Dissertação de Mestrado

Inatel
Instituto Nacional de Telecomunicações

Metodologia para o Desenvolvimento de Antenas Baseada em Inteligência Computacional

MARCELLO CALDANO DE MELO

JULHO / 2021

METODOLOGIA PARA O DESENVOLVIMENTO DE ANTENAS BASEADA EM INTELIGÊNCIA COMPUTACIONAL

MARCELLO CALDANO DE MELO

Dissertação apresentada ao Instituto Nacional de Telecomunicações, como parte dos requisitos para obtenção do Título de Mestre em Telecomunicações.

ORIENTADOR: Prof. Dr. Arismar Cerqueira Sodré Junior.

COORIENTADOR: Prof. Dr. Carmelo Jose Albanez Bastos Filho.

Santa Rita do Sapucaí
2021

Melo, Marcello Caldano de
M528m
Metodologia para o Desenvolvimento de Antenas Baseada em Inteligência Computacional. / Marcello Caldano de Melo. – Santa Rita do Sapucaí, 2021.
98p.

Orientador: Prof. Dr. Arismar Cerqueira Sodré Junior.
Co-orientador: Prof. Dr. Carmelo Jose Albanez Bastos Filho
Dissertação de Mestrado em Telecomunicações – Instituto Nacional de Telecomunicações – INATEL.
Inclui bibliografia e anexo.

1. Aprendizagem de Máquina 2. Desenvolvimento de Antenas 3. Inteligência Artificial 4. Inteligência Computacional 5. Otimização Multiobjetivo 6. Mestrado em Telecomunicações. I. Sodré Junior, Arismar Cerqueira. II. Bastos Filho, Carmelo Jose Albanez. III. Instituto Nacional de Telecomunicações – INATEL. IV. Título.

CDU 621.39

Ficha Catalográfica elaborada pelo Bibliotecário da Instituição
CRB6-2718 – Paulo Eduardo de Almeida

FOLHA DE APROVAÇÃO

Dissertação defendida e aprovada em _____/_____/_____, pela comissão julgadora:

Prof. Dr. Arismar Cerqueira Sodré Junior
INATEL

Prof. Dr. Carmelo Jose Albanez Bastos Filho
UPE

Prof. Dr. Felipe Augusto Pereira de Figueiredo
INATEL

Prof. Dr. João Roberto Moreira Neto
BRADAR

Coordenador do Curso de Mestrado
Prof. Dr. José Marcos Camara Brito

“Aplica teu coração ao ensino e teus ouvidos às palavras que trazem conhecimento”

Provérbios 23.12

Aos meus pais e avós,
meus primeiros grandes professores.

Agradecimentos

Agradeço a Deus por iluminar cada passo e cada decisão tomada durante minha trajetória de vida para que meu caminho chegasse até aqui. Aos meus pais e meu irmão por todo apoio, suporte, paciência e amor, que foram essenciais para que eu tivesse forças para continuar ao decorrer dos anos de mestrado. A todos os meus amigos e familiares, que torceram por mim e compartilharam comigo cada conquista. À minha namorada, Milena Machado Ferreira, pelo companheirismo, apoio, incentivo e amor.

Agradeço ao Instituto Nacional de Telecomunicações (INATEL) por conceder a oportunidade e a estrutura necessária para que eu pudesse tirar o maior proveito possível dessa experiência que começou em 2013 no curso de graduação. A todos os professores que participaram da minha formação acadêmica e me auxiliaram no meu crescimento profissional e pessoal, em especial ao Prof. Arismar Cerqueira Sodré Junior que me acompanhou e orientou no curso de Mestrado em Telecomunicações.

Agradeço a parceria com a Universidade de Pernambuco no decorrer do mestrado, em especial a Pedro dos Santos, Everaldo Faustino e ao Professor Carmelo J. A. Bastos Filho. Tal parceria foi essencial para minha aprendizagem técnica e capacidade de escrita, além de ter rendido muitos frutos do ponto de vista de publicações. Agradeço aos meus colegas de trabalho e amigos do Laboratório WOCA pelo companheirismo, pelos conselhos e momentos compartilhados; tem sido uma aprendizagem constante e um prazer imenso trabalhar com todos vocês.

Por fim, agradeço à Fundação Instituto Nacional de Telecomunicações (FINATEL), à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), à Financiadora de Estudos e Projetos (FINEP) e aos projetos Sistemas de Comunicações Móveis de 6ª Geração (6G) do Centro de Referência em Radiocomunicações (CRR) e Rede Nacional de Ensino e Pesquisa (RNP) IoT pelo apoio financeiro.

Marcello Caldano de Melo

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

2D	Bidimensional
3D	Tridimensional
5G	Sistema de telefonia móvel de quinta geração (5^{th} generation mobile system)
6G	Sistema de telefonia móvel de sexta geração (6^{th} generation mobile system)
ANN	Redes neurais artificiais (*Artificial Neural Networks*)
BP	Propagação reversa (*Backpropagation*)
CNN	Redes neurais convolucionais (*Convolutional Neural Network*)
CPS	Stripline coplanar (*Coplanar stripline*)
CPW	Guia de onda coplanar (*Coplanar Waveguide*)
CSI	Informações de estado do canal (*Channel State Information*)
DE	Algoritmo evolucionário diferencial (*Differential Evolucionary*)
DL	Aprendizado profundo (*Deep Learning*)
DM	Tomador de decisão (*Decision Maker*)
DOE	Desenvolvimento de experimentos (*Design of experiments*)
EM	Eletromagnéticos
FORTRAN	*Formula Translation System*
FP	Fronteira de Pareto
GA	Algoritmo genético (*Genetic Algorithm*)
GAN	Rede adversarial generativa (*Generative Adversarial Networks*)
HFSS	Simulador de estrutura de alta frequência (*High Frequency Structure Simulator*)
I4.0	Indústria 4.0
IA	Inteligência Artificial (*Artificial Intelligence*)
IP	Protocolo de internet (*Internet Protocol*)
LTE	Long Term Evolution
MIMO	*Multiple Input Multiple Output*
ML	Aprendizagem de máquina (*Machine Learning*)
MLP	Perceptron de múltiplas camadas (*Multilayer Perceptron*)
MO	Multiobjetivos (Multiobjective)
MOEA	Algoritmos evolucionários de otimização multiobjetivo (*Multiobjective Evolutionary Algorithm*)
MOEA/D	Algoritmo evolucionário baseado em decomposição (*Multiobjective Evolutionary Algorithm based on Decomposition*)
MOOP	Processo de otimização multiobjetivo (*Multiobjective Optimization*

	Process)
MSE	Erro médio quadrático (*mean square error*)
NSGA-II	Algoritmo genético de ordenação não-dominada (*Non-dominated Sorting Genetic Algorithm*)
P	Conjunto Pareto-ótimo
PCS	Sistemas de comunicação pessoal (*Personal Communication Systems*)
PSO	Otimização por enxame de partículas (*Particle Swarm Optimization*)
RBF	Função de base radial (*Radial Base Function*)
relu	Unidade linear retificada
RF	*Random Forest*
RFC	Relação frente-costa (*Front-to-Back Ratio*)
RNN	Redes neurais recorrentes (*Recurrent Neural Networks*)
SLL	Nível dos lobos lateriais
SM	Modelo substituto (*Surrogate Model*)
SO	Um único objetivo (Single Objective)
SVR	*Support Vector Regression*
tanh	Tangente hiperbólica
WLAN	Redes locais sem fio (*Wireless Local Area Network*)

Lista de Símbolos

x_N	N-ésimo dado da camada de entrada
W_{in}	Peso associado a i-ésima camada escondida com a n-ésima entrada de dados
y_n	N-ésimo dado da camada de saída
$e(x)$	Erro do modelo estimado
$d\,(x)_{ideal}$	Valor ideal da amostra fornecida
$d\,(x)_{est}$	Valor estimado da rede neural
∂p_{ni}	Peso da conexão do n-ésimo neurônio da i-ésima camda oculta
∂s_i	Saída do i-ésimo neurônio
net_n	Soma dos pesos das entradas de n neurônios
η	Taxa de aprendizagem
x	Vetor de entrada de dimensão d
d	Número de dimensões
k	Número de objetivos
$\vec{u}$	Vetor genérico de d-dimensões
$\vec{v}$	Vetor genérico de d-dimensões
Ω	Espaço de objetivos
u_i	I-ésimo valor do vetor $\vec{u}$
v_i	I-ésimo valor do vetor $\vec{v}$
ε_r	Constante dielétrica
tan δ	Tangente de perdas
V	Conjunto de pares entrada-saída da antena MPPD
M	Conjunto de variáveis de entrada da antena MPPD
N	Conjunto de variáveis de saída da antena MPPD
L	Comprimento do dipolo
t	Largura do dipolo
f	Frequência de ressonância
BW	Largura de faixa
E	Conjunto de pares entrada-saída da antena MSPD
Q	Conjunto de variáveis de entrada da antena MSPD
R	Conjunto de variáveis de saída da antena MSPD
l_{dMPPD}	Limite inferior para a antena MPPD
u_{dMPPD}	Limite superior para a antena MPPD
S	Espessura da estrutura de casamento de impedância
l_{dMSPD}	Limite inferior para a antena MSPD
u_{dMSPD}	Limite superior para a antena MSPD

L_1	Comprimento do elemento ativo da antena Quasi-Yagi
L_2	Comprimento do elemento ativo da antena Quasi-Yagi
L_3	Comprimento do elemento ativo da antena Quasi-Yagi
L_4	Elemento diretor da antena Quasi-Yagi
L_5	Comprimento vertical da CPW na antena Quasi-Yagi
L_p	Comprimento horizontal da CPW na antena Quasi-Yagi
W	Conjunto de pares entrada-saída da antena Quasi-Yagi
X	Conjunto de variáveis de entrada da antena Quasi-Yagi
Y	Conjunto de variáveis de saída da antena Quasi-Yagi
l_{QY}	Limite inferior para a antena Quasi-Yagi
u_{QY}	Limite superior para a antena Quasi-Yagi
f_1	Frequência de ressonância para a antena Quasi-Yagi
f_2	Frequência de ressonância para a antena Quasi-Yagi
f_3	Frequência de ressonância para a antena Quasi-Yagi

Lista de Publicações

❶ **M. C. MELO**, P. B. Santos, E. Faustino Jr., C. J. Bastos-Filho e Arismar Cerqueira S. Jr., "Computational Intelligence-based Methodology for Antenna Development," Journal of Engineering Applications of Artificial Intelligence, Nov 2020 (Submetido).

❷ H. R. D. Filgueiras, R. M. Borges, **M. Caldano Melo**, T. H. Brandão and Arismar Cerqueira S. Jr., "Dual-Band Wireless Fronthaul Using a FSS-Based Focal-Point/Cassegrain Antenna Assisted by an Optical Midhaul,"in IEEE Access, vol. 7, pp. 112578-112587, 2019.

❸ E. Faustino, **Melo, Marcello C.**, P. B. Santos, C. J. Bastos-Filho, Arismar Cerqueira S. Jr., E. Barboza, "Comparison of Machine Learning Algorithms for Application in Antenna Design,"Fortaleza: SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC), 2021 (Submetido).

❹ A. A. C. Alves, **M. C. MELO**, J. J. SIQUEIRA, F. Zanella, J. R. Mejía-Salazar e Arismar Cerqueira S. Jr., "Plasmonic Nanoantennas for 6G Intra/Inter-Chip Optical-Wireless Communications," Lapland: 6G Wireless Submit Congress, 2020.

❺ P. B. Santos, **Melo, Marcello C.**, E. Faustino, Arismar Cerqueira S. Jr., and C. J. A. Bastos-Filho, "A Comparison of Evolutionary Multi-Objective Optimization Algorithms Applied to Antenna Design," Lecture Notes in Computer Science, pp. 123–134, 2020.

❻ **M. C. MELO**, P. B. Santos, E. Faustino Jr., C. J. Bastos-Filho e Arismar Cerqueira S. Jr.,"Inteligência Computacional Aplicada no Desenvolvimento de Antenas,"Petrolina: IX Conferência Nacional em Comunicações, Redes e Segurança da Informação (ENCOM), pp. 95–96, Out 2019.

❼ H. R. D. Filgueiras, **M. C. MELO**, T. H. Brandão, and Arismar Cerqueira S. Jr., "Dual-Band Parabolic Antenna for Enhanced Performance in High Throughput Links," Aveiro: SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC), pp. 123–134, 2019.

Resumo

O desenvolvimento de antenas é uma tarefa desafiadora, que pode levar muito tempo utilizando métodos convencionais e demandar alta capacidade computacional, devido à necessidade de muitas varreduras e simulações. Neste trabalho, é proposta uma metodologia eficiente e precisa baseada em Inteligência Computacional para o desenvolvimento e a otimização de antenas. A solução técnica computacional consiste na aplicação de um modelo substituto, composto por uma rede neural artificial do tipo perceptron de múltiplas camadas (*Multilayer Perceptron*, MLP) com retropropagação para regressão. Aliado ao modelo, duas estratégias meta-heurísticas de otimização multiobjetivo, algoritmo genético de ordenação não-dominada (*Non-dominated Sorting Genetic Algorithm*, NSGA-II) e o algoritmo evolucionário baseado em decomposição (*Multiobjective Evolutionary Algorithm based on Decomposition*, MOEA/D), são utilizadas para contornar os desafios apresentados para o desenvolvimento de antenas por métodos tradicionais. Como prova de conceito da metodologia proposta, três estudos de caso são apresentados: uma antena do tipo dipolo de meia onda, uma antena do tipo dipolo de onda completa e uma antena do tipo Quasi-Yagi. Realizou-se comparações entre os modelos desenvolvidos com a metodologia proposta e os resultados obtidos pelo *software* de simulação eletromagnética ANSYS HFSS, com o intuito de comprovar a aplicação e efetividade da metodologia desenvolvida. Seguindo a metodologia proposta, os pares entrada-saída para a antena do tipo dipolo com porta casada (Matched Port Printed Dipole, MPPD) foram obtidos em 53 segundos pelo algoritmo NSGA-II com a finalidade de maximizar a largura de faixa centrada em 3,5 GHz, o que resultou em uma banda de operação de 18% centrada em 3,53 GHz. Na antena dipolo com estrutura de casamento de impedância (Matching Structure Printed Dipole, MSPD) os valores das dimensões ótimas foram obtidas em 5 segundos com o objetivo de maximizar a largura de faixa centrada em 3,5 GHz. Como resultado, as saídas estimadas foram uma banda de 21,4% centrada em 3,5 GHz. Por fim, para o desenvolvimento e otimização da antena do tipo Quasi-Yagi, o algoritmo levou cerca de 2 minutos para encontrar as melhores dimensões da estrutura de casamento de impedância que minimizem os coeficientes de reflexão em 1,9, 2,6 e 3,5 GHz, simultâneamente. Os resultados obtidos comprovam o potencial da metodologia como alternativa complementar aos *softwares* de simulação eletromagnética para o desenvolvimento e otimização de antenas.

***Palavras-Chave*:** Aprendizagem de máquina, Desenvolvimento de antenas, Inteligência Artificial, Inteligência Computacional e Otimização multiobjetivo.

Abstract

The antenna design is a challenging task, which might be time-consuming using computational methods that typically require high computational capability, due to the need for several sweeps and re-running processes. This work proposes an efficient and accurate computational intelligence-based methodology for antenna design and optimization. The computational technical solution consists of a surrogate model application, composed of a Multilayer Perceptron (MLP) artificial neural network with backpropagation for the regression process. Combined with the model, two multiobjective optimization meta-heuristic strategies, Non-dominated Sorting Genetic Algorithm (NSGA-II) and Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D), are used to overcome the mentioned issues from the antenna design traditional method. As proof of the proposed methodology concept, three studies of case are reported: a half-wavelength dipole; a complete-wavelength dipole; Quasi-Yagi antenna. Comparisons between the models developed with the proposed methodology and results by the full-wave numerical simulations software from ANSYS HFSS are performed aiming to demonstrate the developed methodology application and efficiency. According to the proposed methodology, the matched port printed dipole antenna input-output sets were obtained in 53 seconds by the NSGA-II algorithm to maximize the bandwidth centered at 3.5 GHz, which resulted in 18 % operation band centered at 3.53 GHz. In the matching impedance structure printed dipole, the optimal dimensions were obtained in 5 seconds aiming to maximize the bandwidth centered at 3.5 GHz. The estimated output was 21.4% band centered at 3.5 GHz. Finally, for Quasi-Yagi design and optimization, the NSGA-II algorithm spent 2 minutes to find the best impedance matching structure dimensions that minimize simultaneously the reflection coefficient at 1.9, 2.6, and 3.5 GHz. The obtained results prove the methodology potential as a complementary alternative to the electromagnetic simulation software for antenna design and optimization.

Keyword**:** *Antennas design, Artificial Intelligence, Computational Intelligence, Machine learning, Multi-objective optimization.*

Capítulo 1

1. Introdução

1.1 Contextualização e Motivação

Ao longo da história, os avanços da tecnologias revolucionaram a manufatura e provocaram um grande salto no processo de industrialização. Por exemplo, o crescimento na quantidade de dispositivos móveis e novas aplicações em tempo real permitiram a conexão de pessoas e dispositivos em qualquer lugar e a todo momento. A Indústria ganhou a autonomia do controle e monitoramento da produção em tempo real e máquinas pesadas são capazes de trocar informações de gerenciamento e controle entre si durante o processo produtivo [7] [8]. Tecnologias desse tipo criaram um modo de vida, o qual é caracterizado por alto tráfego de dados por minuto, vários dispositivos conectados ao mesmo tempo, clientes cada vez mais exigentes por informações em tempo real e que requerem maior qualidade de serviço [8]. Dessa forma, as revoluções industriais ocorrem de acordo com as significativas mudanças nas tecnologias, bem como na forma com que as pessoas vivem.

A Primeira Revolução Industrial iniciou-se com a criação de máquinas a vapor, o que levou a um aumento na produção. A Segunda Revolução foi norteada pela massificação da produção e pelo uso da energia elétrica, enquanto que, na Terceira Revolução Industrial, inseriu-se o uso de eletrônicos e o aperfeiçoamento das técnicas de produção em massa, o que alavancou ainda mais o processo de fabricação [9]. Atualmente, a Quarta Revolução Industrial, que tem como maior representante a Indústria 4.0 (I4.0) [7], segue uma nova demanda global, na qual propõe uma disrupção do processo produtivo a partir do uso de máquinas inteligentes e conectadas [9]. S. Gallego e colaboradores [10], destacam nove principais tecnologias que norteiam esta nova revolução industrial, que são: *big data* e *analytics*, robôs autônomos, simulações, integração vertical e horizontal, internet das coisas, segurança cibernética, computação

em nuvem, manufatura aditiva e realidade virtual. A Figura 1.1 representa os principais avanços de cada revolução industrial, bem como o aumento da complexidade com o passar dos anos [1]. Tais tecnologias são potencializadas pelos avanços das técnicas de Inteligência Artificial (*Artificial Intelligence*, IA) e sensores, que passam a fazer parte do processo produtivo a fim de estimar, classificar e identificar falhas, danos, ou, até mesmo, pontos a serem melhorados. Dentro desse cenário, o uso de simulação computacional passa a ser uma ferramenta indispensável e fundamental para que sensores e máquinas sejam fabricados com maior precisão, menor custo e tempo [11].

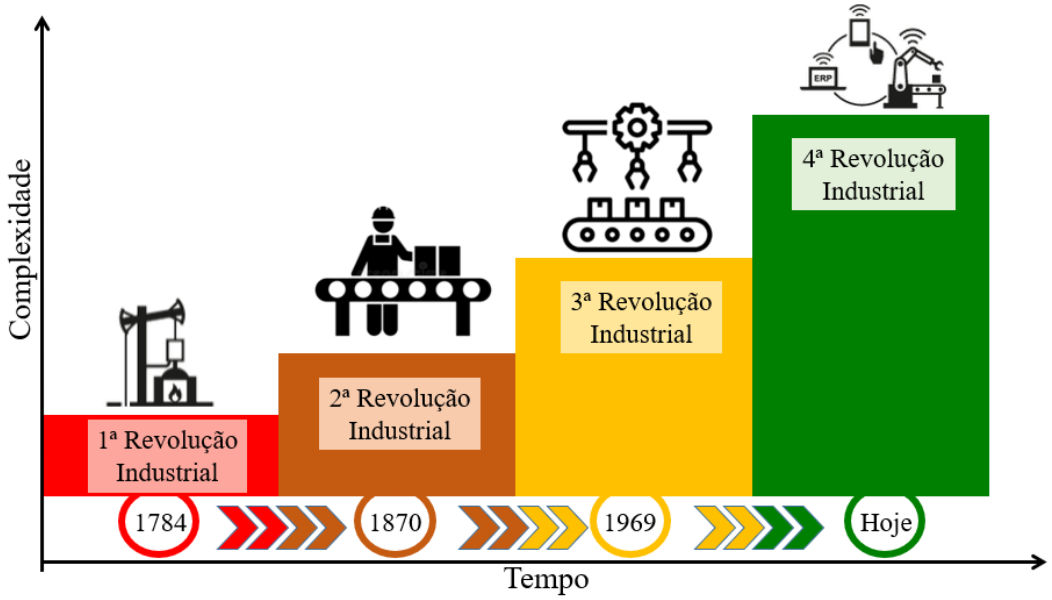


Figura 1.1: *Principais pontos de cada revolução industrial, bem como o aumento da complexidade com o passar dos anos [1].*

A simulação computacional teve início na década de 50, no período pós-guerra, nas indústrias siderúrgicas e aeroespaciais, para solucionar problemas complexos e com modelos de simulação mais complexos ainda [12]. A linguagem de programação usada era *Formula Translation System* (FORTRAN), que posteriormente foi especializada para a criação dos modelos de simulação. A expansão de técnicas de simulação disseminou-se no início dos anos 80, durante a Terceira Revolução Industrial, a qual foi, primeiramente, utilizada nas indústrias automotiva e de base[1]. Os resultados fornecidos eram de baixa qualidade gráfica e, em sua maioria, apresentavam apenas resultados numéricos. No início dos anos 90, a computação gráfica revolucionou o mercado de simulações, fornecendo modelos bidimensionais (2D). Na primeira década dos anos 2000, as simulações computacionais tornaram-se parte do processo produtivo e da manufatura. Atualmente, modelos tridimensionais (tridimensional (3D)) e com alta resolução são apresentados e usados como ferramentas principais de pré-projetos, a

[1] indústria que trabalha com atividades de extração e transformação de matéria-prima.

fim de reduzir custos com prototipagem e desperdício de recursos, além de minimizar potenciais falhas e reduzir o tempo de fabricação [12] [13].

Com o advento dos sistema de telefonia móvel de quinta geração (5^{th} *generation mobile system*, 5G) e, futuramente, os sistema de telefonia móvel de sexta geração (6^{th} *generation mobile system*, 6G), as técnicas de simulações modernas, bem como o uso de técnicas de inteligência artificial, passam a ter grande importância para o progresso das comunicações [14] [15]. Por exemplo, o uso de IA, em conjunto com modelos de simulação computacional, para a estimação de canais *Multiple Input Multiple Output* (MIMO) massivo [16], sensoriamento de canal [17], otimização de redes ópticas [18] e no desenvolvimento e otimização de antenas [19]. Dentro dessa última aplicação, destacam-se as redes neurais artificiais (*Artificial Neural Networks*, ANN), que são usadas para fazer inferências a respeito de outros conjuntos de dados, o que resulta no processo de aprendizagem acerca de um sistema em específico [2]. A Figura 1.2 apresenta uma visão geral da metodologia para a aplicação de Inteligência Artificial ao desenvolvimento e à otimização de antenas de maneira genérica, a qual é fundamentada na contrução de um modelo de aprendizagem, que é formado por uma base de dados, que contém amostras das informações de entrada e saída com relação aos parâmetros da antena, como coeficiente de reflexão, largura de faixa, frequência de ressonância e outros. Depois, o modelo de aprendizagem é otimizado e os melhores pares de entrada, correspondentes as melhores saídas são selecionados.

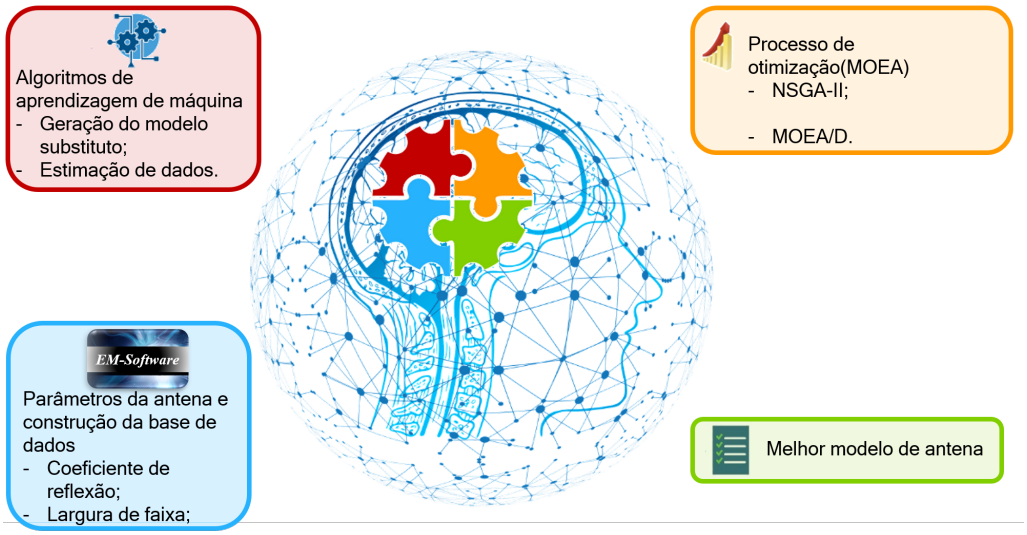


Figura 1.2: *Visão geral da metodologia para a aplicação de inteligência artificial ao desenvolvimento e à otimização de antenas.*

1.2 Revisão Bibliográfica

As redes neurais artificiais foram desenvolvidas por Warren McCulloch e Walter Pitts em 1943 [3]. Essas redes são um modelo simplificado do sistema nervoso humano central, no qual as múltiplas conexões entre neurônios formam uma rede de aprendizado. Na perspectiva de aplicações em engenharia, as ANNs podem ser definidas de muitas formas; porém, a que mais se destaca é a definição proposta por S. Haykin [2], em que: "Uma rede neural é um massivo processo paralelo distribuído, composto por unidades de processos simples, que tem uma propensão natural em armazenar conhecimento de maneira exponencial e torná-lo disponível para o uso".

