Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Engenharia Elétrica

Trabalho de Conclusão de Curso

Técnicas Meta-heurísticas para Seleção de Polo Ótimo das Bases de Funções Ortonormais Aplicadas em Séries de Volterra

Alisson Marden Fonseca Pereira

João Monlevade, MG
2017

Alisson Marden Fonseca Pereira

Técnicas Meta-heurísticas para Seleção de Polo Ótimo das Bases de Funções Ortonormais Aplicadas em Séries de Volterra

Trabalho de Conclusão de curso apresentado à Universidade Federal de Ouro Preto como parte dos requisitos para obtenção do Título de Bacharel em Engenharia Elétrica pelo Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto.
Orientador: Prof. Dr. Márcio Feliciano Braga
Coorientador: Prof. Dr. Víctor Campos da Silva Costa

Universidade Federal de Ouro Preto
João Monlevade
2017

P436t Pereira, Alisson Marden Fonseca.
Técnicas meta-heurísticas para seleção de polo ótimo das bases de funções ortonormais aplicadas em séries de Volterra [manuscrito] / Alisson Marden Fonseca Pereira. - 2017.

51f.: il.: color; grafs; tabs.

Orientador: Prof. Dr. Márcio Feliciano Braga.
Coorientador: Prof. Dr. Víctor Campos da Silva Costa.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Engenharia Elétrica.

1. Engenharia Elétrica. 2. Identificação de sistemas . 3. Volterra, Series de. 4. Heurística. 5. Otimização . I. Braga, Márcio Feliciano. II. Costa, Víctor Campos da Silva. III. Universidade Federal de Ouro Preto. IV. Titulo.

CDU: 621.3

Catalogação: ficha@sisbin.ufop.br

MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Aplicadas
Colegiado do Curso de Engenharia de Elétrica

UFOP

ATA DE DEFESA

Aos 5 dias do mês de setembro de 2017, às 15 horas, no bloco B deste instituto, foi realizada a defesa de monografia pelo formando Alisson Marden Fonseca Pereira, sendo a comissão examinadora constituída pelos professores: Márcio Feliciano Braga, Víctor Costa da Silva Campos, Anny Verly e Rodrigo Augusto Ricco.

O candidato apresentou a monografia intitulada : Técnicas Meta-heurísticas para Seleção de Polo Ótimo das Bases de Funções Ortonormais Aplicadas em Séries de Volterra. A comissão examinadora deliberou, por unanimidade, pela APROVAÇÃO do candidato, com a nota média 10,0, de acordo com a Tabela 1. Na forma regulamentar foi lavrada a presente ata que é assinada pelos membros da comissão examinadora e pelo formando.

Tabela 1 – Notas de avaliação da banca examinadora

Banca Examinadora	Nota
Márcio Feliciano Braga	10,0
Víctor Costa da Silva Campos	10,0
Anny Verly	10,0
Rodrigo Augusto Ricco	10,0
Média	10,0

João Monlevade, 05 de setembro de 2017.

Márcio Feliciano Braga
Professor Orientador

Víctor Costa da Silva Campos
Professor Coorientador

Alisson Marden Fonseca Pereira
Aluno

Anny Verly
Professora Convidada

Rodrigo Augusto Ricco
Professor Convidado

MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Aplicadas
Colegiado do Curso de Engenharia de Elétrica

UFOP

ANEXO X - TERMO DE RESPONSABILIDADE

O texto do trabalho de conclusão de curso intitulado "Técnicas Meta-heurísticas para Seleção de Polo Ótimo das Bases de Funções Ortonormais Aplicadas em Séries de Volterra" é de minha inteira responsabilidade. Declaro que não há utilização indevida de texto, material fotográfico ou qualquer outro material pertencente a terceiros sem a devida citação ou consentimento dos referidos autores.

João Monlevade, 08 de Novembro de 2017.

Alisson Marden Fonseca Pereira
Nome completo do(a) aluno(a)

Agradecimentos

Agradeço primeiramente à minha querida mãe Natália e o meu pai Jaques por todo o suporte e ensinamentos passados durante toda a minha vida. Ao meu pai pelo incentivo e a confiança em mim depositada na busca pelo sonhado diploma. À minha mãe, por toda a ternura e carinho e por me encorajar até mesmo nas situações em que ela é quem estava mais insegura.

À minha irmã Hellen, por todos os sábios conselhos dados. À Anna Júlia, a irmã caçula que em 2009 veio ao mundo para abrilhantar e encher a família de alegria. A todos da minha grande família (primos, tios, avós), também responsáveis pela formação do meu caráter.

Aos meus orientadores Márcio e Víctor, por não medirem esforços para me auxiliar na conclusão deste trabalho (e de muitos outros). Ambos desempenharam não somente o papel de orientadores, mas também de amigos. São, portanto, referências para mim.

Agradeço aos amigos que fiz durante a graduação. Primeiramente, aqueles que conheci nas pensões Maanaim e, posteriormente, Colméia. Apesar das dificuldades, passamos por momentos épicos. Não posso deixar de agradecer aos bravos guerreiros, carinhosamente conhecidos como dinossauros, do Bonde da Elétrica (BDE), com eles foi muito mais fácil enfrentar os desafios da graduação.

“Let the future tell the truth, and evaluate each one according to his work and accomplishments. The present is theirs; the future, for which I have really worked, is mine”

– Nikola Tesla

Resumo

Este trabalho tem como foco a identificação de sistemas não lineares usando as séries de Volterra em conjunto com as Funções de Base Ortonormal (FBO) para representação dos modelos. Os modelos de Volterra são parametrizados por funções multidimensionais, os *kernels*. Tais modelos não possuem realimentação na saída, o que é uma grande vantagem, uma vez que outros tipos de representação realimentam o erro de saída do sistema. O problema de não haver realimentação do sinal de saída é que a quantidade de parâmetros a serem estimados costuma ser grande. Para contornar essa situação, descreve-se cada *kernel* por meio de uma expansão em FBO. O modelo resultante é conhecido como FBO-Volterra. O objetivo é determinar os polos que parametrizam as FBO, de modo a reduzir o número de termos a utilizar em cada base. Para tanto, são expostos métodos de otimização meta-heurísticos responsáveis pela seleção adequada desses polos e subsequente determinação dos parâmetros do modelo. São utilizados os conhecidos algoritmos meta-heurísticos *Simulated Annealing*, Algoritmo Genético (mono e multiobjetivo) e Otimização por enxame de Partículas (PSO) na identificação de um sistema de aquecimento com dissipação variável. Um estudo comparativo do desempenho das diferentes técnicas meta-heurísticas aplicadas ao problema de identificação de sistemas é realizado, em que é possível constatar o melhor desempenho do algoritmo genético multiojetivo para o problema proposto.

Palavras-chave: Identificação de Sistemas, Sistemas não lineares, Modelos de Volterra, Funções de base ortonormais, Otimização, Meta-heurísticas, *Simulated Annealing*, Algoritmo Genético, Otimização por Enxame de Partículas, Otimização Multiobjetivo.

Abstract

This work focuses on the nonlinear systems identification using Volterra series jointly with Orthonormal Basis Functions (OBF) to represent the models. The Volterra models are parameterized by multidimensional functions, the kernels. Such models do not feedback the output, which is an important advantage, since other classes of representation feedback the system output error. The problem of this open-loop representation is that the number of parameters to be estimated is usually large. To overcome this situation, each kernel can be described by an expansion in OBF. The resulting model is known as OBF-Volterra model. The aim is to determine the poles that parameterize the OBF, so that the number os terms in each base can be reduced. Therefore, metaheuristics optimization methods are used to properly select the poles and determine the parameters of the model. The well-known metaheuristics Simulated Annealing, Genetic Algorithm (mono and multiobjective) and Particle Swarm Optimization (PSO) are applied on the identification of a heating system with variable dissipation. A comparative performance evaluation for the different metaheuristic techniques, applied to the system identification problem, is performed. In this comparison, it is possible to note that the multiojective genetic algorithm has better performance while solving the proposed problem.

Keywords: System Identification, Nonlinear Systems, Volterra Models, Orthonormal Basis Function, Optimization, Metaheuristics, Simulated Annealing, Genetic Algorithm, Particle Swarm Optimization, Multiobjective Optimization.

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

TS	Takagi-Sugeno
FBO	Funções de Base Ortonormal
GOBF	*Generalized Ortonormal Basis Functions*
PSO	*Particle Swarm Optimization*
GA	*Genetic Algorithm*
NSGA-II	*Non-dominated Sorting Genetic Algorithm II*
SISO	*Single-Input-Single-Output*
ARMAX	*Autoregressive–moving-average with exogenous inputs model*
NARMAX	*Nonlinear autoregressive moving average with exogenous inputs model*
NARX	*Nonlinear autoregressive exogenous model*
FIR	*Finite Impulse Response*
PCV	Problema do Caixeiro Viajante
VND	*Variable Neighborhood Descent*
SI	*Swarm Intelligence*
MOEA	*Multiobjective Evolutionary Algorithms*
RMS	*Root Mean Square*
EQM	Erro Quadrático Médio
p.u.	por unidade
MQ	Mínimos Quadrados

Lista de símbolos

$\forall$	Para todo
ϕ	Letra grega minúscula phi
Φ	Letra grega maiúscula phi
ε	Letra grega minúscula epsilon
τ	Letra grega minúscula tau
$\Re$	Conjunto de números reais
Ψ	Letra grega maiúscula psi
Δ	Letra grega maiúscula delta
α	Letra grega minúscula alfa
ω	Letra grega minúscula omega
ζ	Letra grega minúscula zeta
$\in$	Pertence

Conteúdo

1 Introdução

O homem tem, desde os seus primórdios, a capacidade de observar situações e guardar informações que podem ser úteis futuramente. As experiências vividas ao longo do tempo ajudam a prever certas situações e, por conseguinte, preparar-se para elas. Assim, corriqueiramente suas atitudes são baseadas em *modelos*. Os quais podem ser entendidos como protótipos ou exemplos que se deseja imitar ou reproduzir. Dos diversos tipos de modelos existentes (mentais, físicos, econômicos, etc.), enfoca-se neste trabalho os modelos matemáticos. Como indicado em Aguirre (2007), a modelagem matemática é a área do conhecimento que estuda maneiras de desenvolver e implementar modelos matemáticos de sistemas reais.

Na engenharia de controle, a modelagem matemática pode ser realizada de três maneiras distintas. Uma delas é a modelagem caixa branca, em que a física do processo é suficientemente conhecida e é possível determinar a dinâmica do sistema. Um exemplo de modelagem caixa branca pode ser o sistema massa-mola, em que é possível determinar o deslocamento (saída) de um bloco/massa simplesmente conhecendo-se a força (entrada) aplicada ao sistema. Isso só é possível porque a dinâmica do processo é simples e conhecida. A modelagem caixa preta é utilizada quando a complexidade do sistema aumenta e não é possível estimar a saída simplesmente tendo conhecimento da entrada. Neste caso, o estudo é conduzido com o intuito de determinar o sistema tendo-se conhecimento das entradas e das saídas. Esse tipo de modelagem é conhecido como empírico, pois necessita de experimentos e medições para sua realização. Há ainda a técnica de modelagem caixa cinza (GARCIA, 2005), caracterizada por combinar as duas técnicas citadas anteriormente. A modelagem caixa cinza é comumente utilizada quando os dados processados são insuficientes, o que leva à necessidade de informações auxiliares (pertinentes à dinâmica ou estática do processo) (AGUIRRE, 2007).

Ao estudo das técnicas de modelagem matemática alternativas à modelagem caixa branca, dá-se o nome de Identificação de Sistemas (AGUIRRE, 2007). A identificação de sistemas possui um amplo campo de aplicações, tais como previsão de séries temporais (previsão do tempo, previsão de mercado, previsão de demanda energética, etc.), estudo do comportamento de sistemas, e controle. No último caso, o modelo identificado pode ser utilizado para o desenvolvimento de um sistema de controle ou até mesmo em um controlador (WESTWICK, 2003).

Dependendo do sistema a ser identificado, diferentes técnicas podem ser utilizadas. Na área de sistemas não lineares, várias abordagens são encontradas na literatura. Em Braga et al. (2011) e Campello, Oliveira e Amaral (2007), é apresentada a representação por meio de séries de Volterra. Também em Campello, Oliveira e Amaral (2007), as representações *fuzzy*-TS e OBF-TS são discutidas. Outra representação é a Neural, discutida em Nelles (2002).

Neste trabalho será utilizada a representação por modelos de Volterra. Esses são modelos polinomiais sem realimentação do sinal de saída (BRAGA, 2011). Tal representação é muito útil na identificação de sistemas devido ao fato de que é possível estender conceitos definidos

somente para problemas lineares a não lineares (ROSA; AMARAL; CAMPELLO, 2006).

Os modelos de Volterra são parametrizados por funções multidimensionais, os *kernels*. Por não possuírem realimentação do sinal de saída, os modelos de Volterra podem demandar um número elevado de termos para representar os *kernels* de Volterra. Esse empecilho pode ser resolvido ao descrever cada *kernel* por meio de uma expansão em uma base de funções ortonormais (FBO), gerando a representação conhecida como FBO-Volterra. Se a base for bem projetada, menor será o número de termos necessários para a representação FBO-Volterra do sistema (BRAGA, 2011). Por outro lado, a não realimentação da saída é uma vantagem, uma vez que a estabilidade do sistema é garantida (NELLES, 2002).

Projetar um modelo FBO-Volterra consiste basicamente em escolher a base de funções ortonormais a ser utilizada na série de Volterra e determinar os polos que caracterizam essa base. As bases discretas mais utilizadas na representação de sistemas lineares e não lineares são as bases de Laguerre, de Kautz e as bases de funções generalizadas (GOBF - *Generalized Orthonormal Basis Functions*) (BRAGA, 2011). As bases de Laguerre utilizam apenas polos reais na sua estrutura, enquanto as bases de Kautz utilizam polos complexos e as GOBF são capazes de inserir tanto polos reais quanto complexos em sua estrutura. Neste trabalho, serão consideradas apenas as bases de Laguerre por uma questão de redução de complexidade de resolução do problema (ROSA; AMARAL; CAMPELLO, 2006).

O objetivo do presente trabalho é obter os polos ótimos da base de Laguerre por meio da utilização de ferramentas de otimização computacional e comparar o desempenho das diferentes técnicas implementadas. Para tanto, este trabalho fará uso de meta-heurísticas, que são procedimentos destinados a encontrar uma solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual deve ser modelada de acordo com o problema em questão (SOUZA, 2011). As meta-heurísticas podem ser baseadas em busca local ou populacional.

As abordagens baseadas em busca local utilizam de movimentos que, a cada passo, são aplicados de modo a alterar (ou não) a solução atual a partir de soluções vizinhas. A meta-heurística *Simulated Annealing* é um exemplo de método que faz uso de tal de mecanismo. Quando a seleção ótima é baseada em buscas populacionais, conjuntos de soluções são gerados e esses são combinados de modo a tentar obter soluções melhores. Exemplos desse tipo de busca são os algoritmos genéticos e a otimização por enxame de partículas (PSO - *Particle Swarm Optimization*) (SOUZA, 2011; BONYADI; MICHALEWICZ, 2016). As três meta-heurísticas citadas, serão as utilizadas na seleção ótima dos polos da base Laguerre.

