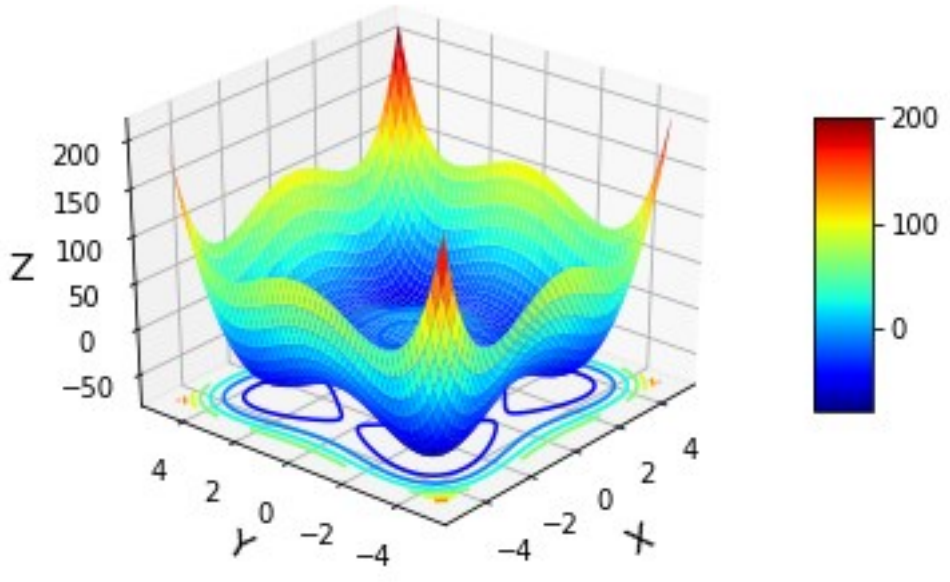

Figura 24: Função Griewank

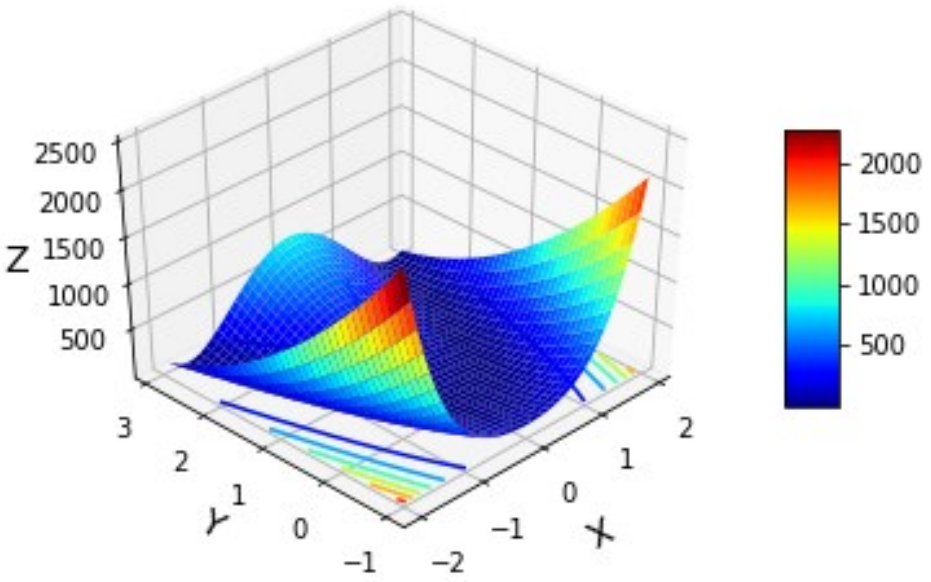

Figura 25: Função Rosenbrock

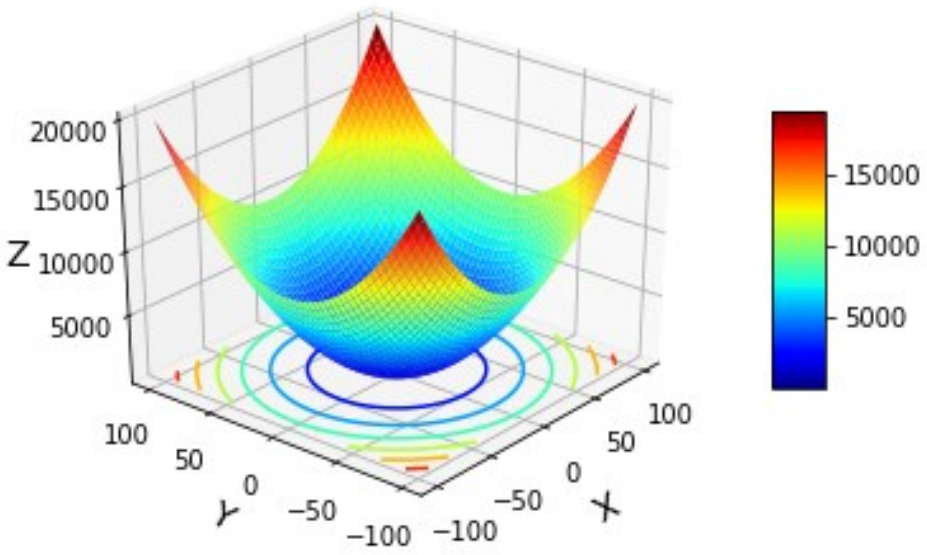

Figura 26: Função Step

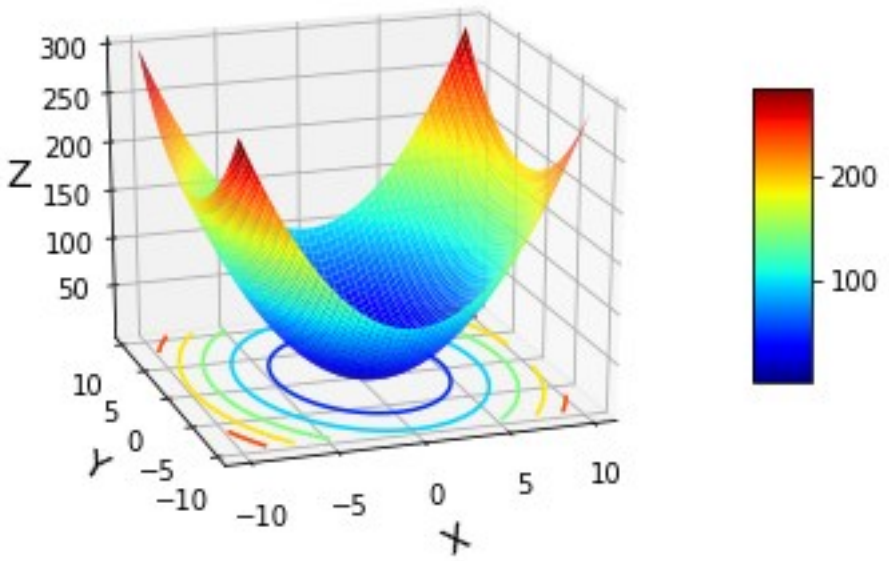

Figura 27: Função Sum Squares

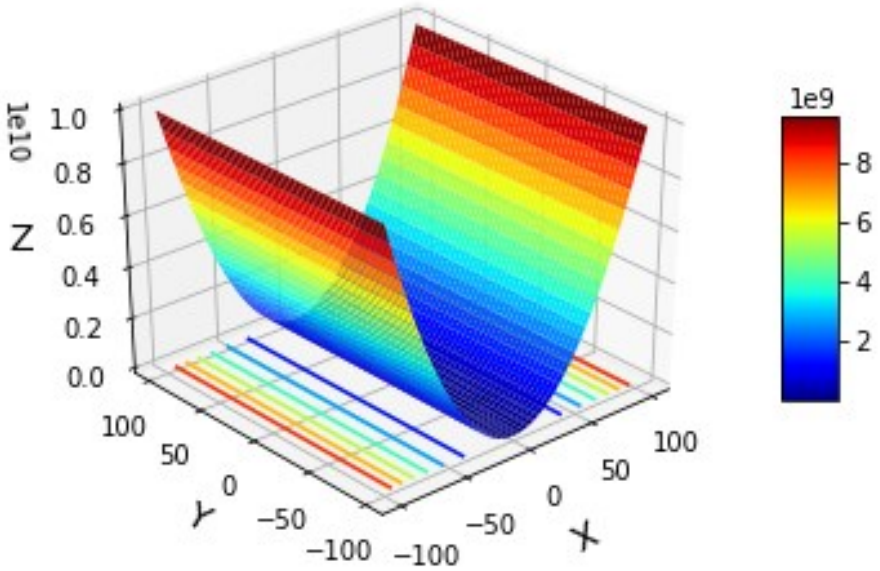

Figura 28: Função Bent Cigar

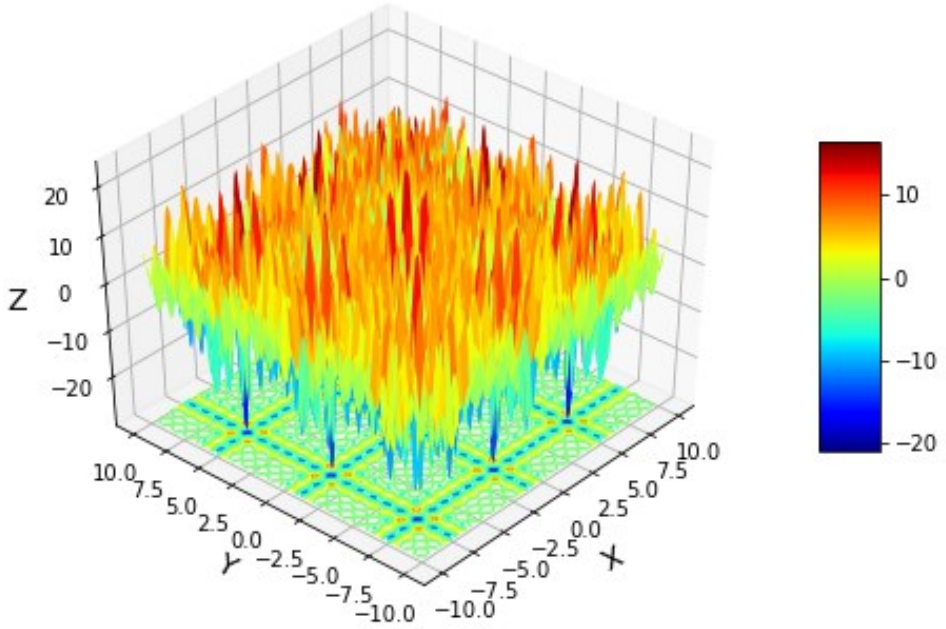

Figura 29: Função Shubert 3

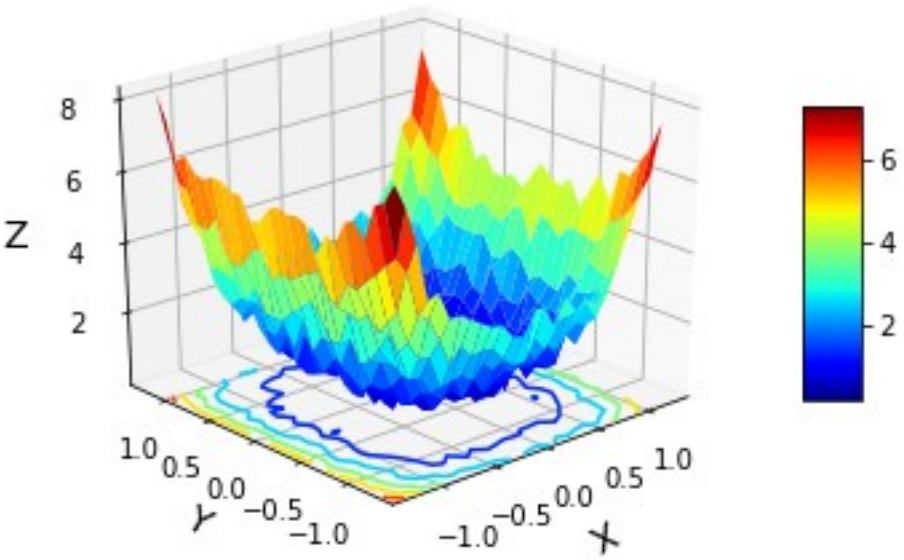

Figura 30: Função Noisy Quartic

RECURSOS COMPUTACIONAIS E CONDIÇÕES DOS ENSAIOS

A fim de testar a eficiência do método de solução proposto, as funções de benchmark apresentadas nas Tabelas 1 e 2 foram testadas para problemas de dimensão 10 e 30. Cada função foi avaliada 50 vezes e os resultados foram expressos em termos de média (Avg) e desvio padrão (Sdt). Todos os algoritmos descritos nos fluxogramas foram implementados em Python 3.7.4 pela autora e os resultados relativos ao métodos JADE e jDE foram implementados, respectivamente, em linguagem Python pelo *toolbox* PyFDE (Negri, 2020) e Linguagem R pelo *toolbox* DeoptimR (Conceicao & Maechler, 2020).

O tamanho da população e número de iterações foram selecionados de maneira a chegar a valores próximos ao ótimo global para a maioria dos problemas testados. Para todos os testes, o tamanho da população usado foi 100 para ambas as dimensões (D=10 e D=30). O número total de iterações ou gerações foi de 2.000 para ambas as dimensões, exceto nos resultados presentes na Tabela 14 (referente a comparação de resultados encontrados na literatura), em que o número total de iterações está descrito na própria tabela. Nas tabelas de resultados, todos os valores menores que 1E-20 foram considerados igual a zero.

RESULTADOS E DISCUSSÕES

SELEÇÃO DO ALGORITMO DE ML

Da Tabela 3 à Tabela 6 são mostradas as métricas de qualidade associadas a cada método de ML referente aos modelos de ajuste dos parâmetros *F* e *CR*, enquanto a Tabela 7 faz uma comparação em termos de acurácia, uma boa métrica de desempenho global para modelos de ML, para cada um dos métodos de ML testados. Nota-se que, de maneira geral, os resultados obtidos foram relativamente similares, sendo que a RNA obteve melhor desempenho para o conjunto de teste no caso de ajuste do parâmetro *F*, e a SVM obteve melhor desempenho para o conjunto de teste no caso do ajuste do parâmetro *CR*. Para dar continuidade ao trabalho, tanto o modelo de RNA como o SVM poderia ter sido selecionado, porém, por uma questão de maior experiência prévia da autora na área de RNAs, estas foram então selecionadas como método de ML para o ajuste dinâmico de parâmetros do otimizador FRANNK.