As primeiras implementações das redes neurais no campo de comunicações sem fio foram na década de 90, em antenas de micro linha de fita e de guia de onda coplanar (*Coplanar Waveguide*, CPW) para reduzir o tempo de análise eletromagnética, minimizar a utilização de complexos processos de iteração e evitar o uso excessivo da unidade central de processamento do computador [20–22]. Tais implementações são possíveis a partir de redes neurais do tipo perceptron de múltiplas camadas (*Multilayer Perceptron*, MLP), Rede de Salto de Campos, Redes de Função de Base Radial, entre outras [2] [23–26].

Nas últimas décadas, a aplicação de ANNs no desenvolvimento de antenas vem crescendo [27–30]. O princípio de funcionamento dessa técnica consiste em a rede neural aprender sobre os dados eletromagnéticos (EM) através de um processo exaustivo de treinamento. Após essa etapa, a rede pode ser usada como um modelo substituto (*Surrogate Model*, SM) capaz de mimetizar o comportamento das variáveis de saída de uma antena, de acordo com as variáveis de entrada. Um dos primeiros SMs implementados no desenvolvimento de antenas foram explorados em [31] [32]. Em 2011, A. P. Anuradha e colaboradores [31], propuseram uma rede neural do tipo MLP baseada no algoritmo de propagação reversa (*Backpropagation*, BP) com otimização por enxame de partículas (*Particle Swarm Optimization*, PSO), a fim de determinar as dimensões ótimas da estrutura de fractais para que a antena opere em múltiplas bandas. Em 2013, T. Khan e colaboradores [33], elaboraram uma rede neural do tipo MLP para estimar o tamanho da fenda do irradiador de uma antena *patch* retangular e a fenda de ar na camada de substrato simultaneamente. Em 2017, uma rede neural do tipo MLP com algoritmos de propagação reversa foi aplicada no desenvolvimento de uma antena *patch* com fendas [30], dupla banda e polarização circular para reduzir os ciclos de otimização e obter as dimensões ótimas para satisfazer os objetivos de campo próximo e distante.

Em conjunto com a implementação de SM baseado em ANN, existe um vasto

número de algoritmos de otimização que podem ser utilizados para a melhoria dos parâmetros das antenas. D. Ding e G. Wang [34] propuseram um algoritmo evolucionário MOEA/D modificado e compararou seu desempenho com um NSGA-II para otimizar o coeficiente de reflexão de uma antena do tipo *bowtie*. S. Goudos e colaboradores, em [35], realizaram uma *survey* sobre algoritmos evolucionários aplicados ao desenvolvimento de antenas e propagação. Nesse estudo foram considerados três principais algoritmos de otimização com um único objetivo: algoritmo genético (*Genetic Algorithm*, GA), PSO e algoritmo evolucionário diferencial (*Differential Evolucionary*, DE) mostrando a frequência do uso de cada algoritmo nos últimos trinta anos. Por fim, em [36], os autores propuseram a otimização de antenas baseadas em gradiente de busca acelerado, com monitoramento de sensibilidade e na mudança da estrutura. Essa técnica é baseada em algoritmos de regiões de confiança para melhorar o coeficiente de reflexão de uma antena do tipo *patch*.

A Tabela 1.1 destaca os principais pontos dos trabalhos relacionados com esta dissertação baseados nos seguintes aspectos: formato da geração do conjunto de dados, o qual leva em conta o uso de *softwares* eletromagnéticos para a construção da base de dados; regressão, considerando os tipos de regressores utilizados em cada estudo e com qual ferramenta computacional foi desenvolvido o modelo de aprendizagem; otimização, verificar se o modelo de aprendizagem treinado passou pelo processo de otimização e o tipo de otimização levando em conta um único objetivo (*Single Objective*, SO) ou multiobjetivos (*Multiobjective*, MO).

Tabela 1.1: *Comparação da metodologia proposta com o estado da arte.*

				Regressão		Otimização	
Ref.	Autor	Ano	Geração da base de dados com softwares eletromagnéticos	Tipo	Ferramenta	Processo	Tipo
[37]	A. Pietrenko et al.	2013	Sim	Kriging	EM-Solver	Sim	MO
[38]	S. Koziel e S. Ogurtsov	2012					
[39]	S. Koziel et al	2013					
[40]	S. Koziel et al.	2014	Não	GP	-	Sim	MO/SO
[41]	B. Liu et al.	2014					
[42]	J. P. Jacobs et al.	2017					
[43]	L. Tenuti et al.	2013	Não	-	-	Sim	MO
[44]	D. Ding e G. Wang	2016					
[45]	E. D. Ulker e S. Ulker	2016					
[46]	R. Diaz et al.	2016	Sim	ANN	Matlab toolbox	Sim	MO
[47]	J. Dong et al.	2017					
[48]	H. Aliakbari et al.	2016	-	GP	-	Não	-
[49]	Z. Zheng	2017	-	ANN	-	Não	-
[50]	B. Liu et al.	2019					
Este trabalho	**M. C. Melo**	**2021**	**Sim**	**ANN**	**Scikit-Learn**	**Sim**	**MO**

1.3 Objetivo da Dissertação

Esta dissertação tem por objetivo apresentar uma metodologia de desenvolvimento e otimização de antenas fundamentada em aplicação de técnicas de Inteligência Computacional, a fim de comprovar a eficácia desta ferramenta na área do eletromagnetismo. Esta abordagem reduz o tempo computacional do processo de desenvolvimento e otimização de antenas, além de reduzir o número de iterações dos *softwares* de simulação eletromagnéticas, fazendo com que haja menor ocupação de memória dos computadores. Neste trabalho, são utilizadas três antenas como estudos de caso, apresentando comparações entre os modelos desenvolvidos com o uso da metodologia e os resultados obtidos pelo *software* de simulação eletromagnética ANSYS HFSS com o intuito de comprovar a aplicação e efetividade do método desenvolvido.

1.4 Contribuições da Dissertação

Apresentam-se como principais contribuições do trabalho:

- O conceito de aplicação de técnicas de Inteligência Artificial ao desenvolvimento e à otimização de antenas;
- A apresentação de uma nova metodologia baseada em Inteligência Computacional para o desenvolvimento e otimização de antenas;
- A comprovação da eficácia do uso de técnicas de Inteligência Computacional como método complementar aos *softwares* comerciais de simulações eletromagnéticas, reduzindo o custo computacional e o número de iterações nas simulações e minimizando o tempo de desenvolvimento e otimização;
- A validação e comprovação da metodologia em três estudos de caso de desenvolvimento e otimização de antenas.

1.5 Estrutura da Dissertação

O trabalho está estruturado da seguinte maneira. No Capítulo 2 são apresentados os fundamentos teóricos para o melhor entendimento do leitor. No Capítulo 3, a metodologia de desenvolvimento de antenas baseado em Inteligência Computacional é apresentada, bem como a sua implementação nas antenas do tipo dipolo impresso de meia-onda e onda completa, e do tipo Quasi-Yagi. No Capítulo 4, os resultados da implementação da metodologia são validados e discutidos para cada uma das antenas. Finalmente, as conclusões e os trabalhos futuros são apresentados no Capítulo 5.

Capítulo 2

2. Fundamentos Teóricos

A Inteligência Artificial e a aprendizagem de máquina, com os avanços das técnicas de computação inteligente, vem sendo empregadas no setor de telecomunicações [51–54]. Particularmente, algoritmos evolucionários são aplicados para a solução de gerenciamento e roteamento de redes, além da aplicação no desenvolvimento e na otimização de antenas [55] [56]. Este Capítulo apresenta os fundamentos teóricos dos principais conceitos envolvidos no trabalho proposto. Inicia-se por uma breve explicação dos termos Inteligência Artificial e Computacional na Seção 2.1. Segue para as principais aplicações de Inteligência Artificial em telecomunicações na Seção 2.2. Na Seção 2.3, são definidos os conceitos e ferramentas de aprendizagem de máquina. Na Seção 2.4, são explorados os algoritmos de otimização, bem como os indicadores de qualidade utilizados nesse trabalho. Tais algoritmos foram definidos levando em conta sua vasta aplicação em problemas na engenharia e no mundo real.

2.1 Inteligência Artificial e Computacional

Com o crescimento da computação digital e a proliferação da Internet, a Inteligência Artificial passou a fazer parte da ciência moderna e marca uma das quebras de paradigmas tecnológicos da atualidade [57]. A ideia é criar máquinas inteligentes, que podem exibir um comportamento inteligente, além de pensar e aprender como seres humanos. Dentro da grande área de IA existem três grandes campos de pesquisa: o aprendizado de máquina, as redes neurais e o aprendizado profundo (*Deep Learning*, DL) [57]. Dentro desses campos está a Inteligência Computacional [2].

A IC é caracterizada por ter a capacidade de adaptação computacional, tolerância de falhas e alta velocidade computacional [58]. Entende-se por adaptação computacional a habilidade de sistemas se adequarem a mudanças em suas variáveis de entrada e

saída. Esse mecanismo é possibilitado por paradigmas que o compõe, que são: aprendizado por enxame, redes neurais artificiais, computação evolucionária e sistemas artificiais imunes [59].

De forma essencial, IA e IC estudam elementos similares, porém considerando metodologias e histórias diferentes. No conceito moderno, IC tende para aplicações de computação bio-inspirada, como algoritmos evolucionários e genéticos [60]. Enquanto que IA segue para a preferência do uso de técnicas com garantias teóricas, além de apresentar uma parcela significante da comunidade focada em deduções puramente teóricas. Sendo assim, a área em que ambos os interesses se sobrepõem é em aprendizagem de máquina, mais especificamente em redes neurais artificias [57].

2.2 Aplicações de Inteligência Artificial em Telecomunicações

A aplicação de técnicas de Inteligência Artificial vem sendo impulsionada pelo crescimento da capacidade computacional, o aumento do número de sensores, a disponibilidade de banda, o elevado tráfego de dados, bem como a necessidade de serviços com baixa latência e alta confiabilidade em muitos setores da indústria. Em telecomunicações, com o advento do 5G e 6G, a implementação de técnicas de IA tem sido cada vez mais explorada [51].

Por exemplo, em [52], técnicas de IA foram aplicadas para realização de alocação dinâmica de recursos em um *backbone*[1], baseado na reconfiguração de volume e direção de tráfego que foi estimado por uma rede neural artificial. Os algoritmos de otimização vem sendo usados para realizarem a escolha do caminho ótimo de pacotes em redes ópticas para o estabelecimento de conexões e roteamento em redes baseadas em protocolo de internet (*Internet Protocol*, IP) [53] [56]. Há também, a otimização de parâmetros de transmissão óptica, como amplitude do laser e ruído de fase [62], além da medição da qualidade do serviço de transmissão [54]. Além disso, técnicas de IA vêm sendo usadas em serviços de autocorreção (*self-healing*)[2], detecção, estimação e localização de falhas [64] [65].

Nas redes *wireless*, as técnicas de IA vêm sendo implementadas para a coleta dos parâmetros ótimos de entrada, os quais representam o sistema e que podem ser usados

[1]Esquema de ligações centrais de um sistema de redes mais amplo, tipicamente de elevado desempenho e com dimensões continentais [61].

[2]Define qualquer aplicação, serviço, ou sistema que tem a capacidade de encontrar falhas e, sem nenhuma intervenção humana, realizar as alterações necessárias para se restaurar para o estado normal ou definido [63].

para treinar a rede neural. Nesse caso, o uso de Inteligência Artificial é de fundamental importância, uma vez que a análise do canal envolve muitos parâmetros com características aleatórias. Tal aplicação vem sendo usada para a seleção de grupos de feixes, estimação de canais e compressão das informações de estado do canal (*Channel State Information*, CSI). Em [16], foi proposto um algoritmo de estimação de canal e de grupos de feixes para redes veiculares baseado na localização, no tipo de receptor e no que está ao redor do veículo, o que leva à ciência da situação[3] (*Situation Awareness*), antes de selecionar o melhor par de feixes para a transmissão. Em [67], T. O'Shea e colaboradores propuseram um sistema de compressão das informações do estado do canal com *autoencoders*. Esse tipo de sistema treina uma rede neural para atuar como um código corretor de canal ou um decodificador.

Diferente do processo de análise de canal, há muitos problemas para os quais a entrada de dados já está bem definida e cujas saídas também podem ser obtidas com o uso de *softwares* de simulação. Por exemplo, no processo de desenvolvimento de antenas, muitas vezes, ocorre um elevado custo computacional para a execução das simulações, além do longo período para atingir os valores esperados dos parâmetros da antena, como ganho, largura de faixa, casamento de impedância, entre outros. Nesses casos, treina-se uma rede neural para reproduzir o comportamento do *software* que gerou as primeiras saídas e, então, otimiza-se a rede por meio de algoritmos evolucionários, que baseiam-se no modelo construído para encontrar as melhores saídas possíveis e, assim, mitigar o custo computacional e o tempo de execução do protótipo. O processo descrito é chamado de "aprendizado para otimização", o qual é usado como ponto de referência para a elaboração da metodologia proposta nesse trabalho.

2.3 ML- Aprendizagem de Máquina

O aprendizado de máquina é um método de análise de dados que automatiza a construção de modelos analíticos [2]. É o ramo da inteligência artificial baseado na ideia de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana. A Figura 2.1 mostra como IA, ML, ANN e redes de aprendizado profundo (*Deep Learning*, DL) se relacionam [2], de forma que o processo de aprendizagem é fundamentado no uso de modelos de aprendizagem, como *Support Vector Regression* (SVR), *CatBoost*, *Random Forest* (RF) e redes neurais artificiais, que é um dos mais difundidos na bibliografia [46, 47, 50].

[3] É a percepção de elementos e eventos ambientais com relação ao tempo ou espaço, a compreensão de seu significado e a projeção de seu status em um futuro próximo [66].

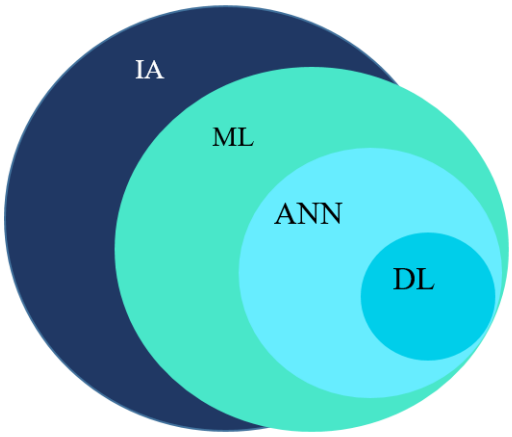


Figura 2.1: *Relação entre IA, ML, ANN e DL [2].*

2.3.1 Redes Neurais Artificiais

Redes neurais artificiais são técnicas de aprendizado computacional inspiradas na estrutura cerebral dos seres humanos, com o objetivo de reproduzir o comportamento deles [2]. As redes neurais são compostas por neurônios artificiais conectados entre si, de forma a realizarem transformações lineares que são ponderadas pelos pesos sinápticos e por um valor de polarização (*bias*), o qual pode ser inicializado para destacar a maior importância de alguma variável de entrada.

A ANN é utilizada para reconhecimento de padrões estatísticos nos dados, que não necessariamente estão explícitos [3]. A arquitetura de uma rede neural consiste em neuronios artificiais, os quais são organizados em camadas e de um conjunto de variáveis conhecidas. Cada neurônio dessa camada distribui seus valores para todos os neurônios da camada escondida. Na conexão entre as camadas de entrada e escondida existe um peso associado, de forma que a camada escondida recebe o produto do valor do neurônio de entrada e o peso da conexão. Cada neurônio na camada escondida agrupa a soma dos valores dos pesos de entrada e, então, aplica uma função não-linear nesse valor. O resultado se torna a saída particular desse neurônio. Então, cada neurônio da camada escondida é conectado aos neurônios de saída, de forma que cada conexão também é ponderada por um peso associado. Por fim, a saída do neurônio usa a soma dos pesos de sua entrada e aplica a função de ativação à eles. O resultado dessa função se torna a saída para a rede neural artificial.

A Figura 2.2 ilustra o processo descrito a partir de uma rede neural genérica. As variáveis de entrada $(x_1, x_2, ..., x_N)$ alimentam a camada de entrada; os neurônios fazem uma ponderação entre os dados de entrada e os pesos associados as conexões; a transformação não-linear é realizada pela função de ativação; as camadas ocultas

processam as informações e enviam o resultado para a camada de saída [3]. Nesse momento, é importante destacar que o processo de treinamento ocorre em duas etapas: **i)** propagação direta, na qual o fluxo de neurônios é em direção a camada de saída, e **ii)** propagação reversa ou retropropagação, no qual os pesos e bias são atualizados com base no erro reportado na propagação direta.

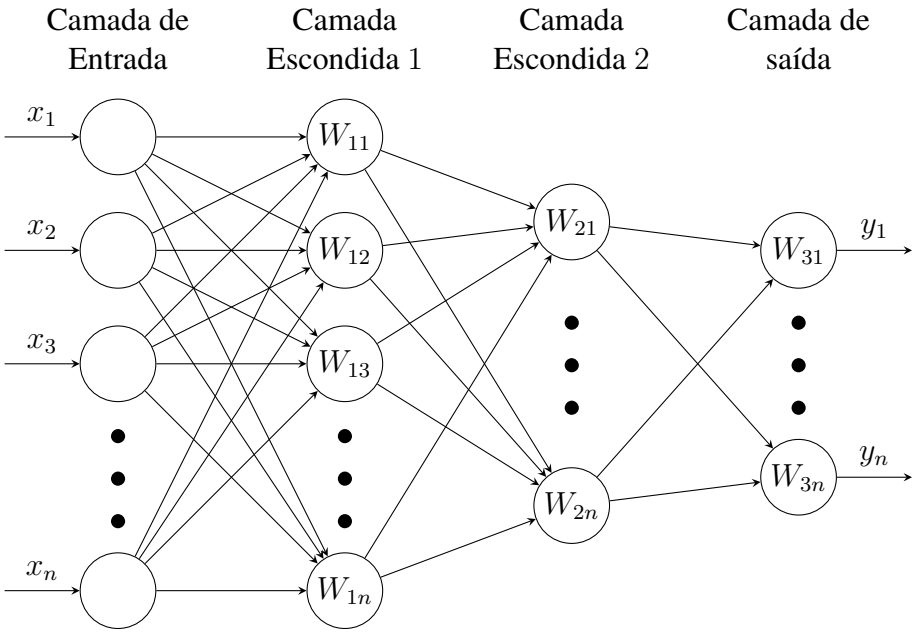


Figura 2.2: *Rede neural artificial genérica.*

A função de ativação é um componente matemático incluído na estrutura das redes neurais a fim de permitir a solução de problemas complexos, como em situações de não-linearidade [2]. A Figura 2.3 [3] representa a inserção da função de ativação dentro de uma rede neural. Como se pode notar, a função de ativação está em um dos últimos estágios de tratamento dos dados até que ela seja enviada à camada de saída. Esse componente tem papel fundamental no processo de aprendizagem da rede, que é identificar a relevância da informação que foi recebida pelo neurônio e decidir se será considerada ou ignorada. O processo de transformação não-linear por meio das funções de ativação é feito após a combinação linear ponderadas das entradas e os pesos de cada conexão. Existem diversos tipos de função de ativação, dos quais os mais populares são: a função linear, sigmóide, tangente hiperbólica (tanh), unidade linear retificada (relu), *leaky* relu e *softmax* [3].

Sendo assim, o processo de treinamento em uma ANN é retratado pela atualização dos pesos da rede, em função dos dados de treinamento utilizados e da função custo (objetivo) usada no algoritmo de aprendizagem da rede, como o algoritmo de propagação reversa [2]. A função custo é a chave para que a rede possa ser treinada com critérios estatísticos no contexto de aprendizagem supervisionada. De forma es-

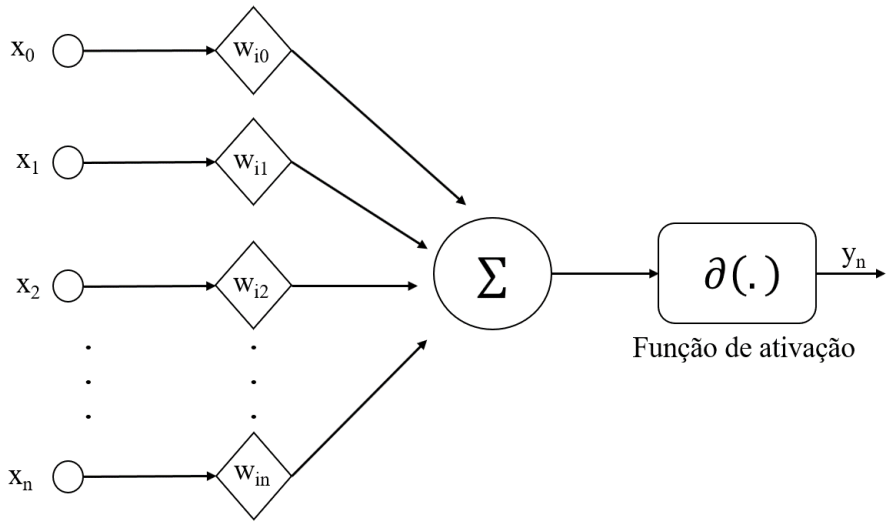


Figura 2.3: *Função de ativação no processo de treinamento da rede neural [3].*

pecífica, conhecendo-se os dados de entrada e saída, a rede vai se ajustando de modo a minimizar o erro entre os neurônios e os dados de saída usados no processo de treinamento. Os dados de teste são utilizados para testar o resultado que o modelo de aprendizagem atravessa. O conjunto de teste não garante a eficácia do modelo, mas sim a forma como ele foi desenvolvido é o que fornece essa análise. O processo de desenvolvimento abrange metodologias e técnicas computacionais eficientes, que levam a um equilíbrio entre desempenho e complexidade. Nesse momento, os dados de teste garantem o desempenho e a capacidade de generalização do modelo. Isso é fundamental para que não ocorra sobreajuste (*overfitting*), quando o modelo aprende em excesso a estrutura dos dados específicos [3]. Ou quando o modelo não captura bem a estrutura dos dados, levando a um subajuste (*underfitting*).

2.3.2 Topologias de Redes Neurais e Aplicações

A maneira como os neurônios estão interconectados define o que se chama topologia de uma rede neural. Algumas dessas topologias tornaram-se muito difundidas, como as topologias do tipo *feedforward* [2], que engloba as redes perceptron de múltiplas camadas e função de base radial (*Radial Base Function*, RBF), redes neurais recorrentes (*Recurrent Neural Networks*, RNN) [3] e redes neurais convolucionais (*Convolutional Neural Network*, CNN) [68]. Por conta de sua vasta aplicação dentro da literatura e potencial de aprendizado [30] [69] [70], a rede do tipo MLP será explorada com mais detalhes e também será implementada nesta dissertação dentro do contexto de antenas.

A rede MLP é formada pelas camadas de entrada, de saída e escondida. A primeira,

comporta os dados de entrada do modelo, que podem ser carregados de forma externa a partir de serviços *web* ou por meio de arquivos no formato *csv*. Na camada de saída, os resultados finais são produzidos advindos das informações providas pelas camadas anteriores. Vale destacar que ambas devem estar presentes na constituição de uma rede. Por fim, a camada escondida, que se localiza entre as camadas de entrada e saída, realiza a soma das entradas com seus respectivos pesos, além de adicionar o valor de *bias* e executar a função de ativação adequada. A rede MLP se distingue das redes *feedforward*, pois possuem uma ou mais camadas intermediárias, as quais são usadas como nós de processamento dos dados de entrada [3]. À medida que o número de camadas intermediárias aumenta, o poder de processamento da rede cresce, uma vez que o número de neurônios e de conexões sinápticas entre eles também se torna maior. As MLPs são comumente aplicadas a problemas de aprendizagem supervisionados, em que é treinado um conjunto de pares entrada-saída, a fim de modelar a correlação, ou dependência, entre cada parâmetro de entrada com sua correspondente saída [23].

Para a topologia do tipo MLP, a equação de saída é a fórmula geral da equação da interpolação linear [3]. A ANN pode ser utilizada para mapear as saídas da rede de acordo com suas entradas, considerando o número adequado de camadas intermediárias; neurônios e a correta parametrização de cada elemento da rede. Tal funcionalidade é de grande valia, considerando a aplicação em antenas, uma vez que é possível fazer o uso da rede neural treinada para estimar parâmetros como coeficiente de reflexão, largura de faixa, ganho e relação frente-costa, fornecendo apenas as entradas ou as saídas.

2.3.3 Processos de Aprendizagem

A característica mais importante das redes neurais é a capacidade de aprender, a partir do ambiente e melhorar seu desempenho por meio deste processo [23]. De maneira geral, assim como nos seres humanos, a rede adquire melhor conhecimento à medida que as informações fornecidas a respeito de determinada situação é maior. Quanto ao tipo das informações, essas são divididas em duas classes: de caráter contínuo, o que caracteriza um processo de regressão; e de caráter discreto, o qual leva a um processo de classificação.

Existem três métodos que são, usualmente, aplicados para o processo de aprendizado de maneira geral [24]: aprendizado supervisionado (*Supervised Learning*); aprendizado não supervisionado (*Unsupervised Learning*); aprendizado de reforço (*Reinforcement Learning*).

- ***Aprendizado supervisionado*** - as redes são treinadas utilizando um conjunto

de entradas e saídas previamente conhecidas, ou seja, para cada combinação de entrada, é fornecida uma saída-alvo (*target value*) e a rede ajusta os pesos sinápticos com o intuito de atingir o menor erro possível [24]. A definição matemática do erro é dada por:

$$e\left(x\right) = d\left(x\right)_{ideal} - d\left(x\right)_{est}, \tag{2.1}$$

onde e(x) é o erro do modelo estimado, $d\left(x\right)_{ideal}$ é o valor ideal da amostra fornecida e $d\left(x\right)_{est}$ é o valor estimado da rede neural. É importante ressaltar que essa sequência de eventos é repetida muitas vezes até que ocorra a convergência e cada peso seja fixado à sua respectiva saída. Em caso de divergência do modelo, faz-se necessária a geração de mais amostras para completar a rede neural a fim de executar o processo de treinamento novamente.

