sid.inpe.br/mtc-m19/2013/03.11.19.44-TDI

DESENVOLVIMENTO E AVALIAÇÃO DE ABORDAGENS DE OTIMIZAÇÃO MULTIDISCIPLINAR E SUA APLICAÇÃO EM PROJETO DE FOGUETES DE SONDAGEM

Alexandre Nogueira Barbosa

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelo Dr. Lamartine Nogueira Frutuoso Guimarães , aprovada em 5 de fevereiro de 2013.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3DMQBH5>

INPE
São José dos Campos
2013

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
Fax: (012) 3208-6919
E-mail: pubtc@sid.inpe.br

CONSELHO DE EDITORAÇÃO E PRESERVAÇÃO DA PRODUÇÃO INTELECTUAL DO INPE (RE/DIR-204):
Presidente:
Marciana Leite Ribeiro - Serviço de Informação e Documentação (SID)
Membros:
Dr. Antonio Fernando Bertachini de Almeida Prado - Coordenação Engenharia e Tecnologia Espacial (ETE)
Drª Inez Staciarini Batista - Coordenação Ciências Espaciais e Atmosféricas (CEA)
Dr. Gerald Jean Francis Banon - Coordenação Observação da Terra (OBT)
Dr. Germano de Souza Kienbaum - Centro de Tecnologias Especiais (CTE)
Dr. Manoel Alonso Gan - Centro de Previsão de Tempo e Estudos Climáticos (CPT)
Drª Maria do Carmo de Andrade Nono - Conselho de Pós-Graduação
Dr. Plínio Carlos Alvalá - Centro de Ciência do Sistema Terrestre (CST)
BIBLIOTECA DIGITAL:
Dr. Gerald Jean Francis Banon - Coordenação de Observação da Terra (OBT)
REVISÃO E NORMALIZAÇÃO DOCUMENTÁRIA:
Marciana Leite Ribeiro - Serviço de Informação e Documentação (SID)
Yolanda Ribeiro da Silva Souza - Serviço de Informação e Documentação (SID)
EDITORAÇÃO ELETRÔNICA:
Vivéca Sant´Ana Lemos - Serviço de Informação e Documentação (SID)

sid.inpe.br/mtc-m19/2013/03.11.19.44-TDI

DESENVOLVIMENTO E AVALIAÇÃO DE ABORDAGENS DE OTIMIZAÇÃO MULTIDISCIPLINAR E SUA APLICAÇÃO EM PROJETO DE FOGUETES DE SONDAGEM

Alexandre Nogueira Barbosa

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelo Dr. Lamartine Nogueira Frutuoso Guimarães , aprovada em 5 de fevereiro de 2013.

URL do documento original:
<http://urlib.net/8JMKD3MGP7W/3DMQBH5>

INPE
São José dos Campos
2013

Dados Internacionais de Catalogação na Publicação (CIP)

B234d

Barbosa, Alexandre Nogueira.
Desenvolvimento e avaliação de abordagens de otimização multidisciplinar e sua aplicação em projeto de foguetes de sondagem / Alexandre Nogueira Barbosa. – São José dos Campos : INPE, 2013.
xxviii + 228 p. ; (sid.inpe.br/mtc-m19/2013/03.11.19.44-TDI)

Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2013.
Orientador : Dr. Lamartine Nogueira Frutuoso Guimarães.

1. otimização multidisciplinar de projeto. 2. algoritmo memético. 3. modelos substitutos. 4. mineração de dados. 5. foguetes de sondagem. . I.Título.

CDU 004.8

Copyright © 2013 do MCT/INPE. Nenhuma parte desta publicação pode ser reproduzida, armazenada em um sistema de recuperação, ou transmitida sob qualquer forma ou por qualquer meio, eletrônico, mecânico, fotográfico, reprográfico, de microfilmagem ou outros, sem a permissão escrita do INPE, com exceção de qualquer material fornecido especificamente com o propósito de ser entrado e executado num sistema computacional, para o uso exclusivo do leitor da obra.

Copyright © 2013 by MCT/INPE. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, microfilming, or otherwise, without written permission from INPE, with the exception of any material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use of the reader of the work.

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de *Doutor(a)* em

Computação Aplicada

Dr. Stephan Stephany

Presidente / INPE / SJCampos - SP

Dr. Lamartine Nogueira Frutuoso Guimarães

Orientador(a) / INPE / SJCampos - SP

Dr. Haroldo Fraga de Campos Velho

Membro da Banca / INPE / São José dos Campos - SP

Dr. Solon Venâncio de Carvalho

Membro da Banca / INPE / SJCampos - SP

Dr. Roberto Luiz Galski

Membro da Banca / INPE / São José dos Campos - SP

Dr. Élcio Jeronimo de Oliveira

Convidado(a) / IAE / São José dos Campos - SP

Dr. João Manoel Losada Moreira

Convidado(a) / UFABC / São Paulo - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

Aluno (a): ***Alexandre Nogueira Barbosa***

São José dos Campos, 05 de Fevereiro de 2013

“Vamos supor que a Terra fosse quente como o Sol, não teria havido nenhum crescimento, ou que fosse fria como a Lua, não teria havido nenhum crescimento. Ela tinha de vir para o centro, onde tinha ambas as coisas em proporções adequadas para crescer. Da mesma forma, um ser humano deve trabalhar de modo que mantenha uma moderação e um equilíbrio, e compreenda que não deve ir aos extremos”.

S.S. Shri Mataji Nirmala Devi
em Sorrento, Itália, 1989

A minha esposa Olga, meus filhos Mikhael e Yasmin e meus pais Felipe e Vera.

AGRADECIMENTOS

Ao Instituto de Aeronáutica e Espaço (IAE) por ter permitido a realização deste trabalho e julgado relevante.

Ao Instituto Nacional de Pesquisas Espaciais (INPE) que, por meio de seus competentes professores, deu-me a base necessária para o meu desenvolvimento como pesquisador.

Ao Dr. Lamartine Nogueira Frutuoso Guimarães pela confiança, pelo estímulo, pelas sucessivas revisões e pela orientação profícua.

Ao Dr. José Demísio Simões da Silva (*in memoriam*) e ao Dr. Rafael Duarte Coelho dos Santos por terem fornecido o auxílio indispensável na fase inicial e de desenvolvimento deste trabalho.

Finalmente, aos meus colegas e familiares pelo incentivo e apoio contínuo.

RESUMO

A otimização multidisciplinar é uma metodologia com promissora aplicabilidade em projetos de engenharia, em particular, no setor aeroespacial, através da qual intensifica-se a sinergia entre as disciplinas consideradas nesse contexto, promovendo ganhos expressivos no desempenho geral dos produtos ao mesmo tempo que torna o processo de seu desenvolvimento mais eficiente. Nesse contexto, os produtos são sistemas complexos, governados pela interação entre diversos fenômenos físicos. Os foguetes de sondagem brasileiros enquadram-se nessa categoria, pois seu projeto requer mais e mais interações com múltiplas disciplinas. Motivado pela busca por versões melhoradas destes projetos, este trabalho teve por objetivo estudar e ampliar abordagens de otimização multidisciplinar. Primeiro, combinou-se o algoritmo genético multiobjetivo não-geracional com o algoritmo multiobjetivo da otimização extrema generalizada, dois métodos inspirados em princípios diferentes, nunca antes combinados, tal que juntos fornecessem resultados melhores do que o emprego de cada um sem o auxílio do outro. Segundo, desenvolveu-se um procedimento para avaliar e comparar a eficiência da rede perceptron de múltiplas camadas e da rede de função de base radial, usadas como modelos substitutos de funções objetivo, quando computacionalmente custosas. Terceiro, desenvolveu-se uma abordagem, baseada nos mapas auto-organizáveis, no método k-médias e no teorema de Bayes, para explorar as soluções de Pareto, quando representadas em espaços multidimensionais, e auxiliar na sumarização dos compromissos entre as variáveis e os objetivos conflitantes. As duas primeiras abordagens foram propostas para reduzir o custo computacional da otimização, enquanto a terceira foi para auxiliar na análise dos resultados. A ideia de propor essas abordagens surgiu da análise do estudo de otimização das empenas do foguete brasileiro VS-40, onde foram considerados os cálculos de aerodinâmica e trajetória na avaliação da função objetivo. Para realizar essa otimização, desenvolveu-se o programa MDO-SONDA (*multidisciplinary design optimization of sounding rockets*). Este executa os programas de aerodinâmica e trajetória, em modo *batch*, calcula as propriedades de massa e inércia do foguete, processa a otimização, e gerencia possíveis falhas na execução desses códigos durante esse processo. Aplicando esta ferramenta, reduziu-se o arrasto causado pelas empenas sem aumentar a probabilidade de efeitos adversos que levassem o foguete a apresentar comportamentos instáveis. Finalmente, traçaram-se os contornos para aprimoramento do MDO-SONDA em trabalhos futuros: a incorporação de programas de mais disciplinas, tais como propulsão e estruturas, a definição de problemas mais complexos envolvendo dois ou mais subsistemas do foguete e a incorporação das abordagens desenvolvidas neste trabalho.

DEVELOPMENT AND EVALUATION OF APPROACHES OF MULTIDISCIPLINARY OPTIMIZATION AND ITS APPLICATION IN SOUNDING ROCKET PROJECTS

ABSTRACT

Multidisciplinary design optimization is a promising methodology with applicability in engineering projects, particularly in the aerospace sector, through which intensifies the synergy between disciplines considered in this context, providing significant gains in the overall performance of products while making its development more efficient. In this context, the products are complex systems, governed by the interaction between different physical phenomena. The Brazilian sounding rockets fall into this category because their design requires more and more interactions with multiple disciplines. Motivated by the pursuit of improved versions of these projects, this work was to study and extend multidisciplinary optimization approaches. First, it was combined the non-generational multi-objective genetic algorithm with the multi-objective generalized extreme optimization algorithm, two methods inspired by different principles, never before combined, such that together provide better results than employing one without the assistance of the other. Second, it was developed a procedure to evaluate and compare the efficiency of the multilayer perceptron network and the radial basis function network, both used as surrogate models of computationally expensive objective functions. Third, it was developed an approach, which was based on self-organizing maps, the k-means method and the Bayes' theorem, to explore Pareto-optimal solutions, when represented in multidimensional spaces, and to assist in summarizing the trade-offs between the variables and conflicting objectives. The first two approaches were proposed to reduce the computational cost of the optimization, while the third approach was to assist in the analysis of results. The idea of proposing these approaches arose from the analysis of the fins optimization of the Brazilian rocket VS-40, where the aerodynamics and trajectory calculations were considered in the evaluation of the objective function. To perform this optimization, it was developed the program MDO-SONDA (multidisciplinary design optimization of sounding rockets). This software executes the aerodynamics and trajectory programs in batch mode, calculates the properties of mass and inertia of the rocket, process the optimization, and manages possible failures in the execution of these programs during this process. Using this tool, the drag caused by the fins was reduced without increasing the chances of adverse effects that could lead the rocket to present unstable behaviors. Finally, the contours for the MDO-SONDA improvement in future works were traced: incorporation of programs of more disciplines, such as propulsion and structures, the definition of more complex problems involving two or more subsystems of the rocket, and the incorporation of the approaches that were developed in this work.

LISTA DE FIGURAS

Pág.

3.1 - Procedimento em cascata da realização dos projetos dos foguetes de sondagem brasileiros. 12
3.2 - Componentes do foguete de sondagem. 14
3.3 - Caracterização do ambiente de microgravidade. 14
3.4 - Correspondência entre os estágios do VS-40 e do VLS-1. 15
3.5 - Perfis da trajetória do VS-40 e do VSB-30, considerando a mesma massa de carga útil e a mesma elevação de lançamento. 16
3.6 - Perfis de voo. (a) Voo suborbital; (b) voo orbital. 17
3.7 - Lançamentos do foguete VS-40. (a) Primeiro lançamento, em 1993, operação Santa Maria; (b) segundo lançamento, em 1998, operação Livramento; (c) terceiro lançamento, em 2012, operação SHEFEX II. Fontes: (a) e (b) Instituto de Aeronáutica e Espaço (IAE); (c) DLR (2012). 18
3.8 - Interação do algoritmo de otimização com a função objetivo. 20
3.9 - Interação da função objetivo com os programas de aerodinâmica e trajetória. 20
3.10 - Fluxo de dados no MDO-SONDA. 22
3.11 - Definição da margem estática. 26
3.12 - Variáveis do problema de otimização. 27
3.13 - Resultado da otimização. 30
3.14 - Arrasto versus área da empena. 30
3.15 - Regime de voo do VS-40 com as empenas originais. 32
4.1 - Conceito de dominância. 37
4.2 - Variantes clássicas do operador de recombinação. (a) Corte em um ponto; (b) corte em dois pontos; (c) recombinação uniforme. 40
4.3 - Esboço da distribuição regular de soluções ao longo da fronteira de Pareto. 41
4.4 - Ponto de criticalidade de uma pilha de grãos. 43
4.5 - Possibilidades de recombinação binária com corte em um ponto, dados dois genitores como exemplos. 49
4.6 - Possibilidades de recombinação binária com corte em dois pontos, dados dois genitores como exemplos. 50
4.7 - Ajuste do operador SBX. 50
4.8 - Espalhamento do operador SBX dentro do espaço normalizado. 51
4.9 - Espalhamento do SBX com soluções inválidas. 51
4.10 - Exemplo de soluções geradas com o operador binário do M-GEO. 53
4.11 - Mapeamento da comutação dos bits realizada pelo operador binário do M-GEO. 54
4.12 - Mutação da variável de projeto. 56
4.13 - Assimetria dinâmica causada pelo operador real imitativo do operador binário do M-GEO. (a) Exemplo 1; (b) exemplo 2, segunda visita ao ponto inicial *C*, em que ocorre a mudança na assimetria das soluções derivadas, em comparação com o Exemplo 1. 56
4.14 - Operador binário versus SBX (*n* igual a 10). 60
4.15 - Operador binário versus SBX (*n* igual a 20). 60
4.16 - Operador binário versus SBX (*n* igual a 30). 61

4.17 - Exemplo de soluções geradas com o ORAD. 63
4.18 - ORAD versus operadores binários do NGMOGA. 64
4.19 - Soluções pesquisadas no domínio da função *G* pelo M-GEO com o operador binário original. 67
4.20 - Soluções de Pareto da função *G* obtidas com o M-GEO usando o operador binário. 67
4.21 - Soluções de Pareto da função *G* obtidas com o M-GEO usando o operador ORAD. 68
4.22 - Uma execução da busca do M-GEO após posicionamento das sementes. 69
4.23 - Posicionamento das sementes do M-GEO com o auxílio do NGMOGA. 70
4.24 - Resultado final com os operadores binários.....71
4.25 - Distribuição das soluções investigadas na fronteira de Pareto.....71
4.26 - Resultado obtido com o NGMOGA.....72
4.27 - Espalhamento causado pelo ORAD. 73
4.28 - Posicionamento das sementes do M-GEO. 74
4.29 - Resultado final com o ORAD. 74
4.30 - Arquitetura do MDO-SONDA com a abordagem memética substituindo o algoritmo de otimização como mostrado na Figura 3.8. 75
5.1 - Método *early stopping* baseado em validação cruzada. Fonte: adaptado de Haykin (1999), p. 238. 83
5.2 - Exemplos de quadrados latinos. (a) 10 amostras; (b) 25 amostras; (c) 50 amostras.87
5.3 - Esquema das células.....89
5.4 - Resultados sobrepostos de 10 execuções da otimização da 1ª função de teste assistida pela MLP e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio. 98
5.5 - Resultados sobrepostos de 10 execuções da otimização da 1ª função de teste assistida pela RBFN e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio. 98
5.6 - Resultados sobrepostos de 10 execuções da otimização da 2ª função de teste assistida pela MLP e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio. 99
5.7 - Resultados sobrepostos de 10 execuções da otimização da 2ª função de teste assistida pela RBFN e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio. 99
5.8 - Resultados sobrepostos de 10 execuções da otimização da 3ª função de teste assistida pela MLP e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-

geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio. 100
5.9 - Resultados sobrepostos de 10 execuções da otimização da 3ª função de teste assistida pela RBFN e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio. 100
5.10 - Comparação da eficiência da MLP e da RBFN como modelos substitutos na otimização assistida das funções de teste. (a) Resultado de η_1 de acordo com a Equação 5.16; (b) resultado médio de η_2, aplicando-se a Equação 5.18, com base em 10 execuções da otimização conforme o passo (h) e considerando 29 células para avaliar a distribuição das soluções ao longo da fronteira de Pareto com base nos pontos gerados no passo (g); (c) resultado médio da eficiência η de cada modelo substituto, dada pela Equação 5.20, com base em 100 execuções do procedimento para determinar a eficiência conforme mencionado no passo (i).101
5.11 - Arquitetura do MDO-SONDA com a abordagem memética e o modelo substituto, no lugar da função objetivo, como apresentado nas Figuras 3.8 e 4.30, respectivamente.....103
6.1 - Complexidade da visualização da fronteira de Pareto. (a) Complexidade baixa; (b) moderada; (c) alta. 105
6.2 - Exemplos de topologia. (a) Topologia retangular; (b) topologia hexagonal. 110
6.3 - Coordenadas do neurônio no reticulado. (a) Topologia retangular; (b) topologia hexagonal. 114
6.4 - Quantização da distribuição das soluções de Pareto no domínio (X) e no contradomínio da função multiobjetivo (F). 120
6.5 - Mapeamento dos perfis de projeto sobre os perfis de desempenho. 120
6.6 - Convergência dos mapas SOM 1 e 2. (a) Conjunto de Pareto; (b) fronteira de Pareto.....123
6.7 - Reticulado dos mapas SOM 1 e 2 após adaptação. (a) Reticulado do SOM 1; (b) reticulado do SOM 2. 124
6.8 - Discretização do SOM 1 e 2.....125
6.9 - Mapeamento dos neurônios do SOM 1 sobre os do SOM 2. 125
6.10 - Neurônios dispersos atribuídos erroneamente a um perfil de projeto. 127
C.1 - Importação dos dados de otimização. 209
C.2 - Visualização de dados por meio de coordenadas paralelas.....209
C.3 - Regressão dos dados através da rede perceptron de múltiplas camadas. 210
C.4 - Identificação dos agrupamentos por meio da visualização dos reticulados em tons de cinza.....210
C.5 - Agrupamento dos neurônios por meio do k-médias. 211
C.6 - Mapeamento dos neurônios do SOM 1 sobre os neurônios do SOM 2 através da rede perceptron de múltiplas camadas. 211
D.1 - Tela de abertura do programa. 213
D.2 - Entrada das definições do problema de otimização de empenas. 213
D.3 - Interface para visualizar o arquivo de entrada do programa *missile datcom* para cada fase de voo. 214
D.4 - Interface para visualizar o arquivo de entrada do programa ROSI. 214

D.5 - Interface para definir as fases de voo. 215
D.6 - Entrada do perfil longitudinal da geometria externa do foguete. 215
D.7 - Entrada do perfil longitudinal das empenas. 216
D.8 - Entrada da seção transversal das empenas. 216
D.9 - Entrada dos dados propulsivos do foguete. 217
D.10 - Entrada das propriedades de massa e inércia do propelente. 217
D.11 - Entrada das propriedades de massa e inércia de componentes do foguete. 218
D.12 - Interface para verificar a interpolação da geometria do foguete, retornada pelo programa *missile datcom*.....218
D.13 - Gráficos dos parâmetros de trajetória calculados pelo programa ROSI. 219
D.14 - Entrada dos parâmetros do NGMOGA com o operador real ORAD.....219
D.15 - Interface para acompanhar e visualizar os resultados da otimização das empenas.220

LISTA DE TABELAS

Pág.

3.1 - Limites das variáveis do problema de otimização. 27
3.2 - Configuração do NGMOGA com o operador ORAD.....28
3.3 - Tempo de avaliação da função objetivo em um computador 3,0 GHz Dual Core.29
3.4 - Dimensões das soluções de empenas para o VS-40.....31
4.1 - Funções de teste. 59
4.2 - Parâmetros do M-GEO.....65
4.3 - Desempenho das versões do M-GEO. 66
4.4 - Análise comparativa.....73
5.1 - Probabilidade de substituição.....80
5.2 - Funções de teste. 91
5.3 - Descritores da complexidade das funções de teste. 92
5.4 - Definições da MLP. 93
5.5 - Definições da RBFN. 93
5.6 - Divisão do número de amostras com e sem o subconjunto de teste. 94
5.7 - Configurações dos modelos substitutos com a melhor capacidade generalização.95
5.8 - Porcentagem média de soluções de Pareto com base em 10 execuções da otimização. 96
5.9 - Configurações do algoritmo genético multiobjetivo não-geracional adotadas no passo (h). 96
5.10 - Valor médio do número de avaliações da função multiobjetivo com base em 10 execuções da otimização. 97
5.11 - Tempo médio de processamento decorrido no passo (h) com base em 10 execuções da otimização em uma máquina 2.0 GHz Quad Core. 97
6.1 - Estimativa da probabilidade dos perfis de desempenho, observando-se cada perfil de projeto.....126
6.2 - Estimativa da probabilidade a priori dos perfis de projeto. 126
6.3 - Estimativa da probabilidade dos perfis de projeto, observando-se os perfis de desempenho.....126

LISTA DE ABREVIATURAS E SIGLAS

ACO	–	*ant colony optimization*
ACO-GA	–	*ant colony optimization in combination with genetic algorithm*
AM	–	algoritmos meméticos
ARB	–	*Architecture Review Board*
BLX	–	crossover blend
CLA	–	Centro de Lançamento de Alcântara
DLR	–	Centro Aeroespacial Alemão
GA	–	*genetic algorithm*
GEO	–	*generalized extreme optimization*
HGSAA	–	*hybrid genetic simulated annealing algorithm*
IA	–	inteligência artificial
IAE	–	Instituto de Aeronáutica e Espaço
LHS	–	*latin hypercube sampling*
MDO	–	*multidisciplinary design optimization*
MDO-SONDA	–	*multidisciplinary design optimization of sounding rockets*
M-GEO	–	*multi-objective generalized extreme optimization*
MLP	–	*multilayer perceptron*
NGMOGA	–	*non-generational multi-objective genetic algorithm*
ORAD	–	operador real de assimetria dinâmica
PCF	–	*Parallel Computing Forum*
PSO	–	*particle swarm optimization*
RBFN	–	*radial basis function network*
RSA	–	*restarted simulated annealing*
RSAPSO	–	*hybrid restarted simulated annealing particle swarm optimization*
RSM	–	*response surface methods*
SA	–	*simulated annealing*
SARA	–	Satélite de Reentrada Atmosférica
SBX	–	*simulated binary crossover*
SOC	–	*self-organized criticality*

SOM	–	*self-organizing maps*
SRE	–	*Space Capsule Recovery Experiment*
SVR	–	*support vector regression*
VLS-1	–	Veículo Lançador de Satélite

LISTA DE SÍMBOLOS

$\#$	–	cardinalidade de conjunto
$\|\|\cdot\|\|_{\Sigma}$	–	distância de Mahalanobis
$\|\|\cdot\|\|$	–	distância euclidiana
$\lceil \cdot \rceil$	–	função de arredondamento teto
$\leftarrow$	–	denota atribuição de valor
$\varphi(\cdot)$	–	função de ativação dos neurônios da MLP
$\nprec$	–	não pior do que
Φ	–	matriz de interpolação
$\succ$	–	melhor do que
C_D	–	coeficiente de arrasto
$C_{N\alpha}$	–	derivada do coeficiente da força normal com o ângulo de ataque
$C_{M\alpha}$	–	derivada do coeficiente do momento em arfagem com o ângulo de ataque
C_{Mq}	–	derivada do coeficiente do momento em arfagem com a taxa de arfagem
C_{lp}	–	derivada do coeficiente do momento em rolamento com a taxa de rolamento
$C_{l\delta}$	–	derivada do coeficiente do momento em rolamento com a deflexão da empena
C_l	–	coeficiente do momento em rolamento
X_{cp}	–	centro de pressão
X_{cg}	–	centro de gravidade
e	–	margem estática

SUMÁRIO

Pág.

1 INTRODUÇÃO 1
1.1 Otimização multidisciplinar: um novo paradigma na engenharia dos foguetes de sondagem brasileiros 1
1.2 Motivação 1
1.3 Objetivo e escopo 2
1.4 Originalidade 2
1.5 Organização do documento 3

2 REVISÃO BIBLIOGRÁFICA 5
2.1 Desafios da otimização multidisciplinar 5
2.2 O estado da arte 6
2.2.1 Considerações 6
2.2.2 Abordagens que tornam a otimização mais eficiente 7
2.2.3 Exploração das soluções de Pareto em espaços multidimensionais 9
2.3 Otimização de foguetes de sondagem 10

3 OTIMIZAÇÃO DAS EMPENAS DO FOGUETE VS-40 11
3.1 Introdução 11
3.2 Conceitos e definições 13
3.2.1 Foguetes de sondagem 13
3.2.2 Ambiente de microgravidade 14
3.3 Fatos que motivaram o estudo do foguete VS-40 15
3.4 Programa MDO-SONDA 18
3.4.1 Aspectos gerais 18
3.4.2 Arquitetura 19
3.4.3 Dados de entrada 23
3.4.4 Interfaces de saída 23
3.4.5 Algoritmo de otimização 24
3.4.6 Procedimento de otimização 24
3.5 Definição do problema de otimização 25
3.6 Discussão dos resultados 28

4 COOPERAÇÃO ENTRE MÉTODOS DE OTIMIZAÇÃO 33
4.1 Introdução 33
4.2 Fundamentação teórica 36
4.2.1 Problema de otimização multiobjetivo 36
4.2.2 Algoritmo genético multiobjetivo não-geracional 37
4.2.3 Algoritmo multiobjetivo do método da otimização extrema generalizada 43
4.2.4 Operador real simulated binary crossover 47
4.2.5 Mutação uniforme de Michalewicz 52
4.3 Abordagem memética baseada no NGMOGA e no M-GEO 53
4.3.1 Operador real de assimetria dinâmica 53

4.3.2 Estratégia de otimização....57
4.4 Estudo de caso....58
4.4.1 Descrição....58
4.4.2 Comparação do NGMOGA com operadores reais e binários....59
4.4.3 Comparação do M-GEO com o ORAD e o operador binário....65
4.4.4 Comparação da abordagem memética com o ORAD e operadores binários....68

5 AVALIAÇÃO DA EFICIÊNCIA DE MODELOS SUBSTITUTOS....77
5.1 Introdução....77
5.2 Fundamentação teórica....79
5.2.1 Otimização Assistida por Modelos Substitutos....79
5.2.2 Rede perceptron de múltiplas camadas....81
5.2.3 Rede de função de base radial....83
5.2.4 Latin hypercube sampling....85
5.3 Procedimento para avaliar a eficiência de modelos substitutos....87
5.4 Estudo de caso....90

6 ANÁLISE EXPLORATÓRIA DE SOLUÇÕES DE PARETO....105
6.1 Introdução....105
6.2 Fundamentação teórica....106
6.2.1 Coordenadas paralelas....106
6.2.2 K-Médias....106
6.2.3 Mapas auto-organizáveis....107
6.2.4 Teorema de Bayes....117
6.3 Sumarização de resultados de otimização....118
6.4 Estudo de caso....121
6.4.1 Descrição....121
6.4.2 Discussão dos resultados....123

7 CONCLUSÃO....129
7.1 Resumo dos resultados principais....129
7.2 Recomendações para trabalhos futuros....131
7.2.1 Abordagem memética....131
7.2.2 Modelos substitutos....131
7.2.3 Sumarização de resultados de otimização....132
7.2.4 Otimização de foguetes de sondagem....132

REFERÊNCIAS BIBLIOGRÁFICAS....135

APÊNDICE A – CÓDIGOS EM MATLAB DO CAPÍTULO 4....143
A.1 Rotina principal....143
A.2 Rotina do algoritmo genético multiobjetivo não-geracional....143
A.3 Rotina da versão multiobjetivo da otimização extrema generalizada....150
A.4 Rotinas de teste da abordagem memética....155
A.4.1 Configuração do algoritmo genético multiobjetivo não-geracional....155
A.4.2 Configuração da versão multiobjetivo da otimização extrema generalizada....156
A.4.3 Funções de teste....157

APÊNDICE B – CÓDIGOS EM MATLAB DO CAPÍTULO 5.....159
B.1 Rotina principal.....159
B.2 Rotinas da perceptron de múltiplas camadas.....177
B.2.1 Definição da arquitetura.....177
B.2.2 Treinamento da rede neural.....177
B.2.3 Ativação da rede neural.....180
B.3 Rotinas da rede de função de base radial.....181
B.3.1 Treinamento da rede neural.....181
B.3.2 Determinação dos pesos sinápticos.....183
B.3.3 Ativação da rede neural.....183
B.3.4 Resposta do neurônio de saída.....184
B.4 Rotinas compartilhadas entre as redes neurais.....185
B.4.1 Normalização da entrada da rede neural.....185
B.4.2 Desnormalização da resposta da rede neural.....185
B.5 Rotinas de otimização.....186
B.5.1 Versão binária do algoritmo genético multiobjetivo não-geracional.....186
B.5.2 Otimização assistida pela rede neural.....194
B.5.3 Função objetivo.....200
B.6 Rotinas de avaliação.....202
B.6.1 Estimativa da eficiência da rede neural.....202
B.6.2 Avaliação da dominância entre conjuntos de soluções de Pareto.....204
B.6.3 Obtenção das soluções não-dominadas de um conjunto de soluções.....205
B.7 Rotina do método *latin hypercube sampling*.....206
B.8 Funções de teste.....207

APÊNDICE C – TELAS DA SUMARIZAÇÃO DE RESULTADOS.....209

APÊNDICE D – TELAS DO PROGRAMA MDO-SONDA.....213

APÊNDICE E – LEVANTAMENTO HISTÓRICO.....221
E.1 Introdução.....221
E.2 Origem dos métodos de otimização.....221
E.3 Ano de 1947: um marco na história da otimização.....222
E.4 Evolução dos algoritmos e das máquinas computacionais.....223
E.5 Surgimento dos métodos estocásticos.....223
E.6 Otimização de múltiplos objetivos conflitantes.....224
E.7 Desenvolvimento do conceito de inteligência artificial.....225
E.8 Métodos baseados em processos biológicos.....226
E.9 Evolução da computação paralela.....227
E.10 Otimização de sistemas complexos de engenharia.....228

1 INTRODUÇÃO

1.1 Otimização multidisciplinar: um novo paradigma na engenharia dos foguetes de sondagem brasileiros

A otimização multidisciplinar é uma metodologia com promissora aplicabilidade em projetos de engenharia, em particular, no setor aeroespacial, através da qual intensifica-se a sinergia entre as disciplinas consideradas nesse contexto, promovendo ganhos expressivos no desempenho geral dos produtos ao mesmo tempo que torna o processo de seu desenvolvimento mais eficiente. Normalmente, esses produtos são sistemas complexos, governados pela interação entre diversos fenômenos físicos. Os foguetes de sondagem brasileiros enquadram-se na categoria de sistemas complexos, pois seu projeto requer mais e mais interações entre múltiplas disciplinas. Cabe ao Instituto de Aeronáutica e Espaço (IAE), situado em São José dos Campos, a missão de pesquisar, desenvolver, inovar, realizar operações de lançamento e prestar serviços tecnológicos em sistemas espaciais, dentre os quais elencam-se os foguetes de sondagem, com o objetivo de fortalecer o poder aeroespacial brasileiro. Visando a essa área, abordagens de otimização multidisciplinar foram desenvolvidas neste trabalho.

1.2 Motivação

A busca por versões melhoradas dos projetos de foguetes de sondagem brasileiros é uma atividade de pesquisa relevante para o IAE. O foguete de sondagem brasileiro VS-40, desenvolvido na década de 1990, não havia sido ainda otimizado. A interação entre as disciplinas envolvidas no projeto do VS-40 ocorreu em cascata, onde as disciplinas que atuaram inicialmente na concepção do foguete estabeleceram restrições para aquelas que atuaram mais tarde, concebendo um foguete sem levar em conta os compromissos existentes entre estas disciplinas. A consequência da interação em cascata, caracterizada pela baixa sinergia entre as disciplinas, foi uma versão subótima do VS-40 em termos globais de projeto. A baixa sinergia entre as disciplinas de aerodinâmica e trajetória no desenvolvimento do VS-40 foi o problema identificado por este autor que motivou a realização deste trabalho.

1.3 Objetivo e escopo

Este trabalho teve por objetivo estudar e ampliar o conhecimento sobre abordagens de otimização multidisciplinar. Limitou-se o escopo dessas abordagens ao seguinte:

a) emprego dos paradigmas da inteligência computacional; e

b) aplicação em problemas contínuos.

Adotou-se, como procedimento de trabalho, a não utilização de pacotes ou códigos prontos dos algoritmos de otimização e das redes neurais utilizados, de modo que este autor pudesse apreciar em detalhes as descrições desses métodos e reproduzir suas próprias versões, adaptadas a esta pesquisa. Além disso, restringiu-se a otimização multidisciplinar do VS-40 a duas disciplinas, aerodinâmica e trajetória, em que os programas computacionais dessas disciplinas foram tratados como caixas-pretas[1].

1.4 Originalidade

O produto deste trabalho resume-se ao seguinte:

1) uma abordagem memética baseada na cooperação entre o algoritmo genético multiobjetivo não-geracional e o algoritmo multiobjetivo da otimização extrema generalizada, dois métodos inspirados em princípios diferentes, que juntos provêem resultados melhores do que o emprego de cada um sem o auxílio do outro;

2) uma abordagem para avaliar e comparar a eficiência da rede perceptron de múltiplas camadas e da rede de função de base radial, usadas na prática como modelos substitutos de funções objetivo, quando tais funções são consideradas computacionalmente custosas; e

3) uma abordagem de sumarização dos resultados da otimização, baseada nos mapas auto-organizáveis, no método k-médias e no teorema de Bayes,

[1] Não se precisa saber como o programa funciona, apenas o formato de suas entradas e saídas.

para explorar as soluções de Pareto em espaços multidimensionais e auxiliar na identificação dos compromissos entre as variáveis e os objetivos conflitantes de projeto.

Adicionalmente, este trabalho apresenta o seguinte:

a) implementações em Matlab dos algoritmos de otimização multiobjetivo, originalmente desenvolvidos em FORTRAN;

b) acoplamento de dois programas computacionais, um de aerodinâmica e outro de trajetória, nunca antes acoplados para aplicação em problemas de otimização de foguetes de sondagem; e

c) desenvolvimento do programa MDO-SONDA[2] (*multidisciplinary design optimization of sounding rockets*).

1.5 Organização do documento

Este trabalho está organizado em sete capítulos, incluindo esta introdução e a conclusão.

O Capítulo 2 destina-se à revisão bibliográfica, onde apresentam-se os desafios da otimização multidisciplinar de projeto[3] e as abordagens mais recentes para superá-los.

Depois da revisão bibliográfica, apresenta-se, no Capítulo 3, o estudo de otimização multidisciplinar e multiobjetivo das empenas do VS-40. A análise desse estudo motivou a proposta das abordagens de otimização multidisciplinar como tema principal desta pesquisa de doutorado, objetivando sua futura aplicação em problemas mais complexos de otimização de foguetes de sondagem. Essas abordagens são, então, apresentadas nos Capítulos 4, 5 e 6, e representam o produto final desta pesquisa. Esses capítulos foram

[2] O MDO-SONDA, desenvolvido em C++, interage com os programas computacionais de aerodinâmica e trajetória, e realiza o cômputo das propriedades de massa e inércia do foguete, a otimização, o gerenciamento das falhas na execução desses programas, a geração automática de gráficos de trajetória, a apresentação dos resultados da otimização por meio de gráficos, dentre outras funções.

[3] Se há interesse em aprofundar-se em fatos históricos, o Apêndice E apresenta um resumo da confluência dos métodos de otimização, das abordagens de inteligência computacional e das arquiteturas de computação paralela com outras estratégias de otimização; confluência, esta, que contribuiu com a origem da metodologia da otimização multidisciplinar de projeto.

estruturados de modo que o leitor pudesse identificar mais facilmente as contribuições deste trabalho. Em cada um, há quatro seções. A primeira é a introdução do capítulo. Na segunda, comentam-se os métodos empregados. Na terceira, descreve-se a abordagem desenvolvida. Na quarta, discute-se um estudo realizado com funções matemáticas, retiradas da literatura, a fim de demonstrar e testar a nova abordagem.

Os Capítulos 3, 4, 5 e 6 apresentam estudos distintos, mas que estão associados da seguinte forma. No Capítulo 4, propõe-se um algoritmo memético, baseado na cooperação do algoritmo genético multiobjetivo não-geracional (NGMOGA) com o algoritmo multiobjetivo da otimização extrema generalizada (M-GEO), e um operador real para esses algoritmos, desenvolvido por este autor, denominado ORAD. A versão do NGMOGA com o ORAD foi incorporada no programa MDO-SONDA, apresentado antes, no Capítulo 3, enquanto a versão em Matlab do NGMOGA, implementada por este autor, foi usada na abordagem apresentada no Capítulo 5. Neste, compara-se a eficiência da rede perceptron de múltiplas camadas (MLP) e da rede de função de base radial (RBFN), usadas como modelos substitutos. A MLP foi empregada como modelo substituto na abordagem apresentada no Capítulo 6.

Nos Capítulos 4 e 5, apresentam-se abordagens para reduzir o número de avaliações da função objetivo durante a otimização. Enquanto, no Capítulo 6, apresenta-se a proposta de sumarização dos resultados da otimização, baseada nos mapas auto-organizáveis (SOM), no método k-médias e no teorema de Bayes. Finalmente, no Capítulo 7, ressaltam-se as principais conclusões de cada capítulo e as recomendações para desenvolvimento em trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

2.1 Desafios da otimização multidisciplinar

Desde a década de 1990, tem-se desenvolvido abordagens para lidar com os desafios da otimização multidisciplinar de projeto (MDO). Ao incorporar duas ou mais disciplinas na otimização de um sistema, o MDO acaba por aumentar a complexidade do problema. O que torna o MDO ainda mais desafiador é o fato de os modelos computacionais empregados pelas disciplinas, tratados como caixas-pretas, serem, às vezes, códigos legados[4], processados em sistemas operacionais diferentes, pouco documentados, e sujeitos a travamentos em situações que não são conhecidas, a priori. Porém, os maiores obstáculos são, frequentemente, o longo tempo de execução dos modelos e a falta de gradientes analíticos.

Os modelos computacionais incorporados são, normalmente, de alta-fidelidade. No entanto, podem não ser robustos, por exemplo, devido à ocorrência de falhas não previstas quando recebem determinados valores de entrada. Também, ao manipular variáveis em mais de uma disciplina, a quantidade total dessas variáveis e a de funções custo podem aumentar significativamente e, consequentemente, dificultar a análise do conjunto e da fronteira de Pareto, respectivamente, para explorar os conflitos existentes entre os objetivos do projeto. Portanto, no MDO, depara-se com três desafios:

1) o de lidar com as dificuldades de considerar mais e mais disciplinas na avaliação do produto, incorporando seus modelos computacionais;

2) o de tornar a otimização mais eficiente no sentido de reduzir o número de avaliações da função objetivo, cujo custo computacional tenderá a ser cada vez maior à medida que mais modelos são incorporados; e

[4] Trata-se de um código que provém do passado, é complexo, de difícil manutenção e, pelo custo para testá-lo com atualizações, é mantido sem modificações porque funciona de forma segura em situações conhecidas.

3) o de explorar soluções de Pareto em espaços multidimensionais, por considerar-se mais variáveis e objetivos de projeto à medida que se objetiva resolver problemas de otimização mais complexos.

O primeiro desafio é de caráter prático, pois requer conhecimento inerente a cada disciplina e habilidade em programação. Como os modelos são, normalmente, tratados como caixas-pretas, a programação acaba mais relacionada ao desenvolvimento de interfaces entre esses modelos. Este foi o desafio enfrentado por este autor ao desenvolver o programa MDO-SONDA.

Os dois últimos desafios são de caráter metodológico, pois requerem o estudo e desenvolvimento de métodos específicos para lidar com eles. Inclusive, são mais gerais, pois também podem estar presentes em outras áreas de pesquisa além do MDO. Por exemplo, o método de Monte Carlo requer centenas ou milhares de avaliações de um ou mais modelos.

Se esses modelos são computacionalmente caros, pode ser necessário reduzir, de algum modo, suas avaliações. Em mineração de dados, por exemplo, onde os problemas recaem sobre a análise de informações contendo inúmeros atributos, a exploração de dados em espaços multidimensionais também é um desafio.

2.2 O estado da arte

2.2.1 Considerações

Este trabalho está em grande parte associado à matemática aplicada e computacional. Essa área é caracterizada essencialmente pelo estudo dos métodos computacionais e sua aplicação em outras áreas do conhecimento. Este trabalho é um exemplo disto, onde métodos de inteligência computacional são aplicados na área de projetos de foguetes de sondagem.

O método computacional é, portanto, o principal objeto de estudo da matemática aplicada e computacional. Assim, de modo geral, há quatro formas de contribuir nesse contexto: desenvolvendo novos métodos, aprimorando os que existem, combinando-os e

comparando-os. Se este estudo contribui com abordagens melhores em eficiência, simplicidade e robustez, ele é considerado relevante.

2.2.2 Abordagens que tornam a otimização mais eficiente

Dentre as abordagens atuais que têm por objetivo tornar a otimização mais eficiente, pode-se citar as seguintes:

a) combinações de múltiplos métodos de otimização, denominadas algoritmos meméticos (AM), ou híbridos (NERI et al., 2012); e

b) técnicas de otimização assistida por modelos substitutos (FORRESTER et al., 2008).

Com respeito à primeira abordagem, os estudos mais recentes têm por objetivo a hibridização de métodos, de modo que, uma vez reunidos, cooperando ou competindo entre eles, são mais eficientes do que sem a interação com os outros métodos. Os seguintes estudos são exemplos dessa abordagem:

a) *hybrid genetic simulated annealing algorithm* (HGSAA), algoritmo híbrido baseado no algoritmo genético (*genetic algorithm* – GA) e no método *simulated annealing* (SA) (MOUSSI et al., 2012);

b) *hybrid restarted simulated annealing particle swarm optimization* (RSAPSO), método que combina a capacidade de busca global do algoritmo *particle swarm optimization* (PSO) e a de busca local do algoritmo *restarted simulated annealing* (RSA) (ZHANG; WU, 2012); e

c) *ant colony optimization in combination with genetic algorithm* (ACO-GA), método que melhora o desempenho do algoritmo *ant colony optimization* (ACO) com o auxílio do GA (AKPINAR et al., 2013).

Na linha de desenvolvimento de algoritmos híbridos, o que este trabalho propõe é uma abordagem memética baseada na cooperação entre o algoritmo genético multiobjetivo não-geracional (*non-generational multi-objective genetic algorithm* – NGMOGA),

proposto por Borges (1999), e o algoritmo multiobjetivo da otimização extrema generalizada (*multi-objective generalized extreme optimization* – M-GEO), proposto por Galski (2006). O primeiro, configurado para busca global, enquanto o segundo, para busca local, a fim de alcançar com essa abordagem um desempenho melhor do que o alcançado com o uso isolado desses algoritmos.

Quanto à segunda abordagem, o estado da arte consiste de técnicas que se destacam pela simplicidade da representação algébrica e computacional dos modelos substitutos, pelo baixo custo para construí-los e executá-los e pela sua capacidade de aproximar-se, satisfatoriamente, de uma ampla variedade de funções objetivo, das mais simples às mais complexas, com o mínimo de ajuste necessário em seus parâmetros de configuração.

As abordagens de modelos substitutos mais conhecidas são: polinômios, *Kriging* e redes neurais. Os modelos baseados em polinômios, também chamados de *response surface models*, não são considerados adequados à maioria das aplicações em engenharia, pois não superam, satisfatoriamente, a maldição da dimensionalidade (*curse of dimensionality*) dessas aplicações, a qual caracteriza-se pela quantidade exponencial de amostras do modelo original, necessárias à construção do modelo substituto, devido à elevada quantidade de variáveis de projeto na modelagem dos problemas de engenharia.

Por outro lado, modelos baseados em *Kriging*, método da geoestatística, oferecem mais recursos à otimização de sistemas complexos do que os polinômios. No entanto, pode haver vantagens em usar polinômios em certas aplicações, por exemplo, quando a modelagem analítica de distribuições de probabilidade revela-se de forma intratável por meio de outras abordagens (FORRESTER et al., 2008, p. 75).

Dentre as redes neurais, as mais conhecidas são: o perceptron de múltiplas camadas (*multilayer perceptron* – MLP) e a rede de função de base radial (*radial basis function network* - RBFN). A RBFN tem sido bastante apreciada em discussões sobre modelos substitutos em função de características que se destacam em relação a outros modelos. Essa rede permite, por exemplo, adequar sua quantidade de parâmetros. Além disso, é algebricamente simples e fácil de ser implementada. Uma abordagem menos

comentada, porém mais recente, é a regressão baseada em vetores suporte (*support vector regression* – SVR) (FORRESTER et al., 2008, p. 76).

Na linha de desenvolvimento da estratégia de otimização com o auxílio de modelos substitutos, o que este trabalho propõe é uma abordagem para comparar a eficiência de modelos substitutos considerando características do algoritmo de otimização. Essa comparação é importante, pois a escolha de um modelo substituto deve basear-se nesse tipo de análise. No Capítulo 5, compara-se a eficiência da MLP e da RBFN. Acrescenta-se, no entanto, que essa abordagem pode ser aplicada com o objetivo de comparar outros tipos de modelos substitutos.

2.2.3 Exploração das soluções de Pareto em espaços multidimensionais

Em paralelo, têm-se, também, as abordagens cujo objetivo é explorar soluções de Pareto definidas em espaços multidimensionais. Explorar dados significa essencialmente reconhecer, no conjunto desses dados, a formação de padrões, ou seja, significa encontrar agrupamentos nesse conjunto. Em se tratando de dados definidos em espaços multidimensionais e presentes em grande quantidade, a visualização de agrupamentos requer o uso de certas técnicas que, atualmente, estão associadas à área do conhecimento denominada mineração de dados, a qual trata do processo de descoberta de padrões em grandes conjuntos de dados. Em mineração de dados, o método de agrupamento mais simples é o k-médias, proposto em 1957 por Stuart Lloyd, embora não tenha sido publicado até 1982 (FRIEDMAN et al., 2009). Mais tarde, em 1990, surgiram os mapas auto-organizáveis (*self-organizing maps* – SOM), criados por Teuvo Kohonen, também usados com o propósito de identificar agrupamentos de dados (KOHONEN, 1990).

Os dois métodos usam abordagens diferentes. Ao contrário do k-médias, o SOM não requer que um número de partições seja especificado a priori. Isso está implícito no SOM. Seus resultados, expostos normalmente em um reticulado 2-D, são examinados para que se possa determinar que coleções de suas unidades neurais indicam, claramente, a formação de agrupamentos (SOUKUP; DAVIDSON, 2002, p. 234-235).

Atualmente, quando em certas aplicações a visualização da fronteira de Pareto mostra-se como uma tarefa difícil, estratégias baseadas em SOM são usadas com o objetivo de auxiliar a análise dos compromissos entre os objetivos conflitantes de projeto (OBAYASHI; SASAKI, 2003).

O que este trabalho oferece de novo, em estratégias baseadas em SOM, é a inferência estatística, fundamentada no teorema de Bayes, entre agrupamentos de soluções de Pareto identificados em dois espaços distintos, o domínio e o contradomínio da função objetivo, onde ao menos um desses espaços é multidimensional. Com essa abordagem, é possível identificar mais facilmente os tipos de solução de Pareto, definidos com base nos valores médios das variáveis em cada agrupamento no domínio da função objetivo, que atendem às possíveis formas de conciliar os objetivos conflitantes em jogo, definidas de acordo com os agrupamentos identificados no contradomínio.

2.3 Otimização de foguetes de sondagem

Constatou-se que há poucas publicações sobre a aplicação do MDO em foguetes de sondagem ou em artefatos similares a estes. Encontraram-se apenas duas referências. A primeira, Roshanian e Keshavarz (2006), propõe a otimização do foguete iraniano RX-250-LPN, envolvendo três disciplinas: trajetória, propulsão e aerodinâmica. Esses autores aplicaram o método *response surface method* (RSM) como modelo substituto da função objetivo, cujo tempo de execução era elevado. Eles obtiveram um ganho de 25% na altitude alcançada por esse foguete. A segunda publicação, Tarhan (2000), apresenta o uso do algoritmo de otimização GADO (RASHEED, 1998), que, segundo o autor, é capaz de reduzir o tempo de execução da otimização. Tarhan se propôs a otimizar a aerodinâmica de um míssil, calculando a trajetória e usando o programa *missile datcom*, também usado neste trabalho, a fim de calcular os coeficientes aerodinâmicos. Como resultado, ele reduziu o tempo e o esforço empreendidos na fase preliminar do projeto do míssil. No próximo capítulo, apresenta-se um estudo similar, de autoria deste autor – a otimização do foguete VS-40 – cuja realização motivou a proposta das abordagens apresentadas nos capítulos subsequentes, com o objetivo de aplicá-las no futuro.

3 OTIMIZAÇÃO DAS EMPENAS DO FOGUETE VS-40

3.1 Introdução

No final da década de 90, dentre os foguetes de sondagem brasileiros, o VS-40 foi apresentado como aquele que oferece as melhores condições para experimentos em ambiente de microgravidade (RIBEIRO, 1999). O foguete VS-40 é um engenho complexo, cuja dinâmica de voo é governada por vários fenômenos físicos complexos e distintos, e que interagem uns com os outros. Esses fenômenos são estudados pelas disciplinas de propulsão, aerodinâmica, estruturas e trajetória. Sistemas complexos como este exigem o equilíbrio entre essas disciplinas e também entre objetivos relacionados aos aspectos de segurança, operacionalidade, desempenho, confiabilidade e custo (ROWELL; KORTE, 2003).

Novos paradigmas aplicados à engenharia de sistemas complexos têm permitido identificar soluções viáveis de forma mais rápida e explorar a sinergia entre as disciplinas envolvidas nesses projetos (ROWELL; KORTE, 2003). No entanto, identificou-se que não havia estudos sob essa ótica com o objetivo de melhorar o projeto do foguete VS-40. As interações entre as disciplinas de projeto desse foguete ocorreram em cascata, onde aquelas que atuam no começo estabelecem restrições para as que seguem mais tarde, levando a um produto final sem ter explorado os compromissos existentes entre os objetivos do projeto. A pouca interação entre as disciplinas de aerodinâmica e trajetória motivou a realização deste trabalho.

No modelo em cascata, os especialistas esforçam-se para melhorar os objetivos e o atendimento às restrições em termos das variáveis que tenham a ver com sua disciplina, deixando de lado efeitos colaterais para que os especialistas das disciplinas seguintes absorvam em detrimento do desempenho geral do foguete.

A Figura 3.1 ilustra a problemática do modelo em cascata, onde as restrições aumentam à medida que as disciplinas atuam, e a possibilidade de alterar definições nas disciplinas anteriores para favorecer o desempenho geral do foguete torna-se cada vez menor. A consequência natural desse procedimento é um projeto subótimo do ponto de vista

global, promovido pela baixa sinergia entre as disciplinas de projeto. Aplica-se, nesses casos, a otimização multidisciplinar de projeto (MDO), pois é a metodologia indicada para explorar a sinergia entre as disciplinas, promovendo o ganho global no desempenho do produto e diminuindo o tempo de seu desenvolvimento (FLOUDAS; PARDALOS, 2009).

Figura 3.1 - Procedimento em cascata da realização dos projetos dos foguetes de sondagem brasileiros.

Apesar de o VS-40 ser um projeto já concluído há mais de duas décadas, acredita-se que alguns de seus subsistemas ainda possam ser modificados com o objetivo de promover ganhos em seu desempenho. O VS-40 promete as melhores condições para experimentos de microgravidade. Recentemente, o VS-40 foi modificado pelo Centro Aeroespacial Alemão (*Deutsches Zentrum für Luft- und Raumfahrt* – DLR) com o objetivo de ter a estabilidade necessária para uma missão específica, uma vez que não foi originalmente concebido para transportar uma carga útil com pequenas aletas expostas (TURNER et al., 2009). Esse fato indica que seus subsistemas podem ser alterados a fim de melhorar sua estabilidade e seu desempenho. O VS-40 ainda não foi beneficiado com as abordagens mais recentes da engenharia de sistemas complexos, tendo, talvez, subsistemas que possam ser melhorados para aumentar seu desempenho nos próximos lançamentos em território brasileiro levando a plataforma SARA Suborbital, desenvolvida no Brasil, para suporte a experimentos em ambiente de microgravidade. Surgiu, então, a ideia de introduzir o MDO no contexto dos foguetes de sondagem brasileiros, realizando este estudo para o VS-40. Como estudo de caso,

propôs-se a optimização da geometria das empenas do VS-40, a fim de melhorar seu desempenho, reduzindo o arrasto provocado por elas, mas sem aumentar a possibilidade de efeitos adversos que poderiam levar a comportamentos instáveis. Para realizar a otimização de foguetes de sondagem, desenvolveu-se o programa MDO-SONDA[5] (*Multidisciplinary Design Optimization of Sounding Rockets*) como parte desta pesquisa de doutorado (BARBOSA, A. N.; GUIMARÃES, 2012b).

Este capítulo está organizado da seguinte forma. Primeiro, conceituam-se os foguetes de sondagem e definem-se as condições para realização de experimentos em ambiente de microgravidade. Em seguida, discorre-se, em detalhes, sobre os fatos que motivam a escolha do VS-40 para demonstrar a aplicação da otimização multidisciplinar de projeto no contexto dos foguetes de sondagem brasileiros. Segundo, apresentam-se os principais aspectos do MDO-SONDA: arquitetura, dados de entrada, interfaces de saída, algoritmo de otimização, e procedimento de otimização. Por último, discute-se o estudo de caso do VS-40, onde o problema de otimização é definido, as configurações do algoritmo de otimização são apresentadas e os resultados da otimização são comentados.

3.2 Conceitos e definições

3.2.1 Foguetes de sondagem

Os foguetes de sondagem, tais como o VS-40 e o VSB-30, são caracterizados pela sua aplicação e seu perfil de voo. De acordo com Montenbruck et al. (2001), essa categoria é constituída de motores a propelente sólido e transporta uma carga útil com instrumentos a fim de realizar medições e experimentos durante o voo parabólico (Figuras 3.2 e 3.3). Assim, o termo sondagem significa “realizar medições”. Comparado ao VSB-30, o foguete VS-40, com dois estágios (Figura 3.2), pode expor a carga útil por mais tempo ao ambiente de microgravidade, beneficiando a realização de experimentos nesse ambiente.

[5] As telas do programa MDO-SONDA são apresentadas no Apêndice D.

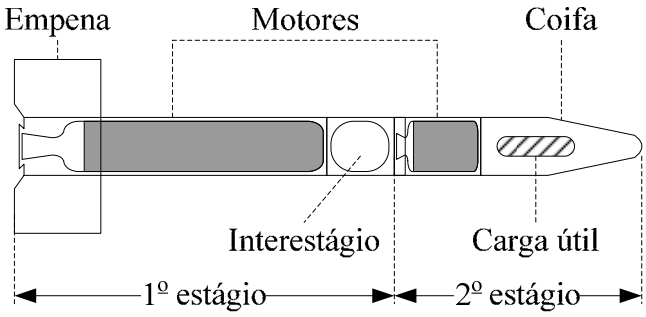


Figura 3.2 - Componentes do foguete de sondagem.

3.2.2 Ambiente de microgravidade

O ambiente de microgravidade é caracterizado pela condição em que a carga útil é submetida a uma força G menor do que 10 μg, por mover-se em queda livre, onde não há outras forças senão a própria gravidade atuando sobre ela (LA NEVE; CORRÊA, 2001). A condição de microgravidade é alcançada quando a força de empuxo torna-se nula, após o fim de queima do último estágio do foguete, e quando a carga útil está fora da atmosfera. A linha de Kármán, definida a 100 km de altitude acima da superfície do mar, é normalmente usada como referência em experimentos de microgravidade para indicar a fronteira entre a atmosfera e o espaço exterior, a partir do qual a atmosfera torna-se tão rarefeita que a força de arrasto é considerada desprezível (Figura 3.3).

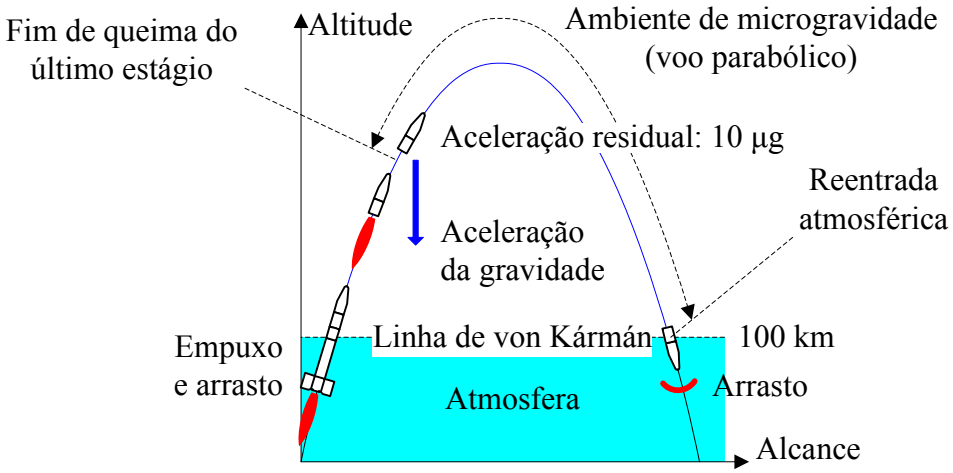


Figura 3.3 - Caracterização do ambiente de microgravidade.

3.3 Fatos que motivaram o estudo do foguete VS-40

O VS-40 foi projetado originalmente para realizar a qualificação em voo do motor S44 (PEREIRA; MORAES, 2000). O S44 constitui o 4º estágio do Veículo Lançador de Satélite (VLS-1) (Figura 3.4). Apesar de o VS-40 prover mais tempo de microgravidade do que o VSB-30, este ainda está a frente em quantidade de lançamentos, pois o VSB-30 já possui uma carga útil preparada para suporte a experimentos embarcados e, além disso, atende aos requisitos da missão com menos custo, pois a recuperação de sua carga útil é menos dispendiosa do que a do VS-40. O impacto em mar da carga útil transportada pelo VS-40 ocorre aproximadamente cinco vezes mais distante da linha costeira do que o impacto do VSB-30, exigindo, portanto, mais autonomia dos meios de resgate para alcançar sua carga útil (Figura 3.5).

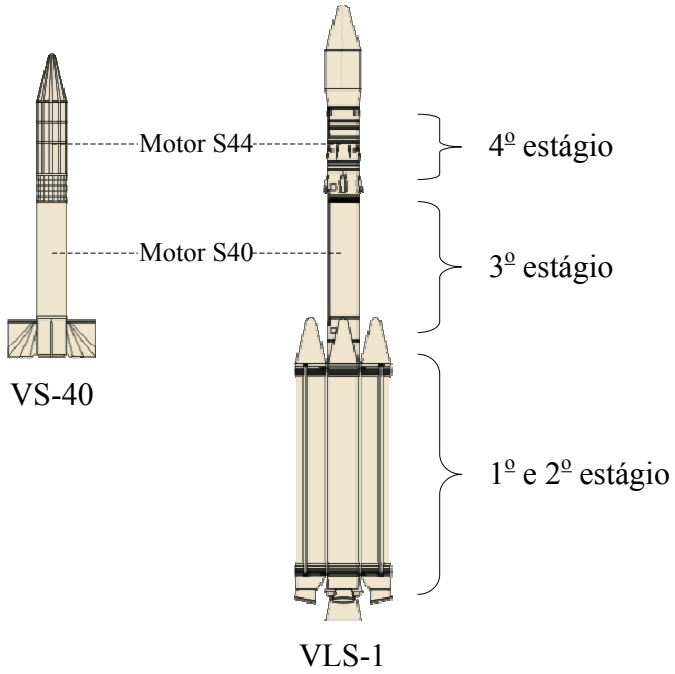


Figura 3.4 - Correspondência entre os estágios do VS-40 e do VLS-1.

De 2004 a 2010, realizaram-se com sucesso dez operações de lançamento do VSB-30, dentre as quais, três em território brasileiro (GARCIA et al., 2011). Por outro lado, apenas três lançamentos do VS-40 ocorreram até o momento, dois em território nacional, o primeiro, em 1993, operação Santa Maria (Figura 3.7a), e o segundo, em

1998, operação Livramento (Figura 3.7b), ambos no Centro de Lançamento de Alcântara (CLA), no Maranhão (IAE, 2012). Em 22 de junho de 2012, uma versão modificada do VS-40, denominada VS-40M, transportando a carga útil alemã SHEFEX II (*Sharp Edge Flight Experiment*) (WEIHS et al., 2008a), foi lançada com sucesso no centro de lançamento de Andøya, no norte da Noruega (DLR, 2012), tornando-se o primeiro foguete VS-40 a ser lançado no exterior (Figura 3.7c).

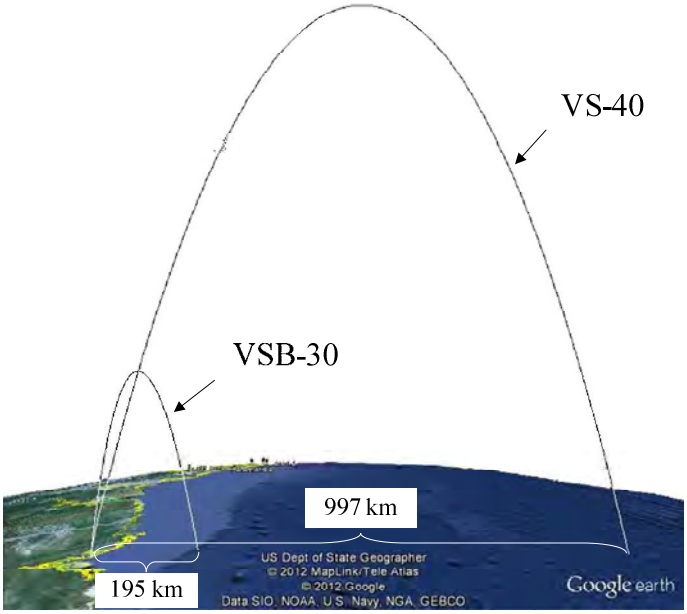


Figura 3.5 - Perfis da trajetória[6] do VS-40 e do VSB-30, considerando a mesma massa de carga útil e a mesma elevação de lançamento.

Em 1997, o projeto de uma plataforma orbital recuperável, denominada SARA (Satélite de Reentrada Atmosférica), foi proposto para atender a experimentos em ambiente de microgravidade (MORAES; PILCHOWSKI, 1997). Em comparação com o voo suborbital (Figura 3.6a), que provia apenas alguns minutos de exposição em ambiente de microgravidade, o voo orbital do SARA poderia prover em torno de 10 dias (Figura 3.6b). A cápsula indiana SRE (*Space Capsule Recovery Experiment*), lançada pela primeira vez em 2007, é bastante similar à plataforma SARA (REDDY, 2007). Outro

[6] Os perfis de trajetória apresentados na Figura 3.5 foram calculados por este autor com o auxílio do programa ROSI (ZIEGLTRUM, 1984) e visualizados no programa Google Earth (GOOGLE, 2011).

exemplo interessante é o experimento alemão *REX-Free Flyer*, derivado da tecnologia do SHEFEX, que também enquadra-se na categoria das plataformas orbitais reutilizáveis para suporte a experimentos em condições de microgravidade (WEIHS et al., 2008b). Como parte do projeto SARA, desenvolveu-se a plataforma SARA Suborbital para ser lançada pelo foguete VS-40 (LA NEVE; CORRÊA, 2001). Quando essa plataforma estiver operacional, a probabilidade de o VS-40 ser escolhido para experimentos de microgravidade será maior do que a do VSB-30, se as condições para resgate da carga útil não forem impeditivas.

Nota-se que a geometria das empenas do VS-40M (Figura 3.7c) é diferente dos dois foguetes anteriores (Figuras 3.7a e 3.7b). As novas empenas são maiores do que as originais, e foram concebidas e construídas exclusivamente pelo DLR para atender ao lançamento do SHEFEX II devido à necessidade de compensar os efeitos aerodinâmicos provocados pelas pequenas aletas localizadas na carga útil (Figura 3.7c) (WEIHS et al., 2008a). Não se encontrou nenhuma publicação sobre a forma como as empenas adotadas no VS-40M foram determinadas pelo DLR. Acredita-se que suas proporções baseiam-se em geometrias conhecidas, como as das empenas do VSB-30, mas em escala maior. Esses fatos motivaram este autor a estudar o VS-40 e a propor a otimização de suas empenas com a finalidade de introduzir a metodologia do MDO no contexto dos foguetes de sondagem brasileiros.

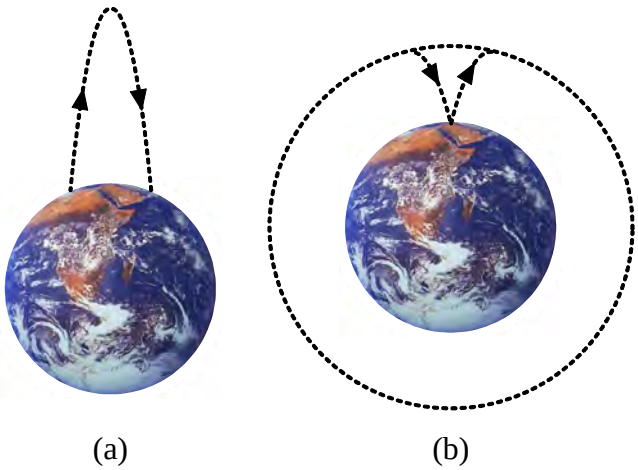


Figura 3.6 - Perfis de voo. (a) Voo suborbital; (b) voo orbital.

(a) (b) (c)

Figura 3.7 - Lançamentos do foguete VS-40. (a) Primeiro lançamento, em 1993, operação Santa Maria; (b) segundo lançamento, em 1998, operação Livramento; (c) terceiro lançamento, em 2012, operação SHEFEX II. Fontes: (a) e (b) Instituto de Aeronáutica e Espaço (IAE); (c) DLR (2012).

3.4 Programa MDO-SONDA

3.4.1 Aspectos gerais

O programa MDO-SONDA foi concebido para explorar a sinergia entre as disciplinas envolvidas no projeto de foguetes de sondagem não guiados[7] com o objetivo de otimizá-los. Dentre essas disciplinas, as que usam modelos de engenharia baseados na física são as seguintes: propulsão, aerodinâmica, estruturas e trajetória.

A versão corrente do MDO-SONDA interage, em modo *batch*[8], com dois programas de alta-fidelidade de simulação, um de aerodinâmica e outro de trajetória. Assim, ele pode explorar a sinergia entre estas disciplinas, tornando-se, portanto, uma ferramenta de

[7] Os foguetes de sondagem brasileiros não são dotados de sistema de controle.
[8] O modo *batch* significa executar um programa até sua conclusão sem intervir manualmente.

otimização multidisciplinar. Além disso, pode ser usado em problemas multiobjetivos, a fim de expor os compromissos entre os objetivos conflitantes de projeto, os quais podem estar associados a uma ou mais disciplinas envolvidas na otimização.

Atualmente, o MDO-SONDA está preparado para realizar somente a otimização das empenas de um foguete de sondagem. Entretanto, em seu código, escrito em C++ e orientado a objetos, há classes e objetos para que o programador possa estruturar interfaces adicionais em trabalhos futuros, visando a otimização de outros subsistemas do foguete, tais como: conjuntos adicionais de empenas, coifa (Figura 3.2), protuberâncias[9] e transições cônicas entre os estágios do foguete com diâmetro diferente. Os principais aspectos do MDO-SONDA são: arquitetura, dados de entrada, interfaces de saída, algoritmo de otimização e como proceder com a otimização.

3.4.2 Arquitetura

A arquitetura do MDO-SONDA é descrita em duas partes: a interação entre a função objetivo e o programa de alta-fidelidade de simulação da disciplina de aerodinâmica e o da disciplina de trajetória (Figura 3.9), e a interação entre o algoritmo de otimização e a função objetivo (Figura 3.8).