Tabela 3: Avaliação do ajuste dos modelos de RNA (biblioteca Scikit-Learn)

	Modelo Ajuste F		**Modelo Ajuste CR**	
Medida	**F treino[a]**	**F teste[b]**	**CR treino[c]**	**CR teste[d]**
Acurácia	70.5%	68.4%	69.3%	55.3%
Precisão	72.2%	65.0%	68.4%	42.9%
Recall	61.9%	72.2%	81.3%	64.3%
Especificidade	78.3%	65.0%	55.0%	50.0%
F1 score	66.7%	68.49%	74.3%	51.4%

Tabela 4: Avaliação do ajuste dos modelos de Regressão Logística

	Modelo Ajuste F		**Modelo Ajuste CR**	
Medida	**F treino[a]**	**F teste[b]**	**CR treino[c]**	**CR teste[d]**
Acurácia	67.0%	63.2%	67.0%	50.0%
Precisão	73.9%	60.0%	52.5%	33.3%
Recall	66.7%	66.7%	67.7%	72.7%
Especificidade	67.6%	60.0%	66.7%	40.7%
F1 score	70.1%	63.1%	59.1%	45.7%

Tabela 5: Avaliação do ajuste dos modelos de *Randon Forest*

	Modelo Ajuste F		**Modelo Ajuste CR**	
Medida	**F treino[a]**	**F teste[b]**	**CR treino[c]**	**CR teste[d]**
Acurácia	77.3%	63.2%	80.7%	63.2%
Precisão	86.9%	65.0%	90.0%	70.8%
Recall	74.1%	65.0%	73.5%	70.8%
Especificidade	82.4%	61.1%	89.7%	50.0%
F1 score	80.0%	65.0%	80.9%	70.8%

Tabela 6: Avaliação do ajuste dos modelos de Máquina de Vetor Suporte

	Modelo Ajuste F		Modelo Ajuste CR	
Medida	**F treino[a]**	**F teste[b]**	**CR treino[c]**	**CR teste[d]**
Acurácia	72.7%	60.5%	69.3%	68.4%
Precisão	84.8%	70.0%	70.3%	68.2%
Recall	69.6%	61.1%	61.9%	75.0%
Especificidade	78.1%	60.0%	76.1%	61.1%
F1 score	76.5%	65.1%	65.8%	71.4%

Tabela 7: Acurácia associada a cada método de ML referente aos modelos de ajuste dos parâmetros *F* e *CR*

	Modelo Ajuste F		Modelo Ajuste CR	
Medida	**F treino[a]**	**F teste[b]**	**CR treino[c]**	**CR teste[d]**
RNA	70.5%	**68.4%**	69.3%	55.3%
RL	67.0%	63.2%	67.0%	50.0%
RF	77.3%	63.2%	80.7%	63.2%
SVM	72.7%	60.5%	69.3%	**68.4%**

DESEMPENHO DAS RNAS NO AJUSTE DE *F* E *CR*

A Tabela 8 apresenta as métricas de qualidade relativas aos modelos de RNAs para ajuste de *F* e *CR*, as quais foram construídas e otimizadas manualmente, sem auxílio de bibliotecas. Tem-se que essas RNAs representaram adequadamente os dados, atingindo pelo menos o valor de 80% de precisão e F1-score para ambos os conjuntos de dados de treinamento e teste. Acurácia e F1-score que são, respectivamente, as melhores métricas para descrever o desempenho global e o desempenho associado a modelos com classes não balanceadas, como é o caso dos modelos desta pesquisa, conforme explanado nas notas da Tabela 3.

Tabela 8: Avaliação do ajuste dos modelos de RNA (construídos e ajustados manualmente)

	RNA-F		RNA-CR	
Medida	**F treino[a]**	**F teste[b]**	**CR treino[c]**	**CR teste[d]**
Acurácia	89.1%	80.0%	91.1%	84.0%
Precisão	87.0%	81.3%	100.0%	85.7%
Recall	92.2%	86.7%	87.8%	94.7%
Especificidade	86.0%	70.0%	100.0%	50.0%
F1 score	89.5%	83.9%	93.5%	90.0%

Notas: a-*VP, FP, FN e VN* são, respectivamente, 47,7,4 e 43; b-*VP, FP, FN e VN* são, respectivamente, 13,3,2 e 7; c-*VP, FP, FN e VN* são, respectivamente, 65,0,9 e 27; d-*VP, FP, FN e VN* são, respectivamente, 18,3,1 e 3; *VP, FP, FN e VN* representam, respectivamente, verdadeiro positivo, falso positivo, falso negativo e verdadeiro negativo; positivo foi considerado redução no valor do parâmetro e negativo foi considerado aumento no valor do parâmetro; a matriz de confusão pode ser descrita como desequilibrada porque *VP* é consideravelmente maior que *VN* em todos os casos.

ESTUDO PRELIMINAR DO EFEITO DAS VARIÁVEIS DE ENTRADA DA RNA

A Figura 31 mostra o comportamento das variáveis de entrada da RNA definida pelas Equações (50) a (56), na resolução das funções benchmark apresentadas na Tabela 1, usando o otimizador DE tradicional. Pode-se observar que a diversidade tende a diminuir ao longo das iterações e as variáveis *RV* e *Delta* são capazes de detectar mudanças associadas a melhorias no valor ótimo durante as iterações.

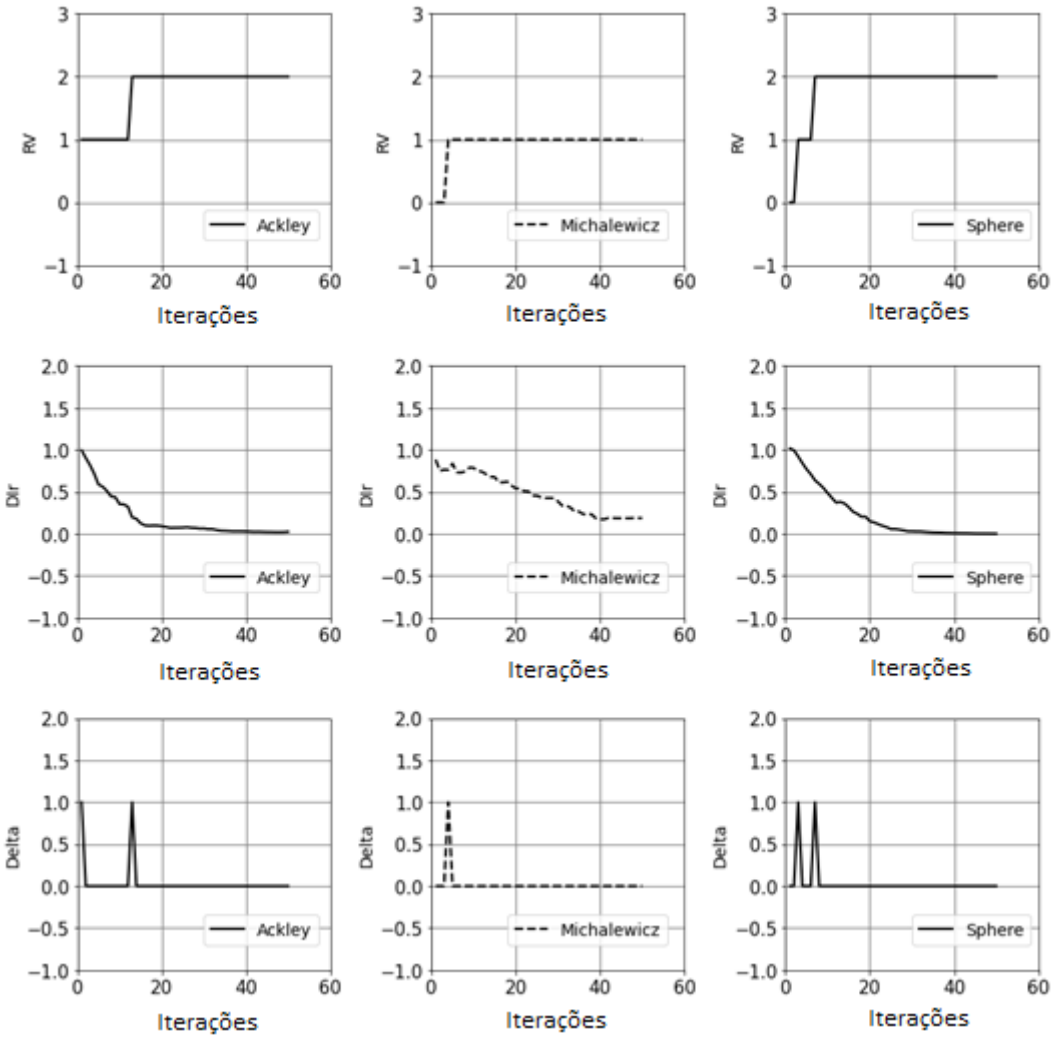

Figura 31: Evolução das variáveis de entrada da RNA ao longo das iterações medidas usando DE (dimensão = 5, população = 10, iterações = 50, F = 0,5 e CR = 0,5)

Todas as funções *benchmark* usadas no presente trabalho têm apenas um ótimo global, com todas as variáveis impactando no valor da função objetivo, com exceção da função *benchmark* SeiCo, que é uma nova função criada durante o presente trabalho. SeiCo é igual à função Rosenbrock para dimensão dois, mas difere-se desta para dimensão maior que dois, apresentando apenas duas variáveis impactando no ótimo. Além disso, o SeiCo difere-se de todas as outras funções em relação à diversidade, pois não apresenta redução significativa da diversidade ao longo das iterações, como pode ser observado na Figura 32.

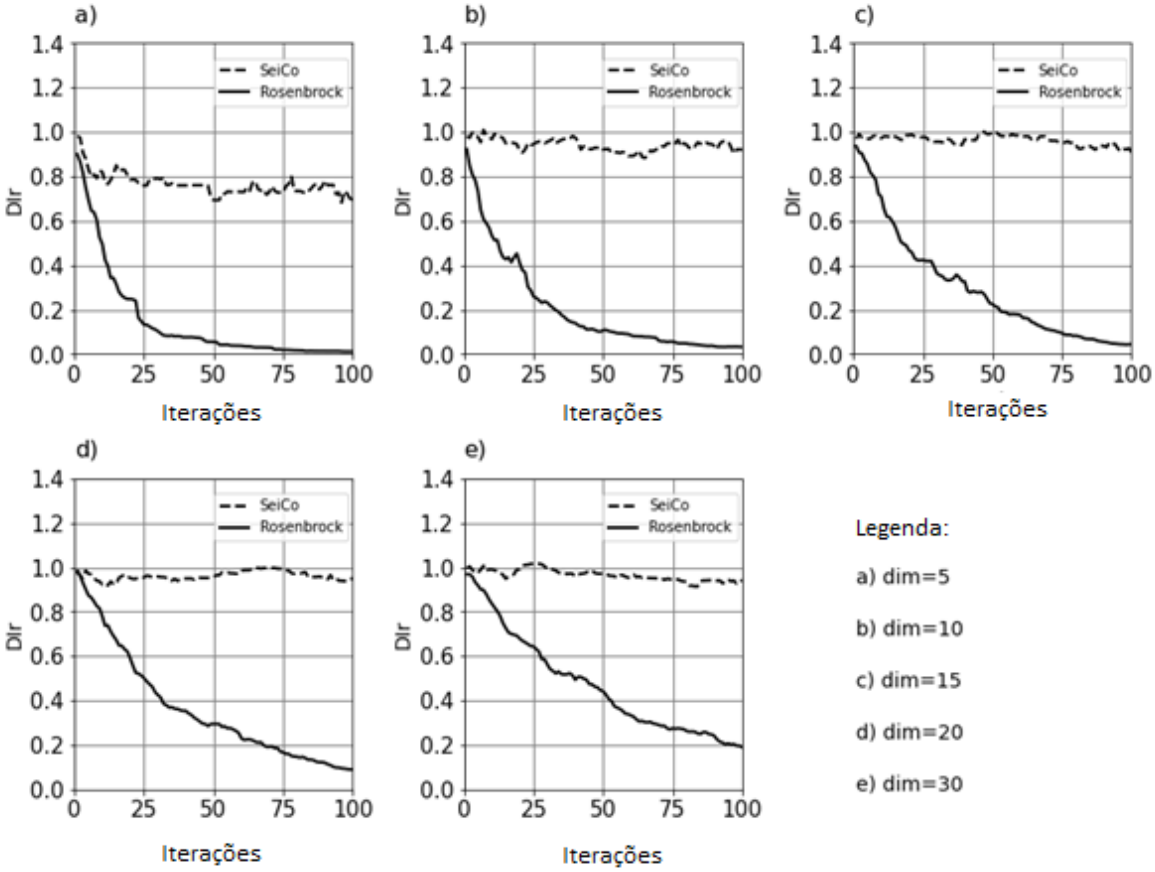

Figura 32: Comparação entre as funções Rosenbrock e a nova função SeiCo criada durante o presente trabalho (otimizador = DE, população = 20, iterações = 100, F = 0,5 e CR = 0,5)

DESENVOLVIMENTO DO OTIMIZADOR FRANNK

A Figura 33 mostra os resultados (conjuntos F e CR) fornecidos pelo meta-otimizador (algoritmo apresentado na Figura 11) para as funções de f1 a f7, as quais geraram uma quantidade total de 126 pontos. Esses dados foram usados em uma etapa subsequente para o desenvolvimento das RNAs, as quais forneceram modelos capazes de ajustar automaticamente F e CR no otimizador FRANNK ao longo das iterações. Conforme observado na Figura 33, a variação dos parâmetros apresentou um comportamento complexo e irregular para os parâmetros F e CR, o que pode ser explicado como resultado do movimento da população do otimizador DE por diferentes condições de diversidade e regiões do espaço de busca, enquanto a evolução prosseguia. Neste contexto, algumas configurações de parâmetros específicas podem ser mais eficazes do que outras, dependendo do estágio de convergência e do tipo de função.

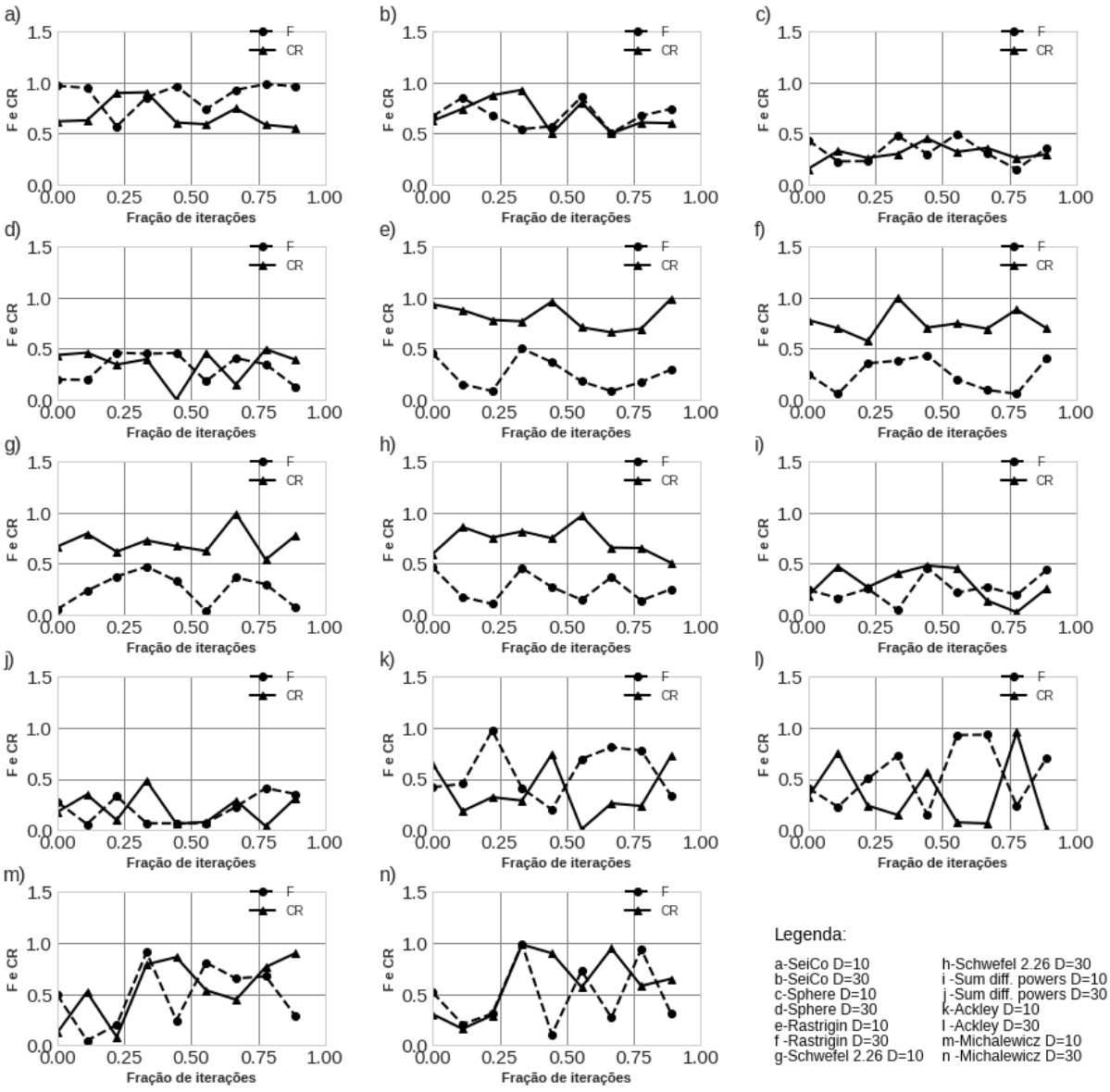

Figura 33: Valores otimizados de *F* e *CR* obtidos pelo meta-otimizador e empregados no desenvolvimento da RNA. Nota: D é a dimensão.