- ***Aprendizado não-supervisionado*** - nesse processo, a rede é capaz de descobrir as características estatísticas do conjunto de entrada, considerando que cada saída corresponde a um conjunto de parâmetros associados à entrada. Diferentemente do aprendizado supervisionado, não há um conjunto de treinamento que relaciona entrada e saída para fornecer um conhecimento prévio à rede. Isso significa que a rede neural desenvolve a própria representação de conhecimento através da identificação ou reconhecimento de padrões entre os dados de entrada [2]. Nesse processo, são desempenhadas duas principais atividades de aprendizagem [24]: redução de dimensionalidade e agrupamento (*clustering*). A primeira é utilizada para encontrar uma representação de baixa dimensionalidade dos dados, de modo a preservar a estrutura ou variabilidade contida no conjunto de dados original. Enquanto que a segunda, é usada para construir agrupamentos de dados a partir da exploração estatística das variáveis de entrada.
- ***Aprendizado de reforço*** - a rede interage, dinamicamente, com o ambiente em que se encontra para atingir um objetivo determinado. Nesse tipo de aprendizado, são elaborados modelos capazes de ter a percepção do ambiente em que se encontram (*concious-awareness*) e de se reorganizar automaticamente. A maior parte das aplicações desse tipo de aprendizagem está em robôs [71], carros autônomos [72] e no gerenciamento de redes de computadores [73].

Dessa forma, independente da maneira de como se adquire conhecimento, o mesmo se dá pelo ajuste dos pesos sinápticos da rede neural. Tal atividade é realizada aplicando algoritmos de aprendizagem, a fim de encontrar o menor erro entre valores preditos e conhecidos. O algoritmo de propagação reversa é um dos métodos mais utilizados para calcular os gradientes da função erro, a qual representa a diferença entre os valores preditos e os valores conhecidos, em relação aos dados da ANN [74]. Ele

funciona computando os gradientes da camada de saída e utiliza-os para calcular os gradientes da camada anterior. Esse processo se repete até a camada de entrada. A ideia básica é aplicar a regra da cadeia para computar a influência dos pesos sinápticos da função erro *e*, a fim de minimizar o erro quadrático médio, ou seja [74]:

$$\frac{\partial e}{\partial p_{ni}} = \frac{\partial e}{\partial s_i} \cdot \frac{\partial s_i}{\partial net_n} \cdot \frac{\partial net_n}{\partial p_{ni}}, \tag{2.2}$$

em que ∂p_{ni} é o peso da conexão do n-ésimo neurônio da i-ésima camada oculta, ∂s_i é o i-ésimo neurônio de saída da ANN e net_n é a soma dos pesos das entradas de *n* neurônios. Nesse caso, para minimização do erro, utiliza-se o método do gradiente descendente [75]. A convergência do erro para o valor esperado está diretamente ligado à taxa de aprendizagem η. Quanto menor o valor de η, mais iterações serão necessárias para que seja atingido o erro que minimize a diferença entre os valores de saída e a predição da ANN. Por outro lado, se η for muito elevado, oscilações podem ocorrer no treinamento e os valores ótimos dos pesos sinápticos não serão atingidos [68].

O algoritmo de propagação reversa é estável, porém para que se atinja um desempenho adequado é preciso fazer com que a taxa de aprendizagem seja pequena, o que o torna lento. Para solucionar esse problema, algoritmos como o de Newton e Gauss-Newton foram desenvolvidos, mas não garantem a mesma estabilidade [23]. Como uma possível solução para a lentidão e a instabilidade, o algoritmo de Levenberg-Marquardt foi desenvolvido. Esse algoritmo utiliza matrizes Hessiana e Jacobiana da função custo para atingir o valor médio ótimo dos pesos sinápticos. No processo de treinamento, as soluções de propagação reversa e de Gauss-Newton são aplicadas alternadamente, o que leva a uma solução mais rápida [68]. Esse algoritmo é utilizado em todos os estudos de caso deste trabalho.

2.4 Otimização Computacional

2.4.1 Otimização de um Único Objetivo e Multiobjetivo

Muitos problemas de tomada de decisão são do tipo multiobjetivo. Por exemplo, minimizar riscos, maximizar a confiabilidade, minimizar desvios de níveis desejados e reduzir custos. O principal objetivo da otimização de um único objetivo é encontrar a melhor solução que corresponda ao valor máximo ou mínimo de uma determinada função objetivo [55]. De forma analítica, um problema de minimização pode ser definido como:

$$\begin{aligned} \underset{x}{\text{minimizar}} \quad & f(x) : \mathbb{R}^d \longrightarrow \mathbb{R} \\ \text{s.a.} \quad & x \in X, \end{aligned} \tag{2.3}$$

em que *x* pode ser um escalar ou um vetor de dimensão *d*, de forma que a saída da função f(x) seja sempre um escalar. Portanto, as saídas são um conjunto ordenado em que todos os pares de pontos podem ser comparados entre si. Uma vez fornecido o domínio da função, torna-se possível projetá-la no espaço de saída e extrair o valor mínimo da solução.

A solução de um único objetivo (*Single Objective*, SO) é útil como uma ferramenta que deve fornecer aos decisores informações sobre a natureza do problema, porém não pode fornecer um conjunto de soluções alternativas que negociem objetivos diferentes um contra o outro. Para esse caso, a otimização multiobjetivos (*Multiobjective*, MO) contempla problemas dessa natureza, que envolvem minimização ou maximização simultânea das funções objetivo, as quais satisfazem um conjunto de restrições [4].

Formalmente, um problema de otimização multiobjetivo pode ser formulado como [4]:

$$\begin{aligned} \underset{x}{\text{minimizar}} \quad & F(x) = (f_1(x), f_2(x), \ldots, f_k(x)) : \mathbb{R}^d \longrightarrow \mathbb{R}^k \\ \text{s.a.} \quad & x \in X, \end{aligned} \tag{2.4}$$

em que *k* é o número de objetivos, *x* é o vetor de entrada de dimensão *d* dentro do espaço euclideano. Muitos dos MOEAs fazem o uso de alguns importantes conceitos em suas implementações, os quais são definidos a seguir [76]:

Definição 1. *Dominância* - seja $\vec{u} = (u_1, \ldots, u_k)$ e $\vec{v} = (v_1, \ldots, v_k)$ dois vetores de mesmo tamanho com d-dimensões. É dito que $\vec{u}$ domina $\vec{v}$, $\vec{u} \preceq \vec{v}$, se e somente se $u_i \leq v_i \; \forall i$ e $\exists i$ s.t. $u_i < v_i$.

Definição 2. *Solução Pareto-ótimo* - um vetor de solução $\vec{u}$, dentro de um espaço Ω, é dito como Pareto-ótimo, se e somente se $\nexists \vec{v} \in \Omega$ s.t. $\vec{v} \preceq \vec{u}$.

Definição 3. *Conjunto Pareto-ótimo (P)* - levando-se em conta a definição de dominância, o conjunto Pareto-ótimo, é definido como o conjunto de todos os vetores de decisão compostos de soluções não dominadas dentro do espaço de objetivos. De maneira matemática, o conjunto Pareto-ótimo é expresso como:

$$P = \{ \, \vec{u} \; \in \; \Omega \; | \; \nexists \, \vec{v} \; \in \; \Omega \; \; s.t. \; \; F(\vec{v}) \preceq F(\vec{u}) \, \} \tag{2.5}$$

Definição 4. *Fronteira de Pareto (FP)* - é a projeção do conjunto Pareto-ótimo

dentro de um espaço objetivo. Portanto, FP pode ser definida como:

$$FP = \{ F(\vec{u}) \mid \vec{u} \in P \} \tag{2.6}$$

A Figura 2.4 representa as definições mencionadas de um conjunto de soluções P, dentro do espaço de objetivos, o que define a fronteira de Pareto.

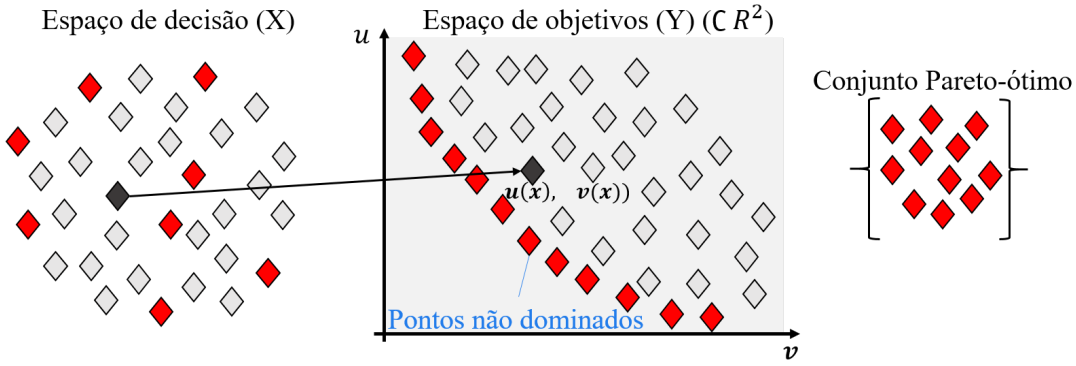


Figura 2.4: *Representação de um conjunto de soluções Pareto-ótimo dentro do espaço de objetivos [4].*

2.4.2 Algoritmos Evolucionários de Otimização Multiobjetivo

Algoritmos evolucionários de otimização multiobjetivo (*Multiobjective Evolutionary Algorithm*, MOEA) são algoritmos baseados em população[4] [6]. Esse tipo de algoritmo pode encontrar uma aproximação para os conjuntos P e FP em uma simples simulação. Um dos maiores desafios para os algoritmos evolucionários de otimização multiobjetivo é balancear as buscas globais e locais dentro do espaço de decisão. A primeira busca leva em conta o espaço de decisão por completo, enquanto que a segunda considera um espaço de busca mais refinado e em áreas mais específicas dentro do espaço de decisão.

Neste trabalho, serão abordados dois algoritmos, com vasta aplicação em problemas de Engenharia e difundidos na comunidade científica [77–79]: o NSGA-II, o qual é baseados em dominância de população; o MOEA/D, que visa a segmentação de processos multiobjetivos em subprocessos para manter a diversidade de soluções. Cada algoritmo é apresentado com mais detalhes a seguir.

- ***NSGA-II*** - Foi proposto por K. Deb e colaboradores [5] como um bloco de construção não-explícito genérico aplicado ao processo de otimização multiobjetivo. A Figura 2.5 apresenta o fluxograma do processo de otimização do

[4]Conjunto de amostras

algoritmo NSGA-II, em que a população de indivíduos compete contra cada um através de um mecanismo elitista. O algoritmo ordena e sorteia cada indivíduo de acordo com seus níveis de não-dominância e utiliza diversos operadores genéticos para gerar diversidade.

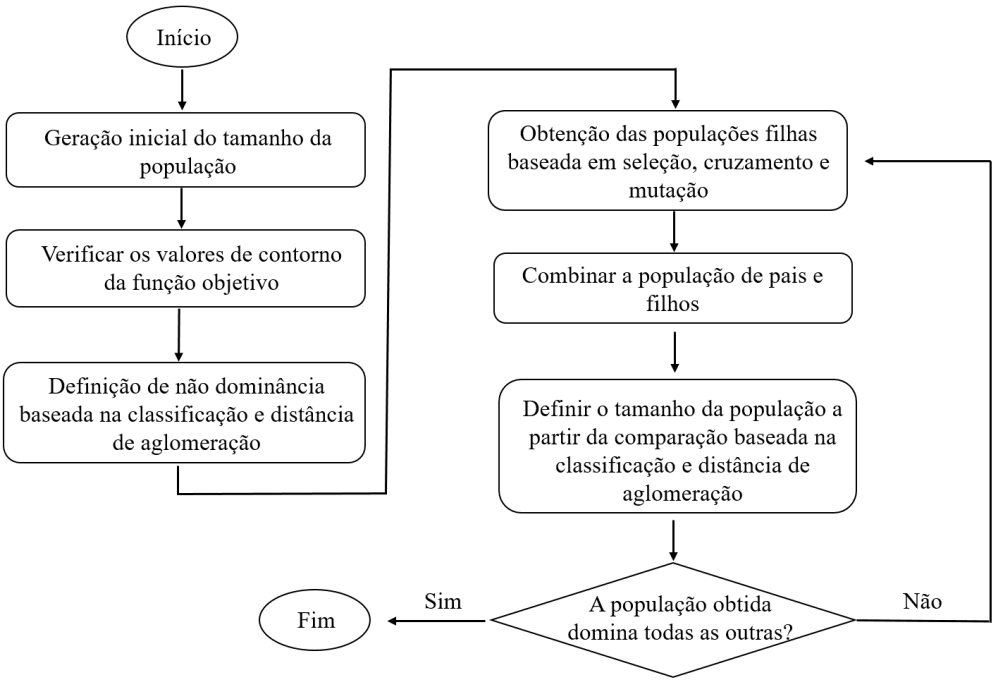


Figura 2.5: *Fluxograma do processo de otimização do algoritmo NSGA-II [5].*

- ***MOEA/D*** - Foi proposto por Q. Zhang e H. Li [6] com o intuito de suprir a deficiência dos MOEAs quando lidam com espaço de objetivos com muitas dimensões, o que leva a dificuldade em manter a diversidade e, rapidamente, toda a população se torna não dominada. A Figura 2.6 mostra o fluxograma do processo de otimização do algoritmo MOEA/D, em que esse algoritmo, baseado em decomposição, visa à segmentação de processos multiobjetivos em vários subprocessos com um único objetivo, associando cada solução a um subproblema e compartilhando informações com seus vizinhos. Cada subproblema mantém na memória uma única solução que é usada para gerar uma nova, a partir da aplicação de operadores genéticos e a ajuda de seus vizinhos. Caso a nova solução seja a melhor, então o processo de um único objetivo é atualizado.

É importante reforçar que os algoritmos utilizados neste trabalho não são exclusivos e outras famílias também podem ser implementadas e analisadas como em [55] [80–82].

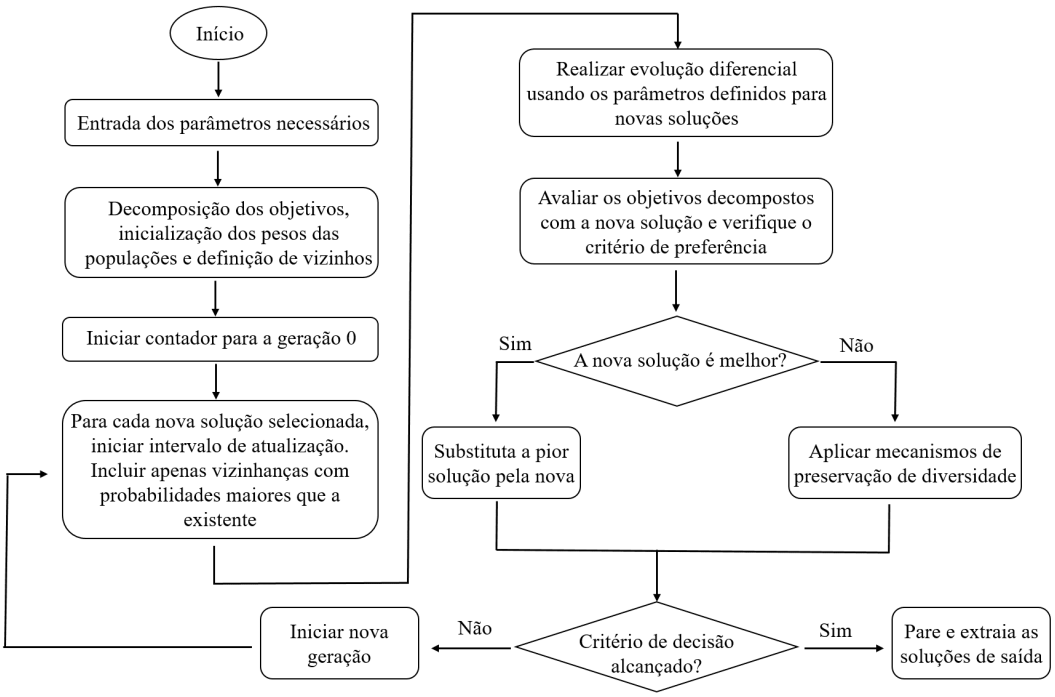


Figura 2.6: *Fluxograma do processo de otimização do algoritmo MOEA/D [6].*

2.4.3 Indicadores de Qualidade

Muitos problemas que envolvem soluções de otimização multiobjetivo apresentam alta complexidade em sua solução [79]. As saídas para a otimização são aproximadas da fronteira de Pareto e a qualidade desse posicionamento precisa ser avaliada. Os indicadores de qualidade são usados para direcionar três aspectos principais: **i)** *Precisão*, que está relacionada com quão perto o algoritmo chegou da fronteira de pareto; **ii)** *Cardinalidade*, a qual está relacionada com a quantidade de soluções que o algoritmo é capaz de encontrar; e **iii)** *Diversidade*, que é quão bem as soluções estão espalhadas nas proximidades da fronteira de Pareto. As métricas relativas a Hypervolume (HV) também vêm sendo largamente implementadas na comunidade científica [83–85]. Particularmente para o presente trabalho, serão definidas apenas três: Hypervolume, Espalhamento e Espaçamento.

- ***Hypervolume*** - também conhecido como hiper área [12], é uma métrica única que quantifica quanto do espaço objetivo foi coberto pela fronteira de Pareto. Para essa métrica, faz-se necessária a definição de um ponto de referência que esteja bem distante da fronteira de Pareto e que seja dominado por todos os outros pontos. Essa métrica pode ser usada para determinar qual MOEA tem a melhor fronteira de Pareto. Sendo assim, quanto maior o valor de HV, melhor.
- ***Espaçamento*** - tem por objetivo mensurar a dispersão das soluções ao longo da fronteira de Pareto proposta pelo MOEA. Isso identifica o aspecto de diversidade

do algoritmo. Nessa métrica são levadas em consideração a distância entre as soluções e sua proximidade com os vizinhos.

- ***Espalhamento*** - É uma extensão da métrica de espaçamento [80], porém leva em conta pontos mais extremos em sua equação a fim de medir o espalhamento das soluções ao longo da fronteira de Pareto.

Capítulo 3

3. Metodologia Proposta para o Desenvolvimento de Antenas baseada em Técnicas de Inteligência Computacional

Em antenas, lida-se com um grande número de funções objetivo com elevado grau de não-linearidade e dispõe de uma vasta gama de parâmetros a serem otimizados, tais como propriedades do diagrama de radiação, casamento de impedância, dimensões máximas, entre outros [86]. As funções objetivo representam uma modelagem matemática de um problema de otimização a ser solucionado que definem o intervalo de valores válidos para a entrada. A aplicação de técnicas de computação inteligente no desenvolvimento de antenas passa a ser uma alternativa viável para a redução do custo computacional e no tempo de execução de projetos, o qual engloba as modificações nas variáveis da antena para melhoria de um, ou mais, parâmetros e a correspondente simulação. Neste Capítulo, será abordada uma nova metodologia para o desenvolvimento e a otimização de antenas baseada em Inteligência Computacional. Além disso, a introdução de três antenas que serão exploradas como estudos de caso para o teste e a validação da metodologia.

3.1 Metodologia Proposta

Algumas metodologias para o desenvolvimento de antenas foram mencionadas no Seção 1.2. Na maioria dos casos, alguns detalhes fundamentais do processo são omitidos, por exemplo: o modo de geração da base de dados, a qual os limites superiores e

inferiores das variáveis de entrada em análise são fundamentais, além da forma como a geração dos valores dentro de cada intervalo foram obtidos; o tipo de modelo substituto empregado para a construção do regresssor, uma vez que existem muitas possiblidades [37] [40] [46] e os algoritmos de otimização, bem como as métricas aplicadas sob o modelo substituto para a minimização ou maximização dos pontos de interesse. Um outro ponto importante a ser ressaltado é a validação de todo o processo. Considerando o modelo substituto, curvas de regressão e valores das métricas de análise de erro, muitas vezes, não são reportados. Ao levar-se em conta a otimização, curvas de convergência da fronteira de Pareto, indicação das métricas de desempenho para os algoritmos e a curva de aproximação da fronteira de Pareto são omitidas.

O contexto acima resulta em lacunas da viabilidade e da precisão do processo para o desenvolvimento de antenas com o auxílio de ferramentas de Inteligência Computacional. Este trabalho apresenta um passo-a-passo bem definido e estruturado de uma metodologia para o desenvolvimento e otimização de antenas assistidas por técnicas de Inteligência Computacional. A Figura 3.1 esquematiza o ciclo de trabalho da metodologia proposta, o qual inicia com a geração da base de dados; passa pelo processo de treinamento e validação do modelo substituto; segue até a definição e aplicação dos algoritmos de otimização; então os valores otimizados são testados e validados no *software* de simulação eletromagnética; e se o objetivo for atingido na validação, o modelo pode ser fabricado, caso contrário, o ciclo é reiniciado para aprimorar a base de dados e o modelo substituto como um todo.

O desenvolvimento e a otimização de antenas com o uso de técnicas de Inteligência Computacional pode ser estruturado nas seguintes etapas: geração da base de dados; geração do modelo substituto; definição dos MOEAs e otimização; e validação do modelo e fabricação.

- Geração da Base de Dados

  O conjunto de dados é a primeira e principal etapa do processo, dado que os pares entrada-saída serão responsáveis por nortear todo o processo de treinamento do modelo substituto. Antes de gerar o conjunto de dados de entrada é preciso definir as variáveis que serão utilizadas. Nesse ponto, é necessário levar em conta os parâmetros da antena que serão utilizados na saída, para alocar, na entrada, as variáveis que mais a influenciam e, também, definir os limites superiores e inferiores que os valores de cada variável podem excursionar.

  Com a preparação realizada, inicia-se a construção do conjunto de dados de entrada. Para essa etapa, duas abordagens podem ser utilizadas: a função *rand* do Matlab, generalista para todos os *softwares* de simulação eletromagnética, ou

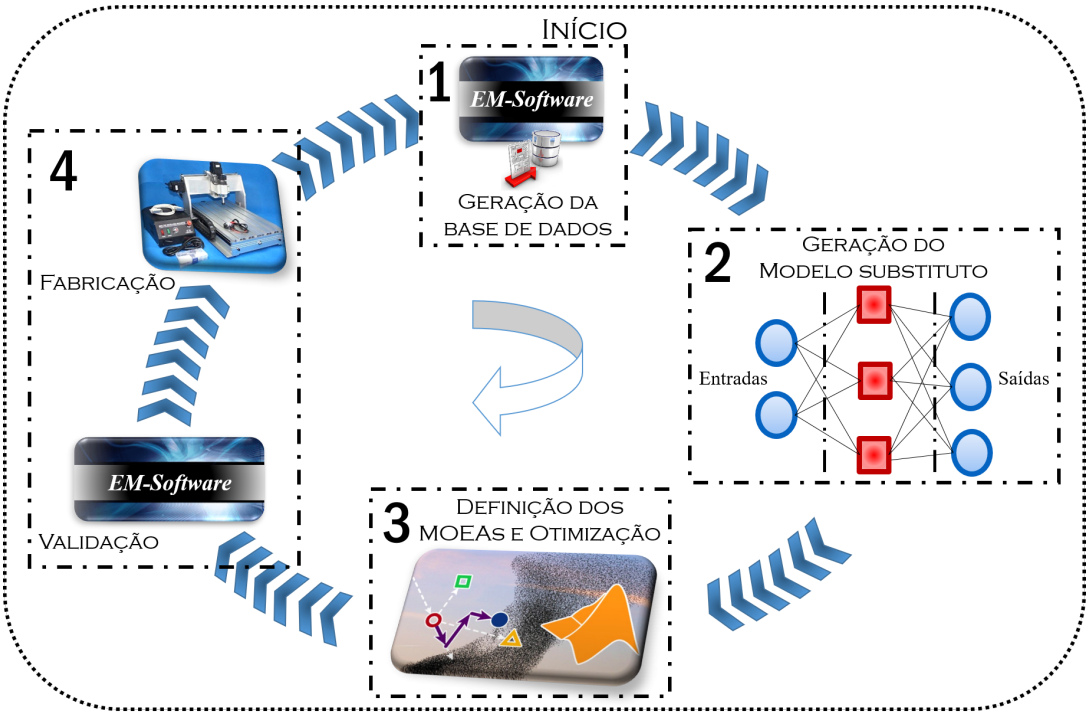


Figura 3.1: *Ciclo de trabalho da metodologia baseada em Inteligência Computacional para o desenvolvimento de antenas.*

por meio da ferramenta de desenvolvimento de experimentos (*Design of experiments*, DOE) do *software* ANSYS HFSS (*High Frequency Structure Simulator*). A função *rand*, aplicada no intervalo dentro dos limites inferiores e superiores das variáveis de entrada, gera arranjos numéricos pseudoaleatórios cujos elementos são uniformemente distribuídos. Após atingir a quantidade de variáveis de entrada, uma tabela contendo esses dados deve ser contruída e importada, no formato *.csv*, até o *software* de simulação eletromagnética, para que cada valor das variáveis de entrada sejam simulados. Com as simulações concluídas, inicia-se a coleta das saídas conforme o parâmetro da antena desejado, concluindo a contrução da base de dados.

Com a segunda abordagem, a ferramenta DOE, o procedimento inicial de definição de variáveis e limites são os mesmos. Com essa ferramenta as variáveis de entrada também são geradas de forma pseudoaleatória, porém com o uso do algoritmo *Latin Hypercube Sampling* (LHS) para evitar que replicações ocorram. Com esse algoritmo, o número de amostras total relacionado a cada variável de entrada é igualmente espaçado em intervalos idênticos entre si, garantindo que apenas uma amostra seja selecionada a cada intervalo de análise. Como o DOE já está disponível dentro do próprio *software* HFSS, a etapa de importação de arquivos pode ser transposta e as simulações para obtenção dos parâmetros de

saídas desejados podem ser iniciadas.

