

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO  
INSTITUTO DE COMPUTAÇÃO  
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

SIDNEY ALVES DE OUTEIRO

ACOPLAMENTO ENTRE UM MODELO CLASSIFICADOR E UM ALGORITMO  
DE OTIMIZAÇÃO POPULACIONAL APLICADO PARA DETECÇÃO DE  
INSTABILIDADE ECONÔMICA EM MALHAS ELÉTRICAS

RIO DE JANEIRO  
2023

SIDNEY ALVES DE OUTEIRO

ACOPLAMENTO ENTRE UM MODELO CLASSIFICADOR E UM ALGORITMO  
DE OTIMIZAÇÃO POPULACIONAL APLICADO PARA DETECÇÃO DE  
INSTABILIDADE ECONÔMICA EM MALHAS ELÉTRICAS

Trabalho de conclusão de curso de graduação  
apresentado ao Instituto de Computação da  
Universidade Federal do Rio de Janeiro como  
parte dos requisitos para obtenção do grau de  
Bacharel em Ciência da Computação.

Orientadora: Profa. Carolina Gil Marcelino

RIO DE JANEIRO

2023

## CIP - Catalogação na Publicação

A474a      Alves de Outeiro, Sidney  
              ACOPLAMENTO ENTRE UM MODELO CLASSIFICADOR E UM  
              ALGORITMO DE OTIMIZAÇÃO POPULACIONAL APLICADO PARA  
              DETECÇÃO DE INSTABILIDADE ECONÔMICA EM MALHAS  
              ELÉTRICAS / Sidney Alves de Outeiro. -- Rio de  
              Janeiro, 2023.  
              52 f.

              Orientadora: Carolina Gil Marcelino.  
              Trabalho de conclusão de curso (graduação) -  
              Universidade Federal do Rio de Janeiro, Instituto  
              de Computação, Bacharel em Ciência da Computação,  
              2023.

              1. inteligência artificial. 2. algoritmos  
              populacionais. 3. malha elétrica inteligente . 4.  
              multiprocessamento. I. Gil Marcelino, Carolina,  
              orient. II. Título.


SIDNEY ALVES DE OUTEIRO

ACOPLAMENTO ENTRE UM MODELO CLASSIFICADOR E UM ALGORITMO  
DE OTIMIZAÇÃO POPULACIONAL APLICADO PARA DETECÇÃO DE  
INSTABILIDADE ECONÔMICA EM MALHAS ELÉTRICAS

Trabalho de conclusão de curso de graduação  
apresentado ao Instituto de Computação da  
Universidade Federal do Rio de Janeiro como  
parte dos requisitos para obtenção do grau de  
Bacharel em Ciência da Computação.


Aprovado em 11 de dezembro de 2023

BANCA EXAMINADORA:

Documento assinado digitalmente  
 **CAROLINA GIL MARCELINO**  
Data: 16/12/2023 21:06:01-0300  
Verifique em <https://validar.iti.gov.br>


---

Carolina Gil Marcelino, DSc.  
(UFRJ)

Documento assinado digitalmente  
 **SILVANA ROSSETTO**  
Data: 17/12/2023 11:35:09-0300  
Verifique em <https://validar.iti.gov.br>

---

Silvana Rossetto, DSc.  
(UFRJ)

Documento assinado digitalmente  
 **LAURA DE OLIVEIRA FERNANDES MORAES**  
Data: 18/12/2023 10:15:18-0300  
Verifique em <https://validar.iti.gov.br>

---

Laura de Oliveira Fernandes Moraes  
DSc. (UNIRIO)

Dedicatória: À minha mãe e ao meu irmão que me ensinaram que juntos podemos enfrentar todas as adversidades que a vida impor sobre nós. E por sempre priorizarem a minha felicidade.

## AGRADECIMENTOS

Primeiro gostaria de agradecer a minha família pelo apoio desde o início.

Também tenho a agradecer a minha professora orientadora Carolina, por me acolher e dar todo o apoio e o material necessário para continuar com o projeto. Obrigado também a todos os professores da UFRJ, que com paciência, construíram meu caminho acadêmico e todo o conhecimento necessário para continuar até o final.

Um agradecimento especial ao Laboratório de Redes e Multimídia do Núcleo de Computação Eletrônica e ao professor coordenador Claudio Miceli, por me darem oportunidade para aprender além da sala de aula e por me fornecer meios para que eu pudesse me manter na faculdade. E um agradecimento ao LC3 e ao Bruno do Carmo pelo auxílio com a infraestrutura utilizada nesse trabalho.

Quero agradecer aos meus amigos que a UFRJ me proporcionou por sempre me incentivarem e me acompanharam nessa etapa da minha vida. E por fim e não menos importante, obrigado aos projetos como GRIS, Analytica e as empresas que atuei que me ajudou a criar um pensamento mais crítico e auxiliou a me tornar um profissional melhor. Muito obrigado a todos.

*"We can only see a short distance ahead  
but we can see plenty there that needs to be done."*

**Alan Turing**

## RESUMO

A energia elétrica é um dos temas dos Objetivos de Desenvolvimento Sustentável (ODS), buscando até 2030 estabelecer acesso confiável, sustentável e economicamente viável para todos. Com o advento das malhas elétricas inteligentes, a transição é facilitada para fontes energéticas mais sustentáveis e possibilita comunicação entre produtor e consumidor, viabilizando a compreensão da demanda energética que a malha possui. Contudo, apesar de existir meios para responder essa demanda da malha, a flutuação do preço da malha torna-se instável no momento que o sistema físico torna-se instável. Cenários de instabilidade econômica inclui superprodução ou subprodução de energia elétrica, o que pode comprometer a operação da própria malha elétrica e da operação da indústria que depende da energia fornecida. Este trabalho de conclusão de curso, propõe um modelo classificador acoplado a um algoritmo de otimização populacional, que realiza treinamento utilizando multiprocessamento, para detectar instabilidade econômica de malhas elétricas inteligentes.

**Palavras-chave:** inteligência artificial; algoritmos populacionais; malha elétrica inteligente; multiprocessamento.



## ABSTRACT

Electric energy is one of the themes of the Sustainable Development Goals (SDGs), seeking to establish reliable, sustainable and economically accessible access for all by 2030. With the advent of smart electrical grids, the transition to more sustainable energy sources is facilitated and enables communication between producer and consumer, enabling the understanding of the energy demand that the grid has. However, despite there being ways to respond to this demand for the network, the fluctuation in the price of the network becomes unstable when the physical system becomes unstable. Economic instability scenarios includes overproduction or underproduction of electricity, which may compromise the operation of the electrical network itself and the operation of the industry that depends on the energy provided. This course completion work proposes a classifier model coupled to a population optimization algorithm, which performs training using multiprocessing, to detect economic instability of intelligent electrical grids.

**Keywords:** artificial intelligence; population algorithms; smart grids; multiprocessing.

## LISTA DE ILUSTRAÇÕES

Figura 1 – Integração de fontes de energias e consumidores em uma Smart Grid. (Fonte: Elaboração Própria) . . . . .	16
Figura 2 – Resultado de uma transformação linear encontrada pelo NCA. (Fonte: Elaboração Própria) . . . . .	20
Figura 3 – Ilustração do processo de classificação do KNN . . . . .	21
Figura 4 – Visualização da operação do algoritmo do gradiente descendente. Após encontrar o vetor gradiente a rotina caminha sentido contrário realizando a descida em busca de um ponto de mínimo. É possível notar passo a passo a evolução da descida. (Fonte: Elaboração Própria). . . . .	23
Figura 5 – Exemplo de distribuição inicial das partículas ao longo do espaço de solução. O símbolo “x” em vermelho representa as partículas no espaço de busca (Fonte: Elaboração Própria) . . . . .	24
Figura 6 – Diagrama do fluxo de execução do PSO . . . . .	25
Figura 7 – Atualização em paralelo do enxame. (Fonte: Elaboração Própria) . . . . .	26
Figura 8 – Gráfico tridimensional da função <i>Sphere</i> . . . . .	27
Figura 9 – Gráfico de curva de nível, onde cada ‘x’ colorido representa um conjunto de partículas que utilizará um dos núcleos de processamento. . . . .	27
Figura 10 – Gráfico de curva de nível da função <i>Sphere</i> , onde cada partícula é representada por um “x’ ’ colorido . . . . .	28
Figura 11 – Gráfico de curva de nível da função <i>Sphere</i> , onde cada partícula é representada por um "x" colorido . . . . .	28
Figura 12 – Topologia da <i>Smart Grid</i> tratada. (Fonte: Elaboração Própria) . . . . .	31
Figura 13 – Boxplot da média das acurácias obtidas para amostragem de 1500 observações. . . . .	35
Figura 14 – Matriz de Confusão da melhor configuração encontrada. . . . .	36
Figura 15 – Boxplot do tempo de execução do modelo sequencial e modelo paralelizado. . . . .	38
Figura 16 – Resultados encontrados ao avaliar modelos baseados em vizinhança. A linha, presente em cada uma das barras, representa o intervalo de confiança de 95%. . . . .	40

## LISTA DE CÓDIGOS

Código 1	classe NCA-C . . . . .	46
Código 2	Função PSO . . . . .	49

## LISTA DE TABELAS

Tabela 1 – Comparação dos modelos de malha elétrica. . . . .	17
Tabela 2 – Atributos presentes no conjunto de dados . . . . .	32
Tabela 3 – Tabela dos resultados encontrados do experimento. . . . .	34
Tabela 4 – Resultados da execução paralela e sequencial do modelo proposto. . . . .	37
Tabela 5 – Resultados encontrados ao comparar modelos baseados em vizinhança. . . . .	39

## LISTA DE ABREVIATURAS E SIGLAS

KNN	<i>K-Nearest Neighbors</i>
NCA	<i>Neighborhood Component Analysis</i>
PSO	<i>Particle Swarm Optimization</i>
ACC	Acurácia
MCC	<i>Matthew Correlation Coefficient</i>
ONU	Organização das Nações Unidas

## LISTA DE SÍMBOLOS

$\cap$	Interseção
$\forall$	Para todo
$\frac{\partial f}{\partial x}$	Derivada parcial de $x$ na função $f$
$\frac{\partial f}{\partial y}$	Derivada parcial de $y$ na função $f$
$\in$	Pertence
$x_i$	Posição do eixo $x$ de uma determinada partícula no instante $i$
$y_i$	Posição do eixo $y$ de uma determinada partícula no instante $i$
$v_i^t$	Velocidade da partícula $i$ no instante $t$
$p_i$	Melhor posição encontrada pela partícula $i$
$p_g$	Melhor posição encontrada pelo enxame
$x_i^t$	Posição da partícula $i$ no instante $t$
$r_1$	Número real aleatório contido no intervalo $[0,1]$
$r_2$	Número real aleatório contido no intervalo $[0,1]$
$w$	Peso associado ao vetor velocidade do estado atual de uma determinada partícula
$c_1$	Peso associado ao vetor gerado pela posição atual de uma partícula com a melhor posição da própria partícula
$c_2$	Peso associado ao vetor gerado pela posição atual de uma partícula com a melhor posição do enxame

