MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-14795-TDI/1238

DESENVOLVIMENTO DE VERSÕES APRIMORADAS, HÍBRIDAS, PARALELA E MULTIOBJETIVO DO MÉTODO DA OTIMIZAÇÃO EXTREMA GENERALIZADA E SUA APLICAÇÃO NO PROJETO DE SISTEMAS ESPACIAIS

Roberto Luiz Galski

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Fernando Manuel Ramos e Fabiano Luis de Sousa, aprovada em 22 de setembro de 2006.

INPE
São José dos Campos
2007

Publicado por: **esta página é responsabilidade do SID**

Instituto Nacional de Pesquisas Espaciais (INPE)
Gabinete do Diretor – (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 – CEP 12.245-970
São José dos Campos – SP – Brasil
Tel.: (012) 3945-6911
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

Solicita-se intercâmbio
We ask for exchange

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-14795-TDI/1238

DESENVOLVIMENTO DE VERSÕES APRIMORADAS, HÍBRIDAS, PARALELA E MULTIOBJETIVO DO MÉTODO DA OTIMIZAÇÃO EXTREMA GENERALIZADA E SUA APLICAÇÃO NO PROJETO DE SISTEMAS ESPACIAIS

Roberto Luiz Galski

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Fernando Manuel Ramos e Fabiano Luis de Sousa, aprovada em 22 de setembro de 2006.

INPE
São José dos Campos
2007

519.863

Galski, R. L.
Desenvolvimento de versões aprimoradas, híbridas, paralela e multiobjetivo do método da otimização extrema generalizada e sua aplicação no projeto de sistemas espaciais / Roberto Luiz Galski. - São José dos Campos: INPE, 2006.
279 p. ; (INPE-14795-TDI/1238)

1. Otimização. 2. Otimização global.
3. Algoritmos evolutivos. 4. Algoritmos estocásticos.
5. Paralelização. 6. Hibridização. I. Título.

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

Computação Aplicada

Dr. Horacio Hideki Yanasse

Presidente / INPE / SJCampos - SP

Dr. Fernando Manuel Ramos

Orientador(a) / INPE / SJCampos - SP

Dr. Fabiano Luis de Sousa

Orientador(a) / INPE / SJCampos - SP

Dr. Stephan Stephany

Membro da Banca / INPE / SJCampos - SP

Dr. José Demisio Simões da Silva

Membro da Banca / INPE / SJCampos - SP

Dr. Fernando José Von Zuben

Convidado(a) / UNICAMP / Campinas - SP

Dr. Hélcio Rangel Barreto Orlande

Convidado(a) / UFRJ / Rio de Janeiro - RJ

Aluno (a): **Roberto Luiz Galski**

São José dos Campos, 22 de setembro de 2006

À família.

AGRADECIMENTOS

Aos orientadores, Dr. Fernando Manuel Ramos e Dr. Fabiano Luis de Sousa, que orientaram este trabalho demonstrando sempre transparência e motivação, além da competência e do conhecimento necessários. Além disso, agradeço aos dois pelo sentimento de coleguismo que me passaram durante a elaboração desta Tese.

Ao Dr. Issamu Muraoka, cuja colaboração preciosa permitiu a realização das aplicações envolvendo o software PCTER.

Ao Dr. Pawel Rozenfeld, chefe do Centro de Rastreio e Controle de Satélites (CRC), pelo apoio e boa vontade demonstrada, permitindo assim, o desenvolvimento deste trabalho.

A todos, familiares, professores e colegas que, direta ou indiretamente, contribuíram para a realização deste objetivo.

RESUMO

Em Sousa (2002), o algoritmo *Generalized Extremal Optimization* (GEO) foi desenvolvido. O GEO é uma generalização que estendeu a aplicabilidade da meta-heurística *Extremal Optimization* (EO) (Boettcher e Percus, 2001) virtualmente a qualquer problema de otimização. O GEO, assim como o EO inspira-se no modelo simplificado do processo evolutivo desenvolvido por Bak e Sneppen (1993). Como acontece com qualquer método novo, uma quantidade significativa de estudo deve ser efetuada, de modo a desenvolver o potencial presente no método original. Esta Tese de Doutorado tem como principal objetivo explorar as potencialidades do GEO, aprofundar os estudos do mesmo visando obter versões mais eficientes e, ao mesmo tempo, preocupando-se em expandir sua aplicabilidade. Dentro desse contexto, é efetuado o desenvolvimento de uma versão paralelizada do algoritmo GEO, denominada GEOPAR-1 e a análise de seu desempenho. Almeja-se, com isso, comprovar a eficiência do GEO como algoritmo e permitir sua aplicação no projeto otimizado de sistemas espaciais complexos onde o alto desempenho computacional é uma necessidade. Atualmente, é comum o emprego de técnicas de hibridização, para melhorar as características dos algoritmos de otimização. Nesta Tese, tais técnicas são utilizadas no desenvolvimento de versões hibridizadas do algoritmo GEO, onde se buscou incorporar ao mesmo, características presentes em outros algoritmos de modo a aumentar sua eficiência quando aplicado a problemas em geral ou, ao menos, aumentar sua eficiência em aplicações específicas. Um outro campo de estudo é o da otimização multiobjetivo. Neste campo, um dos resultados dos estudos efetuados é o desenvolvimento de um novo algoritmo de otimização multiobjetivo, chamado M-GEO. A fim de avaliar seu desempenho, todos os algoritmos desenvolvidos nesta Tese são testados com diversas funções testes comuns na literatura da área. Além disso, a fim de comprovar a real capacidade de otimização das versões desenvolvidas, algumas são utilizadas para a otimização de sistemas espaciais reais: o projeto térmico dos radiadores da Plataforma Multimissão (PMM) do INPE e a obtenção da configuração de uma constelação de satélites de sensoriamento remoto.

DEVELOPMENT OF IMPROVED, HYBRID, PARALLEL AND MULTIOBJECTIVE VERSIONS OF THE GENERALIZED EXTREMAL OPTIMIZATION METHOD AND ITS APPLICATION TO THE DESIGN OF SPATIAL SYSTEMS

ABSTRACT

In Sousa (2002), the Generalized Extremal Optimization (GEO) algorithm was developed. GEO is a generalization that has extended the applicability of the *Extremal Optimization* (EO) meta-heuristic (Boettcher and Percus, 2001) to virtually any optimization problem. Both GEO and EO are inspired on the simplified model of the evolutive process developed by Bak and Sneppen (1993). As it happens with most new method, a reasonable amount of study must be done, in order to fully develop the potential of the original method. This Doctorate thesis has as its main objective to explore GEO's potential, performing deep studies aiming at obtaining more efficient versions and, at the same time, aiming at expanding its applicability. Inside this context, it is done the development of a parallel version of the GEO algorithm, named GEOPAR-1 and a performance analysis. The objective is to validate GEO's efficiency as algorithm and to allow its application in the optimized design of complex spatial systems where computational' high performance is a necessity. Nowadays, it is common the usage of hybridization techniques to improve the optimization algorithms characteristics. In this Thesis, such techniques are applied to the development of hybrid versions of the GEO algorithm, where characteristics of other algorithms are incorporate in GEO in order to improve its efficiency when applied to problems in general or, at least, to improve its efficiency when applied to specific problems. Another field of study is multiobjective optimization. In this field, one result from the studies that were done is the development of a new multiobjective optimization algorithm, called M-GEO. In order to evaluate its performances, all algorithms developed in this Thesis are tested with several test functions commonly used in the optimization field. Moreover, in order to verify the real capacity of optimization of the developed versions, some of them are used to the optimization of real spatial systems: the thermal design of the radiators of INPE's Multimission Platform (In Portuguese, Plataforma Multimissão - PMM) and in the design of the initial configuration of a remote sensing satellite constellation.

SUMÁRIO

Pág.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Latinos

ALE - Número aleatório com distribuição uniforme no intervalo [0,1)

b - base, parâmetro dos algoritmos GEO_4 e GEO_{var4}

b_{MAX} - valor máximo de b

b_{MIN} - valor mínimo de b

b_{ref} - valor b na iteração anterior

$b_{i,q}$ - elemento (i,q,j) da matriz binária **b** de GEO_3, onde $i \in \{1,2,...,nbc\}$, $q \in \{1,2,...,2^{nbc}\}$

$b_{i,q,j}$ - elemento (i,q) da matriz binária **b** de GEO_{var3}, onde $i \in \{1,2,...,nbcv_j\}$, $q \in \{1,2,...,2^{nbcv_j}\}$, e $j \in \{1,2,...,N\}$

beta - ângulo entre a direção do sol e a projeção desta no plano orbital (°)

bcv - número de bits congelados por variável de projeto

c - valor binário sorteado com distribuição uniforme

C - cadeia binária usada para codificar **X**

$\mathbf{C_{esc}}$ - cadeia binária resultante após mutação simultânea dos bits i_{esc_j} , $j \in \{1,2,...,N\}$

$\mathbf{C_{iesc}}$ - cadeia binária resultante após mutação do bit i_{esc}

D - parte inteira da divisão L/NP

e - base natural dos logarítmos

e_n - n-ésima espécie do modelo de Bak-Sneppen

E - contador do número de iterações sucessivas de GEO_2 ou GEO_{var2} nas quais não houve melhora do valor de F(**X**) ou F($\mathbf{X_{melhor}}$)

E_{lim} - valor limite de E e parâmetro de GEO_2 e GEO_{var2}

F - variável auxiliar utilizada na conversão de binário para decimal

F(**X**) - valor da função objetivo

F(**X**) - vetor contendo os valores de NFOBJ funções objetivo

F($\mathbf{X_{melhor}}$) - valor da função objetivo para $\mathbf{X_{melhor}}$

$F(\mathbf{X})_{anterior}$ - valor de F(**X**) da iteração anterior

$F_{NFOBJ}(\mathbf{X})$ - valor da NFOBJ-ésima função objetivo

$g_j(\mathbf{X})$ - j-ésima restrição de desigualdade

$g_{q,ib}$ - elemento (q,ib) da matriz binária **G** de GEO_3, onde $q \in \{1,2,...,2^{nbc}\}$, $ib \in \{1,2,...,L\}$

$g_{j,q,ib}$ - elemento (j,q,ib) da matriz binária **G** de GEO_{var3}, onde $j \in \{1,2,...,N\}$, $q \in \{1,2,...,2^{nbc}\}$, $ib \in \{1,2,...,L\}$

$\mathbf{G}_{i,j}$ - condutância condutiva entre o painel i e o painel j (W/K)

$h_k(\mathbf{X})$ - k-ésima restrição de igualdade

$\mathbf{H}(.)$ - matriz do Hessiano de (.)

i_{esc} - posição, dentro da cadeia binária **C**, do bit escolhido para sofrer mutação

i_{esc_j} - posição, dentro da cadeia binária C_j, do bit escolhido para sofrer mutação

Ic - posição, dentro do vetor $\mathbf{F}(\mathbf{X})$, da função escolhida para calcular os valores de adaptação dos indivíduos da iteração atual do M-GEO

I_A - índice de adaptação da espécie

I_j - inclinação do satélite j (°)

ibc - lista com a posição dos bits congelados da cadeia binária **C**

ibc_{indx} - posição, dentro da cadeia binária **C**, do indx-ésimo bit congelado

indx - índice dos bits congelados, $indx \in \{1,2,...,nbc\}$

l - número de bits que codifica todo e qualquer X_j

l_{MIN} - valor mínimo de l

l_j - número de bits que codifica X_j

$l_{j_{MIN}}$ - valor mínimo de l_j

L - número total de bits usados na codificação das variáveis de projeto

m - tamanho da mutação no híbrido GEO_{var} + EE

M - tamanho da mensagem trocada entre dois nós da rede (bytes)

M_i - anomalia média do satélite i (°)

M_j-M_i - Defasagem em anomalia média, do satélite j em relação ao satélite i (°)

n - número do estágio

n_{MAX} - número máximo de estágios

na - número de iterações da fase de aperfeiçoamento

nd - número de iterações da fase de diversificação

nbc - número de bits congelados

$nbcv_j$ - número de bits mais significativos congelados na variável j

NFOBJ - número de funções objetivo

N - número de variáveis de projeto

NAF - número de avaliações de $F(\mathbf{X})$

NC - número de ciclos

NI - número de iterações

NP - número de processadores

NAF_{MAX} - número total de avaliações de F(**X**)

P(k) - probabilidade do bit de índice de adaptação k sofrer mutação

P_k - probabilidade do bit de índice de adaptação k sofrer mutação

P(s) - probabilidade de ocorrência da avalanche de tamanho s

$P_{\{.\}}$ - probabilidade de ocorrência do evento {.}

q - fator constante de redução de temperatura a cada estágio

q_{ALE} - valor de q obtido por sorteio com distribuição uniforme

q_{melhor} - valor de q para o qual F($\mathbf{X_{melhor}}$) foi encontrado

$\mathbf{q_i}_{1:5}$ - geração interna de calor nos painéis 1 a 5, respectivamente (W)

$\mathbf{q_s}_{1:5;8;9}$ - fluxo de radiação solar nos painéis 1 a 5, 8, e 9, respectivamente; (W/m^2)

$\mathbf{q_t}_{1:5;8;9}$ - fluxo de radiação terrestre nos painéis 1 a 5, 8, e 9, respectivamente; (W/m^2)

r_j - resolução resultante para Xj, como fração de X_{MAX_j}-X_{MIN_j}

$r_{j_{MAX}}$ - valor máximo de r_j

rbc - número de bits restantes para serem congelados

R - número de reinicializações de GEO_2 ou GEO_{var2}

R - variável auxiliar utilizada na conversão de binário para decimal

R - resto da divisão L/NP

$\mathbf{R}_{i,j}$ - condutância radiativa entre o painel i e o painel j (W/K^4)

s - tamanho da avalanche

s - sinal (+ ou -) da mutação no híbrido GEO_{var} + EE

S(NP) - *speed-up* de uma execução paralela com NP processadores

T - parâmetro temperatura (K)

T_0 - valor inicial de T (K)

T_{MAX} - valor máximo de T (K)

T_n - valor de T durante o estágio n (K)

T_∞ - temperatura de um corpo negro no espaço distante (K)

$\mathbf{T}_{1:6}$ - temperaturas nos painéis 1 a 6, respectivamente (°C) ou (K)

$\mathbf{T}_7$ - temperatura nos elementos da propulsão (tanque, tubo, válvula); (°C) ou (K)

$\mathbf{T}_{8:9}$ - temperaturas nos painéis solares; (K)

$\mathbf{T_A}$ - vetor contendo as temperaturas alvo (°C)

$\mathbf{T_{CF}}$ - vetor contendo as temperaturas do Caso Frio (°C)

$\mathbf{T_{CQ}}$ - vetor contendo as temperaturas do Caso Quente (°C)

$\mathbf{T_{CQ1}}$ - vetor contendo as temperaturas do Caso Quente 1 (°C)

$\mathbf{T_{CQ2}}$ - vetor contendo as temperaturas do Caso Quente 2 (°C)

$\mathbf{T_{MAX}}$ - vetor contendo as temperaturas máximas para **T** (°C)

$\mathbf{T_{MIN}}$ - vetor contendo as temperaturas mínimas para **T** (°C)

X - vetor das variáveis de projeto

$\mathbf{X}_{1:5}$ - área do radiador nos painéis 1 a 5, respectivamente (m^2)

$\mathbf{X}_6$ - potência do aquecedor das baterias (painel 2) durante o CF (W)

$\mathbf{X}_7$ - potência do aquecedor dos componentes da propulsão durante o CF (W)

$\mathbf{X_b}$ - vetor resultante da mutação, em **X**, do b-ésimo bit usado na sua codificação

$\mathbf{X_{melhor}}$ - melhor **X** encontrado durante a busca

$\mathbf{X}_{MIN}$ - vetor com os limites inferiores das variáveis de projeto

$\mathbf{X_{MIN}}_6$ - potência mínima do aquecedor das baterias; (W)

$\mathbf{X_{MIN}}_7$ - potência mínima do aquecedor dos componentes da propulsão; (W)

$\mathbf{X}_{MAX}$ - vetor com os limites superiores das variáveis de projeto

$\mathbf{X_{MAX}}_{1:5}$ - áreas máximas dos radiadores 1 a 5, respectivamente; (m^2)

$\mathbf{X_{MAX}}_6$ - potência máxima do aquecedor das baterias; (W)

$\mathbf{X_{MAX}}_7$ - potência máxima do aquecedor dos componentes da propulsão; (W)

X* - valor ótimo de **X**

X_{aux} - variável auxiliar, escalar

X_j - j-ésimo elemento de **X**

X_{MAX_j} - valor máximo de X_j

X_{MIN_j} - valor mínimo de X_j

y - variável aleatória com distribuição gauss(μ,α)

y_{ant} - valor de y na iteração anterior

Gregos

α - taxa de aprendizado do EE e do híbrido GEO_{var} SA4 + EE

α - absortividade solar

α_{abs} - absortividade solar do absorvedor (painel 1);

α_{ps} - absortividade solar dos painéis solares;

α_{rad} - absortividade solar dos radiadores (painéis 2 a 5);

γ - parâmetro real não negativo

$d_{k,j}$ - delta de kronecker

e - emissividade no infravermelho

e(NP) - eficiência de uma execução paralela com NP processadores

e_{abs} - emissividade no infravermelho do absorvedor (painel 1);

e_j - precisão da variável X_j

ε_{max} - eficiência máxima teórica

e_{ps} - emissividade no infravermelho dos painéis solares;

e_{rad} - emissividade no infravermelho dos radiadores (painéis 2 a 5);

λ - número de indivíduos "filhos" no EE

μ - número de indivíduos "pais" no EE

μ - tendenciosidade do aprendizado do híbrido GEO_{var} SA4 + EE

ρ - escalar que define o tamanho da região de confiança

σ - desvio padrão da variável aleatória utilizada para mutar indivíduos no EE

τ - parâmetro real não negativo

τ_0 - valor inicial de τ

τ_{MAX} - valor máximo de τ

τ_n - valor de τ durante o estágio n

τ^* - τ ótimo

$\Delta F(\mathbf{X_i})$ - diferença entre $F(\mathbf{X_i})$ e o valor de referência $F(\mathbf{X_{melhor}})$

$\nabla(.)$ - gradiente de (.)

Ω - espaço amostral

Ω_i - ascensão reta do nodo ascendente do satélite i (°)

Ω_j-Ω_i - defasagem em ascensão reta, do satélite j em relação ao satélite i (°)

LISTA DE SIGLAS E ABREVIATURAS

AE	- Algoritmo Evolutivo
AEP	- Algoritmo Evolutivo Paralelo
AG	- Algoritmo Genético
AGs	- Algoritmos Genéticos
BFGS	- Broyden-Fletcher-Goldfarb-Shanno
CCGA	- *Cooperative Coevolutionary Genetic Algorithm*
CF	- Caso Frio
CFQ	- Caso Frio e Quente
CQ	- Caso Quente
DFP	- Davidon-Fletcher-Powell
EE	- Estratégia Evolutiva
EEs	- Estratégias Evolutivas(s)
EO	- *Extremal Optimization*
τ-EO	- variante do EO
FT	- Função Teste
FT_i	- i-ésima Função Teste
FTM	- Função Teste Multiobjetivo
FTM_i	- i-ésima Função Teste Multiobjetivo
GEO	- *Generalized Extremal Optimization*
$GEO_{\{1,2,3,4\}}$	- Variantes do GEO elaboradas e testadas nesta Tese
GEO_{var}	- Variante do GEO
$GEO_{var\{1,2,3,4\}}$	- Variantes do GEO_{var} elaboradas e testadas nesta Tese
GEOPAR-1	- Versão paralelizada do GEO
GSA	- *Genetic Algorithm with Search Area Adaptation*
LB	- largura de banda
LR	- Latência da rede
Li, i={1,2,3}	- Tamanho L da seqüência “i”
MAPSAR	- *Multi-Application Purpose Synthetic Aperture Radar*
MIMD	- *Multiple Instruction stream, Multiple Data stream*
MLI	- *Multi-Layer Insulation blankets*
MOGA	- *Multiple Objective Genetic Algorithm*
MPI	- *Message Passing Interface*
M-GEO	- Versão multiobjetivo do GEO
NAF	- Número de Avaliações da Função
NPGA	- *Niched Pareto Genetic Algorithm*
NSGA	- *Non-Dominated Sorting Genetic Algorithm*
PAH	- *Parallel Asynchronous Hybridization*
PEAs	- *Parallel Evolutionary Algorithms*
PSH	- *Parallel Synchronous Hybridization*
RS	- Recozimento Simulado
SA	- Sugestão de Aprimoramento
SAi	- Sugestão de Aprimoramento "i"
SAR	- *Synthetic Aperture Radar*

SCD3	- Satélite de Coleta de Dados 3
SH	- *Sequential Hybridization*
SIMD	- *Single Instruction stream, Multiple Data stream*
SOC	- *Self-Organized Criticality*
Ti, i={1,2,3}	- Tempo de execução “i”
VEGA	- *Vector Evaluated Genetic Algorithm*

CAPÍTULO 1

INTRODUÇÃO

Existem inúmeras técnicas numéricas para tratar o problema do projeto otimizado (Fox, 1971; Arora, 1989; Vanderplaats, 1998; Johnson, 1978; Wield, 1978; Reklaitis et al., 1983; Michalewicz e Fogel, 2000). A eficiência de cada uma delas é variável, depende do problema que está sendo resolvido/otimizado. Com isso, pode-se dizer que não existe uma técnica que seja melhor que todas as outras, mas técnicas que são mais indicadas para certos tipos de problemas (Vanderplaats, 1998; Wolpert e Macready, 1995).

Entre os principais desafios que se apresentam no campo do projeto otimizado, pode-se citar:

- Elevados tempos de simulação do sistema a ser otimizado;

- Ausência de expressões analíticas para as derivadas do problema de otimização (função objetivo e restrições);

- Mistura de tipos de variáveis (contínuas e discretas);

- Existência de severas não linearidades e descontinuidades;

- Inexistência de um "ponto de partida" válido para o algoritmo de otimização;

- Inexistência de garantias da melhor solução (solução ótima) ter sido, de fato, encontrada.

A busca por novos algoritmos de otimização se norteia em tentar vencer um ou mais dos desafios apresentados, sendo que, às vezes, a superação de um dos desafios citados é conflitante com relação aos demais. Um exemplo de tal situação é imaginar que uma solução para o desafio da diminuição do tempo de simulação seja utilizar modelos mais simples, aproximados, para a simulação do sistema. Esta solução pode tornar-se conflitante com o desafio de garantir que a melhor solução tenha sido encontrada, pois se a

simplificação do modelo for excessiva, a simulação não refletirá mais o sistema real e, portanto, as soluções rapidamente obtidas também não.

Muitas técnicas numéricas têm sido desenvolvidas para lidar com problemas de otimização em ciência e engenharia (Vanderplaats, 1998; Pardalos e Romeijn; 2002; Glover e Kochenberg, 2003). Nos primórdios, a maior preocupação era encontrar uma solução, começando de um ponto dado (ou um conjunto de pontos) do espaço de busca e usando informação acerca da vizinhança daquele ponto(s). Nesse sentido, uma solução era um ponto onde nenhuma melhoria adicional podia ser feita na função objetivo definida, porque todos os pontos na vizinhança do ponto solução levavam a valores piores da função objetivo. Hoje em dia, aquelas soluções são chamadas soluções locais, extremos locais, ou soluções subótimas do problema de otimização, porque elas são totalmente dependentes da localização do ponto ou pontos iniciais e não existe qualquer garantia de que elas sejam, de fato, as soluções ótimas globais. Estes métodos dos primórdios, ditos tradicionais, são chamados métodos de otimização de busca local. Não há dúvida sobre a utilidade de tais métodos, uma vez que, sob qualquer análise, uma solução melhorada localmente é melhor do que uma não melhorada. Infelizmente, também é sabido que muitos, talvez a maioria, dos problemas de otimização em ciência e engenharia são não lineares, multimodais e muitos deles sujeitos a diversas restrições nas variáveis (Eldred, 1998). Este fato, mais o poder computacional crescente dos computadores, tem aumentado o interesse por métodos de otimização global (Pardalos e Romeijn, 2002), nos quais a maior preocupação não é encontrar somente uma solução melhorada localmente, mas sim uma solução melhorada globalmente. Nos últimos 20 anos, um número considerável de métodos globais tem sido desenvolvido. Muitos deles são baseados em analogias dos fenômenos naturais, tentando copiar a eficiência e a simplicidade de processos auto-otimizados na natureza.

Uma das primeiras idéias que vem a mente quando se pensa no tipo de informação que deve ser usada para direcionar a busca é a idéia de fazer a busca orientando-se pelo gradiente da função objetivo. Métodos de primeira ordem, como o método da “máxima descida” - *steepest descent* (Morse e Feshbach, 1953) e outros mais sofisticados como o método do gradiente conjugado (Fletcher e Reeves, 1964), o método Davidon-Fletcher-Powell (DFP) (Davidon, 1959; Fletcher e Powell, 1963) ou o método Broyden-Fletcher-

Goldfarb-Shanno – BFGS (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970), usam a informação das primeiras derivadas para determinar uma direção de busca a cada iteração. Os métodos DFP e BFGS são ainda conhecidos como métodos de métrica variável ou quase-newtonianos (*quasi-newton methods*), pois utilizam as informações do gradiente nas sucessivas iterações para construir uma aproximação do Hessiano e obter convergência superlinear. Métodos de Segunda ordem, como o método de Newton-Raphson (Sebah e Gourdon, 2001), são empregados se a matriz do Hessiano estiver disponível e for não singular. Em geral, quanto maior a ordem do método de busca local, melhor é sua velocidade de convergência, ou seja, menor o número de iterações para chegar ao mínimo ou máximo local. Todavia, nem sempre o gradiente e o Hessiano da função objetivo estão disponíveis, seja porque são muito custosos computacionalmente (problemas com multidimensionalidade elevada), seja porque a função objetivo possui não linearidades e descontinuidades severas (Eldred, 1998). Nestes casos, métodos de ordem zero são utilizados. Eles dispensam o uso da informação das derivadas, como o método das direções conjugadas (Powell, 1962), o método Simplex (Nelder e Mead, 1965), e outros (Rosenbrock, 1960; Hooke e Jeeves, 1961; Box, 1966; Spendley et al., 1962), além dos métodos baseados em buscas aleatórias pelo ótimo no espaço de projeto (Vanderplaats, 1998).

Um grande limitador da abrangência de aplicação dos métodos tradicionais é o fato deles terem sido desenvolvidos primariamente para problemas com variáveis contínuas apenas. Problemas compostos, que apresentem variáveis contínuas, discretas e/ou inteiras, em geral são atacados tratando as variáveis discretas e inteiras como variáveis contínuas, e depois arredondando os resultados para os valores discretos ou inteiros mais próximos. Este procedimento pode, no entanto, produzir soluções subótimas ou mesmo não viáveis (Lin e Hajela, 1992). Outra estratégia possível é fazer buscas sistemáticas nas variáveis contínuas, fixando valores para as outras variáveis, desdobrando o problema original em diversos subproblemas (Lin e Hajela, 1992). Infelizmente, em problemas reais de engenharia, o espaço de projeto pode se apresentar muito mais difícil do que o recomendável para ser resolvido eficientemente pelos métodos determinísticos de busca local. Além de possuir inúmeros mínimos e eventualmente ser composto por um conjunto de variáveis contínuas, inteiras e/ou discretas, com freqüência, as derivadas analíticas da

função objetivo e restrições não estão disponíveis. Outro fato característico de muitos problemas reais é uma severa não linearidade da função objetivo e das restrições. Embora muitas vezes contornáveis por meio de artifícios numéricos, ou de aproximações, elas podem levar os métodos determinísticos de busca local a apresentarem severas limitações de desempenho.

A existência de problemas de otimização cada vez mais complexos nas diversas áreas da ciência e da engenharia tem estimulado a busca por computadores e algoritmos cada vez mais velozes (Eldred, 1998). No campo do hardware, uma das soluções é o uso de máquinas paralelas, compostas por dezenas, centenas e até mesmo milhares de processadores. No campo do software, um caminho possível é o da hibridização de algoritmos, onde se tenta aumentar a eficiência pela incorporação das melhores características de diversos algoritmos em um único.

A característica mais comum presente em problemas complexos é, provavelmente, a existência de múltiplas soluções subótimas, ou seja, extremos locais (Eldred, 1998). Em um problema real não se conhece o espaço de projeto, e a presença de múltiplos extremos é uma possibilidade a ser considerada sempre. Assim, quando métodos de busca local são usados, várias reinicializações do processo de busca devem ser feitas de pontos distintos do espaço de projeto, de forma a aumentar a probabilidade do extremo global ser encontrado.

Buscando evitar que a procura pela solução ótima fique retida em um extremo local, vários métodos realizam uma busca global pelo ponto ótimo no espaço de projeto (Gray et al., 1997; Michalewicz e Fogel, 2000; Pardalos e Romeijn, 2001; Pintér, 1999). Em geral são métodos que podem ser adaptados para resolver problemas com e sem restrições, contínuos ou combinatórios. Uma classe desses métodos é formada por meta-heurísticas inspiradas em processos que ocorrem na natureza. Meta-heurísticas são heurísticas que guiam, que governam outras heurísticas (Reiners e Voß, 2003). As meta-heurísticas citadas tratam os problemas de otimização por meio de mecanismos numéricos que tentam mimetizar fenômenos naturais. Atualmente, existem vários métodos deste tipo que se baseiam, por exemplo, em colônias de formigas (Bonabeu et al., 2000), enxame de partículas – *particle swarm* (L∅vbjerg, 2002), resfriamento de metais (Kirkpatrick et al.,

1983), seleção natural (Goldberg, 1989; Davis et al., 1999), cérebros biológicos (Freeman e Skapura, 1991) e sistemas imunológicos (Castro e Timmis, 2002). Dentre esses métodos inspirados pela natureza, dois que se destacaram, tanto em ciência quanto em engenharia, são o Recozimento Simulado (RS) e os Algoritmos Genéticos (AGs). No caso do RS, imita-se o processo de disposição dos átomos de um cristal para atingir uma energia interna mínima durante o resfriamento. No caso dos AGs, imita-se o processo de seleção natural que ocorre na evolução das espécies. Ambos os métodos utilizam uma abordagem probabilística na busca do mínimo e, em geral, exigem um número grande de avaliações da função objetivo.

A partir dos anos 60 começou a surgir uma nova classe de algoritmos de otimização que se inspiram no processo de evolução natural para fazer a busca da solução ótima (global) no espaço de projeto. Mais recentemente, estes algoritmos, que podem ser genericamente denominados de algoritmos evolutivos (Eiben e Smith, 2003; Michalewicz e Fogel, 2000), têm recebido grande atenção devido a sua universalidade, podendo ser aplicados em praticamente qualquer tipo de problema de otimização. Em 1993, Bak e Sneppen (Bak e Sneppen, 1993) desenvolveram um modelo simplificado para simular o processo evolutivo das espécies, utilizando como fundamento a teoria da Criticalidade Auto-Organizada (SOC- *Self-Organized Criticality*). Inspirados neste modelo, Boettcher e Percus (Boettcher e Percus, 2001) propuseram uma meta-heurística para problemas difíceis de otimização combinatória e a denominaram *Extremal Optimization* (EO). Em Sousa (2002), um algoritmo variante do EO, denominado *Generalized Extremal Optimization* (GEO) foi desenvolvido. O GEO é uma generalização que estendeu a aplicabilidade do EO virtualmente a qualquer problema de otimização. O algoritmo GEO é uma meta-heurística de otimização global (Sousa, 2002; Sousa e Ramos, 2002; Sousa et al., 2003b, 2004a, 2004b, 2004c). Assim como o Recozimento Simulado (RS) (Kirkpatrick et al., 1983) e os Algoritmos Genéticos (AGs) (Goldberg, 1989), o GEO é um algoritmo estocástico cuja inspiração origina-se na analogia de processos observados na natureza. Destina-se, primeiramente, a resolver problemas de otimização complexos, onde pouco ou nada se conhece a respeito do espaço de busca viável e inviável da função a ser otimizada. Pode ser aplicado a praticamente qualquer problema de otimização contínua ou combinatória, com função objetivo não convexa e espaço de projeto disjunto, com ou sem restrições, podendo,

inclusive, ser aplicado a problemas com miscelânea de variáveis reais, inteiras ou discretas. É de fácil implementação e usa apenas valores da função objetivo, sem derivadas. Não requer nem mesmo o fornecimento de um ponto de partida situado no espaço viável.

Uma outra classe ou categoria de problemas de otimização são os problemas multiobjetivo (Van Veldhuizen, 2000; Coello, 1999; Miettinen, 2001). Neles, há um conjunto (ou vetor) de funções objetivo, as quais devem ser otimizadas simultaneamente. Neste caso, a análise é feita no espaço m-dimensional das m funções objetivo. Freqüentemente, a otimização multiobjetivo implicará na não existência de uma única solução ótima de projeto que otimize todas as funções, mas soluções de compromisso, que não precisam ser ótimos globais para nenhuma função objetivo individualmente. Formalmente, uma solução de compromisso pode ser definida seguindo o conceito de otimização de Pareto (Veldhuizen e Lamont, 2000). Nesse caso, uma solução no espaço de funções objetivo pertence à assim chamada fronteira de Pareto se a melhora no valor de uma das funções objetivo implica, necessariamente, em uma piora no valor de ao menos uma das outras. Qualquer uma delas poderá ser usada como a solução para o problema e caberá ao projetista escolher qual será implementada na prática. Problemas multiobjetivo podem ser resolvidos por diversas técnicas numéricas (Goicoechea et al., 1982; Shapour, 1996). Uma das abordagens possíveis, conhecida como métodos de agregação, combina as diversas funções objetivo em uma única função, que é então otimizada usando-se as técnicas desenvolvidas para os problemas monoobjetivo. Neste caso, a determinação dos diversos pontos da fronteira de Pareto é feita variando-se os pesos aplicados às funções objetivo individuais (Giles, 1997). A montagem da função monoobjetivo pode ser feita de diversas maneiras, inclusive com a introdução de coeficientes que ajustam a curvatura da mesma, de forma a capturar pontos da fronteira que não poderiam ser obtidos usando-se uma simples combinação linear das funções (Messac et al., 2000). Algoritmos evolutivos vêm também sendo utilizados para determinar a fronteira de Pareto em problemas multiobjetivo (Cheng e Li, 1998; Coello e Christiansen, 1998; Proos et al., 2001). No caso do uso de algoritmos genéticos, por exemplo, vários pontos da fronteira de Pareto podem ser aproximados simultaneamente (Cheng e Li, 1998; Fujita et al., 1998) no decorrer do processo de evolução da população de soluções.

Sendo o GEO um método relativamente novo, uma quantidade significativa de estudo deve ser efetuada, de modo a desenvolver o potencial presente no método original. Esta Tese de Doutorado pretende estudar meios de ampliar o alcance e a eficiência do GEO. Para tanto, estão previstas quatro frentes de atuação: (i) aperfeiçoar o GEO em si, por meio de um estudo detalhado de sua forma de operação, buscando com isto, detectar formas de alterá-lo e aumentar sua eficiência; (ii) ampliar a aplicabilidade do GEO com o desenvolvimento de uma versão paralelizada do mesmo; (iii) desenvolver versões híbridas do GEO com outros algoritmos e; (iv) desenvolver um algoritmo de otimização multiobjetivo baseado no GEO.

Esta Tese está organizada como segue. No Capítulo 2, é feita uma descrição do problema de otimização. No Capítulo 3, é feita uma descrição do algoritmo de otimização GEO e, mediante estudos detalhados de seu funcionamento, versões aprimoradas são propostas. No Capítulo 4, o tema da paralelização é abordado e os estudos referentes à paralelização do GEO são descritos, bem como a versão paralelizada desenvolvida, denominada GEOPAR-1. No Capítulo 5, estão os estudos referentes à hibridização e às várias versões híbridas desenvolvidas. No Capítulo 6, o tema da otimização multiobjetivo é desenvolvido, e o algoritmo multiobjetivo M-GEO é descrito e testado com o auxílio de funções teste. No Capítulo 7, duas aplicações de projeto otimizado na área espacial são expostas e os resultados obtidos com a solução dos mesmos utilizando as versões desenvolvidas nesta Tese são apresentados e analisados. Finalmente, no Capítulo 8 estão as conclusões e sugestões para trabalhos futuros.

CAPÍTULO 2

ABORDAGENS PARA O PROBLEMA DE OTIMIZAÇÃO

O problema de otimização consiste em encontrar a melhor solução possível para uma determinada necessidade ou objetivo. Normalmente, o objetivo é definido matematicamente em termos quantitativos em função das variáveis cujos valores se busca encontrar, sendo chamado de função objetivo, F(**X**). As variáveis, aqui agrupadas no vetor **X**, são chamadas de variáveis de projeto. O conjunto de todos os valores possíveis dessas variáveis define o espaço de busca. Normalmente, os problemas de otimização apresentam limitações nos valores possíveis das variáveis de projeto, tanto diretamente, na forma de intervalos de valores possíveis, quanto indiretamente, por meio de relações funcionais entre as mesmas. Estas limitações são chamadas de restrições do problema de otimização. O conjunto particular de valores com o qual a função objetivo apresenta o melhor valor é a solução ótima do problema, também conhecida como solução global. Em contraposição à solução global, uma solução local garante apenas não haver solução melhor na sua vizinhança imediata. Eventualmente, existem problemas em que se quer a melhor solução não para uma única função objetivo, mas sim para várias. Nesse caso, o problema de otimização é dito ser multiobjetivo. No âmbito da otimização numérica, computadores são utilizados na busca da melhor solução.

Em muitos problemas oriundos da vida real, existe a necessidade de se modelar o processo (natural ou artificial) associado à obtenção da função objetivo, de modo que o mesmo possa ser artificialmente reproduzido no computador, ainda que de maneira aproximada, gerando, então, uma simulação computacional. Exemplos de processos são: órbitas de satélites, vôo atmosférico, dinâmica atmosférica, dinâmica estrutural, dinâmica molecular, troca de calor, reações químicas, apenas para citar alguns. Desta forma, na resolução deste tipo de problemas de otimização, os seguintes passos são necessários:

1) Modelar e simular o processo associado ao problema;

2) Definir matematicamente o problema de otimização e suas restrições;

3) Escolher o otimizador mais apropriado ao problema e implementá-lo;

4) Aplicar o otimizador ao problema e obter a solução ou soluções ótimas.

Muito embora a otimização ocorra de fato apenas no passo 4 recém mostrado, os demais passos acabam sendo igualmente importantes na solução do problema de otimização. Se, por exemplo, o passo 1 resultar numa simulação muito custosa computacionalmente, provavelmente inviabilizará o passo 4, onde muitas, possivelmente milhares de simulações se farão necessárias. O oposto também é verdadeiro, pois se o passo 1 resultar numa simulação muito grosseira (e pouco custosa computacionalmente) do processo real, a solução obtida no passo 4 provavelmente não será a solução ótima do problema de otimização real e sim a solução de uma simulação inverídica da realidade, podendo ser, inclusive, uma solução muito distante da solução ótima. O passo 2 é igualmente importante, pois qualquer deslize na definição matemática da função objetivo e de suas restrições levará a soluções equivocadas ou, pior ainda, à inexistência de uma solução viável, ou seja, que atenda a todas as restrições. Raciocínio semelhante pode ser aplicado ao passo 3.

Por simplicidade e sem perda de generalidade, adotar-se-á ao longo do texto que o problema de otimização é de minimização. Matematicamente tal problema pode ser definido como (Vanderplaats , 1998):

Minimizar:	$\min_{\mathbf{X}} F(\mathbf{X})$		(função objetivo)	(1)
Sujeito a:	$g_j(\mathbf{X}) \leq 0$	$j \in \{1,2,...,m\}$	(restrições de desigualdade)	(2.1)
	$h_k(\mathbf{X}) = 0$	$k \in \{1,2,...,l\}$	(restrições de igualdade)	(2.2)
	$\mathbf{X}_{MIN} \leq \mathbf{X} \leq \mathbf{X}_{MAX}$		(restrições laterais)	(2.3)
em que:	$\mathbf{X}^T = [\ X_1\ X_2\ X_3\ \ldots\ X_N\ ]$		(variáveis de projeto)	

A função F(**X**) pode ser linear ou não linear, explícita ou implícita em **X**, mas para que a maioria das técnicas numéricas tradicionais de otimização seja aplicada é importante que a F(X) seja contínua e tenha primeiras derivadas contínuas.

Para uma função objetivo sem restrições, contínua e com derivadas contínuas até a segunda ordem, as condições necessárias para **X*** ser um mínimo local são:

1) $\nabla \mathbf{F}(\mathbf{X}^*) = \mathbf{0}$, (3)

2) $\mathbf{u}^T \mathbf{H}(\mathbf{X}^*)\mathbf{u} > 0 \ , \ \forall \ \mathbf{u} \neq \mathbf{0}$ (4)

em que:

$$\tilde{N}F(\mathbf{X}^*)^T = [\partial F(\mathbf{X}^*)/\partial X_1 \ \ \partial F(\mathbf{X}^*)/\partial X_2 \ \ \ldots \partial F(\mathbf{X}^*)/\partial X_n]$$

$$\mathbf{H}(\mathbf{X}^*) = \nabla^2 F(\mathbf{X}^*) = \begin{bmatrix} \frac{\partial^2 F(\mathbf{X}^*)}{\partial X_1^2} & \frac{\partial^2 F(\mathbf{X}^*)}{\partial X_1 \partial X_2} & \cdots & \frac{\partial^2 F(\mathbf{X}^*)}{\partial X_1 \partial X_n} \\ \frac{\partial^2 F(\mathbf{X}^*)}{\partial X_2 \partial X_1} & \frac{\partial^2 F(\mathbf{X}^*)}{\partial X_2^2} & \cdots & \frac{\partial^2 F(\mathbf{X}^*)}{\partial X_2 \partial X_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 F(\mathbf{X}^*)}{\partial X_n \partial X_1} & \frac{\partial^2 F(\mathbf{X}^*)}{\partial X_n \partial X_2} & \cdots & \frac{\partial^2 F(\mathbf{X}^*)}{\partial X_n^2} \end{bmatrix}$$

Dito com palavras, o vetor gradiente, $\nabla \mathbf{F}(\mathbf{X}^*)$, deve ser nulo e a matriz do Hessiano, $\mathbf{H}(\mathbf{X}^*)$, deve ser definida positiva, ou seja, ter todos os seus autovalores positivos.

Para o problema com restrições, um ponto de mínimo deverá satisfazer as condições de Kuhn-Tucker (Vanderplaats , 1998):

1) **X*** é viável, ou seja, não viola nenhuma restrição.

2) $\lambda_j g_j(\mathbf{X}^*) = 0 \ , \qquad j \in \{1,2,...,m\} \ ; \ \lambda_j \geq 0$ (5)

3) $\nabla F(\mathbf{X}^*) + \sum_{j=1}^{m} ?_j \nabla g_j(\mathbf{X}^*) + \sum_{k=1}^{l} ?_{k+m} \nabla h_k(\mathbf{X}^*) = 0 \ , \quad \text{com} \ \ \tilde{N}(.) = \frac{\partial(.)}{\partial \mathbf{X}}$ (6)

As condições recém mostradas, tanto para problemas com restrições como para problemas sem restrições, são condições necessárias, mas não suficientes para **X*** ser o ótimo global buscado. Elas são suficientes, sim, para garantir que **X*** seja um mínimo local da função objetivo, no caso dos problemas sem restrições, ou ainda, para garantir que **X*** seja um ponto estacionário (ponto de sela ou ponto de mínimo ou de máximo local) da

função objetivo, no caso dos problemas com restrições, como pode ser visto nas Figuras 2.1 e 2.2 a seguir.

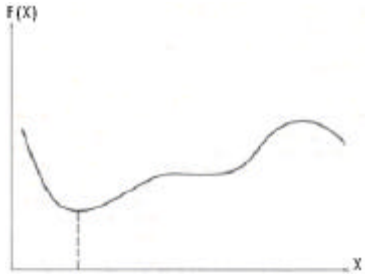


Figura 2.1 – Mínimo local em um problema sem restrições
Fonte: Vanderplaats (1998).

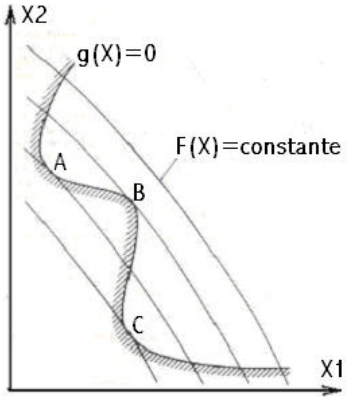


Figura 2.2 – Pontos estacionários em um problema com restrições
Fonte: Vanderplaats (1998).

Problemas com restrições devem ainda satisfazer a seguinte condição de segunda ordem, a fim de garantir que **X*** seja um mínimo local (pontos A e C na Figura 2.2) de F(**X**) (Fourer e Moré, 2004):

$$4)\quad \mathbf{u}^T\mathbf{L}_{xx}\mathbf{u}>0\ ,\ \ \forall u\neq 0:\ \left\{\begin{array}{c}\nabla h_k(\mathbf{X}^*)^T u=0,\ k\in\{1,2,...,m\};\\ \nabla g_j(\mathbf{X}^*)^T u=0,\forall j:?_j>0;\\ \nabla g_j(\mathbf{X}^*)^T u\leq 0,\forall j:?_j=0\end{array}\right\} \tag{7}$$

$$\text{em que: } L_{xx}=\nabla^2 F(\mathbf{X}^*)+\sum_{j=1}^{m}?_j\nabla^2 g_j(\mathbf{X}^*)+\sum_{k=1}^{l}?_{k+m}\nabla^2 h_k(\mathbf{X}^*), \tag{8}$$

$$\text{e } \tilde{N}(.)=\frac{\partial(.)}{\partial\mathbf{X}} \quad \text{e} \quad \tilde{N}^2(.)=\frac{\partial^2(.)}{\partial\mathbf{X}^2}$$

Métodos que se pautam pelo embasamento teórico recém apresentado, aqui ditos métodos tradicionais de minimização, fazem busca local, ou seja, encontram o mínimo local na vizinhança do ponto de partida da busca.

De acordo com Vanderplaats (1998), os métodos tradicionais de otimização podem tratar as restrições de forma indireta, transformando um problema com restrições em um sem restrições por meio de penalidades à função objetivo (como nos métodos da função penalidade interior, exterior, exterior estendida e método dos multiplicadores de Lagrange estendido), ou introduzindo as restrições diretamente (como nos métodos da programação linear – quando a função objetivo e suas restrições são lineares – ou programação linear seqüencial, método dos centros, método das direções viáveis, método do gradiente reduzido generalizado ou a programação quadrática seqüencial) .

No âmbito dos Algoritmos Evolutivos, Michalewicz (1995), Michalewicz e Schoenauer (1996), e Eiben e Smith (2003), apresentam uma revisão das técnicas utilizadas para lidar com as restrições em problemas de otimização. Talvez a primeira abordagem utilizada por estes algoritmos para lidar com indivíduos (soluções) inviáveis tenha sido condená-los sumariamente à morte (*death penalty*), ou seja, rejeição, ao longo de todo o processo de evolução (otimização) de todos os indivíduos inviáveis por meio de sua eliminação (exclusão) da população de indivíduos. Atualmente, já existe um certo consenso de que na maioria das vezes esta abordagem não é adequada, pois não preserva e, portanto, também não utiliza qualquer informação a respeito das restrições violadas para melhorar o processo evolutivo. Alguns autores inclusive concluem que a presença de restrições num problema de otimização não é prejudicial e sim benéfica, pois introduz informação extra a ser utilizada pelo algoritmo de otimização (Eiben e Smith, 2003). Este fato é tanto mais relevante quanto menor for o espaço viável, relativamente ao espaço total de busca de um problema de otimização dado. Para estes problemas, uma abordagem do tipo *death penalty* é quase sem esperança, pois a busca por soluções viáveis é como buscar uma agulha num palheiro (existem problemas onde a razão espaço viável/espaço total é de 1/100.000, ou menos). Portanto, não há outra escolha senão utilizar, sempre que possível, as funções que representam as restrições como “funções objetivo alternativas”, que guiam o algoritmo de otimização dentro do espaço inviável na direção do espaço viável. Neste

sentido, uma abordagem possível é transformar o problema monoobjetivo com restrições, formulando-o como um problema multiobjetivo sem restrições, onde cada restrição que existe no problema monoobjetivo torna-se uma função objetivo a mais do problema multiobjetivo.

Em Eiben e Smith (2003), é feita uma classificação das técnicas existentes para lidar com as restrições no âmbito dos Algoritmos Evolutivos. Resumidamente, existem quatro técnicas: -penalização; -reparação; -preservação e; -decodificação. A primeira delas, penalização, é provavelmente a técnica mais conhecida e na qual as restrições são transformadas em funções objetivo inseridas na função objetivo original, criando, assim, uma função objetivo "aumentada" sem restrições. A técnica de reparação modifica indivíduos (soluções) inviáveis, transformando-os em indivíduos tão similares ao original quanto possível (proximidade no espaço de busca) e que sejam viáveis. A técnica de preservação garante, mediante o uso de operadores adequados, que indivíduos inviáveis não ocorram ao longo do processo de otimização evolutiva, ou seja, ocorre a preservação da viabilidade dos indivíduos. A técnica de decodificação se vale de funções "conversoras" adicionais, que fazem a conversão de genótipos para fenótipos e, ao mesmo tempo, garantem que os fenótipos resultantes sejam viáveis . Os fenótipos são os indivíduos, as soluções na sua representação original do problema de otimização. Já os genótipos são os indivíduos, as soluções na representação requerida pelos operadores do algoritmo evolutivo. Durante uma otimização usando um algoritmo evolutivo, uma das primeiras etapas é realizar a conversão de fenótipos para genótipos, operação denominada codificação. A operação inversa, de conversão de genótipos para fenótipos é denominada decodificação. Durante uma busca, um algoritmo evolutivo manipula genótipos, criando uma nova geração dos mesmos a cada iteração. Em geral, a avaliação da viabilidade e da adaptação destes indivíduos ocorre no contexto do problema original. Torna-se necessário, portanto, realizar-se a decodificação, que é a conversão das novas gerações de genótipos em seus fenótipos equivalentes, a fim de que sejam avaliados. Até aqui, as técnicas de preservação e decodificação são indistintas. A principal diferença entre elas é que na preservação, a formulação e os operadores do algoritmo evolutivo garantem naturalmente que os genótipos gerados sejam viáveis, enquanto na decodificação essa garantia não existe, sendo necessário utilizar-se de funções especiais para essa finalidade na fase de

decodificação. Além disso, na preservação a correspondência entre genótipos e fenótipos é biunívoca, enquanto na decodificação pode ocorrer de genótipos diferentes em graus arbitrários serem "decodificados" para o mesmo fenótipo.

Voltando-se a atenção para a diversidade dos problemas de otimização, é possível classificar genericamente os métodos numéricos de otimização em duas grandes categorias: Métodos de busca local e métodos de busca global. Abaixo dessas duas grandes classes os métodos podem ser postos em subclasses de acordo, por exemplo, com o tipo de abordagem usada para selecionar direções de busca do mínimo. A Figura 2.3 abaixo procura mostrar graficamente alguns métodos de otimização numérica usados em projeto otimizado, classificando-os por características semelhantes. Cabe ressaltar que a classificação exposta na Figura 2.3 pode ser feita de outras formas. Partindo por exemplo de classes genéricas como I) métodos sem restrição x métodos com restrição; II) métodos determinísticos x métodos estocásticos; III) otimização contínua x otimização discreta; etc. Na Figura 2.3, o método GEO aparece destacado em negrito e tamanho de fonte aumentado.

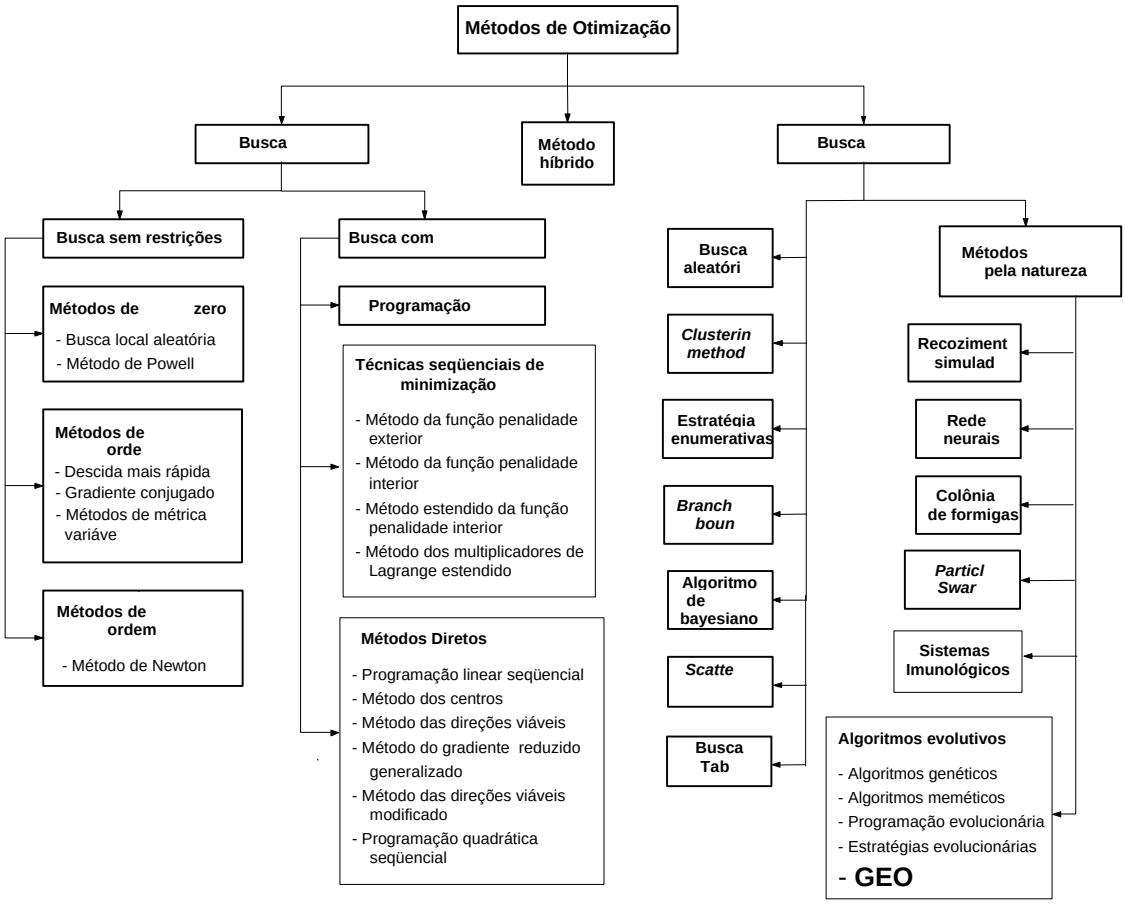


Figura 2.3 – Uma classificação para os métodos de otimização numérica.
Fonte: adaptado de Sousa (2002).

CAPÍTULO 3

APRIMORAMENTO DO MÉTODO DA OTIMIZAÇÃO EXTREMA GENERALIZADA

3.1 – Preâmbulo

Neste Capítulo, o funcionamento do GEO é estudado. O objetivo é o de inferir modificações que o aperfeiçoem, gerando, então, variantes do algoritmo. Inicialmente, é feita uma descrição resumida da teoria da criticalidade auto-organizada (SOC – *Self-Organized Criticality*) e do modelo simplificado de evolução das espécies que Bak e Sneppen (1993) desenvolveram para evidenciar que tais sistemas possuem SOC. Em seguida, o algoritmo de otimização τ-EO (Boettcher e Percus, 2001), desenvolvido baseado na teoria SOC e no modelo de Bak-Sneppen, é descrito. Prosseguindo, o GEO (Sousa, 2002), que pode ser pensado como uma generalização do EO, é descrito. Logo após sua descrição, estão as considerações e constatações obtidas do estudo do funcionamento do algoritmo GEO e os aprimoramentos decorrentes.

3.2 – A teoria da criticalidade auto-organizada (*SOC*)

A teoria da criticalidade auto-organizada tem sido utilizada para explicar o comportamento de uma série de fenômenos naturais e sistemas complexos, tais como a evolução das espécies, a freqüência dos terremotos ou o mercado financeiro, entre outros (Bak, 1996). Criada inicialmente para explicar a origem do ruído 1/f em sistemas físicos (Bak et al., 1987), a teoria de SOC estabelece que sistemas complexos e com muitos elementos interagindo evoluem naturalmente para uma condição crítica em que uma pequena mudança em um deles gera “avalanches” que podem atingir quaisquer dos outros elementos que fazem parte do sistema (Bak e Chen, 1991). A distribuição de probabilidade dos tamanhos das avalanches é descrita por uma lei de potência na forma:

$$P(s) \sim s^{-\gamma} \tag{3.1}$$

onde s é o tamanho da avalanche, ou seja, o número de elementos que dela fazem parte e γ é um parâmetro real não negativo. Assim, a probabilidade de ocorrerem pequenas avalanches é maior, mas avalanches do tamanho do sistema podem ocorrer com

probabilidade não desprezível (dependendo, obviamente, do tamanho do sistema em questão e do valor de γ).

3.3 – O modelo de Bak-Sneppen

Utilizando um modelo numérico simplificado, que não considera os pormenores da iteração biológica entre as espécies, Bak e Sneppen (1993) conseguiram reproduzir características de um modelo de evolução (*punctuated equilibrium*) proposto por Gould e Eldredge (1993), que procura explicar evidências paleontológicas que indicam que a evolução das espécies não se deu da forma gradual proposta pela teoria darwiniana, mas por meio de saltos, intercalados por períodos longos sem atividade evolutiva expressiva (*stasis*). No modelo simplificado de Bak e Sneppen, as espécies que formam o ecossistema são colocadas alinhadas lado a lado, como apresentado na Figura 3.1 abaixo.

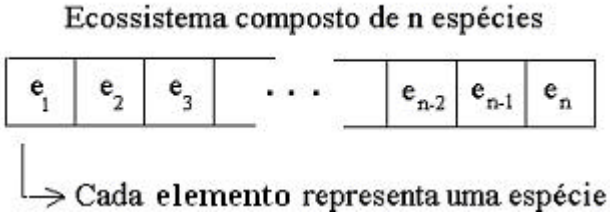


Figura 3.1 – Distribuição das espécies no modelo evolutivo de Bak e Sneppen (1993).

O modelo estabelece ainda que a linha de espécies é fechada, circular. Assim, e_2 e e_n são espécies vizinhas de e_1, da mesma forma que e_1 e e_3 são vizinhas de e_2.

A cada espécie é atribuído um número aleatório no intervalo [0,1] com distribuição uniforme. Este valor representa o índice de adaptação, I_A, da espécie. Quanto menor o I_A, menos adaptada ao seu ambiente está a espécie. O processo de evolução começa pela escolha da espécie de menor I_A da população. A ela é atribuído aleatoriamente um novo I_A, ou seja, ela sofre uma mutação. Às suas vizinhas também são atribuídos novos I_A's. A mudança no I_A da espécie menos adaptada significa para suas vizinhas que estas terão que se adaptar a um novo competidor local e, deste modo, também são forçadas a sofrer mutação, mesmo que seus I_A's sejam altos. Este processo é repetido indefinidamente. Após um certo número de repetições, a população evolui para um estado crítico, onde todas as espécies atingem um I_A acima de um certo patamar, denominado patamar crítico. Então, a

dinâmica do sistema faz com que, eventualmente, ocorram eventos, chamados de avalanches, nos quais o I_A de um número de espécies cai abaixo do patamar crítico. A intensidade (número de espécies afetadas) destas avalanches pode atingir desde uma única espécie até todo o sistema. As espécies que estão abaixo do nível crítico são mais ativas, ou seja, têm mais probabilidade de sofrer mutação. Com isso, a tendência do sistema é sempre a de retornar para a condição crítica e, portanto, avalanches de menor tamanho ocorrem com maior freqüência, como estabelecido pela equação 3.1.

Um sistema que apresenta as características recém expostas é chamado de sistema auto-organizado criticamente. Resumindo, um sistema auto-organizado criticamente caracteriza-se por apresentar um número grande de elementos que interagindo entre si, evoluem para um estado crítico no qual pequenas perturbações podem ocasionar uma reação em cadeia e afetar o sistema, no todo ou em parte.

A utilização do modelo de Bak-Sneppen em um método de otimização, no qual a dinâmica de busca apresente características de SOC, talvez possibilite o aparecimento de soluções ótimas rapidamente, sistematicamente modificando as espécies menos adaptadas da população e valendo-se das avalanches para escapar de mínimos locais.

3.4 – O método t-EO

Tendo como inspiração o modelo simplificado da evolução das espécies desenvolvido por Bak e Sneppen (1993), Boettcher e Percus propuseram uma meta-heurística para problemas difíceis de otimização combinatória (Boettcher e Percus, 1999, 2000, 2001; Boettcher et al., 2000) e a denominaram *Extremal Optimization* (EO). De forma a aumentar o desempenho do algoritmo original, eles introduziram um parâmetro ajustável no método, que permite ao mesmo evitar convergência para mínimos locais. Esta implementação foi denominada τ-EO e encontra-se exposta a seguir.

O algoritmo τ-EO é composto dos seguintes passos (adaptado de Sousa, 2002):

1. Inicialize um ponto C qualquer no espaço busca de N variáveis de projeto x_i, $i \in \{1,2,...,N\}$ e faça $C_{melhor} = C$.
2. Para um ponto corrente C,

a) Atribua um índice de adaptação F_i para cada variável x_i,

b) Ordene x_i, de k=1 para o menor F_i até k=N para o maior F_i, onde i, k $\in$ {1,2,...,N}. Crie uma lista (x_i,k), que relaciona cada variável x_i com seu número de ordem k,

c) Escolha, com probabilidade $P_{x_i}(k) \propto k^{-\tau}$, τ>0 e τ=constante, uma variável x_i a ser modificada. Escolha aleatoriamente um novo ponto C' em uma vizinhança de C tal que apenas a x_i escolhida precise mudar o seu estado,

d) Aceite C = C' incondicionalmente,

e) Se V(C) < V(C_{melhor}) então faça C_{melhor} = C, onde V(C) é o valor da função objetivo para o ponto C.

3. Repita o passo (2) o quanto for desejado.
4. Retorne C_{melhor} e V(C_{melhor}).

O método da EO (τ-EO incluso) foi criado com o intuito de ser uma meta-heurística que pudesse ser aplicada a uma vasta classe de problemas. Entretanto, o método apresenta uma característica que dificulta sua implementação de maneira universal, pois deixa a cargo do implementador de cada aplicação a definição do índice de adaptação de cada variável. Boettcher e Percus reconheceram esta peculiaridade do método EO dizendo textualmente "*... a general definition of fitness for individual variables may prove ambiguous or even impossible*." (Boettcher e Percus, 2001). O algoritmo GEO foi concebido de forma a prover o método EO com a generalização almejada.

3.5 – O Método da Otimização Extrema Generalizada

Posto de forma resumida, o algoritmo da Otimização Extrema Generalizada (*Generalized Extremal Optimization – GEO*) é uma meta-heurística de busca global (Sousa, 2002; Sousa e Ramos, 2002; Sousa et al., 2003b, 2004a, 2004b, 2004c), baseado num modelo de evolução natural das espécies (Bak e Sneppen, 1993) e especialmente destinado para uso em problemas de otimização complexos (Sousa et al., 2002a), tendo seus fundamentos na teoria da Criticalidade Auto-Organizada (*Self-Organized Criticality – SOC*), a qual tem sido usada para explicar as assinaturas de lei de potência que ocorrem em muitos sistemas complexos (Bak et al., 1987; Bak, 1996).

O GEO, assim como os métodos Recozimento Simulado (RS) (Kirkpatrick et al., 1983) e os Algoritmos Genéticos (AGs) (Holland, 1974; Goldberg, 1989; Lacerda e Carvalho, 1999), é um método estocástico, não se utiliza de derivadas e pode ser aplicado a problemas não convexos ou disjuntos. Ele pode também lidar com qualquer tipo de variáveis, seja contínua, discreta ou uma mistura delas. Seu único parâmetro livre para ajuste, τ, permite estabelecer o grau de determinismo da busca, de uma caminhada aleatória (*random walk*) com $\tau = 0$ até uma busca determinística local, com $\tau \rightarrow \infty$. Tem sido observado que existe um valor ótimo de τ para cada problema (às vezes, um intervalo de valores), tal que a eficiência da busca global é maximizada. Neste sentido, este valor de τ pode ser chamado de τ ótimo, τ^*. Para muitos problemas abordados com GEO, o que tem sido visto é que o τ^* situa-se no intervalo de 1 a 5 (Sousa, 2002; Sousa e Ramos, 2002; Sousa et al., 2003b; 2004c, 2005a). Dentre os problemas de otimização já resolvidos com o GEO, citam-se: (i) projeto de tubos de calor (Sousa et al., 2002a; Sousa et al., 2003a; Sousa et al., 2004a; 2004b; Vlassov et al., 2006); (ii) projeto térmico de satélites (Galski et al., 2004a; 2007); (iii) constelações de satélites (Galski et al., 2005a); (iv) perfis aerodinâmicos de planadores (Sousa et al., 2002b; 2003c); (v) testes de software (Abreu et al., 2005); (vi) identificação de propriedades óticas de materiais (Sousa et al., 2005b); e (vii) otimização estrutural (Sousa e Takahashi, 2005a; 2005b).

A seguir, o algoritmo GEO é descrito com mais detalhes. Finalizando esta seção, está uma descrição do algoritmo GEO_{var}, que é uma variante do GEO.

3.5.1 – GEO

Analogamente ao modelo de Bak e Sneppen (1993), L espécies são dispostas ao longo de uma linha e a cada espécie é atribuído um índice de adaptação. Nesta analogia, a população de espécies é formada por uma seqüência (*string*) de bits, onde cada um destes bits representa uma espécie. Cada bit (espécie) pode assumir o valor 0 ou 1. As variáveis de projeto são codificadas na seqüência de bits, que é similar a um cromossomo binário em um GA canônico, como apresentado na Figura 3.2 a seguir, onde os seis primeiros bits codificam a variável de projeto X_1, os seis bits seguintes codificam X_2 e assim por diante, até os últimos seis bits da seqüência, que codificam X_N.

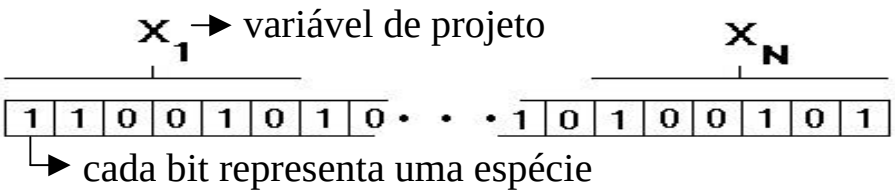


Figura 3.2 – N variáveis de projeto em uma seqüência binária com 6 bits por variável.

A partir daí, o GEO segue a abordagem τ-EO, para definir qual espécie (agora um bit) sofrerá mutação. Para cada bit da seqüência é atribuído um índice de adaptação que é proporcional ao ganho (ou perda) que o valor da função objetivo têm quando aquele bit sofre uma mutação (muda de valor de 0 para 1 ou de 1 para 0). Os bits são então ordenados de k=1 à k=L, usando o índice de adaptação, conforme exposto a seguir. O bit com menor valor do índice de adaptação (bit menos adaptado) recebe o índice k=1. O bit com o segundo menor valor do índice de adaptação (2º bit menos adaptado) recebe o índice k=2, e assim sucessivamente, até que, finalmente, o bit com maior valor do índice de adaptação (bit mais adaptado) recebe o índice k=L. Um bit é então escolhido para sofrer mutação com probabilidade $P(k) \propto k^{-\tau}$, onde τ é um parâmetro ajustável real não negativo. Este processo é repetido até que um dado critério de parada seja satisfeito. A melhor configuração de bits identificada ao longo do processo de busca é guardada e retornada ao final do mesmo.

Na inicialização do GEO ocorre o preenchimento de uma seqüência binária **C** de comprimento L, por meio de L sorteios de 0s e 1s, ambos com igual probabilidade de ocorrência. Esta seqüência binária define uma configuração ou solução candidata a ótimo global, representada no sistema numérico utilizado internamente pelo algoritmo GEO (sistema binário). Entretanto, muitas vezes o sistema numérico adotado para representar as variáveis de projeto no âmbito do problema de otimização não é o sistema binário e sim o sistema decimal. Desta forma, a fim de calcular o valor da função objetivo, F(**X**), o GEO precisa primeiro converter **C** em **X**, ou seja, é necessário converter cada variável de projeto, X_j, $j \in \{1,2,...,N\}$, de sua codificação binária para seu valor decimal correspondente. Esta conversão é feita conforme a equação a seguir:

$$X_j = X_{min_j} + (X_{max_j} - X_{min_j}) \frac{\sum_{i=1}^{l_j} \left(C_i \, 2^{(i-1)} \right)}{2^{(l_j)} - 1} \tag{3.2}$$

onde X_{min_j} e X_{max_j} são, respectivamente, os valores mínimo e máximo para a variável X_j no sistema decimal; C_i é o valor do i-ésimo bit usado na representação de X_j e l_j é o número de bits para a variável X_j.

Outro aspecto importante é a definição do número de bits de cada variável, l_j, em função da precisão mínima desejada para X_j. A equação a seguir calcula o menor número de bits necessário para representar uma dada variável com precisão e_j e levando em conta o intervalo de variação da respectiva variável de projeto.

$$l_j = \log_2 \left(1 + \frac{(X_{max_j} - X_{min_j})}{e_j} \right) \tag{3.3}$$

As restrições de igualdade e desigualdade são tratadas pelo GEO simplesmente atribuindo um mesmo e alto índice de adaptação aos bits que, ao se modificarem, levarem a soluções inviáveis. No GEO, tais bits serão sempre mais adaptados que aqueles bits que, ao se modificarem, levem a soluções viáveis. Além disso, como o índice de adaptação de tais bits é o mesmo, não há distinção entre os mesmos para fins de ordenamento, que ocorre, para eles, de maneira aleatória com igual probabilidade. Este procedimento difere do procedimento comumente usado para tratar as restrições e que pode levar a soluções subótimas (Hajela e Yoo, 1996): o uso de funções penalidade. A função objetivo no GEO não é modificada, apenas modificações em bits que levem a regiões inviáveis do espaço de projeto têm menos chance de ocorrer. É importante salientar que o GEO opera "indiferentemente" para soluções viáveis e inviáveis. Durante a busca pelo ótimo, o algoritmo pode se mover por regiões inviáveis do espaço de projeto e até mesmo iniciar a busca partindo de uma solução inviável. Isto dá uma grande flexibilidade ao algoritmo que pode, por exemplo, ser aplicado a problemas onde o espaço de projeto apresenta regiões viáveis desconectadas, como na otimização de estruturas submetidas a oscilações harmônicas (Johnson, 1976). Além disso, nos problemas em que o espaço de soluções

viáveis é pequeno com relação ao espaço total de soluções, o simples fornecimento de um ponto de partida situado no espaço viável, como requerido por muitos algoritmos, pode por si só, transformar-se num problema à parte.

O algoritmo GEO é composto dos seguintes passos (adaptado de Sousa, 2002):

1. Inicialize aleatoriamente uma seqüência binária **C**, de comprimento L que codifica N variáveis de projeto. Dentro de **C**, cada variável de projeto X_j é codificada em uma parcela com comprimento l_j, $j \in \{1,2,...,N\}$ e tal que $\sum_j l_j = L$. Converta **C** em **X** e calcule o valor da função objetivo F(**X**) e faça $\mathbf{X_{melhor}} = \mathbf{X}$ e $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$.

2. Para cada bit $i \in \{1,2,...,L\}$ da seqüência **C**, faça:
    a. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule $F(\mathbf{X_i})$.
    b. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - F(\mathbf{X_{melhor}})$, que indica o ganho (ou perda) que se têm ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.
    c. Retorne o bit i ao seu valor original.

3. Ordene todos os bits $i \in \{1,2,...,L\}$ de acordo com os seus índices de adaptação $\Delta F(\mathbf{X_i})$, de k=1 para o menos adaptado à k=L, para o mais adaptado. Crie uma lista de valores (i,k) relacionando a posição física i do bit em **C** com seu respectivo número de ordem k, onde $i,k \in \{1,2,...,L\}$. Se $\Delta F(\mathbf{X_i}) = \Delta F(\mathbf{X_j})$, i?j, estabeleça a ordem entre i e j de forma aleatória.

4. Escolha com igual probabilidade um bit candidato i para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual à $P_i(k)$, confirme o bit i para ser modificado. Repetir este passo até que um bit seja confirmado. Quando isso acontecer, faça $i_{esc} = i$.

5. Faça $\mathbf{C} = \mathbf{C_{iesc}}$, converta **C** em **X** e calcule F(**X**).

6. Se $F(\mathbf{X}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$ e $\mathbf{X_{melhor}} = \mathbf{X}$.

7. Repita os passos 2 a 6 até que um dado critério de parada seja satisfeito.
8. Retorne $\mathbf{X}_{\mathbf{melhor}}$ e $F(\mathbf{X}_{\mathbf{melhor}})$.

3.5.2 – GEO_{var}

No caso de GEO_{var}, o algoritmo é composto pelo mesmo número de passos do GEO, mas faz N ordenações separadamente para os bits das N variáveis de projeto. De cada ordenação, um bit é escolhido para mutar. Em seguida, os N bits escolhidos sofrem mutação simultânea.

Os passos para o GEO_{var} são:

1. Inicialize aleatoriamente N seqüências binárias C_j, $j \in \{1,2,...,N\}$ que codificam as N variáveis de projeto X_j. Cada C_j tem l_j bits, ou seja, o elemento $c_{j,i} \in C_j$, $i \in \{1,2,..., l_j\}$ e $c_{j,i} \in \{0,1\}$. Assim, o vetor **X** contendo as variáveis de projeto é codificado em uma matriz **C** cuja linha j contém a seqüência binária C_j. Converta **C** em **X** e calcule o valor da função objetivo $F(\mathbf{X})$ e faça $\mathbf{X}_{\mathbf{melhor}} = \mathbf{X}$ e $F(\mathbf{X}_{\mathbf{melhor}}) = F(\mathbf{X})$.
2. Para cada variável $j \in \{1,2,...,N\}$ faça:
    a. Para cada bit $i \in \{1,2,..., l_j\}$ da seqüência C_j, faça:
        1º. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule $F(\mathbf{X_i})$.
        2º. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - F(\mathbf{X}_{\mathbf{melhor}})$, que indica o ganho (ou perda) obtido ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.
        3º. Retorne o bit i ao seu valor original.
    b. Ordene os bits $i \in \{1,2,..., l_j\}$, ou seja, os bits da variável j, de acordo com os seus índices de adaptação $\Delta F(\mathbf{X_i})$, de k=1 para o menor $\Delta F(\mathbf{X_i})$ até $k=l_j$, para o maior $\Delta F(\mathbf{X_i})$, onde l_j é o número de bits da variável j.

c. Crie uma lista $(i,k)_j$, relacionando a posição física do bit i em C_j com seu número de ordem k. Se $\Delta F(\mathbf{X_i})_m = \Delta F(\mathbf{X_i})_n$, $\forall m?n$, estabeleça a ordem entre m e n de forma aleatória.

d. Escolha com igual probabilidade um bit candidato $i \in \{1,2,..., l_j\}$ para ser modificado.

e. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual a $P_i(k)$, confirme o bit para mutar.

f. Repetir os passos 2.d e 2.e até que um bit seja confirmado para ser modificado. Quando isso acontecer, faça $i_{esc_j} = i$.

3. Faça $\mathbf{C} = \mathbf{C_{esc}}$, onde $\mathbf{C_{esc}}$ é a configuração resultante da mutação em **C** dos N bits escolhidos no passo 2.f (bits i_{esc_j}, $j \in \{1,2,...,N\}$). Converta **C** em **X** e calcule F(**X**).

4. Se $F(\mathbf{X}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$ e $\mathbf{X_{melhor}} = \mathbf{X}$.

5. Repita os passos 2 a 4 até que um dado critério de parada seja satisfeito.

6. Retorne $\mathbf{X_{melhor}}$ e $F(\mathbf{X_{melhor}})$.

3.6 – Diferença entre GEO e GEO_{var}

A principal diferença entre GEO e GEO_{var} é a seguinte: o primeiro muda apenas um bit a cada iteração, enquanto o segundo muda um bit de cada variável a cada iteração. Assim, do ponto de vista do número de bits mudados a cada iteração, o GEO_{var} é N vezes mais "dinâmico" do que o GEO, onde N é o número de variáveis de projeto do problema. Mais informações a respeito de ambos, incluindo exemplos, podem ser encontradas em Sousa (2002), Sousa et al. (2003b) e Sousa et al. (2004c).

A Figura 3.3 apresenta uma visão geral de GEO e de sua variante GEO_{var} sob a forma de um fluxograma. Na figura, F(**X**) é a função objetivo, k é o valor da posição ocupada pelo bit no ordenamento efetuado a cada iteração e l_j é o número de bits da variável de projeto "j".

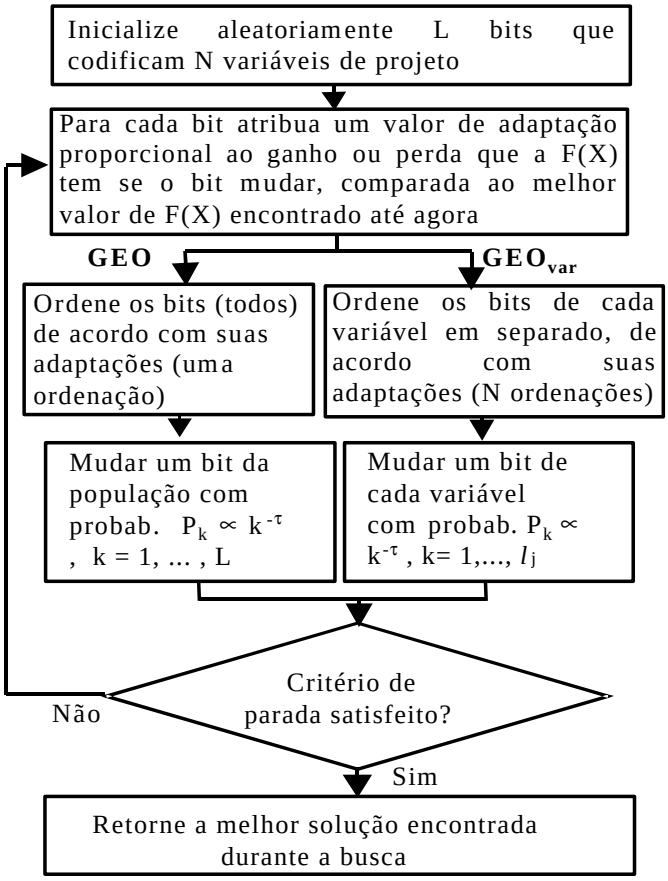


Figura 3.3 - Fluxograma para GEO/GEO_{var}

Fonte: Adaptado de Sousa et al. (2003b)

3.7 – Estudo do comportamento do GEO e do GEO_{var}

Nesta seção, são apresentadas as constatações decorrentes dos estudos efetuados acerca do funcionamento dos algoritmos GEO e GEO_{var} e que resultaram em sugestões de modificações visando aperfeiçoá-los.

3.7.1 – Sugestão de Aprimoramento 1 (SA1)

O primeiro aprimoramento, aqui chamado SA1, sugerido para GEO e GEO_{var} (Galski et al., 2003), decorre da constatação de que o GEO e o GEO_{var} originais podem vir a perder a informação sobre a melhor solução encontrada por eles ao longo da busca.

Analisando primeiramente o que ocorre com o GEO, tem-se que, na forma original, a cada iteração, o GEO executa L avaliações da função objetivo, mas compara com $F(\mathbf{X}_{melhor})$ apenas uma: aquela que efetivamente sofre mutação de seu bit, gerando o novo **C** e o novo F(**X**) (passo 5). Assim, a menos que $\tau \rightarrow \infty$, gerando uma indesejada busca

determinística local, existe uma chance não desprezível de soluções melhores, eventualmente até a solução ótima, serem encontradas e descartadas ao longo da busca. A modificação a ser apresentada elimina tal possibilidade, sem alterar o caminho percorrido pelo algoritmo no espaço de busca.

O aprimoramento SA1, sugerido para GEO e GEO_{var}, afeta os passos 2 e 6 do GEO e 2, 5 e 6 do GEO_{var}. Apresentam-se a seguir, os passos do algoritmo GEO já com as alterações. A fim de diferenciá-lo do original, esta variante é denominada GEO_1. As alterações estão realçadas em negrito.

O algoritmo GEO_1 é composto dos seguintes passos:

1. Inicialize aleatoriamente uma seqüência binária **C**, de comprimento L que codifica N variáveis de projeto. Dentro de **C**, cada variável de projeto X_j é codificada em uma parcela com comprimento l_j, $j \in \{1,2,...,N\}$ e tal que $\sum_j l_j = L$. Converta **C** em **X** e calcule o valor da função objetivo F(**X**) e faça $\mathbf{X_{melhor}} = \mathbf{X}$ e $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$.
2. **Faça $F(\mathbf{X_{melhor}})_{anterior} = F(\mathbf{X_{melhor}})$.** Para cada bit $i \in \{1,2,...,L\}$ da seqüência **C**, faça:
    a. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule $F(\mathbf{X_i})$. **Em seguida, se $F(\mathbf{X_i}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X_i})$ e $\mathbf{X_{melhor}} = \mathbf{X_i}$.**
    b. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - F(\mathbf{X_{melhor}})_{anterior}$, que indica o ganho (ou perda) que se têm ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.
    c. Retorne o bit i ao seu valor original.
3. Ordene todos os bits $i \in \{1,2,...,L\}$ de acordo com os seus índices de adaptação, de k=1 para o menos adaptado à k=L, para o mais adaptado. Crie uma lista de valores (i,k) relacionando a posição física i do bit em **C** com seu respectivo número de ordem k, onde $i,k \in \{1,2,...,L\}$. Se $\Delta F(\mathbf{X_i}) = \Delta F(\mathbf{X_j})$, i?j, estabeleça a ordem entre i e j de forma aleatória.

4. Escolha com igual probabilidade um bit candidato i para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual à $P_i(k)$, confirme o bit i para ser modificado. Repetir este passo até que um bit seja confirmado. Quando isso acontecer, faça $i_{esc} = i$.
5. Faça $\mathbf{C} = \mathbf{C_{iesc}}$, ou seja, em **C**, mute o bit i_{esc}. Converta **C** em **X** e calcule F(**X**).
6. Repita os passos 2 a 5 até que um dado critério de parada seja satisfeito.
7. Retorne $\mathbf{X_{melhor}}$ e F($\mathbf{X_{melhor}}$).

Com esta modificação, o GEO_1 passa a reter a informação da melhor configuração, ou seja, da melhor solução com a qual ele se depara ao longo de toda a busca e para todas as avaliações efetuadas da função objetivo.

A cada iteração do GEO original, a probabilidade $P_{\{\mathbf{X}\neq\mathbf{X}melhor\}}$ da melhor solução dentre todas as soluções da iteração atual, designada $\mathbf{X_{melhor}}$, não ser escolhida como ponto de destino do GEO ao final da iteração (passo 5), é dada por:

$$P_{\{\mathbf{X}\neq\mathbf{X}melhor\}} = \frac{P_{\{k_{esc}\neq 1\}}}{P_{\{\Omega\}}} = \frac{P_{\{\Omega\}}}{P_{\{\Omega\}}} - \frac{P_{\{k_{esc}=1\}}}{P_{\{\Omega\}}} = 1 - \frac{(1)^{-t}}{\sum_{k=1}^{L} k^{-t}} = 1 - \frac{1}{\sum_{k=1}^{L} k^{-t}} = 1 - \frac{1}{1+\sum_{k=2}^{L} k^{-t}} \quad (3.4)$$

Ou seja, $P_{\{\mathbf{X}\neq\mathbf{X}melhor\}}$ é igual à probabilidade do bit escolhido para mutar não ser aquele cujo k=1. Esta, por sua vez, é igual à probabilidade complementar da probabilidade do bit escolhido ser aquele cujo k=1. Na equação 3.4, o termo $P_{\{\Omega\}}$ representa a probabilidade do assim chamado “evento certo”, ou seja, a soma das probabilidades de todos os eventos possíveis (no caso, k=1,2,...,L). Tradicionalmente, $P_{\{\Omega\}}$=1. No caso em questão, no entanto, $P_{\{\Omega\}}$ é dado por $\sum_{k=1}^{L} k^{-t} = 1 + \sum_{k=2}^{L} k^{-t}$, sendo maior do que 1. Torna-se necessário, portanto, utilizá-lo como denominador de todas as probabilidades calculadas pela fórmula $k^{-\tau}$, a fim de adequá-las ao domínio convencional de qualquer probabilidade, que é o intervalo [0, 1].

A Tabela 3.1, a seguir, apresenta valores para esta probabilidade, para diferentes valores de τ e diferentes número de bits para a codificação das variáveis, L.

Tabela 3.1 - Variação de $P_{\{\mathbf{X}\neq\mathbf{Xmelhor}\}}$ com τ e L

τ	L		
	10	20	100
0	0,900	0,950	0,990
1	0,659	0,722	0,807
2	0,355	0,373	0,388
3	0,165	0,167	0,168
4	0,076	0,076	0,076
5	0,036	0,036	0,036

A partir da Tabela 3.1, é possível perceber que para τ>3, o número de bits L utilizado na codificação deixa de ter influência sobre $P_{\{\mathbf{X}\neq\mathbf{Xmelhor}\}}$. Antes disso, para τ<3, L influencia $P_{\{\mathbf{X}\neq\mathbf{Xmelhor}\}}$, fazendo-o aumentar à medida que L aumenta. Para τ=0 (*random walk*), por exemplo, $P_{\{\mathbf{X}\neq\mathbf{Xmelhor}\}}$ passa de 0,900 com L=10 para 0,990 com L=100. Analisando agora a influência sobre $P_{\{\mathbf{X}\neq\mathbf{Xmelhor}\}}$ do parâmetro τ para um mesmo L, tem-se que $P_{\{\mathbf{X}\neq\mathbf{Xmelhor}\}}$ cai à medida que τ aumenta. Para L=10, por exemplo, $P_{\{\mathbf{X}\neq\mathbf{Xmelhor}\}}$ cai de 0,900 com τ=0 para 0,036 com τ=5. Tendo em vista que τ define o grau de determinismo da busca, este resultado é coerente. Assim, para τ=0, a busca é totalmente aleatória e o valor de F(**X**) não tem qualquer importância, fazendo com que existam 9 chances em 10 de $\mathbf{X}_{\mathbf{melhor}}$ não ser escolhido para **X**. Por outro lado, para τ=5, essa chance cai para aproximadamente 4 em 100, evidenciando que com τ=5 o determinismo da busca é bem maior.

Da mesma forma que o GEO, o GEO_{var} também se beneficia com a modificação sugerida. Apresentam-se a seguir, os passos do algoritmo GEO_{var} já com as alterações. A fim de diferenciá-lo do original, esta variante é denominada GEO_{var1}. As alterações estão realçadas em negrito.

Os passos para o GEO_{var1} são:

1. Inicialize aleatoriamente N seqüências binárias C_j, $j\in\{1,2,...,N\}$ que codificam as N variáveis de projeto X_j. Cada C_j tem l_j bits, ou seja, o elemento $c_{j,i}\in C_j$, $i\in\{1,2,..., l_j\}$

e $c_{j,i} \in \{0,1\}$. Assim, o vetor **X** contendo as variáveis de projeto é codificado em uma matriz **C** cuja linha j contém a seqüência binária C_j utilizada na codificação da variável X_j. Converta **C** em **X** e calcule o valor da função objetivo F(**X**) e faça $\mathbf{X_{melhor}} = \mathbf{X}$ e $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$.

2. **Faça $\mathbf{F(X_{melhor})_{anterior} = F(X_{melhor})}$.** Para cada variável $j \in \{1,2,...,N\}$ faça:

    a. Para cada bit $i \in \{1,2,..., l_j\}$ da seqüência C_j, faça:

        1º. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule $F(\mathbf{X_i})$. **Em seguida, se $\mathbf{F(X_i) < F(X_{melhor})}$ então faça $\mathbf{F(X_{melhor}) = F(X_i)}$ e $\mathbf{X_{melhor} = X_i}$.**

        2º. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - \mathbf{F(X_{melhor})_{anterior}}$, que indica o ganho (ou perda) obtido ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.

        3º. Retorne o bit i ao seu valor original.

    b. Ordene os bits $i \in \{1,2,..., l_j\}$, ou seja, os bits da variável j, de acordo com os seus índices de adaptação $\Delta F(\mathbf{X_i})$, de k=1 para o menor $\Delta F(\mathbf{X_i})$ até $k=l_j$, para o maior $\Delta F(\mathbf{X_i})$, onde l_j é o número de bits da variável j.

    c. Crie uma lista $(i,k)_j$, relacionando a posição física do bit i em C_j com seu número de ordem k. Se $\Delta F(\mathbf{X_i})_m = \Delta F(\mathbf{X_i})_n$, $\forall m?n$, estabeleça a ordem entre m e n de forma aleatória.

    d. Escolha com igual probabilidade um bit candidato $i \in \{1,2,..., l_j\}$ para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual a $P_i(k)$, confirme o bit para mutar. Repetir este passo até que um bit seja confirmado para ser modificado. Quando isso acontecer, faça $i_{esc_j} = i$.

3. Faça **C** = $\mathbf{C_{esc}}$, onde $\mathbf{C_{esc}}$ é a configuração resultante da mutação em **C** dos N bits escolhidos no passo 2.d (bits i_{esc_j}, $j \in \{1,2,...,N\}$). Converta **C** em **X** e calcule F(**X**). **Se F(X) < $\mathbf{F(X_{melhor})}$ então faça $\mathbf{F(X_{melhor}) = F(X)}$ e $\mathbf{X_{melhor} = X}$.**
4. Repita os passos 2 e 3 até que um dado critério de parada seja satisfeito.
5. Retorne $\mathbf{X_{melhor}}$ e F($\mathbf{X_{melhor}}$).

A cada iteração do GEO_{var} original, é possível calcular as N probabilidades $P_{|j}$ de o bit escolhido para mutar, em cada variável j, não ser aquele que leva à melhor solução encontrada dentre os l_j bits da variável j. Estas probabilidades são dadas por:

$$P_{|j} = \frac{P_{\{k_{esc} \neq 1\}|j}}{P_{\{\Omega\}|j}} = \frac{P_{\{\Omega\}|j}}{P_{\{\Omega\}|j}} - \frac{P_{\{k_{esc}=1\}|j}}{P_{\{\Omega\}|j}} = 1 - \frac{1}{1 + \sum_{k=2}^{l_j} k^{-t}} \quad , \; j \in \{1,2,..., N\} \tag{3.6}$$

Para GEO_{var}, o cálculo da probabilidade $P_{\{\mathbf{X} \neq \mathbf{X}melhor\}}$ da melhor solução não ser escolhida, a cada iteração, é mais difícil ou, até mesmo, impossível. Isto ocorre porque, a cada iteração do GEO_{var}, ao final do processo de escolha dos N bits a mutar, todos os N bits são mutados simultaneamente. Isto leva o algoritmo a um ponto totalmente novo no espaço de busca (espaço de variáveis de projeto). Ponto este que não teve o valor da função objetivo calculado durante as N ordenações da iteração atual. Assim, não é possível utilizar a equação 3.6 para calcular tal probabilidade.

As versões original e modificada do GEO e do GEO_{var} foram aplicadas a um conjunto de 5 funções teste, doravante denominadas FT_i, $i \in \{1,2,...,5\}$. Estas funções são amplamente usadas para testar algoritmos de otimização (Potter e De Jong, 1994) e todas possuem o valor zero como mínimo global. Os algoritmos GEO e GEO_{var} originais já foram aplicados a estas mesmas funções (Sousa e Ramos, 2002). Os testes efetuados utilizam, para cada uma das funções teste, os mesmos valores de τ utilizados no referido artigo (Sousa e Ramos, 2002). A tabela 3.2 define cada uma destas funções teste, incluindo o número N de variáveis de cada uma, a solução ótima $\mathbf{X}^*$ e o valor de F($\mathbf{X}^*$).

Tabela 3.2 – Funções teste para minimização

FT	Min. F(**X**)	N[1]	Restrições	Descrição	$\mathbf{X}^{*}$[2] $F(\mathbf{X}^{*})$
FT_1	$F(\mathbf{X}) = 100\,(X_1^2 - X_2)^2 + (1 - X_1)^2$	2	$-2{,}048 \le X_i \le 2{,}048$	Função de Rosenbrock	$\mathbf{X}^{*} = [\,1\;\;1\,]$ $F(\mathbf{X}^{*}) = 0$
FT_2	$F(\mathbf{X}) = 3\,N + \sum_{i=1}^{N}\left(X_i^2 - 3\cos(2pX_i)\right)$	20	$-5{,}12 \le X_i \le 5{,}12$	Função de Rastringin	$X_i^{*} = 0$ $F(\mathbf{X}^{*}) = 0$
FT_3	$F(\mathbf{X}) = 418{,}9829\,N - \sum_{i=1}^{N} X_i \sin\left(\sqrt{\lvert X_i\rvert}\right)$	10	$-500 \le X_i \le 500$	Função de Schwefel [3]	$X_i^{*} = 420{,}9687$ $F(\mathbf{X}^{*}) = 0$ [3]
FT_4	$F(\mathbf{X}) = 1 + \sum_{i=1}^{N}\frac{X_i^2}{4000} - \prod_{i=1}^{N}\cos\left(\frac{X_i}{\sqrt{i}}\right)$	10	$-600 \le X_i \le 600$	Função de Griewangk	$X_i^{*} = 0$ $F(\mathbf{X}^{*}) = 0$
FT_5	$F(\mathbf{X}) = 20 + e - 20\,e^{\left(-0{,}2\sqrt{\frac{1}{N}\sum_{i=1}^{N}X_i^2}\right)} - e^{\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2pX_i)\right)}$	30	$-30 \le X_i \le 30$	Função de Ackley	$X_i^{*} = 0$ $F(\mathbf{X}^{*}) = 0$

[1] Número de dimensões (variáveis); [2] $\mathbf{X}^{*}$=Ponto de mínimo global; [3] Verificou-se, durante testes posteriores, que a solução X_i^{*}=420,9687 leva a $F(\mathbf{X}^{*})= 1{,}272783756576246\cdot10^{-4}$ quando calculada com dupla precisão e não $F(\mathbf{X}^{*})=0$. Além disso, o menor valor encontrado foi $F(\mathbf{X})= 1{,}272756617254\cdot10^{-4}$ para dezenas de soluções, todas obedecendo o padrão X_i=420,96874ddddd, onde "ddddd" varia de "47505" a "80212".