Conforme apresentado nos gráficos, o comportamento dos parâmetros ao longo das iterações seguiu diferentes padrões: a) Oscilação considerável dos valores de *F* e *CR* em todas as faixas [0,1], conforme observado nas funções Michalewicz e Ackley; b) valores de *F* e *CR* consideravelmente menores que um na maioria das iterações, como observado na funções unimodais Sphere e Sum of Different Powers; c) valores de *F* maiores que 0,5 combinado com valores de *CR* menores que 0,5, conforme observado nas funções Rastrigin e Schwefel. De acordo com R Storn, 1996, valores de *CR* próximos a um (entre 0,7 e 1,0) podem ser úteis quando a convergência não é facilmente alcançada, e pequenos valores de *CR* (valores entre 0,1 e 0,3) podem ser usados em outros casos. Discussões semelhantes podem ser feitas em relação ao parâmetro *F*.

A análise da variância do conjunto de dados, descrita em termos de DI_R, FI_R, F_0 e CR_0 (que são as variáveis de entrada das RNAs acopladas ao otimizador FRANNK) usando Análise de Componentes Principais (PCA), está apresentada na Figura 34 e na Figura 35. Conforme mostrado na Figura 34 a análise com o uso de dois PCAs pode fornecer uma boa descrição dos dados, uma vez que pelo menos 80% da variabilidade do conjunto de dados pode ser explicado por dois PCAs. De acordo com a Figura 35, algumas discussões podem ser feitas. Primeiramente, pela avaliação do tamanho das setas pode-se inferir quão fortemente as características (DI_R, FI_R, F_0 e CR_0) estão influenciando na variabilidade do conjunto de dados. É possível observar ainda que, para cada função, a importância relativa de cada variável não foi a mesma. Por exemplo, a diversidade teve baixo impacto nas funções SeiCo, Schewefel 2.26 e Michalewicz, porém, teve alto impacto nas demais funções. Além disso, considerando a posição relativa das setas e o ângulo entre elas, observa-se que em quase todos os casos há uma correlação negativa (representada por setas em lados opostos da origem do gráfico) entre a diversidade populacional relativa (DI_R) e a fração do número total de iterações já concluídas (FI_R), o que é esperado em um processo de convergência.

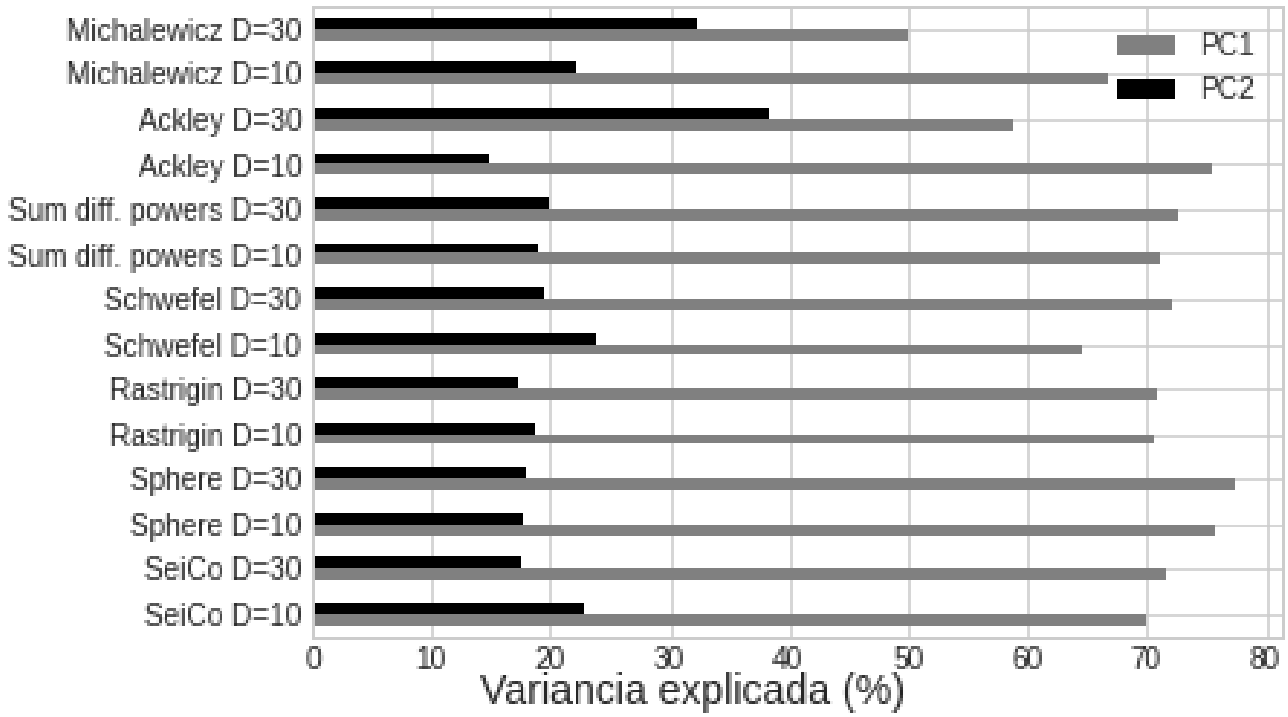

Figura 34: Variância que pode ser explicada pelos dois primeiros componentes principais do conjunto de dados. Nota: D é a dimensão.

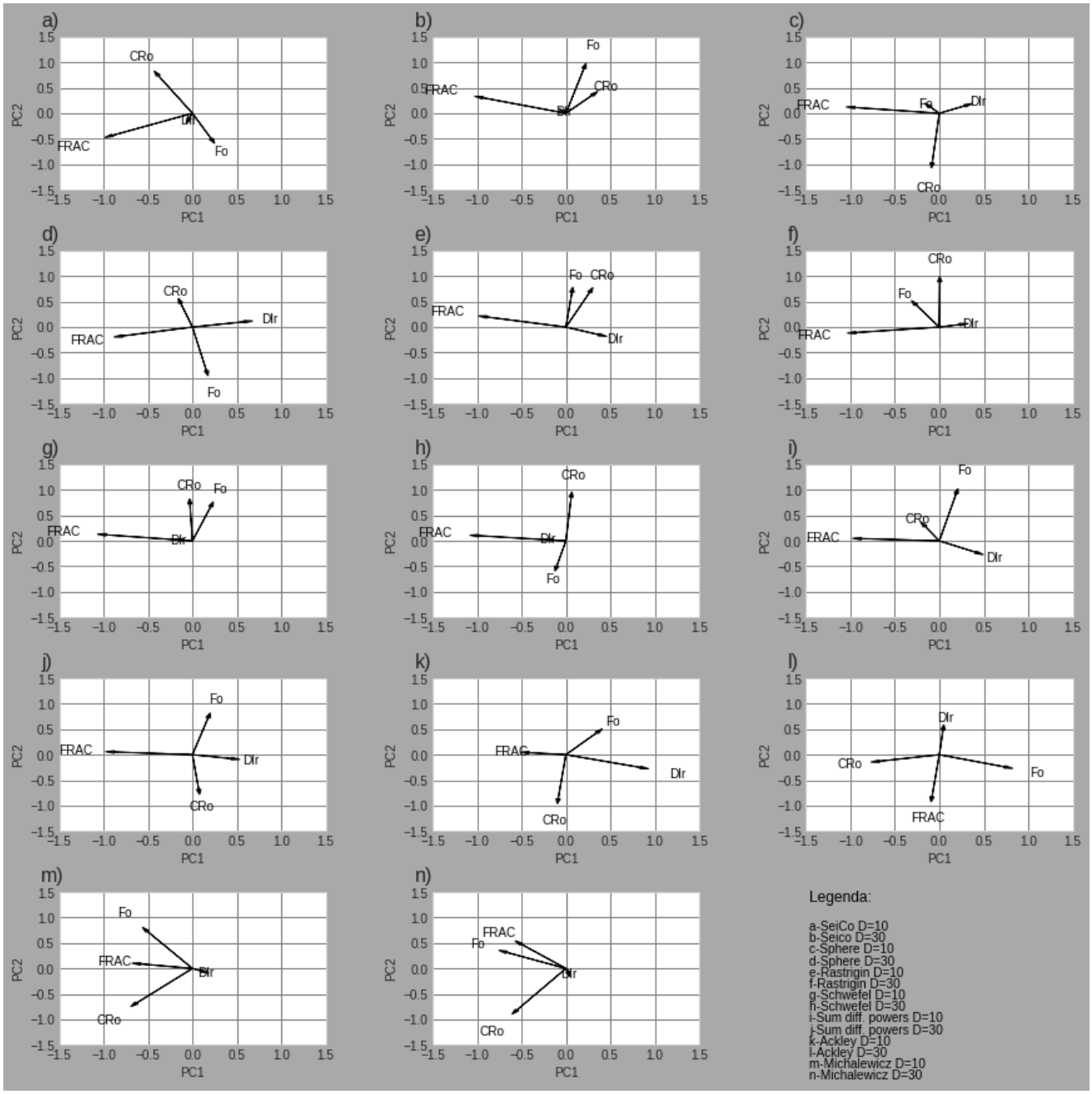

Figura 35: PCA para o conjunto de dados gerados para treinar as RNAs acopladas ao otimizador FRANNK Nota: D é a dimensão.

AVALIAÇÃO DAS INFORMAÇÕES EXTRAÍDAS DAS RNAS

Os efeitos das variáveis de entrada (DI_R, FI_R, F_0 e CR_0) da ANN-F e ANN-CR são apresentados na Tabela 9. Conforme observado na tabela, os valores de F e CR relacionados à iteração anterior (F_0 e CR_0) têm impacto nos valores seguintes de F e CR e todos os efeitos têm uma contribuição significativa no processo de ajuste. Observa-se também que a importância de F_0 em CR foi maior do

que a importância de CR_0 em F. Além disso, a diversidade teve mais impacto no ajuste F do que no ajuste CR.

Tabela 9: Efeitos das variáveis obtidas por partição de peso.

Entradas	**RNA-F**	**RNA-CR**
DI_R	31.8%	23.1%
FI_R	22.5%	27.6%
F_0	28.9%	25.7%
CR_0	16.8%	23.6%

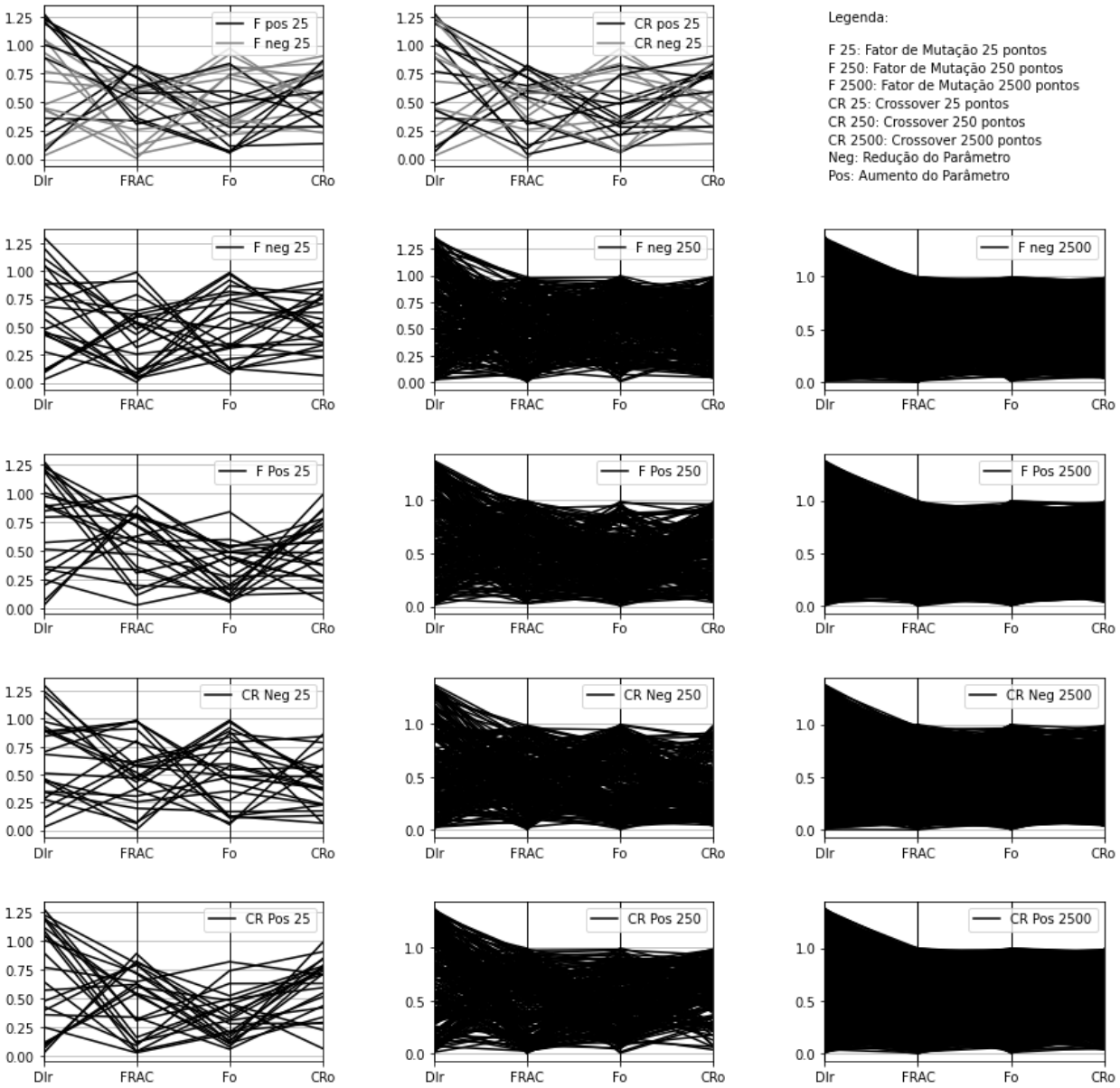

Figura 36: Gráfico de coordenadas paralelas representando as avaliações da RNA no ajuste de F e CR para 25, 250 e 2500 entradas aleatórias.

COMPARAÇÃO ENTRE O OTIMIZADOR FRANNK E OUTROS MÉTODOS

A eficiência do otimizador FRANNK foi testada através da resolução de 18 problemas testes que estão descritos na Tabela 1 e Tabela 2. A comparação entre o otimizador FRANNK e outros otimizadores implementados com base na Evolução Diferencial (jDE, JADE e DE) está apresentada nas Tabelas 9 a 12, para problemas com 10 e 30 dimensões.