Com isso, a contrução da base de dados está concluída. Independente do método utilizado para a construção da base, é importante ressaltar que a quantidade de pares entrada-saída gerados, para a constituição da base de dados, não são definidos e que variam de acordo com a complexidade do modelo em questão. Por exemplo, antenas que são muito sensíveis a pequenas variações das dimensões de sua estrutura e levam a grandes diferenças em termos de casamento de impedância, ganho, largura de faixa e outros parâmetros, demandam uma maior quantidade de dados para a composição do modelo substituto, de forma que seja possível identificar os padrões que caracterizam o comportamento da estrutura. Da mesma maneira, para antenas mais simples, esse número de amostras será reduzidos. Dessa forma, com a base de dados estruturada, inicia-se a construção do modelo substituto baseado em uma rede neural artificial.

- Geração do Modelo Substituto

  Nesse ponto é iniciado o treinamento da rede neural com a biblioteca *Scikit-learn* utilizando a linguagem Python, o qual garante que o modelo substituto construído tenha a capacidade de generalização e não esteja enviesado. O SM trabalha como uma caixa preta, a qual busca reproduzir as atividades do *software* eletromagnético em um menor tempo, baseado no processo de treinamento. Para validar essa metodologia, uma rede do tipo MLP foi construída, com propagação reversa, para o modelo substituto.

  Nessa rede, utilizou-se o método de validação cruzada, k-*fold*[1] e *GridSearchCV*[2], para auxiliar na definição dos hiperparâmetros, que são as variáveis que controlam o processo de treinamento da rede neural, de forma a garantir que não esteja ocorrendo os efeitos de subajuste ou sobreajuste no modelo, o que resultaria na necessidade de aumentar o tamanho da base de dados, caso haja sobreajuste, ou aumentar a complexidade do modelo de aprendizagem em caso de subajuste. Com a configuração dos hiperparâmetro da rede garantida, os pares entrada-saída são divididos em 75% para treinamento da rede neural, enquanto que 25% foram usados para testar e garantir a capacidade de generalização do modelo quando novos conjuntos de dados são inseridos.

  Para medir a capacidade mencionada, utilizou-se a métrica do erro médio

[1] Esse método consiste em dividir o conjunto total de dados em k-subconjuntos mutuamente exclusivos de mesmo tamanho e, a partir daí, um subconjunto é utilizado para teste e os k-1 restantes são usados para a estimação dos parâmetros.

[2] Classe da biblioteca Scikit-Learn usada para automatizar o processo de ajuste dos parâmetros de um algoritmo. Realiza diversas combinações dos parâmetros e, depois de avaliá-los, armazena-os em um único objeto.

quadrático (*mean square error*, MSE), no qual quanto menor o valor, maior é a capacidade de generalização. Com tal indicação atendida, pode-se garantir que o modelo substituto gerado apresenta a capacidade de reproduzir as atividades do *software* de simulação eletromagnética e é possível seguir para a próxima etapa da metodologia, na qual as saídas do modelo substituto serão usadas como função objetivo pelos algoritmos de otimização.

- Definição dos MOEAs e Otimização

  O processo de otimização multiobjetivo (*Multiobjective Optimization Process*, MOOP) baseado em algoritmos evolucionários consiste em duas etapas: definição dos MOEAs que serão utilizados para a otimização e a análise dos indicadores de qualidade. Vários MOEAs podem ser considerados para a otimização de antenas, como NSGA-II, MOEA/D e outros, de acordo com [55]. Inicia-se então o processo de otimização propriamente dito, o qual usa os valores das saídas do modelo substituto como uma função objetivo para encontrar o valor das variáveis de entrada que maximize, ou minimize, a função objetivo. É importante ressaltar que os valores de entrada, que venham a ser obtidos pelo processo de otimização, não necessariamente corresponderão a um valor já existente no conjunto de dados que compõem o modelo substituto. Com o processo de otimização, novos valores para as variáveis de entrada podem ser originados, baseados na saída definida.

  Uma vez que a etapa de otimização esteja concluída, é preciso analisar os resultados referentes aos indicadores de qualidade, para avaliar os aspectos quanto a proximidade da fronteira de Pareto que cada algoritmo chegou, a quantidade de soluções que puderam ser encontradas e qual a distribuição que as soluções apresentam nas proximidades da fronteira de Pareto considerando cada MOEA definido.

  Ao final da análise, quanto aos indicadores de qualidade, espera-se que os MOEAs escolhidos atendam às métricas definidas que, nessa metodologia, foram: o Hypervolume, Espalhamento e Espaçamento. Dessa forma, a etapa de validação e fabricação pode ser iniciada.

- Validação do Modelo e Fabricação

  Os melhores pares entrada-saída encontrados pelo algoritmo de otimização são selecionados e os valores das variáveis de entrada, correspondentes às melhores saídas, são inseridos no *software* de simulação eletromagnética para avaliar a precisão e a veracidade das saídas estimadas comparando com os valores simulados pelo *software*, antes do processo da construção do protótipo.

Existem três possíveis situações na etapa de validação. Na primeira, a simulação eletromagnética é totalmente confiável para o valor de saída do MOEA, o que significa que o valor do MSE, obtido no teste do modelo substituto, é próximo de zero. Garantindo completa identificação do padrão da estrutura eletromagnética. Em outras situações, o valor de saída oferecido pelo MOEA difere da simulação eletromagnética, porém os resultados ainda são aceitáveis, ou seja, necessita de pequenos ajustes para a situação esperada. Isso significa que o MSE obtido tem um valor baixo, não nulo, que identificou parcialmente o padrão da estrutura eletromagnética. Com isso, o resultado desejado dos parâmetros da antena podem ser alcançados com correções do engenheiro de RF no modelo eletromagnético. Nesses dois últimos casos, o processo de fabricação do protótipo já pode ser iniciado. Por fim, na pior situação possível, os pares de entrada-saída diferem-se totalmente dos valores simulados devido à dispersão apontada pelo MSE. Isso indica que a base de dados elaborada tem baixa representatividade do modelo em questão, de tal forma que a rede neural não conseguiu identificar as características do problema para estruturar o modelo substituto. Nesse momento, torna-se necessário reiniciar o processo, gerar um conjunto de dados complementar para aumentar a informações quanto ao problema, treinar o modelo substituto e aplicar os algoritmos evolucionários novamente.

3.2 Estudos de Casos

Serão apresentados os estudos de casos da metodologia proposta utilizando três antenas: um dipolo de meia-onda impresso com porta casada, de menor complexidade; um dipolo de onda completa impresso com estruturas de casamento de impedância, de complexidade mediana e; uma antena do tipo Quasi-Yagi. Além disso, são apresentadas as estrutura das bases de dados e os esquemáticos das redes neurais dos modelos de estudo.

3.2.1 MPPD - Dipolo Impresso com Porta Casada

A Figura 3.2 representa a antena do tipo dipolo de meia onda impresso com porta casada, denominada como MPPD (*Matched Port Printed Dipole*). A porta casada garante que a resistência de carga seja igual a resistência da fonte, além de apresentar a reatância da carga igual ao negativo da reatância da fonte, levando a máxima transferência de potência [86]. Essa relação é definida como:

$$\begin{aligned} R_r + R_L &= R_g \\ X_A &= -X_g, \end{aligned} \tag{3.1}$$

em que R_r e R_L são a resistência de radiação e de perdas, respectivamente, da antena, R_g a resistência da fonte e X_A e X_g as reatâncias da antena e da fonte respectivamente [86]. Como o valor típico da impedância de entrada de uma antena do tipo dipolo de meia onda apresenta parcela reativa, fez-se o uso da porta casada para considerar uma estrutura puramente resistiva.

Nesse sentido, a antena é composta por um elemento ativo feito de cobre, alimentado por uma porta casada em 73 Ω. Foi considerado um substrato dielétrico de fibra de vidro (FR-4) (ε_r = 4,4 e tan δ = 0,02), de 100 x 25 mm de tamanho total com 1,6 mm de espessura. As variáveis de entrada-saída de decisão são $V = [L,\ t,\ f,\ BW]$, em que *L* e *t* são o comprimento e a largura do dipolo usadas como variáveis de entrada na rede neural, enquanto que f e *BW* são referentes aos valores da frequência de ressonância e largura de faixa, respectivamente, correspondentes a saída da rede neural. É importante destacar que todas as dimensões estão em milímetros.

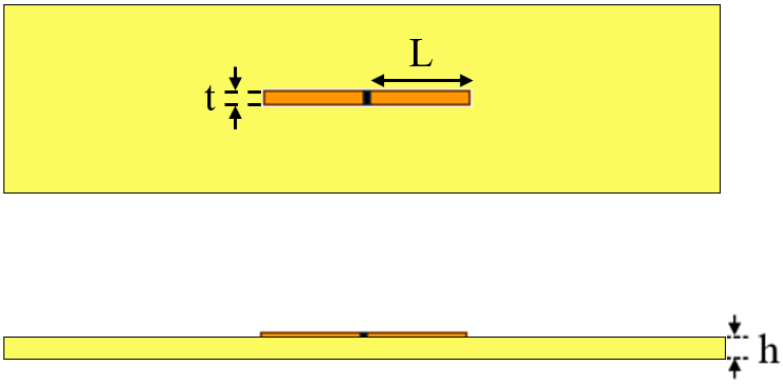


Figura 3.2: *Estrutura da antena do tipo dipolo impressa com porta casada. Vista superior e lateral.*

Quanto à construção da rede neural do modelo substituto, a função *rand*, do Matlab, foi adotada para geração do conjunto de dados de entrada $M = [L,\ t] \in R^n$. Os limites inferiores e superiores para a geração das variáveis de entrada foram definidos como: $l_{dMPPD} = [13,0;\ 0,02]$ e $u_{dMPPD} = [90,0;\ 3,98]$ de *L* e *t* respectivamente. O *software* ANSYS HFSS 2019 R3 [87] foi usado para obter o conjunto de variáveis de saída $N = [f,\ BW]$. A Figura 3.3 apresenta as variáveis de entrada e saída dentro de uma rede neural genérica.

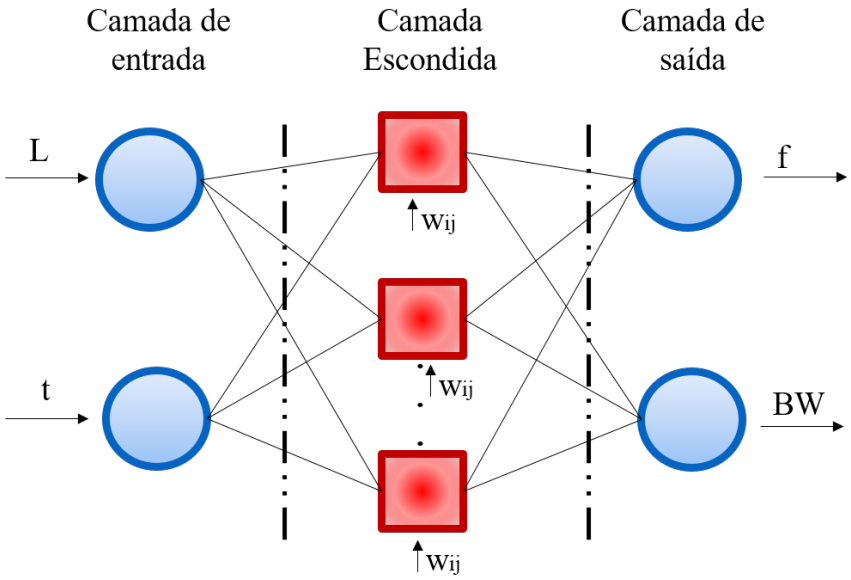


Figura 3.3: *Representação geral da rede neural da antena dipolo impressa com porta casada.*

3.2.2 MSPD - Dipolo Impresso com Estrutura de Casamento de Impedância

A Figura 3.4 mostra a antena do tipo dipolo de onda completa com uma estrutura para casamento de impedância, denominada MSPD (*Matching Structure Printed Dipole*). A estrutura de casamento usada é uma linha de alimentação para casar a impedância real do dipolo, cujo valor típico é 200 Ω originada da expressão da resistencia de radiação disponível em [86], com a impedância de entrada de 50 Ω da fonte. A antena é composta por um elemento ativo, excitado por uma estrutura de casamento de impedância com um conector convencional do tipo SMA. Foi considerado um substrato do tipo FR-4, com 100 x 25 mm de tamanho total e 1,6 mm de espessura. As variáveis de entrada-saída são E = $[L,\ t,\ S,\ f,\ BW]$, em que *L* e *t* são o comprimento e a largura do elemento irradiante, *S* é a espessura da estrutura de casamento de impedância. Tais variáveis serão usadas como entrada na rede neural. Enquanto que *f* e *BW* estão relacionados com a frequência de ressonância e largura de faixa, respectivamente, que correspondem a saída da rede neural.

A função *rand* do Matlab, foi adotada para obter o conjunto de variáveis de entrada, *Q* = $[L,\ t,\ S] \ \epsilon\ R^n$, para a construção da rede neural do modelo substituto. Os limites superiores e inferiores para a geração das variáveis de entrada foram definidos seguindo: l_{dMSPD} = $[13,0;\ 0,02;\ 0,5]$ e u_{dMSPD} = $[90,0;\ 3,98;\ 2,9]$ para *L*, *t* e *S*, respectivamente. O *software* ANSYS HFSS foi usado para obter o conjunto de variáveis de saída *R* = $[f,\ BW]$. A Figura 3.5 apresenta as variáveis de entrada e saída dentro de uma rede neural genérica.

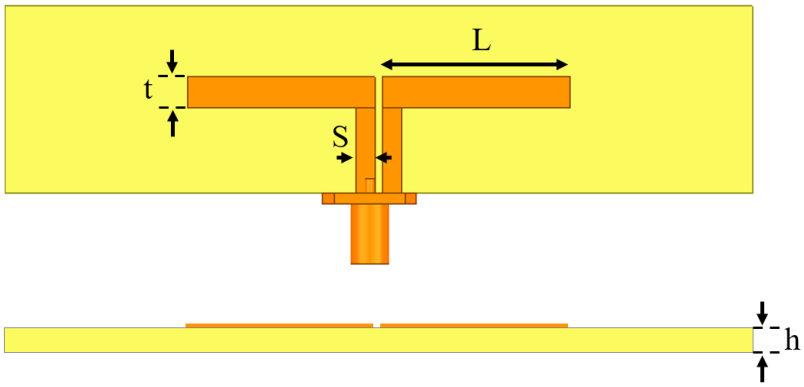


Figura 3.4: *Estrutura da antena do tipo dipolo impressa com estrutura de casamento de impedância. Vista superior e lateral.*

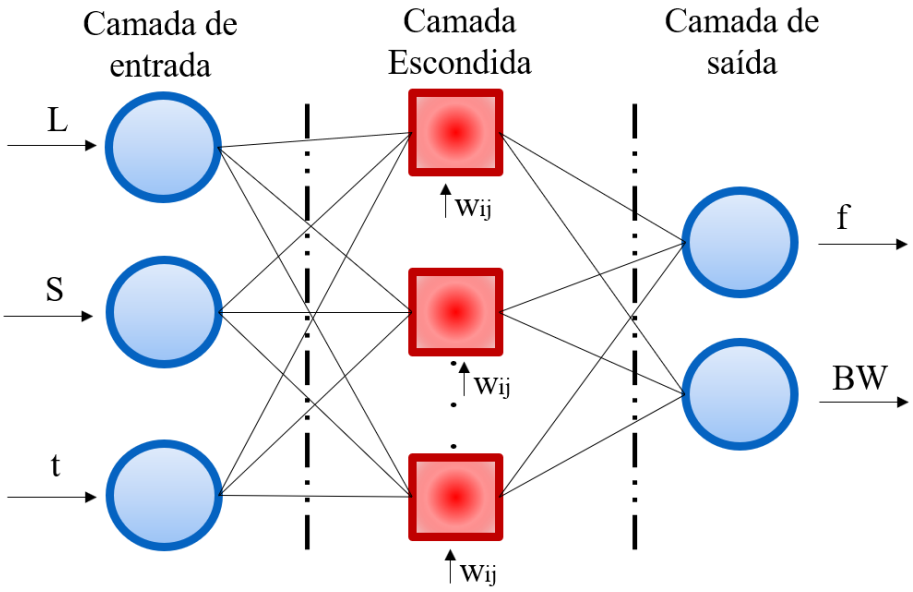


Figura 3.5: *Representação geral da rede neural da antena dipolo impressa com estrutura de casamento de impedância.*

3.2.3 QY - Quasi-Yagi

A antena Quasi-Yagi foi desenvolvida para operar em torno de três bandas distintas: **i)** 1,85 a 1,99 GHz para aplicações sistemas de comunicação pessoal (*Personal Communication Systems*, PCS); **ii)** 2,4 a 2.7 GHz para implementação em redes locais sem fio (*Wireless Local Area Network*, WLAN) (2,4-2,48 GHz) e redes Long Term Evolution (LTE) (2,5-2,69 GHz); e **iii)** 3,3 a 3,7 GHz para aplicação em cenários sub-6 GHz para redes 5G (3,4 - 3,6 GHz). Tal modelo foi inspirado em [88]. A Figura 3.6 representa a antena do tipo QY.

Essa antena é formada por três elementos ativos, L_1 a L_3; um elemento diretor, L_4; uma estrutura de casamento de impedância, L_5 e L_p, que é uma transição de guia de onda coplanar para *stripline* (CPW/CPS) com o objetivo de obter uma defasagem de $180°$ na corrente e atuar como um balun para o casamento de impedância; e um plano de terra truncado que faz a função de um refletor. Foi considerado um substrato do tipo FR-4, com 50 x 60 mm de tamanho total e 1,6 mm de espessura. As variáveis de entrada-saída são W = $[L_5,\ L_p,\ f_1,\ f_2,\ f_3]$, em que L_5 e L_p são as dimensões da estrutura de casamento que serão usadas como variáveis de entrada na rede neural, enquanto que f_1, f_2 e f_3 correspondem às magnitudes do coeficiente de reflexão das frequências de ressonâncias em torno de 1,9, 2,6 e 3,5 GHz, respectivamente, as quais são as variáveis de saída da rede.

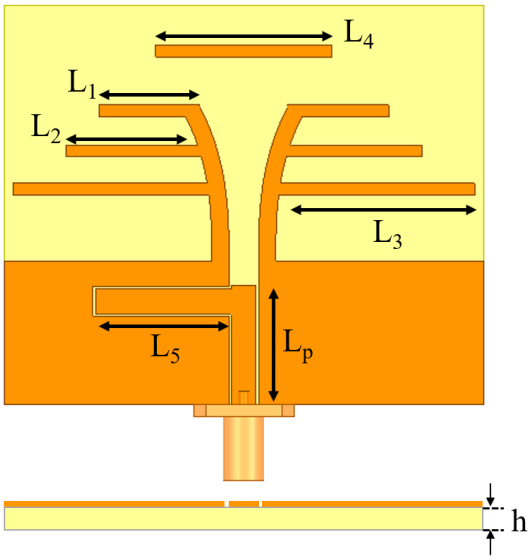


Figura 3.6: *Estrutura da antena Quasi-Yagi. Vista superior e lateral.*

A ferramenta DOE no *software* HFSS com o uso do algoritmo LHS, foi usada para obter o conjunto de variáveis de entrada, X = $[L_5,\ L_p] \in R^n$, de forma pseudoaleatória para a construção da rede neural do modelo substituto. O espaço de desenvolvimento das variáveis de entrada foram definidos seguindo os limites inferiores e superiores: l_{QY} = $[7,5;\ \ 7,5]$ e u_{QY} = $[24,5;\ \ 17,5]$ para L_5 e L_p respectivamente. O *software* ANSYS HFSS foi usado para obter o conjunto de variáveis de saída Y = [f_1, f_2, f_3]. A Figura 3.7 apresenta as variáveis de entrada e saída dentro de uma rede neural genérica.

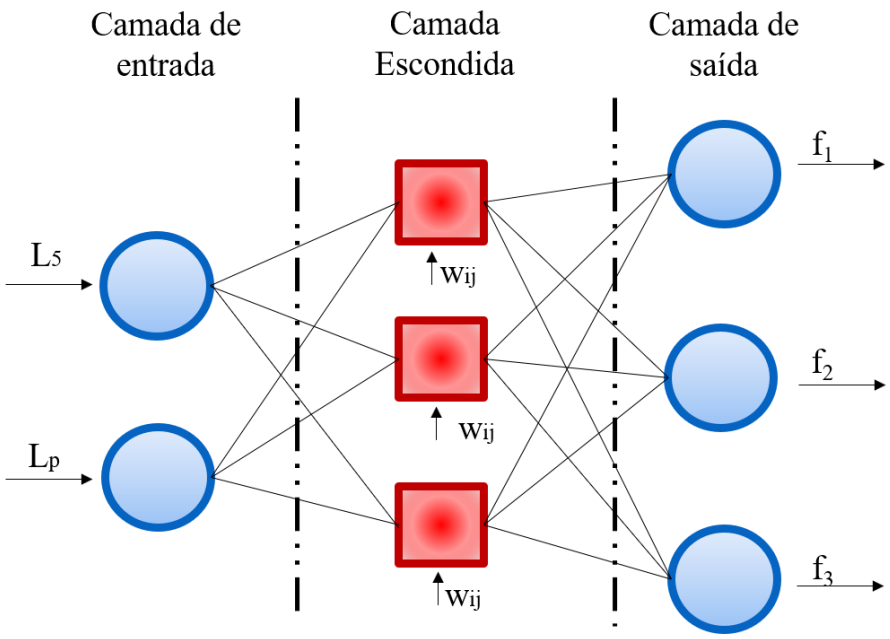


Figura 3.7: *Representação geral da rede neural da antena Quasi-Yagi.*

Capítulo 4

4. Validação da Metodologia para o Desenvolvimento de Antenas

Este Capítulo apresenta a validação da metodologia para o desenvolvimento de antenas, considerando os três estudos de caso. Os algoritmos NSGA-II e MOEA/D foram usados para o processo de otimização dos modelos substitutos. Escolheu-se o algoritmo com melhor desempenho, considerando os indicadores de qualidade Hypervolume, Espaçamento e Espalhamento. Os resultados das dimensões optimizadas de cada caso foram validadas por meio de simulações numéricas com o *software* HFSS e com resultados experimentais. A estratégia de validação do uso de técnicas de IA, seguindo a metodologia, foi desenvolvida seguindo três análises:

Abordagem teórica - leva em conta os tempos de projeto, desenho e simulação da antena com o uso exclusivo da teoria;

Engenheiro de RF - considera o tempo típico gasto por um profissional, especialista no desenvolvimento de antenas;

Técnica de IA - pondera o período gasto pelo algoritmo de otimização entregar os pares entrada-saída, além do tempo de simulação no *software* HFSS para validação dos valores estimados pelo algoritmo.

4.1 MPPD - Dipolo Impresso com Porta Casada

A construção do modelo substituto e o respectivo processo de otimização com os algoritmos multiobjetivo, tem por finalidade minimizar o erro em torno da frequência de ressonância de 3,5 GHz e maximizar a largura de faixa da antena MPPD. Nas próximas subseções, serão apresentados: a estrutura da rede neural usada para a construção do

modelo substituto; o processo de otimização com a definição do melhor algoritmo evolucionário, baseado nas métricas dos indicadores de qualidade; e os resultados simulados da antena dipolo com porta casada.

4.1.1 Rede Neural do MPPD

A rede neural do tipo MLP, para a antena MPPD, foi composta por 171 pares entrada-saída na composição da base de dados, que foram obtidos de maneira empírica com o aumento progressivo na quantidade de amostras, a fim de buscar a melhor relação entre pares entrada-saída que represente a configuração da antena MPPD à rede neural. A base de dados foi então dividida em 75% para composição do treinamento e 25% para o teste do modelo de aprendizagem. Para a definição dos hiperparâmetros que compõem a rede neural, utilizaram-se os métodos k-*fold* com validação cruzada e *GridSearchCV*, de forma a garantir que os efeitos de subajuste ou o sobreajuste não ocorram.

O processo de treinamento consumiu 15 minutos e 48 segundos e obteve-se um erro quadrático médio, MSE, igual a 0,0518. A rede neural foi elaborada com uma camada de entrada com dois neurônios, sete camadas escondidas com vinte neurônios, uma camada de saída com dois neurônios e função de ativação *tanh*, que apresentou melhor comportamento frente as funções *relu*, *leaky-relu* e *sigmoide*. A Figura 4.1 (a) e 4.1 (b) apresentam o comparativo de regressão entre valores base e estimados para as variáveis f (frequência de ressonância), em GHz, e BW (largura de faixa), normalizado, que foram obtidos a partir do treinamento da rede neural com a biblioteca *Scikit-learn* da linguagem Python. Importante destacar que o eixo horizontal, variáveis alvo ordenadas, representam o conjunto de dados usadas no processo de teste posicionados apropriadamente em ordem crescente.

Para essa configuração, notou-se que o regressor identificou um padrão para as amostras de teste, porém não de maneira ideal, no qual seria a sobreposição entre os valores de base e os que foram estimados pela rede neural. Esse resultado passa a ser possível à medida em que o MSE tende a zero. Considerou-se que a indicação do valor de 0,0518, do MSE, garante a capacidade de generalização do modelo de aprendizagem de maneira confiável e eficiente para que o algoritmo de otimização inicie o processo de busca pelos objetivos estabelecidos.

4.1.2 Otimização do MPPD

O modelo substituto treinado, na Subseção 4.1.1 foi usado para a estruturação do processo de otimização multiobjetivo. Além disso, os algoritmos evolucionários

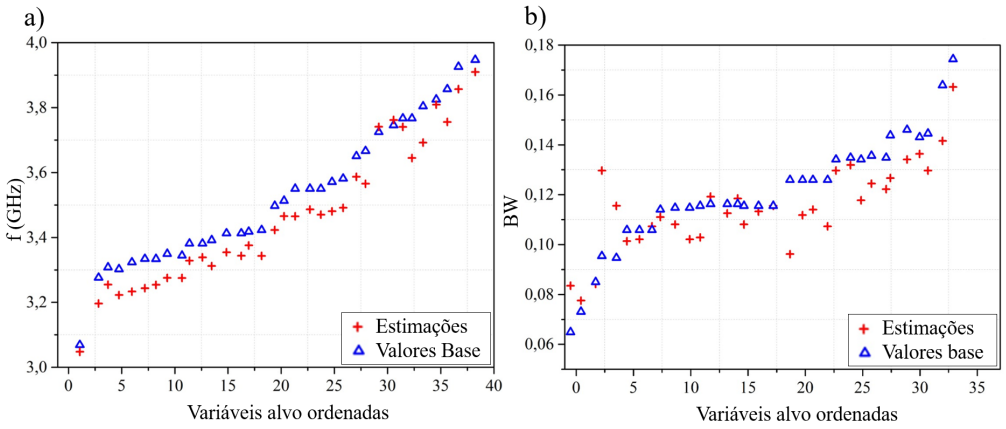


Figura 4.1: *Comparativo de regressão entre valores de base e estimados para as variáveis f (a) e BW (b).*

NSGA-II e MOEA/D foram aplicados em conjunto com o modelo substituto para a obtenção da função objetivo. Transformou-se a fronteira de Pareto para os valores originais das variáveis de saída, a fim de que o tomador de decisão (*Decision Maker*, DM) faça o uso das variáveis. É importante destacar que o estudo da convergência, utilizando os indicadores de qualidade de HV e a análise de Espalhamento e Espaçamento, foi feita em todos os algoritmos evolucionários na fronteira de Pareto da antena.