As Figuras 3.4 até 3.8 apresentam os resultados médios de 50 execuções independentes, obtidos com as versões aprimoradas do GEO e do GEO_{var}, denominadas GEO_1 e GEO_{var1}. O número de avaliações de F(**X**) a cada execução é 10^4 para FT_1 e 10^5 para as demais funções teste. Para a codificação das variáveis de projeto de todas as funções teste foi utilizado l_j=l=16 bits por variável. Em cada figura, são apresentadas também as curvas obtidas com o GEO e o GEO_{var} original.

As curvas obtidas mostram que, de fato, houve melhora no desempenho de GEO e GEO_{var} com a modificação e isto para todas as FTs. A modificação teve maior impacto nas funções de Rosenbrock (FT_1) e de Schwefel (FT_3), sendo a função de Ackley (FT_5) aquela de menor sensibilidade à modificação. Outra constatação importante, mostrada pelas curvas, é a de que a melhora nos resultados ocorre de forma sustentada, ou seja, do início até o final da busca.

Em virtude da natureza da modificação introduzida pela SA1, é impossível a mesma ocasionar piora no desempenho em relação ao GEO ou GEO_{var} original, em termos do melhor F(**X**) encontrado. O motivo é que a modificação SA1 não altera os pontos do espaço de variáveis

de projeto visitados pelo algoritmo, eles continuam sendo exatamente os mesmos, do início ao final. O que ocorre é que, com SA1, todas as avaliações de F(**X**) são monitoradas e a melhor delas é armazenada. Portanto, no pior caso, ambas as versões do algoritmo apresentarão resultados idênticos para alguma função objetivo em particular. Por este motivo, os testes envolvendo as demais sugestões de aprimoramento passarão a usar a versão SA1 como referência.

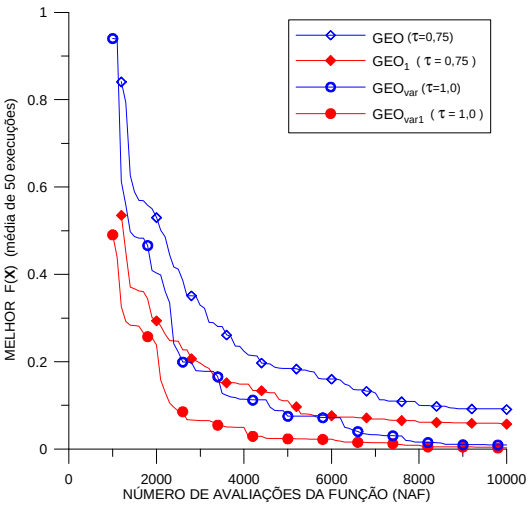


Figura 3.4 - FT_1 x NAF

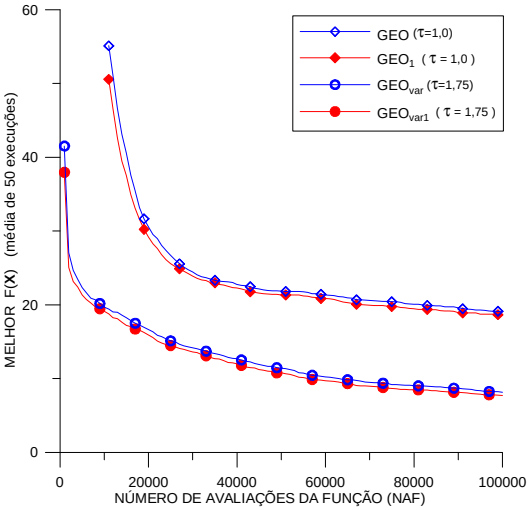


Figura 3.5 - FT_2 x NAF

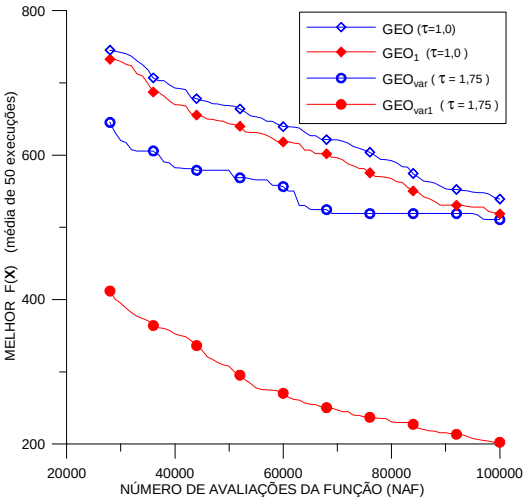


Figura 3.6 - FT_3 x NAF

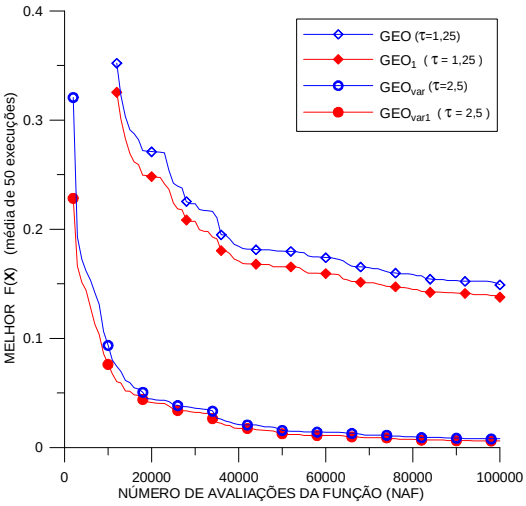


Figura 3.7 - FT_4 x NAF

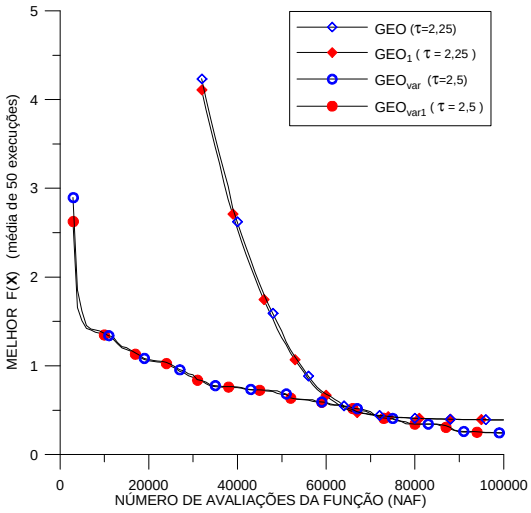


Figura 3.8 - FT_5 x NAF

3.7.2 – Sugestão de Aprimoramento 2 (SA2)

O segundo aprimoramento sugerido tem em vista o algoritmo GEO especificamente e introduz neste algoritmo a capacidade de auto-reinicialização da configuração corrente durante a busca. Esta funcionalidade permite ao algoritmo monitorar, ao longo da busca, o decaimento do valor da função objetivo. Se em algum momento este decaimento estagnar, a auto-reinicialização entra em ação, sorteando um novo ponto no espaço das variáveis de projeto a partir do qual a busca continuará.

Hipoteticamente, ao longo de uma busca o GEO pode, depois de algum tempo, “estacionar”, ficar preso no que pode ser chamado de “pontos de estagnação”. Tais pontos de estagnação ocorrem quando o algoritmo já se encontra em um ponto do espaço de busca tal que, ao realizar o ordenamento dos bits, todos levam a “pontos piores”, ou seja, pontos para os quais o valor da função objetivo é maior. Sob este aspecto, tal ponto pode ser considerado um ponto de mínimo local “relativo”. O termo relativo é usado para expressar que, embora não seja necessariamente um mínimo da função objetivo, é um ponto de mínimo local do ponto de vista do GEO, pois de todos os pontos ao qual ele tem acesso numa dada iteração, aquele é o de menor valor da função objetivo. Como a cada iteração o GEO modifica um bit obrigatoriamente, isto faz com que o GEO saia do mínimo local relativo onde se encontra, indo para um dos “pontos piores”. Se na iteração seguinte o ordenamento dos bits também levar a “pontos piores”, a única exceção será o bit que foi mudado na iteração anterior. Este, ao ser modificado, leva de volta ao ponto de mínimo

local relativo onde o algoritmo estava antes. Sendo assim, se o valor de τ for alto, é grande a probabilidade do algoritmo modificar novamente o mesmo bit e retornar ao ponto de mínimo local relativo. Se isto ocorrer, o algoritmo volta à exata condição que tinha quando chegou ao mínimo local relativo pela primeira vez, caracterizando assim, um círculo vicioso ou círculo de estagnação, onde o algoritmo repete várias vezes os mesmos movimentos no espaço de busca e perde em eficiência. Os pontos de mínimo local relativos para os quais o fenômeno cíclico recém descrito ocorre definem os assim chamados pontos de estagnação. Já o fenômeno cíclico em si, ou seja, o círculo de estagnação, tem probabilidade de ocorrência como função do parâmetro τ. Quanto maior for τ, maior a probabilidade do GEO ficar preso em "círculos de estagnação", pois com τ's elevados diminui a chance do algoritmo deixar de escolher o ponto que leva de volta ao ponto de estagnação, ou seja, ao mínimo local relativo. Além disso, a chance da estagnação acontecer no início da busca é menor, visto que, nesse estágio, o espaço de busca ainda está começando a ser explorado e é provável que o algoritmo leve algum tempo antes de encontrar um mínimo local relativo. Nem todo ponto de mínimo local relativo é necessariamente ponto de estagnação. Como visto, um ponto de mínimo local relativo pode ser identificado em apenas uma iteração do algoritmo, ao passo que um ponto de estagnação precisa de pelo menos duas iterações para ser identificado.

É importante ressaltar que, apesar de não haver correspondência direta entre mínimos locais reais da função objetivo e "mínimos locais relativos", é provável que pelo menos alguns "mínimos locais relativos" sejam mínimos locais da função objetivo, sendo que quanto mais próximo o mínimo local for do mínimo global, em termos do valor da função objetivo, maior a chance dele ser um ponto de estagnação. O mínimo global é o caso extremo que ajuda a visualizar a situação. Se, durante a busca, o algoritmo encontra-se no ponto de mínimo global da função objetivo, então, ao realizar o ordenamento dos bits, todos, obrigatoriamente, levarão à "pontos piores". Sob esse aspecto, o mínimo global pode ser visto como o "maior" dos pontos de estagnação. Para os mínimos locais cujo valor é próximo ao do mínimo global, a chance de um deles ser um ponto de estagnação também é grande, visto que devem ser pequenas as regiões do espaço de busca que possuem valores menores para a função objetivo. Para os mínimos locais cujos valores são distantes ao do mínimo global, a chance de um deles ser um ponto de estagnação é menor, visto que

devem ser grandes as regiões do espaço de busca que possuem valores menores para a função objetivo.

Apresentam-se, a seguir, os passos do algoritmo GEO já com as alterações. Esta variante é denominada GEO_2. Conforme mencionado na seção 3.7.1, GEO_2 foi desenvolvido tomando GEO_1 como base. As alterações estão realçadas em negrito.

O algoritmo GEO_2 é composto dos seguintes passos:

1. Inicialize aleatoriamente uma seqüência binária **C**, de comprimento L que codifica N variáveis de projeto. Dentro de **C**, cada variável de projeto X_j é codificada em uma parcela com comprimento l_j, $j \in \{1,2,...,N\}$ e tal que $\sum_j l_j = L$. Converta **C** em **X** e calcule o valor da função objetivo F(**X**) e faça $\mathbf{X_{melhor}} = \mathbf{X}$ e $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$. **Faça $\mathbf{E = 0}$.**

2. Faça $F(\mathbf{X_{melhor}})_{anterior} = F(\mathbf{X_{melhor}})$ **e $\mathbf{F(X)_{anterior} = F(X)}$.** Para cada bit $i \in \{1,2,...,L\}$ da seqüência **C**, faça:

    a. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule $F(\mathbf{X_i})$. Em seguida, se $F(\mathbf{X_i}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X_i})$ e $\mathbf{X_{melhor} = X_i}$.

    b. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - F(\mathbf{X_{melhor}})_{anterior}$, que indica o ganho (ou perda) que se têm ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.

    c. Retorne o bit i ao seu valor original.

3. Ordene todos os bits $i \in \{1,2,...,L\}$ de acordo com os seus índices de adaptação, de k=1 para o menos adaptado à k=L, para o mais adaptado. Crie uma lista de valores (i,k) relacionando a posição física i do bit em **C** com seu respectivo número de ordem k, onde $i,k \in \{1,2,...,L\}$. Se $\Delta F(\mathbf{X_i}) = \Delta F(\mathbf{X_j})$, i?j, estabeleça a ordem entre i e j de forma aleatória.

4. Escolha com igual probabilidade um bit candidato i para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual à $P_i(k)$, confirme o bit i para ser modificado. Repetir este passo até que um bit seja confirmado. Quando isso acontecer, faça $i_{esc} = i$.
5. Faça $\mathbf{C} = \mathbf{C}_{\mathbf{iesc}}$, ou seja, em **C**, mute o bit i_{esc}. Converta **C** em **X** e calcule F(**X**).
6. **Se** $F(X) \geq F(X)_{anterior}$ **e** $F(X_{melhor}) = F(X_{melhor})_{anterior}$ **faça E = E+1 senão, faça E = 0. Se E =** E_{lim} **,** $E_{lim} \in \{1,2,...\}$**, faça E = 0 e reinicialize aleatoriamente a seqüência binária C, converta C em X e calcule F(X). Se** $F(X) < F(X_{melhor})$ **faça** $F(X_{melhor}) = F(X)$ **e** $X_{melhor} = X$**.**
7. Repita os passos 2 a 6 até que um dado critério de parada seja satisfeito.
8. Retorne $\mathbf{X}_{\mathbf{melhor}}$ e $F(\mathbf{X}_{\mathbf{melhor}})$.

No passo 6 do algoritmo acima, "E" conta o número de iterações consecutivas do algoritmo em que não houve melhora, nem no valor da função objetivo F(**X**) atual com relação à $F(\mathbf{X})_{anterior}$, nem no valor $F(\mathbf{X}_{\mathbf{melhor}})$ com relação à $F(\mathbf{X}_{\mathbf{melhor}})_{anterior}$. Se o valor de E for igual ao valor limite E_{lim}, previamente definido, então ocorre a reinicialização da seqüência binária C.

Conforme já mencionado, a versão SA2 foi desenvolvida tendo em vista apenas o GEO. O algoritmo GEO_{var}, pelo fato de realizar a mutação simultânea de N bits ao invés de um, deve possuir uma espécie de "imunidade natural" ao problema da estagnação. De todo modo, movido pela curiosidade, decidiu-se implementar a versão SA2 também para o GEO_{var}, a fim de ver o que de fato acontece.

Os passos para o GEO_{var2} são:

1. Inicialize aleatoriamente N seqüências binárias C_j, $j \in \{1,2,...,N\}$ que codificam as N variáveis de projeto X_j. Cada C_j tem l_j bits, ou seja, o elemento $c_{j,i} \subset C_j$, $i \in \{1,2,..., l_j\}$ e $c_{j,i} \in \{0,1\}$. Assim, o vetor **X** contendo as variáveis de projeto é codificado em uma matriz **C** cuja linha j contém a seqüência binária C_j utilizada na codificação da

variável X_j. Converta **C** em **X** e calcule o valor da função objetivo F(**X**) e faça $\mathbf{X_{melhor}} = \mathbf{X}$ e $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$. **Faça E = 0.**

2. Faça $F(\mathbf{X_{melhor}})_{anterior} = F(\mathbf{X_{melhor}})$ **e** $\mathbf{F(X)_{anterior} = F(X).}$ Para cada variável $j \in \{1,2,...,N\}$ faça:

    a. Para cada bit $i \in \{1,2,..., l_j\}$ da seqüência C_j, faça:

        1º. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule $F(\mathbf{X_i})$. Em seguida, se $F(\mathbf{X_i}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X_i})$ e $\mathbf{X_{melhor}} = \mathbf{X_i}$.

        2º. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - F(\mathbf{X_{melhor}})_{anterior}$, que indica o ganho (ou perda) obtido ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.

        3º. Retorne o bit i ao seu valor original.

    b. Ordene os bits $i \in \{1,2,..., l_j\}$, ou seja, os bits da variável j, de acordo com os seus índices de adaptação $\Delta F(\mathbf{X_i})$, de k=1 para o menor $\Delta F(\mathbf{X_i})$ até $k=l_j$, para o maior $\Delta F(\mathbf{X_i})$, onde l_j é o número de bits da variável j.

    c. Crie uma lista $(i,k)_j$, relacionando a posição física do bit i em C_j com seu número de ordem k. Se $\Delta F(\mathbf{X_i})_m = \Delta F(\mathbf{X_i})_n$, $\forall m?n$, estabeleça a ordem entre m e n de forma aleatória.

    d. Escolha com igual probabilidade um bit candidato $i \in \{1,2,..., l_j\}$ para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual a $P_i(k)$, confirme o bit para mutar. Repetir este passo até que um bit seja confirmado para ser modificado. Quando isso acontecer, faça $i_{esc_j} = i$.

3. Faça $\mathbf{C} = \mathbf{C_{esc}}$, onde $\mathbf{C_{esc}}$ é a configuração resultante da mutação em **C** dos N bits escolhidos no passo 2.d (bits i_{esc_j}, $j \in \{1,2,...,N\}$). Converta **C** em **X** e calcule F(**X**). Se $F(\mathbf{X}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$ e $\mathbf{X_{melhor}} = \mathbf{X}$.

4. **Se $F(X) \geq F(X)_{anterior}$ e $F(X_{melhor}) = F(X_{melhor})_{anterior}$ faça E = E+1 senão, faça E = 0. Se E = E_{lim} , $E_{lim} \in \{1,2,...\}$, faça E = 0 e reinicialize aleatoriamente as seqüências binárias C_j, $j \in \{1,2,...,N\}$, converta C em X e calcule F(X). Se $F(X) < F(X_{melhor})$ faça $F(X_{melhor}) = F(X)$ e $X_{melhor} = X$.**
5. Repita os passos 2 a 4 até que um dado critério de parada seja satisfeito.
6. Retorne $\mathbf{X}_{melhor}$ e $F(\mathbf{X}_{melhor})$.

As versões GEO_2 e GEO_{var2} foram aplicadas ao conjunto de funções teste FT_i, $i \in \{1,2,...,5\}$ descritas na seção 3.7.1.

Inicialmente, é feita uma busca do tipo varredura a fim de verificar a influência dos parâmetros τ e E_{lim} nas versões SA2 de GEO e GEO_{var}. Para τ, a varredura deu-se no intervalo de 0 a 10 com passo 0,25. Para E_{lim}, utilizou-se $E_{lim} \in \{1;2;3;4;20;100;1000\}$. É importante notar que quando $E_{lim} \rightarrow \infty$, SA2$\rightarrow$SA1, ou seja, a versão SA2 torna-se idêntica à versão SA1 a partir da qual foi desenvolvida. Para a maioria das implementações práticas, isto ocorre para valores finitos de E_{lim}. De fato, para os testes efetuados, o valor E_{lim}=1000 garante que nenhuma reinicialização seja aplicada, e isto para todas as FT's. Com isto, a menos do desempenho temporal, as buscas efetuadas por SA2 com E_{lim}=1000 equivalem a buscas efetuadas por SA1. O motivo de utilizar-se E_{lim}=1000 é exatamente o de ter resultados equivalentes ao da versão SA1 para utilizar como referência na comparação com a versão SA2.

As Figuras 3.9 até 3.13 apresentam os resultados médios de 50 execuções independentes, em termos do melhor $F(\mathbf{X})$ encontrado ao final de cada execução, obtidos com as versões GEO_2 e GEO_{var2}. O número de avaliações de $F(\mathbf{X})$ a cada execução é 10^4 para FT_1 e 10^5 para as demais funções teste. Nas figuras, cada curva apresenta o resultado para um dado valor de E_{lim}, conforme indicado, e para τ variando de 0 a 10. Visando maior clareza, para $E_{lim} \in \{1,2,3,4\}$ apenas duas das quatro curvas obtidas são apresentadas e, para a Figura 3.9, escalas diferentes são utilizadas.

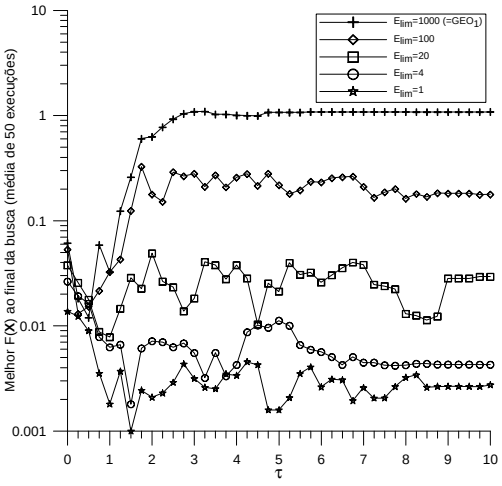


Figura 3.9a - FT_1 x τ x E_{lim} para GEO_2

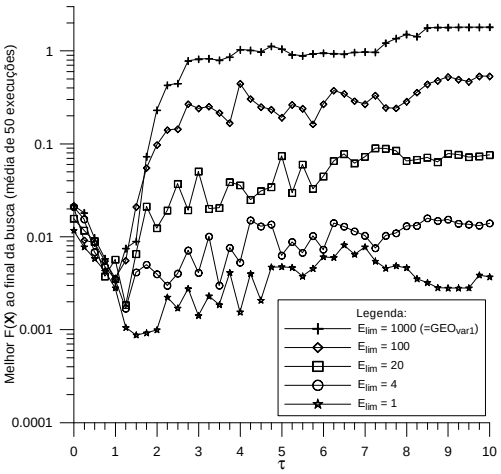


Figura 3.9b - FT_1 x τ x E_{lim} para GEO_{var2}

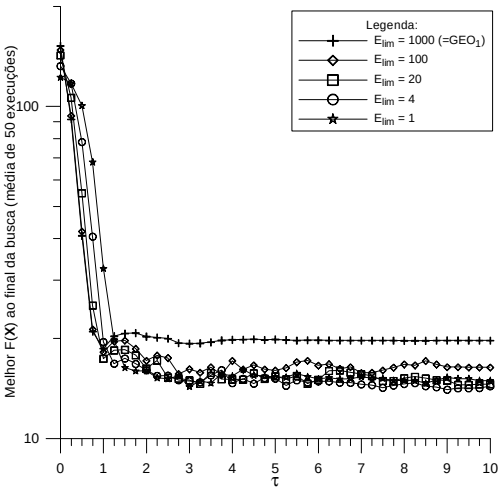


Figura 3.10a - FT_2 x τ x E_{lim} para GEO_2

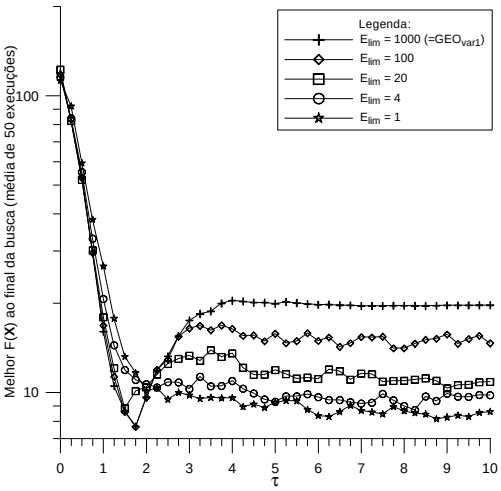


Figura 3.10b - FT_2 x τ x E_{lim} para GEO_{var2}

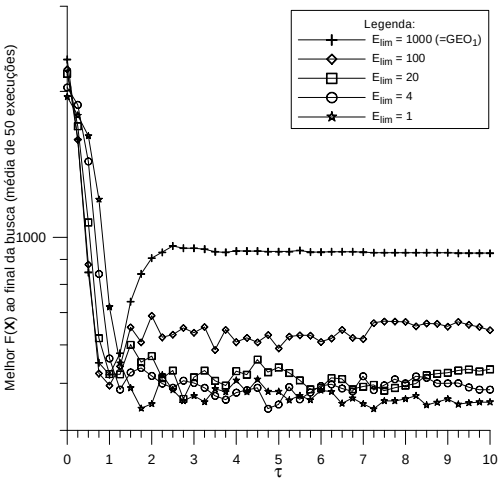


Figura 3.11a - FT_3 x τ x E_{lim} para GEO_2

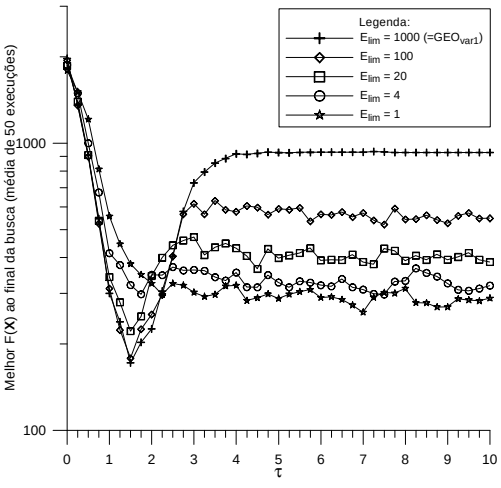


Figura 3.11b - FT_3 x τ x E_{lim} para GEO_{var2}

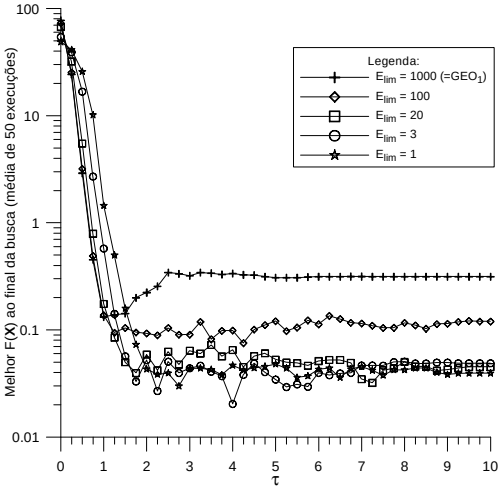


Figura 3.12a - FT_4 x τ x E_{lim} para GEO_2

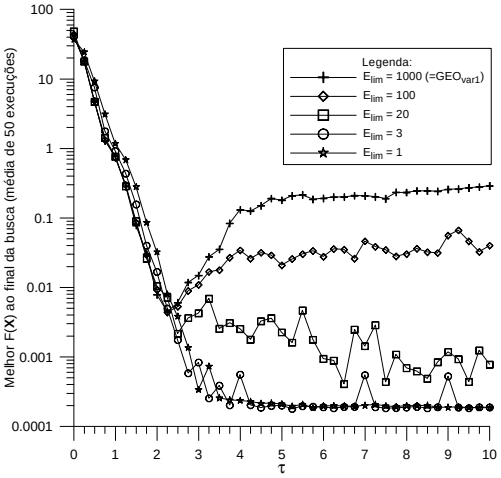


Figura 3.12b - FT_4 x τ x E_{lim} para GEO_{var2}

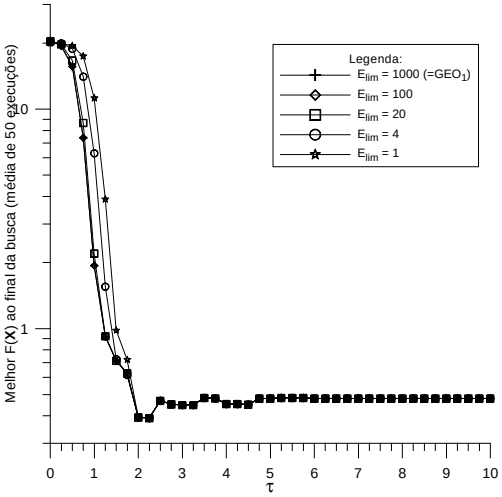


Figura 3.13a - FT_5 x τ x E_{lim} para GEO_2

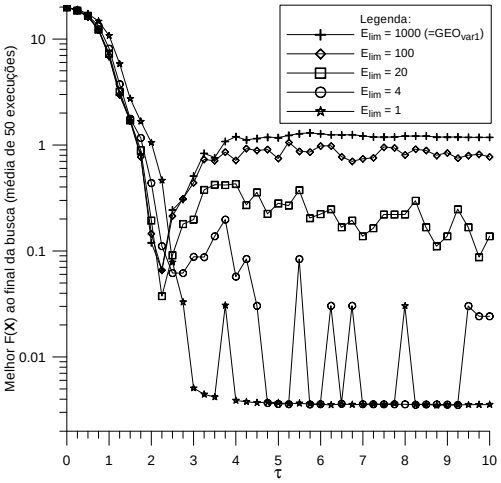


Figura 3.13b - FT_5 x τ x E_{lim} para GEO_{var2}

Analisando primeiramente os resultados obtidos para GEO_2 (Figuras 3.9a a 3.13a), em termos do melhor resultado de todos, observa-se que há melhora do desempenho, relativamente ao GEO_1 (ou seja, GEO_2 com E_{lim}=1000), e isto para todas as funções teste menos FT_5, onde ambas as versões empatam. Em termos gerais, o efeito mais evidente é a melhora do desempenho à medida que E_{lim} diminui. Outro efeito muito importante é a relativa "estabilidade" de desempenho que GEO_2 apresenta para τ>2. Este fato fortalece a hipótese da ocorrência de pontos de estagnação para o GEO, principalmente para τ's elevados, conforme havia sido antecipado.

Analisando agora os resultados obtidos para GEO_{var2} (Figuras 3.9b a 3.13b), em termos do melhor resultado de todos, observa-se que também para esta versão há melhora de desempenho relativamente ao GEO_{var1} (ou seja, GEO_{var2} com E_{lim}=1000). Isto ocorre

para as funções teste FT_1, FT_4 e FT_5. Além disso, é possível observar que GEO_{var2} também apresenta certa estabilidade de desempenho para $\tau>4,0$. Para FT_2 o melhor resultado ocorre com $E_{lim}=100$, mas é muito próximo ao obtido com $E_{lim}=1000$ (diferença de 0,6%). Para FT_3, o melhor resultado ocorre para $E_{lim}=1000$. Para estas duas funções teste, portanto, pode-se dizer que GEO_{var1} obtém vantagem. Para GEO_{var2}, o efeito geral mais evidente é também a melhora do desempenho à medida que E_{lim} diminui, mas com exceções. As exceções no caso, são FT_2 e FT_3. Nestas duas funções teste, para valores de τ até ~2,5 ocorre piora no desempenho à medida que E_{lim} diminui. Somente para valores de τ maiores que 2,5 é que a situação se inverte, ocorrendo melhora do desempenho à medida que E_{lim} diminui. Como o valor ótimo de τ para estas duas funções teste ocorre para $\tau<2,5$, então o uso de GEO_{var2} nestas duas funções ocasiona perda de desempenho. De todo modo, não deixa de ser surpreendente que também para GEO_{var}, que se pensava "imune" à estagnação, GEO_{var2} tenha propiciado ganho de desempenho, ainda que restrito a 3 das 5 funções teste.

A Tabela 3.3 a seguir apresenta os valores ótimos de τ e E_{lim} para GEO_2 e GEO_{var2}, ou seja, aqueles com os quais obteve-se o melhor resultado para cada função teste. Por melhor resultado entenda-se o menor valor médio da respectiva função teste ao final de 50 execuções independentes. Como referência, os valores ótimos de τ para GEO_1 e GEO_{var1} também são mostrados. A Tabela 3.3 informa também a média R do número de reinicializações ocorrido nas 50 execuções independentes. Para FT_5 com GEO_2, houve empate, tendo sido o mesmo e melhor resultado obtido para $E_{lim}\in\{3;4;20;100;1000\}$ e todos com $\tau=2,25$ conforme apresentado na tabela. Todavia, para $E_{lim}\in\{1;2\}$, os resultados foram piores que os demais por uma diferença ínfima (0,013%). Para FT_2 e FT_3 com GEO_{var2}, os melhores resultados foram obtidos com $E_{lim}\geq100$. Como a versão SA1 pode ser vista como um caso particular da SA2 para valores elevados de E_{lim} e tendo em vista que a ênfase da versão SA2 está na reinicialização, decidiu-se, nestes dois casos, informar também, mediante valores entre parênteses na Tabela 3.3, os valores ótimos para GEO_{var2} considerando E_{lim} restrito a $E_{lim}\in\{1;2;3;4\}$, a fim de utilizá-los em futuras comparações entre GEO_{var1} e GEO_{var2}.

Tabela 3.3 – Valores ótimos de τ e E_{lim} e número médio de reinicializações para cada função teste[1].

FT	GEO_1	GEO_2			GEO_{var1}	GEO_{var2}		
	τ	τ	E_{lim}	R	τ	τ	E_{lim}	R
FT_1	0,5	1,5	1	42,1	1,0	1,5	1	61,2
FT_2	1,0	5,5	3	2,00	1,75	1,75 (8,75)	100 (1)	0,78 (34,7)
FT_3	1,0	7,25	1	10,0	1,5	1,5 (7,0)	1000 (1)	0,0 (69,5)
FT_4	1,0	4,0	3	9,24	2,25	5,25	3	53,8
FT_5	2,25	2,25	{3;4;20; 100;1000}[1)]	{0,26;0,2; 0,02;0;0}	2,25	8,75	1	16,2

[1] Convenção: {.}= o melhor resultado foi obtido para mais de um valor do parâmetro; (.) = valor do parâmetro que leva ao melhor resultado, condicionado a $E_{lim} \leq 4$

Lembrando que a SA2 com Elim= 1000 corresponde à SA1, a análise dos dados da Tabela 3.3 para GEO_2 corrobora a análise feita com base nas Figuras 3.9a a 3.13a, qual seja, a de que GEO_2 obteve desempenho superior (4 em 5) ou equivalente (1 em 5) a GEO_1 para todas as funções teste. Além disso, a tabela mostra que para a maioria das funções teste (4 em 5), o τ ótimo de GEO_2 ocorre para $\tau>2$, ou seja, dentro da região de estabilidade onde o desempenho de GEO_2 não varia significativamente com o parâmetro τ. Isto é importante, pois sugere que é possível utilizar-se *a priori* um valor elevado de τ e efetuar a varredura para descobrir o valor ótimo de E_{lim} apenas. O fato do E_{lim} ótimo de todas as funções teste para GEO_2 ter ocorrido para $E_{lim} \in \{1,2,3\}$ sugere ainda que também para este parâmetro a busca pode ser efetuada para um conjunto pequeno de valores. Já para GEO_{var2}, os dados da Tabela 3.3 mostram que a situação é mais complexa, pois além de inesperada, não há nos resultados nenhuma indicação do porque houve melhora no desempenho em 3 das 5 funções teste (FT_1,FT_4 e FT_5), e nem tampouco porque houve piora no desempenho em 2 das 5 funções teste (FT_2 e FT_3). É possível, entretanto, observar que, para as funções teste onde GEO_{var2} ocasionou melhora no desempenho, o valor ótimo para E_{lim} está restrito a um conjunto bastante pequeno ($E_{lim} \in \{1,2,3\}$). Isto, aliado à informação da relativa "estabilidade" de desempenho de GEO_{var2} observada nas Figuras 3.9b a 3.13b para $\tau>4,0$, permite

aventar como possível estratégia para o ajuste dos parâmetros τ e E_{lim} de GEO_{var2}, usar $\tau>4{,}0$ fixo e fazer varredura para $E_{lim}\in\{1;2;3\}$ e, adicionalmente, utilizar GEO_{var1} (ou GEO_{var2} com $E_{lim}\to\infty$) como referência quanto ao desempenho, fazendo neste último varredura para $\tau\in[0;10]$.

A fim de acessar o desempenho apresentado por GEO_2 e GEO_{var2} ao longo da busca, os valores compilados pela Tabela 3.3 são utilizados para gerar as curvas das Figuras 3.14 a 3.18, onde o melhor valor de FT_i *versus* o número de avaliações de FT_i $i\in\{1,2,...,5\}$ é mostrado, numa média de 50 execuções independentes, tanto para GEO_2 como para GEO_{var2}. Os valores ótimos dos parâmetros τ e E_{lim} para GEO_{var2} são tomados da Tabela 3.3 considerando E_{lim} restrito a $E_{lim}\in\{1;2;3;4\}$. Isto é feito a fim de permitir, conforme já mencionado, uma comparação entre SA2 e SA1. As curvas obtidas para GEO_1 e GEO_{var1} com seus respectivos valores ótimos do parâmetro τ também são apresentadas.

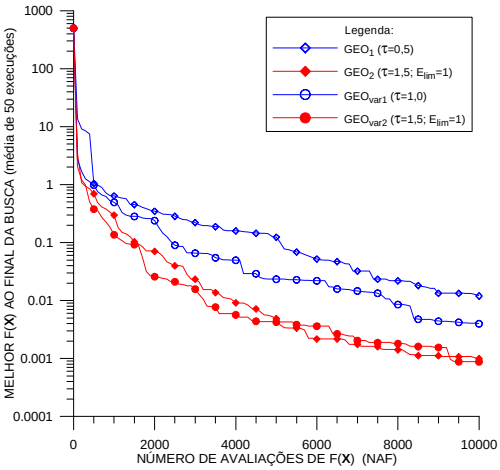


Figura 3.14 - FT_1 x NAF

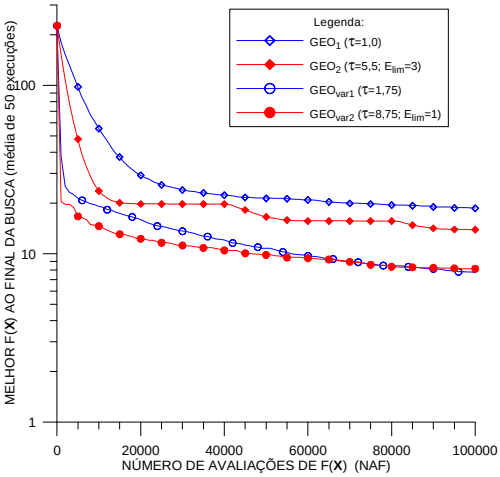


Figura 3.15 - FT_2 x NAF

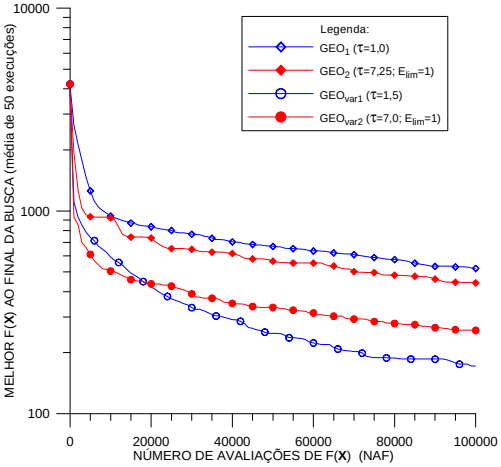


Figura 3.16 - FT_3 x NAF

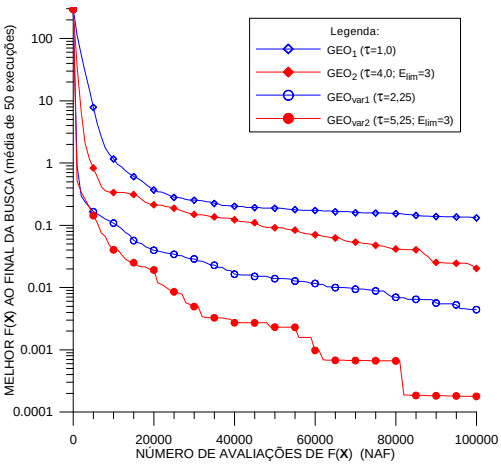


Figura 3.17 - FT_4 x NAF

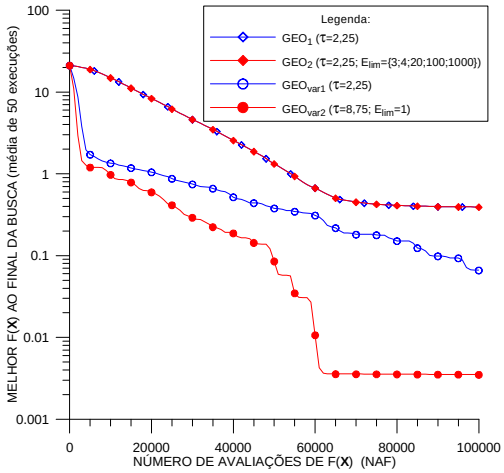


Figura 3.18 - FT_5 x NAF

As curvas das Figuras 3.14 a 3.18 mostram, tanto para GEO como GEO_{var}, que para todos os casos onde a versão SA2 foi melhor do que a versão SA1 no valor médio ao final da busca, conforme dado pela Tabela 3.3, também o foi ao longo da busca. Este resultado é importante, pois demonstra que a versão SA2 consegue sustentar sua vantagem para com a versão SA1 ao longo de toda a busca. Esta característica de sustentabilidade não ocorre naqueles casos em que a versão SA1 obteve valor médio da função objetivo ao final da busca melhor do que o da versão SA2. Tais casos ocorreram apenas para GEO_{var2}, com FT_2 e FT_3.

As curvas das Figuras 3.14 a 3.18 mostram também que, na comparação direta, GEO_2 obteve quatro vitórias e um empate frente a GEO_1 e isto com sustentabilidade. Na comparação direta entre GEO_{var2} e GEO_{var1}, o primeiro obteve três vitórias com sustentabilidade contra duas vitórias sem sustentabilidade do segundo. As vitórias com sustentabilidade podem ser consideradas como vitórias completas, visto que, nelas, um algoritmo é melhor do que o outro ao longo de toda a busca. As vitórias sem sustentabilidade, por sua vez, não gozam de tal prerrogativa. Nestas, o algoritmo "perdedor" começa melhor e se mantém assim até determinado ponto da busca, aqui chamado de ponto de equilíbrio. No ponto de equilíbrio, os dois algoritmos se equivalem em desempenho. Do ponto de equilíbrio em diante, é o algoritmo "vitorioso" que obtém o melhor desempenho, mantendo-se assim até o final. As duas vitórias sem sustentabilidade do GEO_{var1} sobre o GEO_{var2} ocorreram com as funções teste FT_2 e FT_3. Olhando as respectivas figuras (Figura 3.15 e Figura 3.16), é possível verificar que o ponto de

equilíbrio para FT_2 ocorreu com NAF≅20000 e para FT_3 ocorreu com NAF≅75000. A fim de verificar o que acontece para FT_2 e FT_3 com GEO_{var1} e GEO_{var2} numa busca de longo prazo, o número de avaliações limite, utilizado como critério de parada, foi aumentado 100 vezes, passando de 10^5 para 10^7 e 25 execuções independentes foram efetuadas para os dois algoritmos. As Figuras 3.19 e 3.20 apresentam os resultados médios obtidos, sendo que os mesmos mostram claramente que, para FT_2 e FT_3, GEO_{var1} apresenta características de convergência a longo prazo superiores à obtida com GEO_{var2}.

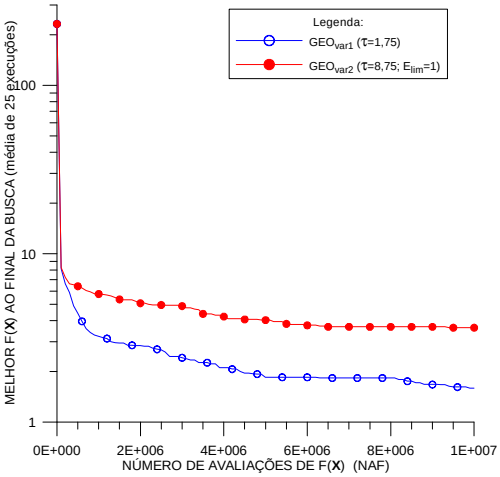


Figura 3.19 - FT_2 x NAF

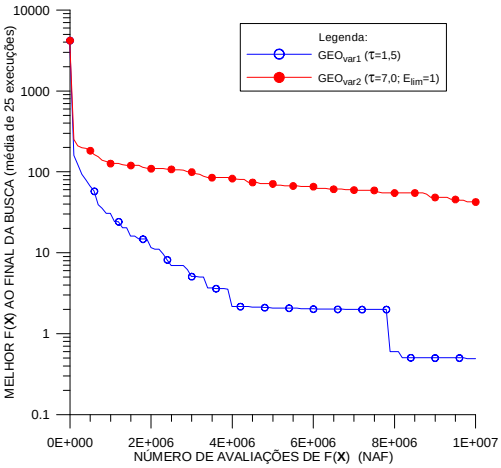


Figura 3.20 - FT_3 x NAF

A análise completa de todos os resultados obtidos nesta seção permite chegar à constatação de que a versão SA2 foi sempre superior à versão SA1 nos trechos iniciais das buscas. Esta constatação é importante, pois sugere que nos problemas em que os recursos disponíveis para as buscas são escassos, a versão SA2 é a escolha mais acertada. É possível também antever que um algoritmo híbrido das versões SA2 e SA1 obterá desempenho superior, visto que ele incorporará a maior rapidez de convergência da versão SA2 nos estágios iniciais da busca com a maior rapidez de convergência da versão SA1 nos estágios finais da busca. O híbrido de SA2 e SA1 pode ser obtido variando-se os parâmetros E_{lim} e τ de SA2 durante a busca. No início da busca, faz-se $E_{lim}=1$ e $\tau\rightarrow\infty$ (valor alto) por algum tempo, seguido por $E_{lim}=2$ e τ=valor mais baixo e assim por diante até que, no final da busca, faz-se $E_{lim}\rightarrow\infty$ e $\tau\rightarrow 0$ (valor baixo, não necessariamente zero). Esta estratégia garante um comportamento híbrido. No início da busca, obtém-se a rapidez de convergência de SA2 e no final da busca obtém-se a rapidez de convergência de SA1.

3.7.3 – Sugestão de Aprimoramento 3 (SA3)

O terceiro aprimoramento sugerido tem em vista os algoritmos GEO e sua variante GEO_{var} e introduz nestes o multiordenamento como forma de tornar a cobertura do espaço de busca, efetuada a cada ordenamento, menos assimétrica. Hipoteticamente, esta assimetria pode influenciar negativamente o desempenho destes algoritmos.

A assimetria na cobertura do espaço de busca, que ocorre em GEO e GEO_{var} durante a fase de cálculo do índice de adaptação dos bits, é ilustrada nas figuras a seguir. Inicialmente, na Figura 3.21, representa-se por meio de um diagrama em árvore a discretização obtida na codificação binária. Nesta figura, utiliza-se, sem perda de generalidade, um espaço de busca com uma única variável, codificada em quatro bits.

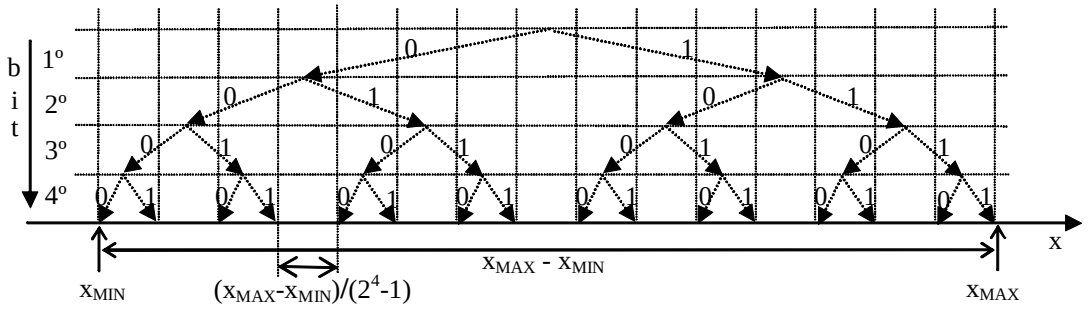


Figura 3.21 – Discretização do espaço de busca (variável x) de uma codificação em 4 bits.

Imaginando, também sem perda de generalidade, que “0101” seja o valor atual desta variável após um número qualquer de iterações do algoritmo GEO ou GEO_{var}, o ponto correspondente “mapeado” no espaço de busca é dado nas Figura 3.22 (setas em negrito) e 3.23.

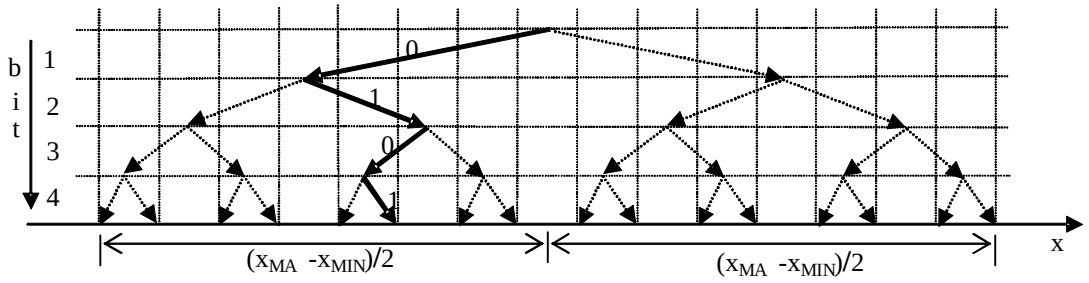


Figura 3.22 – “Mapeamento” (representação gráfica) do ponto “0101”.

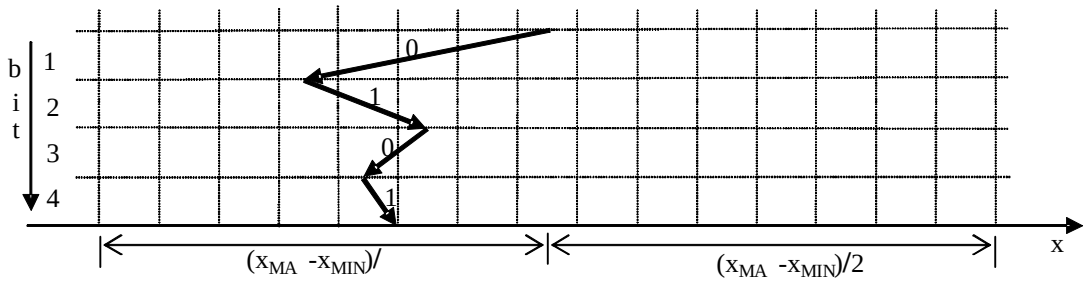


Figura 3.23 – “Mapeamento” (representação gráfica) do ponto “0101”.

Para calcular os seus respectivos índices de adaptação, os quatro bits de “0101” são comutados um a um, gerando a seqüência “1101”, ”0001”, ”0111”, ”0100”. É fácil perceber, com o auxílio da Figura 3.24, que desses quatro pontos apenas “1101” recai sobre a segunda metade do espaço de busca, enquanto os outros três recaem na primeira metade. A primeira metade é justamente aquela que contém o ponto atual “0101”. Para facilitar as análises subseqüentes, chamar-se-á de metade A a que contém o ponto atual e B a outra metade. O fato da metade A ser mais mapeada do que a metade B não é uma coincidência. Ou seja, em uma busca unidimensional, a cada iteração do GEO ou GEO_{var}, haverá sempre um único ponto “mapeando” a metade B do espaço de busca e L-1 pontos mapeando a metade A, onde L é o número de bits utilizado na codificação da variável.

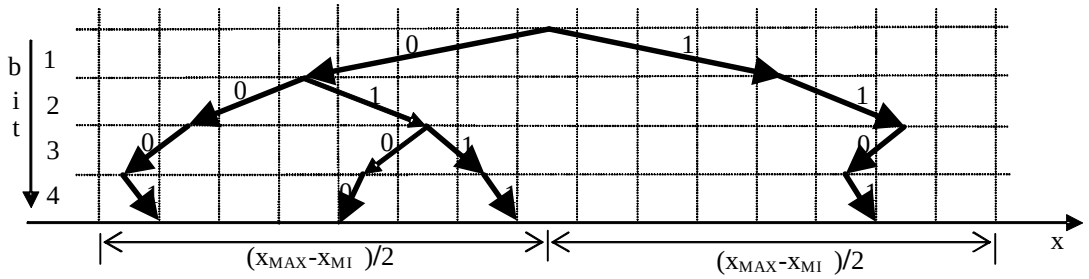


Figura 3.24 – Pontos “1101”, “0001”, “0111”, “0100”.

Esta disparidade no número de pontos mapeados em cada uma das metades caracteriza a assim chamada assimetria na cobertura do espaço de busca, que ocorre a cada iteração do algoritmo GEO ou GEO_{var}. É importante ressaltar que essa assimetria é estática e não dinâmica, ou seja, ela de fato ocorre a cada iteração, mas não significa que ao longo de uma busca, uma metade do espaço é mapeado por uma razão 1/L enquanto a outra é

mapeada por uma razão (L-1)/L do número total de avaliações da função objetivo. Do ponto de vista dinâmico, ou seja, ao longo de várias iterações, a cobertura depende de τ e dos valores da função objetivo a cada iteração, sendo, em geral, impossível de se prever. A razão disto é que o bit mais significativo (1º bit) é o comutador dessa assimetria. Seu valor define em qual metade do espaço de busca o ponto atual se encontra e, portanto, qual metade é mapeada à razão (L-1)/L. Como GEO e GEO_{var} são algoritmos estocásticos e cuja estocasticidade é ajustada pelo parâmetro τ, então o número de comutações do 1º bit em uma busca é estocástico. Por conseqüência, a cobertura do espaço de busca também é estocástica. A dúvida que permanece, a qual esta SA3 tenta responder, é se uma diminuição da assimetria estática tem influência positiva no desempenho dos algoritmos GEO e GEO_{var}.

A exposição do parágrafo anterior sobre a assimetria estática a cada iteração do GEO ou GEO_{var}, causada pelo 1º bit, também é válida recursivamente para os demais bits. Apenas para fins explanativos, imagine-se o 1º bit com valor fixo ao longo de uma busca. Com isto, apenas a metade A do espaço de busca está acessível. Além disso, como o 1º bit está fixo, apenas os outros três bits são comutados. Agora, o bit mais significativo passa a ser o 2º bit. Ele divide a metade A ao meio, da mesma forma que o 1º bit o faz para o espaço de busca total. Raciocínio semelhante pode ser aplicado aos demais bits. Então, pode-se dizer que o 1º bit define uma assimetria de metades, o 2º bit define uma assimetria de quartas partes, o 3º bit uma assimetria de oitavas partes e assim sucessivamente.

Os parágrafos precedentes trataram do caso unidimensional, ou seja, um espaço de busca com apenas uma variável. O caso N-dimensional repete N vezes a situação descrita para o caso unidimensional, ou seja, o domínio de cada variável de projeto é assimetricamente coberto ou mapeado, a cada iteração, de acordo com uma razão 1/L para uma metade do espaço de busca e uma razão (L-1)/L para a outra metade do espaço de busca, isto em termos do número total de avaliações da função objetivo.

A fim de amenizar a assimetria na cobertura do espaço de busca, que ocorre a cada iteração do GEO ou GEO_{var}, a SA3 incorpora nos mesmos o multiordenamento. Os

próximos parágrafos explicam o multiordenamento para o GEO especificamente. Em seguida, o mesmo é feito para o algoritmo GEO_{var}.

Em sua versão mais simples, o multiordenamento de 1 bit, o bit mais significativo de uma das variáveis de projeto é "congelado", isto é, é deixado de fora do processo de comutação dos bits e o ordenamento é feito considerando apenas os outros bits. A contra-partida é que o ordenamento ocorre duas vezes, uma com o bit congelado valendo "0" e outro com o bit congelado valendo "1". A definição de qual bit sofrerá mutação ocorre em duas etapas. Primeiro, é escolhido o ordenamento que ocasionou o maior ganho no valor da função objetivo. Segundo, um bit é escolhido para mutar da forma tradicional com o GEO: de maneira estocástica utilizando-se τ e o ordenamento selecionado na etapa anterior. No multiordenamento de 2 bits, o bit mais significativo de duas das variáveis de projeto é congelado. Agora, o ordenamento ocorre quatro vezes, contemplando todas as combinações de valores possíveis com dois bits, ou seja, $2^2=4$. No multiordenamento de 3 bits, o bit mais significativo de três das variáveis de projeto é congelado e o ordenamento ocorre $2^3=8$ vezes, contemplando todas as combinações de valores possíveis com três bits. Esse procedimento é generalizado conforme segue.

Supondo um multiordenamento com nbc bits congelados e uma codificação das N variáveis de projeto com L bits, podemos ter as seguintes situações:

a) Se nbc<N, então nbc dentre as N variáveis de projeto são escolhidas aleatoriamente com distribuição uniforme para terem seus bits mais significativos congelados ou;

b) Se nbc≥N então calcular os valores de INT(nbc/N) e MOD(nbc/N), onde INT(nbc/N) é a parte inteira e MOD(nbc/N) é o resto da razão nbc/N. Todos os bits com significância 1 até significância INT(nbc/N) em cada variável de projeto são escolhidos para congelamento. Adicionalmente, MOD(nbc/N) variáveis de projeto são escolhidas aleatoriamente para ter seu bit de significância INT(nbc/N)+1 congelado.

Após a definição dos nbc bits a congelar, feito conforme exposto acima, ocorre o multiordenamento, que consiste em efetuar 2^{nbc} ordenamentos, sendo um ordenamento para cada combinação diferente de valores dos bits congelados. Em seguida, dentre os 2^{nbc}

ordenamentos, aquele com o maior ganho no valor da função objetivo é escolhido e utilizado para realizar a escolha do bit a ser mutado pelo GEO na iteração atual.

Apresentam-se, a seguir, os passos do algoritmo GEO para esta SA3. Esta variante é denominada GEO_3. Conforme mencionado na seção 3.7.1, GEO_3 foi desenvolvida tomando GEO_1 como base. As alterações estão realçadas em negrito.

O algoritmo GEO_3 é composto dos seguintes passos:

1. Inicialize aleatoriamente uma seqüência binária **C**, de comprimento L que codifica N variáveis de projeto. Dentro de **C**, cada variável de projeto X_j é codificada em uma parcela com comprimento l_j, j$\in$ {1,2,...,N} e tal que $\sum_j l_j$ = L. **Defina o valor de nbc, onde nbc é o número de bits congelados a cada iteração.** Converta **C** em **X** e calcule o valor da função objetivo F(**X**) e faça $\mathbf{X_{melhor}}$ = **X** e F($\mathbf{X_{melhor}}$) = F(**X**).
2. **Crie 2^{nbc} vetores binários, cada um com nbc elementos, de modo a formar todas as 2^{nbc} combinações possíveis de um vetor binário com nbc elementos. Armazene estes vetores numa matriz de elementos $b_{i,q}$, onde i Î {1,2,..., nbc}, q Î {1,2,...,2^{nbc}}. Ou seja, os índices i, q indicam elemento (bit congelado) i da combinação (vetor) q. Para nbc=2, por exemplo, $b_{i,q}$ Î { {0,0},{0,1},{1,0},{1,1} }.**
3. **Crie uma lista ibc_{indx} Î {1,2,...,L}, onde indx Î {1,2,...,nbc}, que indica quais bits devem ser congelados em C.**
    a. **Faça indx= 0. Calcule bcv= INT(nbc/N), onde INT(nbc/N) retorna a parte inteira da razão nbc/N. Se bcv ³ 1, então sinalize que os bcv bits de maior significância de cada variável j devem ser congelados. Ou seja, para i Î {1,2,...,bcv} faça:**
        1º. **Para j Î {1,2,...,N} faça:**
            I. **indx= (i-1)*N + j.**
            II. $\mathbf{ibc_{indx}} = \sum_{n=1}^{j} l_n - \mathbf{i} + \mathbf{1}.$

b. Calcule rbc= MOD(nbc/N), onde MOD(nbc/N) retorna o resto da razão nbc/N. Se rbc ³ 1, então sorteie uniformemente os rbc bits restantes entre as N variáveis de projeto, de modo que cada variável seja sorteada no máximo uma vez. Para cada variável j sorteada corretamente (não repetida), faça indx= indx + 1 e faça $ibc_{indx} = \sum_{n=1}^{j} l_n$ - bcv-1.

4. Crie 2^{nbc} configurações binárias G_q, $q \in \{1,2,...,2^{nbc}\}$, cada uma com L bits, ou seja, o elemento $g_{q,ib} \in G_q$, $ib \in \{1,2,...,L\}$, $g_{q,ib} \in \{0,1\}$ e onde $L= \sum_{j \in \{1,2,...,N\}} l_j$ é o total de bits usado na representação das variáveis de projeto. Em seguida, para $q \in \{1,2,...,2^{nbc}\}$, faça:

a. G_q= C.

b. Para $i \in \{1,2,..., nbc\}$, faça:

1º. $ib= ibc_i$.

2º. $g_{q,ib}= b_{i,q}$.

5. Para $q \in \{1,2,...,2^{nbc}\}$, faça:

a. C= G_q.

b. Faça $F(\mathbf{X_{melhor}})_{anterior} = F(\mathbf{X_{melhor}})$ **e faça $q_{melhor}= 0$. Sorteie um valor para q_{ALE}, tal que $q_{ALE} \in \{1,2,...,2^{nbc}\}$. Para cada bit $i \in \{1,2,...,L\}$ e tal que $i \notin$ ibc, ou seja, para todos os bits menos os bits congelados da seqüência C, faça:**

1º. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule $F(\mathbf{X_i})$. Em seguida, se $F(\mathbf{X_i}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X_i})$ e $\mathbf{X_{melhor}} = \mathbf{X_i}$.

2º. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - F(\mathbf{X_{melhor}})_{anterior}$, que indica o ganho (ou perda) que se têm ao mudar

o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.

3º. Retorne o bit i ao seu valor original.

c. Ordene todos os bits $i \in \{1,2,...,L\}$ **e tal que** $i \in$ **ibc**, de acordo com os seus índices de adaptação, de k=1 para o menos adaptado à k=L-**nbc**, para o mais adaptado. Crie uma lista de valores (i,k) relacionando a posição física i do bit em **C** com seu respectivo número de ordem k, onde $i \in \{1,2,...,L\}$ **tal que** $i \in$ **ibc** e $k \in \{1,2,...,L\text{-}\mathbf{nbc}\}$. Se $\Delta F(\mathbf{X_i}) = \Delta F(\mathbf{X_j})$, i?j, estabeleça a ordem entre i e j de forma aleatória.

d. Se $F(\mathbf{X_{melhor}}) < F(\mathbf{X_{melhor}})_{anterior}$ então armazene o valor de q e a respectiva lista (i,k): faça $q_{melhor} = q$ e faça $(i,k)_{melhor} = (i,k)$.

e. Se $q = q_{ALE}$, então faça $(i,k)_{ALE} = (i,k)$.

6. Se q_{melhor} ? 0, então faça $\mathbf{C} = \mathbf{G}_{q_{melhor}}$ e $(i,k) = (i,k)_{melhor}$, senão faça $\mathbf{C} = \mathbf{G}_{q_{ALE}}$ e faça $(i,k) = (i,k)_{ALE}$.

7. Escolha com igual probabilidade **entre os bits não congelados** um bit candidato i para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual à $P_i(k)$, confirme o bit i para ser modificado. Repetir este passo até que um bit seja confirmado. Quando isso acontecer, faça $i_{esc} = i$.

8. Faça $\mathbf{C} = \mathbf{C_{iesc}}$, ou seja, em **C**, mute o bit i_{esc}. Converta **C** em **X** e calcule F(**X**).

9. Repita os passos 2 a 8 até que um dado critério de parada seja satisfeito.

10. Retorne $\mathbf{X_{melhor}}$ e $F(\mathbf{X_{melhor}})$.

Para o algoritmo GEO_{var} a SA3 implementa o multiordenamento da seguinte forma: Primeiro, nbc bits são escolhidos para serem congelados. A escolha é feita do modo idêntico ao do GEO, já exposto. Em seguida, ocorre o multiordenamento, que é feito separadamente para cada variável de projeto. Assim, a variável de projeto j efetua 2^{nbcv_j} ordenamentos, onde $nbcv_j$ é o número de bits congelados na variável j. Na seqüência, dentre os 2^{nbcv_j} ordenamentos da variável j, aquele com o maior ganho no valor da função

objetivo é escolhido e utilizado para realizar a escolha do bit da variável j a ser mutado pelo GEO_{var} na iteração atual. O total de ordenamentos a cada iteração é dado por $\sum_{j=1}^{N} 2^{nbcv_j}$ e a soma dos bits congelados é igual a nbc, ou seja, $\sum_{j=1}^{N} nbcv_j = nbc$.

Apresentam-se a seguir, os passos do algoritmo GEO_{var} para esta SA3. Esta variante é denominada GEO_{var3}. Conforme mencionado na seção 3.7.1, GEO_{var3} foi desenvolvida tomando GEO_{var1} como base. As alterações estão realçadas em negrito. O algoritmo GEO_{var3} é composto dos seguintes passos:

1. Inicialize aleatoriamente N seqüências binárias C_j, $j \in \{1,2,...,N\}$ que codificam as N variáveis de projeto X_j. Cada C_j tem l_j bits, ou seja, o elemento $c_{j,i} \subset C_j$, $i \in \{1,2,..., l_j\}$ e $c_{j,i} \in \{0,1\}$. Assim, o vetor **X** contendo as variáveis de projeto é codificado em uma matriz **C** cuja linha j é igual a seqüência binária C_j utilizada na codificação da variável X_j. **Além disso, o total de bits em C é dado por $L=\sum_j l_j$ e representa o total de bits usado na codificação das variáveis de projeto.** Converta **C** em **X** e calcule o valor da função objetivo F(**X**) e faça $\mathbf{X_{melhor}}$ = **X** e F($\mathbf{X_{melhor}}$) = F(**X**). **Defina o valor de nbc, onde nbc é o número de bits congelados a cada iteração. Calcule INT(nbc/N) e MOD(nbc/N), ou seja, a parte inteira e o resto da razão nbc/N, respectivamente.**
2. **Crie uma lista $nbcv_j$, $j \in \{1,2,...,N\}$, que indica o número de bits mais significativos congelados em cada variável de projeto j. Faça $nbcv_j$ = INT(nbc/N) , $j \in \{1,2,...,N\}$, ou seja, sinalize que os INT(nbc/N) bits de maior significância de cada variável j devem ser congelados. Em seguida, sorteie uniformemente os MOD(nbc/N) bits restantes entre as N variáveis de projeto, de modo que cada variável seja sorteada no máximo uma vez. Para as variáveis de projeto sorteadas, faça $nbcv_j = nbcv_j + 1$.**
3. **Para cada variável $j \in \{1,2,...,N\}$, faça:**
    a. **Crie 2^{nbcv_j} vetores binários, cada um com $nbcv_j$ elementos, de modo a formar todas as 2^{nbcv_j} combinações possíveis de um vetor binário com**

$nbcv_j$ elementos. Armazene estes vetores numa matriz de elementos $b_{i,q,j}$, onde $i \in \{1,2,..., nbcv_j\}$, $q \in \{1,2,...,2^{nbcv_j}\}$. Ou seja, os índices i, q, j indicam elemento (bit) i do vetor q da variável j.

b. Crie 2^{nbcvj} configurações binárias $G_{j,q}$, $q \in \{1,2,...,2^{nbcv_j}\}$, cada uma com L bits, ou seja, o elemento $g_{j,q,ib} \in G_{j,q}$, $ib \in \{1,2,...,L\}$, $g_{j,q,ib} \in \{0,1\}$ e onde $L = \sum_j l_j$ é o total de bits usado na representação das variáveis de projeto. Em seguida, para $q \in \{1,2,...,2^{nbcv_j}\}$, faça:

1º. $G_{j,q} = C$.

2º. Para $i \in \{1,2,..., nbcv_j\}$, faça:

I. $ibc = l_j + 1 - i$.

II. $g_{j,q,ibc} = b_{i,q,j}$.

4. Para cada variável $j \in \{1,2,...,N\}$, faça:

a. Para $q \in \{1,2,...,2^{nbcv_j}\}$, faça:

1º. $C = G_{j,q}$.

2º. Faça $F(\mathbf{X_{melhor}})_{anterior} = F(\mathbf{X_{melhor}})$ **e faça $q_{melhor_j} = 0$. Sorteie um valor para q_{ALE_j}, tal que $q_{ALE_j} \in \{1,2,...,2^{nbcv_j}\}$.** Para cada bit $i \in \{1,2,..., l_j - nbcv_j\}$ da seqüência C_j, faça:

I. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule $F(\mathbf{X_i})$. Em seguida, se $F(\mathbf{X_i}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X_i})$ e $\mathbf{X_{melhor}} = \mathbf{X_i}$.

II. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - F(\mathbf{X_{melhor}})_{anterior}$, que indica o ganho (ou perda) que se têm ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.

III. Retorne o bit i ao seu valor original.

3º. Ordene os bits i∈{1,2,..., l_j–**nbcv$_j$**}, ou seja, os bits **não congelados** da variável j, de acordo com os seus índices de adaptação $\Delta F(\mathbf{X_i})$, de k=1 para o menor $\Delta F(\mathbf{X_i})$ até k= l_j–**nbcv$_j$**, para o maior $\Delta F(\mathbf{X_i})$.

4º. Crie uma lista (i,k), relacionando a posição física do bit i em C_j com seu número de ordem k. Se $\Delta F(\mathbf{X_i})_m = \Delta F(\mathbf{X_i})_n$, ∀m?n, estabeleça a ordem entre m e n de forma aleatória.

5º. Se $F(\mathbf{X}_{melhor}) < F(\mathbf{X}_{melhor})_{anterior}$ então armazene o valor de q e a respectiva lista (i,k): faça $q_{melhor_j} = q$ e faça $(i,k)_{melhor_j} = (i,k)$.

6º. Se $q = q_{ALE_j}$, então faça $(i,k)_{ALE_j} = (i,k)$.

5. **Para cada variável j Î {1,2,...,N}, faça:**

   a. **Se q_{melhor_j} ? 0, então faça C= $G_{j,q_{melhor_j}}$ e (i,k)= $(i,k)_{melhor_j}$, senão faça C= $G_{q_{ALE_j}}$ e faça (i,k)= $(i,k)_{ALE_j}$.**

   b. Escolha com igual probabilidade um bit candidato i∈{1,2,..., l_j-**nbcv$_j$**} para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual a $P_i(k)$, confirme o bit para mutar: faça i_{esc}= i. Repetir este passo até que um bit seja confirmado para ser modificado.

   c. Em seguida, na linha j da matriz C, ou seja, em C_j, mute o bit na posição i_{esc}.

6. Converta **C** em **X** e calcule F(**X**). Se F(**X**) < $F(\mathbf{X}_{melhor})$ então faça $F(\mathbf{X}_{melhor})$ = F(**X**) e $\mathbf{X}_{melhor}$ = **X**.

7. Repita os passos 2 a **6** até que um dado critério de parada seja satisfeito.

8. Retorne $\mathbf{X}_{melhor}$ e $F(\mathbf{X}_{melhor})$.

As versões GEO_3 e GEO_{var3} foram aplicadas ao conjunto de funções teste FT_i, i∈{1,2,...,5} descritas na seção 3.7.1.

Inicialmente, é feita uma busca do tipo varredura a fim de verificar a influência dos parâmetros τ e nbc nas versões SA3 de GEO e GEO_{var}. Para τ, a varredura deu-se no intervalo de 0 a 10 com passo 0,25. Para nbc, utilizou-se nbc$\in \{1;2;3\}$.

As Figuras 3.25 até 3.29 apresentam os resultados médios de 50 execuções independentes, em termos do melhor F(**X**) encontrado ao final de cada execução, obtidos com as versões GEO_3 e GEO_{var3}. O número de avaliações de F(**X**) a cada execução é 10^4 para FT_1 e 10^5 para as demais funções teste. Nas figuras, cada curva apresenta o resultado para um dado valor de nbc, conforme indicado entre parênteses, e para τ de 0 a 10.

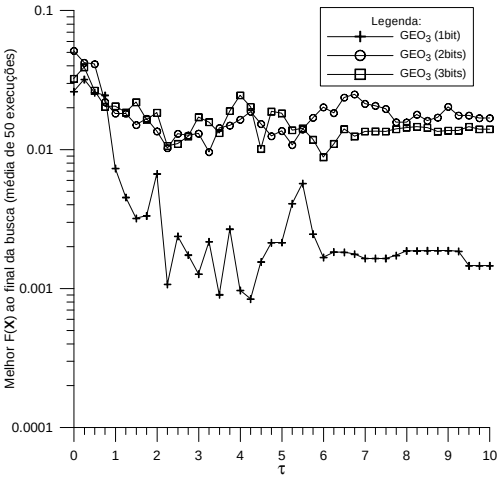


Figura 3.25a - FT_1 x τ x (nbc) com GEO_3

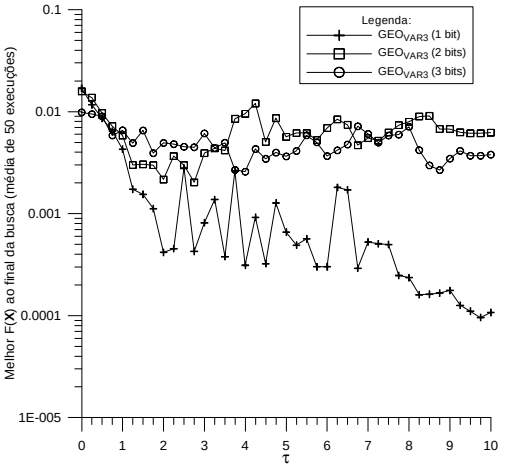


Figura 3.25b - FT_1 x τ x (nbc) com GEO_{var3}

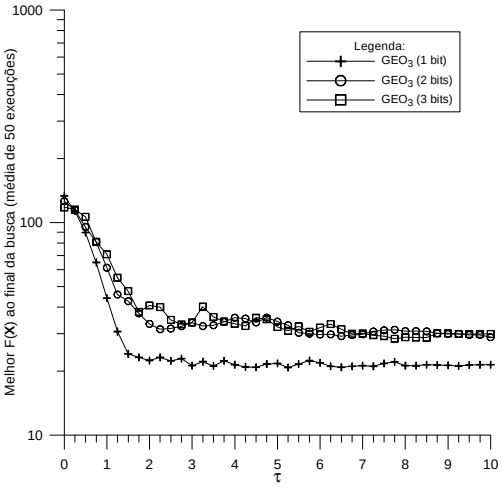


Figura 3.26a - FT_2 x τ x (nbc) para GEO_3

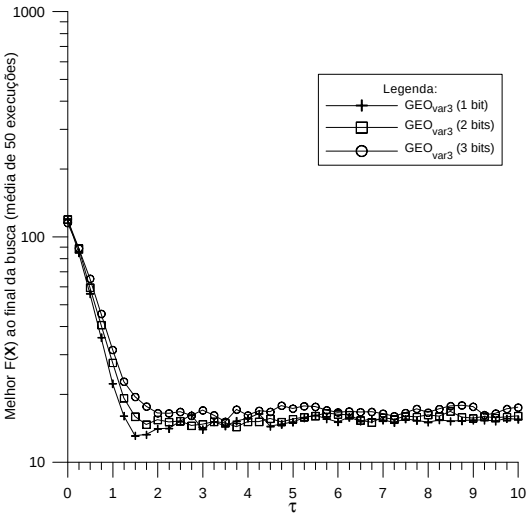


Figura 3.26b - FT_2 x τ x (nbc) para GEO_{var3}

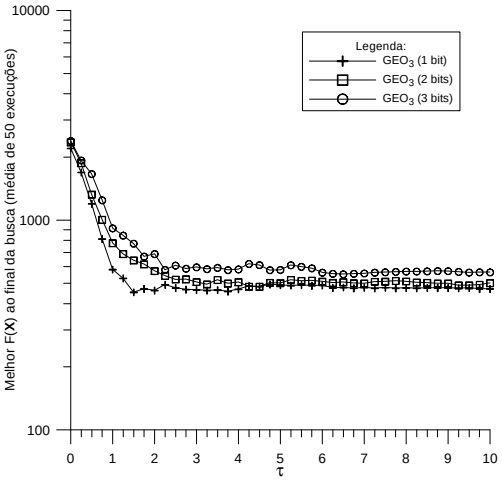


Figura 3.27a - FT_3 x τ x (nbc) para GEO_3

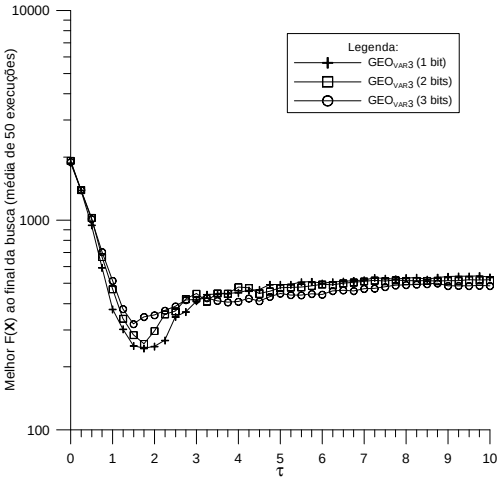


Figura 3.27b - FT_3 x τ x (nbc) para GEO_{var3}

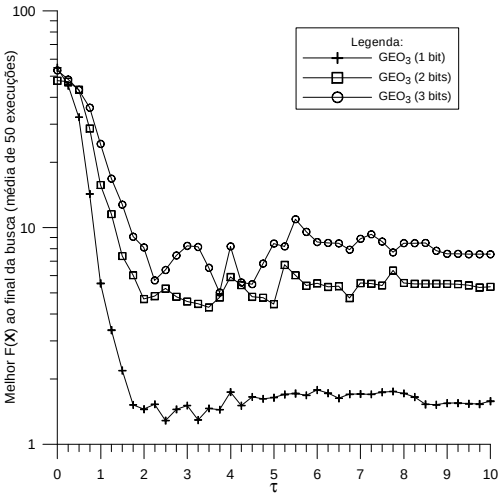


Figura 3.28a - FT_4 x τ x (nbc) para GEO_3

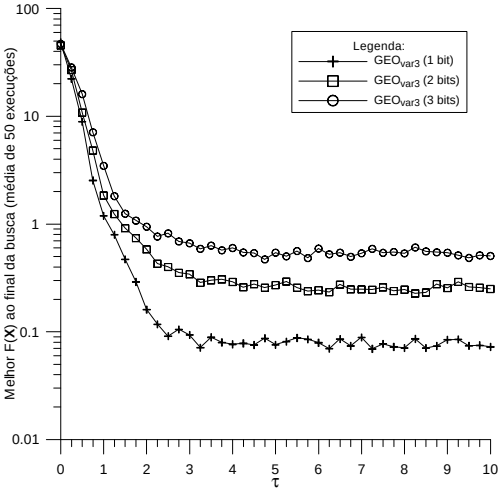


Figura 3.28b - FT_4 x τ x (nbc) para GEO_{var3}

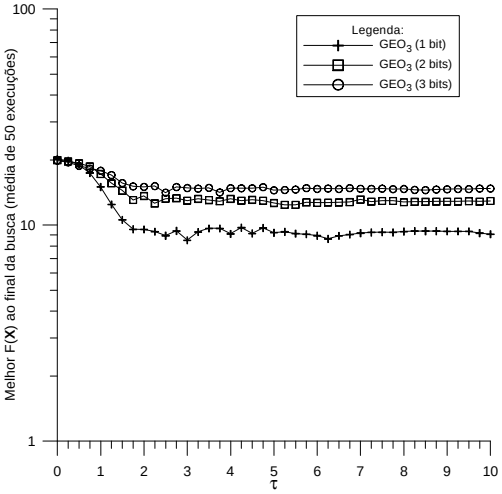


Figura 3.29a - FT_5 x τ x (nbc) para GEO_3

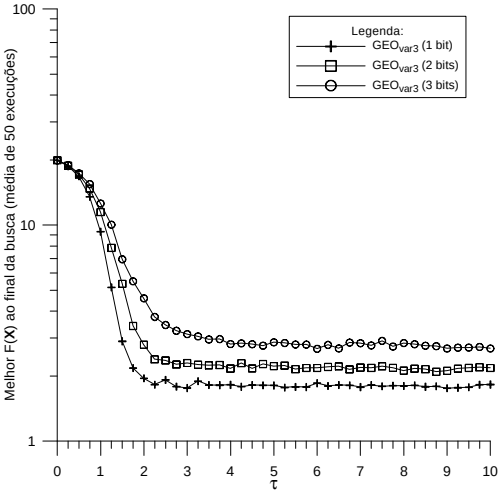


Figura 3.29b - FT_5 x τ x (nbc) para GEO_{var3}

Analisando-se os resultados apresentados nas Figuras 3.25 a 3.29, é possível perceber um padrão comum, tanto com GEO_3 como com GEO_{var3}, onde nbc=1 obtém o melhor desempenho para todas as funções teste.

Os resultados obtidos por GEO_3 e GEO_{var3} com nbc=1 são comparados, com o auxílio das Figuras 3.30 a 3.34, com os melhores resultados obtidos pelas versões SA1 e SA2.

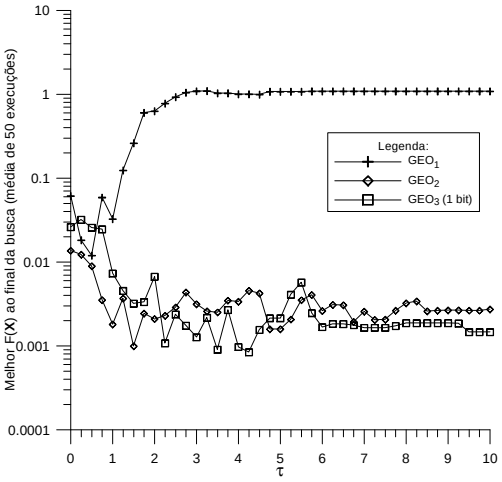


Figura 3.30a - FT_1 x τ x SA para GEO

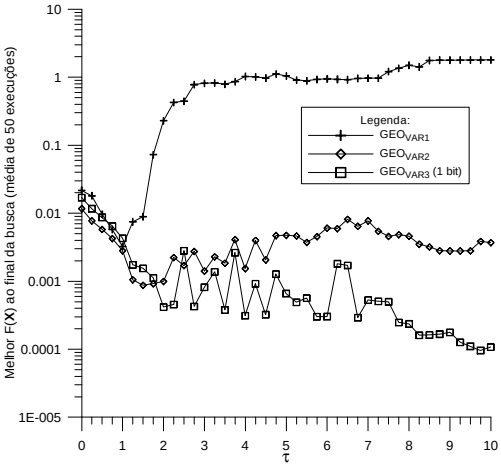


Figura 3.30b - FT_1 x τ x SA para GEO_{var}

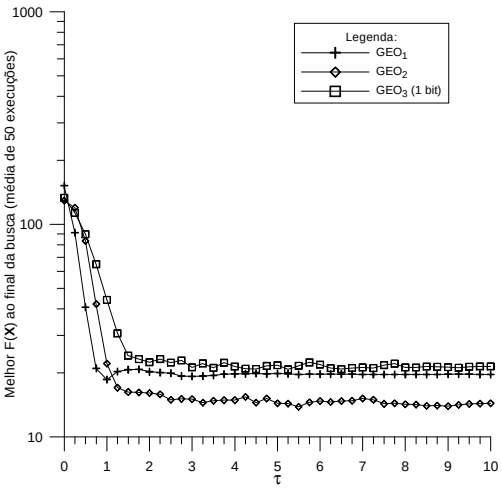


Figura 3.31a - FT_2 x τ x SA para GEO

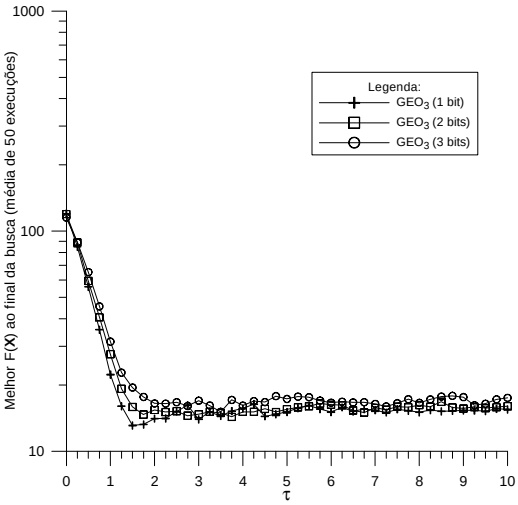


Figura 3.31b - FT_2 x τ x SA para GEO_{var}

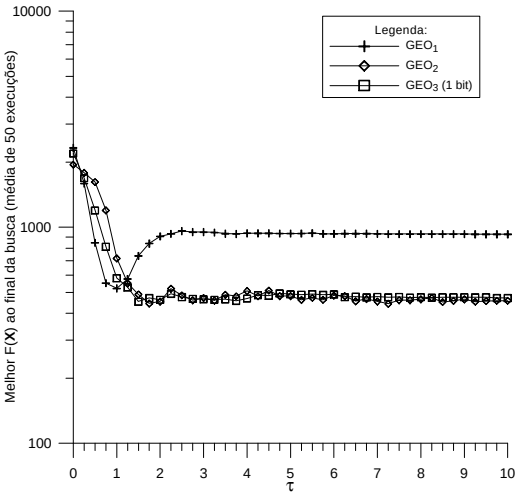


Figura 3.32a - FT_3 x τ x SA para GEO

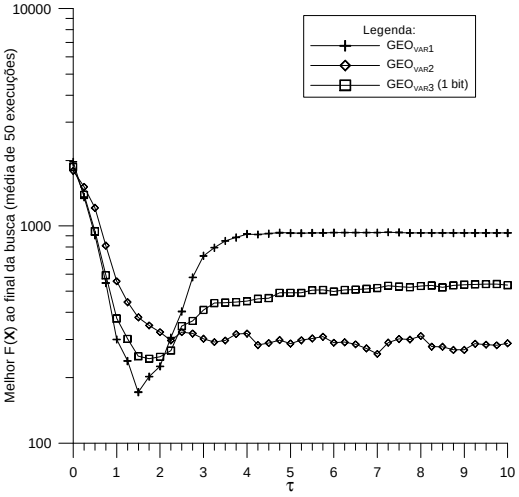


Figura 3.32b - FT_3 x τ x SA para GEO_{var}

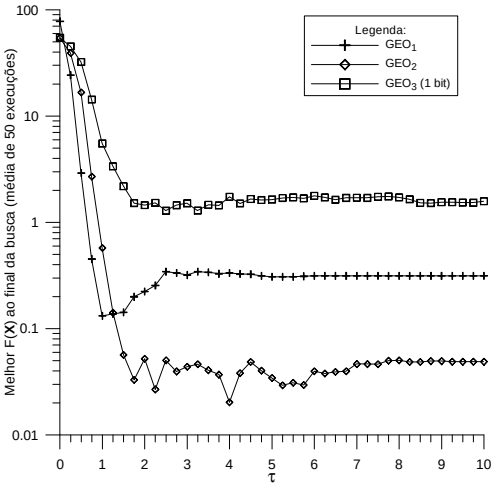


Figura 3.33a - FT_4 x τ x SA para GEO

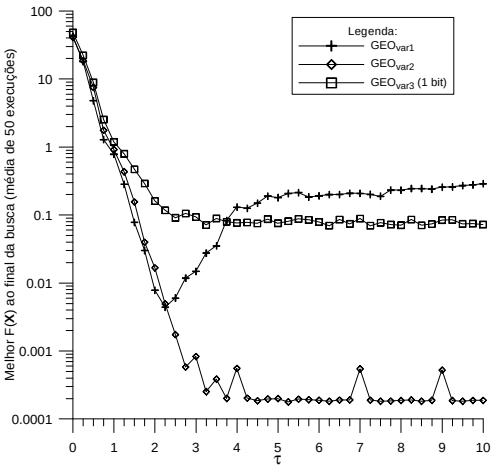


Figura 3.33b - FT_4 x τ x SA para GEO_{var}

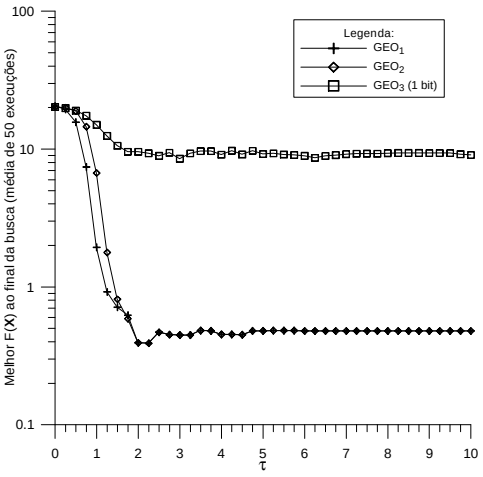


Figura 3.34a - FT_5 x τ x SA para GEO

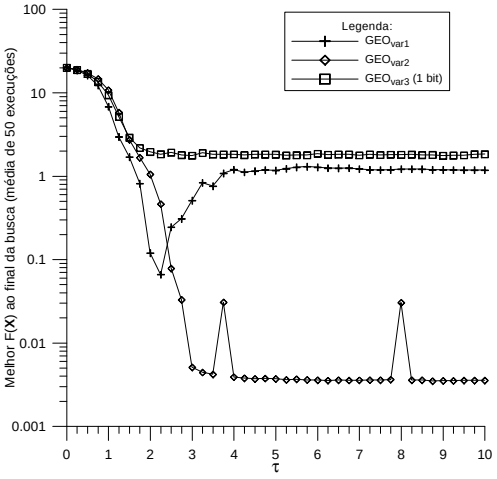


Figura 3.34b - FT_5 x τ x SA para GEO_{var}

É possível perceber, com o auxílio das Figuras 3.30 a 3.34, que a SA3 obteve vantagem relativamente as SA's 1 e 2 apenas para a função teste FT_1 (Figuras 3.30a e

3.30b), onde a SA3 de ambos GEO e GEO_{var} foi superior às demais SA's. Para as demais funções teste, a SA3 foi claramente inferior para FT_2, FT_4, FT_5 e equivalente à SA2 para a função teste FT_2 com o GEO apenas.

Tendo em vista que a função teste FT_1 é uma função de duas variáveis apenas e unimodal, então a vantagem obtida pela SA3 para esta função não é significativa. Por outro lado, a desvantagem da SA3 nas demais funções teste, estas sim multimodais e com dimensionalidades elevadas comparativamente à FT_1, é bastante significativa e aponta na direção de que a assimetria do GEO na cobertura do espaço de busca não é um fator que gera perda de desempenho, pelo menos não para funções com dimensionalidade elevada e multimodais.

3.7.4 – Sugestão de Aprimoramento 4 (SA4)

O quarto aprimoramento sugerido tem em vista os algoritmos GEO e sua variante GEO_{var} e introduz nestes a representação interna das variáveis de projeto no domínio real, bem como uma estruturação alternativa para as mutações. Como visto no início deste Capítulo, GEO e GEO_{var} utilizam internamente codificação binária na representação das variáveis de projeto. Entretanto, existem estudos que apontam que a codificação binária nem sempre é a forma de representação mais indicada (Eiben e Smith, 2003; Hollstein, 1971). Para os problemas em que o domínio e a forma original de representação das variáveis de projeto são, respectivamente, o domínio real e o sistema numérico decimal, a manutenção da representação original parece ser um caminho natural. Este é o caso de grande parte dos problemas de otimização nas áreas científica e de engenharia. Entretanto, para que seja possível manter a forma de representação original, é necessário que o algoritmo de otimização a ser utilizado permita tal escolha.

Talvez a principal razão para a utilização da codificação binária por parte de GEO e GEO_{var} seja a analogia entre bits e genes, que tradicionalmente é feita no âmbito da otimização com algoritmos evolutivos (Eiben e Smith, 2003). Nessa analogia, as possíveis soluções do problema original (muitas vezes números reais e inteiros decimais) são representadas como seqüências binárias equivalentes. Na Biologia, seqüências de genes formam cromossomos e estes, por sua vez, são determinísticos para as características do

indivíduo que representam. As características do indivíduo, por sua vez, são determinísticas da capacidade do mesmo de interagir com o meio que o cerca, adaptar-se a ele, sobreviver e gerar descendentes. Da mesma forma, na otimização evolutiva, bits compõem seqüências binárias e estas são determinísticas das características da solução que representam. De certa forma, a computação evolutiva parece ter uma vantagem *a priori* sobre a Biologia: Na Biologia, nem sempre é fácil ou mesmo possível identificar quais as características (isto para não falar dos genes) que, ao serem alteradas, tornam um indivíduo menos ou mais adaptado. Na computação evolutiva, por outro lado, uma vez formulado o problema de otimização, as características que tornam uma solução pior, ou melhor, estão sempre identificadas: correspondem às variáveis do problema formulado. Estas variáveis definem, por meio dos valores particulares que assumem, se uma dada solução é pior ou melhor.

No GEO e GEO_{var}, durante o processo de ordenamento dos bits, cada um deles é mutado e a solução resultante é avaliada, sendo-lhe atribuída um índice de adaptação. Posteriormente, após a mutação e avaliação de todos os bits, esta informação é usada pelo algoritmo para, estocasticamente com o auxílio de τ, decidir o próximo movimento do algoritmo no espaço de busca, ou seja, decidir qual dos bits deverá, de fato, sofrer mutação.