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>15</b>
1.1	OBJETIVOS . . . . .	18
<b>2</b>	<b>CONCEITOS BÁSICOS . . . . .</b>	<b>19</b>
2.1	APRENDIZADO DE MÁQUINA . . . . .	19
2.1.1	Neighborhood Component Analysis . . . . .	19
2.1.2	K-Nearest Neighbors . . . . .	20
2.2	OTIMIZAÇÃO . . . . .	21
2.2.1	Otimização Mono-objetivo . . . . .	21
2.2.2	Gradiente Descendente . . . . .	22
2.2.3	Algoritmos Populacionais . . . . .	23
2.2.4	Particle Swarm Optimization . . . . .	23
2.3	COMPUTAÇÃO PARALELA . . . . .	25
2.3.1	Paralelismo em Aprendizado de Máquina . . . . .	25
2.3.1.1	Experimento preliminar de PSO sequencial vs. paralelo . . . . .	26
<b>3</b>	<b>PROPOSTA . . . . .</b>	<b>29</b>
3.1	MODELO DE CLASSIFICAÇÃO BASEADO NO NCA . . . . .	29
3.2	NCA + PSO . . . . .	30
<b>4</b>	<b>EXPERIMENTOS REALIZADOS . . . . .</b>	<b>31</b>
4.1	AMOSTRAGEM DOS DADOS E TAMANHO DO ENXAME . . . . .	32
4.1.1	Plano Experimental . . . . .	32
4.1.2	Análise dos Resultados Encontrados . . . . .	33
4.2	AVALIAÇÃO DO TEMPO DE EXECUÇÃO . . . . .	36
4.2.1	Plano Experimental . . . . .	36
4.2.2	Análise dos Resultados Encontrados . . . . .	37
4.3	MODELOS BASEADOS EM VIZINHANÇA . . . . .	38
4.3.1	Plano Experimental . . . . .	38
4.3.2	Análise dos Resultados Encontrados . . . . .	39
<b>5</b>	<b>CONCLUSÃO . . . . .</b>	<b>41</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>43</b>
	<b>APÊNDICE A – CÓDIGO DA CLASSE NCA-C . . . . .</b>	<b>46</b>





## 1 INTRODUÇÃO

A demanda contínua de energia elétrica é um dos desafios mapeados pela Organização das Nações Unidas (ONU) que visam garantir o acesso a fontes de energia viáveis, sustentáveis e modernas para todos. Os Objetivos de Desenvolvimento Sustentável (ODS) são uma coleção de metas estipuladas pela ONU a serem cumpridas até 2030. Dentre esses desafios relacionados ao tema de energia limpa e acessível temos:

**Objetivo 7.** Assegurar o acesso confiável, sustentável, moderno e a preço acessível à energia para todas e todos:

**7.1** Até 2030, assegurar o acesso universal, confiável, moderno e a preços acessíveis a serviços de energia;

**7.2** Até 2030, aumentar substancialmente a participação de energias renováveis na matriz energética global;

**7.3** Até 2030, dobrar a taxa global de melhoria da eficiência energética;

**7.a** Até 2030, reforçar a cooperação internacional para facilitar o acesso à pesquisa e tecnologias de energia limpa, incluindo energias renováveis, eficiência energética e tecnologias de combustíveis fósseis avançadas e mais limpas, e promover o investimento em infraestrutura de energia e em tecnologias de energia limpa, e;

**7.b** Até 2030, expandir a infraestrutura e modernizar a tecnologia para o fornecimento de serviços de energia modernos e sustentáveis para todos nos países em desenvolvimento, particularmente nos países menos desenvolvidos, nos pequenos Estados insulares em desenvolvimento e nos países em desenvolvimento sem litoral, de acordo com seus respectivos programas de apoio. Extraído de (ONU, 2015).

Diante das metas estabelecidas, torna-se necessário o desenvolvimento de novas tecnologias que possibilitem a transição para fonte energéticas renováveis e a produção eficiente de energia. Atualmente, o modelo consolidado de malhas elétricas possui comunicação unidirecional por natureza, em consequência das informações acerca dos consumidores não serem consideradas para realizar o controle de produção sob demanda. Essas malhas convencionais possuem uma topologia hierárquica, onde as entidades presentes entre o produtor e o consumidor final possibilitam pontos de falhas ao realizar a distribuição e transmissão de energia elétrica. Com isso, esse tipo de malha está mais suscetível a falhas causadas por efeito cascata. Além disso, devido ao seu controle ser centralizado, a expansão da malha elétrica possui um custo logístico de manutenção muito elevado. Nesse modelo, a perda da energia gerada torna-se um problema devido à dissipação da energia ao longo da rede de distribuição e da rede de transmissão, necessitando de superprodução de energia para conseguir suprir a demanda dos consumidores (FARHANGI, 2009).

As redes elétricas inteligentes (*Smart Grids*), consideradas a próxima evolução do modelo de malhas elétricas, surgem para corrigir as deficiências associadas às malhas elétricas centralizadas e unidimensionais, por meio da possibilidade de comunicação bidimensional em uma topologia de rede distribuída com capacidade adaptativa. Ela difere da malha tradicional que, devido à incapacidade de coletar dados de consumo, precisa ter uma superprodução de energia elétrica para suprir a necessidade futura dos consumidores. A migração de fontes de energia não-renováveis para renováveis torna-se um processo menos custoso devido à flexibilidade da *Smart Grid* em adotar distintas fontes de energia, como ilustrado na Figura 1 (FARHANGI, 2009).

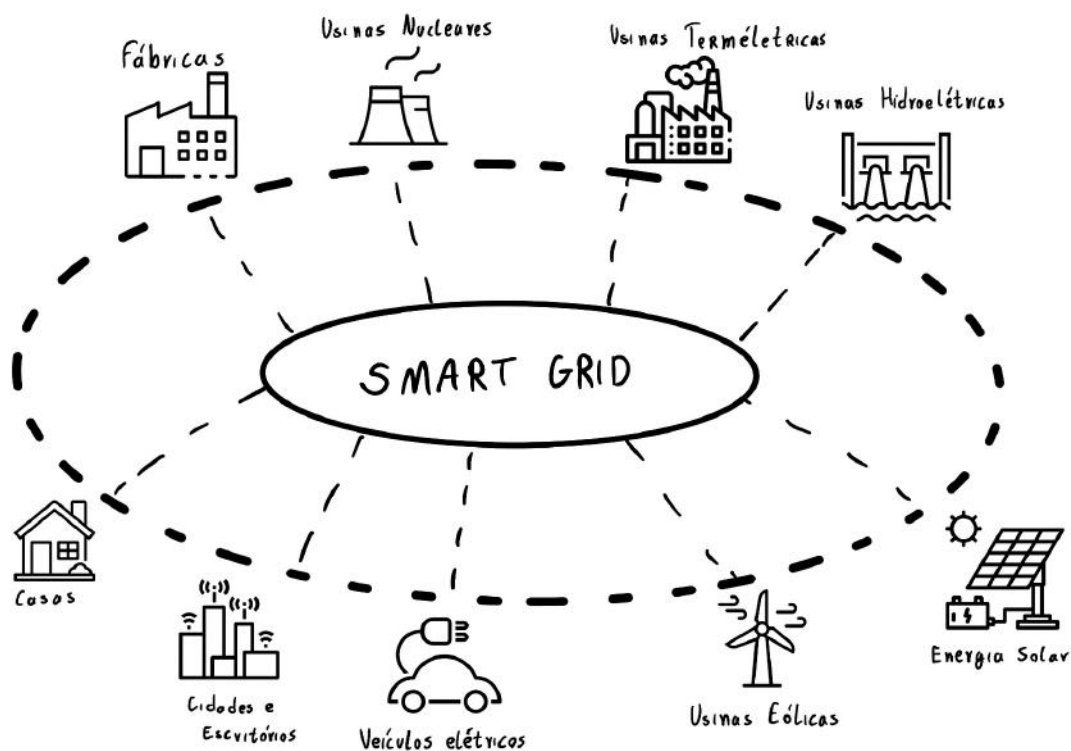


Figura 1 – Integração de fontes de energias e consumidores em uma Smart Grid. (Fonte: Elaboração Própria)

Por meio de um sistema de distribuição de energia elétrica que opera de forma autônoma ou em conjunto com a rede elétrica principal, as *Smart Grids* fornecem eletricidade de forma confiável e eficiente, ao reduzir a perda de energia e ao permitir que a malha se adapte a demanda dinâmica, em pequenas áreas geográficas. Devido a esse potencial de adaptação a demanda dinâmica, a necessidade de investimentos constantes em infraestrutura para armazenar a energia excedente, é reduzida drasticamente. As *Smart Grids* apresentam um progresso tecnológico significativo na produção, distribuição e armazenamento de energia, uma vez que oferecem resiliência energética, possibilitando o suprimento de eletricidade em cenários de emergência e aumentando a integração de fontes renováveis,

contribuindo para a transição energética. Com as recentes propostas de *Smart Grids* que utilizam sensores e modelos de Inteligência Artificial (FANG et al., 2011), há expectativa para que a produção energética sob demanda dinâmica torne-se uma realidade. Na Tabela 1, é possível observar uma breve comparação entre a malha elétrica tradicional e a *Smart Grid*.

<b>Malha Tradicional</b>	<b><i>Smart Grid</i></b>
Geração centralizada, controlada por utilitários com combustíveis fósseis tradicionais.	Operação em tempo real de ativos de geração com demanda e oferta em tempo real.
Operação cega de ativos de geração com apenas uma previsão de demanda	Fluxos bidirecionais com consciência situacional total e opções de resposta.
Erros passivos e ilimitados por tentativa e erro	Consumidores ativos e capacitados com feedback em tempo real.
Foco na adição de infraestrutura para atender à demanda	Foco na otimização da infraestrutura existente para atender à demanda.
Modelos operacionais estáticos com mínima otimização	Modelos operacionais dinâmicos com otimização em tempo real
A rede é vulnerável à intrusão cibernética ou desastres naturais	A rede é mais resiliente a intrusão cibernética e desastres naturais

Tabela 1 – Comparação dos modelos de malha elétrica.

Entretanto, a implementação e operação de *Smart Grids* não estão isentas de desafios como, por exemplo, a complexidade da gestão energética, a necessidade de investimentos substanciais e a regulação adequada (FANG et al., 2011). Uma *Smart Grid* descentralizada, que possui resposta dinâmica de demanda, tem como benefícios a melhora do desempenho do mercado, redução da volatilidade do preço e aumento da oferta de diferentes fontes energéticas para o consumidor final. Além disso, tem o aumento da confiabilidade ao viabilizar a diversidade de fontes energéticas que podem ser incorporados na malha, reduzindo a quantidade de interrupções (SCHÄFER et al., 2015). A partir da estabilidade entre a produção e consumo de energia, considerando a perda devido à dissipação de energia, é possível inferir a estabilidade econômica da *Smart Grid* (SCHÄFER et al., 2015). Contudo, determinar a estabilidade linear do sistema é um problema NP-difícil (GURVITS; OLSHEVSKY, 2009) que escala conforme aumenta o número de participantes da malha elétrica (ARZAMASOV; BÖHM; JOCHEM, 2018). E considerar a necessidade que a operação da malha elétrica seja contínuo, para que o fornecimento de energia não seja interrompido e, conseqüentemente, interromper a operação de hospitais, da indústria e áreas urbanas.

Diante disso, para determinar a estabilidade linear do sistema, pode-se modelar o sistema físico da *Smart Grid* (ARZAMASOV; BÖHM; JOCHEM, 2018) utilizando uma equação diferencial ordinária para geração de energia. Dessa equação, é possível obter um quasi-polinômio com infinitas soluções. Diante das infinitas soluções que o quasi-

polinômio pode possuir, apenas um conjunto finito das soluções pode ter uma parte real e positiva. Essas soluções com partes reais e positivas determinam a estabilidade do sistema (SCHÄFER et al., 2016).

## 1.1 OBJETIVOS

Este trabalho de conclusão de curso propõe determinar a estabilidade econômica de *Smart Grids*, não como um sistema de equações diferenciais ordinária, que é um problema NP-difícil, mas como um problema de classificação de padrões em função de informações dos participantes da malha. E ao considerar que a malha elétrica exigirá um monitoramento contínua a cerca da estabilidade da malha, exige uma resposta com o mínimo de atraso possível. Será proposto um modelo de aprendizado de máquina, baseado no *Neighborhood Component Analysis* (NCA), acoplado ao algoritmo *Particle Swarm Optimization* (PSO), um algoritmo de otimização populacional, para classificar a estabilidade econômica de *Smart Grids*. Nesse modelo proposto, será utilizado processamento paralelo para realizar a minimização do erro de classificação do modelo. Diante disso, será avaliado como o modelo proposto performa comparado a modelos mais consolidados. A partir de informações acerca de cada participante que compõe a *Smart Grid*, podemos caracterizar a estabilidade do sistema físico e determinar, conseqüentemente, a estabilidade econômica da mesma. Diante disso, teremos as seguintes perguntas que este trabalho buscará responder:

- Pergunta 1: O modelo proposto possui desempenho tão bom, ou melhor, que modelos mais consolidados?
- Pergunta 2: O modelo proposto consegue treinar e testar em um período de tempo menor que modelos baseados em vizinhança mais consolidados?