Para cada algoritmo, realizou-se a análise da convergência com o indicador HV normalizado considerando 30 simulações. Além disso, investigou-se, individualmente, os valores de HV, Espaçamento e Espalhamento, levando em conta 30 simulações. A Figura 4.2 (a) e 4.2 (b) mostram a curva de convergência normalizada ao longo da FP para os algoritmos NSGA-II e MOEA/D. Observou-se que o algoritmo NSGA-II teve a maior velocidade de convergência e apresentou comportamento estacionário depois de 10 iterações, aproximadamente. Já o MOEA/D teve uma curva de convergência mais lenta e com condição estacionária atingida próximo de 100 iterações. Apesar disso, ambos os algoritmos foram capazes de convergir para a fronteira de Pareto.

A Tabela 4.1 mostra os resultados da aplicação dos indicadores de qualidade para cada algoritmo evolucionário. Observou-se que ambos os algoritmos apresentaram, aproximadamente, o mesmo desempenho estatístico para os valores absolutos de HV, sendo que diferenças foram notadas apenas na quarta casa decimal. Por outro lado, para as métricas de Espaçamento e Espalhamento, em que valores baixos são desejados, o algoritmo NSGA-II apresentou-se superior. Assim, os pares entrada-saída do algoritmo NSGA-II serão usados na próxima seção.

A Figura 4.3 (a) e 4.3 (b) apresentam as aproximações da fronteira de Pareto para a antena MPPD com NSGA-II e MOEA/D respectivamente. Pode-se observar que,

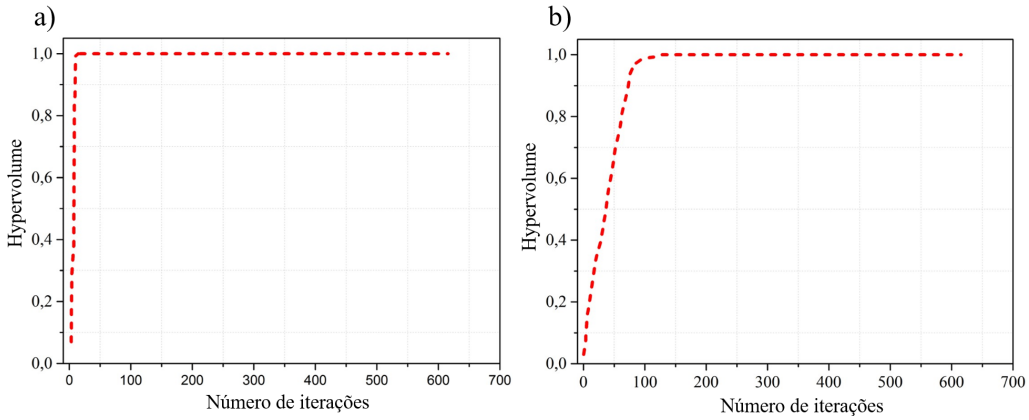


Figura 4.2: *Curva de convergência normalizada ao longo da FP para os algoritmos NSGA-II (a) e MOEA/D (b).*

Tabela 4.1: *Indicador de qualidade para a antena MPPD.*

	Hypervolume	***Espaçamento***	***Espalhamento***
NSGA-II	0,095(±0,001)	**0,017(±0,004)**	**0,553(±0,202)**
MOEA/D	0,095(±0,001)	0,123(±0,121)	1,120(±0,047)

para esse tipo de otimização, existe uma aproximação para a FP de cada algoritmo. A construção do problema fez com que os algoritmos não conseguissem explorar toda a superfície da fronteira de Pareto, obtendo baixa cardinalidade. Notam-se pontos espaçados, porém com regiões de aglutinação que não levam à formação da curva a fim de representar a fronteira de Pareto. Esse tipo de solução é caracterizada como de pouca representatividade, uma vez que não há um conjunto diversificado de soluções para cobrir a fronteira de Pareto em toda a sua extensão. Ainda assim é um modelo válido, com a ressalva de possuir poucas soluções.

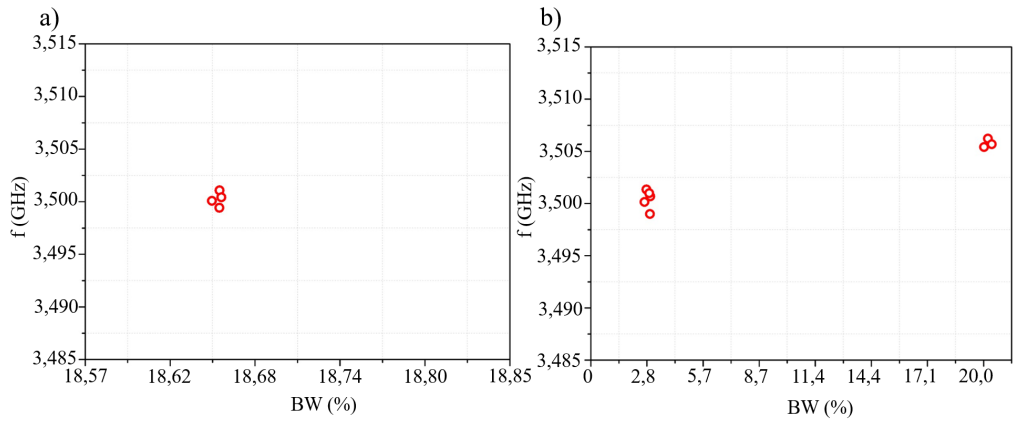


Figura 4.3: *Aproximações da fronteira de Pareto para a antena MPPD com NSGA-II (a) e MOEA/D (b).*

4.1.3 Validação Numérica do MPPD

O intuito desse primeiro estudo é a validação da metodologia proposta, atingir o menor erro possível em torno da frequência de ressonância de 3,5 GHz e maximizar a largura de faixa da antena MPPD. O algoritmo NSGA-II busca o comprimento e espessura ótimos para que a antena tenha um comportamento puramente resistivo centrado na frequência de 3,5 GHz e o maior intervalo de frequências em que a perda por retorno é menor que -10 dB, que resulta em menos de 10% da potência sendo refletida na fonte. A solução ótima estimada pelo algoritmo NSGA-II foi L = 13,41 mm, o qual está diretamente relacionado com o ponto de ressonância da antena, e t = 2,02 mm, que impacta diretamente no comportamento reativo da estrutura, tais valores resultaram nas seguintes variáveis estimadas de saída: frequência de ressonância centrado em 3,53 GHz com uma largura de faixa igual a 0,63 GHz (banda percentual de 18%). Para o processo de validação, os valores das variáveis de entrada estimadas foram inseridos no ANSYS HFSS.

A MPPD foi desenvolvida utilizando três estratégias diferentes avaliadas no HFSS, com o objetivo de analisar o desempenho da metodologia proposta. A Tabela 4.2 mostra as dimensões gerais de cada modelo. Os parâmetros L_d, D_d e g são os valores de comprimento, largura e espaçamento entre os elementos do dipolo respectivamente. Inicialmente, o conceito teórico para o desenvolvimento de um dipolo convencional, juntamente com as equações [86] foram aplicados direta e exclusivamente para a criação do modelo inicial no HFSS. O segundo modelo corresponde ao desenvolvimento de um Engenheiro de RF, que otimiza a estrutura teórica, alterando suas dimensões, a fim de melhorar o desempenho da antena. Por fim, o terceiro tipo provêm dos resultados numéricos obtidos pelo algoritmo de otimização, baseado no modelo substituto treinado, seguindo a metodologia desenvolvida.

Tabela 4.2: *Dimensões gerais dos modelos MPPD.*

Antenas MPPD	L_d **(mm)**	D_d **(mm)**	**g (mm)**
Abordagem teórica	10,3	1,0	3,0
Engenheiro de RF	13,8	2,8	2,0
Ferramenta baseada em IA	13,4	2,02	1,0

A Figura 4.4 apresenta os resultados das estratégias mencionadas. O modelo teórico foi alimentado por uma porta casada em 73 Ω e reportou uma frequência de ressonância em 3,6 GHz. Esse modelo gastou cerca de 30 minutos para ser formulado, desenhado e simulado. Considerando a antena desenvolvida pelo Engenheiro de RF, foram gastos cerca de 1 hora e 30 minutos, considerando o tempo do modelo teórico e mais os ajustes realizados para a otimização e a simulação. Levando em conta a técnica de

IA, foram gastos 53 segundos para a geração otimizada do par entrada-saída entregue pelo algoritmo de otimização. Para que fosse possível validar a estrutura estimada, um modelo simulado foi gerado. Tal modelo foi contruído partindo da estrutura da abordagem teórica e levou cerca de 2 minutos para ser simulado pelo HFSS. Totalizando 32 minutos e 53 segundos. É possível notar que a frequência de ressonância e largura de faixa estão de acordo com os valores estimados, validando a metodologia proposta. No entanto, há uma pequena diferença no valor da largura de faixa devido às variações do MSE no regressor, o que pode ser reduzido com a adição de mais pares entrada-saída na rede neural e o posterior treinamento. A Tabela 4.3 apresenta um resumo do consumo de tempo de cada estratégia.

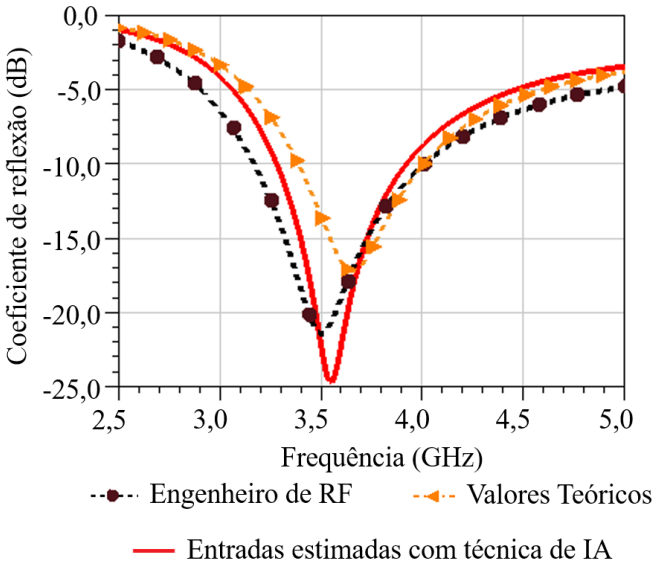


Figura 4.4: *Resultados simulados da antena MPPD considerando as três estratégias de verificação.*

Tabela 4.3: *Resumo do consumo de tempo de cada estratégia para a antena MPPD.*

Estratégia	***Tempo de Desenvolvimento***	
Abordagem teórica	30 minutos	
Engenheiro de RF	1 horas e 30 minutos	
Técnica de IA	***Processamento***	***Validação***
	53 segundos	32 minutos

4.2 MSPD - Dipolo Impresso com Estrutura de Casamento de Impedância

Este estudo de caso é uma variação do modelo anterior, com a inserção de uma estrutura de casamento de impedância a fim de se obter um protótipo real da antena proposta. A construção do modelo substituto e a inserção dos algoritmos multiobjetivo

têm por finalidade minimizar o erro ao redor da frequência de ressonância de 3,5 GHz e maximizar a largura de faixa da antena MSPD. Nas próximas subseções, serão apresentados: a estrutura da rede neural usada para a construção do modelo substituto; o processo de otimização com a definição do melhor algoritmo evolucionário, baseado nas métricas dos indicadores de qualidade; e os resultados simulado e medido da MSPD.

4.2.1 Rede Neural do MSPD

A rede neural do tipo MLP, para a antena MSPD, foi composta por 243 amostras de pares entrada-saída, que foram obtidos de maneira empírica com o aumento progressivo na quantidade de amostras a fim de buscar a melhor relação entre pares entrada-saída que represente a configuração da antena MSPD à rede neural. A base de dados foi então dividida em 75% para a composição do treinamento e 25% para o teste do modelo de aprendizagem. Para a definição dos hiperparâmetros que compõem a rede neural, utilizaram-se os métodos k-*fold* com validação cruzada e *GridSearchCV*, de forma a garantir que os efeitos de subajuste ou o sobreajuste não ocorram.

O processo de treinamento consumiu 6 minutos e obteve-se um erro quadrático médio MSE = 0,0087, com elevada capacidade de generalização. A rede neural foi construída com uma camada de entrada com três neurônios, oito camadas escondidas com cem neurônios, uma camada de saída com dois neurônios e função de ativação do tipo *tanh*, a qual apresentou o melhor comportamento frente as funções *relu*, *leaky-relu* e *sigmoide*. A Figura 4.5 (a) e 4.5 (b) mostram o comparativo de regressão entre valores base e estimados para as variáveis *BW*, normalizado, e *f*, em GHz, obtidos a partir do treinamento da rede neural com a biblioteca *Scikit-learn* da linguagem Python. O eixo horizontal, entitulado como variáveis alvo ordenadas, representa o conjunto variáveis de testes posicionados em ordem crescente.

Nesse estudo, a rede neural atingiu uma capacidade de estimação muito próxima do ideal, com boa parte dos pontos estimados sobrepostos aos valores base, o que garante o erro quadrático médio um valor próximo de zero, representando a alta capacidade de generalização do modelo. Sendo assim, é aceitável que processo de otimização seja iniciado em busca dos objetivos definidos.

4.2.2 Otimização do MSPD

Aplicando-se a rede neural da Subseção 4.2.1, treinou-se o modelo substituto, o qual foi usado para a estruturação do processo de otimização multiobjetivo. Além disso, os algoritmos evolucionários NSGA-II e MOEA/D foram aplicados em conjunto com o modelo substituto para a obtenção da função objetivo desejada. Transformou-se

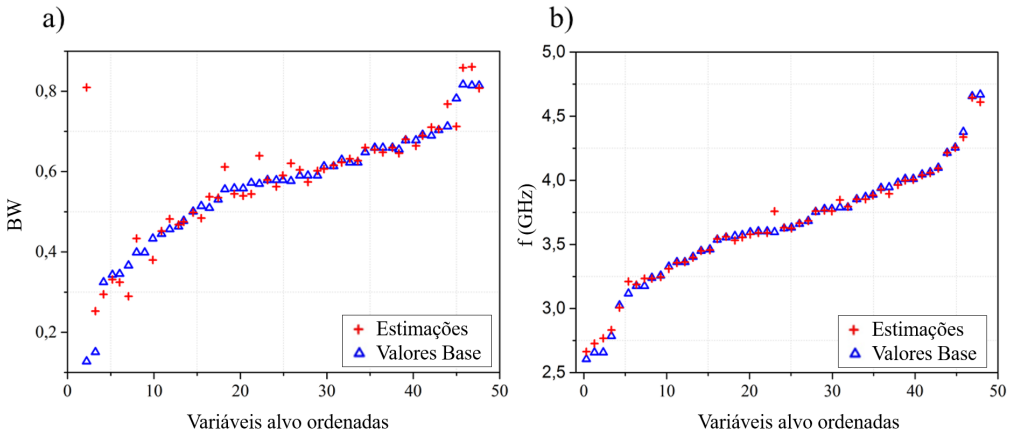


Figura 4.5: *Comparativo de regressão entre valores de base e estimados para as variáveis BW (a) e f (b).*

a fronteira de Pareto para os valores originais das variáveis de saída, a fim de que o tomador de decisão faça o uso das variáveis. É importante destacar que o estudo da convergência utilizando os indicadores de qualidade de HV e a análise de Espalhamento e Espaçamento, foi feito em todos os algoritmos evolucionários na fronteira de Pareto da antena.

Para cada algoritmo, realizou-se a análise da convergência com o indicador HV normalizado considerando 30 simulações. Além disso, investigou-se individualmente os valores de HV, Espaçamento e Espalhamennto levando em conta 30 simulações. A Figura 4.6 (a) e 4.6 (b) mostram a curva de convergência normalizada ao longo da FP para os algoritmos NSGA-II e MOEA/D. Observou-se que o algoritmo NSGA-II teve a maior velocidade de convergência e apresentou comportamento estacionário depois de 20 iterações aproximadamente. Já o MOEA/D teve uma curva de convergência mais lenta e com condição estacionária atingida próxima de 650 iterações. Apesar disso, ambos os algoritmos foram capazes de convergir para a fronteira de Pareto.

A Tabela 4.4 mostra os resultados da aplicação dos indicadores de qualidade para cada algoritmo evolucionário. Observou-se que os dois algoritmos apresentaram aproximadamente o mesmo desempenho estatístico para os valores absolutos de HV, sendo que diferenças foram notadas apenas na terceira casa decimal. Por outro lado, para as métricas de Espaçamento e Espalhamento, em que valores baixos são desejados, o algoritmo NSGA-II apresentou-se superior.

Tabela 4.4: *Indicador de qualidade para a antena MSPD.*

	Hypervolume	***Espaçamento***	***Espalhamento***
NSGA-II	**0,717(±0,000)**	**0,020(±0,002)**	**0,436(±0,087)**
MOEA/D	0,710(±0,007)	0,058(±0.035)	0,698(±0,132)

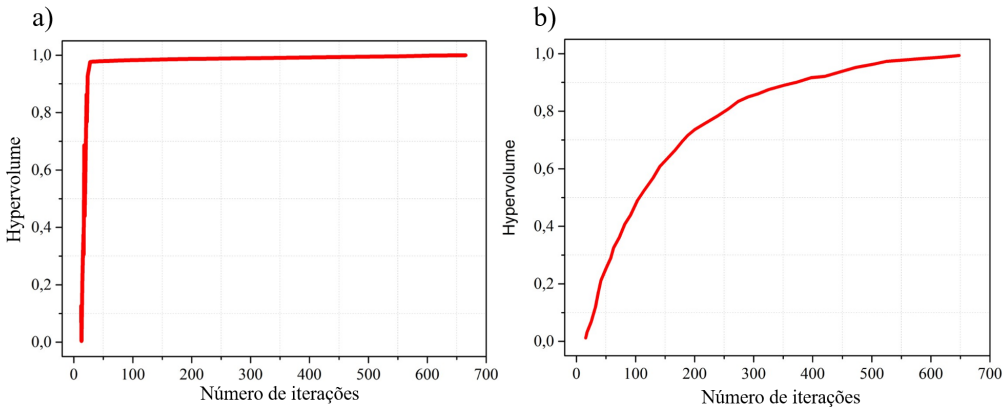


Figura 4.6: *Curva de convergêcnia normalizada ao longo da FP para os algoritmos NSGA-II (a) e MOEA/D (b).*

A Figura 4.7 (a) e 4.7 (b) apresentam as aproximações da fronteira de Pareto para a antena MSPD com NSGA-II e MOEA/D respectivamente. Pode-se observar que, para esse tipo de otimização existe uma aproximação para a FP de cada algoritmo. As soluções encontradas pelo NSGA-II apresentaram um conjunto de pontos discretos, minimamente espaçados e sem a concentração de amostras em uma única região. Isso fez com que menores valores de Espaçamento e Espalhamento fossem atingidos, uma vez que as soluções para esses indicadores são baseadas nas distâncias. Assim, levando em conta os indicadores de qualidade, o algoritmo a ser explorado para a próxima Seção experimental será o NSGA-II.

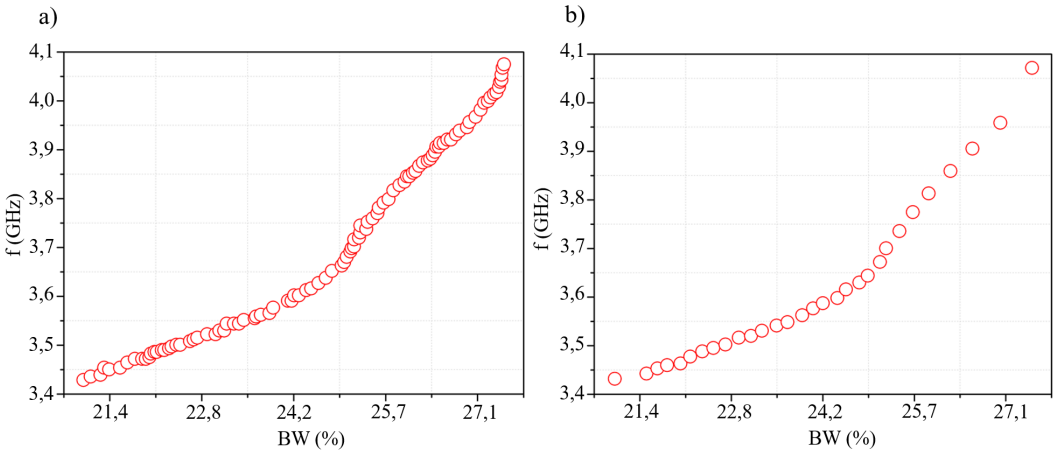


Figura 4.7: *Aproximações da fronteira de Pareto para a antena MSPD com NSGA-II (a) e MOEA/D (b).*

4.2.3 Protótipo e Resultados Experimentais do MSPD

O objetivo deste estudo de caso é obter o menor erro possível em torno de 3,5 GHz e maximizar a largura de faixa da antena MSPD. A solução ótima estimada pelo algo-

ritmo NSGA-II foi L = 23,33 mm, t = 4,02 mm e S = 2.19 mm, que resulta nas seguintes variáveis estimadas de saída: frequência de ressonância centrado em 3,5 GHz com uma largura de faixa igual a 0,75 GHz (banda percentual de 21,4%). Para o processo de validação, os valores das variáveis de entrada estimadas pelo algoritmo de otimização foram inseridos no ANSYS HFSS. Observou-se que o ponto de menor ressonância está centrado em 3,5 GHz e a largura de faixa obtida foi de 0,77 GHz, muito próximo ao valor estimado. Além disso, realizou-se a fabricação e a caracterização da antena MSPD seguindo as dimensões estimadas. A Figura 4.8 (a) apresenta o protótipo construído e a Figura 4.8 (b) reporta o coeficiente de reflexão simulado e medido, utilizando o analisador de rede vetorial *PNA Network Analyzer* N5224A da empresa Keysight. Houve uma ótima concordância entre os valores simulados e medidos, sendo que algumas pequenas diferenças ocorreram devido a imprecisões no processo de fabricação. Além disso, a mínima divergência no valor da largura de faixa é devido às variações do MSE no regressor, o que pode ser reduzido com a adição de mais pares entrada-saída na rede neural e o refinamento do modelo.

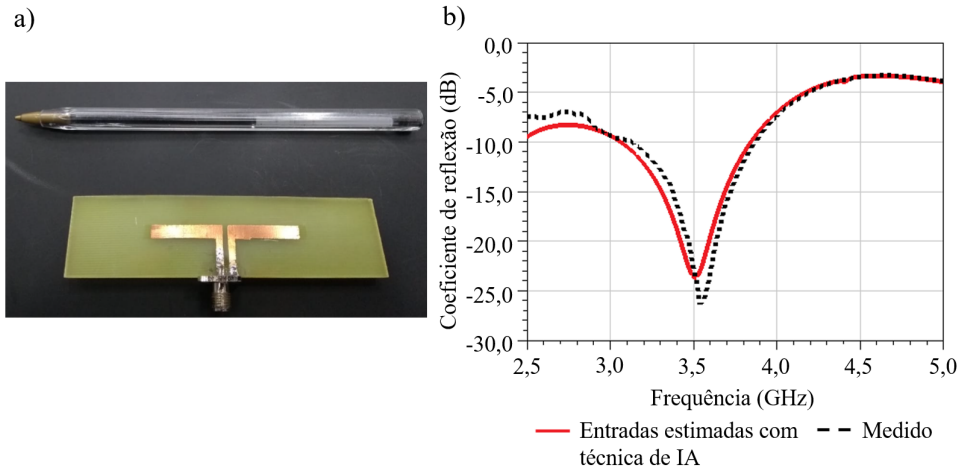


Figura 4.8: *Protótipo da antena MSPD (a) e seu resultado simulado e medido em termos de coeficiente de reflexão (b).*

A MSPD foi desenvolvida utilizando-se três estratégias diferentes, avaliadas no ANSYS HFSS. A Tabela 4.5 mostra as dimensões gerais de cada modelo. Os parâmetros L, W_d e g_d são os valores de comprimento, largura e *gap* entre os elementos do dipolo respectivamente. Enquanto que L_a e W_a são o comprimento e a largura da linha de alimentação. Primeiramente, as equações da antena dipolo convencional foram aplicadas direta e exclusivamente para a criação do modelo inicial no HFSS [86]. O segundo modelo corresponde ao desenvolvimento de um Engenheiro de RF. Por fim, o último tipo provêm dos resultados numéricos obtidos pelo algoritmo de otimização, NSGA-II, baseado no modelo substituto treinado.