A exposição do parágrafo anterior é feita apenas para lembrar que a representação binária é parte central do funcionamento do GEO e GEO_{var}. Partindo desse pressuposto, a primeira etapa na consecução da SA4 foi passar a representação interna que a SA1 de GEO e GEO_{var} fazem das variáveis de projeto de binária para decimal, mas mantendo o funcionamento do algoritmo como se a representação ainda fosse binária. Esta primeira etapa não representa a versão final da SA4, mas sim uma versão preliminar. Esta versão preliminar, no entanto, possui a importante característica de permitir a validação de seus resultados, mediante comparação com os resultados já obtidos pela sua versão binária correspondente (SA1).

Para que a SA4 mantenha os efeitos da codificação binária, é necessário que os efeitos das mutações dos bits das variáveis de projeto do GEO e GEO_{var} SA1 sejam simulados nas variáveis de projeto decimais da SA4. Para isto, é necessário entender o que ocorre cada vez que um bit usado na representação de uma variável de projeto é mutado.

Supondo uma variável de projeto j, codificada em l_j bits, com restrições laterais dadas por X_{min_j} e X_{max_j}, então o tamanho, a magnitude da menor perturbação possível é dada pela resolução ou precisão obtida quando da codificação binária. Rearranjando a equação 3.3, tem-se que tal resolução é dada por:

$$e_j = \frac{(X_{max_j} - X_{min_j})}{\left(2^{l_j} - 1\right)} \tag{3.4}$$

A magnitude e_j obtida pela equação 3.4 ocorre sempre que a variável de projeto j tiver seu bit menos significativo mutado. Para calcular as magnitudes das perturbações devidas aos demais bits, é preciso estabelecer uma relação entre as casas binárias. Em qualquer sistema de representação numérica, a ordem de grandeza entre casas numéricas adjacentes é dada pela base do referido sistema. Assim, a mutação do segundo bit menos significativo da variável j tem magnitude igual a $2e_j$, o terceiro bit menos significativo $4e_j$, e assim por diante. Colocando na forma de equação, tem-se que a magnitude de uma perturbação induzida na variável j pela mutação de seu bit de significância i é dada por:

$$m_{i,j} = 2^{(i-1)} e_j, \quad i \in \{1,2,...,l_j\}, \; j \in \{1,2,...,N\} \tag{3.5}$$

Na equação 3.5, i indica a significância do bit, com i=1 para o bit menos significativo e i= l_j para o bit mais significativo. A equação 3.5 permite calcular a magnitude de uma perturbação induzida por qualquer um dos l_j bits da variável j. No entanto, ela não permite determinar o sinal de tal perturbação. Em outras palavras, a equação 3.5 não especifica se a perturbação deve ser adicionada ou subtraída ao valor decimal atual da variável j. Esta informação é obtida com o auxílio da figura a seguir.

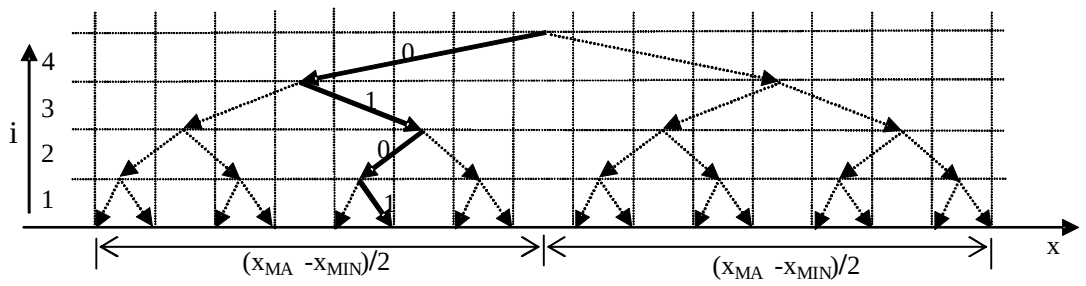


Figura 3.35 – Representação gráfica do ponto “0101”.

Na Figura 3.35, uma variável de projeto x qualquer é codificada em quatro bits. O valor atual é representado graficamente pelas setas em negrito e as demais setas apresentam todas as combinações possíveis com uma codificação de quatro bits. Os pontos onde as setas tocam o eixo da variável (eixo x) representam os pontos do domínio da variável x (aqui suposta real) para os quais existe correspondente na codificação binária. Imaginando-se agora que a SA4, que só utiliza codificação real, inicialize aleatoriamente a variável x com um valor idêntico ao do ponto "0101" representado na figura. Olhando-se a figura, é fácil perceber que, nesse caso, o bit de significância i=1 impõe à variável x uma perturbação negativa, visto que, conforme indica a figura, ao ser mutado tal bit leva a um ponto situado à esquerda do ponto atual. Do mesmo modo, para i=2, vê-se que a perturbação é positiva, para i=3 é negativa e para i=4 é novamente positiva. Claro está que o valor do bit determina o sinal da perturbação: bits com valor 0 impõem perturbações positivas e bits com valor 1 impõem perturbações negativas. Portanto, para obtenção dos sinais das perturbações a serem inseridas diretamente na variável x, é necessário realizar sua conversão para o sistema binário.

O algoritmo descrito a seguir realiza a conversão de uma variável $X_j \in [X_{min_j}, X_{max_j}]$ qualquer para seu equivalente binário usando l_j bits no intervalo $[X_{min_j}, X_{max_j}]$. No algoritmo, o subscrito j indica variável X_j e o subscrito i indica a significância do bit. Assim, por exemplo, $c_{2,3}$ é o terceiro bit menos significativo da variável X_2.

1. Faça $R=X_j$.
2. Para $i \in \{l_j, l_j\text{-}1, ..., 1\}$ faça:
   a. Calcule $F=\frac{(X_{max_j} - X_{min_j})}{2^{(l_j - i + 1)}}$.
   b. Se R>F, então faça $c_{j,i}=1$ e R= R-F. Senão, faça $c_{j,i}=0$.

Agora, estão disponíveis todas as informações necessárias para simular em uma variável de domínio real X_j as perturbações equivalentes às mutações dos bits da configuração binária C_j. Ao longo do texto que segue, estas perturbações são também chamadas de mutações, em alusão às suas origens.

O GEO canônico usa codificação binária e muta as variáveis de projeto assim codificadas pela comutação dos dígitos binários um a um. Como a representação binária tem base 2, isto implicitamente significa que as magnitudes das mutações são incrementadas por um fator de escala multiplicativo igual a 2. Por exemplo, por meio da equação 3.5, tem-se que $m_{i,j} \in \{e_j, 2e_j, 4e_j, 8e_j, ..., 2^{(l_j-1)}e_j\}$. Visto que o bit mais significativo tem ordem de grandeza igual à metade do espaço de busca, isto implica também que a maior mutação possível tem um tamanho fixo que é igual à metade do espaço de busca de cada variável de projeto. É razoável imaginar que para algumas funções este tamanho não seja o mais indicado, acarretando perda de eficiência. Em sua segunda etapa, a SA4 elimina esta limitação instituindo a base b como um parâmetro do algoritmo, não sendo mais igual a 2 implicitamente.

Apresentam-se, a seguir, os passos da segunda etapa da SA4 para o algoritmo GEO e, logo em seguida, o mesmo é feito para GEO_{var}. O algoritmo da primeira etapa da SA4 é um caso particular do algoritmo desta segunda etapa e pode ser obtido a partir deste último fazendo b=2. Por esta razão e para evitar duplicações desnecessárias, os passos do algoritmo SA4 da primeira etapa são omitidos.

O algoritmo GEO_4 da segunda etapa é composto dos seguintes passos:

1. Inicialize aleatoriamente e com distribuição uniforme entre $\mathbf{X_{min}}$ e $\mathbf{X_{max}}$ um vetor $\mathbf{X}$ contendo as N variáveis de projeto. Calcule o valor da função objetivo F($\mathbf{X}$), faça $\mathbf{X_{melhor}} = \mathbf{X}$ e guarde F($\mathbf{X_{best}}$).

2. Defina o número de mutações l_j, $j \in \{1,2,...,N\}$ de cada variável X_j, tal que $\sum_j l_j = L$, sendo L número total de mutações agindo sobre as N variáveis de projeto. Defina um valor para b, a base para a representação das mutações, tal que b>1.

3. Calcule o vetor $\mathbf{e}$, onde $e_j = (x_{maxj} - x_{minj})/(b^{l_j} - 1)$, e j é o índice da variável, isto é, $j \in \{1,2,...,N\}$. O elemento e_j define a resolução de mutação da j-ésima variável de projeto. Por resolução entenda-se o menor valor a ser somado ou subtraído de X_j.

4. Faça $F(\mathbf{X})_{ref} = F(\mathbf{X})$, $F(\mathbf{X_{melhor}})_{ref} = F(\mathbf{X_{melhor}})$.

5. Faça n=0. Para cada variável $j \in \{1,2,...,N\}$ do vetor $\mathbf{X}$, faça:

a. Converta X_j para sua codificação b-equivalente usando l_j casas e obtenha C_j. Ou seja:

1º. Faça $R=X_j$.

2º. Para $i \in \{l_j, l_j\text{-}1, ..., 1\}$ faça:

I. Calcule $F=\frac{(X_{max_j} - X_{min_j})}{b^{(l_j - i + 1)}}$.

II. Se R>F, então faça $c_{j,i}=1$ e R= R-F. Senão, faça $c_{j,i}=0$.

b. Para cada mutação $i \in \{1,2,..., l_j\}$ da variável X_j, faça:

1º. Incremente o número total de mutações do vetor **X**: faça n= n+1.

2º. Calcule o tamanho da mutação $m= b^{(i-1)} \cdot e_j$.

3º. Calcule o sinal da mutação $s=(-1)^{(c_{j,i})}$, onde $c_{j,i}$ é o valor binário calculado no passo 5.a.

4º. Mute X_j. Primeiro, faça $X_{aux}= X_j$. Em seguida, faça $X_j= X_j+s \cdot m$, gerando um vetor mutado $\mathbf{X_n}$. Verifique os limites: Se $X_j > X_{max_j}$ ou $X_j < X_{min_j}$, então sorteie um novo $X_j \in [X_{min_j}, X_{max_j}]$ com distribuição uniforme e faça $m= abs(X_j\text{-}X_{aux})$ e $s=(X_j\text{-}X_{aux})/m$.

5º. Calcule o valor da função objetivo $F(\mathbf{X_n})$. Atribua à mutação n um valor de adaptação $\Delta F(\mathbf{X_n})= F(\mathbf{X_n}) - F(\mathbf{X_{best}})_{ref}$, que indica o ganho ou a perda que a função objetivo tem se a mutação n ocorrer, quando comparada com o melhor valor da função objetivo encontrado até a iteração anterior. Em seguida, se $F(\mathbf{X_n}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}})= F(\mathbf{X_n})$ e $\mathbf{X_{melhor}} = \mathbf{X}_n$.

6º. Retorne **X** à sua condição não mutada: faça $X_j= X_j\text{-}s \cdot m$.

6. Ordene todas as mutações $n \in \{1,2,...,L\}$ de acordo com os seus índices de adaptação, de k=1 para as menos adaptadas à k=L, para a mais adaptada. Crie uma lista de valores (n,k) relacionando a mutação n com seu respectivo número de ordem k, onde $n,k \in \{1,2,...,L\}$. Se $\Delta F(\mathbf{X_n}) = \Delta F(\mathbf{X_u})$, n?u, estabeleça a ordem entre n e u de forma aleatória.

7. Escolha com igual probabilidade uma mutação candidata n para ser efetivada. Calcule $P_n(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual à $P_n(k)$, confirme a mutação n para ser efetivamente aplicada. Repetir este passo até que uma mutação seja confirmada. Quando isso acontecer, faça $n_{esc} = n$.
8. Efetue a mutação da iteração atual: faça $\mathbf{X} = \mathbf{X}_{\mathbf{n_{esc}}}$ e calcule F(**X**).
9. Repita os passos 4 a 8 até que um dado critério de parada seja satisfeito.
10. Retorne $\mathbf{X}_{\mathbf{melhor}}$ e F($\mathbf{X}_{\mathbf{melhor}}$).

O algoritmo GEO_{var4} desta segunda etapa é composto dos seguintes passos:

1. Inicialize aleatoriamente e com distribuição uniforme entre $\mathbf{X}_{\mathbf{min}}$ e $\mathbf{X}_{\mathbf{max}}$ um vetor **X** contendo as N variáveis de projeto. Calcule o valor da função objetivo F(**X**), faça $\mathbf{X}_{\mathbf{melhor}} = \mathbf{X}$ e guarde F($\mathbf{X}_{\mathbf{best}}$).
2. Defina o número de mutações l_j, $j \in \{1,2,...,N\}$ de cada variável X_j, tal que $\sum_j l_j = L$, sendo L número total de mutações agindo sobre as N variáveis de projeto. Defina um valor para b, a base para a representação das mutações, tal que b>1.
3. Calcule o vetor **e**, onde $e_j = (x_{maxj}-x_{minj})/(b^{l_j}-1)$ e j é o índice da variável, isto é, $j \in \{1,2,...,N\}$. O elemento e_j define a resolução de mutação da j-ésima variável de projeto. Por resolução entenda-se o menor valor a ser somado ou subtraído de X_j.
4. Faça $F(\mathbf{X})_{ref} = F(\mathbf{X})$, $F(\mathbf{X}_{\mathbf{melhor}})_{ref} = F(\mathbf{X}_{\mathbf{melhor}})$.
5. Para cada variável $j \in \{1,2,...,N\}$ do vetor **X**, faça:
    a. Converta X_j para sua codificação b-equivalente usando l_j bits e obtenha C_j. Ou seja:
        1º. Faça $R=X_j$.
        2º. Para $i \in \{l_j, l_j-1, ..., 1\}$ faça:

I. Calcule $F=\frac{(X_{max_j} - X_{min_j})}{b^{(l_j - i + 1)}}$.

II. Se R>F, então faça $c_{j,i}$=1 e R= R-F. Senão, faça $c_{j,i}$=0.

b. Para cada mutação $i \in \{1,2,..., l_j\}$ da variável X_j, faça:

1º. Calcule o tamanho da mutação $m= b^{(i-1)} \cdot e_j$.

2º. Calcule o sinal da mutação $s=(-1)^{(c_{j,i})}$, onde $c_{j,i}$ é o valor calculado no passo 5.a.

3º. Mute X_j. Primeiro, faça $X_{aux}= X_j$. Em seguida, faça $X_j= X_j+s \cdot m$, gerando um vetor mutado $\mathbf{X_i}$. Verifique os limites: Se $X_j > X_{max_j}$ ou $X_j < X_{min_j}$, então sorteie um novo $X_j \in [X_{min_j}, X_{max_j}]$ com distribuição uniforme e faça m= abs(X_j-X_{aux}) e s=(X_j-X_{aux})/m.

4º. Calcule o valor da função objetivo F($\mathbf{X_i}$). Atribua à mutação i um valor de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - F(\mathbf{X_{best}})_{ref}$, que indica o ganho ou a perda que a função objetivo tem se a mutação i ocorrer, quando comparada com o melhor valor da função objetivo encontrado até a iteração anterior. Em seguida, se $F(\mathbf{X_i}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X_i})$ e $\mathbf{X_{melhor}} = \mathbf{X_i}$.

5º. Retorne **X** à sua condição não mutada: faça $X_j= X_j-s \cdot m$.

c. Ordene todas as mutações $i \in \{1,2,..., l_j\}$ da variável X_j de acordo com os seus índices de adaptação, de k=1 para a menos adaptada à $k=l_j$, para a mais adaptada. Crie uma lista de valores (i,k) relacionando a mutação i com seu respectivo número de ordem k, onde $i,k \in \{1,2,..., l_j\}$. Se $\Delta F(\mathbf{X_i}) = \Delta F(\mathbf{X_u})$, i?u, estabeleça a ordem entre i e u de forma aleatória.

d. Escolha com igual probabilidade uma mutação candidata i para ser efetivada. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual à $P_i(k)$, confirme a mutação i para ser efetivamente aplicada. Repetir este passo até que uma mutação seja confirmada. Quando isso acontecer, faça $i_{esc_j} = i$.

6. Efetue as mutações da iteração atual: faça $\mathbf{X} = \mathbf{X_{esc}}$, onde $\mathbf{X_{esc}}$ é o vetor resultante da aplicação em $\mathbf{X}$ das mutações escolhidas no passo 5.d para cada variável X_j (mutações i_{esc_j}, $j \in \{1,2,...,N\}$). Em seguida, calcule $F(\mathbf{X})$.

7. Repita os passos 4 a 6 até que um dado critério de parada seja satisfeito.

8. Retorne $\mathbf{X_{melhor}}$ e $F(\mathbf{X_{melhor}})$.

Como já mencionado, os algoritmos recém expostos descrevem a SA4 da primeira e da segunda etapa para GEO e GEO_{var}. A fim de manter a sincronia entre o texto e a seqüência de eventos, nos próximos parágrafos é feita a apresentação dos testes efetuados com a SA4 em sua primeira etapa, ou seja, considerando-se b=2 apenas.

As Figuras 3.36 a 3.40 apresentam os resultados médios de 50 execuções independentes, em termos do melhor $F(\mathbf{X})$ encontrado ao final de cada execução, obtidos com as versões GEO_4 e GEO_{var4} da primeira etapa. O número de mutações de cada variável foi fixado em l_j=16, $j \in \{1;2;...;N\}$, equivalendo aos 16 bits usados por GEO_1 e GEO_{var1} na codificação das variáveis. O número de avaliações de $F(\mathbf{X})$ a cada execução é 10^4 para FT_1 e 10^5 para as demais funções teste. Nas figuras, cada curva apresenta o resultado para τ variando de 0 a 10. Para fins comparativos, são mostradas também as curvas já obtidas para as mesmas funções com GEO_1 e GEO_{var1}.

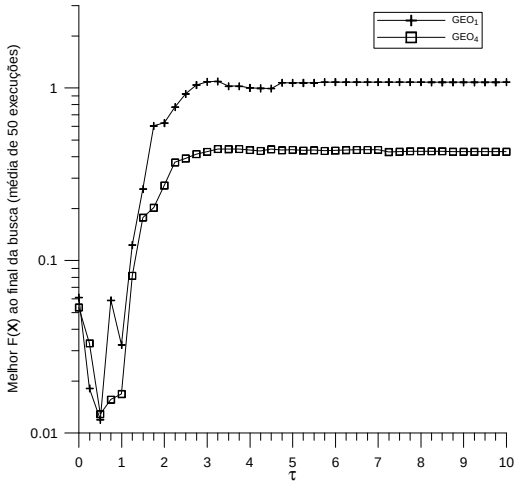


Figura 3.36a - FT_1 x τ para GEO

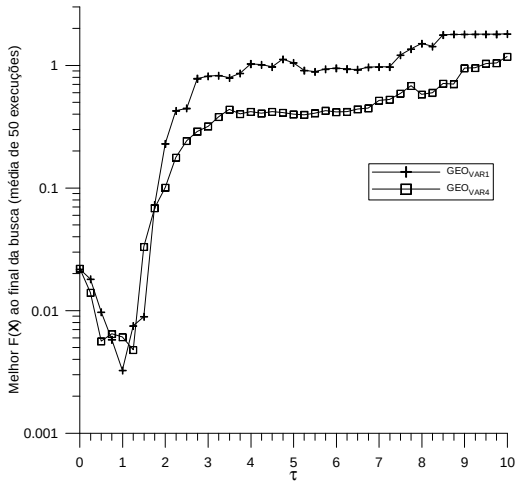


Figura 3.36b - FT_1 x τ para GEO_{var}

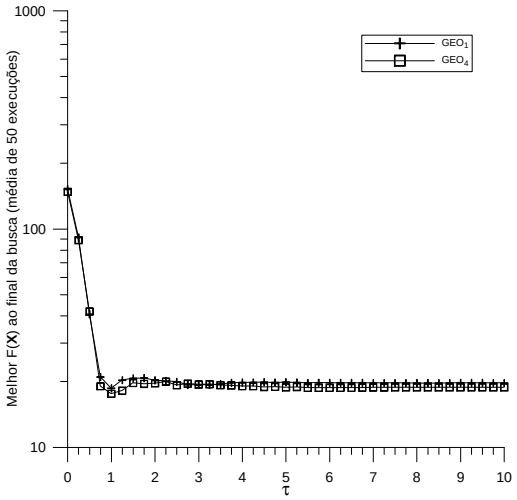


Figura 3.37a - FT_2 x τ para GEO

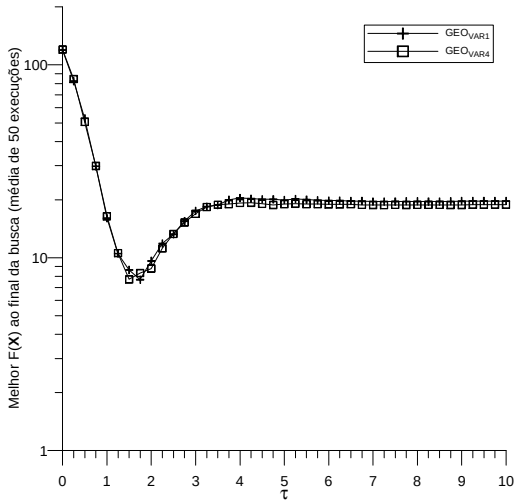


Figura 3.37b - FT_2 x τ para GEO_{var}

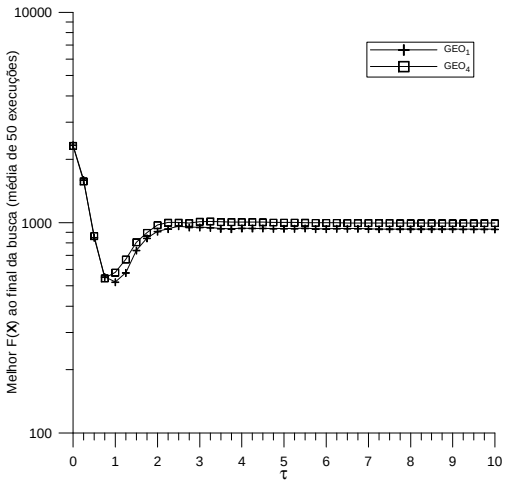


Figura 3.38a - FT_3 x τ para GEO

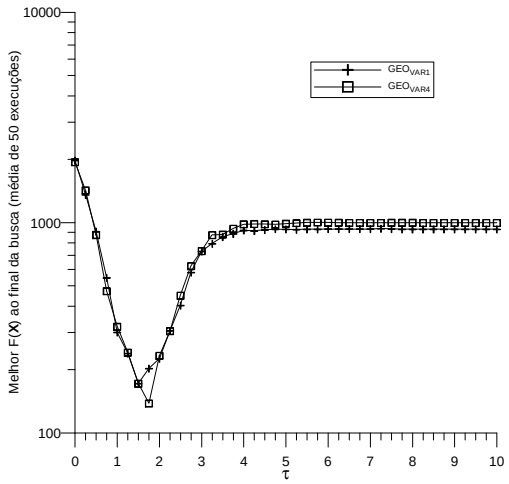


Figura 3.38b - FT_3 x τ para GEO_{var}

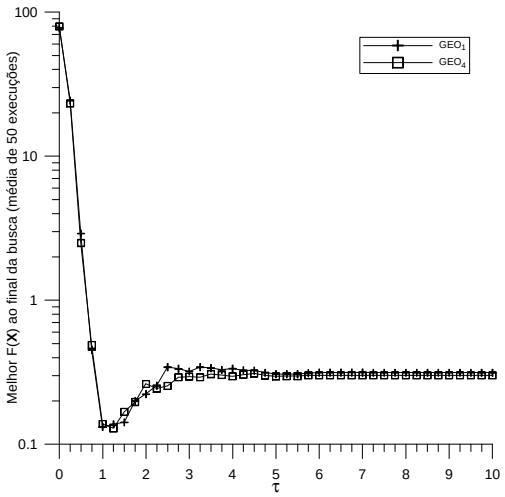


Figura 3.39a - FT_4 x τ para GEO

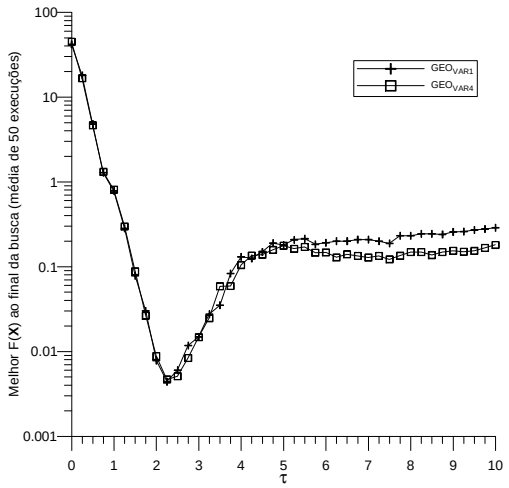


Figura 3.39b - FT_4 x τ para GEO_{var}

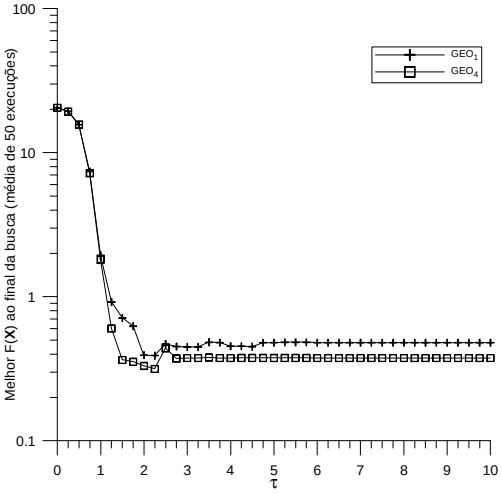


Figura 3.40a - FT_5 x τ para GEO

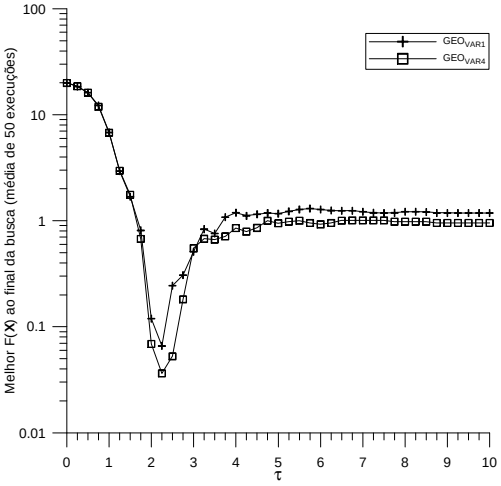


Figura 3.40b - FT_5 x τ para GEO_{var}

Analisando os resultados gráficos recém apresentados, é possível constatar que há grande semelhança entre os resultados obtidos pela SA1 e os obtidos pela SA4 em sua primeira etapa, como esperado. Pequenas diferenças se justificam por ao menos dois motivos. O primeiro motivo é que as inicializações das 50 execuções independentes de SA1 e SA4 não são idênticas. Nestes casos, pequenas variações nos resultados médios são esperadas estatisticamente. O segundo motivo é que o domínio permitido para as variáveis de projeto na SA4 é o domínio dos números reais, ou, para ser mais preciso, o subconjunto dos números racionais oferecido pelo computador empregado nos testes, enquanto na SA1 o domínio é discreto. Muito embora as mutações efetuadas pela SA4 sejam binário-equivalentes, a inicialização das variáveis de projeto não o é, permitindo à SA4 começar de qualquer ponto dentro do domínio real das variáveis de projeto, fato que não ocorre com a SA1, que só pode inicializar as variáveis de projeto dentro do domínio discreto permitido pela codificação binária.

Passando a analisar a SA4 em sua segunda etapa, tem-se que agora o tamanho da menor mutação possível é obtido reescrevendo-se a equação 3.4 conforme segue:

$$e_j = \frac{(X_{max_j} - X_{min_j})}{\left(b^{l_j} - 1\right)} \tag{3.6}$$

onde b é a base do sistema de representação numérica utilizado pela SA4 da segunda etapa. Além disso, tem-se que o tamanho de uma perturbação induzida na variável j pela i-ésima mutação é dado por:

$$m_{i,j} = b^{(i-1)} e_j, \quad i \in \{1,2,...,l_j\}, \quad j \in \{1,2,...,N\} \tag{3.7}$$

Indo um pouco além, tem-se que as mutações de tamanho mínimo e máximo para uma variável j são dadas por:

$$m_{\min j} = \min(m_{i,j} \mid j = j, i \in \{1,2,...,l_j\}) = \min(b^{(i-1)} e_j) = b^{(0)} e_j = e_j = \frac{(X_{\max_j} - X_{\min_j})}{(b^{(l_j)} - 1)} \tag{3.8}$$

$$m_{\max j} = \max(m_{i,j} \mid j = j, i \in \{1,2,...,l_j\}) = \max(b^{(i-1)} e_j) = b^{(l_j - 1)} e_j = \frac{b^{(l_j - 1)}}{(b^{(l_j)} - 1)} (X_{\max_j} - X_{\min_j}) \tag{3.9}$$

Rearranjando as equações 3.8 e 3.9 a fim de obter os tamanhos das mutações como frações adimensionais do intervalo admissível da variável de projeto j, têm-se que:

$$m_{\min j}^{*} = \frac{m_{\min j}}{(X_{\max_j} - X_{\min_j})} = \frac{1}{(b^{(l_j)} - 1)} \tag{3.10}$$

$$m_{\max j}^{*} = \frac{m_{\max j}}{(X_{\max_j} - X_{\min_j})} = \frac{b^{(l_j - 1)}}{(b^{(l_j)} - 1)} \tag{3.11}$$

A partir do equacionamento recém mostrado, é possível perceber que a utilização de b como parâmetro traz como conseqüência que, para um mesmo l_j, quanto maior o valor de b menores os tamanhos da menor e da maior mutação. Isto implica também que quanto maior o valor de b maior a localidade da busca, conforme discutido nos próximos parágrafos.

A Tabela 3.4 apresenta valores percentuais de m_{min_j}* e m_{max_j}* para b$\in$ {1,05; 2; 20} e para diversos valores de l_j. Como pode ser visto na Tabela 3.4, por exemplo, para l_j=16, com um valor para a base tão alto quanto b=20 a menor mutação tem um tamanho menor do que 0,0001% do comprimento do espaço de busca, enquanto a maior mutação tem um tamanho de apenas 5,0%. Então, uma busca feita com b=20 significa que a SA4 a cada iteração alcança somente uma porção do espaço de busca 5% distante do ponto atual, caracterizando uma busca bastante localizada. Agora, ainda com l_j=16, mas com b=2, a

menor mutação tem um tamanho de cerca de 0,002% do comprimento do espaço de busca enquanto a maior delas tem um tamanho de 50%. Conforme já comentado, este é o caso do GEO canônico, onde b=2 é usado implicitamente.

Tabela 3.4 – Tamanhos mínimo e máximo das mutações da variável de projeto X_j.

l_j	b					
	1,05		2		20	
	m_{min_j}* (%)	m_{max_j}* (%)	m_{min_j}* (%)	m_{max_j}* (%)	m_{min_j}* (%)	m_{max_j}* (%)
1	2000,0	2000,0	100,0	100,0	5,26	5,26
2	975,6	1024,4	33,3	66,7	0,25	5,01
3	634,4	699,4	14,3	57,1	0,01	5,00
4	464,0	537,2	6,7	53,3	0,0006	5,00
8	209,4	294,7	0,39	50,2	<0,0001	5,00
16	84,5	175,8	0,002	50,0	<0,0001	5,00
32	26,6	120,5	<0,0001	50,0	<0,0001	5,00

Antes de continuar a análise dos dados da Tabela 3.4, é importante ressaltar alguns fatos. No GEO original, os bits ao serem mutados, definem simultaneamente o tamanho e o sinal da mutação (de 0? 1 é + e de 1? 0 é -). Mais do que isso, o valor da base corresponde ao tamanho (número de componentes) do alfabeto numérico usado (símbolos 0 e 1). Este fato garante que as mutações dos bits resultem em novos valores que estão sempre dentro dos limites laterais estabelecidos para as respectivas variáveis de projeto. As mutações são, por assim dizer, implicitamente autocontidas. Esta propriedade não é exclusividade do sistema binário, ela ocorre em qualquer sistema numérico onde o tamanho do alfabeto e o valor da base coincidam. Na SA4, por sua vez, exceto quando b=2, não há mais correspondência entre a base e o tamanho do alfabeto numérico utilizado. Como conseqüência, as mutações geradas pela SA4 não possuem a propriedade de autocontenção. Por conta disso, mutações que extrapolam os limites laterais estabelecidos para as variáveis de projeto podem ocorrer quando b<2 for usado pela SA4 da segunda etapa. Longe de se tornar um fator negativo, esta característica é utilizada pela SA4 para diminuir a localidade da busca. Agora, após cada mutação, a SA4 verifica os limites da respectiva variável de projeto. Se existe uma violação, então uma nova mutação dentro dos limites é gerada através de um sorteio com distribuição uniforme.

Retomando a análise dos dados da Tabela 3.4, o primeiro valor de base mostrado é para b=1,05. Para esta base e l_j=16, a menor mutação tem um tamanho de cerca de 85% do espaço de busca enquanto a maior tem um tamanho de 175,8%. Agora, a busca é bastante esparsa, porque mesmo a menor mutação tem um comprimento de 85% do espaço de busca. É importante observar que, por causa da verificação de limites laterais feita a cada mutação (passo 5.b.4° do GEO_4), qualquer mutação com tamanho maior do que 100% leva necessariamente a uma nova mutação aleatoriamente escolhida sobre a totalidade do espaço de busca da respectiva variável de projeto. Mutações com tamanho menor do que 100% também podem levar a mutações aleatórias, dependendo de quão próxima da borda do espaço de busca está a variável de projeto antes da mutação e do sinal da mutação.

A seguir, são apresentados os resultados dos testes efetuados com a SA4 em sua segunda etapa. As Figuras 3.41 a 3.45 apresentam os resultados médios de 50 execuções independentes, em termos do melhor F(**X**) encontrado ao final de cada execução, obtidos com a segunda etapa de GEO_4 e GEO_{var4}. O número de mutações de cada variável foi fixado em l_j=16, j∈{1;2;...;N}, equivalendo aos 16 bits usados por GEO_1 e GEO_{var1} na codificação das variáveis. O número de avaliações de F(**X**) a cada execução é 10^4 para FT_1 e 10^5 para as demais funções teste. Nas figuras, cada curva apresenta o resultado para τ variando de 0 a 10 e para valores de b∈{1,2; 1,3; 1,618; 2; 3}. A escolha dos valores de b é arbitrária, mas cumpre a contento o objetivo de averiguar a influência deste parâmetro no desempenho da SA4.

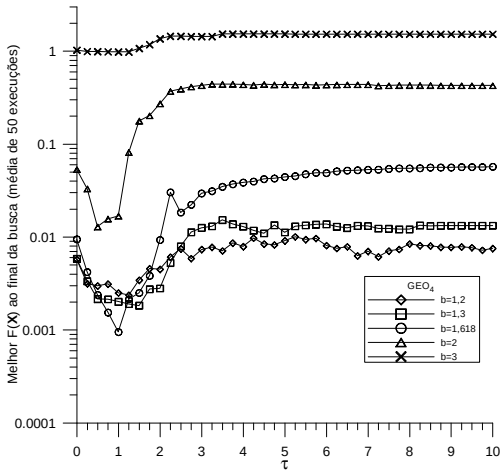


Figura 3.41a - FT_1 x τ x b para GEO_4

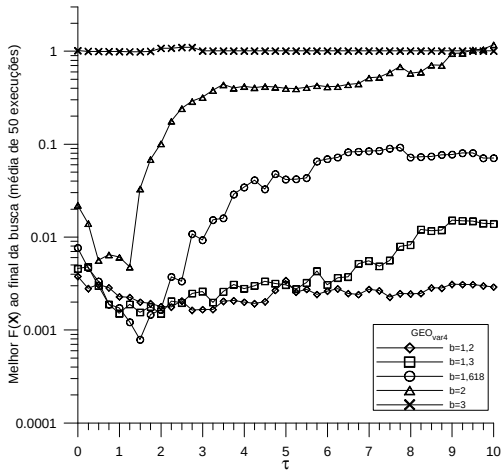


Figura 3.41b - FT_1 x τ x b para GEO_{var4}

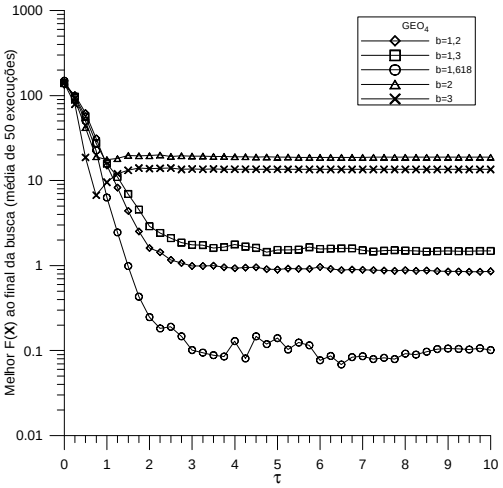


Figura 3.42a - FT_2 x τ x b para GEO_4

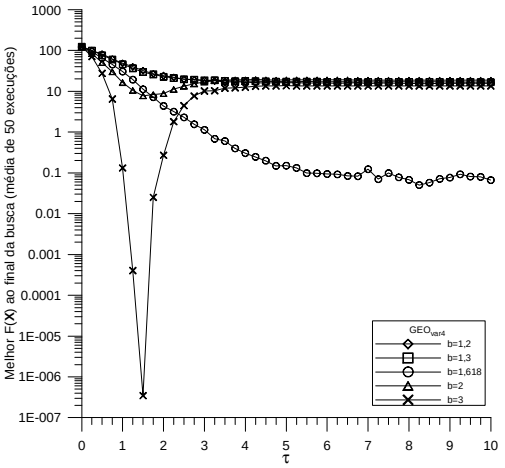


Figura 3.42b - FT_2 x τ x b para GEO_{var4}

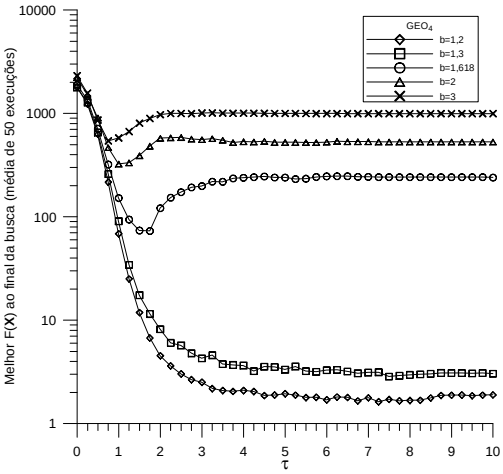


Figura 3.43a - FT_3 x τ x b para GEO_4

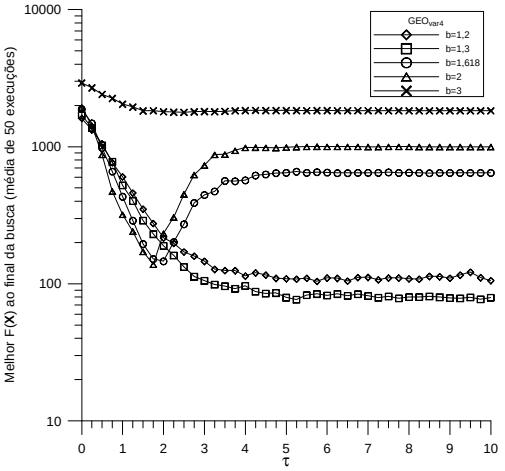


Figura 3.43b - FT_3 x τ x b para GEO_{var4}

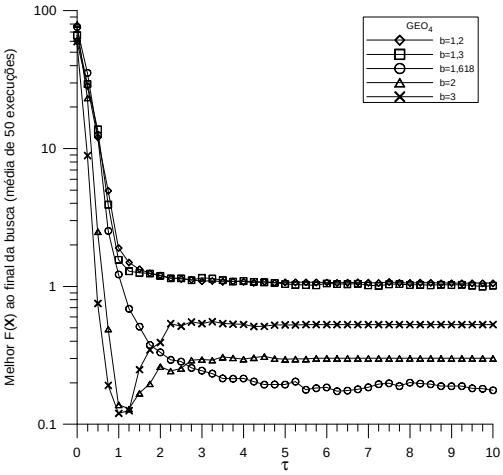


Figura 3.44a - FT_4 x τ x b para GEO_4

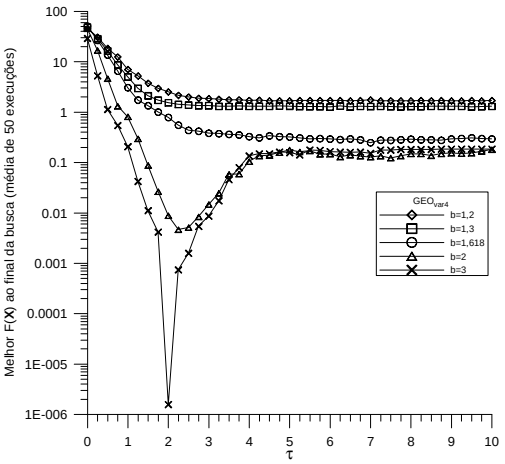


Figura 3.44b - FT_4 x τ x b para GEO_{var4}

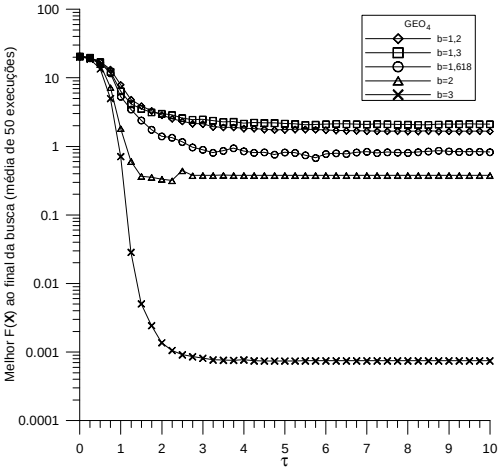


Figura 3.45a - FT_5 x τ x b para GEO_4

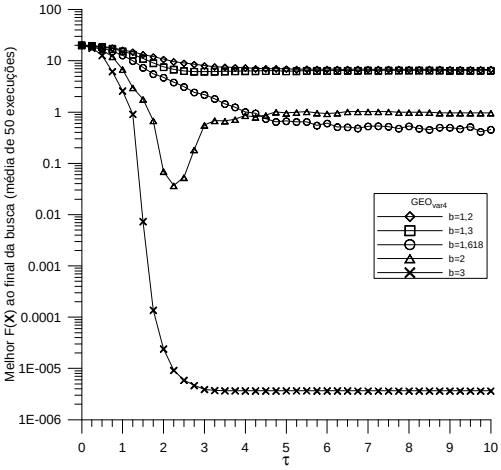


Figura 3.45b - FT_5 x τ x b para GEO_{var4}

A análise das Figuras 3.41 a 3.45 revela que, quando comparada à SA4 anterior (que equivale a b=2 nas Figuras 3.41 a 3.45), houve ganhos significativos de desempenho para praticamente todas as funções teste, seja com GEO_4, seja com GEO_{var4}.

Compilando, com base nas Figuras 3.41 a 3.45, os pares (b;τ) com os quais obteve-se os melhores resultados médios para cada função teste, é possível montar a Tabela 3.5.

Tabela 3.5 – Pares (b;τ) que obtêm o melhor resultado médio (segunda etapa)[1].

FT	GEO_4		GEO_{var4}	
	b	τ	b	τ
FT_1	1,618	1	1,618	1,5
FT_2	1,618	6,5	3	1,5
FT_3	1,2	7,25	1,3	5,25
FT_4	3	1	3	2
FT_5	3	>4	3	>3,5

[1] Convenção: ">4", por exemplo, indica que o melhor resultado foi obtido para qualquer valor do parâmetro que seja maior do que 4.

A partir da Tabela 3.5, é possível perceber que, para uma dada função teste, quase não há variação no valor de b que leva ao melhor resultado, seja com GEO_4, seja com GEO_{var4}. A única exceção é a FT_2, com b=1,618 para GEO_4 e com b=3 para GEO_{var4}. No

caso do parâmetro τ, não existe tal uniformidade/correspondência de valores entre GEO_4 e GEO_{var4}.

Finalmente, observa-se a partir das Figuras 3.42b e 3.44b que ocorrem picos de desempenho para GEO_{var4} com FT_2 e FT_4 sendo estes picos sensíveis ao parâmetro b e ainda mais sensíveis ao parâmetro τ. Não foi possível encontrar qual a causa ou causas para a ocorrência desse fenômeno que afeta apenas o GEO_{var4}.

A terceira e última etapa da SA4 advém de uma característica existente na codificação binária (e nas outras também), que é a impossibilidade de percorrer todo o espaço de busca de uma variável, indo de X_{min_j} a X_{max_j}, alterando-se apenas um bit (um dígito) por vez. Este fato é ilustrado graficamente com o auxílio da Figura 3.46 a seguir.

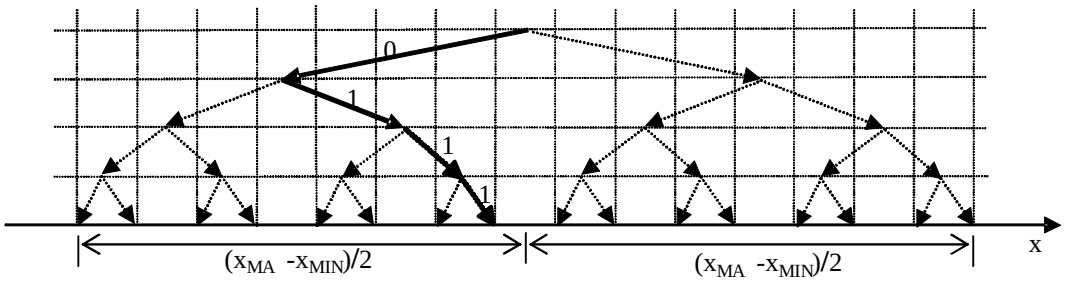


Figura 3.46 – Representação gráfica do ponto “0111”.

Na Figura 3.46, o deslocamento para ir do ponto "0111" para o ponto adjacente situado à direita, requer mutação de todos os quatro bits, passando-os para "1000". O inverso também é verdade quando se quer ir de "1000" para "0111". Embora este seja o caso extremo e só ocorra uma vez, é possível observar na figura que existem mais seis deslocamentos adjacentes que somente são possíveis mudando-se pelo menos dois bits. A fórmula $2^{(l_j-1)}$-1 permite calcular a quantidade destes deslocamentos para qualquer l_j. O GEO e GEO_{var} canônicos, bem como a SA4 atual, utilizam direta ou indiretamente mutações à maneira binária, onde apenas mutações equivalentes à modificação de um único bit em cada variável de projeto são efetuadas. Portanto, estes algoritmos não são capazes durante uma busca de passar do ponto "0111" para o ponto "1000" numa única iteração. É razoável imaginar que, dependendo do relevo da função objetivo, esta característica seja um fator que limite o desempenho ou mesmo impeça o algoritmo de

encontrar a solução ótima. Para evitar esta situação, a terceira etapa da SA4 realiza as mutações com uma estruturação diferente da existente na codificação binária. A nova estruturação maximiza a uniformidade da distribuição das mutações em torno do ponto atual, tornando-a independente do valor dos bits que formam sua codificação binária equivalente. Isto é alcançado sorteando-se, a cada iteração, um lado (esquerdo ou direito) para a primeira mutação (a menos significativa) e alternando as demais, de modo que uma mutação à esquerda seja sempre seguida por uma mutação à direita e vice-versa. A explicação recém apresentada utiliza a analogia ou representação geométrica dada pela Figura 3.46. Em termos matemáticos, o que ocorre é que os sinais das mutações (- ou +) são alternados a partir do sinal da primeira mutação, que é obtido por sorteio com distribuição uniforme. Garante-se, deste modo, que a distribuição espacial das mutações de cada variável de projeto seja o mais uniforme possível no entorno do ponto atual, além de ser independente dos valores assumidos pelos bits da configuração binária equivalente ao ponto atual da variável de projeto.

O algoritmo GEO_4 desta terceira e última etapa é composto dos seguintes passos:

1. Inicialize aleatoriamente e com distribuição uniforme entre $\mathbf{X_{min}}$ e $\mathbf{X_{max}}$ um vetor $\mathbf{X}$ contendo as N variáveis de projeto. Calcule o valor da função objetivo $F(\mathbf{X})$, faça $\mathbf{X_{melhor}} = \mathbf{X}$ e guarde $F(\mathbf{X_{best}})$.

2. Defina o número de mutações l_j, $j \in \{1,2,...,N\}$ de cada variável X_j, tal que $\sum_j l_j = L$, sendo L número total de mutações agindo sobre as N variáveis de projeto. Defina um valor para b, a base para a representação das mutações, tal que b>1.

3. Calcule o vetor **e**, onde $e_j = (X_{maxj}-X_{minj})/(b^{l_j}-1)$, e j é o índice da variável, isto é, $j \in \{1,2,...,N\}$. O elemento e_j define a resolução de mutação da j-ésima variável de projeto. Por resolução entenda-se o menor valor a ser somado ou subtraído de X_j.

4. Faça $F(\mathbf{X})_{ref} = F(\mathbf{X})$, $F(\mathbf{X_{melhor}})_{ref} = F(\mathbf{X_{melhor}})$.

5. Faça n=0. Para cada variável $j \in \{1,2,...,N\}$ do vetor $\mathbf{X}$, faça:

    a. Sorteie com distribuição uniforme um valor para $c \in \{0, 1\}$.

    b. Para cada mutação $i \in \{1,2,..., l_j\}$ da variável X_j, faça:

1º. Incremente o número total de mutações do vetor **X**: faça n= n+1.

2º. Calcule o tamanho da mutação m= $b^{(i-1)} \cdot e_j$.

3º. Calcule o sinal da mutação s= $(-1)^{(i-C)}$, onde c é o valor binário sorteado no passo 5.a.

4º. Mute X_j. Primeiro, faça X_{aux}= X_j. Em seguida, faça X_j= X_j+s·m, gerando um vetor mutado $\mathbf{X_n}$. Verifique os limites: Se $X_j > X_{max_j}$ ou $X_j < X_{min_j}$, então sorteie um novo $X_j \in [X_{min_j}, X_{max_j}]$ com distribuição uniforme e faça m= abs(X_j-X_{aux}) e s=(X_j-X_{aux})/m.

5º. Calcule o valor da função objetivo F($\mathbf{X_n}$). Atribua à mutação n um valor de adaptação $\Delta F(\mathbf{X_n}) = F(\mathbf{X_n}) - F(\mathbf{X_{best}})_{ref}$, que indica o ganho ou a perda que a função objetivo tem se a mutação n ocorrer, quando comparada com o melhor valor da função objetivo encontrado até a iteração anterior. Em seguida, se F($\mathbf{X_n}$) < F($\mathbf{X_{melhor}}$) então faça F($\mathbf{X_{melhor}}$)= F($\mathbf{X_n}$) e $\mathbf{X_{melhor}} = \mathbf{X}_n$.

6º. Retorne **X** à sua condição não mutada: faça X_j= X_j-s·m.

6. Ordene todas as mutações n∈ {1,2,...,L} de acordo com os seus índices de adaptação, de k=1 para as menos adaptadas à k=L, para a mais adaptada. Crie uma lista de valores (n,k) relacionando a mutação n com seu respectivo número de ordem k, onde n,k∈ {1,2,...,L}. Se $\Delta F(\mathbf{X_n}) = \Delta F(\mathbf{X_u})$, n?u, estabeleça a ordem entre n e u de forma aleatória.

7. Escolha com igual probabilidade uma mutação candidata n para ser efetivada. Calcule $P_n(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual à $P_n(k)$, confirme a mutação n para ser efetivamente aplicada. Repetir este passo até que uma mutação seja confirmada. Quando isso acontecer, faça n_{esc} = n.

8. Efetue a mutação da iteração atual: faça $\mathbf{X} = \mathbf{X_{n_{esc}}}$ e calcule F(**X**).

9. Repita os passos 4 a 8 até que um dado critério de parada seja satisfeito.

10. Retorne $\mathbf{X_{melhor}}$ e F($\mathbf{X_{melhor}}$).

O algoritmo GEO_{var4} desta terceira etapa é composto dos seguintes passos:

1. Inicialize aleatoriamente e com distribuição uniforme entre $\mathbf{X_{min}}$ e $\mathbf{X_{max}}$ um vetor $\mathbf{X}$ contendo as N variáveis de projeto. Calcule o valor da função objetivo F($\mathbf{X}$), faça $\mathbf{X_{melhor}} = \mathbf{X}$ e guarde F($\mathbf{X_{best}}$).
2. Defina o número de mutações l_j, $j \in \{1,2,...,N\}$ de cada variável X_j, tal que $\sum_j l_j = L$, sendo L número total de mutações agindo sobre as N variáveis de projeto. Defina um valor para b, a base para a representação das mutações, tal que b>1.
3. Calcule o vetor **e**, onde $e_j = (X_{maxj}-X_{minj})/(b^{l_j}-1)$ e j é o índice da variável, isto é, $j \in \{1,2,...,N\}$. O elemento e_j define a resolução de mutação da j-ésima variável de projeto. Por resolução entenda-se o menor valor a ser somado ou subtraído de X_j.
4. Faça $F(\mathbf{X})_{ref} = F(\mathbf{X})$, $F(\mathbf{X_{melhor}})_{ref} = F(\mathbf{X_{melhor}})$.
5. Para cada variável $j \in \{1,2,...,N\}$ do vetor $\mathbf{X}$, faça:
    a. Sorteie com distribuição uniforme um valor para $c \in \{0, 1\}$.
    b. Para cada mutação $i \in \{1,2,..., l_j\}$ da variável X_j, faça:
        1º. Calcule o tamanho da mutação $m = b^{(i-1)} \cdot e_j$.
        2º. Calcule o sinal da mutação $s = (-1)^{(i-C)}$, onde c é o valor binário sorteado no passo 5.a.
        3º. Mute X_j. Primeiro, faça $X_{aux} = X_j$. Em seguida, faça $X_j = X_j + s \cdot m$, gerando um vetor mutado $\mathbf{X_i}$. Verifique os limites: Se $X_j > X_{max_j}$ ou $X_j < X_{min_j}$, então sorteie um novo $X_j \in [X_{min_j}, X_{max_j}]$ com distribuição uniforme e faça $m = abs(X_j - X_{aux})$ e $s = (X_j - X_{aux})/m$.
        4º. Calcule o valor da função objetivo F($\mathbf{X_i}$). Atribua à mutação i um valor de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - F(\mathbf{X_{best}})_{ref}$, que indica o ganho ou a perda que a função objetivo tem se a mutação i ocorrer, quando comparada com o melhor valor da função objetivo encontrado até a iteração anterior. Em seguida, se $F(\mathbf{X_i}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X_i})$ e $\mathbf{X_{melhor}} = \mathbf{X_i}$.
        5º. Retorne $\mathbf{X}$ à sua condição não mutada: faça $X_j = X_j - s \cdot m$.

c. Ordene todas as mutações $i \in \{1,2,..., l_j\}$ da variável X_j de acordo com os seus índices de adaptação, de k=1 para a menos adaptada à $k=l_j$, para a mais adaptada. Crie uma lista de valores (i,k) relacionando a mutação i com seu respectivo número de ordem k, onde $i,k \in \{1,2,..., l_j\}$. Se $\Delta F(\mathbf{X_i}) = \Delta F(\mathbf{X_u})$, i?u, estabeleça a ordem entre i e u de forma aleatória.

d. Escolha com igual probabilidade uma mutação candidata i para ser efetivada. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual à $P_i(k)$, confirme a mutação i para ser efetivamente aplicada. Repetir este passo até que uma mutação seja confirmada. Quando isso acontecer, faça $i_{esc_j} = i$.

6. Efetue as mutações da iteração atual: faça $\mathbf{X} = \mathbf{X_{esc}}$, onde $\mathbf{X_{esc}}$ é o vetor resultante da aplicação em **X** das mutações escolhidas no passo 5.d para cada variável X_j (mutações i_{esc_j}, $j \in \{1,2,...,N\}$). Em seguida, calcule F(**X**).

7. Repita os passos 4 a 6 até que um dado critério de parada seja satisfeito.

8. Retorne $\mathbf{X_{melhor}}$ e $F(\mathbf{X_{melhor}})$.

As Figuras 3.47 a 3.51 apresentam os resultados médios de 50 execuções independentes, em termos do melhor F(**X**) encontrado ao final de cada execução, obtidos com GEO_4 e GEO_{var4} desta terceira etapa. O número de mutações de cada variável foi fixado em l_j=16, $j \in \{1;2;...;N\}$, equivalendo aos 16 bits já usados anteriormente por GEO_1 e GEO_{var1} na codificação das variáveis. O número de avaliações de F(**X**) a cada execução é 10^4 para FT_1 e 10^5 para as demais funções teste. Nas figuras, cada curva apresenta o resultado para τ variando de 0 a 10 e para valores de $b \in \{1,2; 1,3; 1,618; 2; 3\}$.

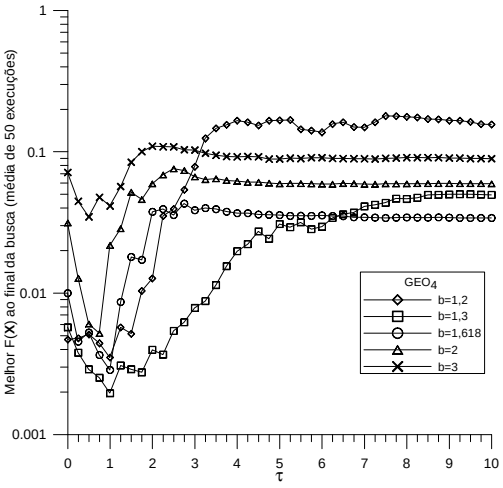


Figura 3.47a - FT_1 x τ x b para GEO_4

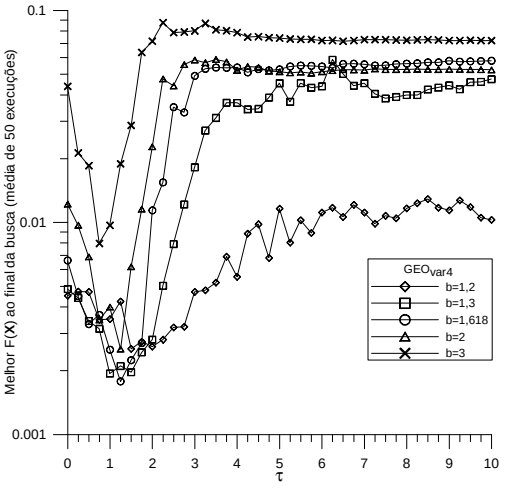


Figura 3.47b - FT_1 x τ x b para GEO_{var4}

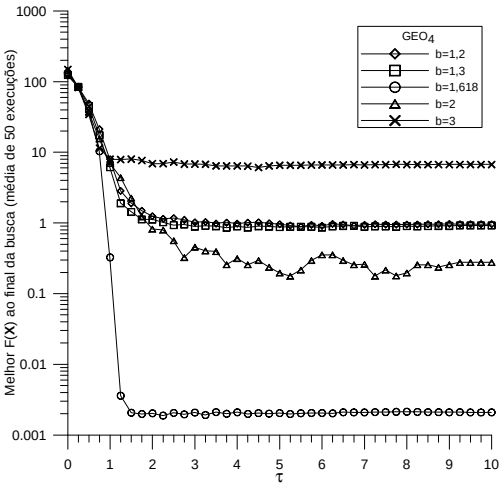


Figura 3.48a - FT_2 x τ x b para GEO_4

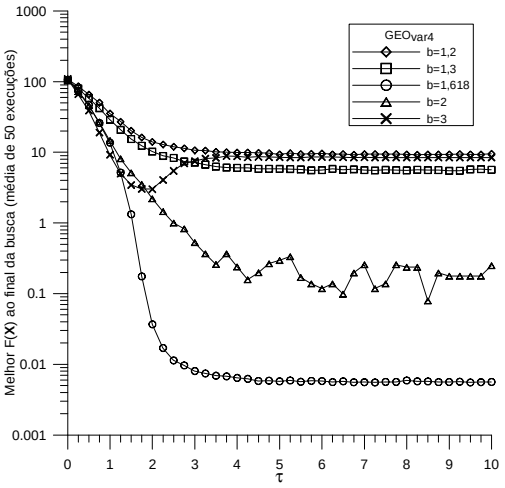


Figura 3.48b - FT_2 x τ x b para GEO_{var4}

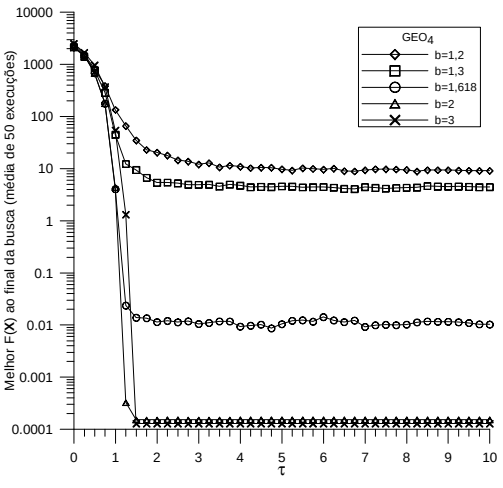


Figura 3.49a - FT_3 x τ x b para GEO_4

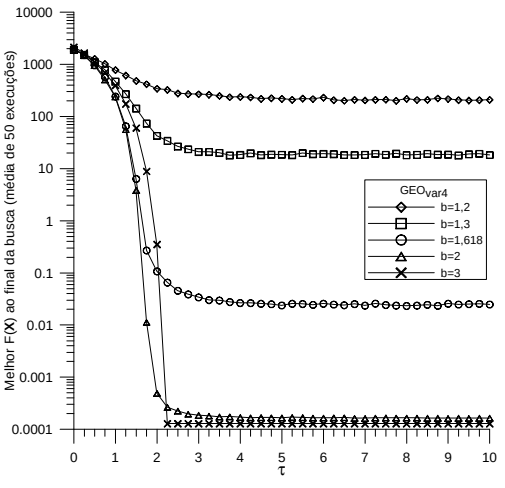


Figura 3.49b - FT_3 x τ x b para GEO_{var4}

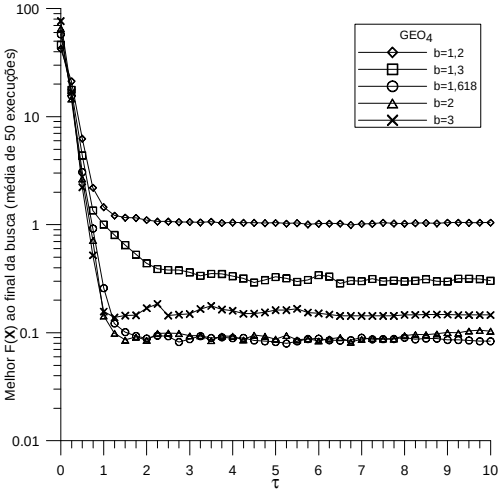


Figura 3.50a - FT_4 x τ x b para GEO_4

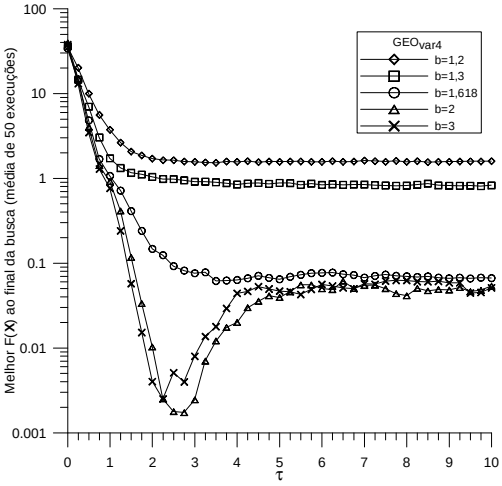


Figura 3.50b - FT_4 x τ x b para GEO_{var4}

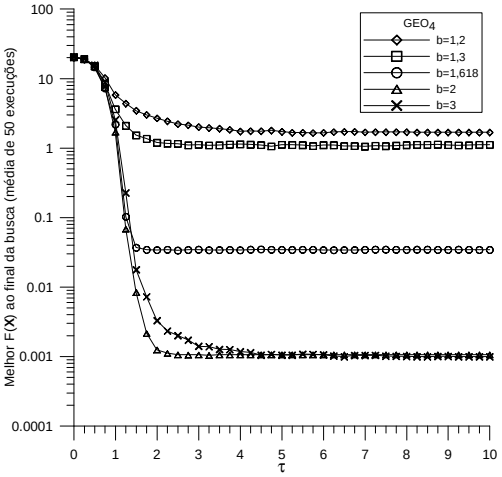


Figura 3.51a - FT_5 x τ x b para GEO_4

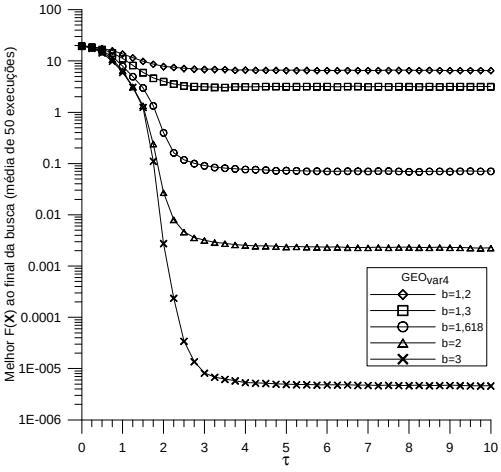


Figura 3.51b - FT_5 x τ x b para GEO_{var4}

Uma análise dos resultados obtidos para cada função teste permite compilar, com base nas Figuras 3.46 a 3.51, os pares (b;τ) com os quais obteve-se os melhores resultados médios para cada função teste. Estas informações estão na Tabela 3.6.

Tabela 3.6 – Pares (b;τ) que obtém o melhor resultado médio (terceira etapa)[1].

FT	GEO_4		GEO_{var4}	
	b	τ	b	τ
FT_1	1,3	1	1,618	1,25
FT_2	1,618	>1,5	1,618	>4
FT_3	3	>1,5	3	>2,25
FT_4	≥2	>1,5	2	2,75
FT_5	≥2	≥4,5	3	>4

[1] Convenção: ">4", por exemplo, indica que o melhor resultado foi obtido para qualquer valor do parâmetro que seja maior do que 4.

Comparando-se os resultados da Tabela 3.6 com os da Tabela 3.5, é possível perceber que agora a dependência ou sensibilidade dos melhores resultados para com os parâmetros b e τ é bem menor, tanto com GEO_4 como com GEO_{var4}. Para τ, por exemplo, exceto pela FT_1 que é atípica por ser unimodal e ter dimensão dois, apenas a FT_4 com GEO_{var4} não obteve um intervalo de valores de τ a partir dos quais o melhor resultado é obtido. Estes resultados indicam que a nova estruturação ou distribuição espacial das mutações torna a SA4 menos sensível aos valores utilizados para τ e, em menor intensidade, aos valores utilizados para b.

A fim de comparar os resultados da segunda etapa com os da terceira etapa da SA4 para cada função teste, apresentam-se nas Figuras 3.52 a 3.56 as curvas que contêm o par (b,τ) de melhor desempenho da segunda e da terceira etapa da SA4. Apresentam-se também as curvas de melhor desempenho da SA1, a fim de permitir uma visualização gráfica dos eventuais ganhos de desempenho da SA4 para com a SA1.

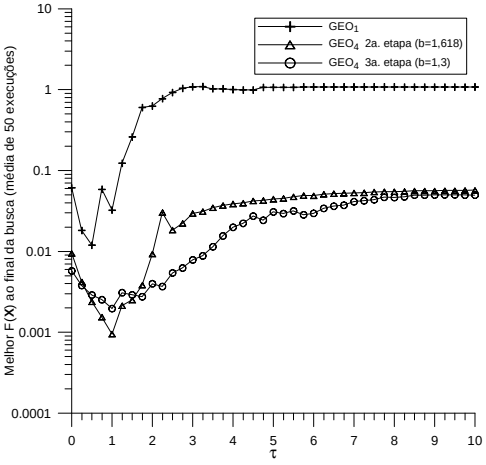


Figura 3.52a - FT_1 x τ para GEO_4

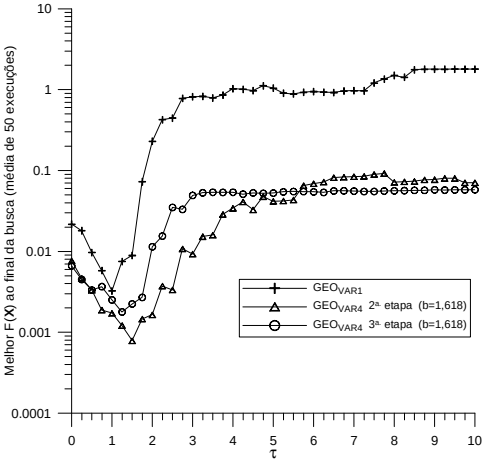


Figura 3.52b - FT_1 x τ para GEO_{var4}

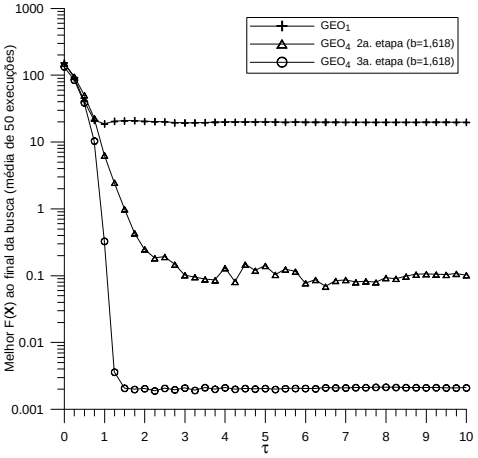


Figura 3.53a - FT_2 x τ para GEO_4

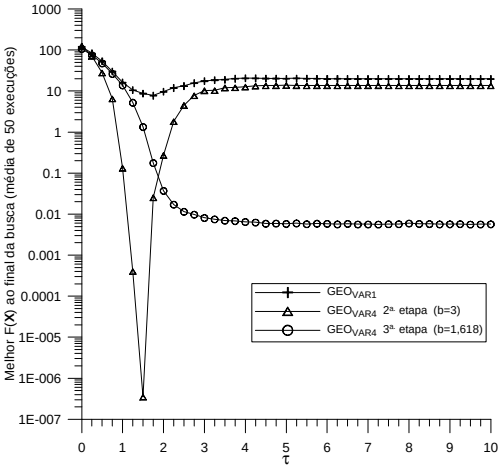


Figura 3.53b - FT_2 x τ para GEO_{var4}

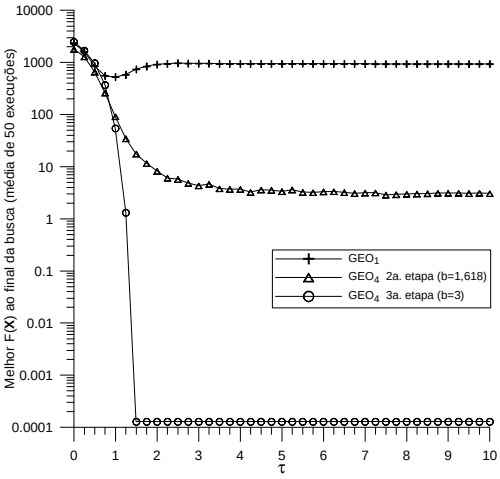


Figura 3.54a - FT_3 x τ para GEO_4

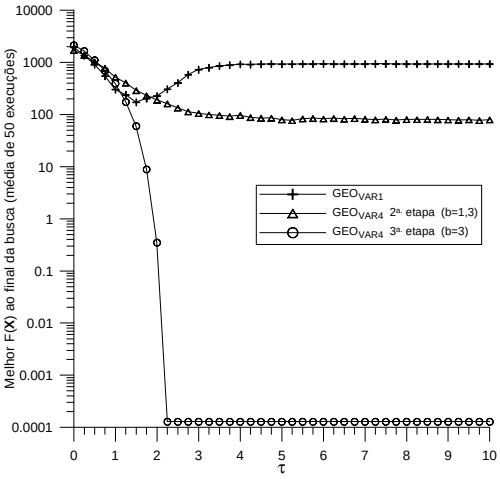


Figura 3.54b - FT_3 x τ para GEO_{var4}

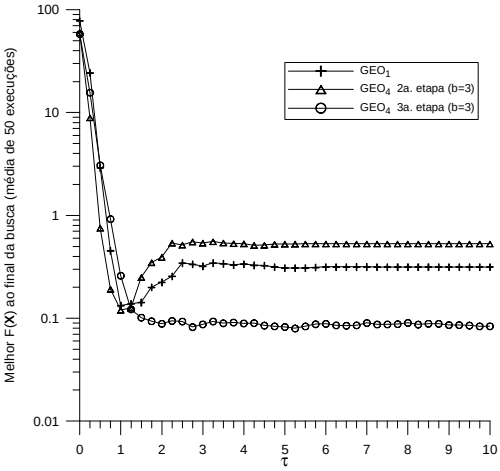


Figura 3.55a - FT_4 x τ para GEO_4

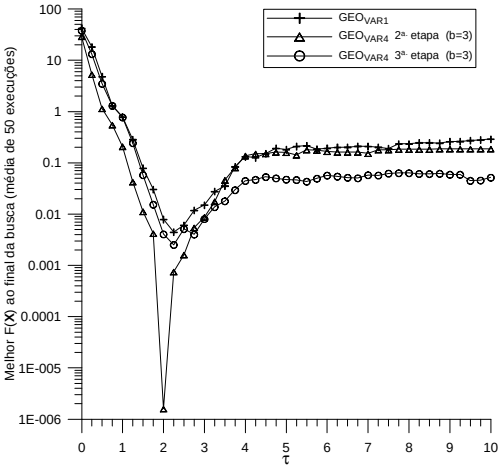


Figura 3.55b - FT_4 x τ para GEO_{var4}

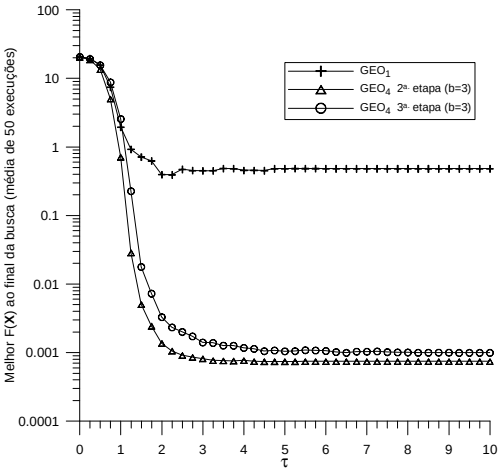


Figura 3.56a - FT_5 x τ para GEO_4

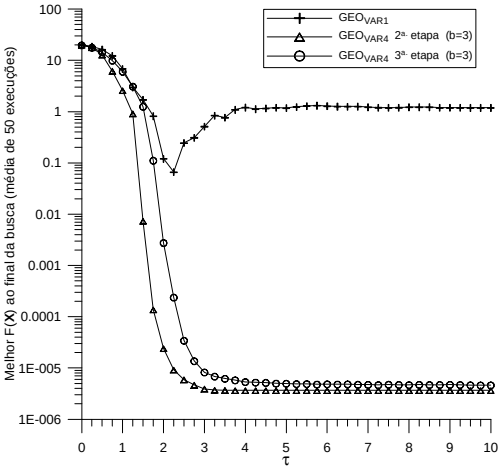


Figura 3.56b - FT_5 x τ para GEO_{var4}

Olhando-se as Figuras 3.52 a 3.56 e comparando-se primeiramente SA4 com SA1, é possível observar que SA4 foi bastante superior à SA1 em todas as figuras, ou seja, para todas as funções teste e para GEO e GEO_{var}. Conclui-se, deste modo, que o uso da

representação real e da base b como parâmetro é fator relevante para melhorar o desempenho de GEO e GEO_{var}. Analisando-se, agora, a influência no desempenho do uso de uma estruturação das mutações não governada pela codificação binária a conclusão não é tão cristalina. Por um lado, a SA4 da segunda etapa apresenta resultados marginalmente superiores em 4 das 10 figuras. Apresenta ainda resultados significativamente superiores em outras 2 das 10 figuras, mas com grande sensibilidade aos parâmetros b e τ. Por outro lado, a SA4 da terceira etapa apresenta resultados significativamente superiores em 3 das 10 figuras e marginalmente superior em 1 das 10 figuras. O placar resumido é então 6 a 4 para a SA4 da segunda etapa. Infelizmente, esse placar não consegue traduzir toda a informação contida nos resultados gráficos das Figuras 3.52 a 3.56. No caso da FT_3, por exemplo, é possível perceber claramente que a SA4 da terceira etapa encontrou com facilidade a solução ótima (vide nota ao pé da Tabela 3.2), o mesmo não ocorrendo para a SA4 da segunda etapa. Este resultado é importante, pois a FT_3 pertence às assim chamadas *deceptive functions*. Essas funções ganharam esta denominação por serem reconhecidamente difíceis de otimizar por algoritmos que utilizam codificação binária, como os Algoritmos Genéticos, por exemplo. Se o resultado obtido pela SA4 da segunda etapa com a FT_3 for representativo de seu desempenho para esta classe de funções, então tal algoritmo não pode ser considerado tão robusto em termos de desempenho quanto o SA4 da terceira etapa. Outro indício de menor robustez da SA4 da segunda etapa são os picos de desempenho obtidos apenas com GEO_{var} para FT_2 e FT_4, que demonstram nestes dois casos grande sensibilidade à variação de b e τ. Esta sensibilidade já foi constatada, inclusive com maior abrangência, quando da análise dos dados da Tabela 3.6.

Os melhores resultados obtidos por GEO_4 e GEO_{var4} , seja da segunda, seja da terceira etapa, são comparados, com o auxílio das Figuras 3.57 a 3.61, com os melhores resultados obtidos pelas versões SA1 e SA2. A versão SA3 não foi incluída na comparação porque, nas comparações já efetuadas da mesma com SA1 e SA2, ficou evidente que a SA3 tem um desempenho bastante inferior.

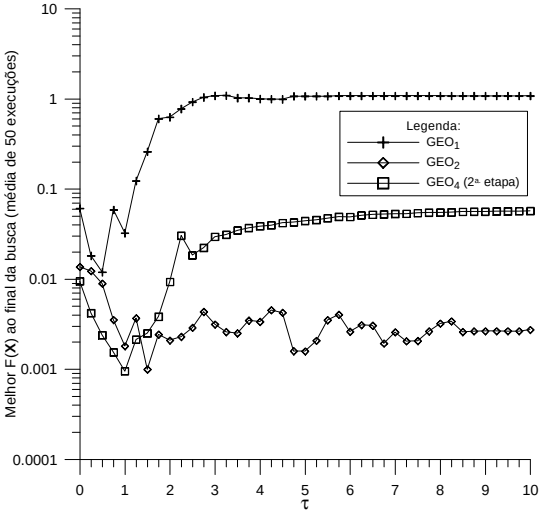


Figura 3.57a - FT_1 x τ x SA para GEO

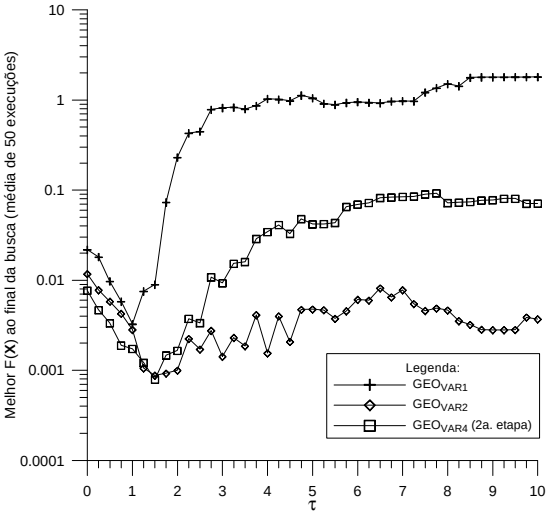


Figura 3.57b - FT_1 x τ x SA para GEO_{var}

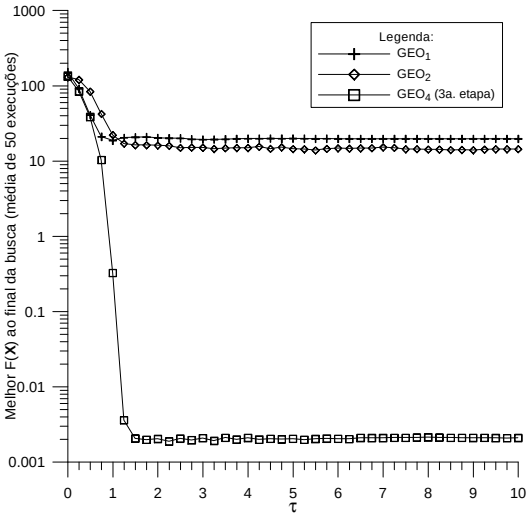


Figura 3.58a - FT_2 x τ x SA para GEO

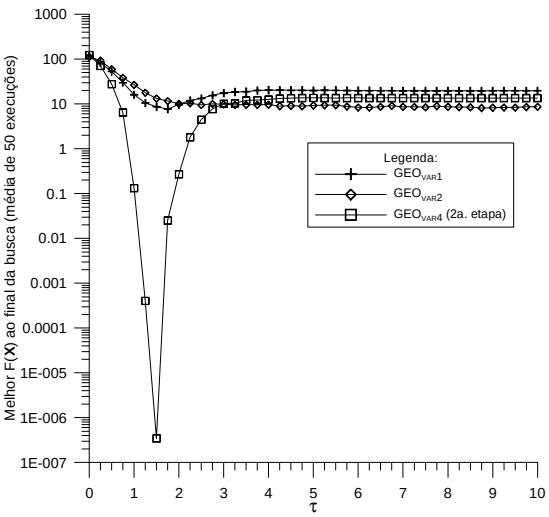


Figura 3.58b - FT_2 x τ x SA para GEO_{var}

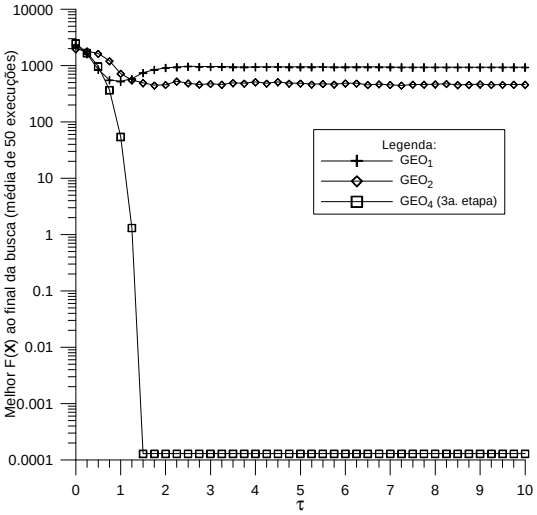


Figura 3.59a - FT_3 x τ x SA para GEO

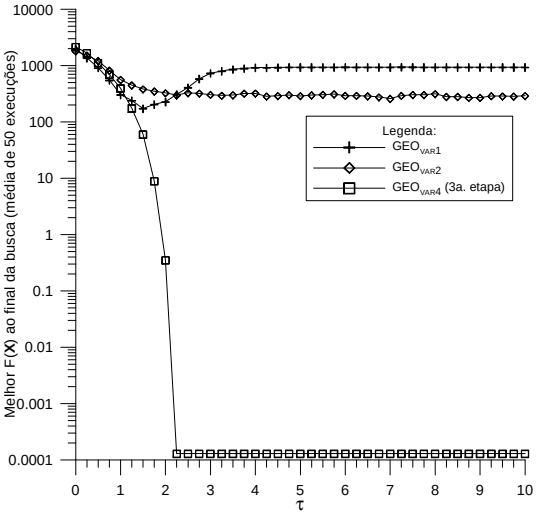


Figura 3.59b - FT_3 x τ x SA para GEO_{var}

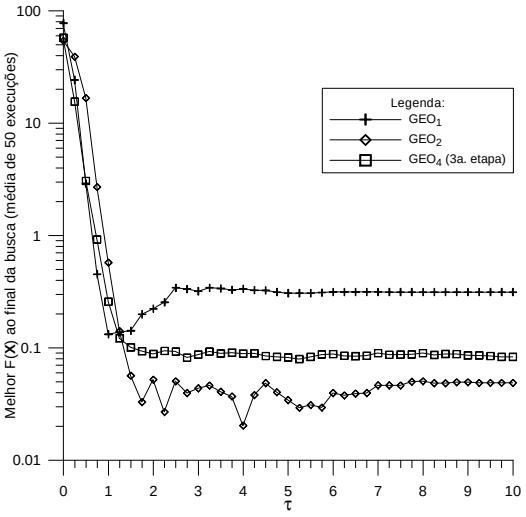


Figura 3.60a - FT_4 x τ x SA para GEO

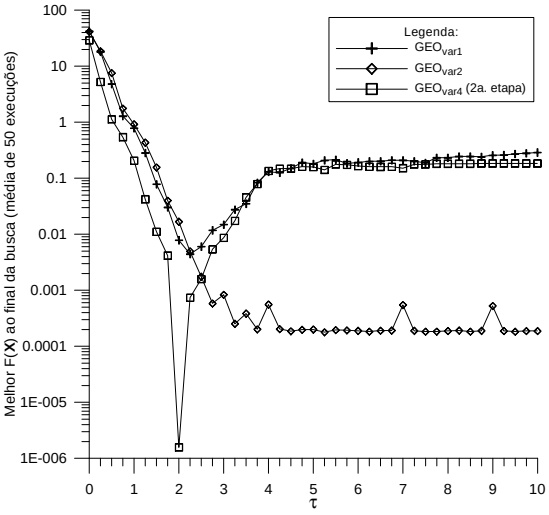


Figura 3.60b - FT_4 x τ x SA para GEO_{var}

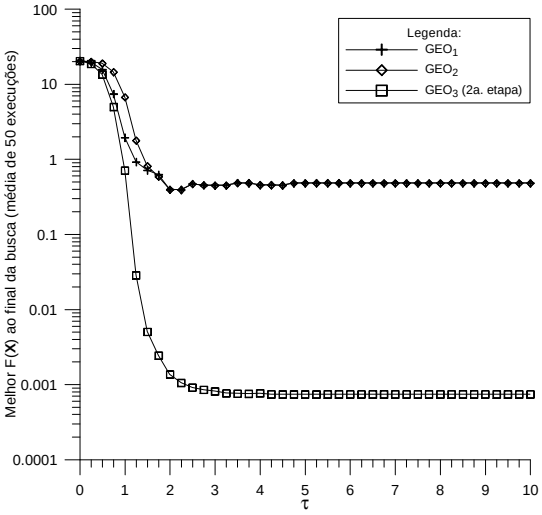


Figura 3.61a - FT_5 x τ x SA para GEO

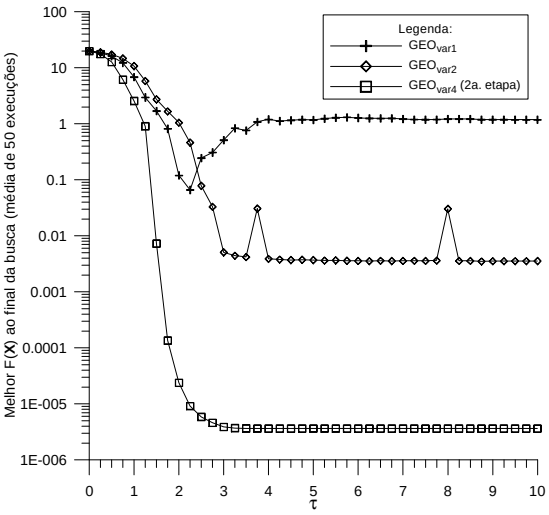


Figura 3.61b - FT_5 x τ x SA para GEO_{var}

É possível perceber, com o auxílio das Figuras 3.57 a 3.61, que a SA4 obteve ampla vantagem relativamente as SA's 1 e 2. As únicas exceções foram a FT_1, onde houve equilíbrio entre SA4 e SA2 para ambos GEO e GEO_{var}, e a FT_4 com o GEO apenas, onde a SA2 obteve melhor desempenho. O placar resumido é de 7 para a SA4 contra 1 da SA2. Este placar tão favorável obtido pela SA4 é uma evidência bastante forte de que esta versão deve ocasionar ganhos significativos de desempenho quando de sua aplicação a outras funções objetivo.

3.8 – Conclusões

Neste Capítulo, a versão original do algoritmo GEO foi apresentada e estudada, dando origem a quatro sugestões de aprimoramento (SA's de 1 a 4). Os testes de desempenho efetuados mostraram que três SAs ocasionaram melhora no desempenho do

GEO (SA1, SA2 e SA4), enquanto uma SA (SA3) ocasionou piora no desempenho. Das três SA's que melhoraram o desempenho do GEO, duas ocasionaram ganhos significativos de desempenho (SA2 e SA4), e uma ocasionou ganhos pouco significativos (SA1). Das duas SA's cujos ganhos de desempenho foram mais significativos, uma introduziu no GEO a capacidade de auto-reinicialização da busca (SA2) e a outra (SA4) introduziu a representação real das variáveis de projeto, com base ajustável, e também, a mudança no esquema de aplicação das mutações, antes implicitamente definido pelo sistema binário de representação das variáveis de projeto. Dentre as duas SA's, a SA4 obteve os ganhos de desempenho mais significativos. A Tabela 3.7 a seguir apresenta uma visão geral das SA's.

Tabela 3.7 – Avaliação geral qualitativa das SA's

Versão	Características Principais	Influência no desempenho	Magnitude da influência[1]
SA1	•Retenção da melhor solução encontrada ao longo da busca.	Positiva	Pequena
SA2	•Auto-reinicialização da busca em caso de estagnação.	Positiva	Média
SA3	•Distribuição mais uniforme das mutações no espaço de busca utilizando, a cada iteração, um esquema combinatório (multiordenamentos).	Negativa	Média
SA4	•Representação real das variáveis de projeto; •Base do sistema de representação é parâmetro do algoritmo; •Esquema alternativo de distribuição das mutações	Positiva	Grande

[1] As magnitudes pequena, média e grande foram definidas de forma aproximada, tendo como referência as variações de desempenho ocasionadas pelas SA's nos testes efetuados. Variações de até uma ordem de grandeza nos resultados numéricos são consideradas pequenas. Variações de até três ordens de grandeza são consideradas médias e variações acima de três ordens de grandeza são consideradas grandes.

CAPÍTULO 4

OTIMIZAÇÃO PARALELA

4.1 - Introdução

Este Capítulo apresenta uma implementação paralela do algoritmo Otimização Extrema Generalizada (*Generalized Extremal Optimization* - GEO), e analisa seu desempenho computacional quando aplicada a uma função teste comumente utilizada para avaliar algoritmos de otimização, a função de Rastringin (Ballester e Carter, 2004). A versão paralela implementada (Galski et al., 2004b), denominada GEOPar-1, foi escrita em Fortran com chamadas à biblioteca de comunicação por troca de mensagens *Message Passing Interface* (MPI). A fim de testar o desempenho dessa versão paralela, foi incluída na função teste uma carga de processamento simulada, de maneira a reproduzir o tempo de execução de aplicações práticas. Os resultados de desempenho foram obtidos utilizando uma máquina paralela de memória distribuída composta de 6 nós biprocessadores interligados por uma rede Gigabit Ethernet. São apresentadas curvas de desempenho em função do número de processadores para várias configurações da função teste, as quais incluem cargas simuladas diversas, bem como diferentes números de bits para a codificação das variáveis. A motivação principal para o desenvolvimento do GEOPar-1 é a de estender seu uso a aplicações práticas de projeto ótimo onde o custo computacional de cada avaliação da função objetivo é expressivo, o que torna a otimização indesejável numa máquina monoprocessada face ao alto número de avaliações exigido em métodos estocásticos como o GEO. Dentro deste contexto, o objetivo deste Capítulo é expor as potencialidades do GEOPar-1, almejando-se antever, através de testes simulados, qual será seu desempenho na execução em paralelo, quando aplicado a problemas reais de otimização da ciência e da engenharia.

A computação paralela tenta explorar o paralelismo intrínseco de tarefas presente nos algoritmos, por meio da execução simultânea destas tarefas em processadores diferentes. A motivação é a redução do tempo de execução. Outra razão para o uso de máquinas paralelas é o de viabilizar a execução de tarefas que,

pelos seus requisitos de memória, não poderiam ser executadas em um computador apenas. Atualmente, a área de computação paralela trabalha também na geração de algoritmos alternativos, que cumpram as mesmas finalidades dos tradicionais/originais, mas que permitam explorar ao máximo o paralelismo de tarefas.

A otimização paralela pode ser considerada uma área da computação paralela que visa primordialmente à redução do tempo de execução de algoritmos de otimização por meio de sua adaptação ou modificação para uso em máquinas paralelas. Ao longo do texto, os termos computador paralelo e máquina paralela são utilizados como sinônimos.

O restante deste Capítulo está organizado como segue: a seção 4.2 apresenta uma classificação dos computadores paralelos; a seção 4.3 descreve a infra-estrutura de hardware e software disponível atualmente na maioria dos centros de pesquisa; a seção 4.4 trata da eficiência nos computadores paralelos; a seção 4.5 descreve os meios de aplicarem-se técnicas paralelas aos problemas de otimização não linear; a seção 4.6 faz um breve resumo da computação paralela com Algoritmos Evolutivos; a seção 4.7 introduz GEOPar-1, a versão paralela desenvolvida a partir do algoritmo GEO; na seção 4.8, define-se a função teste com tempo de avaliação ajustável, a fim de simular aplicações de menor ou maior carga computacional e analisar sua influência no comportamento do algoritmo paralelo; na seção 4.9, apresentam-se e são analisados os resultados obtidos da aplicação do GEOPar-1 à função teste definida na seção anterior e, finalmente, na seção 4.10, estão as conclusões deste Capítulo.