Os programas de alta-fidelidade de simulação são os seguintes:

a) *missile datcom*; e

b) *rocket simulation* (ROSI).

O *missile datcom* é um programa semi-empírico amplamente utilizado para estimar os coeficientes das forças aerodinâmicas, dos momentos e suas derivadas de estabilidade para uma ampla gama de configurações de mísseis em função dos seguintes descritores: número de Mach, altitude e ângulo de ataque (SOOY; SCHMIDT, 2005). Esse programa foi desenvolvido em FORTRAN 77 pela McDonnell Douglas. Mais tarde, a

---

[9] Exemplos de protuberâncias são: calhas, antenas e pequenos propulsores laterais que induzem o rolamento do foguete.

Força Aérea dos EUA produziu e documentou a versão em FORTRAN 90 (BLAKE, 1998), utilizada neste estudo.

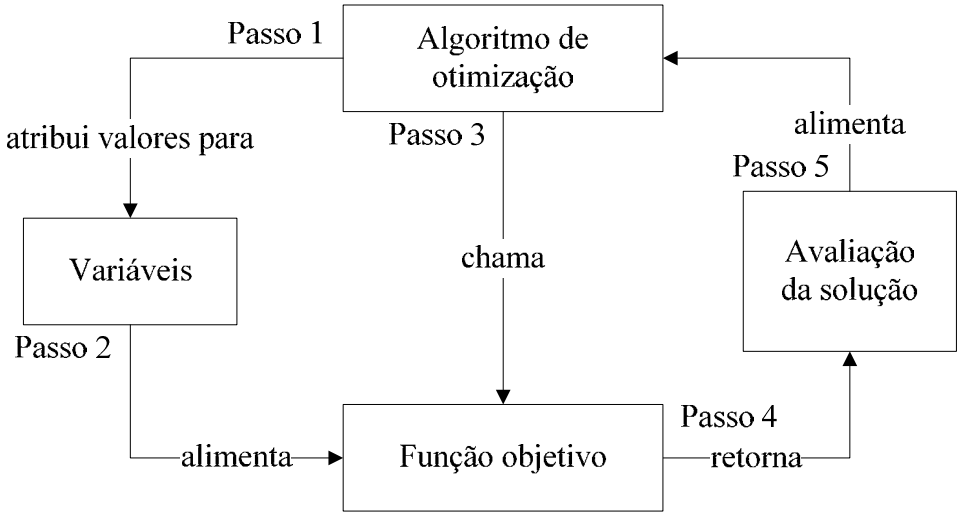


Figura 3.8 - Interação do algoritmo de otimização com a função objetivo.

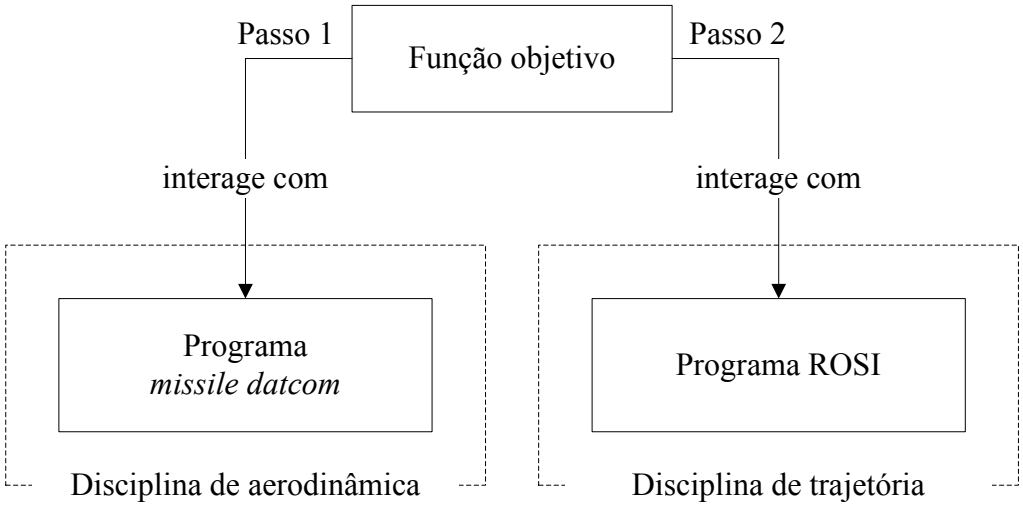


Figura 3.9 - Interação da função objetivo com os programas de aerodinâmica e trajetória.

O programa ROSI também foi escrito em FORTRAN. Esse programa calcula o movimento de um corpo rígido em um espaço tridimensional, considerando também sua rotação em guinada, arfagem e rolamento (ZIEGLTRUM, 1984; GOMES, 2004). Sua versão original foi desenvolvida pelo DLR (ZIEGLTRUM, 1984). Desde 1980, o ROSI

tem sido usado com sucesso para calcular a trajetória dos foguetes de sondagem brasileiros. O ROSI recebe do *missile datcom* as seguintes propriedades aerodinâmicas:

a) coeficiente de arrasto (C_D);

b) derivada do coeficiente da força normal com o ângulo de ataque ($C_{N\alpha}$);

c) derivada do coeficiente do momento em arfagem com o ângulo de ataque ($C_{M\alpha}$);

d) derivada do coeficiente do momento em arfagem com a taxa de arfagem (C_{Mq});

e) derivada do coeficiente do momento em rolamento com a taxa de rolamento (C_{lp}); e

f) centro de pressão (X_{cp}).

Além desses coeficientes, o ROSI também recebe a derivada do coeficiente do momento em rolamento com a deflexão da empena ($C_{l\delta}$). O C_{lp} e o $C_{l\delta}$ são usados para determinar a taxa de rolamento do foguete. Contudo, o *missile datcom* não fornece $C_{l\delta}$, mas fornece o coeficiente do momento em rolamento (C_l). Então, assume-se que o valor do C_l, calculado pelo *missile datcom* para uma deflexão de 0,01º, considerada suficientemente pequena, possa ser usado para estimar o valor de $C_{l\delta}$ (BARBOSA, A. N.; GUIMARÃES, 2012b),

$$C_{l\delta} = \frac{\partial C_l}{\partial \delta} \cong \frac{C_l}{0,01^{\circ}}. \tag{3.1}$$

O MDO-SONDA gerencia o processo de cada um desses programas, escreve seus arquivos de entrada, e lê seus resultados, coordenando a interação entre eles. Durante a optimização, se, por qualquer motivo, os processos desses programas ficam parados, eles são automaticamente encerrados e reiniciados com entradas diferentes. Primeiro, o MDO-SONDA interage com o *missile datcom*, obtendo os coeficientes aerodinâmicos. Em seguida, ele transfere os coeficientes para o arquivo de entrada do ROSI. Este

também recebe as propriedades de massa e inércia do foguete, que são as alterações de massa, centro de gravidade (X_{cg}), momento de inércia e produto de inércia, computadas pelo MDO-SONDA devido às separações de estágios, às ejeções de módulos de serviço e ao consumo de propelente ao longo do voo do foguete (Figura 3.10).

Um dos principais benefícios do MDO-SONDA é prover uma interface amigável[10] para inserir os valores de entrada e verificar, graficamente, as saídas de ambos os programas. Ele também proporciona a visualização do arquivo de entrada de cada programa, o qual é gerado automaticamente, para ter certeza de que não há qualquer erro nesses arquivos.

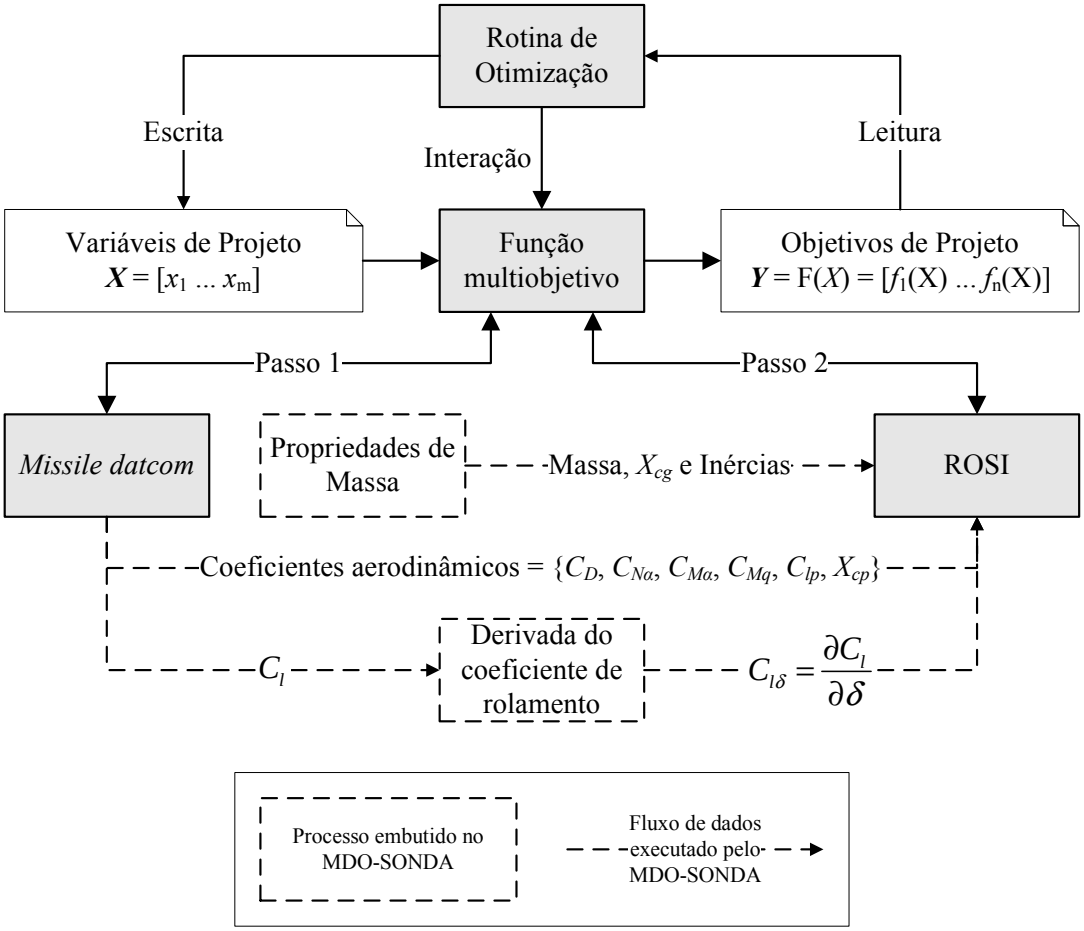


Figura 3.10 - Fluxo de dados no MDO-SONDA.

[10] As telas do MDO-SONDA são mostradas no Apêndice D.

3.4.3 Dados de entrada

As entradas do MDO-SONDA podem ser agrupadas em três partes. A primeira consiste das condições iniciais, dos eventos de voo e das definições do foguete para estimar os coeficientes aerodinâmicos e simular a trajetória. A segunda são os componentes que definem o problema de otimização: os objetivos do problema, as variáveis e as restrições. A terceira parte são as configurações do algoritmo de otimização.

Quanto à primeira parte, os eventos de voo típicos são os seguintes: ignição de motores, fim de queima, separação de estágios, ejeção da coifa e ativação de sistemas que interferem na dinâmica do foguete[11]. Esses eventos dividem o cálculo da trajetória em fases, pois produzem mudanças abruptas na configuração do foguete. Por exemplo, ocorrem mudanças abruptas nos coeficientes aerodinâmicos, dados em função da altitude e do número de Mach, quando há mudanças na geometria do foguete, devido à separação de suas partes, e quando forma-se a pluma de jato, devido ao funcionamento de seus motores.

A configuração das fases de voo consiste das seguintes definições do foguete: geometria do corpo, dados de propulsão e as propriedades de massa e inércia de cada subsistema, que ainda faz parte do foguete durante a fase de voo. Usando o teorema de Huygens-Steiner, o MDO-SONDA computa as inércias totais de cada fase de voo.

3.4.4 Interfaces de saída

O MDO-SONDA provê uma interface para a análise das saídas de cada programa e para o acompanhamento e a análise dos resultados da optimização. Usando essas interfaces, o usuário pode salvar os resultados e analisar posteriormente as soluções de Pareto, usando os recursos das interfaces criadas para o *missile datcom* e para o ROSI, a fim de verificar se os resultados estão consistentes.

---

[11] Exemplo de um sistema que interfere na dinâmica do foguete é o *yo-yo*, presente no VSB-30, cuja finalidade é fazer com que o foguete pare de rolar.

3.4.5 Algoritmo de otimização

Dado que pretende-se tratar com funções objetivo multimodais definidas em espaços contínuos, cujo gradiente é difícil de obter, as metaheurísticas tradicionais são os algoritmos genéticos, com operadores reais de recombinação e mutação, e o método do recozimento simulado (ROWELL; KORTE, 2003).

Considerando uma abordagem baseada em algoritmos genéticos, decidiu-se incorporar no MDO-SONDA o algoritmo genético multiobjetivo não-geracional (NGMOGA) com o operador real ORAD, proposto por este autor. O NGMOGA e o ORAD serão apresentados no próximo capítulo. Nele, descreve-se uma abordagem memética que combina o NGMOGA com o M-GEO. A incorporação do NGMOGA com o ORAD no MDO-SONDA deve-se, também, ao fato de este programa ter sido produzido concomitantemente com o desenvolvimento da abordagem memética.

Outro fator que motivou a decisão de usar o NGMOGA foi seu emprego bem-sucedido no estudo da otimização assistida por modelos substitutos, publicado em Barbosa, A. N. e Guimarães (2012a). Esse estudo será apresentado depois no Capítulo 5.

3.4.6 Procedimento de otimização

A otimização é um processo de tentativas e ajustes até se chegar a um resultado interessante. Primeiro, consiste em escolher intervalos preliminares para as variáveis do problema. A interface que permite a análise dos resultados da otimização é baseada em coordenadas paralelas (Subseção 6.2.1). A cada tentativa de otimização, recomenda-se verificar, por meio dessa técnica, se a região promissora do espaço de busca atinge os limites inferiores e superiores. Se isto ocorrer, sugere-se verificar se há necessidade de estender os limites das variáveis. Caso contrário, sugere-se restringir os limites para diminuir o espaço de busca.

A análise dos resultados a cada tentativa pode expor condições inviáveis, que não foram consideradas na formulação do problema de otimização. Neste sentido, o procedimento

de otimização por tentativas e ajustes torna-se também um processo onde aprende-se como formular corretamente o problema de otimização.

3.5 Definição do problema de otimização

O problema de otimização pode ser definido como segue. Dada a configuração original do VS-40 com uma carga útil fictícia de 240 kg, quer-se otimizar a geometria de suas empenas, a fim de minimizar a força de arrasto causada por elas e maximizar o menor intervalo entre os instantes críticos do voo. Esses instantes são: passagem pelo regime de voo transônico, pressão dinâmica máxima, margem estática mínima e cruzamento das frequências de rolamento e arfagem. É importante evitar a ocorrência simultânea desses instantes, pois podem tornar as condições de voo mais agressivas à estrutura do foguete. O regime de voo transônico refere-se às velocidades desenvolvidas entre Mach 0,8 e 1,4, em que a onda de choque se forma e pode causar forte instabilidade. A pressão dinâmica máxima está normalmente relacionada com o ponto de máximo carregamento aerodinâmico. Sabe-se, porém, que a empena tem pouquíssima influência sobre o instante desses dois eventos. Eles estão mais sujeitos à propulsão e distribuição de massa do foguete. Por outro lado, os dois últimos eventos são diretamente afetados pela geometria da empena, pois estão sujeitos à aerodinâmica do foguete. A margem estática, por exemplo, é função do centro de pressão (Figura 3.11). Nele atuam as forças aerodinâmicas. O centro de pressão varia em função da velocidade do foguete, mas também está sujeita à geometria da empena. As boas práticas recomendam que a margem estática mínima esteja entre 1,5 e 2 calibres, onde 1 calibre refere-se ao maior diâmetro do corpo central do foguete (NAKKA, 2001). Se a margem estática for positiva porém pequena, abaixo de 1 calibre, o foguete continuará estável, mas sujeito a oscilações em arfagem e guinada que poderão comprometer seu desempenho. O instante da margem estática mínima é crítico porque o foguete pode ficar sujeito a um regime maior de oscilações. O cruzamento das frequências de rolamento e arfagem pode induzir o fenômeno físico denominado *roll resonance* seguido por outro fenômeno chamado *roll lock-in*, onde a curva da taxa de rolamento desvia-se do caminho desejado, provocando instabilidade (CORNELISSE et al., 1979, p. 330).

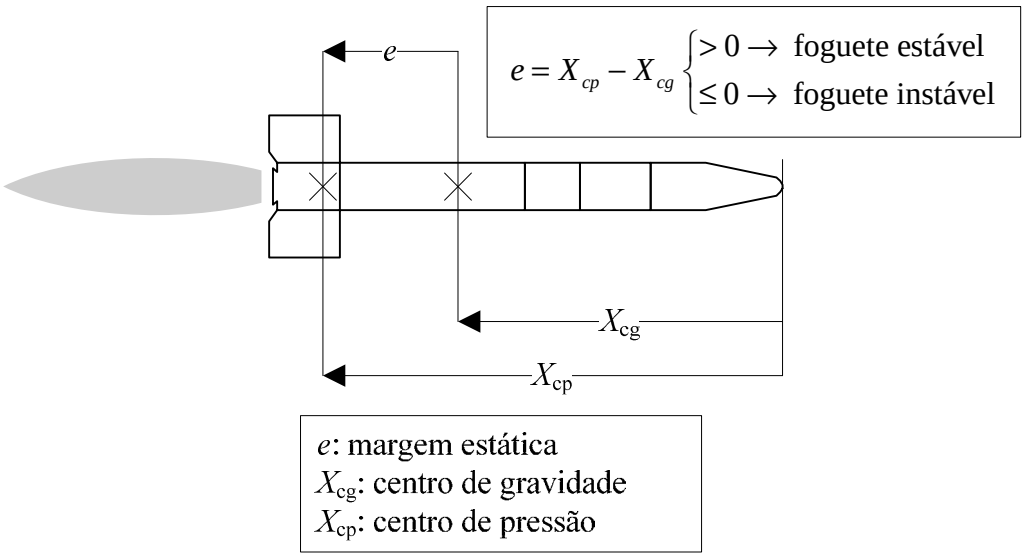


Figura 3.11 - Definição da margem estática.

O VS-40 é um foguete bi-estágio com apenas um conjunto de empenas, distribuídas de forma cruciforme, localizadas em sua cauda. Suas empenas possuem aerofólio hexagonal. Seu painel é constituído de dois segmentos. Neste problema, explora-se, como variáveis de projeto, os parâmetros da geometria do segundo segmento e o ângulo de incidência da empena (Figura 3.12). Considerou-se constante as propriedades de massa e inércia das empenas. Na Tabela 3.1, apresentam-se os limites das variáveis do problema. Os limites inferiores e superiores foram definidos de modo que as soluções geradas não fossem muito diferentes de perfis geralmente adotados para empenas.

As restrições do problema são as seguintes:

taxa de rolamento $\leq$ 2,3 Hz, e

margem estática (e) $\geq$ 1,4 calibres.

É importante manter a taxa de rolamento abaixo do valor admissível do VS-40. Se o limite for ultrapassado, o efeito da rotação poderá afetar o desempenho do foguete. Quanto ao limite inferior da margem estática, mantem-se o valor recomendado para esse foguete.

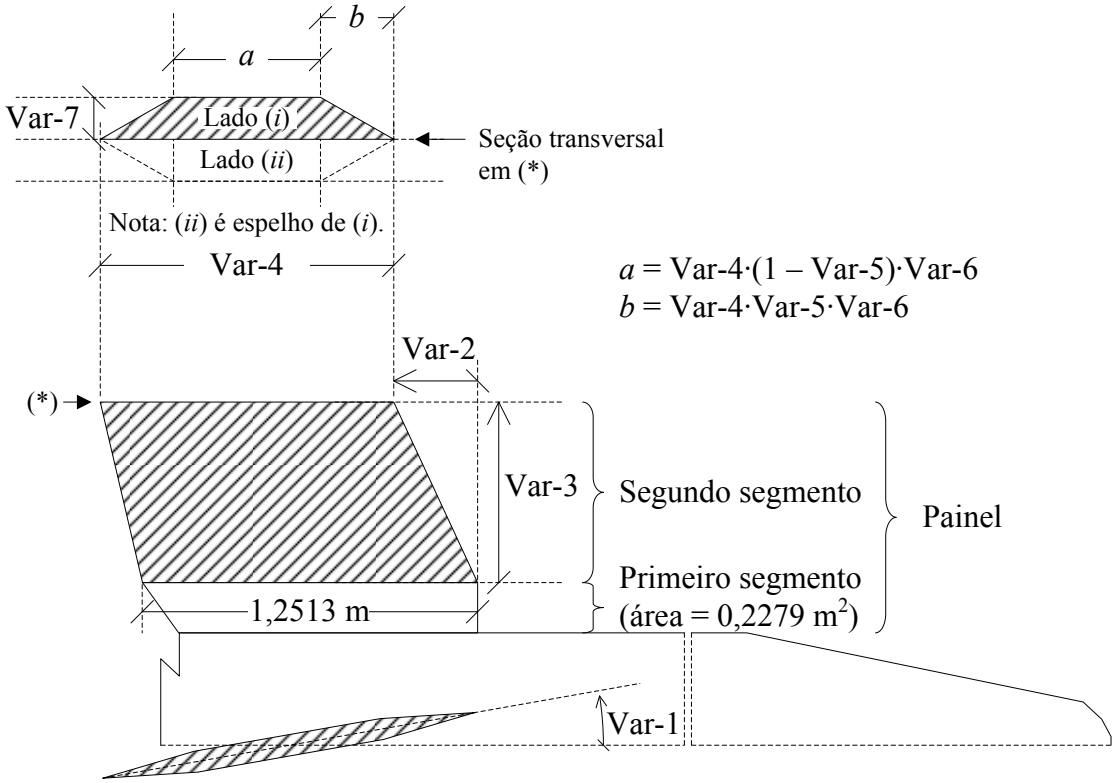


Figura 3.12 - Variáveis do problema de otimização.

Tabela 3.1 - Limites das variáveis do problema de otimização.

Variáveis	Valor nominal	Limite inferior	Limite superior
Var-1 (graus)	0,6	0,42	0,6
Var-2 (m)	0	0	2,48
Var-3 (m)	0,71	0,71	0,91
Var-4 (m)	1,25	1	1,25
Var-5	0,35	0,35	0,42
Var-6	0,8	0,72	0,96
Var-7 (m)	0,017	0,012	0,017

A configuração do algoritmo de otimização é apresentada na Tabela 3.2, com os valores atribuídos aos parâmetros do NGMOGA com o operador ORAD.

Tabela 3.2 - Configuração do NGMOGA com o operador ORAD.

Parâmetro	Valor
Tamanho da população	20
Número de gerações	600
Raio de vizinhança(1)	2
Fator de graduação(1)	0,5
Coeficiente de mutação (c)(2)	1,4
Expoente inferior de mutação (k_{inf})	1
Expoente superior de mutação (k_{sup})	6
Precisão	0,001

Notas: (1) parâmetros da função de aptidão que regulam a distribuição das soluções ao longo da fronteira de Pareto (Equação 4.12); e

(2) valor que, após vários testes, resultou no melhor desempenho do algoritmo.

3.6 Discussão dos resultados

Os resultados indicam que o custo computacional de estudos mais complexos, envolvendo mais variáveis e objetivos na otimização das empenas do VS-40, pode se tornar um problema. Uma única simulação envolvendo interações entre o *missile datcom* e o ROSI leva aproximadamente 12 s em um computador 3,0 GHz Dual Core (Tabela 3.3). Uma vez que 1200 avaliações da função multiobjetivo foram requeridas, a optimização demorou aproximadamente 4 h para ser concluída.

Constata-se que os objetivos do problema são conflitantes, pois a distribuição das soluções indicam a formação de uma curva de Pareto (Figura 3.13). Aparentemente, as soluções estão bem distribuídas (Figura 3.13). No entanto, apesar de a população ter 20 indivíduos (Tabela 3.2), apenas 11 pontos são observados na Figura 3.13. Ocorre que alguns desses indivíduos resultam no mesmo ponto. Na Tabela 3.4, apresentam-se as dimensões de apenas uma solução para cada ponto da Figura 3.13.

O resultado da otimização mostra-se coerente. O intervalo entre o instante da passagem pelo transônico e o da pressão dinâmica máxima é de 15 s, não importa o perfil de

empena que seja usado. Os resultados não excedem esse intervalo (Figura 3.13). O arrasto causado pelas empenas foi reduzido em até 29% (Solução 1, Figura 3.13).

Tabela 3.3 - Tempo de avaliação da função objetivo em um computador 3,0 GHz Dual Core.

Fases de voo do VS-40		Tempo de avaliação (s)	
		missile datcom	ROSI
1	(1)	4,313	0,515[4]
2	(2)	4,094	
3	(2)	1,109	
4	(2)	1,078	
5	(1)	1,094	
6	(2)	-[3]	
	Total por programa:	11,688	0,515
	Total por avaliação da função objetivo:	12,203	

Notas: (1) geometria com a pluma de jato;

(2) geometria sem a pluma de jato;

(3) não se executa o *missile datcom* nessa fase, pois os coeficientes são iguais aos da fase 4, dado que a geometria é idêntica; e

(4) o ROSI calcula a trajetória de todas as fases em uma única execução.

Na Figura 3.13, o eixo das abscissas representa o arrasto causado apenas pelas empenas e não o arrasto total sofrido pelo foguete. Sem empenas, o arrasto total foi de -40,24 kN. Assim, em termos de arrasto total, a redução máxima foi de 5%. Fato interessante é que as soluções são 2,8% a 19,6% maiores em área do que as empenas originais (Figura 3.14). Geralmente, a área tem mais impacto sobre o arrasto do que a geometria da empena. Contudo, apesar de as soluções terem uma área um pouco maior do que as empenas originais, constata-se que o arrasto produzido por elas é menor (Figura 3.14).

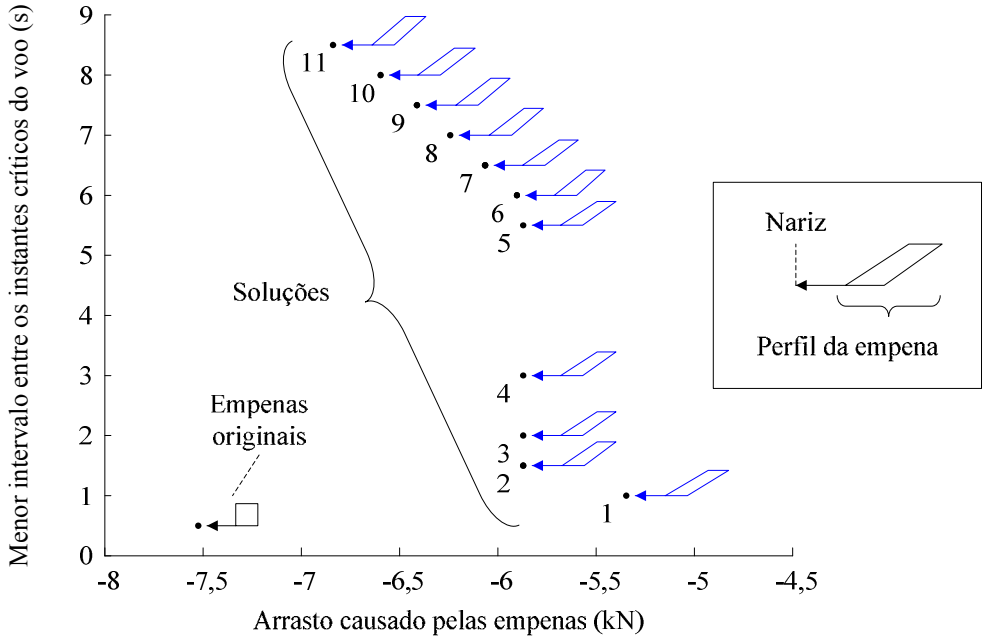


Figura 3.13 - Resultado da otimização.

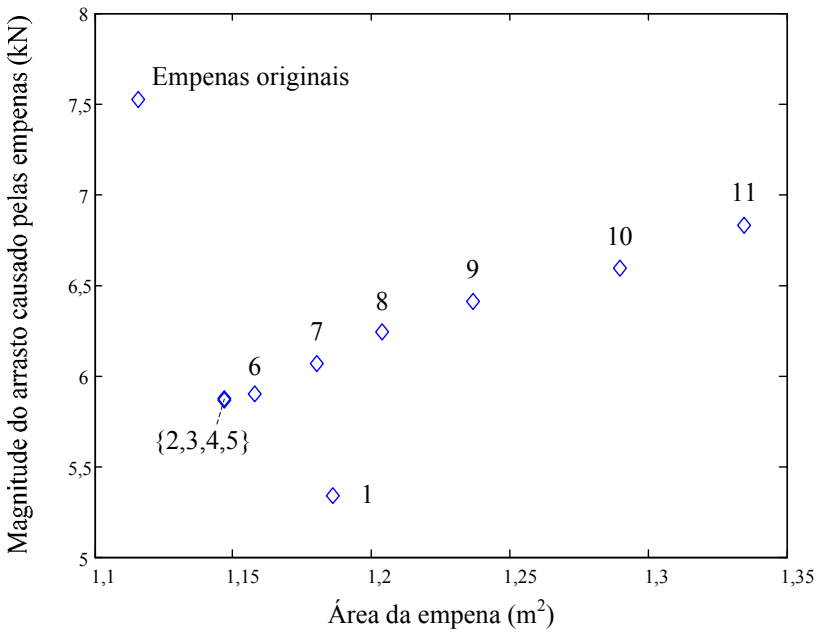


Figura 3.14 - Arrasto versus área da empena[12].

[12] As soluções estão ordenadas conforme apresentado na Figura 3.13.

Tabela 3.4 - Dimensões das soluções de empenas para o VS-40.

Soluções[1]	Var-1 (graus)	Var-2 (m)	Var-3 (m)	Var-4 (m)	$a^{(2)}$ (m)	$b^{(2)}$ (m)	Var-7 (m)
Original	0,600	0,000	0,710	1,251	0,652	0,348	0,0168
1	0,587	2,429	0,793	1,167	0,584	0,382	0,0117
2	0,589	1,995	0,787	1,084	0,608	0,426	0,0118
3	0,550	1,995	0,787	1,084	0,597	0,418	0,0118
4	0,526	1,995	0,787	1,084	0,584	0,409	0,0118
5	0,427	1,995	0,787	1,084	0,577	0,409	0,0118
6	0,420	1,766	0,793	1,095	0,610	0,440	0,0117
7	0,420	1,995	0,812	1,095	0,626	0,423	0,0117
8	0,420	1,995	0,842	1,066	0,603	0,416	0,0118
9	0,421	1,995	0,862	1,089	0,622	0,421	0,0118
10	0,420	2,029	0,862	1,212	0,690	0,470	0,0117
11	0,422	1,855	0,905	1,196	0,684	0,461	0,0118

Notas: [1] as soluções estão ordenadas conforme apresentado na Figura 3.13; e
[2] dimensões calculadas em função de Var-4, Var-5 e Var-6 (Figura 3.12).

As empenas originais possuem uma geometria retangular. Esse tipo de geometria é o que causa maior arrasto no regime de voo supersônico em comparação com os outros tipos com mesma área e semi-envergadura (FLEEMAN, 2006, p. 52). Dado que a maior parte do voo do VS-40 ocorre em regime supersônico (Figura 3.15), a geometria retangular não é a mais adequada para reduzir o arrasto. Este estudo reforça, portanto, que as empenas do VS-40 podem ser melhoradas.

Em desenvolvimentos futuros, pretende-se envolver mais disciplinas no problema de otimização, incorporando, por exemplo, os programas de propulsão e estruturas. No entanto, ao manipular variáveis de mais disciplinas, o tempo de execução da otimização tornar-se-á mais longo e a análise dos resultados, mais complexa. Surgiu então a ideia de desenvolver as abordagens para lidar com o desafio do custo computacional e da sumarização dos resultados da otimização, apresentadas agora nos próximos capítulos.

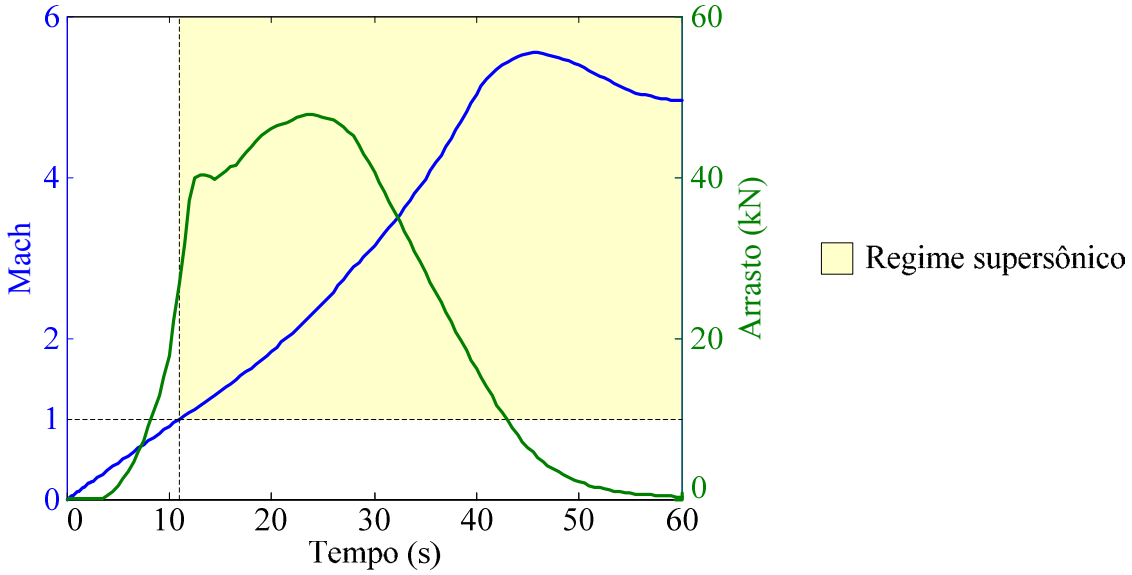


Figura 3.15 - Regime de voo do VS-40 com as empenas originais.

4 COOPERAÇÃO ENTRE MÉTODOS DE OTIMIZAÇÃO

4.1 Introdução

Os algoritmos meméticos podem ser considerados um desdobramento das estratégias evolucionárias. Ambos pertencem à classe dos métodos estocásticos. Esses métodos mostram-se mais eficientes e efetivos em problemas multiobjetivos complexos, onde outras abordagens de otimização não funcionam muito bem (GOH; TAN, 2009).

Uma das principais características dos algoritmos evolucionários é que são baseados na amostragem e avaliação simultânea de múltiplas soluções candidatas para um problema de otimização. O conjunto dessas soluções candidatas é chamado de população. Cada solução desse conjunto é chamada de indivíduo. Uma solução gerada a partir de outra é denominada descendente. Aquelas que a originaram são denominadas genitoras.

Cada iteração do algoritmo representa uma geração, onde novos indivíduos podem ser introduzidos na população e outros, removidos. A cada geração, a população deve evoluir de tal forma que os indivíduos sejam em média mais aprimorados do que os das gerações anteriores. Espera-se, dessa forma, que o algoritmo convirja para uma solução ótima, no caso de problemas mono-objetivos, ou então para um conjunto de soluções de Pareto, no caso de problemas multiobjetivos.

Segundo Dréo et al. (2006), a população pode ser vista como forma de memória, pois a cada geração as características das soluções mais aptas são transmitidas e aprimoradas dos genitores para seus descendentes através de recombinação e mutação.

Novas abordagens de algoritmos evolucionários substituem a operação de mutação, onde determinadas características são alteradas aleatoriamente por uma operação que executa uma busca local com o auxílio de outro método de otimização, como, por exemplo, a busca tabu e o método *simulated annealing*.

Essas abordagens recebem a denominação de algoritmos meméticos. Segundo Moscato e Cotta (2003), a denominação algoritmo memético abrange uma ampla classe de metaheurísticas. Em essência, os algoritmos meméticos podem ser interpretados como

uma estratégia de busca na qual as metaheurísticas incorporadas interagem umas com as outras com o objetivo de explorar as características do problema.

Espera-se, com essa abordagem, que a eficiência do processo de otimização seja maior do que a alcançada por essas metaheurísticas sem a interação entre elas. O ganho de eficiência proporcionado pelo algoritmo memético pode ser atribuído à sinergia entre os diferentes métodos de otimização que incorpora. Os algoritmos meméticos também são chamados de algoritmos evolucionários híbridos.

Em 1988, Moscato introduziu pela primeira vez um algoritmo híbrido resultante da combinação do tradicional algoritmo genético com o método *simulated annealing* e, em 1989, publicou um artigo com o termo algoritmo memético. O termo memético deriva da palavra meme, conceito associado à ideia de uma evolução cultural, análogo ao conceito de gene, associado a evolução biológica.

O conceito de meme foi introduzido por Richard Dawkins, em 1976, em seu primeiro livro “O Gene Egoísta”, motivado pela necessidade de dar um nome a uma unidade de transmissão cultural, ou unidade de imitação. Meme é uma abreviação da palavra mimeme, assim proposta por ser uma palavra monossílaba que soa como gene.

Dawkins também sugeriu que o meme pudesse ser visto como uma forma de memória (DAWKINS, 2006, p. 192). Meme refere-se a uma unidade de informação que se propaga em uma população, de indivíduos para outros, através de um processo coletivo de identificação com algo e o desejo de imitar os outros.

Essas unidades podem ter diversas representações. Podem ser ideias, canções, clichês, formas de se vestir, de construir artefatos, valores estéticos e morais, ou qualquer outra coisa que se propague entre indivíduos na forma de informação, conscientemente ou não. De acordo com Moscato e Cotta (2003), o conceito de meme sugere um processo evolutivo associado à cultura de uma sociedade, onde a informação, constituída de memes, pode sofrer alterações quando transmitida de um indivíduo para outro.

Segundo esses autores, os algoritmos meméticos são interpretados como estratégias nas quais constitui-se uma população de métodos de otimização, vistos como agentes, que cooperam ou competem entre si com o objetivo de aumentar a eficiência dos processos associados à solução de problemas ou à tomada de decisões. Essa interpretação estende o conceito de população, formulado inicialmente no contexto dos algoritmos evolucionários, para também denotar um conjunto de métodos de otimização, onde cada método é visto como um indivíduo, ou agente, dessa população. Também nesse contexto, a população pode ser vista como uma forma de memória, pois informações sobre o processo de otimização são transmitidas de um método para outro.

Neste capítulo, apresenta-se uma abordagem memética baseada na cooperação do algoritmo genético multiobjetivo não-geracional (NGMOGA) com o algoritmo multiobjetivo da otimização extrema generalizada (M-GEO).

Uma vez que o objetivo é aplicá-los, mais adiante, em problemas contínuos, apresenta-se, também, operadores reais para esses algoritmos, pois as versões originais do NGMOGA e do M-GEO, propostas respectivamente por Borges (1999) e Galski (2006), utilizam operadores binários, apesar de Galski ter proposto versões alternativas com operadores reais para o GEO (versão mono-objetivo).

Este capítulo está organizado como segue. Primeiro, apresenta-se a fundamentação teórica deste capítulo, onde abordam-se o conceito de otimização multiobjetivo, os algoritmos de otimização supracitados, e dois operadores reais, um de recombinação, o operador SBX (*simulated binary crossover*), e outro de mutação, proposto por Michalewicz (1996). Em seguida, descreve-se em detalhes a parte teórica que corresponde à contribuição deste trabalho, uma abordagem para combinar o NGMOGA com o M-GEO. Por último, descreve-se o estudo de caso para avaliação dessa abordagem e comentam-se os resultados.

4.2 Fundamentação teórica

4.2.1 Problema de otimização multiobjetivo

Em termos gerais, um problema de otimização multiobjetivo é expresso como segue abaixo.

Minimizar:

$$f_i\left(x_1, x_2, \ldots, x_n\right),\ i = 1, 2, \ldots, p\ . \tag{4.1}$$

Sujeito a:

$$g_j\left(x_1, x_2, \ldots, x_n\right) \leq 0,\ j = 1, 2, \ldots, q \tag{4.2}$$

e

$$l_k \leq x_k \leq u_k,\ x_k \in \mathbb{R},\ k = 1, 2, \ldots, n\ , \tag{4.3}$$

onde x_1, x_2, ..., x_n são as variáveis do problema, também ditas variáveis de projeto, limitadas inferiormente por l_1, l_2, ..., l_n, respectivamente, e limitadas superiormente por u_1, u_2, ..., u_n, respectivamente, f_1, f_2, ..., f_p são as funções componentes da função multiobjetivo, ou objetivos de projeto, e g_1, g_2, ..., g_q são as restrições do problema. Os limites inferiores e superiores das variáveis do problema definem o espaço de busca no domínio da função.

A otimização de um problema multiobjetivo consiste em encontrar um conjunto de soluções viáveis denominadas de Pareto. As soluções de Pareto são definidas pelo conceito de não-dominância, ou critério de Pareto, aplicado ao conjunto de todas as soluções viáveis do problema (HUBAND et al., 2006, p. 478). Seja *V* um conjunto qualquer de soluções viáveis. Dadas duas soluções **a** e **b** pertencentes a *V*, pelo critério de Pareto, **a** domina **b** se, e somente se, **a** é no mínimo tão boa quanto **b** em relação a todos os componentes da função multiobjetivo, e melhor do que **b** em ao menos um componente (Figura 4.1).

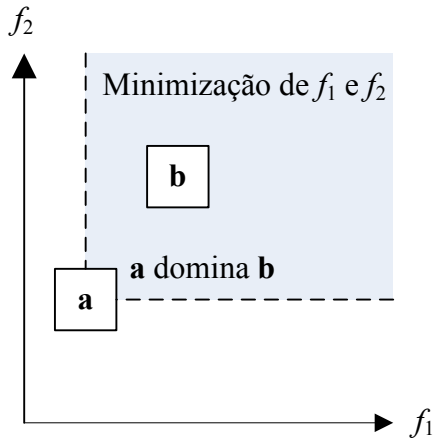


Figura 4.1 - Conceito de dominância.

Uma solução pertencente a V é dita não-dominada se, e somente se, pelo critério de Pareto, não há nenhuma outra solução de V que a domine. Se V é o conjunto de todas as soluções viáveis, as soluções não-dominadas de V constituem o conjunto das soluções de Pareto. A representação dessas soluções no contradomínio da função multiobjetivo define a fronteira de Pareto ideal. Em geral, os métodos de otimização visitam apenas parte do conjunto de soluções viáveis e, por esse motivo, seu resultado final é uma aproximação da fronteira de Pareto ideal. Quando este trabalho refere-se à representação das soluções não-dominadas no domínio da função, usa-se o termo conjunto de Pareto, e, quando refere-se à representação no contradomínio, fronteira de Pareto. Além disso, introduz-se o conceito de dominância entre conjuntos de soluções com a mesma cardinalidade. Seja $D_X(Y)$ a quantidade de soluções do conjunto X dominadas pelas soluções do conjunto Y, podendo ser expressa em termos percentuais. Dados dois conjuntos P e Q, tal que $\#P$ é igual a $\#Q$, P domina Q se, e somente se, $D_P(Q)$ é menor do que $D_Q(P)$ (onde o símbolo # denota a cardinalidade de conjunto).

4.2.2 Algoritmo genético multiobjetivo não-geracional

Os algoritmos genéticos são uma classe de métodos inspirados na evolução biológica das espécies e na genética. De acordo com a crença darwinista, quanto mais bem-sucedido for um indivíduo em relação a outros de sua espécie, mais frequentemente será escolhido para reproduzir-se ou sobreviver (DRÉO et al., 2006, p. 77). Esse princípio

está incorporado no algoritmo genético, onde as melhores soluções de um problema são mais frequentemente escolhidas para originarem outras soluções ou permanecerem no conjunto de soluções, enquanto, em sua maioria, as piores são descartadas. Em termos gerais, os algoritmos genéticos consistem de três operações: a inicialização de uma população de soluções, chamadas de indivíduos; a geração de novos indivíduos a partir de indivíduos dessa população; e a substituição de indivíduos da população por novos indivíduos. As duas últimas operações se repetem, um número de vezes, até que a solução ótima seja encontrada. Essas iterações caracterizam o laço geracional do algoritmo genético.

O algoritmo genético multiobjetivo não-geracional (NGMOGA) baseia-se na versão apresentada por Borges (BORGES, 1999; BORGES; BARBOSA, H. J. C., 2000). O NGMOGA realiza as mesmas operações descritas, acima, em termos gerais. O que o diferencia de outros algoritmos genéticos são as regras que emprega nessas operações. Primeiro, o NGMOGA gera uma população inicial e avalia a aptidão de cada indivíduo, ou seja, o quanto cada indivíduo é melhor do que os outros. Então, iterações ocorrem, onde cada iteração consiste das seguintes operações:

a) seleção de dois indivíduos, denominados genitores;

b) geração dos descendentes desses indivíduos, os quais consistem de dois novos indivíduos;

c) avaliação da aptidão dos descendentes;

d) seleção do descendente mais apto a fim de incluí-lo na população; e

e) substituição do indivíduo menos apto da população pelo novo indivíduo.

No passo (e), acima, a substituição ocorre se, somente se, o novo indivíduo é mais apto do que o indivíduo menos apto da população. Neste trabalho, no entanto, adotou-se uma regra um pouco mais flexível. A substituição ocorre se, somente se, o novo indivíduo não é menos apto do que o indivíduo menos apto da população. Incorporou-se essa mudança porque ela reduziu as estagnações do algoritmo percebidas nos testes iniciais.

A abordagem não-geracional é adequada a problemas multiobjetivos porque preserva as soluções que estão bem próximas da fronteira de Pareto (VALENZUELA-RENDÓN; URESTI-CHARRE, 1997). O nome dado a essa abordagem, não-geracional, é discutível, pois, intuitivamente, cada iteração do algoritmo significa uma geração, em que um novo indivíduo é criado e pode ser introduzido na população. Essa denominação, no entanto, significa que os descendentes não necessariamente substituem seus genitores.

A seleção dos genitores é feita por pressão seletiva, usando uma abordagem baseada no *rank* dos indivíduos, enfileirados em ordem crescente de acordo com o respectivo valor da função de aptidão (BORGES, 1999, p. 18). A escolha do indivíduo consiste em determinar um índice *j* relativo a sua posição no *rank*, dado por

$$j = \left\lceil n_{\text{pop}} \left(b - \sqrt{b^2 - 4(b-1)U} \right) \frac{0,5}{b-1} \right\rceil, \tag{4.4}$$

onde n_{pop} é o número total de indivíduos da população, *b* é um *bias* definido no intervalo (1,2], o qual regula a pressão de seleção, *U* é um número aleatório com distribuição uniforme no intervalo (0,1], e o símbolo $\lceil \cdot \rceil$ denota a função de arredondamento teto. Quanto maior o valor de *b*, maior é a probabilidade de selecionar indivíduos no topo do *rank*, isto é, com índices a partir de 1. Segundo Whitley (1989), p. 120, um *bias* de 1,5 é adequado, e diz que os indivíduos no topo do *rank* são 1,5 vezes mais prováveis de serem escolhidos do que os inferiores. Decidiu-se por aplicar a pressão seletiva com *b* igual a 1,5. Assim, substituindo *b* por 1,5 na Equação 4.4, tem-se a forma adotada neste trabalho:

$$j = \left\lceil n_{\text{pop}} \left(1,5 - \sqrt{2,25 - 2U} \right) \right\rceil. \tag{4.5}$$

A geração de dois novos indivíduos a partir de dois indivíduos selecionados é feita por meio de operadores binários de recombinação e mutação. A probabilidade de o algoritmo aplicar a recombinação é de 80%, enquanto a de aplicar a mutação é de 20%.

Cada indivíduo X é representado por uma sequência binária denominada genótipo de comprimento L. Essa sequência é composta dos valores de N variáveis de projeto justapostos. O valor de cada variável corresponde a uma subsequência de comprimento l_q (q = 1,...,N).

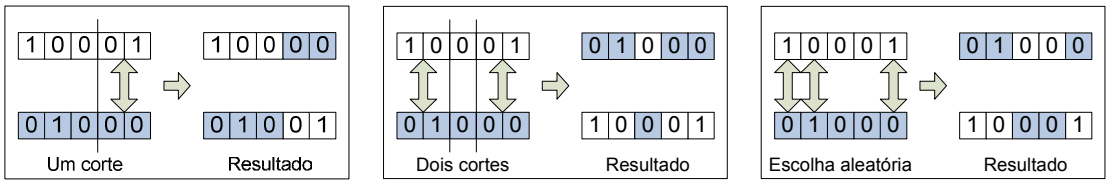
$$X = \underbrace{\underbrace{011...0}_{l_1}\underbrace{101...1}_{l_2}...\underbrace{001...1}_{l_N}}_{L} \tag{4.6}$$

O NGMOGA emprega as variantes clássicas do operador de recombinação (Figura 4.2). Conforme Dréo et al. (2006), p. 98, essas variantes são as seguintes:

a) recombinação com corte em um ponto (*single point crossover*);

b) recombinação com corte em dois pontos (*two point crossover*); e

c) recombinação uniforme (*uniform crossover*).



(a) (b) (c)

Figura 4.2 - Variantes clássicas do operador de recombinação. (a) Corte em um ponto; (b) corte em dois pontos; (c) recombinação uniforme.

Quando o algoritmo decide aplicar a recombinação, uma das variantes acima é usada aleatoriamente. Quando decide aplicar a mutação, a operação é semelhante à recombinação uniforme (Figura 4.2c), mas com uma diferença, a probabilidade de mudar um bit é calculada em função da distância de Hamming entre os genótipos dos genitores (Equação 4.7). A probabilidade (P_{mutat}) de alterar um bit é dada por

$$P_{\text{mutat}} = \left(1 - \frac{d_{\text{Ham}}}{L}\right) \cdot \frac{2}{L}, \tag{4.7}$$

onde d_{Ham} é a distância de Hamming, número de bits diferentes entre os genótipos dos genitores, e L é comprimento do genótipo.

A aptidão de cada indivíduo é uma função não linear de sua dominância e vizinhança, cuja definição é apresentada a seguir. A função de aptidão A do i-ésimo indivíduo da população é dada por

$$A(i) = \left[1 + D(i)\right]^{\beta} \left[1 + N(i)\right], \tag{4.8}$$

onde $D(i)$ é a dominância do i-ésimo indivíduo; $N(i)$ é a vizinhança; e o exponente β é fixo em 1. A função de aptidão (Equação 4.8), baseada em critérios de dominância e vizinhança, tem por objetivo regular a distribuição das soluções ao longo da fronteira de Pareto, como mostra a Figura 4.3.

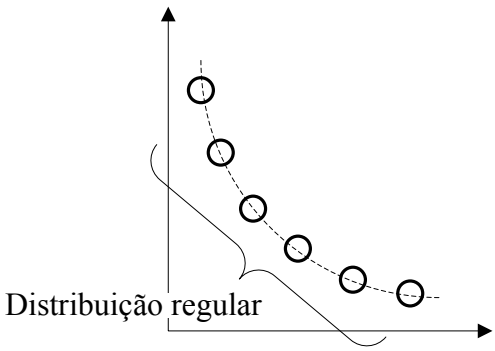


Figura 4.3 - Esboço da distribuição regular de soluções ao longo da fronteira de Pareto.

A dominância e a vizinhança assumem valores não negativos. A dominância é zero quando o indivíduo é não-dominado pelos outros, de acordo com o critério de Pareto[13], e a vizinhança é zero quando o indivíduo está afastado dos demais, no mínimo, a uma distância denominada raio de vizinhança, estabelecida como parâmetro de entrada do algoritmo. A aptidão do indivíduo decresce com a função A (Equação 4.8). Quanto menos dominado e mais isolado um indivíduo encontra-se em relação aos outros, menor

[13] O critério de Pareto é apresentado na Subseção 4.2.1.

é o resultado da função A para esse indivíduo. A dominância D do i-ésimo indivíduo é dada por

$$D(i) = \sum_{j=1}^{n_{\text{pop}}} nd(i, j) \tag{4.9}$$

e

$$nd(i, j) = \begin{cases} 1, \text{ se } f_l(j) \nprec f_l(i) \ \forall l = 1, \ldots, k \ \wedge \ \exists l \in \{1, \ldots, k\} \mid f_l(j) \succ f_l(i) \\ 0, \text{ caso contrário,} \end{cases} \tag{4.10}$$

onde n_{pop} é o número total de indivíduos da população, f_l é o l-ésimo componente da função multiobjetivo de um total de k componentes, e os símbolos $\nprec$ e $\succ$ denotam "não pior do que" e "melhor do que", respectivamente.

A vizinhança N do i-ésimo indivíduo é dada por

$$N(i) = \sum_{j=1}^{n_{pop}} dv(i, j) \tag{4.11}$$

e

$$dv(i, j) = \begin{cases} 1 - \left(\dfrac{d_{ij}}{\sigma} \right)^{\alpha}, & \text{se } i \neq j \text{ e } d_{ij} < \sigma \\ 0, & \text{caso contrário,} \end{cases} \tag{4.12}$$

onde n_{pop} é o número total de indivíduos da população, dv é uma função de similaridade, d_{ij} é a distância euclidiana entre o i-ésimo e o j-ésimo indivíduo, σ é o raio de vizinhança e α é o fator de graduação. O raio de vizinhança e o fator de graduação são parâmetros de entrada que regulam a distribuição das soluções ao longo da fronteira de Pareto.

4.2.3 Algoritmo multiobjetivo do método da otimização extrema generalizada

O algoritmo da otimização extrema generalizada (*generalized extreme optimization* – GEO) baseia-se na teoria da criticalidade auto-organizada (*self-organized criticality* – SOC), que é uma propriedade, ou classe, de sistemas dinâmicos que possuem um ponto crítico como atractor, ou seja, um ponto característico para o qual o sistema evolui independentemente do ponto de origem (DHAR, 2006). Para ilustrar essa propriedade, descreve-se, a seguir, o que ocorre com uma pilha de grãos à medida que partículas são depositadas sobre ela (Figura 4.4). Quando a declividade da pilha alcança seu ponto crítico, ocorrem deslizamentos, fazendo com que a inclinação da superfície retorne para uma condição mais estável. Os deslizamentos são causados por grãos que não estão bem adaptados na estrutura da pilha.

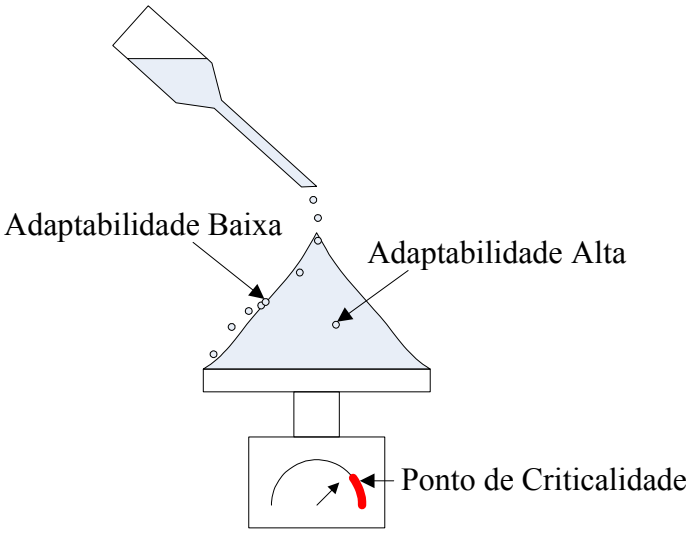


Figura 4.4 - Ponto de criticalidade de uma pilha de grãos.

Quanto menos adaptado, maior é a probabilidade de o grão provocar deslizamentos. De acordo com Bak et al. (1988), essa probabilidade segue uma lei de potência descrita por

$$P(s) \approx s^{-\tau}, \tag{4.13}$$

onde s é um parâmetro proporcional à adaptabilidade do grão, e τ é um número real positivo denominado índice de criticalidade.

À medida que o declive da superfície da pilha aumenta, os grãos mais próximos da superfície tornam-se menos adaptados e, consequentemente, a probabilidade de deslizamentos torna-se maior (Figura 4.4). Traçando uma analogia com o que ocorre em uma pilha de grãos, explica-se como o GEO e sua versão multiobjetivo, o M-GEO, buscam inspiração no SOC. Nesse contexto, as soluções de um problema multiobjetivo são semelhantes aos grãos. O espaço de soluções viáveis é equivalente à pilha de grãos. O conjunto de Pareto é representado pelos grãos menos adaptados, que estão na superfície da pilha. Quanto melhor, menos adaptada a solução está em relação às outras. Quanto menos adaptada, maior é a probabilidade de ser selecionada para gerar novas soluções. À medida que as soluções convergem para a fronteira de Pareto, chega-se a um ponto crítico, onde a busca por soluções ótimas não progride mais, tal como ocorre na pilha de grãos, onde a declividade não se altera mais quando alcança seu ponto crítico. O algoritmo do GEO, baseado em Sousa (2002), p. 48, consiste dos passos a seguir, de (a) até (m).

a) Escolha aleatoriamente uma solução C representada por uma sequência binária de comprimento L. Essa sequência é composta dos valores de N variáveis de projeto justapostos. O valor de cada variável corresponde a uma subsequência de comprimento l_q (q = 1,...,N).

$$C = \underbrace{\underbrace{011...0}_{l_1}\underbrace{101...1}_{l_2}...\underbrace{001...1}_{l_N}}_{L} \tag{4.14}$$

b) Calcule $V = F_{\text{obj}}(C)$, onde F_{obj} é a função objetivo.

c) Faça $C_{\text{melhor}} \leftarrow C$ e $V_{\text{melhor}} \leftarrow V$ (onde "←" denota atribuição de valor).

d) Altere o valor do i-ésimo bit de C, gerando uma sequência C_i (Equação 4.15). A sequência C_i representa uma solução candidata derivada de C.

$$\begin{aligned} C &= 011...0101...\overset{i\text{-ésimo bit}}{\boxed{1}}...001...1 \quad \overset{\text{Alteração do } i\text{-ésimo bit}}{\Rightarrow} \\ C_i &= 011...0101...\overset{\text{Alteração}}{\boxed{0}}...001...1 \end{aligned} \tag{4.15}$$

e) Calcule ΔV_i definido por

$$\Delta V_i = V_i - V_{melhor}\,, \tag{4.16}$$

onde $V_i = F_{\text{obj}}(C_i)$. ΔV_i representa o índice de adaptabilidade da solução C_i.

f) Retorne para o passo (d) até percorrer, sequencialmente, todos os bits de C.

g) Ordene as sequências derivadas de C, da menos a mais adaptada. Dadas duas sequências quaisquer derivadas de C, uma com alteração no i-ésimo bit e outra no j-ésimo, em um problema de minimização, C_i é menos adaptada do que C_j, se ΔV_i é menor do que ΔV_j, enquanto em um problema de maximização, é menos adaptada, se ΔV_i é maior do que ΔV_j. Se duas ou mais sequências derivadas de C empatam, são organizadas em ordem aleatória.

h) Escolha aleatoriamente uma sequência C^* derivada de C.

i) Gere um número aleatório RAN entre 0 e 1 com distribuição uniforme.

j) Calcule a probabilidade P dessa sequência ser escolhida com base na lei de potência dada por

$$P = k^{-\tau}\,, \tag{4.17}$$

onde k é a ordem da sequência C^*, determinada no passo (g), e τ é o índice de criticalidade. Se $P \geq RAN$, a sequência C^* é escolhida. Caso contrário, retorna-se para o passo (h) até que uma sequência derivada de C seja escolhida.

k) Faça $C \leftarrow C^*$ e $V \leftarrow F_{\text{obj}}(C^*)$.

l) Caso V seja melhor do que V_{melhor}, faça $V_{\text{melhor}} \leftarrow V$ e $C_{\text{melhor}} \leftarrow C$.

m) Repita os passos de (d) a (l) até completar o número total de iterações.

O algoritmo do GEO retorna em C_{melhor} a solução ótima do problema. O procedimento do GEO consiste em escolher aleatoriamente uma solução como ponto de partida. Essa solução é denominada semente. No M-GEO, parte-se de um conjunto de sementes e executam-se os passos do GEO para cada semente. No entanto, o GEO foi concebido para problemas do tipo mono-objetivo. Assim, o M-GEO possui um laço intermediário de iterações chamado de torneios. A cada torneio, escolhe-se aleatoriamente um componente da função multiobjetivo. Então, os passos do GEO são realizados, fixando-se esse componente como único objetivo de otimização durante o torneio.

O algoritmo do M-GEO, baseado em Galski (2006), p. 208, consiste dos passos a seguir, de (a) até (j).

a) Escolha aleatoriamente uma solução C (semente).

b) Escolha aleatoriamente um componente f da função multiobjetivo.

c) Altere o valor do i-ésimo bit de C, gerando uma solução C_i.

d) Guarde a solução C_i.

e) Retorne para o passo (c) até percorrer, sequencialmente, todos os bits de C.

f) Execute os passos de (g) a (j) do GEO (página 44) em relação a f e, em seguida, retorne a solução escolhida, derivada de C, denotada por C^*.

g) Faça $C \leftarrow C^*$.

h) Retorne para o passo (b) até completar o número total de torneios.

i) Retorne para o passo (a) até completar o número total de sementes.

j) Identifique as soluções guardadas que são não-dominadas.

Neste trabalho, porém, definiu-se o critério da adaptabilidade como segue. Caso o problema seja de minimização: uma solução C_i qualquer é menos adaptada do que outra

C_j, se $f(C_i) < f(C_j)$, onde f é a componente da função multiobjetivo escolhida em dado torneio, pois, conforme o passo (g) do GEO,

$$\Delta V_i < \Delta V_j \Rightarrow f\left(C_i\right) - V_{melhor} < f\left(C_j\right) - V_{melhor} \Rightarrow f\left(C_i\right) < f\left(C_j\right). \qquad (4.18)$$

Caso contrário: C_i é menos adaptada do que C_j se $f(C_i) > f(C_j)$, pois

$$\Delta V_i > \Delta V_j \Rightarrow f\left(C_i\right) - V_{melhor} > f\left(C_j\right) - V_{melhor} \Rightarrow f\left(C_i\right) > f\left(C_j\right). \qquad (4.19)$$

4.2.4 Operador real simulated binary crossover

O operador real *simulated binary crossover* (SBX), proposto por Deb e Agrawal (1995), foi concebido para simular um operador binário, como o próprio nome sugere. Um operador gera descendentes a partir de um conjunto de indivíduos. O que se espera é que a maior concentração de descendentes ocorra nas proximidades dos genitores, intensificando a busca, pois os genitores são por vezes os mais aptos da população. No entanto, é importante também que uma pequena parte ocorra em regiões distantes dos genitores, diversificando a busca, a fim de que seja uma busca global. A diversificação é um recurso para escapar de mínimos ou máximos locais. Um bom compromisso entre a intensificação e a diversificação é fundamental para o melhor desempenho da metaheurística. No entanto, nem todos os operadores geram descendentes da maneira como se espera. Exemplos são os seguintes operadores: a recombinação geométrica de Michalewicz e o crossover blend (BLX). O primeiro gera um único descendente resultante da média geométrica dos genitores. O segundo gera um espalhamento de descendentes com centro na média geométrica dos genitores. Segundo Medeiros (2002), esses operadores resultam em uma convergência mais lenta do que o SBX, o qual gera um espalhamento centrado nos genitores. A similaridade com operadores binários de recombinação, sugerida pelo SBX, ocorre em problemas com uma variável. Neste caso, o resultado obtido com os operadores binários é um espalhamento de descendentes concentrado nas proximidades dos genitores, como também ocorre com a aplicação do SBX. No entanto, com duas ou mais variáveis em jogo, os operadores binários geram

também concentrações de descendentes em regiões que não coincidem com os genitores (Figuras 4.5 e 4.6). Segue, abaixo, o algoritmo do SBX.

a) Selecione dois indivíduos $\mathbf{p}_1$ e $\mathbf{p}_2$, denominados genitores:

$$\mathbf{p}_1 = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}, \mathbf{p}_2 = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \tag{4.20}$$

onde $\mathbf{p}_1$, $\mathbf{p}_2 \in \mathbb{R}^m$, e suas componentes a_i e b_i (i = 1, ..., m) representam valores atribuídos às variáveis de projeto, denotadas por x_1, x_2, ..., x_m, normalizadas no intervalo [0,1].

b) Gere um número aleatório u com distribuição uniforme no intervalo [0,1).

c) Atribua um número inteiro positivo para o parâmetro de entrada n. Quanto maior o valor de n, maior é a concentração de descendentes nas proximidades dos genitores.

d) Calcule b em função de u e n.

$$b = \begin{cases} (2u)^{\frac{1}{n+1}}, & \text{se } u \leq 0,5 \\ \left(\dfrac{1}{2(1-u)}\right)^{\frac{1}{n+1}}, & \text{caso contrário} \end{cases} \tag{4.21}$$

e) Calcule os novos indivíduos $\mathbf{z}_1$ e $\mathbf{z}_2$, denominados descendentes.

$$\begin{aligned} \mathbf{z}_1 &= 0,5\left[(1+b)\mathbf{p}_1 + (1-b)\mathbf{p}_2\right] \\ \mathbf{z}_2 &= 0,5\left[(1-b)\mathbf{p}_1 + (1+b)\mathbf{p}_2\right] \end{aligned} \tag{4.22}$$

O parâmetro n regula o espalhamento dos descendentes (Figura 4.7). Quanto maior o valor de n, menor é a derivada de b fixando-se u (Equação 4.21). Quanto menor a derivada, menor é o afastamento dos descendentes em relação a seus genitores. Quanto

menor o afastamento, menor é a probabilidade de se gerar descendentes inválidos, isto é, com componentes infringindo o intervalo [0,1], como ilustrado nas Figuras 4.8 e 4.9, em um exemplo considerando duas variáveis, x_1 e x_2.

Quando há mais de uma variável em jogo, os espalhamentos gerados pelo SBX são bem diferentes dos gerados pelos operadores binários (Figuras 4.5, 4.6, 4.8 e 4.9), não havendo, portanto, a referida similaridade entre esses operadores. Diferentemente dos operadores binários, o SBX sempre gera uma distribuição de descendentes colineares com os genitores, como pode-se verificar ao reescrever a Equação 4.22 na forma abaixo.

$$\begin{aligned} \mathbf{z}_1 &= 0,5(1+b)(\mathbf{p}_1-\mathbf{p}_2)+\mathbf{p}_2 \\ \mathbf{z}_2 &= 0,5(1-b)(\mathbf{p}_1-\mathbf{p}_2)+\mathbf{p}_2 \end{aligned} \tag{4.23}$$

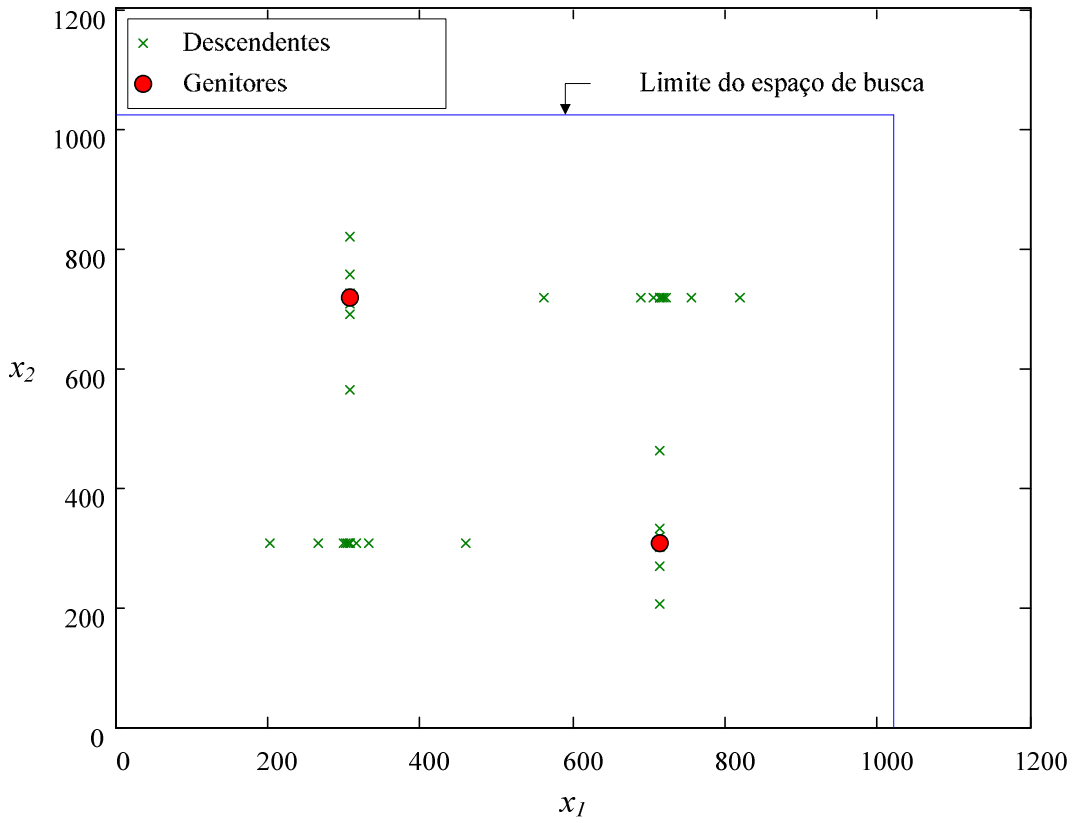


Figura 4.5 - Possibilidades de recombinação binária com corte em um ponto, dados dois genitores como exemplos.