A Figura 4.9 apresenta os resultados da antena MSPD considerando as três es-

Tabela 4.5: *Dimensões gerais dos modelos MSPD.*

Antenas MSPD	***L (mm)***	W_d ***(mm)***	g_d ***(mm)***	L_m ***(mm)***	W_m ***(mm)***
Abordagem Teórica	1,8	3,0	1,0	-	-
Engenheiro de RF	18,4	3,5	2,0	15,0	2,0
Ferramenta baseada em IA	23,3	4,0	1,0	11,5	2,2

tratégias mencionadas, além dos valores medidos do protótipo da antena. O dipolo de onda completa foi alimentado por uma porta casada em 200 Ω, o qual é o valor típico de impedância de entrada para um dipolo de onda completa quando alimentado em seu ponto de máxima corrente [86], e reportou uma frequência de ressonância em 3,4 GHz. O modelo gastou cerca de 50 minutos para ser formulado, desenhado e simulado. Considerando a antena desenvolvida pelo Engenheiro de RF, consumiu-se 4 horas, das quais 50 minutos são referentes ao projeto teórico e mais 3 horas e 10 minutos até a concepção do modelo otimizado final. Para esse protótipo, o Engenheiro adicionou uma estrutura de casamento de impedância para eliminar a parcela reativa a fim de melhorar o desempenho da antena. Por fim, considerando a técnica de IA, foram consumidos 5 segundos para que o par entrada-saída fosse fornecido pelo algoritmo de otimização, sendo esse o tempo de processamento. Para que fosse possível validar o modelo estimado, um modelo simulado foi gerado. Ele foi obtido partindo da abordagem teórica e levou cerca de 5 minutos para ser simulado pelo HFSS, totalizando 55 minutos e 5 segundos. A Tabela 4.6 apresenta um resumo do consumo de tempo de cada estratégia.

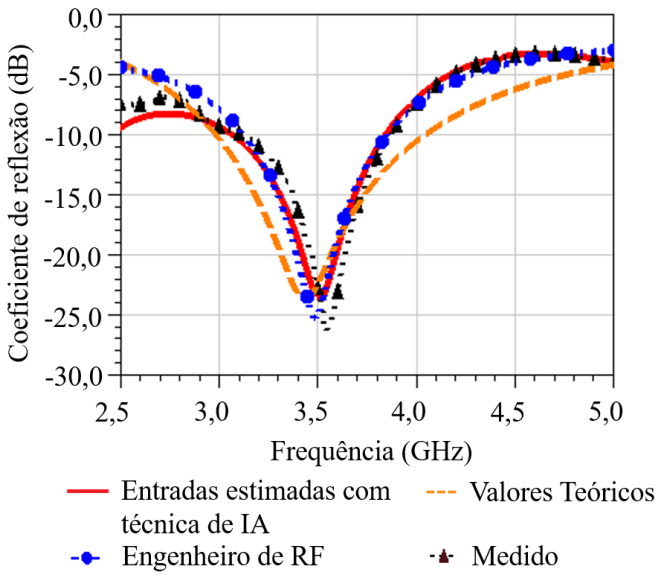


Figura 4.9: *Comparativo de desempenho para a antena MSPD em quatro diferentes cenários.*

Tabela 4.6: *Resumo do consumo de tempo de cada estratégia para a antena MSPD.*

Estratégia	***Tempo de Desenvolvimento***	
Abordagem teórica	40 minutos	
Engenheiro de RF	4 horas	
Técnica de IA	***Processamento***	***Validação***
	5 segundos	55 minutos

Seguindo o processo de caracterização da antena, o diagrama de radiação da MSPD foi medido utilizando os seguintes equipamentos: um *Field Fox Microwave Analyzer* N9952A; duas antenas Hyperlog, da Aaronia; um *BTS Master Anritsu MT8222B*. A antena Hyperlog foi responsável por transmitir um sinal escalar na frequência desejada e, na recepção, a antena MSPD foi girada em torno do seu eixo para medição de potência nos planos de azimute e elevação. As medidas foram realizadas em um ambiente *indoor* amplo. A Figura 4.10 apresenta o cenário utilizado para a caracterização do diagrama de radiação da antena. A fim de avaliar o comportamento do diagrama, as medidas foram realizadas apenas na frequência central de 3,5 GHz. Os resultados simulados e medidos são apresentados na Figura 4.11. Em linhas gerais, o diagrama de radiação da antena MSPD foi considerado de acordo com o esperado para uma antena do tipo dipolo, no qual apresenta um ganho de 2,3 dBi, uma baixa relação frente costas, abertura de feixe, em elevação, de 75º e comportamento omnidirecional em azimute.

Figura 4.10: *Cenário utilizado para a caracterização do diagrama de radiação da antena.*

4.3 QY - Quasi-Yagi

A elaboração do modelo substituto e a inserção de algoritmos multiobjetivos, tem por finalidade definir a melhor dimensão da estrutura de casamento de impedância a

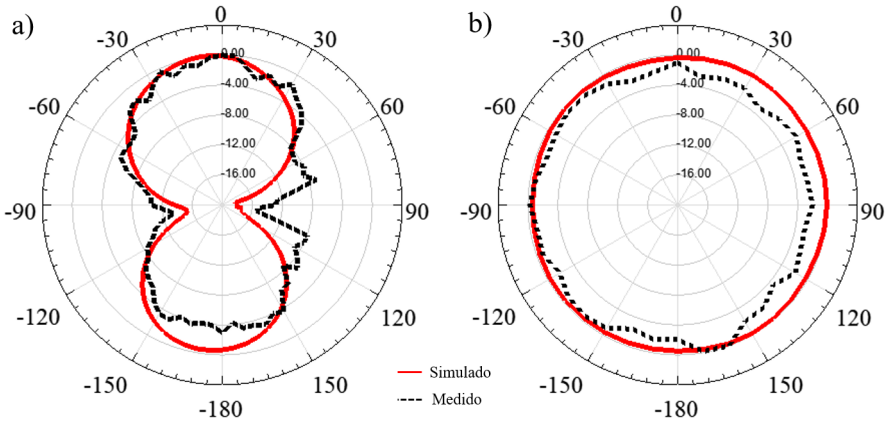


Figura 4.11: *Diagrama de radiação em elevação (a) e azimute (b).*

qual minimize, simultaneamente, o casamento de impedância em 1,9, 2,6 e 3,5 GHz. Nas próximas subseções, serão apresentados: a estrutura da rede neural usada para a construção do modelo substituto; o processo de otimização com a definição do melhor algoritmo evolucionário, baseado nas métricas dos indicadores de qualidades; e os resultados simulados e medidos da antena.

4.3.1 Rede Neural da QY

A rede neural do tipo MLP, para a antena QY, foi composta por 1364 amostras, que foram obtidos de maneira empírica com o aumento progressivo na quantidade de amostras a fim de buscar a melhor relação entre pares entrada-saída que represente a configuração da antena QY à rede neural. A base de dados foi então dividida entre conjuntos de treinamento e teste, para garantir a capacidade de generalização da rede. Para a definição dos hiperparâmetros que compõem a rede neural, utilizaram-se os métodos k-*fold* com validação cruzada e *GridSearchCV*, de forma a garantir que os efeitos de subajuste ou o sobreajuste não ocorram.

O processo de treinamento consumiu 19 segundos e obteve-se um erro quadrático médio, MSE, igual a 1,44, com capacidade de generalização aceitável para o modelo. A rede neural foi construída com uma camada de entrada com dois neurônios; duas camadas escondidas, sendo a primeira composta por 32 neurônios e, a segunda, por 256 neurônios; uma camada de saída com três neurônios e função de ativação *relu*, a qual teve melhor desempenho frente as funções *tanh*, *leaky-relu* e *sigmoide*. A Figura 4.12 mostra o comparativo de regressão entre os valores base e estimados para as variáveis f_1, f_2 e f_3, em dB, respectivamente.

Observou-se que a rede neural obteve a capacidade de predição, visto que uma grande quantidade de pontos estimados convergiram com os valores base utilizados.

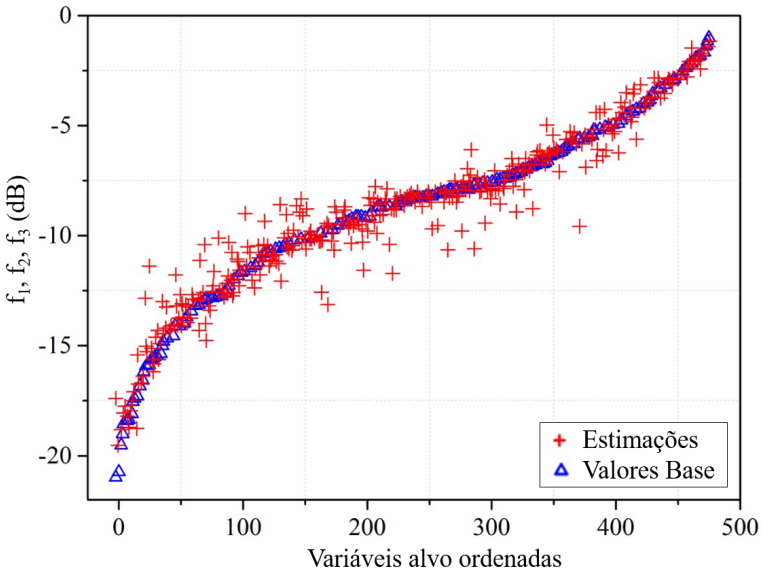


Figura 4.12: *Comparativo de regressão entre valores de base e estimados para as variáveis correspondentes ao coeficiente de reflexão nas frequências de interesse f_1, f_2 e f_3, em dB, respectivamente.*

Entretanto, devido à complexidade do modelo em análise e o número de funções objetivo, houve uma grande quantidade de pontos estimados sem a sobreposição dos valores de base, porém muito próximos. Dessa forma, considerou-se que a indicação do valor do MSE igual a 1,44 garante a capacidade de generalização do modelo de aprendizagem de forma confiável e é aceitável que o processo de otimização seja iniciado em busca dos objetivos que foram propostos para o modelo.

4.3.2 Otimização da QY

Aplicando-se a rede neural da Subseção 4.3.1, treinou-se o modelo substituto, o qual foi usado para a estruturação do processo de otimização multiobjetivo. Além disso, os algoritmos evolucionários NSGA-II e MOEA/D foram aplicados em conjunto com o modelo substituto para a obtenção da função objetivo. Transformou-se a fronteira de Pareto para os valores originais das variáveis de saída, a fim de que o tomador de decisão faça o uso das variáveis. É importante destacar que o estudo da convergência utilizando os indicadores de qualidade de HV e análise de Espalhamento e Espaçamento foi feito em todos os algoritmos evolucionários na FP da antena.

Realizou-se a análise da convergência com o indicador HV normalizado considerando 30 simulações. Além disso, investigou-se, individualmente, os valores de Hypervolume, Espaçamento e Espalhamento, levando em conta 30 simulações. A Figura 4.13 (a) e a Figura 4.13 (b) mostram a curva de convergência normalizada ao longo

da FP para os algoritmos NSGA-II e MOEA/D. Nota-se que o algoritmo NSGA-II, baseado em população, teve maior velocidade de convergência e apresentou comportamento estacionário após 25 iterações aproximadamente. Já o MOEA/D teve uma curva de convergência mais lenta, ou seja, o fato de o algoritmo ser baseado em decomposição fez com que houvesse a necessidade de mais iterações para que o algoritmo conseguisse apresentar sinais de convergência. Além disso, a condição estacionária foi atingida apenas após 600 iterações. Contudo, ambos os algoritmos foram capazes de convergir para a fronteira de Pareto.

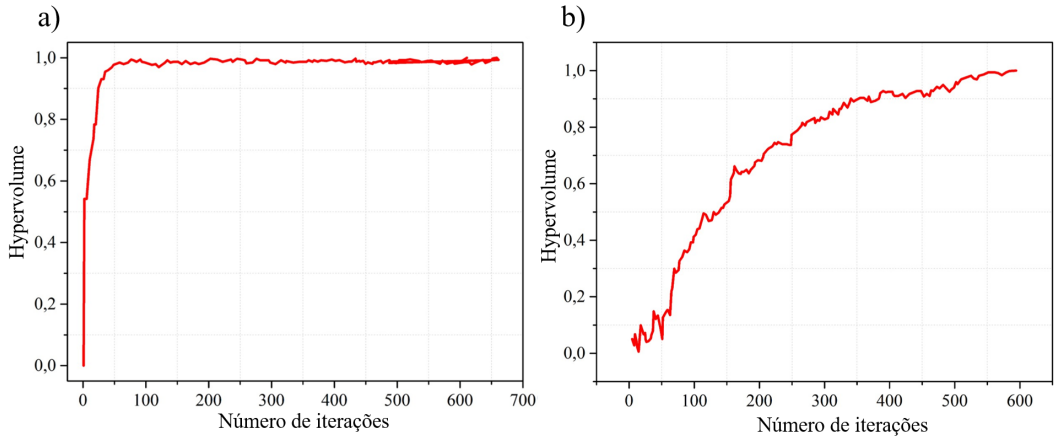


Figura 4.13: *Curva de convergência normalizada ao longo da FP para os algoritmos NSGA-II (a) e MOEA/D (b).*

A Tabela 4.7 mostra os resultados da aplicação dos indicadores de qualidade para cada algoritmo evolucionário. Observou-se que o algoritmo NSGA-II teve um valor de HV maior que o algoritmo MOEA/D. Isso significa que um espaço objetivo maior é coberto pela FP desse algoritmo. Além disso, para as métricas de Espaçamento e Espalhamento, em que valores baixos são desejados, o algoritmo NSGA-II teve desempenho superior.

Tabela 4.7: *Indicador de qualidade para a antena QY.*

	Hypervolume	***Espaçamento***	***Espalhamento***
NSGA-II	**0,591(±0,286)**	**0,055(±0,010)**	**0,456(±0,026)**
MOEA/D	0,498(±0,358)	0,049(±0,009)	0,544(±0,060)

A Figura 4.14 (a) e a Figura 4.14 (b) apresentam as aproximações das fronteira de Pareto para a antena QY com NSGA-II e MOEA/D respectivamente. Pode-se observar que, para esse tipo de otimização, existe uma aproximação para a FP de cada algoritmo. As soluções encontradas pelo NSGA-II apresentaram um conjunto de pontos discretos, minimamente espaçados e sem a concentração de amostras em uma única região. Isso

faz com que menores valores de Espaçamento e Espalhamento sejam atingidos, uma vez que as soluções para esses indicadores são baseadas nas distâncias. Assim, levando em conta os indicadores de qualidade, o algoritmo a ser explorado para a Seção de experimentos será o NSGA-II.

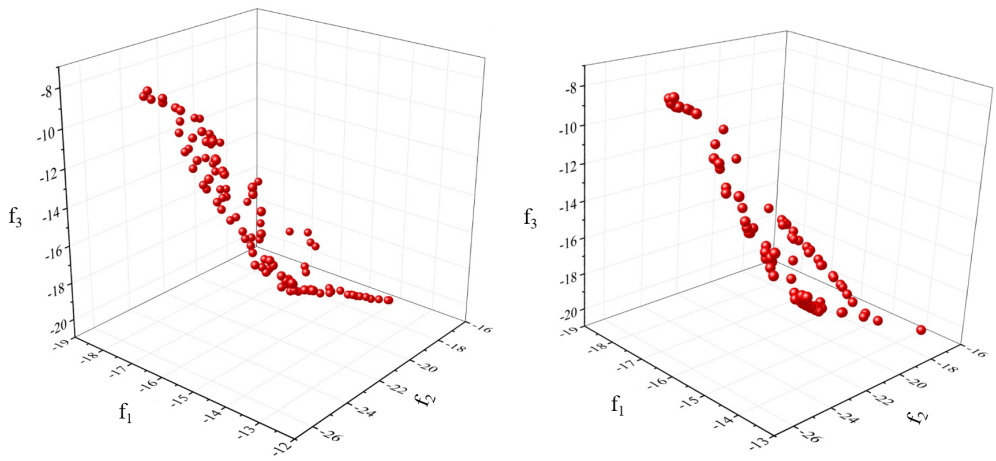


Figura 4.14: *Aproximações da fronteira de Pareto para a antena QY com NSGA-II (a) e MOEA/D (b).*

4.3.3 Protótipo e Resultados Experimentais da QY

A Tabela 4.8 apresenta o conjunto de soluções ótimas estimadas pelo NSGA-II e o conjunto com suas respectivas validações no ANSYS HFSS. Todas as saídas propostas são válidas do ponto de vista do coeficiente de reflexão estar abaixo de -10 dB, porém existem discrepâncias entre as amostras estimadas e simuladas. Isso devido ao fato de o processo regressivo ainda não ser o ideal, porém apresenta qualidade boa o suficiente para conseguir estimar as amostras de saída. Assim, qualquer conjunto de amostras escolhido é aplicável do ponto de vista de validação da metodologia proposta, que é o objetivo desta análise. Por outro lado, como o intuito desse estudo é utilizar do menor valor possível do coeficiente de reflexão, o conjunto ótimo de pares entrada-saída escolhido foi o referente à amostra 8 da Tabela 4.8, com uma solução ótima de entrada L_5 = 15,37 mm e L_p = 13,62 mm que resultaram nas seguintes variáveis simuladas de saída: f_1 = -17,67 dB, f_2 = -28,54 dB e f_3 = -39,25 dB. Para chegar a essa conclusão todas as amostras selecionadas pelo algoritmo NSGA-II foram analisadas e simuladas.

Em continuidade ao processo de validação, realizou-se a fabricação e a caracterização da antena QY seguindo as dimensões estimadas. A Figura 4.15 (a) apresenta o protótipo construído e a Figura 4.15 (b) reporta o coeficiente de reflexão simulado e medido, utilizando o analisador de rede vetorial *PNA Network Analyzer* N5224A. A

Tabela 4.8: *Conjunto de soluções ótimas estimadas pelo algoritmo NSGA-II e simuladas no ANSYS HFSS.*

Amostra	***Variáveis de entrada (mm)***		***Saídas estimadas (dB)***			***Saídas Simuladas HFSS (dB)***		
-	L_5	L_p	f_1	f_2	f_3	f_1	f_2	f_3
1	16,18	15,58	-13,77	-17,40	-20,25	-15,00	-20,30	-19,80
2	15,97	15,34	-14,11	-17,91	-20,15	-14,80	-21,20	-18,75
3	16,15	14,36	-16,20	-19,13	-17,03	-17,39	-22,00	-26,22
4	15,65	13,41	-16,73	-22,26	-14,17	-19,74	-25,75	-17,86
5	14,42	13,53	-13,42	-24,81	-15,29	-14,42	-23,86	-13,88
6	15,72	14,80	-14,70	-19,39	-19,38	-16,10	-21,88	-20,77
7	15,42	13,25	-16,30	-23,32	-13,71	-18,44	-26,89	-18,47
8	**15,37**	**13,62**	**-15,78**	**-22,66**	**-15,62**	**-17,67**	**-28,54**	**-39,25**
9	15,12	14,29	-14,46	-21,99	-17,94	-15,10	-24,40	-15,30
10	14,70	13,87	-14,04	-23,82	-16,52	-14,82	-25,74	-18,80
11	15,34	13,41	-15,98	-23,15	-14,56	-19,74	-25,75	-17,86
12	15,26	13,66	-15,40	-22,94	-16,01	-17,00	-27,40	-26,40
13	14,80	13,35	-14,52	-24,42	-15,42	-16,73	-27,66	-16,89
14	15,22	13,82	-15,04	-22,77	-16,86	-16,46	-27,80	-31,65
15	14,49	13,67	-13,54	-24,54	-15,67	-14,34	-25,60	-14,93
16	15,97	15,14	-14,32	-18,24	-19,97	-16,39	-22,30	-19,48

simulação numérica resultou em três pontos de ressonância em torno das frequências de interesse, validando a metodologia proposta. No entanto, há uma discrepância dos valores de coeficiente de reflexão estimados e simulados pelo HFSS. Tais discrepâncias ocorreram devido a dois fatores principais. Primeiro, o algoritmo de otimização não conseguiu estimar com precisão os valores dos coeficientes de reflexão devido ao fato de o processo regressivo ainda não ser o ideal. Tais divergências podem ser minimizadas com o incremento de amostras à base de dados, bem como o refinamento da rede e, até mesmo, a escolha de outros modelos de aprendizagem. O segundo fator leva em conta imprecisões no processo de fabricação.

Para o desenvolvimento da antena QY, implementou-se três estratégicas diferentes avaliadas no ANSYS HFSS, com o objetivo de analisar o desempenho da metododologia proposta. A Tabela 4.9 mostra as dimensões gerais de cada modelo. Os parâmetros L_1, L_2, L_3 e L_4 tiveram suas dimensões físicas mantidas, as quais seguem a teoria de antenas [86]. Enquanto que L_5 e L_p são as dimensões da estrutura de casamento. Inicialmente, um modelo seguindo as dimensões de uma estrutura teórica de casamento de impedância, segundo [86], foi projetado. O segundo modelo corresponde ao desenvolvimento de um Engenheiro de RF, especialista no desenvolvimento de antenas. Por fim, o terceiro modelo provêm dos resultados numéricos obtidos pelo algoritmo de

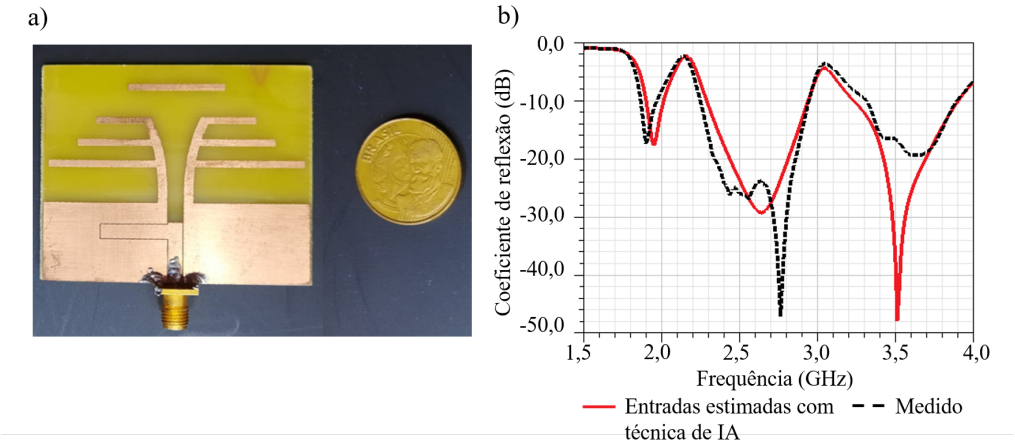


Figura 4.15: *Protótipo da antena QY (a) e seu resultado simulado e medido em termos de coeficiente de reflexão (b).*

otimização, NSGA-II, o qual se baseou no modelo substituto treinado.

Tabela 4.9: *Dimensões gerais dos modelos da antena QY.*

Antena QY	L_5 (mm)	L_p (mm)	L_1 (mm)	L_2 (mm)	L_3 (mm)	L_4 (mm)
Abordagem Teórica	17,0	20,7	12,6	17,0	24,5	22,0
Engenheiro de RF	15,95	13,55	12,6	17,0	24,5	22,0
Ferramenta baseada em IA	15,37	13,62	12,6	17,0	24,5	22,0

A Figura 4.16 apresenta os resultados simulados, considerando os modelos teórico, do Engenheiro especialista de RF e as entradas estimadas com a técnica de IA. Nota-se que a simulação no HFSS, com as amostras de entrada estimadas, teve um comportamento muito semelhante em sua saída, à antena desenvolvida pelo Engenheiro de RF. Os pontos de nulo estão centrados nas mesmas frequências para as duas primeiras faixas e, para a terceira banda houve a presença do vale para ambos os casos, porém com um pequeno deslocamento na frequência para a antena projetada pelo especialista. Ao compararmos os modelos mencionados com a abordagem teórica da antena QY, observa-se uma considerável melhora, em termos de minimização do coeficiente de reflexão, para as duas frequências mais altas e o ajuste no ponto de ressonância em torno de 1,9 GHz.

Em termos de tempo computacional para o desenvolvimento de cada estratégia, a abordagem teórica levou cerca de 3 horas para o modelo ser formulado, desenhado e simulado. Levando em conta a antena desenvolvida pelo Engenheiro de RF, foram expedidas 10 horas, das quais 3 horas são referentes ao projeto teórico e mais 7 horas para a etapa de otimização por parte do especialista. Considerando a técnica de IA, foram consumidos 2 minutos para que os valores de entrada e suas respectivas saídas fossem

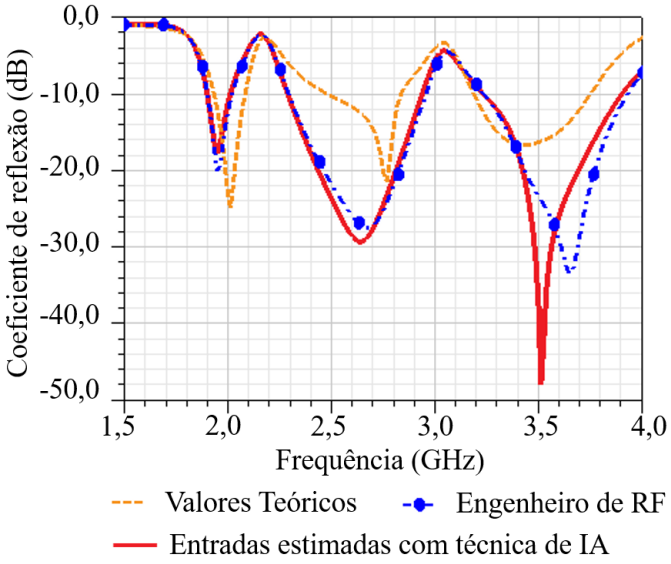


Figura 4.16: *Comparativo de desempenho para a antena QY em três diferentes cenários.*

fornecidos pelo algoritmo de otimização, sendo esse o tempo de processamento. Para que fosse possível validar o modelo estimado, um modelo simulado foi gerado. Essa estrutura foi concebida a partir da abordagem teórica e levou cerca de 15 minutos para ser simulada pelo HFSS, totalizando o tempo de 3 horas e 15 minutos. No entanto, como foram validados cada um dos 15 pares entrada-saída restantes, que consumiram mais 3 horas e 45 minutos, o tempo total para o modelo gerado pela técnica de IA foi de 7 horas e 2 minutos. A Tabela 4.10 apresenta um resumo do consumo de tempo de cada estratégia para a antena QY.

Tabela 4.10: *Resumo do consumo de tempo de cada estratégia para a antena QY.*

Estratégia	***Tempo de Desenvolvimento***	
Abordagem teórica	3 horas	
Engenheiro de RF	10 horas	
Técnica de IA	***Processamento***	***Validação***
	2 minutos	7 horas

Seguindo o processo de caracterização da antena QY, o seu diagrama de radiação foi medido utilizando os seguintes equipamentos: um *Field Fox Microwave Analyzer* N9952A; duas antenas tipo Hyperlog, da Aaronia; um *BTS Master Anritsu MT8222B*. A antena Hyperlog foi responsável por radiar um sinal escalar nas frequências desejadas e, na recepção, a antena QY foi girada em torno do seu eixo para a medição de potência nos planos de azimute e elevação. As medidas foram realizadas em um ambiente *indoor* amplo. A Figura 4.17 apresenta o cenário utilizado para a caracterização do diagrama de radiação da antena.