A organização dos seguintes capítulos será feita da seguinte forma, no segundo capítulo há os conceitos básicos utilizados para estruturar a solução proposta desse trabalho. No terceiro capítulo é apresentada a proposta, como a mesma foi implementada e as vantagens e desvantagens associadas. No quarto capítulo, há a metodologia e os experimentos realizados para responder às perguntas de pesquisa propostas. No quinto capítulo, há a conclusão do trabalho abordando as limitações encontradas e trabalhos futuros.

## 2 CONCEITOS BÁSICOS

Neste capítulo, o objetivo é esclarecer as principais terminologias, técnicas, algoritmos e tecnologias que estão relacionados a este trabalho, fornecendo uma visão geral e incluindo alguns exemplos elucidativos.

### 2.1 APRENDIZADO DE MÁQUINA

Aprendizado de máquina é a subárea de Inteligência Artificial que visa desenvolver modelos capazes de performar tarefas complexas ao detectar, extrapolar e adaptar-se a padrões e circunstâncias. Diante disso, temos três categorias de aprendizado dado pela natureza do problema a ser resolvido, seja esse um problema de classificação, predição ou agrupamento (ABU-MOSTAFA; MAGDON-ISMAIL; LIN, 2012).

- O **aprendizado supervisionado** busca através do conjunto de exemplos associar o atributo alvo com os demais atributos;
- O **aprendizado não-supervisionado** tem em vista mapear os dados fornecidos para um atributo que não se encontra presente explicitamente no conjunto de exemplo, mas que seria possível derivá-lo através dos padrões existentes no conjunto de dados fornecidos, e;
- O **aprendizado por reforço** busca por meio de um agente situado em um ambiente identificar quais ações o agente precisa tomar para executar uma determinada tarefa, as ações podem ser recompensadas ou penalizadas ao longo da etapa de treinamento.

#### 2.1.1 Neighborhood Component Analysis

O *Neighborhood Component Analysis* (NCA) (GOLDBERGER et al., 2004) é um algoritmo de aprendizado de máquina supervisionado cujo objetivo é aprender uma métrica de distância. O treinamento consiste em descobrir uma transformação linear  $A$  que remapeia os dados espacialmente, de maneira que os dados de treino que possuem a mesma classe estejam espacialmente próximo. No processo de treinamento é inicialmente realizado o cálculo das distâncias euclidianas, gerando uma matriz  $D_{N,N}$ , na qual  $N$  é o tamanho da amostra de treino. Ao aplicarmos a função *softmax*, teremos uma distribuição de probabilidades a fim de calcular a probabilidade de um exemplo  $i$  da amostra de treino assumir o mesmo rótulo de outro exemplo  $j$  da mesma. Na Equação 2.1, é possível observar como é calculada a probabilidade de uma observação assumir a mesma classe que outra. Com isso, teremos uma matriz de probabilidades  $P$ , onde a sua coluna principal seria preenchida com zeros.

$$p_{ij} = \begin{cases} \frac{e^{-\|Ax_i - Ax_j\|^2}}{\sum_{k \neq i} e^{-\|Ax_i - Ax_k\|^2}} & \text{se } i \neq j \\ 0 & \text{se } i = j \end{cases} \quad (2.1)$$

Assim, uma função de otimização é usada para descobrir uma transformação linear que maximize a probabilidade de um elemento da amostra de treino ser classificado pela aproximação de indivíduos que pertencem a uma mesma classe. Com isso, na Equação 2.2 teremos uma formulação de função objetivo que pode ser utilizada pelas funções de otimização.

$$\sum_i p_i = \sum_i \sum_{j \in C_i} p_{ij}. \quad (2.2)$$

Diante dessa transformação linear, ao usar uma estratégia de classificação teremos um desempenho melhor. Na Figura 2, temos um ponto com arestas o conectando aos demais pontos e a opacidade destas arestas indica a influência dos vizinhos da classificação do ponto raiz. Após a transformação linear, os vizinhos com a mesma classificação do ponto raiz possuirão as maiores influências na classificação do ponto raiz.

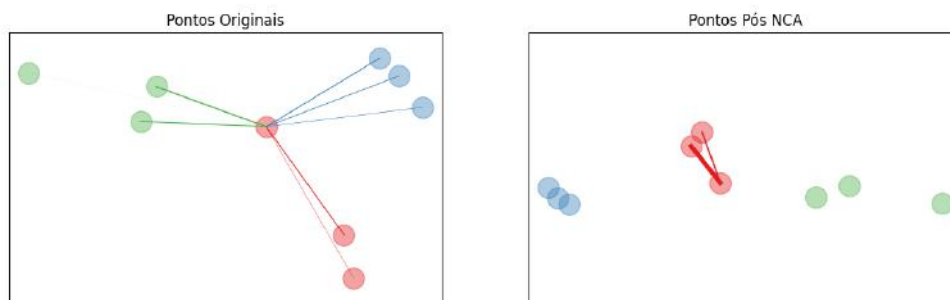


Figura 2 – Resultado de uma transformação linear encontrada pelo NCA. (Fonte: Elaboração Própria)

### 2.1.2 K-Nearest Neighbors

O K-Nearest Neighbors (KNN) (FIX; HODGES, 1989) (COVER; HART, 1967) é um algoritmo de aprendizado de máquina supervisionado baseado em vizinhança, que pode realizar uma classificação ou regressão. A partir de uma observação de teste, sua classificação seria determinada em função das  $k$  observações espacialmente mais próximas. Ao determinar os  $k$  vizinhos mais próximos, será contabilizado a quantidade de vizinhos que pertencem a cada uma das classes presentes no domínio do problema. E assim, a observação de teste será classificada com a classe que possuir a maior quantidade de vizinhos. Na Figura<sup>13</sup>, podemos observar um exemplo de uma classificação realizada pelo

<sup>1</sup> Fonte: <https://commons.wikimedia.org/wiki/File:KnnClassification.svg>

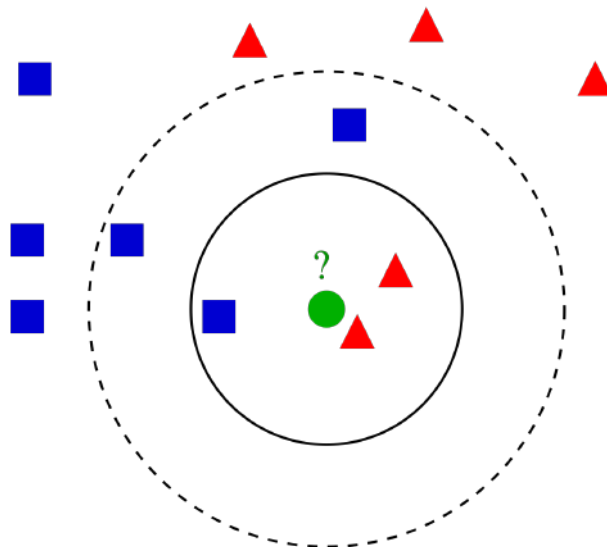


Figura 3 – Ilustração do processo de classificação do KNN

KNN. Ao utilizar três vizinhos para determinar a classificação do círculo verde, seria realizado uma votação entre os três vizinhos mais próximos, resultando que a classe seria a mesma dos triângulos vermelhos. E ao utilizar cinco vizinhos, teríamos como resultado que a classe do círculo verde seria a mesma que a dos quadrados azuis.

## 2.2 OTIMIZAÇÃO

Otimização é a grande área da matemática aplicada que busca por meio de algoritmos encontrar a melhor solução de uma função objetivo e seu domínio.

### 2.2.1 Otimização Mono-objetivo

Temos como um problema de otimização mono-objetivo uma função  $f : X \rightarrow Y$ , cujos elementos do domínio precisam estar sujeitos a um conjunto de restrições  $R$  para serem soluções viáveis. Em problemas de minimização, a solução ótima  $x^*$  da função  $f$  está sujeita ao conjunto de restrições  $R$ , dada pelo menor valor possível da imagem da função  $f$ .

$$f(x^*) \leq f(x) \quad \forall x \in R \cap X. \quad (2.3)$$

Problemas de otimização mono-objetivo que buscam encontrar o ponto máximo da imagem de  $f$  seguem a mesma estrutura com exceção da função objetivo  $f$  que precisa ser trocada por  $g \circ f$ , onde  $g(x) = -x$ . Assim, temos que problemas de minimização e maximização são análogos em otimização (RIBEIRO; KARAS, 2013).

### 2.2.2 Gradiente Descendente

O método do Gradiente Descendente (AMARI, 1967) é um algoritmo de otimização iterativo que, dado uma função  $f : X \rightarrow Y$  e um ponto inicial  $(x_0, y_0) \in X$  inicial, conseguimos, a partir das derivadas parciais, extrair um vetor que indique a direção de melhora para evoluir a posição do ponto inicial para uma solução mais próxima de um mínimo local da função  $f$ . Exemplificando, se temos uma superfície  $f(x, y) = \cos x + \sin y$  e uma partícula no ponto inicial dado por  $(x_0, y_0) = (0, \frac{\pi}{2})$ , teremos que as derivadas parciais da superfície são:

$$\frac{\partial f}{\partial x} = -\sin(x) \quad \frac{\partial f}{\partial y} = \cos y. \quad (2.4)$$

Ao avaliarmos o resultado das derivadas parciais no ponto inicial e combinarmos esses resultados, temos um vetor que indica a direção e o sentido de um ponto de máximo local. Contudo, como estaremos lidando com um problema de minimização, será necessário inverter o sentido do vetor. Na Figura 4, podemos observar que ao longo do tempo  $t$ , a partícula se deslocará do ponto inicial até um ponto de mínimo local ao inverter a direção do vetor gradiente. E, a cada iteração, a posição da partícula seria atualizada utilizando o sentido inverso do vetor gradiente e um coeficiente  $\alpha$ , que irá indicar o tamanho do deslocamento que a partícula irá realizar para se aproximar de um ponto de mínimo local. Assim temos a seguinte relação de recorrência:

$$x_{t+1} = x_t - \alpha \frac{\partial f}{\partial x} \quad y_{t+1} = y_t - \alpha \frac{\partial f}{\partial y}, \quad (2.5)$$

$$x_{t+1} = x_t + \alpha \sin x \quad y_{t+1} = y_t - \alpha \cos y. \quad (2.6)$$



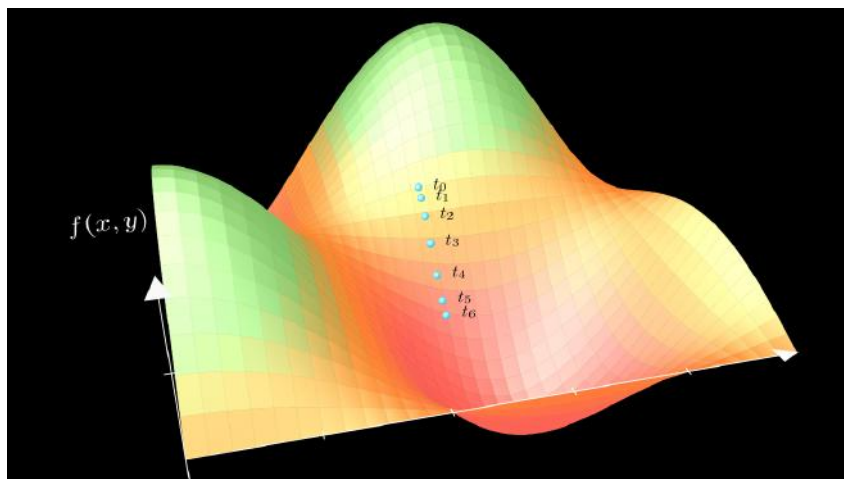


Figura 4 – Visualização da operação do algoritmo do gradiente descendente. Após encontrar o vetor gradiente a rotina caminha sentido contrário realizando a descida em busca de um ponto de mínimo. É possível notar passo a passo a evolução da descida. (Fonte: Elaboração Própria).

