

**PROGRAMA DE PÓS-GRADUAÇÃO EM
ENGENHARIA MECÂNICA**

**OTIMIZAÇÃO MULTI-OBJETIVO PARA O
PROJETO DE SISTEMAS DE ENGENHARIA**



**UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA
2008**

FRAN SÉRGIO LOBATO

**OTIMIZAÇÃO MULTI-OBJETIVO PARA O PROJETO
DE SISTEMAS DE ENGENHARIA**

Tese apresentada ao Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberlândia, como parte dos requisitos para a obtenção do título de DOUTOR EM ENGENHARIA MECÂNICA.

Área de Concentração: Mecânica dos Sólidos e Vibrações.

Orientador: Valder Steffen Jr.

UBERLÂNDIA - MG

2008

Dados Internacionais de Catalogação na Publicação (CIP).

-
- L796o Lobato, Fran Sérgio, 1976
Otimização multi-objetivo para o projeto de sistemas de engenharia/
Fran Sérgio Lobato - 2008.
402 f.: il.
- Orientador: Valder Steffen Jr.
- Tese (doutorado) - Universidade Federal de Uberlândia, Programa de
de Pós-Graduação em Engenharia Mecânica.
Inclui bibliografia.
1. Engenharia mecânica - Teses. 2. Otimização matemática - Teses. I.
Steffen Jr., Valder. II. Universidade Federal de Uberlândia. Programa de
Pós-Graduação em Engenharia Mecânica. III. Título.

CDU: 621

FRAN SÉRGIO LOBATO

**OTIMIZAÇÃO MULTI-OBJETIVO PARA O PROJETO
DE SISTEMAS DE ENGENHARIA**

Tese **APROVADA** pelo Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberlândia.

Área de Concentração: Mecânica dos Sólidos e Vibrações.

Banca Examinadora:

Prof Dr Valder Steffen Jr - Orientador
FEMEC-UFU

Prof Dr Gilmar Guimarães
FEMEC-UFU

Prof Dr^a Valéria Viana Murata
FEQUI-UFU

Prof Dr Gilberto Pechoto de Melo
DEM-UNESP

Prof Dr Antônio José da Silva Neto
IPRJ-UERJ

Uberlândia, 02 de Setembro de 2008

AGRADECIMENTOS

À Deus, pelas oportunidades oferecidas a cada dia.

Aos meus pais que me deram à vida e me forjaram para o futuro à custas de amor e sacrifícios. Aos meus irmãos e minha família que apesar da minha ausência, sempre me incentivaram a continuar em frente com demonstrações de carinho e apoio incondicional.

Ao meu orientador Valder Steffen Jr, que desde o primeiro instante me recebeu com muita atenção e respeito. Professor Valder, serei eternamente grato pela oportunidade de trabalho, pelos conhecimentos transmitidos, pelas palavras de conforto nos momentos mais difíceis, pela amizade e por ser um exemplo de caráter a ser seguido.

Aos professores da minha banca examinadora, Gilmar Guimarães, Valéria Viana Murata, Gilberto Pechoto de Melo e Antônio José da Silva Neto, pela participação, comentários e sugestões. Aos professores da minha banca de qualificação, Sezimária de Fátima Pereira Saramago, Aristeu da Silveira Neto, Valéria Viana Murata e Luis Cláudio Oliveira Lopes pelos valiosos questionamentos e sugestões.

Aos amigos e parceiros que me ajudaram na elaboração e análise dos estudos de caso deste trabalho. Com destaque para Edu Barbosa Arruda, Alice Rosa da Silva, Fábio de Oliveira Arouca, Davi Leonardo de Souza, Ricardo Humberto Oliveira Filho, Valéria Viana Murata, Luís Cláudio Oliveira Lopes e Antônio José da Silva Neto.

À minha professora Eliamar C. M. Coelho e sua família pelos ensinamentos e pela amizade.

Aos meus amigos pela sinceridade, amizade, carinho e incentivo em todos os momentos da minha vida.

Ao Programa de Pós-Graduação da Faculdade de Engenharia Mecânica.

À CAPES pelo apoio financeiro através da minha bolsa de estudo.

Ao término de mais uma etapa da minha vida, saio com a certeza de que, mais que um trabalho realizado, esta tese é a consolidação de amizades que me acompanharão pelo resto da minha vida. Muito obrigado.

DEDICATÓRIA

Dedico este trabalho à meus pais Maria Helena e Luiz Antônio.

SUMÁRIO

Sumário	iii
Lista de Figuras	ix
Lista de Tabelas	xvii
Lista de Símbolos	xxiii
Resumo	xxxvii
Abstract	xxxix
1 Introdução	1
2 O Problema Geral de Otimização: Mono-objetivo e Multi-objetivo	7
2.1 Conceitos Básicos sobre Otimização	7
2.1.1 Otimização: O Que é e Por que Utilizar?	7
2.1.2 Categorias de Otimização	8
2.1.3 Procedimento Geral para a Solução de Problemas de Otimização	9
2.1.4 Obstáculos para se Otimizar	10
2.2 Formulação Geral do Problema de Otimização	11
2.3 O Problema de Otimização Multi-objetivo	13
2.3.1 Mono-objetivo versus Multi-objetivo	13
2.4 Ótimo de Pareto	14
2.4.1 Operador de Dominância de Pareto	16

2.4.2	Otimidade de Pareto	16
2.5	Metas em Otimização Multi-objetivos	18
2.5.1	Métricas de Desempenho	18
2.5.2	Métricas de Convergência	21
2.6	Metodologias para a Resolução do POMO	23
2.6.1	Quanto ao Tipo de Abordagem	23
2.6.2	Quanto à Forma de Tratamento do Problema	24
3	Técnicas de Otimização Multi-objetivos	27
3.1	Aspectos Gerais	27
3.2	Métodos para o Tratamento de POMO	27
3.2.1	Método da Soma Ponderada	29
3.2.2	Método de Otimização Hierárquico	31
3.2.3	Método da Negociação	32
3.2.4	Método do Critério Global	34
3.2.5	Método da Programação de Metas	35
3.2.6	Método do Vetor Objetivo	39
3.2.7	Otimização de Compromisso	39
3.2.8	Método do Critério Ponderado Exponencial	40
3.2.9	Método do Produto Ponderado	40
3.2.10	Métodos Min-Max	41
4	Estudo de POMO usando Métodos Clássicos	43
4.1	Método Randômico Clássico	43
4.2	Métodos de Ordem Superior	44
4.2.1	Métodos para a Determinação da Direção de Busca	45
4.2.2	Determinação do Passo na Direção de Busca	48
4.2.3	Critérios de Convergência	48
4.3	Métodos Seqüenciais	50
4.4	Métodos Diretos	51

4.4.1	Programação Sequencial Linear	51
4.4.2	Gradiente Reduzido Generalizado	52
4.4.3	Programação Quadrática Sequencial	54
4.4.4	Vantagens e Desvantagens dos Métodos Diretos	55
5	Estudo de POMO usando Métodos Heurísticos	59
5.1	Aspectos Gerais	59
5.2	Algoritmos Genéticos	61
5.2.1	Codificação dos Algoritmos Genéticos	63
5.2.2	Geração da População Inicial	64
5.2.3	Avaliação da População	64
5.2.4	Operadores Genéticos	64
5.2.5	Algoritmos Genéticos para Otimização Multi-objetivo	67
5.3	Recozimento Simulado	87
5.4	Busca Tabu	90
5.5	Colônia de Formigas	91
5.6	Sistema Imunológico Artificial	93
5.7	Enxame de Partículas	95
5.8	Evolução Diferencial	97
6	Métodos Híbridos, Métodos para o Tratamento de Restrições e Métodos para a Atualização dos Parâmetros em AE	105
6.1	Tratamento de POMO usando Métodos Híbridos	105
6.1.1	Algoritmos que Acoplam Técnicas Não Determinísticas	107
6.1.2	Algoritmos que Acoplam Técnicas Determinísticas e Não Determinísticas	108
6.2	Tratamento de Restrições de Igualdade e Desigualdade	109
6.2.1	Métodos Baseados em Funções de Penalidade	109
6.2.2	Método da Pena de Morte	112
6.2.3	Métodos Baseados na Busca por Soluções Viáveis	113
6.2.4	Métodos Baseados na Preservação das Soluções Viáveis	114

6.2.5	Métodos Baseados em Decodificadores	114
6.3	Atualização dos Parâmetros em Algoritmos Evolutivos	115
7	Otimização Multi-objetivo via Evolução Diferencial	119
7.1	Aspectos Gerais	119
7.2	O Algoritmo MODE	120
7.2.1	Inicialização	121
7.2.2	Operador de Ordenamento por <i>Rank</i>	121
7.2.3	Operador de Truncamento das Soluções	122
7.2.4	Operador de Seleção	124
7.2.5	Operador de Reprodução	125
7.2.6	Operador de Elitismo	125
7.2.7	Operador para a Exploração das Vizinhanças	126
7.2.8	Operador para o Tratamento de Restrições	130
7.2.9	Critério de Parada	130
7.2.10	Tratamento de Problemas com Variáveis Inteiras e Discretas . .	131
7.2.11	A Complexidade do Algoritmo MODE	133
7.3	Aspectos Gerais de Implementação e Estruturação do MODE	134
7.3.1	Arquivos de Entrada e Saída do Algoritmo MODE	134
7.3.2	Subrotinas do Algoritmo MODE	136
7.3.3	Estratégias de Erros	137
8	Aplicações Matemáticas	141
8.1	Funções-Teste para POMO	142
8.2	Funções-Teste com Restrições para POMO	173
8.3	Estudo de Sensibilidade do Algoritmo MODE	189
8.3.1	Análise da Incorporação do Operador de Exploração das Vizinhanças no MODE	190
8.3.2	Análise de Sensibilidade de Parâmetros do MODE	194
8.4	Conclusões Preliminares	204

9	Aplicações em Engenharia e Áreas Afins	207
9.1	Problemas de Projetos Estruturais	207
9.1.1	Problema de uma Viga em Balanço	207
9.1.2	Problema de Duas Barras	210
9.1.3	Problema da Viga I	212
9.1.4	Problema da Viga Soldada	215
9.2	Problemas Inversos	218
9.2.1	Estimação do Coeficiente de Transferência de Calor em Leito Deslizante	219
9.2.2	Estimação de Parâmetros Cinéticos no Processo de Fermentação Batelada	224
9.2.3	Estimação de Parâmetros Radiativos e Térmicos	233
9.2.4	Estimação de Parâmetros Cinéticos em um Secador Rotativo	240
9.3	Problemas de Controle Ótimo	246
9.3.1	PCO com Duas Variáveis de Controle	247
9.3.2	O Problema da Fermentação em Batelada Alimentada	253
9.3.3	O Problema de <i>Jacobson & Lele</i>	259
9.4	Problemas com Variáveis de Projeto Inteiras e Discretas	265
9.4.1	O Problema do Projeto de Engrenagens	266
9.4.2	O Problema do Projeto de uma Mola	268
9.4.3	O Problema da Viga Discreta	271
9.5	Problemas de Engenharia Química	274
9.5.1	Projeto Ótimo de uma Coluna de Destilação	274
9.5.2	Otimização de um Reator Não-Adiabático	280
9.5.3	Otimização de um Reator PFR com Reações em Paralelo	288
9.6	Otimização de Sistemas Reais usando Meta-Modelos	294
9.6.1	Projeto Ótimo de um Robô Industrial	298
9.6.2	O Problema de Interação Fluido-Estrutura	304
9.7	Conclusões Preliminares	314

10 Conclusões e Sugestões	317
10.1 Conclusões	317
10.1.1 Contribuições	319
10.1.2 Resultados	320
10.2 Sugestões para Trabalhos Futuros	323
Referências Bibliográficas	325
Apêndices	341
Apêndice A - Arquivos de Entrada e Saída do Algoritmo MODE	341
Apêndice B - Artigos Publicados e Submetidos	352
Apêndice C - Sobre o Autor	354

LISTA DE FIGURAS

2.1	Categorias dos algoritmos de otimização	8
2.2	Otimização de compromisso para a minimização de dois objetivos	13
2.3	Idealizadores do conceito de ótimo para o problema multi-objetivo	14
2.4	Opções de compra de um carro: custo versus conforto	15
2.5	Exemplos de conjuntos ótimos de Pareto	17
2.6	Soluções ótimas de Pareto locais e globais	17
2.7	Distribuição de soluções na fronteira de Pareto	19
2.8	Métricas de desempenho em algoritmos genéticos	19
2.9	Distribuição versus convergência na fronteira de Pareto	20
2.10	Comparação entre dois algoritmos hipotéticos	20
3.1	Estrutura geral para a resolução de POMO sem o uso do critério de dominância	28
3.2	Interpretação geométrica do método da soma ponderada: caso convexo	30
3.3	Interpretação geométrica do método da soma ponderada: caso não-convexo	31
3.4	Programação lexocográfica	38
4.1	Definição de regiões dentro do espaço de projeto	49
4.2	Ilustração esquemática das condições de Kuhn-Tucker	50
4.3	Interpretação geométrica do método SLP	52
5.1	Esquema geral de evolução de um AG	63
5.2	Operador de cruzamento	66

5.3	Operador de mutação	67
5.4	<i>Ranking</i> obtido através da aplicação do MOGA	73
5.5	Soluções agrupadas em nichos	74
5.6	Estrutura do NSGA II	78
5.7	Esquema geral do operador distância da multidão - NSGA II	79
5.8	Situação onde o NSGA II falha	81
5.9	Cálculo da aptidão através do algoritmo SPEA II	82
5.10	Analogia entre o processo de otimização e o recozimento simulado	88
5.11	Estrutura multicamadas para o sistema imunológico	93
5.12	Estrutura do princípio da seleção clonal	94
5.13	Fundamentação teórica do algoritmo de ED	98
7.1	Estratégia de ordenamento por <i>rank</i>	122
7.2	Operador distância da multidão	124
7.3	Vizinhos de indivíduos em curvas não-dominadas	127
7.4	Relação entre pseudo-curvas e ordenamento por <i>rank</i>	129
7.5	Tela de apresentação do algoritmo MODE	134
7.6	Arquivos de entrada e saída do algoritmo MODE	135
7.7	Subrotinas do algoritmo MODE	136
7.8	Estratégia de erros no algoritmo MODE	137
8.1	População inicial para a função SCH1	143
8.2	Curva de Pareto para a função SCH1	144
8.3	População inicial para a função SCH2	145
8.4	Curva de Pareto para a função SCH2	146
8.5	População inicial para a função FON	147
8.6	Curva de Pareto para a função FON	148
8.7	População inicial para a função KUR	149
8.8	Curva de Pareto para a função KUR	150
8.9	Espaço de variáveis de projeto para a função KUR	150

8.10	População inicial para a função GTP	152
8.11	Curva de Pareto para a função GTP	152
8.12	Resolução da função ZDT1 usando algoritmos clássicos	154
8.13	População inicial para a função ZDT1	155
8.14	Curva de Pareto para a função ZDT1	156
8.15	Resolução da função ZDT2 usando algoritmos clássicos	157
8.16	População inicial para a função ZDT2	158
8.17	Curva de Pareto para a função ZDT2	158
8.18	Resolução da função ZDT3 usando algoritmos clássicos	159
8.19	População inicial para a função ZDT3	160
8.20	Curva de Pareto para a função ZDT3	161
8.21	Resolução da função ZDT4 usando algoritmos clássicos	162
8.22	População inicial para a função ZDT4	163
8.23	Curva de Pareto para a função ZDT4	164
8.24	Resolução da função ZDT6 usando algoritmos clássicos	165
8.25	População inicial para a função ZDT6	166
8.26	Curva de Pareto para a função ZDT6	167
8.27	População inicial para a função VNT	169
8.28	Curva de Pareto para a função VNT	169
8.29	Espaço de objetivos $f_2 \times f_1$ para a função VNT	170
8.30	Espaço de objetivos $f_3 \times f_1$ para a função VNT	170
8.31	Espaço de objetivos $f_3 \times f_2$ para a função VNT	171
8.32	Curva de Pareto para a função DTLZ2 usando o NSGA II	172
8.33	Curva de Pareto para a função DTLZ2 usando o MODE	173
8.34	População inicial para a função MIN-EX	174
8.35	Curva de Pareto para a função MIN-EX	175
8.36	População inicial para a função BNH	176
8.37	Curva de Pareto para a função BNH	177
8.38	População inicial para a função OSY	179

8.39	Curva de Pareto para a função OSY	179
8.40	População inicial para a função SRN	180
8.41	Curva de Pareto para a função SRN	181
8.42	População inicial para a função TNK	182
8.43	Curva de Pareto para a função TNK	183
8.44	$f_1 \times f_2$ para a função 5OBJ	184
8.45	$f_1 \times f_3$ para a função 5OBJ	185
8.46	$f_1 \times f_4$ para a função 5OBJ	185
8.47	$f_1 \times f_5$ para a função 5OBJ	186
8.48	$f_2 \times f_3$ para a função 5OBJ	186
8.49	$f_2 \times f_4$ para a função 5OBJ	187
8.50	$f_2 \times f_5$ para a função 5OBJ	187
8.51	$f_3 \times f_4$ para a função 5OBJ	188
8.52	$f_3 \times f_5$ para a função 5OBJ	188
8.53	$f_4 \times f_5$ para a função 5OBJ	189
8.54	Convergência métrica $\times$ tamanho da população para 200 gerações	191
8.55	Diversidade métrica $\times$ tamanho da população para 200 gerações	192
8.56	Convergência métrica $\times$ tamanho da população para 100 indivíduos	193
8.57	Diversidade métrica $\times$ tamanho da população para 100 indivíduos	193
8.58	Convergência métrica $\times$ taxa de redução	195
8.59	Distância geracional $\times$ taxa de redução	195
8.60	Diversidade métrica $\times$ taxa de redução	196
8.61	Convergência métrica $\times$ probabilidade de cruzamento	197
8.62	Distância geracional $\times$ probabilidade de cruzamento	197
8.63	Diversidade métrica $\times$ probabilidade de cruzamento	198
8.64	Convergência métrica $\times$ taxa de perturbação	199
8.65	Distância geracional $\times$ taxa de perturbação	199
8.66	Diversidade métrica $\times$ taxa de perturbação	200
8.67	Convergência métrica $\times$ tamanho da população	200

8.68	Distância geracional $\times$ tamanho da população	201
8.69	Diversidade métrica $\times$ tamanho da população	201
8.70	Convergência métrica $\times$ número de gerações	202
8.71	Distância geracional $\times$ número de gerações	203
8.72	Diversidade métrica $\times$ número de gerações	203
9.1	Projeto de uma viga em balanço	207
9.2	População inicial para o projeto de uma viga em balanço	209
9.3	Curva de Pareto para o projeto de uma viga em balanço	209
9.4	Estrutura do problema de duas barras	210
9.5	População inicial para o problema de duas barras	211
9.6	Curva de Pareto para o problema de duas barras	212
9.7	Viga em seção I	212
9.8	População inicial para o problema da viga I	214
9.9	Curva de Pareto para o problema da viga I	215
9.10	Viga soldada	215
9.11	População inicial para o problema da viga soldada	217
9.12	Curva de Pareto para o problema da viga soldada	218
9.13	Fluxograma simplificado da unidade piloto de leito deslizante	219
9.14	Curva de Pareto para o problema da estimação da transferência de calor	222
9.15	Perfis de T_g no plano $x=1,5$ cm em função da altura do secador	223
9.16	Perfis de T_g no plano $z=1,5$ cm em função da posição do secador	223
9.17	Perfis de T_s no plano $x=1,5$ cm em função da altura do secador	224
9.18	Perfis de T_s no plano $z=1,5$ cm em função da posição do secador	224
9.19	Biorreator para fermentação de etanol	225
9.20	$f_1 \times f_2$ para o problema da fermentação em reator batelada	228
9.21	$f_1 \times f_3$ para o problema da fermentação em reator batelada	228
9.22	$f_1 \times f_4$ para o problema da fermentação em reator batelada	229
9.23	$f_2 \times f_3$ para o problema da fermentação em reator batelada	229

9.24	$f_2 \times f_4$ para o problema da fermentação em reator batelada	230
9.25	$f_3 \times f_4$ para o problema da fermentação em reator batelada	230
9.26	Perfis de concentração de células	231
9.27	Perfis de concentração de glucose	232
9.28	Perfis de concentração de etanol	232
9.29	Perfis de concentração de glicerol	233
9.30	Transferência de calor por condução e radiação térmica em um meio partici- pante	233
9.31	População inicial para o problema de transferência de calor por radiação e condução sem ruído	236
9.32	Curva de Pareto para o problema de transferência de calor por radiação e condução sem ruído	236
9.33	Perfil de intensidade de radiação	237
9.34	Perfil de temperatura para o problema de transferência de calor por radiação	238
9.35	Perfis da intensidade da radiação para diferentes valores do cosseno do ân- gulo polar μ com relação à variável óptica τ	238
9.36	Perfis da intensidade da radiação para diferentes valores do cosseno do ân- gulo polar μ com relação à variável óptica τ	239
9.37	Curva de Pareto para o problema de transferência de calor por radiação e condução com ruído	239
9.38	Secador rotativo com cascadeamento	240
9.39	Esquema da unidade piloto do secador rotativo	241
9.40	População inicial para o secador rotativo	244
9.41	Curva de Pareto para o secador rotativo	244
9.42	Alguns perfis de umidade do secador rotativo	245
9.43	Alguns perfis de temperatura do secador rotativo	246
9.44	Biorreator de polimerização de proteína	247
9.45	Curva de Pareto para o processo da produção de proteína	250
9.46	Perfis de volume	251
9.47	Perfis de concentração de células	251
9.48	Perfis de concentração de glicose	252

9.49	Perfis de produção de produto	252
9.50	Perfis médios de alimentação de glicose e de indutor	253
9.51	Biorreator para fermentação de etanol	253
9.52	Curva de Pareto para o problema da fermentação em reator batelada alimentada	256
9.53	Perfis de concentração de células	257
9.54	Perfis de concentração de substrato	257
9.55	Perfis de concentração de produto	258
9.56	Perfis de volume do reator	258
9.57	Perfis de fluxo de alimentação	259
9.58	Curva de Pareto para o problema <i>Jacobson & Lele</i>	263
9.59	Perfis da variável de estado x_1	264
9.60	Perfis da variável de estado x_2	264
9.61	Perfis da variável de controle	265
9.62	Projeto de um sistema de engrenagens	266
9.63	População inicial para o projeto de engrenagens	267
9.64	Curva de Pareto para o problema de projeto de engrenagens	268
9.65	População inicial para o problema do projeto de uma mola	270
9.66	Curva de Pareto para o problema do projeto de uma mola	270
9.67	Problema do projeto de um viga discreta	271
9.68	População inicial para o projeto da viga discreta	272
9.69	Curva de Pareto para o problema do projeto da viga discreta	273
9.70	Estrutura geral de uma coluna de destilação	274
9.71	População inicial para o problema da coluna de destilação	278
9.72	Curva de Pareto para o problema da coluna de destilação	278
9.73	Perfis de composição na coluna de destilação obtidos a partir da simulação da solução B	279
9.74	Reator para conversão de SO_2 em SO_3	281
9.75	População inicial para o problema do reator não-adiabático	285
9.76	Curva de Pareto para o problema do reator não-adiabático	285

9.77	Perfis de conversão de SO_2 para o problema do reator não-adiabático . . .	286
9.78	Perfis de pressão para o problema do reator não-adiabático	287
9.79	Perfis de temperatura para o problema do reator não-adiabático	287
9.80	Reator PFR	288
9.81	População inicial para o problema do reator PFR com reações em paralelo	291
9.82	Curva de Pareto para o problema do reator PFR com reações em paralelo .	291
9.83	Fa para o problema do reator PFR com reações em paralelo	292
9.84	Fb para o problema do reator PFR com reações em paralelo	292
9.85	Fc para o problema do reator PFR com reações em paralelo	293
9.86	Temperatura para o problema do reator PFR com reações em paralelo . .	293
9.87	Meta-modelo, modelo de simulação e problema real	295
9.88	Etapas da resolução do problema real usando meta-modelos	297
9.89	Robô industrial com mecanismo de mola hidráulica	298
9.90	Curva de Pareto para o problema do projeto ótimo de um robô industrial usando o MODE	302
9.91	$f_1 \times f_2$ para o problema do projeto ótimo de um robô industrial	303
9.92	$f_1 \times f_3$ para o problema do projeto ótimo de um robô industrial	303
9.93	$f_2 \times f_3$ para o problema do projeto ótimo de um robô industrial	304
9.94	Representação esquemática do sistema massa-mola-amortecimento (1 GDL)	305
9.95	$f_1 \times f_2$ para o problema de interação fluido-estrutura	311
9.96	$f_1 \times f_3$ para o problema de interação fluido-estrutura	311
9.97	$f_1 \times f_4$ para o problema de interação fluido-estrutura	312
9.98	$f_2 \times f_3$ para o problema de interação fluido-estrutura	312
9.99	$f_2 \times f_4$ para o problema de interação fluido-estrutura	313
9.100	$f_3 \times f_4$ para o problema de interação fluido-estrutura	313

LISTA DE TABELAS

3.1	Tabela de pagamento para o método da negociação	33
3.2	Tabela de pagamento usando a função incremento Δf_{ij}	34
4.1	Vantagens e desvantagens dos algoritmos SLP, GRG e SQP	56
5.1	Algoritmo evolutivo conceitual	61
5.2	Algoritmos evolutivos multi-objetivos baseados em algoritmos genéticos . .	69
5.3	Cálculo da distância da multidão - NPGA II	79
5.4	Representação das estratégias do algoritmo de ED	100
7.1	Distância da multidão	123
7.2	Limites inferiores e superiores dos parâmetros do MODE	139
8.1	Nomenclatura dos parâmetros utilizados no MODE	141
8.2	Parâmetros utilizados na resolução da função SCH1	143
8.3	Métricas de desempenho para a função SCH1	143
8.4	Parâmetros utilizados na resolução da função SCH2	145
8.5	Métricas de desempenho para a função SCH2	145
8.6	Parâmetros utilizados na resolução da função FON	147
8.7	Métricas de desempenho para a função FON	147
8.8	Parâmetros utilizados na resolução da função KUR	149
8.9	Métricas de desempenho para a função KUR	149
8.10	Parâmetros utilizados na resolução da função GTP	151

8.11 Métricas de desempenho para a função GTP	151
8.12 Parâmetros utilizados nos algoritmos clássicos	153
8.13 Parâmetros utilizados na resolução da função ZDT1	155
8.14 Métricas de desempenho para a função ZDT1	155
8.15 Parâmetros utilizados na resolução da função ZDT2	157
8.16 Métricas de desempenho para a função ZDT2	157
8.17 Parâmetros utilizados na resolução da função ZDT3	160
8.18 Métricas de desempenho para a função ZDT3	161
8.19 Parâmetros utilizados na resolução da função ZDT4	162
8.20 Métricas de desempenho para a função ZDT4	163
8.21 Parâmetros utilizados na resolução da função ZDT6	165
8.22 Métricas de desempenho para a função ZDT6	166
8.23 Parâmetros utilizados na resolução da função VNT	168
8.24 Métricas de desempenho para a função VNT	168
8.25 Parâmetros utilizados na resolução da função DTLZ2	172
8.26 Métricas de desempenho para a função DTLZ2	172
8.27 Parâmetros utilizados na resolução da função MIN-EX	174
8.28 Métricas de desempenho para a função MIN-EX	175
8.29 Parâmetros utilizados na resolução da função BNH	176
8.30 Métricas de desempenho para a função BNH	177
8.31 Ótimo de Pareto para a função OSY	178
8.32 Parâmetros utilizados na resolução da função OSY	178
8.33 Métricas de desempenho para a função OSY	178
8.34 Parâmetros utilizados na resolução da função SRN	180
8.35 Métricas de desempenho para a função SRN	181
8.36 Parâmetros utilizados na resolução da função TNK	182
8.37 Métricas de desempenho para a função TNK	183
8.38 Parâmetros utilizados na resolução da função 5OBJ	184
8.39 Parâmetros utilizados pelos algoritmos MODE, ED, NSGA II	191

8.40	Número de avaliações da função objetivo para o NSGA II, MODE e para o ED para 200 gerações	191
8.41	Número de avaliações da função objetivo para o NSGA II, MODE e o ED para 100 indivíduos	194
9.1	Parâmetros utilizados no problema do projeto de uma viga em balanço . .	208
9.2	Soluções da curva de Pareto para o projeto de uma viga em balanço	208
9.3	Parâmetros utilizados na resolução do problema de duas barras	211
9.4	Parâmetros utilizados na resolução do problema da viga I	214
9.5	Soluções da curva de Pareto para o projeto de uma viga I	214
9.6	Parâmetros utilizados na resolução do problema da viga soldada	217
9.7	Soluções da curva de Pareto para o projeto de uma viga soldada	218
9.8	Propriedades do sólido e do gás e condições operacionais	221
9.9	Parâmetros utilizados na resolução do problema da estimação do coeficiente global de transferência de calor em leito deslizante	222
9.10	Parâmetros usados na resolução do problema do fermentador batelada . . .	227
9.11	Soluções de Pareto para o problema da fermentação batelada	231
9.12	Parâmetros usados no problema de transferência de calor por radiação e condução	235
9.13	Curva de Pareto para o problema de transferência de calor por radiação e condução sem ruído	237
9.14	Condições operacionais do secador rotativo	243
9.15	Parâmetros usados no problema do secador rotativo	243
9.16	Algumas soluções da curva de Pareto para o secador rotativo	245
9.17	Condições operacionais do processo da produção de proteína	248
9.18	Parâmetros usados na resolução do problema da produção de proteína . . .	249
9.19	Algumas soluções da curva de Pareto	250
9.20	Parâmetros utilizados na resolução do problema do fermentador batelada alimentada	255
9.21	Algumas soluções da curva de Pareto	256
9.22	Valores dos eventos e da função objetivo	261

9.23	Parâmetros utilizados na resolução do problema do <i>Jacobson & Lele</i> multi-objetivo	262
9.24	Algumas soluções da curva de Pareto	263
9.25	Parâmetros utilizados no problema de projeto de engrenagens	267
9.26	Alguns pontos da curva de Pareto para o problema de projeto de engrenagens	267
9.27	Diâmetros de mola, em polegadas, fornecidos pela ASTM	268
9.28	Parâmetros utilizados na resolução do problema do projeto de uma mola .	269
9.29	Alguns pontos da curva de Pareto para o problema do projeto de uma mola	269
9.30	Parâmetros utilizados na resolução do problema da viga discreta	272
9.31	Alguns pontos da curva de Pareto para o problema da viga discreta	273
9.32	Condições operacionais do problema da coluna de destilação	277
9.33	Parâmetros utilizados na resolução do problema da coluna de destilação . .	277
9.34	Alguns pontos da curva de Pareto para o problema da coluna de destilação	279
9.35	Parâmetros e condições operacionais para o problema do reator não-adiabático	284
9.36	Parâmetros utilizados na resolução do problema do reator não-adiabático .	284
9.37	Alguns pontos da curva de Pareto para o problema do reator não-adiabático	286
9.38	Parâmetros e condições operacionais do reator PFR	290
9.39	Parâmetros utilizados na resolução do problema do reator PFR com reações em paralelo	290
9.40	Alguns pontos da curva de Pareto para o problema do reator PFR com reações em paralelo	292
9.41	Parâmetros usados nas simulações do problema do projeto ótimo de um robô industrial	299
9.42	Planejamento de experimentos e resposta obtida para o problema do projeto ótimo de um robô industrial	300
9.43	Coefficientes polinomiais para o problema do projeto ótimo de um robô industrial	301
9.44	Parâmetros utilizados na resolução do problema do projeto ótimo de um robô industrial	302
9.45	Alguns pontos da curva de Pareto para o problema de projeto de um robô industrial	304

9.46	Planejamento de experimentos e resposta obtida para o problema de interação fluido-estrutura	308
9.47	Coeficientes polinomiais para o problema da interação fluido-estrutura . . .	309
9.48	Parâmetros utilizados na resolução do problema de interação fluido-estrutura	310
9.49	Alguns pontos da curva de Pareto para o problema de interação fluido-estrutura	314
9.50	Tempo de processamento para os problemas de projetos estruturais	315
A.1	Parâmetros utilizados na resolução do problema P1	341

LISTA DE SÍMBOLOS

Glossário

AE	Algoritmo Evolutivo.
AG	Algoritmo Genético.
BFGS	Broyden-Fletcher-Goldfarb-Shanno.
CWA	<i>Conventional Weighted Aggregation.</i>
DFP	Davidon-Fletcher-Powell.
DMOEA	<i>Dynamic Multi-objective Evolutionary Algorithm.</i>
DWA	<i>Dynamic Weighted Aggregation.</i>
EADP	Equação Algébrico-Diferencial Parcial.
ED	Evolução Diferencial.
EDP	Equação Diferencial Parcial.
GDL	Grau de Liberdade.
GENECOP	<i>Genetic Algorithm for Numerical Optimization of Constrained Problems.</i>
GRG	Gradiente Reduzido Generalizado.
HLGA	<i>Hajela-Lin Genetic Algorithm.</i>
LCA	<i>Line-up Competition Algorithm.</i>
MC	Métodos Clássicos.
MOEA	<i>Multi-objective Evolutive Algorithm.</i>
MODE	<i>Multi-objective Differential Evolution.</i>
MOGA	<i>Multi-objective Genetic Algorithm.</i>
MOSA	<i>Multi-objective Simulated Annealing.</i>
NEES	<i>Neighborhood Exploring Evolution Strategy.</i>
NLP	<i>Non-Linear Programming.</i>
NPGA	<i>Niched Pareto Genetic Algorithm.</i>
NSDE	<i>Non-dominated Sorting Differential Evolution.</i>
NSGA	<i>Non-dominated Sorting Genetic Algorithm.</i>
OCF	Otimização por Colônia de Formigas.
PAES	<i>Pareto-Archived Evolution Strategy.</i>
PAMUC	<i>Preferences Applied to Multiobjectivity and Constraints.</i>
PC	Pseudo-Curvas.
PCO	Problema de Controle Ótimo.
PDE	<i>Pareto-frontier Differential Evolution.</i>
PESA	<i>Pareto Envelope-based Selection Algorithm.</i>

POMO	Problema de Otimização Multi-Objetivo.
PMOGA	<i>Pareto Multi-objective Genetic Algorithm.</i>
QP	Problema Quadrático.
RWGA	<i>Random Weighted Genetic Algorithm.</i>
<i>SimpleToolbox</i>	<i>Simple Optimization Toolbox.</i>
SLP	Programação Seqüencial Linear.
SOEA	<i>Strategy Optimization Evolutionary Algorithm.</i>
SPEA	<i>Strength Pareto Evolutionary Algorithm.</i>
SQP	Programação Quadrática Seqüencial.
VEGA	<i>Vector Evaluated Genetic Algorithm.</i>
WBGGA	<i>Weight-Based Genetic Algorithm.</i>

CAPÍTULO 2

Letras Latinas

A, B	Conjunto de soluções não-dominadas.
$C(A, B)$	Métrica de cobertura entre as soluções A e B .
d_i	Distância euclidiana.
$\bar{d}$	Média das distâncias entre duas soluções.
DG	Distância geracional.
e_i	Contador de soluções pertencentes ao Ótimo de Pareto.
ER	Taxa de erro.
f	Função objetivo.
g	Vetor de restrições de desigualdade.
h	Vetor de restrições de igualdade.
J	Número de restrições de desigualdade.
K	Número de restrições de igualdade.
M	Número de objetivos.
$M3$	Espalhamento máximo.
NC	Número de nichos.
p	Contador.
P	Conjunto de soluções.
P^*	Ótimo de Pareto.
Q	Conjunto de soluções dominadas e não-dominadas.
S	Espaço de busca.
Spc	Espaçamento.
u, x	Vetor de variáveis de projeto.
Z	Espaço de objetivos.

Letras Gregas

ϵ	Número real positivo.
σ	
Δ	Diversidade métrica.
Υ	Convergência métrica.

CAPÍTULO 3

Letras Latinas

c, f, T	Função objetivo.
f_i^o	Função objetivo otimizada individualmente (Solução Ideal).
f^r	Função objetivo referência.
f^{worst}	Vetor dos piores valores da função objetivo.
g	Vetor de restrições de desigualdade.
h	Vetor de restrições de igualdade.
J	Número de restrições de desigualdade.
K	Número de restrições de igualdade.
L_p	Métrica de comparação.
m, M	Número de objetivos.
n, k, s	Contador.
n_i, p, p_i	Números reais positivos.
S	Espaço de busca.
x, X	Vetor de variáveis de projeto.
x^*	Projeto ótimo.
t_i	Número real.
w, W	Vetor de parâmetros de ponderação.
z', z'', v	Vetor de objetivos normalizados.

Letras Gregas

α	Contador.
β	Vetor de parâmetros de ponderação.
ξ_h	Coefficientes percentuais.
ε	Coefficiente de penalidade.
Δ	Função desvio.
Δf_i	Função incremento.

CAPÍTULO 4

Letras Latinas

A, B	Matriz de gradientes das restrições de igualdade.
D	Aproximação para a matriz Hessiana.
f	Função objetivo.
g	Vetor de restrições de desigualdade.
G_R	Gradiente Reduzido Generalizado.
h	Vetor de restrições de igualdade.
H	Matriz Hessiana.
p, y	Parâmetros de atualização dos métodos DFP e BFGS.
q	Contador do processo de otimização.
r	Número randômico.
S	Espaço de busca.
x, X, Z, Y	Vetor de variáveis de projeto.
x^*	Projeto ótimo.

Letras Gregas

α^*	Tamanho do passo no processo de otimização.
β^q	Parâmetro de atualização.
δ	Desvio entre iterações consecutivas.
$\bar{\delta}$	Escalar que previne inconsistências nas restrições linearizadas.
λ	Vetor de multiplicadores de Lagrange.
∇f	Gradiente da função f .
θ	Parâmetro de escolha entre os métodos DFP e BFGS.
σ, τ	Parâmetros de atualização dos métodos DFP e BFGS.

CAPÍTULO 5

Letras Latinas

B_i	Relação entre a codificação real e binária.
c	Função auxiliar para a atualização do algoritmo de ED.
c_i	Indivíduos da geração corrente.
c_j	Constantes reais positivas.
C_i	Número de cópias.
d	Densidade relativa.
d_{ij}	Distância Euclidiana entre as soluções i e j .
D	Distância entre vizinhos próximos.
E	População externa.
F	Taxa de perturbação.
F_i, F'_i	Função aptidão.
F_j	Fronteira com <i>ranking</i> j .
$gbest$	Parâmetros de atualização no método OCP.
I	Solução ordenada por meio de <i>rank</i> .
K_b	Constante de Boltzmann.
$l_{inferior}$	Limite inferior das variáveis de projeto.
$l_{superior}$	Limite superior das variáveis de projeto.
L_k	Comprimento da trajetória da k -ésima formiga.
N	Tamanho da população.
	Número de variáveis de projeto.
	Número de caracteres.
n	Número de soluções.
	Geração atual.
	Número de objetivos.
M	Tamanho da população.
m	Número de formigas.
$nMax$	Número de gerações.
nc_i	Contador de nichos.
n_{ij}	Inverso da distância Euclidiana entre as soluções i e j .
$pbest$	Parâmetros de atualização no método OCP.
	Probabilidade da solução i .
p_i	Indivíduos da geração corrente.
	Posição da partícula na iteração corrente.

p_m	Probabilidade ou taxa de mutação.
P	Probabilidade.
P_{aux}	Parâmetro do algoritmo de ED.
P, R_n	População corrente.
P_o	População inicial.
Q	População gerada na iteração corrente.
	Constante de projeto no Método da Colônia de Formigas.
Q_o	População filha inicial.
k, i, i_{rand}, s_i	Contador.
$r_i k$	<i>Ranking</i> .
r^*	<i>Ranking</i> máximo na geração corrente.
r, r_i, ud	Números randômicos.
$Ud, rand_i$	
$RC,$	Probabilidade de cruzamento.
Sh	Função de Compartilhamento.
t	Iteração.
	Contador.
t_i	Tempo.
t_{dom}	Tamanho do torneio.
T	Subconjunto da população corrente.
	Temperatura.
v_i	Velocidade da partícula na iteração corrente.
V_{max}	Velocidade máxima da partícula.
$Var(i)$	Variância da solução i .
x, x_{rj}	Vetor de variáveis de projeto.
x_{best}	Melhor indivíduo da população.
x_{old}	Indivíduo escolhido aleatoriamente na população.
x_{avg}	Indivíduo médio da população na iteração corrente.
w_j	Vetor de parâmetros de ponderação.

Letras Gregas

	Parâmetro referente a Sh .
α	Números randômicos inteiros.
	Fator de ponderação do feromônio.
β	Ponderação heurística no Método da Colônia de Formigas.
	Números randômicos inteiros.
η	Função auxiliar para a atualização do algoritmo de ED.
	Números randômicos inteiros.
γ	Parâmetro de atualização no algoritmo de ED.
ν_i	Vetor resultante no Método da Evolução Diferencial.
$\mu_{r_i} k$	Número de soluções com <i>ranking</i> r_i .
v_j	Parâmetro do algoritmo de ED.
$\mathfrak{S}_j$	Conjunto de soluções na fronteira j .
τ_i	Parâmetros definidos pelo usuário para o algoritmo de ED.
τ_{ij}	Intensidade da trilha de formigas.
σ_{share}	Parâmetro de atualização.
δ	Números randômicos inteiros.
ΔE	Variação de energia.

Δf	Desvio relativo a função objetivo em gerações consecutivas.
ρ	Números randômicos inteiros.
	Persistência da formiga na trilha corrente.
ρ_c	Probabilidade de cruzamento.

CAPÍTULO 6

Letras Latinas

$B(r_b, x)$	Função pseudo-objetivo.
d	Solução codificada.
$eval(x)$	Função de atualização.
f_i	Função objetivo.
g	Vetor de restrições de desigualdade.
h	Vetor de restrições de igualdade.
P	Função de penalidade.
	População corrente.
r, r_b, r_p	Fator de penalidade.
s	Espaço viável de busca.
T	Traçador.
x	Vetor de variáveis de projeto.
y	Indivíduo não-viável.

Letras Gregas

δ	Número real positivo.
λ	Número de filhos gerados.
λ_j	Multiplicador de Lagrange.
ψ_j	Relação de atualização para o tratamento de restrições.
$\Phi(x, r_p), \theta$	Função pseudo-objetivo.
$\mathfrak{S}, \mathfrak{S}_l$	Espaço viável de busca.

CAPÍTULO 7

Letras Latinas

d_i	Distância relativa a solução i .
D_R	Tamanho da vizinhança.
f	Vetor de objetivos.
F	Taxa de perturbação.
g	Número de gerações.
I	Solução ordenada por meio de <i>rank</i> .
L, s^l	Vetor de limites inferiores das variáveis de projeto.
<code>limit[i][j]</code>	Limite inferior (superior) das variáveis de projeto.
<code>maxvar</code>	Número máximo de variáveis de projeto.
<code>maxfunc</code>	Número máximo de funções objetivo.
<code>maxpop</code>	Número máximo de indivíduos da população.
<code>maxcons</code>	Número máximo de restrições de desigualdade.

M	Número de objetivos.
n_k, k	Contador.
n_{viol}	Número de restrições de desigualdade violadas.
N	Parâmetro referente a complexidade do MODE.
$N_{gen,}$	Número de gerações.
N_{obj}	Número de objetivos.
N_{var}	Número de variáveis de projeto.
N	Tamanho da população.
p_c	Probabilidade de cruzamento.
P	Conjunto de indivíduos na geração corrente.
q_j	Função que normaliza o vetor de variáveis de projeto.
r	Taxa de redução.
r_p	Fator de penalização em restrições de desigualdade.
R	Número de pseudo-curvas.
U, s^u	Vetor de limites superiores das variáveis de projeto.
V	Vetor de variáveis de projeto.
x_{disc}	Vetor de variáveis de projeto discretos.
x_{cont}	Vetor de variáveis de projeto contínuos.

Letras Gregas

η	Taxa de contração no MODE.
χ_k	Vetor vizinhança das variáveis de projeto.
λ	Número de filhos gerados no processo evolutivo.
$\mathfrak{S}_j$	Conjunto de soluções na fronteira i .

CAPÍTULO 8

Letras Latinas

DG	Distância geracional.
f	Vetor de objetivos.
F	Taxa de perturbação.
g	Função matemática auxiliar.
g_i	Vetor de restrições de desigualdade.
M	Número de objetivos.
n	Contador.
N	Tamanho da população.
$N_{gen,}$	Número de gerações.
p_c	Probabilidade de cruzamento.
p_m	Probabilidade de mutação.
r	Taxa de redução.
r_p	Fator de penalização em restrições de desigualdade.
R	Número de pseudo-curvas.
t_{dom}	Pressão de dominância.
x	Vetor de variáveis de projeto.
w	Função matemática auxiliar.

Letras Gregas

σ_J^2	Variância do parâmetro J .
σ_{share}	Parâmetro de Nicho.
Δ	Diversidade métrica.
Υ	Convergência métrica.

CAPÍTULO 9

Letras Latinas

a	Inclinação do secador. Área da seção transversal do secador.
A	Amplitude máxima do cilindro. Componente químico. Parâmetro da equação de Page.
A_c	Área da seção transversal.
A_{sr}	Amplitude máxima obtida pelo meta-modelo.
b	Espessura da viga. Componente químico.
B	Parâmetro da equação de Page. Vazão de fundo.
B_i	Coefficientes do problema de <i>Jacobson & Lele</i> .
c	Fator de amortecimento.
C	Componente químico. Parâmetro da equação de Page.
C_i	Concentração da espécie i .
C_{pi}	Calores específicos do componente i .
C_d	Coefficiente de arrasto.
$C_{d\ sr}$	Coefficiente de arrasto obtido pelo meta-modelo.
C_l	Coefficiente de sustentação.
$C_{l\ sr}$	Coefficiente de sustentação obtido pelo meta-modelo.
C_i	Calor específico. Coefficients do problema de <i>Jacobson & Lele</i> .
C_{pi}	Capacidade calorífica do componente i .
C_{To}	Concentração total.
C_{pg}	Calor específico do gás.
C_{ps}	Calor específico do sólido.
d	Diâmetro da viga em balanço. Diâmetro médio da espira. Diâmetro.
D	Diâmetro da mola. Vazão de destilado.
D_i	Coefficients do problema de <i>Jacobson & Lele</i> . Diâmetro da engrenagem i .
D_p	Diâmetro das partículas. Diâmetro.
e_1	Distância do centro de gravidade até o braço.

e_2	Distância do centro de gravidade até o antebraço.
E	Módulo de <i>Young</i> .
f	Vetor de objetivos.
$\bar{f}$	Valor médio da resposta f .
f_i	Resposta medida através do modelo.
f'_i	Resposta obtida via meta-modelo.
f_i^*	Valor calculado no i -ésimo ponto de projeto.
f_{Io}	Parâmetro cinético.
f_n	Frequência natural do cilindro.
	Força aplicada no problema da viga soldada.
F	Força aplicada na extremidade da viga.
	Vazão de alimentação.
F_d	Força de arrasto.
F_i	Vazão do componente i .
F_j	Fluxo molar do componente j .
F_{jo}	Fluxo molar inicial do componente j .
F_l	Força de sustentação.
F_{stat}	F estatístico.
F_T	Vazão de alimentação do reator.
	Fluxo molar total.
g_c	Fator de correção de unidades.
	Concentração de glicose.
G	Distribuição da temperatura no meio.
	Vazão mássica.
G_f	Concentração de alimentação de glicose.
G_g	Velocidade mássica superficial do gás de secagem.
G_o	Concentração inicial de glicose.
G_s	Velocidade mássica superficial do sólido.
	Entalpia.
h	Espessura da solda.
	Coefficiente global de transferência de calor.
H	Distribuição da temperatura no meio.
	Carga total do secador.
H_p	<i>Hold-up</i> de líquido.
I	Momento de inércia.
	Concentração de indutor.
$I(\mu, \tau)$	Intensidade da radiação.
I_f	Concentração de alimentação de indutor.
I_n	Momento de inércia.
I_o	Concentração inicial de indutor.
I_{total}	Quantidade máxima de indutor.
J	Função objetivo.
	Relação de equilíbrio.
k	Coefficiente de perda de calor.
	Parâmetro do problema do projeto de uma mola.
k_m	Coefficiente de perda de calor.
k_{1A}, k_{2A}	Constantes de reação.
K	Rigidez da mola.

K	Parâmetro do problema do projeto de uma mola.
K_P	Constante de equilíbrio.
$K_s, K_{sI}, K_I,$ $K_{p_1I}, K_{p_2},$ $K_{p_2I}, K'_{sI},$ $K'_{p_1I}, K''_s,$ K'_{p_2}, K'_{p_2I} $K_{CI}, K_{CP},$ $K_{CG}, K_{p_1},$ K'_{p_1}, K''_{sI}, K'_s	Parâmetros cinéticos.
l	Comprimento da viga. Comprimento da solda. Comprimento da viga em balanço. Parâmetro do problema do projeto de uma mola.
l_1	Comprimento do braço.
l_2	Comprimento do antebraço. Comprimento da viga com seção I.
L	Comprimento total do secador. Lucro. Vazão de líquido.
m	Massa do cilindro.
m^*	Coeficiente de perda de calor.
m_1	Razão de massa.
m_2	Massa do braço.
m_p	Massa do antebraço.
M	Coeficiente de perda de calor.
MR	Umidade do sólido.
$MMar$	Umidade adimensional.
M_{eq}	Massa molecular do ar.
M_Y, M_Z	Umidade de equilíbrio.
n	Momentos máximos nas direções Y e Z .
n_{tubos}	Número de experimentos. Coeficiente de perda de calor.
N	Número de tubos. Número de pontos experimentais.
N_p	Parâmetro de condução-radiação.
p	Número de molas.
p_o	Números de pontos experimentais.
p_1	Números de termos não constantes do meta-modelo.
p_2	Concentração do produto.
p_{1o}	Concentração inicial do produto.
p_{2o}	Concentração de etanol.
	Parâmetros cinéticos.
	Concentração de glicerol.
	Parâmetros cinéticos.
	Concentração inicial de etanol.
	Concentração inicial de glicerol.

$p_{1expmax}$	Concentração experimental máxima de etanol.
$p_{2expmax}$	Concentração experimental máxima de glicerol.
	Pressão.
P	Concentração de proteína.
	Parâmetro do problema do projeto de uma mola.
	Carga vertical.
P_c	Força máxima.
P_i	Pressão.
P_o	Concentração inicial de proteína.
P_o	Pressão inicial.
Pv_i	Preço de venda do componente i .
Pc_i	Preço de custo do componente i .
P_{max}	Parâmetro do problema do projeto de uma mola.
$PRESS$	Somatório dos erros preditos.
q_{p1}	Cinética de formação de produto p_1 .
q_{p2}	Cinética de formação de produto p_2 .
q_G	Taxa de alimentação de glicose.
q_I	Taxa de alimentação de indutor.
Q	Calor total trocado.
	Carga da viga.
r	Raio do secador.
r_A	Taxa de reação para o componente A .
r_i	Número randômico.
r_I	Taxa específica de inativação do indutor.
r_{p1}	Taxa de reação para etanol.
r_{p2}	Taxa de reação para glicerol.
r_P	Taxa específica de produção de proteína.
r_s	Taxa de reação para glucose.
r_x	Taxa de reação para célula.
r_{1A}, r_{2A}	Taxas das reações I e II.
r_{1max}, r_{Pmax}	Parâmetros cinéticos.
	Raio do cilindro.
R	Constante dos gases ideais.
	Vazão de refluxo.
	Constante dos gases ideais.
$RMSE$	Erro quadrático médio.
Re	Número de Reynolds.
R^2	Coefficiente de correlação.
R_{adj}^2	Coefficiente de correlação ajustado.
R_{pred}^2	Coefficiente de correlação predito.
R_w	Taxa de secagem.
	Concentração de glucose.
s	Concentração de substrato.
s_{expmax}	Concentração experimental máxima de glucose.
s_F	Concentração de substrato na corrente de alimentação.
s_I	Parâmetro cinético.
s_o	Concentração inicial de glucose.
s_o	Concentração inicial de substrato.

	Parâmetro do problema do projeto de uma mola.
S	Área superficial das partículas por unidade de volume do leito.
	Área da seção transversal do secador.
SSE	Somatório dos erros quadráticos.
SST	Somatório dos desvios quadráticos.
S_y	Pressão máxima para o problema da viga em balanço.
t	Tempo de operação do processo.
	Largura da viga.
t_f	Tempo total de operação.
ts_i	Eventos.
T	Temperatura.
T_{amb}	Temperatura ambiente.
T_i	Temperatura.
T_f	Temperatura do fluido.
T_{fo}	Temperatura inicial do fluido.
T_g	Temperatura do gás.
T_{go}	Temperatura inicial do gás.
T_o	Temperatura inicial.
T_s	Temperatura do sólido.
T_{so}	Temperatura inicial do sólido.
TR	Tempo de residência.
u	Fluxo de alimentação de substrato no reator.
	Variável de controle.
$u_{primeirafase}$	
$u_{segundafase}$	Estratégia de controle para o problema de <i>Jacobson & Lele</i> .
$u_{terceirafase}$	
u_{max}	Deslocamento máximo.
$u_{singular}$	Estratégia de controle na região singular.
U, U_a	Coeficiente de transferência de calor.
U_P	Coeficiente de perda de calor.
UR	Umidade relativa.
var	Contador.
v_∞	Velocidade do fluido.
v_3	Carregamento máximo de carga.
	Volume do reator.
V	Volume do secador.
	Vazão do fluido de aquecimento do refervedor.
V_o	Volume inicial do reator.
V_r	Velocidade reduzida.
	Coordenada cartesiana.
x	Concentração de células.
	Coordenada na direção do escoamento do gás.
x_{expmax}	Concentração experimental máxima de células.
x_o	Concentração inicial de células.
X_o	Conversão inicial.
x_i	Dimensões da viga com seção I.
x_1, x_2	Áreas da seções transversais nas barras AC e BC.

	Variáveis de estado.
x_i	Frações mássicas na fase líquida.
$x_i(0)$	Variáveis de estado definidas no tempo inicial.
x_o, x_b	Frações mássicas no refervedor.
x_n	Largura do n -enésimo componente da viga.
x_N	Fração mássica no prato N .
X	Concentração de células.
	Conversão.
X_o	Concentração inicial de células.
x_i	Variáveis de projeto codificadas.
X_1, X_2	Localização do mecanismo de mola em relação à base do robô.
X_3	Pressão no acumulador hidráulico.
X_4	Diâmetro do cilindro.
	Distância vertical no problema de duas barras.
y	Deslocamento transversal do cilindro.
y_i	Frações mássicas na fase vapor.
y_1	Torque estático na haste do motor de direção.
y_2	Trabalho executado pelo motor.
y_3	Força estática na junta A.
$y_{i\ sr}$	Resposta obtida pelo meta-modelo.
y_{sr}	Deslocamento obtido pelo meta-modelo.
Y	Rendimento de biomassa no substrato.
	Variável binária que define em qual estágio a alimentação será feita.
$Y_{p1/s}$	Fator de rendimento de etanol.
$Y_{p2/s}$	Fator de rendimento de glicerol.
Y_{qmedio}	Tempo médio de queda.
Y_{sr}	Resposta do modelo.
z	Coordenada na direção do escoamento do sólido.
z_f	Taxa de perturbação de alimentação do processo.
W	Massa de catalisador.
	Umidade absoluta do ar.

Letras Gregas

α	volatilidade relativa.
β_i	Coefficientes da aproximação polinomial.
δ	Deflexão da viga em balanço.
δ_{max}	Deflexão máxima da viga em balanço.
δ_p, δ_{pm}	Parâmetro do problema do projeto de uma mola.
ΔH_{RX}	Calor de reação.
$\Delta H_{R1}, \Delta H_{R2}$	
ϵ	Razão entre vazões.
γ_i	Coefficientes do problema de <i>Jacobson & Lele</i> .
λ	Calor latente de vaporização da água.
	Viscosidade da mistura.
	Cinética de crescimento de células.
μ	Cosseno do ângulo formado entre a direção do feixe de radiação e o eixo τ .

μ	Taxa específica de crescimento de células.
	Viscosidade cinemática.
μ_m, μ_{Pmax}	Parâmetros cinéticos.
ν_i	Coefficiente estequiométrico do componente i .
ν_{p1}, ν_{p2}	Parâmetros cinéticos.
ω	Albedo - relação entre a radiação espalhada e absorvida.
Ω	Domínio das variáveis independentes.
	Função matemática auxiliar.
Ω^{maxexp}	Máximo valor experimental da variável Ω .
π	Taxa de formação de produto.
ϕ	Esfericidade das partículas.
	Tensão máxima estabelecida.
σ	Desvio padrão.
	Taxa de consumo de substrato.
σ_A	Tensão na barra A .
σ_g	Tensão máxima permitida.
σ_{max}	Tensão normal máxima.
	Densidade da viga em balanço.
ρ	Densidade do fluido.
ρ_o	Densidade da mistura.
ρ_b	Densidade do catalisador.
	Variável óptica espacial.
τ	Tensão de cisalhamento.
τ_0	Espessura óptica do meio.
τ_{max}	Tensão de cisalhamento máximo.
τ_1, τ_2	Função tensão auxiliar.
θ	Temperatura adimensional.
θ_i	Parâmetro estequiométrico.
ε	Porosidade do leito.
ζ_{cal}, ξ_{cal}	Variáveis independentes calculadas pelo modelo.
ζ_{exp}, ξ_{exp}	Variáveis independentes obtidas experimentalmente.

LOBATO, F. S. **Otimização Multi-Objetivo para o Projeto de Sistemas de Engenharia**, 2008. Tese de Doutorado, Universidade Federal de Uberlândia.

Resumo

evido às crescentes necessidades do mercado em se atingir simultaneamente um número cada vez maior de objetivos quando do projeto dos modernos sistemas de engenharia, assim enfocando problemas mais realísticos do ponto de vista industrial, os chamados problemas de otimização multi-critérios, multi-desempenho, multi-objetivos ou otimização vetorial, têm merecido, nos últimos anos, destaque no desenvolvimento de algoritmos e *softwares* específicos para a solução destes problemas. Tais objetivos, por sua vez, são em sua maioria conflitantes, isto é, uma melhora em qualquer um destes objetivos não resulta, necessariamente, na melhora dos demais considerados. A solução ótima desses problemas, diferentemente do que ocorre na otimização com um único objetivo, consiste na obtenção de soluções não-dominadas que formam a Curva ou Fronteira de Pareto. Existem duas abordagens para a obtenção da Curva de Pareto: a Determinística, que faz uso do Cálculo Variacional e a Não Determinística, que é fundamentada nos processos de seleção natural, na genética das populações ou em metodologias puramente estruturais. O uso da Abordagem Não Determinística vem chamando a atenção nas últimas décadas, principalmente devido ao fato de que estas não fazem uso de derivadas, por serem de fácil implementação e de concepção simples. Evidentemente, o célere desenvolvimento da computação digital é um fator também determinante para o sucesso destas técnicas, uma vez que o tempo de processamento, naturalmente maior se comparado aos métodos clássicos, vem diminuindo significativamente com o melhor desempenho dos processadores. Dentre estas, o Algoritmo de Evolução Diferencial, abordagem estrutural desenvolvida inicialmente para problemas sem restrições com um único objetivo, vem se mostrando uma alternativa viável para a finalidade acima descrita. Neste sentido, este trabalho consiste na extensão do Algoritmo de Evolução Diferencial para problemas com múltiplos objetivos, através da incorporação de dois operadores ao algoritmo original: os mecanismos de ordenamento por *rank* e a exploração das vizinhanças de soluções candidatas em potencial. O algoritmo desenvolvido foi testado em uma série de problemas matemáticos clássicos e em problemas de engenharia de diversas áreas, constituindo dessa forma, uma ampla gama de estudos de caso. Os resultados obtidos se mostraram promissores, uma vez que conseguiu-se a redução do número de gerações, sem perda de qualidade na aproximação da Curva de Pareto, quando comparados com aqueles obtidos através de algoritmos evolutivos clássicos.

Palavras-chave: *Otimização Multi-objetivo, Ótimo de Pareto, Evolução Diferencial, Ordenamento por Rank, Exploração das Vizinhanças.*

LOBATO, F. S. **Multi-objective Optimization for Engineering System Design**, 2008. D. Sc. Thesis, Universidade Federal de Uberlândia.

Abstract

ue to the increasing market needs of simultaneously achieving a growing number of objectives when designing modern engineering systems, thus focusing more realistic problems from the industrial point of view, the so-called multi-criteria optimization problems, multi-objective or vectorial optimization, have deserved, recently, emphasis in the development of algorithms and specific software for the solution of these problems. Most of these objectives, in turn, are conflicting, that is, an improvement in any one of these objectives doesn't result, necessarily, in the improvement of the others. The optimal solution for these problems, unlike the optimization with an single objective, is the attainment of non-dominated solutions that form the Pareto Curve, also know as Pareto Optimal. There are two approaches for obtaining Pareto's Curve: the Deterministic one, that makes use of the Variational Calculus and the Non-Deterministic one, which is based in the natural selection processes, in population genetics or in purely structural methodologies. The use of the Non-Deterministic Approach is drawing attention in recent decades, mainly due to the fact that they do not make use of derivates, and are easy implemented and have simple conception. Evidently, the rapid development of the digital computation is also a decisive factor for the success of these techniques, since the processing time, which is greater than that of classic methods, has been decreasing significantly with the better performance of the processors. Among these, the Differential Evolution Algorithm, an structural approach developed initially for problems without restrictions with an single objective, has been shown as a viable alternative for this purpose. Therefore, this work consists in the extension of Differential Evolution Algorithm for problems with multiple objectives, through the incorporation of two operators into the original algorithm: the rank ordering and neighborhood exploration of potential candidates. The developed algorithm was tested on a series of classic mathematical and engineering problems of different areas, thus constituting a wide range of case studies. The obtained results were promising, as it is able to reduce the generation number, without loss of quality of Pareto's Curve, when compared with those obtained through classic evolutionary algorithms.

Keywords: *Multi-objective Optimization, Pareto's Optimal, Differential Evolution, Rank Ordering, Neighborhoods Exploration.*

INTRODUÇÃO

 a medida em que se busca atender às crescentes necessidades de mercado em se atingir mais objetivos simultaneamente e trabalhar com problemas mais realísticos do ponto de vista industrial, os problemas de otimização multi-objetivos (POMO), multi-desempenho, multi-critérios ou otimização vetorial, têm merecido, nos últimos anos, destaque no desenvolvimento de metodologias e aplicações práticas (DULIKRAVICH et al., 1999; DEB, 2001; BABU et al., 2005). Contudo, apesar do esforço empreendido, a complexidade dos modelos, a carência de pesquisas sistemáticas na área de otimização de projetos de engenharia no meio industrial e a falta de garantia de obtenção do ótimo global, têm limitado a exploração do potencial de diversas abordagens emergentes usadas na academia (DEB, 2001; COELHO, 2003). Ainda segundo estes autores, os dois maiores inibidores dessa aplicação são a falta de otimizadores robustos e a falta de confiança do projetista na utilização de novas metodologias.

Os modelos de engenharia são inerentemente complexos, sendo muitas vezes constituídos por sistemas de Equações Diferenciais Parciais (EDPs) ou Equações Algébrico-Diferenciais Parciais (EADPs), que representam os balanços de massa, energia e quantidade de movimento sujeitas a restrições de trajetória de igualdade e de desigualdade, além de restrições no contorno. Tais restrições são advindas de limitações físicas, de segurança, ambientais e econômicas, entre outras. Aliado a isso, a formulação desses problemas exige um conhecimento prévio de áreas distintas (multidisciplinaridade), já que na medida em que se procura obter modelos com maior representatividade dos fenômenos envolvidos, estes fazem uso, por exemplo, da utilização de hipóteses simplificadoras para sua construção. Outro ponto importante é a dificuldade na obtenção da solução analítica de modelos em geral devido à não linearidade inerente dos mesmos. Não é nenhum exagero afirmar que, em sua quase totalidade, os problemas reais não possuem solução analítica, o que constitui um grande desafio para qualquer projetista.

Um dos primeiros relatos que se tem notícia sobre o desenvolvimento de problemas de otimização foi em 1696, quando Johann Bernoulli propôs o problema de otimização dinâmica, conhecido como problema *Brachistochrone*. Este problema representava um desafio aos matemáticos europeus que deveriam apresentar a sua solução dentro de seis meses, prazo que foi estendido para um ano, atendendo a um pedido de Leibniz. O objetivo era o de encontrar a forma de um fio sem fricção que faz com que um corpo inicialmente em repouso se movimente para um ponto específico deste fio, no menor tempo possível, sob a ação da gravidade. O desafio foi entregue a Isaac Newton em vinte e nove de janeiro de 1697, que no dia seguinte, antes de sair para trabalhar, desenvolveu o Cálculo Variacional e resolveu o problema. Apesar de ter publicado a solução anonimamente, quando Bernoulli a viu teria dito: “*um leão é reconhecido pelas suas garras!*” (FEEHERRY, 1998).

Embora a história da programação matemática seja relativamente curta, de aproximadamente 60 anos, existe um grande número de algoritmos desenvolvidos para a resolução numérica de problemas de otimização (VANDERPLAATS, 1999; EDGAR et al., 2001). Todavia, a popularidade das técnicas numéricas só veio anos mais tarde, quando Schmit (1960) utilizou técnicas de otimização não linear para o projeto estrutural e cunhou o termo “síntese estrutural”. Outro aspecto importante é que, a partir do desenvolvimento do Cálculo Variacional, foi possível deduzir as condições necessárias e suficientes para a solução de problemas de otimização dinâmica.

O aumento significativo da aplicação de técnicas de otimização a problemas reais é fruto da popularidade das ferramentas computacionais disponíveis. À medida que a otimização automatiza a tomada de decisões, ela se estabelece como uma tecnologia útil e confiável para a análise de processos e projetos, fomentando novas aplicações, como por exemplo, o tratamento de problemas com restrições, não suaves e multimodais. Estas aplicações estabelecem novos desafios que, uma vez ultrapassados, tornam as estratégias de decisão mais refinadas (BIEGLER, 1984).

Quando se fala dos problemas de otimização, deve-se enfatizar as diferenças entre otimização com um único objetivo e com múltiplos objetivos. A principal meta da otimização com um único objetivo é encontrar o ótimo global que satisfaça a todas as restrições do projeto. Contudo, nos problemas multi-objetivos, não é obtido um único ponto, mas um conjunto de soluções ótimas que visam preservar a diversidade deste projeto. Além disso, este lida com dois espaços (variáveis e objetivos), o que não é trivial quando se procura manter a diversidade em ambos os espaços. A proximidade de duas soluções no espaço de variáveis não implica proximidade no espaço de objetivos. Outro ponto interessante é que, geralmente, esses objetivos são conflitantes, isto é, a melhora de um desses critérios não resulta necessariamente na melhora de qualquer um dos outros (DEB, 2001).

É notório que não existe uma definição universalmente aceita como ótimo de um problema de otimização com um único objetivo, o que dificulta a comparação de resultados de um método para outro. A extensão do conceito de ótimo para o caso multi-critérios não é trivial, pois usualmente a decisão sobre qual é a melhor resposta está vinculada ao responsável pela tomada de decisão (ZITZLER; THIELE, 1998; ZITZLER et al., 2001; DEB, 2001). A noção de “ótimo” foi inicialmente proposta por Francis Ysidro Edgeworth (EDGEWORTH, 1881), sendo posteriormente aprimorada por Vilfredo Pareto (PARETO, 1896). Sua definição é baseada na convicção intuitiva de que um ponto x^* é tomado como ótimo se “*nenhum critério utilizado pode melhorar a solução sem piorar pelo menos um outro critério*”. A partir desse conceito foi formulado o Postulado de Edgeworth-Pareto ou simplesmente, como é mais conhecido, Postulado de Pareto, que não nos fornece quase sempre, como solução do POMO, uma solução única, mas sim um conjunto de soluções denominadas não inferiores ou soluções não-dominadas.