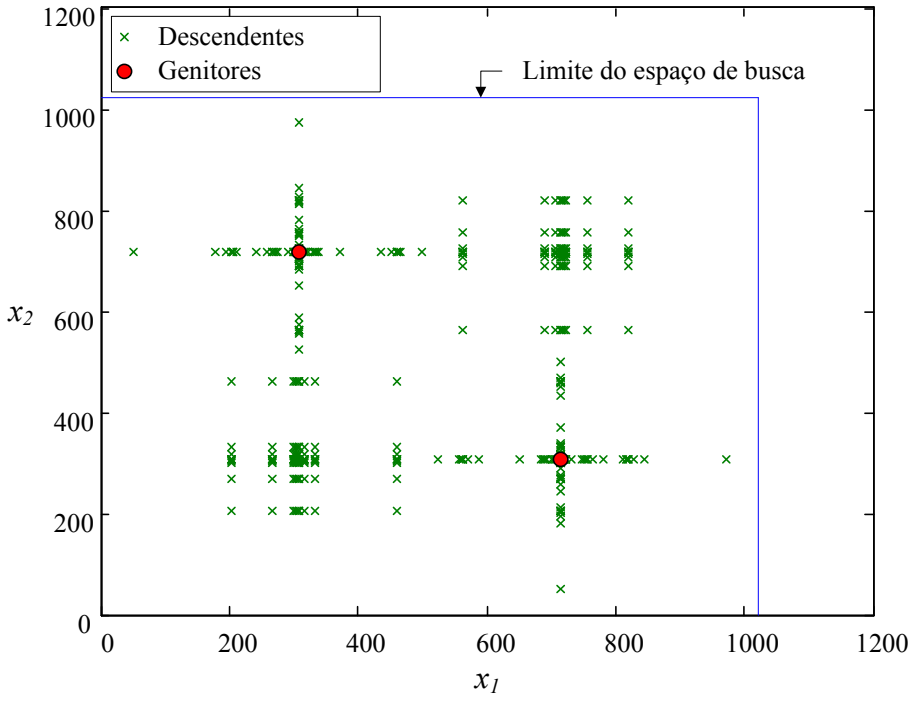


Figura 4.6 - Possibilidades de recombinação binária com corte em dois pontos, dados dois genitores como exemplos.

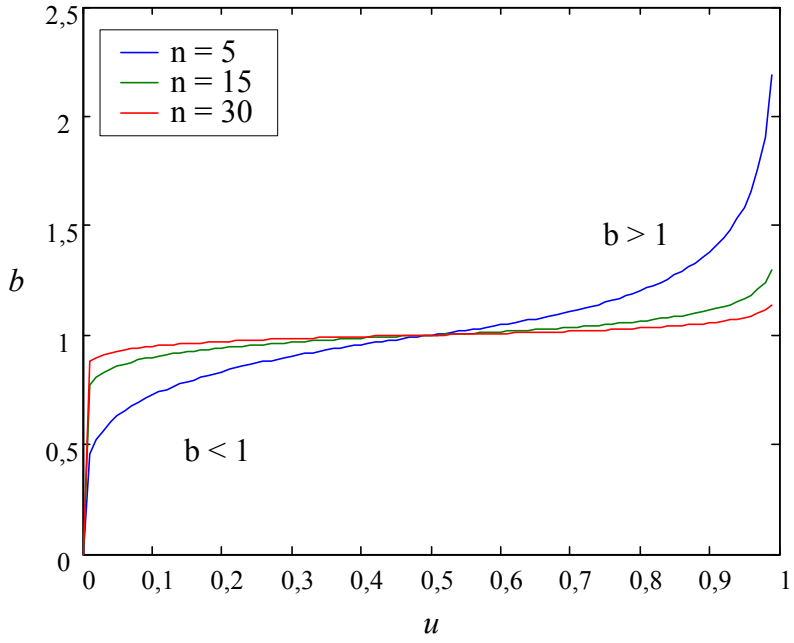


Figura 4.7 - Ajuste do operador SBX.

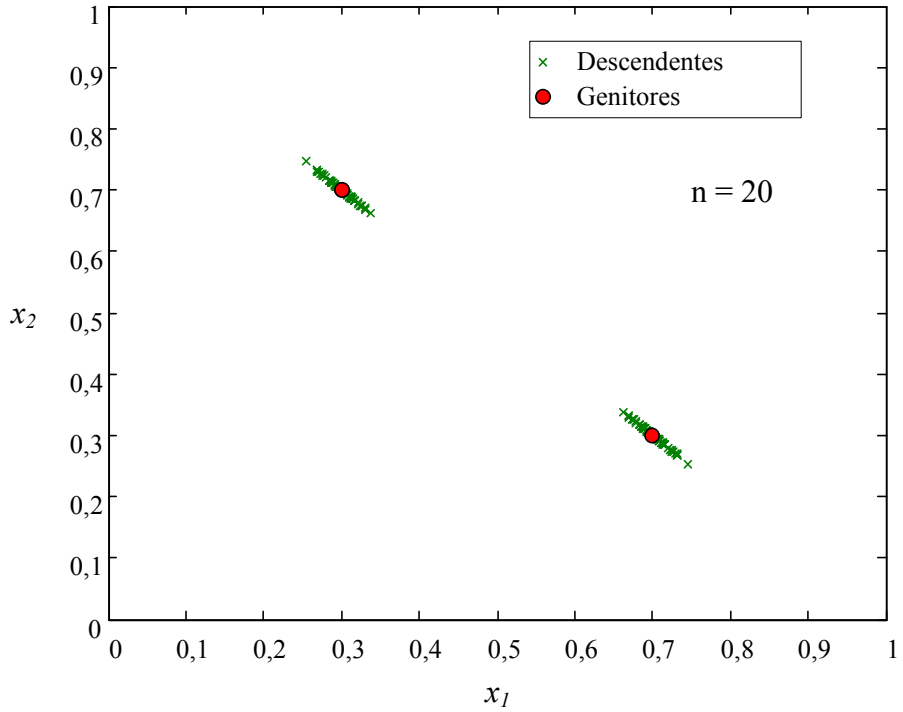


Figura 4.8 - Espalhamento do operador SBX dentro do espaço normalizado.

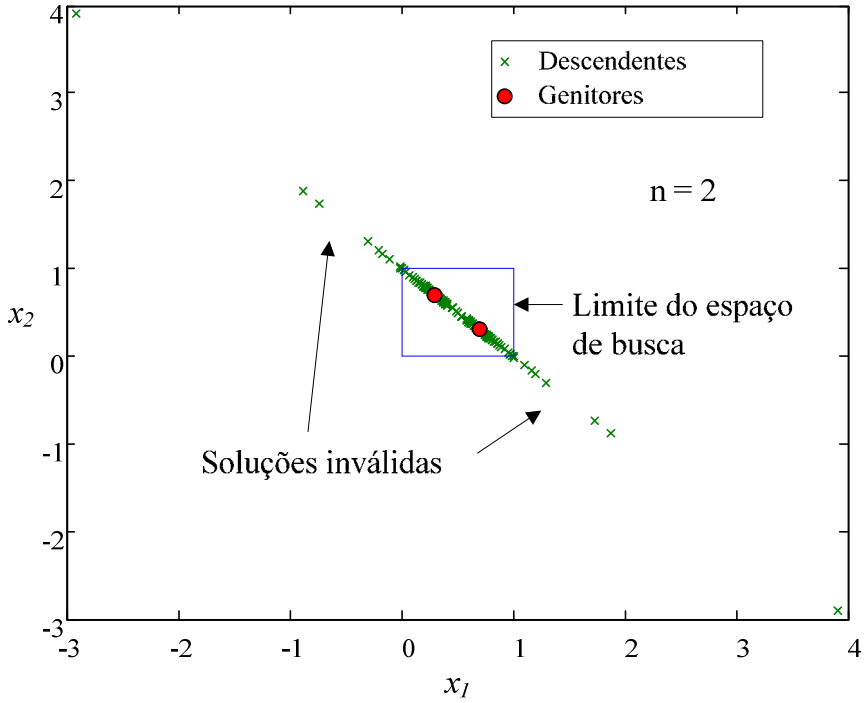


Figura 4.9 - Espalhamento do SBX com soluções inválidas.

4.2.5 Mutação uniforme de Michalewicz

A mutação uniforme de Michalewicz é um operador real de mutação, cujo procedimento é bastante simples (MICHALEWICZ, 1996). Por esse motivo, escolheu-se esse operador para ser usado junto com o SBX. O SBX requer o uso combinado com um operador de mutação a fim de equilibrar a intensificação e diversificação da busca. Segue, abaixo, o procedimento desse operador.

a) Selecione um indivíduo **p**:

$$\mathbf{p} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}, \tag{4.24}$$

onde $\mathbf{p} \in \mathbb{R}^m$, e suas componentes a_i (i = 1, ..., m) representam valores atribuídos às variáveis, denotadas por x_1, x_2, ..., x_m, normalizadas no intervalo [0,1].

b) Escolha aleatoriamente um componente k de **p**.

c) Escolha um número real aleatório q com distribuição uniforme no intervalo [0,1].

d) Gere um novo indivíduo **z** da seguinte forma:

$$\mathbf{z} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ x_k = q \\ \vdots \\ a_m \end{bmatrix}. \tag{4.25}$$

4.3 Abordagem memética baseada no NGMOGA e no M-GEO

4.3.1 Operador real de assimetria dinâmica

Propõe-se, nesta subseção, um operador real mímico do operador binário do M-GEO. Esse operador foi concebido com o objetivo de manusear diretamente variáveis reais, normalizadas no intervalo [0,1], substituindo os operadores de recombinação e mutação binários do NGMOGA e o operador binário do M-GEO. Sua principal motivação foi a busca por um operador que fosse tão eficiente quanto o SBX, que não gerasse soluções fora do espaço normalizado e que pudesse ser configurado tanto para uma busca global quanto para uma local. Trata-se, aqui, de um operador que imita a distribuição das soluções geradas pelo operador binário do M-GEO, contudo, de modo que a distribuição tenha assimetria variável, isto é, que as soluções geradas não se repitam toda vez que a mesma solução inicial é visitada.

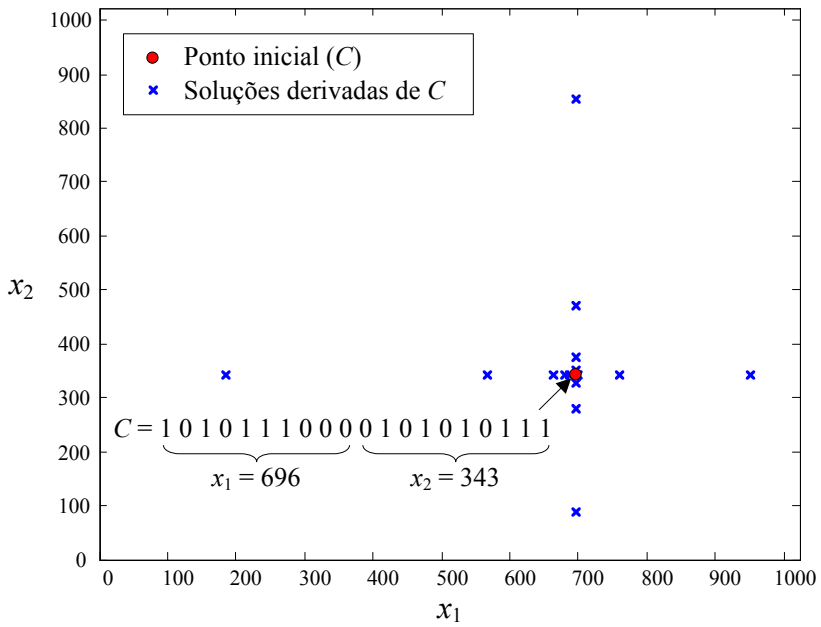


Figura 4.10 - Exemplo de soluções geradas com o operador binário do M-GEO.

Conforme discutido por Galski (2006), o operador binário sempre gera uma distribuição de soluções com assimetria estática. Suponha, por exemplo, um espaço definido por duas variáveis, x_1 e x_2, que assumem valores entre 0 e 1000 (Figura 4.10). Dado um

ponto qualquer desse espaço, por exemplo, x_1 igual a 696 e x_2 igual a 343, como mostra a Figura 4.10, o operador binário do M-GEO sempre gera as mesmas soluções em posições assimétricas em relação a esse ponto.

O operador binário do M-GEO gera uma distribuição de pontos com concentração maior nas proximidades do ponto de origem, como mostra a Figura 4.10, de modo que a intensificação da busca é sempre mais provável do que a diversificação, como geralmente ocorre nos métodos de busca global. A distribuição dos pontos assim ocorre porque a comutação de um bit de *C* (ponto de origem) adiciona ou subtrai uma potência de dois (Figura 4.11).

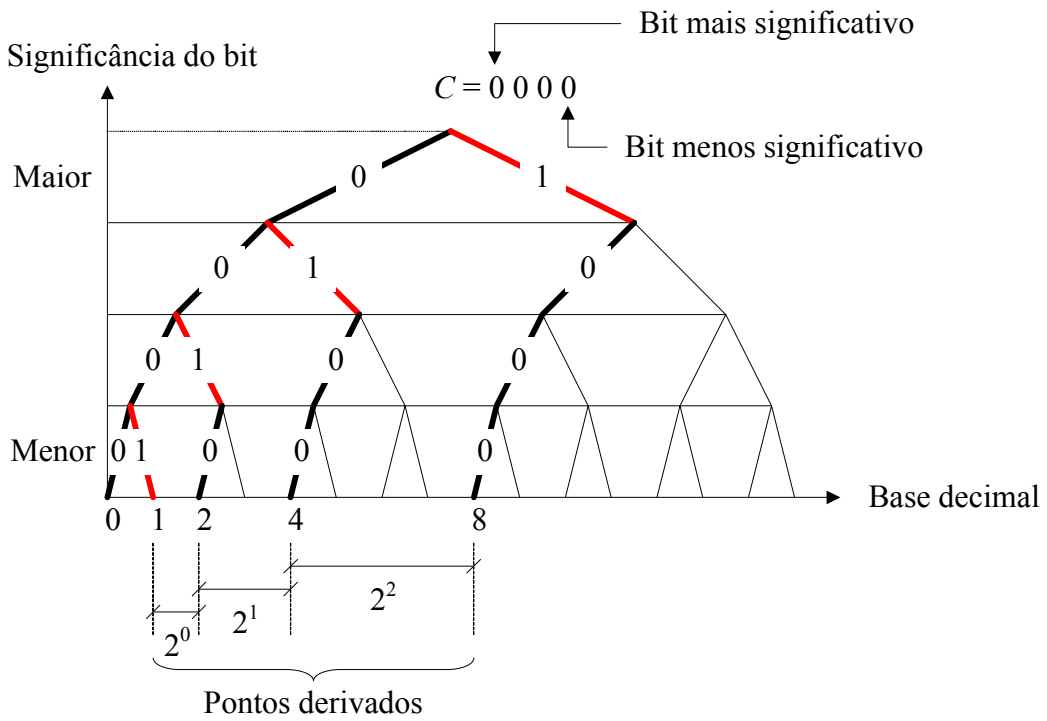


Figura 4.11 - Mapeamento[14] da comutação dos bits realizada pelo operador binário do M-GEO.

No entanto, o operador binário do M-GEO não possui um parâmetro que permita alterar o compromisso entre a intensificação e diversificação da busca. Essa limitação tinha que

[14] O mapeamento é a forma de representação gráfica da comutação dos bits apresentada por Galski (GALSKI, 2006, p. 78-79).

ser resolvida para se alcançar os objetivos deste trabalho, pois a abordagem memética somente faria sentido se o M-GEO pudesse ser configurado para realizar uma busca local após uma busca global conduzida pelo NGMOGA. Assim, surgiu a necessidade de se propor um operador real que pudesse ser configurado tanto para uma busca global quanto para uma local.

O operador real imitativo do operador binário do M-GEO requer que as variáveis de projeto sejam, em primeiro lugar, normalizadas no intervalo [0,1].

Em seguida, atribui-se valores apropriados para os parâmetros desse operador: coeficiente de mutação (c), limite inferior de mutação (k_{inf}) e limite superior de mutação (k_{sup}), onde o primeiro parâmetro é um número real e os dois últimos são inteiros.

O procedimento do operador consiste dos passos a seguir.

a) Dada uma solução inicial, escolha aleatoriamente uma variável de projeto v para ter seu valor alterado.

   Na Figura 4.10, por exemplo, x_1 e x_2 representam variáveis de projeto.

b) Em seguida, gere um inteiro k entre k_{inf} e k_{sup} e um valor $p \in \{0, 1\}$ com probabilidade uniforme.

c) Finalmente, determine o novo valor de v definido por (Figura 4.12)

$$v \leftarrow v + m\,, \tag{4.26}$$

   onde

$$m = \begin{cases} -tv, & \text{se } p = 0, \\ t(1-v), & \text{caso contrário}, \end{cases} \tag{4.27}$$

   e

$$t = c \cdot e^{-k/2}. \tag{4.28}$$

Figura 4.12 - Mutação da variável de projeto.

Quanto maior o valor atribuído ao parâmetro *c*, maior é a diversificação da busca. Quanto mais diversificada, mais orientada fica à procura de um ótimo global. Contudo, é importante estabelecer um compromisso entre a diversificação e a intensificação com o objetivo de evitar avaliações excessivas da função objetivo, no caso de a diversificação ter um peso considerável em relação à intensificação, ou então a convergência prematura do algoritmo de otimização, no caso contrário.

A Figura 4.13 ilustra como o operador real gera espalhamentos parecidos com o operador binário do M-GEO (Figura 4.10), concentrando mais pontos nas proximidades do ponto inicial *C*, mas sem repetir as soluções derivadas, no caso de uma segunda visita a esse ponto.

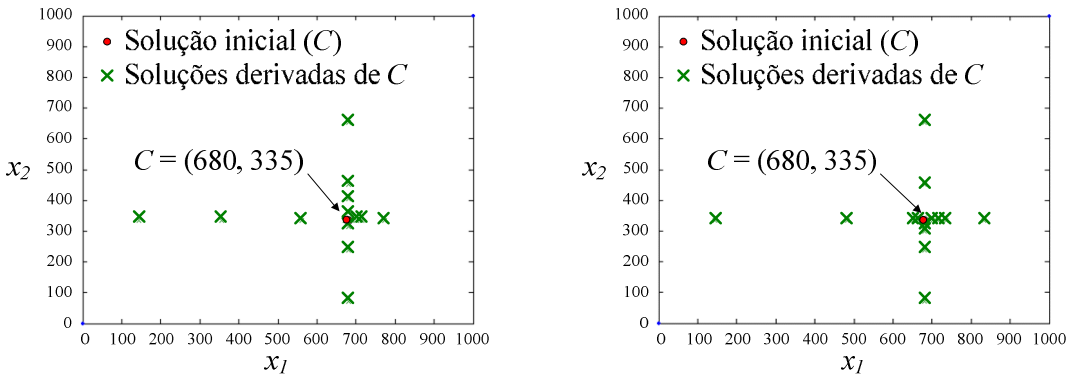


Figura 4.13 - Assimetria dinâmica causada pelo operador real imitativo do operador binário do M-GEO. (a) Exemplo 1; (b) exemplo 2, segunda visita ao ponto inicial *C*, em que ocorre a mudança na assimetria das soluções derivadas, em comparação com o Exemplo 1.

Dada a característica desse operador real, ilustrada na Figura 4.13, atribuiu-se a ele a denominação de operador real de assimetria dinâmica (ORAD). Se nas análises seguintes o ORAD gerasse resultados tão bons quanto os obtidos com o SBX e, além

disso, com a indicação de ser mais simples, robusto e eficiente, dispensaria então a necessidade de avaliar outros operadores reais.

4.3.2 Estratégia de otimização

A estratégia de otimização baseia-se na cooperação do NGMOGA com o M-GEO. Quando este estudo foi realizado, ao contrário do NGMOGA, o M-GEO não possuía um mecanismo para regular a distribuição das soluções ao longo da fronteira de Pareto. Constatou-se, então, que se algumas sementes estivessem aglutinadas e outras espalhadas o M-GEO forneceria soluções mal distribuídas ao longo da fronteira. Entretanto, se estivessem bem distribuídas nas proximidades de regiões promissoras do espaço de busca, o M-GEO forneceria uma quantidade maior de soluções não-dominadas e bem distribuídas.

Visto que, durante o estudo desses algoritmos, o NGMOGA foi capaz de encontrar uma pequena quantidade de soluções, bem distribuídas, nas proximidades dessas regiões, realizando poucas avaliações da função multiobjetivo, surgiu a ideia de usar essas soluções como sementes do M-GEO. Assim, a estratégia de otimização consiste das seguintes etapas:

a) execute a busca global com o NGMOGA;

b) use as soluções encontradas como sementes do M-GEO; e

c) execute a busca local com o M-GEO.

Para que fosse válida, esta abordagem deveria fornecer uma quantidade de soluções de Pareto maior do que a obtida com cada método sem o auxílio do outro, fixando-se a quantidade de avaliações da função multiobjetivo.

4.4 Estudo de caso

4.4.1 Descrição

O estudo de caso consistiu de três partes com o objetivo de verificar em quais condições a estratégia de otimização pode ser considerada válida.

Na primeira parte, compararam-se os operadores reais com os operadores binários, incorporando-os somente no NGMOGA. Usou-se, como função de teste, a função *F* definida na Tabela 4.1. Se o SBX e a mutação uniforme de Michalewicz fossem tão bons quanto os operadores binários de recombinação e mutação, respectivamente, esses seriam comparados com o ORAD. E, se fossem melhores do que o ORAD, seriam então adotados como substitutos dos operadores binários. Caso contrário, se o ORAD fosse tão bom quanto os operadores binários e melhor do que o SBX, junto com a mutação uniforme de Michalewicz, esse seria então adotado no lugar dos operadores binários do NGMOGA. Se nenhum dos operadores estudados fosse tão bom quanto os operadores binários, outros operadores reais seriam avaliados. Variou-se o parâmetro *n* do SBX com o objetivo de obter o melhor desempenho desse operador.

Na segunda parte, incorporou-se o ORAD no M-GEO a fim de compará-lo com a versão original desse método baseada em operação binária[15]. Usaram-se, como função de teste, as funções *F* e *G* apresentadas na Tabela 4.1.

Na terceira parte, avaliou-se a abordagem memética, comparando duas versões, uma usando o ORAD no NGMOGA e no M-GEO e outra usando os operadores originais desses algoritmos. Adotou-se novamente a função *F* como função de teste (Tabela 4.1).

O estudo consistiu em minimizar as funções *F* e *G*, usadas neste estudo porque eram relativamente simples de serem analisadas e já tinham sido empregadas nas avaliações do M-GEO realizadas por Galski (2006). Não houve interesse em considerar a função *G* na primeira e terceira parte deste estudo porque era importante que o contradomínio da

---

[15] Veja a descrição do algoritmo do GEO (página 44).

função fosse definido no $\mathbb{R}^2$ a fim de facilitar a análise gráfica da fronteira de Pareto. Nas comparações do NGMOGA, fixaram-se a população inicial e o número de gerações. Nas do M-GEO, fixaram-se as sementes.

Tabela 4.1 - Funções de teste.

Função	Variáveis	Componentes	Restrições
F (1)	x_1, x_2	$f_1 = (x_1 - 2)^2 + (x_2 - 1)^2 + 2$ $f_2 = 9x_1 - (x_2 - 1)^2$	$x_1^2 + x_2^2 - 225 \leq 0$ $x_1 - 3 \cdot x_2 + 10 \leq 0$ $-20 \leq x_i \leq 20,\ i \in \{1, 2\}$
G (2)	x_1, x_2	$g_1 = (x_1 - 2)^2 + (x_2 - 2)^2$ $g_2 = (x_1 + 2)^2 + (x_2 - 2)^2$ $g_3 = (x_1 + 2)^2 + (x_2 + 2)^2$ $g_4 = (x_1 - 2)^2 + (x_2 + 2)^2$	$-10 \leq x_i \leq 10,\ i \in \{1, 2\}$

Fontes: (1) Srinivas e Deb (1994); e
(2) Mason et al. (1998).

4.4.2 Comparação do NGMOGA com operadores reais e binários

Em primeiro lugar, compararam-se os operadores reais de recombinação e mutação, os quais são o SBX com a mutação uniforme de Michalewicz, respectivamente, com os operadores binários originalmente incorporados no NGMOGA. Após várias execuções, variando o parâmetro *n* do SBX, constatou-se que esse operador com a mutação uniforme de Michalewicz não foram capazes de prover resultados tão bons quanto os operadores binários de recombinação e mutação (Figuras 4.14, 4.15 e 4.16).

Em quase todas as execuções, os operadores binários apresentaram soluções bem distribuídas, espaçadas de forma mais regular do que as soluções obtidas com os operadores reais. As Figuras 4.14, 4.15 e 4.16 mostram que o espaçamento entre os pontos é mais uniforme nos resultados obtidos com os operadores binários.

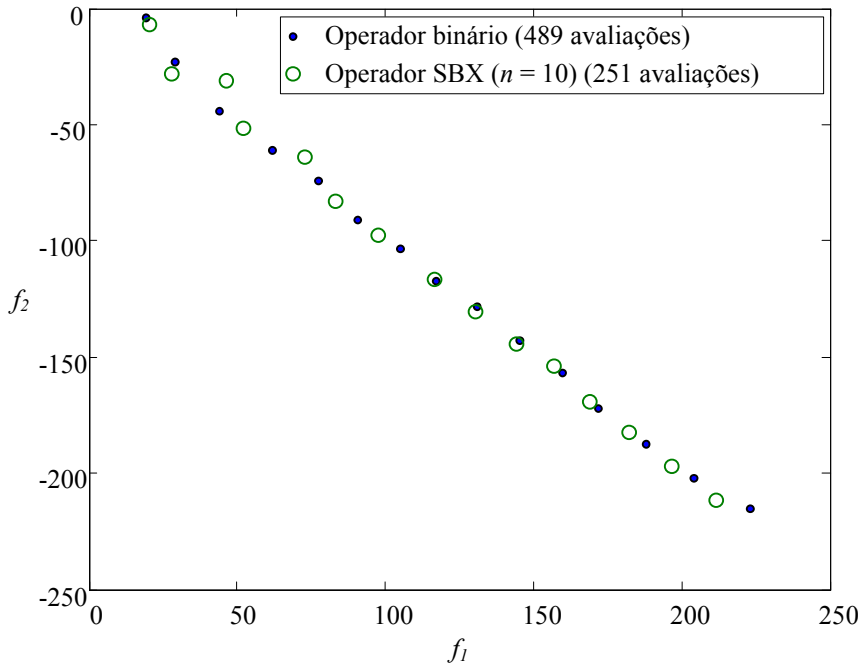


Figura 4.14 - Operador binário versus SBX (*n* igual a 10).

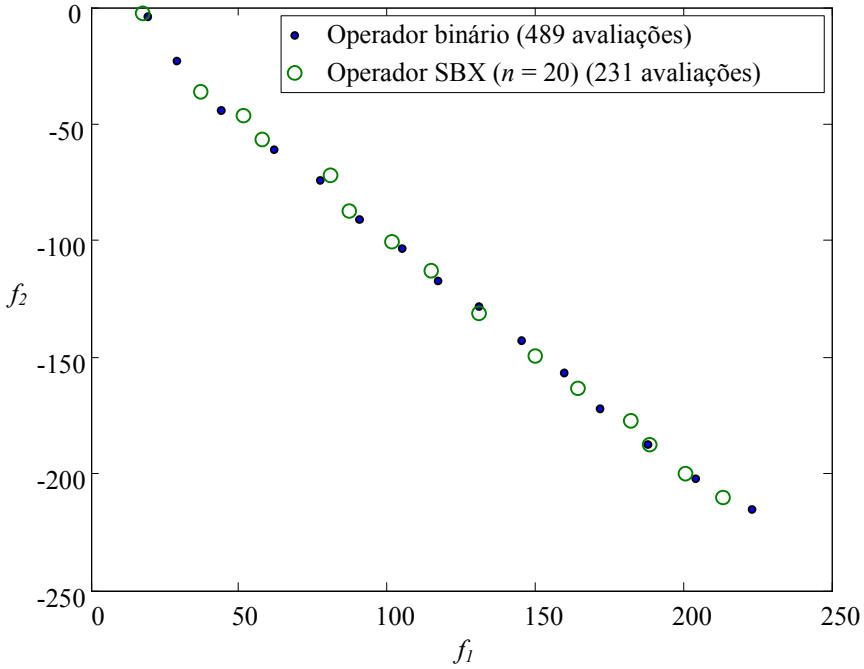


Figura 4.15 - Operador binário versus SBX (*n* igual a 20).

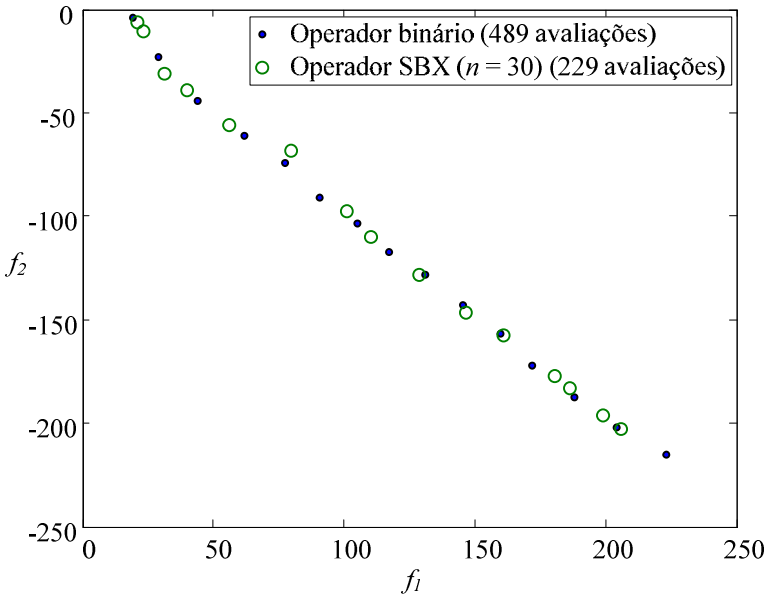


Figura 4.16 - Operador binário versus SBX (*n* igual a 30).

O espaçamento regular entre as soluções obtidas com o NGMOGA é um aspecto importante, pois essas soluções são usadas como sementes do M-GEO. Se as sementes não forem bem distribuídas nas proximidades da fronteira de Pareto, o M-GEO fará a busca em regiões muito próximas ou afastadas umas das outras, resultando em uma quantidade menor de soluções não-dominadas e regiões pouco exploradas.

O NGMOGA realizou menos avaliações da função multiobjetivo usando os operadores reais do que usando os binários (Figuras 4.14, 4.15 e 4.16). Dado que o problema com a função *F* possui restrições, há regiões de soluções inválidas no espaço de busca. Nessas condições, o SBX leva vantagem sobre os operadores binários, pois gera espalhamentos de soluções descendentes centrados somente nos genitores, o que não ocorre com os operadores binários. Assim, é provável que os operadores binários gerem mais descendentes inválidos do que os operadores reais. Quando um descendente inválido é gerado, faz-se novas tentativas, requerendo mais avaliações da função multiobjetivo. No entanto, se o SBX fosse incorporado no NGMOGA, sua vantagem seria pouco significativa para o desempenho da abordagem memética, dado que a quantidade mais significativa de avaliações, a qual impacta no desempenho dessa abordagem, seria devido à busca local realizada pelo M-GEO.

Como o SBX com a mutação uniforme de Michalewicz não proveram resultados tão bons quanto os operadores binários, não houve interesse em compará-los com o ORAD e, portanto, em adotá-los como substitutos dos operadores binários. Assim, dispensaram-se esses operadores reais e avaliou-se, em seguida, o ORAD como substituto dos operadores binários do NGMOGA.

O ORAD, quando incorporado no NGMOGA como substituto dos operadores binários, traz uma consequência embaraçosa, pois não está relacionado com o princípio genético que inspira os operadores binários de recombinação e mutação do NGMOGA e que, por conseguinte, justifica seu paradigma. No entanto, a versão do NGMOGA com o ORAD ainda pode ser classificada dentro das estratégias evolucionárias. Segundo Linden (2006), as estratégias evolucionárias foram propostas, na década de 1960, por Rechenberg e Schwefel, na Alemanha, para tratar problemas de otimização contínua. As primeiras versões dessas estratégias utilizavam apenas operadores de mutação e seleção.

O ORAD pode ser considerado, essencialmente, um operador de mutação. Quando o NGMOGA seleciona dois indivíduos para cruzamento, o operador binário de recombinação troca partes de seus genótipos e gera dois novos indivíduos. O ORAD, no entanto, não realiza essa troca. Ele apenas provoca mutações de tal forma que os novos indivíduos sejam gerados com maior probabilidade nas proximidades dos indivíduos selecionados. A Figura 4.17 ilustra a distribuição de pontos gerados pelo ORAD a partir de dois indivíduos quaisquer em um problema com duas variáveis definidas em um espaço normalizado no intervalo [0,1]. A distribuição de pontos gerados pelo ORAD é mais similar à dos operadores binários de recombinação do que à do SBX[16].

A avaliação do ORAD consistiu em verificar se o seu desempenho se equiparava ao dos operadores binários incorporados no NGMOGA. Compararam-se o número de avaliações da função multiobjetivo por solução não-dominada e a dominância entre as

[16] Compare, por exemplo, as Figuras 4.13 e 4.17, relativas à distribuição de pontos gerados pelo ORAD, com as Figuras 4.5 e 4.6, páginas 49 e 50, relativas à distribuição gerada pelos operadores binários de recombinação, e com as Figuras 4.8 e 4.9, página 51, relativas ao SBX.

soluções obtidas com o ORAD e com os operadores binários[17]. Se o ORAD apresentasse uma quantidade menor de avaliações e um percentual menor de soluções dominadas, observando a regularidade do espaçamento entre as soluções, ele seria considerado tão bom quanto ou melhor do que os operadores binários.

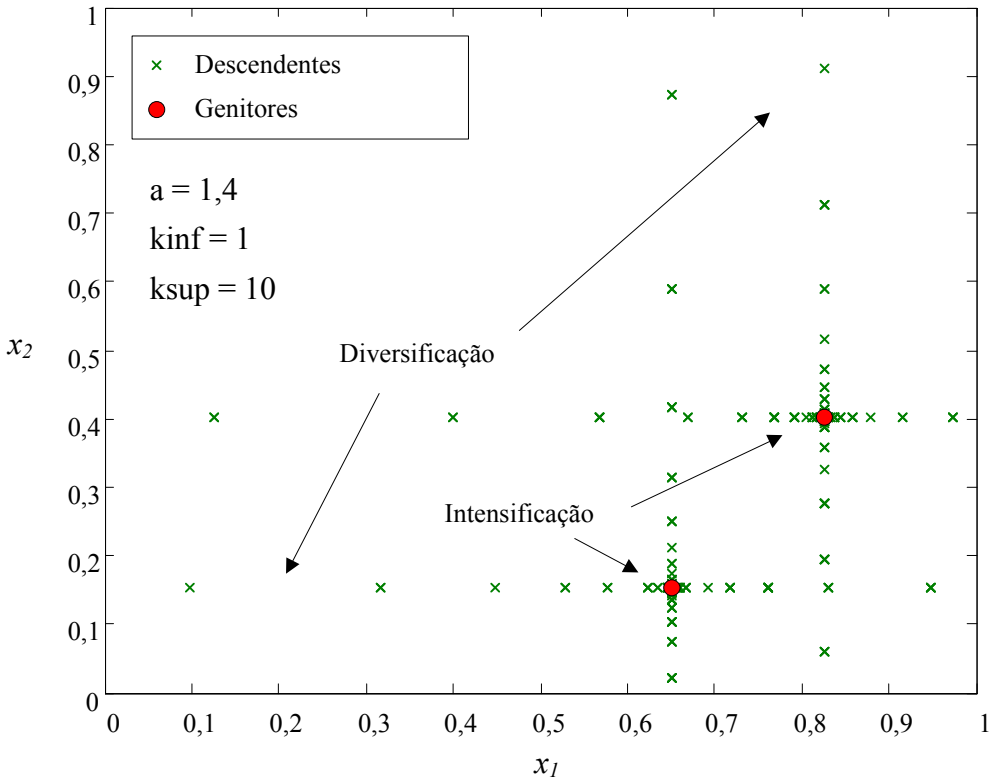


Figura 4.17 - Exemplo de soluções geradas com o ORAD.

O ORAD gerou resultados tão bons quanto os dos operadores binários. Para que a comparação fosse equilibrada, fixou-se a população inicial para ambos os casos. A Figura 4.18 apresenta o resultado da minimização da função *F* (Tabela 4.1) obtido com o ORAD e com os operadores binários incorporados no NGMOGA. A quantidade de avaliações da função multiobjetivo realizada com o ORAD foi mais próxima da realizada com os operadores binários (Figura 4.18) do que com o SBX (Figuras 4.14, 4.15 e 4.16), indicando maior similaridade do ORAD com os operadores binários.

[17] A aplicação do critério de Pareto para avaliar a dominância entre conjuntos de soluções é uma proposta apresentada por este autor, definida na Subseção 4.2.1.

Os resultados significam que o ORAD é tão bom quanto os operadores binários e, ao que tudo indica, mais robusto e eficiente do que o SBX. Mais robusto porque não gera soluções fora do espaço normalizado. Mais eficiente porque pode ser configurado tanto para uma busca global quanto para uma local, dispensando a necessidade de um operador real de mutação. Além disso, o ORAD é tão simples quanto o SBX, pois ambos realizam praticamente a mesma quantidade de operações, levando-se em conta a seleção e geração de dois indivíduos. O SBX executa dezoito operações aritméticas, uma lógica e uma escolha aleatória com distribuição uniforme, totalizando vinte operações. Para cada dois indivíduos selecionados, o ORAD executa doze operações aritméticas, duas lógicas e seis escolhas aleatórias com distribuição uniforme, totalizando também vinte operações. Observa-se, entretanto, que o SBX possui a vantagem de realizar menos escolhas aleatórias do que o ORAD.

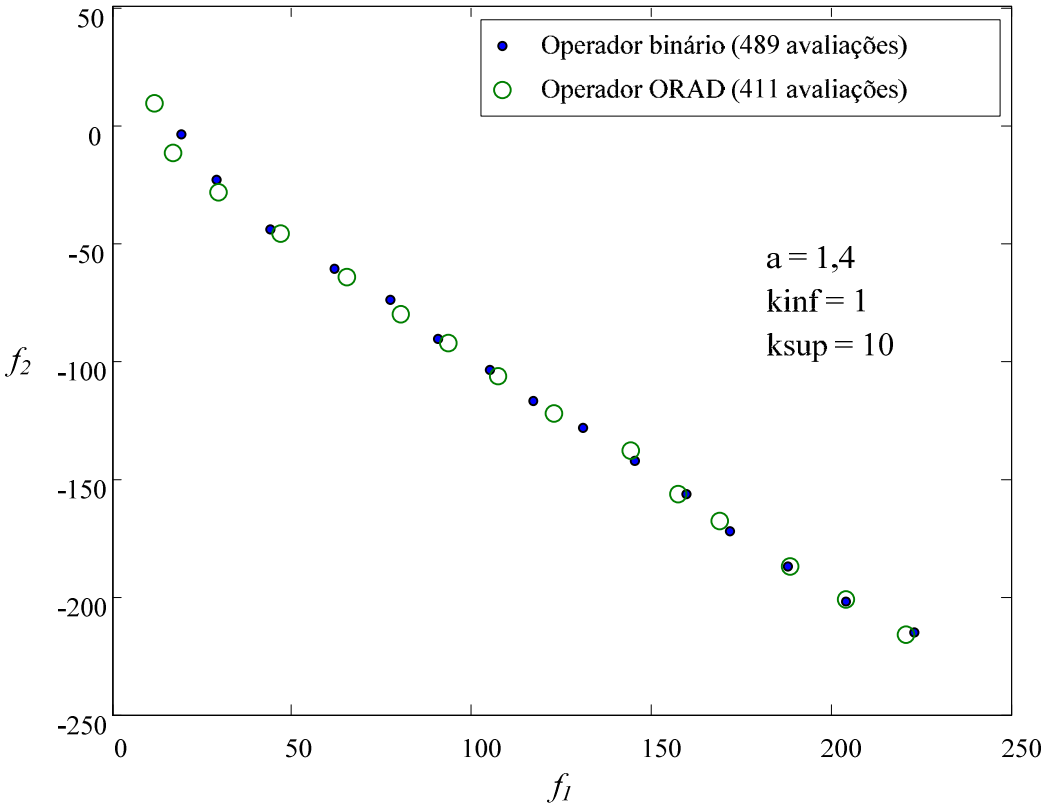


Figura 4.18 - ORAD versus operadores binários do NGMOGA.

4.4.3 Comparação do M-GEO com o ORAD e o operador binário

Compararam-se os resultados de duas versões do M-GEO, a primeira com o ORAD e a segunda com o seu operador binário original. Ambas as versões foram igualmente definidas conforme a Tabela 4.2 e, em seguida, executadas com o mesmo conjunto de sementes. Para comparar os operadores, aplicou-se o critério de dominância entre conjuntos de soluções. A avaliação consistiu em minimizar as funções *F* e *G* definidas na Tabela 4.1. Para ambas as funções, Galski realizou em torno de 50000 avaliações da função multiobjetivo, definindo sequências de 16 bits para codificar cada variável, 50 sementes e certos valores de criticalidade $\tau = \{0{,}5, 2, 4, 8\}$ (GALSKI, 2006, p. 214).

O melhor resultado de *F*, encontrado por Galski, foi de 1395 soluções não-dominadas, para τ igual a 2, alcançando, portanto, uma quantidade de 35,8 avaliações de *F* por solução não-dominada (GALSKI, 2006, p. 217). Quanto à função *G*, Galski encontrou 13913 soluções, também para τ igual a 2, resultando em 3,6 avaliações de *G* por solução não-dominada (GALSKI, 2006, p. 221).

Tabela 4.2 - Parâmetros do M-GEO.

Parâmetro	Valor
Criticalidade (τ)	3
Número de sementes	5
Número de torneios	50
Soluções visitadas por torneio	20

A quantidade *Q* de avaliações da função multiobjetivo executada pelo M-GEO, sem levar em conta quantidades adicionais para corrigir soluções inválidas, é dada por

$$Q = S \cdot T \cdot L\,, \tag{4.29}$$

onde *S* é o número de sementes, *T*, o número de torneios, e *L*, o comprimento da sequência binária que representa uma solução do problema. Na versão do M-GEO com o ORAD, o parâmetro *L* corresponde ao número de soluções visitadas por torneio (Tabela 4.2). Substituindo os valores da Tabela 4.2 na Equação 4.29, obtém-se um valor

aproximado da quantidade de avaliações das funções *F* e *G* realizadas neste teste. Realizaram-se, portanto, em torno de 5000 avaliações, dez vezes menos do que a quantidade realizada por Galski.

A Tabela 4.3 apresenta o desempenho das duas versões do M-GEO para cada função de teste. Não houve a necessidade de observar a distribuição das soluções ao longo da fronteira de Pareto, uma vez que, para garantir que a distribuição fosse satisfatória, fixaram-se sementes bem espaçadas entre si.

Os resultados indicam que o ORAD é capaz de prover um desempenho melhor do que o operador binário do M-GEO em problemas contínuos, realizando menos avaliações da função multiobjetivo e gerando mais soluções não-dominadas (Tabela 4.3). As Figuras 4.20 e 4.21 apresentam os resultados da minimização da função *G* (Tabela 4.1), com o operador binário e com o ORAD, respectivamente, considerando-se as mesmas sementes para ambos os casos (Figura 4.19). Nota-se que as soluções de Pareto da função *G*, obtidas com o ORAD, preenchem o conjunto de Pareto, que consiste dos pontos compreendidos na região definida por x_1 e $x_2 \in$ [-2,2], de forma mais uniforme do que as soluções obtidas com o operador binário, indicando que o resultado do ORAD é melhor (Figuras 4.20 e 4.21).

Tabela 4.3 - Desempenho das versões do M-GEO.

Resultados	Função *F*		Função *G*	
	Op. binário	ORAD	Op. binário	ORAD
Soluções não-dominadas	376	508	1072	2024
Avaliações da função multiobjetivo	5043	5005	5005	5005
Avaliações por solução não-dominada	13,4	9,8	4,7	2,5
Soluções dominadas pelas soluções obtidas com o uso do outro operador	147	71	50	8

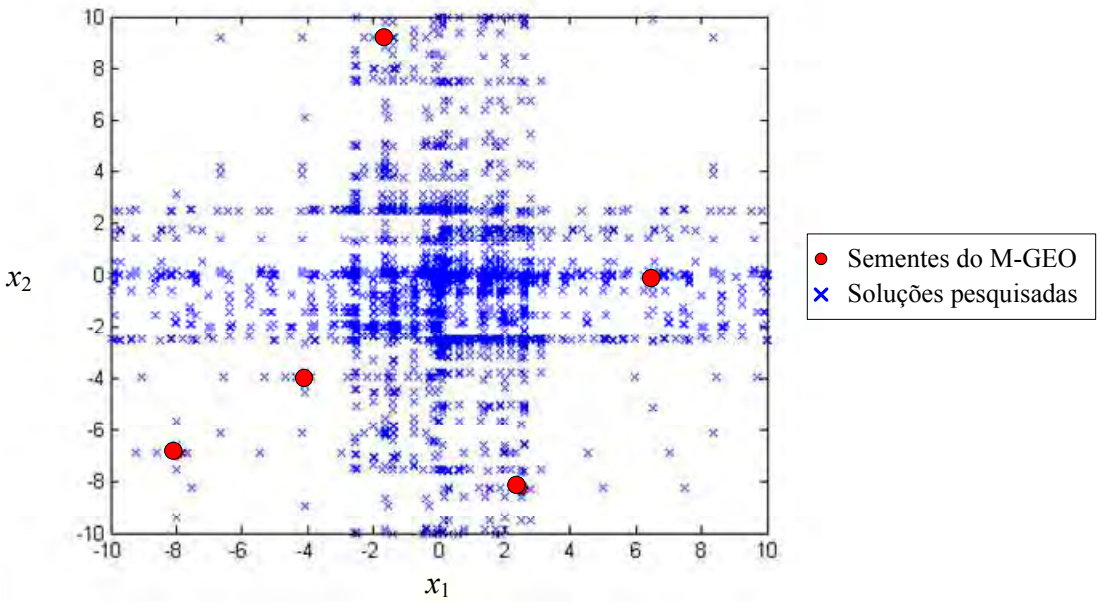


Figura 4.19 - Soluções pesquisadas no domínio da função *G* pelo M-GEO com o operador binário original.

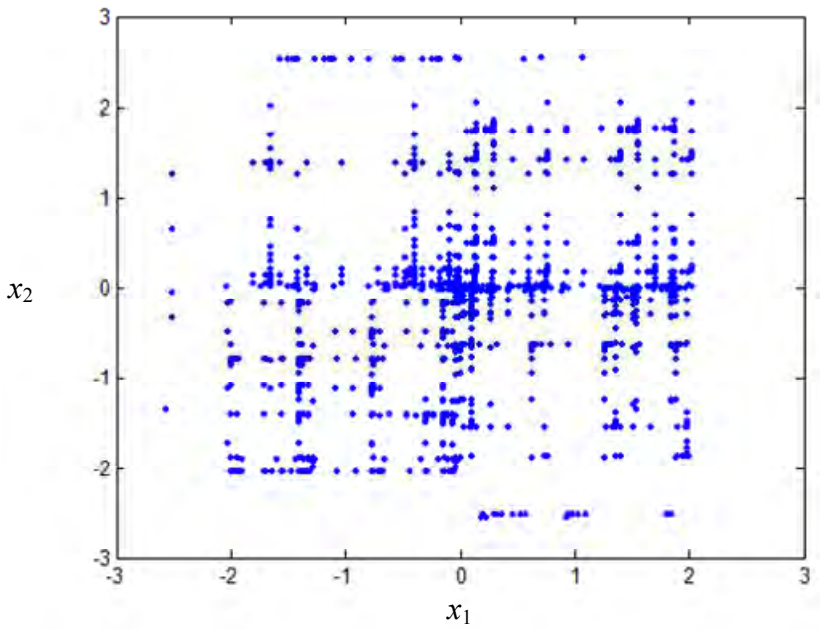


Figura 4.20 - Soluções de Pareto da função *G* obtidas com o M-GEO usando o operador binário.

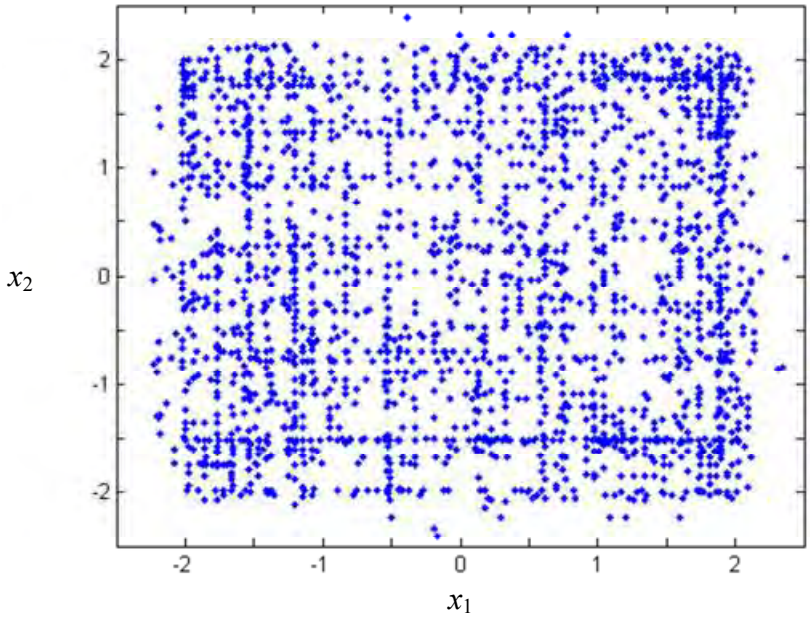


Figura 4.21 - Soluções de Pareto da função *G* obtidas com o M-GEO usando o operador ORAD.

4.4.4 Comparação da abordagem memética com o ORAD e operadores binários

A abordagem memética proposta baseia-se na cooperação do algoritmo genético multiobjetivo não-geracional (NGMOGA) com o algoritmo multiobjetivo da otimização extrema generalizada (M-GEO), onde o primeiro realiza uma busca global e o segundo, executado após a finalização do primeiro, realiza uma busca local.

Esta subseção apresenta a avaliação de duas versões da abordagem memética, uma com o operador real de assimetria dinâmica (ORAD) e outra com os operadores binários originalmente incorporados no NGMOGA e no M-GEO. A avaliação consistiu da minimização da função *F* (Tabela 4.1).

Em primeiro lugar, avaliou-se a abordagem memética com os operadores binários, realizando duas tentativas com o objetivo de determinar o número mais adequado de sementes e torneios. Consideraram-se, inicialmente, 5 sementes, 50 torneios e demais parâmetros do M-GEO com os valores estabelecidos na Tabela 4.2. No entanto, nessa

primeira tentativa, após 10 execuções, não se constatou uma vantagem significativa no uso da abordagem memética.

Apesar de, com o auxílio do NGMOGA, as sementes terem sido posicionadas com espaçamento regular ao longo da fronteira de Pareto (Figura 4.22), em média, 360 soluções não-dominadas foram encontradas, contendo, em média, apenas 100 soluções a mais do que o obtido com o M-GEO sem o auxílio do NGMOGA, fixando-se o número de avaliações da função multiobjetivo. O NGMOGA realizou, em média, 300 avaliações da função multiobjetivo a fim de posicionar as sementes do M-GEO em regiões promissoras do espaço de busca. No total, realizou-se, em média, 5300 avaliações da função multiobjetivo. Esse resultado indica que considerar poucas sementes e muitos torneios não favorece o desempenho da abordagem memética.

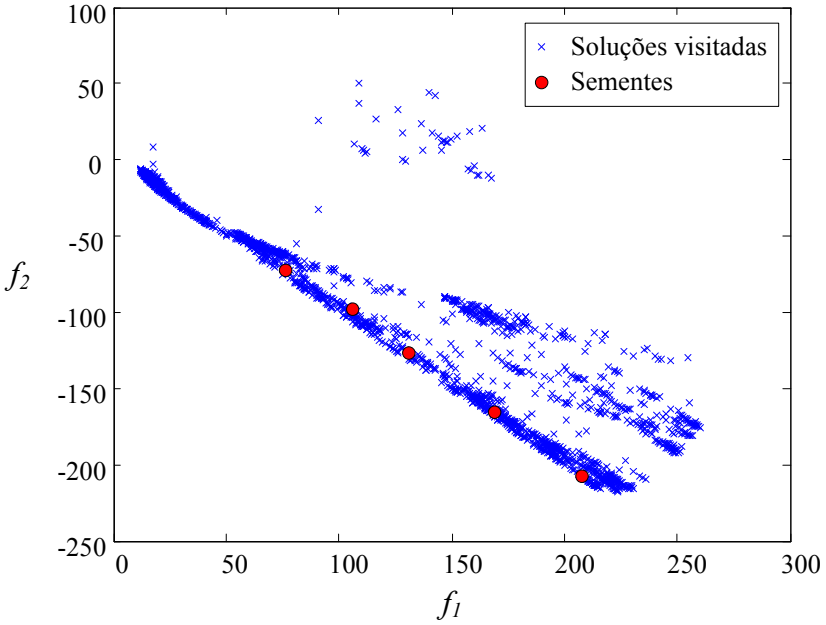


Figura 4.22 - Uma execução da busca do M-GEO após posicionamento das sementes.

Resultados expressivos foram alcançados ao reduzir o número de torneios e aumentar o número de sementes. A redução do número de torneios se justifica pelo fato de as sementes já estarem posicionadas nas proximidades da fronteira de Pareto. Nessas condições, uma quantidade maior de torneios é desnecessária, pois não melhora os resultados. Ao invés disso, piora o desempenho da abordagem memética, pois causa

mais avaliações da função multiobjetivo. O aumento do número de sementes se justifica pelo fato de cobrirem uma extensão maior da fronteira de Pareto, fazendo com que o M-GEO encontre uma quantidade maior de soluções não-dominadas.

Na segunda tentativa, consideraram-se 15 sementes e 10 torneios, e demais parâmetros conforme a Tabela 4.2 (Figura 4.24). Na primeira execução, obtiveram-se 341 soluções não-dominadas, quantidade muito próxima da obtida na primeira tentativa, realizando, no total, 3578 avaliações da função multiobjetivo, 1671 avaliações a menos do que foi realizado na primeira tentativa. O NGMOGA realizou 578 avaliações da função multiobjetivo. Executou-se o NGMOGA com 15 indivíduos, número de sementes do M-GEO, e 60 gerações, a fim de posicionar os indivíduos nas proximidades da fronteira de Pareto (Figura 4.23).

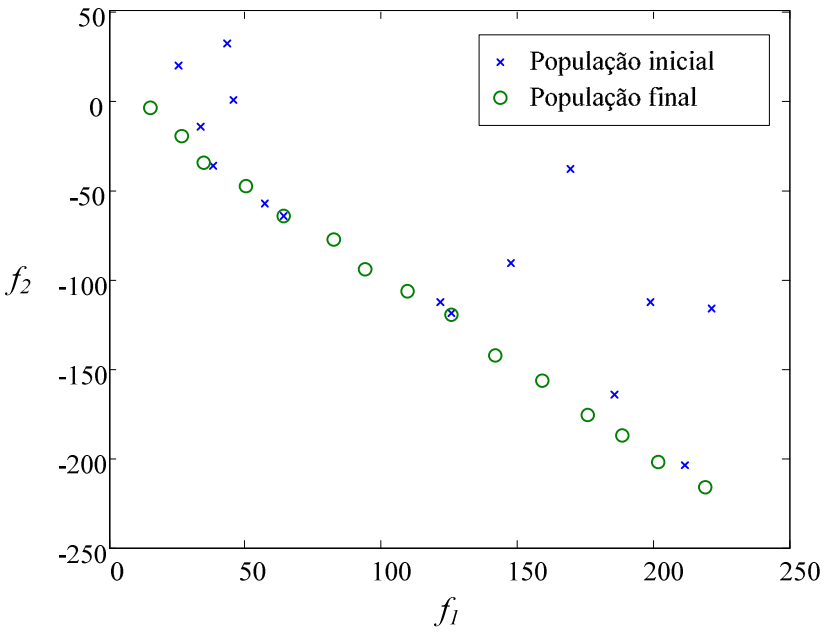


Figura 4.23 - Posicionamento das sementes do M-GEO com o auxílio do NGMOGA.

Nessa segunda tentativa, a abordagem memética teve um desempenho melhor. O posicionamento das sementes nas proximidades da fronteira de Pareto (Figura 4.25) fez com que o M-GEO encontrasse aproximadamente a mesma quantidade de soluções não-dominadas, realizando menos avaliações da função multiobjetivo.

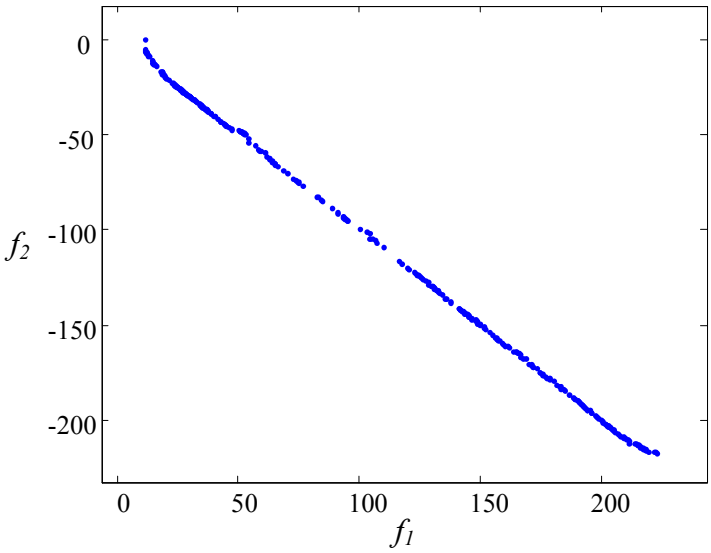


Figura 4.24 - Resultado final com os operadores binários.

Se o NGMOGA não tivesse sido utilizado para posicionar as sementes em regiões promissoras do espaço de busca, o desempenho do M-GEO teria sido certamente pior. Para verificar que o desempenho da abordagem memética era superior ao do NGMOGA, avaliou-se a dominância entre as soluções obtidas com esses algoritmos.

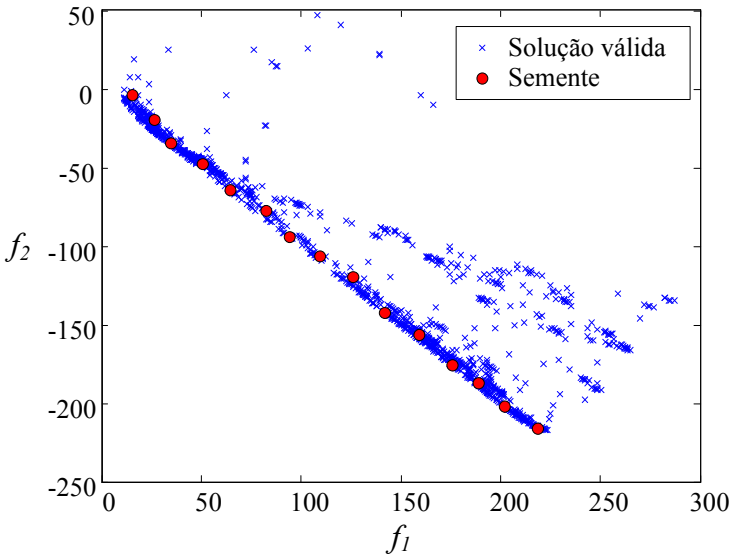


Figura 4.25 - Distribuição das soluções investigadas na fronteira de Pareto.

O NGMOGA foi então executado, isoladamente, com 341 indivíduos e 600 gerações, resultando em 3747 avaliações da função multiobjetivo, 169 avaliações a mais do que foi realizado pela abordagem memética (Figura 4.26).

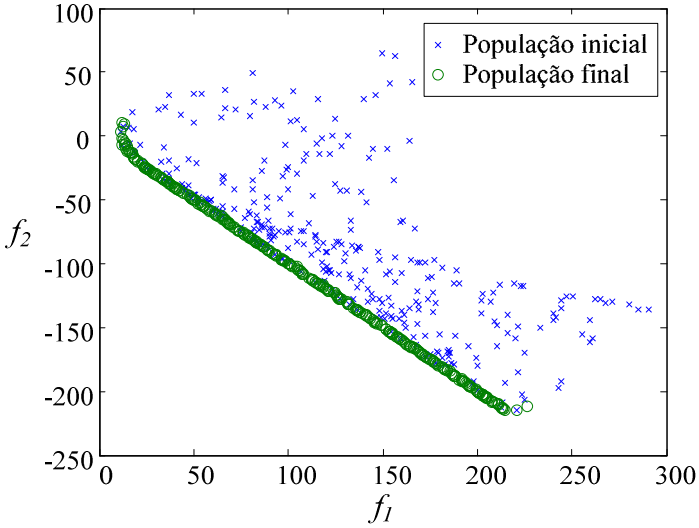


Figura 4.26 - Resultado obtido com o NGMOGA.

O conjunto de soluções obtido com a abordagem memética dominou 99 soluções do conjunto obtido com o NGMOGA. Enquanto, este dominou 63 soluções da abordagem memética. Esses resultados indicam, portanto, que a abordagem memética é capaz de encontrar soluções melhores do que as obtidas com o NGMOGA e o M-GEO, realizando menos avaliações da função multiobjetivo.

Em seguida, avaliou-se a abordagem memética com o ORAD, última etapa deste estudo de caso. Adaptou-se o ORAD ao M-GEO ajustando-o para uma busca local. A Figura 4.27 ilustra o espalhamento causado pelo ORAD adaptado ao M-GEO em um espaço definido por duas variáveis, x_1 e x_2.

Fez-se com que o ORAD gerasse um espalhamento de soluções com mutações em mais de uma variável, permitindo que elas ocorressem em qualquer direção no entorno da solução inicial (Figura 4.27). Compararam-se duas versões da abordagem memética, a primeira com o ORAD e a segunda com os operadores binários, fixando-se a população

inicial. As duas versões posicionaram as sementes do M-GEO com espaçamento regular e nas proximidades da fronteira de Pareto (Figura 4.28).

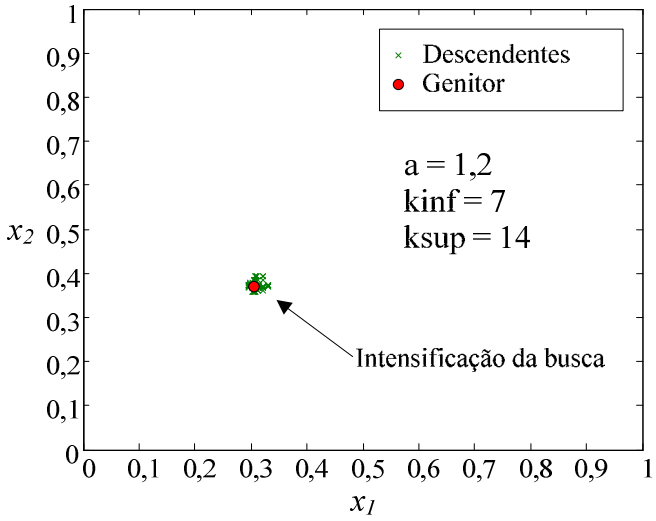


Figura 4.27 - Espalhamento causado pelo ORAD.

O NGMOGA, com o ORAD, executou 735 avaliações da função multiobjetivo, 157 avaliações a mais do que o obtido com os operadores binários. Contudo, este acréscimo foi compensado pelo M-GEO, que obteve mais soluções, realizando uma busca local. Por fim, o desempenho da abordagem memética foi melhor com o ORAD (Tabela 4.4):

a) realizou menos avaliações por solução não-dominada (Tabela 4.4);

b) teve um percentual menor de soluções dominadas (Tabela 4.4); e

c) apresentou uma distribuição mais uniforme ao longo da fronteira de Pareto (Figuras 4.24 e 4.29).

Tabela 4.4 - Análise comparativa.

Resultados da abordagem memética	Operador binário	ORAD
Avaliações da função multiobjetivo por solução não-dominada	10,5	3,7
Percentual de soluções dominadas pelo conjunto de soluções obtido com o outro operador (%)	33,4	10,8

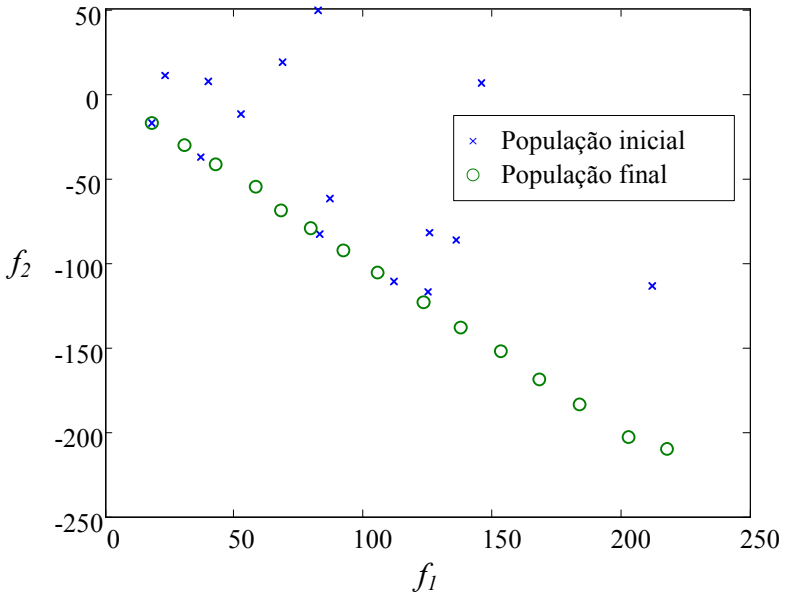


Figura 4.28 - Posicionamento das sementes do M-GEO.