Além disso, outro interesse ao obter representações FBO-Volterra de um sistema é que essa seja a mais simplificada possível. Ou seja, deseja-se obter uma representação em que seja necessário estimar um menor número de parâmetros sem que haja redução da precisão. Para isso, será utilizada a otimização multiobjetivo. Problemas dessa natureza envolvem a minimização simultânea de um conjunto de objetivos, em que são satisfeitas uma série de restrições (PANTUZA, 2011).

Ao final da revisão bibliográfica necessária, será realizada a identificação de um sistema aquecedor com dissipação variável aplicando-se as estratégias apresentadas no trabalho. A massa de dados referente a esse experimento é disponibilizada em Aguirre (2007). A escolha desse problema foi feita devido ao fato de que sistemas desse tipo podem apresentar um comportamento altamente não linear (OREA et al., 1994), característica desejável para o presente trabalho.

1.1 Organização do trabalho

O primeiro capítulo deste trabalho foi iniciado apresentando-se uma rápida contextualização do tema de pesquisa. Essa contextualização incluiu os seguintes tópicos: introdução, motivação para realização do trabalho e objetivos do trabalho.

A sequência do trabalho está estruturada da seguinte forma:

- **Capítulo 2 - Técnicas de Identificação de Sistemas**: No Capítulo 2 serão descritos de forma sucinta os principais passos para resolução de um problema de identificação de sistemas, bem como a aplicação dos modelos de Volterra e a consequente necessidade da representação de tais modelos em funções de base ortonormal. A base de funções ortonormais de Laguerre será descrita e suas vantagens e desvantagens em relação às outras bases encontradas na literatura também serão discutidas.

- **Capítulo 3 - Otimização: Algoritmos Meta-heurísticos**: Fará uma breve contextualização do estudo da inteligência computacional para otimização, definem-se os algoritmos meta-heurísticos, discutem-se suas principais vantagens e, por fim, realiza-se um estudo comparativo entre algumas das diferentes técnicas meta-heurísticas encontradas na literatura.

- **Capítulo 4 - Otimização Multiobjetivo**: Apresentará a otimização multiobjetivo e o algoritmo genético multiobjetivo, no caso, o NSGA-II (*Non-dominated Sorting Genetic Algorithm II*).

- **Capítulo 5 - Resultados e Discussão**: No capítulo 5, serão apresentados os resultados obtidos por meio da aplicação das diferentes técnicas meta-heurísticas na identificação do sistema do aquecedor com dissipação variável. Sendo a representação do sistema feita pelo modelo FBO-Volterra com funções de Laguerre.

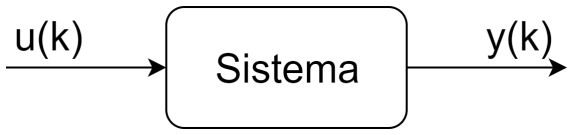
- **Capítulo 6 - Conclusão**: Por fim, serão feitas as devidas considerações finais, bem como a sugestão de trabalhos futuros.

2 Técnicas de Identificação de Sistemas

2.1 Introdução

A identificação experimental, ou modelagem empírica, faz o uso de informações obtidas por meio do processo de funcionamento do sistema. Na Figura 1, o sinal discreto *u(k)* representa a entrada de um sistema *Single-Input-Single-Output* (SISO), ao passo que *y(k)* representa a saída. Com esses dados disponíveis, é possível obter um modelo paramétrico do sistema, sem a necessidade de conhecimento prévio do sistema em análise (AGUIRRE, 2007).

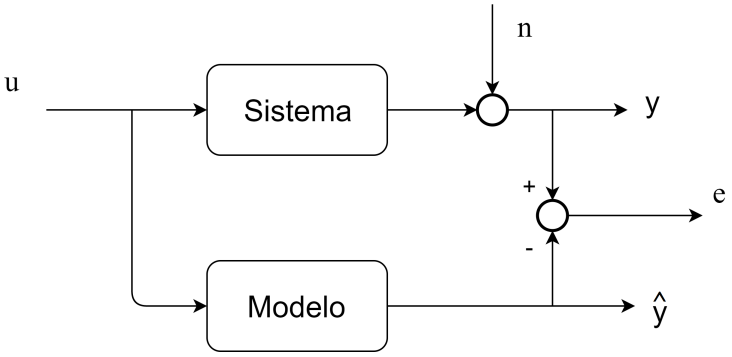
Figura 1 – Representação de um sistema SISO.



Fonte: Do autor.

De maneira geral, o objetivo da identificação de sistemas é ilustrado na Figura 2. Um mesmo sinal u alimenta o sistema original e o modelo obtido na identificação. O sinal y representa a saída perturbada do sistema e $\hat{y}$ a saída do modelo estimado. O erro, e, é a diferença entre esses dois sinais, isto é, $e = y - \hat{y}$. Dessa forma, quanto menor for o erro médio, melhor representado é o sistema. Essa representação matemática é obtida por meio de técnicas de identificação. Os principais passos para resolução deste tipo de problema são apresentados na Figura 3 (AGUIRRE, 2007; NELLES, 2002), que resumidamente são:

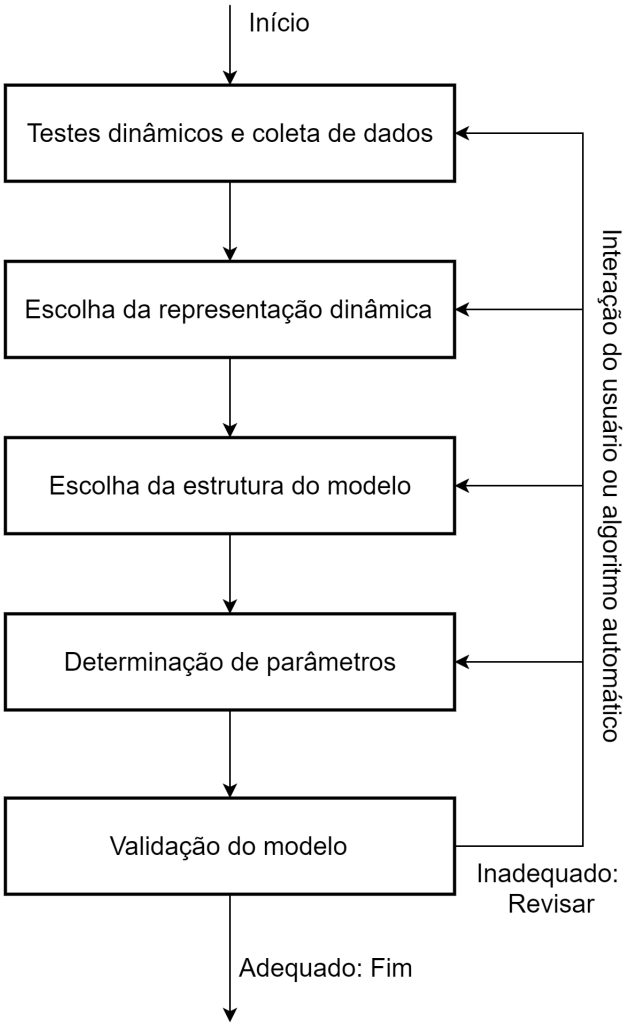
Figura 2 – Objetivo da identificação de sistemas: obter um modelo que leve à redução do erro médio e.



Fonte: Do autor.

- *Testes dinâmicos e coleta de dados:* A escolha do sinal de excitação é feita com o intuito de se inferir informações da dinâmica do processo. Na maioria dos casos práticos isso não

Figura 3 – Etapas para resolução de um problema de identificação.



Fonte: Do autor.

é possível e os únicos dados disponíveis serão os dados de "operação normal"ou dados históricos. Segundo Aguirre (2007), os problemas relacionados a esta etapa são a escolha dos sinais de excitação, a execução do teste e a escolha da taxa de amostragem.

- *Escolha da representação dinâmica:* A escolha é feita de acordo com o tipo de identificação utilizado. Funções de transferência em tempo contínuo são utilizadas em problemas simples e métodos de identificação determinística podem ser aplicados. Representações ARMAX são frequentemente utilizadas em identificação estocástica. Quando deseja-se representar sistemas discretos não lineares a representação por série de Volterra é uma opção (AGUIRRE, 2007). Mais detalhes sobre este tipo de representação são discutidos na Seção 2.2.

- *Escolha da estrutura do modelo:* Determinar a estrutura de um modelo linear consiste basicamente em escolher o número de polos e de zeros. Quando o modelo é não linear,

critérios sistemáticos podem ser empregados, alguns destes são apresentadas em Aguirre (2007).

- *Determinação de parâmetros:* É o passo mais conhecido na identificação de sistemas. O método de mínimos quadrados aplicado conjuntamente com algoritmos de otimização costumam ser utilizados (NELLES, 2002).

- *Validação do modelo:* Define se os passos realizados anteriormente foram bem sucedidos ou não. A validação pode ser entendida como verificação da generalização do modelo. Uma vez levantado o modelo a partir dos dados de treinamento, a qualidade do modelo pode ser verificada usando simulação (AGUIRRE, 2007).

No desenvolvimento deste trabalho é importante destacar em quais passos da identificação de sistemas serão empregadas as técnicas de otimização. Na abordagem mono-objetivo, a otimização é aplicada apenas na determinação de parâmetros. Por outro lado, na abordagem multiobjetivo aplica-se a otimização tanto na escolha da estrutura do modelo quanto na determinação de parâmetros.

Como exposto em Joberg et al. (1995), no caso da modelagem caixa preta de modelos lineares, o objetivo é simplesmente descrever a resposta ao impulso do sistema. Modelos lineares podem gerar resultados insatisfatórios dependendo, entre outros, do comportamento não linear do processo. Ao mudar de um modelo linear para um não linear pode acontecer do último ter uma performance pior em relação ao primeiro, caso não seja escolhido flexível o suficiente (NELLES, 2002). Uma boa estratégia é usar uma arquitetura não linear que contenha um termo linear como caso especial (NELLES, 2002), gerando uma representação híbrida. Modelos baseados em séries de Volterra atendem a esse requisito.

Quando se trata da modelagem caixa preta de modelos não lineares o nível de detalhamento desejado e a considerável variedade de representações possíveis torna o problema muito mais fatigante (JOBERG et al., 1995). No que tange à diversidade dos tópicos vislumbrados na área, pode-se enumerar, por exemplo, alguns dos diferentes tipos de modelos encontrados na bibliografia: Volterra, *fuzzy*, neurais, NARX, NARMAX (CAMPELLO; OLIVEIRA; AMARAL, 2007; AGUIRRE, 2007).

2.2 Modelos de Volterra

As séries de Volterra são um tipo de aproximação clássica baseada em polinômios para a realização do mapeamento não linear. Sua estrutura pode ser vista como uma generalização direta do modelo *Finite Impulse Response* (FIR), o qual é utilizado em identificação de sistemas lineares (NELLES, 2002; AGUIRRE, 2007; CAMPELLO, 2002). A estrutura utilizada para a representação de processos não lineares por meio do modelo de Volterra é observada em (2.1), que descreve matematicamente a relação entre a saída *y(k)* de um processo físico causal com sua

entrada *u(k)* (RUGH, 1981).

$$y(k) = h_0 + \sum_{k_1=0}^{\infty} h_1(k_1)u(k-k_1) + \sum_{k_1=0}^{\infty}\sum_{k_2=0}^{\infty} h_2(k_1,k_2)u(k-k_1)u(k-k_2) + \\ + \cdots + \sum_{k_1=0}^{\infty}\cdots\sum_{k_m=0}^{\infty} h_m(k_1,\cdots,k_m)u(k-k_1)\cdots u(k-k_m) + \cdots \quad (2.1)$$

Uma das vantagens da utilização das séries de Volterra está no fato destas serem categorizadas como modelos não-paramétricos, uma vez que estes não necessitam da especificação explícita e prévia da ordem do processo. No caso das representações paramétricas, se a ordem do modelo não concorda com a ordem do processo, os resultados obtidos podem ser insatisfatórios (BRAGA, 2011).

Em (2.1), o termo h_0 é visto como um nível constante não nulo (DC) na saída do sistema, enquanto $h_1(k_1)\cdots h_m(k_1,\cdots,k_m)$ são denominados *kernels* da série de Volterra (OROSKI, 2015; BRAGA, 2011).

Os *kernels* da série de Volterra de sistemas dinâmicos estáveis possuem as seguintes propriedades (EYKHOFF, 1974)

- $h_m(k_1,\cdots,k_m) = 0, \qquad \forall k_i < 0 \qquad i = 1,2,3,\cdots,m;$
- $\lim_{k_i\to\infty} h_m(k_1,\cdots,k_m) = 0, \qquad i = 1,2,3,\cdots,m;$
- $h_m(k_1,\cdots,k_m)$ são ou podem ser construídos de forma simétrica.

A primeira propriedade está relacionada ao fato de um sistema físico realizável não depender de valores futuros da entrada, ou seja, o sistema é causal. Além disso, as condições iniciais são nulas (BRAGA, 2011). A segunda propriedade é característica de sistemas com *fading memory*, cujas respostas a determinadas entradas desaparecem com o tempo e deixam de contribuir com a saída após determinado período de tempo finito. Tal propriedade permite realizar o truncamento da série de Volterra sem que qualidade significativa da representação seja perdida (BRAGA, 2011). Finalmente, a terceira propriedade relaciona-se à possibilidade de simetrização dos *kernels*, que reduz a quantidade de termos a serem estimados. Em Braga (2011) é apresentado o procedimento para simetrização desses *kernels*. Essa propriedade reduz consideravelmente o custo computacional em problemas de identificação de sistemas. Isso é comprovado neste trabalho (Capítulo 4).

Se cada elemento dos *kernels* for tratado como um parâmetro independente a ser estimado, o modelo de Volterra torna-se linear nos parâmetros e assim pode-se aplicar algoritmos de estimação clássicos, como, por exemplo, mínimos quadrados. Essa técnica, no entanto, pode sobre-parametrizar o modelo. Uma estratégia é desenvolver os *kernels* da série de Volterra em funções de base ortonormais. Ao realizar tal operação, a complexidade da série é reduzida e, ao proceder com a estimação, o condicionamento numérico do modelo é melhorado e a variância do estimador é reduzida (CAMPELLO; OLIVEIRA; AMARAL, 2007). A próxima seção é destinada à aplicação das FBO em representações por série de Volterra.

2.3 Funções Ortonormais (FBO)

Ao utilizar funções de base ortonormais na obtenção do modelo de Volterra, o conhecimento *a priori* dos *kernels* do sistema não é necessário (BRAGA, 2011), uma vez que os polos da base ortonormal são parâmetros livres de projeto e são os responsáveis pelo conhecimento *a priori* utilizado.