4.2 - Classificação

Quanto ao número de processadores diferentes utilizados, os computadores paralelos podem ser classificados em homogêneos, se um único tipo de processador for usado, e heterogêneos, se mais de um tipo de processador fizer parte do computador paralelo.

Quanto à arquitetura de memória, a paralelização pode ser conseguida por meio da utilização de máquinas paralelas de arquitetura de memória distribuída ou de máquinas paralelas de arquitetura de memória compartilhada. Máquinas paralelas de arquitetura de memória distribuída, também chamadas multicomputadores, são aquelas nas quais cada processador possui seu próprio espaço de endereçamento de memória e o acesso aos espaços dos demais processadores é feito por troca de mensagens numa rede de interconexão (comunicação). As máquinas de arquitetura de memória compartilhada, também chamadas multiprocessadores, são aquelas nas quais todos os processadores têm acesso direto, por meio de operações de leitura e escrita, a um mesmo espaço de endereçamento de memória (Preto e Mendes, 1999b; Preto e Stephany, 2002; Alba e Tomassini; 2002; Talbi, 2002; Barney, 2006). Segundo Talbi (2002), a simplicidade de implementação do programa paralelo é uma das vantagens dos computadores paralelos com arquitetura compartilhada, mas, por outro lado, computadores paralelos de arquitetura distribuída oferecem uma plataforma de programação mais flexível e tolerante à falhas. Já Barney (2006) destaca a falta de escalabilidade entre a memória e os processadores como a principal desvantagem dos sistemas de memória compartilhada, onde a adição de processadores pode aumentar geometricamente o tráfego processador-memória. O mesmo autor coloca como primeira desvantagem dos sistemas de memória distribuída o aumento do trabalho do programador, que se torna responsável por muitos dos detalhes associados com a comunicação de dados entre processadores.

Arquiteturas para computadores paralelos podem ainda ser classificadas segundo a taxonomia clássica de Flynn, que se baseia na quantificação dos fluxos de instruções e dos fluxos de dados. Das quatro combinações possíveis, duas se destacam: SIMD – *Single Instruction stream, Multiple Data stream* e MIMD – *Multiple Instruction stream, Multiple Data stream* (Flynn, 1966, 1972; Preto e Stephany, 2002; Alba e Tomassini; 2002; Barney, 2006). A arquitetura SIMD típica emprega uma unidade central de controle, que faz a difusão de uma única instrução por vez. Todos os processadores do computador paralelo restringem-se a executar a mesma instrução, mas atuam em porções distintas dos dados a serem processados. Porções estas localizadas nas respectivas memórias locais de cada processador. Computadores

baseados na arquitetura SIMD exploram o paralelismo espacial eventualmente presente num problema. São muito eficientes quando executam algoritmos paralelos sincronizados que contêm conjuntos de dados que são homogêneos, uniformemente distribuídos (tanto do ponto de vista da quantidade quanto do ponto de vista do tempo de processamento de cada um) e que requerem transferências de dados também homogêneas. Quando o processamento ou as transferências de dados são irregulares ou assíncronas, arquiteturas SIMD tornam-se muito menos eficientes. Em arquiteturas paralelas MIMD, cada processador tem suas próprias instruções e opera em seus dados. Os processadores podem executar tipos diferentes de instruções e em tipos diferentes de dados do problema, permitindo, assim, alcançar uma maior eficiência em aplicações específicas, devido à sua maior flexibilidade na alocação das tarefas aos processadores. A contrapartida é o esforço adicional necessário na elaboração do programa paralelo, que, normalmente, torna-se mais complexo. Parte desta complexidade decorre diretamente das necessidades de comunicação e sincronização entre os processadores que são naturalmente mais complicadas nessa arquitetura. De acordo com Alba e Tomassini (2002), a arquitetura MIMD é geralmente a arquitetura mais útil e muitos sistemas comerciais paralelos pertencem à mesma.

Quanto à forma de gerenciar a alocação de tarefas, um programa paralelo pode ser classificado como sendo de alocação estática, de alocação dinâmica ou de alocação adaptativa (Talbi, 2002).

- Alocação estática: O número de tarefas da aplicação e a localização de cada uma são estabelecidos *a priori*, em tempo de compilação. A alocação processador-tarefa-dados permanece inalterada durante a execução, independentemente do estado corrente do computador paralelo. Quando existir diferença significativa de carga ou desempenho entre processadores, a alocação estática fará com que o tempo de execução da aplicação fique atrelado ao maior tempo de execução, dentre todos os processadores (presumivelmente aquele do processador com maior carga ou do processador com menor desempenho).

- Alocação dinâmica: O número de tarefas continua sendo fixado *a priori*, em tempo de execução. No entanto, a localização de cada uma é dinâmica, sendo estabelecida e modificada em tempo de execução.

- Alocação adaptativa: Tanto o conjunto de tarefas quanto a localização são estabelecidas *a posteriori*, em tempo de execução. Tarefas podem ser criadas ou eliminadas em função do nível de carga do computador paralelo.

4.3 – Infra-estrutura

No que diz respeito à infra-estrutura de software disponível para desenvolvimento de programas paralelos, vale destacar (entre outras) a existência de extensões às linguagens de programação usuais e que são de domínio público, tais como OpenMP (*OpenMP Architecture Review Board*, 2004; Burkardt, 2004; Chandra et al., 2001), para uso em máquinas paralelas de arquitetura compartilhada e *Message Passing Interface* - MPI (*MPI* Forum, 2004; Preto e Mendes, 1999c; Gropp e Lusk, 1997; Pacheco, 1997; Snir et al., 1995; Gropp et al., 1994), para uso em máquinas de arquitetura distribuída. As mesmas facilitaram em muito o desenvolvimento de programas paralelos. OpenMP e MPI podem ser considerados como modelos ou padrões de programação paralela.

O padrão OpenMP constitui-se de um conjunto de diretivas de compilação e biblioteca de rotinas que são usadas para expressar paralelismo em arquiteturas de memória compartilhada (disponível em C++ e Fortran). A maior parte do padrão OpenMP são as diretivas de compilação que, adicionadas a um programa seqüencial existente, avisam ao compilador quais partes do programa devem ser executadas em paralelo e quais são os pontos de sincronização.

O padrão MPI constitui-se de uma biblioteca de rotinas para troca de mensagens entre computadores interconectados. Existem rotinas disponíveis para permitir comunicação entre processos, comunicações para grupos, estabelecimento, configuração e gerenciamento destes grupos, além de interação com o restante do sistema computacional. De acordo com Alba e Tomassini (2002), o padrão MPI é atualmente a melhor escolha para

a implementação de Algoritmos Evolutivos Paralelos (em inglês: *Parallel Evolutionary Algorithms – PEAs*) de larga escala e de uso geral.

Quanto à infra-estrutura de hardware, o baixo custo dos microcomputadores PC e o crescente poder computacional dos mesmos está propiciando a popularização de computadores paralelos que são montados usando arquitetura de memória distribuída, onde cada nó da máquina paralela é, essencialmente, um microcomputador PC constituído apenas dos componentes necessários à sua função dedicada: processador, memória RAM, disco rígido e interface de comunicação em rede. Tais computadores paralelos têm como principal vantagem o fato de seu custo escalonar quase linearmente com o número de processadores utilizados. A contrapartida é a limitação no número total de processadores, que não deve ultrapassar algumas dezenas, de modo que a interface de comunicação não degrade excessivamente o desempenho do conjunto. Ainda assim, em diversas aplicações, o ganho em tempo de processamento é também quase linear. Deste modo, nada mais natural para instituições de pesquisa com carência de recursos do que investir neste tipo de máquinas paralelas, como constatado pelo número cada vez maior de "*clusters*" de memória distribuída sendo montados nas instituições de pesquisa brasileiras.

Mesmo instituições pertencentes aos países mais desenvolvidos, nas quais a disponibilidade de recursos é maior, estão montando computadores paralelos de arquitetura de memória distribuída e com processadores do tipo PC, ou seja, processadores destinados ao usuário doméstico. A rede de comunicação utilizada, porém, geralmente é de alto desempenho (e alto custo), permitindo assim, a utilização de centenas e até milhares de processadores. Os resultados, quando comparados com as soluções “proprietárias” dos gigantes do setor, são surpreendentemente positivos. Num estudo recente de desempenho de computadores (Dongarra, 2003) numa aplicação específica, a solução de sistemas de equações lineares (usando LINPACK em Fortran), de uma listagem de cerca de mil computadores paralelos, entre os 50 mais velozes, aparecem quatro computadores que foram construídos segundo a abordagem exposta. O melhor deles, figura em vigésimo terceiro lugar na lista, obtendo 2207Gflop/s com 1024 processadores Intel Pentium 4 de 1,8GHz. Um feito notável, sem dúvida.

4.4 - Eficiência

Um outro aspecto muito importante nos computadores paralelos é a eficiência, ou seja, a utilização efetiva do poder de cálculo disponível em cada um dos processadores. O mesmo estudo já citado (Dongarra, 2003), apresenta uma tabela mostrando a eficiência de dezenas de computadores paralelos diferentes e para diversas configurações (número de processadores em uso). O tempo uniprocessador também é mostrado, pois serve como base de cálculo da eficiência (eficiência = Δt_{1P} / (Δt_{NP} . NP), onde NP = número de processadores, Δt_{1P} = tempo de execução com um processador, Δt_{NP} = tempo de execução com NP processadores; (Preto e Mendes, 1999a)). Uma compilação dos dados da referida tabela é apresentada a seguir, também de forma tabular (vide Tabela 4.1), onde a eficiência média é calculada e mostrada para diversos números de processadores. Entre parênteses, estão os computadores utilizados no cálculo da média e suas respectivas eficiências. O fato mais importante que pode ser observado na Tabela 4.1 é que a eficiência cai significativamente com o número de processadores. De 0,932 com 2 processadores para 0,12 com 1024 processadores. Eficiência de 0,12 significa um poder de cálculo de apenas 0,12 x 1024 ≅ 123 processadores. Ou seja, 1024 processadores estão sendo utilizados para gerar um desempenho que seria possível com apenas 123 processadores, se fosse possível manter a eficiência igual a um. Outra forma de enxergar este fato é pensar que o poder de cálculo de 1024-123 = 901 processadores não está sendo utilizado. Mesmo com apenas 32 processadores, a Tabela 4.1 mostra que menos de 0,4 do poder de cálculo disponível é utilizado. Alguém poderia argumentar, com razão, que esta é uma análise circunstancial, vinculada a uma única aplicação, qual seja, a solução de sistemas de equações lineares. No entanto, esta aplicação é extensivamente utilizada em várias áreas e, em particular, como parte intrínseca de métodos de otimização de primeira e segunda ordem (Migdalas et al., 2003). Desta forma, a análise recém efetuada é importante e pertinente à computação paralela em problemas de otimização. De fato, na revisão feita em Migdalas et al. (2003), é destacado que a área de computação paralela em otimização avança em duas frentes: uma tenta paralelizar os pacotes de álgebra linear existentes, como forma de alavancar o desempenho dos métodos de otimização que deles dependem. A outra frente, se esforça em descobrir novas estratégias de otimização global que sejam naturalmente eficientes do ponto de vista da paralelização.

Tabela 4.1 – Eficiência média de computadores paralelos na solução de equações lineares.

Número de processadores	Eficiência média
2	0,932 (Cray C90=0,94; IBM ES/9000=0,91; Cray Y-MP/8=0,98; Cray Y-MP/98=0,97; Convex C3820=0,98; Convex SPP-1000=1,03; Meiko CS2=0,87; Fujitsu AP1000=0,98; Intel Delta=0,95; Alliant FX/2800-200=0,97; IBM PVS=0,96; IBM RS/6000 Cluster=0,80; nCUBE 2=0,97; Intel iPSC860=0,84; Meiko CS=0,94; Suprenum S1C1=0,75; Alliant FX/80=0,97; SGI 4D/420=0,94; SGI 4D/320=0,95)
8	0,720 (Cray C90=0,86; IBM ES/9000=0,67; Cray Y-MP/8=0,87; Cray Y-MP/98=0,71; Convex C3880=0,88; Convex SPP-1000=0,77; Meiko CS2=0,54; Fujitsu AP1000=0,89; Kendall SR=0,59; Intel Delta=0,67; SunSparc2000= 0,99; Alliant FX/2800-200=0,88; IBM PVS=0,74; IBM RS/6000 Cluster=0,37; nCUBE 2=0,88; Intel iPSC860=0,41; Meiko CS=0,60; Suprenum S1C1=0,61; Alliant FX/80=0,75; SGI 4D/480=0,71; SGI 4D/380=0,73)
32	0,396 (Meiko CS2=0,21;Fujitsu AP1000=0,75, Kendall SR=0,52; Intel Delta=0,31; IBM PVS=0,29; nCUBE 2=0,71; Intel iPSC860=0,17; Meiko CS=0,21)
128	0,357 (Fujitsu AP1000=0,52; Intel Delta=0,10; nCUBE 2=0,45)
512	0,173 (Fujitsu AP1000=0,29; Intel Delta=0,03; nCUBE 2=0,20)
1024	0,12 (nCUBE 2)

4.5 – Computação paralela e otimização não linear

De acordo com Schnabel (1995), trazer a computação paralela para a otimização não linear pode ser feito de três formas:

1) Paralelização das avaliações da função objetivo e/ou de avaliações de derivadas no algoritmo;

2) Paralelização dos pacotes (*kernels*) de álgebra linear;

3) Modificação de algoritmos básicos, que aumentam o grau de paralelismo intrínseco, por exemplo, perfazendo múltiplas avaliações da função objetivo ou derivada(s).

Segundo Migdalas et al. (2003), as formas 1 e 2 já vêm sendo atacadas há duas décadas, tendo produzido algoritmos altamente sofisticados, enquanto a forma 3 tem sido alvo de estudos com maior freqüência em anos recentes (Bordfeldt et al., 2003; Gómez et al., 2003; Ralphs, 2003; Wang et al., 2002; Wang e Damodaran, 2002; Wang e Damodaran, 2001; Dennis e Wu, 2000; Toulouse et al.; 2000; Jones et al., 2000; Meza e Martinez, 1994; Dennis e Torczon, 1991).

Dentro deste contexto, o desenvolvimento de algoritmos paralelos eficientes para otimização e que possam ser utilizados em praticamente qualquer tipo de problema é de enorme interesse. Tendo esta idéia em mente, esta Tese de Doutorado pretende ampliar o horizonte de aplicabilidade do GEO, ao criar uma versão paralela do mesmo que mantenha, tanto quanto for possível, a característica de universalidade de aplicação e com escalonamento linear ou quase linear com o número de processadores. Haja vista que o GEO foi desenvolvido para ser um otimizador global e não necessita fazer uso de pacotes de álgebra linear, nada mais natural do que usá-lo para atacar a terceira forma de paralelização descrita por Schnabel, que, como visto, busca descobrir diretamente novas estratégias de otimização global paralelizada.

4.6 – Computação paralela e algoritmos evolutivos

Algoritmos Evolutivos Paralelos, aqui chamados AEPs, são uma área de pesquisa que tem recebido interesse considerável, manifestado pelo grande número de implementações e modelos de AEPs desenvolvidos (Alba e Tomassini, 2002; Migdalas et al., 2003; Nowostawski e Poli, 1999). Parte deste interesse pode ser creditado ao fato dos Algoritmos Evolutivos (AEs) (Eiben e Smith, 2003) serem naturalmente propensos à paralelização, visto que lidam com populações de indivíduos onde muitas das operações podem ser realizadas em paralelo. Uma outra justificativa possível para tal interesse decorre do fato de que os AEs, assim como a maioria dos métodos estocásticos, geralmente necessitam de um número muito grande de avaliações da função objetivo para obterem uma solução ótima ou quase ótima. O terceiro fator a ser considerado é a universalidade de

aplicação dos AEs aos problemas de otimização em geral. A influência conjunta destes fatores transforma a paralelização dos AEs numa resposta ou solução lógica ao panorama exposto. Por um lado, os AEs são grandemente paralelizáveis. De outro lado, após paralelizados, quando aplicados a problemas temporalmente custosos, obtêm soluções equivalentes àquelas dos AEs seqüenciais em uma fração do tempo. Considerando ainda que as versões paralelas dos AEs são aplicáveis à maioria dos problemas de otimização, o interesse que os AEs paralelos têm recebido se justifica.

Quanto à topologia de sua população total, um AEP pode ser classificado como possuindo uma população:

- global (população única e esquema mestre-escravo): O operador seleção (de indivíduos) é global e existe a possibilidade de qualquer indivíduo poder cruzar com qualquer outro. Este é o modelo mais simples de AEP, onde as avaliações de adaptação dos indivíduos (ou seja, a avaliação de F(X)) são distribuídas entre todos os processadores do computador paralelo e calculadas concorrentemente. O processador mestre aplica os operadores seleção, cruzamento e mutação. Por conta disso, neste tipo de AEP, o comportamento do AEP e do seu AE correspondente são idênticos, em termos das soluções encontradas. Apenas o desempenho temporal é afetado.

- distribuída ou ilha: Nesse caso, a população original do AE seqüencial é dividida em várias subpopulações. A mesma versão do AE é aplicada (executada) separadamente e em paralelo a cada subpopulação. Esse modelo é conhecido como "ilha", pois cada subpopulação representa uma ilha onde o processo evolutivo ocorre independentemente das demais ilhas. A migração surge, então, como um novo tipo de operador, que passa a existir no AEP resultante. Após um período determinado, geralmente especificado em número de gerações, ocorre a migração, ou seja, a troca de indivíduos entre as ilhas. Em geral, o número de indivíduos que migram é baixo, caracterizando uma troca esparsa de indivíduos (material genético na analogia dos AEs) e justificando, assim, o termo ilha.

- celular ou difusa: Nesse caso, a população original do AE seqüencial é dividida em um número elevado de subpopulações, chamadas células. Além disso, em uma célula, cada indivíduo possui seu próprio conjunto de vizinhos, com o qual pode

cruzar. Ocorre a sobreposição de pequenas vizinhanças, o que ajuda na exploração do espaço de busca (Alba e Tomassini, 2002). Em geral, os vizinhos de cada indivíduo são definidos segundo uma distribuição unidimensional (1-D) ou bidimensional (2-D).

- híbrida: Ao menos dois dos três tipos de topologia definidos anteriormente são aplicados hierarquicamente na divisão da população do AE original. Assim, criam-se subpopulações de subpopulações do AE original, cada uma sendo de um dos três tipos básicos (global, ilha, celular). Assim, por exemplo, é possível criar uma topologia híbrida ilha-celular, onde o primeiro nível hierárquico compõe-se de ilhas sujeitas a migrações esparsas e onde cada ilha tem sua respectiva subpopulação dividida em inúmeras células, caracterizando o segundo nível hierárquico da topologia híbrida resultante. Um segundo exemplo pode ser dado por uma topologia ilha-global, onde novamente, o primeiro nível hierárquico compõe-se de ilhas sujeitas a migrações esparsas. O segundo nível hierárquico, entretanto, utiliza o modelo global. Portanto, cada ilha tem sua respectiva subpopulação dividida segundo o esquema mestre-escravo. Assim, neste segundo exemplo, cada ilha possui um processador mestre que distribui as avaliações dos indivíduos entre os vários processadores escravos pertencentes à respectiva ilha.

Quanto à uniformidade dos indivíduos (sua representação) existentes, bem como quanto à uniformidade do AE aplicado em cada subpopulação, os AEPs podem ser classificados em uniformes e não uniformes. Nos AEPs uniformes, é utilizada sempre a mesma representação e os AEs atuando em cada subpopulação são absolutamente idênticos. Nos AEPs não uniformes, tipos diferentes de representação e diferentes versões de AEs são aplicados em cada subpopulação.

4.7 – GEOPAR-1

A versão paralela implementada, denominada GEOPar-1, foi escrita em Fortran com chamadas à biblioteca de comunicação por troca de mensagens MPI (*Message Passing Interface*). A escolha do padrão MPI para a implementação ocorreu devido aos seguintes fatores:

1) MPI permite maior escalonabilidade, em termos do número de processadores utilizado.
2) existência de *clusters* de arquitetura distribuída no INPE, onde o padrão MPI é utilizado;

Em Galski et al. (2003), uma versão aprimorada do GEO foi exposta e testada, obtendo bons resultados. Este artigo apresenta o GEOPar-1, a versão paralela desta última. A função teste utiliza a função de Rastringin, que foi selecionada dentre funções teste não lineares utilizadas em trabalhos anteriores sobre o GEO, tendo as mesmas sido utilizadas para avaliação de AGs (Potter e De Jong, 1994) e também para avaliação do GEO original (Sousa e Ramos, 2002).

No GEO, as variáveis de projeto são codificadas em uma seqüência de bits, como exemplificado na Figura 4.1. A solução candidata inicial do GEO, gerada aleatoriamente, fornece uma seqüência de bits, associados às variáveis de projeto X_j. O conjunto dessas variáveis é codificado por L bits. Como se sabe (vide Capítulo 3), a cada iteração do GEO, através da função objetivo, a influência da alteração de cada um dos L bits da seqüência é mensurada individualmente. Existe, portanto, a necessidade de avaliar-se o valor de F(**X**) L vezes a cada iteração. Nada mais natural, do que dividir uniformemente as L avaliações de F(**X**) entre os NP processadores de uma máquina paralela, conforme ilustra a Figura 4.1.

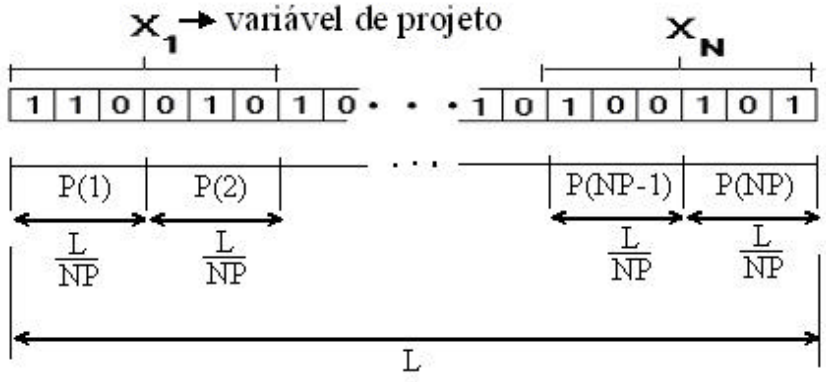


Figura 4.1 – Divisão da seqüência binária de L bits entre os NP processadores.

Explora-se, assim, o paralelismo inerente ao algoritmo dividindo-se as avaliações da seqüência binária alterada de um único bit entre os processadores seguindo um esquema mestre-escravo. O processador mestre envia aos demais processadores as seqüências a

serem avaliadas e, ao receber os resultados, executa a ordenação e a escolha do bit a ser alterado para a próxima iteração. Durante a execução, todos os processadores verificam o critério de parada estabelecido. Neste trabalho, o critério adotado foi o número máximo de avaliações da função objetivo.

A versão paralela foi escrita em Fortran e a paralelização foi obtida com chamadas à biblioteca de comunicação por troca de mensagens MPI (*Message Passing Interface*), para execução numa máquina paralela de memória distribuída, um *cluster*.

Eventualmente, a divisão do domínio entre processadores pode levar a um pequeno desbalanceamento de carga, pois o número de bits L pode não ser múltiplo do número de processadores. Uma consideração importante é que o esquema de paralelização adotado reproduz exatamente o mesmo processo evolutivo do GEO seqüencial, para qualquer número de avaliações de F(**X**). Portanto, exceto pelo desempenho temporal, as análises e comparações de desempenho já efetuadas para o GEO valem para o GEOPar-1. Essa foi a razão de se utilizar, neste trabalho, uma função teste baseada apenas na função de Rastringin, uma vez que objetiva-se a análise do desempenho do algoritmo paralelo. Esta função tem dimensionalidade ajustável, é não linear e multimodal.

Define-se granularidade de um algoritmo paralelo como a razão média entre o tempo de processamento e o de comunicação. Uma paralelização eficiente apresenta alta granularidade, isto é, granularidade grossa, além de apresentar um bom balanceamento de carga entre processadores. No caso de máquinas paralelas de memória distribuída, constituídas por nós monoprocessados ou multiprocessados, a rede de interconexão dos nós tem grande influência no tempo de comunicação. Os parâmetros relevantes são a latência e a largura de banda da rede. A latência da rede (LR) é o tempo gasto para se enviar uma mensagem de tamanho nulo entre dois nós, enquanto que a largura de banda (LB) é a taxa de transmissão da rede. O *cluster* utilizado neste trabalho tem rede padrão Gigabit Ethernet, com taxa nominal de 1 Gbit/s. O tempo gasto para o envio de uma mensagem de tamanho M entre dois nós pode ser calculado como:

$$\text{tempo de envio} = LR + M/LB \tag{4.1}$$

Assim, tempos de envio elevados podem, conforme o programa em execução, deixar processadores ociosos, à espera de uma comunicação ser completada, ocasionando baixa granularidade, ou seja, perda de desempenho.

4.8 – Função teste com tempo de avaliação escalonável

Para realizar a avaliação de desempenho proposta, optou-se por modificar o tempo de avaliação da função de Rastringin, capacitando a mesma a emular cargas de processamento de problemas reais. A modificação consistiu em introduzir na função, após o cálculo de seu valor algébrico (vide equação 4.2), um laço de repetição que faz chamadas a uma função de temporização e utiliza esta informação como condição para seu término. O laço só termina quando o tempo previamente estipulado é atingido. Foram definidos 3 patamares Ti, i={1,2,3}, de tempo de execução, conforme a Tabela 4.2, com base em exemplos abordados em trabalhos anteriores do GEO (Sousa et al., 2003a; 2003b; 2004a; 2004b; Galski et al., 2004a). Simula-se, assim, do ponto de vista da carga de processamento, 3 classes de problemas de otimização. Cabe ressaltar que T1 foi definido mais para testar os limites de GEOPar-1 (pela diminuição da granularidade) do que para simular uma aplicação custosa.

A função objetivo F(**X**) é dada pela função de Rastringin, a qual é definida por:

$$F(\mathbf{X}) = 3{,}0N + \sum_{i=1}^{N} \left(X_i^2 - 3{,}0\cos(2pX_i) \right) \qquad (4.2)$$
$$-5{,}12 \le X_i \ge 5{,}12 \quad N \in \{1,2,3,...\}$$

Esta função é escalonável dimensionalmente, por meio de N, o número de variáveis. A solução ótima (mínimo) é F(**X**)=0 para **X**=**0**. É válido informar que esta função, com N=20, também é usada no Capítulo 3 (vide Tabela 3.2), onde foi denominada FT_2. Com relação ao tamanho L da seqüência de bits, foram definidos 3 níveis de codificação, conforme a Tabela 4.3. Simula-se, assim, do ponto de vista da complexidade, a qual depende do tamanho da seqüência L de bits, 3 níveis deste problema de otimização. A complexidade não está unicamente relacionada a N, mas é diretamente proporcional a este no presente caso. Em todos os níveis, cada variável de projeto utiliza 16 bits (logo, N =

L/16). Os 9 casos de teste escolhidos estão definidos na Tabela 4.4, por meio das combinações possíveis {T,L}.

Tabela 4.2 - Patamares de tempo para a F(**X**)

Patamar	Tempo de execução, T (s)
T1	0,001
T2	0,1
T3	1,0

Tabela 4.3 - Níveis de codificação (L) e variáveis de projeto (N)

Nível	Número de bits da seqüência (L)	N
L1	16	1
L2	64	4
L3	640	40

Tabela 4.4 – Casos de teste adotados (combinações {T,L})

	L1	L2	L3
T1	{0,001; 16}	{0,001; 64}	{0,001; 640}
T2	{0,1; 16}	{0,1; 64}	{0,1; 640}
T3	{1,0; 16}	{1,0; 64}	{1,0; 640}

4.9 – Análise de desempenho

A fim de testar o desempenho do GEOPar-1, é necessário saber avaliar a influência dos diversos parâmetros do problema. Esses parâmetros foram escolhidos para os diversos casos de teste, conforme as tabelas apresentadas na seção anterior.

A cada iteração do GEOPar-1, considerando NP processadores e L sendo múltiplo de NP, cada processador efetua (L/NP) avaliações de F(**X**) e retorna uma mensagem com o resultado para o nó mestre. Portanto, o tamanho desta mensagem é diretamente proporcional à razão (L/NP).

Conseqüentemente, L influi, na Equação (4.1), no termo associado à largura de banda. O tempo T de avaliação de F(**X**), por sua vez, pode, em função do valor de LR, ocasionar ociosidade

de processamento, pois se o intervalo entre troca de mensagens for menor do que LR haverá tempo perdido por todos os nós simplesmente aguardando na fila para enviar sua mensagem. Além disso, como a razão L/NP está associada ao número de avaliações de F(**X**) realizadas por cada processador, L também pode ocasionar ociosidade de processamento devido à latência LR.

As combinações {T,L} ilustradas na Tabela 4.4 foram testadas com o GEOPar-1, sendo os tempos de execução apresentados nas Figuras 4.2 a 4.10, a seguir, para NP variando entre 1 e 12. Para os mesmos casos, a versão seqüencial original do GEO foi executada num nó do *cluster*, para servir de referência no cálculo da eficiência ε. O número de avaliações limite utilizado foi 2000, servindo de critério de parada. Para cada caso, três execuções independentes foram feitas utilizando sementes diferentes na geração dos números aleatórios. Em seguida, os tempos médios para cada semente foram calculados e utilizados para gerar as curvas em escala logarítmica (Figuras 4.2 a 4.10). O valor limite de 2000 avaliações foi escolhido de maneira a garantir um mínimo de 3 iterações do algoritmo, conforme exposto na Seção 4.7, quando utilizado o nível de codificação L3, em que cada iteração requer 640 avaliações. Quando os níveis L2 e L1 são utilizados, o número mínimo de iterações completas sobe para 30 e 120, respectivamente.

O critério de parada adotado, que impõe um limite no número de avaliações, é adequado, pois visa manter um custo computacional equivalente para os diferentes níveis de codificação em cada patamar Ti.

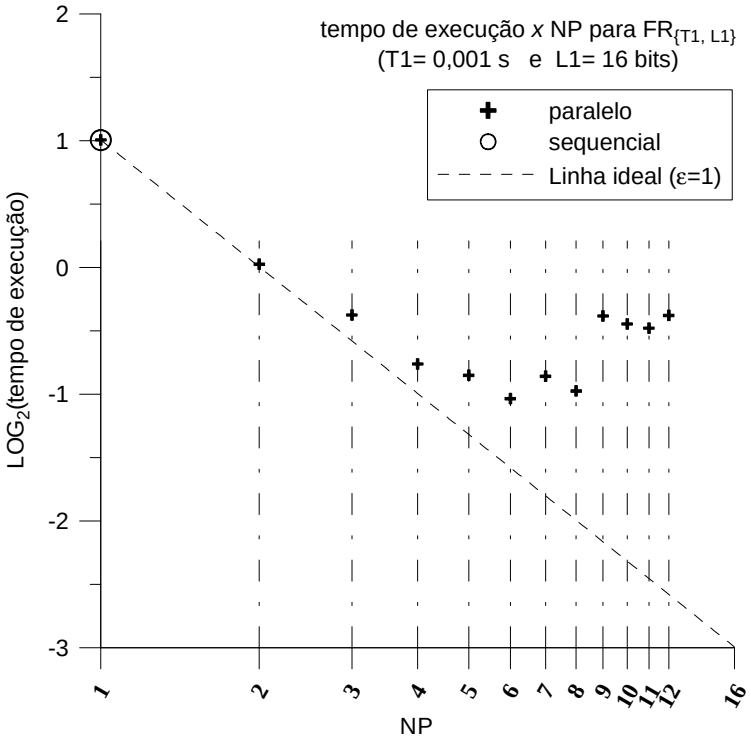


Figura 4.2 - Tempos de execução para {T1; L1}

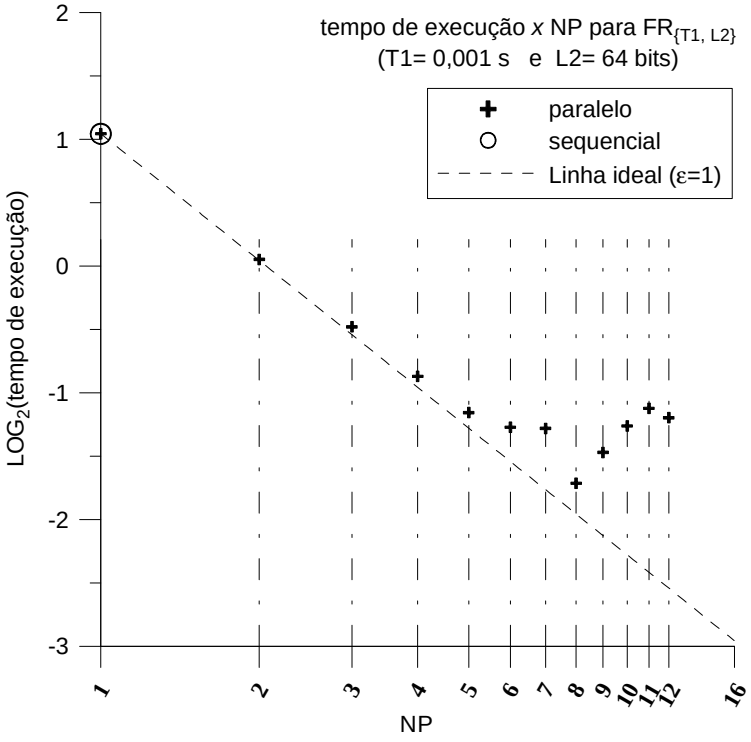


Figura 4.3 - Tempos de execução para {T1; L2}

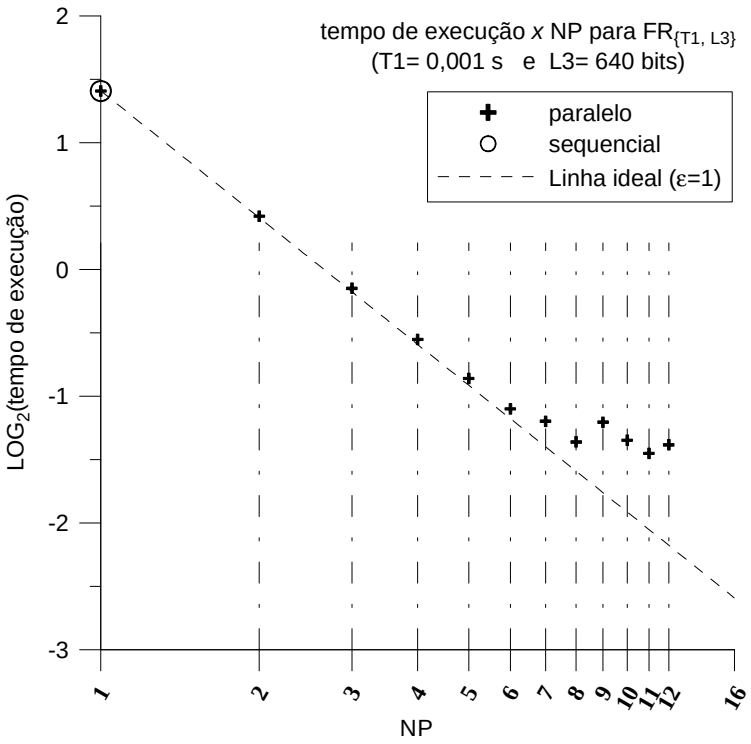


Figura 4.4 - Tempos de execução para {T1; L3}

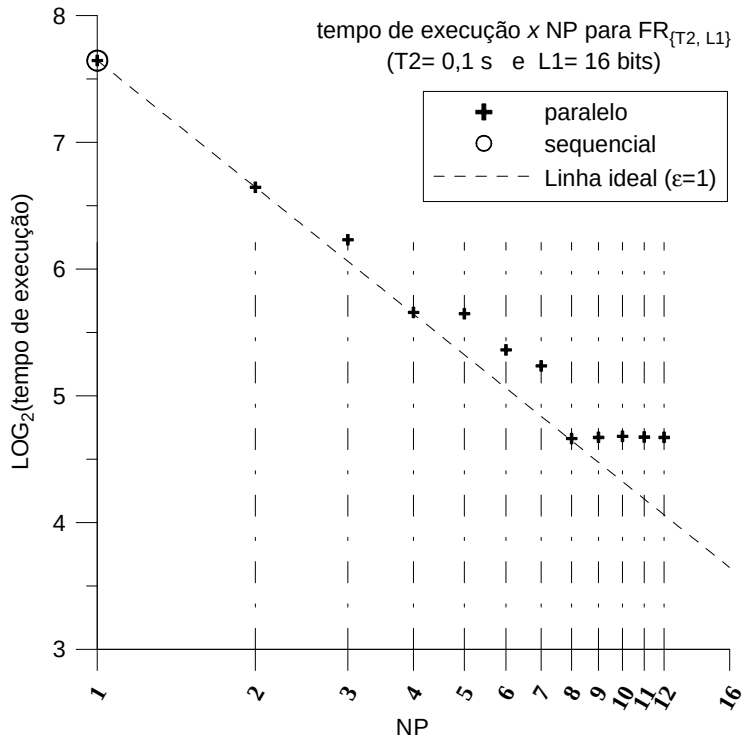


Figura 4.5 - Tempos de execução para {T2; L1}

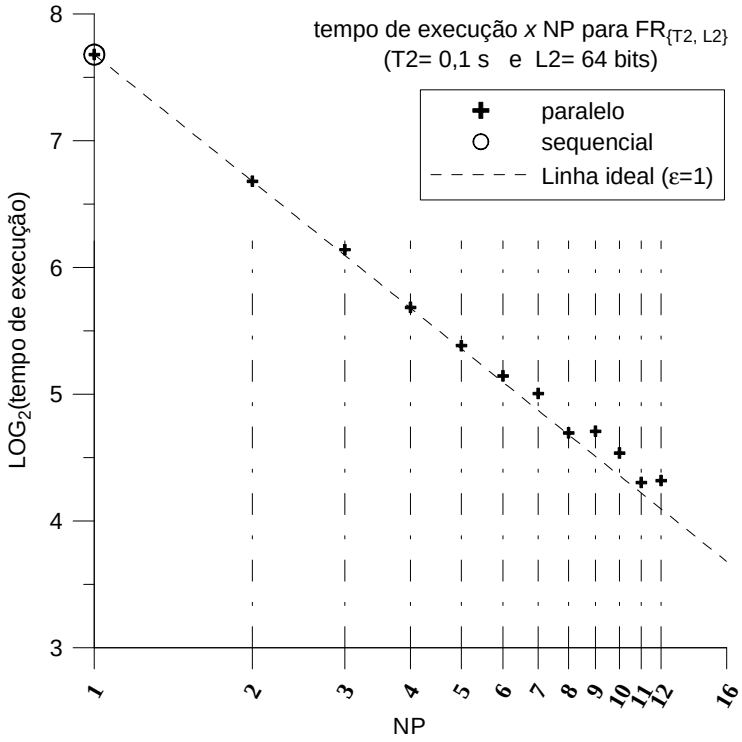


Figura 4.6 - Tempos de execução para {T2; L2}

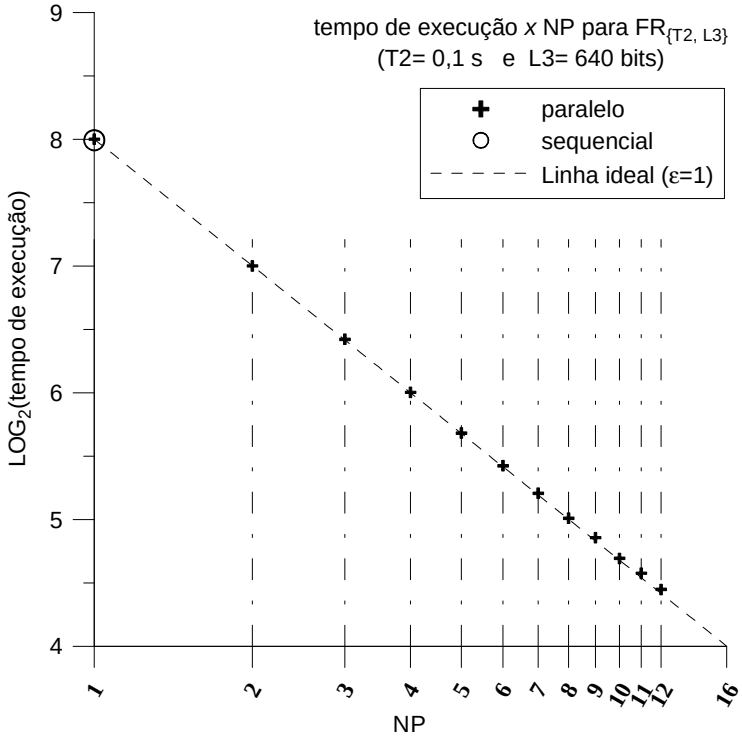


Figura 4.7 - Tempos de execução para {T2; L3}

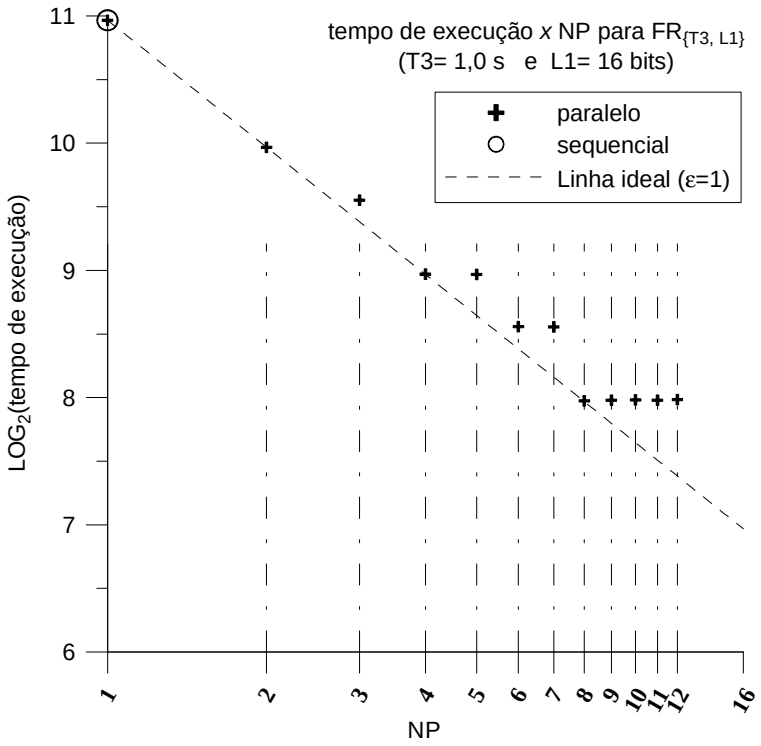


Figura 4.8 - Tempos de execução para {T3; L1}

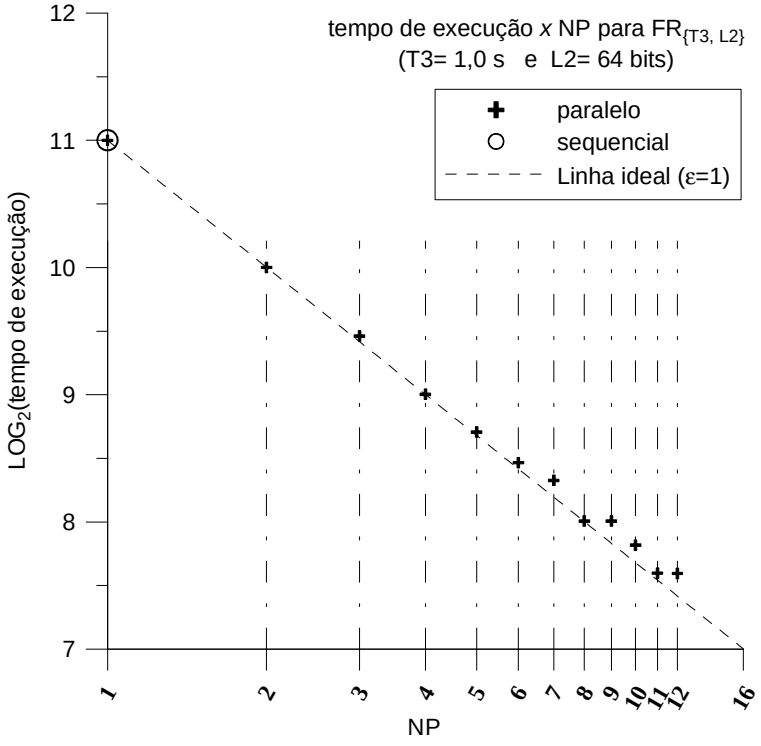


Figura 4.9 - Tempos de execução para {T3; L2}

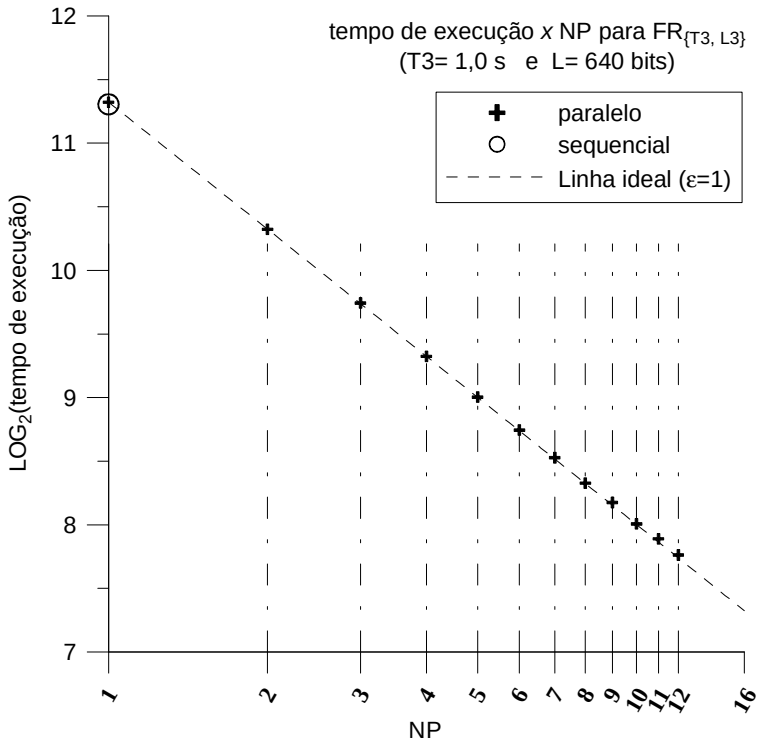


Figura 4.10 - Tempos de execução para {T3; L3}

O *speed-up* S(NP) de uma execução paralela é dado pela razão entre o tempo de execução da versão seqüencial e o tempo de execução da versão paralela com NP processadores:

$$S(NP) = t_{sequencial}/t_{paralelo} \quad (4.3)$$

A eficiência de uma execução paralela em função do *speed-up* é dada por:

$$\varepsilon(NP) = S(NP)/NP \quad (4.4)$$

Observando-se as Figuras 4.2 a 4.10, pode-se notar que: (i) na classe de problemas T1 (Figuras 4.2 a 4.4) há queda de eficiência com o aumento de NP, sendo esta queda mais significativa para valores menores de L; (ii) nas classes T2 (Figuras 4.5 a 4.7) e T3 (Figuras 4.8 a 4.10) há queda de eficiência com o aumento de NP para L1 e, em menor proporção, também para L2.

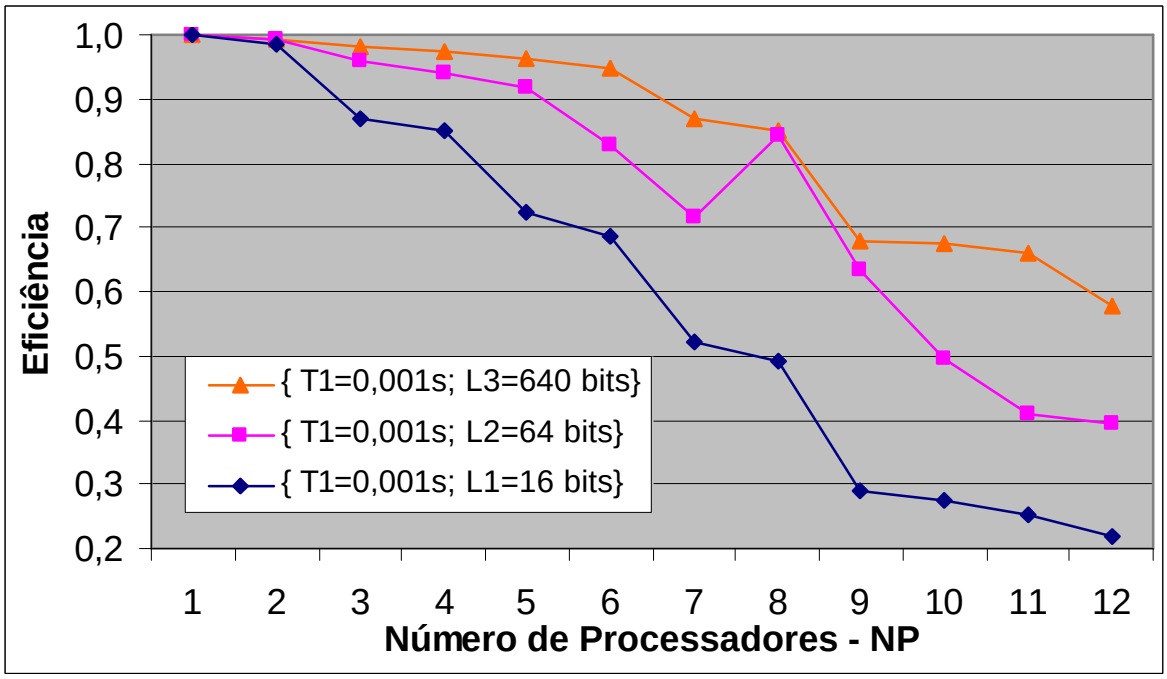


Figura 4.11 - Eficiência com T1

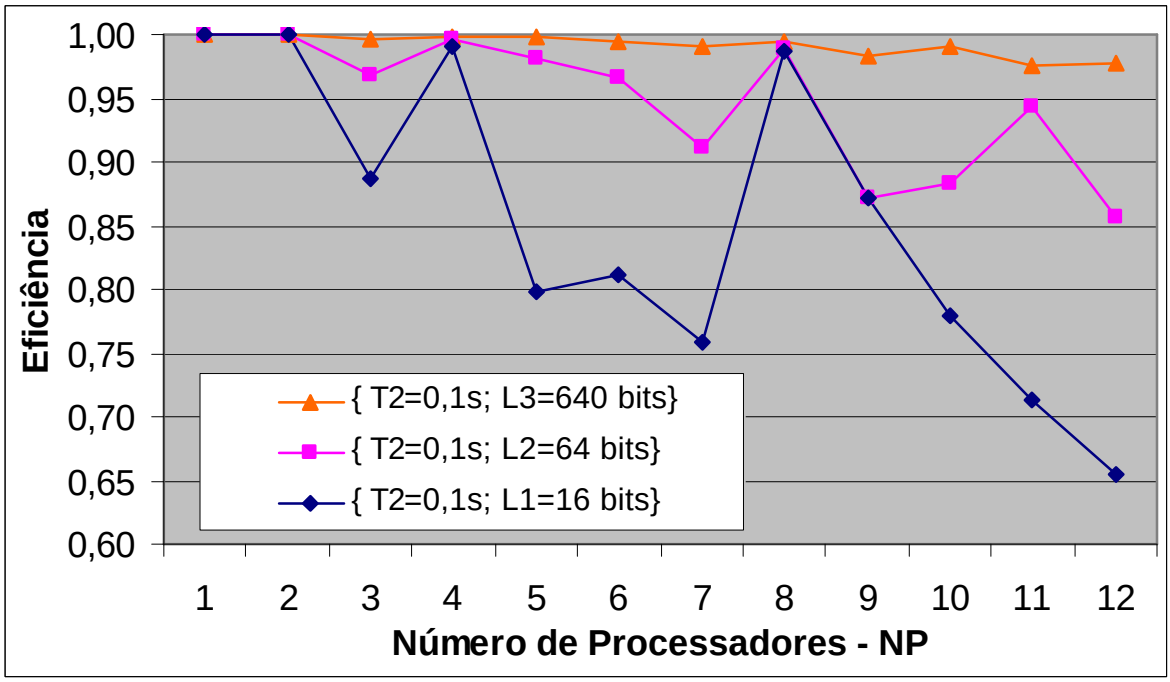


Figura 4.12 - Eficiência com T2

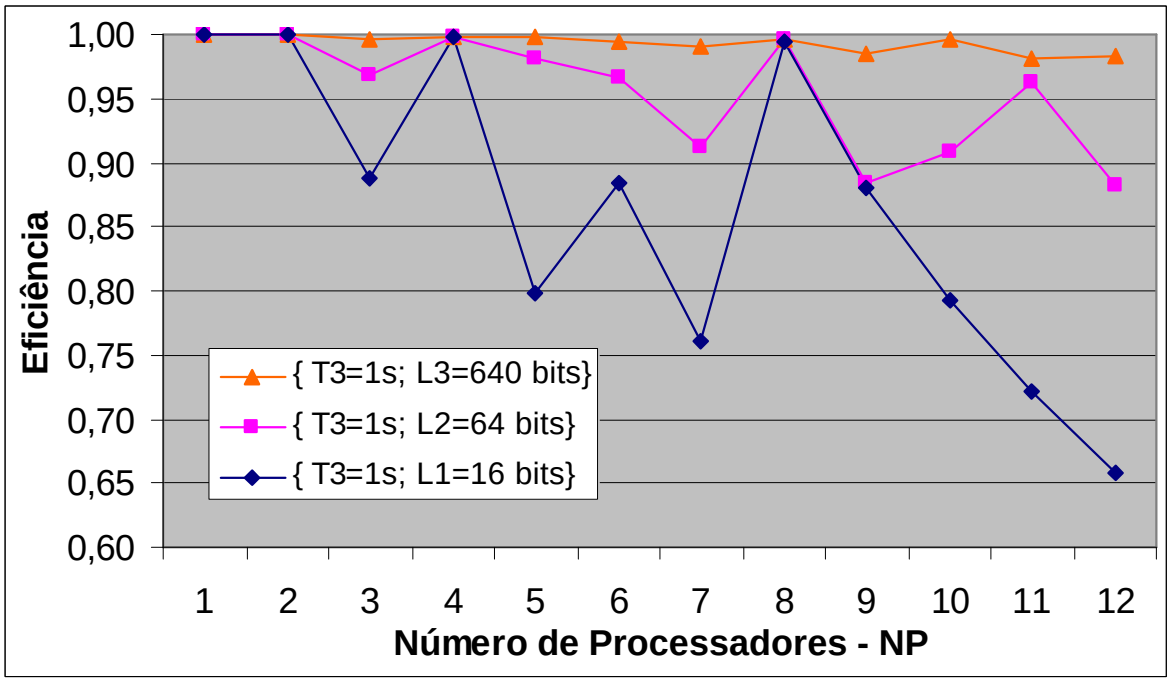


Figura 4.13 - Eficiência com T3

As Figuras 4.11, 4.12 e 4.13, apresentam as eficiências obtidas pelo GEOPar-1 para T1, T2 e T3, respectivamente. A análise dessas figuras mostra que, para T2 e T3, as eficiências são similares, apesar do tempo de processamento maior de T3. Isto significa que, nesses casos, a granularidade é satisfatória, isto é, o tempo de processamento é grande o suficiente em relação ao tempo de comunicação, resultando em boa eficiência, a menos do problema de desbalanceamento de carga, que tende a ser maior para valores menores de L. Observam-se nitidamente os picos de eficiência nas Figuras 4.12 e 4.13 quando o número de processadores é múltiplo de L (razão L/NP inteira) e, portanto, há balanceamento de carga. No caso T1, ilustrado na Figura 4.11, o tempo de avaliação é muito pequeno, implicando numa granularidade mais fina que penaliza a eficiência para maior número de processadores, uma vez que isso implica obviamente em mais comunicação. Em função disso, o efeito de desbalanceamento de carga observado em T2 e T3 fica visivelmente atenuado em T1.

Quando a razão L/NP é não inteira, a parte inteira e o resto são dados respectivamente por:

$$D = INT(L/NP) \tag{4.5}$$

e $R = MOD(L/NP)$ (4.6)

Isso implica que R processadores executam (D + 1) avaliações e os demais, D avaliações. Uma vez que o tempo de avaliação é uniforme, o desbalanceamento de carga pode ser quantificado: há (NP-R) processadores ociosos durante 1 avaliação. Uma eficiência ótima pressupõe balanceamento de carga entre processadores. Pode-se demonstrar que, desprezando custos de comunicação, a eficiência máxima teórica considerando o desbalanceamento acima exposto é dada por:

$\varepsilon_{max} = L/(NP.(D+1))$ (R?0) (4.7)

A Tabela 4.5, a seguir, compara os valores máximos teóricos de eficiência, obtidos da Equação 4.7, aos valores reais observados nos testes. Se, por um lado, a eficiência real tende a ser inferior à máxima teórica acima devido à comunicação, por outro lado, há otimização do acesso à memória cache, uma vez que, com o aumento de NP, a quantidade de memória exigida por processador tende a ser menor do que no algoritmo seqüencial. Isto diminui o tempo de processamento por processador, podendo-se obter eficiências maiores que as da Equação 4.7, conforme observado nos valores destacados em negrito na Tabela 4.5.

Tabela 4.5 – Valor teórico máximo e observado para ε

L	NP	ε (máx.)	ε (ocorrido) com{T1; T2; T3}
L1 (16 bits)	3	0,8889	{0,8674; 0,8875; 0,8881}
	5	0,8000	{0,7239; 0,7979; 0,7987}
	6	0,8889	{0,6857; 0,8110; 0,8846}
	7	0,7619	{0,5199; 0,7585; 0,7599}
	9	0,8889	{0,2906; 0,8722; 0,8807}
	10	0,8000	{0,2733; 0,7800; 0,7917}
	11	0,7273	{0,2541; 0,7126; 0,7209}
	12	0,6667	{0,2174; 0,6538; 0,6581}
L2 (64 bits)	3	0,9697	{0,9572; 0,9686; 0,9688}
	5	0,9846	{0,9183; 0,9819; 0,9810}
	6	0,9697	{0,8293; 0,9659; 0,9656}
	7	0,9143	{0,7149; 0,9114; 0,9118}
	9	0,8889	{0,6340; 0,8715; 0,8848}
	10	0,9143	{0,4936; 0,8835; 0,9084}
	11	0,9697	{0,4077; 0,9434; 0,9619}
	12	0,8889	{0,3934; 0,8561; 0,8825}
L3	3	0,9969	{0,9801; 0,9959; 0,9961}

(640 bits)	5	0,9922	{0,9630; **0,9979**; **0,9983**}
	6	0,9969	{0,9475; 0,9943; 0,9948}
	7	0,9938	{0,8693; 0,9907; 0,9913}
	9	0,9877	{0,6797; 0,9823; 0,9845}
	10	0,9846	{0,6751; **0,9897**; **0,9959**}
	11	0,9861	{0,6596; 0,9756; 0,9820}
	12	0,9877	{0,5767; 0,9775; 0,9829}

Como esperado, as combinações envolvendo T1 e L1 demarcaram o limite inferior de desempenho do GEOPar-1. Ressalta-se, mais uma vez, que essas combinações não são representativas de problemas reais complexos e custosos. Analisando-se as combinações que são representativas ({T2,L2}, {T2,L3}, {T3,L2}, {T3,L3}), tem-se $\varepsilon > 0{,}85$ para qualquer NP, sendo que com balanceamento de carga, tem-se $\varepsilon > 0{,}95$.

4.10 – Conclusões

Neste Capítulo, a versão paralelizada GEOPar-1 do algoritmo estocástico de otimização global GEO foi apresentada e testada num *cluster* composto por 6 nós biprocessadores interligados por uma rede Gigabit Ethernet. Foi utilizada uma função teste com dimensionalidade e tempo de avaliação configuráveis, de forma a simular cargas de processamento típicas de aplicações complexas. Os testes de desempenho efetuados comprovaram que o GEOPar-1 tem um grande potencial como otimizador paralelo, obtendo eficiência acima de 0,85 em todos os testes que procuraram simular aplicações complexas, computacionalmente custosas. Este fato, somado à universalidade de aplicação do GEO, característica mantida inalterada no GEOPar-1, aumenta ainda mais a aplicabilidade do algoritmo em problemas reais de otimização, principalmente naqueles em que o custo de processamento da avaliação da função objetivo seja elevado.

CAPÍTULO 5

HIBRIDIZAÇÃO

5.1 - Introdução

É cada vez maior na literatura o número de algoritmos híbridos, em que características de dois ou mais algoritmos são mesclados, unidos de forma a fazer parte do algoritmo híbrido (ou hibridizado) resultante (Xu, 2003; L∅vbjerg, 2002; Talbi, 2002; Barbulescu et al., 2000; Crain et al, 2000; Preux e Talbi, 1999; Vicini e Quagliarella, 1999; Desai e Patil, 1996; Hart, 1994; Ingber, 1993).

Em Wolpert e Macready (1995), o Teorema conhecido como *No Free Lunch Theorem* estabelece, basicamente, que não existe o algoritmo de otimização ideal, que seja superior a todos os demais e para todas as aplicações. Por conseguinte, torna-se um passo bastante natural tentar obter algoritmos novos e mais eficientes em determinadas aplicações por meio da hibridização. Eventualmente, esta busca pode levar ao desenvolvimento de algoritmos híbridos que sejam superiores aos seus "pais" em uma grande gama de aplicações.

A hibridização de técnicas de otimização explora as virtudes de abordagens diferentes enquanto tenta evitar suas fraquezas. Dito de outro modo, algoritmos de otimização híbridos buscam melhorar a eficiência e a robustez global de uma abordagem de otimização pela modificação das virtudes do algoritmo ao longo das diferentes fases do processo de otimização. Por exemplo, para um problema de otimização cujo espaço de projeto pode conter muitos mínimos locais, os estágios iniciais do processo de otimização poderiam ser caracterizados pela identificação de regiões promissoras do espaço de projeto, as assim chamadas bacias de atração dos mínimos locais. Algoritmos apropriados para isto, como por exemplo, Algoritmos Genéticos, desde que o fenômeno da convergência prematura seja evitado, freqüentemente não são tão eficientes na fase final de convergência para o mínimo local. Então, estes algoritmos podem ser usados com o propósito de identificação. Uma vez que regiões promissoras tenham sido identificadas, uma técnica eficiente de busca local pode ser usada para convergir sobre os mínimos locais com precisão. Uma técnica associada importante é a de modelagem de complexidade das variáveis, na qual a complexidade é gerenciada de forma a cumprir as necessidades do

algoritmo ou fase do processo de otimização atual. No exemplo acima, é claramente vantajoso usar tolerâncias de convergência maiores na fase inicial de identificação de regiões, seguido por tolerâncias apropriadamente mais apertadas na fase de convergência local. Em Amirjanov (2004) esta estratégia é usada como parte intrínseca de uma versão de Algoritmo Genético, propiciando melhoras significativas no desempenho e na capacidade de convergência do algoritmo, quando aplicado a funções de variáveis contínuas, se comparado com um Algoritmo Genético em sua versão canônica.

Outro ponto importante para pesquisa é o desenvolvimento de métricas apropriadas de comutação de algoritmos. Uma abordagem possível é manter-se com um algoritmo enquanto ele estiver fazendo progresso na busca. Quando o progresso do algoritmo diminui ou ele ultrapassar sua quota de avaliações da função objetivo, a estratégia híbrida comuta para o próximo algoritmo e continua a busca.

5.2 - Classificação

Preux e Talbi (1999) fazem uma classificação de algoritmos híbridos que se aplica principalmente para algoritmos evolutivos. No entanto, conforme mencionado pelos próprios autores, a classificação sugerida por eles não se limita apenas aos algoritmos evolutivos, podendo ser aplicada a algoritmos híbridos em geral. A figura 5.1 a seguir reproduz a classificação sugerida em Preux e Talbi (1999).

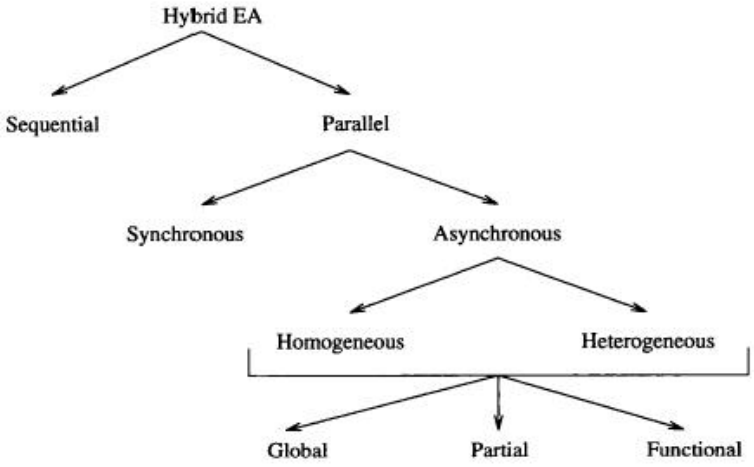


Figura 5.1 – Uma classificação para os algoritmos híbridos.
Fonte: Preux e Talbi (1999).

Segundo Preux e Talbi (1999), algoritmos de hibridização seqüencial (*Sequential Hybridization – SH*) são aqueles em que um conjunto de algoritmos é aplicado um depois do outro, cada um utilizando a saída do anterior como entrada. Os algoritmos de hibridização paralela síncrona (*Parallel Synchronous Hybridization – PSH*) são aqueles em que parte da tarefa de um algoritmo é feita por um outro algoritmo, como, por exemplo, usar um método de busca local ou global para substituir o operador mutação de um Algoritmo Genético. O termo síncrono refere-se ao fato de que o funcionamento dos algoritmos é perfeitamente sincronizado, ou seja, existe dependência de fluxo. Os algoritmos de hibridização paralela assíncrona (*Parallel Asynchronous Hybridization - PAH*) são aqueles em que um conjunto de algoritmos pesquisa o espaço de busca (por sobreposição ou divisão deste) e cooperam entre si na busca do ótimo global. Eles podem ser homogêneos, se instâncias diferentes do mesmo algoritmo são utilizadas ou heterogêneos, se algoritmos diferentes são usados. Além disso, os algoritmos de hibridização paralela assíncrona (homogêneos ou heterogêneos) podem ser classificados em globais, parciais e funcionais. Os globais são aqueles em que todos os algoritmos fazem pesquisa em todo o espaço de busca. Os parciais são aqueles em que o problema de otimização é decomposto em subproblemas, cada um tendo seu próprio subespaço de busca e em que cada algoritmo faz a busca em um destes subespaços. Em termos gerais, cada subproblema está vinculado aos demais, envolvendo restrições às soluções ou subsoluções encontradas. Assim, os algoritmos se comunicam a respeito destas restrições a fim de construir uma solução viável para o problema original. Já os funcionais são aqueles em que cada algoritmo resolve um problema diferente, muito embora, relacionado, de algum modo, à solução do problema de otimização. Um exemplo possível é um híbrido com dois algoritmos, um com a função específica de buscar o ótimo local e o outro com a função de gerar pontos de partida diversificados para o primeiro algoritmo, e usando a informação obtida com as buscas locais para evitar pontos de partida em regiões já visitadas.

5.3 – Propriedades principais

De acordo com Colorni et al. (1996), duas propriedades principais devem ser balanceadas na elaboração de um algoritmo (ou heurística):

- O grau de aprofundamento (ou aprimoramento), isto é, a quantidade de esforço direcionado a uma busca local na região atual do espaço de busca (se a região é promissora, continuar na busca local);

- O grau de exploração (ou diversificação), isto é, a quantidade de esforço gasto para efetuar buscas em regiões distantes do espaço de busca (escolher às vezes uma solução em uma região distante (e nova) mesmo que pior, para possibilitar a descoberta de soluções ainda melhores).

Estas duas propriedades são conflitantes, sendo um bom balanceamento entre elas muito importante e algo a ser cuidadosamente ajustado em cada algoritmo. A meta-heurística *Scatter Search*, por exemplo, utiliza estas duas propriedades (entre outras) como parte intrínseca de sua metodologia de busca (Glover et al., 2003; Corne et al., 1999; Glover, 1977). De um total de cinco estágios, existem dois estágios particulares da busca "*scatter*", cada um responsável por atuar numa das propriedades citadas e usando um método apropriado ao problema. Na terminologia dos autores da meta-heurística *Scatter Search*, os dois estágios são denominados de aprimoramento (*improvement*) e diversificação (*diversification*).

A hibridização de algoritmos deve permitir que as propriedades diversificação e aperfeiçoamento do algoritmo híbrido resultante sejam balanceadas na medida apropriada, de forma a obter-se o máximo de eficiência na busca pelo ótimo global da função objetivo. De fato, Goldberg e Voessner (2002) chegam a propor uma teoria de modo a tornar buscas híbridas global-local mais eficientes. Eles idealizam o algoritmo híbrido como consistindo de passos de um (método) buscador global seguido por passos de um (método) buscador local. Além disso, idealizaram o espaço de busca como um conjunto de bacias de atração que levam a "ilhas" alvo (soluções que atendem critérios especificados, além das eventuais restrições do problema original). Uma ilha-alvo é uma região do espaço de busca que é menor do que a bacia de atração, mas que contém o ótimo local daquela bacia. Eles definem também o tempo médio para se chegar a cada ilha-alvo, partindo de um ponto qualquer de sua bacia de atração. Assim, cada bacia de atração possui seu tempo médio, sendo este um dos parâmetros da teoria. Outro parâmetro é a probabilidade de uma bacia de atração ser localizada a cada chamada do buscador global. A teoria antevê ainda a

existência de dois tipos de zonas mortas, ou seja, zonas que não levam o buscador local a encontrar uma ilha-alvo. O primeiro tipo de zona morta é aquele no qual o buscador converge para o ótimo local da bacia de atração, mas este mínimo não atende o critério estabelecido para as ilhas-alvo, ou seja, a bacia de atração em questão não possui ilha-alvo, ou ainda, dito de outro modo, sua respectiva ilha-alvo é um conjunto vazio. Zonas mortas do segundo tipo são aquelas onde o buscador local não consegue convergir para a ilha-alvo (vazia ou não) dentro do tempo limite alocado para este fim. A partir dos parâmetros e conceitos expostos, a teoria mencionada equaciona o tempo total de cada iteração de busca global-local e estabelece formas de balancear os tempos de busca global e local de forma a minimizar o tempo da busca ou maximizar a confiabilidade da busca pela solução ótima global.

5.4 – Hibridizando o GEO

Norteando-se pelo objetivo primeiro da hibridização, qual seja o da melhora do desempenho, foram desenvolvidas versões híbridas do GEO com o Recozimento Simulado (RS) e do GEO com as Estratégias Evolutivas (EEs) (em inglês, *Evolutionary Strategies - ES*). A escolha do RS deve-se a três fatores: (i) sabe-se que uma das propriedades que aumentam a eficiência de um algoritmo de otimização é sua capacidade de balanceamento entre diversificação e aperfeiçoamento (vide seção 5.3); (ii) o RS usa um cronograma que estabelece valores decrescentes para seu parâmetro temperatura, obtendo com isso, uma estocasticidade decrescente ao longo da busca. O cronograma gera no RS, deste modo, um balanceamento entre diversificação, que ocorre no início da busca, e aperfeiçoamento, que ocorre no final da busca; (iii) o GEO canônico usa um valor fixo para seu parâmetro τ, mantendo assim, uma estocasticidade constante. É razoável, portanto, imaginar que a aplicação de um cronograma para variar o valor de τ ao longo da busca possa aumentar sua eficiência. A escolha das EEs deve-se principalmente a característica de auto-adaptação de parâmetros que estes algoritmos apresentam, onde o ajuste dos valores dos parâmetros ocorre de forma dinâmica, durante a própria busca. A aplicação desta característica ao GEO, gera o que poderia ser chamado de cronograma dinâmico do parâmetro τ. Em relação ao RS, possui a vantagem de não requerer o estabelecimento *a priori* de um cronograma. As versões híbridas desenvolvidas estão descritas a seguir.

5.4.1 – GEO + método do Recozimento Simulado

A versão híbrida desenvolvida incorpora ao GEO a característica de cronograma de resfriamento do algoritmo Recozimento Simulado (RS) (Kirkpatrick et al., 1983). No RS, a estocasticidade da busca é determinada pelo parâmetro "temperatura" T, que é constante por estágios e onde a definição do número de estágios, o número de avaliações da função objetivo e o valor de T em cada estágio define o cronograma de resfriamento. No GEO, o parâmetro τ define a estocasticidade da busca. Portanto, a idéia é criar para τ um cronograma semelhante ao do RS, gerando assim, um algoritmo híbrido. No âmbito da analogia do RS, a temperatura define a quantidade de energia disponível no sistema e, por consequência, define qual a probabilidade de mudar de um estado para outro. No RS, portanto, quanto maior a temperatura, maior a aleatoriedade.

No RS canônico (Kirkpatrick et al., 1983), a equação utilizada para definição da temperatura de cada estágio, n, é dada por:

$$T_n = q^n \cdot T_0 \tag{5.1}$$

onde q é o fator constante de redução de temperatura a cada estágio. Ou seja:

$$q = \frac{T_1}{T_0} = \frac{T_2}{T_1} = ... = \frac{T_n}{T_{n-1}} \quad , \quad \text{tal que } 0 < q < 1 \tag{5.2}$$

A partir da equação 5.1 é possível escrever:

$$T_n = T_0 \cdot e^{n \cdot \ln(q)} \tag{5.3}$$

Lembrando que $0 < q < 1$ (vide equação 5.2), então $\ln(q) < 0$ e a equação 5.3 indica que o parâmetro T no algoritmo RS canônico decai exponencialmente com o estágio n.

Ao contrário do que ocorre com o parâmetro T no RS, com o GEO quanto maior o valor de τ menor é a estocasticidade da busca. Sendo assim, para conseguir variar τ obtendo um efeito equivalente ao obtido com a diminuição de T, é preciso começar com um valor de τ baixo e ir aumentando-o progressivamente. A equação a seguir foi definida tendo este critério em mente:

$$t_n = t_{MAX} \cdot (1 - q^n) \quad , \quad 0 < q < 1 \tag{5.4}$$

onde τ_{MAX} é o limite máximo teórico estabelecido para τ. A exponencial usada na equação 5.1 é a mesma da equação 5.2. Isto foi feito para garantir a simetria dos cronogramas obtidos com as equações 5.1 e 5.4. Como consequência, o valor de q na equação 5.4 ainda é dado pela equação 5.1, ou seja, refere-se à taxa constante de decréscimo da temperatura T.

A partir da equação 5.4, é possível escrever:

$$\frac{t_n}{t_{n-1}} = \frac{t_{MAX} \cdot (1 - q^n)}{t_{MAX} \cdot (1 - q^{(n-1)})} = \frac{(1 - q^n)}{(1 - q^{(n-1)})} \tag{5.5}$$

A equação 5.5 revela que, diferentemente do que ocorre para T, a taxa de variação entre τ's consecutivos não é uma constante.

A título de exemplo, a Figura 5.2 ilustra as curvas obtidas para T e τ com $T_{MAX}=\tau_{MAX}=10$, n variando de 0 a 50 e q=0,9. O valor q=0,9 foi utilizado em Kirkpatrick et al. (1983). Como é possível observar pelas figuras, a variação dos parâmetros T e τ se dá de forma equivalente. Este fato pode ser constatado observando-se a simetria existente entre as duas curvas. Ambos começam com valores de elevada estocasticidade e tendem assintoticamente para valores de baixa estocasticidade. No caso de T, o valor $T_n = T_0 = 10$ representa uma estocasticidade elevada, enquanto que para τ é o valor $\tau_n = \tau_0 = 0$ que garante tal efeito. Já o efeito oposto, qual seja, o de estocasticidade baixa, é obtido para $T_n = T_{50} \cong 0{,}052$ e para $\tau_n = \tau_{50} \cong 9{,}95$. A simetria existente entre as curvas garante, para um mesmo valor de n, que tanto T como τ encontram-se a uma mesma distância do valor de referência que sinaliza a estocasticidade mais baixa possível. No caso de T, o valor de referência é o eixo horizontal T=0 e no caso de τ o valor de referência é o eixo horizontal $\tau = \tau_{MAX} = 10$.

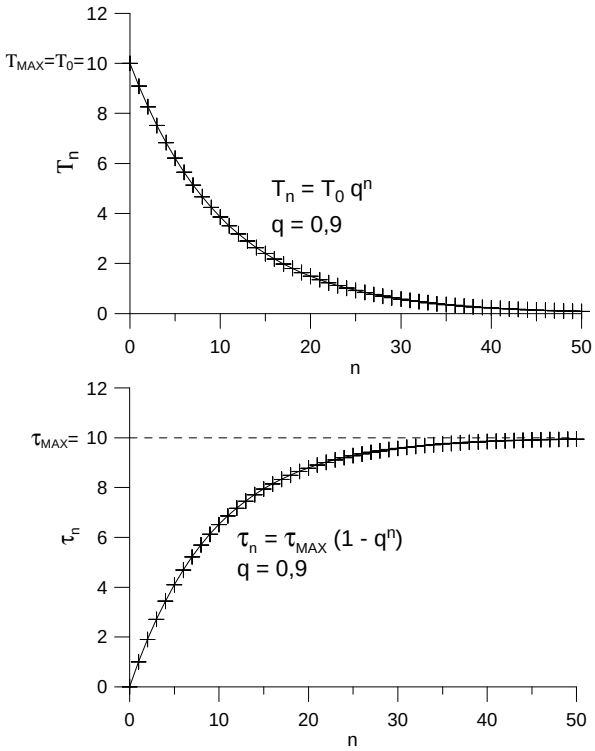


Figura 5.2 – Exemplos de Cronograma para T e τ.

Quando do estudo do equacionamento recém mostrado, observou-se que as curvas obtidas, para τ ou para T, têm grande dependência do número de estágios, n, utilizado na sua geração. Quando o número de estágios é pequeno, ocorre a situação ilustrada na Figura 5.3, onde um total de 11 estágios é utilizado ao invés de 51. Os cronogramas apresentados na Figura 5.3 são denominados, no âmbito deste trabalho, de cronogramas curtos. Tais cronogramas não são desejáveis, pois não apresentam a convergência para valores, de T ou τ, representativos de uma estocasticidade baixa ao final do cronograma. A palavra "curto" é empregada na denominação, pois, em tais cronogramas, não há um número suficiente de estágios que permita aproximar-se assintoticamente dos limites T=0 e $\tau=\tau_{MAX}$. No caso do RS, esta situação não é tão séria, pois cada iteração do RS consome apenas uma avaliação de F(**X**). Assim, o número de estágios de um cronograma do RS pode ser tão grande quanto o número total de avaliações de F(**X**) utilizado na busca pelo ótimo. Este valor, em geral, situa-se na casa dos milhares, sendo suficiente para elaborar cronogramas que evitem a situação apresentada na Figura 5.3. No caso do GEO, por outro lado, o número de avaliações de F(**X**) consumido por iteração do algoritmo depende do número de bits usado na codificação das variáveis de projeto e estas, por sua vez, dependem do problema. Para as 5 funções teste utilizadas no Capítulo 3 (vide Tabela 3.2), esse número vai de 32 até

640. Assim, é fácil perceber que, dependendo do problema, o número máximo de estágios de um cronograma para τ pode ser dezenas até milhares de vezes menor do que o número total de avaliações de F(**X**) utilizado na busca pelo ótimo. Este fato torna a ocorrência de cronogramas curtos muito mais provável com o GEO do que com o RS.

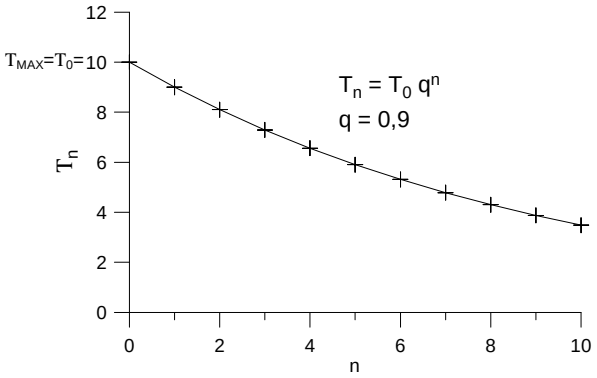


Figura 5.3a - Cronograma "curto" para T.

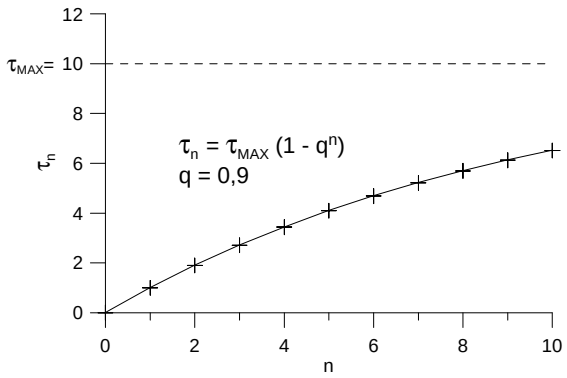


Figura 5.3b - Cronograma "curto" para τ.

A fim de evitar a ocorrência de cronogramas curtos com o GEO, a equação 5.4 é modificada conforme segue:

$$t_n = \frac{t_{MAX}}{(1-q^{n_{MAX}})} \cdot (1-q^n) \quad , \quad q > 0 \ , \ q \neq 1 \tag{5.6}$$

onde n_{MAX} é o limite superior para o número de estágios utilizado no cronograma. A Figura 5.4a mostra o resultado da aplicação desta equação na geração do cronograma originalmente obtido com a equação 5.5 e apresentada na Figura 5.3b. Uma vantagem adicional da equação 5.7 é que ela é mais flexível, permitindo a utilização de valores de q menores ou maiores do que 1. A Figura 5.4b mostra os cronogramas obtidos para $q \in \{0,1;0,5;0,8;0,99;1,25;2;10\}$. Embora a utilização de cronogramas de τ com q>1 seja contrário ao princípio que norteia o algoritmo do RS, que é o de realizar uma busca pelo ótimo aproximando-se da região de baixa estocasticidade cada vez mais lentamente e não o contrário, ainda assim, a possibilidade de usar cronogramas de τ com q>1 é interessante, pois permite averiguar se tal princípio também é válido para o GEO.

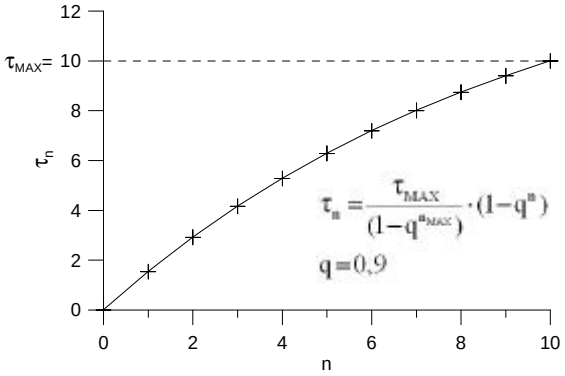


Figura 5.4a – Novo cronograma para τ.

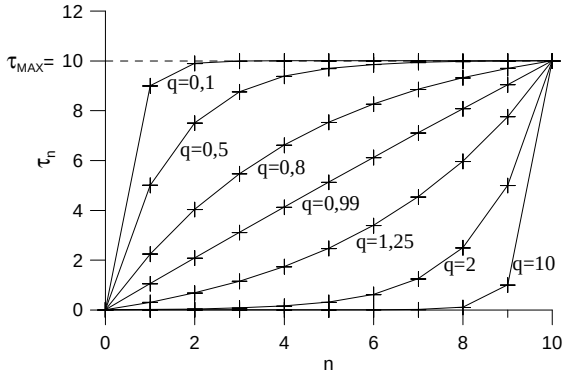


Figura 5.4b - Cronogramas para vários q's.

Indo um pouco além, os cronogramas de τ a serem testados com o auxílio de funções teste prevêem também a existência de ciclos que se repetem ao longo de uma busca com o GEO. A idéia é definir-se um número de iterações do GEO por ciclo, criar o cronograma usando o número de iterações definido para o ciclo e aplicá-lo sucessivamente até o final da busca.

No GEO, cada iteração consome um número de avaliações de F(**X**) igual à soma dos bits usados na codificação das N variáveis de projeto, ou seja:

$$L = \sum_{j=1}^{N} l_j \tag{5.7}$$

Em geral, utiliza-se um mesmo número de bits na codificação das variáveis. Sendo assim, $l_j = l$=cte. Com isso, a equação 5.7 reduz-se a:

$$L = N \cdot l \tag{5.8}$$

O critério de parada comumente utilizado nas implementações do GEO é a definição de um número máximo de avaliações de F(**X**), aqui chamado de NAF_{MAX}. Tendo isto em vista, o número de iterações do GEO durante uma busca é calculado por:

$$NI = \frac{NAF_{MAX}}{N \cdot l} \tag{5.9}$$

Já o número de ciclos de τ aplicados ao longo da busca é determinado por:

$$NC = \frac{NI}{(n_{MAX} + 1)} \tag{5.10}$$

onde NI é calculado com o auxílio da equação 5.9.

Imaginando-se, por exemplo, que o GEO seja aplicado a um problema com N=5 variáveis de projeto, *l*=16 bits por variável, $NAF_{MAX}= 10^4$ avaliações de F(**X**) e n_{MAX}=19 (=20 iterações por ciclo), então, da equação 5.9 tem-se que NI=125 iterações e, da equação 5.10, tem-se que NC=6,25. Assim, nesse exemplo, seriam aplicados 6,25 ciclos de variação de τ. A Figura 5.5 ilustra o cronograma total resultante para τ como função do número de avaliações de F(**X**) e para q=0,9.

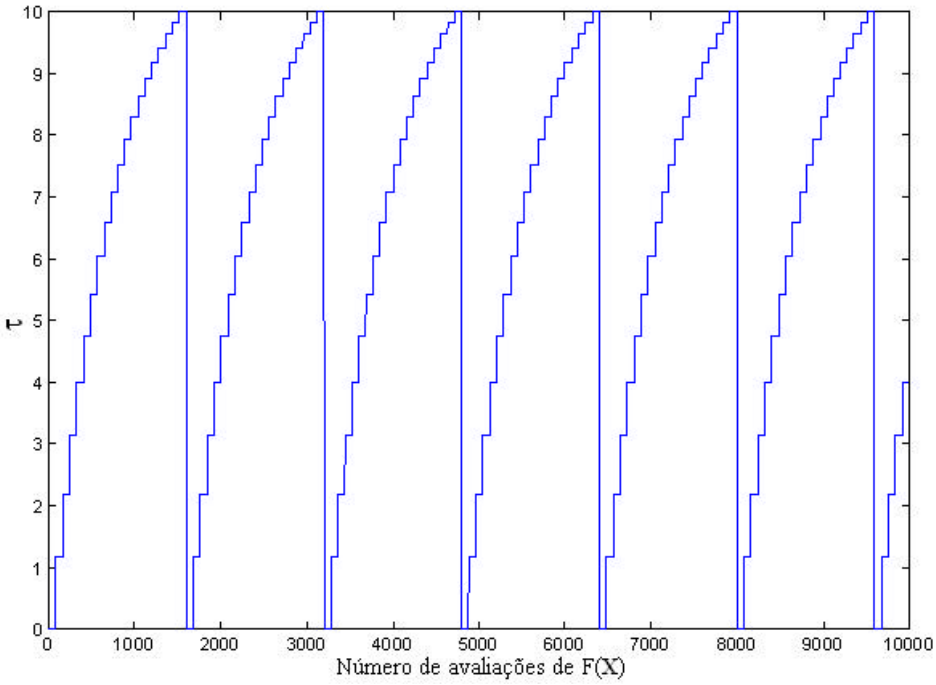


Figura 5.5 – Exemplo de cronograma cíclico para τ com n_{MAX}=19 e q=0,9.

As funcionalidades descritas nos parágrafos precedentes foram incorporadas ao GEO e GEO_{var} da SA1 (vide seção 3.7.1, Capítulo 3). A versão híbrida resultante não mais possui τ como um parâmetro de ajuste do algoritmo. Por outro lado, dois outros parâmetros surgem, vinculados ao cronograma de τ. São eles, o número de iterações por ciclo, n_{MAX} e q, que controla a inflexão do cronograma (vide Figura 5.3b). A rigor, τ_{MAX} também deveria ser considerado um parâmetro. No entanto, como τ? τ_{MAX} deve corresponder a T? 0, ou seja, uma condição de estocasticidade muito baixa, então é suficiente arbitrar-se para τ_{MAX}

um valor que assegure tal condição, mantendo-o inalterado. Nesta Tese, o valor τ_{MAX}=10 é utilizado. Apresentam-se a seguir, os passos das versões híbridas de GEO e de GEO_{var}.

A versão híbrida de GEO é composta dos seguintes passos:

1. Estabeleça um cronograma para τ. Faça τ_{MAX}=10 e defina valores para a inflexão q, e para n_{MAX}, o número limite de iterações por ciclo do cronograma. Faça n=-1.
2. Inicialize aleatoriamente uma seqüência binária **C**, de comprimento L que codifica N variáveis de projeto. Dentro de **C**, cada variável de projeto X_j é codificada em uma parcela com comprimento l_j, $j \in \{1,2,...,N\}$ e tal que $\sum_j l_j = L$. Converta **C** em **X** e calcule o valor da função objetivo F(**X**) e faça $\mathbf{X_{melhor}} = \mathbf{X}$ e $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$.
3. Faça n=n+1. Se $n>n_{MAX}$, faça n= 0. Faça $F(X_{melhor})_{anterior} = F(X_{melhor})$. Para cada bit $i \in \{1,2,...,L\}$ da seqüência **C**, faça:
    a. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule $F(\mathbf{X_i})$. Em seguida, se $F(\mathbf{X_i}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X_i})$ e $\mathbf{X_{melhor}} = \mathbf{X_i}$.
    b. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - F(\mathbf{X_{melhor}})_{anterior}$, que indica o ganho (ou perda) que se têm ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.
    c. Retorne o bit i ao seu valor original.
4. Ordene todos os bits $i \in \{1,2,...,L\}$ de acordo com os seus índices de adaptação, de k=1 para o menos adaptado à k=L, para o mais adaptado. Crie uma lista de valores (i,k) relacionando a posição física i do bit em **C** com seu respectivo número de ordem k, onde $i,k \in \{1,2,...,L\}$. Se $\Delta F(\mathbf{X_i}) = \Delta F(\mathbf{X_j})$, i?j, estabeleça a ordem entre i e j de forma aleatória.
5. Calcule o valor de τ para a iteração atual: $t = t_n = \frac{t_{MAX}}{(1-q^{n_{MAX}})} \cdot (1-q^n)$

6. Escolha com igual probabilidade um bit candidato i para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual à $P_i(k)$, confirme o bit i para ser modificado. Repetir este passo até que um bit seja confirmado. Quando isso acontecer, faça $i_{esc} = i$.
7. Faça $\mathbf{C} = \mathbf{C_{iesc}}$, ou seja, em **C**, mute o bit i_{esc}. Converta **C** em **X** e calcule F(**X**).
8. Repita os passos 3 a 7 até que um dado critério de parada seja satisfeito.
9. Retorne $\mathbf{X_{melhor}}$ e F($\mathbf{X_{melhor}}$).

A versão híbrida de GEO_{var} é composta dos seguintes passos:

1. Estabeleça um cronograma para τ. Faça τ_{MAX}=10 e defina valores para a inflexão q, e para n_{MAX}, o número limite de iterações por ciclo do cronograma. Faça n=-1.
2. Inicialize aleatoriamente N seqüências binárias C_j, $j \in \{1,2,...,N\}$ que codificam as N variáveis de projeto X_j. Cada C_j tem l_j bits, ou seja, o elemento $c_{j,i} \subset C_j$, $i \in \{1,2,..., l_j\}$ e $c_{j,i} \in \{0,1\}$. Assim, o vetor **X** contendo as variáveis de projeto é codificado em uma matriz **C** cuja linha j contém a seqüência binária C_j utilizada na codificação da variável X_j. Converta **C** em **X** e calcule o valor da função objetivo F(**X**) e faça $\mathbf{X_{melhor}} = \mathbf{X}$ e F($\mathbf{X_{melhor}}$) = F(**X**).
3. Faça n=n+1. Se $n>n_{MAX}$, faça n= 0. Faça F($\mathbf{X_{melhor}}$)$_{anterior}$ = F($\mathbf{X_{melhor}}$).
4. Calcule o valor de τ para a iteração atual: $t = t_n = \frac{t_{MAX}}{(1-q^{n_{MAX}})} \cdot (1-q^n)$
5. Para cada variável $j \in \{1,2,...,N\}$ faça:
    a. Para cada bit $i \in \{1,2,..., l_j\}$ da seqüência C_j, faça:
        1º. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule F($\mathbf{X_i}$). Em seguida, se F($\mathbf{X_i}$) < F($\mathbf{X_{melhor}}$) então faça F($\mathbf{X_{melhor}}$) = F($\mathbf{X_i}$) e $\mathbf{X_{melhor}} = \mathbf{X_i}$.

2º. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - \mathbf{F(X_{melhor})_{anterior}}$, que indica o ganho (ou perda) obtido ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.

3º. Retorne o bit i ao seu valor original.

b. Ordene os bits $i \in \{1,2,..., l_j\}$, ou seja, os bits da variável j, de acordo com os seus índices de adaptação $\Delta F(\mathbf{X_i})$, de k=1 para o menor $\Delta F(\mathbf{X_i})$ até $k=l_j$, para o maior $\Delta F(\mathbf{X_i})$, onde l_j é o número de bits da variável j.

c. Crie uma lista $(i,k)_j$, relacionando a posição física do bit i em C_j com seu número de ordem k. Se $\Delta F(\mathbf{X_i})_m = \Delta F(\mathbf{X_i})_n$, $\forall m?n$, estabeleça a ordem entre m e n de forma aleatória.

d. Escolha com igual probabilidade um bit candidato $i \in \{1,2,..., l_j\}$ para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual a $P_i(k)$, confirme o bit para mutar. Repetir este passo até que um bit seja confirmado para ser modificado. Quando isso acontecer, faça $i_{esc_j} = i$.