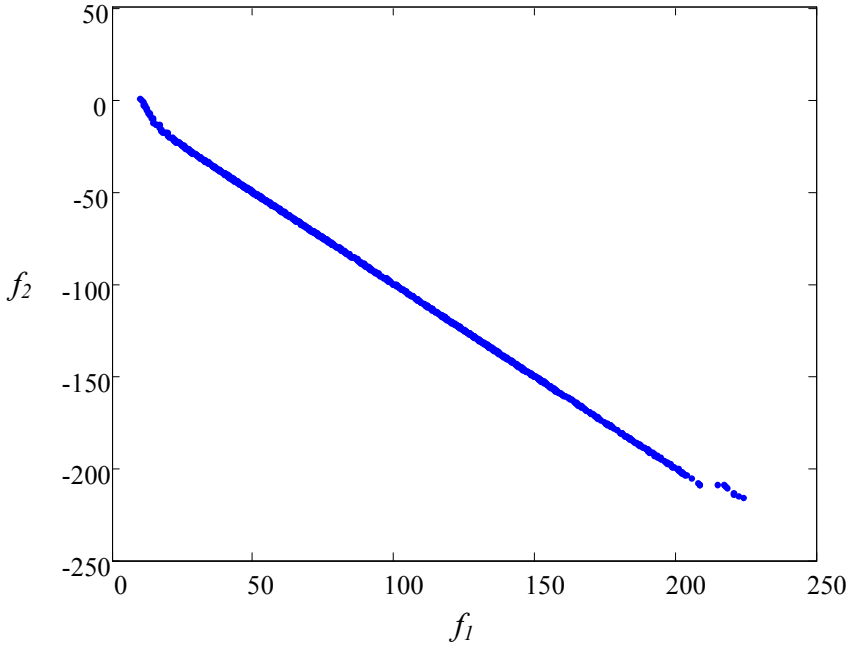


Figura 4.29 - Resultado final com o ORAD.

A abordagem memética com o operador real ORAD mostrou-se, portanto, válida para uso em problemas contínuos. Em trabalhos futuros, essa abordagem poderá ser incorporada no MDO-SONDA, como ilustrado na Figura 4.30.

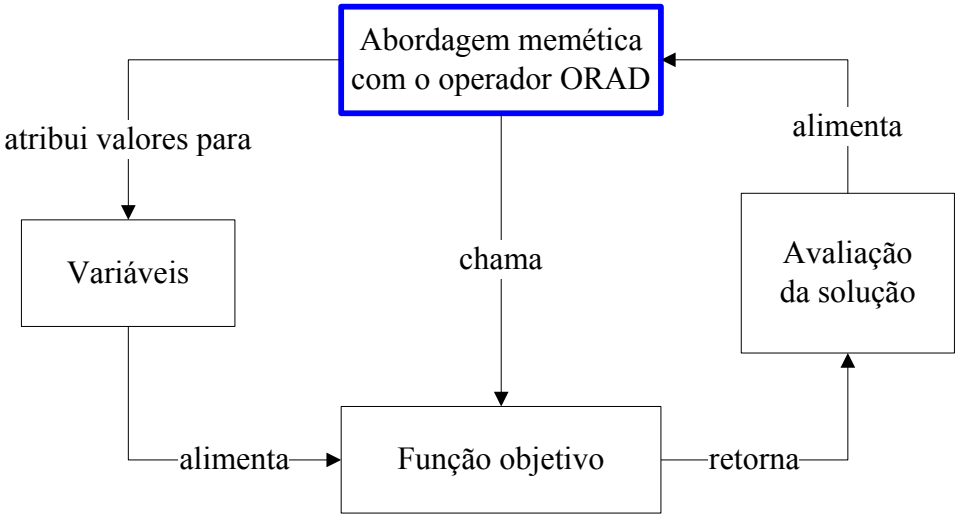


Figura 4.30 - Arquitetura do MDO-SONDA com a abordagem memética substituindo o algoritmo de otimização como mostrado na Figura 3.8.

5 AVALIAÇÃO DA EFICIÊNCIA DE MODELOS SUBSTITUTOS

5.1 Introdução

Em otimização, os modelos substitutos atuam como funções de aproximação, no lugar da função objetivo original, computacionalmente dispendiosa, com o objetivo de reduzir significativamente o número de avaliações desta e, consequentemente, o tempo de execução da otimização, mas sem prejudicar a qualidade dos resultados.

A otimização de foguetes de sondagem requer, por exemplo, simulações para avaliar a estabilidade do foguete em função da geometria de suas empenas. A busca pela melhor geometria é um processo exaustivo de tentativas e ajustes, cujos objetivos conflitantes são a estabilidade e o desempenho do foguete. Ao beneficiar a estabilidade aumentando a área da empena, tem-se como consequência o aumento do arrasto, o que afeta negativamente o desempenho do foguete. Uma única avaliação da função multiobjetivo envolve interações entre cálculos de aerodinâmica e trajetória, podendo levar mais do que 10 s em uma máquina 3.0 GHz Dual Core. Como resultado, a optimização torna-se demorada, uma vez que requer centenas de avaliações da função multiobjetivo.

A despeito de ser possível o emprego da computação paralela neste caso, o método abordado neste trabalho para contornar esta dificuldade foi o da otimização assistida por modelos substitutos. Este capítulo limita-se ao estudo das redes neurais como modelos substitutos. A rede perceptron de múltiplas camadas (MLP) e a rede de função de base radial (RBFN) têm sido amplamente empregadas na construção de modelos substitutos em otimização de projetos de engenharia (ONG et al., 2005). O auxílio à otimização multiobjetivo das empenas do VS-40, abordada mais tarde no Capítulo 3, motivou esta investigação subjacente, objetivando a escolha de um modelo substituto baseado em redes neurais, a fim de tornar o processo de otimização menos dispendioso.

Este trabalho contribui com um procedimento para estimar a eficiência dos modelos substitutos, em termos da quantidade de avaliações da função objetivo, considerando as características do método de optimização adotado. Dado que o gradiente da função

multiobjetivo, a qual avalia a forma da empena do foguete, é difícil de obter, emprega-se, neste estudo, o algoritmo genético multiobjetivo não-geracional (NGMOGA)[18].

Para investigar a eficiência dos modelos substitutos, três funções de teste disponíveis na literatura foram consideradas. Para cada função de teste, executou-se o procedimento a seguir. Primeiro, diferentes configurações de cada modelo substituto foram selecionadas e avaliadas. Escolheu-se a configuração com menor erro de generalização. Adotou-se o método *latin hypercube sampling* (LHS), apresentado por Tang (1993), a fim de gerar o conjunto de dados iniciais para estimação, validação e teste dos modelos substitutos. Em seguida, executou-se a otimização assistida por modelo substituto.

Essa abordagem consistiu de iterações de busca por soluções e atualização do modelo. A cada iteração, as soluções de Pareto encontradas pelo algoritmo genético com o auxílio do modelo substituto eram avaliadas pela função objetivo original e adicionadas ao conjunto de dados. Em seguida, reconstruía-se o modelo substituto com base no conjunto de dados. Repetiu-se, certo número de vezes, o processo de busca por soluções de Pareto, atualização do banco de dados e reconstrução do modelo substituto, com o objetivo de aumentar a eficiência da otimização assistida por estes modelos.

Finalmente, comparou-se a eficiência da MLP e da RBFN como modelos substitutos.

Este capítulo está organizado como segue. Primeiro, apresenta-se a fundamentação teórica deste capítulo, onde abordam-se a técnica de otimização assistida por modelo substitutos, a rede perceptron de múltiplas camadas, a rede de função de base radial, e o método de amostragem LHS. Em seguida, descreve-se o procedimento proposto por este autor para avaliar a eficiência de modelos substitutos. Por último, discute-se, em um estudo de caso com três funções de teste, a aplicação deste procedimento para comparar a eficiência da MLP e da RBFN.

---

[18] Este método de otimização está descrito na Subseção 4.2.2.

5.2 Fundamentação teórica

5.2.1 Otimização Assistida por Modelos Substitutos

O modelo substituto é uma função construída para aproximar outra função, cujo custo computacional é elevado, com o objetivo de melhorar a eficiência das análises que requerem quantidades significativas de avaliações desta função (ONG et al., 2005). Isto é alcançado substituindo-se a função original pelo modelo substituto durante uma quantidade significativa de vezes em que a avaliação dessa função é requerida. No contexto da optimização, os modelos substitutos são frequentemente empregados para substituir o cálculo realizado por programas de computador, os quais não são originalmente planejados para serem chamados, centenas de vezes, por uma rotina de otimização. Seja a função objetivo expressa por

$$y = f\left(x\right), \tag{5.1}$$

onde x representa as variáveis de projeto. Seja o modelo substituto uma aproximação de y expresso por

$$\hat{y} = \hat{f}\left(x\right). \tag{5.2}$$

O erro de aproximação ε é descrito por

$$\varepsilon = y - \hat{y}. \tag{5.3}$$

O desafio deste método é conceber modelos substitutos de fácil implementação, com baixo custo computacional, robustos, com ε mínimo, e que não necessitem de muitas avaliações da função objetivo para serem construídos.

Um modelo substituto bastante popular é a regressão por mínimos quadrados usando polinômios de baixa ordem, também conhecido como métodos de superfície de resposta (*response surface methods* – RSM) (ONG et al., 2005). No entanto, o custo computacional de geração do RSM aumenta significativamente com o número de variáveis de projeto e, usando polinômios de baixa ordem, esse método dificilmente

fornece uma aproximação global satisfatória ao longo de todo o espaço de busca definido sobre o domínio da função objetivo. Um método com origem na geoestatística denominado *kriging* tem sido bastante aplicado como função de aproximação global. A MLP e a RBFN também têm sido bastante utilizadas nesse sentido a fim de auxiliar a optimização de funções objetivo com custo elevado (ONG et al., 2005). A escolha de um modelo substituto simples, rápido e robusto baseado em redes neurais foi um dos objetivos deste trabalho. Neste estudo, empregou-se um procedimento de busca e atualização para executar a otimização assistida por modelo substituto. Entre a função objetivo original e o algoritmo de otimização, inseriu-se um modelo substituto, com o objetivo de absorver uma quantidade significativa das avaliações da função objetivo requeridas pelo algoritmo de otimização.

O processo de geração do modelo substituto consiste de um problema de interpolação, onde o modelo substituto deve aproximar-se de amostras de avaliações da função objetivo. Para construir o modelo substituto, amostras, uniformemente distribuída sobre o espaço de busca, foram escolhidas aleatoriamente, utilizando a técnica de amostragem LHS (TANG, 1993). Realizaram-se três iterações de busca por soluções e atualização do modelo substituto, denotadas por ciclos de otimização. Após cada ciclo, guardava-se as soluções encontradas pelo algoritmo de otimização para serem usadas como amostras no ciclo seguinte.

A Tabela 5.1 apresenta os valores da probabilidade de substituição adotada neste estudo para cada ciclo de otimização, ou seja, a probabilidade de a função objetivo ser substituída pelo modelo substituto durante o processo de otimização.

Tabela 5.1 - Probabilidade de substituição.

Ciclo de otimização	Probabilidade de substituição(1)
1º	0,94
2º	0,96
3º	1

Nota: (1) os valores foram determinados com base em tentativas e ajustes.

No final de cada ciclo de otimização, as soluções obtidas pelo algoritmo de otimização são adicionadas ao conjunto de amostras. Dessa forma, o conjunto de amostras é atualizado progressivamente a cada ciclo de otimização com soluções cada vez mais próximas das regiões promissoras do espaço de busca. O processo de busca e atualização progressivas é uma tentativa de melhorar a acurácia do modelo substituto nas regiões promissoras do espaço de busca. A participação do modelo substituto também aumenta progressivamente e torna-se exclusiva no último ciclo de otimização (Tabela 5.1). No final do último ciclo, as soluções de Pareto obtidas com o auxílio do modelo substituto são finalmente reavaliadas com a função objetivo original, removendo qualquer interferência do modelo substituto nos resultados da otimização, que porventura possa existir devido ao erro de aproximação (ε).

5.2.2 Rede perceptron de múltiplas camadas

A rede perceptron de múltiplas camadas (MLP) é organizada em camadas constituídas por neurônios artificiais. A arquitetura mais simples da MLP consiste de três camadas, a qual é suficiente para realizar a aproximação de funções, de acordo com o teorema da aproximação universal (HAYKIN, 1999). A primeira é a camada de entrada dos dados apresentados à rede, onde o número de neurônios deve coincidir com o número de componentes do vetor de entrada. A segunda é a camada oculta, a qual pode possuir menos ou mais neurônios do que a camada de entrada. O número de neurônios representa a dimensionalidade da camada. O mapeamento do espaço dos dados de entrada para um espaço dimensional superior, definido pela quantidade de neurônios na camada oculta, aumenta a capacidade de interpolação da rede neural. A última camada é a de saída, onde o número de neurônios deve coincidir com o número de componentes do vetor de saída. Matematicamente, os neurônios da camada oculta são descritos da seguinte forma:

$$y_k = \varphi(v_k), \tag{5.4}$$

onde y_k é a saída do k-ésimo neurônio; $\varphi(\cdot)$ é a função de ativação; e v_k denomina-se campo local induzido e é expresso por

$$v_k = b_k + \sum_{j=1}^{p} w_{kj} x_j \,, \tag{5.5}$$

onde x_1, x_2, ..., x_p são os sinais de entrada; w_{k1}, w_{k2}, ..., w_{kp} são os pesos sinápticos do k-ésimo neurônio; e b_k é a polarização (*bias*), a qual é fixada em 1 para todos os neurônios. As saídas da MLP podem ser combinações lineares das saídas dos neurônios da camada oculta, tal como representado pelas equações a seguir.

$$\hat{f}\left(x_1, x_2, \ldots, x_p\right) = \begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \\ \vdots \\ \hat{f}_s \end{bmatrix}, \tag{5.6}$$

onde $\hat{f}_1$, $\hat{f}_2$, ..., $\hat{f}_s$ são as saídas da rede neural, expressas abaixo.

$$\hat{f}_q = \sum_{k=1}^{h} \alpha_{qk} y_k \,, \tag{5.7}$$

onde α_{q1}, α_{q2}, ..., α_{qh} são os pesos sinápticos do q-ésimo neurônio da camada de saída; e y_1, y_2, ..., y_h são as saídas da camada oculta. Quando a MLP é usada para aproximar funções não-lineares, como proposto neste trabalho, suas saídas são geralmente combinações lineares das saídas da camada oculta (HAYKIN, 1999, p. 315). A função de ativação deve ser limitada e monótona crescente (HAYKIN, 1999). Tais condições são satisfeitas por uma função logística, expressa por

$$\varphi(v) = \frac{1}{1+e^{-v}} \,, \tag{5.8}$$

onde v é o campo local induzido (Equação 5.5). Neste trabalho, usou-se o algoritmo *back-propagation*[19] para treinar a MLP. Esse algoritmo minimiza o erro quadrático médio usando o método *steepest descent* (método do gradiente) (HAYKIN, 1999). A taxa de aprendizagem adotada foi de 0,1. Aplicou-se a regra delta generalizada com

[19] O algoritmo *back-propagation* é descrito em Haykin (1999), p. 183.

momento de 0,5. Para determinar o melhor instante de término do treinamento, garantindo erro mínimo de generalização, adotou-se o método *early stopping* baseado em validação cruzada (Figura 5.1).

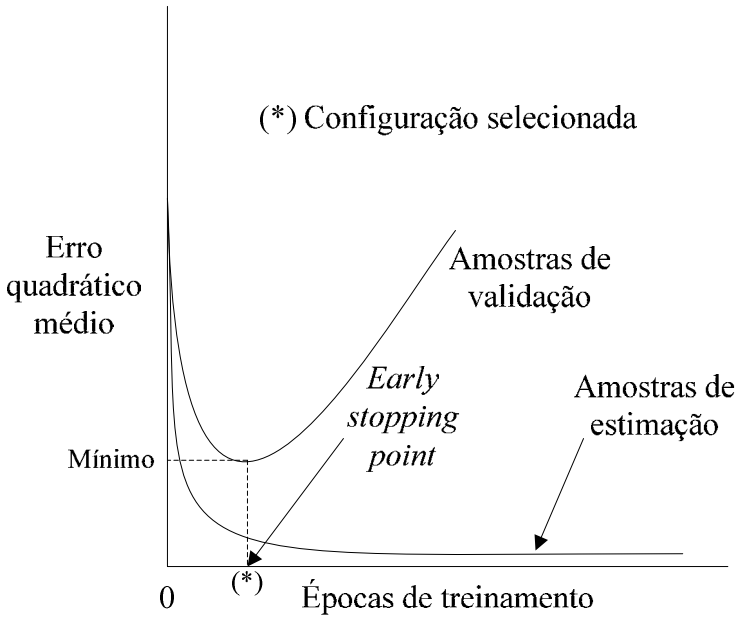


Figura 5.1 - Método *early stopping* baseado em validação cruzada.
Fonte: adaptado de Haykin (1999), p. 238.

Neste estudo, selecionaram-se diferentes configurações da MLP usando a validação cruzada, e avaliaram-se essas configurações com o objetivo de determinar aquela com melhor número de neurônios na camada oculta, que reduzisse o erro quadrático médio sobre o conjunto de amostras.

5.2.3 Rede de função de base radial

A rede de função de base radial (RBFN), tal como a MLP, também é organizada em camadas, cada uma constituída por certo número de neurônios artificiais. No entanto, ao contrário da MLP que pode assumir configurações com mais de uma camada oculta, o número de camadas da RBFN é fixo em três por definição, e os neurônios da camada oculta incorporam funções matemáticas diferentes daquelas incorporadas pelos neurônios da MLP. As funções que as camadas da RBFN executam são similares às da arquitetura da MLP com três camadas. A camada de entrada conecta a rede neural ao

espaço de atributos. A segunda camada, denominada oculta, aplica uma transformação não linear do espaço de atributos para o espaço oculto, cuja dimensionalidade é proporcional ao número de neurônios nesta camada. Os neurônios da camada de saída são combinações lineares das saídas dos neurônios da camada oculta. Diferente da MLP, a polarização não é aplicada aos neurônios da camada escondida, mas aos neurônios da camada de saída. Considerando a mesma notação da Equação 5.6 para representar as saídas da rede neural, as saídas do RBFN têm a seguinte forma:

$$\hat{f}_q = b_q + \sum_{i=1}^{h} w_{qi} \cdot \varphi\left(\left\|\mathbf{x} - \mathbf{t}_i\right\|\right), \tag{5.9}$$

onde $\mathbf{x}$ é o vetor de entrada com os sinais x_1, x_2, ..., x_p; $\mathbf{t}_1$, $\mathbf{t}_2$, ..., $\mathbf{t}_h$ são vetores, denominados centros; $\{\varphi(\|\mathbf{x}\text{-}\mathbf{t}_i\|)|i=1,2,...,h\}$ é um conjunto de h funções arbitrárias, conhecidas como funções de base radial; w_{q1}, w_{q2}, ..., w_{qh} representam os pesos sinápticos do q-ésimo neurônio da camada de saída, e b_q é a polarização do q-ésimo neurônio. Há uma grande variedade de funções de base radial. Neste trabalho, aplicou-se a função gaussiana multivariada, expressa por

$$\varphi\left(\left\|\mathbf{x} - \mathbf{t}_i\right\|_\Sigma\right) = e^{-\left[\frac{1}{2}(\mathbf{x}-\mathbf{t}_i)^T \cdot \Sigma^{-1} \cdot (\mathbf{x}-\mathbf{t}_i)\right]}, \tag{5.10}$$

onde $\|\cdot\|_\Sigma$ denota a distância de Mahalanobis para dada matriz de covariâncias Σ, expressa, neste trabalho, por

$$\Sigma = \sigma^2 \cdot I\ , \tag{5.11}$$

onde I é a matriz identidade; e σ é o desvio padrão da distribuição das amostras estabelecido no espaço de atributos. O vetor de pesos $\mathbf{w}_q$ do q-ésimo neurônio da camada de saída é representado por

$$\mathbf{w}_q = \begin{bmatrix} b_q \\ w_{q1} \\ \vdots \\ w_{qh} \end{bmatrix}, \tag{5.12}$$

e determinado por

$$\mathbf{w}_q = \left(\Phi^T \Phi\right)^{-1} \Phi^T \mathbf{d}_q \,, \tag{5.13}$$

onde

$$\mathbf{d}_q = \begin{bmatrix} f_q\left(x(1)\right) \\ f_q\left(x(2)\right) \\ \vdots \\ f_q\left(x(n)\right) \end{bmatrix}; \tag{5.14}$$

$\{\mathbf{x}(i)|i=1,2,...,n\}$ são os vetores de entrada; $\{f_q(\mathbf{x}(i))|i=1,2,...,n\}$ são os padrões de saída associados aos vetores de entrada; e Φ é denominada matriz de interpolação, expressa por

$$\Phi = \begin{bmatrix} 1 & \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1h} \\ 1 & \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2h} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \varphi_{n1} & \varphi_{n2} & \cdots & \varphi_{nh} \end{bmatrix}, \tag{5.15}$$

onde $\varphi_{ij}=\varphi(\|\mathbf{x}(i)-\mathbf{t}_j\|_\Sigma)$ (Equação 5.10). Da mesma forma como procedeu-se com a MLP, selecionaram-se várias configurações da RBFN, com base no critério do mínimo erro quadrático médio, diferentes entre si em função da posição dos centros, e avaliou-se cada uma das configurações selecionadas a fim de determinar aquela com o melhor número de centros (h) e o melhor valor atribuído ao desvio padrão da distribuição das amostras no espaço de atributos (σ), também com base no erro quadrático médio.

5.2.4 Latin hypercube sampling

O método *latin hypercube sampling* (LHS) é uma generalização do conceito do quadrado latino para um número arbitrário de dimensões. No contexto da amostragem estatística, uma grade quadrada, com as posições das amostras, é um quadrado latino se e somente se há apenas uma amostra em cada linha e coluna da grade. Com base neste conceito, o LHS garante que a distribuição das amostras, em pouca quantidade, é uma

boa representação da variabilidade real ao passo que a amostragem puramente aleatória não oferece nenhuma garantia. O procedimento do LHS para distribuição uniforme é descrito a seguir.

a) Seja $RAN_{m,n}$ uma matriz de valores aleatórios, entre 0 e 1, com distribuição uniforme, onde m é o número de amostras e n é o número de variáveis; $\mathbf{x}_{\min}$, o vetor contendo os limites inferiores das variáveis; e $\mathbf{x}_{\max}$, o vetor contendo os limites superiores das variáveis.

b) Faça $j \leftarrow 0$.

c) Faça $j \leftarrow j+1$.

d) Gere o vetor **idx** cujas componentes são os números inteiros de 1 até o número de amostras permutados aleatoriamente.

e) Faça $i \leftarrow 0$.

f) Faça $i \leftarrow i+1$.

g) Faça $P(i,j) \leftarrow [\mathbf{idx}(i)\text{-}RAN(i,j)]/m$, onde m é o número de amostras.

h) Faça $S(i,j) \leftarrow \mathbf{x}_{\min}(j) + P(i,j)[\mathbf{x}_{\max}(j)\text{-}\mathbf{x}_{\min}(j)]$.

i) Retorne para o passo (f) até que i seja igual ao número de amostras.

j) Retorne para o passo (c) até que j seja igual ao número de variáveis.

Cada linha da matriz S corresponde a um ponto gerado pelo LHS na representação de uma distribuição uniforme no espaço definido pelas variáveis, aplicando o conceito de quadrado latino. A Figura 5.2 exemplifica o espalhamento causado pelo LHS para distribuição uniforme, considerando duas variáveis e diferentes quantidades de amostras.

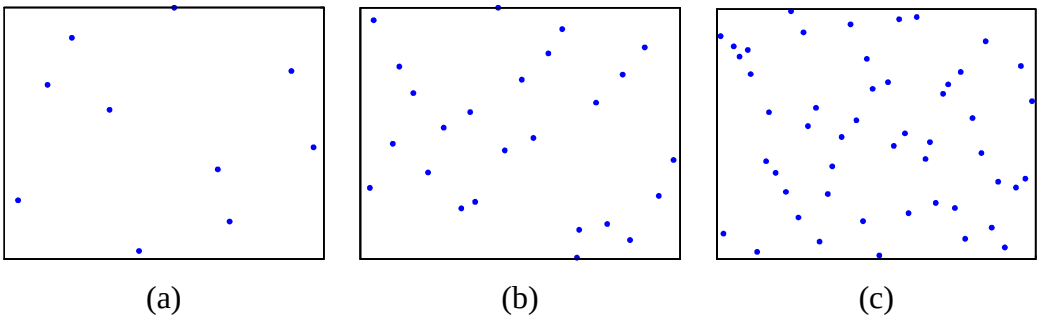


Figura 5.2 - Exemplos de quadrados latinos. (a) 10 amostras; (b) 25 amostras; (c) 50 amostras.

5.3 Procedimento para avaliar a eficiência de modelos substitutos

Nesta seção, apresenta-se a abordagem para estimar a eficiência dos modelos substitutos em termos da quantidade de avaliações da função objetivo requerida para realizar a otimização, considerando as características do NGMOGA. Este procedimento é descrito em duas partes.

Fixando-se a quantidade de avaliações da função objetivo, um modelo substituto é considerado eficiente quando, sem o seu auxílio, o algoritmo de otimização gera uma quantidade de soluções de Pareto menor do que com o seu auxílio. Considerando que o algoritmo de otimização adotado é o NGMOGA, então, para alcançar a mesma quantidade de soluções de Pareto da otimização assistida, a otimização não-assistida deve exceder em quantidade de gerações do NGMOGA com o objetivo de realizar mais avaliações da função objetivo. Quanto mais avaliações são realizadas, mais soluções são encontradas.

A primeira parte deste procedimento consiste em considerar, na estimativa da eficiência do modelo substituto, a quantidade de gerações excedida na otimização não-assistida. Seja Q_1 a quantidade de gerações do NGMOGA realizadas na optimização não-assistida tal que a quantidade de avaliações da função objetivo seja menor do que ou igual à quantidade de avaliações realizadas na optimização assistida. Seja Q_2 a quantidade de gerações realizadas na optimização não-assistida tal que o percentual de soluções de Pareto seja maior do que ou igual à quantidade obtida com a optimização assistida, fixando-se o tamanho da população do NGMOGA. A eficiência do modelo substituto

em termos da quantidade de gerações excedida pela optimização não-assistida, denotada por η_1, é definida por

$$\eta_1 = \frac{Q_2 - Q_1}{Q_2} . \tag{5.16}$$

Considerando que a cada geração do NGMOGA dois novos indivíduos são avaliados[20], a quantidade Q_1 é estimada por

$$Q_1 = \frac{(1-P)N}{2} p + (1-P) g + \frac{1}{2} e , \tag{5.17}$$

onde P é a probabilidade média de substituição, que é 0,9667, com base na Tabela 5.1; N é a quantidade de ciclos de optimização, fixada em 3 (Tabela 5.1); p é o tamanho da população; g é a quantidade de gerações realizadas na optimização assistida; e e é a quantidade de amostras da função objetivo.

Para determinar a quantidade Q_2, a optimização não-assistida é executada de forma iterativa, incrementando-se a quantidade de gerações, até que o mesmo percentual de soluções de Pareto, produzida pela optimização assistida, seja alcançado. Fixou-se o incremento em 5 gerações.

No entanto, avaliar a eficiência somente em termos da quantidade de gerações excedidas não é suficiente. É necessário, também, considerar a distribuição das soluções. Segue, portanto, a segunda parte deste procedimento. Para que se possa comparar modelos substitutos com base em η_1, as soluções de Pareto geradas na optimização assistida devem estar bem distribuídas ao longo da fronteira de Pareto. A fim de avaliar a distribuição das soluções de Pareto, elaborou-se o conceito de célula.

As células são representadas por pontos ao longo da fronteira de Pareto, como esboçado na Figura 5.3, onde assume-se que o problema seja contínuo e de minimização e que o contradomínio da função multiobjetivo seja definido no espaço $\mathbb{R}^2$. As células

[20] O procedimento do NGMOGA é descrito na página 38.

identificam porções do espaço de soluções viáveis distribuídas ao longo da fronteira de Pareto (Figura 5.3).

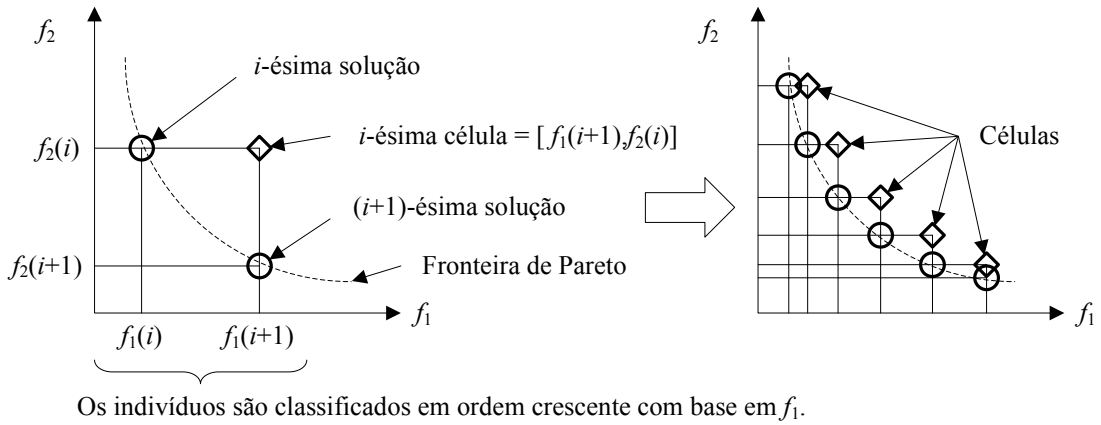


Figura 5.3 - Esquema das células.

Considerando N pontos bem distribuídos da fronteira de Pareto ideal, a eficiência do modelo substituto, denotada por η_2, definida em termos da distribuição das soluções obtidas com o auxílio da rede neural, é expressa por

$$\eta_2 = \frac{1}{N-1}\sum_{i=1}^{N-1} D_{\text{célula}}(i) \tag{5.18}$$

e

$$D_{\text{célula}}(i) = \begin{cases} 1, \text{ se } D(i) > 0; \\ 0, \text{ caso contrário,} \end{cases} \tag{5.19}$$

onde D(i) denota a dominância da i-ésima célula aplicando-se a Equação 4.9. Quanto melhor for a distribuição das soluções de Pareto ao longo da fronteira de Pareto, maior será o resultado de η_2, podendo alcançar seu valor máximo igual a 1 se a fronteira for conexa. Se a fronteira de Pareto for desconexa, algumas células representarão o espaço entre suas partições. Assim, dadas soluções preenchendo todas as células das partições de uma fronteira de Pareto desconexa, o resultado de η_2 será inferior a 1, pois as células dos espaços entre as partições dessa fronteira também estarão computadas no

denominador da Equação 5.18. No entanto, apesar dessa observação, se o objetivo é comparar a qualidade da distribuição causada por dois ou mais métodos ao longo da mesma fronteira de Pareto, a comparação em termos de η_2 é adequada, pois o maior resultado indicará o melhor método. Combinando-se a primeira e segunda parte deste procedimento, a eficiência do modelo substituto, denotada por η, foi então definida da seguinte forma:

$$\eta = \begin{cases} \eta_1 \eta_2, & \text{se } \eta_1 \geq 0; \\ 0, & \text{caso contrário.} \end{cases} \tag{5.20}$$

O termo η_2 ocorre no intervalo [0, 1]. O termo η_1, no entanto, pode ser negativo, o que significa que, neste caso, o modelo substituto é ineficiente. Uma vez que não há interesse em comparar a magnitude da ineficiência de um modelo substituto, faz-se η igual a zero se η_1 é negativo (Equação 5.20).

5.4 Estudo de caso

Selecionaram-se três funções de teste, disponíveis na literatura (Tabela 5.2), a fim de avaliar a eficiência da MLP e da RBFN como modelos substitutos. Essas funções foram escolhidas de acordo com os seguintes critérios:

a) dimensionalidade do espaço do domínio e do contradomínio; e

b) complexidade do problema.

A fim de facilitar a análise dos resultados, escolheram-se funções de $\mathbb{R}^2$ em $\mathbb{R}^2$, como sugerido em Sóbester e Forrester (2007). Evitaram-se funções deceptivas[21] e definidas com restrições a fim de não exacerbar a complexidade do problema. Escolheram-se funções de modo que uma fosse mais complexa do que a anterior, a fim de analisar o efeito da complexidade sobre o desempenho das redes neurais (Tabela 5.2).

[21] As funções deceptivas possuem ótimo global em lugar improvável do espaço de busca, incapacitando qualquer tipo de método em realizar a otimização de forma eficiente (HUBAND et al., 2006).

Tabela 5.2 - Funções de teste.

Identificação	Definição	Domínio
1ª(1)	$f(x_1,x_2)=\begin{bmatrix} f_1 & = & (x_1-2)^2+(x_2-1)^2+2 \\ f_2 & = & 9x_1-(x_2-1)^2 \end{bmatrix}$	[-20, 20]
2ª(2)	$f(x_1,x_2)=\begin{bmatrix} f_1 & = & 1+(\mathrm{A}_1-\mathrm{B}_1)^2+(\mathrm{A}_2-\mathrm{B}_2)^2 \\ f_2 & = & (x_1+3)^2+(x_2+1)^2 \end{bmatrix}$ onde $\mathrm{A}_1=0,5\sin 1-2\cos 1+\sin 2-1,5\cos 2$ $\mathrm{A}_2=1,5\sin 1-\cos 1+2\sin 2-0,5\cos 2$ $\mathrm{B}_1=0,5\sin x_1-2\cos x_1+\sin x_2-1,5\cos x_2$ $\mathrm{B}_2=1,5\sin x_1-\cos x_1+2\sin x_2-0,5\cos x_2$	[-π, π]
3ª(3)	$f(x_1,x_2)=\begin{bmatrix} f_1 & = & -2\exp\left(15\left(-(x_1-0,1)^2-x_2^2\right)\right) \\ & & -\exp\left(20\left(-(x_1-0,6)^2-(x_2-0,6)^2\right)\right) \\ & & +\exp\left(20\left(-(x_1+0,6)^2-(x_2-0,6)^2\right)\right) \\ & & +\exp\left(20\left(-(x_1-0,6)^2-(x_2+0,6)^2\right)\right) \\ & & +\exp\left(20\left(-(x_1+0,6)^2-(x_2+0,6)^2\right)\right) \\ f_2 & = & 2\exp\left(20\left(-x_1^2-x_2^2\right)\right) \\ & & +\exp\left(20\left(-(x_1-0,4)^2-(x_2-0,6)^2\right)\right) \\ & & -\exp\left(20\left(-(x_1+0,5)^2-(x_2-0,7)^2\right)\right) \\ & & -\exp\left(20\left(-(x_1-0,5)^2-(x_2+0,7)^2\right)\right) \\ & & +\exp\left(20\left(-(x_1+0,4)^2-(x_2+0,8)^2\right)\right) \end{bmatrix}$	[-1, 1]

Fontes: (1) Srinivas e Deb (1994);
(2) Deb et al. (2000); e
(3) Huband et al. (2006).

A complexidade das funções de teste (Tabela 5.2) varia em termos da modalidade[22], da separabilidade e da geometria da fronteira de Pareto, conforme descrito na Tabela 5.3.

Tabela 5.3 - Descritores da complexidade das funções de teste.

Funções		Descritores		
		Modalidade	Separabilidade	Geometria
1ª	f_1	Unimodal	Separável	Convexa
	f_2	Unimodal	Separável	
2ª	f_1	Multimodal	Não separável	Desconexa
	f_2	Unimodal	Separável	
3ª	f_1	Multimodal	Não separável	Mista
	f_2	Multimodal	Não separável	

Os descritores apresentados na Tabela 5.3 foram abordados por Huband et al. (2006). As funções podem ser unimodais ou multimodais, separáveis ou não separáveis. A função é multimodal quando possui máximos e mínimos locais, e não separável quando não pode ser decomposta em funções dependentes de uma única variável. A geometria da fronteira de Pareto pode ser convexa, linear, côncava, mista, degenerada, desconexa, ou qualquer combinação mais complexa dessas formas. A geometria é desconexa quando a fronteira apresenta descontinuidades. A forma mista é uma combinação das formas convexa, linear e côncava. Para todos os casos de otimização, executados com as funções de teste, as definições das diferentes configurações dos modelos substitutos e as do procedimento para estimar sua eficiência foram iguais. As definições da MLP são apresentadas na Tabela 5.4, enquanto as da RBFN, na Tabela 5.5.

O procedimento para estimar a eficiência dos modelos substitutos consistiu dos passos de (a) a (i).

a) O conjunto das amostras para gerar o modelo substituto foi produzido através do método LHS.

[22] A modalidade de uma função refere-se à complexidade em termos do número de máximos e mínimos locais.

Tabela 5.4 - Definições da MLP.

Parâmetros	Valores
Número mínimo de neurônios na camada oculta	25
Número máximo de neurônios na camada oculta	75
Passo de variação da quantidade de neurônios na camada oculta	5
Taxa de aprendizagem	0,1
Momento	0,5
Número máximo de épocas de treinamento	6000
Tolerância de erro quadrático médio	0,0001

Tabela 5.5 - Definições da RBFN.

Parâmetros	Valores
Desvio padrão mínimo da função de base radial	200
Desvio padrão máximo da função de base radial	7000
Passo de variação do desvio padrão	200
Porcentagem mínima de centros em relação ao número de amostras	20
Porcentagem máxima de centros em relação ao número de amostras	80
Passo de variação da porcentagem de centros em relação às amostras	20
Número de tentativas para encontrar a melhor distribuição dos centros	10

b) Esse conjunto foi dividido aleatoriamente em três partes, como proposto em Friedman et al. (2009): um subconjunto de estimação, outro de validação, e um terceiro de teste. O subconjunto de estimação foi usado para obter diferentes estados do modelo substituto para cada configuração, fixando-se a quantidade de neurônios, ou de centros, na camada oculta e, no caso da RBFN, o desvio padrão da função de base radial. As configurações da MLP e da RBFN foram geradas considerando-se os passos de variação de suas definições apresentados nas Tabelas 5.4 e 5.5, respectivamente. O subconjunto de validação foi usado para selecionar o melhor estado de cada configuração. Estes dois subconjuntos constituem a parte das amostras de treinamento. O subconjunto de teste foi usado para

determinar a configuração de cada modelo substituto com a melhor capacidade de generalização. Uma divisão típica consiste em atribuir 25% do conjunto de amostras ao subconjunto de teste e 25% dessas amostras ao subconjunto de validação, mantendo-se 50% do conjunto de amostras para o subconjunto de estimação (FRIEDMAN et al., 2009). No entanto, na prática da otimização multidisciplinar, onde, na maioria dos casos, a função objetivo tem custo computacional elevado, não se dispõe de muitas amostras. Também é usual atribuir 80% das amostras para compor o subconjunto de treinamento (estimação e validação) e 20% para compor o de teste (HAYKIN, 1999). Adotou-se então essa divisão, prevendo as condições de uma optimização com poucas amostras disponíveis. Portanto, atribuiu-se ao subconjunto de teste 20% do conjunto de amostras e ao de validação 16% dessas amostras, mantendo-se 64% do conjunto de amostras para o subconjunto de estimação (Tabela 5.6).

c) Estimaram-se e validaram-se até 11 configurações da MLP, variando o número de neurônios na camada oculta (Tabela 5.4). Realizou-se o mesmo procedimento com as 140 configurações da RBFN, variando o desvio padrão da função de base radial e a porcentagem de centros em relação ao número total de amostras (Tabela 5.5).

d) Selecionou-se a melhor configuração de cada modelo substituto.

Tabela 5.6 - Divisão do número de amostras com e sem o subconjunto de teste.

Subconjunto	Funções de teste					
	1ª		2ª		3ª	
	Com teste	Sem teste	Com teste	Sem teste	Com teste	Sem teste
Estimação	52	64	64	80	128	160
Validação	12	16	16	20	32	40
Teste	16	-	20	-	40	-
Total	80		100		200	

Tabela 5.7 - Configurações dos modelos substitutos com a melhor capacidade generalização.

Parâmetros	Funções de teste					
	1ª		2ª		3ª	
	MLP	RBFN	MLP	RBFN	MLP	RBFN
Número de nós[(1)] na camada oculta	75	20	60	51	65	102
Desvio padrão da função de base radial	-	5000	-	600	-	200

Nota: [(1)] o termo nó denota um neurônio ou um centro, dependendo da rede neural, MLP ou RBFN, respectivamente.

e) Dividiu-se, novamente, o conjunto de amostras, mas em duas partes, um subconjunto de estimação e outro de validação, uma vez que as amostras de teste não eram mais necessárias e podiam então ser aproveitadas no treinamento dos modelos substitutos, considerando a melhor configuração determinada nos passos de (b) a (d).

f) Então, a melhor configuração de cada modelo substituto foi, novamente, estimada e validada, desta vez com mais amostras de treinamento.

g) Geraram-se 30 pontos bem distribuídos ao longo da fronteira de Pareto, realizando inúmeras avaliações da função multiobjetivo, a fim de ter uma representação da fronteira ideal. Esses pontos foram usados para determinar o percentual de soluções de Pareto, obtidas com a otimização assistida e com a não-assistida, fixando-se o número de avaliações da função multiobjetivo. Além disso, foram também usados para definir as células (Figura 5.3), usadas na avaliação da distribuição das soluções ao longo da fronteira de Pareto.

h) A optimização assistida e a não-assistida foram executadas 10 vezes, a fim de proporcionar uma boa estimativa, com base estatística, dos seguintes dados para avaliar a eficiência dos modelos substitutos: o percentual

médio de soluções de Pareto obtidas com a otimização assistida (Tabela 5.8); e o valor de η_2 (Equação 5.18).

Tabela 5.8 - Porcentagem média de soluções de Pareto com base em 10 execuções da otimização.

Otimização	Funções de teste					
	1ª		2ª		3ª	
	MLP	RBFN	MLP	RBFN	MLP	RBFN
Assistida	80	82	46	58	91	87
Não-assistida	54	52	4	5	79	80

Tabela 5.9 - Configurações do algoritmo genético multiobjetivo não-geracional adotadas no passo (h).

Parâmetros	Funções de teste		
	1ª	2ª	3ª
Tamanho da população(1)	70	80	100
Número de gerações(2)	150	270	360

Notas: (1) o tamanho da população é denotado por p e o mesmo valor é aplicado à otimização assistida e à não-assistida; e

(2) o número de gerações é denotado por g e os valores são relativos à otimização assistida.

O número de gerações relativo à optimização não-assistida, denotado por g', é dado por

$$g' = \left\lceil \frac{N_{eval} - p}{2} \right\rceil, \tag{5.21}$$

onde N_{eval} é a quantidade de avaliações da função multiobjetivo, contada durante a execução da optimização assistida; e p é o tamanho da população (Tabela 5.9). A Equação 5.21 fornece a quantidade necessária de gerações para que a optimização não-assistida realize uma quantidade

de avaliações da função multiobjetivo igual ou maior que a quantidade realizada pela optimização assistida. As Figuras 5.4, 5.6 e 5.8 apresentam os resultados do passo (h) relativos ao uso da MLP como modelo substituto, enquanto as Figuras 5.5, 5.7 e 5.9 apresentam os resultados relativos à RBFN.

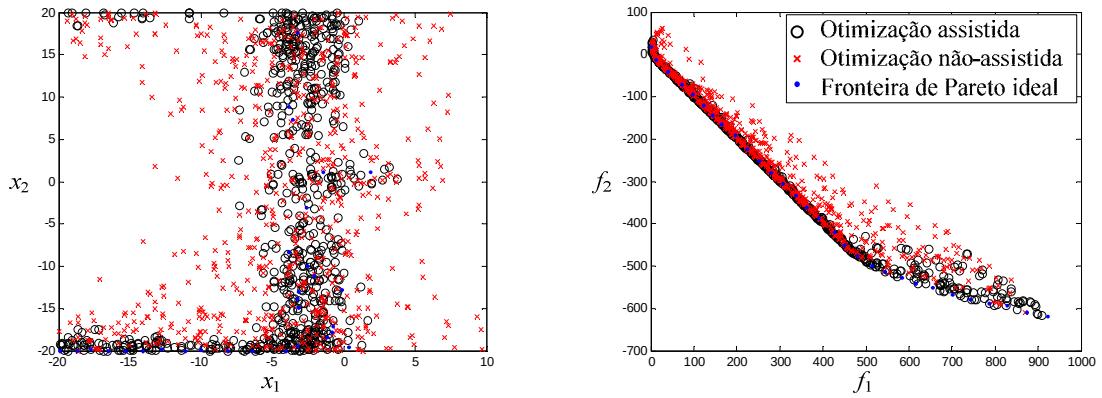
Tabela 5.10 - Valor médio do número de avaliações da função multiobjetivo com base em 10 execuções da otimização.

Otimização	Função de teste					
	1ª		2ª		3ª	
	MLP	RBFN	MLP	RBFN	MLP	RBFN
Assistida	170	170	210	209	336	349
Não-assistida	175	176	223	223	380	395

Tabela 5.11 - Tempo médio de processamento decorrido no passo (h) com base em 10 execuções da otimização em uma máquina 2.0 GHz Quad Core.

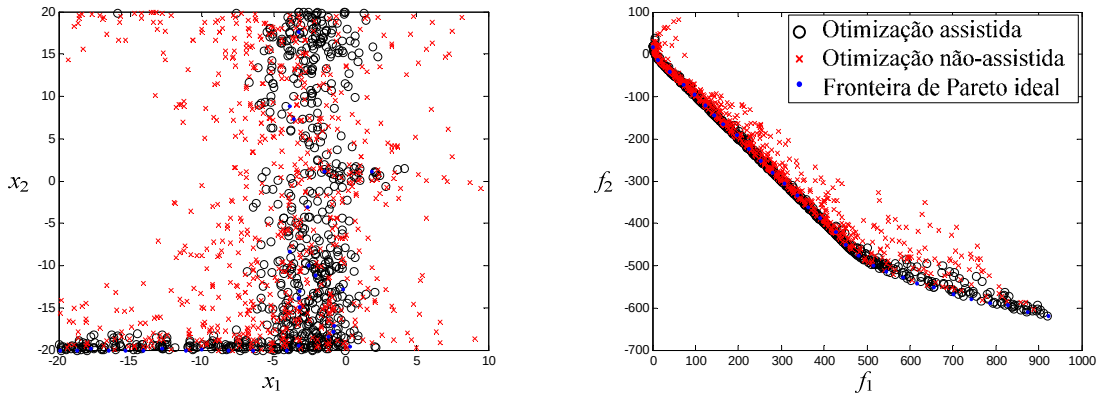
Modelo substituto	Tempo de processamento da otimização (s)		
	1ª função de teste	2ª função de teste	3ª função de teste
MLP	684	518	40
RBFN	6	21	78

i) Por fim, executou-se 100 vezes o procedimento da avaliação da eficiência dos modelos substitutos, para cada modelo, com o objetivo de compará-los com base no valor médio da eficiência. Determinaram-se as quantidades Q_1 e Q_2 para então inseri-las na Equação 5.16 a fim de obter η_1. A quantidade Q_1 foi determinada através da Equação 5.17 com base na média das probabilidades de substituição (Tabela 5.1), no número total de amostras (Tabela 5.6) e nas configurações do NGMOGA (Tabela 5.9). A quantidade Q_2 foi determinada executando-se a otimização não-assistida até que o percentual de soluções de Pareto fosse igual a ou maior do que os valores médios da otimização assistida apresentados na Tabela 5.8.



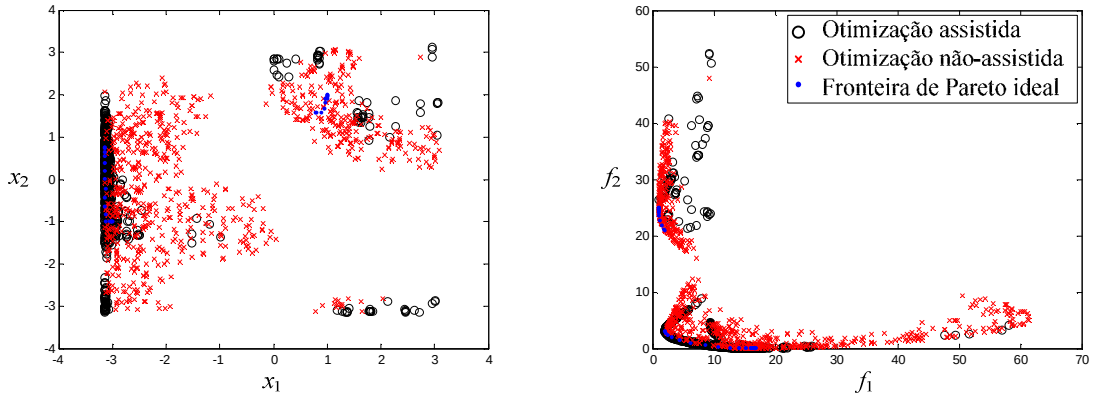
(a) (b)

Figura 5.4 - Resultados sobrepostos de 10 execuções da otimização da 1ª função de teste assistida pela MLP e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio.



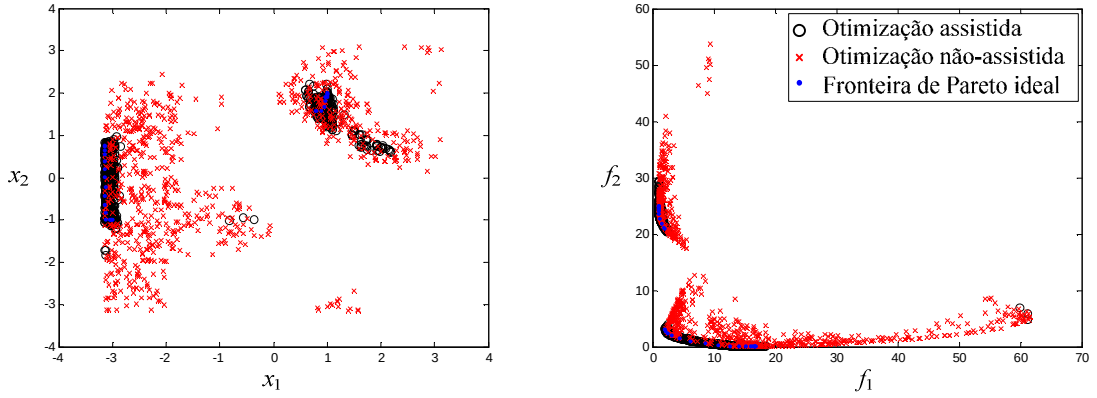
(a) (b)

Figura 5.5 - Resultados sobrepostos de 10 execuções da otimização da 1ª função de teste assistida pela RBFN e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio.



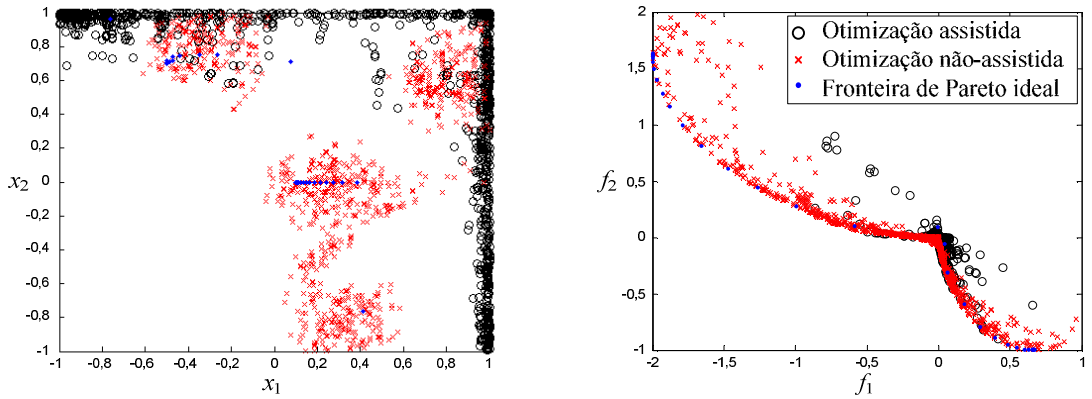
(a) (b)

Figura 5.6 - Resultados sobrepostos de 10 execuções da otimização da 2ª função de teste assistida pela MLP e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio.



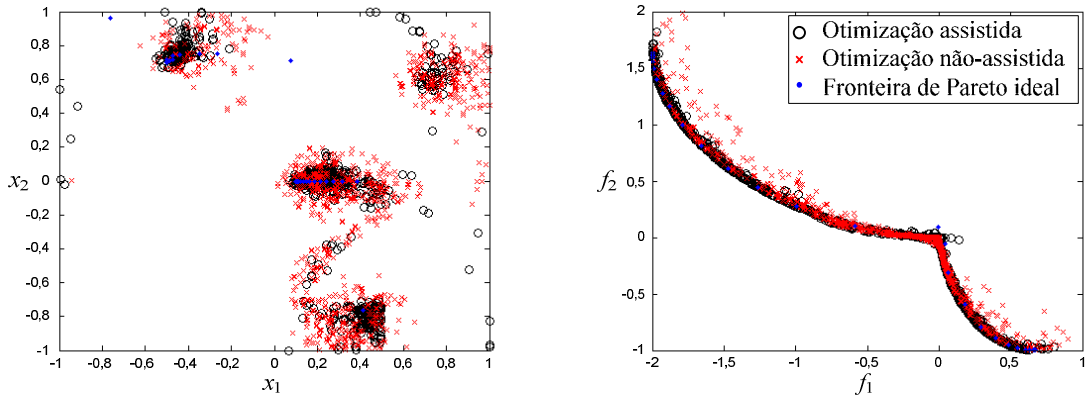
(a) (b)

Figura 5.7 - Resultados sobrepostos de 10 execuções da otimização da 2ª função de teste assistida pela RBFN e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio.



(a) (b)

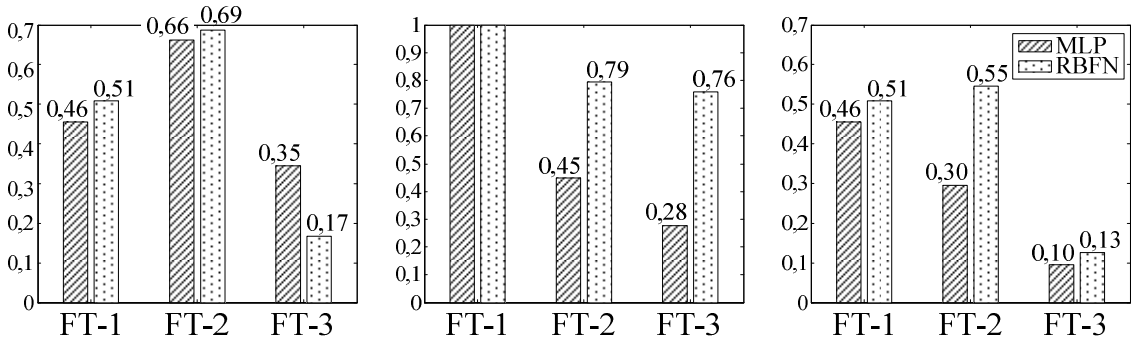
Figura 5.8 - Resultados sobrepostos de 10 execuções da otimização da 3ª função de teste assistida pela MLP e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio.



(a) (b)

Figura 5.9 - Resultados sobrepostos de 10 execuções da otimização da 3ª função de teste assistida pela RBFN e 10 da otimização não-assistida, ambos os casos considerando as configurações do algoritmo genético multiobjetivo não-geracional apresentadas na Tabela 5.9. (a) Domínio da função multiobjetivo; (b) contradomínio.

A Figura 5.10 apresenta o resultado deste estudo.



(a) (b) (c)

FT: Função de teste

Figura 5.10 - Comparação da eficiência da MLP e da RBFN como modelos substitutos na otimização assistida das funções de teste. (a) Resultado de η_1 de acordo com a Equação 5.16; (b) resultado médio de η_2, aplicando-se a Equação 5.18, com base em 10 execuções da otimização conforme o passo (h) e considerando 29 células para avaliar a distribuição das soluções ao longo da fronteira de Pareto com base nos pontos gerados no passo (g); (c) resultado médio da eficiência η de cada modelo substituto, dada pela Equação 5.20, com base em 100 execuções do procedimento para determinar a eficiência conforme mencionado no passo (i).

Não é possível chegar a conclusões gerais sobre a eficiência dos modelos substitutos, pois esta análise ainda carece de muitas avaliações com funções de teste. Além disso, certos parâmetros ainda são arbitrados sem um perfeito entendimento sobre sua influência na eficiência dos modelos substitutos. Por exemplo, não se investigou se os valores atribuídos à taxa de aprendizagem e ao momento da MLP eram realmente adequados para a aplicação dessa rede na otimização assistida de cada função de teste. Contudo, apesar das limitações desta análise, os resultados sugerem que RBFN é mais simples, rápida e robusta do que a MLP. A implementação da RBFN é significativamente mais simples do que a da MLP. Os pesos sinápticos da RBFN são obtidos através da pseudo-inversa da matriz de interpolação (Equação 5.13), cujo procedimento é mais simples de implementar do que o algoritmo *back-propagation*, utilizado para obter os pesos sinápticos da MLP.

De acordo com a Tabela 5.11, o tempo de processamento decorrido no passo (h) para a otimização assistida das funções de teste 1 e 2 são significativamente diferentes entre a RBFN e a MLP, indicando que a RBFN é mais rápida do que a MLP, ao menos, 25 vezes mais rápida. Examinou-se por que a RBFN não foi mais rápida do que a MLP no caso da 3ª função de teste. Neste caso, em particular, constatou-se que o treinamento da MLP foi interrompido precocemente por evoluir abaixo da tolerância do erro quadrático médio (Tabela 5.4). O desempenho da MLP, aparentemente melhor do que o da RBFN, na otimização assistida da 3ª função de teste (Figura 5.10a) não tem significado, tendo em vista o fraco desempenho dessa rede na aproximação dessa função de teste. O mau desempenho da MLP na otimização da 3ª função de teste é evidenciado na Figura 5.10b, onde constata-se que as soluções obtidas com a MLP dominaram apenas 28% das células distribuídas ao longo da fronteira de Pareto, indicando que essas soluções estão más distribuídas em comparação com as soluções obtidas com a RBFN, as quais dominaram 76% das células. Essa diferença de resultados entre a MLP e a RBFN também pode ser percebida observando-se as Figuras 5.8b e 5.9b, respectivamente.

A RBFN foi eficiente na otimização da 1ª e 2ª função de teste, enquanto a eficiência da MLP caiu com a complexidade do problema de otimização, a qual aumentou progressivamente da 1ª para a 3ª função de teste (Figura 5.10c). A eficiência da RBFN caiu na otimização assistida da 3ª função de teste. Apesar de as soluções obtidas com a RBFN terem dominado 76% das células na optimização assistida da 3ª função de teste (Figura 5.10b), indicando uma boa distribuição das soluções ao longo da fronteira de Pareto, na otimização dessa função, a diferença entre a optimização assistida e não-assistida, em termos de quantidade excedida de avaliações da função multiobjetivo pela otimização não-assistida para alcançar o mesmo percentual de soluções de Pareto obtido com a otimização assistida, foi de apenas 17%, a menor eficiência alcançada pela RBFN (figura 5.10a). Provavelmente, se a influência das definições do NGMOGA sobre o desempenho da RBFN tivesse sido investigada mais a fundo, teria sido possível obter uma eficiência melhor dessa rede na otimização assistida da 3ª função de teste.

Em busca de trabalhos semelhantes a este, encontrou-se o de Peter e Marcelet (2008). Seus resultados também indicam que o desempenho da RBFN é melhor em comparação

com a MLP em problemas de aproximação de funções, corroborando com este trabalho. Além da MLP e da RBFN, estes autores analisaram o uso de polinômios e do método *simple kriging* como modelos substitutos, e chegaram à conclusão que o *kriging*, método oriundo da geoestatística, é tão bom quanto a RBFN.

Esta análise foi motivada pelo propósito de aplicar esta técnica na otimização multidisciplinar de foguetes de sondagem. Para esta aplicação, recomenda-se o uso da RBFN como modelo substituto. Portanto, em trabalhos futuros, a técnica de otimização assistida por modelo substituto poderá ser incorporada no MDO-SONDA, juntamente com a abordagem memética, apresentada no Capítulo 4, como mostra a Figura 5.11.

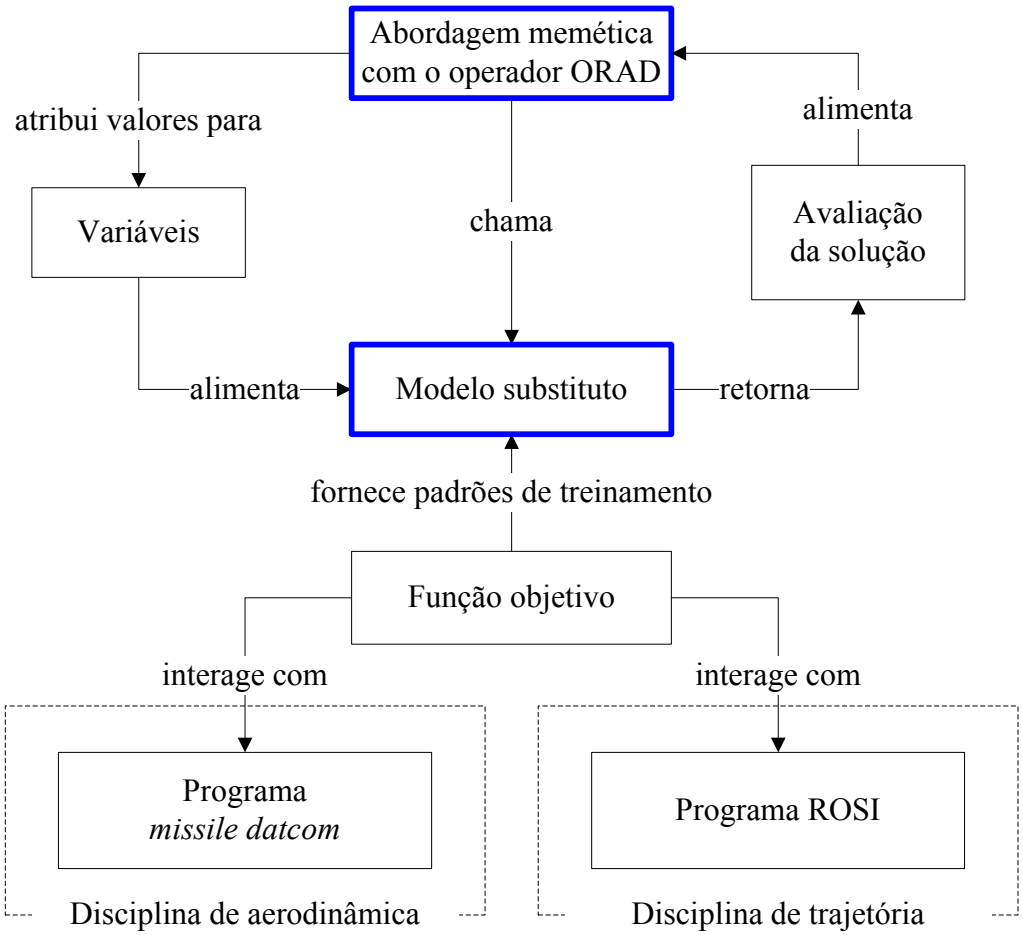


Figura 5.11 - Arquitetura do MDO-SONDA com a abordagem memética e o modelo substituto, no lugar da função objetivo, como apresentado nas Figuras 3.8 e 4.30, respectivamente.

6 ANÁLISE EXPLORATÓRIA DE SOLUÇÕES DE PARETO

6.1 Introdução

Nos dois capítulos anteriores, exploraram-se abordagens para reduzir a quantidade de avaliações da função objetivo na otimização, com o propósito de tornar a otimização mais eficiente. Primeiro, propôs-se uma abordagem memética baseada na cooperação entre o algoritmo genético multiobjetivo não-geracional (NGMOGA) e o algoritmo multiobjetivo da otimização extrema generalizada (M-GEO), tal que essa cooperação requeresse uma quantidade de avaliações menor do que o exigido por esses algoritmos, sem o auxílio do outro, alcançando-se a mesma qualidade nos resultados. Segundo, comparou-se a eficiência de redes neurais como modelos substitutos. O propósito, agora, é explorar soluções de Pareto definidas em espaços multidimensionais. Neste capítulo, portanto, apresenta-se uma abordagem para sumarizar dados de otimização, quando a quantidade de variáveis e objetivos de projeto supera a capacidade de visualização de dados por meio de gráficos convencionais, dificultando a identificação de suas características (Figura 6.1).

A Figura 6.1 ilustra o aumento progressivo da complexidade de visualização de dados à medida que o número de componentes da função multiobjetivo aumenta, indicando que, para uma quantidade de componentes superior a três, a visualização requer o uso de técnicas especiais como, por exemplo, as coordenadas paralelas (Figura 6.1c).

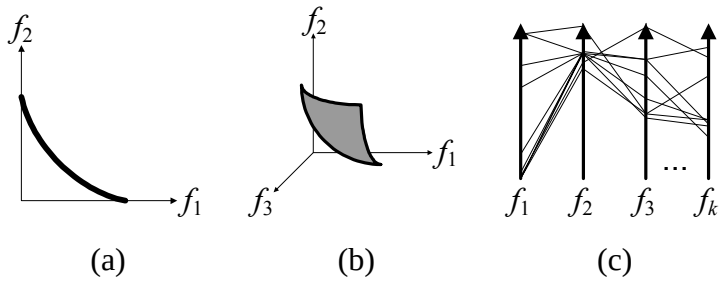


Figura 6.1 - Complexidade da visualização da fronteira de Pareto. (a) Complexidade baixa; (b) moderada; (c) alta.

A sumarização consiste em identificar os compromissos entre as variáveis e os objetivos de projeto. Primeiro, aplicam-se os mapas auto-organizáveis com o k-médias, a fim de

identificar os agrupamentos de soluções no domínio da função multiobjetivo, chamados de perfis de projeto, e no contradomínio, chamados de perfis de desempenho. Em seguida, estima-se a probabilidade dos perfis de desempenho observando-se os de projeto, e a probabilidade a priori dos perfis de projeto. Por último, com base nas estimativas anteriores, estima-se a probabilidade condicional inversa, ou seja, a dos perfis de projeto observando-se os de desempenho. Do ponto de vista de engenharia de sistemas, a probabilidade condicional inversa é mais interessante, pois possibilita que decisões sejam mais facilmente tomadas a respeito de que conceito de produto adequa-se mais aos requisitos de projeto, representando-se esses requisitos através de perfis de projeto.

Este capítulo está organizado como segue. Primeiro, apresenta-se a técnica de visualização baseada em coordenadas paralelas e aborda-se a fundamentação teórica deste capítulo: o método k-médias, os mapas auto-organizáveis e o teorema de Bayes. Em seguida, apresenta-se a proposta deste autor para sumarizar dados de otimização. Por último, discute-se um estudo de caso, cujo objetivo é demonstrar essa abordagem.

6.2 Fundamentação teórica

6.2.1 Coordenadas paralelas

A técnica das coordenadas paralelas é comumente empregada na visualização de espaços contínuos multidimensionais. A visualização consiste em dispor os eixos das variáveis em linhas paralelas, verticais, igualmente espaçadas e em mesmo sentido, onde os valores máximos e mínimos da variável correspondem aos limites superiores e inferiores da linha (GRINSTEIN et al., 2001).

6.2.2 K-Médias

O k-médias é um método de agrupamento em espaço contínuo bastante popular por sua simplicidade computacional. Baseia-se no uso de centroides para dividir o conjunto de pontos em partições, onde o quadrado da distância euclidiana é adotado para mensurar a

dissimilaridade entre os pontos e os centroides (THEODORIDIS; KOUTROUMBAS, 2003, p. 531-532). Apresenta-se, abaixo, o algoritmo básico do k-médias.

a) Selecione arbitrariamente K centroides iniciais ($\mathbf{c}_1$, $\mathbf{c}_2$, ..., $\mathbf{c}_K$).

b) Associe cada ponto **p** ao centroide mais próximo **c**, onde cada conjunto de pontos associados a um centroide constitui uma partição.

$$\mathbf{c} = \min_i \left\| \mathbf{p} - \mathbf{c}_i \right\|^2 , \; i = 1, \ldots, K , \tag{6.1}$$

onde o símbolo ||·|| denota a distância euclidiana.

c) Atualize os centroides atribuindo a média dos pontos associados a eles, de acordo com a equação

$$\mathbf{c} = \frac{1}{m} \sum_{p \in c} \mathbf{p} , \tag{6.2}$$

onde m é o número de pontos **p** associados ao centroide **c**.

d) Retorne para o passo (b) até que os centroides permaneçam os mesmos.