Considera-se a seguinte representação polinomial do modelo de Volterra (CAMPELLO; OLIVEIRA; AMARAL, 2007)

$$y(k) = \sum_{m=1}^{M} \sum_{k_1=0}^{\varepsilon_m} \cdots \sum_{k_1=0}^{\varepsilon_m} h_m(k_1, k_2, ..., k_m) \prod_{j=1}^{m} u(k-k_j). \tag{2.2}$$

Para que os *kernels* h_m possam ser desenvolvidos em funções de base ortonormal estes devem ser absolutamente somáveis em $[0, \infty)$, ou seja, o modelo deve ser estável. Para isso, assume-se por hipótese que $h_m(k_1, k_2, ..., k_m) = 0$ para $k_i > \varepsilon_m$ $(\forall i \in 1, 2, ..., m)$ (CAMPELLO; OLIVEIRA; AMARAL, 2007). Se todos os *kernels* são desenvolvidos em uma mesma base de funções, o desenvolvimento m-dimensional do *kernel* de grau m qualquer descrito por Campello, Oliveira e Amaral (2007) é dado por

$$h_m(k_1, k_2, ..., k_m) = \sum_{i_1=1}^{\infty} \cdots \sum_{i_m=1}^{\infty} c_{i_1,...i_m} \prod_{j=1}^{m} \phi_{i_j}(k_j), \tag{2.3}$$

em que $c_{i_1,...i_m}$ são os coeficientes do desenvolvimento (parâmetros a serem estimados). De (2.2) e (2.3), o modelo de Volterra pode ser reescrito como

$$y(k) = \sum_{m=1}^{M} \sum_{i_1=1}^{\infty} \cdots \sum_{i_m=1}^{\infty} c_{i_1,...i_m} \prod_{j=1}^{m} l_{i_j}(k), \tag{2.4}$$

em que a saída do i-ésimo filtro ortonormal l_{i_j} é dado por

$$l_{i_j} = \sum_{\tau=0}^{\infty} \phi(\tau) u(k-\tau), \tag{2.5}$$

com $\phi(\tau)$ sendo a função de Laguerre no domínio temporal.

Segundo Dumont e Fu (1993) e Campello, Oliveira e Amaral (2007), o truncamento do modelo de ordem 2 $(M = 2)$ em (2.4) é adotado tanto em problemas acadêmicos quanto reais devido a razões práticas. Tal premissa leva à simplificação de (2.4) a

$$y(k) = c_0 + \sum_{i_1=1}^{n_1} c_{i_1} l_{i_1}(k) + \sum_{i_1=1}^{n_2} \sum_{i_2=1}^{n_2} c_{i_1,i_2} l_{i_1}(k) l_{i_2}(k), \tag{2.6}$$

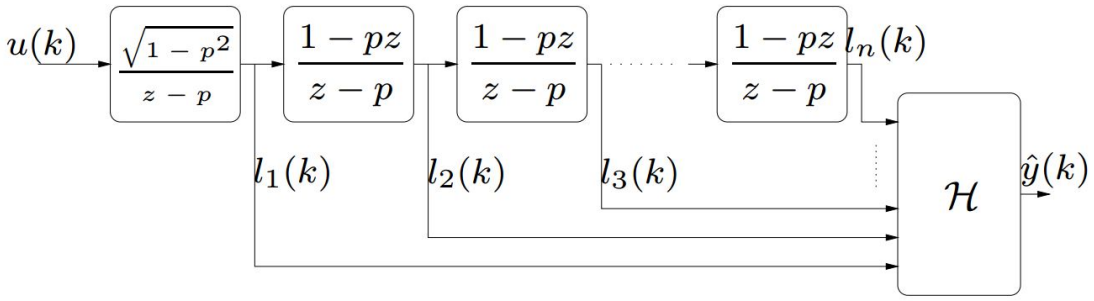
em que c_0 é um coeficiente de ordem zero que representa um nível não nulo na saída.

2.3.1 Funções de Laguerre

A base de Laguerre incorpora apenas um conjunto de polos reais no modelo do sistema. Se comparado às bases de Kautz e GOBF (*Generalized Orthonormal Basis Functions*), a base

de Laguerre tem como desvantagem a impossibilidade de trabalhar com polos complexos no modelo do sistema. No entanto, as funções de Laguerre envolvem uma parametrização muito menos complexa da base ortonormal (BRAGA, 2011). A Figura 4 apresenta um modelo FBO com funções de Laguerre. O sinal de entrada, $u(k)$, passa pelos diversos estágios do filtro de Laguerre para que, posteriormente, a saída de cada filtro passe por um mapeamento híbrido (parte linear e parte não linear) para obter a saída do modelo, $\hat{y}(k)$.

Figura 4 – Modelo FBO com dinâmica de Laguerre.



Fonte: Campello, Oliveira e Amaral (2007).

A expressão no domínio z das funções de Laguerre pode ser escrita como (CAMPELLO; OLIVEIRA; AMARAL, 2007)

$$\Phi_i(z) = \frac{\sqrt{1-p^2}}{z-p}\left(\frac{1-pz}{z-p}\right)^{i-1}, \tag{2.7}$$

em que $i = 1,2,\ldots$, e $p \in \{\Re : |p| < 1\}$ é o polo estável que parametriza as FBO. Se $p = 0$ em (2.7), $\Phi_i(z) = z^{-i}$. Logo, tem-se o modelo FIR.

Representar funções que possuem pares de polos complexos dominantes por meio das funções de Laguerre faz com que um número elevado de termos seja necessário. Isso acontece porque os polos em (2.7) ficam longe dos polos conjugados do sistema (BRAGA, 2011). Dessa forma, pode ser interessante implementar as FBO por funções de Kautz, embora não seja objeto de estudo deste trabalho.

2.3.2 Determinação dos parâmetros via Mínimos Quadrados

Dadas as funções de base ortonormais, descritas em (2.7) e tendo-se o mapeamento estático do modelo FBO-Volterra em (2.6), a matriz de regressores Ψ é

$$\Psi = [1 \;\; l_1(k) \;\; \cdots \;\; l_{n_1}(k) \;\; l_1(k)l_1(k) \;\; l_1(k)l_2(k) \;\; \cdots \;\; l_1(k)l_{n_2}(k) \;\; l_{n2}(k)l_1(k) \cdots \;\; l_{n2}(k)^2] \tag{2.8}$$

e

$$\hat{\zeta} = [c_0 \;\; c_1 \;\; \cdots \;\; c_{n_1} \;\; c_{1,1} \;\; c_{2,1} \;\; c_{2,2} \;\; \cdots \;\; c_{n2,1} \cdots \;\; c_{n2,n2}]^T \tag{2.9}$$

é o vetor de parâmetros a ser estimado, que pode ser obtido fazendo

$$\hat{\zeta} = (\Psi^T\Psi)^{-1}\Psi^T y, \tag{2.10}$$

em que y é a saída real do sistema.

3 Otimização: Algoritmos Meta-heurísticos

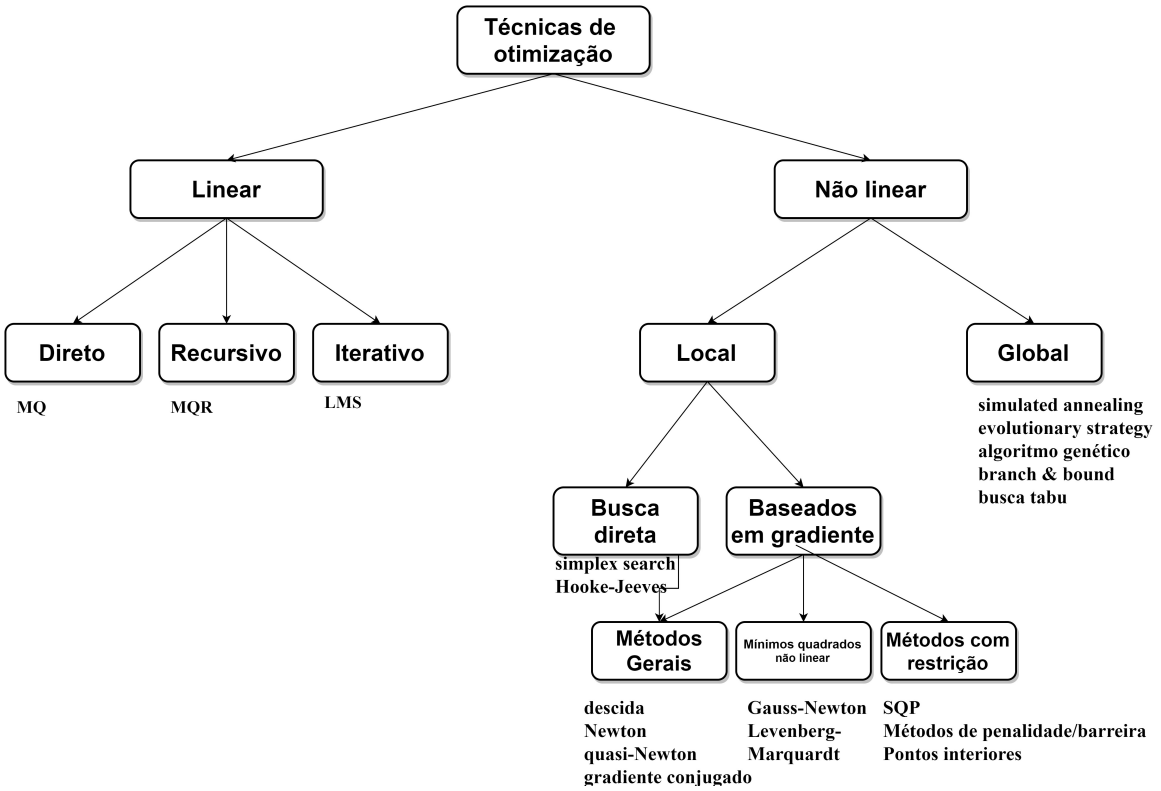
3.1 Introdução

Muitos problemas de engenharia podem ser formulados como problemas de otimização, cujo objetivo é minimizar ou maximizar (dependendo do problema) uma função custo *f(x)* levando em consideração uma série de restrições *g(x)* em que *x* é considerado como pertencente ao conjunto de soluções *S* (ANSARI; HOU, 2012). A expressão matemática para o caso em que o objetivo é a minimização do custo é

$$\min_{x \in S} f(x) \qquad \text{sujeito a restrições } g(x). \tag{3.1}$$

As técnicas utilizadas para a otimização dependem do tipo de problema em questão. Se *f(x)* e *g(x)* em (3.1) forem funções lineares, o problema de otimização pode ser facilmente resolvido. Caso contrário, a resolução se torna mais complicada (ANSARI; HOU, 2012). A Figura 5 apresenta uma visão geral e sucinta das técnicas de otimização mais utilizadas.

Figura 5 – Visão geral da otimização linear e não linear.



Fonte: Adaptado de Nelles (2002).

Problemas lineares podem ser resolvidos facilmente a partir de algoritmos determinísticos. Assim, a solução ótima pode ser encontrada em um número finito de etapas. A maioria dos

problemas encontrados na engenharia são classificados como "difíceis de resolver" em que o espaço de busca da solução ótima é muito grande. A tal classe de problemas geralmente são aplicados os métodos heurísticos de busca. O intuito é utilizar certas informações que direcionem a busca de modo que o tempo computacional gasto seja reduzido.

A título de exemplo, considera-se o Problema do Caixeiro Viajante (PCV). Tal problema consiste em encontrar a menor rota possível para o caixeiro viajante, que sai de uma determinada cidade e deve visitar todas as $n-1$ cidades restantes e retornar à cidade de partida. Para isso, é usada uma matriz simétrica quadrada de tamanho n com informações das distâncias entre as cidades. Como evidenciado por Souza (2011), o custo computacional para a exploração de todas as possíveis soluções do PCV é extremamente alto. Isso é evidente, pois o número de rotas possíveis é de $(n-1)!/2$. Caso o caixeiro viajante precise passar por 20 cidades, o número de rotas possíveis é de 6×10^{16}. Se um computador avaliar cada rota em 10^{-8} segundos, seriam necessários 19 anos para encontrar a rota ótima. Essa estimativa evidencia a dificuldade para a resolução de um problema razoavelmente simples. Ao utilizar as heurísticas, problemas como esse são resolvidos em minutos. Mostrando, assim, a importância dessas técnicas na engenharia.

Os problemas de otimização geralmente são resolvidos utilizando heurísticas construtivas e heurísticas de refinamento. As heurísticas construtivas, como o nome sugere, tem como objetivo construir uma solução inicial para o problema. Como exemplos de heurísticas construtivas pode-se citar a construção gulosa e a construção aleatória (SOUZA, 2011).

Uma vez gerada a solução inicial, a melhora dessa solução é feita a partir das heurísticas de refinamento. Também chamadas de técnicas de busca local, essas heurísticas são baseadas na noção de vizinhança. Considera-se o espaço de soluções S em (3.1). Supõe-se que uma determinada solução $x \in S$ tem uma função $N \subseteq S$ representando a sua vizinhança. As heurísticas de refinamento realizam um movimento de modo a gerar uma nova solução $x' \in N$. Essa nova solução x' é dita vizinha de x. A solução é obtida (partindo de uma solução inicial) "caminhando", a cada iteração, por essas soluções vizinhas (SOUZA, 2011). De uma determinada solução é necessário que seja possível alcançar qualquer outra solução a partir de um número finito de passos. Alguns dos procedimentos de refinamento são: Método da Descida/Subida, Método de Primeira Melhora, Método de Descida/Subida Randômica, VND (*Variable Neighborhood Descent*). Esses métodos variam entre si pela forma em que realizam as buscas por ótimos locais.

Uma das desvantagens de utilizar as heurísticas está no fato de que essas ficam presas com muita facilidade a ótimos locais (ainda distantes do ótimo global). Além disso, as heurísticas são montadas de acordo com cada problema, ou seja, não se adaptam facilmente a diferentes aplicações (SOUZA, 2011). Assim, introduziu-se os algoritmos meta-heurísticos. O prefixo de origem grega "meta", presente no nome, indica o comportamento "alto nível" das heurísticas, caracterizando a fácil adaptatividade do método a diferentes problemas (BOUSSAÏD; LEPAGNOT; SIARRY, 2013). A maioria das meta-heurísticas compartilham das seguintes características:

- são providas de mecanismos para fuga de ótimos locais;

- inspiradas em fenômenos naturais (princípios da física, biologia, etologia);
- utilizam ferramentas estocásticas (envolvendo variáveis aleatórias);
- possuem parâmetros que precisam ser ajustados de acordo com o problema em questão.

Basicamente, as meta-heurísticas diferem entre si pelo método utilizado para fuga dos ótimos locais. Elas devem ser capazes de resolver diversas categorias de problemas: discretos e contínuos, com ou sem restrições, mono ou multiobjetivo, estáticos ou dinâmicos. As meta-heurísticas podem ser divididas em dois grupos, de acordo com o método utilizado para exploração das soluções: aquelas baseadas em busca local e as baseadas em busca populacional. Alguns dos métodos mais conhecidos, classificados em grupos, são mostrados na Tabela 1.

Tabela 1 – Técnicas meta-heurísticas separadas por tipo de busca.

Busca Local – *(Single-solution)*	**Busca populacional – *(Population-based)***
Busca Tabu	Algoritmos Genéticos (GA)
Simulated Annealing	Algoritmos Meméticos
Busca em Vizinhança Variável (VNS)	Algoritmo Colônia de Formigas
Iterated Local Search (ILS)	*Particle Swarm Optimization* (PSO)

Fonte: Do autor

Na sequência serão apresentadas algumas das técnicas meta-heurísticas mais conhecidas e que são utilizadas neste trabalho.