As metodologias existentes para a resolução do POMO podem ser classificadas em Otimização Determinística (*Deterministic Optimization*), que se fundamenta no Cálculo Variacional, sendo que sua utilização se dá principalmente devido à sofisticação dos recursos computacionais desenvolvidos, e a Otimização Não Determinística ou Randômica ou Aleatória (*Random Strategies*), que fazem uso de procedimentos heurísticos, como por exemplo os processos de seleção natural da luta pela vida e a genética de populações, base dos Algoritmos Genéticos (AG). Ambas as metodologias, além de apresentarem dificuldades para a determinação do ótimo global, também demandam a análise de todas as amostras do processo a cada avaliação da função de adaptação, o que se torna um aspecto de importância relevante quando se trabalha com problemas de controle em tempo real (COELHO, 2003).

Quando se trabalha com POMO, os Métodos Determinísticos apresentam sérias limitações. Primeiramente, esses não podem encontrar múltiplas soluções em uma simples execução. Além disso, aplicações múltiplas desses métodos não garantem a obtenção de uma boa aproximação para o Ótimo de Pareto, nem tampouco diversidade nesse conjunto. Em segundo lugar, várias abordagens clássicas não podem lidar com problemas com variáveis discretas e problemas que apresentam ótimos locais (ZITZLER et al., 2001; DEB, 2001; BABU et al., 2005). Essas dificuldades fizeram com que os métodos não baseados em gradientes se tornassem populares, já que estes apresentam uma maior flexibilidade no tratamento de problemas de naturezas distintas, são de fácil concepção, entendimento e manuseio. Diferentemente da abordagem clássica, esses algoritmos trabalham com uma população de pontos (candidatos a solução), o que pode possibilitar a obtenção, em uma única execução, do Ótimo de Pareto (DEB, 2001).

A primeira implementação de um algoritmo multi-objetivo evolutivo se deu por volta

de 1980 (SCHAFFER, 1984; SCHAFFER, 1994). Desde então, uma considerável quantidade de trabalhos com esse enfoque tem sido publicada. Aliado a isso, a realização de eventos como a Conferência de Otimização Evolutiva em Problemas Multi-objetivos (*Conference on Evolutionary Multi-Criterion Optimization*) vem contribuindo com a difusão de novos conceitos e aplicações em áreas afins. Nos últimos anos, os algoritmos meta-heurísticos têm galgado espaço na comunidade científica. Dentre os mais promissores, pode-se citar: Otimização por Colônia (*Colony Optimization*) (DORIGO; STUTZLE, 2004), Sistema Imune Artificial (*Artificial Immune Systems*) (CASTRO; TIMMIS, 2002), Recozimento Simulado (*Simulated Annealing*) (AARTS; KORST, 1989) e Evolução Diferencial (*Differential Evolution*) (STORN; PRICE, 1995; ANGIRA; BABU, 2005; BABU et al., 2005).

Dentro deste contexto, o algoritmo da Evolução Diferencial (ED) (STORN; PRICE, 1995), apesar de recente, já apresenta bons resultados, tanto no que diz respeito à otimização com um, como com mais objetivos (STORN; PRICE, 1995; STORN, 1999; PAHNER; HAMEYER, 2000; FAN; LAMPINEN, 2003; KAPADI; GUDI, 2004; ANGIRA; BABU, 2005; BABU et al., 2005; LOBATO; STEFFEN JR, 2006; ANGIRA; SANTOSH, 2007; LOBATO; STEFFEN JR, 2007; LOBATO et al., 2007a; LOBATO et al., 2007b; MARIANI et al., 2008; SARAVANANA et al., 2008; ARRUDA et al., 2008; LOBATO et al., 2008b). Em sua concepção original, o valor de cada variável é representado por um valor real. O procedimento consiste em: *a*) gerar uma população inicial randomicamente; *b*) selecionar um indivíduo, de forma aleatória, para ser substituído e três outros indivíduos diferentes como genitores (pais); *c*) a partir desses genitores, selecionar um deles para ser o genitor principal; *d*) realizar a modificação pela adição do valor atual da variável a uma taxa, denominada taxa de perturbação, resultante da diferença entre dois valores desta variável nos dois outros genitores; *e*) se o vetor resultante apresenta uma função de adaptação melhor que o genitor principal, ele o substitui; caso contrário, esse vetor é mantido na população. Segundo Babu et al. (2005), a ED surge como uma estratégia que irá superar os atuais algoritmos genéticos, sendo que suas principais vantagens são: robustez, simplicidade conceitual e de fácil implementação.

Este trabalho surge no contexto do esforço de pesquisa na área de Otimização de Sistemas Mecânicos na Faculdade de Engenharia Mecânica da UFU. Com maior afinidade em relação ao estudo apresentado nesta tese, pode-se destacar o trabalho pioneiro de Butkewitsch (1998), que estudou a otimização de sistemas mecânicos, por meio de métodos clássicos, de componentes da indústria automotiva através da modelagem por elementos finitos. No mesmo ano, Braga (1998) realizou um estudo do uso de algoritmos genéticos na resolução de problemas de Engenharia Mecânica com um único objetivo. Leal (2001) estudou o comportamento dinâmico de veículos utilizando a técnica de superfície de resposta aliada aos algoritmos clássicos de otimização. A primeira pesquisa envolvendo métodos de otimização multi-objetivos foi realizada por Oliveira (2005a). Nesse trabalho, o autor faz uso de métodos clássicos para a transformação do problema multi-objetivo

em um problema com um único objetivo, para a resolução de funções matemáticas e casos de engenharia. Oliveira (2005b) apresenta em seu trabalho o desenvolvimento de uma formulação genérica para a otimização de trajetórias de estruturas paralelas. A obtenção da trajetória ótima é definida como um problema de otimização multi-objetivo, na qual considera-se a minimização da energia mecânica consumida pelos atuadores, a minimização do tempo total do percurso e da variação da aceleração. Oliveira (2006a) apresenta em seu trabalho, um estudo detalhado sobre o método da Evolução Diferencial aplicado a funções matemáticas e a dois problemas da área de robótica. Viana (2006) desenvolveu, em ambiente MATLAB®, o pacote de otimização *SimpleToolBox* (SIMPLE OPTIMIZATION TOOLBOX). Tal pacote consiste na implementação de metodologias evolutivas inspiradas na natureza para a resolução de problemas de otimização. Em uma de suas aplicações, o pacote *SimpleToolBox* foi utilizado para avaliar a rigidez torsional e a massa de um *chassis* tipo *space frame* através do método de elementos finitos (OLIVEIRA; BORGES, 2006). Recentemente, Viana (2008) combinando técnicas de otimização global e de meta-modelagem desenvolveu o *Surrogates Toolbox* em ambiente MATLAB®. Dentre as aplicações realizadas pode-se citar: a solução do hipercubo latino ótimo, a identificação de derivadas de controle e estabilidade longitudinal de aviões e modelos não lineares de trem de pouso, o melhoramento de meta-modelos via adição de simulações e a solução de problemas contínuo-discretos do projeto ótimo de uma estrutura veicular e componentes estruturais aeronáuticos.

O objetivo geral desta tese é definir procedimentos gerais que facilitem a solução do POMO. Quatro linhas básicas de atuação podem ser destacadas neste trabalho: apresentação do que há de mais recente na literatura com relação aos métodos existentes para o tratamento do POMO; desenvolvimento de um algoritmo multi-objetivo baseado no Algoritmo de Evolução Diferencial (STORN; PRICE, 1995); solução de problemas clássicos selecionados na literatura; avaliação do desempenho da metodologia proposta na solução de problemas multidisciplinares. Cabe enfatizar que, individualmente, os operadores utilizados na elaboração do algoritmo proposto nesta tese não foram desenvolvidos neste trabalho. Entretanto, o acoplamento de todas estas ferramentas constitui um algoritmo de otimização que pode ser considerado novo na medida em poucos trabalhos em termos de dissertações e teses têm sido propostos nesta direção.

Esta tese possui a estrutura conforme segue. O Capítulo 2 apresenta conceitos fundamentais sobre otimização simples e multi-objetivo, o conceito de otimalidade de Pareto e as métricas existentes para a avaliação dos algoritmos evolutivos. O Capítulo 3 traz uma revisão sobre os principais métodos existentes para o tratamento do POMO. Os Capítulos 4 e 5 apresentam uma revisão dos principais métodos para a resolução dos problemas de otimização via abordagem Determinística e Não Determinística, respectivamente. Já no Capítulo 6 são apresentados os métodos híbridos e os métodos para o tratamento de

restrições. O detalhamento do algoritmo proposto, baseado em Evolução Diferencial, é descrito no Capítulo 7. Nos Capítulos 8 e 9, são apresentadas aplicações clássicas, que abrangem problemas matemáticos e problemas de engenharia, respectivamente. As conclusões e sugestões para trabalhos futuros são descritos no Capítulo 10.

O PROBLEMA GERAL DE OTIMIZAÇÃO: MONO-OBJETIVO E MULTI-OBJETIVO



ste capítulo apresenta noções básicas sobre os problemas de otimização, levando em consideração sua formulação, as dificuldades associadas à obtenção da(s) solução(ões) e a definição das soluções não-dominadas.

2.1 Conceitos Básicos sobre Otimização

Nesta seção é apresentada a idéia geral sobre o conceito de otimização, a classificação dos métodos de otimização, o procedimento geral para resolvê-los e os obstáculos encontrados durante sua resolução.

2.1.1 Otimização: O Que é e Por que Utilizar?

Otimizar é melhorar o que já existe, projetar o novo com mais eficiência e menor custo. A otimização visa determinar a melhor configuração de projeto sem ter que testar todas as possibilidades. A otimização tem como vantagens: diminuir o tempo dedicado ao projeto, possibilitar o tratamento simultâneo de uma grande quantidade de variáveis e restrições de difícil visualização gráfica, possibilitar a obtenção de “algo melhor” com menor custo. Como limitações, tem-se o aumento do tempo computacional quando o número de variáveis de projeto cresce, o surgimento de funções descontínuas que apresentam lenta convergência, ou de funções com vários mínimos locais onde o mínimo global raramente é obtido (SARAMAGO, 1999).

2.1.2 Categorias de Otimização

Os algoritmos de otimização podem ser classificados segundo seis categorias. A Figura 2.1 ilustra esse particionamento, cujas principais características são apresentadas a seguir (HAUPT; HAUPT, 1998):

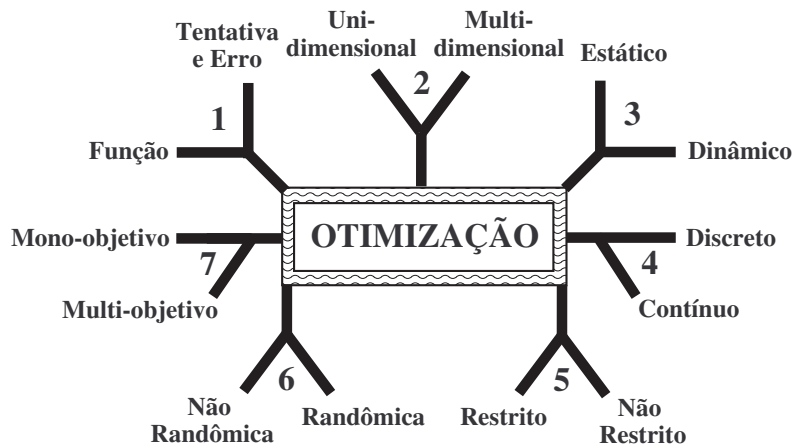


Figura 2.1: Categorias dos algoritmos de otimização (Reproduzido de Haupt e Haupt (1998)).

- ❶ A Otimização por Tentativa e Erro refere-se ao ajuste de parâmetros de um processo onde não se conhece *a priori* sua formulação, como por exemplo, o ajuste da imagem de um televisor e o descobrimento e refinamento da Penicilina como antibiótico. Por outro lado, quando se conhece o processo através de sua formulação matemática, esta é definida como Otimização por Função.
- ❷ Essa classificação leva em consideração a quantidade de variáveis do projeto. Caso exista apenas uma variável, este é denominado uni-dimensional, caso contrário, multi-dimensional.
- ❸ A Otimização Dinâmica caracteriza-se pela dependência da saída com respeito ao tempo, enquanto a Estática é independente do tempo. Por exemplo, deseja-se encontrar a melhor rota para chegar ao trabalho: se for analisada a distância, o problema é estático, e a solução pode ser obtida através da escolha do percurso por meio do mapa da cidade ou de um medidor de distância. Na prática, este problema não é simples devido à combinação de rotas. Além disso, a menor distância não é necessariamente a mais rápida, pois deve-se levar em conta a hora do dia, os acidentes, dentre outros aspectos. O problema estático já é difícil, mas a adição do tempo em sua formulação aumenta consideravelmente a dificuldade de solução de problemas dinâmicos.

- ④ O processo de otimização pode ser discreto ou contínuo. O primeiro se refere ao tratamento de problemas com um número finito de possíveis valores para as variáveis, enquanto que, no segundo, essas tendem a infinitas possibilidades. Devido à complexidade inerente aos processos reais, a modelagem é geralmente caracterizada por formulações algébrico-diferenciais. Isto faz com que tais problemas sejam tratados como um conjunto de pontos obtidos por discretização das equações originais.
- ⑤ Os problemas de otimização podem ser divididos em restritos (que apresentam restrições) e não restritos (que não apresentam nenhum tipo de restrição). Contudo, qualquer processo, por mais simples que seja, possui algum tipo de restrição: igualdade, desigualdade ou laterais, oriundas de limitações operacionais, físicas, ambientais, entre outras. Os algoritmos de otimização tradicionalmente foram concebidos para o tratamento de problemas sem restrição. Devido aos fatores descritos anteriormente, a literatura apresenta metodologias para a transformação de problemas restritos em irrestritos (EDGAR et al., 2001; VANDERPLAATS, 1999). Uma maneira simples, mas de pouca aplicabilidade, seria a mudança de variáveis para essa finalidade. Por exemplo, para minimizar $f(x)$ no intervalo $-1 \leq x \leq 1$, o problema se torna sem restrições fazendo $x = \sin(u)$. Como $\sin(u)$ é uma função limitada, o problema se reduz à minimização de $f(\sin(u))$.
- ⑥ Os algoritmos clássicos tentam minimizar uma função partindo de uma configuração inicial de projeto, movendo-se em direção ao ponto ótimo através de uma sequência de passos. Por outro lado, os métodos randômicos, que são baseados no cálculo de probabilidades, fazem uso de um conjunto de configurações iniciais de projeto para a obtenção do ótimo. Neste sentido, não investem tudo em um único ponto, ou seja, o enfoque de otimização é repartido na população de pontos.
- ⑦ O problema de otimização pode ter um objetivo (mono) ou múltiplos objetivos (multi). Esses se diferenciam pela forma como o ótimo é definido, pela metodologia de tratamento empregada, e pela presença, além do espaço de projeto, do espaço de objetivos para o problema multi-objetivo.

2.1.3 Procedimento Geral para a Solução de Problemas de Otimização

Não existe nenhum algoritmo de otimização que possa ser aplicado eficientemente a todas as classes de problemas (EDGAR et al., 2001). O método escolhido para um caso particular é fortemente dependente da natureza da função objetivo, das restrições e do número de variáveis dependentes e independentes.

A seguir são apresentados os passos gerais para a análise e solução de problemas de otimização (EDGAR et al., 2001):

- PASSO 1: Análise do problema, identificando suas variáveis e principais características (número de graus de liberdade);
- PASSO 2: Especificação do critério a ser alcançado (função objetivo em termos das variáveis definidas anteriormente);
- PASSO 3: Uso de expressões matemáticas que validam o processo e relacionam variáveis de entrada e parâmetros. Inclusão das restrições de igualdade, desigualdade e laterais;
- PASSO 4: Se o problema é complexo, pode-se tentar quebrá-lo em problemas menores ou simplificar suas equações através de hipóteses simplificadoras;
- PASSO 5: Aplicação de uma técnica de otimização conveniente;
- PASSO 6: Verificação das respostas, examinando a sensibilidade dos resultados a mudanças nos parâmetros do processo, do algoritmo utilizado e das hipóteses utilizadas na formulação do modelo.

Os passos 1, 2 e 3 tratam da formulação matemática do problema, que é a identificação das variáveis, a especificação da função objetivo e restrições. O quarto passo sugere que o problema possa ser simplificado. Inicialmente, pode-se decidir ignorar algumas variáveis que não afetam significativamente a função objetivo. Isto é baseado numa análise física, matemática ou na importância de cada variável (análise de sensibilidade). Uma outra maneira seria a eliminação de variável(is) através da utilização de restrições de igualdade. O quinto passo busca o ponto de ótimo. Geralmente a solução de problemas de otimização envolvem o uso de computadores, através de um processo iterativo. A eficiência da técnica dependerá de sua metodologia e da estimativa inicial utilizada. Já o sexto passo consiste, primeiramente, no atendimento das condições necessárias e suficientes para a obtenção do ótimo e posterior análise da sensibilidade do ótimo em relação à mudança dos parâmetros do processo, do algoritmo e das hipóteses utilizadas na formulação do problema (VANDERPLAATS, 1999; EDGAR et al., 2001).

2.1.4 Obstáculos para se Otimizar

Se a função objetivo e as restrições do problema de otimização são “*bem comportadas*” ou são funções lineares, o problema não apresenta grandes dificuldades. Entretanto, a maioria dos problemas são inerentemente não lineares.

Freqüentemente, os problemas de otimização exibem características que podem causar a falha de uma determinada metodologia. As principais são listadas abaixo (EDGAR et al., 2001):

- ❑ A função objetivo ou as restrições podem ser descontínuas;
- ❑ A função objetivo ou as restrições podem ser funções não lineares. Em problemas reais, a existência de comportamento linear nas restrições é praticamente nula. Isto não exclui o uso de aproximações lineares, mas os resultados dessas aproximações devem ser interpretados com cuidado;
- ❑ A função objetivo ou as restrições podem ser definidas por interações complexas das variáveis. Um caso desta interação é, por exemplo, a dependência entre a temperatura e a pressão no projeto de vasos de pressão. Se essa função objetivo é dada pela relação $f(x_1, x_2) = 15,5 x_1 x_2^{0,5}$, a interação entre x_1 e x_2 evita a determinação de uma única solução para essas variáveis;
- ❑ A função objetivo ou as restrições podem não ser “*bem comportadas*” para algumas faixas de valores das variáveis ou ter comportamento exponencial para outras faixas. Isto significa que o valor da função objetivo ou das restrições pode ser insensível ou muito sensível à mudança dos valores dessas variáveis;
- ❑ A função objetivo pode exibir ótimos locais. Isto aumenta consideravelmente a dificuldade durante a resolução do problema. Não existe uma abordagem que garanta fielmente que o ótimo global foi alcançado (COELHO, 2003; EDGAR et al., 2001; VANDERPLAATS, 1999).

2.2 Formulação Geral do Problema de Otimização

De forma conceitual, o problema de otimização apresenta as seguintes características (VANDERPLAATS, 1999):

- ❑ FUNÇÃO OBJETIVO $\Rightarrow$ define a característica do sistema que se deseja melhorar. Matematicamente, esta característica é representada por uma equação matemática dependente (explicitamente ou não) das variáveis de projeto;
- ❑ VARIÁVEIS DE PROJETO $\Rightarrow$ conjunto de parâmetros que podem influenciar os valores da função objetivo. Devidamente manipuladas, promovem modificações no sentido de aumentar ou diminuir os valores da função objetivo. São também denominadas variáveis de decisão ou de busca;

□ RESTRIÇÕES $\Rightarrow$ são características que dependem matematicamente das variáveis de projeto e limitam os valores da função objetivo a certas regiões do espaço de projeto. Estas podem ser classificadas em:

- Restrições de Desigualdade: estabelecem uma região do espaço de projeto dentro da qual o seu valor deve ser maior ou igual/menor ou igual, a um valor pré-estabelecido;
- Restrições de Igualdade: definem uma região onde as variáveis de projeto conferem à restrição um valor pré-determinado;
- Restrições Laterais: delimitam uma faixa de variação para cada variável de projeto, ou seja, definem os valores máximos e mínimos que podem ser adotados para tais variáveis.

Matematicamente, o Problema de Otimização Multi-objetivo (POMO) é definido como (DEB, 2001):

$$\min f_m(x), \quad m = 1, 2, \dots, M \quad (2.1)$$

$$\begin{cases} g_j(x) \leq 0 & j = 1, 2, \dots, J \\ h_k(x) = 0 & k = 1, 2, \dots, K \\ x_i^L \leq x_i \leq x_i^U & i = 1, 2, \dots, n \end{cases} \quad (2.2)$$

onde x é o vetor de n variáveis de projeto $x = (x_1, x_2, \dots, x_n)^T$. Os valores de x_i^L e x_i^U , representam os valores mínimo e máximo para a variável x_i , respectivamente. As J desigualdades (g_j) e as K igualdades (h_k) são chamadas de funções de restrição.

Cada uma das M funções objetivo $f_1(x), f_2(x), \dots, f_m(x)$ pode ser maximizada ou minimizada. O vetor de funções objetivo $f(x)$ forma um espaço multi-dimensional denominado espaço de objetivos Z .

Uma vez formulado o problema geral de otimização, faz-se necessário o entendimento de conceitos, que serão inseridos ao longo deste capítulo, e que são de suma importância para a compreensão do enfoque multi-objetivo e de suas estratégias de solução.

Definição 1: No contexto multi-objetivo, define-se Solução Ideal como sendo o valor ótimo de cada objetivo individualmente, levando em consideração todas as restrições do problema original.

2.3 O Problema de Otimização Multi-objetivo

Vários aspectos têm que ser levados em consideração quando da formulação de um problema de projeto como um problema de otimização. Primeiramente, tem-se que considerar as propriedades do sistema, isto é, quais os parâmetros que se deseja obter e como eles poderão ser medidos. Além disso, deve-se considerar quais são as variáveis de projeto que podem ser manipuladas para obter o melhor projeto. Finalmente, como decidir qual será o melhor projeto (ANDERSON, 2001).

Outro aspecto que faz com que muitas vezes sejam feitas simplificações no problema original é que a extensão do conceito de otimização com um único objetivo para o caso multi-critérios não é trivial, pois usualmente a decisão sobre qual é a melhor resposta está vinculada ao responsável pela tomada de decisão, o que dificulta a comparação de resultados de um método para outro (DEB, 2001). A Figura 2.2 exemplifica a dificuldade associada à escolha de um ótimo, pois esta depende de critérios como viabilidade econômica, limitações físicas, entre outros.

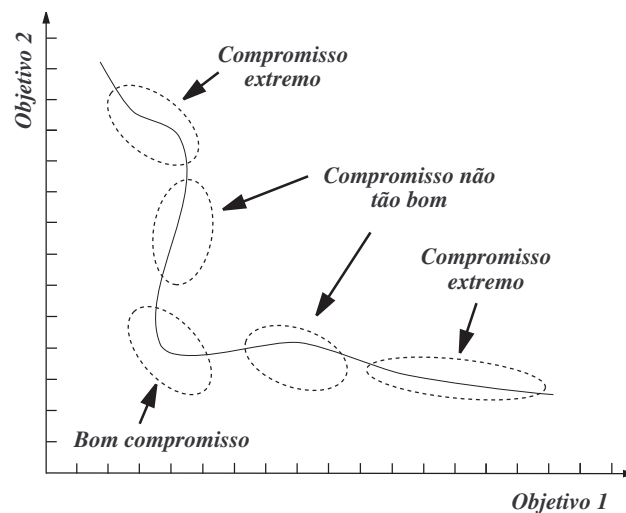


Figura 2.2: Otimização de compromisso para a minimização de dois objetivos (Reproduzido de Coelho (2003)).

2.3.1 Mono-objetivo versus Multi-objetivo

As principais diferenças entre Mono-objetivo e Multi-objetivo são descritas a seguir (DEB, 2001):

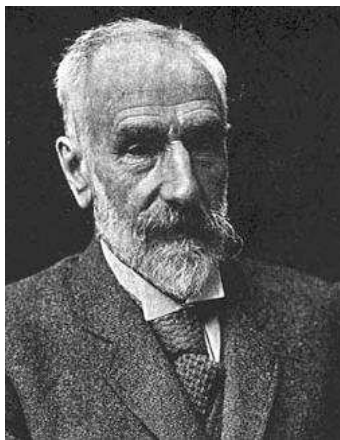
- Em problemas com um único objetivo, a meta é obter uma única solução global (máximo ou mínimo). Já em problemas multi-objetivos, é obtido um conjunto de

soluções ótimas que deve preservar a diversidade do conjunto solução no espaço dos objetivos;

- Um problema multi-objetivo trabalha com dois espaços (variáveis e objetivos) no lugar de apenas um, como no caso de um único objetivo. Manter a diversidade faz com que o problema se torne mais complicado, dado que a proximidade de duas soluções no espaço de variáveis não implica, necessariamente, em proximidade no espaço dos objetivos.

2.4 Ótimo de Pareto

Em contraste com a otimização de um único objetivo, a solução de um problema com múltiplos objetivos é mais um conceito do que uma definição (MARLER; ARORA, 2004). A noção de “ótimo” foi inicialmente proposta por Francis Ysidro Edgeworth (Fig. 2.3(a)) em 1881 (EDGEWORTH, 1881). O aperfeiçoamento dessa idéia foi apresentado pelo economista e socialista Vilfredo Pareto (PARETO, 1896) (Fig. 2.3(b)).



(a) Francis Y. Edgeworth.



(b) Vilfredo Pareto.

Figura 2.3: Idealizadores do conceito de ótimo para o problema multi-objetivo (Reproduzido de <http://www.lania.mx/~ccoello/EMOO> (acessado em 15 abril de 2005)).

A definição de ótimo segundo Edgeworth-Pareto é baseada na convicção intuitiva de que um ponto x^* é tomado como ótimo se “*nenhum critério utilizado pode melhorar a solução, sem piorar pelo menos um outro critério*”. Todavia, o ótimo de Edgeworth-Pareto quase sempre não nos fornece uma solução única, mas sim um conjunto de soluções denominadas não inferiores ou soluções não-dominadas. O Postulado de Edgeworth-Pareto ou Postulado de Pareto, como é mais conhecido, tem sido base para o desenvolvimento de

teoremas importantes na teoria de otimização multi-critérios (ESCHENAUER et al., 1990; OSYCZKA, 1984).

Para se fixar o conceito de dominância, considera-se o seguinte exemplo descrito em Ticona (2003): na hora de comprar um carro, suponha-se que se está procurando por preço e conforto. A Figura 2.4 ilustra várias alternativas de escolha.

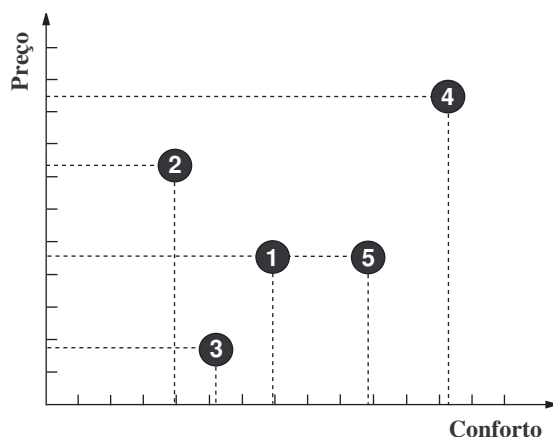


Figura 2.4: Opções de compra de um carro: custo versus conforto (Reproduzido de Ticona (2003)).

O objetivo é minimizar o custo e maximizar o conforto. Neste caso, tem-se cinco possíveis opções de compra. Intuitivamente, descarta-se a solução 1, já que a solução 5 fornece mais conforto por igual preço. A solução 2 é descartada pela mesma razão. Tem-se então três soluções: 3, 4, 5, que são boas alternativas de compra. Em termos quantitativos, nenhuma é melhor que a outra, pois uma é mais confortável, mas menos barata, e vice-versa. Existe então um “*compromisso*” entre os objetivos. Quanto maior o conforto, maior o preço do automóvel e vice-versa. Diz-se que uma solução domina uma outra se seus valores são melhores em todos os objetivos. Por exemplo, a solução 5 domina a solução 1. Então a solução 5 não é dominada por nenhuma outra. O mesmo acontece com as soluções 3 e 4. Se não se conhece *a priori* a importância relativa de cada objetivo, pode-se dizer que as soluções 3, 4, e 5 são igualmente boas. Portanto, existe um conjunto de soluções ótimas, sendo este conjunto chamado de conjunto não-dominado. As outras soluções (1 e 2) formam o conjunto dominado.

Estes conjuntos têm as seguintes propriedades (DEB, 2001):

- Qualquer solução do conjunto não-dominado deve ser não-dominado em relação à outra solução desse mesmo conjunto;
- Qualquer solução não contida no conjunto não-dominado deve ser dominado, no mínimo, por uma solução do conjunto não-dominado.

2.4.1 Operador de Dominância de Pareto

A maioria dos algoritmos para otimização multi-objetivo usam o conceito de dominância. Se existem M funções objetivo f_j , com $j = 1, \dots, M$, o operador $\triangleleft$ entre duas soluções, $x_1 \triangleleft x_2$, significa que a solução x_1 é melhor que x_2 para um objetivo em particular. Reciprocamente, $x_1 \triangleright x_2$ denota que a solução x_1 é pior que x_2 para algum objetivo. Já o operador $\not\triangleright$ denota a negação para $\triangleright$.

Definição 2: Uma solução x_1 domina uma outra solução x_2 (representado como $x_1 \preceq x_2$) se as seguintes condições são satisfeitas:

- A solução x_1 não é pior que x_2 para todos os objetivos, ou seja $f_j(x_1) \not\triangleright f_j(x_2)$ para todo $j = 1, 2, \dots, M$;
- A solução x_1 é estritamente melhor que x_2 pelo menos em um objetivo, ou seja $f_j(x_1) \triangleleft f_j(x_2)$ pelo menos para um valor de $j = 1, 2, \dots, M$.

Se ambas as condições são satisfeitas, pode-se dizer que x_2 é dominada por x_1 , x_1 é não-dominada por x_2 e x_1 é não-dominada com relação a x_2 .

2.4.2 Otimalidade de Pareto

Quando o conjunto de soluções é finito, é possível fazer comparação entre as soluções duas a duas e pode-se dividir o conjunto em soluções dominadas e não-dominadas. Portanto, tem-se um conjunto não-dominado e um conjunto dominado.

Definição 3: Dado o conjunto de soluções P^1 , o conjunto não-dominado P^2 é formado por aquelas soluções que são não-dominadas por qualquer elemento de P^1 .

Quando o conjunto P^1 é o espaço completo de busca ($P^1 = S$), o conjunto não-dominado P^2 é chamado de conjunto ótimo de Pareto. A Figura 2.5 mostra exemplos de conjuntos ótimos de Pareto, conforme várias combinações de objetivos para as funções f_1 e f_2 . A curva cheia indica onde o conjunto está localizado. Então, é possível ter conjuntos ótimos de Pareto formados por uma região contínua ou pela união de regiões descontínuas.

Definição 4: O conjunto não dominado para a totalidade do espaço de busca factível S , é chamado de conjunto ótimo de Pareto global.

Portanto, as soluções contidas neste conjunto são as soluções ótimas do POMO.

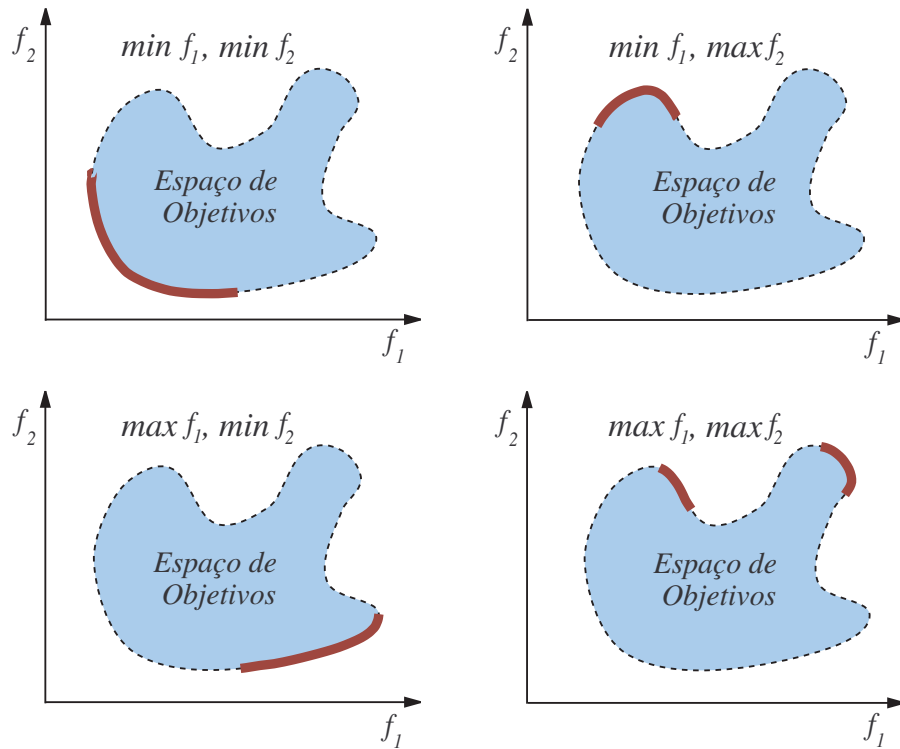


Figura 2.5: Exemplos de conjuntos ótimos de Pareto (Reproduzido de Deb (2001)).

Definição 5: Se cada elemento x_1 do conjunto P^1 não é dominado por alguma solução x_2 na vizinhança de x_1 tal que $\|x_2 - x_1\|_\infty \leq \epsilon$, onde ϵ é um número positivo arbitrariamente pequeno, então o conjunto P é chamado de conjunto ótimo de Pareto local. A Figura 2.6 mostra dois conjuntos ótimos de Pareto locais que são não-dominados, mostrando a sua vizinhança no seu espaço de objetivos e no espaço de variáveis (à direita).

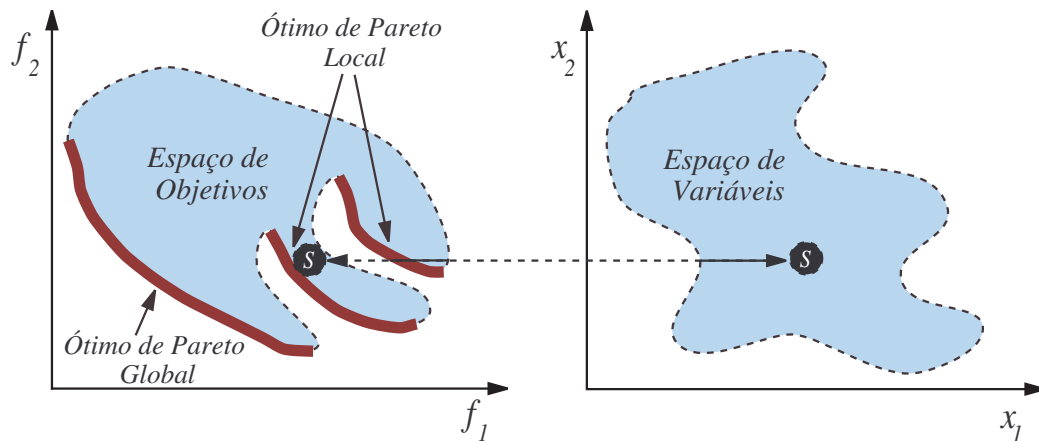


Figura 2.6: Soluções ótimas de Pareto locais e globais (Reproduzido de Deb (2001)).

Definição 6: A Fronteira de Pareto é formada pelo conjunto de vetores de funções objetivo $f(x) = (f_1(x), f_2(x), \dots, f_M(x))^T$, para cada solução x que está no conjunto ótimo de Pareto.

A Fronteira de Pareto é formada então por valores das funções objetivo (ponto no espaço de objetivos) correspondentes a cada solução no espaço de busca. A relação de dominância também pode ser classificada em dominância forte e fraca. A dominância forte é definida como:

Definição 7: A solução x_1 domina fortemente a solução x_2 (representado como $x_1 \prec x_2$) se é estritamente melhor que a solução x_2 para todos os M objetivos. Se os pontos não-dominados estão em um espaço contínuo, pode-se desenhar uma curva. Todos os pontos contidos na curva formam a Frente de Pareto ou Fronteira de Pareto.

2.5 Metas em Otimização Multi-objetivos

Quando quaisquer informações adicionais sobre o projeto analisado são desconhecidas, como por exemplo, a ordem de prioridade de atendimento dos objetivos, todas as soluções ótimas de Pareto são igualmente importantes. Devido a esse fator, é de suma importância que se tenha um critério que garanta a qualidade da solução encontrada. A seguir são apresentadas as metas em otimização multi-objetivos (DEB, 2001):

- Encontrar um conjunto de soluções o mais próximo possível da Fronteira de Pareto;
- Encontrar um conjunto de soluções com a maior diversidade possível.

A primeira meta é comum para qualquer processo de otimização. Soluções muito distantes da Fronteira de Pareto não são desejáveis. Contudo, encontrar uma maior diversidade dentro das soluções é uma meta específica para Otimização Multi-Objetivo. A Figura 2.7(a) mostra uma boa distribuição de soluções na fronteira de Pareto, enquanto que, na Fig. 2.7(b), as soluções estão distribuídas apenas em algumas regiões. É necessário assegurar a maior cobertura possível da fronteira, já que esta característica indica um bom conjunto de soluções “*comprometidas*” com os objetivos desejados.

2.5.1 Métricas de Desempenho

Nesta seção serão apresentadas algumas métricas para avaliar o desempenho dos algoritmos evolutivos. Para o caso da otimização multi-objetivo, medir a qualidade é mais

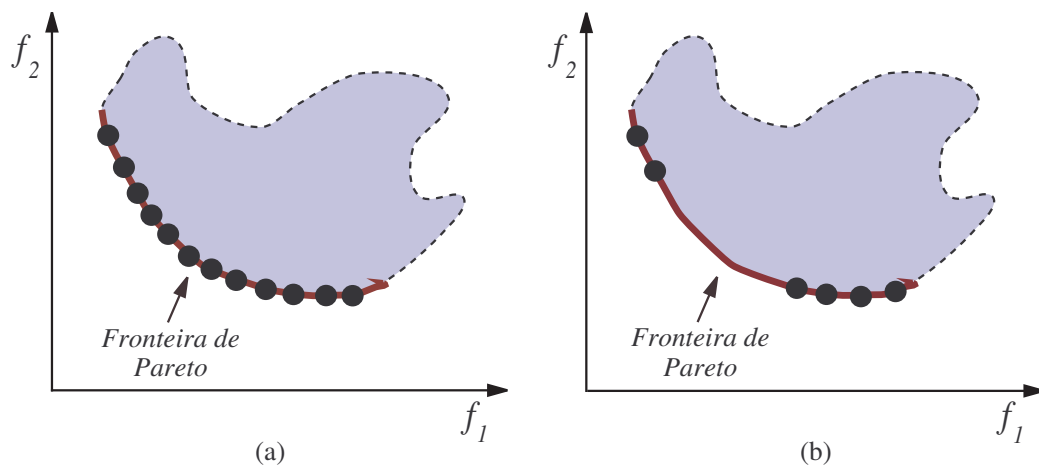


Figura 2.7: Distribuição de soluções na fronteira de Pareto (Reproduzido de Deb (2001)).

complexo que no caso da otimização de um único objetivo (ZITZLER et al., 2001). Conforme mencionado anteriormente, as metas da Otimização Multi-Objetivo são: encontrar soluções o mais perto possível da Fronteira de Pareto e obter a maior diversidade de soluções na fronteira.

A Figura 2.8 ilustra ambas as metas. É necessário notar que convergência e diversidade podem ser metas conflitantes, portanto usar apenas uma das métricas não avalia completamente o desempenho de um algoritmo (DEB, 2001).

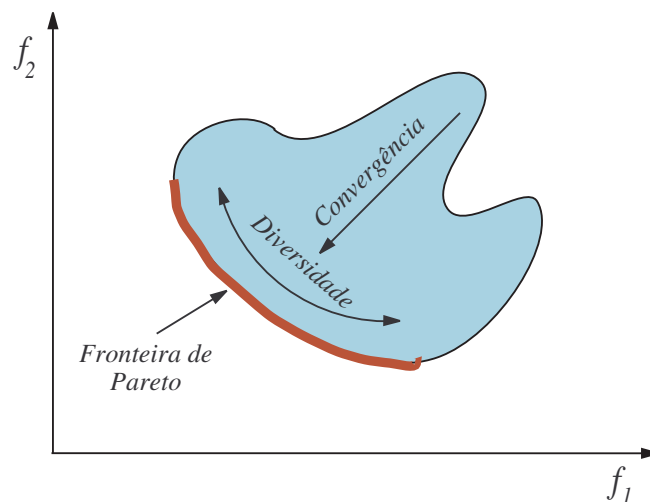


Figura 2.8: Métricas de desempenho em algoritmos genéticos (Reproduzido de Deb (2001)).

A Figura 2.9(a) ilustra um caso hipotético onde os resultados do algoritmo A têm boa convergência e pouca diversidade, ao contrário do algoritmo B (Fig. 2.9(b)). Esta

situação implica que um algoritmo não é superior ao outro com relação aos critérios de convergência e diversidade. Então, são necessárias pelo menos duas métricas, uma para medir a convergência e outra para avaliar a diversidade.

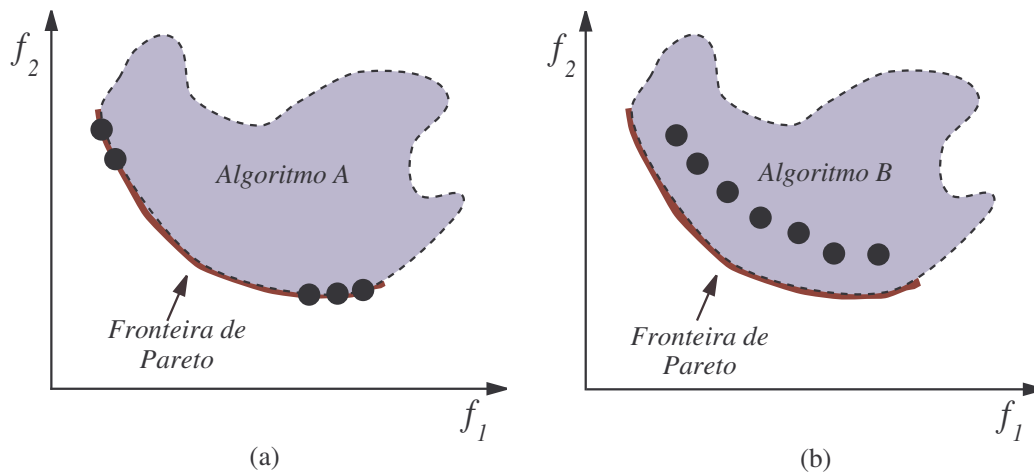


Figura 2.9: Distribuição versus convergência na fronteira de Pareto (Reproduzido de Deb (2001)).

A Figura 2.10 ilustra outras situações. No primeiro caso (Fig. 2.10(a)), o algoritmo A é melhor que o algoritmo B. No segundo caso (Fig. 2.10(b)), é difícil determinar qual algoritmo tem melhor desempenho.

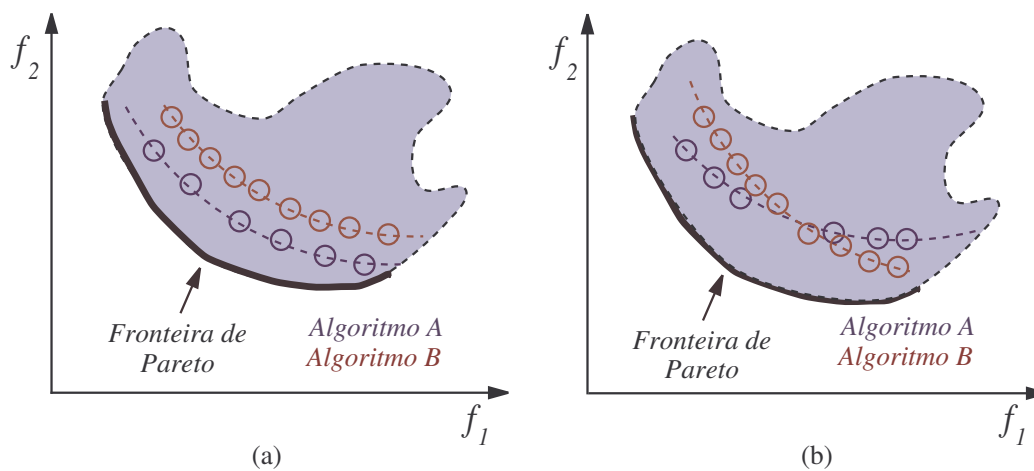


Figura 2.10: Comparação entre dois algoritmos hipotéticos (Reproduzido de Deb (2001)).

2.5.2 Métricas de Convergência

Essas métricas calculam a distância do conjunto de N soluções Q ao conjunto de soluções ótimas de Pareto P^* .

◦ Taxa de Erro (ER)

Esta métrica conta as soluções de Q que não estão no conjunto de soluções ótimas de Pareto P^* , formalmente (DEB, 2001):

$$ER = \frac{\sum_{i=1}^{|Q|} e_i}{|Q|} \quad (2.3)$$

onde $e_i = 0$ se $i \in P^*$, e $e_i = 1$, caso contrário. Quanto menor for o valor de ER , melhor será a convergência. Se $ER = 0$, significa que $Q \subseteq P^*$.

◦ Convergência Métrica (Υ)

Esta calcula a distância entre a solução não-dominada Q obtida e a solução ótima de Pareto da seguinte forma (DEB, 2001):

$$\Upsilon = \frac{\sum_{i=1}^{|Q|} d_i}{|Q|} \quad (2.4)$$

onde d_i é a distância Euclidiana (no espaço dos objetivos) entre a solução $i \in Q$ e membro mais próximo de P^* .

◦ Distância Geracional (DG)

É a distância média entre os conjuntos Q e P^* (DEB, 2001):

$$DG = \frac{(\sum_{i=1}^{|Q|} d_i^p)^{1/p}}{|Q|} \quad (2.5)$$

Para $p = 2$, d_i é a distância euclidiana no espaço de objetivos entre a solução $i \in Q$ e o elemento $p \in P^*$ mais próximo. Formalmente:

$$d_i = \min_{k=1}^{|P^*|} \sqrt{\sum_{m=1}^M (f_m^i - f_m^{*k})^2} \quad (2.6)$$

onde f_m^{*k} é o valor do m -ésimo objetivo da k -ésima solução de P^* . Quanto menor for DG melhor a convergência de Q .

◦ Espaçamento (Spc)

Esta métrica calcula o desvio padrão entre as distâncias de soluções consecutivas (ZITZLER et al., 2000). Formalmente, tem-se:

$$Spc = \sqrt{\frac{1}{|Q|} \sum_{i=1}^{|Q|} (d_i - \bar{d})^2} \quad (2.7)$$

onde

$$d_i = \min_{k \in Q, k \neq i} \sum_{m=1}^M |f_m^i - f_m^k| \quad (2.8)$$

$$\bar{d} = \frac{\sum_{i=1}^{|Q|} d_i}{|Q|} \quad (2.9)$$

A variável d_i representa o valor absoluto da diferença dos valores das funções objetivo entre a solução i e qualquer outra solução em Q e $\bar{d}$ representa a média dos valores d_i . Quanto menor for o desvio Spc , melhor distribuídas estarão as soluções.

◦ Número de Nichos (NC)

Esta métrica calcula o número de nichos dentro de um conjunto de soluções Q (ZITZLER et al., 2000).

$$NC = \frac{1}{|Q| - 1} \sum_{i=1}^{|Q|} |j \in Q, d_{ij} > \sigma| \quad (2.10)$$

sendo d_{ij} a distância entre as soluções i e j de Q . Esta distância pode ser calculada no espaço de variáveis ou de objetivos. O valor NC representa o número de soluções cuja distância entre elas é maior que o parâmetro σ . Quando $d_{ij} < \sigma$, as soluções i e j estão no mesmo nicho. Quanto maior a quantidade de nichos formados em Q , melhor distribuídas estão as soluções.

◦ Espalhamento Máximo ($M3$)

Esta métrica retorna a distribuição das soluções em Q (ZITZLER et al., 2000). Formalmente temos:

$$M3 = \sqrt{\sum_{m=1}^M \left(\max_{i=1}^{|Q|} f_m^i - \min_{i=1}^{|Q|} f_m^i \right)^2} \quad (2.11)$$

Um maior valor para $M3$ significa uma melhor cobertura do espaço (neste caso, espaço de objetivos).

o Diversidade Métrica (Δ)

Esta consiste em medir o espalhamento das soluções no domínio de objetivos da seguinte forma (DEB et al., 2000):

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{|Q|-1} |d_i - \bar{d}|}{d_f + d_l + (|Q| - 1)\bar{d}} \quad (2.12)$$

onde d_i é distância Euclidiana (no espaço de objetivos) entre a solução $i \in Q$ e o membro mais próximo de P^* , $\bar{d}$ é a média dessas distâncias. Os parâmetros d_f e d_l representam a distância Euclidiana entre as soluções extremas do conjunto ótimo de Pareto P^* e as fronteiras das soluções não-dominadas (Q) obtidas.

2.6 Metodologias para a Resolução do POMO

Na literatura podem ser encontradas diferentes metodologias para o tratamento de POMO. A classificação destas metodologias depende do tipo de algoritmo (randômico ou não-randômico) e da forma de tratamento (agregação de funções objetivo, abordagem não baseada no critério de dominância de Pareto, e abordagem baseada no critério de dominância de Pareto) adotada. As seções 2.6.1 e 2.6.2 apresentam os princípios fundamentais destas abordagens.

2.6.1 Quanto ao Tipo de Abordagem

Segundo o tipo de abordagem, os métodos para a solução dos problemas de otimização podem ser classificados em (SARAMAGO, 1999):

- Otimização Determinística (*Deterministic Optimization*) ou Clássica: são técnicas baseadas no cálculo, de ampla aplicabilidade e estudo na literatura. Sua utilização se dá principalmente devido à sofisticação dos recursos computacionais desenvolvidos nos últimos anos, aliado ao desenvolvimento do Cálculo Variacional. Entretanto, estas técnicas podem apresentar dificuldades numéricas e problemas de robustez relacionados com a falta de continuidade das funções a serem otimizadas ou de suas restrições, funções não convexas, multi-modalidade, existência de ruídos nas funções, necessidade de se trabalhar com valores discretos para as variáveis e existência de mínimos ou máximos locais;

- Otimização Randômica (*Random Strategies*) ou Não Determinística: são técnicas de busca baseadas nos processos de seleção natural da luta pela vida e da genética de populações ou em abordagens puramente estruturais. Tratando-se de um método pseudo-aleatório, pode-se dizer que o método estabelece um procedimento de exploração inteligente no espaço de parâmetros codificados (BRAGA, 1998). O interesse por tais métodos se deu por volta de 1950 com o surgimento dos algoritmos genéticos, quando biólogos começaram a utilizar técnicas computacionais para a simulação de sistemas biológicos. Deve-se ressaltar que geralmente esses algoritmos diferem de outras técnicas de minimização, especialmente pela sua simplicidade conceitual, e por dispensarem o uso de derivadas da função custo e das restrições para determinar a direção de busca. Outro aspecto importante é que, geralmente, não investem todo o esforço computacional num único ponto, mas sim operam sobre uma população de pontos, sendo portanto capazes de lidar com problemas multi-modais. Contudo, como esses métodos são estocásticos, seu desempenho varia de execução para execução (a menos que o mesmo gerador de números aleatórios com a mesma semente seja utilizado) e o número de avaliações da função objetivo é sensivelmente superior aos métodos clássicos (SARAMAGO, 1999; COELHO, 2003).

2.6.2 Quanto à Forma de Tratamento do Problema

Quanto à forma de tratamento, os POMO podem ser divididos em três categorias principais (COELHO, 2004):

- Métodos *A Posteriori*: utilizados quando a Curva de Pareto, que é o conjunto das soluções não-dominadas do espaço de funções objetivo, é completamente determinada. Esta categoria foi introduzida por Goldberg (1989) através do conceito de dominância por *ranking* de Pareto associado a algum tipo de mecanismo de nicho, para superar as limitações do VEGA (*Vector Evaluated Genetic Algorithm*) (SCHAFFER, 1984).
- Métodos Progressivos: utilizados durante o processo de otimização. Na literatura, podem ser encontradas abordagens que não fazem uso de agregação de funções ou do conceito de otimalidade de Pareto (SCHAFFER, 1984; ZITZLER et al., 2001; DEB, 2001). Tais técnicas são computacionalmente eficientes e de fácil implementação (COELHO, 2004). As suas principais características serão apresentadas no Capítulo 5.
- Métodos *A Priori*: utilizados antes da inicialização do processo de otimização, onde o usuário atribui ponderações aos critérios ou pelo menos hierarquiza os m objetivos.

Esta metodologia é a mais simples e mais óbvia entre as abordagens para o tratamento de problemas multi-objetivos. O problema, inicialmente com m objetivos, é transformado em um outro, com os objetivos agregados, resultando num problema com um único objetivo. Essas abordagens de agregação de funções serão detalhadas no Capítulo 3.

TÉCNICAS DE OTIMIZAÇÃO MULTI-OBJETIVOS

3.1 Aspectos Gerais

 notório que as primeiras técnicas usadas no contexto multi-objetivo fazem uso da transformação do problema original em um problema com um único objetivo. Dessa maneira, mediante a utilização de Métodos Determinísticos ou de Métodos Não Determinísticos, se constrói a curva ou fronteira de Pareto. Cabe ressaltar que, a extensão dos Métodos Não Determinísticos para o enfoque multi-objetivo pode permitir que, em uma única execução, seja determinado a curva de Pareto, o que não acontece com o uso de Métodos Determinísticos.

Neste capítulo serão apresentadas as principais técnicas desenvolvidas para a transformação do problema multi-objetivo em um problema com um único objetivo. Já nos dois próximos capítulos, de posse das técnicas aqui consideradas, serão estudadas as metodologias para a resolução do POMO via Métodos Determinísticos e Não Determinísticos, sendo que, para essa última classe, será apresentada a sua extensão para o contexto multi-objetivo.

3.2 Métodos para o Tratamento de POMO

Originalmente, como se pode observar no fluxograma da Fig. 3.1, a metodologia para a resolução de POMO, sem o uso do critério de dominância visto no capítulo anterior, consiste nas seguintes etapas:

- transformação do problema multi-objetivo original em um problema com um único objetivo, utilizando os métodos que serão abordados neste capítulo;
- utilização de uma técnica específica para o tratamento de restrições, que será visto posteriormente no Capítulo 6;
- de posse dessa reformulação do problema, resolvê-lo n vezes através dos métodos propostos nos Capítulos 4 e 5, buscando obter a Curva ou Fronteira de Pareto.

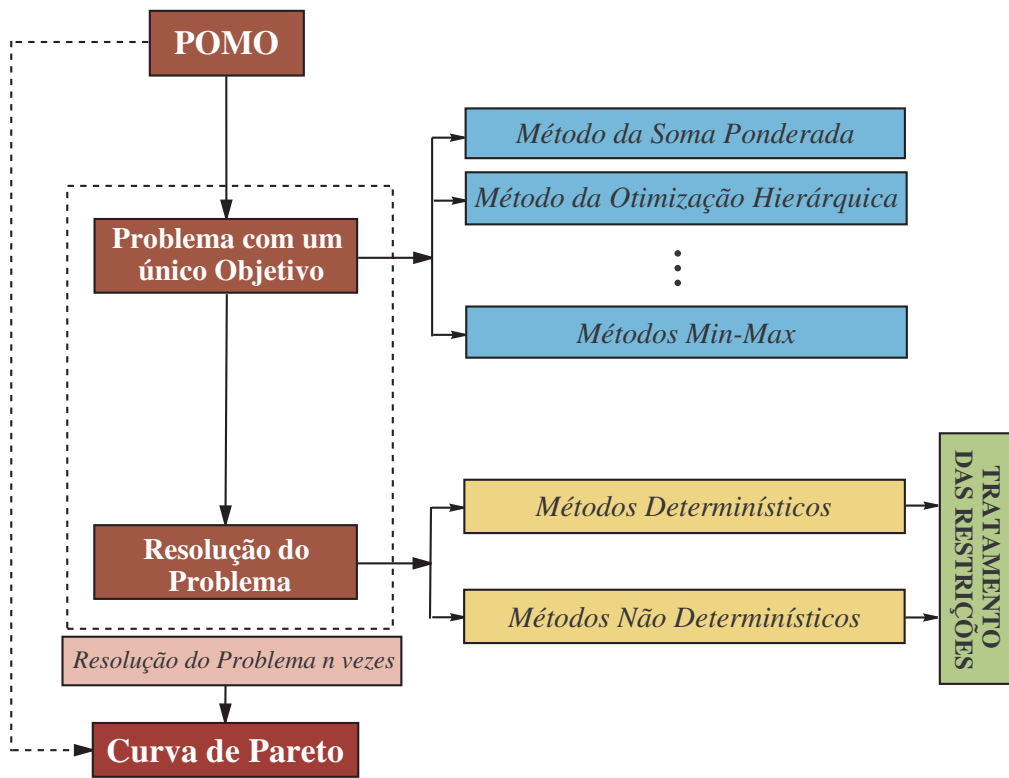


Figura 3.1: Estrutura geral para a resolução de POMO sem o uso do critério de dominância.

A formulação para POMO foi apresentada no Capítulo 2. Como o enfoque deste capítulo é o estudo das técnicas clássicas para transformação do problema original em um problema com um único objetivo, neste caso a função objetivo passa a ser uma função escalar e não mais vetorial, como mostrado a seguir:

$$\min f(x) \quad (3.1)$$

$$\begin{cases} g_j(x) \leq 0 & j = 1, 2, \dots, J \\ h_k(x) = 0 & k = 1, 2, \dots, K \\ x_i^L \leq x_i \leq x_i^U & i = 1, 2, \dots, n \end{cases} \quad (3.2)$$

A solução do problema de otimização apresentado consiste na determinação dos valores das variáveis de projeto para os quais a função objetivo atinge um valor mínimo ou máximo, satisfazendo as restrições impostas pela formulação.

Dentre as abordagens apresentadas na literatura, pode-se observar a tendência no manuseio de metodologias que são baseadas nos Métodos *a Priori*. Nessas metodologias, é necessário que o usuário especifique preferências, que podem ser articuladas em termos de metas ou da importância relativa dos diferentes objetivos (MARLER; ARORA, 2004). Vários desses métodos incorporam parâmetros, que são coeficientes, expoentes, limites constantes para as m funções objetivo, entre outros, o que implica forte dependência dos resultados com relação a esses parâmetros (OSYCZKA, 1978; OSYCZKA, 1984). Os principais métodos que fazem uso destas características são descritos a seguir (OSYCZKA, 1978; OSYCZKA, 1984; MARLER; ARORA, 2004).

3.2.1 Método da Soma Ponderada (*Weighted Sum Method*)

Este é sem dúvida o mais popular entre todos os métodos *a Priori*. Consiste em transformar a formulação original dos objetivos, expressas em uma forma vetorial, em uma escalar, através da agregação das m funções objetivo, como apresentado a seguir:

$$\min f(x) = \sum_{i=1}^m w_i f_i(x) \quad (3.3)$$

onde os pesos obedecem à seguinte relação:

$$\sum_{i=1}^m w_i = 1 \quad (3.4)$$

Os resultados obtidos através desta abordagem são fortemente dependentes dos valores de w_i , além disso, esses coeficientes não refletem proporcionalmente a importância relativa dos objetivos, são somente ponderações (OSYCZKA, 1984). Para os problemas de otimização, a localização do mínimo depende não apenas dos valores de w_i , mas também das unidades nas quais essas funções são expressas. Caso se deseje que esses pesos reflitam a importância de cada objetivo, todas as funções devem ser expressas em unidades que assumem aproximadamente o mesmo valor numérico (OSYCZKA, 1978). Isso pode ser feito da seguinte forma:

$$\min f^*(x) = \sum_{i=1}^m w_i f_i(x) c_i \quad (3.5)$$

onde c_i são constantes que normalizam a função objetivo e que, segundo Osyczka (1984), apresentam os melhores resultados quando $c_i = 1/f_i^\circ$, onde f_i° é o i -ésimo valor da função objetivo otimizada individualmente com a presença de todas as restrições originais.

- *Interpretação Geométrica do Método da Soma Ponderada*

O Método da Soma Ponderada é uma abordagem desenvolvida para o tratamento de problemas convexos. Seu uso para problemas não-convexos ou côncavos não garante que uma solução possa ser obtida e que mesmo variando w_i não existe nenhuma garantia que a curva de Pareto será bem distribuída (OSYCHKA, 1984).

Considere o problema de otimização com dois objetivos apresentado na Fig. 3.2.

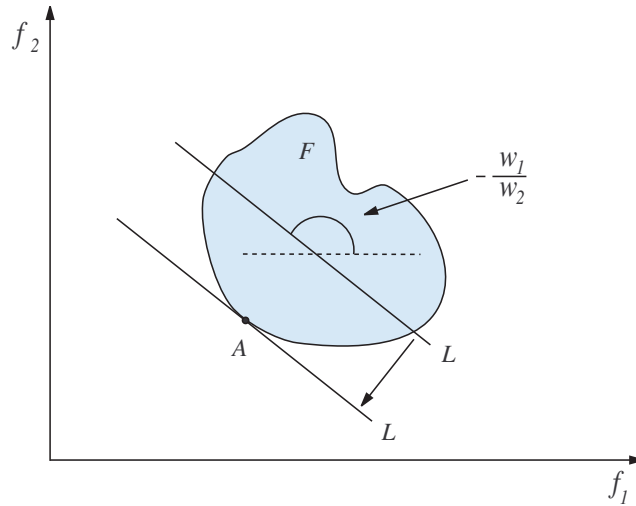


Figura 3.2: Interpretação geométrica do método da soma ponderada: caso convexo (Reproduzido de Osyczka (1978)).

Pode-se traçar uma reta no espaço de objetivos com inclinação $-w_1/w_2$. O conjunto L que representa essa reta é tal que satisfaz a seguinte relação:

$$w_1 f_1(x) + w_2 f_2(x) = c \quad (3.6)$$

onde c é uma constante.

A minimização da Eq. (3.3) pode ser interpretada como o movimento da reta L com os valores de w_1 e w_2 na direção positiva tão longe quanto possível da origem, mas conservando a interseção dos conjuntos L e F . O ponto A para o qual L é tangente a F será o mínimo da Eq. (3.3).

Contudo, para um problema não-convexo, uma grande parte do conjunto de soluções não-inferiores não pode ser avaliada, isto é, não existem valores de w_i que possam localizar

os pontos em uma certa região do conjunto F (OSYCZKA, 1984), como apresentado na Fig. 3.3.

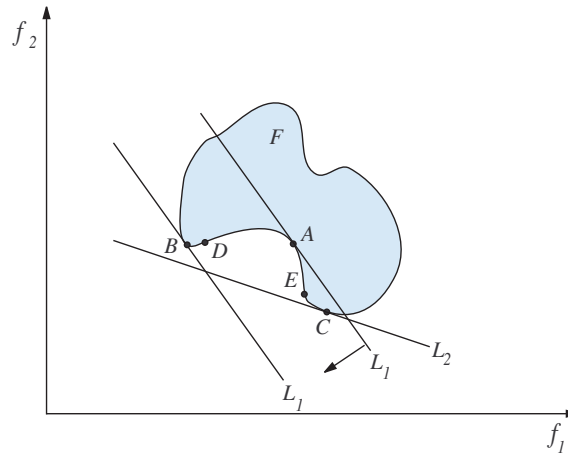


Figura 3.3: Interpretação geométrica do método da soma ponderada: caso não-convexo (Reproduzido de Osyczka (1978)).

Neste caso, o método pode encontrar os pontos B e C , mas não o ponto A . Além disso, é fácil observar que os pontos compreendidos entre D e E não serão avaliados por este método.

Vários autores abordam as dificuldades encontradas quando se faz uso desta metodologia (KOSKI, 1985; STADLER, 1995; MESSAC, 1996; DAS; DENNIS, 1997):

- A escolha dos pesos não necessariamente garante que a solução final será aceita, sendo necessária a resolução do problema com novos pesos. Segundo Messac (1996), os pesos devem ser funções dos objetivos originais;
- É impossível obter pontos em regiões não convexas do conjunto ótimo de Pareto, embora esses problemas sejam incomuns (DAS; DENNIS, 1997);
- A variação dos pesos continuamente não garante resultados distribuídos ao longo da curva de Pareto.

3.2.2 Método de Otimização Hierárquico (*Hierarchical Optimization Method*)

O método de otimização hierárquico foi proposto por Walz (1967) e se baseia no ordenamento dos critérios em termos de importância. Seja a numeração de 1 a k que reflete esta ordem, conforme a importância decrescente dos objetivos.

Conservando esta ordem, minimiza-se cada função objetivo separadamente, adicionando em cada passo uma nova restrição, escrita como sendo a função objetivo limitada por um acréscimo ou decréscimo (ξ_{h_i}), previamente adotado.

A idéia geral da abordagem é:

- Encontra-se o mínimo para o primeiro critério, isto é, $x^1 = [x_1^1, x_2^1, \dots, x_n^1]^T$ tal que:

$$f_1(x^{(1)}) = \min f_1(x), \quad x \in X \quad (3.7)$$

Repetir o passo anterior para $i = 2 \dots k$

- Encontra-se o mínimo da i -ésima função objetivo, isto é, $x^i = [x_1^i, x_2^i, \dots, x_n^i]^T$ tal que:

$$f_i(x^{(i)}) = \min f_i(x), \quad x \in X \quad (3.8)$$

com as restrições adicionais

$$f_{j-1}(x) \leq \left(1 \pm \frac{\xi_{h_{j-1}}}{100}\right) f_{j-1}(x^{j-1}), \quad j = 2, \dots, i \quad (3.9)$$

onde $\xi_{h_{j-1}}$ são os coeficientes percentuais para acréscimos ou decréscimos das funções dadas percentualmente. Os sinais “+” e “-” referem-se às funções que estão sendo minimizadas e maximizadas, respectivamente.

Iterativamente, o método exige que o usuário forneça, no i -ésimo passo, o valor de $\xi_{h_{j-1}}$ com base no mínimo obtido no passo $i-1$. Entretanto, é importante ressaltar que o valor desse parâmetro afeta a qualidade final da solução (WALZ, 1967; OSYCZKA, 1978).

3.2.3 Método da Negociação (*Trade-Off Method*)

Nos métodos de otimização multi-objetivo, os termos de negociação são bastante usados em diferentes contextos, pois existem várias versões do método de negociação com o objetivo de buscar soluções mais eficientes (HAIMES; HALL, 1974; OPPENHEINER, 1978; NULSSEALM; TAVALAGE, 1980).

Em muitos casos o conceito acima é utilizado através da minimização de uma das funções objetivo, considerando outras como restrições flexíveis. Desse modo, o método também é denominado restrito ou ξ -restrito.

A descrição da metodologia é apresentada a seguir:

1) Encontrar o mínimo da r -ésima função objetivo, isto é, encontrar x^* tal que

$$f_r(x^*) = \min f_r(x), \quad x \in X \quad (3.10)$$

sujeita às restrições adicionais da forma:

• para minimização:

$$f_i(x) \leq \xi_{n_i}, \quad i = 1, \dots, k \text{ e } i \neq r \quad (3.11)$$

• para maximização:

$$f_i(x) \geq \xi_{n_i}, \quad i = 1, \dots, k \text{ e } i \neq r \quad (3.12)$$

onde ξ_{n_i} são os valores adotados, os quais não podem ser ultrapassados pelas funções objetivo.

2) Repetir o primeiro passo para diferentes valores de ξ_{n_i} , até que uma solução satisfatória seja obtida.