6. Faça $\mathbf{C} = \mathbf{C_{esc}}$, onde $\mathbf{C_{esc}}$ é a configuração resultante da mutação em **C** dos N bits escolhidos no passo 2.d (bits i_{esc_j}, $j \in \{1,2,...,N\}$). Converta **C** em **X** e calcule F(**X**). Se $F(\mathbf{X}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$ e $\mathbf{X_{melhor}} = \mathbf{X}$.
7. Repita os passos 3 a 6 até que um dado critério de parada seja satisfeito.
8. Retorne $\mathbf{X_{melhor}}$ e $F(\mathbf{X_{melhor}})$.

As versões híbridas foram aplicadas às 5 funções teste FT_i, $i \in \{1,2,..,5\}$ definidas no Capítulo 3 (vide Tabela 3.2). Os resultados a serem apresentados refletem o melhor ajuste, obtido por múltiplas tentativas, para os parâmetros n_{MAX} e q, conforme Tabela 5.1. Surpreendentemente, a maioria dos melhores resultados foi obtida para valores de q maiores do que 1. Isto significa que os cronogramas de τ resultantes possuem inflexão contrária àquela preconizada pelo RS. Como pode ser visto na Figura 5.4b, cronogramas de

τ com q>1 passam a maior parte do tempo de busca (iterações do algoritmo) na região de estocasticidade alta. Outro fato curioso é que, na maioria das vezes, n_{MAX} é bastante baixo, o que significa que o número de ciclos tende a ser elevado. Os valores dos parâmetros ajustados levam a crer que, com GEO e GEO_{var}, ciclos de τ de pequena duração e alta freqüência de aplicação ao longo da busca geram, na maioria das vezes, melhores resultados.

Tabela 5.1 – Valores dos parâmetros q e n_{MAX}.

FT	GEO híbrido		GEO_{var} híbrido	
	q	n_{MAX}	q	n_{MAX}
FT_1	1,9	13	2,32	4
FT_2	67	2	6,67	2
FT_3	125	2	2,13	2
FT_4	125	2	0,16	5
FT_5	0,001	4	1,25	16

As Figuras 5.6 até 5.10 apresentam os resultados médios de 50 execuções independentes, obtidos com as versões híbridas de GEO e GEO_{var}. O número de avaliações de F(**X**) a cada execução é 10^4 para FT_1 e 10^5 para as demais funções teste. Para a codificação das variáveis de projeto de todas as funções teste foi utilizado l_j=l=16 bits por variável. Em cada figura, são apresentadas também as curvas obtidas com o GEO_1 e o GEO_{var1}.

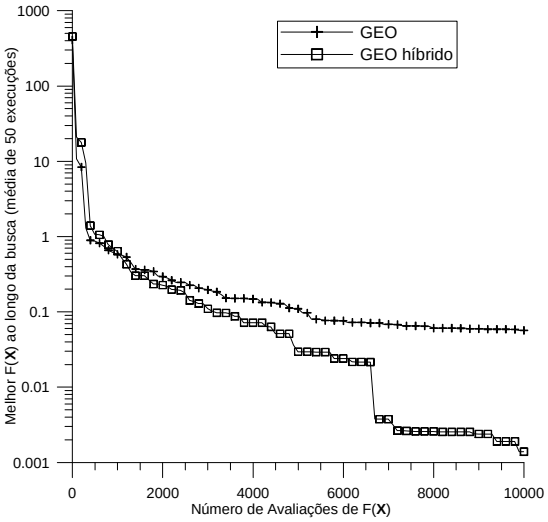


Figura 5.6a - FT_1 x NAF (GEO)

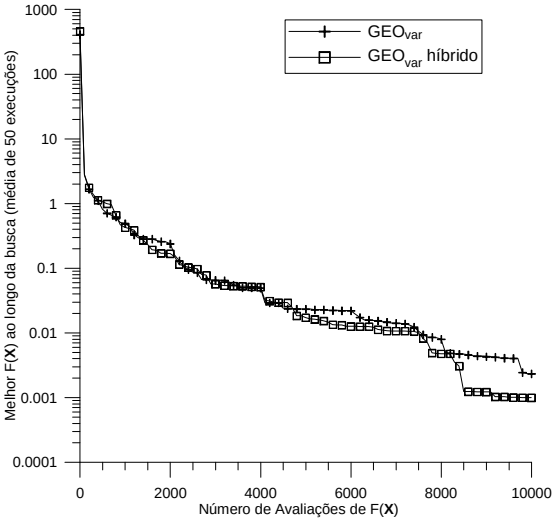


Figura 5.6b - FT_1 x NAF (GEO_{var})

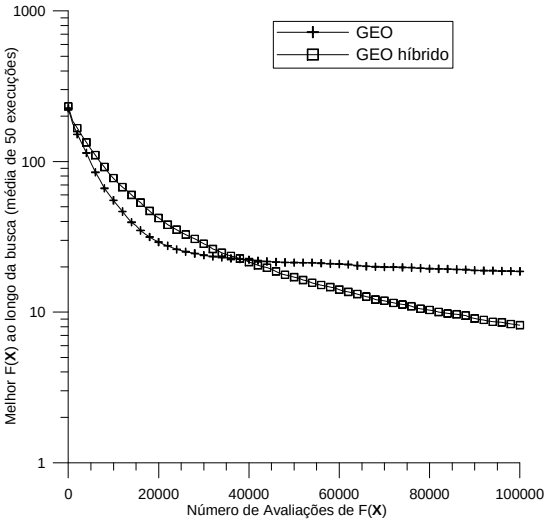


Figura 5.7a - FT_2 x NAF (GEO)

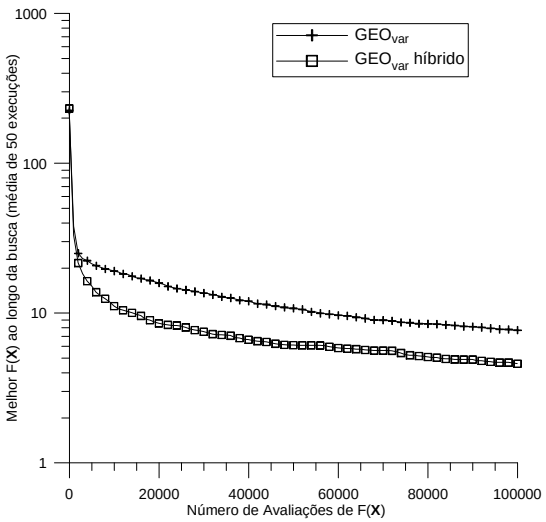


Figura 5.7b - FT_2 x NAF (GEO_{var})

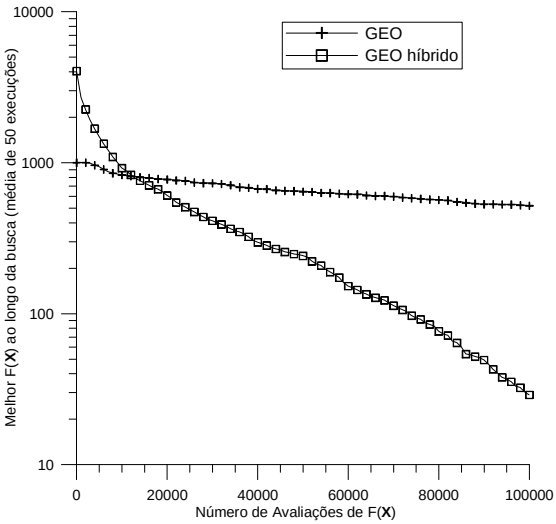


Figura 5.8a - FT_3 x NAF (GEO)

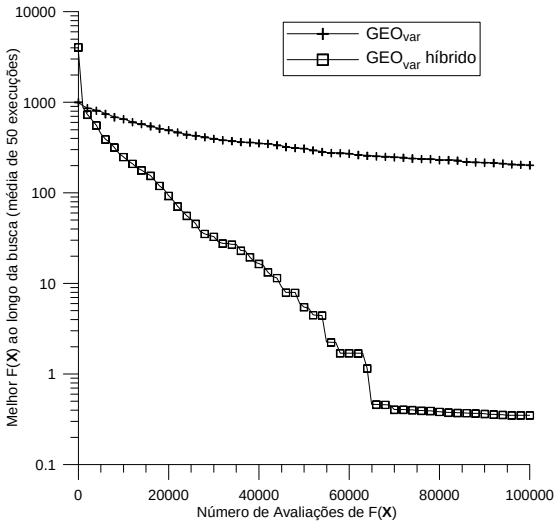


Figura 5.8b - FT_3 x NAF (GEO_{var})

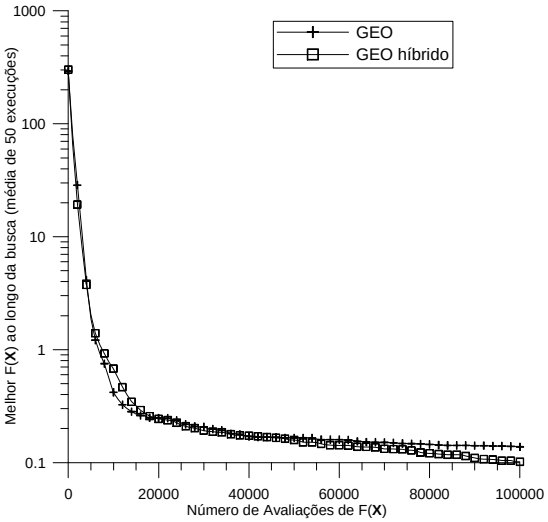


Figura 5.9a - FT_4 x NAF (GEO)

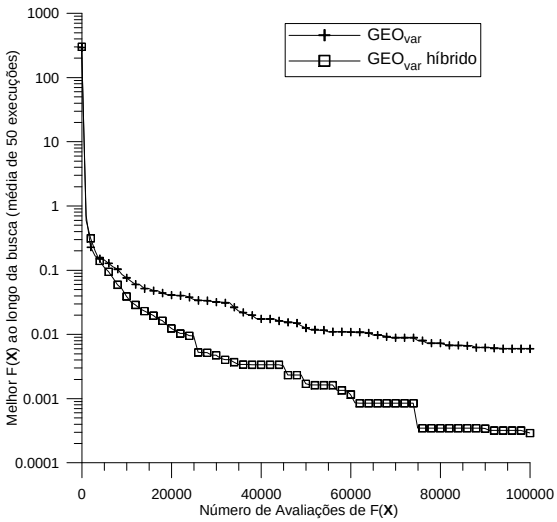


Figura 5.9b - FT_4 x NAF (GEO_{var})

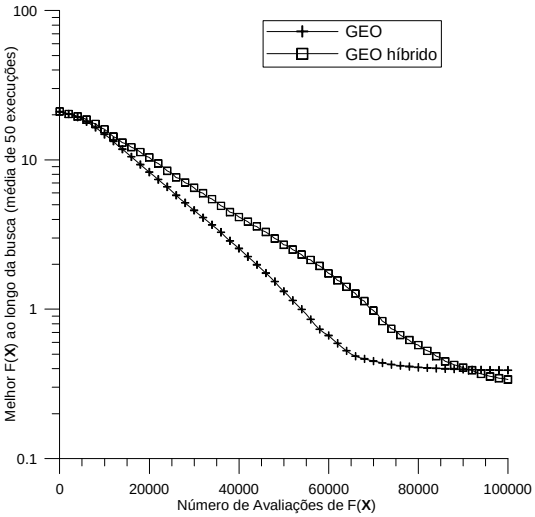


Figura 5.10a - FT_5 x NAF (GEO)

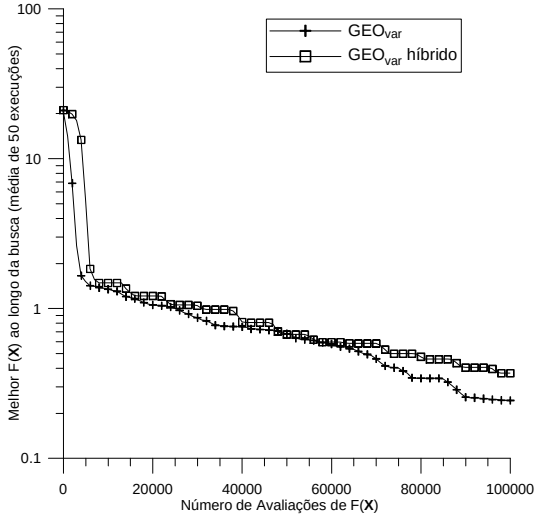


Figura 5.10b - FT_5 x NAF (GEO_{var})

As curvas obtidas mostram que, de fato, houve melhora no desempenho de GEO e GEO_{var} com a hibridização e isto para todas as FTs. A única exceção foi a FT_5 com GEO_{var}. A hibridização teve maior impacto na função de Schwefel (FT_3), sendo a função de Ackley (FT_5) aquela de menor sensibilidade à modificação. Na comparação entre GEO e GEO_{var}, em geral, o GEO_{var} híbrido apresentou ganhos de desempenho mais significativos do que o GEO híbrido.

Conforme já comentado, os valores dos parâmetros ajustados de q e n_{MAX} (vide Tabela 5.1) correspondem a ciclos de variação de τ extremamente curtos e, portanto, de alta freqüência. Em alguns casos, $n_{MAX}=2$, totalizando apenas 3 iterações por ciclo. Como o equacionamento elaborado impõe $\tau_0=0$ e $\tau_{n_{MAX}}=\tau_{MAX}=10$, então sobra apenas uma iteração onde τ pode assumir valores intermediários entre as regiões de estocasticidade alta ($\tau=\tau_0=0$) e baixa ($\tau=\tau_{MAX}=10$). Na seção 5.3, o balanceamento entre duas propriedades ou fases foi estabelecido como de importância fundamental para o desempenho de um algoritmo de otimização: (i) a fase de diversificação e (ii) a fase de aperfeiçoamento das soluções. No contexto dos cronogramas com ciclos extremamente curtos de τ, a fase (i) corresponde a $\tau=\tau_0=0$ e a fase (ii) corresponde a $\tau=\tau_{MAX}=10$. O fato dos melhores resultados das versões híbridas de GEO e GEO_{var} ter ocorrido para ciclos tão curtos de τ sugere que a rápida alternância entre fases de estocasticidade baixa e fases de estocasticidade alta é mais importante, no caso destes dois híbridos, do que a existência de ciclos longos em que a variação de τ ocorra gradualmente.

A fim de explorar tal hipótese, uma segunda versão híbrida foi implementada. Nela, não há mais uma equação governando a variação de τ e o cronograma compõe-se de duas

fases que se alternam. Na primeira fase, durante nd iterações o valor de τ é constante e igual a 0, gerando uma fase de diversificação. Em seguida, na segunda fase, durante "na" iterações o valor de τ é mantido constante e igual a 10, gerando uma fase de aperfeiçoamento. Os parâmetros dessa segunda versão híbrida são, portanto, nd e na.

A segunda versão híbrida de GEO é composta dos seguintes passos:

1. Estabeleça um cronograma para τ. Faça τ_{MAX}=10 e defina valores para o número de iterações alocadas para a fase de diversificação, nd, e para o número de iterações alocadas para a fase de aperfeiçoamento, na. Faça ni=nd + na e faça n=0.
2. Inicialize aleatoriamente uma seqüência binária **C**, de comprimento L que codifica N variáveis de projeto. Dentro de **C**, cada variável de projeto X_j é codificada em uma parcela com comprimento l_j, $j \in \{1,2,...,N\}$ e tal que $\sum_j l_j = L$. Converta **C** em **X** e calcule o valor da função objetivo F(**X**) e faça $\mathbf{X_{melhor}} = \mathbf{X}$ e $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$.
3. Faça n=n+1. Faça $F(X_{melhor})_{anterior} = F(X_{melhor})$. Para cada bit $i \in \{1,2,...,L\}$ da seqüência **C**, faça:
    a. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule $F(\mathbf{X_i})$. Em seguida, se $F(\mathbf{X_i}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X_i})$ e $\mathbf{X_{melhor}} = \mathbf{X_i}$.
    b. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - F(\mathbf{X_{melhor}})_{anterior}$, que indica o ganho (ou perda) que se têm ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.
    c. Retorne o bit i ao seu valor original.
4. Ordene todos os bits $i \in \{1,2,...,L\}$ de acordo com os seus índices de adaptação, de k=1 para o menos adaptado à k=L, para o mais adaptado. Crie uma lista de valores (i,k) relacionando a posição física i do bit em **C** com seu respectivo número de ordem k, onde $i,k \in \{1,2,...,L\}$. Se $\Delta F(\mathbf{X_i}) = \Delta F(\mathbf{X_j})$, i?j, estabeleça a ordem entre i e j de forma aleatória.

5. Calcule o valor de τ para a iteração atual: Se MOD(n/ni)>nd então faça $\tau=\tau_{MAX}$, senão faça $\tau=0$.
6. Escolha com igual probabilidade um bit candidato i para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual à $P_i(k)$, confirme o bit i para ser modificado. Repetir este passo até que um bit seja confirmado. Quando isso acontecer, faça $i_{esc} = i$.
7. Faça $\mathbf{C} = \mathbf{C_{iesc}}$, ou seja, em **C**, mute o bit i_{esc}. Converta **C** em **X** e calcule F(**X**).
8. Repita os passos 3 a 7 até que um dado critério de parada seja satisfeito.
9. Retorne $\mathbf{X_{melhor}}$ e F($\mathbf{X_{melhor}}$).

A segunda versão híbrida de GEO_{var} é composta dos seguintes passos:

1. Estabeleça um cronograma para τ. Faça $\tau_{MAX}=10$ e defina valores para o número de iterações alocadas para a fase de diversificação, nd, e para o número de iterações alocadas para a fase de aperfeiçoamento, na. Faça ni=nd + na e faça n=0.
2. Inicialize aleatoriamente N seqüências binárias C_j, $j\in\{1,2,...,N\}$ que codificam as N variáveis de projeto X_j. Cada C_j tem l_j bits, ou seja, o elemento $c_{j,i}\subset C_j$, $i\in\{1,2,..., l_j\}$ e $c_{j,i}\in\{0,1\}$. Assim, o vetor **X** contendo as variáveis de projeto é codificado em uma matriz **C** cuja linha j contém a seqüência binária C_j utilizada na codificação da variável X_j. Converta **C** em **X** e calcule o valor da função objetivo F(**X**) e faça $\mathbf{X_{melhor}} = \mathbf{X}$ e F($\mathbf{X_{melhor}}$) = F(**X**).
3. Faça n=n+1. Faça F($\mathbf{X_{melhor}}$)$_{anterior}$ = F($\mathbf{X_{melhor}}$).
4. Calcule o valor de τ para a iteração atual: Se MOD(n/ni)>nd então faça $\tau=\tau_{MAX}$, senão faça $\tau=0$.
5. Para cada variável $j\in\{1,2,...,N\}$ faça:
    a. Para cada bit $i\in\{1,2,..., l_j\}$ da seqüência C_j, faça:

1º. Mude o valor do bit i (de 0 para 1 ou 1 para 0), obtendo assim uma configuração de bits $\mathbf{C_i}$. Converta $\mathbf{C_i}$ em $\mathbf{X_i}$ e calcule $F(\mathbf{X_i})$. Em seguida, se $F(\mathbf{X_i}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X_i})$ e $\mathbf{X_{melhor}} = \mathbf{X_i}$.

2º. Atribua ao bit i um índice de adaptação $\Delta F(\mathbf{X_i}) = F(\mathbf{X_i}) - \mathbf{F(X_{melhor})_{anterior}}$, que indica o ganho (ou perda) obtido ao mudar o valor do bit, comparado com o melhor valor encontrado para a função objetivo até a iteração anterior.

3º. Retorne o bit i ao seu valor original.

b. Ordene os bits $i \in \{1,2,..., l_j\}$, ou seja, os bits da variável j, de acordo com os seus índices de adaptação $\Delta F(\mathbf{X_i})$, de k=1 para o menor $\Delta F(\mathbf{X_i})$ até $k=l_j$, para o maior $\Delta F(\mathbf{X_i})$, onde l_j é o número de bits da variável j.

c. Crie uma lista $(i,k)_j$, relacionando a posição física do bit i em C_j com seu número de ordem k. Se $\Delta F(\mathbf{X_i})_m = \Delta F(\mathbf{X_i})_n$, $\forall m?n$, estabeleça a ordem entre m e n de forma aleatória.

d. Escolha com igual probabilidade um bit candidato $i \in \{1,2,..., l_j\}$ para ser modificado. Calcule $P_i(k) = k^{-\tau}$ e gere um número aleatório ALE com distribuição uniforme no intervalo [0,1]. Se ALE for menor ou igual a $P_i(k)$, confirme o bit para mutar. Repetir este passo até que um bit seja confirmado para ser modificado. Quando isso acontecer, faça $i_{esc_j} = i$.

6. Faça $\mathbf{C} = \mathbf{C_{esc}}$, onde $\mathbf{C_{esc}}$ é a configuração resultante da mutação em **C** dos N bits escolhidos no passo 2.d (bits i_{esc_j}, $j \in \{1,2,...,N\}$). Converta **C** em **X** e calcule $F(\mathbf{X})$. Se $F(\mathbf{X}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X})$ e $\mathbf{X_{melhor}} = \mathbf{X}$.
7. Repita os passos 3 a 6 até que um dado critério de parada seja satisfeito.
8. Retorne $\mathbf{X_{melhor}}$ e $F(\mathbf{X_{melhor}})$.

A segunda versão híbrida de GEO e GEO_{var} foi aplicada às 5 funções teste FT_i, $i \in \{1,2,..,5\}$ definidas no Capítulo 3 (vide Tabela 3.2). Os resultados a serem apresentados

refletem o melhor ajuste, obtido por múltiplas tentativas, para os parâmetros nd e na, conforme Tabela 5.2.

Tabela 5.2 – Valores dos parâmetros nd e na, e da razão nd/na.

FT	GEO híbrido			GEO_{var} híbrido		
	nd	na	nd/na	nd	na	nd/na
FT_1	13	6	2,2	2	4	0,5
FT_2	5	3	1,7	2	2	1,0
FT_3	2	1	2,0	1	2	0,5
FT_4	2	2	1,0	1	6	0,2
FT_5	2	9	0,2	5	11	0,5

A razão nd/na, mostrada na Tabela 5.2, diz qual o balanço entre as fases de diversificação e aperfeiçoamento que obtém os melhores resultados para cada uma das FT's. Com isso, sabe-se que para o híbrido de GEO tal razão varia de 0,2 para a FT_5 até 2,2 para a FT_1. Assim, a FT_1 necessita 2,2 vezes mais iterações de diversificação do que de aperfeiçoamento. Já para a FT_5 a situação se inverte, visto que esta função necessita 5 vezes menos iterações de diversificação do que de aperfeiçoamento. Para o híbrido de GEO_{var}, a razão nd/na varia de 0,2 para a FT_4 até 1 para a FT_2. É possível observar também que, para todas as FT's menos FT_5, o híbrido de GEO requer mais diversificação do que GEO_{var}, isto é, possui maior razão nd/na.

As Figuras 5.11 até 5.15 apresentam os resultados médios de 50 execuções independentes, obtidos com a segunda versão híbrida de GEO e GEO_{var}. O número de avaliações de $F(\mathbf{X})$ a cada execução é 10^4 para FT_1 e 10^5 para as demais funções teste. Para a codificação das variáveis de projeto de todas as funções teste foi utilizado $l_j = l = 16$ bits por variável. Em cada figura, são apresentadas também as curvas obtidas pela primeira versão híbrida, bem como os resultados obtidos com o GEO_1 e o GEO_{var1}.

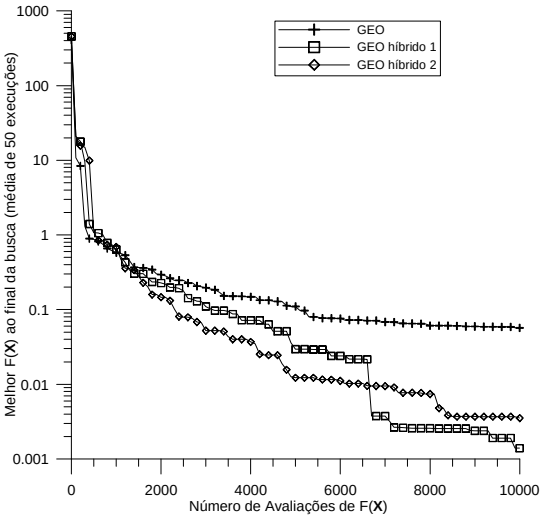


Figura 5.11a - FT_1 x NAF (GEO)

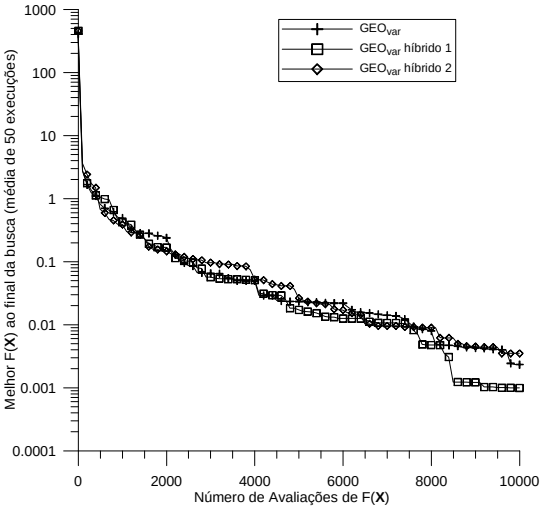


Figura 5.11b - FT_1 x NAF (GEO_{var})

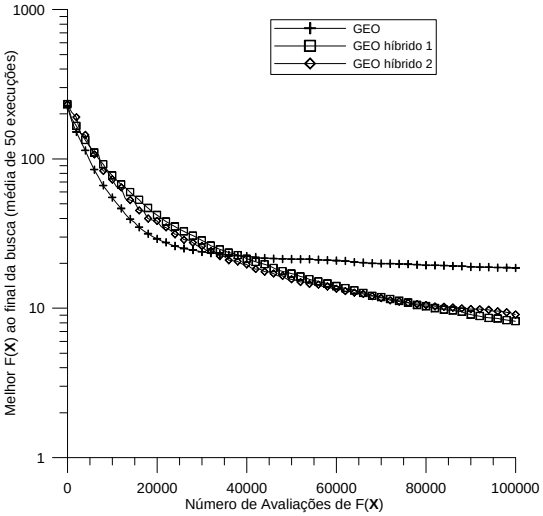


Figura 5.12a - FT_2 x NAF (GEO)

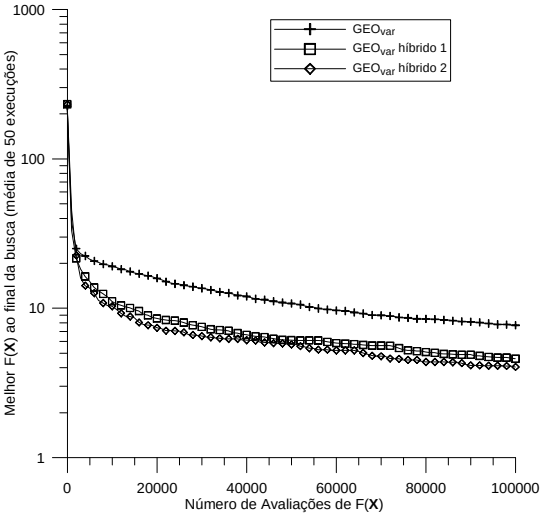


Figura 5.12b - FT_2 x NAF (GEO_{var})

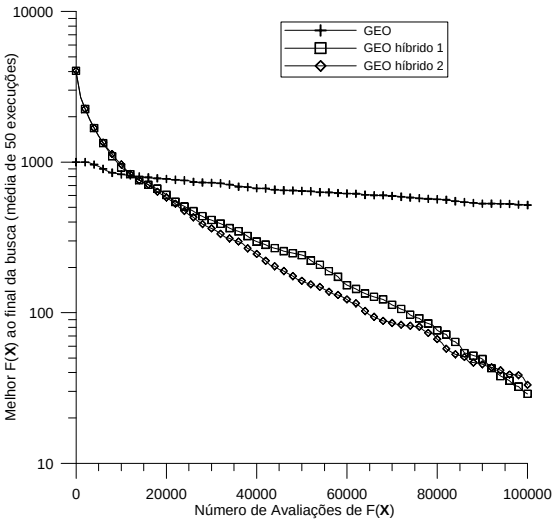


Figura 5.13a - FT_3 x NAF (GEO)

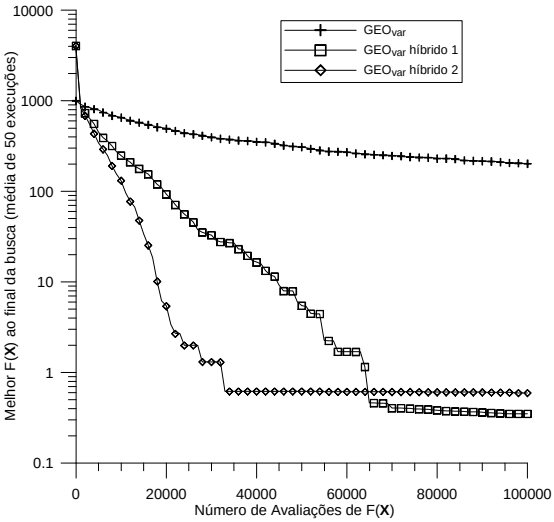


Figura 5.13b - FT_3 x NAF (GEO_{var})

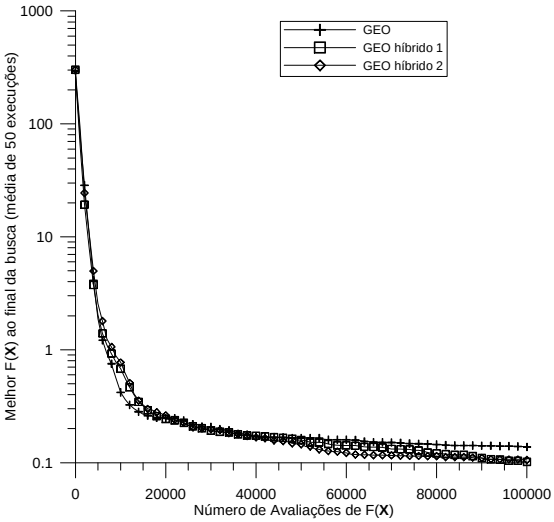


Figura 5.14a - FT_4 x NAF (GEO)

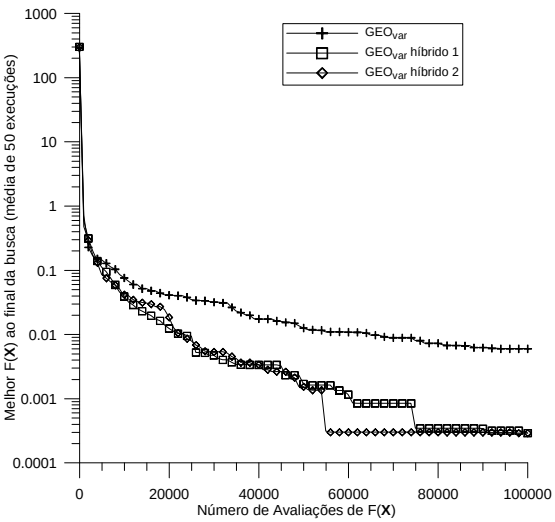


Figura 5.14b - FT_4 x NAF (GEO_{var})

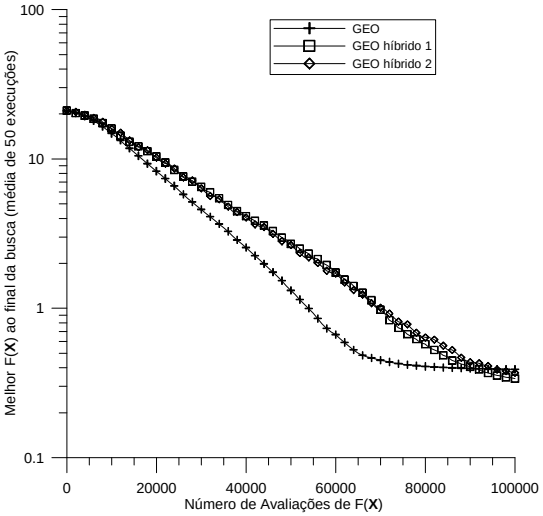


Figura 5.15a - FT_5 x NAF (GEO)

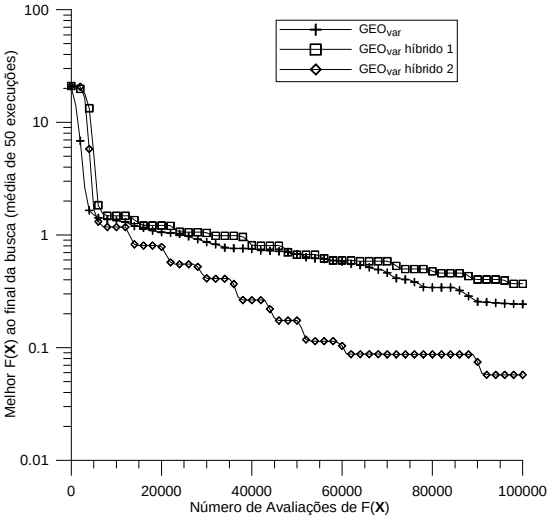


Figura 5.15b - FT_5 x NAF (GEO_{var})

As curvas obtidas mostram que esta segunda versão híbrida de GEO e GEO_{var} apresenta um desempenho muito similar ao já alcançado pela primeira versão. Talvez a única exceção digna de nota seja o desempenho da segunda versão híbrida de GEO_{var} com a FT_5, que foi consideravelmente melhor que o da primeira versão.

Conforme já mencionado no Capítulo 3, as funções teste FT_i, $i \in \{1,2,...,5\}$ foram utilizadas por Potter e De Jong (1994) para testar uma versão de AG denominada CCGA. Posteriormente, Sousa e Ramos (2002) compararam os resultados do CCGA com o GEO e GEO_{var} originais, mantendo, inclusive, as escalas lineares originais dos gráficos do CCGA. Os resultados das versões híbridas recém desenvolvidas são comparados com aqueles apresentados nos artigos citados com o auxílio das Figuras 5.16 a 5.20. A escala original linear é mantida, bem como os limites utilizados para os eixos. Visando maior clareza,

dada a equivalência de desempenho entre a primeira e a segunda versão híbrida, escolheu-se apenas a segunda versão para fazer parte da comparação. As curvas mostradas para as versões híbridas são os resultados médios de 50 execuções independentes.

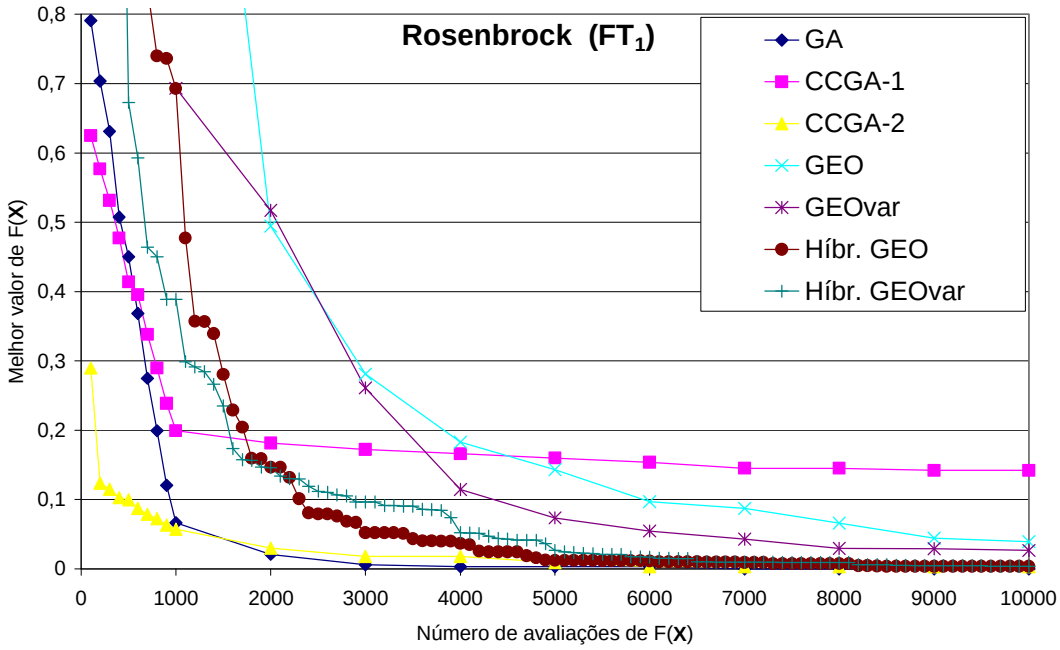


Figura 5.16 – Comparação dos resultados da segunda versão do híbrido de GEO e GEO_{var} + RS com os obtidos pelo CCGA e pelo GEO e GEO_{var} originais para a FT_1.

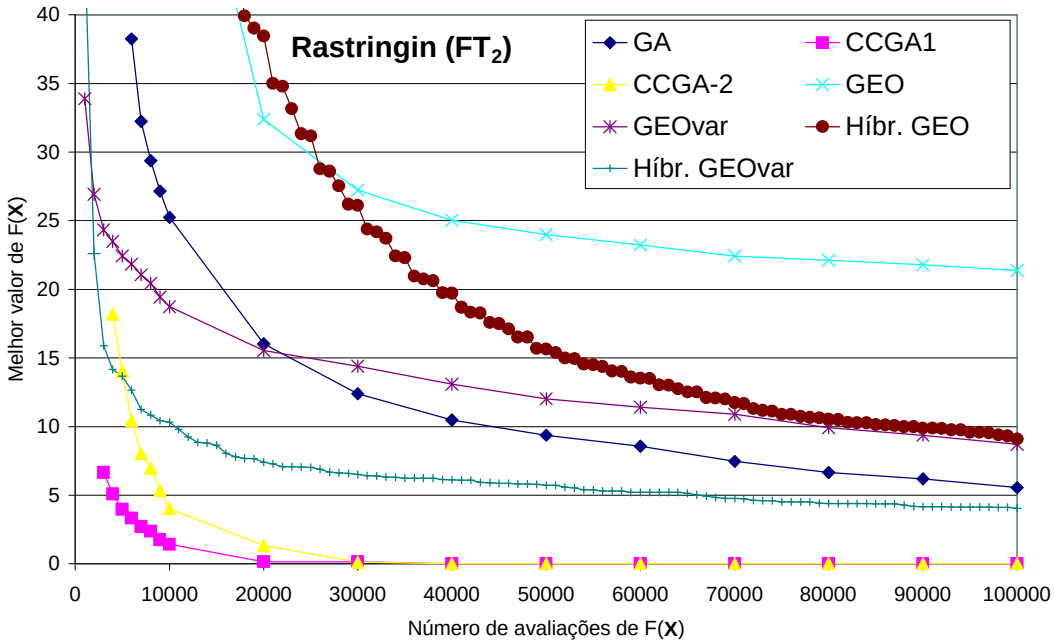


Figura 5.17 – Comparação dos resultados da segunda versão do híbrido de GEO e GEO_{var} + RS com os obtidos pelo CCGA e pelo GEO e GEO_{var} originais para a FT_2.

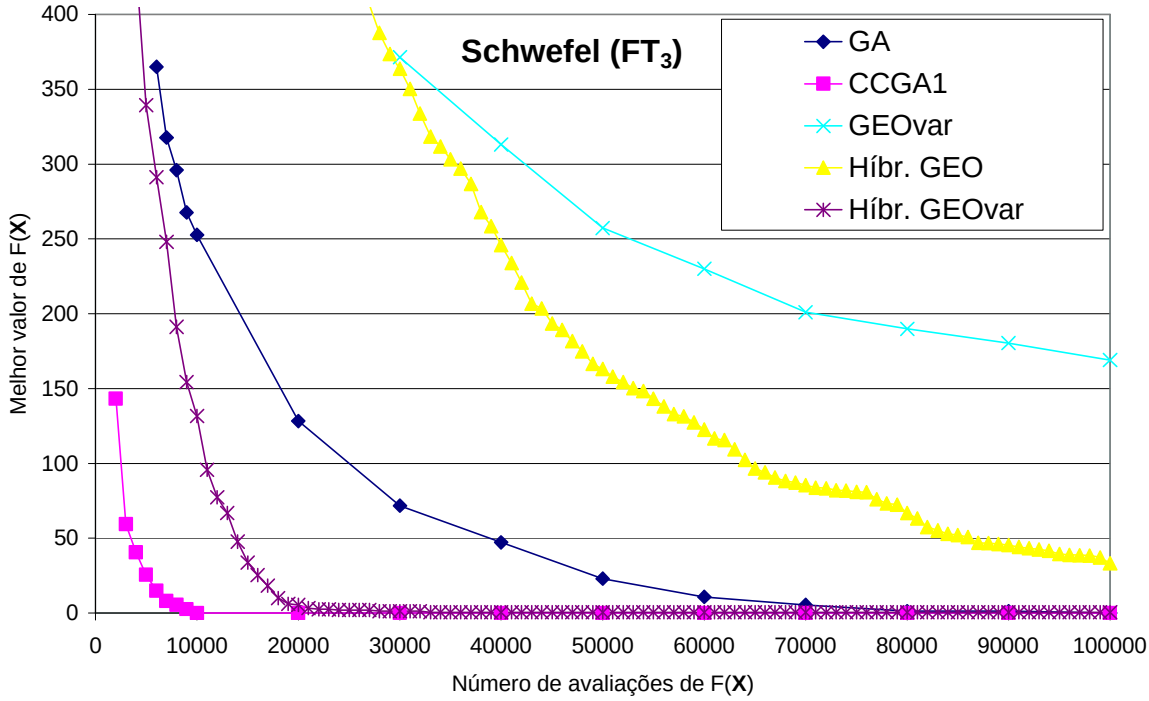


Figura 5.18 – Comparação dos resultados da segunda versão do híbrido de GEO e GEO_{var} + RS com os obtidos pelo CCGA e pelo GEO e GEO_{var} originais para a FT_3.

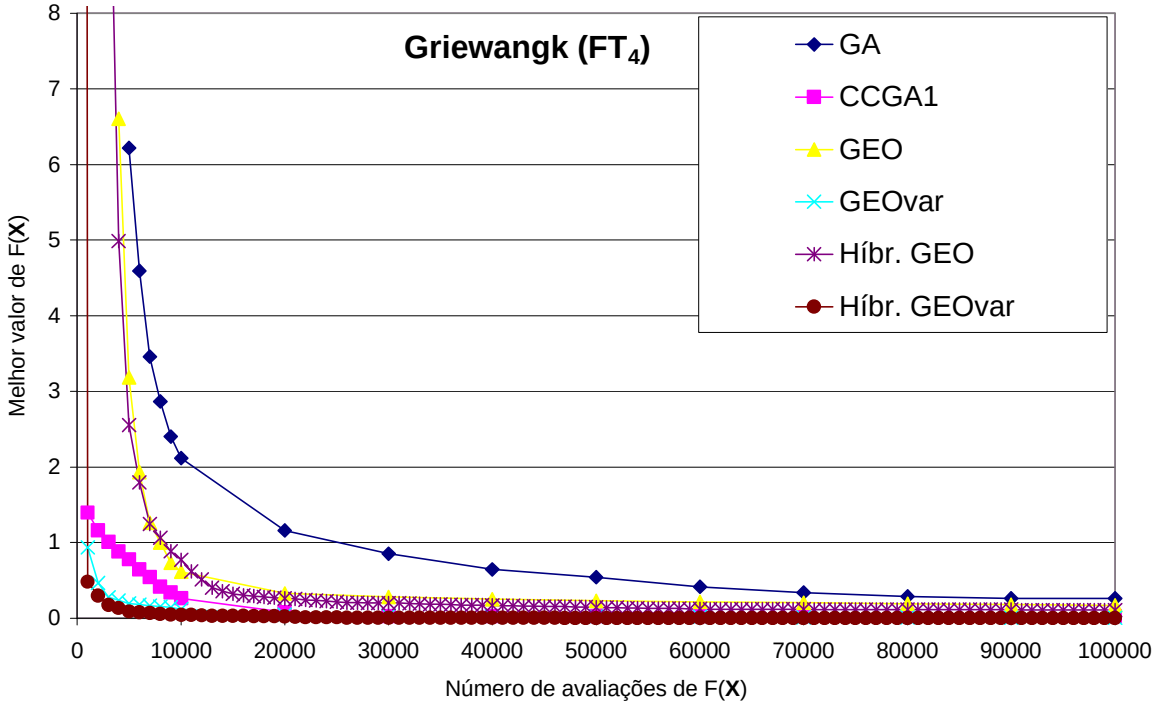


Figura 5.19 – Comparação dos resultados da segunda versão do híbrido de GEO e GEO_{var} + RS com os obtidos pelo CCGA e pelo GEO e GEO_{var} originais para a FT_4.

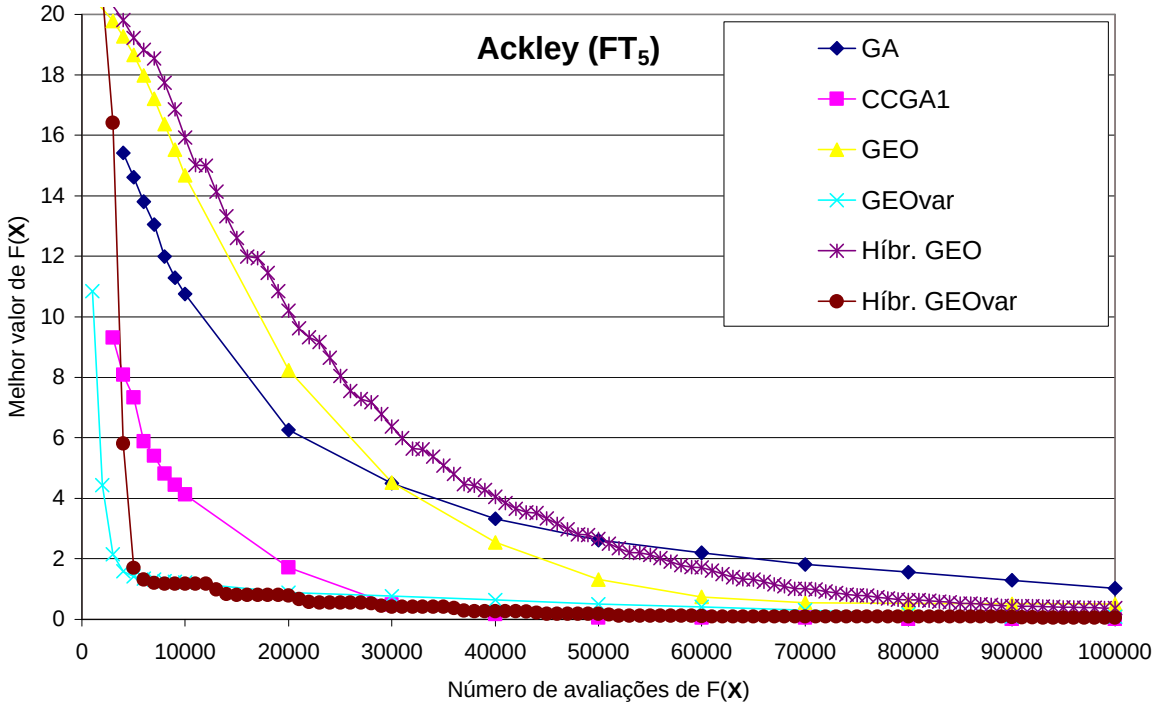


Figura 5.20 – Comparação dos resultados da segunda versão do híbrido de GEO e GEO_{var} + RS com os obtidos pelo CCGA e pelo GEO e GEO_{var} originais para a FT_5.

A análise das figuras revela que a versão híbrida de GEO e GEO_{var} com o RS obteve ganhos significativos de desempenho com FT_1, FT_2, e FT_3. Para FT_4, não houve ganho significativo e, para FT_5, além de não ter havido ganho significativo com o híbrido de GEO_{var}, o híbrido de GEO apresentou perda de desempenho com relação ao GEO original. No balanço geral, a versão híbrida obteve o melhor desempenho dentre todos apenas para a FT_4 e obteve o segundo melhor desempenho para a FT_5, sempre com o híbrido de GEO_{var}. Em ambos os casos, no entanto, talvez seja mais correto dizer que houve empate entre GEO_{var} e seu híbrido.

5.4.2 – GEO + método das Estratégias Evolutivas

A segunda versão híbrida consiste em hibridizar GEO com o método das Estratégias Evolutivas (EEs) (em inglês, *ES - Evolution Strategies*). EE é uma técnica de otimização baseada nas idéias de adaptação e evolução, sendo mais um membro da família dos Algoritmos Evolutivos (AEs) (Eiben e Smith, 2003). EEs usam vetores com codificação real (não binária) e principalmente mutação, entre outros, como operadores. Como é comum em AEs, os operadores são aplicados em ordem: recombinação, mutação,

avaliação da função de adaptação e seleção natural. A aplicação deste laço uma vez é chamada de uma geração, continuando por gerações até que um critério de parada é atingido.

Os primeiros EEs baseavam-se em uma população de apenas dois indivíduos a cada geração: um ponto no espaço de busca (o pai) e uma mutação deste (o filho), sendo usada a notação (1+1)-EE. Versões mais atuais empregam populações de pais e filhos, sendo que a notação ($\mu+\lambda$)-EE indica que ambas as populações participam da seleção natural e a notação (μ,λ)-EE indica que apenas a população dos filhos, λ , participa da seleção natural. A pressão de seleção dos EEs é bastante alta, pois, tipicamente, $\lambda>>\mu$. A mutação, em sua versão mais simples, é obtida pela adição de valores procedentes de uma mesma distribuição gaussiana a cada componente do vetor de variáveis de projeto. O tamanho ou força desta mutação, isto é, o desvio padrão da distribuição gaussiana, usualmente varia durante a busca, evoluindo juntamente com as variáveis de projeto, num processo conhecido como auto-adaptação. A auto-adaptação talvez seja a principal contribuição dos EEs para a área de otimização.

Uma versão de mutação com EE, denominada mutação não correlacionada de tamanho único, utiliza uma única distribuição gaussiana, descrita por gauss(0,σ) = σ·gauss(0,1), como fonte de perturbação de tamanho σ para todas os N componentes do vetor $\mathbf{X}$ contendo as N variáveis de projeto. Sendo assim, cada vetor $\mathbf{X}$ gerado pelo operador de mutação possui um σ associado. O parâmetro σ, por sua vez, é considerado como uma variável adicional pelo EE. Assim, cada indivíduo do EE é formado pelo conjunto {$\mathbf{X}$,σ}, sendo que σ é mutado pela sua multiplicação por uma variável com distribuição lognormal com média 0 e desvio padrão α. Desta forma:

$$s^{n} = s^{n-1} \cdot e^{gauss(0,a)} \tag{5.12}$$

$$X_j^{n} = X_j^{n-1} + gauss_j(0, s^{n}) \tag{5.13}$$

onde os sobrescritos n-1 e n são utilizados apenas para sinalizar valor antes da mutação e após a mutação, respectivamente. Na equação 5.13, o subscrito j da função gauss(.) indica que cada variável de projeto sofre uma mutação diferente, embora proveniente da mesma

variável aleatória (mesma distribuição de probabilidade). É importante notar que, embora cada indivíduo seja um par $\{\mathbf{X},\sigma\}$, a função de adaptação utilizada no processo de seleção é a função objetivo, $F(\mathbf{X})$, do problema de otimização originalmente formulado e não uma $F(\{\mathbf{X},\sigma\})$. O mecanismo de mutação descrito pelas equações 5.12 e 5.13 gera uma auto-adaptação do tamanho da mutação σ, visto que o processo de seleção atua, a cada geração, sobre vetores $\mathbf{X}$ resultantes de mutações efetuadas sob diversos σ's, mas apenas os melhores vetores $\mathbf{X}$ sobrevivem, carregando consigo seus respectivos σ's. O parâmetro a da equação 5.11 torna-se, portanto, o único parâmetro livre desta versão do método EE. Esse parâmetro é comumente chamado de taxa de aprendizado (do EE em relação à $F(\mathbf{X})$). Basicamente, o parâmetro a define quão rapidamente o valor de σ pode variar a cada geração (iteração) do EE. Valores elevados para a sinalizam que o EE pode realizar a busca passando rapidamente de uma busca global (σ's elevados) para uma busca local (σs's pequenos) e vice-versa.

De acordo com Eiben e Smith (2003), as características essenciais presentes no método EE são:

1) EEs em geral são utilizados para otimização de problemas com variáveis de projeto contínuas (reais);

2) Mutação é o principal operador utilizado para gerar novos indivíduos;

3) Mutação das variáveis de projeto é implementada pela adição de ruído proveniente de distribuição gaussiana;

4) Os parâmetros que controlam a mutação são modificados ao longo da execução;

Todas as versões de GEO (incluso GEO_{var}) descritas ou desenvolvidas nesta Tese utilizam apenas mutações para gerar novos indivíduos (pontos no espaço de busca). Portanto, GEO e EE têm esta característica em comum. Já o item 1 da lista de características essenciais do método EE só é compartilhada pela versão SA4 do GEO, desenvolvida no Capítulo 3, seção 3.7.4, que, assim como EE, utiliza variáveis contínuas (reais) diretamente. Este fato motivou a escolha da versão SA4 para o desenvolvimento do híbrido GEO + EE. Dentre as variantes da SA4 testadas na seção 3.7.4 do Capítulo 3,

decidiu-se utilizar a SA4 da terceira etapa. O motivo principal dessa escolha foi a maior robustez demonstrada pela SA4 da terceira etapa, traduzida, nos resultados dos testes, por uma menor sensibilidade aos parâmetros de ajuste.

No EE, o parâmetro σ define o tamanho das mutações que afetam **X** e, portanto, define também a localidade da busca. No GEO versão SA4, a localidade da busca é definida pelo parâmetro b. Assim, por analogia ao EE, a idéia é aplicar ao parâmetro b um mecanismo de variação similar àquele utilizado no EE para o parâmetro σ. Entretanto, depois de testes preliminares, observou-se que a melhor maneira de mutar b não é pela multiplicação por uma variável aleatória com distribuição lognormal, do tipo indicado na equação 5.12, e sim pela adição de uma perturbação com distribuição gaussiana. Assim:

$$b^n = b^{n-1} + \text{gauss}(\mu, a) \tag{5.14}$$

onde os sobrescritos n-1 e n sinalizam valor antes da mutação e após a mutação, respectivamente.

Além da taxa de aprendizado, a, a equação 5.14 apresenta ainda o parâmetro μ, que é a média da distribuição gaussiana usada para mutar b. Ressalta-se que o parâmetro μ da equação 5.14 nada tem a ver com o μ tradicionalmente usado nos EEs para definir o número de indivíduos "pais" da população. Aqui, esse parâmetro sinaliza a tendenciosidade imposta na mutação de b. Se μ=0, não há tendenciosidade. Imaginando uma busca que comece com $b^0 = b_{MIN}$ (>1), então a utilização de μ>0 de valor não desprezível gera, ao longo de muitas iterações, um cronograma de valores crescentes de b, ainda que não monotônicos (exceto se a=0). Lembrando, da seção 3.7.4, que valores pequenos de b impõem busca esparsa e valores altos de b impõe busca local, então um cronograma de valores crescentes para b significa uma busca que começa esparsa, quando $b \sim b_{MIN}$, e que acaba local, quando b>>1. A idéia por trás da utilização de μ é permitir uma maior flexibilidade de ajuste ao algoritmo e verificar se esta abordagem gera vantagens significativas, relativamente àquela preconizada pelo EE, ou seja, μ=0. Alguém poderia alegar, com razão, que a EE possui a vantagem de ter um parâmetro a menos para ajustar. Entretanto, esta possibilidade também existe para o híbrido GEO SA4 + EE, basta fazer μ=0. Na verdade, a possibilidade oposta, de fazer $\mu \neq 0$ no EE, também existe, só que praticamente nunca é empregada. O algoritmo híbrido pode ser empregado de dois modos, dependendo da disposição ou disponibilidade do

usuário: (i) com μ e α como parâmetros ou (ii) μ=0=constante e apenas α como parâmetro. Da mesma forma que α é chamado taxa de aprendizado, μ pode ser chamado de tendenciosidade do aprendizado.

A combinação de valores de a e μ pode gerar cronogramas de b com as seguintes características:

1) μ=0 e α=0 : cronograma com um único valor b=b_{MIN}=constante;

2) μ=0 e $\alpha \rightarrow 0$: cronograma não tendencioso com variabilidade baixa;

3) μ=0 e $\alpha >> 0$: cronograma não tendencioso com variabilidade alta;

4) μ>0 e α=0 : cronograma fixo dado por $b^n = b^{n-1} + \mu$;

5) $\mu \rightarrow 0$ e $\alpha \rightarrow 0$: cronograma com tendência crescente lenta e variabilidade baixa;

6) $\mu \rightarrow 0$ e $\alpha >> 0$: cronograma com tendência crescente lenta e variabilidade alta;

7) $\mu >> 0$ e $\alpha \rightarrow 0$: cronograma com tendência crescente rápida e variabilidade baixa;

8) $\mu >> 0$ e $\alpha >> 0$: cronograma com tendência crescente rápida e variabilidade alta;

Em relação à maneira de mutar **X**, o vetor das variáveis de projeto, a sistemática continua igual àquela utilizada pela SA4 da terceira etapa descrita na seção 3.7.4, ou seja, as magnitudes das l_j mutações de cada variável de projeto X_j começam por um valor igual a e_j e são incrementadas por um fator de escala multiplicativo igual a b. Os sinais (+ ou -) das l_j mutações são sucessivamente alternados, sendo que o sinal da menor delas, a de tamanho e_j, é obtido por sorteio com distribuição uniforme.

Uma mudança importante introduzida na versão híbrida, relativamente ao GEO SA4, é a utilização de seleção por elitismo ao invés de seleção estocástica com τ. A seleção por elitismo, no caso do GEO, implica apenas em selecionar o melhor dentre todos os L indivíduos gerados a cada iteração, não requerendo, portanto, nenhum parâmetro adicional. A consequência mais importante dessa mudança é a eliminação de τ como parâmetro de ajuste do algoritmo. Inicialmente, uma

versão híbrida que impunha também para τ mutações do tipo imposto ao parâmetro b (vide equação 5.14) foi implementada. No entanto, testes preliminares envolvendo as duas versões com as 5 FT's demonstraram que o desempenho de ambas era praticamente o mesmo. Sendo assim, optou-se pela versão mais simples, com o menor número de parâmetros.

Na seqüência, os passos do novo algoritmo são apresentados, sendo comentados nos parágrafos subseqüentes. O algoritmo híbrido de GEO SA4 + EE é definido conforme segue:

1. Inicialize aleatoriamente e com distribuição uniforme entre $\mathbf{X_{min}}$ e $\mathbf{X_{max}}$ um vetor $\mathbf{X}$ contendo as N variáveis de projeto. Calcule o valor da função objetivo F($\mathbf{X}$), faça $\mathbf{X_{melhor}} = \mathbf{X}$ e guarde F($\mathbf{X_{melhor}}$).
2. Defina o número de mutações l_j, $j \in \{1,2,...,N\}$ de cada variável X_j, tal que $\sum_j l_j = L$, sendo L número total de mutações agindo sobre as N variáveis de projeto. Defina valores para os parâmetros μ e α. Defina valores dos limites inferior, b_{MIN}, e superior, b_{MAX}, para a base, com $b_{MIN}>1$. Faça $b= b_{MIN}$.
3. Calcule o vetor $\mathbf{e}$, onde $e_j= (X_{MAX_j}- X_{MIN_j})/(b^{l_j}-1)$, e j é o índice da variável, isto é, $j \in \{1,2,...,N\}$. O elemento e_j define a resolução de mutação da j-ésima variável de projeto. Por resolução entenda-se o menor valor a ser somado ou subtraído de X_j.
4. Faça $F(\mathbf{X})_{ref}= F(\mathbf{X})$, $F(\mathbf{X_{melhor}})_{ref}= F(\mathbf{X_{melhor}})$.
5. Faça n=0. Para cada variável $j \in \{1,2,...,N\}$ do vetor $\mathbf{X}$, faça:
    a. Sorteie com distribuição uniforme um valor para $c \in \{0, 1\}$.
    b. Para cada mutação $i \in \{1,2,..., l_j\}$ da variável X_j, faça:
        1º. Incremente o número total de mutações do vetor $\mathbf{X}$: faça n= n+1.
        2º. Calcule o tamanho da mutação $m= b^{(i-1)} \cdot e_j$.
        3º. Calcule o sinal da mutação $s=(-1)^{(i-C)}$, onde c é o valor binário sorteado no passo 5.a.
        4º. Mute X_j. Primeiro, faça $X_{aux}= X_j$. Em seguida, faça $X_j= X_j+s \cdot m$, gerando um vetor mutado $\mathbf{X_n}$. Verifique os limites: Se $X_j > X_{MAX_j}$ ou

$X_j < X_{MIN_j}$, então sorteie um novo $X_j \in [X_{MIN_j}, X_{MAX_j}]$ com distribuição uniforme e faça m= abs(X_j-X_{aux}) e s=(X_j-X_{aux})/m.

5º. Calcule o valor da função objetivo F($\mathbf{X_n}$). Atribua à mutação n um valor de adaptação $\Delta F(\mathbf{X_n}) = F(\mathbf{X_n}) - F(\mathbf{X_{best}})_{ref}$, que indica o ganho ou a perda que a função objetivo tem se a mutação n ocorrer, quando comparada com o melhor valor da função objetivo encontrado até a iteração anterior. Em seguida, se $F(\mathbf{X_n}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}}) = F(\mathbf{X_n})$ e $\mathbf{X_{melhor}} = \mathbf{X_n}$.

6º. Retorne **X** à sua condição não mutada: faça $X_j = X_j - s \cdot m$.

6. Mute **X** para a configuração $\mathbf{X_n}$ que obteve o menor $\Delta F(\mathbf{X_n})$ no passo 5.b.

7. Verifique a adaptação da base: Calcule o novo valor de F(**X**). Se $F(\mathbf{X}) \geq F(\mathbf{X})_{ref}$ então faça $F(\mathbf{X}) = F(\mathbf{X_{pior}})$ e faça $b = b_{ref} - 20 \cdot y_{ant}$, onde y_{ant} é o valor de y da iteração anterior e $F(\mathbf{X_{pior}})$ é o maior valor de F(**X**) encontrado até agora. Do contrário, faça b_{ref}=b e b= b + y, com y=gauss(μ,α), onde gauss(μ,α) é o valor obtido de uma distribuição gaussiana com média μ e desvio padrão α. Verifique os limites da base: Se $b > b_{MAX}$, faça $b = b_{MAX}$. Se $b < b_{MIN}$, faça $b = b_{MIN}$.

8. Repita os passos 3 a 7 até que um dado critério de parada seja satisfeito.

9. Retorne $\mathbf{X_{melhor}}$ e $F(\mathbf{X_{melhor}})$.

Com relação aos passos do algoritmo recém exposto, os comentários a seguir podem ser úteis.

No passo 2, o número de mutações é definido para cada variável de projeto. Em muitos problemas, é possível e também conveniente usar o mesmo número de mutações para todas as variáveis de projeto, tal que $l_j = l$ = constante, para $j \in \{1,2,...,N\}$. Se não for este o caso, o valor l_j de cada variável j é atribuído, gerando uma lista da forma $l=\{l_1,l_2,\ldots,l_N\}$ e tal que o número total de mutações é dado por $L = \sum l_j$.

No passo 5.b, existe um laço de repetição de $i \in \{1,2,..., l_j\}$ para calcular as l_j mutações de cada variável de projeto.

Dentro do laço, em 5.b.2, o tamanho m da mutação é definido, tal que, para i=1, 2, 3, …, l_j, o respectivo tamanho é m= e_j, $b \cdot e_j$, $b^2 \cdot e_j$, …, $b^{(l_j-1)} \cdot e_j$. Como visto, as magnitudes das mutações iniciam em e_j para i=1 e são escalonadas, de um tamanho para o próximo, por uma fator de valor b. Portanto, o índice i pode ser interpretado como uma espécie de definidor da ordem de magnitude da mutação, tal que existem mutações com magnitudes de 1a., 2a., 3a., …, l_j-ésima ordem. Neste sentido, **e** pode ser visto como o vetor das mutações de primeira ordem de magnitude.

Em 5.b.3, o sinal de cada mutação é obtido. Esta meta é atingida por alternância de sinal, isto é, cada mutação tem sinal oposto ao sinal da mutação anterior. O sinal da primeira mutação é escolhido aleatoriamente com o auxílio de c (vide passo 5.a). Em testes preliminares, uma escolha de sinal totalmente aleatória também foi testada, mas os resultados obtidos foram ligeiramente piores do que aqueles obtidos com a escolha de sinal feita conforme definido em 5.b.3. Uma vez que somente 5 funções teste foram usadas, os resultados não podem ser considerados como conclusivos com relação a qual método de escolha dos sinais das mutações é superior. Uma busca exaustiva sobre qual método é superior, ou ainda, sobre outros métodos possíveis, está além do escopo desta Tese.

Da mesma forma que ocorre com a SA4, o novo híbrido de GEO faz com que qualquer mutação maior do que 100% de X_{MAX_j}-X_{MIN_j} seja substituída, necessariamente, por uma nova mutação aleatória, escolhida sobre a totalidade do espaço de busca da respectiva variável de projeto (vide passo 5.b.4). Mutações menores do que 100% em tamanho também podem levar a mutações aleatórias, dependendo de quão próxima da borda do espaço de busca está a variável de projeto antes da mutação e do sinal da mutação. Obviamente, dentre as mutações menores do que 100%, quanto maior o tamanho da mutação, maior a chance dela acarretar mutação aleatória.

No passo 7, um novo valor de b é escolhido. Se o valor de F(**X**) ao final da iteração atual é menor do que o valor de F(**X**) ao final da iteração anterior, $F(\mathbf{X})_{ref}$, então b é mutado estocasticamente, ou seja, b= b + y, com y=gauss(μ,α). Senão, b é mutado deterministicamente, b= b_{ref}-20·y_{ant}, onde y_{ant} é o valor de y da iteração anterior. A estratégia contida nesta última equação foi, dentre as várias testadas, aquela que obteve o melhor desempenho. Ela promove uma espécie de efeito de rebatimento no valor de b. Quando $y_{ant} > 0$, o valor de b sofre uma redução igual a

$20 \cdot y_{ant}$. Por outro lado, quando $y_{ant} > 0$, o valor de b sofre um acréscimo igual a $20 \cdot y_{ant}$. A idéia por trás desta estratégia é a seguinte: Dado que o valor de b da iteração atual não ocasionou melhoria em F(X), e sabendo que ele foi obtido somando-se y_{ant} ao b da iteração anterior (=b_{ref} da iteração atual), então a parcela "$-20 \cdot y_{ant}$" tem por função inverter o sentido e amplificar 20 vezes o passo aplicado em b na iteração anterior e aplicá-lo ao b da iteração atual.

No caso do híbrido de GEO_{var} SA4 + EE, N mutações simultâneas são efetuadas a cada iteração do algoritmo, sendo uma mutação por variável, ao invés de uma mutação por iteração. Neste caso, a mutação que ocasiona a maior diminuição em F(**X**) é escolhida separadamente para cada variável de projeto. Ao final de cada iteração, as mutações são efetuadas ao mesmo tempo em todas as variáveis de projeto, levando a um novo vetor **X**.

Os passos da versão híbrida de GEO_{var} SA4 + EE são dados por:

1. Inicialize aleatoriamente e com distribuição uniforme entre $\mathbf{X_{min}}$ e $\mathbf{X_{max}}$ um vetor **X** contendo as N variáveis de projeto. Calcule o valor da função objetivo F(**X**), faça $\mathbf{X_{melhor}} = \mathbf{X}$ e guarde F($\mathbf{X_{melhor}}$).
2. Defina o número de mutações l_j, $j \in \{1,2,...,N\}$ de cada variável X_j, tal que $\sum_j l_j = L$, sendo L número total de mutações agindo sobre as N variáveis de projeto. Defina valores para os parâmetros μ e α. Defina valores dos limites inferior, b_{MIN}, e superior, b_{MAX}, para a base, com $b_{MIN}>1$. Faça $b= b_{MIN}$.
3. Calcule o vetor **e**, onde $e_j= (X_{MAX_j}- X_{MIN_j})/(b^{l_j}-1)$ e j é o índice da variável, isto é, $j \in \{1,2,...,N\}$. O elemento e_j define a resolução de mutação da j-ésima variável de projeto. Por resolução entenda-se o menor valor a ser somado ou subtraído de X_j.
4. Faça $F(\mathbf{X})_{ref}= F(\mathbf{X})$, $F(\mathbf{X_{melhor}})_{ref}= F(\mathbf{X_{melhor}})$.
5. Para cada variável $j \in \{1,2,...,N\}$ do vetor **X**, faça:
    a. Sorteie com distribuição uniforme um valor para $c \in \{0, 1\}$.
    b. Para cada mutação $i \in \{1,2,..., l_j\}$ da variável X_j, faça:
        1º. Calcule o tamanho da mutação $m= b^{(i-1)} \cdot e_j$.

2º. Calcule o sinal da mutação $s=(-1)^{(i-C)}$, onde c é o valor binário sorteado no passo 5.a.

3º. Mute X_j. Primeiro, faça $X_{aux}= X_j$. Em seguida, faça $X_j= X_j+s\cdot m$, gerando um vetor mutado $\mathbf{X_i}$. Verifique os limites: Se $X_j > X_{MAX_j}$ ou $X_j < X_{MIN_j}$, então sorteie um novo $X_j \in [X_{MIN_j} , X_{MAX_j}]$ com distribuição uniforme e faça $m= abs(X_j-X_{aux})$ e $s=(X_j-X_{aux})/m$.

4º. Calcule o valor da função objetivo $F(\mathbf{X_i})$. Atribua à mutação i um valor de adaptação $\Delta F(\mathbf{X_i})= F(\mathbf{X_i}) - F(\mathbf{X_{melhor}})_{ref}$, que indica o ganho ou a perda que a função objetivo tem se a mutação i ocorrer, quando comparada com o melhor valor da função objetivo encontrado até a iteração anterior. Em seguida, se $F(\mathbf{X_i}) < F(\mathbf{X_{melhor}})$ então faça $F(\mathbf{X_{melhor}})= F(\mathbf{X_i})$ e $\mathbf{X_{melhor}} = \mathbf{X}_i$.

5º. Retorne **X** à sua condição não mutada: faça $X_j= X_j-s\cdot m$.

c. Escolha, dentre as l_j mutações geradas no passo 5.b, aquela que obteve o menor $\Delta F(\mathbf{X_i})$. Guarde o respectivo X_j no j-ésimo elemento do vetor $\mathbf{X_{esc}}$.

6. Efetue as mutações da iteração atual: faça $\mathbf{X} = \mathbf{X_{esc}}$, onde $\mathbf{X_{esc}}$ é o vetor resultante da execução do passo 5.c para cada variável $j \in \{1,2,...,N\}$.

7. Verifique a adaptação da base: Calcule o novo valor de $F(\mathbf{X})$. Se $F(\mathbf{X}) \geq F(\mathbf{X})_{ref}$ então faça $F(\mathbf{X})= F(\mathbf{X_{pior}})$ e faça $b= b_{ref}-20\cdot y_{ant}$, onde y_{ant} é o valor de y da iteração anterior e $F(\mathbf{X_{pior}})$ é o maior valor de $F(\mathbf{X})$ encontrado até agora. Do contrário, faça $b_{ref}=b$ e $b= b + y$, com $y=gauss(\mu,\sigma)$, onde $gauss(\mu,\alpha)$ é o valor obtido de uma distribuição gaussiana com média μ e desvio padrão α. Verifique os limites da base: Se $b > b_{MAX}$, faça $b= b_{MAX}$. Se $b < b_{MIN}$, faça $b= b_{MIN}$.

8. Repita os passos 3 a 7 até que um dado critério de parada seja satisfeito.

9. Retorne $\mathbf{X_{melhor}}$ e $F(\mathbf{X_{melhor}})$.

Quando o híbrido GEO/GEO_{var} SA4 + EE é comparado ao GEO/GEOvar canônico, as diferenças principais podem ser sumarizadas como:

- Vetor de variáveis reais ao invés de cadeia binária é usada para representar as

variáveis de projeto;

- Mutações sobre as variáveis de projeto são feitas por magnitudes (base qualquer) ao invés de bits (base 2);
- O número de mutações é dissociado da precisão numérica das variáveis de projeto;
- Ausência do parâmetro τ: Seleção é feita por elitismo, ao invés de estocasticamente;
- Adaptatividade da base, b, que controla ambos a localidade e a estocasticidade da busca;
- Existência de três parâmetros de ajuste: α, a taxa de aprendizado, μ, a tendenciosidade do aprendizado, e l_j, o número de mutações;

A rigor, o número de mutações, l_j, representa N parâmetros. Entretanto, na maioria dos casos, um mesmo valor $l_j=l$ é usado para todas as N variáveis. Além disso, algumas premissas podem ser úteis para se estimar o valor de l ou l_j.

Premissas:

1) O número total de mutações por iteração, L, deve ser suficiente para obter-se amostragem estatisticamente válida ou adequada;

2) A resolução ou precisão r_j necessária a cada variável de projeto é conhecida;

3) Quando $b=b_{MAX}$, o tamanho da menor mutação, e_j, deve ser menor do que a resolução requerida para a variável de projeto j;

Reescrevendo-se a equação 3.6 (vide Capítulo 3) como:

$$e_j = r_j \cdot (X_{MAX_j} - X_{MIN_j}), \quad \text{com } r_j \equiv \frac{1}{\left(b^{l_j} - 1\right)} \tag{5.16}$$

onde r_j é a resolução resultante, em fração de X_{MAX_j}-X_{MIN_j}, para uma base de valor b e l_j mutações. A resolução máxima é obtida com $r_{j_{MAX}}=\min(r_j)$, ou seja:

$$r_{j_{MAX}} = \frac{1}{b_{MAX}^{l_j} - 1} \tag{5.17}$$

Isolando-se l_j na equação 5.17, obtém-se:

$$l_j = \frac{\ln\left(\frac{1}{r_{j_{MAX}}} + 1\right)}{\ln(b_{MAX})} \tag{5.18}$$

Como l_j deve ser inteiro, tem-se que o valor mínimo de l_j é dado por:

$$l_{j_{MIN}} = \mathrm{INT}\left(\frac{\ln(\frac{1}{r_{j_{MAX}}} + 1)}{\ln(b_{MAX})} + 0{,}5\right) \tag{5.19}$$

A Figura 5.21 a seguir apresenta curvas $l_{j_{MIN}}$ *versus* b_{MAX} para vários valores de $r_{j_{MAX}}$, calculadas a partir da equação 5.19. Como pode ser visto na figura, para uma mesma resolução máxima, várias combinações ($l_{j_{MIN}}$, b_{MAX}) são possíveis, sendo que, quanto maior o valor de b_{MAX}, menor o valor de $l_{j_{MIN}}$. Olhando-se, por exemplo, a curva que estabelece uma resolução máxima igual a um bilionésimo de X_{MAX_j}-X_{MIN_j}, ou seja, $r_{j_{MAX}}=10^{-9}$, vê-se que para b_{MAX}=200, é possível obter-se esta resolução com apenas 4 mutações por variável de projeto, ou seja, $l_{j_{MIN}}$=4. Já para b_{MAX}=1000, o número mínimo de mutações cai para $l_{j_{MIN}}$=3.

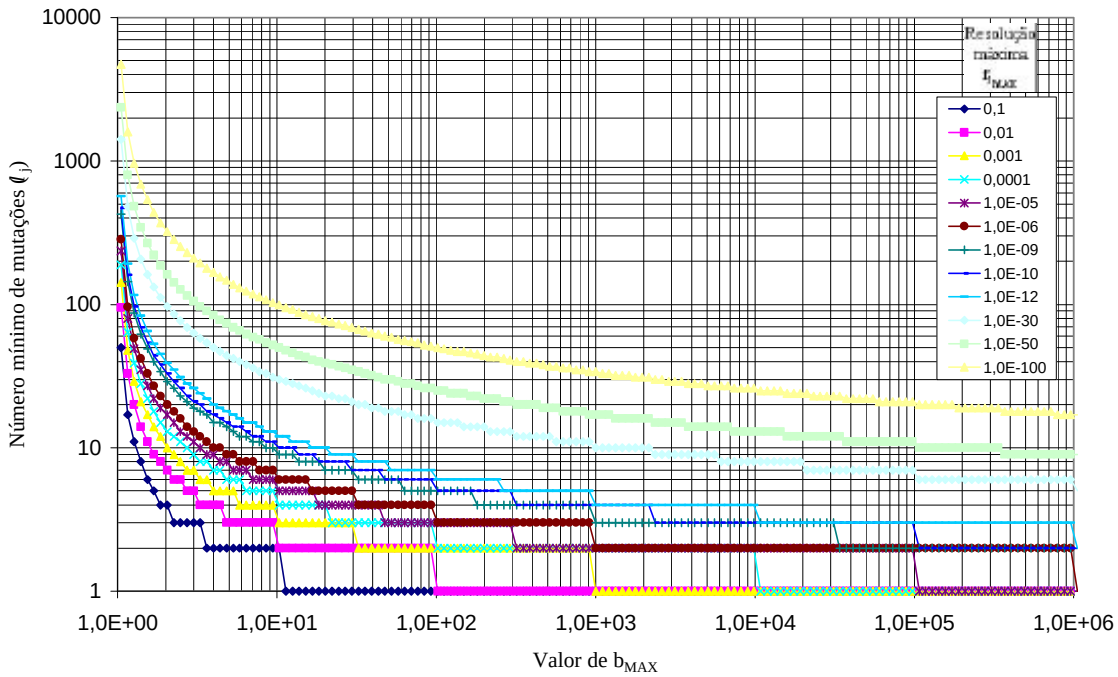


Figura 5.21 – Curvas $l_{j_{MIN}}$ *versus* b_{MAX} para vários valores de $r_{j_{MAX}}$.

O resultado mais importante da Figura 5.21 é que ela mostra, visualmente, que uma mesma resolução pode ser obtida para diversos pares de valores ($l_{j_{MIN}}$, b_{MAX}). Portanto, como b_{MAX} é um limite que também deve ser arbitrado, uma equação adicional é necessária para eliminar o indeterminismo presente na equação 5.19.

Ao lidar com populações de amostras (realizações de um processo estocástico), em geral, dezenas de amostras são necessárias a fim de que informações estatisticamente válidas a respeito da população possam ser obtidas. No âmbito dos AE's, a cada geração ou iteração deste tipo de algoritmos, uma população de indivíduos é gerada. Os indivíduos são possíveis soluções do problema de otimização e, ao mesmo tempo, são amostras de um processo estocástico, o processo de otimização. Em sendo amostras, estão sujeitas às mesmas regras que qualquer população de amostras. Sendo assim, um número mínimo de indivíduos deve existir a cada geração, para que o processo de seleção atue sobre um conjunto com validade estatística e não invalide o processo como um todo. Eiben et al. (1999), relata os estudos, de vários autores, para tentar determinar qual a melhor população a ser usada por AGs. Os intervalos [50,...,100], [30,...,80] e [20,...,30] foram obtidos. No caso do híbrido de GEO e GEO_{var} + EE, a população de indivíduos por iteração é igual a $L = \sum_{j=1}^{N} l_j$, o número total de mutações aplicadas às variáveis de projeto. Deste modo, ao estabelecer um limite mínimo para L, do tipo $L \geq L_{MIN}$, então o mesmo, no limite da igualdade, fornece a equação adicional para definir os valores de $l_{j_{MIN}}$ e, por meio da equação 5.19, de b_{MAX}.

$$\sum_{j=1}^{N} l_{j_{MIN}} = L_{MIN} \tag{5.20}$$

Se o mesmo número de mutações, l_{MIN}, for imposto a todas as N variáveis, tem-se:

$$l_{MIN} \cdot N \geq L_{MIN} \quad \Rightarrow \quad l_{MIN} = INT\left(\frac{L_{MIN}}{N} + 0{,}5 \right) \tag{5.21}$$

Reescrevendo a equação 5.17, obtém-se:

$$b_{MAX} = \sqrt[l_j]{1 + \frac{1}{r_{j_{MAX}}}} \tag{5.22}$$

Considerando que o valor de b_{MAX} é um só e que a equação 5.22 fornece, no caso geral, N valores diferentes para b_{MAX}, então o valor escolhido para b_{MAX} deve ser o maior deles, ou seja:

$$b_{MAX} = \max\left(\sqrt[l_j]{1+\frac{1}{r_{j_{MAX}}}}\right) = \sqrt[\min(l_j)]{1+\frac{1}{\min(r_{j_{MAX}})}} \tag{5.23}$$

Denominando r_{MAX} a maior resolução de todas as N $r_{j_{MAX}}$ existentes, tal que $r_{MAX}=\min(r_{j_{MAX}})$, então a equação 5.23 torna-se:

$$b_{MAX} = \sqrt[\min(l_j)]{1+\frac{1}{r_{MAX}}} \tag{5.24}$$

Considerando agora que o mesmo número de mutações l_{MIN} dado pela equação 5.21 seja aplicado a todas as N variáveis de projeto, então a equação 5.24 torna-se:

$$b_{MAX} = \sqrt[(l_{MIN})]{1+\frac{1}{r_{MAX}}} \tag{5.25}$$

Resumindo o equacionamento recém exposto, uma vez definido o valor de L_{MIN}, calcula-se l_{MIN} com o auxílio da equação 5.21 e, sabendo-se qual o valor de r_{MAX}, calcula-se qual o valor de b_{MAX} com o auxílio da equação 5.25. Imaginando-se, por exemplo, um problema de otimização com N=3, r_{MAX}=0,0001 e estabelecendo L_{MIN}=20, então, de 5.21 l_{MIN}=7 e, de 5.25, b_{MAX}=3,73. O problema seria então implementado com 7 mutações por variável de projeto, totalizando L=21 mutações por iteração e com a base variando até um valor máximo igual a 3,73. Considerando o mesmo problema, mas agora com $r_{MAX}=10^{-10}$, o valor l_{MIN}=7 continua inalterado e o valor de b_{MAX} sobe para b_{MAX}=19,31.

Acredita-se que a forma de estimar valores para l_j e b_{MAX} apresentada seja útil principalmente nas aplicações práticas, oriundas de problemas do mundo real, tendo sido esta a principal razão para incluí-la no texto desta Tese. No caso de funções teste, criadas analiticamente, as variáveis de projeto, na maioria das vezes, não representam nenhuma grandeza física. Com isso, dados como a resolução máxima a ser empregada carecem de sentido e/ou são dados de difícil obtenção. Por esta razão, optou-se por não utilizar

estimativas de l_j e b_{MAX}. nas avaliações de desempenho envolvendo funções teste a serem apresentadas. Ao invés disso, l_j foi mantido livre como parâmetro a ser ajustado e um valor suficientemente alto foi atribuído para b_{MAX}, a fim de garantir um leque apropriado de resoluções máximas.

Primeira Comparação

Na seqüência, os resultados preliminares da aplicação da versão híbrida de GEO e GEO_{var} + EE recém desenvolvida às funções teste FT_i, $i \in \{1,2,...,5\}$ são comparados diretamente com aqueles apresentados nos artigos de Potter e De Jong (1994) e de Sousa e Ramos (2002) com o auxílio das Figuras 5.22 a 5.26. Os resultados referem-se a uma versão anterior do híbrido de GEO e GEO_{var} + EE, na qual o parâmetro μ não existia, ou seja, $\mu=0$, a exemplo do que é feito com os algoritmos baseados no EE. A versão mais atual, que permite μ não nulo, foi aplicada às funções teste utilizadas em outros dois estudos e os resultados serão mostrados na seqüência. Os resultados apresentados nas Figuras 5.22 a 5.26, obtidos com a versão onde $\mu=0$, refletem o melhor ajuste obtido considerando-se os parâmetros α e l_j, conforme Tabela 5.3. Um mesmo número de mutações $l_j=l$ foi utilizado para todas as variáveis de projeto. Os limites para a base foram fixados em $b_{MIN}=1,05$ e $b_{MAX}=120$. As curvas mostradas para as versões híbridas são resultados médios de 50 execuções independentes.

Tabela 5.3 – Valores dos parâmetros α e l.

FT	GEO híbrido		GEO_{var} híbrido	
	α	l	α	l
FT_1	0,001	2	0,001	4
FT_2	0,15	4	0,1	10
FT_3	0,15	16	0,02	17
FT_4	0,05	5	8,5	3
FT_5	0,25	4	0,1	16

Como pode ser visto na Tabela 5.3, os valores da taxa de aprendizado α para o híbrido de GEO_{var} SA4 foram sempre iguais ou menores do que os valores de α para o híbrido de GEO SA4. A única exceção ocorreu para a FT_4. Neste caso, o valor de $\alpha=8,5$ utilizado para GEO_{var} é também uma exceção em relação aos valores baixos de α utilizados em todos os demais casos, onde α foi sempre menor ou igual a 0,25. O número de mutações por variável de projeto, l, variou bastante,

de 2 até 17, sendo que, para a mesma FT, *l* foi sempre maior com o híbrido de GEO_{var} do que com o híbrido de GEO. A única exceção foi, novamente, a FT_4.

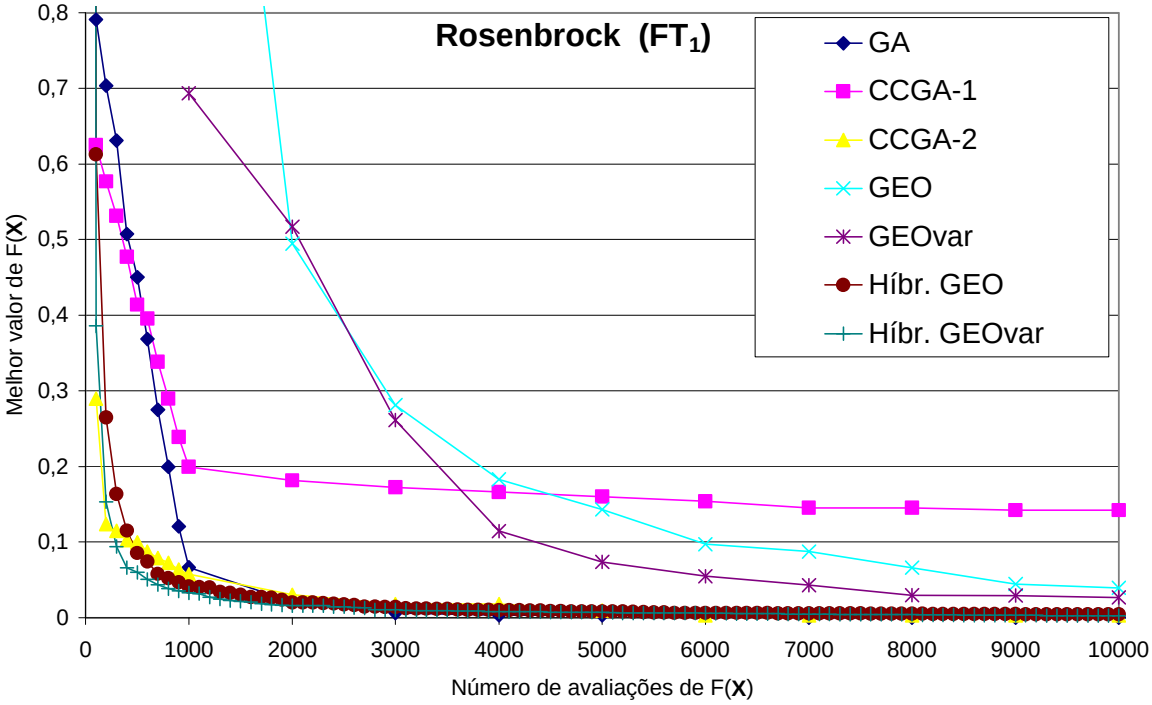


Figura 5.22 – Comparação dos resultados do híbrido de GEO e GEO_{var} + EE com os obtidos pelo CCGA e pelo GEO e GEO_{var} originais para a FT_1.

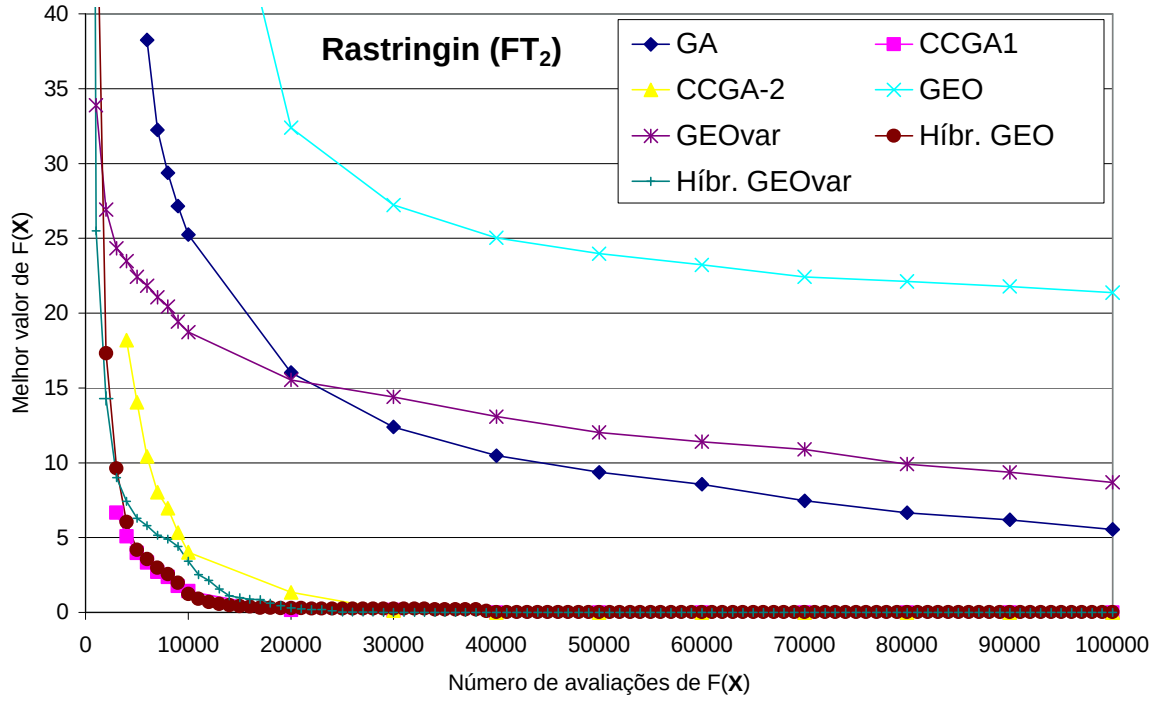


Figura 5.23 – Comparação dos resultados do híbrido de GEO e GEO_{var} + EE com os obtidos pelo CCGA e pelo GEO e GEO_{var} originais para a FT_2.

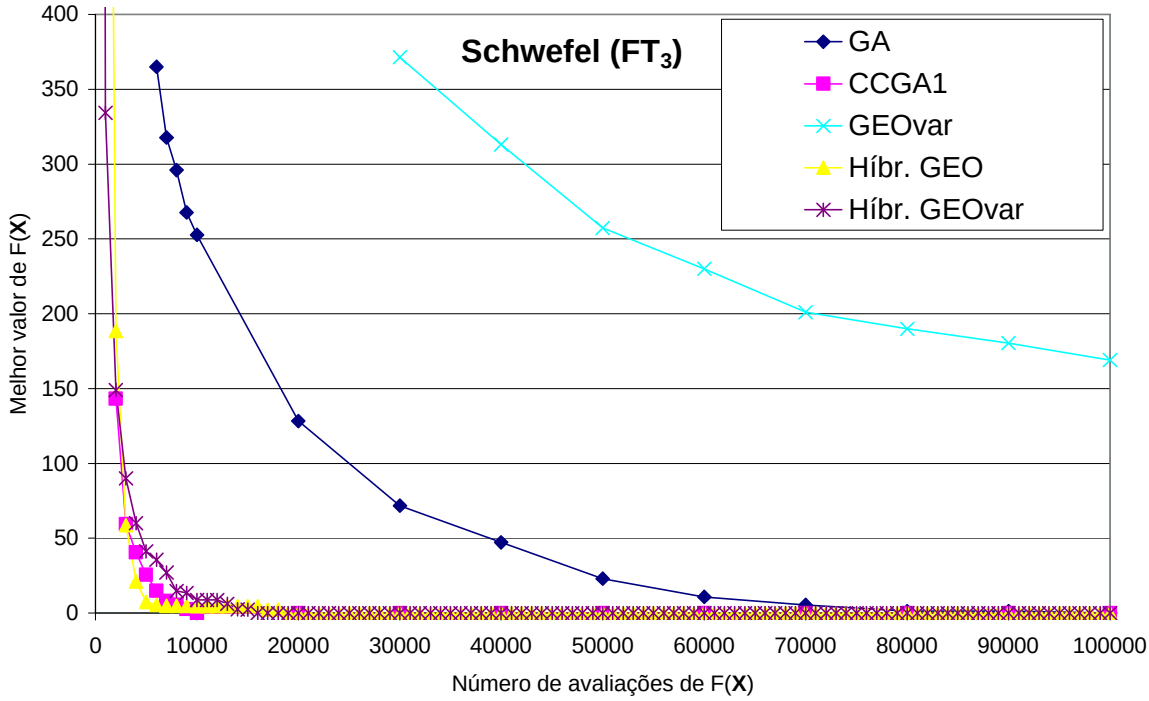


Figura 5.24 – Comparação dos resultados do híbrido de GEO e GEO_{var} + EE com os obtidos pelo CCGA e pelo GEO e GEO_{var} originais para a FT_3.