3.2 *Simulated Annealing*

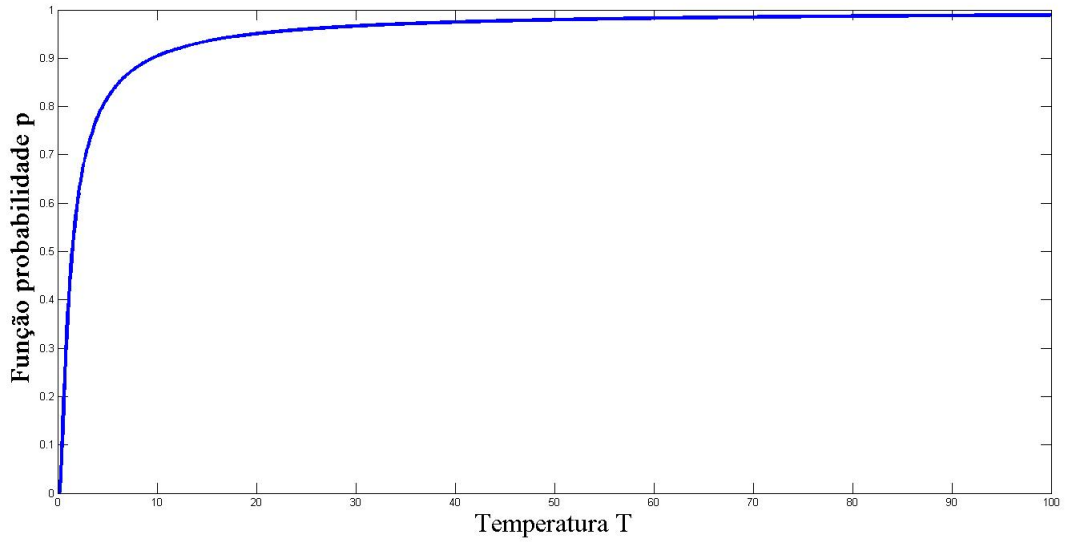
A meta-heurística *Simulated Annealing* é originária da analogia entre a natureza estatística dos movimentos de partículas em substâncias líquidas ou sólidas (Algoritmo Metrópolis (METROPOLIS et al., 1953)) e problemas complexos de otimização (ANSARI; HOU, 2012). Inspirada em técnicas de recozimento usadas por metalúrgicos para obtenção de sólidos "bem ordenados" (cristais) de energia mínima. Para tal, é feito o resfriamento gradual de materiais em alta temperatura à temperaturas mais baixas. É baseada em busca local probabilística, realizando sua busca a partir de uma solução inicial aleatória.

Seja um determinado problema de minimização qualquer (discreto ou contínuo), o algoritmo *Simulated Annealing* inicia gerando uma solução utilizando uma heurística construtiva. Além disso, uma temperatura (fictícia) inicial T_0 é definida. A cada iteração uma nova solução *x'* é obtida. Caso o custo *f(x')* seja menor que o custo da solução corrente *f(x)*, a atualização da solução ótima é realizada. Caso contrário, é realizado o cálculo de probabilidade de aceitação de soluções piores *p(T,f(x),f(x'))*. A função p é computada como

$$p(T, f(x), f(x')) = e^{-\Delta/T}, \tag{3.2}$$

em que T é a temperatura corrente e $\Delta = f(x') - f(x)$. Para valores altos de T, maior é p. A cada iteração essa temperatura é decrementada e, consequentemente, a probabilidade de movimentos de piora diminui. O comportamento da função p pode ser visto na Figura 6 .

Figura 6 – Função probabilidade de movimentos de piora do algoritmo *Simulated Annealing*.



Fonte: Do autor.

O resfriamento (ou decréscimo da temperatura) pode ser feito geometricamente por meio de uma variável chamada razão de resfriamento, representada pela letra α (SOUZA, 2011). A temperatura na k-ésima iteração é dada por

$$T_k = \alpha T_{k-1}, \tag{3.3}$$

em que $0 < \alpha < 1$. Assim, junto com a temperatura inicial T_0 e o número máximo de iterações (*SAmax*), a razão de resfriamento α é um parâmetro de controle do algoritmo *Simulated Annealing*. A Figura 7 mostra o pseudocódigo para esse método.

A determinação dos parâmetros de controle geralmente é feita por experimentação (SOUZA, 2011) . Um valor usual para a razão de resfriamento é $\alpha = 0,9$ (KIRKPATRICK et al., 1983). É importante ressaltar que para valores de α muito altos (próximos de 1), o tempo de execução do algoritmo será consideravelmente maior, pois a temperatura decairá muito lentamente. Por outro lado, caso α seja muito pequeno, a temperatura decairá muito rapidamente e o algoritmo dificilmente convergirá para o ótimo global. O número máximo de iterações pode ser estimado de acordo com as dimensões do problema. Para o problema do caixeiro viajante, por exemplo, com n cidades, a determinação do número máximo de iterações pode ser dado por $SAmax = k \times n$, em que k é um parâmetro inteiro a ser determinado.

A estimação da temperatura inicial T_0 requer um pouco mais de esforço, sua determinação pode ser feita de duas maneiras distintas, como enumerado a seguir.

Figura 7 – Algoritmo *Simulated Annealing*.

```
procedimento SA(f(.), N(.), α, SAmax, T0, s)
1  s* ← s;                    {Melhor solução obtida até então}
2  IterT ← 0;                 {Número de iterações na temperatura T}
3  T ← T0;                    {Temperatura corrente}
4  enquanto (T > 0) faça
5      enquanto (IterT < SAmax) faça
6          IterT ← IterT + 1;
7          Gere um vizinho qualquer s' ∈ N(s);
8          Δ = f(s') − f(s);
9          se (Δ < 0)
10             então
11                 s ← s';
12                 se (f(s') < f(s*)) então s* ← s';
13             senão
14                 Tome x ∈ [0, 1];
15                 se (x < e^(−Δ/T)) então s ← s';
16         fim-se;
17     fim-enquanto;
18     T ← α × T;
19     IterT ← 0;
20 fim-enquanto;
21 s ← s*;
22 Retorne s;
fim SA;
```

Fonte: Souza (2011).

1. Estimação por simulação

   a) Gerar uma solução inicial;

   b) Partir de uma temperatura inicial baixa ($T_0 = 1$, por exemplo);

   c) Contabilizar a quantidade de vizinhos aceitos em *SAmax* iterações nessa temperatura;

   d) Caso o número de vizinhos aceitos seja alto (95%, por exemplo) retornar a temperatura corrente como a temperatura inicial para o método;

   e) Caso contrário, aumentar a temperatura (em 10%, por exemplo) e repetir o processo.

2. Estimação pela média de custos das soluções vizinhas

   a) Gerar uma solução inicial;

   b) Gerar um certo número de vizinhos;

   c) Calcular o respectivo custo para cada vizinho;

   d) Retornar como temperatura o maior custo das soluções vizinhas.

A Figura 8 apresenta o procedimento para determinação autoadaptativa da temperatura inicial por simulação.

Figura 8 – Procedimento de determinação autoadaptativa da temperatura inicial - Algoritmo *Simulated Annealing*.

```
procedimento TemperaturaInicial(f(.), N(.), β, γ, SAmax, T0, s)
1  T ← T0;            {Temperatura corrente}
2  Continua ← TRUE;
3  enquanto (Continua) faça
4      Aceitos ← 0; {Número de vizinhos aceitos na temperatura T}
5      para IterT = 1 até SAmax faça
6          Gere um vizinho qualquer s' ∈ N(s);
7          Δ = f(s') − f(s);
8          se (Δ < 0)
9              então
10                 Aceitos ← Aceitos + 1;
11             senão
12                 Tome x ∈ [0, 1];
13                 se (x < e^(−Δ/T)) então Aceitos ← Aceitos + 1;
14         fim-se;
15     fim-para;
16     se (Aceitos ≥ γ × SAmax)
17         então Continua ← FALSE;
18         senão T ← β × T;
19     fim-se;
20 fim-enquanto;
21 Retorne T;
fim TemperaturaInicial;
```

Fonte: Souza (2011).

Na teoria, a temperatura final deveria ser zero. No entanto, como pode ser visto em (3.3) essa temperatura nunca atingirá esse valor. Portanto, na prática, um valor próximo de zero é aceitável. Um valor usual para a temperatura final é $T_f = 0,0001$ (SOUZA, 2011).

O método *Simulated Annealing* tem sido utilizado em diversos tipos de problemas e tem apresentado resultados satisfatórios, exceto para o caso de problemas combinatórios em que o método tem se mostrado excessivamente guloso (converge muito rapidamente sem garantia de encontrar um ótimo global) ou incapaz de resolvê-los (BOUSSAÏD; LEPAGNOT; SIARRY, 2013).

3.3 Algoritmo Genético

O algoritmo genético (*Genetic Algorithm* - GA) é uma técnica de otimização evolutiva baseada nos princípios da genética e seleção natural. É a técnica evolucionária mais conhecida e utilizada (BOUSSAÏD; LEPAGNOT; SIARRY, 2013). O GA permite a evolução de uma população a partir de critérios de seleção definidos que buscam obter os indivíduos capazes de se adequar melhor aos objetivos traçados na otimização (minimização da função custo, por exemplo) (HAUPT; HAUPT, 2004). Os indivíduos são capazes de gerar filhos que se adequam melhor, enquanto os indivíduos menos aptos tendem a desaparecer. Assim, a cada nova geração mais a população tende, de modo geral, a se adaptar melhor aos objetivos.

O algoritmo começa com a definição de uma população inicial que é geralmente aleatória. Cada indivíduo pode ser entendido como o responsável por uma solução do problema. Esses indivíduos são analisados pela função custo, que determina a aptidão de cada um. Supõe-se uma população inicial de M indivíduos. A primeira geração é obtida a partir da reprodução e a seleção de sobreviventes. Assim, se N novos indivíduos são gerados, a população passa a ser de $M+N$ indivíduos. Não é interessante o crescimento populacional. Convém, portanto, selecionar quais serão os sobreviventes de modo que o tamanho da população na geração seguinte seja sempre o mesmo da inicial (M).

A atualização da população pode acontecer de duas maneiras distintas: reprodução sexuada e mutação (reprodução assexuada). A seleção dos pais para a reprodução ou dos indivíduos a sofrerem mutação pode ser feita de várias maneiras. No método de torneio, por exemplo, é realizado o torneio de x indivíduos e só um será selecionado como pai. Da mesma forma o segundo pai é selecionado. Outra maneira de selecionar esses indivíduos é a seleção aleatória. A reprodução acontece a partir da combinação dos genes dos pais. Essa operação é denominada *crossover*. O cromossomo filho é composto por frações dos genes de cada pai. A mutação é realizada alterando-se aleatoriamente parte do gene de um indivíduo. É atribuída uma probabilidade de ocorrência à reprodução sexuada e outra à mutação. Geralmente, a probabilidade de ocorrência da segunda é consideravelmente menor (SOUZA, 2011; BOUSSAÏD; LEPAGNOT; SIARRY, 2013).

Os novos indivíduos são avaliados (função de aptidão) e se juntam ao restante da população para a seleção dos sobreviventes. Essa seleção é feita utilizando-se as técnicas de torneio, aleatória e roleta (probabilidade de sobrevivência proporcional à aptidão). Todas as técnicas permitem que indivíduos menos aptos sobrevivam. Isso é importante devido ao fato de que esses indivíduos podem ser os responsáveis pela fuga de ótimos locais (SOUZA, 2011).

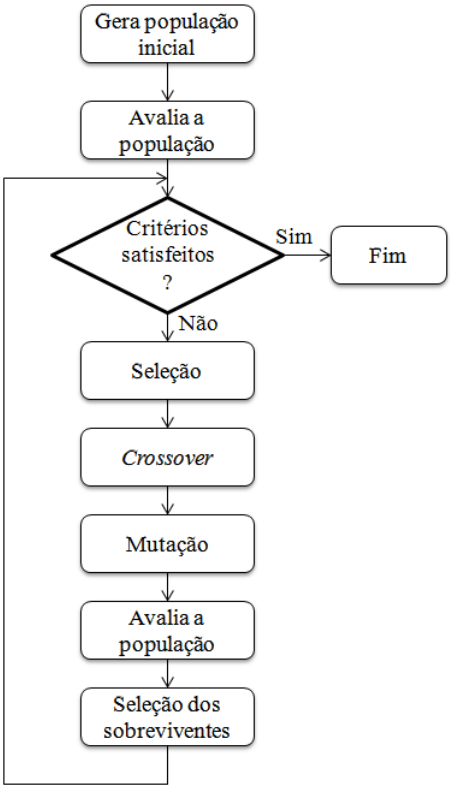
Algumas das vantagens da utilização dos algoritmos genéticos são (HAUPT; HAUPT, 2004)

- Aplicáveis tanto a problemas contínuos quanto a problemas discretos;
- Fornecem conjuntos de soluções ótimas (diferentemente do algoritmo *Simulated Annealing*, por exemplo, que fornece apenas uma);
- Não necessitam de informações de derivada;
- Trabalham com um elevado número de variáveis;
- Pesquisam simultaneamente uma amostra vasta da superfície de custo;
- São altamente paralelizáveis;
- São aplicáveis a diferentes tipos de dados: dados experimentais, dados numéricos e funções analíticas;

- Podem fugir de mínimos locais.

A estrutura básica de um Algoritmo Genético qualquer é visto na Figura 9. O pseudocódigo para essa técnica é mostrado na Figura 10.

Figura 9 – Fluxograma do Algoritmo Genético.



Fonte: Adaptado de Maldonado, Castillo e Melin (2013).

Figura 10 – Procedimento básico do algoritmo genético.

procedimento *AG*
1 $t \leftarrow 0$;
2 Gere a população inicial $P(t)$;
3 Avalie $P(t)$;
4 enquanto (os critérios de parada não estiverem satisfeitos) faça
5 $t \leftarrow t + 1$;
6 Gere $P(t)$ a partir de $P(t-1)$;
7 Avalie $P(t)$;
8 Defina a população sobrevivente;
9 fim-enquanto;
fim *AG*;

Fonte: Souza (2011).

Os parâmetros de projeto do GA são

- Tamanho da população: Quanto maior for a população, mais buscas em paralelo são realizadas;
- Tamanho da população de filhos: Está diretamente relacionada à diversificação/foco (*exploration/exploitation*) do algoritmo. Quanto maior a população de filhos, mais diversificada será a nova geração;
- Probabilidade de operação do *crossover*;
- Probabilidade de mutação (geralmente 1%);
- Número de gerações.

3.4 Otimização por Enxame de Partículas (PSO)

O PSO pertence à classe dos algoritmos SI (*Swarm Intelligence*). O SI é uma inteligência coletiva baseada em princípios psico-sociológicos, em que os indivíduos obtêm consciência perceptiva por meio da influência e aprendizagem social (ANDRÉ, 2007). Assim, a tomada de decisões de um indivíduo é definida por uma parcela de experiência própria e outra relacionada a sociedade em que esse se encontra inserido. Além disso, as percepções de um indivíduo mudam conforme este interage. Algoritmos do tipo SI apresentam um método inovador de solução de problemas inspirando-se no comportamento coletivo de grupos de colônias de insetos e também de sociedades de outros animais. Um sistema SI é composto por uma população de agentes capazes de executar certas operações. Cada agente apresenta recursos limitados. No entanto, havendo a cooperação e colaboração entre os indivíduos, a população é capaz de resolver problemas de alta complexidade (BOUSSAÏD; LEPAGNOT; SIARRY, 2013).