Segundo Osyczka (1984), para auxiliar na escolha razoável para ξ_{n_i} , é útil minimizar cada função objetivo individualmente, isto é, encontrar a denominada solução ideal f_i° para $i = 1, \dots, k$. De posse desses valores, deve-se escrever a Eq. (3.11) e a Eq. (3.12) como segue:

$$f_i(x) \leq f_i^\circ + \Delta f_i, \quad i = 1, \dots, k \text{ e } i \neq r. \quad (3.13)$$

onde Δf_i são os valores assumidos dessa função incremento.

Pode ser também conveniente montar a denominada “Tabela de Pagamento” (*Pay-Off Table*) conforme mostrado na Tab. 3.1.

Tabela 3.1: Tabela de pagamento para o método da negociação.

	$f_1(x)$	$f_2(x)$	...	$f_i(x)$	...	$f_k(x)$
$x^{o(1)}$	f_1°	f_{21}	...	f_{i1}	...	f_{k1}
$x^{o(2)}$	f_{12}°	f_2	...	f_{i2}	...	f_{k2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x^{o(i)}$	f_{1i}°	f_{2i}	...	f_i°	...	f_{ki}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x^{o(k)}$	f_{1k}°	f_{2k}	...	f_{ik}	...	f_k°

Nesta tabela, a linha i corresponde à solução ótima $x^{o(i)}$, que minimiza a i -ésima

função objetivo, f_{ij} são os valores obtidos da j -ésima função $f_j(x)$ quando a i -ésima função $f_i(x)$ alcança o mínimo f_i° .

A Tabela 3.2 refere-se à função incremento $\Delta f_{ij} = |f_j(x^{o(i)}) - f_j^\circ|$.

Tabela 3.2: Tabela de pagamento usando a função incremento Δf_{ij} .

	$\Delta f_1(x)$	$\Delta f_2(x)$	...	$\Delta f_i(x)$	...	$\Delta f_k(x)$
$x^{o(1)}$	0	Δf_{21}	...	Δf_{i1}	...	Δf_{k1}
$x^{o(2)}$	Δf_{12}°	0	...	Δf_{i2}	...	Δf_{k2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x^{o(i)}$	Δf_{1i}°	Δf_{2i}	...	0	...	Δf_{ki}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x^{o(k)}$	Δf_{1k}°	Δf_{2k}	...	Δf_{ik}	...	0

Similarmente, pode-se construir a tabela de pagamento utilizando incrementos relativos das funções.

$$\Delta f_{ij}^r = \left| \frac{f_j(x^{o(i)}) - f_j^\circ}{f_j(x^{o(i)})} \right| \quad (3.14)$$

3.2.4 Método do Critério Global (*Global Criterion Method*)

Nessa abordagem, a solução ótima é um vetor de variáveis de projeto que minimiza algum critério global. A função que descreve este critério global deve ser definida pelo usuário de forma que se obtenha uma solução mais próxima da solução ideal (f_i°) (OSYCHKA, 1984). Segundo o autor, a forma mais comum desta função é:

$$\min f(x) = \sum_{i=1}^k \left(\frac{f_i^\circ - f_i(x)}{f_i^\circ} \right)^s \quad (3.15)$$

Na literatura, encontram-se alguns valores para s , como por exemplo, $s = 1$ e $s = 2$ (BOYCHUK; OVCHINNIKOV, 1973; SALUKVADZE, 1974). Segundo estes autores, a escolha de tais valores têm grande influência sobre a qualidade da solução, sendo possível a obtenção de resultados inaceitáveis. Uma outra medida de “proximidade para a solução ideal” é uma família de métrica L_p , definida como:

$$L_p(f) = \left[\sum_{i=1}^k |f_i^\circ - f_i(x)|^s \right]^{1/s}, \quad 1 \leq s \leq \infty \quad (3.16)$$

Alternativamente, pode-se trabalhar com distâncias relativas:

$$L_p(f) = \left[\sum_{i=1}^k \frac{|f_i^\circ - f_i(x)|^s}{f_i^\circ} \right]^{1/s}, \quad 1 \leq s \leq \infty \quad (3.17)$$

Em Vanderplaats (1999) e em Edgar et al. (2001), pode-se encontrar uma métrica L_p , que é um vetor normalizado da função objetivo:

$$f_i(x) = \frac{f_i(x) - \min f_i(x)}{\max f_i(x) - \min f_i(x)}, \quad x \in X \quad (3.18)$$

Neste caso, os valores de todas as funções normalizadas são limitadas pelo intervalo de $(0, 1)$.

Uma modificação desse método foi proposta por Wierzbichi (1978). Nesta abordagem, a função global tem uma forma tal que penaliza os desvios em relação a uma função objetivo definida como referência. Seja f^r um ponto interessante do espaço de trabalho, escolhido pelo usuário para o cálculo da função objetivo de referência:

$$\min f(x, f^r) = - \sum_{i=1}^k (f_i(x) - f_i^r)^2 + \varepsilon \sum_{i=1}^k (\max(0, f_i(x) - f_i^r))^2 \quad (3.19)$$

onde $\varepsilon > 0$ é um coeficiente de penalidade, podendo ser constante ou não.

A minimização da Eq. (3.19), para o ponto f^r , permite obter a solução não-inferior que é próxima deste ponto. Se pontos diferentes f^r são escolhidos, o procedimento resulta em soluções diferentes (OSYCHKA, 1984).

3.2.5 Método da Programação de Metas (*Goal Programming Method*)

O Método da Programação de Metas foi proposto por Charnes e Cooper (1961), cuja idéia baseia-se em encontrar uma solução que atinja metas pré-definidas para cada uma das funções objetivo. Contudo, se não existir uma solução que satisfaça esse critério, faz-se necessário, então, encontrar soluções que minimizam os desvios em relação às metas. Por outro lado, se esta solução existir, a tarefa do método é identificar esta solução particular.

A programação de metas ganhou popularidade depois dos trabalhos de Lee (1972) e Ignizio (1976). Romero (1991) fez um levantamento do “estado da arte” desta técnica e listou algumas aplicações em engenharia.

Essa abordagem considera essas metas como sendo restrições adicionais, fazendo assim com que novas variáveis sejam acrescentadas para representar os desvios com relação às metas pré-determinadas. Com isso, a função objetivo especifica os desvios em relação a essas metas e prioriza o sucesso de cada meta, em termos quantitativos.

Seja a minimização de várias funções objetivo $f_i(x)$ com um vetor solução x . No método da Programação de Metas, um valor alvo t_i é escolhido pelo usuário para cada uma das funções objetivo. A tarefa agora é encontrar uma solução que tenha um valor objetivo igual a t_i , sujeito à condição de que a solução encontrada seja viável, isto é, $x \in S$ ($S \rightarrow$ região viável). Desse modo, o problema de otimização pode então ser reformulado como:

$$\text{Meta} \Rightarrow f_i(x) = t_i, \quad i = 1, \dots, M \quad (3.20)$$

Se a meta t_i é menor que o valor ótimo, $f(x^*)$, naturalmente não existe uma solução viável que atingirá exatamente a meta acima. Portanto, o objetivo dessa abordagem é a obtenção da solução que minimiza o desvio Δ entre a meta encontrada e a desejada (t_i). A solução para este problema é x^* , e o desvio é definido como $\Delta = f_i(x^*) - t_i$.

Analogamente, se o valor da função objetivo na meta t_i é maior que o máximo possível $f_{i,max}$, a solução do problema é x , que faz $f_i(x) = f_{i,max}$. Por outro lado, se o valor da função objetivo na meta t_i está dentro da faixa $[f_i(x), f_{i,max}]$, a solução para o problema é o valor de x que faz o valor do objetivo exatamente igual a t_i . Embora esta solução possa não ser a solução ótima da função restrita $f_i(x)$, esta será o resultado ótimo obtido através do método da programação de metas (OSYCHKA, 1984).

De forma a permitir a obtenção das metas apresentadas a seguir, duas variáveis não negativas denominadas desvios (n_i e p_i) são usualmente introduzidas (OSYCHKA, 1978). Segundo o autor, as metas podem ser de quatro tipos diferentes:

- $f_i(x) \leq t_i \Rightarrow$ nessa meta o desvio positivo p_i é subtraído da função objetivo, então, $f_i(x) - p_i \leq t_i$ com $n_i = 0$, isto é, o desvio p_i , representa uma quantidade a ser subtraída do valor do objetivo quando esta ultrapassar a meta t_i . Portanto, o objetivo é minimizar o desvio p_i para encontrar a solução:

$$\begin{cases} f_i(x) > t_i & \text{se } p_i > 0 \\ f_i(x) \leq t_i & \text{se } p_i = 0 \end{cases} \quad (3.21)$$

- $f_i(x) \geq t_i \Rightarrow$ nesse caso, um desvio negativo n_i é adicionado à função objetivo, então, $f_i(x) + n_i \geq t_i$ com $p_i = 0$. O desvio n_i representa uma quantidade a ser

adicionada à função objetivo quando esta não atinge a meta t_i . Aqui, o objetivo é minimizar o desvio n_i :

$$\begin{cases} f_i(x) < t_i & \text{se } n_i > 0 \\ f_i(x) \geq t_i & \text{se } n_i = 0 \end{cases} \quad (3.22)$$

- $f_i(x) = t_i \Rightarrow$ a função objetivo precisa ter o valor t_i , assim são usados ambos os desvios positivos e negativos, então $f_i(x) - p_i + n_i = t_i$. O método minimiza a soma $p_i + n_i$, logo a solução obtida é a distância mínima em relação à meta.

Se $f_i(x) > t_i$, o desvio p_i deve ser um valor positivo diferente de zero, se $f_i(x) < t_i$, o desvio n_i deve ser um valor positivo diferente de zero e para $f_i(x) = t_i$, ambos os desvios p_i e n_i devem ser zeros.

- $f_i(x) \in [t_i^{\min}, t_i^{\max}] \Rightarrow$ esse tipo de meta é tratada usando duas restrições:

$$\begin{cases} f_i(x) - p_i \leq t_i^{\min} \\ f_i(x) + n_i \geq t_i^{\max} \end{cases} \quad (3.23)$$

Todas as restrições acima podem ser substituídas por uma restrição de igualdade:

$$f_i(x) - p_i + n_i = t_i \quad (3.24)$$

Resumidamente, no problema de programação de metas, cada meta é convertida em ao menos uma restrição de igualdade, o que nesse caso resulta em um problema onde se deseja minimizar todos os desvios p_i e n_i . Existem vários tipos de Métodos de Programação de Metas. A seguir são apresentados e discutidos apenas os mais populares.

◦ Programação de Meta Ponderada

Seja o problema de minimização multi-objetivo, onde para cada função objetivo se estabelece uma meta:

$$\min \sum_{j=1}^k (w_j p_j + \beta_j n_j) \quad (3.25)$$

$$\begin{cases} f_j(x) - p_j + n_j = t_j, & j = 1, \dots, k \\ n_j, p_j \geq 0, & j = 1, \dots, k \end{cases} \quad (3.26)$$

Aqui os parâmetros w_j e β_j são os fatores de ponderação para minimizar os desvios do j -ésimo objetivo em relação à j -ésima meta.

Para a meta menor ou igual a t_i ($f_i(x) \leq t_i$), o parâmetro β_j é zero ($n_j = 0$). Analogamente, para a meta maior ou igual a t_i ($f_i(x) \geq t_i$), o parâmetro w_j é zero ($p_j = 0$). Para o intervalo de t_i ($f_i(x) \in [t_i^{min}, t_i^{max}]$), existe um par de restrições para cada função objetivo. Usualmente, os fatores de ponderação w_j e β_j são fixados pelo usuário.

◦ *Programação Lexocográfica*

Nessa abordagem, o usuário tem que definir uma ordem relativa ou níveis de prioridades, sem o uso de pesos para ponderar os objetivos. Assim, a meta definida como primeira prioridade é muito mais importante que a definida como segunda.

De forma geral, seja o problema com dois objetivos f_1 e f_2 representados na Fig. 3.4.

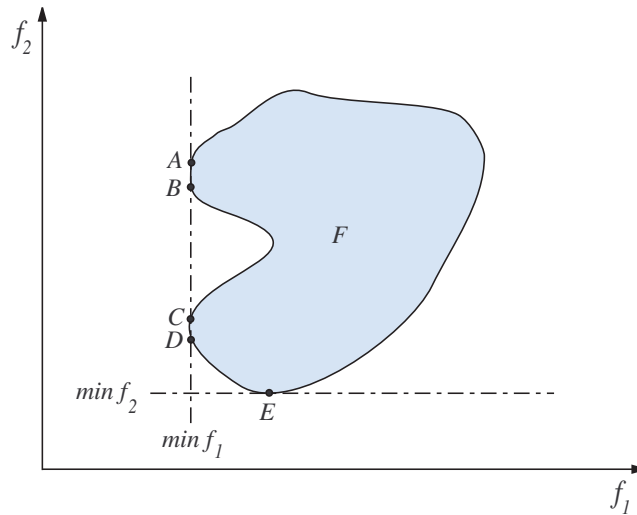


Figura 3.4: Programação lexocográfica (Reproduzido de Osyczka (1978)).

Se o objetivo f_1 é mais importante que f_2 , minimiza-se o problema com f_1 primeiro, ignorando f_2 . Neste caso, encontram-se soluções múltiplas em AB e CD no primeiro nível da programação de meta. Daí, como existe mais de uma solução para este problema, procede-se para o segundo nível de otimização, onde f_2 é minimizada. A procura é limitada entre soluções encontradas no primeiro nível de programação. A solução do segundo nível da programação é D , que é solução mínima de f_2 entre todas as soluções em AB e CD . Portanto, a solução D é a solução principal do problema. É interessante notar que se f_2 é considerada mais importante que f_1 , então a solução E poderá ser a única do primeiro nível de otimização e o processo poderá parar aqui. Nesse caso, a solução E será declarada como solução do problema.

Segundo Osyczka (1978), a principal limitação dessa abordagem é que, quando o número de objetivos é grande, ele tende a otimizar prioritariamente os mais importantes.

o *Programação de Meta Min-Max*

Esta metodologia é análoga à programação de meta ponderada, mas em vez de minimizar a soma ponderada dos desvios em relação às suas metas, esta deve obedecer a um valor máximo de desvio Δ , escrito na forma de restrição, e este desvio máximo é minimizado. Resulta um problema de programação não-linear dado por:

$$\min \Delta \quad (3.27)$$

$$\begin{cases} w_j p_j + \beta_j n_j \leq \Delta \\ f_j(x) - p_j + n_j = t_j \\ p_j, n_j \geq 0 \end{cases} \quad (3.28)$$

para $j = 1, \dots, k$.

3.2.6 Método do Vetor Objetivo (*Target Vector Technique*)

Nessa abordagem, o POMO é substituído por um problema com um único objetivo, cuja meta é minimizar:

$$\min f^*(x) = \|[f(x) - T]W^{-1}\|_\alpha \quad (3.29)$$

onde T é o vetor objetivo definido pelo responsável pela tomada de decisões, W é a matriz de pesos que considera diferentes escalas para os tamanhos entre as m metas, e α é geralmente tomado como sendo igual a 2 (distância Euclidiana).

3.2.7 Otimização de Compromisso (*Compromise Optimization*)

A Otimização de Compromisso se fundamenta em uma combinação dos objetivos a serem alcançados, realizada da seguinte forma (VANDERPLAATS, 1999):

$$\min f(x) = \left(\sum_{k=1}^m \left(\frac{w_k(f_k(x) - f_k^*(x))}{f_k^{worst}(x) - f_k^*(x)} \right)^2 \right)^{0,5} \quad (3.30)$$

onde w_k são os fatores de ponderação das k -ésimas funções objetivo, $f_k(x)$ é a k -ésima função objetivo, $f_k^*(x)$ é o melhor valor da k -ésima função objetivo e $f_k^{worst}(x)$ é o pior valor da k -ésima função objetivo.

Vanderplaats (1999) ressalta que a dificuldade associada com essa metodologia é o fato de se ter que arbitrar os valores dos pesos, assim como dos melhores e piores valores para cada função objetivo, individualmente. Ainda segundo o autor, os fatores de ponderação devem ser escolhidos como unitários, a menos que alguns objetivos sejam mais importantes que outros, o que significa que, para esses casos, se utiliza um fator de ponderação maior. $f_k^{worst}(x)$ é normalmente a função objetivo associada com a estimativa inicial de projeto x^o , $f_k^*(x)$ é o valor desejado para esta função objetivo, sendo este o valor mais difícil de ser atribuído. Alternativamente, pode-se considerar esses valores de $f_k^*(x)$ como sendo o valor ótimo de cada objetivo individualmente, levando em consideração todas as restrições do problema original. Apesar dessa metodologia ser uma ferramenta poderosa na otimização multi-objetivo, deve ser enfatizado que a solução não é única e que uma alteração em qualquer um dos parâmetros w_k , $f_k^{worst}(x)$ ou $f_k^*(x)$ pode levar a valores ótimos diferentes (VANDERPLAATS, 1999).

3.2.8 Método do Critério Ponderado Exponencial (*Exponential Weighted Criterion*)

Em resposta à inabilidade do Método da Soma Ponderada em capturar pontos em regiões não convexas da Curva de Pareto, Athan e Papalambros (1996) propuseram o Método do Critério Ponderado Exponencial:

$$\min f^*(x) = \sum_{i=1}^k (e^{pw_i} - 1) e^{pf_i(x)} \quad (3.31)$$

onde o argumento do somatório representa uma função utilidade individual para a i -ésima função objetivo. Segundo os autores, grandes valores atribuídos a p podem conduzir a uma saturação do processo.

3.2.9 Método do Produto Ponderado (*Weighted Product Method*)

Bridgman (1922) apresentou o Método do Produto Ponderado. Sua metodologia baseia-se em fazer com que funções de diferentes ordens tenham importâncias similares

através do seguinte critério:

$$\min f^*(x) = \prod_{i=1}^k (f_i(x))^{w_i} \quad (3.32)$$

onde w_i são pesos que indicam a importância relativa das funções objetivo.

3.2.10 Métodos Min-Max

Essa abordagem consiste na comparação dos desvios relativos obtidos de cada função objetivo $f_i(x)$ com o vetor dado pela solução ideal (f_i°). O desvio pode ser calculado pelas seguintes relações:

$$z'_i(x) = \frac{|f_i(x) - f_i^\circ|}{|f_i^\circ|}, \quad i \in \{1, \dots, m\} \quad (3.33)$$

ou

$$z''_i(x) = \frac{|f_i(x) - f_i^\circ|}{|f_i(x)|}, \quad i \in \{1, \dots, m\} \quad (3.34)$$

Os primeiros trabalhos nessa área foram desenvolvidos por Jutler (1967) e Solich (1969) com a aplicação em um modelo linear, com posterior aprimoramento por Osyczka (1978).

Um ponto $x^* \in X$ é o ótimo min-max se, para todo $x \in X$, o seguinte procedimento é satisfeito:

- Passo 1): $v_1(x^*) = \min \max\{z_i(x)\}$, $i \in I$ e $x \in X \Rightarrow$ Existe $I_1 = \{i_1\}$, onde i_1 é o índice para o qual o valor de $z_i(x)$ é máximo. Se existir um conjunto de soluções $x_1 \subset X$ que satisfaz o passo 1, então siga para o próximo passo;
- Passo 2): $v_2(x^*) = \min \max\{z_i(x)\}$, $i \in I$, $x \in X_1$ e $i \not\subset I_1 \Rightarrow I_1 = \{i_1, i_2\}$, onde i_2 é o índice para o qual o valor de $z_i(x)$ nesse passo é máximo. Se existir um conjunto de soluções $x_{r-1} \subset X$ que satisfazem o passo $r - 1$, então siga para o r -ésimo passo;
- Passo r): $v_r(x^*) = \min \max\{z_i(x)\}$, $i \in I$, $x \in X_{r-1}$ e $i \not\subset I_{r-1} \Rightarrow I_r = \{i_{r-1}, i_r\}$, onde i_r é o índice para o qual o valor de $z_i(x)$ no r -ésimo passo é máximo. Se existir um conjunto de soluções $x_{k-1} \subset X$ que satisfazem o passo $k - 1$, então vá para o r -ésimo passo;

- Passo k): $v_k(x^*) = \min\{z_i(x)\}$, $i \in I$, $x \in X_{k-1}$ e $i \not\subseteq I_{k-1}$, onde $v_i(x^*), \dots, v_k(x^*)$ é o conjunto dos valores ótimos dos desvios fracionários e ordenados não-decrescentes.

Esse procedimento pode ser resumido como: conhecendo-se a solução ideal de cada função objetivo, as quais podem ser obtidas considerando cada critério individualmente, a solução desejada é aquela que fornece o menor valor dos incrementos relativos de todas as funções objetivo.

Nesse capítulo foram mostrados os principais métodos utilizados para a transformação do problema vetorial em um problema escalar. No próximo capítulo será apresentado um estudo sobre a resolução de POMO através de métodos clássicos de otimização, utilizando as técnicas citadas.

ESTUDO DE POMO USANDO MÉTODOS CLÁSSICOS

 notório que com o aprimoramento dos métodos baseados no Cálculo Variacional e com desenvolvimento dos computadores, tornou-se viável o tratamento de problemas de engenharia numericamente, já que, em sua grande maioria estes não possuem solução analítica. Os métodos propostos na literatura para a solução dos Problemas de Otimização via Métodos Clássicos (MC) se caracterizam pelo fato de resolverem o problema em sua forma original, isto é, com a presença de todas suas restrições de igualdade e desigualdade (Métodos Diretos) ou fazendo uso de abordagens que transformam o problema com restrições em um problema sem restrições (Métodos Sequenciais).

De uma forma geral, os MC se baseiam na procura iterativa caracterizada pela determinação da direção de busca, através da execução de uma busca unidimensional, e da obtenção do tamanho do passo nessa direção. Este processo iterativo é finalizado através da definição do número máximo de iterações ou se nenhuma alteração absoluta ou relativa no valor da função objetivo é obtida ou se existe obediência no atendimento às condições de Kuhn-Tucker (VANDERPLAATS, 1999).

Nas próximas seções serão apresentadas, de forma detalhada, as principais metodologias e as suas principais características.

4.1 Método Randômico Clássico

O Método Randômico ou Método de Ordem Zero é a abordagem mais simples já concebida, sendo de fácil implementação, de razoável confiabilidade e capaz de trabalhar com

valores discretos (VANDERPLAATS, 1999). Requer apenas o cálculo de $f(x)$, não dependendo de informações e de avaliações de derivadas de ordem superior. Além disso, não necessita de garantia de continuidade da função objetivo e de suas restrições. Entretanto, essa abordagem necessita de um grande número de avaliações da função objetivo, o que aumenta, em comparação com outros métodos, o seu custo computacional, tornando-o inadequado para a maioria das aplicações (SARAMAGO, 1999).

O Método Randômico Clássico fundamenta-se na geração de um vetor de variáveis de projeto, dentro de um domínio especificado, sendo que a configuração que atribuir à função objetivo o seu melhor resultado é considerada como sendo o valor ótimo x^* . Conseqüentemente, nas várias iterações, são gerados candidatos a pontos ótimos e esses valores são confrontados com o valor ótimo obtido na iteração anterior. Se algum destes candidatos for melhor, em comparação com o valor ótimo obtido anteriormente, esta configuração de projeto é considerada como novo ponto ótimo (x^*). Esse processo se repete até que um determinado critério de convergência seja atendido. No caso do Método Randômico Clássico, utiliza-se como critério de parada o número máximo de iterações. O vetor x é obtido a partir da geração do número randômico r em cada iteração, levando em consideração os limites inferior (L) e superior (U) das variáveis projeto, da seguinte maneira:

$$x^q = x^L + r(x^U - x^L) \quad (4.1)$$

Cabe aqui uma ressalva, este método apesar de ser uma técnica puramente randômica, se diferencia dos Métodos Não Determinísticos pelo simples fato de não ser fundamentado por nenhuma “estratégia evolutiva” para a atualização das variáveis de projeto, assim como ocorre no algoritmo genético, por exemplo.

4.2 Métodos de Ordem Superior

Como descrito anteriormente, a solução iterativa do problema de otimização consiste em obter a direção de busca S e determinar o tamanho do passo α . A utilização do Método de Ordem Zero implica em um número por demais grande de avaliações da função objetivo, sendo que não existe nenhuma garantia da obtenção do ótimo (VANDERPLAATS, 1999). Por esse motivo, foram desenvolvidos métodos que fazem uso de informações sobre derivadas de primeira e de segunda ordem, para a determinação da direção de busca ao longo do espaço de projeto. Estes são caracterizados por uma busca do ponto ótimo da

função objetivo através de procedimentos iterativos, conforme mostrado na Eq. (4.2):

$$x^q = x^{q-1} + \alpha^* S^{q-1} \quad (4.2)$$

onde x^q é o vetor de variáveis de projeto na iteração atual, x^{q-1} é o vetor de variáveis de projeto na iteração anterior, α^* é um escalar que fornece o tamanho do passo ao longo da direção de busca e S^{q-1} é o vetor que define a direção de busca.

Portanto, o problema da busca do valor ótimo é composto por duas etapas: a primeira, realizada a cada iteração, identificando uma direção de busca S^{q-1} e, a partir daí, a otimização se torna um problema de busca uni-dimensional, onde a cada iteração um escalar α^* estabelece o passo a ser dado sobre a direção obtida. A seguir, são apresentadas algumas abordagens para a determinação da direção de busca S .

4.2.1 Métodos para a Determinação da Direção de Busca

Os métodos para determinação da Direção de Busca podem ser classificados de acordo com o grau da derivada da função objetivo considerada no cálculo. Os mais importantes são descritos a seguir.

a) Métodos de Ordem Zero: são caracterizados por sua simplicidade conceitual e por sua facilidade de implementação, não envolvendo o cálculo de derivadas durante todo o procedimento de otimização. Porém, devido ao fato de requerer um grande número de avaliações da função objetivo, estes tornam-se inadequados para utilização em casos onde o modelo numérico exija elevado esforço computacional. Dentre os métodos de ordem zero existentes, o Método da Busca Aleatória é o mais fácil de ser implementado, sendo, porém, considerado como um dos menos eficientes. Este consiste basicamente na avaliação da função objetivo em pontos selecionados aleatoriamente dentro do espaço de projeto definido pelas restrições (SARAMAGO, 1999). Já o Método de Powell, é um dos mais eficientes métodos de ordem zero. Caracteriza-se por realizar inicialmente buscas uni-direcionais na direção de cada variável de projeto de forma iterativa, definindo a partir daí a próxima direção de busca como uma combinação linear das direções anteriores (VANDERPLAATS, 1999).

b) Métodos de Primeira Ordem: são baseados na utilização de informações do gradiente da função objetivo com relação às variáveis de projeto. Por isso, são mais eficientes que os métodos de ordem zero, mas em contrapartida, adicionam um certo nível de dificuldade à sua implementação quando necessitam lidar com problemas de alta dimensão. Um método bastante conhecido é o método da Máxima Descida (VANDERPLAATS, 1999), que estabelece como direção de busca aquela contrária à do gradiente da função objetivo

no ponto avaliado, conforme é mostrado na Eq. (4.3):

$$S = -\nabla f(x) \quad (4.3)$$

Segundo Vanderplaats (1999) e Edgar et al. (2001), a desvantagem deste método é a baixa taxa de convergência, devido à redução das distâncias percorridas em cada iteração, nas proximidades do ponto ótimo, exigindo assim a realização de várias iterações sucessivas, até que o ponto ótimo seja alcançado.

No método das direções conjugadas (Método de Fletcher e Reeves), ao contrário do método da máxima descida, a direção de busca atual é calculada levando em consideração a direção de busca anterior, proporcionando maior eficiência do ponto de vista da convergência. Inicialmente a direção de busca é dada pelo vetor oposto ao gradiente da função objetivo no ponto de partida e, nas demais iterações, as direções de busca são as direções conjugadas, conforme definido na Eq. (4.4):

$$S = -\nabla f(x^q) + \beta^q S^{q-1} \quad (4.4)$$

onde β é dado pela Eq. (4.5):

$$\beta^q = \frac{|\nabla f(x^q)|^2}{|\nabla f(x^{q-1})|^2} \quad (4.5)$$

A eficiência desse algoritmo, comparado ao da Máxima Descida, está no fato de que ele utiliza informações das iterações anteriores para determinar a iteração atual, fazendo com que alcance o ótimo em menos iterações. Além disso, este método não apresenta a perda de eficiência de busca nas proximidades do ponto ótimo (VANDERPLAATS, 1999; EDGAR et al., 2001).

De maneira semelhante ao Método das Direções Conjugadas, o Método da Variável Métrica incorpora ao cálculo de uma direção de busca informações das direções precedentes. Contudo, isto é feito através de uma aproximação da matriz Hessiana H , e não de um escalar β^q , conforme descrito a seguir:

$$S^q = -H^q \nabla f(x^q) \quad (4.6)$$

$$H^q = H^{q-1} + D^q \quad (4.7)$$

$$D^q = \frac{\sigma + \theta\tau}{\sigma^2} pp^T + \frac{\theta - 1}{\tau} (H^q y)(H^q y)^T - \frac{\theta}{\sigma} (H^q y p^T + p(H^q y)^T) \quad (4.8)$$

$$p = x^q - x^{q-1} \quad (4.9)$$

$$y = \nabla f(x^q) - \nabla f(x^{q-1}) \quad (4.10)$$

$$\sigma = py \quad (4.11)$$

$$\tau = y^T H^q y \quad (4.12)$$

O Método da Variável Métrica apresenta duas variantes, conforme o valor dado para o coeficiente θ na Eq. (4.8), de tal forma que:

- Para $\theta = 1 \Rightarrow$ Método de Davidon-Fletcher-Powell (DFP)
- Para $\theta = 0 \Rightarrow$ Método de Broyden-Fletcher-Goldfarb-Shanno (BFGS)

c) Métodos de Segunda Ordem: são métodos que utilizam informações da matriz Hessiana em sua formulação. O método clássico de segunda ordem é o Método de Newton, sendo caracterizado pela obtenção da matriz Hessiana através da Série de *Taylor*, conforme a Eq. (4.13):

$$f(x^q) \approx f(x^{q-1}) + \nabla f(x^{q-1})^T \delta x + \frac{1}{2} \delta x^T H^{q-1} \delta x \quad (4.13)$$

onde

$$\delta x = x^{q-1} - x^q \quad (4.14)$$

Resolvendo a Eq. (4.13) de acordo com as condições de estacionariedade, tem-se:

$$\delta x = -(H^{q-1})^{-1} \nabla(f(x^{q-1})) \quad (4.15)$$

Substituindo Eq. (4.14) na Eq. (4.15), tem-se

$$x^q = x^{q-1} - (H^{q-1})^{-1} \nabla(f(x^{q-1})) \quad (4.16)$$

Finalmente, substituindo a Eq. (4.16) na Eq. (4.2), resulta:

$$S^{q-1} = -(H^{q-1})^{-1} \nabla(f(x^{q-1})) \quad (4.17)$$

Segundo Vanderplaats (1999), a principal dificuldade associada com a utilização do Método de Newton, é o fato de que a matriz Hessiana pode ser singular ou não ser positiva definida. Além disso, se as variáveis da função aparecem na forma linear, o vetor de busca S pode se tornar mal condicionado. Se a matriz Hessiana tem autovalores negativos, estamos lidando com um problema não convexo, o que pode causar oscilações na solução. Outro ponto interessante é a dificuldade da obtenção analítica da matriz Hessiana para problemas mais complexos ou de grande dimensão, tornando necessário o

uso de aproximações baseadas nas fórmulas de diferenças finitas para a obtenção dessa matriz (VANDERPLAATS, 1999; EDGAR et al., 2001).

4.2.2 Determinação do Passo na Direção de Busca

Uma vez definida uma direção de busca, o próximo passo consiste na determinação do tamanho do passo de otimização α . Para esta finalidade, dois métodos são muito utilizados:

a) O Método da Seção Áurea: consiste na redução do intervalo inicial $[x^l, x^s]$, através da avaliação da função uni-direcional, formada pela substituição da metodologia utilizada para a determinação de S , em dois pontos simétricos (x_1 e x_2), contidos neste intervalo, de tal forma que satisfaçam a Eq. (4.18):

$$\frac{x_1 - x^l}{x^s - x^l} = \frac{x_2 - x_1}{x^s - x_1} \quad (4.18)$$

Considerando a função uni-modal na direção de busca, se $f(x_1) > f(x_2)$, então o novo intervalo que engloba o mínimo será $[x_1, x^s]$, caso contrário o intervalo passa a ser definido entre $[x^l, x_2]$. A execução sucessiva deste procedimento resulta na convergência do intervalo à vizinhança do ponto de mínimo.

b) O Método da Interpolação Polinomial Cúbica: este método aplica-se após a redução do intervalo de busca, consistindo na obtenção da raiz real positiva da função, gerada a partir de uma função polinomial de terceiro grau que aproxima a função unidimensional.

4.2.3 Critérios de Convergência

Os métodos baseados no cálculo do gradiente apresentam característica iterativa, isto é, a busca do ponto ótimo é feita através de várias iterações, onde direções de busca são calculadas e sobre as quais passos maiores ou menores são dados. Esta característica iterativa representa uma vantagem na implementação computacional, pois passam a ser executadas de forma automática, até que um critério de parada ou convergência seja satisfeito (VANDERPLAATS, 1999; EDGAR et al., 2001). As regiões definidas dentro do espaço de projeto pela função objetivo e pelas restrições, são classificadas, respectivamente, como região usável (que proporciona a redução da função objetivo) e viável (onde não há violação das restrições), como mostrado na Fig. 4.1.

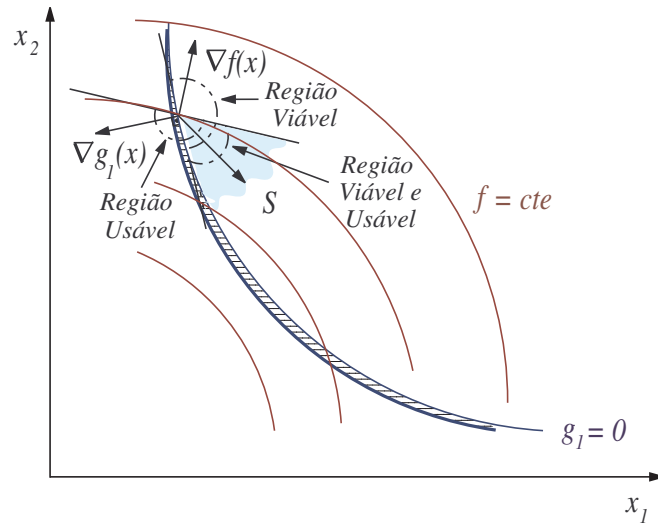


Figura 4.1: Definição de regiões dentro do espaço de projeto (Reproduzido de Vanderplaats (1999)).

As condições de usabilidade e viabilidade podem ser expressas matematicamente como:

$$\nabla(f(x)) \cdot S \leq 0 \quad (4.19)$$

$$\nabla(g_j(x)) \cdot S \leq 0 \quad (4.20)$$

Na Figura 4.1 são mostradas as regiões usável e viável, considerando as condições expressas nas Eq. (4.19) e Eq. (4.20), sendo que $f(x)$ é uma função objetivo genérica de duas variáveis (x_1 e x_2), restrita pela função $g_1(x)$. Um conjunto de equações conhecido como “Condições de Kuhn-Tucker” é utilizado na verificação periódica da convergência, sendo responsável por examinar os valores das restrições e da função objetivo, decidindo sobre a continuidade do procedimento de otimização. A condição da Eq. (4.20) estabelece que a configuração ótima deve estar inserida na região viável do espaço de projeto, isto é, todas as restrições impostas são satisfeitas. Se algumas das restrições de desigualdade não puderem ser estritamente satisfeitas, então os multiplicadores de Lagrange λ_j correspondentes devem ser nulos. Os multiplicadores de Lagrange são escalares que representam um fator de escala para os vetores gradiente das funções de restrição. Assim sendo, a Eq. (4.22) estabelece que os vetores gradiente associados às restrições de desigualdade que não podem ser satisfeitas durante o processo de otimização devem ser reduzidos a zero. A Equação (4.23) expressa o equilíbrio vetorial entre os vetores gradiente das funções envolvidas na formulação do problema de otimização (função objetivo, restrições de igualdade e desigualdade). Uma vez satisfeita essa equação, não existirá uma direção de busca preponderante pela qual o processo de otimização deva prosseguir, uma vez que

todas as parcelas da equação se anulam, conforme se pode observar na Fig. 4.2.

$$x \text{ é viável} \quad (4.21)$$

$$\lambda_j g_j(x^*) = 0, \lambda_j \geq 0, \quad j = 1 \dots J \quad (4.22)$$

$$\nabla f(x^*) + \sum_{j=1}^J \lambda_j \nabla g_j(x^*) + \sum_{k=1}^L \lambda_{k+m} \nabla h_k(x^*) = 0, \lambda_j \geq 0 \quad (4.23)$$

As condições de Kuhn-Tucker demonstram que a convergência do processo de otimização é fortemente influenciada pelas restrições de igualdade e desigualdade. Assim, a formulação do problema de otimização deve ser feita com uma atenção especial à forma pela qual as restrições são impostas dentro do espaço de projeto.

Na Figura 4.2 é mostrado como a soma das componentes $\nabla f(x)$, $\lambda_1 \nabla g_1(x)$ e $\lambda_2 \nabla g_2(x)$ se anulam no ponto x^* , para a função $f(x_1, x_2)$ restrita pelas funções $g_1(x)$ e $g_2(x)$.

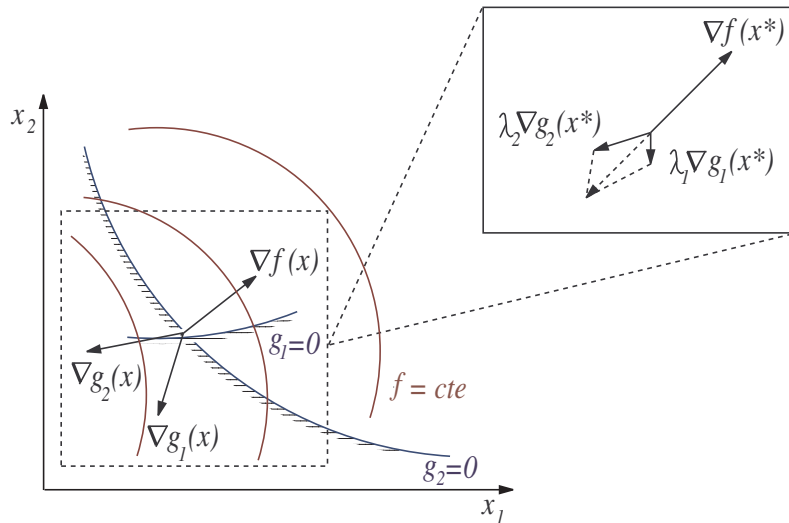


Figura 4.2: Ilustração esquemática das condições de Kuhn-Tucker (Reproduzido de Vanderplaats (1999)).

4.3 Métodos Seqüenciais

Os métodos seqüenciais não são capazes de lidar diretamente com restrições. A sua utilização se dá pela transformação do problema original com restrições em um problema sem restrições, através da adição de uma função de penalidade ao objetivo original, ponderada por um escalar. A forma como é definida a função de penalidade e os principais métodos serão abordados posteriormente no Capítulo 6.

4.4 Métodos Diretos

Estes métodos, de forma análoga aos métodos sequenciais, buscam o ponto ótimo através de procedimentos iterativos com base no cálculo de gradientes. Contudo, a formulação destes métodos é caracterizada pela determinação de direções de busca ao longo das restrições, através do manuseio direto das restrições de caminho sem que nenhuma modificação com relação à eliminação dessas restrições seja feita (VANDERPLAATS, 1999; EDGAR et al., 2001). São descritos a seguir as três principais metodologias existentes para esta classe de métodos.

4.4.1 Programação Seqüencial Linear - SLP (*Sequential Linear Programming*)

Apesar da maioria dos projetos serem inerentemente não lineares, muitas vezes é possível linearizar o problema para a obtenção de uma solução aproximada através do uso de Métodos de Programação Linear. De posse desta solução aproximada, utilizando a linearização em torno dessa solução, obtida anteriormente, resolve-se o novo problema de Programação Linear, até que um determinado critério de parada seja alcançado. Esse procedimento iterativo é denominado Programação Seqüencial Linear (SLP).

O problema geral de programação não linear é dado por Eq. (3.1) e Eq. (3.2). A partir da linearização, através de aproximações dadas por Séries de Taylor, chega-se a:

$$f(x) \approx f(x^\circ) + \nabla f(x^\circ)^T \delta x \quad (4.24)$$

$$\begin{cases} g_j(x^\circ) \approx g_j(x^\circ) + \nabla g_j(x^\circ)^T \delta x \leq 0 & j = 1, \dots, J \\ h_k(x^\circ) \approx h_k(x^\circ) + \nabla h_k(x^\circ)^T \delta x = 0 & k = 1, \dots, K \\ x_i^L \leq x_i + \delta x_i \leq x_i^U & i = 1, \dots, n \end{cases} \quad (4.25)$$

O sobrescrito *zero* identifica o ponto sobre o qual a expansão em série de Taylor é realizada. A Figura 4.3 apresenta a interpretação geométrica do Método SLP, onde se vê o projeto inicial x° e as aproximações lineares da função objetivo e das restrições são dadas pelas linhas pontilhadas. A solução do problema aproximado, após certo número de iterações, se aproxima da solução do problema não linear, mas é inviável por violar as restrições. Contudo, ao se continuar aplicando esse procedimento, pode-se aproximar satisfatoriamente da solução do problema (EDGAR et al., 2001).

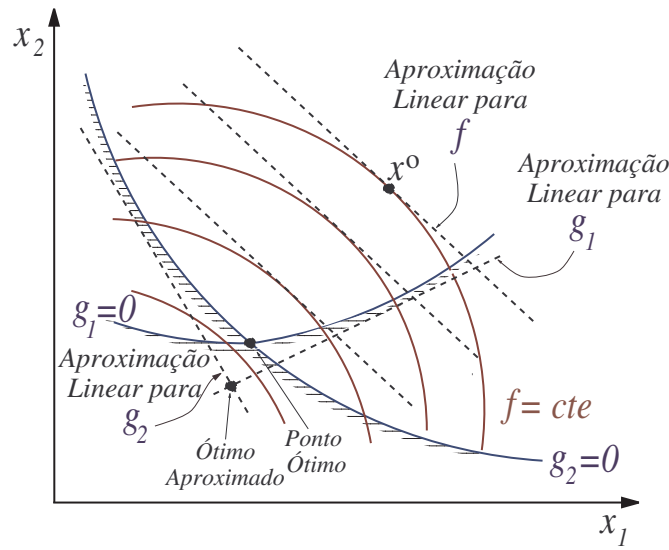


Figura 4.3: Interpretação geométrica do método SLP (Reproduzido de Vanderplaats (1999)).

4.4.2 Gradiente Reduzido Generalizado - GRG (*Generalized Reduced Gradient*)

No Método do Gradiente Reduzido Generalizado (GRG), a direção de busca é obtida de forma que qualquer ativação de restrição permanece sempre ativa para um pequeno movimento nesta direção. Isso é feito através da adição de uma variável de folga (*slack variable*) a cada restrição de desigualdade, permitindo obter o seguinte problema:

$$\begin{cases} g_j(x) + x_{j+n} = 0 & j = 1, \dots, J \\ x_{j+n} \geq 0 & j = 1, \dots, J \end{cases} \quad (4.26)$$

A Equação (4.26) requer a adição de J variáveis de folga não negativas, num total de $n+J$ variáveis de projeto. Se houver várias restrições de desigualdade, essa metodologia se torna ineficaz, na medida que fica muito complicado o controle dessas novas variáveis. Entretanto, segundo Vanderplaats (1999), para um número pequeno de restrições, essa é uma ferramenta poderosa.

De maneira geral, o problema GRG apresenta a seguinte forma:

$$\min f(x) \quad (4.27)$$

$$\begin{cases} g_j(x) + x_{j+n} = 0 & j = 1, 2, \dots, J \\ h_k(x) = 0 & k = 1, 2, \dots, K \\ x_i^L \leq x_i \leq x_i^U & i = 1, 2, \dots, n \\ x_{j+n} \geq 0 & j = 1, 2, \dots, J \end{cases} \quad (4.28)$$

O vetor x contém as n variáveis iniciais do problema e as J variáveis de folga. Por conveniência, o sistema é particionado como segue:

$$x = [Z \ Y]^T \quad (4.29)$$

com $n-l$ variáveis independentes e $J+l$ variáveis dependentes, respectivamente. Dessa forma, obtêm-se o problema de otimização modificado:

$$\min f(x, Z) \quad (4.30)$$

$$\begin{cases} h_k(x) = 0 & j = 1, 2, \dots, J + l \\ x_i^L \leq x_i \leq x_i^U & i = 1, 2, \dots, n + J \end{cases} \quad (4.31)$$

Diferenciando a função objetivo e as restrições chega-se a:

$$\begin{cases} df(x) = \nabla_Z f(x)^T dZ + \nabla_Y f(x)^T dY \\ dh_j(x) = \nabla_Z h_j(x)^T dZ + \nabla_Y h_j(x)^T dY \end{cases} \quad (4.32)$$

onde os subscritos Z e Y indicam em relação a qual vetor de variáveis os gradientes estão sendo tomados.

Adotando-se o fato de que as restrições de igualdade são inicialmente satisfeitas, para qualquer mudança nas variáveis de decisão as restrições de igualdade devem permanecer satisfeitas para garantir viabilidade. Se for considerado que $dh_j(x) = 0$, para $j=1, \dots, J+l$ e que Y contém $J+l$ variáveis dependentes, pode-se escrever:

$$dh(x) = \begin{bmatrix} \nabla_Z h_1(x) \\ \nabla_Z h_2(x) \\ \vdots \\ \nabla_Z h_{J+l}(x) \end{bmatrix} dZ + \begin{bmatrix} \nabla_Y h_1(x) \\ \nabla_Y h_2(x) \\ \vdots \\ \nabla_Y h_{J+l}(x) \end{bmatrix} dY \quad (4.33)$$

$$dh(x) = AdZ + BdY \quad (4.34)$$

onde A é uma matriz $(J+l) \times (n-l)$ e B é uma matriz $(J+l) \times (J+l)$. Para qualquer mudança das variáveis independentes dZ , pode-se resolver a Eq. (4.34) para manter a viabilidade:

$$dY = -B^{-1}AdZ \quad (4.35)$$

Substituindo a Eq. (4.35) na Eq. (4.32) chega-se a:

$$\begin{cases} df = \nabla_Z f(x)^T dZ - \nabla_Y f(x)^T B^{-1} AdZ \\ df = (\nabla_Z f(x)^T - \nabla_Y f(x)^T B^{-1} A) dZ \end{cases} \quad (4.36)$$

que fornece

$$G_R = \frac{df}{dZ} = (\nabla_Z f(x)^T - \nabla_Y f(x)^T B^{-1} A) \quad (4.37)$$

definido como *Gradiente Reduzido Generalizado*, G_R , que pode ser entendido como um problema sem restrição. O G_R pode agora ser usado para determinar a direção de busca S . Para cada valor de α proposto, as variáveis Y são atualizadas segundo a Eq. (4.35). Entretanto, deve-se ressaltar que, como a Eq. (4.35) é uma aproximação linear do problema original, quando as restrições forem avaliadas para esse valor de α , elas podem não ser anuladas exatamente.

4.4.3 Programação Quadrática Sequencial - SQP (*Sequential Quadratic Programming*)

O Método da Programação Quadrática Sequencial (SQP) resolve uma seqüência de aproximações quadráticas do problema não linear original. Os Problemas Quadráticos (QP) têm a função objetivo quadrática e as restrições lineares. Assim como no SLP, as restrições não lineares são linearizadas em torno de um ponto selecionado. Sua metodologia tem por objetivo obter uma aproximação quadrática da função Lagrangeano e utilizar um dos métodos de penalidade para transformá-lo em um problema sem restrições.

A direção de busca é obtida através da criação da seguinte aproximação quadrática:

$$\min Q(S) = f(x) + \nabla f(x)^T S + \frac{1}{2} S^T B S \quad (4.38)$$

$$\begin{cases} \nabla g_j(x)^T S + \delta_j g_j(x) \leq 0 & j = 1, \dots, J \\ \nabla h_k(x)^T S + \bar{\delta} h_k(x) = 0 & k = 1, \dots, l \end{cases} \quad (4.39)$$

Aqui as variáveis de projeto são os componentes de S . A matriz B é positiva definida, a qual assume, no início do processo iterativo, o valor da matriz identidade, sendo atualizada em iterações subseqüentes através de aproximações da matriz Hessiana da função Lagrangeano. Os parâmetros escalares δ_j e $\bar{\delta}$ são dependentes do problema e são usados para prevenir inconsistências entre as restrições linearizadas, sendo definidos como (VANDERPLAATS, 1999):

$$\delta_j = \begin{cases} 1 & \text{se } g_j(x) < 0 \\ \bar{\delta} & \text{se } g_j(x) \geq 0 \end{cases} \quad (4.40)$$

Segundo Vanderplaats (1999) e Edgar et al. (2001), o parâmetro $\bar{\delta}$ é escolhido como sendo 0,9 ou 0,95. Utilizando, por exemplo, o Método da Função de Penalidade Exterior (Seção 4.3), e de posse da direção de busca S do problema redefinido (Eq. (4.38) - Eq. (4.39)), obtém-se o tamanho do passo de otimização α . O próximo passo consiste em atualizar a aproximação da matriz Hessiana B , o que pode ser feito utilizando o Método BFGS.

Por causa da utilização do Método BFGS, também conhecido como Método Quasi-Newton, para a atualização de B , o algoritmo SQP obtém informações de segunda ordem, o que implica taxas de convergência muito próximas dos métodos clássicos de segunda ordem (VANDERPLAATS, 1999). Segundo Edgar et al. (2001) podem surgir problemas ligados ao armazenamento da matriz Hessiana para problemas com mais de 1000 variáveis.

4.4.4 Vantagens e Desvantagens dos Métodos Diretos

A Tabela 4.1 apresenta, de forma compacta, as principais características dos algoritmos SLP, GRG e SQP. Os algoritmos SLP e SQP normalmente geram pontos com grandes violações, causando dificuldades, especialmente em modelos com funções logarítmicas ou expressões fracionárias, devido ao fato de que estes podem provocar mau condicionamento ao sistema (EDGAR et al., 2001). Exemplos de processos químicos, onde essas características são ressaltadas, podem ser obtidas em Sarma e Reklaitis (1982).

Claramente, todos os três algoritmos têm vantagens que habilitam o uso em determinada situação. Para problemas de dimensões razoáveis, o algoritmo SLP se mostra confiável e eficaz. Contudo, para problemas de grande dimensão, versões do GRG e do SQP, são amplamente empregadas.

Tabela 4.1: Vantagens e desvantagens dos algoritmos SLP, GRG e SQP.

Algoritmo	Vantagens	Desvantagens
SLP	✓ Rápida convergência quando o ótimo está em um vértice; ✓ Não tenta satisfazer restrições de igualdade a cada iteração.	✓ Enfrenta dificuldades quando o ótimo não está em um vértice; ✓ Usualmente viola restrições não lineares até a convergência, frequentemente por valores elevados.
GRG	✓ Provavelmente é o mais robusto entre os três algoritmos; ✓ É versátil, principalmente no tratamento de problemas sem restrições ou com restrições lineares; ✓ Uma vez encontrada uma solução viável, é mantida mesmo que soluções não viáveis sejam obtidas.	✓ Necessidade de satisfazer restrições a cada iteração.
SQP	✓ Usualmente requer poucas avaliações da função e do gradiente em comparação aos outros dois algoritmos; ✓ Não tenta satisfazer restrições de igualdade a cada iteração.	✓ Usualmente viola restrições não lineares até a convergência, frequentemente por valores elevados.

No próximo capítulo será apresentado um estudo sobre a resolução de POMO através de métodos heurísticos, com extensão para o contexto multi-objetivo.

ESTUDO DE POMO USANDO MÉTODOS HEURÍSTICOS

5.1 Aspectos Gerais

s métodos de otimização e busca heurística, que são baseados nos princípios e modelos de evolução biológica, têm recebido crescente interesse nas últimas décadas, devido principalmente a sua simplicidade conceitual e versatilidade na resolução de problemas realísticos (FONSECA; FLEMING, 1993; DEB, 2001; COELHO, 2004). Além disso, o desenvolvimento de modelos computacionais, inspirados nos mecanismos evolutivos, caracterizam-se pela configuração de algoritmos de otimização robustos (DEB, 2001). Neste sentido, será apresentada uma revisão sobre as principais metodologias evolutivas descritas na literatura.

O surgimento das técnicas heurísticas teve seu embasamento a partir das idéias desenvolvidas no século XIX, quando Darwin (1809-1882), assim como Lamarck (1744-1829), observaram a adaptação e a adequação entre a forma e o desempenho dos órgãos dos seres vivos e sua maneira de viver. A diferença é que Darwin apresentou em suas pesquisas um mecanismo de evolução baseado na seleção natural. Em síntese, Lamarck propôs que mudanças ambientais encontradas durante a vida de um organismo causam mudanças estruturais, que são transmitidas ao seu descendente (DIAS et al., 1998).

No século XIX, Mendel (1822-1884) propôs uma teoria de hereditariedade baseada em genes que se encontravam nos gametas. Atualmente, os fatores mendelianos são denominados genes. As duas leis de Mendel procuram explicar a troca de material genético através de operações de cruzamento e mutação. O enunciado das leis de Mendel é apresentado a seguir (LOPES, 2002):

◇ 1ª LEI DE MENDEL: “*Cada caráter é condicionado por um par de genes alelos que se segregam entre si, com a mesma probabilidade na formação dos gametas, indo apenas um gene para cada gameta*”.

◇ 2ª LEI DE MENDEL: “*Um par de alelos localizados em um par de cromossomos homólogos separa-se independentemente de outro par de alelos localizados em outro par de cromossomos homólogos*”.

Os aspectos relacionados à inspiração biológica da computação evolutiva e o aprendizado estão fundamentados nos estudos de Charles Darwin, Jean-Baptiste de Lamarck, Gregor Mendel, James Baldwin, Hugo De Vries, Carl Correns, Erich von Tschermack, entre outros (COELHO, 2003).

De forma análoga ao processo evolutivo natural, os Algoritmos Evolutivos (AE) trabalham com um conjunto de soluções potenciais, que competem entre si durante o processo evolutivo em direção à solução ótima. Tal comportamento é descrito pelo princípio da teoria evolucionista de Darwin denominado “sobrevivência dos mais aptos”. O nível de sucesso das soluções está associado ao grau de adaptação, medido por meio de uma função, denominada função de adaptação. Esta é usada como critério para ordenar uma sequência de soluções, comparar soluções individuais e selecionar as mais aptas para futuras avaliações. O foco original de pesquisa destes algoritmos é o uso do princípio da evolução para simular a natureza adaptativa de processos naturais em sistemas artificiais (SILVA, 2004).

As abordagens correntes dos AEs descendem dos princípios de diferentes metodologias (COELHO, 2004; CASTRO, 2001):

i) Algoritmos Genéticos → são algoritmos que operam com uma população de indivíduos representados por cromossomos, os quais durante o processo de evolução são submetidos ao procedimento de seleção e reprodução, onde são aplicados os operadores de recombinação e mutação, que são inspirados nos mecanismos naturais da genética (HOLLAND, 1975; GOLDBERG, 1989);

ii) Programação Evolutiva → concebida originalmente por Lawrence J. Fogel, em 1960, como uma estratégia de otimização estocástica similar aos algoritmos genéticos. Contudo, enfatizam o relacionamento entre genitores e seus descendentes ao invés de tentar emular operadores genéticos específicos observados na natureza ¹;

iii) Estratégias Evolutivas → foram concebidos para o estudo de problemas de engenharia como alternativa aos métodos convencionais. Estes operam com cromossomos na forma de vetores de números reais e originalmente na proporção (1+1), isto é, cada genitor gera um herdeiro por geração, normalmente por mutações distribuídas;

¹Fonte: <http://www.kneehighs.com/intro.html>, acessado em 11/06/2006.

iv) Programação Genética $\longrightarrow$ opera sobre representações de trechos de programas na forma de árvores, de modo que possam ser combinados para gerarem novos trechos de programas mais complexos;

v) Estratégias de Colônia $\longrightarrow$ é inspirada no comportamento de enxames, ou seja, no comportamento coletivo de colônias de insetos, pássaros ou de peixes. Estas operam com a idéia da comunicação indireta explorada pelas sociedades de insetos, que formam algoritmos distribuídos de multi-genes.

De forma geral, os Algoritmos Evolutivos possuem a seguinte estrutura (CASTRO, 2001):

Tabela 5.1: Algoritmo evolutivo conceitual.

Algoritmo Evolutivo Conceitual
Inicialização da população
Avaliação dos indivíduos na população
Repetição (<i>Evolução</i>)
Seleção dos indivíduos da população
Alteração desses indivíduos para criar uma nova população
Avaliação da nova população
Até que um critério de parada seja satisfeito
Fim

Goldberg (1989), em seu célebre livro “*Genetic Algorithms in Search, Optimization, and Machine Learning*”, fez com que os AGs se tornassem populares de tal modo que, apesar de quase duas décadas, esta seja uma das abordagens mais pesquisadas e empregadas dentre todas as metodologias existentes e emergentes para a resolução dos problemas de otimização.

Este capítulo apresenta, em linhas gerais, as principais características e abordagens existentes para a solução de POMO via AGs. Além disso, é apresentada a idéia básica das principais técnicas emergentes e de sua extensão para problemas de otimização multi-objetivo, tais como Recozimento Simulado, Enxame de Partículas, Colônia de Formigas e Evolução Diferencial, dando ênfase a essa última.

5.2 Algoritmos Genéticos

O surgimento dos Algoritmos Genéticos (AGs) se deu a partir de uma abstração do livro “*Adaptation in Natural and Artificial Systems*” de John Holland (HOLLAND,

1975): “uma população de indivíduos para adaptar-se coletivamente em um ambiente, deve comportar-se como um sistema natural, onde a sobrevivência é promovida através da eliminação de comportamentos inúteis ou prejudiciais e recompensando-se os comportamentos úteis”.

Essa idéia permite a compreensão de que os mecanismos biológicos permitem a adaptação do sistema natural de forma que estes poderiam ser expressos matematicamente e simuladas computacionalmente. A conexão entre a busca, o problema de otimização e o algoritmo genético é o indivíduo, que representa uma solução viável em um espaço de busca (DEB, 2001).

De forma geral, os AGs transformam uma população de indivíduos, cada um com um valor associado de adaptação, numa nova geração de indivíduos usando princípios de reprodução e de sobrevivência dos mais aptos, pela aplicação de operações genéticas tais como recombinação e mutação. Cada indivíduo na população representa uma solução potencial para um dado problema. O que o AG faz é procurar aquela que seja muito boa ou a melhor seleção para o problema analisado, pela criação genética de populações de indivíduos cada vez mais aptos (CASTRO, 2001).

A Figura 5.1 apresenta a estrutura básica do mecanismo de evolução do AG ao longo de uma geração (SILVA, 2001; COELHO, 2003). Esta pode ser traduzida nas seguinte etapas (COELHO, 2003):

- ❶ criar uma população inicial composta por N indivíduos (candidatos a ser solução do problema). Cada uma das soluções consiste de vetores $x_i \in (0,1)$ (representação binária) ou $x_i \in \mathbb{R}$ (representação real);
- ❷ classificar cada solução x_i , $i = 1, \dots, N$, com relação ao cálculo da função de adaptação;
- ❸ selecionar os indivíduos mais aptos de acordo com uma estratégia de seleção;
- ❹ aplicar o operador genético de cruzamento (representação binária) ou recombinação (representação por ponto flutuante);
- ❺ aplicar o operador genético de mutação;
- ❻ gerar uma nova população;
- ❼ repetir as etapas 2 a 6 até que um critério de convergência seja satisfeito.

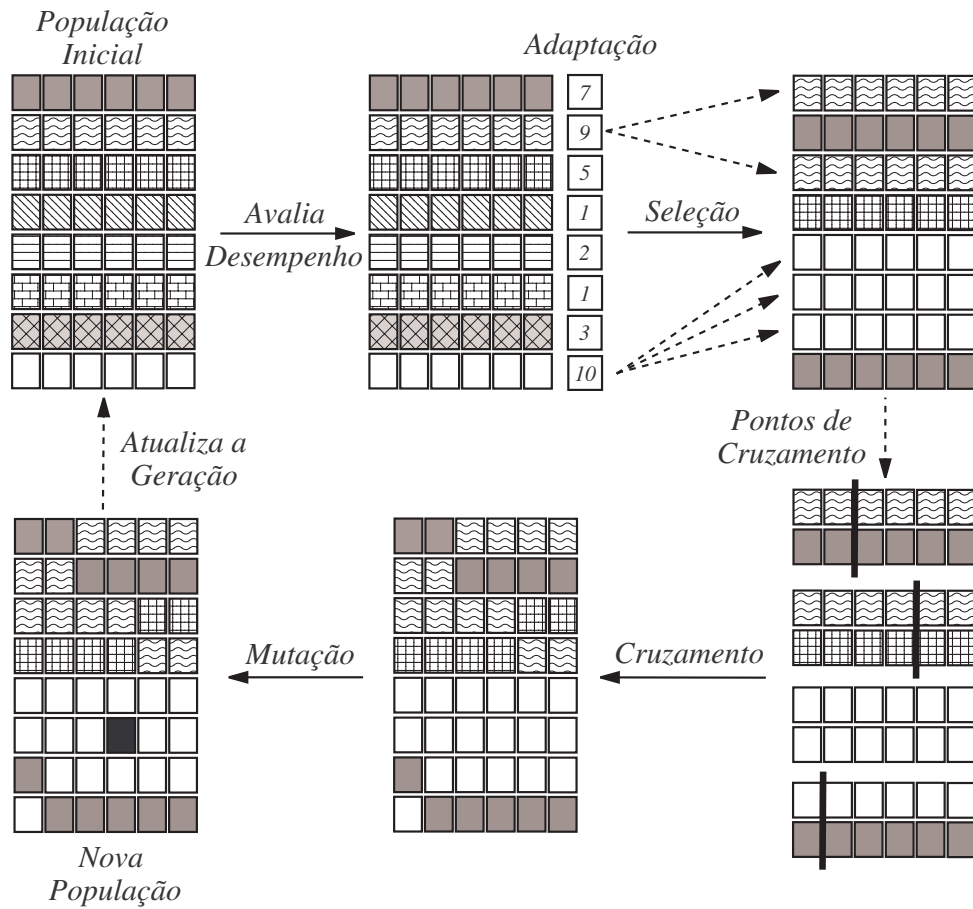


Figura 5.1: Esquema geral de evolução de um AG (Reproduzido de Silva (2001)).

5.2.1 Codificação dos Algoritmos Genéticos

Em sua concepção original, os AGs fazem uso de codificação binária para a representação dos indivíduos (cada cromossomo é um vetor composto por zeros e uns), sendo ideal para o tratamento de problemas com variáveis discretas (HAUPT; HAUPT, 1998). Além disso, segundo os autores, para o manuseio de variáveis contínuas faz-se necessária a codificação das variáveis, que pode ser feita, por exemplo, do seguinte modo (HAUPT; HAUPT, 1998):

$$x_i = x_{i,min} + \frac{B_i}{2^{ni}}(x_{i,max} - x_{i,min}) \quad (5.1)$$

onde B_i é a relação entre o valor real da variável x_i e a sua representação binária de caracteres, n é o número de caracteres e $x_{i,min}$ e $x_{i,max}$ são os vetores que representam os limites da variável i , respectivamente.

Segundo Castro (2001), a representação ou codificação das variáveis de projeto a serem otimizadas proporciona um grande impacto no desempenho de busca, devendo ser o mais

simples possível sem perder, no entanto, as características de representação do problema tratado.

5.2.2 Geração da População Inicial

Um ponto interessante quando se trabalha com abordagens heurísticas é a maneira como a população inicial é gerada. A grande maioria dos algoritmos inicializam essa população de forma randômica. Contudo, existem abordagens que geram a população inicial através de distribuição gaussiana, como por exemplo no trabalho de Angira e Babu (2005). De modo geral, a inicialização de um problema não é crítica, desde que a população inicial contenha cromossomos suficientemente variados (GOLDBERG, 1989; CASTRO, 2001).

5.2.3 Avaliação da População

A avaliação da população é realizada através da função de adaptação, que deve indicar a “*qualidade*” de cada indivíduo na população (CASTRO, 2001). Com a evolução do processo, os indivíduos passam a ter aptidões cada vez mais semelhantes entre si, podendo ser necessário aumentar a pressão de seleção, adotando alguma estratégia própria para fazê-lo (GOLDBERG, 1989).

Silva (2004) calcula o valor da aptidão através do conceito de função de transferência, conforme definido em seu trabalho. A função de transferência refere-se a uma medida da posição relativa do ponto no espaço de classificação, diferentemente de outros métodos, que a consideram simplesmente igual ao valor da função objetivo. Assim, segundo o autor, as condições do ponto em relação às restrições do problema devem ser incluídas no valor da sua função adaptação sob a forma de penalidade. Além disso, ainda neste trabalho, se fez uso de um método de penalização baseado na teoria da lógica nebulosa, com o intuito de transformar um problema com restrições em um outro, sem restrições.

5.2.4 Operadores Genéticos

Conforme comentado anteriormente, os algoritmos genéticos possuem procedimentos probabilísticos de busca, baseados nos princípios decorrentes da dinâmica das populações naturais. Nestes operadores uma população de soluções candidatas é aleatoriamente gerada e “evolui” para uma solução, através da aplicação de operadores genéticos. Esta seção aborda os principais operadores genéticos e suas variações mais utilizadas.

i) Reprodução:

O operador de seleção emprega o princípio de sobrevivência dos indivíduos mais aptos, através de uma metáfora dos procedimentos de reprodução assexuada e seleção natural, de acordo com o grau de adaptação do indivíduo ao ambiente (COELHO, 2003). Este operador é a versão artificial da seleção das espécies do Darwinismo, que estabelece que os seres mais aptos têm maiores chances de sobreviver, ou seja, os mais fortes e menos vulneráveis aos predadores e doenças (TICONA, 2003). O objetivo principal do operador é copiar as melhores soluções, eliminando as de baixa aptidão, enquanto o tamanho da população é constante (DEB, 2001). Essa estratégia pode ser realizada através dos seguintes passos:

- *Identificar as melhores soluções na população;*
- *Realizar cópias dessas melhores soluções;*
- *Eliminar as soluções de baixa aptidão da população.*

Existe um grande número de estratégias de seleção, dentre estas podemos destacar (DEB, 2001):

a) Seleção Proporcional: nessa estratégia, o número de cópias de uma solução na lista de soluções escolhidas é proporcional ao seu valor de aptidão. Para calcular o número de cópias esperado, é necessário obter a probabilidade de cada solução:

$$p_i = \frac{F_i}{\sum_{j=1}^N F_j} \quad (5.2)$$

sendo que F_i é a aptidão da solução i e N é o tamanho da população. O número de cópias na lista de soluções é calculado por $C_i = p_i N$, isto é, as soluções com melhor valor de aptidão terão mais cópias na lista de soluções. O escalonamento é um problema associado com esta estratégia. Quando existe uma solução com um valor de aptidão muito maior comparado com o restante da população, esta “super-solução” terá uma probabilidade de escolha perto de 1, e terá muitas cópias na lista de soluções. Caso todas as soluções possuam valores similares de aptidão, terão a mesma probabilidade de serem escolhidas, e cada uma será copiada na lista de soluções, o que equivale a não se realizar a operação de seleção;

b) Seleção Elitista com Truncamento: o modelo de seleção elitista é normalmente acoplado a outros métodos de seleção, na tentativa de se aumentar a velocidade de convergência do algoritmo, bem como em aplicações onde possa ser necessário o seu emprego. Este processo simplesmente copia os N ($N \geq 1$) melhores indivíduos da população corrente para a próxima geração, garantindo que estes cromossomos não sejam destruídos

nas etapas de recombinação e mutação. Na maioria das implementações, normalmente, pelo menos o elitismo do melhor indivíduo é utilizado. A principal vantagem do elitismo é garantir que não se perca o melhor indivíduo durante o processo evolutivo e, com isto, gerar uma seqüência, onde o resultado não piora. Sua desvantagem é a possibilidade de forçar a busca, pela presença de mais uma cópia do melhor indivíduo, na direção de algum ponto ótimo local que tenha sido descoberto antes do global, embora um algoritmo genético possa escapar de tais armadilhas. Alternativamente, pode-se guardar separadamente a melhor solução encontrada durante a evolução, para no final da execução designá-la como sendo o indivíduo ótimo encontrado, mesmo que ele não esteja presente na última geração da execução;

c) *Seleção por Torneio*: na seleção por torneio, são realizadas várias competições entre duas soluções, e a melhor solução é copiada na lista de soluções. Este processo é repetido até se preencher a lista. Goel e Deb (2001) mostraram que este método possui uma convergência igual ou melhor que outras estratégias de seleção, além de possuir uma complexidade computacional menor;

d) *Seleção por Ranking*: ordena as soluções conforme seu valor de aptidão, desde a pior solução (*ranking* 1) até a melhor (*ranking* N). Depois, o número de cópias das soluções é proporcional ao valor desse *ranking*.

ii) Cruzamento:

Este operador gera novas soluções (filhas) a partir da lista de soluções corrente (pais). Geralmente o cruzamento possui uma probabilidade associada com essa mudança, usando-se um valor próximo de 1. A forma mais simples de cruzamento é conhecida como cruzamento em um ponto, ilustrada na Fig. 5.2 (TICONA, 2003).