### 2.2.3 Algoritmos Populacionais

Algoritmos populacionais são uma família de heurísticas que buscam realizar a busca no domínio do problema a partir da cooperação ou competição entre indivíduos de um grupo. Diante disso, algoritmos populacionais podem trabalhar com informações de mais de uma solução e/ou as informações obtidas serão utilizadas para definir o próximo estado da sequência de soluções (RH, 2005). Exemplos de algoritmos populacionais são: *Particle Swarm Optimization*, *PSO* (KENNEDY; EBERHART, 1995), *Artificial Bee Colony*, *ABC* (KARABOGA et al., 2005) e *Ant Colony Optimization* (DORIGO; BIRATTARI; STUTZLE, 2006).

### 2.2.4 Particle Swarm Optimization

O *Particle Swarm Optimization* é um método de otimização por populações de funções contínuas não-lineares. Diante da função objetivo, serão geradas  $N$  partículas que irão compor o enxame, e cada partícula representa uma solução para a função objetivo. Tipicamente, as partículas são distribuídas de forma aleatória pelo espaço de solução. É possível observar um exemplo da distribuição homogênea na Figura 5.

A partir da distribuição inicial das partículas, a posição de todas as partículas do enxame será atualizada através do vetor velocidade de cada partícula, como pode ser observado na Equação 2.7. Esse processo será feito iterativamente até que todas as partículas tenham convergido ou que o número máximo de iterações determinada tenha sido atingida.

$$x_i^{t+1} = x_i^t + v_i^{t+1}. \quad (2.7)$$

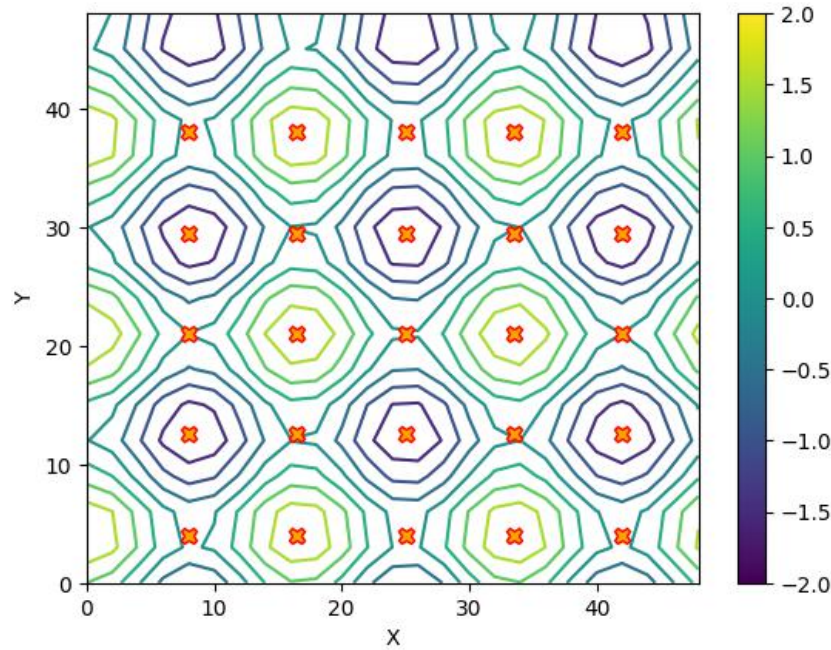


Figura 5 – Exemplo de distribuição inicial das partículas ao longo do espaço de solução. O símbolo “x” em vermelho representa as partículas no espaço de busca (Fonte: Elaboração Própria)

Para calcular o vetor velocidade da próxima iteração, é utilizado a Equação 2.8, que considera o vetor da melhor posição encontrada pelo enxame e seu respectivo peso  $((p_g - x_i^t), c_2)$ , o vetor da melhor posição encontrada pela própria partícula e seu respectivo peso  $((p_i - x_i^t), c_1)$  e a inércia do vetor velocidade do estado atual com seu respectivo peso  $(v_i^t, w)$ . Na Figura 6, temos um diagrama que mostra o fluxo de execução do PSO.

$$v_i^{t+1} = w * v_i^t + c_1 r_1^t (p_i - x_i^t) + c_2 r_2^t (p_g - x_i^t). \quad (2.8)$$

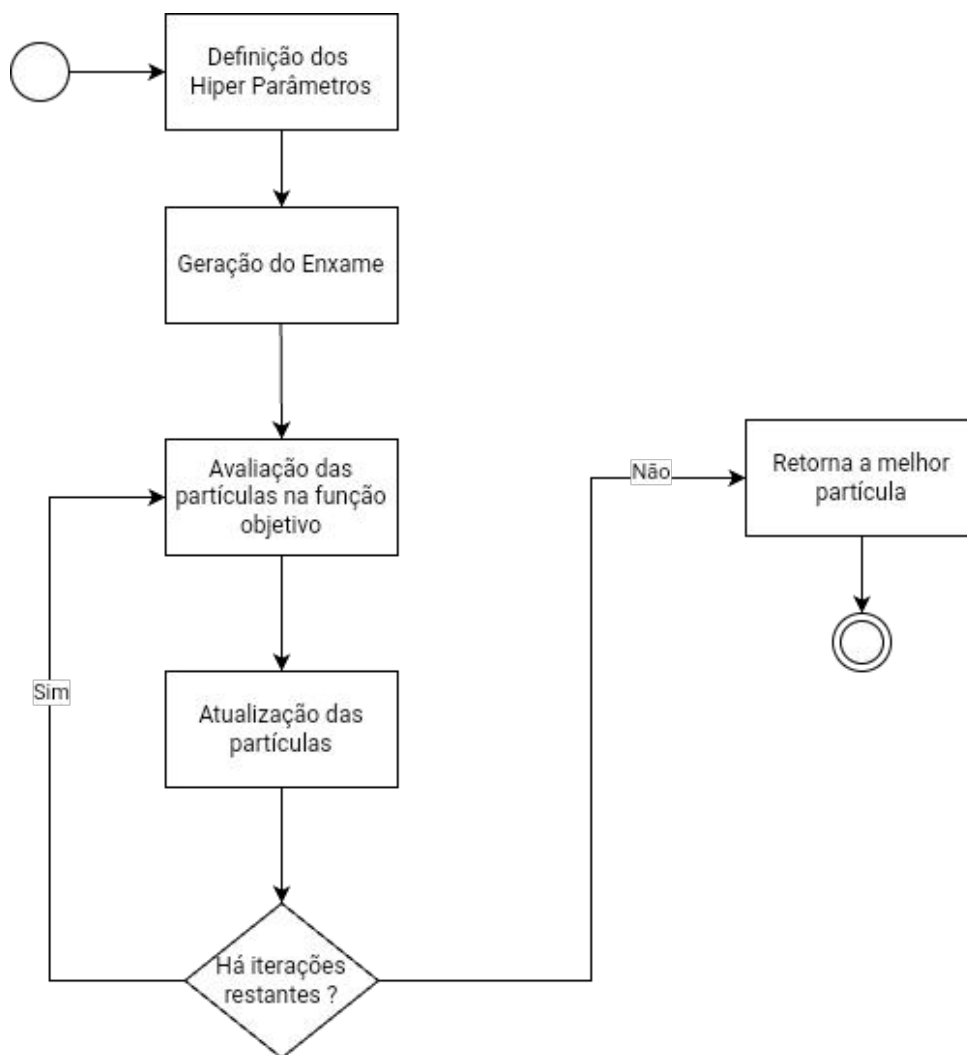


Figura 6 – Diagrama do fluxo de execução do PSO

## 2.3 COMPUTAÇÃO PARALELA

A Computação Paralela é um campo que busca o ganho de desempenho na execução de software através da utilização de mais de um núcleo de processamento — CPU e/ou GPU — ou distribuindo o processamento entre mais máquinas para que duas ou mais rotinas de *software* sejam executadas simultaneamente (PACHECO, 2011).

### 2.3.1 Paralelismo em Aprendizado de Máquina

A utilização de estratégias de paralelismo para modelos de aprendizagem de máquina surge com a intenção de viabilizar modelos robustos ao torná-los mais eficientes. Usualmente, o processo mais custoso em um modelo de aprendizado de máquina é o treinamento do modelo. Ao utilizar um modelo acoplado a um algoritmo de otimização baseado em inteligência de enxame, temos que a atualização da posição das partículas é um processo custoso. Diante disso, o particionamento do enxame entre os núcleos da CPU para realizar

a atualização do enxame em paralelo se torna uma estratégia para minimizar a perda de desempenho que essa categoria de meta-heurística possui. Na Figura 7, podemos ver como a atualização do enxame seria realizada. Cada subconjunto de partículas do enxame seria avaliada em um dos núcleos físicos do processador para determinar o erro de classificação e, em função das partículas avaliadas, seria possível identificar o próximo estado de cada uma delas.

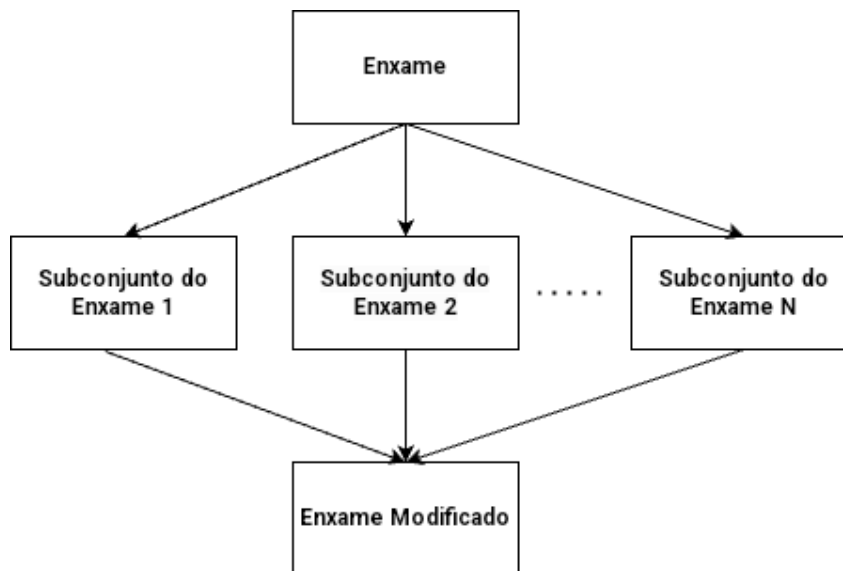


Figura 7 – Atualização em paralelo do enxame. (Fonte: Elaboração Própria)

### 2.3.1.1 Experimento preliminar de PSO sequencial vs. paralelo

A implementação do PSO utilizada paraleliza a avaliação de cada partícula na função objetivo de tal forma que um subconjunto de partículas será alocado a um dos núcleos físicos do processador. Para fins de avaliar o comportamento dos enxames a função de teste “Esférica” foi escolhida e é definida pela Equação 2.9,

$$f(x) = \sum_{i=1}^d x_i^2 \quad (2.9)$$

A Função Esférica é unimodal em 2D e apresenta a paisagem mostrada pela Figura<sup>2</sup> 8. O PSO foi executado com 10 partículas durante 100 iterações com os valores default da biblioteca *pyswarms* do Python. Na Figura 9 podemos observar que dado um espaço de solução e o enxame de partículas, onde as partículas são representadas por "x", teremos subconjuntos de partículas que serão avaliados em um mesmo núcleo de processamento.