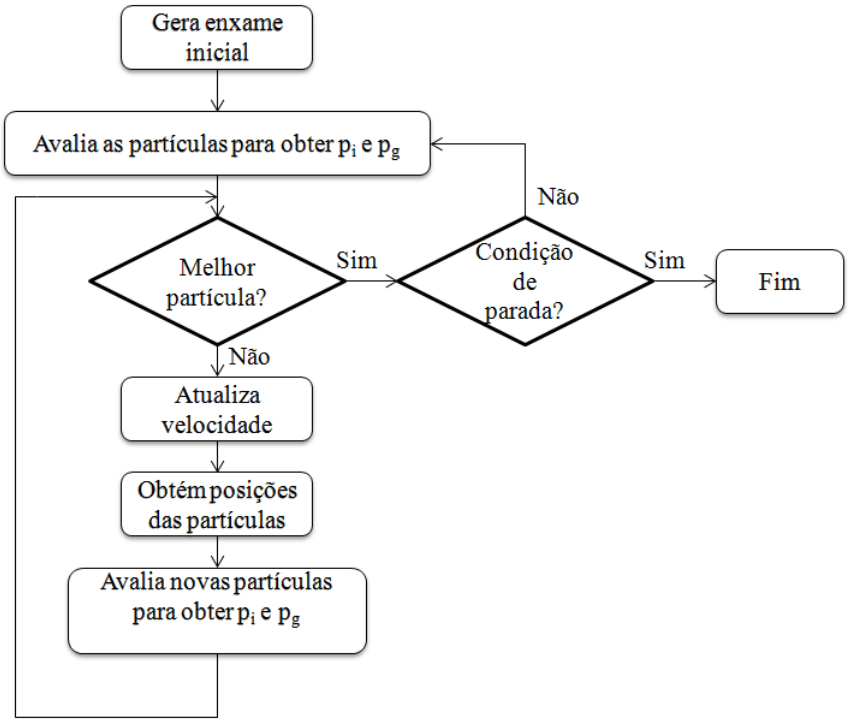
A otimização por enxame de partículas é baseada no movimento de grupos de pássaros. Cada pássaro pode ser interpretado como uma partícula. O conjunto de partículas é gerado de forma estocástica no espaço de busca e cada partícula é uma solução candidata do problema (ANDRÉ, 2007). Um enxame contém N partículas voando por um espaço de busca multidimensional. Cada enxame é representado por sua velocidade V_t^i (velocidade da i-ésima partícula na iteração t), sua localização no espaço de busca x_t^i (posição da i-ésima partícula na iteração t) e a informação da sua última melhor posição p_t^i. Nas Figuras 11 e 12 são mostrados o fluxograma e o pseudocódigo do PSO, respectivamente.

No início do procedimento, as posições e velocidades são inicializadas de maneira aleatória. A cada iteração essas informações são atualizadas para todas as partículas. A posição atualizada é dada por $x_{t+1}^i = x_t^i + V_{t+1}^i$. A atualização da última melhor posição é feita caso p_{t+1}^i seja melhor que p_t^i. A velocidade pode ser atualizada computando

$$V_{t+1}^i = \omega V_t^i + \varphi_1 R_{1t}^i (p_t^i - x_t^i) + \varphi_2 R_{2t}^i (p_g - x_t^i), \tag{3.4}$$

em que ω é chamado de coeficiente de inércia (usado para evitar convergência prematura do PSO), φ_1 e φ_2 são a porcentagem da influência do próprio sucesso e o sucesso social, respectivamente.

Figura 11 – Fluxograma para o algoritmo PSO.



Fonte: Adaptado de Maldonado, Castillo e Melin (2013).

Já R^i_{1t} e R^i_{2t} são matrizes diagonais com números aleatórios entre 0 e 1. Além disso, p_g representa o melhor global.

Caso $\omega = 1$ em (3.4), a atualização é classificada como *original*. Se todos os termos diagonais das matrizes R_1 e R_2 forem iguais, a atualização é *linear* e, caso contrário, *padrão* (BOUSSAÏD; LEPAGNOT; SIARRY, 2013; BONYADI; MICHALEWICZ, 2016). A Figura 13 mostra o espaço de busca onde $\varphi_1 R^i_{1t}$ e $\varphi_2 R^i_{2t}$ podem ser encontrados. Fica evidente a diferença entre o comportamento da busca para atualização padrão e linear da velocidade.

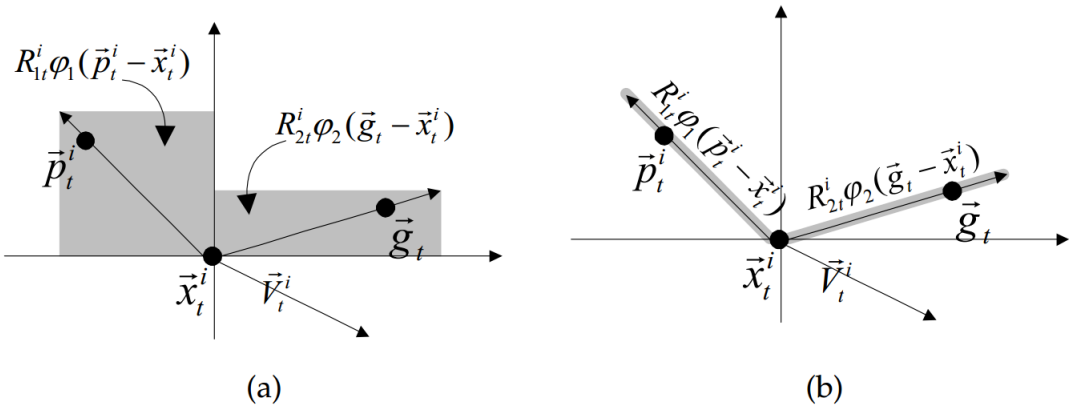
Todo enxame possui uma topologia descritiva do tipo de conexão entre as partículas. Em (3.4), a topologia utilizada é a *estrela*, em que faz-se o uso dos melhores local e global. Uma topologia alternativa é a *aleatória*, em que é considerado o melhor local de uma partícula aleatória, ao invés do melhor global. Há ainda as topologias em *anel* (ou círculo) e *roda*. No caso da topologia em anel, apenas duas partículas vizinhas são consideradas no cálculo do melhor global, ao passo que na topologia em roda uma partícula fixa é escolhida para a atualização da velocidade de todas as outras partículas.

Figura 12 – Pseudocódigo para o algoritmo PSO.

```
procedimento PSO
1   Gera população de partículas com posições e velocidades aleatórias no espaço
    multidimensional de soluções;
2   avalia enxame;
3   enquanto (os critérios de parada não estiverem satisfeitos) faça
4       para  cada partícula i faça
5           atualiza velocidade;
6           atualiza posição;
7           avalia custo da partícula i;
8           se (custo da partícula i  for melhor que p_i) então
9               p_i ← custo da partícula i;
10          fim-se
11          se (custo da partícula i  for melhor que p_g) então
12              p_g ← custo da partícula i;
13          fim-se ;
14      fim-para ;
15  fim-enquanto ;
fim PSO;
```

Fonte: Adaptado de BoussaïD, Lepagnot e Siarry (2013).

Figura 13 – As áreas em cinza representam onde $\varphi_1 R^i_{1t}$ e $\varphi_2 R^i_{2t}$ podem ser encontrados, (a) atualização padrão, (b) atualização linear.



Fonte: Bonyadi e Michalewicz (2016)

4 Otimização Multiobjetivo

4.1 Introdução

Em muitos casos reais, problemas de otimização apresentam mais de um objetivo. O intuito da otimização multiobjetivo é encontrar um conjunto de soluções ótimas que satisfaçam a uma gama de metas obedecendo a um conjunto de restrições (PANTUZA, 2011). Do conjunto de soluções obtido, cabe ao engenheiro/pesquisador realizar a tomada de decisão de acordo com o problema em questão.

Como visto em Pantuza (2011), um problema de otimização multiobjetivo pode ser formulado como

$$\min z = f(x) = (f_1(x), f_2(x), \cdots, f_{OB}(x))$$

$$\text{sujeito a restrições } g(x) = (g_1(x), g_2(x), \cdots, g_r(x)) \leq b$$
$$x = (x_1, x_2, \cdots, x_n) \in X$$
$$z = (z_1, z_2, \cdots, z_r) \in Z,$$

em que x é o vetor de decisão, OB é o número de objetivos, z é o vetor objetivo, e X é o espaço de busca de decisões.

Ao conjunto de soluções ótimas dá-se o nome de Pareto-ótimo. Um vetor z pertence ao conjunto Pareto-Ótimo se não existe nenhum outro vetor z^* que possa melhorar algum objetivo, sem causar piora em pelo menos um outro objetivo (PANTUZA, 2011). Quando uma solução é pior do que a outra em todos os objetivos, ela está dominada por esta outra.

Métodos de otimização mono-objetivo podem ser utilizados para resolver um problema multiobjetivo, entretanto devem ser executados múltiplas vezes com condições diferentes de forma a levantar o conjunto de Pareto (DEB et al., 2002). Assim, se um problema for biobjetivo, por exemplo, "trava-se" um dos objetivos e obtém-se o resultado da otimização para aquele ponto por meio de otimização mono-objetivo. O levantamento das Tabelas 2-7 apresentadas no Capítulo 5 é feito fixando-se os valores de n_1 e n_2, que são o número de elementos utilizados em cada base ortonormal conforme (2.6), e, consequentemente, o número de parâmetros que será, neste trabalho, um dos objetos de minimização.

4.2 Algoritmo Genético Multiobjetivo

Uma das metaheurísticas mais comumente utilizadas em problemas de otimização multiobjetivo é o Algoritmo Genético. Esse tipo de algoritmo pertence à classe dos *MOEA* (*Multiobjective Evolutionary Algorithms*) (DEB et al., 2002). Em Pantuza (2011) pode-se encontrar um breve histórico do desenvolvimento dessa classe de algoritmos.

4.2.1 NSGA-II

O algoritmo multiobjetivo escolhido para aplicação neste trabalho é o NSGA-II (*Non-dominated Sorting Genetic Algorithm II*). A escolha desse algoritmo se deu, principalmente, por sua fácil implementação. O NSGA-II foi proposto por Deb et al. (2002) como uma melhora do NSGA (SRINIVAS; DEB, 1994).

Trata-se de um algoritmo baseado em uma ordenação elitista por dominância, que busca classificar os indivíduos de um conjunto em níveis relacionados ao seu grau de dominância (PANTUZA, 2011). A Figura 14 apresenta o pseudocódigo do algoritmo *fast non-dominanted sort*, que é responsável por ordenar as soluções dentro da população em diferentes frentes de Pareto.

Figura 14 – Pseudocódigo do algoritmo *fast non-dominanted sort*.

```
fast-non-dominated-sort(P)
para cada p ∈ P
    se (p ≺ q) então                    se p domina q
       S_p = S_p ∪ {q}                  adiciona q ao conjunto de soluções dominadas por p
    senão
       se (q ≺ p) então
            n_p = n_p + 1               incrementa o contador de dominantes de p
       fim se
    fim se
    se n_p = 0 então
       p_rank = 1
       F_1 = F_1 ∪ {p}
    fim se
    i=1                                 inicializa o contador da frente
fim para
enquanto F_i ≠ Ø
    Q = Ø                               usado para armazenar os membros da próxima frente
    para cada p ∈ F_i
       para cada q ∈ S_p
            n_q = n_q + 1
            se n_q = 0 então            q pertence à próxima frente
               q_rank = i+1
               Q = Q ∪ {q}
            fim se
       fim para
    fim para
    i = i + 1
    F_i = Q
fim enquanto
```

Fonte: Adaptado de Deb et al. (2002).

Figura 15 – Pseudocódigo do algoritmo *crowding distance*.

```
crowding-distance-assigment(I)
l = |I|                                  número de soluções em I
para cada i, faça I[i]_distância = 0      inicializa a distância
para cada objetivo m
    I = ordena(I,m)                      ordena usando cada valor do objetivo
    I[1]_distância = I[l]_distância = ∞   de modo que pontos na fronteiras são sempre
    para i = 2 até (l-1)                 selecionados para todos os outros pontos
        I[i]_distância = I[i]_distância + (I[i+1].m - I[i-1].m)/(f_m^max - f_m^min)
fim para
```

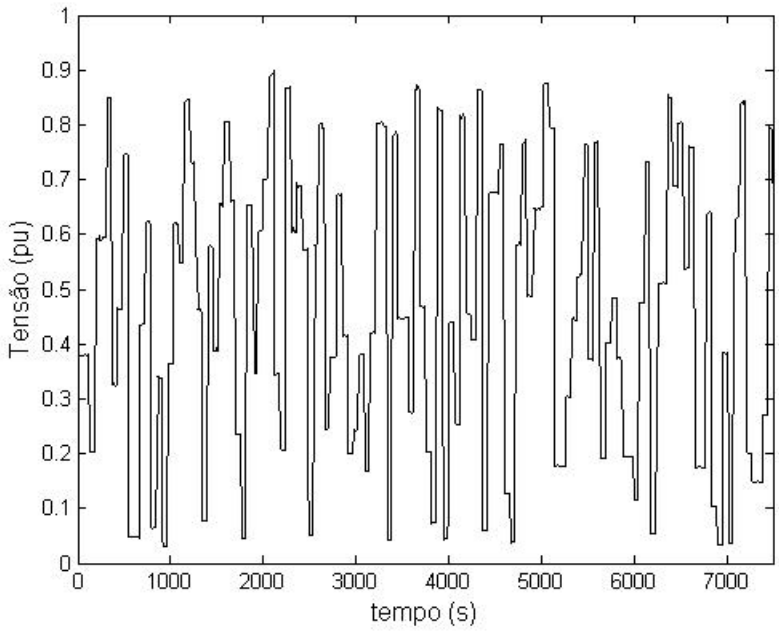
Fonte: Adaptado de Deb et al. (2002).

Outro critério utilizado pelo NSGA-II é a avaliação da distância de uma solução para outra dentro de uma mesma Frente de Pareto, para garantir a diversidade da frente de Pareto (DEB et al., 2002). A Figura 15 apresenta o pseudocódigo do algoritmo *crowding distance*, que é responsável pelo cálculo da distância entre duas soluções dentro da mesma frente de Pareto (busca-se sempre maximizar esta distância, de modo a garantir uma maior diversidade dentro da Frente de Pareto).

5 Resultados e Discussão

O sistema usado como exemplo para a identificação neste trabalho, é o sistema de aquecimento com dissipação variável encontrado em Aguirre (2007). O problema consiste de um sistema, cuja entrada é a tensão (em Volts) aplicada a um aquecedor e a saída é a temperatura (em graus Celsius). O experimento contou com o elemento responsável pela dissipação variável da temperatura, no caso, um ventilador. Para o teste dinâmico, foi aplicada uma tensão aleatória para alimentação do aquecedor e a temperatura foi medida por um termopar. O experimento completo durou cerca de 15000 segundos, sendo que metade desses dados foram usados para a identificação do sistema (7500 primeiros segundos) e o restante para a validação do modelo. Nas Figuras 16 e 17 são mostrados os sinais de entrada e de saída, respectivamente. Tanto a entrada quanto a saída estão em p.u. (por unidade), sendo que 1 p.u. da entrada equivale a 136 Volts (RMS) e 1 p.u. da saída corresponde a 998,51 °C.