As funções *benchmark* presentes na Tabela 10 e Tabela 11, também foram empregadas para ajustar o modelo das RNAs presente no otimizador FRANNK. Conforme observado nessas tabelas, os resultados do otimizador FRANNK superaram os resultados do DE tradicional em todas as funções benchmark (*f1* a *f7*) para D=10 e em seis das sete funções benchmark para D=30. Quando o otimizador FRANNK foi comparado com outros dois eficientes otimizadores adaptativos (jDE e JADE), forneceu o mesmo resultado em quatro das sete funções de teste para D=10 e em três das sete funções de benchmark para D=30. Esses resultados sugerem que o direcionamento fornecido pela RNA pode ser descrito como uma metodologia promissora para o ajuste de parâmetros, desde que o método continue a fornecer bons resultados relativo às metodologias tradicionais no contexto de utilização de funções diferentes das utilizadas no desenvolvimento do otimizador FRANNK.

A Tabela 12 e a Tabela 13 mostram os resultados comparativos entre o otimizador FRANNK e outros métodos, os quais foram obtidos usando funções *benchmark* que não foram previamente usadas durante o treinamento das ANN-F e ANN-CR vinculadas ao otimizador FRANNK. Esses resultados provaram a capacidade do método de solução proposto para melhorar o desempenho do DE tradicional através do ajuste dinâmico de seus parâmetros seguindo por meio de RNAs; e como as funções teste usadas nesta etapa não foram usadas no desenvolvimento de FRANNK, podemos concluir que a ANN-F e a ANN-CR associadas ao FRANNK são capazes de generalizar. Além disso, em seis das dez funções de teste (para ambas as dimensões), os otimizadores adaptativos (FRANNK, jDE e JADE) alcançaram simultaneamente o mesmo ótimo.

Tabela 10: Resultados comparativos para funções *benchmark* usadas no desenvolvimento FRANNK (dimensão 10)

Função		**Estatística**	**FRANNK**	**jDE**	**JADE**	**DE**
SeiCo	f_1	Avg	1.474E-02	**0.000E+00**	**0.000E+00**	3.837E-01
		Std	7.758E-05	0.000E+00	0.000E+00	1.212E+00
Sphere	f_2	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	4.211E+01
		Std	0.000E+00	0.000E+00	0.000E+00	1.840E+02
Rastrigin	f_3	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	2.440E+00
		Std	0.000E+00	0.000E+00	0.000E+00	2.421E+00
Schwefel 2.26	f_4	Avg	9.095E-14	**0.000E+00**	**0.000E+00**	5.315E+02
		Std	2.728E-13	0.000E+00	0.000E+00	2.207E+02
Sum of diff. powers	f_5	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	5.071E-05
		Std	0.000E+00	0.000E+00	0.000E+00	1.572E-04
Ackley	f_6	Avg	3.997E-15	3.997E-15	**3.713E-15**	1.595E+00
		Std	0.000E+00	0.000E+00	9.638E-16	1.679E+00
Michalewicz	f_7	Avg	**-9.660E+00**	**-9.660E+00**	**-9.660E+00**	-9.284E+00
		Std	3.553E-16	7.613E-16	7.536E-16	2.714E-01
Performance		Avg	4	6	**7**	0

Nota: Avg= Média; Sdt=Desvio padrão

Tabela 11: Resultados comparativos para funções *benchmark* usadas no desenvolvimento FRANNK (dimensão 30)

Função		**Estatística**	**FRANNK**	**jDE**	**JADE**	**DE**
SeiCo	f_1	Avg	2.000E-02	**0.000E+00**	**0.000E+00**	**0.000E+00**
		Std	1.953E-04	0.000E+00	0.000E+00	0.000E+00
Sphere	f_2	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	0.000E+00
		Std	0.000E+00	0.000E+00	0.000E+00	0.000E+00
Rastrigin	f_3	Avg	1.933E-14	**0.000E+00**	**0.000E+00**	1.124E+02
		Std	2.693E-14	0.000E+00	0.000E+00	7.073E+00
Schwefel 2.26	f_4	Avg	1.637E-12	**1.164E-12**	2.369E+00	2.785E+03
		Std	5.457E-13	8.820E-13	1.658E+01	1.082E+03
Sum of diff. powers	f_5	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
		Std	0.000E+00	0.000E+00	0.000E+00	0.000E+00
Ackley	f_6	Avg	1.210E-14	7.905E-15	**3.997E-15**	9.255E-15
		Std	8.103E-15	1.794E-15	0.000E+00	2.040E-15
Michalewicz	f_7	Avg	-2.863E+01	**-2.963E+01**	-2.863E+01	-1.819E+01
		Std	8.920E-04	1.808E-03	3.625E-03	7.332E-01

Performance		2	**5**	**5**	2

Tabela 12: Resultados comparativos para funções não utilizadas no desenvolvimento do otimizador FRANNK (dimensão 10)

Função		**Estatística**	**FRANNK**	**jDE**	**JADE**	**DE**
Schwefel 2.22	f_8	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	1.556E-01
		Std	0.000E+00	0.000E+00	0.000E+00	9.186E-01
Schwefel 2.21	f_9	Avg	1.626E-03	**0.000E+00**	**0.000E+00**	9.607E+00
		Std	8.722E-04	0.000E+00	0.000E+00	7.499E+00
Alpine N1	f_{10}	Avg	5.432E-15	5.730E-07	**2.575E-17**	6.516E-03
		Std	2.699E-14	3.393E-06	2.498E-17	2.307E-02
Styblinski-Tang	f_{11}	Avg	**-7.833E+02**	**-7.833E+02**	**-7.833E+02**	-7.612E+02
		Std	9.647E-14	1.125E-13	2.537E-13	2.202E+01
Griewank	f_{12}	Avg	2.442E-17	**0.000E+00**	**0.000E+00**	1.330E-01
		Std	1.319E-16	0.000E+00	0.000E+00	2.498E-01
Rosenbrock	f_{13}	Avg	8.756E-02	1.410E-19	**0.000E+00**	1.667E+02
		Std	7.099E-02	9.499E-19	0.000E+00	5.092E+02
Step	f_{14}	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	2.938E+01
		Std	0.000E+00	0.000E+00	0.000E+00	7.233E+01
Sum Squares	f_{15}	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	1.285E+00
		Std	0.000E+00	0.000E+00	0.000E+00	2.825E+00
Bent Cigar	f_{16}	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	2.415E+07
		Std	0.000E+00	0.000E+00	0.000E+00	9.239E+07
Shubert3	f_{17}	Avg	**-1.484E+02**	**-1.484E+02**	**-1.484E+02**	-1.477E+02
		Std	2.605E-14	8.120E-15	1.928E-14	2.091E+00
Performance		Avg	6	8	10	0

Tabela 13: Resultados comparativos para funções não utilizadas no desenvolvimento do otimizador FRANNK (dimensão 30)

Função		**Estatística**	**FRANNK**	**jDE**	**JADE**	**DE**
Schwefel 2.22	f_8	Avg	9.979E-19	**0.000E+00**	**0.000E+00**	2.396E-17
		Std	2.326E-18	0.000E+00	0.000E+00	1.353E-17
Schwefel 2.21	f_9	Avg	4.838E+00	**0.000E+00**	1.485E-09	3.674E-01
		Std	1.519E+00	0.000E+00	2.808E-09	1.237E+00
Alpine N1	f_{10}	Avg	**1.785E-06**	2.119E+00	5.988E-06	1.132E-02
		Std	6.735E-06	4.634E-01	2.754E-06	4.532E-03
Styblinski-Tang	f_{11}	Avg	**-1.175E+03**	**-1.175E+03**	-1.173E+03	-1.174E+03
		Std	2.766E-13	1.558E-13	5.662E+00	4.241E+00
Griewank	f_{12}	Avg	**0.000E+00**	**0.000E+00**	1.479E-04	**0.000E+00**
		Std	0.000E+00	0.000E+00	1.035E-03	0.000E+00
Rosenbrock	f_{13}	Avg	8.286E-01	**1.519E+01**	3.189E-01	2.470E+01
		Std	8.694E-01	1.126E+00	1.082E+00	8.998E-01
Step	f_{14}	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
		Std	0.000E+00	0.000E+00	0.000E+00	0.000E+00
Sum Squares	f_{15}	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
		Std	0.000E+00	0.000E+00	0.000E+00	0.000E+00
Bent Cigar	f_{16}	Avg	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
		Std	0.000E+00	0.000E+00	0.000E+00	0.000E+00
Shubert3	f_{17}	Avg	**-4.451E+02**	**-4.451E+02**	**-4.451E+02**	-2.220E+02
		Std	1.999E-02	5.151E-04	1.309E-04	1.266E+01
Performance		Avg	7	9	5	4

Na Tabela 14, os resultados de otimização de 10 funções *benchmark* relatados por Zhang & Sanderson (2009) são comparados com o desempenho do otimizador FRANNK. Como pode ser observado, o otimizador FRANNK superou o DE tradicional e o PSO em 9 de 10 casos e atingiu o mínimo global em cinco das dez funções propostas. JADE e jDE encontraram o mínimo global, respectivamente, 8 e 3 dos 10 problemas testados.

Tabela 14: Resultados comparativos entre o otimizador FRANNK e outros métodos de solução (valores encontrados na literatura (Zhang & Sanderson, 2009))

Função	Gen	Estatística	FRANNK	JADE	jDE	SaDE	DE rand/1/bin	PSO
Sphere	1500	Avg	**0.0E+00**	**0.0E+00**	**0.0E+00**	4.5E-20	9.8E-14	**0.0E+00**
		Std	0.0E+00	0.0E+00	0.0E+00	6.9E-20	8.4E-14	0.0E+00
Schwefel 2.22	2000	Avg	**0.0E+00**	**0.0E+00**	**0.0E+00**	1.9E-14	1.6E-09	0.0E+00
		Std	0.0E+00	0.0E+00	0.0E+00	1.0E-14	1.1E-09	6.3E-20
Schwefel 2.21	5000	Avg	1.5E-01	**0.0E+00**	1.4E-15	7.4E-11	4.2E-01	4.4E-14
		Std	2.4E-02	0.0E+00	1.0E-15	1.8E-10	1.1E+00	9.3E-14
Rosenbrock	3000	Avg	1.4E-01	**8.0E-02**	1.3E+01	2.1E+01	2.1E+00	2.5E+01
		Std	3.7E-01	5.6E-01	1.4E+01	7.8E+00	1.5E+00	3.2E+01
Step	1500	Avg	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	8.0E-02
		Std	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E-01
Quartic	3000	Avg	8.9E-02	**6.4E-04**	3.3E-03	4.8E-03	4.7E-03	2.5E-03
		Std	3.0E-02	2.5E-04	8.5E-04	1.2E-03	1.2E-03	1.4E-03
Schwefel 2.26	1000	Avg	**3.1E-12**	3.3E-05	7.9E-11	4.7E+00	5.9E+03	2.4E+03
		Std	9.1E-13	2.3E-05	1.3E-10	3.3E+01	1.1E+03	6.7E+02
Rastrigin	1000	Avg	**2.8E-12**	1.0E-04	1.5E-04	1.2E-03	1.8E+02	5.2E+01
		Std	4.8E-12	6.0E-05	2.0E-04	6.5E-04	1.3E+01	1.6E+01
Ackley	500	Avg	6.3E-04	**8.2E-10**	3.5E-04	2.7E-03	1.1E+00	4.6E-01
		Std	8.7E-05	6.9E-10	1.0E-04	5.1E-04	3.9E-02	6.6E-01
Griewank	500	Avg	5.3E-04	**9.9E-08**	1.9E-05	7.8E-04	2.0E-01	1.3E-02
		Std	3.7E-04	6.0E-07	5.8E-05	1.2E-03	1.1E-01	1.7E-02
Performance		Avg						

Nota: * Gen = Gerações ou iterações.

Com base nos resultados apresentados nas tabelas acima, pode-se concluir que o otimizador FRANNK possui uma forte capacidade de busca global na resolução de problemas. Esses resultados são muito promissores, especialmente porque ainda não foram realizados muitos testes em termos de como ajustar os parâmetros após a avaliação da RNA (classificação binária de aumento ou diminuição

do valor do parâmetro), o que nos testes apresentados foi feito de maneira empírica através das Equações (57) e (58). Um estudo nesse sentido poderia melhorar a eficiência do otimizador FRANNK.

CONCLUSÕES

Este capítulo explorou a aplicação direta de RNAs para ajustar parâmetros de otimizadores. Ao contrário de outros métodos adaptativos, nos quais os parâmetros ajustados seguem regras empíricas, este método aprendeu a ajustar os parâmetros pela experiência obtida durante o treinamento de uma RNA usando um conjunto de dados reduzido. Esta estratégia, que ainda não tinha sido avaliada por outros autores, funcionou surpreendentemente bem quando comparada com o método Evolução Diferencial original e teve um desempenho similar a outros métodos adaptativos considerados eficientes (JADE e jDE). Embora esses resultados sejam encorajadores, muitos desafios ainda permanecem. Um deles é a realização de uma pesquisa sobre as melhores formas de ajustar os parâmetros após obter a direção (de aumento ou diminuição) fornecida pela RNA. Outro, é o aumento do número de problemas considerados no processo de treinamento, a fim de melhor avaliar os pontos fortes e fracos dessa metodologia de ajuste de parâmetros. Seria interessante ainda testar o uso de RNAs para ajuste de parâmetros de outros métodos heurísticos, tais como o PSO, FA e GA, bem como realizar a criação de métodos colaborativos, com uma parcela da população trabalhando com parâmetros fixos, e outra parcela trabalhado com parâmetros ajustados por RNAs, obtendo desta maneira o melhor resultado de cada um dos métodos.

CAPÍTULO 3 - OTIMIZAÇÃO DE UMA REDE DE TROCADORES DE CALOR USANDO O OTIMIZADOR FRANNK E OUTROS MÉTODOS COLABORATIVOS

INTRODUÇÃO

A integração energética é vital para que vários processos industriais sejam realizados a custos aceitáveis, proporcionando a utilização de recursos energéticos escassos de forma econômica (Rathjens & Fieg, 2020). Neste contexto, a síntese de uma rede de trocadores de calor (RTC) que atenda as demandas energéticas de um processo representa uma das mais importantes etapas da integração, a qual se caracteriza matematicamente como um problema não linear, não convexo e com alta demanda combinatória (Aguitoni et al., 2018). Nas últimas quatro décadas esse tipo de problema foi motivo de diversas pesquisas em razão do tempo polinomial associado à solução, pela crescente complexidade de processos industriais e também pela melhora dos resultados encontrados com o uso de métodos populacionais (Morar & Agachi, 2010; Rathjens & Fieg, 2020).

Este capítulo possui dois objetivos principais. O primeiro deles é testar a eficiência do otimizador FRANNK na solução de um problema de aplicação prática relevante na área da Engenharia Química. Para isso, escolheu-se um problema de integração energética, no qual buscou-se minimizar o gasto anual total (TAC) associado a uma rede de trocadores de calor. O segundo objetivo é apresentar uma nova abordagem para métodos de solução populacionais, a qual foi denominada Método Colaborativo (MC). Em um MC, uma parcela da população trabalha seguindo uma estratégia e a outra parcela trabalha de acordo com outro método. Ao final de cada iteração ou de um número pré-determinado de iterações, o melhor indivíduo de um dos métodos substitui o indivíduo com pior desempenho do outro, e vice-versa, com o objetivo de potencializar a eficiência da otimização.

Assim, um problema otimização de uma RTC foi utilizado para fazer um estudo comparativo entre 4 diferentes estratégias de solução: DE, FRANNK e dois MCs (DE-FRANNK e DE-DE).