Em uma verificação mais detalhada, os enxames foram capturados exatamente aos 20%, 60% e 100% do número de iterações utilizadas, tanto na versão sequencial quanto

<sup>2</sup> Fonte: <https://www.sfu.ca/ssurjano/spheref.html>

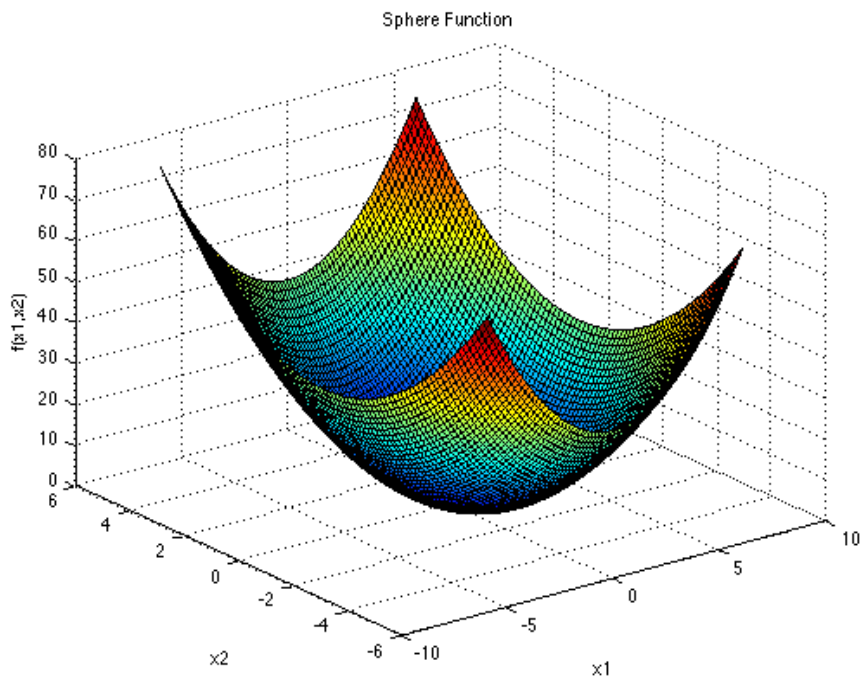


Figura 8 – Gráfico tridimensional da função *Sphere*

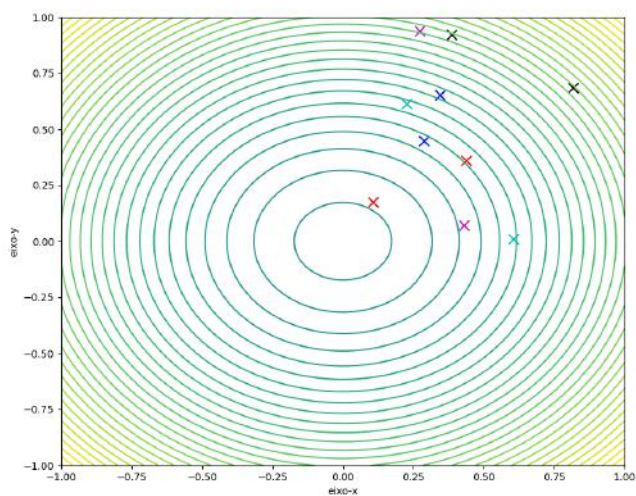


Figura 9 – Gráfico de curva de nível, onde cada 'x' colorido representa um conjunto de partículas que utilizará um dos núcleos de processamento.

na versão paralela. As Figuras 10 e 11 mostram que em termos gerais, a versão paralela se mostra similar a sequencial em termos de comportamento de distribuição e espaço em que as partículas se movem no espaço de busca. Assim, fica entendido que a versão paralela não acarreta prejuízos no decorrer da avaliação de função feita pelo PSO. É possível notar que as partículas se movimentam no espaço e que gradativamente vão se encaminhando

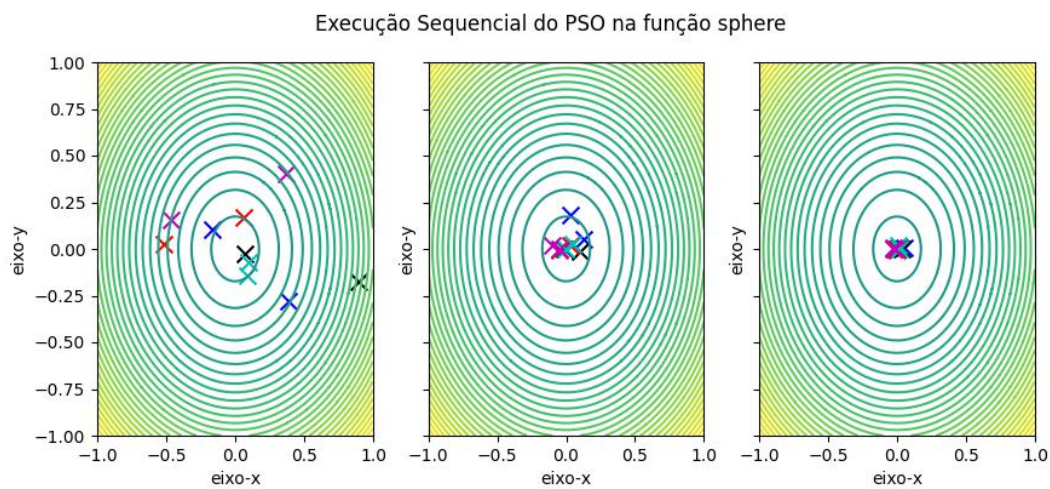


Figura 10 – Gráfico de curva de nível da função *Sphere*, onde cada partícula é representada por um "x" colorido

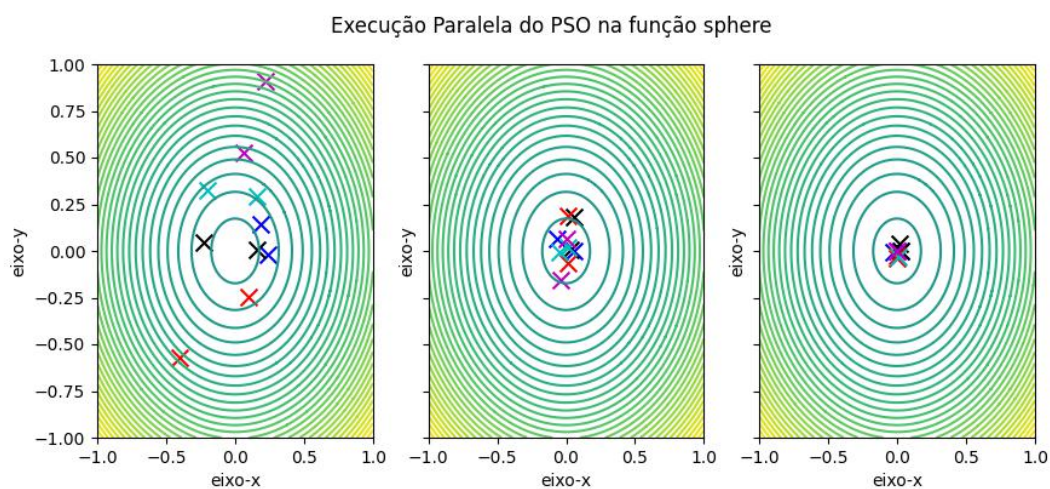


Figura 11 – Gráfico de curva de nível da função *Sphere*, onde cada partícula é representada por um "x" colorido

a uma região mais promissora perto ao ponto de ótimo do problema.

### 3 PROPOSTA

Neste capítulo, será abordado como foi implementada a solução proposta neste trabalho, quais as vantagens e desvantagens associadas.

#### 3.1 MODELO DE CLASSIFICAÇÃO BASEADO NO NCA

O NCA, por ser uma modelo que aprende uma transformação linear a fim de minimizar as distâncias entre observações de mesma classe, precisa ser complementado por uma estratégia de classificação também baseado em vizinhanças. Ao considerar essa estratégia, passamos a considerar a quantidade de vizinhos mais próximos que serão utilizados para classificar um determinado dado de teste.

Utilizando o paradigma de orientação a objetos, foi desenvolvido uma classe nomeada NCA-C que representasse o modelo NCA, e que pudesse receber como atributos: a função de otimização que será usada pelo NCA para encontrar a transformação linear mais adequada para o conjunto de dados de treinamento; a quantidade máxima de iterações que será realizada pela função de otimização; a quantidade de vizinhos que será considerada para realizar a classificação, e a quantidade de núcleos do processador que serão utilizados para realizar o treinamento. Devido à implementação ter sido feito utilizando a linguagem de programação Python, os algoritmos de otimização que forem realizar paralelismo precisam utilizar multiprocessamento na execução ao invés de *multithread* para evitar um aumento no tempo de execução. Esse fenômeno é ocasionado devido à presença do Global Interpreter Locker (GIL) no interpretador do Python (MEIER; GROSS, 2019).

---

#### Algoritmo 1 treinamento e classificação do NCA-C

---

**Precisa:** Dados de treino  $X_{tr} = \{x_{tr0}, x_{tr1}, \dots\}$ ; classificações de treino  $Y_{tr} = \{y_{tr0}, y_{tr1}, \dots\}$ ; dados de teste  $X_{te} = \{x_{te0}, x_{te1}, \dots\}$ ; algoritmo de otimização  $o$

**Garante:** Classificações dos dados de teste  $Y_{te} = \{y_{te0}, y_{te1}, \dots\}$

$Y_{te} \leftarrow \{\}$

Inicializa  $A$  como uma matriz identidade

Minimiza o erro de  $f(A)$  usando  $o$  em  $X_{tr}$  e  $Y_{tr}$

$AX_{tr} \leftarrow A \cdot X_{tr}$

**para**  $x_{te} \leftarrow X_{te}$  **até**  $X_{te} = \emptyset$  **faça**

$Ax_{te} \leftarrow A \cdot x_{te}$

**para**  $Ax_{tr} \leftarrow X_{tr}$  **até**  $X_{tr} = \emptyset$  **faça**

        Calcula  $p_i$

**fim para**

$y_{te} \leftarrow \operatorname{argmax} p_i$

$Y_{te} \leftarrow Y_{te} + \{y_{te}\}$

**fim para**

**devolve**  $Y_{te}$

---

## 3.2 NCA + PSO

O modelo classificador proposto é a utilização do modelo NCA, com métodos adicionais que viabilizam a classificação, e a utilização do PSO como algoritmo de otimização. A utilização do PSO não nos garante encontrar o erro de mínimo global, além de demandar mais tempo que uma estratégia de otimização baseada em gradiente ao utilizar a mesma quantidade de iterações. Contudo, ao adotar o PSO, teríamos um espaço de busca maior em função da quantidade de partículas utilizadas. E, ao termos a transformação linear, seria possível compreender como o espaço de solução será transformado e compreender como essa distorção poderá impactar na classificação.



## 4 EXPERIMENTOS REALIZADOS

Neste capítulo será abordado o escopo do plano experimental, metas estabelecidas, perguntas de pesquisa, hipóteses elaboradas, as métricas de interesse e análise dos resultados encontrados. Todos os experimentos realizados nessa seção foram conduzidos utilizando o seguinte ambiente experimental:

- Processador: 13th Gen Intel<sup>®</sup> Core<sup>™</sup> i7-13700K x16
- Memória RAM: 32 GB
- SSD: 500 GB
- Sistema Operacional: Linux Mint
- Os experimentos foram executados<sup>1</sup> em:
  - Python (3.10.12)
  - scikit-learn (1.3.2)
  - scipy (1.11.3)
  - numpy (1.26.1)
  - pyswarms (1.3.0)

Ao decorrer deste trabalho, foi utilizado um conjunto de dados simulados de uma *Smart Grid* com um produtor e 3 consumidores gerado por (SCHÄFER et al., 2016) e disponibilizado em (ARZAMASOV, 2018). Na Figura 12 temos a topologia da *Smart Grid* que será utilizada.

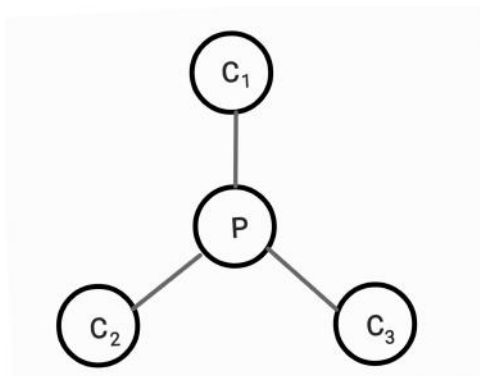


Figura 12 – Topologia da *Smart Grid* tratada. (Fonte: Elaboração Própria)

---

<sup>1</sup> Código Fonte pode ser encontrado em: <https://github.com/sidneyouteiro/TCC>

Dentro deste conjunto de dados, que originalmente possui 10000 observações com atributos descritos na Tabela 2, serão utilizadas distintas amostragens probabilísticas visando manter a mesma proporção de classes do conjunto de dados original. Essa etapa contribuirá para a redução do custo, do tempo de treinamento e tempo de validação dos modelos com perda de desempenho mínimo (ROBB, 1963) (ELRAFEY; WOJTUSIAK, 2017). Dos atributos presentes, será desconsiderado o atributo *'stab'*, pois o atributo objetivo *'stabf'* é extraído a partir do atributo *'stab'*.