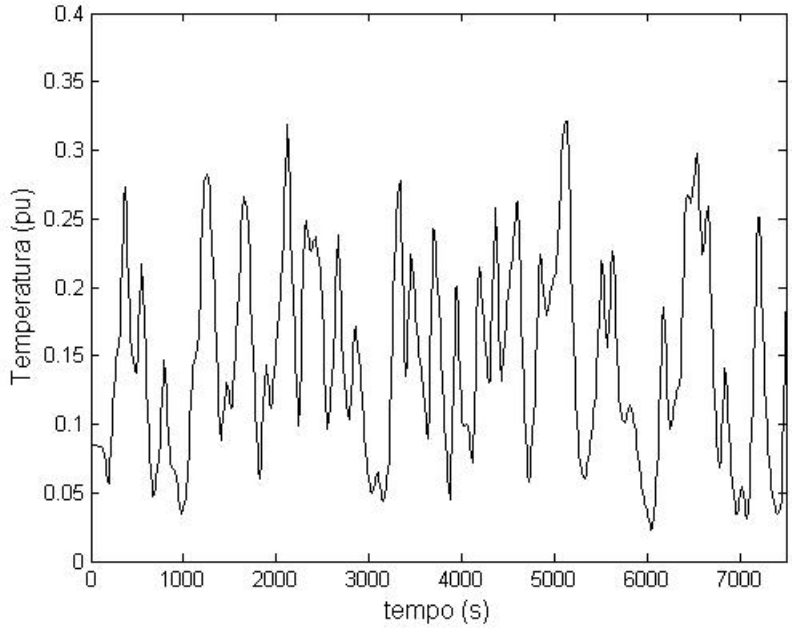
Figura 16 – Dados de entrada do aquecedor com dissipação variável (identificação).



Fonte: Aguirre (2007).

O sistema será representado pelo modelo FBO-Volterra, previamente apresentado. O problema deste trabalho resume-se a obter os polos ótimos das funções de Laguerre que parametrizam o sistema. O polo será ótimo quando o erro quadrático médio (EQM) entre a saída real do sistema (Figura 17) e a saída do modelo obtido for mínimo. Assim, se a saída real do sistema for y e a saída do modelo obtido por identificação for y^{modelo} então EQM será (WANG; BOVIK,

Figura 17 – Dados de saída do aquecedor com dissipação variável (identificação).



Fonte: Aguirre (2007).

2009)

$$EQM = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_i^{modelo})^2. \tag{5.1}$$

Para obter y^{modelo} basta utilizar técnicas de estimação convencionais, neste caso, será empregado o estimador de Mínimos Quadrados (MQ). Isso só é possível devido ao fato das FBO serem lineares nos parâmetros.

Finalmente, o problema de otimização pode ser modelado considerando a função de custo como sendo responsável por calcular o erro de estimação (EQM) no caso mono-objetivo e o EQM e número de parâmetros, no caso multiobjetivo. Para isso, a função deve receber os polos do filtro de Laguerre (Figura 4) e retornar o EQM para o polo em questão. As meta-heurísticas serão responsáveis pela geração dos polos candidatos e seleção do polo ótimo por meio da minimização do custo. Os resultados para cada um dos métodos meta-heurísticos implementados serão apresentados a seguir. Foi utilizado o *software* Matlab versão 2013b para implementação dos algoritmos. Os algoritmos foram executados em uma máquina com um processador *Intel Core i7* (4^a geração), 1.8 GHz e 8 GB de memória RAM.

A fim de verificar o funcionamento adequado dos métodos de otimização, repetidas execuções foram realizadas. Considerou-se 50 execuções como uma quantidade suficiente. Os parâmetros de projeto de cada uma das meta-heurísticas são mostrados a seguir.

- *Simulated Annealing*:
  - Temperatura inicial: 30 °C;

  - Número de iterações por temperatura: 50;
  - Razão de resfriamento da temperatura: $\alpha = 0,9$.

- Algoritmo Genético:
  - Tamanho da população: 30;
  - Número de gerações: 200;
  - Método de seleção dos pais: Seleção Uniforme;
  - Método de seleção dos indivíduos sobreviventes: Seleção Uniforme.

- PSO:
  - Tamanho do enxame (número de partículas): 30;
  - Número de iterações: 200;
  - Atualização da velocidade: linear;
  - Topologia: Estrela;
  - Coeficiente de inércia (ω): 0,9;
  - Porcentagem da influência do sucesso individual (φ_1): 70%;
  - Porcentagem da influência do sucesso social (φ_2): 30%.

5.1 Polos dos *kernels* de Volterra simétricos

Como visto, os *kernels* da série de Volterra podem ser simetrizados sem que haja perda de capacidade de representação do modelo. A simetrização faz com que o número de termos a serem estimados seja menor (BRAGA, 2011). Da Tabela 2 à 7, são exibidos os resultados para as diferentes meta-heurísticas, além dos resultados para o caso de polos dos *kernels* da série Volterra simétricos e assimétricos. Os dados encontrados nas tabelas são

- n_1: Número de funções no *kernel* de ordem 1;
- n_2: Número de funções no *kernel* de ordem 2;
- p_1: Polo que parametriza as funções do *kernel* de ordem 1;
- p_2: Polos que parametrizam as funções do *kernel* de ordem 2;
- EQM: Menor erro de estimação dentre as 50 execuções do algoritmo;
- EQM Val: Erro de estimação da validação;
- n^o par: Número de parâmetros estimados;

Tabela 2 – Resultados referentes à meta-heurística *Simulated Annealing* para polos iguais nos *kernels* de ordem 2.

Simulated Annealing								
n_1	n_2	p_1	p_2		EQM	EQM Val	n^o par	tempo (s)
2	2	0,9475	0,7157	0,7157	$28,60 \times 10^{-4}$	$70,68 \times 10^{-4}$	6	1,5483
3	3	0,9683	0,6612	0,6612	$20,46 \times 10^{-4}$	$61,27 \times 10^{-4}$	10	2,3946
6	4	0,9877	0,5996	0,5996	$14,34 \times 10^{-4}$	$57,10 \times 10^{-4}$	17	4,6484
5	5	0,9876	0,745	0,745	$10,49 \times 10^{-4}$	$46,30 \times 10^{-4}$	21	5,4045
3	6	0,9855	0,7488	0,7488	$8,44 \times 10^{-4}$	$42,41 \times 10^{-4}$	25	6,0314
6	6	0,9928	0,7436	0,7436	$8,058 \times 10^{-4}$	$41,35 \times 10^{-4}$	28	7,0885
7	7	0,9979	0,7315	0,7315	$6,02 \times 10^{-4}$	$38,55 \times 10^{-4}$	36	9,927

Fonte: Do autor.

Tabela 3 – Resultados referentes à meta-heurística Algoritmo Genético para polos iguais nos *kernels* de ordem 2.

Algoritmo Genético								
n_1	n_2	p_1	p_2		EQM	EQM Val	n^o par	tempo (s)
2	2	0,9476	0,7151	0,7151	$28,60 \times 10^{-4}$	$70,68 \times 10^{-4}$	6	1,3232
3	3	0,9679	0,6644	0,6644	$20,47 \times 10^{-4}$	$61,20 \times 10^{-4}$	10	1,8795
6	4	0,9874	0,5988	0,5988	$14,34 \times 10^{-4}$	$57,06 \times 10^{-4}$	17	3,5475
5	5	0,9881	0,7437	0,7437	$10,49 \times 10^{-4}$	$46,32 \times 10^{-4}$	21	4,0669
3	6	0,9855	0,7501	0,7501	$8,44 \times 10^{-4}$	$42,41 \times 10^{-4}$	25	4,5444
6	6	0,9927	0,7431	0,7431	$8,06 \times 10^{-4}$	$41,35 \times 10^{-4}$	28	5,4203
7	7	0,9978	0,7325	0,7325	$6,02 \times 10^{-4}$	$38,51 \times 10^{-4}$	36	7,2876

Fonte: Do autor.

Tabela 4 – Resultados referentes à meta-heurística PSO para polos iguais nos *kernels* de ordem 2.

PSO								
n_1	n_2	p_1	p_2		EQM	EQM Val	n^o par	tempo (s)
2	2	0,9475	0,7159	0,7159	$28,60 \times 10^{-4}$	$70,68 \times 10^{-4}$	6	1,059
3	3	0,9684	0,6619	0,6619	$20,46 \times 10^{-4}$	$61,24 \times 10^{-4}$	10	1,6534
6	4	0,9876	0,5996	0,5996	$14,34 \times 10^{-4}$	$57,08 \times 10^{-4}$	17	3,1644
5	5	0,9876	0,7452	0,7452	$10,49 \times 10^{-4}$	$46,30 \times 10^{-4}$	21	3,6899
3	6	0,9854	0,7487	0,7487	$8,44 \times 10^{-4}$	$42,37 \times 10^{-4}$	25	4,1662
6	6	0,9928	0,744	0,744	$8,057 \times 10^{-4}$	$41,34 \times 10^{-4}$	28	4,8554
7	7	0,9979	0,7316	0,7316	$6,02 \times 10^{-4}$	$38,54 \times 10^{-4}$	36	6,9946

Fonte: Do autor.

- tempo: tempo médio (em segundos) gasto para as 50 execuções do algoritmo.

Para verificar se os métodos meta-heurísticos estão bem projetados e avaliar a convergência, foi utilizada a função *errorbar* do Matlab. Essa função foi usada para gerar um gráfico

do desenvolvimento da média e desvio padrão do custo mínimo encontrado a cada iteração das meta-heurísticas. O algoritmo *Simulated Annealing*, por exemplo, foi projetado com uma temperatura inicial de 30 °C e razão de resfriamento $\alpha = 0,9$. Para a temperatura inicial, polos iniciais são inicializados aleatoriamente. O algoritmo foi executado 50 vezes e a média do custo (EQM) foi calculada para todos os valores de temperatura. Dessa forma, espera-se que para valores altos de temperatura a média do custo seja alta e o desvio padrão também (soluções iniciais aleatórias). À medida que a temperatura diminui, a média e o desvio padrão do custo devem diminuir. O que indica a convergência do método. Os gráficos do desenvolvimento da média e 3 desvios padrão para cada meta-heurística para o caso em que $n_1 = n_2 = 6$ são mostrados na Figura 18.

Do ponto de vista do custo computacional, foram utilizadas as funções *tic* e *toc* para medição do tempo de execução das meta-heurísticas, cujos resultados são apresentados nas Tabelas 2 a 7.

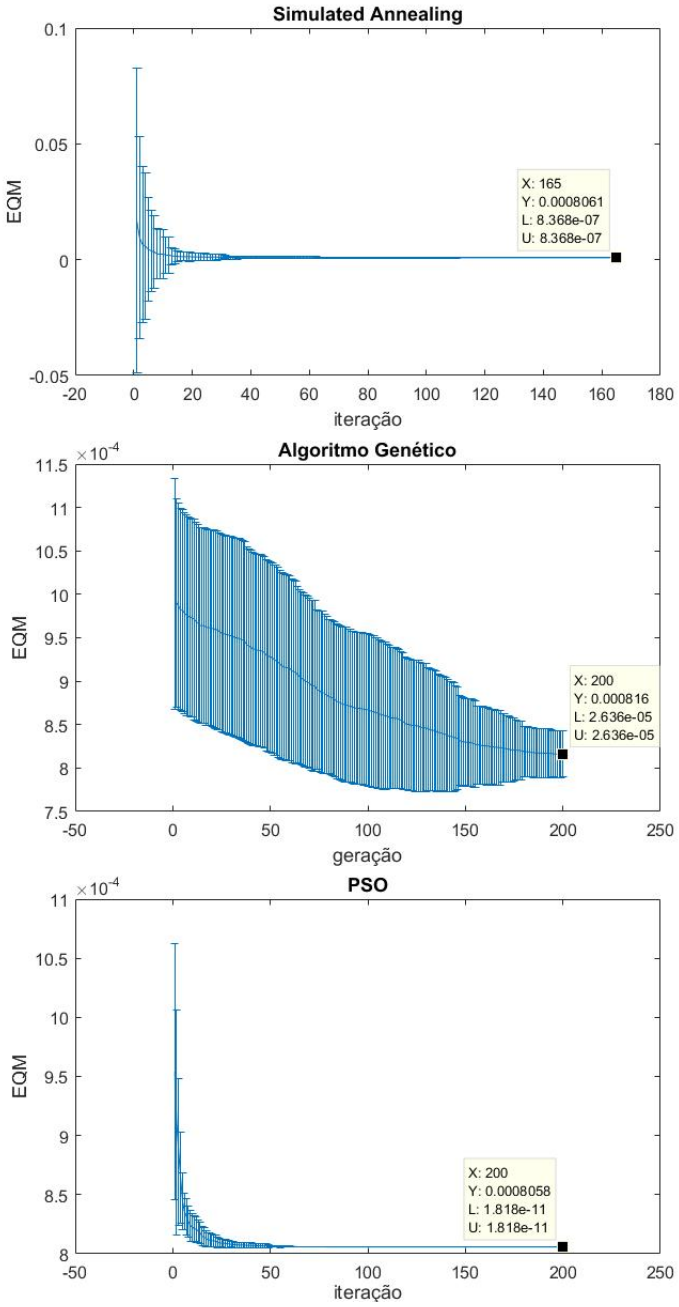
Nos gráficos, pode-se perceber que as meta-heurísticas Algoritmo Genético e PSO apresentaram um custo inicial muito menor que no caso do *Simulated Annealing*. Isso é explicado pelo fato das primeiras se basearem em busca populacional. Ao final da execução dos métodos, o PSO foi quem apresentou o menor custo médio ($EQM = 8,058 \times 10^{-4}$) e menor variação do resultado final. Se realizada a comparação desse valor médio e o mínimo, mostrado na Tabela 4 ($EQM = 8,057 \times 10^{-4}$), nota-se que o método convergiu. Analogamente, a análise pode ser feita para os outros algoritmos e chega-se à conclusão de que os métodos também convergem.

As Figuras 19 e 20 mostram, respectivamente, erro de estimação e saída estimada para o modelo obtido por meio da meta-heurística PSO com $n_1 = n_2 = 6$ e polos iguais nos *kernels* de ordem 2. As Figuras 21 e 22 mostram erro e saída do modelo com os dados de validação.

5.2 Polos dos *kernels* de Volterra assimétricos

A análise feita anteriormente também pode ser realizada para o caso em que os polos dos *kernels* de Volterra são assimétricos. A diferença está na quantidade de parâmetros a serem estimados. Para o caso em que o modelo é representado com $n_1 = n_2 = 6$ funções nos *kernels* de Volterra, a quantidade de parâmetros varia de 28 no caso simétrico para 43 no caso assimétrico. Percebe-se que o EQM melhora ligeiramente para o caso assimétrico.

Figura 18 – Gráfico de desenvolvimento da média e 3 desvios padrão para as meta-heurísticas *Simulated Annealing*, Algoritmo Genético e PSO, respectivamente.



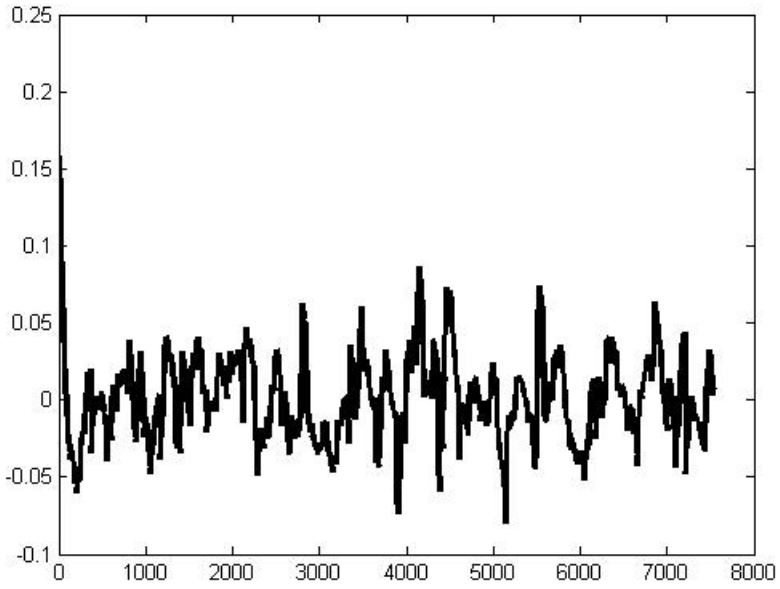
Fonte: Do autor

5.3 NSGA-II

O algoritmo NSGA-II foi implementado e executado com os seguintes parâmetros (aqueles que apresentaram melhores resultados e custo computacional aceitável).