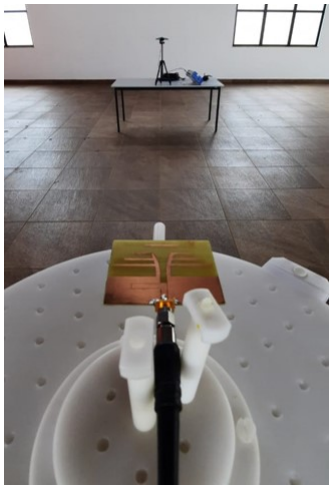


Figura 4.17: *Cenário utilizado para a caracterização do diagrama de radiação da antena QY.*

A fim de avaliar o diagrama de radiação da antena QY fabricada, as medidas foram realizadas nas frequências centrais de cada uma das três bandas. A comparação entre os diagramas de radiação simulado e medido em elevação e azimute em 1,9, 2,6 e 3,5 GHz estão representados na Figura 4.18. Em 1,9 GHz a antena tem um ganho de 3,5 dBi, abertura de feixe Φ_{ab} e θ_{ab} de 95º e 140º e relação frente-costa (*Front-to-Back Ratio*, RFC) de 3 dB. O diagrama de radiação da antena QY apresenta boa concordância entre os resultados simulados e medidos. Entretanto, em azimute, reportou baixa relação frente-costa. Isso pode ser explicado, devido ao fato da pequena dimensão do elemento refletor comparado ao comprimento de onda para a frequência de 1,9 GHz [86]. A melhora desse comportamento, ou seja, aumento da relação frente-costa é minimizada conforme ocorre o aumento da frequência. Para a frequência de 2,6 GHz o ganho medido foi de 4,4 dBi, com abertura de feixe Φ_{ab} e θ_{ab} ambos de 80º e RFC de 12 dB. Por fim, em 3,5 GHz, o ganho medido foi de 6,3 dBi, com Φ_{ab} e θ_{ab} de 60º e 90º e RFC de 13 dB.

Tabela 4.11: *Resultados obtidos para a antena QY nas frequências de 1,9, 2,6 e 3,5 GHz.*

Frequência (GHz)	***Ganho (dBi)***	***Abertura de feixe (º)***	
		Φ_{ab}	Θ_{ab}
1,9	3,5	95	140
2,6	4,4	80	80
3,5	6,3	60	90

Após os resultados obtidos, realizou-se um comparativo das propriedades eletromagnéticas com outras antenas do tipo Quasi-Yagi presentes na literatura para demonstrar que a antena desenvolvida com técnicas de Inteligência Computacional tem o comportamento conforme o esperado. A comparação foi feita em termos das seguintes características: largura de faixa; ganho; relação frente-costas; eficiencia de radiação; ca-

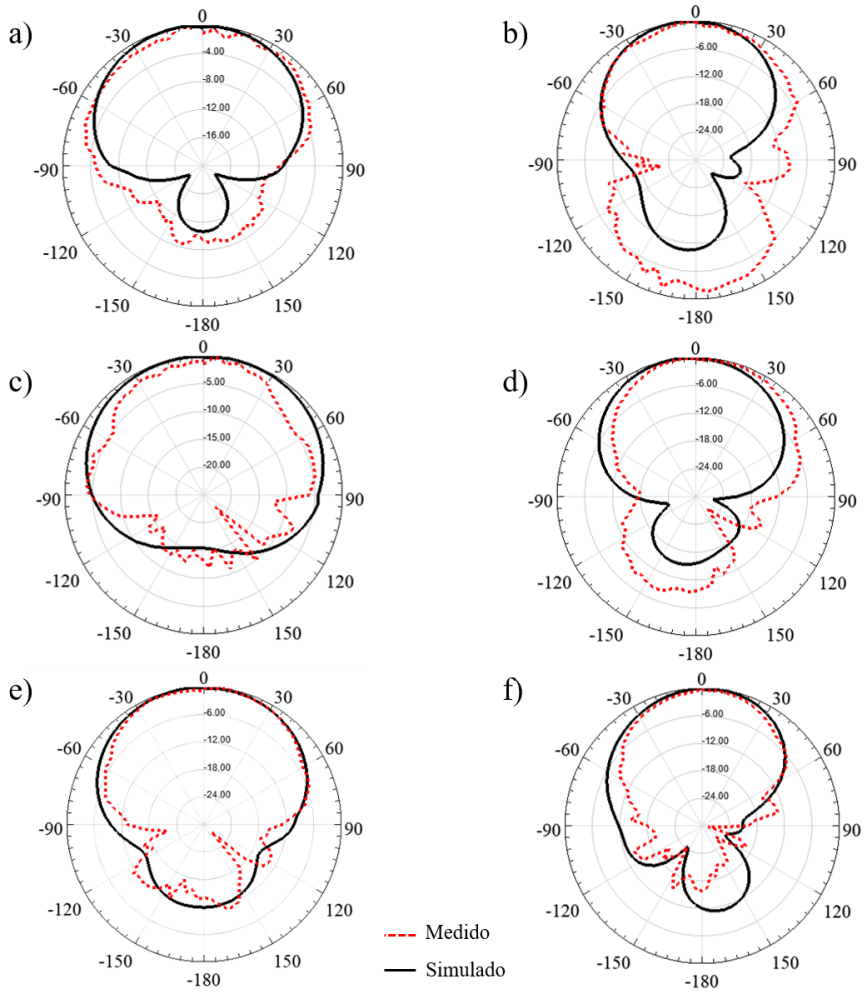


Figura 4.18: *Diagrama de radiação simulado e medido da antena QY em elevação em (a) 1,9 GHz, (c) 2,6 GHz, (e) 3,5 GHz e em azimute em em elevação (a) e azimute (b) 1,9 GHz, (d) 2,6 GHz, (f) 3,5 GHz.*

racterísticas construtivas. A Tabela 4.12 apresenta um resumo da comparação.Em [89] e [90], os autores propuseram uma antena quasi yagi, miniaturizada e com banda larga centrada em 0,9 e 7 GHz, respectivamente. Em [91] e [90], foi proposta uma antena Quasi-Yagi com dupla banda centradas em 1,6, 2,6 GHz e 0,43 e 0,88 GHz, respectivamente. Por fim, uma antena Quasi-Yagi com tripla banda, proposta em [92], centrada em 4,0, 5,6 e 8,2 GHz. A última linha da Tabela 4.12 apresenta os resultados medidos da antena Quasi-Yagi desenvolvida nesse trabalho. A antena opera nas frequências centrais de 1,9, 2,6 e 3,5 GHz, com ganho medido de 3,5, 4,4, 6,3 dBi e RFC de 3, 12 e 13 dB, respectivamente. Considerando os resultados obtidos, nota-se o comportamento similar da antena desenvolvida e otimizada com o uso de técnicas de Inteligência Computacional com as antenas Quasi-Yagi que vem sendo exploradas na literatura, comprovando a efetividade da metodologia proposta.

Tabela 4.12: *Comparação entre as antenas do tipo Quasi-Yagi.*

Ref.	***Frequência de operação (GHz)***	***Banda (GHz)***	***Ganho (dBi)***	***RFC (dB)***	***Eficiência de radiação***	***Dimensão física (mm)***	***Dimensão elétrica (λ)***
[89]	0,9	0,43	5,5	10,0	-	93,0 x 76.6	0,28 x 0,23
[91]	1,6 2,6	0,064 0,17	6,0 3,7	13,1 10,3	0,91 0,80	68,0 x 54,0	0,37 x 0,29
[90]	7,0	8,0	4,0	10,0	-	34,0 x 30,0	0,79 x 0,73
[92]	4,0 5,62 8,27	0,6 0,25 0,65	3,6 2,3 3,84	18,0 15,0 9,0	-	29,1 x 18,1	0,35 x 0,34
[93]	2,45 5,0	0,3 0,5	6,8 5,3	-	-	55,0 x 48,0	0,45 x 0,39
[94]	0,43 0,88	0,45	5,5 5,0	12,0 14,0	-	304,8 x 228,6	0,44 x 0,33
QY proposta	1,9 2,6 3,5	0,15 0,65 0,6	3,5 4,4 6,5	3,0 12,0 13,0	0,86 0,91 0,84	50,0 x 60,0	0,32 x 0,38

4.4 Tabela Comparativa dos Estudos de Caso

Esta Seção tem por objetivo facilitar a análise comparativa dos resultados obtidos para as antenas MPPD, MSPD e QY. A Tabela 4.13 apresenta um resumo dos estudos de caso utilizados nesse trabalho em termos de quantidade de amostras, configuração dos hiperparâmetros da rede neural, métricas de qualidade e tempo de desenvolvimento. É importante relembrar que para todos os estudos de casos as bases de dados foram divididas em 75% para treinamento e 25% para teste, uma vez que se trata de um aprendizado supervisionado em que essa divisão busca garantir a capacidade de generalização da rede. Essa quantidade de amostras foi obtida de maneira empírica, em que o número utilizado foi considerado adequado para representar cada estudo. Com relação aos hiperparâmetros, o algoritmo *GridSearch* com validação cruzada foi aplicado para auxiliar no processo de definição da melhor configuração da rede neural. Desta forma, a base de dados para as antenas MPPD, MSPD e QY foi composta por 171, 243 e 1364 amostras, respectivamente. A rede neural da antena MPPD foi formada por uma camada escondida com dois neurônios, sete camadas escondidas com vinte neurônios, uma camada de saída com dois neurônios, função de ativação *tanh* e apresentou um MSE de 0,0518. Para a antena MSPD, a rede neural foi composta por uma camada de entrada com três neurônios, oito camadas escondidas com cem neurônios, uma camada de saída com dois neurônios, função de ativação *tanh* e teve um MSE de 0,0087, enquanto que a rede neural da antena QY foi formada por uma camada de saída com dois neurônios, duas camadas escondidas com 32 e 256 neurônios, respectivamente, uma camada de saída com três neurônios, função de ativação *relu* e obteve-se um MSE de 1,44. Para o processo de otimização, algoritmo NSGA-II apresentou superioridade ao MOEA/D considerando os resultados dos indicadores de qualidade, de tal forma que os pares entrada-saída para o processo de otimização levaram em conta os valores estimados pelo NSGA-II. Por fim, a validação para os três estudos de caso houve uma boa concordância entre os valores estimados pelo algoritmo de otimização e simulados. Além disso, em termos de tempo de desenvolvimento, os resultado obtidos de 32 minutos e 53 segundos para a MPPD, 55 minutos e 5 segundos para a MSPD e 7 horas e 2 minutos para a QY, comprovaram a eficácia da metodologia desenvolvida e a contribuição ao Engenheiro de RF, como ferramenta auxiliar no processo de desenvolvimento e otimização de antenas.

Tabela 4.13: *Resumo dos resultados obtidos para os estudos de caso MPPD, MSPD e QY.*

Antenas	***Base de Dados***	***Hiperparâmetros da Rede Neural***				***Indicadores de Qualidade***	***Tempo de Desenvolvimento***
		Camada de Entrada	***Camadas Escondidas***	***Camadas de Saída***	***Função de Ativação***	***Hypervolume / Espaçamento/Espalhamento***	***Processamento e Validação***
MPPD	171	1 camada 2 neurônios	7 camadas 20 neurônios	1 camada 2 neurônios	tanh	0,095/ 0,017/ 0,553	32 minutos 53 segundos
MSPD	243	1 camada 3 neurônios	8 camadas 20 neurônios	1 camada 2 neurônios	tanh	0,717/ 0,020/ 0,436	55 minutos 5 segundos
QY	1364	1 camada 2 neurônios	1 canada 32 neurônios 1 camada 256 neurônios	1 camada 2 neurônios	relu	0,591/ 0,055/ 0,456	7 horas 2 minutos

Capítulo 5

5. Conclusões e Trabalhos Futuros

A dissertação teve como principal objetivo a apresentação de uma metodologia de desenvolvimento e otimização de antenas fundamentado em aplicações de técnicas de inteligência computacional, a fim de comprovar a eficácia dessa ferramenta na área do eletromagnetismo. O trabalho baseou-se na criação da metodologia e na validação do método, a partir de três estudos de caso, nos quais simulações numéricas e fabricação foram explorados.

No Capítulo 1, foi apresentada a contextualização sobre como o avanço tecnológico vem revolucionando os processos industriais. Partindo desde a máquina a vapor, na 1ª Revolução Industrial, até chegar a implementações de robôs autônomos, conceito de *big data* e *analytics*, máquinas inteligentes e o uso de simulações nos processos de desenvolvimento. Tais informações foram essenciais para justificar a importância do trabalho proposto. Adicionalmente foi apresentada uma revisão bibliográfica sobre o desenvolvimento de antenas com o uso de técnicas de Inteligência Artificial, bem como os algoritmos de otimização computacional mais aplicados dentro desse contexto. Tal revisão foi importante para a definição de como seria explorada a metodologia deste trabalho.

No Capítulo 2, foram apresentados os fundamentos teóricos necessários para o bom entendimento do trabalho, incluindo as explicações com relação à distinção entre inteligência artificial e computacional; formas de implementação de técnicas de inteligência artificial em telecomunicações; conceitos gerais sobre redes neurais artificiais e aprendizagem máquina, além de exposição e formas de identificação dos processos de aprendizagem de máquina; por fim, os tipos de otimização computacional, algoritmos que estão sendo aplicados no trabalho, e as métricas de desempenho.

O Capítulo 3 apresentou e detalhou a metodologia elaborada para o desenvolvimento de antenas, a partir de técnicas de Inteligência Computacional. Foi realizada

uma explicação detalhada da metodologia, além de um passo-a-passo para a construção do conjuto de dados; criação e treinamento do modelo substituto; análise e definição dos algoritmos evolucionários para a otimização dos resultados provindos do modelo substituto; e a validação, contrução e análise dos resultados obtidos. Ainda nesse Capítulo, foram apresentadas as antenas usadas nos estudos de caso para a validação da metodologia proposta. A primeira foi uma antena do tipo dipolo de meia onda, alimentada por uma porta casada. Depois, uma antena dipolo de onda completa, com estrutura de casamento de impedância adicionada. Por fim, um modelo mais complexo, uma antena do tipo Quasi-Yagi que opera em três frequências diferentes em torno de 1,9, 2,6 e 3,5 GHz.

Os resultados numéricos e experimentais para a validação da metodologia proposta foram descritos no Capítulo 4. A estrutura da base de dados da antena MPPD foi composta por 171 amostras, conseguindo atingir um erro quadrático médio de 0,0518, o que garante a capacidade de generalização do modelo. O algoritmo NSGA-II foi implementado para o processo de otimização, uma vez que apresentou os melhores indicadores de qualidade. As soluções propostas pelo algoritmo foram formadas por pontos minimamente espaçados, porém com regiões de aglutinação que levaram o modelo a ter poucas soluções que recobrissem a fronteira de Pareto em sua extensão. A solução do par entrada-saída ótima encontrada pelo algoritmo levou cerca de 53 segundos, enquanto que o processo de simulação com as mesmas dimensões consumiu cerca de 2 minutos. A resposta estimada pelo NSGA-II foi uma banda de operação de 18% centrada em 3,53 GHz. Na etapa validação, foram utilizadas três estratégias para a verificação da precisão e do consumo de tempo que a metodologia proposta pode garantir.

No segundo estudo de caso, a MSPD, a rede neural foi formada por 243 amostras, atingindo um erro de 0,0087. O algoritmo NSGA-II também foi utilizado para o processo de otimização, com o intuito de aumentar a largura de faixa centrada em 3,5 GHz. As soluções propostas pelo algoritmo apresentaram um conjunto de pontos discretos, minimamente espaçados e localizados em diferentes regiões da fronteira de Pareto, o que garante diversificadas soluções de pares entrada-saída. A solução ótima encontrada pelo MOEA gastou 5 segundos, com banda de operação de 21,4% centrada em 3,5 GHz. Na etapa de validação, o modelo fabricado obteve uma banda de operação idêntica à estimada, porém com um pequeno deslocamento no ponto de mínimo, passando a ser centrado em 3,54 GHz.

Finalmente, a antena QY foi proposta como terceiro e mais complexo estudo de caso. Para este estudo, o objetivo foi minimizar o coeficiente de reflexão centrado em torno de 1,9, 2,6 e 3,5 GHz. A rede neural foi composta por 1364 amostras e

atingiu um MSE = 1,44. O algoritmo NSGA-II foi usado no processo de otimização, apresentou amostras discretas, espaçadas e que recobriram a fronteira de Pareto em sua extensão, o que fornece um grande conjunto de pares entrada-saída. A solução ótima encontrada consumiu 2 minutos, aproximadamente, com valores de coeficientes de reflexão de f_1 = -15,78 dB, f_2 = -22,66 dB e f_3 = -15,62 dB, centradas em 1,9, 2,6 e 3,5 GHz, respectivamente. Na etapa de validação, existiram algumas discrepâncias entre as amostras estimadas e simuladas devido ao fato de o regressor ainda não ser o ideal, porém o algoritmo de otimização foi capaz de estimar as amostras de saída, respeitando o limiar do coeficiente de reflexão estar abaixo de -10 dB, o que ainda é adequado para a validação da metodologia proposta.

Os protótipos dos casos de estudo MSPD e QY apresentaram diferenças em suas propriedades eletromagnéticas, quando comparado às simulações. Tais diferenças foram provocadas devido às imprecisões do processo de fabricação da antena. Além disso, as diferenças na largura de faixa e valores do coeficiente de reflexão foram por conta das variações do MSE do processo de regressão, além da adequação da rede neural artificial.

Com base nos resultados apresentados, é importante destacar que a metodologia proposta pode ser uma ferramenta em potencial tanto no desenvolvimento quanto na otimização de projetos generalistas de antenas. Esse método pode ser útil como um modelo preliminar e complementar para *softwares* comerciais de simulações eletromagnética, com o objetivo de economia de tempo e redução de custo computacional. Pode-se levar em conta essa proposta para soluções que demandem diferentes faixas de frequências. Para isso, é necessário que sejam inseridos pares de entrada-saída no banco de dados, ajustar o número de camadas e neurônios, bem como seus respectivos pesos sinápticos na rede neural. Em curto prazo, a solução pode se apresentar como uma tarefa demorada, na qual o engenheiro de RF precisará adicionar os pares de entrada-saída à uma plataforma de hospedagem de arquivos, como o GitHub, e construir um amplo conjunto de dados. Sendo assim, cada modificação na estrutura da antena precisará ser carregada para alimentar o conjunto. No entanto, a longo prazo, a plataforma será composta por uma grande quantidade de conteúdo na base de dados, em uma estrutura *open source* e, então, será necessário o recurso de um único engenheiro de RF para a definição dos objetivos de projeto e um posterior ajuste fino no *software* de simulação eletromagnética, caso necessário, uma vez que a metodologia já está definida.

Como futuros trabalhos sugere-se investigar e analisar o regressor mais adequado para o desenvolvimento e a otimização de antenas, dentro dos variados tipos, como SVR, *CatBoost*, Processo Gaussiano, *Random Forest*. Um segundo trabalho poderia

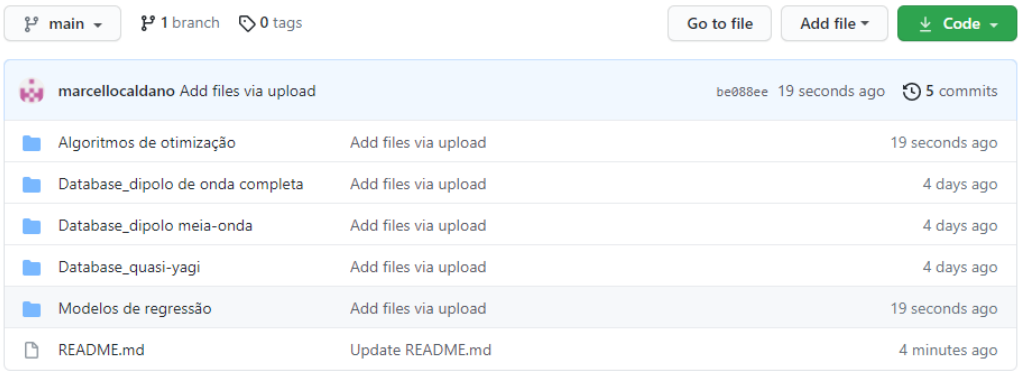
ser o uso desta metodologia para o desenvolvimento e a otimização de outros modelos de antenas, bem como outras variáveis de saída relativas aos parâmetros das antenas, como o nível dos lobos lateriais (SLL), a abertura de feixe e o ganho, para aplicações em redes de antenas e radares. Outra sugestão é o incremento da base de dados para outros tipos de antenas que operem em outras faixas de frequências para o aumento das opções de protótipos de antenas desenvolvidas por meio de técnicas de inteligência computacional. Para facilitar o processo de construção da base de dados e aumentar o número de amostras, sugere-se o uso da rede adversarial generativa (*Generative Adversarial Networks*, GAN). Essa técnica aprende a gerar novos dados com as mesmas estatísticas do conjunto usado no treinamento. Por exemplo, para o desenvolvimento de antenas. uma GAN pode gerar novos pares entrada-saída para a composição da base de dados, fundamentada na base inicialmente construída, reduzindo a carga de trabalho e o tempo de contrução da base de dados. Por fim, uma última sugestão, é aproveitar a metodologia proposta para realização de estimação de canais de propagação para as redes 5G e futuras tecnologias 6G.

Apêndice I

Códigos Completos e Base de Dados Disponíveis para Acesso

Endereço para acesso:

https://github.com/marcellocaldano/Modelo-de-aprendizagem-ANN-.git



Apêndice II

Códigos Parciais para o Processo de Aprendizagem

Código de configuração para as antenas dipolo:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.multioutput import MultiOutputRegressor
from sklearn.preprocessing import StandardScaler

df=pd.read_excel('BW_SMA_database1.xlsx')
data=df.copy()
x=data.drop(labels=['BW','f'],axis=1)
y=data[['BW','f']]

X_train,X_test,y_train,
y_test=train_test_split(x,y,test_size=0.30,
random_state=200)

std = StandardScaler()
X_train_std = std.fit_transform(X_train)
X_test_std = std.transform(X_test)
Y_train_std = std.fit_transform(y_train)
Y_test_std = std.transform(y_test)
```

```
mlp=MLPRegressor()
param_grid={'hidden_layer_sizes':
[(20,20,20,20,20,20)],
'max_iter':[5000],'activation':
['tanh','relu','logistic']
,'solver':['adam','sgd'],
'learning_rate_init': [0.001,0.01,0.1]}

grid = GridSearchCV(estimator=mlp,
param_grid=param_grid,
scoring='neg_mean_squared_error',
cv=7, verbose=0)

grid.fit(X_train_std, Y_train_std)
grid.best_estimator_ #BW

Y_pred = grid.best_estimator_
.predict(X_test_std)
Y_pred=std.inverse_transform(Y_pred)
msq_err = mean_squared_error
(y_test, Y_pred)
print("Mean Squared Error:
{}".format(msq_err))
```

Código de configuração para a antena quasi-yagi:

```
import numpy as np
from numpy import sqrt
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.model_selection import
train_test_split
from tensorflow.keras import Sequential
from tensorflow.keras.layers
import Dense
from sklearn.model_selection
import KFold
from sklearn.preprocessing
import StandardScaler
from sklearn.preprocessing
import MinMaxScaler
from numpy import asarray
from pandas import read_csv
from sklearn.model_selection import train_test_split
```

```
#from autokeras import StructuredDataRegressor
from numpy import asarray
from timeit import default_timer as timer
new_model = tf.keras.models.

load_model('Modelo_Quasi_Yagi_Antenna.h5')
df1=pd.read_excel('Quasi-yagi L5 e Lp dataset.xlsx')
df2=pd.read_excel('Quasi-yagi L5 e Lp dataset (1).xlsx')
df=pd.concat([df1,df2])
data=df.copy()
data = df.drop_duplicates()
x=data.drop(labels=['f1', 'f2','f3'],axis=1)
y=data[['f1', 'f2','f3']]

from sklearn.ensemble import IsolationForest
iso_forest = IsolationForest(n_estimators=400,
contamination=0.5,random_state=1000)
iso_forest = iso_forest.fit(data)
isof_outliers = iso_forest.predict(data)
isoF_outliers_values = data[iso_forest.
predict(data) == -1]
data = data.drop(isoF_outliers_values.
index.values.tolist())

x = data.drop(labels=['f1', 'f2', 'f3'], axis=1)
y = data[[ 'f1', 'f2', 'f3']]
X_train,X_test,y_train,
y_test=train_test_split(x,y,
test_size=0.30,random_state=7000)
print(X_train.shape, X_test.shape,
y_train.shape, y_test.shape)
n_features = X_train.shape[1]
std = StandardScaler()
#std = MinMaxScaler()
#x_std=std.fit_transform(x)

X_train_std = std.fit_transform(X_train)
X_test_std = std.transform(X_test)
Y_train_std = std.fit_transform(y_train)
Y_test_std = std.transform(y_test)
es = EarlyStopping(monitor='val_loss',
mode='min',
verbose=0, patience=400)
neuron=1024
history=model.fit(X_train_std, Y_train_std,
```

```
epochs=6000,
validation_data=(X_test_std,Y_test_std),
batch_size=256,callbacks=[es], verbose=0)
end = timer()
print(end - start)
model.summary( )
from matplotlib import pyplot
#pyplot.subplot(211)
plt.figure(figsize=(20,10))
pyplot.title('Cross-Entropy Loss',
pad=-40)
pyplot.plot(history.history['loss'],
label='train')
pyplot.plot(history.history['val_loss'],
label='test')
pyplot.legend()
yhat =model.predict(X_test_std)
yhat=std.inverse_transform(yhat)
mse = (mean_squared_error(y_test, yhat))
rmse = (mean_squared_error(y_test, yhat,
squared=False))
mae = (mean_absolute_error(y_test, yhat))
print('MSE = ',mse,'RMSE = ',rmse, 'MAE = ', mae)
preds= model.predict(X_test_std)
preds=std.inverse_transform(preds)
rmse = (mean_squared_error(y_test.values[:,0]
, preds[:,0]))
print(rmse)
preds= model.predict(X_test_std)
preds=std.inverse_transform(preds)
rmse = (mean_squared_error(y_test.values[:,1]
, preds[:,1]))
print(rmse)
preds= new_model.predict(X_test_std)
preds=std.inverse_transform(preds)
rmse = (mean_squared_error(y_test.values[:,2]
, preds[:,2]))
print(rmse)
def plot_target_var(var_name, var_indx, y_true, y_pred):
    if var_indx is not None:
      target_true = y_true[:, var_indx]
      target_pred = y_pred[:, var_indx]
    else:
      target_true = y_true
      target_pred = y_pred
```

