# OTIMIZAÇÃO PARAMÉTRICA COM COMPUTAÇÃO EVOLUTIVA

SÉRGIO LUCIANO ÁVILA

1° EDIÇÃO

PUBLICAÇÃO DO IFSC

GitHub

creative commons

PECCE

Reitor pro tempore
*André Dala Possa*

Diretor do Campus Florianópolis
*Zízimo Moreira Filho*

Chefe do Departamento Acadêmico de Eletrotécnica
*James Silveira*

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DE SANTA CATARINA – IFSC – CAMPUS FLORIANÓPOLIS
AV. MAURO RAMOS N. 950, FLORIANÓPOLIS, SC.

Sérgio Luciano Ávila

# Otimização Paramétrica com Computação Evolutiva


Florianópolis
2020

Revisão e diagramação: Fabiano Avila, jornalista.

Catalogação na fonte pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina – IFSC

A958o Ávila, Sérgio Luciano

Otimização paramétrica com computação evolutiva. [recurso eletrônico] / Sérgio Luciano Ávila. - Florianópolis: Publicações do IFSC, 2020.

192 p. : il. color.

Inclui índice remissivo.

ISBN 978-65-88663-00-4

1. Otimização estrutural. 2. Otimização paramétrica. 3. Computação evolutiva. 4. Algoritmos genéticos. 5. Análise de sensibilidade. 6. Inteligência artificial. I. Título

CDD 621.3

Elaborada pela Bibliotecária Renata Ivone Garcia – CRB-14/1417

GITHUB

HTTPS://GITHUB.COM/PECCE-IFSC/OTIMIZAÇÃO

TODOS OS ALGORITMOS ESTÃO DISPONIBILIZADOS

MATLAB ©

LICENÇA R2019A 40822288


GRUPO DE PESQUISAS EM COMPUTAÇÃO CIENTÍFICA PARA ENGENHARIA

SITES.FLORIANOPOLIS.IFSC.EDU.BR/PECCE

Sergio.Avila@ifsc.edu.br

Eletrotécnico (ETFSC-1994) e Engenheiro Industrial Eletricista (FURB-2000). Possui mestrado pela UFSC (2002) e doutorado com duplo-diploma pela Ecole Centrale de Lyon/França e UFSC (2006), ambos em engenharia elétrica e versando sobre otimização de dispositivos e sistemas. Pós-doutoramento industrial na Schneider Electric (INPG - França, 2006). Pós-doutoramento na Escola Politécnica da USP (2008). Desde 2010 é Professor no Instituto Federal de Santa Catarina (IFSC). Líder do Grupo de Pesquisas em Computação Científica para Engenharia (PECCE), que envolve pesquisadores do IFSC e empresas da Associação Catarinense de Empresas de Tecnologia (ACATE). Atua também como consultor ad-hoc para o MEC e a FAPESC. Atualmente, trabalha no desenvolvimento de soluções para o monitoramento e prognósticos de comportamento em máquinas e sistemas elétricos, o que envolve instrumentação, modelagem numérica, otimização e algoritmos de inteligência artificial. A intenção é transformar dados em informação útil para tomadas de decisão mais assertivas.

lattes.cnpq.br/7864845374871073

"Naturalmente deves trabalhar de maneira a não atentar contra a própria consciência."

Umberto Eco

Agradecimentos,

aos colegas do IFSC.
ao PECCE.
a minha família, obrigado pela paciência.
ei, tem onda. toca!

# Otimização paramétrica com computação evolutiva

| | | |
|---|---|---|
| Área de concentração | : | Engenharias |
| Público | : | graduandos e pós-graduandos |
| Palavras-chave | : | otimização paramétrica, escalar e vetorial, computação evolutiva, algoritmos genéticos, análise sensibilidade, inteligência artificial. |

Otimizar significa buscar a melhor solução para um problema. Assim, antes de discutir ferramentas para tal, deve-se pensar sobre a formulação do problema em questão e as suas características de comportamento.

É comum encontrar literatura livros sobre 'programação linear', para resolver problemas equacionados por sistemas lineares e/ou sobre 'programação não linear', para sistemas não lineares. Obviamente, esta definição implica num conhecimento prévio do problema.

Há alguns anos surgiu a 'inteligência computacional'. Ela consiste de conceitos, paradigmas, algoritmos e implementações de sistemas que em teoria exibem comportamentos inteligentes em ambientes complexos. Essas técnicas tentam minimizar a dependência da experiência do engenheiro na resolução dos problemas.

Dentro da inteligência computacional, tem-se a computação evolucionária (CE). Trata-se de métodos de otimização inspirados na evolução biológica. São estratégias heurísticas ou meta-heurísticas baseadas em probabilidades.

Notadamente, as técnicas de CE pouco ou nada dependem da natureza do problema, como não linearidades, descontinuidades, não convexidades, multimodalidades etc. Também, não utilizam a informação da derivada das equações que regem o problema a ser resolvido. Esta independência pode ser considerada vantagem em problemas complexos. O esforço computacional e as incertezas inerentes a um processo probabilístico podem ser considerados desvantagens.

Este livro é organizado em capítulos de forma a apresentar ao leitor um crescente entendimento do seu próprio problema e das ferramentas mais eficientes para a sua resolução.

Reforça-se que o propósito maior deste livro é abrir portas, nunca esgotar assuntos. Para obter maior detalhamento dos métodos bem como suas validações, sugere-se a busca pelas obras específicas de cada tema.

# *Sobre otimização*

Engenharia é a geração e a aplicação do conhecimento para inventar ou melhorar soluções que promovam avanços na qualidade de vida. Portanto, o engenheiro pode ser definido como aquele que conjuga conhecimentos multidisciplinares e os aplica tendo em conta a sociedade, o meio ambiente, a técnica e a economia.

Otimizar significa buscar a melhor solução para um problema. A intenção é maximizar ou minimizar um ou mais objetivos, como por exemplo a eficiência, o custo... respeitando restrições como orçamentos, especificações, leis... por meio do ajuste de características (parâmetros) que compõem uma possível solução, como por exemplo dimensões, tipo de materiais...

A determinação do ótimo – a melhor solução – depende da possibilidade de quantificar o grau de adequação da solução à necessidade em causa. Também, métodos de otimização buscam a melhor resposta para a pergunta que foi formulada. Questões pertinentes:

- Como bem representar na forma matemática um fenômeno físico ou problema a resolver?
- Como definir os parâmetros a serem ajustados sob quais restrições e na busca por quais objetivos?
- Qual a precisão desejada?

A Figura 1.1 ilustra estas questões. Para bem as responder, deve-se ter uma compreensão adequada do problema em discussão. Nas últimas décadas, o processo pela busca da melhor solução possível tornou-se mais simples com a ajuda dos computadores.

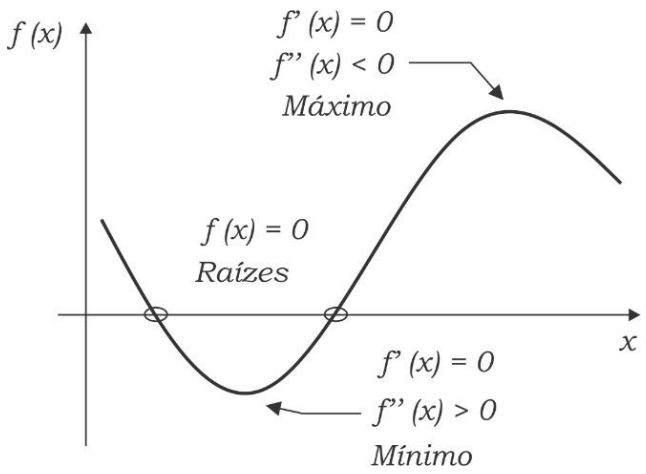



Figura 1.1 Otimização.

## 1.1 Resolvendo problemas da engenharia com computador

Engenheiros constroem suas hipóteses tendo por início os princípios de conservação da natureza, os quais levam a equações de continuidade e de balanço [1]. O que o engenheiro faz é observar e entender um fenômeno ou problema, traduzi-lo numa modelagem matemática coerente, aplicar técnicas numéricas eficientes para resolvê-lo e, por fim, validar as soluções encontradas. Caso as soluções não sejam adequadas, volta-se à definição do problema. A Figura 1.2 ilustra esse processo iterativo de desenvolvimento da solução.

Cálculo numérico, métodos numéricos, engenharia assistida por computador, projeto assistido por computador, *computer aided engineering* (CAE), *computer aided design* (CAD), matemática computacional, computação científica... estes são alguns dos nomes utilizados para designar o campo de estudo interessado na construção de modelos matemáticos e de técnicas de soluções numéricas utilizando computadores para analisar e resolver problemas de engenharia [2][3].


Figura 1.2. Modelagem numérica [2].

Quando a resolução analítica é possível, ótimo. Entretanto, quanto mais próximo se deseja a modelagem de um problema real, mais complexo será o seu entendimento e, consequentemente, é provável que mais complexa seja a sua formulação matemática.

Utiliza-se a modelagem e a simulação computacional quando o sistema real é de difícil entendimento na sua totalidade. Outros aspectos que as motivam são a redução de custos para realização de ensaios ou de provas, bem como a necessidade de repetibilidade de resultados ou o estudo de situações críticas.

A observação de situações aguça a busca por conhecimento para as entender. Experimentos, testes e busca por teorias existentes podem ser feitos para aumentar o conhecimento prévio. A experiência do profissional acerca do tema é de grande valia neste momento inicial.

A análise estatística (*numerical design of experiments* – DoE) ajuda a definir uma sequência ordenada de ações na definição do problema [4]. Além disto, procedimentos de análise de sensibilidade podem fornecer uma compreensão abrangente das influências dos diversos parâmetros e suas consequências nos resultados do modelo [5].

Nesta fase inicial, 'problema em estudo' conforme Figura 1.2, os objetivos são avaliar a influência dos parâmetros e quantificar a incerteza do modelo. Com base nos resultados das ferramentas do DoE e de análise de sensibilidade, um modelo reduzido com um conjunto menor de parâmetros significativos pode ser obtido. Isso pode implicar em modelos numéricos mais simples. E, em assim sendo, a dificuldade de resolução pode diminuir.

Uma vez que o problema está claro, a sua representação física e matemática pode ser definida. Desta forma, novas escolhas devem ser feitas. Equações algébricas podem ser lineares ou não lineares. Equações diferenciais podem ser ordinárias ou parciais. Equações diferenciais podem ser convertidas em equações integrais equivalentes. A escolha por uma formulação é um equilíbrio entre a abstração física e a correção da solução [2].

Outras tomadas de decisão devem ser feitas entre a definição do problema e a validação da solução. Cada escolha pode levar a soluções mais ou menos precisas, com maior ou menor esforço computacional.

Outra discussão pertinente é a escolha da linguagem de programação a ser utilizada. É cada vez mais comum encontrar softwares comerciais para simulação. Eles são úteis e geralmente fornecem muitas ferramentas de pós-processamento que ajudam a visualizar os resultados.

Por outro lado, uma desvantagem no uso de software comercial é a falta de controle sobre todas as variáveis que um engenheiro deve abordar durante uma investigação. Assim, o desenvolvimento de programação própria é válido.

Pascal, Fortran e C, entre outras linguagens de programação exigem um grande conhecimento em programação estruturada [6]. Este foi o paradigma dominante na criação de software até a programação orientada a objetos, sendo as mais populares C ++ e Visual C ++.

Recentemente, surgiu a linguagem interpretada. Ela utiliza simplificações na maneira de se escrever um código. Muitas vezes isto resulta em um formalismo pobre e no uso de bibliotecas prontas, gerando dependência similar aos softwares comerciais. Como enorme vantagem, a linguagem interpretada permite a(o) engenheira(o) testar suas ideias sem a necessidade de dedicar um tempo considerável com a rigidez da programação estruturada. Matlab [7] e Python [8] são exemplos de linguagens interpretadas.

Além disso, o uso de bibliotecas *open source* também está se tornando uma opção popular. Pode-se encontrar muitos exemplos de ferramentas de código aberto. No entanto, até mesmo um código bem escrito pode conter erros.

Como dito, métodos de otimização buscam a melhor resposta para a pergunta que foi formulada. Entender o problema físico, traduzi-lo numa modelagem matemática adequada e sobre ela buscar a melhor solução não é trivial. Ainda, outras ferramentas podem ser usadas para entender melhor a solução numérica obtida. Cita-se como exemplo a regressão e a interpolação. Entender o comportamento do problema, não apenas o valor bruto da solução final, também pode ser possível em muitos casos.

Utilizada em uma ampla gama de situações, técnicas de aprendizado de máquina ou de forma genérica, a chamada inteligência artificial, estão sendo aplicadas para a modelagem numérica e a sua solução. O uso dessas técnicas é um marco importante, a saber, como interpretar dados sem assistência humana [9].

## 1.2 Sobre aproximações e erros

Exatidão pode ser definida como o quão próximo os valores calculados ou medidos estão próximos do verdadeiro. Importante não confundir com precisão. Precisão é o quão próximo os valores individuais medidos ou calculados estão uns dos outros.

Como discutido antes, a primeira questão é saber até que ponto o modelo matemático bem representa a situação em estudo. Deve-se ter conhecimentos relativos às etapas de modelagem suficientes para, pelo menos, ter ciência de qual o grau de exatidão que se pode conseguir e, consequentemente, qual o erro tolerável.

Associam-se a isto os chamados erros numéricos. Erros numéricos são aqueles causados pelo uso de aproximações para representar quantidades matemáticas exatas. Eles podem ser de dois tipos: erros de arredondamento e de truncamento.

Erros de arredondamento surgem porque os computadores têm limites de tamanho e precisão em sua capacidade de representar números. Em alguns casos, esses erros podem direcionar os cálculos para instabilidades numéricas, fornecendo resultados incorretos ou mal condicionados. Erros de arredondamento podem levar a discrepâncias sutis difíceis de detectar.

Erros de truncamento resultam do uso de uma aproximação numérica no lugar de um procedimento matemático exato, geralmente uma tomada de decisão do engenheiro. Por exemplo, um número máximo de repetições em um processo iterativo.

Além disso, na ciência da computação, a eficiência pode ser uma característica relacionada ao número de recursos computacionais utilizados pelo algoritmo, como o tempo, o espaço e a memória [6].

Assim, na análise numérica, os erros de discretização são aqueles que resultam do fato de que uma função de uma variável contínua é representada por um número finito de avaliações. De forma geral, os erros de discretização podem ser reduzidos usando um número maior de avaliações. Logo, tem-se o aumento do custo computacional.

Em resumo, a metodologia para desenvolver uma representação numérica exige uma grandeza de detalhes e cuidados proporcionais à complexidade da situação em estudo. Simplificações, aproximações e erros são comuns. Entender tais limitações é fundamental para se obter e julgar soluções como adequadas.

## 1.3 Terminologia da otimização

A Figura 1.2 ilustra os passos para a de análise do problema. Entendido o problema, pode-se buscar a melhor solução para ele. A Figura 1.3 apresenta uma metodologia de otimização.

De forma geral, os métodos de otimização são desenvolvidos separados da análise do problema. Entretanto, em alguns casos, eles podem estar intricados a essa análise, conforme explicado na próxima seção. Em ambos os casos, o processo iterativo está sujeito a declarações e condições que conduzirão a uma determinada solução. Ainda, como qualquer processo iterativo demanda, um número máximo de iterações ou qualquer outro critério de parada deve ser verificado. Não se pode permitir um esforço computacional sem fim. Portanto, a solução final obtida será dita ótima de acordo com as declarações e condições impostas, bem como conforme a qualidade na análise do problema que se tem.

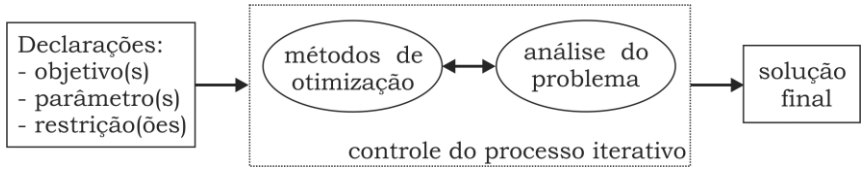



Figura 1.3. Metodologia de otimização.

Algumas definições básicas para otimização são:

- Funções objetivo – equações matemáticas que representam o que se deseja melhorar (função de mérito ou *fitness function*). Multiobjetivo ou otimização vetorial pressupõe duas ou mais funções de mérito. Mono-objetivo ou otimização escalar significa uma função de mérito, que pode sim possuir vários objetivos ponderados. A formulação matemática para otimização escalar e vetorial será apresentada no capítulo 2. Ver também o capítulo 3, que versa sobre o comportamento do problema a ser resolvido;
- Parâmetros – são os aspectos do problema que podem ser ajustados visando obter a(s) solução(ões) ótima(s). Podem ser chamados de variáveis de otimização, variáveis objeto, variáveis de concepção ou de projeto (*design variables*);
- Espaço de busca – domínio (delimitado ou não) que contém os valores dos parâmetros. Corresponde ao espaço de soluções. A dimensão do espaço de busca é definida pelo número de parâmetros envolvidos nas soluções. Se cada solução é formada por três parâmetros, o espaço de busca é tridimensional;
- Espaço de objetivos – conjunto imagem do espaço de busca determinado por todos os valores possíveis das funções objetivo. A dimensionalidade deste espaço depende do número de objetivos, similar ao espaço de busca;
- Restrições – especificações do problema que delimitam os espaços de parâmetros (restrições construtivas, normas etc.) e/ou que não permitem determinada faixa de valores nos objetivos. Por exemplo, requisitos de projeto podem impor que abaixo de certo valor a solução não seja considerada;
- Domínio realizável – região do espaço (dos parâmetros e/ou objetivos) onde as restrições são respeitadas;
- Domínio não-viável – região do espaço onde as restrições são violadas.

- Minimização e maximização – o processo de busca pela solução ótima significa a busca pelo valor mínimo ou máximo de uma ou mais funções. Se o desejo é um motor elétrico com mais torque, usa-se a maximização. Se o desejo é um motor elétrico mais barato, usa-se a minimização. Processos de minimização podem ser transformados em maximização, e vice-versa, dependendo da maneira como é escrita a função de mérito.

A Figura 1.4 apresenta um problema com duas variáveis e dois objetivos, portanto ambos espaços bidimensionais. Quanto mais dimensões estes espaços possuem, mais complexo é o problema de otimização. Eles estão sujeitos a duas restrições ($g_1$ e $g_2$) sobre os parâmetros e a uma restrição sobre os objetivos ($e_1$). Aqui, tem-se os ótimos das funções não em termos de maximização ou minimização, mas por uma região no espaço dos objetivos.