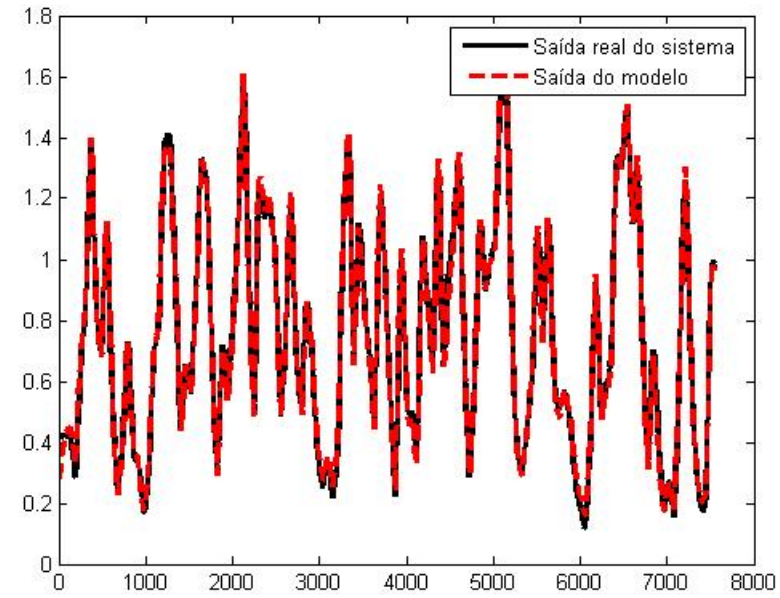
- Tamanho da população: 50;
- Número de gerações: 2000;

Figura 19 – Erro de estimação ($y - y_{modelo}$) para o modelo obtido por PSO com $n_1 = n_2 = 6$ e polos iguais nos *kernels* de ordem 2.



Fonte: Do autor.

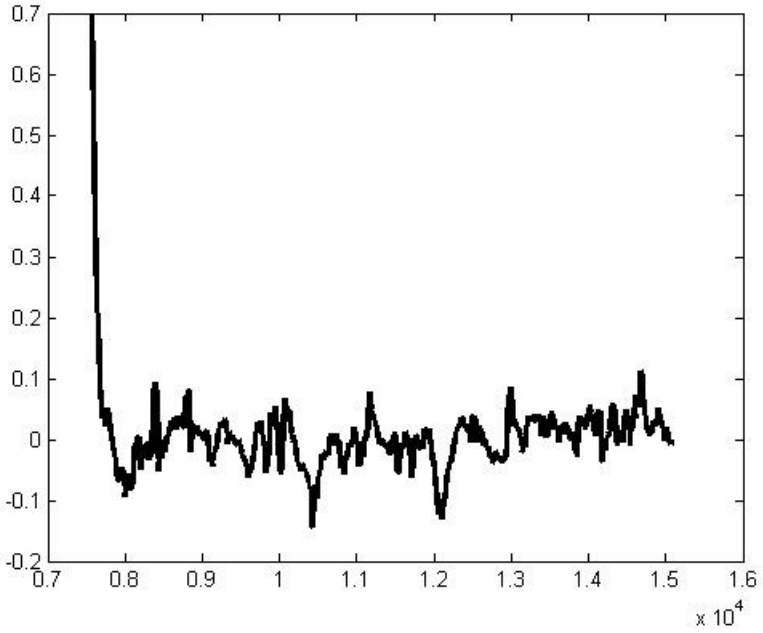
Figura 20 – Saída estimada para o modelo obtido por PSO com $n_1 = n_2 = 6$ e polos iguais nos *kernels* de ordem 2.



Fonte: Do autor.

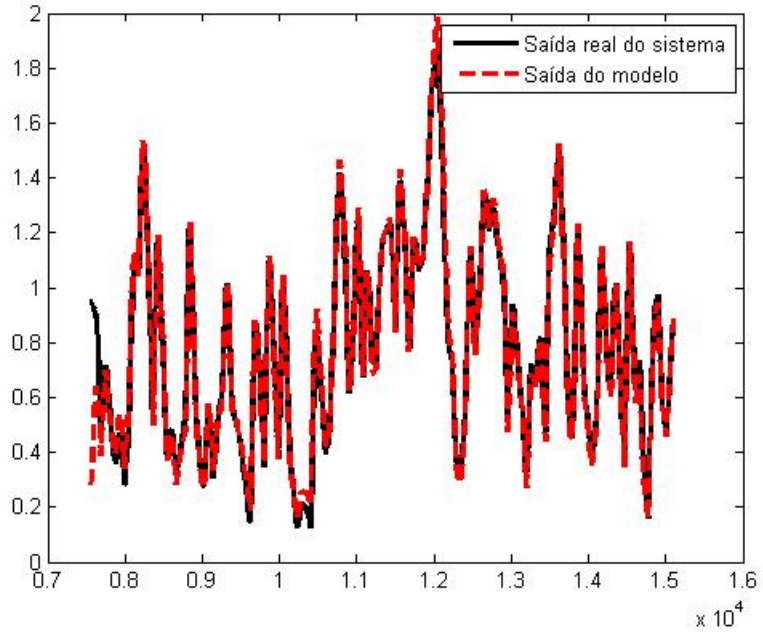
- Método de seleção dos pais: Torneio de 2 elementos;

Figura 21 – Erro de validação para o modelo obtido por PSO com $n_1 = n_2 = 6$ e polos iguais nos *kernels* de ordem 2.



Fonte: Do autor.

Figura 22 – Saída de validação para o modelo obtido por PSO com $n_1 = n_2 = 6$ e polos iguais nos *kernels* de ordem 2.



Fonte: Do autor.

Assim como no caso mono-objetivo, avaliou-se a simetrização dos *kernels* de ordem 2. A Tabela 8 mostra os resultados obtidos para o caso em que é realizada a simetrização. O

Tabela 5 – Resultados referentes à meta-heurística *Simulated Annealing* para polos distintos nos *kernels* de ordem 2.

Simulated Annealing								
n_1	n_2	p_1	p_2		EQM	EQM Val	n^o par	tempo (s)
2	2	0,9486	0,7122	0,7106	$28,43 \times 10^{-4}$	$70,03 \times 10^{-4}$	7	1,7336
3	3	0,993	0,6773	0,6771	$16,19 \times 10^{-4}$	$55,27 \times 10^{-4}$	13	2,7111
6	4	0,9916	0,6812	0,683	$9,75 \times 10^{-4}$	$45,41 \times 10^{-4}$	23	5,6343
5	5	0,9887	0,6835	0,6733	$7,31 \times 10^{-4}$	$40,23 \times 10^{-4}$	31	7,9863
3	6	0,9834	0,6419	0,6170	$6,14 \times 10^{-4}$	$38,66 \times 10^{-4}$	40	10,9786
6	6	0,9853	0,6184	0,6418	$5,75 \times 10^{-4}$	$38,24 \times 10^{-4}$	43	12,2144
7	7	0,9984	0,7187	0,7548	$4,67 \times 10^{-4}$	$38,09 \times 10^{-4}$	57	17,6441

Fonte: Do autor.

Tabela 6 – Resultados referentes ao Algoritmo Genético para polos distintos nos *kernels* de ordem 2.

Algoritmo Genético								
n_1	n_2	p_1	p_2		EQM	EQM Val	n^o par	tempo (s)
2	2	0,9463	0,7066	0,7159	$28,46 \times 10^{-4}$	$70,11 \times 10^{-4}$	7	1,4303
3	3	0,993	0,6865	0,6624	$16,21 \times 10^{-4}$	$55,27 \times 10^{-4}$	13	2,1847
6	4	0,9912	0,6881	0,6807	$9,76 \times 10^{-4}$	$45,37 \times 10^{-4}$	23	4,3373
5	5	0,9881	0,6831	0,6707	$7,32 \times 10^{-4}$	$40,30 \times 10^{-4}$	31	6,3858
3	6	0,9835	0,6427	0,6195	$6,14 \times 10^{-4}$	$38,62 \times 10^{-4}$	40	9,5449
6	6	0,9846	0,642	0,6152	$5,76 \times 10^{-4}$	$38,20 \times 10^{-4}$	43	10,3899
7	7	0,9963	0,7279	0,7534	$4,70 \times 10^{-4}$	$38,79 \times 10^{-4}$	57	15,5355

Fonte: Do autor.

Tabela 7 – Resultados referentes à meta-heurística PSO para polos distintos nos *kernels* de ordem 2.

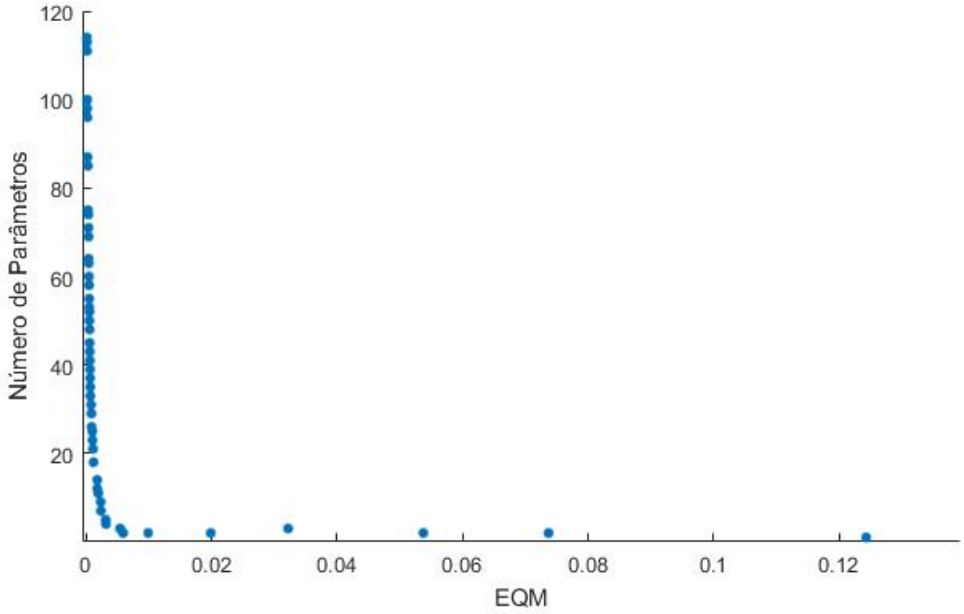
PSO								
n_1	n_2	p_1	p_2		EQM	EQM Val	n^o par	tempo (s)
2	2	0,9483	0,7103	0,7103	$28,43 \times 10^{-4}$	$70,02 \times 10^{-4}$	7	1,18
3	3	0,9927	0,677	0,677	$16,15 \times 10^{-4}$	$55,25 \times 10^{-4}$	13	1,9528
6	4	0,9909	0,684	0,6851	$9,71 \times 10^{-4}$	$45,36 \times 10^{-4}$	23	4,3451
5	5	0,9886	0,6784	0,6854	$7,29 \times 10^{-4}$	$40,17 \times 10^{-4}$	31	5,8417
3	6	0,9832	0,6203	0,6437	$6,11 \times 10^{-4}$	$38,53 \times 10^{-4}$	40	7,8744
6	6	0,9926	0,6155	0,6391	$5,74 \times 10^{-4}$	$38,32 \times 10^{-4}$	43	8,7209
7	7	0,9972	0,735	0,7511	$4,64 \times 10^{-4}$	$38,84 \times 10^{-4}$	57	13,604

Fonte: Do autor.

tempo total de execução é, nesse caso, de 420 segundos (ou 7 minutos). Apesar de parecer ser um tempo muito maior do que aqueles vistos nas Tabelas 2, 3 e 4, o tempo de execução naqueles casos é médio (para cada linha da tabela). Além disso, vale ressaltar que a execução do NSGA-II gerou tabelas com uma quantidade de informação muito maior que aquelas geradas no caso

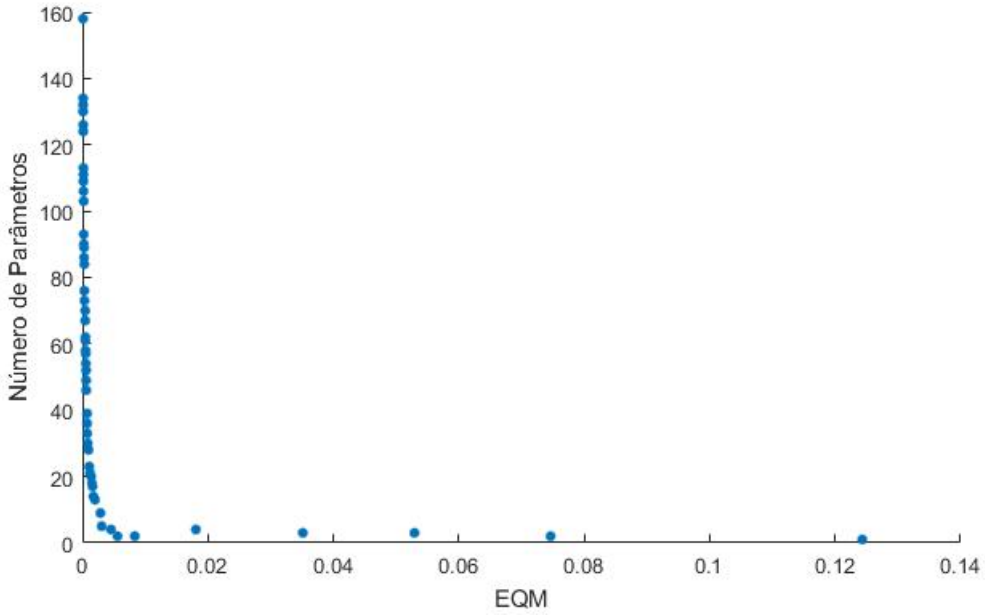
mono-objetivo. A Tabela 9 mostra os resultados obtidos para o caso em que não é realizada a simetrização. Nesse cenário, 553 segundos ($\approx$ 9 minutos) foram necessários para a execução do código.

Figura 23 – Pareto-ótimo para polos iguais nos *kernels* de ordem 2.



Fonte: Do autor.

Figura 24 – Pareto-ótimo para polos distintos nos *kernels* de ordem 2.



Fonte: Do autor.

As Figuras 23 e 24 apresentam as curvas obtidas como resultado da otimização biobjetivo realizada para os casos em que usa-se polos iguais e polos distintos, respectivamente, nos *kernels* de ordem 2. O objetivo da otimização multiobjetivo é fornecer, ao mesmo tempo, o polo ótimo e o número de funções a ser utilizado nas FBOs.

No caso deste trabalho, o EQM na validação é um importante fator na tomada de decisões. Levando isso em conta, o melhor modelo obtido foi o simétrico (polos iguais), com 9 polos em n_1 e 9 polos em n_2 (55 parâmetros), apresentado na Tabela 8 .

Tabela 8 – Resultados referentes ao NSGA-II para polos iguais nos *kernels* de ordem 2.