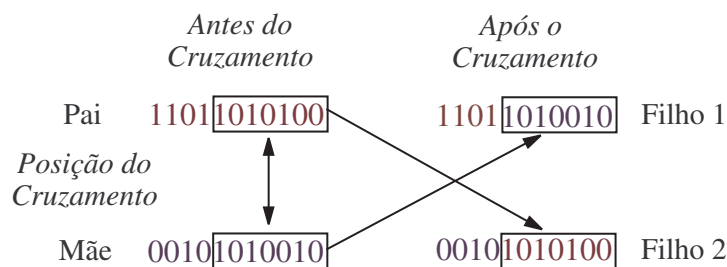


Figura 5.2: Operador de cruzamento (Reproduzido de Ticona (2003)).

Este tipo de cruzamento consiste em:

- Escolher arbitrariamente dois indivíduos no conjunto de soluções;
- Escolher dentro da cadeia o indivíduo da posição k chamada posição de cruzamento;

- Criar novos descendentes trocando as cadeias parciais de cada um dos indivíduos.

iii) Mutação:

O operador de mutação é necessário para se introduzir e manter a diversidade genética na população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida, assim como ilustrado na Fig. 5.3.

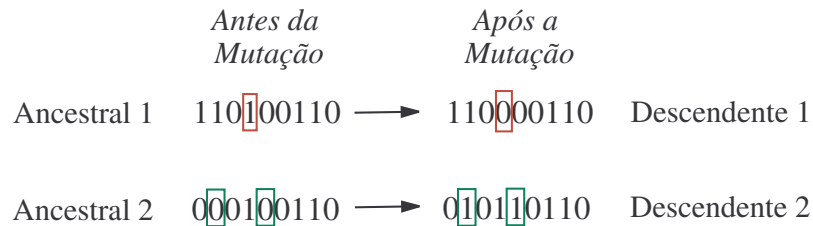


Figura 5.3: Operador de mutação (Reproduzido de Ticona (2003)).

Dessa maneira, esse operador nos fornece meios para a incorporação de novos elementos na população, assegurando que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca seja zero, com o intuito de tentar contornar o problema de ótimos locais (CASTRO, 2001). O operador de mutação é aplicado aos indivíduos com uma probabilidade dada pela taxa de mutação p_m . Segundo Deb (2001), geralmente se utiliza uma taxa de mutação pequena (como na genética natural), pois é um operador genético secundário.

iv) Elitismo:

Esse operador consiste em manter as melhores soluções encontradas nas gerações anteriores nas gerações posteriores, o que não provoca perda de possíveis candidatos ao ponto ótimo. Segundo Castro (2001), uma forma de implementar o elitismo é copiar diretamente as $n\%$ das soluções da população atual na população seguinte. O restante das $(100-n)\%$ soluções é gerada usando os operadores genéticos usuais sobre a população atual. Desta forma, as melhores soluções passam diretamente para a população seguinte, além de participar da criação do restante das soluções. Outra forma de elitismo consiste em criar a população seguinte a partir da população atual usando os operadores genéticos usuais, e escolher as melhores N soluções de ambas populações (TICONA, 2003).

5.2.5 Algoritmos Genéticos para Otimização Multi-objetivo

O primeiro Algoritmo Genético Multi-objetivo, denominado de VEGA - *Vector Evaluated Genetic Algorithm*, foi proposto por Schaffer (1984). A partir desse, inúmeros algo-

ritmos evolutivos baseados em AG foram desenvolvidos. Dentre os mais populares e inovadores pode-se citar: MOGA - *Multi-objective Genetic Algorithm* (FONSECA; FLEMING, 1993), WBGA - *Weight-based Genetic Algorithm* (HAJELA; LIN, 1992), NPGA - *Niched Pareto Genetic Algorithm* (HORN et al., 1994), RWGA - *Random Weighted Genetic Algorithm* (MURATA; ISHIBUCHI, 1995), PESA - *Pareto Envelope-based Selection Algorithm* (CORNE et al., 2000; CORNE et al., 2001), PAES - *Pareto-Archived Evolution Strategy* (KNOWLES; CORNE, 2000), NSGA - *Nondominated Sorting Genetic Algorithm* (SRINIVAS; DEB, 1994), NSGA II - *Nondominated Sorting Genetic Algorithm II* (DEB et al., 2000), SPEA - *Strength Pareto Evolutionary Algorithm* (ZITZLER; THIELE, 1998), SPEA II - (ZITZLER et al., 2001), DMOEA - *Dynamic Multi-objective Evolutionary Algorithm* (YEN; LU, 2003), PMOGA - *Pareto Multi-objective Genetic Algorithm* (CASTRO, 2001), MOGA I² - *Multi-objective Genetic Algorithm* (SILVA, 2004), PAMUC - *Preferences Applied to Multiobjectivity and Constraints* (COELHO, 2004).

Embora existam várias possibilidades para otimizadores baseados em Algoritmos Genéticos na literatura, conforme citados acima, além de serem algoritmos bem conhecidos, têm seu desempenho já estudado em vários trabalhos (ZITZLER; THIELE, 1998; DEB, 1998; DEB et al., 2000; ZITZLER et al., 2001; DEB, 2001; CASTRO, 2001; SILVA, 2004). De uma forma geral, esses algoritmos se diferem pelo modo como é feito o procedimento de aptidão, pelo mecanismo de elitismo e diversidade.

A Tabela 5.2 lista os principais MOEA (*Multi-objective Evolutionary Algorithm*), com suas respectivas características, bem como suas vantagens e desvantagens.

²Originalmente o algoritmo desenvolvido por Silva (2004) não foi denominado MOGA I, sendo utilizado aqui para diferenciá-lo dos outros.

Tabela 5.2: Algoritmos evolutivos multi-objetivos baseados em algoritmos genéticos.

Algoritmo	Tarefa de Aptidão	Mecanismo de Diversidade	Elitismo	População Externa	Vantagens	Desvantagens
VEGA	cada sub-população é avaliada de acordo com cada objetivo	não possui	não possui	não possui	foi a primeira implementação de MOGA	tende a convergir para o extremo de cada objetivo
MOGA	ranking de Pareto	aptidão por comparação por nicho	não possui	não possui	extensão simples do AG para um único objetivo	usualmente possui baixa taxa de convergência e forte dependência do parâmetro do nicho
WPGA	média ponderada dos objetivos normalizados	pré-definição dos pesos	não possui	não possui	extensão simples do AG para um único objetivo	dificuldades em problemas não convexos
NPGA	seleção por meio de torneio	o nicho conta como o vínculo na seleção por torneio	não possui	não possui	processo de seleção muito simples com seleção por torneio	dependência com o parâmetro tamanho do nicho e parâmetro da seleção por torneio
RWGA	média ponderada dos objetivos normalizados	pesos obtidos randomicamente	possui	possui	eficiente e de fácil implementação	dificuldades em problemas não convexos
PESA	sem tarefa de aptidão		elitismo puro	possui	fácil implementação	elevado custo computacional

Continua na próxima página

Continuação da Tab. 4.2

Algoritmo	Tarefa de Aptidão	Mecanismo de Diversidade	Elitismo	População Externa	Vantagens	Desvantagens
PAES	<i>a dominância de Pareto é utilizada para substituir um parente se o filho o dermina</i>	<i>densidade baseada no conceito de célula</i>	possui	possui	<i>mutação randômica e de fácil implementação</i>	<i>não é uma abordagem baseada em populações e seu desempenho depende da célula</i>
NSGA	<i>ranking baseado em organização de indivíduos não dominantes</i>	<i>adaptação por compartilhamento de nicho</i>	não possui	não possui	<i>rápida convergência</i>	<i>problemas com o parâmetro tamanho do nicho</i>
NSGA II	<i>ranking baseado em organização de indivíduos não dominantes</i>	<i>conceito de distância aglomerada</i>	possui	não possui	<i>bem testada e eficiente</i>	<i>a distância aglomerada trabalha somente com espaços de objetivos</i>
SPEA	<i>ranking baseado no arquivamento externo das soluções não-dominadas</i>	<i>truncamento da população externa</i>	possui	possui	<i>bem testada e sem a adição de mais parâmetros</i>	<i>a estratégia de truncamento não é trivial</i>
SPEA II	<i>baseado na força dos dominadores</i>	<i>densidade baseada na k-ésima vizinhança mais próxima</i>	possui	possui	<i>assegura que os pontos extremos não sejam perdidos</i>	<i>é computacionalmente caro</i>
DMOEA	<i>ranking baseado em células</i>	<i>densidade baseada em células adaptativas</i>	possui	não possui	<i>melhoria na adaptação da densidade das células</i>	<i>de difícil implementação</i>

Continua na próxima página

Continuação da Tab. 4.2

Algoritmo	Tarefa de Aptidão	Mecanismo de Diversidade	Elitismo	População Externa	Vantagens	Desvantagens
PMOGA	utiliza o conceito de dominância	operadores de adição, exclusão e otimização individual	operador filtro	não possui	rápida convergência tratamento de problemas discretos	não trabalha com codificação real
MOGA I ^a	baseado no conceito de função de transferência acoplada ao de dominância	operadores de adição, exclusão	operador filtro	não possui	rápida convergência baixo tempo de processamento	não trabalha com problemas de programação mista-inteira
PAMUC	faz uso das funções em sua forma original acoplado ao Método da Soma Ponderada, onde os pesos são atualizados a cada geração	não possui	não possui	não possui	incorporação de módulos baseados no conceito de inteligência artificial	não possui mecanismo de elitismo nem de diversidade

^aOriginalmente o algoritmo desenvolvido por Silva (2004) não foi denominado MOGA I, sendo utilizado aqui para distingui-lo dos outros.

Vistas as principais implementações baseadas em Algoritmos Genéticos, a seguir serão apresentados os princípios básicos de alguns desses algoritmos, focalizando suas vantagens, desvantagens e principais contribuições.

a) VEGA (*Vector Evaluated Genetic Algorithms*)

Schaffer (1984) desenvolveu o primeiro algoritmo multi-objetivo baseado em AG, o VEGA - *Vector Evaluated Genetic Algorithms*, cuja implementação fundamenta-se na modificação do *software* GENESIS através da criação de um laço no procedimento de seleção original, fazendo com que o procedimento seja repetido para cada um dos objetivos separadamente, até atingir-se um determinado número pré-definido de indivíduos para cada objetivo, para reprodução. A seguir, esses indivíduos são randomicamente sorteados para as etapas de recombinação e mutação. A principal limitação dessa metodologia é que, ao se fazer a seleção independente dos indivíduos, isso provoca a especialização da população, tendo como resultado a convergência na direção da região das soluções ótimas individuais após um grande número de gerações. Tal característica não é interessante, já que não adianta uma solução apresentar alta qualidade em um objetivo, se conseguida às custas de valores ruins ou inaceitáveis para algum(ns) outro(s) objetivo(s) (CASTRO, 2001). Schaffer tentou minimizar os efeitos da especialização através do desenvolvimento de dois procedimentos heurísticos de seleção que foram denominados: seleção não-dominada, na qual os indivíduos dominados são penalizados pela subtração de uma pequena penalidade fixa sobre o número esperado de cópias durante a seleção, e a seleção cruzada, que promove o cruzamento de indivíduos especializados de diferentes subgrupos. Entretanto, ambos os procedimentos não funcionam satisfatoriamente por não prevenir a participação de indivíduos piores na primeira seleção randômica ou pela possibilidade de haver uma grande distância Euclidiana entre um indivíduo campeão e um medíocre (SCHAFFER, 1984).

b) MOGA (*Multi-Objective Genetic Algorithm*)

Fonseca e Fleming (1993) desenvolveram o MOGA, que utiliza o conceito de dominância e diversidade para as soluções obtidas em cada geração. Essa implementação se diferencia de outras através da maneira pela qual se atribui o valor de aptidão às soluções da população, associando a cada solução potencial um valor de *ranking* que é igual ao número de soluções n_i que a dominam mais um, assim:

$$r_i = 1 + n_i \quad (5.3)$$

Dessa maneira, as soluções não-dominadas possuem *ranking* 1, o que faz com que pelo menos um indivíduo da população possua $r_i = 1$, e o valor máximo de r_i não ultrapasse

o tamanho da população (N).

As Figuras 5.4(a) e 5.4(b) mostram um conjunto de soluções e seus valores r_i , respectivamente.

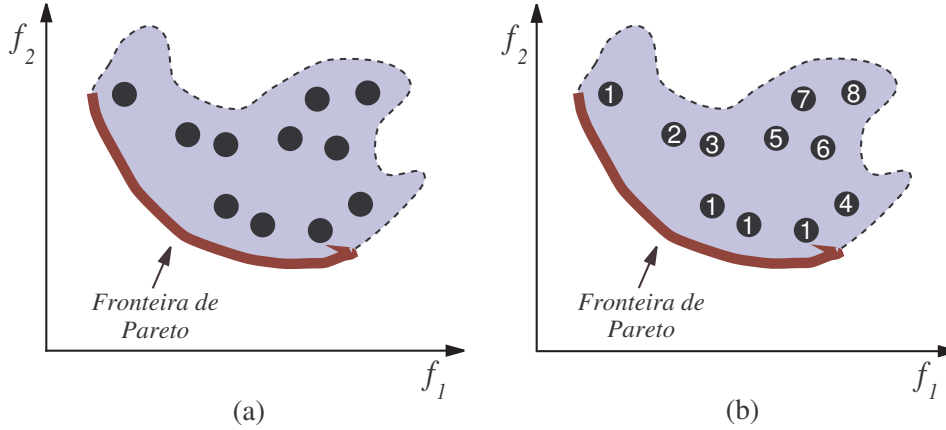


Figura 5.4: *Ranking* obtido através da aplicação do MOGA (Reproduzido de Deb (2001)).

No próximo passo do algoritmo, a população é ordenada conforme esse valor de r_i . Posteriormente, o valor de aptidão média para as soluções de mesmo *ranking* é calculado, permitindo que as soluções com melhor *ranking* tenham valores de aptidão mais altos. A diversidade entre soluções não-dominadas é obtida utilizando o conceito de nichos para cada *ranking*. Uma vez obtidos os nichos, calcula-se a aptidão compartilhada de cada solução no *ranking* $r = 1$, no $r = 2$ e assim sucessivamente. Finalmente, este valor é multiplicado por um fator de conversão de escala.

$$F'_i = \frac{\mu(r_i)F_i}{\sum_{j=1}^{\mu(r_i)} F'_j} F'_i \quad (5.4)$$

onde $\mu(r_i)$ é o número de soluções no *ranking* r_i .

Logo após esses cálculos, o MOGA comporta-se como um AG simples, com operadores de seleção, cruzamento e mutação usuais.

O compartilhamento da aptidão é realizado no sentido de distribuir as soluções sob diferentes regiões ou nichos no espaço de busca. A Figura 5.5 ilustra um conjunto de soluções distribuídas em vários nichos (círculos).

Para cada solução i é calculado um valor de contador de nicho nc_i usando Eq. (5.5) e Eq. (5.6), que caracterizam a região que i pertence.

$$nc_i = \sum_{j=1}^{\mu(r_i)} Sh(d_{ij}) \quad (5.5)$$

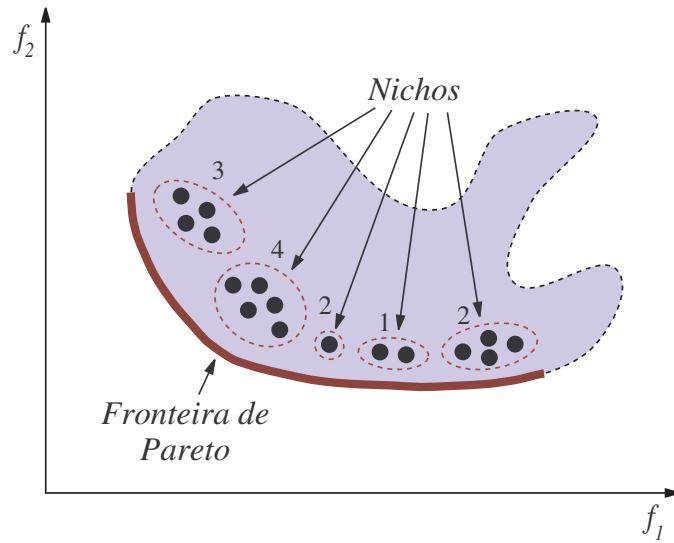


Figura 5.5: Soluções agrupadas em nichos (Reproduzido de Deb (2001)).

$$d_{ij} = \sqrt{\sum_{j=1}^m \left(\frac{f_k^i - f_k^j}{f_k^{max} - f_k^{min}} \right)^2} \quad (5.6)$$

onde d_{ij} representa a distância entre duas soluções i e j que possuem o mesmo valor de *ranking* r_i , f_k^{max} e f_k^{min} são os valores mínimos e máximos para a k -ésima função objetivo, respectivamente. A função Sh da Eq. (5.7) é conhecida como função de compartilhamento, que foi originalmente proposta por (GOLDBERG, 1989), sendo definida como:

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}} \right)^\alpha & \text{se } d_{ij} \leq \sigma_{share} \\ 0 & \text{caso contrário} \end{cases} \quad (5.7)$$

onde o parâmetro α define o comportamento da função Sh e σ_{share} é chamado de raio de nicho. O valor do raio de nicho define a vizinhança de uma solução, sendo que quando $d_{ij} > \sigma_{share}$ as soluções estão em nichos separados e $Sh(d_{ij}) = 0$. Caso contrário, $Sh(d_{ij})$ assume um comportamento decrescente, de acordo com a relação d_{ij}/σ_{share} . Desta forma, quanto maior o número de soluções dentro do nicho, maior o valor de nc . Além disso, o MOGA usa a função Sh com $\alpha = 1$ e o parâmetro σ_{share} é atualizado dinamicamente (FONSECA; FLEMING, 1993). A aptidão compartilhada para cada solução i é:

$$F'_i = \frac{F_i}{nc_i} \quad (5.8)$$

Dessa maneira as soluções que residem em um nicho menos ocupado terão melhor

aptidão de compartilhamento, destacando soluções pouco representativas em cada *ranking*.

A estrutura do Algoritmo MOGA é apresentado a seguir:

Parâmetros de Entrada: *População* (P), *Tamanho da população* (N) e *Parâmetro de compartilhamento* (σ_{share})

- ❶ Inicializar os contadores $\mu_j = 0$, para todos os *rankings* possíveis j , $j = 1 \dots N$;

para cada solução i em P ;

- ❷ Calcular o número de soluções n_i que dominam a solução i ;
- ❸ Obter r_i conforme Eq. (5.3);
- ❹ Incrementar o contador μ_{r_i} ;
- ❺ Ordenar P de forma decrescente por r_i , identificar o valor máximo $r^* = \max(r_i)$, para cada solução i em P , calcular a aptidão média para as soluções em P conforme Eq. (5.4), fazer $r = 1$;
- ❻ Para cada solução i com $r_i = r$: calcular o contador de nicho nc_i utilizando Eq. (5.5);
- ❼ Determinar a aptidão final utilizando Eq. (5.4);
- ❽ Se $r < r^*$ fazer $r = r + 1 \rightarrow$ ir para o sexto passo, senão terminar.

Saída: *Soluções Não-dominadas*.

Segundo Deb (2001), a principal vantagem do MOGA é sua simplicidade no cálculo do valor de adaptação. Segundo Coelho (2003), o MOGA tem desempenho melhor comparado com outros algoritmos não elitistas. Entretanto, além desse algoritmo possuir certa complexidade computacional, o cálculo da aptidão média, dos nichos e da aptidão final, no pior caso, quando todas as soluções estão no mesmo *ranking* e no mesmo nicho, é da ordem de $\mathcal{O}(N^2)$, sendo a complexidade do MOGA da ordem de $\mathcal{O}(MN^2)$ (DEB, 2001). Outro aspecto importante é que o cálculo do compartilhamento não garante que as soluções com valor de r_i alto terão pior aptidão que soluções com r_i baixo. Isto acontece porque podem existir muitas soluções próximas entre si com r_i baixo, sendo o valor de contador de nicho alto nesse caso e a aptidão compartilhada baixa. Isso faz com que não exista uma boa seleção de soluções de melhor r_i , degradando a convergência do algoritmo (DEB, 2001).

c) NPGA (*Niched Pareto Genetic Algorithm*)

O Algoritmo NPGA, que se fundamenta no conceito de não-dominância, foi proposto por Horn et al. (1994). Essa abordagem se caracteriza por não precisar calcular o valor de aptidão que destaque as soluções não-dominadas. Ao contrário, utiliza um método de seleção por torneio denominado de Torneio de Pareto. Durante esse torneio, dois candidatos potenciais i e j são escolhidos aleatoriamente na população P . Estes são comparados com um subconjunto T , cuja quantidade de elementos é menor que P . Nesse caso, podem acontecer os seguintes casos:

- Se a solução i domina o subconjunto T e a solução j é dominada por algum elemento de T , a solução i é a vencedora. Reciprocamente, se a solução j domina o subconjunto T e a solução i é dominada por algum elemento de T , a solução j é a vencedora.
- Caso contrário, se acontecer de ambas as soluções dominarem T , ou que ao menos uma solução em T domine i e j , calcula-se o contador de nicho para escolher a solução vencedora.

O cálculo do nicho para i e j é realizado sobre a nova população gerada (Q). No início do processo, Q é vazio, assim o cálculo de nicho não pode ser realizado e no caso de empate entre soluções candidatas, uma delas é escolhida aleatoriamente. As soluções ganhadoras p_1 e p_2 geram as soluções filhas c_1 e c_2 que são incluídas em Q . Nas próximas gerações, o cálculo de nicho para escolher a solução ganhadora é baseada na seguinte medida de distância:

$$d_{ik} = \sqrt{\sum_{m=1}^M \left(\frac{f_m^i - f_m^k}{f_m^{max} - f_m^{min}} \right)^2} \quad (5.9)$$

onde f_m^{max} e f_m^{min} são os valores mínimos e máximos da k -ésima função objetivo.

A estrutura do Algoritmo NPGA é apresentada a seguir:

Parâmetros de Entrada: *População pai* (P), *Tamanho da população* (N), *População filha* (Q), *Parâmetro de compartilhamento* (σ_{share}), *Tamanho do torneio* (t_{dom}), *Número de geração atual* (n) e *Número máximo de gerações* ($nMax$).

- ❶ Criar a população inicial P_o e a população filha Q_o vazia;

Fazer $t = 0$ e $i = 0$

- ❷ Realizar a seleção por torneio com os indivíduos i e $i+1$, o ganhador será o pai p_1 , $i = i+2$;

- ❸ Aplicar a seleção por torneio com os indivíduos i e $i+1$, o ganhador será o pai p_2 ;
- ❹ Fazer o cruzamento de p_1 e p_2 para gerar os filhos c_1 e c_2 ;
- ❺ Aplicar o operador de mutação;
- ❻ Atualizar a população filha $Q_n = Q_n \cup \{c_1, c_2\}$, $i = i+1$;
- ❼ Se $i < N$ ir para o terceiro passo;
- ❽ Se $|Q| = N/2$ então misturar a população P ; fazer $i = 1$, voltar para o segundo passo;
- ❾ Se $n < nMax$ terminar, senão fazer $P_{n+1} = Q_n$, $Q_{n+1} = \{\}$, e voltar para o segundo passo.

Saída: *Soluções Não-dominadas.*

Segundo Deb (2001), a principal vantagem do algoritmo NPGA é que não precisa de um método para calcular o valor de aptidão das soluções, ficando livre da subjetividade da função aptidão, diferentemente de outros algoritmos multi-objetivos. Outra vantagem é o uso do operador de seleção por torneio, sendo computacionalmente eficiente quando o parâmetro t_{dom} é muito menor que o tamanho da população, pois as comparações de dominância se realizam apenas em um subconjunto pequeno. O NPGA apresenta como desvantagem a incorporação de dois novos parâmetros σ_{share} e t_{dom} . Além disso, para o caso em que t_{dom} é pequeno comparado com N , tem-se que a complexidade dessa abordagem é governada pelo cálculo do nicho, sendo da ordem de $\mathcal{O}(N^2)$. Contudo, quando t_{dom} é um valor da mesma ordem de N , a complexidade estará determinada pelos torneios realizados, obtendo-se assim uma complexidade da ordem de $\mathcal{O}(MN^2)$ (DEB, 2001).

d) NSGA II (*Non-dominated Sorting Genetic Algorithm*)

O algoritmo NSGA II é uma extensão do NSGA desenvolvido por Deb et al. (2000), cuja idéia básica é a realização de um ordenamento elitista por não dominância. Este trabalha com uma população pai P para gerar a população filha Q , como nos AGs convencionais. Na primeira geração, gera-se uma população P_o ordenada por não dominância. Cada solução tem um valor de aptidão igual ao seu nível de não dominância, sendo que 1 é o melhor nível, 2 é o segundo melhor nível e assim por diante. Aplicando os operadores de seleção por torneio, cruzamento e mutação, obtém-se a população filha Q_o , de mesmo tamanho que P . Ambas as populações são reunidas em um conjunto R_o . Para as n gerações seguintes o algoritmo NSGA II trabalha com a população R_n .

Realiza-se um ordenamento por não-dominância sobre a t -ésima geração R_t , obtendo-se as fronteiras $F_1, F_2, \dots, F_n$, que são inseridas na nova população P_{n+1} . Dado que apenas N soluções podem ser inseridas, N soluções de R_n são descartadas de modo que deve-se inserir primeiramente as soluções em F_1 , em seguida F_2 e assim por diante, como apresentado na Fig. 5.6.

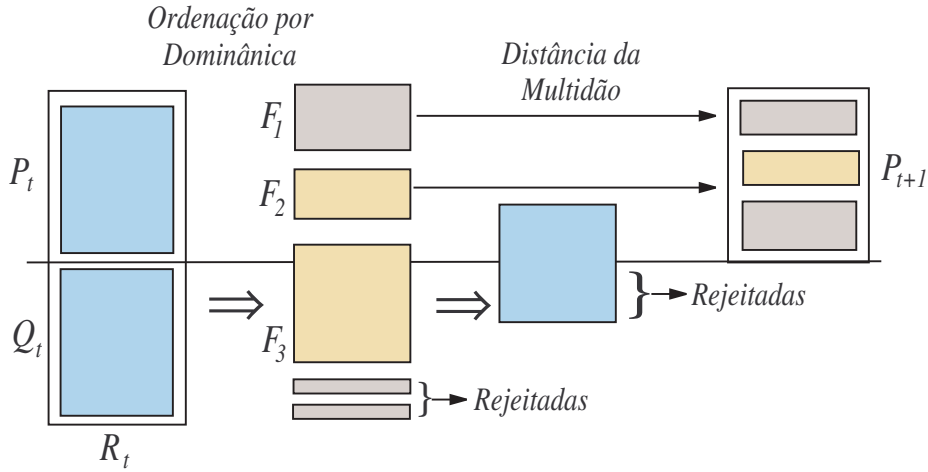


Figura 5.6: Estrutura do NSGA II (Reproduzido de Deb (2001)).

Cada conjunto F_i deve ser inserido na sua totalidade em P_{t+1} , o que acontece enquanto $P_{n+1} + |F_i| \leq N$. Ao inserir F_j tal que $|F_j| > N - P_{n+1}$, o algoritmo NSGA II escolhe aquelas soluções de F_j que estejam melhor espalhadas.

O algoritmo NSGA II introduz um método chamado de distância da multidão (*crowding distance*). Essa abordagem consiste em, após determinar as distâncias, ordenar os conjuntos F_j decrescentemente em relação às suas distâncias, copiando-se as primeiras $N - |P_{n+1}|$ soluções de F_j para P_{n+1} . Finalmente, gera-se Q_{n+1} a partir de P_{n+1} usando o operador de seleção de torneio por multidão, cruzamento e mutação.

O operador distância da multidão d_i de uma solução i representa uma estimativa do perímetro formado pelo cubóide cujos vértices são os seus vizinhos mais próximos. A Figura 5.7 mostra a distância da multidão para a solução i . Quanto maior o cubóide da solução i , mais afastada se encontra i dos seus vizinhos. Já as soluções extremas em cada objetivo terão um cubóide infinito.

O procedimento para achar a distância da multidão é descrito na Tab. 5.3, onde I_i^m representa a i -ésima solução na lista ordenada pelo objetivo m . I_1^m e I_l^m são os elementos da lista com menor e maior valor em um objetivo m . $f_m^{I_{i+1}^m}$ e $f_m^{I_{i-1}^m}$ são os valores dos vizinhos de i na m -ésima função objetivo. f_m^{max} e f_m^{min} são parâmetros dos limites máximo e mínimo em cada objetivo.

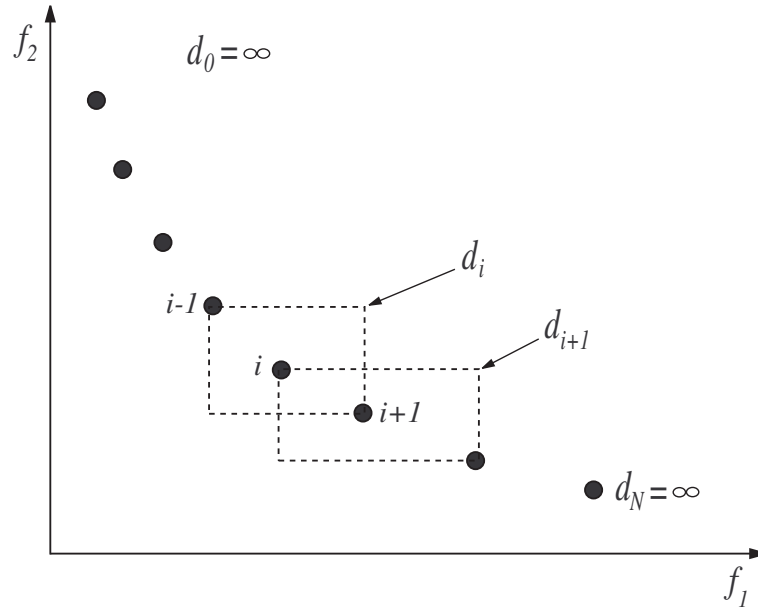


Figura 5.7: Esquema geral do operador distância da multidão - NSGA II (Reproduzido de Deb (2001)).

Tabela 5.3: Cálculo da distância da multidão - NPGA II.

<i>Parâmetros de Entrada:</i>
$\mathfrak{S}_j \rightarrow$ Conjunto de soluções na fronteira i
l denota o número de soluções em $\mathfrak{S}_j$
Para cada solução em $\mathfrak{S}_j$ atribui-se $d_i = 0$
Para cada função objetivo $m = 1, \dots, M$
Ordenar decrescentemente as soluções por f_m na lista I^m
Para cada solução extrema (mínimo e máximo) em cada um dos M objetivos fazer $d_{I^m} = \infty$
Para as soluções $i = 1, \dots, l - 1$ calcular d_{I^m} através de Eq. (5.10)

A Equação (5.10) assegura que as soluções mais afastadas tenham d_i maior. Alternativamente, é possível usar também o operador de compartilhamento para calcular o contador de nicho.

$$d_{I_i^m} = d_{I_i^m} + \frac{f_m^{I_{i+1}^m} - f_m^{I_{i-1}^m}}{f_m^{\max} - f_m^{\min}} \quad (5.10)$$

O operador de seleção por torneio de multidão do NSGA II incorpora uma pequena modificação no método de seleção por torneio, usando o operador comparativo que leva em conta a multidão de uma solução (*crowded tournament selection operator*) $< c$. Uma solução i é considerada ganhadora em um torneio contra uma solução j , se (DEB, 2001):

- A solução i possui um melhor nível de não dominância, $r_i < r_j$;
- Se ambas as soluções estão no mesmo nível, mas i tem uma distância da multidão maior, $d_i > d_j$.

A estrutura do Algoritmo NSGA II é apresentado a seguir:

Parâmetros de Entrada: *População pai (P), População filha (Q), Tamanho fixo para P e Q , Conjunto de soluções na fronteira j (F_j), Número máximo de gerações ($nMax$) e Número de geração atual (n).*

- ❶ Gerar a população inicial P_o e $Q_o = \{\}$;

Fazer $n = 0$

- ❷ Realizar a seleção, cruzamento e mutação para gerar a filha Q_o . Fazer $R_n = P_n \cup Q_n$;
- ❸ Realizar a ordenação por não dominância em R_n ;
- ❹ Criar $P_{n+1} = \{\}$;
- ❺ Enquanto $|P_{n+1} + F_j| \leq N$, copiar as soluções de F_j em P_{n+1} ;
- ❻ Calcular as distâncias da multidão em F_j , ordenando F_j conforme as distâncias d_j e copiando as primeiras $N - |P_{n+1}|$ soluções de F_j para P_{n+1} ;
- ❼ Aplicar seleção, cruzamento e mutação para gerar a nova população Q_{n+1} ;
- ❽ Se $n > nMax$ então pare, caso contrário atribuir $n = n + 1$ e voltar ao segundo passo.

Saída: *Soluções Não-dominadas.*

A principal vantagem do NSGA II é a maneira como mantém a diversidade entre as soluções não-dominadas. Além disso, não há a incorporação do parâmetro σ_{share} , como no MOGA (DEB, 2001). Sua complexidade computacional é a soma de três situações: para se ordenar R por não dominância é necessário comparar cada uma das $2N$ soluções contra $2N-1$ soluções em cada um dos M objetivos. Para obter as distâncias da multidão no pior caso, quando todas as soluções de R estão em F_1 , é necessário ordenar F_1 para cada objetivo, obtendo uma ordem de $\mathcal{O}(MN \log N)$. Finalmente, para passar as N melhores soluções de F_1 a P_{n+1} , ordena-se conforme o operador $<_c$, resultando em $\mathcal{O}(N \log N)$ comparações. Portanto a complexidade do algoritmo NSGA II é da ordem de $\mathcal{O}(MN^2)$ (DEB, 2001).

Um ponto negativo deste algoritmo é que, se o conjunto F_1 tem um tamanho maior que N , o processo de escolha de apenas N soluções, usando a distância da multidão, faz com que sejam perdidas soluções potenciais. Por exemplo, seja o conjunto F_1 onde existam várias soluções ótimas de Pareto muito próximas e alguma solução distante não ótima de Pareto, mas não-dominada no momento. Dado que o cubóide da solução não-dominada é maior, esta solução será copiada em $|P_{n+1}|$ enquanto uma solução ótima de Pareto é eliminada. Esse efeito é representado na Fig. 5.8(a). Depois de aplicar o algoritmo de corte para NSGA II, como mostrado na Fig. 5.8(b), elimina-se uma solução ótima de Pareto e se mantém a solução não-dominada (mas não ótima de Pareto). Esta situação faz com que o NSGA II possa cair em um ciclo de gerar soluções ótimas de Pareto e não ótimas de Pareto até convergir, finalmente, para um conjunto de soluções ótimas de Pareto (DEB, 2001).

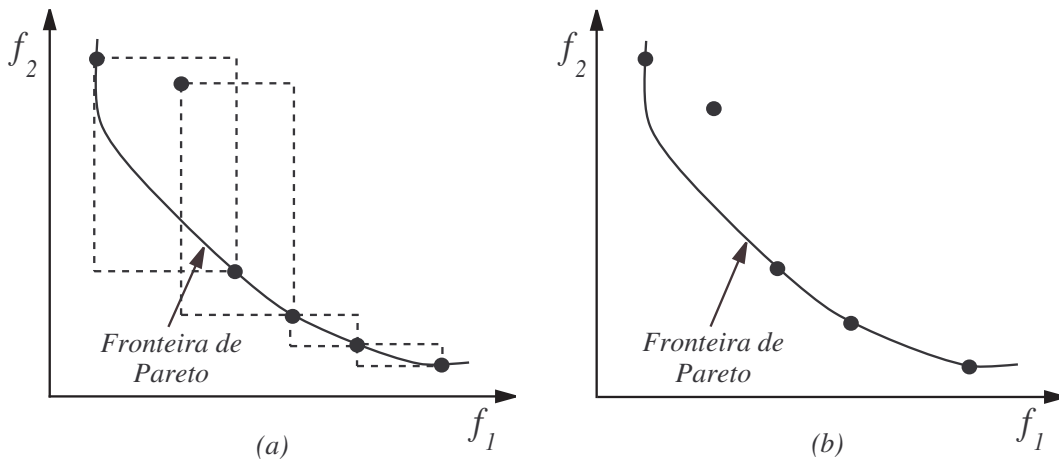


Figura 5.8: Situação onde o NSGA II falha (Reproduzido de Deb (2001)).

e) SPEA II (*Strength Pareto Evolutionary Algorithm*)

O algoritmo SPEA II, desenvolvido por Zitzler et al. (2001), faz uso da estratégia de elitismo através de uma população externa E onde são armazenadas todas as soluções não-dominadas. A população E tem tamanho (N) fixo definido pelo usuário. Inicialmente o SPEA II cria uma população aleatória P_o e uma população externa E (conjunto vazio). O valor de aptidão é dada através de Eq. (5.11):

$$s_i = |s_j, j \in Q, \text{ para } i \preceq j| \quad (5.11)$$

onde s_i é o número de soluções que i domina em Q . As soluções não dominadas são

assinaladas como $s_i = 0$. Logo em seguida, calcula-se o valor de r_i , através de:

$$r_i = \sum_{j \in Q, j \preceq i} s_j \quad (5.12)$$

Isto significa que r_i é a soma dos s_j das soluções j que dominam i em Q . Para as soluções não-dominadas, tem-se que $r_i = 0$. As soluções com um alto valor de r_i são dominadas por muitas soluções em Q . De forma esquemática, a Fig. 5.9 apresenta um conjunto de soluções e seus respectivos valores (r_i, s_i) .

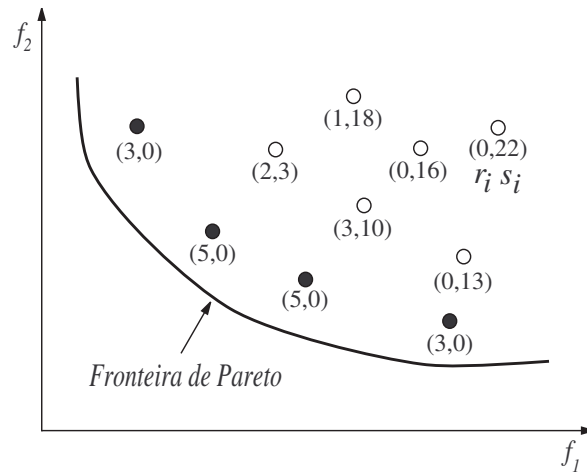


Figura 5.9: Cálculo da aptidão através do algoritmo SPEA II (Reproduzido de Deb (2001)).

Esse mecanismo permite a ordenação de soluções por dominância, mas com possibilidade de falha, quando existem muitas soluções não-dominadas. Neste caso, existiriam muitas soluções com $r_i=0$ e não se enfatizaria a preferência de uma solução sobre uma outra (ZITZLER et al., 2001). Para superar essa dificuldade, o algoritmo SPEA II usa informação sobre densidade, que é baseada no método de k -vizinhos, onde a densidade em qualquer ponto é uma função decrescente em relação ao k -ésimo ponto mais próximo. Para cada solução i em Q , obtém-se as distâncias aos $|Q| - 1$ indivíduos de Q . A densidade d_i é formulada como:

$$d_i = \frac{1}{\sigma_i^k + 2} \quad (5.13)$$

A densidade de i é inversamente proporcional à distância a seu k -vizinho mais próximo σ_i^k , onde $k=\sqrt{Q}$. Já a distância d_i está dentro do intervalo aberto $(0, 1)$, com aptidão para i dada por:

$$F_i = d_i + r_i \quad (5.14)$$

Através da relação acima, as soluções não dominadas terão $F_i < 1$, e as demais soluções $F_i \geq 1$. Uma vez calculado o valor de aptidão, copia-se as soluções não-dominadas de Q para a nova população externa E_{t+1} . O próximo passo dependerá de três possíveis situações:

- $|E_{n+1}| = N$, e não se fazem modificações sobre $|E_{t+1}|$;
- $|E_{n+1}| < N$, então ordena-se Q por F_i e copia-se as primeiras $N - |E_{n+1}|$ soluções i de Q , tal que $F_i \geq 1$;
- $|E_{n+1}| > N$, neste caso se utiliza um algoritmo de corte sobre $|E_{n+1}|$.

Finalmente, realiza-se o processo de seleção por torneio, cruzamento e mutação sobre E_{n+1} para gerar uma nova população P_{n+1} .

A redução do tamanho da população de E_{n+1} para N é feita através do algoritmo de corte. Este consiste, em cada geração, da escolha de uma solução tal que a sua distância em relação ao seu vizinho mais próximo seja a menor possível. No caso de empate, calcula-se a segunda menor distância, e assim por diante.

A estrutura do Algoritmo SPEA II é apresentada a seguir:

Parâmetros de Entrada: *População interna (P), População externa (E), Tamanho máximo para E (N), Número máximo de gerações ($nMax$) e Número de geração atual (n).*

Fazer $n = 0$.

- ❶ Gerar a população inicial P_o e $E_o = \{\}$;

Para cada solução i em $Q_n = P_n \cup E_n$.

- ❷ Calcular os valores s_i , r_i , d_i conforme Eq. (5.11), Eq. (5.12) e Eq. (5.13), respectivamente;
- ❸ Calcular o valor de aptidão F_i dada pela Eq. (5.14);
- ❹ Copiar as soluções não-dominadas de Q_n em E_{n+1} ;
- ❺ Se $|E_{n+1}| > N$: reduzir P_{n+1} utilizando o algoritmo de corte. Se $|E_{n+1}| < N$, ordenar Q_n conforme a F_i ;
- ❻ Copiar as melhores $N - |E_{n+1}|$ soluções de i de Q tal que $F_i \geq 1$;

- ⑦ Se $n > nMax$, então parar. Se não, fazer $n = n + 1$;
- ⑧ Aplicar a seleção para os indivíduos de E_{n+1} ;
- ⑨ Aplicar os operadores de cruzamento, mutação e gerar a nova população P_{n+1} , voltar para o segundo passo.

Saída: *Soluções Não-dominadas.*

O SPEA II introduz um método baseado na dominância de Pareto que não requer ordenação por não-dominância, assim como é feito no NSGA II, por exemplo. Cada solução tem uma aptidão proporcional ao número de indivíduos que domina, e também ao número de indivíduos que a dominam. O uso da população externa garante que as soluções ótimas de Pareto sejam conservadas para a geração seguinte. Entretanto, o algoritmo de corte para SPEA II padece do mesmo problema que o algoritmo de corte para NSGA II. Segundo Zitzler et al. (2001), o cálculo da aptidão em SPEA II não garante que as soluções externas que dominam o maior número de indivíduos tenham uma melhor aptidão. Outro fator relevante é a introdução de um parâmetro adicional, o tamanho da população externa E .

A complexidade do SPEA II é governada pela ordenação das distâncias em Q , sendo que sua complexidade total é então $\mathcal{O}(MS^2) \log E$. No caso onde $|E| = N$, a complexidade é $\mathcal{O}(MN^2)$.

f) PMOGA (*Pareto Multi-objective Optimization Genetic Algorithm*)

Castro (2001) desenvolveu o algoritmo evolutivo denominado PMOGA (*Pareto Multi-objective Optimization Genetic Algorithm*). Este algoritmo é baseado nos conceitos de AG clássicos, acoplado a novos operadores multi-objetivo no armazenamento de soluções não-dominadas em um filtro externo. Esses são formados por um operador de exclusão, que descarta as soluções excedentes no filtro; um operador de adição para geração de soluções intermediárias entre as soluções mais distantes; um operador de otimização individual para melhorar o valor ótimo de cada função individualmente e um operador de elitismo a todo conjunto de pontos retidos no filtro.

Segundo esse autor, o operador de otimização individual atua de duas formas diferentes: promovendo a recombinação de duas soluções mínimas de cada função objetivo n vezes, e copiando a solução mínima de cada função objetivo n vezes com mutação. As novas soluções geradas são inseridas no próprio filtro.

A seguir é apresentada a estrutura do PMOGA.

Parâmetros de Entrada: *Tamanho da população, Probabilidade de mutação, Probabilidade de recombinação, Tamanho do filtro e Número máximo de gerações.*

- ❶ Gerar, avaliar e ordenar a população inicial;
 - Cópia dos melhores indivíduos da população para o filtro;
 - Operadores de exclusão no filtro até reduzi-lo ao tamanho desejado;
 - Inserção das soluções do filtro na próxima geração como filhos;
- ❷ Se a geração corrente for menor que o número de gerações faça:
 - Seleção;
 - Mutação;
 - Avaliação;
 - Completar o tamanho da população, se necessário, atualizando e ordenando a população. Cópia dos melhores indivíduos da população para o filtro;
- ❸ Utilizar os operadores:
 - Operador de otimização individual no filtro (recombinação e mutação);
 - Operador de adição no filtro;
 - Ordenamento do filtro;
 - Eliminação das soluções dominadas do filtro;
 - Operador de exclusão no filtro até reduzi-las ao tamanho desejado;
 - Inserção das soluções do filtro na próxima geração como filhos;
- ❹ Avançar até o número máximo de gerações;
- ❺ Fim;

Saída: *Soluções Não-dominadas.*

Castro (2001) aplicou o PMOGA a problemas clássicos da literatura e de Engenharia Civil, obtendo resultados satisfatórios no que tange o número de gerações obtidas pela utilização de outros algoritmos genéticos.

g) MOGA I (*Multi-objective Optimization Genetic Algorithm*)

Recentemente, Silva (2004) desenvolveu o algoritmo MOGA I³ (*Multi-objective Optimization Genetic Algorithm*). O algoritmo desenvolvido apresenta as seguintes características: desenvolvimento de uma metodologia de classificação de pontos, baseada no conceito de dominância, para tratar problemas multifuncionais; introdução de um operador filtro para evitar a perda de pontos não-dominados ao longo do processo evolutivo; a utilização de um operador teste de nicho, para proporcionar a propagação dos melhores indivíduos na etapa de cruzamento, evitando o fenômeno de tendência genética da população; desenvolvimento de um operador de elitismo para a preservação dos melhores pontos de cada função individual; implementação de um novo conceito em função aptidão, que tem incorporada a posição relativa dos pontos no espaço populacional; elaboração de uma estratégia de reinicialização para prevenir a estagnação do algoritmo e melhorar a distribuição dos pontos em problemas com múltiplas funções objetivo; formulação de uma metodologia de penalização baseada na teoria da lógica nebulosa, para tratar problemas com restrições.

A seguir é apresentada a estrutura do Algoritmo MOGA I.

Parâmetros de Entrada: *Tamanho da população, Probabilidade de mutação, Probabilidade de recombinação, Tamanho do filtro e Número máximo de gerações.*

- ❶ Gerar, avaliar a população inicial, aplicando penalidade se necessário;
- ❷ Registrar o melhor ponto de cada função objetivo no arquivo de elitismo;
- ❸ Classificar a população e calcular a função adaptação;
- ❹ Registrar os pontos da classe 1 no filtro de Pareto;
- ❺ Se a geração corrente for menor que o número de gerações faça:
 - Reprodução;
 - Cruzamento usando nicho;
 - Mutação;
 - Substituição de pontos por elitismo;
 - Aplicar penalidade;
 - Registrar o melhor ponto de cada função objetivo no arquivo de elitismo;
 - Classificar a população e calcular a função de adaptação;

³Originalmente o algoritmo desenvolvido por Silva (2004) não foi denominado MOGA I, sendo este nome utilizado aqui apenas para distingui-lo dos outros.

- Registrar os pontos da classe 1 no filtro de Pareto;
 - Guardar as soluções em um filtro;
 - Descartar pontos dominados;
- ⑥ Se o filtro estiver cheio, descartar pontos similares e arquivar os pontos não-dominados;
 - ⑦ Avançar até o número máximo de gerações;
 - ⑧ Fim

Saída: *Soluções Não-dominadas.*

Silva (2004) aplicou o MOGA em uma série de problemas matemáticos e em processos de Engenharia Química, dentre eles, processos que são focos de estudo de projetos de pesquisa desenvolvidos pelo grupo de Modelagem, Simulação e Controle de Processos do PEQ/COPPE/UFRJ. As fronteiras de Pareto foram geradas, em todos os casos, com um desempenho bastante satisfatório, além de uma boa distribuição de pontos.

5.3 Recozimento Simulado

O Recozimento Simulado (*Simulated Annealing*) baseia-se em uma analogia com a metalurgia, mais especificamente no recozimento, onde os metais são levados a temperaturas elevadas e, posteriormente, são gradualmente resfriados até alcançar um estado mínimo de energia. Sob outras condições, menos cuidadosas de resfriamento, o material se cristalizaria com uma energia “localmente mínima”, o que freqüentemente se traduz em imperfeições estruturais (KIRKPATRICK et al., 1983).

Analogia entre o Processo de Otimização com o de Recozimento da Metalurgia:

Sabe-se da Metalurgia que, se um metal é resfriado em condições apropriadas, uma estrutura cristalina estável pode ser obtida (KIRKPATRICK et al., 1983). No recozimento o metal é aquecido a altas temperaturas, causando um choque violento entre os átomos. Se o metal for resfriado de forma brusca, a microestrutura tende a um estado randomicamente instável. Porém, se o metal é resfriado de forma suficientemente lenta, o sistema procurará um ponto de equilíbrio caracterizado por uma microestrutura ordenada e estável. Do ponto de vista matemático, as variáveis de projeto são perturbadas randomicamente, armazenando-se o melhor valor da função objetivo a cada perturbação. A temperatura é então reduzida e novas tentativas executadas. Tal procedimento continua até escapar-se

de um mínimo local, sendo que, ao final do processo, é possível que se obtenha um mínimo global (METROPOLIS et al., 1953).

Metropolis et al. (1953) introduziram um método numérico simples que representa o estado de equilíbrio de um conjunto de átomos a uma dada temperatura. A analogia com o processo de otimização pode ser feita analisando a Fig. 5.10:

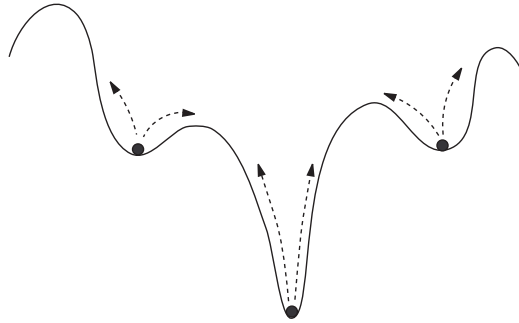


Figura 5.10: Analogia entre o processo de otimização e o recozimento simulado (Reproduzido de Metropolis et al. (1953)).

Seja ΔE a variação da energia de um sistema de átomos a uma temperatura T . Em cada passo do algoritmo é dado um deslocamento aleatório a um átomo, o que implica uma nova variação da energia do sistema ΔE . Se esta nova variação da energia ΔE é menor ou igual a zero ($\Delta E \leq 0$), o deslocamento é aceito, caso contrário ($\Delta E > 0$), a probabilidade da configuração ser aceita será dada pela equação:

$$P(\Delta E) = \exp \left(\frac{-\Delta E}{K_b T} \right) \quad (5.15)$$

onde, K_b é a constante de Boltzmann.

Um número randômico r , uniformemente distribuído, deve ser gerado no intervalo (0, 1). Se $r \leq P(\Delta E)$ a nova configuração é aceita. Se $r > P(\Delta E)$ a configuração anterior é utilizada para iniciar um novo passo. A escolha da função de probabilidade $P(\Delta E)$, conforme acima descrito, se deve ao fato de que o sistema evolui segundo uma distribuição de Boltzman.

Corana et al. (1987) ressalta que se T tiver magnitude muito superior ao desvio padrão da função no intervalo, quase todos os pontos são aceitos. Já se T for igual a zero, o método se torna uma busca aleatória do mínimo. Assim, adota-se T_i como o valor do desvio padrão da função objetivo no intervalo estudado e T_f como a ordem de grandeza desejada para o precisão do ponto ótimo.

A extensão dessa abordagem a POMO foi inicialmente proposta por Serafini (1992). Segundo o autor, os dois principais obstáculos a essa extensão são a ponderação entre a viabilidade *versus* eficiência e como definir uma regra de aceitação que determine a probabilidade de uma dada configuração ser a vencedora. Dentre todas as regras sugeridas, as quatro mais freqüentemente empregadas são:

- *Regra de Chebyshev*:

$$p_i \leftarrow \min \left(1; \exp \left(\max_{k=1 \dots M} \left(- \frac{f_k(x) - f_k(x_i)}{T_i} \right) \right) \right) \quad (5.16)$$

- *Regra Linear-Escalar*:

$$p_i \leftarrow \min \left(1; \exp \left(\sum_{k=1}^M \left(- \frac{f_k(x) - f_k(x_i)}{T_i} \right) \right) \right) \quad (5.17)$$

- *Regra Weak*:

$$p_i \leftarrow \min \left(1; \exp \left(\min_{k=1 \dots M} \left(- \frac{f_k(x) - f_k(x_i)}{T_i} \right) \right) \right) \quad (5.18)$$

- *Regra do Produto*:

$$p_i \leftarrow \prod_{k=1}^M \min \left(1; \exp \left(- \frac{f_k(x) - f_k(x_i)}{T_i} \right) \right) \quad (5.19)$$

onde T_i é a temperatura corrente na iteração i e M é o número de objetivos.

Segundo Ehrgott e Gandibleux (2004), os algoritmos baseados em Recozimento Simulado se distinguem por três pontos principais: o papel da aceitação de uma nova solução com relação à probabilidade dependente da temperatura; o esquema de decréscimo da temperatura e o mecanismo que leva à aproximação da curva de Pareto.

Ulungu e Teghem (1992) desenvolveram um dos mais populares algoritmos baseados em Recozimento Simulado, o MOSA (*Multiobjective Simulated Annealing*). Este algoritmo consiste na obtenção de vizinhos (soluções em potencial) através do uso do Método da Soma Ponderada para definir direções de busca. A comparação entre x e a solução em potencial x_i ($\Delta f_k = f_k(x) - f_k(x_i)$), de acordo com M objetivos, se dá por meio de três possíveis casos:

- $\forall k \Delta f_k \leq 0$: todos os objetivos são melhorados em relação a x ;

- $\forall k' \Delta f_k < 0$ e $\Delta f_{k'} > 0$: melhoria e deterioração ocorrem simultaneamente. Ambas as soluções x e x_i são potencialmente eficientes;
- $\forall k \Delta f_k \geq 0$: todos os objetivos estão deteriorados. A solução x é dominada pela solução x_i ;

Quando uma vizinhança é aceita, o conjunto de soluções potenciais dadas naquela direção é atualizado. A busca termina após um certo número de iterações, ou quando uma temperatura pré-definida é encontrada.

5.4 Busca Tabu

A técnica meta-heurística Busca Tabu (*Search Tabu*) foi desenvolvida por Glover (1977). Essa abordagem caracteriza-se pelo uso de uma memória flexível, baseada na avaliação de uma função que escolhe a solução, com melhor resultado a cada iteração, selecionando o movimento que produz melhoria ou, ao menos, uma perturbação na função objetivo. Uma lista tabu é empregada para armazenar as características dos movimentos aceitos de forma que este conjunto de características tenha sido usado para classificar certos movimentos como tabu (a ser evitado) nas próximas repetições. Em outras palavras, a lista tabu determina quais soluções podem ser alcançadas por um movimento a partir da solução atual. Além disso, considerando que são aceitos movimentos que não conduzem a melhorias em busca tabu, é possível voltar a soluções já visitadas, o que poderia causar a reavaliação de um mesmo projeto e consequentemente o aumento do custo computacional (PHAM; KARABOGA, 2000).

A primeira extensão dessa metodologia no contexto multi-objetivo (*Multi-objective Tabu Search*) foi proposta por Hertz et al. (1994). Esta abordagem baseia-se em resolver uma sequência de objetivos simples e sub-problemas com multi-restrições. Nesses sub-problemas, cada objetivo é considerado individualmente e otimizado de acordo com sua importância relativa. O primeiro algoritmo projetado para a obtenção de soluções potencialmente eficientes foi desenvolvido por Gandibleux et al. (1997). Neste algoritmo usa-se a escalarização das funções e o método executa uma série de processos tabus guiados automaticamente no espaço de objetivos pela aproximação corrente da curva de Pareto. Duas memórias tabus são usadas, uma no espaço de decisão das variáveis e outra no espaço dos objetivos. O algoritmo desenvolvido por Hansen (1997) consiste na busca do conjunto de soluções não-dominadas usando o Método da Soma Ponderada para obter candidatos em potencial nas vizinhanças da população corrente. Nessa abordagem, os pesos são dinamicamente atualizados para a exploração de regiões que se tornem atrativas. Em outra abordagem, desenvolvida por Baykasoglu et al. (1999), o melhor candidato é randomica-

mente selecionado da solução atual, a qual considera somente soluções não-dominadas. Entretanto, esta metodologia não apresenta nenhum mecanismo de diversidade de busca. Similarmente, Alves e Climaco (2000) propuseram um método multi-objetivo iterativo que faz uso de um acoplamento entre Busca Tabu e Recozimento Simulado. Em cada iteração, o usuário (tomador de decisões) deve primeiro especificar as regiões de maior interesse, escolhendo entre aplicar nessa iteração o algoritmo da Busca Tabu ou o do Recozimento Simulado.

5.5 Colônia de Formigas

O Algoritmo da Colônia de Formigas (*Ant Colony Algorithm*) é inspirado pelo comportamento de colônias de formigas reais, em particular, pelo seu comportamento na procura de alimento. Basicamente, a idéia geral é: primeiro, quando as formigas chegam a um ponto em que têm que decidir mover-se numa dada direção elas selecionam aleatoriamente o caminho e depositam feromônio no solo, sem ter ainda noção de qual é a melhor escolha. Depois de um período de tempo, a diferença entre a quantidade de feromônio entre os caminhos é suficientemente grande para influenciar a decisão de novas formigas que estão igualmente no impasse de tomada de decisão por qual caminho seguir. Neste caso, as novas formigas escolhem o caminho com maior quantidade de feromônio. Conseqüentemente, as formigas podem detectar o feromônio e escolher, com dada probabilidade, os caminhos marcados com concentrações mais acentuadas (COELHO, 2003).

Uma das idéias centrais dessa metodologia foi originalmente proposta por Dorigo et al. (1991), cuja fundamentação é baseada na comunicação indireta entre uma colônia de agentes ou formigas, baseada em trilhas de feromônios. As trilhas de feromônios são modificadas pelas formigas, o que reflete, segundo o autor, sua experiência quanto à resolução de um problema em particular.

O algoritmo da colônia de formigas é uma meta heurística baseada em uma população de agentes (formigas) que pode ser utilizada para resolução de problemas de otimização combinatória. Essa abordagem é um algoritmo paralelo e adaptativo Não Determinístico, pois uma população de agentes move-se simultaneamente, de forma independente. Além disso, cada agente escolhe um caminho com base na informação depositada por outros agentes que tenham selecionado previamente o mesmo caminho. Este comportamento cooperativo tem ingredientes de autocatálise (catálise provocada por uma substância, o feromônio, que se forma no próprio sistema reacional), isto é, o método providencia uma realimentação positiva, desde que a probabilidade de um agente escolher o caminho aumenta com o número de agentes que escolheu previamente aquele caminho (COELHO, 2003).

Pode-se traduzir matematicamente a idéia geral do algoritmo da colônia de formigas. Para isso, seja uma população que tem m formigas, onde k é uma formiga genérica ($k = 1, \dots, m$). A probabilidade da k -ésima formiga indicar o caminho (i, j) é dada por:

$$p_{ij}^k = \frac{|\tau_{ij}(t)|^\alpha |n_{ij}(t)|^\beta}{\sum_{k \in k_{\text{permitidos}}} (\tau_{ik}(t))^\alpha (n_{ik}(t))^\beta} \quad (5.20)$$

onde α é a ponderação do feromônio ($0 \leq \alpha \leq 1$) e β é a ponderação da informação heurística ($0 \leq \beta \leq 1$), geralmente ajustados heurísticamente (COELHO, 2003), $n_{ij} = 1/d_{ij}$ é a visibilidade entre a variável j até a variável i e vice-versa, d_{ij} é a distância Euclidiana entre i e j e τ_{ij} é a intensidade da trilha do caminho (i, j) no tempo t (em $t = 0$, a intensidade da trilha é gerada aleatoriamente com distribuição uniforme).

Ao longo da trilha de i até j , a formiga deposita o feromônio na trilha definida por:

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k} & \text{se a formiga } k \text{ usa a trilha } (i, j) \text{ no seu trajeto} \\ 0 & \text{nos outros casos} \end{cases} \quad (5.21)$$

onde Q é uma constante de projeto e L_k é o comprimento do trajeto da k -ésima formiga. Este valor, avaliado quando a formiga completa um trajeto no tempo $[t_o, t_o + n]$ consiste de um ciclo de n iterações, é então utilizado para atualizar a quantidade de substância depositada previamente na trilha, com base em:

$$\tau_{ij}^k(t + n) = \rho\tau_{ij}^k(t) + \Delta\tau_{ij}^k \quad (5.22)$$

onde ρ é um coeficiente que representa a persistência da trilha durante o ciclo (entre o tempo t e $t+n$), usualmente definido heurísticamente. Por outro lado, o valor de $(1-\rho)$ representa a evaporação da trilha entre o tempo t e $t+n$, onde

$$\Delta\tau_{ij}^k = \sum_{k=1}^m \tau_{ij}^k \quad (5.23)$$

Os primeiros algoritmos multi-objetivos baseados em Colônia de Formigas faziam uso do acoplamento com o Método da Soma Ponderada (DOERNER et al., 2001), com otimização lexocográfica (GRAVEL et al., 2002) e com uso do conceito de otimalidade de Pareto (IREDI et al., 2001; DOERNER et al., 2004). Alguns desses algoritmos fazem uso de informações heurísticas e do feromônio para o cálculo da probabilidade para a construção do caminho no próximo passo. Mariano e Morales (1999) utilizaram uma colônia para cada função objetivo, com todas tendo o mesmo número de formigas. Soluções (parciais) de cada colônia são usadas na formação da próxima colônia. O algoritmo foi aplicado ao projeto de distribuição global de água para irrigação através da minimização do custo

global e da maximização do lucro, e comparado ao VEGA (SCHAFFER, 1984). Uma combinação entre o algoritmo da Colônia de Formigas com o SPEA (ZITZLER; THIELE, 1998) foi proposta por Shelokar et al. (2000).

5.6 Sistema Imunológico Artificial

A imunologia é uma ciência relativamente recente. Sua origem é atribuída à Edward Jenner, que descobriu, há aproximadamente 210 anos, em 1796, que a “vacínia” (ou *cow-pox*), induzia proteção contra a varíola, uma doença freqüentemente fatal. Jenner batizou seu processo de vacinação, uma expressão ainda utilizada para descrever a inoculação de indivíduos sãos, com amostras atenuadas ou mortas de agentes causadores de doenças, objetivando a proteção futura contra a enfermidade (SILVA, 2001).

A principal característica de um sistema imunológico é reconhecer todas as células (ou moléculas) no corpo e categorizá-las como sendo próprias ou externas. As células externas são, por sua vez, identificadas de forma a estimular um ou mais sistemas defensivos. O sistema imunológico aprende, através da evolução, a distinguir entre antígenos (bactéria, vírus, etc) e anticorpos. Na Figura 5.11 é apresentada a estrutura multicamadas do sistema imunológico (SILVA, 2001).

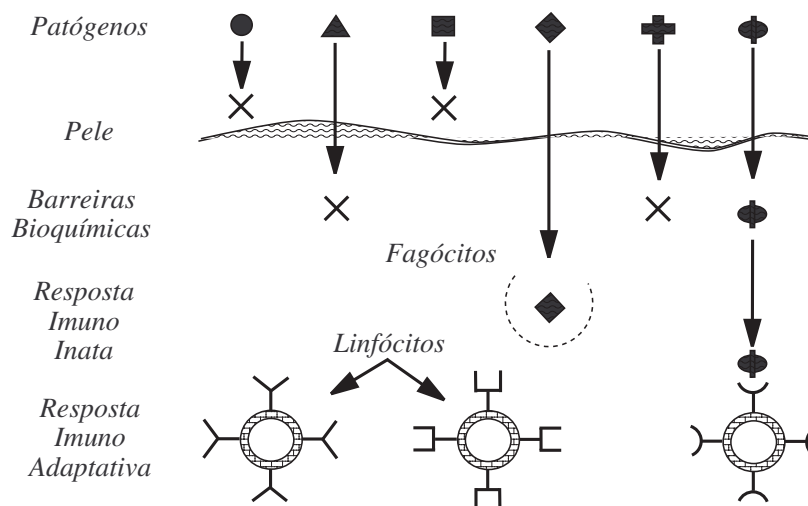


Figura 5.11: Estrutura multicamadas para o sistema imunológico (Reproduzido de Silva (2001)).

O interesse acadêmico pelo sistema imunológico se deve ao seu poder de processar informações, executando processos complexos de forma paralela, distribuída, além de possuir a propriedade de interagir localmente. O sistema imunológico age como um “segundo cérebro”, pois pode armazenar as experiências passadas, fortalecer as interações de suas

células componentes e gerar uma nova resposta para novos padrões de antígenos (CARVALHO; FREITAS, 2002).

O sistema imunológico artificial é constituído de dois operadores: hipermutação e expansão clonal. O conceito de hipermutação consiste no fato de que o mesmo anticorpo pode gerar n clones, e que cada gene do clone pode ser mutado. Na próxima geração, cada um destes n clones poderá gerar outros n clones, e assim sucessivamente. Isto significa que cada gene de um anticorpo pode sofrer mutação diversas vezes entre a primeira geração e a última. Um anticorpo, gerado a partir da clonagem, só é adicionado à população de anticorpos se, e somente se, o valor da sua respectiva função de avaliação for maior ou igual a um limiar de expansão clonal (CARVALHO; FREITAS, 2002).

O mecanismo de expansão clonal pode prover a regulação do processo de hipermutação, a qual é dependente da afinidade do anticorpo. Um anticorpo com baixo valor de avaliação deve ser eliminado. Por outro lado, para anticorpos com alto valor de avaliação, o processo de seleção clonal é iniciado. Alguns detalhes do esquema do princípio da seleção clonal são apresentados na Fig. 5.12.