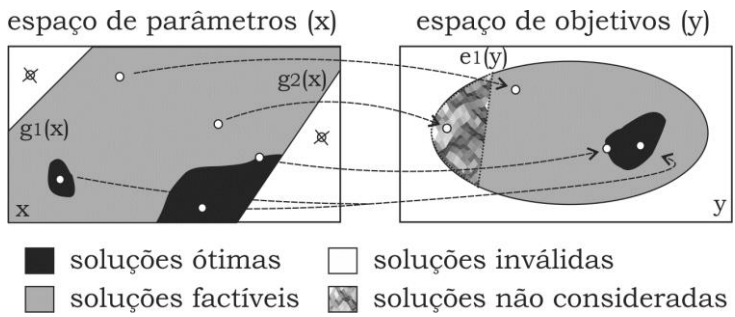



Figura 1.4. Relações entre os diferentes espaços de um problema.

Esta representação permite notar que:

- A correspondência de uma solução de x (espaço de parâmetros) em y (espaço de objetivos) nem sempre é possível, notadamente para as soluções inválidas;

- Mesmo as soluções que atendem às restrições impostas aos parâmetros estão também sujeitas às exigências impostas aos objetivos;

- Duas soluções muito distintas (ou diferentes) no espaço de parâmetros podem corresponder a pontos próximos no espaço de objetivos (chama-se problema multimodal). O contrário também é possível: duas soluções próximas no espaço de parâmetros podem gerar pontos distantes no espaço de objetivos (descontinuidades ou região muito 'sensível'). A matemática dessas características é apresentada no capítulo 2.

Os mecanismos para a exploração do espaço de busca, específicos a cada metodologia de otimização, são condicionados por parâmetros de controle (números de iterações, direção de procura, verificação de convergência, etc.) e por condições iniciais (valores iniciais dos parâmetros, limites dos domínios, etc.).

Neste contexto, para bem formular o problema e buscar a sua melhor solução, faz-se necessário aliar a experiência do projetista, aquele que entende do problema, de forma a bem especificar os apontamentos acima; com a escolha da metodologia de otimização mais adequada.

## 1.4 Metodologias para otimização

De forma geral, existem quatro áreas de desenvolvimento de metodologias de otimização:

- **Pesquisa operacional** é aplicada a problemas que compreendem a condução de operações e atividades. Exemplos de algoritmos são as árvores de decisão, teoria de jogos, teoria de filas e processos de decisão de Markov [10]. Os resultados são utilizados como indicadores para tomadas de decisão ou linha de conduta.

Exemplo: A comercialização de energia elétrica no Brasil pode ser feita por leilão. Num leilão, assim como na teoria dos jogos, existe os agentes concorrentes (não cooperativos) que lidam com informações incompletas (não se sabe a proposta do concorrente). A teoria de jogos analisa as informações e as expectativas de um agente e informa qual a melhor estratégia correspondente àquilo que ele acredita e sabe [11].

- **Metodologias específicas de projeto**, como a teoria de controle ótimo e programação dinâmica [12]. São mecanismos de otimização indivisíveis da modelagem do processo. Assim, algoritmos especializados para cada caso.

  Exemplo: a teoria de controle ótimo para circuitos elétricos e eletrônicos trata do comportamento de sistemas dinâmicos. A saída desejada de um sistema é chamada de referência. Quando uma ou mais variáveis de saída necessitam seguir uma certa referência ao longo do tempo, um controlador manipula as entradas do sistema para obter o efeito desejado nas saídas deste sistema. Portanto, uma programação dinâmica deve ser feita [13].

- **Otimização topológica** é uma metodologia que tem a capacidade de criar formas para uma estrutura [14]. Para tal, um conjunto de variáveis de projeto descrevem a presença ou ausência de material dentro de volumes. Isto é definido dentro de cada elemento de uma malha. Buracos na estrutura podem aparecer, desaparecer e mesclar, assim como os limites podem assumir formas arbitrárias. Portanto, as variáveis de concepção (os parâmetros) são definidas de forma autônoma e ligados à discretização. A Figura 1.5 ilustra a otimização topológica.

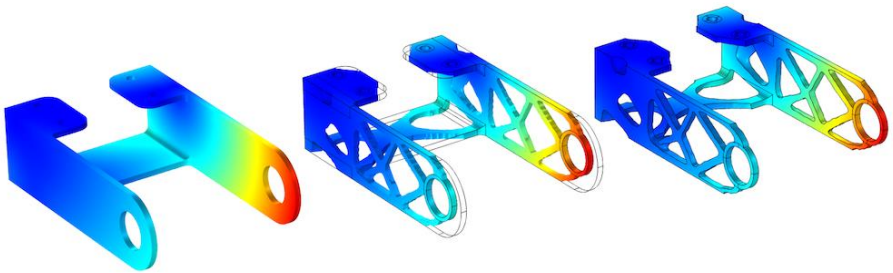


Figura 1.5. Exemplo de otimização topológica [15].

- **Otimização paramétrica** tem como pressuposto a definição de quais são as variáveis para a busca do ótimo. O problema é resolvido em função dos parâmetros. A definição de quem são as variáveis e quais os seus limites depende da experiência do projetista.

Exemplo: otimização de uma antena Yagi-Uda. O tipo da antena foi definido previamente. Portanto, não se trata de otimização topológica, mas sim paramétrica: altera-se as dimensões (comprimentos, raios e distancias entre os elementos) na busca de um comportamento eletromagnético com determinadas características.

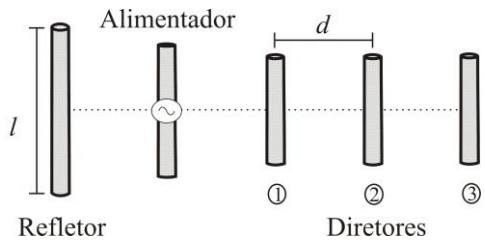



Figura 1.6. Exemplo de otimização paramétrica.

As quatro metodologias são empregadas à engenharia. Cada uma com suas vantagens e desvantagens predominantemente em função do tipo de problema que ser quer resolver.

## 1.5 Escopo deste livro

Este livro é dedicado a **otimização paramétrica com computação evolutiva.** De forma geral, otimização paramétrica é dívida pela literatura em duas grandes áreas: programação linear e programação não-linear.

- **Programação linear:** é uma família de métodos que trata problemas de optimização nos quais a função objetivo e as restrições são todas lineares. O método mais popular talvez seja o Simplex [16].

O uso mais comum do Simplex é para se maximizar um resultado, ou seja, encontrar o maior valor possível para um total. Problemas típicos para se resolver com o Simplex são os que buscam quantidades ideais de produtos a serem comercializados, com restrições referentes ao armazenamento e à fabricação dos mesmos. As restrições são apresentadas como inequações. Elas indicam peculiaridades como o fato de uma empresa só conseguir armazenar uma quantidade determinada de produtos, por exemplo. Dentre as possibilidades de valores para as variáveis que atendam às restrições, o algoritmo deve encontrar aqueles que dão à função objetivo o maior total possível.

- **Programação não-linear:** é uma família de métodos que trata problemas de optimização de características não-lineares, seja da função objetivo, das restrições ou de ambas. De forma geral, eles podem ser subdivididos em dois grupos: aqueles que tem seu processo de busca pelo ótimo baseado na informação do gradiente da função objetivo [17]; e aqueles que são baseados em processos probabilísticos [18].

O Capítulo 3 debate o comportamento matemático do problema a ser resolvido. É este comportamento ou natureza que deve orientar a escolha da metodologia de otimização mais apropriada.

De toda forma, é correto afirmar que a imensa maioria dos problemas da engenharia tem natureza não-linear. Quanto mais próximo a formulação está da vida real, com aspectos multifísicos por exemplo, maior a probabilidade de o problema ser não-linear.

Um exemplo simples é o resistor elétrico. Se a temperatura nele não varia, ele é dito ôhmico – o valor da sua resistência é constante. Entretanto, com a circulação de corrente elétrica por ele, certamente ele aquecerá, variando assim o seu valor resistivo. Isto implica numa análise não-linear de um circuito eletrônico, se o desejo for um estudo preciso.

Resolver problemas não-lineares com métodos lineares dificilmente produzirá resultados satisfatórios. O contrário é possível, mas não com tanta eficiência.

Os métodos de programação não-linear clássicos têm sua busca pelo ótimo orientada pelo gradiente das funções que representam o problema. A obtenção do gradiente, ou aproximações deste, em problemas complexos pode ser uma tarefa árdua ou mesmo impraticável. Com o avanço do poder computacional, surgiram os métodos probabilísticos ou estocásticos, que usam a própria figura de mérito da(s) função(ões) objetivo(s) como estratégia para caminhar na direção do ótimo. Ambos possuem vantagens e desvantagens. O capítulo 3 discute o tema.

Há alguns anos surgiu a denominação 'inteligência computacional'. São conceitos, paradigmas, algoritmos e implementações de sistemas que em teoria exibem comportamentos inteligentes em ambientes complexos [19]. Essas técnicas tentam minimizar a dependência da experiência do engenheiro na resolução dos problemas, permitindo relevar a tomada de decisão sobre usar programação linear ou não-linear, por exemplo.

Dentro da inteligência computacional, tem-se a computação evolucionária (CE). Trata-se de métodos de otimização inspirados na evolução natural [18]. São estratégias baseadas em probabilidades.

Notadamente, as técnicas de CE pouco ou nada dependem da natureza do problema, como não linearidades, descontinuidades, não convexidades, multimodalidades etc. (ver capítulo 2 e 3). Também, não utilizam a informação da derivada das equações que regem o problema a ser resolvido. Esta independência pode ser considerada vantagem em problemas complexos. O esforço computacional e as incertezas inerentes a um processo probabilístico podem ser considerados desvantagens.

Os capítulos 2 e 3 tem por objetivo discutir as definições matemáticas de um problema de otimização. Dentre as muitas técnicas de CE, apresentadas no capítulo 4, este livro é consagrado aos Algoritmos Genéticos, visto em detalhes nos capítulos 5 a 8.

**Referências**

[1] J. Jewett and R. Serway, Física para cientistas e engenheiros, Ed. Cegange Learning, 2011. ISBN 9788522111107.

[2] S. L. Avila, Cálculo numérico aplicado à engenharia elétrica com Matlab, Ed. IFSC, 2019. ISBN 9788584641383.

[3] W. Shen, An introduction to numerical computation, E. World Scientific Publishing, 2018. ISBN 9789814730068.

[4] T. Elser, Factorial Design: Understanding design of experiments (DoE) and applying it in practice, Ed. CreateSpace Independent Publishing Platform, 2017. ISBN 9781542906111.

[5] Il H. Park, Design Sensitivity analysis and optimization of electromagnetic systems (Mathematical and Analytical Techniques with Applications to Engineering). Ed. Springer, 2019. ISBN 9789811343643.

[6] M. Back, Programação na engenharia! Guia prático para começar a programar em Linguagem C. e-book Kindle Edition, 2017.

[7] https://www.mathworks.com/products/matlab.html

[8] https://python.org.br/

[9] M. Awad and R. Khanna, Efficient Learning Machine: theories, concepts and applications for engineers and system designers, eBook Kindle, 2015. ISBN 1430259892.

[10] F. S. Hiller and G. J. Lieberman, Introdução à pesquisa operacional, Ed. McGraw Hill, 2013. ISBN 9780073376299.

[11] E. M. Azevedo, Modelo computacional de teoria dos jogos aplicado aos leilões brasileiros de energia elétrica. 2004. Tese, Universidade Estadual de Campinas, São Paulo. https://www.repositorio.unicamp.br/handle/REPOSIP/264528

[12] J. Baumaister and A. Leitão, Introdução à teoria de controle e programação dinâmica, Ed. IMPA, 2014. ISBN 9788524402715.

[13] L. Xiang, F. Chen, W. Ren and G. Chen, Advances in network controllability, in IEEE Circuits and Systems Magazine, vol. 19, no. 2, pp. 8-32, 2019. doi: 10.1109/MCAS.2019.2909446.

[14] M. P. Bendsoe and O. Sigmund, Topology optimization: theory, methods and applications, Ed. Springer-Verlag Berlin Heidelberg, 2004. ISBN 9783662050866.

[15] K. E. Jensen, Performing topology optimization with the density method, COMSOL Multiphysics, 2019. https://www.comsol.com/blogs/performing-topology-optimization-with-the-density-method/

[16] S. K. Pundir, Linear Programming, Ed. CBS Publishers, 2020. ISBN 9789389396324.

[17] M. S. Bazaraa, Nonlinear Programming: Theory and Algorithms, Ed. Wiley Publishing, 2013. ISBN 9781118857564.

[18] C. A. Coello Coello, G. B. Lamont and D. A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd ed., Ed. Springer, 2007. ISBN 9780387332543.

[19] J. Fulcher, Computational Intelligence: a compendium, Ed. Springer-Verlag, 2008. ISBN 9783540782933.

# *Formulação do problema*

Nas metodologias de otimização paramétrica, o processo de otimização pode ser dividido em duas partes distintas: o modelo físico-matemático do problema e o mecanismo otimizador. As escolhas feitas em cada parte implicarão na qualidade da solução dita ótima. O processo iterativo pode ser resumido como:

(i) O modelo do problema, recebendo uma solução possível, deve prover uma avaliação de mérito(s);

(ii) O método otimizador deve receber o(s) mérito(s) e utilizá-lo(s) como informação para a proposta de uma nova solução.

O processo iterativo deve ser conduzido de forma que as propostas de soluções vão obtendo avaliações de mérito(s) cada vez melhores.

A escolha de qual metodologia de otimização usar de maneira a ser mais eficiente depende das características do problema. As mais significativas são:

- Formulação escalar ou vetorial;
- Problema de natureza predominantemente linear ou não linear;
- Superfícies de nível e modalidade;
- Continuidade e diferenciabilidade;
- Convexidade, quasi-convexidade ou funções não convexas;
- Mínimos locais e mínimos globais (ou máximos);

Cada item, se existente, aporta mais complexidade. O capítulo 3 apresenta a matemática para tal entendimento. Muitas vezes esse diagnóstico somente é possível com a experiência do projetista.

A questão de um único ou múltiplos objetivos exige ainda maior atenção. Este capítulo é dedicado a isto: formulação escalar ou vetorial.

O projeto ótimo é aquele que possui máximo desempenho com mínimo custo e não viola restrições, por exemplo. Se tal solução existe, basta resolver o problema com uma abordagem mono-objetivo. A solução ótima para um objetivo também o será para o outro objetivo. Entretanto, a abordagem multicritério é relevante quando a solução ótima correspondente a cada função objetivo é diferente da(s) outra(s). Neste caso, os objetivos são ditos conflitantes, ou seja, a melhora de um acarreta na deterioração de outro(s), e não devem ser otimizados de maneira escalar. Este conflito ou compromisso entre os objetivos deve ser compreendido.

De modo geral, sistemas ou dispositivos com alto desempenho tendem a ter alto custo, enquanto dispositivos mais simples e baratos usualmente resultam em baixo desempenho. Dependendo das demandas do mercado, uma solução intermediária (desempenho satisfatório e custo aceitável) pode ser dita 'ótima'.

Esta discussão torna claro que uma outra noção do que vem a ser 'solução ótima' é necessária para problemas multiobjetivos. A formulação multiobjetivo também é chamada multicritério ou vetorial.

## 2.1 Abordagem escalar ou mono-objetivo

A abordagem mono-objetivo de um problema significa que a função de mérito cujo método de otimização deve minimizar (ou maximizar) é um funcional, ou seja, uma função na qual a imagem é um escalar. Seja $\vec{x} \in \mathbb{R}^n$ o vetor de parâmetros que devem ser ajustados e $f(.): \mathbb{R}^n \mapsto \mathbb{R}$ o funcional-objetivo que quantifica cada solução $\vec{x}$ (por convenção, quanto menor $f(\vec{x})$ melhor será a solução $\vec{x}$). O problema de otimização

escalar irrestrito pode ser expresso como [1]:

$$\vec{x}^* = arg\min_x f(\vec{x}) \quad , \tag{2.1}$$

ou seja, o método de otimização deve ser capaz de determinar o vetor $\vec{x}^*$ que minimiza o funcional *f(.)*.

Em problemas da engenharia, além de minimizar determinada função objetivo, é comum ter que atender a certas restrições como por exemplo limitações construtivas. Para expressar tais restrições, definem-se regiões no espaço de parâmetros $\mathbb{R}^n$ através de $m$ desigualdades ($\vec{g}(\vec{x})$) ou $n$ de igualdades ($\vec{h}(\vec{x})$).

Assim, o problema de otimização escalar restrito pode ser expresso por:

$$\vec{x}^* = arg\min_x f(\vec{x}) \text{ sujeito a:}$$

$$\vec{g}(\vec{x}) = \left(g_1(\vec{x}), g_2(\vec{x}), \dots, g_m(\vec{x})\right) \leq 0 \tag{2.2}$$

$$\vec{h}(\vec{x}) = \left(h_1(\vec{x}), h_2(\vec{x}), \dots, h_n(\vec{x})\right) = 0$$

**Exemplo de otimização escalar**

Seja um problema com uma única variável (*x*) e um objetivo (*f(x)*):

$$f(x) = x^2 \quad , \tag{2.3}$$

sendo que $x \in$ [-2 3], restrição de limites. De acordo com (2.3), e considerando o desejo por minimização, o resultado da otimização mono-objetivo é uma solução única, chamada ótima: $x^* = 0$, pois *f(0)=0* é o menor valor possível.

