Universidade Estadual de Campinas

LCSI Faculdade de Engenharia Elétrica e de Computação
Departamento de Máquinas, Componentes e Sistemas Inteligentes
Laboratório de Controle e Sistemas Inteligentes

Observadores Inteligentes de Estado: Propostas

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de

Mestre em Engenharia Elétrica

por

César Daltoé Berci

Prof. Dr. Celso Pascoli Bottura
Orientador - FEEC/UNICAMP/SP

30 de julho de 2008

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

B451o

Berci, César Daltoé
Observadores inteligentes de estado: propostas / César Daltoé Berci. --Campinas, SP: [s.n.], 2008.

Orientador: Celso Pascoli Bottura
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas não-lineares. 2. Inteligência artificial. 3. Teoria do controle não-linear. I. Bottura, Celso Pascoli. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Intelligent state observers: proposals
Palavras-chave em Inglês: Intelligent observers, Non linear systems, Computational intelligence
Área de concentração: Automação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: João Viana da Fonseca Neto, Romis Ribeiro de Faissol Attux
Data da defesa: 30/07/2008
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: César Daltoé Berci

Data da Defesa: 30 de julho de 2008

Título da Tese: "Observadores Inteligentes de Estado: Propostas"

Prof. Dr. Celso Pascoli Bottura (Presidente): _____

Prof. Dr. João Viana da Fonseca Neto: _____

Prof. Dr. Romis Ribeiro de Faissol Attux: _____

Resumo

O problema de se observar o estado do sistema nasce da necessidade de se controlar plantas cujo estado não se pode medir. Convencionalmente são utilizados observadores baseados em modelos dinâmicos para resolver esse problema, através da construção de estimativas do estado desconhecido. Nesta tese, é abordada uma solução alternativa para este problema. Várias propostas de observadores inteligentes, são realizadas com base em mapeamentos entre subespaços gerados pelas equações que regem a dinâmica não linear do sistema, utilizando várias técnicas de inteligência computacional, capazes de apresentar estimativas do estado do sistema dinâmico através de observadores que não apresentam um comportamento dinâmico com relação ao tempo.

Palavras-chave: Observadores Inteligentes, Sistemas Não Lineares, Inteligência Computacional.

Abstract

The state observer problem arises from the necessity of controlling plants whose state can not be measured. Usually dynamic model based observers are used to solve this problem, for estimating unknown state. In this thesis an alternative solution for this problem is presented. Various proposals for intelligent observers are made based on mappings between subspaces generated by the equations describing the nonlinear dynamics of the system, using various computational intelligence techniques able to present state estimates for the dynamic system through nonlinear observers that do not exhibit a dynamic behavior with respect to time.

Keywords: Intelligent Observers, Non Linear Systems, Computational Intelligence.

Para o homem só é impossível aprender aquilo que ele pensa que já sabe

Epíteto

Agradecimentos

Ao meu orientador, Prof. Celso Pascoli Bottura. sou grato pela orientação.

À minha esposa Rafaela dos Santos Costa Berci e aos meus pais, Antonio Berci e Nelci Ana Daltoé Berci, pelo apoio prestado.

Aos demais colegas de pós-graduação, pelas críticas e sugestões.

À minha esposa, meus pais, irmão, avós e tios

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

AG Algoritmo Genético

ART *Adaptive Resonance Theory*

BFGS Broyden Fletcher Goldfarb Shanno

CUO Completamente Uniformemente Observável

DFP Davidon Fletcher Powell

FR Fletcher Reeves

GE Gradiente Evolutivo

GP Gradiente das Partículas

MCP McCulloch and Pitts Perceptron

MLP *Multi Layer Perceptron*

OAN Observador Adaptativo Neural

OBE Observador Evolutivo

OBN Observador Neural

OEP Observador Baseado em Enxame de Partículas

PSO *Particle Swarm Optimization*

RNA Rede Neural Artificial

SCG Scaled Conjugated Gradient

Lista de Símbolos

n_g	-	Número de gerações
N	-	Número de variáveis
C^i	-	Cromossomo do *i-ésimo* indivíduo de uma população
H	-	Cromossomo herdeiro
t_{mut}	-	Taxa de Mutação
t_{rep}	-	Taxa de Reprodução
p_{mut}	-	Intensidade da mutação
x_i	-	Posição da *i-ésima* partícula
v_i	-	Velocidade da *i-ésima* partícula
p_i	-	Melhor posição da *i-ésima* partícula
p_g	-	Melhor posição entre todas as partículas da população
c_1, c_2	-	Coeficientes de aceleração das partículas
β_i	-	Valor de adaptação da *i-ésima* partícula
n_i	-	Número de iterações
n_p	-	Número de partículas da população
V_{max}	-	Limitante superior de velocidade das partículas
V_{min}	-	Limitante inferior de velocidade das partículas
w	-	Peso de inércia
χ	-	Fator de constrição
θ	-	Ângulo entre vetores
ω	-	Vetor de parâmetros livres de um rede neural artificial
W^k	-	Matriz de pesos entre as camadas k e $k-1$
B^k	-	Vetor de polarizações da camada k
n	-	Dimensão do vetor de estado do sistemas
m	-	Dimensão da entrada do sistema
p	-	Dimensão da saída do sistema
$\mathcal{X}$	-	Espaço de estados
$\mathcal{O}$	-	Espaço de observações
$d\mathcal{O}$	-	Cofatores de observação
$\mathcal{H}$	-	Difeomorfismo

Trabalhos Publicados Pelo Autor

1. Berci, C. D. & Bottura, C. P. Observador Neural Inteligente Não Baseado em Modelo Para Sistemas Não Lineares. *Artigo Aceito: Congresso Brasileiro de Automática, Juiz de Fora, MG*, 2008

2. Berci, C. D. & Bottura, C. P. Observador Inteligente Para Sistema Não Lineares Baseado em Inteligência de Enxame. *Artigo Aceito: Congresso Brasileiro de Automática, Juiz de Fora, MG*, 2008

3. Berci, C. D. & Bottura, C. P. Modulação de Velocidade em Otimização Por Enxame de Partículas. *Proceedings of 7th Brazilian Conference on Dynamics, Control and Applications*, 2008, 7, 241-248.

4. Berci, C. D. & Bottura, C. P. The Evolving Gradient: An Hybrid Algorithm Based on Genetic Algorithms and Backpropagation for Neural Networks Training. *Artigo Aceito: Fifth International Symposium on Neural Networks, Beijing, China*, 2008

5. Berci, C. D. & Bottura, C. P. Observador Inteligente Adaptativo Neural Não Baseado em Modelo Para Sistemas Não Lineares. *Proceedings of 7th Brazilian Conference on Dynamics, Control and Applications*, 2008, 7, 209-215

6. Berci, C. D. & Bottura, C. P. Speed Modulation: A New Approach to Improve PSO Performance. *Artigo Aceito: IEEE Swarm Intelligence Symposium 2008, Sheraton Westport at St. Louis, Missouri*, 2008

Capítulo 1

Introdução

Em meados do século passado houve uma grande mudança na área de controle, quando se começa a realizar controle por realimentação de estados. Essa abordagem do problema de controle traz inúmeros atrativos quando comparada ao controle no domínio da freqüência, conhecido até então. Por outro lado, como toda nova solução gera ao menos um novo problema, nasce aqui o problema de estimação de estado. Muitas vezes não se possui conhecimento a respeito de todas as componentes do vetor de estado, o que pode ocorrer por várias razões, pois, essas componentes podem não ser mensuráveis, ou medi-las pode não ser viável para a aplicação.

Inicialmente a solução encontrada para o problema de estimação de estado foi a utilização de dispositivos, intitulados observadores, que possuem a capacidade de estimar o estado desconhecido por meio de um segundo sistema dinâmico, construído com base no modelo dinâmico do sistema a ser observado. Essa abordagem, ainda hoje, é a principal ferramenta utilizada na estimação de estado.

Um requisito para a aplicação desses observadores é que o sistema possua uma propriedade, chamada observabilidade, que assegura que seu estado pode ser recuperado com base em sua saída mensurável, requisito este que deve ser avaliado *a priori* da concepção do observador. Existem na literatura, muitas técnicas que permitem ao projetista verificar a observabilidade de um sistema dinâmico qualquer, dentre essas as quais, encontram-se as condições geométricas de observabilidade. Essas condições são baseadas em subespaços gerados pelo estado e pela saída do sistema e na relação existente entre esses subespaços, que é dada pela dinâmica do sistema.

Este documento trata basicamente da aplicação de técnicas de inteligência computacional ao problema de observação de estado. Porém, este é abordado de forma diferente daquela estudada tradicionalmente. Aqui, através de uma análise geométrica do sistema dinâmico a ser observado, são descritos mapeamentos estáticos que relacionam a saída ao estado. A descrição desses mapeamentos, por sua vez, possibilita a aplicação de diversas ferramentas numéricas à estimação de estado, como é o caso das ferramentas de inteligência computacional.

São aqui estudadas três áreas da inteligência computacional, onde são descritas em detalhes algumas propriedades dessas ferramentas, empregadas na concepção de observadores inteligentes. Esse estudo tem o claro intuito de prover um melhor uso das capacidades apresentadas pelas diferentes técnicas analisadas. Por fim, o ultimo Capítulo traz várias propostas para construção de observadores inteligentes, não baseados em modelos, com base nesse mapeamento estático.

Os tópicos da inteligência computacional abordados neste documento são os seguintes:

- Computação Evolutiva.
- Inteligência de Enxame.
- Redes Neurais Artificiais.

No Capítulo 2, é mostrada uma introdução didática a respeito dos algoritmos genéticos, no qual são detalhados aspectos relativos a sua concepção e inspiração biológicas, de forma a criar uma base sólida sobre a qual são desenvolvidas novas propostas nessa área. Ainda nesse Capítulo, são expostos os operadores genéticos mais eficientes encontrados na literatura pertinente ao assunto, criando um documento que expressa o estado da arte nessa área. Posteriormente, são propostos novos métodos, que visam melhorar a eficiência dos algoritmos genéticos, no contexto de problemas de otimização de funções. Tais propostas são comparadas, com os métodos atualmente utilizados, visando com isso, prover sua validação.

Um tema relativamente mais novo, a otimização por enxame de partículas, um método heurístico de otimização baseado em inteligência de enxame, é discutido no Capítulo 3, no qual novamente é apresentado um texto didático que abrange desde a inspiração biológica e criação do algoritmo original, até um estudo dos principais métodos de ganho de eficiência encontrados na literatura. Com isso, espera-se familiarizar o leitor com esse tema, bem como apresentar o estado da arte nessa área, que vem se expandindo atualmente. É também discutida uma nova proposta para aumentar a eficiência do algoritmo: um método de modulação [1] que proporciona um significativo ganho, não somente com relação à taxa de convergência, como também em preservação da diversidade da população de partículas.

São também realizados alguns testes comparativos entre algoritmos genéticos e otimização por enxame de partículas, com o intuito de analisar de forma concreta as capacidades e deficiências relativas de cada um dos métodos, criando um fundamento sólido para futuras aplicações de ambas as técnicas.

As redes neurais artificiais, última ferramenta computacional a ser analisada, são descritas no Capítulo 4, no qual é apresentada um ampla introdução didática ao assunto, discutindo vários aspectos, inclusive históricos, pertinentes à área da computação conexionista. O foco principal desse Capítulo

são as redes neurais do tipo MLP, devido a sua melhor adequação ao problema de observação de estado, considerando a forma como este é aqui abordado. São analisados vários algoritmos de treinamento para essa arquitetura de redes neurais, sendo também propostos novos algoritmos baseados em meta-heurísticas. Os algoritmos propostos apresentam uma eficiência comparável à dos algoritmos procedurais mais eficientes conhecidos atualmente, porém, preservam as características positivas dos métodos de busca cega, podendo ser implementados em uma gama maior de problemas, abrangendo situações onde não é possível a utilização de métodos procedurais.

No Capítulo 5, é apresentada a abordagem geométrica da condição de observabilidade e descrito o mapeamento estático que relaciona a saída mensurável do sistema com o seu estado. A assimilação desse mapeamento através de alguma ferramenta de aproximação é sem dúvida a solução mais intuitiva para o problema de estimação de estado. Assim, são propostos dois observadores: *Observador Neural* e o *Observador Adaptativo Neural*. Baseados em redes neurais artificiais, esses observadores são capazes de assimilar o mapeamento em questão, que contrastando com o paradigma clássico de estimação de estado, não são baseados em um modelo dinâmico.

São ainda discutidas propostas em que de certa forma, tenta-se transformar o problema de estimação de estado em um problema de busca, com a clara intenção de aplicar meta-heurísticas de busca e otimização, como algoritmos genéticos e otimização por enxame de partículas, nessa estimação. Com base nesse conceito, são propostos outros dois observadores: *Observador Evolutivo* e *Observador Baseado em Inteligência de Enxame*, que fazem um uso alternativo do mesmo mapeamento estático.

Todos os observadores propostos são avaliados através de exemplos de aplicação baseados em experimentos computacionais. A qualidade dos resultados apresentados expressa a grande potencialidade desse paradigma de observação, ainda muito pouco explorado.

Capítulo 2

Algoritmos Genéticos

2.1 Darwin e a Seleção Natural

Durante séculos, a humanidade acreditou ter sido formada por um criador inteligente e onipotente, responsável pela criação do universo e toda sua existência. Essa teoria criacionista foi sustentada principalmente por religiosos, e ganhou força durante o intervalo de tempo conhecido como idade das trevas.

O principal trabalho a questionar essa hipótese de forma cientifica, foi elaborado por Charles Darwin[1], no século XIX, que em seu livro *On the Origin of Species by Means of Natural Selection* [2], apresenta uma teoria com a qual ele tenta explicar, de forma lógica, a existência de um número tão grande de seres vivos diferentes na natureza.

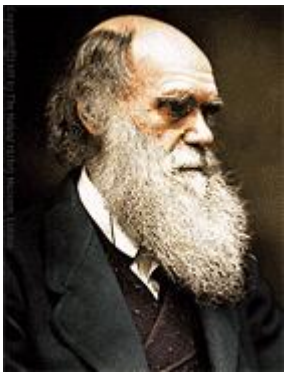

Figura 2.1: Charles Darwin

[1]Charles Darwin: naturalista britânico nascido na casa da sua família em Shrewsbury, Shropshire, Inglaterra, em 12 de fevereiro de 1809 vindo a falecer em Downe, Kent, Inglaterra, em 19 de abril de 1882

Darwin propôs a existência de um mecanismo, batizado por ele de seleção natural, dotado da capacidade de manutenção das variações favoráveis à sobrevivência das espécies no ambiente onde estas estão inseridas. Tais variações, acumuladas por um longo período de tempo, seriam capazes de prover o surgimento de organismos suficientemente distintos de seus antecessores, a ponto de serem caracterizados como uma nova espécie.

A existência desse mecanismo faz com que nem todos os organismos que nascem, sobrevivam ou reproduzam-se, mas somente uma porção destes organismos, melhor adaptados para enfrentar as adversidades ambientais do sistema no qual estão inseridos. Dessa forma, após várias gerações, as características favoráveis à adaptação do organismo serão favorecidas, enquanto as variações desfavoráveis tendem a desaparecer.

As teorias de Darwin resultam em uma lei geral de evolução que quando aplicada a uma população de organismos, resulta de forma emergente em uma nova população contendo organismos mais adaptados ao meio onde vivem. Este processo evolutivo pode ser descrito, de maneira formal, por meio do algoritmo mostrado a seguir.

```
Considerar uma população de organismos inicial;
while not fim do
    Reproduza os organismos;
    Varie aleatoriamente os organismos;
    Aplique a Seleção Natural
end
```

Algoritmo 1: Algoritmo evolutivo

Um exemplo clássico dessa teoria é a evolução das girafas. Os ancestrais das girafas, de acordo com o documentário fóssil, tinham pescoço significativamente mais curtos. O comprimento do pescoço variava entre os ancestrais das girafas e essa variação era de natureza hereditária.

Segundo Darwin, após várias gerações, a vegetação mais baixa tornou-se escassa, devido ao fato de esta ser fonte de alimento para toda a população, uma vez que todas as girafas possuíam condições físicas para se alimentar delas. Com o passar do tempo, imperou uma severa parcimônia de alimento para as girafas de pescoço mais curto, devido à escassez de vegetação baixa, resultando na morte desses indivíduos e conseqüente sobrevivência apenas das girafas de pescoço mais longo [3].

Neste exemplo, ilustrado na Figura 2.2, a seleção natural agiu de forma a privilegiar variações que culminaram em girafas de pescoço mais longo, pois estas se tornaram melhor adaptadas às diversidades do novo ambiente.

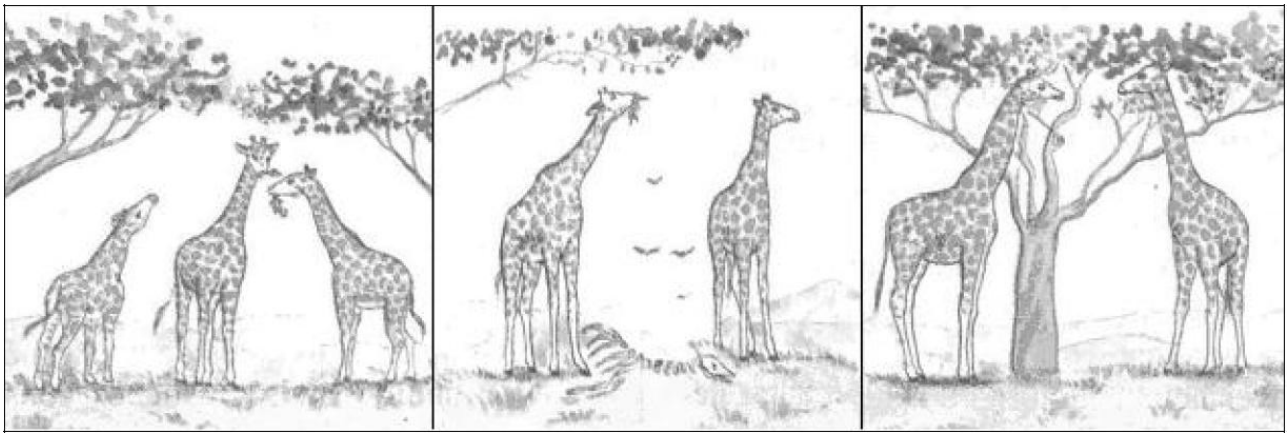

Figura 2.2: População de girafas ancestrais (à esquerda); Extinção das girafas de pescoço mais curto (ao centro); População evoluída apenas com girafas de pescoço mais longo (à direita).

2.1.1 Histórico

As idéias de Darwin foram o objeto de estudo de vários pesquisadores durante os anos de 30 e 40 do século XX, que desenvolveram o princípio da preservação da variabilidade por meio da mutação e recombinação genética. Posteriormente, nos anos de 50 e 60, biólogos e matemáticos desenvolveram simulações computacionais de sistemas evolutivos com base nesses estudos.

Dando seqüência a esses eventos, John Holland, professor da Faculdade de Engenharia Elétrica e Computação da Universidade de Michigan, juntamente com seus alunos e colegas professores, desenvolveram em meados da década de 60, um método heurístico o qual foi chamado de algoritmo genético, tendo como objetivo estudar formalmente o fenômeno de adaptação ocorrido na natureza, para com isso assimilar seus mecanismos e traduzi-los para processos computacionais.

Em seu livro *Adaptation in Natural and Artificial Systems* [4], Holland descreve os conceitos básicos dos algoritmos genéticos e em 1989, é publicado o livro *Genetic Algorithms in Search, Optimization & Machine Learning* [5], no qual David Goldberg trata da maior parte dos tópicos de relevante mérito na área. Atualmente, estes dois livros são as referências mais importantes sobre algoritmos genéticos, e sua aplicação se estende aos mais diversos problemas na área de otimização e aprendizado de máquina.

2.2 Base Biológica

Em busca de sintetizar formalmente o processo biológico de evolução, algumas definições e conceitos tornam-se indispensáveis para um correto entendimento e tradução dos mecanismos envolvidos neste processo.

2.2.1 Terminologia Biológica

A estrutura básica que descreve um organismo vivo é o cromossomo, formado por uma cadeia de DNA (*Deoxyribonucleic acid*). Cada espécie apresenta um número característico de cromossomos, como pode ser visto na Tabela 2.1, da qual conclui-se que o número de cromossomos de uma espécie não tem relação direta com seu grau de evolução relativo a outras espécies.

São observadas duas estruturas básicas para o arranjo dos cromossomos: Diplóides e Haplóides. Organismos diplóides apresentam cromossomos em pares, onde cada um deles é oriundo de uma célula somática. Esta configuração é comum em organismos que se reproduzem sexuadamente. Haplóides são organismos que apresentam apenas um cromossomo, como os gâmetas, que são células reprodutoras.

Espécie	$2n$ Cromossomos	Espécie	$2n$ Cromossomos
Drosófila	8	Humano	46
Centeio	14	Macaco	48
Coelho	44	Rato	44
Avoante	16	Cavalo	64
Caracol	24	Galo	78
Minhoca	32	Carpa	104
Porco	40	Borboleta	380
Trigo	42	Samambaia	1200

Tabela 2.1: Número de Cromossomos em Diferentes Espécies

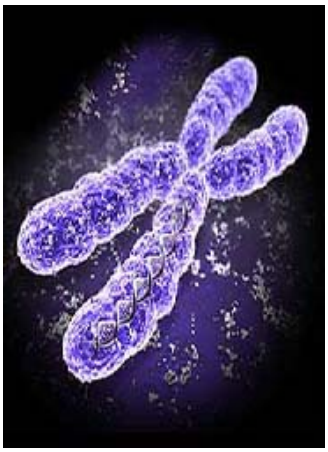

Figura 2.3: Cromossomo

Cada cromossomo de um ser vivo é formado por genes, que são blocos funcionais de DNA. Cada gene está localizado em uma posição (*locus*) particular de um cromossomo.

Um gene, ocupando um dado locus, pode apresentar várias formas alternativas. À essas formas dá-se o nome de alelo. A título de exemplo, consideremos o gene que determina a cor de uma certa flor. Esse único gene apresenta diferentes versões, ou alelos diferentes, um podendo resultar em flores brancas e outro em flores vermelhas.

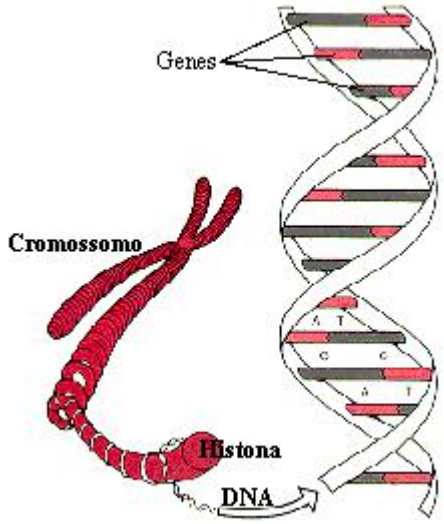

Figura 2.4: Cromossomo, DNA, gene

O conjunto de cromossomos existente no interior do núcleo das células carrega a herança genética do indivíduo, herdada de seus genitores. Como os cromossomos são compostos de genes, é possível concluir que esta informação está armazenada em um conjunto dessas estruturas. À esse conjunto de genes dá-se o nome de genótipo.

O genótipo carrega informação a respeito de várias características do indivíduo. Na espécie humana, por exemplo, pode-se citar a cor dos olhos, dos cabelos, altura, entre outros. A manifestação destas características, porém, é muitas vezes influenciada pela ação do meio, gerando manifestações causadas pela expressão do genótipo somada à interação do indivíduo com o ambiente em que se encontra. Essa manifestação é conhecida como fenótipo e como esta gera fatores observáveis, a genética clássica utiliza-se do fenótipo de um indivíduo para deduzir conclusões a respeito de seu genótipo.

Uma tentativa de modelar esse fenômeno de manifestação genotípica no fenótipo de um indivíduo, é proposta por Atmar [6], que explica os resultados encontrados até então na literatura através do emprego de dois espaços de estado, o espaço genótipo G e o espaço fenótipo F.

Utilizando estes espaços, são definidas 4 funções que relacionam os indivíduos de uma população com as mudanças por esta sofridas a cada nova geração. As funções em questão são as seguintes [7]:

- $f_1 : I \times G \rightarrow F$. Esta função representa a epigênese[2], onde ocorre a formação de novos indivíduos baseados na geração anterior. Aqui I representa um conjunto de efeitos do ambiente.
- $f_2 : F \rightarrow F$. Seleção natural baseada nas manifestações no fenótipo dos indivíduos.
- $f_3 : F \rightarrow G$. Sobrevivência genotípica.
- $f_4 : G \rightarrow G$. Variações genéticas aleatórias, mutações.

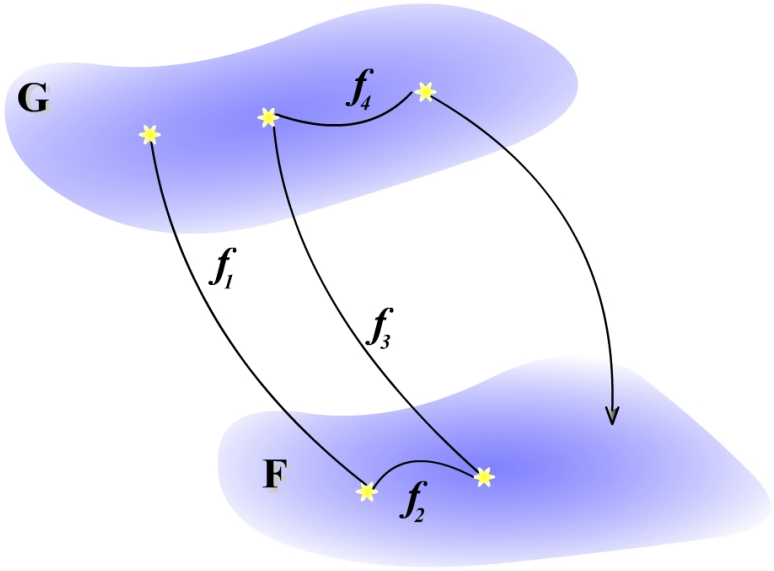

Figura 2.5: Processo evolutivo com relação aos espaços genótipos e fenótipos

2.2.2 Hereditariedade

Durante aproximadamente sete anos, Gregor Mendel[3] estudou variações no aspecto de plantas, como a cor das flores, e propôs que a existência destas características é proveniente de um par de unidades elementares de hereditariedade.

Após a morte de Mendel, suas idéias foram redescobertas, e estas unidades elementares hoje são conhecidas como genes. Um dos resultados mais importantes revelados pelo trabalho de Mendel é observado na 1^a *lei de Mendel*, que sumariza a transmissão de características de forma hereditária para as novas gerações.

[2]Processo de geração no qual o embrião é constituído por uma série de formações novas ou diferenciações sucessivas do ovo ou espermatozóide, nos quais não existem quaisquer esboços, mesmo rudimentares, da futura organização do indivíduo resultante da união daqueles elementos.

[3]Gregor Johann Mendel, Heizendorf, 20 de Julho de 1822 - Brno, 6 de Janeiro de 1884: Monge agostiniano, botânico e meteorologista austríaco.

Esta lei afirma que dois fatores hereditários (alelos) de uma característica (gene) segregam-se independentemente para os gâmetas, de modo que a metade destes possui um dos fatores, e a outra metade possui o outro fator da mesma característica.

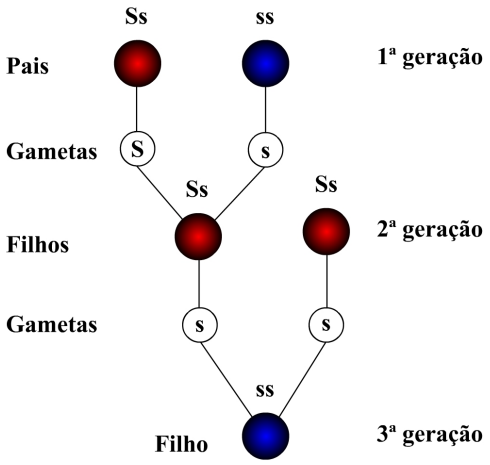

Figura 2.6: Herança genética segundo Mendel

Com isso, Mendel descreveu de maneira formal, o mecanismo pelo qual indivíduos de uma população herdam características das gerações anteriores, explicando como fatores que desaparecem em uma geração podem ressurgir em gerações posteriores.

2.2.3 Reprodução

Milhares de anos em constante processo de evolução, levaram ao surgimento de várias espécies de seres vivos, com muitas diferenças entre elas, sendo uma delas o meio pelo qual os organismos se reproduzem. Basicamente, existem apenas duas formas distintas de reprodução: assexuada e sexuada.

Na reprodução assexuada, o organismo filho possui apenas um pai, e com isso herda sua história genética apenas desse indivíduo. Existe um grande número de seres vivos dotados deste meio de reprodução, dentre os quais pode-se citar a maior parte das bactérias.

Os organismos que se reproduzem sexuadamente, por outro lado, possuem dois pais e sua informação genética é uma combinação da informação apresentada pelos pais. Esse mecanismo provê uma maior diversidade à população e, com isso, uma maior chance de gerar indivíduos mais bem adaptados ao meio.

A recombinação das informações genéticas dos pais, ocorre por meio de um processo conhecido como: *crossover*, que consiste em uma troca aleatória de material genético entre dois cromossomos distintos, ocorrida durante a meiose celular.

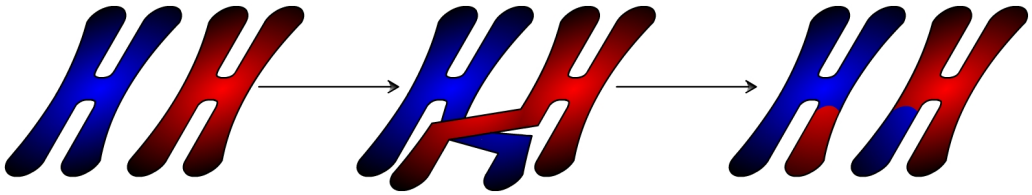

Figura 2.7: Recombinação genética: *crossover*

2.2.4 Mutação

Na biologia, mutações são mudanças na seqüência de pares de base do DNA de um organismo. Essas alterações são aleatórias e pontuais, envolvendo a eliminação ou substituição de um ou de poucos alelos.

Para que ocorra a mutação, é preciso primeiramente que haja um dano na seqüência de DNA, que é protegida por vários mecanismos de prevenção de falhas. Muitas vezes, porém, esses mecanismos de proteção fracassam em eliminar o defeito, ou o dano é simplesmente irreversível. Nesta situação, a célula se replica e, se esta nova célula replicada sobreviver com a estrutura alterada em seu DNA e, por sua vez, continuar o processo de replicação, então esta alteração será transmitida, e assim uma nova geração de células mutantes será formada.

As mutações formam um mecanismo crucial na evolução das espécies. Alterações morfológicas, frutos das manifestações fenotípicas das mutações, são a base da moderna teoria da seleção natural, pois promovem o surgimento de novas características na população, possibilitando o surgimento de elementos melhor adaptados ao meio onde vivem. Essas mutações podem também criar organismos menos adaptados que a média da população, neste caso, o mecanismo de seleção tende a eliminar esses organismos, privilegiando apenas as alterações favoráveis à adaptação dos indivíduos.

2.3 Algoritmos Genéticos

Algoritmos genéticos são métodos de busca e otimização baseados em processos biológicos de evolução. Estes algoritmos foram desenvolvidos tendo basicamente dois objetivos:

- Abstrair de maneira formal e rigorosa processos biológicos envolvidos na evolução dos seres vivos, possibilitando um melhor entendimento dos organismos vivos e de suas origens.
- Utilizar computacionalmente a capacidade existente nos mecanismos de evolução natural, desenvolvendo sistemas evolutivos capazes de resolver problemas complexos baseados na evolução de agentes simples.

Estes algoritmos utilizam uma população de soluções candidatas ao problema, sendo cada solução representada por um indivíduo da população. Estes indivíduos, assim como em um ambiente natural, encontram-se sujeitos a interações com o meio onde estão inseridos. O processo de evolução acontece de forma continuada e simultânea com relação a todos indivíduos da população, caracterizando uma estratégia paralela de busca.

Assim como em processos naturais de evolução, os algoritmos genéticos estão sujeitos a variáveis probabilísticas e aleatórias. A interação dessas variáveis gera um processo essencialmente estocástico, característica essa presente nos algoritmos genéticos e contrastante com os métodos de busca e otimização procedurais[4], que são, em sua maioria, determinísticos.

É possível reconhecer a existência de várias diferenças entre o paradigma evolutivo e os métodos tradicionais de busca e otimização. Dentre as características mais relevantes presentes nos algoritmos genéticos que contrastam com métodos procedurais, as seguintes podem ser citadas:

- Algoritmos genéticos atuam sobre codificações do conjunto de parâmetros, como indivíduos e populações e não sobre os próprios parâmetros. Esse método tem por objetivo evoluir sua população, e não otimizar uma dada função de custo, o que é alcançado através da escolha adequada da codificação.

- Métodos baseados no princípio da evolução baseiam-se na adaptação do indivíduo como fruto de sua interação com o meio, e não em derivadas ou outro conhecimento mais criterioso do problema. Essa característica faz com que os algoritmos genéticos sejam considerados algoritmos de busca cega.

- A seleção e a evolução são processos probabilísticos e aleatórios, assim, os algoritmos genéticos não são caracterizados por um processo determinístico como são os métodos procedurais.

Esses algoritmos são considerados ferramentas de busca global, dada a maior probabilidade de se encontrar uma solução globalmente ótima da função em comparação a algoritmos procedurais, que apresentam uma grande probabilidade de encontrar um ótimo local mais próximo de onde o método foi inicializado.

A título de exemplo, consideremos a função de Rastrigin para 2 variáveis x_1 e x_2:

$$f = x_1^2 + x_2^2 + 10\cos(2\pi x_1) + 10\cos(2\pi x_2) + 20 \tag{2.1}$$

[4]Neste documento, métodos baseados em meta-heurísticas são ditos não procedurais dada a sua inspiração e comportamento emergente, porém, os referidos métodos são implementados de forma sequencial, o que poderia classifica-los como procedurais em algum outro contexto que não o deste documento.

Essa função possui um mínimo global em $x_1 = 0$ e $x_2 = 0$, e vários mínimos locais devido às funções cosseno. O processo de otimização através do algoritmo do gradiente, aplicado por 20 iterações, obtém o resultado mostrado na figura 2.8, em que foi encontrada uma solução sub-ótima local, próxima ao ponto $x_1 = -1$ e $x_2 = 1$.

Ao contrário de métodos procedurais, como o método do gradiente, a aplicação de um algoritmo genético a esse problema apresenta uma probabilidade muito maior de encontrar o mínimo global da função. Consideremos um caso ilustrativo de aplicação de um algoritmo genético a esse problema. Para tanto, foi gerada uma população inicial aleatória contendo 100 indivíduos, descrita na figura 2.9, onde cada ponto representa um indivíduo da população.

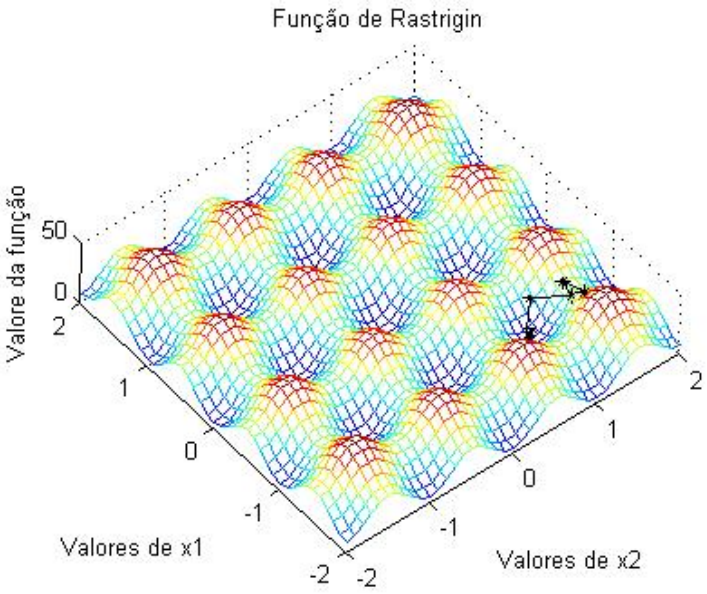

Figura 2.8: Otimização da função de Rastrigin, método do gradiente

Aplicando métodos de seleção, reprodução e mutação a essa população, após 20 gerações, o posicionamento dos indivíduos no espaço de busca é bastante alterado, assumindo a configuração dada pela figura 2.10.

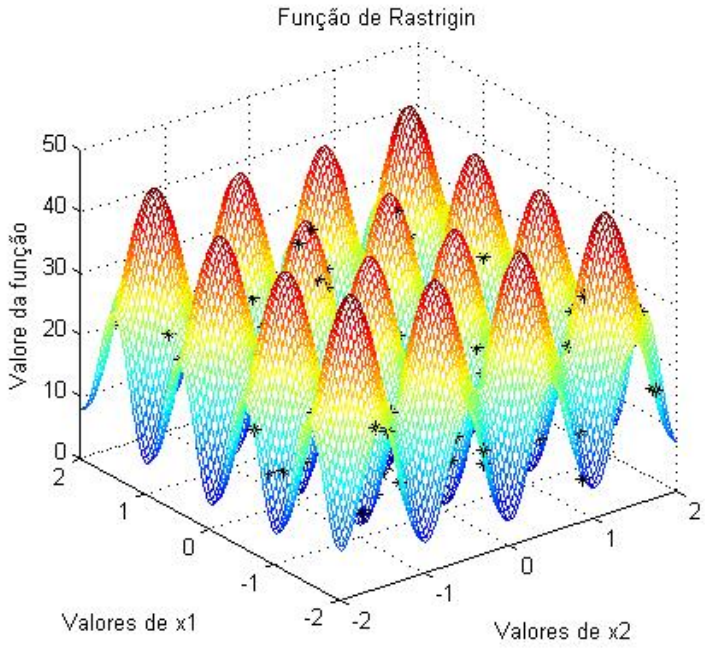

Figura 2.9: Otimização da função de Rastrigin, população inicial do algoritmo genético

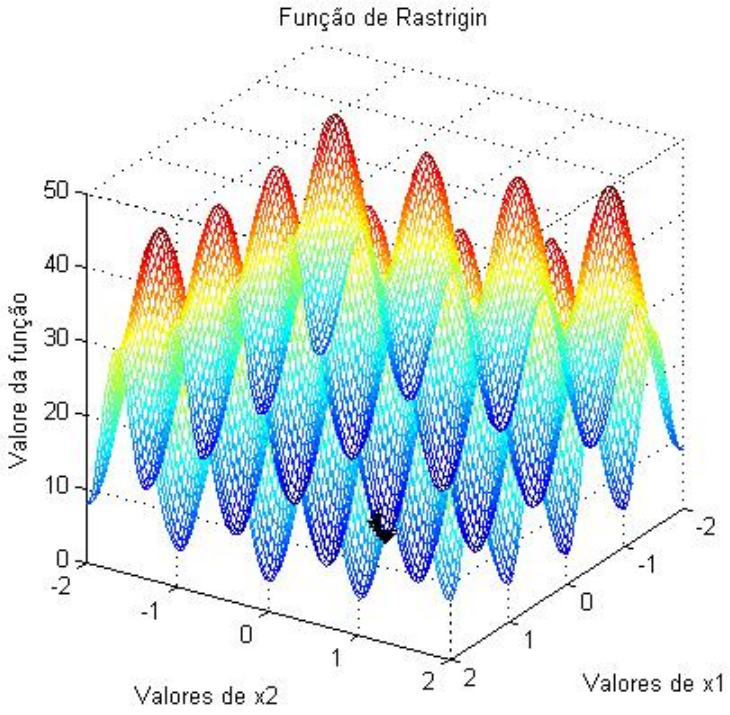

Figura 2.10: Otimização da função de Rastrigin, população após 20 gerações

Neste exemplo, o algoritmo genético utilizado encontrou o mínimo global da função de Rastrigin, tarefa essa que seria muito difícil de ser completada por um algoritmo procedural de otimização, dado que a função objetivo apresenta um número muito grande de mínimos locais dentro do espaço de busca. Dessa forma, é grande a probabilidade de um ponto inicial aleatório encontrar-se na vizinhança de um desses ótimos locais.

2.4 Algoritmo Genético Básico

Em [5], o autor introduz a estrutura básica de funcionamento de um algoritmo genético, explicitando cada etapa do algoritmo. Mais detalhadamente, esse algoritmo pode ser descrito pela sequência de passos:

- É gerada uma população inicial formada por um conjunto aleatório de indivíduos, onde cada indivíduo representa uma possível solução para o problema.
- É iniciado o processo evolutivo: cada indivíduo da população é avaliado e recebe um conceito, chamado de *fitness*, que reflete a qualidade da solução por ele representada. Em processos naturais, essa avaliação representa o quão adaptado um organismo esta em relação ao meio onde ele vive.
- São selecionados membros da população com base no conceito que estes receberam na etapa anterior. Parte dos indivíduos é mantida para a nova geração. Enquanto isso, os indivíduos menos adaptados não são selecionados e desaparecem da população, em um processo que tende a extinguir variações desfavoráveis.
- São escolhidos pares para reprodução também com base em seu fitness. Esses indivíduos se reproduzem e deixarão descendentes na nova geração.
- A nova população é sujeita a mutações aleatórias, visando introduzir variabilidade nos indivíduos. Essas variações serão avaliadas na próxima geração, onde características favoráveis à solução do problema serão privilegiadas.
- O processo evolutivo continua criando novas gerações até que um dado critério seja atendido e uma solução satisfatória seja encontrada.

Esse algoritmo, apesar de parecer bastante simples do ponto de vista biológico, representa uma poderosa ferramenta de busca adaptativa e robusta [3], de grande poder computacional. A Figura 2.11 traz um fluxograma que ilustra de maneira bastante clara o algoritmo genético aqui apresentado.

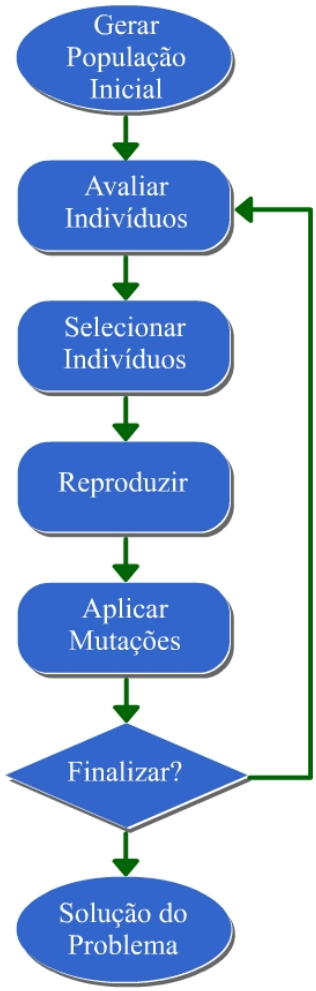

Figura 2.11: Algoritmo genético

2.5 Codificações

Uma das etapas mais complexas e que despende maior tempo e trabalho na aplicação dos algoritmos genéticos, é a fase de codificação do problema. Como já foi dito, os algoritmos genéticos atuam sobre a codificação do problema, evoluindo os indivíduos da população, sem levar em conta o que eles representam e sim seu nível de adaptação medido pelo seu fitness.

A codificação consta basicamente da definição de como os cromossomos irão representar as possíveis soluções, funções de fitness para avaliar a adaptação de cada indivíduo, operadores genéticos de mutação e reprodução e mecanismos de seleção. O termo codificação se aplica normalmente de forma mais restrita, tratando apenas da representação dos cromossomos. Neste documento optou-se por uma definição mais ampla da codificação, devido à variações necessárias para que o algoritmo

possa ser implementado em diferentes tipos de problemas, como por exemplo: otimização de funções, clusterização de dados, entre outros.

Serão analisadas aqui duas codificações, binária e real, porém, existe ainda a codificação inteira, que não é aqui discutida, pois sua gama de aplicações não coincide com os objetivos deste documento.

2.5.1 Codificação Binária

Nesta codificação os cromossomos são representados por cadeias binárias (strings binárias) de comprimento fixo l. Nessas cadeias, cada *bit* ocupa um *locus* do cromossomo e representa um gene que pode então possuir dois alelos: 0 e 1.

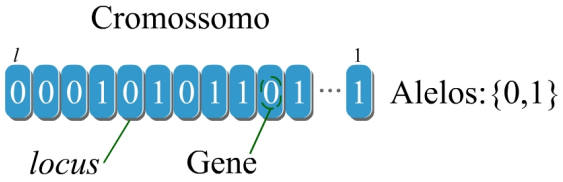

Figura 2.12: Algoritmo genético

Esta foi a primeira codificação para os cromossomos criada para um algoritmo genético, em parte pela semelhança com cromossomos biológicos, por apresentar com clareza os conceitos de Gene, Alelo e *Locus* e em parte também pela própria natureza da computação digital. Devido a esses fatores, essa codificação é conhecida na literatura também como codificação clássica.

Quando se opta por trabalhar com essa representação, é preciso converter a cadeia binária em um valor numérico pertencente ao espaço de busca do problema ao qual o algoritmo será aplicado, a menos que o problema seja intrinsecamente binário. Esse processo pode ser realizado conforme é indicado na equação (2.2), quando o espaço de busca do problema é um subespaço do R^N. Essa equação faz o cálculo do mapeamento da cadeia binária em um intervalo real utilizando a codificação de ponto fixo:

$$R_i = \min + \frac{B_i}{2^l - 1}(\max - \min) \tag{2.2}$$

onde:
R_i: é o valor real que representa o cromossomo i;
B_i: valor binário em ponto fixo do cromossomo i;
min: menor valor do intervalo real de interesse;
max: maior valor do intervalo real;

2.5.2 Codificação Real

Esta codificação representa os cromossomos por números ou vetores reais. Nota-se aqui, que são perdidas a noções de *Locus*, Alelo e Gene, quando utilizada esta codificação.

Por outro lado, essa representação permite cobrir um domínio bastante abrangente [8], o que seria difícil trabalhando com cadeias binárias, pois, para aumentar o domínio sem perder precisão, faz-se necessário aumentar o comprimento das cadeias binárias, o que causa um conseqüente aumento exponencial em complexidade ao problema. Além deste fator, essa codificação permite que a função de *fitness* seja atribuída diretamente ao cromossomo, sem necessidade de conversão.

São possíveis ainda várias outras codificações, visto que os cromossomos são sempre representados por símbolos de um alfabeto pertencente a uma dada linguagem.

2.5.3 Mecanismos de Seleção

Existem vários mecanismos de seleção propostos na literatura, que podem ser divididos em três grandes grupos. O primeiro grupo contém mecanismos baseados em amostragens diretas, onde são escolhidos os indivíduos a partir de algum critério determinístico. Existem também mecanismos de amostragem aleatória equiprovável, onde são escolhidos indivíduos de forma totalmente aleatória. Por fim, é possível ainda citar os métodos estocásticos de amostragem.

Neste documento serão analisados três tipos de seleção: Roleta, Elitista e Torneio.

1. **Roleta**. Este é um método estocástico de seleção proposto por Goldberg [5], no qual o *fitness* dos indivíduos é convertido em uma fração de uma roleta com área proporcional a esse *fitness*. Assim, um indivíduo com maior *fitness* representa uma maior porção da roleta. Após *girada*, essa roleta irá parar em uma posição aleatória, que aponta para um determinado indivíduo, o qual será então escolhido. A título de exemplo, consideremos os seguintes indivíduos pertencentes à população exposta na Tabela 2.2 e seus respectivos *fitness*:

Indivíduo	Fitness	Fatia da Roleta
Indivíduo 1	2	20% - 72^o
Indivíduo 2	1	10% - 36^o
Indivíduo 3	4	40% - 144^o
Indivíduo 4	3	30% - 108^o

Tabela 2.2: *Fitness* dos Indivíduos

A Figura 2.13 ilustra a divisão da roleta para os indivíduos descritos na Tabela 2.2.

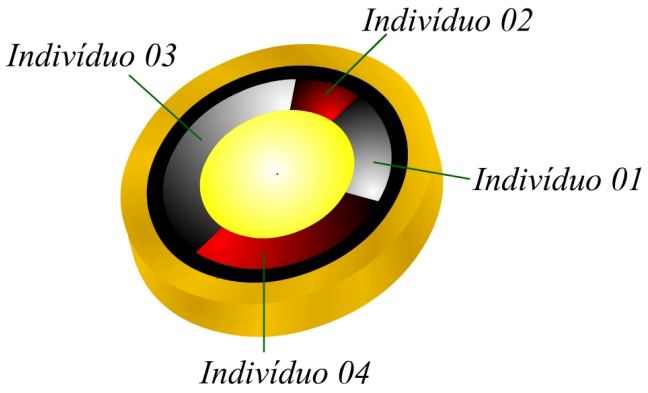

Figura 2.13: Divisão da roleta

O processo de seleção é completado *girando* a roleta 4 vezes, o que resulta em:

Iteração	Resultado	Indivíduo Escolhido
01	109^o	3
02	53^o	1
03	281^o	4
04	174^o	3

Tabela 2.3: Resultados da roleta

Dessa forma, o indivíduo 3 foi escolhido duas vezes, enquanto o indivíduo 2, que possui uma fatia muito pequena da roleta, não foi selecionado e desaparecerá da próxima geração.

Este mecanismo é bastante interessante do ponto de vista computacional, devido a sua característica estocástica que prove uma boa diversidade para a próxima geração, além do fato de não ser provável que o pior indivíduo seja selecionado para a próxima geração. Por outro lado, um indivíduo pode se tornar muito mais adaptado que os demais, transformando-se em um *super-indivíduo*, que será selecionado em quase todas as vezes que for aplicada a roleta.

Este problema cria uma convergência prematura do método, muitas vezes podendo resultar em uma solução indesejada. Portanto, devem ser adotados mecanismos para prevenir essa convergência, como limitar o número de vezes que um mesmo indivíduo pode aparecer na próxima geração.

2. **Torneio**. Neste método, cada indivíduo da próxima geração é selecionada através de um torneio entre dois ou mais indivíduos selecionados com base na roleta, onde o mais adaptado entre eles (indivíduo de maior *fitness*), será escolhido para a próxima geração. Consideremos novamente a população discutida anteriormente. Para exemplificar esse método, é mostrado na Tabela (2.4) o resultado da seleção por torneio.

1^a Roleta	2^a Roleta	Indivíduo Escolhido
3	1	3
4	2	4
3	4	3
1	3	3

Tabela 2.4: Resultados do torneio

O método de seleção por torneio visa prevenir, porém não garante, a escolha de indivíduos pouco adaptados. Ele também leva à uma convergência prematura, ainda mais acentuada do que aquela observada no método da roleta, tornando-se preferível quando essa convergência for desejável.

3. **Elitista**. Este é um método bastante simples de seleção, baseado em um critério determinístico. Nele são escolhidos os n indivíduos melhor adaptados para a próxima geração. Esse critério de seleção, apesar de bastante simplista, pode ser utilizado em conjunto com o método da roleta para prevenir, tanto a convergência prematura quanto a perda do indivíduo melhor adaptado, levando, em muitas situações, a bons resultados.

2.5.4 Reprodução

Nos algoritmos genéticos, o processo de reprodução tenta implementar as características naturais da reprodução dos organismos, ou seja, recombinar o material genético dos pais para gerar um filho, conseguindo com isso propagar a herança genética dos pais para a próxima geração.

Codificação Binária

Em sistemas biológicos, o processo de recombinação do material genético, na reprodução sexuada, se dá por meio de crossover. Neste mecanismo ocorre o sobrecruzamento dos cromossomos dos pais, gerando um novo cromossomo, que é a combinação de outros dois.

Quando utilizando a codificação binária, esse processo é computacionalmente reproduzido, principalmente de sua formas, através do cruzamento de um ou dois pontos.

- Cruzamento de um ponto (*Single Point Crossover*)[9]. Um ponto de cruzamento é escolhido aleatoriamente dentro da *string* que representa o cromossomo. Baseado neste ponto, as cadeias binárias são quebradas e recombinadas em um novo cromossomo, formado pelas partes das cadeias que representam os cromossomos pais. Um exemplo deste tipo de recombinação pode ser visto na figura 2.14.

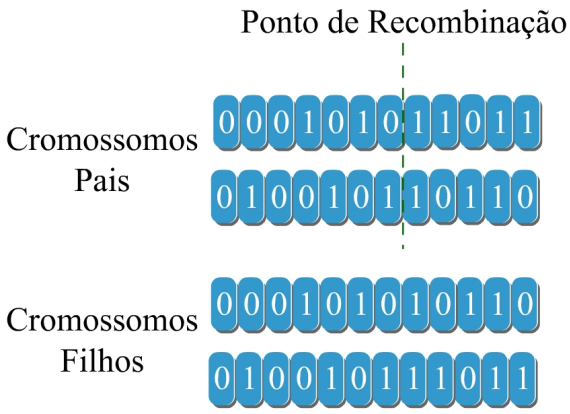

Figura 2.14: Cruzamento de um ponto

- Cruzamento de dois pontos (*Two Point Crossover*)[3]. Basicamente apresenta o mesmo mecanismo do cruzamento mostrado anteriormente, porém, neste caso são utilizados dois pontos de crossover, conforme ilustra a figura 2.15.

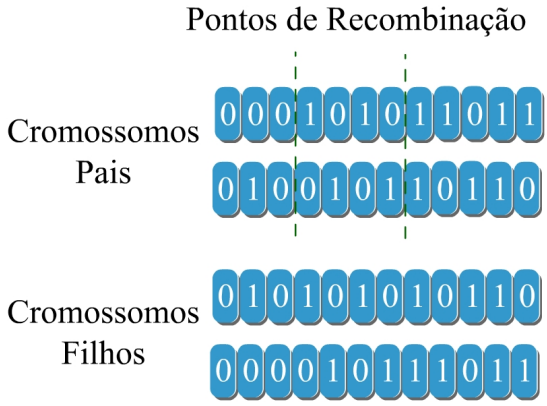

Figura 2.15: Cruzamento de dois pontos

Codificação Real

Consideremos dois indivíduos pais representados pelos cromossomos: $C^1 = (c_1^1, c_2^1, ..., c_N^1)$ e $C^2 = (c_1^2, c_2^2, ..., c_N^2)$, selecionados em uma população representada por pontos do espaço R^N, onde todo $c_i^j \in R$ é um número real. A reprodução desses cromossomos pode ser realizada de inúmeras maneiras, tendo cada uma delas, características próprias favoráveis a aplicações diferentes.

São observadas na literatura algumas propostas de reprodução voltadas à codificação real, sendo várias delas aqui tratadas, além de duas novas propostas aqui apresentadas com objetivo de melhorar a performance do algoritmo em determinados problemas. A seguir, são apresentadas as principais propostas de reprodução encontradas na literatura.

- **Flat Crossover** [10]. Neste método de reprodução, o cromossomo filho $H = (h_1, h_2, ..., h_N)$ é gerado componente a componente através da seguinte equação:

$$h_i = c_i^1 + r(c_i^2 - c_i^1) \ \ \forall i \in \{1...N\} \tag{2.3}$$

onde r é um número aleatório tal que: $r \in [0, 1]$. Dessa forma, cada componente h_i do vetor H é escolhida aleatoriamente no intervalo: $[c_i^1, c_i^2]$.

Consideremos agora o subespaço S, formado pelos possíveis filhos de um determinado par de cromossomos C_1 e C_2. Para o método de reprodução em questão, este subespaço formará um hipercubo limitado pelos cromossomos pais. A Figura 2.16 ilustra o subespaço S, formado por dois cromossomos pertencentes ao R^2.

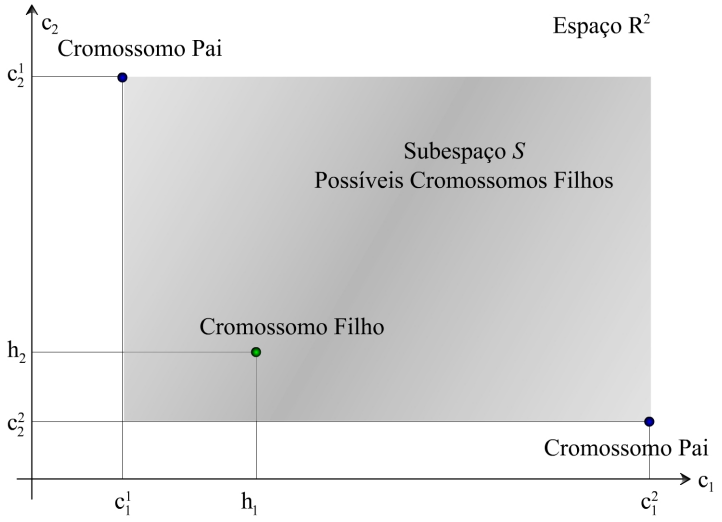

Figura 2.16: Cruzamento *flat*

Dessa forma o cruzamento *flat* produz herdeiros uniformemente distribuídos no espaço existente entre os pais.

- **Crossover Aritmético** [11]. Similar ao Flat Crossover, porém, é determinado um parâmetro r constante e dois cromossomos filhos H^1 e H^2 são gerados a partir dele como segue:

$$\begin{aligned} h_i^1 &= c_i^1 + r(c_i^2 - c_i^1) \ \ \forall i \in \{1...N\} \\ h_i^2 &= c_i^2 + r(c_i^1 - c_i^2) \ \ \forall i \in \{1...N\} \end{aligned} \tag{2.4}$$

- **BLX-α** [12]. Outro método similar ao Flat Crossover, onde são escolhidos componentes h_i para os cromossomos filhos dentro do intervalo: $[c_{min} - I\alpha, c_{max} + I\alpha]$, onde: $c_{min} = \min(c_i^1, c_i^2)$ e $c_{max} = \max(c_i^1, c_i^2)$.

  Esse processo pode expandir (ou contrair para $\alpha < 0$) o subespaço S, possibilitando um aumento da diversidade da população, que poderá se expandir mais facilmente para regiões fora do subespaço onde o método foi inicializado. É possível notar também que para $\alpha = 0$, o método será idêntico ao Flat Crossover.

- **Crossover Simples** [13]. Um ponto de crossover é escolhido e dois novos cromossomos são gerados pela recombinação das componentes dos cromossomos pais, como segue:

$$\begin{aligned} H_1 &= (c_1^1, ..., c_r^1, c_{r+1}^2, ..., c_N^2) \\ H_2 &= (c_1^2, ..., c_r^2, c_{r+1}^1, ..., c_N^1) \end{aligned}$$

  onde r é um número aleatório inteiro escolhido entre 1 e N.

  O subespaço S dos possíveis herdeiros para este método, é então representado por um espaço discreto contendo $2N$ pontos. A figura 2.17 ilustra um exemplo de aplicação do operador em questão para o espaço R^2.

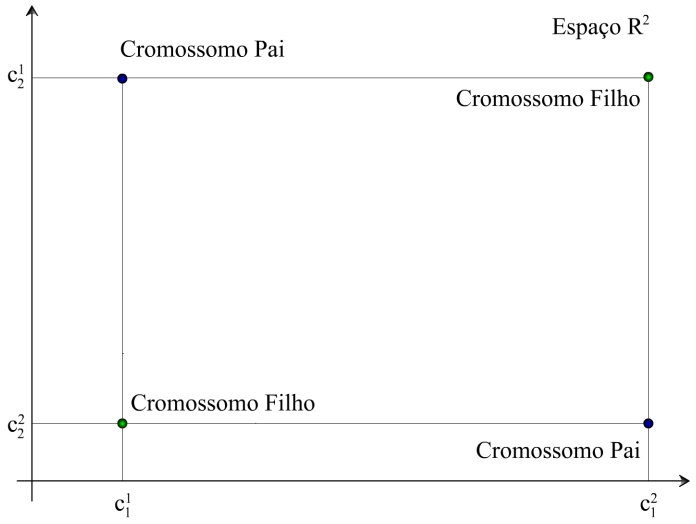

Figura 2.17: Crossover simples

Este método tenta preservar características do crossover biológico, assim como é feito na codificação binária, porém, o método gera uma subespaço discreto dentro de um espaço de busca

continuo, o que reduz de um valor que tende a infinito o número de possíveis herdeiros. Essa drástica redução do número de elementos do subespaço S, pode em algumas situações levar a uma perda de diversidade da população.

- **Crossover de Linha Estendida** [14]. Neste método de recombinação, é escolhido um escalar $\alpha \in [-0.25, 1.25]$ aleatoriamente e o cromossomo filho é então expresso pela seguinte equação:

$$H = C^1 + \alpha(C^2 - C^1) \tag{2.5}$$

O subespaço S que representa os possíveis cromossomos filhos, é formado por um segmento de reta que passa pelos cromossomos pais e se estende por 25% do comprimento da distância entre os dois, em cada uma de suas extremidades, assim como é ilustrado na figura 2.18.

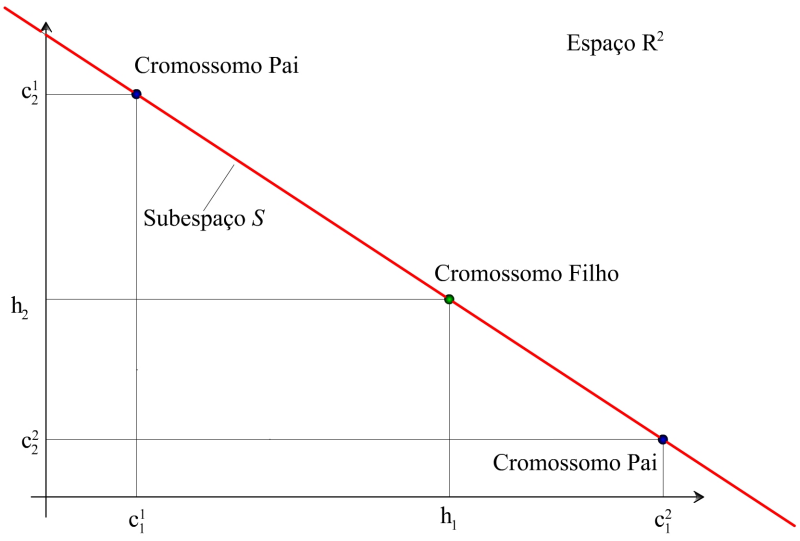

Figura 2.18: Crossover de linha estendida

Este é um processo de cruzamento bastante simples, e muito utilizado em problemas de otimização irrestrita. Assim como o método BLX-α, este método também tem a capacidade de expandir o espaço de busca, para, com isso, encontrar soluções fora do subespaço onde o método foi inicializado.

- **Crossover Heurístico de Wright** [13]. Este método de cruzamento leva em conta o *fitness* do melhor indivíduo entre os dois pais, tentando gerar herdeiros mais semelhantes a esse pai melhor adaptado.

  Consideremos aqui que o cromossomo C^1 possui *fitness* mais elevado; sendo assim, o herdeiro H será dado pela seguinte equação:

$$H = C^1 + r(C^1 - C^2) \tag{2.6}$$

onde r é um número aleatório pertencente ao intervalo: $[0, 1]$.

Este método vai afastar o herdeiro do cromossomo C^2 (menos adaptado), gerando cromossomos na região descrita na figura 2.19

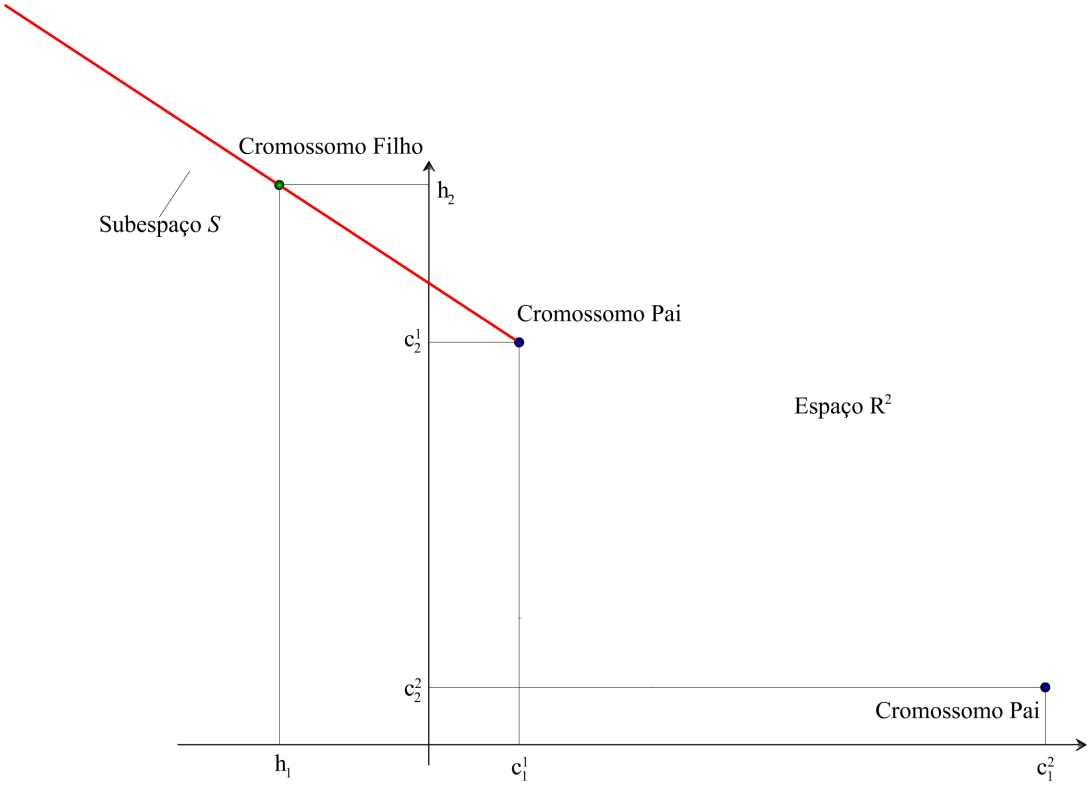

Figura 2.19: Crossover heurístico de Wright

Dessa forma o espaço de busca pode ser expandido consideravelmente, ampliando a capacidade exploratória dos algoritmos genéticos.

- **Recombinação Matricial**. Este método, proposto neste documento, visa explorar de forma mais eficiente o espaço de busca. Para tanto, é realizado o processo inverso na definição do operador de reprodução, onde primeiramente é estipulado um subespaço onde deseja-se que se encontrem os herdeiros, para, numa fase posterior, definir-se um equacionamento matemático que gere esse subespaço.

  Consideremos uma função objetivo que gerou as curvas de nível ilustradas na figura 2.20. O método proposto visa gerar herdeiros no subespaço S descrito na figura.

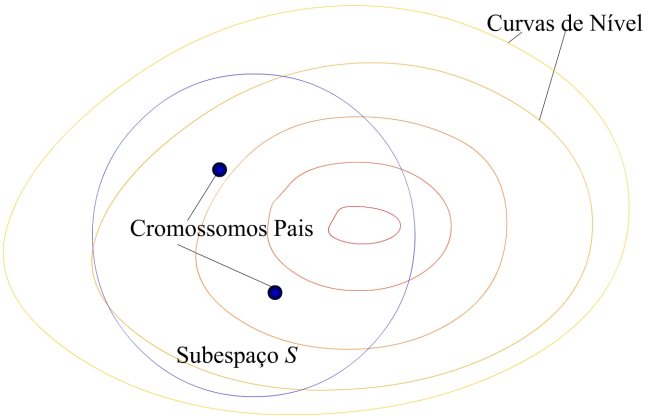

Figura 2.20: Curvas de nível

O subespaço S é então definido como o interior de uma hiperesfera centrada no ponto médio entre os cromossomos pais, dado por: $Ct = (C^1 + C^2)/2$. Esta hiperesfera tem seu raio igual à distância entre os vetores pais: $R = norma(C^2 - C^1)$. Se considerarmos um vetor unitário $v(\theta)$, formando um angulo θ com a reta que passa pelos cromossomos pais r, essa hiperesfera pode ser determinada variando o ângulo θ. A Figura 2.21 traz uma ilustração dessa definição para o espaço R^2.

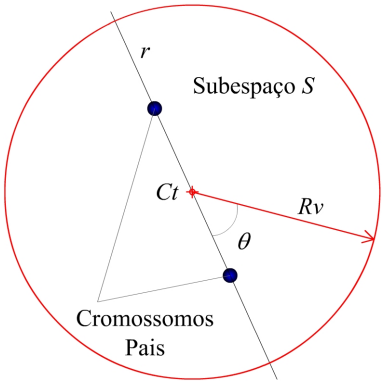

Figura 2.21: Subespaço S

Dessa forma um herdeiro H pertence ao subespaço S se: $norma(H - Ct) \leq Rv$.

Uma maneira de gerar esse espaço, é definir: $Rv = V = C^2 - C^1$ e rotacionar V de um ângulo θ através de uma matriz de rotação. Tal matriz tem por propriedade prover somente a rotação de um vetor quando por ela multiplicado, sem alterar a norma deste, além de apresentar algumas características bastante particulares.

Consideremos a matriz de rotação $M \in R^{N \times N}$, sobre a qual pode-se afirmar:

1. Todos autovalores de M são iguais a 1.
2. Possui inversa igual à transposta $M^T = M^{-1}$
3. Dado $X \in R^N$, $norma(X) = norma(MX)$
4. Dadas C_j, a *j-ésima* coluna da matriz M, e C_i a *i-ésima* coluna dessa matriz, pode-se afirmar: $C_j^T C_i = 0$ para $i \neq j$ e $C_j^T C_i = 1$ para $i = j$. Essa propriedade implica que todas as colunas da matriz M são ortonormais.

Com base nesta matriz, o herdeiro H pode ser gerado da seguinte forma:

$$\begin{aligned} H &= C_t + \alpha MV \\ &= \frac{C^1 + C^2}{2} + \alpha M(C^2 - C^1) \end{aligned} \tag{2.7}$$

onde α é um número aleatório entre 0 e 1.

A matriz M precisa então ser obtida para todo par de cromossomos pais C^1 e C^2, de forma a produzir uma rotação aleatória. Uma maneira de gerar essa matriz é utilizar o processo de ortogonalização de *Gram-Schmidt* [15]. Para tanto, devem ser seguidos os seguintes passos:

1. Definir uma matriz M^0 inicialmente aleatória, onde $m_{i,j}^0$ é um número real aleatório gerado em um intervalo simétrico com relação a 0, gerado a partir de uma distribuição retangular de probabilidades.
2. Definir a primeira coluna da matriz M, C_1^m, como sendo igual a primeira coluna de M^0, $C_1^m = C_1^{m^0}$.
3. Para $i = 2..N$ definir a *i-ésima* coluna da matriz M da seguinte forma:

$$C_i^m = C_i^{m^0} - \sum_{j=1}^{i-1} \frac{C_i^{m^0 T} C_j^m}{||C_j^m||^2} C_j^m \tag{2.8}$$

   onde $\frac{C_i^{m^0 T} C_j^m}{||C_j^m||^2} C_j^m$ é a projeção ortogonal de $C_i^{m^0}$ em C_j^m.
4. Normalizar as colunas de M.

Consideremos a título de exemplo, dois cromossomos pais, C^1 e C^2, escolhidos ao acaso e pertencentes ao espaço R^2, bem como números aleatórios α gerados a partir de uma distribuição retangular de probabilidades. Foram gerados 1500 herdeiros H para os cromossomos C^1 e C^2 com base no método proposto, sendo o resultado ilustrado pela Figura 2.22.

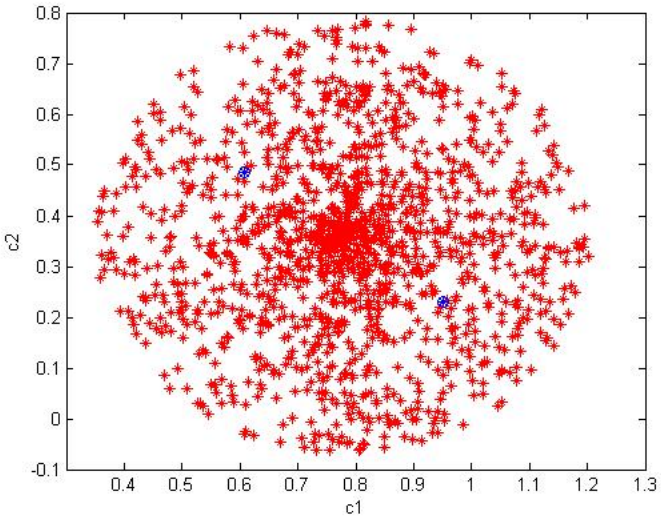

Figura 2.22: Reprodução matricial, exemplo de aplicação. pontos vermelhos representam os herdeiros H, gerados a partir dos cromossomos pais em azul

Este método de reprodução traz resultados semelhantes aos métodos BLX-α e de linha estendida, quanto à expansão da população, porém, gera herdeiros pertencentes a um subespaço S diferente, escolhido na tentativa de aumentar a eficiência do algoritmo, através de uma melhor exploração do espaço de busca.

Computacionalmente, o método proposto mostra-se bastante oneroso, principalmente devido à criação da matriz M. Uma forma de contornar esse problema é considerar uma matriz M aleatória e pseudo-normalizada. Para tanto, consideremos uma matriz M com $m_{i,j}$ pertencente ao intervalo simétrico $I = [-\beta, \beta]$ para todo i e todo j.

A norma de *Frobenius* da matriz M é aqui estimada por: $||M|| \leq N\sqrt{(\beta)}$. Para controlar a recombinação essa norma deve obedecer à inequação: $||M|| \leq 2$. O parâmetro α da equação (2.7) pode ser desprezado devido à característica aleatória de M, resultando na recombinação dada pela equação (2.9):

$$H = \frac{C^1 + C^2}{2} + \frac{2}{N\sqrt{\beta}} M(C^2 - C^1) \tag{2.9}$$

A Figura 2.23 ilustra um exemplo onde foram gerados 500 herdeiros com base na equação (2.9).

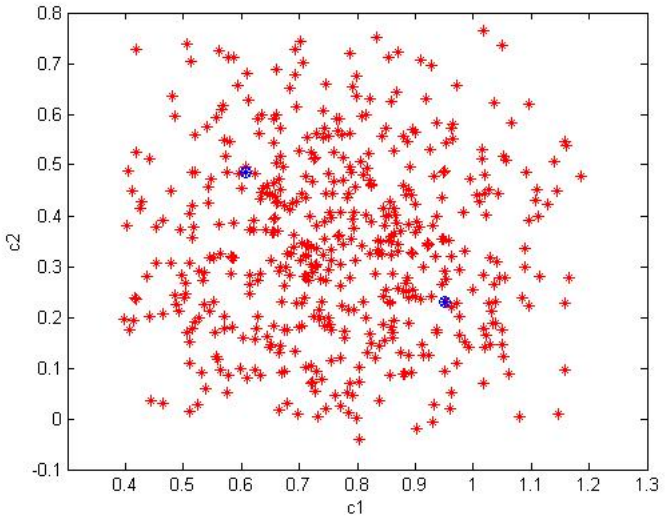

Figura 2.23: Reprodução matricial modificada, exemplo de aplicação. Pontos vermelhos representam os herdeiros H gerados a partir dos cromossomos pais em azul

- **Reprodução Heurística Inspirada em Enxames**. Este método de recombinação, proposto neste documento, introduz uma meta-heurística no processo de geração dos herdeiros, com a intenção de aprimorar a exploração do espaço de busca. Nele, é utilizado um princípio de comportamento social observado no movimento de pássaros e peixes, estudado por Heppner e Grenander [16], e por Reynolds [17].

  Este operador de reprodução tenta introduzir um conceito híbrido, com a clara intenção de polarizar a reprodução dos cromossomos, porém, de forma diferente daquela proposta por Wright em [13], onde o cromossomo filho é *repelido* pelo pai menos adaptado.

  O método proposto incorpora o termo social de um método de otimização por enxame de partículas ao processo de reprodução. No caso, os herdeiros serão atraídos pelo indivíduo da população que possui o maior *fitness* entre todos os outros. A incorporação dessa meta-heurística pode ser adotada por qualquer um dos métodos estudados neste documento. Para o caso da reprodução matricial, a inserção do termo social resulta em:

$$\begin{aligned} H &= \frac{C^1 + C^2}{2} + \frac{2}{N\sqrt{\beta}} M(C^2 - C^1) + \alpha \left(P_g - \frac{C^1 + C^2}{2} \right) \\ &= \frac{C^1 + C^2}{2}(1 - \alpha) + \alpha P_g + \frac{2}{N\sqrt{\beta}} M(C^2 - C^1) \end{aligned} \tag{2.10}$$

onde P_g é o cromossomo com maior fitness da população e $\alpha > 0$ é um número aleatório que controla a intensidade com que o herdeiro será atraído por P_g.

Existe uma sensível diferença na eficiência alcançada pelo algoritmo genético com a utilização dos diversos operadores de reprodução expostos neste documento. A grande diferença entre esses métodos, está no lugar geométrico dos herdeiros e em como o operador de recombinação explora o espaço factível do problema.

A *priori*, é possível imaginar que operadores que explorem uma área maior, como a reprodução matricial e o BLX-α, produzem melhores resultados. Essa conclusão, apesar de precipitada, é observada em problemas de busca irrestritos quando o método é inicializado em um subespaço que não contém a solução desejada. Por outro lado, esses operadores não são os mais indicados para o tratamento de infactibilidades, onde outros métodos como o crossover de linha estendida e o crossover simples podem ser preferíveis em algumas situações.

2.5.5 Mutação

Este operador visa inserir pequenas variações aleatórias nos cromossomos, tal como é observado em sistemas biológicos. O número de indivíduos que irão sofrer mutação, bem como a intensidade da mutação, são fatores determinantes para o bom desempenho do algoritmo genético, e sua definição requer certa experiência e observação.

Codificação Binária

Na codificação binária, a mutação é dada pela modificação de um ou mais bits do cromossomo. É escolhido previamente o ponto de mutação (*locus*), e seu alelo é alterado de 0 para 1 ou de 1 para 0. A Figura 2.24 ilustra o processo de mutação aplicado à cadeias binárias.

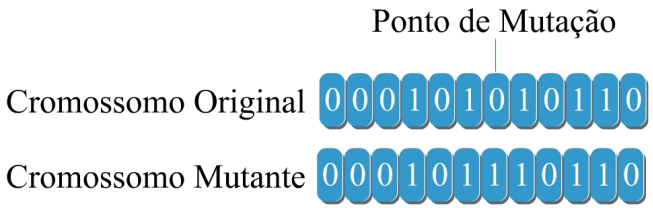

Figura 2.24: Mutação Binária

Codificação Real

Consideremos o cromossomo $C = [c_1, c_2, ..., c_N]$. Este pode sofrer mutação em um ou mais elementos c_i, resultando em genes c_i' mutantes. Existem diversas maneiras de criar essa mutação, sendo algumas delas discutidas a seguir.

- **Mutação Aleatória** [11]. Neste método um ou mais componentes do cromossomo C são substituídos por elementos c_i' gerados aleatoriamente em um intervalo $I = [a_i, b_i]$, com uma distribuição de probabilidades qualquer.

- **Mutação Não-Uniforme**[11]. Este operador de mutação leva em conta o desenvolvimento do algoritmo, tentando, com isso, proporcionar uma mutação de menor intensidade a cada nova geração. Quando aplicado à iteração t do algoritmo programado para executar n_i iterações, este método cria a mutação dada pela equação (2.11):

$$c_i' = \begin{cases} c_i + \Delta(t, b_i - c_i) & se \quad \tau = 0 \\ c_i - \Delta(t, c_i - a_i) & se \quad \tau = 1 \end{cases} \tag{2.11}$$

  onde τ é um número inteiro aleatório entre 0 e 1, a_i e b_i são os limites do intervalo no qual será gerada a componente mutante c_i' , e:

$$\Delta(t, p_m) = p_m(1 - r^{(1-\frac{t}{n_i})^b}) \tag{2.12}$$

  Na equação (2.12), r é um número real aleatório pertencente ao intervalo $[0, 1]$ e b é um parâmetro a ser escolhido, que determina a dependência entre a intensidade da mutação e o desenvolvimento do algoritmo (número de iterações e iteração atual). O número p_m representa a magnitude máxima da mutação e a função $\Delta(t, p_m)$, retorna um número no intervalo $[0, p_m]$, que será mais próximo de zero à medida que o algoritmo progride.