DESCRIÇÃO DO PROBLEMA

As características da RTC em estudo estão descritas na Tabela 15. Essa RTC tem importância como referência comparativa, pois trata-se de um problema benchmark já discutido por diferentes autores (Aguitoni et al., 2018; Dolan et al., 1989; Huo et al., 2012; Khorasany & Fesanghary, 2009; Nielsen et al., 1996).

Tabela 15: Propriedades das correntes da rede de trocadores de calor em estudo

Corrente	**Temperatura entrada (K)**	**Temperatura saída (K)**	**mCp (kW/K)**
Q1	533	433	3.0
Q2	523	403	1.5
F1	393	508	2.0
F2	453	513	4.0

Nota: Coeficiente global de transferência de calor (U) = 0.2 kW/m^2

No estudo da RTC por otimização, considerou-se como função objetivo a Equação (60) composta pela soma gastos anuais de energia (kW/ano) com utilidade fria (Q_F) e utilidade quente(Q_Q), somada à depreciação associada ao desgaste dos trocadores de calor definida em termos da soma de todas as áreas (A) dos trocadores de calor medidas em m^2 (Aguitoni et al., 2018).

$$TAC\left(\frac{\$}{ano}\right) = 300\,.A^{0.5} + 110.\,Q_Q + 12.2\,.\,Q_F \tag{60}$$

No cálculo das utilidades requeridas foram utilizadas as informações presentes na Tabela 16, sendo que o modelo matemático do sistema está descrito pelas Equações (61) - (67). Nestas equações, T representa a temperatura, os valores θ_1 e θ_2 são as diferenças de temperatura para cada corrente na troca térmica, U é o coeficiente global de troca térmica, A é área associada a cada trocador de calor envolvido na troca térmica, $\dot{Q}$ é a carga térmica, o índice i representa o estágio da troca, os índices j e k representam as correntes, respectivamente, quente e fria e os índices $_{HU}$ e $_{CU}$ representam respectivamente, as utilidades quente e fria. Z_{ijk} representa uma variável binária que assume valores zero ou um, sendo que o valor zero refere-se às situações em que as trocas de energia entre as correntes j e k não ocorreram no o estágio i.

Tabela 16: Informações adicionais associadas ao estudo gasto das utilidades e trocas térmicas segundo a literatura (Aguitoni et al., 2018)

Identificação	**Temperaturas**
Utilidade quente	553 e 552 K
Utilidade fria	303 e 353 K
Coeficiente global de troca térmica	0.2 kW/m^2.K

$$DMLT = \frac{\theta_2 - \theta_1}{ln\left(\frac{\theta_2}{\theta_1}\right)} \quad (61)$$

$$A_{ijk} = \frac{Z_{ijk}\dot{Q}_{ijk}}{DMLT.U} \quad (62)$$

$$\dot{Q}_{ijk} = U_{jk}A_{ijk}DMLT_{ijk} \quad (63)$$

$$T'_{h,ijk} - T''_{h,ijk} = \frac{Z_{ijk}\dot{Q}_{ijk}}{\dot{m}C_{p,h,ijk}} \quad (64)$$

$$T'_{c,ijk} - T''_{c,ijk} = \frac{\dot{Q}_{ijk}}{\dot{m}C_{p,c,ijk}} \quad (65)$$

$$\dot{Q}_{HU} = \dot{Q}_j - \sum_j \sum_i \dot{Q}_{ijk} \quad (66)$$

$$\dot{Q}_{CU} = \dot{Q}_k - \sum_k \sum_i \dot{Q}_{ijk} \quad (67)$$

METODOLOGIA

Incialmente a RTC foi caracterizada utilizando o *software* Hint 2.2 (maiores detalhes são apresentados no Anexo A) que também foi utilizado para avaliar a solução otimizada. Quanto à otimização, foram avaliadas doze estratégias descritas na Tabela 17 para comparação entre as metodologias DE, FRANNK e o Método Colaborativo DE-FRANNK nas proporções 1:1, 2:1, 1:2 e 1:3 nas quais os métodos DE e FRANNK coexistem sendo que a fração da população do FRANNK em relação a população total representa, respectivamente, 1/2, 1/3, 2/3 e 1/4 da população total.

Tabela 17: Planejamento dos ensaios comparativos entre os métodos DE, FRANNK e DE – FRANNK.

Ensaio	Método*	Iterações	População	Proporção DE-FRANNK
1	DE	100	60	-
2	FRANNK	100	60	-
3	MC	100	60	1:1
4	MC	100	60	2:1
5	MC	100	60	1:2
6	MC	100	60	3:1
7	DE	300	150	-
8	FRANNK	300	150	-
9	MC	300	150	1:1
10	MC	300	150	2:1
11	MC	300	150	1:2
12	MC	300	150	3:1
13	DE	500	300	-
14	FRANNK	500	300	-
15	MC	500	300	1:1
16	MC	500	300	2:1
17	MC	500	300	1:2
18	MC	500	300	3:1

Nota: * MC se refere ao método DE-FRANNK.

RESULTADOS E DISCUSSÕES

CARACTERIZAÇÃO DO PROBLEMA POR ANÁLISE *PINCH*

Na Figura 37 são apresentados detalhes da referida RTC gerada pelo *software* Hint 2.2 (Martín & Mato, 2008) por meio de metodologia *Pinch* para diferentes valores de diferença mínima de temperatura entre correntes durante as trocas e considerando os valores de custos citados na metodologia. Pode-se observar que as temperaturas *Pinch* variam entre 453.5 e 474.56 K para variações de temperatura mínima escolhidas entre 1 e 43.1 K e que o custo mínimo previsto por esta metodologia é de 10780.4 $/ano para uma diferença mínima de temperatura de 4.91 K. Segundo o mesmo *software*, são necessários no mínimo, 7 trocas térmicas: cinco trocas entre as correntes e duas trocas adicionais associadas ao uso de utilidades fria e quente.

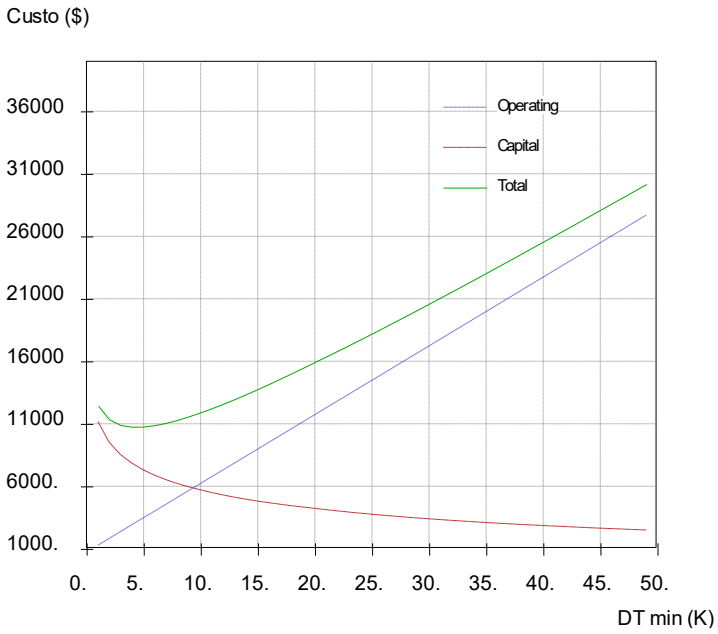

Figura 37: RTC segundo análise *Pinch* obtida no software Hint 2.2

RESULTADOS DA OTIMIZAÇÃO

Na Tabela 18 são apresentados os valores obtidos nas diferentes simulações realizadas. Pode-se observar que o MC composto pelo método DE associado ao método FRANNK com 1/3 da população associada ao FRANNK se destacou como a melhor opção em termos de encontrar o melhor resultado em 50 resoluções. Pode-se observar que ótimo obtido nesta condição de 12852 $/ano foi satisfatório comparado com valores encontrados na literatura que cita, em diferentes trabalhos, valores de ótimo global entre 11540,43 e 12870,00 $/ano (Aguitoni et al., 2018; Dolan et al., 1989; Huo et al., 2012; Khorasany & Fesanghary, 2009; Nielsen et al., 1996). Cabe salientar, no entanto, que o mínimo valor obtido neste trabalho poderia ser melhorado adotando-se estratégias que não foram aqui usadas como, por exemplo, separar as variáveis inteiras das contínuas e aplicar diferentes métodos a cada tipo de variável, como foi feito em alguns trabalhos da literatura (Aguitoni et al., 2018; Zhaoyi et al., 2013).

Ainda referente à Tabela 18, a comparação entre os métodos FRANNK e DE para 100, 300 e 500 iterações e populações de, respectivamente, 60, 150 e 300 mostraram que, tanto em termos de valores médios como em termos do melhor resultado em 50 repetições, o método DE foi superior ao otimizador FRANNK. Esse resultado pode estar associado ao fato de a RTC, por ser um problema misto, com restrições, que contém diversos estágios e certa complexidade combinatória, representar um problema de natureza distinta das funções *benchmark* (contínuas e sem restrições) usadas para treinar RNA acoplada ao otimizador FRANNK.

A Figura 38 mostra o esquema da RTC otimizada pelo método DE-FRANNK 2:1 obtido no *software* Hint 2.2. Pode-se observar que: a) a solução fornecida tem o mesmo número de trocadores de calor previsto pelo método *Pinch* (7 trocadores); b) todas correntes efetuaram duas trocas térmicas com outras correntes e uma terceira troca com as utilidades com exceção da corrente *Q1* que efetuou uma troca com a corrente *F2* seguida de outra com a corrente *F1* e já atingiu sua temperatura alvo; c) os quatro primeiros trocadores de calor descreveram as trocas térmicas entre as correntes e os três últimos entre correntes e utilidades; d) a numeração dos trocadores de calor 1 a 4 (trocas entre as correntes) associa, respectivamente, *Q1* e *F2, Q2* e *F2, Q1* e *F1, Q2* e *F1*; e) os trocadores de calor 5 a 7 associam, respectivamente, as correntes *Q2, F1* e *F2* com as utilidades.

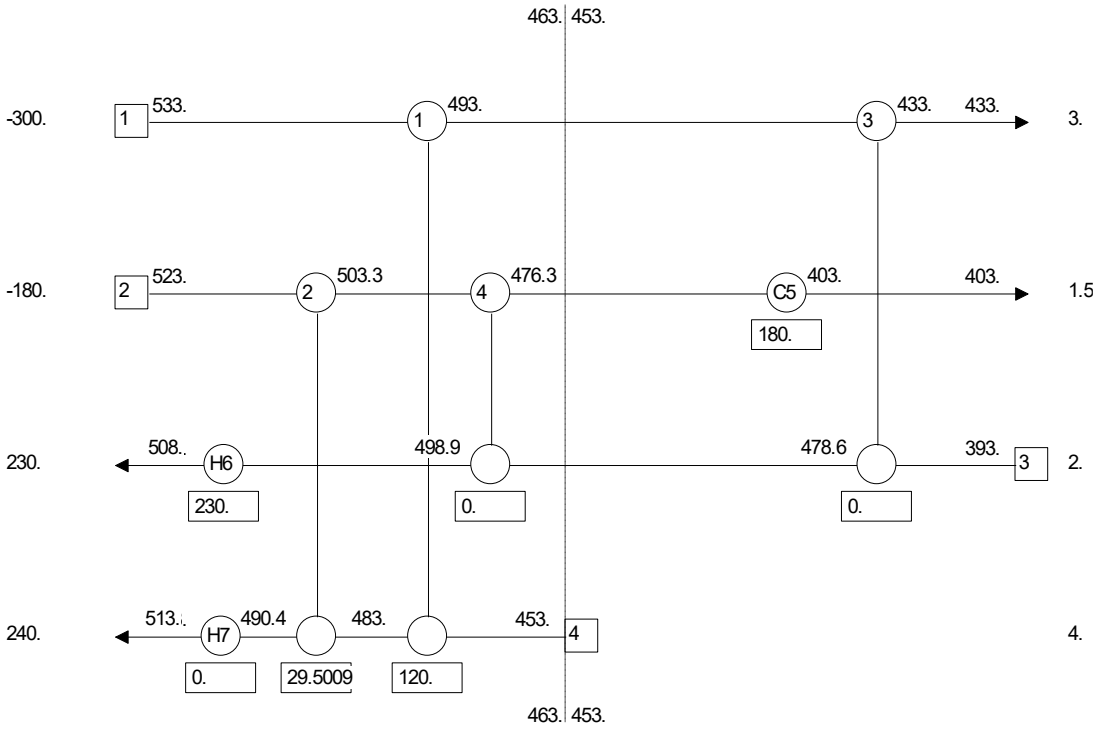

Figura 38: Esquema da RTC otimizada pelo método DE-FRANNK 2:1 com 500 iterações e 300 partículas obtido no *software* Hint 2.2

Na Tabela 19 são mostrados os resultados do Método Colaborativo DE-DE com 1/3 das partículas em um dos métodos e 2/3 no outro para as mesmas condições em termos de população e iterações do método colaborativo DE-FRANNK 2:1. O método DE-FRANNK 2:1 possui 1/3 da população no método FRANNK e 2/3 no método DE (ensaio que mostrou os melhores resultados em relação às outras metodologias apresentadas na Tabela 18). Os resultados apresentados na Tabela 19 mostraram que a estratégia colaborativa também funciona quando se utiliza dois métodos idênticos (no caso, dois DE). Já a comparação entre os resultados da Tabela 19 com os ensaios 4, 10 e 16 da Tabela 18 mostrou o método colaborativo DE-DE obteve um resultado ligeiramente melhor que o método

colaborativo DE-FRANNK 2:1 considerando 50 repetições. Apesar disso, em todas as três condições testadas, a diferença relativa entre os melhores resultados obtidos pelos métodos foi inferior a 0,30%.

Tabela 18: Resultados comparativos entre o método colaborativo DE-FRANNK e os métodos DE, FRANNK

Ens	Mét*	Gen	Pop	Prop	TAC ($/ano) [a]	TAC ($/ano) [b]
1	DE	100	60	-	17615.96±1528.78	14458.73
2	FRANNK	100	60	-	22852.32±2449.16	16568.77
3	MC	100	60	1:1	19951.37± 2895.28	14809.08
4	**MC**	**100**	**60**	**2:1**	**18805.17±3755.74**	**12888.16**
5	MC	100	60	1:2	20037.60±2938.48	14276.25
6	MC	100	60	3:1	19322.83±3099.51	13179.67
7	DE	300	150	-	14905.89±658.87	13409.22
8	FRANNK	300	150	-	18418.81±2280.30	14170.27
9	MC	300	150	1:1	16249.24±2228.57	13191.52
10	**MC**	**300**	**150**	**2:1**	**16218.80±1832.15**	**12852.35**
11	MC	300	150	1:2	17292.65±1989.69	13633.05
12	MC	300	150	3:1	16360.41±2302.27	12868.26
13	DE	500	300	-	13535.11±229.71	13091.17
14	FRANNK	500	300	-	16154.72±1235.72	13471.09
15	MC	500	300	1:1	15019.99±1392.709	12956.53
16	**MC**	**500**	**300**	**2:1**	**15412.52±1571.83**	**12852.08**
17	MC	500	300	1:2	15427.26±1502.39	13044.52
18	MC	500	300	3:1	14904.585±1252.62	12889.49

Nota: a- média e desvio padrão para 50 resoluções; b- melhor resultado obtido em 50 resoluções

Tabela 19: Otimização por metodologia colaborativa DE-DE com 1/3 da população em um dos métodos

Ens	Gen	Pop	TAC ($/ano)[a]	TAC ($/ano)[b]	Erro (%)[c]
1	100	60	15731.22± 2542.21	12855.56	0.25%
2	300	150	13844.59 ± 1087.65	12851.66	0.0054%
3	500	300	13515.78± 599.04	12851.66	0.0034%

Nota: a- média e desvio padrão para 50 resoluções; b- melhor resultado obtido em 50 resoluções; c -Diferença percentual relativa entre os melhores resultados obtidos considerando os dados desta tabela e os valores de referência (Tabela 18).