## 2.2 Abordagem vetorial ou multiobjetivo

Em grande parte dos problemas reais, um projeto deve atender a $k$ funções objetivo, a um conjunto de $m$ restrições aos parâmetros ($\vec{g}(\vec{x})$ e $\vec{h}(\vec{x})$) assim como a $e$ restrições aos objetivos ($\vec{e}(\vec{x})$). Portanto, trata-se de otimização multiobjetivo restrita [2], podendo ser assim descrito:

$$\begin{aligned} &\text{Minimizar} \quad \vec{y} = \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})) \quad , \\ &\text{sujeito à} \quad \vec{g}(\vec{x}) = (g_1(\vec{x}), g_2(\vec{x}), \dots, g_m(\vec{x})) \leq 0 \quad , \\ &\qquad\qquad \vec{h}(\vec{x}) = \left(h_1(\vec{x}), h_2(\vec{x}), \dots, h_m(\vec{x})\right) = 0 \quad \text{e} \\ &\qquad\qquad \vec{e}(\vec{y}) = (e_1(\vec{y}), e_2(\vec{y}), \dots, e_j(\vec{y})) \leq 0 \quad , \end{aligned} \tag{2.4}$$

$$\text{com} \qquad \vec{x} = (x_1, x_2, \dots, x_n) \in X^n \quad \text{e} \quad \vec{y} = (y_1, y_2, \dots, y_k) \in Y^k \ ,$$

onde $\vec{x}$ é o vetor de parâmetros, $\vec{y}$ é o vetor de objetivos, $X$ determina o espaço de parâmetros ($n$ dimensões) e $Y$ o espaço de objetivos ($k$ dimensões). As restrições $\vec{g}(\vec{x})$ e $\vec{h}(\vec{x})$ determinam o domínio de soluções factíveis:

$$X_f = \left\{\vec{x} \in X^n \quad \middle| \quad \vec{g}(\vec{x}) \leq 0 \; e \; \vec{h}(\vec{x}) = 0\right\} . \tag{2.5}$$

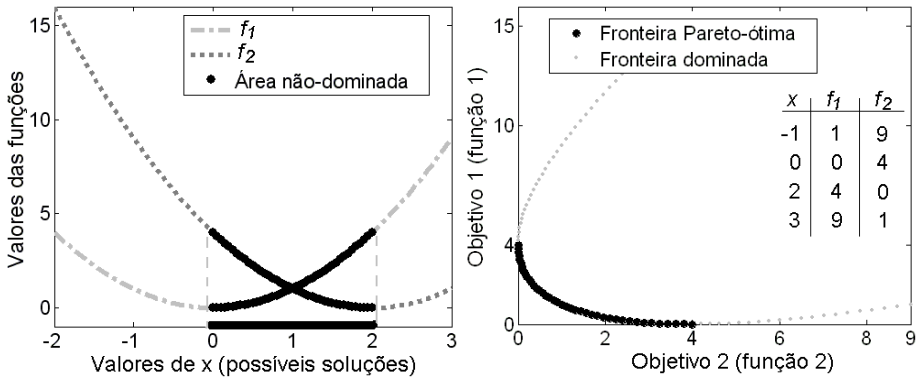
**Exemplo de otimização vetorial**

Seja um problema com uma única variável ($x$) e dois objetivos ($f_1$ e $f_2$):

$$f_1(x) = x^2 \qquad \text{e} \qquad f_2(x) = (x-2)^2 \ , \tag{2.6}$$

sendo que $x \in [-2\ 3]$, restrição de limites. Os objetivos consistem na minimização destas duas funções simultaneamente. De acordo com (2.4), o resultado de um problema multicritério é um grupo de soluções ótimas $x^*$, aqui pertencente ao intervalo [0 2].

A Figura 2.1a mostra as funções e as soluções aqui chamadas de não-dominadas. Observa-se que todas as soluções são não-inferiores para $x$ no intervalo [0 2]. A Figura 2.1b apresenta $f_1(x)$ em relação a $f_2(x)$. Os pontos não-dominados identificam a fronteira Pareto-ótima.




(a) funções versus possíveis soluções. (b) objetivo 1 versus objetivo 2.

Figura 2.1 Exemplo de minimização com um parâmetro e dois objetivos.

### 2.2.1 Comparação de vetores

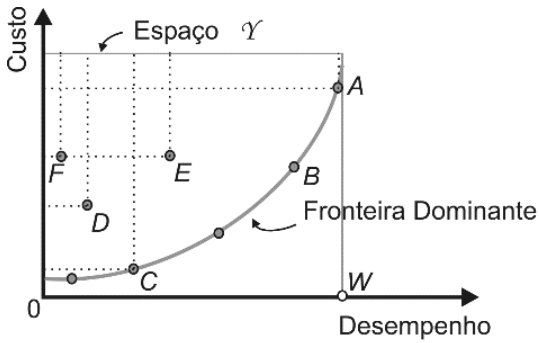
Na otimização escalar, as soluções ótimas podem ser totalmente ordenadas de acordo com a função de mérito $f$: para duas soluções $\vec{a}$, $\vec{b} \in X_f$ tem-se que $f(\vec{a}) \geq f(\vec{b})$ ou $f(\vec{b}) \geq f(\vec{a})$. O objetivo é encontrar a solução que possui o maior (ou o menor) valor para $f$. Quando múltiplos objetivos estão envolvidos, a situação muda: $X_f$ não pode, em geral, ser totalmente ordenado, mas apenas ordenado parcialmente. Isto porque $f$ passa a ser um vetor (2.4). Para dois vetores quaisquer ($\vec{u}$ e $\vec{v}$) de $k$ objetivos, esta situação pode ser matematicamente expressa da seguinte maneira [2][3]:

$$\vec{u} = \vec{v} \text{ se e somente se } \forall i \in \{1,2,\dots,k\} \quad : \quad u_i = v_i$$

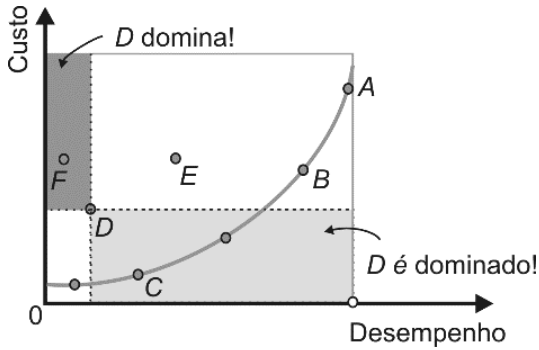
$$\vec{u} \geq \vec{v} \text{ se e somente se } \forall i \in \{1,2,\dots,k\} \quad : \quad u_i \geq v_i \qquad (2.7)$$

$$\vec{u} > \vec{v} \text{ se e somente se } \quad \vec{u} \geq \vec{v} \quad \wedge \quad \vec{u} \neq \vec{v} \,.$$

Para as relações "≤" e "<" as expressões são análogas. Seguindo com o exemplo de desempenho versus custo, a Figura 2.2 apresenta soluções que ilustram este ordenamento parcial.




(a) comparação de vetores.




(b) conceito de dominância.

Figura 2.2 Relações entre objetivos – dominância.

A solução representada pelo ponto *D* é melhor que a representada pelo ponto *F:* ela possui maior desempenho e menor custo. Comparando *F* e *E*, percebe-se que *E* é também melhor que *F*, pois com um mesmo custo gera um desempenho melhor.

Para utilizar a notação (2.4), deve-se fazer uso primeiro de uma *função de ajuste* de modo a obter dois problemas de maximização ou dois de minimização. Uma maneira simples de construir esta função de ajuste para transformar, por exemplo, um problema de minimização em maximização é: *f* = (*constante – custo*), onde a 'constante' é um valor superior a qualquer possível valor de custo.

Utilizando esta transferência, obtém-se de (2.4): $C > D$, $D > F$, e, por consequência, $C > F$. Entretanto, quando as soluções *C* e *A* são comparadas, não se pode definir qual a melhor, pois *A* tem melhor desempenho, mas *C* tem menor custo ( $A \not> C$ e $C \not> A$ ). Consequentemente, para os problemas multiobjetivos, quando duas possíveis soluções *a* e *b* são confrontadas, existem três possibilidades:

$$\vec{f}(\vec{a}) \geq \vec{f}(\vec{b}), \quad \vec{f}(\vec{b}) \geq \vec{f}(\vec{a}) \quad \text{e} \quad \vec{f}(\vec{a}) \ngeq \vec{f}(\vec{b}) \wedge \vec{f}(\vec{b}) \ngeq \vec{f}(\vec{a}). \qquad (2.8)$$

A solução *W* – desempenho máximo com custo mínimo – é irreal quando os objetivos são conflitantes. Para classificar estas diferentes situações, pode-se utilizar o conceito de dominância por Pareto.

### 2.2.2 Dominância de Pareto

Para quaisquer dois vetores de parâmetros $\vec{a}$ e $\vec{b}$ [2][3]:

$\vec{a} > \vec{b}$ ($\vec{a}$ domina $\vec{b}$) se e somente se $\vec{f}(\vec{a}) > \vec{f}(\vec{b})$

$\vec{a} \geq \vec{b}$ ($\vec{a}$ domina fracamente $\vec{b}$) se e somente se $\vec{f}(\vec{a}) \geq \vec{f}(\vec{b})$ (2.9)

$\vec{a} \sim \vec{b}$ ($\vec{a}$ é indiferente a $\vec{b}$) se somente se

$$\vec{f}(\vec{a}) \not\succeq \vec{f}(\vec{b}) \wedge \vec{f}(\vec{b}) \not\succeq \vec{f}(\vec{a}).$$

As definições para problemas de minimização ($\prec, \preceq, \sim$) são análogas.

Na Figura 2.2(b), o retângulo cinza escuro delimita a região no espaço de objetivos que é dominada pelo vetor de parâmetros representado por D. Qualquer solução que corresponda a posições dentro do retângulo cinza claro domina a solução representada por D. Para qualquer outro caso, soluções fora dos dois retângulos, D será indiferente.

Portanto, devido a este caráter vetorial, distinguem-se dois tipos de soluções:

- Haverá soluções que, considerando *todos* os objetivos propostos, serão piores que outras. Estas são chamadas de soluções *dominadas* ou *não-eficientes*;

- Haverá ainda soluções que, quando comparadas com *todas* as outras, serão melhores em um ou mais objetivos e piores em outro ou outros. Neste caso, elas são consideradas como indiferentes; não é possível comparar essas soluções entre elas ou dizer qual é a melhor. Estas soluções são chamadas *eficientes* ou *não-dominadas*.

### 2.2.3 Otimalidade de Pareto[1]

O vetor de parâmetros $\vec{a} \in X_f$ é dito não-dominado se e somente se [2][3]:

$$\nexists \vec{x} \in X_f : \vec{x} \succ \vec{a} \ . \tag{2.10}$$

[1] Pode-se definir otimalidade pela informação do gradiente das funções: condições de Karush-Kuhn-Tucker [4]. O capítulo 3 discute esse conceito.

Assim, $\vec{a}$ é declarado *Pareto-ótimo* se e somente se $\vec{a}$ é não-dominado em $X_f$. Fica a ressalva que só se obtêm o conjunto *Pareto-ótimo* se as soluções não-dominadas forem ótimas, lembrando que otimalidade é usualmente irreal em problemas reais.

Na Figura 2.2 os pontos *A*, *B* e *C* representam soluções Pareto-ótimas. Uma não é melhor nem pior do que as outras quando se levam em conta todos os objetivos.

Esta é a principal diferença com relação à abordagem escalar: para problemas multiobjetivos não existe uma única solução ótima, mas um conjunto ótimo no qual nenhuma destas soluções pode ser identificada como melhor sem uma nova classificação (por exemplo, a preferência por um dos objetivos).

A união de todas as soluções não dominadas é chamada de *conjunto Pareto-ótimo*. Por correspondência, o conjunto de seus vetores objetivo forma a *Fronteira Pareto-ótima*.

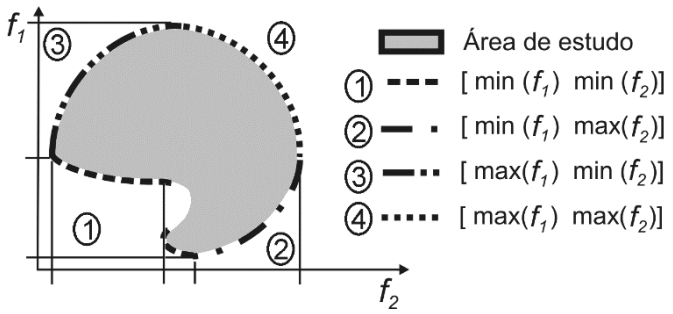
### 2.2.4 Conjuntos não-dominados e fronteiras

Seja $A \subseteq X_f$ um grupo de dispositivos factíveis, e $p(.)$ uma função que determina as soluções não-dominadas em não importa qual subdomínio de $X$ ($n$ parâmetros e $m$ objetivos). $p(A)$ é então o conjunto de elementos de $A$ não-dominados, e o grupo de vetores $\vec{f}(p(A))$ é a fronteira não-dominada correspondente à $A$ no espaço de objetivos. Além disso, o conjunto $X_p = p(X_f)$ é chamado de Pareto-ótimo e $Y_p = \vec{f}(X_p)$ é conhecido como fronteira Pareto-ótima.

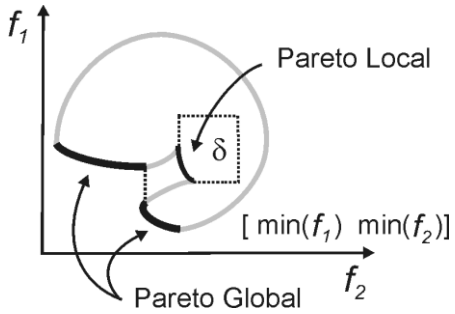
O principal desafio da otimização vetorial é encontrar o maior número possível de soluções não-dominadas. Isto porque, com a fronteira Pareto-ótima, consegue-se entender a dependência entre os objetivos e, portanto, o comportamento do problema.

A Figura 2.3 apresenta um exemplo de problema restrito a um domínio de estudo. Neste exemplo, têm-se duas funções ($f_1$ e $f_2$) quaisquer e

são apresentadas combinações para minimização (*min*) e maximização (*max*) de cada um dos objetivos. Estas diferentes combinações definem as várias fronteiras Pareto-ótimas.




(a) espaço bidimensional.




(b) Pareto global e local.

Figura 2.3. Fronteiras Pareto.

A fronteira Pareto-ótima contém a totalidade das soluções ótimas. Entretanto, como na abordagem escalar para problemas multimodais, podem existir também ótimos locais que acabam constituindo conjuntos não-dominados para determinadas vizinhanças. Neste contexto, tem-se o conceito correspondente à fronteira Pareto-ótima local. Seja $A \subseteq X_f$ um conjunto de vetores de parâmetros:

1. O conjunto $A$ é Pareto-ótimo local se e somente se:

$$\forall \quad \vec{a} \in A \quad : \quad \nexists \vec{x} \in X_f : \vec{x} \succ \vec{a} \wedge \|\vec{x} - \vec{a}\| < \varepsilon \wedge \left\|\vec{f}(\vec{x}) - \vec{f}(\vec{a})\right\| < \delta \qquad (2.11)$$

onde $||.||$ é uma métrica para distância e $\varepsilon > 0$ (raio mínimo no espaço de parâmetros), $\delta > 0$ (raio mínimo no espaço de objetivos).

Estes raios mínimos são determinados pelo projetista com a intenção de estudar uma região específica. O capítulo 7 apresenta uma maneira alternativa para determinação de Pareto locais, sem a necessidade da experiência do engenheiro na especificação dos raios mínimos.

2. O conjunto $A$ é Pareto-ótimo global se e somente se:

$$\forall \vec{a} \in A : \nexists \vec{x} \in X_f : \vec{x} \succ \vec{a} \ . \qquad (2.12)$$

A diferença entre ótimo local ou global pode ser vista na Figura 2.3(b).

### 2.2.5 Busca e decisão

A resolução de problemas vetoriais é dividida, basicamente, em duas etapas: determinação das soluções eficientes e a etapa de decisão. O primeiro aspecto consiste na busca de soluções Pareto-ótimas dentro do espaço factível. O segundo aspecto, que envolve um procedimento que pode ser chamado de decisor, diz respeito à seleção da solução que é um compromisso final dentre aquelas de Pareto. Para tal ação, o projetista toma uma decisão externa ao processo de otimização.

Dependendo de como e quando o processo de otimização e a etapa de decisão são combinados, os métodos de resolução podem ser classificados em três categorias:

- Decisão antes do processo de procura (a priori): a(o) engenheira(o) decide o compromisso que ele quer obter antes de lançar o método de resolução (busca). Isto acaba por transformar um problema multicritério em uma abordagem escalar. Tal decisão é crítica e será abordada em detalhes na seção 2.3.

- Decisão durante o processo de procura (progressivo): é o procedimento que faz escolhas durante o processo de obtenção das soluções não-dominadas. O resultado da consulta ao decisor é utilizado na busca de novas soluções eficientes.

  Nesta abordagem também se faz necessária certa experiência do projetista, já que as escolhas deverão ser tomadas de modo a orientar o processo de otimização a caminhar na direção da formação da fronteira Pareto-ótima.

  Um dos métodos progressivos mais conhecidos é o *MiniMax* [5]. Eventualmente, a tomada de decisão progressiva pode ser utilizada para a redução do espaço de busca.

- Decisão após o processo de procura (a posteriori): a apresentação das decisões após a etapa de definição das soluções eficientes talvez seja a mais lógica das três, isto porque as escolhas serão feitas de acordo com as respostas finais encontradas.

  Como já dito, com o conjunto Pareto-ótimo definido torna-se possível conhecer o comportamento do problema em relação aos objetivos analisados. Conhecendo-se as relações de dependência entre eles, a escolha final é facilitada.

Na próxima seção são apresentadas algumas das dificuldades usualmente encontradas nos problemas multiobjetivos. O conhecimento dessas dificuldades permite criar critérios para nortear a qualificação das metodologias de busca das soluções não-dominadas.

### 2.2.6 Dificuldades adicionais da otimização vetorial

Assim como na otimização escalar, as dificuldades de resolução de problemas multicritério são decorrentes da presença de restrições e do comportamento das funções objetivo. As principais estão ilustradas na Figura 2.4: (a) convexidade, (b) descontinuidades e (c) multimodalidade (múltiplos ótimos locais e/ou globais). A matemática destas dificuldades é apresentada no capítulo 3.

Além destas dificuldades, pode-se citar também a 'não uniformidade' das soluções no espaço dos objetivos (d). Certos problemas e/ou métodos de resolução podem apresentar características que concentram as soluções em determinadas áreas (esta 'concentração de soluções' só diz respeito aos métodos por populações, como será discutido nos capítulos seguintes).