  Este método tenta diminuir a intensidade das mutações gradualmente, preservando uma diversidade grande da população no início do processo evolutivo, porém, diminuindo as alterações aleatórias dos cromossomos após algumas iterações, esperando com isso conseguir um *ajuste fino* da solução quando o algoritmo se aproximar da solução ótima. Este procedimento é também inspirado por processos evolutivos biológicos, nos quais constata-se uma diminuição nas mutação em espécies bem adaptadas.

- **Real Number Creep** [18]. Esta técnica gera cromossomos mutantes ao redor de um cromossomo que encontra-se próximo a uma boa solução.

- **Mutação de Mühlenbein** [14]. Esta operação é definida por:

$$c' - i = c_i \pm p_m \gamma \tag{2.13}$$

onde p_m é a mutação máxima, e γ é um número aleatório gerado pela equação:

$$\gamma = \sum_{k=0}^{15} \alpha_k 2^{-k} \tag{2.14}$$

onde $\alpha_i \in \{0, 1\}$ é um número aleatório com a seguinte distribuição de probabilidade: $p(\alpha = 1) = \frac{1}{16}$ e $p(\alpha = 0) = 1 - p(\alpha = 1)$.

Com isso são gerados genes mutantes c_i' dentro do intervalo $[c_i - p_m, c_i + p_m]$, com alta probabilidade de c_1' ser muito próximo de c_i.

- **Mutação Discreta Modal** [19]. Método similar ao anterior, diferindo apenas no cálculo de γ que é dado por:

$$\gamma = \sum_{k=0}^{\phi} \alpha_k B_m^k \tag{2.15}$$

onde $\phi = (\log(p_m))/(\log(B_m))$ e $B_m > 1$ é um parâmetro a ser determinado chamado de base de mutação.

- **Mutação Vetorial**. Consiste em somar um ruído, representado por um vetor $V \in R^N$, ao cromossomo C. O vetor V é completamente determinado de forma aleatória, sendo cada elemento v_i um número real pertencente ao intervalo simétrico I com distribuição retangular de probabilidade:

$$C' = C + V \tag{2.16}$$

Este método proporciona mutações bastante eficientes no que diz respeito à exploração do espaço de busca, tendo em vista que todas as componentes do cromossomo são alteradas, explorando assim todas as possíveis direções.

Existe ainda a possibilidade de combinar este operador de mutação com a mutação não uniforme, definindo um vetor aleatório inicial V_0 da mesma forma que V e aplicando a seguinte equação:

$$\begin{aligned} V &= V_0\Delta(t) \\ \Delta(t) &= 1 - r^{(1-\frac{t}{n_i})^b} \end{aligned}$$

onde b é um parâmetro a ser escolhido.

Da mesma forma que na mutação não-uniforme, utilizando essa combinação, o método de mutação vetorial também irá gradativamente diminuir a intensidade das variações, visando prover um ajuste fino da solução.

2.6 Parâmetros dos Algoritmos Genéticos

A aplicação computacional dos processos evolutivos, além dos operadores definidos anteriormente, requer a definição de uma serie de parâmetros, que afetam diretamente a eficiência e a robustez do método.

Um algoritmo genético pode ter sido gerado seguindo criteriosamente os métodos aqui descritos e ainda assim falhar em alcançar seu objetivo. Isto pode acontecer principalmente pela escolha equivocada dos parâmetros do algoritmo.

Dentre os principais parâmetros a serem escolhidos ao se aplicar um algoritmo genético, os seguintes podem ser citados como sendo os de maior relevância:

- **Tamanho da População** n_p. É o número de indivíduos presentes na população que será evoluída. A população precisa possuir um tamanho suficiente para cobrir de forma satisfatória o espaço de busca.

  É importante notar que definida uma população de n_P^N indivíduos para um espaço de dimensão N, um aumento de ΔN nessa dimensão requer a utilização de uma população com tamanho $\frac{(N+\Delta N)!}{N!}n_p^N$, a fim de assegurar a mesma *densidade*[5] de indivíduos dentro do espaço de busca. Esse aumento no tamanho da população gera um conseqüente aumento no custo computacional, que pode tornar o método bastante oneroso neste sentido.

- **Taxa de Reprodução** t_{rep}. Representa a quantidade de pares de indivíduos que deixaram herdeiros na próxima geração. Geralmente esse número é escolhido de forma aleatória dentro do intervalo $[0, n_p]$ a cada geração, podendo também ser utilizado um número pré-fixado.

[5]Esse conceito de *densidade* de indivíduos, bem como as equações que regem sua expansão em espaços N-dimensionais, são derivados de um estudo relacionado à hiperplanos e ortogonalidade, que é aqui omitido.

- **Taxa de Mutação** t_{mut}. Número de indivíduos que irão sofrer mutação. Assim como a taxa de reprodução, esse número costuma ser escolhido de forma aleatória, porém, utilizando um intervalo menor, como: $t_{mut} \in [0, 0.3n_p]$, onde o número máximo de indivíduos que podem sofrer mutação, será 30% do total da população.

- **Intensidade da Mutação** p_{mut}. Na codificação real, essa grandeza representa a intensidade da mutação aplicada ao indivíduo. Em mutações vetoriais por exemplo, essa grandeza representa os limites do vetor aleatório $V \in [-p_{mut}, p_{mut}]$[6] que será somado ao cromossomo.

2.7 Exemplos e Comparações

Uma das principais áreas de aplicação dos algoritmos genéticos é a otimização de funções, na qual esses algoritmos representam uma relevante ferramenta, principalmente em situações onde pouco se conhece a respeito da função objetivo, sendo impossível determinar suas derivadas.

A título de exemplo, serão consideradas algumas funções de teste, utilizadas como estudo de casos da aplicação de algoritmos genéticos à otimização de funções, possibilitando comparações refentes ao uso dos vários operados genéticos aqui discutidos, objetivando, com isso, validar os operadores propostos neste documento, bem como identificar aspectos práticos da aplicação de cada operador, tornando possível uma aplicação mais eficiente dos algoritmos genéticos.

Ao todo, serão analisadas quatro funções de teste, descritas a seguir, sendo que cada uma delas apresenta características diferentes com relação a não linearidade, número de ótimos locais entre outros. Essas diferenças tornam esse conjunto de funções bastante representativo do ponto de vista do problema de otimização, expondo várias situações contendo fatores facilitadores e complicadores diferentes, permitindo a inferência a respeito de propriedades gerais dos algoritmos genéticos a partir de seu estudo.

- Esfera:

$$\begin{aligned} f(x) &= \sum_{d=1}^{N} x_d^2 \\ x^* &= 0 \\ f(x^*) &= 0 \end{aligned} \tag{2.17}$$

[6]Esses limites podem ser aplicados a vetores escolhidos a partir de distribuições limitadas, como a uniforme e a retangular, em casos como o da distribuição gaussiana, a intensidade da mutação pode controlar a variância da distribuição.

- Função de Rosenbrock:

$$\begin{aligned} f(x) &= \sum_{d=1}^{N-1}(100(x_{d+1}-x_d^2)^2+(x_d-1)^2) \\ x^* &= [1,1,...,1]^T \\ f(x^*) &= 0 \end{aligned} \tag{2.18}$$

- Função de Rastrigin:

$$\begin{aligned} f(x) &= \sum_{d=1}^{N}(x_d^2-10\cos(2\pi x_d)+10) \\ x^* &= 0 \\ f(x^*) &= 0 \end{aligned} \tag{2.19}$$

- Função de Griewank:

$$\begin{aligned} f(x) &= \frac{1}{4000}\sum_{d=1}^{N}-\prod_{d=1}^{N}\cos\left(\frac{x_d}{\sqrt{d}}\right)+1 \\ x^* &= 0 \\ f(x^*) &= 0 \end{aligned} \tag{2.20}$$

onde x^* (vetorial) é o ótimo global da função e $f(x^*)$ (escalar) é o valor assumido pela função nesse ponto.

A característica estocástica dos algoritmos genéticos dificulta a análise da sua eficiência, dado que será encontrado um resultado diferente a cada execução do método para o mesmo problema. Dessa forma, torna-se necessário introduzir um método de inferência estatística, analisando não apenas resultados pontuais, mais sim a média e a variância obtidos de uma série de n_r repetições do algoritmo.

Com esses dados é possível inferir a respeito da existência de diferença entre as médias populacionais dos resultados obtidos, utilizando configurações diferentes e estimar qual delas apresenta uma maior eficiência quando aplicada a um dado problema.

O algoritmo genético utilizado no processo de otimização das funções propostas emprega uma configuração bastante particular, escolhida de forma a aumentar a eficiência do método. As principais características do algoritmo utilizado estão na seleção de cromossomos para reprodução e para compor a próxima geração.

Os cromossomos para reprodução são escolhidos através do mecanismo da roleta, porém, tomando o cuidado de impedir que um mesmo cromossomo seja escolhido duas vezes para gerar o

mesmo herdeiro. Se isso acontecer, boa parte dos métodos de reprodução estudados irão produzir um herdeiro idêntico ao pai, e não é difícil perceber a redução drástica na população após algumas gerações, causada caso se permita esse tipo de escolha, pois um indivíduo que possua *fitness* muito acima da média irá rapidamente dominar toda a população, podendo causar uma convergência prematura.

Após terem sido gerados os n_{rep} herdeiros da nova geração, o número restante de indivíduos é escolhido utilizando o mecanismo de seleção *elitista*, que, por ser um método determinístico, não irá duplicar elementos na nova geração, preservando a diversidade. Essa opção de seleção combinada à restrição imposta ao mecanismo da roleta, garante ao algoritmo genético uma preservação mínima da diversidade, diminuindo a probabilidade da ocorrência de uma convergência prematura.

A mutação dos indivíduos é feita pelo mecanismo da mutação vetorial, sem a utilização de métodos que diminuem a intensidade da mutação durante o desenvolvimento do método. Esse processo de redução, apesar de observado em sistemas naturais, pode diminuir prematuramente a diversidade da população, levando em alguns casos à convergência prematura.

Ao todo, serão estudas 6 variações do algoritmo genético, diferindo apenas no mecanismo de reprodução utilizado. As variações consideradas são classificadas como segue.

- Linha Estendida. Algoritmo genético utilizando reprodução de linha estendida.
- Matricial Modificada. Utiliza a reprodução matricial modificada.
- Matricial. Utiliza reprodução matricial.
- M.M. Heurística. Utiliza reprodução matricial modificada combinada com meta-heurística inspirada em enxame de partículas.
- M. Heurística. Utiliza reprodução matricial combinada com meta-heurística baseada em enxame de partículas.
- BLX 0.25. Utiliza reprodução BLXα, com $alpha = 0.25$.

O objetivo principal desse estudo é identificar as potencialidades de cada um dos métodos de recombinação em questão, além de analisar a relação entre essas potencialidades, as características dos problemas e o desempenho alcançado pelo algoritmo. Dessa forma, tem-se interesse principalmente na evolução do algoritmo, sendo irrelevante o critério de parada, assim, é adotado de forma empírica um número máximo de gerações suficientes para realização da análise proposta.

2.7.1 Função de Rosenbrock

Em um primeiro exemplo, consideremos a função de Rosenbrock, utilizando em sua otimização o seguinte conjunto de parâmetros, empiricamente escolhidos:

- $N = 10$
- $n_r = 100$
- $n_g = 20$
- $n_p = 100$

Como o ótimo global dessa função é $x^* = [1, 1, ..., 1]^T$, foi escolhido convenientemente um intervalo inicial $J = [-0.5I, 1.5I]$ contendo esse ponto como condição inicial para a população, que é inicializada aleatoriamente[7] distribuída nesse intervalo. O gráfico mostrado na Figura 2.25 ilustra os valores médios da função, obtidos por cada uma das 6 variações propostas.

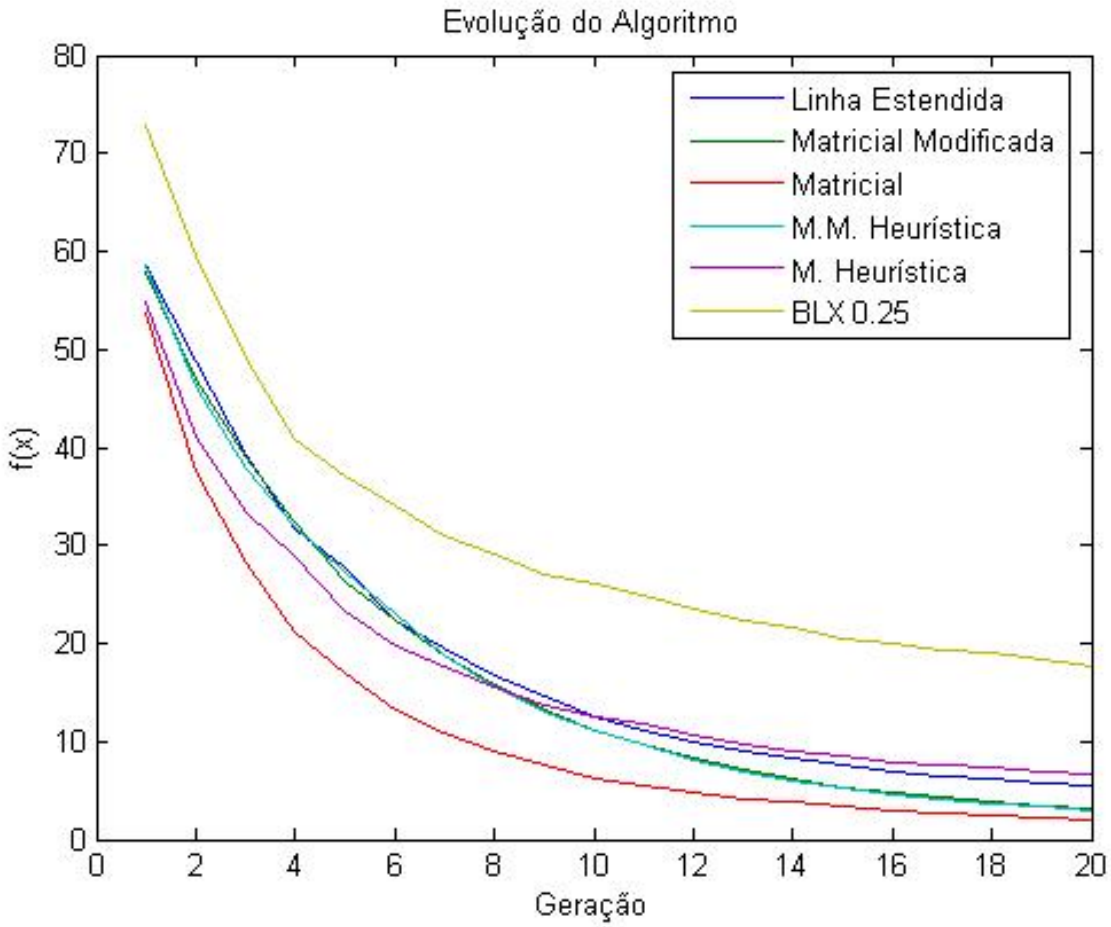

Figura 2.25: Valor médio da função Rosenbrock obtido por cada uma das variações propostas após 100 repetições

[7] A inicialização da população é tomada de forma aleatória com base em uma distribuição uniforme. Essa mesma distribuição será utilizada neste contexto no restante deste documento, exceto onde seja dito contrário.

É possível notar o desempenho superior do método de reprodução matricial, proposto neste documento. A combinação desse operador de reprodução com a meta-heurística, por sua vez, não proporcionou o resultado esperado, sendo esse inferior ao da aplicação da recombinação matricial de forma isolada. O método de reprodução matricial modificado, bem como a combinação desse método com a meta-heurística em questão, apesar de apresentar resultados inferiores aos do método matricial, foi superior às outras propostas de recombinação utilizadas.

Outra característica fundamental na análise do desenvolvimento das variações propostas é a avaliação da população como um todo e não apenas do melhor indivíduo, oque é feito por meio da Figura 2.25. Essa avaliação é realizada através da diversidade da população, estimada com base na variância[8] σ^2 observada nos valores da função calculados para todos os indivíduos da população.

O gráfico mostrado na Figura 2.26, mostra o valor médio obtido para a diversidade da população em cada geração, plotado em uma escala logarítmica.

[8] A diversidade da população em um algoritmo genético, ou outro algoritmo similar, representa as diferenças entre os indivíduos, podendo, no caso em questão, ser estimada por uma métrica da distancia entre os cromossomos, sendo a grandeza aqui escolhida uma entre inúmeras possibilidades

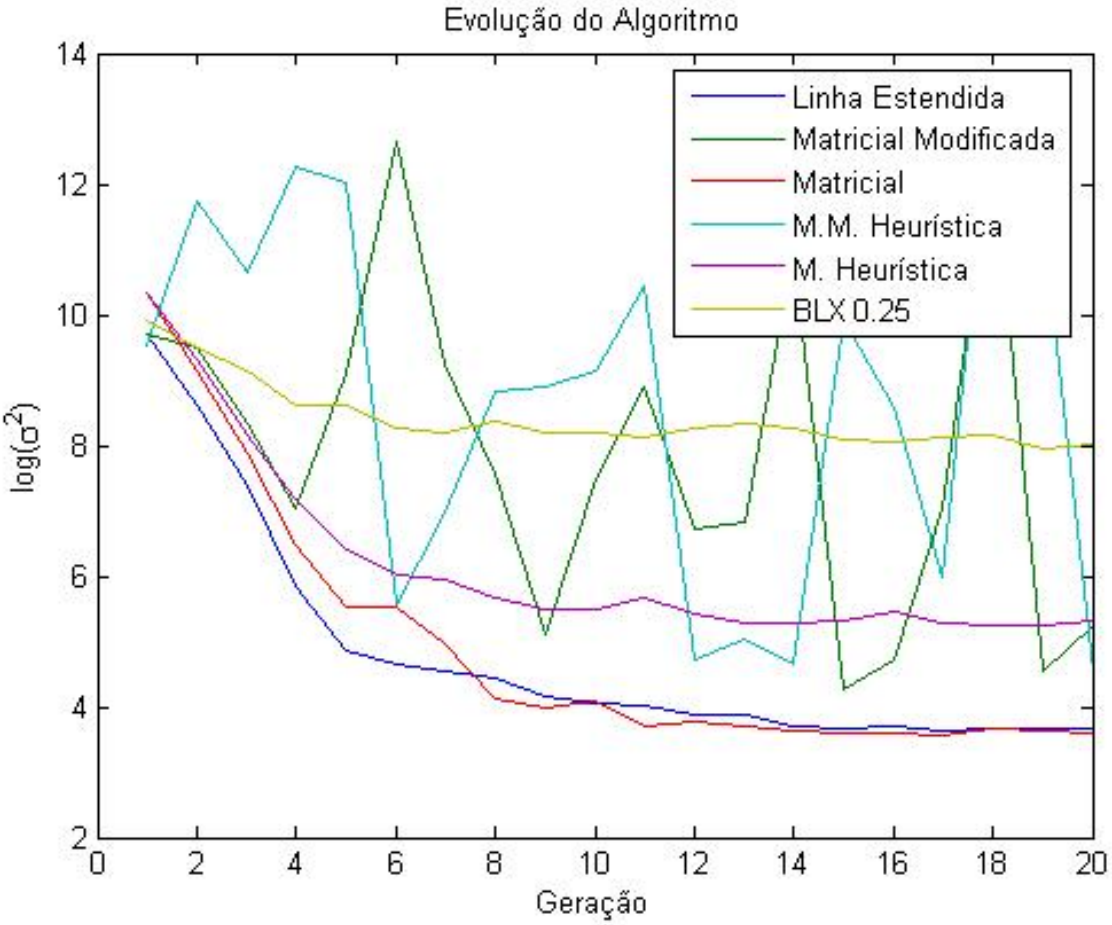

Figura 2.26: Diversidade média da população, estimada pela variância do valor da função Rosenbrock para cada cromossomo da população

É fácil perceber que o operador de reprodução matricial, assim como a reprodução de linha estendida, apresentam uma redução bastante acentuada na diversidade da população durante a evolução do algoritmo. Essa redução pode causar uma convergência prematura do método em algumas situações, o que não aconteceu no exemplo proposto. No caso do operador de reprodução matricial, essa diminuição da diversidade aumentou a taxa de convergência do algoritmo genético, tendo em vista que esse é o método que obteve o melhor resultado dentre as variações analisadas.

Geralmente, quando se tenta aumentar a taxa de convergência de um algoritmo genético, seja reduzindo a taxa de mutação ou pela escolha apropriada dos operadores, é observada a redução também da diversidade da população, que em alguns casos pode se tornar um problema, devendo assim, ser respeitado um compromisso claro, existente entre a diversidade da população e a taxa de convergência.

A diversidade da população para o método de reprodução matricial modificada e sua combinação com a meta-heurística, por outro lado, se manteve consideravelmente elevada durante o desenvolvimento do algoritmo.

Essa preservação da diversidade também é constatada no método BLX 0.25, porém, este não encontra bons resultados por apresentar uma taxa de convergência muito pequena. Os métodos de reprodução matricial modificada, ao contrário do método BLX 0.25, apesar de preservarem a diversidade da população, apresentam uma taxa de convergência relativamente alta, sendo inferior apenas àquela observada no método de recombinação matricial.

Esses resultados levam à conclusão de que apesar do método de reprodução matricial apresentar uma taxa de convergência bastante elevada, os métodos de reprodução matricial modificada, são superiores em preservação da diversidade da população e também apresentam uma boa taxa de convergência, representando assim um melhor compromisso entre esses dois fatores para o problema em questão.

Consideremos agora, a título de complementariedade, uma nova abordagem, descrita pela configuração a seguir, utilizando dessa vez um número maior de gerações.

- $N = 10$
- $n_r = 100$
- $n_g = 60$
- $n_p = 100$

A Figura 2.27 ilustra os resultados médios obtidos com a aplicação da configuração em questão, onde percebe-se que apesar de manter a diversidade da população o método BLX 0.25, não obtém um resultado satisfatório, sendo que os métodos propostos neste documento, baseados em reproduções matriciais apresentam resultados superiores quando comparados aos outros operadores de reprodução analisados.

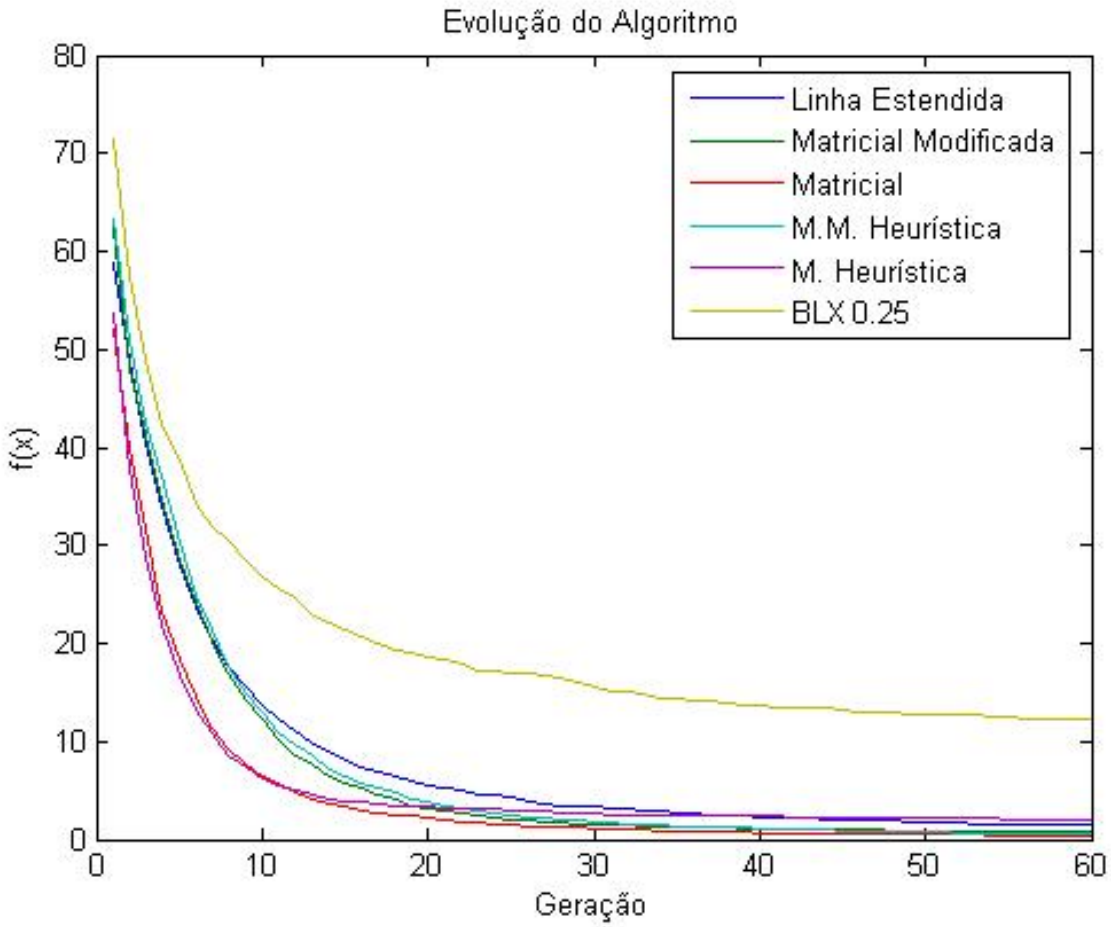

Figura 2.27: Valores médios da função de Rosenbrock para 60 gerações

Outro teste pertinente com respeito ao problema em questão, é averiguar o comportamento do algoritmo quando a população inicial é inicializada em uma região que não contém o ótimo global da função.

Nesta situação o algoritmo deve ser capaz de expandir sua população, além do espaço de busca original, a fim de alcançar a solução desejada onde ela esteja. Essa capacidade de expansão, geralmente observada em algoritmos genéticos, é devida em parte ao processo de mutação, que introduz variações aleatórias nos cromossomos, podendo com isso gerar novos indivíduos fora do espaço de busca original e principalmente devido ao mecanismo de reprodução, quando este possui essa capacidade de expansão.

Dentre os vários mecanismos de reprodução aqui estudados, boa parte deles foi concebida de forma a prover essa expansão, como é caso dos operadores de recombinação de linha estendia, BLX 0.25 e dos operadores matriciais propostos neste documento. Dessa forma, as 6 variações em análise,

possuem capacidade de expansão do processo de reprodução, logo, apresentam os requisitos para gerar resultados satisfatórios nesse cenário de aplicação.

Para averiguar de forma prática essa característica do algoritmo genético utilizado, consideremos uma população inicial contida no intervalo: $J = [-0.25I, 0.25I]$, que não contém o ótimo global da função de Rosenbrock. O comportamento médio das 6 variações do algoritmo testadas é mostrado na Figura 2.28.

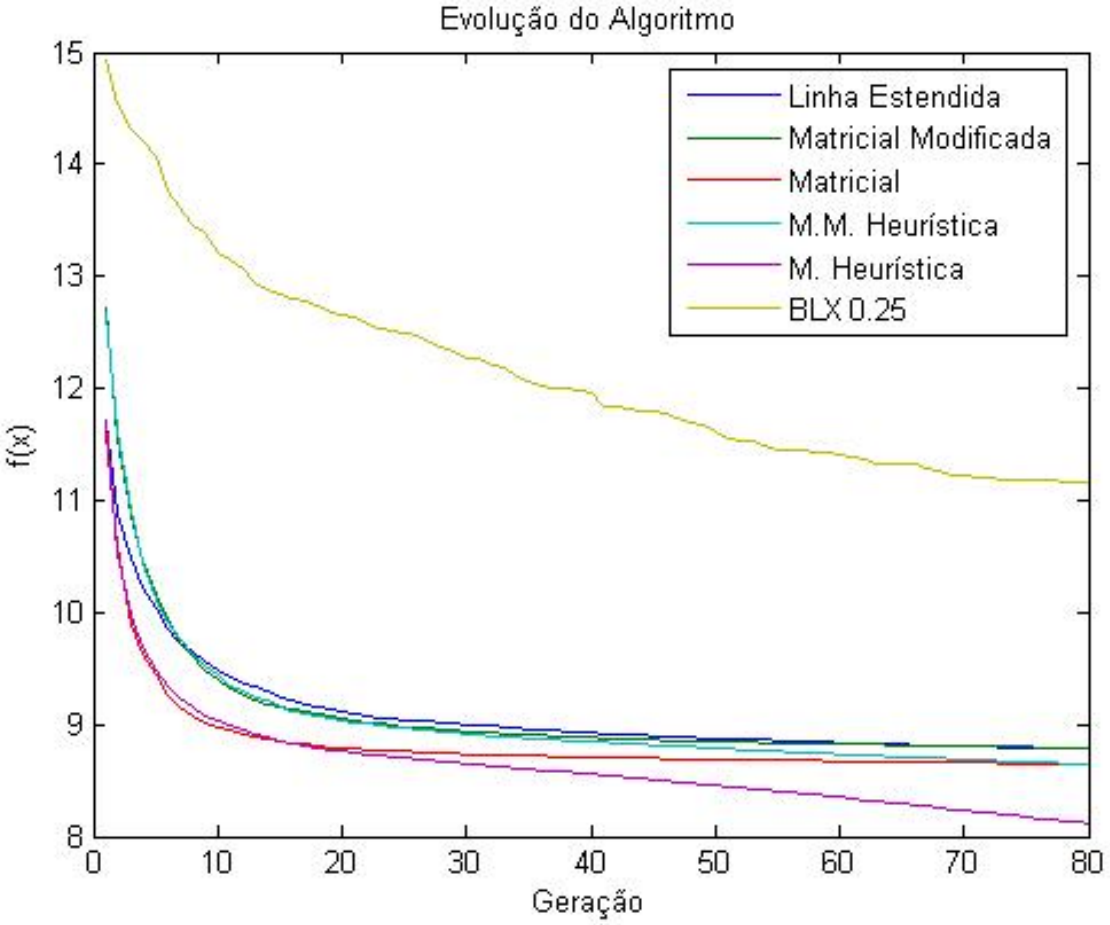

Figura 2.28: Valores médios da função de Rosenbrock para 80 gerações, com método inicializado longe do objetivo

A evolução do algoritmo genético para este cenário de aplicação mostrou-se bastante superior quando foram utilizados os operadores de recombinação matriciais propostos neste documento, principalmente quando utilizada a reprodução matricial combinada com a meta-heurística, onde observou-se uma solução consideravelmente superior a de outros métodos.

Esse resultado indica que a combinação da reprodução matricial com o termo social de um enxame de partículas proporcionou uma maior capacidade de extrapolação do espaço de busca original. Essa característica de expansão observada é intuitivamente bastante razoável, dado que essa combinação irá gerar herdeiros mais próximos ao melhor indivíduo da população, que, no caso em questão, certamente encontra-se na fronteira do subespaço onde a população está inserida, tendo assim, uma probabilidade muito maior de gerar herdeiros fora desse subespaço.

No caso exposto anteriormente, onde o ótimo global da função objetivo encontra-se inserido no espaço inicial de busca, a diversidade da população é muito maior para as variações que utilizam recombinação BLX 0.25, Matricial Modificada e M.M. Heurística. Essa característica de preservação da diversidade permanece a mesma averiguada anteriormente, como pode ser observado no gráfico da Figura 2.29, onde nota-se claramente a alta diversidade apresentada pelo algoritmo genético quando se utiliza esses métodos de reprodução (BLX 0.25, M. Modificada, M.M. Heurística).

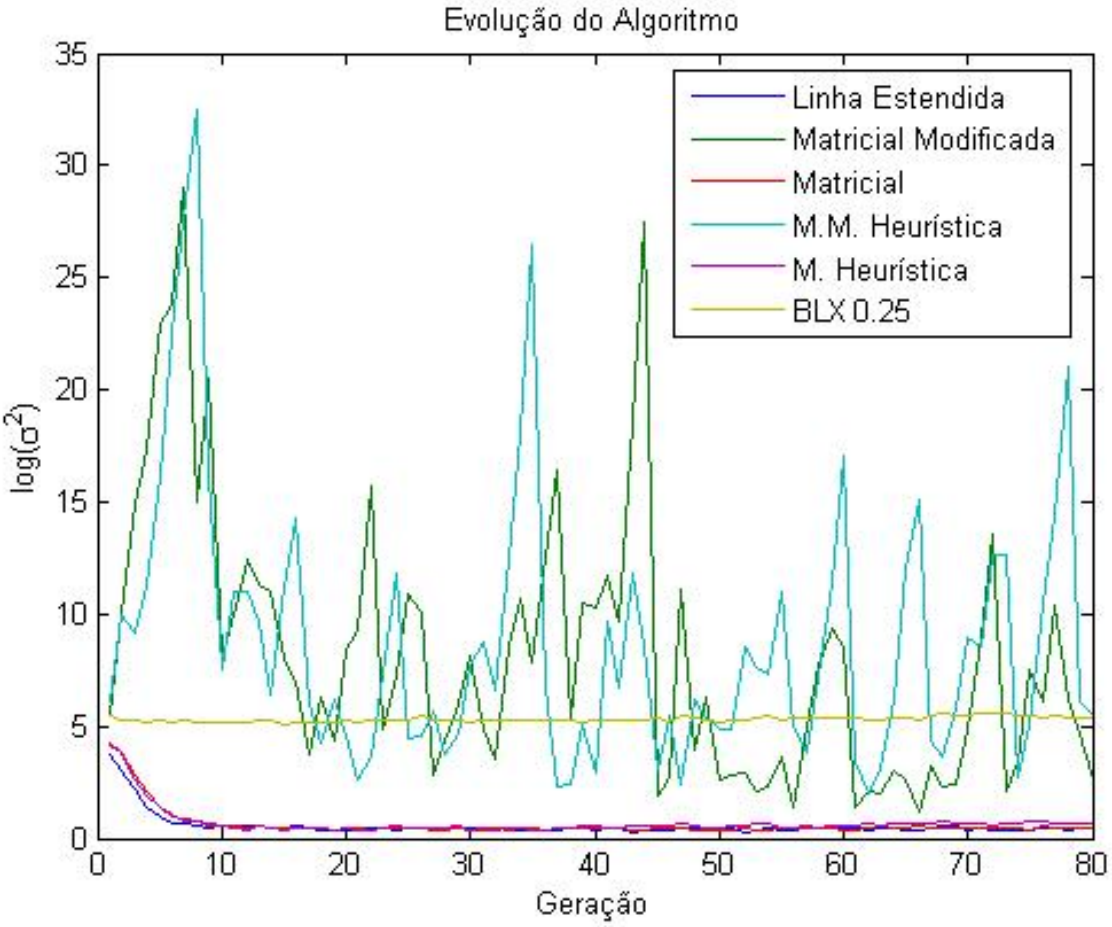

Figura 2.29: Diversidade média da população, função de Rosenbrock

Novamente os métodos de recombinação matricial propostos neste documento, quando analisadas a taxa de convergência e a preservação da diversidade, apresentam um maior compromisso entre esses dois fatores, o que pode torná-los mais robustos e evidentemente menos susceptíveis a convergências prematuras.

2.7.2 Função Esfera

Dando continuidade à análise das variações propostas para o algoritmo genético estudado, consideremos agora a função esfera, que possui um ótimo global $x^* = 0$ e não possui ótimos locais. Outra característica importante dessa função é a sua não linearidade, que é relativamente bastante simples e não representa um grande desafio do ponto de vista da otimização.

Em uma primeira análise, consideremos a configuração descrita a seguir:

- $N = 10$
- $n_r = 100$
- $n_g = 20$
- $n_p = 100$

O gráfico ilustrado na Figura 2.30 mostra o resultado médio obtido a cada geração do algoritmo genético para as 6 variações em análise, quando a população inicial é gerada dentro do intervalo $J = [-I, I]$, que contém o ótimo global da função esfera.

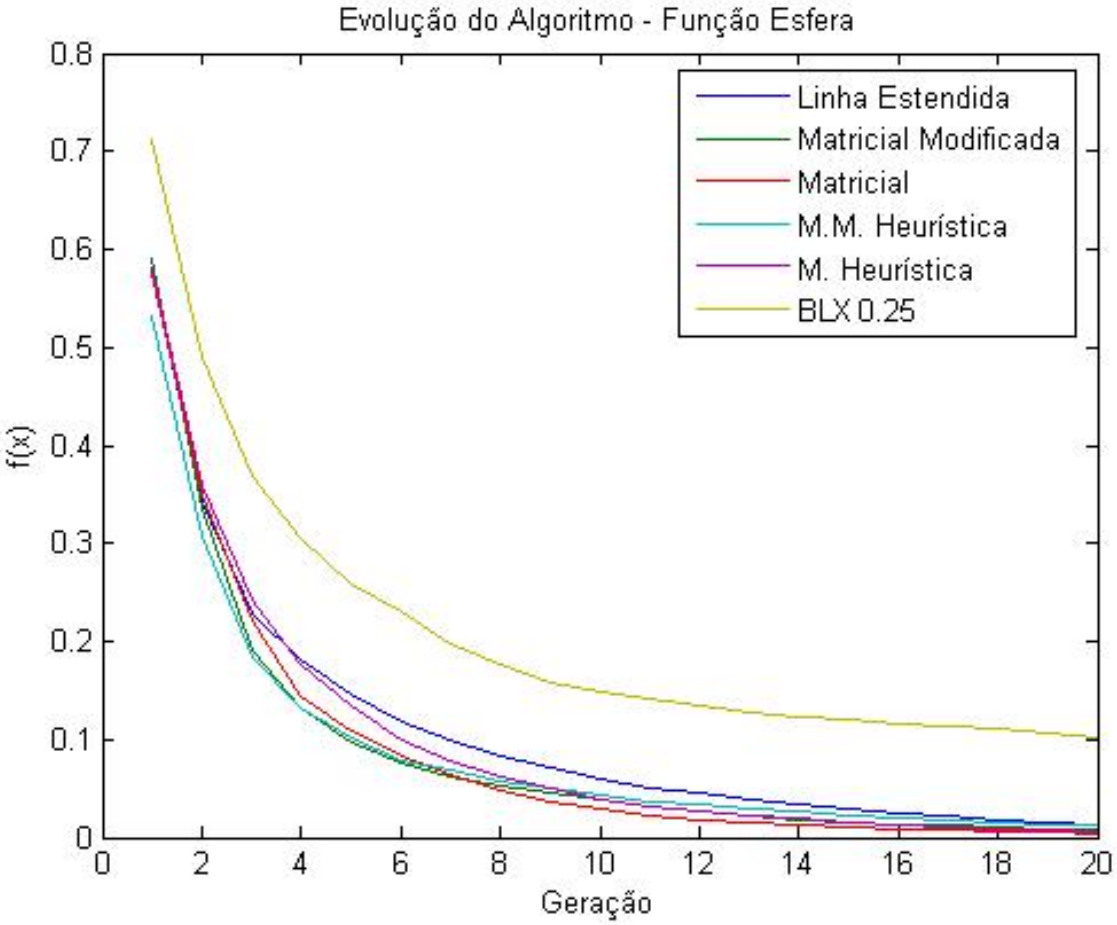

Figura 2.30: Valores médios da função esfera para 20 gerações

Para esse problema, assim como para o problema estudado anteriormente, os operadores de recombinação propostos neste documento mostram-se mais eficientes que os demais analisados neste estudo, apresentando uma maior taxa de convergência.

A diversidade da população para este problema, por outro lado, apresenta um comportamento bastante diferente daquele observado quando se otimiza a função de Rosenbrock, situação na qual são obtidos valores relativamente elevados para a diversidade da população, utilizando os operadores de reprodução matriciais modificados e BLX 0.25.

Para o problema em questão, a diversidade da população do método BLX 0.25 permanece bastante elevada, enquanto, para os métodos de recombinação matricial modificada, essa diversidade diminui consideravelmente ao longo das gerações. O método de reprodução matricial e sua combinação com a meta-heurística, por outro lado, preservam uma diversidade superior à dos outros métodos, sendo inferior apenas aquela observada quando se utiliza o método BLX 0.25.

O gráfico mostrado na Figura 2.31, ilustra o comportamento da diversidade da população em cada geração do processo de otimização.

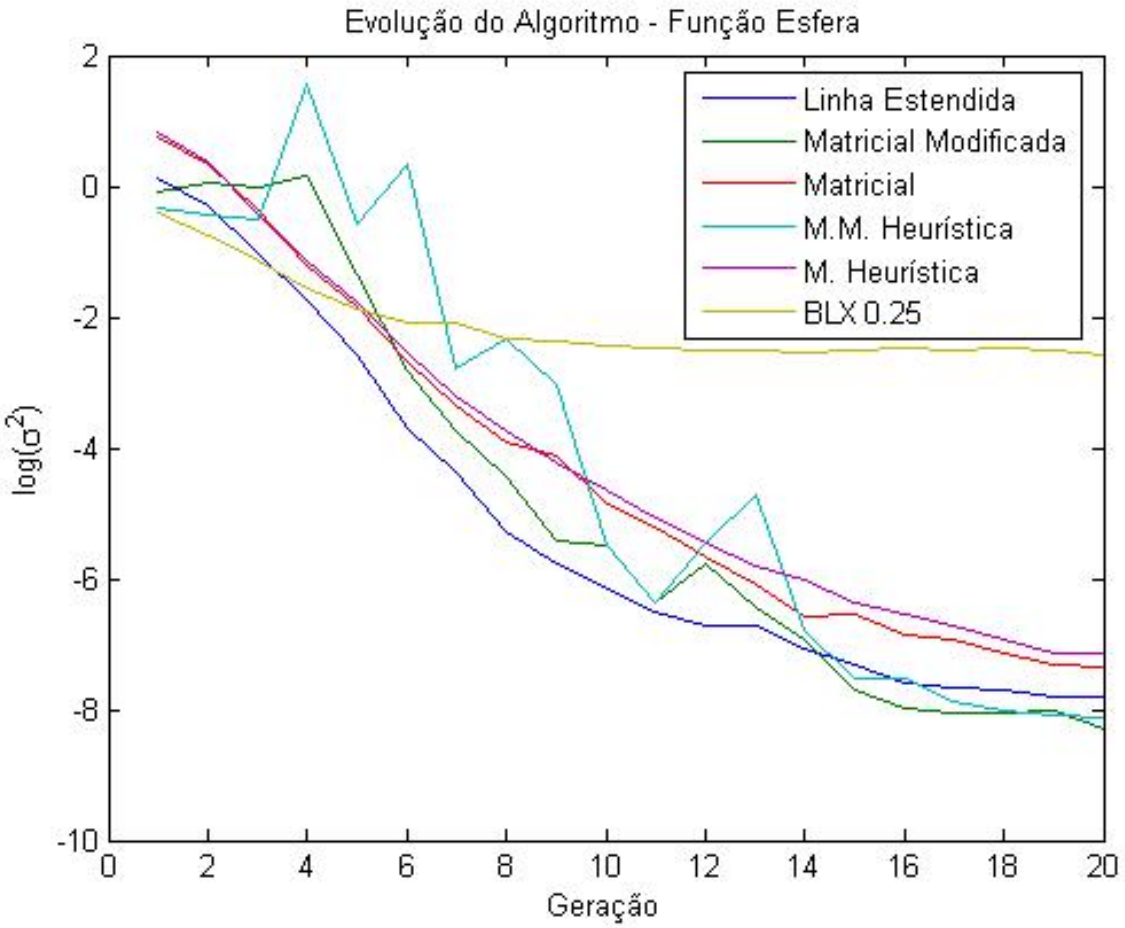

Figura 2.31: Diversidade média da população, função esfera

Novamente, consideremos uma situação onde os algoritmos em questão são inicializados em um subespaço que não contenha o ótimo global da função. Para tanto, será evoluída a população inicialmente encontrada no intervalo $J = [1.5I, 2.5I]$, que não contém o ponto $x = 0$.

O resultado do processo evolutivo para essa inicialização pode ser verificado no gráfico mostrado na Figura 2.32, que ilustra o valor médio da função obtido pelas 6 variações testadas, utilizando a mesma configuração anterior, porém, agora são realizadas 80 gerações do processo evolutivo.

É possível verificar nesse gráfico que o método BLX 0.25 obteve um ótimo resultado, sendo superior às variações do algoritmo que utilizam recombinação matricial e de linha estendida. Esse resultado ainda não havia sido verificado em nenhum dos casos estudados, nos quais geralmente, o operador de recombinação BLX 0.25 apresenta resultados inferiores aos demais testados.

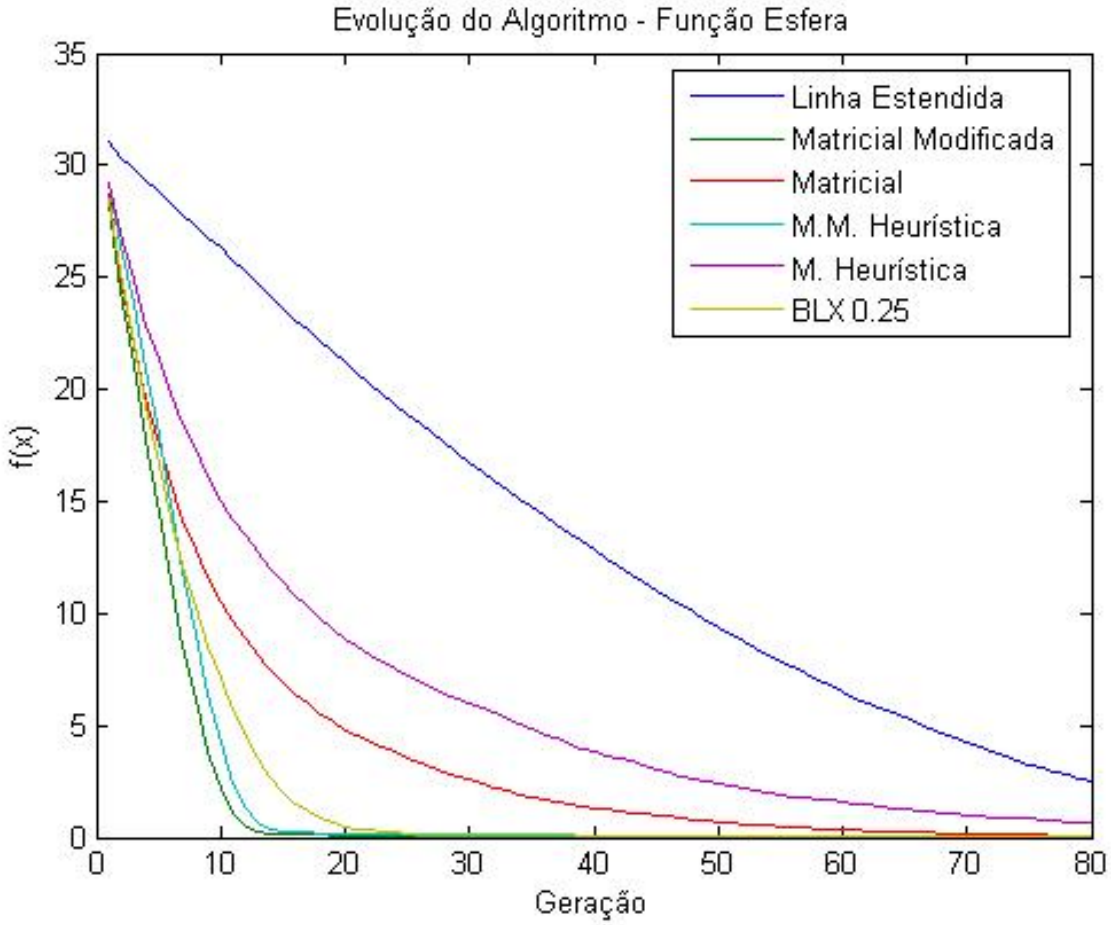

Figura 2.32: Valores médios da função esfera para 80 gerações

Os operadores de recombinação matricial modificada (Matricial Modificada e M.M. Heurística), apresentam os melhores resultados nesse cenário de aplicação, sendo superiores à todos os outros observados. Os outros operadores de reprodução propostos neste documento (Matricial e M. Heurística), também apresentam resultados consistentes, sendo a recombinação de linha estendida, o método que apresenta os piores resultados quando comparado aos demais.

A diversidade das populações para esse cenário de aplicação pode ser averiguada no gráfico ilustrado na Figura 2.33. Novamente nota-se a grande oscilação na diversidade da população quando são utilizados os operadores de recombinação: Matricial Modificado e M.M. Heurística. Essa variação na diversidade auxilia a compreender o bom desempenho apresentado por essas variações do algoritmo, dado que a solução do problema encontra-se distante da população inicial. Essa alta preservação da diversidade favorece a expansão da população.

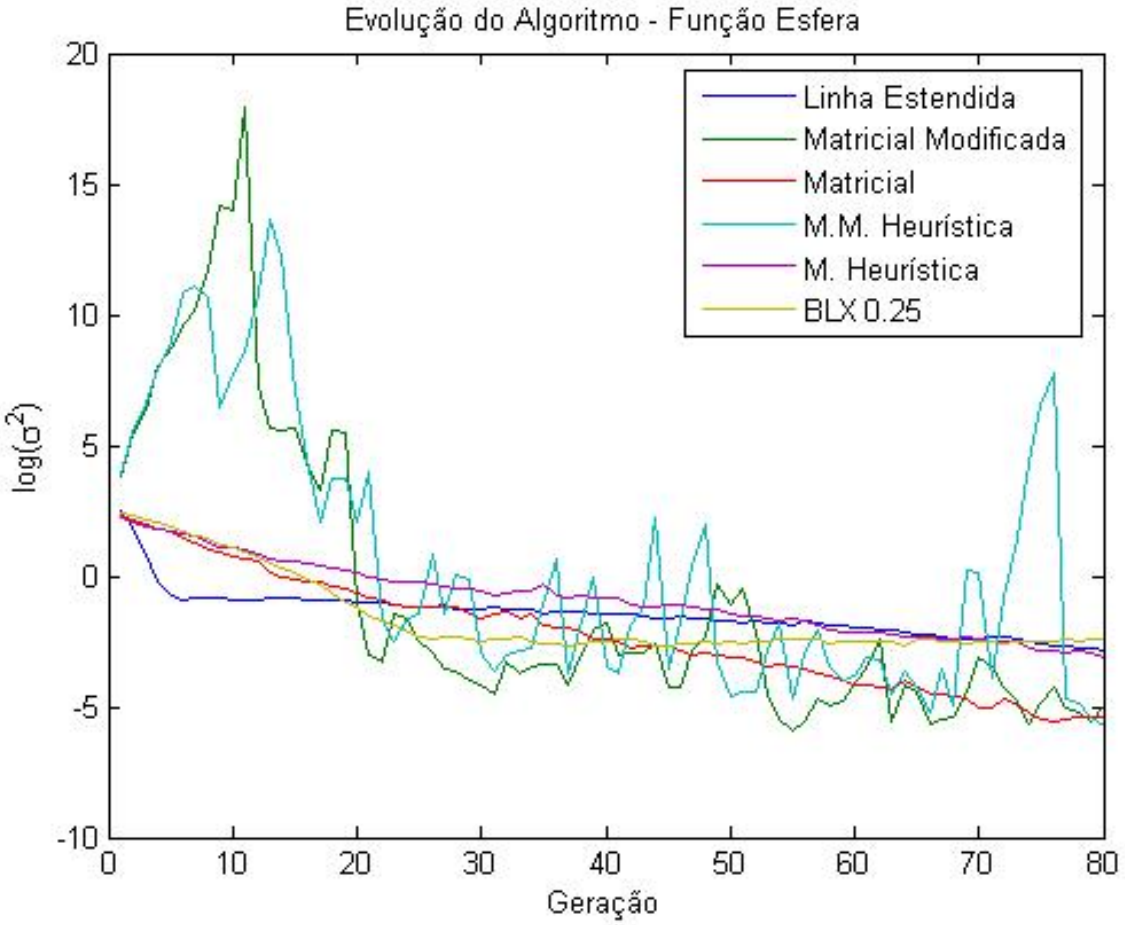

Figura 2.33: Diversidade média da população para 80 gerações, função esfera

Os métodos de reprodução propostos neste documento obtiveram bons resultados também para o problema de otimização da função esfera, assim como para o problema de otimização da função de Rosenbrock, mostrando-se eficientes dentro dos cenários de aplicação considerados.

2.7.3 Função de Rastrigin

Um próxima etapa da análise consiste em averiguar o comportamento das variações propostas na otimização da função de Rastrigin. Essa função apresenta uma série de mínimos locais devidos a funções cosseno, sendo considerável a probabilidade de convergência para um desses mínimos locais.

Consideremos a seguinte configuração para o processo de otimização da função de Rastrigin, onde a população inicial é tomada no intervalo $J = [-2I, 2I]$, que contém o ótimo global dessa função, bem como vários outros ótimos locais:

- $N = 4$

- $n_r = 100$
- $n_g = 80$
- $n_p = 40$

A evolução média do processo de otimização pode ser verificada no gráfico descrito na Figura 2.34.

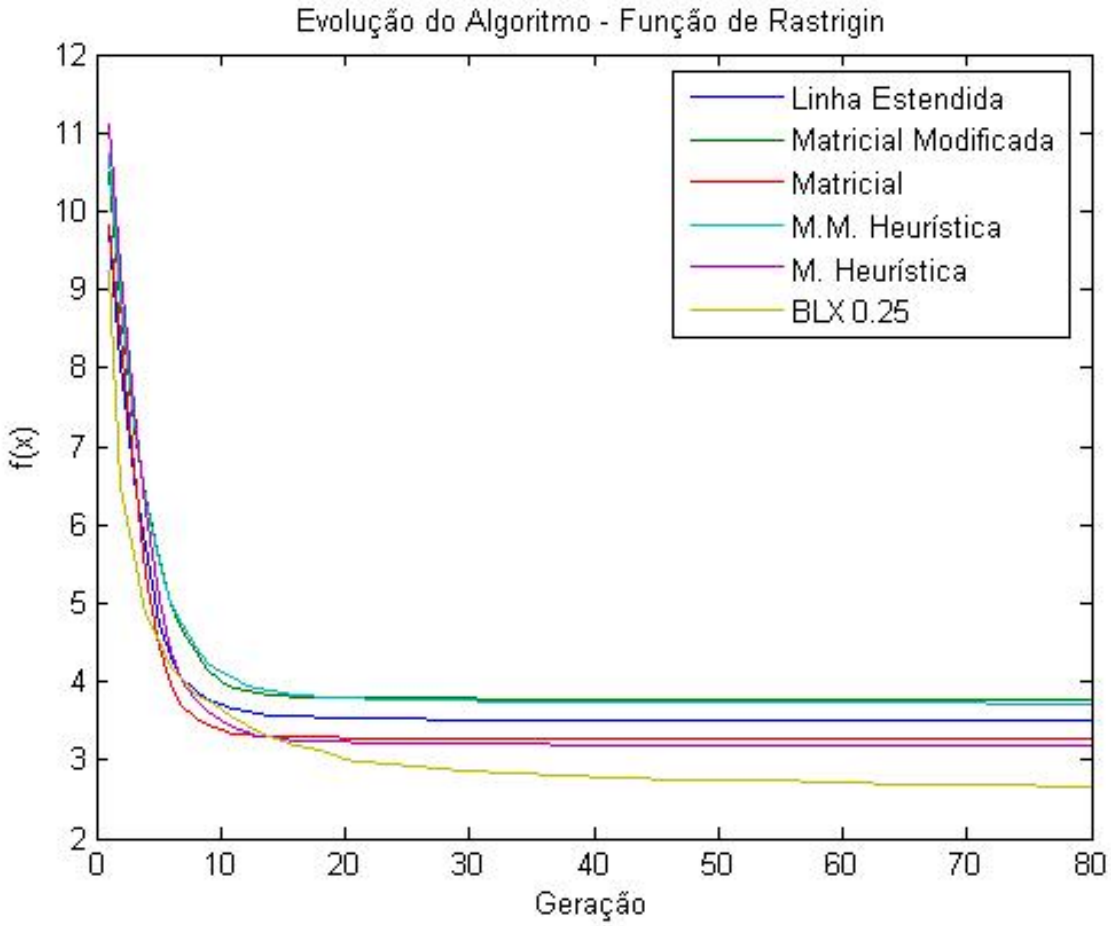

Figura 2.34: Valores médios da função de Rastrigin para 80 gerações

Nesta aplicação, o operador de recombinação BLX 0.25 apresentou resultados consideravelmente superiores aos de outros operadores de recombinação. Esse fato pode ser explicado pela alta e constante diversidade da população preservada por esse método, que torna-se um grande diferencial neste caso, onde existem vários mínimos locais dentro do espaço de busca.

Mantendo uma diversidade alta, o algoritmo genético pode ser capaz de encontrar a solução ótima mais facilmente sem se prender, prematuramente, à soluções sub-ótimas. Esse comportamento da

população pode ser observado quando analisadas simultaneamente a evolução das soluções e da diversidade da população.

A diversidade da população para este estudo de caso pode ser observada no gráfico ilustrado na Figura 2.35, que mostra a diversidade média da população para cada geração, plotada em uma escala logarítmica.

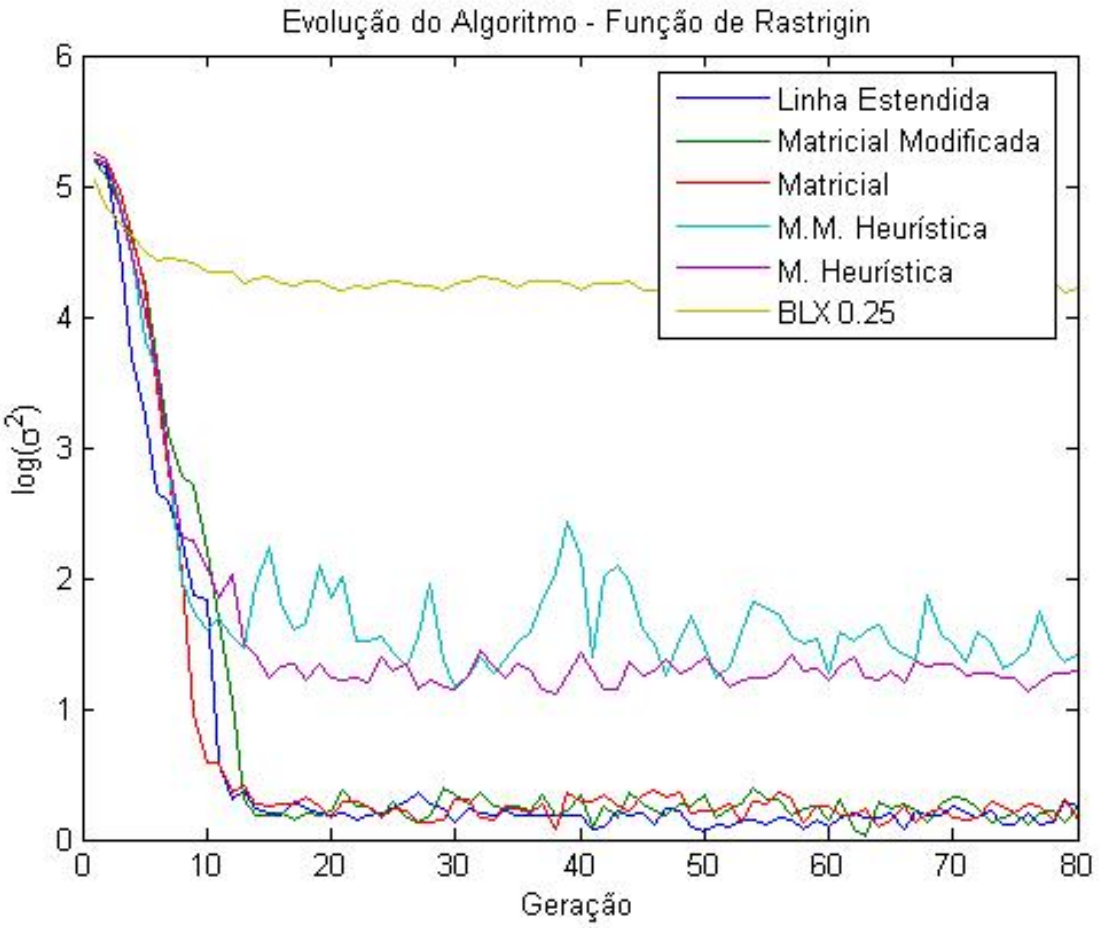

Figura 2.35: Diversidade média da população, função de Rastrigin

Tendo em vista que a função de Rastrigin apresenta vários mínimos locais dentro do subespaço onde a população foi inicializada, é conveniente examinar quais as soluções encontras pelas variações do algoritmo genético e em com qual freqüência essas soluções ocorrem.

Para o resultado exposto anteriormente, é observada a ocorrência de mínimos locais que representam 13 valores distintos da função $f(x^*)$, sendo eles: $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13]$. A freqüência com a qual, cada variante do algoritmo genético encontra cada um desses ótimos locais, pode ser observada na Tabela 2.5.

Valor da Função $f(x^*)$	Linha Estendida	Matricial Modificada	Matricial	M.M. Heurística	M. Heurística	BLX 0.25
0	2%	2%	5%	2%	6%	6%
1	11%	16%	13%	12%	12%	26%
2	24%	25%	27%	30%	21%	28%
3	23%	21%	23%	21%	16%	24%
4	13%	7%	12%	3%	7%	7%
5	7%	11%	6%	8%	11%	5%
6	13%	9%	8%	13%	15%	3%
7	3%	7%	3%	7%	8%	1%
8	0%	0%	1%	0%	1%	0%
9	1%	2%	2%	2%	0%	0%
10	2%	0%	0%	1%	2%	0%
11	0%	0%	0%	0%	1%	0%
13	1%	0%	0%	1%	0%	0%

Tabela 2.5: Resultados das repetições - Freqüência dos mínimos locais

É possível observar que apesar do método de recombinação BLX 0.25 alcançar um valor médio para a função, menor do que os outros métodos analisados, a variação do algoritmo que utiliza a recombinação matricial encontra o mínimo global da função com a mesma freqüência que este método, apresentando um erro médio superior, por exibir uma freqüência maior na localização de pontos que correspondem a valores da função entre 4 e 7.

O problema em questão mostra-se um grande desafio para os algoritmos genéticos quando se tem interesse em encontrar apenas o mínimo global. É possível notar que a freqüência com que o algoritmo converge para esta solução é pequena, sendo o resultado mais freqüente encontrar um dos ótimos locais referentes a $f(x^*) = 2$.

Nesta aplicação, o método BLX 0.25 demonstrou uma capacidade superior a todos os outros métodos, não só quando se analisa o resultado médio obtido mostrado na figura 2.34, mas também na análise da tabela 2.5, onde nota-se que esta variação do algoritmo genético encontra com muito pouca freqüência soluções sub-ótimas, que representam um alto valor para função objetivo.

Os métodos propostos neste documento, por sua vez, apresentam resultados intermediários entre aqueles obtidos pelos operadores de recombinação BLX 0.25 e linha estendida, que apresentou um dos piores resultados. Conclui-se também, que quando se utiliza a recombinação matricial e a recombinação matricial combinada com a meta heurística, são obtidos os melhores resultados entre os métodos propostos.

		Linha Estendida		Matricial Modificada		Matricial		M.M. Heurística		M. Heurística		BLX 0.25	
N	n_p	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2
2	20	0.2898	0.5130	0.4219	0.9582	0.2173	0.3353	0.3703	1.1618	0.1522	0.3667	0.0027	0.0000
4	40	2.0785	2.0416	1.8919	2.2875	1.7768	1.8750	2.0197	2.5677	1.5870	2.6812	1.1305	1.5445
8	80	1.3685	2.5183	1.2943	6.2239	1.1858	3.3628	1.2613	4.2862	1.3736	2.8273	5.3701	3.3025
16	160	6.333	9.391	2.874	4.822	2.285	4.976	5.443	14.03	8.648	10.31	37.98	81.58
32	320	12.297	4.255	3.0728	6.8912	2.433	1.865	24.78	70.03	30.24	284.1	281.7	879.6
64	256	37.54	67.06	31.31	75.38	21.24	34.42	118	355.1	81.32	2232	1114	2e+004

Tabela 2.6: Otimização da função de Rosenbrock - Valor final da função

		Linha Estendida		Matricial Modificada		Matricial		M.M. Heurística		M. Heurística		BLX 0.25	
N	n_p	div.	σ^2	div.	σ^2	div.	σ^2	div.	σ^2	div.	σ^2	div.	σ^2
2	20	1.017	10.95	1.695	59.479	1.160	6.690	4.642	171.8	8.184	389	325	1e+005
4	40	1.722	8.658	2.319	14.83	2.571	27.97	5.559	205	7.261	270	610	1e+006
8	80	18.8	235	18.54	272.9	18.04	237.3	1e+007	1e+016	39.63	2276	1703	5e+006
16	160	92.06	2924	93.94	2352	96.21	5883	270.2	7e+005	174	8464	6243	1e+007
32	320	510.3	7e+004	4e+005	2e+013	622.7	2e+005	1e+004	1e+010	1e+004	3e+008	1e+005	6e+009
64	256	1901	8e+005	1557	6e+005	1984	1+006	7e+007	5e+017	6e+004	5e+009	4e+005	1e+011

Tabela 2.7: Otimização da função de Rosenbrock - Diversidade final da população

2.7.4 Resumo das Análises

Visando agora estudar o comportamento do algoritmo genético e suas variações para uma gama maior de possíveis cenários de aplicação, são apresentados, nas tabelas que seguem, os resultados de várias aplicações do algoritmo a diferentes problemas, contendo diferentes números de variáveis e funções objetivo. Novamente é utilizada a inferência estatística para analisar o comportamento médio das variações propostas para o algoritmo genético.

São tomados como resultado do processo iterativo, o valor obtido para a função de custo após concluída a execução do algoritmo, também referido como valor final da função, e a diversidade final da população, avaliada para cada indivíduo da população, após a ultima geração do processo de otimização. Em cada uma das situações expostas nas tabelas, a variante do algoritmo genético a ser analisada é executada por 80 gerações, repetidas 100 vezes, gerando 100 resultados distintos expressos estatisticamente por suas médias e variâncias.

Inicialmente, consideremos as Tabelas 2.6 e 2.7, que ilustram o resultado da aplicação do algoritmo genético à otimização da função de Rosenbrock.

Existe uma clara diferença entre a média dos valores finais da função obtidos utilizando o operador de recombinação BLX 0.25 e os outros métodos aqui analisados, sendo que a varição do algoritmo que utiliza essa recombinação apresentou valores muito superiores (para $N > 4$), indicando uma baixa eficiência da reprodução BLX 0.25 quando comparada aos demais métodos. Para comparar os resultados dos métodos propostos neste documento com aqueles obtidos com a utilização da reprodução do tipo linha estendida, é necessário utilizar algum método de inferência menos intuitivo do que a simples análise dos resultados, dada a similaridade existente nos dados.

Optou-se aqui pela utilização de um teste de hipóteses sobre as médias amostrais, a fim de inferir sobre uma possível diferença entre as médias populacionais dos resultados. Para tanto, são formuladas as seguintes hipóteses:

- $H_0 : \mu_1 = \mu_2$
- $H_1 : \mu_1 \neq \mu_2$

onde mu_1 é a média da população da qual foi retirada a amostra 1, e mu_2 é a média da população da qual foi retirada a amostra 2.

A Tabela 2.8 mostra um resumo do resultado obtido quando comparados os métodos de reprodução matricial e em linha estendida. Nela são mostradas as médias para os valores finais da função e a estatística do teste para o Erro Tipo I, que é o erro em se rejeitar a hipótese nula H_0.

N	n_p	Linha Estendida	Matricial	Erro Tipo I %
2	20	0.2898	0.2173	43
4	40	2.079	1.777	12
8	80	1.369	1.186	45
16	160	6.333	2.285	0
32	320	12.3	2.433	0
64	256	37.54	21.24	0

Tabela 2.8: Resultados do teste de hipóteses para os valores finais da função de Rosenbrock: Linha Estendida $\times$ Matricial

Fica evidente aqui a superioridade do método de reprodução matricial quando aplicado a problemas com maior número de variáveis, dado que, para esses casos, é possível afirmar com 100% de certeza que o método irá gerar resultados, em média, superiores àqueles obtidos com a utilização do método de linha estendida.

Resultado similar a esse é observado quando são comparados os métodos de reprodução matricial modificada e o método de linha estendida. Conforme a Tabela 2.9, pode-se notar que existe também uma diferença clara para o problema com maior número de variáveis.

N	n_p	Linha Estendida	Matricial Modificada	Erro Tipo I %
2	20	0.2898	0.4219	28
4	40	2.079	1.892	37
8	80	1.369	1.294	80
16	160	6.333	2.874	0
32	320	12.3	3.073	0
64	256	37.54	31.31	0

Tabela 2.9: Resultados do teste de hipóteses para o valor final da função de Rosenbrock: Linha Estendida $\times$ Matricial Modificada

A diversidade final da população para os métodos matricial, matricial modificado e linha estendida apresenta resultados similares, porém, inferiores àqueles observados para o método BLX 0.25 e para os métodos que utilizam a meta eurística. Esses métodos, apesar de apresentarem uma alta preservação da diversidade, não apresentaram bons resultados para o valor final da função em relação aos demais.

As Tabelas 2.10 e 2.11 ilustram o resultado da aplicação dos métodos na otimização da função de Rastrigin, que, como já foi discutido, possui vários ótimos locais.

		Linha Estendida		Matricial Modificada		Matricial		M.M. Heurística		M. Heurística		BLX 0.25	
N	n_p	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2
2	20	2e-005	2e-009	0.0099	0.0098	0.0199	0.0196	0.0765	0.1089	3e-006	2e-010	0.0219	0.0004
4	40	0.0005	5e-007	0.0001	1e-007	4e-006	1e-009	0.01082	0.0105	0.01042	0.0108	0.5055	0.0864
8	80	0.0074	6e-005	0.0030	3.807e-005	0.0199	0.0396	0.0566	0.0191	0.0243	0.0291	5.974	3.761
16	160	0.0907	0.0043	0.0076	0.0033	0.009	3e-006	0.593	0.0630	0.0075	6e-005	32.49	33.77
32	320	0.7884	0.1308	0.0901	0.0993	0.0728	0.0017	5.463	2.608	0.4975	0.0582	117.9	105.9
64	640	5.356	2.39	7.007	1.848	2.096	0.4664	35.3	57.34	12.2	15.09	336.3	247.5

Tabela 2.10: Otimização da função de Rastrigin - Valor final da função

		Linha Estendida		Matricial Modificada		Matricial		M.M. Heurística		M. Heurística		BLX 0.25	
2	20	0.4476	0.1316	0.4084	0.1295	0.5744	1.059	13.95	1785	0.6316	0.6189	59.2	704.6
4	40	1.618	1.668	1.423	3.634	1.684	1.514	34.4	1e+004	2.576	30.99	164.1	3729
8	80	5.225	10.77	4.867	5.898	6.32	14.17	22.46	4880	7.942	169.7	388.9	1e+004
16	160	19.42	85.96	16.31	113.3	21.29	179.3	115	3e+005	23.08	172.4	983.3	9e+004
32	320	77	1639	66.95	1117	84.83	1775	534.5	1.21e+007	90.49	1975	2183	5e+005
64	640	322.8	2e+004	236.4	1e+004	377.4	4e+004	1326	1e+007	370.2	4e+004	4682	2e+006

Tabela 2.11: Otimização da função de Rastrigin - Diversidade final da população

Para esta função, observam-se resultados semelhantes àqueles verificados anteriormente para a função de Rosenbrock. O teste de hipóteses, aqui omitido, revela novamente a superioridade dos re-

sultados obtidos para o método de reprodução matricial, principalmente quando aplicado ao problema com um número relativamente alto de variáveis.

Resultados muito semelhantes aos obtidos até então, a menos de pequenas diferenças pontuais, são observados na otimização das duas funções de teste restantes, como pode ser verificado nas Tabelas 2.12, 2.13, 2.14 e 2.15.

		Linha Estendida		Matricial Modificada		Matricial		M.M. Heurística		M. Heurística		BLX 0.25	
N	n_p	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2
2	20	1e-007	9e-014	4e-009	1e-016	2e-009	1e-016	1e-007	6e-013	1e-008	2e-015	0.00010	8e-009
4	40	3e-006	7e-011	1e-006	5e-012	5e-009	1e-015	5e-006	1e-010	4e-008	7e-014	0.0025	2e-006
8	80	4e-005	2e-009	4e-005	2e-009	3e-008	5e-015	0.0002	3e-008	2e-007	9e-014	0.0316	0.0001
6	160	0.0004	6e-008	0.0003	1e-007	4e-006	2e-011	0.0029	1e-006	3e-005	1e-009	0.1867	0.0012
32	320	0.0037	2e-006	0.0021	3e-006	0.0003	7e-008	0.0257	5e-005	0.0025	1e-006	0.7832	0.0099
64	640	0.0269	6e-005	0.0358	7e-005	0.0095	7e-006	0.1816	0.0009	0.0636	0.0002	2.792	0.0457

Tabela 2.12: Otimização da função Esfera - Valor final da função

		Linha Estendida		Matricial Modificada		Matricial		M.M. Heurística		M. Heurística		BLX 0.25	
2	20	1e-005	7e-010	1e-005	5e-010	1e-005	2e-010	0.0467	0.1513	2e-005	8e-010	0.0072	8e-005
4	40	3e-005	7e-010	4e-005	1e-009	4e-005	1e-009	0.0316	0.0516	6e-005	1e-008	0.0227	0.0004
8	80	0.0002	9e-009	0.0001	3e-007	0.0001	2e-008	1.04	97.16	0.0001	1e-008	0.0625	0.0024
16	160	0.0006	1e-007	0.0004	4e-008	0.0006	1e-007	0.0063	0.0014	0.00065	16e-007	0.2133	0.0436
32	320	0.0021	9e-007	0.0018	7e-007	0.0025	2e-006	0.7424	45.3	0.0024	2e-006	0.6295	0.1069
64	640	0.0089	3e-005	0.0063	1e-005	0.0093	2e-005	84.66	7e+005	0.0105	3e-005	2.377	2.451

Tabela 2.13: Otimização da função Esfera - Diversidades da população

		Linha Estendida		Matricial Modificada		Matricial		M.M. Heurística		M. Heurística		BLX 0.25	
N	n_p	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2	$f(x^*)$	σ^2
2	20	4e-008	1e-014	6e-009	2e-015	4e-009	4e-016	8e-008	2e-013	4e-009	8e-016	4e-005	1e-009
4	40	1e-006	2e-012	3e-007	3e-013	3e-008	7e-014	2e-006	1e-011	3e-008	5e-015	0.0006	1e-007
8	80	1e-005	1e-010	1e-005	2e-010	4e-007	5e-013	7e-005	3e-009	2e-006	1e-011	0.0043	1e-006
16	160	0.0001	6e-009	0.0001	6e-009	2e-005	2e-010	0.0007	1e-007	0.0001	5e-009	0.0146	8e-006
32	192	0.0011	2e-007	0.0010	1e-007	0.0004	2e-008	0.0041	1e-006	0.0022	5e-007	0.0392	2e-005
64	256	0.0030	5e-007	0.0033	4e-007	0.0022	2e-007	0.0103	4e-006	0.0081	3e-006	0.0768	4e-005

Tabela 2.14: Otimização da função de Griewank - Valor final da função

		Linha Estendida		Matricial Modificada		Matricial		M.M. Heurística		M. Heurística		BLX 0.25	
2	20	1e-006	2e-012	6e-006	1e-009	1e-006	4e-012	0.0009	2e-005	2e-006	7e-012	0.0008	7e-007
4	40	3e-006	5e-012	2e-006	1e-011	5e-006	3e-010	0.0003	9e-006	4e-006	4e-011	0.0010	7e-007
8	80	3e-006	5e-012	3e-006	3e-012	5e-006	1e-011	0.0010	4e-005	6e-006	5e-011	0.0021	6e-006
16	160	7e-006	1e-011	5e-006	1e-011	7e-006	3e-011	0.0003	77e-006	8e-006	4e-011	0.0021	2e-006
32	192	9e-006	3e-011	7e-006	1e-011	9e-006	2e-011	9e-005	3e-007	1e-005	1e-010	0.0022	1e-006
64	256	1e-005	3e-011	9e-006	2e-011	1e-005	7e-011	0.0007	3e-005	2e-005	3e-010	0.0023	2e-006

Tabela 2.15: Otimização da função de Griewank - Diversidades da população

2.8 Conclusões

Os problemas apresentados comprovam a eficiência dos métodos de reprodução propostos, quando aplicados aos casos de estudo aqui analisados, evidenciando também a sua superioridade em algumas situações, o que os torna boas alternativas de codificação para os algoritmos genéticos.

Os vários operadores de recombinação estudados apresentam diferentes características, que os tornam preferíveis em cenários de aplicação diferentes. Um exemplo claro da especialização oriunda das características particulares de cada método é a otimização de parâmetros de uma função fuzzy.

Consideremos que essa função realize um mapeamento que pode ser devidamente avaliado por uma função de *fitness* e possua parâmetros ajustáveis, conforme mostrado na Figura 2.36. Dessa forma seus parâmetros poderão ser otimizados por meio de um algoritmo genético, visando com isso, obter uma configuração ótima. Dadas as características do problema, os operadores de reprodução propostos podem ser usados neste processo, porém, com a implementação de restrições que garantam a factibilidade dos herdeiros.

O operador de recombinação: linha estendida, por sua vez, pode ser aplicado diretamente a esse problema, preservando a factibilidade dos herdeiros, sem a necessidade da aplicação de nenhuma restrição, isso devido as características intrínsecas do operador, o que o torna claramente preferível nesse cenário de aplicação.

Os algoritmos genéticos apresentam uma capacidade de busca e exploração muito útil em várias aplicações, sendo uma delas a otimização de funções, aplicação estudada neste documento. Sua utilização, por outro lado, requer certo conhecimento e experiência, como pode ser observado nos exemplos mostrados, onde resultados consideravelmente diferentes foram obtidos, através da variação dos parâmetros do algoritmo.

A codificação do problema mostra-se um fator relevante e com considerável impacto sobre os resultados obtidos, portanto, a escolha da função de *fitness*, do método de seleção, e dos operadores de reprodução e mutação, despende certa dedicação e análise para a obtenção de bons resultados para o problema ao qual o algoritmo genético será aplicado.

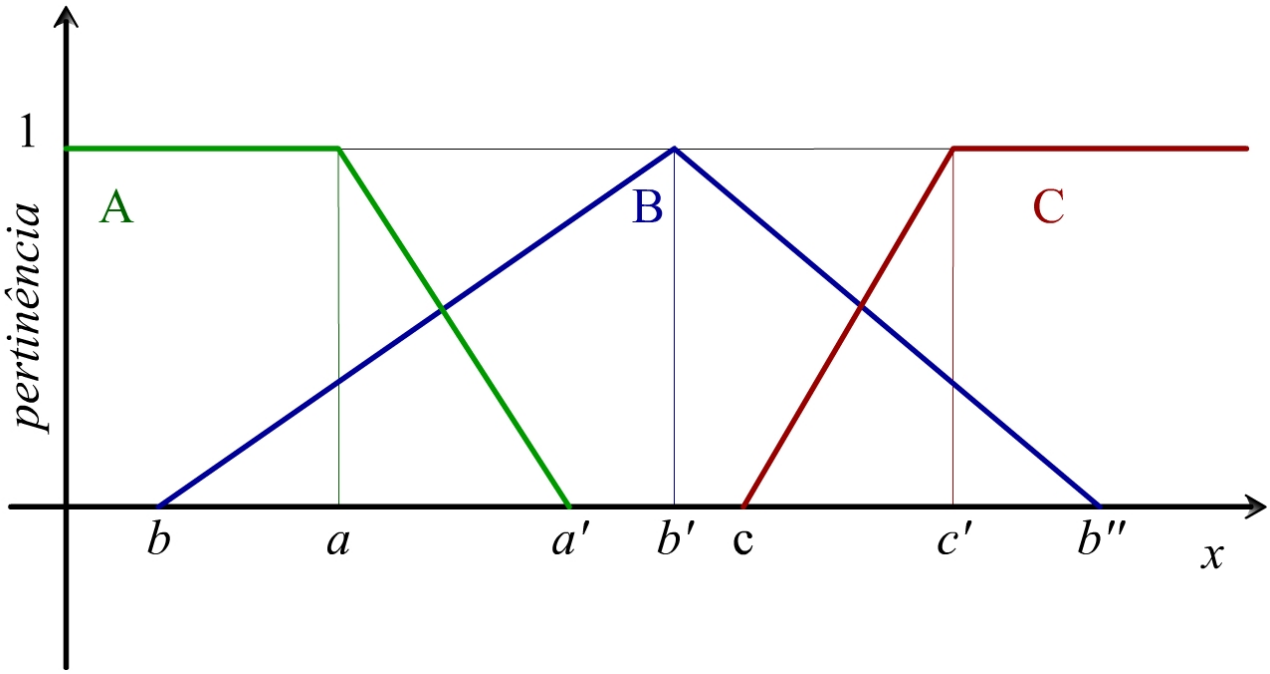

Figura 2.36: Função fuzzy contendo 3 conjuntos A B e C e parâmetros ajustáveis: a, a', b b', b'', c e c'

Capítulo 3

Otimização por Enxame de Partículas

3.1 Introdução

Assim como os algoritmos genéticos, a otimização por enxame de partículas é uma técnica bio-inspirada que faz uso de uma computação massiva para percorrer o espaço de busca a procura de seu objetivo. Esse método é baseado no conceito de inteligência de enxame, proposto inicialmente no final da década de 80 por Geraldo Beni e Jing Wang em seu artigo *"Swarm Intelligence"* [20], onde os autores utilizam o conceito de enxame para descrever sistemas robóticos formados por múltiplos agentes, que expressam relacionamentos sociais relativamente complexos, implementando apenas regras locais.