6.2.3 Mapas auto-organizáveis

Os mapas auto-organizáveis baseiam-se na forma como o córtex cerebral se adapta representando um mapa topológico dos sinais percebidos pelo sistema do córtex visual. Os estímulos nervosos sensibilizam determinadas áreas do cérebro, preservando a topologia do sinal que o originou. Por exemplo, quando o cérebro recebe estímulos sonoros, os sons altos se distinguem topologicamente dos baixos no córtex cerebral. O mesmo ocorre quando o cérebro recebe estímulos visuais. Assim, o cérebro se adapta de modo a preservar a topologia dos sinais. Os mapas auto-organizáveis são inspirados nesse mecanismo.

O modelo dos mapas auto-organizáveis foi introduzido em 1981. Todavia, foi em 1989 que um artigo completo sobre os mapas, apresentado por Teuvo Kohonen, fez com que

este modelo ganhasse notoriedade e ficasse conhecido como as redes neurais de Kohonen. Em seu artigo, Kohonen apresenta um modelo de rede neural, com apenas uma camada, dotada de certa quantidade de neurônios, dispostos de forma a constituir um reticulado uniforme, onde os resultados dos pesos sinápticos de cada neurônio se aproximam da distribuição dos dados na medida em que são apresentados à rede (MEHRA, 1992).

A representação dos dados nos mapas auto-organizáveis é obtida através de um mecanismo de aprendizagem onde os neurônios competem para se aproximar de um dado quando este é apresentado à rede. Há critérios de comparação para determinar o neurônio vencedor. Um dos critérios apresentados por Kohonen foi o da distância euclidiana entre os pesos sinápticos e as entradas. O neurônio vencedor adapta-se com maior magnitude do que os seus vizinhos no reticulado a fim de aproximar-se mais do dado apresentado à rede, ou estímulo. Para a rede se auto-organizar, preservando a topologia dos estímulos, é preciso que os vizinhos do neurônio vencedor se adaptem, mas com intensidade menor e estritamente decrescente em função de sua distância em relação ao neurônio vencedor.

Os mapas auto-organizáveis possuem uma representação algébrica simples comparados a outros modelos de redes neurais. Os estímulos e os neurônios são representados na forma de vetores. Denota-se por V o conjunto de todas as entradas.

$$V = \{\mathbf{v}_i\}, \quad i = 1, ..., N, \tag{6.3}$$

onde

$$\mathbf{v}_i = \begin{bmatrix} v_{i1} \\ v_{i2} \\ \vdots \\ v_{iD} \end{bmatrix} \tag{6.4}$$

representa o i-ésimo dado de entrada. Denota-se por M o conjunto de todos os neurônios.

$$M = \{\mathbf{m}_j\}, \quad j = 1, ..., Q, \tag{6.5}$$

onde

$$\mathbf{m}_j = \begin{bmatrix} m_{j1} \\ m_{j2} \\ \vdots \\ m_{jD} \end{bmatrix} \tag{6.6}$$

representa os pesos sinápticos do j-ésimo neurônio. Os vetores $\mathbf{v}_i$ e $\mathbf{m}_j$ possuem a mesma dimensão, ($\mathbf{v}_i$, $\mathbf{m}_j \in \mathbb{R}^D$), uma vez que são definidos no mesmo espaço, denominado espaço de atributos. A adaptação dos pesos é descrita por

$$\mathbf{m}_j(i+1) = \mathbf{m}_j(i) + h(c, j, i)\left[\mathbf{v}_i - \mathbf{m}_j(i)\right], \tag{6.7}$$

onde $\mathbf{m}_j(i+1)$ é o vetor de pesos sinápticos do j-ésimo neurônio após adaptação; i é o índice do dado de entrada corrente; $\mathbf{m}_j(i)$ é o vetor de pesos sinápticos do j-ésimo neurônio antes da apresentação do i-ésimo dado de entrada $\mathbf{v}_i$; e $h(c,j,i)$ é uma função monótona decrescente que significa a influência de $\mathbf{v}_i$ sobre o j-ésimo neurônio considerando a distância deste no reticulado em relação ao neurônio vencedor, indicado pelo índice c. Para determinar o neurônio vencedor, Kohonen propôs avaliar a distância euclidiana entre o vetor de pesos sinápticos de cada neurônio e o dado de entrada, de modo que o neurônio que estivesse menos distante seria o vencedor,

$$\|\mathbf{v}_i - \mathbf{m}_c\| = \min_j \left\{\|\mathbf{v}_i - \mathbf{m}_j\|\right\}. \tag{6.8}$$

A influência do estímulo na adaptação das unidades neurais do reticulado, vizinhas ao neurônio vencedor, pode ser definida por uma gaussiana com centro neste neurônio. O desvio padrão dessa gaussiana deve diminuir assintoticamente, tendendo a zero, à medida que os dados de entrada são apresentados à rede, a fim de fazer a rede convergir para esses dados. A influência na adaptação dos vizinhos, portanto, representada pelo termo $h(c,j,i)$ na Equação 6.7, é expressa por

$$h(c, j, i) = h_0 \cdot e^{\frac{-\|\mathbf{r}_j - \mathbf{r}_c\|^2}{\sigma^2}}, \tag{6.9}$$

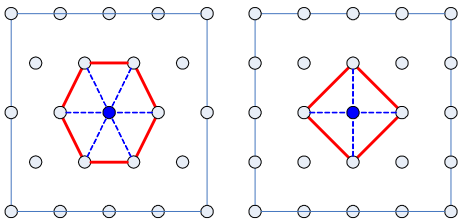
onde $\mathbf{r}_j$ é o vetor posição do j-ésimo neurônio no reticulado; $\mathbf{r}_c$ é o vetor posição do neurônio vencedor; h_0 é a taxa de aprendizagem da rede; e σ é o desvio padrão, denominado raio de vizinhança. A taxa de aprendizagem h_0 e o raio de vizinhança σ decrescem, tendendo a zero, à medida que os dados são apresentados à rede. Neste trabalho, utilizaram-se as funções apresentadas em Ritter et al. (1992):

$$h_0 = \alpha \left(\frac{0,05}{\alpha} \right)^{i/N}, \tag{6.10}$$

onde α é o valor inicial da taxa de aprendizagem, e N, a quantidade total de dados de entrada; e

$$\sigma = \beta \cdot 0,2^{i/N}, \tag{6.11}$$

onde β é o valor inicial do raio de vizinhança. A distância entre dois neurônios é calculada em função da topologia do reticulado. Se a topologia for retangular, o neurônio vencedor terá quatro vizinhos mais próximos (Figura 6.2b), enquanto, se a topologia for hexagonal, terá seis vizinhos (Figura 6.2a). Neste trabalho, implementou-se a topologia retangular (Figura 6.2a).



(a) (b)

○ Representação de um neurônio
● Neurônio vencedor

Figura 6.2 - Exemplos de topologia. (a) Topologia retangular; (b) topologia hexagonal.

Há também outras formas de representar os mapas auto-organizáveis. Por exemplo, para melhorar a precisão numérica, pode-se normalizar os dados de entrada antes de apresentá-los à rede, embora a normalização implique maior custo computacional. Se os dados de entrada forem representados por pontos em uma superfície esférica, o produto interno poderá ser usado para determinar a distância entre esses dados e os neurônios, ao invés da distância euclidiana. Nesse caso, substitui-se a Equação 6.8 por

$$\mathbf{v}_i^T \mathbf{m}_c = \max_j \left\{ \mathbf{v}_i^T \mathbf{m}_j \right\}, \tag{6.12}$$

onde $\|\mathbf{v}_i\|$ e $\|\mathbf{m}_j\|$ são iguais a 1. Para trabalhar com pesos normalizados, substitui-se a Equação 6.7 por

$$\mathbf{m}_j(i+1) = \frac{\mathbf{m}_j(i) + h(c, j, i)\left[\mathbf{v}_i - \mathbf{m}_j(i)\right]}{\left\| \mathbf{m}_j(i) + h(c, j, i)\left[\mathbf{v}_i - \mathbf{m}_j(i)\right] \right\|}. \tag{6.13}$$

Os mapas auto-organizáveis também comportam-se como processos markovianos, em que o termo $h(c,j,i)$ é tratado como uma função de probabilidade que define a transição do mapa de um estado para outro. O estímulo é em essência a realização de uma variável aleatória. Os mapas aproximam-se da distribuição de probabilidade dessa variável aleatória, pois as realizações de maior probabilidade sensibilizam regiões maiores do mapa, enquanto as de menor probabilidade sensibilizam regiões menores (MEHRA, 1992). A representação da distribuição dos dados de entrada pela distribuição dos neurônios é chamada de quantização vetorial, compactação, ou prototipagem, considerando que a quantidade de neurônios seja menor do que a de dados de entrada.

Há duas formas de implementar os mapas: incremental e em lote. Este estudo apresenta, em detalhes, a forma incremental. A forma incremental é sensível à ordem de apresentação das entradas. A implementação em lote é uma abordagem com o objetivo de resolver esse problema. Sua desvantagem é que a adaptação dos pesos só ocorre depois de todos os dados de entrada terem sido apresentados à rede. Apesar de a definição algébrica dos mapas ser simples, sua implementação poderá ser complicada se for necessário construir uma interface gráfica, a fim de representar o reticulado, dotada

de funcionalidades para analisar o estado dos neurônios e permitir a identificação de agrupamentos no espaço de atributos.

A seguir, apresenta-se, detalhadamente, o procedimento da forma incremental.

1) Atribua valores iniciais à matriz de pesos da rede (passos de (2) a (10)). Neste trabalho, usou-se um mapa logístico, em regime caótico, para gerar esses valores iniciais. Seja x uma variável, Q o número total de neurônios e D o número de conexões sinápticas por neurônio.

2) Atribua um valor inicial para x, por exemplo, faça $x \leftarrow 0{,}001$.

3) Faça $j \leftarrow 0$.

4) Faça $j \leftarrow j+1$.

5) Faça $k \leftarrow 0$.

6) Faça $k \leftarrow k+1$.

7) Faça $x \leftarrow 4x(1-x)$.

8) Faça $M(j,k) \leftarrow x$, onde M é a matriz de pesos sinápticos.

9) Retorne para (6) até que k seja igual a D.

10) Retorne para (4) até que j seja igual a Q. Para o bom desempenho da rede, é importante que não haja, inicialmente, neurônios com vetores de pesos sinápticos iguais.

11) Realize a adaptação da rede (passos de (12) a (61)).

12) Faça $I \leftarrow \emptyset$, onde I é o conjunto dos índices de referência dos dados apresentados à rede.

13) Escolha um dado de entrada, podendo ser de forma aleatória, ou na ordem em que os dados estão dispostos (passos de (14) a (18)). Seja $V_{N,D}$ a

representação matricial do conjunto dos dados de entrada, onde N é o número total de dados e D é o número de componentes dos dados de entrada (também o número de conexões sinápticas dos neurônios). Seja i o índice do dado selecionado, tal que $i \notin I$.

14) Faça $I \leftarrow i$.

15) Faça $k \leftarrow 0$.

16) Faça $k \leftarrow k+1$.

17) Faça $\mathbf{v}(k) \leftarrow V(i,k)$.

18) Retorne para (16) até que k seja igual a D.

19) Determine o neurônio vencedor (passos de (20) a (34)).

20) Faça $d_{\min} \leftarrow 0$.

21) Faça $k \leftarrow 0$.

22) Faça $k \leftarrow k+1$.

23) Faça $d_{\min} \leftarrow d_{\min}+[M(1,k)-\mathbf{v}(k)]^2$.

24) Retorne para (22) até que k seja igual a D.

25) Faça $c \leftarrow 1$.

26) Faça $j \leftarrow 1$.

27) Faça $j \leftarrow j+1$.

28) Faça $d \leftarrow 0$.

29) Faça $k \leftarrow 0$.

30) Faça $k \leftarrow k+1$.

31) Faça $d \leftarrow d+[M(j,k)-\mathbf{v}(k)]^2$.

32) Retorne para (30) até que k seja igual a D.

33) Se $d < d_{\min}$, faça $d_{\min} \leftarrow d$ e $c \leftarrow j$.

34) Retorne para (27) até que j seja igual a Q.

35) Determine a posição do neurônio vencedor em função da topologia da rede (passos de (36) a (47)). A Figura 6.3 apresenta dois exemplos de topologia, a retangular e a hexagonal, cada qual com 25 neurônios, e a indicação das coordenadas r_x e r_y do décimo terceiro neurônio na topologia retangular. Seja *col* o número de colunas de neurônios que constituem o reticulado. Nos dois exemplos da Figura 6.3, *col* é igual a 5. Seja c o índice do neurônio vencedor, r_{cx} sua posição horizontal no reticulado e r_{cy} sua posição vertical.

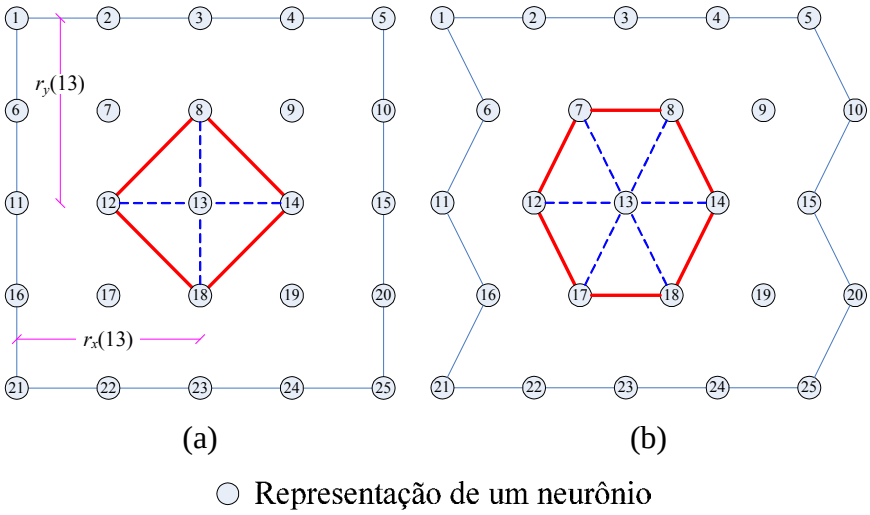


Figura 6.3 - Coordenadas do neurônio no reticulado. (a) Topologia retangular; (b) topologia hexagonal.

36) Se a topologia for retangular, realize os passos de (37) a (39).

37) Se o resto q da divisão de c por *col* for maior do que zero, faça $r_{cx} \leftarrow q$.

38) Senão, faça $r_{cx} \leftarrow col$.

39) Faça $r_{cy} \leftarrow \left\lceil \frac{c}{col} \right\rceil$.

40) Se a topologia for hexagonal, realize os passos de (41) a (47).

41) Se o resto p da divisão de c por 2 for maior do que zero, realize os passos de (42) a (43).

42) Se o resto q da divisão de c por col for maior do que zero, faça $r_{cx} \leftarrow q$.

43) Senão, faça $r_{cx} \leftarrow col$.

44) Senão, realize os passos de (45) a (46).

45) Se o resto q da divisão de c por col for maior do que zero, faça $r_{cx} \leftarrow q+0{,}5$.

46) Senão, faça $r_{cx} \leftarrow col+0{,}5$.

47) Faça $r_{cy} \leftarrow \frac{\sqrt{3}}{2}\left\lceil \frac{c}{col} \right\rceil$.

48) Calcule h_0 e σ em função do índice i (Equações 6.10 e 6.11).

49) Adapte a rede neural (passos de (50) a (59)). Seja r_x a posição horizontal do neurônio no reticulado, r_y sua posição vertical, h a influência na adaptação dos neurônios vizinhos ao vencedor e d_r o quadrado da distância de um neurônio em relação ao vencedor no reticulado.

50) Faça $j \leftarrow 0$.

51) Faça $j \leftarrow j+1$.

52) Obtenha a posição do j-ésimo neurônio. Para tanto, execute o passo (35), substituindo c por j, r_{cx} por r_x e r_{cy} por r_y.

53) Faça $d_r \leftarrow (r_x-r_{cx})^2+(r_y-r_{cy})^2$.

54) Faça $h \leftarrow h_0 \cdot e^{-d_r / \sigma^2}$.

55) Faça $k \leftarrow 0$.

56) Faça $k \leftarrow k+1$.

57) Faça $M(j,k) \leftarrow M(j,k)+h[\mathbf{v}(k)-M(j,k)]$.

58) Retorne para (56) até que k seja igual a D.

59) Retorne para (51) até que j seja igual a Q.

60) Se $\#I < N$, retorne para (13).

61) Senão, finalize a adaptação da rede.

No procedimento em lote, as unidades acumulam as contribuições parciais de cada entrada

$$\Delta\mathbf{m}_j(i+1) = \Delta\mathbf{m}_j(i) + e^{\frac{-\|\mathbf{r}_j - \mathbf{r}_c\|^2}{\sigma^2}} \left[\mathbf{v}_i - \mathbf{m}_j(i)\right],\ i = 1, 2, ..., N\ . \tag{6.14}$$

A soma das contribuições, de acordo com a Equação 6.14, define uma época de treinamento. Ao término de uma época, as unidades são adaptadas de acordo com a equação

$$\mathbf{m}_j(it+1) \leftarrow \mathbf{m}_j(it) + \frac{1}{N} h_0(it)\Delta\mathbf{m}_j,\ it = 1, 2, ..., it_{\max}\ . \tag{6.15}$$

Os resultados dos mapas podem ser apresentados de diversas formas. Por exemplo, pode-se representá-los através de cores, no reticulado, usando o esquema RGB. A combinação das componentes RGB é definida em função dos pesos. Pode-se, também, apresentá-los através de tons de cinza, onde a intensidade do cinza é determinada, por exemplo, em função da soma das distâncias euclidianas do neurônio em relação a determinado número de neurônios mais próximos no espaço de atributos. Pode-se

representar os mapas no espaço cartesiano, a fim de visualizar a convergência da rede para a distribuição dos dados.

Os valores atribuídos inicialmente aos pesos impactam no tempo de adaptação da rede. Adotam-se, então, artifícios para reduzir o tempo de convergência, gerando pesos iniciais, aleatoriamente, com distribuição normal, considerando a matriz de covariâncias Σ_v da distribuição dos dados de entrada. Dado um conjunto de entradas $V \sim N(\boldsymbol{\mu}_v, \Sigma_v)$ e um gerador padrão $Z \sim N(\mathbf{0}, I)$, onde $\mathbf{0}$ é o vetor nulo e I a matriz identidade com dimensão igual à da matriz Σ_v, obtêm-se, através da equação abaixo, os valores iniciais da matriz de pesos $M=[\mathbf{m}_j]$, com base em Σ_v, os quais são mais propícios à adaptação da rede, acelerando a convergência.

$$M = P\Lambda^{1/2}Z + \boldsymbol{\mu}_v, \tag{6.16}$$

onde P é a matriz de autovetores e Λ a matriz diagonal de autovalores de Σ_v. Pode-se, ainda, adotar um gerador padrão Z baseado no método LHS[23], para distribuição normal, a fim de garantir uma boa representação da variabilidade do estado inicial dos neurônios, normalmente, em pouca quantidade, se comparada à dos dados de entrada. Neste estudo, no entanto, implementou-se o procedimento mais simples, a escolha aleatória dos pesos sem levar em consideração a distribuição das entradas.

6.2.4 Teorema de Bayes

A inferência estatística tem sido amplamente empregada devido ao trabalho pioneiro de Thomas Bayes. Em seu artigo "*An essay towards solving a problem in the doctrine of chances*", ele definiu o conceito de probabilidade condicional, do qual deriva o teorema de Bayes (YUEN, 2010). Usando A e B para denotar dois eventos quaisquer, a probabilidade condicional $P(A|B)$, a qual denota a probabilidade do evento A ocorrer dado que observa-se a ocorrência do evento B, é expressa por

[23] O método LHS é descrito na Subseção 5.2.4.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \tag{6.17}$$

onde $P(B) > 0$; e $P(A \cap B)$ denota a probabilidade de ocorrência simultânea de ambos os eventos. Dado que $P(A \cap B)$ é igual a $P(B \cap A)$, tem-se o teorema de Bayes expresso por

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{6.18}$$

O teorema de Bayes, portanto, expressa a relação entre uma probabilidade condicional e a sua inversa. Conforme o teorema da probabilidade total, se o espaço amostral é subdividido em N eventos mutuamente excludentes, A_1, A_2, ..., A_N, então a probabilidade de qualquer evento B desse espaço amostral pode ser expressa por

$$P(B) = \sum_{k=1}^{N} P(B \cap A_k). \tag{6.19}$$

Aplicando o conceito de probabilidade condicional na Equação 6.19, tem-se que

$$P(B) = \sum_{k=1}^{N} P(B|A_k)P(A_k). \tag{6.20}$$

Com base nas Equações 6.18 e 6.20, tem-se o teorema de Bayes para eventos discretos expresso por

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_k P(B|A_k)P(A_k)}. \tag{6.21}$$

6.3 Sumarização de resultados de otimização

A sumarização de resultados de otimização consiste em estimar, antes, a probabilidade de um perfil de desempenho, observando-se um perfil de projeto, e, em seguida, a probabilidade condicional inversa, aplicando o teorema de Bayes (Equação 6.21). Nesta abordagem, as probabilidades do lado direito da Equação 6.21 são estimadas através de

aproximações quantizadas da distribuição das soluções de Pareto no domínio da função multiobjetivo. O LVQ (*learning vector quantization*) foi considerado inicialmente, mas optou-se por seu precursor, o SOM (*self-organizing map*), junto com o k-médias. Tanto o LVQ quanto o SOM foram propostos por Kohonen. O SOM é mais adequado para agrupamento e visualização (KOHONEN, 2001). Um recurso do SOM que o LVQ não oferece é o de permitir a análise da topologia do espaço de atributos, espaço onde são definidos os dados de entrada. Ao contrário do LVQ, o SOM é não-supervisionado, permitindo que se explore a distribuição das soluções de Pareto antes de agrupá-las por semelhança. Emprega-se um mapa auto-organizável (SOM 1) para quantizar a distribuição do conjunto de Pareto no domínio da função multiobjetivo, e outro mapa (SOM 2) para quantizar a distribuição da fronteira de Pareto no contradomínio (Figura 6.4). Faz-se o agrupamento com o auxílio do k-médias. Através do SOM, identificam-se as partições existentes. Para cada partição identificada no reticulado de cada SOM, seleciona-se um neurônio como seu centroide inicial. Então, executa-se o k-médias a fim de agrupar os neurônios.

A Figura 6.4 ilustra o procedimento de quantização de distribuições usando os mapas auto-organizáveis e o método k-médias. As reticências que aparecem em X e Y foram inseridas para denotar que a quantização não depende da dimensão desses espaços, sugerindo que podem ser espaços multidimensionais. Para estimar a probabilidade de um perfil de desempenho, observando-se um perfil de projeto, mapeiam-se os perfis de projeto sobre os perfis de desempenho, conta-se o número de incidências de um perfil sobre o outro, através da representação reticular do SOM 1 (Figura 6.5), e, então, aplica-se a Equação 6.22.

Como mostra a Figura 6.5, cada neurônio do SOM 1 é avaliado pela função multiobjetivo (F) e o resultado apresentado ao SOM 2 que o classifica, ativando o neurônio mais próximo a esse resultado. Então, o índice do perfil de desempenho, ao qual esse neurônio pertence, é copiado no neurônio do SOM 1 responsável por sua ativação, indicando que aquele neurônio incide sobre aquele índice de perfil de desempenho. Se um modelo substituto tiver sido usado na otimização, esse modelo também poderá ser usado no lugar da função multiobjetivo (F) para mapear os perfis de

projeto sobre os de desempenho. Usando o modelo substituto, evita-se a dificuldade associada ao custo computacional da função multiobjetivo.

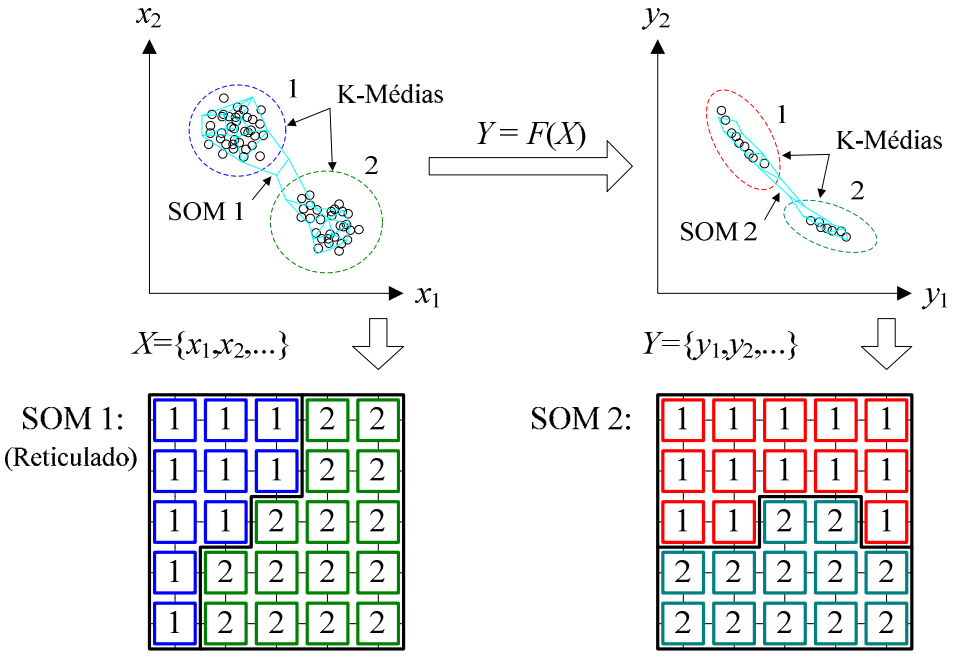


Figura 6.4 - Quantização da distribuição das soluções de Pareto no domínio (X) e no contradomínio da função multiobjetivo (F).

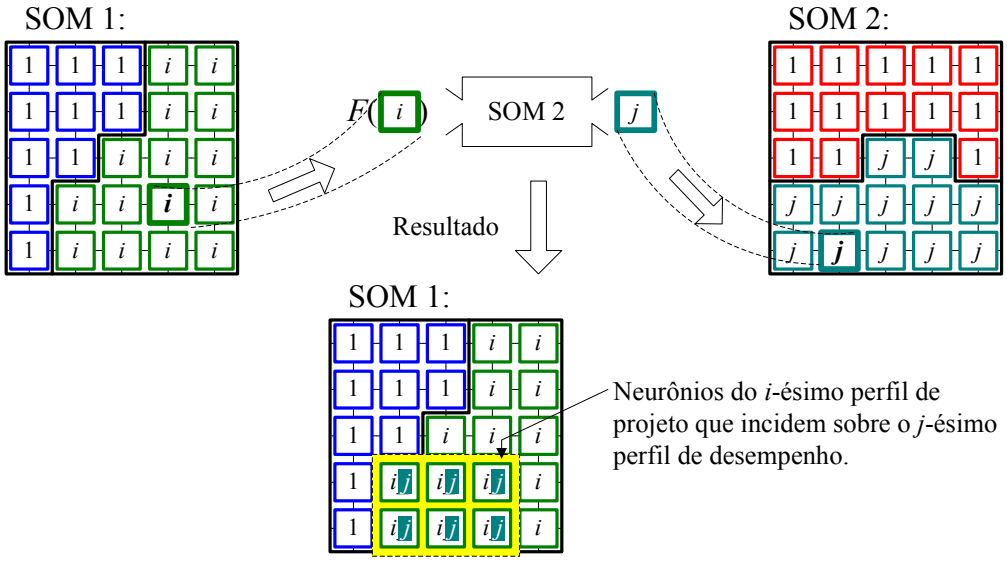


Figura 6.5 - Mapeamento dos perfis de projeto sobre os perfis de desempenho.

A estimativa da probabilidade de um perfil de desempenho, observando-se um perfil de projeto, é dada por

$$P\left(B_j \mid A_i\right) \cong \frac{N_{ij}}{\sum_k N_{ik}}, \tag{6.22}$$

onde A_i é o i-ésimo perfil de projeto, B_j é o j-ésimo perfil de desempenho, N_{ij} é o número de neurônios do SOM 1 pertencentes a A_i que incidem sobre B_j. Para estimar a probabilidade condicional inversa $P(A_i|B_j)$, é necessário estimar a probabilidade a priori de cada perfil de projeto, expressa por

$$P\left(A_i\right) \cong \frac{1}{M} \sum_k N_{ik}, \tag{6.23}$$

onde M é o número de neurônios do SOM 1. Finalmente, com base nas estimativas obtidas através das Equações 6.22 e 6.23, tem-se a estimativa da probabilidade condicional inversa, expressa por

$$P\left(A_i \mid B_j\right) = \frac{P\left(B_j \mid A_i\right) P\left(A_i\right)}{\sum_k P\left(B_j \mid A_k\right) P\left(A_k\right)}. \tag{6.24}$$

6.4 Estudo de caso

6.4.1 Descrição

Para demonstrar a abordagem de sumarização de resultados de otimização, tomou-se, como exemplo, o conjunto de soluções de Pareto resultante da minimização da função F (Tabela 4.1). Usando uma função com apenas duas variáveis e dois objetivos, foi possível visualizar mais facilmente a convergência dos mapas auto-organizáveis através de representações cartesianas da distribuição dos pesos sobre a distribuição das soluções (Figura 6.6). Considerou-se a rede perceptron de múltiplas camadas (MLP) como modelo substituto da função F, com o objetivo de demonstrar esta abordagem com o uso dessa técnica de otimização multidisciplinar apresentada no Capítulo 5.

Executaram-se, sequencialmente, os seguintes procedimentos: regressão, agrupamento, mapeamento e sumarização. Na regressão, treinou-se a MLP, definida com 5 camadas escondidas e 12 neurônios em cada camada. No treinamento da MLP, adotou-se o algoritmo *back-propagation*[24], considerando a taxa de aprendizagem e o momento monotonicamente decrescentes. Consideraram-se 1264 soluções de Pareto da função *F*. Então, geraram-se, com essas soluções, dois conjuntos de dados para treinamento, onde 50% dos dados foram usados para estimar os parâmetros da MLP e os outros 50% para selecionar os valores desses parâmetros que implicavam melhor generalização da rede, usando para isso o método da validação cruzada. Utilizou-se a tangente hiperbólica como função de ativação dos neurônios da MLP, de acordo com:

$$\varphi(v) = \frac{1-e^{-v}}{1+e^{-v}}, \tag{6.25}$$

onde v é o campo local induzido (Equação 5.5).

No agrupamento, treinaram-se os mapas SOM 1 e 2. Para cada SOM, adotou-se a topologia retangular, contendo 100 unidades, distribuídas em 10 linhas e 10 colunas. Os mapas SOM 1 e 2 foram treinados com 500000 épocas, taxa de aprendizagem igual a 0,01 e raio de vizinhança igual a 3.

No mapeamento, procedeu-se conforme ilustrado na Figura 6.5, até que todos os neurônios do SOM 1 contivessem o índice do perfil de desempenho sobre o qual incidiram.

Na sumarização, estimou-se a probabilidade de um perfil de desempenho, observando-se cada perfil de projeto, aplicando a Equação 6.22, estimou-se, em seguida, a probabilidade a priori de cada perfil de projeto, aplicando a Equação 6.23, e, por último, com base nessas avaliações, estimou-se a probabilidade de um perfil de projeto, observando-se um perfil de desempenho, aplicando a Equação 6.24.

---

[24] O algoritmo *back-propagation* é descrito em Haykin (1999), p. 183.

Desenvolveu-se um programa[25] para demonstrar essa abordagem, incorporando a rede perceptron de múltiplas camadas (MLP) como modelo substituto da função objetivo original, a fim de mapear os perfis de projeto sobre os de desempenho.

6.4.2 Discussão dos resultados

Alguns neurônios do SOM 1 não convergiram para a distribuição das entradas (Figura 6.6a). Essas unidades delimitam os agrupamentos de soluções no conjunto de Pareto, como pode ser visto no reticulado do SOM 1 (Figura 6.7a).

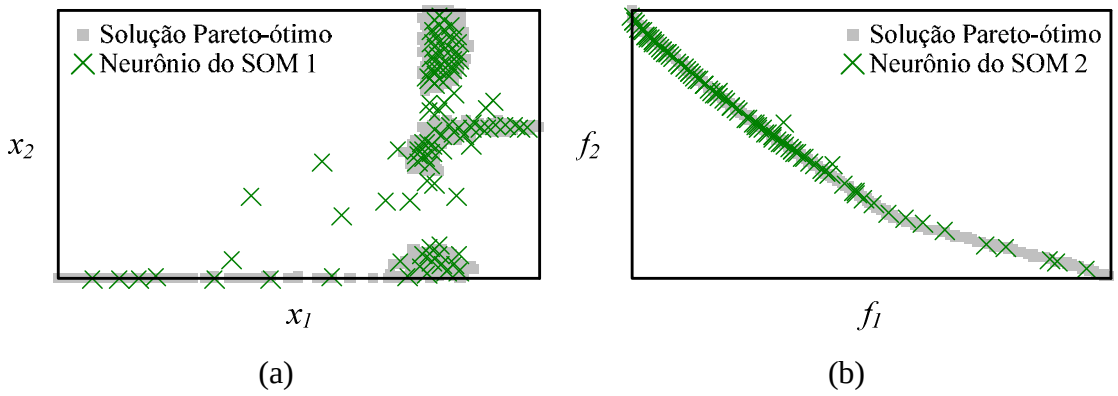


Figura 6.6 - Convergência dos mapas SOM 1 e 2. (a) Conjunto de Pareto; (b) fronteira de Pareto.

No reticulado do SOM 1, verifica-se a existência de agrupamentos no espaço de atributos através da observação de fileiras de neurônios demarcadas por tons de cinza um pouco mais acentuados do que os dos neurônios vizinhos (Figura 6.7a). O tom de cinza indica o quão afastado está o neurônio de seus vizinhos no espaço de atributos. Quanto mais claro, menor é a distância euclidiana da unidade em relação a determinado número de vizinhos. Quanto mais escuro, maior é a distância euclidiana (Figura 6.7). Constata-se que há três agrupamentos bem definidos no SOM 1.

No SOM 2, não se observa a divisão das soluções em agrupamentos (Figura 6.7b). Verifica-se, na Figura 6.6, que o número de agrupamentos identificados nos reticulados está correto.

---

[25] As telas do programa de sumarização de resultados de otimização são apresentadas no Apêndice C.

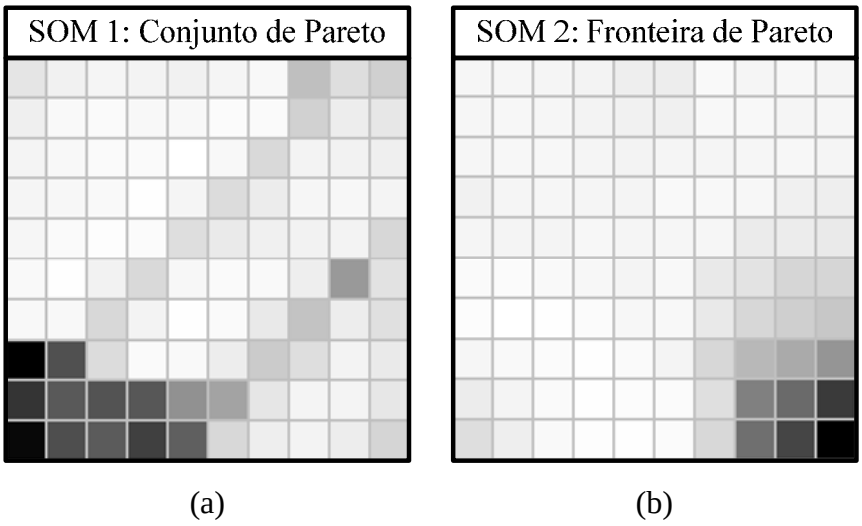


Figura 6.7 - Reticulado dos mapas SOM 1 e 2 após adaptação. (a) Reticulado do SOM 1; (b) reticulado do SOM 2.

No agrupamento, definem-se as partições através da discretização dos mapas, usando-se o método k-médias (Figura 6.8). Selecionaram-se, como centroides, os neurônios que estavam localizados no centro dos agrupamentos identificados em cada reticulado. No SOM 1, selecionaram-se quatro centroides, um a mais do que o número de agrupamentos identificados, a fim de aumentar a discretização do conjunto de Pareto. No SOM 2, escolheram-se dois centroides, a fim de discretizar a fronteira de Pareto. A Figura 6.8 apresenta o resultado da discretização dos mapas usando o k-médias.

A escolha de um número de centroides maior do que o de agrupamentos identificados tem a ver com o fato de o k-médias ser restrito a agrupamentos convexos. Ao discretizar os mapas SOM 1 e 2, buscou-se um número adequado de centroides a fim de limitar a aplicação do k-médias a porções convexas da distribuição das unidades neurais (Figura 6.6). Se o número de centroides não fosse adequado, o resultado da discretização mostraria grupos disjuntos de neurônios, mas agrupados em um mesmo perfil.

A Figura 6.9 apresenta o resultado do mapeamento dos neurônios do SOM 1 sobre os do SOM 2, usando a MLP. Dois índices foram atribuídos a cada unidade do SOM 1 (Figura 6.9). O primeiro índice informa o perfil de projeto e o segundo, o de desempenho. A Tabela 6.1 apresenta a estimativa da probabilidade dos perfis de desempenho, observando-se cada perfil de projeto, obtida com a Equação 6.22.

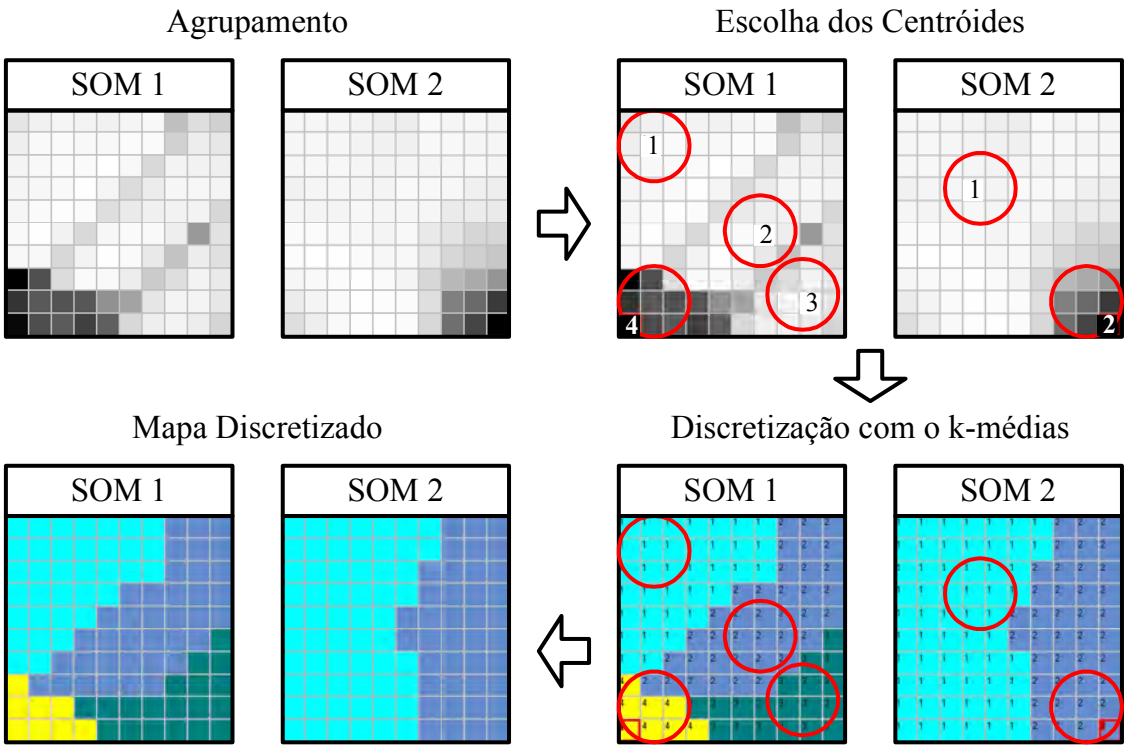


Figura 6.8 - Discretização do SOM 1 e 2.

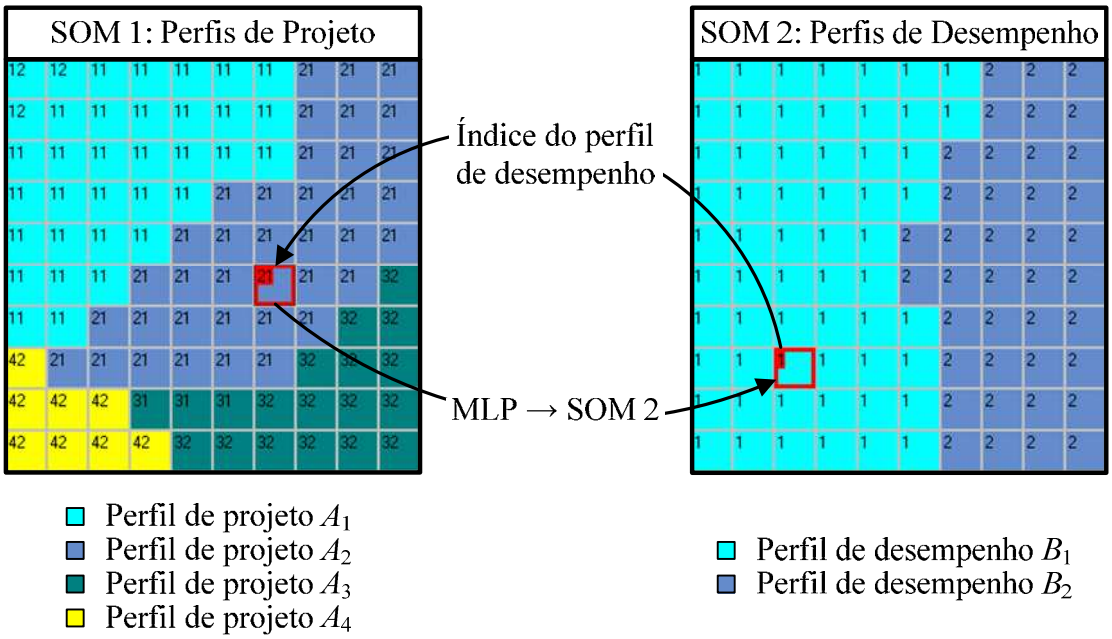


Figura 6.9 - Mapeamento dos neurônios do SOM 1 sobre os do SOM 2.

Tabela 6.1 - Estimativa da probabilidade dos perfis de desempenho, observando-se cada perfil de projeto.

Perfis de projeto	Perfis de desempenho	
	B_1	B_2
A_1	0,91	0,09
A_2	1,00	0,00
A_3	0,16	0,84
A_4	0,00	1,00

A Tabela 6.2 apresenta a estimativa da probabilidade a priori dos perfis de projeto, obtida com a Equação 6.23. Inserindo os valores das Tabelas 6.1 e 6.2 na Equação 6.24, tem-se a estimativa da probabilidade dos perfis de projeto, observando-se os perfis de desempenho (Tabela 6.3).

Tabela 6.2 - Estimativa da probabilidade a priori dos perfis de projeto.

Perfil de projeto	Estimativa da probabilidade a priori
A_1	0,35
A_2	0,38
A_3	0,19
A_4	0,08

Da Tabela 6.3, obtém-se, portanto, que o perfil de projeto mais provável observando-se o perfil de desempenho B_1 é o A_2, e que observando-se o B_2 é o A_3.

Tabela 6.3 - Estimativa da probabilidade dos perfis de projeto, observando-se os perfis de desempenho.

Perfis de desempenho	Perfis de projeto			
	A_1	A_2	A_3	A_4
B_1	0,44	0,52	0,04	0
B_2	0,11	0	0,59	0,30

Esta abordagem, no entanto, ainda carece de aprimoramentos. Por exemplo, os neurônios que não convergiram para a distribuição dos dados foram considerados na contagem do número de incidências de um perfil de projeto sobre outro de desempenho, interferindo na estimativa das probabilidades apresentadas na Tabela 6.3 (Figura 6.10).

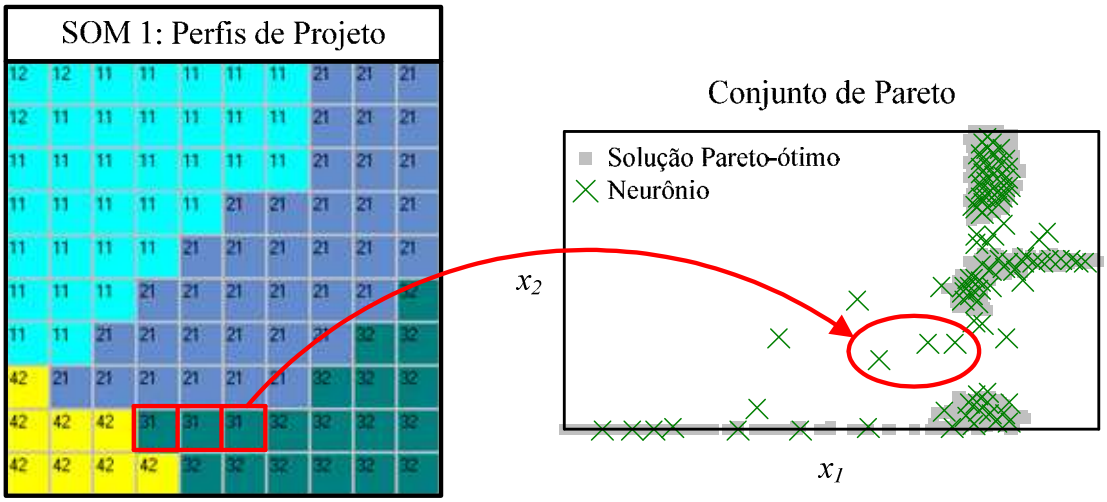


Figura 6.10 - Neurônios dispersos atribuídos erroneamente a um perfil de projeto.

Além de eliminar a atribuição errônea desses neurônios dispersos, sugere-se combinar esta abordagem com o método proposto por Obayashi e Sasaki (2003). Segundo esses autores, o projeto é um processo através do qual, dado um ponto no espaço da função objetivo (contradomínio), busca-se um ponto associado a ele no espaço das variáveis de projeto (domínio). Se a dimensão do espaço das variáveis de projeto for bastante elevada, encontrar esse ponto poderá ser uma tarefa muito difícil. Uma forma de atenuar essa dificuldade é agrupar as variáveis de projeto para que possam ser analisadas juntas, de uma única forma. Essa é a proposta de Obayashi e Sasaki (2003), que consiste em usar um mapa auto-organizável para agrupar as variáveis de projeto. Para tanto, eles aplicam outro mapa para agrupar as soluções no espaço da função objetivo (contradomínio), tal como o SOM 2 é aplicado nesta abordagem. Portanto, eles usam dois mapas, um para visualizar os compromissos de projeto, e outro para minerar as variáveis de projeto. No entanto, o segundo mapa proposto por esses autores não equivale ao SOM 1 desta abordagem. Seja, por exemplo, o mapa proposto por Obayashi e Sasaki (2003) denotado por SOM 1'. Ao invés de este ser usado para encontrar

agrupamentos de soluções no domínio da função, como faz o SOM 1, o SOM 1' é usado para encontrar similaridades entre as variáveis de projeto, baseadas na correspondência dessas variáveis com as partições identificadas no SOM 2 (um mapa similar ao desta abordagem).

Primeiro, o SOM 2 é discretizado considerando um número maior de partições. Executa-se, então, o procedimento a seguir para cada variável de projeto. Seja, por exemplo, x uma variável de projeto. Para cada partição do SOM 2, obtém-se o valor médio de x correspondente. Isso é feito ativando cada neurônio do SOM 2 por um conjunto de soluções. Cada solução corresponde a um ponto no domínio da função objetivo. Obtém-se, então, o valor médio de x relativo a cada neurônio do SOM 2 considerando as soluções que ativam o neurônio por estarem mais próximas.

Em seguida, obtém-se o valor médio de x para cada partição do SOM 2 considerando os valores médios de x relativos aos neurônios da partição. Então, após executar esse procedimento para todas as variáveis de projeto, cada variável terá um vetor associado a ela, onde as componentes desse vetor são os valores médios dessa variável que estão associados às partições do SOM 2. Aplica-se, então, o SOM 1' no espaço definido por esses vetores, com o objetivo de agrupar as variáveis de projeto por similaridade, observando seus valores médios associados às partições do SOM 2. Por fim, em cada agrupamento identificado no SOM 1', seleciona-se uma variável de projeto para representar o grupo. Mantendo-se os contornos iniciais do reticulado do SOM 2, visualiza-se, através de uma representação em tons de cores, a magnitude de cada uma dessas variáveis selecionadas. Cada qual representa um grupo de variáveis de projeto que são similares a ela, reduzindo, dessa forma, o esforço da análise dos resultados da otimização.

7 CONCLUSÃO

7.1 Resumo dos resultados principais

Esta pesquisa de doutorado em computação aplicada contribuiu com três abordagens para suporte à otimização multidisciplinar, baseadas em métodos de inteligência computacional, com o objetivo de aplicá-las na otimização de foguetes de sondagem. A baixa sinergia entre as disciplinas de aerodinâmica e trajetória no desenvolvimento do foguete VS-40 motivou a realização deste trabalho.

Quanto ao aspecto prático desta pesquisa, desenvolveu-se o programa MDO-SONDA, com o objetivo de realizar estudos de otimização multidisciplinar de foguetes de sondagem. Acoplaram-se dois programas ao MDO-SONDA, um de aerodinâmica, o *missile datcom*, e outro de trajetória, o ROSI. O NGMOGA com o ORAD foi incorporado no MDO-SONDA. Como estudo de caso, realizou-se a otimização das empenas do foguete VS-40. O tempo de execução da otimização indicou que o custo computacional de estudos mais complexos, envolvendo mais variáveis e objetivos, poderia se tornar um problema. Constatou-se que os objetivos do problema eram conflitantes, pois a distribuição das soluções mostraram a formação de uma curva de Pareto. Esse resultado foi relevante pois revelou que há um compromisso entre o arrasto provocado pelas empenas e o menor intervalo entre os instantes críticos do voo. Constatou-se, também, que, apesar de as soluções de Pareto das empenas terem uma área um pouco maior do que as empenas originais, o arrasto produzido por elas era menor. O estudo indicou, portanto, que as empenas do VS-40 eram subótimas, sob a ótica das disciplinas de aerodinâmica e trajetória.

A primeira contribuição, descrita no Capítulo 4, consistiu de uma abordagem memética baseada na cooperação entre o algoritmo genético multiobjetivo não-geracional (NGMOGA) e o algoritmo da otimização extrema generalizada (M-GEO). Propôs-se, para uso em ambos os algoritmos um operador real, inspirado no operador binário do M-GEO, denominado operador real de assimetria dinâmica (ORAD). Os resultados da comparação do NGMOGA com o ORAD, o operador real SBX com a mutação uniforme de Michalewicz, e os operadores binários de recombinação e mutação,

originalmente incorporados nesse algoritmo, mostraram que o ORAD era tão bom quanto os operadores binários e mais robusto e eficiente do que o SBX com a mutação uniforme de Michalewicz. Os resultados também indicaram que o M-GEO com o ORAD teve um desempenho melhor do que com o operador binário. Resultados expressivos foram alcançados com a abordagem memética reduzindo-se o número de torneios e aumentando-se o número de sementes do M-GEO. Os resultados do ORAD na abordagem memética foram mais expressivos do que os obtidos com os operadores binários, pois, com o ORAD, realizaram-se menos avaliações da função objetivo por solução não-dominada, e teve-se um percentual menor de soluções dominadas.

A segunda contribuição, descrita no Capítulo 5, consistiu de um procedimento para estimar a eficiência da rede perceptron de múltiplas camadas (MLP) e da rede de função de base radial (RBFN) como modelos substitutos, considerando as características do NGMOGA, método de otimização adotado. Implementou-se a RBFN com maior facilidade do que a MLP, e constatou-se, além disso, que a RBFN era mais rápida e mais robusta do que a MLP. A RBFN foi eficiente como modelo substituto na otimização da 1ª e 2ª função de teste, enquanto a eficiência da MLP caiu com a complexidade do problema de otimização, ficando abaixo da RBFN.

As duas primeiras contribuições exploraram abordagens para reduzir a quantidade de avaliações da função objetivo, com o propósito de tornar a otimização mais eficiente.

A terceira contribuição, descrita no Capítulo 6, teve o propósito de explorar soluções de Pareto definidas em espaços multidimensionais, e consistiu de uma abordagem para sumarizar dados de otimização. Desenvolveu-se um programa para demonstrar o uso dessa abordagem, incorporando a MLP como modelo substituto a fim de mapear os perfis de projeto sobre os de desempenho. Os resultados mostraram que os mapas SOM 1 e SOM 2 convergiram corretamente para a distribuição das soluções de Pareto, indicando o número de partições existentes no domínio e no contradomínio da função objetivo. No entanto, ressaltou-se a principal limitação dessa abordagem. Os neurônios que não convergiram para a distribuição dos dados foram considerados na contagem do

número de incidências de um perfil de projeto sobre outro de desempenho, interferindo na estimativa das probabilidades.

7.2 Recomendações para trabalhos futuros

7.2.1 Abordagem memética

Apesar de o estudo de caso ter indicado que a abordagem memética baseada no NGMOGA e no M-GEO é uma ideia válida, esta abordagem ainda carece de análises exaustivas a fim de também indicar que é uma abordagem promissora diante de outras mais recentes (MOUSSI et al., 2012; ZHANG; WU, 2012; AKPINAR et al., 2013). Sugere-se, portanto, para trabalhos futuros, as seguintes ações:

a) analisar a abordagem exaustivamente com funções de teste;

b) compará-la com outras abordagens de otimização;

c) avaliar sua complexidade computacional;

d) explorar o paralelismo intrínseco dos métodos;

e) investigar a fundo o compromisso entre a configuração dos métodos a fim de obter o melhor desempenho possível da abordagem memética; e

f) incorporar no procedimento de comparação dos operadores um artifício para levar em consideração, automaticamente, a distribuição dos pontos ao longo da fronteira de Pareto, como proposto, por exemplo, no procedimento para avaliar a eficiência de modelos substitutos (Seção 5.3).

7.2.2 Modelos substitutos

Pesquisas futuras podem incidir sobre os seguintes tópicos:

a) comparação entre o *kriging* e a RBFN, aplicando a abordagem para avaliação da eficiência de modelos substitutos apresentada neste trabalho;

b) análise exaustiva com mais funções de teste;

c) avaliação da eficiência de modelos substitutos considerando problemas de otimização com muitas variáveis e objetivos;

d) generalização da definição de células para avaliação de distribuições definidas em espaços multidimensionais;

e) investigação de como a eficiência do modelo substituto é afetada pelas definições do algoritmo de otimização, pela probabilidade de substituição e pelo número de ciclos de otimização;

f) busca exaustiva por configurações mais promissoras da MLP e da RBFN, estendendo os limites de pesquisa de seus parâmetros; e

g) estudo do uso de redes neurais construtivas como modelos substitutos, tais como a *cascade-correlation* (FAHLMAN; LEBIERE, 1990).

7.2.3 Sumarização de resultados de otimização

Recomenda-se, para trabalhos futuros, as seguintes ações:

a) eliminar a atribuição errônea de neurônios dispersos a perfis de projeto e de desempenho;

b) substituir o SOM 2 por um simples método de classificação, caso a visualização da topologia da fronteira de Pareto não seja necessária; e

c) combinar a abordagem de sumarização de resultados de otimização com o método apresentado por Obayashi e Sasaki (2003).

7.2.4 Otimização de foguetes de sondagem

Sugerem-se quatro linhas de desenvolvimento para trabalhos futuros.

1) A primeira linha consiste em acrescentar novas funcionalidades ao MDO-SONDA. Por exemplo, interfaces podem ser criadas para permitir a comparação, por meio de gráficos, das configurações do foguete em cada fase de voo, inclusive adicionando sua visualização em três dimensões, e também permitir a customização de gráficos para examinar os resultados dos programas das disciplinas. É importante que se desenvolva uma interface para que o usuário possa definir o problema de otimização e configurar a interação com os programas das disciplinas, ao invés de ter que adicionar linhas de código ao MDO-SONDA. O MDO-SONDA deverá ser capaz de calcular as propriedades de massa e inércia dos subsistemas submetidos à otimização, se houver alteração de sua geometria. A sumarização de resultados de otimização, apresentada no Capítulo 6, poderá ser incorporada posteriormente no MDO-SONDA, a fim de auxiliar na identificação dos compromissos entre os objetivos conflitantes de projeto, se a análise convencional através de gráficos comuns não for suficiente para torná-los explícitos.

2) A segunda linha consiste em acoplar outros programas, envolvendo mais disciplinas no problema de otimização. Por exemplo, trabalhos em equipe envolvendo especialistas de propulsão e estruturas podem contribuir com programas para gerar a curva de empuxo dos motores a partir das propriedades variáveis do propelente, e para estimar a resistência estrutural do foguete contra os carregamentos aerodinâmicos que ocorrem durante o voo. Além disso, pode-se estudar a propagação do erro inerente a esses programas no resultado da otimização.

3) A terceira linha consiste em incorporar métodos de optimização sofisticados, robustos e eficientes. Por exemplo, os algoritmos meméticos e os modelos substitutos podem melhorar o desempenho geral da optimização, reduzindo o número de avaliações da função objetivo, como demonstrado nos Capítulos 4 e 5, respectivamente. A computação paralela pode ser aplicada em conjunto com essas abordagens para resolver

problemas de grande escala. Além disso, é importante estudar abordagens para determinar os melhores valores para os parâmetros dos métodos de otimização e dos modelos substitutos.

4) Finalmente, a quarta linha de desenvolvimento consiste em estudar a otimização conjunta de dois ou mais subsistemas do foguete, considerando esses subsistemas em uma mesma configuração ou em configurações distintas do foguete. Por exemplo, na mesma configuração: dois ou mais conjuntos de empenas do foguete, sua coifa, e assim por diante. Em configurações distintas: três painéis no conjunto de empenas do foguete, ou a opção com quatro painéis. Recomenda-se, também, investigar a sensibilidade dos objetivos às variáveis em jogo a fim de reduzir sua quantidade na otimização, selecionando apenas aquelas que causam variações significativas em comparação com as outras.

REFERÊNCIAS BIBLIOGRÁFICAS

AKPINAR, S.; BAYHAN, G. M.; BAYKASOGLU, A. Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. **Applied Soft Computing**, v. 13, n. 1, Jan., p. 574-589, 2013.

BAK, P.; TANG, C.; WIESENFELD, K. Self-organized criticality. **Physical Review A**, v. 38, n. 1, p. 364-374, July 1988.

BARBOSA, A. N.; GUIMARÃES, L. N. F. Investigating the Efficiency of the Surrogates Based on Neural Networks in Assisting Multi-objective Optimization of Test-problems Performed by a Non-generational Genetic Algorithm. In: INTERNATIONAL CONFERENCE ON ENGINEERING OPTIMIZATION, 3., 2012, Rio de Janeiro, Brazil. **Proceedings...** Rio de Janeiro: COPPE, 2012. ISBN 978-85-7650-344-6.

BARBOSA, A. N.; GUIMARÃES, L. N. F. Multidisciplinary Design Optimization of Sounding Rocket Fins Shape Using a Tool Called MDO-SONDA. **Journal of Aerospace Technology and Management**, v. 4, n. 4, p. 431-442, December 2012.

BAZARAA, M. S.; JARVIS, J. J. **Linear programming and network flows**. New York: John Wiley & Sons, 1977. 565 p.

BITTENCOURT, G. **Inteligência Artificial: Ferramentas e Teorias**. 2. ed. Florianópolis: UFSC, 2001.

BLAKE, W. B. **Missile datcom user's manual** – 1997 Fortran 90 revision. Ohio: Wright-Patterson Air Force Base, 1998. 103 p. (AFRL-VA-WP-TR-1998-3009).

BORGES, C. C. H. **Algoritmos genéticos para otimização em dinâmica de estruturas**. 216 p. Tese (Doutorado em Ciências) – Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, 1999.

BORGES, C. C. H.; BARBOSA, H. J. C. A non-generational genetic algorithm for multiobjective optimization. In: CONGRESS ON EVOLUTIONARY COMPUTATION, 2000, La Jolla, USA. **Proceedings...** La Jolla: IEEE Press, 2000. p. 172-179.

CHAPMAN, B.; JOST, G.; PAS, R. V. D. **Using OpenMP: portable shared memory parallel programming**. Cambridge: Massachusetts Institute of Technology, 2008. 353 p.

CORNELISSE, J. W.; SCHÖYER, H. F. R.; WAKKER, K. F. **Rocket propulsion and spaceflight dynamics**. London: Pitman, 1979. 505 p.

DAWKINS, R. **The selfish gene:** 30th anniversary edition. Oxford: Oxford University Press, 2006. 369 p.

DEB, K.; AGRAWAL, R. B. Simulated binary crossover for continuous search space. **Complex Systems**, v. 9, p. 115-148, 1995.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. **A fast and elitist multi-objective genetic algorithm:** NSGA –II. Kanpur: Indian Institute of Technology, 2000. KanGAL Report No. 200001.

DHAR, D. Theoretical studies of self-organized criticality. **Physica A: Statistical and Theoretical Physics**, v. 369, n. 1, p. 29-70, September 2006.

DRÉO, J.; SIARRY, P.; PÉTROWSKI, A.; TAILLARD, E. **Metaheuristics for hard optimization: simulated annealing, tabu search, evolutionary and genetic algorithms, ant colonies, methods and case studies**. Berlin: Springer, 2006. 369 p.

FAHLMAN, S. E.; LEBIERE, C. The cascade-correlation learning architecture. In: TOURETZKY, D. S. (ed.). **Advances in neural information processing Systems**. San Mateo: Morgan Kaufmann, 1990. v. 2, p. 524-532.

FERGUSON, J. **A brief survey of the history of the calculus of variations and its applications**. University of Victoria, 2004. Disponível em: <http://arxiv.org/pdf/math/0402357> Acesso em: 9 abr. 2009.

FLEEMAN, E. L. **Tactical missile design**. 2. ed. Reston: AIAA, 2006. 468 p.

FLOUDAS, C. A.; PARDALOS, P. M. **Encyclopedia of optimization**. 2. ed. New York: Springer, 2009. 4626 p.

FODOR, J.; KLEMPOUS, R.; ARAUJO, C. P. S. **Recent advances in intelligent engineering systems**. Berlin: Springer, 2012. 439 p.

FORRESTER, A. I. J.; SÓBESTER, A.; KEANE, A. J. **Engineering design via surrogate modelling:** a practical guide. Chichester: John Wiley, 2008. 210 p.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. **The elements of statistical learning:** data mining, inference and prediction. 2. ed. New York: Springer, 2009. 464 p.

GALSKI, R. L. **Desenvolvimento de versões aprimoradas, híbridas, paralela e multiobjetivo do método da otimização extrema generalizada e sua aplicação no projeto de sistemas espaciais**. 2007. 282 p (INPE-14795-TDI/1238). Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2006. Disponível em: <http://urlib.net/sid.inpe.br/mtc-m17@80/2006/11.30.19.04>. Acesso em: 13 mar. 2013.

GARCIA, A.; YAMANAKA, S. S. C.; BARBOSA, A. N.; BIZARRIA, F. C. P.; JUNG, W.; SCHEUERPFLUG, F. VSB-30 sounding rocket: history of flight performance.

Journal of Aerospace Technology and Management, v. 3, n. 3, p. 325-330, December 2011.

GERMANY'S NATIONAL RESEARCH CENTER FOR AERONAUTICS AND SPACE (DLR). **Experiments on SHEFEX II successful**. DLR Portal web site, 2012. Disponível em: <http://www.dlr.de/dlr/en/desktopdefault.aspx/tabid-10081/151_read-4100/>. Acesso em: 6 jul. 2012.

GOH, C. K.; TAN, K. C. **Evolutionary multi-objective optimization in uncertain environments:** issues and algorithms. Berlin: Springer, 2009. 271 p.

GOMES, R., M. **Trajectory calculation using the ROSI program** – nominal trajectory analysis. São José dos Campos: Institute of Aeronautics and Space, 2004. (ASE-RT-010-2004).

GOOGLE INC. **Google Earth**. 6.1.0.5001. Google, 2011. (software). Disponível em: <http://www.google.com/earth/index.html>. Acesso em: 21 jul. 2011.

GRINSTEIN, G.; TRUTSCHL, M.; CVEK, U. High-Dimensional Visualizations. In: VISUAL DATA MINING WORKSHOP, 2001, San Francisco, USA. **Proceedings...** San Francisco: ACM, 2001.

HAYKIN, S. **Neural networks:** a comprehensive foundation. 2. ed. New York: Prentice Hall, 1999. 823 p.

HUBAND, S.; HINGSTON, P.; BARONE, L. C.; WHILE, R. L. A review of multiobjective test problems and a scalable test problem toolkit. **IEEE Transactions on Evolutionary Computation**, v. 10, n. 5, p. 477-506, October 2006.

INSTITUTO DE AERONÁUTICA E ESPAÇO (IAE). **O Foguete de sondagem VS-40**. São José dos Campos. 2012. Disponível em: <www.iae.cta.br/?action=vs40>. Acesso em: 4 out. 2012.

KOHONEN, T. **Self- organizing maps**. 3. ed. Berlin: Springer, 2001. 501 p.

KOHONEN, T. The self-organizing map. **IEEE**, v. 78, n. 9, p. 1464-1480, September 1990.

LA NEVE, A.; CORRÊA Jr, F., A. SARA Experiment Module: An Interdisciplinary Project through Partnership with the Institute of Aeronautics and Space. In: INTERNATIONAL CONFERENCE ON ENGINEERING EDUCATION, 2001, Oslo, Norway. **Proceedings...** Oslo, 2001. p. 26-30.

LINDEN, R. **Algoritmos genéticos:** uma importante ferramenta da inteligência computacional. Rio de Janeiro: Brasport, 2006. 348 p.

MASON, W. J; COVERSTONE-CARROLL, V.; HARTMANN, J. W. Optimal Earth orbiting satellite constellations via a Pareto genetic algorithm. In: ASTRODYNAMICS

SPECIALIST CONFERENCE AND EXHIBIT, 1998, Boston, USA. **Proceedings...** Boston: AIAA/AAS, 1998. p. 169-177. (Paper No. AIAA 98-4381).

MEDEIROS, F. L. L. **Algoritmo genético híbrido como um método de busca de estados estacionários de sistemas dinâmicos**. 2006. 191 p. (INPE-13913-TDI/1058). Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2002. Disponível em: <http://urlib.net/sid.inpe.br/iris@1905/2005/08.03.21.24.54>. Acesso em: 13 mar. 2013.

MEHRA, P.; WAH, B. W. **Artificial neural networks:** concepts and theory. Los Alamitos: IEEE Computer Society Press, 1992. 667 p.

MICHALEWICZ, Z.; SCHOENAUER, M. Evolutionary algorithms for constrained parameter optimization problems. **Evolutionary Computation**, v. 4, n. 1, p. 1-32, 1996.

MONTENBRUCK, O.; MARKGRAF, M.; TURNER, P.; ENGLER, W.; SCHMITT, G. GPS tracking of sounding rockets – a European perspective. In: WORKSHOP ON SATELLITE NAVIGATION USER EQUIPMENT TECHNOLOGIES, 1., 2001, Noordwijk, Holland. **Proceedings...** Noordwijk: ESA/ESTEC/NAVITEC, 2001.

MORAES Jr, P.; PILCHOWSKI, H. U. Plataforma orbital para experimentação em ambiente de micro-gravidade. In: CONGRESSO BRASILEIRO DE ENGENHARIA MECÂNICA, 14., 1997, Bauru, Brazil. **Proceedings...** Rio de Janeiro: ABCM, 1997. v. 1, p. 9.

MOSCATO, P.; COTTA, C. A gentle introduction to memetic algorithms. In: GLOVER, F.; KOCHENBERGER, G. A. (eds.). **Handbook of Metaheuristics**. New York: Springer, 2003. p. 105-144.