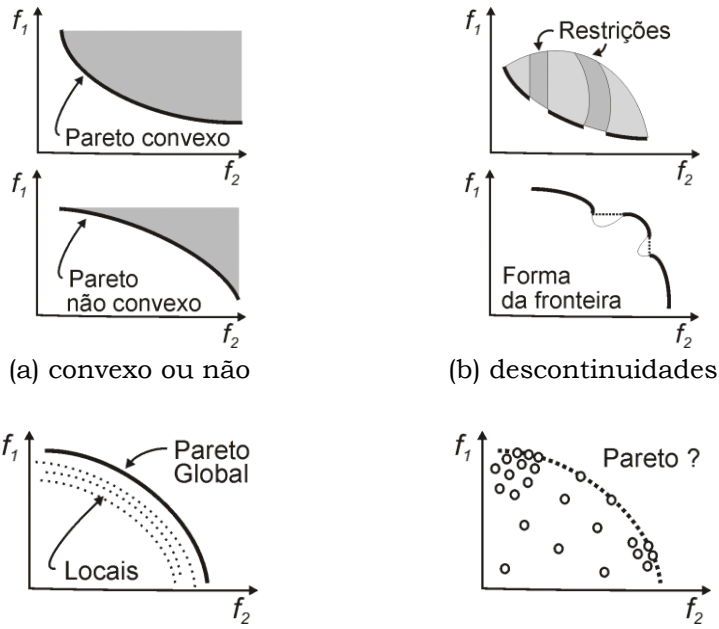



Figura 2.4. Dificuldades adicionais de problemas multiobjetivo.

Se estas regiões não forem próximas às soluções Pareto ou se elas contemplarem apenas um pedaço da fronteira ótima, a caracterização de todo o conjunto Pareto-ótimo pode ser comprometida. A correta determinação do grupo de soluções eficientes é fundamental para entender o comprometimento entre os objetivos.

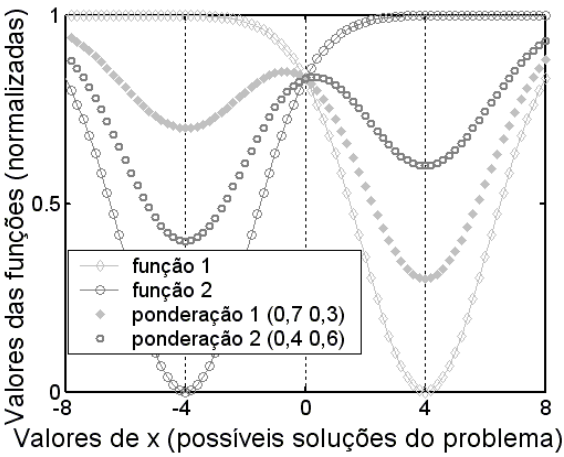
## 2.3 Perigos da alteração de um problema vetorial para escalar

Um problema multicritério pode ser transformado em uma abordagem mono-objetivo. Para tal, a(o) projetista deve tomar decisões antes do processo de otimização. Algo similar a:

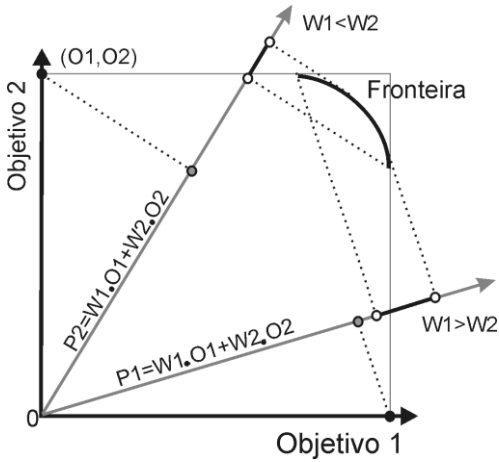
$$\begin{aligned} &\text{Maximizar } \; y = f(\vec{x}) = w_1 f_1(\vec{x}) + w_2 f_2(\vec{x}) +, \dots, + w_k f_k(\vec{x}) \;, \\ &\qquad \text{sujeito a } \; \vec{x} \in X_f \quad \text{e, geralmente, } \sum w_i = 1. \end{aligned} \tag{2.13}$$

Após está transformação, pode-se aplicar uma técnica mono-objetivo para a resolução do problema. Entretanto, a adequação dos pesos $w$ para cada objetivo não é evidente, notadamente quando os objetivos são extremamente conflitantes (isto é, quando o máximo de um objetivo é o mínimo de outro(s)) como ilustrado na Figura 2.5a.

A Figura 2.5b oferece uma possível interpretação da ponderação de objetivos para duas situações diferentes (uma privilegiando o objetivo 1 e outra o objetivo 2). A ponderação correspondente a uma projeção da fronteira Pareto sobre uma direção ($w_1$, $w_2$) e acaba criando uma ordenação. De acordo com a escolha dos valores relativos de $w_1$ e $w_2$, o 'melhor' indivíduo é totalmente outro. Decisões malfeitas podem acarretar na imposição desnecessária de restrições.




(a) problema multiobjetivo com objetivos conflitantes




(b) ponderações P1 privilegiando o objetivo 1 (W1>W2) e P2 privilegiando o objetivo 2 (W1<W2)

Figura 2.5. Ponderação de objetivos conflitantes.

**Referências**

[1] A. Izmailov and M. Solodov, Otimização – volume 1, Ed. IMPA, 2014. ISBN 9788524403897.

[2] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Ed. Wiley, 2002. ISBN 9780471873396.

[3] D. T. Luc, Pareto Optimality. In: A., Pardalos P.M., Migdalas A., Pitsoulis L. (eds) Pareto Optimality, Game Theory And Equilibria. Springer Optimization and Its Applications, vol 17. Ed. Springer, 2008. ISBN 9780387772479.

[4] M. S. Bazaraa, Nonlinear Programming: Theory and Algorithms, Ed. Wiley Publishing, 2013. ISBN 9781118857564.

[5] V. F. Dem'yanov and V. N. Malozemov, Introduction to Minimax, Ed. Dover Publications, 2014. ISBN 9780486664231.

[6] C. A. Coello Coello, G. B. Lamont and D. A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd ed., Ed. Springer, 2007. ISBN 9780387332543.

[7] A. Izmailov and M. Solodov, Otimização – volume 2, Ed. IMPA, 2012. ISBN 9788524402685.

# *Comportamento do problema a ser resolvido*

O capítulo anterior discutiu as formulações escalar ou vetorial para o problema de otimização. A escolha de métodos adequados para resolve-lo depende da natureza das funções objetivo ($\vec{f}(\vec{x})$) e das funções que traduzem as restrições nos parâmetros ($\vec{g}(\vec{x})$ e $\vec{h}(\vec{x})$) e nos objetivos ($\vec{e}(\vec{x})$).

Pode-se afirmar que não existe uma técnica que melhor resolve todos os problemas, pois eles têm natureza diversa. De forma geral, ou os métodos lidam com menos informação do problema (baseado em probabilidades, por exemplo) e obtém resultados que podem ser questionados e apenas confirmados pela repetição; ou os métodos necessitam de muita informação (baseado no gradiente das funções envolvidas, por exemplo), o que nem sempre se tem disponível de forma simples mas pode levar a certeza do ótimo, quando alcançam sucesso. Essas reflexões foram introduzidas no capítulo 1 e voltam ao final do presente capítulo.

Antes, este capítulo apresenta a caracterização do problema de otimização em termos do comportamento das suas funções. As definições aqui apresentadas são relacionadas a questão de o quê são as soluções do problema de otimização:

1. Dado $\vec{f}(\vec{x})$, sujeito as restrições $\vec{g}(\vec{x}) \leq 0$ , $\vec{h}(\vec{x}) = 0$ e $\vec{e}(\vec{y}) \leq 0$ , o que são os pontos de mínimo ou máximo de $\vec{f}(\vec{x})$? portanto, o que são as soluções de problema de otimização?

2. Dado um ponto $\vec{x} \in \mathbb{R}^n$ , que tipo de avaliações pode ser feitas para se determinar se esse ponto é ou não é um ponto de mínimo ou máximo de $\vec{f}(\vec{x})$?

Neste capítulo são estabelecidas definições matemáticas, as quais buscam responder as perguntas realizadas. Esses conceitos são bem definidos na literatura especializada. Aqui, eles são apresentados de forma rápida. O intuito aqui é capacitar o leitor para a sequência do livro. Algumas referências são citadas ao fim deste capítulo, caso se deseje aprofundar estudos sobre a matemática aqui apresentada.

## 3.1 Superfície de nível e modalidade

**Superfície de nível –** seja $f(\cdot): C \subset \mathbb{R}^n \to \mathbb{R}$. A superfície de nível $S(f, \alpha)$, associada ao nível $\alpha$, é definido como:

$$S(f, \alpha) = \{x \in C \mid f(x) = \alpha\} \quad . \tag{3.1}$$

**Região subnível –** seja $f(\cdot): C \subset \mathbb{R}^n \to \mathbb{R}$. A região de nível $R(f, \alpha)$, associada ao nível $\alpha$, é definido como:

$$R(f, \alpha) = \{x \in C \mid f(x) \leq \alpha\} \quad . \tag{3.2}$$

Usualmente $S(f, \alpha)$ corresponde a uma fronteira de $R(f, \alpha)$, embora seja possível escolher $C$ de forma que isso não ocorra. É válida, portanto, uma relação de ordenação das regiões de subnível de uma função. Seja $f(\cdot): C \subset \mathbb{R}^n \to \mathbb{R}$, as regiões de subnível dessa função obedecem a

$$R(f, \alpha_1) \supset R(f, \alpha_2) \leftrightarrow \alpha_1 > \alpha_2 \quad . \tag{3.3}$$

Assim, problemas de otimização podem ser equivalentes a um problema de determinar pontos que estejam sucessivamente no interior de subnível cada vez mais inferiores.

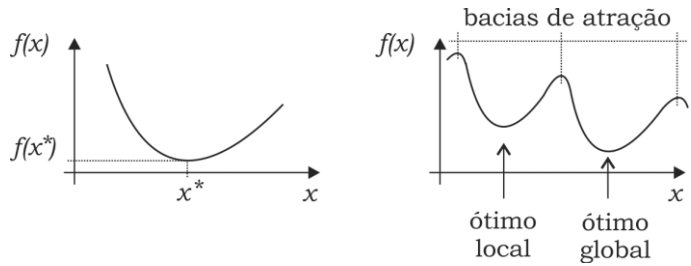
As regiões de subnível, analisadas sob o ponto de vista topológico, definem uma categorização para as funções:

**Função unimodal –** seja $f(\cdot): C \subset \mathbb{R}^n \to \mathbb{R}$, diz-se que $f(\cdot)$ é unimodal se $R(f, \alpha)$ é conexo[1] para todo $\alpha \in \mathbb{R}$. Ainda, que $f(\cdot)$ é estritamente unimodal se, além disso, $R(f, \alpha)$ é um conjunto compacto[2] para todo $\alpha \in \mathbb{R}$.

Por simetria, tem-se:

**Função multimodal –** seja $f(\cdot): C \subset \mathbb{R}^n \to \mathbb{R}$, diz-se que $f(\cdot)$ é multimodal se existe $\alpha \in \mathbb{R}$ tal que $R(f, \alpha)$ não é conexo.

Ao redor de mínimos locais, sempre haverá regiões nas quais a função se comportará de maneira unimodal. Tais regiões podem ser definidas como bacias de atração associadas a tais mínimos. Para isto, é necessário definir preliminarmente a região conexa de subnível. A Figura 3.1 tais situações.




(a) função unimodal (b) função multimodal

Figura 3.1 Ilustração de ótimo local, ótimo global e bacias de atração.

[1] um espaço conexo é um espaço topológico que não pode ser representado com a união de dois ou mais conjuntos abertos disjuntos e não-vazios.

[2] em topologia geral, o conceito de compacidade é uma extensão topológica das ideias de finitude e limitação.

**Região conexa de subnível –** seja $f(\cdot)\colon C \subset \mathbb{R}^n \to \mathbb{R}$, seja a região de subnível $R(f,\alpha)$, associada ao nível $\alpha$, e seja um ponto $x \in R(f,\alpha)$. A região conexa de subnível $R_c(f,\alpha,x_0)$ é definida como o maior subconjunto conexo de $R(f,\alpha)$ que contém $x_0$.

**Bacia de atração –** seja $f(\cdot)\colon C \subset \mathbb{R}^n \to \mathbb{R}$, seja $x^* \in C$ um mínimo local de $f(\cdot)$. A bacia de atração de $x^*$ é definida como a maior região conexa de subnível associada a $x^*$, sendo $\alpha^*$ o nível correspondente, tal que a função restrita a essa região:

$$f(\cdot)\colon R_c(f,\alpha^*,x^*) \to \mathbb{R} \tag{3.4}$$

é unimodal. A bacia de atração é dita estrita se nessa região a função é estritamente unimodal.

## 3.2 Continuidade e diferenciabilidade

Hipóteses de continuidade e de diferenciabilidade das funções permitem entender implicações a respeito de superfícies de nível e bacias de atração:

seja $f(\cdot)\colon C \subset \mathbb{R}^n \to \mathbb{R}$, se $f(\cdot)$ é contínua no domínio $C$ , então:

$$dist\big(S(f,\alpha_1),S(f,\alpha_2)\big) > 0 \quad \forall \quad (\alpha_1\,,\alpha_2)||\alpha_1 - \alpha_2| > 0 \tag{3.5}$$

sendo $dist(.,.)$ a função distância.

**Corolário[3] –** superfícies de nível de funções contínuas não se tocam nem se cruzam. Seja $f(\cdot)\colon C \subset \mathbb{R}^n \to \mathbb{R}$, se $f(\cdot)$ é diferenciável no domínio $C$ , então toda a superfície de nível $S(f,\alpha)$ é suave, sendo o

[3] Corolário é uma afirmação deduzida de uma verdade já demonstrada.

hiperplano tangente à superfície em cada ponto perpendicular ao gradiente da função no ponto.

A proposição de diferenciabilidade de uma função permite pensar em estratégias de otimização baseadas no fato de que o gradiente de uma função indica quais são as direções do espaço para as quais, partindo-se de um ponto, ocorre localmente a diminuição da função. Isso equivale à determinação das direções para as quais se caminha para regiões de subnível inferiores. A proposição a seguir formaliza esse fato.

seja $f(\cdot): C \subset \mathbb{R}^n \to \mathbb{R}$, se $f(\cdot)$ uma função diferenciável no domínio $C$, seja $x_0$ um ponto pertencente à superfície de nível $S(f,\alpha)$ e seja $\nabla f(x_0)$ o gradiente de $f(\cdot)$ no ponto $x_0$. Seja ainda um vetor $d \in \mathbb{R}^n$. Então, se

$$d \,.\, \nabla f(x_0) < 0 \tag{3.6}$$

então existe $\epsilon > 0$ tal que:

$$f(x_0 + \epsilon d) < f(x_0)\ . \tag{3.7}$$

### 3.3 Convexidade e quase-convexidade

**Função convexa –** diz-se que uma função $f(\cdot): C \subset \mathbb{R}^n \to \mathbb{R}$ definida sobre um conjunto convexo $C$ é convexa se para quaisquer $x, y \in C$ ,

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y) \tag{3.8}$$

para todo $\alpha \in [0,1]$. Se para quaisquer $x, y \in C$, sendo $x \neq y$ e $0 < \alpha < 1$, a desigualdade é estrita, então $f(\cdot)$ é estritamente convexa. Igualmente, $f(\cdot)$ é côncava se $-f(\cdot)$ for convexa.

**Subgradiente –** seja uma função convexa $f(\cdot): C \subset \mathbb{R}^n \to \mathbb{R}$, um funcional linear $f^{sb}$ é um subgradiente de $f$ no ponto $x_0$ se:

$$f(x) \geq f(x_0) + f^{sb}(x - x_0)\,,\ \forall\, x \quad . \tag{3.9}$$

**Caracterização de funções convexas –** seja $f(\cdot)$ uma função duas vezes diferenciável, sobre um conjunto convexo $C \subset \mathbb{R}^n$. Então são equivalentes as afirmativas:

*i.* $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y) \ \ \forall\, \alpha \in [0,1]$

*ii.* $f(y) \geq f(x) + \nabla f(x)'(y - x) \ \ \forall\, x, y \in C$

*iii.* $F(x) \geq 0 \ \ \forall\, x \in C$

sendo $\nabla f(x)$ o vetor gradiente no ponto $x$ e $F(x)$ a matriz hessiana no ponto $x$.

Como no caso de conjuntos convexos, é possível obter funções convexas a partir de combinações convexas.

**Combinações convexas –** sejam $f_i(\cdot): C_i \subset \mathbb{R}^n \to \mathbb{R}$ funções convexas definidas sobre conjuntos convexos $C_i\,, i = 1, \dots, m$. Então:

*i.* $\alpha f_i(.)$ é *convexa sobre* $C_i \ \ \forall\, \alpha \ \geq 0$

*ii.* $\sum_{i=1}^{m} \alpha_i f_i(.)$ é *convexa sobre* $\bigcap_{i=1}^{m} \mathrm{C}_i$ *para* $\alpha_i \geq 0, i = 1, \dots, m$

sejam $f(\cdot): C \subset \mathbb{R}^n \to \mathbb{R}$ uma função convexa sobre $C$ convexo. Então a região de subnível $R(f, \alpha)$ é convexa para todo $\alpha \in \mathbb{R}$.

A reciproca não é verdadeira. A convexidade de $R(f,\alpha)$ define um novo tipo de função, as funções quase-convexas.

**Função quasi-convexa –** sejam $f(\cdot): C \subset \mathbb{R}^n \rightarrow \mathbb{R}$ uma função tal que suas regiões de subnível $R(f,\alpha)$ são convexas para todo $\alpha \in \mathbb{R}$. Nesse caso, diz-se que $f(\cdot)$ é quasi-convexa no domínio $C$.

Se $f(\cdot): C \subset \mathbb{R}^n \rightarrow \mathbb{R}$ é uma função quase-convexa, então:

$$f(\alpha x + (1-\alpha)y) \leq max\{f(x), f(y)\} \ \forall\, x, y \in C\,, \forall\, \alpha \in [0,1] \qquad (3.10)$$

Todas as regiões de subnível de uma função convexa num domínio convexo são conjuntos convexos. Uma função convexa em um domínio convexo possui uma única bacia de atração, a qual é um conjunto convexo.