O conceito de enxame é utilizado para definir genericamente uma coleção de agentes inseridos em um ambiente com capacidade de interação local. Tomando como base a biologia, a definição de enxame pode ser aplicada ao comportamento do alguns grupos de peixes, insetos, pássaros e a micro-organismos como alguns tipos de bactérias. Em todos esses organismos, observa-se um comportamento coletivo com características similares, no qual cada indivíduo do grupo locomove-se seguindo um mesmo padrão e em geral na mesma direção dos demais.

Cada um desses animais apresenta pouca inteligência quando analisado de forma isolada, assim como pode-se observar em uma abelha, ou em uma formiga por exemplo. A coletividade do enxame, por outro lado, da origem a um comportamento sofisticado, apresentando uma inteligência muito maior do que aquela observada em um único agente separadamente.

Existem inúmeras razões biológicas que levam diferentes animais a apresentar comportamentos sociais com características de um enxame, como proteger-se de predadores que podem encontrar maior resistência ao atacar um grande número de indivíduos, encontrar alimento, principalmente quando este está distribuído de maneira desuniforme por uma região extensa, encontrar lugares se-

guros, entre outros fatores. Esse comportamento coletivo provê, a cada indivíduo do enxame, uma maior probabilidade de sobrevivência e conseqüente preservação de sua herança genética.

São encontrados na literatura alguns trabalhos voltados ao modelamento de enxames naturais, como o trabalho de Reynolds, que, em seu artigo: *Flocks, herds, and schools: A distributed behavior model* [17], estudou os padrões existentes no movimento de uma revoada de pássaros com o intuito de aplicar esses padrões em simulações computacionais para fins unicamente visuais. Os resultados obtidos pelo autor descrevem o movimento das aves de forma local, sem a influência de um controle central, que, em uma análise superficial do enxame, parece estar presente devido às características do comportamento global do grupo.

O número de pássaros que compõem uma revoada ou a quantidade de peixes em um cardume, claramente não afeta o comportamento emergente do grupo. Em diversos enxames são observados os mesmos padrões, independentemente do número de indivíduos que os compõem. Se considerarmos que cada indivíduo tem conhecimento a respeito de todos os outros, a inteligência ou capacidade de processamento de informação de cada indivíduo cresce exponencialmente com o crescimento do grupo, visto que cada animal deve interagir com todos os outros animais da população. Como a inteligência desses animais não varia com o tamanho do grupo, pode-se afirmar que cada animal do enxame tem conhecimento apenas de um número limitado de outros animais que lhe são próximos, e não da totalidade da população, o que poderia extrapolar sua capacidade de processamento de informações.

Em [21], Maja J Matarìc descreve alguns padrões comportamentais observados em um grupo de agentes que apresentam comportamento emergente similar a um enxame.

- **Vagar com segurança:** Habilidade dos agentes do grupo em vagar livremente evitando colisões.
- **Dispersão:** Capacidade de se dispersar mantendo uma distância mínima entre os agentes do grupo.
- **Agregação:** Comportamento contrário à dispersão, segundo o qual agentes tendem a se unir mantendo uma máxima distancia entre si.
- **Homing:** Habilidade do grupo de encontrar uma determinada região ou localização.

O modelo de Reynolds implementa três regras simples para simular o comportamento de uma revoada de pássaros: evitar colisões, emparelhar as velocidades e centralização. Essas regras fazem com que os animais da simulação aproximem-se cada vez mais uns dos outros (centralização), locomovendo-se com velocidades similares (emparelhar velocidades) e garantindo que nenhum animal toque nos outros de sua vizinhança (evitar colisões). A regra de centralização calculada para cada

animal foi primeiramente formulada utilizando todos os indivíduos da população, sendo este modelo intitulado pelo autor de *Central Force Model*. Este método causa uma convergência simultânea e indesejada na aplicação proposta, onde todos os animais do grupo, inicialmente espalhados por um espaço extenso, locomovem-se para um único ponto, o centróide do enxame. Posteriormente é introduzido o modelo *Localised Centering Model* em alternativa ao modelo *Central Force Model*, que tenta evitar essa convergência.

O resultado da implementação desse modelo, conhecido por *Reynold's Flocking Boids*, demonstra alguns dos princípios da vida artificial, onde um comportamento relativamente complexo emerge da interação de regras locais simples. Os modelos de otimização por enxame de partículas fazem uso dessa característica emergente, alcançando bons resultados através de uma implementação bastante simples. Esse método implementa apenas dois dos comportamentos dos quatro definidos por Matarič, *Homing* e *Agregação*, deixando de fora *Dispersão* e *Vagar com segurança*, que não agregam características úteis com relação à otimização.

Inspirados nos estudos de Heppner e Grenander [16] e em Reynolds [17], Kennedy e Eberhart apresentaram o modelo original do método de otimização por enxame de partículas em seu artigo *Particle Swarm Otimization* [22]. Nele os autores descrevem uma população de partículas onde cada uma representa um ponto do espaço euclidiano R^N. Essas partículas locomovem-se por este espaço de forma estocástica, baseando-se em regras individuais de interação local e tendo um objetivo comum regido por uma função de custo f a qual se deseja otimizar.

Cada indivíduo da população nos algoritmos de otimização por enxame de partículas é chamado de *partícula*, mesmo não possuindo massa nem volume, o que o torna candidato a ser designado por *ponto*. O termo partícula é utilizado devido ao fato de cada um dessas entidades apresentar velocidade e posição, mesmo não possuindo massa e volume, situação melhor caracterizada por uma partícula.

A princípio, Kennedy e Eberhart propõem regras bastante simples para reger o movimento de cada partícula individualmente. Essa regras podem ser expressas de maneira formal pelas equações que seguem:

$$\begin{aligned} v_i(t+1) &= v_i(t) + c_1\alpha_1(p_i - x_i(t)) + \\ &\quad + c_2\alpha_2(g - x_i(t)) \qquad (3.1) \\ x_i(t+1) &= x_i(t) + v_i(t+1) \qquad (3.2) \end{aligned}$$

onde $v_i(t) \in R^N$ é um vetor que representa a velocidade da partícula i na iteração t, $x_i(t) \in R^N$ é a posição da partícula i na iteração t, p_i é a melhor posição já registrada pela partícula, avaliada por

algum critério relativo à função de custo do problema, g é a melhor posição registrada entre todas as partículas contidas na população em toda sua história.

As constantes c_1 e c_2 são conhecidas como coeficientes de aceleração e regem a influência das posições p_i e g na composição da velocidade da partícula, definindo como essa irá acelerar em direção a melhor posição já observada pela própria partícula e na direção da melhor posição já encontrada por toda a população.

Os escalares α_1 e α_2 são escolhidos de forma aleatória para cada partícula a cada iteração do algoritmo. Essa aleatoriedade introduz características estocásticas ao processo de busca, gerando movimentos imprevisíveis e ampliando a capacidade da população de partículas em explorar o espaço de busca.

O movimento das partículas, inspirado no movimento de enxames biológicos, é influenciado por dois fatores. Um desses fatores diz respeito apenas ao próprio indivíduo do enxame e suas experiências passadas, que, por expressar o conhecimento da própria partícula, é chamado de termo cognitivo. O outro fator expressa as interações entre indivíduos e o comportamento social da população. Esse fator de influência no movimento das partículas, por ser relativo ao comportamento social das mesmas, é conhecido como termo social.

O termo $(p_i - x_i(t))$ da equação (3.1) representa o termo cognitivo do movimento da partícula, por levar em conta a experiência da própria partícula, retratada pela posição p_i. O comportamento social das partículas é dado pela expressão: $(g - x_i(t))$, que implementa a influência da população em sua totalidade na variação de posição de cada uma das partículas.

A junção destes dois termos, proporciona um método com uma grande capacidade de explorar o espaço de busca, aliado a uma boa taxa de convergência, o que não é observado se utiliza apenas um desses dois termos isoladamente [22, 23], exceto quando se utiliza apenas o termo social, situação na qual são observados bons resultados em alguns casos específicos, Porém de forma mais abrangente essa variante é inferior aquela utilizando ambos os termos social e cognitivo.

3.2 Otimização por Enxame de Partículas

Consideremos uma função $f(x)$ a qual se deseja otimizar em relação ao parâmetro x. Será aqui considerado como otimização encontrar o ponto x^* onde o valor da função atinge seu mínimo. Esta hipótese não restringe de forma alguma o problema de otimização, pois, ao se maximizar f, é obtido o mesmo resultado que ao se minimizar $-f$, dada a dualidade existente entre os dois problemas.

Dessa forma, o problema de otimização passa a ser o problema de se encontrar x^* que satisfaça:

$$f(x^*) = \beta \tag{3.3}$$

$$f(x) \geq \beta \ \forall x \neq x^* \tag{3.4}$$

Se a equação (3.4) for válida para todo $x \in R^N$, onde N é a dimensão do vetor x, então x^* será um mínimo global da função f. Caso essa desigualdade seja verificada apenas em um subconjunto $\mathcal{W}$ pertencente ao R^N contendo x^*, então x^* será um minimo local de f em $\mathcal{W}$.

O método de otimização por enxame de partículas visa encontrar x^* por meio de uma população de partículas representadas por pontos x_i de um subespaço $\mathcal{V}$ do R^N, que locomovem-se nesse espaço segundo as equações (3.1) e (3.2). Para tanto, são necessárias algumas definições e formalizações dadas a seguir.

Valor de Adaptação: β_i

Cada partícula i tem associada a ela um valor de adaptação $\beta_i(t)$, que reflete a qualidade da solução por ela apresentada na iteração t. A equação (3.5) ilustra o calculo de β_i:

$$\beta_i(t) = f(x_i(t)) \tag{3.5}$$

Posição da Partícula $x_i(t)$

Toda partícula da população tem associada a ela um vetor $x_i(t)$ que indica sua posição no subespaço $\mathcal{V}$ pertencente ao espaço euclidiano R^N. Cada vetor x_i representa uma possível solução para o problema de otimização da função f.

Velocidade da Partícula $v_i(t)$

Cada uma das partículas da população possui uma velocidade $v_i(t) \in R^n$, variante com relação a t. Essa velocidade é responsável pela locomoção da partícula pelo espaço de busca e é definida pela equação 3.1.

Melhor Posição Individual p_i

A melhor posição individual de cada partícula i da população é dada pela posição $x_i(t_0)$ para $t > t_0$, em que foi observado o menor valor de adaptação da partícula:

$$p_i = x_i(t_0)$$

$$\beta_i(t_0) \leq \beta_i(t) \ \forall t$$

Melhor Posição Global g

O ponto $g(t)$ é aquele onde a população de partículas obteve seu menor valor de adaptação até a iteração t. Esse valor pode ser definido com base nos valores de p_i como segue.

$$\begin{aligned} g(t) &= p_i(t) \\ f(g) &\leq f(p_j) \ \forall j \neq i \end{aligned}$$

A inicialização da população de partículas é feita associando uma posição x_i aleatória a cada partícula, dentro de um subespaço $\mathcal{V}$, essa população inicial contém n_p partículas, número que permanece constante durante o processo iterativo, não sendo considerados morte nem nascimento de partículas. A velocidade inicial v_i de cada partícula i da população é tomada de forma também aleatória. Porém, existe ainda a possibilidade de inicializar as velocidades em zero, dado que as posições já foram tomadas aleatoriamente [23].

A locomoção de cada partícula durante o processo iterativo ocorre em uma direção estocástica que é influenciada pela composição de sua inércia (velocidade anterior), do seu termo cognitivo p_i, e do termo social g, como é observado nas equações (3.1) e (3.2).

Após algumas iterações, espera-se que a população apresente uma maior densidade de partículas em $\mathcal{W}$, um subespaço contendo x^* e sua vizinhança. O valor retornado pelo algoritmo, que espera-se que represente o ponto ótimo da função, será expresso por g. O critério de parada do método de otimização por enxame de partículas pode ser baseado em um número máximo de operações de atualização, ou em um número máximo de iterações n_i. Uma alternativa ao critério de parada é interromper o processo iterativo, quando a melhor posição global g atingir um determinado valor. Este último critério é bastante útil quando se minimizam funções que possuam um único mínimo global em $\mathcal{V}$ bem conhecido, como por exemplo funções de erro e algumas formas quadráticas:

$$f = x^T Q x \tag{3.6}$$

onde $Q \in R^{n \times n}$ é uma matriz simétrica definida positiva. O mínimo global dessa função ocorre em: $x^* = 0$, $f(x^*) = 0$.

Porém, mesmo nestes casos onde o mínimo da função é único e tem seu valor bem determinado, é importante especificar um número máximo de iterações para garantir a convergência do método, caso contrário, é possível que o algoritmo, por uma série de razões, não seja capaz de atingir o seu objetivo, prendendo o processo iterativo em um laço infinito.

O Algoritmo 2 mostra o método original de otimização por enxame de partículas proposto por Kennedy e Eberhart [23, 22], que implementa de forma bastante simples as regras de interação local

das partículas e apresenta bons resultados, quando comparado a outras meta-heurísticas aplicadas a problemas de busca e otimização.

Inicializar População Aleatória;
for *t=1 até* n_i **do**
  for *i=1 até* n_p **do**
    $v_i(t+1) = v_i(t) + c_1\alpha_1(p_i - x_i(t)) + c_2\alpha_2(g - x_i(t))$;
    $x_i(t+1) = x_i(t) + v_i(t+1)$;
    $\beta_i = f(x_i)$;
    if $\beta_i \leq f(p_i)$ **then**
      $p_i = x_i$
    end
    if $\beta_i \leq f(g)$ **then**
      $g = x_i$
    end
  end
end

Algoritmo 2: Otimização por Enxame de Partículas, Método Original

Como a população inicial de partículas é inicializada em um subespaço $\mathcal{V}$ do R^N, espera-se que o Algoritmo 2 encontre um mínimo local da função f contido em $\mathcal{V}$, ou em alguns casos, o ponto ótimo mais próximo deste subespaço, o que ocorre geralmente quando $\mathcal{V}$ não possui nenhum ótimo local de f.

Esta variante do método é conhecida por modelo *Gbest*: nela existe apenas uma melhor posição global g para toda população. Esse ponto atua como um atrator, trazendo para si todas as partículas da população, o que eventualmente poderia fazer com que o algoritmo convergisse prematuramente para este ponto. Porém, como é realizada uma atualização constante da posição g, é prevenida a ocorrência desse fenômeno, garantindo uma convergência regular na maioria dos casos.

Uma alternativa a essa implementação do algoritmo de otimização por enxame de partículas, é a manutenção de l ótimos locais g_k. Essa variante é conhecida como modelo *Lbest*, nela existe uma subdivisão do espaço de busca em l subespaços $\mathcal{V}_k$. Cada partícula da população está associada a um desses subespaços e é atraída pela melhor posição observada pelas partículas de sua vizinhança e não pela melhor posição global de toda a população. A definição dos pontos g_k é feita com base nas seguintes equações:

$$\begin{aligned} g_k(t) &= x_i(t_0) \mid x_i \in \mathcal{V}_k \ \ \forall\, k \in \{0, 1, 2...l\} \\ g_k(t) &\leq \beta_i(t_0) \ \ \forall\, i \mid x_i \in \mathcal{V}_k \ \ e \ \ \forall\, t_0 \in [0, t] \end{aligned}$$

É possível notar que, neste modelo, uma determinada partícula i não se relaciona com todas as demais partículas da população, e sim apenas com as partículas de sua vizinhança $\mathcal{V}_k$, o que melhora a expansão da informação refletindo em boas soluções para todas as partículas [23].

O modelo *Gbest* pode ser visto como um caso especial do modelo *Lbest*, onde foi tomado $l = 1$. Comparações feitas com valores de $l = 1$ e $l > 1$, revelam que o modelo *Gbest* apresenta uma taxa de convergência superior na maioria dos casos, porém, apresenta uma maior susceptibilidade a ser atraído por um mínimo local da função objetivo, ficando preso a ele [24].

3.3 Parâmetros e Modificações no Método de Busca e Otimização Por Enxame de Partículas - PSO

Após o primeiro artigo propondo o método de busca e otimização por enxame de partículas, várias modificações foram sugeridas por diversos autores, a fim de melhorar a eficiência do método proposto originalmente. Dentre as diversas modificações propostas, pode-se citar, por exemplo, a utilização de coeficientes de aceleração c_1 e c_2 variantes no tempo [25], a escolha de parâmetros ótimos segundo algum critério que seja relevante do ponto de vista da eficiência do método [26], entre outras propostas. Algumas das modificações mais relevantes para o presente estudo são discutidas nos parágrafos que seguem.

3.3.1 Limitação de Velocidade

A fim de limitar a máxima mudança de posição que uma partícula pode realizar em uma única iteração [27], é introduzido ao método original o limitante superior de velocidade V_{max}. Esse limitante é inserido com o intuito de impedir que as partículas aumentem indefinidamente a sua velocidade, o que poderia gerar uma instabilidade no método em algumas situações específicas.

A velocidade inferior das partículas também pode ser limitada por V_{mim}, garantindo que as partículas irão realizar uma mudança de posição no mínimo igual a V_{min} a cada iteração. Esta alteração visa garantir a diversidade da população, mesmo após muitas iterações, o que conseqüentemente acarreta em aumento da capacidade exploratória do método, mesmo quando a população se encontra em uma pequena vizinhança $\mathcal{W}$ de x^*. Porém, quando mal dimensionado, esse limitante inferior pode diminuir a taxa de convergência do algoritmo, levando a resultados relativamente pobres em comparação com aqueles observados para o método original.

A utilização destes dois limitantes é bastante simples, e inclui apenas duas condições adicionais no procedimento original, resultando no Algoritmo 3:

Inicializar População Aleatória;
for *t=1 até* n_i **do**
  for *i=1 até* n_p **do**
    $v_i(t+1) = v_i(t) + c_1\alpha_1(p_i - x_i(t)) + c_2\alpha_2(g - x_i(t))$;
    if $norma(v_i(t+1)) > V_{max}$ **then**
      $v(t+1) = \frac{V_{max}}{norma(v_i(t+1))} v_i(t+1)$
    end
    if $norma(v_i(t+1)) < V_{min}$ **then**
      $v(t+1) = \frac{V_{min}}{norma(v_i(t+1))} v_i(t+1)$
    end
    $x_i(t+1) = x_i(t) + v_i(t+1)$;
    $\beta_i = f(x_i)$;
    if $\beta_i \leq f(p_i)$ **then**
      $p_i = x_i$
    end
    if $\beta_i \leq f(g)$ **then**
      $g = x_i$
    end
  end
end

Algoritmo 3: Otimização por Enxame de Partículas Com Limitação de Velocidade

A título de exemplo, consideremos a função de Rosenbrock com 10 variáveis, que possui mínimo global em $x^* = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]^T$. O gráfico mostrado na Figura 3.1 exibe o resultado médio de 100 repetições do processo de otimização da função em questão, utilizando o método original e o método com limitação de velocidade, onde foi escolhido um limitante inferior $V_{mim} = 0.001$.

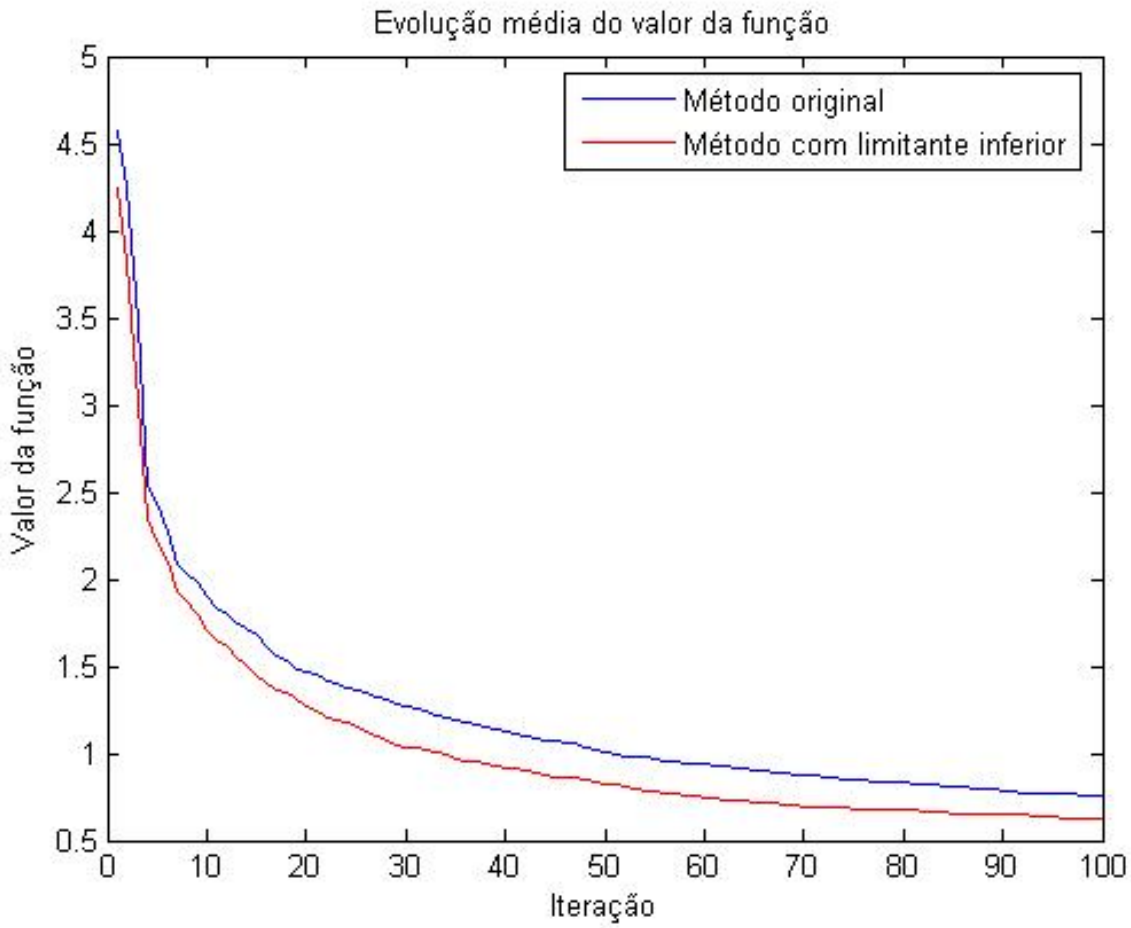

Figura 3.1: Resultado da otimização, $V_{mim} = 0,001$

Como o método de otimização por enxame de partículas tende a atrair todos os indivíduos para um único ponto, é clara a relação entre convergência do algoritmo e a queda de diversidade da população. Uma diminuição prematura nessa diversidade pode dificultar a movimentação das partículas após muitas iterações do método, tornando a exploração do espaço de busca ineficiente.

A introdução do limitante inferior de velocidade visa prevenir esse fenômeno, garantindo uma diversidade minima a população. Apesar de dificultar a total convergência das partículas para um único ponto, esse limitante aumenta significativamente a capacidade exploratória do método, em uma fase bastante oportuna do processo iterativo, quando o algoritmo encontra dificuldades em explorar o espaço de busca devido a queda na diversidade da população.

O limitante inferior de velocidade, mesmo trazendo um ganho em convergência para o algoritmo, precisa ser utilizado com bastante cautela, pois, quando mal dimensionado, esse limitante pode causar um decréscimo acentuado na taxa de convergência do algoritmo, tornando-o bastante ineficiente, como pode ser observado no gráfico da Figura 3.2, onde foi utilizado um limitante inferior $V_{min} = 0.5$.

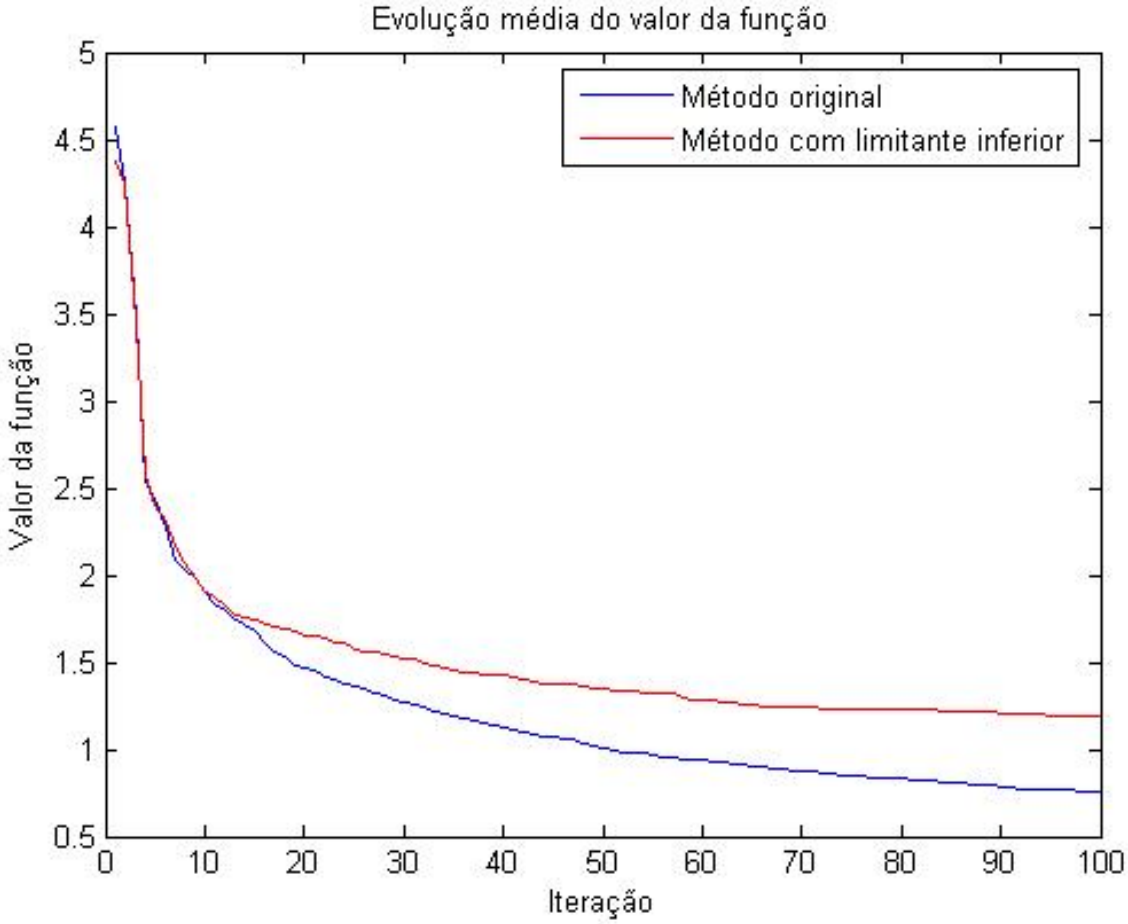

Figura 3.2: Resultado da otimização, $V_{min} = 0.5$

3.3.2 Peso de Inércia w

Este fator visa controlar a influência da velocidade que a partícula possuía na iteração anterior, que representa a inércia da partícula, sobre a velocidade da iteração corrente, de forma a proporcionar uma maior eficiência ao algoritmo de otimização [28, 27]. A modificação necessária para introduzir o peso de inércia w no método original consiste em inserir w à equação de atualização de velocidade, que passa a ser dada pela equação (3.7). A atualização da velocidade do método original pode ser então obtida escolhendo $w = 1$.

$$v_i(t+1) = wv_i(t) + c_1\alpha_1(p_i - x_i(t)) + c_2\alpha_2(g - x_i(t)) \tag{3.7}$$

O efeito da adição deste termo, quando é determinado um $w > 1$, pode ser observado escolhendo $c_1 = c_2 = 0$. A velocidade da partícula irá aumentar constantemente a cada iteração, assim, quando

$c_1, c_2 \neq 0$, a influência dos coeficientes de aceleração irá diminuir a cada iteração, dado que será gradativamente maior a influência da inércia, que é constantemente amplificada. Portanto escolhendo um $w > 1$ aumenta-se capacidade exploratória do método, devido ao aumento da velocidade, porém, ao custo de um sério comprometimento de sua taxa de convergência, pois será menor a atração exercida pelos pontos p_i e g, podendo causar uma aceleração descontrolada das partículas.

Escolhendo agora um $w < 0$, com $c_1 = c_2 = 0$. A velocidade irá diminuir de forma constante até cessar o movimento das partículas. Neste caso quando são escolhidos $c_1, c_2 \neq 0$, a influência dos coeficientes de aceleração na determinação da posição da partícula irá aumentar gradativamente durante a evolução do processo iterativo, enquanto que a influência da inércia irá diminuir. Esses fatores claramente aumentam a taxa de convergência do método, porém, além de degradarem sua capacidade de exploração, podem causar uma convergência prematura para um ponto que não representa satisfatoriamente um ótimo local, estando distante do objetivo do algoritmo.

É evidente a existência de um compromisso entre capacidade de exploração e taxa de convergência que deve ser preservado na escolha de w. Em [28], é feito um estudo com w variando entre 0 e 1.4, bem como variando w com o tempo durante o processo iterativo. Com base nestes estudos, os autores chegaram a conclusão que uma escolha apropriada seria w pertencente ao intervalo: $[0.8; 1.2]$. Ficou constatado também que $w = 1$ é um valor preferível para esta variável em várias aplicações, o que retorna ao método original.

3.3.3 Fator de Constrição

Uma forma de criar um modelo de otimização por enxame de partículas que se auto adapta a seus parâmetros c_1 e c_2 é utilizar o fator de constrição χ (é importante salientar que χ é escolhido no início do processo iterativo com base nos parâmetros utilizados e não se adapta durante esse processo). Esse fator possibilita a concepção de um algoritmo mais eficiente que o original, sem levar em conta o peso de inércia e os limitantes superior e inferior da velocidade, através da equação (3.8) que utiliza o termo de constrição para a atualização da velocidade das partículas [29, 30, 31].

$$v_i(t+1) = \chi \left[v_i(t) + c_1\alpha_1(p_i - x_i(t)) + c_2\alpha_2(g - x_i(t)) \right] \tag{3.8}$$

onde χ é descrito como segue:

$$\chi = \frac{2}{\left| 4 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|} \tag{3.9}$$

$$\varphi = c_1 + c_2 \tag{3.10}$$

Um estudo comparativo da eficiência alcançada ao se utilizar o fator de constrição, com relação ao algoritmo PSO original, pode ser encontrada em [32], onde são obtidos bons resultados usando apenas esse fator, sem a utilização do peso de inércia e dos limitantes de velocidade.

Esse fator resulta geralmente em uma convergência mais rápida do método [27], dado que a velocidade das partículas será reduzida e com isso também oscilações indesejáveis que dificultam a convergência. Apesar de aumentar satisfatoriamente a taxa de convergência do método, o uso do fator de constrição traz uma perda de capacidade exploratória, que em alguns casos pode se tornar bastante significativa, podendo fazer com que o método falhe quando iniciado muito distante do seu objetivo [23].

3.4 Modulação de Velocidade

Uma análise mais atenta do método original de otimização por enxame de partículas revela uma tendência dessa técnica de criar oscilações em torno de um ponto de ótimo da função após algumas iterações. Para exemplificar esse fenômeno, consideremos um enxame relativamente pequeno, contendo 3 partículas, utilizado para otimizar a função de Rosenbrock com 2 variáveis x_1 e x_2. A trajetória descrita pelas partículas da população pode ser observada no gráfico mostrado na Figura 3.3.

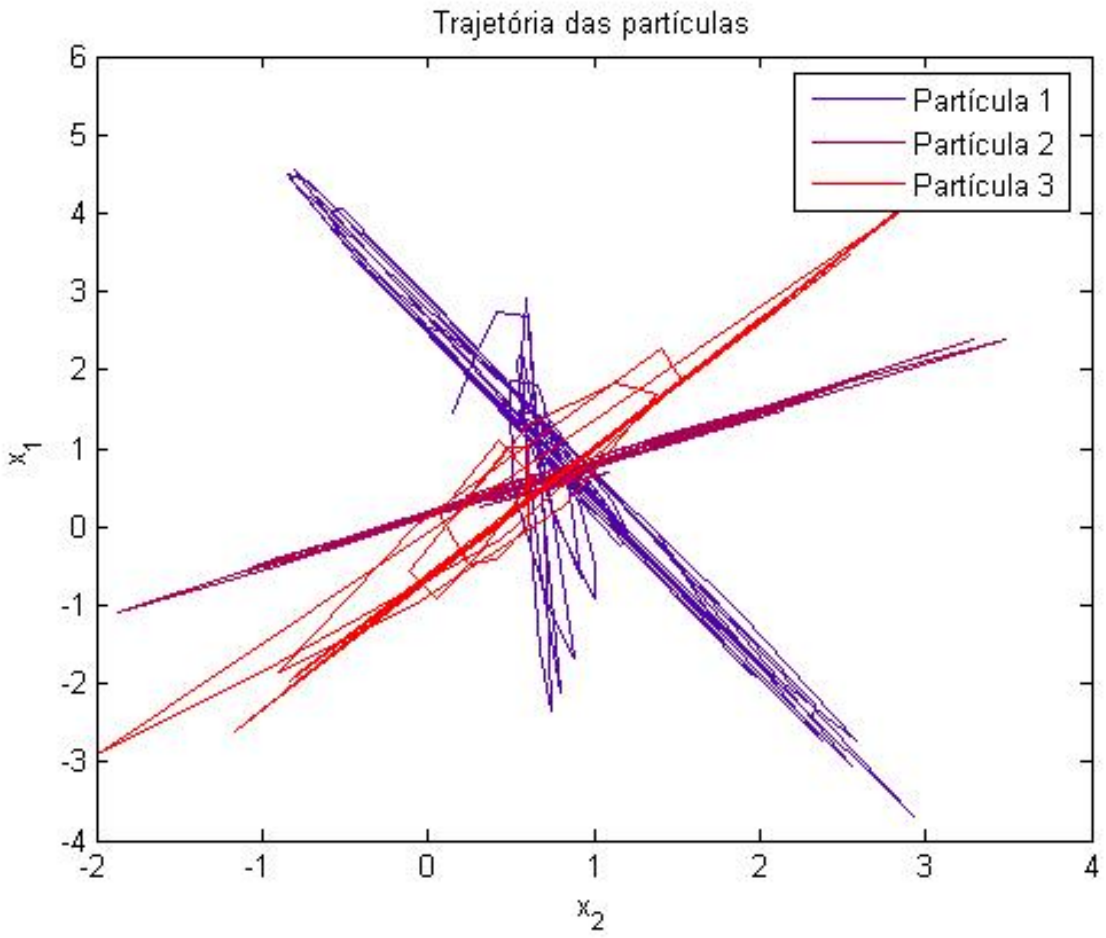

Figura 3.3: Trajetórias das partículas, PSO original

É nítido na Figura 3.3 a grande quantidade de oscilações em torno do mínimo global da função ($x = [1, 1]^T$), sendo que essas possuem uma amplitude considerável. Essa oscilações, apesar de expressarem a capacidade exploratória do método, são desnecessárias e dificultam o processo de convergência do algoritmo, sendo a sua amortização, uma possível maneira de aumentar a velocidade com que as partículas convergem para o seu objetivo.

O presente documento propõe um método que suaviza a ocorrência destas oscilações [1, 33], provendo um conseqüente aumento na taxa de convergência do algoritmo, semelhante ao que pode ser observado quando se utiliza um peso de inércia $w < 1$ ou um limitante superior de velocidade adequado, ou ainda quando é utilizada a equação de atualização da velocidade com fator de constrição. Esses métodos citados acarretam uma perda significativa na capacidade exploratória do método de otimização por enxame de partículas, sendo essa perda maior quanto maior for o aumento alcançado na taxa de convergência, explicitando assim o compromisso existente entre esses dois fatores, que

deve ser respeitado na escolha dos parâmetros quando utilizados os métodos tradicionais de otimização por enxame de partículas.

A intenção aqui é proporcionar uma maneira de reduzir as oscilações, *frenando* as partículas, porém, somente quando for necessário. Com isso, evita-se uma conseqüente perda na capacidade exploratória do método. Para tanto, é proposta a seguinte equação de modulação para a velocidade:

$$v_i^m(t+1) = \gamma(\theta) v_i(t+1) \tag{3.11}$$

onde γ é uma função monotonicamente contínua tal que: $\gamma(\theta) \leq 1 \ \forall\theta$, responsável por prover a modulação desejada na velocidade e θ representa o ângulo existente entre duas velocidades consecutivas, apresentadas por uma mesma partícula, como é exemplificado na Figura 3.4. A função γ é definida como segue:

$$\gamma = e^{\alpha(\cos(\theta)-1)} \tag{3.12}$$

$$\theta = \angle(v_i(t+1), v_i(t)) \tag{3.13}$$

onde α é um escalar a ser definido

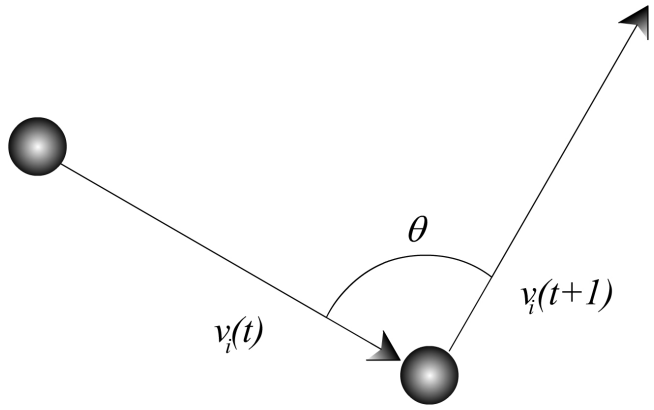

Figura 3.4: Ângulo θ

A idéia principal do método é garantir que as partículas não serão inibidas de acelerar quando estiverem longe do seu objetivo, preservando a capacidade exploratória do método original. Na vizinhança do ponto ótimo da função objetivo, a modulação proposta provê uma *frenagem* das partículas através da equação (3.11), onde espera-se que esta redução na velocidade aumente a taxa de convergência do algoritmo dentro do subespaço $\mathcal{W}$ que contém x^*.

A observação das trajetórias indica que, quando distantes do objetivo, as partículas apresentem velocidades consecutivas alinhadas e em um mesmo sentido, ou ao menos em direções muito pró-

ximas, apontando para o subespaço $\mathcal{W}$ que contém x^* e uma pequena vizinhança desse ponto. Esta situação é claramente observada quando analisadas as trajetórias das partículas, no caso onde o algoritmo foi inicializado em uma região $\mathcal{V}$ muito distante de $\mathcal{W}$. Neste caso a modulação proposta não interfere na velocidade que pode crescer indiscriminadamente enquanto as velocidades consecutivas estiverem na mesma direção e no mesmo sentido, mantendo a capacidade exploratória máxima para o método de otimização por enxame de partículas.

No caso contrário, situação na qual as partículas se encontram em uma vizinhança $\mathcal{W}$ de x^*, estas apresentam um movimento oscilatório em torno deste ponto, onde seria interessante reduzir a velocidade das partículas criando assim um ajuste fino das posições. Este procedimento aumenta consideravelmente a taxa de convergência do método dentro de $\mathcal{W}$. Novamente a modulação proposta provê a variação de velocidade necessária para o cenário apresentado, dado que a mesma reduz a velocidade da partícula sempre que forem constatadas velocidades consecutivas em direções diferentes, sendo maior a redução quanto maior for o ângulo entre as velocidades.

O parâmetro α da equação de modulação, determina o fator de redução da velocidade; neste trabalho foi utilizado um $\alpha = 0,3466$ correspondente a uma redução de 50% na velocidade quando $\theta = \pi$. Este valor é obtido da seguinte maneira:

$$\begin{aligned} \gamma(\frac{\pi}{2}) &= 0,5 \\ e^{\alpha(\cos(\pi)-1)} &= 0,5 \\ e^{-2\alpha} &= 0,5 \\ \alpha &= -\frac{\ln(0,5)}{2} \\ \alpha &= 0,3466 \end{aligned} \tag{3.14}$$

A seguir, é mostrado o algoritmo de otimização por enxame de partículas utilizando o conceito de modulação de velocidade proposto neste trabalho.

Inicializar População Aleatória;
for *t=1 até* n_i **do**
  for *i=1 até* n_p **do**
    $v_i(t+1) = v_i(t) + c_1\alpha_1(p_i - x_i(t)) + c_2\alpha_2(g - x_i(t))$;
    $\theta = ang(v_i(t+1), v_i(t))$;
    $v_i^m(t+1) = \gamma(\theta)v_i(t+1)$;
    $x_i(t+1) = x_i(t) + v_i^m(t+1)$;
    $\beta_i = f(x_i)$;
    if $\beta_i \leq f(p_i)$ **then**
      $p_i = x_i$
    end
    if $\beta_i \leq f(g)$ **then**
      $g = x_i$
    end
  end
end

Algoritmo 4: Algoritmo de otimização por enxame de partículas com modulação de velocidade

O resultado da proposta aqui apresentada pode ser observado no gráfico mostrado na Figura 3.5, que ilustra as trajetórias descritas pelas partículas para o mesmo problema exposto anteriormente, porém, desta vez utilizando o algoritmo com modulação de velocidade.

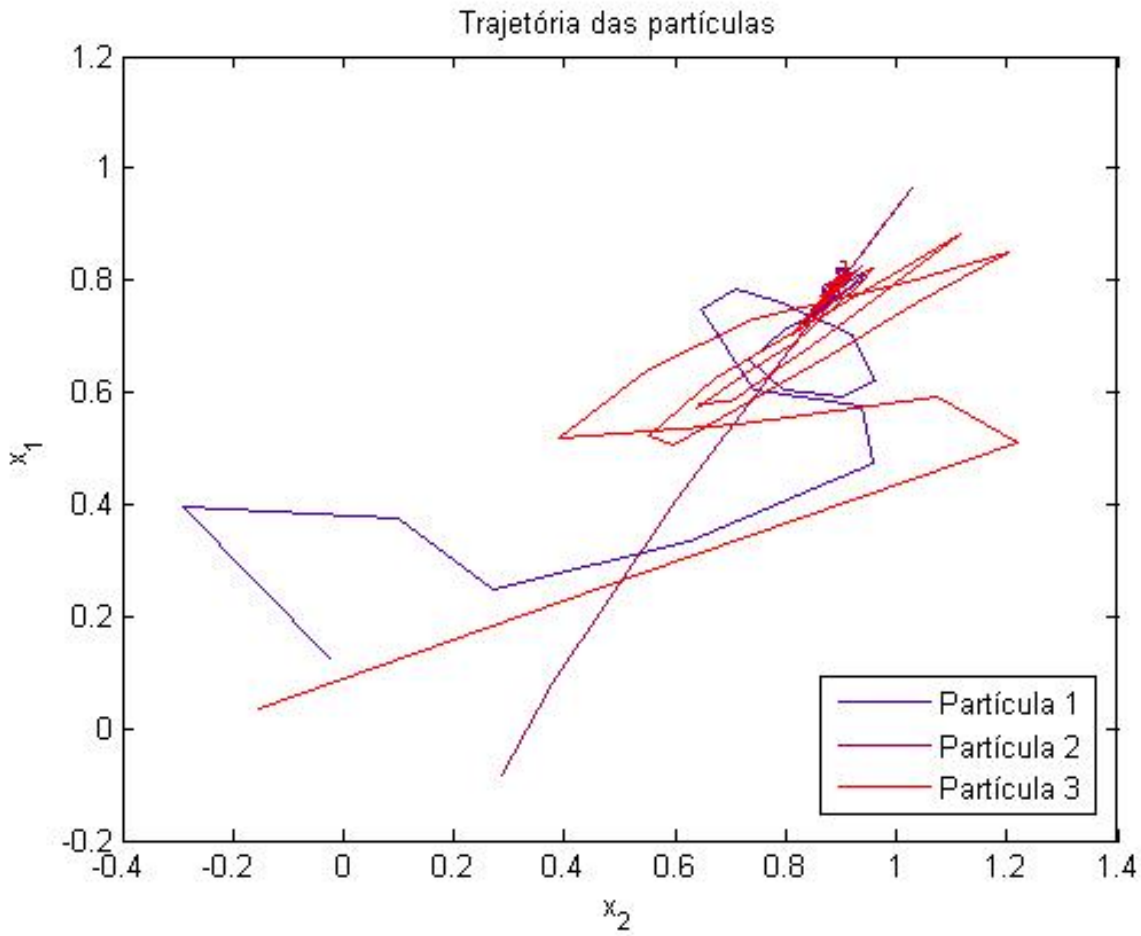

Figura 3.5: Trajetória das partículas

É evidente no exemplo apresentado, o comportamento superior em regularidade das partículas quando utiliza-se a modulação de velocidade proposta, em comparação ao método original. O objetivo almejado de reduzir as oscilações é claramente alcançado e expresso nesse exemplo de aplicação. A suavização das oscilações criada pela modulação possui a capacidade de prover um significativo aumento na taxa de convergência do método de otimização por enxame de partículas, como será observado em uma série de exemplos discutidos nos parágrafos que seguem.

3.5 Exemplos e Comparações

Cada uma das variações do método de otimização por enxame de partículas estudadas neste documento, apresenta um comportamento diferente com relação à sua taxa de convergência e dispersão das partículas. A eficiência relativa alcançada com o uso dessas variações será aqui avaliada com

o auxilio de quatro funções de teste, dadas a seguir, tendo estas sido definidas no Capítulo 2, onde avaliação similar é realizada:

- Rosenbrock.
- Rastrigin.
- Esfera.
- Griewank.

A característica estocástica do processo de otimização por enxame de partículas novamente leva ao uso de ferramentas de estatística para uma melhor compreensão e análise dos resultados. Para cada caso de análise, serão realizadas 100 repetições do processo de otimização, sendo que todo valor aqui apresentado refere-se à média e/ou variância amostrais obtidas ao final dessas repetições. Os algoritmos analisados serão: o método proposto de modulação de velocidade, o método com fator de constrição, por ser em geral superior em eficiência aos outros métodos encontrados na literatura, e por fim, o método originalmente proposto de otimização por enxame de partículas. São adotados alguns parâmetros, descritos a seguir, que estão presentes em todas as análises realizadas, exceto quando for dito o contrário.

- Número de iterações: $n_i = 80$;
- $c_1 = c_2 = 1$;
- $v_{min} = 0.001$;

3.5.1 Função de Rosenbrock

Em uma primeira análise, consideremos o problema de otimização da função de Rosenbrock. Os resultados médios obtidos com a aplicação dos três algoritmos em questão podem ser verificados nas Tabelas 3.1 e 3.2, que ilustram as médias dos valores finais da função, obtidos ao final do processo iterativo utilizando cada um dos métodos analisados, e a dispersão das partículas também ao final desse processo.

A dispersão das partículas, ou diversidade da população, pode ser utilizada para medir a capacidade exploratória da população de partículas, sendo assim, um parâmetro relevante na análise do algoritmo. Esse parâmetro será aqui estimado pela variância observada nos valores de adaptação β_i das partículas, conforme equação (3.15):

$$diversidade = var\left(\beta_1, \beta_2, ..., \beta_{n_p}\right) \tag{3.15}$$

onde $diversidade$ representa a diversidade da população e $var(\cdot)$ é a função de variância, definida no apêndice B.

N	n_p	Método Com Modulação		Método Original		Método Com Constrição	
		média	σ	média	σ	média	σ
2	28	8.046e-007	2.257e-012	0.02621	0.004249	4.538e-011	1.067e-019
4	40	0.5389	1.274	2.611	8.019	0.912	2.117
8	57	5.896	3.176	19.6	178	5.702	3.778
16	80	16.96	10.94	75.22	648.1	17.25	12.41
32	113	53.79	146.7	221.1	3718	57.01	155.6
64	160	158.4	1356	498.6	1.414e+004	169.4	748.8
128	226	401.5	7139	1026	5.149e+004	429.9	3948
256	320	902	4.984e+004	2050	9.749e+004	942	1.908e+004

Tabela 3.1: Rosenbrock: Resultados para o valor final da função

N	n_p	Método Com Modulação		Método Original		Método Com Constrição	
		média	σ	média	σ	média	σ
2	28	0.04652	0.0613	5.01e+004	2.25e+011	0.03449	0.02471
4	40	0.01409	0.002615	1.983e+004	5.139e+009	2.759e-005	1.109e-008
8	57	0.01911	0.00885	1.741e+006	2.765e+014	4.775e-005	4.971e-008
16	80	0.06046	0.0365	6.525e+005	1.483e+012	0.0002135	1.036e-006
32	113	0.7111	8.26	3.738e+006	1.368e+014	0.001254	1.036e-005
64	160	5.019	479	2.045e+007	1.749e+015	0.00938	0.0006008
128	226	46.5	1.094e+005	7.507e+007	1.964e+016	0.02927	0.002916
256	320	211.9	1.805e+006	3.291e+008	5.36e+017	0.1183	0.05593

Tabela 3.2: Rosenbrock: Resultados para a diversidade final da população

O algoritmo que utiliza o método proposto de modulação de velocidade, mostra-se claramente superior ao método original assim como também o é o método com fator de constrição, conclusão essa que pode ser afirmada com clareza pela simples observação dos resultados mostrados na Tabela

3.1. A diversidade final da população para o método proposto, porém, é bastante inferior àquela observada quando utilizado o algoritmo original, sendo por outro lado consideravelmente superior à diversidade apresentada pelo algoritmo com fator de constrição, indicando uma maior capacidade exploratória do método proposto com relação a este.

Uma comparação entre os valores finais da função obtidos para o método com fator de constrição e o método proposto, não pode ser feita pela simples observação das tabelas, dada a grande semelhança entre os valores médios obtidos. Dessa forma será utilizado o teste de hipóteses para análise desses resultados.

A Tabela 3.3 ilustra o resultado do teste de hipótese, mostrando o erro cometido em se rejeitar a hipótese nula, ou seja, o erro que se comete ao afirmar que existe diferença nas *médias populacionais* dos resultados da aplicação desses dois algoritmos ao problema em questão.

N	n_p	Modulação de Velocidade	Fator de Constrição	Erro Tipo I %
2	28	8.046e-007	4.538e-011	0
4	40	0.5389	0.912	4
8	57	5.896	5.702	46
16	80	16.96	17.25	54
32	113	53.79	57.01	6
64	160	158.4	169.4	1
128	226	401.5	429.9	0
256	320	902	942	2

Tabela 3.3: Resultado do teste de hipóteses - valor final da função de Rosenbrock

Do resultado do teste de hipóteses, conclui-se que o método proposto apresenta um resultado superior para problemas de ordem mais elevada, superioridade essa que se evidencia cada vez mais com o aumento do número de variáveis do problema.

O resultado expresso nas tabelas anteriormente discutidas é relativo apenas ao final do processo de otimização, não trazendo informação a respeito da dinâmica do processo iterativo. A fim de avaliar essa característica, a Figura 3.6 ilustra a media do valor da função obtida a cada iteração pelos três métodos em estudo, utilizando a seguinte configuração:

- $N = 32$;
- $n_p = 128$;
- $n_i = 100$

- $c_1 = c_2 = 1$
- $v_{min} = 0.001$

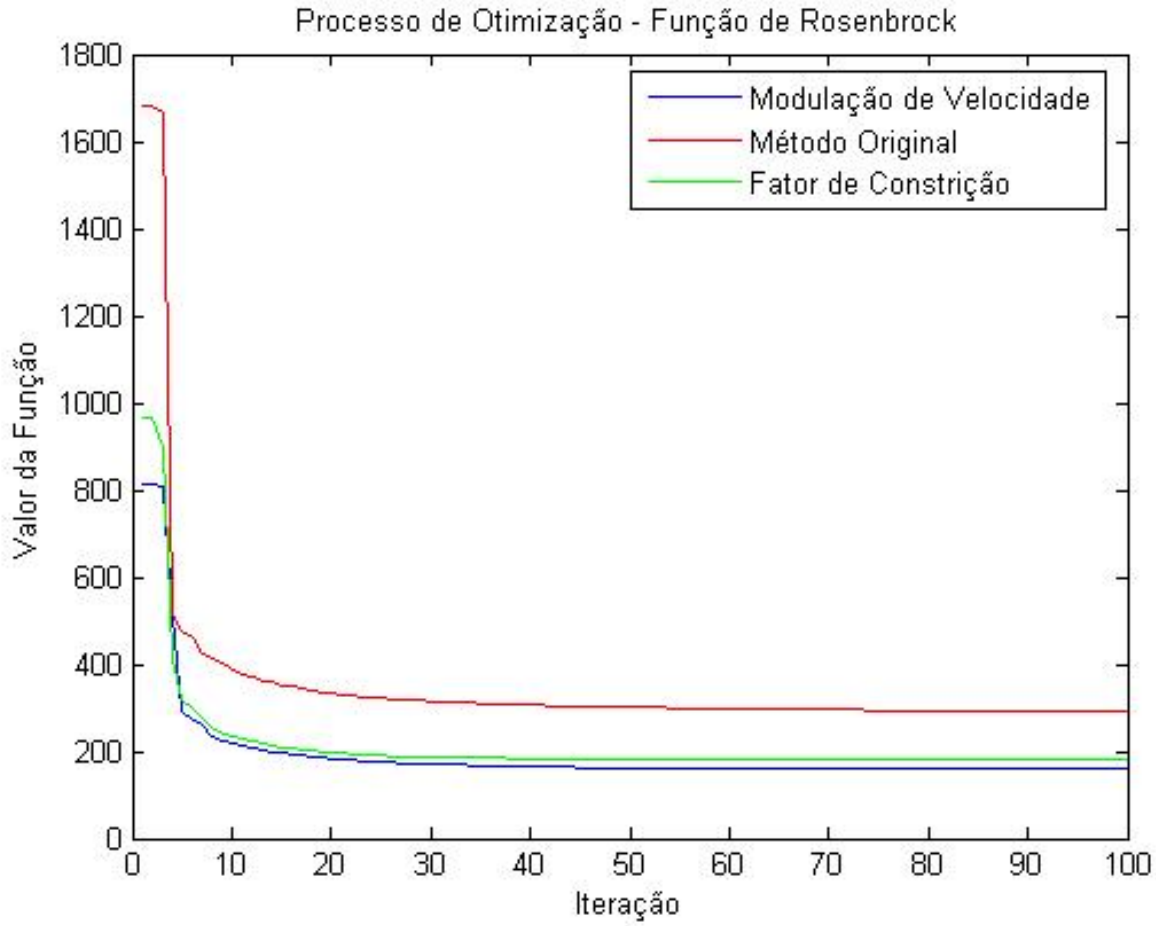

Figura 3.6: Evolução do valor da função - Rosenbrock

Esse gráfico confirma os resultados até agora obtidos, evidenciando a superioridade dos métodos com modulação de velocidade e com fator de constrição com relação ao método original, não somente com relação ao resultado final do processo iterativo mais também durante toda a execução desse processo.

3.5.2 Função de Rastrigin

Dando continuidade às análises, é tratado o problema de otimização da função de Rastrigin, sendo mostradas as médias dos valores finais na função na Tabela 3.4 e a dispersão média das partículas na ultima iteração na Tabela 3.5.

N	n_p	Método Com Modulação		Método Original		Método Com Constrição	
		média	σ	média	σ	média	σ
2	28	0.01991	0.0196	0.1125	0.0872	0.01059	0.009928
4	40	0.1095	0.1379	1.103	1.812	0.0796	0.09359
8	57	0.1795	0.3077	4.319	12.28	0.1492	0.1675
16	80	0.145	0.4661	19.5	148.5	0.1558	0.3786
32	113	0.27	0.03375	58.85	1205	0.5606	0.1099
64	160	3.826	1.714	108.7	5408	5.8	2.358
128	226	17.73	9.353	165.2	6370	21.59	14.49
256	320	50.07	34.45	296.3	1940	58.36	53.01

Tabela 3.4: Rastrigin: Resultados para o valor final da função

N	n_p	Método Com Modulação		Método Original		Método Com Constrição	
		média	σ	média	σ	média	σ
2	28	0.0004319	1.857e-005	2.555	5.076	0.002157	0.0001685
4	40	0.0004034	1.611e-005	25.5	184	1.101e-005	1.211e-008
8	57	8.936e-006	1.816e-009	152.8	3487	2.152e-013	4.186e-024
16	80	8.4e-005	3.542e-007	636.3	6.501e+004	5.228e-008	1.077e-013
32	113	0.0006771	1.001e-005	2584	8.702e+005	1.513e-006	7.268e-012
64	160	0.003655	0.0001342	1.056e+004	2.183e+007	1.746e-005	6.632e-010
128	226	0.01287	0.0002419	4.146e+004	1.992e+008	0.0001229	5.714e-008
256	320	0.06398	0.01034	1.598e+005	1.123e+009	0.0003823	6.818e-007

Tabela 3.5: Rastrigin: Resultados para a diversidade final da população

Resultados semelhantes àqueles observados para a função de Rosenbrock são obtidos nessa análise, onde o algoritmo proposto é claramente superior ao algoritmo original em convergência e superior também ao algoritmo com fator de constrição em preservação da diversidade. O teste de hipóteses, aplicado aos valores finais da função obtidos pelos algoritmos com modulação de velocidade e fator de constrição, revela que o método de modulação de velocidade proposto provê resultados superiores para problemas de ordem elevada. Por outro lado, não é possível afirmar nada (a menos que se admita um erro consideravelmente alto nessa afirmação) a respeito da diferença entre as médias obtidas para problemas com um menor número de variáveis, como pode ser verificado na Tabela 3.6.

N	n_p	Modulação de Velocidade	Fator de Constrição	Erro Tipo I %
2	28	0.01991	0.01059	58
4	40	0.1095	0.0796	53
8	57	0.1795	0.1492	66
16	80	0.145	0.1558	91
32	113	0.27	0.5606	0
64	160	3.826	5.8	0
128	226	17.73	21.59	0
256	320	50.07	58.36	0

Tabela 3.6: Resultado do teste de hipóteses - valor final da função de Rastrigin

A Figura 3.7 ilustra a evolução do valor da função durante o processo iterativo de otimização da função de Rastrigin, utilizando a seguinte configuração para os três algoritmos em questão:

- $N = 32$;
- $n_p = 128$;
- $n_i = 100$
- $c_1 = c_2 = 1$
- $v_{min} = 0.001$

Observa-se nessa figura resultados semelhantes àqueles obtidos para a função de Rosenbrock, tendo os algoritmos com modulação de velocidade e com fator de constrição obtido resultados superiores àqueles apresentados pelo original, o que é constatado durante todo o processo iterativo.

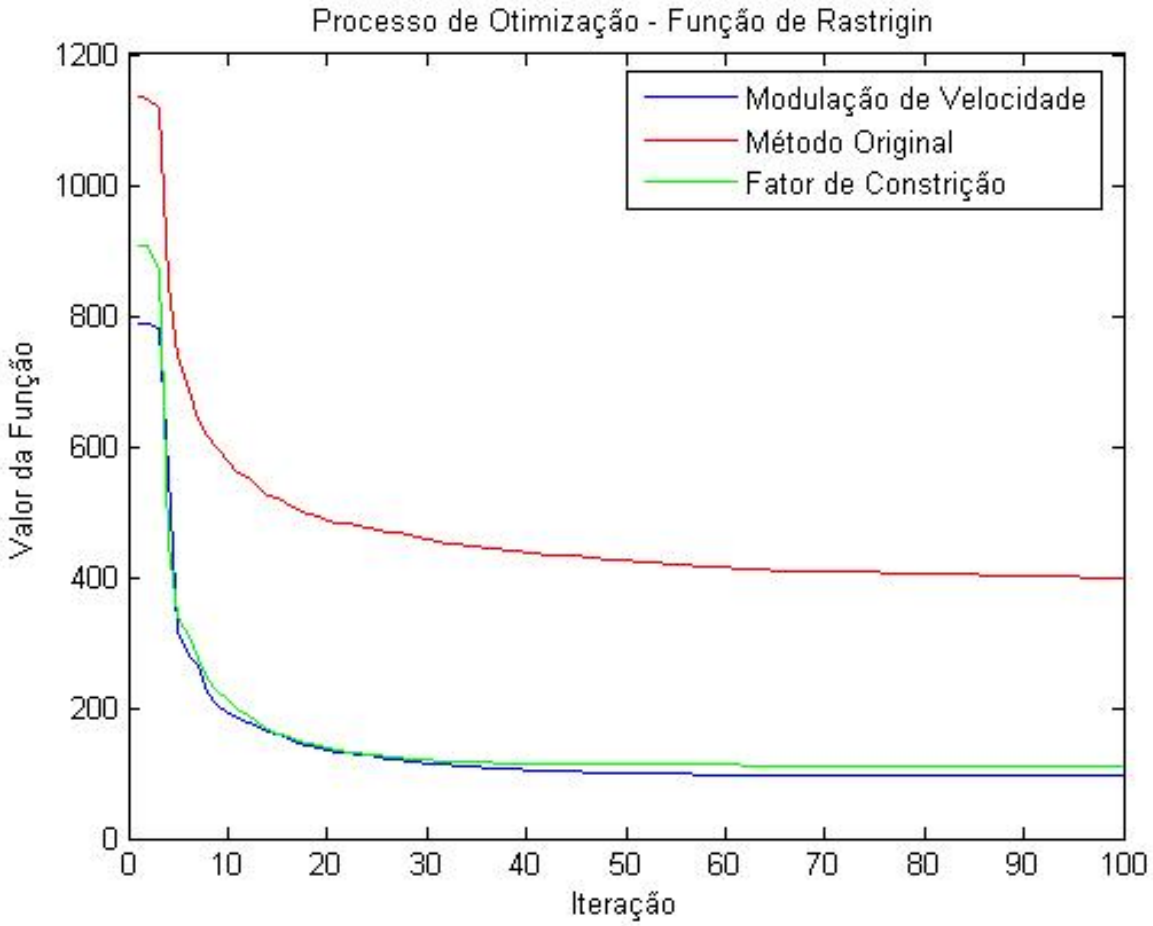

Figura 3.7: Evolução do valor da função - Rastrigin

3.5.3 Funções Esfera e de Griewank

Os resultados obtidos para a função de Rosenbrock e para a função de Rastrigin se estendem para as duas outras funções de teste analisadas: Esfera e de Griewank, como é possível observar nas Tabelas 3.7, 3.8, 3.9 e 3.10, que ilustram as medias e variâncias dos valores finais da função e das diversidades finais das populações obtidos para essas duas funções de teste. Esses resultados são bastante semelhantes àqueles já analisados, sendo as conclusões até aqui obtidas extensíveis também a esses resultados, evidenciando a alta taxa de convergência e preservação da diversidade apresentadas pelo método proposto neste documento.

N	n_p	Método Com Modulação		Método Original		Método Com Constrição	
		média	σ	média	σ	média	σ
2	28	6.274e-010	5.687e-019	2.104e-005	5.945e-010	8.14e-016	1.765e-030
4	40	5.369e-009	2.988e-017	0.0003921	1.194e-007	1.916e-014	7.63e-028
8	57	4.701e-008	1.244e-015	0.004854	9.272e-006	3.958e-012	3.721e-023
16	80	5.608e-006	4.916e-011	0.03221	0.0001958	1.127e-005	3.578e-010
32	113	0.001191	5.961e-007	0.1139	0.0008482	0.00221	2.128e-006
64	160	0.01813	2.54e-005	0.3043	0.004697	0.02505	5.022e-005
128	226	0.08238	0.0002149	0.6572	0.01041	0.1076	0.0003016
256	320	0.2494	0.0007729	1.333	0.04603	0.2923	0.001295

Tabela 3.7: Esfera: Resultados para o valor final da função

N	n_p	Método Com Modulação		Método Original		Método Com Constrição	
		média	σ	média	σ	média	σ
2	28	8.248e-009	6.781e-015	0.0004196	1.15e-006	3.356e-021	1.051e-040
4	40	5.589e-011	5.276e-020	0.00155	6.434e-006	7.385e-020	1.271e-037
8	57	8.426e-011	1.071e-019	0.009429	0.0002536	2.049e-019	1.02e-036
16	80	2.261e-009	1.225e-016	0.03777	0.0009405	4.248e-014	2.083e-026
32	113	7.9e-009	6.834e-016	0.1807	0.02079	2.536e-011	2.455e-021
64	160	5.603e-008	2.013e-014	0.7159	0.1411	5.438e-010	8.25e-019
128	226	4.602e-007	1.156e-012	3.021	2.488	2.031e-009	1.179e-017
256	320	2.691e-006	7.567e-011	12.41	15.73	9.478e-009	1.595e-016

Tabela 3.8: Esfera: Resultados para a diversidade final da população

N	n_p	Método Com Modulação		Método Original		Método Com Constrição	
		média	σ	média	σ	média	σ
2	28	3.27e-010	1.418e-019	9.589e-006	1.788e-010	2.92e-016	2.566e-031
4	40	1.212e-009	1.123e-018	0.0001133	1.051e-008	4.734e-015	4.289e-029
8	57	7.905e-009	4.905e-017	0.0009605	4.552e-007	2.098e-011	3.281e-020
16	80	5.145e-006	4.686e-011	0.00316	1.553e-006	6.689e-006	7.805e-011
32	113	0.0001754	9.071e-009	0.005629	2.536e-006	0.0002247	1.393e-008
64	160	0.0006974	2.98e-008	0.007863	2.453e-006	0.0008665	4.975e-008
128	226	0.001355	6.377e-008	0.009784	3.616e-006	0.001597	6.824e-008
256	320	0.001967	6.692e-008	0.01113	4.231e-006	0.002207	9.787e-008

Tabela 3.9: Griewank: Resultados para o valor final da função

N	n_p	Método Com Modulação		Método Original		Método Com Constrição	
		média	σ	média	σ	média	σ
2	28	1.351e-012	4.339e-023	3.751e-005	7.632e-009	5.046e-021	1.06e-039
4	40	3.456e-011	7.693e-020	8.237e-005	1.068e-008	1.458e-020	8.775e-039
8	57	7.39e-011	4.369e-019	0.0002093	3.084e-008	7.89e-021	1.315e-039
16	80	1.916e-011	7.912e-021	0.000392	1.205e-007	2.853e-015	5.093e-029
32	113	3.647e-011	6.919e-021	0.0004717	5.115e-008	3.161e-013	2.369e-024
64	160	2.995e-009	8.546e-016	0.0007262	1.54e-007	7.234e-013	4.719e-024
128	226	1.261e-010	5.493e-020	0.0009465	1.601e-007	8.233e-013	1.607e-024
256	320	5.878e-010	5.956e-018	0.001207	1.182e-007	8.172e-013	3.401e-024

Tabela 3.10: Griewank: Resultados para a diversidade final da população

3.5.4 Algoritmos Genéticos $\times$ Otimização por Enxame de Partículas

Uma outra análise aqui pertinente, é a comparação entre a aplicação de algoritmos genéticos e otimização por enxame de partículas ao mesmo problema. Essa comparação é feita com base nas quatro funções de teste utilizadas até então para avaliar a eficiência desses algoritmos separadamente. O processo de análise também é implementado de forma semelhante àquele realizado anteriormente, onde são analisados os valores médios de 100 repetições.

O algoritmo genético implementado nessa análise faz uso do operador de reprodução matricial, também proposto neste documento, por este ter demonstrado os melhores resultados dentre as variações estudadas para essa classe de algoritmos. A seguir, é descrita a configuração do algoritmo genético utilizada na presente análise:

- Número de gerações: $n_g = 80$;
- Mecanismo de seleção: Roleta / Elitista;
- Operador de reprodução: Matricial;
- Operador de mutação: Vetorial;

O algoritmo de otimização por enxame de partículas escolhido é aquele que utiliza a modulação de velocidade proposta neste documento, por apresentar resultados superiores as outras variações do algoritmo PSO analisadas. Dessa forma, pretende-se comparar os melhores resultados expressos tanto pela abordagem evolutiva quanto pela utilização de inteligência de enxame para a solução dos

problemas de otimização em questão. A seguir, é descrito o conjunto de parâmetros escolhido para o algoritmo de otimização por enxame de partículas utilizado nesta análise.

- Número de gerações: $n_i = 80$;
- $c_1 = c_2 = 1$;

A primeira função a ser analisada é a função de Rosenbrock, cujos resultados dos vários processos de otimização considerados podem ser observados nas Tabelas 3.11 e 3.12, através dos valores médios obtidos ao final do processo iterativo para o valor da função e para a diversidade da população, respectivamente.

N	n_p	Otimização por enxame de partículas		Otimização por algoritmo genético	
		média	σ	média	σ
2	28	2.693e-007	4.301e-012	0.0003002	4.886e-007
4	40	0.0001601	2.908e-007	0.01966	0.0007183
8	57	0.02593	0.05113	0.03321	0.0005512
16	80	0.02209	0.0004531	0.5222	0.2332
32	113	1.102	0.4391	5.76	11.96
64	160	9.218	7.269	27.94	71.73
128	226	37.02	53.28	133.8	656.6
256	320	110.8	225.7	529.1	5846

Tabela 3.11: Valor final da função de Rosenbrock: Resultados comparativos PSO $\times$ AG

N	n_p	Otimização por enxame de partículas		Otimização por algoritmo genético	
		média	σ	média	σ
2	28	5.582e-007	6.229e-013	4.575	0.5897
4	40	1.986e-006	7.051e-012	3.552	0.00978
8	57	0.0007879	1.18e-006	39.16	57.07
16	80	1.872e-005	2.821e-010	176	199.1
32	113	0.001558	3.928e-006	517.5	6.944e+004
64	160	0.0772	0.01028	414.2	6706
128	226	0.05137	3.001e-005	7308	1.768e+007
256	320	0.1866	0.01522	2.599e+004	3.574e+006

Tabela 3.12: Diversidade final da população, Rosenbrock: Resultados comparativos PSO $\times$ AG

É evidente a superioridade da otimização por enxame de partículas quando são analisados os valores finais da função. Porém, a diversidade da população ao término do processo iterativo é muito maior quando é utilizado o algoritmo genético. Desse resultado conclui-se que a utilização da algoritmo de otimização por enxame de partículas é preferível com relação ao problema em questão, devido à evidente superioridade em taxa de convergência por ele obtido.

A mesma análise, agora utilizando o problema de otimização da função de Rastrigin, leva aos resultados expressos na Tabelas 3.13 e 3.14, que novamente ilustram os valores médios do valor da função e da diversidade da população, após a ultima iteração do processo de otimização.

N	n_p	Otimização por enxame de partículas		Otimização por algoritmo genético	
		média	σ	média	σ
2	28	0.00995	0.009899	2.417e-008	2.539e-014
4	40	0.0597	0.0764	0.0199	0.0196
8	57	0.2587	0.6124	0.01	0.009899
16	80	0.1217	0.2458	0.01233	0.00012
32	113	0.2919	0.04237	0.6021	0.1154
64	160	4.029	1.928	9.086	8.682
128	226	17.89	9.747	63.39	145.4
256	320	50.47	47.58	240.1	1250

Tabela 3.13: Valor final da função de Rastrigin: Resultados comparativos PSO $\times$ AG

N	n_p	Otimização por enxame de partículas		Otimização por algoritmo genético	
		média	σ	média	σ
2	28	0.002852	0.0005134	0.5813	0.4982
4	40	0.04506	0.1783	1.619	1.219
8	57	0.06192	0.3821	6.168	15.85
16	80	5.999e-005	1.279e-007	22.7	206.8
32	113	0.006096	0.002758	85.13	2136
64	160	0.002276	1.411e-005	314.1	2.694e+004
128	226	0.02917	0.008856	1338	5.745e+005
256	320	0.08079	0.02951	4389	4.862e+006

Tabela 3.14: Diversidade final da população, Rastrigin: Resultados comparativos PSO $\times$ AG

O grande número de mínimos locais da função de Rastrigin torna o problema de se otimizar essa função bastante complexo, e na análise em questão, favorece a alta diversidade da população, observada com a utilização do algoritmo genético (Tabela 3.14). Em algumas configurações, são obtidos resultados superiores através do algoritmo genético, contrastando com os resultados obtidos anteriormente. Conclusões mais objetivas com relação a diferenças nos resultados da Tabela 3.13 são obtidas com base no resultado do testes de hipóteses mostrado na Tabela 3.15.

N	n_p	PSO	AG	Erro Tipo I %
2	28	0.00995	2.417e-008	31
4	40	0.0597	0.0199	20
8	57	0.2587	0.01	0
16	80	0.1217	0.01233	2
32	113	0.2919	0.6021	0
64	160	4.029	9.086	0
128	226	17.89	63.39	0
256	320	50.47	240.1	0

Tabela 3.15: Resultado do teste de hipóteses - valor final da função de Rastrigin

Esse teste de hipóteses mostra que, para problemas contendo 2 ou 4 variáveis, existe uma indefinição considerável em se afirmar uma possível diferença entre as médias populacionais para os resultados obtidos com base nestes dois algoritmos. Contudo, para problemas com 16 e 32 variáveis, o algoritmo genético apresenta resultados claramente superiores, situação que se inverte ao serem

considerados problemas de ordem mais elevada, onde o algoritmo de otimização por enxame de partículas passa a apresentar uma eficiência maior.

A indefinição nos resultados anteriores fortalece a necessidade de se avaliar outras funções de teste, como é feito com base no problema de otimização da função esfera. Nessa análise, são obtidos resultados semelhantes àqueles observados para a função de Rosenbrock, como pode ser verificado nas Tabelas 3.16 e 3.17:

N	n_p	Otimização por enxame de partículas		Otimização por algoritmo genético	
		média	σ	média	σ
2	28	7.394e-010	1.074e-018	2.533e-010	1.525e-018
4	40	5.697e-009	2.968e-017	3.982e-009	3.101e-016
8	57	4.702e-008	1.452e-015	5.525e-007	2.723e-012
16	80	5.904e-006	5.204e-011	6.726e-005	4.398e-009
32	113	0.001216	5.361e-007	0.002725	1.287e-006
64	160	0.01791	2.836e-005	0.04326	0.0001734
128	226	0.0828	0.0002001	0.295	0.002312
256	320	0.251	0.0009997	1.252	0.03563

Tabela 3.16: Valor final da função Esfera: Resultados comparativos PSO $\times$ AG

N	n_p	Otimização por enxame de partículas		Otimização por algoritmo genético	
		média	σ	média	σ
2	28	8.981e-011	6.049e-019	1.541e-005	3.441e-010
4	40	1.746e-011	4.901e-021	5.128e-005	1.959e-009
8	57	3.007e-009	7.293e-016	0.0001684	1.374e-008
16	80	1.154e-009	3.315e-017	0.0006332	3.123e-007
32	113	8.556e-009	5.142e-016	0.002122	1.589e-006
64	160	3.665e-007	7.674e-012	0.01025	4.862e-005
128	226	3.902e-007	5.498e-013	0.03777	0.0003048
256	320	1.594e-006	4.263e-012	0.1534	0.006845

Tabela 3.17: Diversidade final da população, Esfera: Resultados comparativos PSO $\times$ AG

Nessas tabelas, torna-se evidente a superioridade dos resultados obtidos com a utilização do algoritmo PSO para problemas de ordem mais elevada, assim como foi observado para a função de

Rosenbrock. Porém, nesta análise, são observados resultados superiores com a aplicação do algoritmo genético para problemas com um número pequeno de variáveis (2 e 4).

Uma ultima análise é realizada para o problema de otimização da função de Griewank, que apesar de possuir vários mínimo locais, assim como acontece com a função de Rastrigin, não apresenta resultados superiores com a utilização do algoritmo genético, sendo a otimização por enxame de partículas o método que claramente obteve os melhores resultados para problemas com maior número de variáveis (acima de 4), como pode ser observado nas Tabelas 3.18 e 3.19.

N	n_p	Otimização por enxame de partículas		Otimização por algoritmo genético	
		média	σ	média	σ
2	28	2.748e-010	1.433e-019	1.585e-011	8.189e-021
4	40	1.165e-009	1.635e-018	5.59e-009	5.298e-016
8	57	8.841e-009	6.333e-017	1.867e-006	9.448e-012
16	80	3.837e-006	2.334e-011	8.704e-005	5.338e-009
32	113	0.0001534	5.842e-009	0.0007398	6.989e-008
64	160	0.0006593	2.759e-008	0.002883	5.761e-007
128	226	0.001338	4.447e-008	0.006087	1.299e-006
256	320	0.001958	7.238e-008	0.01041	3.281e-006

Tabela 3.18: Valor final da função de Griewank: Resultados comparativos PSO $\times$ AG

		Otimização por enxame de partículas		Otimização por algoritmo genético	
N	n_p	média	σ	média	σ
2	28	2.727e-012	4.428e-022	1.709e-006	3.955e-012
4	40	4.682e-011	2.014e-019	3.683e-006	3.703e-011
8	57	4.063e-012	3.528e-022	5.52e-006	1.962e-011
16	80	4.598e-011	1.578e-019	8.288e-006	4.356e-011
32	113	3.092e-011	1.109e-020	1.057e-005	5.464e-011
64	160	4.028e-010	1.01e-017	1.414e-005	7.531e-011
128	226	3.013e-010	2.582e-018	1.712e-005	5.92e-011
256	320	4.741e-010	5.333e-018	2.282e-005	1.339e-010

Tabela 3.19: Diversidade final da população de Griewank: Resultados comparativos PSO $\times$ AG

3.6 Conclusões

A eficiência do método de otimização por enxame de partículas fica claramente comprovada através dos exemplos comparativos expostos, onde os algoritmos baseados nesta meta-heurística mostraram-se comparáveis e, em várias situações, superiores ao algoritmo genético utilizado nas comparações. Superioridade essa que pode ser notada principalmente em problemas com maior número de variáveis, onde o algoritmo PSO com modulação de velocidade apresenta uma maior eficiência com relação a taxa de convergência e preservação da diversidade.

O método de modulação da velocidade, proposto neste documento, representa uma alternativa para o aumento de performance do algoritmo PSO, dentre outras encontradas na literatura, sendo que sua aplicação torna o método consideravelmente mais eficiente. Este ganho de eficiência mostrou-se comparável ao ganho atingido por outras modificações, onde cita-se aqui o método do fator de constrição, que representa uma maneira simples e eficaz de aumentar a taxa de convergência do algoritmo original de otimização por enxame de partículas, a um custo computacional menor do que àquele necessário para implementação da modulação de velocidade, tendo em vista que essa requer o cálculo do ângulo θ, que por sua vês utiliza o cálculo de normas de vetores.

A modulação feita na velocidade das partículas é relativa à sua movimentação e não inibe o aumento da velocidade das partículas, quando estas apresentam velocidades consecutivas colineares e em uma mesma direção. Essa abordagem, além de trazer um significativo aumento na taxa de convergência do processo de busca, preserva a diversidade da população, o que é comprovado pelos exemplos propostos. Dessa forma, o método proposto apesar de computacionalmente mais oneroso que o método com fator de constrição, é bastante superior à esse em preservação da diversidade. Essas características tornam a proposta de modulação de velocidade uma alternativa de grande relevância quando se deseja aumentar a eficiência do método de otimização por enxame de partículas.