NSGA-II							
n_1	n_2	p_1	p_2		EQM	EQM Val	n^o par
0	0	-	-	-	0,12434	0,12435	1
0	1	-	0,48109	0,48109	0,053755	0,050079	2
1	0	0,82761	0,80823	0,80823	0,0098762	0,014179	2
2	0	0,37069	-	-	0,032196	0,03344	3
1	1	0,84605	0,86487	0,86487	0,0054032	0,010654	3
0	2	0,76537	0,74496	0,74496	0,003156	0,0070747	4
1	2	0,72885	0,75036	0,75036	0,0031175	0,0071413	5
0	3	0,73094	0,68049	0,68049	0,0023531	0,0059776	7
2	3	0,65134	0,69601	0,69601	0,0023186	0,0061343	9
0	4	0,70083	0,729	0,729	0,0019505	0,005986	11
1	4	0,637	0,61838	0,61838	0,0017707	0,0055611	12
3	4	0,78506	0,71494	0,71494	0,0017618	0,0053093	14
2	5	0,87133	0,774	0,774	0,0011768	0,0045585	18
5	5	0,75988	0,73104	0,73104	0,0010966	0,004451	21
7	5	0,77706	0,70626	0,70626	0,0010079	0,0043268	23
3	6	0,77366	0,70222	0,70222	0,00097291	0,0043487	25
4	6	0,89473	0,74711	0,74711	0,00085254	0,004099	26
7	6	0,81574	0,74429	0,74429	0,00083927	0,0040605	29
2	7	0,85628	0,72072	0,72072	0,00077287	0,0040566	31
4	7	0,87349	0,72865	0,72865	0,00066013	0,0038626	33
6	7	0,80147	0,74469	0,74469	0,00064867	0,0038645	35
8	7	0,80001	0,74037	0,74037	0,00064111	0,0038451	37
2	8	0,88865	0,77361	0,77361	0,00062799	0,0038529	39
4	8	0,84646	0,71504	0,71504	0,00057419	0,0037038	41
6	8	0,80105	0,74548	0,74548	0,00056961	0,0037234	43
8	8	0,73479	0,72405	0,72405	0,00054721	0,003737	45
2	9	0,83682	0,83777	0,83777	0,00054475	0,0044979	48
4	9	0,88491	0,7553	0,7553	0,00051254	0,0037288	50
6	9	0,78079	0,77349	0,77349	0,00052423	0,0037095	52
7	9	0,78667	0,73309	0,73309	0,00048298	0,0037066	53
9	9	0,74684	0,73656	0,73656	0,00048068	0,0036947	55
2	10	0,80942	0,83219	0,83219	0,00045344	0,0046296	58
4	10	0,89996	0,79071	0,79071	0,00043302	0,0038992	60
7	10	0,80826	0,78671	0,78671	0,00041561	0,0038817	63
8	10	0,88825	0,88171	0,88171	0,00039298	0,010045	64
2	11	0,81834	0,82791	0,82791	0,00038803	0,004694	69
4	11	0,93594	0,91492	0,91492	0,00036574	0,15564	71
7	11	0,82415	0,78987	0,78987	0,00035888	0,0040181	74
8	11	0,88619	0,86871	0,86871	0,00029861	0,0084929	75
6	12	0,82021	0,8564	0,8564	0,00026625	0,0060706	85
8	12	0,88875	0,86411	0,86411	0,00021664	0,0088451	87
4	13	0,91149	0,92478	0,92478	0,00020595	17,712	96
6	13	0,82644	0,92333	0,92333	0,00018719	14,417	98
8	13	0,8917	0,85187	0,85187	0,00017399	0,0092914	100
5	14	0,95225	0,93804	0,93804	0,00015934	14520	111
7	14	0,9674	0,9156	0,9156	0,00013295	227,01	113
8	14	0,83298	0,9093	0,9093	0,00012589	38,119	114

Tabela 9 – Resultados referentes ao NSGA-II para polos distintos nos *kernels* de ordem 2.

NSGA-II							
n_1	n_2	p_1	p_2		EQM	EQM Val	n^o par
0	0	-	-	-	0.12434	0.12435	1
0	1	-	0.83641	0.80367	0.0083525	0.0099307	2
1	1	0.61365	0.66496	0.60964	0.035145	0.031341	3
2	1	0.76185	0.80058	0.7905	0.0045637	0.010124	4
0	2	-	0.72595	0.73185	0.0030685	0.0070493	5
4	2	0.6716	0.66276	0.69326	0.0028678	0.0071549	9
3	3	0.64323	0.61489	0.73993	0.0020193	0.0060494	13
4	3	0.73997	0.72297	0.72104	0.0017609	0.0053924	14
0	4	-	0.7956	0.7736	0.0016116	0.005775	17
1	4	0.73644	0.7422	0.72094	0.0015332	0.0056132	18
3	4	0.68766	0.735	0.71691	0.0013793	0.0051996	20
4	4	0.85342	0.79001	0.77715	0.0012097	0.0045493	21
6	4	0.74938	0.72595	0.69819	0.0010902	0.0044804	23
2	5	0.84387	0.78074	0.74742	0.00097481	0.004379	28
4	5	0.83468	0.75801	0.73486	0.0008431	0.0040492	30
7	5	0.8021	0.69554	0.72424	0.00077042	0.0039378	33
10	5	0.74417	0.70679	0.69006	0.00076364	0.0039324	36
2	6	0.82762	0.74581	0.766	0.00074874	0.0042102	39
9	6	0.76063	0.69012	0.67151	0.00060136	0.0037331	46
12	6	0.83334	0.77525	0.76125	0.00059127	0.0040723	49
2	7	0.82588	0.72896	0.75378	0.00058564	0.0040976	52
4	7	0.79543	0.76414	0.72245	0.00054859	0.0038779	54
7	7	0.7573	0.78112	0.72471	0.00053431	0.0037967	57
8	7	0.78634	0.71745	0.76358	0.00050148	0.0037627	58
11	7	0.71447	0.70623	0.74981	0.00049285	0.0037827	61
12	7	0.75397	0.75385	0.72889	0.00048662	0.00379	62
2	8	0.79953	0.88203	0.85196	0.00045423	0.0052044	67
5	8	0.8299	0.85361	0.87696	0.00045256	0.0057146	70
8	8	0.96775	0.88832	0.8515	0.00034259	0.016724	73
11	8	0.95059	0.86245	0.88297	0.00030554	0.014049	76
2	9	0.82488	0.82462	0.89167	0.00030448	0.0053543	84
4	9	0.91799	0.81888	0.89852	0.00026837	0.0068116	86
7	9	0.93011	0.86988	0.80896	0.00026495	0.0086242	89
8	9	0.86635	0.81634	0.88534	0.00023634	0.006703	90
11	9	0.87403	0.84609	0.81109	0.00021217	0.0073439	93
2	10	0.82548	0.86159	0.90184	0.00020527	0.013417	103
5	10	0.89265	0.90445	0.84257	0.00018326	0.013943	106
8	10	0.81661	0.78765	0.86844	0.00017186	0.007144	109
10	10	0.87313	0.80795	0.9146	0.00016911	0.020383	111
12	10	0.87916	0.81127	0.86663	0.00015702	0.008215	113
2	11	0.78909	0.87885	0.94048	0.00012879	1.2605	124
4	11	0.91966	0.86871	0.93176	0.00012812	0.83778	126
8	11	0.86793	0.86368	0.93262	0.00010909	0.18498	130
10	11	0.90979	0.92267	0.8726	0.00010707	0.56719	132
12	11	0.8962	0.90777	0.87287	0.00010505	1.5017	134
13	12	0.87944	0.879	0.82217	9.0399e-05	0.38878	158

6 Conclusão

A modelagem de sistemas não lineares por meio da representação FBO-Volterra é uma técnica que se mostra satisfatória, principalmente pelo fato dessas representações terem a característica de serem do tipo"malha aberta". Dessa forma, o erro da saída não é realimentado à entrada. Outra vantagem é que as funções de bases ortonormais são capazes de reduzir o número de parâmetros a serem estimados na identificação. Um importante procedimento no projeto do modelo FBO-Volterra é a seleção adequada dos polos das bases das funções ortonormais. Escolher o polo adequado significa reduzir o número de termos das bases ortonormais.

Os polos encontrados por otimização meta-heurística são reais. Caracterizando a base de Laguerre, que se mostrou uma importante ferramenta na identificação de sistemas. Os resultados apresentados e discutidos no Capítulo 4 confirmam essa afirmação.

Com relação ao desempenho das técnicas de otimização aplicadas ao problema, pode-se concluir que todas alcançaram resultados interessantes quando avaliamos o objetivo principal do trabalho, que é a identificação de sistemas. Se a análise for direcionada às técnicas de otimização mono-objetivo, é notório o desempenho ótimo da meta-heurística PSO. Tanto do ponto de vista de minimização do EQM quanto do custo computacional, tal método foi superior aos demais. Por outro lado, ao realizar uma análise geral de todas as técnicas empregadas nesse trabalho, a otimização multiobjetivo é, sem dúvidas, a ferramenta mais poderosa. Isso pode ser dito, principalmente, porque o algoritmo NSGA-II obteve uma diversidade muito grande de resultados, deixando para o engenheiro/pesquisador uma gama maior de opções de projeto, sendo capaz de determinar o polo ótimo e o número de funções a ser utilizado nas FBOs.

6.1 Trabalhos Futuros

A partir da monografia realizada, pode-se sugerir as ideias abaixo descritas.

Uma vez que o modelo FBO-Volterra utilizando Laguerre se mostraram eficientes na descrição do sistema, uma sugestão é encontrar um novo sistema em que seja necessária a utilização de polos complexos que parametrizem as FBO.

Outra sugestão é utilizar um algoritmo PSO multiobjetivo como técnica metaheurística de otimização. Tal sugestão vem do fato do algoritmo PSO ter apresentado melhores resultados quando tratava-se da otimização mono-objetivo.

Além disso, é intuitivo pensar na utilização da técnica de identificação aqui apresentada como primeiro passo para implementação de alguma técnica de controle aplicada a um sistema real.

Uma outra ideia seria utilizar modelos *fuzzy* e compará-los ao modelo FBO-Volterra.

Bibliografia

AGUIRRE, L. A. *Introdução à Identificação de Sistemas: técnicas lineares e não-lineares aplicadas a sistemas reais.* [S.l.]: Editora UFMG, 2007. 1, 3, 4, 5, 6, 25, 26

ANDRÉ, N. M. D. S. Particle swarm optimization. 2007. 19

ANSARI, N.; HOU, E. *Computational intelligence for optimization*. [S.l.]: Springer Science & Business Media, 2012. 11, 13

BONYADI, M. R.; MICHALEWICZ, Z. Particle swarm optimization for single objective continuous space problems: a review. *Evolutionary computation*, MIT Press, 2016. 2, 20, 21

BOUSSAÏD, I.; LEPAGNOT, J.; SIARRY, P. A survey on optimization metaheuristics. *Information Sciences*, Elsevier, v. 237, p. 82–117, 2013. 12, 16, 17, 19, 20, 21

BRAGA, M. *Modelos de Volterra: Identificação Não Paramétrica e Robusta Utilizando Funções Ortonormais de Kautz e Generalizadas*. Dissertação (Mestrado) — Universidade de Campinas, 2011. 1, 2, 7, 8, 9, 27

BRAGA, M. F. et al. Optimization of volterra models with asymmetrical kernels based on generalized orthonormal functions. In: IEEE. *Control & Automation (MED), 2011 19th Mediterranean Conference on*. [S.l.], 2011. p. 1052–1058. 1

CAMPELLO, R.; OLIVEIRA, G.; AMARAL, W. Identificação e controle de processos via desevolvimento em séries ortonormais. parte a: Identificação. *Revista Controle & Automação*, 2007. 1, 6, 7, 8, 9

CAMPELLO, R. J. G. B. *Arquiteturas e Metodologias para Modelagem e Controle de Sistemas Complexos Utilizando Ferramentas Clássicas e Modernas*. Tese (Doutorado) — Universidade Estadual de Campinas, 2002. 6

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, IEEE, v. 6, n. 2, p. 182–197, 2002. 22, 23, 24

DUMONT, G. A.; FU, Y. Non-linear adaptive control via laguerre expansion of volterra kernels. *International Journal of Adaptive Control and Signal Processing*, Wiley Online Library, v. 7, n. 5, p. 367–382, 1993. 8

EYKHOFF, P. System identification, 197. *JW Arrowsmith Ltd, Bristol, Great Britain*, 1974. 7

GARCIA, C. *Modelagem e Simulação de Processos Industriais e de Sistemas Eletromecânicos Vol. 1*. [S.l.]: Edusp, 2005. 1

HAUPT, R. L.; HAUPT, S. E. *Practical genetic algorithms*. [S.l.]: John Wiley & Sons, 2004. 16, 17

JOBERG, J. et al. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, v. 31, n. 12, p. 1691–1724, 1995. 6

KIRKPATRICK, S. et al. Optimization by simulated annealing. *science*, Washington, v. 220, n. 4598, p. 671–680, 1983. 14

MALDONADO, Y.; CASTILLO, O.; MELIN, P. Particle swarm optimization of interval type-2 fuzzy systems for fpga applications. *Applied Soft Computing*, Elsevier, v. 13, n. 1, p. 496–508, 2013. 18, 20

METROPOLIS, N. et al. Equation of state calculations by fast computing machines. *The journal of chemical physics*, AIP, v. 21, n. 6, p. 1087–1092, 1953. 13

NELLES, O. Nonlinear system identification. *Measurement Science and Technology*, v. 13, n. 4, p. 646, 2002. 1, 2, 4, 6, 11

OREA, A. C. et al. Modelo termodinâmico para o aquecimento não-linear, a laser, e suas aplicações ao processamento de materiais. Campinas, SP, 1994. 3

OROSKI, E. *IDENTIFICAÇÃO DE SISTEMAS NÃO LINEARES UTILIZANDO MODELOS NARX, FUNÇÕES ORTONORMAIS E OTIMIZAÇÃO HEURÍSTICA*. Tese (Doutorado) — Universidade de Brasília, 2015. 7

PANTUZA, G. *Métodos de otimização multiobjetivo e de simulação aplicados ao problema de planejamento operacional de lavra em minas a céu aberto*. Tese (Doutorado) — Dissertação de mestrado, Programa de Pós-Graduação em Engenharia Mineral-PPGEM da Universidade Federal de Ouro Preto, Ouro Preto, 2011. 2, 22, 23

ROSA, A. da; AMARAL, W. C.; CAMPELLO, R. Desenvolvimento de modelos de volterra usando funções de kautz e sua aplicação à modelagem de um sistema de levitação magnética. In: *Anais do Congresso Brasileiro de Automática*. [S.l.: s.n.], 2006. p. 274–279. 2

RUGH, W. J. *Nonlinear system theory*. [S.l.]: Johns Hopkins University Press Baltimore, 1981. 7

SOUZA, M. J. F. *Inteligência computacional para otimizaçao*. 2011. Acesso em 16/03/2017. Disponível em: <http://www.decom.ufop.br/prof/marcone/InteligenciaComputacional/InteligenciaComputacional.pdf>. 2, 12, 14, 15, 16, 17, 18

SRINIVAS, N.; DEB, K. *Multiobjective optimization using nondominated sorting in genetic algorithms*. 221–248 p. Tese (Doutorado), 1994. 23

WANG, Z.; BOVIK, A. C. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, IEEE, v. 26, n. 1, p. 98–117, 2009. 26

WESTWICK, R. E. K. D. T. *Identification of Nonlinear Physiological Systems (IEEE Press Series on Biomedical Engineering)*. 1. ed. [S.l.]: Wiley-IEEE Press, 2003. 1