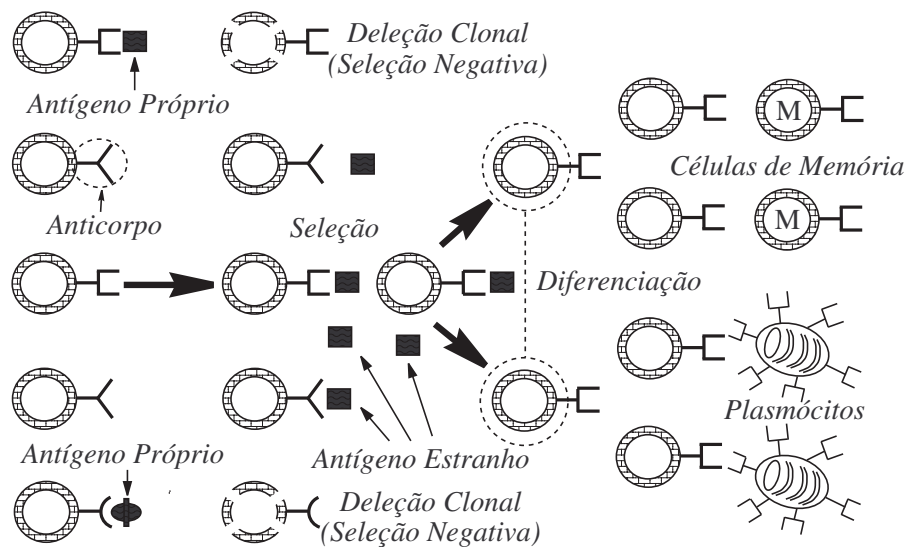


Figura 5.12: Estrutura do princípio da seleção clonal (Reproduzido de Silva (2001)).

No algoritmo imunológico, citado em Carvalho e Freitas (2002), um anticorpo só é clonado se este tem seu valor de avaliação maior que um limiar pré-determinado. A cada geração do algoritmo de seleção clonal, se um anticorpo obtém o limiar da expansão clonal, o sistema gera clones deste anticorpo. Estes clones são submetidos ao processo de hipermutação. Neste caso, tanto o limiar quanto o número de clones são estabelecidos previamente.

Existem várias experiências de desenvolvimento de metodologias, baseadas no sistema imune, para o tratamento de problemas de interesse comercial ou industrial. Algumas organizações financeiras estão utilizando técnicas de imunização para prevenir contra fraudes em processos de hipotecas e empréstimos (HUNT et al., 1996).

Da mesma forma que o sistema imune pode detectar uma infecção viral no organismo, é possível desenvolver uma aplicação capaz de detectar a presença de vírus em computadores. Os vírus, em geral, infectam programas, setores de *boot* e arquivos em geral. A partir desse ponto de vista, o problema de proteção é semelhante ao apresentado nos organismos, detecção *self-nonself*, onde *self* são dados não corrompidos e *nonself* é interpretado como alguma alteração do *self* (SOMAYAJI et al., 1997).

Um trabalho pioneiro utilizando Sistema Imunológico como base da construção de um algoritmo multi-objetivo foi publicado em 1999 (YOO; HAJELA, 1999), quando os autores fizeram uso do Método da Soma Ponderada para a agregação dos objetivos. Contudo, mais recentemente, outras propostas (das quais somente poucas fazem uso do conceito de otimalidade de Pareto) têm sido desenvolvidas (KURPATI; AZARM, 2000; ANHOR et al., 2002; CORTES; COELLO, 2003).

5.7 Enxame de Partículas

A Otimização por Colônia de Partículas (OCP) (*Particle Swarm Optimization*) é baseada em uma população de indivíduos e motivada pela simulação do comportamento social de enxames de pássaros, peixes ou de insetos. Nessa abordagem cada solução candidata, denominada partícula, possui associada a ela uma velocidade, que é ajustada através de uma equação de atualização que considera a experiência da partícula correspondente e a experiência das outras partículas da população.

Seu embasamento teórico foi desenvolvido originalmente por Eberhart e Kennedy (1995) e Kennedy e Eberhart (1995) em populações de soluções, de forma similar a outros algoritmos evolutivos. Entretanto, aqui, a cada solução potencial (indivíduo), é também atribuída uma velocidade aleatória. As soluções potenciais, denominadas partículas, são então “movimentadas” pelo espaço de busca do problema.

O procedimento para implementação do algoritmo é regido pelas seguintes etapas (BONISSONE et al., 1999):

- Iniciar uma população (matriz) de partículas, com posições e velocidades definidas em um espaço n dimensional, de forma aleatória, com distribuição uniforme;

- Para cada uma das partículas, avaliar a função de adaptação;
- Comparar o valor da função de adaptação da partícula com o *pbest* da partícula. Se o valor corrente é melhor que *pbest*, então o valor de *pbest* passa a ser igual ao valor da função de adaptação da partícula, e a localização do *pbest* passa a ser igual à localização atual no espaço n dimensional;
- Comparar a avaliação da função de adaptação com o melhor valor prévio de adaptação da população. Se o valor atual é melhor que *gbest*, atualizar o valor de *gbest* para o índice e valor da partícula atual;
- Modificar a velocidade e a posição da partícula de acordo com Eq. (5.24) e Eq. (5.25).

$$v'_i(t) = wv_i(t) + c_1ud(p_i(t) - x_i(t)) + c_2Ud(p_g(t) - v_i(t)) \quad (5.24)$$

$$x'_i(t) = x_i(t) + v'_i(t) \quad (5.25)$$

onde $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$ armazena a posição da i -ésima partícula. O valor de $p_i = [p_{i1}, p_{i2}, \dots, p_{in}]^T$ representa a posição do melhor valor de adaptação da i -ésima partícula. O índice g representa o índice de melhor partícula entre todas as partículas do grupo. A variável w é a ponderação de inércia, c_1 e c_2 são constantes positivas, ud e Ud são duas funções para a geração de números aleatórios com distribuição uniforme no intervalo $(0,1)$.

- Ir para a segunda etapa até que um critério de parada seja satisfeito, usualmente uma função de adaptação considerada como suficientemente boa ou um número máximo de iterações (gerações).

As velocidades das partículas em cada dimensão são limitadas a um valor máximo de velocidade, V_{max} , pois este determina a resolução com que a região próxima a soluções atuais é explorada. Se V_{max} é muito alto, facilita-se a busca global, enquanto para um valor de V_{max} pequeno enfatiza-se as buscas locais.

A primeira parte da Eq. (5.24) é um termo de momento da partícula. A ponderação de inércia w representa o grau de momento da partícula. A segunda parte consiste da parte “cognitiva”, que representa o “conhecimento” da partícula, independentemente. A terceira é a parte “social”, que representa a colaboração entre as partículas.

As constantes c_1 e c_2 representam a ponderação das partes de “cognição” e “social” que influenciam cada partícula em direção a *pbest* e *gbest*. Estes parâmetros são usualmente ajustados por heurísticas de tentativa e erro (COELHO, 2003).

Com o intuito de trabalhar com problemas multi-objetivo, Parsopoulos e Vrahatis (2002) propuseram o uso do método de OCP em combinação com o Método da Soma Ponderada. Além disso, é apresentada uma versão modificada do método OCP, empregando os princípios do algoritmo VEGA (SCHAFER, 1984). Foram testados dois métodos de agregação por pesos:

- CWA (*Conventional Weighted Aggregation*): utiliza pesos fixos e origina apenas um ponto por execução do algoritmo de otimização. O processo de busca deve ser repetido várias vezes para a obtenção do conjunto de Pareto. Este método requer um conhecimento prévio do espaço de busca, para a determinação apropriada dos pesos;
- DWA (*Dynamic Weighted Aggregation*): modifica gradualmente os pesos durante a otimização, de acordo com a relação

$$w_1(t) = \left| \sin \left(\frac{2\pi t}{F} \right) \right| \quad (5.26)$$

onde t é a iteração e F é a frequência de modificação dos pesos e $w_2(t) = 1 - w_1(t)$.

Ainda segundo Parsopoulos e Vrahatis (2002), a mudança lenta dos pesos força o algoritmo a mover-se sobre a frente de Pareto. No caso de conjuntos de Pareto convexos, este método apresenta melhor desempenho. A versão modificada do método, baseada no algoritmo VEGA (SCHAFER, 1984), utiliza dois algoritmos, cada qual trabalhando em prol de uma função objetivo, porém, compartilhando informações entre si. Assim, a melhor partícula do primeiro algoritmo é usada para determinar as novas velocidades das partículas do segundo algoritmo.

5.8 Evolução Diferencial

A Evolução Diferencial - ED (*Differential Evolution*) é um algoritmo evolutivo proposto por Storn e Price (1995) para problemas de otimização. Uma das principais características dessa metodologia é que esta apresenta uma concepção puramente matemática, baseada em operações vetoriais, sendo por este motivo considerada uma abordagem estrutural (COELHO, 2003). A Figura 5.13 apresenta a fundamentação teórica do algoritmo de Evolução Diferencial.

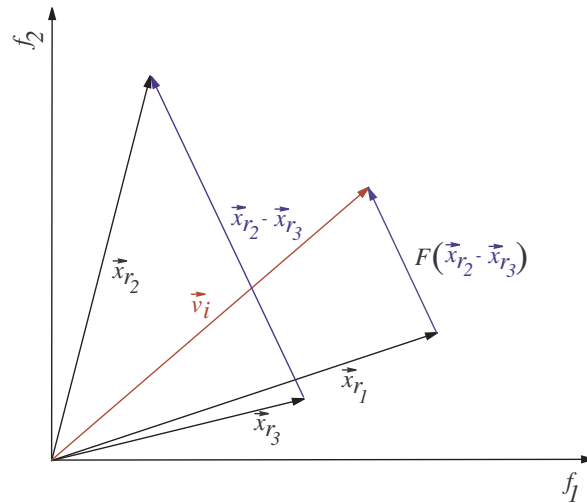


Figura 5.13: Fundamentação teórica do algoritmo de ED (Reproduzido de Price et al. (2005)).

A figura acima nos mostra que, a partir de três vetores $\vec{x}_{r_1}$, $\vec{x}_{r_2}$ e $\vec{x}_{r_3}$, são escolhidos aleatoriamente dois deles (nesse caso, $\vec{x}_{r_2}$ e $\vec{x}_{r_3}$), sendo realizada a subtração dos mesmos. O resultado é multiplicado por um escalar F , gerando assim um vetor com módulo diferente da subtração original. Esse novo vetor é então somado ao vetor $\vec{x}_{r_1}$, fornecendo assim um novo vetor $\vec{v}_i$. Esse vetor $\vec{v}_i$ indicará uma nova posição no espaço, isto é, em termos do algoritmo de ED, um novo indivíduo é gerado.

No algoritmo da ED, o valor de cada variável é representado por um valor real (ponto flutuante) e o procedimento de otimização é regido pelas seguintes etapas:

- gera-se uma população inicial com soluções factíveis para o problema em questão, onde garante-se por “*regras de reparo*” que os valores atribuídos às variáveis estão dentro das fronteiras delimitadas pelo projetista;
- seleciona-se um indivíduo, de forma aleatória, para ser substituído. Três diferentes indivíduos são selecionados como genitores (pais), sendo que um destes é selecionado como genitor principal;
- modifica-se cada variável do genitor principal, conforme uma certa probabilidade;
- adiciona-se ao valor atual da variável (genitor principal) a diferença entre duas outras variáveis (genitores secundários) ponderada por uma taxa de perturbação F . Este procedimento representa o operador de cruzamento na ED;
- se o vetor resultante apresenta uma função de adaptação melhor que o genitor principal, ele o substitui; caso contrário, esse vetor é mantido na população.

Em outras palavras, adotando-se um formalismo matemático, na ED uma solução l , na geração w , é um vetor multidimensional $x_{G=w}^l = (x_1^l, x_2^l, \dots, x_N^l)^T$. Uma população $P_{G=k}$ na geração $G = k$ é um vetor de M soluções, onde $M > 4$. A população inicial $P_{G=0} = (x_{G=0}^l, x_{G=0}^l, \dots, x_{G=0}^l)^T$ é gerada inicialmente com distribuição uniforme, adotando-se:

$$x_{G=0}^l = l_{inferior}(x_i) + rand_i(0, 1)(l_{superior}(x_i) - l_{inferior}(x_i)) \quad (5.27)$$

onde $l_{inferior}(x_i)$ e $l_{superior}(x_i)$ são os limites inferior e superior de valores admissíveis para a variável x_i , respectivamente; M é o tamanho da população; N é a dimensão da solução e $rand_i(0, 1)$ gera um número aleatório no intervalo entre 0 e 1. A seleção é realizada para escolher quatro índices diferentes r_1, r_2, r_3 e $j \in (1, M)$. Os valores de cada variável, na solução descendente (filha), são modificados com uma mesma probabilidade de cruzamento, CR , para $\forall i \leq N$:

$$x_{G=k}^l = \begin{cases} x_{i,G=k-1}^{r_3} + F(x_{i,G=k-1}^{r_1} - x_{i,G=k-1}^{r_2}) & \text{se } rand(0, 1) < CR \\ x_{i,G=k-1}^j & \text{nos outros casos} \end{cases} \quad (5.28)$$

A nova solução substitui a solução anterior (antiga) se ela é melhor e pelo menos uma das variáveis é modificada. Esta solução é representada na ED pela seleção aleatória de uma variável, $i_{rand} \in (1, N)$. Depois da operação de cruzamento, se uma ou mais variáveis na nova solução estão fora da fronteira (limites), pode-se fazer uso de uma regra de “reparo” para corrigir os valores dessas variáveis da seguinte maneira:

$$x_{G=k}^l = \begin{cases} (x_{i,G}^j + l_{inferior}(x_i))/2 & \text{se } x_{i,G+1}^j < l_{inferior}(x_i) \\ (x_{i,G}^j - l_{superior}(x_i))/2 & \text{se } x_{i,G+1}^j > l_{superior}(x_i) \\ x_{i,G+1}^j & \text{nos outros casos} \end{cases} \quad (5.29)$$

Storn e Price (1995) aconselham o uso dos seguintes valores para os parâmetros do algoritmo de ED: número de indivíduos da população como sendo um valor entre 5 e 10 vezes o número de variáveis de projeto, taxa de perturbação F entre 0,2 e 2,0 e probabilidade de cruzamento CR entre 0,1 e 1,0.

A Equação (5.28) apresenta, de forma geral, uma das formas pela qual o candidato pode ser gerado no algoritmo de ED. A Tabela 5.4 apresenta as estratégias para a geração de candidatos em potencial propostas por Storn e Price (1995).

A convenção utilizada por Storn e Price (1995) é DE/x/y/z, onde x representa o vetor que será perturbado. Por exemplo, optando-se por RAND, o vetor que será perturbado

Tabela 5.4: Representação das estratégias do algoritmo de ED.

Estratégia	Representação	Mecanismo de Mutação
1	$x^{j+1} = x_{best}^j + F(x_\beta^j - x_\gamma^j)$	DE/BEST/1/EXP
2	$x^{j+1} = x_\alpha^j + F(x_\beta^j - x_\gamma^j)$	DE/RAND/1/EXP
3	$x^{j+1} = x_{old}^j + F(x_{best}^j - x_{old}^j) + F(x_\gamma^j - x_\delta^j)$	DE/RAN-TO-BEST/2/EXP
4	$x^{j+1} = x_{best}^j + F(x_\alpha^j - x_\beta^j) + F(x_\gamma^j - x_\delta^j)$	DE/BEST/2/EXP
5	$x^{j+1} = x_\alpha^j + F(x_\rho^j - x_\beta^j) + F(x_\gamma^j - x_\delta^j)$	DE/RAND/2/EXP
6	$x^{j+1} = x_{best}^j + F(x_\beta^j - x_\gamma^j)$	DE/BEST/1/BIN
7	$x^{j+1} = x_\alpha^j + F(x_\beta^j - x_\gamma^j)$	DE/RAND/1/BIN
8	$x^{j+1} = x_{old}^j + F(x_{best}^j - x_{old}^j) + F(x_\gamma^j - x_\delta^j)$	DE/RAND-TO-BEST/2/BIN
9	$x^{j+1} = x_{best}^j + F(x_\alpha^j - x_\beta^j) + F(x_\gamma^j - x_\delta^j)$	DE/BEST/2/BIN
10	$x^{j+1} = x_\alpha^j + F(x_\rho^j - x_\beta^j) + F(x_\gamma^j - x_\delta^j)$	DE/RAND/2/BIN

é escolhido aleatoriamente na população. Y é o número de pares de vetores que são considerados durante a perturbação e Z é o tipo de cruzamento usado para a geração do candidato, BIN no caso binomial e EXP no caso exponencial. Os subscritos $\alpha, \beta, \gamma, \rho$ e δ são índices escolhidos aleatoriamente na população. x_{best} é o melhor indivíduo da população na geração anterior e x_{old} é um indivíduo, escolhido aleatoriamente da população, na geração anterior.

Recentemente, novas estratégias para a representação do mecanismo de mutação em ED têm sido propostos, dentre os quais, pode-se citar o trabalho de Fan e Lampinen (2003). Em tal abordagem, fatores de ponderação para cada um dos vetores que serão utilizados para a geração do novo indivíduo são calculados conforme mostrado a seguir:

$$x^j = x_{avg}^j + F_1(x_\alpha^j - x_\beta^j) + F_2(x_\beta^j - x_\gamma^j) + F_3(x_\gamma^j - x_\alpha^j) \quad (5.30)$$

onde

$$x_{avg}^j = (x_\alpha^j + x_\beta^j + x_\gamma^j)/3 \quad (5.31)$$

$$F_1 = v_\beta - v_\alpha \quad (5.32)$$

$$F_2 = v_\gamma - v_\beta \quad (5.33)$$

$$F_3 = v_\alpha - v_\gamma \quad (5.34)$$

$$v_\alpha = |f(x_\alpha^j)|/P_{aux} \quad (5.35)$$

$$v_\beta = |f(x_\beta^j)|/P_{aux} \quad (5.36)$$

$$v_\gamma = |f(x_\gamma^j)|/P_{aux} \quad (5.37)$$

$$P_{aux} = |f(x_\alpha^j)| + |f(x_\beta^j)| + |f(x_\gamma^j)| \quad (5.38)$$

Dessa maneira, os valores para as taxas de perturbação são atualizados a cada geração. Esse mecanismo foi utilizado com sucesso, em termos de tempo de processamento quando comparando com os mecanismos originais do algoritmo de ED, para o tratamento de problemas de controle ótimo e problemas inversos (ANGIRA; SANTOSH, 2007).

Nos últimos anos, devido a sua simplicidade conceitual e implementacional, a técnica de ED tem atraído atenção para o desenvolvimento de metodologias de otimização multi-objetivo baseados no algoritmo de Storn e Price (1995). Os principais trabalhos são apresentados a seguir, enfatizando-se suas principais características.

Abbass et al. (2001) desenvolveram o PDE (*Pareto-frontier Differential Evolution*), baseado no conceito de dominância.

A estrutura do PDE é apresentada a seguir.

Parâmetros de Entrada: *Tamanho da população*, *Probabilidade de cruzamento (CR)*, *Número máximo de soluções não-dominadas* e *Número máximo de gerações*. A taxa de perturbação F é gerada a partir de uma Distribuição Gaussiana.

- ❶ Gerar e avaliar a população inicial através de distribuição Gaussiana. Se houver violação de qualquer um dos indivíduos, corrigir esse valor;

Enquanto o número de gerações não for alcançado:

- ❷ Remova todas as soluções dominadas da geração atual. Se o número de soluções não-dominadas for maior que α , remova as soluções excedentes;
- ❸ Para $j=1$ até o número máximo de soluções não-dominadas selecione randomicamente 3 genitores que pertencem a essa população não-dominada;
- ❹ Selecione randomicamente uma posição dessa população ($i_{rand} \in (1, \dots, \alpha)$). Se $rand < CR$ ou $i=i_{rand}$, um candidato é criado utilizando esses genitores. Caso contrário o indivíduo da população é mantido;
- ❺ Se o candidato criado é melhor que o genitor principal, o candidato substitui o genitor na população, caso contrário ele é descartado. Contudo, se o candidato é melhor que o genitor em pelo menos um objetivo, ele é adicionado à população. Incremente j .

Saída: *Soluções Não-dominadas*.

A eliminação das soluções excedentes é feita através da seguinte função de distância entre vizinhos próximos:

$$D(x) = \frac{(\min \|x - x_i\| - \min \|x - x_j\|)}{2} \quad (5.39)$$

onde $x \neq x_i \neq x_j$.

Abbass et al. (2001) testaram esse algoritmo em dois problemas clássicos da literatura, obtendo resultados superiores aos obtidos pelo algoritmo evolucionário SPEA.

Angira e Babu (2005) desenvolveram o NSDE (*Non-dominated Sorting Differential Evolution*), que é uma extensão do algoritmo genético NSGA II (DEB et al., 2000). O algoritmo foi testado em dois problemas clássicos, a função de *Schaffer* e ao problema do projeto de uma viga. Em ambos os problemas foram feitas análises de sensibilidade com relação ao número máximo de gerações, taxa de cruzamento e taxa de perturbação F . Os resultados obtidos mostraram, para esses dois casos, que o aumento do número máximo de gerações causa um aumento do número de soluções não-dominadas até um certo momento. A partir daí, este passa a não ter acréscimos significativos. Já a qualidade da solução implicou em um alto valor da taxa de cruzamento ($\simeq 1$). Com relação à taxa de perturbação, esta não só afeta o número de soluções não-dominadas, mas também a distribuição das soluções na curva de Pareto. Contudo, nenhum resultado conclusivo pode ser tomado somente com a análise desses dois casos.

O algoritmo abaixo apresenta os passos do algoritmo NSDE.

Parâmetros de Entrada: *Tamanho da população, Probabilidade de cruzamento, Taxa de perturbação (CR) e Número máximo de gerações.*

- ❶ Gerar e avaliar a população inicial randomicamente. Se houver violação de qualquer um dos indivíduos, corrigir esse valor;

Enquanto o número de gerações não for alcançado faça:

- ❷ Para $j=1$ até o tamanho da população: escolher randomicamente 3 indivíduos da população, sendo que 1 deles é escolhido para ser substituído (genitor principal) e os outros dois (genitores secundários) para serem utilizados na construção do ruído;
- ❸ Realizar mutação, seleção e avaliação dos candidatos;
- ❹ Se esse candidato for melhor que o genitor principal, ele o substitui. Se o genitor principal for melhor que o candidato, o candidato é descartado. Se o candidato não é melhor, com relação a todos os objetivos, então ele é adicionado à população. Incremente j ;

- ⑤ Da população atual, remova todas as soluções dominadas. Caso o número de soluções não-dominadas extrapole um valor estipulado pelo usuário, use alguma métrica para remover as excedentes (norma Euclidiana, por exemplo).

Saída: *Soluções Não-dominadas.*

Recentemente, Babu et al. (2005) apresentaram o MODE (*Multi-objective Differential Evolution*), que só difere do NSDE no que diz respeito à quantidade de soluções que são utilizadas para a geração dos candidatos e a escolha de F (constante ou atualizado via distribuição Gaussiana). No MODE são gerados sempre o mesmo número de candidatos (mesmo número de indivíduos da população) e no NSDE são sempre gerados q candidatos a partir dos q elementos da população não-dominada anterior. Estatisticamente, caso se tenha um número menor de candidatos gerados, significa que se pode tender, *a priori*, para um número pequeno de soluções não-dominadas no final dessa geração. Isso faz com que seja necessário um número maior de gerações para obter a mesma qualidade de um outro algoritmo. Além disso, utilizando somente soluções não-dominadas não implica que serão obtidos candidatos não-dominados. Por outro lado, a grande vantagem em se utilizar a estratégia implementada no MODE é que se pode obter uma solução de qualidade com menor tempo computacional. Babu et al. (2005) aplicaram o MODE a um caso industrial, a otimização das condições de operação de um reator adiabático de estireno, tendo como principais objetivos a produtividade, a seletividade e o rendimento. Cinco casos são considerados, dentre eles 4 foram formulados com dois objetivos e 1 formulado com três objetivos. Os resultados obtidos foram melhores que os alcançados pelo NSGA.

De forma geral, a diferença entre os algoritmos propostos até o momento está na estratégia escolhida para formar a curva de Pareto, ou seja, utilizando somente soluções não-dominadas ou todas as soluções durante o processo evolutivo.

O próximo capítulo apresenta os métodos híbridos obtidos através do acoplamento entre técnicas clássicas e evolutivas ou entre técnicas evolutivas. Além disso, o capítulo mostra as principais abordagens utilizadas para o tratamento de restrições de caminho e para a atualização dos parâmetros nos algoritmos evolutivos.

MÉTODOS HÍBRIDOS, MÉTODOS PARA O TRATAMENTO DE RESTRIÇÕES E MÉTODOS PARA A ATUALIZAÇÃO DOS PARÂMETROS EM AE

este capítulo, serão abordadas metodologias propostas na literatura para o acoplamento entre várias técnicas Determinísticas e entre técnicas Determinísticas e Não Determinísticas, com o intuito de aumentar o desempenho dos algoritmos de otimização. Além disso, este capítulo também apresenta as principais abordagens propostas para o tratamento de problemas restritos e para a atualização dos parâmetros em algoritmos evolutivos.

6.1 Tratamento de POMO usando Métodos Híbridos

De uma forma geral, pode-se dizer que o uso de uma dada técnica de otimização pode enfrentar dificuldades de convergência. No caso de técnicas clássicas, onde se exige o cumprimento de requisitos matemáticos estritos (diferenciação, continuidade, unimodalidade), a existência de mínimos locais pode inviabilizar a obtenção do ótimo global. Por outro lado, ao se usar uma técnica evolutiva sem prévio conhecimento da região do ótimo, o procedimento pode implicar um número de iterações por demais elevado, dificultando sua utilização em muitas aplicações (LOBATO; STEFFEN JR, 2006).

A eficiência dos Métodos de Otimização Determinísticos dependem de fatores como: a configuração inicial de projeto, a precisão da avaliação da direção descendente, o método utilizado para executar a busca em linha e o critério de parada (COELHO, 2003; VANDERPLAATS, 1999; EDGAR et al., 2001). A solução obtida é geralmente um ponto de mínimo local, que pode ser o mínimo global, caso a função apresente apenas um ponto de mínimo. Como principais desvantagens pode-se citar a necessidade de avaliação do gradiente e da matriz Hessiana e a falta de garantia de se ter mesmo chegado ao mínimo global (VANDERPLAATS, 1999; EDGAR et al., 2001).

Já os Métodos Não Determinísticos não necessitam do cálculo do gradiente e são aptos a encontrar a solução global. Contudo, exigem um número elevado de avaliações da função objetivo, necessárias para encontrar a solução (COELHO, 2003).

Para superar as dificuldades na utilização das duas abordagens utilizadas individualmente, os Métodos Híbridos, que fazem uso do acoplamento entre a abordagem Determinística e a Não Determinística, vem galgando espaço em aplicações mais realísticas (MICHALEWICZ et al., 1999; LAGARROS et al., 2002; VIANA; STEFFEN JR, 2006; LOBATO; STEFFEN JR, 2006). Segundo Deb (2001), existem pelo menos duas razões para a utilização de métodos híbridos em problemas de otimização multi-objetivo realísticos:

- assegura uma melhor convergência para o ótimo global de Pareto;
- exige menor esforço computacional quando comparado aos métodos considerados individualmente.

Devido aos fatores acima citados, a junção entre abordagens para formar as denominadas metodologias híbridas se configura como uma alternativa promissora para o tratamento de problemas mais próximos à realidade da engenharia, já que esses apresentam alta dimensão e restrições advindas de limitações operacionais e físicas (DULIKRAVICH et al., 1999).

Segundo Coelho (2003), uma estratégia eficiente para o manuseio de abordagens híbridas é executar, inicialmente, um algoritmo evolutivo para localizar a região do ótimo global e, em seguida passa-se para outra técnica de otimização de busca local. A vantagem da utilização de um método de busca local neste caso, está no aumento da velocidade de convergência. Além disso, o valor final obtido pelo método de busca local provavelmente será mais preciso que o obtido pelo algoritmo evolutivo.

A maior dificuldade no uso de metodologias híbridas é encontrar o momento de parar o procedimento de busca estocástico para iniciar o procedimento de busca determinística (VASCONCELOS et al., 1999). A partir disso, os autores propõem os seguintes critérios para a comutação entre algoritmos evolutivos e métodos de busca local:

- *número de gerações*: no caso mais simples, o procedimento de otimização do algoritmo evolutivo é finalizado em um número especificado de gerações e o melhor resultado é transferido para o Método Determinístico. A principal desvantagem deste critério é que nem todas as funções objetivo apresentam o mesmo comportamento;
- *diferença entre os melhores valores da função objetivo no conjunto de gerações*: este procedimento é menos sensível ao tipo de problema que o do primeiro caso;
- *diferença entre os melhores valores da função objetivo em uma mesma geração*: neste caso analisa-se a similaridade dos indivíduos que compõem a população.

Na literatura, apesar de não se encontrar ainda uma classificação rigorosa quanto a nomenclatura, pode-se distinguir duas vertentes no tratamento das metodologias híbridas: as que fazem uso da agregação de técnicas evolutivas com técnicas clássicas e as que fazem uso apenas do acoplamento de técnicas evolutivas.

6.1.1 Algoritmos que Acoplam Técnicas Não Determinísticas

Murata e Ishibuchi (1995) introduziram uma combinação de um algoritmo evolutivo, que é baseado na soma dos objetivos ponderados por pesos, com um algoritmo de busca local que previne a perda de soluções não-dominadas pelo armazenamento destas em um filtro. Depois de atualizar o conjunto de Pareto, são escolhidos pares de indivíduos na fase de seleção, de acordo com pesos randomicamente gerados. Então, uma fração do conjunto de Pareto é injetada dentro da população. Finalmente, uma busca local é efetuada para cada indivíduo com o intuito de melhorar as soluções correntes.

Hu et al. (2006) propuseram um algoritmo evolutivo multi-objetivo de busca local híbrida, denominado NEES (*Neighborhood Exploring Evolution Strategy*). Nessa abordagem, os candidatos são organizados em ordem crescente com relação a uma função objetivo principal (definida pelo usuário). Em seguida, esses candidatos evoluem formando famílias independentemente, sendo que cada família escolhe seu melhor indivíduo como pai para a próxima geração. Essas duas fases são executadas continuamente até que alguma condição de parada seja alcançada. A idéia da exploração das vizinhanças origina-se da aplicação em problemas com um único objetivo. De forma geral, esta se baseia na exploração das vizinhanças de pontos candidatos através da formação de fronteiras de tamanhos diferentes para soluções diferentes. Dentro de cada fronteira, são geradas novas soluções usando uma estratégia de evolução $(1+\lambda)$. Segundo os autores, este esquema equilibra o efeito de busca local naturalmente (que é finalizado pela estratégia de evolução) com o da busca global executada pelo algoritmo, e gradualmente direciona a população para a solução

ótima de Pareto. Três versões da estratégia de evolução foram propostas: (1+1)-NEES, (1+2)-NEES e (1+5)-NEES. Os resultados das simulações indicam que, para problemas de otimização contínuos, a proposta (1+1)-NEES obteve os melhores resultados com relação ao NSGA II. Além disso, todas as versões do algoritmo proposto apresentaram melhores resultados que o NSGA II ao tratar problemas de otimização discreto.

6.1.2 Algoritmos que Acoplam Técnicas Determinísticas e Não Determinísticas

De uma forma geral, permite-se que um determinado algoritmo multi-objetivo evolutivo execute um número fixo de gerações afim de produzir soluções não-dominadas. Após esta etapa, um método de busca local que, na sua concepção mais básica, é fundamentada apenas na geração de vizinhos candidatos a ótimo, é inicializado para cada uma destas soluções independentemente (GOEL; DEB, 2001). O que diferencia esses algoritmos de busca local é a forma com que a geração dos vizinhos é obtida.

Ishibuchi e Yoshida (2003)¹ em seu trabalho, examinaram seis alternativas para especificar a função objetivo usada para a busca local. Foram utilizados o Método da Soma Ponderada com pesos randômicos (DEB, 2001) e o uso direto da função adaptação do algoritmo SPEA II (ZITZLER et al., 2001) para montar essas variações. Os resultados de simulação mostraram que os melhores resultados foram obtidos a partir do uso do Método das Somas Ponderadas consideradas individualmente. Além disso, é mostrado nesse trabalho que o uso direto da função de adaptação do SPEA II como função objetivo na busca local demanda um maior tempo de processamento quando comparado com outras metodologias testadas. Um outro aspecto muito importante observado pelos autores, é que a utilização de hibridação com busca local deteriorou a habilidade de busca do SPEA II. Contudo, segundo os mesmos autores, essas conclusões preliminares não podem ser tomadas como definitivas, pelo simples fato de ter sido analisado apenas um estudo de caso naquele trabalho.

Hu et al. (2003)² incorporaram o algoritmo de busca local SQP aos algoritmos SPEA II (ZITZLER et al., 2001) e ao NSGA II (DEB et al., 2000), através do Método da Negociação (*Trade-Off Method*). Essa metodologia híbrida consiste em gerar uma população inicial e, a partir dessa, utilizar o Método da Negociação para melhorar um dos objetivos (objetivo principal), enquanto os outros são restritos por limites superiores previamente definidos pelo usuário. A partir dessa busca local realizada pelo algoritmo SQP, os pontos encontrados são submetidos a todos os operadores dos dois algoritmos genéticos, gerando

¹<http://www.lania.mx/~ccoello/EMOO>, acessado em 10/06/2006.

²<http://www.lania.mx/~ccoello/EMOO>, acessado em 15/07/2006.

assim uma nova população que será submetida novamente à busca local. Os autores obtiveram resultados promissores a partir da aplicação dessa abordagem a problemas clássicos da literatura que apresentam taxa de convergência lenta e vários ótimos locais.

6.2 Tratamento de Restrições de Igualdade e Desigualdade

Os problemas de otimização são inerentemente constituídos por restrições, advindas de limitações operacionais, físicas, ambientais, econômicas, entre outras. Neste sentido, na literatura são apresentadas algumas abordagens para a manipulação dessas restrições em problemas com um único objetivo, que podem ser estendidas para o caso multi-objetivo. Nas próximas subseções serão apresentados os principais métodos para o tratamento de restrições (COELHO, 2004).

6.2.1 Métodos Baseados em Funções de Penalidade

Os métodos baseados em penalidades constituem, sem sombra de dúvida, a abordagem de concepção mais simples e a mais utilizada na literatura para a penalização de violações em restrições de trajetória. Seu procedimento consiste em reescrever o problema original com restrições, gerando um problema sem restrições, através da redefinição da função objetivo, de maneira que qualquer violação de qualquer uma das restrições é penalizada, forçando a solução a se manter na região viável. Matematicamente, essa redefinição do problema original é dada por:

$$\min \Phi(x, r_p) = f(x) + r_p P(x) \quad (6.1)$$

onde $\Phi(x, r_p)$ é a função Pseudo-objetivo, $P(x)$ é denominada função de penalidade e r_p é um escalar associado a $P(x)$.

Dentre as principais metodologias existentes para a realização dessa transformação, pode-se destacar três classes principais (VANDERPLAATS, 1999; EDGAR et al., 2001):

a) Método da Função de Penalidade Interior (*Interior Penalty Function Method*):

Penaliza a função pseudo-objetivo ainda no interior da região viável, através de uma

tolerância pré-estabelecida pelo usuário. A função de penalidade é dada por:

$$P(x) = r'_p \sum_{j=1}^m \frac{-1}{g_j(x)} + \sum_{k=1}^l (h_k(x))^2 \quad (6.2)$$

Alternativamente, pode-se também utilizar:

$$P(x) = \sum_{j=1}^m -\log[-g_j(x)] \quad (6.3)$$

que é freqüentemente recomendada por ser melhor condicionada numericamente que a fornecida pela Eq. (6.2). Observe que $g_j(x)$ deve ser uma parcela negativa, $-1/g_j(x)$ será um número positivo que se aproximará do infinito quando $g_j(x) \rightarrow 0$ (VANDERPLAATS, 1999). Ainda segundo o autor, o parâmetro r'_p é inicialmente tomado como sendo um valor elevado (da ordem de 10^6), e este, durante o processo iterativo, é reduzido por um fator de 0,3.

b) Método da Função de Penalidade Exterior (*Exterior Penalty Function Method*):

Penaliza a função pseudo-objetivo quando qualquer uma das restrições é violada. Nesse caso, a função penalidade é dada por:

$$P(x) = \sum_{j=1}^m (\max[0, g_j(x)])^2 + \sum_{k=1}^l (h_k(x))^2 \quad (6.4)$$

Um ponto interessante a se destacar é o valor de r_p . Se este assumir pequenos valores (na ordem de 10), $\Phi(x, r_p)$ é facilmente minimizada, contudo com grandes violações nas restrições. Por outro lado, para valores elevados de r_p (na ordem de 10^6), as restrições serão atendidas, com relação a uma tolerância pré-definida. Contudo esse valor de r_p pode levar a um mau condicionamento numérico do procedimento usado (VANDERPLAATS, 1999). Usualmente, ainda segundo o autor, o processo iterativo é inicializado com r_p igual a 1, sendo que a cada iteração, tal valor é aumentado através de um fator de correção (como por exemplo, 3).

c) Método dos Multiplicadores de Lagrange Aumentado (*Augmented Lagrange Multiplier Method*):

Esta técnica estabelece, a partir das condições de Kuhn-Tucker, uma função pseudo-objetivo que combina a função objetivo inicial com os multiplicadores de Lagrange aplicados às restrições e com funções de penalidade exterior. A principal vantagem deste método é a redução da dependência da escolha de parâmetros de penalidade e da forma

pela qual estes são atualizados durante o processo de otimização. Nesta abordagem, a função de penalidade adquire a seguinte forma:

$$P(x) = \sum_{j=1}^m (\lambda_j \psi_j + r_p \psi_j^2) + \sum_{k=1}^l (\lambda_{k+m} h_k(x) + r_p (h_k(x))^2) \quad (6.5)$$

onde

$$\psi_j = \max \left(g_j(x), -\frac{\lambda_j}{2r_p} \right) \quad (6.6)$$

com fórmulas de atualização dos multiplicadores dadas por:

$$\begin{cases} \lambda_j^{p+1} = \lambda_j^p + 2r_p \left(\max \left(g_j(x), -\frac{\lambda_j^p}{2r_p} \right) \right) & j = 1, \dots, m \\ \lambda_{k+m}^{p+1} = \lambda_{k+m}^p + 2r_p h_k(x) & k = 1, \dots, l \end{cases} \quad (6.7)$$

para $j = 1, \dots, k$.

Um aspecto a ser destacado é que esse fator de penalidade pode ser estático ou dinâmico. No primeiro caso, para cada restrição, um fator de penalidade é mantido constante durante todo o processo de otimização. Segundo Vanderplaats (1999) e Edgar et al. (2001), um valor muito grande para esse fator pode conduzir à convergência prematura, próxima de regiões não viáveis, enquanto que, para valores muito pequenos, a convergência pode ser demasiadamente lenta. Como os fatores de penalidade estática são difíceis de ser arbitrados, Vanderplaats (1999) e Edgar et al. (2001) aconselham o uso de fatores que são dinamicamente atualizadas a cada iteração. Segundo os autores, inicialmente se faz uso de valores pequenos para esses fatores, permitindo uma ampla exploração do espaço de busca, com progressivo incremento dessas penalidades ao longo do processo iterativo.

Uma outra classe interessante de métodos de penalidade são os Métodos de Barreira. Assim como os métodos de penalidade, os Métodos de Barreira, convertem um problema de otimização restrito em um problema irrestrito. A solução ótima para este sub-problema está no interior da região viável, onde $g_j(x)$ é positivo, convergindo à solução com restrição através de um parâmetro escalar positivo denominado parâmetro de barreira. O principal contraste com o comportamento de métodos de penalidade está no fato de que as soluções do sub-problema sem restrição convergem fora da região viável (EDGAR et al., 2001).

Matematicamente essa abordagem pode ser escrita da seguinte forma:

$$\min B(r_b, x) = f(x) - r_b \sum_{j=1}^J \ln(g_j(x)) \quad (6.8)$$

onde $B(r_b, x)$ é a nova função objetivo e $g_j(x)$ são as j -ésimas restrições de desigualdade maiores que zero. O termo $B(r_b, x)$ é muito difícil de ser minimizado, pois este tende rapidamente ao infinito com o movimento da fronteira da região viável. Devido a isso, esse método não é usado como um método direto na resolução de problemas de programação não linear (EDGAR et al., 2001).

Como destacado anteriormente, a principal dificuldade na utilização dos métodos de penalidade é a escolha do valor do parâmetro de penalidade (VANDERPLAATS, 1999; EDGAR et al., 2001; DEB, 2001). Para tentar superar essa dificuldade, Deb (2000) propôs uma abordagem que não faz uso de fatores de penalidade. Esta consiste, inicialmente, da transformação das restrições de igualdade, se existirem, em desigualdades, da seguinte maneira:

$$-\delta \leq h_k(x) \leq \delta \quad (6.9)$$

onde δ é um número positivo de ordem de grandeza muito pequena. Assim, o método proposto consiste em adicionar apenas as violações das restrições à função objetivo, como apresentado abaixo:

$$\Phi(x, r_p) = \begin{cases} f(x) & \text{se } g_j(x) \leq 0 \ (\forall j = 1, \dots, J) \text{ e } -\delta \leq h_k(x) \leq \delta \ (\forall k = 1, \dots, K) \\ f_{max} + \sum_{j=1}^J g_j(x) + \sum_{k=1}^K |h_k(x)| & \text{caso contrário} \end{cases} \quad (6.10)$$

onde o parâmetro f_{max} é a pior solução viável na população.

6.2.2 Método da Pena de Morte

É um dos caminhos mais diretos para o tratamento de restrições. Este método consiste, no caso de algoritmos genéticos, em simplesmente eliminar as soluções não viáveis. Contudo, segundo van Kampen et al. (1996), a principal desvantagem dessa abordagem é que nenhuma informação sobre o domínio não viável é explorada.

6.2.3 Métodos Baseados na Busca por Soluções Viáveis

Existem alguns métodos que priorizam a distinção entre as soluções viáveis e não-viáveis no espaço de busca S . Um método, proposto por Schoenauer e Xanthakis (1993), chamado Abordagem de Memória Comportamental (*Behavioral Memory Approach*), considera o problema com restrições em sequência, na qual a mudança de atendimento de uma restrição para outra é feita a partir da geração de um número pré-determinado de indivíduos viáveis.

Uma segunda abordagem, descrita em Powell e Skolnick (1993), é baseada na abordagem clássica de função de penalidade. Cada indivíduo é avaliado pela seguinte relação:

$$eval(x) = f(x) + r \sum_{j=1}^m f_j^2(x) + \theta(x) \quad (6.11)$$

onde r é uma constante. Contudo, o componente $\theta(x)$ é um termo adicional no método tradicional de função de penalidade, que influencia as avaliações das soluções não-viáveis. A distinção entre soluções viáveis e não-viáveis é feita através da adição da seguinte regra heurística: para qualquer indivíduo viável x e qualquer indivíduo não-viável y , $f(x) < f(y)$, isto é, qualquer solução viável é melhor que qualquer uma não-viável ³.

Deb (2001) apresenta em seu trabalho uma modificação nessa abordagem através da utilização de um operador de seleção por torneio associada com a seguinte função de avaliação:

$$eval(x) = \begin{cases} f(x), & \text{se } x \text{ viável} \\ f_{max} + \sum_{j=1}^m f_j(x), & \text{nos outros casos} \end{cases} \quad (6.12)$$

onde f_{max} é o pior valor da solução viável na população. A principal diferença dessa abordagem com a desenvolvida por Powell e Skolnick (1993) é que esta não considera a solução não-viável. O algoritmo busca por soluções viáveis e, quando um número pré-determinado de soluções viáveis é encontrado, este encontra as melhores soluções viáveis mantendo a diversidade das soluções na região viável. É importante observar que, neste caso, não é necessário o uso do coeficiente de penalidade r , devido ao fato de que as soluções viáveis são sempre avaliadas para serem melhores que as não-viáveis. O autor sugere a normalização das restrições.

Um outro método (GENECOP III), proposto em Michalewicz e Nazhiyath (1995) é fundamentado na idéia do reparo de indivíduos não-viáveis. O GENECOP III incorpora o sistema original do Genecop, extendendo principalmente para duas populações separadas,

³Para problemas de minimização.

onde o desenvolvimento de uma população influencia a avaliação dos indivíduos da outra população. A primeira população, P_s , consiste na busca por pontos do espaço viável $\mathfrak{S}$, que satisfazem as restrições lineares do problema. A viabilidade desses pontos é mantida através de operadores especiais. A segunda população P_r consiste de pontos viáveis e que satisfazem todas as restrições.

6.2.4 Métodos Baseados na Preservação das Soluções Viáveis

Essa abordagem é baseada na utilização de operadores especializados que transformam indivíduos não-viáveis em indivíduos viáveis. O método assume apenas restrições lineares e a inicialização se dá a partir de uma configuração viável. As restrições lineares de igualdade são utilizadas para que se possa eliminar algumas variáveis a partir destas restrições. As restrições de desigualdade são atualizadas adequadamente a partir dos operadores que garantem que uma região viável seja respeitada. Um exemplo da aplicação desta abordagem é o GENECOP (*Genetic Algorithm for Numerical Optimization of Constrained Problems*) (MICHALEWICZ; JANIKOW, 1991; MICHALEWICZ et al., 1994).

6.2.5 Métodos Baseados em Decodificadores

Freqüentemente, os decodificadores representam uma opção interessante para os usuários de técnicas evolucionárias. Nessas técnicas, o cromossomo “dá instruções” de como construir a solução viável.

Cada decodificador impõe um traçador T entre a solução viável e a solução decodificada. Para isso, algumas condições devem ser satisfeitas (MICHALEWICZ et al., 1999):

- Para cada solução $s \in \mathfrak{S}$ existe uma solução codificada d ;
- Cada solução codificada d corresponde a uma região viável s ;
- Todas as soluções em $\mathfrak{S}$ poderão ser representadas pelo mesmo número d codificado;
- A transformação de T é computacionalmente rápida.

6.3 Atualização dos Parâmetros em Algoritmos Evolutivos

Normalmente, durante o processo evolutivo os parâmetros desses algoritmos são mantidos constantes. Para cada abordagem existem valores típicos para esses parâmetros que apresentam bons resultados para uma gama de problemas matemáticos e de engenharia. Entretanto, o uso de tais valores não garante a capacidade de prevenção de convergência prematura que estes acrescentam aos métodos evolutivos (COELHO; MARIANI, 2006). Além disso, sabe-se que os algoritmos evolutivos são sensíveis a escolha dos parâmetros de controle (GAMPERLE et al., 2002; PRICE et al., 2005) e são altamente dependentes do problema (GAMPERLE et al., 2002; ZAHARIE, 2002; ZAHARIE, 2003; BREST et al., 2006; QIN; SUGANTHAN, 2005). De acordo com Nobakhti e Wang (2006), por causa do mecanismo de mutação proposto no algoritmo de ED, uma má escolha no valor da taxa de perturbação F pode fazer com que a população perca diversidade.

Para superar tais dificuldades, algumas metodologias tem sido propostas. Ali e Torn (2004) propuseram atualizar o valor da taxa de perturbação através da seguinte relação

$$F = \begin{cases} \max \left(F_{min}, 1 - \left| \frac{f_{max}}{f_{min}} \right| \right) & \text{se } \left| \frac{f_{max}}{f_{min}} \right| < 1 \\ \max \left(F_{min}, 1 - \left| \frac{f_{min}}{f_{max}} \right| \right) & \text{caso contrário} \end{cases} \quad (6.13)$$

onde f_{min} , f_{max} são respectivamente os valores mínimo e máximo da função objetivo na população e F_{min} é 0,4 (STORN; PRICE, 1995; PRICE et al., 2005). Segundo os autores, o valor de F reflete a demanda para fazer com que a busca seja mais diversificada no início do processo evolutivo, isto é, para produzir maiores valores de F para grandes diferenças $f_{max}-f_{min}$, e menores valores de F caso contrário.

Segundo Brest et al. (2006) os valores de F e de CR podem ser mudados em cada geração g com probabilidade τ_1 e τ_2 , respectivamente. O valor de F , não geração $g+1$, é uniformemente distribuído entre F_{min} e F_{max} :

$$F(g+1) = \begin{cases} F_{min} + rand_1 \times F_{max} & \text{se } rand_2 < \tau_1 \\ F(g) & \text{caso contrário} \end{cases} \quad (6.14)$$

O novo valor para a taxa de cruzamento CR é um número randômico entre 0 e 1 de

acordo com a relação:

$$CR(g+1) = \begin{cases} rand_3 & \text{se } rand_4 < \tau_2 \\ CR(g) & \text{caso contrário} \end{cases} \quad (6.15)$$

Em ambas as relações (Eq.(6.14) e Eq.(6.15)), $rand_j \in \{1, 2, 3, 4\}$ e são valores randômicos uniformes entre 0 e 1. τ_1 e τ_2 são valores fixos tomados como sendo 0,1 e 0,1, respectivamente.

Zaharie (2002) e Zaharie (2003) propuseram uma abordagem que atualiza os parâmetros F e CR através da diversidade da população quantificada através da variância da população dada por:

$$Var(x) = \overline{x^2} - \bar{x}^2 \quad (6.16)$$

onde $\bar{x} = \sum_{i=1}^m x_i/N$ (N o tamanho da população).

Segundo Zaharie (2002) e Zaharie (2003) o valor esperado para a variância da população obtida após a recombinação dos indivíduos se o melhor indivíduo da população não é considerado é:

$$E(Var(x)) = \left(2F^2CR + 1 - \frac{2CR}{N} - \frac{CR^2}{N} \right) Var(x) \quad (6.17)$$

Considerando que $x(g)$ é a população obtida na geração $g-1$ (população inicial para a geração g) e que durante a geração g esta é transformada em x' (por recombinação) e em x'' (pelo processo de seleção). x'' representará a nova população, isto é, a população inicial na geração $x(g+1)$.

Definindo γ como $Var(x(g+1))/Var(x(g))$, podemos verificar a tendência da variância da população. Se $\gamma < 1$, podemos compensar o incremento da variância, acelerando assim a convergência e correndo o risco de induzir o processo a convergência prematura. Se $\gamma > 1$, podemos compensar o alto decréscimo da variância, evitando assim o risco de convergência prematura.

A idéia da atualização dos parâmetros através do conceito de variância consiste na escolha dos parâmetros F e CR que compensem o efeito da aplicação prévia dos mecanismos

de recombinação e de seleção:

$$1 + 2F^2CR - \frac{2CR}{NP} + \frac{CR^2}{NP} = c \quad (6.18)$$

onde $c \equiv \gamma \frac{Var(x(g+1))}{Var(x(g))}$.

Resolvendo a Eq. (6.18) para F :

$$F = \begin{cases} \sqrt{\frac{1}{N}} \sqrt{\frac{\eta}{2CR}} & \text{se } \eta \geq 0 \\ F_{min} & \text{caso contrário} \end{cases} \quad (6.19)$$

onde $\eta \equiv N(c-1) + CR(2-CR)$ e F_{min} o mínimo valor de F . Uma condição suficiente para o incremento da variância da população por recombinação é que $F > \sqrt{1/N}$, então podemos usar $F_{min}=1/\sqrt{N}$. Um valor superior para F pode ser imposto segundo sugerido por Storn e Price (1995) ($F_{max}=2$). De forma similar podemos atualizar para CR .

Uma outra linha de pesquisa para a atualização de parâmetros evolutivos que vem ganhado espaço é o uso de modelos baseados na Teoria do Caos acoplados a abordagens Determinísticas e Não Determinísticas. Tal direcionamento se deve à incorporação de diversidade e a capacidade de prevenção de convergência prematura acrescentadas a essas metodologias (TAVAZOEI; HAERI, 2007). Dentre os trabalhos mais recentes, pode-se citar o trabalho de Coelho e Mariani (2006), que faz uso de modelos caóticos associados a um algoritmo híbrido que usa Evolução Diferencial e Programação Quadrática Sequencial. Já em Tavazoei e Haeri (2007) é realizado um estudo sobre o desempenho de diferentes modelos caótico de busca quando estes são incorporados a abordagens clássicas de otimização. Yang et al. (2007) propõem combinações entre o Método de Monte Carlo, o Algoritmo Clássico BFGS (Broyden-Fletcher-Goldfarb-Shanno) (VANDERPLAATS, 1999) e alguns modelos caóticos para formar algoritmos híbridos que são aplicados a um conjunto de funções matemáticas. Uma versão híbrida do Algoritmo de Enxame de Partículas com a incorporação de Mapas Logísticos pode ser encontrada em Alatas et al. (2008). Nesta linha de atuação, Coelho e Mariani (2008) aplicaram o Algoritmo de Colônia de Formigas com Mapas Logísticos em dois problemas de projeto de engenharia e Lobato et al. (2007a) realizaram um estudo sobre a utilização de modelos caóticos incorporados ao algoritmo de Evolução Diferencial aplicado ao projeto de uma viga.

No contexto multi-objetivo, Qian e li (2008) propuseram o algoritmo ADEA (*Adaptive Differential Evolution Algorithm*), que incorpora um novo operador de seleção baseado na idéia de *ranking* e um mecanismo para a atualização da taxa de perturbação em cada

geração, baseada na diversidade da população na curva de Pareto da geração corrente.

Até o presente momento foi apresentado uma revisão detalhada sobre otimização multi-objetivo com destaque nas metodologias existentes para a sua resolução, tratamento de restrições e atualização de seus parâmetros. No próximo capítulo é descrito o algoritmo baseado em Evolução Diferencial proposto neste trabalho. É dada ênfase em cada um dos seus operadores, além de apresentar os aspectos gerais de implementação e estruturação do algoritmo.

OTIMIZAÇÃO MULTI-OBJETIVO VIA EVOLUÇÃO DIFERENCIAL



escopo deste trabalho é apresentado neste capítulo com o detalhamento do algoritmo multi-objetivo proposto. Suas linhas gerais são minuciosamente discutidas, levando em consideração suas principais características em relação às metodologias que já foram propostas na literatura.

7.1 Aspectos Gerais

Conforme anteriormente apresentado, o Algoritmo de Evolução Diferencial, concebido inicialmente para a solução de problemas com um único objetivo, foi proposto por Storn e Price (1995). Esta abordagem difere de outras técnicas Não Determinísticas por não apresentar concepção inspirada na genética, em fenômenos físicos ou no comportamento de colônias de pássaros ou de insetos.

Apesar dessa metodologia ser relativamente recente, ela já é consagrada no tratamento de problemas com um único objetivo (STORN; PRICE, 1995; STORN, 1999; PAHNER; HAMMEYER, 2000; KAPADI; GUDI, 2004; LOBATO; STEFFEN JR, 2006; LOBATO et al., 2007a; LOBATO et al., 2007b). Segundo Babu et al. (2005) e Angira e Babu (2005), essa abordagem se desenvolve de tal forma que logo irá superar os atuais algoritmos genéticos. Ainda segundo esses autores, suas principais vantagens são: robustez, simplicidade conceitual e fácil implementação. Contudo, apesar dos trabalhos citados anteriormente, existem poucas teses em andamento ou finalizadas, dentre as quais pode-se citar o trabalho de Xue (2004) e de Oliveira (2006a).

Neste sentido é proposto o Algoritmo MODE (*Multi-objective Optimization Differential Evolution*). De uma maneira geral, em sua concepção são empregados os seguintes operadores: ordenamento por *rank*, truncamento das soluções não-dominadas, seleção e exploração das vizinhanças dos indivíduos da população. Tais operadores serão descritos nas próximas seções.

Cabe salientar que durante a concepção deste trabalho foi implementada uma primeira versão do algoritmo proposto em MATLAB®. Entretanto, devido às conhecidas vantagens das plataformas C e FORTRAN com relação a outras, optou-se pela implementação do algoritmo também em C, a partir da estrutura do programa NSGA II implementado por Deb et al. (2000) e disponível no endereço <http://www.lania.mx/~ccoello/EMOO>. A escolha desse ambiente está diretamente relacionada à facilidade e velocidade no manuseio de operações matriciais e vetoriais com maior eficiência.

7.2 O Algoritmo MODE

De uma forma geral, as principais características do algoritmo proposto são: ordenamento das soluções através do mecanismo de *rank*, no qual classifica-se as soluções em pseudo-curvas baseadas no critério de dominância de Pareto; exploração das vizinhanças dos indivíduos da população a partir da geração de vizinhos; truncamento de soluções não-dominadas com o intuito de garantir a diversidade das soluções através de um número fixo de indivíduos na população; incorporação da estratégia de seleção e de elitismo com o intuito de utilizar os melhores indivíduos para a formação de candidatos em potencial.

O algoritmo MODE apresenta os seguintes parâmetros:

Número de Gerações (N_{gen}), Número de Variáveis de Projeto (V) e seus respectivos domínios ($L \leq \mathbf{x} \leq U$), Número de Objetivos (N_{obj}), Tamanho da População (N), Probabilidade de Cruzamento (p_c), Taxa de Perturbação (F), Número de Pseudo-Curvas (R) e Taxa de Redução (r).

Resumidamente, o MODE obedece a seguinte seqüência de procedimentos:

- ❶ Gerar a população inicial randomicamente e avaliá-la segundo as funções-objetivo. Se houver violação do domínio de qualquer uma das variáveis durante a geração, esta deve ser corrigida através da regra de reparo dada pela Eq. (5.29);
- ❷ Classificar a população segundo o critério de dominância através do ordenamento das soluções não-dominadas por meio de *rank* e do truncamento dessas soluções;

Enquanto o número de gerações não for alcançado:

- ③ Gerar N candidatos a partir da combinação de indivíduos selecionados aleatoriamente através do operador de reprodução. Esses candidatos são agrupados com a população inicial formando uma nova população P^1 de tamanho $2N$;
- ④ O próximo passo do algoritmo consiste na eliminação dos N indivíduos dominados da população P^1 formando a população P^2 . A partir daí são gerados N vizinhos com relação a P^2 . Os vizinhos são acrescentados à população P^2 gerando assim a população P^3 ;
- ⑤ P^3 então é classificada com relação ao critério de dominância através do ordenamento por *rank* e submetida ao operador de truncamento para manter a diversidade da população. Essa população atualizada, de tamanho igual a N , é armazenada em um filtro, que é atualizado a cada geração, para que nenhuma solução não-dominada seja perdida durante o processo evolutivo.

A saída do MODE é o conjunto de soluções não-dominadas. O detalhamento de cada operador utilizado no algoritmo é apresentado nas seções seguintes.

7.2.1 Inicialização

Classicamente nos algoritmos evolutivos, o processo de inicialização consiste na geração de uma população inicial, obtida de forma randômica. Para tal, necessita-se apenas do domínio de cada variável de projeto. A partir daí, geram-se números randômicos que serão aplicados a esse intervalo, obtendo-se assim um vetor de indivíduos da população.

Contudo, pode-se encontrar na literatura um outro processo de inicialização da população através da utilização de uma Distribuição Gaussiana, onde são gerados indivíduos com média e desvio padrão definidos pelo usuário (ABBASS et al., 2001).

Neste trabalho foi utilizado a primeira abordagem citada, onde indivíduos são gerados aleatoriamente, dentro do domínio especificado.

7.2.2 Operador de Ordenamento por *Rank*

Goldberg (1989) propôs em seu trabalho variantes de como se pode, além de avaliar a adaptação de um indivíduo, classificá-lo em termos do critério de dominância (EDGEWORTH, 1881; PARETO, 1896). Fonseca e Fleming (1993) propuseram um esquema de

avaliação da adaptação baseado em um esquema de *rank*, que incorporava informações sobre a diversidade das soluções no valor da função objetivo. Desta maneira, os indivíduos não-dominados na população tem *rank* 1 e representam o maior valor de aptidão, enquanto o valor de aptidão para os outros indivíduos da população correspondem ao número de soluções não-dominadas mais uma unidade.

Uma das metodologias de maior evidência na literatura para classificação de indivíduos é a desenvolvida por Deb et al. (2000) no algoritmo NSGA II. A idéia básica é a realização de um ordenamento elitista por não dominância. Esta abordagem consiste na organização dos indivíduos da população em fronteiras que refletem sua importância no processo evolutivo da seguinte maneira: inicialmente, através do critério de dominância, a população é classificada e tomada como *rank* 1. Esses indivíduos de *rank* 1 são retirados da população. A população restante é novamente classificada segundo esse critério de dominância, sendo que essa população assume *rank* 2. Esses indivíduos são retirados da população atual e novamente é realizada a classificação dos indivíduos que restaram. Tal procedimento é repetido até que todos os indivíduos da população sejam classificados.

A estrutura de ordenamento por *rank* é apresentada na Fig. 7.1.

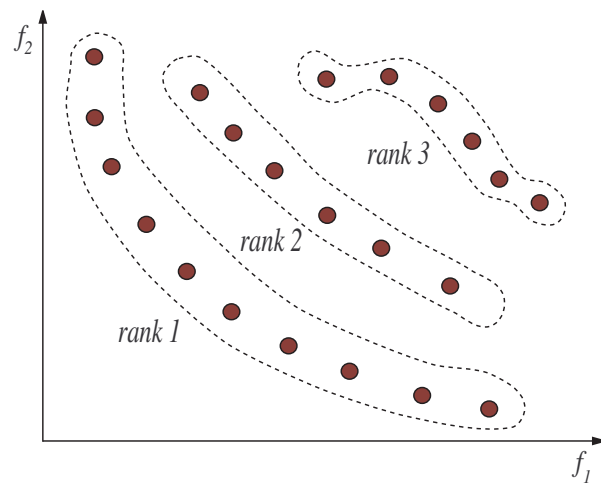


Figura 7.1: Estratégia de ordenamento por *rank* (Reproduzido de Deb (2001)).

A classificação por ordenamento das soluções favorece a evolução das curvas ao invés de pontos individuais, permitindo uma melhor distribuição das soluções no espaço de busca (SILVA, 2004). Essa abordagem tem sido utilizada na elaboração de vários algoritmos (CASTRO, 2001; SILVA, 2004; XUE, 2004; BABU et al., 2005).

7.2.3 Operador de Truncamento das Soluções

Ao longo do processo evolutivo, sabe-se que o número de soluções não-dominadas pode aumentar muito, o que implica no aumento do tempo de processamento. Dessa forma,

faz-se necessário o acoplamento de um operador específico para a redução do número de indivíduos. Tal mecanismo tem por objetivo eliminar as soluções não-dominadas que estão muito próximas umas das outras, garantindo a diversidade na população, já que não é interessante se ter uma grande quantidade de indivíduos, mas sim que eles estejam bem distribuídos ao longo da curva de Pareto (DEB, 2001).

Neste contexto, o mecanismo utilizado no MODE para manter a diversidade é o mesmo descrito no NSGA II (Seção 5.2.5): a Distância Métrica Aglomerada (*Crowded Distance Metrics*), também denominada de Distância da Multidão (TICONA, 2003). De forma simplificada, essa abordagem consiste em, após aplicar o operador de ordenamento por *rank*, calcular a distância entre os indivíduos da população através do valor da função aptidão. O procedimento para achar a distância de aptidão é descrito na Tab. 7.1, onde I_i^m representa a i -ésima solução na lista ordenada pelo objetivo m . I_1^m e I_l^m são os elementos da lista com menor e maior valor em um objetivo m . $f_m^{I_{i+1}^m}$ e $f_m^{I_{i-1}^m}$ são os valores dos vizinhos de i na m -ésima função objetivo. $f_m^{\max}$ e $f_m^{\min}$ são parâmetros dos limites máximo e mínimo em cada objetivo.

Tabela 7.1: Distância da multidão.

<i>Parâmetros de Entrada:</i>
$\mathfrak{S}_j \rightarrow$ Conjunto de soluções na fronteira i
l denota o número de soluções em $\mathfrak{S}_j$
Para cada solução em $\mathfrak{S}_j$ atribui-se $d_i = 0$
Para cada função objetivo $m = 1, \dots, M$
Ordenar decrescentemente as soluções por f_m na lista I^m
Para cada solução extrema (mínimo e máximo) em cada um dos M
objetivos fazer $d_{I_1^m} = \infty$
Para as soluções $i = 1, \dots, l - 1$ calcular $d_{I_i^m}$ através de Eq. (7.1)

A Equação (7.1) assegura que as soluções mais afastadas tenham d_i maior. Alternativamente, é possível usar também o operador de compartilhamento para calcular o contador de nicho.

$$d_{I_i^m} = d_{I_i^m} + \frac{f_m^{I_{i+1}^m} - f_m^{I_{i-1}^m}}{f_m^{\max} - f_m^{\min}} \quad (7.1)$$

O operador distância da multidão d_i de uma solução i representa uma estimativa do perímetro formado pelo cubóide cujos vértices são os seus vizinhos mais próximos. A Figura 7.2 mostra a distância da multidão para a solução i . Quanto maior o cubóide de i , mais afastada se encontra i dos seus vizinhos. Já as soluções extremas em cada objetivo

terão um cubóide infinito.

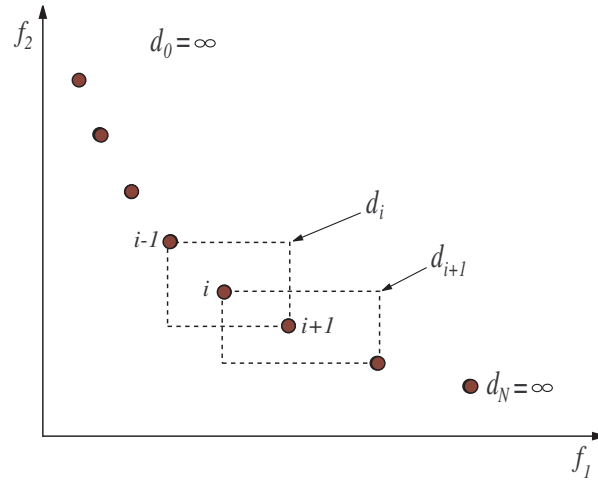


Figura 7.2: Operador distância da multidão (Reproduzido de Deb (2001)).

7.2.4 Operador de Seleção

Na prática qualquer operador de seleção utilizado em otimização com um único objetivo via Métodos Não Determinísticos poderia ser aplicado para otimização com múltiplos objetivos. Segundo Deb (2001), o operador de seleção consiste em guiar os indivíduos em direção do espaço de busca no qual as soluções têm maiores valores de aptidão e estão menos aglomeradas. Em otimização com um único objetivo, a competição entre pais e filhos é muito interessante. Contudo, no contexto multi-objetivo, essa prática não é aconselhável, pois os pais com relação aos filhos podem ser não-dominados. Deb (2001) ressalta a importância de se colocar ambos na mesma população para que se possa, a partir daí, capturar os indivíduos não-dominados dentro dessa população.

Essa idéia, com algumas diferenças, é a fundamentação teórica do algoritmo NSGA II, onde os indivíduos são primeiramente comparados segundo o *rank* e a distância dos indivíduos que tem o mesmo *rank*, se no processo de seleção houver empate. Nessa abordagem, somente os indivíduos com menor *rank* poderão ser escolhidos para o processo de reprodução, o que impede que indivíduos com maior *rank* possam ser utilizados no processo de reprodução. Todavia, como relatado por Deb (2001), essa forte estratégia de elitismo utilizada no NSGA II não produz bons resultados, pois seu uso poderá levar a uma solução de baixa densidade, isto é, com baixa diversidade de soluções. Deb e Goel (2001) dão ênfase à importância de se manter a diversidade no *rank* e permitir que indivíduos com maior *rank* possam entrar na próxima geração.

Dessa forma, no algoritmo MODE, todos os pais e filhos são colocados dentro da mesma população, independentemente do seu *rank* e de sua distância métrica aglomerada, para que possam competir entre si. Desse modo, um indivíduo de alto *rank* (pior valor de aptidão), poderá ser escolhido no processo de seleção e gerar um filho não-dominado que fará parte da população na próxima geração.

7.2.5 Operador de Reprodução

Um dos fatores que distinguem os algoritmos evolutivos é a maneira como os filhos são gerados. Esse mecanismo é o operador de reprodução, que consiste da geração de candidatos em potencial para substituir, segundo o critério de dominância, indivíduos da população atual.