Capítulo 4

Redes Neurais Artificiais

4.1 Introdução

Redes neurais artificiais são sistemas de processamento paralelo e distribuído, constituídos por unidades simples de processamento interconectadas entre si. A área de redes neurais, também conhecida por computação conexicionista, é caracterizada por uma computação não algorítmica[1] que em algum nível lembra a estrutura de um cérebro biológico. Por não ser baseada em regras ou algoritmos, a computação neural constitui uma alternativa à computação algorítmica convencional, apresentando soluções para problemas de difícil tratamento quando utilizado outro paradigma de computação.

O final da década de 80, marca o ressurgimento dessa área após um período de tempo onde essa teoria ficou praticamente esquecida. Um breve histórico da evolução da computação neural é discutido na sessão seguinte.

4.1.1 Histórico

O primeiro trabalho realizado na área de redes neurais artificiais é fruto da pesquisa pioneira de Warren McCulloch e Walter Pitts, que apresentaram no ano de 1943 seu artigo: *"A logical calculus of the ideas immanent in nervous activity"* [34]. Nele, os autores elaboram uma discussão sofisticada a respeito de redes lógicas e apresentam novas idéias sobre máquinas de estado finito, elementos de decisão de limiar e representações lógicas de várias formas de comportamento e memória associada a processos biológicos de processamento da informação.

O trabalho de McCulloch e Pitts se concentra muito mais na descrição de modelos artificiais do neurônio e na representação computacional das suas capacidades do que na aplicação prática de redes

[1] Neste documento quando são discutidas meta-heurísticas como Redes Neurais Artificiais, se está interessado em sua estrutura e funcionamento, e não em sua implementação, o que justifica o uso de termos como: *"não algorítmica"* entre outros.

neurais artificiais. Somente alguns anos depois os autores se preocuparam com o aprendizado da rede neural, que teve sua primeira contribuição significativa em 1949 com o trabalho de Donald Hebb: *"The Organization of Behavior a Neuropsychological Theory"* [35] onde o autor apresenta a regra de Hebb para o aprendizado. Hebb demonstra em seu trabalho que a especialização de uma rede neural pode ser obtida através do ajuste dos pesos atribuídos a ligação existente entre os neurônios, tendo se baseado em uma teoria que tenta explicar o aprendizado de redes neurais biológicas através do reforço das ligações sinápticas. Posteriormente, Widrow e Hoff sugerem a regra delta, baseada no método do gradiente para a minimização do erro da rede.

No ano de 1958, Frank Rosenblatt apresenta em seu trabalho: *"The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain"* um novo modelo de neurônio artificial, o perceptron [36], e demonstra como é possível unir vários perceptrons formando uma rede e como é possível realizar o aprendizado dessa rede de forma que a mesma funcione como um classificador, com capacidade de identificar padrões linearmente separáveis. Mais tarde, em 1969 Minsky e Papert [37] chamam a atenção para uma série de tarefas que o perceptron não era capaz de cumprir, dado que este aplica-se apenas a problemas linearmente separáveis, o que representa um subconjunto muito pequeno dentro das possíveis aplicações das redes neurais.

Minsky e Papert utilizaram vários argumentos para questionar a viabilidade do perceptron, como a incapacidade deste em resolver problemas simples como a detecção de paridade e representação da função do *ou-exclusivo*, bem como a inexistência de uma forma de treinamento para redes com mais de uma camada. Tarefas essas que são facilmente implementadas utilizando portas lógicas, que surgiam na época como a base da computação digital, paradigma distinto da computação neural. Essas críticas foram em parte responsáveis pelo abandono parcial dos estudos relativos a teoria conexionista na década de 70, período no qual apenas alguns poucos pesquisadores continuaram trabalhando na área.

O ressurgimento da teoria conexionista se dá gradualmente na década de 80, iniciando-se pelo trabalho de Hopfield [38], que chama a atenção para as propriedades associativas das redes neurais e mostra a relação entre redes recorrentes e sistemas físicos. Alguns anos mais tarde, David E. Rumelhart, Geoffrey E. Hinton e Ronald J. Williams mostram em [39] que as conjeturas de Minsky e Papert eram equivocadas e além de apresentar um algoritmo para o treinamento de redes neurais com mais de uma camada, demonstram que essas redes são capazes de *aprender* problemas não linearmente separáveis. Desde então, ouve uma renovação nos interesses pela teoria conexionista, impulsionada ainda mais pelo avanço tecnológico que possibilita a realização de redes neurais artificiais, de um modo que antes não parecia ser possível. A Tabela 4.1 ilustra resumidamente o histórico das pesquisas com redes neurais artificiais.

1943	McCulloch e Pitts	Primeiro trabalho sobre representação artificial de sistemas neurais
1949	Hebb	Primeiro trabalho voltado para o aprendizado de redes neurais
1957	Rosenblatt	Apresentação do modelo perceptron
1958	Widrow e Hoff	Regra de aprendizagem baseada no gradiente do erro
—	—	—
1969	Minsky e Papert	Conjeturas equivocadas influenciam no abandono das pesquisas com redes neurais artificiais
—	—	—
1960-1980	Kohonen, Grossberg, Widrow, Anderson, Caianiello, Fukushima, Aleksander	Memórias associativas e auto-organizáveis, redes sem pesos, sistemas auto-adaptativos, cognitron e neocognitron,
—	—	—
1982	Hopfield	Relação entre redes recorrentes e sistemas físicos
1986	Rumelhart e McClelland	Backpropagation

Tabela 4.1: Histórico da pesquisa com redes neurais artificiais

4.1.2 Base Biológica

A principal inspiração para as redes neurais artificiais, é sem dúvida o cérebro humano, que apresenta uma vasta potencialidade almejada por sistemas computacionais. Infelizmente, não se tem uma descrição muito detalhada do funcionamento do cérebro, apenas uma noção teórica do processamento por ele realizado. Por outro lado, sabe-se que a estrutura física básica desse computador biológico é o neurônio, sobre o qual são conhecidos vários aspectos da sua fisiologia e funcionamento.

O neurônio é a célula do sistema nervoso responsável pela produção de impulsos nervosos, que são transmitidos por todo o organismo através de uma vasta rede neuronial. Estima-se que, na espécie humana, devam existir cerca de 10^{11} neurônios distribuídos pelo sistema nervoso. A Figura 4.1 traz uma ilustração do neurônio encontrado no tecido nervoso com suas partes componentes.

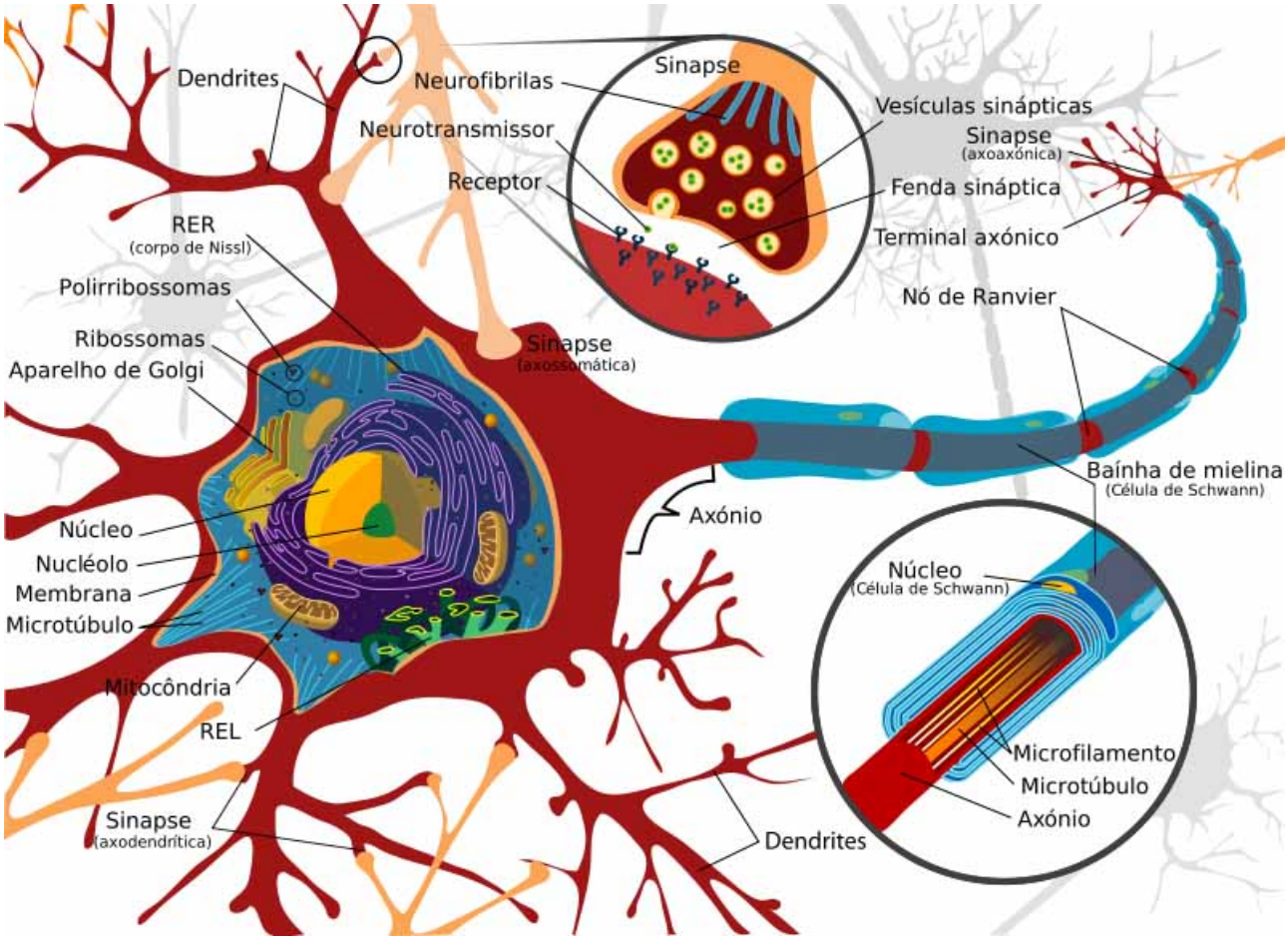

Figura 4.1: Neurônio

A seguir são apresentadas definições sucintas das principais partes constituintes do neurônio.

- Membrana celular é a estrutura que delimita o neurônio, separando o espaço intra-celular do meio extra-celular.
- Citoplasma é espaço intra-celular existente entre a membrana e o envoltório do núcleo.
- Núcleo é a estrutura do neurônio que contém seu material genético (DNA). Ele está separado do restante da célula por um envoltório e comunica-se com o citoplasma por meio de poros.
- Ribossomos são estruturas responsáveis por gerar proteínas a partir de moléculas de RNA.
- Mitocôndria é uma das mais importantes organelas celulares, sendo responsável por processar e converter energia na forma de ATP (Trifosfato de adenosina). Ela precisa ser abastecida pela célula que a hospeda com substâncias orgânicas como oxigênio e glicose.

- Soma é o corpo celular a menos dos dendritos e axônio.
- Dendritos ou dendrites são prolongações da membrana celular do neurônio responsáveis por receber sinais elétricos oriundos de outros neurônios ou do ambiente.
- Axônios são prolongações mais extensas do neurônio, com a finalidade de transportar sinais elétricos para outros neurônios mais distantes, ou para outros tecidos, como o tecido muscular.
- Bainha de mielina, presente somente nos animais vertebrados, tem a função de isolação do sinal elétrico, facilitando a seu transporte a longas distâncias.
- Sinapses nervosas são os pontos onde as extremidades do axônio de um neurônio se encontram com o dendrito de outro neurônio e ocorre a transmissão do estímulo elétrico por meio de mediadores químicos.
- Neurotransmissores são substâncias químicas produzidas pelos neurônios para a transmissão do impulso elétrico nas sinapses.

Existem basicamente três tipos distintos de neurônios, os sensoriais, com dendritos mais longos e axônios curtos, contrastando com os neurônios motores, que possuem axônios longos e são responsáveis pela comunicação com o tecido nervoso. Exitem ainda os neurônios conectores ou interneurônios, que realizam a comunicação inter neuronial e compõem o sistema nervoso central. Em seres humanos aproximadamente 70% dos neurônios do sistema nervoso central estão localizados no córtex cerebral, que é a camada mais externa do cerebro onde são realizadas as operações computacionalemte mais complexas como memória, atenção, consciência, linguagem, percepção e pensamento.

O processamento da informação em cada neurônio é feito através dos impulsos elétricos trocados entre neurônios. Esses impulsos podem ser inibitórios ou excitatórios, e atravessam a membrana celular na região sináptica, alterando a diferença de potencial elétrico intra-celular e extra-celular. Essa diferença de potencial que se forma na membrana do neurônio é quem determina se o mesmo irá ou não ser ativado, repassando o impulso elétrico para outros neurônios.

O cérebro humano é responsável pelo processamento das emoções, pensamentos, percepção e cognição, realizando também funções motoras, sensoriais e autônomas. Todas essas funcionalidades são implementadas por meio de redes neurais. O meio pelo qual toda essa variedade de funções é implementada ainda não é completamente compreendido, sendo as redes neurais artificias dotadas apenas de um reduzido número de características básicas conhecidas da dinâmica do cérebro biológico.

Apesar dessa pouca similaridade existente entre as redes neurais artificiais e o cérebro, as características em comum entre eles são capazes de reproduzir vários fenômenos de grande relevância

computacional, levando a resultados que seriam muito difíceis de se atingir através de outros paradigmas de computação. Esses resultados mostram que as redes neurais artificiais, além de representarem um paradigma de computação paralelo e distribuído, caracterizam uma ferramenta para auxiliar na compreensão do processamento biológico da informação realizado pelo cérebro.

4.1.3 Processamento Neural da Informação

Dentre as funcionalidades observadas no processamento biológico da informação, os principais aspectos, da ótica da aplicação computacional, são descritos a seguir [40]:

- **Robustez e tolerância a erros.** Como já foi dito anteriormente, o cérebro possui um número muito grande de neurônios, sendo que milhares deles morrem todos os dias. Porém, o cérebro continua funcionando por muitos anos sem apresentar perdas de funcionalidade devido à falta desses neurônios.

- **Flexibilidade.** Uma mudança de ambiente não acarreta necessidade de reprogramação do cérebro, que baseado em padrões armazenados em experiências passadas, na interação com o novo ambiente, e na capacidade de generalização, se auto-adapta às novas condições, ou seja, aprende a se comportar no novo ambiente.

- **Capacidade de lidar com informações nebulosas, inconsistentes, ruidosas e probabilísticas**. Informações ruidosas e inconsistentes são passíveis de tratamento via computação digital, porém, esse processamento é bastante custoso e complexo, o que requer um alto nível de sofisticação em sua programação e uma análise detalhada dos dados. Por outro lado, computadores digitais não possuem a capacidade de tratar incertezas, capacidade essa inerente a computadores neurais, que processam com naturalidade todas essas classes de informações.

- **Paralelismo.** Computadores digitais são essencialmente seqüenciais, enquanto o cérebro apresenta um paralelismo intrínseco, além de apresentar componentes com tamanho físico muito reduzido e baixo consumo de energia, quando comparados a computadores digitais.

A emulação de sistemas neurais de processamento através de computadores digitais é limitada a constrições impostas por esse paradigma, como por exemplo a representação numérica binária e o processamento exclusivamente seqüencial. Apesar dessas limitações, são preservadas muitas das características do processamento neural nos modelos de redes neurais artificiais conhecidos atualmente, que mesmo presos a essas constrições, implementam funcionalidades não observadas em outros sistemas artificiais.

4.2 Redes Neurais Artificiais

A exemplo do que acontece em uma rede neural biológica, o componente básico de uma rede neural artificial é o neurônio. O modelo mais clássico de neurônio artificial, proposto por McCulloch e Pitts [34], é uma representação do que se sabia até então sobre o funcionamento de uma célula nervosa do cérebro humano.

4.2.1 Modelo MCP (McCulloch and Pitts Perceptron)

Este modelo é representado por uma entidade matemática que possui n terminais de entrada representando os dendritos e apenas um terminal de saída representando o axônio. A saída do neurônio possui apenas dois estados distintos, ativo e inativo. As entradas representadas por $x_1, x_2, x_3, ..., x_n$, são pré-multiplicadas por escalares $w_1, w_2, w_3, ..., w_n$ constantes, que representam os pesos sinápticos associados às entradas ou sinapses do neurônio.

Na descrição do modelo original do neurônio MCP. o neurônio dispara quando a soma dos sinais $w_i x_i$ atinge um determinado limiar, equação (4.1), de modo similar ao observado em neurônios biológicos que disparam quando o saldo da soma dos impulsos elétricos recebidos por ele de outros neurônios faz com que o potencial elétrico da membrana atinja um determinado limiar. Um impulso elétrico recebido pelo neurônio pode estimulá-lo a disparar, no caso de sinais excitatórios, ou inibir sua ação, sinais inibitórios. Artificialmente essas propriedades são emuladas pelos pesos w_i, onde valores positivos e negativos de w_i correspondem a sinais excitatórios e inibitórios respectivamente.

$$y = \begin{cases} 1 & se \quad \sum_{i=1}^{n} x_i w_i \geq \theta \\ 0 & se \quad \sum_{i=1}^{n} x_i w_i < \theta \end{cases} \tag{4.1}$$

onde $y = 1$ representa o estado ativo do neurônio e $y = 0$ o estado contrário. O limiar de ativação do neurônio é dado por θ e o seu número de entradas é igual a n.

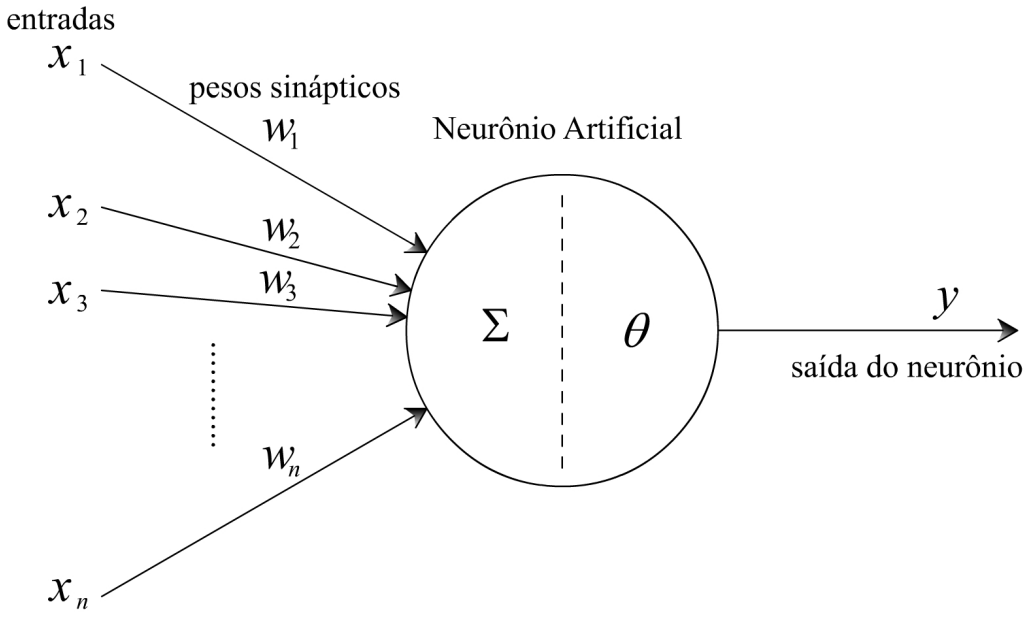

Figura 4.2: Neurônio MCP proposto por McCulloch e Pitts

Esse modelo foi baseado em algumas simplificações necessárias para sua viabilização. Uma dessas simplificações é quanto à sincronização dos disparos em redes artificiais. Tal característica não é existe em redes neurais biológicas, onde não é observado nenhum mecanismo de sincronização, tampouco de temporização, que calcule o momento exato em que um neurônio da rede deva disparar ou não. O disparo de um neurônio biológico requer o uso de muitos neurotransmissores, que estarão indisponíveis instantes após a sinapse, logo, o nível de ativação observado em um disparo consecutivo será atenuado, sendo assim, sabe-se que o valor da próxima saída de um neurônio, depende diretamente das ativações anteriores, o que não é verificado no modelo MCP.

4.2.2 Neurônio *Integrate-and-Fire*

Outro modelo clássico de neurônio é o *Integrate-and-Fire*, representado por um modelo contínuo e não discreto, dado temporalmente por:

$$I(t) = C_m \frac{dVm}{dt} \tag{4.2}$$

Essa equação representa a lei da capacitância. Assim, quando uma corrente é aplicada a membrana do neurônio, a tensão da membrana aumenta até um dado limiar de disparo V_{th}, gerando um impulso e retornando o potencial da membrana para uma tensão de repouso.

4.2.3 Neurônio Genérico em RNA

O elemento computacional básico da maioria das redes neurais artificiais utilizadas atualmente é o neurônio integrador, que realiza a soma ponderada dos seus sinais de entrada e aplica a esses uma função de ativação. Este neurônio é composto basicamente por:

- Sinapses, caracterizadas pelos pesos w
- Junção somadora
- Função de ativação

Uma descrição mais formal desse neurônio genérico é dada a seguir:

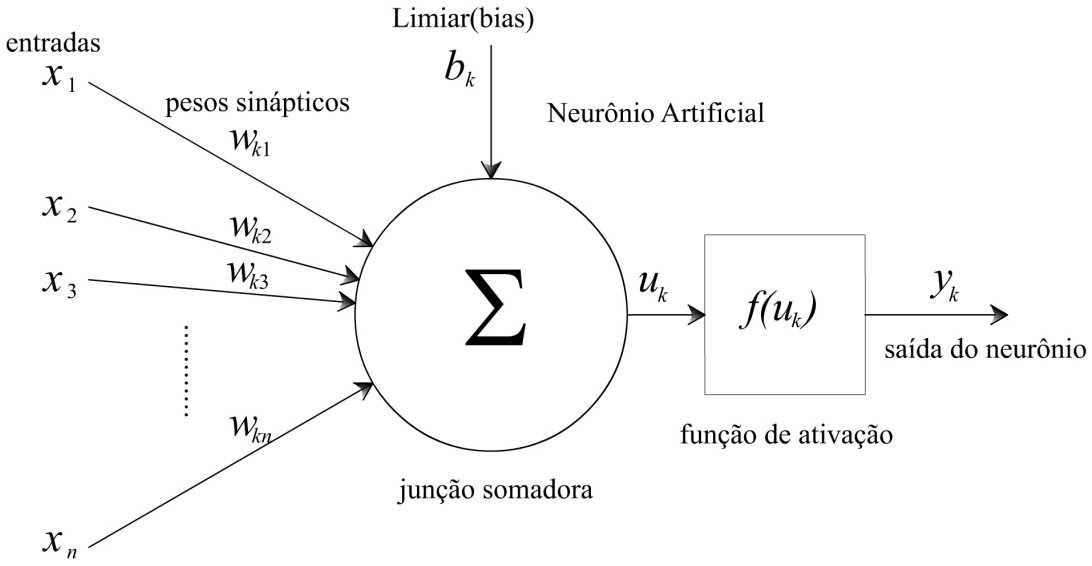

Figura 4.3: Neurônio Genérico

$$
\begin{aligned}
u_k &= \sum_{i=1}^{n}(w_{ki}x_i) + b_k \\
y_k &= f(u_k)
\end{aligned}
$$

Nesta representação, w_{ki} corresponde ao peso da sinapse entre o neurônio pós-sináptico k e o neurônio pré-sináptico i. O limiar b_k é introduzido com a função de controlar a influência da soma das entradas na ativação do neurônio. A título de exemplo, consideremos o neurônio MCP, onde a ativação é dada por um limiar θ; se for utilizado um termo de polarização b neste neurônio, pode-se

considerar $\theta = 0$ e definir $b = -\theta$ obtendo assim o mesmo resultado. A saída do neurônio genérico é matematicamente descrita na equação (4.3)

$$y_k = f\left(\sum_{i=1}^{n}(w_{ki}x_i) + b_k\right) \tag{4.3}$$

A função de ativação $f(.)$ tem o objetivo de limitar a saída do neurônio, dado que a soma das entradas não é explicitamente limitada, bem como introduzir não-linearidades ao modelo do neurônio. Essa não-linearidade é um requisito essencial para a concepção de redes neurais com mais de uma camada. Para exemplificar essa afirmação, consideremos uma rede neural contendo duas camadas A e B, com m e n neurônios respectivamente. Cada camada contém seu próprio conjunto de parâmetros, $A : (W_A, B_A)$ e $B : (W_B, B_B)$. A entrada dessa rede é dada pelo vetor X que é aplicado à camada A, enquanto a entrada da camada B é a saída da camada A. Nesta rede neural, a saída da camada A é dada por:

$$Y^A = XW^A + B^A \tag{4.4}$$

onde a notação matricial é utilizada a fim de minimizar a representação.

A saída da camada B será então dada por:

$$\begin{aligned} Y^B &= Y^A W^B + B^B \\ &= (XW^A + B^A)W^B + B^B \\ &= XW^A W^B + B^A W^B + B^B \end{aligned} \tag{4.5}$$

que representa o mesmo resultado obtido quando utilizada uma rede neural de uma única camada contendo os parâmetros $C : (W^C, B^C)$ onde: $W^C = W^A W^B$ e $B^C = B^A W^B + B_B$. Assim redes neurais de múltiplas camadas utilizando funções de ativação lineares são equivalentes a redes de uma única camada, e como tais, são incapazes de resolver uma vasta gama de problemas. A fim de inserir a não-linearidade necessária, além de prover o comportamento esperado do neurônio, vários tipos de funções de ativação são utilizados, relativamente a sua aplicação.

4.2.4 Funções de Ativação

O modelo proposto por McCulloch e Pitts utilizava uma função de ativação não linear e descontínua, dada pela Figura 4.4.

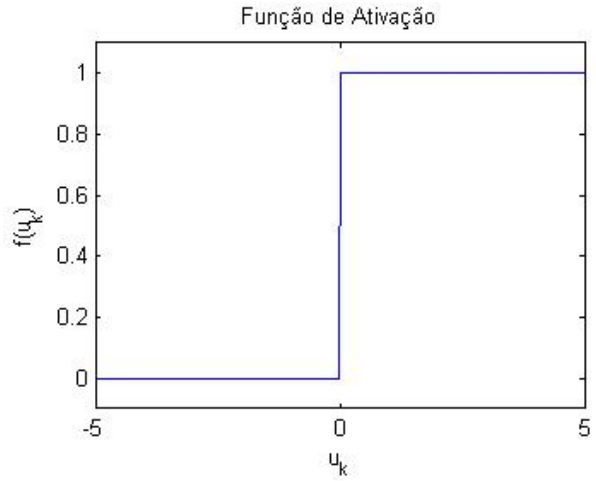

Figura 4.4: Função de ativação lógica

Essa função de ativação apresenta uma descontinuidade, a qual representa um empecilho à utilização de métodos de treinamento baseados no gradiente do erro, pois, o ponto de descontinuidade da função não possui derivadas. Outra função de ativação bastante comum é a função linear, dada por:

$$\begin{aligned} f(u) &= pu \\ \frac{\partial f(u)}{\partial u} &= p \end{aligned}$$

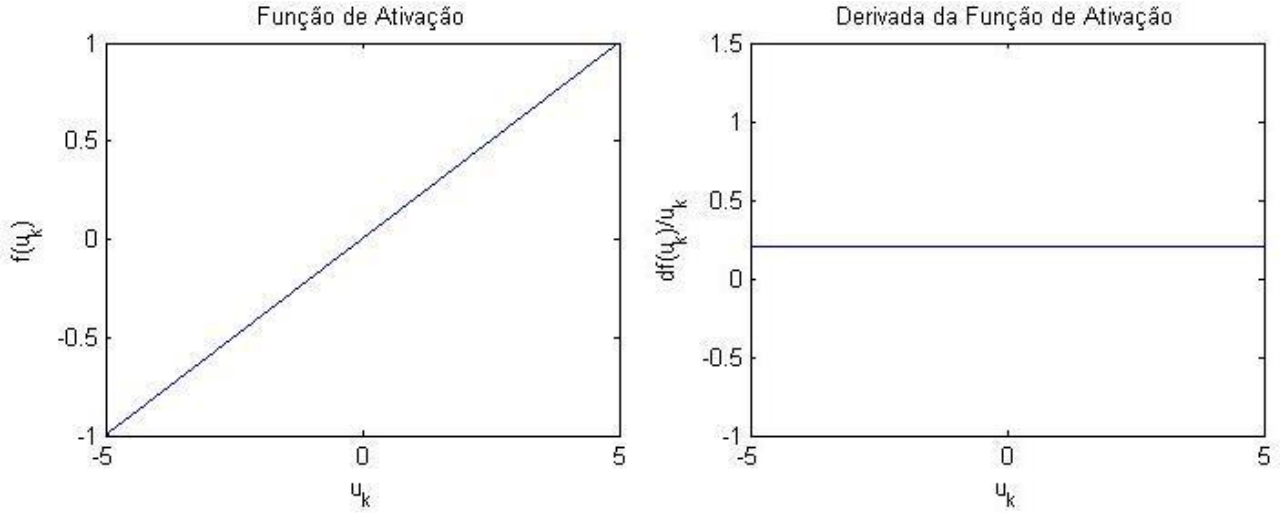

Figura 4.5: Função de ativação Linear, $p = 0,2$

Essa função de ativação, apesar do problema discutido anteriormente, é bastante utilizada na camada de saída de redes neurais com múltiplas camadas. As funções de ativação mais relevantes na aplicação desse tipo de rede são as funções sigmoidais, descritas a seguir:

Função logística:

$$\begin{aligned} f(u) &= \frac{1}{1+e^{-pu}} \\ \frac{\partial f(u)}{\partial u} &= p\frac{e^{-pu}}{(1+e^{-pu})^2} \end{aligned} \tag{4.6}$$

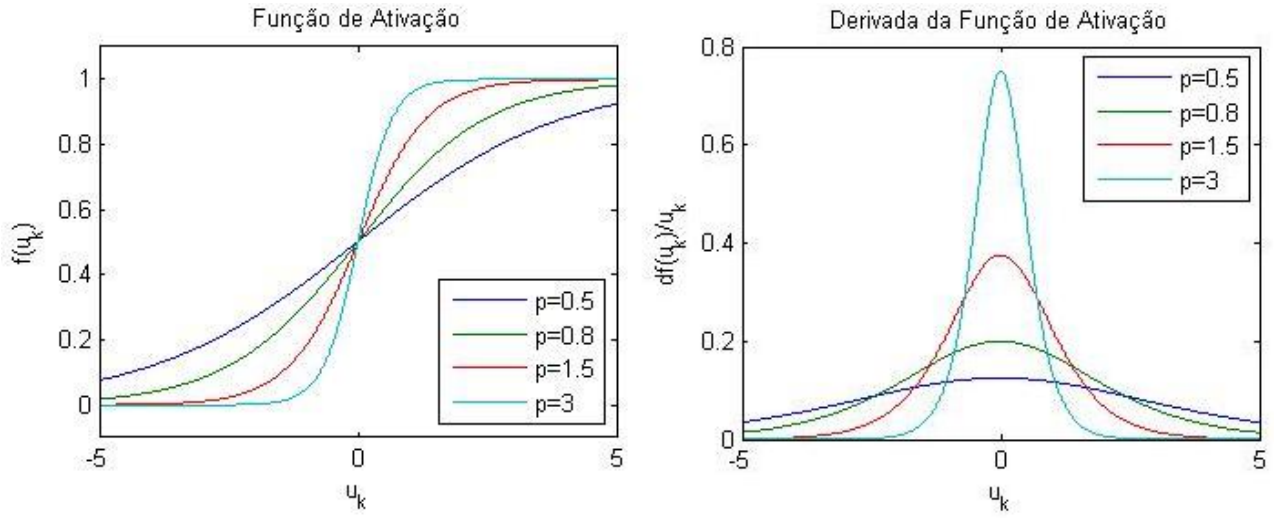

Figura 4.6: Função de ativação logística

Função tangente hiperbólica:

$$\begin{aligned} f(u) &= \tanh(pu) = \frac{e^{pu}-e^{-pu}}{e^{pu}+e^{-pu}} \\ \frac{\partial f(u)}{\partial u} &= p\left(1-\tanh^2(pu)\right) \end{aligned} \tag{4.7}$$

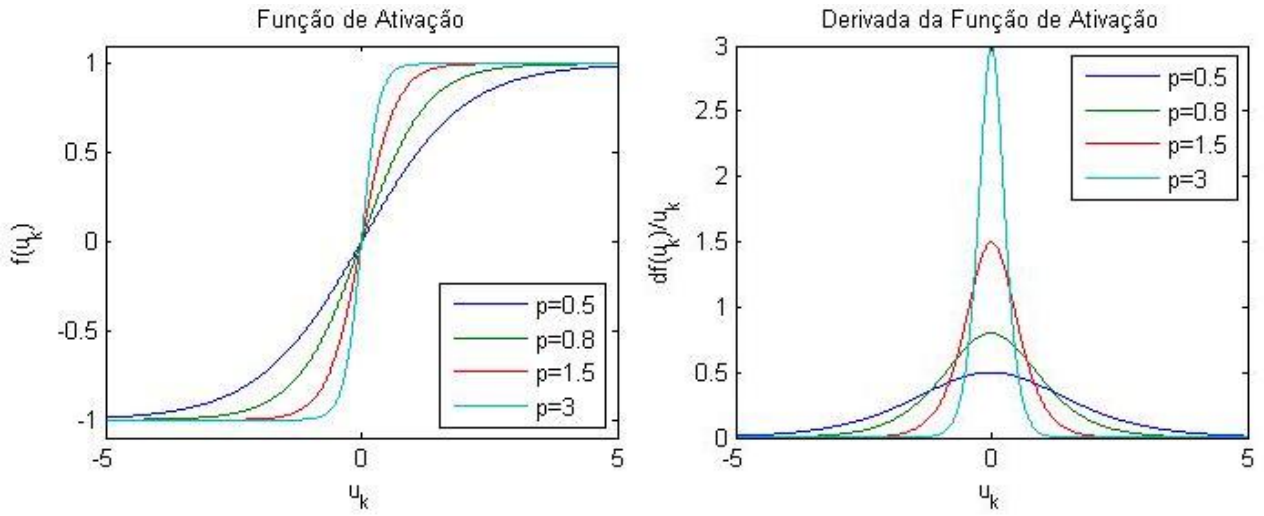

Figura 4.7: Função de ativação tangente hiperbólica

Essas funções de ativação sigmoidais são utilizadas na construção de diversos modelos de redes neurais artificiais nas mais variadas áreas, sendo a função tangente hiperbólica preferível em muitas situações, devido à sua capacidade de apresentar valores de ativação negativos para o neurônio.

4.3 Arquiteturas de Redes Neurais

A estrutura básica de ligação entre os neurônios no cérebro biológico é pouco conhecida. Sabe-se que no córtex cerebral humano por exemplo, cada neurônio está conectado a aproximadamente 10^4 outros neurônios e que o caminho entre quaisquer dois neurônios não conectados, não passa por mais do que duas dessas células. Esse padrão de interconexão ainda não é possível de ser artificialmente emulado, pelo menos não obtendo as mesmas funcionalidades observadas em um sistema biológico. Entretanto, é possível a utilização de arquiteturas padronizadas para o desenvolvimento de uma RNA, onde cada padrão de arquitetura é criado para apresentar um comportamento específico.

Sendo assim, a arquitetura da rede é um aspecto de suma importância em sua concepção, pois este irá definir a sua funcionalidade, e assim, será restringido o problema ao qual a rede pode vir a ser aplicada, como no caso de uma RNA contendo apenas uma única camada, onde só é possível tratar de problemas lineares[2].

A arquitetura da rede é definida por um conjunto de parâmetros onde constam: número de camadas, número de neurônios que geralmente é diferente em cada camada, tipo de conexão entre os

[2]O termo *lineares* aqui empregado, diz respeito a uma vasta gama de problemas, onde pode-se citar: classificação de padrões linearmente separáveis, aproximação linear de funções, predição linear de séries temporais, entre outros.

neurônios e topologia da rede. A seguir são expostas algumas das principais arquiteturas utilizadas atualmente na concepção de uma RNA.

Rede Feedforward com Uma Única Camada

Este é o modelo mais simples de RNA, onde existe uma única camada com capacidade de processamento da informação, além de uma camada sensorial, que apenas propaga o sinal para a camada seguinte.

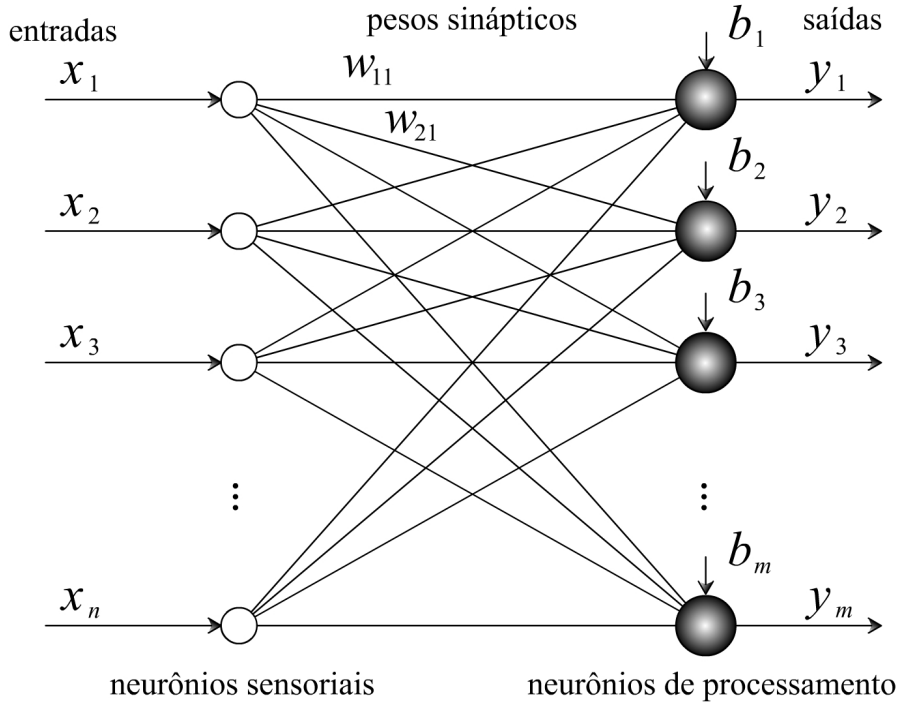

Figura 4.8: Rede feedforward de uma camada

A operação matemática realizada por essa rede pode ser resumida pela equação (4.8):

$$Y = F(WX + B) \tag{4.8}$$

onde $Y \in R^m$ é um vetor contendo as saídas da rede, $W \in R^{m \times n}$ contém os pesos de todas as sinapses da rede, equação (4.9), $B \in R^m$ é o vetor que contém as polarizações, equação (4.10) e $F(Z)$ é um campo vetorial que representa as funções de ativação de cada neurônio da rede, equação (4.11).

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix} \tag{4.9}$$

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{4.10}$$

$$F_i(Z) = f(z_i) \tag{4.11}$$

onde z_i representa a soma das estradas e polarização do *i-ésimo* neurônio da camada de saída e $f(z_i)$ é a função de ativação desse neurônio.

Redes Feedforward de Múltiplas Camadas: MLP (Multi Layer Peceptron)

São uma extensão do modelo anteriormente apresentado, onde é inserida uma ou mais camadas intermediárias entre a camada sensorial e a camada de saída. Essa camada intermediária, ou escondida, é responsável por aumentar a capacidade de processamento da rede, tornando-a apta a tratar não mais apenas problemas lineares, ampliando significativamente a sua gama de aplicações. A Figura 4.9 ilustra uma MLP contendo uma camada intermediária.

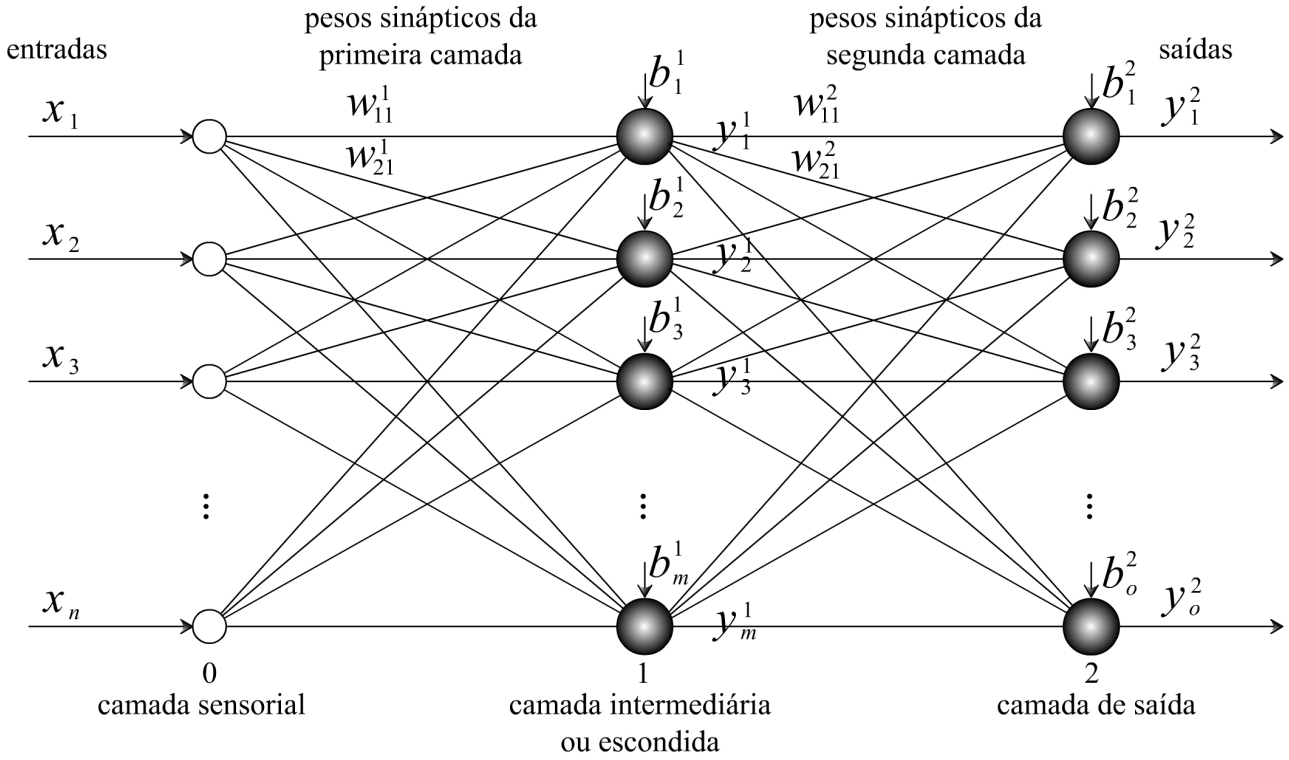

Figura 4.9: Rede MLP com uma camada intermediária

Consideremos agora os seguinte parâmetros: a matriz de pesos W^k contendo os pesos sinápticos entre as camadas k e $k-1$, o vetor de bias da camada k: B^k e a saída Y^{k-1} da camada $k-1$. É possível então expressar a saída da camada k de forma matricial como sendo:

$$\begin{aligned} Y^k &= F^k(U^k + B^k) \\ &= F(Y^{k-1}W^k + B^k) \\ U_k &= Y^{k-1}W^k \end{aligned} \tag{4.12}$$

onde F^k é o campo vetorial formado pelas funções de ativação dos neurônios da camada k e U_k é um vetor contendo as entradas liquidas de cada neurônio da camada k.

Podem ser utilizadas funções de ativação diferentes para cada camada, ou para cada neurônio da rede neural, como é observado na implementação de uma determinada classe de redes neurais artificias, conhecidas como redes construtivas, embora resultados satisfatórios sejam obtidos utilizando a mesma função de ativação para toda a rede.

Assim, a saída da primeira camada intermediária da rede ($k = 1$) pode ser expressa por:

$$Y^1 = F^1(W^1 X + B^1) \tag{4.13}$$

Para a rede apresentada na Figura 4.9, a saída da rede será a saída da camada 2, que é obtida propagando para frente o sinal obtido na saída da primeira camada:

$$Y^2 = F^2(W^2 Y^1 + B^2) \tag{4.14}$$

Dessa forma, se a rede possuir uma terceira camada de processamento, basta propagar esse sinal por mais essa camada, e assim sucessivamente, até ser obtida a saída da última camada da rede. Durante o aprendizado de uma RNA, essa fase onde o sinal é propagado a partir da entrada da rede até a sua saída é conhecido como fase *forward* (para frente).

Esse tipo de arquitetura tem uma grande expressão na área de redes neurais, devido à sua capacidade de aproximar qualquer função contínua que encaixe em um hipercubo unitário, como é evidenciado com base no teorema da aproximação universal 4.3.1. Essa capacidade torna a rede capaz de resolver uma vasta gama de problemas, como a estimação de séries temporais, aproximação de curvas e a classificação de padrões.

Teorema 4.3.1 (Aproximação Universal) *Seja uma função f contínua não-linear e monotonicamente crescente. Seja I_m um hipercubo unitário m-dimensional. O espaço das funções contínuas em I_m é $C(I_m)$. Então dada uma função $g(x) \in C(I_m)$, existe $\epsilon > 0$ tal que:*

$$\left\| g(x) - \sum_{i=1}^{m} \alpha_i f\left(\sum_{j=1}^{n} w_{ij} x_j - x_{0i} \right) \right\| < \epsilon \; \forall \, x \in I_m \tag{4.15}$$

onde m é um inteiro, n é a dimensão de x, e w_{ij} e α_i são escalares [41].

Quando escolhidas funções de ativação sigmoidais para a rede MLP, esta se enquadra claramente nos requisitos do Teorema 4.3.1, representando então um aproximador universal.

Outras Arquiteturas de Redes Neurais

Existem inúmeras outras arquiteturas propostas na literatura para a concepção de redes neurais, desenvolvidas com base em conceitos diferenciados, com a finalidade de resolver problemas específicos. Dentre as arquiteturas mais importantes pode-se citar:

- **Redes Recorrentes**. O exemplo mais clássico de rede recorrente é sem dúvida a rede de Hopfield. Baseada em sistemas dinâmicos, esse tipo de rede neural armazena padrões em uma

superfície de energia na forma de pontos de equilíbrio, com a finalidade de recuperar os padrões originais com base em versões distorcidas desses padrões.

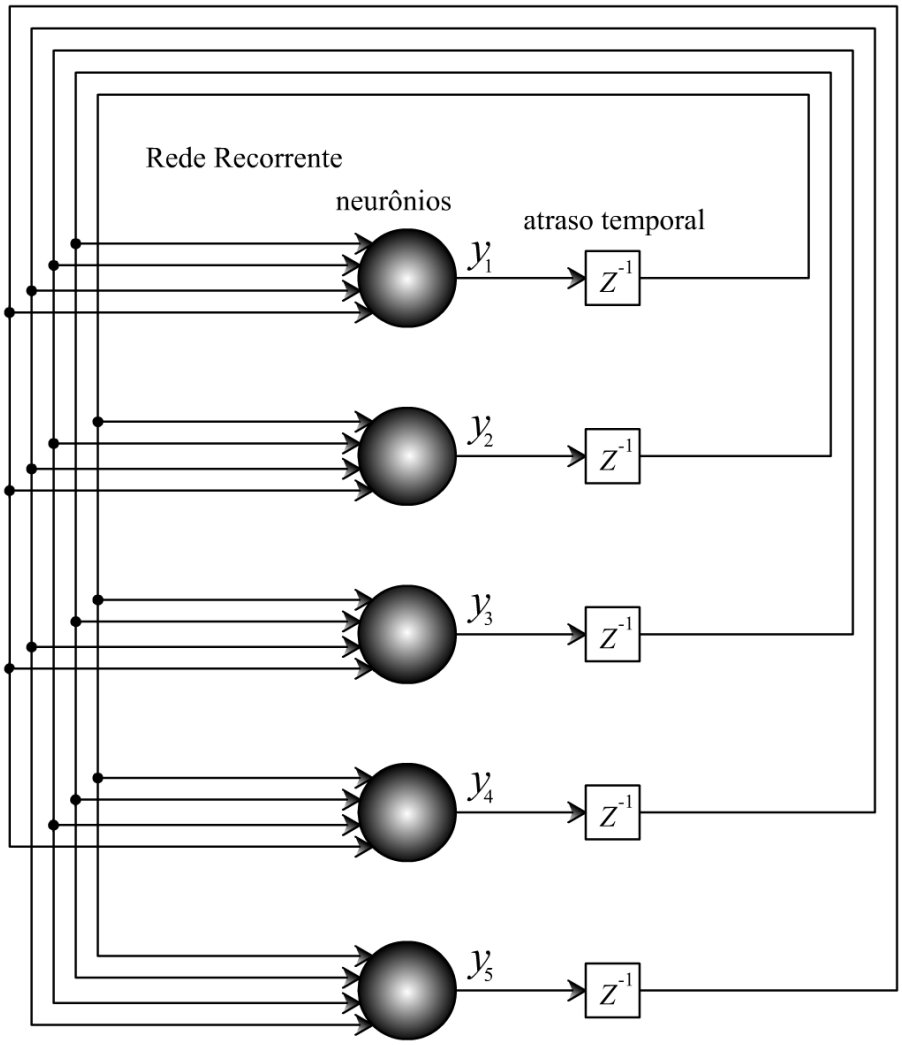

Figura 4.10: Exemplo de rede recorrente

- **Redes Auto-Organizáveis**. Arquitetura utilizada principalmente na clusterização de dados e solução de problemas combinatórios.
- **Redes RBF**. Basicamente possuem a mesma funcionalidade das redes MLP, porém, utilizam funções de base radial para a ativação dos neurônios, o que requer uma outra estratégia de treinamento, devido principalmente ao ajuste dos centros das funções radiais.

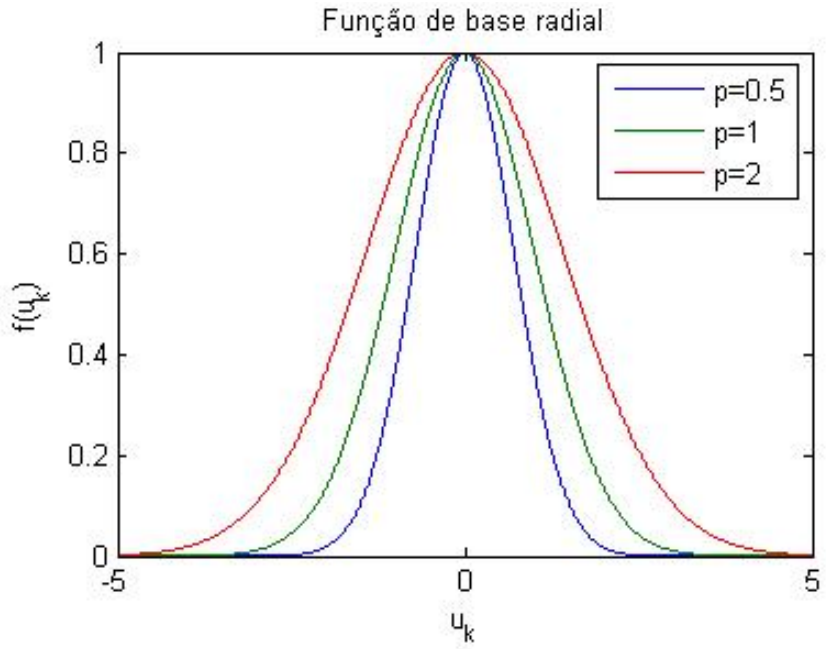

Figura 4.11: Função de ativação dos neurônios de uma rede RBF

- **Redes Construtivas**. Utilizam um processo construtivo de treinamento, onde neurônios são inseridos na rede iterativamente em função dos dados apresentados a rede e do seu treinamento.

Existem ainda outras propostas importantes na literatura como as redes sem peso, redes ART (*Adaptive Resonance Theory*) entre outras. Esse trabalho, porém, mantem seu foco nas redes MLP, objetivando estudar características específicas do treinamento dessa arquitetura de redes neurais.

4.4 Aprendizado de Redes Neurais

Uma das capacidades mais importantes das redes neurais artificiais é sua capacidade de aprendizado, que introduz uma grande versatilidade a esses dispositivos. A aplicação de uma RNA passa invariavelmente por um processo de adaptação, no qual a rede irá aprender através de exemplos e fazer interpolações e extrapolações do conhecimento por ela adquirido.

O conceito de aprendizagem pode ser definido de várias maneiras distintas, aplicáveis a contextos diferentes, porém, preservando o aspecto da adaptação com base cognitiva. A seguir, é mostrada uma definição formal de aprendizado voltado para redes neurais [42]. Nesse contexto, o aprendizado consiste basicamente no ajuste dos parâmetros livres da rede, que se dá através de um mecanismo que é específico para a arquitetura escolhida.

Aprendizagem é o processo pelo qual os parâmetros de uma rede neural são ajustados através de uma forma de estímulo pelo ambiente no qual a rede está operando, sendo o tipo específico de aprendizagem realizada definido pela maneira particular como ocorrem os ajustes realizados nos parâmetros.

Com as diversas arquiteturas propostas para redes neurais artificiais, surgiram também vários métodos de treinamento, que se dividem em dois grandes paradigmas de aprendizagem: *Aprendizagem Supervisionada* e *Aprendizagem Não-Supervisionada*. Existem também dois outros paradigmas de relevância nesta área: o aprendizado por reforço (caso particular do aprendizado supervisionado) e o aprendizado por competição (caso particular da aprendizagem não-supervisionada). Os dois paradigmas principais de treinamento serão apresentados em maiores detalhes nas próximas seções, com ênfase na aprendizagem supervisionada, que é o objetivo principal deste documento.

4.4.1 Aprendizagem Supervisionada

O mecanismo básico deste paradigma de aprendizado é o ajuste dos parâmetros da rede com base em um conjunto de amostras de entradas e saídas desejadas, que são apresentadas à rede por um dispositivo supervisor. Esse supervisor apresenta as entradas para a rede, que por sua vez, responde com uma saída calculada com base em seus parâmetros atuais. Ao supervisor cabe então comparar a saída da rede com a saída desejada e realizar um ajuste incremental nos parâmetros da rede, definido por algum método de aprendizagem. A Figura 4.12 ilustra de forma simplificada o processo de treinamento supervisionado.

Esse paradigma de aprendizagem representa uma das técnicas mais intuitivas de treinamento de redes neurais, por se assemelhar ao aprendizado humano a nível social, tendo considerável relevância nessa área devido ao grande número de problemas que podem ser facilmente modelados por seu intermédio. Dentre eles pode-se citar:

- Classificação e reconhecimento de padrões.
- Predição de séries temporais.
- Identificação de sistemas.
- Controle de processos.
- Projeto de filtros

O supervisor deve ser capaz de avaliar a qualidade da resposta apresentada pela rede para uma entrada conhecida. Para tanto é possível utilizar o critério do erro médio quadrático, no qual é tomada

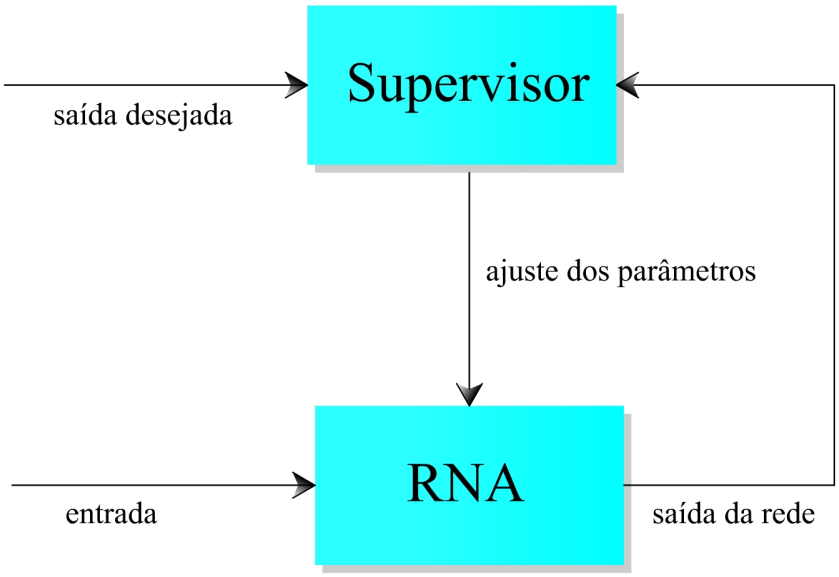

Figura 4.12: Paradigma de aprendizagem supervisionada

a média do quadrado da diferença entre a saída desejada e a observada para a rede neural. Consideremos um conjunto de amostras: x^{tre} e y^{tre}, contendo n_a pares de entrada/saída para o treinamento da rede. O erro médio quadrático é definido como segue:

$$E = \frac{1}{n_a} \sum_{i=1}^{n_a} \sum_{j=1}^{m} (y_j(i) - y_j^{tre}(i))^2 \tag{4.16}$$

onde $y_j(i)$ é a saída do *j-ésimo* neurônio da última camada da rede neural, calculada para a entrada $x^{tre}(i)$.

Portanto, a tarefa do supervisor no treinamento supervisionado é minimizar o erro E através do ajuste dos parâmetros livres da rede. Para tanto, é necessário que sejam conhecidas amostras de entrada da rede, x^{tre} e amostras de saídas esperadas para essas entradas y^{tre}. Essa coleção de amostras de entrada e saída $(x(i), y(i))$ expressa o comportamento que é desejado da rede, e sua escolha adequada é considerada um fator crítico, exigindo experiência e observação, pois dados pouco representativos do ponto de vista global do problema não irão gerar resultados satisfatórios, independentemente do método de treinamento escolhido e do algoritmo utilizado.

Existem inúmeros algoritmos para o treinamento supervisionado de redes neurais, sendo os exemplos mais conhecidos o algoritmo *Backpropagation* e a regra delta. Cada um desses algoritmos apresenta suas próprias características, que podem torná-los preferíveis em determinada aplicação. Porém, definido o problema e a arquitetura da rede, é pouco provável que o algoritmo de treinamento utilizado influencie no resultado final quando analisado o erro da rede.

O processo de aprendizado de redes neurais artificiais, apresenta uma dinâmica com relação aos incrementos aplicados aos parâmetros livres e a apresentação dos dados à rede. Essa dinâmica se dá geralmente de duas formas distintas, através do treinamento em lote e *on-line*.

Treinamento *On-Line*

Neste processo, o treinamento é realizado imediatamente após a apresentação de cada uma das amostras. Consideremos uma coleção contendo n_a amostras:

$$\{(x(1), y(1)), (x(2), y(2)), ..., (x(n_a), y(n_a))\}$$

Os parâmetros livres da rede são atualizados de forma independente para cada uma das amostras de treinamento: $(x(i), y(i))$; com isso, consegue-se uma representação mais fiel do vetor gradiente para cada uma das amostras. Porém, incrementos nos parâmetros livres que são minimizantes para a amostra $(x(i), y(i))$, podem aumentar o erro da rede para a amostra $(x(j), y(j))$. Outra vantagem em se utilizar esse modelo de treinamento é a facilidade de inserção de novas amostras na coleção, tornando o processo mais flexível, além de uma menor pré disposição em prender o processo de treinamento em um mínimo local.

Treinamento em Lote

Ao contrário do método anterior, no treinamento em lote todas as amostras da coleção são apresentadas a rede e o vetor gradiente é calculado para cada uma dessas amostras e devidamente armazenado. Ao final desse processo, quando já tiverem sido tomadas todas as amostras do conjunto de treinamento, é calculado o vetor gradiente médio, equação (4.17), que é utilizado na atualização dos parâmetros da rede. Cada operação de atualização realizada constitui uma época (ou iteração) do processo de treinamento, que é repetido por n_e épocas até ser atendido algum critério de parada. Esse processo em geral apresenta resultados superiores àquele apresentado anteriormente, por fornecer uma estimativa mais abrangente do vetor gradiente. Porém, esse método requer um maior armazenamento de memória física se forem alocados os vetores gradiente de cada amostra [43, 44], problema que pode ser contornado com a utilização de um método recursivo para obtenção do gradiente médio, $gradM$ (4.18).

$$grad = \frac{1}{n_a} \sum_{i=1}^{n_a} (gd(i)) \tag{4.17}$$

$$gradM(i+1) \;=\; gradM(i) + \frac{1}{i+1}\left(gd(i) - gradM(i)\right) \tag{4.18}$$

onde $gd(i)$ é o gradiente do erro calculado para *i-ésima* amostra.

Na equação (4.18) o gradiente médio é obtido iterativamente a partir do gradiente de cada amostra i, $gd(i)$, sendo o gradiente obtido ao final desse processo igual àquele proveniente da equação (4.17): $gradM(n_a + 1) = grad$, onde n_a é o número de amostras consideradas. Neste documento, será considerado como valor médio do gradiente a expressão: $grad$, sempre que for utilizado o treinamento em lote.

Ambas as estratégias de treinamento apresentadas possuem vantagens e desvantagens, sendo preferíveis em problemas diferentes [44]. Porém, o treinamento em lote mostra-se mais atrativo com relação as aplicações de interesse neste documento. Assim é dada uma enfase maior a esta técnica.

4.4.2 Aprendizagem Por Reforço

O aprendizado por reforço representa um caso particular do aprendizado supervisionado, a diferença aqui é que não existe nenhuma referência de como obter o comportamento desejado, como é feito pelo vetor gradiente, que representa a direção de maior acréscimo da função de erro, sendo possível assim, realizar incrementos nos parâmetros da rede na direção oposta a esse vetor e com isso reduzir o erro da rede.

A única informação disponível no método de treinamento por reforço é uma grandeza comumente conhecida por sinal de reforço, que pode ser um índice que indica a qualidade da solução apresentada pela rede, ou simplesmente um valor lógico (sim ou não, verdadeiro ou falso) indicando que a solução atende ou não a um dado requisito.

4.4.3 Aprendizagem Não-Supervisionada

Como o próprio nome sugere, neste método de aprendizado, não existe um supervisor que avalia a solução apresentada pela RNA, pois, ao contrário do aprendizado supervisionado a intenção deste método não é assimilar um padrão, e sim, mapear a redundância dos dados.

O aprendizado dos seres humanos, se observado superficialmente, principalmente a nível social, é bastante semelhante ao aprendizado supervisionado. Porém, existem muitos processos biológicos que não utilizam esse tipo de paradigma, como é o caso dos estágios iniciais dos sistemas de visão e audição humanos, que se auto-adaptam ao meio sem auxilio de um supervisor ou de padrões a serem seguidos.

Esse tipo de adaptação é semelhante àquele observado no aprendizado não supervisionado, onde a rede modifica seus parâmetros livres em resposta exclusivamente aos dados de entrada. Esse processo

é bastante útil em problemas de agrupamento de dados, onde são obtidos bons resultados com a utilização de redes neurais treinadas a partir de algoritmos de aprendizagem não supervisionada, como é o caso das redes de Kohonen.

4.5 Algoritmos de Treinamento Supervisionado

Grande parte dos algoritmos de treinamento supervisionado, requer o conhecimento do vetor gradiente do erro, utilizado pela primeira vez no contexto de redes com múltiplas camadas pelo método *backpropagation*, proposto por Rumelhart, Hinton e Williams em [45, 39, 46]. No método em questão, o vetor gradiente é proveniente de uma fase de retro-propagação, onde o erro da camada de saída da rede é retro-propagado até a primeira camada, sendo possível assim obter-se o vetor gradiente. De posse desse vetor, incrementos nos parâmetros da rede são tomados na direção oposta a ele, utilizando um passo pré-determinado conhecido como taxa de aprendizado.

A fim de demonstrar com clareza o método de obtenção do vetor gradiente, é exposto um exemplo onde são obtidas as equações em forma matricial para o cálculo desse vetor. Consideremos uma rede neural MLP contendo uma camada intermediária com n, m e o neurônios na 1^a, 2^a e 3^a camadas respectivamente. É apresentada à rede uma amostra de treinamento: $(X^{tre}(1), Y^{tre}(1))$; a saída da rede $y^{mlp}(1)$ é obtida pela fase *feedforward* descrita anteriormente, utilizando um conjunto de parâmetros aleatoriamente escolhidos. O erro da última camada pode então ser calculado como segue:

$$\begin{aligned} E &= (Y^{tre}(1) - Y^{mlp}(1)).^2 \\ &= \left(Y^1_{tre} - F^3(W^3 Y^2 + B^3)\right).^2 \end{aligned} \tag{4.19}$$

onde a operação matricial $A.^n$ é definida a seguir:

Definição 1 (Exponenciação Matricial Modificada) *A operação de exponenciação matricial modificada é dada pela equação (4.20) e é aqui introduzida com o intuito de simplicidade de notação.*

$$\begin{aligned} A.^n &= B \\ B_{i,j} &= A^n_{i,j} \ \forall i \ e \ \forall j \end{aligned} \tag{4.20}$$

A derivada do erro com relação as matrizes de parâmetros W^3 e B^3 é calculada utilizando a regra da cadeia, conforme mostram as equações que seguem:

$$\frac{\partial E}{\partial W^3} = \frac{\partial \left((Y^{tre}(1) - F^3(W^3 Y^2 + B^3)).^2\right)}{\partial W^3}$$

$$\begin{aligned} &= -2\left(Y^{tre}(1) - F^3(W^3Y^2 + B^3)\right) dF^3 {Y^2}^T \\ &= -2\left(Y^{tre}(1) - Y^3\right) dF^3 {Y^2}^T \end{aligned} \tag{4.21}$$

$$\begin{aligned} \frac{\partial E}{\partial B^3} &= \frac{\partial \left(\left(Y^{tre}(1) - F^3(W^3Y^2 + B^3)\right).^2\right)}{\partial B^3} \\ &= -2\left(Y^{tre}(1) - F^3(W^3Y^2 + B^3)\right) dF^3 \\ &= -2\left(Y^{tre}(1) - Y^3\right) dF^3 \end{aligned} \tag{4.22}$$

a grandeza dF^3 é definida pela matriz diagonal dada pela equação (4.23), onde m é o número de neurônios da camada k.

$$dF^k = \begin{bmatrix} f_1^{k\prime} & 0 & \cdots & 0 \\ 0 & f_2^{k\prime} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f_m^{k\,\prime} \end{bmatrix} \tag{4.23}$$

Essas equações (4.21) e (4.22) são utilizadas para calcular a variação do erro com relação aos parâmetros livres da rede quando é utilizada uma função de ativação sigmoidal na última camada. No caso dessa camada realizar apenas uma combinação linear dos sinais das camadas anteriores, podem ser utilizadas as equações (4.24) e (4.25) para o cálculo das grandezas em questão:

$$\frac{\partial E}{\partial W^3} = -2\left(Y^{tre}(1) - Y^3\right) {Y^2}^T \tag{4.24}$$

$$\frac{\partial E}{\partial B^3} = -2\left(Y^{tre}(1) - Y^3\right) \tag{4.25}$$

É possível agora definir a sensibilidade da última camada δ^3 como sendo:

$$\delta^3 = -2\left(Y^{tre}(1) - Y^3\right) dF^3 \tag{4.26}$$

onde δ^k é um vetor com dimensão igual ao número de neurônios da *k-ésima* camada.

Logo tem-se:

$$\begin{aligned} \frac{\partial E}{\partial W^3} &= \delta^3 {Y^2}^T \\ \frac{\partial E}{\partial B^3} &= \delta^3 \end{aligned}$$

Com base na sensibilidade da última camada, as taxas de variação do erro com relação aos parâmetros W^2 e B^2 podem ser calculadas conforme as equações que seguem, nas quais detalhes a respeito das transformações matriciais necessárias para respeitar as dimensões de cada matriz foram omitidos:

$$\begin{aligned}
\frac{\partial E}{\partial W^2} &= \frac{\partial \left((Y^{tre}(1) - F^3(W^3 Y^2 + B^3)) .^2 \right)}{\partial W^2} \\
&= dF^3 \frac{\partial F^3(W^3 Y^2 + B^3)}{\partial Y^2} \frac{\partial Y^2}{\partial W^2} \left(-2 \left(Y^{tre}(1) - Y^3 \right) \right) \\
&= \frac{\partial F^2(W^2 Y^1 + B^2)}{\partial W^2} {W^3}^T \delta^3 \\
&= dF^2 {W^3}^T \delta^3 {Y^1}^T \qquad (4.27) \\
\frac{\partial E}{\partial W^2} &= dF^2 {W^3}^T \delta^3 \qquad (4.28)
\end{aligned}$$

É possível então definir a sensibilidade da segunda camada δ^2 com base nas equações (4.27) e (4.28) como segue:

$$\delta^2 = dF^2 {W^3}^T \delta^3 \qquad (4.29)$$

É possível agora reescrever as taxas de variação em relação aos parâmetros livres da segunda camada em função de δ^3 como segue:

$$\begin{aligned}
\frac{\partial E}{\partial W^2} &= \delta^2 {Y^2}^T \qquad (4.30) \\
\frac{\partial E}{\partial B^2} &= \delta^2 \qquad (4.31)
\end{aligned}$$

Uma expressão genérica para o cálculo das sensibilidades é apresentada na equação (4.32), que demonstra claramente que as sensibilidades são retro-propagadas da última camada em direção à primeira camada para a obtenção do gradiente (dado pela taxa de variação do erro com relação aos parâmetros livres da rede):

$$\delta^k = dF^k \left(W^{k+1} \right)^T \delta^{k+1} \qquad (4.32)$$

As taxas de variação com relação aos parâmetros livres da camada k podem então ser calculados em função de δ^k:

$$\frac{\partial E}{\partial W^k} = \delta^k \left(Y^{k-1}\right)^T \tag{4.33}$$

$$\frac{\partial E}{\partial B^k} = \delta^k \tag{4.34}$$

Se os parâmetros livres da rede forem reorganizados em um vetor, aqui chamado de vetor de parâmetros: $\omega \in R^N$, onde N é o número de parâmetros livres da rede, então as taxas de variação do erro em relação a esses parâmetros também podem ser organizadas para formar o vetor gradiente. Para tanto, deve-se manter a mesma estrutura em ambos os vetores, ou seja, dado que um elemento $w_{i,j}^k$ ocupa a posição α do vetor de parâmetros: $w_{i,j}^k = \omega_\alpha$, então $\partial E/\partial w_{i,j}^k$ deve ocupar a posição α do vetor gradiente: $\partial E/\partial w_{i,j}^k = grad_\alpha$. O erro da rede pode ser redefinido em função dos parâmetros livres da rede, vetor ω:

$$e(\omega) = (S - Y^{n_c}).^2 \tag{4.35}$$

onde S é a saída desejada e Y^{n_c} é a saída da última camada da rede contendo n_c camadas, calculada utilizando o conjunto de parâmetros contidos no vetor ω.

4.5.1 Backpropagation

Estudado inicialmente por Werbos em 1974 [47], e posteriormente redescoberto por Rumelhart, Hinton e Williams em 1986 [45], o backpropagation ainda é um dos algoritmos mais difundidos para o treinamento de redes MLP. Nesse algoritmo são calculados incrementos nos parâmetros livres com base nas seguintes equações:

$$\Delta W^k(t) = \eta \delta^k(t)(Y^k(t))^T \tag{4.36}$$

$$\Delta B^k(t) = \eta \delta^k(t)(Y^k(t))^T \tag{4.37}$$

onde η é a taxa de aprendizagem a ser definida.

Os pesos e polarizações da próxima iteração podem então ser calculados como segue:

$$W^k(t+1) = W^k(t) - \Delta W^k(t) \tag{4.38}$$

$$B^k(t+1) = B^k(t) - \Delta B^k(t) \tag{4.39}$$

4.5.2 Gradiente Ótimo

O algoritmo backpropagation pode ser classificado como um método de gradiente descendente, pois realiza incrementos nos parâmetro livres na direção oposta ao vetor gradiente. Nesta categoria de métodos de otimização, também conhecida como otimização de ordem 1 por utilizar informação a respeito da primeira derivada, é utilizado apenas o vetor gradiente da função na obtenção de uma direção de descida. Todos os métodos de ordem 1 apresentam uma taxa de convergência linear, sendo a convergência super-linear, observada no método gradiente ótimo, a maior dentre os algoritmos dessa classe.

O algoritmo gradiente ótimo consiste basicamente no mesmo processo utilizado no método backpropagation. Porém, são calculados passos ótimos a cada iteração, ao invés de utilizar um passo constante η. A equação para a atualização dos parâmetros utilizando o algoritmo do gradiente ótimo é mostrada na equação (4.40)[48]:

$$\omega(t+1) = \omega(t) - \alpha(t) grad(t) \tag{4.40}$$

onde o passo ótimo $\alpha(t)$ deve ser calculado a cada iteração através de um método de busca unidimensional, assim conhecida por realizar uma busca ao longo apenas de uma direção, no caso, a direção do vetor gradiente. Neste documento, sempre que tornar-se necessária a busca unidimensional, é implementado o método da razão áurea [48].

A utilização do passo ótimo α provê um ganho em convergência considerável quando comparado ao método backpropagation, que utiliza um passo constante η. É possível ainda utilizar um passo α sub-ótimo, calculado com base no método da seção aurea [48] por exemplo, o que ainda garante uma convergência superior àquela utilizando um passo constante. A seguir, é mostrado o algoritmo básico do gradiente ótimo para treinamento de redes MLP.

Inicializar o vetor de parâmetros ω;
for $t = 1$ *até* n_e **do**
  Calcular o vetor gradiente $grad(t)$;
  $\alpha = \arg\min_{\alpha \geq 0} erro\left(\omega(t) - \alpha\, grad(t)\right)$;
  $\omega(t+1) = \omega(t) - \alpha(t) grad(t)$;
end

Algoritmo 5: Gradiente Ótimo

4.5.3 DFP: Davidon Fletcher Powell

Este é um método classificado como quasi-Newton, no qual a informação a respeito da matriz hessiana do problema é construída de forma iterativa. Como é utilizada informação a respeito da segunda derivada, esse método é considerado como sendo de ordem 2. O algoritmo DFP é baseado em uma correção de posto 2, no qual, a cada iteração, são somadas à inversa matriz da hessiana duas matrizes de posto unitário. A equação (4.41) descreve a atualização da matriz H implementada no método DFP [48]:

$$H(t+1) \quad = \quad H(t) + \frac{p(t)p(t)^T}{p(t)^T q(t)} - \frac{H(t)q(t)q(t)^T H(t)}{q(t)^T H(t) q(t)} \tag{4.41}$$

O Algoritmo 6 descreve o método DFP aplicado ao treinamento de uma rede MLP [48].

Inicializar o vetor de parâmetros $\omega(0)$;
$H(0) = I$;
while *continua* **do**
  $g(t) = grad(t)$;
  $d(t) = -H(t)g(t)$;
  $\alpha = \arg\min_{\alpha \geq 0} e\left(\omega(t) + \alpha\, d(t)\right)$;
  $\omega(t+1) = \omega(t) + \alpha(t)d(t)$;
  if $k < n_\omega - 1$ **then**
    $g(t+1) = grad(t+1)$;
    $q(t) = g(t+1) - g(t)$;
    $p(t) = -\alpha(t)d(t)$;
    $H(t+1) = H(t) + \frac{p(t)p(t)^T}{p(t)^T q(t)} - \frac{H(t)q(t)q(t)^T H(t)}{q(t)^T H(t) q(t)}$;
  else
    $\omega(0) = \omega(n_\omega)$;
    $g(0) = grad(n_\omega)$;
    $t = 0$;
  end
  Analisa critério de convergência;
end

Algoritmo 6: Davidon Fletcher Powell, DFP

4.5.4 BFGS: Broyden Fletcher Goldfarb Shanno

Este algoritmo representa outro método quasi-Newton, com características semelhantes ao método DFP [49, 48]. A principal diferença entre os dois métodos consiste no mecanismo de aproximação iterativo da matriz hessiana. No algoritmo BFGS, os autores desenvolveram equações iterativas para o cálculo da matriz hessiana, e posteriormente é feita a inversão dessa matriz com base nas equações obtidas, diferentemente do método DFP, onde é estimada diretamente a inversa da hessiana.

O método BFGS apresenta, em geral, um melhor condicionamento numérico do que o método DFP. O algoritmo para aplicação desse método é o mesmo descrito anteriormente no Algoritmo 6, diferindo apenas pela matriz H. A equação (4.42) descreve a matriz B, que deve ser substituída pela matriz H no Algoritmo 6 para a obtenção do algoritmo BFGS:

$$B(t+1) = B(t) + \frac{q(t)q(t)^T}{q(t)^T q(t)} - \frac{B(t)p(t)p(t)^T B(t)}{p(t)^T B(t) p(t)} \tag{4.42}$$

4.5.5 FR: Fletcher Reeves

Este é um método de direções conjugadas de ordem 2, com convergência n-passos quadrática. Nele, o cálculo da matriz hessiana é substituído por buscas unidimensionais. A seguir, é mostrado o algoritmo básico para utilização do método FR:

Inicializar o vetor de parâmetros $\omega(0)$;
while *continua* **do**
  $g(0) = grad(0)$;
  $d(0) = -g(0)$;
  for $t = 0$ *até* $n_\omega - 1$ **do**
    $\alpha = \arg\min_{\alpha \geq 0} e\left(\omega(t) + \alpha\, d(t)\right)$;
    $\omega(t+1) = \omega(t) + \alpha(t) d(t)$;
    $g(t+1) = grad(t+1)$;
    if $t < n_\omega - 1$ **then**
      $\beta(t) = g(t+1)^T g(t+1) / g(t)^T g(t)$;
      $d(t+1) = -g(t+1) + \beta(t) d(t)$;
    end
    $\omega(0) = \omega(n_\omega)$;
  end
  Avalia o critério de parada;
end

Algoritmo 7: Fletcher Reeves, FR

4.5.6 SCG: Gradiente Conjugado Escalonado

Este também é um método de direções conjugadas, com propriedades semelhantes ao método FR. Sua grande vantagem em relação aos outros métodos de alto desempenho apresentados até o momento é a inexistência de buscas unidimensionais, que exigem um grande esforço computacional tomando um tempo considerável em seu processamento.

Dentre os métodos procedurais de otimização encontrados na literatura pertinente ao assunto, o método SCG é o que apresenta uma maior eficiência quando se analisa a relação entre velocidade de convergência do método e número de operações necessárias para realizar uma iteração. O algoritmo que segue descreve o método SCG aplicado ao treinamento de redes neurais [50].