MOUSSI, R.; NDIAYE, N. F.; YASSINE, A. Hybrid genetic simulated annealing algorithm (HGSAA) to solve storage container problem in port. In: Pan, J.-S.; Chen, S.-M.; Nguyen, N. (eds.). **Intelligent Information and Database Systems**. Heidelberg: Springer Berlin, 2012. v. 7197, p. 301-310.

MUNAKATA, T. **Fundamentals of the new artificial intelligence:** neural, evolutionary, fuzzy and more. 2. ed. London: Springer, 2008. 255 p.

NAKKA, R.A. **Richard Nakka's experimental rocketry web site**. 2001. Disponível em: <www.nakka-rocketry.net/fins.html>. Acesso em: 30 ago. 2012.

NERI, F.; COTTA, C.; MOSCATO, P. **Handbook of memetic algorithms**. Berlin: Springer, 2012. 368 p.

OBAYASHI, S.; SASAKI, D. Visualization and data mining of Pareto solutions using self-organizing map. In: FONSECA, C. M.; FLEMING, P. J.; ZITZLER, E.; DEB, K.; THIELE, L. (eds.). **EMO 2003**. Heidelberg : Springer, 2003. LNCS, v. 2632, p. 796-809.

OLIVER, D. W.; KELLIHER, T. P.; KEEGAN, Jr., J. G. **Engineering complex systems with models and objects**. New York: McGraw-Hill, 1997. 325 p.

ONG, Y. S.; NAIR, P. B.; KEANE, A. J.; WONG, K. W. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In: YAOCHU, J. (ed.). **Knowledge Incorporation in Evolutionary Computing**. Heidelberg: Springer, 2005.

PEREIRA, A. L.; MORAES Jr, P. Ensaios aerodinâmicos em voo do foguete suborbital VS-40. In: CONGRESSO NACIONAL DE ENGENHARIA MECÂNICA, 15., 2000, Natal, Brazil. **Proceedings...** Natal, 2000. Disponível em: <http://pintassilgo2.ipen.br/biblioteca/ cd/conem/2000/ IC9790.pdf>. Acesso em: 6 jul. 2012.

PETER, J.; MARCELET, M. Comparison of surrogate models for turbomachinery design. **WSEAS Transactions on Fluid Mechanics**, v. 3, n. 1, p. 10-17, 2008.

PETROVA, S. S.; SOLOV'EV, A. D. The origin of the method of steepest descent. **Historia Mathematica**, v. 24, n. 4, p. 361-375, November 1997.

PRESSMAN, R. S. **Software engineering:** a practitioner's approach. 3. ed. McGraw-Hill, 1992. 1056 p.

RAO, S. S. **Engineering optimization:** theory and practice. 3. ed. New York: John Wiley & Sons, 1996. 920 p.

RASHEED, K. M. **GADO: A genetic algorithm for continuous design optimization**. 1998. 107 p. (Technical Report DCS-TR-352). Ph.D. Thesis – The State University of New Jersey, New Brunswick, 1998.

REDDY, K. P. J. Hypersonic flight and ground testing activities in India. In: AUSTRALASIAN FLUID MECHANICS CONFERENCE, 16., 2007, Gold Coast, Australia. **Proceedings...** 2007.

RIBEIRO, T. S. Veículos Lançadores de Satélites – cenário atual e futuro. **Parcerias Estratégicas**, n.7, pp. 235-247, 1999.

RITTER, H.; MARTINETZ, T.; SCHULTEN, K. **Neural computation and self-organizing maps**. Reading: Addison-Wesley, 1992. 350 p.

ROSHANIAN, J.; KESHAVARZ, Z. Multidisciplinary design optimization applied to a sounding rocket. **Journal of the Indian Institute of Science**, v. 86, p. 363-375, July-August 2006.

ROWELL, L. F.; KORTE, J. J. **Launch vehicle design and optimization methods and priority for the advanced engineering environment**. Hampton: National Aeronautics and Space Administration, 2003. 29 p. NASA/TM-2003-212654.

RUSSELL, S.; NORVIG, P. **Artificial intelligence**. 3. ed. Prentice Hall, 2010. 1021 p.

SÓBESTER, A.; FORRESTER, A. I. J. On the use of surrogate models in global optimization: a practical approach. In: GLOBAL OPTIMIZATION WORKSHOP, 2007, London. **Proceedings...** London: Imperial College, 2007. p. 1-14.

SOOY, T. J.; SCHMIDT, R. Z. Aerodynamic predictions, comparisons, and validations using missile DATCOM (97) and aeroprediction 98 (AP98). **Journal of Spacecraft and Rockets**, v. 42, n. 2, p. 257-265, April 2005.

SOUKUP, T.; DAVIDSON, I. **Visual data mining:** techniques and tools for data visualization and mining. New York: John Wiley & Sons, 2002. 382 p.

SOUSA, F. L. **Otimização extrema generalizada:** um novo algoritmo estocástico para o projeto ótimo. 2003. 142 p. (INPE-9564-TDI/836). Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2002. Disponível em: <http://urlib.net/sid.inpe.br/marciana/2003/03.18.15.39>. Acesso em: 13 mar. 2013.

SRINIVAS, N.; DEB, K. Multiobjective optimization using nondominated sorting in genetic algorithms. **Evolutionary Computation**, v. 2, n. 3, p. 221-248, 1994.

TANG, B. Orthogonal array-based latin hypercubes. **Journal of the American Statistical Association**, v. 88, n. 424, p. 1392-1397, 1993.

TARHAN, E. **Aerodynamic design optimization of solid propellant missiles**. Turkey: ROKETSAN Missile Industries Inc., 2000.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern recognition**. 2. ed. San Diego: Elsevier, 2003. 689 p.

TURNER, J.; HÖRSCHGEN, M.; ETTL, J.; JUNG, W.; TURNER, P. Shefex 2 development status of the vehicle and sub-systems for a hypersonic re-entry flight experiment. In: SYMPOSIUM ON EUROPEAN ROCKET AND BALLOON PROGRAMMES AND RELATED RESEARCH, 19., 2009, Bad Reichenhall, Germany. **Proceedings...** Bad Reichenhall: ESA, 2009.

VALENZUELA-RENDÓN, M.; URESTI-CHARRE, E. A non-generational genetic algorithm for multiobjective optimization. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 7., 1997, San Mateo, USA. **Proceedings...** 1997. p. 658-665.

WEIHS, H.; LONGO, J. M. A.; TURNER, J. The sharp edge flight experiment SHEFEX II, a mission overview and status. In: INTERNATIONAL SPACE PLANES AND HYPERSONIC SYSTEMS AND TECHNOLOGIES CONFERENCE, 15., 2008, Ohio, USA. **Proceedings...** Ohio: AIAA, 2008.

WEIHS, H.; TURNER, J.; LONGO, J.; GÜLHAN, A. Key experiments within the shefex II mission. In: INTERNATIONAL ASTRONAUTICAL CONGRESS, 59., 2008, Glasgow, Scotland. **Proceedings...** Glasgow: IAF, 2008.

WHITLEY, D. The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In: INTERNATIONAL CONFERENCE. ON GENETIC ALGORITHMS AND THEIR APPLICATIONS, 3., 1989, San Mateo, USA. **Proceedings...** San Mateo: Morgan Kaufmann, 1989. p. 116-121.

YUEN, K.-V. **Bayesian methods for structural dynamics and civil engineering**. Singapore: John Wiley & Sons, 2010. 294 p.

ZHANG, Y.; WU, L. A robust hybrid restarted simulated annealing particle swarm optimization technique. **Advances in Computer Science and its Applications**. v. 1, n. 1, p. 5-8, March 2012.

ZIEGLTRUM, W. **Analysis and specification of trajectory program ROSI, input program WEIGHT, plot program ROSIPLOT**. Oberpfaffenhofen: German Aerospace Center, 1984. v. 1. (DFVLR TN 12/76).

APÊNDICE A – CÓDIGOS EM MATLAB DO CAPÍTULO 4

A.1 Rotina principal

```
close all;  % Fechamento das janelas de figuras
clc;        % Limpeza da janela de comando do Matlab
clear;      % Limpeza do espaço de trabalho

% EXECUÇÃO DA ABORDAGEM MEMÉTICA COM O OPERADOR ORAD

moga_com_mgeo_acoplado_real;        % Subseção A.2
mgeo_com_moga_acoplado_real;        % Subseção A.3
```

A.2 Rotina do algoritmo genético multiobjetivo não-geracional

```
% NOME DA ROTINA: moga_com_mgeo_acoplado_real

% VARIÁVEIS GLOBAIS

global nv;              % Número de variáveis do problema de otimização
global xInf;            % Limite inferior do valor da variável
global xSup;            % Limite superior do valor da variável
global nobj;            % Número de funções objetivo
global pSementes;       % Sementes do M-GEO
global xSementes;
global fSementes;
global contAvaliacoes;  % Contagem de avaliações da função objetivo
global contDuplicados;  % Contagem de soluções duplicadas
global contInvalidos;   % Contagem de soluções inválidas
global pInicial;        % População inicial
global melhor;          % Se o problema é de maximização ou minimização

% CONFIGURAÇÃO

FuncObj( 0, false );    % Subseção A.4.3
config_geral_moga;      % Subseção A.4.1

% INICIALIZAÇÃO

      % Escolha dos indivíduos da população

p = rand( np, nv );           % Domínio (normalizado entre 0 e 1)
x = zeros( np, nv );          % Domínio
f = zeros( np, nobj );        % Contradomínio

      % Remoção de indivíduos inválidos e duplicados

contInvalidos =  0;
contDuplicados = 0;
contAvaliacoes = 0;
for ii = 1:np
    repeteIndividuo = true;
```

```
    while( repeteIndividuo == true )
        validade = false;
        while( validade == false )
            [ x(ii,:), f(ii,:), validade ] = ...
                FuncObj( p(ii,:), true );      % Subseção A.4.3
            contAvaliacoes = contAvaliacoes + 1;
            if( validade == false )
                contInvalidos = contInvalidos + 1;
                p(ii,:) = rand( 1, nv );
            end
        end
        repeteIndividuo = false;
        for jj = 1:ii-1
            ijIguais = true;
            for kk = 1:nv
                if( p(ii,kk) ~= p(jj,kk) )
                    ijIguais = false;
                    break;
                end
            end
            if( ijIguais == true )
                contDuplicados = contDuplicados + 1;
                p(ii,:) = rand( 1, nv );
                repeteIndividuo = true;
                break;
            end
        end
    end
end

% AVALIAÇÃO DA POPULAÇÃO

nd =           zeros(np);
dv =           zeros(np);
aptidao =      zeros(1,np);
aptidaoTeste = zeros(1,np);
ordem =        zeros(1,np);
dominancia =   zeros(1,np);
vizinhanca =   zeros(1,np);
for ii = 1:np-1
    for jj = ii+1:np
        iDomina =   0;
        jDomina =   0;
        distancia = 0;
        for kk = 1:nobj
            if( melhor(kk) == 0 )
                if( f(ii,kk) < f(jj,kk) )
                    iDomina = iDomina + 1;
                elseif( f(jj,kk) < f(ii,kk) )
                    jDomina = jDomina + 1;
                end
            else
                if( f(ii,kk) > f(jj,kk) )
                    iDomina = iDomina + 1;
                elseif( f(jj,kk) > f(ii,kk) )
                    jDomina = jDomina + 1;
```

```
                end
            end
            distancia = distancia + ( f(ii,kk) - f(jj,kk) )^2;
        end
        nd(ii,jj) = 0;
        nd(jj,ii) = 0;
        if( iDomina > 0 & jDomina == 0 )
            nd(ii,jj) = 1;
        elseif( iDomina == 0 & jDomina > 0 )
            nd(jj,ii) = 1;
        end
        dv(ii,jj) = 0;
        dv(jj,ii) = 0;
        distancia = sqrt( distancia );
        if( distancia < raioVizinhanca )
            dv(ii,jj) = ...
                1 - ( distancia/ raioVizinhanca )^fatorGraduacao;
            dv(jj,ii) = dv(ii,jj);
        end
    end
end
for ii = 1:np
    for jj = 1:np
        dominancia(ii) = ...
            dominancia(ii) + nd(jj,ii);
        vizinhanca(ii) = vizinhanca(ii) + dv(ii,jj);
    end
    aptidao(ii) = ...
        ( 1 + dominancia(ii) )*( 1 + vizinhanca(ii) );
end
 [ aptidaoTeste, ordem ] = sort( aptidao );
x0 = x;
f0 = f;

% GERAÇÕES DA POPULAÇÃO

filho =           zeros(2,nv);
xFilho =          zeros(2,nv);
fFilho =          zeros(2,nobj);
validadeFilho =   zeros(1,2);
ndFilho =         zeros(2,np);
ndFilhoJ =        zeros(np,2);
dvFilho =         zeros(2,np);
aptidaoFilho =    zeros(1,2);
dominanciaFilho = zeros(1,2);
vizinhancaFilho = zeros(1,2);
contGeracoes =    0;
termino =         false;
while( termino == false )
    filhosValidos = false;
    contPerda = 0;
    while( filhosValidos == false )

      % Seleção dos genitores

        selecao =   ceil( np*( 1.5 - sqrt( 2.25 - 2*rand(1) )) );
```

```
  indicePai = ordem(selecao);
  while( ordem(selecao) == indicePai )
      selecao = ceil( np*( 1.5 - sqrt( 2.25 - 2*rand(1) )) );
  end
  indiceMae = ordem(selecao);
  pai(1,:) =  p(indicePai,:);
  pai(2,:) =  p(indiceMae,:);

% Aplicação do operador real ORAD

  for kk = 1:2
      filho(kk,:) = pai(kk,:);
      qq = randint(1,1,[1,nv]);
      ff = randint(1,1,[kinf,ksup]);
      fator = a*exp(-ff/2);
      if( randint(1) == 0 )
          mutacao = -filho(kk,qq)*fator;
      else
          mutacao = (1-filho(kk,qq))*fator;
      end
      filho(kk,qq) = filho(kk,qq) + mutacao;
  end

% Comparação dos descendentes com os indivíduos da população

  filhosValidos = true;
  for ii = 1:2
      for jj = 1:np
          filhoIgual(ii) = true;
          for kk = 1:nv
              if( filho(ii,kk) ~= p(jj,kk) )
                  filhoIgual(ii) = false;
                  break;
              end
          end
          if( filhoIgual(ii) == true )
              break;
          end
      end
  end
  if( filhoIgual(1) == true & filhoIgual(2) == true )
      filhosValidos = false;
      contPerda = contPerda + 1;
  elseif( filhoIgual(1) == false & filhoIgual(2) == true )
      filho(2,:) = filho(1,:);
  elseif( filhoIgual(1) == true & filhoIgual(2) == false )
      filho(1,:) = filho(2,:);
  end

% Verificação da validade dos descendentes

  if( filhosValidos == true )
      [ xFilho(1,:), fFilho(1,:), validadeFilho(1) ] = ...
          FuncObj( filho(1,:), true );    % Subseção A.4.3
      contAvaliacoes = contAvaliacoes + 1;
      [ xFilho(2,:), fFilho(2,:), validadeFilho(2) ] = ...
```

```
            FuncObj( filho(2,:), true );    % Subseção A.4.3
        contAvaliacoes = contAvaliacoes + 1;
        if( validadeFilho(1) == false & ...
            validadeFilho(2) == false )
            filhosValidos = false;
            contPerda = contPerda + 1;
        elseif( validadeFilho(1) == true & ...
                validadeFilho(2) == false )
            filho(2,:) =  filho(1,:);
            xFilho(2,:) = xFilho(1,:);
            fFilho(2,:) = fFilho(1,:);
        elseif( validadeFilho(1) == false & ...
                validadeFilho(2) == true )
            filho(1,:) =  filho(2,:);
            xFilho(1,:) = xFilho(2,:);
            fFilho(1,:) = fFilho(2,:);
        end
    end

% Seleção do descendente mais apto

    if( filhosValidos == true )
        for ii = 1:2
            for jj = 1:np
                iDomina =   0;
                jDomina =   0;
                distancia = 0;
                for kk = 1:nobj
                    if( melhor(kk) == 0 )
                        if( fFilho(ii,kk) < f(jj,kk) )
                            iDomina = iDomina + 1;
                        elseif( f(jj,kk) < fFilho(ii,kk) )
                            jDomina = jDomina + 1;
                        end
                    else
                        if( fFilho(ii,kk) > f(jj,kk) )
                            iDomina = iDomina + 1;
                        elseif( f(jj,kk) > fFilho(ii,kk) )
                            jDomina = jDomina + 1;
                        end
                    end
                    distancia = distancia + ...
                                ( fFilho(ii,kk) - f(jj,kk) )^2;
                end
                ndFilho(ii,jj) =  0;
                ndFilhoJ(jj,ii) = 0;
                if( iDomina > 0 & jDomina == 0 )
                    ndFilho(ii,jj) = 1;
                elseif( iDomina == 0 & jDomina > 0 )
                    ndFilhoJ(jj,ii) = 1;
                end
                dvFilho(ii,jj) = 0;
                distancia = sqrt( distancia );
                if( distancia < raioVizinhanca )
                    dvFilho(ii,jj) = ...
                        1 - ( distancia/ ...
```

```
                        raioVizinhanca )^fatorGraduacao;
                    end
                end
            end
            for ii = 1:2
                dominanciaFilho(ii) = 0;
                vizinhancaFilho(ii) = 0;
                for jj = 1:np
                    dominanciaFilho(ii) = dominanciaFilho(ii) + ...
                                          ndFilhoJ(jj,ii);
                    vizinhancaFilho(ii) = vizinhancaFilho(ii) + ...
                                          dvFilho(ii,jj);
                end
                aptidaoFilho(ii) = ( 1 + dominanciaFilho(ii) )* ...
                                   ( 1 + vizinhancaFilho(ii) );
            end
            if( aptidaoFilho(1) <= aptidaoFilho(2) )
                filhoVencedor = 1;
            else
                filhoVencedor = 2;
            end

  % Avaliação da população considerando o descendente mais apto

            for ii = 1:np
                aptidaoTeste1 = 1 + dominancia(ii) + ...
                                ndFilho(filhoVencedor,ii);
                aptidaoTeste2 = 1 + vizinhanca(ii) + ...
                                dvFilho(filhoVencedor,ii);
                aptidaoTeste(ii) = aptidaoTeste1 * aptidaoTeste2;
            end
            [ aptidaoTeste, ordem ] = sort( aptidaoTeste );
            indicePior = ordem(np);
            if( aptidaoFilho(filhoVencedor) > aptidaoTeste(np) )
                filhosValidos = false;
                contPerda = contPerda + 1;
            end
        end
    end

  % Substituição do pior indivíduo pelo descendente mais apto

    p(indicePior,:) = filho(filhoVencedor,:);
    x(indicePior,:) = xFilho(filhoVencedor,:);
    f(indicePior,:) = fFilho(filhoVencedor,:);
    dominancia(indicePior) = dominanciaFilho(filhoVencedor) - ...
                             ndFilhoJ(indicePior,filhoVencedor);
    vizinhanca(indicePior) = vizinhancaFilho(filhoVencedor) - ...
                             dvFilho(filhoVencedor,indicePior);
    aptidao(indicePior) = ( 1 + dominancia(indicePior) )* ...
                          ( 1 + vizinhanca(indicePior) );

  % Reavaliação da população

    for ii = 1:np
        if( ii ~= indicePior )
```

```
            dominancia(ii) = dominancia(ii) - nd(indicePior,ii) + ...
                             ndFilho(filhoVencedor,ii);
            vizinhanca(ii) = vizinhanca(ii) - dv(ii,indicePior) + ...
                             dvFilho(filhoVencedor,ii);
            aptidao(ii) = ( 1 + dominancia(ii) )* ...
                          ( 1 + vizinhanca(ii) );
            nd(indicePior,ii) = ndFilho(filhoVencedor,ii);
            nd(ii,indicePior) = ndFilhoJ(ii,filhoVencedor);
            dv(ii,indicePior) = dvFilho(filhoVencedor,ii);
            dv(indicePior,ii) = dv(ii,indicePior);
        end
    end
    [ aptidaoTeste, ordem ] = sort( aptidao );

      % Condição de término: número de gerações

    contGeracoes = contGeracoes + 1;
    termino = contGeracoes >= ng;
end

% GERAÇÃO DE GRÁFICOS

figure;
plot(x0(:,1)',x0(:,2)','x',x(:,1)',x(:,2)','o');
xlabel('x1','FontSize',12,'FontName','Times','FontAngle','italic');
ylabel('x2','FontSize',12,'FontName','Times','FontAngle','italic');
h = legend('População inicial','População final',1);
set(h,'FontSize',10,'FontName','Times');
set(gca,'FontSize',10,'FontName','Times');
figure;
plot(f0(:,1)',f0(:,2)','x',f(:,1)',f(:,2)','o');
xlabel('f1','FontSize',12,'FontName','Times','FontAngle','italic');
ylabel('f2','FontSize',12,'FontName','Times','FontAngle','italic');
h = legend('População inicial','População final',1);
set(h,'FontSize',10,'FontName','Times');
set(gca,'FontSize',10,'FontName','Times');
fprintf('MOGA =>    AVALIACOES = %d\n', contAvaliacoes);
fprintf('MOGA =>    DUPLICADOS = %d\n', contDuplicados);
fprintf('MOGA =>    INVALIDOS =  %d\n', contInvalidos);

% FINALIZAÇÃO

clear aptidao aptidaoFilho aptidaoTeste aptidaoTeste1 aptidaoTeste2;
clear contGeracoes contPerda corte1 corte2 distancia dominancia;
clear dominanciaFilho dv dvFilho f0 fFilho fatorGraduacao filho;
clear filhoIgual filhoVencedor filhoValidos;
clear hamming iDomina ii ijIguais;
clear indiceMae indicePai indicePior jDomina jj kk mae melhor nobj np;
clear nv operador operadorAleatorio ordem pai;
clear raioVizinhanca repeteIndividuo selecao termino;
clear validade validadeFilho vizinhanca vizinhancaFilho x0 xFilho;
clear xInf xSup filhosValidos nd ndFilho ndFilhoJ ng;
pSementes = p;
xSementes = x;
fSementes = f;
avalMoga =  contAvaliacoes;
```

```
x2 = x;
f2 = f;
clear p x f;
```

A.3 Rotina da versão multiobjetivo da otimização extrema generalizada

```
% NOME DA ROTINA: mgeo_com_moga_acoplado_real

% VARIÁVEIS GLOBAIS

global nv;              % Número de variáveis do problema de otimização
global xInf;            % Limite inferior do valor da variável
global xSup;            % Limite superior do valor da variável
global nobj;            % Número de funções objetivo
global pSementes;       % Sementes do M-GEO
global xSementes;
global fSementes;
global contAvaliacoes;  % Contagem de avaliações da função objetivo
global contDuplicados;  % Contagem de soluções duplicadas
global contInvalidos;   % Contagem de soluções inválidas
global melhor;          % Se o problema é de maximização ou minimização
global nf;              % Número de soluções visitadas por torneio
clear xValido fValido xPareto fPareto;

% CONFIGURAÇÃO

FuncObj( 0, false );          % Subseção A.4.3
config_geral_mgeo;            % Subseção A.4.2
if( mogaAcoplado == true )
    nSementes = length(pSementes(:,1));
end

% INICIALIZAÇÃO

np = nSementes*nTorneios*nf;
nfv = nf/nv;
bF = zeros( nf, nv );
xF = zeros( nf, nv );
fF = zeros( nf, nobj );
validadeF = zeros( 1, nf );
adaptF = zeros( 1, nf );
indiceF = zeros( 1, nf );
ordemF = zeros( 1, nf );
x = zeros( np, nv );
f = zeros( np, nobj );
validade = zeros( 1, np );
pai = zeros( 1, nv );
xPai = zeros( 1, nv );
fPai = zeros( 1, nobj );
xPais = zeros( nSementes*nTorneios, nv );
fPais = zeros( nSementes*nTorneios, nobj );
if( mogaAcoplado == false )
    xSementes = zeros( nSementes, nv );
    fSementes = zeros( nSementes, nobj );
```

```
end

% ITERAÇÕES DO M-GEO

indice = 1;
indiceTorneios = 1;
contAvaliacoes = 0;
for ii = 1:nSementes
    if( mogaAcoplado == false )
        validadePai = false;
        while( validadePai == false )
            pai = rand( 1, nv );
            [ xPai, fPai, validadePai ] = ...
                FuncObj( pai, true );     % Subseção A.4.3
            contAvaliacoes = contAvaliacoes + 1;
        end
        xSementes(ii,:) = xPai;
        fSementes(ii,:) = fPai;
    else
        pai = pSementes(ii,:);
        xPai = xSementes(ii,:);
        fPai = fSementes(ii,:);
    end
    for jj = 1:nTorneios
        objEscolhido = randint( 1, 1, [1, nobj] );
        cont = 0;
        for kk = 1:nf
            bF(kk,:) = pai;
            if( flagBuscaLocal == true )

      % Aplicação do operador real ORAD

                for qq = 1:nv
                    ff = randint(1,1,[kinf,ksup]);
                    fator = a*exp(-ff/2);
                    if( randint(1) == 0 )
                        mutacao = -bF(kk,qq)*fator;
                    else
                        mutacao = (1-bF(kk,qq))*fator;
                    end
                    bF(kk,qq) = bF(kk,qq) + mutacao;
                end
            else
                qq = randint(1,1,[1,nv]);
                ff = randint(1,1,[kinf,ksup]);
                fator = a*exp(-ff/2);
                if( randint(1) == 0 )
                    mutacao = -pai(qq)*fator;
                else
                    mutacao = (1-pai(qq))*fator;
                end
                bF(kk,qq) = pai(qq) + mutacao;
            end
```

```
        % Avalição das soluções visitadas

                [ xF(kk,:), fF(kk,:), validadeF(kk) ] = ...
                    FuncObj( bF(kk,:), true );     % Subseção A.4.3
                contAvaliacoes = contAvaliacoes + 1;
                x(indice,:) = xF(kk,:);
                f(indice,:) = fF(kk,:);
                validade(indice) = validadeF(kk);
                indice = indice + 1;
                if( validadeF(kk) == true )
                    if( melhor(objEscolhido) == 0 )
                        adaptF(kk-cont) = fF(kk,objEscolhido);
                    else
                        adaptF(kk-cont) = -fF(kk,objEscolhido);
                    end
                    indiceF(kk-cont) = kk;
                else
                    cont = cont + 1;
                end
            end
            [ adaptF(1:nf-cont), ordemF(1:nf-cont) ] = ...
                sort( adaptF(1:nf-cont) );
            candidato = randint( 1, 1, [1, nf-cont] );
            while( candidato^-criticalidade < rand(1) )
                candidato = randint( 1, 1, [1, nf-cont] );
            end
            pai = bF(indiceF(ordemF(candidato)),:);
            xPais(indiceTorneios,:) = xF(indiceF(ordemF(candidato)),:);
            fPais(indiceTorneios,:) = fF(indiceF(ordemF(candidato)),:);
            indiceTorneios = indiceTorneios + 1;
        end
    end
    clear bF;
    clear xF;
    clear fF;
    clear validadeF;
    clear adaptF;
    clear indiceF;
    clear ordemF;
    clear pai;
    clear xPai;
    clear fPai;

    % REMOÇÃO DAS SOLUÇÕES DUPLICADAS

    if( mogaAcoplado == false )
        contDuplicados = 0;
    end
    for ii = 1:np
        for jj = 1:ii-1
            ijIguais = true;
            for kk = 1:nv
                if( x(ii,kk) ~= x(jj,kk) )
                    ijIguais = false;
                    break;
                end
```

```
        end
        if( ijIguais == true )
            contDuplicados = contDuplicados + 1;
            validade(ii) = false;
            break;
        end
    end
end

% EXTRAÇÃO DAS SOLUÇÕES VÁLIDAS

if( mogaAcoplado == false )
    contInvalidos = 0;
end
npValido = 0;
for ii = 1:np
    if( validade(ii) == true )
        npValido = npValido + 1;
        xValido(npValido,:) = x(ii,:);
        fValido(npValido,:) = f(ii,:);
    else
        contInvalidos = contInvalidos + 1;
    end
end
clear x;
clear f;
clear validade;

% EXTRAÇÃO DAS SOLUÇÕES NÃO-DOMINADAS

nd = zeros(npValido);
dominancia = zeros(1,npValido);
for ii = 1:npValido-1
    for jj = ii+1:npValido
        iDomina = 0;
        jDomina = 0;
        for kk = 1:nobj
            if( melhor(kk) == 0 )
                if( fValido(ii,kk) < fValido(jj,kk) )
                    iDomina = iDomina + 1;
                elseif( fValido(jj,kk) < fValido(ii,kk) )
                    jDomina = jDomina + 1;
                end
            else
                if( fValido(ii,kk) > fValido(jj,kk) )
                    iDomina = iDomina + 1;
                elseif( fValido(jj,kk) > fValido(ii,kk) )
                    jDomina = jDomina + 1;
                end
            end
        end
        nd(ii,jj) = 0;
        nd(jj,ii) = 0;
        if( iDomina > 0 & jDomina == 0 )
            nd(ii,jj) = 1;
        elseif( iDomina == 0 & jDomina > 0 )
```

```
            nd(jj,ii) = 1;
        end
    end
end
npPareto = 0;
for ii = 1:npValido
    for jj = 1:npValido
        dominancia(ii) = dominancia(ii) + nd(jj,ii);
    end
    if( dominancia(ii) == 0 )
        npPareto = npPareto + 1;
        xPareto(npPareto,:) = xValido(ii,:);
        fPareto(npPareto,:) = fValido(ii,:);
    end
end
fp2 = fPareto;
xp2 = xPareto;
contAvaliacoesMgeo2 = contAvaliacoes;

% GERAÇÃO DE GRÁFICOS

figure;
plot(fPareto(:,1)',fPareto(:,2)','.');
xlabel('f1','FontSize',12,'FontName','Times','FontAngle','italic');
ylabel('f2','FontSize',12,'FontName','Times','FontAngle','italic');
set(gca,'FontSize',10,'FontName','Times');
figure;
plot(xPareto(:,1)',xPareto(:,2)','.');
xlabel('x1','FontSize',12,'FontName','Times','FontAngle','italic');
ylabel('x2','FontSize',12,'FontName','Times','FontAngle','italic');
set(gca,'FontSize',10,'FontName','Times');
figure;
plot(fValido(:,1)',fValido(:,2)','x',fPareto(:,1)',fPareto(:,2)','o');
xlabel('f1','FontSize',12,'FontName','Times','FontAngle','italic');
ylabel('f2','FontSize',12,'FontName','Times','FontAngle','italic');
h = legend('Solução válida','Pareto ótimo',1);
set(h,'FontSize',10,'FontName','Times');
set(gca,'FontSize',10,'FontName','Times');
figure;
plot(xValido(:,1)',xValido(:,2)','x',xPareto(:,1)',xPareto(:,2)','o');
xlabel('x1','FontSize',12,'FontName','Times','FontAngle','italic');
ylabel('x2','FontSize',12,'FontName','Times','FontAngle','italic');
h = legend('Solução válida','Pareto ótimo',1);
set(h,'FontSize',10,'FontName','Times');
set(gca,'FontSize',10,'FontName','Times');
figure;
plot(fValido(:,1)',fValido(:,2)','x', ...
     fSementes(:,1)',fSementes(:,2)','ko', ...
     'MarkerFaceColor',[1 0 0]);
xlabel('f1','FontSize',12,'FontName','Times','FontAngle','italic');
ylabel('f2','FontSize',12,'FontName','Times','FontAngle','italic');
h = legend('Solução válida','Semente',1);
set(h,'FontSize',10,'FontName','Times');
set(gca,'FontSize',10,'FontName','Times');
figure;
plot(xValido(:,1)',xValido(:,2)','x', ...
```

```
     xSementes(:,1)',xSementes(:,2)','ko', ...
     'MarkerFaceColor',[1 0 0]);
xlabel('x1','FontSize',12,'FontName','Times','FontAngle','italic');
ylabel('x2','FontSize',12,'FontName','Times','FontAngle','italic');
h = legend('Solução válida','Semente',1);
set(h,'FontSize',10,'FontName','Times');
set(gca,'FontSize',10,'FontName','Times');
figure;
plot(fPais(:,1)',fPais(:,2)','x', ...
     fSementes(:,1)',fSementes(:,2)','ko', ...
     'MarkerFaceColor',[1 0 0]);
xlabel('f1','FontSize',12,'FontName','Times','FontAngle','italic');
ylabel('f2','FontSize',12,'FontName','Times','FontAngle','italic');
h = legend('Pai','Semente',1);
set(h,'FontSize',10,'FontName','Times');
set(gca,'FontSize',10,'FontName','Times');
figure;
plot(xPais(:,1)',xPais(:,2)','x', ...
     xSementes(:,1)',xSementes(:,2)','ko', ...
     'MarkerFaceColor',[1 0 0]);
xlabel('x1','FontSize',12,'FontName','Times','FontAngle','italic');
ylabel('x2','FontSize',12,'FontName','Times','FontAngle','italic');
h = legend('Pai','Semente',1);
set(h,'FontSize',10,'FontName','Times');
set(gca,'FontSize',10,'FontName','Times');
fprintf('M-GEO =>    AVALIACOES = %d\n', contAvaliacoes);
fprintf('M-GEO =>    DUPLICADOS = %d\n', contDuplicados);
fprintf('M-GEO =>    VALIDOS =    %d\n', npValido);
fprintf('M-GEO =>    INVALIDOS =  %d\n', contInvalidos);
fprintf('M-GEO =>    PARETO =     %d\n', npPareto);
if( mogaAcoplado == true )
    avalTotal = avalMoga + contAvaliacoes;
    fprintf('HIBRIDO => AVALIACOES = %d\n', avalTotal);
end

% FINALIZAÇÃO

clear candidato cont criticalidade dominancia fPais iDomina;
clear ii ijIguais indice indiceTorneios jDomina jj kk;
clear mogaAcoplado nSementes;
clear nTorneios nd nf nobj np nv objEscolhido validadePai xInf;
clear xPais xSementes xSup;
```

A.4 Rotinas de teste da abordagem memética

A.4.1 Configuração do algoritmo genético multiobjetivo não-geracional

```
% NOME DA ROTINA: config_geral_moga

% CONFIGURAÇÃO DOS ENSAIOS

ensaio = 1;
switch( ensaio )
```

```
    case 1
        np = 5;         % Quantidade de indivíduos na população
        ng = 90;        % Número de gerações
    case 2
        np = 15;
        ng = 60;
    case 3
        np = 341;
        ng = 600;
    case 4
        np = 30;
        ng = 1200;
end

% FUNÇÃO DE COMPARTILHAMENTO

raioVizinhanca = 20;
fatorGraduacao = .5;

% OPERADOR ORAD

kinf = 1;
ksup = 6;
a =    1.4;
```

A.4.2 Configuração da versão multiobjetivo da otimização extrema generalizada

```
% NOME DA ROTINA: config_geral_mgeo

% DECISÃO SOBRE O ACOPLAMENTO COM O ALGORITMO GENÉTICO (NGMOGA)

resposta = input('MOGA acoplado? Y/N [Y]: ','s');
mogaAcoplado = true;
if( resposta == 'N' | resposta == 'n' )
    mogaAcoplado = false;
end

% CONFIGURAÇÃO

nf = 20;          % Número de soluções visitadas
if( mogaAcoplado == true )
    flagBuscaLocal = true;
    switch( ensaio )
        case 1
            nTorneios = 50;
        otherwise
            nTorneios = 10;
    end
    criticalidade = 3;

      % Configuração do ORAD para busca local

    kinf = 7;
    ksup = 14;
```

```
    a =    1.2;
else
    flagBuscaLocal = false;
    nSementes =      5;
    nTorneios =      53;
    criticalidade =  3;

      % Configuração do ORAD para busca global

    kinf = 4;
    ksup = 11;
    a =    .8;
end
```

A.4.3 Funções de teste

```
function [ x, f, v ] = FuncObj( individuo, flagExec )

% ENTRADA

      % Indivíduo: se binário: vetor binário das variáveis justapostas
      %            se real:    variáveis no intervalo de 0 a 1
      % flagExec:  modo de execução
      %            TRUE:   executa a função objetivo
      %            FALSE:  passa os limites das variáveis da função

% SAÍDA

      % x:         vetor das variáveis
      % f:         vetor das funções objetivo
      % v:         validade do indivíduo em função das restrições do
      %            problema

% VARIÁVEIS GLOBAIS

global nv;        % Número de variáveis do problema de otimização
global xInf;      % Limite inferior do valor da variável
global xSup;      % Limite superior do valor da variável
global nobj;      % Número de funções objetivo
global melhor;    % Máximo/ mínimo
global nf;        % Número de soluções visitadas

% INICIALIZAÇÃO

if( flagExec == true )
    x = zeros(1,nv);
    f = zeros(1,nobj);
    for ii = 1:nv
        x(ii) = (xSup(ii) - xInf(ii))*individuo(ii) + xInf(ii);
    end
end
```

```
% FUNÇÕES DE TESTE

switch( 1 )       % Escolha da função objetivo
    case 1        % Função F
        if( flagExec == false )
            nv = 2;                           % Número de variáveis
            xInf(1) = -20; xSup(1) = 20;
            xInf(2) = -20; xSup(2) = 20;      % Limites das variáveis
            nobj = 2;                         % Número de objetivos
            melhor(1) = 0;                    % Minimização
            melhor(2) = 0;                    % Minimização
        else
            v = ( x(1)^2 + x(2)^2 - 225 <= 0 );
            v = v & ( x(1) - 3*x(2) + 10 <= 0 );
            f(1) = ( x(1) - 2 )^2 + ( x(2) - 1 )^2 + 2;
            f(2) = 9*x(1) - ( x(2) - 1 )^2;
        end
    case 2        % Função G
        if( flagExec == false )
            nv = 2
            xInf(1) = -10; xSup(1) = 10;
            xInf(2) = -10; xSup(2) = 10
            nobj = 4;
            melhor(1) = 0;
            melhor(2) = 0;
            melhor(3) = 0;
            melhor(4) = 0;
        else
            v = true;
            f(1) = ( x(1) - 2 )^2 + ( x(2) - 2 )^2;
            f(2) = ( x(1) + 2 )^2 + ( x(2) - 2 )^2;
            f(3) = ( x(1) + 2 )^2 + ( x(2) + 2 )^2;
            f(4) = ( x(1) - 2 )^2 + ( x(2) + 2 )^2;
        end
    case 3        % Função F (sem restrições)
        if( flagExec == false )
            nv = 2;
            xInf(1) = -20; xSup(1) = 20;
            xInf(2) = -20; xSup(2) = 20;
            nobj = 2;
            melhor(1) = 0;
            melhor(2) = 0;
        else
            v = true;
            f(1) = ( x(1) - 2 )^2 + ( x(2) - 1 )^2 + 2;
            f(2) = 9*x(1) - ( x(2) - 1 )^2;
        end
end
```

APÊNDICE B – CÓDIGOS EM MATLAB DO CAPÍTULO 5

B.1 Rotina principal

```
close all;        % Fechamento das janelas abertas com figure
clc;              % Limpeza da janela de comando
clear;            % Limpeza do espaço de trabalho
warning off all;
global dataExec;
dataExec = datestr(now, 'yyyy.mm.dd.HH.MM');
diary(['Registros\v20120407_comparar_eficiencia- ' dataExec '.log']);
diary on;
fprintf('#####################################################\n');
fprintf('# COMPARAÇÃO DA EFICIÊNCIA DOS MODELOS SUBSTITUTOS #\n');
fprintf('#####################################################\n\n');

% PARTES DO CÓDIGO

      % Ativação de partes do código
      % Rotinas externas
      % Problemas-teste
      % Configurações da MLP
      % Configurações da RBFN
      % Configuração da plotagem
      % Divisão do conjunto de exemplos em três subconjuntos
      % Estimação, validação e seleção da MLP
      % Estimação, validação e seleção da RBFN
      % Divisão do conjunto de exemplos em dois subconjuntos
      % Estimação e validação da MLP selecionada
      % Estimação e validação da RBFN selecionada
      % Ativação da MLP
      % Ativação da RBFN
      % Plotagem das funções objetivo
      % Plotagem da MLP
      % Plotagem da RBFN
      % Otimização sem o auxílio das redes neurais
      % Otimização com o auxílio da MLP
      % Otimização com o auxílio da RBFN
      % Comparação da eficiência das redes neurais

global funcaoObjSelecionada;
global nbv;
global melhor;

% ATIVAÇÃO DE PARTES DO CÓDIGO

      % Perceptron de múltiplas camadas (MLP)

flagConfiguracoesMLP =                 true;
flagEstimacaoValidacaoSelecaoMLP =     true;
flagEstimacaoValidacaoMLPSelecionada = true;
flagAtivacaoMLP =                      true;
flagPlotagemMLP =                      true;
flagOtimizacaoAuxilioMLP =             true;
flagCalculoEficienciaMLP =             true;
```

```
      % Rede de função de base radial (RBFN)

flagConfiguracoesRBF =                  true;
flagEstimacaoValidacaoSelecaoRBF =      true;
flagEstimacaoValidacaoRBFSelecionada = true;
flagAtivacaoRBF =                       true;
flagPlotagemRBF =                       true;
flagOtimizacaoAuxilioRBF =              true;
flagCalculoEficienciaRBF =              true;

      % Função objetivo

flagPlotagemFuncoesObjetivo =           true;

% ROTINAS EXTERNAS

fprintf(['Identificando as rotinas externas... \n']);
global executar_modelo_original;
global FuncObj;
global normalizar;
global desnormalizar;
global montar_mlp;
global treinar_mlp;
global ativar_mlp;
global determinar_rbf_bias;
global executar_rbf_bias_2;
global treinar_rbf;
global ativar_rbf;
global moga_binario;
global DConj;
global encontrar_solucoes_nao_dominadas;
montar_mlp = ...
    'v20110711_montar_mlp';                          % Subseção B.2.1
treinar_mlp = ...
    'v20120523_treinar_mlp';                         % Subseção B.2.2
ativar_mlp = ...
    'v20120523_ativar_mlp';                          % Subseção B.2.3
treinar_rbf = ...
    'v20120130_treinar_rbf';                         % Subseção B.3.1
determinar_rbf_bias = ...
    'v20110711_determinar_rbf_bias';                 % Subseção B.3.2
ativar_rbf = ...
    'v20120130_ativar_rbf';                          % Subseção B.3.3
executar_rbf_bias_2 = ...
    'v20110711_executar_rbf_bias_2';                 % Subseção B.3.4
normalizar = ...
    'v20110711_normalizar';                          % Subseção B.4.1
desnormalizar = ...
    'v20110711_desnormalizar';                       % Subseção B.4.2
moga_binario = ...
    'v20110711_moga_binario';                        % Subseção B.5.1
executar_moga = ...
    'v20120513_executar_moga';                       % Subseção B.5.2
FuncObj = ...
    'v20110711_func_obj';                            % Subseção B.5.3
```

```
calcular_eficiencia = ...
    'v20120519_calcular_eficiencia';                 % Subseção B.6.1
DConj = ...
    'v20110711_avaliar_dominancia_conjuntos';        % Subseção B.6.2
encontrar_solucoes_nao_dominadas = ...
    'v20120513_encontrar_solucoes_nao_dominadas';    % Subseção B.6.3
executar_modelo_original = ...
    'v20110711_executar_modelo_original';            % Subseção B.8
fprintf('Concluído.\n\n');

% PROBLEMAS-TESTE

fprintf(['Definindo os problemas-teste... \n']);
nbv = 10;   % Número de bits na representação binária da variável
flagFuncaoObjetivo = [1 1 1];
i = 0;
if flagFuncaoObjetivo(1) == 1
    i= i + 1;
    funcaoObjetivo{i} =    'Srinivas1994';
    nrTotalExemplos(i) =   80;
    pTeste(i) =            .2;
    pValidacao(i) =        .2;
    flagMogaExtremo(i) =   false;
    npSemRedes(i) =        30;
    ngSemRedes(i) =        400;
    melhorFObj{i} =        [0 0];
    nrEnsaios(i) =         10;
    npComRedes(i) =        70;
    ngComRedesPorEpoc(i) = 50;
    nEpoc(i) =             3;
    probUso0(i) =          .94;
    nrEnsaiosEta(i) =      100;
    passoNg(i) =           5;
    intervaloGrid(i) =     1.;
end
if flagFuncaoObjetivo(2) == 1
    i = i + 1;
    funcaoObjetivo{i} =    'Deb2000';
    nrTotalExemplos(i) =   100;
    pTeste(i) =            .2;
    pValidacao(i) =        .2;
    flagMogaExtremo(i) =   false;
    npSemRedes(i) =        30;
    ngSemRedes(i) =        400;
    melhorFObj{i} =        [0 0];
    nrEnsaios(i) =         10;
    npComRedes(i) =        80;
    ngComRedesPorEpoc(i) = 90;
    nEpoc(i) =             3;
    probUso0(i) =          .94;
    nrEnsaiosEta(i) =      100;
    passoNg(i) =           5;
    intervaloGrid(i) =     .2;
end
if flagFuncaoObjetivo(3) == 1
    i = i + 1;
```

```
    funcaoObjetivo{i} =    'Huband2006';
    nrTotalExemplos(i) =   200;
    pTeste(i) =            .2;
    pValidacao(i) =        .2;
    flagMogaExtremo(i) =   false;
    npSemRedes(i) =        30;
    ngSemRedes(i) =        800;
    melhorFObj{i} =        [0 0];
    nrEnsaios(i) =         10;
    npComRedes(i) =        100;
    ngComRedesPorEpoc(i) = 120;
    nEpoc(i) =             3;
    probUso0(i) =          .94;
    nrEnsaiosEta(i) =      100;
    passoNg(i) =           5;
    intervaloGrid(i) =     .05;
end
nrFObj = length(funcaoObjetivo);
for i = 1:nrFObj
    fprintf(['--> Problema-teste ' num2str(i) ' = ' ...
             funcaoObjetivo{i} '\n']);
    fprintf(['--> Número total de exemplos = ' ...
             num2str(nrTotalExemplos(i)) '\n']);
    fprintf(['--> Percentual de teste = ' ...
             num2str(pTeste(i)*100) '\n']);
    fprintf(['--> Percentual de validação = ' ...
             num2str(pValidacao(i)*100) '\n']);
    fprintf(['--> Número de pontos na curva de Pareto ideal = ' ...
             num2str(npSemRedes(i)) '\n']);
    fprintf(['--> Número de tentativas de otimização com redes = ' ...
             num2str(nrEnsaios(i)) '\n']);
    fprintf(['--> População na otimização com redes = ' ...
             num2str(npComRedes(i)) '\n']);
    fprintf(['--> Número de gerações por época de otimização = ' ...
             num2str(ngComRedesPorEpoc(i)) '\n']);
    fprintf(['--> Número de épocas de otimização = ' ...
             num2str(nEpoc(i)) '\n']);
    fprintf(['--> Probabilidade de substituição inicial = ' ...
             num2str(probUso0(i)*100) '\n']);
    fprintf(['--> Número de tentativas para avaliar eficiência = ' ...
             num2str(nrEnsaiosEta(i)) '\n']);
    fprintf(['--> Incremento de gerações = ' ...
             num2str(passoNg(i)) '\n']);
end
fprintf('Concluído.\n\n');

% CONFIGURAÇÕES DA MLP

if flagConfiguracoesMLP == true
    fprintf(['Definindo as configurações da MLP... \n']);
    a = 2;                    % Parâmetro da função de ativação
    nrNeuroniosMin   = 30;    % Mínimo de neurônios na camada escondida
    nrNeuroniosMax   = 80;    % Máximo de neurônios na camada escondida
    nrNeuroniosPasso = 5;     % Variação de neurônios na camada
                              % escondida
    nrNeuronios = nrNeuroniosMin:nrNeuroniosPasso:nrNeuroniosMax;
```

```
    nrConfMLP = length(nrNeuronios);
    eta =   .1;                % Taxa de aprendizagem
    alf =   .5;                % Momento
    condP = 6000;              % Limite de épocas de treinamento
    tol =   .0001;             % Tolerância do erro
    bins = 50;                 % Parâmetro para plotagem dos resultados
    fprintf(['--> Número de configurações = ' ...
             num2str(nrConfMLP) '\n']);
    fprintf(['--> Número mínimo de neurônios = ' ...
             num2str(nrNeuroniosMin) '\n']);
    fprintf(['--> Número máximo de neurônios =      ' ...
             num2str(nrNeuroniosMax) '\n']);
    fprintf(['--> Passo = ' ...
             num2str(nrNeuroniosPasso) '\n']);
    fprintf(['--> Taxa de aprendizagem = ' ...
             num2str(eta) '\n']);
    fprintf(['--> Momento = ' ...
             num2str(alf) '\n']);
    fprintf(['--> Limite de épocas = ' ...
             num2str(condP) '\n']);
    fprintf(['--> Tolerância do erro = ' ...
             num2str(tol) '\n']);
    fprintf(['--> Parâmetro da função de ativação = ' ...
             num2str(a) '\n']);
    fprintf('Concluído.\n\n');
end

% CONFIGURAÇÕES DA RBFN

if flagConfiguracoesRBF == true
    fprintf(['Definindo as configurações da RBF... \n']);
    cMin   = 200;       % Mínimo desvio padrão da função de base radial
    cMax   = 7000;      % Máximo desvio padrão da função de base radial
    cPasso = 200;       % Variação do desvio padrão
    c = cMin:cPasso:cMax;
    nrC = length(c);
    pCentrosMin =   .2; % Mínima fração de centros
    pCentrosMax =   .8; % Máxima fração de centros
    pCentrosPasso = .2; % Variação da fração de centros
    pCentros = pCentrosMin:pCentrosPasso:pCentrosMax;

      % pCentros: fração de centros em relação ao número de
      %           padrões de treinamento

    nrPCentros = length(pCentros);
    flagPCentros = false;
    nrTentativasCentros = 10;
    lambda = 0;
    rbf = 'v20110711_gaussian';
    fprintf(['--> Número de configurações = ' ...
             num2str(nrC*nrPCentros) '\n']);
    fprintf(['--> Desvio padrão mínimo = ' ...
             num2str(cMin) '\n']);
    fprintf(['--> Desvio padrão Máximo = ' ...
             num2str(cMax) '\n']);
    fprintf(['--> Passo (desvio padrão) = ' ...
```

```
            num2str(cPasso) '\n']);
    fprintf(['--> Percentual mínimo de centros = ' ...
            num2str(pCentrosMin) '\n']);
    fprintf(['--> Percentual máximo de centros = ' ...
            num2str(pCentrosMax) '\n']);
    fprintf(['--> Passo (percentual máximo de centros) = ' ...
            num2str(pCentrosPasso) '\n']);
    fprintf(['--> Lambda = ' ...
            num2str(lambda) '\n']);
    fprintf(['--> Número de tentativas de posicionamento dos ' ...
            'centros = ' num2str(nrTentativasCentros) '\n']);
    fprintf(['--> Função de base radial = ' ...
            rbf '\n']);
    fprintf('Concluído.\n\n');
end
bins = 50;  % Parâmetro para plotagem dos resultados

% DIVISÃO DO CONJUNTO DE EXEMPLOS EM TRÊS SUBCONJUNTOS

fprintf(['Dividindo o conjunto de exemplos ' ...
         'em três subconjuntos... \n']);
for k = 1:nrFObj
    nrTreinamento = ceil(nrTotalExemplos(k)*(1-pTeste(k)));
    nrTeste =       floor(nrTotalExemplos(k)*pTeste(k));
    nrEstimacao =   ceil(nrTreinamento*(1-pValidacao(k)));
    nrValidacao =   floor(nrTreinamento*pValidacao(k));
    fprintf(['--> Função objetivo = ' ...
            funcaoObjetivo{k} '\n']);
    fprintf(['--> Número total de exemplos = ' ...
            num2str(nrTotalExemplos(k)) '\n']);
    fprintf(['--> Percentual de teste = ' ...
            num2str(pTeste(k)*100) '\n']);
    fprintf(['--> Percentual de validação = ' ...
            num2str(pValidacao(k)*100) '\n']);
    fprintf(['--> Número de exemplos para estimação = ' ...
            num2str(nrEstimacao) '\n']);
    fprintf(['--> Número de exemplos para validação = ' ...
            num2str(nrValidacao) '\n']);
    fprintf(['--> Número de exemplos para teste = ' ...
            num2str(nrTeste) '\n']);
    funcaoObjSelecionada = funcaoObjetivo{k};
    xmin{k} = feval(executar_modelo_original,0,'xInf');
    xmax{k} = feval(executar_modelo_original,0,'xSup');
    x0{k} = [xmin{k}; ...
            lhsu(xmin{k},xmax{k}, ...   % Subseção B.7
                 nrTotalExemplos(k)-2); xmax{k}];
    for i = 1:length(x0{k})
        d0{k}(i,:) = feval(executar_modelo_original,x0{k}(i,:),'f');
    end
    dmin{k} = min(d0{k});
    dmax{k} = max(d0{k});
    x0_{k} = x0{k};
    d0_{k} = d0{k};
```

```
      % Divisão entre exemplos de treinamento e teste

    n = nrTotalExemplos(k);
    m = nrTeste;
    ord = randperm(n);
    xTeste{k}       = x0{k}(ord(1:m),:);
    dTeste{k}       = d0{k}(ord(1:m),:);
    xTreinamento{k} = x0{k}(ord(m+1:n),:);
    dTreinamento{k} = d0{k}(ord(m+1:n),:);

      % Divisão entre exemplos de estimação e validação

    n = nrTreinamento;
    m = nrValidacao;
    ord = randperm(n);
    xValidacao{k} = xTreinamento{k}(ord(1:m),:);
    dValidacao{k} = dTreinamento{k}(ord(1:m),:);
    xEstimacao{k} = xTreinamento{k}(ord(m+1:n),:);
    dEstimacao{k} = dTreinamento{k}(ord(m+1:n),:);
    nrTotalExemplos_ = length(xEstimacao{k}(:,1)) + ...
                       length(xValidacao{k}(:,1)) + ...
                       length(xTeste{k}(:,1));
    fprintf(['--> Número de exemplos após divisão =   ' ...
             num2str(nrTotalExemplos_) '\n']);
end
fprintf('Concluído.\n\n');

% ESTIMAÇÃO, VALIDAÇÃO E SELEÇÃO DA MLP

if flagEstimacaoValidacaoSelecaoMLP == true
    fprintf(['Estimando, validando e selecionando a MLP... \n']);
    for i = 1:nrFObj
        if length(xValidacao{i}) == 0
            pt = 0;
        else
            pt = -1;
        end
        eqmMLPMelhor = -1;
        tempo1_ = now;
        for j = 1:nrConfMLP
            fprintf(['--> Função objetivo = ' ...
                     funcaoObjetivo{i} '\n']);
            fprintf(['--> Nr de neurônios = ' ...
                     num2str(nrNeuronios(j)) '\n']);
            tempo1 = now;
            camadasInternas = [nrNeuronios(j)];
            w0 = feval(montar_mlp,length(x0{i}(1,:)), ...
                       [camadasInternas length(d0{i}(1,:))]);
            [w_,err_,et_,indice_] = feval(treinar_mlp, ...
                                          xEstimacao{i}, ...
                                          dEstimacao{i}, ...
                                          pt,w0,a,eta,alf, ...
                                          condP,tol, ...
                                          xValidacao{i}, ...
                                          dValidacao{i}, ...
                                          xmin{i}, ...
```

```
                                        xmax{i},dmin{i},dmax{i});
            et = et_(indice_);
            if eqmMLPMelhor < 0 | eqmMLPMelhor > et
                eqmMLPMelhor =  et;
                wMLPMelhor{i} = w_;
                nrNeuroniosMelhor(i) = nrNeuronios(j);
            end
            tempo2 = now;
            tempo = 24*60*60*(tempo2-tempo1);
            fprintf(['--> Duração (seg) = ' ...
                     num2str(tempo) '\n']);
        end
        tempo2_ = now;
        tempo_ = 24*60*60*(tempo2_-tempo1_);
        fprintf(['--> Função objetivo = ' funcaoObjetivo{i} '\n']);
        fprintf(['--> Melhor nr de neurônios = ' ...
                 num2str(nrNeuroniosMelhor(i)) '\n']);
        fprintf(['--> Erro quadrático médio = ' ...
                 num2str(eqmMLPMelhor) '\n']);
        fprintf(['--> Duração (seg) = ' num2str(tempo_) '\n']);
    end
    fprintf('Concluído.\n\n');
end

% ESTIMAÇÃO, VALIDAÇÃO E SELEÇÃO DA RBFN

if flagEstimacaoValidacaoSelecaoRBF == true
    fprintf(['Estimando, validando e selecionando a RBF... \n']);
    for i = 1:nrFObj
        if length(xValidacao{i}) == 0
            pt = 0;
        else
            pt = -1;
        end
        eqmRBFMelhor = -1;
        tempo1_ = now;
        for j = 1:nrC
            for k = 1:nrPCentros
                nrAmostrasTreinamento = length(xEstimacao{i});
                nrCentros = floor(nrAmostrasTreinamento*pCentros(k));
                fprintf(['--> Função objetivo = ' ...
                         funcaoObjetivo{i} '\n']);
                fprintf(['--> c = ' num2str(c(j)) '\n']);
                fprintf(['--> Nr de centros = ' ...
                         num2str(nrCentros) '\n']);
                eqmRBF = -1;
                tempo1 = now;
                for q = 1:nrTentativasCentros
                    t_ =  [xmin{i}; ...
                           lhsu(xmin{i},xmax{i}, ... % Subseção B.7
                                nrCentros-2); xmax{i}];
                    [wRBF_,eqmRBF_] = feval(treinar_rbf, ...
                                            xEstimacao{i}, ...
                                            dEstimacao{i}, ...
                                            c(j),rbf,t_,lambda, ...
                                            xmin{i},xmax{i}, ...
```

```
                                                dmin{i},dmax{i},pt, ...
                                                xValidacao{i}, ...
                                                dValidacao{i});
                        if eqmRBF < 0 | eqmRBF > eqmRBF_
                            eqmRBF = eqmRBF_;
                            t =      t_;
                            w =      wRBF_;
                        end
                    end
                    tempo2 = now;
                    tempo = 24*60*60*(tempo2-tempo1);
                    if eqmRBFMelhor < 0 | eqmRBFMelhor > eqmRBF
                        eqmRBFMelhor =      eqmRBF;
                        tMelhor{i} =        t;
                        wRBFMelhor{i} =     w;
                        cMelhor(i) =        c(j);
                        pCentrosMelhor(i) = pCentros(k);
                        nrCentrosMelhor(i) = nrCentros;
                    end
                    fprintf(['--> Duração (seg) = ' ...
                             num2str(tempo) '\n']);
                end
            end
            tempo2_ = now;
            tempo_ = 24*60*60*(tempo2_-tempo1_);
            fprintf(['--> Função objetivo = ' funcaoObjetivo{i} '\n']);
            fprintf(['--> Melhor c = ' num2str(cMelhor(i)) '\n']);
            fprintf(['--> Melhor percentual de centros = ' ...
                     num2str(pCentrosMelhor(i)*100) '\n']);
            fprintf(['--> Melhor quantidade de centros = ' ...
                     num2str(nrCentrosMelhor(i)) '\n']);
            fprintf(['--> Erro quadrático médio = ' ...
                     num2str(eqmRBFMelhor) '\n']);
            fprintf(['--> Duração (seg) = ' ...
                     num2str(tempo_) '\n']);
        end
        fprintf('Concluído.\n\n');
end

% DIVISÃO DO CONJUNTO DE EXEMPLOS EM DOIS SUBCONJUNTOS

fprintf(['Dividindo o conjunto de exemplos ' ...
         'em dois subconjuntos... \n']);
for k = 1:nrFObj
    nrEstimacao =   ceil(nrTotalExemplos(k)*(1-pValidacao(k)));
    nrValidacao =   floor(nrTotalExemplos(k)*pValidacao(k));
    fprintf(['--> Função objetivo = ' funcaoObjetivo{k} '\n']);
    fprintf(['--> Número de exemplos para estimação = ' ...
             num2str(nrEstimacao) '\n']);
    fprintf(['--> Número de exemplos para validação = ' ...
             num2str(nrValidacao) '\n']);

      % Divisão entre exemplos de estimação e validação

    n = nrTotalExemplos(k);
    m = nrValidacao;
```

```
    ord = randperm(n);
    xValidacao{k} = x0{k}(ord(1:m),:);
    dValidacao{k} = d0{k}(ord(1:m),:);
    xEstimacao{k} = x0{k}(ord(m+1:n),:);
    dEstimacao{k} = d0{k}(ord(m+1:n),:);
    nrTotalExemplos_ = length(xEstimacao{k}(:,1)) + ...
                       length(xValidacao{k}(:,1));
    fprintf(['--> Número de exemplos após divisão =   ' ...
             num2str(nrTotalExemplos_) '\n']);
end
fprintf('Concluído.\n\n');

% ESTIMAÇÃO E VALIDAÇÃO DA MLP SELECIONADA

if flagEstimacaoValidacaoMLPSelecionada == true
    fprintf(['Estimando e validando a MLP selecionada... \n']);
    for i = 1:nrFObj
        if length(xValidacao{i}) == 0
            pt = 0;
        else
            pt = -1;
        end
        tempo1 = now;
        camadasInternas = [nrNeuroniosMelhor(i)];
        w0 = feval(montar_mlp,length(x0{i}(1,:)), ...
                   [camadasInternas length(d0{i}(1,:))]);
        [w_,err_,et_,indice_] = feval(treinar_mlp, ...
                                      xEstimacao{i}, ...
                                      dEstimacao{i}, ...
                                      pt,w0,a,eta,alf,condP,tol, ...
                                      xValidacao{i}, ...
                                      dValidacao{i}, ...
                                      xmin{i},xmax{i}, ...
                                      dmin{i},dmax{i});
        tempo2 = now;
        tempo = 24*60*60*(tempo2-tempo1);
        wMLP{i} =      w_;
        errMLP{i} =    err_;
        etMLP{i} =     et_;
        indiceMLP(i) = indice_;
        hFig = figure;
        semilogy(1:length(err_),err_,'-b', ...
                 1:length(et_),et_,'-m', ...
                 indice_,err_(indice_),'db');
        xlabel('Épocas de Treinamento');
        ylabel('Erro Quadrático Médio');
        title(funcaoObjetivo{i});
        saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' ...
                     dataExec '- ' ...
                     funcaoObjetivo{i} '- Eqm da MLP.emf'],'emf');
        fprintf(['--> Função objetivo = ' ...
                 funcaoObjetivo{i} '\n']);
        fprintf(['--> Erro quadrático médio = ' ...
                 num2str(etMLP{i}(indiceMLP(i))) '\n']);
        fprintf(['--> Duração (seg) = ' ...
                 num2str(tempo) '\n']);
```

```
    end
    fprintf('Concluído.\n\n');
end

% ESTIMAÇÃO E VALIDAÇÃO DA RBFN SELECIONADA

if flagEstimacaoValidacaoRBFSelecionada == true
    fprintf(['Estimando e validando a RBF selecionada... \n']);
    for i = 1:nrFObj
        if length(xValidacao{i}) == 0
            pt = 0;
        else
            pt = -1;
        end
        if flagPCentros == true
            nrAmostrasTreinamento = length(xEstimacao{i});
            nrCentros = ...
                floor(nrAmostrasTreinamento*pCentrosMelhor(i));
        else
            nrCentros = nrCentrosMelhor(i);
        end
        eqmRBF = -1;
        tempo1 = now;
        for q = 1:nrTentativasCentros
            t_ =  [xmin{i}; ...
                   lhsu(xmin{i},xmax{i}, ...   % Subseção B.7
                        nrCentros-2); xmax{i}];
            [wRBF_,eqmRBF_] = feval(treinar_rbf, ...
                                    xEstimacao{i}, ...
                                    dEstimacao{i}, ...
                                    cMelhor(i),rbf,t_,lambda, ...
                                    xmin{i},xmax{i}, ...
                                    dmin{i},dmax{i},pt, ...
                                    xValidacao{i}, ...
                                    dValidacao{i});
            if eqmRBF < 0 | eqmRBF > eqmRBF_
                eqmRBF = eqmRBF_;
                t =      t_;
                w =      wRBF_;
            end
        end
        tempo2 = now;
        tempo = 24*60*60*(tempo2-tempo1);
        cRBF(i) =  cMelhor(i);
        tRBF{i} =  t;
        wRBF{i} =  w;
        etRBF{i} = eqmRBF;
        fprintf(['--> Função objetivo = ' funcaoObjetivo{i} '\n']);
        fprintf(['--> Erro quadrático médio = ' ...
                 num2str(etRBF{i}) '\n']);
        fprintf(['--> Duração (seg) = ' num2str(tempo) '\n']);
    end
    fprintf('Concluído.\n\n');
end
```

```
% ATIVAÇÃO DA MLP

if flagAtivacaoMLP == true
    fprintf(['Ativando a MLP... \n']);
    for k = 1:nrFObj
        grid1 = xmin{k}(1):(xmax{k}(1)-xmin{k}(1))/bins:xmax{k}(1);
        grid2 = xmin{k}(2):(xmax{k}(2)-xmin{k}(2))/bins:xmax{k}(2);
        modelo = [];
        for i = 1:length(grid1)
            for j = 1:length(grid2)
                entrada = [grid1(i) grid2(j)]';
                saida = feval(ativar_mlp,entrada,wMLP{k}, ...
                              a,xmin{k},xmax{k}, ...
                              dmin{k},dmax{k});
                modelo = [ modelo; entrada' saida{length(saida)}' ];
            end
        end
        modeloMLP{k} = modelo;
    end
    fprintf('Concluído.\n\n');
end

% ATIVAÇÃO DA RBFN

if flagAtivacaoRBF == true
    fprintf(['Ativando a RBF... \n']);
    for k = 1:nrFObj
        grid1 = xmin{k}(1):(xmax{k}(1)-xmin{k}(1))/bins:xmax{k}(1);
        grid2 = xmin{k}(2):(xmax{k}(2)-xmin{k}(2))/bins:xmax{k}(2);
        modelo = [];
        for i = 1:length(grid1)
            for j = 1:length(grid2)
                entrada = [grid1(i) grid2(j)];
                saida = feval(ativar_rbf,entrada,wRBF{k},cRBF(k), ...
                              rbf,tRBF{k}, ...
                              xmin{k},xmax{k}, ...
                              dmin{k},dmax{k});
                modelo = [ modelo; entrada saida ];
            end
        end
        modeloRBF{k} = modelo;
    end
    fprintf('Concluído.\n\n');
end

% PLOTAGEM DAS FUNÇÕES OBJETIVO

fprintf(['Plotando as funções objetivo... \n']);
if flagPlotagemFuncoesObjetivo == true
    for k = 1:nrFObj
        funcaoObjSelecionada = funcaoObjetivo{k};
        [x1,x2] = meshgrid(xmin{k}(1):intervaloGrid(k):xmax{k}(1), ...
                           xmin{k}(2):intervaloGrid(k):xmax{k}(2));
        m = length(x1(1,:));
        n = length(x2(:,1));
        clear f1 f2;
```

```
        for i = 1:m
            for j = 1:n
                x_ = [ x1(i,j) x2(i,j) ];
                f_ = feval(executar_modelo_original,x_,'f');
                f1(i,j) = f_(1);
                f2(i,j) = f_(2);
            end
        end
        hFig = figure;
        surfc(x1,x2,f1);
        title(funcaoObjetivo{k});
        xlabel('x1');
        ylabel('x2');
        zlabel('f1');
        saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' ...
                     dataExec '- ' ...
                     funcaoObjetivo{k} '- x1 x2 f1.emf'],'emf');
        hFig = figure;
        surfc(x1,x2,f2);
        title(funcaoObjetivo{k});
        xlabel('x1');
        ylabel('x2');
        zlabel('f2');
        saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' ...
                     dataExec '- ' ...
                     funcaoObjetivo{k} '- x1 x2 f2.emf'],'emf');
        hFig = figure;
        plot(f1,f2,'.');
        title(funcaoObjetivo{k});
        xlabel('f1');
        ylabel('f2');
        saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' ...
                     dataExec '- ' ...
                     funcaoObjetivo{k} '- f1 f2.emf'],'emf');
    end
end
fprintf('Concluído.\n\n');

% PLOTAGEM DA MLP

if flagPlotagemMLP == true
    fprintf(['Plotando a MLP... \n']);
    for i = 1:nrFObj
        title_ = [ funcaoObjetivo{i} ' - MLP (EQM = ' ...
                   num2str(etMLP{i}(indiceMLP(i))) ')'];
        hFig = figure;

plot3(modeloMLP{i}(:,1),modeloMLP{i}(:,2),modeloMLP{i}(:,3),'.b');
        hold on;
        plot3(x0{i}(:,1),x0{i}(:,2),d0{i}(:,1),'.r');
        hold off;
        title(title_);
        xlabel('x1');
        ylabel('x2');
        zlabel('f1');
        saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' ...
```

```
                    dataExec '- ' ...
                    funcaoObjetivo{i} '- MLP- x1 x2 f1.emf'],'emf');
        hFig = figure;
        plot3(modeloMLP{i}(:,1), ...
              modeloMLP{i}(:,2),modeloMLP{i}(:,4),'.b');
        hold on;
        plot3(x0{i}(:,1),x0{i}(:,2),d0{i}(:,2),'.r');
        hold off;
        title(title_);
        xlabel('x1');
        ylabel('x2');
        zlabel('f2');
        saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' ...
                    dataExec '- ' ...
                    funcaoObjetivo{i} '- MLP- x1 x2 f2.emf'],'emf');
        hFig = figure;
        plot(modeloMLP{i}(:,3),modeloMLP{i}(:,4),'.b');
        hold on;
        plot(d0{i}(:,1),d0{i}(:,2),'.r');
        hold off;
        title(title_);
        xlabel('f1');
        ylabel('f2');
        saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' ...
                    dataExec '- ' ...
                    funcaoObjetivo{i} '- MLP- f1 f2.emf'],'emf');
    end
    fprintf('Concluído.\n\n');
end

% PLOTAGEM DA RBFN

if flagPlotagemRBF == true
    fprintf(['Plotando a RBF... \n']);
    for i = 1:nrFObj
        title_ = [ funcaoObjetivo{i} ' - RBF (EQM = ' ...
                   num2str(etRBF{i}) ')'];
        hFig = figure;
        plot3(modeloRBF{i}(:,1), ...
              modeloRBF{i}(:,2),modeloRBF{i}(:,3),'.b');
        hold on;
        plot3(x0{i}(:,1),x0{i}(:,2),d0{i}(:,1),'.r');
        hold off;
        title(title_);
        xlabel('x1');
        ylabel('x2');
        zlabel('f1');
        saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' ...
                    dataExec '- ' ...
                    funcaoObjetivo{i} '- RBF- x1 x2 f1.emf'],'emf');
        hFig = figure;
        plot3(modeloRBF{i}(:,1), ...
              modeloRBF{i}(:,2),modeloRBF{i}(:,4),'.b');
        hold on;
        plot3(x0{i}(:,1),x0{i}(:,2),d0{i}(:,2),'.r');
        hold off;
```

```
        title(title_);
        xlabel('x1');
        ylabel('x2');
        zlabel('f2');
        saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' ...
                     dataExec '- ' ...
                     funcaoObjetivo{i} '- RBF- x1 x2 f2.emf'],'emf');
        hFig = figure;
        plot(modeloRBF{i}(:,3),modeloRBF{i}(:,4),'.b');
        hold on;
        plot(d0{i}(:,1),d0{i}(:,2),'.r');
        hold off;
        title(title_);
        xlabel('f1');
        ylabel('f2');
        saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' ...
                     dataExec '- ' ...
                     funcaoObjetivo{i} '- RBF- f1 f2.emf'],'emf');
    end
    fprintf('Concluído.\n\n');
end

% OTIMIZAÇÃO SEM O AUXÍLIO DAS REDES NEURAIS

fprintf(['Otimizando sem o auxílio das redes neurais... \n']);
for i = 1:nrFObj
    funcaoObjSelecionada = funcaoObjetivo{i};
    arqPareto = ['Pareto- ' funcaoObjetivo{i} '.txt'];
    if flagMogaExtremo(i) == true
        fprintf(['--> Problema-teste executado ' num2str(i) ' = ' ...
                 funcaoObjetivo{i} '\n']);
        np = npSemRedes(i);
        ng = ngSemRedes(i);
        melhor = melhorFObj{i};
        [pP_,xP_,fP_,xPopIniP_,fPopIniP_,xOriginalP_,fOriginalP_, ...
         contAvaliacoesP_, ...
         contDuplicadosP_, ...
         contInvalidosP_] = feval(moga_binario,np,ng,0,true,0,0, ...
                                  xmin{i},xmax{i}, ...
                                  dmin{i},dmax{i},0, ...
                                  '',0,'',0);
        [ind, id] = ...
            feval(encontrar_solucoes_nao_dominadas,fP_,melhor);
        arq = [xP_(ind,:) fP_(ind,:)];
        save(arqPareto,'arq','-ASCII');
    else
        fprintf(['--> Problema-teste carregado ' num2str(i) ' = ' ...
                 funcaoObjetivo{i} '\n']);
        arq = load(arqPareto);
        xP_ = arq(:,1:2);
        fP_ = arq(:,3:4);
    end
    xPSemRedes{i} = xP_;
    fPSemRedes{i} = fP_;
    [xP_ordenado,indice] = sort(fP_(:,1));
    xP_ = xP_(indice,:);
```

```
    fP_ = fP_(indice,:);
    nrPontos = length(indice);
    fP_cel = [];
    for j = 1:nrPontos-1
        fP_cel = [fP_cel; fP_(j+1,1) fP_(j,2)];
    end
    fPCelula{i} = fP_cel;
    hFig = figure;
    plot(fP_(:,1),fP_(:,2),'o',fP_cel(:,1),fP_cel(:,2),'+');
    xlabel('f1');
    ylabel('f2');
    legend('Curva de Pareto','Células');
    title(funcaoObjetivo{i});
    saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' ...
                 dataExec '- ' ...
                 funcaoObjetivo{i} '- Pareto- f1 f2.emf'],'emf');
end
fprintf('Concluído.\n\n');

% OTIMIZAÇÃO COM O AUXÍLIO DA MLP

if flagOtimizacaoAuxilioMLP == true
    fprintf(['Otimizando com o auxílio da MLP... \n']);
    for i = 1:nrFObj
        melhor = melhorFObj{i};
        if length(xValidacao{i}) == 0
            pt = 0;
        else
            pt = -1;
        end
        camadasInternas = [nrNeuroniosMelhor(i)];
        [probAceite_, ...
         probUsoMedio_, ...
         avalModSub_, ...
         ip_] = feval(executar_moga, ...
                     funcaoObjetivo{i},nrEnsaios(i), ...
                     a,camadasInternas, ...
                     pt,eta,alf,condP,tol, ...
                     npComRedes(i),ngComRedesPorEpoc(i), ...
                     nEpoc(i),probUso0(i), ...
                     xEstimacao{i},dEstimacao{i}, ...
                     xValidacao{i},dValidacao{i}, ...
                     xPSemRedes{i},fPSemRedes{i}, ...
                     xmin{i},xmax{i}, ...
                     dmin{i},dmax{i}, ...
                     'MLP',0,'',0,0,0, ...
                     fPCelula{i});
        probAceiteMLP(i) = probAceite_;
        probUsoMedio(i) =  probUsoMedio_;
        avalModSubMLP(i) = avalModSub_;
        ipMLP(i) =         ip_;
    end
    fprintf('Concluído.\n\n');
end
```

```
% OTIMIZAÇÃO COM O AUXÍLIO DA RBFN

if flagOtimizacaoAuxilioRBF == true
    fprintf(['Otimizando com o auxílio da RBF... \n']);
    for i = 1:nrFObj
        melhor = melhorFObj{i};
        if length(xValidacao{i}) == 0
            pt = 0;
        else
            pt = -1;
        end
        if flagPCentros == true
            nrAmostrasTreinamento = length(xEstimacao{i});
            nrCentros = floor(nrAmostrasTreinamento* ...
                              pCentrosMelhor(i));
        else
            nrCentros = nrCentrosMelhor(i);
        end
        [probAceite_, ...
         probUsoMedio_, ...
         avalModSub_, ...
         ip_] = feval(executar_moga, ...
                      funcaoObjetivo{i},nrEnsaios(i), ...
                      0,0,pt,0,0,0,0, ...
                      npComRedes(i),ngComRedesPorEpoc(i), ...
                      nEpoc(i),probUso0(i), ...
                      xEstimacao{i},dEstimacao{i}, ...
                      xValidacao{i},dValidacao{i}, ...
                      xPSemRedes{i},fPSemRedes{i}, ...
                      xmin{i},xmax{i}, ...
                      dmin{i},dmax{i}, ...
                      'RBF',cRBF(i),rbf,nrCentros, ...
                      lambda,nrTentativasCentros, ...
                      fPCelula{i});
        probAceiteRBF(i) = probAceite_;
        probUsoMedio(i) =  probUsoMedio_;
        avalModSubRBF(i) = avalModSub_;
        ipRBF(i) =         ip_;
    end
    fprintf('Concluído.\n\n');
end

% COMPARAÇÃO DA EFICIÊNCIA DAS REDES NEURAIS

fprintf(['Comparando a eficiência das redes neurais... \n']);
for i = 1:nrFObj
    fprintf(['--> Problema-teste ' num2str(i) ' = ...
             ' funcaoObjetivo{i} '\n']);
    funcaoObjetivo_{i} = funcaoObjetivo{i};
    nrAmostras = length(xEstimacao{i}(:,1)) + ...
                 length(xValidacao{i}(:,1));
    melhor = melhorFObj{i};
    if flagCalculoEficienciaMLP == true
        ganhoMLP(i) = feval(calcular_eficiencia, ...
                            funcaoObjetivo{i},probAceiteMLP(i), ...
                            nrEnsaiosEta(i),nrAmostras, ...
```

```
                            probUsoMedio(i),passoNg(i), ...
                            npComRedes(i),ngComRedesPorEpoc(i), ...
                            nEpoc(i), ...
                            xmin{i},xmax{i}, ...
                            dmin{i},dmax{i}, ...
                            xPSemRedes{i},fPSemRedes{i});
        etaMLP(i) = ganhoMLP(i)*ipMLP(i);
        fprintf(['--> Ganho da MLP = ' num2str(ganhoMLP(i)) '\n']);
        fprintf(['--> Índice de preenchimento da MLP = ' ...
                 num2str(ipMLP(i)) '\n']);
        fprintf(['--> Eficiência da MLP = ' ...
                 num2str(etaMLP(i)) '\n']);
    else
        ganhoMLP(i) = 0;
        ipMLP(i) =    0;
        etaMLP(i) =   0;
    end
    if flagCalculoEficienciaRBF == true
        ganhoRBF(i) = feval(calcular_eficiencia, ...
                            funcaoObjetivo{i},probAceiteRBF(i), ...
                            nrEnsaiosEta(i),nrAmostras, ...
                            probUsoMedio(i),passoNg(i), ...
                            npComRedes(i),ngComRedesPorEpoc(i), ...
                            nEpoc(i), ...
                            xmin{i},xmax{i}, ...
                            dmin{i},dmax{i}, ...
                            xPSemRedes{i},fPSemRedes{i});
        etaRBF(i) = ganhoRBF(i)*ipRBF(i);
        fprintf(['--> Ganho da RBF = ' num2str(ganhoRBF(i)) '\n']);
        fprintf(['--> Índice de preenchimento da RBF = ' ...
                 num2str(ipRBF(i)) '\n']);
        fprintf(['--> Eficiência da RBF = ' num2str(etaRBF(i)) '\n']);
    else
        ganhoRBF(i) = 0;
        ipRBF(i) =    0;
        etaRBF(i) =   0;
    end
end
hFig = figure;
bar([ganhoMLP' ganhoRBF'],'group');
title('Ganho das redes neurais');
legend('MLP','RBF',-1);
set(gca,'XTickLabel',funcaoObjetivo_);
saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' dataExec ...
             '- Ganho.emf'],'emf');
hFig = figure;
bar([ipMLP' ipRBF'],'group');
title('Índice de preenchimento');
legend('MLP','RBF',-1);
set(gca,'XTickLabel',funcaoObjetivo_);
saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' dataExec ...
             '- Índice de preenchimento.emf'],'emf');
hFig = figure;
bar([etaMLP' etaRBF'],'group');
title('Eficiência das redes neurais');
legend('MLP','RBF',-1);
```

```
set(gca,'XTickLabel',funcaoObjetivo_);
saveas(hFig,['Registros\v20120407_comparar_eficiencia- ' dataExec ...
             '- Eficiência.emf'],'emf');
fprintf('Concluído.\n\n');
fprintf('Fim da rotina.\n\n');
diary off;
warning on all;
```