Atributo	Descrição
tau1	tempo de reação do produtor de energia
p1	energia nominal consumida / produzida do produtor de energia
g1	coeficiente proporcional a elasticidade do preço do produtor de energia
tau2	tempo de reação do consumidor 1
p2	energia nominal consumida / produzida do consumidor 1
g2	coeficiente proporcional a elasticidade do preço do consumidor 1
tau3	tempo de reação do consumidor 2
p3	energia nominal consumida / produzida do consumidor 2
g3	coeficiente proporcional a elasticidade do preço do consumidor 2
tau4	tempo de reação do consumidor 3
p4	energia nominal consumida / produzida do consumidor 3
g4	coeficiente proporcional a elasticidade do preço do consumidor 3
stab	A parte real máxima da raiz da equação característica
stabf	Classificação da estabilidade econômica do sistema

Tabela 2 – Atributos presentes no conjunto de dados

#### 4.1 AMOSTRAGEM DOS DADOS E TAMANHO DO ENXAME

Neste experimento, temos como objetivo identificar, dado uma amostragem probabilística, qual o tamanho da amostra ideal em conjunto com o tamanho do enxame. Para assim, determinar qual configuração irá obter o melhor desempenho.

##### 4.1.1 Plano Experimental

Neste estudo experimental será utilizado o modelo proposto, NCA+PSO, para avaliar como cada configuração irá desempenhar. As variáveis dependentes, ou seja, as variáveis que irão assumir valores distintos para entender como elas afetam as métricas, que serão utilizadas nesse estudo experimental serão:

- **O tamanho da amostra**, de tal forma que as proporções entre as classes sejam mantidas.
- **O tamanho do enxame** de partículas.

As variáveis independentes, e seus respectivos valores fixados, que serão utilizadas nesse estudo experimental:

- **A quantidade de gerações** que o enxame irá realizar a busca será de 800 gerações, determinado arbitrariamente.
- Para realizar uma classificação, **a quantidade de vizinhos** que será utilizada é 50 vizinhos, determinado arbitrariamente.
- **A quantidade de núcleos de CPU** que serão utilizados para avaliar o enxame de partículas serão de 16 núcleos físicos da CPU, determinado para reduzir o tempo de execução do experimento.
- **Os pesos dos vetores** de inércia, melhor posição da partícula e melhor posição do enxame possuirão, respectivamente, os seguintes pesos 0.9, 0.5 e 0.3, escolhidos arbitrariamente.

As métricas que serão utilizadas para comparar as combinações serão a acurácia (ACC), a *Matthew Correlation Coefficient* (MCC) (CHICCO; JURMAN, 2020) e o tempo de execução do modelo em segundos. Será inspecionado a matriz de confusão da configuração que obtiver o melhor desempenho. Em cenários que houver configurações com desempenhos similares, a configuração avaliada no menor tempo será tida como a melhor.

Nas equações 4.1 e 4.2 temos como é calculada as métricas ACC e MCC, em função da classificação gerada pelo modelo e da classificação real da observação. Tome  $TN$  e  $TP$  como classificações corretas respectivamente das classes estável e instável. E  $FN$  e  $FP$  como classificações incorretas respectivamente das classes estável e instável.

$$ACC = \frac{TN + TP}{TN + TP + FN + FP}. \quad (4.1)$$

$$MCC = \frac{TN * TP - FN * FP}{\sqrt{(TN + FN)(TN + FP)(TP + FN)(TP + FP)}}. \quad (4.2)$$

Diante disso, serão realizadas, para cada avaliação de configuração, uma validação cruzada utilizando o 5-Fold (ABU-MOSTAFA; MAGDON-ISMAIL; LIN, 2012), onde cada configuração será avaliada 5 vezes.

#### 4.1.2 Análise dos Resultados Encontrados

A partir do experimento foi possível obter os seguintes dados, que estão contidos na Tabela 3, em função de cada configuração do experimento. Diante dos resultados encontrados, foi possível observar que os melhores resultados da média de acurácia com os menores desvios foram encontrados ao realizar a amostragem das observações com 1500 observações. Ao analisar a métrica MCC que varia de -1 a 1, onde o resultado -1 indica uma classificação imperfeita e 1 indica uma classificação perfeita. Logos, temos que os

Tamanho do Enxame	Tamanho do dataset	ACC		MCC		Tempo (s)	
		Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
50	500	0.681	0.222	0.071	0.146	223.014	9.801
	1000	0.698	0.189	0.059	0.121	656.603	26.675
	<b>1500</b>	<b>0.737</b>	<b>0.175</b>	<b>0.069</b>	<b>0.141</b>	<b>1011.841</b>	<b>13.347</b>
	2000	0.705	0.194	0.069	0.141	1270.594	13.127
100	500	0.705	0.189	0.077	0.159	482.988	32.277
	1000	0.697	0.185	0.059	0.121	1534.887	71.532
	<b>1500</b>	<b>0.733</b>	<b>0.182</b>	<b>0.071</b>	<b>0.145</b>	<b>2227.226</b>	<b>37.365</b>
	2000	0.716	0.190	0.067	0.138	2763.338	36.012
150	500	0.722	0.179	0.077	0.159	739.355	45.825
	1000	0.687	0.198	0.057	0.117	2494.952	64.033
	<b>1500</b>	<b>0.732</b>	<b>0.186</b>	<b>0.070</b>	<b>0.143</b>	<b>3494.799</b>	<b>81.787</b>
	2000	0.711	0.196	0.069	0.141	4301.908	58.635
200	500	0.719	0.184	0.082	0.168	1056.841	61.203
	1000	0.694	0.193	0.055	0.114	3576.936	79.133
	<b>1500</b>	<b>0.739</b>	<b>0.172</b>	<b>0.070</b>	<b>0.144</b>	<b>5012.063</b>	<b>60.758</b>
	2000	0.712	0.196	0.068	0.140	6161.141	72.736
250	500	0.702	0.189	0.072	0.148	1412.374	72.639
	1000	0.694	0.197	0.056	0.115	4582.585	89.739
	<b>1500</b>	<b>0.728</b>	<b>0.191</b>	<b>0.064</b>	<b>0.131</b>	<b>6371.441</b>	<b>86.696</b>
	2000	0.707	0.204	0.065	0.134	7827.318	49.981
300	500	0.715	0.188	0.082	0.168	1768.570	71.49
	1000	0.694	0.202	0.059	0.121	5522.356	86.117
	<b>1500</b>	<b>0.741</b>	<b>0.180</b>	<b>0.070</b>	<b>0.143</b>	<b>7656.438</b>	<b>49.807</b>
	2000	0.706	0.193	0.070	0.142	9527.781	67.647

Tabela 3 – Tabela dos resultados encontrados do experimento.

resultados encontrados no experimento indicam as configurações resultam em classificações aleatórias. Esta é uma informação importante. Indica que a acurácia encontrada é instável. Assim, não se pode garantir que o modelo de fato possa ser usado em um cenário real. No entanto, para fins de estudo, este resultado nos mostra uma boa possibilidade de continuidade e averiguação de como se pode resolver melhor o problema. A avaliação estatística se segue com o intuito de verificar possíveis vantagens da proposta, mesmo sabendo que a acurácia encontrada pode ter sido obtida ao acaso. Devido ao desvio padrão ser muito pequeno e pouco diferente, foi elaborado um gráfico de Boxplot. Este é um teste não-paramétrico visual que auxilia na tomada de decisão. Boxplots não são úteis apenas para analisar o alcance e a distribuição dos dados, mas, às vezes, podem fornecer informações sobre a verdadeira diferença entre as médias.

Caso as caixas nos Boxplots apresentem sobreposição, pode-se concluir, com 95% de confiança, que as médias não se diferem (MARCELINO et al., 2021). Tendo isso em mente e observando a Figura 13, é possível concluir que:

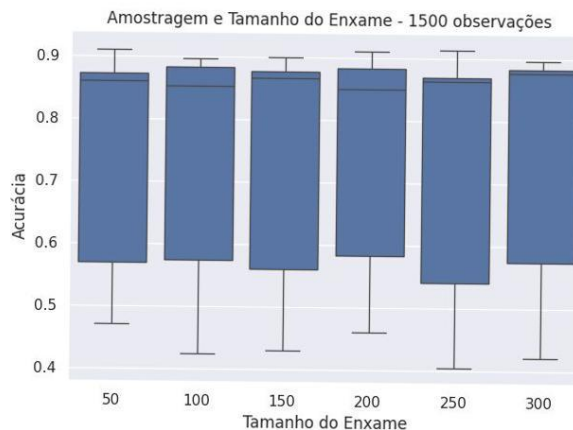


Figura 13 – Boxplot da média das acurácias obtidas para amostragem de 1500 observações.

- As caixas apresentadas pelos conjuntos amostrais da variação da população do PSO são sobrepostas, e;
- Logo, é possível concluir que não existem diferenças estatísticas significantes entre as médias das amostras apresentadas ao utilizarmos 1500 observações.

Uma vez que o teste estatístico mostrou que o tamanho da população não interfere na resposta, o tempo de execução foi analisado. Quando o PSO usa uma população de 50 partículas, a classificação realizada de detecção de instabilidade atinge 73,7% de acurácia em tempo de 1011.841s, com baixo desvio padrão frente as demais populações testadas para o PSO. Na Figura 14, temos a matriz de confusão ao utilizarmos 50 partículas em um conjunto de dados de 1500 observações. Apesar do desempenho, o percentual de falsos positivos é preocupante devido possibilidade de gerar fadiga de alerta (POLY et al., 2020). Uma hipótese que explique o motivo de existir uma quantidade de falsos positivos superior a de verdadeiros positivos, seria devido às classes do conjunto de dados estarem desbalanceadas.

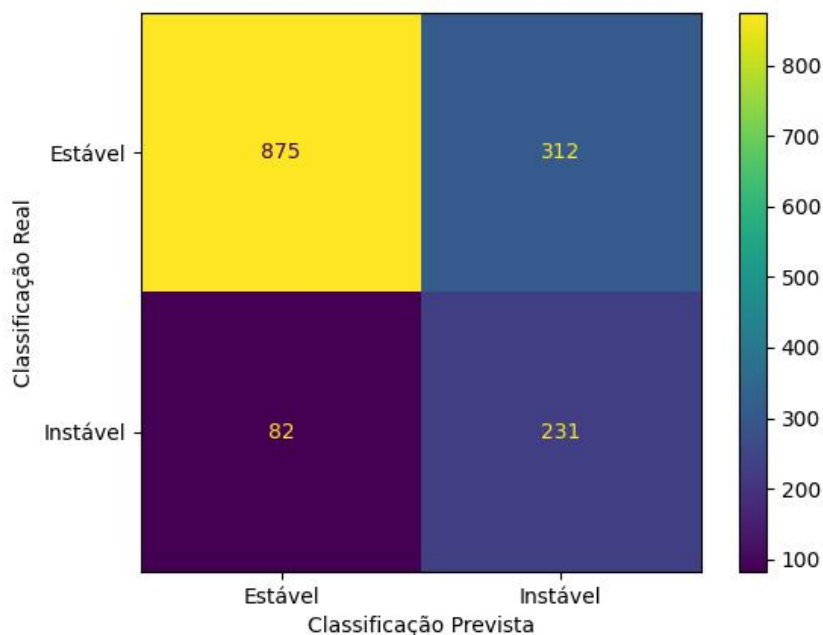


Figura 14 – Matriz de Confusão da melhor configuração encontrada.

## 4.2 AVALIAÇÃO DO TEMPO DE EXECUÇÃO

Este experimento visa avaliar como o paralelismo impacta na métrica de tempo de execução. Será realizada uma comparação do tempo decorrido de processamento ao utilizar execução sequencial e multiprocessamento.