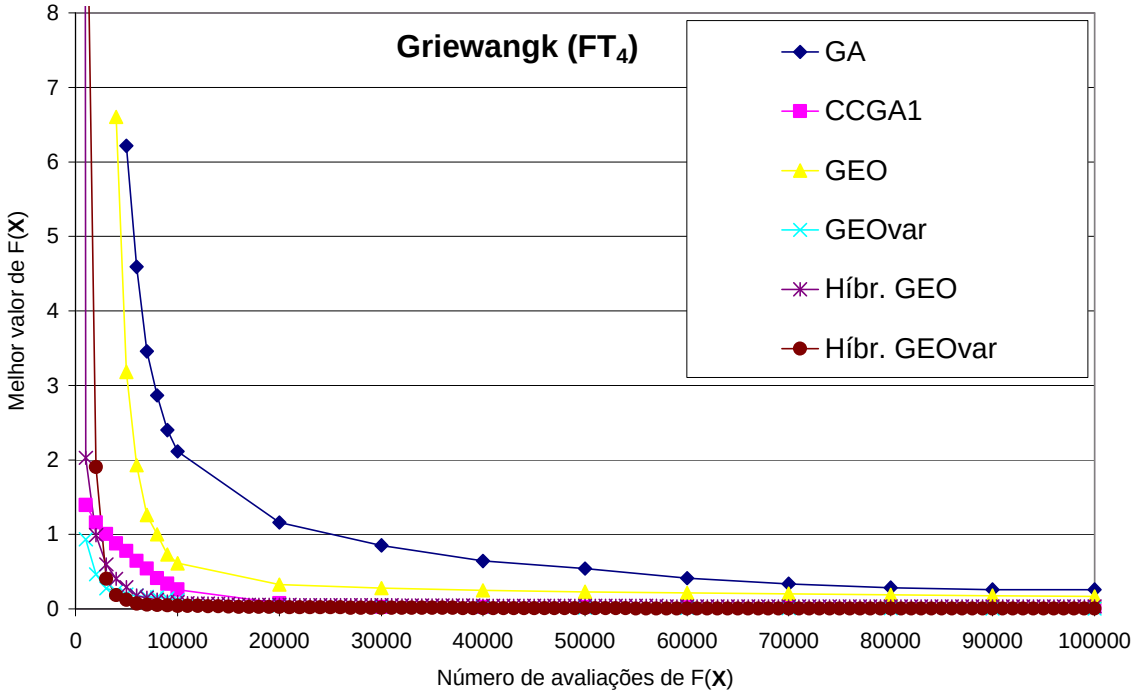


Figura 5.25 – Comparação dos resultados do híbrido de GEO e GEO_{var} + EE com os obtidos pelo CCGA e pelo GEO e GEO_{var} originais para a FT_4.

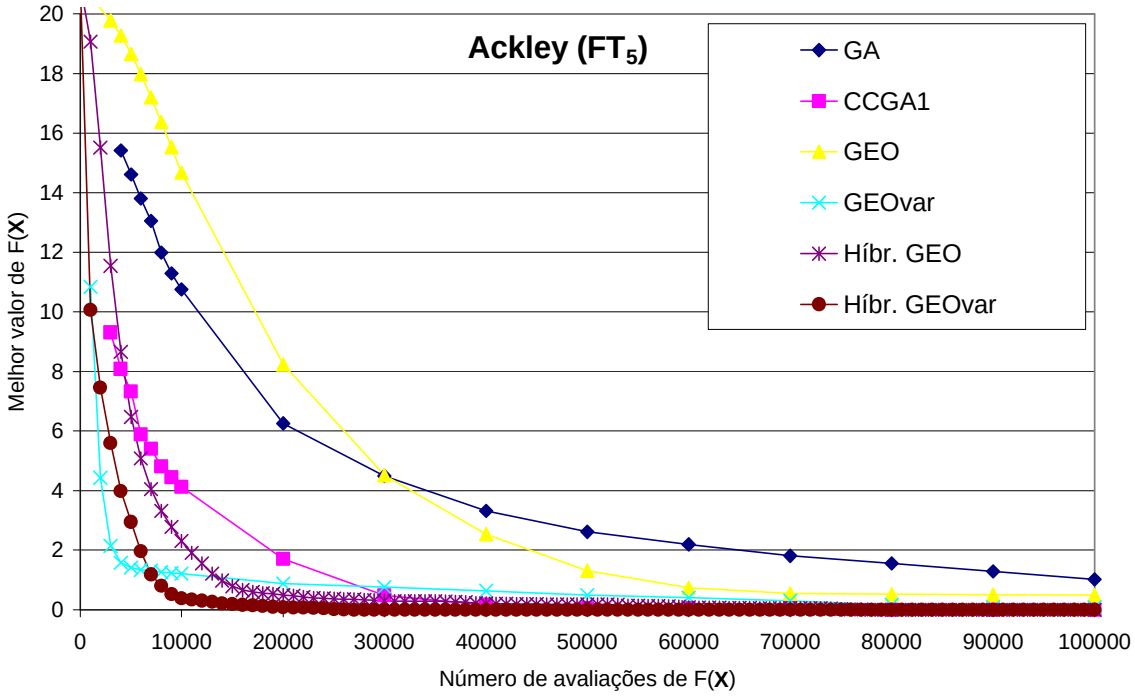


Figura 5.26 – Comparação dos resultados do híbrido de GEO e GEO_{var} + EE com os obtidos pelo CCGA e pelo GEO e GEO_{var} originais para a FT_5.

A análise das figuras revela que a versão híbrida de GEO e GEO_{var} SA4 com o EE, mesmo limitada com μ=0 fixo, obteve ganhos significativos de desempenho para todas as FT's. No balanço geral, a versão híbrida obteve o melhor desempenho dentre todos para FT_1, FT_3, FT_4, e FT_5. Além disso, obteve o segundo melhor desempenho para a FT_2. Na comparação entre os híbridos apenas, o placar foi de 3 a 2 a favor do híbrido de GEO_{var}. O híbrido de GEO_{var} foi melhor para FT_1, FT_4, e FT_5, enquanto o híbrido de GEO foi melhor para FT_2 e FT_3.

Os resultados obtidos pelos híbridos de GEO e GEO_{var} SA4 com o EE são significativamente melhores do que aqueles obtidos pelos híbridos de GEO e GEO_{var} SA1 com o RS (vide seção 5.4.1). Este fato motivou a realização de mais testes envolvendo os híbridos de GEO e GEO_{var} SA4 com o EE. Dois estudos foram selecionados (Someya e Yamamura, 2001; Ballester e Carter, 2004). Nos dois estudos, novas versões de AGs com codificação real foram desenvolvidas e testadas.

Em Ali et al. (2005), num teste de desempenho envolvendo mais de 50 funções teste amplamente utilizadas para avaliação de algoritmos de otimização com variáveis contínuas e dimensionalidade variando de N=2 até N=20, comparou-se o desempenho de 5

algoritmos estocásticos de otimização global, utilizando-se o desempenho médio de 30 execuções independentes para cada função teste. No teste com critério de parada após $100 \cdot N^2$ avaliações de F(**X**), uma versão de AG com codificação real obteve o melhor resultado dentre todos em mais da metade das 56 funções teste utilizadas. Considerando o número de funções teste utilizado, este resultado é extremamente significativo e aponta o AG com codificação real como sendo um algoritmo de desempenho notável. Este fato torna ainda mais importante a comparação entre os híbridos de GEO e GEO_{var} SA4 + EE e as versões de AGs desenvolvidas nos artigos de Someya e Yamamura (2001) e Ballester e Carter (2004), pois ambos apresentam novas versões de AGs com codificação real.

Para as comparações que seguem, optou-se por utilizar apenas uma das versões híbridas desenvolvidas. O híbrido de GEO_{var} com EE foi escolhido, em função do escore favorável obtido no teste com as 5 FT's.

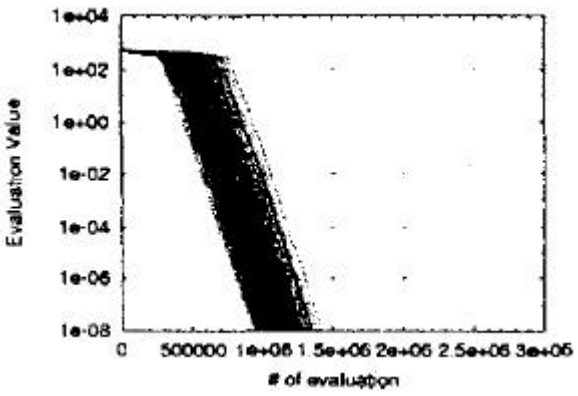
Segunda Comparação

Em Someya e Yamamura (2001), resultados gráficos interessantes são apresentados, relativamente às propriedades de convergência de um AG com codificação real: O AG com Adaptação da Área de Busca (em inglês, *Genetic Algorithm with Search Area Adaptation - GSA*). No referido artigo, o algoritmo GSA foi testado com o auxílio de 6 funções teste e obteve um desempenho favorável, quando comparado a uma versão de AG anterior, denominada UNDX+MGG (Ono e Kobayashi, 1997). De acordo com Someya e Yamamura (2001), UNDX+MGG mostrou desempenho superior ao de algoritmos baseados em EEs para funções teste multimodais ou com dimensionalidade elevada. A Tabela 5.4 define cada uma das 6 funções teste utilizadas em Someya e Yamamura (2001), incluindo o número N de variáveis de cada uma, as restrições laterais, a solução ótima $\mathbf{X}^*$ e o valor de $F(\mathbf{X}^*)$. Logo em seguida, a Figura 5.27, extraída de Someya e Yamamura (2001) e ampliada, reproduz os resultados (de todas as 100 execuções) do GSA para cada uma das 6 funções teste. O valor de N, o número de variáveis de projeto, está indicado entre parênteses, e varia de N=15 para a função de Rastringin até N=100 para a função *Sphere*. Como pode ser observado na Figura 5.27, o número máximo de avaliações de F(**X**), utilizado como critério de parada, varia de $3 \cdot 10^6$ para a função *Sphere* até $3 \cdot 10^7$ para as funções Rastringin e Griewangk.

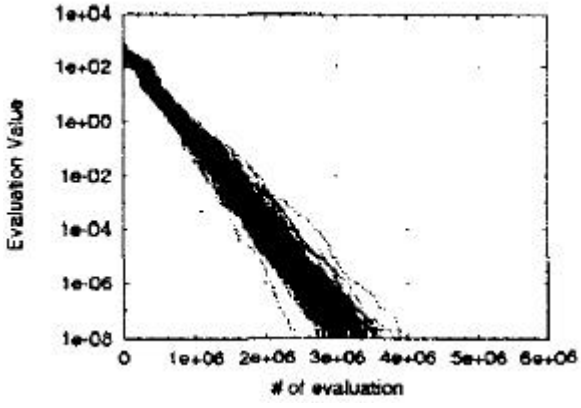
Tabela 5.4 – Funções teste utilizadas em Someya e Yamamura (2001)

Função	Min. F(**X**)	N[1]	Restrições	**X***[2] F(**X***)
Sphere	$F(\mathbf{X}) = \sum_{i=1}^{N} (X_i \cdot X_i)$	100	$-5{,}12 \le X_i \le 5{,}12$	$X_i^* = 0$ $F(\mathbf{X}^*) = 0$
Rosenbrock	$F(\mathbf{X}) = \sum_{i=1}^{N-1} [100 \cdot (X_i^2 - X_{i+1})^2 + (1 - X_i)^2]$	20	$-2{,}048 \le X_i \le 2{,}048$	$X_i^* = 1$ $F(\mathbf{X}^*) = 0$
Step	$F(\mathbf{X}) = \sum_{i=1}^{N} (\mathrm{INT}(X_i + 0{,}5)^2)$	70	$-5{,}12 \le X_i \le 5{,}12$	$\lvert X_i^* \rvert < 0{,}5$ $F(\mathbf{X}^*) = 0$
Schwefel [3]	$F(\mathbf{X}) = 418{,}9829N - \sum_{i=1}^{N} X_i \sin(\sqrt{\lvert X_i \rvert})$	60	$-512 \le X_i \le 512$	$X_i^* = 420{,}9687$ $F(\mathbf{X}^*) = 0$ [3]
Rastringin	$F(\mathbf{X}) = 3N + \sum_{i=1}^{N} (X_i^2 - 3\cos(2\pi X_i))$	15	$-5{,}12 \le X_i \le 5{,}12$	$X_i^* = 0$ $F(\mathbf{X}^*) = 0$
Griewangk	$F(\mathbf{X}) = 1 + \sum_{i=1}^{N} \frac{X_i^2}{4000} - \prod_{i=1}^{N} \cos\left(\frac{X_i}{\sqrt{i}}\right)$	30	$-512 \le X_i \le 512$	$X_i^* = 0$ $F(\mathbf{X}^*) = 0$

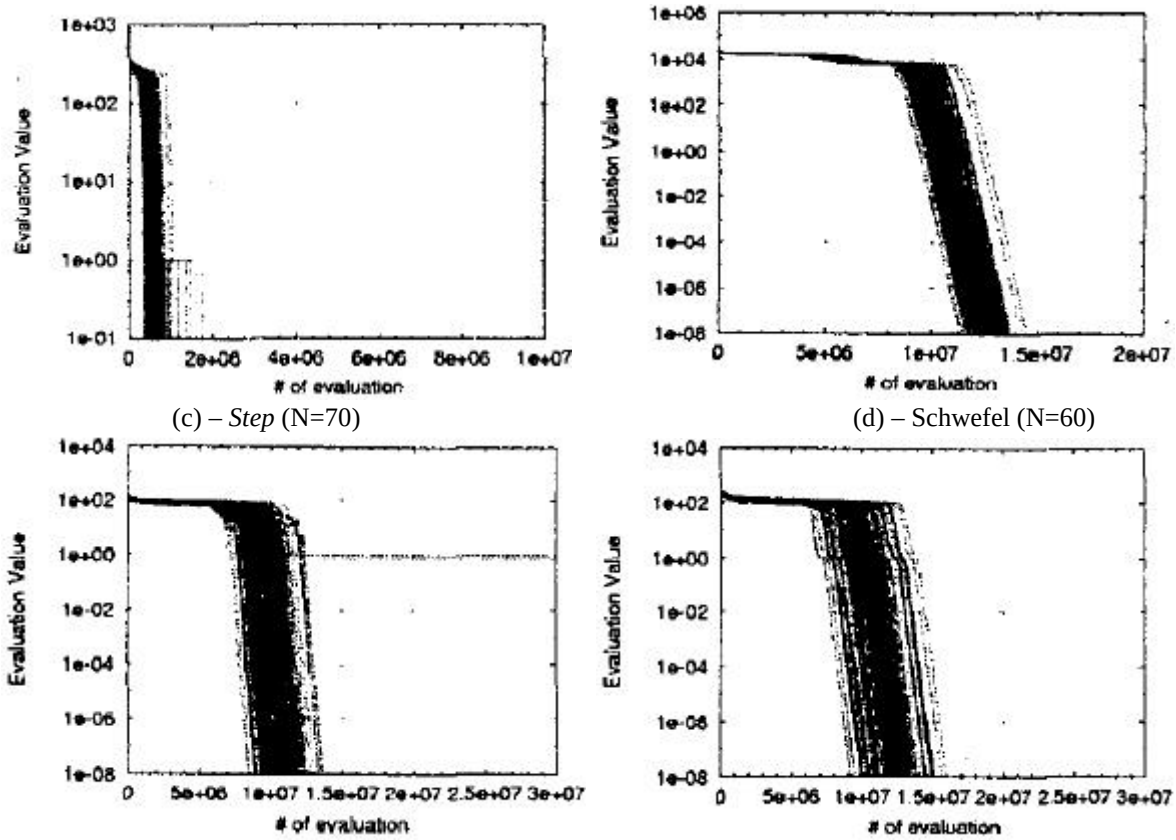
[1] Número de dimensões (variáveis); [2] **X***=Ponto de mínimo global; [3] Verificou-se que a solução $X_i^* = 420{,}9687$ leva a F(**X**)= $1{,}6229023458436 13 \cdot 10^{-8}$ quando calculada com dupla precisão e não F(**X***)=0. Valores de F(**X**) da ordem de 10^{-10} forma encontrados para X_i=420,96875.



(a) – *Sphere* (N=100)



(b) – Rosenbrock (N=20)



(c) – *Step* (N=70) (d) – Schwefel (N=60)

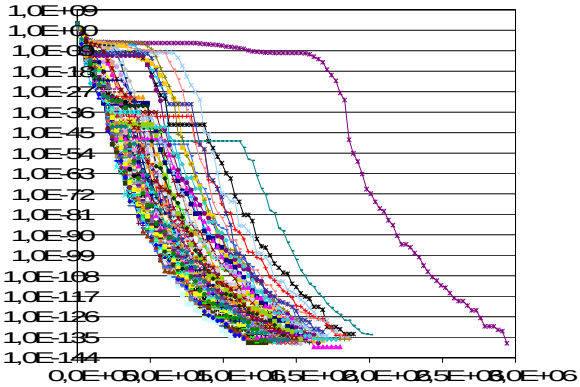
(e) – Rastringin (N=15) (f) – Griewangk (N=30)

Fig. 5.27 - Evolução de $F(\mathbf{X}_{melhor})$ obtido pelo GSA (100 execuções) para as 6 funções teste. (Fonte: Someya e Yamamura (2001))

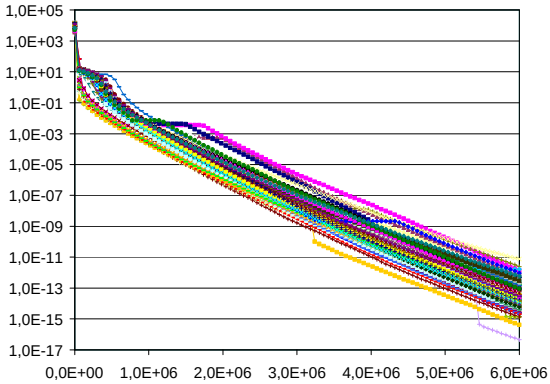
Os resultados obtidos pela aplicação do híbrido de GEO_{var} SA4 + EE às mesmas funções estão na Figura 5.28, e foram obtidos com o melhor ajuste de parâmetros encontrado para cada função, incluindo l_j. Um mesmo número de mutações $l_j=l$ foi utilizado para todas as variáveis de projeto. A Tabela 5.5 lista os valores dos parâmetros principais μ e a para os quais os melhores resultados foram encontrados e também *l*. Os valores dos limites b_{MIN} e b_{MAX} foram fixados em 1,05 e 120,0, respectivamente para todas as 6 funções teste. Todas as demais condições de teste foram as mesmas de Someya e Yamamura (2001), como o critério de parada, limites laterais das variáveis de projeto, etc.

Tabela 5.5 – Valores dos parâmetros μ, a e *l*.

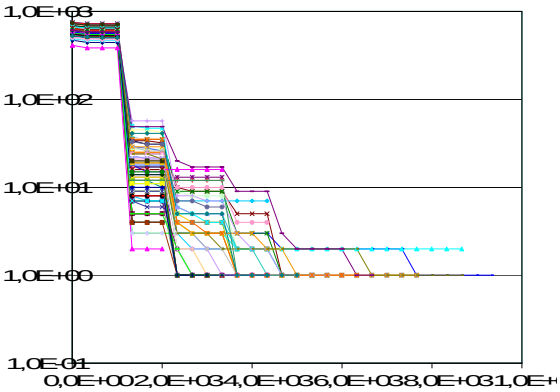
Function	GEO_{var} híbrido		
	μ	a	*l*
Sphere	0,01	0,1	32
Rosenbrock	0,0001	0,01	16
Step	0,3	0,2	16
Schwefel	0,005	0,05	16
Rastringin	0,01	0,05	16
Griewangk	0,075	0,075	16



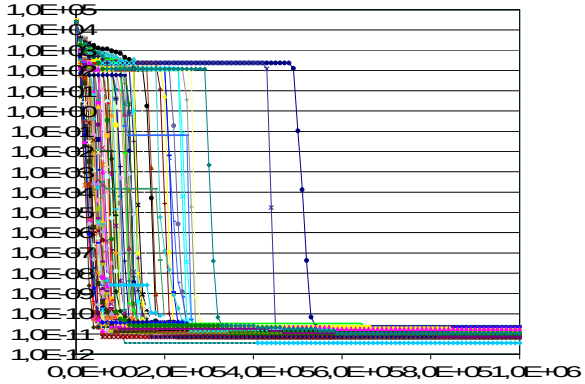
(a) – *Sphere* (N=100)



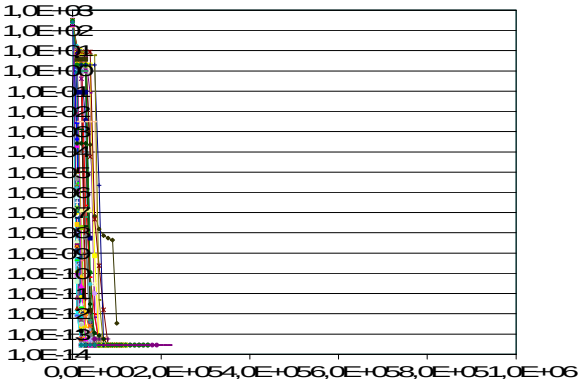
(b) – Rosenbrock (N=20)



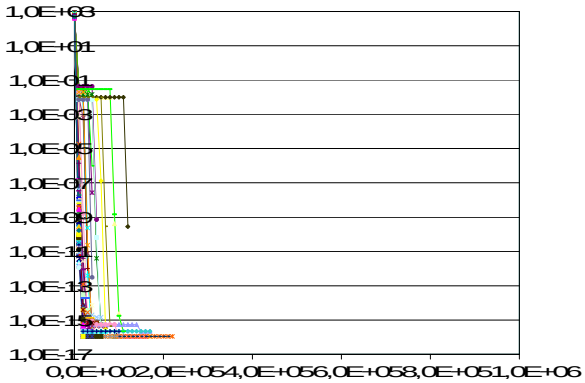
(c) – *Step* (N=70)



(d) – Schwefel (N=60)



(e) – Rastringin (N=15)



(f) – Griewangk (N=30)

Fig. 5.28 - Evolução de F($\mathbf{X}_{\mathbf{melhor}}$) obtido pelo híbrido de GEO_{var} SA4 + EE (100 execuções)
para as 6 funções teste.

Nas Figuras 5.28a, 5.28c, 5.28e, e 5.28f, as curvas todas terminando antes de tocar o eixo horizontal (borda do gráfico) indicam que o híbrido de GEO_{var} SA4 com o EE conseguiu fazer os valores de F($\mathbf{X}_{\mathbf{melhor}}$) atingirem o valor do ótimo global, igual a zero (F($\mathbf{X}^*$)=0)), valor que não pode ser representado em um gráfico com escala logarítmica. Comparando-se os resultados do híbrido de GEO_{var} com os do GSA, o placar é 5 a 1 em favor do híbrido, na convergência de médio e longo prazo do valores de F($\mathbf{X}_{\mathbf{melhor}}$). O

algoritmo GSA parece superior para a função de Rosenbrock (Figuras 5.27b e 5.28b). Por outro lado, o híbrido GEO_{var} SA4 + EE é indubitavelmente superior para todas as outras funções. Para a função *sphere*, a Figura 5.27a mostra que, com 10^6 avaliações da função, as curvas de GSA atingem $F(\mathbf{X_{melhor}})<10^{-8}$ enquanto as curvas de GEO_{var} SA4 + EE (Figura 5.28a) atingem $F(\mathbf{X_{melhor}})<10^{-100}$. Para a função *step*, GEO_{var} SA4 + EE parece ser uma espécie de "solucionador natural", uma vez que ele foi capaz de resolvê-la em todas as 100 execuções em menos de dez mil avaliações da função, o que significa menos do que 10 iterações do algoritmo. Este resultado impressiona ainda mais quando comparado aos resultados do GSA (Figura 5.27c), onde, nas piores execuções, mais de um milhão de avaliações da função foram necessárias. Para a função de Schwefel, a Figura 5.28d mostra que todas as curvas de GEO_{var} SA4 + EE atingem $F(\mathbf{X_{melhor}})<10^{-11}$ em até $6\cdot10^5$ avaliações da função, enquanto a Figura 5.27d mostra que, para GSA, nenhuma curva atinge $F(\mathbf{X_{best}})<10^{-8}$ antes de 10^7 (dez milhões) de avaliações da função. Para a função de Rastringin, a Figura 5.28e mostra que todas as curvas de GEO_{var} SA4 + EE encontram a solução ótima $F(\mathbf{X_{melhor}})= F(\mathbf{X^*})=0$ com menos de $3\cdot10^5$ avaliações da função, enquanto a Figura 5.27e mostra que as curvas de GSA atingem $F(\mathbf{X_{melhor}})<10^{-8}$ com cerca de 10^7 (dez milhões) de avaliações da função. O caso da função de Griewangk é muito parecido ao da função de Rastringin. Agora, a Figura 5.28f mostra que, novamente, todas as curvas de GEO_{var} SA4 + EE encontram a solução ótima global $F(\mathbf{X_{melhor}})=F(\mathbf{X^*})=0$ em menos de $3\cdot10^5$ avaliações da função, enquanto a Figura 5.27f mostra que somente algumas curvas de GSA atingem $F(\mathbf{X_{melhor}})<10^{-8}$ com cerca de 10^7 (dez milhões) de avaliações da função.

Com relação aos valores dos parâmetros μ, a, e *l*, apresentados na Tabela 5.5, procedeu-se o ajuste do seguinte modo. Primeiro, o valor de *l* foi fixado em l=16 e os valores de μ e a foram ajustados. Depois, os melhores valores obtidos para μ e a foram fixados e alguns valores de $l\neq16$ foram testados. Os valores tabelados indicam que a função *Step* é a que requer ou permite a maior tendenciosidade do aprendizado, com μ=0,3 e também a maior taxa de aprendizado, com a=0,2. Por outro lado, a função de Rosenbrock é a que requer ou permite a menor tendenciosidade do aprendizado, com μ=0,0001 e também a menor taxa de aprendizado, com a=0,01. Com relação à *l*, a única novidade fica por conta da função *Sphere*, que obteve os melhores resultados para l=32 ao invés de l=16. Para esta função, a Figura 5.28a mostra que, antes de atingir $F(\mathbf{X_{melhor}})=F(\mathbf{X^*})=0$, valores tão

pequenos quanto 10^{-138} foram obtidos. Para b_{MAX}=120 e l_j=l=16, a equação 5.17 permite calcular um $r_{j_{MAX}}$=5,41·10^{-34}. Lembrando que $r_{j_{MAX}}$ é dada em frações do espaço de projeto e sabendo que X_{MAX_j}-X_{MIN_j}=5,12-(-5,12)=10,24 para a função *Sphere*, então, a resolução em unidades de X_j vale 10,24·5,41·10^{-34}=5,54·10^{-33}. A consulta aos arquivos gerados quando da execução dos testes para a função *Sphere*, revela que valores de F(**X**) da ordem de 10^{-138} foram obtidos quando apenas o último X_j (j=100) era não nulo e da ordem de 10^{-69}. Nesse caso, F(**X**) reduz-se ao quadrado deste último X_j, já que os demais são nulos, resultando em 10^{-138}. A diferença entre resoluções da ordem de 10^{-33} e resoluções da ordem 10^{-69} é muito grande, praticamente impossibilitando que seja superada por mutações aleatórias ou por variações fracionárias da base ao longo de várias iterações. Portanto, com l=16 é impossível atingir F(**X**) da ordem de 10^{-138}. Agora, com b_{MAX}=120 e l_j=l=32, usando-se a equação 5.17 novamente encontra-se $r_{j_{MAX}}$=2,93·10^{-67}, o que leva a uma resolução de 2,99·10^{-66} em unidades de X_j. Esta resolução dista apenas três ordens de magnitude da necessária para atingir diretamente (em uma única iteração) F(**X**) da ordem de 10^{-138}. A resolução de X_j da ordem de 10^{-69} é atingida, portanto, ao longo de várias iterações do algoritmo.[1]

Terceira Comparação

Em Ballester e Carter (2004), os autores reportaram os melhores resultados encontrados até então para um conjunto de funções teste e os compararam com aqueles obtidos pelo algoritmo desenvolvido por eles próprios. Estes resultados são comparados aqui com os obtidos pelo híbrido de GEO_{var} SA4 com o EE.

As seis funções teste: *Ellipsoidal*, Schwefel, Rosenbrock, Ackley, Rastringin, e Rastringin Rotacionada usam N=20 variáveis de projeto e estão definidas na Tabela 5.6. Nesta terceira comparação, a função de Schwefel não é a mesma utilizada na primeira e na segunda comparação, apesar do nome ser o mesmo.

[1] Os valores convergidos de F(**X**)<10^{-100} para a função *Sphere* geraram ainda uma descoberta inesperada. O compilador Fortran utilizado (*Digital Visual Fortran* 6.0 para *Windows*) apresentou um *bug* na formatação numérica dos arquivos de saída gerados durante os testes. Para números abaixo de 10^{-100}, ele deixou de escrever a letra "E" que separa a mantissa do expoente. Assim, um número que deveria ser escrito "1,5087651E-132" era escrito "1,5087651-132". Ao tentar gerar gráficos com estes arquivos tais números eram interpretados como texto e houve necessidade de incluir o "E" que faltava "manualmente".

Tabela 5.6 – Funções teste utilizadas em Ballester e Carter (2004)

Função	Min. $F(\mathbf{X})$	N[1]	Restrições[2]	$\mathbf{X}^*$[3] $F(\mathbf{X}^*)$
Ellipsoidal	$F(X) = \sum_{i=1}^{D} i \cdot X_i^2$	20	$-10 \le X_i \le 10$	$X_i^* = 0$ $F(\mathbf{X}^*) = 0$
Schwefel	$F(X) = \sum_{i=1}^{N} \left(\sum_{j=1}^{i} X_j \right)^2$	20	$-10 \le X_i \le 10$	$X_i^* = 0$ $F(\mathbf{X}^*) = 0$
Rosenbrock	$F(\mathbf{X}) = \sum_{i=1}^{N-1} \left[100 \cdot (X_i^2 - X_{i+1})^2 + (1 - X_i)^2 \right]$	20	$-2{,}048 \le X_i \le 2{,}048$	$X_i^* = 1$ $F(\mathbf{X}^*) = 0$
Ackley	$F(\mathbf{X}) = 20 + e - 20\, e^{\left(-0{,}2 \sqrt{\frac{1}{N} \sum_{i=1}^{N} X_i^2} \right)} - e^{\left(\frac{1}{N} \sum_{i=1}^{N} \cos(2pX_i) \right)}$	20	$-30 \le X_i \le 30$	$X_i^* = 0$ $F(\mathbf{X}^*) = 0$
Rastringin	$F(\mathbf{X}) = 10\, N + \sum_{i=1}^{N} \left(X_i^2 - 10 \cos(2pX_i) \right)$	20	$-5{,}12 \le X_i \le 5{,}12$	$X_i^* = 0$ $F(\mathbf{X}^*) = 0$
Rastringin Rotacionada	$F(\mathbf{Y}) = 10 \cdot N + \sum_{i=1}^{N} \left(Y_i^2 - 10 \cdot \cos(2pY_i) \right)$ $\mathbf{Y} = \mathbf{A} \cdot \mathbf{X}$, com $A_{i,i} = 4/5$, $A_{i,i+1} = 3/5$ (i impar), $A_{i,i-1} = -3/5$ (i par), $A_{i,j} = 0$ (os demais)	20	$-5{,}12 \le X_i \le 5{,}12$	$X_i^* = 0$ $F(\mathbf{X}^*) = 0$

[1] Número de dimensões (variáveis); [2] Ballester e Carter (2004) utilizaram inicialização não simétrica a partir do intervalo [-10,-5] para todas as variáveis, a fim de evitar que uma inicialização simétrica facilitasse o trabalho do AG desenvolvido e testado por eles. Diferente do AG, o GEO e suas versões híbridas não se beneficiam da simetria do intervalo. Além disso, o GEO e suas versões híbridas só trabalham dentro do intervalo definido para as variáveis e, por isso, o intervalo deve conter a solução ótima. Tendo em vista estes aspectos, utilizou-se os limites que tradicionalmente são usados para estas funções e, em geral, são maiores do que os usados por Ballester e Carter (2004); [3] $\mathbf{X}^*$=Ponto de mínimo global

Adicionalmente, uma versão modificada do híbrido de GEO_{var} SA4 + EE foi desenvolvida e testada. Como se sabe, o GEO_{var} SA4 + EE efetua as ordenações das mutações separadamente para cada variável, escolhe elitisticamente a melhor delas e, ao final da iteração, promove a alteração simultânea de todas. Na versão modificada, a mutação da primeira variável é escolhida e efetuada, ou seja, o vetor **X** é alterado. Em seguida, é feito o mesmo para a segunda variável e assim sucessivamente. Portanto, nesta

nova versão de GEO_{var} SA4 + EE, o valor de **X** que serve de referência para as mutações é diferente para cada variável de projeto.

Os resultados a serem apresentados foram obtidos com o melhor ajuste de parâmetros encontrado para cada função teste. Um mesmo número de mutações $l_j=l$ foi utilizado para todas as variáveis de projeto. A Tabela 5.7 lista os valores dos parâmetros principais μ e a para os quais os melhores resultados foram encontrados e também *l*. Os valores dos limites b_{MIN} e b_{MAX} foram fixados em 1,05 e 120, respectivamente para todas as 6 funções teste. Com relação à *l*, o ajuste foi não intensivo, começando com *l*=16 e somente após ajustar e fixar os outros parâmetros é que algumas tentativas foram feitas com outros valores de *l*. Todas as outras condições foram mantidas tão próximas quanto possível das usadas em Ballester e Carter (2004), a fim de permitir uma comparação apropriada.

Tabela 5.7 – Valores dos parâmetros μ, a e *l*.

Função	GEO_{var} híbrido			GEO_{var} híbrido modificado		
	μ	a	*l*	μ	a	*l*
Ellipsoidal	0,01	0,05	32	0,002	0,2	32
Schwefel	0,0	0,15	3	0,002	0,04	16
Rosenbrock	0,0	0,005	16	0,0	0,005	16
Ackley	0,01	0,1	16	0,01	0,1	16
Rastringin	0,02	0,05	16	0,02	0,05	16
Rastringin Rotacionada	0,0	0,007	3	0,0	0,007	3

A Tabela 5.8 apresenta os resultados obtidos pelo híbrido de GEO_{var} SA4 + EE e sua versão modificada (em negrito), juntamente com uma seleção dos resultados apresentados em Ballester e Carter (2004). O AG denominado G3-PCX (Deb et al., 2002) é uma versão de AG com codificação real que foi usado em Ballester e Carter (2004) e também é usado aqui como uma espécie de algoritmo de referência para comparação, por conta de seus excelentes resultados (de acordo com Ballester e Carter (2004), G3-PCX é o Algoritmo Evolucionário com os melhores resultados reportados para as funções *Ellipsoidal*, Schwefel, e Rosenbrock). Adicionalmente, dos dois AGs com codificação real desenvolvidos em Ballester e Carter (2004), chamados SPC-vSBX e SPC-PNX, aquele com o melhor desempenho em cada função teste também é apresentado na Tabela 5.8. No

caso do híbrido de GEOvar SA4 + EE e de sua versão modificada, 50 execuções independentes foram utilizadas para a geração dos resultados.

Assim como em Ballester e Carter (2004), o melhor, o mediano, e o pior número de avaliações da função necessários para cada algoritmo atingir uma solução com um valor alvo de $F(\mathbf{X}_{\mathbf{melhor}})=10^{-20}$ é mostrado na Tabela 5.8. Se o alvo não é alcançado então o melhor valor obtido para $F(\mathbf{X}_{\mathbf{melhor}})$ com 10^6 avaliações é informado. A coluna Sucesso se refere a quantas execuções independentes atingiram o valor alvo (para as funções unimodais: *Ellipsoidal* e Schwefel), ou terminaram dentro da base de atração do mínimo global (para as quatro funções multimodais restantes).

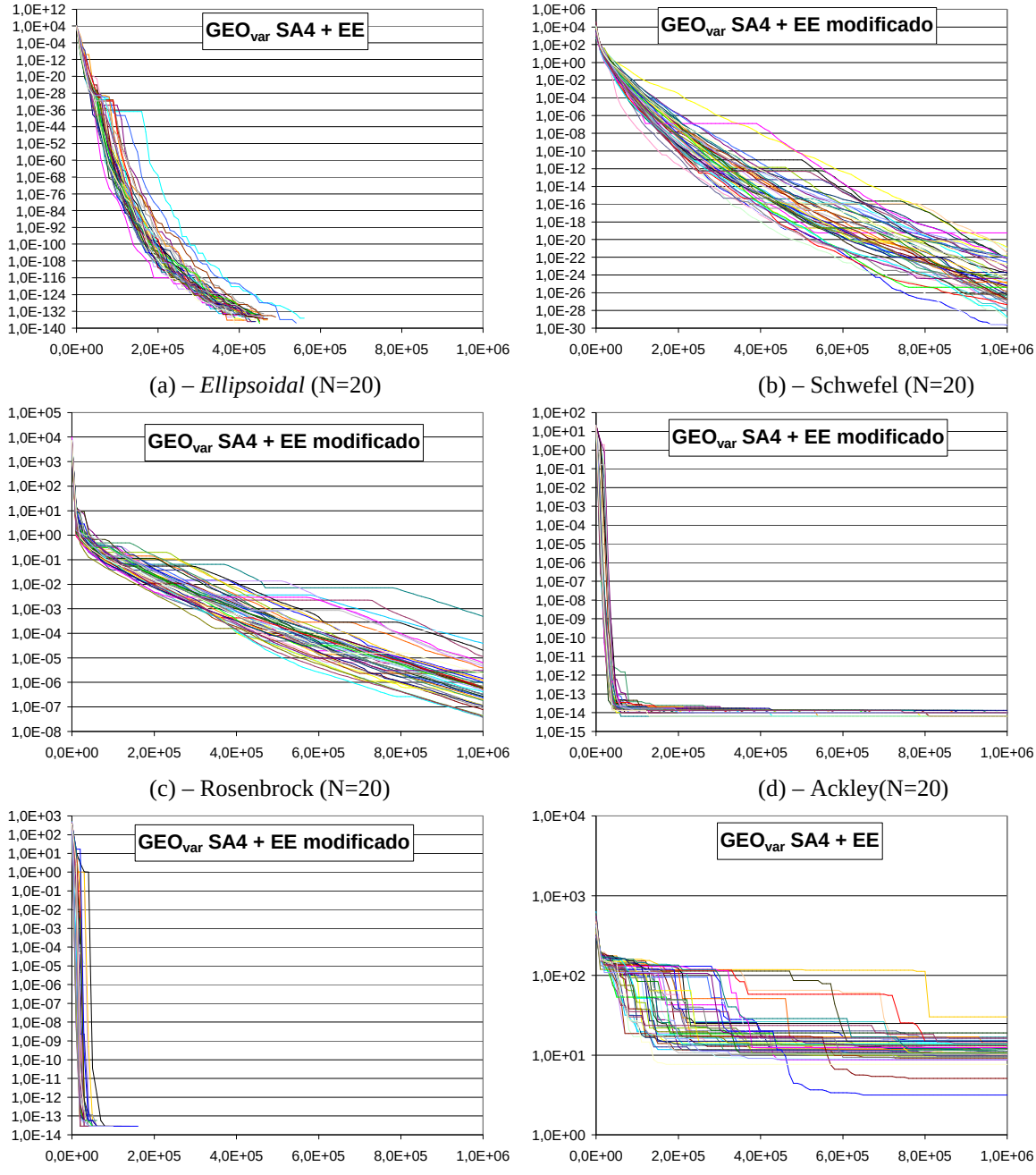
Comparando-se os resultados de todos os algoritmos e usando a mediana ou, se necessário, o melhor $F(\mathbf{X}_{\mathbf{melhor}})$ após 10^6 avaliações com critério, o escore é favorável ao G3-PCX com 3 vitórias, seguido pela versão modificada do híbrido GEO_{var} SA4 + EE com 2 vitórias e pelo híbrido GEO_{var} SA4 + EE com uma vitória. Analisando os resultados de forma um pouco mais aprofundada, G3-PCX obteve o melhor desempenho para as funções *Ellipsoidal*, Schwefel, e Rosenbrock, enquanto híbrido modificado de GEO_{var} SA4 + EE obteve o melhor desempenho para as funções de Ackley e Rastringin e, GEO_{var} SA4 + EE obteve o melhor desempenho para a função de Rastringin Rotacionada. É importante notar que G3-PCX foi claramente superior para as funções unimodais ou quase unimodais do conjunto (*Ellipsoidal*, Schwefel, e Rosenbrock), mas foi claramente inferior para as altamente multimodais (Ackley, Rastringin, e Rastringin Rotacionada). Uma vez que atualmente algoritmos de otimização de funções multimodais são necessários para resolver problemas complexos da ciência e da engenharia, é importante analisar os resultados sob este prisma. Efetuando uma comparação somente com os algoritmos eficientes com funções multimodais, isto é, todos menos G3-PCX, o escore geral é 2 vitórias para SPC-PNX (Schwefel e Rosenbrock), 2 vitórias para GEO_{var} SA4 + EE (*Ellipsoidal* e Rastringin Rotacionada), e também 2 vitórias para GEO_{var} SA4 + EE modificado (Ackley e Rastringin). Entre GEO_{var} SA4 + EE e sua versão modificada, a última foi superior em 4 dos 6 testes. Para as funções *Ellipsoidal* e Rastringin Rotacionada GEO_{var} SA4 + EE foi superior enquanto GEO_{var} SA4 + EE modificado foi superior para as demais (Schwefel, Rosenbrock, Ackley, e Rastringin).

Tabela 5.8 – Desempenho comparativo entre GEO_{var} SA4 + EE, GEO_{var} SA4 + EE modificado, G3-PCX, SPC-PNX, e SPC-vSBX.

Função	Algoritmo	Número de avaliações			$F(\mathbf{X}_{melhor})$	Sucesso
		Melhor	Mediano	Pior		
Ellipsoidal	G3-PCX	5.826	6.800	7.728	10^{-20}	10/10
	SPC-PNX	36.360	39.360	40.905	10^{-20}	10/10
	GEO_{var} + EE	**16.005**	**24.359**	**37.640**	**10^{-20}**	**50/50**
	GEO_{var} + EE modificado	**16.808**	**33.315**	**197.309**	**10^{-20}**	**50/50**
Schwefel	G3-PCX	13.988	15.602	17.188	10^{-20}	10/10
	SPC-PNX	236.342	283.321	299.301	10^{-20}	10/10
	GEO_{var} + EE	**10^6**	-	-	**$4 \cdot 10^{-5}$**	**0/50**
	GEO_{var} + EE modificado	**477.887**	**685.512**	**957.552**	**10^{-20}**	**49/50**
Rosenbrock	G3-PCX	16.508	21.452	25.520	10^{-20}	36/50
	SPC-PNX	10^6	-	-	10^{-10}	38/50
	GEO_{var} + EE	**10^6**	-	-	**$3 \cdot 10^{-4}$**	**50/50**
	GEO_{var} + EE modificado	**10^6**	-	-	**$4 \cdot 10^{-8}$**	**50/50**
Ackley	G3-PCX	10^6	-	-	3,96	0/10
	SPC-PNX	45.736	48.095	49.392	10^{-10}	10/10
	GEO_{var} + EE	**17.616**	**24.681**	**50.936**	**10^{-10}**	**50/50**
	GEO_{var} + EE modificado	**17.914**	**23.616**	**39.308**	**10^{-10}**	**50/50**
Rastringin	G3-PCX	10^6	-	-	15,9	0/10
	SPC-vSBX	260.658	306.819	418.482	10^{-20}	6/10
	GEO_{var} + EE	**18.346**	**60.457**	**342.803**	**10^{-20}**	**50/50**
	GEO_{var} + EE modificado	**18.584**	**46.655**	**166.877**	**10^{-20}**	**50/50**
Rastringin Rotacionada	G3-PCX	10^6	-	-	309,0	0/10
	SPC-PNX	10^6	-	-	3,98	0/10
	GEO_{var} + EE	**10^6**	-	-	**3,15**	**0/50**
	GEO_{var} + EE modificado	**10^6**	-	-	**4,62**	**0/50**

Durante as 50 execuções independentes cujos resultados estão na Tabela 5.8, o número de avaliações necessário para atingir o valor alvo de $F(\mathbf{X}_{melhor})$ foi salvo em arquivo, mas permitiu-se que todas as execuções continuassem até 10^6 avaliações da respectiva função. Isto foi feito de forma a permitir o acesso às propriedades de convergência do híbrido de GEO_{var} SA4 + EE e de

sua versão modificada. A Figura 5.29 mostra curvas com a evolução de $F(\mathbf{X}_{melhor})$ para todas as 50 execuções, obtidas por GEO_{var} SA4 + EE ou por sua versão modificada, para as 6 funções teste. Entre GEO_{var} SA4 + EE e sua versão modificada, aquela com a melhor mediana (Tabela 5.8) foi selecionada para ter suas curvas apresentadas.



(a) – *Ellipsoidal* (N=20)

(b) – Schwefel (N=20)

(c) – Rosenbrock (N=20)

(d) – Ackley(N=20)

(e) – Rastringin (N=20)

(f) – Rastringin Rotacionada (N=20)

Fig. 5.29 - Evolução de $F(\mathbf{X}_{melhor})$ obtido pelas versões original ou modificada do GEO_{var} SA4 + EE (50 execuções) para as 6 funções teste utilizadas em Ballester e Carter (2004)

Alguns comentários sobre os resultados gráficos são úteis. Na Figura 5.29a (função *Ellipsoidal*), todas as linhas de $F(\mathbf{X}_{\mathbf{melhor}})$ terminando antes de encostar na borda inferior indicam que os respectivos valores de $F(\mathbf{X}_{\mathbf{melhor}})$ foram para 0 (zero), um valor que não pode ser representado em escala logarítmica. Uma vez que o ótimo global é $F(\mathbf{X}^*)=0$, então GEO_{var} SA4 + EE conseguiu encontrar a solução ótima em todas as 50 execuções independentes. Antes de atingir a solução ótima, GEO_{var} SA4 + EE encontrou valores da ordem de 10^{-137}, um valor muito pequeno. De fato, o GEO_{var} SA4 + EE modificado também obteve algo similar e, embora não tenha sido capaz de encontrar $F(\mathbf{X}_{\mathbf{melhor}})=0$, conseguiu terminar todas as 50 execuções com valores de $F(\mathbf{X}_{\mathbf{melhor}})$ entre 10^{-176} e 10^{-185}. Na Figura 5.29e (Rastringin), novamente todos os valores de $F(\mathbf{X}_{\mathbf{melhor}})$ convergiram para o valor ótimo global de 0 (zero). Para esta função (Rastringin), ambos GEO_{var} SA4 + EE e sua versão modificada foram capazes de encontrar a solução ótima. Como mostra a Figura 5.29e para o GEO_{var} SA4 + EE modificado, isto foi feito com menos de $2{\cdot}10^5$ avaliações da função. Em todos os gráficos da Figura 5.29, observa-se que GEO_{var} SA4 + EE e sua versão modificada demonstraram boas propriedades de convergência, uma vez que todas as execuções de cada função teste terminaram com valores da função relativamente próximos. Se a diferença em magnitude entre o maior e o menor valor final de $F(\mathbf{X}_{\mathbf{melhor}})$ encontrado nas 50 execuções é tomado como uma medida de dispersão, então a maior dispersão, com cerca de 11 ordens de magnitude, ocorreu para a função de Schwefel (Figura 5.29b).

5.5 – Conclusões

Neste Capítulo, duas das versões aprimoradas do algoritmo GEO desenvolvidas no Capítulo 3 foram hibridizadas, uma com o método Recozimento Simulado - RS, e a outra com o método das Estratégias Evolutivas - EE. Para a hibridização com o RS foi a escolhida a versão SA1, que introduziu no GEO a capacidade de manter a melhor solução encontrada ao longo da busca. Para a hibridização com o EE foi escolhida a versão SA4, que introduziu no GEO a representação real das variáveis de projeto e um esquema alternativo para as mutações. Os testes de desempenho efetuados mostraram que, na grande maioria dos casos, os híbridos GEO + RS e GEO + EE ocasionaram melhora no desempenho, relativamente à versão não hibridizada. Os ganhos obtidos com GEO + EE foram substancialmente maiores do que os obtidos com GEO + RS. Este fato motivou a

realização de mais testes de desempenho envolvendo o GEO + EE, que foi então comparado a duas novas versões de AGs com codificação real. Nas duas comparações, os bons resultados obtidos pelo GEO + EE colocaram-no em vantagem frente aos demais algoritmos utilizados na comparação, principalmente no caso das funções altamente multimodais. Este fato sugere que o híbrido GEO + EE é especialmente eficiente quando aplicado às funções altamente multimodais. Esta conclusão é importante, visto que funções altamente multimodais são o tipo de função que se espera encontrar em problemas práticos complexos da ciência e da engenharia e estes, por sua vez, foram a razão principal, a motivação maior presente no desenvolvimento das versões híbridas apresentadas.

CAPÍTULO 6

OTIMIZAÇÃO MULTIOBJETIVO

6.1 - Introdução

Este Capítulo descreve uma versão multiobjetivo derivada do algoritmo GEO e os resultados de sua aplicação a um conjunto de funções teste escolhidas dentre funções teste utilizadas em publicações anteriores (Srinivas e Deb, 1994; Mason et al., 1998; Messac et al., 2000). A versão multiobjetivo apresentada aqui foi desenvolvida para gerar a fronteira de Pareto mantendo a característica essencial do GEO: universalidade de aplicação e apenas um parâmetro para ajuste do algoritmo.

Em problemas de otimização multiobjetivo (Goicoechea et al., 1982; Van Veldhuizen e Lamont, 2000; Coello, 1999; Miettinen, 2001), deseja-se otimizar (minimizar or maximizar) simultaneamente um conjunto de objetivos. Normalmente, para estes problemas, não existe uma solução ótima, mas sim uma família de soluções de compromisso, que não necessitam ser a solução ótima global de nenhum dos objetivos. Seguindo a noção de otimização de Pareto, esta família é chamada de conjunto de Pareto ou conjunto ótimo de Pareto e os pontos correspondentes no espaço dos objetivos são chamados de frente ou fronteira de Pareto (Veldhuizen e Lamont, 2000). Um ponto viável no espaço objetivo pertence à fronteira de Pareto se nenhum dos objetivos pode ser melhorado ainda mais sem piorar qualquer dos outros objetivos. Qualquer solução do conjunto de Pareto pode ser usada e é tarefa do projetista escolher qual será implementada, baseado na própria fronteira de Pareto ou outros fatores.

Existem, na literatura, muitas técnicas numéricas desenvolvidas para lidar com problemas de otimização multiobjetivo (Shapour, 1996; Miettinen, 2001; Coello, 2005). Estas técnicas podem ser agrupadas basicamente em três abordagens (Fonseca e Fleming, 1995; Coello, 1999).

A primeira abordagem utilizada para obtenção da fronteira de Pareto,, chamada de métodos de agregação, foi transformar em monoobjetivo o problema multiobjetivo, pela combinação de todos os objetivos em uma única função objetivo,

também denominada função de compromisso. O exemplo mais simples de tais métodos é a abordagem da soma ponderada, onde a função compromisso é uma combinação linear dos objetivos. Se o projetista tem a informação *a priori* sobre quais pesos devem ser usados na soma ponderada, o problema é, de fato, monoobjetivo e não há necessidade de encontrar a fronteira de Pareto. Se não for este o caso, a fronteira de Pareto é mapeada resolvendo-se diversas vezes o problema monoobjetivo, cada vez com um conjunto diferente de pesos (coeficientes). O projetista é livre para escolher o método de otimização global de sua preferência para resolver os problemas monoobjetivos resultantes. Já é sabido que está abordagem não tem a habilidade de capturar a totalidade da fronteira de Pareto para problemas em que a superfície de Pareto é côncava (Messac et al., 2000). Deve ser notado, entretanto, que a montagem da função monoobjetivo pode ser feita de diversas maneiras, incluindo as não-lineares, onde os coeficientes introduzidos alteram a curvatura da função monoobjetivo, permitindo então a captura dos pontos da fronteira de Pareto que não podiam ser capturados pela abordagem da soma ponderada. Baseadas neste fato, diversas fórmulas não-lineares tem sido desenvolvidas. Messac et al. (2000) lista diversas fórmulas de agregação de funções que tinham sido usadas até então e comenta brevemente seus pontos fortes e fracos, enquanto Coello (1999) o faz de uma forma mais extensiva, mas no contexto dos Algoritmos Evolutivos Multiobjetivo.

Uma segunda abordagem, aqui denominada abordagem Pareto direta, utiliza diretamente a não-dominância como um critério para guiar a busca. Na literatura, essa abordagem é muitas vezes dita abordagem baseada em Pareto ou abordagem de Pareto. Infelizmente, esta designação leva a uma ambíguidade, visto que todos os métodos multiobjetivo usam, necessariamente, em algum momento, o conceito de otimalidade de Pareto. Por esta razão, esta designação não é usada nesta Tese. Não-dominância é um conceito de otimalidade de Pareto. Um ponto viável no espaço de busca domina outros se ele melhora pelo menos um dos objetivos sem piorar qualquer dos demais. Este ponto é dito ser um ponto não-dominado. Considerando a totalidade do espaço objetivo viável, o conjunto de todos os pontos não-dominados forma a própria fronteira de Pareto. Mantendo-se um conjunto de pontos não-

dominados ao longo da busca, consegue-se uma aproximação da fronteira de Pareto. Esta abordagem, manter um conjunto de pontos não-dominados ao longo da busca, nasceu dentro do ramo de métodos dos Algoritmos Evolutivos Multiobjetivo, onde o critério de não-dominância foi usado para definir a adaptação dos indivíduos (soluções) da população dos Algoritmos Genéticos (Goldberg, 1989). Como consequência, não apenas o processo de seleção é influenciado diretamente pelo conceito de otimalidade de Pareto (a não-dominância), como também, os indivíduos da população tornam-se uma aproximação direta das soluções de Pareto. Exemplos de algoritmos baseados nesta abordagem são o *Multiple Objective Genetic Algorithm* (*MOGA*), o *Non-Dominated Sorting Genetic Algorithm* (*NSGA*), e o *Niched Pareto Genetic Algorithm* (*NPGA*). Coello (1999) e Van Veldhuizen e Lamont (2000) analisam com mais profundidade técnicas baseadas nesta abordagem.

E a terceira abordagem, aqui denominada como abordagem Pareto indireta, preenche o conjunto complementar, relativamente às duas primeiras abordagens. Ela agrupa todos os métodos que não utilizam nem funções de agregação nem a abordagem de Pareto direta. É importante dizer que, a rigor, a abordagem dos métodos de agregação também pertence à abordagem Pareto indireta. Entretanto, por razões históricas, cronológicas, ou mesmo de costume, os métodos de agregação são mantidos como uma abordagem à parte. Na abordagem Pareto indireta, o conceito de otimalidade de Pareto é usado de maneira indireta, ou seja, ele não é usado para influenciar diretamente o processo de busca. Por processo de busca entendam-se os pontos do espaço de busca visitados pelo algoritmo durante a busca. No âmbito dos Algoritmos Evolutivos Multiobjetivos, isto quer dizer que o conceito de não-dominância não é usado para calcular as adaptações dos indivíduos da população de soluções e, por conta disso, não exerce influência direta no processo de evolução das soluções. Em geral, um conjunto à parte contendo as soluções não-dominadas e seus respectivos vetores função objetivo é criado e mantido ao longo de toda a busca, constituindo-se numa aproximação das soluções e da fronteira de Pareto, respectivamente. Exemplos de algoritmos baseados nesta abordagem são o *Vector Evaluated Genetic Algorithm* (*VEGA*), o *Lexicographic ordering*, e AGs baseados na

Teoria de Jogos. Novamente, Coello (1999) e Van Veldhuizen e Lamont (2000) analisam com mais profundidade técnicas baseadas nesta abordagem.

Segundo Van Veldhuizen e Lamont (2000), até o ano de 1999 o número de pesquisas publicadas sobre Algoritmos Evolutivos Multiobjetivo era ao redor de 450 (baseado em Coello, 2005). Usando a mesma fonte (Coello, 2005), este número agora é maior do que 2000 (julho/2005), significando que nos últimos seis anos o número de pesquisas publicadas sobre Algoritmos Evolutivos Multiobjetivo mais do que quadruplicou. Significa também uma enorme expansão desse campo de pesquisas. A maioria dos novos Algoritmos Evolutivos Multiobjetivo produzidos neste período são versões especiais de AGs (Algoritmos Genéticos), tornando os AGs a abordagem mais usada para tal classe de problemas de otimização. AGs são algoritmos de otimização muito populares, com larga abrangência de uso como solucionadores para uma grande variedade de problemas. Quase todos eles compartilham como característica o uso de pelo menos três parâmetros de ajuste (por exemplo, tamanho da população, taxa de reprodução, taxa de mutação) que devem ser objeto de ajuste fino, a fim de obter-se bom desempenho. As versões multiobjetivo dos AGs geralmente têm pelo menos mais um parâmetro de ajuste: o parâmetro de nicho ou compartilhamento (Goicoechea et al., 1982; Van Veldhuizen e Lamont, 2000). A versão multiobjetivo do Algoritmo da Otimização Extrema Generalizada, chamada M-GEO, tem, assim como o GEO, apenas τ como parâmetro a ser ajustado.

6.2 - Formulação

A meta ao resolver-se um problema de otimização multiobjetivo é encontrar os valores viáveis das variáveis de projeto (ou soluções de projeto) que otimizam um vetor de funções objetivo. Matematicamente e sem perda de generalidade, o problema de otimização multiobjetivo pode ser descrito como (Tappeta et al., 2000; Coello, 1999; Srinivas e Deb, 1994):

$$\underset{\mathbf{X}}{\text{Minimizar:}} \quad \mathbf{F}(\mathbf{X}) = [\, F_1(\mathbf{X})\ F_2(\mathbf{X}) \ldots F_{NFOBJ}(\mathbf{X}) \,] \text{ (vetor de funções objetivo)} \qquad (1)$$

$$\text{Sujeito a:} \quad g_j(\mathbf{X}) \leq 0 \qquad j \in \{1,2,...,m\} \quad \text{(restrições de desigualdade)} \qquad (2.1)$$

$$h_k(\mathbf{X}) = 0 \qquad k \in \{1,2,...,l\} \qquad \text{(restrições de igualdade)} \qquad (2.2)$$

$$\mathbf{X}_{MIN} \leq \mathbf{X} \leq \mathbf{X}_{MAX} \qquad \text{(restrições laterais)} \qquad (2.3)$$

Onde: $\mathbf{X}^T = [\ X_1\ X_2\ X_3\ \ldots\ X_N\ ]$ (variáveis de projeto)

A principal diferença relativamente ao problema de otimização simples (monoobjetivo) formulado no Capítulo 2 é que ao invés de uma única função objetivo agora tem-se um vetor de funções objetivo, com dimensão dada por NFOBJ, ou seja, o número de funções objetivo que se deseja minimizar simultaneamente. Na verdade, o problema de otimização multiobjetivo pode ser pensado como uma generalização do problema de otimização simples. Desta forma, no âmbito da formulação geral multiobjetivo, um problema de otimização simples é um caso particular onde NFOBJ=1. Nesse caso, independentemente do número de ótimos globais existentes para o problema de otimização simples, a fronteira de Pareto resultante tem apenas um único ponto em um espaço de funções objetivo unidimensional. Caso o problema de otimização simples possua um único ótimo global, o conjunto de soluções de Pareto possui apenas uma única solução de projeto. Se o problema de otimização simples possuir vários ótimos globais, então o conjunto de soluções de Pareto tem tantos pontos quantos forem os ótimos globais, mas todos eles mapeiam um único ponto na fronteira de Pareto.

6.3 – M-GEO

Um fluxograma do algoritmo M-GEO é apresentado na Figura 6.1. Ele não faz uso de nenhum tipo de agregação das funções objetivo. Além disso, também não utiliza diretamente o conceito de não-dominância como critério para guiar a busca. Pertence, portanto, à terceira abordagem descrita na seção 6.1, a abordagem Pareto indireta.

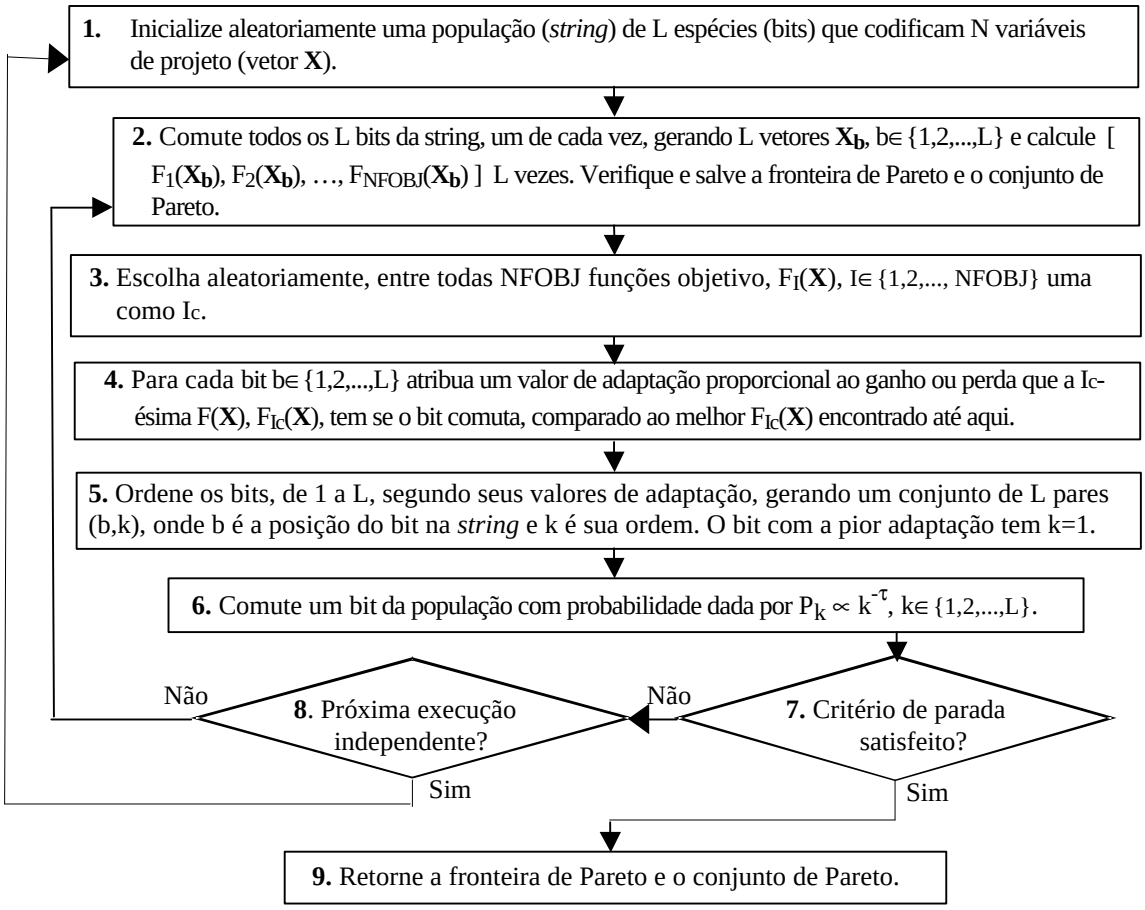


Figura 6.1 - O algoritmo M-GEO.

Em essência, a estratégia multiobjetivo do M-GEO é a estratégia de um torneio n-para-um, onde, a cada iteração do algoritmo, todas as NFOBJ funções objetivo competem pelo privilégio de serem usadas como sendo a função de atribuição de adaptação e apenas uma é escolhida. Como a escolha é totalmente aleatória (distribuição uniforme), teoricamente, qualquer seqüência de escolhas é possível, mesmo aquelas em que apenas uma função objetivo é usada como a função de atribuição de adaptação durante toda a busca. Esta aleatoriedade permite ao M-GEO acessar a totalidade da fronteira de Pareto, sendo que a eficiência esperada do algoritmo repousa na pressão de seleção imposta pelo parâmetro τ sobre todas as funções objetivo, a qual força os movimentos do algoritmo no espaço de objetivos na direção da fronteira de Pareto. Tal raciocínio pode ser melhor compreendido com o auxílio da Figura 6.2 a seguir. Nela, sem perda de generalidade, estão representados os pontos percorridos durante três buscas em um espaço biobjetivo, ambos para minimização. A primeira busca, identificada pela letra A, corresponde à situação

extrema onde o M-GEO sorteou sempre a F_1 para calcular as adaptações. Com isso, o algoritmo tende a convergir para a extremidade da fronteira de Pareto onde a F_1 é otimizada ao máximo. A segunda busca, identificada pela letra B, corresponde à situação extrema de uma busca onde o M-GEO sorteou sempre a F_2 para calcular as adaptações. Com isso, o algoritmo tende a convergir para a extremidade da fronteira de Pareto onde a F_2 é otimizada ao máximo. A terceira busca, identificada pela letra C, corresponde a uma busca qualquer, onde uma sequência arbitrária de alternâncias entre F_1 e F_2 ocorreu. Nesse caso, a cada iteração do M-GEO, ele tende para o extremo em F1 ou para o extremo em F2, em função de qual função objetivo (F1 ou F2) foi sorteada na iteração atual. O resultado esperado ao longo de várias iterações é um movimento aleatório em direção à fronteira de Pareto, com o M-GEO convergindo para um ponto qualquer da mesma, conforme indicado. Na Figura 6.2, as setas indicam a direção resultante para cada uma das buscas do exemplo. Cada direção expressa graficamente a pressão de seleção ocorrida. Os três exemplos descritos permitem visualizar o mecanismo pelo qual o M-GEO consegue capturar uma boa aproximação para a totalidade da fronteira de Pareto.

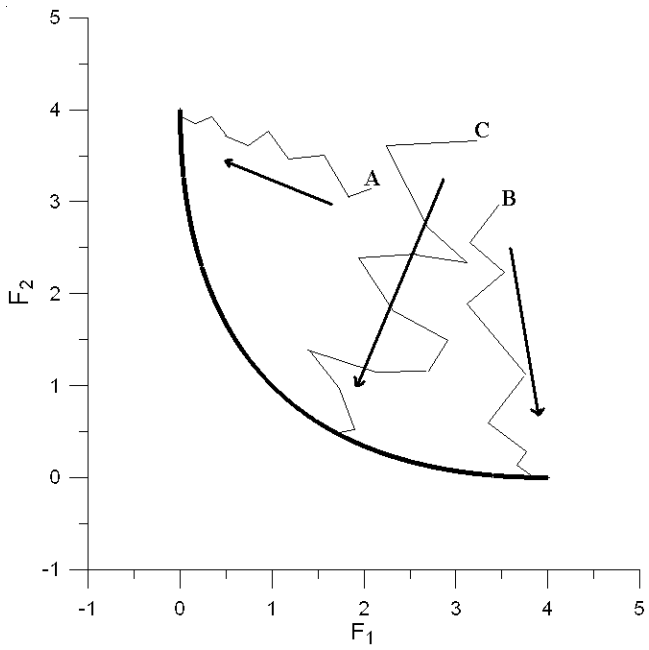


Figura 6.2 – Exemplos de buscas com o M-GEO.

A seguir, os passos do algoritmo M-GEO apresentados no fluxograma da Figura 6.1 são descritos. Relativamente à similaridade entre M-GEO e GEO, o segundo pode ser obtido a partir do primeiro apenas impondo NOBJ=1 no fluxograma da Figura 6.1. Neste

caso, o passo 3 torna-se redundante e pode ser omitido. Além disso, menções à “fronteira de Pareto” e “conjunto de Pareto” podem ser lidas como “melhor F(**X**)” e “vetor **X** que leva à melhor F(**X**)”, respectivamente.

O primeiro passo de M-GEO é idêntico ao primeiro passo de GEO: uma cadeia (*string*) de L bits é aleatoriamente preenchida com 0s and 1s. No segundo passo, todos os L bits são comutados, um de cada vez, gerando L vetores diferentes no espaço de busca, $\mathbf{X_b}$, $b \in \{1,2,\ldots,L\}$. "Um de cada vez" significa que antes do próximo bit ser comutado, o bit anterior é retornado à sua condição original (comutado uma segunda vez). Todas as NFOBJ funções objetivo são avaliadas para todos os L vetores $\mathbf{X_b}$, gerando L vetores função objetivo na forma $[F_1(\mathbf{X_b}), F_2(\mathbf{X_b}),\ldots, F_{NFOBJ}(\mathbf{X_b})]$. Ainda no segundo passo, o conjunto contendo a aproximação da fronteira de Pareto é criado, verificado, alterado e salvado, conforme se fizer necessário. No terceiro passo, uma função objetivo entre todas as NFOBJ é aleatoriamente escolhida. Uma vez escolhida uma função objetivo I_c, a busca ocorre (do quarto ao sétimo passo na Figura 6.1) exatamente como acontece numa busca com o GEO, onde o parâmetro τ impõe a pressão de seleção sobre a população de indivíduos (os L bits e seus L vetores $\mathbf{X_b}$ associados), baseado nas suas adaptações relativamente à função objetivo. Um dos indivíduos é escolhido (o bit associado é comutado), definindo desta forma, o movimento do algoritmo no espaço de busca. Dito de uma forma mais passo a passo, no quarto passo, para cada um dos L bits (e seus vetores X_b associados) um valor de adaptação é atribuído, baseado no ganho ou perda que a I_c-ésima função objetivo tem se o respectivo bit é comutado, comparado ao melhor valor encontrado até aqui para a I_c-ésima função objetivo. Matematicamente, o valor de adaptação é obtido calculando-se a diferença $\Delta F = F_{Ic}(\mathbf{X_b}) - \text{melhor } F_{Ic}(\mathbf{X})$, $b \varepsilon \{1,2,\ldots,L\}$, onde “melhor $F_{Ic}(\mathbf{X})$” é o “melhor valor até aqui” de $F_{Ic}(\mathbf{X})$ mencionado e é o melhor valor de $F_{Ic}(\mathbf{X})$ encontrado durante a busca até a iteração anterior do algoritmo (passos **2.** até **7.**). Para um problema de minimização, valores negativos de ΔF significam espécies precariamente adaptadas (bits), porque comutando-se um destes bits leva a projetos melhores. Então, no quinto passo, todos os bits são ordenados de acordo com seus valores de adaptação, de 1 até L, gerando um conjunto de pares (b,k), onde b é a posição linear do bit na cadeia (*string*) de bits e k é a sua respectiva ordem. O bit com a pior adaptação tem k=1 e o bit com a melhor adaptação tem k=L. Para problemas de minimização, o bit com o ΔF mais negativo (ou menos positivo)

tem ordem k=1, enquanto que para problemas de maximização o bit com ΔF mais positivo (ou menos negativo) tem ordem k=1. Em seguida, no sexto passo, um bit, $b_{escolhido}$, é escolhido e comutado, de acordo com uma probabilidade dada por $P_k \propto k^{-\tau}$. Na versão de M-GEO em questão, a tarefa do sexto passo é realizada por meio de dois sorteios com distribuição uniforme. O primeiro sorteio escolhe um valor para k e $P_k = k^{-\tau}$ é calculado. Então, o segundo sorteio retorna um valor real no intervalo 0~1. Se este valor é menor ou igual à P_k, o bit com ordem igual a k é comutado. Estes dois sorteios são repetidos tantas vezes quanto necessários até que um bit seja comutado. Quando um bit é comutado, ele torna-se o bit $b_{escolhido}$ e o sexto passo está completo. No sétimo passo, o critério de parada é verificado; se ele se verifica a fronteira de Pareto encontrada é apresentada e o algoritmo termina. Se não, o algoritmo vai para o oitavo passo e verifica se uma nova execução independente deve ser iniciada. Se sim, o algoritmo volta para o primeiro passo. Nesse caso, um novo ponto no espaço de busca é sorteado. A fronteira de Pareto, no entanto, é mantida. Se não, o algoritmo volta para o segundo passo e inicia outra iteração. O critério para definir se uma nova execução independente deve ser iniciada pode ser, por exemplo, o número de avaliações de **F**(**X**). Assim, se o critério de parada for NAF=10^5, o critério para nova execução independente pode ser, por exemplo, NAF=int(10^5/ni), onde NAF= número de avaliações de **F**(**X**) e ni é o número de execuções independentes desejado.

O número de execuções independentes foi incluído como parte intrínseca do M-GEO para propiciar robustez estatística ao funcionamento do algoritmo. A maioria dos autores, quando realiza testes em algoritmos estocásticos (seus ou de terceiros), utiliza entre 10 e 50 execuções independentes. Além disso, o uso de várias execuções independentes aumenta a possibilidade do M-GEO encontrar pontos situados ao longo de toda a extensão da fronteira de Pareto, por meio do mecanismo descrito com o auxílio da Figura 6.2.

6.4 – Resultados com as funções teste

Nesta seção, os resultados da aplicação do M-GEO a algumas funções teste são apresentados. O objetivo principal é o de comprovar a capacidade do algoritmo de encontrar apropriadamente a fronteira de Pareto. A Tabela 6.1 apresenta as funções teste

utilizadas. As mesmas foram escolhidas dentre funções teste utilizadas em publicações anteriores (Srinivas e Deb, 1994; Mason et al., 1998; Messac et al., 2000). Verificou-se a influência do parâmetro τ procedendo-se a busca pela fronteira de Pareto para $\tau \in \{0{,}5;2;4;8\}$. A escolha é arbitrária, cumprindo somente o propósito de evidenciar a influência de τ no desempenho do M-GEO. Apenas os resultados mais relevantes serão apresentados e discutidos.

Tabela 6.1 – Funções teste para minimização multiobjetivo

	Min. $\mathbf{F}(\mathbf{X})$	NFOBJ	N	Restrições
FTM_1	$F(\mathbf{X}) = \begin{bmatrix} e^{-X_1} + 1{,}4\, e^{-X_1^2} \\ e^{X_1} + 1{,}4\, e^{-X_1^2} \end{bmatrix}$	2	1	$-10 \le X_1 \le 10$ [1]
FTM_2	$F(\mathbf{X}) = \begin{bmatrix} f_1 = X_1^2 \\ f_2 = (X_1 - 2)^2 \end{bmatrix}$	2	1	$-10 \le X_1 \le 10$
FTM_3	$F(\mathbf{X}) = \begin{bmatrix} f_1 = \begin{cases} -X_1, \text{se } X_1 \le 1 \\ -2 + X_1, \text{se } 1 < X_1 \le 3 \\ 4 - X_1, \text{se } 3 < X_1 \le 4 \\ -4 + X_1, \text{se } X_1 > 4 \end{cases} \\ f_2 = (X_1 - 2)^2 \end{bmatrix}$	2	1	$-10 \le X_1 \le 10$
FTM_4	$F(\mathbf{X}) = \begin{bmatrix} f_1 = (X_1 - 2)^2 + (X_2 - 1)^2 + 2 \\ f_2 = 9X_1 - (X_2 - 1)^2 \end{bmatrix}$	2	2	$X_1^2 + X_2^2 - 225 \le 0$ $X_1 - 3X_2 + 10 \le 0$ $-20 \le X_i \le 20$, $i \in \{1,2\}$
FTM_5	$F(\mathbf{X}) = \begin{bmatrix} f_1 = (X_1 - 2)^2 + (X_2 - 2)^2 \\ f_2 = (X_1 + 2)^2 + (X_2 - 2)^2 \\ f_3 = (X_1 + 2)^2 + (X_2 + 2)^2 \\ f_4 = (X_1 - 2)^2 + (X_2 + 2)^2 \end{bmatrix}$	4	2	$-10 \le X_i \le 10$, $i \in \{1,2\}$

[1] Limites laterais definidos para os testes efetuados neste trabalho. Em Messac et al. (2000) tais limites não foram definidos.

A primeira função teste, aqui designada FTM_1, foi utilizada em Messac et al. (2000) para demonstrar que a abordagem das funções de agregação por soma ponderada não consegue capturar a totalidade da fronteira de Pareto desta função teste, enquanto outras funções de agregação conseguem. A Figura 6.3, extraída de Messac et al. (2000), ilustra a

Fronteira de Pareto da FTM_1 e as regiões capturáveis e não capturáveis ao utilizar-se uma abordagem de função de agregação de soma ponderada.

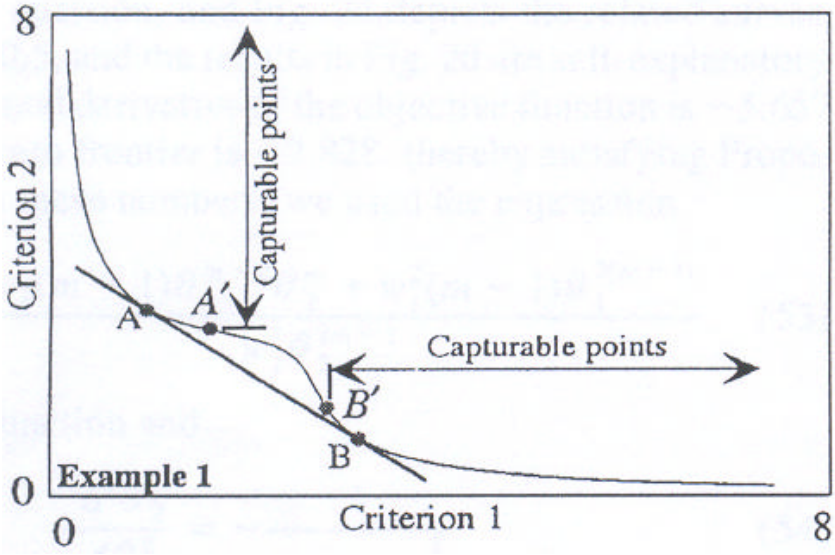


Figura 6.3 - Regiões da Fronteira de Pareto da FTM_1.
Fonte: Messac et al. (2000)

Em Messac et al. (2000), não foi informado o número de avaliações de F(**X**) utilizado. Após testes preliminares, decidiu-se usar um máximo de 5000 avaliações nos testes envolvendo FTM_1. Além disso, a abordagem lá empregada não utiliza codificação binária das variáveis de projeto. Assim, adotou-se l=16 bits e estabeleceu-se [-10, 10] como intervalo de variação da variável de projeto, o que implica uma resolução de $\sim 3 \cdot 10^{-4}$. O número de execuções independentes usado foi ni=50. A Figura 6.4 apresenta a fronteira de Pareto obtida por M-GEO para FTM_1 com $\tau \in \{4; 8\}$.

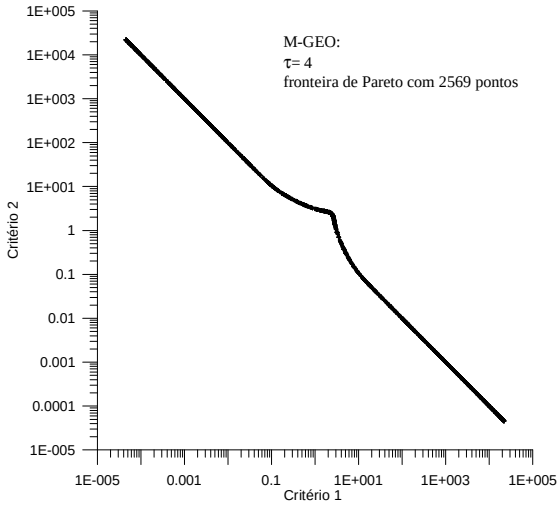


Figura 6.4a – Fronteira de Pareto FTM_1

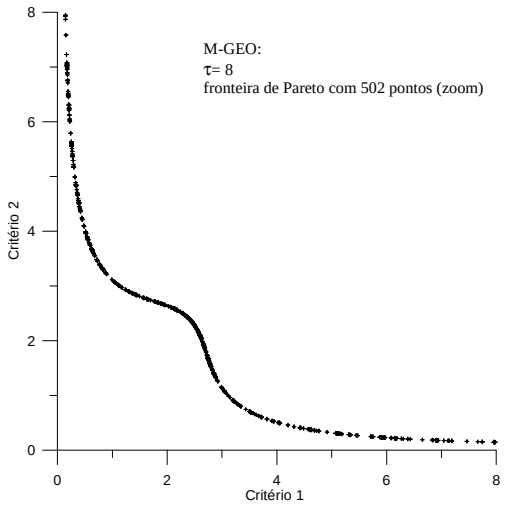


Figura 6.4b – Front. Pareto FTM_1 (zoom)

Na Figura 6.4a, a totalidade da fronteira de Pareto encontrada por M-GEO para a FTM_1 é apresentada. A fronteira mostrada foi obtida com τ=4 e foi aquela com o maior número de pontos (2569). A fronteira com o menor número de pontos, 2126, foi obtida com τ=0,5. Para FTM_1, τ não exerce grande influência sobre o desempenho de M-GEO, em termos de número de pontos obtidos para a fronteira de Pareto. Na Figura 6.4b, os mesmos limites e a mesma escala da Figura 6.2 foram utilizados para os eixos, a fim de propiciar comparação direta. Nesse caso, a fronteira com o maior número de pontos foi obtida com τ=8, 502 pontos, e a menor com τ=2, 388 pontos. É possível observar que o M-GEO não teve dificuldades em capturar pontos da fronteira de Pareto situados na região dos pontos não capturáveis por funções de agregação por soma ponderada.

As funções teste FTM_i, $i \in \{2,3,4,5\}$ foram utilizadas, entre outros, por Srinivas e Deb (1994) para testar o *NSGA - Non-Dominated Sorting Genetic Algorithm* e em Mason et al. (1998) para validar uma variante do *NSGA*. Com exceção da FTM_5, todas as funções teste possuem NFOBJ=2, facilitando a visualização gráfica da fronteira de Pareto (bidimensional). No caso da FTM_5, NFOBJ=4, o que implica uma fronteira de Pareto quadridimensional, não permitindo sua visualização gráfica, a não ser parcialmente.

Em Srinivas e Deb (1994) e também em Mason et al. (1998), um máximo de 50000 avaliações de $\mathbf{F}(\mathbf{X})$ foi usado nos testes envolvendo FTM_i, $i \in \{2,3,4,5\}$. A fim de possibilitar uma comparação equilibrada, esse mesmo valor foi utilizado como critério de parada para M-GEO nos testes envolvendo FTM_i, $i \in \{1,2,3,4\}$. Além disso, o número de execuções independentes foi fixado em ni=50 e o número de bits que codifica cada variável de projeto foi fixado em l=16.

A Figura 6.5 apresenta a fronteira de Pareto da FTM_2 obtida em Srinivas e Deb (1994) e também aquelas obtidas por M-GEO para $\tau \in \{0,5; 4; 8\}$.

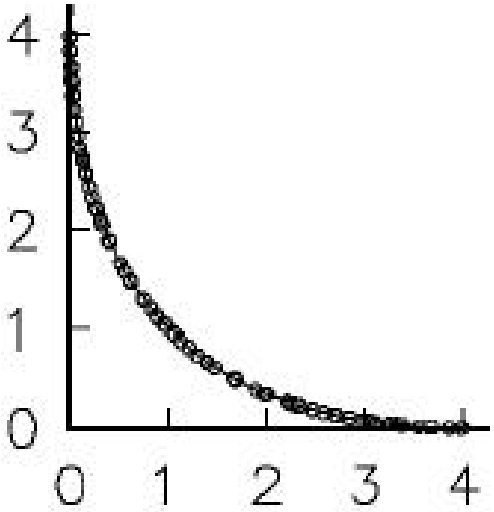


Figura 6.5a – F.P. FTM_2 (Srinivas e Deb, 1994)

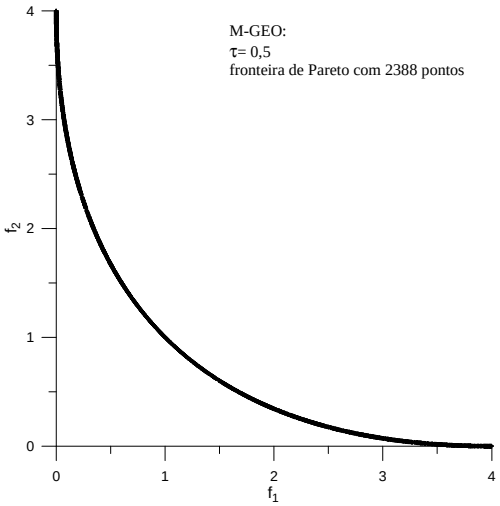


Figura 6.5b - F.P. FTM_2 (M-GEO,τ=0,5)

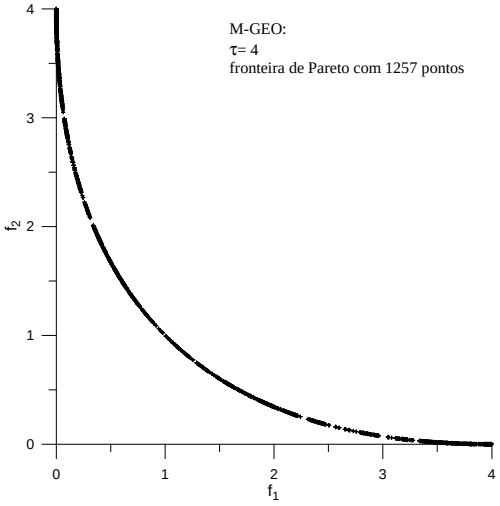


Figura 6.5c – F.P. FTM_2 (M-GEO,τ=4)

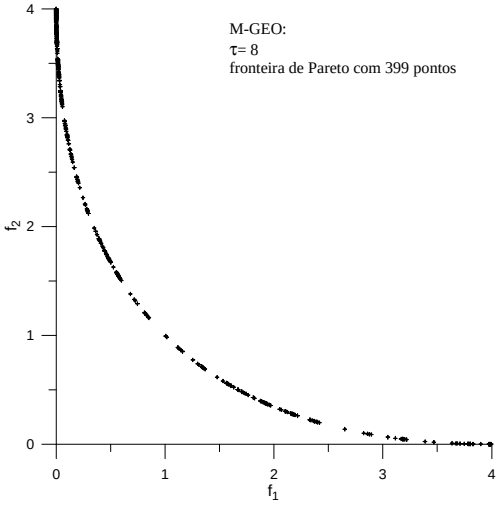


Figura 6.5d - F.P. FTM_2 (M-GEO,τ=8)

Olhando-se as Figuras 6.5b, 6.5c e 6.5d, observa-se que τ exerce uma influência significativa no desempenho de M-GEO para a FTM_2. Quanto maior τ, menor o número de pontos obtidos para a fronteira de Pareto, caindo de 2388 com τ=0,5 para apenas 399 com τ=8.

A Figura 6.6 apresenta a fronteira de Pareto da FTM_3 obtida em Srinivas e Deb (1994) e também aquelas obtidas por M-GEO para $\tau \in \{0,5; 2; 4\}$.

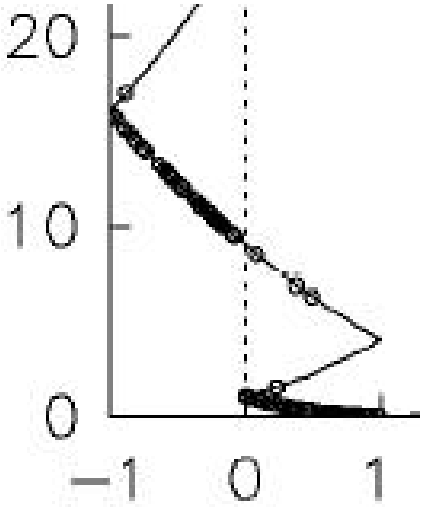


Figura 6.6a – F.P. FTM_3 (Srinivas e Deb, 1994)

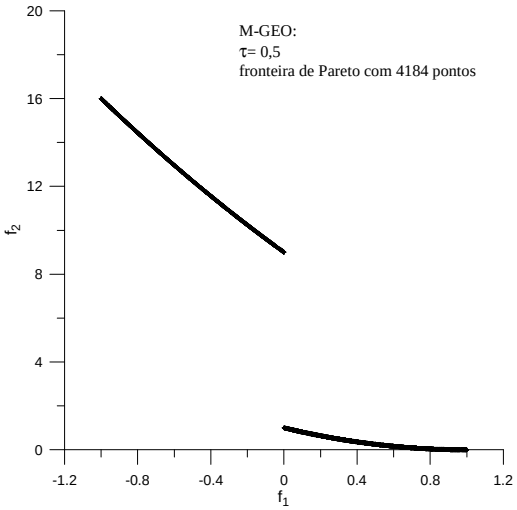


Figura 6.6b - F.P. FTM_3 (M-GEO,τ=0,5)

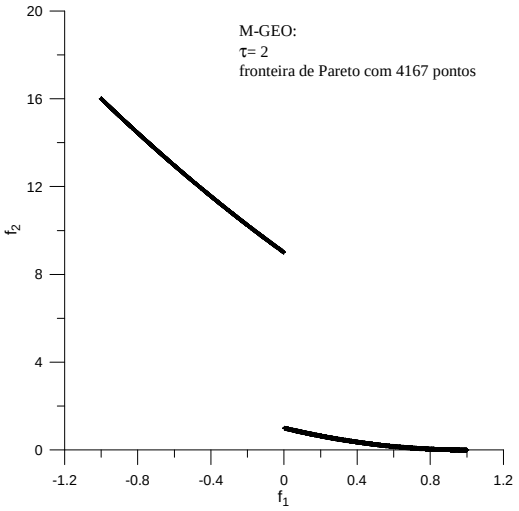


Figura 6.6c – F.P. FTM_3 (M-GEO,τ=2)

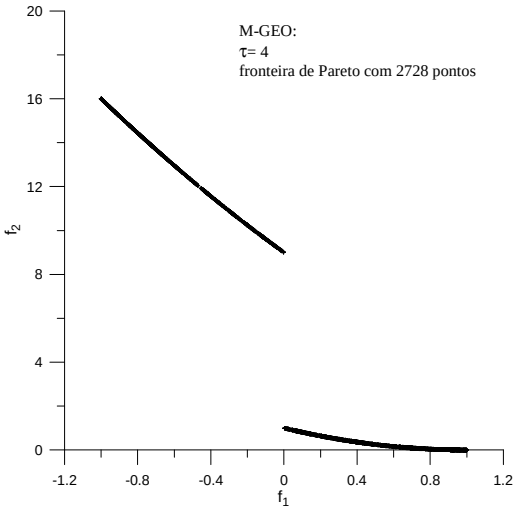


Figura 6.6d - F.P. FTM_3 (M-GEO,τ=4)

Na Figura 6.6a, é possível observar a existência de cinco pontos que estão claramente dominados pelos demais, não pertencendo à fronteira de Pareto. Este fato é possível, no caso do NSGA, porque o mesmo usa um critério de seleção baseado em níveis de dominância, onde os indivíduos (pontos) menos dominados da população (fronteira) são selecionados preferencialmente, mas nada impede, no entanto, que indivíduos dominados sejam mantidos como parte da população do NSGA ao longo do processo de busca. Este fato pode, na verdade, ser considerado um efeito colateral, que acomete todos os métodos de otimização multiobjetivo baseados na abordagem Pareto direta. No caso do M-GEO, o fenômeno observado na Figura 6.6a não é possível, pois na abordagem Pareto indireta, o processo de seleção e o processo de manutenção da fronteira de Pareto são independentes, o que permite manter, ao longo de toda a busca, uma aproximação para a fronteira de Pareto que segue rigorosamente o critério de não

dominância, ou seja, apenas os pontos não dominados são mantidos. Olhando-se as Figuras 6.6b, 6.6c e 6.6d, observa-se que também para FTM_3 τ exerce uma influência significativa no desempenho de M-GEO. Novamente, quanto maior τ, menor o número de pontos obtidos para a fronteira de Pareto, caindo de 4184 com τ=0,5 para 2728 com τ=4. Para τ=8, resultado não mostrado nas figuras, o número de pontos foi ainda menor, 2059 pontos.

A Figura 6.7 apresenta a fronteira de Pareto da FTM_4 obtida por M-GEO para $\tau \in \{0,5; 2; 4, 8\}$. Dentre as funções teste utilizadas, a FTM_4 é a única que possui outras restrições, além das restrições laterais (vide Tabela 6.1). Procura, deste modo, testar o desempenho dos algoritmos aos quais é submetida em problemas com restrições, muito comuns em aplicações práticas.