B.2 Rotinas da perceptron de múltiplas camadas

B.2.1 Definição da arquitetura

```
% NOME DA ROTINA: v20110711_montar_mlp

function y = v20110711_montar_mlp(e,n)

% ENTRADAS

      % e: número de entradas
      % n: vetor contendo número de neurônios por camada

% SAÍDAS

      % y: matrizes de pesos (w)

% DEFINIÇÃO DA ESTRUTURA

n = [e n];
for i = 1:length(n)-1
    y{i} = rand(n(i+1),n(i)+1)*2-ones(n(i+1),n(i)+1);
end
```

B.2.2 Treinamento da rede neural

```
% NOME DA ROTINA: v20120523_treinar_mlp

function [y,err,et,indice] = ...
    v20120523_treinar_mlp(x,d,pt,w,a,eta,alf, ...
                          condP,tol,xt,dt, ...
                          xmin,xmax,dmin,dmax)

% ENTRADAS

      % x:      padrões de entrada
      %         Linha:  ocorrência
      %         Coluna: entrada
      % d:      padrões de saída
      % pt:     percentual de padrões para teste
      %         Se pt = -1, xt e dt são considerados
      % w:      matrizes de pesos
```

```
      %         Lista:  camada
      %         Linha:  neurônio
      %         Coluna: entrada (+ bias)
      % a:      coeficiente da função de ativação
      % eta:    taxa de aprendizagem
      % alf:    momento
      % condP:  limite de épocas de treinamento
      % tol:    tolerância do erro
      % xt:     padrões de entrada para validação
      % dt:     padrões de saída para validação
      % xmin:   valor mínimo dos padrões de entrada
      % xmax:   valor máximo dos padrões de entrada
      % dmin:   valor mínimo dos padrões de saída
      % dmax:   valor máximo dos padrões de saída

% SAÍDAS

      % y:      w após o treinamento
      % err:    erro quadrático médio dos padrões de treinamento
      % et:     erro quadrático médio dos padrões de teste
      % indice: epoca em que a diferença absoluta entre err e
      %         et é mínima

% PARTES DO CÓDIGO

      % Rotinas externas
      % Divisão aleatória dos padrões para treinamento e teste
      % Normalização
      % Inicialização
      % Treinamento

% ROTINAS EXTERNAS

global ativar_mlp;      % Subseção B.2.3
global normalizar;      % Subseção B.4.1

% DIVISÃO ALEATÓRIA DOS PADRÕES PARA TREINAMENTO E TESTE

if pt >= 0
    n = length(x(:,1));
    m = floor(n*pt/100);
    ord = randperm(n);
    xt = x(ord(1:m),:);       % Padrões para teste
    dt = d(ord(1:m),:);
    Nt = length(xt(:,1));
    x = x(ord(m+1:n),:);      % Padrões para treinamento
    d = d(ord(m+1:n),:);
    N = length(x(:,1));       % Quantidade de padrões para treinamento
else
    Nt = length(xt(:,1));     % Quantidade de padrões de teste
    N = length(x(:,1));       % Quantidade de padrões de treinamento
end

% NORMALIZAÇÃO

x = feval(normalizar,x,xmin,xmax);
```

```
d = feval(normalizar,d,dmin,dmax);

% INICIALIZAÇÃO

dw = w;
s = length(w);
for i = 1:s
    dw{i} = dw{i}*0;
    grad{i} = 0;
end
for i = 1:s-1
    bias(i) = 1;
end
bias(s) = 0;

% TREINAMENTO

dErr = tol + 1;
epoca = 0;
err_ = 0;
while epoca < condP
    epoca = epoca + 1;
    err(epoca) = 0;
    ord = randperm(length(x(:,1)));      % Embaralhamento dos índices
    for i = 1:N

      % Ativação

        entrada = x(ord(i),:)';
        padrao  = d(ord(i),:)';
        saida = feval(ativar_mlp,entrada,w,a, ...
                      zeros(1,length(entrada)), ...
                      ones(1,length(entrada)), ...
                      zeros(1,length(padrao)),ones(1,length(padrao)));

      % Cálculo do erro

        e = padrao - saida{s};
        e_ = e.*(dmax-dmin)';
        err(epoca) = err(epoca) + e_'*e_;

      % Cálculo dos gradientes locais

        grad{s} = e;    % Saída linear
        for j = s-1:-1:1
            v = w{j+1}(:,2:length(w{j+1}))'*grad{j+1};
            grad{j} = a*saida{j}.*(ones(length(saida{j}),1)- ...
                      saida{j}).*v;
        end

      % Ajuste dos pesos sinápticos

        w{1} = w{1} + alf*dw{1} + eta*(grad{1}*[bias(1); entrada]');
        dw{1} = eta*(grad{1}*[bias(1); entrada]');
        for j = 2:s
            w{j} = w{j} + alf*dw{j} + ...
```

```
                    eta*(grad{j}*[bias(j); saida{j-1}]');
            dw{j} = eta*(grad{j}*[bias(j); saida{j-1}]');
        end
    end
    err(epoca) = err(epoca)/(2*N); % Cálculo do erro quadrático médio
    dErr = abs(err(epoca)-err_);
    err_ = err(epoca);
    eta = eta*.9985^(epoca/condP); % Decaimento monotônico da taxa de
                                   % apredizagem
    alf = alf*.9985^(epoca/condP); % Decaimento monotônico do momento

      % Validação

    et(epoca) =  0;
    if pt ~= 0
        for i = 1:Nt
            saida = feval(ativar_mlp,xt(i,:)',w,a, ...
                          xmin,xmax,dmin,dmax);
            e = dt(i,:)' - saida{s};
            et(epoca) =  et(epoca)  + e'*e;
        end
        et(epoca) = ...
            et(epoca)/(2*Nt);      % Erro quadrático médio do teste
    end
    w_{epoca} = w;
end
if pt ~= 0
    indice = find(et==min(et));
    indice = indice(1);
    y = w_{indice};
else
    indice = epoca;
    y = w_{indice};
end
```

B.2.3 Ativação da rede neural

```
% NOME DA ROTINA: v20120523_ativar_mlp

function y = v20120523_ativar_mlp(x,w,a,xmin,xmax,dmin,dmax)

% ENTRADAS

      % x:    entrada sem bias (vetor coluna)
      % w:    matrizes de pesos
      %       Lista:  camada
      %       Linha:  neurônio
      %       Coluna: entrada (+ bias)
      % a:    coeficiente da função de ativação
      % xmin:   valor mínimo dos padrões de entrada
      % xmax:   valor máximo dos padrões de entrada
      % dmin:   valor mínimo dos padrões de saída
      % dmax:   valor máximo dos padrões de saída
```

```
% SAÍDA

      % y: saída da última camada (vetor coluna)

global contAvaliacoesMLP;
global normalizar;      % Subseção B.4.1
global desnormalizar;   % Subseção B.4.2

x = feval(normalizar,x',xmin,xmax)';
clear y;
n = length(w);
for i = 1:n-1
    bias(i) = 1;
end
bias(n) = 0;
for camada = 1:n
    x = [bias(camada); x];
    v = w{camada}*x;
    if camada < n
        y{camada} = 1./(1+exp(-a*v));
    else
        y{camada} = v;
    end
    x = y{camada};
end
y{length(y)} = feval(desnormalizar,y{length(y)}',dmin,dmax)';
contAvaliacoesMLP = contAvaliacoesMLP + 1;
```

B.3 Rotinas da rede de função de base radial

B.3.1 Treinamento da rede neural

```
% NOME DA ROTINA: v20120130_treinar_rbf

function [w,eqm] = v20120130_treinar_rbf(x,d,c,rbf,t,lambda,xmin, ...
                                         xmax,dmin,dmax,pt,xt,dt)

% ENTRADAS

      % x:      padrões de entrada
      %         Linha:  ocorrência
      %         Coluna: entrada
      % d:      padrões de saída
      % c:      desvio padrão da função de base radial
      % rbf:    função de base radial
      % t:      centros
      % lambda: parâmetro regularização
      % xmin:   valor mínimo de x
      % xmax:   valor máximo de x
      % dmin:   valor mínimo de d
      % dmax:   valor máximo de d
      % pt:     percentual de padrões para teste
      %         Se pt = -1, xt e dt passados são considerados
```

```
      % xt:     padrões de entrada para validação
      % dt:     padrões de saída para validação

% SAÍDAS

      % w:      pesos sinápticos
      % eqm:    erro quadrático médio do teste

% PARTES DO CÓDIGO

      % Divisão dos padrões para treinamento e teste
      % Normalização de 0 a 100
      % Treinamento
      % Teste

% ROTINAS EXTERNAS

global determinar_rbf_bias;   % Subseção B.3.2
global ativar_rbf;            % Subseção B.3.3
global normalizar;            % Subseção B.4.1

% DIVISÃO DOS PADRÕES PARA TREINAMENTO E TESTE

if pt >= 0
    n = length(x(:,1));
    m = floor(n*pt/100);
    ord = randperm(n);
    xt = x(ord(1:m),:);       % Padrões para teste
    dt = d(ord(1:m),:);
    Nt = length(xt(:,1));
    x = x(ord(m+1:n),:);      % Padrões para treinamento
    d = d(ord(m+1:n),:);
    N = length(x(:,1));       % Quantidade de padrões para treinamento
else
    Nt = length(xt(:,1));     % Quantidade de padrões de teste
    N = length(x(:,1));       % Quantidade de padrões de treinamento
end

% NORMALIZAÇÃO ENTRE 0 E 100

x =  feval(normalizar,x,xmin,xmax)*100;
d =  feval(normalizar,d,dmin,dmax)*100;
t_ = feval(normalizar,t,xmin,xmax)*100;

% TREINAMENTO

n = length(d(1,:));
for i = 1:n
    w(:,i) = feval(determinar_rbf_bias,x,d(:,i),c,rbf,t_,lambda);
end

% TESTE

y = feval(ativar_rbf,xt,w,c,rbf,t,xmin,xmax,dmin,dmax);
eqm =  0;
Nt = length(xt(:,1));
```

```
for i = 1:Nt
    e = dt(i,:) - y(i,:);
    eqm =  eqm  + e*e';
end
eqm =  eqm/(2*Nt);            % Erro quadrático médio do teste
```

B.3.2 Determinação dos pesos sinápticos

```
% NOME DA ROTINA: v20110711_determinar_rbf_bias

function w = v20110711_determinar_rbf_bias(x,d,c,rbf,t,lambda)

% ENTRADAS

      % x:      padrões de entrada
      %         Linha:  ocorrência
      %         Coluna: entrada
      % d:      padrões de saída
      % c:      desvio padrão da função de base radial
      % rbf:    função de base radial
      % t:      centros
      % lambda: parâmetro regularização

% SAÍDA

      % w:      pesos sinápticos

N = length(x(:,1));
M = length(t(:,1));
phi =  zeros(N,M);
phi0 = zeros(M,M);
for j = 1:N
    for i = 1:M
        phi(j,i) = feval(rbf,(x(j,:)-t(i,:))',c);
    end
end
for j = 1:M
    for i = 1:M
        phi0(j,i) = feval(rbf,(t(j,:)-t(i,:))',c);
    end
end
phi =  [ ones(N,1) phi  ];
phi0 = [ 1         ones(1,M); ...
         ones(M,1) phi0    ];
w = inv( phi'*phi + lambda*phi0 )*phi'*d;
```

B.3.3 Ativação da rede neural

```
% NOME DA ROTINA: v20120130_ativar_rbf

function y = v20120130_ativar_rbf(x,w,c,rbf,t,xmin,xmax,dmin,dmax)
```

```
% ENTRADAS

      % x:      padrões de entrada
      %         Linha:  ocorrência
      %         Coluna: entrada
      % w:      pesos sinápticos
      % c:      desvio padrão da função de base radial
      % rbf:    função de base radial
      % t:      centros
      % xmin:   valor mínimo de x
      % xmax:   valor máximo de x
      % dmin:   valor mínimo dos padrões de saída
      % dmax:   valor máximo dos padrões de saída

% SAÍDA

      % y:      resposta da rede

global executar_rbf_bias_2;   % Subseção B.3.4
global normalizar;            % Subseção B.4.1
global desnormalizar;         % Subseção B.4.2

% NORMALIZAÇÃO

x = feval(normalizar,x,xmin,xmax)*100;
t = feval(normalizar,t,xmin,xmax)*100;

% ATIVAÇÃO

n = length(w(1,:));
for i = 1:n
    y(:,i) = feval(executar_rbf_bias_2,x,w(:,i),c,rbf,t);
end

% DESNORMALIZAÇÃO

y = feval(desnormalizar,y/100,dmin,dmax);
```

B.3.4 Resposta do neurônio de saída

```
% NOME DA ROTINA: v20110711_executar_rbf_bias_2

function y = v20110711_executar_rbf_bias_2(x,w,c,rbf,t)

% ENTRADAS

      % x:      padrões de entrada
      %         Linha:  ocorrência
      %         Coluna: entrada
      % w:      pesos sinápticos
      % c:      desvio padrão da função de base radial
      % rbf:    função de base radial
      % t:      centros
```

```
% SAÍDA

      % y:      resposta do neurônio de saída

N = length(x(:,1));
M = length(t(:,1));
phi = zeros(N,M);
for j = 1:N
    for i = 1:M
        phi(j,i) = feval(rbf,(x(j,:)-t(i,:))',c);
    end
end
phi = [ ones(N,1) phi ];
y = phi*w;
```

B.4 Rotinas compartilhadas entre as redes neurais

B.4.1 Normalização da entrada da rede neural

```
% NOME DA ROTINA: v20110711_normalizar

function y = v20110711_normalizar(x,xmin,xmax)

% ENTRADAS

      % x:    original
      %       Linha:  ocorrência
      %       Coluna: entrada
      % xmin: valor mínimo de x
      % xmax: valor máximo de x

% SAÍDA

      % y:    x normalizado no intervalo de 0 a 1

y = [];
for ocorrencia = 1:length(x(:,1))
    y = [y; (x(ocorrencia,:)-xmin)./(xmax-xmin)];
end
```

B.4.2 Desnormalização da resposta da rede neural

```
% NOME DA ROTINA: v20110711_desnormalizar

function y = v20110711_desnormalizar(x,xmin,xmax)

% ENTRADA

      % x:    normalizado no intervalo de 0 a 1
```

```
      %       Linha:  ocorrência
      %       Coluna: entrada
      % xmin: valor mínimo de x
      % xmax: valor máximo de x

% SAÍDA

      % y: x desnormalizado

y = [];
for ocorrencia = 1:length(x(:,1))
    y = [y; x(ocorrencia,:).*(xmax-xmin)+xmin];
end
```

B.5 Rotinas de otimização

B.5.1 Versão binária do algoritmo genético multiobjetivo não-geracional

```
% NOME DA ROTINA: v20110711_moga_binario

function [p,x,f,x0,f0,xOriginal,fOriginal, ...
          contAvaliacoes, ...
          contDuplicados, ...
          contInvalidos] = ...
    v20110711_moga_binario(np,ng,p0,flagRand, ...
                           w,a,xmin,xmax,dmin,dmax, ...
                           probUsoMLP, ...
                           modSub,c,rbf,t)

% ENTRADAS

      % np:                       número de indivíduos na população
      % ng:                       número de gerações
      % p0:                       população inicial na representação
      %                           binária
      % flagRand:                 se TRUE despreza p0 e gera população
      %                           inicial
      %                           aleatoriamente
      % w:                        pesos sinápticos da MLP
      % a:                        parâmetro da função de ativação
      % xmin:                     limite inferior do domínio
      % xmax:                     limite superior do domínio
      % dmin:                     limite inferior do contradomínio
      % dmax:                     limite superior do contradomínio
      % probUsoMLP:               probabilidade de uso da MLP
      % modSub:                   modelo substituto, se 'MLP' ou 'RBF'
      % c:                        parâmetro da função de base radial
      % rbf:                      função de base radial
      % t:                        centros da função de base radial

% SAÍDAS

      % p:                        população na representação binária
```

```
      % x:                          população na representação real
      % f:                          avaliação da população
      % x0:                         população inicial na representação
      %                             real
      % f0:                         avaliação de x0
      % xOriginal:                  população avaliada pela função
      %                             objetivo
      %                             original
      % fOriginal:                  avaliação de xOriginal
      % contAvaliacoes:             contador do número de avaliações
      % contDuplicados:             contador do número de indivíduos
      %                             duplicados
      % contInvalidos:              contador do número de indivíduos
      %                             inválidos

global nv;        % Número de variáveis objetivo
global xInf;      % Limite inferior da variável objetivo
global xSup;      % Limite superior da variável objetivo
global nobj;      % Número de funões objetivo
global nbv;       % Número de bits necessários para representar uma
                  % variável
global pSementes;
global xSementes;
global fSementes;
global contAvaliacoes;
global contDuplicados;
global contInvalidos;
global melhor;    % Máximo/ mínimo

% ROTINAS EXTERNAS

global FuncObj;   % Subseção B.5.3

% CONFIGURAÇÃO

feval(FuncObj,0,false,true,w,a, ...
              xmin,xmax,dmin,dmax, ...
              probUsoMLP, ...
              modSub,c,rbf,t);
xOriginal = [];
fOriginal = [];
raioVizinhanca = 50;
fatorGraduacao = .5;
probRecombinacao = .8;
operadorAleatorio = true;
operador = 2;
hamming = true;
probMutacao = .2;

% INICIALIZAÇÃO DA POPULAÇÃO

ni = nv*nbv;            % Tamanho de um indivíduo da população em bits
if flagRand == true
    p = randint( np, ni );    % Domínio binário
else
    p = p0;
```

```
end
x = zeros( np, nv );    % Domínio real
f = zeros( np, nobj );  % Contradomínio real
contInvalidos = 0;
contDuplicados = 0;
contAvaliacoes = 0;
for ii = 1:np
    repeteIndividuo = true;
    while( repeteIndividuo == true )
        validade = false;
        while( validade == false )
            [x(ii,:),f(ii,:),validade,flagOriginal] = ...
                feval(FuncObj,p(ii,:),true,true,w,a, ...
                              xmin,xmax,dmin,dmax, ...
                              probUsoMLP, ...
                              modSub,c,rbf,t);
            if flagOriginal == 1
                xOriginal = [xOriginal; x(ii,:)];
                fOriginal = [fOriginal; f(ii,:)];
            end
            contAvaliacoes = contAvaliacoes + 1;
            if( validade == false )
                contInvalidos = contInvalidos + 1;
                p(ii,:) = randint( 1, ni );
            end
        end
        repeteIndividuo = false;
        for jj = 1:ii-1
            ijIguais = true;
            for kk = 1:ni
                if( p(ii,kk) ~= p(jj,kk) )
                    ijIguais = false;
                    break;
                end
            end
            if( ijIguais == true )
                contDuplicados = contDuplicados + 1;
                p(ii,:) = randint( 1, ni );
                repeteIndividuo = true;
                break;
            end
        end
    end
end

% AVALIAÇÃO DA POPULAÇÃO

nd = zeros(np);
dv = zeros(np);
aptidao = zeros(1,np);
aptidaoTeste = zeros(1,np);
ordem = zeros(1,np);
dominancia = zeros(1,np);
vizinhanca = zeros(1,np);
for ii = 1:np-1
    for jj = ii+1:np
```

```
        iDomina = 0;
        jDomina = 0;
        distancia = 0;
        for kk = 1:nobj
            if( melhor(kk) == 0 )
                if( f(ii,kk) < f(jj,kk) )
                    iDomina = iDomina + 1;
                elseif( f(jj,kk) < f(ii,kk) )
                    jDomina = jDomina + 1;
                end
            else
                if( f(ii,kk) > f(jj,kk) )
                    iDomina = iDomina + 1;
                elseif( f(jj,kk) > f(ii,kk) )
                    jDomina = jDomina + 1;
                end
            end
            distancia = distancia + ( f(ii,kk) - f(jj,kk) )^2;
        end
        nd(ii,jj) = 0;
        nd(jj,ii) = 0;
        if( iDomina > 0 & jDomina == 0 )
            nd(ii,jj) = 1;
        elseif( iDomina == 0 & jDomina > 0 )
            nd(jj,ii) = 1;
        end
        dv(ii,jj) = 0;
        dv(jj,ii) = 0;
        distancia = sqrt( distancia );
        if( distancia < raioVizinhanca )
            dv(ii,jj) = 1 - ...
                        ( distancia/ raioVizinhanca )^fatorGraduacao;
            dv(jj,ii) = dv(ii,jj);
        end
    end
end
for ii = 1:np
    for jj = 1:np
        dominancia(ii) = dominancia(ii) + nd(jj,ii);
        vizinhanca(ii) = vizinhanca(ii) + dv(ii,jj);
    end
    aptidao(ii) = ( 1 + dominancia(ii) )*( 1 + vizinhanca(ii) );
end
[ aptidaoTeste, ordem ] = sort( aptidao );
x0 = x;
f0 = f;

% EVOLUÇÃO DA POPULAÇÃO

filho = zeros(2,ni);
xFilho = zeros(2,nv);
fFilho = zeros(2,nobj);
validadeFilho = zeros(1,2);
ndFilho = zeros(2,np);
ndFilhoJ = zeros(np,2);
dvFilho = zeros(2,np);
```

```
aptidaoFilho = zeros(1,2);
dominanciaFilho = zeros(1,2);
vizinhancaFilho = zeros(1,2);
contGeracoes = 0;
termino = false;
while( termino == false )
    filhosValidos = false;
    contPerda = 0;
    while( filhosValidos == false )

      % Seleção

        selecao = ceil( np*( 1.5 - sqrt( 2.25 - 2*rand(1) )) );
        indicePai = ordem(selecao);
        while( ordem(selecao) == indicePai )
            selecao = ceil( np*( 1.5 - sqrt( 2.25 - 2*rand(1) )) );
        end
        indiceMae = ordem(selecao);
        pai = p(indicePai,:);
        mae = p(indiceMae,:);

      % Recombinação

        filho(1,:) = pai;
        filho(2,:) = mae;
        if( probRecombinacao >= rand(1) )
            if( operadorAleatorio == true )
                operador = randint( 1, 1, [1, 3] );
            end
            switch( operador )
                case 1
                    corte1 = randint( 1, 1, [2, ni] );
                    for ii = corte1:ni
                        filho(1,ii) = mae(ii);
                        filho(2,ii) = pai(ii);
                    end
                case 2
                    corte2 = randint( 1, 2, [2, ni] );
                    corte2 = sort(corte2);
                    for ii = corte2(1):corte2(2)
                        filho(1,ii) = mae(ii);
                        filho(2,ii) = pai(ii);
                    end
                otherwise
                    for ii = 1:ni
                        if( pai(ii) ~= mae(ii) )
                            if( rand(1) > .5 )
                                filho(1,ii) = mae(ii);
                                filho(2,ii) = pai(ii);
                            end
                        end
                    end
            end
        end
```

```
% Mutação

  if( hamming == true )
      probMutacao = ( 1 - ( sum( pai ~= mae )/ni) )*2/ni;
  end
  for ii = 1:2
      for jj = 1:ni
          if( probMutacao >= rand(1) )
              filho(ii,jj) = ~filho(ii,jj);
          end
      end
  end

% Comparação dos descendentes com a população

  filhosValidos = true;
  for ii = 1:2
      for jj = 1:np
          filhoIgual(ii) = true;
          for kk = 1:ni
              if( filho(ii,kk) ~= p(jj,kk) )
                  filhoIgual(ii) = false;
                  break;
              end
          end
          if( filhoIgual(ii) == true )
              break;
          end
      end
  end
  if( filhoIgual(1) == true & filhoIgual(2) == true )
      filhosValidos = false;
      contPerda = contPerda + 1;
  elseif( filhoIgual(1) == false & filhoIgual(2) == true )
      filho(2,:) = filho(1,:);
  elseif( filhoIgual(1) == true & filhoIgual(2) == false )
      filho(1,:) = filho(2,:);
  end

% Verificação da validade dos descendentes

  if( filhosValidos == true )
      [xFilho(1,:),fFilho(1,:), ...
       validadeFilho(1),flagOriginal] = ...
          feval(FuncObj,filho(1,:),true,true,w,a, ...
                        xmin,xmax,dmin,dmax, ...
                        probUsoMLP, ...
                        modSub,c,rbf,t);
      if flagOriginal == 1
          xOriginal = [xOriginal; xFilho(1,:)];
          fOriginal = [fOriginal; fFilho(1,:)];
      end
      contAvaliacoes = contAvaliacoes + 1;
      [xFilho(2,:),fFilho(2,:), ...
       validadeFilho(2),flagOriginal] = ...
          feval(FuncObj,filho(2,:),true,true,w,a, ...
```

```
                        xmin,xmax,dmin,dmax, ...
                        probUsoMLP, ...
                        modSub,c,rbf,t);
        if flagOriginal == 1
            xOriginal = [xOriginal; xFilho(2,:)];
            fOriginal = [fOriginal; fFilho(2,:)];
        end
        contAvaliacoes = contAvaliacoes + 1;
        if( validadeFilho(1) == false & ...
            validadeFilho(2) == false )
            filhosValidos = false;
            contPerda = contPerda + 1;
        elseif( validadeFilho(1) == true & ...
                validadeFilho(2) == false )
            filho(2,:) = filho(1,:);
            xFilho(2,:) = xFilho(1,:);
            fFilho(2,:) = fFilho(1,:);
        elseif( validadeFilho(1) == false & ...
                validadeFilho(2) == true )
            filho(1,:) = filho(2,:);
            xFilho(1,:) = xFilho(2,:);
            fFilho(1,:) = fFilho(2,:);
        end
    end

% Seleção do descendente

    if( filhosValidos == true )
        for ii = 1:2
            for jj = 1:np
                iDomina = 0;
                jDomina = 0;
                distancia = 0;
                for kk = 1:nobj
                    if( melhor(kk) == 0 )
                        if( fFilho(ii,kk) < f(jj,kk) )
                            iDomina = iDomina + 1;
                        elseif( f(jj,kk) < fFilho(ii,kk) )
                            jDomina = jDomina + 1;
                        end
                    else
                        if( fFilho(ii,kk) > f(jj,kk) )
                            iDomina = iDomina + 1;
                        elseif( f(jj,kk) > fFilho(ii,kk) )
                            jDomina = jDomina + 1;
                        end
                    end
                    distancia = distancia + ...
                                ( fFilho(ii,kk) - f(jj,kk) )^2;
                end
                ndFilho(ii,jj) = 0;
                ndFilhoJ(jj,ii) = 0;
                if( iDomina > 0 & jDomina == 0 )
                    ndFilho(ii,jj) = 1;
                elseif( iDomina == 0 & jDomina > 0 )
                    ndFilhoJ(jj,ii) = 1;
```

```
                end
                dvFilho(ii,jj) = 0;
                distancia = sqrt( distancia );
                if( distancia < raioVizinhanca )
                    dvFilho(ii,jj) = 1 - ...
                        ( distancia/ ...
                          raioVizinhanca )^fatorGraduacao;
                end
            end
        end
        for ii = 1:2
            dominanciaFilho(ii) = 0;
            vizinhancaFilho(ii) = 0;
            for jj = 1:np
                dominanciaFilho(ii) = dominanciaFilho(ii) + ...
                                      ndFilhoJ(jj,ii);
                vizinhancaFilho(ii) = vizinhancaFilho(ii) + ...
                                      dvFilho(ii,jj);
            end
            aptidaoFilho(ii) = ( 1 + dominanciaFilho(ii) )* ...
                               ( 1 + vizinhancaFilho(ii) );
        end
        if( aptidaoFilho(1) <= aptidaoFilho(2) )
            filhoVencedor = 1;
        else
            filhoVencedor = 2;
        end
        for ii = 1:np
            aptidaoTeste1 = 1 + dominancia(ii) + ...
                            ndFilho(filhoVencedor,ii);
            aptidaoTeste2 = 1 + vizinhanca(ii) + ...
                            dvFilho(filhoVencedor,ii);
            aptidaoTeste(ii) = aptidaoTeste1 * aptidaoTeste2;
        end
        [ aptidaoTeste, ordem ] = sort( aptidaoTeste );
        indicePior = ordem(np);
        if( aptidaoFilho(filhoVencedor) > aptidaoTeste(np) )
            filhosValidos = false;
            contPerda = contPerda + 1;
        end
    end
end

  % Substituição do pior indivíduo

p(indicePior,:) = filho(filhoVencedor,:);
x(indicePior,:) = xFilho(filhoVencedor,:);
f(indicePior,:) = fFilho(filhoVencedor,:);
dominancia(indicePior) = dominanciaFilho(filhoVencedor) - ...
                         ndFilhoJ(indicePior,filhoVencedor);
vizinhanca(indicePior) = vizinhancaFilho(filhoVencedor) - ...
                         dvFilho(filhoVencedor,indicePior);
aptidao(indicePior) = ( 1 + dominancia(indicePior) )* ...
                      ( 1 + vizinhanca(indicePior) );
```

```
      % Avaliação da população

    for ii = 1:np
        if( ii ~= indicePior )
            dominancia(ii) = dominancia(ii) - nd(indicePior,ii) + ...
                             ndFilho(filhoVencedor,ii);
            vizinhanca(ii) = vizinhanca(ii) - dv(ii,indicePior) + ...
                             dvFilho(filhoVencedor,ii);
            aptidao(ii) = ( 1 + dominancia(ii) )* ...
                          ( 1 + vizinhanca(ii) );
            nd(indicePior,ii) = ndFilho(filhoVencedor,ii);
            nd(ii,indicePior) = ndFilhoJ(ii,filhoVencedor);
            dv(ii,indicePior) = dvFilho(filhoVencedor,ii);
            dv(indicePior,ii) = dv(ii,indicePior);
        end
    end
    [ aptidaoTeste, ordem ] = sort( aptidao );
    contGeracoes = contGeracoes + 1;
    termino = contGeracoes >= ng;
end
```

B.5.2 Otimização assistida pela rede neural

```
% NOME DA ROTINA: v20120513_executar_moga

function [probAceite, ...
          probUsoMedio, ...
          avalModSub, ...
          indicePreenchimento] = ...
    v20120513_executar_moga(funcObjSel,nrEnsaios, ...
                            a,camadasInternas,pt,eta,alf,condP, ...
                            tol,np,ng,nEpoc,probUso0, ...
                            x0_estimacao,d0_estimacao, ...
                            x0_validacao,d0_validacao, ...
                            xP_,fP_,xmin,xmax,dmin,dmax, ...
                            modSub,c,rbf,nrCentros, ...
                            lambda,nrTentCentros,fP_cel)

% ENTRADAS

      % funcObjSel:          função objetivo selecionada
      % nrEnsaios:           número de ensaios
      % a:                   parâmetro da função de ativação
      % camadasInternas:     arquitetura da MLP
      % pt:                  percentual de padrões para teste
      % eta:                 taxa de aprendizagem
      % alf:                 momento
      % condP:               condição de parada (limite de épocas)
      % tol:                 tolerância do erro
      % np:                  quantidade de individuos
      % ng:                  número de gerações por época
      % nEpoc:               número de épocas
      % probUso0:            probabilidade de uso da MLP na
      %                      primeira época de
```

```
      %                       otimização
      % x0_estimacao:         amostragem inicial para estimação
      % d0_estimacao:         amostragem inicial para estimação
      % x0_validacao:         amostragem inicial para validação
      % d0_validacao:         amostragem inicial para validação
      % xP_:                  Pareto ideal
      % fP_:                  Pareto ideal
      % xmin:                 limite mínimo do domínio
      % xmax:                 limite máximo do domínio
      % dmin:                 limite mínimo do contradomínio
      % dmax:                 limite máximo do contradomínio
      % modSub:               modelo substituto, se 'MLP' ou 'RBF'
      % c:                    parâmetro da função de base radial
      % rbf:                  função de base radial
      % nrCentros:            número de centros da rbf
      % lambda:               parâmetro de regularização da rbf
      % nrTentCentros:        número de tentativas de posicionamento
      %                       dos centros da rede de função de base
      %                       radial
      % fP_cel:               células da curva de Pareto ideal

% SAÍDAS

      % probAceite:           percentual de soluções dominadas pelos
      %                       pontos que definem a curva de Pareto
      %                       considerada ideal
      % probUsoMedio:         probabilidade média de substituição
      % avalModSub:           quantidade de avaliações do modelo
      %                       original na otimização com o auxílio
      %                       do modelo substituto
      % indicePreenchimento: índice de preenchimento da curva de
      %                       Pareto ideal

fprintf('##############################################\n');
fprintf('# OTIMIZAÇÃO COM O AUXÍLIO DA REDE NEURAL #\n');
fprintf('##############################################\n\n');

% PARTES DO CÓDIGO

      % Rotinas externas
      % Inicializações
      % Ensaios
      % Plotagem

global dataExec;
global contAvaliacoesOriginal;
global contAvaliacoesMLP;
global funcaoObjSelecionada;
global melhor;

% ROTINAS EXTERNAS

global montar_mlp;                              % Subseção B.2.1
global treinar_mlp;                             % Subseção B.2.2
global treinar_rbf;                             % Subseção B.3.1
global moga_binario;                            % Subseção B.5.1
```

```
global DConj;                                   % Subseção B.6.2
global encontrar_solucoes_nao_dominadas;        % Subseção B.6.3
global executar_modelo_original;                % Subseção B.8

% INICIALIZAÇÕES

funcaoObjSelecionada = funcObjSel;
nrAmostras = length(x0_estimacao) + length(x0_validacao);
k1 = 0:1/(nEpoc-1):1;
k2 = k1.^1.585;
k3 = 1*k2 + probUso0*(1 - k2);
probUsoMLPLista = k3;
xq = [];
fq = [];
avaliacoesModeloSubstituto = [];
xPq = [];
fPq = [];
avaliacoesModeloOriginal = [];

% ENSAIOS

fprintf(['Ensaiando... \n']);
fprintf(['--> Função objetivo = ' funcaoObjSelecionada '\n']);
fprintf(['--> Rede neural = ' modSub '\n']);
ng_ = ng;
tempo1_ = now;
for q = 1:nrEnsaios
    ng = ng_;
    fprintf(['--> Ensaio = ' num2str(q) '\n']);
    tempo1 = now;

      % Amostragem

    contAvaliacoesOriginalTot = 0;
    contAvaliacoesOriginal = length(x0_estimacao(:,1)) + ...
                             length(x0_validacao(:,1));
    contAvaliacoesMLP = 0;
    x0 = x0_estimacao;
    d0 = d0_estimacao;
    x0_ = x0;
    d0_ = d0;
    contAvaliacoesOriginalTot = contAvaliacoesOriginalTot + ...
                                contAvaliacoesOriginal;
    fprintf(['--> Avaliações do modelo original =        ' ...
             num2str(contAvaliacoesOriginal) '\n']);
    fprintf(['--> Avaliações da rede neural =            ' ...
             num2str(contAvaliacoesMLP) '\n']);
    contAvaliacoesOriginal = 0;
    contAvaliacoesMLP = 0;
    fprintf(['--> Época de otimização = 1\n']);
    fprintf(['--> Etapa da otimização = Treinamento\n']);
    fprintf(['--> Quantidade de padrões de treinamento = ' ...
             num2str(length(x0)) '\n']);
    if modSub == 'MLP'
        w0 = feval(montar_mlp,length(x0(1,:)), ...
                   [camadasInternas length(d0(1,:))]);
```

```
    [w,err,et,indice] = feval(treinar_mlp,x0,d0,pt,w0, ...
                              a,eta,alf,condP,tol, ...
                              x0_validacao,d0_validacao, ...
                              xmin,xmax,dmin,dmax);
    t = 0;
else
    eqmRBF = -1;
    for k = 1:nrTentCentros
        t_ =  [xmin; lhsu(xmin,xmax, ...   % Subseção B.7
                          nrCentros-2); xmax];
        [wRBF_,eqmRBF_] = feval(treinar_rbf,x0,d0, ...
                                c,rbf,t_,lambda, ...
                                xmin,xmax,dmin,dmax,pt, ...
                                x0_validacao,d0_validacao);
        if eqmRBF < 0 | eqmRBF > eqmRBF_
            eqmRBF = eqmRBF_;
            t =      t_;
            w =      wRBF_;
        end
    end
end
fprintf(['--> Etapa da otimização = MOGA + ' modSub '\n']);
probUsoMLP = probUsoMLPLista(1);
[p,x,f,xPopIni,fPopIni,xOriginal,fOriginal, ...
 contAvaliacoes, ...
 contDuplicados, ...
 contInvalidos] = feval(moga_binario,np,ng,0,true,w,a, ...
                        xmin,xmax, ...
                        dmin,dmax,probUsoMLP, ...
                        modSub,c,rbf,t);
xPopIni_ = xPopIni;
fPopIni_ = fPopIni;
x0 = [x0; xOriginal];
d0 = [d0; fOriginal];
contAvaliacoesOriginalTot = contAvaliacoesOriginalTot + ...
                            contAvaliacoesOriginal;
fprintf(['--> Avaliações do modelo original =        ' ...
         num2str(contAvaliacoesOriginal) '\n']);
fprintf(['--> Avaliações da rede neural =            ' ...
         num2str(contAvaliacoesMLP) '\n']);
fprintf(['--> Uso percentual do modelo original =    ' ...
         num2str(contAvaliacoesOriginal*100/ ...
                 contAvaliacoes) '\n']);
for i = 2:nEpoc
    fprintf(['--> Época de otimização = ' num2str(i) '\n']);
    contAvaliacoesOriginal = 0;
    contAvaliacoesMLP = 0;
    fprintf(['--> Etapa da otimização = Treinamento\n']);
    fprintf(['--> Quantidade de padrões de treinamento = ' ...
             num2str(length(x0)) '\n']);
    if modSub == 'MLP'
        [w,err,et,indice] = feval(treinar_mlp,x0,d0,pt,w0, ...
                                  a,eta,alf,condP,tol, ...
                                  x0_validacao,d0_validacao, ...
                                  xmin,xmax,dmin,dmax);
        t = 0;
```

```
    else
        eqmRBF = -1;
        for k = 1:nrTentCentros
            t_ =  [xmin; lhsu(xmin,xmax, ...     % Subseção B.7
                              nrCentros-2); xmax];
            [wRBF_,eqmRBF_] = feval(treinar_rbf,x0,d0, ...
                                    c,rbf,t_,lambda, ...
                                    xmin,xmax,dmin,dmax,pt, ...
                                    x0_validacao,d0_validacao);
            if eqmRBF < 0 | eqmRBF > eqmRBF_
                eqmRBF = eqmRBF_;
                t =      t_;
                w =      wRBF_;
            end
        end
    end
    fprintf(['--> Etapa da otimização = MOGA + ' modSub '\n']);
    probUsoMLP = probUsoMLPLista(i);
    [p,x,f,xPopIni,fPopIni,xOriginal,fOriginal, ...
     contAvaliacoes, ...
     contDuplicados, ...
     contInvalidos] = feval(moga_binario,np,ng,p,false,w,a, ...
                            xmin,xmax, ...
                            dmin,dmax,probUsoMLP, ...
                            modSub,c,rbf,t);
    if i < nEpoc
        x0 = [x0; xOriginal];
        d0 = [d0; fOriginal];
    end
    contAvaliacoesOriginalTot = contAvaliacoesOriginalTot + ...
                                contAvaliacoesOriginal;
    fprintf(['--> Avaliações do modelo original =        ' ...
             num2str(contAvaliacoesOriginal) '\n']);
    fprintf(['--> Avaliações da rede neural =            ' ...
             num2str(contAvaliacoesMLP) '\n']);
    fprintf(['--> Uso percentual do modelo original =    ' ...
             num2str(contAvaliacoesOriginal*100/ ...
                     contAvaliacoes) '\n']);
end

  % Soluções de Pareto avaliadas com o modelo original

fprintf(['--> Etapa da otimização = Avaliação\n']);
contAvaliacoesOriginal = 0;
contAvaliacoesMLP = 0;
for i = 1:length(x(:,1))
    f(i,:) = feval(executar_modelo_original,x(i,:),'f');
end
contAvaliacoesOriginalTot = contAvaliacoesOriginalTot + ...
                            contAvaliacoesOriginal;
fprintf(['--> Avaliações do modelo original =        ' ...
         num2str(contAvaliacoesOriginal) '\n']);
fprintf(['--> Avaliações totais do modelo original = ' ...
         num2str(contAvaliacoesOriginalTot) '\n']);
fprintf(['--> Avaliações da rede neural =            ' ...
         num2str(contAvaliacoesMLP) '\n']);
```

```
    xq = [xq; x];
    fq = [fq; f];
    avaliacoesModeloSubstituto = [avaliacoesModeloSubstituto; ...
                                  contAvaliacoesOriginalTot];

      % Pareto

    fprintf(['--> Etapa da otimização = MOGA - ' modSub '\n']);
    contAvaliacoesOriginal = 0;
    contAvaliacoesMLP = 0;
    ng = ceil((contAvaliacoesOriginalTot - np)/2);
    fprintf(['--> Número de gerações = ' num2str(ng) '\n']);
    [pP,xP,fP,xPopIniP,fPopIniP,xOriginalP,fOriginalP, ...
     contAvaliacoesP, ...
     contDuplicadosP, ...
     contInvalidosP] = feval(moga_binario,np,ng,0,true,0,0, ...
                             xmin,xmax, ...
                             dmin,dmax,0, ...
                             '',0,'',0);
    fprintf(['--> Avaliações do modelo original = ' ...
             num2str(contAvaliacoesOriginal) '\n']);
    fprintf(['--> Avaliações da rede neural = ' ...
             num2str(contAvaliacoesMLP) '\n']);
    xPq = [xPq; xP];
    fPq = [fPq; fP];
    avaliacoesModeloOriginal = [avaliacoesModeloOriginal; ...
                                contAvaliacoesOriginal];
    tempo2 = now;
    tempo = 24*60*60*(tempo2-tempo1);
    fprintf(['--> Duração do ensaio (seg) = ' num2str(tempo) '\n']);
end
tempo2_ = now;
tempo_ = 24*60*60*(tempo2_-tempo1_);
[ndPq,ndqP] =   feval(DConj,fP_,fq,melhor);
[ndPPq,ndPqP] = feval(DConj,fP_,fPq,melhor);
[ind, id] = feval(encontrar_solucoes_nao_dominadas,fq,melhor);
fqPareto = fq(ind,:);
[ndParetoCell, ndCellPareto] = feval(DConj,fqPareto,fP_cel,melhor);
probAceite =          ndPq/length(fq); % Quantos do conjunto fq,
                                       % otimização assistida,
                                       % sao dominados por elementos
                                       % do conjunto fP_, Pareto
                                       % ideal, divido pelo total de
                                       % elementos de fq
probUsoMedio =        mean(probUsoMLPLista);
avalModSub =          mean(avaliacoesModeloSubstituto);
indicePreenchimento = ndParetoCell/length(fP_cel);
fprintf(['--> Duração total da ' modSub ' (seg) = ' ...
         num2str(tempo_) '\n']);
fprintf('Concluído.\n\n');

% PLOTAGEM

fprintf(['Plotando... \n']);
titulo = [modSub '- ' funcObjSel '- Preenchimento = ' ...
          num2str(indicePreenchimento)];
```

```
hFig = figure;
plot(xq(:,1)',xq(:,2)','ok', ...
     xPq(:,1)',xPq(:,2)','xr', ...
     xP_(:,1)',xP_(:,2)','.b');
title(titulo);
xlabel('x1','FontSize',12,'FontName','Arial','FontAngle','italic');
ylabel('x2','FontSize',12,'FontName','Arial','FontAngle','italic');
set(gca,'FontSize',10,'FontName','Arial');
saveas(hFig,['Registros\v20120513_executar_moga- ' ...
             dataExec '- ' modSub '- ' funcObjSel ...
             '- Domínio.emf'],'emf');
hFig = figure;
plot(fq(:,1)',fq(:,2)','ok', ...
     fPq(:,1)',fPq(:,2)','xr', ...
     fP_(:,1)',fP_(:,2)','.b');
title(titulo);
xlabel('f1','FontSize',12,'FontName','Arial','FontAngle','italic');
ylabel('f2','FontSize',12,'FontName','Arial','FontAngle','italic');
h = legend(['MOGA + ' modSub ' -> ' num2str(avalModSub) '/ ' ...
           num2str(probAceite)], ['MOGA - ' modSub ' -> ' ...
           num2str(mean(avaliacoesModeloOriginal)) '/ ' ...
           num2str(ndPPq/length(fPq))], ...
           'Curva de Pareto',1);
set(h,'FontSize',10,'FontName','Arial');
set(gca,'FontSize',10,'FontName','Arial');
saveas(hFig,['Registros\v20120513_executar_moga- ' dataExec '- ' ...
             modSub '- ' funcObjSel '- Contradomínio.emf'],'emf');
fprintf('Concluído.\n\n');
fprintf('Fim da rotina.\n\n');
```

B.5.3 Função objetivo

```
% NOME DA ROTINA: v20110711_func_obj

function [x,f,v,flagOriginal] = ...
    v20110711_func_obj(individuo,flagExec, ...
                       flagBinario,w,a, ...
                       xmin,xmax,dmin,dmax, ...
                       probUsoModSub, ...
                       modSub,c,rbf,t)

% ENTRADAS

      % individuo:                 indivíduo na representação binária
      % flagExec:                  se false somente retorna valores nas
      %                            variáveis globais
      % flagBinario:               se TRUE converte de binário para
      %                            real
      % w:                         pesos sinápticos da MLP
      % a:                         parâmetro da função de ativação
      % xmin:                      limite inferior do domínio
      % xmax:                      limite superior do domínio
      % dmin:                      limite inferior do contradomínio
      % dmax:                      limite superior do contradomínio
```

```
      % probUsoModSub:            probabilidade de uso da MLP
      % modSub:                   modelo substituto, se 'MLP' ou 'RBF'
      % c:                        parâmetro da função de base radial
      % rbf:                      função de base radial
      % t:                        centros da função de base radial

% SAÍDAS

      % x:                        vetor das variáveis objetivo em real
      % f:                        vetor das funções objetivo
      % v:                        validade do indivíduo em função das
      %                           restrições do problema
      % flagOriginal:             se 1 significa que o modelo original
      %                           foi usado

global nv;        % Número de variáveis objetivo
global xInf;      % Limite inferior da variável objetivo
global xSup;      % Limite superior da variável objetivo
global nobj;      % Número de funcoes objetivo
global nbv;       % Número de bits para representar uma variável
global melhor;    % Máximo/ mínimo
global nf;        % Número de soluções visitadas no M-GEO com o ORAD
global ativar_mlp;                  % Subseção B.2.3
global ativar_rbf;                  % Subseção B.3.3
global executar_modelo_original;    % Subseção B.8

flagOriginal = 0;
if( flagExec == true )
    x = zeros(1,nv);
    f = zeros(1,nobj);
    if( flagBinario == true )
        maxInteiro = 2^nbv - 1;
        for ii = 1:nv
            expoente = nbv - 1;
            for jj = (ii-1)*nbv+1 : ii*nbv
                if( individuo(jj) == 1 )
                    x(ii) = x(ii) + 2^expoente;
                end
                expoente = expoente - 1;
            end
            x(ii) = (xSup(ii) - xInf(ii))*x(ii)/maxInteiro + xInf(ii);
        end
    else
        for ii = 1:nv
            x(ii) = (xSup(ii) - xInf(ii))*individuo(ii) + xInf(ii);
        end
    end
end
if( flagExec == false )
    nv =     length(xmin);
    xInf =   xmin;
    xSup =   xmax;
    nobj =   length(dmin);
else
    v = feval(executar_modelo_original,x,'v');
    if( probUsoModSub >= rand(1) )
```

```
        if modSub == 'MLP'
            saida = feval(ativar_mlp,x',w,a,xmin,xmax,dmin,dmax);
            f = saida{length(saida)}';
        else
            f = feval(ativar_rbf,x,w,c,rbf,t,xmin,xmax,dmin,dmax);
        end
    else
        f = feval(executar_modelo_original,x,'f');
        flagOriginal = 1;
    end
end
```

B.6 Rotinas de avaliação

B.6.1 Estimativa da eficiência da rede neural

```
% NOME DA ROTINA: v20120519_calcular_eficiencia

function eta = ...
    v20120519_calcular_eficiencia(funcObjSel,probAceite, ...
                                  nrEnsaios,nrAmostras, ...
                                  P,passoNg,np,ng,nEpoc, ...
                                  xmin,xmax, ...
                                  dmin,dmax, ...
                                  xP_,fP_)

% ENTRADAS

      % funcObjSel: função objetivo selecionada
      % probAceite: percentual de soluções dominadas pelos pontos que
      %             definem a curva de Pareto considerada ideal
      % nrEnsaios:  número de ensaios
      % nrAmostras: número total de exemplos
      % P:          probabilidade média de substituição
      % passoNg:    incremento do número de gerações
      % np:         quantidade de individuos
      % ng:         número de gerações por época de otimização
      % nEpoc:      número de épocas de otimização
      % xmin:       limite mínimo do domínio
      % xmax:       limite máximo do domínio
      % dmin:       limite mínimo do contradomínio
      % dmax:       limite máximo do contradomínio
      % xP_:        Pareto ideal
      % fP_:        Pareto ideal

% SAÍDAS

      % eta:        eficiência

fprintf('#########################################\n');
fprintf('# CÁLCULO DA EFICIÊNCIA DA REDE NEURAL #\n');
fprintf('#########################################\n\n');
```

```
% PARTES DO CÓDIGO

      % Rotinas externas
      % Cálculo da eficiência

global funcaoObjSelecionada;
global melhor;

% ROTINAS EXTERNAS

global moga_binario;    % Subseção B.5.1
global DConj;           % Subseção B.6.2

% CÁLCULO DA EFICIÊNCIA

fprintf(['Calculando a eficiência... \n']);
funcaoObjSelecionada = funcObjSel;
ngTot = nEpoc*ng;
np_ = np;
ng_ensaios = [];
for q = 1:nrEnsaios
    fprintf(['--> Ensaio = ' num2str(q) '\n']);
    ng_ = [];
    for i = 1:length(np_)
        np = np_(i);
        ng = 0;
        ndP_P = probAceite*np + 1;
        while ndP_P > probAceite*np
            ng = ng + passoNg;
            [pP,xP,fP,xPopIniP,fPopIniP,xOriginalP,fOriginalP, ...
             contAvaliacoesP, ...
             contDuplicadosP, ...
             contInvalidosP] = feval(moga_binario,np,ng,0, ...
                                     true,0,0, ...
                                     xmin,xmax,dmin,dmax, ...
                                     0,'',0,'',0);
            [ndP_P,ndqPP_] = feval(DConj,fP_,fP,melhor);
        end
        ng_ = [ ng_ ng ];
    end
    ng_ensaios = [ng_ensaios; ng_];
end
if nrEnsaios > 1
    q2 = mean(ng_ensaios);
    U =  max(ng_ensaios-ones(length(ng_ensaios(:,1)),1)*q2);
    L =  abs(min(ng_ensaios-ones(length(ng_ensaios(:,1)),1)*q2));
else
    q2 = ng_ensaios;
    U =  zeros(1,length(ng_ensaios));
    L =  U;
end
q1 =  (1-P)*nEpoc*np_/2 + (1-P)*ngTot + nrAmostras/2;
eta = (q2 - q1)/q2;
fprintf('Concluído.\n\n');
fprintf('Fim da rotina.\n\n');
```

B.6.2 Avaliação da dominância entre conjuntos de soluções de Pareto

```
% NOME DA ROTINA: v20110711_avaliar_dominancia_conjuntos

function [ndij,ndji] = ...
    v20110711_avaliar_dominancia_conjuntos(fi,fj,melhor)

% ENTRADAS

      % fi:     primeiro conjunto
      % fj:     segundo conjunto
      % melhor: 0 se minimização e 1 se maximização

% SAÍDAS

      % ndij:   dominancia do primeiro conjunto sobre o segundo
      % ndji:   dominancia do segundo conjunto sobre o primeiro

npi = length(fi(:,1));
npj = length(fj(:,1));
nobj = length(fi(1,:));
ndi = zeros(npi,npj);
ndj = zeros(npj,npi);
for ii = 1:npi
    for jj = 1:npj
        iDomina = 0;
        jDomina = 0;
        for kk = 1:nobj
            if( melhor(kk) == 0 )
                if( fi(ii,kk) < fj(jj,kk) )
                    iDomina = iDomina + 1;
                elseif( fj(jj,kk) < fi(ii,kk) )
                    jDomina = jDomina + 1;
                end
            else
                if( fi(ii,kk) > fj(jj,kk) )
                    iDomina = iDomina + 1;
                elseif( fj(jj,kk) > fi(ii,kk) )
                    jDomina = jDomina + 1;
                end
            end
        end
        ndi(ii,jj) = 0;
        ndj(jj,ii) = 0;
        if( iDomina > 0 & jDomina == 0 )
            ndi(ii,jj) = 1;
        elseif( iDomina == 0 & jDomina > 0 )
            ndj(jj,ii) = 1;
        end
    end
end
ndij = 0;
ndji = 0;
```

```
% QUANTOS DO CONJUNTO j SAO DOMINADOS POR ELEMENTOS DO CONJUNTO i

for jj = 1:npj
    solucaoDominada = false;
    for ii = 1:npi
        if( ndi(ii,jj) == 1 )
            solucaoDominada = true;
            break;
        end
    end
    if( solucaoDominada == true )
        ndij = ndij + 1;
    end
end

% QUANTOS DO CONJUNTO i SAO DOMINADOS POR ELEMENTOS DO CONJUNTO j

for ii = 1:npi
    solucaoDominada = false;
    for jj = 1:npj
        if( ndj(jj,ii) == 1 )
            solucaoDominada = true;
            break;
        end
    end
    if( solucaoDominada == true )
        ndji = ndji + 1;
    end
end
```

B.6.3 Obtenção das soluções não-dominadas de um conjunto de soluções

```
% NOME DA ROTINA: v20120513_encontrar_solucoes_nao_dominadas

function [ind, id] = ...
    v20120513_encontrar_solucoes_nao_dominadas(f,melhor)

% ENTRADA

      % f:      soluções
      %         Linha:  solução
      %         Coluna: objetivo
      % melhor: vetor
      %         Componente: 0 se for minimização e
      %                     1 se for maximização do objetivo

% SAÍDA

      % ind:    índice das soluções não dominadas
      % id:     índide das soluções dominadas

np =   length(f(:,1));
nobj = length(f(1,:));
nd = zeros(np);
```

```
dominancia = zeros(1,np);
for ii = 1:np-1
    for jj = ii+1:np
        iDomina = 0;
        jDomina = 0;
        for kk = 1:nobj
            if( melhor(kk) == 0 )
                if( f(ii,kk) < f(jj,kk) )
                    iDomina = iDomina + 1;
                elseif( f(jj,kk) < f(ii,kk) )
                    jDomina = jDomina + 1;
                end
            else
                if( f(ii,kk) > f(jj,kk) )
                    iDomina = iDomina + 1;
                elseif( f(jj,kk) > f(ii,kk) )
                    jDomina = jDomina + 1;
                end
            end
        end
        nd(ii,jj) = 0;
        nd(jj,ii) = 0;
        if( iDomina > 0 & jDomina == 0 )
            nd(ii,jj) = 1;
        elseif( iDomina == 0 & jDomina > 0 )
            nd(jj,ii) = 1;
        end
    end
end
for ii = 1:np
    for jj = 1:np
        dominancia(ii) = dominancia(ii) + nd(jj,ii);
    end
end
ind = find(dominancia == 0);
id =  find(dominancia >  0);
```

B.7 Rotina do método *latin hypercube sampling*

```
% NOME DA ROTINA: lhsu

% AUTOR: Budiman Minasny (2003)

function s = lhsu(xmin,xmax,nsample)

% ENTRADA

      % xmin:    limite inferior de cada variável (vetor linha)
      % xmax:    limite superior de cada variável (vetor linha)
      % nsample: número de amostras a serem geradas

% SAÍDA

      % s:       amostras
```

```
nvar = length(xmin);
ran =  rand(nsample,nvar);
s =    zeros(nsample,nvar);
for j= 1:nvar
    idx =    randperm(nsample);
    P =      (idx'-ran(:,j))/nsample;
    s(:,j) = xmin(j) + P.*(xmax(j)-xmin(j));
end
```

B.8 Funções de teste

```
% NOME DA ROTINA: v20110711_executar_modelo_original

function s = v20110711_executar_modelo_original(x,retorno)

% ENTRADAS

      % x:            entrada da função (vetor linha)
      % retorno:      'f':    avaliação da função objetivo
      %               'v':    validade da entrada x
      %               'xInf': limite inferior de x
      %               'xSup': limite superior de x

% SAÍDAS

      % s:            o que foi especificado como retorno
      %               f:            saída da função (vetor linha)
      %               v:            validade da entrada (se TRUE é
      %                             válido e se FALSE não é válido)
      %               xInf e xSup:  limites do espaço de busca

global contAvaliacoesOriginal;
global funcaoObjSelecionada;

switch funcaoObjSelecionada
    case 'Srinivas1994'
        switch retorno
            case 'f'
                s(1) = ( x(1) - 2 )^2 + ( x(2) - 1 )^2 + 2;
                s(2) = 9*x(1) - ( x(2) - 1 )^2;
                contAvaliacoesOriginal = contAvaliacoesOriginal + 1;
            case 'v'
                s = true;
            case 'xInf'
                s(1) = -20;
                s(2) = -20;
            case 'xSup'
                s(1) = 20;
                s(2) = 20;
        end
    case 'Deb2000'
        switch retorno
            case 'f'
```

```
                A1 =   0.5*sin(1)    - 2*cos(1) + ...
                       sin(2)        - 1.5*cos(2);
                A2 =   1.5*sin(1)    - cos(1) + ...
                       2*sin(2)      - 0.5*cos(2);
                B1 =   0.5*sin(x(1)) - 2*cos(x(1)) + ...
                       sin(x(2))     - 1.5*cos(x(2));
                B2 =   1.5*sin(x(1)) -   cos(x(1)) + ...
                       2*sin(x(2))   - 0.5*cos(x(2));
                s(1) = 1 + ( A1 - B1 )^2 + ( A2 - B2 )^2;
                s(2) = ( x(1) + 3 )^2    + ( x(2) + 1 )^2;
                contAvaliacoesOriginal = contAvaliacoesOriginal + 1;
            case 'v'
                s = true;
            case 'xInf'
                s(1) = -pi;
                s(2) = -pi;
            case 'xSup'
                s(1) = pi;
                s(2) = pi;
        end
    case 'Huband2006'
        switch retorno
            case 'f'
                s(1) = -2*exp( 15*(-(x(1)-.1)^2 - x(2)^2) ) - ...
                          exp( 20*(-(x(1)-.6)^2 - (x(2)-.6)^2) ) + ...
                          exp( 20*(-(x(1)+.6)^2 - (x(2)-.6)^2) ) + ...
                          exp( 20*(-(x(1)-.6)^2 - (x(2)+.6)^2) ) + ...
                          exp( 20*(-(x(1)+.6)^2 - (x(2)+.6)^2) );
                s(2) =  2*exp( 20*(-x(1)^2      - x(2)^2) ) + ...
                          exp( 20*(-(x(1)-.4)^2 - (x(2)-.6)^2) ) - ...
                          exp( 20*(-(x(1)+.5)^2 - (x(2)-.7)^2) ) - ...
                          exp( 20*(-(x(1)-.5)^2 - (x(2)+.7)^2) ) + ...
                          exp( 20*(-(x(1)+.4)^2 - (x(2)+.8)^2) );
                contAvaliacoesOriginal = contAvaliacoesOriginal + 1;
            case 'v'
                s = true;
            case 'xInf'
                s(1) = -1;
                s(2) = -1;
            case 'xSup'
                s(1) =  1;
                s(2) =  1;
        end
end
```

APÊNDICE C – TELAS DA SUMARIZAÇÃO DE RESULTADOS

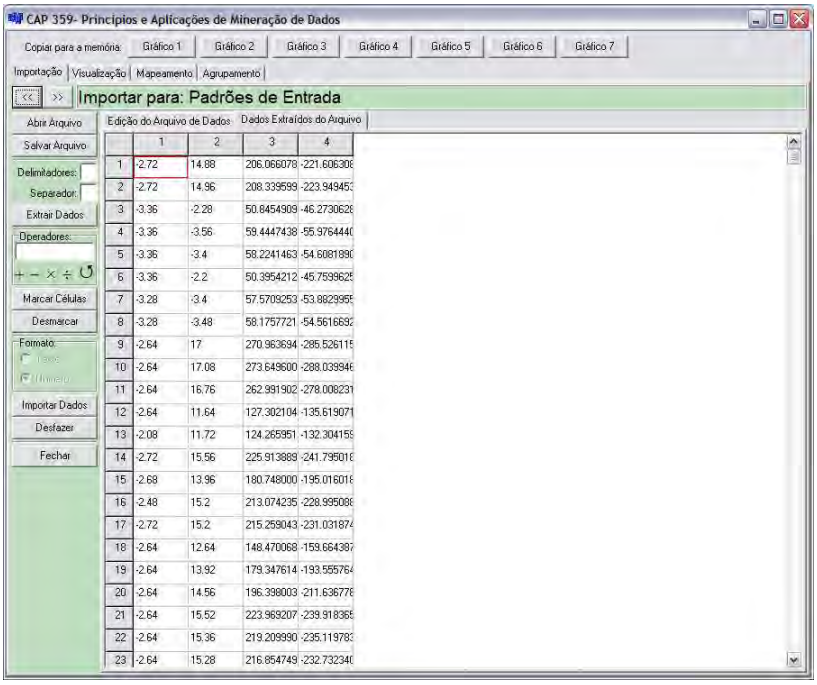


Figura C.1 - Importação dos dados de otimização.