Segundo Xue (2004), no algoritmo de Evolução Diferencial, esse operador é constituído por duas partes: o vetor diferencial e o vetor perturbação. Quando se trabalha com um único objetivo, esse vetor diferencial é definido como a diferença entre o melhor indivíduo da população e aquele considerado para reprodução. Entretanto, segundo a abordagem multi-objetivo, esse melhor indivíduo é qualquer um pertencente à população cujo *rank* é 1. Já o vetor perturbação é obtido pela diferença entre dois indivíduos da população escolhidos aleatoriamente. A quantidade de pares de indivíduos que serão tomados para a formação desse candidato é um parâmetro que deve ser informado pelo usuário. Uma vez definidos esses pares, o processo de reprodução é similar ao que foi apresentado na Seção 5.8 para problemas com um único objetivo.

Angira e Babu (2005), Babu et al. (2005) e Babu e Anbarasu (2005) constataram em seus trabalhos que, os melhores resultados em termos de convergência e diversidade, para os problemas estudados, foram obtidos quando utilizada a estratégia DE/RAND/1/BIN. Em contra partida, recentemente, Oliveira (2006a) realizou um estudo das diversas estratégias aplicadas a alguns estudos de caso e verificou que os resultados são pouco sensíveis à escolha das estratégias. Diante dessa constatação, o MODE faz uso apenas da estratégia DE/RAND/1/BIN em seu operador de reprodução.

7.2.6 Operador de Elitismo

Quando se trabalha com um único objetivo, a melhor solução da geração atual é utilizada como valor de referência para a geração seguinte. Contudo, para problemas com mais de um objetivo, essa idéia de levar a melhor solução para uma próxima geração deve ser feita através do operador de elitismo. Este procedimento garante a propagação dos

melhores indivíduos no processo evolutivo, aumentando a convergência e a robustez do algoritmo (DEB, 2001; CASTRO, 2001; SILVA, 2004).

Segundo Castro (2001), a utilização desse operador permite que o conjunto de soluções seja gerado durante todo o processo evolutivo, enquanto que na sua ausência, as soluções resultariam exclusivamente da última geração. Assim, a perda de pontos do conjunto de Pareto durante o processo é evitada, e as soluções obtidas no final do processo são constituídas dos pontos pertencentes ou mais próximos ao conjunto de Pareto. A cada geração, as soluções não-dominadas são armazenadas em um filtro. Segundo Deb (2001), esse filtro não influencia o processo de busca, evitando, assim, uma pressão de seleção excessiva, que seria originada por uma população constituída basicamente por pontos ótimos.

7.2.7 Operador para a Exploração das Vizinhanças

A idéia em se explorar as vizinhanças de indivíduos da população pode ser vista como um tipo de refinamento ou procedimento de busca local. Procedimentos de refinamento ou de busca local, através de acoplamento entre algoritmos, têm sido usados com sucesso em diversos trabalhos (DULIKRAVICH et al., 1999; GOEL; DEB, 2001; ISHIBUCHI; YOSHIDA, 2003; HU et al., 2003; ROJAS et al., 2004; LOBATO; STEFFEN JR, 2006).

Hu et al. (2006) propuseram uma nova metodologia para realizar refinamentos nas soluções não-dominadas. Os autores se basearam no *Line-up Competition Algorithm* - LCA, proposto por Yan (1998) para problemas de minimização com um único objetivo. O LCA consiste na geração de vizinhos dos indivíduos da população segundo o procedimento descrito a seguir:

- ❶ Gerar e avaliar a população inicial de tamanho N randomicamente.

Enquanto o número de gerações não for alcançado:

- ❷ Ordenar os N filhos em ordem crescente de acordo com o valor da função objetivo;
- ❸ Determinar uma vizinhança para cada indivíduo de acordo com sua posição relativa definida no item anterior. O indivíduo com melhor valor de aptidão (problemas de minimização) tem o domínio da vizinhança menor, comparado ao que tem o pior valor de função adaptação, que nesse caso, tem o maior domínio de vizinhança;
- ❹ Os pais são submetidos ao operador de mutação λ vezes, produzindo λ filhos. Esses pais e filhos constituem a nova população;

- ⑤ Os $(\lambda+1)$ indivíduos gerados competem com os pais, sendo que os melhores são colocados na próxima geração;
- ⑥ Se o número de soluções ultrapassar um determinado valor definido pelo usuário, utilizar uma métrica para truncá-la.

Nesse algoritmo existem dois níveis de competição: entre os melhores indivíduos de cada família e entre os indivíduos contidos dentro da mesma família. De maneira resumida, aqueles indivíduos que possuem menor valor de *rank* têm menor distância entre esses indivíduos e seus vizinhos. Por outro lado, os indivíduos de maior *rank* têm maior distanciamento com relação a seus vizinhos, o que permite a geração de indivíduos que, após o processo de classificação por *rank*, consigam reduzir seu *rank*.

O desenvolvimento do Operador de Exploração das Vizinhanças está fortemente fundamentado no trabalho de Hu et al. (2006), o qual, primeiramente, consiste no ordenamento por meio de *rank* das soluções não-dominadas.

A Figura 7.3 exemplifica a associação entre a classificação da população e a geração das vizinhanças. Na Figura 7.3 existem 10 soluções no espaço de objetivos que são classificadas por *rank* em 4 grupos, $\{1, 2, 3\}$, $\{4, 5, 6, 7\}$, $\{8, 9\}$ e $\{10\}$. As soluções contidas na curva de *rank* 2 são dominadas pelas de *rank* 1 e assim por diante. Isto implica que, intuitivamente, as soluções contidas na curva de *rank* 1 são melhores que as contidas na curva de *rank* 2.

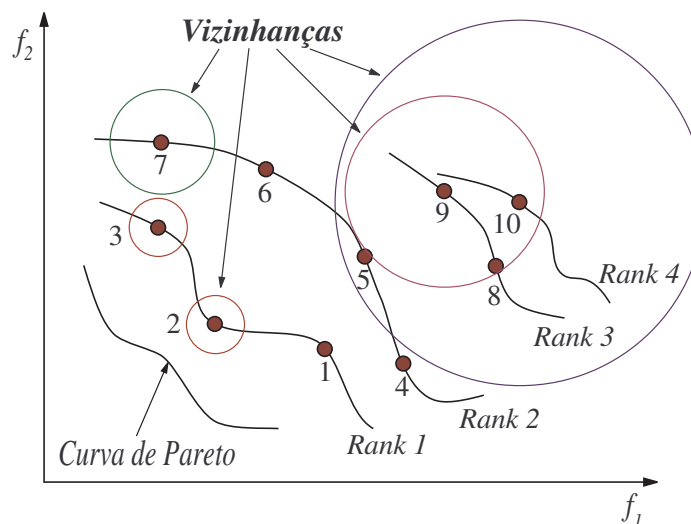


Figura 7.3: Vizinhos de indivíduos em curvas não-dominadas (Reproduzido de Hu et al. (2006)).

Após realizado o processo de classificação das soluções, o próximo passo consiste na obtenção dos vizinhos para cada curva definida por seu *rank*. De forma intuitiva, os

indivíduos que têm menor *rank* deverão apresentar a menor distância com relação a seus vizinhos gerados. Já para os de *rank* 2, deverão apresentar a segunda menor distância, e assim por diante. Entretanto, segundo Deb (2001) e Hu et al. (2006) essa estratégia usada para definir os vizinhos acarretará num aumento de pressão no processo de seleção para a escolha de indivíduos de baixo *rank*, o que pode levar à convergência prematura do processo evolutivo. Para superar essa dificuldade, Hu et al. (2006) propuseram reclassificar a população em pseudo-curvas. Isto é feito da seguinte maneira: após a classificação através do ordenamento por *rank*, os indivíduos ordenados segundo esse critério são reclassificados de acordo com um número pré-definido de indivíduos que deverão estar em cada pseudo-curva. Este procedimento permite que indivíduos que tenham *ranks* diferentes possam estar presentes na mesma curva.

Para esta finalidade, a seguinte distribuição geométrica é adotada:

$$n_k = r n_{k-1}, \quad k = 2, \dots, R \quad (7.2)$$

onde n_k é o número de indivíduos da k -ésima curva, R é o número de pseudo-curvas e r é a taxa de redução. Para uma dada população com N indivíduos, n_k é dado por:

$$n_k = N \frac{1 - r}{1 - r^R} r^{k-1} \quad (7.3)$$

Para $r < 1$, o número de indivíduos na primeira pseudo-curva é alto. Conforme pode-se observar na Eq. (7.3), o número de indivíduos vai diminuindo exponencialmente, o que enfatiza a busca local. Já para um valor maior de r , existem mais soluções na última pseudo-curva, o que enfatiza a busca global (HU et al., 2006).

A Figura 7.4 mostra a relação entre pseudo-curvas (*PC*) e a estratégia de ordenamento por *rank* (*Rank*).

Nesse procedimento, segundo Hu et al. (2006), cada curva possui um pseudo *rank* que não é o mesmo obtido através do ordenamento por *rank*. Esse pseudo *rank* serve para decidir o tamanho da fronteira que será aplicado a cada conjunto definido pela quantidade de elementos n_k .

Definida a estratégia para redefinir as pseudo-curvas, o próximo passo consiste em assumir que existam R curvas não-dominadas na g -ésima geração e, sendo D_R o tamanho da vizinhança da última pseudo-curva na g -ésima geração, o tamanho da k -ésima curva nesta geração é dada por:

$$D_k(g) = \frac{k}{R} D_R(g) \quad k = 1, 2, \dots, R \quad (7.4)$$

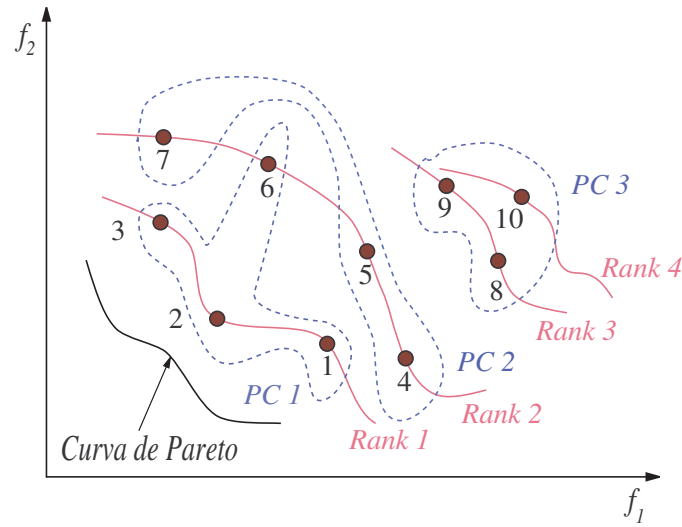


Figura 7.4: Relação entre pseudo-curvas e ordenamento por *rank* (Reproduzido de Hu et al. (2006)).

No algoritmo LCA é sugerido um método bastante simples para obtenção de $D_R(g)$:

$$D_R(g) = \eta D_R(g-1) \text{ ou } D_R(g) = \eta^{g-1} D_R(0) \quad (7.5)$$

onde $0 < \eta < 1$ e $g \geq 1$.

O tamanho máximo da vizinhança, inicialmente, é

$$D_R(0) = U - L \quad (7.6)$$

onde L e U são os limites estabelecidos pelo domínio das variáveis de projeto. Dessa forma, a Eq. (7.4) pode ser expressa como:

$$D_k(g) = \frac{k\eta^{g-1}}{R}(U - L) \quad (7.7)$$

Assim, a vizinhança das variáveis de projeto $\mathbf{x}$ é dada por:

$$\chi_k(\mathbf{x}) = \left[\mathbf{x} - \frac{D_k(g)}{2}, \mathbf{x} + \frac{D_k(g)}{2} \right] \quad (7.8)$$

onde $\chi_k(\mathbf{x})$ é um sub-espço hiper-retangular de $\mathbb{R}^n$.

De posse desses vizinhos, o procedimento de busca pelas soluções não-dominadas pode ser realizado.

7.2.8 Operador para o Tratamento de Restrições

Sabemos que, em sua concepção original, os métodos de otimização não foram idealizados para o tratamento de restrições. O manuseio de restrições, seja na abordagem Determinística ou Não Determinística, requer a incorporação de algum mecanismo que penalize qualquer violação das restrições (Seção 6.2). Classicamente, os métodos de penalização são difíceis de serem manuseados, já que existe a necessidade da escolha de parâmetros de penalização que necessitam de uma definição iterativa e não trivial (VANDERPLAATS, 1999; CASTRO, 2001; EDGAR et al., 2001).

Já ciente das dificuldades dos métodos de penalização, nesta tese, optou-se pelo uso da metodologia proposta por Castro (2001), que consiste de uma estratégia de concepção bastante simples. Esta recomenda a atribuição de valores limites de cada objetivo para serem os parâmetros de penalização. Deste modo, garante-se que qualquer solução não-dominada domine qualquer solução que viole pelo menos uma restrição, do mesmo modo qualquer solução que viole uma restrição dominará qualquer solução que apresente duas violações de restrições e assim por diante. Assim, obtêm-se camadas de soluções, e conseqüentemente ordens, com um mesmo número de violações. Desse modo, para um problema com restrição, o vetor de objetivos é dado por:

$$f \equiv f + r_p n_{viol} \quad (7.9)$$

onde f é o vetor de objetivos, r_p é um vetor de parâmetros, que dependem de cada problema em particular, e n_{viol} é o número de restrições violadas. De modo geral, a determinação de r_p não demanda muitos testes, pois basta escolher-se valores suficientemente elevados e que sejam superiores aos limites máximos possíveis para cada objetivo.

7.2.9 Critério de Parada

São relatados na literatura poucos critérios de parada para os algoritmos evolutivos. Dentre os mais interessantes, o processo evolutivo é interrompido se:

- nenhuma solução não-dominada é adicionada à curva de Pareto para um número especificado de gerações (BABU et al., 2005). Segundo os autores, o uso dessa abordagem limita em muito sua aplicabilidade, pois se um problema apresentar ótimos locais, tal idéia terá muita chance de fracassar durante o processo evolutivo;
- nenhuma melhoria, com relação à taxa de convergência, é conseguida. Tal critério, proposto por Silva (2004), consiste na determinação de taxas de convergência, avaliadas a partir da melhor solução de cada função objetivo. As taxas de convergência

são calculadas sobre um número fixo de gerações como, por exemplo, as 100 últimas gerações, começando a partir de uma determinada geração, por exemplo, a 200ª geração. Dessa maneira, o processo de otimização é interrompido quando nenhum progresso significativo é obtido em nenhuma das funções objetivo. Uma taxa mínima é usada como referência de baixa convergência dos resultados. Ainda segundo o autor, a determinação dos parâmetros usados nesse critério deve ser feita a partir de uma monitoração criteriosa da evolução do algoritmo, a fim de garantir um número suficiente de gerações para a convergência. Estes parâmetros dependem da complexidade do problema e devem ser determinados para cada situação em particular.

- o número máximo de gerações pré-definido pelo usuário é alcançado. Esse é sem sombra de dúvida o critério de parada mais utilizado na interrupção dos algoritmos evolutivos (CASTRO, 2001; DEB, 2001; ABBASS et al., 2001; COELHO, 2003; COELHO, 2004; BABU et al., 2005).

Neste trabalho foi utilizado o terceiro critério de parada, ou seja, número máximo de gerações.

7.2.10 Tratamento de Problemas com Variáveis Inteiras e Discretas

Naturalmente espera-se que as variáveis de projeto sejam quantidades contínuas em problemas de otimização não linear. Em muitos casos, essa suposição é válida. Contudo, muito frequentemente, as variáveis binárias, inteiras e discretas surgem durante a fase de elaboração dos projetos, como por exemplo, o número de dentes de uma engrenagem, a especificação do número de reatores em um processo químico, o número de barras em uma treliça, diâmetros de tubulações pré-definidos e encontrados no mercado, entre outros. A presença de variáveis inteiras e discretas junto com variáveis contínuas acrescenta complexidade aos problemas de otimização (KANNAN; KRAMER, 1994).

Uma das técnicas mais conhecidas e utilizadas para o tratamento de problemas com variáveis inteiras e discretas é o Método *Branch and Bound* (LAND; DOIG, 1960). Inicialmente, este foi concebido para a resolução de problemas lineares, sendo posteriormente aplicado em programação não linear (MESQUITA; KAMAT, 1987; SANDGREN, 1990). No Método *Branch and Bound* não linear, uma sequência de sub-problemas NLP com variáveis contínuas é gerado e resolvido por um *solver* correspondente. Segundo Vanderplaats (1999) e Edgar et al. (2001) a principal desvantagem dessa metodologia é que esta requer um alto número de resoluções de problemas NLP gerados durante a resolução através desta abordagem.

Socha (2004) propôs uma abordagem simples para o tratamento de variáveis discretas. Esta faz uso de uma função de distribuição de probabilidade para geração de um novo conjunto de variáveis de projeto que podem ser implementadas como variáveis discretas. Uma outra estratégia consiste apenas de uma modificação simples e direta no algoritmo: o conjunto de variáveis de projeto é arredondado para o valor do inteiro mais próximo. Tal estratégia, embora bastante simples, mostrou-se eficiente quando testada em uma gama de problemas (VENTER; SOBIESKI, 2002; VIANA; STEFFEN JR, 2006), apesar da possibilidade da obtenção de valores sub-ótimos, como destacados por Kannan e Kramer (1994) e Vanderplaats (1999).

Gisvold e Moe (1972) desenvolveram uma abordagem para problemas de programação não linear mista inteira através de funções de penalidade. Neste trabalho, as variáveis discretas são tratadas como restrições explícitas e desvios nestes valores discretos são penalizados. Segundo os autores, a principal dificuldade deste método é que os parâmetros de penalidade são difíceis de serem escolhidos e a qualidade da solução obtida depende da escolha inicial desses parâmetros. Uma abordagem similar, mas que consegue lidar melhor com o problema da escolha dos parâmetros das penalidades foi proposta por Kannan e Kramer (1994). Na metodologia proposta por estes autores, a penalização é realizada a partir do uso do Método do Multiplicador de Lagrange Aumentado (VANDERPLAATS, 1999) acoplado ao Método de *Powell* e o Método do Gradiente Conjugado (Capítulo 4). Além disso, os autores definiram restrições de igualdade que penalizam violações nos valores inteiros e discretos, da seguinte maneira:

$$q_j(1 - q_j) = 0 \quad (7.10)$$

onde q_j é a variável inteira/discreta normalizada, expressa como sendo

$$q_j = \frac{x_j - s_j^l}{s_j^u - s_j^l} \quad (7.11)$$

onde s_j^u e s_j^l são os limites superior e inferior da variável j , respectivamente.

Com o intuito de aproveitar a estrutura do algoritmo MODE e a implementação de uma metodologia simples, este trabalho faz uso da estratégia descrita a seguir.

✓ Tratamento de Problemas com Variáveis de Projeto Inteiras e Discretas

Como mencionado anteriormente, a estratégia que será implementada neste trabalho, consiste em simplesmente gerar o candidato (filho) do modo como citado na Seção 7.2.4 e, a partir daí, efetuar possíveis correções em seu valor de acordo com o exemplo apresentado a seguir:

- Seja um problema com uma única variável de projeto x_{disc} que assume apenas valores discretos ($x_{disc} \in \{1,24 \ 2,45 \ 3,09 \ 4,67 \ 5,87\}$). Inicialmente define-se como domínio da variável de projeto um intervalo qualquer, por exemplo $0 \leq x_{cont} \leq 1$. Dessa maneira, como se tem 5 valores discretos, por exemplo, pode-se escrever o seguinte critério de escolha:

$$x_{disc} = \begin{cases} 1,24 & \text{se } 0,0 \leq x_{cont} < 0,2 \\ 2,45 & \text{se } 0,2 \leq x_{cont} < 0,4 \\ 3,09 & \text{se } 0,4 \leq x_{cont} < 0,6 \\ 4,67 & \text{se } 0,6 \leq x_{cont} < 0,8 \\ 5,87 & \text{se } 0,8 \leq x_{cont} \leq 1,0 \end{cases} \quad (7.12)$$

Assim, se um filho tiver valor contínuo arbitrado como sendo 0,564, esse valor é corrigido para 3,09 (que seria o domínio discreto correspondente para essa variável contínua).

- O mesmo raciocínio pode ser utilizado para variáveis inteiras e binárias.

7.2.11 A Complexidade do Algoritmo MODE

Um ponto interessante do desenvolvimento do algoritmo é a caracterização de sua complexidade. Tal característica permite a comparação de algoritmos sem a necessidade de executá-los. Dessa maneira, os operadores do MODE apresentam as seguintes ordens de complexidades para M objetivos e N indivíduos na população:

- classificação da população ($\mathcal{O}(N)$);
- geração de vizinhos ($\mathcal{O}(N)$);
- ordenamento por *rank* ($\mathcal{O}(MN^2)$) (DEB et al., 2000);
- operador de truncamento das soluções ($\mathcal{O}(MN \log N)$) (DEB et al., 2000).

A complexidade do algoritmo MODE é $\mathcal{O}(MN^2)$, onde $N=(1+\lambda)N$ (λ é o número de candidatos gerados), e é governado, assim como o NSGA II, pelo operador de ordenamento por *rank*. Além disso, observa-se que a complexidade total do algoritmo NSGA-II também é $\mathcal{O}(MN^2)$. Entretanto, no NSGA-II N é igual a $2N$. Conseqüentemente para $\lambda > 1$, o MODE requer um maior tempo de processamento em cada geração, comparado ao NSGA-II.

7.3 Aspectos Gerais de Implementação e Estruturação do MODE

A primeira versão do algoritmo MODE foi implementada em ambiente MATLAB®. Entretanto, como se sabe, a resolução de problemas de otimização sujeito a restrições algébrico-diferenciais exige uma plataforma propícia para que se obtenha redução significativa do tempo de processamento. Sabe-se que as plataformas FORTRAN® e C são as mais indicadas para essa finalidade. Desta forma, uma versão do MODE em ambiente C também foi implementada.

A Figura 7.5 apresenta a tela de apresentação do MODE.

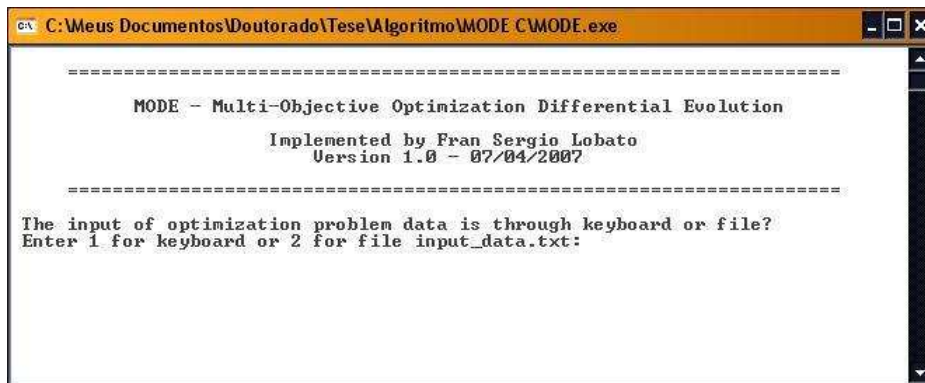


Figura 7.5: Tela de apresentação do algoritmo MODE.

7.3.1 Arquivos de Entrada e Saída do Algoritmo MODE

A Figura 7.6 apresenta os arquivos de entrada e saída do MODE. O conteúdo de cada arquivo é detalhado a seguir:

ENTRADA

- Se a opção de entrada do usuário é via arquivo `input_data.txt`, este deve ser estruturado conforme ilustrado no Apêndice A, caso contrário, o usuário deve informar os parâmetros do problema via teclado;
- `func.h`: arquivo `.h` que apresenta os objetivos e as restrições do problema;
- `metric.h`: arquivo responsável pelo cálculo das métricas de convergência e de diversidade;
- `solana.h`: caso exista solução analítica, essa deve ser informada neste arquivo;

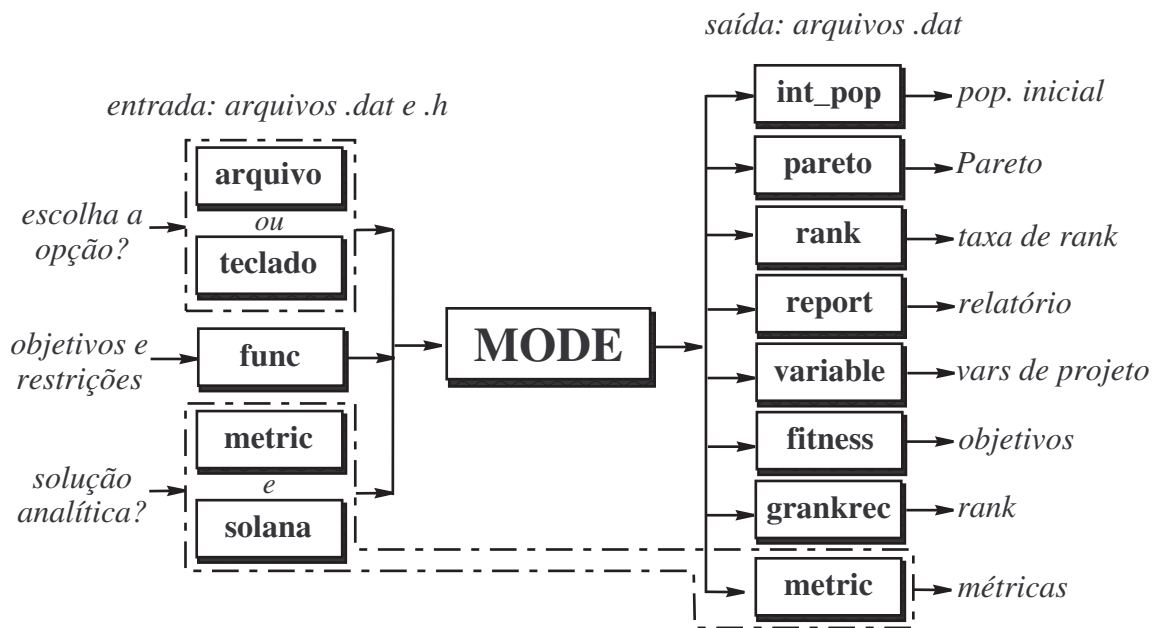


Figura 7.6: Arquivos de entrada e saída do algoritmo MODE.

SAÍDA

- **int_pop.dat**: apresenta a população inicial que é gerada randomicamente;
- **pareto.dat**: apresenta os valores dos objetivos na última geração;
- **rank.dat**: mostra a taxa de *ranking* dos indivíduos em cada geração. A taxa de *ranking* é definida como sendo a relação entre o *ranking* do indivíduo na geração i sobre a soma do *ranking* do indivíduo na geração $i - 1$ com o *ranking* do indivíduo na geração i (DEB, 2001);
- **report.dat**: arquivo que apresenta todo o detalhamento do problema e dos parâmetros do algoritmo, do processo evolutivo em cada geração, a distância entre os indivíduos da população e tempo de processamento;
- **variable.dat**: mostra a população em cada geração;
- **fitness.dat**: mostra os objetivos em cada geração;
- **grankrec.dat**: apresenta o *ranking* dos indivíduos em cada geração;
- **metric.dat**: arquivo que contém o cálculo das métricas, mas que só é impresso se a solução analítica **solana.h** (ou uma solução referência) é informada pelo usuário.

No Apêndice A é apresentado um estudo de caso bastante simples e com pequeno número de gerações e de indivíduos visando ilustrar tanto o arquivo de entrada quanto os arquivos de saída do programa MODE.

7.3.2 Subrotinas do Algoritmo MODE

As subrotinas do algoritmo MODE.C são apresentadas na Fig. 7.7, sendo estas detalhadas logo abaixo:

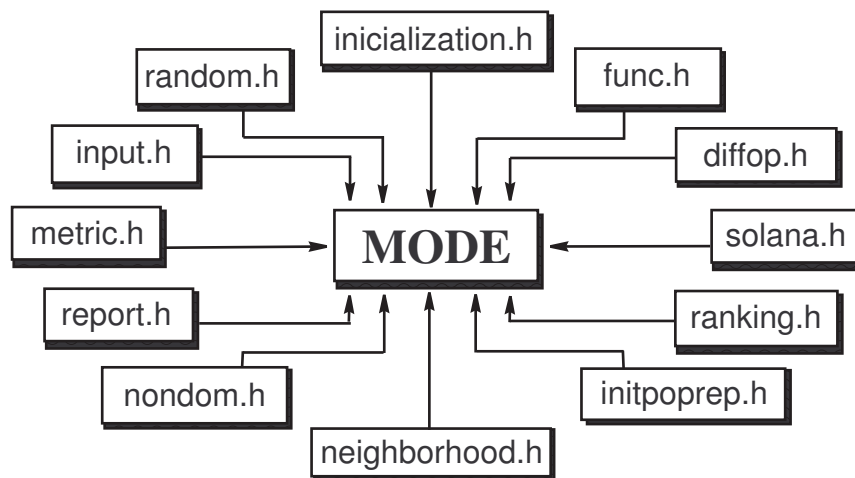


Figura 7.7: Subrotinas do algoritmo MODE.

- **input.h:** arquivo responsável pela entrada dos parâmetros do algoritmo, das características do problema multi-objetivo, da avaliação de erros de entrada. Cabe destacar que a entrada pode ser via arquivo de texto ou via teclado;
- **random.h:** arquivo responsável pela geração de um número randômico entre 0 e 1;
- **initialization.h:** inicialização randômica do processo evolutivo;
- **func.h:** função que lista os objetivos e restrições de desigualdade, se estas existirem;
- **diffop.h:** operador para geração de candidatos em potencial;
- **nondom.h:** operador que classifica os candidatos segundo o critério de dominância, ordenamento por *ranking* (**ranking.h**) e truncamento das soluções;
- **neighborhood.h:** gerador de vizinhos relacionados aos candidatos em potencial;
- **metric.h:** arquivo que calcula as métricas do problema. A solução analítica (ou uma solução referência) usada nessa comparação é apresentada em **solana.h**;

- `initpopprep.h`: arquivo que grava a população inicial;
- `report.h`: arquivo que gera um relatório detalhado de todo o processo evolutivo. Este contém os parâmetros do algoritmo, do problema analisado, das variáveis de projeto, objetivos, *ranking*, número de avaliações da função objetivo, métricas de desempenho (caso o problema tenha solução analítica) e tempo de processamento.

7.3.3 Estratégias de Erros

Com o intuito de evitar erros premeditados ou não por parte do usuário durante a entrada de parâmetros do MODE, foi adicionada uma estrutura de testes ao algoritmo. Estes testes previnem a entrada de parâmetros que não sejam adequados ou que estejam fora dos limites recomendados pela literatura.

Os testes são realizados independentemente se a entrada é via teclado ou arquivo de texto. A Figura 7.8 apresenta a estrutura geral desses testes.

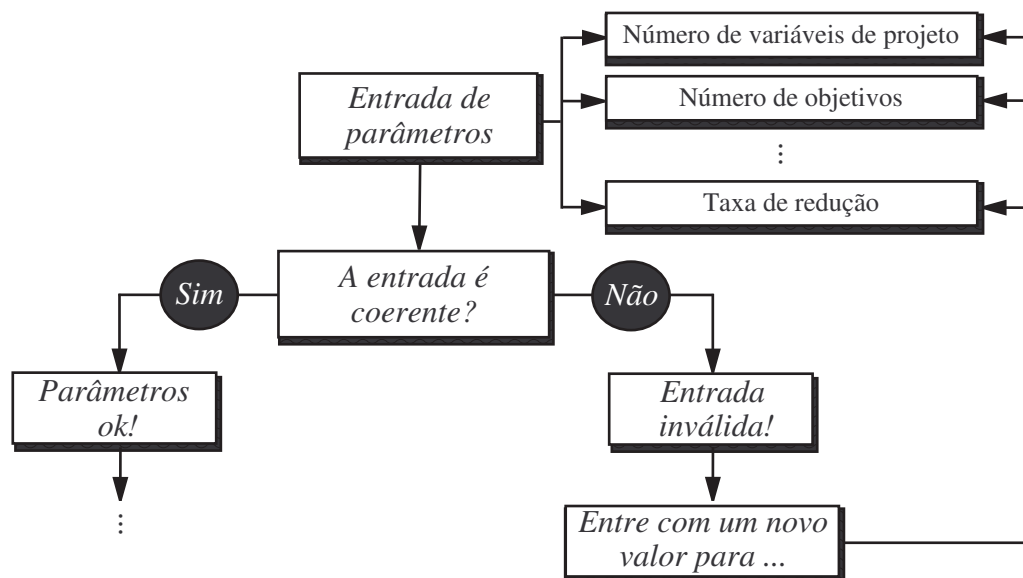


Figura 7.8: Estratégia de erros no algoritmo MODE.

A transcrição dessa estrutura de erros para a linguagem C usando como exemplo a entrada do parâmetro número de variáveis de projeto é descrita a seguir:

```

printf(" Give number of real variables:\n\n");
printf(" ");
scanf("%d",&Nvar);
while ((Nvar <= 0) || (Nvar > maxvar))
{

```

```

printf(" \n Error: The number of the variables should be positive integer and smaller
than %d.\n\n",maxvar);
printf(" Give number of real variables:\n\n");
printf(" ");
scanf("%d",&Nvar);
}

```

No bloco de comandos acima, o usuário entra com o número de variáveis **Nvar** e o laço **while** faz o teste: enquanto **Nvar** for negativo ou zero (≤ 0), (diferente de **%d**) ou maior que **maxvar** (**Nvar** > **maxvar**) é exigido que o usuário entre com um novo valor para **Nvar**. Esse processo é repetido até que um valor coerente para esse parâmetro seja especificado.

Outro exemplo interessante é o que considera a entrada dos limites inferiores e superiores das variáveis de projeto:

```

for(i = 0; i < Nvar; i++) {
    printf(" \n");
    printf(" Give lower & upper limits of the %dth real-coded variable:\n\n",i+1);
    printf(" ");
    scanf(" %f %f",&limit[i][0],&limit[i][1]);
    while(limit[i][1] < limit[i][0])
    {
        printf(" \n Error: The lower & upper limits of the %dth are inconsistent (upper: %f
        < lower: %f).\n\n",i+1,limit[i][1],limit[i][0]);
        printf(" Give lower & upper limits of the %dth real-coded variable:\n\n",i+1);
        printf(" ");
        scanf(" %f %f",&limit[i][0],&limit[i][1]);
    }
}

```

Nesse caso, se o usuário entrar com limites inconsistentes para qualquer uma das i -ésimas variáveis de projeto ($i=1, \dots, \mathbf{Nvar}$), isto é, se o limite inferior (**limit**[i][0]) for maior que o limite superior (**limit**[i][1]), o programa exige que uma nova entrada seja feita. Este procedimento é repetido até que a consistência seja alcançada.

Os limites inferiores e superiores dos parâmetros do MODE foram definidos de acordo com a literatura (STORN; PRICE, 1995; STORN, 1999; BABU et al., 2005) e são apresentados na Tab. 7.2.

Tabela 7.2: Limites inferiores e superiores dos parâmetros do MODE.

Parâmetro	Limite inferior	Limite superior
Número de objetivos*	2	10
Número de restrições*	0	∞
Número de variáveis de projeto*	1	1000
Número de gerações	> 1	∞
Tamanho da população	≥ 5	200
Probabilidade de cruzamento	0,001	1
Taxa de perturbação	0,001	2
Número de pseudo-curvas	2	20
Taxa de redução	0,001	2

* Esses limites podem ser alterados pelo usuário.

Nos próximos capítulos serão apresentados estudo de casos matemáticos e problemas de engenharia com o objetivo de demonstrar a eficiência do algoritmo proposto.

APLICAÇÕES MATEMÁTICAS

este capítulo, o algoritmo desenvolvido é testado com problemas clássicos relatados na literatura. Essas aplicações visam garantir a confiabilidade nos resultados obtidos, já que a maioria dos estudos de caso utilizados possuem solução analítica. Tais problemas foram escolhidos por proporcionarem diferentes níveis de complexidade ao algoritmo proposto. Além disso, é realizada uma análise de sensibilidade com relação à incorporação do operador de exploração das vizinhanças e de alguns parâmetros do próprio algoritmo.

A Tabela 8.1 apresenta a nomenclatura utilizada no algoritmo MODE.

Tabela 8.1: Nomenclatura dos parâmetros utilizados no MODE.

Parâmetro	Nomenclatura
Numero de gerações	N_{gen}
Tamanho da população	N
Probabilidade de cruzamento	p_c
Probabilidade de mutação	p_m
Taxa de perturbação	F
Número de pseudo-curvas	R
Taxa de redução	r
Parâmetro de penalidade	r_p

A qualidade da solução obtida pelo MODE é mensurada através de métricas de convergência (Υ), diversidade (Δ) e distância geracional (DG) (Seção 2.5.2). Assim, os estudos de caso foram resolvidos 20 vezes para a obtenção dos valores médios ($\bar{\Upsilon}$, $\bar{\Delta}$ e $\bar{DG}$) e das variâncias (σ_{Υ}^2 , σ_{Δ}^2 e σ_{DG}^2). Para comparar a solução obtida, foram gerados 1000 pontos pertencentes a solução analítica referente a cada problema. Para garantir a

qualidade dessa comparação, é importante que se tenha uma boa quantidade de pontos da solução analítica, de tal forma a evitar resultados “enganosos” das métricas referentes ao uso de poucos pontos.

Todos os estudos de casos foram resolvidos usando um microcomputador PENTIUM IV com 3,2 GHz e 2 GB de memória RAM.

Na validação dos algoritmos propostos na literatura, os pesquisadores têm feito uso de funções-teste, que apresentam solução analítica, com diferentes níveis de complexidade, tais como o tratamento de problemas convexos e côncavos, a presença de ótimos locais, descontinuidade no espaço de objetivos e no espaço das variáveis de projeto. Neste contexto, serão abordados nesta seção problemas puramente matemáticos que apresentam uma ordem crescente de dificuldade numérica.

8.1 Funções-Teste para POMO

Função de Schaffer - SCH1

A função SCH1 foi proposta por Schaffer (1984) em sua dissertação de doutorado. Apesar de ser um problema extremamente simples, a função SCH1 é bastante utilizada não só para a validação de algoritmos (DEB, 2001; CASTRO, 2001; SILVA, 2004), mas também, devido a sua configuração de espaço com relação aos objetivos, para testar qualquer operador que se fundamente no critério de dominância.

O problema é definido como (SCHAFFER, 1984):

$$\text{SCH1} = \begin{cases} \min f_1(x) = x^2 \\ \min f_2(x) = (x - 2)^2 \end{cases} \quad (8.1)$$

com $x \in (-10, 10)$.

A solução ótima de Pareto para esse problema é obtida quando o espaço de projeto da variável de projeto x pertence ao intervalo $(0, 2)$.

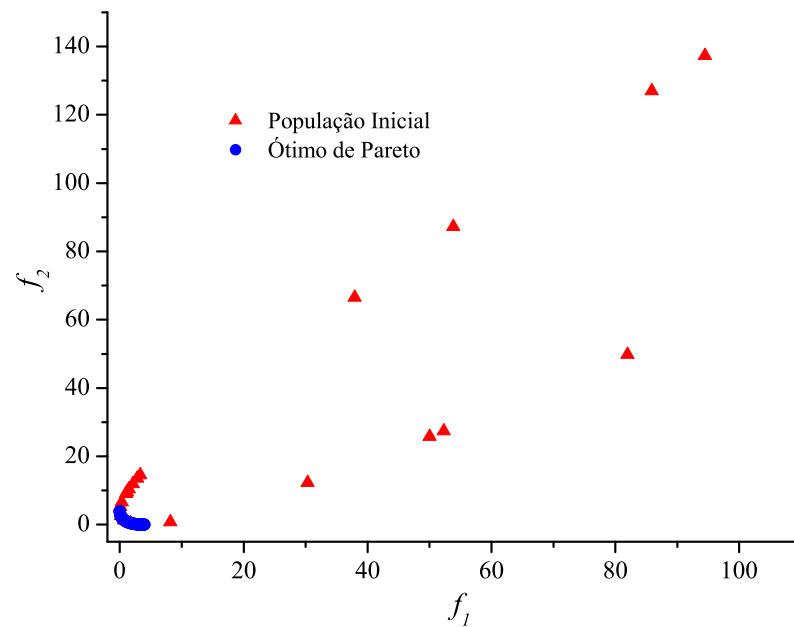
Os parâmetros utilizados por alguns algoritmos e pelo MODE são apresentados na Tab. 8.2. Os resultados obtidos para as métricas de desempenho são apresentados na Tab. 8.3. A população inicial e a solução obtida pelo MODE são apresentados nas Figs. 8.1 e 8.2, respectivamente. O resultado obtido pelo algoritmo MODE foi alcançado com 2420 avaliações da função objetivo contra 25100 avaliações do NSGA II.

Tabela 8.2: Parâmetros utilizados na resolução da função SCH1.

Parâmetro	ALGORITMO		
	NSGA II	MOGA I	MODE
N_{gen}	250	100	60
N	100	20	20
p_c	0,85	0,85	0,85
p_m	0,05	0,05	-
F	-	-	0,50
R	-	-	10
r	-	-	0,90

Tabela 8.3: Métricas de desempenho para a função SCH1.

Métrica		NSGA II (250 Gerações)	MODE (60 Gerações)
Υ	$\overline{\Upsilon}$	0,0033	0,0017
	σ_{Υ}^2	0	0
Δ	$\overline{\Delta}$	0,4778	0,4365
	σ_{Δ}^2	0,0034	0
DG	$\overline{DG}$	-	0,0007
	σ_{DG}^2	-	0

**Figura 8.1:** População inicial para a função SCH1.

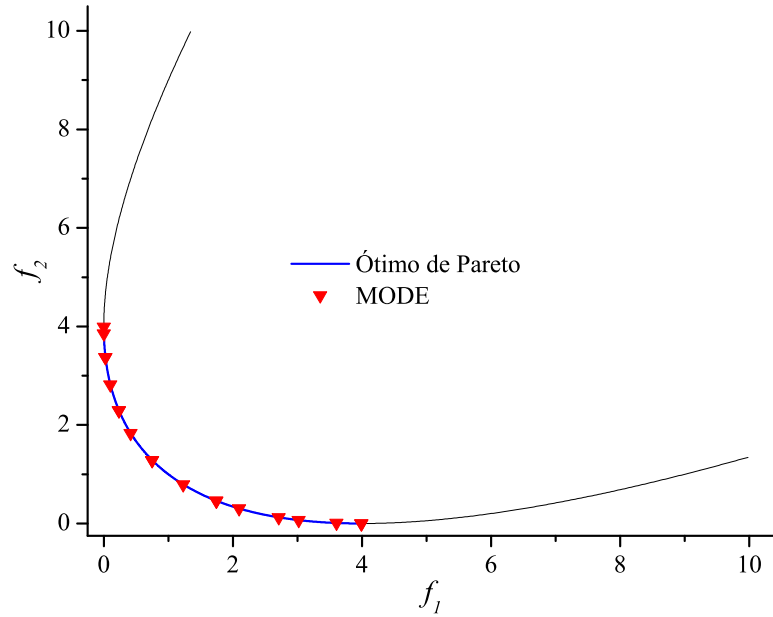


Figura 8.2: Curva de Pareto para a função SCH1.

Função de Schaffer - SCH2

A função SCH2 também foi proposta por Schaffer (1984). Esse problema apresenta como principal característica a descontinuidade no espaço de objetivos.

SCH2 é definida como sendo:

$$\text{SCH2} = \begin{cases} \min f_1(x) = \begin{cases} -x & \text{se } x \leq 1 \\ x - 2 & \text{se } 1 < x \leq 3 \\ 4 - x & \text{se } 3 < x \leq 4 \\ x - 4 & \text{se } x > 4 \end{cases} \\ \min f_2(x) = (x - 5)^2 \end{cases} \quad (8.2)$$

com $x \in (-5, 10)$.

O conjunto ótimo de Pareto consiste de duas regiões: $x \in \{(1, 2) \cup (4, 5)\}$. Os parâmetros utilizados por alguns algoritmos e pelo MODE são apresentados na Tab. 8.4. Os resultados obtidos para as métricas de desempenho são apresentados na Tab. 8.5 com 10050 avaliações da função objetivo usando o MODE e 25100 avaliações com o NSGA II.

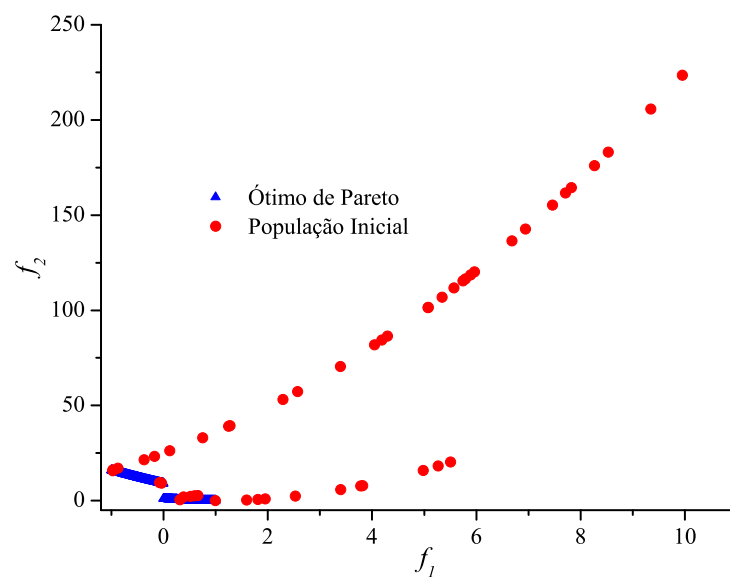
Tabela 8.4: Parâmetros utilizados na resolução da função SCH2.

Parâmetro	ALGORITMO		
	NSGA II	PMOGA	MODE
N_{gen}	250	100	100
N	100	50	50
p_c	0,85	0,90	0,85
p_m	0,05	0,05	-
F	-	-	0,50
R	-	-	10
r	-	-	0,90

Tabela 8.5: Métricas de desempenho para a função SCH2.

Métrica		NSGA II (250 Gerações)	MODE (100 Gerações)
Υ	$\overline{\Upsilon}$	0,0019	0,0020
	σ_{Υ}^2	-	0
Δ	$\overline{\Delta}$	0,0003	0,0003
	σ_{Δ}^2	-	0
DG	$\overline{DG}$	0,3098	0,3148
	σ_{DG}^2	-	0,0009

Nas Figuras 8.3 e 8.4 são apresentados os perfis da população inicial e os resultados obtidos para a função SCH2, respectivamente.

**Figura 8.3:** População inicial para a função SCH2.

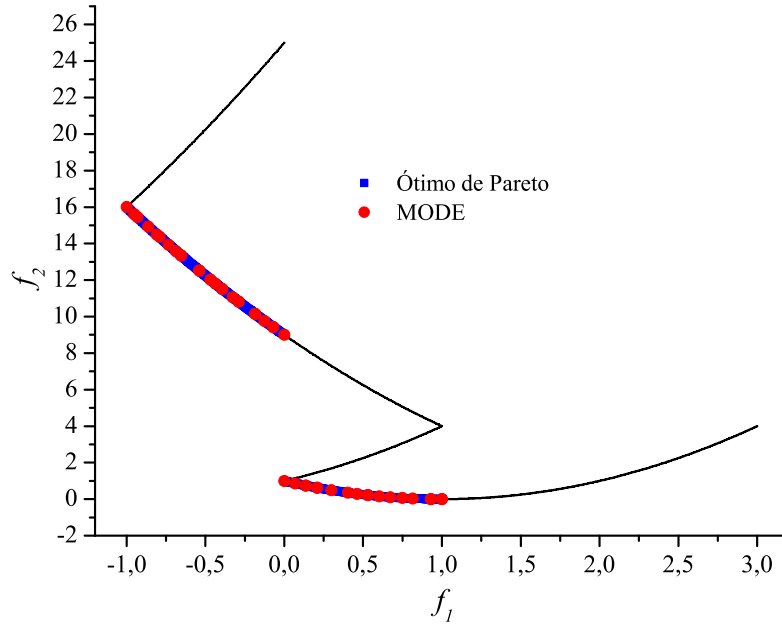


Figura 8.4: Curva de Pareto para a função SCH2.

Função FON

Fonseca e Fleming (1995) propuseram um problema com dois objetivos com n variáveis:

$$\text{FON} = \begin{cases} \min f_1(x_1, x_2, \dots, x_n) = 1 - \exp \left(- \sum_{i=1}^n \left(x_i - \frac{1}{\sqrt{n}} \right)^2 \right) \\ \min f_2(x_1, x_2, \dots, x_n) = 1 - \exp \left(- \sum_{i=1}^n \left(x_i + \frac{1}{\sqrt{n}} \right)^2 \right) \end{cases} \quad (8.3)$$

com $x_i \in (-4, 4)$ ($i = 1, 2, \dots, n$).

O conjunto ótimo de Pareto para esse problema é $x \in (-1/\sqrt{n}, 1/\sqrt{n})$, para $i = 1, 2, \dots, n$. Um aspecto interessante desse caso é que o espaço de busca dos objetivos não depende do valor de n .

Os parâmetros utilizados no MODE são apresentados na Tab. 8.6. As métricas de desempenho são mostradas na Tab. 8.7 para n igual a 3.

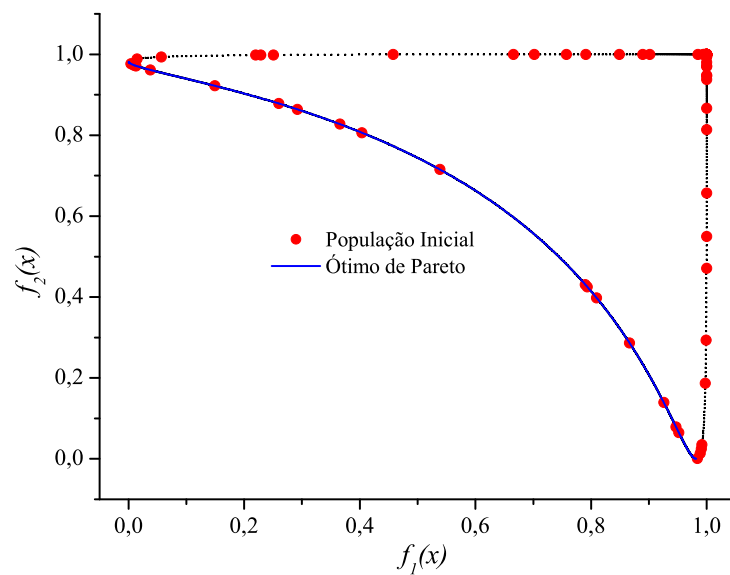
Tabela 8.6: Parâmetros utilizados na resolução da função FON.

Parâmetro	ALGORITMO	
	NSGA II	MODE
N_{gen}	250	100
N	100	100
p_c	0,85	0,85
p_m	0,05	-
F	-	0,50
R	-	10
r	-	0,90

Tabela 8.7: Métricas de desempenho para a função FON.

Métrica		NSGA II (250 Gerações)	MODE (100 Gerações)
Υ	$\overline{\Upsilon}$	0,00018	0,00019
	σ_{Υ}^2	0	0
Δ	$\overline{\Delta}$	0,2045	0,2577
	σ_{Δ}^2	0,00036	0,00022
DG	$\overline{DG}$	-	0
	σ_{DG}^2	-	0

A população inicial e os resultados obtidos pelo MODE são mostrados nas Figs. 8.5 e 8.6, respectivamente.

**Figura 8.5:** População inicial para a função FON.

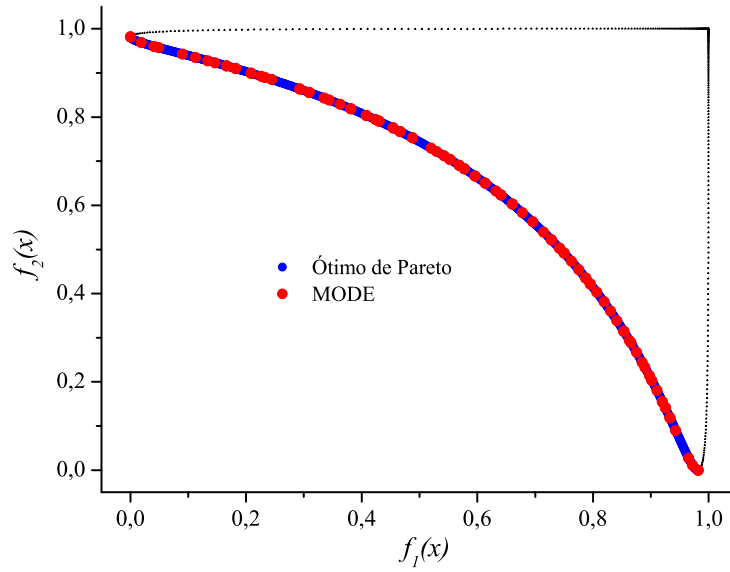


Figura 8.6: Curva de Pareto para a função FON.

Foram realizadas 20100 avaliações da função objetivo usando o MODE e 25100 avaliações com o NSGA II.

Função KUR

Kursawe (1990) propôs um problema que apresenta regiões descontínuas:

$$\text{KUR} = \begin{cases} \min f_1(x_1, x_2, x_3) = \sum_{i=1}^2 \left(-10 \exp \left(-0,2 \sqrt{x_i^2 + x_{i+1}^2} \right) \right) \\ \min f_2(x_1, x_2, x_3) = \sum_{i=1}^3 \left(|x_i|^{0,8} + 5 \sin(x_i^3) \right) \end{cases} \quad (8.4)$$

com $x_i \in (-5, 5)$ ($i = 1, 2, 3$).

O conjunto ótimo de Pareto é formado por quatro regiões distintas: um único ponto que corresponde a $x_i = 0$ ($i = 1, 2, 3$), a primeira região descontínua ($x_1 = x_2 = 0$) e outras duas regiões desconectadas ($x_1 = x_3 = 0$). Os parâmetros do algoritmo são apresentados na Tab. 8.8. Já as métricas de desempenho são mostradas na Tab. 8.9.

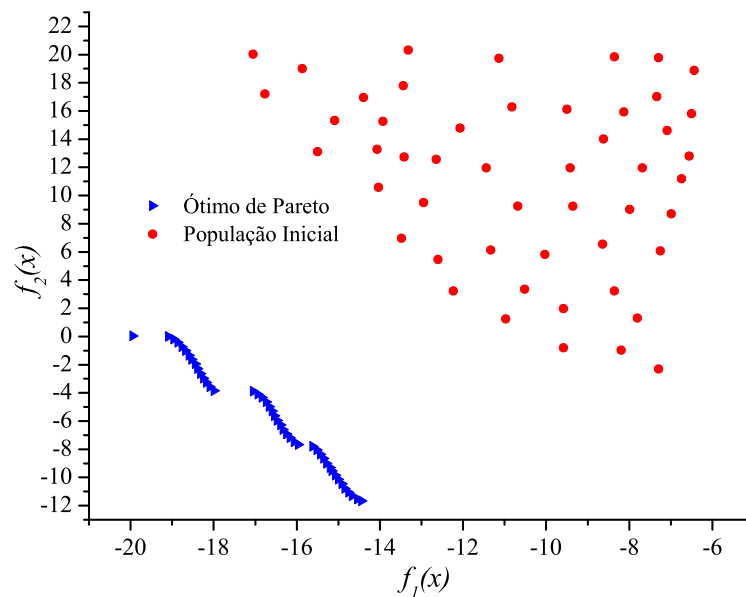
Nas Figuras 8.7 e 8.8 são apresentados os perfis da população inicial e da solução encontrada pelo MODE, respectivamente.

Tabela 8.8: Parâmetros utilizados na resolução da função KUR.

Parâmetro	ALGORITMO		
	NSGA II	SPEA	MODE
N_{gen}	250	250	150
N	100	100	50
p_c	0,85	0,80	0,85
p_m	0,05	0,10	-
F	-	-	0,50
R	-	-	10
r	-	-	0,90

Tabela 8.9: Métricas de desempenho para a função KUR.

Métrica		NSGA II (250 Gerações)	MODE (150 Gerações)
Υ	$\overline{\Upsilon}$	0,0289	0,0341
	$\sigma_{\overline{\Upsilon}}^2$	0	0
Δ	$\overline{\Delta}$	0,4114	0,2245
	$\sigma_{\overline{\Delta}}^2$	0,0009	0,0077
DG	$\overline{DG}$	-	0,0032
	$\sigma_{\overline{DG}}^2$	-	0,0001

**Figura 8.7:** População inicial para a função KUR.

Deve-se destacar a Fig. 8.9, que apresenta a solução ótima das variáveis de projeto. Essa solução no espaço de projeto é constituída por regiões descontínuas.

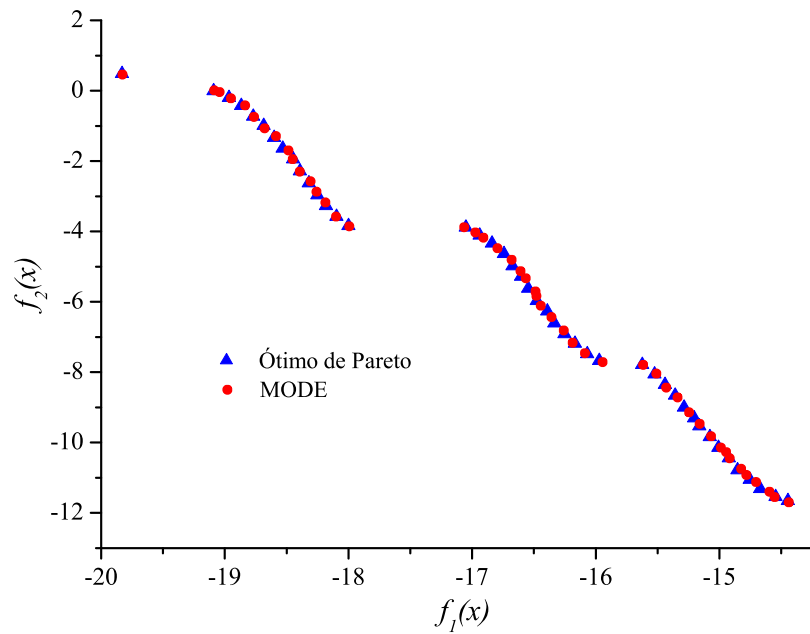


Figura 8.8: Curva de Pareto para a função KUR.

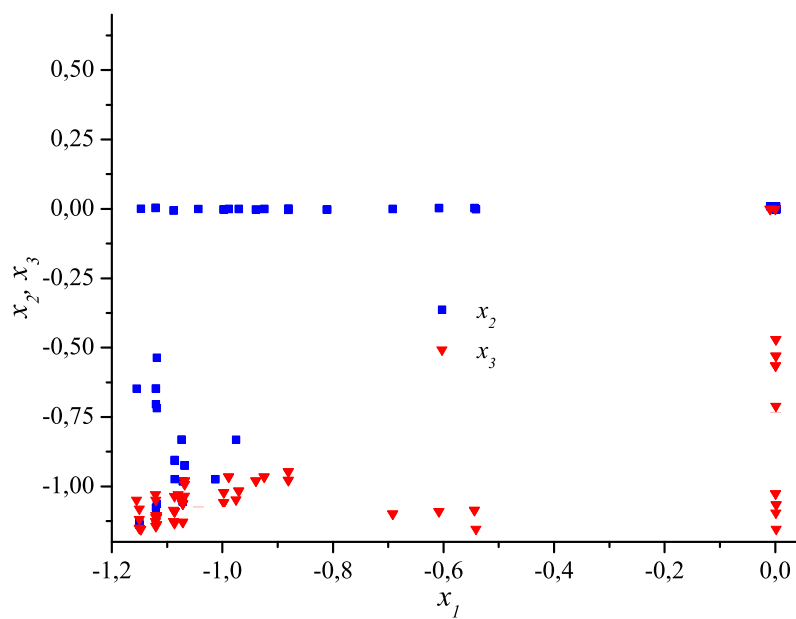


Figura 8.9: Espaço de variáveis de projeto para a função KUR.

Na função KUR o algoritmo MODE realizou 15050 avaliações, contra 25100 avaliações obtidas pelo algoritmo NSGA II.

Função GTP

Seja o seguinte problema (DEB, 2001):

$$\text{GTP} = \begin{cases} \min f_1(x_1, x_2, \dots, x_{30}) = x_1 \\ \min f_2(x_1, x_2, \dots, x_{30}) = g(x) \left(1 - \sqrt{\frac{x_1}{g(x)}} \right) \\ g(x) = 2 + \sum_{i=2}^{30} \frac{x_i^2}{4000} - \prod_{i=2}^{30} \cos \left(\frac{x_i}{\sqrt{i}} \right) \end{cases} \quad (8.5)$$

com $x_1 \in (0, 1)$ e $x_i \in (-5, 12; 5, 12)$ ($i = 2, 3, \dots, 30$).

A principal característica desse problema é a presença de inúmeros ótimos locais. O conjunto ótimo de Pareto para esse problema é $x_1 \in (0, 1)$ e $x_i = 0$ ($i = 2, 3, \dots, 30$).

A Tabela 8.10 traz os parâmetros utilizados pelos algoritmos evolutivos. Os resultados obtidos pelas métricas de desempenho são mostrados na Tab. 8.11.

Tabela 8.10: Parâmetros utilizados na resolução da função GTP.

Parâmetro	ALGORITMO		
	NSGA II	SPEA II	MODE
N_{gen}	250	250	100
N	100	100	100
p_c	0,85	0,85	0,85
p_m	0,05	0,05	-
F	-	-	0,50
R	-	-	10
r	-	-	0,90

Tabela 8.11: Métricas de desempenho para a função GTP.

Métrica		NSGA II	MODE
		(200 Gerações)	(100 Gerações)
Υ	$\overline{\Upsilon}$	0,0297	0,0160
	σ_{Υ}^2	0,0101	0,0004
Δ	$\overline{\Delta}$	-	0,0002
	σ_{Δ}^2	-	0
DG	$\overline{DG}$	-	0,3315
	σ_{DG}^2	-	0,0008

Os perfis da população inicial e o conjunto ótimo de Pareto obtido são apresentados nas Figs. 8.10 e 8.11, respectivamente. Na função GTP o algoritmo MODE avaliou 20100 vezes os objetivos, contra 25100 avaliações obtidas pelo algoritmo NSGA II.

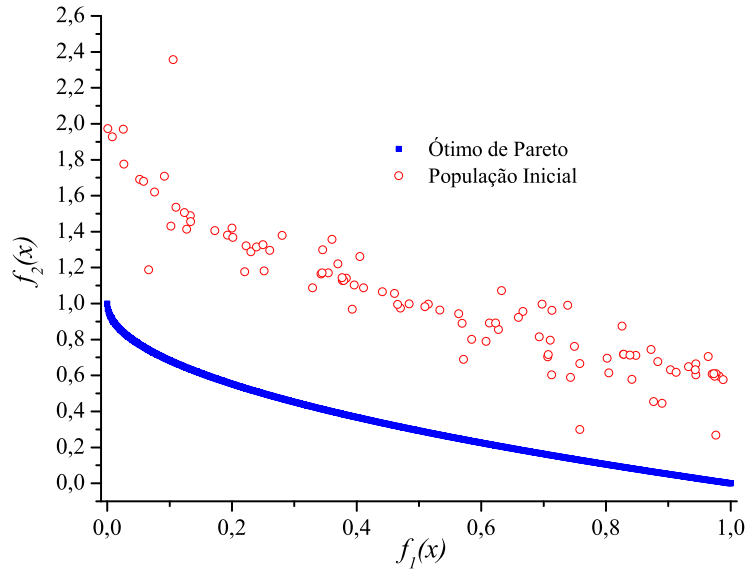


Figura 8.10: População inicial para a função GTP.

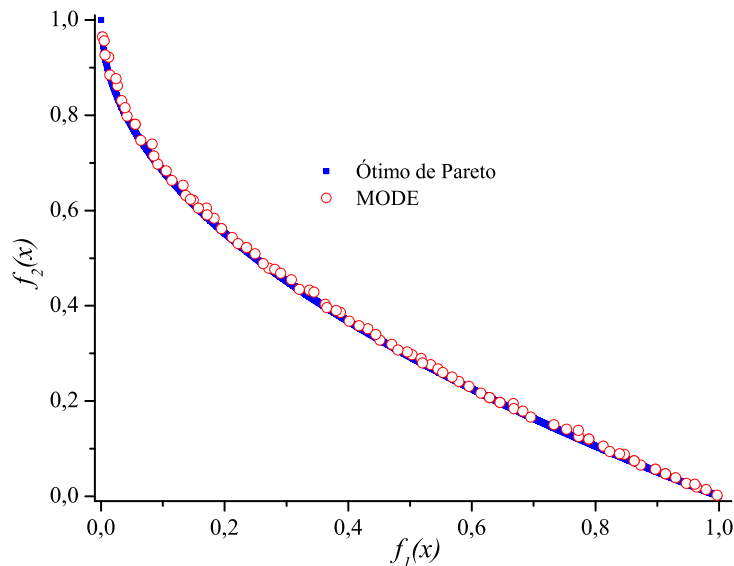


Figura 8.11: Curva de Pareto para a função GTP.

Dentre todas as funções-teste apresentadas na literatura, sem dúvida as desenvolvidas por Zitzler et al. (2000) (ZDT1, ZDT2, ZDT3, ZDT4, ZDT5 e ZDT6) são as mais empregadas para a validação dos algoritmos propostos. Isto se deve às dificuldades referentes à convergência e diversidade destas funções-teste. Suas principais características são:

convexidade, não-convexidade, fronteiras discretas ou descontínuas, multi-modalidade, problemas enganosos com espaços de busca tendenciosos.

Cada uma das funções-teste é constituída por um problema de minimização com dois objetivos, estruturadas como segue:

$$\text{ZDT} = \begin{cases} \min f_1(x_1, x_2, \dots, x_m) = x_1 \\ \min f_2(x_1, x_2, \dots, x_m) = g(x)h(f_1(x), g(x)) \end{cases} \quad (8.6)$$

onde f_1 é função apenas da variável x_1 e g é uma função das $m-1$ variáveis de projeto restantes. A função h envolve f_1 e g , sendo, portanto, dependente das m variáveis do problema.

Estas funções-teste propostas diferem-se pelas funções f_1 , g e h , pelo número de variáveis m e pelo domínio das variáveis de projeto.

Em Zitzler et al. (2000) e Deb (2001), são apresentadas comparações de algoritmos genéticos clássicos como o NPGA, HLGA, VEGA, NSGA, SPEA, SOEA e o algoritmo RAND, que faz uso de estratégia puramente randômica. De forma geral, como será visto posteriormente, os resultados obtidos por esses algoritmos mostram que a incorporação de estratégias evolutivas mais elaboradas produzem melhores resultados. Por exemplo, segundo Zitzler et al. (2000) e Deb (2001), o operador de elitismo é um fator importante no processo de otimização evolutiva multi-objetivos. O algoritmo SPEA, que dentre os apresentados é o único que incorpora esse operador, supera todos os demais com relação a essas funções-teste. Por outro lado, segundo os mesmos autores, o desempenho dos demais algoritmos é melhorado quando a estratégia de elitismo presente no SPEA foi a eles incorporada.

Os parâmetros utilizados em todos os algoritmos citados anteriormente são apresentados na Tab. 8.12 (ZITZLER et al., 2000; DEB, 2001).

Tabela 8.12: Parâmetros utilizados nos algoritmos clássicos.

Parâmetro	Valor	ALGORITMO
Numero de gerações	250	todos
Tamanho da população	100	todos
Probabilidade de cruzamento	0,8	todos
Probabilidade de mutação	0,01	todos
Parâmetro de Nicho (σ_{share})	0,48862	NSGA, MOGA, NPGA
Pressão de dominância (t_{dom})	10	NGPA

Função ZDT1

A função ZDT1 apresenta fronteira ótima de Pareto convexa. Além disso, tem distribuição uniforme das soluções ao longo da curva. Tal função é definida como:

$$\text{ZDT1} = \begin{cases} f_1(x_1) = x_1 \\ g(x_2, \dots, x_m) = 1 + \frac{9}{m-1} \sum_{i=2}^m x_i \\ h(f_1(x_1), g(x_2, \dots, x_m)) = 1 - \sqrt{f_1(x_1)/g(x_2, \dots, x_m)} \end{cases} \quad (8.7)$$

Todas as variáveis são definidas no intervalo (0,1). A região ótima de Pareto corresponde a $x_1 \in (0,1)$ e $x_i = 0$ ($i = 2, \dots, m = 30$). Segundo Deb (2001) a principal dificuldade associada com a resolução desse problema é o número de variáveis.