CONCLUSÕES

Como principais conclusões referentes ao uso do método FRANNK e dos MCs na otimização de uma RTC têm-se:

a) A natureza combinatória do problema faz com que o aprendizado da RNA não possa ser transferido da forma mais satisfatória das funções de *benchmark* para a RTC. Para que se tenha a condição de melhor avaliar o aprendizado, o ideal seria treinar a RNA acoplada ao otimizador FRANNK usando problemas de RTC ou ao menos outros problemas de natureza mista.

b) Embora a transferência de aprendizado possa dificultar o uso da metodologia FRANNK em contextos distintos do treinamento, a mesma em sua forma colaborativa mostrou-se capaz de gerar resultados satisfatórios, mesmo em um contexto reduzido de interações nas quais a metodologia tradicional DE sem abordagem colaborativa gera resultados consideravelmente piores.

c) O uso do otimizador FRANNK na forma colaborativa com o DE mostrou-se uma metodologia interessante. Observou-se que a fração da população em cada método impacta na eficiência do método e o uso de 1/3 da população no otimizador FRANNK é uma proporção satisfatória para o estudo da RTC proposta.

d) De modo geral, a metodologia colaborativa mostrou-se bastante promissora na solução de problemas, sendo capaz de aumentar significativamente a eficiência da otimização mesmo quando se trata de 2 otimizadores iguais trabalhando simultaneamente, como foi o caso do método DE - DE.

CAPÍTULO 4 – DESENVOLVIMENTO E AJUSTE DE UMA NOVA VERSÃO DO MÉTODO COLABORATIVO

INTRODUÇÃO

A eficiência dos algoritmos de *Swarm Intelligence* depende, em grande parte, dos valores dos parâmetros definidos no início da resolução do problema, os quais podem ser ajustados ou não durante a evolução da otimização. No caso específico do *Differential Evolution* (DE), os valores dos parâmetros *F* e *CR* são particularmente importantes, afetando consideravelmente tanto a velocidade de convergência da otimização, como a capacidade do método de escapar de soluções sub ótimas.

O procedimento de ajuste de parâmetros geralmente requer um grande número de execuções do algoritmo para analisar sua eficiência em uma ou mais instâncias de problemas com diferentes definições de parâmetro, devido à grande quantidade de diferentes valores que cada parâmetro pode assumir. Este procedimento pode demandar um alto custo do ponto de vista computacional. De acordo com Adenso-Diaz e Laguna (Adenso-Díaz & Laguna, 2006), ao desenvolver um novo método de solução meta-heurístico, apenas 10% do tempo é gasto no desenvolvimento do algoritmo, sendo os 90% do tempo restantes gastos no ajuste fino de seus parâmetros. E, apesar do grande esforço necessário nesse ajuste, não há garantias de que um algoritmo bem ajustado que funciona bem para um tipo de problema funcione igualmente bem no case de outro tipo de problema.

Um dos objetivos do método FRANNK, apresentado no Capítulo 2, o qual faz o ajuste dinâmico dos parâmetros do DE durante a otimização, é, justamente, a redução do impacto da escolha inicial dos parâmetros no resultado final da otimização. Já o método colaborativo DE-FRANNK, apresentado no Capítulo 3, foi desenvolvido de maneira a aumentar a eficiência do método FRANNK, ao criar 2 subpopulações interdependentes, de modo que uma parcela da população trabalhe com parâmetros fixos, e a outra parcela trabalhe com parâmetros ajustados dinamicamente ao longo da otimização, a fim de obter o melhor resultado de cada um dos métodos. Surpreendentemente, durante o desenvolvimento desse método colaborativo, constatou-se que dividir a população do método e transferir as melhores soluções de uma população para a outra gera uma sinergia, mesmo quando as populações trabalham com métodos de solução idênticos e com parâmetros iniciais iguais.

A partir daí, surgiu então a ideia de dividir a população do método, não apenas em 2 partes, mas em outras diversas partes, nas quais os indivíduos trabalhassem usando o mesmo algoritmo, porém com parâmetros diferentes. Essa seria uma ideia de fácil implementação, porém com alto potencial de impacto no resultado da otimização e que reduziria ainda mais o impacto da escolha inicial dos parâmetros do algoritmo, tornando-o adequado para uma gama maior de problemas, e potencialmente

eliminando a necessidade de ajuste de parâmetros antes da otimização. Assim surgiu a segunda versão do método colaborativo DE-FRANNK, denominada C-FRANNK, a qual será descrita neste Capítulo.

METODOLOGIA

IMPLEMENTAÇÃO DO OTIMIZADOR C- FRANNK

O método colaborativo C-FRANNK tem a mesma estrutura presente na versão do método DE- FRANNK de melhor eficiência descrito no Capítulo 2, com um terço da população trabalhando com a metodologia FRANNK, cujos parâmetros são ajustados dinamicamente ao longo da otimização por uma rede neural, e a outra parte da população composta por 2/3 dos indivíduos trabalhando com a metodologia tradicional do método DE (DE/rand/1/bin), o qual possui parâmetros fixos. A novidade de C- FRANNK está no fato de que a população que trabalha empregando o método DE é dividida em outras 3 subpopulações de mesmo número de indivíduos. Para cada uma dessas subpopulações são definidos valores diferentes para o par de parâmetros *CR* - *F*, que assumem valores iguais a, respectivamente, 0.5 – 0.5, 0.25 – 0.75 e 0.25 – 0.25. Ao final de cada iteração, o melhor indivíduo de cada uma das 4 subpopulações de C-FRANNK (1 trabalhando com o método FRANNK e 3 com o método DE) substitui o indivíduo com pior desempenho dos outros, e vice-versa, com o objetivo de potencializar a eficiência da otimização.

AVALIAÇÃO DO OTIMIZADOR C-FRANNK

A avaliação do método C- FRANNK foi realizada em de duas formas distintas. Uma delas refere-se à comparação dos resultados de C-FRANNK com resultados obtidos a partir de códigos computacionais implementados pela autora referente aos métodos FRANNK, jDE, JADE e DE na resolução das 15 funções de benchmark as quais não foram utilizadas durante o treinamento das RNAs acopladas a FRANNK. Como forma alternativa de avaliar C-FRANNK foi feito um estudo comparativo entre os resultados de C-FRANNK e resultados retirados da literatura, considerando métodos que possuem um ajuste dinâmico dos parâmetros realizados de forma semelhante a C- FRANNK, ou seja, com o uso de técnicas de inteligência artificial.

RESULTADOS

Esta seção faz a comparação do método C-FRANNK tanto em relação a métodos tradicionais baseados em DE, como em relação a métodos da literatura que usam inteligência artificial do tipo lógica *fuzzy* para adaptar parâmetros.

ESTUDO COMPARATIVO ENTRE O OTIMIZADOR C-FRANNK E MÉTODOS TRADICIONAIS

A Tabela 20 e a Tabela 21 mostram os resultados da comparação do método colaborativo C- FRANNK com as metodologias FRANNK, DE, jDE e JADE para 15 funções benchmark com dimensões 10 e 30. Como pode-se observar nestes resultados, de modo geral, o método colaborativo C- FRANNK obteve um desempenho melhor que as metodologias adaptativas tradicionais (jDE e JADE) e, da mesma forma, alcançou melhores resultados do que as metodologias DE e FRANNK, as quais foram empregadas em sua construção. Para dimensão 10, C- FRANNK atingiu melhores resultados em 11 e 13 das 15 funções considerando, respectivamente, a média de 50 execuções e o melhor resultado neste mesmo conjunto de 50 execuções. Para dimensão 30 a metodologia JADE foi a melhor quando se considerou o melhor resultado em 50 execuções, mas C- FRANNK também se destacou fornecendo o melhor resultado quando se considerou a média das 50 execuções.

Tabela 20: Resultados comparativos considerando funções teste diferentes das usadas no treinamento da RNA acoplada a FRANNK (dimensão 10)

Função	Estatística	C- FRANNK	FRANNK	jDE	JADE	DE
Schwefel 2.22	Avg	6.300E-60	2.689E-26	2.738E-64	2.143E-46	1.556E-01
	Std	1.854E-59	1.097E-26	7.804E-64	4.435E-46	9.186E-01
	Min	1.541E-73	4.030E-27	1.595E-66	4.132E-50	1.471E-34
Schwefel 2.21	Avg	1.250E-08	1.626E-03	5.323E-32	5.364E-63	9.607E+00
	Std	1.041E-08	8.722E-04	1.132E-31	2.984E-62	7.499E+00
	Min	9.095E-14	1.279E-05	4.781E-33	2.801E-67	2.120E-02
Alpine N1	Avg	4.441E-18	5.432E-15	5.730E-07	2.298E-17	6.516E-03
	Std	3.109E-17	2.699E-14	3.393E-06	2.077E-17	2.307E-02
	Min	2.507E-97	8.956E-25	1.443E-15	5.595E-22	2.017E-26
Styblinski-Tang	Avg	-3.917E+02	-3.917E+02	-3.917E+02	-3.917E+02	-3.806E+02
	Std	4.890E-14	9.647E-14	1.125E-13	1.235E-13	2.202E+01
	Min	-3.917E+02	-3.917E+02	-3.917E+02	-3.917E+02	-3.917E+02
Griewank	Avg	0.000E+00	2.442E-17	0.000E+00	0.000E+00	1.330E-01
	Std	0.000E+00	1.319E-16	0.000E+00	0.000E+00	2.498E-01
	Min	0.000E+00	0.000E+00	0.000E+00	0.000E+00	3.696E-06

Rosenbrock	Avg	1.322E-01	8.756E-02	1.410E-19	0.000E+00	1.667E+02
	Std	5.711E-01	7.099E-02	9.499E-19	0.000E+00	5.092E+02
	Min	4.554E-07	6.784E-03	1.091E-21	0.000E+00	1.163E+00
Step	Avg	0.000E+00	0.000E+00	0.000E+00	0.000E+00	2.938E+01
	Std	0.000E+00	0.000E+00	0.000E+00	0.000E+00	7.233E+01
	Min	0.000E+00	0.000E+00	0.000E+00	0.000E+00	1.203E-11
Sum Squares	Avg	2.419E-112	4.333E-49	2.757E-101	1.819E-91	1.285E+00
	Std	9.874E-112	4.407E-49	1.067E-101	1.097E-90	2.825E+00
	Min	1.474E-130	3.644E-53	5.414E-102	4.165E-112	3.838E-08
Bent Cigar	Avg	9.771E-105	4.567E-42	2.283E-114	2.729E-83	2.415E+07
	Std	6.393E-104	5.436E-42	6.545E-114	1.892E-82	9.239E+07
	Min	9.301E-125	1.220E-47	1.135E-119	2.173E-105	2.315E-10
Shubert3	Avg	-1.484E+02	-1.484E+02	-1.484E+02	-1.484E+02	-1.477E+02
	Std	2.165E-14	2.605E-14	1.928E-14	1.798E-14	2.091E+00
	Min	-1.484E+02	-1.484E+02	-1.484E+02	-1.484E+02	-1.484E+02
Penalized 1	Avg	4.71E-32	4.71E-32	2.65E-21	4.71E-32	2.61E+02
	Std	1.10E-47	1.09E-47	9.97E-22	1.09E-47	1.82E+03
	Min	4.71E-32	4.71E-32	6.21E-22	4.71E-32	4.88E-16
Penalized 2	Avg	1.35E-32	1.35E-32	2.41E-21	1.35E-32	1.84E+04
	Std	2.74E-48	2.74E-48	9.60E-22	2.74E-48	6.60E+04
	Min	1.35E-32	1.35E-32	1.15E-21	1.35E-32	5.54E-06
Schwefel 2.23	Avg	0.00E+00	5.09E-230	0.00E+00	0.00E+00	3.18E+03
	Std	0.00E+00	2.98E-248	0.00E+00	0.00E+00	1.29E+04
	Min	0.00E+00	2.51E-228	0.00E+00	0.00E+00	4.12E-23
Dixon Price	Avg	6.00E-06	3.01E-03	6.67E-01	6.67E-01	6.67E-01
	Std	2.09E-05	8.80E-03	6.73E-17	8.96E-12	1.04E-06
	Min	8.94E-24	7.61E-09	6.67E-01	6.67E-01	6.67E-01
Salomon	Avg	9.79E-02	9.99E-02	9.99E-02	9.99E-02	2.69E-01
	Std	1.38E-02	1.02E+11	1.24E-11	1.04E-11	2.39E-01
	Min	1.42E-03	9.99E-02	9.99E-02	9.99E-02	9.99E-02
Performance		**C- FRANNK**	**FRANNK**	**jDE**	**JADE**	**DE**
	Avg	**11**	4	8	9	0
	Min	**13**	5	5	10	2

Nota: No campo Performance, Avg é a contagem do melhor método considerando a média em 50 execuções; Min é a contagem relativa ao método que atingiu o melhor resultado em 50 execuções. Todos os ensaios desta tabela foram realizados com uma população de 100 agentes e 2000 gerações.

Tabela 21: Resultados comparativos entre C- FRANNK e metodologias tradicionais (média de 50 resoluções e dimensão 30)

Função	**Estatística**	**C- FRANNK**	**FRANNK**	**jDE**	**JADE**	**DE**
Schwefel 2.22	Avg	1.135E-21	9.979E-19	2.515E-36	6.132E-24	2.396E-17
	Std	2.954E-21	2.326E-18	3.755E-36	4.289E-23	1.353E-17
	Min	9.224E-28	4.434E-22	5.643E-38	2.463E-48	9.227E-18
Schwefel 2.21	Avg	2.336E+00	4.838E+00	4.166E-32	7.078E-10	3.674E-01
	Std	8.151E-01	1.519E+00	5.142E-32	2.113E-09	1.237E+00