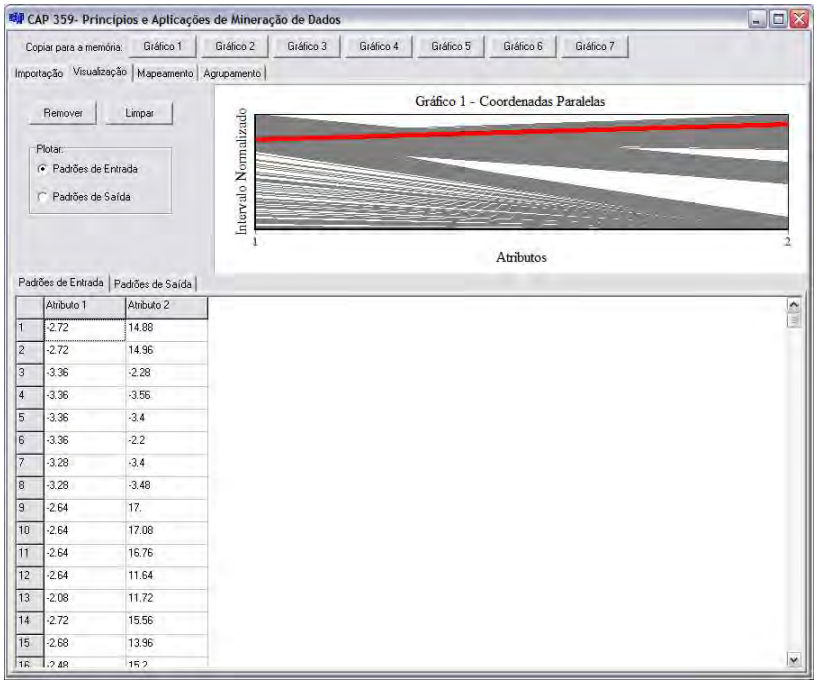


Figura C.2 - Visualização de dados por meio de coordenadas paralelas.

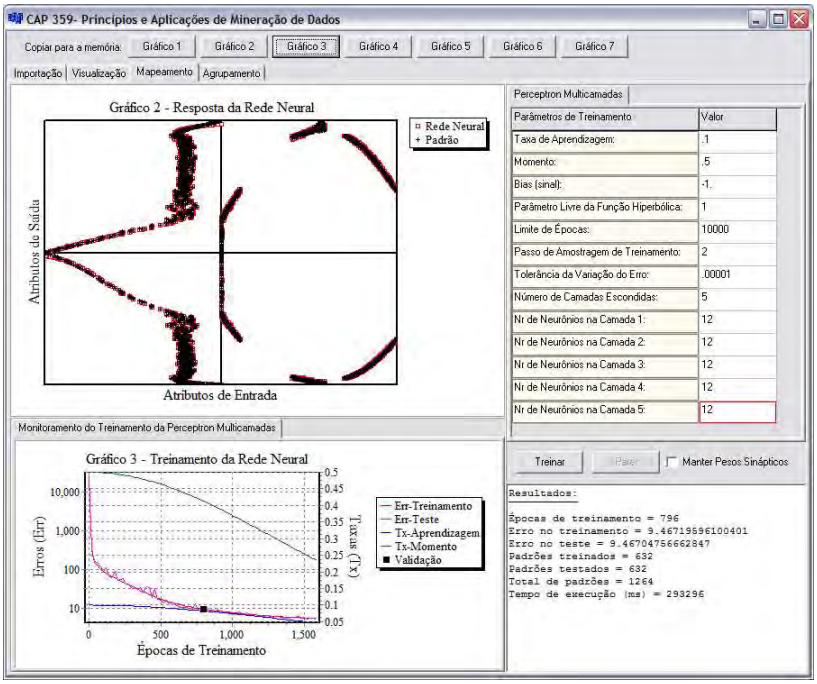


Figura C.3 - Regressão dos dados através da rede perceptron de múltiplas camadas.

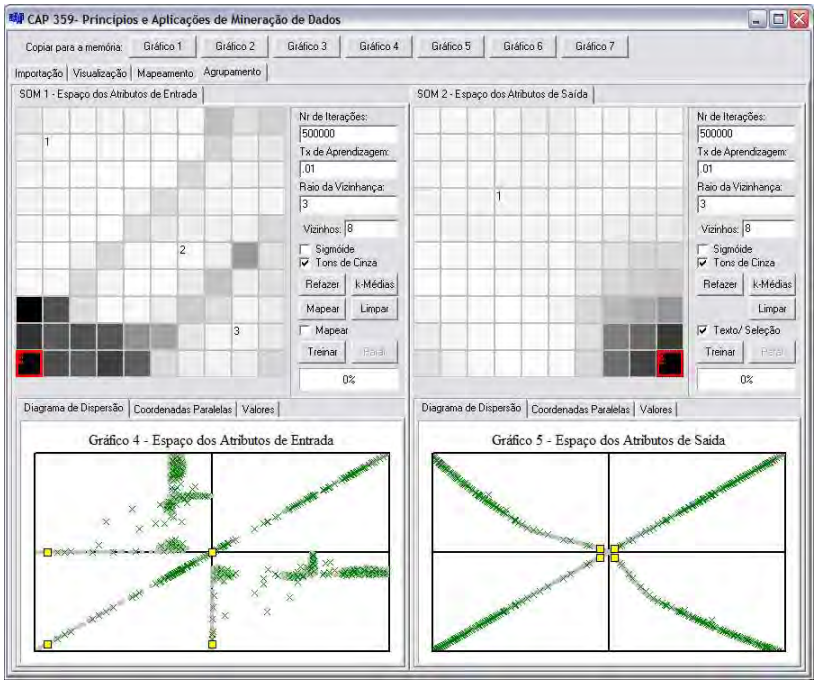


Figura C.4 - Identificação dos agrupamentos por meio da visualização dos reticulados em tons de cinza.

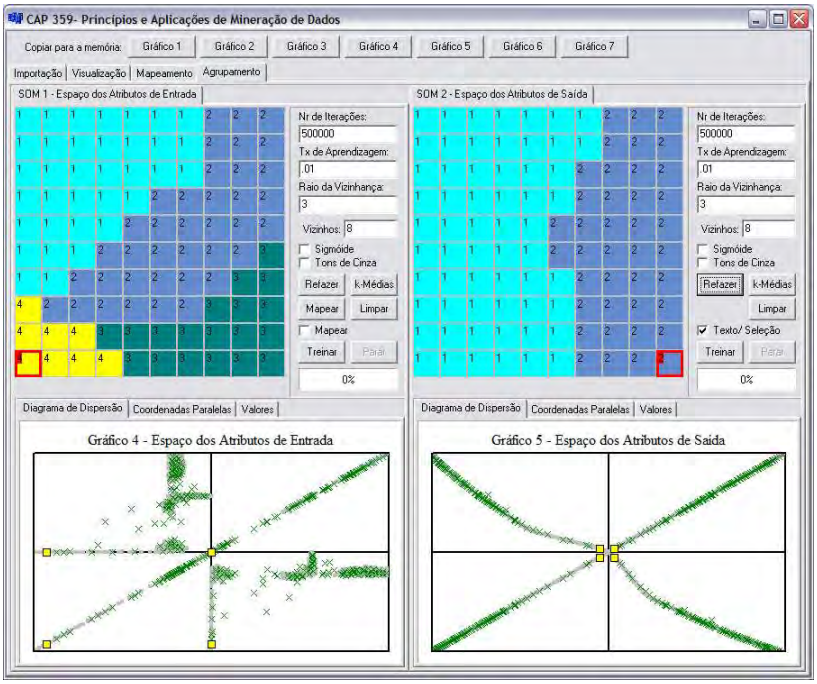


Figura C.5 - Agrupamento dos neurônios por meio do k-médias.

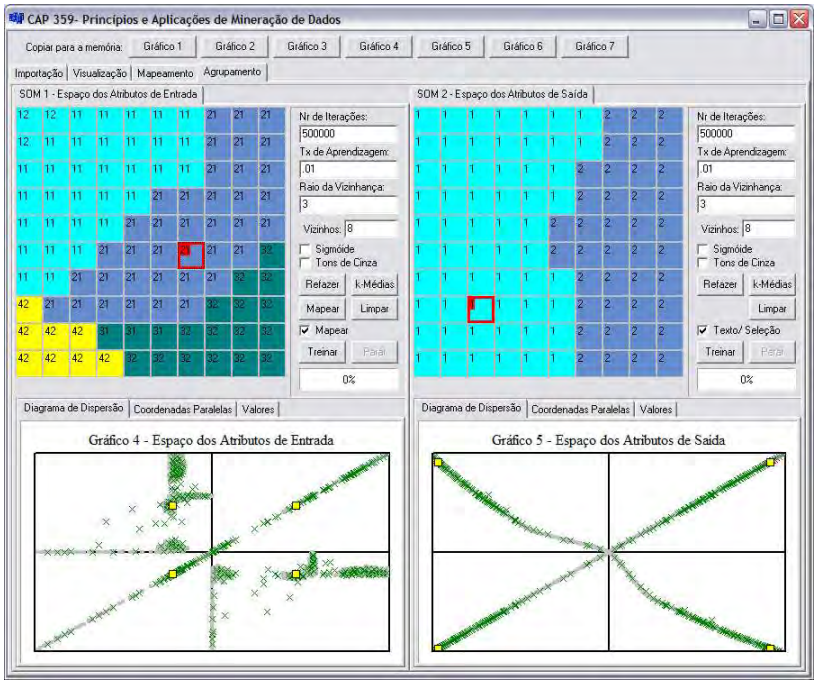


Figura C.6 - Mapeamento dos neurônios do SOM 1 sobre os neurônios do SOM 2 através da rede perceptron de múltiplas camadas.

APÊNDICE D – TELAS DO PROGRAMA MDO-SONDA

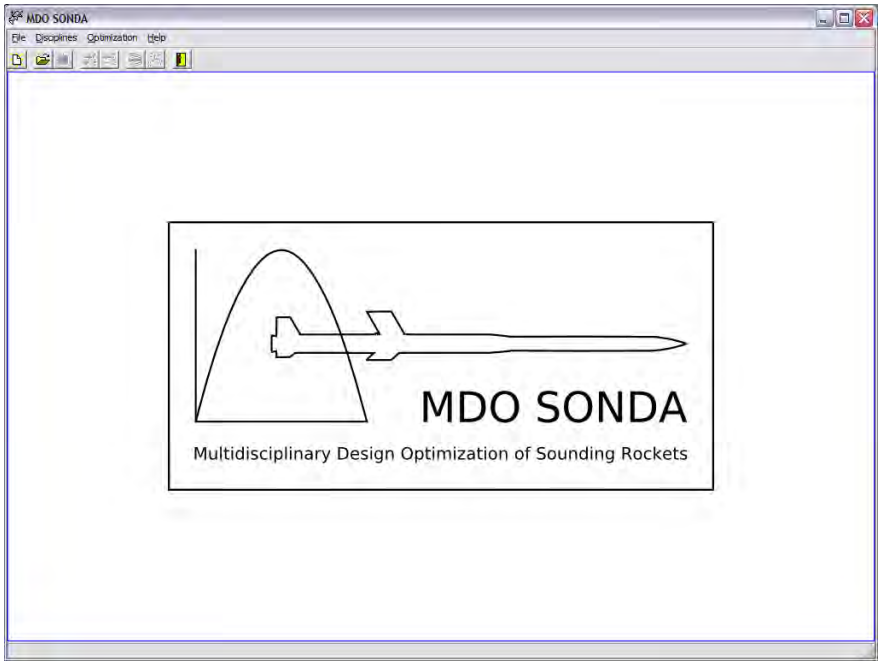


Figura D.1 - Tela de abertura do programa.

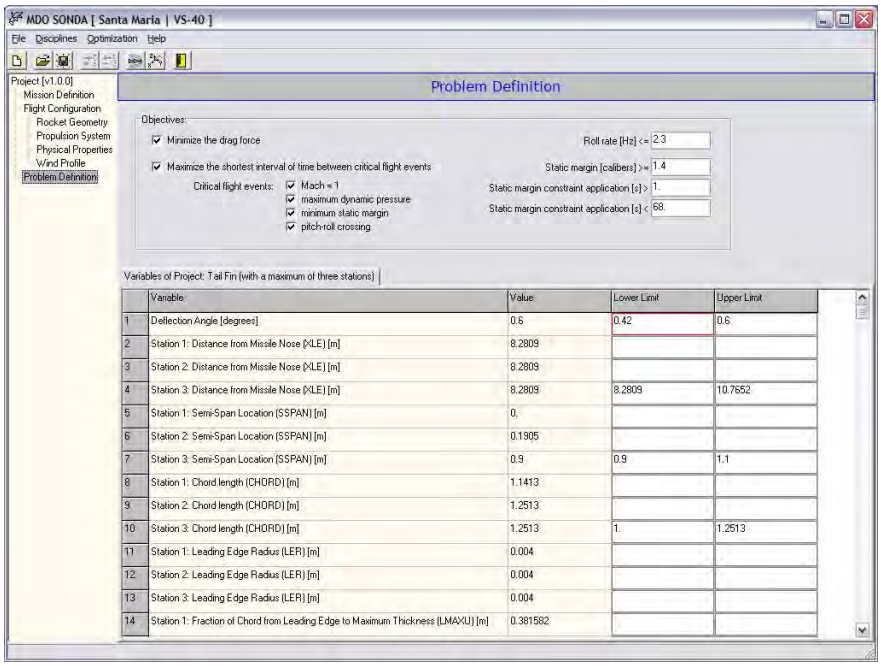


Figura D.2 - Entrada das definições do problema de otimização de empenas.

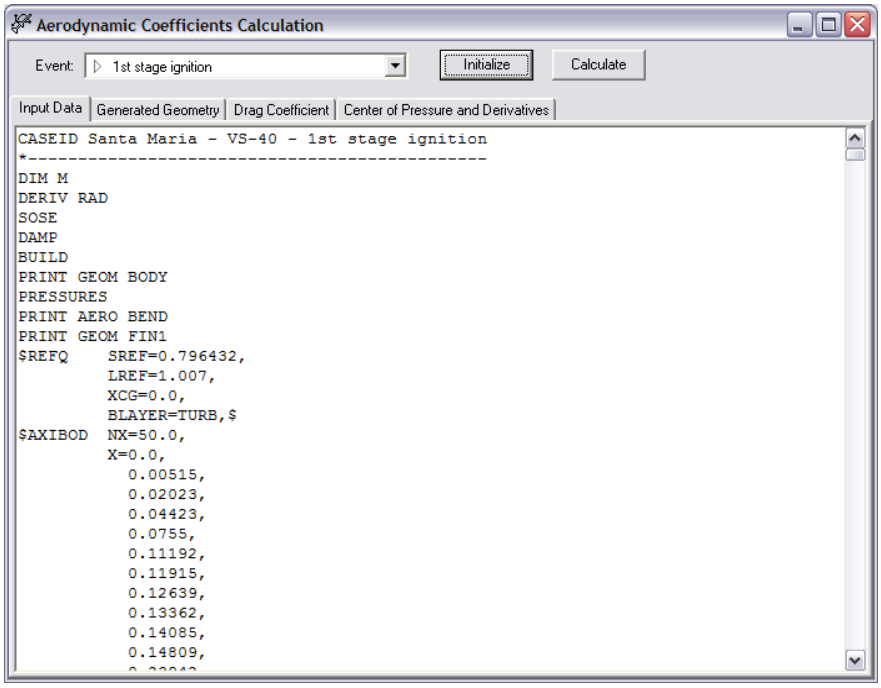


Figura D.3 - Interface para visualizar o arquivo de entrada do programa *missile datcom* para cada fase de voo.

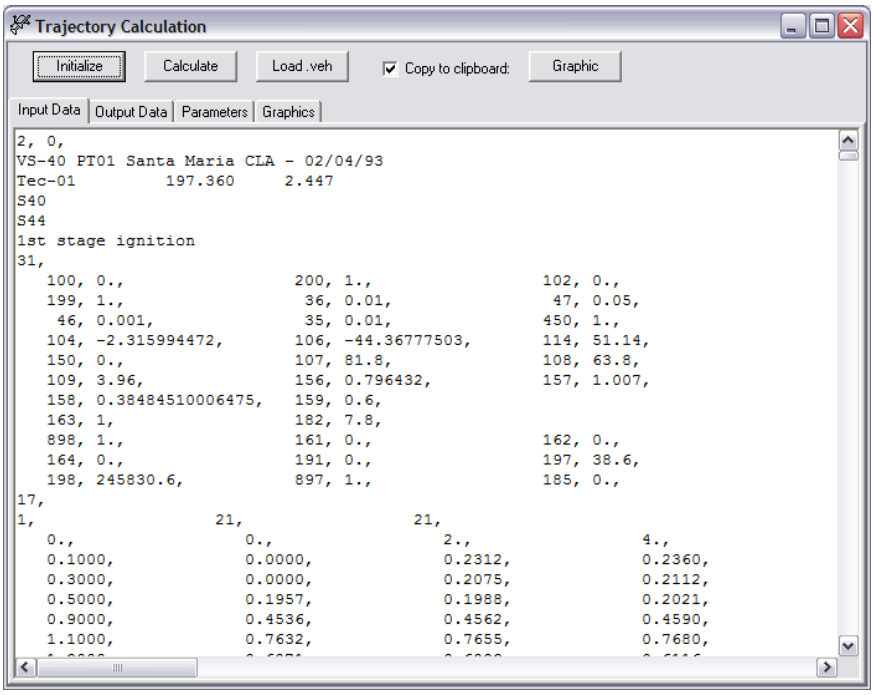


Figura D.4 - Interface para visualizar o arquivo de entrada do programa ROSI.

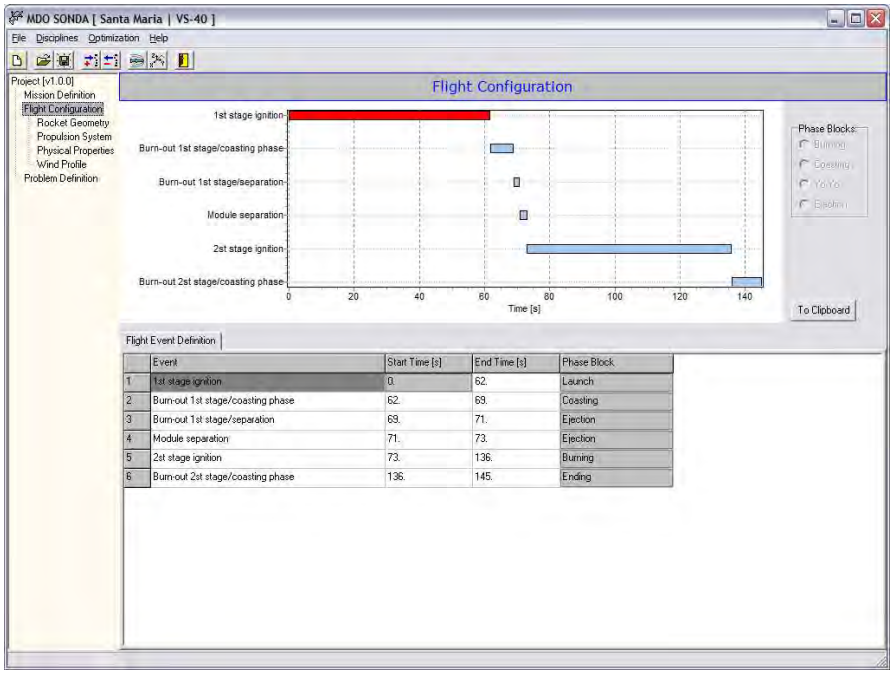


Figura D.5 - Interface para definir as fases de voo.

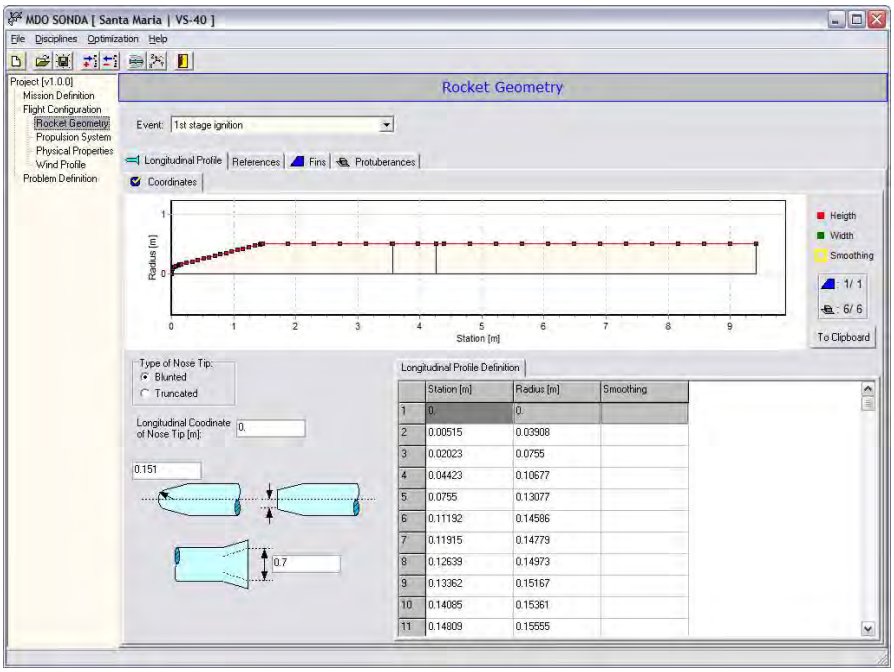


Figura D.6 - Entrada do perfil longitudinal da geometria externa do foguete.

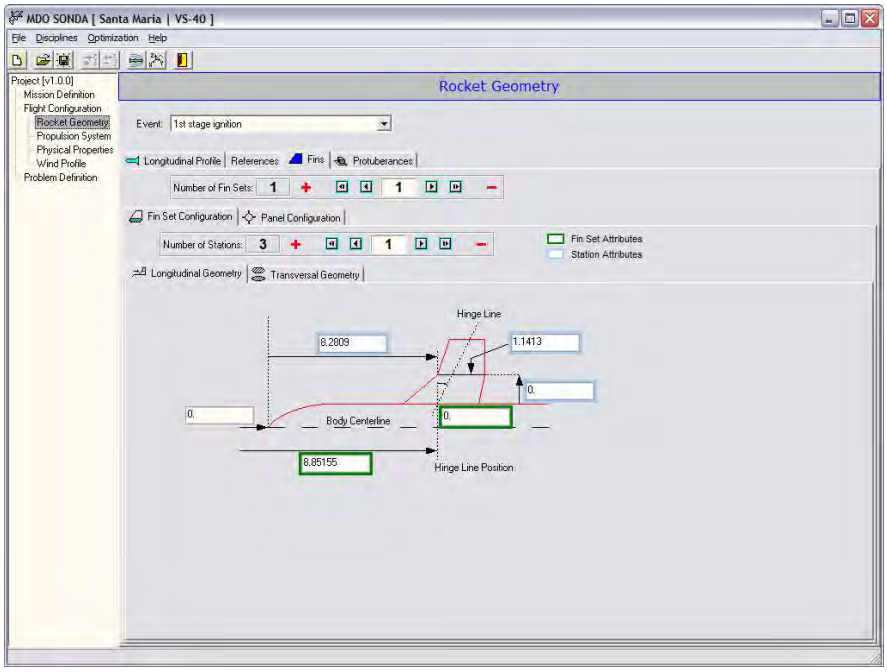


Figura D.7 - Entrada do perfil longitudinal das empenas.

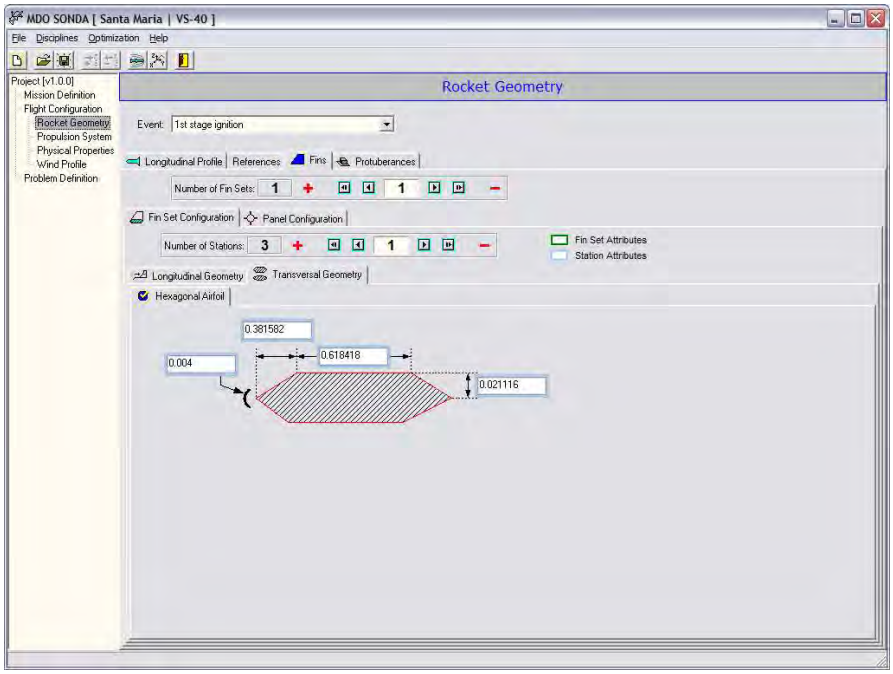


Figura D.8 - Entrada da seção transversal das empenas.

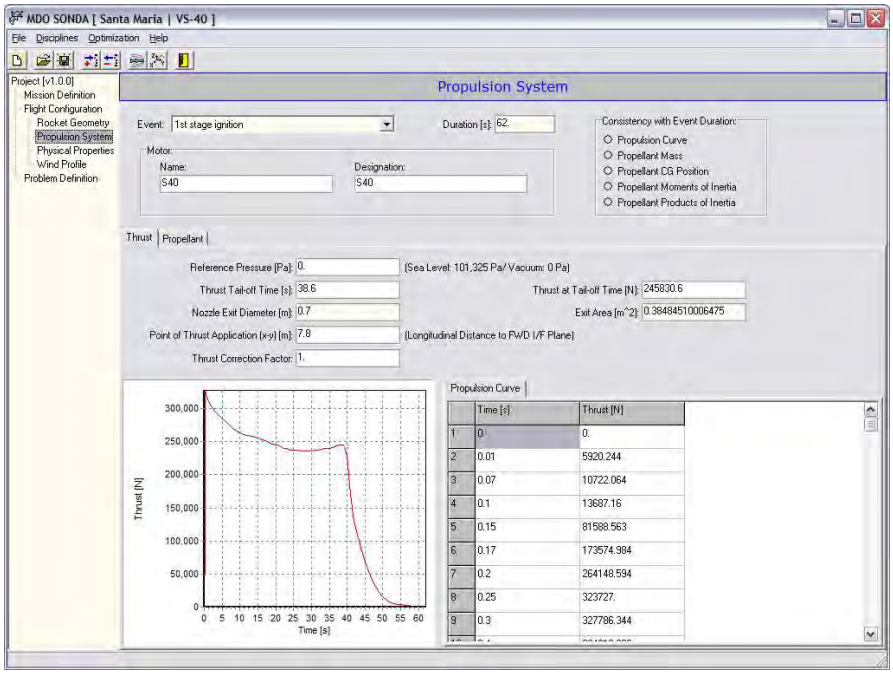


Figura D.9 - Entrada dos dados propulsivos do foguete.

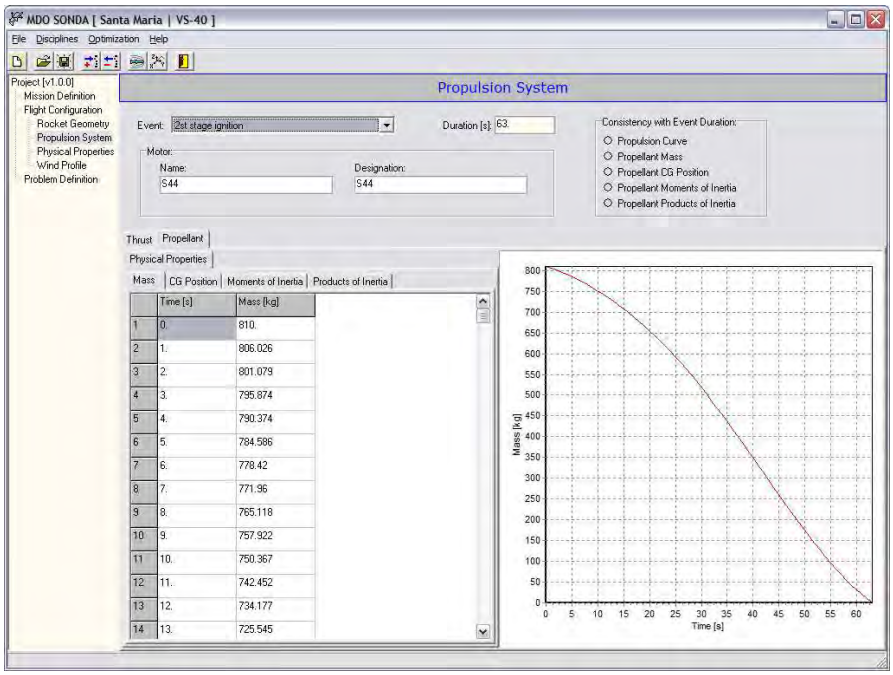
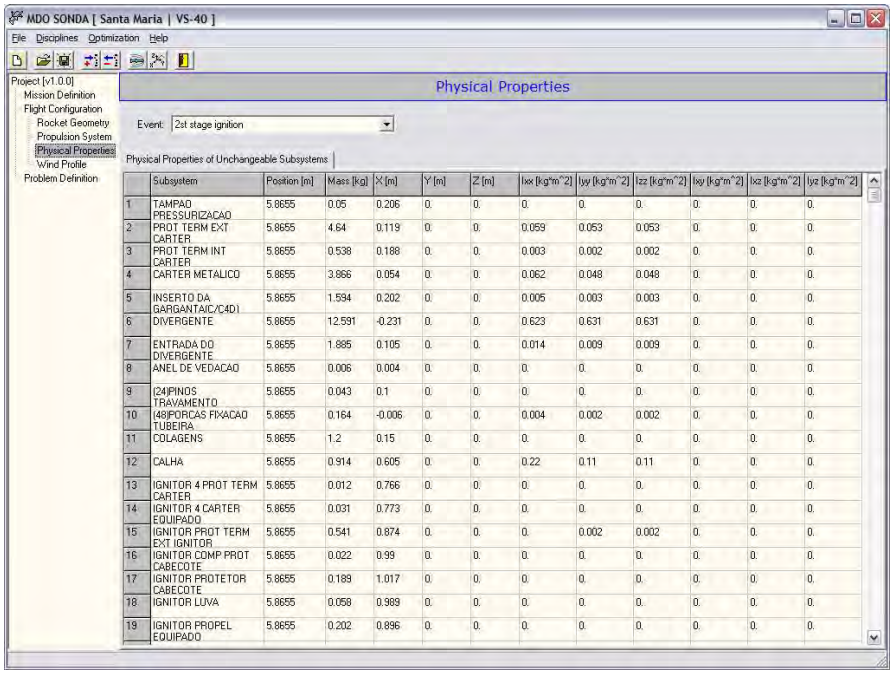


Figura D.10 -Entrada das propriedades de massa e inércia do propelente.



	Subsystem	Position [m]	Mass [kg]	X [m]	Y [m]	Z [m]	Ixx [kg*m^2]	Iyy [kg*m^2]	Izz [kg*m^2]	Ixy [kg*m^2]	Ixz [kg*m^2]	Iyz [kg*m^2]
1	TAMPAO PRESSURIZACAO	5.8655	0.05	0.206	0.	0.	0.	0.	0.	0.	0.	0.
2	PROT TERM EXT CARTER	5.8655	4.64	0.119	0.	0.	0.059	0.053	0.053	0.	0.	0.
3	PROT TERM INT CARTER	5.8655	0.538	0.188	0.	0.	0.003	0.002	0.002	0.	0.	0.
4	CARTER METALICO	5.8655	3.866	0.054	0.	0.	0.062	0.048	0.048	0.	0.	0.
5	INSERTO DA GARGANTA(C/C4D)	5.8655	1.594	0.202	0.	0.	0.005	0.003	0.003	0.	0.	0.
6	DIVERGENTE	5.8655	12.591	-0.231	0.	0.	0.623	0.631	0.631	0.	0.	0.
7	ENTRADA DO DIVERGENTE	5.8655	1.885	0.105	0.	0.	0.014	0.009	0.009	0.	0.	0.
8	ANEL DE VEDACAO	5.8655	0.006	0.004	0.	0.	0.	0.	0.	0.	0.	0.
9	(24)PINOS TRAVAMENTO	5.8655	0.043	0.1	0.	0.	0.	0.	0.	0.	0.	0.
10	(48)PORCAS FIXACAO TUBEIRA	5.8655	0.164	-0.006	0.	0.	0.004	0.002	0.002	0.	0.	0.
11	COLAGENS	5.8655	1.2	0.15	0.	0.	0.	0.	0.	0.	0.	0.
12	CALHA	5.8655	0.914	0.605	0.	0.	0.22	0.11	0.11	0.	0.	0.
13	IGNITOR 4 PROT TERM CARTER	5.8655	0.012	0.766	0.	0.	0.	0.	0.	0.	0.	0.
14	IGNITOR 4 CARTER EQUIPADO	5.8655	0.031	0.773	0.	0.	0.	0.	0.	0.	0.	0.
15	IGNITOR PROT TERM EXT IGNITOR	5.8655	0.541	0.874	0.	0.	0.	0.002	0.002	0.	0.	0.
16	IGNITOR COMP PROT CABECOTE	5.8655	0.022	0.99	0.	0.	0.	0.	0.	0.	0.	0.
17	IGNITOR PROTETOR CABECOTE	5.8655	0.189	1.017	0.	0.	0.	0.	0.	0.	0.	0.
18	IGNITOR LUVA	5.8655	0.058	0.989	0.	0.	0.	0.	0.	0.	0.	0.
19	IGNITOR PROPEL EQUIPADO	5.8655	0.202	0.896	0.	0.	0.	0.	0.	0.	0.	0.

Figura D.11 -Entrada das propriedades de massa e inércia de componentes do foguete.

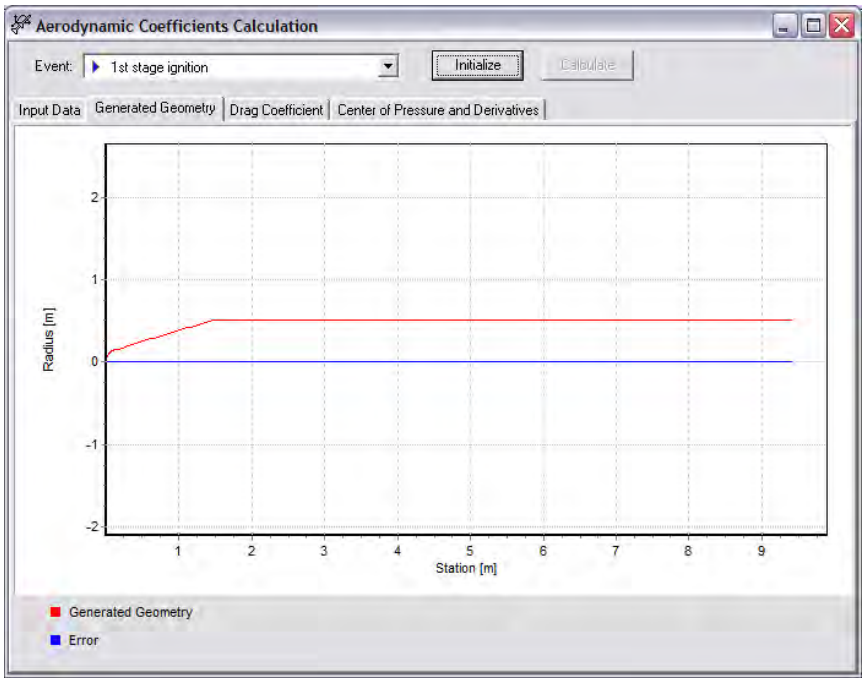


Figura D.12 -Interface para verificar a interpolação da geometria do foguete, retornada pelo programa *missile datcom*.

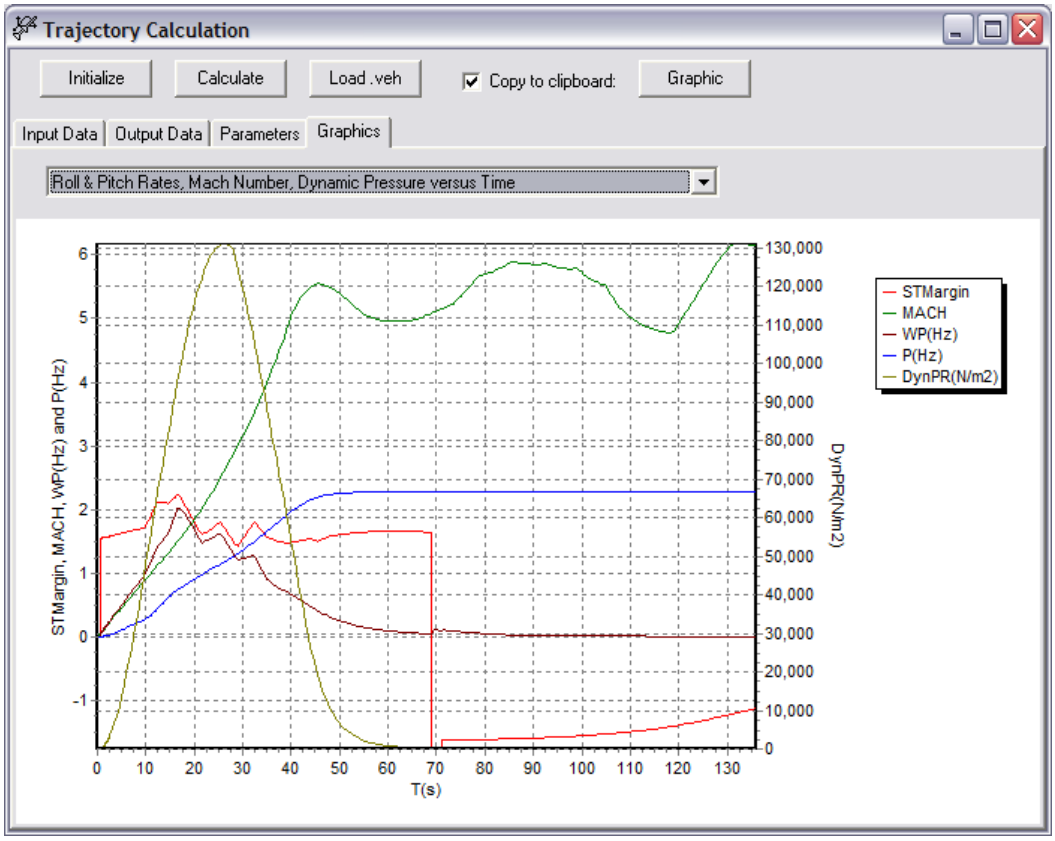
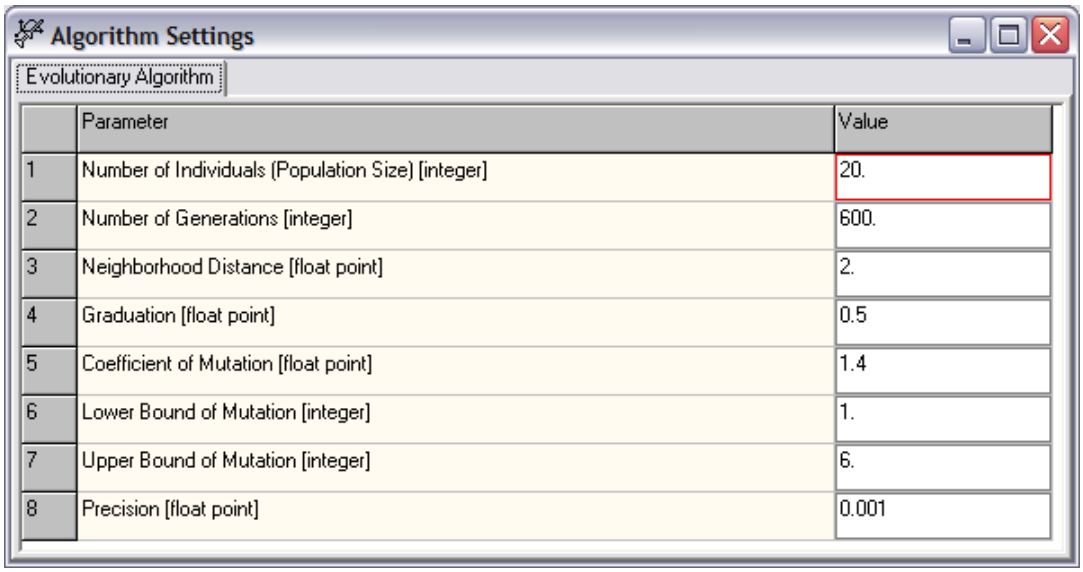


Figura D.13 -Gráficos dos parâmetros de trajetória calculados pelo programa ROSI.



Algorithm Settings

Evolutionary Algorithm

	Parameter	Value
1	Number of Individuals (Population Size) [integer]	20.
2	Number of Generations [integer]	600.
3	Neighborhood Distance [float point]	2.
4	Graduation [float point]	0.5
5	Coefficient of Mutation [float point]	1.4
6	Lower Bound of Mutation [integer]	1.
7	Upper Bound of Mutation [integer]	6.
8	Precision [float point]	0.001

Figura D.14 -Entrada dos parâmetros do NGMOGA com o operador real ORAD.

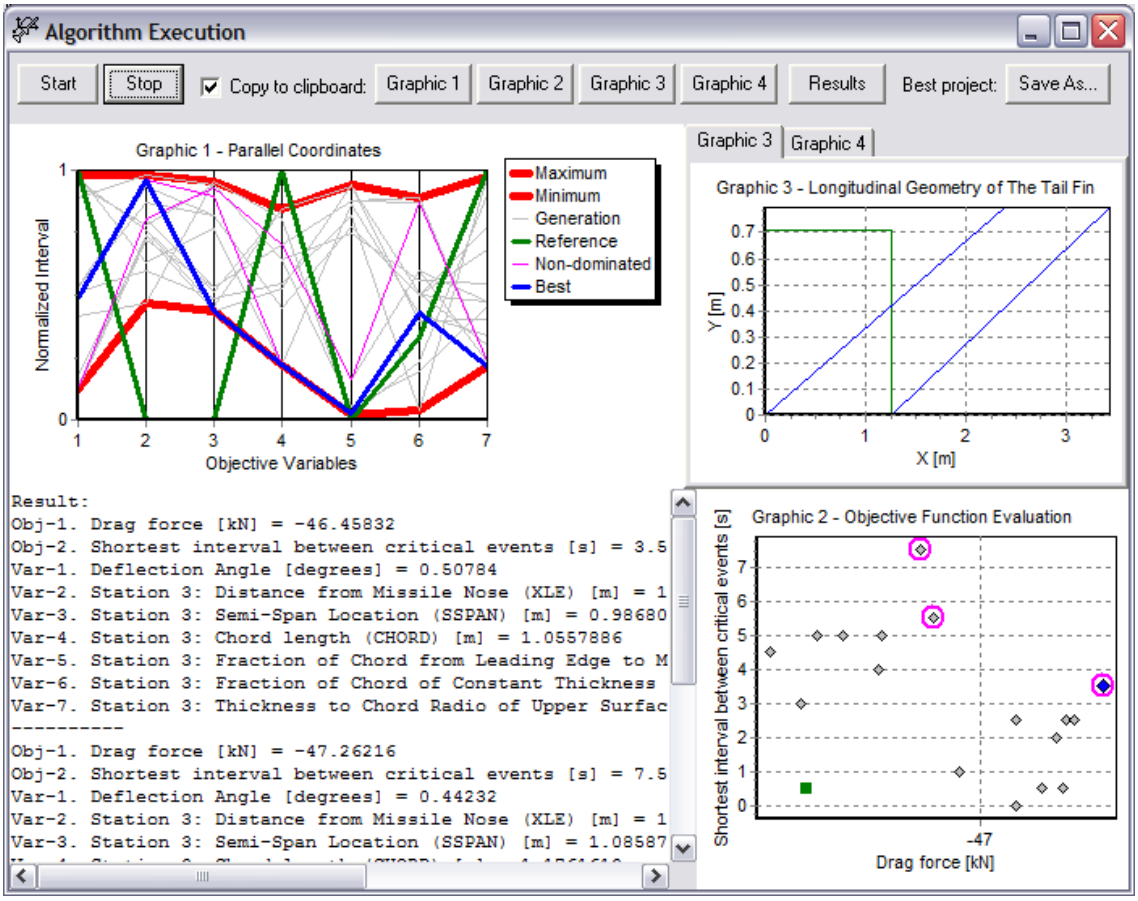


Figura D.15 -Interface para acompanhar e visualizar os resultados da otimização das empenas.

APÊNDICE E – LEVANTAMENTO HISTÓRICO

E.1 Introdução

Neste apêndice, faz-se uma revisão da evolução dos algoritmos e do avanço das máquinas computacionais até sua confluência com a metodologia denominada *multidisciplinary design optimization* (MDO), aplicada a sistemas complexos de engenharia. Primeiro, discutem-se a origem dos métodos de otimização, o crescente interesse sobre os métodos a partir do ano de 1947, considerado um marco na história da otimização, a evolução dos algoritmos e das máquinas computacionais, o surgimento dos métodos estocásticos e a otimização de múltiplos objetivos conflitantes. Em seguida, o desenvolvimento do conceito de inteligência artificial, os métodos baseados em processos biológicos e a evolução da computação paralela. Por último, a otimização de sistemas complexos de engenharia.

E.2 Origem dos métodos de otimização

Os métodos de otimização remontam do surgimento do cálculo diferencial, introduzido por Sir Isaac Newton (1643-1727), físico e matemático inglês, e seu contemporâneo Gottfried W. von Leibniz (1646-1716), matemático alemão. Na década de 1870, K. Weierstrass (1815-1897), matemático alemão, formulou um teorema importante. Esse teorema diz que qualquer função contínua tomada em um intervalo fechado é limitada e possui mínimo e máximo nesse intervalo (FERGUSON, 2004, p. 20). Ainda no século XIX, um artigo não publicado de Bernhard Riemann e exposições detalhadas feitas pelo matemático russo P. A. Nekrasov abordaram a ideia de um dos precursores dos métodos de otimização, o método *Steepest Descent*, atribuído com frequência ao físico Peter Debye, por ter publicado o método em 1909, embora os elementos que o originaram remontassem de aplicações feitas por Augustin L. Cauchy (1789-1857), matemático francês, em problemas de otimização irrestrita (RAO, 1996, p. 3; PETROVA; SOLOV'EV, 1997). Naquela época, já se experimentava uma nova visão da ciência, que teve origem em 1760, na Inglaterra, com a Revolução Industrial: "o uso da máquina como substituta da força muscular humana" (MUNAKATA, 2008, p. 1, tradução

nossa). Mais tarde, no século XX, escritores lançariam luz a horizontes futurísticos, provocando predições adversas sobre a construção de máquinas inteligentes.

E.3 Ano de 1947: um marco na história da otimização

Em 1947, George B. Dantzig (1914-2005), matemático estadunidense, desenvolveu o método simplex com o objetivo de resolver problemas complexos de programação linear (RAO, 1996). De 1946 a 1952, Dantzig assumiu a função de matemático conselheiro do responsável pelo controle de finanças da força aérea americana, no Pentágono, onde foi solicitado a encontrar um método mais eficiente para planejar a logística das forças armadas (FLOUDAS; PARDALOS, 2009). Dantzig desenvolveu o método simplex para aplicação em diversas áreas, tais como alocação de recursos, programação do quadro horário de produção e de trabalhadores, planejamento de pasta de investimentos e marketing, dentre outras, além daquelas ligadas às estratégias militares. Foram as estratégias militares e de ordem econômica que geraram a necessidade de se desenvolver muitas das técnicas atuais usadas para resolver problemas de engenharia (RAO, 1996). O ano de 1947 foi um marco na história da otimização, pois o real interesse por essa área manifestou-se a partir dessa época (FLOUDAS; PARDALOS, 2009). Devido à falta de interesse que havia antes, muitas contribuições de outrora foram ignoradas, tais como o trabalho de Karush, em 1939, que tratou de forma satisfatória as restrições em problemas de otimização, colocadas na forma de desigualdades, e que teria aplicação importante na programação não linear. Em 1951, Harold W. Kuhn, matemático estadunidense, e seu professor Albert W. Tucker (1905-1995), matemático com dupla nacionalidade, americana e canadense, trabalharam sobre as condições necessárias e suficientes para determinar se uma solução era ótima, ou não, em problemas de programação linear, construindo os alicerces para que, mais tarde, muito fosse feito em programação não linear (BAZARAA; JARVIS, 1977; FLOUDAS; PARDALOS, 2009). Em 1954, L. R. Ford e D. R. Fulkerson iniciaram os estudos sobre problemas de otimização de fluxo em redes, considerados o ponto de origem da otimização combinatória, dada a importância dos resultados alcançados por eles, embora Pierre de Fermat (1601-1665), matemático francês, tivesse sido o primeiro a estudar grande parte dos problemas de otimização combinatória

(FLOUDAS; PARDALOS, 2009). Na época em que o interesse pelos métodos de otimização despontou, a automatização dos processos de cálculo significava ainda o uso de dispositivos analógicos.

E.4 Evolução dos algoritmos e das máquinas computacionais

Em 1950, quando Alan Turing publicou seu famoso artigo, "*Computing Machinery and Intelligence*", as máquinas computacionais ainda funcionavam a válvula. Apesar de o transistor ter sido inventado no final da década de 1940, pela Bell Telephone Laboratories, a indústria de computadores só começou a utilizá-lo na década de 1960. Além disso, avanços realmente significativos só começaram a surgir a partir da década de 1970 com o desenvolvimento do microprocessador pela Intel. Gordon Moore, fundador da Intel, observou que, em 1950, eram necessários dez bilhões de bilhões de átomos para representar uma única unidade de informação. Esse número se reduzia à metade a cada ano e meio. Moore previu que, em 2020, uma unidade de informação seria representada por um único átomo, o que ficou conhecido como a "Lei de Moore" (RUSSELL; NORVIG, 2010, p. 1038). O desenvolvimento dos algoritmos foi tão importante quanto o das máquinas computacionais (PRESSMAN, 1992). Enquanto o desenvolvimento das máquinas avançava, produzindo microprocessadores mais eficientes e baratos, desenvolvia-se, em paralelo, versões cada vez mais sofisticadas de algoritmos genéticos, redes neurais, lógica nebulosa, sistemas especialistas e outros algoritmos.

E.5 Surgimento dos métodos estocásticos

Em 1955, Dantzig publicou um artigo sobre um modo de lidar com o conhecimento incompleto ou a incerteza dos valores de parâmetros em um problema de programação linear, uma vez que na prática esta ocorrência era comum. E. M. L. Beale propôs um algoritmo para resolver problemas similares (FLOUDAS; PARDALOS, 2009). Este estudo ficou conhecido como *Stochastic Programming* (RAO, 1996). Os métodos de otimização estocásticos (*stochastic methods*) também tiveram origem na década de 1950, e ficaram conhecidos pela denominação de Metaheurísticas. Segundo Dréo et al. (2006), p. 2-3, esses métodos possuíam outras características: não dependiam da

informação do gradiente da função objetivo; eram inspirados por analogia com princípios físicos, biológicos e etológicos; era difícil encontrar valores adequados para seus parâmetros; e seu custo computacional era elevado. Em 1965, Rechenberg concebeu o primeiro algoritmo baseado em estratégias evolucionárias, introduzidas no final da década de 1950, relativas aos métodos de busca inspirados na evolução biológica das espécies (DRÉO et al., 2006). Desde então, os métodos estocásticos começaram a ganhar mais importância. Dez anos depois, em 1975, John Holland propôs as primeiras versões do algoritmo genético. Em 1980, Smith introduziu a Programação Genética. Nessa mesma década, S. Kirkpatrick e seus colegas, especialistas em física estatística, estavam interessados em configurações de baixa energia de materiais magnéticos desordenados, reunidos em vidros de spin. A determinação numérica dessas configurações revelou-se na forma de um problema de otimização bastante difícil, pois os níveis de energia alcançados pelos vidros de spin apresentavam um comportamento multimodal complexo. Em 1983, S. Kirkpatrick, Gelatt e Vecchi propuseram o método *simulated annealing* (recozimento simulado) para resolver esse problema, baseado na técnica experimental de recozimento, onde elevava-se a temperatura do material a níveis altos seguido de um resfriamento lento, a qual era empregada na metalurgia para obter-se um estado sólido bem ordenado de mínima energia, evitando estruturas metaestáveis caracterizadas por mínimos locais de energia (DRÉO et al., 2006). O método era uma adaptação do algoritmo Metropolis-Hastings, apresentado em 1953 por N. Metropolis, o qual permitia descrever o comportamento de um sistema termodinâmico em equilíbrio a certa temperatura. A função objetivo, de modo análogo à energia do material, era minimizada em função de um parâmetro de controle do algoritmo que simbolizava uma temperatura fictícia. Atualmente, o método *Simulated Annealing* e os Algoritmos Genéticos estão dentre as metaheurísticas mais populares, bastante usadas para resolver problemas complexos de otimização.

E.6 Otimização de múltiplos objetivos conflitantes

Entre as décadas de 1950 e 1960, cresceu o interesse por técnicas para solucionar tipos específicos de problemas multiobjetivo, tais como a técnica *Goal Programming*. Essa técnica começou a ser usada a partir de 1955, por Charnes, Cooper e Ferguson, a fim de

resolver problemas do tipo linear multiobjetivo (RAO, 1996). Em 1961, sua denominação apareceu em um texto publicado por Charnes e Cooper. Sua primeira aplicação em projetos de engenharia foi realizada por Ignizio, em 1962, com o objetivo de projetar e posicionar as antenas do segundo estágio do foguete Saturno V, usado no lançamento da cápsula espacial Apollo em uma missão tripulada à Lua. Essa técnica tornou-se um ramo da otimização multiobjetivo, também conhecida pelo nome de *multi-criteria decision analysis* (MCDA) e *multiple-criteria decision making* (MCDM), considerada uma generalização da programação linear com múltiplos objetivos conflitantes.

E.7 Desenvolvimento do conceito de inteligência artificial

A década de 1950 foi marcada por avanços na área da otimização, mas também foi o início dos trabalhos sobre inteligência artificial (IA), cujo termo foi introduzido em 1956 (MUNAKATA, 2008). O termo IA foi descrito como: "a capacidade de computadores ou programas de operar como mímicos do processo do pensamento humano, tal como o processo do raciocínio e da aprendizagem". Apesar de ser uma descrição ortodoxa, o campo da IA evoluiu, conquistando um largo espectro de aplicações (MUNAKATA, 2008). A IA sempre manteve fortes laços com as abordagens de otimização, mas, também, incluía outras abordagens, tais como: inferência baseada em conhecimento, raciocínio com incerteza ou informação incompleta; técnicas de percepção e aprendizagem; e aplicações em problemas de controle, predição e classificação. No final da década de 1950, quando um programa de computador resolveu a maioria dos problemas de um exame final para iniciantes do Instituto de Tecnologia de Massachusetts, houve agitação sobre o futuro promissor da IA (MUNAKATA, 2008). No entanto, a falta de computadores potentes para tratar a complexidade dos métodos da IA e a publicação do livro "Perceptrons: *an introduction to computational geometry*" por Marvin Minsky e Seymour Papert, publicado em 1969, onde ressaltaram-se as limitações das redes artificiais de neurônios, como, por exemplo, a impossibilidade de simular o operador ou-exclusivo com perceptrons de uma camada, contribuíram para uma visão negativa da IA que perdurou até a década de 80 (BITTENCOURT, 2001). Em 2011, o programa de televisão "Jeopardy!", criado por

Merv Griffin, em 1964, promoveu uma disputa acirrada entre o homem e a máquina. Uma complexa bateria de perguntas cobrindo uma ampla variedade de tópicos foi dirigida a três competidores, dentre os quais, dois eram humanos, Ken Jennings e Brad Rutter, reconhecidos por seu histórico de vitórias, e outro era um supercomputador chamado Watson, desenvolvido pela IBM, capaz de responder às perguntas colocadas em linguagem natural. Watson venceu a disputa, levando um prêmio de 1 milhão de dólares. Bittencourt (2001) apresentou três eras distintas do desenvolvimento do conceito de inteligência artificial: a clássica, a romântica e a moderna. Na clássica, que se estendeu até o final da década de 1960, o conceito de IA referia-se a um sistema que pudesse manifestar a inteligência humana. Na década de 1970, período relativo à era romântica, o homem acreditava ser capaz de construir um sistema que pudesse simular a inteligência sob determinadas condições. Somente na era moderna, a partir da década de 1980, foi que o conceito evoluiu para aplicações específicas tais como o desenvolvimento de sistemas capazes de simular especialistas humanos. Atualmente, a inteligência artificial pode ser classificada em dois ramos: aquele que tem por objetivo a concepção de sistemas que desenvolvem processos de raciocínio e aprendizagem como os seres humanos; e outro que busca apenas a concepção de sistemas que atuam como especialistas humanos (RUSSELL; NORVIG, 2010, p. 2).

E.8 Métodos baseados em processos biológicos

A forma atual de se caracterizar a inteligência artificial é baseada nos mecanismos fundamentais dos processos biológicos e evolucionários usados para se chegar a uma solução. Os exemplos mais populares dessa categoria são as redes neurais e os algoritmos genéticos. As redes neurais são baseadas em uma estratégia que explora o poder de computação através de uma rede interconectada de unidades de processamento, chamadas de neurônios (RAO, 1996). Essa visão da inteligência artificial é considerada mesmo quando essas abordagens são usadas para solucionar problemas onde o processo não parece imitar a capacidade de raciocínio e aprendizagem humana (MUNAKATA, 2008). Quando surgiram, os métodos baseados em mecanismos evolucionários não eram muito interessantes, pois tinham custo computacional alto (DRÉO et al., 2006). Contudo, nos últimos 20 anos, têm surgido

mais e mais aplicações dos métodos evolucionários devido à crescente capacidade de processamento dos computadores e, em particular, devido ao aprimoramento da computação paralela, a qual explora o paralelismo intrínseco desses métodos.

E.9 Evolução da computação paralela

A computação paralela surgiu a partir da década de 1970 (CHAPMAN et al., 2008). O FORTRAN e o C receberam diversas extensões sem que houvesse uma padronização até que, na década de 1980, os esforços de um comitê de representantes da indústria, denominado *Parallel Computing Forum* (PCF), resultaram na aceitação oficial de um padrão preliminar, o ANSI X3.H5. No entanto, novas arquiteturas de computação paralela continuaram a surgir, causando, novamente, divergências de padronização. Várias ações se seguiram com o objetivo de refinar o padrão que havia anteriormente, o que constituiu o *Architecture Review Board* (ARB). O novo padrão pressionou cientistas e engenheiros a escolher estações de última geração, chamadas de supercomputadores, a fim de executar aplicações computacionais complexas. Em outubro de 1997, surgiu a primeira versão da extensão OpenMP, cujo suporte foi garantido pelas empresas: Compaq, Digital, Intel, IBM e Silicon Graphics. Essa extensão, desenvolvida para o sistema Unix e Windows NT, permitia aos programadores escrever códigos e executá-los em processadores paralelos. A extensão OpenMP tem sido adotada por muitos desenvolvedores de programas de computador. Apesar disso, ainda compete com pacotes tradicionais para aplicações multitarefa e com o pacote MPI. Esses pacotes baseiam-se em expressões de paralelismo de baixo nível. No entanto, todos requerem esforço inicial para organizar os detalhes do programa, a decomposição da estrutura de dados e como programar em um estilo parecido com a linguagem assembly (CHAPMAN et al., 2008, p. xvi). Atualmente, a tecnologia *multicore* supre o desenvolvimento de aplicações paralelas simples, o qual pode ser realizado com a extensão OpenMP. Os antigos supercomputadores são agora substituídos por *clusters* de processadores *multicore*. Em certas aplicações, as tarefas, antes realizadas por estações de trabalho, são agora executadas por computadores *multicore*, acessíveis por um custo relativamente baixo.

E.10 Otimização de sistemas complexos de engenharia

“Um sistema é uma coisa construída a partir de muitas outras coisas, componentes, os quais interagem para um propósito comum” (OLIVER et al., 1997, p. 26, tradução nossa). Com o passar do tempo, os métodos clássicos de otimização, as abordagens de inteligência computacional e as arquiteturas de computação paralela confluíram com outras estratégias de otimização, em engenharia de sistemas complexos. Hoje, essa confluência de métodos, combinada com o desenvolvimento de modelos de engenharia cada vez mais complexos, constitui o que se chama de engenharia de sistemas inteligente (FODOR et al., 2012). Sistemas complexos são governados pela interação entre diversos fenômenos físicos, agrupados em áreas distintas de pesquisa que dão origem a disciplinas do conhecimento humano. Veículos espaciais, aeronaves, embarcações de grande porte, automóveis e outros engenhos modernos são exemplos de sistemas complexos, pois requerem mais e mais interações com múltiplas disciplinas (FLOUDAS; PARDALOS, 2009). Surgiu, então, uma área de pesquisa denominada *multidisciplinary design optimization* (MDO). Esta trata dos métodos de otimização empregados na solução de sistemas complexos de engenharia, a qual explora a interação entre duas ou mais disciplinas consideradas no contexto desses sistemas, por meio de seus modelos computacionais, com o objetivo de promover o ganho global no desempenho do produto e minimizar seu tempo de desenvolvimento. A forma tradicional aplicada à engenharia de sistemas complexos, caracterizada pelo processamento em cascata das disciplinas, gera soluções subótimas, por não explorar, sinergicamente, a interação entre elas. O MDO, à medida que se desenvolve, causa interesse crescente em incorporar modelos computacionais do produto a fim de manipular, simultaneamente, variáveis em mais de uma disciplina, indo além da forma tradicional de desenvolvimento de um projeto.