```
    fig, ax = plt.subplots(figsize=(20,10))
    ax.scatter(target_true, target_pred)
    ax.plot([target_true.min(),target_true.max()],
    [target_true.min(), target_true.max()],
    'k--', lw=4)
    ax.set_xlabel('Measured')
    ax.set_ylabel('Predicted')
    plt.title(var_name)
    for item in ([ax.title, ax.xaxis.label,
                  ax.yaxis.label] +
                  ax.get_xticklabels() +
                  ax.get_yticklabels()):
        item.set_fontsize(20)
    plt.show()
plot_target_var(var_name='f1', var_indx=0,
y_true=y_test.values,y_pred=preds)
plot_target_var(var_name='f2', var_indx=1,
y_true=y_test.values,y_pred=preds)
plot_target_var(var_name='f3', var_indx=2,
y_true=y_test.values,y_pred=preds)
model.save('Modelo_Quasi_Yagi_Antenna.h5')
```

Apêndice III

Códigos Parciais para Otimização Multiobjetivo

Código parcial para implementação dos algoritmos multiobjetivo para as antenas dipolo:

```
%%Multiobjective definition problem%%
 def __init__(self, number_of_variables: int = 3,
                number_of_objectives: int = 2,
                model_dict = None, lb = None, up = None):
     super(DipoloSimplesMO2, self).__init__()
     self.number_of_variables = number_of_variables
     self.number_of_objectives = number_of_objectives
     self.number_of_constraints = 0
     self.obj_directions = [self.MINIMIZE, self.MINIMIZE]
     self.obj_labels = ['BW', 'f']
     self.lower_bound = lb
     self.upper_bound = up
     self.reg_model = model_dict['regmodel']
     self.std_in = model_dict['std_in']
     self.name = 'DipoloSimplesMO2'
 def evaluate(self, solution: FloatSolution)->
     FloatSolution:
     input_vars = np.array(solution.variables)
     .reshape(1, -1)
     input_vars = self.std_in.transform(input_vars)
     objct_vals = self.reg_model.predict(input_vars)[0]
     objct_vals[0] = -1 * objct_vals[0]
     objct_vals[1] = np.abs(3.5 - objct_vals[1])
     solution.objectives = list(objct_vals)
     return solution
 def get_name(self):
     return self.name
```

```
%%Algorithm definition and execution%%
class NSGAII(nsgaii.NSGAII):
    def __init__(self, **kwargs):
        super(NSGAII, self).__init__(**kwargs)
        self.front_history = dict()
    def get_current_front(self):
        return copy(get_non_dominated_solutions
        (self.get_result()))
    def front_progress(self):
        sorted_keys = sorted(self.front_history.keys())
        for key in sorted_keys:
            front = self.front_history[key]
            yield key, front
    def run(self):
        """ Execute the algorithm. """
        self.start_computing_time = time.time()
        self.solutions = self.create_initial_solutions()
        self.solutions = self.evaluate(self.solutions)
        LOGGER.debug('Initializing progress')
        self.init_progress()
        LOGGER.debug('Running main loop until
        termination criteria is met')
        iterations_index = 0
        while not self.stopping_condition_is_met():
            self.step()
            self.update_progress()
            self.front_history[iterations_index]
            = self.get_current_front()
            iterations_index += 1
        self.total_computing_time = time.time()
        - self.start_computing_time

class MOEAD(moead.MOEAD):
    def __init__(self, **kwargs):
        super(MOEAD, self).__init__(**kwargs)
        self.front_history = dict()
    def get_current_front(self):
        return copy(get_non_dominated_solutions
        (self.get_result()))
    def front_progress(self):
        sorted_keys = sorted(self.front_history.keys())
        for key in sorted_keys:
            front = self.front_history[key]
            yield key, front
```

```
    def run(self):
        """ Execute the algorithm. """
        self.start_computing_time = time.time()
        self.solutions = self.create_initial_solutions()
        self.solutions = self.evaluate(self.solutions)
        LOGGER.debug('Initializing progress')
        self.init_progress()
        LOGGER.debug('Running main loop until
        termination criteria is met')
        iterations_index = 0
        while not self.stopping_condition_is_met():
            self.step()
            self.update_progress()
            self.front_history[iterations_index] =
            self.get_current_front()
            iterations_index += 1
        self.total_computing_time = time.time()
        -self.start_computing_time
```

Código parcial para implementação dos algoritmos multiobjetivo para a antena quasi-yagi:

```
%%Multiobjective definition problem%%
def __init__(self, number_of_variables: int = 6,
             number_of_objectives: int = 3,
             keras_model = None,
             model_dict = None, lb = None, up = None):
    super(QuasiYagiMO3OBJ, self).__init__()
    self.number_of_variables = number_of_variables
    self.number_of_objectives = number_of_objectives
    self.number_of_constraints = 0
    self.obj_directions = [self.MINIMIZE, self.MINIMIZE,
    self.MINIMIZE]
    self.obj_labels = ['RF1', 'RF2', 'RF3']
    self.lower_bound = lb
    self.upper_bound = up
    self.reg_model = keras_model
    self.std_in = model_dict['std_in']
    self.std_out = model_dict['std_out']

    self.name = 'QuasiYagiMO3OBJ'
def evaluate(self, solution: FloatSolution)
    -> FloatSolution:
   input_vars = np.array(solution.variables).
   reshape(1, -1)
   input_vars = self.std_in.transform(input_vars)
   objct_vals = self.reg_model.predict(input_vars)[0]
```

```
    solution.objectives = list(objct_vals)
    return solution
def get_name(self):
    return self.name

%%Algorithm definition and execution%%
class NSGAII(nsgaii.NSGAII):
    def __init__(self, **kwargs):
        super(NSGAII, self).__init__(**kwargs)
        self.front_history = dict()
    def get_current_front(self):
        return copy(get_non_dominated_solutions
        (self.get_result()))
    def front_progress(self):
        sorted_keys = sorted(self.front_history.keys())
        for key in sorted_keys:
            front = self.front_history[key]
            yield key, front
    def run(self):
        """ Execute the algorithm. """
        self.start_computing_time = time.time()
        self.solutions = self.create_initial_solutions()
        self.solutions = self.evaluate(self.solutions)
        LOGGER.debug('Initializing progress')
        self.init_progress()
        LOGGER.debug('Running main loop
        until termination criteria is met')
        iterations_index = 0
        while not self.stopping_condition_is_met():
            self.step()
            self.update_progress()
            self.front_history[iterations_index]
            = self.get_current_front()
            iterations_index += 1
        self.total_computing_time = time.time()
        - self.start_computing_time
class MOEAD(moead.MOEAD):

    def __init__(self, **kwargs):
        super(MOEAD, self).__init__(**kwargs)
        self.front_history = dict()
    def get_current_front(self):
        return copy(get_non_dominated_solutions
        (self.get_result()))
    def front_progress(self):
```

```
        sorted_keys = sorted(self.front_history.keys())
        for key in sorted_keys:
            front = self.front_history[key]
            yield key, front
    def run(self):
        """ Execute the algorithm. """
        self.start_computing_time = time.time()
        self.solutions = self.create_initial_solutions()
        self.solutions = self.evaluate(self.solutions)
        LOGGER.debug('Initializing progress')
        self.init_progress()
        LOGGER.debug('Running main loop until
        termination criteria is met')
        iterations_index = 0
        while not self.stopping_condition_is_met():
            self.step()
            self.update_progress()
            self.front_history[iterations_index]
            = self.get_current_front()
            iterations_index += 1
        self.total_computing_time = time.time()
        - self.start_computing_time
```

Referências Bibliográficas

[1] K. Henning, "Recommendations for implementing the strategic initiative industrie 4.0," *National Academy of Science and Engineering*, 2013.

[2] S. S. Haykin, *Neural networks and learning machines*, 3rd ed. New York: Pearson Prentice Hall, 2009.

[3] D. Graupe, *Principles of artificial neural networks: basic designs to deep learning*. World Scientific, 2019, vol. 8.

[4] M. Mahrach, G. Miranda, C. León, and E. Segredo, "Comparison between single and multi-objective evolutionary algorithms to solve the knapsack problem and the travelling salesman problem," *Mathematics*, vol. 8, no. 11, 2020.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[6] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[7] D. Klitou, J. Conrads, and M. Rasmussen, "Digital transformation monitor germany: Industrie 4.0," *White paper*, pp. 1–8, Jan. 2017.

[8] I. Cisco, "Cisco 5g vision series: Laying the foundation for new technologies, use cases, and business models," *White Paper*, pp. 1–19, 2016.

[9] T. Bueno, E. Morais, L. Almeida, R. Righi, and A. Alberti, *"Blockchain and Industry 4.0: Overview, Convergence, and Analysis"*. Singapure: Springer, 2020.

[10] S. Gallego and M. García, "Industry 4.0 implications in production and maintenance management: An overview," *Procedia Manufacturing*, vol. 41, pp. 415 – 422, 2019.

[11] M. E. Haque and U. Baroudi, "Ambient self-powered cluster-based wireless sensor networks for industry 4.0 applications," *Soft Computing*, pp. 1–26, 2020.

[12] D. Goldsman, R. Nance, and J. Wilson, "A brief history of simulation." 2010 Winter Simulation Conference, 2010, pp. 310 – 313.

[13] I. Karlsson, J. Bernedixen, A. H. C. Ng, and L. Pehrsson, "Combining augmented reality and simulation-based optimization for decision support in manufacturing," in *2017 Winter Simulation Conference (WSC)*, 2017, pp. 3988–3999.

[14] I. GSMA, "Road to 5g: Introduction and migration," *White paper*, pp. 1–54, 2018.

[15] K. L. M. Latva-aho, "6g research visions 1 - key drivers and research challenges for 6g ubiquitous wireless intelligence," *White Paper*, pp. 1–33, Sept. 2019.

[16] A. Klautau, N. González-Prelcic, and R. W. Heath, "Lidar data for deep learning-based mmwave beam-selection," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 909–912, 2019.

[17] Y. Wang, A. Klautau, M. Ribero, M. Narasimha, and R. W. Heath, "Mmwave vehicular beam training with situational awareness by machine learning," in *2018 IEEE Globecom Workshops (GC Wkshps)*, 2018, pp. 1–6.

[18] C. A. Kyriakopoulos, G. I. Papadimitriou, P. Nicopolitidis, and E. Varvarigos, "Energy-efficient lightpath establishment in backbone optical networks based on ant colony optimization," *J. Lightwave Technol.*, vol. 34, no. 23, pp. 5534–5541, Dec 2016.

[19] B. Choudhury, S. Thomas, and R. M. Jha, "Implementation of soft computing optimization techniques in antenna engineering [antenna applications corner]," *IEEE Antennas and Propagation Magazine*, vol. 57, no. 6, pp. 122–131, 2015.

[20] P. Watson and K. C. Gupta, "Em-ann models for via interconnects in microstrip circuits," in *1996 IEEE MTT-S International Microwave Symposium Digest*, vol. 3, 1996, pp. 1819–1822 vol.3.

[21] S. Gangwar, R. Gangwar, B. Kanaujia, and P. , "Resonant frequency of circular microstrip antenna using artificial neural networks," vol. 37, pp. 204–208, 06 2008.

[22] P. Gupta, R. Gupta, R. Sharma, and B. yadav, "Calculation of resonating frequency of an equilateral triangular microstrip antenna using artificial neural network," *Advanced Computational Techniques in Electromagnetics*, vol. 2013, pp. 1–8, 05 2013.

[23] L. V. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. New York: Prentice Hall, 2009.

[24] S. Haykin, "Neural networks: a comprehensive foundation," vol. 13, no. 4, pp. 409–412, 1999.

[25] B. Banerjee, "A self-organizing auto-associative network for the generalized physical design of microstrip patches," *IEEE Transactions on Antennas and Propagation*, vol. 51, no. 6, pp. 1301–1306, 2003.

[26] Y. Kim, S. Keely, J. Ghosh, and H. Ling, "Application of artificial neural networks to broadband antenna design based on a parametric frequency model," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3, pp. 669–674, 2007.

[27] Z. Zheng, X. Chen, and K. Huang, "Application of support vector machines to the antenna design," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 21, pp. 85 – 90, 01 2011.

[28] B. Liu, H. Aliakbarian, Z. Ma, G. A. E. Vandenbosch, G. Gielen, and P. Excell, "An efficient method for antenna design optimization based on evolutionary computation and machine learning techniques," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 1, pp. 7–18, 2014.

[29] H. I. Volos and R. M. Buehrer, "Cognitive engine design for link adaptation: An application to multi-antenna systems," *IEEE Transactions on Wireless Communications*, vol. 9, no. 9, pp. 2902–2913, 2010.

[30] H. Aliakbari, A. Abdipour, A. Costanzo, D. Masotti, R. Mirzavand, and P. Mousavi, "Ann-based design of a versatile millimetre-wave slotted patch multi-antenna configuration for 5g scenarios," *IET Microwaves, Antennas Propagation*, vol. 11, no. 9, pp. 1288–1295, 2017.

[31] Anuradha, A. Patnaik, and S. N. Sinha, "Design of custom-made fractal multi-band antennas using ann-pso [antenna designer's notebook]," *IEEE Antennas and Propagation Magazine*, vol. 53, no. 4, pp. 94–101, 2011.

[32] J. P. Jacobs and S. Koziel, "Two-stage framework for efficient gaussian process modeling of antenna input characteristics," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 2, pp. 706–713, 2014.

[33] T. Khan, A. De, and M. Uddin, "Prediction of slot-size and inserted air-gap for improving the performance of rectangular microstrip antennas using artificial neural networks," *IEEE Antennas and Wireless Propagation Letters*, vol. 12, pp. 1367–1371, 2013.

[34] D. Ding and G. Wang, "Modified multiobjective evolutionary algorithm based on decomposition for antenna design," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 10, pp. 5301–5307, 2013.

[35] S. Goudos, C. Kalialakis, and R. Mittra, "Evolutionary algorithms applied to antennas and propagation: A review of state of the art," *International Journal of Antennas and Propagation*, vol. 2016, pp. 1–12, 01 2016.

[36] A. Pietrenko-Dabrowska and S. Koziel, "Computationally-efficient design optimisation of antennas by accelerated gradient search with sensitivity and design change monitoring," *IET Microwaves, Antennas Propagation*, vol. 14, no. 2, pp. 165–170, 2020.

[37] S. Koziel and S. Ogurtsov, "Rapid optimisation of omnidirectional antennas using adaptively adjusted design specifications and kriging surrogates," *IET Microwaves, Antennas Propagation*, vol. 7, no. 15, pp. 1194–1200, 2013.

[38] S. Koziel, S. Ogurtsov, I. Couckuyt, and T. Dhaene, "Cost-efficient electromagnetic-simulation-driven antenna design using co-kriging," *IET Microwaves, Antennas Propagation*, vol. 6, no. 14, pp. 1521–1528, 2012.

[39] S. Koziel and S. Ogurtsov, "Multi-objective design of antennas using variable-fidelity simulations and surrogate models," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 12, pp. 5931–5939, 2013.

[40] B. Liu, H. Aliakbarian, Z. Ma, G. A. E. Vandenbosch, G. Gielen, and P. Excell, "An efficient method for antenna design optimization based on evolutionary computation and machine learning techniques," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 1, pp. 7–18, 2014.

[41] J. P. Jacobs and S. Koziel, "Two-stage framework for efficient gaussian process modeling of antenna input characteristics," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 2, pp. 706–713, 2014.

[42] L. Tenuti, G. Oliveri, D. Bresciani, and A. Massa, "Advanced learning-based approaches for reflectarrays design," in *2017 11th European Conference on Antennas and Propagation (EUCAP)*, 2017, pp. 84–87.

[43] D. Ding and G. Wang, "Modified multiobjective evolutionary algorithm based on decomposition for antenna design," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 10, pp. 5301–5307, 2013.

[44] E. Deniz Ülker and S. Ulker, "Antenna design using animal migration optimization algorithm," *The Journal of Engineering*, vol. 1, 05 2016.

[45] R. Diaz de Leon, G. Gonzalez, E. Flores-Garcia, A. Rodriguez, and F. Gonzalez, "Evolutionary algorithm geometry optimization of optical antennas," *International Journal of Antennas and Propagation*, vol. 2016, pp. 1–7, 06 2016.

[46] J. Dong, Q. Li, and L. Deng, "Fast multi-objective optimization of multi-parameter antenna structures based on improved moea/d with surrogate-assisted model," *AEU - International Journal of Electronics and Communications*, vol. 72, pp. 192–199, 02 2017.

[47] H. Aliakbari, A. Abdipour, A. Costanzo, D. Masotti, R. Mirzavand, and P. Mousavi, "Ann-based design of a versatile millimetre-wave slotted patch multi-antenna configuration for 5g scenarios," *IET Microwaves, Antennas Propagation*, vol. 11, no. 9, pp. 1288–1295, 2017.

[48] Z. Zheng, X. Chen, and K. Huang, "Application of support vector machines to the antenna design," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 21, no. 1, pp. 85–90, 2016.

[49] B. Liu, M. O. Akinsolu, N. Ali, and R. Abd-Alhameed, "Efficient global optimisation of microwave antennas based on a parallel surrogate model-assisted evolutionary algorithm," *IET Microwaves, Antennas Propagation*, vol. 13, no. 2, pp. 149–155, 2017.

[50] A. Pietrenko-Dabrowska and S. Koziel, "Computationally-efficient design optimization of antennas by accelerated gradient search with sensitivity and design change monitoring," *IET Microwaves, Antennas Propagation*, vol. 14, 10 2019.

[51] R. Balmer, S. Levin, and S. Schmidt, "Artificial intelligence applications in telecommunications and other network industries," *Telecommunications Policy*, vol. 44, p. 101977, 05 2020.

[52] F. Morales, M. Ruiz, L. Gifre, L. M. Contreras, V. López, and L. Velasco, "Virtual network topology adaptability based on data analytics for traffic prediction," *J. Opt. Commun. Netw.*, vol. 9, no. 1, pp. A35–A45, Jan 2017.

[53] N. Koji, I. Reljin, and B. Reljin, "Neural network for optimization of routing in communication networks," *Facta universitatis - series: Electronics and Energetics*, vol. 19, 01 2006.

[54] J. Mata, I. de Miguel, R. J. Durán, J. C. Aguado, N. Merayo, L. Ruiz, P. Fernández, R. M. Lorenzo, and E. J. Abril, "A svm approach for lightpath qot estimation in optical transport networks," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 4795–4797.

[55] P. B. Santos, M. C. Melo, E. Faustino Jr., A. Cerqueira S. Jr., and C. J. A. Bastos-Filho, "A comparison of evolutionary multi-objective optimization algorithms applied to antenna design," in *Intelligent Data Engineering and Automated Learning – IDEAL 2020*, C. Analide, P. Novais, D. Camacho, and H. Yin, Eds. Cham: Springer International Publishing, 2020, pp. 123–134.

[56] C. A. Kyriakopoulos, G. I. Papadimitriou, P. Nicopolitidis, and E. Varvarigos, "Energy-efficient lightpath establishment in backbone optical networks based on ant colony optimization," *J. Lightwave Technol.*, vol. 34, no. 23, pp. 5534–5541, Dec 2016.

[57] A. P. Engelbrecht, *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[58] J. C. Bezdek, "(computational) intelligence: What's in a name?" *IEEE Systems, Man, and Cybernetics Magazine*, vol. 2, no. 2, pp. 4–14, 2016.

[59] J. C. Bezdek., "Computational intelligence defined - by everyone !" in *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 10–37.

[60] W. Pedrycz, A. Sillitti, and G. Succi, *Computational Intelligence: An Introduction*, 01 2016, vol. 617, pp. 13–31.

[61] J. Rutherford, A. Gillespie, and R. Richardson, "The territoriality of pan-european telecommunications backbone networks," *Journal of Urban Technology*, vol. 11, no. 3, pp. 1–34, 2004.

[62] D. Zibar, L. H. H. de Carvalho, M. Piels, A. Doberstein, J. Diniz, B. Nebendahl, C. Franciscangelis, J. Estaran, H. Haisch, N. G. Gonzalez, J. C. R. F. de Oliveira, and I. T. Monroy, "Application of machine learning techniques for amplitude and phase noise characterization," *Journal of Lightwave Technology*, vol. 33, no. 7, pp. 1333–1343, 2015.

[63] C. Elliott and B. Heile, "Self-organizing, self-healing wireless networks," in *2000 IEEE Aerospace Conference. Proceedings (Cat. No.00TH8484)*, vol. 1, 2000, pp. 149–156 vol.1.

[64] G. Xu, Y. Mu, and J. Liu, "Inclusion of artificial intelligence in communication networks and services," 2017.

[65] S. Sendra, A. Rego, J. Lloret, J. M. Jimenez, and O. Romero, "Including artificial intelligence in a routing protocol using software defined networks," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2017, pp. 670–674.

[66] T. Nguyen, C. P. Lim, N. D. Nguyen, L. Gordon-Brown, and S. Nahavandi, "A review of situation awareness assessment approaches in aviation environments," *IEEE Systems Journal*, vol. 13, no. 3, pp. 3590–3603, 2019.

[67] T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention," in *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2016, pp. 223–228.

[68] B. M. Wilamowski, "Neural network architectures and learning algorithms," *IEEE Industrial Electronics Magazine*, vol. 3, no. 4, pp. 56–63, 2009.

[69] M. Amin and A. Ali, "Application of multilayer perceptron (mlp) for data mining in healthcare operations," 02 2017.

[70] J. Dong, W. Qin, and M. Wang, "Fast multi-objective optimization of multi-parameter antenna structures based on improved bpnn surrogate model," *IEEE Access*, vol. 7, pp. 77 692–77 701, 2019.

[71] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[72] I. Carlucho, M. De Paula, S. Wang, Y. Petillot, and G. G. Acosta, "Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning," *Robotics and Autonomous Systems*, vol. 107, pp. 71–86, 2018.

[73] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[74] R. Miikkulainen, *Topology of a Neural Network*. Boston, MA: Springer US, 2010, pp. 988–989.

[75] T. Poggio, Q. Liao, and A. Banburski, "Complexity control by gradient descent in deep networks," *Nature communications*, vol. 11, no. 1, pp. 1–5, 2020.

[76] K. Deb, *Multiobjective Optimization Using Evolutionary Algorithms. Wiley, New York*, 01 2001.

[77] S. Chand and M. Wagner, "Evolutionary many-objective optimization: A quick-start guide," *Surveys in Operations Research and Management Science*, vol. 20, no. 2, pp. 35–42, 2015.

[78] S. Mirjalili and A. Lewis, "Novel performance metrics for robust multi-objective optimization algorithms," *Swarm and Evolutionary Computation*, vol. 21, pp. 1–23, 2015.

[79] N. Riquelme, C. Von Lücken, and B. Baran, "Performance metrics in multi-objective optimization," in *2015 Latin American Computing Conference (CLEI)*. IEEE, 2015, pp. 1–11.

[80] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *International conference on parallel problem solving from nature*. Springer, 2004, pp. 832–842.

[81] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. C. Coello, F. Luna, and E. Alba, "Smpso: A new pso-based metaheuristic for multi-objective optimization," in *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*. IEEE, 2009, pp. 66–73.

[82] M. Reyes-Sierra, C. C. Coello *et al.*, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *International journal of computational intelligence research*, vol. 2, no. 3, pp. 287–308, 2006.

[83] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29–38, 2006.

[84] K. Bringmann and T. Friedrich, "An efficient algorithm for computing hypervolume contributions*," *Evolutionary Computation*, vol. 18, no. 3, pp. 383–402, 2010.

[85] J. Bader and E. Zitzler, "Hype: An algorithm for fast hypervolume-based many-objective optimization," *Evolutionary Computation*, vol. 19, no. 1, pp. 45–76, 2011.

[86] C. A. Balanis, *Antenna theory: analysis and design*. John wiley & sons, 2016.

[87] A. Inc., "High frequency simulator system (hfss)," vol. R3, 2019.

[88] Y. Ding, Y. C. Jiao, P. Fei, B. Li, and Q. T. Zhang, "Design of a multiband quasi-yagi-type antenna with cpw-to-cps transition," *IEEE Antennas and Wireless Propagation Letters*, vol. 10, pp. 1120–1123, 2011.

[89] S. A. Rezaeieh, M. A. Antoniades, and A. M. Abbosh, "Miniaturized planar yagi antenna utilizing capacitively coupled folded reflector," *IEEE Antennas and Wireless Propagation Letters*, vol. 16, pp. 1977–1980, 2017.

[90] J. Wu, Z. Zhao, Z. Nie, and Q.-H. Liu, "Bandwidth enhancement of a planar printed quasi-yagi antenna with size reduction," *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 1, pp. 463–467, 2014.

[91] H.-C. Huang, J.-C. Lu, and P. Hsu, "A compact dual-band printed yagi-uda antenna for gnss and cmmb applications," *IEEE Transactions on Antennas and Propagation*, vol. 63, no. 5, pp. 2342–2348, 2015.

[92] K. D. Xu, D. Li, Y. Liu, and Q. H. Liu, "Printed quasi-yagi antennas using double dipoles and stub-loaded technique for multi-band and broadband applications," *IEEE Access*, vol. 6, pp. 31 695–31 702, 2018.

[93] W. Zhang, Y. Zhuang, C. Song, Y. Huang, and J. Zhou, "A dual-band quasi-yagi wearable antenna with high directivity," in *2018 IEEE MTT-S International Wireless Symposium (IWS)*, 2018, pp. 1–3.

[94] Y. Bakirli, A. Selek, and M. Secmen, "Broadband compact quasi yagi antenna for uhf wireless communication systems with enhanced performance at uhf ism bands," *Radioengineering*, vol. 29, pp. 460–470, 09 2020.