A informação a respeito da convexidade de uma função pode ser usada em um processo de otimização a partir da proposição a seguir:

Seja uma função convexa $f(\cdot): C \subset \mathbb{R}^n \rightarrow \mathbb{R}$, seja um ponto qualquer $x_0 \in \mathbb{R}^n$ , e seja $s(x_0) \in \mathbb{R}^n$ um vetor subgradiente da função no ponto. O conceito de subgradiente, subderivada e subdiferencial surgem do estudo de funções convexas. Então a região de subnível que possui o ponto $x_0$ em sua fronteira está contida no semi-espaço fechado negativo definido pelo vetor subgradiente no ponto $x_0$, ou seja:

$$E_s = \{x \in \mathbb{R}^n \mid (x - x_0).s(x_0) \leq 0\} \qquad (3.11)$$

$$R\big(f, f(x_0)\big) \subset E_s$$

### 3.4 Mínimos locais e mínimos globais

Analisa-se aqui a definição do que são as soluções do problema de otimização. De forma geral:

**Mínimo local –** seja $f(\cdot): C \subset \mathbb{R}^n \to \mathbb{R}$, e $x^*$ é um mínimo local de $f(\cdot)$ sobre $C$ se existe $\epsilon > 0$ tal que

$$f(x^*) \leq f(x)\,, \qquad \forall\ x \in V(x^*, \epsilon)\ \cap\ C \tag{3.12}$$

onde $V(x^*, \epsilon) \triangleq \{x: \|x - x^*\| < \epsilon\}$. O ponto $x^* \in C$ é um mínimo local estrito se vale a desigualdade estrita para $x \neq x^*$.

Portanto, o conjunto $C$ é o subconjunto do espaço $\mathbb{R}^n$ definido pelas restrições:

$$C \triangleq \left\{x \in \mathbb{R}^n \mid g_i(x) \leq 0\,; i = 1, \dots, m\,;\ h_j(x) = 0\,; j = 1, \dots, p\right\} \tag{3.13}$$

O mínimo global do funcional é construído a partir de (3.11). Se for possível escolher $\epsilon > 0$ tal que $V(x^*, \epsilon)\ \cap\ C = C$, então $x^*$ é um mínimo global de $f(\cdot)$ sobre C. O mínimo global é ainda estrito se a desigualdade for satisfeita de modo estrito.

### 3.5 Caracterização do ótimo

Considerando a notação $C^n$ para representar o conjunto das funções n vezes continuamente diferenciáveis, tem-se:

**Otimização irrestrita –** seja $f(\cdot) \in C^2$ e $x \in \mathbb{R}^n$, se forem simultaneamente satisfeitas:

*i.* $\nabla f(x^*) = 0$ e

*ii.* $F(x^*) > 0$

então $x^*$ é um mínimo local estrito de $f(\cdot)$ sobre $\mathbb{R}^n$.

**Otimização restrita, igualdade –** sejam $f(\cdot)$ e $g_i(\cdot) \in C^2, i = 1, \dots, r$ *e* $x^*$ tal que $h_i(x^*) = 0$, $i = 1, \dots, r$ . Se existem multiplicadores $\lambda_1, \lambda_2, \dots, \lambda_r$ tais que

*i.* $\nabla f(x^*) + \sum_{i=1}^{r} \lambda_i \nabla h_i(x^*) = 0$

*ii.* $F(x^*) + \sum_{i=1}^{r} \lambda_i H_i(x^*) > 0$ *sobre* $M = \{y \in \mathbb{R}^n : \nabla h_i(x^*)' y = 0, i = 1, \dots, r)\}$,

são simultaneamente satisfeitos, então $x^*$ é um mínimo local estrito de $f(\cdot)$ sujeito a $h_i(x^*) = 0$, $i = 1, \dots, r$.

**Otimização restrita, desigualdade –** sejam $f(\cdot)$ e $h_i(\cdot) \in C^2$, $i = 1, \dots, p$ *e* $x^*$ tal que $g_i(x^*) \leq 0$, $i = 1, \dots, p$ . Se existem multiplicadores $\lambda_1, \lambda_2, \dots, \lambda_p$ tais que

*i.* $\lambda_i \geq 0, i = 1, \dots, p$

*ii.* $\lambda_i g_i(x^*) = 0, i = 1, \dots, p$

*iii.* $\nabla f(x^*) + \sum_{i=1}^{r} \lambda_i \nabla g_i(x^*) = 0$

$iv.\ F(x^*) + \sum_{i=1}^{p} \lambda_i G_i(x^*) > 0\ sobre\ M = \{y \in \mathbb{R}^n : \nabla g_i(x^*)'y = 0, i \in I(x^*)\},$

$I(x^*) = \{\, i :\ g_i(x^*) = 0,\ \lambda_i > 0 \,\}$

são simultaneamente satisfeitos, então $x^*$ é um mínimo local estrito de $f(\cdot)$ sujeito a $g_i(x^*) \leq 0\ ,\ i = 1, \dots, p$.

Considerando $f(\cdot)$ convexa e considerando que as restrições acabam por determinar uma região factível também convexa, assim, qualquer mínimo local de $f(\cdot)$ é também um mínimo global.

**Direção factível –** seja $\Omega \subset \mathbb{R}^n$ e $f$ uma função diferenciável sobre $\Omega$. Se $x^*$ é um mínimo local de $f$ sobre $\Omega$, então para qualquer $d \in \mathbb{R}^n$ que seja uma direção factível em $x^*$ tem-se que:

$$\nabla f(x^*)'d \geq 0 \qquad (3.14)$$

Demonstração: se $d$ é factível em $x^*$, então

$$x(\alpha) = x^* + \alpha d \quad \in \quad \Omega\ \forall\ \alpha \in [0, \bar{\alpha}]$$

seja $v(\alpha) = f(x(\alpha))$; então $v$ possui um mínimo em $\alpha = 0$. Em série de Taylor:

$$v(\alpha) = v(0) + \dot{v}(0)\alpha + O(\alpha)$$

se $\dot{v}(0) < 0$ então para $\alpha > 0$ suficientemente pequeno tem-se $v(\alpha) < v(0)$, o que contraria a hipótese de otimalidade de $v(\alpha)$. Portanto, $\dot{v}(0) = \nabla f(x^*)'d \geq 0$.

**Condições necessárias da 2° ordem –** seja $\Omega \subset \mathbb{R}^n$ e $f$ uma função duas vezes diferenciável sobre Ω. Se $x^*$ é um mínimo local de $f$ sobre Ω, então para qualquer $d \in \mathbb{R}^n$ factível em $x^*$ tem-se que:

$i.\quad \nabla f(x^*)'d \geq 0$

$ii.\quad \text{se } \nabla f(x^*)'d = 0, \text{então } d'F(x^*)d \geq 0$

Demonstração: suponha que $\dot{v}(0) = \nabla f(x^*)'d = 0$. Neste caso:

$$v(\alpha) = v(0) + \frac{1}{2}\ddot{v}(0)\alpha^2 + O(\alpha^2)$$

seja $\ddot{v}(0) < 0$, então $v(\alpha) < v(0)$ para $\alpha > 0$ suficientemente pequeno, o que contraria a hipótese de otimalidade. Portanto, $\ddot{v}(0) = d'F(x^*)d \geq 0$.

Observa-se que se $x^*$ é um ponto interior de Ω então as condições se reduzem a:

$i.\quad \nabla f(x^*) = 0$

$ii.\quad d'F(x^*)d \geq 0\,,\ \forall\ d\ \in \mathbb{R}^n$

**Ponto regular –** um ponto $x^*$ satisfazendo um conjunto de restrições de igualdade $h(x^*) = 0$ e um conjunto de restrições de desigualdade $g(x^*) \leq 0$ é chamado de ponto regular dessas restrições se os vetores $\nabla h_i(x^*)$ e $\nabla g_j(x^*)$ para todo $j$ tal que $g_j(x^*) = 0$ forem linearmente independentes.

Entendido a caracterização dos mínimos locais, chega-se à condição de primeira ordem para otimalidade de um problema genérico de otimização, apresentada por Kuhn e Tucker em 1951. Tal referência tem importância histórica pois serve de base para diversos algoritmos de otimização existentes.

**Condições necessárias de Karush-Kuhn-Tucker para otimalidade –** seja $x^*$ um ponto regular das restrições do problema de otimização:

$$\min f(x)$$
$$sujeito\ a \begin{cases} h_i(x) = 0, i = 1, \dots, l \\ g_j(x) \leq 0, i = 1, \dots, m \end{cases} \tag{3.15}$$

sendo $f, g, h \in C^1$. Para $x^*$ ser um ótimo local do problema, deve existir um conjunto de multiplicadores de Kuhn-Tucker $\lambda^* \in \mathbb{R}^l$ e $\mu^* \in \mathbb{R}^m$ com $\mu^* \geq 0$ tal que:

$$\nabla f(x^*) + \sum_{i=1}^{l} \lambda_i^* \nabla h_i(x^*) + \sum_{j=1}^{m} \mu_j^* \nabla g_j(x^*) = 0$$
$$\mu_j^* g_j(x^*) = 0 \ \forall\ j = 1, \dots, m \tag{3.16}$$

Como visto no capítulo 2, também é possível definir otimalidade pelo conceito de Pareto.

O projetista, quando diante de um problema formulado em termos de otimização, deve realizar estudos sobre o seu comportamento. Entender se existem superfícies de nível e modalidade, bacia de nível, continuidade e diferenciabilidade, convexidade ou quase-convexidade, mínimos locais e globais é relevante para a escolha do método de otimização mais adaptado a tais características.

## 3.6 Sobre a função objetivo

Conforme o discutido até este momento, fica claro que para se definir uma função objetivo, deve-se primeiro ter um entendimento muito claro do que se deseja otimizar, das variáveis e das restrições bem como da natureza e do comportamento do problema.

Uma função objetivo é a medida para a qualidade de uma solução e a razão da existência do algoritmo de otimização. Em muitos problemas, elas são usualmente intuitivas: o vendedor ambulante deseja encontrar o caminho mais curto; ao lidar com corte de tecido, a programação da máquina deve ser eficiente de tal forma a economizar simultaneamente o material e o tempo da máquina. À medida que os problemas se tornam mais complexos, o mesmo ocorre com os objetivos.

A análise estatística (*numerical design of experiments* – DoE) pode ajudar na definição do problema, em específico quais seriam os parâmetros, as restrições e os objetivos mais significativos para uma determinada situação [8]. A DoE também pode identificar variáveis de controle que podem ser mantidas constantes para evitar que fatores externos afetem os resultados. Dadas as restrições de conhecimento sobre o problema, a DoE ajuda a planejar as definições iniciais baseado em dados estatísticos.

Além disto, procedimentos de análise de sensibilidade podem fornecer uma compreensão abrangente das influências dos diversos parâmetros e suas consequências nos resultados do modelo. O capítulo 9 é dedicado a análise de sensibilidade.

A implementação de um algoritmo de otimização é um processo contínuo e sempre será possível melhorias, principalmente com o aumento do entendimento do problema.

De forma breve, uma função objetivo deve ser solucionável e simples. Isto parece óbvio, mas muitas vezes é doloroso abandonar um modelo (provavelmente mais preciso) em favor de algo mais simples. Partir

com uma implementação muito complexa, onde se leva muito tempo para a avaliação ou aumenta em demasiado a incerteza da análise de uma função objetivo, pode ser irritante logo na primeira implementação. Melhor começar de maneira simples com um modelo com bom desempenho e depois trabalhar com um modelo mais completo se ainda houver demanda por melhorias.

Uma função objetivo de alto desempenho é suave. Ele precisa modelar o espaço de busca o mais suave possível, em face do discutido neste capítulo. Isso facilita a convergência do algoritmo.

## 3.7 O comportamento do problema, as funções que o representam e a escolha do método de otimização

Se a função objetivo (ou o conjunto delas) é linear e o espaço de restrições forma uma região contida, resultante da intersecção de um conjunto de semiespaços, o problema pode ser equacionado por equações lineares. Assim, ele pode ser resolvido utilizando-se conhecidas técnicas de Programação Linear, tais como o método Simplex [9].

Se a função objetivo é diferençável (possui gradiente em todos os pontos) e unimodal (as curvas de nível correspondentes a um dado valor $f(x) = \alpha_1$) ) delimitam regiões conexas que contem inteiramente as curvas de nível valor $f(x) = \alpha_2$, sempre que $\alpha_1 > \alpha_2$), e se o problema não possui restrições (otimização irrestrita), os pontos de ótimo desses problemas podem ser determinados de maneira eficiente e com garantia de convergência por métodos tais como os quase-Newton. Esses problemas e seus respectivos métodos formam a chamada Programação Não-Linear [5][13].

Se a função objetivo é côncava (problema de maximização), ou convexa (problema de minimização) e o conjunto de restrições é convexo, então o problema é chamado convexo e métodos gerais de otimização convexa podem ser usados na maioria dos casos [1][2].

Se a função objetivo é quadrática e as restrições são do tipo linear, técnicas de programação quadrática podem ser mais eficientes [10].

Se a função objetivo é a razão de uma função côncava e uma função convexa (no caso de maximização) e as restrições são convexas, então o problema pode ser transformado em um problema de otimização convexa usando técnicas de programação fracional [11].

Vários métodos estão disponíveis para a resolução de problemas não convexos. Uma abordagem é a utilização de formulações especiais de problemas de Programação Linear. Outra abordagem é usar técnicas *branch and bound*, onde o problema é dividido em subclasses para ser resolvido com aproximações convexas (problema de minimização) ou lineares que formam um limite inferior sobre o custo global dentro da subdivisão [12].

Ainda, quando todas as variáveis são discretas, tem a Otimização Combinatória [14].

Os métodos citados são excelentes, quando respeitados as suas limitações quanto a natureza e o comportamento do problema. Dificilmente o Simplex resolverá um problema não-linear, ou um método quasi-Newton encontrará o ótimo global num problema com múltiplos ótimos locais. Como já dito, quando mais próximo a modelagem de um problema se aproxima do mundo real, maior a probabilidade desse equacionamento ter um comportamento não linear e repleto de restrições.

De forma bastante geral, pode-se classificar a Programação Não-Linear em dois grandes grupos: métodos de 'direção de busca' e métodos de 'exclusão de semi-espaços'.

**Métodos de 'direção de busca'**

Estes métodos são baseados na busca sucessiva de pontos no espaço de otimização. Esta procura necessita do conhecimento de um vetor na direção de decrescimento da função, o qual depende do gradiente

da função a ser minimizada (no caso de problemas de minimização). A procura pelo ponto ótimo usa o ponto corrente ($\vec{x}_k$) como ponto de partida para a próxima iteração ($k$+1). Existem muitas maneiras de realizar estas iterações, uma das quais é dada por:

$$\vec{x}_{k+1} = \vec{x}_k + \lambda_k \vec{d}_k \quad , \tag{3.17}$$

onde $\lambda_k$ é o passo de cálculo e $\vec{d}_k$ é a direção de busca do ponto ótimo.

O passo de cálculo controla a evolução da solução. O valor deste passo de cálculo pode ser obtido por métodos do tipo *Golden Section*, *Fibonacci*, dentre outros. Já a direção de busca é responsável pela direção da trajetória até a solução e pode ser determinada por muitos métodos, dentre os quais, o de Fletcher-Reeves, Newton e os algoritmos de Broyden–Fletcher–Goldfarb–Shanno (BFGS) e Davidon–Fletcher–Powell (DFP) [5][7]. O método Fletcher-Reeves utiliza a informação do gradiente da função. Já o método de Newton, que se caracteriza pela rápida convergência, utiliza, além da informação do gradiente da função, o cálculo da matriz Hessiana inversa. Os algoritmos BFGS e DFP usam a mesma metodologia do método de *Newton*, mas substituem a matriz Hessiana por uma aproximação desta. Os quatro métodos anteriores podem utilizar o método *Golden Section* para a obtenção do passo de cálculo. Este consiste em reduzir os limites do universo de busca da função na direção indicada, até que o intervalo formado pelos limites seja menor que um erro admissível.

Com relação ao comportamento do problema a ser otimizado:

- Descontinuidades: a existência de descontinuidades e não-diferenciabilidade nas funções podem causar problemas para o cálculo do gradiente ou de aproximações deste. Em muitos casos, as descontinuidades e não-diferenciabilidades podem até inviabilizar a resolução do problema;
- Não-Convexidade: se a função for unimodal e não possuir descontinuidades, os algoritmos de 'direção de busca' não encontram dificuldades para convergência em funções não-convexas.

- Multimodalidade: sendo garantida a continuidade e a diferenciabilidade, este tipo de algoritmo chegará ao ponto ótimo. Entretanto, nunca se terá certeza se este ótimo é local ou global. É necessário repetir inúmeras vezes o processo de otimização, sempre com pontos iniciais diferentes, de modo a confirmar a solução final.

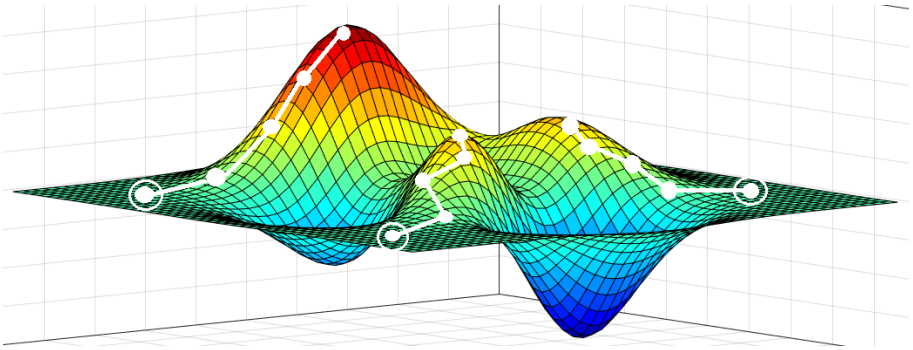
**Métodos de 'exclusão de semi-espaços'**

Métodos de 'exclusão de semi-espaços' são aqueles que utilizam aproximações do gradiente dos problemas para definir um plano que divide o espaço de objetivos em dois semi-espaços, sendo que o gradiente deve decrescer em um dos semi-espaços [7][13]. Fazem parte deste grupo os diversos métodos de 'planos de corte' através de restrições, os métodos de pontos interiores, os métodos elipsoidais etc. Basicamente, estes métodos envolvem três etapas: primeiro calcula-se o 'subgradiente' das funções a serem otimizadas em um ponto; após, divide-se o espaço de busca neste ponto em dois, excluindo um deles; na região restante, faz-se uma nova estimação do ponto ótimo. O processo segue esta rotina até alguma condição de convergência ser atendida.