### 4.2.1 Plano Experimental

Neste plano experimental será utilizado duas variações do modelo proposto. Uma das variações utilizará 16 núcleos físicos do processador, enquanto a outra variação realizará execução sequencial. As variáveis dependentes que serão utilizadas nesse estudo experimental:

- **Quantidade de núcleos de processamento** que serão utilizados para realizar o treinamento

As variáveis independentes e seus respectivos valores fixados, que serão utilizados nesse estudo experimental:

- **A quantidade de partículas** do enxame será de 50 partículas.
- **A quantidade de observações** presente na amostragem do dataset será de 1500 observações.

- **A quantidade de gerações** que o enxame utilizará para realizar a busca será de 800 gerações.
- **Os pesos dos vetores** de inércia, melhor posição da partícula e melhor posição do enxame possuirão, respectivamente, os seguintes pesos 0.9, 0.5 e 0.3.

Diante disso, foi realizado, para cada variação do modelo, uma validação cruzada utilizando 5-Fold (ABU-MOSTAFA; MAGDON-ISMAIL; LIN, 2012), onde cada variação foi executada 5 vezes. A partir dos resultados, será avaliada se não há diferença estatística entre as métricas propostas no experimento anterior.

#### 4.2.2 Análise dos Resultados Encontrados

Na Tabela 4 podemos observar os resultados encontrados pelo experimento, dada as métricas consideradas no experimento anterior. Os resultados encontrados mostram que número de núcleos de processamento não impacta na avaliação das métricas, acurácia e MCC, e que houve uma melhora na eficiência ao avaliar a média e desvio padrão do tempo de execução. Diante disso, foi realizado o teste não-paramétrico de *Mann-Whitney U* (MANN; WHITNEY, 1947) para aferir se não há diferença estatística entre as métricas ACC e MCC ao variar a quantidade de unidades de processamento. Nesse caso, a hipótese nula do teste é que uma variável aleatória é estocasticamente igual a outra, e a hipótese alternativa é que uma variável aleatória é estocasticamente maior do que a outra. Ao executar o teste de Mann-Whitney U, foi encontrado o *p-value* de 0,7049 ao avaliar a ACC e *p-value* de 0,8458 ao avaliar o MCC. Com isso, para obter uma conclusão com 95% de confiança, é necessário que alfa assuma o valor de 0,05. Dito isso, não foi possível rejeitar a hipótese nula para as métricas ACC e MCC, portanto, temos que não há diferença estatística significativa entre a ACC e MCC da execução sequencial e a execução que utilizou 16 núcleos.

Modelo	ACC		MCC		Tempo(s)	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
NCA+PSO Paralelizado	0.736	0.175	0.068	0.140	1011.840	13.346
NCA+PSO Sequencial	0.732	0.185	0.063	0.129	1276.434	32.157

Tabela 4 – Resultados da execução paralela e sequencial do modelo proposto.

Restando apenas o tempo de execução para avaliar, foi elaborado um gráfico Boxplot para executar o mesmo teste realizado no experimento anterior. Na Figura 15, podemos observar o gráfico gerado e, diante disso, podemos concluir:

- As caixas apresentadas pelos conjuntos amostrais da variação de unidades de processamento não são sobrepostas, e;

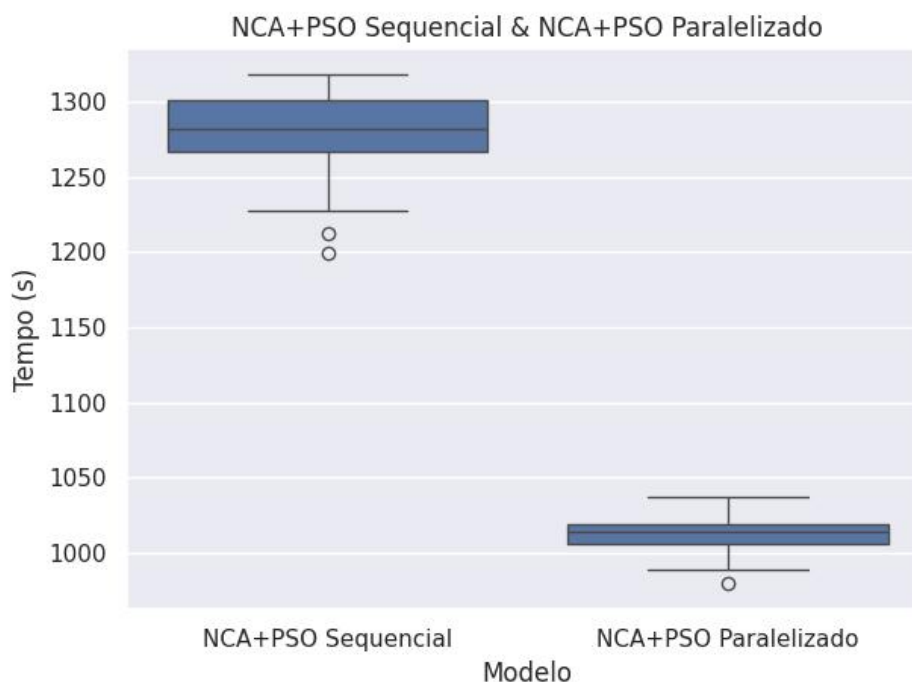


Figura 15 – Boxplot do tempo de execução do modelo sequencial e modelo paralelizado.

- Logo, é possível concluir que existem diferenças estatísticas significantes entre as médias *de tempo de execução* das amostras apresentadas.

Os testes estatísticos mostraram que a variação entre a quantidade de núcleos de processamento não interfere nas métricas de ACC e MCC. Contudo, com o aumento de núcleos de processamento foi possível uma redução significativa no tempo de execução médio.

### 4.3 MODELOS BASEADOS EM VIZINHANÇA

Neste experimento, temos como objetivo comparar modelos baseados em vizinhança. Para assim determinar qual modelo irá obter o melhor desempenho.

#### 4.3.1 Plano Experimental

Este estudo experimental utilizará os seguintes modelos de aprendizado de máquina: NCA acoplado ao PSO, NCA acoplado a um algoritmo de otimização baseada no gradiente e o KNN. Os modelos que foram utilizados nesse experimento, foram selecionados para comparar como o algoritmo de otimização impacta nas métricas de interesse e utilizar os resultados do KNN, um modelo de classificação de vizinhanças mais consolidado, como referencial. Será utilizado como tratamento os modelos apresentados anteriormente. Nos modelos baseados no NCA, será utilizado a mesma quantidade de iterações. E em todos



os modelos será utilizado a mesma quantidade de vizinhos para realizar a classificação. Diante disso, será realizado, para cada modelo, uma validação cruzada, utilizando 5-Fold e cada configuração será avaliada 30 vezes. A partir dos resultados será avaliado se há diferença estatística entre as métricas propostas.

### 4.3.2 Análise dos Resultados Encontrados

Na Tabela 5, podemos observar os resultados encontrados pelo experimento, dada as métricas consideradas ao longo deste capítulo. Utilizando o teste de Mann-Whitney U foi possível encontrar  $p$ -value de 0,0002 ao avaliar as acurácias do modelo NCA+PSO com o NCA+Gradiente, com isso, refutando a hipótese nula. Contudo, ao avaliar o MCC do NCA+PSO e NCA+Gradiente foi encontrado o  $p$ -value de 0,5678, o que impossibilita refutar a hipótese nula.

Model	ACC		MCC		Tempo(s)	
	Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
KNN	0.6293	0.3825	0.0346	0.0694	0.0100	0.0003
NCA+Gradient	0.7160	0.2605	0.0634	0.1272	2.0102	0.1888
NCA+PSO Paralelo	<b>0.7346</b>	0.1770	<b>0.0674</b>	0.1358	1030.8897	15.9050

Tabela 5 – Resultados encontrados ao comparar modelos baseados em vizinhança.

A partir dos resultados encontrados, o acoplamento de um modelo baseado no NCA ao PSO mostrou-se competitivo nas métricas ACC e MCC, o que pode ser observado também na Figura 16. Apesar de não obter um tempo de execução tão baixo quanto o KNN e o NCA com o gradiente, é preciso considerar que as implementações destes modelos foram extraídas e utilizadas da biblioteca *scikit-learn*. O NCA é uma implementação independente construída neste trabalho, que usa bibliotecas prontas, porém sua junção é fruto deste trabalho. Um fator que indique o potencial de melhoria do acoplamento do NCA com o PSO é a realização futura de um ajuste fino nos hiper parâmetros do PSO. No escopo deste trabalho, devido ao Global Interpreter Locker (GIL), mecanismo presente no interpretador do Python, não foi possível paralelizar em função das *threads* processador, assim realizando o paralelismo em função de processos alocados nos núcleos físicos do processador. Este foi um impeditivo para, neste trabalho, encontrar soluções que minimizassem o tempo de execução da proposta NCA. Diante disso, pode-se concluir que o acoplamento de meta-heurísticas populacionais a modelos de aprendizado de máquina tem potencial de ser uma alternativa ao uso padrão de algoritmos de otimização baseados no gradiente.

Apesar dos resultados apresentados entre as variações do NCA terem obtido, em média, desempenhos de acurácia próximos é importante ressaltar o potencial do acoplamento do

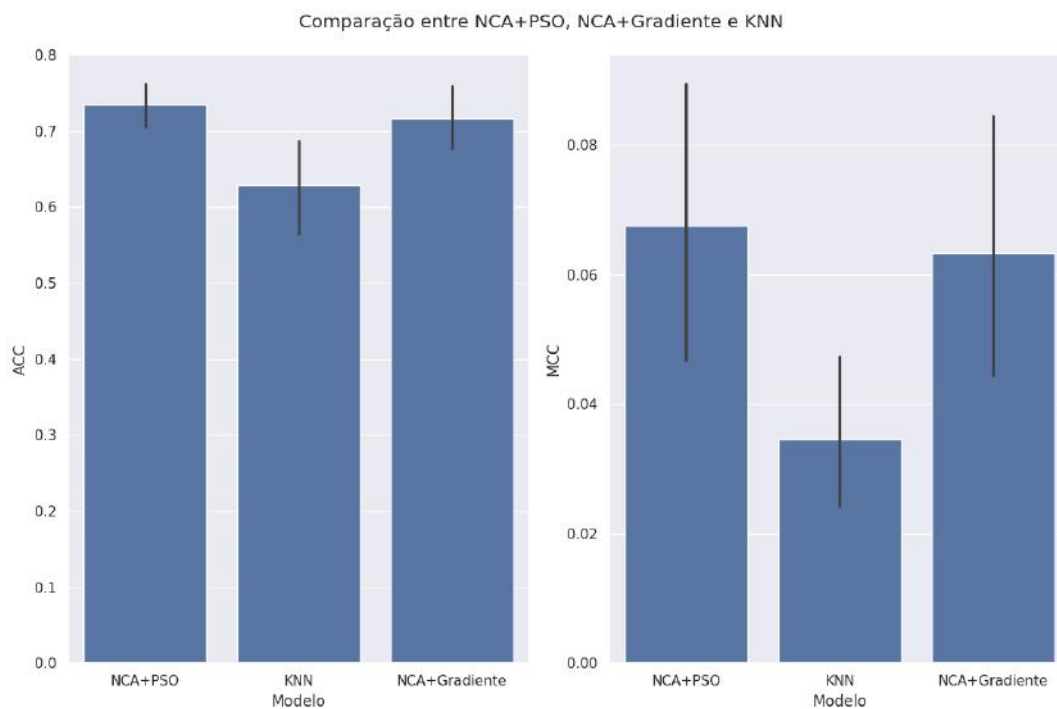


Figura 16 – Resultados encontrados ao avaliar modelos baseados em vizinhança. A linha, presente em cada uma das barras, representa o intervalo de confiança de 95%.

NCA com o PSO neste critério. O NCA+PSO se mostrou no plano experimental uma solução promissora estatisticamente se diferenciando das demais. Um ponto de atenção observável na matriz de confusão, da melhor configuração, foi a presença de classificações de estabilidade incorretas. A redução de casos de falsos negativos é necessária, pois os mesmos podem ocasionar situações de superprodução ou subprodução de energia e, conseqüentemente, impactar na viabilidade econômica da *Smart Grid* e disponibilidade de energia para os consumidores finais.