A Figura 8.12 apresenta os perfis obtidos pelos algoritmos clássicos citados anteriormente para o problema ZDT1, com configurações apresentadas na Tab. 8.12.

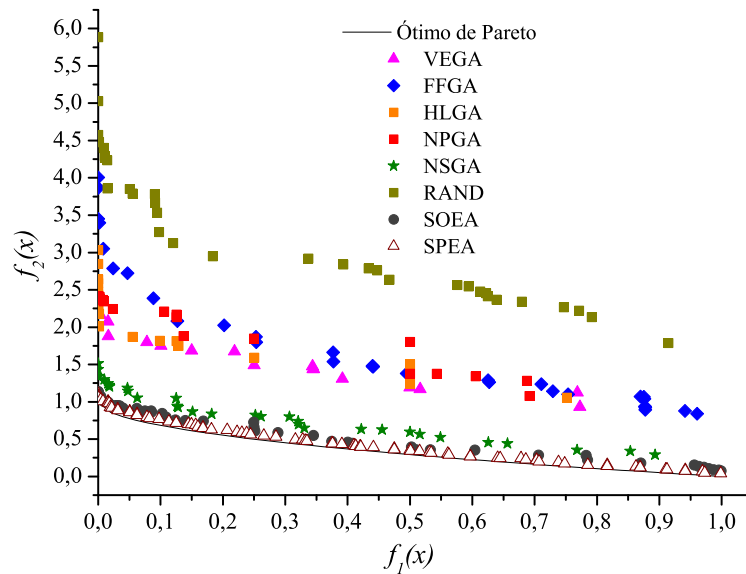


Figura 8.12: Resolução da função ZDT1 usando algoritmos clássicos.

Os parâmetros utilizados na resolução através de algoritmos evolutivos para o problema ZDT1 são listados na Tab. 8.13. Deve ser destacado o número baixo de indivíduos e o elevado número de gerações no algoritmo MOGA I.

A Tabela 8.14 compara as métricas de desempenho obtidas pelo MODE, com 30100 avaliações da função objetivo, contra 25100 avaliações do algoritmo NSGA II.

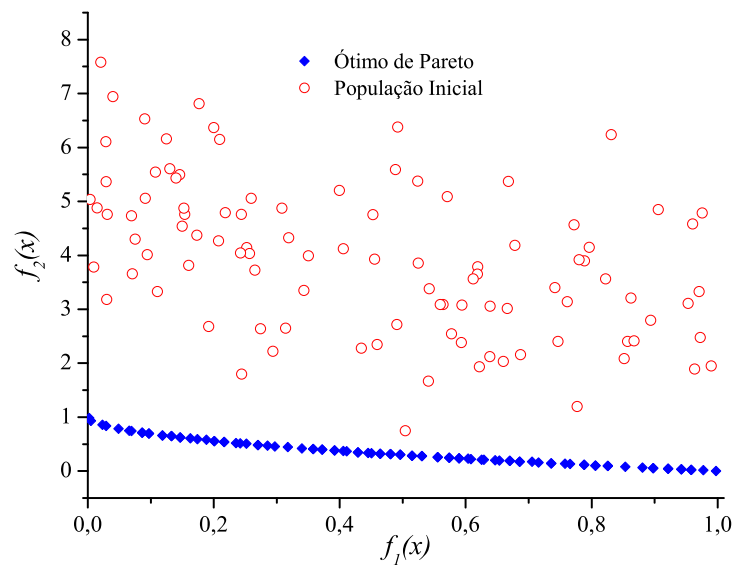
Tabela 8.13: Parâmetros utilizados na resolução da função ZDT1.

Parâmetro	ALGORITMO		
	PMOGA	MOGA I	MODE
N_{gen}	250	50000	150
N	100	20	100
p_c	0,80	0,80	0,85
p_m	0,01	0,05	-
F	-	-	0,50
R	-	-	10
r	-	-	0,90

Tabela 8.14: Métricas de desempenho para a função ZDT1.

Métrica		NSGA II (250 Gerações)	MODE (150 Gerações)
Υ	$\overline{\Upsilon}$	0,0008	0,0016
	σ_{Υ}^2	0	0
Δ	$\overline{\Delta}$	0,4632	0,1988
	σ_{Δ}^2	0,0416	0,0025
DG	$\overline{DG}$	0,0006	0,0001
	σ_{DG}^2	0	0

A Figura 8.13 mostra a população inicial gerada pelo MODE. A solução obtida pelo algoritmo proposto, comparado ao MOGA I e ao PMOGA, é apresentada na Fig. 8.14.

**Figura 8.13:** População inicial para a função ZDT1.

Verifica-se pela análise das Figs. 8.12 e 8.14 que apenas os algoritmos SPEA, PMOGA

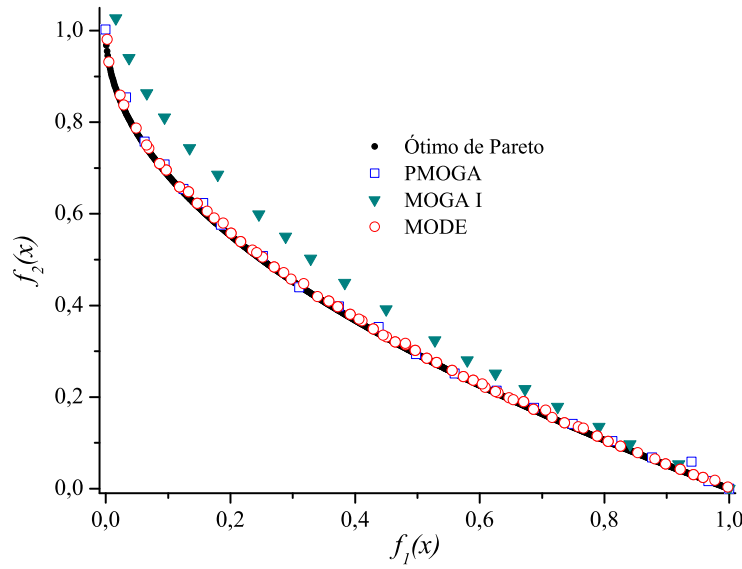


Figura 8.14: Curva de Pareto para a função ZDT1.

e MODE conseguem se aproximar do ótimo de Pareto. Já o algoritmo MOGA I, apesar do elevado número de gerações em comparação com as outras abordagens, não consegue ao final do seu processo evolutivo se aproximar satisfatoriamente da solução analítica.

Função ZDT2

A função ZDT2 apresenta fronteira ótima de Pareto não-convexa. Este é definido como:

$$\text{ZDT2} = \begin{cases} f_1(x_1) = x_1 \\ g(x_2, \dots, x_m) = 1 + \frac{9}{m-1} \sum_{i=2}^m x_i \\ h(f_1(x_1), g(x_2, \dots, x_m)) = 1 - (f_1(x_1)/g(x_2, \dots, x_m))^2 \end{cases} \quad (8.8)$$

Todas as variáveis são definidas no intervalo $(0,1)$. A região ótima de Pareto corresponde a $x_1 \in (0,1)$ e $x_i = 0$ ($i = 2, \dots, m = 30$).

A seguir é apresentado na Fig. 8.15 os perfis obtidos pelos algoritmos clássicos citados anteriormente para a configuração dada na Tab. 8.12. Os parâmetros utilizados na resolução do problema ZDT2 são listados na Tab. 8.15. Deve ser destacado no algoritmo MOGA I o baixo número de indivíduos da população e o alto número de gerações necessárias para esta abordagem.

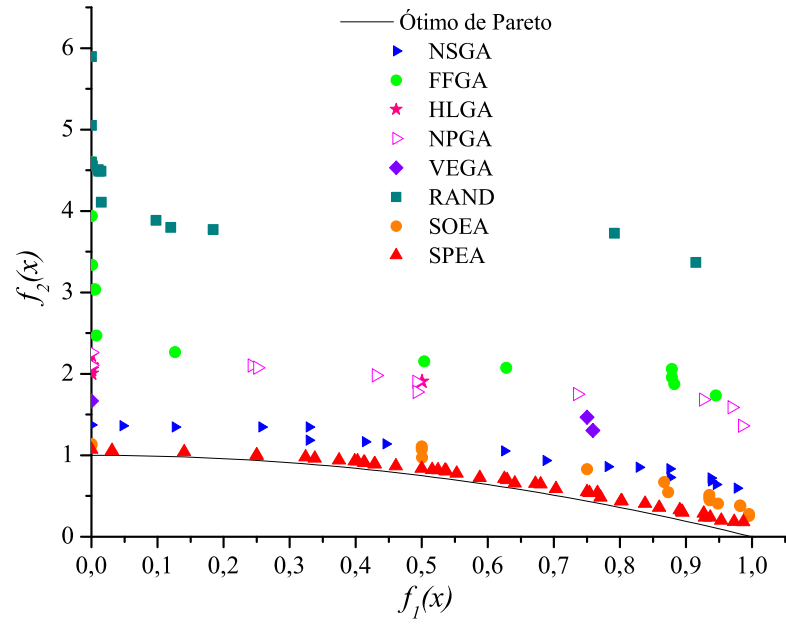


Figura 8.15: Resolução da função ZDT2 usando algoritmos clássicos.

Tabela 8.15: Parâmetros utilizados na resolução da função ZDT2.

Parâmetro	ALGORITMO		
	PMOGA	MOGA I	MODE
N_{gen}	250	50000	150
N	100	20	100
p_c	0,80	0,80	0,85
p_m	0,01	0,05	-
F	-	-	0,50
R	-	-	10
r	-	-	0,90

A Tabela 8.16 apresenta as métricas de desempenho do MODE comparadas ao do NSGA II.

Tabela 8.16: Métricas de desempenho para a função ZDT2.

Métrica		NSGA II (250 Gerações)	MODE (150 Gerações)
Υ	$\overline{\Upsilon}$	0,0008	0,0011
	σ_{Υ}^2	0	0
Δ	$\overline{\Delta}$	0,4351	0,2549
	σ_{Δ}^2	0,0246	0,0009
DG	$\overline{DG}$	0,0006	0,0001
	σ_{DG}^2	0	0

A população inicial gerada pelo MODE é mostrada na Fig. 8.16. A comparação entre MODE, o MOGA I e o PMOGA é apresentada na Fig. 8.17.

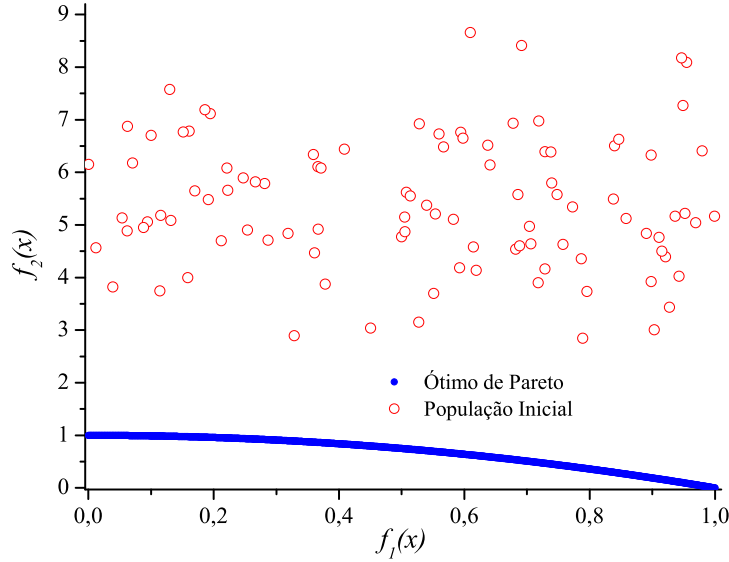


Figura 8.16: População inicial para a função ZDT2.

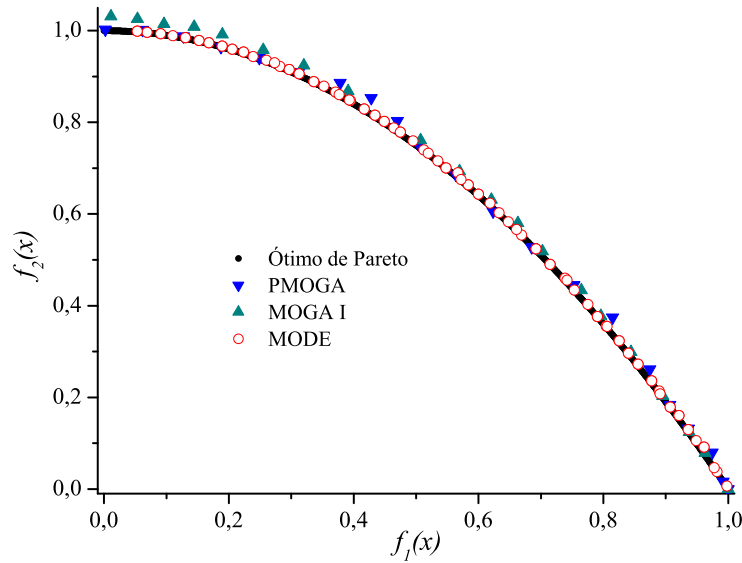


Figura 8.17: Curva de Pareto para a função ZDT2.

O algoritmo MODE avalia 30100 vezes a função objetivo, contra 25100 avaliações do algoritmo NSGA II.

A análise das Figs. 8.15 e 8.17 permite constatar que os algoritmos SPEA, PMOGA, MOGA I e MODE conseguem obter boa aproximação da solução analítica. Todavia, o algoritmo MOGA I, como apresentado na Tab. 8.15, tem bom resultado às custas de um número elevado de gerações.

Função ZDT3

A função ZDT3 apresenta fronteira ótima de Pareto formada por partes descontínuas, provocada pela presença da função *seno* em h , conforme abaixo.

Esse problema é definido como:

$$\text{ZDT3} = \begin{cases} f_1(x_1) = x_1 \\ g(x_2, \dots, x_m) = 1 + \frac{9}{m-1} \sum_{i=2}^m x_i \\ h(f_1(x_1), g(x_2, \dots, x_m)) = 1 - \sqrt{f_1(x_1)/g(x_2, \dots, x_m)} + \\ \quad -(f_1(x_1)/g(x_2, \dots, x_m)) \sin(10\pi f_1(x_1)) \end{cases} \quad (8.9)$$

Todas as variáveis são definidas no intervalo $(0,1)$. A região ótima de Pareto corresponde a $x_1 \in (0,1)$ e $x_i = 0$ ($i = 2, \dots, 30$).

A seguir são apresentados na Fig. 8.18 os perfis obtidos pelos algoritmos clássicos citados anteriormente para a configuração dada na Tab. 8.12.

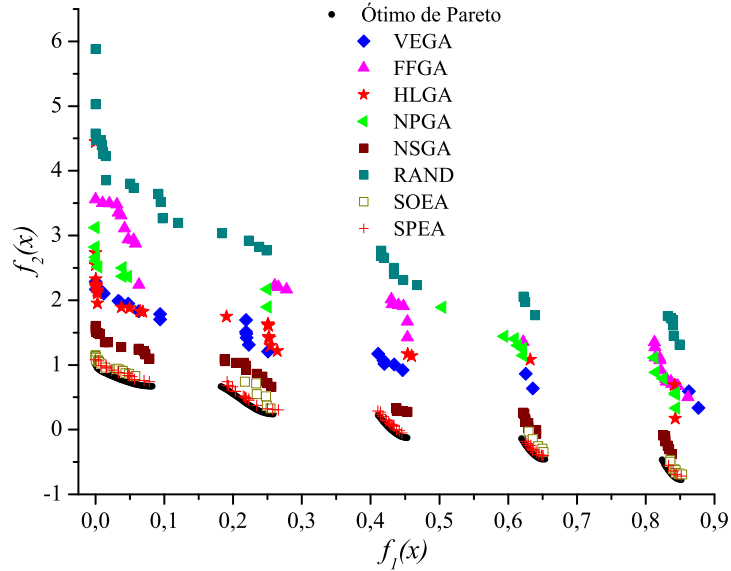


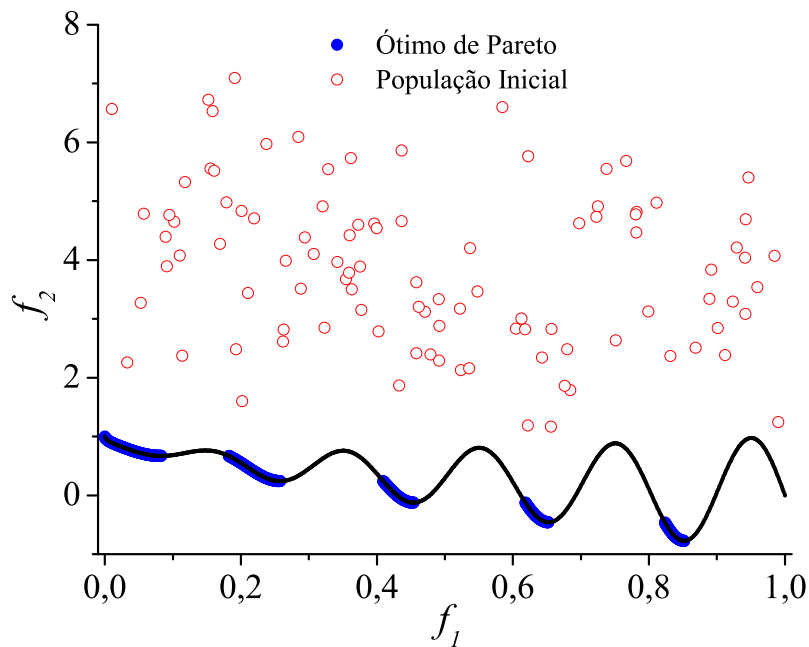
Figura 8.18: Resolução da função ZDT3 usando algoritmos clássicos.

Os parâmetros utilizados na resolução através de algoritmos evolutivos para o problema ZDT3 são listados na Tab. 8.17.

A Figura 8.19 mostra a população inicial gerada pelo MODE.

Tabela 8.17: Parâmetros utilizados na resolução da função ZDT3.

Parâmetro	ALGORITMO		
	PMOGA	MOGA I	MODE
N_{gen}	250	50000	150
N	100	20	100
p_c	0,80	0,80	0,85
p_m	0,01	0,05	-
F	-	-	0,50
R	-	-	10
r	-	-	0,90

**Figura 8.19:** População inicial para a função ZDT3.

A Figura 8.20 apresenta a comparação entre as soluções obtidas pelo MODE, MOGA I e PMOGA. Conforme pode ser observado a partir dessas figuras, dentre todos os algoritmos utilizados, somente os algoritmos SPEA, PMOGA, MOGA I e MODE conseguem obter boa aproximação do ótimo de Pareto. Os demais algoritmos não conseguem resultados satisfatórios com esse número de gerações. Além disso, deve ser destacado que os resultados obtidos pelo algoritmo MODE foram as custas de 30100 avaliações da função objetivo, contra 25100 avaliações do algoritmo NSGA II.

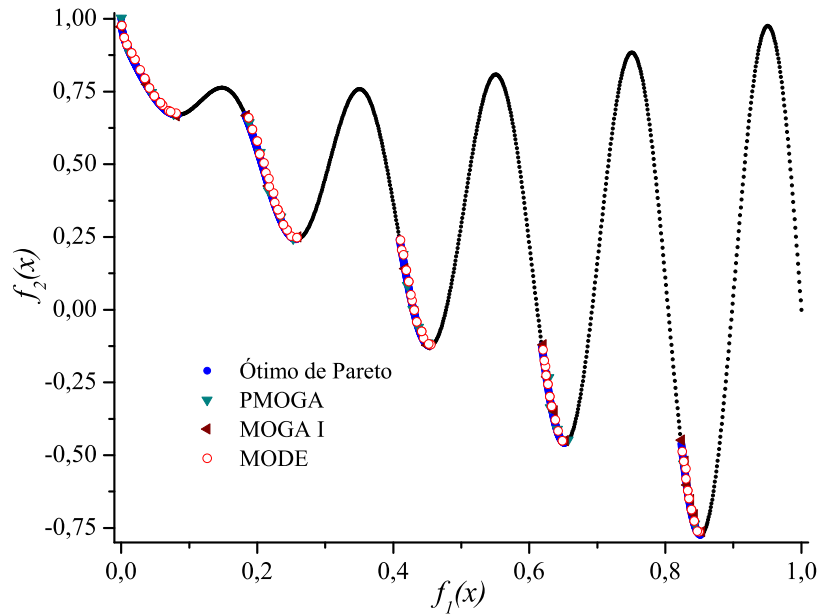


Figura 8.20: Curva de Pareto para a função ZDT3.

A Tabela 8.18 apresenta as métricas de desempenho do MODE comparadas ao algoritmo genético NSGA II.

Tabela 8.18: Métricas de desempenho para a função ZDT3.

Métrica		NSGA II (250 Gerações)	MODE (150 Gerações)
Υ	$\overline{\Upsilon}$	0,0434	0,0010
	σ_{Υ}^2	0,0000	0
Δ	$\overline{\Delta}$	0,5756	0,2881
	σ_{Δ}^2	0,0050	0,0032
DG	$\overline{DG}$	0,0005	0,0001
	σ_{DG}^2	0	0

Função ZDT4

A função ZDT4 é composto por 21⁹ ótimos locais, o que representa uma grande dificuldade na busca pelo ótimo global. ZDT4 é definido como:

$$\text{ZDT4} = \begin{cases} f_1(x_1) = x_1 \\ g(x_2, \dots, x_m) = 1 + 10(m-1) + \sum_{i=2}^m (x_i^2 - 10 \cos(4\pi x_i)) \\ h(f_1(x_1), g(x_2, \dots, x_m)) = 1 - \sqrt{f_1(x_1)/g(x_2, \dots, x_m)} \end{cases} \quad (8.10)$$

Todas as variáveis são definidas no intervalo (0,1). A região ótima de Pareto corresponde a $x_1 \in (0, 1)$ e $x_i = 0$ ($i = 2, \dots, 10$). Os parâmetros utilizados na resolução através de algoritmos evolutivos para o problema ZDT4 são listados na Tab. 8.19.

Tabela 8.19: Parâmetros utilizados na resolução da função ZDT4.

Parâmetro	ALGORITMO		
	PMOGA	MOGA I	MODE
N_{gen}	250	50000	150
N	100	20	100
p_c	0,80	0,80	0,85
p_m	0,01	0,05	-
F	-	-	0,50
R	-	-	10
r	-	-	0,90

A Figura 8.21 traz os perfis obtidos para esse caso pelos algoritmos clássicos, citados anteriormente, para a configuração dada na Tab. 8.12.

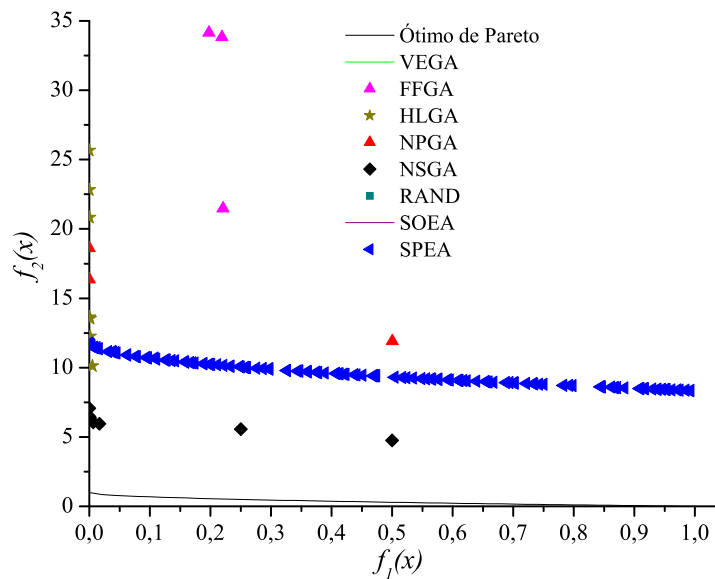


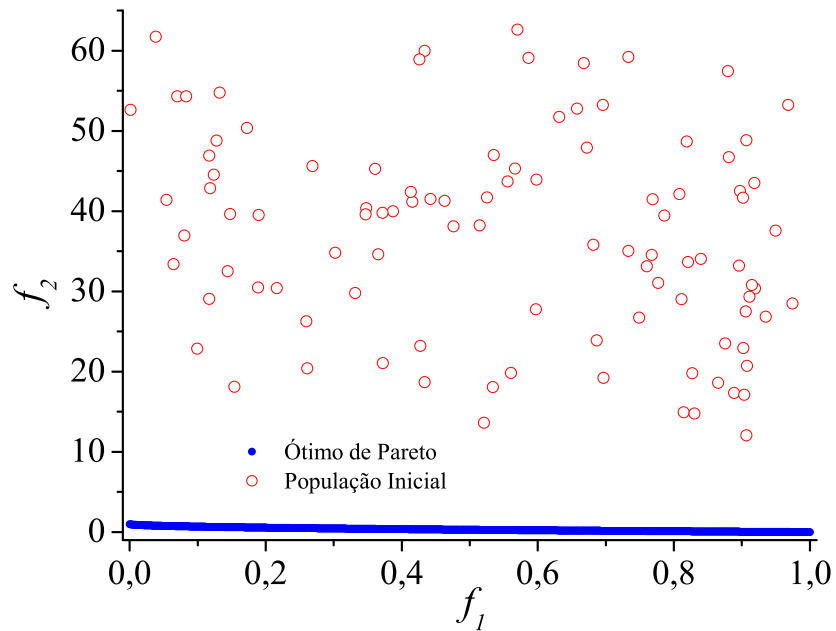
Figura 8.21: Resolução da função ZDT4 usando algoritmos clássicos.

A Tabela 8.20 apresenta as métricas de desempenho do MODE comparadas com o algoritmo genético NSGA II.

Tabela 8.20: Métricas de desempenho para a função ZDT4.

Métrica		NSGA II (250 Gerações)	MODE (150 Gerações)
Υ	$\overline{\Upsilon}$	0,0434	0,0010
	σ_{Υ}^2	0,0000	0
Δ	$\overline{\Delta}$	0,5756	0,2881
	σ_{Δ}^2	0,0050	0,0032
DG	$\overline{DG}$	0,0002	0,0001
	σ_{DG}^2	0	0

A população inicial gerada pelo MODE é mostrada na Fig. 8.22.

**Figura 8.22:** População inicial para a função ZDT4.

As comparações das soluções obtidas pelo MODE, MOGA I e PMOGA são apresentadas na Fig. 8.23.

Os algoritmos MODE e NSGA II avaliam 30100 e 25100 vezes a função objetivo, respectivamente.

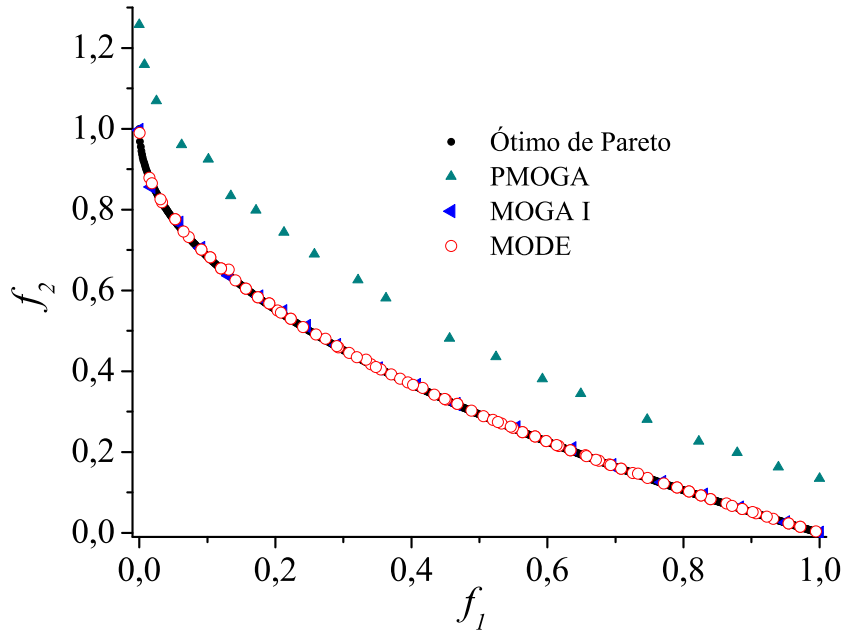


Figura 8.23: Curva de Pareto para a função ZDT4.

Observa-se, a partir da Fig. 8.21, que nenhum dos algoritmos apresentados no trabalho de Zitzler et al. (2000) consegue obter uma boa aproximação para a curva de Pareto. Já na Fig. 8.23, observa-se que o PMOGA atinge apenas um ótimo local e tanto o MOGA I quanto o MODE conseguem uma boa aproximação da solução analítica.

Função ZDT6

A função ZDT6 apresenta duas dificuldades causadas pela não uniformidade do espaço de busca: as soluções ótimas de Pareto não estão uniformemente distribuídas ao longo da fronteira global de Pareto e a densidade das soluções é menor perto da fronteira ótima de Pareto e maior na região afastada dela (ZITZLER et al., 2000; DEB, 2001). ZDT6 é dada por:

$$\text{ZDT6} = \begin{cases} f_1(x_1) = 1 - \exp(-4x_1) \sin^6(6\pi x_1) \\ g(x_2, \dots, x_m) = 1 + 9 \left(\sum_{i=2}^m \frac{x_i}{9} \right)^{0,25} \\ h(f_1(x_1), g(x_2, \dots, x_m)) = 1 - (f_1(x_1)/g(x_2, \dots, x_m))^2 \end{cases} \quad (8.11)$$

Todas as variáveis são definidas no intervalo $(0,1)$. A região ótima de Pareto corresponde a $x_1 \in (0,1)$ e $x_i = 0$ ($i = 2, \dots, 10$).

A Figura 8.24 traz os perfis obtidos, para esse caso, pelos algoritmos clássicos citados anteriormente, para a configuração dada na Tab. 8.12.

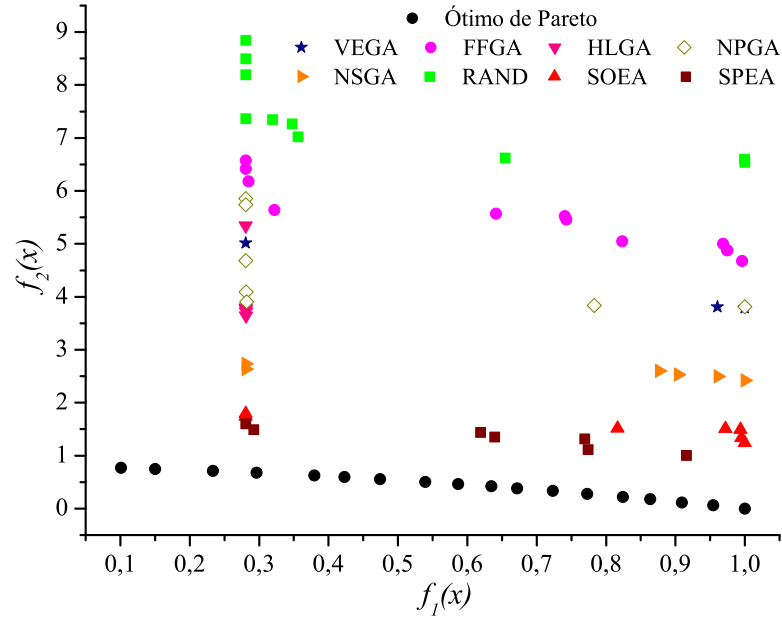


Figura 8.24: Resolução da função ZDT6 usando algoritmos clássicos.

Os parâmetros utilizados na resolução através de algoritmos evolutivos para o problema ZDT6 são listados na Tab. 8.21.

Tabela 8.21: Parâmetros utilizados na resolução da função ZDT6.

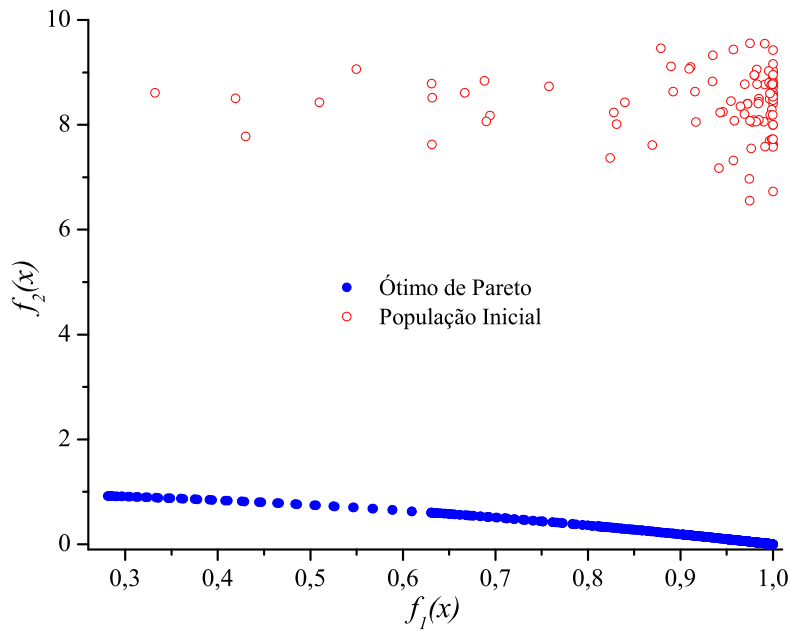
Parâmetro	ALGORITMO		
	PMOGA	MOGA I	MODE
N_{gen}	250	50000	150
N	100	20	100
p_c	0,80	0,80	0,85
p_m	0,01	0,05	-
F	-	-	0,50
R	-	-	10
r	-	-	0,90

A Tabela 8.22 apresenta as métricas de desempenho do MODE comparadas ao algoritmo genético NSGA II.

Tabela 8.22: Métricas de desempenho para a função ZDT6.

Métrica		NSGA II	MODE
		(250 Gerações)	(150 Gerações)
Υ	$\overline{\Upsilon}$	0,2965	0,0016
	σ_{Υ}^2	0,0131	0,0003
Δ	$\overline{\Delta}$	0,6680	0,3763
	σ_{Δ}^2	0,0099	0,0078
DG	$\overline{DG}$	0,0023	0,0001
	σ_{DG}^2	0,0003	0

A população inicial gerada pelo MODE é mostrada na Fig. 8.25.

**Figura 8.25:** População inicial para a função ZDT6.

A solução obtida pelo MODE, comparada ao MOGA I e ao PMOGA, é apresentada na Fig. 8.26.

Os algoritmos MODE e NSGA II avaliaram 30100 e 25100 vezes a função objetivo, respectivamente.

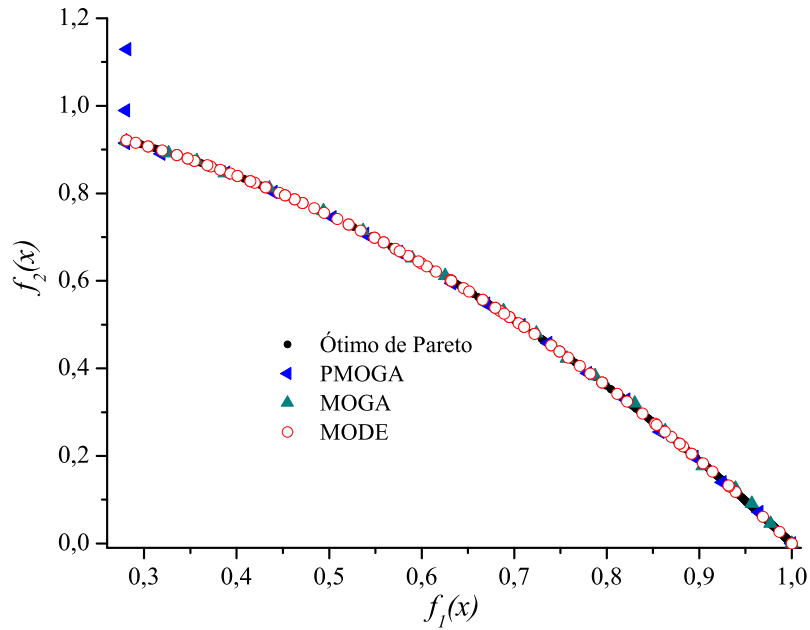


Figura 8.26: Curva de Pareto para a função ZDT6.

Observa-se nestas figuras que somente o SPEA, o PMOGA, o MOGA I e o MODE conseguem obter uma boa aproximação para a curva de Pareto, ainda que no caso do SPEA, a distribuição das soluções não-dominadas seja muito pobre.

Função VNT

Seja o seguinte problema de otimização com três objetivos e duas variáveis de projeto, apresentado em Deb (2001):

$$\text{VNT} = \begin{cases} \min f_1(x_1, x_2, x_3) = 0,5(x_1^2 + x_2^2) + \sin(x_1^2 + x_2^2) \\ \min f_2(x_1, x_2, x_3) = \frac{(3x_1 - 2x_2 + 4)^2}{8} + \frac{(x_1 - x_2 + 1)^2}{27} + 15 \\ \min f_3(x_1, x_2, x_3) = \frac{1}{x_1^2 + x_2^2 + 1} - 1,1 \exp(-(x_1^2 + x_2^2)) \end{cases} \quad (8.12)$$

com $x_i \in (-3, 3)$ ($i = 1, 2$).

A localização do ótimo de Pareto pode ser obtida através da seguinte mudança de variável:

$$w = x_1^2 + x_2^2 \quad (8.13)$$

Dessa forma, os objetivos $f_1(x_1, x_2, x_3)$ e $f_3(x_1, x_2, x_3)$ ficam:

$$f_1(w) = 0,5w + \sin(w) \quad (8.14)$$

$$f_3(w) = \frac{1}{1+w} - 1,1 \exp(-w) \quad (8.15)$$

com $0 < w < 18$. Para cada valor de w , o mínimo da segunda função objetivo pode ser uma solução candidata para o verdadeiro conjunto ótimo de Pareto.

Os parâmetros utilizados para os algoritmos NSGA II, SPEA II e MODE são mostrados na Tab. 8.23.

Tabela 8.23: Parâmetros utilizados na resolução da função VNT.

Parâmetro	ALGORITMO		
	NSGA II	SPEA II	MODE
N_{gen}	250	250	150
N	100	100	100
p_c	0,80	0,80	0,85
p_m	0,01	0,05	-
F	-	-	0,50
R	-	-	10
r	-	-	0,90

A Tabela 8.24 apresenta a comparação das métricas de desempenho do MODE com relação ao NSGA II.

Tabela 8.24: Métricas de desempenho para a função VNT.

Métrica		NSGA II	MODE
		(250 Gerações)	(150 Gerações)
Υ	$\overline{\Upsilon}$	0,1145	0,1241
	σ_{Υ}^2	0,0074	0,0004
Δ	$\overline{\Delta}$	0,7358	0,7747
	σ_{Δ}^2	0,1478	0,1699
DG	$\overline{DG}$	-	0,0014
	σ_{DG}^2	-	0,0004

A população inicial gerada e a solução obtida pelo MODE são apresentadas nas Figs. 8.27 e 8.28, respectivamente.

Os algoritmos MODE e NSGA II avaliaram 30100 e 25100 vezes a função objetivo, respectivamente.

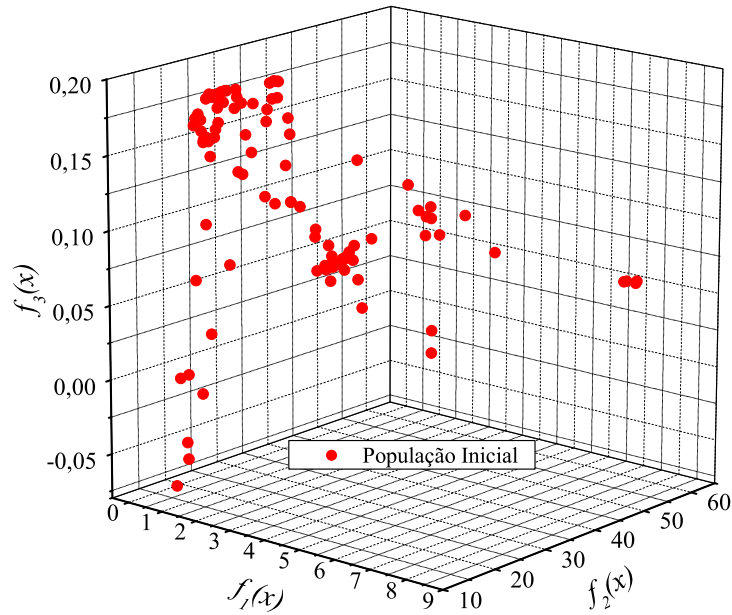


Figura 8.27: População inicial para a função VNT.

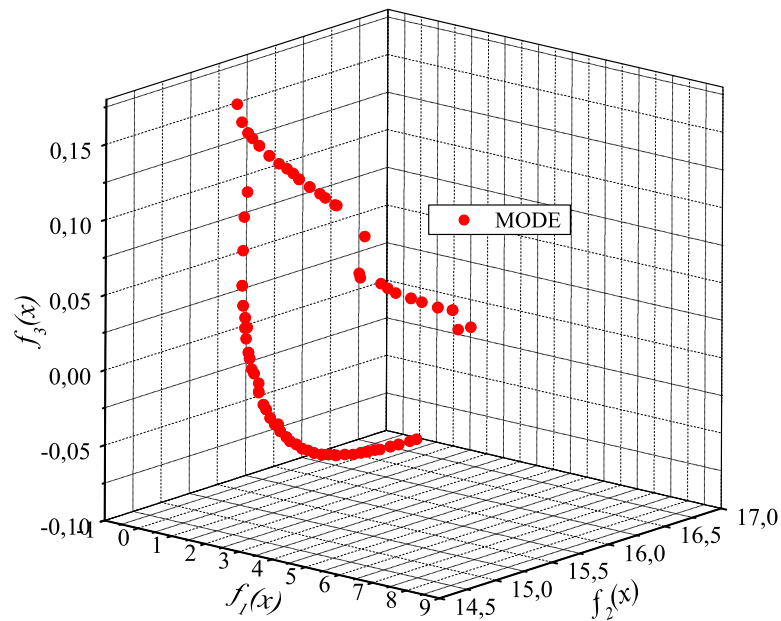


Figura 8.28: Curva de Pareto para a função VNT.

Com o intuito de verificar a qualidade da aproximação da curva de Pareto, nas Figs. 8.29, 8.30 e 8.31 são plotados os objetivos obtidos pelo MODE e pelo NSGA II (DEB, 2001) nos planos $f_1 \times f_2$, $f_1 \times f_3$ e $f_2 \times f_3$, respectivamente. A partir dessas figuras é possível observar-se boa concordância dos resultados obtidos com os encontrados na literatura (DEB, 2001).

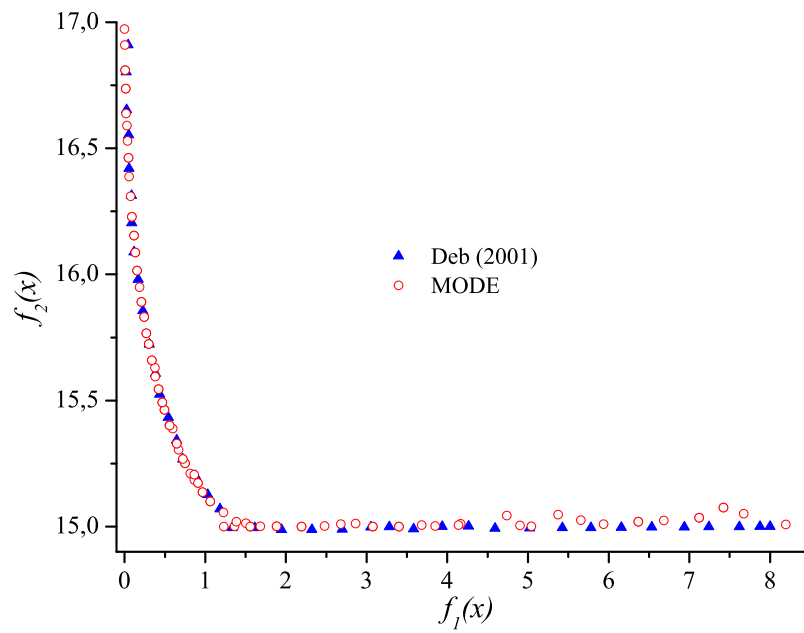


Figura 8.29: Espaço de objetivos $f_2 \times f_1$ para a função VNT.

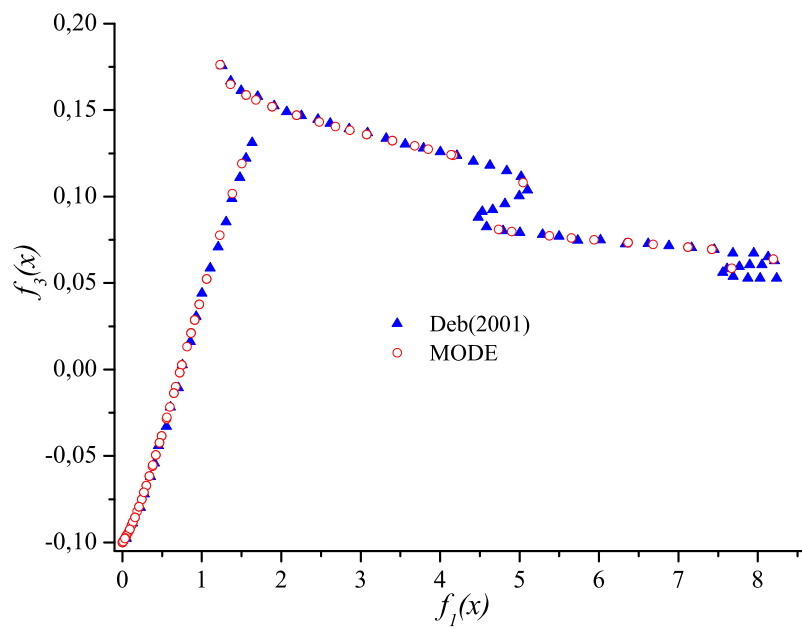


Figura 8.30: Espaço de objetivos $f_3 \times f_1$ para a função VNT.

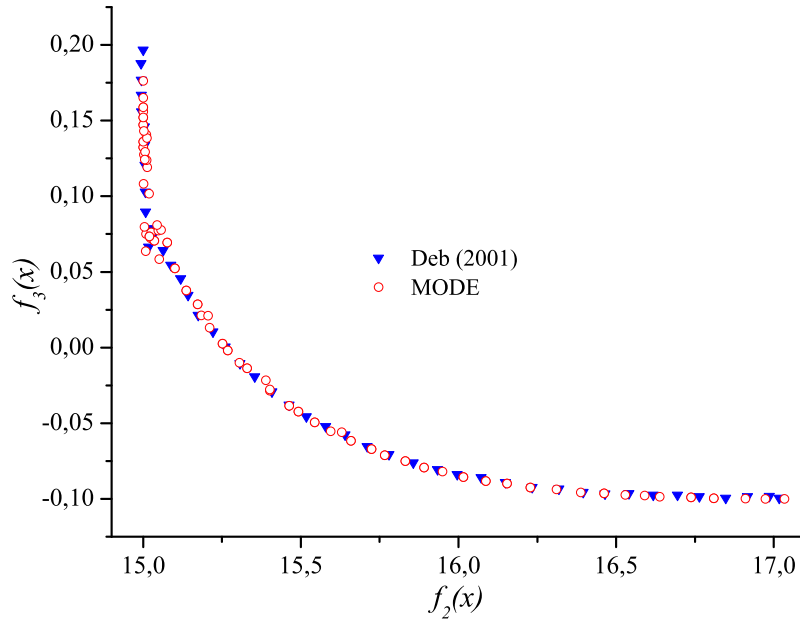


Figura 8.31: Espaço de objetivos $f_3 \times f_2$ para a função VNT.

Função DTLZ2

Seja o problema de otimização com três objetivos e com doze variáveis de projeto descrito em Deb (2001):

$$\text{DTLZ2} = \begin{cases} \min f_1(x_1, x_2, \dots, x_{12}) = (1 + g(x)) \cos(0,5\pi x_1) \cos(0,5\pi x_2) \\ \min f_2(x_1, x_2, \dots, x_{12}) = (1 + g(x)) \cos(0,5\pi x_1) \sin(0,5\pi x_2) \\ \min f_3(x_1, x_2, \dots, x_{12}) = (1 + g(x)) \sin(0,5\pi x_1) \\ g(x_3, \dots, x_{12}) = \sum_{i=3}^{12} (x_i - 0,5)^2 \end{cases} \quad (8.16)$$

com $x_i \in (0, 1)$ ($i = 1, 2, \dots, 12$).

DTLZ2 apresenta solução analítica dada por x_1 e $x_2 \in (0,1)$ e $x_i=0,5$ ($i = 3, \dots, 12$).

Os parâmetros utilizados para os algoritmos NSGA II, SPEA II e MODE são mostrados na Tab. 8.25. A Tabela 8.26 apresenta a comparação das métricas de desempenho do MODE com relação ao NSGA II.

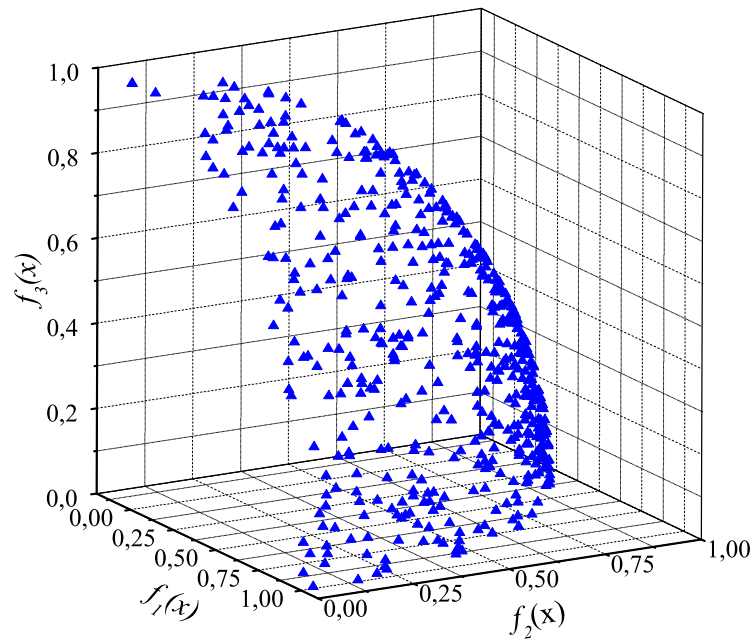
A população inicial gerada e a solução obtida pelo MODE, são apresentadas nas Figs. 8.32 e 8.33, respectivamente, com 30100 avaliações da função objetivo pelo MODE e 25100 avaliações pelo NSGA II.

Tabela 8.25: Parâmetros utilizados na resolução da função DTLZ2.

Parâmetro	ALGORITMO		
	NSGA II	SPEA II	MODE
N_{gen}	250	250	150
N	100	100	100
p_c	0,80	0,80	0,85
p_m	0,01	0,05	-
F	-	-	0,50
R	-	-	10
r	-	-	0,90

Tabela 8.26: Métricas de desempenho para a função DTLZ2.

Métrica		NSGA II (250 Gerações)	MODE (150 Gerações)
Υ	$\overline{\Upsilon}$	0,5123	0,6547
	σ_{Υ}^2	0,1124	0,0455
	$\overline{\Delta}$	0,8958	0,9878
Δ	σ_{Δ}^2	0,0478	0,1489
	$\overline{DG}$	-	0,0145
DG	σ_{DG}^2	-	0,0047

**Figura 8.32:** Curva de Pareto para a função DTLZ2 usando o NSGA II.

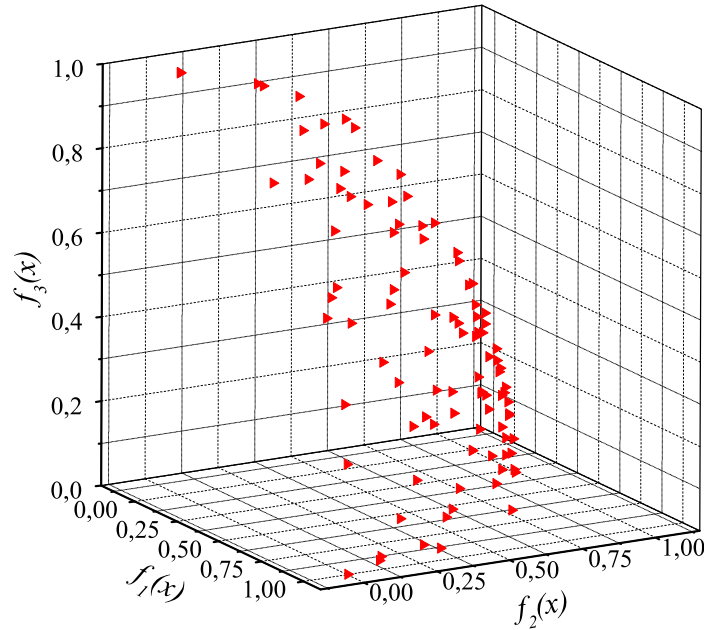


Figura 8.33: Curva de Pareto para a função DTLZ2 usando o MODE.

8.2 Funções-Teste com Restrições para POMO

Os problemas de otimização contemplam inerentemente restrições, mesmo que essas sejam dadas apenas pelos domínios das variáveis de projeto. As restrições de igualdade e desigualdade são oriundas de balanços de massa, energia e quantidade de movimento, de questões ambientais, de projeto, de custo, entre outras, sendo bastante comuns nas situações reais de engenharia. A presença de restrições em problemas com um ou com múltiplos objetivos pode causar enorme dificuldade para a obtenção de sua(s) solução(ões) em termos de convergência e de diversidade (este último para o caso multi-objetivo).

Função Min-Ex

Seja o problema com duas variáveis e com dois objetivos, dado por (DEB, 2001):

$$\text{MIN-EX} = \begin{cases} \min f_1(x) = x_1 \\ \min f_2(x) = \frac{1+x_2}{x_1} \\ g_1 \equiv x_2 + 9x_1 \geq 6 \\ g_2 \equiv -x_2 + 9x_1 \geq 1 \end{cases} \quad (8.17)$$

com $0,1 \leq x_1 \leq 1$ e $0 \leq x_2 \leq 5$.

A solução ótima de Pareto para esse problema consiste de duas fases. Na primeira, $0,39 \leq x_1 \leq 0,67$ e $x_2 = 6 - 9x_1$ e, na segunda, $0,67 \leq x_1 \leq 1,0$ e $x_2 = 0$.

Os parâmetros utilizados pelo MODE são apresentados na Tab. 8.27.

Tabela 8.27: Parâmetros utilizados na resolução da função MIN-EX.

Parâmetro	MODE
N_{gen}	50
N	100
p_c	0,85
F	0,50
R	10
r	0,90
r_{p1}	1,2
r_{p2}	9

A população inicial e a solução obtida pelo MODE são apresentadas nas Figs. 8.34 e 8.35, respectivamente.

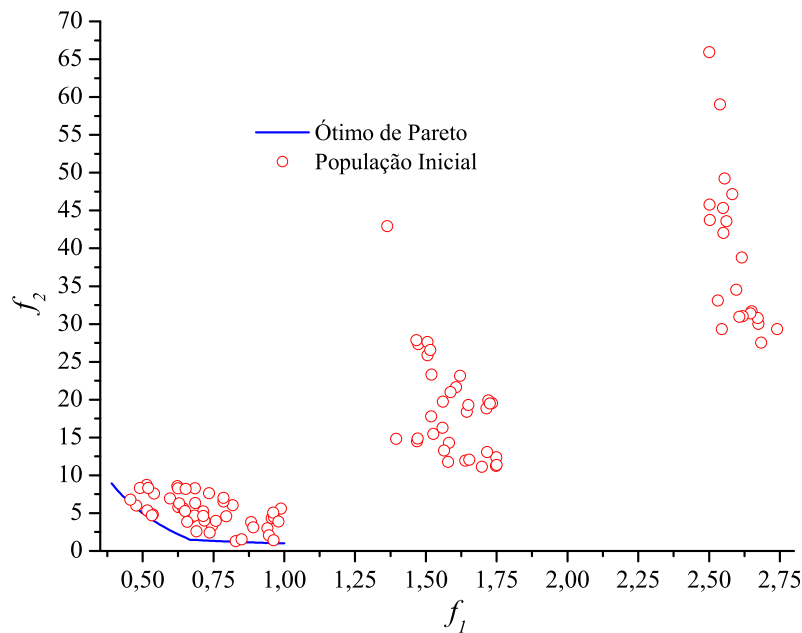


Figura 8.34: População inicial para a função MIN-EX.

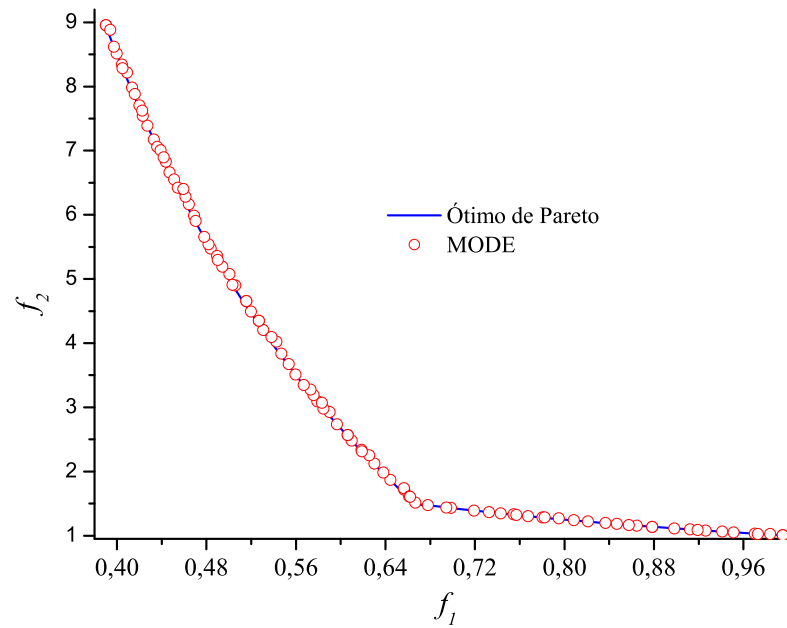


Figura 8.35: Curva de Pareto para a função MIN-EX.

A Tabela 8.28 apresenta as métricas de desempenho obtidas pelo MODE com 10100 avaliações da função objetivo.

Tabela 8.28: Métricas de desempenho para a função MIN-EX.

Métrica		MODE
Υ	$\overline{\Upsilon}$	0,0001
	σ_{Υ}^2	0
Δ	$\overline{\Delta}$	0,0047
	σ_{Δ}^2	0
DG	$\overline{DG}$	0,0001
	σ_{DG}^2	0

Função BNH

Considere o seguinte problema proposto por Binh e Korn (1997):

$$\text{BNH} = \begin{cases} \min f_1(x) = 4x_1^2 + 4x_2^2 \\ \min f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2 \\ g_1 \equiv (x_1 - 5)^2 + x_2^2 \leq 25 \\ g_2 \equiv (x_1 - 8)^2 + (x_2 + 3)^2 \geq 7,7 \end{cases} \quad (8.18)$$

com $0 \leq x_1 \leq 5$ e $0 \leq x_2 \leq 3$.

A solução ótima de Pareto para esse problema consiste de duas fases. Na primeira, $x_1 = x_2 \in (0, 3)$ e, na segunda, $x_1 \in (3, 5)$ e $x_2 = 3$.

A Tabela 8.29 apresenta os parâmetros utilizados pelo algoritmo MODE.

Tabela 8.29: Parâmetros utilizados na resolução da função BNH.

Parâmetro	MODE
N_{gen}	100
N	50
p_c	0,85
F	0,50
R	10
r	0,90

A população inicial é apresentada na Fig. 8.36.

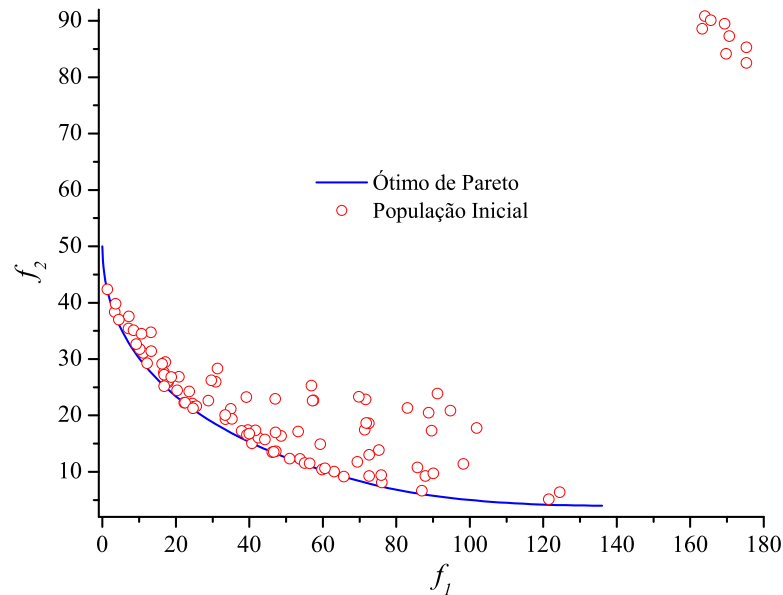


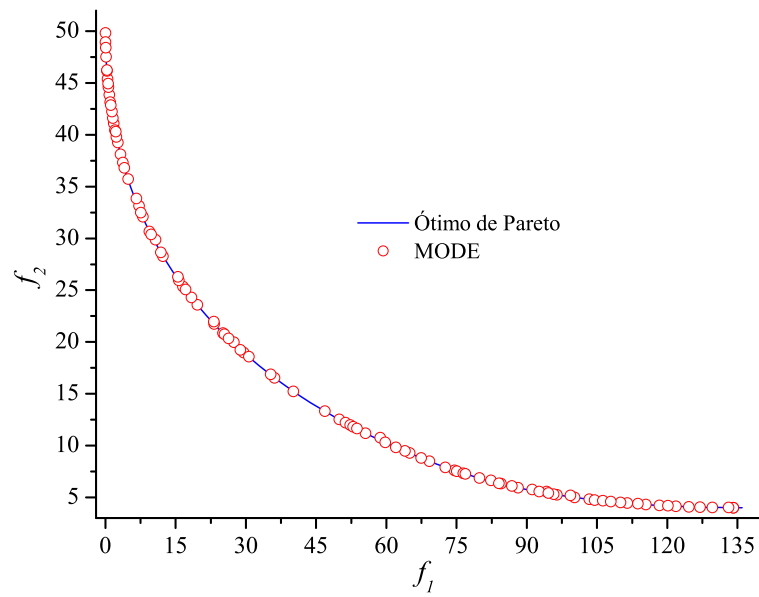
Figura 8.36: População inicial para a função BNH.

A Tabela 8.30 apresenta as métricas obtidas pelo algoritmo MODE com 10050 avaliações da função objetivo. Os parâmetros de penalidade, r_{p1} e r_{p2} , são definidos como sendo iguais a 140 e 60, respectivamente.

A solução obtida pelo MODE é apresentada na Fig. 8.37.

Tabela 8.30: Métricas de desempenho para a função BNH.

Métrica		MODE
Υ	$\overline{\Upsilon}$	0,2027
	σ_{Υ}^2	0,0021
Δ	$\overline{\Delta}$	0,0249
	σ_{Δ}^2	0,0001
DG	$\overline{DG}$	0,2727
	σ_{DG}^2	0,0124

**Figura 8.37:** Curva de Pareto para a função BNH.

Função OSY

Seja o problema com seis variáveis de projeto proposto por Osyczka e Kundu (1995):

$$\text{OSY} = \begin{cases} \min f_1(x) = -(25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2 + \\ \quad \quad \quad + (x_4 - 4)^2 + (x_5 - 1)^2) \\ \min f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \\ g_1 \equiv x_1 + x_2 - 2 \geq 0 \\ g_2 \equiv 6 - x_1 - x_2 \geq 0 \\ g_3 \equiv 2 - x_2 + x_1 \geq 0 \\ g_4 \equiv 2 - x_1 + 3x_2 \geq 0 \\ g_5 \equiv 4 - (x_3 - 3)^2 - x_4 \geq 0 \\ g_6 \equiv (x_5 - 3)^2 + x_6 - 4 \geq 0 \end{cases} \quad (8.19)$$

com $0 \leq x_1, x_2, x_6 \leq 10$, $1 \leq x_3, x_5 \leq 5$ e $0 \leq x_4 \leq 6$.

A solução ótima de Pareto é apresentada na Tab. 8.31.

Tabela 8.31: Ótimo de Pareto para a função OSY.

Solução Analítica				Restrições
x_1	x_2	x_3	x_5	Ativas
5	1	(1 ... 5)	5	g_2, g_4, g_6
5	1	(1 ... 5)	1	g_2, g_4, g_6
(4,056 ... 5)	$(x_1-2)/3$	1	1	g_4, g_5, g_6
0	2	(1 ... 3,732)	1	g_1, g_3, g_6
(0 ... 1)	$2-x_1$	1	1	g_1, g_5, g_6

Os parâmetros e os resultados obtidos pelo MODE, com 10100 avaliações da função objetivo, são apresentados na Tab. 8.32 e na Tab. 8.33. Os parâmetros de penalidade, r_{p1} e r_{p2} , são definidos como sendo 25 e 80, respectivamente.

Tabela 8.32: Parâmetros utilizados na resolução da função OSY.

Parâmetro	MODE
N_{gen}	50
N	100
p_c	0,85
F	0,50
R	10
r	0,90

Tabela 8.33: Métricas de desempenho para a função OSY.

Métrica	MODE
Υ	$\overline{\Upsilon}$
	σ_{Υ}^2
Δ	$\overline{\Delta}$
	σ_{Δ}^2
DG	$\overline{DG}$
	σ_{DG}^2

A população inicial e a solução obtida pelo MODE são apresentadas nas Figs. 8.38 e 8.39, respectivamente.

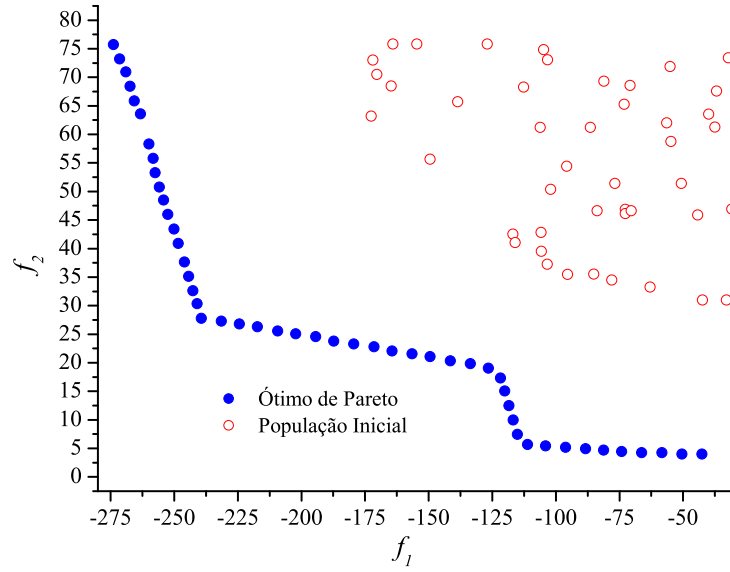


Figura 8.38: População inicial para a função OSY.