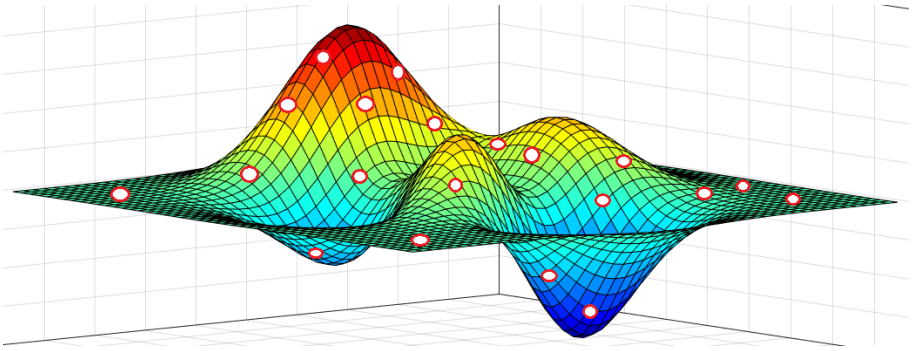
Com relação ao comportamento do problema a ser otimizado:

- Descontinuidades: a existência de descontinuidades e não-diferenciabilidades nas funções não constituem problema e, portanto, não impedem o funcionamento correto dos métodos. No processo iterativo, o próximo ponto não é estimado sobre uma trajetória, mas sim localizado a uma distância finita do ponto anterior;
- Não-Convexidade: a convexidade de todas as funções do problema é o requisito principal destes métodos. Se tal condição for violada, o processo de exclusão trabalhará 'às cegas', de forma que a evolução do método se torna imprevisível;
- Multimodalidade: esta característica pode ser entendida como um caso particular de não-convexidade, assim conclusões são análogas.

O software didático OPTIMAL, apresentado no capítulo 10, possui alguns desses métodos aplicados a diversas funções teste. Com elas, pode-se validar os apontamentos aqui feitos. A Figura 3.2 ilustra (a) três caminhos traçados por método baseado na informação da derivada, localizando máximos locais e o global; (b) uma população de possíveis soluções, estratégia utilizada por métodos da computação evolutiva.



(a) Busca orientada pela informação do gradiente.



(b) População de possíveis soluções, onde a estratégia de busca é orientada pela troca de informações entre elas.

Figura 3.2 Ilustração de estratégia para a busca do ótimo.

Assim, volta-se ao discutido no fim do Capítulo 1, na seção 1.5 sobre o escopo deste livro.

Os algoritmos clássicos de otimização de problemas não-lineares têm sua busca pelo ótimo orientada pelo gradiente das funções que representam o problema, ou aproximações deste – tanto os de 'direção de busca' quanto os 'exclusão de semi-espaços'. A obtenção do gradiente em problemas complexos pode ser uma tarefa árdua ou mesmo impraticável. Outras dificuldades, conforme já discutido aqui, são as descontinuidades, não-convexidade e multimodalidades.

Com o avanço do poder computacional, surgiram os métodos estocásticos, que usam a própria figura de mérito da(s) função(ões) objetivo(s) como estratégia para caminhar em direção do ótimo. Tais métodos constituem uma área de estudo chamado computação evolucionária [15].

Notadamente, as técnicas de computação evolucionária pouco ou nada dependem da natureza do problema, como não linearidades, descontinuidades, não convexidades e multimodalidades. Também, não utilizam a informação da derivada das equações que regem o problema a ser resolvido. Esta independência pode ser consideradas vantagens em problemas complexos. O esforço computacional e as incertezas inerentes a um processo probabilístico podem ser considerados desvantagens.

Todos esses aspectos e os principais métodos de computação evolucionária são apresentados e discutidos no capítulo 4.

**Referências**

[1] J.-B. Hiriart-Urruty and C. Lemaréchal, Fundamentals of convex analysis, Springer, 2001. ISBN 3540422056.

[2] J. Peypouquet, Convex Optimization in Normed Spaces, Ed. Springer, 2015. ISBN 9783319137100.

[3] H. W. Kuhn and A. W. Tucker (1951), Nonlinear Programming, In: proceedings of Berkeley Symposium on Mathematics, Statistics and Probability, University of California Press, Berkeley, 481-492.

[4] A. Izmailov and M. Solodov, Otimização – volume 1, Ed. IMPA, 2014. ISBN 9788524403897.

[5] M. S. Bazaraa, Nonlinear Programming: Theory and Algorithms, Ed. Wiley Publishing, 2013. ISBN 9781118857564.

[6] J. M. Borwein and A. S. Lewis, Karush-Kuhn-Tucker Theory. In: Convex Analysis and Nonlinear Optimization. CMS Books in Mathematics / Ouvrages de mathématiques de la SMC. Ed. Springer, 2000. ISBN 9781475798616.

[7] R. H. Takahashi, Otimização escalar e vetorial, UFMG, 2007. [online] http://arquivoescolar.org/handle/arquivo-e/143

Este capítulo baseia-se no capítulo 3 desta referência, com a autorização do autor.

[8] T. Elser, Factorial Design: Understanding design of experiments (DoE) and applying it in practice, Ed. CreateSpace Independent Publishing Platform, 2017. ISBN 9781542906111.

[9] S. K. Pundir, Linear Programming, Ed. CBS Publishers, 2020. ISBN 9789389396324.

[10] A. Izmailov and M. Solodov, Otimização – volume 2, Ed. IMPA, 2012. ISBN 9788524402685.

[11] S. Schaible, Fractional programming. Zeitschrift für Operations Research 27, 1983. https://doi.org/10.1007/BF01916898.

[12] S. K. Mishra, Topics in Nonconvex Optimization: Theory and Applications, Springer Optimization and Its Applications Book 50. 2011. ISBN 9781441996398.

[13] F. A. Potra and S. J. Wright (2000), Interior-point methods, Journal of Computational and Applied Mathematics. 124 (1–2): 281–302. doi:10.1016/S0377-0427(00)00433-7.

[14] E. F. G. Goldbarg et al., Otimização Combinatória e Meta-Heurísticas, Ed Gen LTC. 2015. ISBN 9788535278132.

[15] C. A. Coello Coello, G. B. Lamont and D. A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd ed., Ed. Springer, 2007. ISBN 9780387332543.

# *Introdução à computação evolucionária*

A melhora na compreensão e na modelagem dos problemas aliado ao incremento do poder computacional, tem permitido solucionar situações até pouco tempo atrás ditas insolúveis. Entretanto, conforme discutido no capítulo passado, a aumento na complexidade dos problemas pode dificultar a aplicação de métodos clássicos. Nestes casos, abordagens heurísticas ou meta-heurísticas ganham cada vez mais espaço.

Procedimentos heurísticos são aqueles que tratam problemas sem há garantia teórica de que a solução ótima exata seja obtida. A lógica de uma abordagem heurística é garantir que a solução obtida, ao final de um processo, seja a melhor quando comparada com outras soluções possíveis.

A computação evolucionária, também chamada computação evolutiva, é uma família de algoritmos heurísticos inspirados na evolução natural ou no comportamento biológico aplicados na resolução de situações modeladas como problemas de otimização. São métodos que fazem parte do que vem sendo chamado de 'inteligência computacional' ou 'computação natural' [1][2].

## 4.1 Sobre inteligência computacional

Inteligência computacional significa conceitos, paradigmas, algoritmos e implementações de sistemas que em teoria exibem comportamentos inteligentes em ambientes complexos [1].

As suas técnicas são inspiradas pela natureza, como a evolução natural e o comportamento coletivo. Esses métodos possuem a premissa de serem tolerantes a conhecimento incompleto, impreciso e incerto do problema, o que facilitaria o encontro de soluções próximas as ótimas, viáveis e robustas – na maioria das vezes.

A Figura 4.1 ilustra as áreas abarcadas pela inteligência computacional. As próximas seções discutem de forma breve cada área. Não é objetivo deste livro tratá-las com detalhes. Aqui, serão discutidos os conceitos gerais. Existe literatura disponível para o seu aprofundamento. Ao final do capítulo, maior ênfase é dada a computação evolutiva.

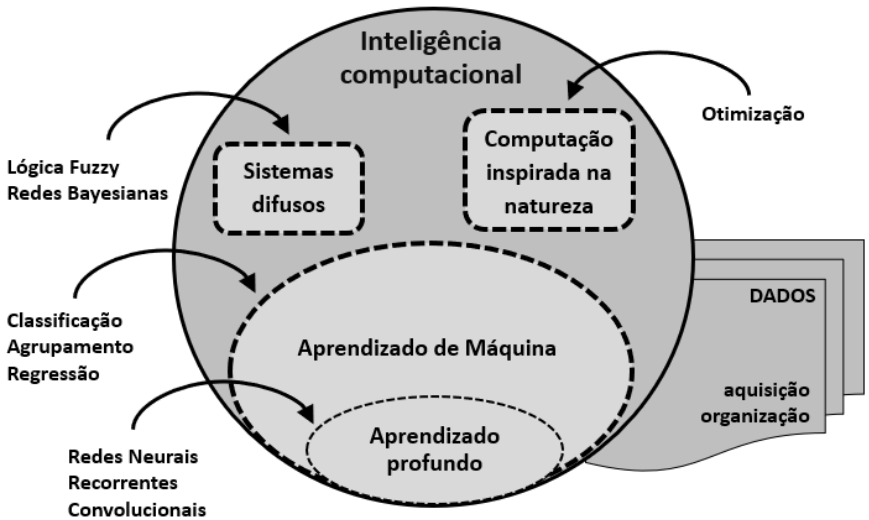



Figura 4.1 Inteligência computacional e suas áreas.

### 4.1.1 Lógica difusa, lógica nebulosa ou lógica fuzzy

Lógica nebulosa (LN) lida com imprecisão semântica (indefinição conceitual) e de ocorrência (verdadeiro/falso, existência) [3].

A lógica clássica, desenvolvida por Aristóteles, estabeleceu um conjunto de regras rígidas baseadas em premissas e conclusões. A lógica binária, aplicada na computação, utiliza dois valores, 0 ou 1, que representam decisões falsas ou verdadeiras, sim ou não, nunca com o mesmo significado.

Uma extensão da lógica binária é a lógica multivalorada. Nela, uma variável pode assumir vários valores ao mesmo tempo, falso e verdadeiro por exemplo. A LN é uma lógica multivalorada capaz de capturar informações vagas, descritas numa linguagem natural, e convertê-las em um formato numérico.

Esta lógica está ligada à importância relativa da precisão. A LN é baseada em palavras e não em números, ou seja, através dela os valores são expressos linguisticamente. Esta lógica utiliza vários modificadores de predicado como, por exemplo: “muito, mais ou menos, pouco, bastante e médio” e um amplo conjunto de quantificadores, como por exemplo: “poucos, vários, em torno de”. O uso de variáveis linguísticas se aproxima do pensamento humano, simplificando a solução de problemas. Isto pode proporcionar um rápido protótipo dos sistemas e simplificar a aquisição da base do conhecimento do problema a ser resolvido.

O diferencial da LN é modelar funções não lineares de complexidade arbitrária, possibilitando a criação de um sistema nebuloso que se combina com qualquer conjunto de dados de entrada-saída.

Um exemplo para aplicação de LN é a percepção humana sobre alimentos. Como garantir que um produto atenda às expectativas de consumidores e se destaque em um setor altamente competitivo como o da indústria alimentícia? Aspectos como aparência e cor (visão), textura (tato), odor (olfato) e sabor (paladar) podem ser tratados pela LN [4].

### 4.1.2 Aprendizado de máquina (*machine learning*)

Um problema de aprendizado considera um conjunto de *n* amostras de dados e tenta prever propriedades de dados desconhecidos [5][6]. Aqui a busca pela solução é baseada em dados, e não pela modelagem matemática do problema. Os problemas de aprendizagem podem ser enquadrados em algumas categorias:

Aprendizado supervisionado, onde existem amostras conhecidas e sabidamente relacionadas de entradas e saídas. O objetivo é estabelecer uma regra que mapeie as entradas para as saídas, conforme as relações previamente conhecidas e ditas adequadas. Duas famílias de métodos:

Classificação – entradas são divididas em duas ou mais classes. O algoritmo deve produzir um modelo que vincula entradas não conhecidas a uma ou mais dessas classes. Alguns métodos são: *support vector machines* (SVM), *nearest neighbor* (kNN) e naive bayes;

Regressão – aqui, ao contrário da classificação, o resultado é uma curva de tendência contínua, não um resultado discreto. São exemplos de métodos: árvores de decisão e redes neurais artificiais;

Aprendizado não supervisionado, onde nenhum tipo de conhecimento prévio é sabido entre a relação entradas e saídas. Aqui, o algoritmo de aprendizado busca padrões de comportamento das amostras.

Clustering – visa fazer agrupamentos automáticos de dados segundo o seu grau de semelhança. O critério de semelhança faz parte da definição do problema e, dependendo, do algoritmo. A cada conjunto de dados resultante do processo dá-se o nome de grupo, aglomerado ou agrupamento (*cluster*). Alguns métodos são: *hierarchical clustering*, *k-means*, e *density-based spatial clustering of applications with noise*.

Aprendizado por reforço, um algoritmo interage com um ambiente dinâmico. É fornecido, ao algoritmo, uma resposta quanto a premiações e punições, na medida em que ele navega no problema. Aqui se destacam a família de métodos probabilísticos de Monte Carlo.

### 4.1.3 Redes neurais artificiais – aprendizado profundo

Redes neurais artificiais (NN) são nós interconectados que funcionam como os neurônios do cérebro humano. Tais algoritmos podem reconhecer padrões e correlações em dados, classificá-los, aprender e melhorar continuamente [6]. Portanto, assim como no aprendizado de máquina, a busca da solução é orientada por dados. Uma rede neural simples inclui uma camada de entrada, outra de saída (ou alvo) e, entre elas, uma camada oculta. As camadas são conectadas através de nós e essas conexões formam uma rede de nós interconectados.

Um nó é modelado conforme o comportamento de um neurônio humano. Os nós são ativados quando há estímulos ou entradas suficientes. Essa ativação se espalha através da rede, criando uma resposta ao estímulo (resultado). As conexões entre esses neurônios artificiais agem como sinapses, fazendo os sinais serem transmitidos de um para o outro.

Quando confrontados com uma requisição ou problema para resolver, os neurônios executam cálculos matemáticos para decidir se há informações suficientes a serem enviadas para o próximo neurônio. Eles leem todos os dados e decidem onde as relações mais fortes estão. No tipo mais simples de rede, as entradas de dados recebidas são somadas e, se a soma for maior que um valor, o neurônio ativa os neurônios conectados a ele.

Conforme o número de camadas ocultas dentro de uma rede neural aumenta, redes neurais profundas são formadas (*deep learning*). Usando *deep learning*, um computador pode ser treinado para emular tarefas humanas como reconhecer fala, identificar imagens ou realizar previsões de comportamento.

Existem muitas metodologias de NN, como por exemplo a *convolutional neural network* para reconhecimento de imagens [8] e *recurrent neural network* para o tratamento de longas séries temporais de dados [9].

Lógica Fuzzy, aprendizado de máquina e redes neurais resolvem problemas complexos, mas não propriamente otimização. Dentro da inteligência computacional, a área dedicada a otimização é a computação evolutiva.

### 4.1.4 Computação evolutiva

A adaptação das espécies ao ambiente natural, por meio do processo evolutivo, ou a busca de alimentos de maneira coordenada por colônia de formigas ou bando de pássaros ou insetos, ou ainda a resposta do sistema imunológico de mamíferos à invasão de agentes patógenos, envolvem essencialmente a realização de buscas em espaços de elevada dimensão, sobre funções que não apenas são de elevada complexidade, mas que também mudam com o passar do tempo, exigindo uma continua execução desses mecanismos de adaptação.

A computação evolutiva (CE) utiliza o princípio da evolução natural e/ou comportamento coletivo para inspirar estratégias computacionais para otimização. São métodos heurísticos ou meta-heurísticos inspirados em mecanismos de adaptação dos seres vivos, conforme observado na natureza [2][10].

Os métodos clássicos de otimização trabalham com uma solução possível que vai melhorando, baseado na informação do gradiente do problema, por exemplo. Em CE, tem-se um conjunto de soluções possíveis (indivíduos) que constituem a população corrente. Através de estratégias aleatórias-orientadas e conhecimento histórico dos resultados anteriores adquiridos, o processo de otimização se guia para a melhora de seus indivíduos.

Por óbvio, existe um alinhamento entre a computação evolucionária e uma área da pesquisa operacional, a que trabalha com técnicas heurísticas estocásticas para a resolução de otimização combinatória.

As definições de ótimo e suas premissas tais como convexidade, diferenciabilidade e unimodalidade, conforme explorado no Capítulo 3, são de pouca relevância para a CE justamente por se trabalhar com

um processo estocástico com múltiplas soluções factíveis (população). Entretanto, nenhum método com característica estocástica garante a obtenção da solução ótima exata.

Para tal, a CE trabalha com a premissa de *localidade fraca* [10]. A propriedade de localidade fraca pode ser definida como a presença de correlação entre os valores assumidos pela(s) função(ões) objetivo(s), sendo tal correlação tanto maior quanto menor for a distância entre os pontos do espaço de variáveis.

Em síntese, para se ter um processo evolutivo que caminhe em direção da melhor solução possível (não se tem certeza de otimalidade), o algoritmo precisa saber quanto uma solução é próxima da outra, tanto no espaço dos objetivos como no espaço das variáveis.

A próxima seção detalha as estratégias de CE mais conhecidas. Na sequência é apresentada em detalhes a implementação de um Algoritmo Genético, a técnica de CE mais popular.

## 4.2 Estratégias de computação evolutiva

As estratégias de computação evolutiva, ou seja, os métodos bio-inspirados mais populares talvez sejam:

1. Algoritmos evolucionários
    a. Algoritmos genéticos
    b. Programação genética
2. Inteligência coletiva
    a. Colônia de formigas
    b. Exames de partículas
3. Sistemas imunológicos artificiais

Existem muitos mais métodos, inclusive com variações hibridas. A intenção aqui é apresentar os conceitos básicos.

### 4.2.1 Algoritmos evolucionários (AE)

A teoria da evolução como é conhecida nos dias de hoje combina genética e seleção natural, sendo Charles Darwin o pesquisador mais conhecido. Pode-se definir genética natural como a diversidade entre indivíduos em uma população de organismos que se reproduzem. Esta diversidade é produzida pela recombinação e pela inserção de material genético novo na população.

Desde Darwin, esta definição vem sendo assimilada e utilizada principalmente no desenvolvimento das áreas que envolvem a biologia e a matemática, através de simulações de sistemas genéticos. Em 1975, o engenheiro eletricista John H. Holland escreveu o livro intitulado *Adaptation in Natural and Artificial Systems* [11], em que aborda diretamente os Algoritmos Genéticos (AGs), o que deu origem ao uso desta técnica para a otimização de sistemas.

Posteriormente a metodologia foi desenvolvida por David E. Goldberg, antigo aluno de Holland. Os estudos de Goldberg foram publicados no seu livro *Genetic Algorithms in Search, Optimization & Machine Learning* [12].