Inicializar o vetor de parâmetros: $\omega(1)$;
Escolher: $\sigma_0 > 0$, $\lambda(1) > 0$ e $\bar{\lambda}(1) = 0$;
Determinar: $p(1) = r(1) = -grad(1)$, $k = 1$ e $sucesso = verdadeiro$;
while *continua* **do**
  if *sucesso* **then**
    $\sigma(t) = \sigma_0/|p(t)|$;$s(t) = \frac{grad(\omega(t)+\sigma(t)p(t))-grad(\omega)}{\sigma(t)}$; $\delta(t) = p(t)^T s(t)$;
  end
  $s(t) = s(t) + (\lambda(t) - \bar{\lambda}(t))p(t)$; $\delta(t) = \delta(t) + (\lambda(t) - \bar{\lambda}(t))|p(t)|^2$;
  if $\delta(t) \leq 0$ **then**
    $s(t) = s(t) + \left(\lambda(t) - c\frac{\delta(t)}{|p(t)|^2}\right)p(t)$; $\bar{\lambda}(t) = 2\left(\lambda(t) - \frac{\delta(t)}{|p(t)|^2}\right)$;
    $\delta(t) = -\delta(t) + \lambda(t)|p(t)|^2$; $\lambda(t) = \bar{\lambda}(t)$;
  end
  $\mu(t) = p(t)^T r(t)$; $\alpha(t) = \frac{\mu(t)}{\delta(t)}$; $\Delta(t) = \frac{2\delta(t)[e(\omega(t))-e(\omega(t)+\alpha(t)p(t))]}{\mu^2(t)}$;
  if $\Delta(t) \geq 0$ **then**
    $\omega(t+1) = \omega(t) + \alpha(t)p(t)$; $r(t+1) = -grad(\omega(t+1))$;
    $\bar{\lambda}(t) = 0$;sucesso=verdadeiro;
    if k *mod* n_ω*=0* **then**
      $p(t+1) = r(t+1)$;
    else
      $\beta(t) = \frac{|r(t+1)|^2 - r(t+1)^T r(t)}{\mu(t)}$; $p(t+1) = r(t+1) + \beta(t)p(t)$;
    end
    if $\Delta(t) \geq 0.75$ **then**
      $\lambda(t) = \frac{1}{2}\lambda(t)$;
    else
      $\bar{\lambda(t)} = \lambda(t)$;sucesso=falso;
    end
    if $\Delta(t) < 0.25$ **then**
      $\lambda(t) = 4\lambda$;
    end
    $k = k + 1$;
  end
  Avaliar critério de parada;
end

Algoritmo 8: Gradiente Conjugado Escalonado, SCG

4.5.7 Algoritmos Genéticos

Os algoritmos genéticos representam um popular método de otimização de ordem 0, também conhecido como busca cega, por não levar em consideração a informação a respeito do gradiente tampouco da hessiana da função objetivo. Sua aplicação ao treinamento de redes neurais é bastante simples, já tendo sido estudada por diversos pesquisadores [51, 52, 53]. Geralmente, é utilizada uma codificação real para os cromossomos, que dessa forma, serão representados por pontos do espaço R^N. Os operadores genéticos de reprodução, bem como o mecanismo de seleção, podem ser escolhidos arbitrariamente, levando a resultados diferentes dependendo dessa escolha. A população inicial é escolhida geralmente de forma aleatória, contida em um hipercubo H centrado na origem do espaço R^N.

A avaliação do nível de adaptação de cada indivíduo da população é feita com base no erro médio quadrático apresentado pela rede quando é utilizado o conjunto de parâmetros definido pelo cromossomo desse indivíduo. Uma possível escolha para a função de fitness é mostrada na equação (4.43); várias outras propostas são possíveis, sendo necessário apenas que a função escolhida seja inversamente proporcional ao erro da rede e estritamente positiva:

$$fitness_i = e^{-erro(p_i)\tau} \tag{4.43}$$

onde p_i é um ponto do espaço R^N representado pelo cromossomo i, τ é um parâmetro a ser escolhido, que define a sensibilidade da função e $erro(\omega)$ é o erro médio quadrático da rede utilizando o conjunto de parâmetros (pesos e polarizações) ω.

A escolha do parâmetro τ tem relevante influência na eficiência do algoritmo. Valores pequenos de τ irão tornar a população mais homogênea, preservando a diversidade, enquanto valores mais elevados de τ criam uma diferença maior entre o *fitness* dos indivíduos melhor e pior adaptados. Essa diferença gera uma aceleração na taxa de convergência do método, podendo causar também a convergência prematura quando o parâmetro τ é mal dimensionado.

Outros parâmetros a serem escolhidos para a aplicação do método são: a taxa de reprodução, a taxa de mutação, o tamanho da população (n_p) e o critério de convergência. Uma possível implementação de um algoritmo genético para treinamento de redes MLP pode ser observada no Algoritmo 9:

Definir $\tau > 0$;
Inicializar a população aleatoriamente contida $H \in R^N$;
for $i = 1$ *até* n_p **do**
  $fitness_i = e^{-erro(p_i)\tau}$;
end
while *continua* **do**
  Selecionar indivíduos da próxima geração;
  Selecionar pares para reprodução;
  Aplicar operador de reprodução;
  Selecionar indivíduos para mutação;
  Aplicar operador de mutação;
  for $i = 1$ *até* n_p **do**
    $fitness_i = e^{-erro(p_i)\tau}$;
  end
  $k = \arg\max_k fitness_k$;
  Avaliar critério de parada
end
$\omega = p_k$;

Algoritmo 9: Algoritmo genético para treinamento de redes neurais

4.5.8 Otimização Por Enxame de Partículas: PSO

Assim como os algoritmos genéticos, a otimização por enxame de partículas pode ser utilizada como uma ferramenta de busca cega para o treinamento de redes MLP. A codificação do problema, neste caso, também representa os indivíduos da população por pontos do espaço R^N, da mesma forma como é feito para os algoritmos genéticos. A adaptação desses indivíduos pode ser calculada por uma função direta do erro da rede, e as regras de iteração dos indivíduos podem ser aquelas utilizadas no algoritmo original de otimização por enxame de partículas, ou ainda utilizar algum método que proporcione um ganho de performance, como a modulação de velocidade.

É necessária a definição de alguns parâmetros para a aplicação desse método, dentre os quais pode-se citar:

- Tamanho da população: n_p.
- Coeficientes de aceleração: c_1 e c_2.
- Peso de inércia: w, quando utilizado.

- Critério de convergência.

O Algoritmo 10 mostra uma possível implementação da otimização por enxame de partículas utilizando modulação de velocidade para o treinamento de redes MLP.

Definir $\tau > 0$;
Inicializar a posição das partículas $x(0)$, aleatoriamente contidas em: $H \in R^N$;
Inicializar as velocidades $v(0)$, aleatoriamente contidas em: $H \in R^N$;
Inicializar os termos cognitivos $p_i(0)$, aleatoriamente contidas em: $H \in R^N$;
for $i = 1$ *até* n_p **do**
  $\beta_i = erro(p_i)$;
end
$i = \arg\min_i beta_i$;
$p_g = x_i(0)$;
$t = 0$;
while *continua* **do**
  for $i = 1$ *até* n_p **do**
    Escolher α_1 e α_2 aleatoriamente;
    $v_i(t+1) = v_i(t) + c_1\alpha_1(p_i - x_i(t)) + c_2\alpha_2(p_g - x_i(t))$;
    $\theta = ang(v_i(t+1), v_i(t))$;
    $v_i(t+1) = \gamma(\theta)v_i(t+1)$;
    $x_i(t+1) = x_i(t) + v(t+1)$;
    $\beta_i = f(x_i)$;
    if $\beta_i \leq erro(p_i)$ **then**
      $p_i = x_i$
    end
    if $\beta_i \leq erro(p_g)$ **then**
      $p_g = x_i$
    end
  end
  Avaliar critério de parada;
  $t = t + 1$;
end
$\omega = p_g$;

Algoritmo 10: Otimização por enxame de partículas para o treinamento de redes neurais, utilizando modulação de velocidade

4.5.9 Gradiente Evolutivo

A aplicação de algoritmos genéticos, otimização por enxame de partículas e outras meta-heurísticas ao treinamento de redes MLP não traz em geral resultados comparáveis àqueles obtidos com a utilização de métodos procedurais, como BFGS e SCG, quando é analisada a taxa de convergência do treinamento. Essa característica é devida em partes à utilização de informação sobre o gradiente do erro e da sua hessiana, o que proporciona um aumento na eficiência do método de busca.

Uma alternativa encontrada para a utilização de algoritmos genéticos de forma eficiente no treinamento de redes MLP é a utilização do método hibrido GE: *Gradiente Evolutivo* [54], inspirado no trabalho de Chalmers: *The Evolution of Learning: An Experiment in Genetic Connectionism*[55], que propõe um processo para evoluir a regra delta para treinamento de redes perceptron de uma camada. O algoritmo GE aplica o processo evolutivo não diretamente na otimização dos parâmetros livres da rede, mais sim na evolução do vetor gradiente. Nesse algoritmo é utilizada a estrutura básica do método backpropagation ou do método do gradiente, porém, o vetor gradiente e o passo são obtidos por meio de um processo evolutivo.

Codificação

O método GE utiliza uma codificação real, onde os indivíduos da população representam possíveis vetores gradiente. A seguir são descritos detalhes da codificação utilizada.

- Cromossomo: vetores p do espaço R^N.
- Fitness: $fitness_i = \exp(-erro(\omega - p_i))$, onde $erro(\omega)$ é o erro médio quadrático da rede.
- Seleção: Roleta e Elitista.
- Reprodução: Operador de reprodução matricial, exposto previamente neste documento.
- Mutação: Mutação vetorial, onde é somado um ruído a todas as componentes do cromossomo.
- População inicial: Escolhida aleatoriamente em um hipercubo $H \in R^N$ centrado na origem e de lado unitário.

O Algoritmo GE

Inicialmente é proposto o algoritmo que utiliza como base o método do gradiente. Nele, a cada iteração é calculado o vetor gradiente inicial $grad^0$, através da retro-propagação do erro, e esse vetor é posteriormente evoluído via algoritmo genético durante n_g gerações, utilizando uma população de n_p indivíduos. Após o processo evolutivo, é realizado um incremento no vetor ω na direção oposta ao

gradiente evoluído, utilizando um passo sempre unitário, tendo em vista que a aplicação do algoritmo genético evolui simultaneamente o gradiente e o passo.

A utilização da fase de retro-propagação do erro visa proporcionar uma condição inicial favorável ao processo evolutivo, aumentando com isso a taxa de convergência do método. Em algumas aplicações não é possível calcular o gradiente da rede, como ocorre em [56]. Nesses casos é possível ignorar essa etapa, escolhendo $grad^0 = 0$, obtendo ainda assim resultados bastante satisfatórios [56], dado que esta escolha também representa uma boa condição inicial, em algumas situações até mesmo superior à que utiliza o gradiente do erro.

O Algoritmo 11 ilustra a implementação do método proposto, onde $AG(n_p, n_g, grad^0)$ representa o processo evolutivo tomando como condição inicial o ponto: $grad^0$.

Determinar: n_p, n_g;
Inicializar o vetor de parâmetros: $\omega(1)$;
$t = 1$;
while *continua* **do**
  Calcular $grad^0$ através da retro-propagação do erro;
  Evoluir o gradiente: $grad = AG(n_p, n_g, grad^0)$;
  if $erro(\omega(t) - grad) < erro(\omega(t))$ **then**
    $\omega(t+1) = \omega(t) - grad$;
  end
  Avaliar critério de parada;
  $t = t + 1$;
end

Algoritmo 11: Gradiente Evolutivo

O algoritmo AG, utilizado no Algoritmo 11 é descrito no Algoritmo 12

Entrada: n_p, n_g
Saída: $grad$
Definir $\tau > 0$;
Inicializar a população aleatoriamente contida $H \in R^N$;
Inserir $-grad^0$ na população inicial;
for $i = 1$ *até* n_p **do**
  $fitness_i = e^{-erro(\omega - p_i)\tau}$;
end
for $t = 1$ *até* n_g **do**
  Selecionar indivíduos da próxima geração;
  Selecionar pares para reprodução;
  Aplicar operador de reprodução;
  Selecionar indivíduos para mutação;
  Aplicar operador de mutação;
  for $i = 1$ *até* n_p **do**
    $fitness_i = e^{-erro(\omega - p_i)\tau}$;
  end
  $k = \arg\max_k fitness_k$;
end
$grad = p_k$;

Algoritmo 12: AG

A implementação desse método mostra-se bastante vantajosa do ponto de vista da sua eficiência, que é bastante superior àquela observada quando aplicado um algoritmo genético diretamente na otimização dos parâmetros livres da rede neural, bem como na utilização do método backpropagation, que leva em conta informação a respeito do vetor gradiente. A combinação dessas duas técnicas cria um método heurístico de alta performance, que pode ser utilizado mesmo quando não é possível realizar a retro-propagação do erro, capacidade observada apenas nos métodos de busca cega. Dessa forma, são unidas características de grande interesse em um único algoritmo de treinamento.

4.5.10 Gradiente das Partículas

Seguindo a mesma metodologia anterior, o algoritmo GP: *Gradiente das Partículas*, apresenta uma solução hibrida para o problema de treinamento de uma rede MLP. Nesta abordagem, é utilizada a otimização por enxame de partículas aliada ao método do gradiente para a otimização dos parâmetros livres da rede.

Este método proporciona os mesmos benefícios obtidos no algoritmo GE, utilizando também a mesma estrutura. A sua diferença encontra-se na evolução do gradiente, que aqui é realizada através de um enxame de partículas, e não mais por um algoritmo genético. A codificação do algoritmo de enxame exige a definição de alguns parâmetros, sendo eles:

- Cada partícula representa um possível vetor gradiente.
- A qualidade da solução representada pela partícula i é dada por: $\beta_i = erro(\omega - x_i)$.
- A população inicial contém n_p partículas.
- A busca pelo vetor gradiente é realizada por n_g iterações.

O algoritmo GP é basicamente o mesmo mostrado no Algoritmo 11 para o método GE, diferindo apenas no método de busca do gradiente, que agora passa a ser dado por: $grad = PSO(n_p, n_g, grad^0)$ e não mais por $grad = AG(n_p, n_g, grad^0)$. O algoritmo PSO, utilizado pelo algoritmo GP, é mostrado a seguir no Algoritmo 13.

Entrada: n_p, n_g
Saída: $grad$
Definir $\tau > 0$;
Inicializar a posição das partículas $x(1)$, aleatoriamente contidas em: $H \in R^N$;
Inicializar as velocidades $v(1)$, aleatoriamente contidas em: $H \in R^N$;
Inicializar os termos cognitivos $p_i(1)$, aleatoriamente contidas em: $H \in R^N$;
Inserir o vetor $grad^0$ na população inicial;
for $i = 1$ *até* n_p **do**
  $beta_i = erro(\omega - x_i)$;
end
$i = \arg\min_i beta_i$;
$p_g = x_i(0)$;
for $t = 1$ *até* n_g **do**
  for $i = 1$ *até* n_p **do**
    Escolher α_1 e α_2 aleatoriamente;
    $v_i(t+1) = v_i(t) + c_1\alpha_1(p_i - x_i(t)) + c_2\alpha_2(p_g - x_i(t))$;
    $\theta = ang(v_i(t+1), v_i(t))$;
    $v_i(t+1) = \gamma(\theta)v_i(t+1)$;
    $x_i(t+1) = x_i(t) + v(t+1)$;
    $\beta_i = erro(\omega - x_i)$;
    if $\beta_i \leq erro(\omega - p_i)$ **then**
      $p_i = x_i$
    end
    if $\beta_i \leq erro(\omega - p_g)$ **then**
      $p_g = x_i$
    end
  end
end
$grad = p_g$;

Algoritmo 13: PSO

Nas seções seguintes, são considerados vários exemplos de aplicação de redes MLP utilizados na análise da eficiência relativa dos vários algoritmos de treinamento estudados neste documento, a fim de criar uma referência concreta com relação aos ganhos alcançados com a aplicação de cada método.

4.6 Exemplos e Comparações

A fim de avaliar a eficiência relativa entre os métodos de treinamento estudados neste documento, serão analisados alguns exemplos de aplicação das redes MLP, mais precisamente nas áreas de aproximação de modelos e predição de séries temporais, sendo a aproximação de modelos o foco principal desse estudo devido a sua relação com o problema de estimação de estado da forma como este é abordado no Capítulo 5.

Como o intuito aqui é analisar apenas a eficiência do processo de treinamento, não serão definidos os conjuntos de teste e validação, que são utilizados na avaliação e manutenção da qualidade do treinamento e em nada influenciam na taxa de convergência desse processo. Seguindo esse mesmo raciocínio, o critério de parada do processo iterativo de treinamento é definido sempre com relação a um número pré-fixado de épocas/iterações e o erro final apresentado pela rede é relativo ao conjunto de dados de treinamento.

A seguir, são apresentados três problemas que serão a base das análises realizadas nas próximas seções deste documento.

- *Problema 1*: Aproximação da função $y = x^2$.

  Para este problema, é utilizado um conjunto de treinamento contendo 100 amostras de entrada da rede: $x^{tre} \in R^{100}$ uniformemente espaçadas dentro do intervalo $[0, 1]$ e com as 100 respectivas saídas da rede $Y^{tre} \in R^{100}$, onde $y_i^{tre} = (x_i^{tre})^2$ para todo $i = \{1, 2, ..., 100\}$.

- *Problema 2*: Correntes de motores de indução.

  Em tese, a corrente de um motor de indução trifásico pode ser facilmente definida com base na tensão e na potência desse motor, como mostra a equação (4.44).

$$I = \frac{P}{\sqrt{3}V\eta} \tag{4.44}$$

  onde P e V representam a potência e a tensão do motor respectivamente. A variável η leva em conta o rendimento e o fator de potência do motor, que por sua vez, são função de fatores construtivos, da carga mecânica e da rotação do motor. Dessa forma, é clara a dificuldade existente em se especificar a variável η, bem como a corrente do motor.

  O problema em questão utiliza um estimador neural para o cálculo dessa corrente, baseando-se na potência, rotação e tensão do motor, através de uma rede MLP contendo 3 neurônios em sua camada sensorial e 1 neurônio em sua camada de saída.

  O conjunto de treinamento utilizado consta de 100 amostras obtidas de catálogos de fabricantes, compreendendo motores que se enquadrem nas seguintes faixas de valores:

- Potência: 0.1 a 330 KW.
- Rotação: 600,900,1200,1800 e 3600 rpm.
- Tensão: 220, 380 e 440 V.
- Corrente: 0.3 a 580 A.

- *Problema 3*: Predição da série temporal *Sunspot*

  Por fim, a título de complementariedade, é analisado o problema de estimação da série temporal *Sunspot*. Essa predição é feita com base em 10 amostras anteriores para prever a atual, sendo o conjunto de treinamento composto por 230 amostras de entrada da rede, $x \in R^{230\times 10}$, e 230 amostras da saída esperada da rede, $y \in R^{230}$ mostradas na Figura 4.13.

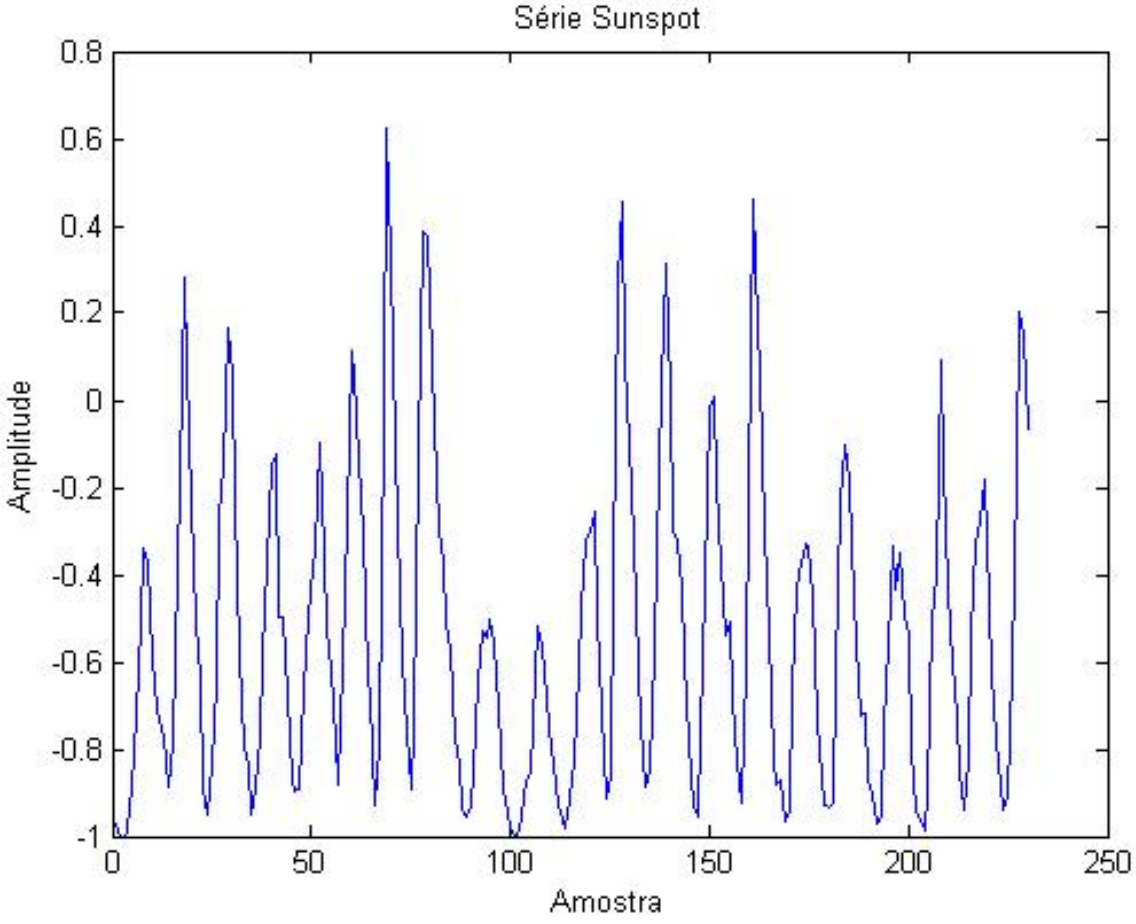

Figura 4.13: Série Sunspot

4.6.1 Algoritmo Genético $\times$ Otimização por Enxame de Partículas

A intenção aqui é comparar a eficiência relativa entre a aplicação de algoritmos genéticos e a otimização por enxame de partículas no treinamento de redes neurais MLP. Como ambos os métodos são estocásticos, serão realizadas várias repetições em cada processo de treinamento e analisados os valores médios do erro e da diversidade média da população obtidos durante o processo iterativo.

O algoritmo genético analisado neste documento faz uso do operador de reprodução matricial, enquanto o algoritmo de otimização por enxame de partículas utiliza a modulação de velocidade. Ambas essas escolhas são feitas com base nos resultados obtidos anteriormente, que demonstram que essas configurações são as mais eficientes aqui estudadas, quando analisadas as funções de teste propostas.

A topologia da rede neural será definida por um vetor linha: $[a_1, a_2, ..., a_{n_c}]$, onde o primeiro componente: a_1 representa o número de neurônios da camada sensorial, o ultimo componente a_{n_c} representa o número de neurônios na camada de saída e os demais componentes representam os números de neurônios nas camadas intermediárias.

Consideremos em uma primeira análise o problema 1, onde são aplicados algoritmos de treinamento com os seguintes parâmetros:

- Topologia da rede: [1,3,1].
- Número de repetições do processo de treinamento: 20.
- Tamanho das populações: $n_p = 100$.
- Número de épocas por repetição: $n_e = 40$.

Os valores médios obtidos para o erro da rede durante a fase de treinamento são mostrados no gráfico da Figura 4.14.

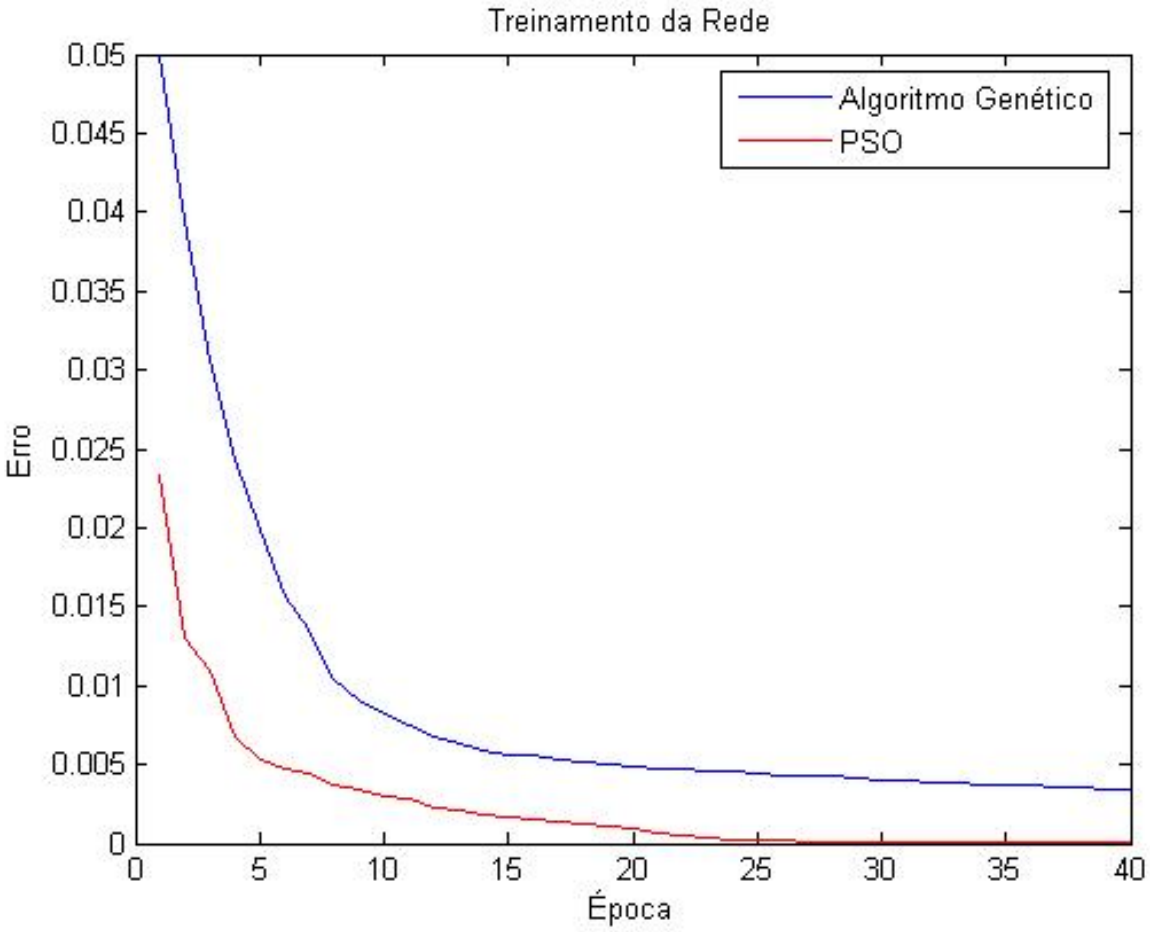

Figura 4.14: Erro médio durante o treinamento da rede, AG $\times$ PSO - Problema 1

Como era esperado, o resultado obtido com a utilização do método PSO é consideravelmente superior àquele observado com a utilização do algoritmo genético, confirmando os resultados obtidos no Capítulo 3, onde foi constatada uma superioridade em convergência para o algoritmo PSO, sendo obtida, por outro lado, uma maior preservação da diversidade[3] com o algoritmo genético.

A Figura 4.15 expressa claramente essa diferença na preservação de diversidade dos dois algoritmos em questão quando aplicados ao problema em análise.

[3]Novamente a diversidade é tomada como sendo uma métrica da dispersão da população, sendo considerada a variância obtida no valor da função calculada para cada indivíduo/partícula.

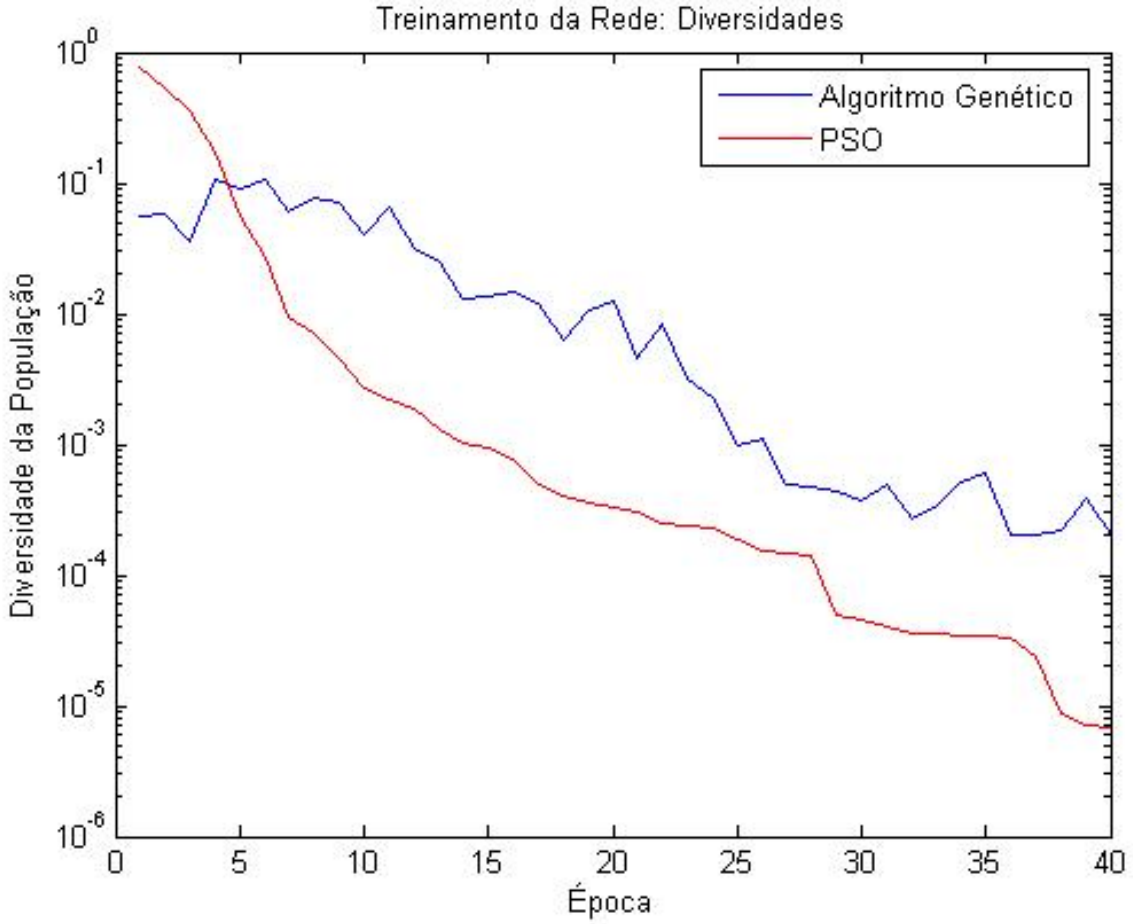

Figura 4.15: Diversidade média da população, AG $\times$ PSO - Problema 1

Consideremos agora o problema 2, que apresenta uma complexidade maior na assimilação do mapeamento dos dados de entrada e saída, devido às características do relacionamento entre esses dados, bem como à dimensão do problema, que requer 3 neurônios na camada sensorial. Os parâmetros escolhidos para este teste são mostrados a seguir.

- Topologia da rede: [3,10,1].
- Número de repetições do processo de treinamento: 20.
- Tamanho das populações: $n_p = 100$.
- Número de épocas por repetição: $n_e = 40$.

Os valores médios obtidos para o erro da rede e para a diversidade da população durante a fase de treinamento são mostrados nos gráficos que seguem, sendo o valor médio do erro exibido no gráfico

da Figura 4.16, no qual é possível perceber, que novamente, o treinamento utilizando otimização por enxame de partículas é superior em convergência ao algoritmo genético, confirmando o resultado obtido para o problema 1.

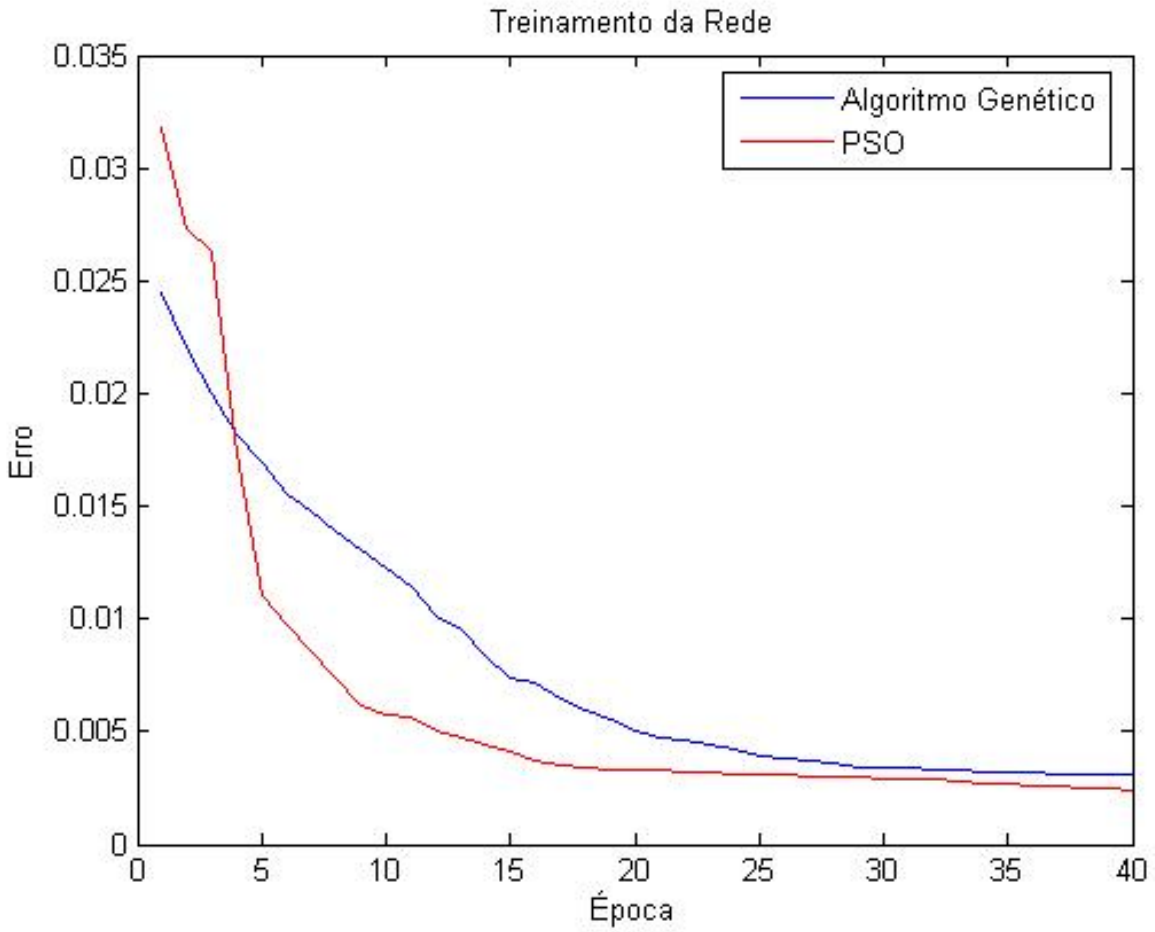

Figura 4.16: Erro médio durante o treinamento da rede, AG $\times$ PSO - Problema 2

Neste problema, a diferença entre os dois métodos é bem menor do que aquela observada anteriormente, ocorrendo ainda uma inversão na quinta época, até onde o algoritmo genético obteve resultados superiores. Esse resultado apresenta algumas diferenças com relação ao resultado anterior, no qual o método PSO foi absolutamente superior.

Uma possível razão para a diminuição da diferença entre os resultados obtidos para os dois algoritmos de treinamento em questão pode ser a alta preservação da diversidade apresentada pelo algoritmo genético. Essa diversidade proporciona ao método uma probabilidade maior de encontrar o ótimo global da função de erro, por outro lado, o método PSO apresenta uma diversidade muito

pequena, que por sua vez, aumenta a probabilidade do método ficar preso a um mínimo local dessa função.

A Figura 4.17 mostra o resultado médio obtido para a diversidade da população durante o processo de treinamento, onde fica clara a alta superioridade do algoritmo genético com relação a este parâmetro.

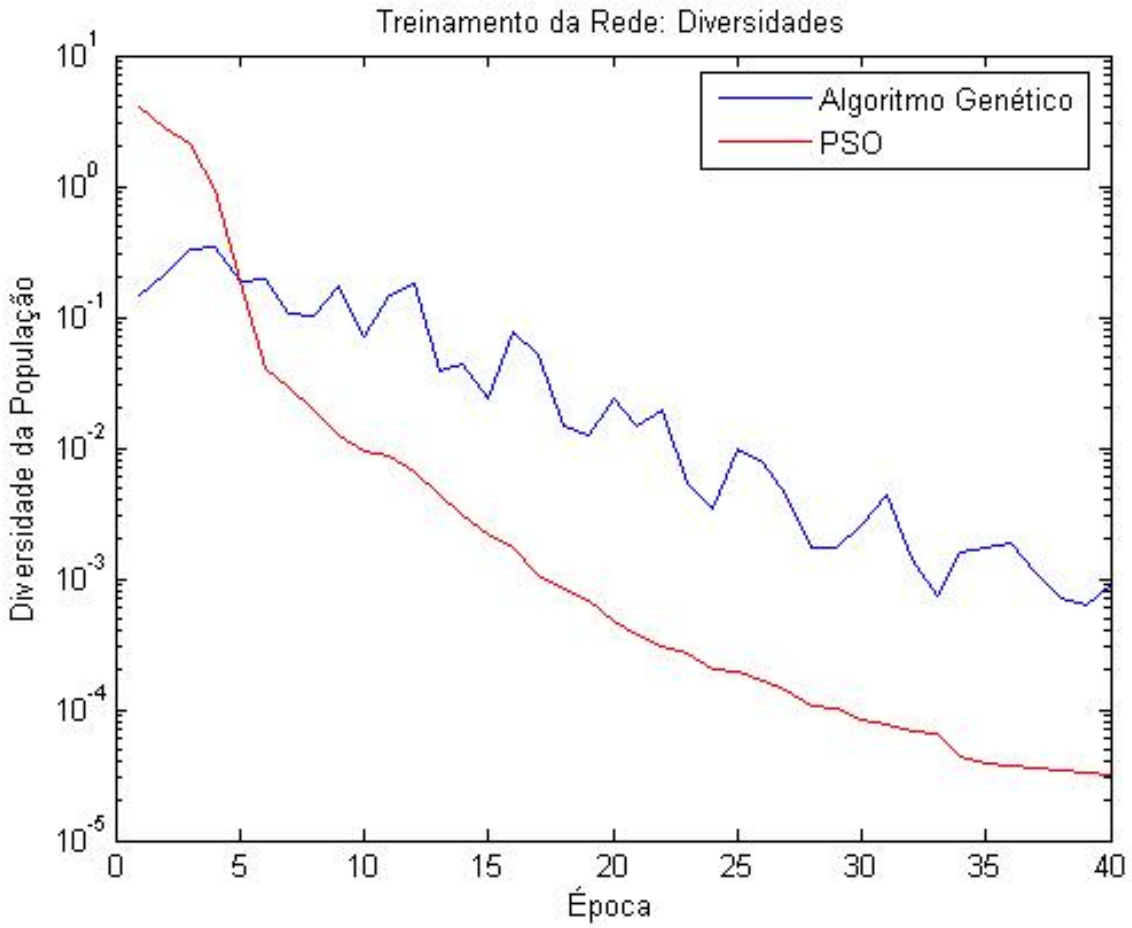

Figura 4.17: Diversidade média da população, AG $\times$ PSO - Problema 2

Outros testes foram realizados utilizando os três problemas propostos, os resultados obtidos para o problema 3 são basicamente intermediários entre os resultados anteriormente discutidos para os problemas 1 e 2. Por não acrescentar nenhuma informação significativa ao estudo em questão, detalhes da análise desse problema serão aqui omitidos.

A Tabela 4.2 resume os resultados obtidos para vários testes utilizando topologias e tamanhos populacionais diferentes, com os parâmetros:

- Número de repetições do processo de treinamento: 50.

- Número de épocas por repetição: $n_e = 40$.

Os resultados mostrados são referentes ao valor médio do erro apresentado pela rede MLP e a variância desse erro σ, quando essa rede, já treinada, é aplicada aos dados do seu conjunto de treinamento.

Parâmetros			Treinamento por algoritmo genético		Treinamento por enxame de partículas	
Problema	Topologia	n_p	média	σ	média	σ
Problema 1	$[1, 3, 1]$	63	0.004177	1.614e-006	0.0001321	1.173e-007
Problema 1	$[1, 18, 1]$	148	0.004517	8.376e-007	0.0001039	6.158e-009
Problema 1	$[1, 3, 3, 1]$	94	0.001782	2.055e-006	0.0001291	1.326e-008
Problema 2	$[3, 3, 1]$	80	0.003287	1.06e-007	0.001324	3.112e-006
Problema 2	$[3, 18, 1]$	191	0.003077	1.126e-007	0.000976	6.2e-007
Problema 2	$[3, 3, 3, 1]$	106	0.003063	1.109e-007	0.001191	1.019e-006
Problema 3	$[10, 3, 1]$	122	0.02697	9.199e-006	0.01295	4.038e-006
Problema 3	$[10, 12, 1]$	241	0.02419	3.924e-006	0.01014	1.473e-006
Problema 3	$[10, 3, 3, 1]$	140	0.0285	1.82e-005	0.01474	2.187e-005

Tabela 4.2: Comparação entre o treinamento utilizando algoritmo genético e PSO

Os resultados obtidos para o treinamento via otimização por enxame de partículas são bastante superiores àqueles obtidos com a utilização de algoritmos genéticos, conclusão essa que pode ser observada na Tabela 4.2, dada a significativa diferença presente nos resultados.

4.6.2 Gradiente Evolutivo $\times$ Gradiente das Partículas

Dado o resultado dos testes realizados anteriormente, espera-se aqui que a superioridade da otimização por enxame de partículas reflita em uma superioridade do algoritmo gradiente das partículas. A fim de avaliar se essa expectativa é justificada, serão novamente considerados alguns exemplos de aplicação de ambos os algoritmos no treinamento de redes MLP para solução dos três problemas em questão.

Em uma primeira análise consideremos o problema 1, ao qual são aplicados os algoritmos de treinamento em questão com o seguinte conjunto de parâmetros:

- Topologia da rede: [1,3,1].
- Número de repetições do processo de treinamento: 20.

- Tamanho das populações: $n_p = 63$.
- Número de épocas por repetição: $n_e = 5$.
- Número de iterações por época: $n_i = n_g = 30$.

Os resultados médios obtidos durante a fase de treinamento da rede são mostrados nos gráficos descritos nas Figuras 4.18 e 4.19

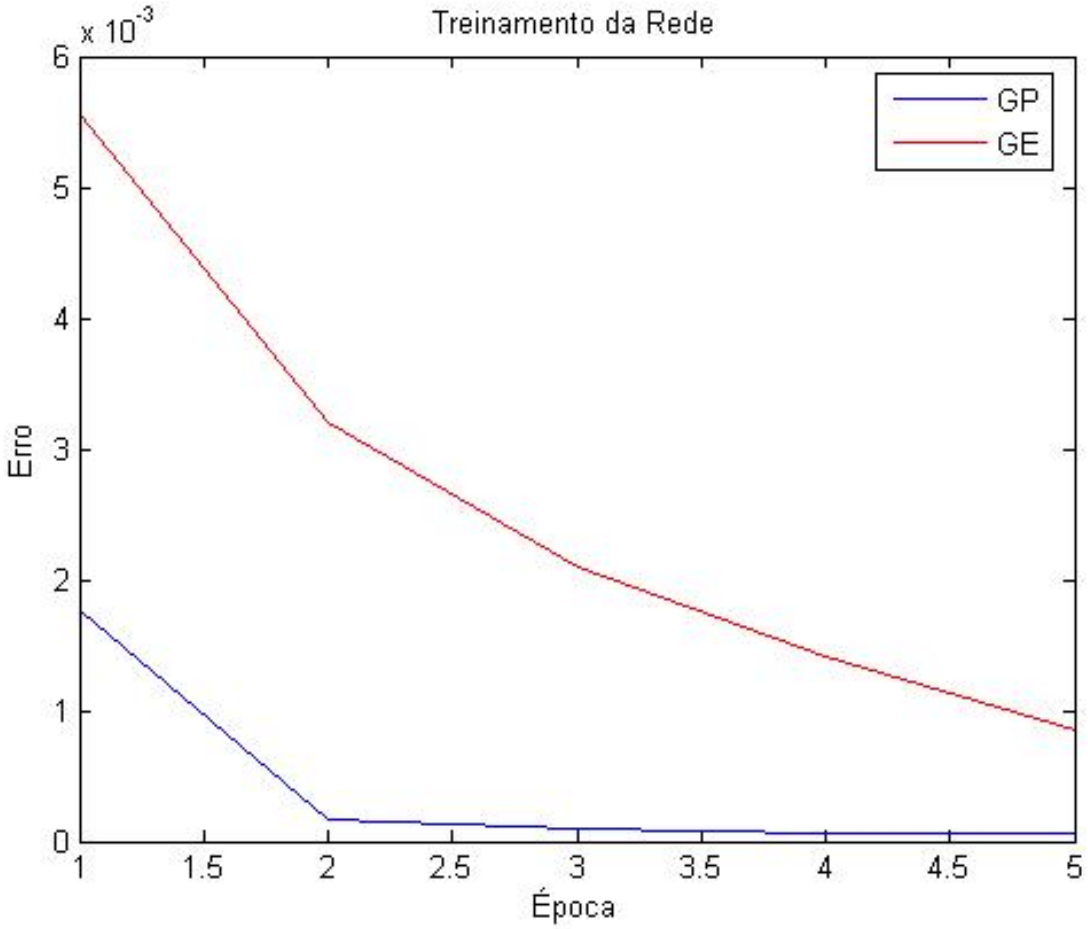

Figura 4.18: Erro médio durante o treinamento da rede, GP $\times$ GE - Problema 1

O resultado médio obtido para o erro da rede durante a fase de treinamento demonstra uma grande superioridade do algoritmo gradiente das partículas com relação ao gradiente evolutivo para o problema em questão, o que confirma as expectativas sustentadas pela maior eficiência demonstrada pelo método PSO com relação ao algoritmo genético.

Os algoritmos gradiente evolutivo e gradiente das partículas reiniciam o processo iterativo de busca do gradiente a cada época, com isso, a diversidade da população em ambos os métodos apresenta uma preservação maior, pois esta é mais elevada no inicio do processo iterativo. Essa característica pode ser observada no gráfico da figura 4.19, onde contrariando as expectativas, o algoritmo gradiente das partículas apresenta um preservação da diversidade consideravelmente maior, diferente do que acontece com o algoritmo PSO, que, apesar da boa taxa de convergência, não mantém uma diversidade muito elevada na população, o que proporciona uma maior probabilidade do método se prender à um mínimo local.

Para o problema em questão, o algoritmo gradiente das partículas demonstra uma taxa de convergência bastante elevada, aliada a uma grande preservação da diversidade. Essas duas características unidas proporcionam uma grande eficiência ao algoritmo em alcançar seu objetivo.

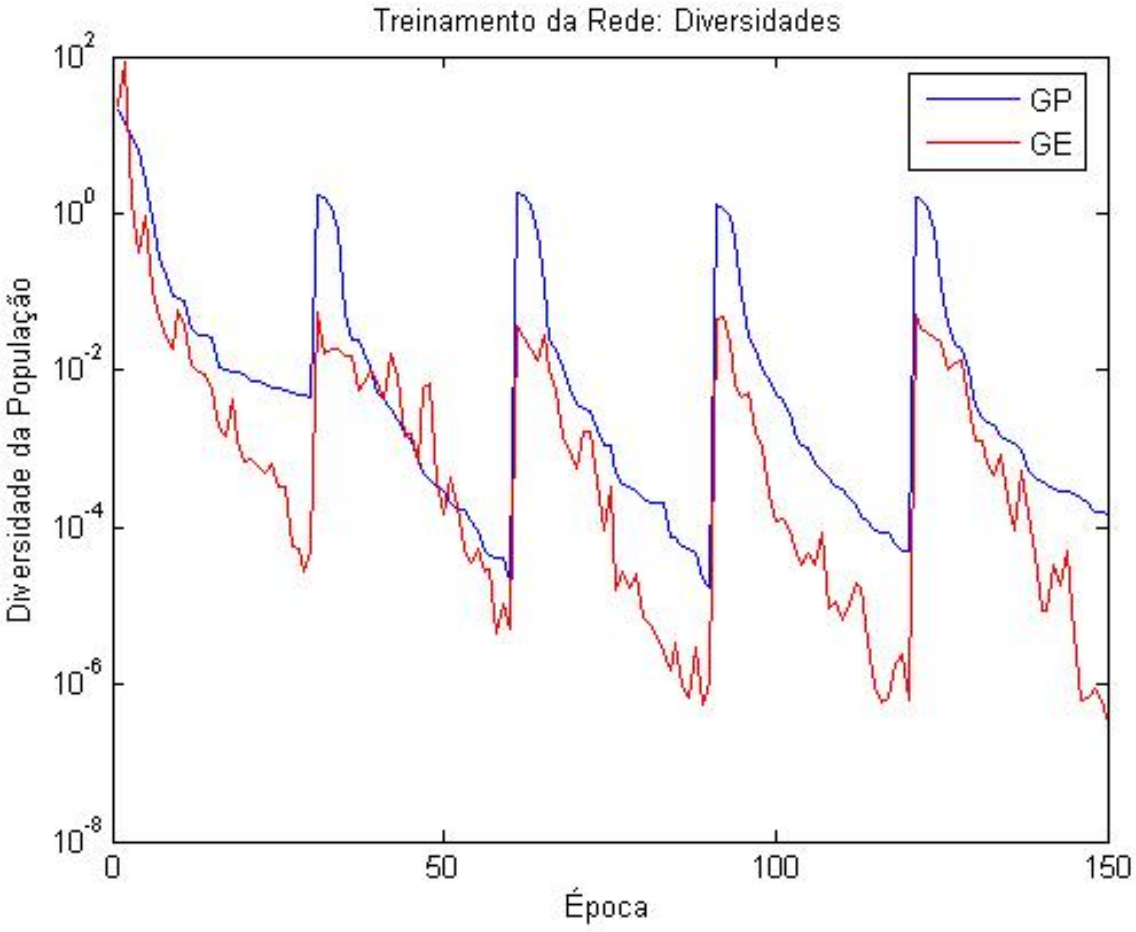

Figura 4.19: Diversidade média da população, GP $\times$ GE - Problema 1

A fim de avaliar o comportamento dos algoritmos em questão para outro cenário de aplicação, consideremos agora o problema 2, utilizando no treinamento o seguinte conjunto de parâmetros:

- Topologia da rede: [3,10,1].
- Número de repetições do processo de treinamento: 20.
- Tamanho das populações: $n_p = 143$.
- Número de épocas por repetição: $n_e = 5$.
- Número de iterações por época: $n_i = n_g = 30$.

O erro médio obtido durante a fase de treinamento da rede neural é mostrado na Figura 4.20.

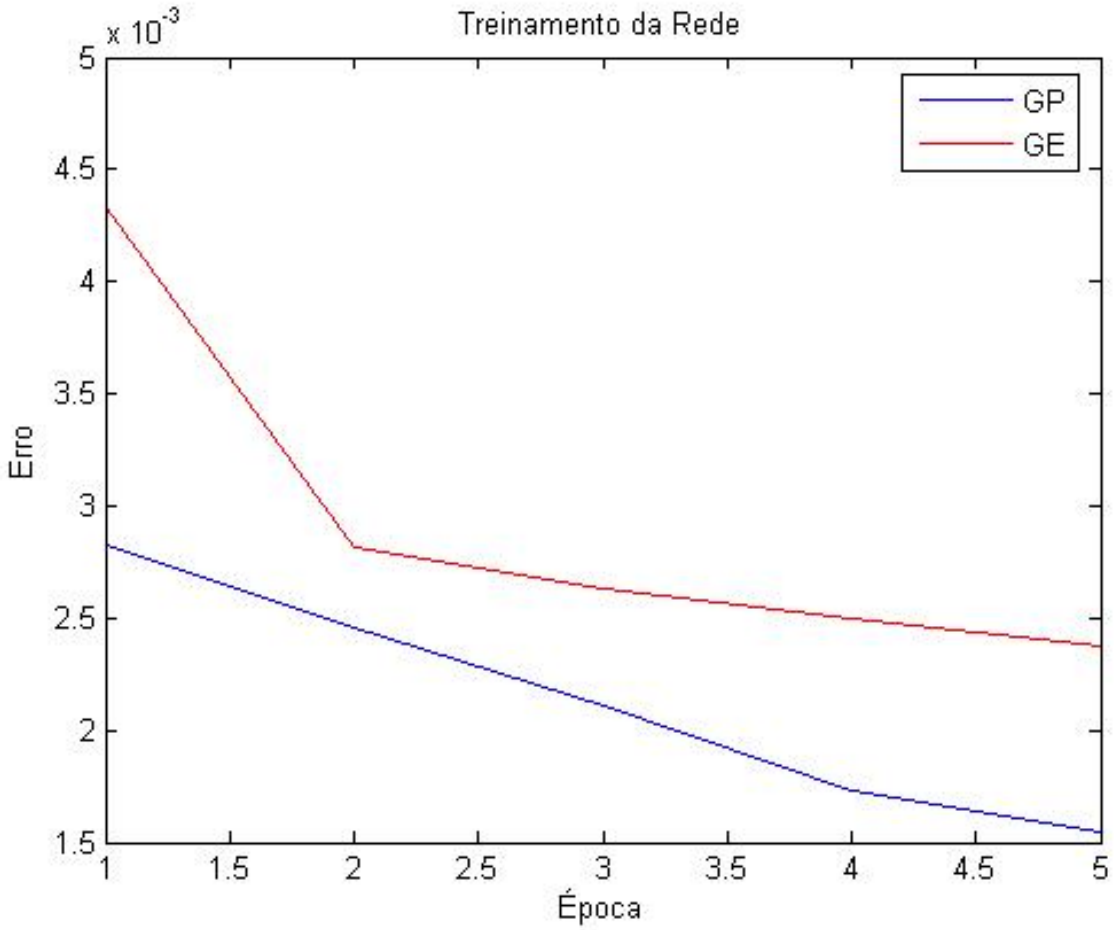

Figura 4.20: Erro médio durante o treinamento da rede, GP $\times$ GE - Problema 2

Novamente, é obtido um resultado semelhante àquele observado para o problema 1, onde a eficiência do algoritmo gradiente das partículas é consideravelmente superior em relação à taxa de convergência. Resultado semelhante também é observado quando é analisada a diversidade da população durante o treinamento da rede para este problema, Figura 4.21, onde existe uma grande diferença entre os valores encontrados para os dois algoritmos analisados.

Resultados semelhantes aos analisados até o momento para os problemas 1 e 2 são obtidos também para o problema 3, que novamente não acrescenta nenhuma informação relevante à presente análise. Essa repetibilidade dos resultados leva à conclusão clara de que o algoritmo gradiente das partículas é mais eficiente do que o algoritmo gradiente evolutivo, sendo superior tanto em taxa de convergência quanto em preservação da diversidade da população, para os cenários de aplicação analisados até então.

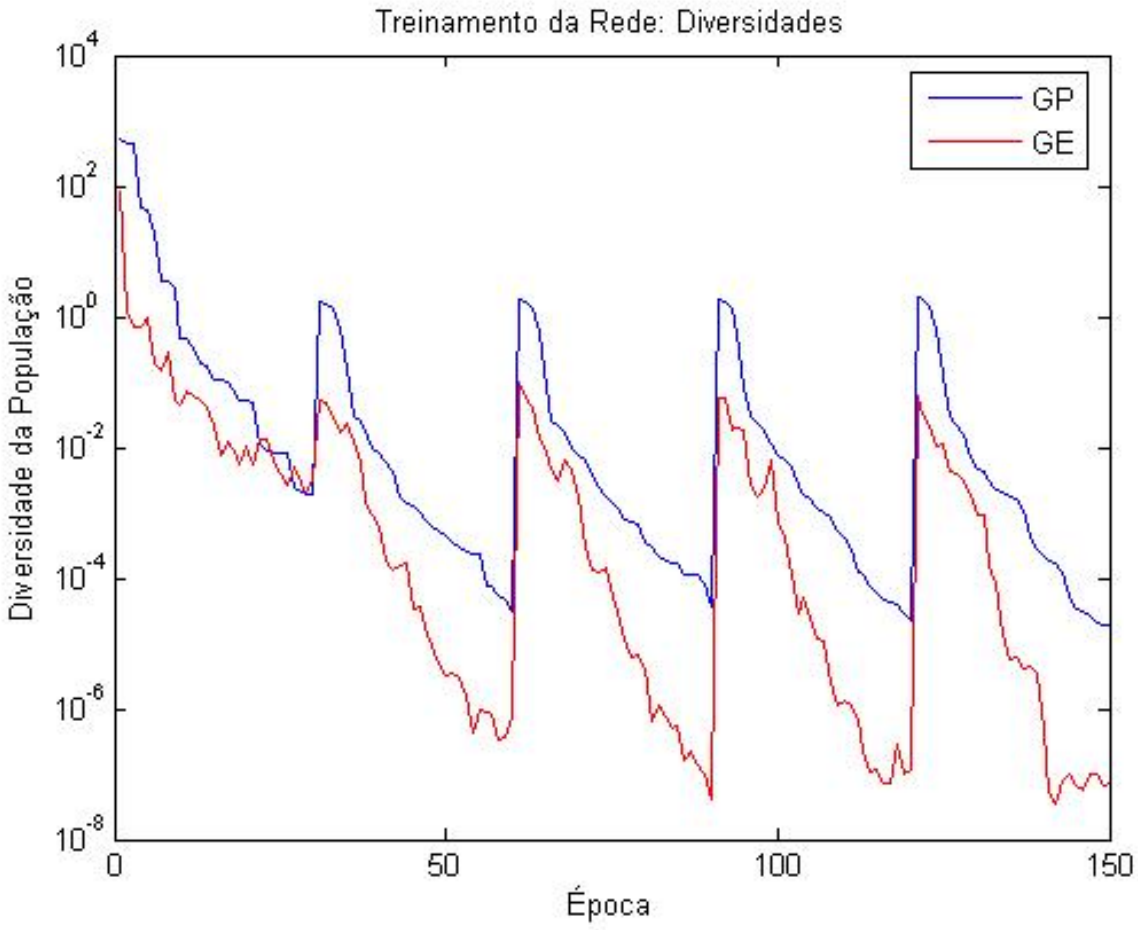

Figura 4.21: Diversidade média da população, GP × GE - Problema 2

Por fim, é apresentado nas Tabelas 4.3 e 4.4 um resumo dos resultados obtidos para vários testes realizados com base nos três problemas em questão, primeiramente utilizando os seguintes parâmetros:

- Número de repetições do processo de treinamento: 20.
- Número de épocas por repetição: $n_e = 5$.
- Número de iterações por época: $n_i = n_g = 30$.

Na Tabela 4.3, são mostrados os valores médios e as variâncias do erro apresentado pela rede, após treinada, utilizando algoritmos com o conjunto de parâmetros em questão.

Parâmetros			Gradiente Evolutivo		Gradiente das Partículas	
Problema	Topologia	n_p	média	σ	média	σ
Problema 1	$[1, 3, 1]$	63	0.0002388	8.614e-007	3.246e-005	7.332e-010
Problema 1	$[1, 18, 1]$	148	0.0002878	1.589e-006	8.082e-006	1.283e-010
Problema 1	$[1, 3, 3, 1]$	94	0.0001049	1.141e-007	3.011e-005	5.527e-011
Problema 2	$[3, 3, 1]$	80	0.0003444	4.164e-007	0.0002408	1.747e-007
Problema 2	$[3, 18, 1]$	191	0.0003445	4.342e-007	0.0002423	1.536e-007
Problema 2	$[3, 3, 3, 1]$	106	0.0002946	4.347e-007	0.0001493	6.954e-010
Problema 3	$[10, 3, 1]$	122	0.01829	7.341e-007	0.00816	1.554e-007
Problema 3	$[10, 12, 1]$	241	0.01596	1.781e-005	0.00534	5.925e-006
Problema 3	$[10, 3, 3, 1]$	140	0.01713	4.45e-006	0.00675	6.759e-007

Tabela 4.3: Comparação entre o treinamento utilizando gradiente evolutivo e gradiente das partículas

Consideremos agora um treinamento com um maior número de épocas e também um maior número de iterações por época, caracterizado pelos seguintes parâmetros:

- Número de repetições do processo de treinamento: 20.
- Número de épocas por repetição: $n_e = 20$.
- Número de iterações por época: $n_i = n_g = 80$.

Os resultados desse treinamento são mostrados na Tabela 4.4, seguindo a mesma estrutura da tabela anterior.

Parâmetros			Gradiente Evolutivo		Gradiente das Partículas	
Problema 1	$[1, 3, 1]$	63	4.216e-006	7.495e-011	2.451e-006	1.535e-011
Problema 1	$[1, 18, 1]$	148	9.146e-007	6.86e-012	3.226e-007	8.033e-014
Problema 1	$[1, 3, 3, 1]$	94	3.787e-006	1.215e-010	1.336e-006	3.291e-012
Problema 2	$[3, 3, 1]$	80	9.671e-005	9.747e-010	8.994e-005	1.33e-011
Problema 2	$[3, 18, 1]$	191	4.022e-005	7.011e-010	3.452e-005	7.926e-011
Problema 2	$[3, 3, 3, 1]$	106	8.479e-005	1.508e-010	8.234e-005	2.177e-011
Problema 3	$[10, 3, 1]$	122	0.01253	2.92e-006	0.01219	8.143e-007
Problema 3	$[10, 12, 1]$	241	0.009022	2.422e-006	0.008722	8.323e-007
Problema 3	$[10, 3, 3, 1]$	140	0.01248	5.245e-007	0.01233	9.821e-008

Tabela 4.4: Comparação entre o treinamento utilizando gradiente evolutivo e gradiente das partículas

A Tabela 4.3 expressa uma sensível diferença entre os resultados obtidos pelos dois algoritmos em questão, sendo suficiente para concluir a superioridade do algoritmo gradiente das partículas no cenário de aplicação em questão. A Tabela 4.4 por outro lado, apresenta uma semelhança maior entre os dados nela apresentados, quando comparados os dois algoritmos, fazendo-se necessária a aplicação de um teste de hipóteses resumido na Tabela 4.5, a fim de se inferir sobre qualquer diferença existente nesses resultados.

Parâmetros			Gradiente Evolutivo	Gradiente das Partículas	Erro Tipo 1 %
Problema 1	$[1, 3, 1]$	63	4.216e-006	2.451e-006	6
Problema 1	$[1, 18, 1]$	148	9.146e-007	3.226e-007	2
Problema 1	$[1, 3, 3, 1]$	94	3.787e-006	1.336e-006	2
Problema 2	$[3, 3, 1]$	80	9.671e-005	8.994e-005	3
Problema 2	$[3, 18, 1]$	191	4.022e-005	3.452e-005	4
Problema 2	$[3, 3, 3, 1]$	106	8.479e-005	8.234e-005	6
Problema 3	$[10, 3, 1]$	122	0.01253	0.01219	8
Problema 3	$[10, 12, 1]$	241	0.009022	0.008722	9
Problema 3	$[10, 3, 3, 1]$	140	0.01248	0.01233	5

Tabela 4.5: Resultado do teste de hipótese com relação aos dados da tabela 4.4

A Tabela 4.5 mostra que o erro cometido em se assumir que existe diferença entre os resultados médios obtidos é pequeno, logo pode-se afirmar, a menos de uma pequena margem de erro, que o algoritmo gradiente das partículas, obteve resultados superiores àqueles obtidos pelo algoritmo

gradiente evolutivo também para esse cenário de aplicação, no qual foram realizadas mais épocas e iterações.

4.6.3 Evolução do Gradiente $\times$ Evolução do Vetor de Parâmetros

A fim de comparar a eficiência dos métodos de treinamento utilizando a evolução do gradiente e a evolução do vetor de parâmetros são utilizados os algoritmos gradiente das partículas e PSO, por representarem as melhores soluções dentre as estudadas neste documento para essas duas áreas. Em uma primeira análise, consideremos o problema 1, utilizando para seu treinamento os algoritmos com os seguintes conjuntos de parâmetros:

Gradiente das partículas:

- Topologia da rede: [1,3,1].
- Número de repetições do processo de treinamento: 20.
- Tamanho das populações: $n_p = 63$.
- Número de épocas por repetição: $n_e = 5$.
- Número de iterações por época: $n_i = 150$.

PSO:

- Topologia da rede: [1,3,1].
- Número de repetições do processo de treinamento: 20.
- Tamanho das populações: $n_p = 63$.
- Número de épocas por repetição: $n_e = 750$.

Nesta análise, foi utilizado um número de épocas para o algoritmo PSO $n_e = 5 \cdot 150$, a fim de proporcionar a mesma quantidade de iterações observada para o algoritmo gradiente das partículas. A Figura 4.22 ilustra o erro da rede, obtido por ambos os métodos durante o processo de treinamento, sendo que para o algoritmo gradiente das partículas são plotados os erros intermediários obtidos a cada iteração do processo de busca do vetor gradiente.

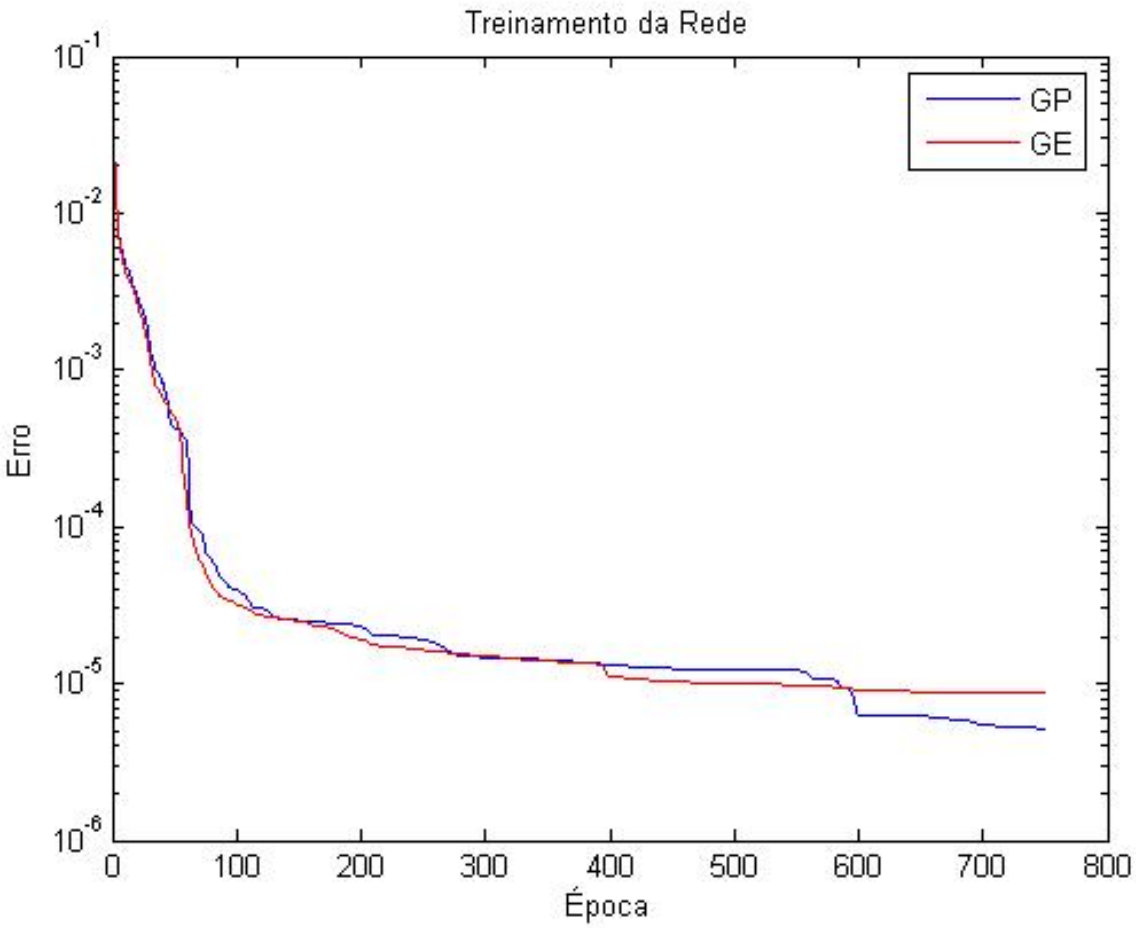

Figura 4.22: Erro médio durante o treinamento da rede, GP $\times$ PSO, Problema 1

É importante salientar que ambos os algoritmos, GP e PSO, utilizam a mesma meta-heurística, com a mesma codificação e os mesmos parâmetros, exceto pela função de custo, o que lhes proporciona exatamente a mesma capacidade de busca. Porém, nota-se claramente na Figura 4.22 que o algoritmo GP obteve um resultado superior em taxa de convergência com relação ao algoritmo PSO, sendo essa diferença clara ao final do processo iterativo. Muitos fatores podem levar a esse resultado, como o fato do método GP explorar o espaço dos gradientes, o que é geometricamente favorável, pois sabe-se *a priori* que boas soluções podem ser encontradas próximas a origem.

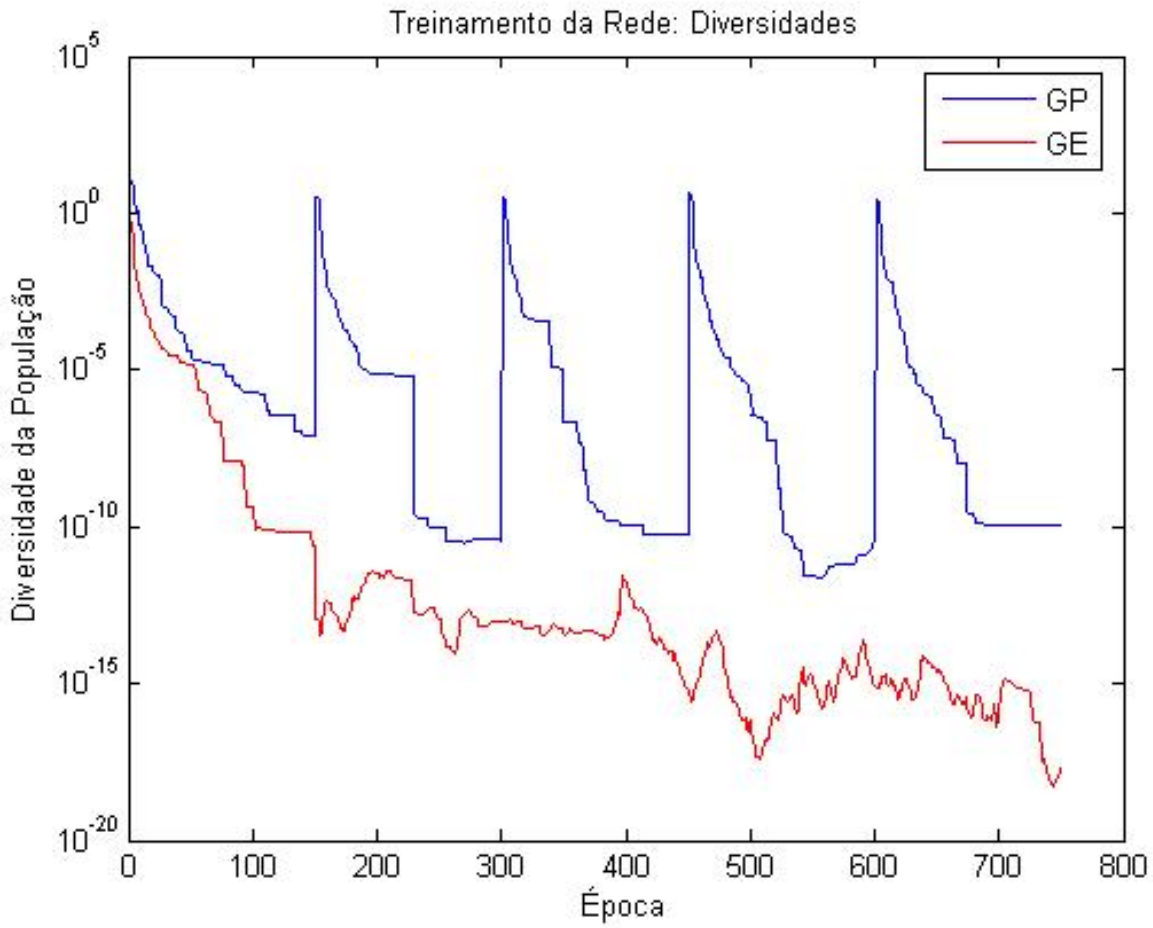

Figura 4.23: Diversidade média da população, GP $\times$ PSO - Problema 1

Outra característica importante à essa análise, que contribuí para a diferença observada entre os dois algoritmos com relação a taxa de convergência, é a preservação da diversidade. Essa característica é ilustrada na Figura 4.23, na qual são plotadas as diversidades médias para ambos os algoritmos de treinamento durante o processo iterativo.

Nota-se claramente a superioridade do algoritmo gradiente das partículas com relação à preservação da diversidade, o que pode ser atribuído ao fato desse reiniciar a população a cada época, retomando com isso a diversidade inicial, que é bastante elevada. Essa característica do algoritmo GP lhe confere, para o problema em questão, uma maior capacidade exploratória, mesmo após várias iterações, criando um cenário favorável à evolução do método, o que resulta na superior taxa de convergência observada.

Outra análise semelhante é feita com relação ao problema 2, que é relativamente mais complexo que o anterior. Para tanto, é utilizada a seguinte configuração para a rede neural e para os algoritmos de treinamento.

Gradiente das partículas:

- Topologia da rede: [3,9,1].
- Número de repetições do processo de treinamento: 20.
- Tamanho das populações: $n_p = 136$.
- Número de épocas por repetição: $n_e = 5$.
- Número de iterações por época: $n_i = 30$.

PSO:

- Topologia da rede: [3,9,1].
- Número de repetições do processo de treinamento: 20.
- Tamanho das populações: $n_p = 136$.
- Número de épocas por repetição: $n_e = 150$.

Os resultados médios obtidos para o erro da rede são mostrados na Figura 4.24, onde novamente são plotados erros de iterações intermediárias para o algoritmo gradiente das partículas.

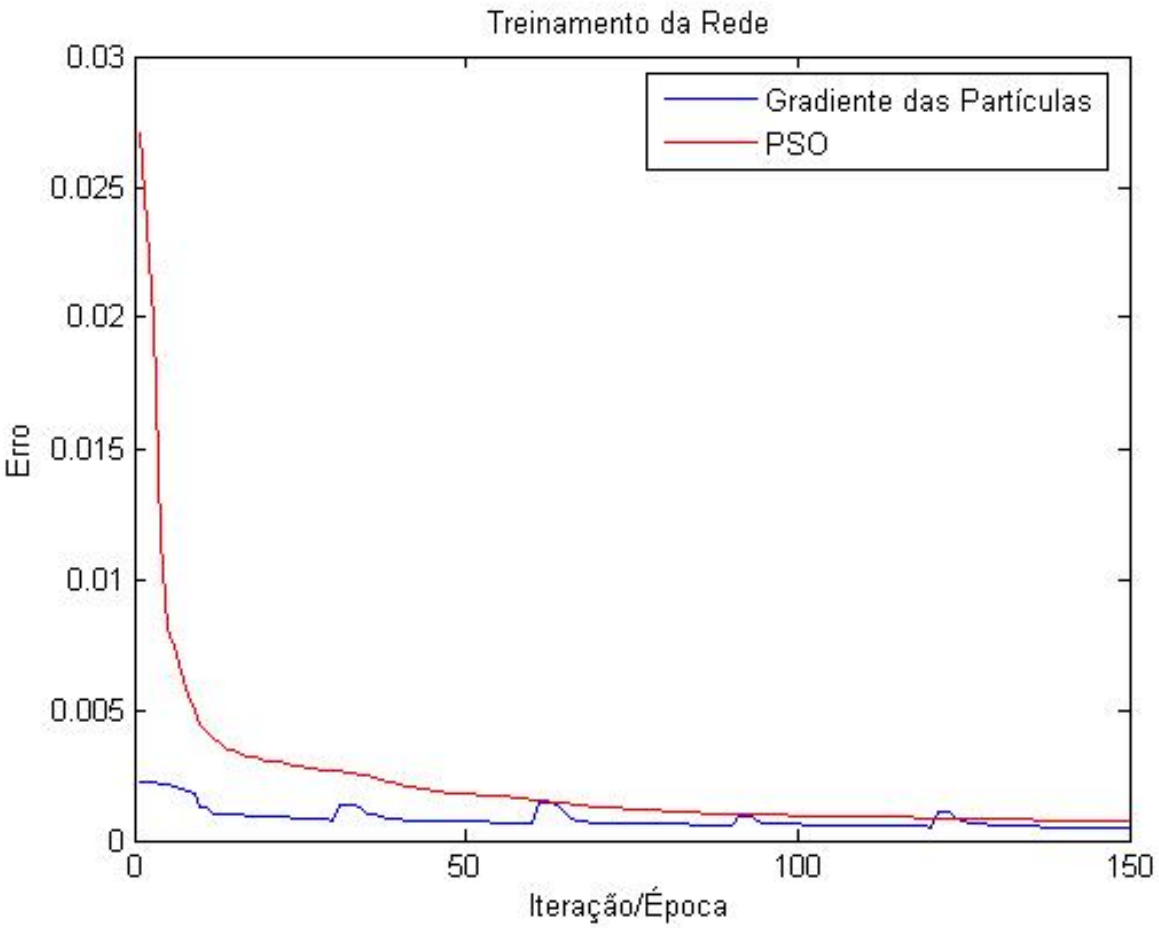

Figura 4.24: Erro médio durante o treinamento da rede, GP $\times$ PSO - Problema 2

O resultado obtido novamente demonstra uma taxa de convergência maior do algoritmo gradiente das partículas no início do processo de treinamento. Porém, desta vez nota-se que o erro para esse algoritmo cresce em algumas iterações intermediárias, o que ocorre devido ao fato da condição inicial da população, que contém o gradiente do erro, $grad^0$, não possuir nenhuma partícula que represente um vetor de parâmetros (pesos e polarizações) capaz de minimizar o erro da rede.

Esse aumento momentâneo do erro da rede tende a ser suprimido antes do final da busca pelo vetor gradiente a cada época do processo iterativo. Porém, incrementos que aumentem o erro da rede são suprimidos no algoritmo principal, a fim de evitar instabilidades no processo. É possível também inserir na população inicial a origem do espaço R^N, o que garante a existência de pelo menos uma partícula que leve a um erro igual ao erro atual da rede, de forma tal que minimamente o algoritmo não irá aumentar o erro. É possível ainda inserir na população inicial vários vetores colineares ao vetor gradiente e de mesmo sentido que este, com isso, consegue-se uma maior probabilidade de se obter uma condição inicial satisfatória para as partículas.

A diversidade das populações durante o processo de treinamento em questão pode ser verificada na Figura 4.25.

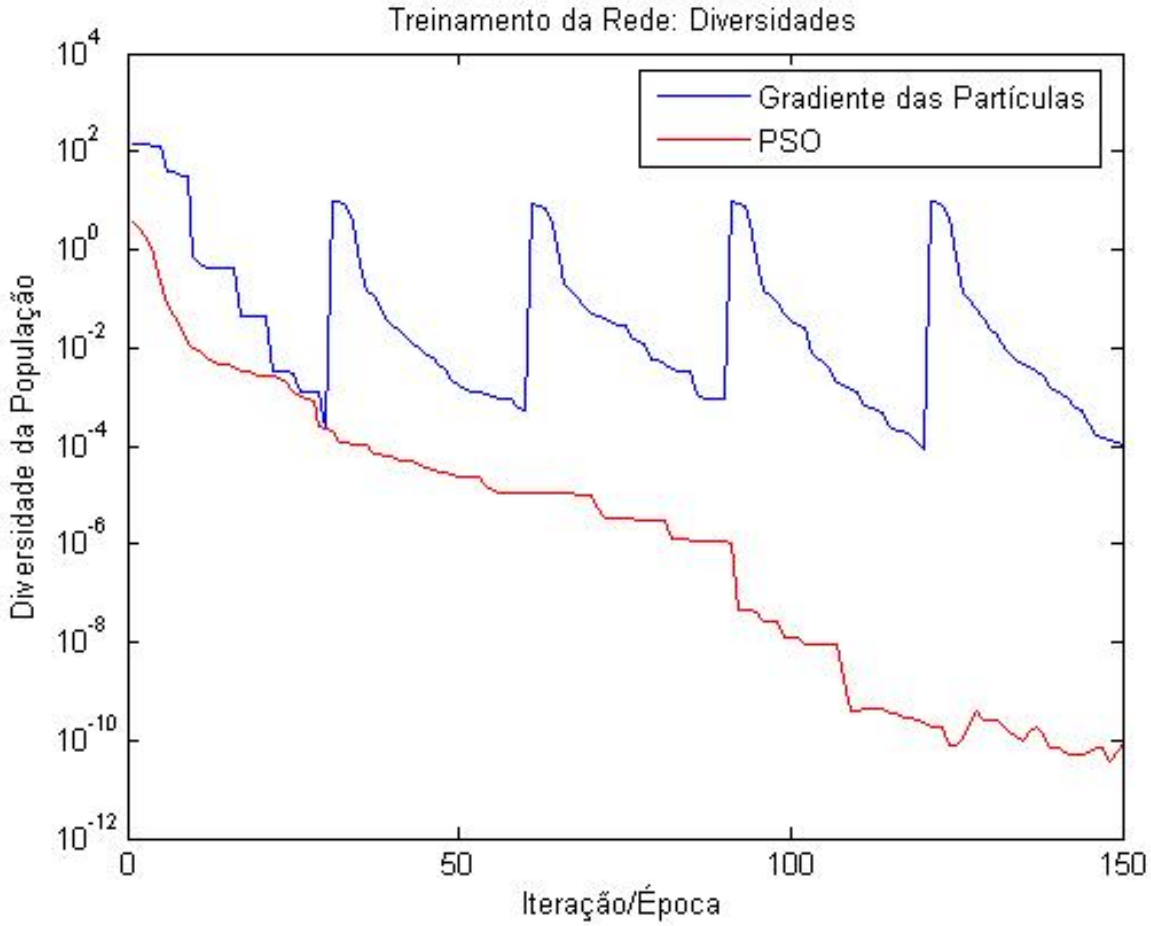

Figura 4.25: Diversidade média da população, GP $\times$ PSO - Problema 2

Novamente, observa-se aqui uma superioridade do algoritmo gradiente das partículas em relação à preservação da diversidade da população.