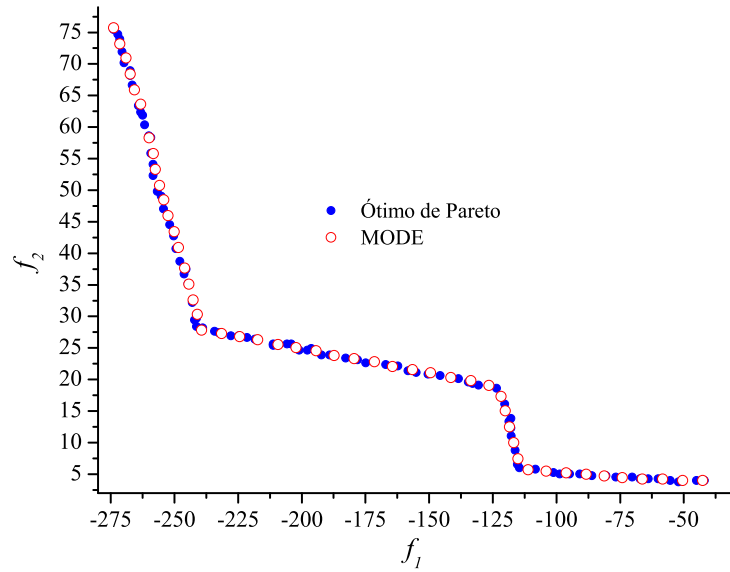


Figura 8.39: Curva de Pareto para a função OSY.

Função SRN

Seja o problema estudado por Srinivas e Deb (1994):

$$\text{SRN} = \begin{cases} \min f_1(x) = 2 + (x_1 - 2)^2 + (x_2 - 1)^2 \\ \min f_2(x) = 9x_1 - (x_2 - 1)^2 \\ g_1 \equiv x_1^2 + x_2^2 \leq 225 \\ g_2 \equiv x_1 - 3x_2 + 10 \leq 0 \end{cases} \quad (8.20)$$

com $-20 \leq x_1, x_2 \leq 20$.

A solução ótima de Pareto para esse problema consiste de $x_1 = -2,5$ e $x_2 \in (2,5 \text{ } 14,79)$. Os parâmetros utilizados pelo MODE são apresentados na Tab. 8.34. Os parâmetros de penalidade, r_{p1} e r_{p2} , são definidos como 200 e 20, respectivamente.

Tabela 8.34: Parâmetros utilizados na resolução da função SRN.

Parâmetro	MODE
N_{gen}	50
N	100
p_c	0,85
F	0,50
R	10
r	0,90

A população inicial e a solução obtida pelo MODE são apresentadas nas Figs. 8.40 e 8.41, respectivamente.

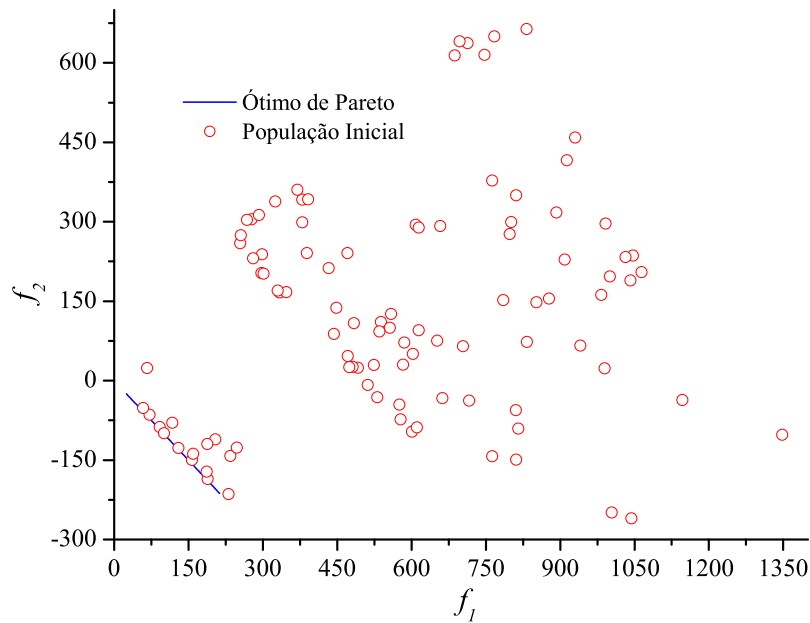


Figura 8.40: População inicial para a função SRN.

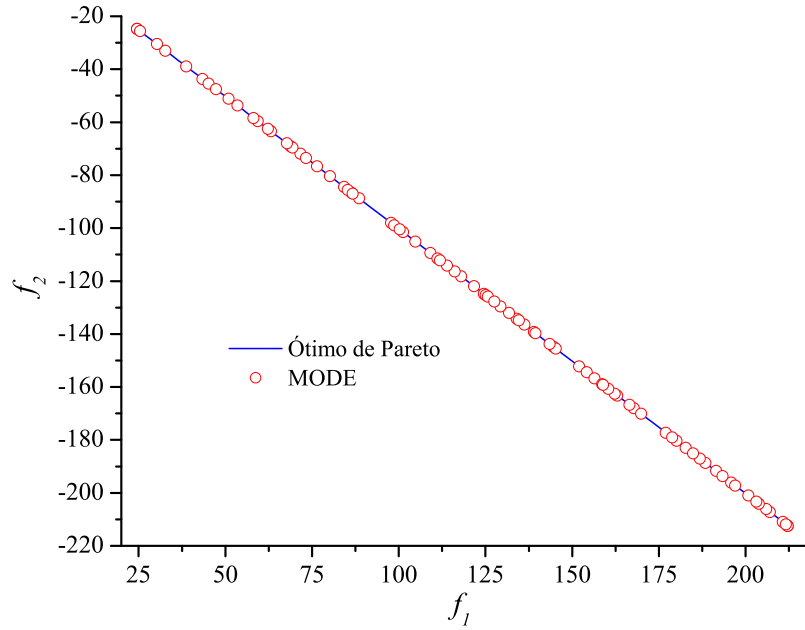


Figura 8.41: Curva de Pareto para a função SRN.

A Tabela 8.35 apresenta as métricas de desempenho para a função SRN com 10100 avaliações da função objetivo.

Tabela 8.35: Métricas de desempenho para a função SRN.

Métrica		MODE
Υ	$\overline{\Upsilon}$	0,3398
	$\sigma_{\overline{\Upsilon}}^2$	0,0231
Δ	$\overline{\Delta}$	0,0418
	$\sigma_{\overline{\Delta}}^2$	0,0012
DG	$\overline{DG}$	0,2723
	$\sigma_{\overline{DG}}^2$	0,0041

Função TNK

Seja o problema proposto por Tanaka (1995):

$$\text{TNK} = \begin{cases} \min f_1(x) = x_1 \\ \min f_2(x) = x_2 \\ g_1 \equiv x_1^2 + x_2^2 - 1 - 0,1 \cos \left(16 \arctan \left(\frac{x_1}{x_2} \right) \right) \geq 0 \\ g_2 \equiv (x_1 - 0,5)^2 + (x_2 - 0,5)^2 \leq 0,5 \end{cases} \quad (8.21)$$

com $0 \leq x_1, x_2 \leq \pi$.

Os parâmetros utilizados pelo MODE são apresentados na Tab. 8.36. Os parâmetros de penalidade, r_{p1} e r_{p2} , são ambos definidos como iguais a 1,2.

Tabela 8.36: Parâmetros utilizados na resolução da função TNK.

Parâmetro	MODE
N_{gen}	80
N	100
p_c	0,85
F	0,50
R	10
r	0,90

A população inicial e a solução obtida pelo MODE são apresentadas nas Figs. 8.42 e 8.43, respectivamente.

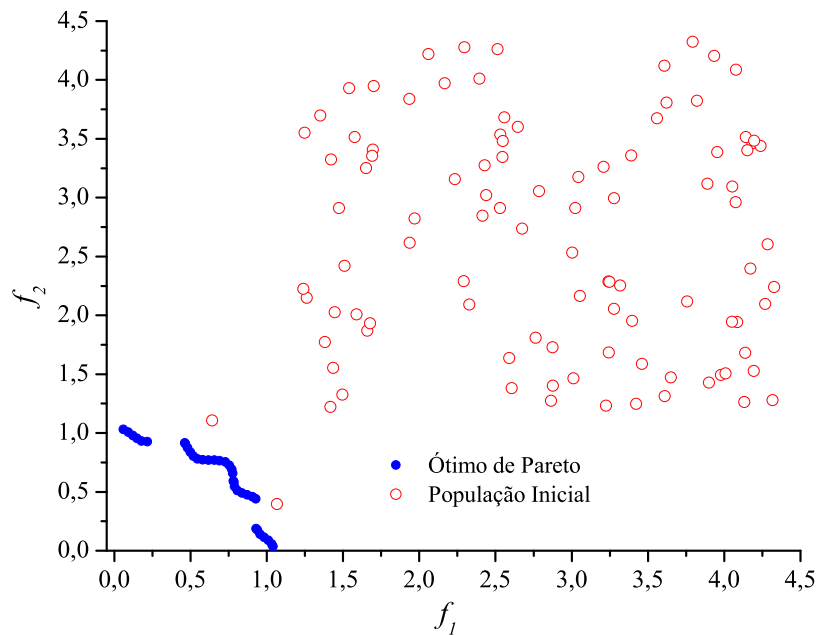


Figura 8.42: População inicial para a função TNK.

As métricas de desempenho são apresentadas na Tab. 8.37 com 16100 avaliações da função objetivo.

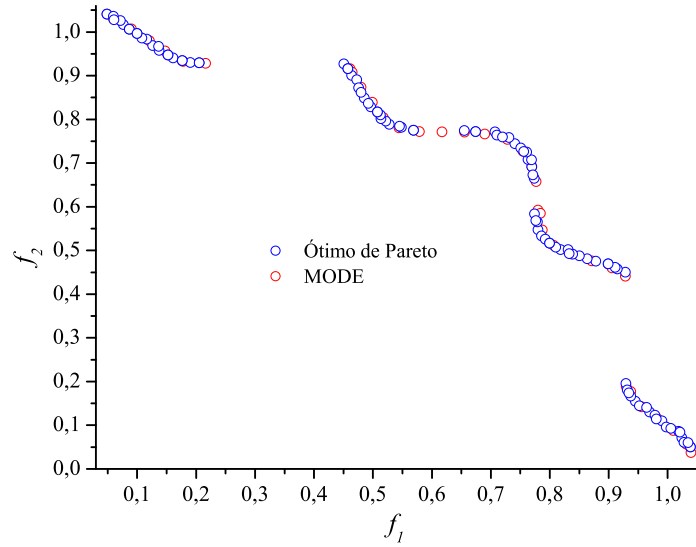


Figura 8.43: Curva de Pareto para a função TNK.

Tabela 8.37: Métricas de desempenho para a função TNK.

Métrica		MODE
Υ	$\overline{\Upsilon}$	0,0046
	σ_{Υ}^2	0,0001
Δ	$\overline{\Delta}$	0,0005
	σ_{Δ}^2	0,0000
DG	$\overline{DG}$	0,1974
	σ_{DG}^2	0,0019

Função 5OBJ

Seja o problema estudado por Ray et al. (2001):

$$5\text{OBJ} = \begin{cases} \min f_1(x) = 106780,37(x_2 + x_3) + 61704,67 \\ \min f_2(x) = 3000x_1 \\ \min f_3(x) = 28534689,65x_2 \\ \min f_4(x) = 572250 \exp(-39,75x_2 + 9,9x_3 + 2,74) \\ \min f_5(x) = 34,75/(x_1x_2) + 123500x_3 - 2000 \\ g_1 \equiv 0,00139/(x_1x_2) + 4,94x_3 - 1,08 \leq 0 \\ g_2 \equiv 0,000306/(x_1x_2) + 1,082x_3 - 1,0986 \leq 0 \\ g_3 \equiv 12,307/(x_1x_2) + 49408,24x_3 - 45948,98 \leq 0 \\ g_4 \equiv 2,098/(x_1x_2) + 8046,33x_3 - 16696,71 \leq 0 \\ g_5 \equiv 2,138/(x_1x_2) + 7883,39x_3 - 10705,04 \leq 0 \\ g_6 \equiv 0,417x_1x_2 + 1721,26x_3 - 2136,54 \leq 0 \\ g_7 \equiv 0,164/(x_1x_2) + 631,13x_3 - 604,48 \leq 0 \end{cases} \quad (8.22)$$

com $0,01 \leq x_1 \leq 0,45$ e $0,01 \leq x_2, x_3 \leq 0,10$.

A função 5OBJ foi normalizada por (RAY et al., 2001) da seguinte maneira: $f_1/(8 \times 10^6)$, $f_2/(1,5 \times 10^3)$, $f_3/(3 \times 10^6)$, $f_4/(6 \times 10^6)$ e $f_5/(8 \times 10^3)$.

Os parâmetros utilizados pelos algoritmos NSGA II e MODE são mostrados na Tab. 8.38.

Tabela 8.38: Parâmetros utilizados na resolução da função 5OBJ.

Parâmetro	ALGORITMO	
	NSGA II	MODE
N_{gen}	500	250
N	100	50
p_c	0,80	0,85
p_m	0,01	-
F	-	0,50
R	-	10
r	-	0,90

Cabe destacar que em ambos os algoritmos foram usados $r_{pi}=10$ ($i = 1, \dots, 5$).

As soluções obtidas e apresentadas no plano, pelo NSGA II e MODE são mostradas nas Figs. 8.44 a 8.53.

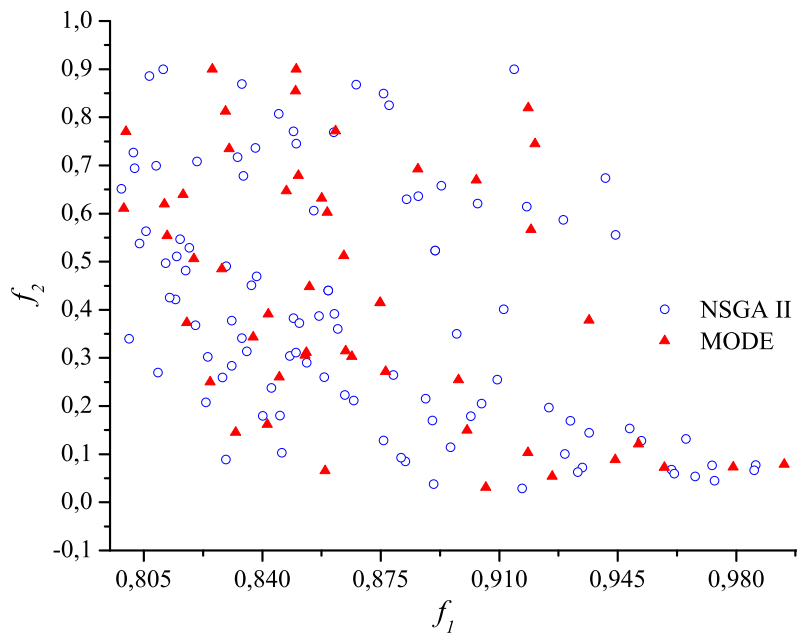


Figura 8.44: $f_1 \times f_2$ para a função 5OBJ.

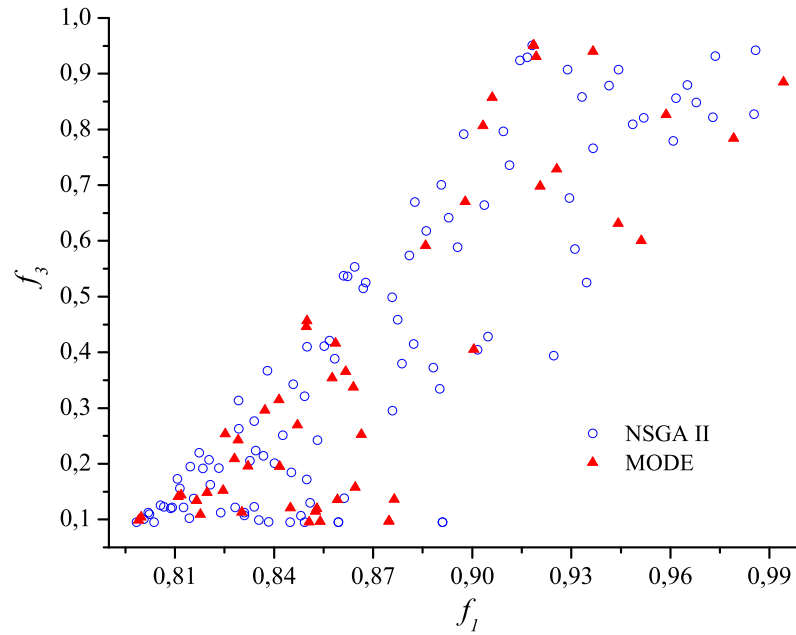


Figura 8.45: $f_1 \times f_3$ para a função 5OBJ.

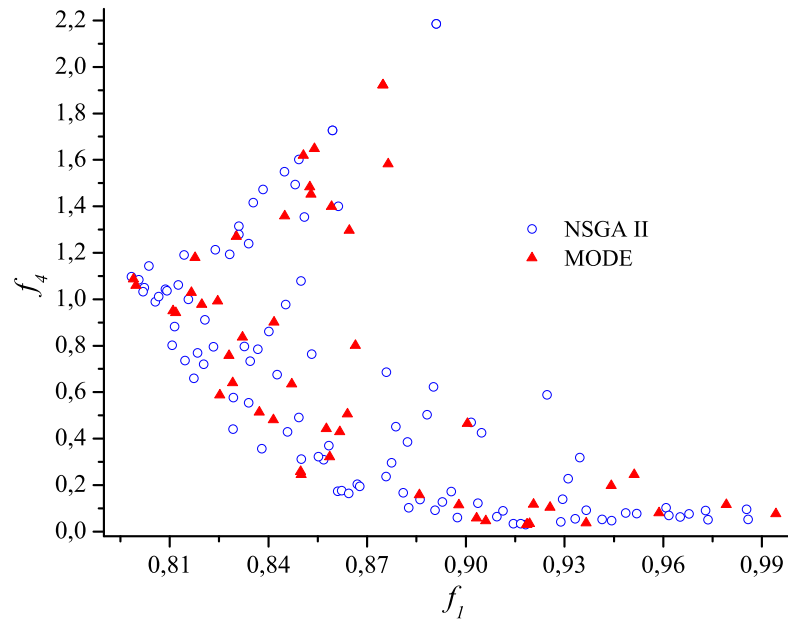


Figura 8.46: $f_1 \times f_4$ para a função 5OBJ.

Os algoritmos MODE e NSGA II avaliaram 25050 e 25100 vezes a função objetivo durante a solução do problema 5OBJ. Cabe ressaltar que, apesar da dificuldade de comparação entre os resultados obtidos pelos algoritmos, observa-se coerência entre os resultados reportados por (RAY et al., 2001) e (DEB, 2001).

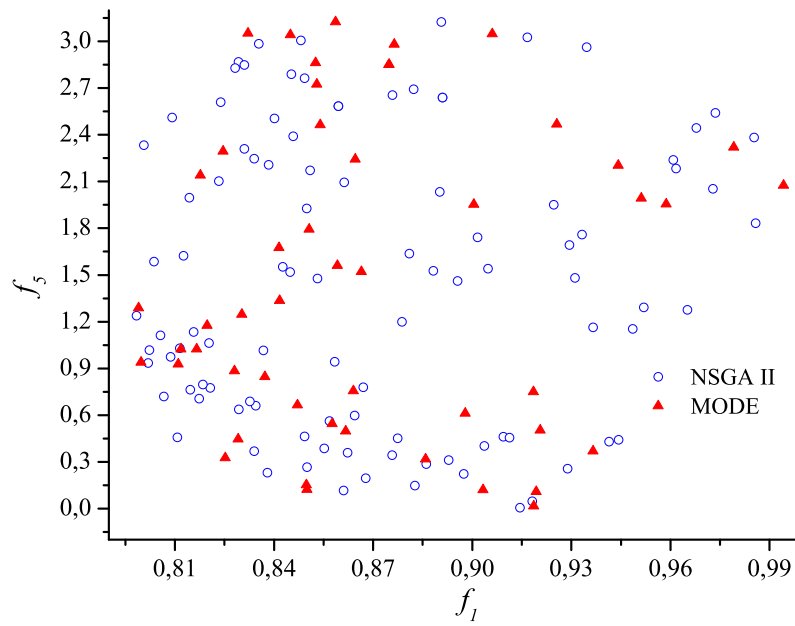


Figura 8.47: $f_1 \times f_5$ para a função 5OBJ.

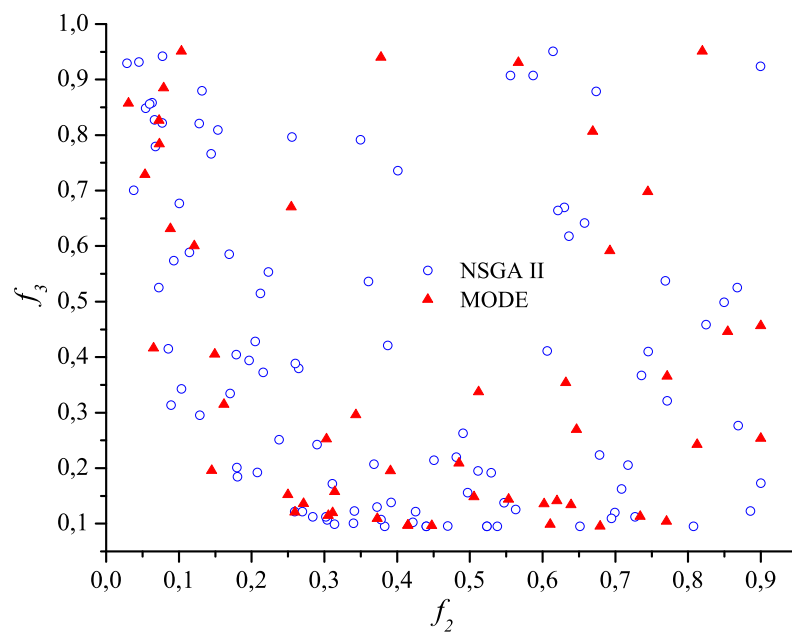


Figura 8.48: $f_2 \times f_3$ para a função 5OBJ.

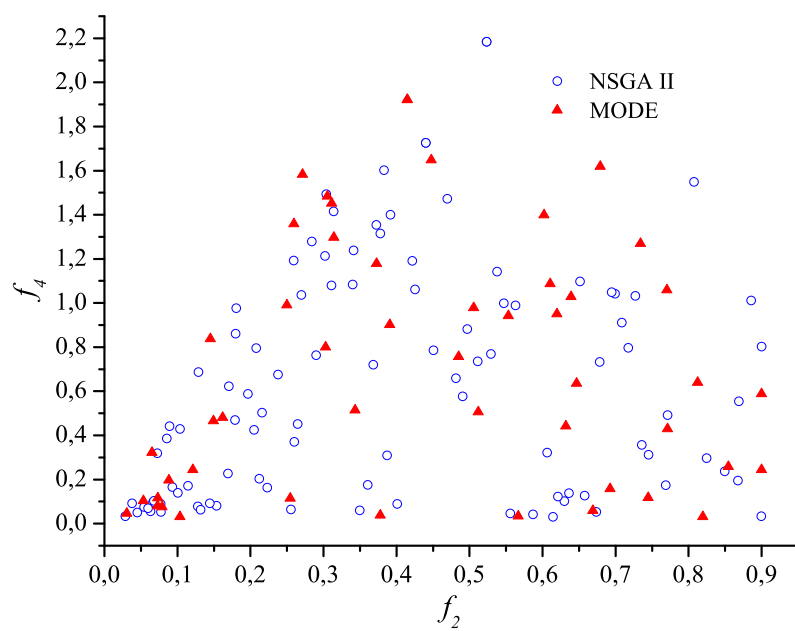


Figura 8.49: $f_2 \times f_4$ para a função 5OBJ.

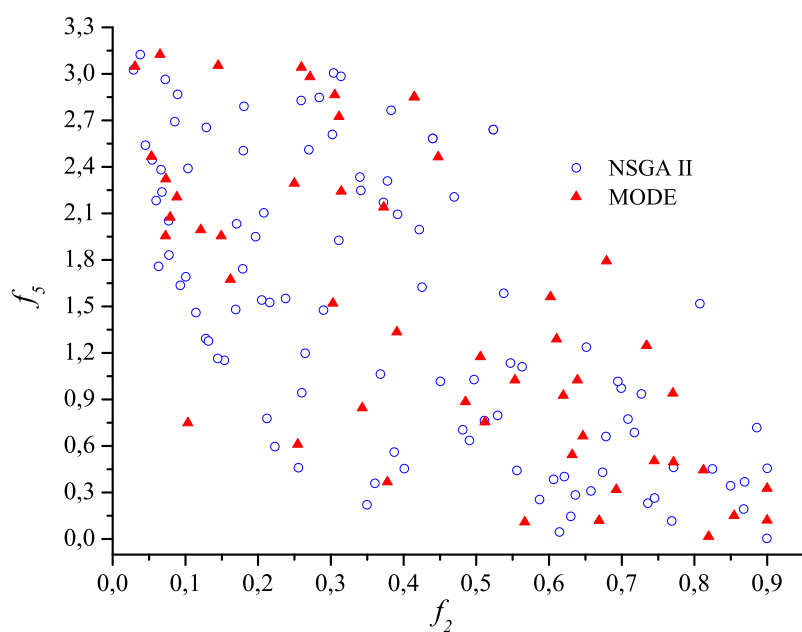


Figura 8.50: $f_2 \times f_5$ para a função 5OBJ.

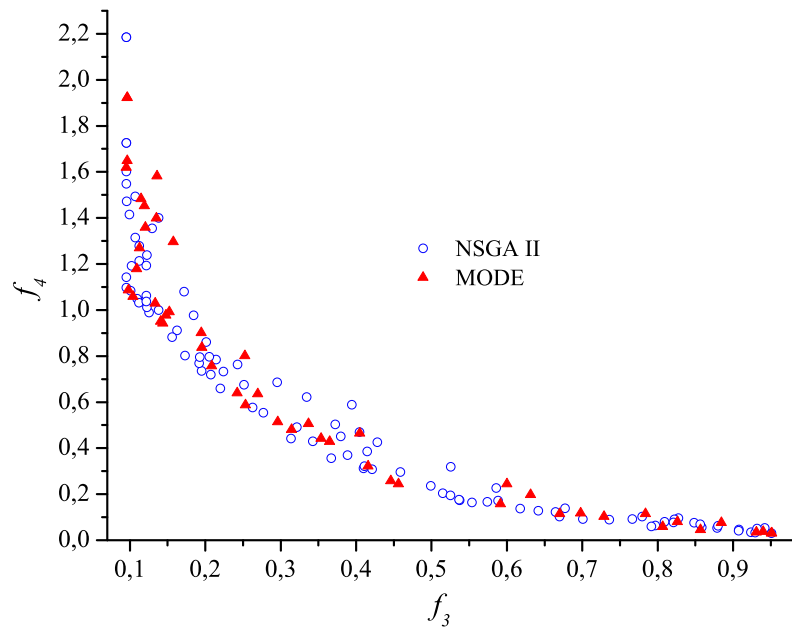


Figura 8.51: $f_3 \times f_4$ para a função 5OBJ.

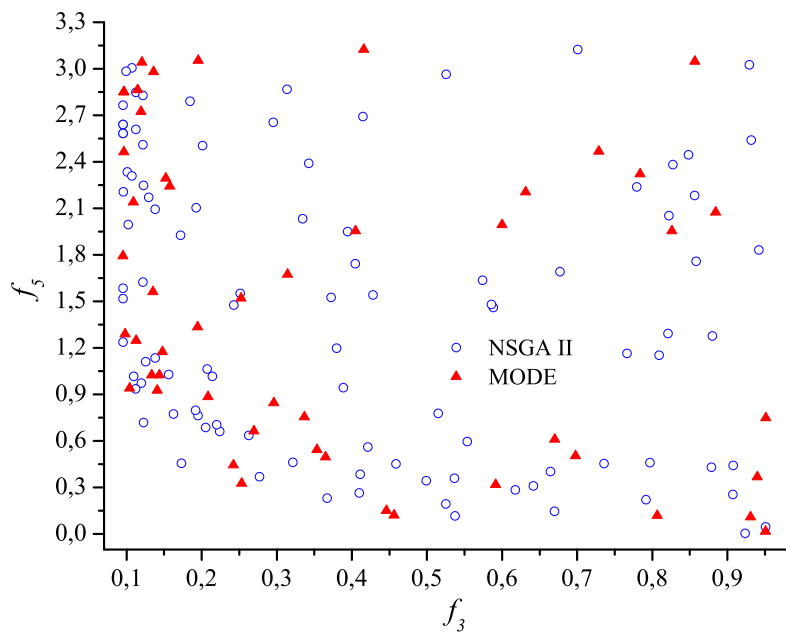


Figura 8.52: $f_3 \times f_5$ para a função 5OBJ.

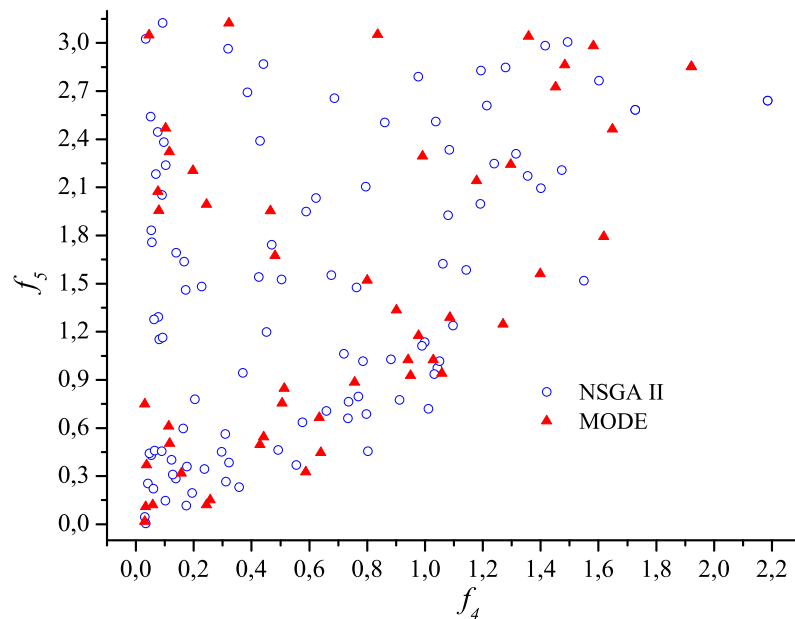


Figura 8.53: $f_4 \times f_5$ para a função 5OBJ.

8.3 Estudo de Sensibilidade do Algoritmo MODE

Uma das principais dificuldades encontradas no desenvolvimento de Algoritmos Não Determinísticos é a utilização de um número considerável de parâmetros, tais como o tamanho da população e a probabilidade de cruzamento, e como escolher valores adequados para estes parâmetros. De forma similar, porém menos significativa, os Métodos Determinísticos também passam por essas dificuldades, como por exemplo, o valor do fator de penalidade em problemas com restrições e os valores utilizados em parâmetros associados ao critério de convergência utilizado.

A análise de como tais parâmetros afetam a qualidade da solução dos problemas de otimização deve ser realizada para se definir faixas de utilização, tornando o procedimento mais robusto. Todavia, é claro que essa análise fica muito limitada devido à quantidade de parâmetros e à variedade dos problemas analisados.

Apesar da existência de inúmeros trabalhos referentes à otimização multi-objetivo utilizando Métodos Não Determinísticos, são poucos os que dão ênfase ao estudo da sensibilidade ou influência de seus parâmetros na qualidade da solução obtida (BABU; ANGIRA, 2001; SILVA, 2004; COELHO, 2004; BABU; ANBARASU, 2005).

Neste contexto, esta seção procura estudar a influência:

- do operador de Exploração das Vizinhanças acoplado ao Algoritmo de Evolução

Diferencial, comparado ao algoritmo evolutivo clássico NSGA II;

- de alguns parâmetros do MODE em problemas clássicos, escolhidos aqui por apresentar certas características, que do ponto de vista numérico, são muito relevantes.

As próximas seções apresentam estudos sobre cada um dos aspectos acima anunciados.

8.3.1 Análise da Incorporação do Operador de Exploração das Vizinhanças no MODE

Sabe-se que o uso do Operador de Exploração das Vizinhanças apresentado na seção 7.2.7 faz com que o número de avaliações da função objetivo do algoritmo MODE praticamente dobre quando comparado com o NSGA II. Deve ser ressaltado que esse aumento se deve à necessidade de avaliar a função objetivo do vizinho gerado.

Com o intuito de se avaliar o desempenho do algoritmo proposto (MODE) com e sem o operador de exploração de vizinhanças (Evolução Diferencial - ED), será utilizado o problema ZDT1 estudado na seção 8.1:

$$\text{ZDT1} = \begin{cases} f_1(x_1) = x_1 \\ g(x_2, \dots, x_m) = 1 + \frac{9}{m-1} \sum_{i=2}^m x_i \\ h(f_1(x_1), g(x_2, \dots, x_m)) = 1 - \sqrt{f_1(x_1)/g(x_2, \dots, x_m)} \end{cases} \quad (8.23)$$

O domínio das variáveis de projeto é definido como sendo o intervalo (0,1). A região ótima de Pareto corresponde a $x_1 \in (0, 1)$ e $x_i = 0$ ($i = 2, \dots, m = 30$).

Para avaliações da incorporação do mecanismo de Exploração das Vizinhanças será feita a análise da influência do número de indivíduos da população e do número de gerações no valor das métricas de convergência.

Os parâmetros utilizados pelo MODE, NSGA II e pelo ED, em ambos os estudos, são apresentados na Tab. 8.39.

◦ Influência da Variação do Número de Indivíduos na População

A Tabela 8.40 apresenta a comparação entre o número de avaliações da função objetivo para três algoritmos: NSGA II, ED e o MODE.

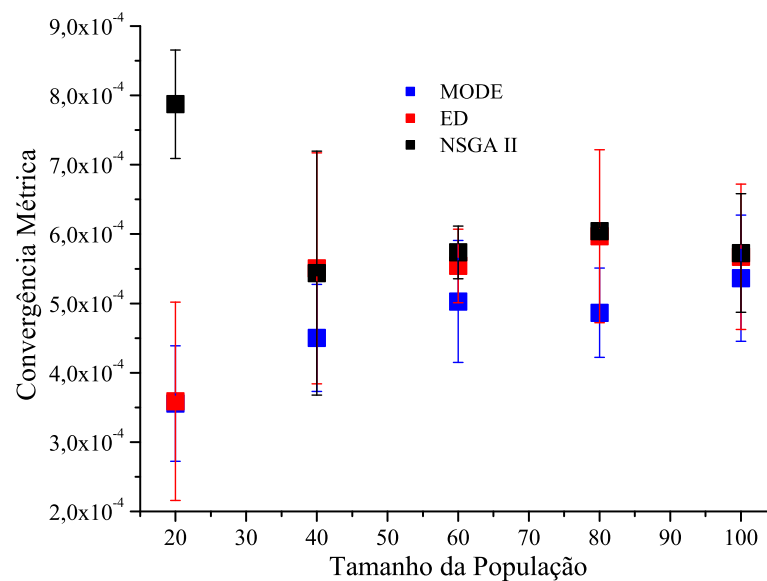
Tabela 8.39: Parâmetros utilizados pelos algoritmos MODE, ED, NSGA II.

Parâmetro	ALGORITMO		
	NSGA II	ED	MODE
N_{gen}	VARIÁVEL	VARIÁVEL	VARIÁVEL
N	VARIÁVEL	VARIÁVEL	VARIÁVEL
p_c	0,58	0,58	0,58
p_m	0,05	-	-
F	-	0,60	0,60
R	-	-	3
r	-	-	0,8

Tabela 8.40: Número de avaliações da função objetivo para o NSGA II, MODE e para o ED para 200 gerações.

Tamanho da População	ALGORITMO		
	NSGA II	ED	MODE
20	4020	4020	8020
40	8040	8040	16040
60	12060	12060	24060
80	16080	16080	32080
100	20100	20100	40100

A Figura 8.54 apresenta a média e o desvio padrão da Convergência Métrica para 200 gerações.

**Figura 8.54:** Convergência métrica $\times$ tamanho da população para 200 gerações.

Já na Figura 8.55 é apresentada a média e o desvio padrão da Diversidade Métrica para 200 gerações.

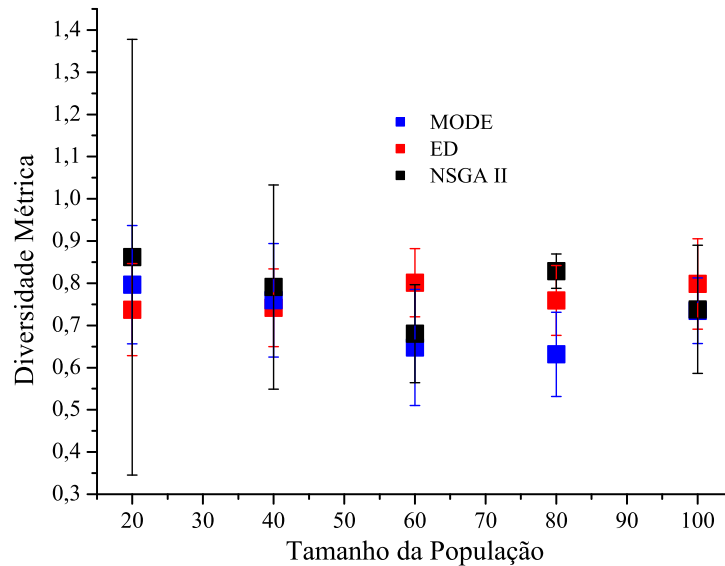


Figura 8.55: Diversidade métrica × tamanho da população para 200 gerações.

A análise da Tab. 8.40 mostra que o número de avaliações da função objetivo é muito maior para o MODE, com relação ao número de indivíduos da população, quando se compara com os dois outros algoritmos. Contudo, nas Figs. 8.54 e 8.55 observa-se que o desempenho do MODE, com relação às métricas de convergência e diversidade, é superior aos algoritmos NSGA II e ED. Além disso, tomando como base o número de indivíduos que sempre é utilizado na literatura para o problema ZDT1 (100 indivíduos) (ZITZLER et al., 2000; DEB, 2001; CASTRO, 2001), verifica-se na Tab. 8.40 que o MODE apresenta um número menor de avaliações da função objetivo para 20 e 40 indivíduos e valores melhores para as métricas, se comparado ao algoritmo NSGA II.

o Influência da Variação do Número de Gerações

Para avaliar a influência do número de gerações são considerados 100 indivíduos na população. Os outros parâmetros do algoritmo são os mesmos apresentados na Tab. 8.39.

A Figura 8.56 apresenta a média e o desvio padrão da Convergência Métrica para 100 indivíduos. Já na Figura 8.57 é apresentada a média e o desvio padrão da Diversidade Métrica para 100 indivíduos.

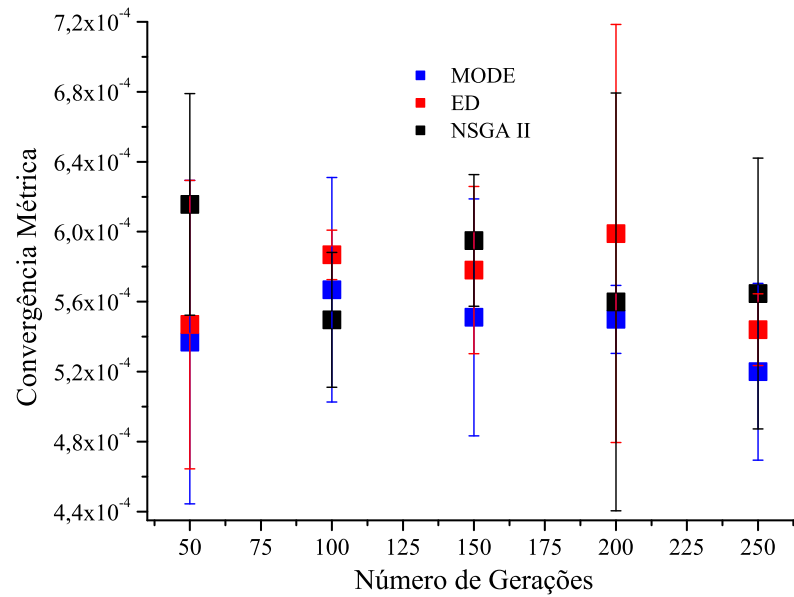


Figura 8.56: Convergência métrica $\times$ tamanho da população para 100 indivíduos.

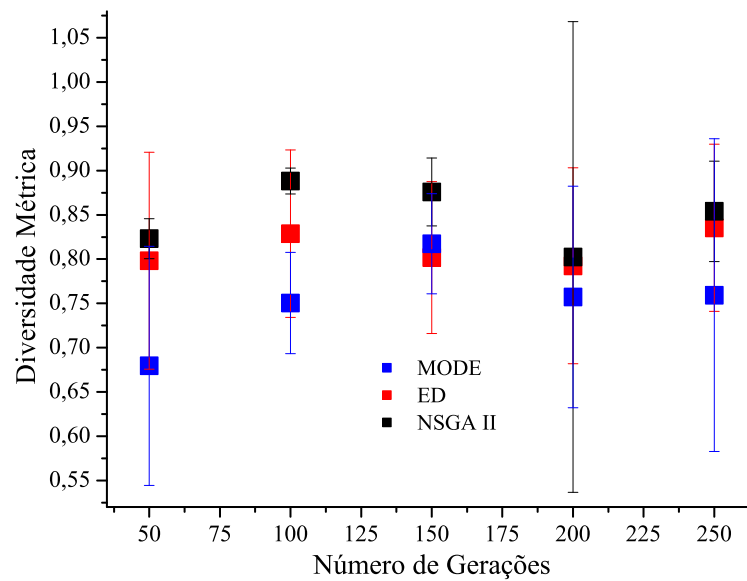


Figura 8.57: Diversidade métrica $\times$ tamanho da população para 100 indivíduos.

A Tabela 8.41 apresenta uma comparação entre o número de gerações para três algoritmos: NSGA II, ED e o MODE. É possível observar que, devido à incorporação da estratégia de Exploração das Vizinhanças no algoritmo MODE, este apresenta um número de avaliações da função objetivo muito maior quando comparado com o NSGA II e o ED. As Figuras 8.56 e 8.57 mostram que o desempenho do MODE, com relação às métricas de convergência e diversidade, é superior aos algoritmos NSGA II e ED. Além disso, ao se tomar como base os parâmetros que são sempre utilizados na literatura para

Tabela 8.41: Número de avaliações da função objetivo para o NSGA II, MODE e o ED para 100 indivíduos.

Número de Gerações	ALGORITMO		
	NSGA II	ED	MODE
50	5100	5100	10100
100	10100	10100	20100
150	15100	15100	30100
200	20100	20100	40100
250	25100	25100	50100

o problema ZDT1 (100 indivíduos e 250 gerações, totalizando 25100 avaliações da função objetivo) (ZITZLER et al., 2000; DEB, 2001; CASTRO, 2001), e considerando a variação do tamanho da população com relação ao número de avaliações da função objetivo apresentado na Tab. 8.40, é possível observar que, para uma população com até 60 indivíduos, o MODE apresenta um número menor de avaliações da função objetivo quando comparado com os algoritmos NSGA II e o ED.

Neste sentido, conclui-se com essa análise que é possível melhorar os valores das métricas de convergência e de diversidade através da redução do número de avaliações da função objetivo pela utilização de um número menor de indivíduos na população, ou através da diminuição do número de gerações usando o algoritmo MODE. Tal aspecto é importantíssimo, pois é este que justifica o uso do operador de Exploração das Vizinhanças acoplado ao algoritmo de Evolução Diferencial.

8.3.2 Análise de Sensibilidade de Parâmetros do MODE

O MODE possui seis parâmetros a serem escolhidos: o número de gerações (N_{gen}), o tamanho da população (N), a probabilidade de cruzamento (p_c), a taxa de perturbação (F), o número de pseudo-curvas (R) e a taxa de redução (r).

A seguir é apresentada uma análise da influência de alguns desses parâmetros quando aplicados aos problemas ZDT1, ZDT2 e ZDT3, descritos e estudados anteriormente.

• Influência da Taxa de Redução - r

Parâmetros utilizados pelo MODE: $N_{gen}=150$, $N=50$, $p_c=0,8$, $F=0,8$, $R=10$ e $r=\{0,1 \ 0,2 \ 0,3 \ 0,4 \ 0,5 \ 0,6 \ 0,7 \ 0,8 \ 0,9\}$.

A Figura 8.58 apresenta a média e o desvio padrão da Convergência Métrica.

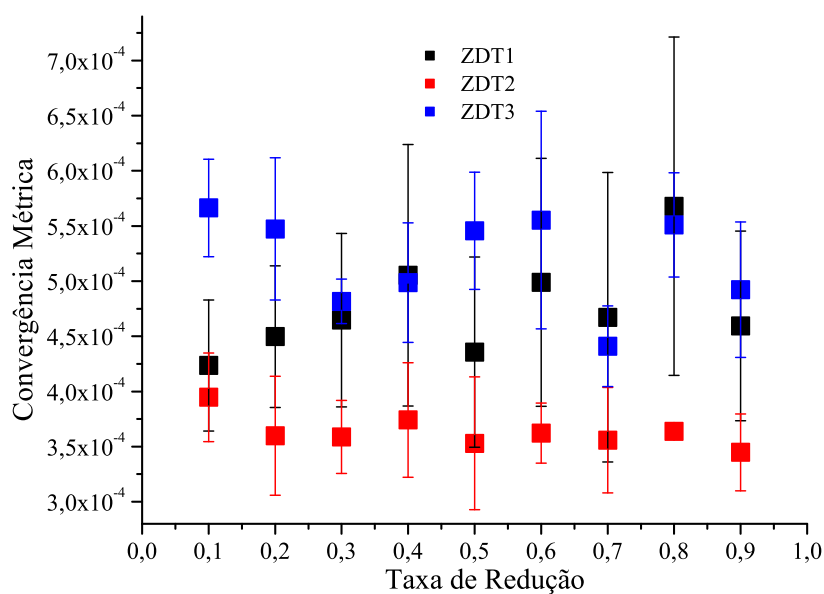


Figura 8.58: Convergência métrica $\times$ taxa de redução.

A Figura 8.59 apresenta a média e o desvio padrão da Distância Geracional.

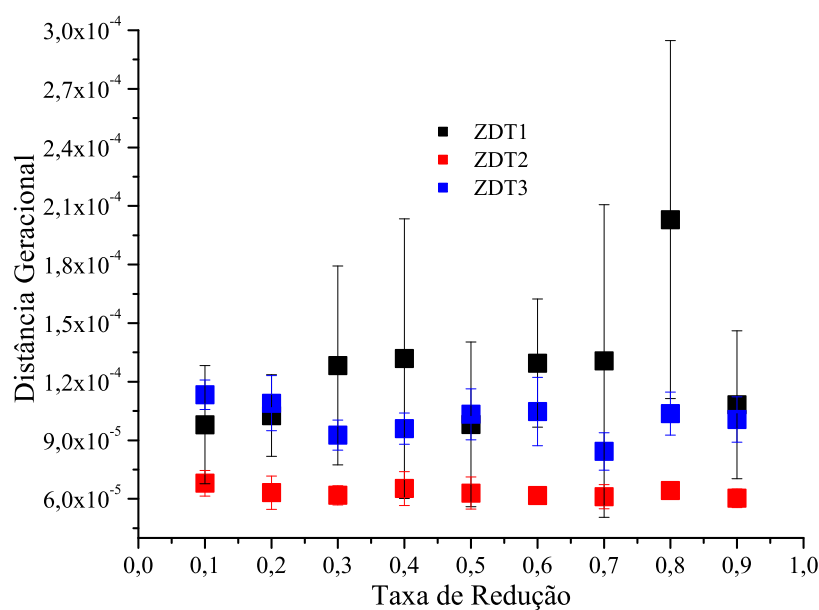


Figura 8.59: Distância geracional $\times$ taxa de redução.

A Figura 8.60 apresenta a média e o desvio padrão da Diversidade Métrica.

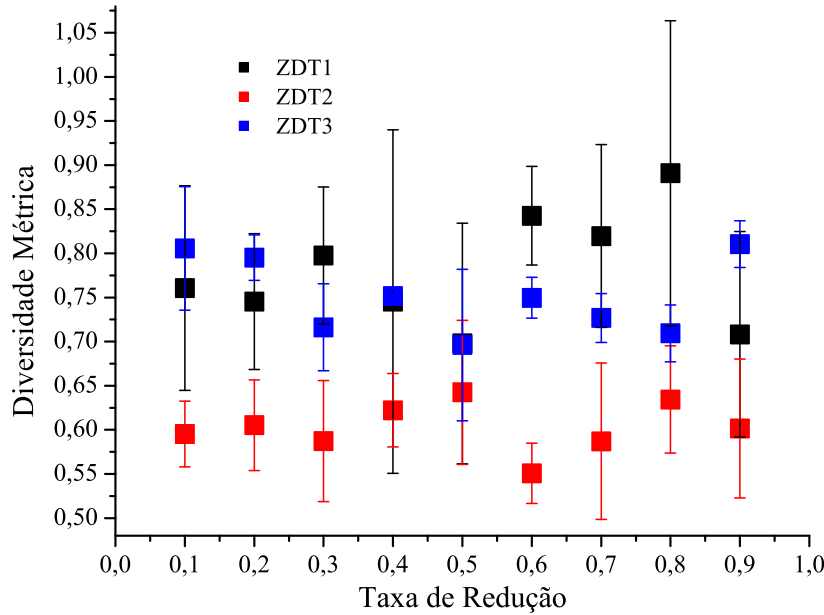


Figura 8.60: Diversidade métrica $\times$ taxa de redução.

De modo geral, os resultados obtidos são satisfatórios na medida que, em comparação com o NSGA II (usando 100 indivíduos na população, 250 gerações, probabilidade de cruzamento igual a 0,9 e probabilidade de mutação igual a $1/(\text{número de variáveis de projeto})$), a convergência métrica é da ordem de 8×10^{-4} e a diversidade métrica da ordem de 0,46 (DEB, 2001). Entretanto, como se pode observar nas Figs. 8.58 a 8.60, não é possível escolher um bom valor para r , para todos os problemas analisados. Para a função ZDT1, o melhor resultado com relação à Convergência Métrica e a Distância Geracional é 0,1, mas com relação à Diversidade Métrica é 0,5. Já para as funções ZDT2 e ZDT3, 0,9 e 0,7 são valores considerados bons para a Convergência Métrica e a Distância Geracional, respectivamente.

Resumidamente, tem-se que não é possível uma conclusão definitiva sobre a influência desse parâmetro nos resultados das métricas utilizadas, sendo dessa forma temerária a atribuição de um valor ou de uma faixa de operação para todos os três problemas. Assim, faz-se necessário um estudo mais aprofundado da sensibilidade das métricas com relação a esse parâmetro.

• Influência da Probabilidade de Cruzamento - ρ_c

Os parâmetros utilizados pelo MODE são: $N_{gen}=150$, $N=50$, $r=0,9$, $F=0,8$, $R=10$ e $p_c=\{0,1 \ 0,2 \ 0,3 \ 0,4 \ 0,5 \ 0,6 \ 0,7 \ 0,8 \ 0,9\}$.

A Figura 8.61 apresenta a média e o desvio padrão da Convergência Métrica.

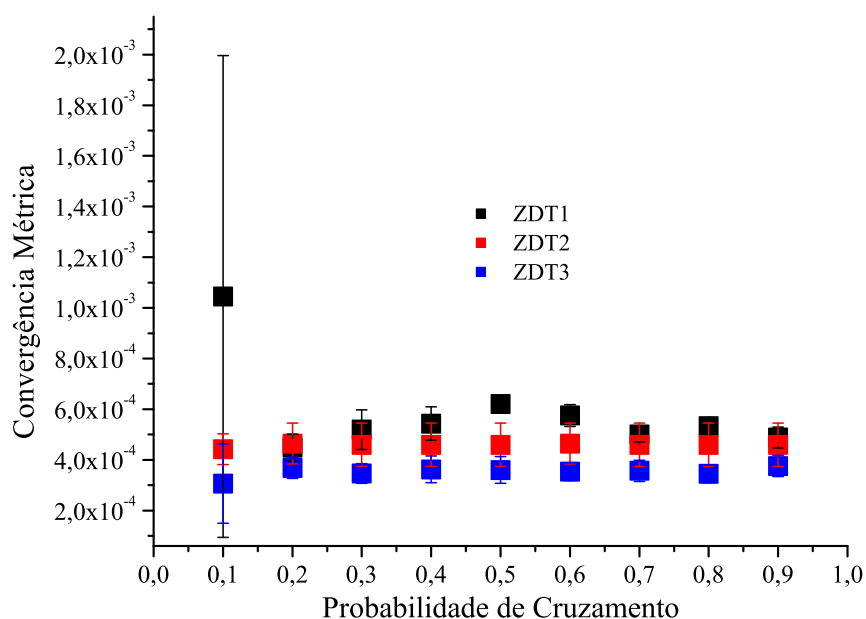


Figura 8.61: Convergência métrica $\times$ probabilidade de cruzamento.

A Figura 8.62 apresenta a média e o desvio padrão da Distância Geracional.

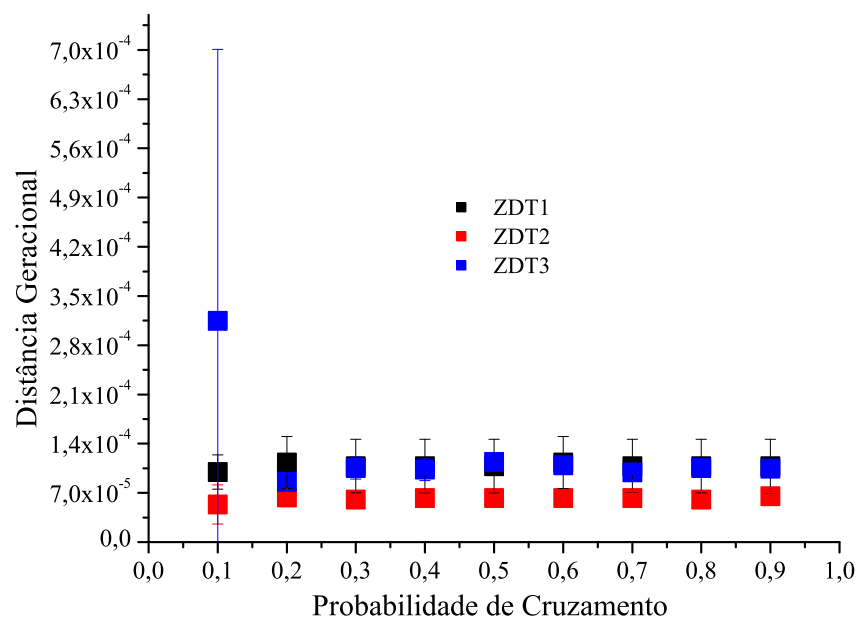


Figura 8.62: Distância geracional $\times$ probabilidade de cruzamento.

A Figura 8.63 apresenta a média e o desvio padrão da Diversidade Métrica.

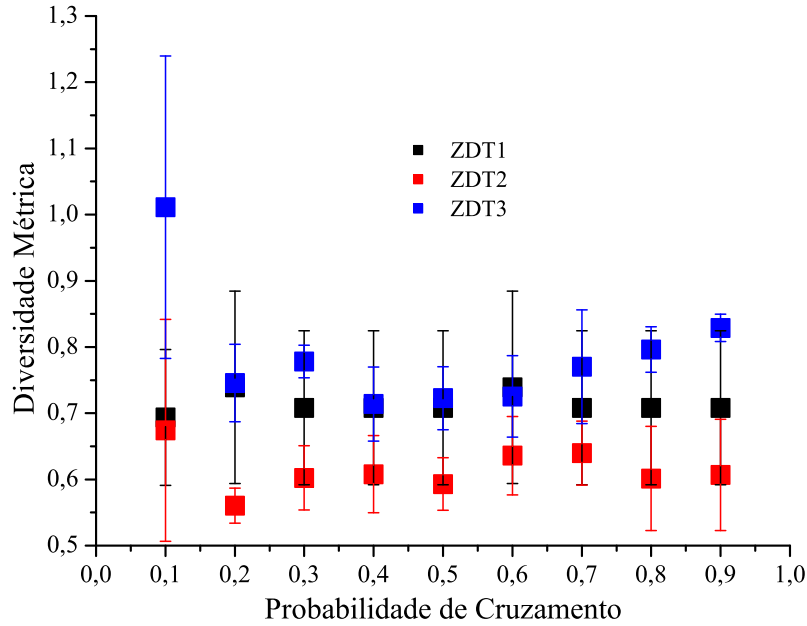


Figura 8.63: Diversidade métrica $\times$ probabilidade de cruzamento.

Como se pode observar nas Figs. 8.61 a 8.63, os resultados obtidos são satisfatórios quando comparados aos do NSGA II ($\Upsilon=8 \times 10^{-4}$ e $\Delta=0,46$) apresentados por Deb (2001). Com relação à Convergência Métrica e a Distância Geracional, valores de ρ_c maiores que 0,1 resultam em valores melhores dessas métricas para os casos estudados. Já para a Diversidade Métrica, os resultados obtidos pelo MODE não são superiores aos reportados por Deb (2001). Individualmente, 0,1, 0,2 e 0,4 são considerados como valores bons de ρ_c para as funções ZDT1, ZDT2 e ZDT3, respectivamente.

De maneira geral, recomenda-se usar, para o novo estudo de caso, valores de ρ_c acima de 0,1.

• Influência da Taxa de Perturbação - F

Parâmetros utilizados pelo MODE são: $N_{gen}=150$, $N=50$, $r=0,9$, $p_c=0,8$, $R=10$ e $F=\{0,2 \ 0,4 \ 0,6 \ 0,8 \ 1,0 \ 1,2 \ 1,4 \ 1,6 \ 1,8 \ 2,0\}$.

As Figuras 8.64, 8.65 e 8.66 apresentam a média e o desvio padrão da Convergência Métrica, da Distância Geracional e da Diversidade Métrica, respectivamente.

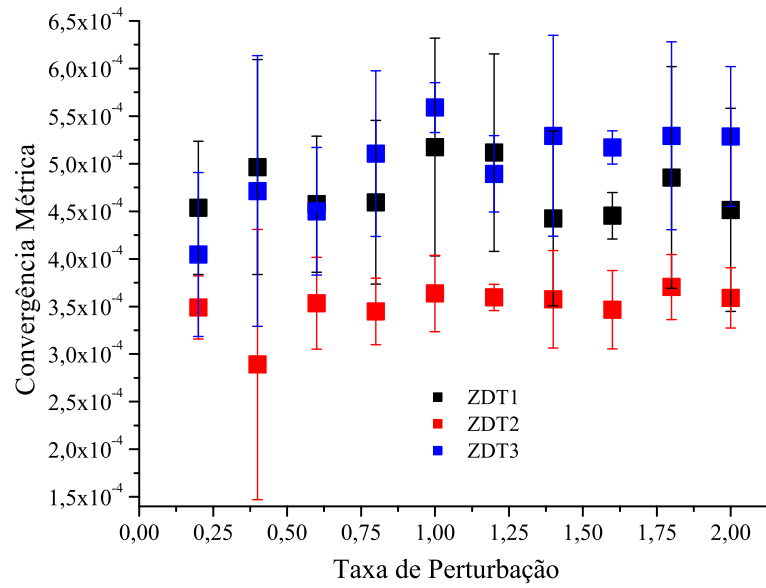


Figura 8.64: Convergência métrica $\times$ taxa de perturbação.

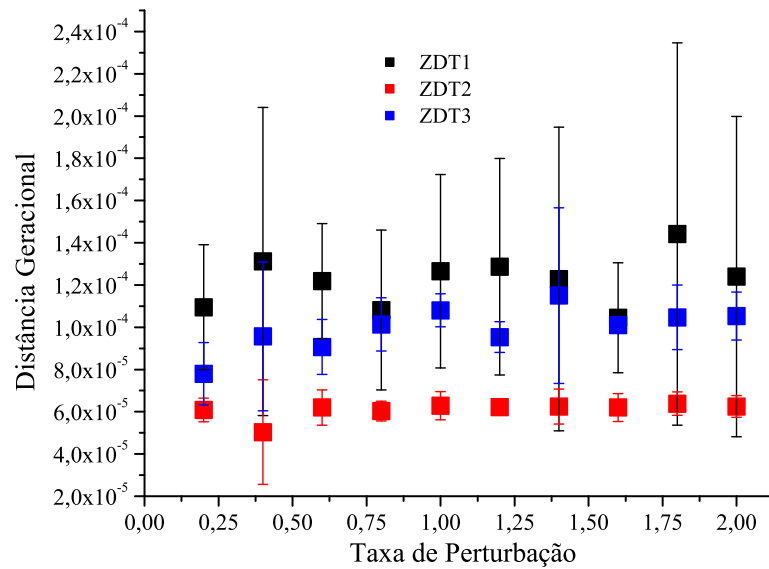


Figura 8.65: Distância geracional $\times$ taxa de perturbação.

Para cada uma das funções ZDT foram obtidos diferentes valores de F que conduzem a bons valores de métricas. Por exemplo, 1,4 e 1,6 são valores bons para a Convergência Métrica e a Diversidade Métrica, para a função ZDT1, mas não são os melhores valores de F para as funções ZDT2 e ZDT3. Apesar da variação F , pode-se observar que qualquer valor entre 0,2 e 2 é uma boa escolha para esse parâmetro, exceto para F igual a 0,2, no caso da Diversidade Métrica, onde um maior desvio padrão é observado.

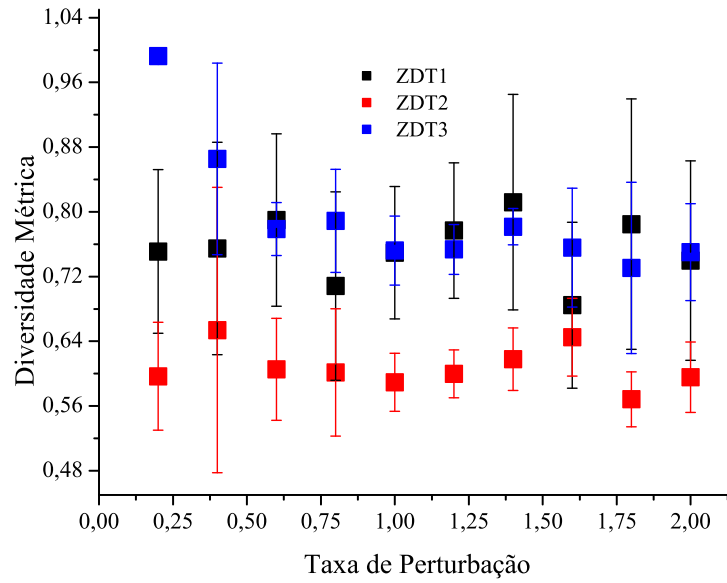


Figura 8.66: Diversidade métrica $\times$ taxa de perturbação.

• Influência do Tamanho da População - N

Os parâmetros utilizados pelo MODE para esse estudo são: $N_{gen}=150$, $F=0,8$, $r=0,9$, $p_c=0,8$, $R=10$ e $N=\{20\ 40\ 60\ 80\ 100\ 120\ 140\ 160\ 180\}$.

A média e o desvio padrão da Convergência Métrica é mostrada na Fig. 8.67.

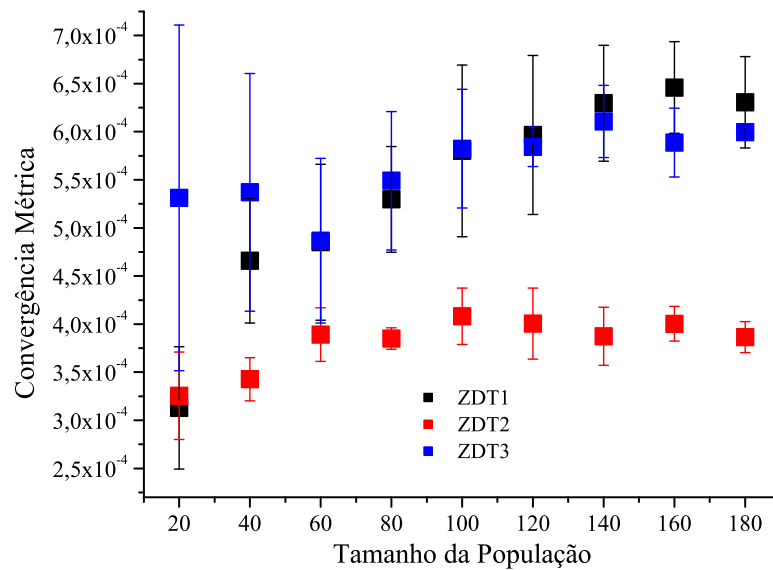


Figura 8.67: Convergência métrica $\times$ tamanho da população.

A média e o desvio padrão da Distância Geracional é apresentada na Fig. 8.68.

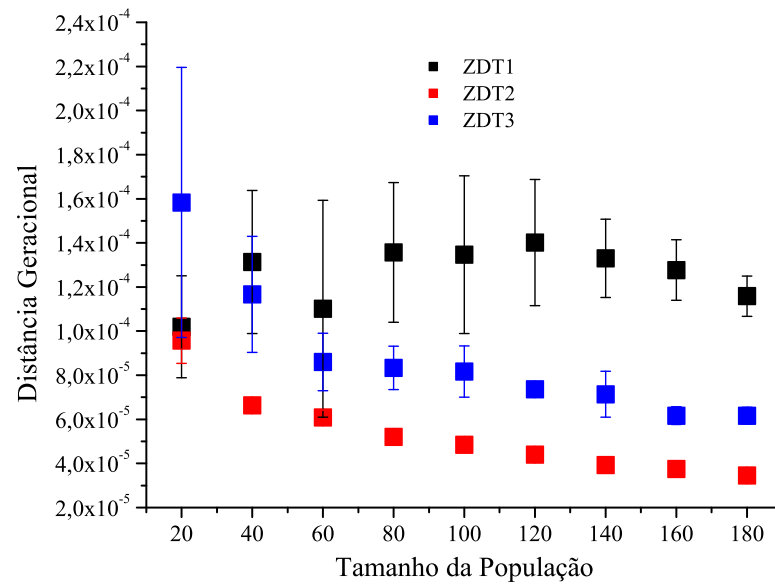


Figura 8.68: Distância geracional $\times$ tamanho da população.

A Figura 8.69 apresenta a média e o desvio padrão da Diversidade Métrica.

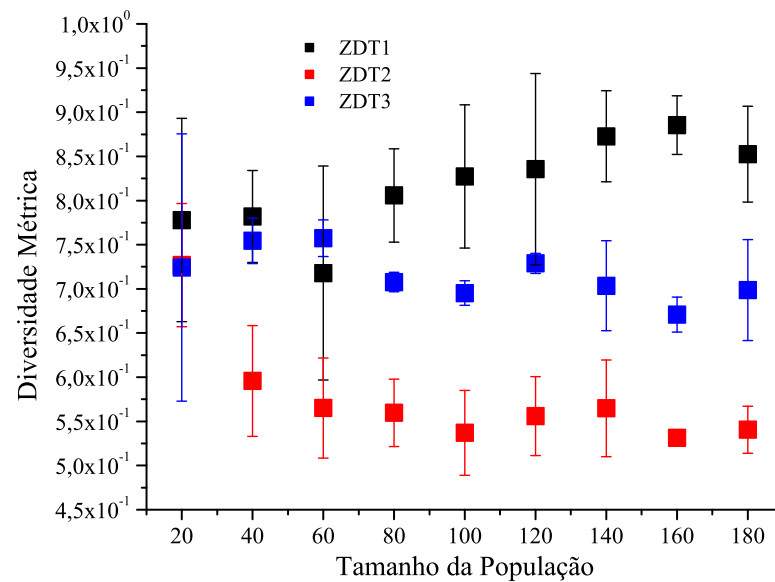


Figura 8.69: Diversidade métrica $\times$ tamanho da população.

Pode-se observar nas Figs. 8.68 e 8.69 que o aumento do número de indivíduos na população faz com que essas duas métricas melhorem seus valores para os problemas ZDT2 e ZDT3. Contudo, para o problema ZDT1, em todas as figuras, e para os