De forma geral, os AEs são algoritmos de otimização estocásticos que trabalham de forma aleatória-orientada de acordo com regras probabilísticas baseadas numa analogia com os mecanismos da genética e seleção natural. Isto é, inicialmente cria-se uma população de indivíduos aptos a ser solução do problema proposto. Realizam-se então reproduções entre os indivíduos, gerando permutações de material genético através de cruzamentos, e insere-se material genético novo através de mutações. Tudo isto respeitando a lei da genética natural que diz que os mais aptos têm mais probabilidade de sobreviver. Com isto vai-se melhorando a população inicial, sendo que os mais aptos correspondem aos indivíduos que obtêm um valor maior (maximização) ou menor (minimização) em uma equação de mérito que representa o objetivo do problema. A Figura 4.3 ilustra um processo evolucionário.

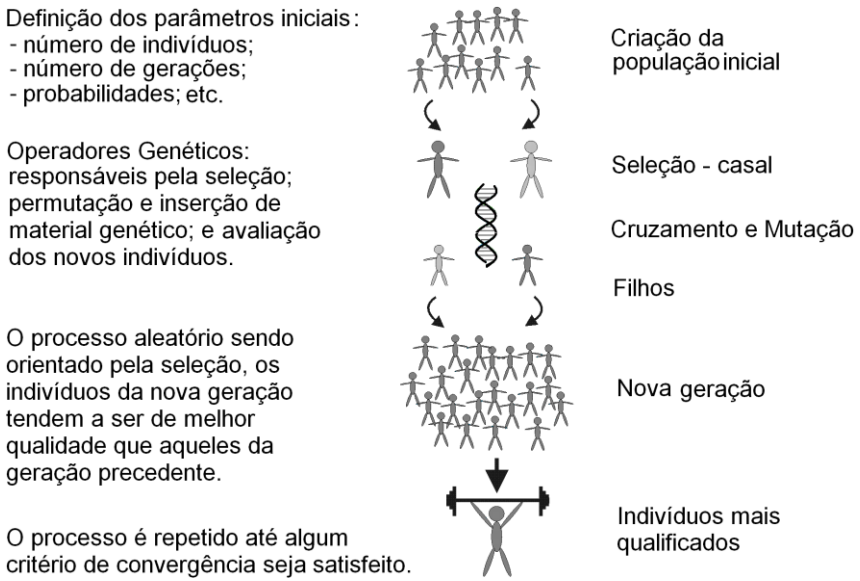



Figura 4.3. Conceitos básicos de um Algoritmo Evolucionário.

Existem muitos métodos que foram desdobrados dos AGs, mas sempre bem semelhantes [13]. Talvez a vertente mais surpreendente seja a programação genética.

Programação genética é uma técnica automática de programação que propicia a evolução de programas de computadores que resolvem problemas. Os indivíduos da população são programas de computador armazenados na forma de árvores sintáticas. Tais programas é que são os candidatos à solução do problema proposto [14].

A seção 4.3 apresenta a implementação completa de um algoritmo genético simples. Porém, antes, são apresentados a inteligência coletiva e os sistemas imunológicos artificiais.

### 4.2.2 Inteligência coletiva

A inteligência coletiva é um conceito de um tipo de inteligência compartilhada que surge da colaboração de muitos indivíduos em suas diversidades. De forma mais específica, a inteligência de enxame (*swarm intelligence*) é aquela encontrada no comportamento coletivo de sistemas descentralizados, auto-organizados, autônomos, flexíveis e dinâmicos [15].

Um algoritmo de inteligência coletiva tipicamente consiste em uma população de agentes simples que interagem localmente entre si e com o ambiente. Esses métodos apresentam capacidades similares as técnicas de aprendizado não-supervisionado.

A capacidade de comunicação entre os agentes permite captar as mudanças no ambiente geradas pelo comportamento de seus pares. Embora não exista uma estrutura centralizada de controle que estabeleça como os agentes devem se comportar, e mesmo não havendo um modelo explícito do ambiente, as interações locais entre os agentes geralmente levam ao surgimento de um comportamento global que se aproxima da solução do problema.

As principais propriedades de um sistema de inteligência coletiva são: proximidade, os agentes devem ser capazes de interagir; qualidade, os agentes devem ser capazes de avaliar seus comportamentos; diversidade, permite ao sistema reagir a situações inesperadas; estabilidade, nem todas as variações ambientais devem afetar o comportamento de um agente; e adaptabilidade, capacidade de adequação a variações ambientais.

Duas principais linhas de pesquisa que emergem dessas propriedades podem ser observadas na inteligência de enxames: trabalhos inspirados no estudo do comportamento de insetos sociais, como formigas, abelhas, cupins e vespas; e trabalhos inspirados na habilidade das sociedades humanas.

Em específico para otimização, as técnicas talvez mais conhecidas de inteligência de enxames são: colônia de formigas, que se baseia no comportamento de formigas em busca de alimentos [16], e enxame de partículas, que busca criar uma simulação do comportamento social humano, particularmente a capacidade humana de processar conhecimento [17].

A Figura 4.4 ilustra o método de otimização por colônia de formigas. Adaptado de Goss et al. [16], a formiga inicia a exploração do ambiente (a) seguindo o sentido (S1) em busca do alimento (A). Ao encontrar (A), ela retorna ao ninho (N) deixando informações ao longo do caminho percorrido (S2) (depósito de feromônio, na vida natural). Demais formigas são recrutadas para a exploração de caminhos até (A), conforme ilustrado em (b). Após algum tempo, a maioria das formigas escolhem a menor rota (c). Eventualmente, algumas formigas são liberadas para explorar o território.

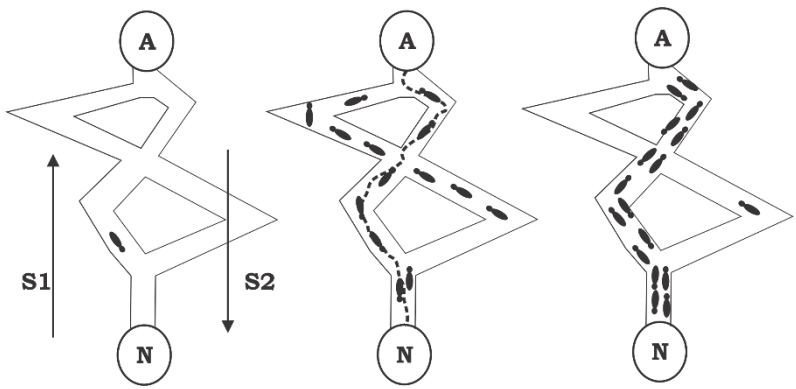



Figura 4.4. Conceitos básicos de inteligência coletiva – colônia de formigas.

### 4.2.3 Sistemas imunológicos artificiais

O sistema imunológico natural é um dos mais importantes componentes dos organismos vivos. Seus mecanismos de reconhecimento e combate a agentes infecciosos externos (patógenos) tem o objetivo de manter o organismo saudável. Em geral, um sistema imunológico deve: reconhecer padrões, detectar anomalias de forma distribuída, tolerar 'ruídos', aprender e ter memória.

O sistema imunológico natural pode ser dividido em duas partes que operam de forma complementar: sistema inato, possui componentes capazes de responder a uma ampla gama de agentes invasores, sem a necessidade de uma exposição prévia a eles; sistema adaptativo, possui componentes que são estimulados em resposta a infecções específicas – imunidade contra reinfecções.

Juntos, os sistemas imunológicos inato e adaptativo contribuem para um mecanismo de defesa extremamente eficiente, que opera em paralelo, recorrendo a uma diversidade de agentes e componentes distribuídos espacialmente e operando em rede.

Sistemas imunológicos artificiais são sistemas adaptativos, inspirados na imunologia natural e em funções e modelos matemáticos aplicados na resolução de problemas [18][19].

Um sistema imunológico artificial básico trabalha com uma população de soluções possíveis, chamadas anticorpos. Os dados do problema são os antígenos, que é toda substância estranha ao organismo que desencadeia a produção de anticorpos. Essa troca de informações, chamada de seleção clonal, se bem orientada, aumenta a concentração das melhores soluções. Soluções similares são eliminadas da população, levando a teoria de rede imunológica. A Figura 4.5 ilustra um pseudocódigo possível.

criar aleatoriamente a população;

**Enquanto** não for atendida alguma condição de parada, **faça**

**Para** antígeno i

Avaliar cada anticorpo na população para *i*;

Selecionar parte dos melhores anticorpos para clonagem – quanto melhor o anticorpo, maior o número de clones;

Aplicar mutação nos clones – variabilidade inversamente proporcional à qualidade do clone;

Adicionar os novos anticorpos à população original e selecionar os melhores;

Substituir os piores indivíduos por novos aleatoriamente gerados.

**fim_Para**

**fim_Enquanto**

Figura 4.5. Pseudocódigo de um sistema imunológico básico.

Os dois princípios fundamentais, seleção clonal e teoria de rede, fazem com que o sistema imunológico artificial apresente algumas características interessantes para a otimização:

- são inerentemente capazes de manter a diversidade da população;
- o tamanho da população a cada geração é automaticamente definido de acordo com a demanda da aplicação;
- e as soluções ótimas locais tendem a ser simultaneamente preservadas quando localizadas.

Estas características fazem com que os sistemas imunológicos artificiais possuam mecanismos eficientes de exploração do espaço de busca.

## 4.3 Algoritmos Genéticos simples

Os Algoritmos Genéticos (AG) têm como princípio a evolução através de gerações de uma população de indivíduos [12][20]. Indivíduos são soluções do problema, ou seja, são pontos dispostos dentro do universo de busca da solução ótima. Um indivíduo (X) pode ser representado da seguinte forma:

$$X = [X_1 \quad X_2 \quad \cdots \quad X_{nvar}] \quad , \tag{4.1}$$

onde $X_1$, $X_2$, $\cdots$, $X_{nvar}$ representam as variáveis que formam o indivíduo, as quais são parâmetros que dependem do problema. O número de variáveis determina a dimensão do espaço de busca.

Um conjunto de indivíduos é chamado de população (P), assim representada:

$$P^n = \begin{bmatrix} X_1^{n,1} & X_2^{n,1} & \cdots & X_{nvar}^{n,1} \\ X_1^{n,2} & X_2^{n,2} & \cdots & X_{nvar}^{n,2} \\ \vdots & \vdots & \vdots & \vdots \\ X_1^{n,nbpop} & X_2^{n,nbpop} & \cdots & X_{nvar}^{n,nbpop} \end{bmatrix} , \tag{4.2}$$

onde *nvar* é o número de variáveis de cada indivíduo, *nbpop* é o número de indivíduos da população e *n* indica a geração corrente.

O número de indivíduos na população pode ser escolhido em função da dificuldade do problema a ser resolvido. Com um número baixo de indivíduos, o universo de busca pode estar sendo representado de maneira muito pobre. Já com um número muito grande de indivíduos, o tempo computacional pode se tornar inviável.

Como ilustração, a Figura 4.6 mostra o universo de busca de um dado problema, em que os indivíduos têm apenas uma variável. Desta forma, o espaço de busca é unidimensional. A função a ser otimizada está representada pelo traço azul e no eixo vertical tem-se o valor da função correspondente a cada indivíduo. Estes estão distribuídos aleatoriamente no universo de busca. O processo de otimização consiste em fazê-los migrar para uma região onde a função é maximizada (ou minimizada).

Tendo-se definido os termos população e indivíduo, bem como o significado das variáveis por indivíduo, pode-se passar à codificação destas variáveis.

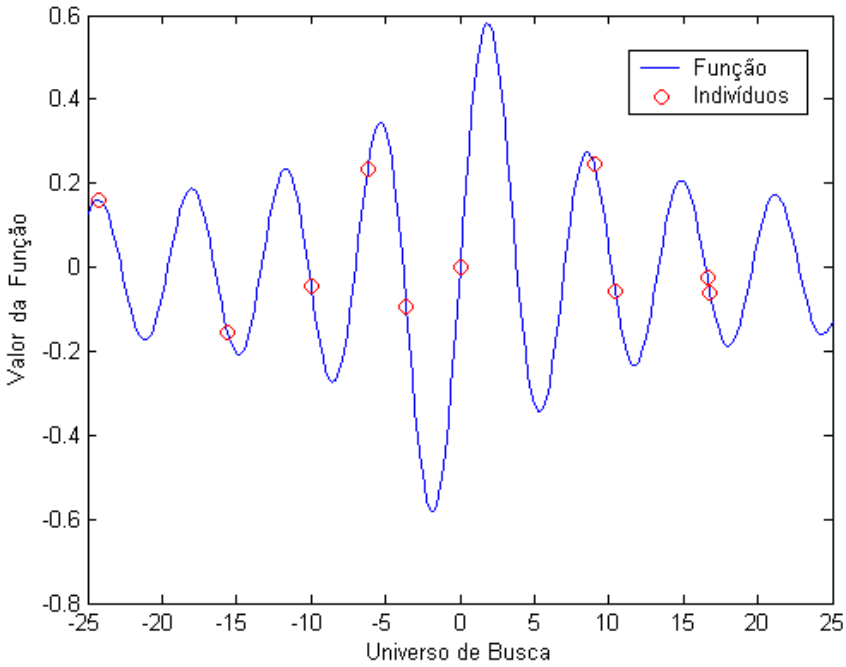



Figura 4.6 Identificação dos indivíduos dentro do universo de busca.

### 4.3.1 Codificação das Variáveis

Existem três tipos de codificação mais utilizados: a codificação binária, a codificação Gray e a codificação real.

**Codificação Binária**

O código binário foi o primeiro a ser explorado por causa de sua analogia direta com a genética natural. Como seu próprio nome diz, este código utiliza números binários, ou seja, apenas conjuntos de 0 e 1 para representar as variáveis. Um indivíduo com codificação binária é representado da seguinte forma:

$$X = [010101 \quad 110100 \quad \cdots \quad X_{nvar}] \quad , \tag{4.3}$$

onde cada variável é representada por um conjunto de bits (genes). O número de bits pode ser diferente para cada variável, estando relacionado à precisão requerida. Por exemplo, se os limites de uma variável estão entre -2 e 2 e a precisão é de quatro casas decimais, tem-se 40 000 divisões. Portanto a variável deverá ter 16 bits ($2^{15}$ = 32 768 ; $2^{16}$ = 65 536) para ser corretamente representada. Se o indivíduo tem 10 variáveis e todas têm a mesma precisão, seria representado por um vetor de 160 bits.

Existem algumas dificuldades em trabalhar com a codificação binária. Uma delas é o fato que, para se ter uma precisão alta, deve-se representar o indivíduo por um vetor bastante extenso. Outro problema é a presença de *Hamming cliffs*, que são grandes diferenças nas cadeias de bits que codificam dois números inteiros próximos [12]. Esta dificuldade fica evidente quando, por exemplo, se realiza uma perturbação nos bits mais significativos da variável. Esta perturbação pode causar um grande deslocamento da variável no universo de busca, o que nem sempre é desejado. Para se evitar este último problema pode-se utilizar o código Gray.

**Codificação Gray**

Na codificação Gray, como na codificação binária, utilizam-se apenas cadeias de 0 e 1 para representar as variáveis. A diferença está na facilidade de operação. Isto ocorre devido à propriedade de semelhança existente na cadeia codificada que representa números inteiros adjacentes. Pode-se esclarecer melhor isto através da tabela 4.1.

Tabela 4.1 – Comparação entre o código binário e o código Gray.

| Números decimais | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| Código binário | 0 | 1 | 10 | 11 | 100 |
| Código Gray | 0 | 1 | 11 | 10 | 110 |

Com a utilização do código Gray, uma pequena taxa de perturbação ajuda na convergência final dos AGs, enquanto que no binário poderia ampliar a região de exploração. Com isso pode-se verificar que o código Gray favorece a precisão da solução, mas pode levar a um ótimo local. Já o código binário se torna mais "livre" para explorar novas regiões e localizar o ótimo global, mas o refinamento da solução torna-se mais difícil.

**Codificação Real**

A codificação real trabalha diretamente com números reais. Isto é muito prático quando se trabalha com variáveis reais por natureza e se usa uma linguagem de programação que lida diretamente com números reais. Entretanto, tal codificação torna os métodos de troca de informações genéticas mais complexas. Como exemplo de indivíduo com codificação real pode-se ter:

$$X = [3{,}3133 \quad 0{,}0001 \quad \cdots \quad X_{nvar}] \quad , \tag{4.4}$$

A escolha de qual codificação utilizar depende principalmente da natureza do problema.

### 4.3.2 Fluxograma de um AG simples

Em todo problema de otimização existe um objetivo a ser alcançado (ou vários), que é representado por uma função objetivo (ou várias). A avaliação desta função permite calcular a aptidão de cada indivíduo. Os AGs procuram sempre melhorar a população, ou seja, buscam os indivíduos de melhor aptidão. Desta forma, quando se quer maximizar uma solução pode-se utilizar a função objetivo diretamente. Já quando se trata de um problema de minimização, tem-se que ajustar a função objetivo. Após este ajuste ela passa a ser chamada de equação de mérito. Como ilustração tem-se o seguinte exemplo: seja um problema representado pela função objetivo dada por (4.5).

$$Função\ objetivo = f(x) = 2x \quad . \tag{4.5}$$

Caso se queira maximizar a função objetivo pode-se escolher a equação de mérito como abaixo:

$$Maximização: equação\ de\ mérito = f(x) = 2x \quad . \tag{4.6}$$

Se, por outro lado, quer-se a minimização da função objetivo, uma escolha possível para a equação de mérito é a seguinte:

$$Minimização: equação\ de\ mérito = m(x) = Cmax - 2x \quad . \tag{4.7}$$

onde Cmax é uma constante de valor elevado.

O fluxograma de um AG simples demonstrando o processo evolutivo pode ser visto na Figura 4.7.

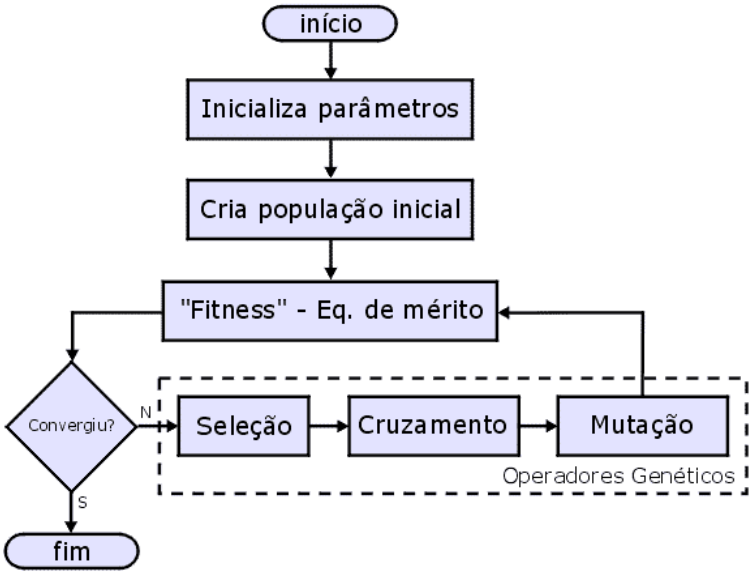



Figura 4.7 – Fluxograma de um AG simples.