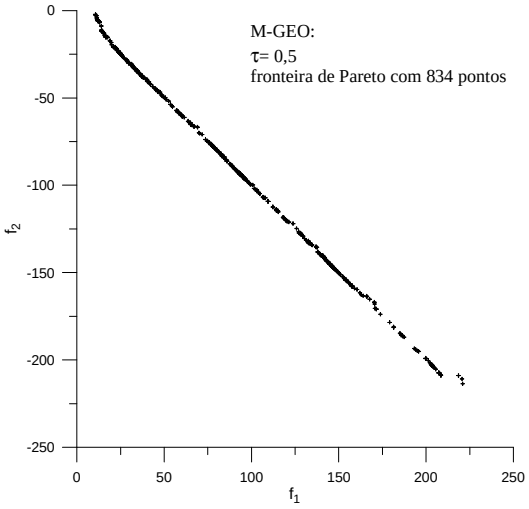


Figura 6.7a - F.P. FTM_4 (M-GEO,τ=0,5)

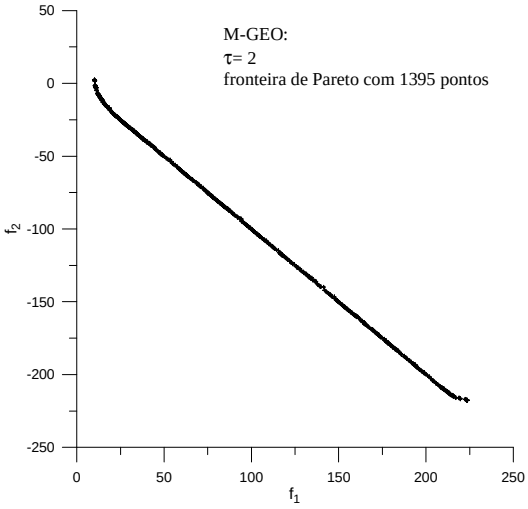


Figura 6.7b - F.P. FTM_4 (M-GEO,τ=2)

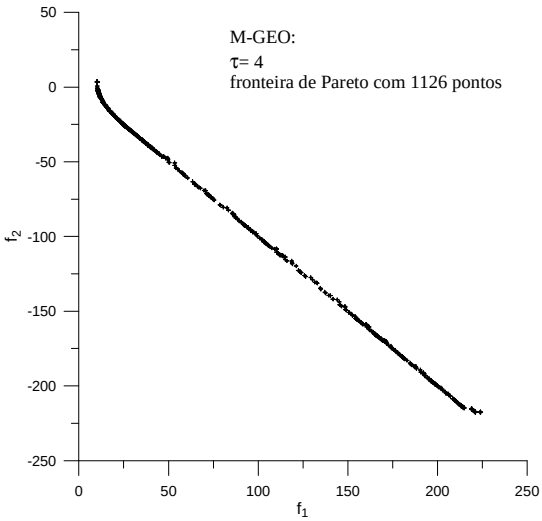


Figura 6.7c – F.P. FTM_4 (M-GEO,τ=4)

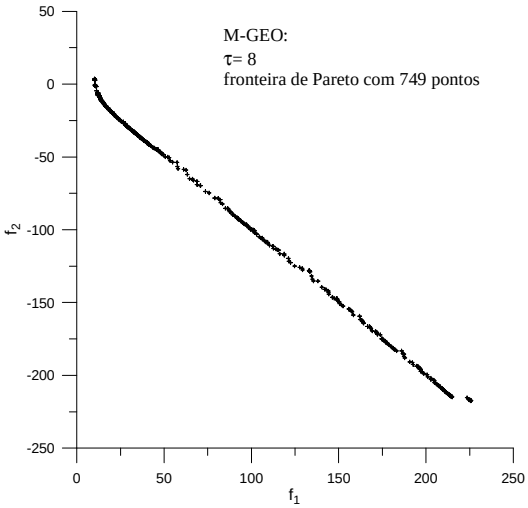


Figura 6.7d - F.P. FTM_4 (M-GEO,τ=8)

Olhando-se as Figuras 6.7a, 6.7b, 6.7c e 6.7d, observa-se que também para FTM_4 τ exerce uma influência significativa no desempenho de M-GEO. Agora, tanto valores muito pequenos quanto valores muito grandes para τ acarretam degradação na fronteira de Pareto obtida por M-GEO, fato que pode ser percebido nas figuras tanto visualmente quanto pelo número de pontos obtidos para a fronteira de Pareto. Para a FTM_4, portanto, o τ ótimo deve situar-se no entorno de τ=2.

Adicionalmente, a FTM_4 também foi estudada no espaço das variáveis de projeto. A Figura 6.8 apresenta o conjunto de Pareto (abreviado por c.P. nos títulos das figuras) obtido por Srinivas e Deb (1994), por Mason et al. (1998) e também aqueles obtidos com M-GEO para $\tau \in \{0{,}5;\ 2;\ 8\}$. O conjunto de Pareto, também chamado de soluções de Pareto, consiste dos pontos no espaço das variáveis de projeto que formam, no espaço das funções objetivo, os pontos da fronteira de Pareto.

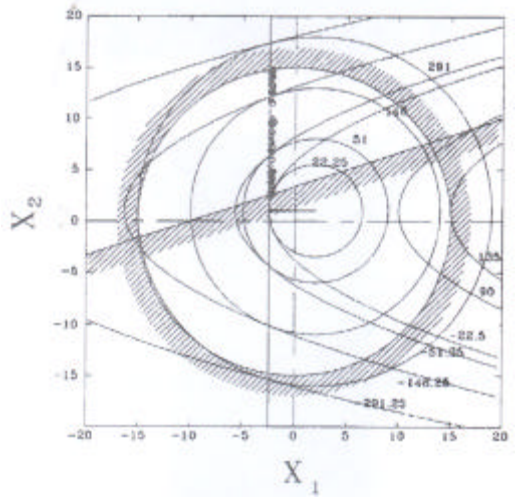


Figura 6.8a – c.P. FTM_4 (Srinivas e Deb, 1994)

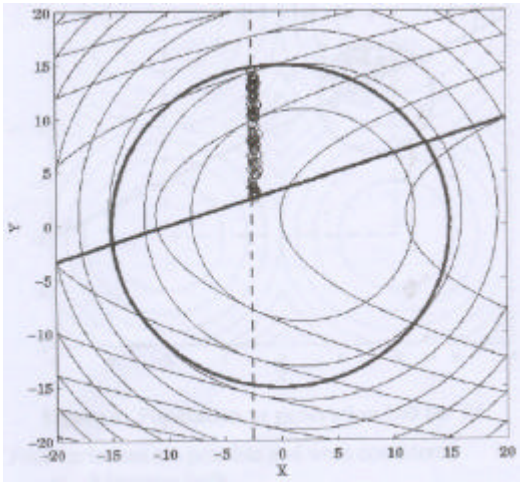


Figura 6.8b - c.P. FTM_4 (Mason et al., 1998)

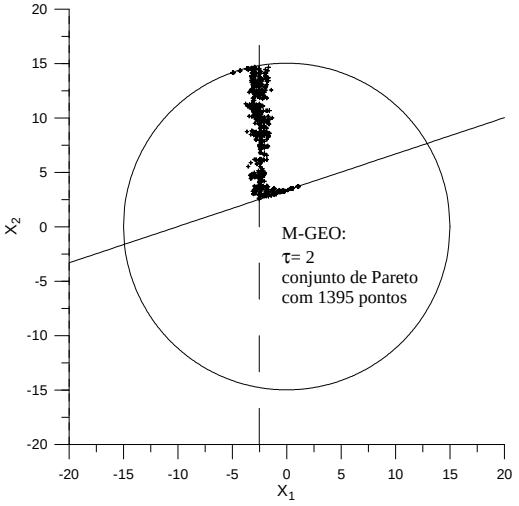


Figura 6.8c – c.P. FTM_4 (M-GEO,τ=2)

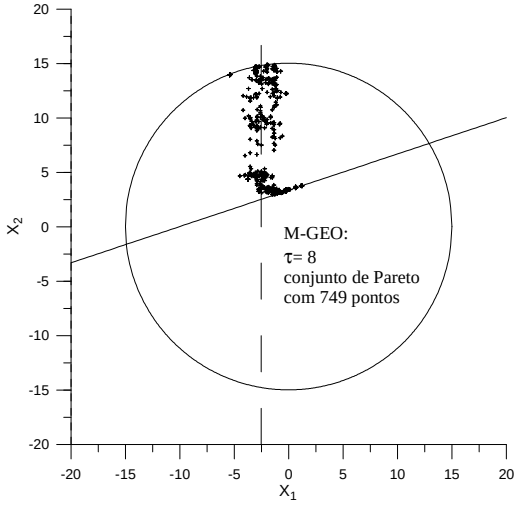


Figura 6.8d - c.P. FTM_4 (M-GEO,τ=8)

Na Figura 6.8a, a reta inclinada e o círculo, ambos hachurados, representam as restrições (não laterais) do problema. As restrições impõem que as soluções de Pareto estejam dentro do círculo e acima da reta inclinada. Estão representadas também na Figura 6.8a curvas de contorno para um mesmo valor das funções objetivo f_1 (circunferências) e f_2 (parábolas). As circunferências de contorno de f_1 todas têm $(X_1;X_2)=(2;1)$ como centro, enquanto as parábolas de contorno de f_2 têm $X_2=1$ como eixo de simetria. Srinivas e Deb (1994) deduziram analiticamente que as soluções de Pareto encontram-se ao longo do eixo $X_1=2,5$ (linha tracejada nas Figuras 6.8b, 6.8c, e 6.8d). O eixo $X_1=2,5$ representa o lugar geométrico onde as circunferências de contorno de f_1 tangenciam as parábolas de contorno de f_2. Analisando agora o conjunto de Pareto correspondente à melhor fronteira obtida por M-GEO ($\tau=2$, Figura 6.8c), é possível observar que, além dos pontos situados ao longo do eixo $X_1=2,5$ (linha tracejada longa) existem pontos adicionais, que se distribuem ao longo dos limites inferior (reta inclinada em linha contínua) e superior (circunferência em linha contínua) das restrições. Como estes pontos estão em desacordo com o equacionamento analítico proposto por Srinivas e Deb (1994), uma nova busca foi efetuada por M-GEO, dessa vez com um total de 10^6 avaliações de F(**X**) como critério de parada e ni=100. O resultado é apresentado na Figura 6.9 a seguir.

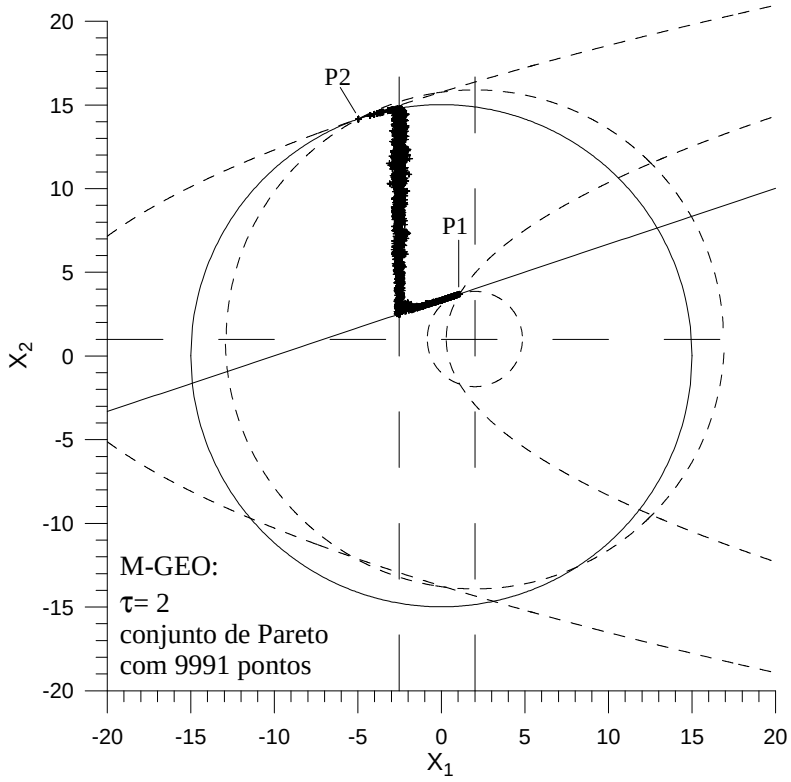


Figura 6.9 – Conjunto de Pareto FTM_4 (M-GEO, $\tau=2$)

Quando comparado àquele da Figura 6.8c, o novo resultado apresenta os pontos situados mais próximos ao eixo X_1=2,5. Entretanto, os pontos situados sobre as restrições inferior (reta inclinada em linha contínua) e superior (circunferência em linha contínua) aparecem ainda mais claramente. Após uma análise minuciosa, descobriu-se que tais pontos são realmente parte do conjunto de Pareto e correspondem às soluções não dominadas que surgem quando uma das restrições de desigualdade da FTM_4 está ativa (encontra-se no limite da igualdade). A solução analítica proposta em Srinivas e Deb (1994) é válida apenas quando não há restrições de desigualdade ativas. Na Figura 6.9, as curvas de contorno de valor constante de f_1 e f_2 estão representadas para os pontos do conjunto de Pareto assinalados (P1 e P2). As curvas de contorno de f_1 são circunferências centradas em (X_1;X_2)=(2;1) e tal que, quanto maior seu diâmetro maior f_1. O ponto P1, (X_1;X_2)=(1,1;3,7), ocorre quando a circunferência de contorno da f_1 tangencia a reta inclinada, tornando esta restrição ativa e a solução que P1 representa, viável. O ponto P1 corresponde ao ponto extremo da fronteira de Pareto para o qual f_1 atinge seu valor mínimo (f_1=10,1 e f_2=2,6). Graficamente, é possível perceber que valores menores para f_1 não são possíveis, pois implicariam circunferências de contorno de f_1 ainda menores e estas não possuem nenhum ponto dentro ou mesmo no limite da região viável. As curvas de contorno de f_2 são parábolas com eixo de simetria em X_2=1 (reta horizontal tracejada longa) e tais que, quanto mais à esquerda a parábola menor o valor de f_2. O ponto P2, (X_1;X_2)=(-4,98;14,15), ocorre quando a parábola de contorno da f_2 tangencia a circunferência de linha contínua centrada em (X_1;X_2)=(0;0), tornando esta restrição ativa e a solução que P2 representa, viável. O ponto P2 corresponde ao ponto extremo da fronteira de Pareto para o qual f_2 atinge seu valor mínimo (f_1=223,6 e f_2= -217,7). Mais uma vez, é possível visualizar graficamente que valores menores para f_2 não são possíveis, pois implicariam parábolas de contorno de f_2 ainda maiores e estas não possuem nenhum ponto dentro ou mesmo no limite da região viável.

Curiosamente, nem em Srinivas e Deb (1994), nem em Mason et al. (1998) as soluções de Pareto, encontradas pelos algoritmos lá testados, continham pontos com restrições ativas. Da mesma forma que com M-GEO a ocorrência de tais pontos motivou análises subseqüentes e a confirmação de sua existência, talvez sua não ocorrência nas soluções obtidas em Srinivas e Deb (1994) e Mason et al. (1998) seja um dos fatores responsáveis por nenhum dos autores citados ter percebido que a solução analítica proposta era incompleta.

Para a FTM_5 a análise dos resultados é feita somente no espaço das variáveis de projeto. Conforme já mencionado, a fronteira de Pareto da FTM_5 é quadridimensional, o que dificulta sua visualização. O espaço das variáveis de projeto, por outro lado, é bidimensional e a área que origina a fronteira de Pareto é perfeitamente conhecida e de fácil visualização. A Figura 6.10 apresenta o conjunto de Pareto (abreviado por c.P. nos títulos das figuras) obtido em Mason et al. (1998) e também aqueles obtidos com M-GEO para $\tau \in \{0{,}5;\ 2;\ 4\}$. Nas figuras, o conjunto de soluções de Pareto consiste dos pontos compreendidos no quadrado delimitado por {(-2;-2) ; (2;-2) ; (2;2) , (-2;2)}. Os vértices do quadrado são também os pontos de mínimo de cada uma das funções objetivo. Na Figura 6.10a circunferências concêntricas formam as curvas de contorno de valor constante das quatro funções objetivo.

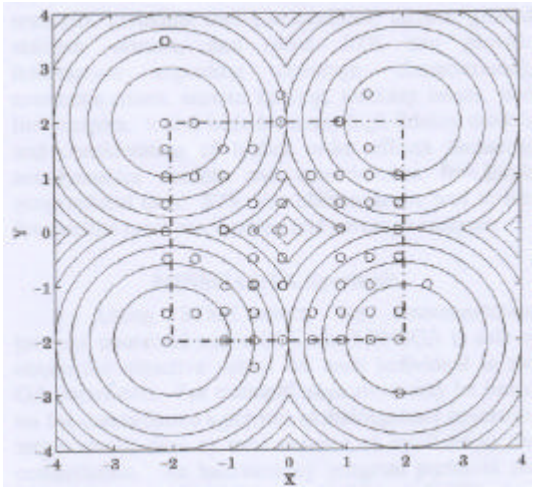


Figura 6.10a – c.P. FTM_5 (Mason et al., 1998)

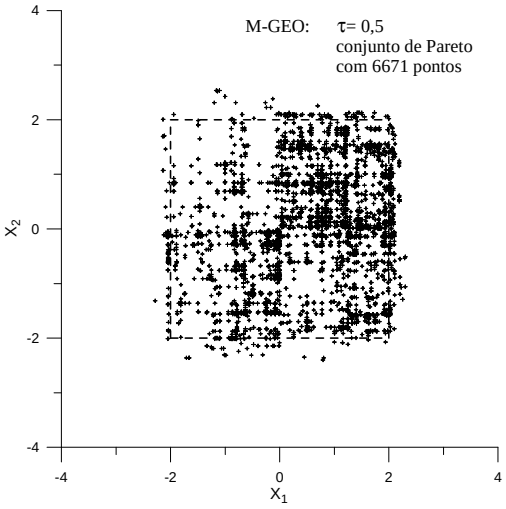


Figura 6.10b - c.P. FTM_5 (M-GEO,τ=0,5)

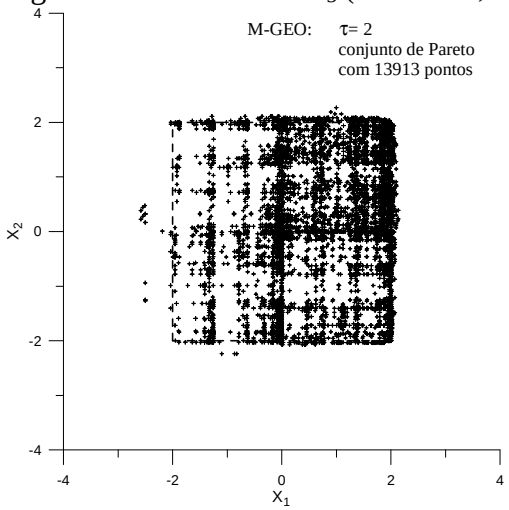


Figura 6.10c – c.P. FTM_5 (M-GEO,τ=2)

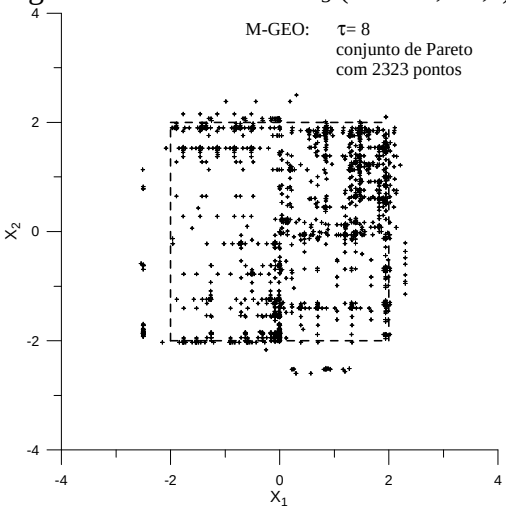


Figura 6.10d - c.P. FTM_5 (M-GEO,τ=8)

Olhando-se as Figuras 6.10b, 6.10c e 6.10d, observa-se que também para FTM_5 τ exerce uma influência significativa no desempenho de M-GEO. O comportamento é similar àquele observado para FTM_4. Mais uma vez, tanto valores muito pequenos quanto valores muito grandes para τ acarretam degradação nas soluções de Pareto obtidas por M-GEO, fato que pode ser percebido nas figuras tanto visualmente quanto pelo número de pontos obtidos. Para τ=4, cujo resultado não é mostrado graficamente, a fronteira de Pareto foi obtida com 8919 pontos. Portanto, assim como ocorre para FTM_4, também para FTM_5 o τ ótimo deve situar-se no entorno de τ=2.

6.5 – Conclusões

Este Capítulo descreveu M-GEO, uma versão multiobjetivo do GEO e a sua aplicação a algumas funções teste para validação. A conclusão mais importante é a de que os objetivos foram atingidos satisfatoriamente, com o algoritmo M-GEO tendo sido capaz de gerar as fronteiras de Pareto aproximadas para os problemas propostos, obtendo as soluções associadas. Especificamente no caso do teste envolvendo uma função teste com restrições de desigualdade adicionais e não apenas restrições laterais, o M-GEO demonstrou ser mais robusto do que os algoritmos de otimização multiobjetivo aos quais foi comparado, pois, diferentemente dos outros, encontrou soluções que fazem parte da fronteira de Pareto quando as restrições adicionais encontram-se ativas. Adicionalmente, é válido mencionar que, a partir dos testes executados, é possível constatar que, a exemplo do que ocorre com o GEO, também com o M-GEO existe para cada F(**X**) um valor do parâmetro τ para o qual a eficiência da busca é a melhor de todas. Em outras palavras, existe um τ ótimo do M-GEO para cada F(**X**).

CAPÍTULO 7

APLICAÇÕES

7.1 - Introdução

A fim de comprovar a capacidade de otimização das versões desenvolvidas em aplicações práticas, algumas delas foram utilizadas para a otimização de dois sistemas espaciais: (i) o projeto das áreas dos radiadores da Plataforma Multimissão (PMM) do Inpe e; (ii) o projeto da configuração de uma constelação de satélites de sensoriamento remoto.

No caso da PMM, o problema foi formulado e resolvido como um problema monoobjetivo, usando o GEO, e também como um problema multiobjetivo, usando o M-GEO. Dos dois jeitos, as soluções obtidas foram bastante satisfatórias. A versão do GEO utilizada foi a versão SA1 (vide Capítulo 3, seção 3.7.1). Dentro deste Capítulo, as referências a GEO e GEO_{var} devem ser entendidas como GEO_1 e GEO_{var1}.

No caso da constelação de satélites, o problema foi posto e resolvido apenas na forma multiobjetivo. A aplicação reportada refere-se ao projeto ótimo de uma pequena constelação de satélites do tipo *Multi-Application Purpose Synthetic Aperture Radar* (MAPSAR). O satélite MAPSAR é um projeto conjunto Brasil-Alemanha.

7.2 - Projeto otimizado do sistema de controle térmico da Plataforma Multimissão (PMM)

Todo satélite, qualquer que seja sua carga útil, deve poder realizar funções tais como:

- suporte estrutural para montagem de equipamentos;
- suprimento de potência elétrica à carga útil;
- controle de órbita e propulsão;
- comunicações de serviço (telemetria/telecomandos/localização);
- gestão de dados a bordo;
- controle térmico.

A PMM é uma plataforma espacial multiuso, desenvolvida pelo INPE, que deve ser usada em diferentes tipos de missões, tais como observação da Terra, científica e meteorológica. A PMM é um conceito de arquitetura de satélites que reúne em uma plataforma todos os equipamentos que desempenham funções necessárias à sobrevivência de um satélite independente do tipo de órbita ou de apontamento, ou seja, independente da sua carga útil. Neste tipo de arquitetura, existe uma separação física entre os módulos de serviço (PMM) e de carga útil, que podem então ser desenvolvidos, construídos e testados separadamente, antes da integração dos módulos e teste finais. Existe também a vantagem da reutilização do projeto da plataforma, e redução dos custos de desenvolvimento de novos satélites. A Figura 7.1 apresenta uma vista em perspectiva da PMM (futuro módulo de serviço do SCD3 e de outros satélites a serem construídos) na sua configuração operacional, já acoplada com a carga útil. A Figura 7.2 mostra a configuração interna e a Tabela 7.1 apresenta as características da PMM.

Tabela 7.1 – Características da Plataforma Multimissão

Característica	**Valor**
Dimensões	1 m x 1 m x 1 m
Massa	250 kg
Consumo de Potência	150W
Potência fornecida à Carga Útil	175W (80W durante eclipse)
Inclinações de órbita	de 0º a 90º
Altitudes de órbita	de 400 Km a 1500 Km
Atitude	Apontamento para a Terra, para o Sol, ou inercial
Capacidade de Manobras	ΔV=150m/s
Capacidade de Armazenamento de dados	5 Gbits

Fonte: *http://www.inpe.br/programas/mecb/default.htm*

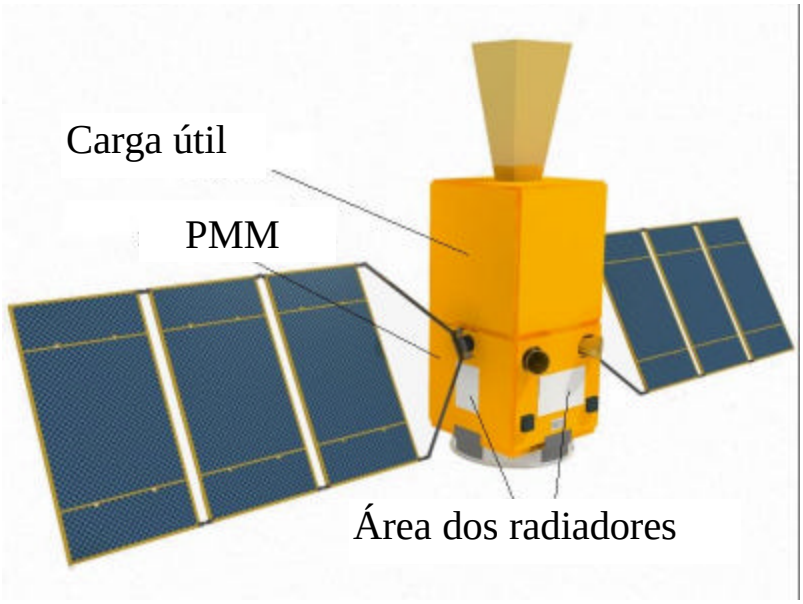


Figura 7.1 – Esquema da PMM com uma carga útil.

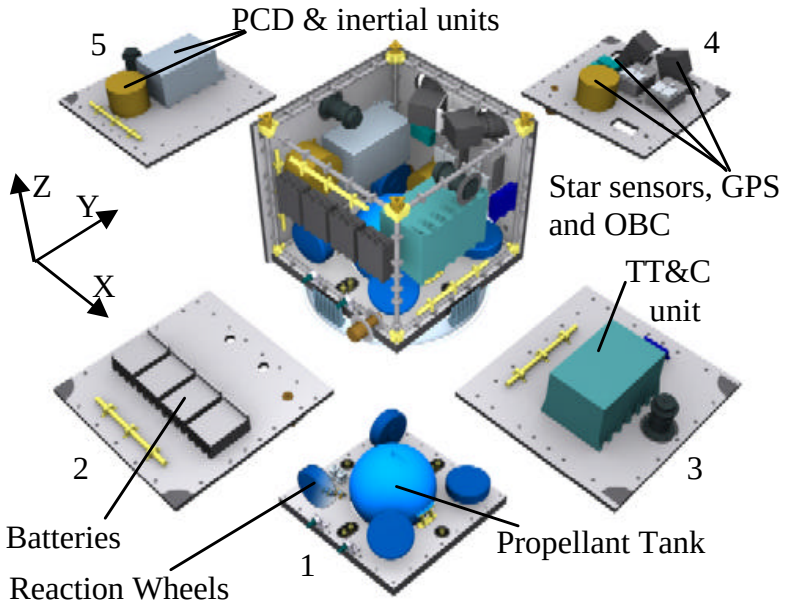


Figura 7.2 – Vista simplificada da PMM e dos painéis 1 a 5 com alguns componentes internos.

O modelamento térmico de satélites envolve diversas tarefas de pré-processamento. Em uma delas, os dados do satélite (geométricos, térmicos, orbitais e de atitude) são transformados em um sistema de equações diferenciais, o qual representa o comportamento térmico do satélite quando de sua operação em órbita. A maioria das análises térmicas é efetuada considerando-se condições de regime permanente com cargas térmicas orbitais médias. Neste caso, o sistema de equações diferenciais reduz-se a um sistema de equações algébricas. A solução deste sistema fornece a distribuição de temperatura do satélite.

Geralmente, o pré-processamento consome um tempo computacional muito maior do que aquele gasto na obtenção da distribuição de temperatura do satélite.

Na abordagem de projeto tradicional, o analista térmico procura por uma solução de projeto de maneira interativa, resolvendo o problema direto. Isto significa que para cada configuração de projeto candidata as temperaturas são calculadas de acordo com a seqüência descrita no parágrafo anterior.

O processo de otimização é um problema inverso no qual, para uma distribuição de temperaturas desejada, informada no início do processo, uma solução de projeto térmico é buscada. Neste processo, o problema direto deve ser resolvido muitas vezes. Se um modelo detalhado é usado, o tempo computacional torna-se muito longo e o processo de otimização impraticável, a menos que recursos computacionais de altíssimo desempenho estejam disponíveis.

O objetivo principal do projeto térmico de um satélite é o de manter a temperatura dos elementos do veículo dentro dos limites requeridos para funcionamento apropriado. Existem muitas técnicas e dispositivos térmicos usados para atingir esta meta para as diferentes partes e equipamentos do satélite (Gilmore, 1994; Karam, 1998).

No caso da PMM, a concepção do controle térmico baseia-se em meios passivos, com as seguintes linhas mestras para o projeto (Muraoka et al., 2006):

(i) A superfície externa dos painéis 1, 2, 3, 4 e 5 são cobertas com *Multi-Layer Insulation blankets* (mantas *MLI*), deixando algumas áreas para o radiador térmico;

(ii) Todas as superfícies internas são pintadas de preto, exceto o painel das baterias. Este último é coberto com mantas *MLI*;

(iii) O painel das baterias é condutivamente isolado do resto da PMM;

(iv) A PMM é termicamente isolada do módulo com a carga útil e;

(v) Um aquecedor é usado para controle de temperatura das baterias.

A abordagem recém descrita é comum a todas as missões previstas para a PMM. Entretanto, o dimensionamento da área do radiador em cada painel (vide Figura 7.1) deve ser feito para cada missão específica, a fim de cumprir os requisitos de temperatura, sendo este um dos aspectos mais importantes com o qual um engenheiro térmico de satélites tem de lidar. Radiadores são áreas do satélite cobertas com revestimento de alta emissividade, tal que eles possam irradiar calor para o espaço a fim de manter a temperatura dos equipamentos do satélite dentro dos intervalos especificados por projeto durante períodos de alta dissipação de calor nos equipamentos eletrônicos e/ou altas cargas térmicas externas. Por outro lado, estas áreas não devem ser excessivamente grandes para que, durante períodos de baixas cargas térmicas, as temperaturas não caiam abaixo do mínimo permitido. Como as áreas dos radiadores são fixas, atuando tanto nos casos de alta como nos casos de baixa dissipação térmica, então é necessário projetar as mesmas obedecendo a uma relação de compromisso entre as duas situações antagônicas. No caso da PMM, radiadores podem ser posicionados em 5 dos 6 lados do corpo da plataforma, uma vez que o lado de cima não "vê" o espaço devido a carga útil ser montada naquele lado (vide Figura 7.1).

No projeto térmico de satélites geralmente são identificadas duas situações críticas, onde a ocorrência das temperaturas mínimas e máximas é esperada: i) O Caso Frio (CF), quando as cargas térmicas externas (radiação solar, radiação terrestre e albedo) são mínimas, o satélite está operando com a mais baixa dissipação nos equipamentos eletrônicos, e as propriedades termo-óticas de seus revestimentos ainda não estão degradadas e; ii) O Caso Quente (CQ), quando a radiação térmica externa é máxima, o satélite está em operação com a maior dissipação térmica e as propriedades termo-óticas dos revestimentos estão degradadas. O projeto térmico deve gerir o fluxo de calor de forma que, em ambas as situações, as temperaturas de todos os elementos permaneçam dentro dos intervalos requeridos. Existem muitas variáveis, tais como o tamanho dos radiadores, que afetam a distribuição de temperatura, e o engenheiro térmico deve encontrar uma combinação destas variáveis que resulte num projeto satisfatório. Esta tarefa necessita de muitas simulações e análises, considerando o grande número de variáveis envolvidas.

Em equipamentos onde um controle severo de temperatura se faz necessário, tal como nas baterias da PMM (vide Figura 7.2), freqüentemente são usados aquecedores para o equipamento durante o CF. Por outro lado, como a fonte de energia elétrica é bastante

limitada a bordo de um satélite, a energia gasta nos aquecedores deve ser a menor possível. É importante ressaltar que a necessidade de um aquecedor durante o CF existe principalmente por causa da observância à relação de compromisso entre CF e CQ na hora de dimensionar as áreas dos radiadores. Em outras palavras, se os radiadores pudessem ser dimensionados para atender somente o CF, na maioria das vezes não haveria necessidade de aquecedor.

O objetivo da aplicação do GEO no projeto térmico da PMM é o de reduzir o tempo gasto pelo engenheiro para encontrar, não apenas uma solução satisfatória, mas também uma solução otimizada, um processo que tradicionalmente é feito "manualmente", com o projetista térmico executando múltiplos casos de análise.

7.2.1 - Formulação do problema de otimização

Na aplicação em questão, o objetivo específico é otimizar as áreas dos cinco radiadores e a potência necessária para o aquecedor das baterias. Uma solução consiste, portanto, de um conjunto de 5 áreas de radiadores e uma potência de um aquecedor.

O problema de otimização consiste, variando-se as áreas dos radiadores em cada painel, em minimizar a diferença entre as temperaturas calculadas para o CF e o CQ e temperaturas alvo definidas para cada painel, e ao mesmo tempo minimizar a potência dissipada pelo aquecedor das baterias. Ademais, restrições são impostas às temperaturas dos painéis, as quais devem permanecer dentro de intervalos requeridos para o projeto. A temperatura alvo e o intervalo de temperaturas permitido para cada painel são definidos como uma função dos requisitos térmicos dos equipamentos montados no respectivo painel.

Matematicamente, o problema de otimização é formulado como segue.

$$\text{Minimize: } F(\mathbf{X}) = \| \mathbf{T_{CF}}(\mathbf{X}) - \mathbf{T_A} \|_2 + \| \mathbf{T_{CQ}}(\mathbf{X}) - \mathbf{T_A} \|_2 + X_6 \qquad (7.1)$$

$$\text{sujeito a: } \mathbf{X_{MIN}} \leq \mathbf{X} \leq \mathbf{X_{MAX}}$$

$$\mathbf{T_{MIN}} \leq \mathbf{T_{CF}}(\mathbf{X}) \leq \mathbf{T_{MAX}}$$

$$\mathbf{T_{MIN}} \leq \mathbf{T_{CQ}}(\mathbf{X}[1{:}5]) \leq \mathbf{T_{MAX}}$$

onde:

$\mathbf{X}_{1:5}$ = área do radiador nos painéis 1 a 5, respectivamente; (m^2)

X_6 = potência do aquecedor das baterias (painel 2) durante o CF; (W)

$\mathbf{T_{CF}(X)}_{1:6}$ = temperaturas do CF nos painéis 1 a 6, respectivamente; (°C)

$\mathbf{T_{CQ}(X)}_{1:6}$ = temperaturas do CQ nos painéis 1 a 6, respectivamente; (°C)

$\mathbf{T_A}_{1:6}$ = temperaturas alvo nos painéis 1 a 6, respectivamente; (°C)

$\mathbf{X_{MIN}}_{1:5}$ = áreas mínimas dos radiadores 1 a 5, respectivamente; (m^2)

X_{MIN_6} = potência mínima do aquecedor das baterias; (W)

$\mathbf{X_{MAX}}_{1:5}$ = áreas máximas dos radiadores 1 a 5, respectivamente; (m^2)

X_{MAX_6} = potência máxima do aquecedor das baterias; (W)

$\mathbf{T_{MIN}}_{1:6}$ = temperaturas mínimas admissíveis nos painéis 1 a 6, respectivamente; (°C)

$\mathbf{T_{MAX}}_{1:6}$ = temperaturas máximas admissíveis nos painéis 1 a 6, respectivamente; (°C)

Daqui em diante, o problema de otimização completo recém formulado (equação 7.1) é denominado Caso Frio e Quente (CFQ).

Adicionalmente, CF e CQ são formulados como problemas de otimização e resolvidos separadamente. A idéia é obter-se informação adicional pela comparação das soluções em separado de CF e CQ com a solução completa CFQ. Matematicamente, os problemas de otimização CF e CQ são definidos conforme segue.

Caso Frio (CF):

Minimize: $F(\mathbf{X}) = \|\mathbf{T_{CF}(X)} - \mathbf{T_A}\|_2$ (7.2)

Sujeito a: $\mathbf{X_{MIN}} \leq \mathbf{X} \leq \mathbf{X_{MAX}}$

$\mathbf{T_{MIN}} \leq \mathbf{T_{CF}(X)} \leq \mathbf{T_{MAX}}$

Caso Quente (CQ):

Minimize: $F(\mathbf{X}) = \|\mathbf{T_{CQ}(X)} - \mathbf{T_A}\|_2$ (7.3)

Sujeito a: $\mathbf{X_{MIN}} \leq \mathbf{X} \leq \mathbf{X_{MAX}}$

$\mathbf{T_{MIN}} \leq \mathbf{T_{CQ}(X)} \leq \mathbf{T_{MAX}}$

É importante observar que no CF e no CQ recém formulados, $\mathbf{X}= \mathbf{X}_{1:5}$ = área dos radiadores nos painéis 1 a 5, respectivamente, (não há X_6). A razão disso é que não existe aquecedor das baterias sendo usado no CF quando formulado separadamente.

7.2.2 - Modelo térmico simplificado da PMM

O modelo térmico simplificado, necessário para obtenção de $\mathbf{T_{CQ}}$ e $\mathbf{T_{CF}}$, foi elaborado considerando somente os seis painéis laterais da PMM, com os equipamentos sendo simulados como fontes de calor sobre os seus respectivos painéis. Desse modo, a dissipação interna em cada painel é o somatório de dissipação de calor dos dispositivos montados no mesmo.Os painéis 1 a 5 trocam calor entre si, por condução e/ou radiação, e com o ambiente espacial, por radiação, através dos radiadores colocados sobre os mesmos. O painel superior (não visível na Figura 7.1) faz a interface com a carga útil e está termicamente isolado da mesma, mas troca calor com os outros painéis da PMM. O intervalo de temperaturas permitido para cada painel é obtido considerando-se os limites impostos pelos equipamentos montados internamente sobre ele. As mantas *MLI* aplicadas sobre a superfície externa dos seis painéis são consideradas ideais, isto é, os seis painéis trocam calor com o ambiente espacial através das áreas dos radiadores somente.

Modelos térmicos de satélites usualmente são obtidos usando-se a formulação dos parâmetros concentrados (Gilmore, 1994). Neste método, o satélite é dividido em um número de massas concentradas, chamadas nós, as quais são consideradas isotérmicas. Uma rede térmica é construída conectando os nós. Uma expressão governante da energia térmica é escrita para cada nó, resultando em um sistema de equações acopladas cuja solução são as temperaturas dos nós.

Utilizando a representação de parâmetros concentrados (Gilmore, 1994) e considerando uma situação de regime permanente com cargas térmicas orbitais médias, o balanço de calor em cada um dos seis painéis leva ao conjunto de equações:

$$\sum_{j=1}^{6} \mathbf{G}_{k,j}\left(\mathbf{T}_k - \mathbf{T}_j\right) + \sum_{j=1}^{6} \mathbf{R}_{k,j}\left(\mathbf{T}_k^4 - \mathbf{T}_j^4\right) + (1-d_{k,6})\,e\,\mathbf{X}_k\left(\mathbf{T}_k^4 - \mathbf{T}_\infty^4\right) = \tag{7.4}$$

$$(1-d_{k,6})\,(\mathbf{q}_{ik} + a\,\mathbf{X}_k\,\mathbf{q}_{sk} + e\,\mathbf{X}_k\,\mathbf{q}_{tk}) + d_{k,2}\mathbf{X}_6 \quad , \quad k \in \{1,2,\ldots,6\}$$

onde:

$\mathbf{X}_{1:5}$ = área do radiador nos painéis 1 a 5, respectivamente; (m^2)

$\mathbf{X}_6$ = potência do aquecedor das baterias (painel 2) durante o CF; (W)

$\mathbf{T}_{1:6}$ = temperaturas nos painéis 1 a 6, respectivamente; (K)

$\mathbf{q_i}_{1:5}$ = geração interna de calor nos painéis 1 a 5, respectivamente; (W)

$\mathbf{q_s}_{1:5}$ = fluxo de radiação solar nos painéis 1 a 5, respectivamente; (W/m^2)

$\mathbf{q_t}_{1:5}$ = fluxo de radiação terrestre nos painéis 1 a 5, respectivamente; (W/m^2)

$\mathbf{R}_{i,j}$ = condutância radiativa entre o painel i e o painel j; (W/K^4)

$\mathbf{G}_{i,j}$ = condutância condutiva entre o painel i e o painel j; (W/K)

α = absortividade solar;

e = emissividade no infravermelho;

$\mathbf{T}_\infty$ = temperatura de um corpo negro no espaço distante; (K)

$d_{k,j}$ = delta de kronecker;

As condutâncias radiativas $\mathbf{R}_{i,j}$, para i=1 até 6 e j=1 até 6 (painéis principais da PMM), estão relacionados à troca de calor por radiação entre as superfícies internas dos painéis. Seus valores foram determinados usando o módulo pré-processador do software PCTER (vide Figura 7.3), considerando uma emissividade de e=0,8 (tinta preta) para os nós 1,3,4,5,6 e e=0,1 (manta *MLI*) para o nó 2. As condutâncias condutivas $\mathbf{G}_{i,j}$ (para i=1 a 6 e j=1 a 6) foram determinadas a partir das características das juntas aparafusadas entre os painéis.

A solução do sistema algébrico de equações 7.4, no qual as variáveis de projeto (componentes do vetor **X**) aparecem explicitamente, resulta nas temperaturas de cada painel do satélite. No CFQ, para um dado vetor **X**, o sistema de equações é resolvido duas vezes, uma para as condições do CF (valores mínimos de $\mathbf{q_i}$, $\mathbf{q_s}$, $\mathbf{q_t}$, α e e) resultando em $\mathbf{T_{CF}}(\mathbf{X})_{1:6}$ e outra para as condições do CQ (valores máximos de $\mathbf{q_i}$, $\mathbf{q_s}$, $\mathbf{q_t}$, α e e) resultando em $\mathbf{T_{CQ}}(\mathbf{X})_{1:6}$.

O modelo térmico simplificado da PMM, descrito pelas equações 7.4, foi implementado usando o pacote de software térmico PCTER (Cardoso et al, 1990), desenvolvido no INPE. Este pacote cobre todo o ciclo de tarefas necessárias para a análise térmica de satélites, incluindo o cálculo de fatores de forma, condutâncias condutivas e radiativas, fluxos de radiação externos e temperaturas em regime permanente e em regime transitório. A Figura 7.3 mostra um diagrama esquemático do

PCTER e dos módulos que o compõem. O módulo ANATER é o responsável pelo cálculo das temperaturas, tendo sido acoplado ao GEO a fim de permitir o cálculo das temperaturas para cada nova configuração das áreas dos radiadores gerada durante a busca. Menciona-se, a fim de evitar confusão, que o módulo "GEO" que aparece na Figura 7.3 não representa o algoritmo GEO e sim o módulo do PCTER responsável pelos cálculos geométricos realizados durante o pré-processamento. As temperaturas do satélite são calculadas para a condição de regime permanente usando cargas térmicas orbitais médias obtidas por meio do módulo CATER.

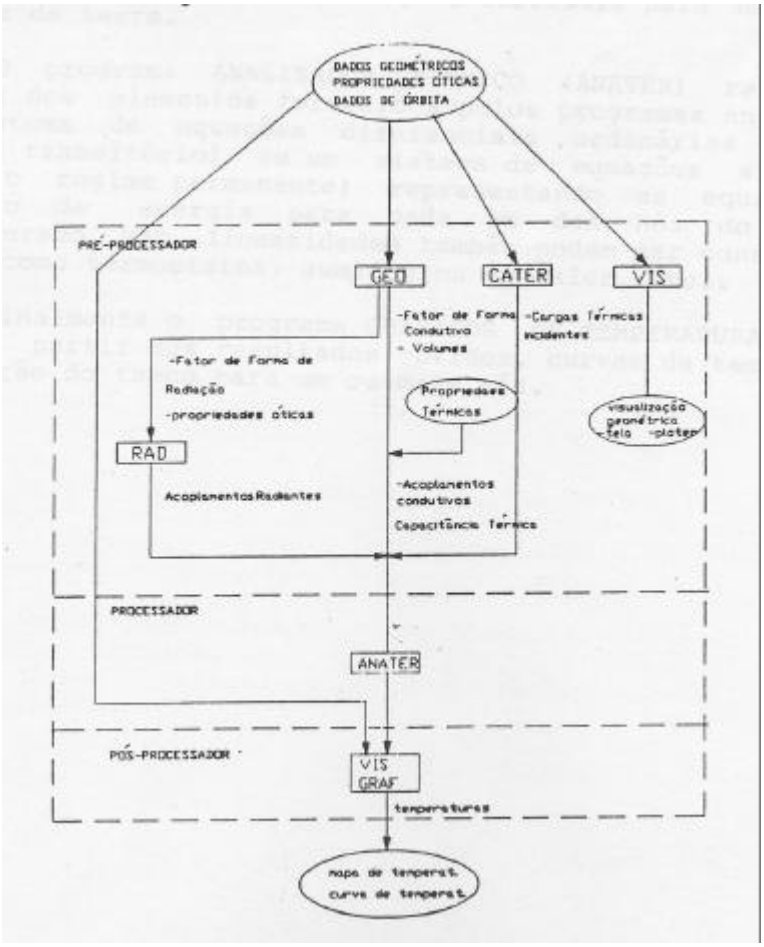


Figura 7.3 – Diagrama do software PCTER.

7.2.3 - Características da missão analisada

No estudo ora descrito, uma missão particular da PMM foi analisada, definida por uma órbita equatorial com altitude de 600 km e com o painel 2 (painel das baterias, -X, vide Figura 7.2) sempre apontando para a Terra ao longo da órbita. A Tabela 7.2 resume as principais características da missão.

Tabela 7.2 Características da Missão Analisada

Característica	Valor
Órbita	Equatorial (inclinação = 15^o)
Altitude	600 km
Orientação	Painel 2 sempre apontado para a Terra (CQ) ou painel 1 sempre apontado para o Sol (CF)
Intervalo do ângulo Beta	0° a $+38^o$
Período	5800 s
Duração do eclipse	2188 s – 2344 s

O projeto térmico do satélite baseia-se na análise de casos críticos, isto é, aqueles que geram condições térmicas extremas. Do ponto de vista do ambiente externo, dois casos foram identificados. O primeiro é o CF, que ocorre quando o Sol está no plano da órbita do satélite (ângulo beta é 0°) e o painel 1 (painel inferior, -Z) aponta sempre para o Sol ao longo. Ele combina uma órbita com o menor período de iluminação solar direta (eclipse de máxima duração) com a menor área do satélite projetada na direção da luz solar. Este caso está ilustrado na Figura 7.4a. O segundo caso crítico é o CQ, que ocorre quando o ângulo beta atinge seu valor máximo (+38°) e combina o maior período de iluminação com a maior área projetada na direção do sol. Este caso está ilustrado na Figura 7.4b.

Figura 7.4a – Órbita e atitude da missão analisada (CF)

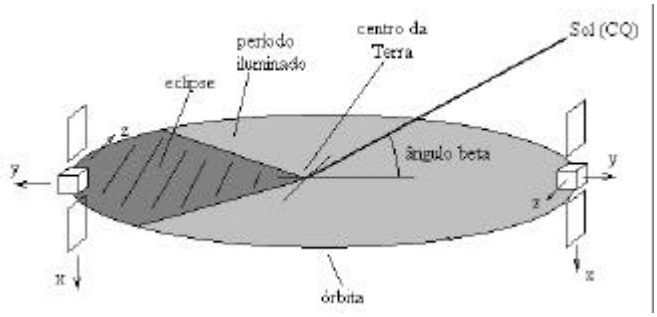


Figura 7.4b – Órbita e atitude da missão analisada (CQ)

Para cada caso crítico, as radiações médias incidentes ($\mathbf{q_s}$ e $\mathbf{q_t}$) em cada painel são calculadas pelo módulo CATER (vide Figura 7.3), considerando os seguintes valores dos parâmetros: (i) constante solar igual a 1326 W/m^2 para CF e 1418 W/m^2 para CQ; (ii) fator de albedo igual a 0,20 para CF e 0,28 para CQ; (iii) radiação terrestre igual a 233 W/m^2 para CF e 265 W/m^2 para CQ. Os valores de absortividade e emissividade dos radiadores (vide equação 7.4) foram fixados em a=0,14 para CF e a=0,2 para CQ e e=0,8 para CF e CQ. Em adição às cargas externas, a dissipação de calor dos equipamentos dentro da PMM é considerada de acordo com os valores na Tabela 7.3. A Tabela 7.3 sumariza os limites para as variáveis de projeto, os limites operacionais de temperatura, bem com as dissipações de calor internas devidas aos dispositivos eletrônicos que são aplicadas aos painéis.

Tabela 7.3 - Limites operacionais, das variáveis de projeto e dissipação de calor nos painéis.

Parâmetro		Painel					
		1	2	3	4	5	6 [1]
Área limite dos radiadores (m^2) [2]	X_{min}	0,0	0,0	0,0	0,0	0,0	-
	X_{max}	0,902	0,952	0,952	0,952	0,952	-
Limites de temperatura (^{o}C) [3]	T_{min}	-5,0	-10,0	-20,0	-20,0	-10,0	-20,0
	T_{max}	+50,0	+20,0	+50,0	+45,0	+45,0	+50,0
Temperatura alvo (^{o}C)	T_A	+22,5	+15,0	+15,0	+12,5	+17,5	+15,0
Dissipação interna de calor (W) [4]	CF	15,0	13, 0[(5)]	8,6	40,0	20,0	0,0
	CQ	40,0	47,5	27,2	55,0	90,5	0,0

[1]Painel superior da PMM, termicamente isolado da carga útil. [2]O sexto elemento deste vetor existe apenas no CFQ e não é uma área de radiador. Ele representa a potência do aquecedor no painel 2, com os seguintes limites: $X_{MIN_6} = 0$ W e $X_{MAX_6} = 65$ W. [3]Os limites de temperatura de cada painel foram determinados de acordo com o equipamento mais restritivo montado no mesmo. [4]A dissipação de calor interna de cada painel é o somatório das dissipações de todos os equipamentos montados no respectivo painel. [5]Em adição à dissipação mínima de calor deste painel, o aquecedor das baterias dissipa no intervalo $0.0 \leq X_6 \leq 65.0$ W.

7.2.4 - Resultados

Cada variável de projeto foi codificada em 7 bits, o que significa uma resolução melhor do que 0,01 m^2 e aproximadamente 0,5 W para as áreas dos radiadores e a potência do aquecedor, respectivamente.

A busca pelo projeto ótimo foi feita para diferentes valores de τ no intervalo [0,00; 5,00]. Como o problema é computacionalmente custoso, o número de avaliações de F(**X**) por execução foi limitado a 5 x 10^3 e uma média de 25 execuções independentes foi tomada. A Tabela 7.4 mostra os valores $\tau = \tau^*$ para os quais os melhores resultados foram obtidos.

Tabela 7.4 – τ^* encontrados para CF, CQ e para CFQ.

Caso	CF		CQ		CFQ	
Algoritmo	GEO	GEO_{var}	GEO	GEO_{var}	GEO	GEO_{var}
τ^*	2,7	3,2	1,3	1,3	2,0	2,3

Após isso, os valores τ^* foram usados e 25 execuções independentes foram realizadas para CF, CQ e CFQ, e onde cada execução independente usou 10^5 avaliações de F(**X**). As melhores soluções de projeto obtidas por GEO e GEO_{var} para cada caso estão na Tabela 7.5

Tabela 7.5 – Melhores soluções para CF, CQ e CFQ.

Caso	Algoritmo	F(**X**)	X_1 m^2	X_2 m^2	X_3 m^2	X_4 m^2	X_5 m^2	X_6 W
CF	GEO	4,018	0,001	0,062	0,048	0,251	0,030	-
	GEO_{var}	4,360	0,008	0,062	0,035	0,271	0,022	-
CQ	GEO	1,487	0,028	0,433	0,143	0,359	0,457	-
	GEO_{var}	1,789	0,028	0,433	0,129	0,372	0,457	-
CFQ	GEO	142,1	0,838	0,433	0,055	0,042	0,044	57,3
	GEO_{var}	142,5	0,676	0,460	0,089	0,008	0,072	49,7

A partir dos resultados mostrados na Tabela 7.5 pode ser visto que ambos GEO e GEO_{var} encontraram soluções de projeto muito parecidas para CF e para CQ. Para CFQ, as soluções diferem consideravelmente, mas ambas possuem valores de F(**X**) muito próximos, significando, possivelmente, a existência de diversas soluções quase ótimas.

As Figuras 7.5a e 7.5b apresentam, para GEO e GEO_{var}, respectivamente, as áreas de radiador obtidas para CF, CQ e CFQ. O limite inferior e superior para a área de radiador de cada painel também são mostrados. As soluções obtidas para CFQ, quando comparadas com as obtidas para CF e CQ em separado, mostram que uma solução intuitiva, baseada na interpolação entre CF e CQ, não levaria à solução obtida em CFQ. Este fato é mais

evidente olhando-se, nas Figuras 7.5a e 7.5b, o painel 1. Nele, as áreas dadas por CF e CQ são ambas próximas a zero, enquanto a área de CFQ é maior que 0,6 m^2. No painel 4, ocorre o oposto, as áreas dadas por CF e CQ estão ambas entre 0,2 m^2 e 0,4 m^2 e para CFQ é igual ou menor do que 0,05m^2.

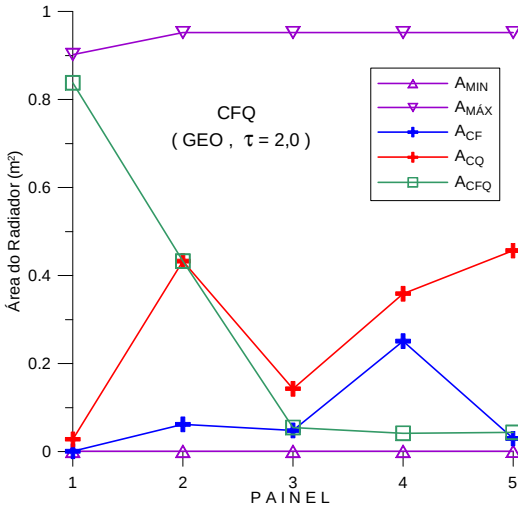


Figura 7.5a – Áreas dos radiadores (GEO)

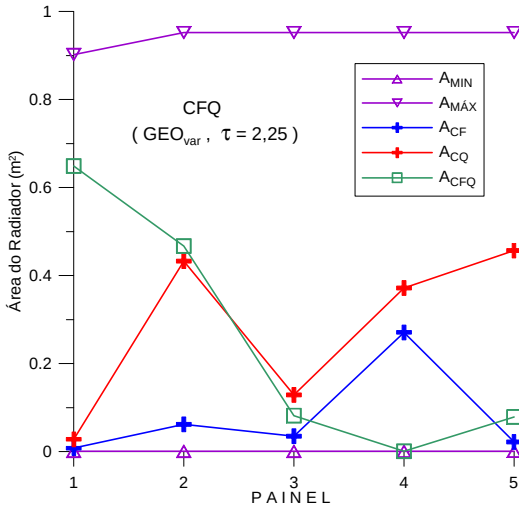


Figura 7.5b - Áreas dos radiadores (GEO_{var})

As temperaturas nos painéis, calculadas com os dados da Tabela 7.6 são mostradas nas Figuras 7.6 a 7.8. Pode ser visto facilmente que, para CF e CQ, as soluções de projeto otimizadas trazem as temperaturas resultantes nos painéis muito próximas das temperaturas alvo. Este não é o caso para CFQ, onde a maioria das temperaturas está distante das temperaturas alvo. De fato, algumas delas estão na temperatura limite do respectivo painel. Apesar de terem valores diferentes nas variáveis de projeto, ambas as soluções para CFQ tiveram os painéis 1, 2, e 5 como aqueles onde as temperaturas estão no limite.

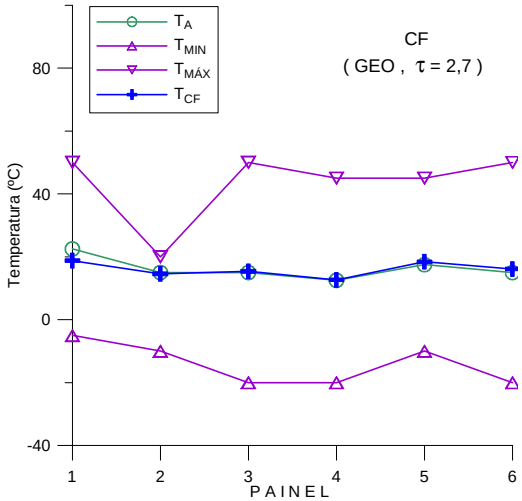


Figura 7.6a – Temperaturas CF (GEO)

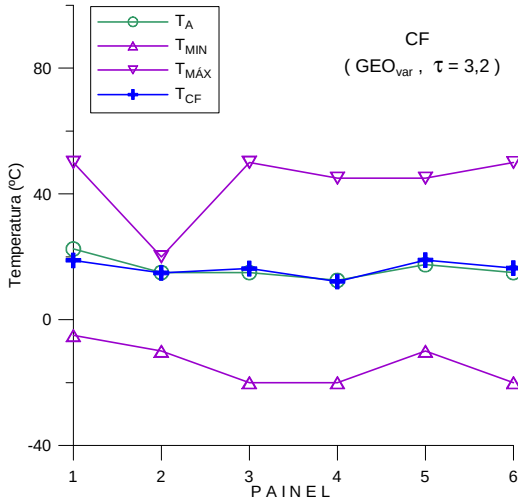


Figura 7.6b - Temperaturas CF (GEO_{var})

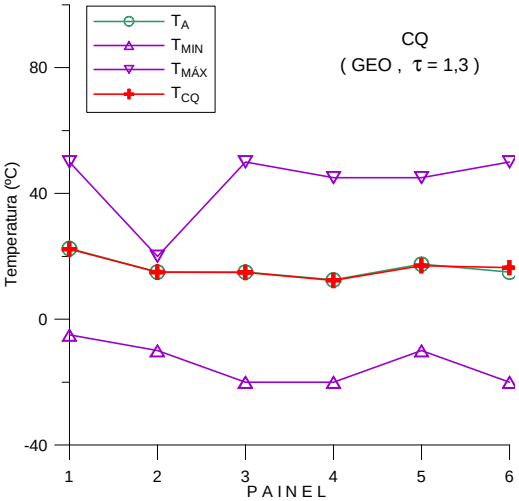


Figura 7.7a – Temperaturas CQ (GEO)

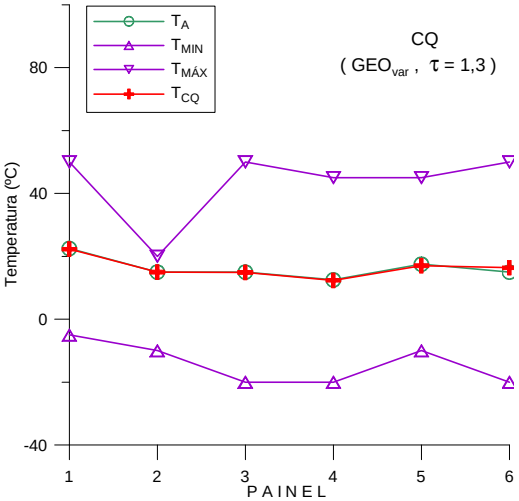


Figura 7.7b - Temperaturas CQ (GEO_{var})

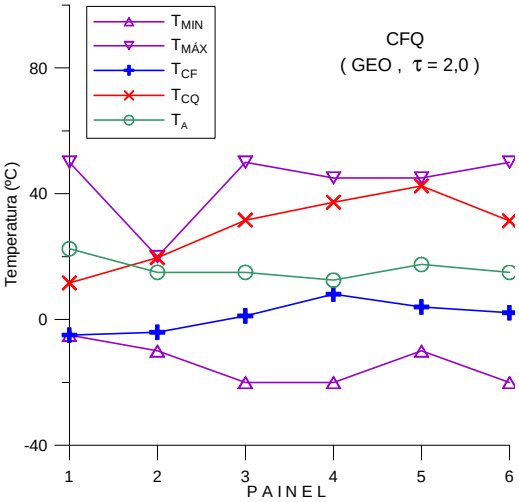


Figura 7.8a – Temperaturas CFQ (GEO)

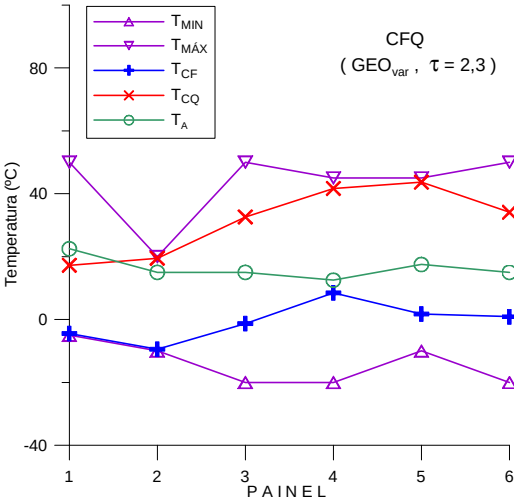


Figura 7.8b - Temperaturas CFQ (GEO_{var})

Outra observação importante é que, embora seja numericamente muito próxima em termos do valor de F(**X**), do ponto de vista de engenharia, a solução do GEO_{var} para CFQ é bastante diferente da solução obtida pelo GEO. A primeira necessita de apenas 49,7 W para a potência do aquecedor (X_6), contra 57,3 W da segunda. Relembrando que X_6 é o terceiro termo de F(**X**) e, como ambas as soluções tem quase o mesmo F(**X**), a contrapartida é que a solução com 49,7 W tem pior desempenho nos dois primeiros termos (termos de temperatura) de F(**X**), a fim de restabelecer a igualdade. Talvez o ponto mais importante seja a possibilidade de escolha entre estas duas soluções. O projetista pode decidir qual delas é melhor considerando outros quantificadores, ausentes na formulação de F(**X**), simplesmente porque não ocorreram como possibilidades ou mesmo porque eram muito difíceis de quantificar. Exemplos de tais quantificadores são facilidade de montagem, durabilidade, e robustez, apenas para mencionar

alguns. A possibilidade de escolha somente existirá se houver conhecimento da existência de duas ou mais soluções equivalentes.

7.2.5 - Aplicações correlatas

Os resultados obtidos com a PMM e descritos até aqui foram apresentados em Galski et al. (2004a; 2007). Além disso, outros estudos envolvendo o projeto otimizado da PMM foram efetuados. Em Sousa et al. (2005a), foi feito um estudo muito parecido com o aqui exposto, mas para uma condição crítica correspondente ao Caso Frio (CF) diferente. Em Muraoka et al. (2006), uma abordagem para o projeto térmico detalhado de satélites foi apresentada. A abordagem consiste de duas etapas. Na primeira etapa, um modelo térmico simplificado é desenvolvido e integrado a um algoritmo de otimização, com a conseqüente obtenção de soluções otimizadas. Na segunda etapa, as soluções obtidas na primeira etapa são aplicadas ao modelo térmico detalhado do satélite, com a geração de novos e mais precisos perfis de temperatura. Havendo necessidade, devido a eventuais violações de limites das temperaturas, o projetista se vale de sua experiência e realiza pequenas correções nas soluções obtidas na primeira etapa, a fim de evitar a ocorrência das violações. No estudo citado, a aplicação usada como exemplo foi a mesma descrita neste Capítulo, ou seja, o projeto térmico da PMM. O modelo térmico simplificado utilizado foi incrementado com a inclusão dos painéis solares e com a utilização de aquecedor adicional para os propulsores. Os resultados obtidos foram amplamente satisfatórios, sendo que duas missões foram analisadas e em apenas uma houve a necessidade de pequena modificação na solução obtida na primeira etapa. O resultado obtido é importante, pois valida a abordagem de otimização apresentada neste Capítulo como método viável de obtenção de soluções preliminares otimizadas.

7.2.6 - Abordagem multiobjetivo

Até aqui todos os resultados obtidos no projeto térmico da PMM, que foram apresentados ou comentados nas seções precedentes, utilizaram uma abordagem monoobjetivo. Entretanto, o antagonismo que existe entre a otimização da distribuição de temperatura dos painéis e a otimização da potência gasta com aquecedores pode ser tratado por meio de uma abordagem multiobjetivo. Nesse caso, tanto a distribuição de temperaturas quanto a potência dos aquecedores tornam-se funções objetivo separadas, gerando um problema biobjetivo. A solução desse problema biobjetivo gera, então, não uma, mas um conjunto de soluções otimizadas onde as

melhores relações de compromisso possíveis entre os dois objetivos aparecem naturalmente nas soluções obtidas, formando a fronteira de Pareto. Dessa forma, a solução do problema biobjetivo é mais completa, pois permite ao projetista escolher uma solução baseado, por exemplo, na potência dos aquecedores, mas sabendo exatamente qual é a contrapartida em termos da distribuição de temperatura nos painéis.

Nas próximas seções, é feita a formulação do problema biobjetivo para o projeto térmico da PMM e a apresentação dos resultados. Na aplicação desta abordagem, utilizou-se o mesmo modelo térmico simplificado utilizado em Muraoka et al. (2006), o qual, além dos seis painéis laterais (nós 1 a 6), modela também os painéis solares (nós 8 e 9, vide Figura 7.9) e os elementos da propulsão (nó 7). Além disso, uma das missões analisadas no trabalho citado foi escolhida para geração da respectiva fronteira de Pareto com o algoritmo M-GEO.

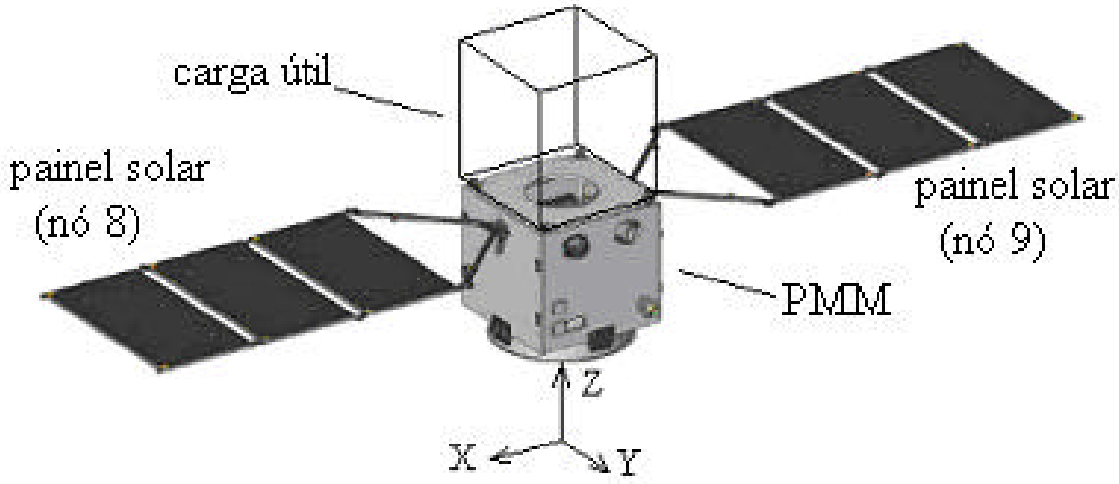


Figura 7.9 – Vista da PMM incluindo os painéis solares (nós 8 e 9)

7.2.7 - Formulação do problema biobjetivo e resultados obtidos

Matematicamente, o problema é posto como:

Minimizar:

$$\mathbf{F(X)} = \begin{bmatrix} F_1(\mathbf{X}) = 0{,}5 \cdot \left\| \mathbf{T_{CF}}(\mathbf{X}) - \mathbf{T_A} \right\|_2 + 0{,}25 \cdot \left\| \mathbf{T_{CQ1}}(\mathbf{X}) - \mathbf{T_A} \right\|_2 + 0{,}25 \cdot \left\| \mathbf{T_{CQ2}}(\mathbf{X}) - \mathbf{T_A} \right\|_2 \\ F_2(\mathbf{X}) = X_6 + X_7 \end{bmatrix} \tag{7.5}$$

Sujeito a: $\mathbf{X_{MIN}} \leq \mathbf{X} \leq \mathbf{X_{MAX}}$

$\mathbf{T_{MIN}} \leq \mathbf{T_{CF}(X)} \leq \mathbf{T_{MAX}}$

$$\mathbf{T_{MIN}} \leq \mathbf{T_{CQ1}(X)} \leq \mathbf{T_{MAX}}$$

$$\mathbf{T_{MIN}} \leq \mathbf{T_{CQ2}(X)} \leq \mathbf{T_{MAX}}$$

onde:

$\mathbf{X}_{1:5}$ = área do radiador nos painéis 1 a 5, respectivamente; (m^2)

$\mathbf{X}_6$ = potência do aquecedor das baterias durante o CF; (W)

$\mathbf{X}_7$ = potência do aquecedor dos componentes da propulsão durante o CF; (W)

$\mathbf{T_{CF}(X)}_{1:7}$ = temperaturas do CF nos nós 1 a 7, respectivamente; (°C)

$\mathbf{T_{CQ1}(X)}_{1:7}$ = temperaturas do CQ1 nos nós 1 a 7, respectivamente; (°C)

$\mathbf{T_{CQ2}(X)}_{1:7}$ = temperaturas do CQ2 nos nós 1 a 7, respectivamente; (°C)

$\mathbf{T_A}_{1:7}$ = temperaturas alvo nos nós 1 a 7, respectivamente; (°C)

$\mathbf{X_{MIN}}_{1:5}$ = áreas mínimas dos radiadores 1 a 5, respectivamente; (m^2)

$\mathbf{X_{MIN}}_6$ = potência mínima do aquecedor das baterias; (W)

$\mathbf{X_{MIN}}_7$ = potência mínima do aquecedor dos componentes da propulsão; (W)

$\mathbf{X_{MAX}}_{1:5}$ = áreas máximas dos radiadores 1 a 5, respectivamente; (m^2)

$\mathbf{X_{MAX}}_6$ = potência máxima do aquecedor das baterias; (W)

$\mathbf{X_{MAX}}_7$ = potência máxima do aquecedor dos componentes da propulsão; (W)

$\mathbf{T_{MIN}}_{1:7}$ = temperaturas mínimas admissíveis nos nós 1 a 7, respectivamente; (°C)

$\mathbf{T_{MAX}}_{1:7}$ = temperaturas máximas admissíveis nos nós 1 a 7, respectivamente; (°C)

Na equação 7.5, $F_1(\mathbf{X})$ é o somatório dos resíduos quadráticos das temperaturas obtidas nos nós 1 a 7 do modelo térmico simplificado utilizado. Os nós 1 a 6 representam os painéis 1 a 6 da PMM e o nó 7 representa os elementos da propulsão. Na missão analisada em Muraoka et al. (2006) e ora utilizada, três condições críticas foram antevistas, um Caso Frio (CF) e dois Casos Quentes (CQ1 e CQ2). A $F_1(\mathbf{X})$ foi formulada de modo a manter inalterada a importância relativa entre o CF e os dois CQs, ou seja, o CF vale tanto quanto os dois CQs. Os fatores multiplicativos 0,5 e 0,25 que aparecem na formulação da F_1 existem para garantir tal efeito. Já a $F_2(\mathbf{X})$ representa a potência total gasta pelos aquecedores (das baterias e da propulsão) durante o CF. Agora, $\mathbf{F}(\mathbf{X})$ é uma função vetorial com dois elementos, $\mathbf{F}(\mathbf{X}) = [F_1(\mathbf{X}), F_2(\mathbf{X})]^T$ e o objetivo é minimizar $F_1(\mathbf{X})$ e $F_2(\mathbf{X})$, obtendo a fronteira de Pareto e as respectivas soluções de projeto. As sete variáveis de

projeto são representadas pelo vetor **X** no processo de otimização. Os vetores $\mathbf{X_{MIN}}$ e $\mathbf{X_{MAX}}$ contêm as restrições laterais para cada uma das variáveis, estabelecendo seus limites admissíveis. Além disso, restrições adicionais são impostas as temperaturas dos painéis, representadas pelos limites mínimos e máximos $\mathbf{T_{MIN}}$ e $\mathbf{T_{MAX}}$ indicados na equação 7.5.

O modelo térmico simplificado utilizado considera os seis painéis laterais, os dois painéis solares e o conjunto de componentes da propulsão como nós da representação de parâmetros concentrados (Gilmore, 1994), totalizando nove nós. Considerando uma situação de regime permanente com cargas térmicas orbitais médias, o balanço de calor em cada um dos nós leva ao conjunto de equações:

$$\sum_{j=1}^{7} \mathbf{G}_{1,j}\left(\mathbf{T}_1 - \mathbf{T}_j\right) + \sum_{j=1}^{6} \mathbf{R}_{1,j}\left(\mathbf{T}_1^4 - \mathbf{T}_j^4\right) + e_{rad}\mathbf{X}_1\left(\mathbf{T}_1^4 - \mathbf{T}_\infty^4\right) = \mathbf{q}_{i1} + a_{abs}\mathbf{X}_1\,\mathbf{q}_{s1} + e_{abs}\mathbf{X}_1\,\mathbf{q}_{t1} \tag{7.6}$$

$$\sum_{j=1}^{6} \mathbf{G}_{2,j}\left(\mathbf{T}_2 - \mathbf{T}_j\right) + \sum_{j=1}^{6} \mathbf{R}_{2,j}\left(\mathbf{T}_2^4 - \mathbf{T}_j^4\right) + e_{rad}\mathbf{X}_2\left(\mathbf{T}_2^4 - \mathbf{T}_\infty^4\right) = \mathbf{q}_{i2} + a_{rad}\mathbf{X}_2\,\mathbf{q}_{s2} + e_{rad}\mathbf{X}_2\,\mathbf{q}_{t2} + \mathbf{X}_6 \tag{7.7}$$

$$\sum_{j=1}^{6} \mathbf{G}_{3,j}\left(\mathbf{T}_3 - \mathbf{T}_j\right) + \sum_{j=1}^{6} \mathbf{R}_{3,j}\left(\mathbf{T}_3^4 - \mathbf{T}_j^4\right) + e_{rad}F_{3,\infty}\mathbf{X}_3\left(\mathbf{T}_3^4 - \mathbf{T}_\infty^4\right) + e_{rad}\mathbf{X}_3\,F_{3,8}\left(\mathbf{T}_3^4 - \mathbf{T}_8^4\right) = \mathbf{q}_{i3} + a_{rad}\mathbf{X}_3\,\mathbf{q}_{s3} + e_{rad}\mathbf{X}_3\,\mathbf{q}_{t3} \tag{7.8}$$

$$\sum_{j=1}^{6} \mathbf{G}_{4,j}\left(\mathbf{T}_4 - \mathbf{T}_j\right) + \sum_{j=1}^{6} \mathbf{R}_{4,j}\left(\mathbf{T}_4^4 - \mathbf{T}_j^4\right) + e_{rad}\mathbf{X}_4\left(\mathbf{T}_4^4 - \mathbf{T}_\infty^4\right) = \mathbf{q}_{i4} + a_{rad}\mathbf{X}_4\,\mathbf{q}_{s4} + e_{rad}\mathbf{X}_4\,\mathbf{q}_{t4} \tag{7.9}$$

$$\sum_{j=1}^{6} \mathbf{G}_{5,j}\left(\mathbf{T}_5 - \mathbf{T}_j\right) + \sum_{j=1}^{6} \mathbf{R}_{5,j}\left(\mathbf{T}_5^4 - \mathbf{T}_j^4\right) + e_{rad}F_{5,\infty}\mathbf{X}_5\left(\mathbf{T}_5^4 - \mathbf{T}_\infty^4\right) + e_{rad}\mathbf{X}_5\,F_{5,8}\left(\mathbf{T}_5^4 - \mathbf{T}_9^4\right) = \mathbf{q}_{i5} + a_{rad}\mathbf{X}_5\,\mathbf{q}_{s5} + e_{rad}\mathbf{X}_5\,\mathbf{q}_{t5} \tag{7.10}$$

$$\sum_{j=1}^{6} \mathbf{G}_{6,j}\left(\mathbf{T}_6 - \mathbf{T}_j\right) + \sum_{j=1}^{6} \mathbf{R}_{6,j}\left(\mathbf{T}_6^4 - \mathbf{T}_j^4\right) = 0 \tag{7.11}$$

$$\mathbf{G}_{7,1}\left(\mathbf{T}_7 - \mathbf{T}_1\right) = \mathbf{X}_7 \tag{7.12}$$

$$e_{ps}F_{8,3}\mathbf{X}_8\left(\mathbf{T}_8^4 - \mathbf{T}_3^4\right) + e_{ps}F_{8,\infty}\mathbf{X}_8\left(\mathbf{T}_8^4 - \mathbf{T}_\infty^4\right) = a_{ps}\mathbf{X}_8\,\mathbf{q}_{s8} + e_{ps}\mathbf{X}_8\,\mathbf{q}_{t8} \tag{7.13}$$

$$e_{ps}F_{9,5}\mathbf{X}_9\left(\mathbf{T}_9^4 - \mathbf{T}_5^4\right) + e_{ps}F_{9,\infty}\mathbf{X}_9\left(\mathbf{T}_9^4 - \mathbf{T}_\infty^4\right) = a_{ps}\mathbf{X}_9\,\mathbf{q}_{s9} + e_{ps}\mathbf{X}_9\,\mathbf{q}_{t9} \tag{7.14}$$

onde:

$\mathbf{X}_{1:5}$ = área do radiador nos painéis 1 a 5, respectivamente; (m^2)

$\mathbf{X}_6$ = potência do aquecedor das baterias durante o CF; (W)

$\mathbf{X}_7$ = potência do aquecedor dos componentes da propulsão durante o CF; (W)

$\mathbf{T}_{1:6}$ = temperaturas nos painéis laterais 1 a 6, respectivamente; (K)

$\mathbf{T}_7$ = temperatura nos elementos da propulsão (tanque, tubo, válvula); (K)

$\mathbf{T}_{8:9}$ = temperaturas nos painéis solares; (K)

$\mathbf{q_i}_{1:5}$ = geração interna de calor nos painéis 1 a 5, respectivamente; (W)

$\mathbf{q_s}_{1:5;8;9}$ = fluxo de radiação solar nos painéis 1 a 5, 8, e 9, respectivamente; (W/m^2)

$\mathbf{q_t}_{1:5;8;9}$ = fluxo de radiação terrestre nos painéis 1 a 5, 8, e 9, respectivamente; (W/m^2)

$\mathbf{R}_{i,j}$ = condutância radiativa entre o painel i e o painel j; (W/K^4)

$\mathbf{G}_{i,j}$ = condutância condutiva entre o painel i e o painel j; (W/K)

α_{abs} = absortividade solar do absorvedor (painel 1);

e_{abs} = emissividade no infravermelho do absorvedor (painel 1);

α_{rad} = absortividade solar dos radiadores (painéis 2 a 5);

e_{rad} = emissividade no infravermelho dos radiadores (painéis 2 a 5);

α_{ps} = absortividade solar dos painéis solares;

e_{ps} = emissividade no infravermelho dos painéis solares;

$\mathbf{T}_\infty$ = temperatura de um corpo negro no espaço distante; (K)

As condutâncias radiativas $\mathbf{R}_{i,j}$, para i=1 até 6 e j=1 até 6 (painéis laterais da PMM), devidas à troca de calor por radiação entre as superfícies internas dos painéis foram determinadas usando o módulo pré-processador do software PCTER (vide Figura 7.3), considerando uma emissividade de e=0,85 (tinta preta) para os nós 1,3,4,5,6 e e=0,05 (manta *MLI*) para o nó 2. As condutâncias condutivas $\mathbf{G}_{i,j}$ (para i=1 a 7 e j=1 a 7) foram determinadas a partir das características das juntas aparafusadas entre os painéis. Os elementos da propulsão (nó 7) estão montados sobre o painel 1 (-Z) e isolados termicamente do mesmo. Para simplificar o modelo, a troca de calor radiativa entre os nós 3 e 8 não considera multireflexão. Além disso, os fatores de forma ($F_{3,8}$ and $F_{3,\infty}$) são calculados para a área total do nó 3. Os mesmos critérios foram adotados para a troca de calor entre os nós 5 e 9.

De forma similar ao problema monoobjetivo apresentado antes, para cada caso crítico as radiações médias externas ($\mathbf{q_s}$ e $\mathbf{q_t}$) incidentes em cada painel são

calculadas pelo módulo CATER (vide Figura 7.3), considerando os valores já informados dos parâmetros constante solar, fator de albedo e radiação terrestre. Menciona-se que CQ1 e CQ2 utilizam um mesmo conjunto de valores destes parâmetros. Os valores de absortividade e emissividade dos radiadores (vide equações 7.6 a 7.14) foram fixados em a_{rad}=0,16 para CF e a_{rad}=0,21 para CQ1, CQ2 e e_{rad}=0,87 para todos os três casos. Para o absorvedor do painel 1, os valores utilizados foram a_{abs}=0,15 e e_{abs}=0,05 para todos os três casos. Para os painéis solares, os valores utilizados foram a_{ps}=0,76 e e_{ps}=0,88 para todos os três casos.

Os três casos críticos identificados e utilizados na otimização multiobjetivo estão ilustrados na Figura 7.10. O CF ocorre com o Sol no plano orbital do satélite (ângulo beta=0°). Os casos CQ1 e CQ2 ocorrem para ângulos beta iguais a +38° e -38°, respectivamente.

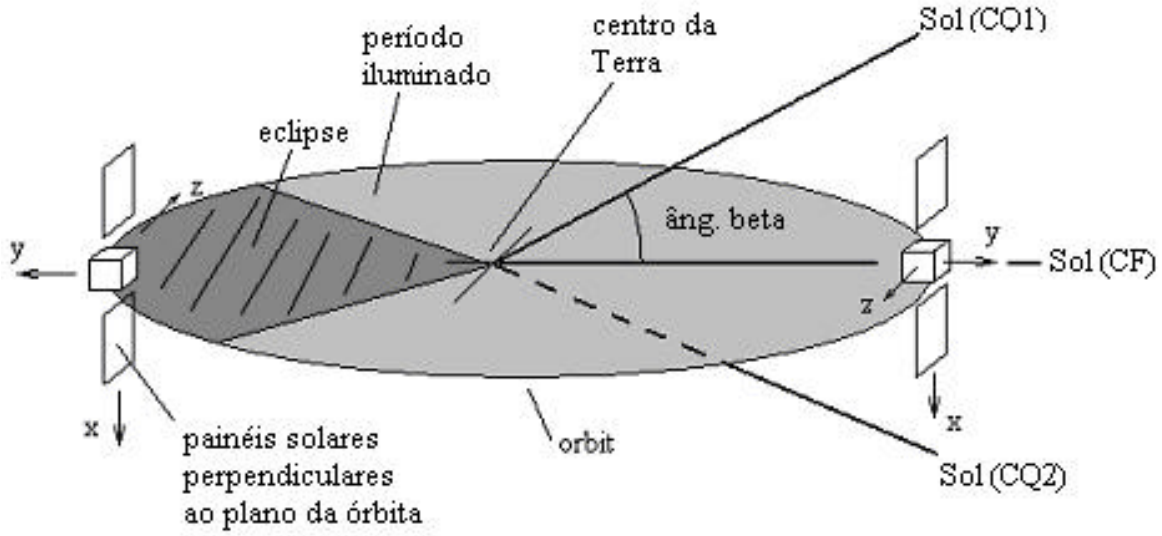


Figura 7.10 – Órbita e atitude da missão analisada (CF, CQ1 e CQ2)

Em adição às cargas externas, a dissipação interna de calor ($\mathbf{q_i}$) que ocorre nos painéis devido aos equipamentos montados nos mesmos é considerada. A Tabela 7.6 sumariza as dissipações de calor internas devidas aos dispositivos eletrônicos, bem como os limites operacionais de temperatura e a temperatura alvo desejada para cada painel ou componente. Logo em seguida, a Tabela 7.7 descreve e define os limites das variáveis de projeto.

Tabela 7.6 - Limites operacionais de temperatura, temperaturas alvo e dissipação de calor nos painéis (modelo com painéis solares e elementos da propulsão).

Parâmetro		Painel ou Componente [1]								
		1	2	3	4	5	6 [2]	7	8	9
Temperaturas limites (^oC) [3]	T_{min}	-13,0	-7,0	-8,0	-16,0	-6,0	-15,0	15,0	-80,0	-80,0
	T_{max}	+57,0	+13,0	+32,0	+37,0	+37,0	+45,0	+45,0	+80,0	+80,0
Temperatura alvo (^oC)	T_A	+22,0	+3,0	+12,0	+11,0	+16,0	+15,0	+30,0	-	-
Dissipação interna de calor (W) [4]	CF	8,7	8, 0	58,7	26,0	46,0	0,0	0,0	0,0	0,0
	CQ	11,4	12,0	73,7	50,8	78,0	0,0	0,0	0,0	0,0

[1] 1 a 6 = painéis laterais da PMM. 8 e 9 = painéis solares. 7 = componentes da propulsão. [2]Painel superior da PMM, termicamente isolado da carga útil. [3]Os limites de temperatura de cada painel foram determinados de acordo com o equipamento mais restritivo montado no mesmo. [4]A dissipação de calor interna de cada painel é o somatório das dissipações de todos os equipamentos montados no respectivo painel.

Tabela 7.7. Variáveis de projeto para otimização.

i	**Descrição Variável X_i**	**Intervalo Permitido $X_{MIN} \sim X_{MAX}$**
1	Área de absorvedor no painel 1	0,0 ~ 0,1940 m^2
2	Área de radiador no painel 2	0,0 ~ 0,8054 m^2
3	Área de radiador no painel 3	0,0 ~ 0,8054 m^2
4	Área de radiador no painel 4	0,0 ~ 0,8054 m^2
5	Área de radiador no painel 5	0,0 ~ 0,8054 m^2
6	Potência do aquecedor das baterias (painel 2)	0 ~ 15 W
7	Potência do aquecedor da propulsão (painel 1)	0 ~ 15 W

Cada variável de projeto foi codificada em 4 bits, o que significa uma resolução de 0,0129 m^2 para a área de absorvedor (painel 1) e 0,0537 m^2 para as áreas dos radiadores (painéis 2 a 5) e ainda 1,0 W para a potência dos aquecedores, respectivamente. Em Muraoka et al. (2006) um estudo comparativo preliminar foi efetuado usando 7 e 4 bits. Os valores otimizados da função objetivo lá utilizada não diferiram muito nas duas codificações, o que motivou a adoção de 4 bits por variável de projeto e a conseqüente vantagem da redução do espaço de busca.

Inicialmente, uma busca preliminar foi efetuada pelo M-GEO para $\tau \in \{1;2;3\}$ e critério de parada de $2x10^5$ avaliações de $\mathbf{F}(\mathbf{X})$ (vide equação 7.5), distribuídas em 50 execuções independentes. O objetivo foi identificar qual dos três valores de τ apresenta melhor desempenho na obtenção da fronteira de Pareto. A Figura 7.11 apresenta as fronteiras de Pareto resultantes.

Como pode ser visto, τ=2 obteve o melhor desempenho, tendo sido escolhido para proceder a busca principal.

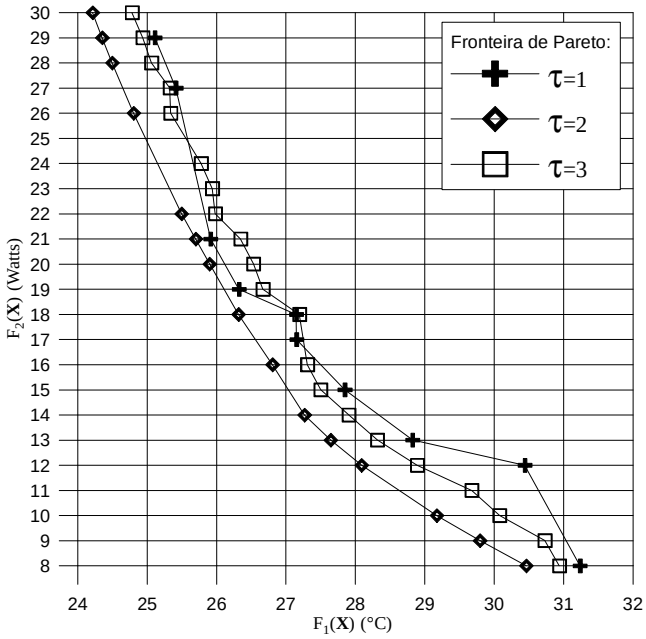


Figura 7.11 – Fronteira de Pareto x τ com $2 \cdot 10^5$ avaliações de **F**(**X**)

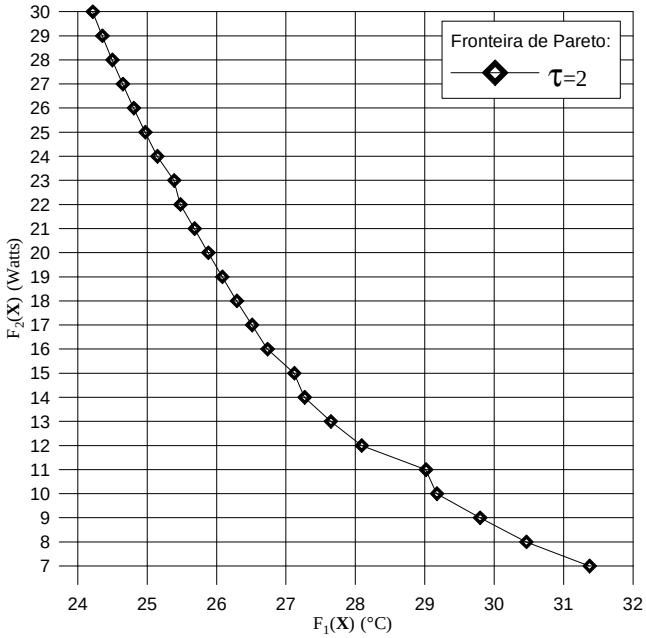


Figura 7.12 – Fronteira de Pareto para τ=2 e $5 \cdot 10^6$ avaliações de **F**(**X**)

A busca principal foi efetuada pelo M-GEO com τ=2 e critério de parada de 5×10^6 avaliações de **F**(**X**), distribuídas em 50 execuções independentes. A Figura 7.12 apresenta graficamente a fronteira de Pareto resultante e a Tabela 7.8 as respectivas soluções.

Tabela 7.8 - Soluções de projeto dos pontos da fronteira de Pareto.

Conjunto de soluções de Pareto							Fronteira de Pareto	
$\mathbf{X}_1$ (m^2)	$\mathbf{X}_2$ (m^2)	$\mathbf{X}_3$ (m^2)	$\mathbf{X}_4$ (m^2)	$\mathbf{X}_5$ (m^2)	$\mathbf{X}_6$ (W)	$\mathbf{X}_7$ (W)	$\mathbf{F}_1(\mathbf{X})$ (°C)	$\mathbf{F}_2(\mathbf{X})$ (W)
0,1940	0,2685	0,2685	0,3759	0,1075	0	7	31,38	7
		0,2148		0,1612		8	30,46	8
						9	29,80	9
						10	29,18	10
	0,2148			0,2148		11	28,98	11
	0,2685			0,1612		12	28,09	12
						13	27,65	13
						14	27,27	14
	0,2148			0,2148		**15**	27,10	15
	0,2685		0,3222		1		26,74	16
					2		26,51	17
					3		26,29	18
					4		26,08	19
					5		25,88	20
					6		25,69	21
		0,2685			7		25,48	22
	0,3222			0,1612	8		25,39	23
	0,2685	0,2148	0,3759	0,2148	9		25,15	24
					10		24,97	25
					11		24,81	26
					12		24,65	27
					13		24,50	28
					14		24,35	29
					15		24,22	30

Nota: Valores em negrito indicam limite superior atingido para a respectiva variável.

A fronteira de Pareto apresentada na Figura 7.12 encerra em si mesma as melhores possibilidades existentes no jogo de perde-ganha que ocorre entre a distribuição de temperaturas, representada pela $\mathbf{F}_1(\mathbf{X})$, e a potência necessária aos aquecedores, representada pela $\mathbf{F}_2(\mathbf{X})$. Agora o projetista sabe, por exemplo, que passando de 7 para 8 watts a potência total dos aquecedores ocasiona uma melhora de aproximadamente 1°C na distribuição de temperaturas. Sabe também, que passando de 29 para 30 watts ocasiona

uma melhora de menos de 0,2°C. Ou seja, os benefícios advindos do acréscimo de 1 watt na potência são fortemente influenciados pelo ponto ou região da fronteira de Pareto em que se está. É exatamente esse tipo de conhecimento, propiciado pela fronteira de Pareto, que torna a abordagem multiobjetivo tão interessante e importante.

Olhando-se agora a Tabela 7.8, vê-se que, como esperado, aumentando-se a potência dos aquecedores ocorre melhora na distribuição de temperatura (quanto menor o valor de $\mathbf{F}_1(\mathbf{X})$, mais próximas das temperaturas alvo foram as temperaturas ocorridas nos painéis). Adicionalmente, outras duas observações podem ser feitas. A primeira é que se o projetista estiver interessado em melhorar ainda mais a distribuição de temperaturas e considerar as potências dos aquecedores como opção, deve aumentar preferencialmente a potência do aquecedor dos elementos da propulsão ($\mathbf{X}_7$), já que, como visto pela tabela, este aquecedor foi o primeiro a atingir seu valor máximo de 15 W. Na Tabela 7.8, os valores em negrito indicam que a respectiva variável de projeto atingiu seu limite superior. A segunda observação diz respeito à área de absorvedor colocado no painel 1 ($\mathbf{X}_1$). Como visto, todas as soluções de Pareto ocorrem quando a área desse absorvedor atinge seu valor máximo permitido (0,1940 m^2). Então, é possível inferir que, possivelmente, áreas maiores para o absorvedor do painel 1 levem a distribuições de temperatura ainda melhores e isto sem aumentar os gastos com os aquecedores ou talvez até, diminuindo-os.