	Min	5.783E-03	3.093E-02	1.607E-33	2.476E-12	3.533E-03
Alpine N1	Avg	2.672E-14	1.785E-06	2.119E+00	7.045E-06	1.132E-02
	Std	1.852E-13	6.735E-06	4.634E-01	3.174E-06	4.532E-03
	Min	3.610E-27	2.627E-12	1.386E+00	1.165E-06	1.332E-05
Styblinski-Tang	Avg	-1.175E+03	-1.175E+03	-1.175E+03	-1.174E+03	-1.174E+03
	Std	2.965E-13	2.766E-13	1.558E-13	4.241E+00	4.241E+00
	Min	-1.175E+03	-1.175E+03	-1.175E+03	-1.175E+03	-1.175E+03
Griewank	Avg	0.000E+00	0.000E+00	0.000E+00	1.479E-04	0.000E+00
	Std	0.000E+00	0.000E+00	0.000E+00	1.035E-03	0.000E+00
	Min	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
Rosenbrock	Avg	2.084E+01	8.286E-01	1.519E+01	2.392E-01	2.470E+01
	Std	2.535E+01	8.694E-01	1.126E+00	9.468E-01	8.998E-01
	Min	3.859E-03	5.410E-02	1.284E+01	1.148E-24	1.984E+01
Step	Avg	0.000E+00	3.698E-34	0.000E+00	0.000E+00	3.919E-29
	Std	0.000E+00	2.191E-33	0.000E+00	0.000E+00	4.518E-29
	Min	0.000E+00	0.000E+00	0.000E+00	0.000E+00	3.306E-30
Sum Squares	Avg	1.386E-40	2.614E-34	2.681E-101	4.925E-78	4.441E-30
	Std	2.798E-40	3.289E-34	1.131E-101	3.445E-77	4.636E-30
	Min	1.075E-47	2.768E-37	8.395E-102	8.200E-99	2.396E-31
Bent Cigar	Avg	1.762E-34	1.704E-27	5.476E-62	6.721E-79	1.944E-23
	Std	8.361E-34	2.126E-27	1.629E-61	3.871E-78	3.256E-23
	Min	7.397E-41	1.830E-30	7.004E-65	3.557E-92	9.502E-25
Shubert3	Avg	-4.451E+02	-4.451E+02	-4.451E+02	-4.451E+02	-2.220E+02
	Std	6.580E-14	1.999E-02	5.151E-04	1.526E-04	1.266E+01
	Min	-4.451E+02	-4.451E+02	-4.451E+02	-4.451E+02	-2.486E+02
Penalized 1	Avg	1.57E-32	1.57E-32	7.18E-21	1.57E-32	3.33E-29
	Std	0.00E+00	0.00E+00	1.83E-21	0.00E+00	6.82E-29
	Min	1.571E-32	1.571E-32	4.255E-21	1.571E-32	3.035E-31
Penalized 2	Avg	1.350E-32	1.35E-32	7.36E-21	1.35E-32	1.24E-28
	Std	2.737E-48	2.74E-48	1.89E-21	2.74E-48	2.22E-28
	Min	1.350E-32	1.350E-32	3.596E-21	1.350E-32	1.224E-30
Schwefel 2.23	Avg	5.16E-162	1.49E-143	1.13E-142	2.07E-133	1.28E-38
	Std	3.22E-161	8.75E-143	5.32E-142	1.45E-132	8.98E-38
	Min	1.380E-169	1.018E-166	2.281E-195	1.313E-183	3.541E-76
Dixon Price	Avg	1.70E-01	1.46E-02	6.67E-01	6.67E-01	6.67E-01
	Std	3.06E-01	9.46E-02	6.73E-17	1.58E-10	1.04E-06
	Min	1.506E-09	3.146E-08	6.667E-01	6.667E-01	6.667E-01
Salomon	Avg	3.10E-01	3.38E-01	1.55E-01	2.00E-01	2.00E-01
	Std	4.10E-02	5.20E-02	4.88E-02	3.46E-02	1.84E-03
	Min	1.999E-01	2.059E-01	9.987E-02	9.987E-02	1.999E-01
Performance		**C- FRANNK**	**FRANNK**	**jDE**	**JADE**	**DE**
	Avg	**9**	5	8	6	0
	Min	8	6	8	**10**	2

Nota: No campo Performance, Avg é a contagem do melhor método considerando a média em 50 execuções; Min é a contagem relativa ao método que atingiu o melhor resultado em 50 execuções. Todos os ensaios desta tabela foram realizados com uma população de 100 agentes e 2000 gerações.

ESTUDO COMPARATIVO ENTRE C-FRANNK E MÉTODOS COM AJUSTE DE PARÂMETROS QUE UTILIZAM INTELIGÊNCIA ARTIFICIAL DO TIPO LÓGICA *FUZZY*

O estudo comparativo do método C-FRANNK com os métodos adaptativos que usam inteligência artificial do tipo lógica *fuzzy* (os únicos métodos disponíveis na literatura que apresentam o ajuste dos parâmetros segundo AI) confirmam que C-FRANNK tem uma forte habilidade para encontrar o ótimo global quando comparado aos demais métodos. A comparação de C-FRANNK com duas versões do PSO com lógica *fuzzy* (Tabela 22) em doze condições diferentes da literatura (Olivas et al., 2019) mostram que C- FRANNK e T2FPSO encontraram os melhores resultados em seis dos casos, mas o C- FRANNK se destacou em relação aos dois métodos citados nesta tabela por ter encontrado o mínimo global para todas as 50 execuções em quatro casos, enquanto os demais métodos não encontraram o mínimo global em todas as execuções para nenhum dos problemas testados. A comparação entre C- FRANNK e três métodos baseados em força gravitacional com parâmetros ajustados segundo lógica *fuzzy* (Tabela 23) para dez funções diferentes da literatura (Olivas et al., 2019) forneceu como resultado que C-FRANNK foi melhor que os três outros métodos, encontrando o ótimo global em todas as execuções para 2 funções e o melhor resultado médio 5 vezes.

Tabela 22: Resultados comparativos entre C- FRANNK e metodologias baseadas em PSO com ajuste de parâmetros por lógica *fuzzy*

Função	**D**	**Gen**	**Pop**	**C- FRANNK**	**FGSA**	**T2FPSO**
Rosenbrock	10	1000	20	5.37E+02	6.60E+01	**6.14E+00**
	10	1000	80	**1.32E-03**	1.58E+01	4.20E+00
	30	2000	20	1.93E+05	1.84E+02	**4.14E+01**
	30	2000	80	**1.15E-02**	1.24E+02	2.80E+01
Rastrigin	10	1000	20	1.62E+01	4.96E+00	**2.47E+00**
	10	1000	80	**0.00E+00**	2.33E+00	7.36E-01
	30	2000	20	1.51E+02	4.85E+01	**1.86E+01**
	30	2000	80	**0.00E+00**	2.25E+01	1.25E+01
Griewank	10	1000	20	3.91E+00	9.16E-02	**8.94E-02**
	10	1000	80	**0.00E+00**	6.83E-02	7.71E-02
	30	2000	20	7.85E+01	2.16E-02	**1.42E-02**
	30	2000	80	**0.00E+00**	1.49E-02	1.24E-02
Performance						
Avg				**6**	0	**6**
G-Min				**4**	0	0

Nota: Avg é a contagem do melhor método considerando a média de 50 resoluções; G-Min é a contagem relativa ao número de vezes em que a média de 50 resoluções atingiu o ótimo global.

Tabela 23: Resultados comparativos entre a metodologia C-FRANNK e metodologias Força Gravitacional com ajuste de parâmetros por lógica *fuzzy* (50 agentes e 1000 gerações)

Função	**C-FRANNK**	**FGSA**	**T1FGSA**	**T2FGSA**
Rosenbrock	**3.12E-01**	6.12E+01	9.56E+00	1.00E+00
Schwefel 2.22	1.42E-14	1.16E-10	2.76E−19	**6.34E-24**
Schwefel 2.21	1.18E+00	9.12E−02	6.58E−11	**9.49E-16**
Step	2.93E-23	1.00E-01	**0.00E+00**	**0.00E+00**
Noisy Quartic	5.75E-02	2.60E-02	6.15E-04	**1.00E-05**
Schwefel 2.26	**-1.26E+04**	-2.68E+03	-9.80E+03	-3.60E+03
Rastrigin	**0.00E+00**	1.72E+01	6.87E+00	2.59E-03
Penalized 1	**1.31E-24**	1.10E-01	2.54E-04	2.11E-05
Penalized 2	1.79E-23	5.81E-02	3.45E-43	**4.76E-45**
Shekel's Foxholes	**2.60E-53**	2.80E+00	1.30E-07	9.10E-08
Performance				
Avg	**5**	0	0	**5**
G-Min	**2**	0	1	1

Notas: Avg é a contagem do melhor método considerando a média de 50 resoluções; G-Min é a contagem relativa ao número de vezes em que a média de 50 resoluções atingiu o ótimo global.

CONCLUSÕES

O uso de subpopulações mostrou-se uma estratégia capaz de gerar uma melhora no desempenho da metodologia FRANNK, sendo que o método C-FRANNK apresentou ótimo desempenho tanto quando comparado com os métodos tradicionais que ajustam parâmetros, como quando comparado com os métodos que utilizam AI para este ajuste. Assim com a versão tradicional do método FRANNK, o método C- FRANNK tem forte habilidade de encontrar o ótimo global, apresentando, porém, melhores resultados que o método FRANNK original, para as funções avaliadas.

CAPÍTULO 5 – CONCLUSÕES GERAIS

A otimização com ajuste de parâmetros por aprendizado neural (FRANNK) tem capacidade de generalização satisfatória na resolução de funções *benchmark*, mesmo quando o treinamento foi realizado usando um reduzido conjunto de funções teste. O desempenho da otimização obtido através do otimizador FRANNK foi tão satisfatório quanto à performance de métodos considerados eficientes como o JADE e jDE.

Embora o método FRANNK treinada com funções *benchmark* contínuas e irrestritas não seja capaz de generalizar em todas as situações distintas do seu treinamento, como por exemplo, no caso de problemas de redes de trocador de calor que são de programação mista com natureza combinatória, o mesmo pode ser utilizado combinado a outras metodologias em uma forma colaborativa DE- FRANNK, com desempenho superior ao método DE.

A combinação de métodos na forma colaborativa representa uma estratégia promissora na solução de problemas diversos, uma vez que mesmo quando a população subdividida é otimizada de forma interdependente pelo mesmo método (como foi o caso do método colaborativo DE-DE) apresentou um desempenho significativamente melhor que o método DE tradicional, trabalhando em condições similares em termos de população total e número de iterações. Neste contexto, a proporção entre as populações em cada método influi na eficiência do método colaborativo resultante.

A reavaliação das funções *benchmark* no contexto C-FRANNK mostra claramente que o método com subpopulações tem melhores resultados que o método FRANNK, e ainda é melhor que métodos adaptativos tradicionais (JADE, jDE). C-FRANNK também apresentou resultados promissores quando comparado com outros métodos que usam AI para o ajuste dinâmico de parâmetros, tais como os que utilizam lógica *fuzzy* combinada com os métodos PSO e *Gravitational Search,* as quais foram as únicas metodologias que utilizam AI no ajuste de parâmetros disponíveis até o momento na literatura para estudo comparativo.

O aprendizado neural associado à otimização, da maneira como foi proposto nesta tese, representa uma forma inovadora e eficiente para resolução de diferentes tipos de problemas de otimização, o que faz como natural a continuidade de pesquisas desta natureza buscando ampliar o que aqui foi visto, como a melhor discussão das características do aprendizado, a realização do treinamento usando problemas de outra natureza e o uso de outras técnicas de aprendizado de máquinas.

APÊNDICE A - ANÁLISE *PINCH*

A análise *Pinch* é uma metodologia desenvolvida para o cálculo de redes de trocadores de calor (RTC) que se baseia nos fundamentos termodinâmicos segundo o qual identifica-se um ponto crítico (ponto *Pinch*) no qual qualquer troca térmica entre as correntes passando por ele necessita de consumo adicional de utilidades quente e fria para que não violem a primeira e segunda leis da termodinâmica. Assim, o ponto *Pinch* possibilita identificar e selecionar as trocas térmicas entre as correntes da RTC que não necessitam do consumo de utilidades, ou seja, possibilita encontrar o menor gasto de utilidades durante o processo de integração energética (WESTPHALEN & MACIEL, 1999).

Na Figura 39 é apresentado o resultado do uso do método *Pinch* descrevendo as temperaturas em função das entalpias com curvas compostas de corrente quente e fria. Neste gráfico é possível observar o ponto *Pinch* em amarelo e os valores de diferença de entalpia associados aos retângulos em verde, azul e vermelho, representam, respectivamente os gastos de utilidade fria, recuperação de calor e gasto de utilidade quente.

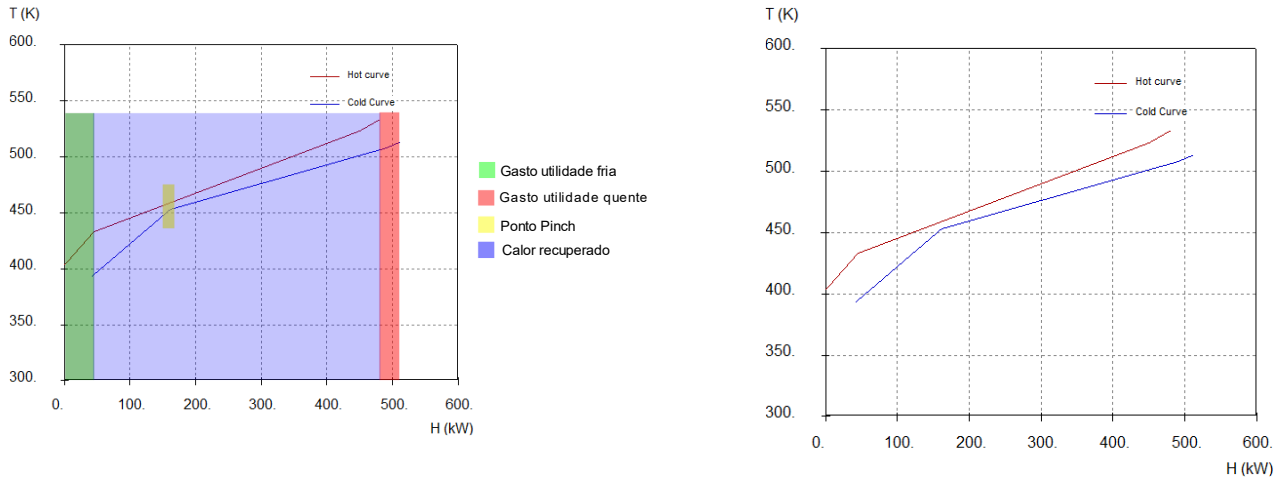

Figura 39: Análise *Pinch* (gerada utilizando o software Hint 2.2).

A obtenção das curvas compostas cumulativas que caracterizam a análise *Pinch* é iniciada fazendo a escolha de uma diferença mínima de temperatura de troca térmica (*DTmin*) e seguindo as etapas descritas abaixo (Linnhoff & Hindmarsh, 1983; WESTPHALEN & MACIEL, 1999).

Etapa 1: depois de fixar *DTmin*, gere uma curva composta de todas as correntes quentes considerando *-DTmin/2* e as correntes frias em *DTmin/2*.

Etapa 2: ajuste as temperaturas das curvas compostas considerando *DTmin/2* descrito na etapa anterior.

Etapa 3: avalie o balanço de temperatura para cada intervalo e confira se resulta em déficit ou excesso de energia considerando todas as trocas com variação de temperatura maior ou igual a *DTmin*.

Etapa 4: associe os trocadores através das dos intervalos de temperatura.

Etapa 5: calcule o maior fluxo negativo de calor e considere este como utilidade quente (*QH*).

Etapa 6: associe novamente os trocadores através do intervalo de temperatura começando com *QH* no primeiro intervalo. O valor de utilidade fria é obtido no último intervalo e o *Pinch* representa a temperatura para qual o fluxo de calor é zero.

A obtenção do ponto *Pinch*, assim como a descrição de múltiplas informações associadas a diferentes valores de *DTmin*, podem ser obtidas por softwares como o Hint 2.2 que é um *software* livre e educacional para projeto de RTC. O Hint 2.2 é capaz de gerar informações variadas associando diferentes valores de *DTmin*, tais como o cálculo de áreas, custos da RTC, a representação dos trocadores de calor em *grid*, a análise energética e as representações tradicionais de curvas compostas de entalpia e temperatura separadas em correntes quentes e frias , assim como uma curva composta de todas correntes denominada curva composta grande ou *grand coposite curve* (Martín & Mato, 2008).

O uso do Hint 2.2 é intuitivo sendo que o primeiro passo refere-se a adição das correntes que pode ser feita como mostra a interface gráfica representada na Figura 40 na qual a representação da esquerda refere-se à seleção da opção adição de corrente e a representação da direita refere-se às informações associadas a corrente que devem ser preenchidas (cada corrente pode ser especificada em termos da taxa de capacidade de calor , temperatura inicial e temperatura alvo).