## 5 CONCLUSÃO

No decorrer deste trabalho, foi proposto a caracterização de um modelo de aprendizado acoplado a um algoritmo de otimização populacional. O cenário utilizado para realizar a caracterização foi um dos desafios enfrentados pela nova geração de malhas elétricas: a detecção de estabilidade econômica de uma *Smart Grid*. Ao endereçar essa demanda, é possível viabilizar a adoção deste novo modelo de malhas elétricas que pode nos auxiliar na adoção de fontes energéticas mais sustentáveis.

A escolha do NCA como o modelo a ser acoplado ao PSO se deu pela facilidade em gerar uma implementação que possibilitasse o acoplamento sem que fosse necessário realizar mudanças estruturais. Ao longo deste trabalho foi observado que o alto nível de abstração da linguagem Python e de suas bibliotecas foram importantes na concepção de uma prova de conceito. Isso porque ela possibilita avaliar os requisitos necessários para implementar o modelo proposto de uma maneira mais eficiente. Contudo, para ser possível caracterizar o comportamento de algoritmos populacionais e aproveitar melhor os recursos computacionais, será necessário a utilização de uma linguagem de programação com nível de abstração menor. Assim, possibilitaria um maior controle acerca dos processos que compõem o treinamento do modelo e permitiria o paralelismo real.

Diante dos experimentos executados, conclui-se que a utilização de paralelismo viabiliza o investimento no estudo de acoplamento de meta-heurísticas populacionais de classificação. E os resultados encontrados ao avaliar modelos de vizinhança, estratégia de classificação adotada para reduzir o custo computacional, mostrou o potencial competitivo do modelo proposto, podendo obter um desempenho melhor do que foi encontrado nesse trabalho ao realizar um ajuste fino nos hiper-parâmetros do modelo.

Ao longo do trabalho houve limitações encontradas ao implementar a solução proposta na linguagem de programação Python. Por exemplo, mecanismos presentes nos interpretadores da linguagem como o Global Interpreter Locker (GIL) impossibilita execução paralela nos núcleos lógicos do processador — as *threads* — sem que haja perda de desempenho (MEIER; GROSS, 2019) (ZHANG et al., 2022). Um próximo passo para o trabalho é utilização de uma linguagem de programação com nível de abstração menor para implementar o modelo NCA+PSO para poder avaliar o impacto na utilização de *threads*. Como trabalhos futuros, esta monografia aponta as seguintes frentes:

- Buscar um resultado de acurácia em que o MCC encontre valores mais próximos a 1, levando a uma robustez estatística.
- Possibilitar que o modelo utilize Unidades de Processamento Gráfica (GPU), para realizar o treinamento e a classificação.

- Realizar um ajuste fino, considerando todos os hiper-parâmetros do modelo.
- Avaliar outras meta-heurísticas populacionais e bioinspiradas.

## REFERÊNCIAS

- ABU-MOSTAFA, Y. S.; MAGDON-ISMAIL, M.; LIN, H.-T. **Learning from data**. [S.l.]: AMLBook New York, 2012. v. 4.
- AMARI, S. A theory of adaptive pattern classifiers. **IEEE Transactions on Electronic Computers**, IEEE, n. 3, p. 299–307, 1967.
- ARZAMASOV, V. **Electrical Grid Stability Simulated Data** . 2018. UCI Machine Learning Repository. DOI: <https://doi.org/10.24432/C5PG66>.
- ARZAMASOV, V.; BÖHM, K.; JOCHEM, P. Towards concise models of grid stability. In: IEEE. **2018 IEEE international conference on communications, control, and computing technologies for smart grids (SmartGridComm)**. [S.l.], 2018. p. 1–6.
- CHICCO, D.; JURMAN, G. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. **BMC genomics**, BioMed Central, v. 21, n. 1, p. 1–13, 2020.
- COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE transactions on information theory**, IEEE, v. 13, n. 1, p. 21–27, 1967.
- DORIGO, M.; BIRATTARI, M.; STUTZLE, T. Ant colony optimization. **IEEE computational intelligence magazine**, IEEE, v. 1, n. 4, p. 28–39, 2006.
- ELRAFEY, A.; WOJTUSIAK, J. Recent advances in scaling-down sampling methods in machine learning. **Wiley Interdisciplinary Reviews: Computational Statistics**, Wiley Online Library, v. 9, n. 6, p. e1414, 2017.
- FANG, X. et al. Smart grid—the new and improved power grid: A survey. **IEEE communications surveys & tutorials**, Ieee, v. 14, n. 4, p. 944–980, 2011.
- FARHANGI, H. The path of the smart grid. **IEEE power and energy magazine**, IEEE, v. 8, n. 1, p. 18–28, 2009.
- FIX, E.; HODGES, J. L. Discriminatory analysis. nonparametric discrimination: Consistency properties. **International Statistical Review/Revue Internationale de Statistique**, JSTOR, v. 57, n. 3, p. 238–247, 1989.
- GOLDBERGER, J. et al. Neighbourhood components analysis. **Advances in neural information processing systems**, v. 17, 2004.
- GURVITS, L.; OLSHEVSKY, A. On the np-hardness of checking matrix polytope stability and continuous-time switching stability. **IEEE Transactions on Automatic Control**, IEEE, v. 54, n. 2, p. 337–341, 2009.
- KARABOGA, D. et al. **An idea based on honey bee swarm for numerical optimization**. [S.l.], 2005.

- KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE. **Proceedings of ICNN'95-international conference on neural networks**. [S.l.], 1995. v. 4, p. 1942–1948.
- MANN, H. B.; WHITNEY, D. R. On a test of whether one of two random variables is stochastically larger than the other. **The annals of mathematical statistics**, JSTOR, p. 50–60, 1947.
- MARCELINO, C. G. et al. An efficient multi-objective evolutionary approach for solving the operation of multi-reservoir system scheduling in hydro-power plants. **Expert Systems with Applications**, Elsevier, v. 185, p. 115638, 2021.
- MEIER, R.; GROSS, T. R. Reflections on the compatibility, performance, and scalability of parallel python. In: **Proceedings of the 15th ACM SIGPLAN international symposium on dynamic languages**. [S.l.: s.n.], 2019. p. 91–103.
- ONU. **Objetivo de Desenvolvimento Sustentável 7: Energia limpa e acessível — brasil.un.org**. 2015. <https://brasil.un.org/pt-br/sdgs/7>. [Acessado em 01/12/2023].
- PACHECO, P. **An introduction to parallel programming**. [S.l.]: Elsevier, 2011.
- POLY, T. N. et al. Machine learning approach to reduce alert fatigue using a disease medication–related clinical decision support system: model development and validation. **JMIR medical informatics**, JMIR Publications Toronto, Canada, v. 8, n. 11, p. e19489, 2020.
- RH, C. T. **Notas de Aula: Otimização Escalar e Vetorial**. [S.l.]: Universidade Federal de Minas Gerais,[Online]. <http://www.mat.ufmg.br/~taka>, 2005.
- RIBEIRO, A. A.; KARAS, E. W. Otimização continua: aspectos teóricos e computacionais. **Cengage Learning, Sao Paulo**, 2013.
- ROBB, R. w. g. cochran, sampling techniques (john wiley & sons, 1963), ix+ 413 pp., 72s. **Proceedings of the Edinburgh Mathematical Society**, Cambridge University Press, v. 13, n. 4, p. 342–343, 1963.
- SCHÄFER, B. et al. Taming instabilities in power grid networks by decentralized control. **The European Physical Journal Special Topics**, Springer, v. 225, p. 569–582, 2016.
- SCHÄFER, B. et al. Decentral smart grid control. **New journal of physics**, IOP Publishing, v. 17, n. 1, p. 015002, 2015.
- ZHANG, Q. et al. Quantifying the interpretation overhead of python. **Science of Computer Programming**, Elsevier, v. 215, p. 102759, 2022.

## APÊNDICES

## APÊNDICE A – CÓDIGO DA CLASSE NCA-C

Código 1 – classe NCA-C

```

import time
import numpy as np
import pandas as pd

from scipy.spatial.distance import euclidean

class NCA_C:
    def __init__(self, optimization_func=None,
                 max_iter=50, k=5, cpu_count=None,
                 swarm_size=10):
        self.optimization_func = optimization_func
        self.max_iter = max_iter
        self.k = k
        self.cpu_count = cpu_count
        self.swarm_size = swarm_size

    def fit(self, X, y):
        t_train = time.time()
        if isinstance(X, pd.core.frame.DataFrame):
            self.accepted_columns = X.columns.copy()
            X = np.asarray(X)
        if isinstance(y, pd.Series):
            y = np.asarray(y)
        mask = y[:, np.newaxis] == y[np.newaxis, :]

        transformation = np.ravel(np.eye(X.shape[1], X.shape[1]))
        params = {
            'X': X, 'same_class_mask': mask,
            'transformation': transformation,
            'max_iter': self.max_iter,
            'cpu_count': self.cpu_count,
            'swarm_size': self.swarm_size
        }
        opt_result = self.optimization_func(**params)

```



```

self.components_ = opt_result.reshape(-1, X.shape[1])
self.X_embedded = self.transform(X)
self.y_train = y
t_train = time.time() - t_train
self.fit_time = t_train

return self

def transform(self, X):
    return np.dot(X, self.components_.T)

def _pred_t_object(self, t_object):
    k = self.k
    class_p = {i:0 for i in np.unique(self.y_train)}
    d = np.array(
        [euclidean(t_object, i) for i in self.X_embedded])
    k_neighbors = np.argsort(d, k)[:k]
    sum_d = np.sum(d[k_neighbors])
    for n_index in k_neighbors:
        class_p[self.y_train[n_index]] += sum_d / d[n_index]
    return max(class_p, key=class_p.get)

def predict(self, X_test):
    try:
        np_type = isinstance(X_test, (np.ndarray, np.generic))
        pd_type = isinstance(X_test, pd.DataFrame)
        if np_type:
            X_test_e = self.transform(X_test)
            y_predict = np.array(
                [self._pred_t_object(i) for i in X_test_e]
            )
            return y_predict
        elif pd_type:
            X_test = np.asarray(X_test)
            X_test_r = self.transform(X_test)
            y_predict = pd.Series(
                [self._pred_t_object(i) for i in X_test_e],
                name=self.accepted_columns[-1:]
            )

```

```
        return y_predict
    else:
        raise RuntimeError('runtime_error: {X_test}')

except Exception as err:
    print(f'error: {repr(err)}')
```

## APÊNDICE B – FUNÇÃO DE OTIMIZAÇÃO DO PSO

### Código 2 – Função PSO

```

def PSO(X, same_class_mask, transformation, max_iter=1000,
        cpu_count=None, swarm_size=10):
    from sklearn.metrics import pairwise_distances
    from sklearn.utils.extmath import softmax
    from numpy import fill_diagonal, dot, mean
    import pyswarms as ps
    import pickle
    import time

    global obj_func
    def obj_func(x, same_class_mask, X_train):
        results = []
        for i in x:
            x = x.reshape(-1, X_train.shape[1])
            X_embedded = dot(X_train, x.T)

            p_ij = pairwise_distances(X_embedded, squared=True)
            p_ij = softmax(-p_ij) # (n_samples, n_samples)
            fill_diagonal(p_ij, 0.0)
            masked_p_ij = p_ij * same_class_mask
            p = masked_p_ij.sum(axis=1, keepdims=True)
            loss = -1.0 * p.sum()
            results.append(loss)
        return results

    #peso dos vetores: melhor pessoal, melhor do enxame
    #e inercia
    options = {'c1': 0.5, 'c2': 0.3, 'w': 0.9}
    GBPSO_options = {
        'n_particles': swarm_size,
        'dimensions': len(transformation),
        'options': options
    }

```

```
optimizer = ps.single.GlobalBestPSO(**GBPSO_options)

_, A = optimizer.optimize(
    obj_func, max_iter, same_class_mask=same_class_mask,
    X_train=X, verbose=False, n_processes=cpu_count)

return A
```