Uma série do outros testes é resumida na Tabela 4.6, que ilustra o valor médio do erro obtido pela rede para o conjunto de treinamento após ter sido treinada e a variância desse erro. O conjunto de parâmetros utilizado nestes testes foi o seguinte:

- Número de repetições do processo de treinamento: 20.
- Número de épocas por repetição: PSO:$n_e = 150$, GP:$n_e = 5$
- Número de iteração por época: $n_i = 30$.

Parâmetros			Gradiente das Partículas		PSO	
Problema	Topologia	n_p	média	σ	média	σ
Problema 1	$[1, 3, 1]$	63	3.246e-005	7.332e-010	0.0001321	1.173e-007
Problema 1	$[1, 18, 1]$	148	8.082e-006	1.283e-010	0.0001039	6.158e-009
Problema 1	$[1, 3, 3, 1]$	94	3.011e-005	5.527e-011	0.0001291	1.326e-008
Problema 2	$[3, 3, 1]$	80	0.0002408	1.747e-007	0.001324	3.112e-006
Problema 2	$[3, 18, 1]$	191	0.0002423	1.536e-007	0.000976	6.2e-007
Problema 2	$[3, 3, 3, 1]$	106	0.0001493	6.954e-010	0.001191	1.019e-006
Problema 3	$[10, 3, 1]$	122	0.00816	1.554e-007	0.01295	4.038e-006
Problema 3	$[10, 12, 1]$	241	0.00534	5.925e-006	0.01014	1.473e-006
Problema 3	$[10, 3, 3, 1]$	140	0.00675	6.759e-007	0.01474	2.187e-005

Tabela 4.6: Comparação entre o treinamento utilizando algoritmos de busca do gradiente e PSO

Os resultados descritos na Tabela 4.6 mostram que o algoritmo gradiente das partículas obteve erros menores do que aqueles obtidos pelo algoritmo PSO. Como ambos os algoritmos são construídos com base na mesma heurística, a otimização por enxame de partículas, e em ambos foi implementado o método de modulação de velocidade, o que provê o mesmo aumento de performance nos dois algoritmos, pode-se concluir que a abordagem do algoritmo gradiente das partículas, ou seja, a evolução do gradiente, gera resultados superiores àqueles observados quando é evoluída a solução diretamente. Esses resultados podem ser gerados por dois fatores, sendo que o primeiro deles diz respeito à reinicialização da população a cada época, o que gera um aumento da dispersão das partículas e pode favorecer o processo de busca. Porém, é possível ainda que a busca pelo espaço $\mathcal{W}$, com: $\omega \in \mathcal{W}$, apresente um desafio maior para o método do que a busca pelo espaço $\mathcal{G}$, onde: $grad \in \mathcal{G}$.

4.6.4 Algoritmos de Treinamento Procedurais $\times$ Heurísticos

Em situações onde os dados apresentam baixo nível de ruido e sua relação é bem definida, assim como o erro da rede neural e o gradiente desse erro, os algoritmos de treinamento procedurais apresentam uma grande vantagem em taxa de convergência com relação aos algoritmos heurísticos. Isso se deve basicamente a melhor exploração da regularidade do problema por parte desses algoritmos. Por outro lado, algoritmos heurísticos, têm uma capacidade maior em lidar com ruídos e mal condicionamento, além de uma maior probabilidade de encontrar o ótimo global da função de erro da rede. A fim de avaliar a eficiência relativa entre os métodos de treinamento procedurais e heurísticos estudados neste documento, são realizados vários testes comparativos.

Consideremos em uma primeira análise o problema 1. Os parâmetros utilizados para este teste são os seguintes:

- Topologia da rede: [1,3,1].
- Número de repetições do processo de treinamento: 20.
- Número de épocas por repetição: GA, PSO, DFP, BFGS, SCG: $n_e = 800$, GP, GE: $n_e = 10$
- Número de iterações por época: $n_i = 80$.

O erro médio obtido durante o processo de treinamento pelos 8 algoritmos em análise pode ser observado no gráfico da Figura 4.26, que mostra esses valores plotados em uma escala logarítmica para uma melhor visualização. Novamente são aqui plotados erros intermediários para os algoritmos gradiente das partículas e gradiente evolutivo, a fim de expressar o resultado de cada uma das iterações realizadas, tornando viável sua comparação com os algoritmos AG e PSO.

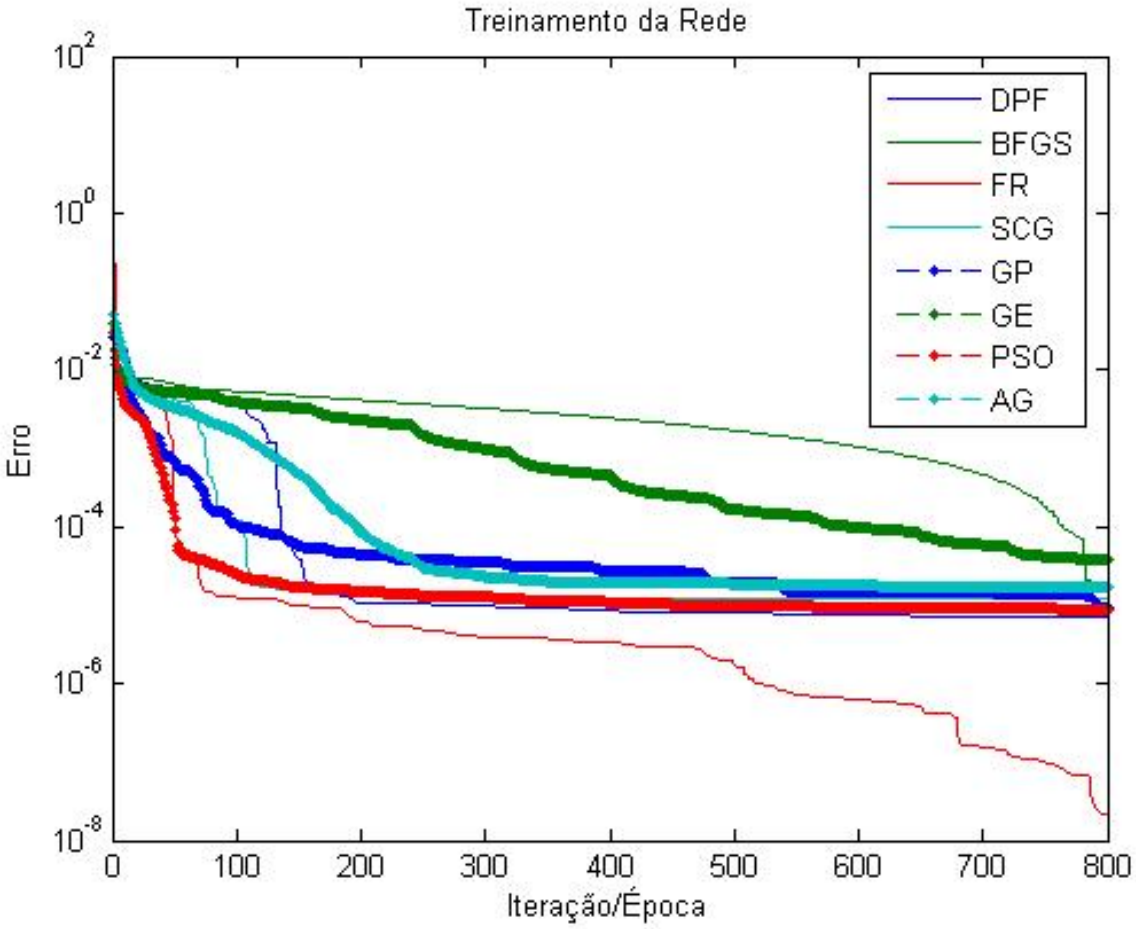

Figura 4.26: Treinamento da rede, comparação geral - Problema 1

Os resultados obtidos demonstram um resultado final bastante superior com a utilização do método de direções conjugadas FR, além de uma superioridade do algoritmos PSO com relação às demais meta-heurísticas.

Esses resultados são frutos de configurações específicas para cada método, onde foram escolhidos parâmetros livres empiricamente. Portanto resultados diferentes podem ser observados para o mesmo problema, como é o caso dos algoritmos GP e PSO, que em uma análise anterior demonstraram um resultado diferente, o que é atribuído a diferença no número de iterações por época utilizado. Isto ocorre porque o algoritmo GP não realizou iterações suficientes para reduzir satisfatoriamente o erro da rede. Diferenças como essas são observadas também quando alterados os parâmetros dos métodos procedurais. Dessa forma, os resultados aqui discutidos não são tomados como referências globais, e sim como comparações pontuais que permitem a sólida construção de uma base de conhecimentos fundamental para o presente trabalho.

Outra análise, agora com relação ao problema 2, utilizando a configuração a seguir, é mostrada na Figura 4.27

- Topologia da rede: [3,9,1].
- Número de repetições do processo de treinamento: 20.
- Número de épocas por repetição: GA, PSO, DFP, BFGS, SCG: $n_e = 800$, GP, GE: $n_e = 10$
- Número de iterações por época: $n_i = 80$.

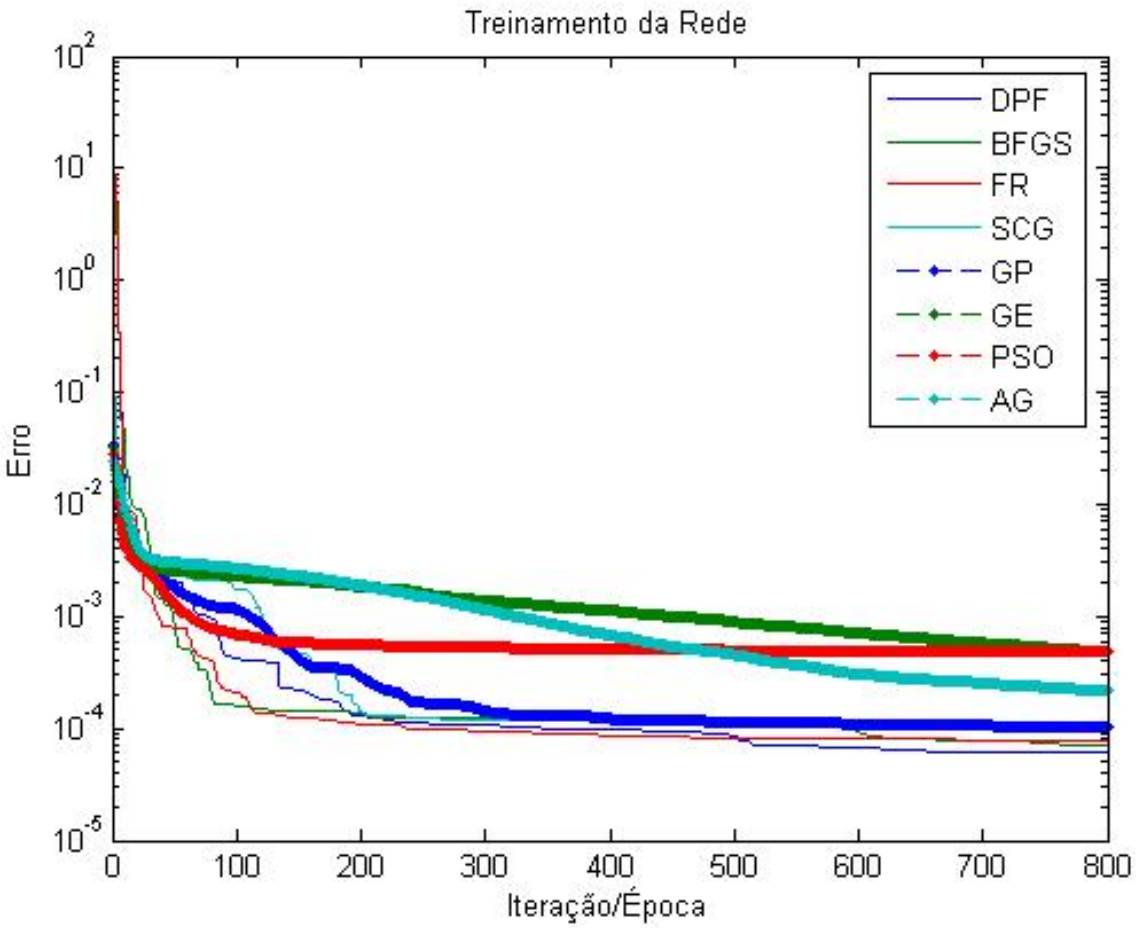

Figura 4.27: Treinamento da rede, comparação geral - Problema 2

Para este problema, são observados novamente resultados superiores com a utilização de métodos procedurais, tendo o algoritmo gradiente das partículas obtido um resultado consideravelmente superior dentre os métodos heurísticos.

Por fim, a aplicação desses algoritmos ao treinamento da rede, utilizando os dados do problema 3, com o conjunto de parâmetros que segue, leva aos resultados descritos na Figura 4.28.

- Topologia da rede: [10,6,1].
- Número de repetições do processo de treinamento: 20.
- Número de épocas por repetição: GA, PSO, DFP, BFGS, SCG: $n_e = 800$, GP, GE: $n_e = 10$
- Número de iterações por época: $n_i = 80$.

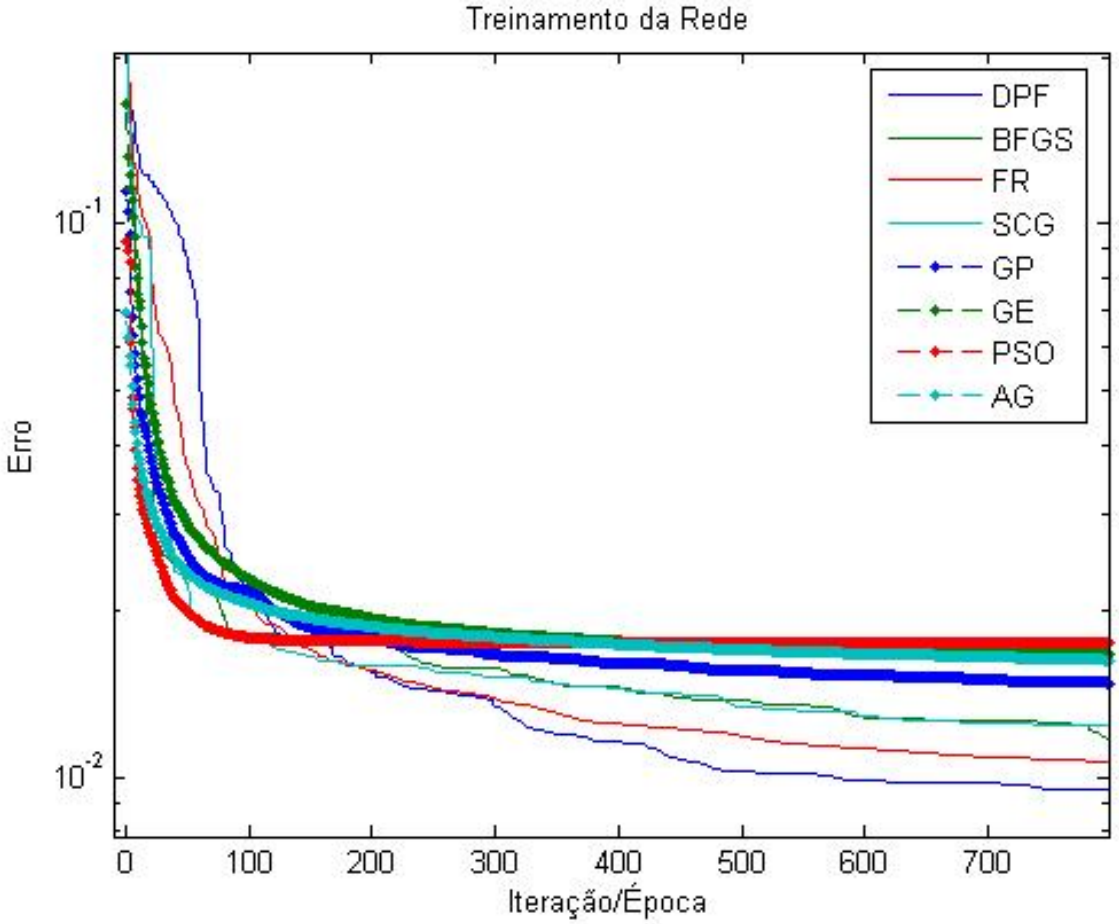

Figura 4.28: Treinamento da rede, comparação geral - Problema 3

Os resultados obtidos para os três problemas em questão comprovam, entre outros fatores, a alta eficiência do algoritmo proposto: gradiente das partículas, com relação as demais meta-heurísticas aqui aplicadas ao treinamento de redes MLP. É evidente também a eficiência superior dos métodos procedurais de treinamento, com relação à taxa de convergência, quando analisada simplesmente a redução do erro por época do treinamento.

As análises realizadas até então não levam em conta o número de operações efetuadas por cada um dos métodos em cada época do processo iterativo, o que interfere diretamente no tempo de processamento e capacidade computacional necessária. Essa análise não foi necessária quando comparados os algoritmos heurísticos, pois esses métodos possuem um custo computacional semelhante[4], dado

[4]Essa semelhança é significativa quando comparados algoritmos heurísticos e procedurais, porém, em uma comparação apenas entre os algoritmos heurísticos, tal semelhança não pode ser considerada devido a diferenças no custo computacional de cada implementação, que vária com a meta-heurística utilizada e com a codificação escolhida, entre outros fatores.

por: $O(n_p N^2)$ para os algoritmos AG e PSO e $N(n_i n_p N^2)$ para os algoritmos GP e GE, o que possibilitou uma escolha do número de épocas em cada um dos métodos de forma a proporcionar um custo computacional comparável para todos os algoritmos. Esse processo, porém, não é aplicável aos algoritmos procedurais, que inevitavelmente possuem custos computacionais diferentes.

Dentre os métodos procedurais estudados, o método SCG, é aquele que apresenta um menor custo computacional ($O(N^2)$) [50] e possui uma boa taxa de convergência, sendo mais *"rápido"* do ponto de vista computacional [50, 57], quando comparado aos métodos DFP, BFGS e FR.

Uma análise dos custos computacionais dos algoritmos SCG e do algoritmo gradiente das partículas indica que o custo total do processo de treinamento utilizando SCG é: $C_t^{SCG} = n_e^{SCG} O(N^2)$ enquanto que utilizando GP é de: $C_t^{GP} = n_e^{GP} O(n_i n_p N^2)$. Supondo que a função $O(\cdot)$ seja linear com relação aos parâmetros [5] n_p e n_i, pode-se escrever:

$$\begin{aligned} C_t^{SCG} &= C_t^{GP} \\ n_e^{SCG} O(N^2) &= n_e^{GP} O(n_i n_p N^2) \\ n_e^{SCG} O(N^2) &= n_e^{GP} n_i n_p O(N^2) \\ n_e^{SCG} &= n_e^{GP} n_i n_p \\ n_e^{GP} &= \frac{n_e^{SCG}}{n_i n_p} \end{aligned} \tag{4.45}$$

Portanto, se for escolhido um número de épocas para o algoritmo SCG: $n_e^{SCG} = n_e^{GP} n_i n_p$, estima-se que ambos os algoritmos apresentem um custo computacional muito semelhante, possibilitando uma análise da eficiência relativa de cada um com relação à taxa de convergência $\times$ tempo de processamento[6]. A Figura 4.29 ilustra uma situação onde o algoritmo gradiente das partículas é aplicado ao problema 2 utilizando a topologia da rede: [3,9,9,1] com uma população de 80 partículas, realizando 80 iterações por época. O erro da rede é plotado em função do tempo de processamento para os métodos de treinamento GP e SCG.

[5] Tal suposição é feita para permitir uma análise simplificada do problema, tendo em vista que considerar a não linearidade dessa função a fim de obter um resultado preciso, não acrescentaria informações significativas a presenta análise.

[6] O tempo de processamento considerado é o tempo em segundos necessário para o algoritmo completar o processo de treinamento em uma máquina digital, sendo apenas uma referência utilizada na análise, não podendo ser considerado como característica de nenhum dos métodos em questão.

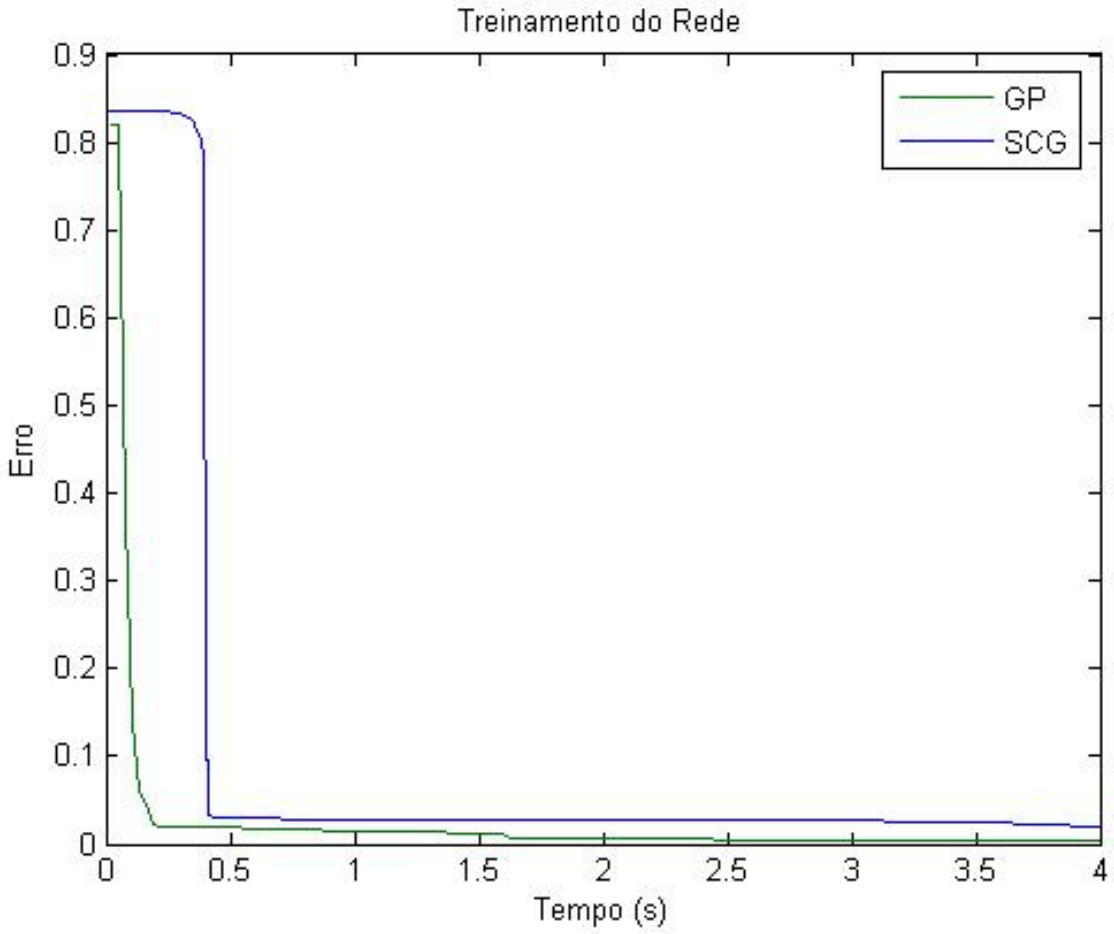

Figura 4.29: Comparação entre tempo de processamento utilizando GP e SCG

Na aplicação mostrada na Figura 4.29, o algoritmo gradiente das partículas foi mais eficiente que o algoritmo procedural SCG, demonstrando a grande capacidade computacional do algoritmo proposto, porém, dada a estocasticidade desse algoritmo, não é possível se afirmar que esse resultado irá se repetir em uma nova aplicação.

Uma outra análise é realizada, ainda com relação ao problema 2, utilizando os seguintes parâmetros:

- Número de épocas: SCG: $n_e^{SCG} = 3200$, GP: $n_e^{GP} = 4$
- Número de iterações por época: $n_i = 60$
- Número de partículas: $n_p = 30$
- Número de repetições: $n_r = 5$

Os resultados obtidos para o erro médio da rede após o trinamento são os seguintes:

- **GP:** erro médio: 0.01137, variância: 2.435e-004
- **SCG:** erro médio: 0.0104, variância: 2.696e-004

Na Figura 4.30, são plotados os erros médios da rede após o treinamento, em cada uma das repetições, para os dois métodos em questão.

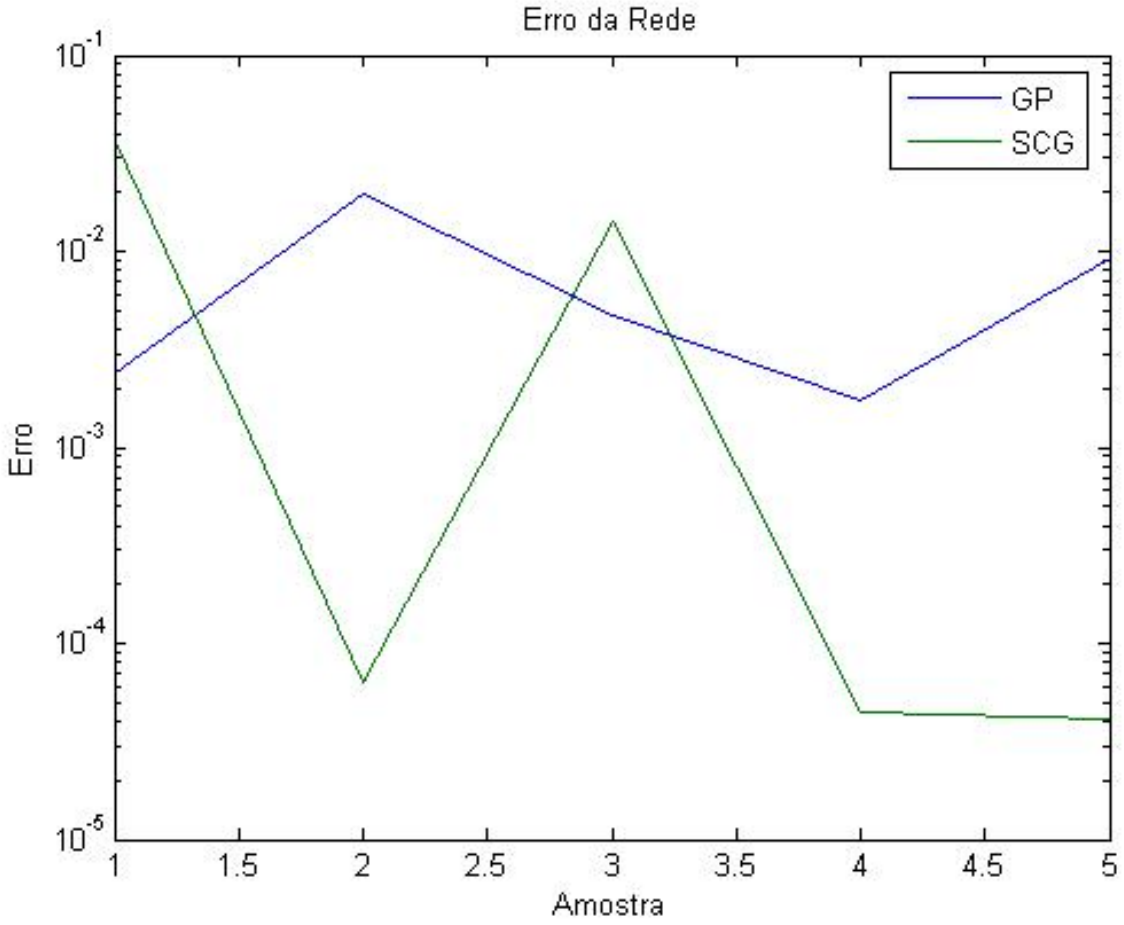

Figura 4.30: Comparação entre resultados obtidos pelos algoritmos GP e SCG

É possível observar, que apesar do erro médio obtido pelo algoritmo SCG ser menor do que aquele obtido com o algoritmo GP, o algoritmo proposto é comparável em eficiência com o algoritmo SCG com relação ao erro alcançada após o treinamento, obtendo resultados até mesmo superiores em algumas ocasiões mesmo sendo, em média, inferior. Dada a alta performance do algoritmo SCG, os resultados obtidos com a aplicação do algoritmo proposto são de grande relevância no treinamento de redes MLP.

4.7 Conclusões

Neste Capítulo são estudados vários algoritmos de treinamento para redes MLP, abrangendo os métodos mais eficientes encontrados na literatura. Também são discutidas novas propostas desenvolvidas com base em meta-heurísticas, que são comparadas a outros métodos através de aplicações práticas.

Os algoritmos de treinamento propostos neste documento: gradiente das partículas e gradiente evolutivo, mostram-se bastante eficientes com relação à taxa de convergência, sendo o algoritmo gradiente das partículas comparável ao algoritmo SCG com respeito a relação convergência $\times$ custo computacional. Como o algoritmo SCG representa o que existe de mais eficiente neste sentido [50], o método proposto torna-se uma opção de grande relevância no treinamento de redes MLP.

Ao contrario dos métodos procedurais, os algoritmos GP e GE, não tem como condição necessária à sua aplicação o conhecimento do vetor gradiente, o que os torna candidatos a serem aplicados a uma gama maior de problemas, onde antes somente era possível se utilizar métodos já conhecidos de busca cega como algoritmos genéticos e otimização por enxame de partículas.

A eficiência relativamente maior do algoritmo GP, quando comparado aos métodos de busca cega aqui estudados, o torna preferível em aplicações onde o vetor gradiente não é conhecido, como é o caso do observador neural adaptativo estudado no Capítulo 5 [56].

Capítulo 5

Observadores Inteligentes

5.1 Introdução

O problema de se observar o estado de um dado sistema surge diretamente da necessidade de controlá-lo, tendo em vista que esta tarefa requer, na maioria dos casos, o conhecimento completo do seu estado. Este, porém, pode não ser completamente conhecido, podendo ainda apresentar não-linearidades, o que representa um significativo agravante ao problema. Muitas vezes, o estado possui componentes não mensuráveis, ou mesmo que mensuráveis, o custo para se obter essas variáveis pode ser suficientemente alto para inviabilizar a sua medição. Nessas situações, faz-se uso de estimadores de estado, que, com base nas saídas mensuráveis do sistema, estimam os componentes desconhecidos do vetor de estado.

Por ter sido originado do problema de controle, o problema de estimação de estado não é tão antigo quanto este. Porém, já vem sendo estudado há várias décadas por inúmeros cientistas e engenheiros em todas as regiões do mundo. O empenho desses profissionais já rendeu à humanidade, incontáveis métodos para a solução do problema de estimação de estado. Porém, em geral, todos eles evoluíram de um único paradigma. Esta situação, onde um dado paradigma domina uma área do conhecimento, é observado também em outros campos da ciência, como por exemplo, a aviação.

No início do seculo XIX, Alberto Santos Dumont realizou em Paris, o primeiro vôo de uma aeronave mais pesada do que o ar, a qual recebeu o nome de 14-Bis. Muitos anos se passaram após este histórico evento e nossa civilização tem criado grandes avanços nesta área desde então, como grandes e sofisticados aviões para o transporte de pessoas, empregando o que há de mais avançado em tecnologia para controle de vôo e segurança da tripulação e da aeronave, modernos helicópteros que parecem desafiar a gravidade voando parados sob o ar, jatos ultra-sônicos capazes de alcançar velocidades outrora impensáveis, locomovendo-se mais rápido que o som, entre outros exemplos de maravilhas tecnológicas criadas para realizar o antigo sonho do homem de voar.

Porém, todas essas aeronaves têm algo em comum: elas voam devido à diferença de pressão gerada pela passagem do ar pelas asas, que cria uma força empurrando a aeronave para cima. Este é o mesmo princípio utilizado no início do seculo XIX por Alberto Santos Dumont para construir o 14-Bis. Portanto, todas essas aeronaves, são aperfeiçoamentos tecnológicos de um mesmo paradigma.

No campo da estimação de estado não é diferente. Por volta do início dos anos sessenta do século passado, o mundo conheceu o observador de Kalman e o observador de Luenberger, ambos formados por sistemas dinâmicos baseados no modelo do sistema a ser observado, e possuidores da incrível capacidade de fazer com que seu estado tenda a se igualar ao estado do sistema observado. Durante anos, surgiram novas propostas, oriundas de trabalhos de diversos pesquisadores que tentam melhorar a dinâmica do observador, de forma a adequá-lo cada vez mais à determinada classe de problemas, obtendo em geral, soluções mais criativas e eficientes para o problema de observação de estado. Porém, os observadores usuais estudados atualmente, apesar de apresentar um nível bastante elevado de sofisticação, preservam as idéias originais de Kalman e Luenberger, utilizando um sistema dinâmico para estimar o estado de outro sistema, pertencendo a um paradigma de observação que é praticamente o único conhecido.

O presente documento propõe uma solução alternativa à este problema, diferente daquela proposta por Kalman e Luenberger. O observador aqui estudado não é um sistema dinâmico como a grande parte das propostas encontradas na literatura, e sim um mapeamento estático, que cria uma relação única entre o subespaço formado pela saída do sistema e o subespaço formado pelo estado deste mesmo sistema, caracterizando assim uma nova classe de observadores que não são mais baseados em modelo, não pertencentes ao paradigma de estimação de estado usual e sim a um novo paradigma ainda inexplorado. Porém, com base nos resultados aqui obtidos, pode-se concluir que esta abordagem apresenta um grande potencial.

A garantia de existência dos observadores aqui propostos tem como única condição a observabilidade uniforme do sistema a ser observado, o que em hipótese alguma constitui uma condição demasiadamente restritiva, tendo em vista que sistemas que não possuem essa propriedade não podem ter seu estado corretamente estimados, seja qual for o paradigma de observação utilizado. Essa condição garante a existência de um difeomorfismo entre os subespaços formados pela saída do sistema e suas derivadas temporais e o subespaço formado pelo estado do sistema.

Tal difeomorfismo, muitas vezes, não pode ser expresso de forma analítica, pois, nesse caso, bastaria expressar essa solução e aplicá-la à saída do sistema para obter o estado, dispensando a utilização do observador. Por outro lado, mesmo que não possa ser expresso analiticamente, sempre que esse mapeamento existir, será possível aproximá-lo com o uso de alguma ferramenta com a capacidade necessária para realizar essa aproximação, como é o caso, de algumas técnicas de inteligência compu-

tacional. Essa capacidade de aproximação torna viável a aplicação desse mapeamento na estimação de estados, alcançando com isso resultados bastante satisfatórios [58, 59].

Consideremos o sistema dinâmico não linear genérico descrito a seguir:

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned} \tag{5.1}$$

onde $x \in R^n$ é o estado do sistema, $u \in R^m$ é a entrada do sistema, $y \in R^p$ é a saída do sistema, $f : R^n \rightarrow R^n$ é um campo vetorial[1]. onde, f_i para todo $i \in \{1...n\}$, é uma função suave e Lipschitz, e $h : R^n \rightarrow R^p$ é um mapeamento[2] onde, h_i para todo $i \in \{1...p\}$, é uma função suave e Lipschitz.

A concepção dos observadores inteligentes propostos neste documento é feita com base nas funções f e h, não sendo condição relevante à sua aplicação a forma destas funções, tampouco faz-se necessário que seja realizada qualquer transformação de estado para levar o sistema a uma dada forma canônica, como é feito em alguns dos observadores propostos na literatura. A condição suficiente para aplicação do estimador de estado proposto ao sistema (5.1) é que este seja uniformemente observável.

5.2 Observabilidade Em Sistemas Não Lineares

Em termos práticos, um dado sistema ser considerado observável implica na possibilidade de recuperar informações sobre o seu estado com base na entrada aplicada ao sistema e na saída mensurável obtida durante um intervalo de tempo t, o que só é possível se y carregar informação suficiente a respeito do vetor de estado completo. Esta é uma definição bastante ampla e rudimentar de observabilidade, que pode ser verificada na análise do sistema (5.2):

$$\begin{aligned} \dot{x} &= f(x) + g(x,u) \\ y &= 0 \end{aligned} \tag{5.2}$$

Neste sistema, a saída y claramente não carrega informação alguma a respeito do vetor de estado, sendo assim, não se possuem métodos para a estimação desse vetor com base em y. A fim de expor uma definição formal da condição de observabilidade, que seja passível de aplicação a um maior conjunto de sistemas, são apresentadas várias definições dessa condição, baseadas principalmente na teoria geométrica de controle, que provê condições baseadas em subespaços formados pelas funções f e h do sistema (5.1).

[1] f será um campo vetorial caso se considere uma entrada u constante, do contrário, f é um mapeamento: $f : R^{n+m} \rightarrow R^n$

[2] Novamente para uma entrada u não-constante tem-se: $h : R^{n+m} \rightarrow R^p$.

5.2.1 Condições Geométricas de Observabilidade

A definição de condições de controlabilidade e observabilidade para sistemas não lineares é uma tarefa relativamente complexa quando comparada a definição desses conceitos para sistemas lineares. A abordagem adotada neste documento é a utilização de condições oriundas do controle geométrico [60, 61, 62]. Essa escolha é tomada principalmente por favorecer a aplicação das ferramentas de inteligência computacional ao problema de estimação de estado.

Recuperar a informação sobre o vetor de estado baseando-se em y implica em dizer que um observador, com base na informação contida no vetor y, consegue distinguir, entre vários (infinitos) estados diferentes, o real estado do sistema. O que é equivalente a afirmar que não é admitida a indistinguibilidade, dada pela Definição 1.

Definição 1 (Indistinguibilidade) *O par de estados do sistema (5.1) (x_0, x_0'), é considerado indistinguível, se for verificada a seguinte condição [63, 61]:*

$$\exists u \in \mathcal{U} | \ \forall t \geq 0, y(x_0, u, t) = y(x_0', u, t) \tag{5.3}$$

A observabilidade pode então ser estabelecida com base nesta definição da forma como segue:

Definição 2 (Observabilidade) *Dado um estado x_0, o sistema (5.1) será considerado observável, se não existir x_0' tal que o par (x_0, x_0') seja indistinguível [63, 61].*

A condição dada pela Definição 2 é bastante geral e pouco aplicável de forma prática, devido principalmente à sua representação global da observabilidade. Assim, faz-se necessário definir uma condição menos restritiva, como é feito na Definição 3.

Definição 3 (Observabilidade Fraca) *O sistema (5.1) é considerado fracamente observável com respeito a um dado estado x_0, se existir uma vizinhança $\mathcal{X}^0$, que não contenha nenhum estado x_0', tal que o par (x_0, x_0') seja indistinguível [63, 61].*

Essa definição, é validada somente para o estado x_0, e não localmente para a sua vizinhança. A Definição 4, traz uma condição local de observabilidade para um subespaço $\mathcal{V} \subset R^n$:

Definição 4 (Observabilidade Local Fraca) *O sistema (5.1) é localmente fracamente observável com relação a um estado x_0, se existir uma vizinhança $\mathcal{V}$ de x_0, tal que para todo $x \in \mathcal{V}$ não exista $x' \in \mathcal{Z}$ onde $\mathcal{Z}$ também é uma vizinhança de x_0, de forma que o par (x, x') seja indistinguível [63, 61].*

Esta condição garante que, dado um estado x_0, não existam estados indistinguíveis em sua vizinhança, criando um subespaço $\mathcal{V}$ onde o sistema é observável.

Em sistemas lineares, a seguinte condição é suficiente para garantir a observabilidade do sistema [63, 64, 65]:

$$posto \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{pmatrix} = n \tag{5.4}$$

onde $f(x,u) = Ax + Bu$ e $h(x,u) = Cx$ são as equações que definem a dinâmica do sistema.

É possível notar que a observabilidade para essa classe de sistemas não depende da entrada, característica essa não verificada em sistemas não-lineares, onde mesmo assegurada a observabilidade para algumas entradas, pode existir $u \in \mathcal{U}$ tal que exista x_0 e x_0' onde: (x_0, x_0') é indistinguível. Dessa forma, a observabilidade de sistemas não lineares pode se tornar dependente da entrada, o que requer uma definição mais abrangente desse conceito, discriminando essa possibilidade. Mais precisamente, essa dependência é introduzida pelas definições de *entrada universal* e *observabilidade uniforme*.

Definição 5 (Entrada Universal) *Uma entrada $u \in \mathcal{U}$ é universal se atendida a seguinte condição [63]:*

$$\forall x_0 \neq x_0', \forall t \in T \; y(x_0, u, t) \neq y(x_0', u, t) \tag{5.5}$$

A entrada u será considerada uma entrada singular sempre que esta não for universal.

Com base nesta definição, a observabilidade uniforme pode ser definida como segue:

Definição 6 (Observabilidade Uniforme) *Um sistema é considerado uniformemente observável, se todo $u \in \mathcal{U}$ for uma entrada universal [63].*

A Definição 6 remove a dependência da entrada na condição de observabilidade, permitindo a construção de observadores para sistemas não-lineares de forma independente da entrada u, similarmente ao que é feito para sistemas lineares. Para sistemas não uniformemente observáveis, essa construção ainda pode ser possível, porém, o observador resultante terá sua atuação limitada às entradas u pertencentes ao subespaço $\mathcal{U}'$ tal que todo $u \in \mathcal{U}'$ é uma entrada universal do sistema.

A condição de observabilidade uniforme pode ser expressa por meio de subespaços, através da condição geométrica conhecida como condição de posto para observabilidade. Essa condição provê

uma ferramenta para verificação prática do conceito de observabilidade uniforme e tem grande relevância na área do controle geométrico. A formalização dessa condição requer a definição dos espaços de observação e da codistribuição de observabilidade.

Definição 7 (Espaço de Observação) *Considerando um vetor de entrada u constante, o espaço de observação associado ao sistema (5.1): $\mathcal{O}$ é por definição o menor espaço vetorial gerado pelos componentes da função h tal que: $\mathcal{L}_f^k h_i \in \mathcal{O}$ para todo $h_i \in \mathcal{O}$ e para todo $k \in \{1...n-1\}$. [62, 63], onde $\mathcal{L}_f h_i$ representa a derivada de Lie de h_i na direção de f*

A codistribuição de observabilidade é então descrita com base no espaço de observação como segue:

Definição 8 (Codistribuição de Observabilidade $d\mathcal{O}$) *Codistribuição de observabilidade $d\mathcal{O}$ é a matriz formada pelas componentes de $\mathcal{O}$, onde cada linha dessa matriz é dada segundo a expressão (5.6) [62].*

$$d\mathcal{O}_i = \nabla_x o_i | o_i \in \mathcal{O} \tag{5.6}$$

onde $\nabla_x o_i$ representa o vetor gradiente da função o_i com relação à x.

O espaço $d\mathcal{O}$ é então utilizado para descrever a condição de posto para observabilidade, o que é feito na Definição 9

Definição 9 (Condição de Posto Para Observabilidade) *O sistema satisfaz a condição de posto para observabilidade, para um dado estado x_0, se a seguinte equação for verdadeira [62, 61, 63]:*

$$r = posto(d\mathcal{O})|_{x_0} = n \tag{5.7}$$

Se o sistema satisfaz a condição de posto para todo $x \in \mathcal{X}'$, então o mesmo será localmente fracamente observável em $\mathcal{X}'$ (para um resultado global, deve-se considerar $\mathcal{X}' = \mathcal{X}$)[3]. Se o sistema atender a condição de posto, para todo $x \in \mathcal{X}'$ e para todo $u \in \mathcal{U}'$ o mesmo será *Uniformemente Observável* em $\mathcal{X}' \times \mathcal{U}'$ e se esta condição for satisfeita para $\mathcal{X}' = \mathcal{X}$ e $\mathcal{U}' = \mathcal{U}$, então o sistema será completamente uniformemente observável[4] (CUO).

A título de exemplo, consideremos o sistema linear descrito na equação (5.8):

[3] Aqui $\mathcal{X}$ representa o espaço de estado, ou mais detalhadamente, o menor espaço vetorial que contém todos os possíveis estados do sistema.

[4] $\mathcal{U}$ representa o menor espaço vetorial que contém todas as possíveis entradas do sistema.

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{5.8}$$

onde $A \in R^{n\times n}$, $B \in R^{n\times m}$ e $C \in R^{p\times n}$.

O espaço de observação para este sistema, com u constante é dado pela equação (5.9)

$$\mathcal{O} = \begin{bmatrix} \mathcal{L}_f^0 h_1 \\ \vdots \\ \mathcal{L}_f^{n-1} h_1 \\ \mathcal{L}_f^0 h_2 \\ \vdots \\ \mathcal{L}_f^{n-1} h_2 \\ \vdots \\ \mathcal{L}_f^0 h_p \\ \vdots \\ \mathcal{L}_f^{n-1} h_p \end{bmatrix} = \begin{bmatrix} c_1 x \\ \vdots \\ c_1 A^{n-1} x + c_1 A^{n-2} B u \\ c_2 x \\ \vdots \\ c_2 A^{n-1} x + c_2 A^{n-2} B u \\ \vdots \\ c_p x \\ \vdots \\ c_p A^{n-1} x + c_p A^{n-2} B u \end{bmatrix} \tag{5.9}$$

onde c_k para $k \in \{1...p\}$ é a k-ésima linha da matriz C.

A matriz de codistribuição de observabilidade é calculada com base no gradiente de $\mathcal{O}$. O resultado obtido para o sistema em questão é mostrado na equação (5.10).

$$d\mathcal{O} = \begin{bmatrix} c_1 \\ \vdots \\ c_1 A^{n-1} \\ c_2 \\ \vdots \\ c_2 A^{n-1} \\ \vdots \\ c_p \\ \vdots \\ c_p A^{n-1} \end{bmatrix} \tag{5.10}$$

A condição de posto para observabilidade requer que o posto de $d\mathcal{O}$ seja igual à dimensão do estado, logo, a seguinte igualdade deve ser verificada:

$$posto(d\mathcal{O}) = posto \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{pmatrix} = n \tag{5.11}$$

Com esse exemplo, conclui-se que a condição de posto para a observabilidade é equivalente à condição de observabilidade clássica, quando aplicada a sistemas lineares.

Consideremos agora um sistema não linear, descrito pela equação (5.12):

$$\begin{aligned} \dot{x_1} &= x_1^2 + 2x_2 - u \\ \dot{x_2} &= x_1 x_2 \\ y &= x_1 \end{aligned} \tag{5.12}$$

As funções f e h são dadas a seguir:

$$f = \begin{bmatrix} x_1^2 + 2x_2 - u \\ x_1 x_2 \end{bmatrix} \tag{5.13}$$

$$h = x_1 \tag{5.14}$$

O espaço de observação e a matriz de codistribuição de observabilidade para este sistema, com u constante são descritos nas equações (5.15) e (5.16).

$$\mathcal{O} = \begin{bmatrix} x_1 \\ x_1^2 + 2x_2 - u \end{bmatrix} \tag{5.15}$$

$$d\mathcal{O} = \begin{bmatrix} 1 & 0 \\ 2x_1 & 2 \end{bmatrix} \tag{5.16}$$

Através da análise da matriz $d\mathcal{O}$, conclui-se que o sistema é completamente uniformemente observável, dado que $posto(d\mathcal{O}) = 2$ para todo $[x_1, x_2] \in R^2$ e todo $u \in R$.

5.2.2 Observabilidade e Difeomorfismo

Seja o sistema não linear genérico (5.1), com u constante e $p = 1$, a partir do qual pode-se definir o vetor y_e da seguinte forma [65]:

$$y_e = \begin{bmatrix} y \\ \dot{y} \\ \vdots \\ y^{(n-1)} \end{bmatrix} = \begin{bmatrix} \phi_0(x, u, \dot{u}, ..., u^{(n_u)}) \\ \phi_1(x, u, \dot{u}, ..., u^{(n_u)}) \\ \vdots \\ \phi_{n-1}(x, u, \dot{u}, ..., u^{(n_u)}) \end{bmatrix} \tag{5.17}$$

onde y_e é um vetor contendo a saída mensurável do sistema e suas $n-1$ derivadas e n_u representa o número de derivadas de u que surgem na representação do vetor y_e.

Com base nas funções ϕ_i é possível definir o mapeamento $\mathcal{H}$ como segue:

$$\mathcal{H} \triangleq \begin{bmatrix} \phi_0(x, u, \dot{u}, ..., u^{(n_u)}) \\ \phi_1(x, u, \dot{u}, ..., u^{(n_u)}) \\ \vdots \\ \phi_{n-1}(x, u, \dot{u}, ..., u^{(n_u)}) \end{bmatrix} \tag{5.18}$$

Tem-se então a seguinte igualdade:

$$y_e = \mathcal{H}(x, u, \dot{u}, ..., u^{(n_u)}) \tag{5.19}$$

Por economia de notação, deste ponto em diante não será explicitada a dependência das derivadas de u no mapeamento $\mathcal{H}$, sendo esse então reescrito como segue:

$$y_e = \mathcal{H}(x, u) \tag{5.20}$$

A equação (5.19) descreve o mapeamento $\mathcal{H}$, explicitando sua dependência com relação à entrada do sistema u, que surge quando a função h é também explicitamente dependente dessa variável, ou quando o sistema possui um grau relativo forte $n_\nu \leq n$. Em sistemas onde se observa $n_v \geq n$ e a função h não depende de u, o mapeamento $\mathcal{H}$ pode ser expresso em função das derivadas de Lie de h na direção de f, equação (5.21).

$$\mathcal{H} \triangleq \begin{bmatrix} h(x) \\ \mathcal{L}_f h(x) \\ \vdots \\ \mathcal{L}_f^{n-1} h(x) \end{bmatrix} \tag{5.21}$$

É possível afirmar, caso o sistema seja completamente uniformemente observável, que o mapeamento $\mathcal{H}$ constitui um difeomorfismo [65, 63] entre os espaços vetoriais $\mathcal{O}$ e $\mathcal{X}$ para $\mathcal{H}$ definido na

equação (5.21), ou entre os espaços vetoriais[5] $\mathcal{Y} \times \mathcal{U}$ e $\mathcal{X}$ de forma mais abrangente. Dessa forma, pode-se escrever sua relação inversa como segue:

$$x = \mathcal{H}^{-1}(y_e, u) \tag{5.22}$$

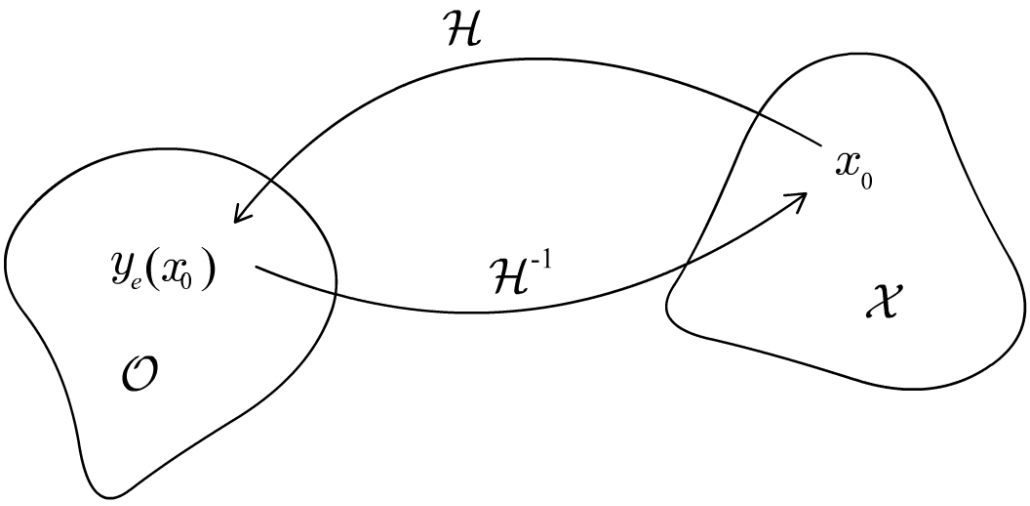

Figura 5.1: Difeomorfismo

Consideremos agora um sistema MIMO, que possui $p > 1$; neste caso o vetor y_e pode ser expresso da seguinte forma.

[5] Aqui $\mathcal{Y}$ é o menor espaço vetorial formado pelas funções ϕ_i^j que compõe o vetor y_e, e $\mathcal{Y} \times \mathcal{U}$ é o produto cartesiano entre esse espaço e o espaço formado pelos vetores de entrada u

$$y_e = \begin{bmatrix} y_1 \\ \dot{y_1} \\ \vdots \\ y_1^{(n-1)} \\ y_2 \\ \dot{y_2} \\ \vdots \\ y_2^{(n-1)} \\ \vdots \\ y_p \\ \dot{y_p} \\ \vdots \\ y_p^{(n-1)} \end{bmatrix} = \begin{bmatrix} \phi_1^1(x,u) \\ \phi_2^1(x,u) \\ \vdots \\ \phi_{n-1}^1(x,u) \\ \phi_1^2(x,u) \\ \phi_2^2(x,u) \\ \vdots \\ \phi_{n-1}^2(x,u) \\ \vdots \\ \phi_1^p(x,u) \\ \phi_2^p(x,u) \\ \vdots \\ \phi_{n-1}^p(x,u) \end{bmatrix} \tag{5.23}$$

Esse vetor pertence ao espaço R^r, onde: $r = n \cdot p$. O mapeamento $\mathcal{H}$, neste caso, é definido por r funções ϕ_i^j, e será inversível sempre que a matriz $d\mathcal{O} \in R^{r \times n}$, equação (5.24), tiver posto n.

$$d\mathcal{O} = \begin{bmatrix} \nabla_x \phi_0^1(x,u) \\ \nabla_x \phi_1^1(x,u) \\ \vdots \\ \nabla_x \phi_{n-1}^1(x,u) \\ \nabla_x \phi_0^2(x,u) \\ \nabla_x \phi_1^2(x,u) \\ \vdots \\ \nabla_x \phi_{n-1}^2(x,u) \\ \vdots \\ \nabla_x \phi_0^p(x,u) \\ \nabla_x \phi_1^p(x,u) \\ \vdots \\ \nabla_x \phi_{n-1}^p(x,u) \end{bmatrix} \tag{5.24}$$

É possível notar, caso $posto(d\mathcal{O}) = n$, que existem $r - n$ linhas redundantes na matriz $d\mathcal{O}$, o que implica em redundância também no vetor y_e e no mapeamento $\mathcal{H}$, dado que são necessárias apenas n funções $\phi(x,u)$ com gradientes linearmente independentes, para representar um difeomorfismo entre

os espaços $\mathcal{Y} \times \mathcal{U}$ e $\mathcal{X}$ e assim ser possível recuperar o estado a partir do vetor y_e e da entrada do sistema u, através da transformação inversa do mapeamento $\mathcal{H}$.

A equação (5.22) obtém uma relação explicita entre o estado do sistema e sua saída mensurável (considera-se aqui que saída mensurável y pode ser facilmente derivada com relação ao tempo, sendo assim, o vetor y_e é considerado mensurável). Sempre que for possível obter uma representação analítica para o mapeamento $\mathcal{H}^{-1}$, o problema de estimação de estado pode ser resolvido de forma trivial, como é observado no sistema não-linear mostrado anteriormente na equação (5.12). O difeomorfismo para este sistema é expresso pelas seguintes equações:

$$y_e = \mathcal{H}(x, u) = \begin{bmatrix} x_1 \\ x_1^2 + 2x_2 - u \end{bmatrix} \tag{5.25}$$

$$x = \mathcal{H}^{-1}(y_e, u) = \begin{bmatrix} y_{e_1} \\ \frac{y_{e_2} - y_{e_1}^2 + u}{2} \end{bmatrix} \tag{5.26}$$

Esta solução, porém, nem sempre é possível, sendo sua aplicação restrita a um pequeno grupo de sistemas para os quais pode-se escrever a relação inversa do mapeamento $\mathcal{H}$ analiticamente. A solução proposta neste documento visa obter esse mapeamento por meio de uma aproximação numérica. Tal aproximação pode ser realizada teoricamente através de vários métodos, dentre os quais pode-se citar as redes neurais e sistemas *fuzzy*, que representam importantes ferramentas de aproximação universal. Neste trabalho, optou-se pela utilização de redes neurais do tipo MLP na implementação deste mapeamento, e também é proposta uma abordagem alternativa, onde tenta-se transformar o problema de aproximação em um problema de busca, ao qual são aplicadas técnicas evolutivas e de inteligência de enxame.

5.3 Observador Neural: OBN

Classicamente, são estudados estimadores de estado baseados em modelos dinâmicos concebidos com base na dinâmica do sistema a ser observado. Essa solução foi introduzida por volta dos anos 60, é de grande relevância nesta área e vem sendo estudada até os dias atuais. No caso do sistema (5.1), um observador que segue este paradigma clássico é descrito pela equação a seguir:

$$\dot{\hat{x}} = f(\hat{x}, u) + \Gamma(h(\hat{x}), h(y)) \tag{5.27}$$

onde $\hat{x}$ é o estado estimado do sistema e Γ é uma função que define o observador

Na estimação de estado em sistemas não-lineares, são propostas várias funções Γ diferentes, que se adaptam às não-linearidades específicas de cada sistema. Assim, a tarefa de se observar essa classe

de sistemas exige um estudo prévio a fim de identificar as características das não-linearidades do sistema e classificá-lo de acordo com elas, para então utilizar-se uma função Γ apropriada para o problema.

A aplicação de redes neurais ao problema de estimação de estado é bastante conhecida na literatura, principalmente por trabalhos onde tenta-se aproximar não-linearidades do sistema utilizando essa ferramenta [66, 67] dentre outras. Essa abordagem costuma apresentar bons resultados, principalmente na construção de observadores adaptativos, onde não se possui informação completa a respeito das funções f e h, ou essas podem sofrer variações ao longo do tempo.

Neste documento, será utilizada uma abordagem alternativa na construção de estimadores de estado, onde as redes neurais são utilizadas na aproximação do mapeamento $\mathcal{H}^{-1}$ [58]. Nesta abordagem, faz-se necessário o conhecimento de um conjunto de dados para o treinamento da rede (este conjunto é constituído de dados de treinamento, teste e validação da rede), que consta de amostras do vetor y_e, da entrada do sistema u e de seu estado x, tal que: $y_e = \mathcal{H}(x, u)$. A Figura 5.2 ilustra o observador neural proposto.

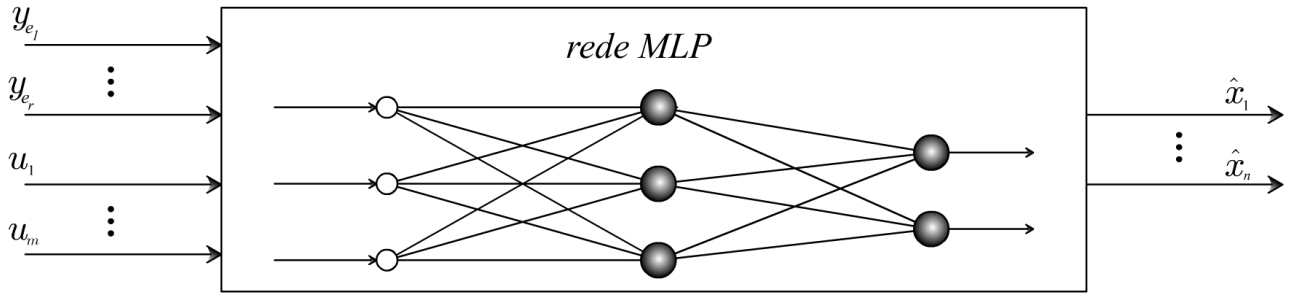

Figura 5.2: Observador Neural

O observador neural é descrito pela equação 5.28:

$$\hat{x} = \mathcal{F}(y_e, u, \omega) \tag{5.28}$$

onde ω é um vetor de parâmetros livres da rede MLP, que são ajustados em uma fase de aprendizagem do observador, na qual este é treinado para assimilar o mapeamento $\mathcal{H}^{-1}$.

A camada de entrada da rede MLP, irá possuir $r + m$ neurônios, onde r é a dimensão do vetor y_e. A camada de saída da rede, que deve representar o estado do sistema , possui n neurônios, sendo que cada um representa uma componente do vetor de estado. Em sistemas MIMO, é possível também eliminar componentes redundantes do vetor y_e, diminuindo a dimensão da camada de entrada da rede para: $n + m$, de forma que se observem as seguintes igualdades para uma dada entrada u constante:

$$
\begin{aligned}
y_e &= \begin{bmatrix} y_{e_1} \\ \vdots \\ y_{e_k} \\ y_{e_{k+l+1}} \\ \vdots \\ y_{e_r} \end{bmatrix} \\
\nabla y_e &= \begin{bmatrix} \nabla y_{e_1} \\ \vdots \\ \nabla y_{e_k} \\ \nabla y_{e_{k+l+1}} \\ \vdots \\ \nabla y_{e_r} \end{bmatrix} \\
posto(\nabla y_e) &= n
\end{aligned} \tag{5.29}
$$

onde l componentes foram removidas do vetor y_e na posição k.

Existem ainda alguns parâmetros a serem definidos para o observador neural, sendo eles:

- **Número de camadas intermediárias.** Esse número deve ser definido pelo projetista, e requer um certo conhecimento e observação por parte deste, não havendo uma regra geral para sua escolha, que é feita de forma empírica em função das características dos dados que serão mapeados pela rede.
- **Número de neurônios em cada camada intermediária.** Da mesma forma como o número de camadas intermediárias, o número de neurônios nessas camadas deve ser escolhido pelo projetista de forma empírica.
- **Funções de ativação.** Neste documento, optou-se por utilizar funções de ativação sigmoidais do tipo tangente hiperbólica, porém, qualquer outra função não-linear e monotonicamente crescente, que obedeça o teorema da aproximação universal, pode ser utilizada.
- **Parâmetro p das funções de ativação.** Esse parâmetro define a inclinação da função sigmoidal, no caso, a função tangente hiperbólica. Aqui foi escolhido adotar $p = 1$ para todos os neurônios. Porém, é possível escolher um valor de p diferente para cada neurônio da rede.

O treinamento do observador pode ser realizado utilizando qualquer algoritmos de treinamento supervisionado, como aqueles apresentados neste documento, bem como qualquer outro capaz de

ajustar os parâmetros da rede MLP com base nas amostras de entradas e saídas esperadas. Esse treinamento é realizado previamente (de forma *off-line*) à utilização do observador, que deve ser treinado com base em um conjunto de dados significativo com relação a sua aplicação.

A escolha dos dados de treinamento é a parte mais crítica do projeto do observador neural, tendo em vista que faz-se necessário apresentar dados para rede capazes de representar corretamente as condições às quais o observador será exposto. Consideremos, a título de exemplo, que o observador foi treinado utilizando amostras $y_e \in \mathcal{Y}^1 \subset \mathcal{Y}$ e $u \in \mathcal{U}^1 \subset \mathcal{U}$, é provável que este não apresente uma estimativa muito precisa para dados: $y_e \in \mathcal{Y}^2 \subset \mathcal{Y}$ e $u \in \mathcal{U}^2 \subset \mathcal{U}$ tal que: $\mathcal{Y}^1 \bigcap \mathcal{Y}^2 = \emptyset$ e $\mathcal{U}^1 \bigcap \mathcal{U}^2 = \emptyset$. A escolha de amostras representativas, porém, pode não ser uma tarefa trivial, dependendo da aplicação do observador.

Após concluído o treinamento da rede MLP, essa irá reproduzir de forma aproximada o mapeamento $\mathcal{H}^{-1}$. Assim o observador neural proposto será capaz de apresentar estimativas do estado do sistema com base nos vetores y_e e u. O erro de estimação do observador, caracterizado por alguma métrica da diferença entre a estimativa e o valor real do estado: $O(x - \hat{x})$, é aqui representado pelo erro médio quadrático da rede MLP para o conjunto de validação, mostrado na equação (5.30).

$$O = \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \sum_{j=1}^{n} \left(\mathcal{F}(y_e^{val}(i), u^{val}(i), \omega)_j - x_j^{val}(i) \right)^2 \tag{5.30}$$

onde O é o erro de estimação do observador, $y_e^{val}(i)$, $u^{val}(i)$ e $x^{val}(i)$ representam a i-ésima amostra de um conjunto de validação e n_{val} é o número de amostras contidas neste conjunto.

Dadas as características da rede MLP, é possível afirmar ainda que esse erro obedece a seguinte equação:

$$O \leq O_{max} \tag{5.31}$$

onde O_{max} é um escalar, que representa o erro máximo cometido pela rede neural.

Como o observador proposto não é um sistema dinâmico, e sim um mapeamento estático, não é pertinente a ele o conceito de estabilidade, não fazendo sentido a sua análise. Em estimadores de estado clássicos, o estudo da estabilidade do observador é de fundamental relevância, sendo indispensável na fase de projeto. Essa análise de estabilidade geralmente não é uma tarefa trivial, demandando um esforço significativo por parte do projetista.

5.3.1 Exemplo de Aplicação

Consideremos o seguinte sistema dinâmico não linear:

$$\begin{aligned} \dot{x}_1 &= x_1^2 - x_2 - u \\ \dot{x}_2 &= x_1 x_2 - 2x_1^2 \\ y &= x_1 \end{aligned} \tag{5.32}$$

Com base no sistema genérico (5.1), as funções f e h são descritas como segue:

$$\begin{aligned} f &= \begin{bmatrix} x_1^2 - x_2 - u \\ x_1 x_2 - 2x_1^2 \end{bmatrix} \\ h &= x_1 \end{aligned} \tag{5.33}$$

Como o grau relativo forte desse sistema é: $n_v = 2$, e $h(x)$ não depende de u, pode-se escrever o mapeamento $\mathcal{H}$ utilizando a derivada de Lie, equação (5.34). A matriz de codistribuição de observabilidade é então calculada através do gradiente das componentes de $\mathcal{H}$:

$$\begin{aligned} \mathcal{H} &= \begin{bmatrix} h(x) \\ \mathcal{L}_f h(x) \end{bmatrix} \\ &= \begin{bmatrix} x_1 \\ x_1^2 - x_2 - u \end{bmatrix} \end{aligned} \tag{5.34}$$

Codistribuição de observabilidade:

$$\begin{aligned} d\mathcal{O} &= \begin{bmatrix} \nabla_x (h(x)) \\ \nabla_x \left(\mathcal{L}_f h(x) \right) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 2x_1 & -1 \end{bmatrix} \end{aligned} \tag{5.35}$$

Como é possível notar claramente, $posto(d\mathcal{O}) = 2$ para todo $[x_1, x_2] \in R^2$; pode-se assim concluir que o sistema é completamente uniformemente observável, o que garante a existência do mapeamento $\mathcal{H}^{-1}$ e também do observador neural proposto.

A escolha da topologia da rede MLP utilizada na estimação do estado não segue um padrão prédefinido, sendo sua definição uma função, entre outros fatores, da estrutura do sistema a ser observado. Quanto maior a complexidade do mapeamento $\mathcal{H}^{-1}$, maior o número de neurônios necessários

para assimilá-lo. Porém, um excesso de neurônios pode prejudicar o treinamento da rede ou até mesmo causar *overfitting*. Portanto, a escolha deste parâmetro requer certa experiência e observação. Para o sistema em questão, foi utilizada uma rede com uma única camada intermediária, contendo 8 neurônios (Figura 5.3). As funções de ativação da camada intermediária já foram previamente definidas, e a saída é uma combinação linear dos sinais dos neurônios da camada intermediária.

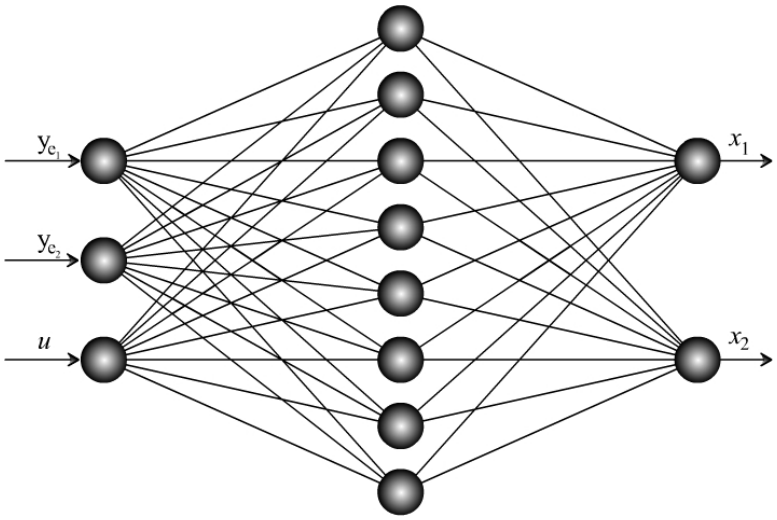

Figura 5.3: Rede Neural

Os conjuntos de treinamento e de teste utilizados no aprendizado do observador constam de 183 amostras obtidas com base em uma simulação computacional do sistema descrito na equação (5.32). Nesta simulação, são tomadas novas amostras com base em um período de amostragem $\tau = 0.05$ segundos, durante um período de tempo total de 9 segundos. Durante esse intervalo de tempo, foi aplicada ao sistema a entrada mostrada na Figura 5.4, que consta de três degraus consecutivos com amplitudes 1, 2 e 3, nessa ordem. Na simulação, a cada 3 segundos o estado do sistema é forçado à retornar a origem do espaço de estado, o que pode ser entendido como a remoção da energia do sistema.

Essa *reinicialização* do estado é feita com o claro intuito de fornecer dados amostrais, que reflitam em um conjunto de teste significativo com relação a uma região de operação desejada. No caso em questão, espera-se que o observador, após ter sido devidamente treinado, seja capaz de reconhecer o estado do sistema sempre que for aplicada a este uma entrada em degrau com qualquer amplitude pertencente ao intervalo $[1, 3]$.

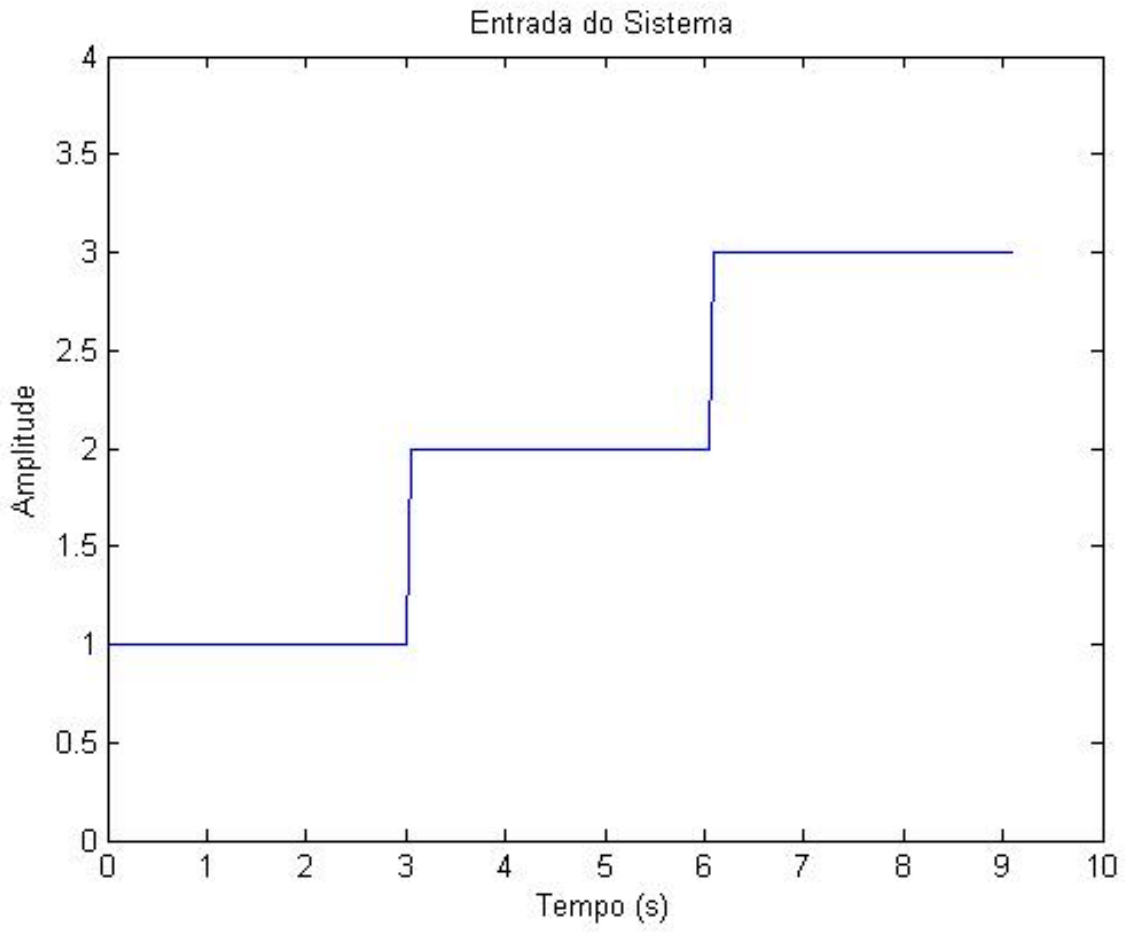

Figura 5.4: Entrada do sistema, exemplo de aplicação: OBN

A dinâmica do sistema para esta simulação gera a saída mostrada na Figura 5.5. Este gráfico ilustra as trajetórias das duas componentes do vetor y_e: a saída do sistema e sua primeira derivada com relação ao tempo, ambas dadas a seguir:

$$\begin{aligned} y_{e_1} &= y \\ y_{e_2} &= \mathcal{L}_f h(x) \end{aligned}$$

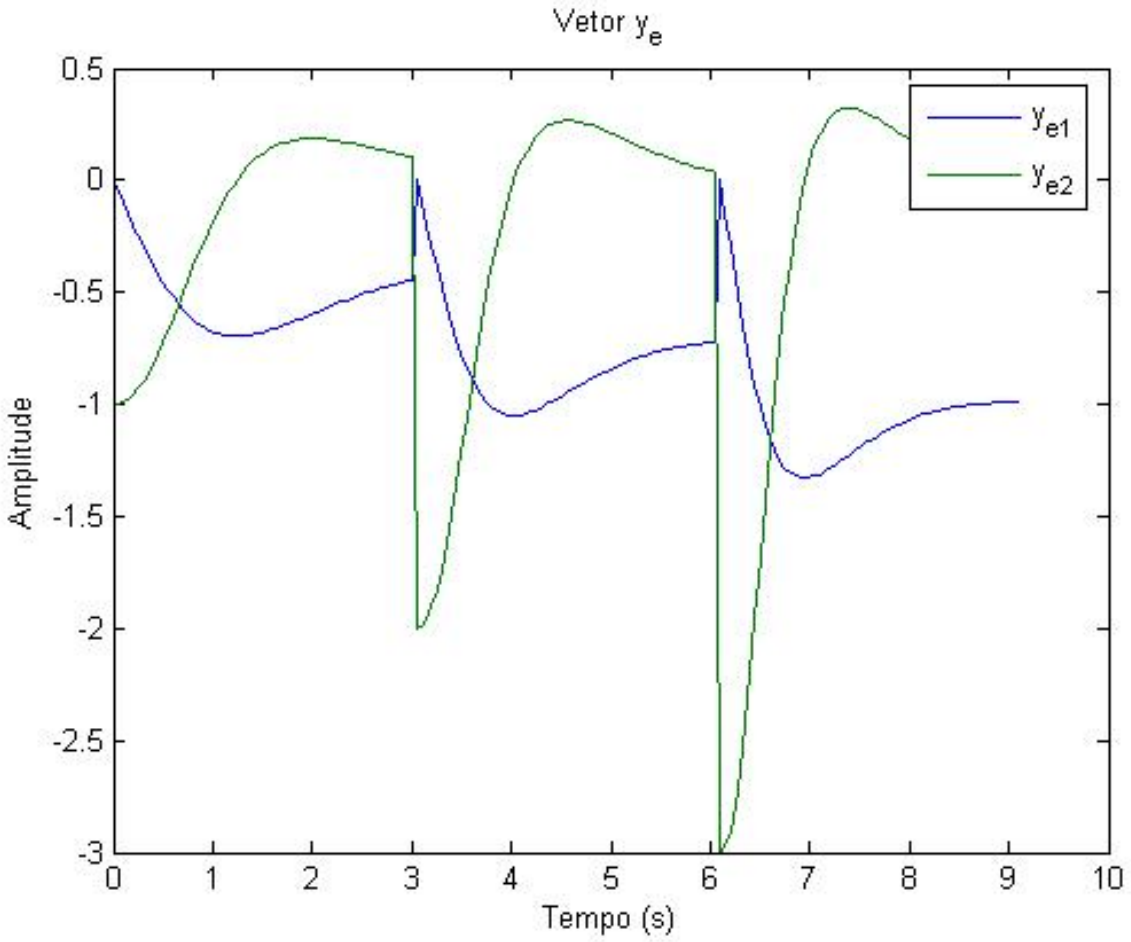

Figura 5.5: Trajetória do vetor y_e, exemplo de aplicação: OBN

O estado real do sistema, obtido durante a simulação em questão, é mostrado na Figura 5.6, que descreve as trajetórias das duas componentes do vetor de estado, na qual nota-se claramente a reinicialização desse vetor a cada 3 segundos.