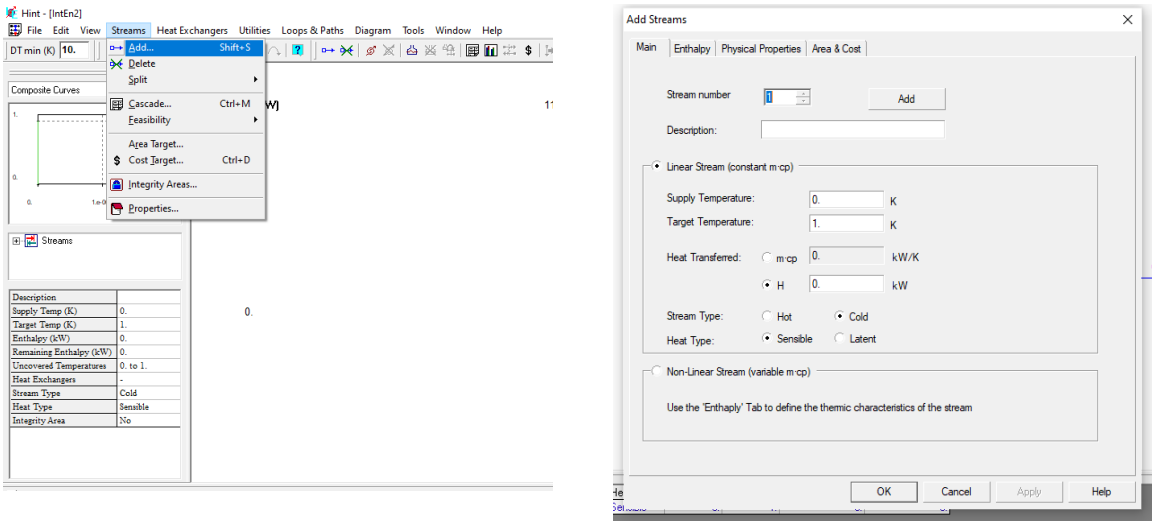

Figura 40: Adição de correntes (primeira etapa associada ao uso do Hint).

Uma vez adicionadas as correntes, o *software* já é capaz de fazer todas as análises descritas. Além disso, o mesmo tem outras funcionalidades como adição de trocadores de calor, o que possibilita avaliar uma RTC específica e mesmo representa-la. A Figura 41 mostra o acesso à referida opção.

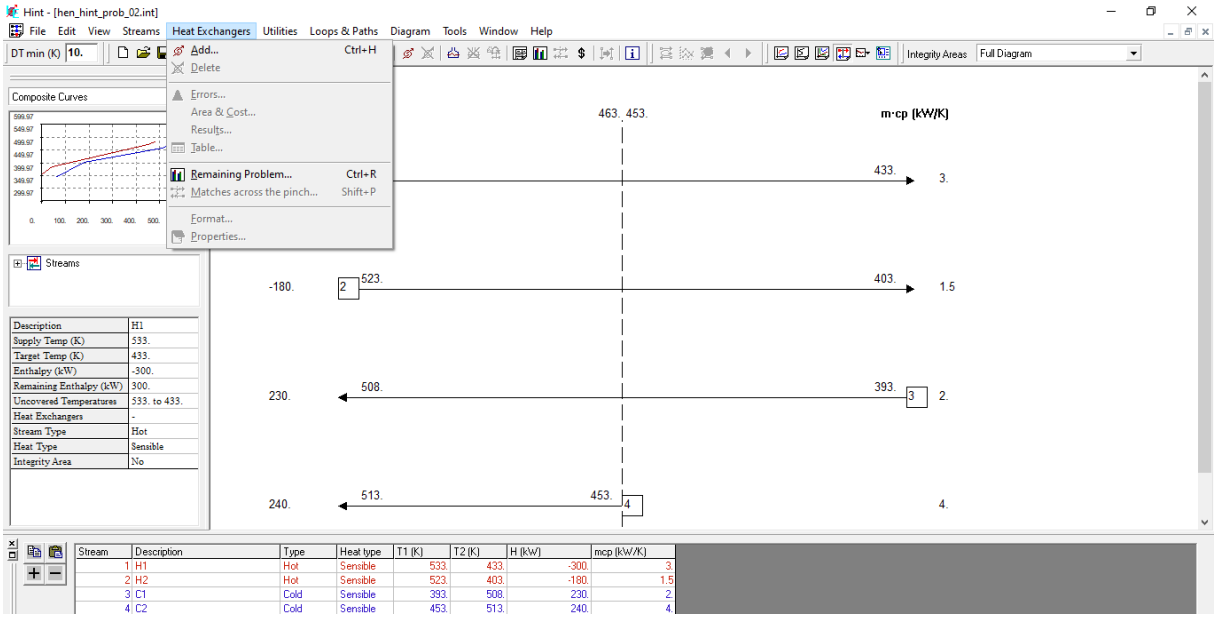

Figura 41: Adição de Trocadores de calor (para análise de uma RTC específica).

REFERÊNCIAS BIBLIOGRÁFICAS

Abdi, H., & Williams, L. J. (2010). Principal component analysis. *WIREs Computational Statistics* , *2*(4), 433–459. https://doi.org/https://doi.org/10.1002/wics.101

Adenso-Díaz, B., & Laguna, M. (2006). Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search. *Operations Research*, *54*(1), 99–114. http://www.jstor.org/stable/25146951

Aguitoni, M. C., Pavão, L. V., Siqueira, P. H., Jiménez, L., & Ravagnani, M. A. da S. S. (2018). Heat exchanger network synthesis using genetic algorithm and differential evolution. *Computers & Chemical Engineering*, *117*, 82–96. https://doi.org/https://doi.org/10.1016/j.compchemeng.2018.06.005

Albuquerque, A. T. de, Debs, M. K. El, & Melo, A. M. C. de. (2010). OTIMIZAÇÃO DE PAVIMENTOS DE EDIFÍCIOS COM ESTRUTURAS DE CONCRETO PRÉ-MOLDADO UTILIZANDO ALGORITMOS GENÉTICOS. In *Cadernos de Engenharia de Estruturas* (Vol. 12, Issue 54).

Aslan, S., Badem, H., & Karaboga, D. (2019). Improved quick artificial bee colony (iqABC) algorithm for global optimization. *Soft Computing*, *23*(24), 13161–13182. https://doi.org/10.1007/s00500-019-03858-y

Bansal, J. C., Joshi, S. K., & Sharma, H. (2018). Modified global best artificial bee colony for constrained optimization problems. *Computers & Electrical Engineering*, *67*, 365–382. https://doi.org/https://doi.org/10.1016/j.compeleceng.2017.10.021

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation*, *10*(6), 646–657. https://doi.org/10.1109/TEVC.2006.872133

Conceicao, E., & Maechler, M. (2020). DEoptimR. In *2020*. https://cran.r-project.org/web/packages/DEoptimR

Covington, P., Adams, J., & Sargin, E. (2016). Deep Neural Networks for YouTube Recommendations. *Proceedings of the 10th ACM Conference on Recommender Systems*, 191–198. https://doi.org/10.1145/2959100.2959190

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303–314. https://doi.org/10.1007/BF02551274

Den, J. Van, Fabian PedregosaFan, T. J., Grisel, O., & Jalali, A. (2020). *scikit-learn*. https://scikit-learn.org

Dolan, W. B., Cummings, P. T., & LeVan, M. D. (1989). Process optimization via simulated annealing: Application to network design. *AIChE Journal*, *35*(5), 725–736. https://doi.org/https://doi.org/10.1002/aic.690350504

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39. https://doi.org/10.1109/MCI.2006.329691

Fister, I., Suganthan, P. N., Fister, I., Kamal, S. M., Al-Marzouki, F. M., Perc, M., & Strnad, D. (2016). Artificial neural network regression as a local search heuristic for ensemble strategies in

differential evolution. *Nonlinear Dynamics*, *84*(2), 895–914. https://doi.org/10.1007/s11071-015-2537-8

Foulds, L. R. (1984). *Combinatorial Optimization for Undergraduates*. Springer US. https://doi.org/10.1007/978-1-4613-9511-9

Gong, W., Cai, Z., & Jiang, L. (2008). Enhancing the performance of differential evolution using orthogonal design method. *Applied Mathematics and Computation*, *206*(1), 56–69. https://doi.org/https://doi.org/10.1016/j.amc.2008.08.053

Goudos, S. K., Baltzis, K. B., Antoniadis, K., Zaharis, Z. D., & Hilas, C. S. (2011). A comparative study of common and self-adaptive differential evolution strategies on numerical benchmark problems. *Procedia Computer Science*, *3*, 83–88. https://doi.org/https://doi.org/10.1016/j.procs.2010.12.015

Holland, J. H. (1975). Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. In *Ann Arbor University of Michigan Press 1975*.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251–257. https://doi.org/https://doi.org/10.1016/0893-6080(91)90009-T

Huo, Z., Zhao, L., Yin, H., & Ye, J. (2012). A hybrid optimization strategy for simultaneous synthesis of heat exchanger network. *Korean Journal of Chemical Engineering*, *29*(10), 1298–1309. https://doi.org/10.1007/s11814-012-0007-2

Jamil, M., & Yang, X. S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, *4*(2), 150–194. https://doi.org/10.1504/IJMMNO.2013.055204

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–471. https://doi.org/10.1007/s10898-007-9149-x

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

Khorasany, R. M., & Fesanghary, M. (2009). A novel approach for synthesis of cost-optimal heat exchanger networks. *Computers & Chemical Engineering*, *33*(8), 1363–1370. https://doi.org/https://doi.org/10.1016/j.compchemeng.2008.12.004

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. *R News*, *2/3*, 12–22.

Linnhoff, B., & Hindmarsh, E. (1983). The pinch design method for heat exchanger networks. *Chemical Engineering Science*, *38*(5), 745–763. https://doi.org/https://doi.org/10.1016/0009-2509(83)80185-7

Liu, H., Cai, Z., & Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing*, *10*(2), 629–640. https://doi.org/https://doi.org/10.1016/j.asoc.2009.08.031

Liu, J., & Lampinen, J. (2005). A Fuzzy Adaptive Differential Evolution Algorithm. *Soft Computing*, *9*(6), 448–462. https://doi.org/10.1007/s00500-004-0363-x

Martín, Á., & Mato, F. A. (2008). Hint: An educational software for heat exchanger network design with the pinch method. *Education for Chemical Engineers*, *3*(1), e6–e14.

https://doi.org/https://doi.org/10.1016/j.ece.2007.08.001

Mayo, M. (2017). *Neural Network Foundations, Explained: Updating Weights with Gradient Descent & Backpropagation*. KDnuggets. https://www.kdnuggets.com/2017/10/neural-network-foundations-explained-gradient-descent.html

Morar, M., & Agachi, P. S. (2010). Review: Important contributions in development and improvement of the heat integration techniques. *Computers & Chemical Engineering*, *34*(8), 1171–1179. https://doi.org/https://doi.org/10.1016/j.compchemeng.2010.02.038

Moussa, T. M., & Awotunde, A. A. (2018). Self-adaptive differential evolution with a novel adaptation technique and its application to optimize ES-SAGD recovery process. *Computers & Chemical Engineering*, *118*, 64–76. https://doi.org/https://doi.org/10.1016/j.compchemeng.2018.07.018

Negri, L. H. (2020). A fast differential evolution module. In *23/01/2020*. https://pypi.org/project/PyFDE/

Nielsen, J. S., Weel Hansen, M., & bay Joergensen, S. (1996). Heat exchanger network modelling framework for optimal design and retrofitting. *Computers & Chemical Engineering*, *20*, S249–S254. https://doi.org/https://doi.org/10.1016/0098-1354(96)00052-X

Olivas, F., Valdez, F., Melin, P., Sombra, A., & Castillo, O. (2019). Interval type-2 fuzzy logic for dynamic parameter adaptation in a modified gravitational search algorithm. *Information Sciences*, *476*, 159–175. https://doi.org/https://doi.org/10.1016/j.ins.2018.10.025

Ortiz-Boyer, D., Hervás-Martínez, C., & García-Pedrajas, N. (2005). CIXL2: A crossover operator for evolutionary algorithms based on population features. *Journal of Artificial Intelligence Research*, *24*(June 2014), 1–48. https://doi.org/10.1613/jair.1660

Pagano, M., & Gauvreau, K. (2013). Regressão Logística. In *Princípios de Bioestatística* (2nd ed., pp. 415–430). Cengage Learning.

Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. *Trans. Evol. Comp*, *13*(2), 398–417. https://doi.org/10.1109/TEVC.2008.927706

Rathjens, M., & Fieg, G. (2020). A novel hybrid strategy for cost-optimal heat exchanger network synthesis suited for large-scale problems. *Applied Thermal Engineering*, *167*, 114771. https://doi.org/https://doi.org/10.1016/j.applthermaleng.2019.114771

Richert, W., & Coelho, L. P. (2013). *Building Machine Learning Systems with Python* (pp. 105–109). Packt Publishing.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Segaran, T. (2007). *Programming Collective Intelligence* (M. T. O'Brien (ed.); 1st ed.). O'Brien, Mary Treseler.

Simon, P. W. and. (2013). *Too Big to Ignore : The Business Case for Big Data*. Wiley. https://doi.org/DOI:10.1002/9781119204039

Storn, R. (1996). On the usage of differential evolution for function optimization. *Proceedings of North American Fuzzy Information Processing*, 519–523. https://doi.org/10.1109/NAFIPS.1996.534789

Storn, Rainer, & Price, K. (1997). Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, *11*(4), 341–359.

https://doi.org/10.1023/A:1008202821328

Tsafarakis, S., Zervoudakis, K., Andronikidis, A., & Altsitsiadis, E. (2020). Fuzzy self-tuning differential evolution for optimal product line design. *European Journal of Operational Research*. https://doi.org/https://doi.org/10.1016/j.ejor.2020.05.018

Wang, H., Wang, W., Zhou, X., Sun, H., Zhao, J., Yu, X., & Cui, Z. (2017). Firefly algorithm with neighborhood attraction. *Information Sciences*, *382–383*, 374–387. https://doi.org/https://doi.org/10.1016/j.ins.2016.12.024

Wang, S. L., Morsidi, F., Ng, T. F., Budiman, H., & Neoh, S. C. (2020). Insights into the effects of control parameters and mutation strategy on self-adaptive ensemble-based differential evolution. *Information Sciences*, *514*, 203–233. https://doi.org/https://doi.org/10.1016/j.ins.2019.11.046

Wang, S., Li, Y., & Yang, H. (2019). Self-adaptive mutation differential evolution algorithm based on particle swarm optimization. *Applied Soft Computing*, *81*, 105496. https://doi.org/https://doi.org/10.1016/j.asoc.2019.105496

WESTPHALEN, D. L., & MACIEL, M. . . W. (1999). Pinch analysis based on rigorous physical properties. *Brazilian Journal of Chemical Engineering*, *16*(3), 279–284. https://doi.org/10.1590/S0104-66321999000300006

White, T., Pagurek, B., & Oppacher, F. (1998). Ant Search with Genetic Algorithms: Application to Path Finding in Networks. *Combinatorial Optimization '98*.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/4235.585893

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-04944-6_14

Zhang, J., & Sanderson, A. C. (2009). JADE: Adaptive Differential Evolution With Optional External Archive. *IEEE Transactions on Evolutionary Computation*, *13*(5), 945–958. https://doi.org/10.1109/TEVC.2009.2014613

Zhaoyi, H., Liang, Z., Hongchao, Y., & Jianxiong, Y. (2013). Simultaneous synthesis of structural-constrained heat exchanger networks with and without stream splits. *The Canadian Journal of Chemical Engineering*, *91*(5), 830–842. https://doi.org/https://doi.org/10.1002/cjce.21702

Zomaia, A. Y. (2006). *Handbook of Nature-Inspired and Innovative Computing*.