O aspecto mais importante das observações feitas no parágrafo anterior é o de que a análise dos resultados obtidos em termos das soluções que formam a fronteira de Pareto dá ao projetista do satélite dicas de como modificar o projeto de forma a melhorá-lo ainda mais. Mesmo considerando não ser este o caso geral, mas sim o particular, ainda assim, só o fato de saber que ele pode ocorrer torna a abordagem multiobjetivo ainda mais valiosa.

7.3 – Projeto da configuração de uma constelação de satélites de sensoriamento remoto do tipo MAPSAR

A aplicação reportada refere-se ao projeto ótimo de uma pequena constelação de satélites MAPSAR (*Multi-Application Purpose Synthetic Aperture Radar*). O programa do satélite MAPSAR é um projeto conjunto Brasil-Alemanha destinado à mensuração, gerenciamento e monitoração de recursos naturais terrestres (Schröder et al., 2005). O

procedimento de projeto é atacado como um problema de otimização multiobjetivo, considerando dois parâmetros principais de desempenho, o tempo de revisita médio e o tempo de revisita máximo sobre uma região de interesse. O tempo de revisita é uma métrica tradicionalmente usada para avaliar constelações de cobertura esparsa, ou seja, constelações que não conseguem cobertura contínua de uma área na superfície terrestre. Estudos de compromisso são efetuados para diversos números de satélites na constelação, a fim de acessar a melhor razão entre desempenho e custo.

7.3.1 – O satélite MAPSAR

O satélite MAPSAR (*Multi-Application Purpose Synthetic Aperture Radar*) (Schröder et al., 2005) tem um Radar de Abertura Sintética (em inglês, *Synthetic Aperture Radar* - SAR) como carga útil. A antena do radar tem um refletor principal elíptico-parabólico com 7,5m de comprimento em azimute e 5m em elevação. As resoluções previstas são 3m, 10m e 20m. Ele deve ser colocado em uma órbita heliosíncrona com altitude entre 600km e 650km. A informação mais recente indica um ciclo de sete dias com 104 órbitas (altitude de 606km). Após sete dias, o ciclo se repete, com a órbita 105 sendo igual à órbita 1, relativamente ao traço orbital resultante sobre a superfície terrestre. Os traços orbitais são caracterizados por terem latitudes fixas e distâncias constantes entre traços no plano do equador. A distância entre traços orbitais adjacentes (distância inter-traço) é 385,3km para o ciclo de sete dias e duas órbitas sucessivas produzem traços orbitais distantes 2697,4km. O resultado é uma grade orbital "fechada" (relativamente à superfície terrestre) com 104 órbitas/traços orbitais. A Figura 7.13 apresenta três vistas do satélite MAPSAR e a Figura 7.14 mostra a geometria de cobertura do seu sensor SAR embarcado. A aquisição da imagem ocorre num plano, aqui chamado de plano de aquisição, o qual é perpendicular ao traço orbital. O sensor é "cego" em uma região definida por ângulos de incidência (vide Figura 7.14) entre 0^o e 20^o, em ambos os lados do traço orbital Nadir do satélite. Esta região, chamada Zona Nadir, é indicada por uma linha pontilhada na Figura 7.14. A totalidade da região de aquisição (linhas grossas na cor verde) é definida por ângulos de incidência entre 20^o e $48{,}1^o$, em ambos os lados do traço Nadir. Deve ser notado que o projeto da antena do sensor SAR alcança instantaneamente (*swath*) somente uma extensão de até 55km aproximadamente, dentro da região de aquisição de

395km mostrada na Figura 7.14. É importante mencionar também que o sensor SAR permite a obtenção de imagens de alta resolução independentemente das condições do tempo ou de luz solar, o que é particularmente importante para regiões que têm ocorrência de chuva, nuvens, névoa e fumaça, como, por exemplo, a região amazônica.

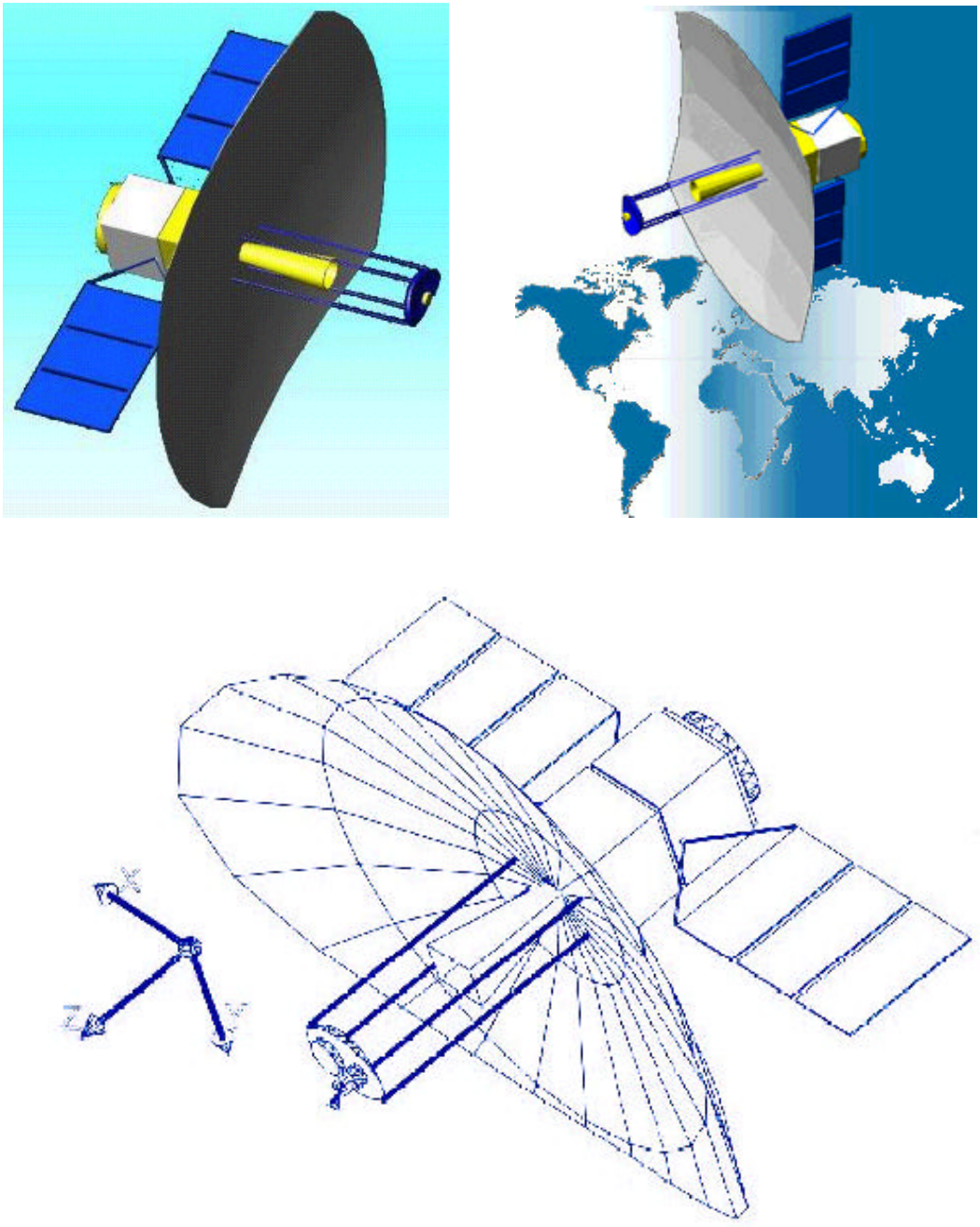


Figura 7.13 – Três vistas em perspectiva do MAPSAR
Fonte: INPE e Schröder et al. (2005).

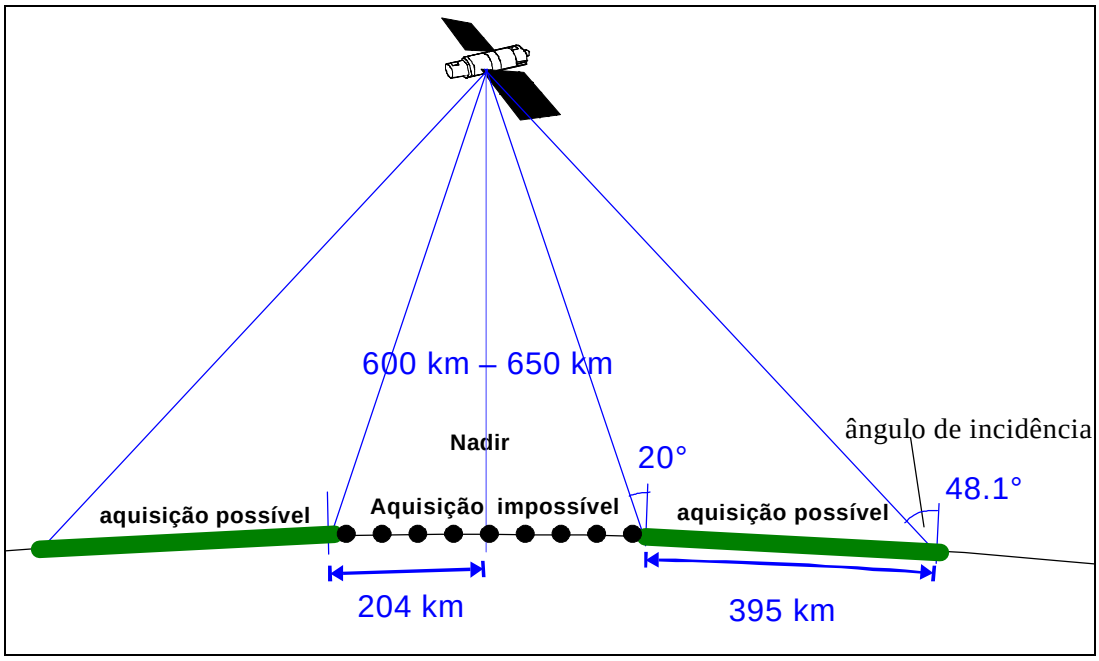


Figura 7.14 - Alcance (20^o a 48.1^o) do sensor SAR
Fonte: adaptado de Schröder et al. (2005).

7.3.2 – Constelação MAPSAR

Em Abril de 2002, um grupo de trabalho de usuários potênciais e especialistas em SAR reuniu-se no INPE, em São José dos Campos, e foi fortemente recomendado que uma série de satélites MAPSAR fosse construída, visando aspectos operacionais e a continuidade do programa. Além disso, especificamente no que se refere ao monitoramento de desastres, um tempo de revisita menor do que um dia foi colocado como requisito desejável (Schröder et al., 2005). Como já mencionado na seção anterior, a informação mais recente aponta para uma periodicidade (tempo de revisita) de 7 dias, a qual não atende o requisito dos usuários para monitoramento de desastres. Deste modo, o estudo de uma constelação de satélites MAPSAR pode vir a ser relevante num futuro próximo. Considerando a importância do monitoramento de desastres sobre uma região escassamente povoada, como é o caso da região amazônica brasileira, este estudo tenta responder qual é a melhor configuração de uma constelação de satélites MAPSAR, a fim de obter o menor tempo de revisita possível, enquanto mantém um dos satélites da constelação na órbita heliosíncrona original do MAPSAR, para manter a capacidade de cobertura global.

É muito comum escolher constelação de cobertura esparsa a partir das assim chamadas Constelações de Walker. Infelizmente, esta abordagem não permite o estabelecimento de relações de compromisso entre o tempo de revisita médio e o tempo de revisita máximo, para fins de projeto da constelação. Além disso, constelações de Walker impõem que somente constelações simétricas são consideradas na busca e tem sido mostrado que isto freqüentemente impede chegar-se ao projeto ótimo da constelação (Crossley e Williams, 2000; George, 1997). No presente estudo, constelações de Walker não foram consideradas como uma informação *a priori* (ou imposição) ao processo de otimização do projeto da constelação.

Recapitulando e complementando a informação dada na seção 7.3.1, o estudo referente ao projeto da constelação é levado a cabo considerando 2, 3 e 4 satélites. Em todos os três casos, ela é composta pelo satélite MAPSAR original, com sua órbita heliosíncrona e 1, 2 ou 3 satélites adicionais do tipo MAPSAR. Os parâmetros orbitais do primeiro satélite são fixos e servem como referência para configurar o resto da constelação. Os elementos keplerianos usados para o satélite MAPSAR original (ou de referência) são: semi-eixo maior (6989,483km); excentricidade (0,0011); inclinação ($97,7722^o$); ascensão reta do nodo ascendente ($230,962^o$); argumento do perigeu (90^o) e anomalia média (90^o).

As variáveis de projeto para os demais satélites são: inclinação; ascensão reta do nodo ascendente e anomalia média. Os outros parâmetros orbitais são constantes e iguais àqueles do satélite de referência: semi-eixo maior (6989,483km); excentricidade (0,0011) e argumento do perigeu (90^o).

A busca pelo projeto ótimo da constelação é feita para os números de satélites previamente mencionados. A simulação da órbita dos satélites, com a propagação do vetor de estado, é feita por um programa de computador desenvolvido especificamente para a propagação de órbita de vários satélites. Ele se baseia nas rotinas de dinâmica orbital (em Fortran) desenvolvidas no INPE pelo grupo de dinâmica orbital para a propagação da órbita de apenas um satélite (Kuga e Orlando, 1988a; 1988b; Kuga et al., 1988). De forma similar ao que foi feito em Williams et al. (2001), a propagação é feita considerando apenas o efeito do campo gravitacional terrestre, modelado por um campo de força central (ponto de massa) mais o segundo harmônico zonal, J_2, o qual modela o achatamento dos pólos.

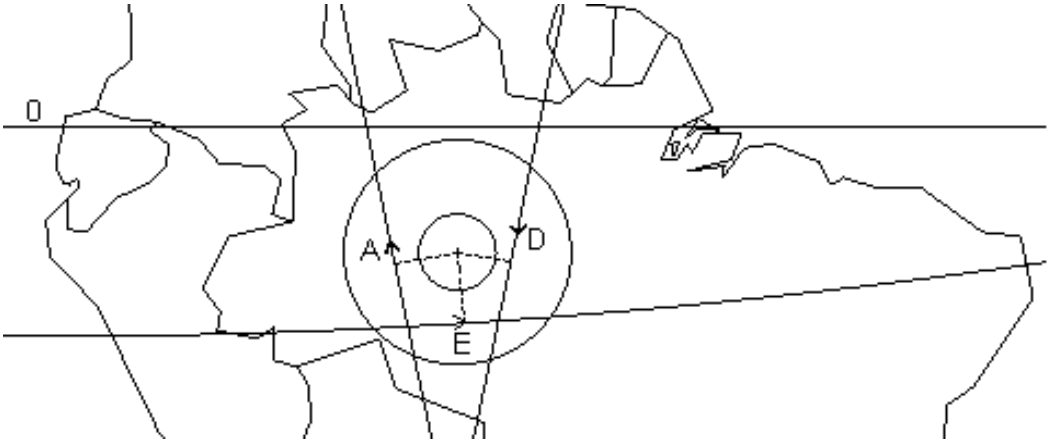


Figura 7.15 – Círculo de visibilidade do ponto alvo (Amazônia) e três exemplos de traços orbitais.

A região de interesse é definida pelo ponto com coordenadas: longitude (-62^o) e latitude (-6^o). Ele está localizado na porção central da Amazônia brasileira. A Figura 7.15 mostra-o no centro de dois círculos concêntricos. As coordenadas definem um ponto alvo sobre (ou para) o qual os tempos de revisita médio e máximo são calculados e usados como valores das duas funções objetivo para o projeto ótimo com M-GEO. Este cálculo deve levar em conta as restrições de aquisição de imagens impostas pelas limitações do sensor SAR.

As passagens visíveis de um satélite sobre uma estação de rastreio caracterizam-se por um conjunto de ângulos de azimute e elevação, usados para apontar a antena da estação na direção do satélite, enquanto o satélite passa. O ângulo de elevação é aquele formado entre o plano horizontal local e a linha conectando a estação de rastreio e o satélite. Uma passagem visível ocorre quando o ângulo de elevação é maior do que zero. Como mencionado na seção 7.3.1, a região de acesso do sensor do MAPSAR é definida por ângulos de incidência entre 20^o e $48{,}1^o$. Para um ponto dado, o ângulo de incidência e o ângulo de elevação são complementares dentro do quadrante, isto é, o ponto com ângulo de incidência de 20^o "vê" o satélite com um ângulo de elevação de 90^o-20^o=70^o, enquanto o ponto com ângulo de incidência de $48{,}1^o$ faz o mesmo com um ângulo de elevação de 90^o-$48{,}1^o$=$41{,}9^o$. Então, para um ponto sobre a superfície terrestre o qual está dentro do plano de aquisição de imagem do MAPSAR, se o ângulo de elevação do satélite MAPSAR para aquele ponto é maior do que $41{,}9^o$ e menor do que 70^o, o ponto está dentro da região de aquisição de imagem do sensor do MAPSAR. Em termos de traços orbitais, isto é equivalente aos dois círculos de visibilidade concêntricos mostrados na Figura 7.15, um com raio de 204km e outro com 599km. Quando os traços orbitais do satélite passam dentro do círculo externo e não cruzam o círculo interno, uma

imagem do ponto alvo pode ser feita, significando que uma revisita ocorreu. O momento de aquisição da imagem ocorre quando o plano de aquisição da Figura 7.14 contém o ponto alvo (centro dos círculos). Nesse momento, a situação é como mostrado na Figura 7.14. Portanto, se a distância na superfície entre o traço orbital e o ponto alvo é menor do que 204km, o ponto alvo está dentro da Zona Nadir e a imagem não pode ser feita. Por outro lado, se a distância é maior do que 204+395= 599km, o ponto alvo está fora de alcance e a imagem também não pode ser feita. Na Figura 7.15, os dois círculos concêntricos são representativos dessas restrições, relativamente ao traço orbital de qualquer satélite. A Figura 7.15 ilustra também três traços orbitais que permitem o imageamento por radar do ponto alvo. Isto ocorre porque, no momento de aquisição da imagem, indicada pelas linhas pontilhadas, os traços orbitais estão dentro do círculo externo e fora do círculo interno. As setas indicam a direção do movimento dos satélites. O traço orbital rotulado por "A" é a parte ascendente de uma das órbitas de referência de um satélite tipo MAPSAR (órbita heliosíncrona), enquanto o rotulado por "D" é uma descendente. O traço orbital rotulado por "E" é a parte ascendente de uma órbita quase equatorial, com inclinação ao redor de 10^o e demais parâmetros similares aos da órbita de referência do MAPSAR.

Durante a propagação de órbita, a seguinte estratégia é usada para verificar se uma revisita ocorreu. Todas as passagens cujos ângulos de elevação são maiores que $41{,}9^o$ são monitorados e os valores de elevação correspondentes são armazenados. Imediatamente após essas passagens, o valor máximo de elevação é verificado. Se ele é menor do que 70^o, uma revisita é computada. O ângulo de elevação máximo é usado porque ele define o instante no qual o ponto alvo está dentro do plano de aquisição de imagem do satélite. Dito de uma maneira um pouco mais detalhada, a cada passo da propagação de órbita o ângulo de elevação do satélite relativamente ao ponto alvo é calculado. Se ele ultrapassar $41{,}9^o$, os ângulos de elevação dos passos subseqüentes são armazenados até o final da passagem, isto é, até a elevação ir abaixo de $41{,}9^o$ novamente. Quando isto acontece, o maior valor dentre os armazenados é recuperado e se ele for menor do que 70^o, uma revisita é computada com a atualização dos tempos de revisita médio e máximo.

7.3.3 – Formulação do problema biobjetivo de projeto da constelação MAPSAR

Matematicamente, o problema pode ser posto como:

$$\text{Minimizar } \mathbf{F}(\mathbf{X}) = [\ F_1(\mathbf{X}) \quad F_2(\mathbf{X})\ ] \tag{6.1}$$

Sujeito a: $\mathbf{X_{MIN}} \leq \mathbf{X} \leq \mathbf{X_{MAX}}$

onde $F_1(\mathbf{X})$ é o tempo de revisita médio e $F_2(\mathbf{X})$ é o tempo de revisita máximo sobre o ponto alvo. O objetivo é minimizar ambos $F_1(\mathbf{X})$ e $F_2(\mathbf{X})$, obtendo a fronteira de Pareto e as respectivas soluções de projeto. As variáveis de projeto são representadas pelo vetor **X** no processo de otimização. Os vetores $\mathbf{X_{MIN}}$ e $\mathbf{X_{MAX}}$ contêm as restrições laterais para cada uma das variáveis, estabelecendo seus limites admissíveis. O número de variáveis de projeto é diferente para cada um dos três tamanhos de constelação (número de satélites na mesma) sob investigação. Essas constelações, chamadas constelações 1, 2 e 3 de agora em diante, têm o vetor **X** descrito na Tabela 7.9.

Tabela 7.9. Variáveis para otimização.

Constelação	**X**	**Variável**	**Descrição**	**Satélite**	**Intervalo Permitido $\mathbf{X_{MIN}} \sim \mathbf{X_{MAX}}$**
1	1	I_2	Inclinação	2	0 ~ 20°
	2	Ω_2-Ω_1	Defasagem em ascensão reta	2	0 ~ 360°
	3	M_2-M_1	Defasagem em anomalia média	2	0 ~ 360°
2	1	I_2	Inclinação	2	0 ~ 20°
	2	I_3	Inclinação	3	0 ~ 20°
	3	Ω_2-Ω_1	Defasagem em ascensão reta	2	0 ~ 360°
	4	Ω_3-Ω_1	Defasagem em ascensão reta	3	0 ~ 360°
	5	M_2-M_1	Defasagem em anomalia média	2	0 ~ 360°
	6	M_3-M_1	Defasagem em anomalia média	3	0 ~ 360°
3	1	I_2	Inclinação	2	0 ~ 20°
	2	I_3	Inclinação	3	0 ~ 20°
	3	I_4	Inclinação	4	0 ~ 20°
	4	Ω_2-Ω_1	Defasagem em ascensão reta	2	0 ~ 360°
	5	Ω_3-Ω_1	Defasagem em ascensão reta	3	0 ~ 360°
	6	Ω_4-Ω_1	Defasagem em ascensão reta	4	0 ~ 360°
	7	M_2-M_1	Defasagem em anomalia média	2	0 ~ 360°
	8	M_3-M_1	Defasagem em anomalia média	3	0 ~ 360°
	9	M_4-M_1	Defasagem em anomalia média	4	0 ~ 360°

Nota: Ω_1 e M_1 pertencem ao satélite 1 (satélite de referência) e são constantes.

Exceto pela inclinação, todas as variáveis de projeto são definidas pela diferença do parâmetro de referência correspondente do satélite 1. Parece ser útil visualizar as constelações obtidas pela comparação direta dos deslocamentos angulares relativamente ao satélite de referência.

Para um **X** dado, os respectivos elementos keplerianos de todos os satélites são calculados e também os vetores de estado correspondentes. As órbitas dos satélites são então propagandas por um período de 7 dias com um passo de 30 segundos. No final, os valores finais de $F_1(\mathbf{X})$ e $F_2(\mathbf{X})$ são retornados (o modelo do problema direto). Os valores de **X** são gerados por M-GEO, durante sua busca, e então repassados ao programa de propagação de órbita, o qual está acoplado ao M-GEO. Todas as variáveis de projeto foram codificadas usando 8 bits, o que significa uma resolução de 0,078° para a inclinação e 1,406° para as demais variáveis. O intervalo admissível para a inclinação foi encurtado de 0~180° para 0~20° porque, como o ponto alvo é próximo ao equador (-6° em Latitude), é previsível que inclinações quase equatoriais terão desempenho melhor, em termos de tempo de revisita.

7.3.4 – Resultados

O algoritmo M-GEO foi executado para cada Constelação descrita na seção anterior. Para cada execução, 25 sementes diferentes (vetores **X** aleatórios) foram utilizadas para reiniciar a busca. Para cada semente, 10000 avaliações de $\mathbf{F}(\mathbf{X}) = [\ F_1(\mathbf{X})\ \ F_2(\mathbf{X})\ ]$ foram disponibilizadas e este foi o critério de parada utilizado. Durante todo o tempo, a fronteira de Pareto foi armazenada e atualizada. Deste modo, a busca pela fronteira de Pareto de cada constelação usou 250000 avaliações de $\mathbf{F}(\mathbf{X})$. Sementes diferentes foram usadas de modo a gerar resultados que são robustos do ponto de vista estatístico. Para as três execuções τ=2 foi usado. Para o problema “mais pesado” da Constelação 3, o tempo de computação foi de aproximadamente 3 dias em um Intel Pentium 4 2,6GHz PC com 512MB RAM.

As Figuras 7.16 até 7.18 mostram as fronteiras de Pareto para as Constelações 1 a 3, respectivamente. Os pontos da fronteira de Pareto aproximada obtidos com o

M-GEO estão conectados por linhas para melhor visualização. Para a primeira Constelação, somente cinco soluções de projeto diferentes foram encontradas para o conjunto de Pareto, levando a uma fronteira de Pareto com quatro pontos. Para a segunda Constelação, 26 soluções de projeto diferentes foram encontradas para o conjunto de Pareto, levando a uma fronteira de Pareto com 20 pontos. Para a terceira Constelação, 10 soluções de projeto e 10 pontos da fronteira de Pareto diferentes foram encontrados.

O que é prontamente visto nessas figuras é que uma otimização somente do tempo de revisita médio $F_1(\mathbf{X})$ leva a uma enorme deterioração do tempo de revisita máximo $F_2(\mathbf{X})$. Por outro lado, otimização somente do tempo de revisita máximo $F_2(\mathbf{X})$ acarreta apenas uma leve piora do tempo de revisita médio $F_1(\mathbf{X})$. Um comportamento similar foi observado em (Williams et al., 2001), onde constelações para cobertura esparsa global foram estudadas. Este resultado é importante, porque sugere que ao lidar com otimizações monoobjetivo de constelações destes tipos, usar o tempo de revisita máximo como critério é preferível.

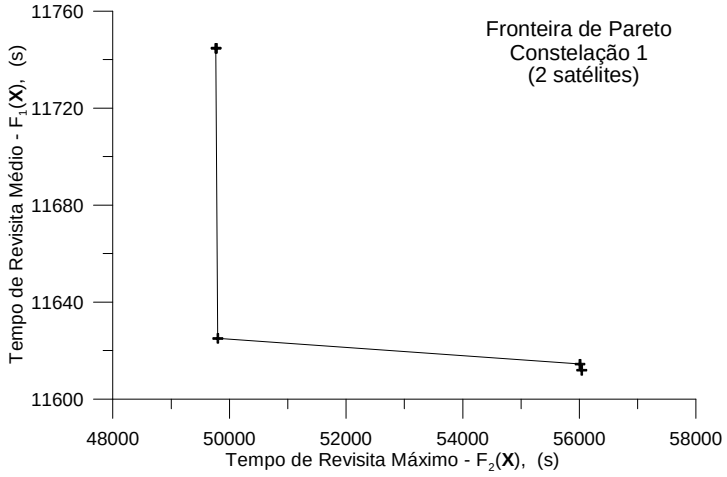


Figura 7.16 – Fronteira de Pareto da Constelação 1.

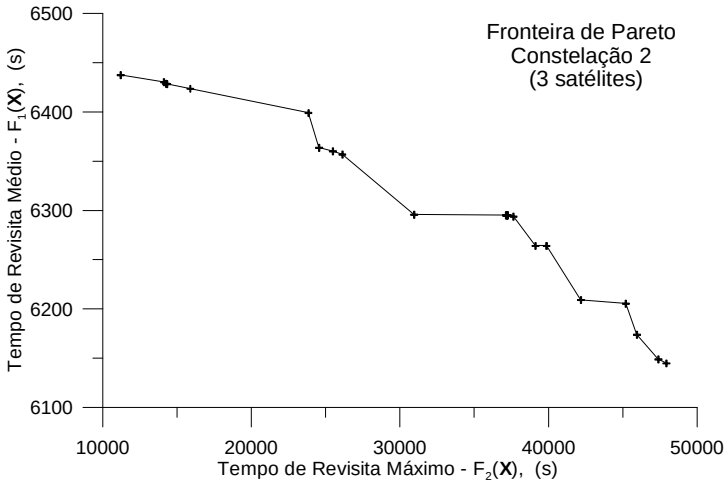


Figura 7.17 – Fronteira de Pareto da Constelação 2.

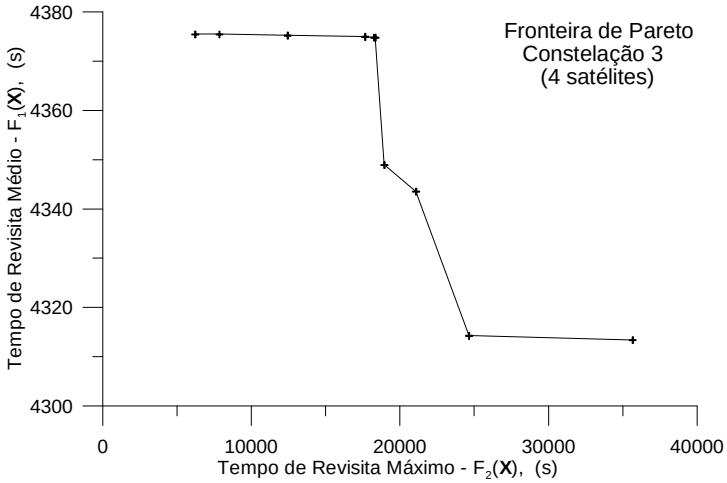


Figura 7.18 – Fronteira de Pareto da Constelação 3.

Na Tabela 7.10, o conjunto de soluções de projeto (vetores **X**) encontrado para todos os pontos da fronteira de Pareto (valores em negrito) da Constelação 1 são mostrados, enquanto as Tabelas 7.11 e 7.12 fazem o mesmo para as Constelações 2 e 3, respectivamente.

Tabela 7.10 – Soluções da Constelação 1 para os pontos da fronteira de Pareto.

I_2	Ω_2-Ω_1	M_2-M_1	F_1	F_2
3,76	272,5	63,5	**11744,7**	**49770,0**
3,84	272,5	63,5		
4,16	272,5	77,6	**11625,0**	**49800,0**
4,24	285,2	94,6	**11614,4**	**56010,0**
4,24	285,2	101,6	**11611,9**	**56040,0**

Tabela 7.11 - Soluções da Constelação 2 para os pontos da fronteira de Pareto.

I_2	I_3	Ω_2-Ω_1	Ω_3-Ω_1	M_2-M_1	M_3-M_1	F_1	F_2
4,16	4,16	187,8	360,0	328,9	86,1	**6437,4**	**11220,0**
4,16	4,16	180,7	360,0	1,4	86,1	**6430,2**	**14130,0**
4,08	4,16	180,7	360,0	9,9	86,1	**6428,6**	**14280,0**
4,16	4,16	142,6	303,5	50,8	357,2	**6428,3**	**14340,0**
4,16	4,16	344,5	176,5	187,8	43,8	**6423,5**	**15900,0**
4,16	4,16	224,5	98,8	128,5	314,8	**6398,9**	**23850,0**
3,92	4,24	234,4	114,4	303,5	83,3	**6363,5**	**24570,0**
3,92	4,16	234,4	114,4	357,2	83,3	**6360,0**	**25500,0**
4,16	4,00	97,4	213,2	45,2	360,0	**6356,8**	**26130,0**
3,92	4,16	211,8	114,4	334,6	83,3	**6295,6**	**30960,0**
3,92	4,16	189,2	114,4	360,0	88,9	**6295,0**	**37140,0**
3,92	4,08	189,2	114,4	360,0	88,9		
3,92	4,16	189,2	114,4	360,0	83,3		**37230,0**
3,92	4,08	189,2	114,4	360,0	83,3		
3,92	4,24	189,2	114,4	360,0	83,3		
4,00	4,16	189,2	114,4	360,0	83,3		
4,00	4,08	189,2	114,4	360,0	83,3		
4,00	4,24	189,2	114,4	360,0	83,3		
4,16	4,00	145,4	213,2	50,8	360,0	**6293,8**	**37650,0**
4,16	4,16	323,3	278,1	52,2	354,4	**6264,0**	**39120,0**
4,16	4,16	323,3	280,9	52,2	309,2		**39870,0**
4,16	4,08	266,8	302,1	117,2	360,0	**6209,0**	**42180,0**
4,16	4,08	280,9	300,7	306,4	21,2	**6205,3**	**45210,0**
4,08	4,16	271,1	286,6	77,6	199,1	**6173,7**	**45960,0**
4,16	4,16	266,8	279,5	128,5	337,4	**6148,7**	**47400,0**
4,16	4,16	266,8	276,7	117,2	360,0	**6144,7**	**47940,0**

Tabela 7.12 - Soluções da Constelação 3 para os pontos da fronteira de Pareto.

I_2	I_3	I_4	Ω_2-Ω_1	Ω_3-Ω_1	Ω_4-Ω_1	M_2-M_1	M_3-M_1	M_4-M_1	F_1	F_2
4,00	4,08	3,84	355,8	240,0	121,4	168,0	343,1	98,8	**4375,4**	**6240,0**
4,00	4,08	3,84	175,1	330,4	121,4	179,3	343,1	98,8		**7860,0**
4,00	4,00	3,69	148,2	132,7	309,2	76,2	88,9	271,1	**4375,2**	**12450,0**
4,00	4,08	4,16	175,1	151,1	21,2	300,7	340,2	153,9	**4375,0**	**17670,0**
4,00	4,00	3,69	148,2	64,9	218,8	76,2	88,9	271,1	**4374,8**	**18270,0**
4,00	4,00	3,69	148,2	64,9	218,8	76,2	88,9	276,7		**18360,0**
4,00	4,08	4,16	84,7	240,0	201,9	122,8	343,1	337,4	**4348,9**	**18960,0**
4,00	4,08	4,16	175,1	240,0	105,9	122,8	343,1	337,4	**4343,5**	**21090,0**
4,00	4,08	3,84	176,5	240,0	111,5	32,5	338,8	121,4	**4314,2**	**24660,0**
4,00	4,00	3,69	148,2	132,7	207,5	76,2	83,3	271,1	**4313,4**	**35670,0**

A partir das Tabelas 7.11 e 7.12 é possível observar um padrão comum, onde os satélites estão praticamente uniformemente distribuídos em termos da defasagem em ascensão reta Ω_j-Ω_1 , $j \in \{2,3,4\}$, com relação ao melhor valor de F_2 da fronteira de Pareto, enquanto eles estão muito mais próximos um do outro com relação ao melhor valor de F_1. Por exemplo, tomando as melhores soluções em F_2 como referência e explicando um pouco mais o que acaba de ser dito, para a Constelação 2, com melhor F_2=11220, significa $(\Omega_3-\Omega_1)-(\Omega_2-\Omega_1)$ = 360,0°-187,8°= 172,2°, o que é próximo de uma distribuição uniforme de 360°/2=180° para dois satélites. Para a Constelação 3, com melhor F_2=6240, significa $(\Omega_2-\Omega_1)-(\Omega_3-\Omega_1)$ = 355,8°-240,0°= 115,8° ≅ $(\Omega_3-\Omega_1)-(\Omega_4-\Omega_1)$ = 240,0°-121,4°= 118,6° ≅ $(\Omega_4-\Omega_1)-(\Omega_2-\Omega_1)$ = 121,4°-355,8° = -234,4° = 125,6°, o que é próximo de uma distribuição uniforme de 360°/3=120° para três satélites. Em contrapartida, agora tomando as melhores soluções em F_1, a maior diferença $(\Omega_j-\Omega_1)-(\Omega_k-\Omega_1)$, $\forall j \neq k$, j e k $\in$ {2,3,4} foi 9,9° para a Constelação 2 e 74,8° para a Constelação 3. Portanto, é válido dizer que, caso uma distribuição uniforme da defasagem em ascensão reta fosse utilizada *a priori* como critério de projeto, a solução obtida seria ótima do ponto de vista do tempo de revisita máximo e seria péssima do ponto de vista do tempo de revisita médio, com perda de quase 300 segundos no caso da Constelação 2 (vide diferença entre o maior e o menor valor de F_1 na Tabela 7.11).

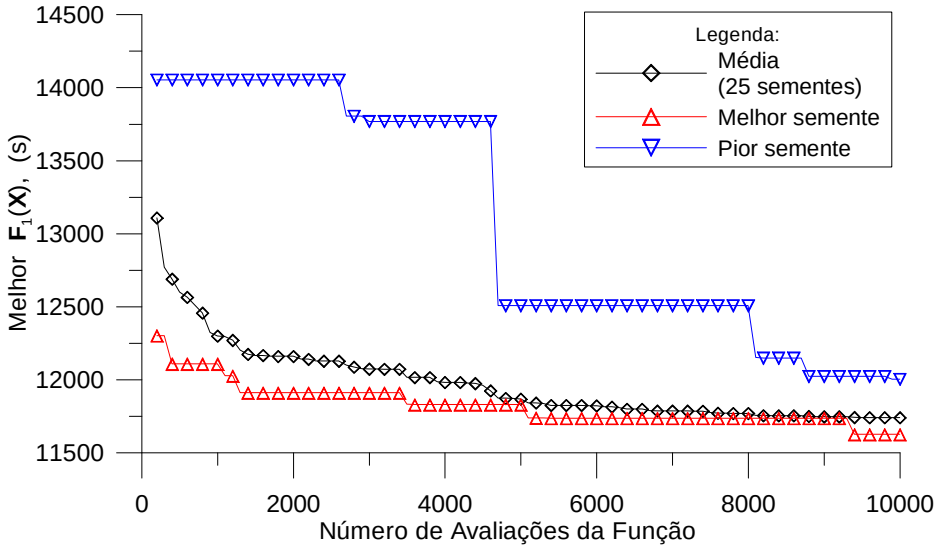


Figura 7.19a – Evolução de $F_1(\mathbf{X})$ (Constelação 1).

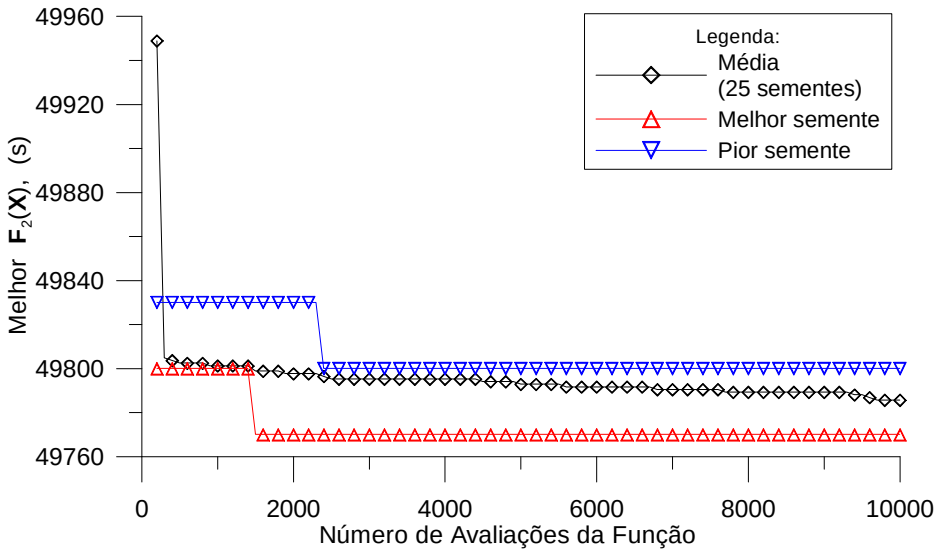


Figura 7.19b – Evolução de $F_2(\mathbf{X})$ (Constelação 1).

Nas Figuras 7.19 a 7.21, curvas mostrando o comportamento médio (sobre as 25 sementes) para o melhor $F_1(\mathbf{X})$ e para o melhor $F_2(\mathbf{X})$ são traçadas para as Constelações 1, 2 e 3. Adicionalmente, as curvas da melhor e da pior semente dentre as 25 também são traçadas. Olhando as Figuras 7.19a e 7.19b, é possível considerar que para a Constelação 1, a evolução média de $F_1(\mathbf{X})$ e $F_2(\mathbf{X})$ praticamente estabiliza após 6000 avaliações. O mesmo parece acontecer após 7000 avaliações para a Constelação 2 e após 4000 para a Constelação 3.

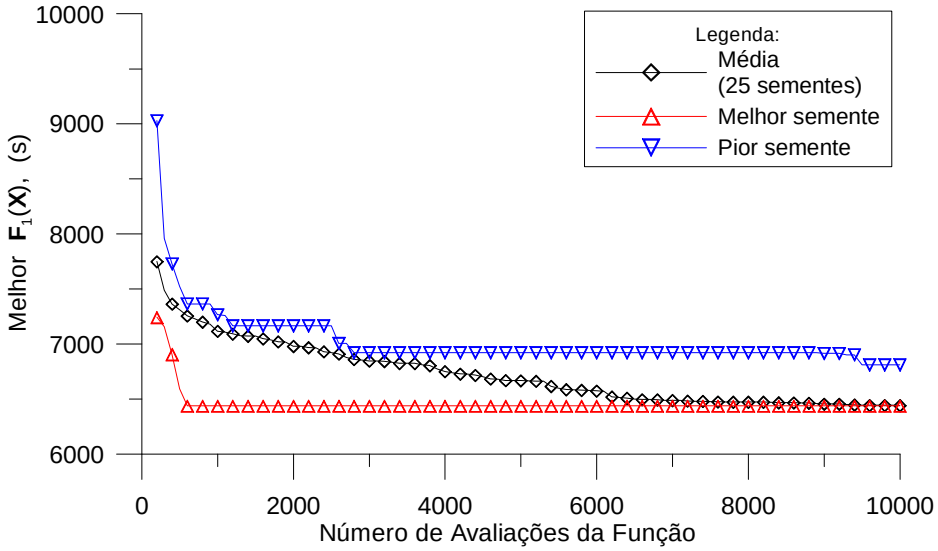


Figura 7.20a – Evolução de $F_1(\mathbf{X})$ (Constelação 2).

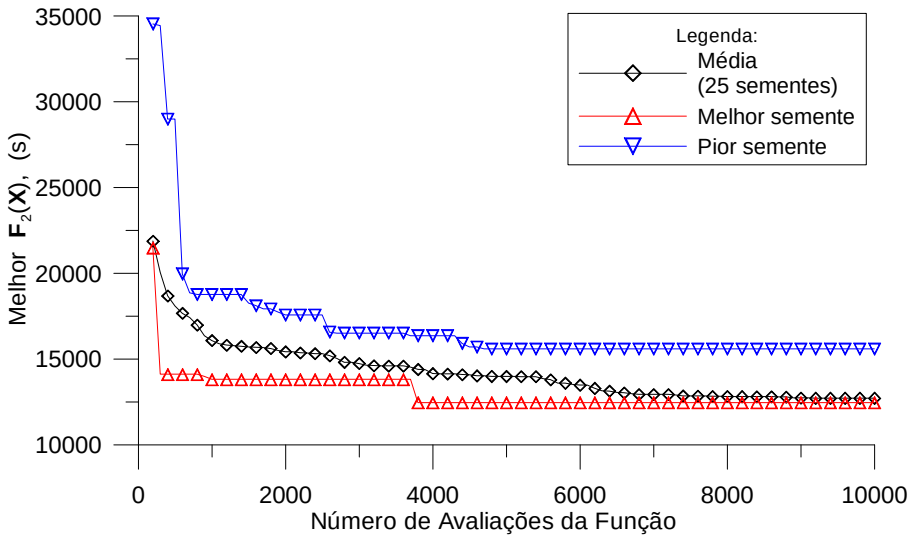


Figura 7.20b – Evolução de $F_2(\mathbf{X})$ (Constelação 2).

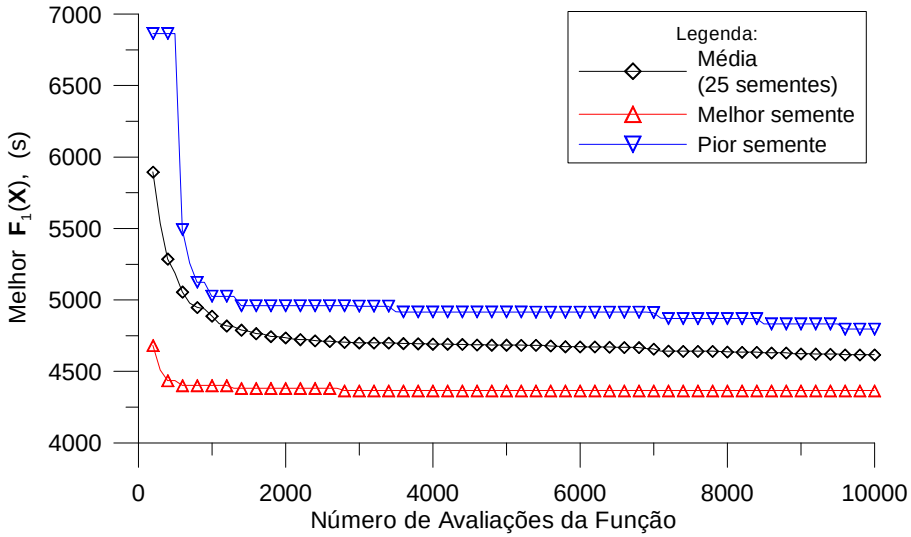


Figura 7.21a – Evolução de $F_1(\mathbf{X})$ (Constelação 3).

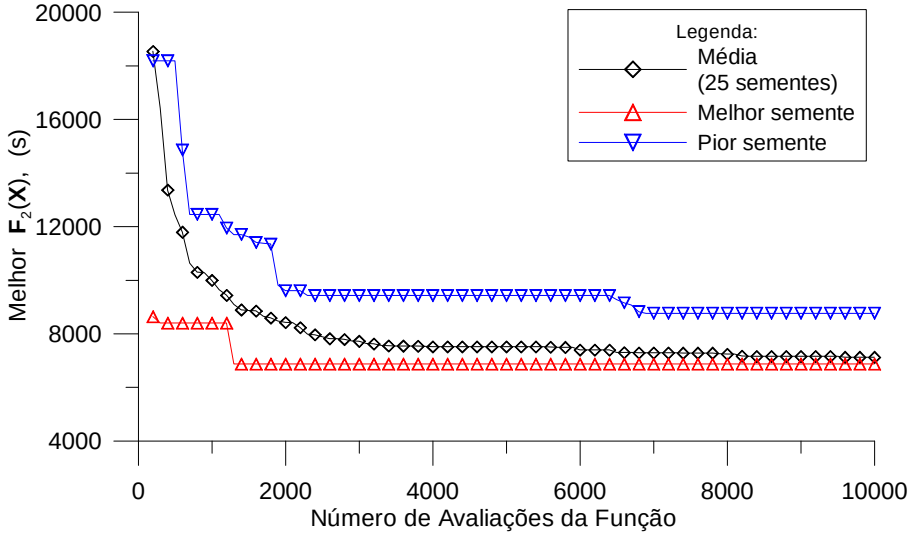


Figura 7.21b – Evolução de $F_2(\mathbf{X})$ (Constelação 3).

7.4 – Conclusões

Este Capítulo descreveu as aplicações envolvendo versões mono e multiobjetivo do algoritmo GEO no projeto otimizado do sistema de controle térmico de uma plataforma espacial multimissão (PMM) e também na configuração de uma constelação de satélites do tipo MAPSAR.

No caso da PMM, o balanço geral que pode ser feito com base nos resultados alcançados, é bastante positivo. A abordagem de otimização baseada em um modelo térmico simplificado revelou-se plenamente satisfatória, sendo que as soluções obtidas, quando aplicadas num modelo térmico detalhado, revelaram distribuições de temperatura bastante próximas daquelas previstas pelo modelo simplificado.

No caso da constelação MAPSAR, os resultados indicaram que com apenas dois satélites é possível ter cobertura global e um tempo de revisita máximo sobre o ponto alvo de 49770s, isto é, 13,825 horas, menos que um dia, como almejado para o monitoramento de desastres. Os resultados indicaram ainda que uma solução tradicional, baseada em distribuições simétricas de planos orbitais, é otimizada do ponto de vista do tempo de revisita máximo apenas.

CAPÍTULO 8

CONCLUSÕES E SUGESTÕES

Em 1993, Bak e Sneppen desenvolveram um modelo simplificado para simular o processo evolutivo. Inspirados neste modelo, Boettcher e Percus (2001) propuseram uma meta-heurística para problemas difíceis de otimização combinatória e a denominaram EO - *Extremal Optimization.* Em Sousa (2002), um algoritmo variante do EO, denominado GEO – *Generalized Extremal Optimization* foi desenvolvido. O GEO é uma generalização do EO, que estendeu sua aplicabilidade a virtualmente qualquer problema de otimização.

Como acontece com qualquer método novo, uma quantidade significativa de estudo deve ser efetuada, de modo a desenvolver o potencial presente no método original. Esta Tese de Doutorado explorou e ampliou as potencialidades do GEO, tendo efetuado estudos aprofundados do seu funcionamento e, com isso, obtendo, na maioria das vezes, versões mais eficientes e/ou de maior aplicabilidade.

Baseados nos estudos sobre o funcionamento do GEO, quatro sugestões de aperfeiçoamentos foram colocadas e desenvolvidas. Destes, três apresentaram bons resultados. Dois dos aperfeiçoamentos destacaram-se pela melhora significativa de desempenho que propiciaram. Um deles se baseia na detecção da estagnação da busca e sua eliminação por meio de auto-reinicializações. O outro se baseia na mudança da representação das variáveis de projeto, de binária para real e na realização das mutações dessas variáveis por meio de estruturas mais flexíveis do que aquela permitida pela representação binária. Os resultados obtidos por este último aperfeiçoamento foram muito significativos, tendo sido utilizados na elaboração de uma versão híbrida do GEO com EE.

Dando prosseguimento aos estudos das potencialidades do GEO, sua paralelização foi estudada. Dentro desse contexto, uma versão paralelizada do GEO, denominada GEOPAR-1 foi desenvolvida e seu desempenho foi analisado para várias cargas de processamento. Os testes foram realizados num *cluster* composto por 6 nós biprocessadores interligados por uma rede Gigabit Ethernet. Uma função teste com dimensionalidade e tempo de avaliação configuráveis foi desenvolvida, de forma a simular cargas de processamento típicas de aplicações complexas. Almejou-se, com isso,

comprovar a eficiência do GEO como algoritmo paralelo e permitir sua aplicação no projeto otimizado de sistemas espaciais complexos, onde o alto desempenho computacional seja uma necessidade. Os testes de desempenho efetuados comprovaram que o GEOPar-1 tem um grande potencial como otimizador paralelo, obtendo eficiência acima de 0,85 em todos os testes que procuraram simular aplicações complexas, computacionalmente custosas. Este fato, somado à universalidade de aplicação do GEO, característica mantida inalterada no GEOPar-1, aumentou a aplicabilidade do algoritmo em problemas reais de otimização, principalmente naqueles em que o custo de processamento da avaliação da função objetivo seja elevado.

Atualmente é comum o emprego de técnicas de hibridização para melhorar as características dos algoritmos de otimização. A hibridização era outro campo até então não explorado com o GEO. Nesta Tese, tais técnicas foram utilizadas no desenvolvimento de versões híbridas do algoritmo GEO com os algoritmos RS e EE. Conseguiu-se, com isso, como demonstrado pelos bons resultados obtidos, incorporar ao GEO parte das boas características presentes nos dois outros algoritmos citados e aumentar sua eficiência quando aplicado a problemas em geral ou, ao menos, aumentar sua eficiência em aplicações específicas. No caso específico do híbrido GEO + EE, comparações de desempenho com versões recentes de AGs com codificação real (Someya e Yamamura, 2001; Ballester e Carter, 2004), alguns reportados como detentores dos melhores desempenhos para algumas das funções testadas, demonstraram um desempenho geral superior para o híbrido GEO + EE, especialmente quando o foco da comparação concentrou-se nas funções altamente multimodais.

No campo da otimização multiobjetivo, área que também não havia sido explorada com o GEO, um novo algoritmo de otimização multiobjetivo, chamado M-GEO, foi desenvolvido e testado com diversas funções teste comuns na literatura da área. Mais uma vez, os bons resultados obtidos comprovaram que também na área de otimização multiobjetivo o GEO tem potencial. De fato, a partir dos resultados obtidos pelo M-GEO foi possível concluir que a fronteira de Pareto obtida em dois outros estudos para uma função teste específica estava incompleta (Srinivas e Deb, 1994; Mason et al., 1998).

A fim de comprovar a capacidade de otimização das versões desenvolvidas em aplicações práticas, algumas delas foram utilizadas para a otimização de dois sistemas espaciais: (i) o projeto térmico dos radiadores da Plataforma Multimissão (PMM) do Inpe e; (ii) a otimização da configuração de uma constelação de satélites de sensoriamento remoto do tipo MAPSAR. No caso da PMM, o problema foi formulado e resolvido como um problema monoobjetivo e também como um problema multiobjetivo. Dos dois jeitos, as soluções obtidas foram extremamente satisfatórias. Obviamente, a solução do problema multiobjetivo trouxe à tona um panorama mais completo, fato propiciado pelo conjunto de soluções de projeto associadas à fronteira de Pareto obtida. No caso do MAPSAR, o problema foi posto e resolvido apenas na forma multiobjetivo. Os resultados indicaram que com apenas dois satélites é possível ter tanto a cobertura global quanto um tempo de revisita máximo sobre o ponto alvo menor que um dia, como almejado para o monitoramento de desastres.

Fazendo uma pequena retrospectiva, das oito sugestões de desenvolvimentos futuros feitas ao final do trabalho que deu origem ao GEO (Sousa, 2002), seis foram abordadas nesta Tese. Quatro referem-se a estudos comentados nos parágrafos anteriores: paralelização, hibridização, representação das variáveis, e abordagem multiobjetivo. As outras duas referem-se a sugestões mais específicas, que visavam descobrir porque o desempenho do GEO canônico para função de Schwefel (FT_3) tinha sido tão desfavorável. A partir dos estudos efetuados quando do desenvolvimento da Sugestão de Aprimoramento 4, foi possível concluir que a representação binária foi a causa do fraco desempenho apresentado pelo GEO canônico com a função de Schwefel, função pertencente à classe das assim chamadas *deceptive functions*.

Como sugestões de futuros desenvolvimentos colocam-se:

1) Aprofundar a análise de desempenho das versões desenvolvidas nesta Tese às funções com restrições, visando o desenvolvimento de formas eficientes de lidar com as restrições, transformando-as em fontes de informação útil ao processo de otimização.

2) Desenvolver versões paralelizadas e multiobjetivo com as versões desenvolvidas nesta Tese e que apresentaram os melhores desempenhos.

3) Implementar a aplicação monoobjetivo da PMM descrita nesta Tese usando o GEOPar-1.

4) Implementar uma versão paralela do algoritmo M-GEO. O esquema mestre-escravo utilizado no GEOPar-1 também pode ser aplicado no M-GEO, inclusive com vantagens, visto que o número de avaliações de funções objetivo por iteração do algoritmo M-GEO é, relativamente ao GEO, multiplicado por NFOBJ, o número de funções objetivo.

5) Aprofundar os estudos sobre a melhor forma de distribuir as mutações verificando, por exemplo, se a tendência da distribuição espacial uniforme ser melhor do que a distribuição totalmente aleatória se mantém quando testada num conjunto maior de problemas.

Finalmente, cabe mencionar as publicações originadas deste trabalho. A primeira delas foi sobre a primeira versão aprimorada do GEO, apresentada no 3º Workshop dos Cursos de Computação Aplicada do INPE – III WORCAP (Galski et al., 2003). Em seguida, a aplicação desta versão aprimorada no projeto térmico da Plataforma Multimissão do INPE foi apresentada no *Inverse Problems, Design and Optimization Symposium* , realizado no Rio de Janeiro, em março de 2004 (Galski et al.,2004a). A versão revisada deste trabalho foi aceita para publicação no periódico *Inverse Problems in Science and Engineering* (Galski et al., 2007). Ainda em 2004, uma nova leva de resultados referentes à mesma aplicação foi incluída como exemplo de aplicação prática no Capítulo que falava sobre o GEO no livro *Recent Developments in Biologically Inspired Computing* (Sousa et al., 2005a). No 4º Workshop dos Cursos de Computação Aplicada do INPE – IV WORCAP, a versão paralela do GEO, GEOPar-1, foi apresentada (Galski et al., 2004b). Em 2005, a aplicação envolvendo o M-GEO e o projeto otimizado da constelação de satélites do tipo MAPSAR descrita nesta Tese foi apresentada na *5th International Conference on Inverse Problems in Engineering: Theory and Practice*, realizada em Cambridge, Inglaterra (Galski et al., 2005a). No 5º Workshop dos Cursos de Computação Aplicada do INPE – V WORCAP, a segunda versão aprimorada do GEO foi apresentada (Galski et al., 2005b). Ainda em 2005, uma nova metodologia para o projeto térmico otimizado de satélites foi submetida para publicação no periódico *Journal of Spacecraft and Rockets*, tendo sido aceita (Muraoka et al., 2006).

REFERÊNCIAS BIBLIOGRÁFICAS

ABREU, B. T.; MARTINS, E.; SOUSA, F. L. de Automatic test data generation for path testing using a new stochastic algorithm. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE (SBES2005), 19., 2005, Uberlândia. **Anais...** Uberlândia, UFU, 2005. p. 247-262. ISBN 85-7669-030-6.

ALBA, E.; TOMASSINI, M. Parallelism and Evolutionary Algorithms. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 5, p. 443-462, 2002.

ALI, M. M.; KHOMPATRAPORN, C.; ZABINSKY, Z. B. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. **Journal of Global Optimization**, v. 31, p. 635-672, 2005.

AMIRJANOV, A. A changing rate Genetic Algorithm. **International Journal for Numerical Methods in Engineering**, v. 61, p. 2660-2674, 2004.

ARORA, J. S. **Introduction to optimum design**. New York: MacGraw-Hill Series in Mechanical Engineering, 1989.

BALLESTER, P. J.; CARTER, J. N. An effective real-parameter Genetic Algorithm with parent centric normal crossover for multimodal optimization. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE (GECCO-2004), 2004, Seattle, USA. **Proceedings...** Seattle: Kalyanmoy Deb et al. (Eds.), 2004. p. 901-913. (LNCS 3102) (ISBN-10: 3540223444) (ISBN-13: 978-3540223443)

BAK, P. **How nature works**. New York: Copernicus, Springer-Verlag, 1996.

BAK, P.; CHEN, K. Self-organized criticality. **Scientific American**, v. 264, p. 46-53, 1991.

BAK, P.; SNEPPEN, K. Punctuated equilibrium and criticality in a simple model of evolution. **Physical Review Letters**, v. 71, n. 24, p. 4083-4086, 1993.

BAK, P.; TANG, C.; WIESENFELD, K. Self-organized criticality: An explanation of 1/f noise. **Physical Review Letters**, v. 59, n. 4, p. 381-384, 1987.

BARBULESCU, L.; WATSON, J.P.; WHITLEY, L.D. Dynamic representations and escaping local optima: improving genetic algorithms and local search. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI-2000), 17., 2000, Austin, Texas, USA. **Proceedings...** Austin: American Association for Artificial Intelligence, 2000. (ISBN 0-262-51112-6).

BARNEY, B. **Introduction to Parallel Computing**. Livermore Computing, Lawrence Livermore National Laboratory, Livermore, CA, USA. 2004-2006. Disponível em: <http://www.llnl.gov/computing/tutorials/parallel_comp/> Acesso em: 31 maio 2004.

BAUER, J.; GUTKOWSKI, W. Discrete structural optimization: a review. In: WORLD CONGRESS OF STRUCTURAL AND MULTIDISCIPLINARY OPTIMIZATION, 1., 1995, Goslar, Lower Saxony, Germany. **Proceedings...** Goslar: N. Olhoff e G. Rozvany (Eds.), 1996. p. 901-908. (ISBN-10: 0-08-042267-5) (ISBN-13: 978-0-08-042267-1)

BOETTCHER, S.; PERCUS A. Extremal optimization: methods derived from co-evolution. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE (GECCO-99), 1999, Orlando, Florida, USA. **Proceedings...** New York: Morgan Kaufmann, 1999. p. 825-832. (ISBN-10: 1558606114) (ISBN-13: 978-1558606111)

BOETTCHER, S.; PERCUS, A. Nature's way of optimizing. **Artificial Intelligence**, v. 119, n. 1-2, p. 275-286, 2000.

BOETTCHER, S.; PERCUS, A.; GRIGNI, M. Optimizing through co-evolutionary avalanches. In: INTERNATIONAL CONFERENCE ON PARALLEL PROBLEM SOLVING FROM NATURE, 6., 2000, Paris, France. **Proceedings...** Paris: Springer Verlag, 2000. p. 447-456. (ISBN: 9783540410560)

BOETTCHER, S.; PERCUS, A. Optimization with extremal dynamics. **Physical Review Letters**, v. 86, n. 23, p. 5211-5214, 2001.

BONABEAU, E.; DORIGOÚ, M.; THERAULAZ, G. Inspiration for optimization from social insect behaviour. **Nature**, v. 406, n. 6791, p. 39-42, 2000.

BORDFELDT, A.; GEHRING, H.; MACK, D. A parallel tabu search algorithm for solving the container loading problem. **Parallel Computing**, v. 29, n. 5, p. 641-662, 2003.

BOX, M.J. A new method of constrained optimization and a comparison with other methods. **The Computer Journal**, v. 8, p. 42-52, 1965.

BROYDEN, C.G. The convergence of a class of double rank minimization algorithms, parts 1 and 2. **Journal of the Institute for Mathematics and its Applications**, v. 6, p. 76-90 and p. 222-231, 1970.

BURKARDT, J. **Open MP Portable Parallel Processing**. School of Computational Science, Florida State University, Tallahassee, Florida, USA. 2004-2006. Disponível em: <http://www.csit.fsu.edu/~burkardt/cpp_src/open_mp/open_mp.html> Acesso em: 01? maio? 2004.

CARDOSO, H. P.; MURAOKA, I.; BASTOS, J. L. F.; BAMBACE, L. A. W.; FILHO, O. B. O.; LEITE, R. M. G. **Manual do software PCTER (INPE)**. São José dos Campos, 1990. 107 p.

CASTRO, L.N.; TIMMIS, J. An Artificial Immune Network for Multimodal Function Optimization. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION (CEC'02), 2002, Honolulu, HI, USA. **Proceedings...** Honolulu: IEEE Xplore, 2002. v. 1, p. 699-674. (ISBN: 0-7803-7282-4)

CHANDRA, R.; DAGUM, L.; KOHR, D.; MAYDAN, D.; MCDONALD, J.; MENON, R. **Parallel programming in OpenMP**, San Fransisco, USA: Morgan Kaufmann, 2001. (ISBN-10: 1558606718) (ISBN-13: 978-1558606715)

CHENG, F.Y.; LI, DAN. Genetic algorithm development for multiobjective optimization of structures. **AIAA Journal**, v. 36, n. 6, p. 1105-1112, 1998.

COELLO, C.A.C. **Evolutionary Multiobjective Optimization (EMOO) Web page**. CINVESTAV-IPN, Sección de Computación, Departamento de Ingeniería Eléctrica, Col. San Pedro Zacatenco, Mexico city, 2005-2006. Disponível em: <www.lania.mx/~ccoello/EMOO/> Acesso em: 01? jan? 2005.

COELLO, C.A.C. A comprehensive survey of evolutionary-based multiobjective optimization techniques. **Knowledge and Information Systems: An International Journal**, v.1, n.3, p.269-308, 1999.

COELLO, C.A.C.; CHRISTIANSEN, A.D. Two new GA-based methods for multiobjective optimization. **Civil Engineering Systems**, v. 15, n. 3, p. 207-243, 1998.

COELLO, C.A.C. A Short Tutorial on Evolutionary Multiobjective Optimization, **Int. Trans. Opl. Res.**, v. 3, n. 1, p. 1-21, 1996.

COLORNI, A.; DORIGO, M.; MAFFIOLI, F.; MANIEZZO, V.; RIGHINI, G.; TRUBIAN, M. Heuristics from nature for hard combinatorial optimization problems. **AIAA Journal**, v. 36, n. 6, p. 1105-1112, 1998.

CORNE, D.; DORIGO, M. E GLOVER, F**. New methods in optimization**. New York: McGraw Hill, 1999.

CRAIN, T.; BISHOP, R.H.; FOWLER, W. Interplanetary flyby mission optimization using a hybrid global-local search method. **Journal of Spacecraft and Rockets**, v. 37, n. 4, p. 468-474, 2000.

CROSSLEY, W.A.; WILLIAMS, E.A. Simulated annealing and genetic algorithms approaches for discontinuous coverage satellite constellation design. **Engineering Optimization**, v. 32, n. 1, p. 353-371, 2000.

DAVIS, L.D.; DE JONG, K.; VOSE, M.D.; WHITLEY, L.D. **Evolutionary algorithms.** The IMA Volumes in Mathematics and its Applications, v. 111. New York?: Springer, 1999.

DAVIDON, W.C. **Variable metric method for minimization.** Argonne, Illinois, USA: Argonne National Laboratory, p. 13-19, 1959. Report ANL-5990.

DEB, K.; ANAND, A.; JOSHI, D. A computationally efficient evolutionary algorithm for real-parameter optimization. **Evolutionary Computation**, v. 10, p. 371-395, 2002.

DENNIS, J.E.; WU, Z. **Parallel Continuous optimization**. Houston, TX, USA: University of Rice, Computational and applied mathematics department, Technical Report

TR-00-01, 2000. 38p. Disponível em: <http://www.caam.rice.edu/techrep.html>. Acesso em: 01 abr 2004.

DENNIS, J.E.; TORCZON, V. Direct search methods on parallel machines. **SIAM Journal on Optimization**, v. 1, n. 4, p. 448-474, Nov 1991.

DESAI, R.; PATIL, R. SALO: combining simulated annealing and local optimization for efficient global optimization, In: FLORIDA AI RESEARCH SYMPOSIUM (FLAIRS-'96), 9., 1996, Key West, FL, USA. **Proceedings...** Gainesville, FL, USA: University of Florida, 1996. p. 233-237.

DONGARRA, J. J. **Performance of various computers using standard linear equations software**. Knoxville, TN, USA: University of Tenessee, Computer Science Department, Report CS-89-85, 2003. 78p. Disponível em: <http://www.netlib.org/benchmark/performance.ps> Acesso em: 24 mar 2004.

EIBEN, A.E.; HINTERDING, R.; MICHALEWICZ, Z. Parameter control in Evolutionary Algorithms. **IEEE Transactions on Evolutionary Computation**, v. 3, n. 2, p. 124-141, 1999.

EIBEN, A. E.; SMITH, J. E. **Introduction to evolutionary computing**. Berlim, Alemanha: Springer-Verlag, 2003. 300 p.

ELDRED, M. S. **Optimization strategies for complex engineering applications.** Albuquerque, NM and Livermore, CA, USA: Sandia National Laboratories, Technical Report, 1998. 109p. (SAND98-0340-UC-705)

FONSECA, C.M.; FLEMING, P.J. An overview of evolutionary algorithms in multiobjective optimization. **Evolutionary Computation**, v.3, n.1, p. 1-16, 1995.

FOURER, B.; MORÉ, J. **NEOS Guide**. Optimization Technology Center, Argonne National Laboratory and Northwestern University, Argonne, Illinois, USA, 2004-2006. Disponível em: <http://www-fp.mcs.anl.gov/otc/Guide/>. Acesso em: 01? mar? 2004.

FOX, R. L. **Optimization methods for engineering design**. Boston, MA, USA: Addison-Wesley, 1971. 270p.

FREEMAN, J. A.; SKAPURA, I. **Neural networks: algorithms, applications and programming techniques**. New York, USA: Addison-Wesley, 1991. 550p.

FLETCHER, R. A new approach to variable metric algorithms. **The Computer Journal**, v. 13, n. 1?, p. 317-322, 1970.

FLETCHER, R.; POWELL, M. J. D. A rapidly convergent descent method for minimization. **The Computer Journal**, v. 6, n. 1?, p. 163, 1963.

FLETCHER, R.; REEVES, C. M. Function Minimization by Conjugate Gradients. **The Computer Journal**, v. 7, n. 2, p. 149-154, 1964.

FLYNN, M. J. Very high-speed computing systems. **Proceedings of the IEEE**, v. 54 , n. 12, p. 1901-1909, Dec. 1966.

FLYNN, M. J. Some computer organizations and their effectiveness. **IEEE Transactions on Computers**, v. C-21, n. 9, p. 948-960, 1972.

FUJITA, K.; HIROKAWA, N.; AKAGI, S.; KITAMURA, S.; YOKOHATA, H. Multi-objective optimal design of automotive engine using genetic algorithms. In: DESIGN ENGINEERING TECHNICAL CONFERENCES (DETC'98), 1998, Atlanta, Georgia, USA. **Proceedings...** New York, NY, USA: ASME, 1998. p. 1-11.

GALSKI, R. L.; SOUSA, F. L. de; RAMOS, F. M. Application of a New Evolutionary Algorithm to the Optimum Design of a Remote Sensing Satellite Constelation. In: INTERNATIONAL CONFERENCE ON INVERSE PROBLEMS IN ENGINEERING: THEORY AND PRACTICE, 5., 2005, Cambridge, UK. **Proceedings...** Cambridge: D. Lesnic (Ed.), v. II, G01, 2005a.

GALSKI, R. L.; SOUSA, F. L. de; RAMOS, F. M.; MURAOKA, I. Spacecraft thermal design with the generalized extremal optimization algorithm. In: INVERSE PROBLEMS, DESIGN AND OPTIMIZATION SYMPOSIUM, 2004, Rio de Janeiro, RJ, Brasil. **Proceedings...** Rio de Janeiro: E-papers Publishing House Ltd., 2004a. p. 264-271.

GALSKI, R. L.; SOUSA, F. L. de; RAMOS, F. M.; MURAOKA, I. Spacecraft thermal design with the generalized extremal optimization algorithm. **Inverse Problems in Science and Engineering**, v. 15, n. 1, p. 61-75, 2007.

GALSKI, R. L.; RAMOS, F. M.; SOUSA, F. L. de; PRETO, A. J.; STEPHANY, S. Implementação paralela do método da Otimização Extrema Generalizada. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE (IV WORCAP), 4., 2004, São José dos Campos, SP, Brasil. **Proceedings...** São José dos Campos: INPE (cd-rom), 2004b.

GALSKI, R. L.; RAMOS, F. M.; SOUSA, F. L. de Uma nova versão aprimorada do método da Otimização Extrema Generalizada. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE (V WORCAP), 5., 2005, São José dos Campos, SP, Brasil. **Proceedings...** São José dos Campos: INPE (cd-rom), 2005b.

GALSKI, R. L.; SOUSA, F. L. de; RAMOS, F. M. Um estudo para o aprimoramento do método da Otimização Extrema Generalizada. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA DO INPE (III WORCAP), 3., 2003, São José dos Campos, SP, Brasil. **Proceedings...** São José dos Campos: INPE (cd-rom), 2003.

GEORGE, E. Optimization of satellite constellations for discontinuous global coverage via genetic algorithms. In: AAS/AIAA ASTRODYNAMICS SPECIALIST CONFERENCE, 1997, Sun Valley, ID, USA. **Proceedings...** Sun Valley?: AAS, 1997. (Paper AAS 97-621).

GILES, M. B. Aerospace design: a complex task. **VKI Lecture course on inverse design.** Oxford: Oxford University Computing Laboratory. Report no. 97/07, 1997.

GILMORE, D.G. (Ed.) **Satellite thermal control handbook**. El Segundo, CA, USA: The Aerospace Corporation Press, 1994.

GLOVER, F. Heuristics for integer programming using surrogate constraints. **Decision Sciences**, v. 8, n. 1, p. 156-166, 1977.

GLOVER, F. W.; KOCHENBERG, G. A. (Eds). **Handbook of metaheuristics**. v. 57. Boston: Kluwer Academic Publishers, 570p. 2003. International Series in Operations Research & Management Science.

GLOVER, F.; LAGUNA, M.; MARTÍ, R. Scatter search. In: A. Ghosh; S. Tsutsui (Eds.). **Advances in evolutionary computation: theory and applications.** New York, NY, USA: Springer-Verlag, p. 519-537, 2003.

GOICOECHEA, A.; HANSEN, D.R.; DUCKSTEIN, L**. Multiobjective decision analysis with engineering and business applications**. New York: John Wiley and Sons, 1982.

GOLDBERG, D.E. **Genetic algorithms in search, optimization, and machine learning**. New York: Addison-Wesley Publishing, 1989.

GOLDBERG, D. E.; VOESSNER, S. Optimizing global-local search hybrids. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE (GECCO'99), 1999, San Francisco, CA, USA. **Proceedings...** New York: Morgan Kaufmann, 1999. v. 1, p. 220-228.

GOLDFARB, D. A family of variable metric methods derived by variational means. **Mathematics of Computation**, v. 24, p. 23-26, 1970.

GÓMEZ, S.; CASTILLO, N.; CASTELLANOS, L.; SOLANO, J. The parallel tunneling method. **Parallel Computing**, v. 29, p. 523-533, 2003.

GOULD, S. J.; ELDREDGE, N. Punctuated equilibrium comes of age. **Nature**, v. 366, p. 223-227, 1993.

GRAY, P.; PAINTON, L.; PHILLIPS, C.; TRAHAN, M.; WAGNER, J. A survey of global optimization methods. **Sandia National Laboratories Report**. Disponível em: <http://www.cs.sandia.gov/opt/survey/main.html>, Acesso em: 01? jan? 2004.

GROPP, W.; LUSK, E. Why are PVM and MPI so different?. In: EUROPEAN PVM/MPI USERS' GROUP MEETING, 4., 1997, Cracow, Poland. **Proceedings...** Berlin?: Springer, Recent advances in parallel virtual machine and message passing interface, Lecture notes in Computer Science, v. 1332, 1997. p. 3-10. (ISBN:3-540-63697-8)

GROPP, W.; LUSK, E.; SKJELLUM, A. **Using MPI: portable parallel programming with the message passing interface**. Cambridge, MA, USA: MIT Press, 1994.

HAJELA, P.; YOO, J. Constraint handling in genetic search using expression strategies. **AIAA Journal**, v. 34, n. 12, 1996.

HART, W. E. **Adaptative global optimization with local search.** 1994. Tese (Doutorado) - University of California, San Diego, 1994.

HOOKE, R.; JEEVES, T. A. "Direct Search" solution of numerical and statistical problems. **Journal of the ACM**, v. 8, n. 2, p. 212-221, 1961.

HOLLAND, J. H. **Adaptation in natural and artificial systems**. Nan Arbor, MI, USA: University of Michigan Press, 1974.

HOLLSTEIN, R. B. **Artificial genetic adaptation in computer control systems**. 1971. Tese (Doutorado) - Universidade de Michigan, Ann Arbor, MI, EUA,1971.

INGBER, L. Simulated annealing: practice versus theory, **Journal of Mathematical and Computer Modelling**, v. 18, n. 11, p. 29-57, 1993.

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **MAPSAR (Multi-Application Purpose SAR)** São José dos Campos, SP, Brasil: 2004-2006. Disponível em: <http://www.obt.inpe.br/mapsar/mapsarindex1.htm> Acesso em: 01? mar? 2004.

JOHNSON, E. H. Disjoint design spaces in the optimization of harmonically excited structures. **AIAA Journal**, v. 14, n. 2, p. 259-261, 1976.

JOHNSON, R. C. **Mechanical design synthesis – creative design and optimization**. 2 ed. New York, NI, USA: Robert E. Krieger Publishing Company, 1978.

JONES, D. Sem Título. Disponível em: <http://www.informatics.bangor.ac.uk/~dewi/modules/algarc/taxonomy.pdf> Acesso em: 01? mar? 2006.

JONES, B. R.; CROSSLEY, W. A.; LYRINTZIS, A. Aerodynamic and aeroacoustic optimization of rotocraft airfoils via a parallel genetic algorithm**. Journal of Aircraft**, v. 37, n. 6, p. 1088-1096, 2000.

KARAM, R. D. Satellite Thermal Control for Systems Engineers. In: **Progress in Astronautics and Aeronautics**, v. 181. Reston: 1998. American Institute of Aeronautics and Astronautics Publishing.

Kirkpatrick, S.; Gellat Jr., C.D.; Vecchi, M.P. Optimizing by simulated annealing. **Science**, Vol. 220, No. 4598, pp. 671-680, 1983.

KUGA, H. K.; ORLANDO, V. **Structure of software for orbit determination and propagation**. São José dos Campos: INPE, ago 1988a. (CCS-SWR-0024).

KUGA, H. K.; ORLANDO, V. **Structure of the software for prediction and analysis of orbit and attitude**. São José dos Campos: INPE, ago 1988b. (CCS-SWR-0023).

KUGA, H. K.; RAO, K. R.; ORLANDO, V. **Mathematical theory of the software for orbit determination and propagation**. São José dos Campos: INPE, ago 1988. (CCS-SWR-0028).

LACERDA, E. G. M.; CARVALHO, A. C. P. L. F. Introdução aos algoritmos genéticos. In: CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 19., 1999, Rio de Janeiro, RJ, Brasil. **Anais...** Rio de Janeiro: EntreLugar, 1999, v. 2, p. 51-125.

LIN, C.-Y.; HAJELA, P. Genetic algorithms in optimization problems with discrete and integer design variables. **Engineering Optimization**, v. 19, p. 309-327, 1992.

LØVBJERG, M. **Improving particle swarm optimization by hybridization of stochastic search heuristics and self-organized criticality.** 2002. Tese (Mestrado) - University of Aarhus, Aarhus, 2002.

MASON, W. J; COVERSTONE-CARROLL, V.; HARTMANN, J. W. Optimal Earth orbiting satellite constellations via a Pareto Genetic Algorithm. In: AIAA/AAS Astrodynamics Specialist Conference and Exhibit, 1998, Boston, Massachusetts. **Proceedings...** Boston: AIAA, 1998. p. 169-177. (Paper No. AIAA 98-4381)

MESSAC, A.; SUNDARARAJ, G. J.; TAPPETA, R.V.; RENAUD, J. E. Ability of objective functions to generate points on nonconvex Pareto frontiers. **AIAA Journal**, v. 38, n. 6, p. 1084-1091, 2000.

MEZA, J. C.; MARTINEZ, M. L. On the use of direct search methods for the molecular conformation problem. **Journal of Computational Chemistry**, v. 15, n. 6, p. 627-632, 1994.

MICHALEWICZ, Z. A survey of constraint handling techniques in evolutionary computation methods. In: ANNUAL CONFERENCE ON EVOLUTIONARY PROGRAMMING, 4., 1995, San Diego, CA, USA. **Proceedings...** Cambridge, MA, USA: MIT Press, p. 135-155, 1995.

MICHALEWICZ, Z.; FOGEL, D. B. **How to solve it: modern heuristics**. Berlin: Springer-Verlag publishing, 2000.

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

MIGDALAS, A.; TORALDO, G.; KUMAR, V. Nonlinear optimization and parallel computing. **Parallel Computing**, v. 29, p. 375-391, 2003.

MIETTINEN, K. Some methods for non-linear multi-objective optimization, **Lecture Notes in Computer Science: Evolutionary Multi-Criterion Optimization**, v1993, 1-20, 2001.

MORSE, P.M.; FESHBACH, H. Asymptotic Series; Method of Steepest Descent. §4.6 in **Methods of Theoretical Physics, Part I**. New York: McGraw-Hill, p. 434-443, 1953.

MPI Forum. **The Message Passing Interface (MPI) standard**. Disponível em: <http://www-unix.mcs.anl.gov/mpi/>, 2004-2006. Acesso em: 01? mar? 2004.

MURAOKA, I.; GALSKI, R. L.; SOUSA, F. L. de; RAMOS, F. M. Stochastic spacecraft thermal design optimization with low computational cost. **Journal of Spacecraft and Rockets**. (aceito para publicação, 2006)

NELDER, J. A.; MEAD, R. A simplex method for function minimization. **The Computer Journal**, v. 7, p. 308-313, 1965.

NOWOSTAWSKI, M.; POLI, R. Parallel genetic algorithms taxonomy. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE-BASED INTELLIGENT INFORMATION ENGINEERING SYSTEMS (KES'99), 3., 1999, Adelaide, Australia. **Proceedings...** Washington, DC, EUA: IEEE Computer Society, p. 88-92, 1999.

ONO, I.; KOBAYASHI, S. A real-coded Genetic Algorithm for function optimization using unimodal normal distribution crossover. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, 7., 2001, East Lansing, MI, EUA. **Proceedings...** New York, EUA: Morgan Kaufmann, p. 246-253, 2001.

OpenMP Architecture Review Board. **OpenMP**. Disponível em: <http://www.openmp.org/>. Acesso em: 01? mar? 2004.

PARDALOS, P. M.; ROMEIJN, H. E. (ed). **Handbook of global optimization**. Dordrecht: Kluwer Academic Publishers, 2001.

PARDALOS, P. M.; ROMEIJN, H. E. (ed). **Handbook of global optimization**. v. 2. Norwell, MA: Kluwer Academic Publishers, 2002.

PARETO, V. **Manuel d'economie politique**. Paris: Giard, 1909.

PACHECO, P. S. **Parallel Programming with MPI**. San Francisco, EUA: Morgan Kaufmann, 1997.

PINTÉR, J. D. Continuous global optimization: an introduction to models, solution approaches, tests and applications. **Interactive Transactions of ORMS**, v. 2, n. 2, 1999. Disponível em: <http://catt.bus.okstate.edu:80/itorms/index.html>. Acesso em: 01? mar? 2004?.

POTTER, M.; DE JONG, K. A. A cooperative coevolutionary approach to function optimization. In: PARALLEL PROBLEM SOLVING FROM NATURE, 3., 1994, Jerusalem, Israel. **Proceedings...** Berlim, Alemanha: Springer-Verlag, p. 249-257, 1994.

POWELL, M. J. D. An iterative method for finding stationary values of a function of several variables. **The Computer Journal**, v. 5, n. 2, p. 147-151, 1962.

PRETO, A. J.; MENDES, C. L. **Modelos de desempenho**. São José dos Campos: INPE, 1999a (Arquivo em powerpoint utilizado na disciplina Processamento de Alto Desempenho II, do curso de pós-graduação em Computação Aplicada do INPE).

PRETO, A. J.; MENDES, C. L. **Multicomputadores**. São José dos Campos: INPE, 1999b (Arquivo em powerpoint utilizado na disciplina Processamento de Alto Desempenho II, do curso de pós-graduação em Computação Aplicada do INPE).

PRETO, A. J.; MENDES, C. L. **Troca de Mensagens MPI**. São José dos Campos: INPE, 1999c (Arquivo em powerpoint utilizado na disciplina Processamento de Alto Desempenho II, do curso de pós-graduação em Computação Aplicada do INPE).

PRETO, A. J.; STEPHANY, S. **Introdução ao PAD**. São José dos Campos: INPE, 2002 (Apostila da disciplina Processamento de Alto Desempenho I, do curso de pós-graduação em Computação Aplicada do INPE).

PREUX, P.; TALBI, E. G. Towards hybrid evolutionary algorithms. **Intl. Trans. In Op. Res.**, v. 6, p. 557-570, 1999.

PROOS, K. A.; STEVEN, G. P.; QUERIN, O. M.; XIE, Y. M. Multicriterion evolutionary structural optimization using the weighting and the global criterion methods. **AIAA Journal**, v. 39, n. 10, p. 2006-2012, 2001.

RALPHS, T. K. Parallel branch and cut for capacitated vehicle routing. **Parallel Computing**, v. 29, p. 607-629, 2003.

REINERS, T.; VOß, S. Teaching meta-heuristics within virtual learning environments. **International Transactions in Operational Research**, v. 11, p. 225-238, 2003.

REKLAITIS, G. V.; RAVINDRAN, A.; RAGSDELL, K. M**. Engineering optimization** – Methods and applications**.** New York: John Wiley and Sons, 1983.

ROSENBROCK, H. H. An automatic method for finding the greatest or least value of a function. **The Computer Journal**, v. 3, p. 175-184, 1960.

SCHNABEL, R. B. A view of the limitations, opportunities, and challenges in parallel nonlinear optimization. **Parallel Computing**, v. 21, n. 6, p. 875-905, 1995.

SCHRÖDER, R.; PULS, J.; HAJNSEK, I.; JOCHIM, F.; NEFF, T.; KONO, J.; PARADELLA, W. R.; SILVA, M. M. Q.; VALERIANO, D. M.; COSTA, M. P. F. MAPSAR: A small L-band SAR mission for land observation. **Acta Astronautica Journal**, v. 56, n. 1-2, 2005, p. 35-43.

SEBAH, P.; GOURDON, X. **Newton's method and high order iterations**. Disponível em: <numbers.computation.free.fr/Constants/ Algorithms/newton.html>. Acesso em: 15? out? 2001.

SHANNO, D. F. Conditioning of quasi-newton methods for function minimization. **Mathematics of Computation**, v. 24, p. 647-656, 1970.

SHAPOUR, A. **Multiobjective optimal design**. College Park: University of Maryland. Disponível em: <http://www.glue.umd.edu/~azarm/optimum_notes/multi/multi.html>. Acesso em: 1996.

SNIR, M.; OTTO, S.; HUSS-LEDERMAN, S.; WALKER, D.; DONGARRA, J. **MPI: the complete reference**. Cambridge, MA, EUA: MIT Press, 1995.

SOMEYA H.; YAMAMURA, M. Genetic Algorithm with search area adaptation for the function optimization and its experimental analysis. In: 2001 Congress on Evolutionary Computation, 2001, Seul, Coréia do Sul. **Proceedings...** Seul: IEEE Transactions on Evolutionary Computation, v.2, p. 933-940, 2001.

SOUSA, F. L. de **Otimização extrema generalizada: um novo algoritmo estocástico para o projeto ótimo.** 2002. 142p. (INPE-9564-TDI/836). Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2002.

SOUSA, F. L. de; RAMOS, F. M. Function optimization using extremal dynamics. In: INTERNATIONAL CONFERENCE ON INVERSE PROBLEMS IN ENGINEERING, 4., 2002, Angra dos Reis, Rio de Janeiro, Brasil. **Proceedings...** New York: Engineering Conferences International (cd-rom), 2002.

SOUSA, F. L. de; TAKAHASHI, W. K. Discrete optimal design of trusses by generalized extremal optimization. In: WORLD CONGRESS IN STRUCTURAL AND MULTIDISCIPLINARY OPTIMIZATION, 6., 2005, Rio de Janeiro, Brasil. **Proceedings...** Rio de Janeiro: International Society for Structural and Multidisciplinary Optimization (ISSMO), 2005a. (ISBN: 85 – 285 – 0070 - 5)

SOUSA, F. L. de; TAKAHASHI, W. K. Generalized extremal optimization applied to three-dimensional truss design. In: INTERNATIONAL CONGRESS OF MECHANICAL ENGINEERING (COBEM2005), 18., 2005, Ouro Preto, Brasil. **Proceedings...** Ouro Preto: ABCM, 2005b.

SOUSA, F. L. de; VLASSOV, V.; RAMOS, F. M. Generalized Extremal Optimization for Solving Complex Optimal Design Problems. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE (GECCO 2003), 2003, Chicago, IL, USA. **Proceedings...** New York?: Springer, Lecture Notes on Computer Science, v. 2723, 2003a. p. 375-376. (ISBN-10: 3540406026) (ISBN-13: 978-3540406020).

SOUSA, F. L. de; VLASSOV, V.; RAMOS, F. M. Heat pipe design through generalized extremal optimization.In: BRAZILIAN CONGRESS OF ENGINEERING AND THERMAL SCIENCES (ENCIT 2002), 9., Caxambu, MG, Brazil, 2002. **Proceedings...** Caxambu?: ABCM?, 2002a.

SOUSA, F. L. de; VLASSOV, V.; RAMOS, F. M. Heat pipe design through generalized extremal optimization. **Heat Transfer Engineering,** v. 25, n. 7, p. 34-45, 2004a. (Selected papers from the 9th Brazilian Congress of Thermal Sciences and Engineering) (ISSN 0145-7632).

SOUSA, F. L. de; VLASSOV, V.; RAMOS, F. M. Generalized extremal optimization: an application in heat pipe design. **Applied Mathematical Modeling**, v. 28, n. 1?, p. 911-931, 2004b. (ISSN 0307-904X).

SOUSA, F. L. de; RAMOS, F. M; PAGLIONE, P.; GIRARDI, R. M. Laminar profile design using extremal dynamics optimization. In: WORLD CONGRESS ON COMPUTATIONAL MECHANICS (WCCM V), 5., Vienna, Austria, 2002. **Proceedings...** Viena?: Book of Abstracts, v. I, p. I-311, 2002b.

SOUSA, F. L. de; RAMOS, F. M.; PAGLIONE, P.; GIRARDI, R. M. New Stochastic Algorithm for Design Optimization, **AIAA Journal**, v. 41, n. 9, p. 1808-1818, 2003b.

SOUSA, F. L. de; RAMOS, F. M.; PAGLIONE, P.; GIRARDI, R. M. Algoritmo da otimização extrema generalizada aplicado à aerodinâmica de planadores. In: CONGRESSO TEMÁTICO DE APLICAÇÕES DE DINÂMICA E CONTROLE (DINCON 2003) DA SOCIEDADE BRASILEIRA DE MATEMÁTICA APLICADA E COMPUTACIONAL (SBMAC), 2., Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, Brasil. **Anais...** (em CDROM) São José dos Campos: ITA?, p. 133-148, 2003c.

SOUSA, F. L. de; RAMOS, F. M.; GALSKI, R. L.; MURAOKA, I. Generalized Extremal Optimization: A New Meta-heuristic Inspired by a Model of Natural Evolution. In: De Castro, L. N. & Von Zuben, F. J. (Eds.). **Recent developments in biologically inspired computing**, Hershey, PA, USA: Idea Group Inc., Cap. 3, p. 41-60, 2005a. (ISBN 1-59140-312-X).

SOUSA, F. L. de, RAMOS, F. M.; SOEIRO; F. J. C. P.; SILVA NETO; A. J. Application of the generalized extremal optimization algorithm to an inverse radiative transfer problem. In: INTERNATIONAL CONFERENCE ON INVERSE PROBLEMS IN ENGINEERING: THEORY AND PRACTICE, 5., Cambridge, UK, 2005. **Proceedings...** Cambridge: D. Lesnic (Ed.), 2005b. v. 1, D03.

SOUSA, F. L. de, RAMOS, F. M., VLASSOV, V., PAGLIONE, P.; GIRARDI, R. M. Generalized extremal optimization: an overview of a new evolutionary optimum design tool. In: BIANNUAL BRAZILIAN SYMPOSIUM ON ARTIFICIAL NEURAL NETWORKS (SBRN 2004) (in CDROM), 8., 2004, São Luis, Maranhão, Brazil. **Proceedings...** São Luis?: IEEE Computer Society, 2004c. (ISBN 85-89029-04-2).

SPENDLEY, W.; HEXT, G. R.; HIMSWORTH, F. R. Sequential application of simplex designs in optimisation and evolutionary operation. **Technometrics**, v. 4, p. 441, 1962.

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

TALBI, E. G. A taxonomy of hybrid metaheuristics, **Journal of Heuristics**, v. 8, p.541-564, 2002.

TAPPETA, R. V.; RENAUD, J. E.; MESSAC, A.; SUNDARARAJ, G. J. Interactive physical programming: tradeoff analysis and decision making in multicriteria optimization, **AIAA Journal**, v. 38, n. 5, p. 917-926, 2000.

TOULOUSE, M.; CRAINIC, T. G.; THULASIRAMAN, K. Global optimization properties of parallel cooperative search algorithms: a simulation study. **Parallel Computing**, v. 26, p. 91-112, 2000.

VANDERPLAATS, G. N. **Numerical optimization techniques for engineering design**. 2 ed. Colorado Springs: Vanderplaats Research & Development, 1998.

VAN VELDHUIZEN, A.; LAMONT, G. B. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art, **Evolutionary Computation**, v. 8, n. 2, p.125-147, 2000.

VICINI, A.; QUAGLIARELLA, D. Airfoil and wing design through hybrid optimization strategies. **AIAA Journal**, v. 37, n. 5, p. 634-641, 1999.

VLASSOV, V. V.; SOUSA, F. L. de; TAKAHASHI, W. K. Comprehensive optimization of a heat pipe radiator assembly filled with ammonia or acetone. **International Journal of Heat and Mass Transfer**, v. 46, n. 2, p. 4584-4595, 2006.

WALKER, J. G. Some circular orbits patterns providing continuous whole-Earth coverage. J**ournal of the British Interplanetary Society**, v. 24, p. 369-384, 1971.

WANG, X.; DAMODARAN, M. Aerodynamic Shape Optimization using computational fluid dynamics and parallel simulated annealing algorithms. **AIAA Journal**, v. 39, n. 8, p. 1500-1508, Aug 2001.

WANG, X.; DAMODARAN, M. Optimal three-dimensional nozzle shape design using CFD and parallel simulated annealing. **Journal of Propulsion and Power**, v. 18, n. 1, p.217-221, 2002.

WANG, X.; DAMODARAN, M.; LEE, S. L. Inverse transonic airfoil design using parallel simulated annealing and computational fluid dynamics. **AIAA Journal**, v. 40, n. 4, p. 791-794, Apr 2002.

WIELD, D. J. **Globally optimal design**. New York: John Wiley & Sons, 1978.

WILLIAMS, E. A.; CROSSLEY, W. A.; AND LANG, T. J. Average and maximum revisit time trade studies for satellite constellations using a multiobjective genetic algorithm. **The Journal of the Astronautical Sciences**, v. 49, n. 3, p. 385-400, July-Sept 2001.

WOLPERT, D. H.; MACREADY, W. G. **No free lunch theorems for search**. Santa Fe, NM, USA: Santa Fe Institute. Technical Report SFI-TR-95-02-010, 1995.

XU, P. A hybrid global optimization method: The multi-dimensional case. **Journal of Computational and Applied Mathematics**, v. 155, n. 2, p. 423-446, 2003.

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programa de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. São aceitos tanto programas fonte quanto executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.