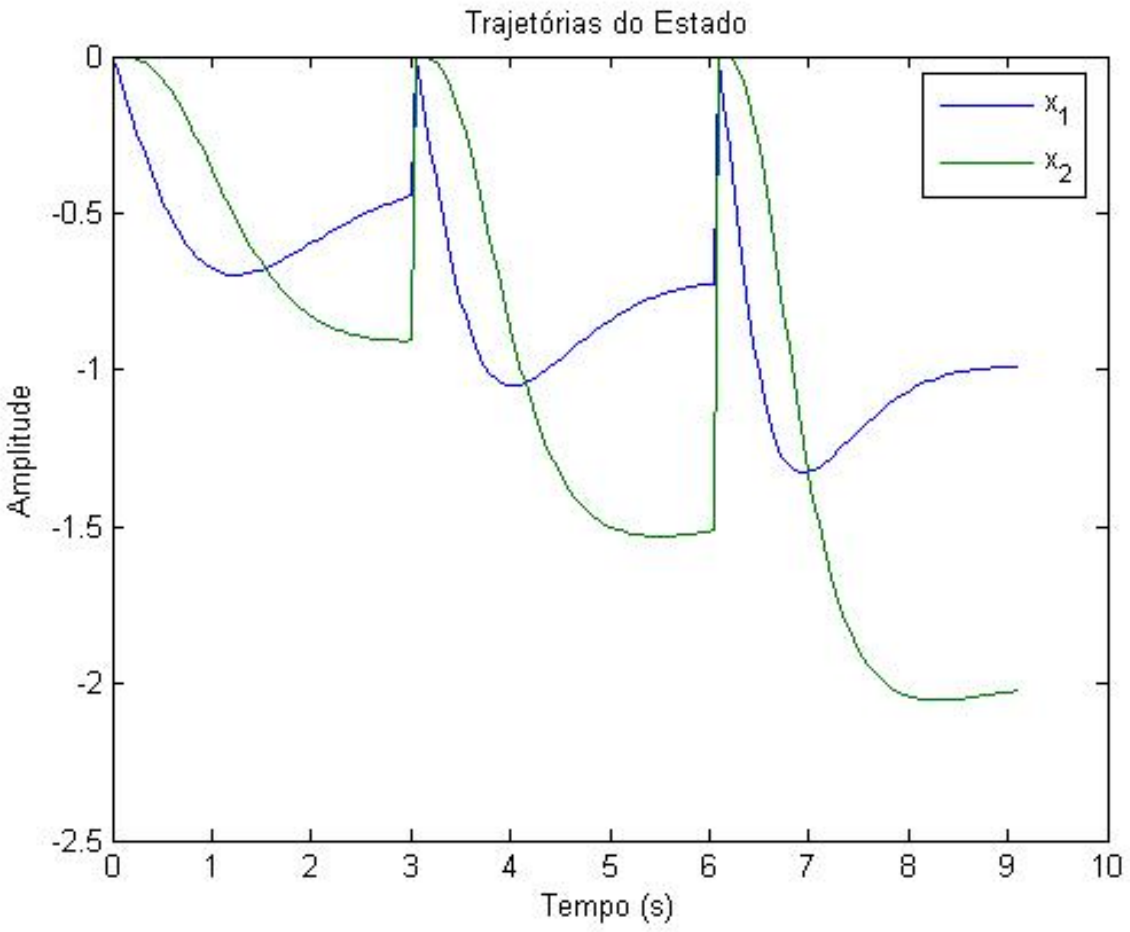

Figura 5.6: Trajetória do estado, exemplo de aplicação: OBN

As trajetórias do estado do sistema Figura 5.6, do vetor y_e Figura 5.5 e da entrada do sistema u completam o conjunto de treinamento para o observador, com relação ao cenário de aplicação proposto.

O treinamento desse observador pode então ser realizado por qualquer algoritmo de treinamento de redes MLP. Para este exemplo, foi utilizado o algoritmo Fletcher-Reeves, aplicado por 200 épocas. O resultado desse treinamento é um vetor de parâmetros ω que completa o projeto do observador neural proposto.

A Figura 5.7 demonstra o estado estimado do sistema, obtido pelo observador quando aplicado aos dados do conjunto de treinamento utilizado. Nessa situação, como era esperado, é obtido um excelente resultado, tendo em vista que esses dados foram utilizados no treinamento da rede MLP.

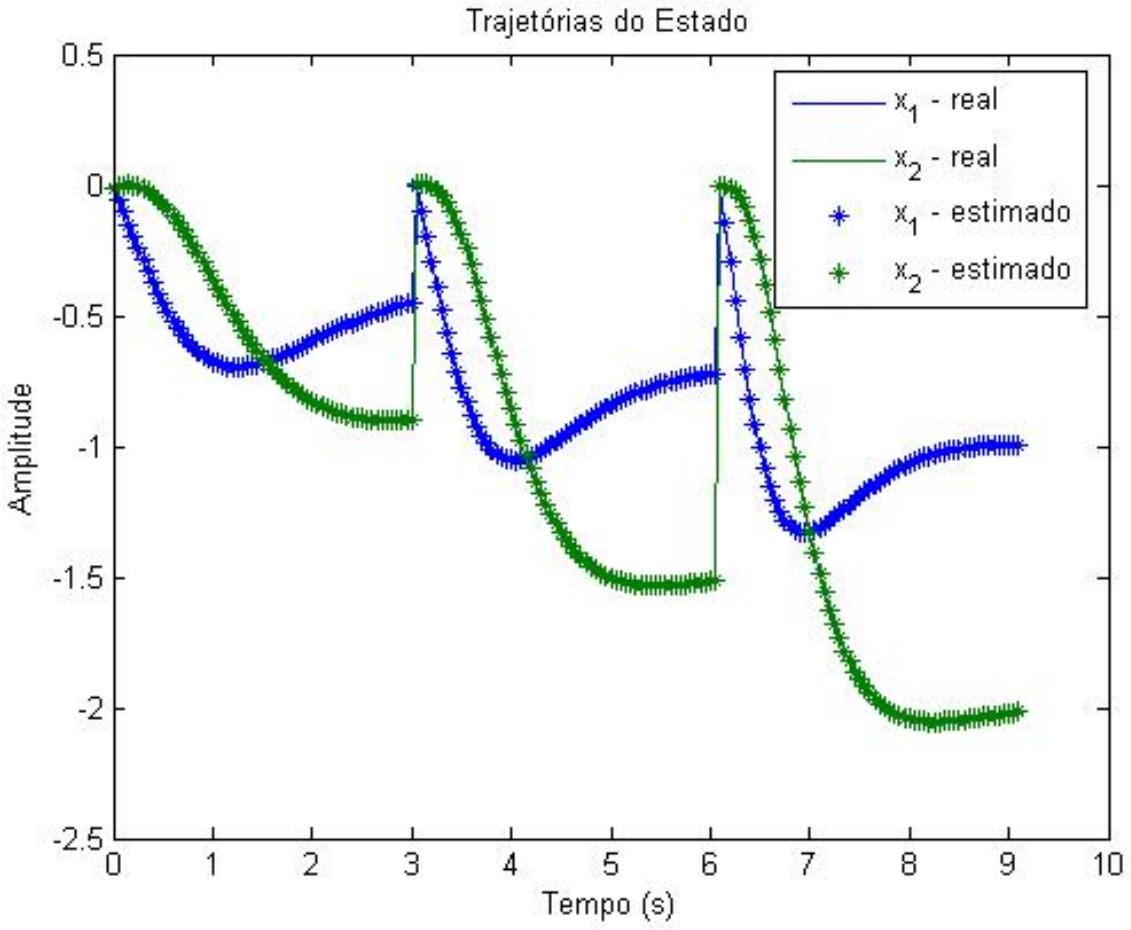

Figura 5.7: Trajetória do estado, conjunto de treinamento: OBN

Pode-se concluir que a rede assimilou o mapeamento $\mathcal{H}^{-1}$ existente entre a entrada e a saída do sistema para os dados do conjunto de treinamento, demonstrando de forma prática a capacidade de estimação do observador proposto. Esse resultado, porém, não demonstra a capacidade de generalização do observador, que deve ser capaz de estimar estados que não façam parte do seu conjunto de treinamento.

É tomado então um conjunto contendo 61 amostras de validação de y_e, u e x, tomadas com o mesmo período de amostragem utilizado anteriormente, durante 3 segundos, quando é aplicada ao sistema uma entrada u descrita por um degrau com amplitude 1.5. A validação do observador é feita então através da aplicação do mesmo na estimação do estado do sistema para esse conjunto de amostras, que não consta no conjunto utilizado no seu treinamento. A resposta da rede para o conjunto de validação, é descrita na Figura 5.8, onde são plotadas as trajetórias do estado real e do estado estimado do sistema.

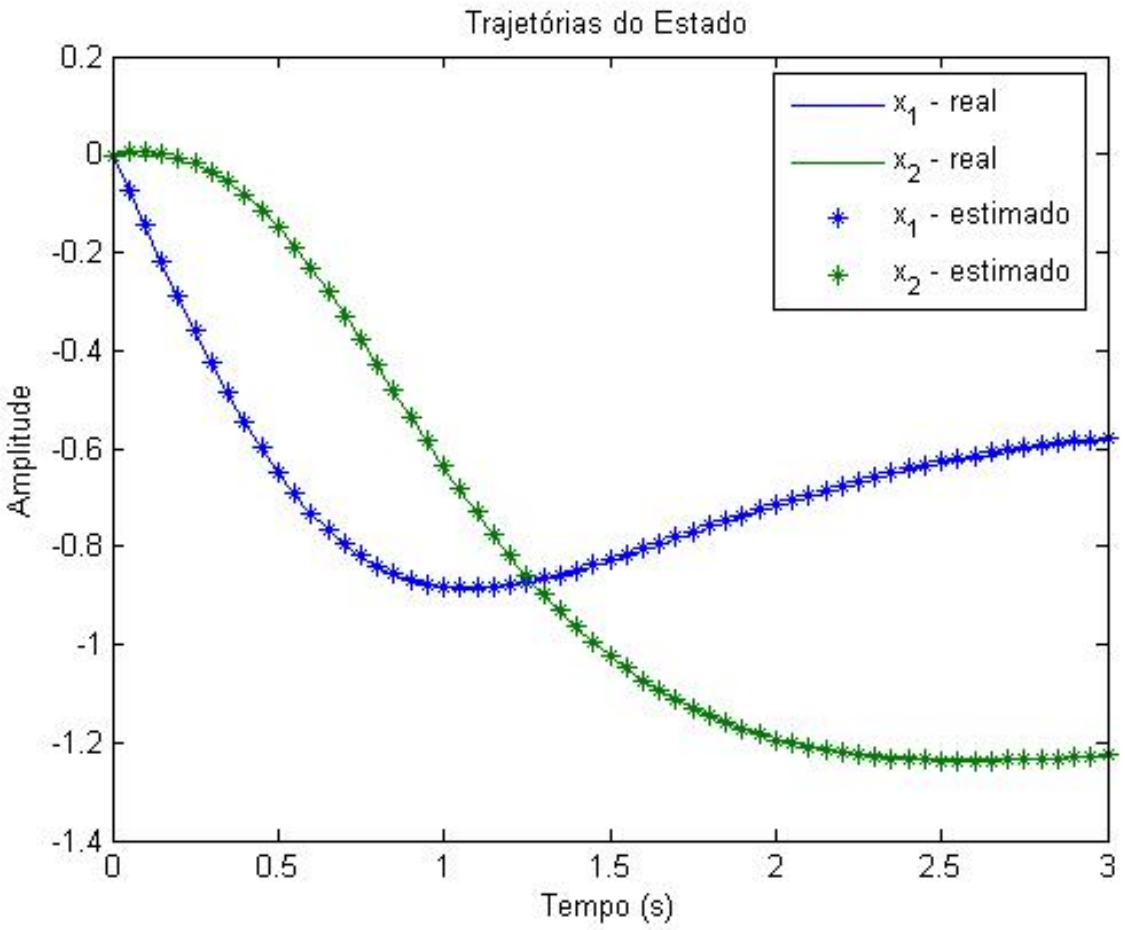

Figura 5.8: Trajetória do estado, conjunto de validação: OBN

Nota-se claramente no resultado da estimação em questão, que o observador neural apresentou uma boa estimativa do estado do sistema x, mesmo quando aplicado a dados não pertencentes ao conjunto utilizado em seu treinamento, apresentando um erro $W = 1.0529 \cdot 10^{-005}$. O erro de estimação para cada estado desse conjunto de validação, mostrado na Figura 5.9, confirma a qualidade da capacidade de generalização apresentada pelo observador neural.

Essa capacidade de generalização, juntamente com a capacidade de aproximação da rede MLP, torna viável a aplicação do observador proposto em várias situações praticas, desde que apropriadamente treinado, utilizando um conjunto de treinamento contendo dados suficientemente representativos dentro do contexto da aplicação final do observador.

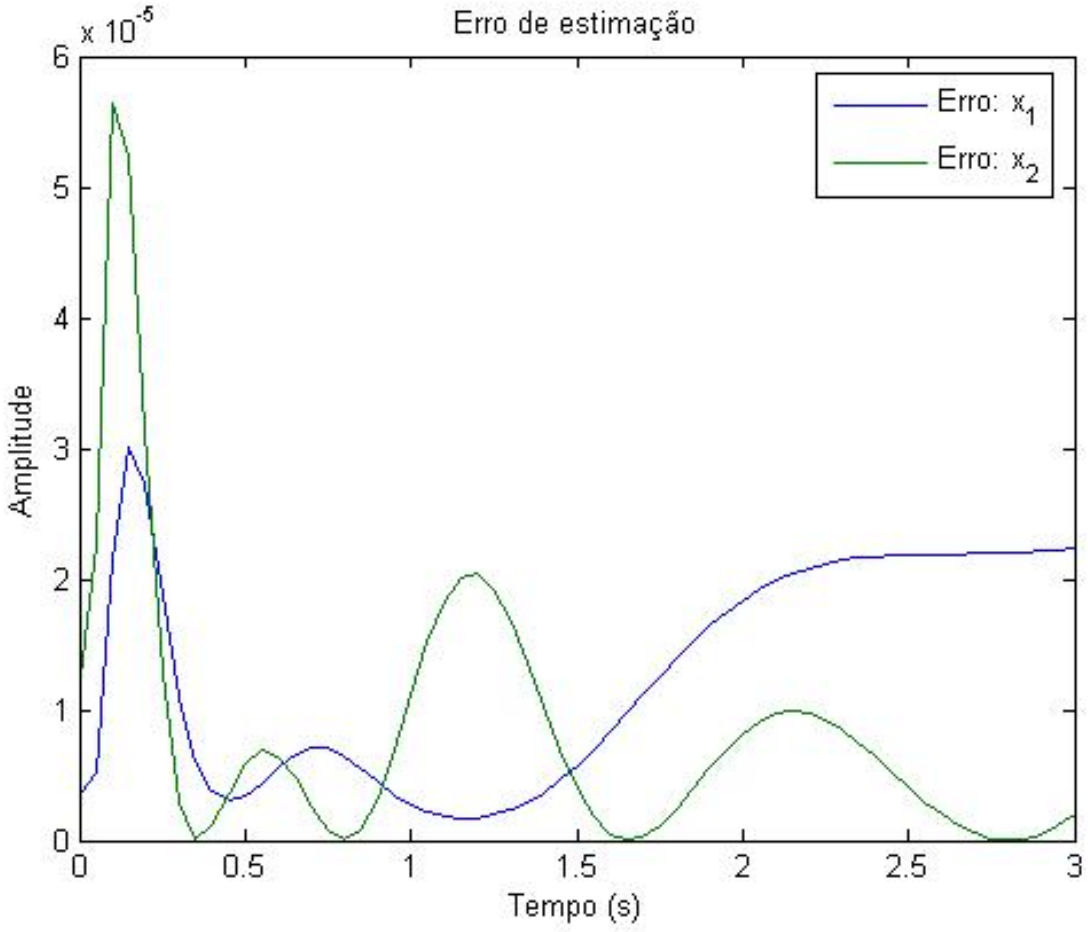

Figura 5.9: Erro de estimação dos estados, conjunto de validação: OBN

A aplicação do observador neural proposto requer a definição empírica de uma série de parâmetros, como o número de camadas intermediárias, número de neurônios nessas camadas, funções de ativação, entre outros. A escolha desses parâmetros afeta diretamente a capacidade de aproximação e generalização da rede MLP, que, por sua vez, afeta o desempenho do observador neural.

A fim de analisar a sensibilidade do observador com relação aos parâmetros topológicos da rede, consideremos um observador com apenas 4 neurônios na camada intermediária, o que representa metade da quantidade utilizada no observador considerado anteriormente. Aqui são empregados os mesmos conjuntos de treinamento e validação utilizados na análise anterior. Nestas condições, são obtidos os resultados mostrados na Figura 5.10, onde são plotadas as trajetórias estimadas para o vetor de estado quando o observador é aplicado aos dados do conjunto de validação.

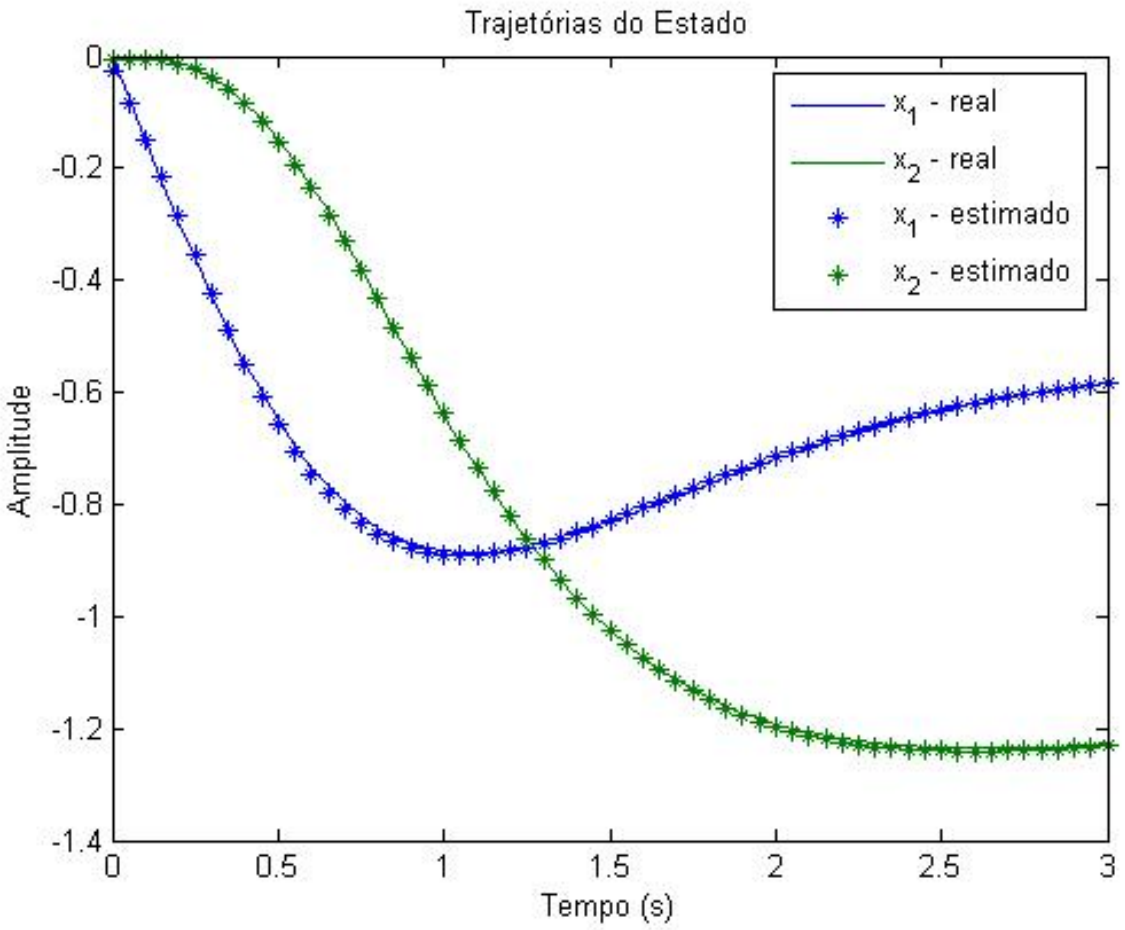

Figura 5.10: Trajetória do estado estimado para o conjunto de validação, utilizando 4 neurônios na camada intermediária

A qualidade da estimativa apresentada utilizando essa configuração é bastante semelhante àquela observada anteriormente, na qual a rede MLP possuía o dobro de neurônios em sua camada intermediária. O erro do observador, porém, é superior, tendo sido obtido para esta configuração um erro $W = 2.7869 \cdot 10^{-005}$, que é 2.65 vezes superior ao erro atingido utilizando a configuração anterior.

Esse erro ainda é razoavelmente pequeno, mesmo quando comparado com aquele obtido anteriormente, não implicando em grande diferença nos resultados práticos obtidos. Essa afirmação pode ser facilmente verificada na Figura 5.11, onde são plotados, em escala logarítmica, os erros de estimação de cada estado para as duas configurações analisadas, sendo o erro obtido pela rede com 8 neurônios na camada intermediária designado por: Erro 01 e o erro obtido com a outra configuração designado por Erro 02.

É possível notar uma grade semelhança entre os erros de estimação obtidos com as duas configurações em questão, o que comprova que a variação imposta na topologia da rede não causa grandes

impactos no resultado da estimação. Dessa forma, a escolha dessa topologia não é considerada um fator crítico no projeto do observador neural proposto.

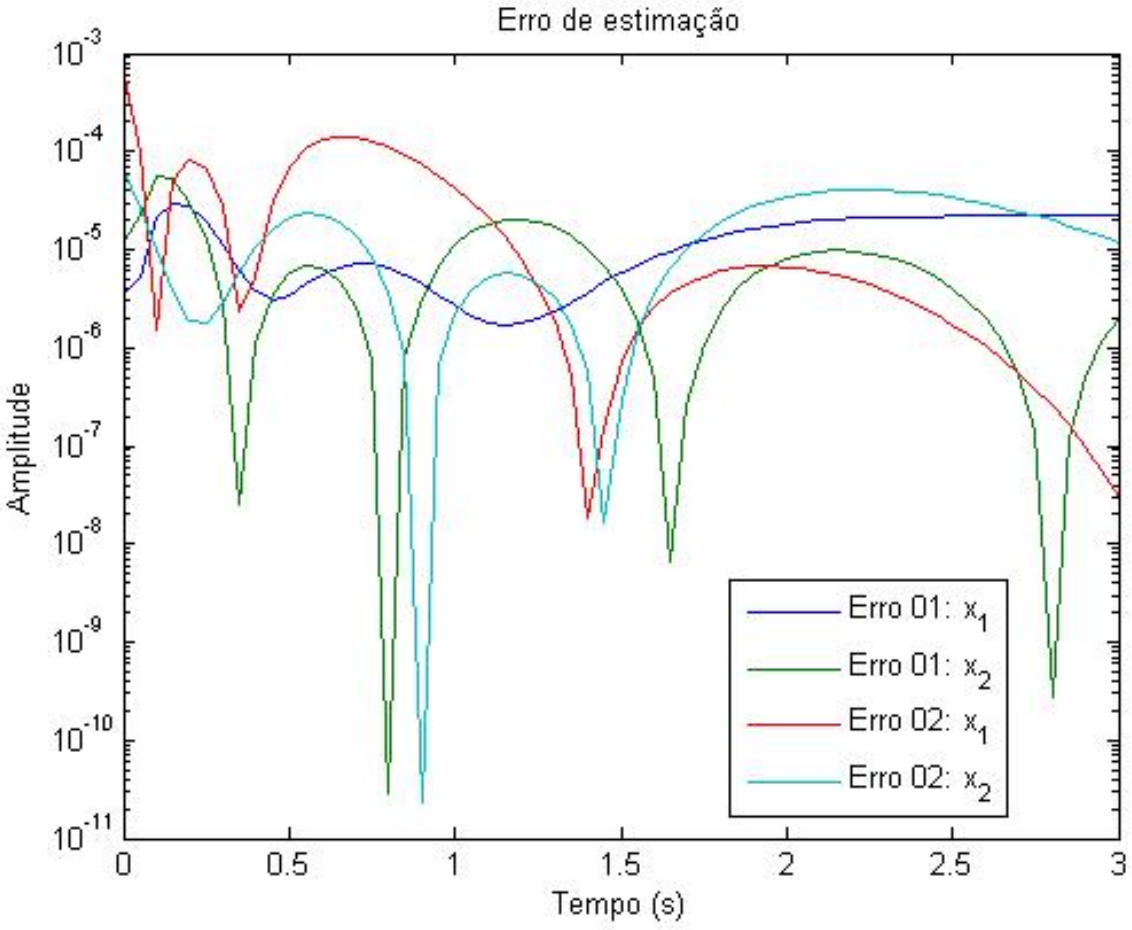

Figura 5.11: Comparação entre os erros de estimação: OBN

Outros testes foram realizados utilizando redes MLP com diferentes topologias na estimação do estado deste mesmo sistema, obtendo-se, em todos, um resultado semelhante àqueles aqui apresentado, o que indica claramente que o observador proposto é consideravelmente robusto em relação aos parâmetros topológicos da rede MLP, robustez esta que representa um facilitador considerável na implementação deste observador.

Consideremos agora o sistema MIMO não-linear mostrado na equação (5.36), com $p = m = 2$ e $n = 4$.

$$\begin{aligned} \dot{x}_1 &= x_2^2 - u_1 \\ \dot{x}_2 &= -x_3 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= x_1 x_2 x_3 - x_3^2 + u_2 \\ y_1 &= x_1 + u_1 \\ y_2 &= x_2 - \dot{u}_2 \end{aligned} \tag{5.36}$$

O mapeamento $\mathcal{H}(x,u)$ para este sistema é descrito na equação (5.37):

$$\mathcal{H}(x,u) = \begin{bmatrix} x_1 + u_1 \\ x_2^2 - u_1 + \dot{u}_1 \\ -2x_2x_3 - \dot{u}_1 + \ddot{u}_1 \\ -2x_2x_4 + 2x_3^2 - \ddot{u}_1 + u_1^{(3)} \\ x_2 - \dot{u}_2 \\ -x_3 - \ddot{u}_2 \\ -x_4 - u_2^{(3)} \\ -x_1x_2x_3 + x_3^2 + u_2 - u_2^{(4)} \end{bmatrix} \tag{5.37}$$

A matriz de codistribuição de observabilidade para o sistema (5.36) é mostrada na equação (5.38):

$$d\mathcal{O} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2x_2 & 0 & 0 \\ 0 & -2x_3 & -2x_2 & 0 \\ 0 & -2x_4 & 4x_3 & -2x_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ -x_2x_3 & -x_1x_3 & -x_1x_2 + 2x_3 & 0 \end{bmatrix} \tag{5.38}$$

É evidente na equação (5.38) que $posto(d\mathcal{O}) = n$ para todo vetor $x \in R^4$. Portanto, o sistema 5.36 é completamente uniformemente observável, logo, pode-se afirmar que o mapeamento inverso $\mathcal{H}^{-1}(y_e, u)$ existe. É possível também utilizar apenas 4 funções componentes do mapeamento $\mathcal{H}$ para expressar esse difeomorfismo, dado que as demais são redundantes. Uma análise da matriz $d\mathcal{O}$ mostra claramente que uma possível escolha seria utilizar as componentes 1, 5, 6 e 7 do vetor y_e como entradas do observador, dado que essas componentes expressam completamente o difeomorfismo entre os espaços $\mathcal{Y} \times \mathcal{U}$ e $\mathcal{X}$.

Neste caso, pode-se utilizar uma rede MLP com $4+2$ neurônios na camada sensorial e 4 neurônios na camada de saída na implementação do observador neural, e como é comprovada a existência do mapeamento $\mathcal{H}^{-1}$, essa rede neural será capaz de apresentar estimativas do estado desse sistema após ter sido devidamente treinada.

5.4 Observador Adaptativo Neural: OAN

O observador neural anteriormente proposto deve ser treinado antes de sua aplicação na estimação do estado, além de exigir o conhecimento de um conjunto de amostras do estado sistema. Por outro lado, a adaptação progressiva observada durante o processo de treinamento das redes neurais, sugere a possibilidade de implementar um observador capaz de se adaptar ao sistema de forma *on-line* durante a sua operação e (não antes desta). Essa motivação leva ao desenvolvimento do observador adaptativo neural, uma ferramenta capaz de assimilar o mapeamento $\mathcal{H}^{-1}$ enquanto observa o sistema, e não a partir de um conjunto de treinamento pré-determinado.

Neste observador, o conjunto de treinamento é constituído de amostras dos vetores y_e e u. Porém, nesta abordagem não é conhecido o real estado do sistema a ser observado x, logo, este não pode fazer parte do conjunto de treinamento da rede MLP. Essa ausência do estado inviabiliza o aprendizado da rede por meios usuais, pois, além de não ser possível comparar sua saída com uma saída desejada para a estipulação de um erro, é impossível também a construção do vetor gradiente desse erro, eliminando a possibilidade de utilização de métodos de treinamento procedurais na aprendizagem do observador.

A solução aqui encontrada foi a utilização de um critério alternativo de erro que possibilite a aplicação de métodos heurísticos na implementação do processo de treinamento [56]. Neste documento, é implementado no aprendizado do observador o método proposto: *Gradiente das Partículas*, devido ao seu bom desempenho quando comparado a outros métodos heurísticos de treinamento de redes MLP, podendo ainda ser utilizada outra ferramenta de busca cega capaz de localizar um ponto no espaço R^N com base no critério de erro aqui utilizado.

A avaliação do erro do observador é feita com base na descrição analítica do mapeamento $\mathcal{H}$ e na saída da rede MLP, que supostamente deve ser igual ao estado do sistema x, que é desconhecido. O critério de erro em questão é mostrado na equação (5.39).

$$\begin{aligned} erro_o &= \sum_{i=1}^{n}(\hat{y}_{e_i} - y_{e_i})^2 \\ &= \sum_{i=1}^{n}(\mathcal{H}(\hat{x})_i - y_{e_i})^2 \end{aligned}$$

$como:$

$$\begin{aligned} \hat{x} &= \mathcal{F}(y_e, u, \omega) \\ portanto: & \\ erro_o(y_e, u, \omega) &= \sum_{i=1}^{n} (\mathcal{H}(\mathcal{F}(y_e, u, \omega), u)_i - y_{e_i})^2 \end{aligned} \tag{5.39}$$

Na equação (5.39), proposta para avaliação do erro do observador, o resultado da aplicação do mapeamento $\mathcal{H}$ ao estado estimado $\hat{x}$, dado por: $\hat{y}_e = \mathcal{H}(\hat{x}, u)$, é comparado com o vetor mensurável y_e, obtendo assim uma métrica do erro da rede MLP, sem que seja requerido, para tanto, o conhecimento do estado real do sistema.

Consideremos ainda a aplicação da função h ao estado estimado $\hat{x}$: $\hat{y} = h(\hat{x})$. Essa grandeza pode ser comparada à saída do sistema y, dado que tem-se $y = h(x)$, o que também poderia gerar uma métrica para o erro da rede MLP. Essa abordagem, apesar de mais intuitiva, é claramente equivocada, pois a saída do sistema não necessariamente contém informação a respeito do estado completo deste. Isto é facilmente verificado no sistema (5.12), que é completamente uniformemente observável, porém, $y = h(x) = x_1$ não traz informação alguma a respeito do estado x_2.

Definido o erro da rede neural, a implementação do observador adaptativo em questão segue o mesmo processo anteriormente utilizado para o observador neural, empregando também o mesmo conjunto de parâmetros na definição da rede MLP utilizada. A aplicação do observador adaptativo, porém, é bastante diferente e requer a definição de uma série de conceitos.

A seguir, é mostrado na Figura 5.12, o diagrama de blocos que ilustra a aplicação do observador adaptativo proposto na estimação de estado de um sistema não-linear.

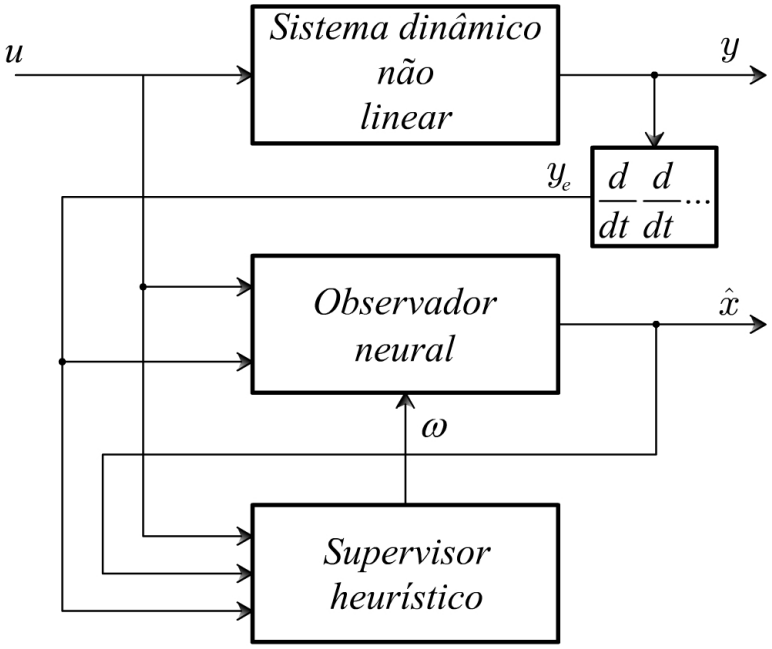

Figura 5.12: Observador adaptativo neural

A estrutura do supervisor, mostrada na Figura 5.13, permite a aplicação de uma vasta gama de algoritmos heurísticos de busca cega no treinamento do observador adaptativo, porém, a utilização de algoritmos híbridos, como o *gradiente evolutivo* e o *gradiente das partículas* favorece consideravelmente o processo de treinamento, pois o vetor de parâmetros ω, conhecido em um dado instante t, é facilmente utilizado como ponto de partida para um novo processo de treinamento, que inicie posteriormente em $t + \alpha$.

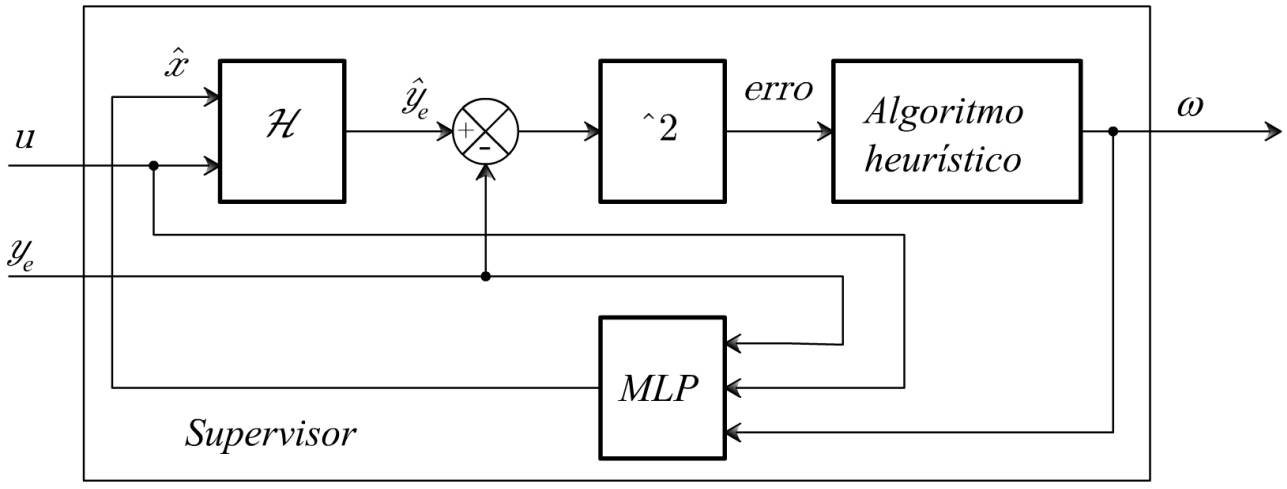

Figura 5.13: Mecanismo supervisor

O conjunto de treinamento utilizado para o aprendizado desse observador é construído de forma dinâmica, durante a operação do sistema, quando são adicionadas a esse conjunto amostras dos vetores y_e e u a cada τ intervalos de tempo, onde τ é o período de amostragem do observador.

Essa constante inserção de amostras ao conjunto de treinamento gera um constante aumento dessa coleção de dados, assim, devem ser tomadas certas precauções que levem em conta a capacidade limitada de armazenamento da informação, a fim de evitar problemas computacionais. Esse problema pode ser resolvido limitando o número máximo de amostras permitido para o conjunto de treinamento. Porém, deve-se estabelecer um critério apropriado para excluir dados desse conjunto, não apenas rejeitando amostras obtidas após ter sido atingido o número máximo, o que provavelmente prejudicaria a capacidade de adaptação do observador.

O aprendizado da rede também é realizado durante a operação do sistema, enquanto o conjunto de treinamento é atualizado. Uma possível implementação desse processo de aprendizagem, aqui sugerida, é realização de uma ou mais épocas do algoritmo de treinamento, a cada período de amostragem τ, sendo possível também realizar o processo de treinamento a cada k novas amostras, o que é bastante útil quando não é possível concluir o processo de treinamento em um tempo inferior a τ. Como nesse intervalo de tempo é acrescentada uma nova amostra ao conjunto de treinamento, a rede neural será treinada sempre utilizando dados diferentes e atualizados, o que garante uma adaptação gradual do observador ao problema de estimação.

A estimação do estado pode ser realizada de várias formas distintas, dependendo basicamente das necessidades oriundas da aplicação do observador. São aqui sugeridos dois processos distintos de estimação: *seqüencial* e *paralelo*. A abordagem seqüencial tenta maximizar a eficiência do observador, fornecendo o estado estimado do sistema $\hat{x}(t)$ a cada τ intervalos de tempo, baseando-se em toda a informação disponível até o instante de tempo t, como mostra o diagrama representado na Figura (5.14). Em uma aplicação onde o observador é utilizado para realimentar o estado para um controlador, torna-se imprescindível que a soma do tempo demandado nos processos de treinamento e de estimação seja inferior a τ, para que o estado estimado possa ser coerentemente utilizado no processo de controle. Para que isso seja possível, podem ser tomadas algumas medidas para reduzir o tempo empregado no treinamento da rede, como, por exemplo diminuir o número de épocas implementadas nesse processo, ou mesmo aumentar o período de amostragem τ. Porém, sempre que essa abordagem não for realizável, ainda é possível utilizar o modelo paralelo de estimação.

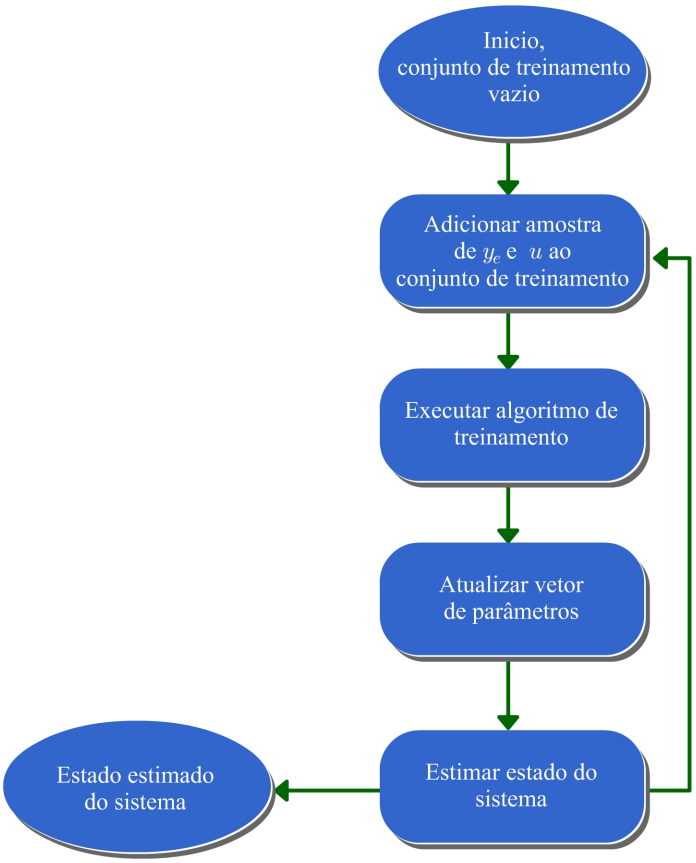

Figura 5.14: Fluxograma de estimação seqüencial

No processo de estimação paralela (Figura 5.15), o treinamento e a estimação são executados paralelamente. O treinamento é realizado constantemente, mantendo atualizado o vetor de parâmetros ω. Quando é solicitada ao observador uma estimativa do estado do sistema, é então utilizado o vetor de parâmetros atual, obtido ao término do último processo iterativo de treinamento realizado. Essa abordagem é bastante útil quando faz-se necessária uma manutenção constante ou assíncrona da estimativa do estado, ou quando o processo de treinamento não pode ser concluído em tempo hábil.

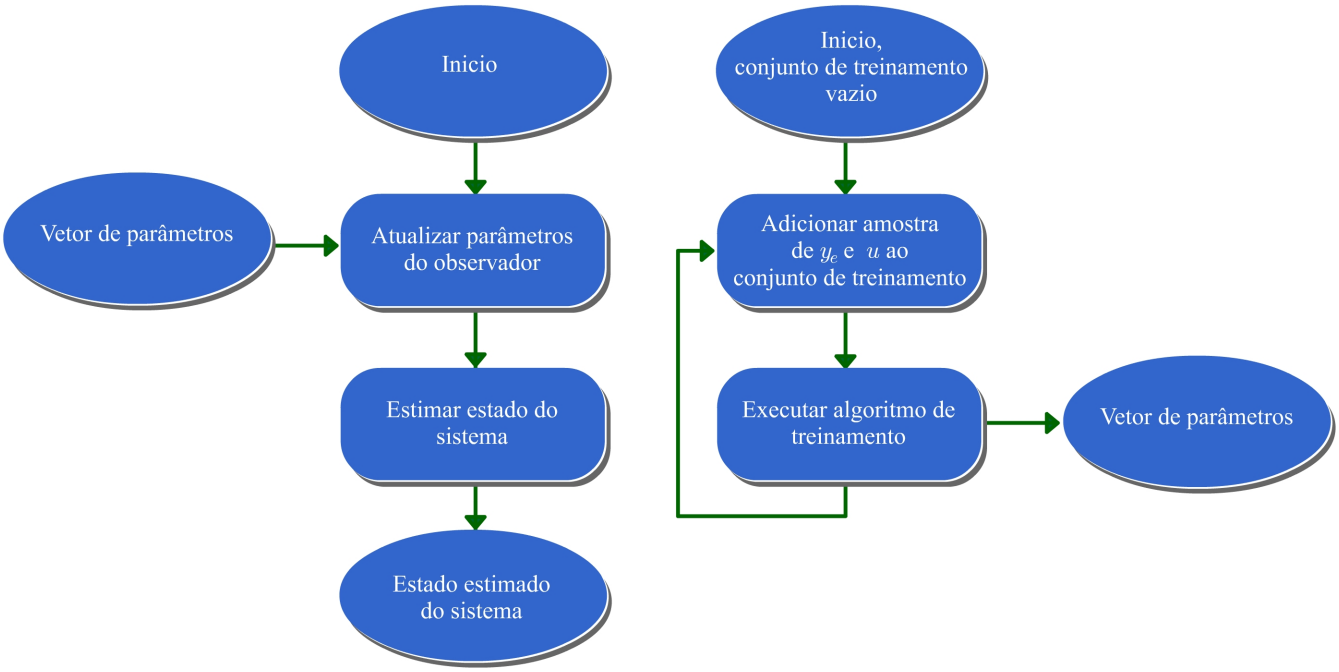

Figura 5.15: Fluxograma de estimação paralela

Novamente, quando o observador adaptativo é aplicado com a finalidade de realimentar o estado para um controlador, a implementação *paralela* é favorável quando se dispõe de mecanismos de computação paralela, situação na qual pode-se realizar ambos os processos de treinamento e de estimação simultaneamente, conseguindo com isso reduzir significativamente o período de amostragem τ, necessário para que o observador seja capaz de apresentar uma amostra do estado estimado do sistema.

5.4.1 Exemplo de Aplicação

Consideremos o sistema não-linear completamente uniformemente observável descrito na equação (5.32). O erro do observador para este sistema é definido pela seguinte equação:

$$e = (y_{e1} - \hat{x}_1^2 + \hat{x}_2 + u)^2 + (y_{e2} - \hat{x}_1\hat{x}_2 + 2\hat{x}_1^2)^2 \tag{5.40}$$

É realizada uma simulação computacional da dinâmica do sistema (5.32), a fim de avaliar a capacidade de estimação do observador proposto. Nesta simulação, é aplicada ao sistema a entrada u descrita na Figura 5.16, durante 10 segundos, período de tempo suficiente para a estabilização da dinâmica do estado. A saída do sistema e sua primeira derivada temporal (vetor y_e), observadas durante a simulação em questão, são mostrados na Figura 5.17.

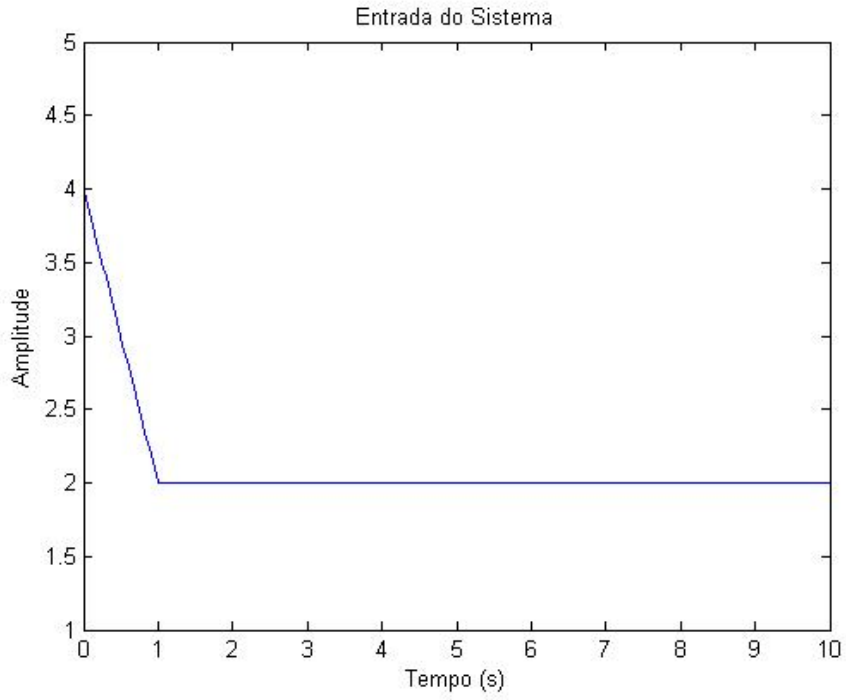

Figura 5.16: Entrada u, exemplo de aplicação: OAN

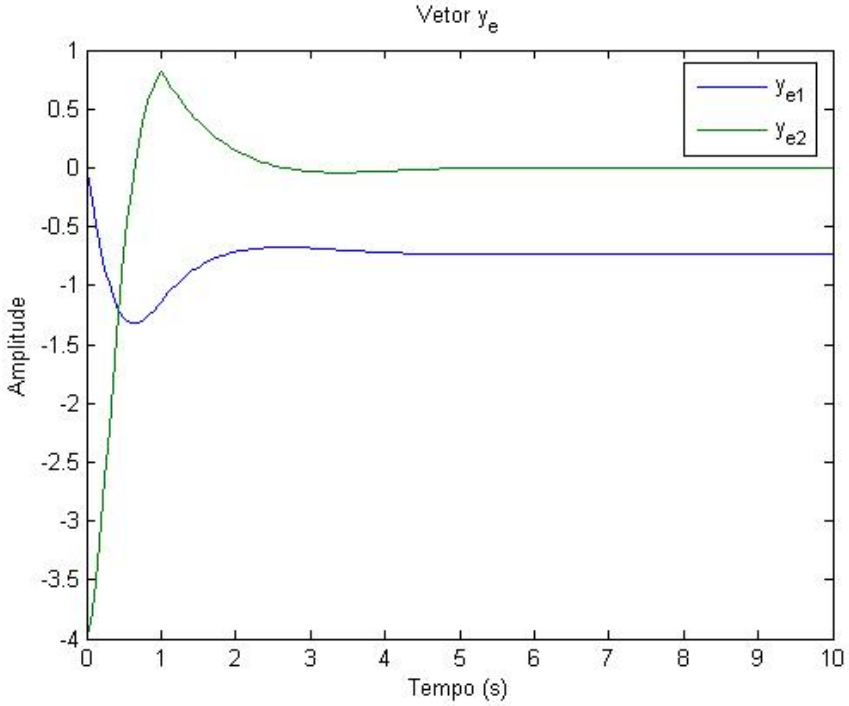

Figura 5.17: Vetor y_e, exemplo de aplicação: OAN

Foi adotado aqui um período de amostragem $\tau = 0.05$ segundos, o que gera um conjunto de treinamento final contendo 201 amostras. Como esse número é relativamente pequeno, não foi aqui limitado o tamanho desse conjunto, visto que a simulação tem uma duração determinada e não gera problemas de armazenamento. O observador adaptativo neural proposto é implementado na estimação do estado do sistema, utilizando a abordagem sequencial. A rede MLP aqui utilizada é aquela descrita na Figura 5.3, que contém apenas uma camada intermediária com 8 neurônios e os mesmos parâmetros utilizados anteriormente para o observador neural.

O resultado desse processo de estimação pode ser verificado na Figura 5.18, onde são plotados simultaneamente os estados estimado e real do sistema, durante a simulação em questão. Esse resultado demonstra a real capacidade de adaptação do observador proposto, assimilando corretamento o mapeamento $\mathcal{H}^{-1}$, durante essa simulação.

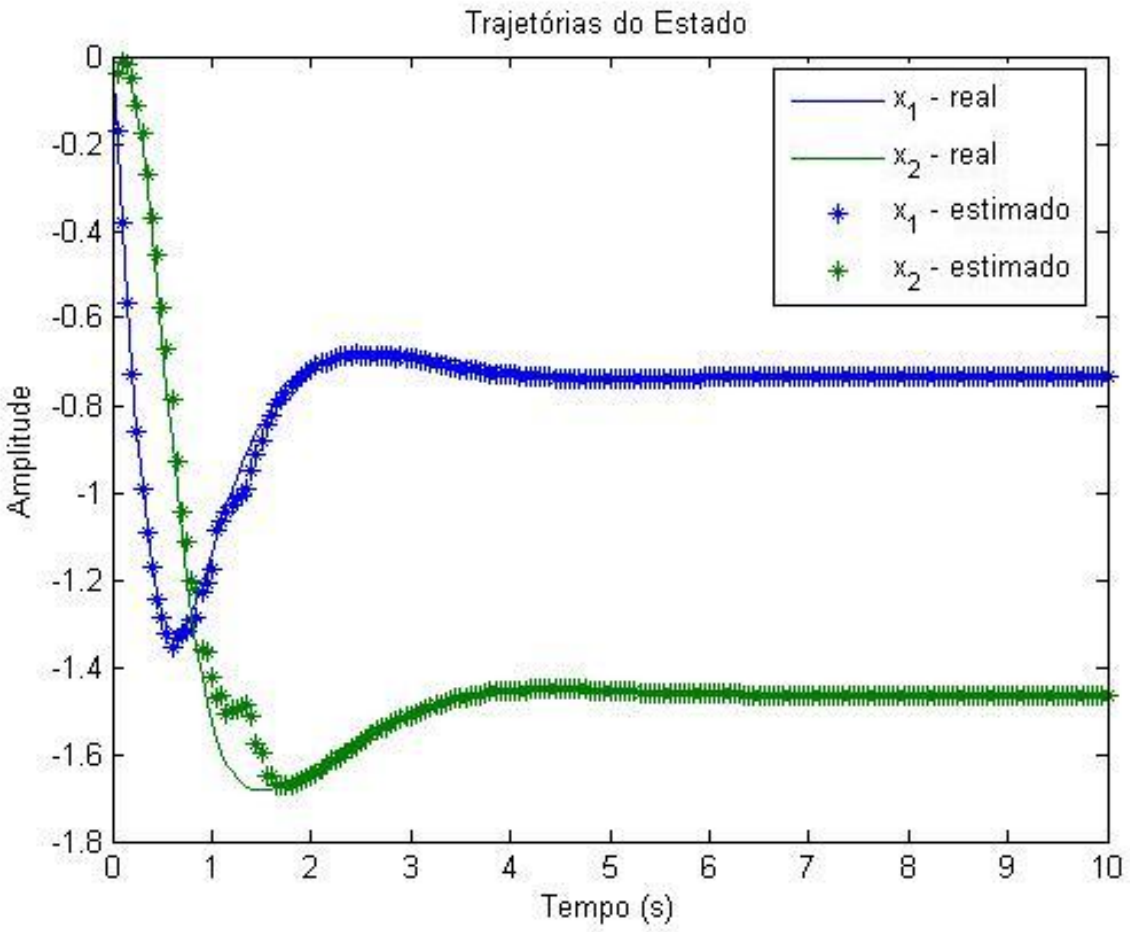

Figura 5.18: Estados estimados: OAN

Nota-se claramente na Figura 5.18 que o erro de estimação do observador diminui com relação ao tempo, o que é esperado de uma ferramenta adaptativa. No gráfico da Figura 5.19, são plotados, em escala logarítmica, os erros de estimação das componentes do estado x_1 e x_2, calculadas a título de análise com base na equação (5.41), que compara o estado estimado com o estado real do sistema.

$$erro_i = (\hat{x}_i - x_i)^2 \tag{5.41}$$

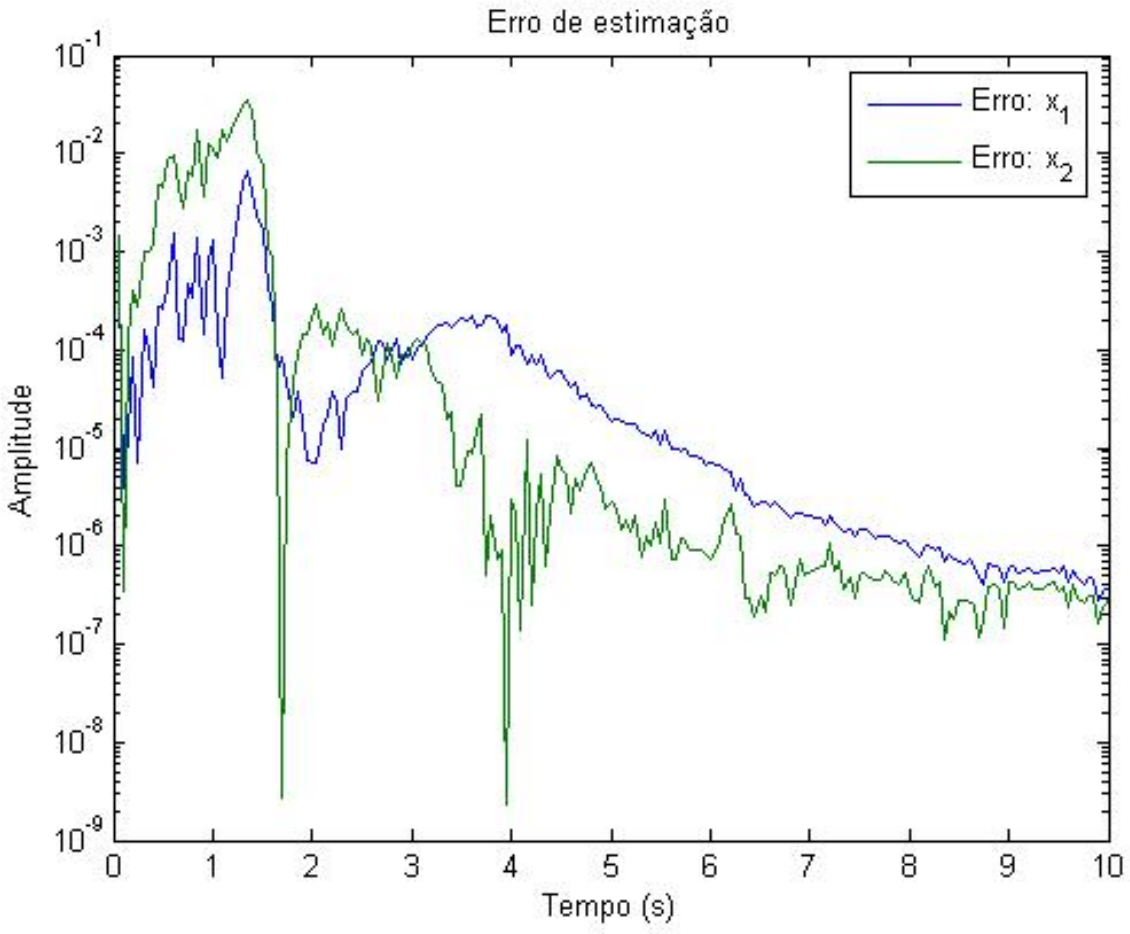

Figura 5.19: Erro de estimação: OAN

Consideremos agora a Figura 5.20, que ilustra a estimativa do estado apresentada pelo observador utilizando o vetor de parâmetros ω obtido ao final do processo de adaptação anteriormente descrito. Nesse estimação, foram utilizados dados do conjunto de treinamento contendo amostras dos primeiros 5 segundos da simulação. Esse resultado ilustra claramente o baixo erro de estimação obtido pelo observador para todas as amostras do conjunto considerado, o que é bastante relevante, dado que o processo de treinamento executa apenas uma única época a cada nova amostra, portanto, o observador

foi treinado por 200 épocas utilizando a primeira amostra, obtida para $t = 0.05$ segundos, e apenas 100 épocas do treinamento são realizadas considerando a amostra para o tempo $t = 5$ segundos.

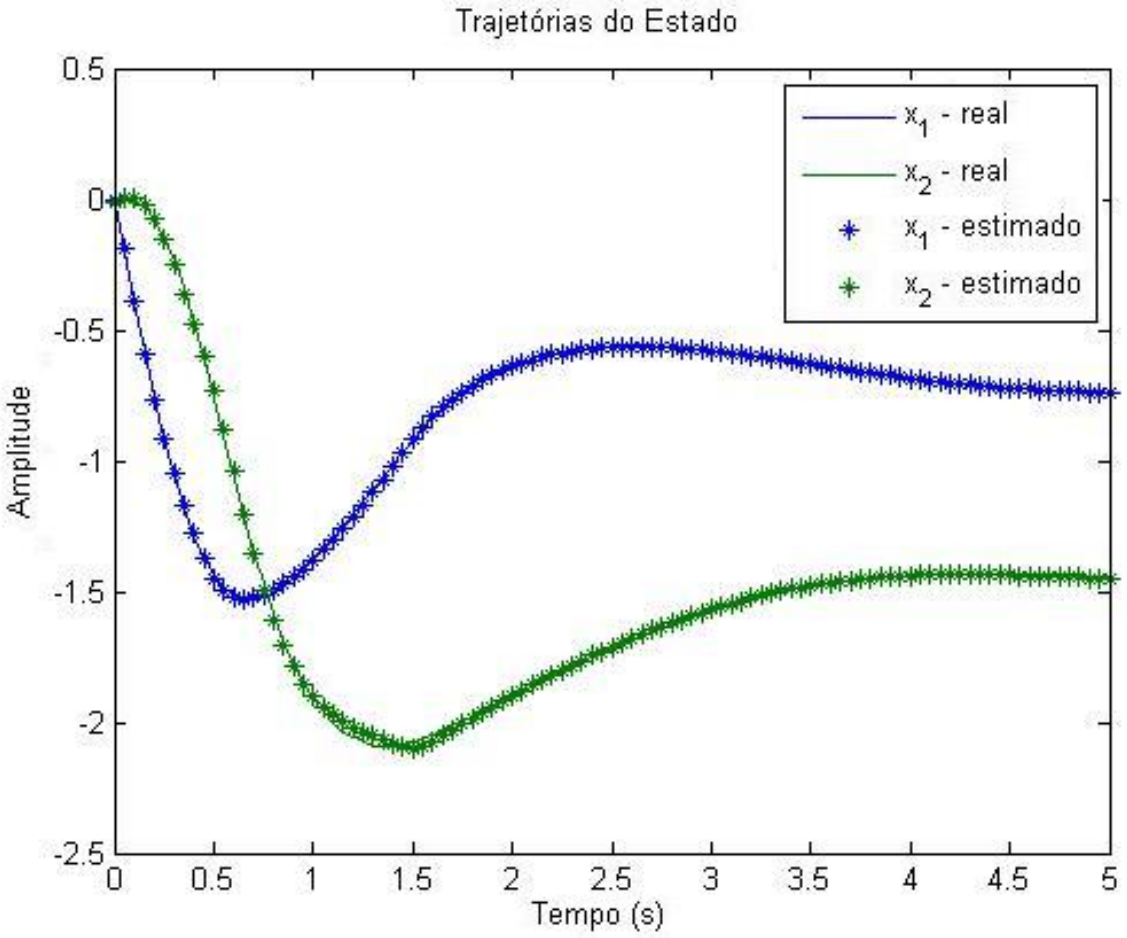

Figura 5.20: Estados estimados utilizando observador previamente treinado: OAN

A fim de avaliar o comportamento do observador adaptativo proposto, com relação a sua sensibilidade relativa à variações na topologia da rede neural, vários outros testes foram realizados com números diferentes de neurônios e de camadas intermediárias. A título de exemplo, consideremos desta vez o observador constituído de uma rede MLP contendo apenas 4 neurônios em sua única camada intermediária, o que representa exatamente a metade do número utilizado anteriormente. A Figura 5.21 mostra a estimativa do estado obtida pelo observador com essa configuração, quando esse é aplicado a exatamente o mesmo conjunto de amostras analisado anteriormente. Nesta aplicação são consideradas também, as mesmas condições de implementação, uma época do treinamento por amostra e período de amostragem $\tau = 0.05$ segundos.

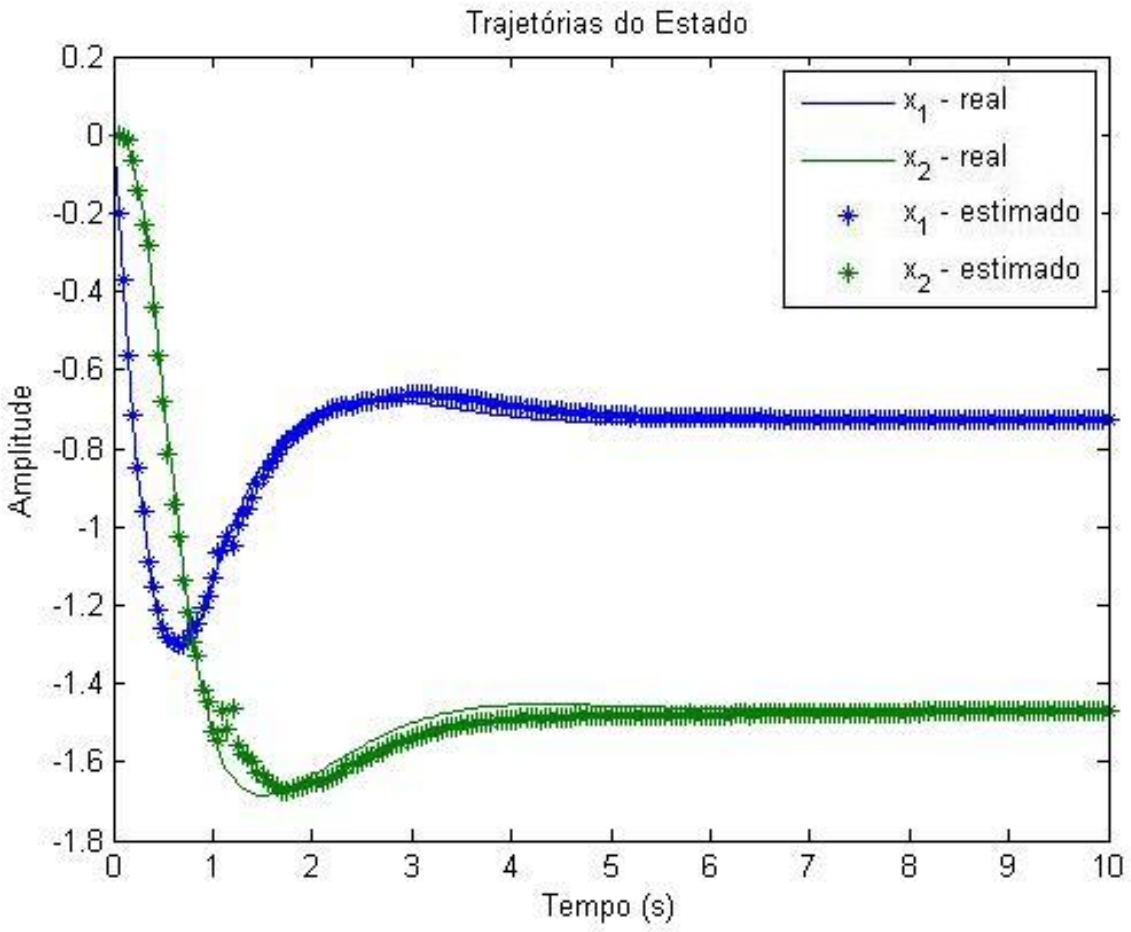

Figura 5.21: Estado estimado com 4 neurônios na camada intermediária: OAN

O resultado obtido com a implementação dessas duas configurações para o observador adaptativo é bastante semelhante. Dada a característica de estocasticidade, introduzida pela inicialização aleatória do vetor de parâmetros ω e pela meta-heurística utilizada no algoritmo de treinamento, não é possível inferir sobre a existência de uma real diferença entre os resultados para as duas configurações em questão.

Outras configurações também foram testadas contendo um número diferente de neurônios na camada intermediária, e também foi avaliado o caso onde a rede possui mais do que uma camada intermediária. Em todas as análises foram obtidos resultados bastante semelhantes, diferindo basicamente no tempo de processamento, dado que cada configuração apresenta um número diferente de variáveis livres N, onde $\omega \in R^N$. Esta análise demonstra que o observador inteligente adaptativo proposto é bastante robusto com relação aos parâmetros topológicos da rede MLP. Sendo assim, a definição desses parâmetros não representa um fator crítico na construção do observador adaptativo neural proposto, o que facilita sua concepção na fase de projeto.

Essa robustez apresentada pelos observadores neurais propostos (OBN e OAN), é uma característica herdada da computação neural. Como já foi aqui discutido, uma das principais características do processamento biológico da informação, emulada pelas redes neurais artificiais, é robustez com relação ao número de neurônios da rede. A definição de de número de camadas intermediárias e do número de neurônios para aplicação de uma rede MLP, apesar de variar de forma relevante dependendo do problema ao qual essa será aplicada, não requer uma precisão muito grande, sendo considerável a gama de valores que podem ser atribuídos a essas variáveis em cada aplicação.

5.5 Observador Evolutivo: OBE

Os algoritmos genéticos são comumente aplicados em procedimentos de busca e otimização. Sendo o problema de observação abordado com base no difeomorfismo entre os espaços $\mathcal{Y} \times \mathcal{U}$ e $\mathcal{X}$, caracterizado por um problema de aproximação, isto não dá margem à aplicação de técnicas evolutivas de forma direta. É possível, porém, a implementação de uma abordagem alternativa para o problema de estimação de estado, possibilitando a aplicação de técnicas de busca na obtenção da estimativa do estado.

O problema de se estimar o estado de um sistema qualquer é basicamente de encontrar o estado do sistema com base em variáveis mensuráveis (aqui consideradas y_e e u). É possível então, descrever o estado do sistema como sendo um objetivo a ser buscado. Neste cenário, pode-se definir os cromossomos de uma população como possíveis representantes desse estado, agora um objetivo. Tendo sido definida uma meta para a população, resta apenas definir qual o critério de avaliação dos cromossomos que será aplicado pelo mecanismo de seleção, de forma que os indivíduos da população evoluam para uma solução que atenda ao objetivo do problema.

Para tanto, consideremos uma população de indivíduos, onde cada cromossomo i é caracterizado por um ponto p_i do espaço R^n. Essa população é aqui representada por $\mathcal{F}$ e corresponde ao observador evolutivo proposto.

$$\hat{x} = \mathcal{F}(y_e, u) \tag{5.42}$$

onde $\hat{x}$ representa o melhor indivíduo da população.

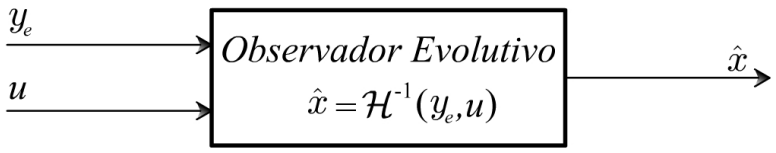

Figura 5.22: Observador Evolutivo

Consideremos agora o sistema não linear genérico (5.1), completamente uniformemente observável. O erro cometido em se utilizar $\hat{x}$ como estimativa para o estado x do sistema é dado por:

$$erro_o = \sum_{i=1}^{n} (\hat{x}_i - x_i)^2 \tag{5.43}$$

O problema, neste caso, é que não se conhece x de forma a calcular $erro_o$ utilizando a equação (5.43). Porém, da mesma forma como foi sugerido para o observador adaptativo neural, é possível definir o erro de estimação com base nos vetores y_e e u, equação (5.44).

$$erro_o(\hat{x}, y_e, u) = \sum_{i=1}^{n} (\mathcal{H}(\hat{x}, u)_i - y_{e_i})^2 \tag{5.44}$$

Dessa forma não é preciso conhecer o estado real do sistema para definir uma métrica para o erro cometido em sua estimação. Com base no erro de estimação $erro_o$, definido na equação (5.44), é possível avaliar a adaptação de cada indivíduo da população, que deve ser tanto maior quanto menor for o erro por ele apresentado e estritamente positiva. Com base nesses requisitos é definida uma proposta para função de *fitness* do observador, equação (5.45):

$$fitness_i = e^{-erro_o(p_i, y_e, u)\rho} \tag{5.45}$$

onde p_i é um ponto do espaço R^n representado pelo cromossomo i e ρ é um parâmetro a ser escolhido pelo projetista, que irá afetar diretamente a diversidade da população.

Existem ainda outros parâmetros a serem definidos durante o projeto do observador evolutivo. São eles:

- **Número de indivíduos da população.** Este número deve ser escolhido pelo projetista de forma empírica, baseando-se principalmente na dimensão do estado n.
- **Número de gerações evoluídas a cada amostra.** Este parâmetro é definido com base na precisão que se deseja obter em cada estimativa do estado, sendo essa tanto maior quanto maior o número de gerações utilizadas em sua busca. Porém, quanto maior for esse número, maior também será o tempo demandado na obtenção dessa estimativa.
- **Mecanismo de seleção.** É utilizado um mecanismo de seleção elitista na escolha da geração seguinte, e uma seleção tipo roleta é aplicada aos pares para reprodução, visando com isso, preservar a diversidade da população e diminuir os riscos de uma convergência prematura. Porém, a escolha de outros mecanismos de seleção é possível e pode levar a resultados igualmente satisfatórios.

- **Operadores genéticos.** A determinação dos operadores de reprodução e mutação que serão aplicados à população é também tarefa do projetista. Neste documento, são expostos vários desses operadores, sendo todos aqueles baseados na codificação real aplicáveis ao observador proposto (uma proposta binária não é aqui discutida), bem como outros possíveis operadores encontrados na literatura. Nas análises aqui realizadas, são utilizados os operadores de reprodução matricial e mutação vetorial.

Essa configuração proposta permite a população de indivíduos estimar o estado instantâneo do sistema, com base em uma amostra das trajetórias do vetores $y_e(t)$ e $u(t)$ em um instante de tempo t. Na definição do sistema mostrada em (5.1) foram utilizadas funções f e h contínuas e Lipschitz, o que garante que o estado do sistema vai descrever trajetórias continuas no espaço $\mathcal{X}$. Essa característica implica no seguinte limite:

$$\lim_{\tau \to 0}(x(t) - x(t+\tau)) = 0 \;\; \forall\, t \in R \tag{5.46}$$

Dessa forma, se for conhecido o estado do sistema $x(t_0)$ no instante de tempo t_0, é possível afirmar que o estado $x(t_0 + \tau)$ no instante $t_0 + \tau$ pertence a uma vizinhança $\mathcal{V}$ de $x(t_0)$ para algum τ suficientemente pequeno. Assim, se o observador evolutivo for utilizado na estimação do estado $x(t_0)$ a busca pela estimativa do estado $x(t_0 + \tau)$ pode ser facilitada inicializando-se a população de indivíduos inserida na menor vizinhança $\mathcal{V}$ que contém $x(t_0 + \tau)$. Essa abordagem é bastante útil quando o observador é utilizado para fornecer estimativas do estado de forma continuada, como é o caso quando o observador é empregado para realimentar o estado do sistema para um controlador. Neste caso, faz-se necessária a manutenção de uma estimativa atualizada do estado do sistema.

Uma maneira de implementar esse processo, é utilizar a mesma população na estimação de estados consecutivos, assim, é gerada uma população inicial aleatória e estimado o primeiro estado $x(t_0)$, posteriormente essa mesma população é utilizada para estimar o estado seguinte $x(t_0 + \tau)$ e assim sucessivamente. Essa abordagem pressupõe que a diversidade final da população vai ser grande o suficiente para cobrir um espaço que contenha o próximo estado; se isso não acontecer o observador pode vir a apresentar um erro de estimação elevado. Esse alto erro é obtido devido a diversidade final da população ser geralmente muito pequena e com isso torna-se difícil encontrar um objetivo que esteja distante da população.

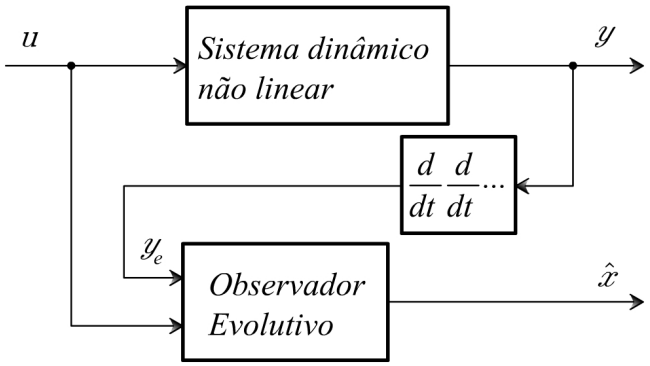

Figura 5.23: Estimação de estado utilizando observador evolutivo

Outra maneira de implementar o observador evolutivo de forma a estimar estados consecutivos, ainda utilizando a continuidade das trajetórias do estado como um facilitador a esse processo, é inicializar a população de forma aleatória, em um subespaço pequeno que contenha o estado encontrado anteriormente. Fazendo isso, é possível atribuir a diversidade necessária para dispersar a população dentro de um subespaço $\mathcal{V}$ que contenha o estado procurado.

Nesta abordagem, inicialmente, é tomada uma população inicial aleatória contida em um hipercubo $H(t_0)$ centrado da origem do espaço $\mathcal{X}$; é então estimado o primeiro estado $\hat{x}(t_0)$; para estimar o estado seguinte, a população é iniciada aleatoriamente também dentro de um hipercubo $H(t_0 + \tau)$, porém, centralizado no ponto $\hat{x}(t_0)$ e possuindo um lado de comprimento pequeno, porém, grande o suficiente para conter $x(t_0 + \tau)$. A Figura 5.24 mostra o fluxograma de implementação para o observador evolutivo utilizando essa abordagem.

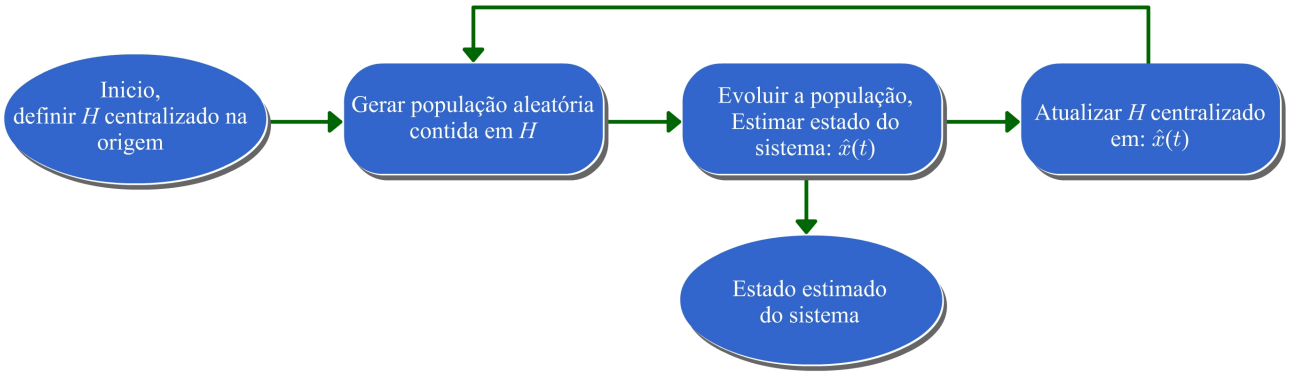

Figura 5.24: Fluxograma de implementação do observador evolutivo

Na abordagem adotada na concepção desse observador, é utilizada uma relação entre subespaços jamais explorada desta maneira na teoria de controle. Aqui, uma população de indivíduos é confinada no espaço de estado $\mathcal{X}$, onde tem liberdade para se locomover, porém, seu objetivo é desconhecido.

É então utilizado o mapeamento $\mathcal{H}$ para mapear a posição de cada um desses indivíduos no espaço $\mathcal{Y} \times \mathcal{U}$, no qual é possível criar uma métrica da distância entre o ponto mapeado pelo indivíduo, e o ponto onde encontram-se os vetores y_e e u. Dessa forma, observar o sistema, reduz-se a tarefa de minimizar essa distância, o que pode ser feito de inúmeras maneiras, sendo que a suposição de observabilidade uniforme, garante que apenas o real estado do sistema irá mapear um ponto em $\mathcal{Y} \times \mathcal{U}$ tal que a distância desse ponto à $[y_e^T u^T]^T$ seja zero.

5.5.1 Exemplo de Aplicação

Consideremos o sistema dinâmico não-linear completamente uniformemente observável mostrado em (5.32), ao qual é aplicada uma entrada u, descrita pela Figura 5.25. É então realizada uma simulação computacional da dinâmica desse sistema utilizando a entrada em questão, da qual é obtida a resposta y, que, juntamente com sua primeira derivada temporal, formam o vetor y_e que é mostrado na Figura 5.26.

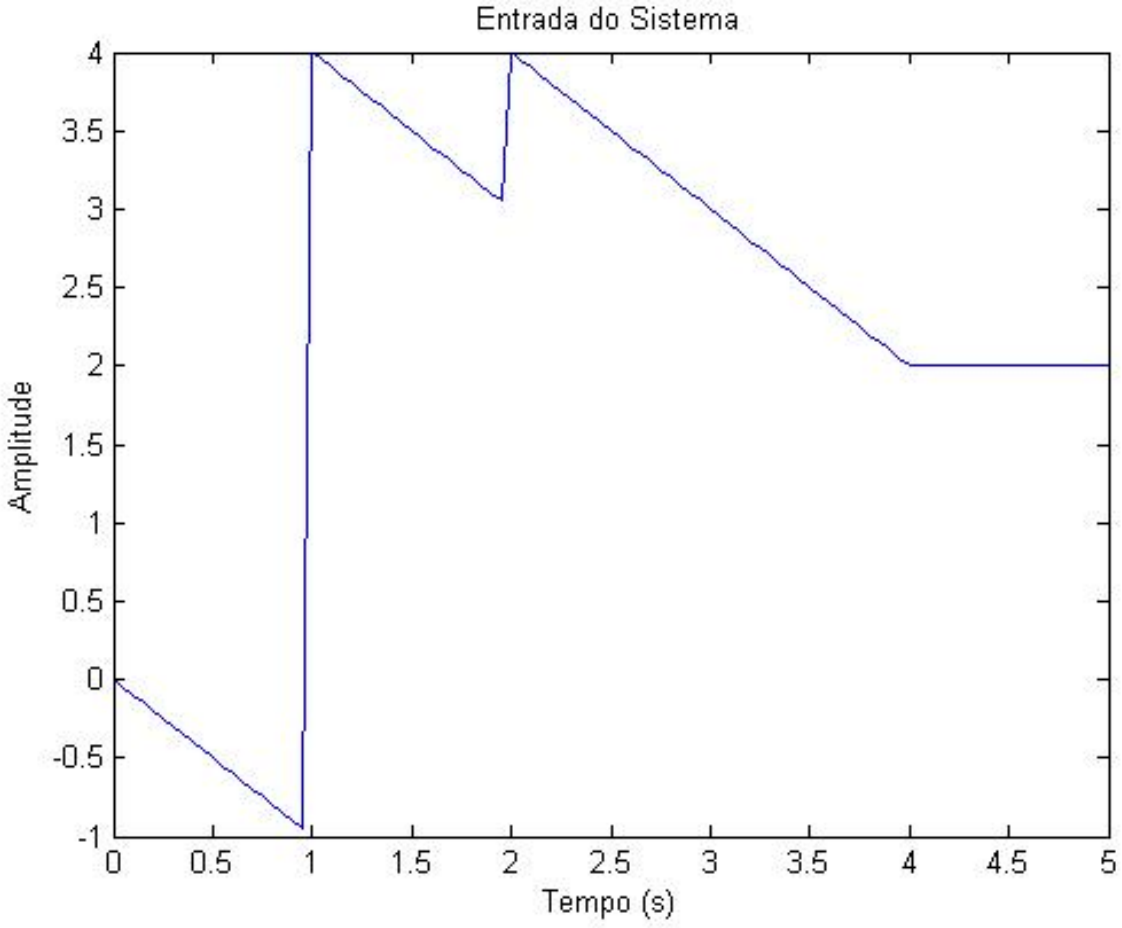

Figura 5.25: Entrada do sistema: OBE

O observador evolutivo proposto é então utilizado para estimar o estado do sistema quando é aplicada a entrada u em questão, durante um período de tempo de 5 segundos. O período de amostragem aqui adotado para esta análise é de: $\tau = 0.05$ segundos, o que fornece um total de 100 amostras durante o intervalo de tempo analisado. A aplicação do observador evolutivo requer a definição de uma série de parâmetros, que combinados definem as características do observador e afetam o resultado da estimação do estado. São testadas aqui várias configurações diferentes para os parâmetros do observador, a fim de analisar a sensibilidade com relação a variação desses parâmetros, bem como definir qual escolha é preferível para o problema proposto. Todas as variações testadas são discutidas em detalhes adiante.

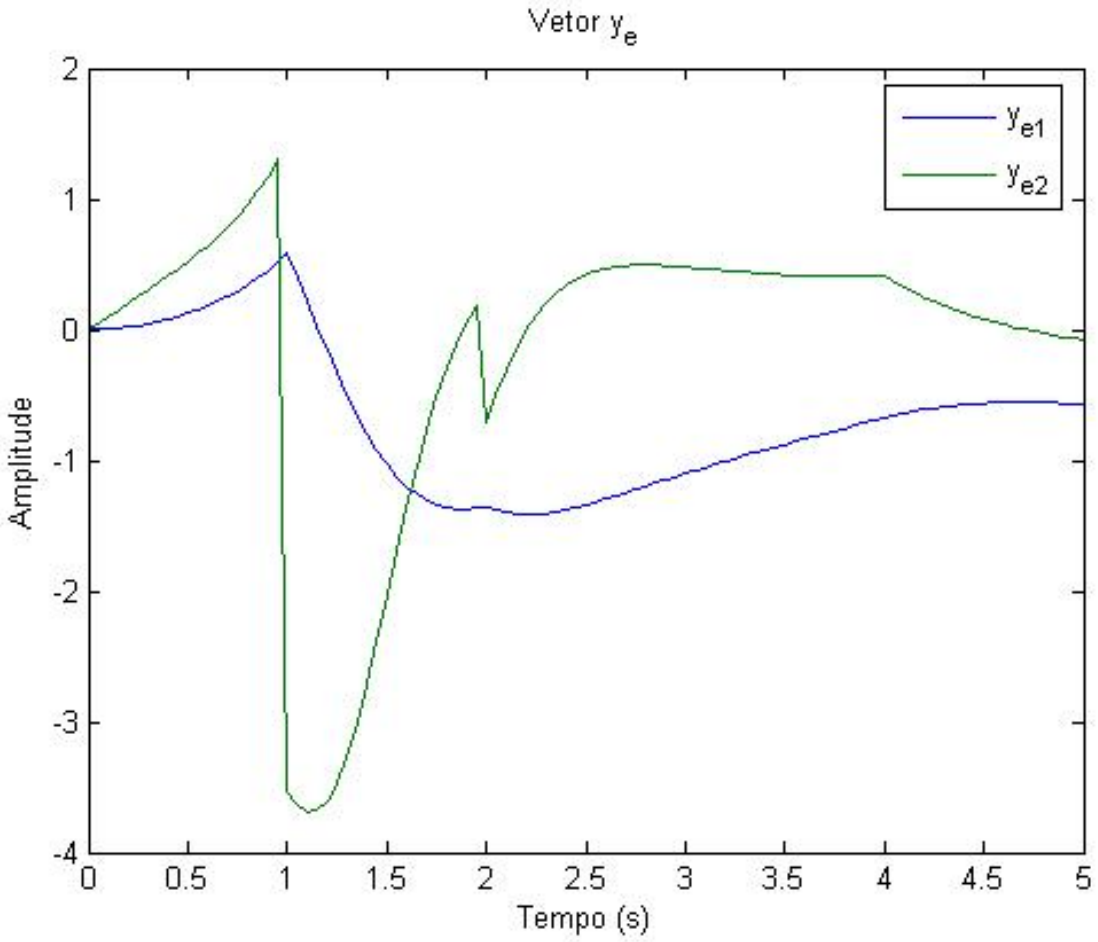

Figura 5.26: Vetor y_e: OBE

Em uma primeira análise, consideremos o observador evolutivo que é definido pelo seguinte conjunto de parâmetros.

Configuração 1:

- Número de partículas: $n_p = 10$
- Número de gerações por estado estimado: $n_g = 10$
- Lado do hipercubo H: $l = 1$

O gráfico mostrado na Figura 5.27 ilustra o resultado obtido pelo observador evolutivo quando se utiliza a configuração proposta anteriormente.

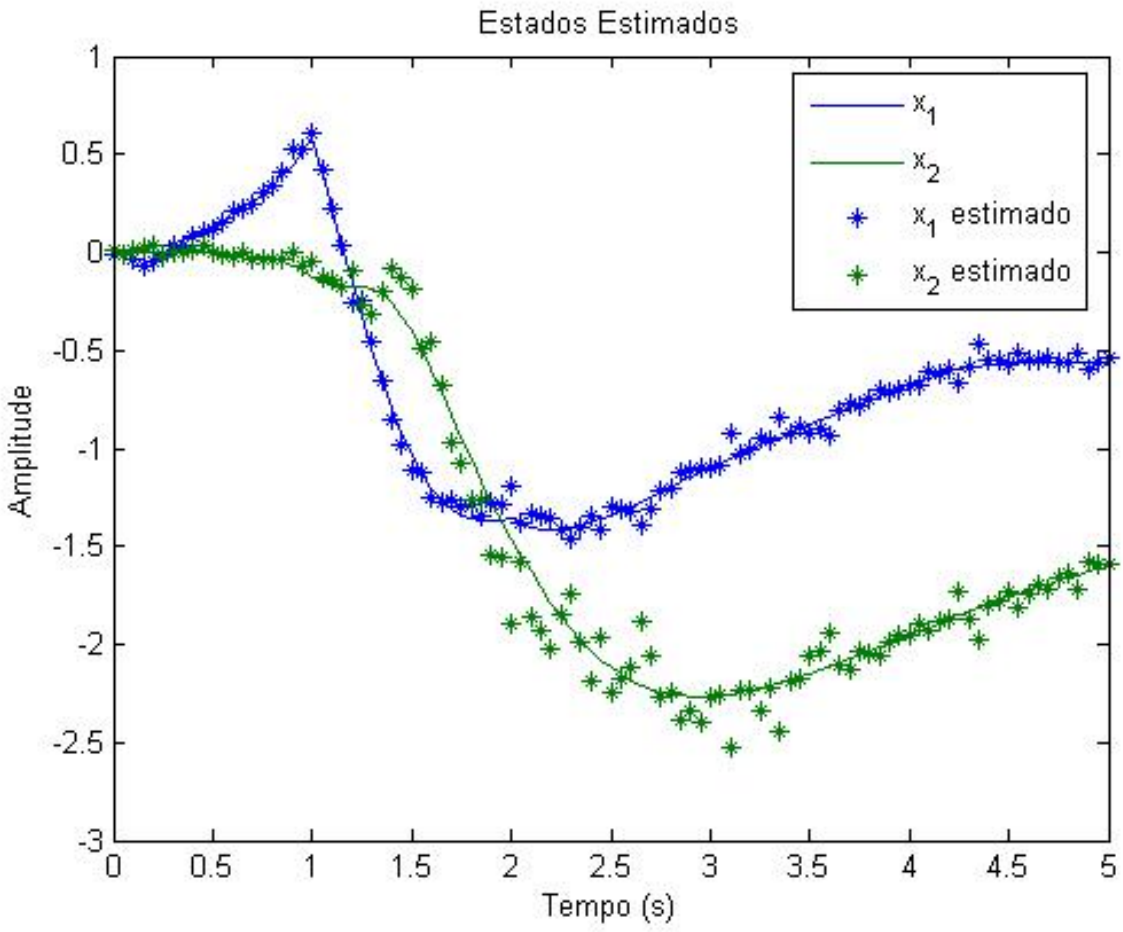

Figura 5.27: Estados estimados, configuração 1: OBE

Na figura anterior, é possível verificar a ocorrência de um erro de estimação considerável em alguns estados, quando esse resultado é comparado com aqueles apresentados por outros observadores propostos neste documento, apesar de não apresentar um erro elevado como pode ser observado na Figura 5.28. Esse resultado insatisfatório pode ser causado por vários fatores que levem a população a não encontrar o seu objetivo, como a utilização de um número muito reduzido de indivíduos ou mesmo quando é considerado um número insuficiente de gerações. Sendo assim, outras configurações para o observador podem render resultados superiores a esse, quando melhor exploradas as potencialidades do algoritmo genético, que apresenta condições suficientes para obter resultados com um erro de estimação consideravelmente inferior aos observados para a configuração testada.

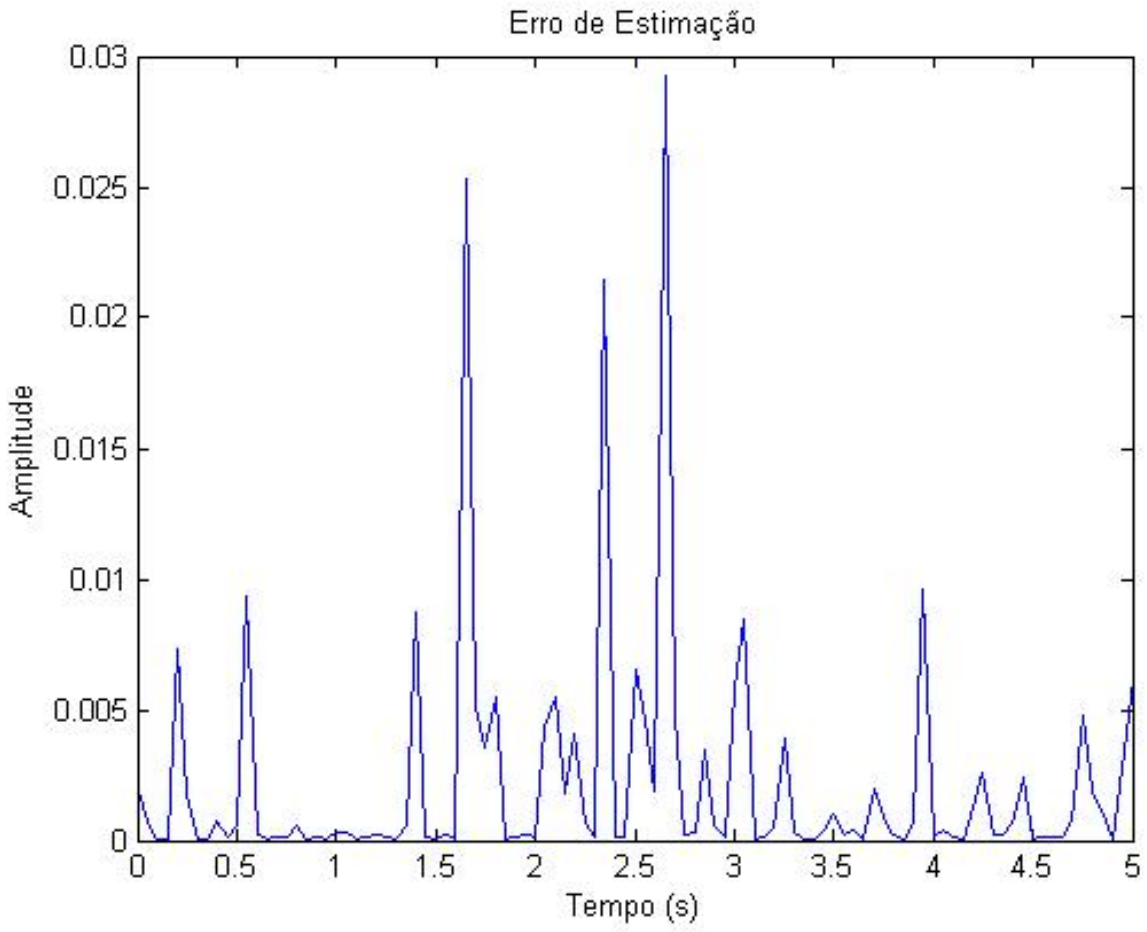

Figura 5.28: Erro de estimação, configuração 1: OBE

Em uma nova análise, consideremos um observador que implemente um número maior de gerações na estimação de cada estado, tentando com isso alcançar um erro de estimação inferior àquele apresentado anteriormente. Para tanto é utilizada a seguinte configuração para o observador:

Configuração 2:

- $n_p = 10$
- $n_g = 100$
- $l = 1$

O resultado da aplicação do observador com essa configuração ao problema em questão, pode ser observado no gráfico descrito na Figura 5.29.

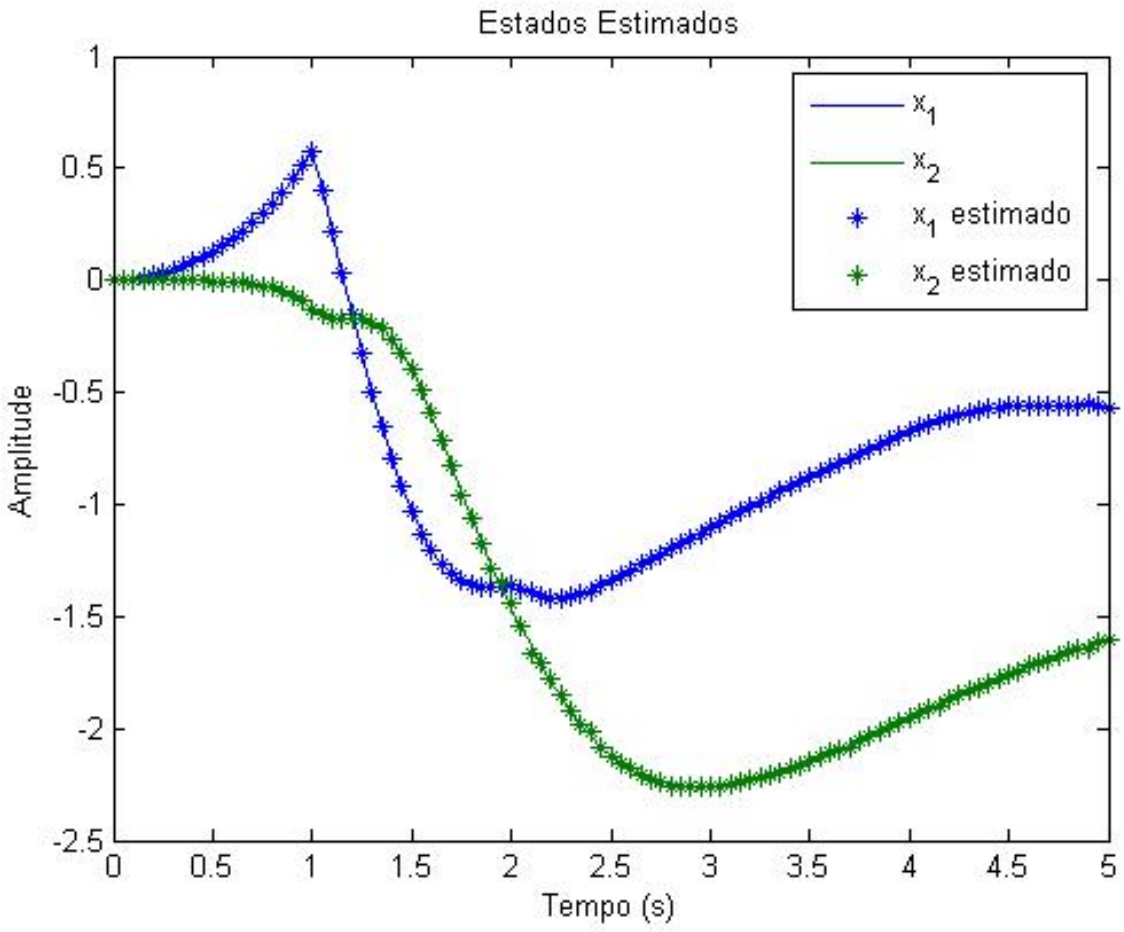

Figura 5.29: Estados estimados, configuração 2: OBE

Esse resultado é claramente superior àquele alcançado anteriormente, apresentando um erro de estimação bastante reduzido quando comparado com o anterior. Na configuração 2, foi mantido o tamanho da população e aumentado apenas o número de gerações implementadas na obtenção de cada estimativa. Com isso, o algoritmo genético foi capaz de alcançar pontos mais próximos a seu objetivo. Outra maneira de melhorar o resultado da estimação é aumentar o número de indivíduos da população, com isso, a capacidade do algoritmo genético deve também aumentar e conseqüentemente espera-se que o resultado seja superior.

Existe, porém, outro parâmetro ainda não explorado: o tamanho do subespaço onde a população é inicializada a cada nova amostra. Nos observadores testados anteriormente, a população estava sendo inicializada em um hipercubo de lado $l = 1$, o que representa um espaço de busca consideravelmente grande para o problema. Se esse espaço for reduzido, possivelmente os indivíduos terão uma condição inicial melhor e o algoritmo genético então pode alcançar seu objetivo com um número menor de gerações.

A fim de avaliar o resultado da estimação obtido pelo observador evolutivo, quando o algoritmo genético é inicializado utilizando um espaço de busca inicial reduzido com relação àquele anteriormente testado, consideremos o observador descrito pela seguinte configuração.

Configuração 3:

- $n_p = 10$
- $n_g = 10$
- $l = 0.1$

A Figura 5.30 descreve o resultado obtido pela aplicação do observador utilizando a configuração 3.

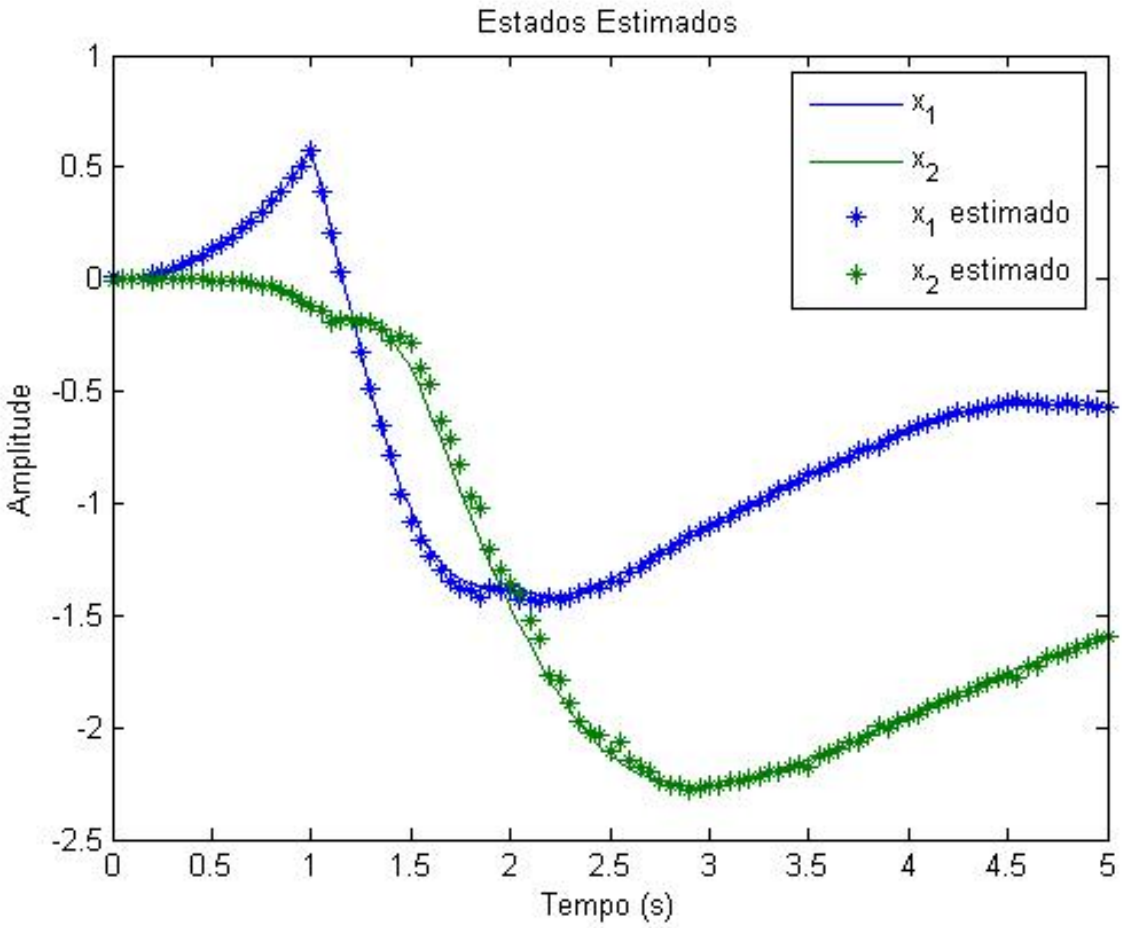

Figura 5.30: Estados estimados, configuração 3: OBE

Nessa configuração, o número de gerações foi reduzido consideravelmente quando comparado ao número utilizado na configuração 2; o resultado obtido, porém, também é inferior, o que é coerente,

pois foi reduzida a capacidade do algoritmo genético. Por outro lado, a configuração 3 apresenta as mesmas características da configuração 1, diferindo apenas na dimensão do espaço de busca inicial, o que não afeta a complexidade computacional do observador, conseguindo com isso um ganho significativo em qualidade do resultado.

Como última análise, é avaliado o terceiro parâmetro considerado na definição do observador evolutivo: o tamanho da população. Tenta-se aqui avaliar a qualidade da estimativa do estado obtida com a variação desse parâmetro. Para tanto, consideremos agora a seguinte configuração:

Configuração 4:

- $n_p = 100$
- $n_g = 10$
- $l = 1$

O resultado obtido com a aplicação do observador utilizando a configuração 4 é mostrado no gráfico descrito na Figura 5.31.

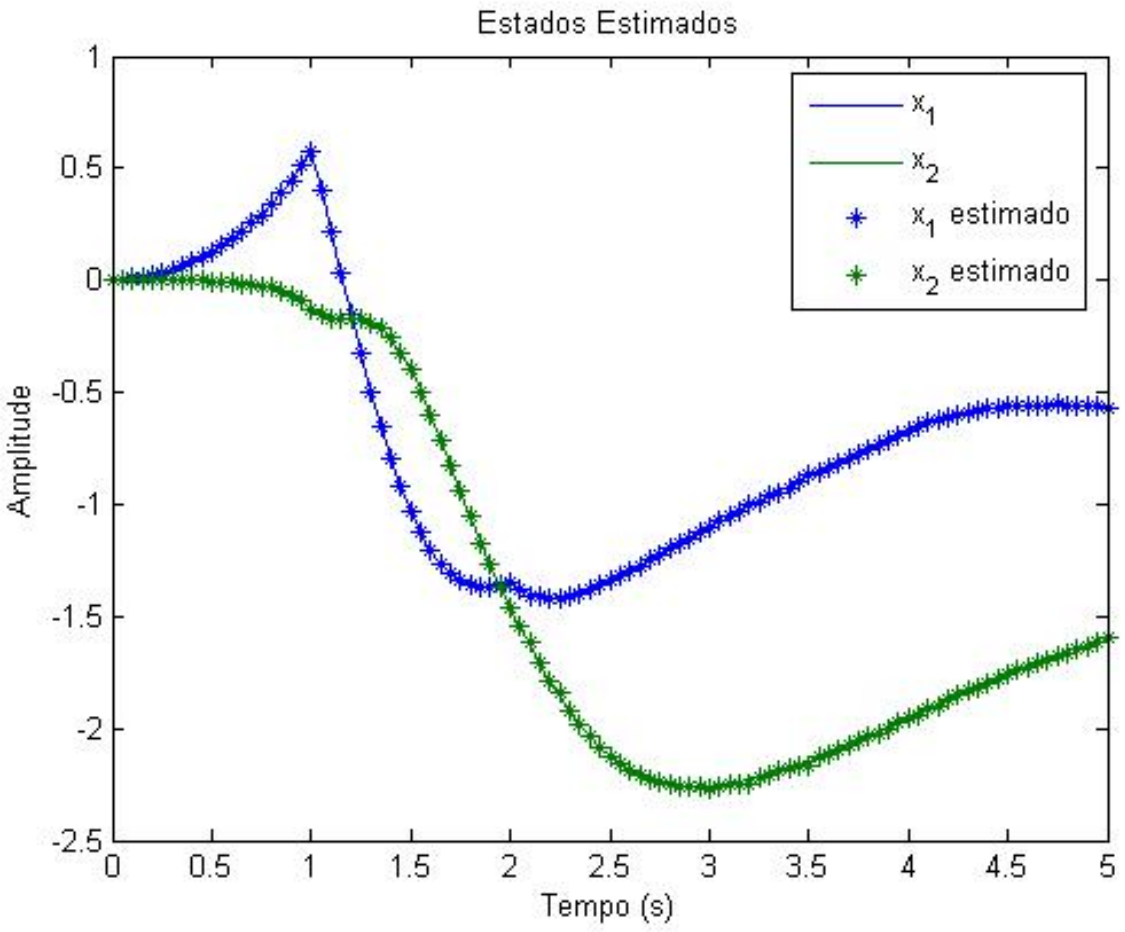

Figura 5.31: Estados estimados, configuração 4: OBE

Esse resultado é bastante semelhante àquele obtido com a utilização da configuração 2, em que foram implementadas 100 gerações por estimativa. As quatro configurações aqui avaliadas têm como objetivo descrever a sensibilidade do observador evolutivo com relação a seus parâmetros de projeto, possibilitando a sua implementação de forma eficiente e explorando integralmente a sua capacidade.

O gráfico da Figura 5.32 mostra os erros de estimação obtidos pelas quatro configurações analisadas, plotados em uma escala logarítmica. Esse gráfico permite uma comparação entre a qualidade da solução apresentada por cada uma das configurações, criando uma intuição a respeito da influência da escolha de cada parâmetro no comportamento final do observador.

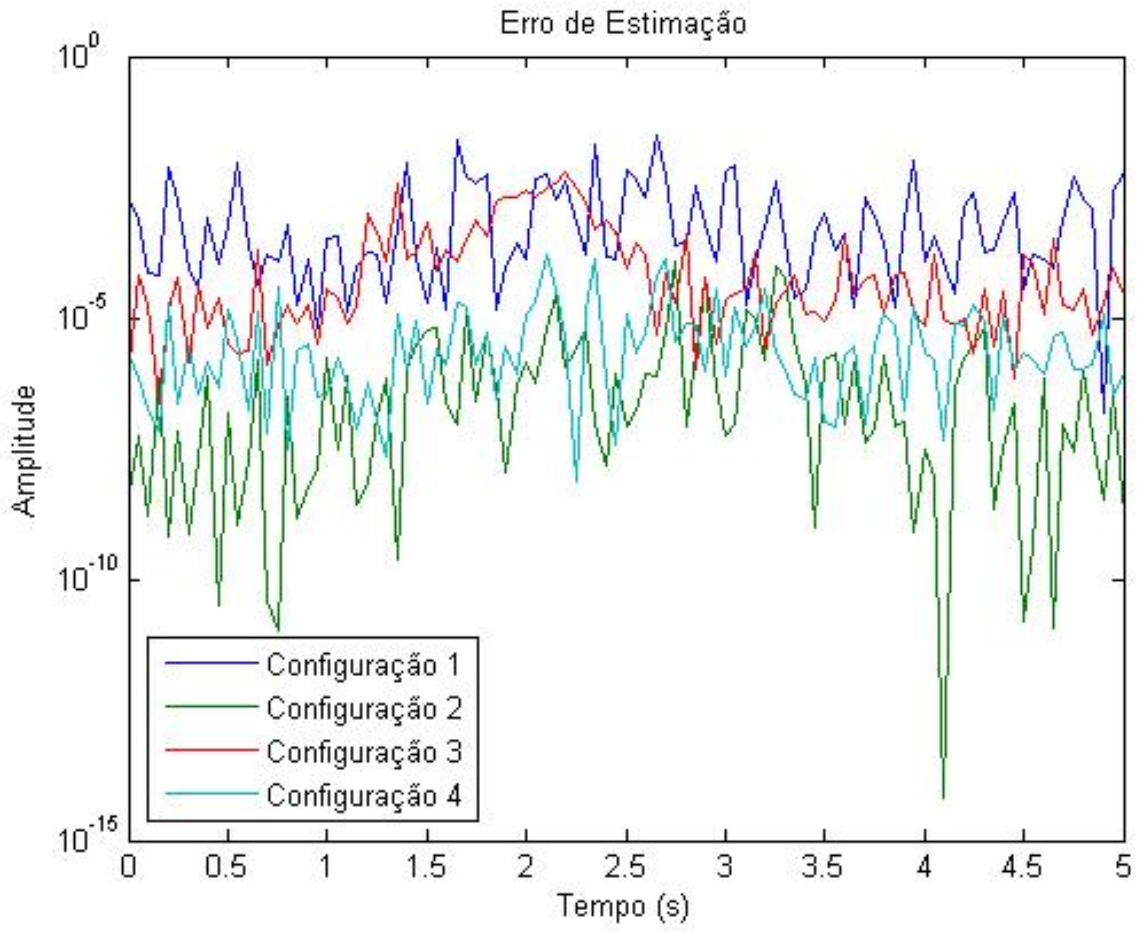

Figura 5.32: Erro de estimação para as quatro configurações analisadas: OBE

5.6 Observador Baseado em Inteligência de Enxame: OEP

O observador baseado em inteligência de enxame: OEP, proposto neste documento, é baseado nos mesmos conceitos que o observador evolutivo discutido anteriormente, dado que ambos utilizam métodos de busca cega na estimação do estado do sistema. A diferença é que neste observador, é implementado um algoritmo de otimização por enxame de partículas, devidamente especializado para estimar o estado do sistema [59].

O algoritmo básico desse observador utiliza as partículas para representar possíveis estados do sistema, que locomovem-se pelo espaço de estado com base nas regras de movimentação descritas anteriormente para essa classe de algoritmos. A avaliação da adaptação de cada partícula, β_i, é feita com base no erro de estimação, através da seguinte equação:

$$\beta_i = \sum_{j=1}^{n} \left(\mathcal{H}(z_i, u)_j - y_{e_j} \right) \tag{5.47}$$

onde z_i representa a posição da partícula i.

Consideremos aqui que o método de otimização por enxame de partículas retorna, ao final do processo iterativo, um valor $\alpha \in R^n$ que representa um ótimo local da função de erro (5.44). Então, é possível descrever o estado estimado do sistema através da equação (5.48):

$$\hat{x} = \mathcal{F}(\dagger_\urcorner, \sqcap) \tag{5.48}$$

onde $\hat{x}$ é o resultado do processo de otimização por enxame de partículas.

Os parâmetros a serem escolhidos na definição do observador são:

- Número de partículas: n_p.
- Número de iterações por estimativa: n_i.
- Tamanho do espaço de busca inicial, definido pelo comprimento do lado do hipercubo H, dado por: l
- Peso de inércia (quando utilizado): w.
- Coeficientes de aceleração: c_1 e c_2.

Neste documento, será utilizado o método de otimização por enxame de partículas com modulação de velocidade. Assim, é dispensável a utilização do peso de inércia w, que não será definido.

A estimação de estados consecutivos segue os mesmos padrões discutidos anteriormente para o observador evolutivo, sendo válida a mesma análise. Este observador oferece uma alternativa à estimação de estado via métodos de busca, e proporciona vantagens em algumas situações, pois, como foi demonstrado neste documento, essas duas técnicas, evolutiva e de enxame, apresentam características diferentes. Os algoritmos genéticos apresentam, em geral, uma diversidade maior da população, enquanto a otimização por enxame de partículas demonstra possuir uma convergência mais acelerada. Em problemas contendo poucas variáveis, a alta diversidade dos algoritmos genéticos acaba levando geralmente a resultados melhores, sendo que, em problemas de ordem mais elevada, é preferível a utilização de otimização por enxame de partículas.

5.6.1 Exemplo de Aplicação

Novamente, consideremos o sistema não-linear completamente uniformemente observável descrito em (5.32), sob as mesmas condições estudadas para o observador evolutivo: entrada u descrita

pela Figura 5.25 e resposta y, que juntamente com sua derivada temporal, formam o vetor y_e descrito na Figura 5.26.

Em uma primeira análise, consideremos o observador com a seguinte configuração:

Configuração 1:

- $n_p = 10$
- $n_g = 10$
- $l = 1$
- $c_1 = c_2 = 1$

O resultado obtido com a aplicação desse observador é mostrado na Figura 5.33.

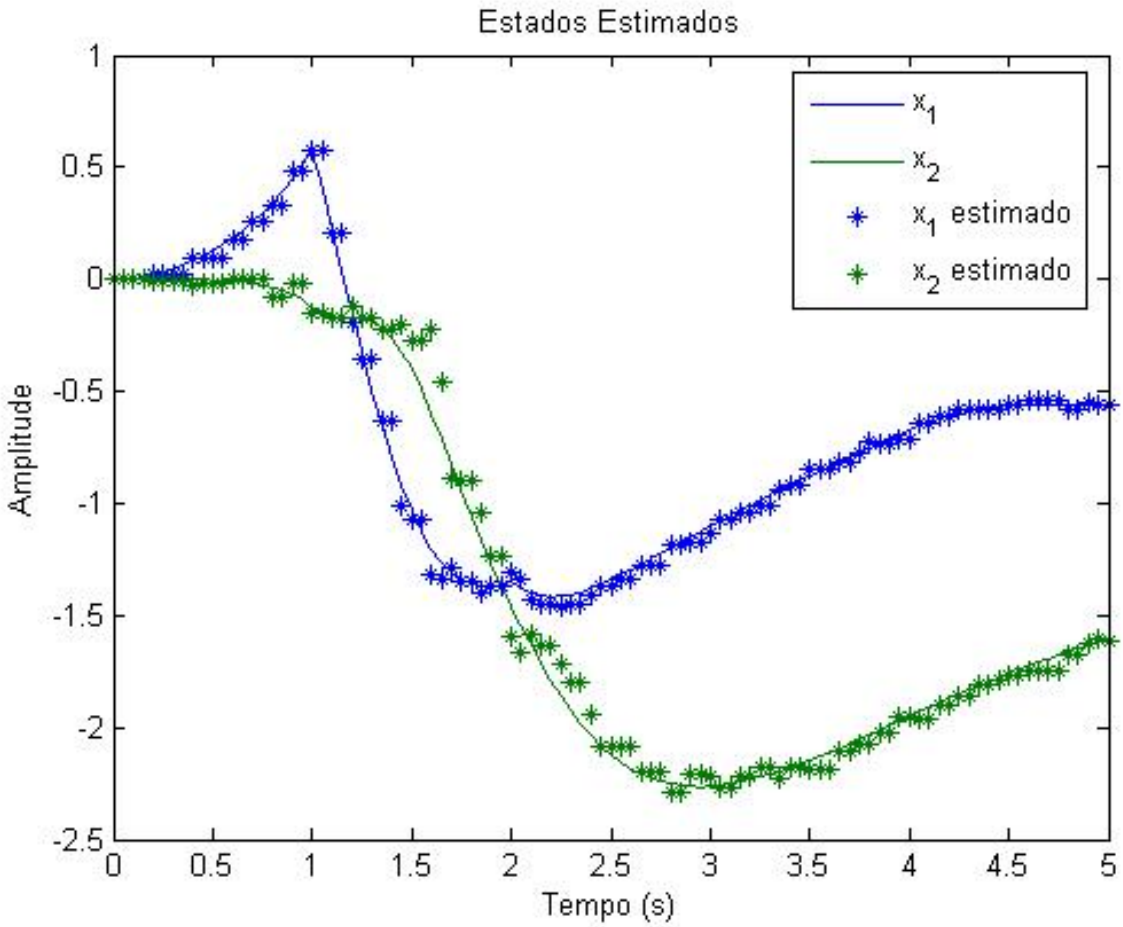

Figura 5.33: Estado estimado, configuração 1: OEP

Esse resultado é bastante similar àquele obtido para o observador evolutivo utilizando um conjunto de parâmetros semelhantes. Dada a característica estocástica do processo de estimação dos dois métodos em questão, não pode-se afirmar a superioridade de nem um deles com relação à qualidade da solução apresentada para este cenário de aplicação.

A Figura 5.34 exibe um gráfico onde é plotado em escala logarítmica os erros de estimação cometidos pelo observador evolutivo (configuração 1) e pelo observador OEP (configuração 1), para o problema em questão.

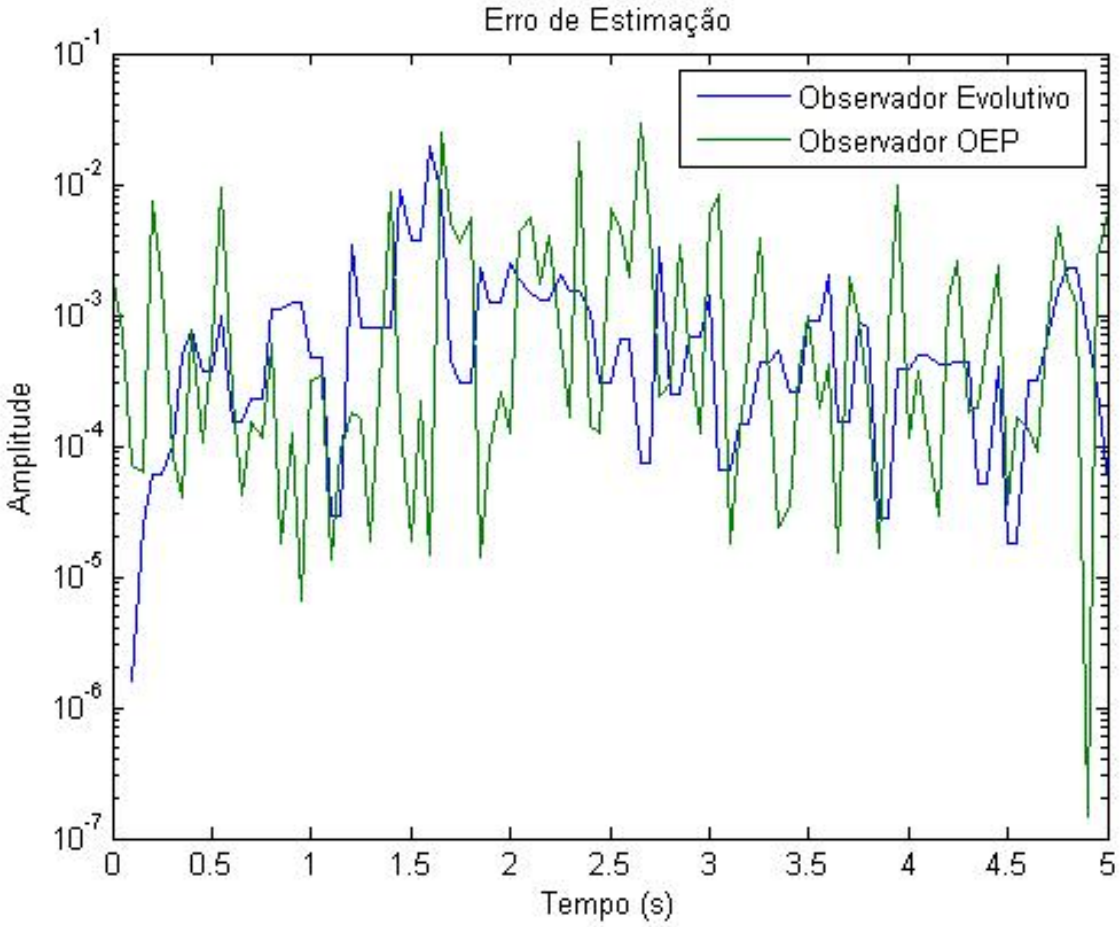

Figura 5.34: Erro de estimação, comparação entre o observador evolutivo e o observador: OEP

Consideremos agora a configuração 2, onde é aumentado o número de partículas da população, visando prover um ganho de eficiência ao método de otimização.

Configuração 2:

- $n_p = 100$

- $n_g = 10$
- $l = 1$
- $c_1 = c_2 = 1$

O resultado da aplicação desse observador é mostrado na Figura 5.35.

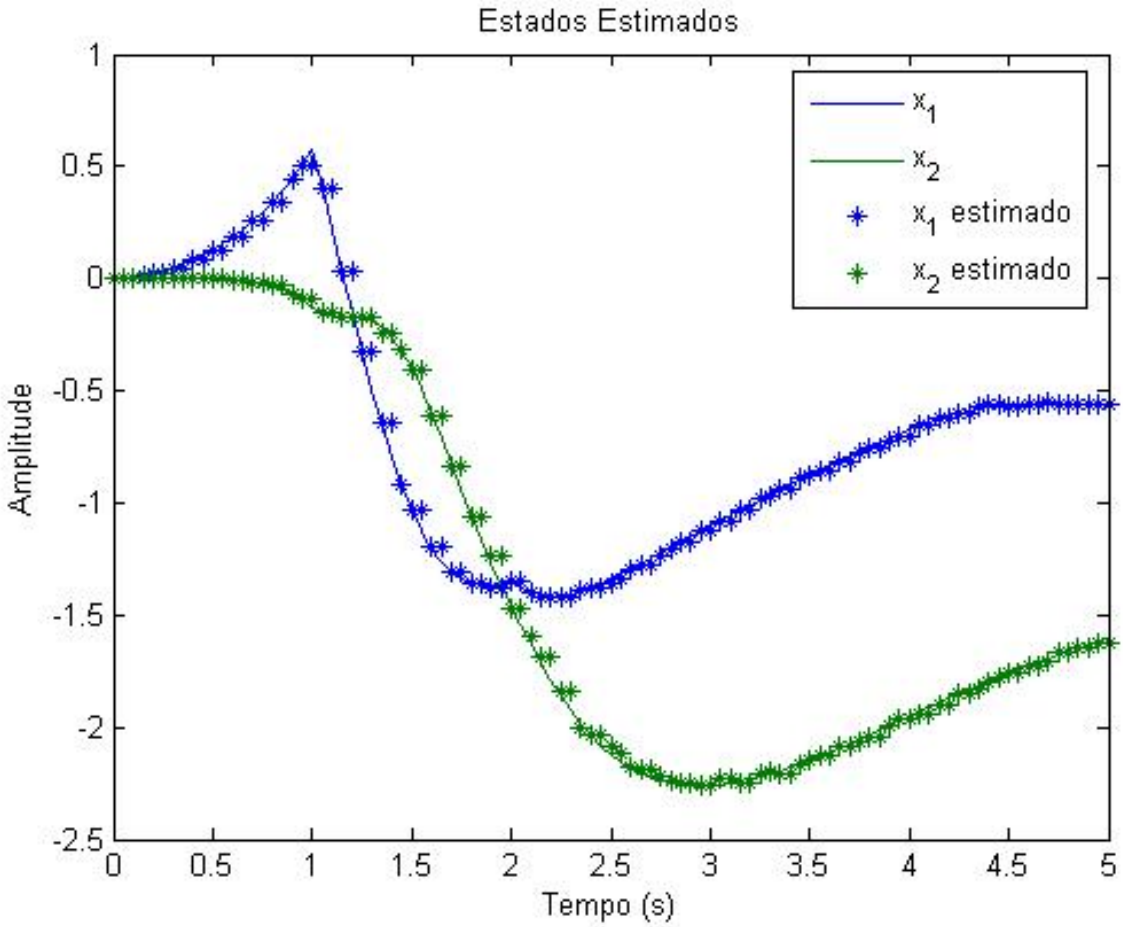

Figura 5.35: Estado estimado, configuração 2: OEP

Esse resultado é bastante superior ao anterior, mostrando que, analogamente ao que é obtido para o observador evolutivo, no observador OEP, o aumento do número de indivíduos (partículas) da população proporciona ao algoritmo um ganho em capacidade de exploração do espaço de estado. Esse resultado superior é fruto do aumento da densidade de partículas na região do espaço de estado onde encontra-se o estado do sistema, que aumenta a probabilidade de uma dessas partículas obter uma boa estimativa desse ponto.

Dando sequência ao processo de análise, é apresentada a configuração 3, na qual é aumentado o número de iterações utilizadas no processo iterativo de otimização com o mesmo objetivo: melhorar a qualidade das estimativas.

Configuração 3:

- $n_p = 10$
- $n_g = 100$
- $l = 1$
- $c_1 = c_2 = 1$

O resultado da aplicação desse observador é mostrado na Figura 5.36.

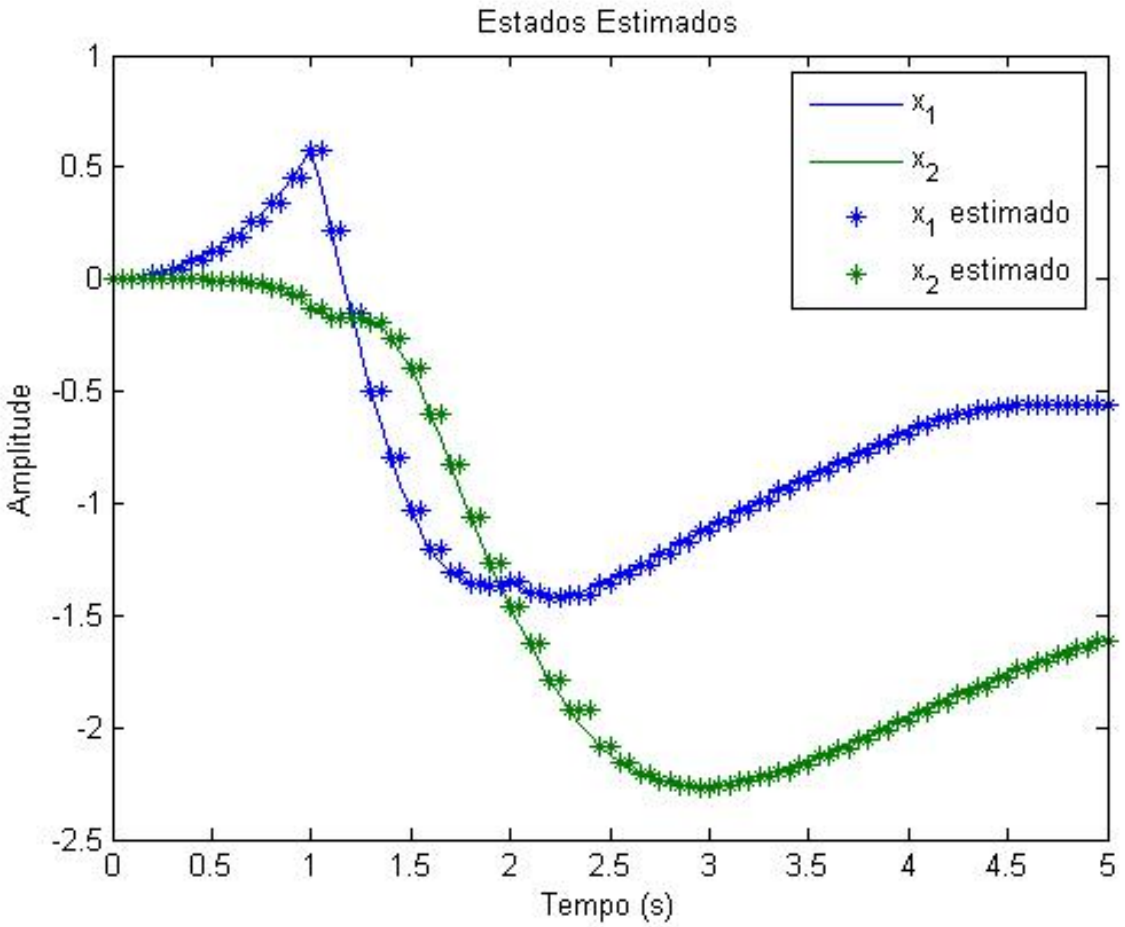

Figura 5.36: Estado estimado, configuração 3: OEP

Como era esperado, o aumento no número de iterações proporcionou um ganho significativo na qualidade das estimativas apresentadas, o que também acontece quando se aumenta o número de gerações do observador evolutivo.

Como última análise, consideremos a configuração 4, onde é modificado o comprimento l, visando prover uma condição inicial melhor ao algoritmo, na busca de cada nova estimativa.

Configuração 4:

- $n_p = 10$
- $n_g = 10$
- $l = 0.1$
- $c_1 = c_2 = 1$

O resultado da aplicação desse observador é mostrado na Figura 5.37.

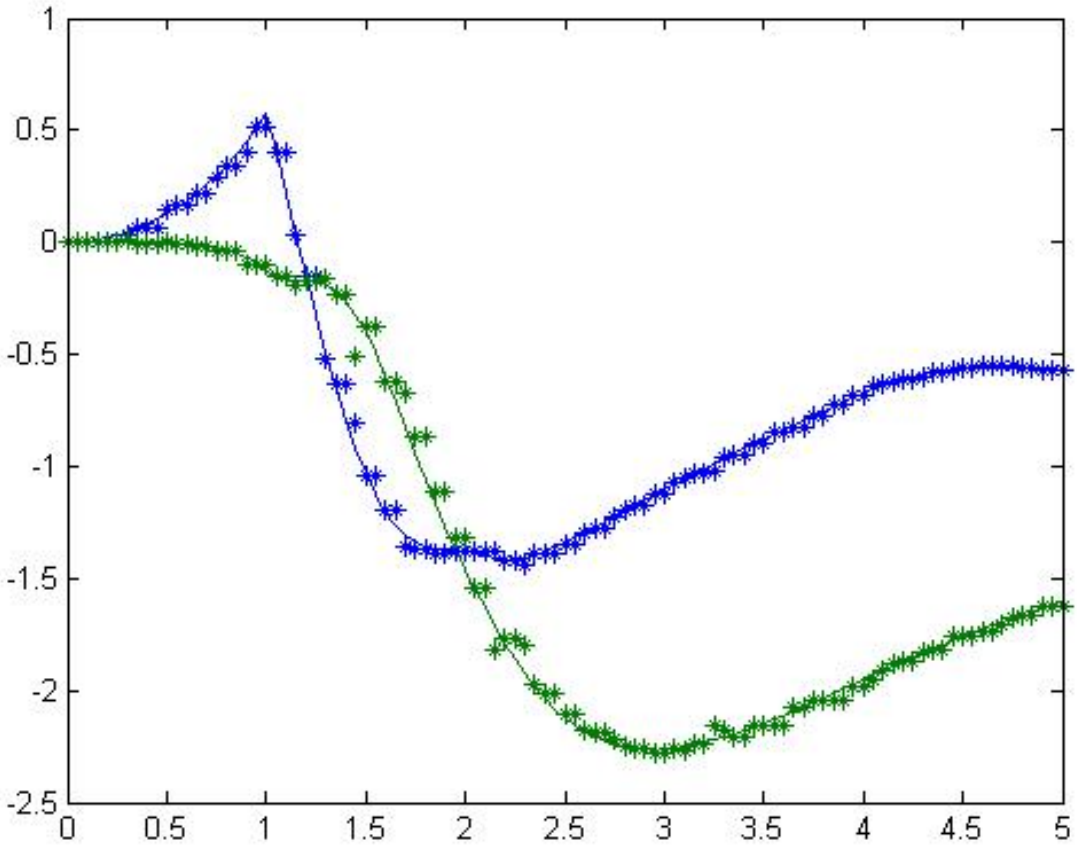

Figura 5.37: Estado estimado, configuração 4: OEP

O resultado obtido com a configuração 4 é novamente bastante previsível e semelhante àquele observado para o observador evolutivo, indicando claramente a grande semelhança existente nos resultados encontrados por essas duas técnicas. Considerando que o problema de se estimar o estado do sistema mostrado em (5.32) possui duas variáveis, os resultados aqui obtidos estão coerentes com as conclusões obtidas com relação a aplicação de algoritmos genéticos e otimização por enxame de partículas feita no Capítulo 3.

A seguir, é mostrado um gráfico na Figura 5.38, que ilustra os erros de estimação cometidos por cada uma das configurações analisadas para o observador OEP, plotados em uma escala logarítmica.

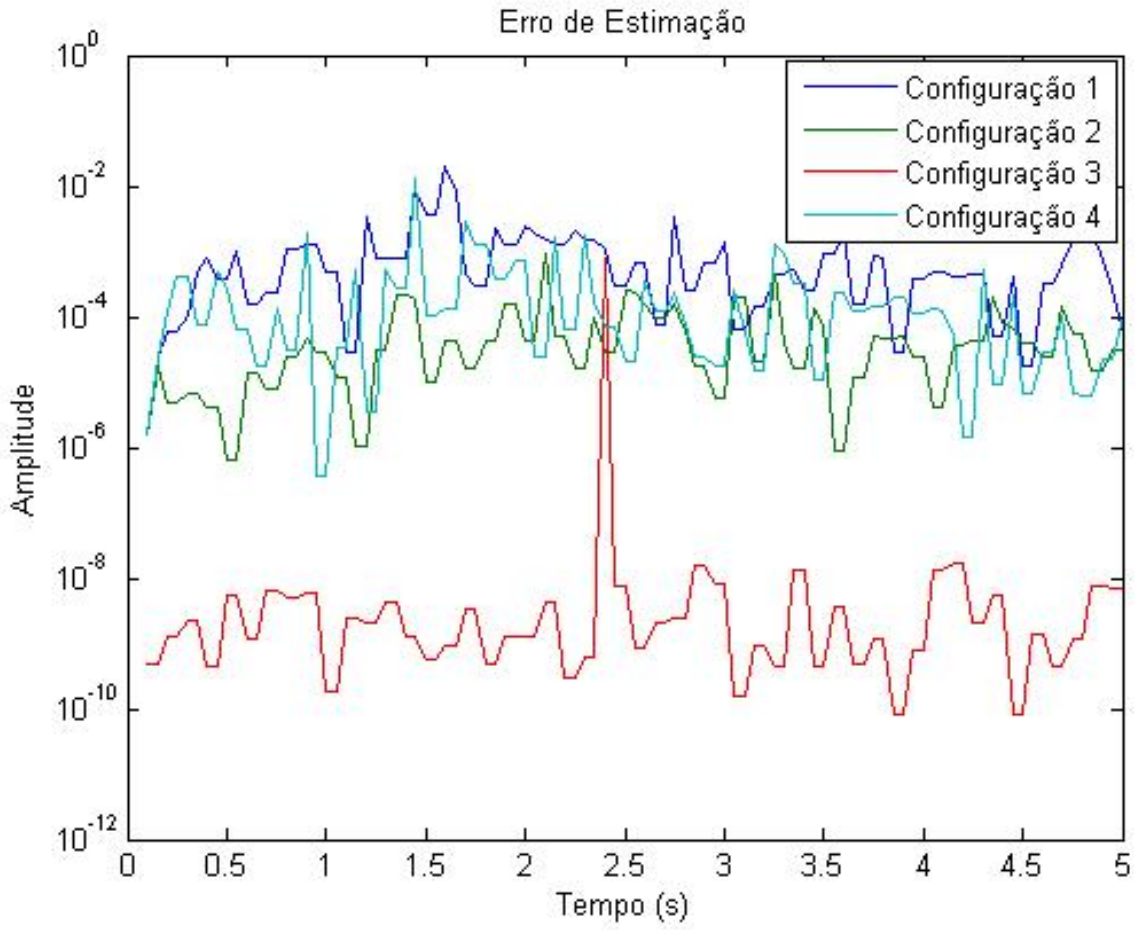

Figura 5.38: Erro de estimação para as quatro configurações analisadas: OEP

A Figura 5.38 indica que, para o observador OEP, assim como para o observador evolutivo, o aumento no número de iterações (ou número de gerações para o observador evolutivo) proporciona um ganho em eficiência considerável, aumentando a qualidade das estimativas, sendo esse ganho superior àquele observado quando é aumentado o tamanho da população n_p, ou diminuído o comprimento

l. Para o observador OEP, esse ganho de eficiência com o aumento do número de iterações é ainda maior do que aquele obtido para o observador evolutivo, sendo os resultados da estimação para a configuração 4 bastante superiores aos obtidos para as demais configurações.

Capítulo 6

Conclusão

O desenvolvimento dos observadores inteligentes propostos neste documento passa por uma fase inicial, onde são analisadas as ferramentas computacionais empregadas. Nesta etapa, são analisadas todas as ferramentas desde sua inspiração biológica até o estado da arte de cada uma delas. Esta análise proporcionou também a concepção das diversas propostas nas áreas de inteligência computacional consideradas neste estudo.

No Capítulo 2, foi apresentada uma análise detalhada de diversos operadores de reprodução encontrados na literatura. Com base em uma abordagem não convencional, são propostos 4 novos operadores. Vários testes realizados com os operadores propostos e aqueles encontrados na literatura demonstram claramente a superioridade em taxa de convergência alcançada, tendo sido obtidos resultados superiores com a utilização dos operadores propostos em todos os cenários de aplicação analisados. Esta superioridade é comprovada de maneira formal através de um processo de inferência estatística, onde pode-se inferir, com base em um intervalo de confiança, que em média, os operadores propostos levam a resultados melhores.

Posteriormente, é discutido o algoritmo de otimização por enxame de partículas, o qual é relativamente recente. Porém, já existem muitos trabalhos nessa área, que proporcionaram a criação de vários métodos para aumentar a eficiência do algoritmo original. Existe uma relação forte entre taxa de convergência e preservação da diversidade, tanto nos algoritmos genéticos quanto no algoritmo PSO. Métodos que tentam maximizar a taxa de convergência acabam, invariavelmente, reduzindo a diversidade da população. O método proposto no Capítulo 3, por outro lado, tem a capacidade de aumentar a taxa de convergência, sem degradar a capacidade exploratória do algoritmo, através de uma modulação de velocidade, que explora de forma racional a trajetória descrita pelas partículas.

Comparações realizadas demonstram a superioridade dos resultados obtidos também com a utilização dessa modulação proposta, quando comparada a outros métodos de melhoramento para algoritmo PSO encontrados na literatura, dentro do contexto aqui estudado. São realizadas também

várias comparações entre os algoritmos genéticos e o algoritmo PSO, cujos resultados servem como fundamento na análise do comportamento de cada um desses métodos. Com base nesses resultados e no conhecimento do problema a ser resolvido, pode-se então optar de forma coerente pela aplicação de um desses algoritmos.

Essas meta-heurísticas bio-inspiradas, AG e PSO, servem como base para a concepção de dois novos algoritmos de treinamento para redes MLP: *Gradiente Evolutivo* e *Gradiente das Partículas*, que através de exemplos práticos de aplicação, mostraram-se comparáveis em eficiência a algoritmos de treinamento procedurais. Esses algoritmos, além da eficiência, possuem características de interesse dos métodos de busca cega, utilizados na evolução do gradiente. Com isso é possível sua aplicação em uma vasta gama de problemas, dentre os quais pode-se citar o observador adaptativo neural proposto, que não pode ser treinado com base em algoritmos de treinamento procedurais, baseados no gradiente do erro.

Essas propostas geram um aperfeiçoamento maior das ferramentas de inteligência computacional utilizadas. Porém, o objetivo principal deste documento é a concepção de observadores de estado inteligentes, baseados nessas ferramentas e não em modelos dinâmicos. Esse objetivo é alcançado no Capítulo 5, onde com base em um sólida fundamentação teórica, relacionada aos subespaços gerados pelo estado, entrada e saída do sistema, são apresentadas 4 propostas de observadores inteligentes:

- Observador Neural
- Observador Adaptativo Neural
- Observador Evolutivo
- Observador Baseado em Inteligência de Enxame

O primeiro observador proposto, é sem dúvida o mais intuitivo por explorar diretamente o mapeamento $\mathcal{H}^{-1}$. Esse observador apresenta um erro de estimação relativamente pequeno e sua implementação é bastante simples, não sendo necessário um estudo detalhado da dinâmica do sistema, como é o caso na implementação de observadores clássicos. A escolha dos parâmetros do observador mostrou-se robusta, não implicando em um fator crítico no projeto do mesmo.

A partir desse observador, é desenvolvido o observador adaptativo neural, que preserva todas as caraterísticas de simplicidade e robustez do observador neural dentro do contexto estudado, porém, não requer uma fase de treinamento anterior à sua aplicação, o que representa uma expansão significativa da gama de problemas aos quais esse observador pode ser aplicado. Ambos os observadores, baseados na aproximação do mapeamento $\mathcal{H}$, apresentaram bons resultados para os exemplos de aplicação discutidos no Capítulo 5, confirmando a potencialidade desse paradigma.

É desenvolvida ainda uma nova abordagem, onde se consegue, com sucesso, transformar o problema de observação de estado em um problema de busca, com o intuito de aplicar os algoritmos AG e PSO diretamente na busca da estimativa do estado. Dessa abordagem surgiram as duas outras propostas: *Observador Evolutivo* e *Observador Baseado em Inteligência de Enxame*. Esses observadores, concebidos com base nesse processo de busca, demonstram nos exemplos discutidos, uma grande capacidade de estimação. A forma como foram descritos favorece ainda a sua aplicação em sistemas de controle, onde os mesmos podem ser utilizados para realimentar o estado para um controlador.

Os observadores propostos apresentam ainda outra grande vantagem, pois não possuem um tempo de acomodação para apresentar uma estimativa correta do estado, como acontece com os observadores baseados em modelos dinâmicos, podendo fornecer estimativas do estado, com um erro muito pequeno, a partir do início de sua operação, desde a primeira amostra obtida.

Referências Bibliográficas

[1] César Daltoé Berci and Celso Pascoli Bottura. Modulação de velocidade em otimização por enxame de partículas. *Proceedings of 7th Brazilian Conference on Dynamics, Control and Applications*, 7:241–248, 2008.

[2] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, London, 1859.

[3] André Luiz Brun. *Algoritimos Genéticos*. Apostila EPAC - Encontro Paranaense de Computação, 2007.

[4] J.H. Holland. Adaptation in natural and artificial systems. *University of Michigan Press*, 1975.

[5] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, 1989.

[6] Wirt Atmar. Notes on the simulation of evolution. *IEEE Trans. on Neural Networks*, 1:130–148, 1992.

[7] Fernando J. Von Zuben. Computação evolutiva - uma abordagem pragmática. *Anais da I Jornada de Estudos em Computação de Piracicaba e Região*, 2000.

[8] F. Herrera, M. Lozano, and J. L. Verdegay. Taking real-coded genetic algorithms: Operators and tools for behavior analysis. *Artificial Intelligence Review*, 12, 1998.

[9] L. N. de Castro. *Fundamentals of Natural Computing: Basic Concepts*. Chapman & Hall/CRC, 2006.

[10] N.J. Radcliffe. Equivalent class analysis of genetic algorithms. *Complex Systems*, 5:183–205, 1991.

[11] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, 1992.

[12] L.J. Eshelman and J.D. Schafter. Real-coded genetic algorithm and interval schemata. *Foundation of Genetic Algorithms 2*, pages 375–382, 1993.

[13] Alden H. Wright. Genetic algorithms for real parameter optimization. *Department of Computer Science University of Montana*, pages 205–220, 1991.

[14] Heinz. Mühlenbein and Dirk. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm i. continuous parameter optimization. *Evolutionary Computation*, 1:25–49, 1993.

[15] José Luiz Boldrini, Sueli Rodrigues Costa, Vera Lúcia Figueiredo, and Henry Wetzler. *Álgebra Linear*. Departamento de Matemática da Universidade Estadual de Campinas, Editora Harbra ltda, 1986.

[16] F. Heppner and U. Grenander. *A Stochastic Nonlinear Model For Coordinate Bird Flocks*. AAAS Publications, Washington, DC, 1990.

[17] C.W. Reynolds. Flocks, herds, and schools: A distributed behavior model. *Computer Graphics*, 21(4):25–34, 1987.

[18] Lawrence Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[19] H. M. Voigt and T. Anheyer. Modal mutation in evolutionary algorithms. *Proceedings of the IEEE Conference on Evolutionary Computation*, 1:88–92, 1994.

[20] Geraldo Beni and Jing Wang. Swarm intelligence. *Proceedings of 7th Annual. Meeting of the Robotics Society of Japan, RJS Press*, 7:425–428, 1989.

[21] Maja J Matarìc̀. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4:51–80, 1995.

[22] J. Kennedy and R.C Eberhart. Particle swarm optimization. *Proceedings of IEEE Int. Conf. on Neural Network*, Piscataway, NJ:1942–1948, 1995.

[23] F. den van Bergh. *An Analysis of Particle Swarm Optimization*. PhD thesis, Departament of Computer Science, University of Pretoria, 2002.

[24] R. C. Eberhart, P. Simpson, and R. Dobbins. *Computational Intelligence PC Tools*. Academic Press Professional, 1996.

[25] A. Ratnaweeera, S.K. Halgamuge, and H.C. Watson. Self-organized hierarchical particle swarm optimizer with time-varying acceleration coeficients. *IEEE Transactions on Evolutionary Computation*, 8(3):240–250, 2004.

[26] Li-ping Zhang, Huan-jun Yu, and Shang-xu Hu. Optimal choice of parameters for particle swarm optimization. *Journal of Zhejiang University*, 6A:528–534, 2005.

[27] D. Swagatam, A. Ajith, and K. Amit. *Particle Swarm Optimization and Diferential Evolution Algorithms: Techinical Analysis, Applications and Hybridization Perspectives*. Dept. of Eletronics and Telecommunication Engineering, Jadavpur University, Kolkata, India, 2007.

[28] Y Shi and C. Russ Eberhart. A modified particle swarm optimizer. *IEEE Int. Conf. on Evolutionary Computation*, 1998.

[29] M. Clerc. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. *IEEE Int. Conf. on Evolutionary Computation*, 2001.

[30] M. Clerc and J. Kennedy. The particle swarm-explosion stability, and convergence in a multidimensional complex space. *IEEE Int. Conf. on Evolutionary Computation*, 6(1):58–73, 2002.

[31] D. Corne, M. Dorigo, and F. Glover. *New Ideas In Optimazation*. McGraw Hill, 1999.

[32] C. Russ Eberhart and Y Shi. Comparing inertia weights and constriction factors in particle swarm optimization. *Proceedings of the Congress on Evolutionary Computing IEEE*, pages 84–89, 2000.

[33] César Daltoé Berci and Celso Pascoli Bottura. Speed modulation: A new approach to improve pso performance. *Artigo Aceito: IEEE Swarm Intelligence Symposium 2008, Sheraton Westport at St. Louis, Missouri*, 2008.

[34] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Neurocomputing: foundations of research, MIT Press*, pages 15–27, 1988.

[35] D.O. Hebb. *The Organization of Behavior A Neuropsychological Theory*. Wiley, 1949.

[36] Fran Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Cornell Aeronautical Laboratory, Psychological Review, v65*, 6:386–408, 1958.

[37] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

[38] John J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities*. Perseus Books, Cambridge, MA, USA, 1999.

[39] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning representations by back-propagating errors*. MIT Press, Cambridge, MA, USA, 1988.

[40] C. Fyfe. *Artificial Neural Networks*. Department of Computing and Information Systems, The University of Paisley, Edition 1.1, 1996.

[41] G. Cybenko. Approximations by superpositions of a sigmoidal funcion. *Math Control Signal Systems*, pages 303–314, 1989.

[42] J.M. Mendel and K.S. Fu. *Adaptive, learning, and pattern recognition systems: theory and applications*. Academic Press, New York, 1970.

[43] Antônio P. Braga, André C. P. L. F. Carvalho, and Teresa B. Ludermir. *Redes Neurais Artificiais*. LTC, 1ª edição, 2000.

[44] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation, in: Parallel distributed processing: Exploration in the microstructure of cognition. *Eds. D.E. Rumelhart, J.L. McClelland, MIT Press, Cambridge, MA.*, pages 318–362, 1986.

[46] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin. *Backpropagation: The basic theory*. Lawrence Erlbaum Associates, Inc., 1995.

[47] P J Werbos. *New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA., 1974.

[48] D.G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 2nd edition, 1984.

[49] R. Battiti and F. Masulli. Bfgs optimization for faster and automated supervised learning. *INCC 90 Paris, International Neural Network Conference,*, pages 757–760, 1990.

[50] M.F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. neural networks,. *Computer Science Department, University of Aarhus Denmark*, 6:525–533, 1990.

[51] Udo Seiffert. Multiple layer perceptron training using genetic algorithms. *ESANN'2001 proceedings - European Symposium on Artificial Neural Networks*, pages 159–164, 2001.

[52] D. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 762–767, 1989.

[53] P. G. Korning. Training neural networks by means of genetic algorithms working on very long chromosomes. *International Journal of Neural Systems*, 6(3):299–316, 1995.

[54] César Daltoé Berci and Celso Pascoli Bottura. The evolving gradient: An hybrid algorithm based on genetic algorithms and backpropagation for neural networks training. *Artigo Aceito: Fifth International Symposium on Neural Networks, Beijing, China.*, 2008.

[55] D.J. Chalmers. The evolution of learning: An experiment in genetic connectionism. *Proceedings of the 1990 Connectionist Summer School*, pages 81–90, 1990.

[56] César Daltoé Berci and Celso Pascoli Bottura. Observador inteligente adaptativo neural não baseado em modelo para sistemas não lineares. *Proceedings of 7th Brazilian Conference on Dynamics, Control and Applications*, 7:209–215, 2008.

[57] M.F. Møller. Learning by conjugate gradients. *The 6th International Meeting of Young Computer Scientists*, 1990.

[58] César Daltoé Berci and Celso Pascoli Bottura. Observador neural inteligente não baseado em modelo para sistemas não lineares. *Artigo Aceito: Congresso Brasileiro de Automática, Juiz de Fora, MG*, 2008.

[59] César Daltoé Berci and Celso Pascoli Bottura. Observador inteligente para sistema não lineares baseado em inteligência de enxame. *Artigo Aceito: Congresso Brasileiro de Automática, Juiz de Fora, MG*, 2008.

[60] Ron M. Hirschorn and Andrew D. Lewis. Geometric local controllability: Second-order conditions. *Transactions Of The American Mathematical Society*, 1997.

[61] Milena Anguelova. Nonlinear observability and identifiability: General theory and a case study of a kinetic model for s. cerevisiae. *Thesis for the degree of licentiate of engineering: Department of Mathematics, School of Mathematical Sciences, Chalmers University of Technology and the Göteborg University, Sweden*, 2004.

[62] Kristi A. Morgansen. Geometric methods for nonlinear control systems. *Notas de Aula, Aeronautics&Astronautics, California Institute of Technology*, 2004.

[63] G. Besançon. *Nonlinear Observers and Aplications*. Springer, 2007.

[64] Katsuhiko Ogata. *Engenharia de Controle Moderno*. Prentice Hall, 2003.

[65] J.T. Spooner, M. Maggiore, R. Ordóñez, and K. Passino. *Stable Adaptive Controll and Estimation for Nonlinear Systems*. Wiley Interscience, 2002.

[66] José A. R. Vargas and Elder M. Hemerly. Observador adaptativo estável usando redes neurais artificiais. *Proceedings of the IV Brazilian Conference on Neural Networks*, IV:090–095, 1999.

[67] Guang Leu Yih and Tian Lee Tsu. Adaptive fuzzy-neural observer for a class of nonlinear systems. *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, pages 2130–2135, 2000.

[68] Heinrich W. Guggenheimer. *Differential Geometry*. Dover Publications INC., New York, 1963.

[69] Henrique Fleming. *Geometria Diferencial*. 2002. Notas de Aula, Instituto de Física, Universidade de São Paulo.

[70] Andrew D. Lewis. Nonlinear control theory. *Transactions Of The American Mathematical Society*, 2002.

[71] Jairo Simon Fonseca and Gilberto A. Martins. *Curso de Estatística*. Editora Atlas, São Paulo, 6ª edição, 1996.

[72] Douglas C. Montgomery and George C. Runger. *Estatística Aplicada e Probabilidade para Engenheiros*. LTC, 2ª edição, 2003.

Apêndice A

Apêndice A - Geometria Diferencial

Este apêndice tem por objetivo fornecer algumas definições básicas de geometria diferencial e de notação utilizadas neste documento.

Primeiramente, consideremos a definição de espaço vetorial:

Definição 1 (Espaço Vetorial) *Um espaço vetorial $\mathcal{V}$, é um grupo abeliano, definido sobre um corpo K, tal que se observe [68]:*

$$\alpha X \in \mathcal{V}\,,\, \forall\, X \in \mathcal{V}\ e\ \forall\, \alpha \in K \tag{A.1}$$

e também:

$$X_1 + X_2 \in \mathcal{V}\,,\, \forall\, X_1, X_2 \in \mathcal{V} \tag{A.2}$$

Considerando esta definição, onde o corpo K representa o conjunto dos números reais R, é possível então definir um campo vetorial como segue:

Definição 2 (Campo Vetorial) *Um campo vetorial, $V : R^n \rightarrow R^n$, é uma função que associa um vetor pertencente ao espaço R^n, a cada ponto x de um espaço vetorial R^n, como é mostrado na equação (A.3) [69, 70]:*

$$V(x) = \begin{bmatrix} V_1(x) \\ V_2(x) \\ \vdots \\ V_n(x) \end{bmatrix} \tag{A.3}$$

Nesta equação cada componente $V_i(x)$ do campo vetorial V representa um mapeamento.

A formalização do conceito de mapeamento é dada na Definição 3.

Definição 3 *Um mapeamento $H : R^n \rightarrow R^m$ é uma função que associa a cada ponto do espaço vetorial R^n, um ponto pertencente ao espaço vetorial R^m, "mapeando" esses dois espaços por meio de uma transformação [69, 70].*

$$H(x) = \begin{bmatrix} H_1(x) \\ H_2(x) \\ \vdots \\ H_m(x) \end{bmatrix} \tag{A.4}$$

Os conceitos de campo vetorial e mapeamento são de grande relevância na área de controle, tendo em vista que um sistema dinâmico pode ser genericamente descrito por meio dessas duas entidades da seguinte forma:

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned} \tag{A.5}$$

Sendo que, para um entrada u constante, tem-se que, $f : R^n \rightarrow R^n$ é um campo vetorial e $h : R^n \rightarrow R^p$ é um mapeamento.

Outra ferramenta de grande relevância na analise de sistemas dinâmicos é a derivada de Lie.

Definição 4 (Derivada de Lie) *Dada uma função escalar $h(x)$, tal que: $h : R^n \rightarrow R$, e um campo vetorial $f(x) \mid f : R^n \rightarrow R^n$, a **Derivada de Lie**, de h com relação a f, será um escalar definido por:*

$$L_f h(x) = \frac{\partial h(x)}{\partial x} f(x) = \nabla h(x) f(x) \tag{A.6}$$

A derivada de Lie, pode ser interpretada como sendo a derivada do mapeamento $h(x)$, na direção do campo vetorial $f(x)$.

Consideremos agora dois espaços vetoriais $\mathcal{X} \in R^n$ e $\mathcal{Y} \in R^m$, definidos sobre um mesmo corpo K, onde $K = R$ e um mapeamento $H : \mathcal{X} \rightarrow \mathcal{Y}$ tal que:

$$H(x) = y, \quad x \in \mathcal{X},\ e\ \ y \in \mathcal{Y} \tag{A.7}$$

Supondo agora que seja possível expressar o mapeamento inverso $H^{-1} : \mathcal{Y} \rightarrow \mathcal{X}$ da seguinte forma:

$$H^{-1}(y) = x, \quad x \in \mathcal{X},\ e\ \ y \in \mathcal{Y} \tag{A.8}$$

Então o mapeamento H é considerado um difeomorfismo entre os espaços vetoriais $\mathcal{X}$ e $\mathcal{Y}$. Neste caso se observa a seguinte igualdade [63]:

$$posto(dH) = n \tag{A.9}$$

Onde dH é uma matriz formada pelos vetores gradiente do mapeamento H:

$$dH(x) = \begin{bmatrix} \nabla H_1(x) \\ \nabla H_2(x) \\ \vdots \\ \nabla H_m(x) \end{bmatrix} \tag{A.10}$$

Apêndice B

Apêndice B - Estatística

Dada a característica estocástica da maioria dos métodos de inteligência computacional estudados neste documento, faz-se necessária a introdução de alguns conceitos básicos de estatística, abordados neste apêndice.

Um dos principais conceitos de estatística é a definição de variável aleatória dada a seguir:

Definição 5 (Variável Aleatória) *Seja E um experimento não determinístico qualquer, com um espaço S que descreve os resultados desse experimento. Uma variável aleatória é uma função x, que associa cada elemento do espaço S a um número real [71, 72]:*

$$x(s) = \alpha \tag{B.1}$$

onde $s \in S$ e $\alpha \in R$.

Desta forma, uma variável aleatória pode ser entendida como o resultado numérico de um experimento não determinístico.

Para uma variável aleatória podem ser definidas algumas propriedades, como média e variância.

Definição 6 (Media ou Esperança) *A esperança de uma variável aleatória x é definida com base na seguinte equação [71, 72]:*

$$E(x) = \mu_x = \int_{-\infty}^{\infty} x f(x) dx \tag{B.2}$$

onde $f(x)$ representa a função densidade de probabilidade da variável aleatória x.

Consideremos agora, uma coleção de amostras, extraídas aleatoriamente de um população X. A média amostral dessa coleção é definida por:

$$E(X) = \mu_X = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{B.3}$$

onde x_i representa a *i-ésima* amostra de uma coleção contendo n amostras.

A variância de uma variável aleatória é definida como segue:

Definição 7 *Dada um variável aleatória x, com função densidade de probabilidade $f(x)$, a variância de x é definida pela equação (B.4) [71, 72]:*

$$var(x) = \sigma_x^2 = E[(x - \mu_x)^2] \tag{B.4}$$

Novamente, para o caso onde é extraída uma coleção de amostras de uma população, a variância amostral pode ser definida como segue, sendo uma boa estimativa da variância populacional.

$$var(X) = \sigma_X^2 = \sum_{i=1}^{n} E[(x_i - \mu_X)^2] \tag{B.5}$$

Consideremos agora dois conjuntos amostrais X e Y, extraídos de duas populações diferentes, cujas médias e variâncias são desconhecidas. É possível calcular a média e a variância de cada um dos conjuntos amostrais:

$$\begin{aligned}
\mu_X &= \frac{1}{n_x}\sum_{i=1}^{n_x} x_i \\
\sigma_X^2 &= \sum_{i=1}^{n_x} E[(x_i - \mu_X)^2] \\
\mu_Y &= \frac{1}{n_y}\sum_{i=1}^{n_y} y_i \\
\sigma_Y^2 &= \sum_{i=1}^{n_y} E[(x_i - \mu_X)^2]
\end{aligned}$$

Porém, não é possível definir as médias populacionais das variáveis aleatórias x e y, com base nas medias amostrais.

Supondo que seja necessário decidir qual das duas populações x e y tem maior valor médio. Neste caso, é possível construir intervalos de confiança a partir dos dados amostrais e com base nesses intervalos, pode-se inferir a respeito dos parâmetros populacionais desconhecidos. Para tanto, consideremos o teste de hipóteses para duas médias amostrais, onde são definidas duas hipóteses:

$$H_0 : \quad \mu_x = \mu_y$$

$$H_1 : \qquad \mu_x \neq \mu_y$$

onde H_0 é a hipótese nula, que é a hipótese estatística a ser testada e H_1 é a hipótese alternativa.

Dois tipos de erros podem ser cometidos ao se aceitar alguma dessas hipóteses, sendo eles:

- **Erro Tipo I**: Rejeitar uma hipótese verdadeira.
- **Erro Tipo II**: Aceitar uma hipótese falsa.

Assim será cometido um erro tipo I, sempre que a hipótese nula for rejeitada quando essa é verdadeira, ou seja, é o erro cometido em se assumir que existe uma diferença nas médias populacionais, quando elas de fato são iguais.

O cálculo da estatística de teste que define a probabilidade de ocorrência do erro tipo I para populações normalmente distribuídas, é realizado com base na seguinte equação:

$$Z = \frac{\mu_X - \mu_Y}{\sqrt{\frac{\sigma_X^2}{n_x} + \frac{\sigma_Y^2}{n_y}}} \tag{B.6}$$

A probabilidade de ocorrência do erro tipo I, pode então ser obtida em uma tabela de distribuição normal.