Primeiramente especificam-se os parâmetros iniciais (por exemplo, os limites do universo de busca) e cria-se aleatoriamente uma população inicial de indivíduos dentro destes limites. Em seguida verifica-se através da equação de mérito a aptidão de cada indivíduo. Aplicam-se então os operadores genéticos que modificam a população no intuito de melhorá-la. Este processo iterativo, correspondente às sucessivas gerações, prossegue até que se obtenha a convergência (baseada em algum critério pré-estabelecido).

### 4.3.3 Operadores genéticos básicos

O objetivo dos operadores genéticos é transformar a população através de sucessivas gerações, buscando melhorar a aptidão dos indivíduos. Os operadores genéticos são necessários para que a população se diversifique e mantenha as características de adaptação adquiridas pelas gerações anteriores. Na maior parte dos casos, os AGs utilizam três operadores: seleção, cruzamento e mutação.

**Seleção**

Também chamado reprodução, este operador seleciona os indivíduos que sofrerão cruzamento e mutação. Da mesma forma que ocorre no processo de seleção natural, os indivíduos mais qualificados, de acordo com a equação de mérito, têm mais chances de serem escolhidos.

Existem muitos métodos para fazer a seleção. Os principais são: Roleta, Torneio, *Deterministic Sampling* (DS), *Stochastic Remainder Sampling* (SRS) e *Stochastic Universal Sampling* (SUS) [12][20].

A fim de ilustrar o processo de seleção, o método da roleta será descrito. Neste método, cada indivíduo da população é representado em uma roleta proporcionalmente ao seu índice de aptidão (calculado com a equação de mérito). Desta forma, dá-se uma porção maior da roleta aos indivíduos com alta aptidão, cabendo aos indivíduos menos aptos uma porção menor.

Como exemplo, a tabela 4.2 apresenta uma população com quatro indivíduos, seus respectivos valores de mérito calculados de acordo com uma dada equação e os valores percentuais relativos à soma de todos os valores de mérito da população.

Tabela 4.2 Representação do método da roleta.

| | Código | Mérito | % População |
|---|---|---|---|
| Indivíduo 1 | 1010 | 10 | 23,81 |
| Indivíduo 2 | 1011 | 5 | 11,90 |
| Indivíduo 3 | 0011 | 2 | 4,76 |
| Indivíduo 4 | 1001 | 25 | 59,52 |
| | Soma: | 42 | 100% |

Sabendo o quanto cada indivíduo é apto dentro da população, pode-se representá-lo na roleta de forma proporcional, como mostrado na Figura 4.8.

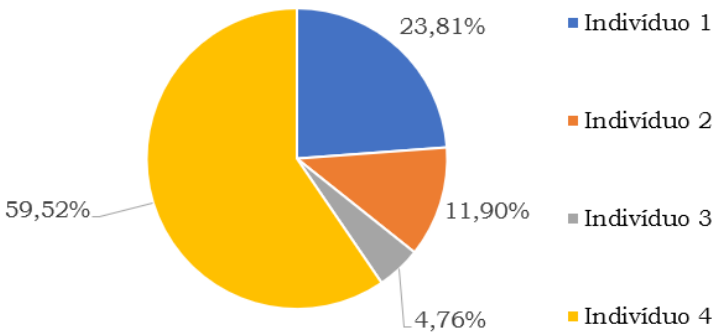



Figura 4.8 Distribuição dos indivíduos na roleta.

A roleta será girada tantas vezes quantas forem necessárias para obter o número requerido de pares de indivíduos para o cruzamento e mutação. O número de pares selecionados a cada geração define os diferentes tipos de AGs. Selecionando o mesmo número de indivíduos que a população inicial, tem-se o tipo SGA (*Simple* GA). Selecionando o número mínimo de indivíduos, ou seja, dois (um par), tem-se o RGA (*Replacement* GA). Qualquer percentual entre o número mínimo ou máximo de indivíduos é denominado SSGA (*Steady State* GA) [12][20].

Esta diferenciação é importante para a determinação do número de gerações, já que o número de pares de indivíduos define a quantidade de avaliações da equação de mérito a cada geração. Fica evidente que, utilizando o RGA, o número de gerações deve ser muito superior ao do SGA, isto se o intuito for manter o mesmo número de avaliações.

Com os pares formados, passa-se aos demais operadores genéticos básicos: o cruzamento e a mutação.

**Cruzamento**

O objetivo do cruzamento é a permutação de material genético entre os pares de indivíduos previamente selecionados. Após a formação dos pares, os indivíduos são submetidos ao processo de cruzamento,

sendo que este processo pode ou não ocorrer, de acordo com uma dada probabilidade de cruzamento (pcross).

Este operador genético é o responsável maior pela criação de novos indivíduos. Por isto pcross deve ser alta (geralmente entre 70 e 100%). Isto é similar ao que ocorre na natureza, onde a maioria dos casais possui filhos.

Os AGs são caracterizados pela alta flexibilidade de implementação, e isto vale também para o cruzamento, que pode ser realizado de diferentes maneiras. A Figura 4.9 ilustra a operação cruzamento.

Pai_1 = [ 1 0 1 0 0 1 1 1 1 1 ]  Pai_2 = [ 1 1 0 1 0 0 0 0 0 1 ]

Filho_1 = [ 1 0 1 0 0 0 0 0 1 1 ]  Filho_2 = [ 1 1 0 1 0 1 1 1 0 1 ]

Figura 4.9 Cruzamento de indivíduos com codificação binária.

Na Figura 4.9 tem-se um par formado pelo Pai_1 e pelo Pai_2, que geram dois filhos. O cruzamento acontece pela troca de informação genética entre os dois pais, como na genética natural. No caso deste exemplo foram trocados os bits seis, sete e oito dos indivíduos (contando-se da esquerda para a direita). Uma infinidade de outros tipos de cruzamento são possíveis.

**Mutação**

Entende-se por mutação a inserção de material genético novo na população. Este processo pode ou não ocorrer, da mesma forma que o cruzamento, de acordo com uma dada probabilidade de mutação (pmut). Esta probabilidade deve ser bem baixa, algo em torno de 0 a 5%, para que a busca pelo indivíduo ótimo não seja puramente aleatória. Isto é análogo ao comportamento da natureza, onde raramente se veem mutações ou anormalidades nos indivíduos. Como no cruzamento, a mutação pode ser feita de muitas maneiras, uma das quais é apresentada na Figura 4.10.

Filho_2 = [ 1 1 0 1 0 1 1 1 0 1 ]  Filho_2_mut = [ 1 1 0 1 0 1 1 0 1 1 ]

Figura 4.10 Mutação de indivíduos com codificação binária.

No exemplo foram invertidos os valores dos bits oito e nove, criando-se um novo indivíduo chamado filho_2_mut.

Após a realização dos operadores genéticos, os novos indivíduos são inseridos na população inicial. Esta inserção pode ser feita também de muitas maneiras. Pode-se citar como exemplo: substituição aleatória e substituição dos pais originais. Assim, uma vez que a população original é alterada, tem-se uma nova geração.

### 4.3.4 Critérios de Convergência

Como dito no início deste capítulo, a convergência acontece de acordo com um critério pré-determinado. Se a aptidão requerida é conhecida, pode-se trabalhar com a opção de um erro máximo admissível. Desta forma, assim que os AGs encontrarem um indivíduo que proporcione um erro menor que o estipulado, finaliza-se o processo.

Outro método interessante de testar a convergência é através da diversidade genética da população. Se os indivíduos estão muito parecidos entre si, ou seja, se a avaliação da equação de mérito de cada indivíduo der resultados muito próximos, pode significar que eles estejam na mesma região. Isto caracteriza a presença de um máximo ou mínimo da função.

Um controle final deve ser feito de maneira obrigatória, pois não se pode ficar simulando indefinidamente. Este controle pode ser realizado, por exemplo, estipulando um número máximo de gerações admissível.

Todas estas metodologias possuem falhas. A convergência por diversidade genética falha quando os AGs convergem para um mínimo local, ou seja, quando acontece convergência prematura. Já o número máximo de gerações falha quando não se dá tempo suficiente ao algoritmo para investigar todo o universo de busca. Uma metodologia

inteligente para ser adotada seria a utilização racional destas duas citadas. Por exemplo, se ao final do processo evolutivo a diversidade genética ainda for elevada, pode-se permitir que o número de gerações seja estendido.

Lembrando que o AG, assim como qualquer outro método probabilístico, não dá a certeza da obtenção da solução ótima. Entretanto, repetindo o processo evolutivo inúmeras vezes, e se na maioria das vezes os resultados finais forem os mesmos, pode-se afirmar que se obteve a melhor solução que o método encontra. Isto remete a premissa de *localidade fraca*, apresentada na seção 4.1.4.

### 4.3.5 Exemplo de Otimização usando AGs

Para exemplificar o que foi visto até o momento, a Tabela 4.4 apresenta um exemplo dos AGs feito manualmente, correspondente à maximização de:

$$f(x) = -x^2 + 16 \quad . \tag{4.8}$$

Estipulou-se o universo de busca como sendo o intervalo [-4 ; 3,5]. Usou-se codificação binária para a otimização desta equação, conforme Tabela 4.3. Desta forma, é necessário fazer a decodificação dos indivíduos a cada verificação de mérito. Neste caso, a equação de mérito foi escolhida como sendo a própria função a ser maximizada. Assim, a aptidão de cada indivíduo é calculada diretamente usando (4.8).

A população inicial foi escolhida aleatoriamente e o método de seleção escolhido foi o da roleta. O cruzamento foi feito através da permutação dos bits dois e três de cada indivíduo. A mutação ocorreu com a simples inversão do bit dois. Estes dois operadores genéticos foram usados respeitando o princípio de que o cruzamento tem uma probabilidade alta e a mutação, baixa. A substituição ocorreu de maneira integral, ou seja, todos os pais são substituídos pelos respectivos filhos. A população foi formada por quatro indivíduos.

Tabela 4.3 Codificação binária para o universo de busca correspondente ao intervalo [-4 ; 3,5].

| Codificado: | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Decodificado: | -4 | -3,5 | -3 | -2,5 | -2 | -1,5 | -1 | -0,5 | 0 | 0,5 | 1 | 1,5 | 2 | 2,5 | 3 | 3,5 |

Tabela 4.4 Exemplo dos AGs feito manualmente.

| Geração 1 | Indivíduo: | Codificado | Decodificado | Mérito | % População |
|---|---|---|---|---|---|
| | a | 0 0 1 1 | -2,5 | 9,75 | 26,00 |
| | b | 1 1 1 0 | 3,0 | 7,00 | 18,67 |
| | c | 0 0 1 0 | -3,0 | 7,00 | 18,67 |
| | d | 0 1 0 1 | -1,5 | 13,75 | 36,67 |
| | | | Soma: | 37,50 | 100% |

| Geração 2 | Seleção | Cruzamento | Mutação | Decodificado | Mérito | % População | Indivíduo |
|---|---|---|---|---|---|---|---|
| par 1 | b 1 1 1 0 | 1 1 0 0 | 1 1 0 0 | 2,0 | 12,00 | 23,41 | a |
| | d 0 1 0 1 | 0 1 1 1 | 0 1 1 1 | -0,5 | 15,75 | 30,73 | b |
| par 2 | d 0 1 0 1 | 0 0 1 1 | 0 0 1 1 | -2,5 | 9,75 | 19,02 | c |
| | a 0 0 1 1 | 0 1 0 1 | 0 1 0 1 | -1,5 | 13,75 | 26,83 | d |
| | | | | Soma: | 51,25 | 100% | |

| Geração 3 | Seleção | Cruzamento | Mutação | Decodificado | Mérito | % População |
|---|---|---|---|---|---|---|
| par 1 | d 0 1 0 1 | 0 1 0 1 | 0 1 0 1 | -1,5 | 13,75 | 23,21 |
| | a 1 1 0 0 | 1 1 0 0 | 1 0 0 0 | 0,0 | 16,00 | 27,00 |
| par 2 | b 0 1 1 1 | 0 1 0 1 | 0 1 0 1 | -1,5 | 13,75 | 23,21 |
| | d 0 1 0 1 | 0 1 1 1 | 0 1 1 1 | -0,5 | 15,75 | 26,58 |
| | | | | Soma: | 59,25 | 100% |

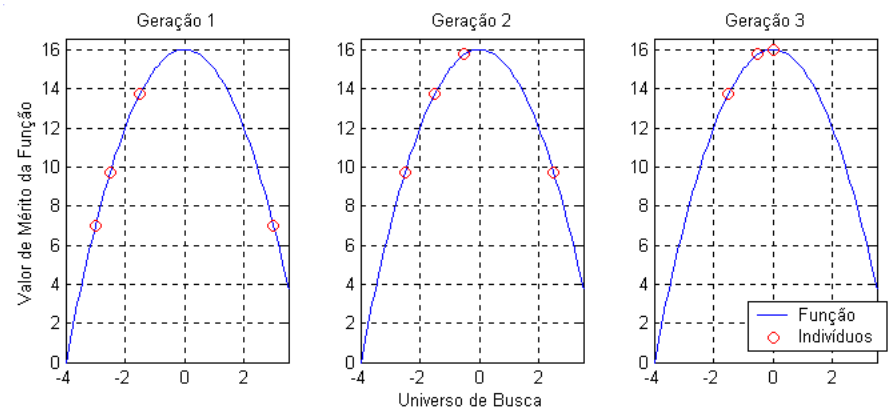



Figura 4.11 Evolução da população conforme Tabela 4.4.

Nota-se que na terceira geração surgiu um indivíduo (x = 0) que alcança o valor máximo na equação de mérito (f(x) = 16). Tem-se, portanto a convergência do algoritmo.

Pode-se ver, através da Figura 4.11, a evolução da aptidão dos indivíduos a cada geração. Partiu-se de uma população aleatoriamente distribuída dentro do universo de busca e, à medida que o processo caminha, vê-se a concentração dos indivíduos em torno do valor correspondente ao máximo da função. Observa-se também no gráfico referente à geração 3 que dois indivíduos são idênticos.

Com este exemplo termina-se a apresentação dos conceitos básicos dos AGs. Como já dito, a abordagem dada até aqui foi simples, com o objetivo de enfocar os aspectos mais relevantes. O capítulo 5 apresenta um AG mais eficiente e mais complexo.

## 4.4 Considerações sobre a computação evolutiva

Quando usar computação natural? As técnicas de computação natural são técnicas alternativas. O ideal é utiliza-las quando os métodos de otimização clássica são de difícil aplicação ou falham.

De forma geral, são vantagens das estratégias aqui discutidas:

- Não requerem um conhecimento matemático profundo do problema ao qual é aplicado;
- Requerem pouco esforço de implementação;
- São facilmente hibridizáveis com outras técnicas;
- São facilmente adaptáveis a muitas classes de problemas, inclusive problemas multiobjetivo;
- São implícita e explicitamente aptas a computação paralela;
- São robustas e capazes de manipular de restrições.

Também de forma geral, são desvantagens das estratégias discutidas:

- Não garantem encontrar a solução ótima exata;
- Há pouco embasamento teórico – a prática se desenvolveu mais do que a teoria;
- O ajuste dos parâmetros de controle requer conhecimento prévio, até por tentativa-e-erro;

Em específico sobre as dificuldades citadas no capítulo 3:

- Descontinuidades: como estes métodos não utilizam nenhuma informação sobre os gradientes das funções, não existe, portanto, qualquer dificuldade relativa à não-diferenciabilidade dos problemas;
- Não-Convexidade: por trabalhar com populações, esses métodos não são influenciados por não-convexidades;
- Multimodalidade: a grande vantagem destes métodos em relação àqueles baseados no gradiente ou em aproximações deste é a possibilidade da detecção de ótimos locais e globais. Os métodos de busca por populações são os que mais se aproximam do conceito de 'algoritmo de otimização para problemas genéricos' (multimodais, com restrições, convexos e não convexos, etc.).

Claro que os métodos de busca por populações têm limitações. A primeira é que resultados obtidos por esses métodos dependem da distribuição da população inicial. Dificilmente o método convergirá se o ótimo global estiver afastado da região onde se concentra a população inicial. Fica então evidente que a eficiência desses métodos depende de uma população inicial que explore bem todo o espaço de busca.

O segundo aspecto negativo é a velocidade de convergência. Comparando com as outras famílias, os métodos que utilizam populações podem ser mais 'lentos', no sentido de que eles podem necessitar de um maior número de avaliações do problema. O esforço computacional desses métodos pode ser maior para atingir os mesmos resultados (quando os outros métodos conseguem obter um resultado).

**Considerações sobre o Problema Multiobjetivo**

A computação natural, por trabalhar com um grupo de soluções (população), torna mais simples a descoberta da fronteira Pareto. Nos outros métodos existe a necessidade de repetir o processo de otimização com restrições ou 'pesos' diferentes para obter uma aproximação da fronteira Pareto-ótima. Isto foi discutido no capítulo 2.

Também é notório que essas metodologias são aptas e lidam de forma bastante simples com grandes e complexos espaços de busca (muitos parâmetros para ajustar, objetivos para alcançar, restrições para obedecer, etc.).

De forma geral, a vantagem mais significativa da utilização de uma ferramenta de busca evolucionária talvez seja o ganho de flexibilidade e adaptação ao problema em questão, combinada a um desempenho robusto e uma característica de busca global.

Este capítulo apresentou uma introdução a computação evolutiva. Um AG simples foi implementado e testado. O próximo capítulo apresenta detalhes para a implementação de um AG mais eficiente. O capítulo 6, por sua vez, apresenta a implementação de um AG multiobjetivo.

**Referências**

[1] J. Fulcher, Computational Intelligence: a compendium, Ed. Springer-Verlag, 2008. ISBN 9783540782933.

[2] L. N. de Castro, Fundamentals of Natural Computing: basic concepts, algorithms, and applications, CRC Press, 2006. ISBN 9781420011449.

[3] T. Nguyen et al. "Fuzzy Control Systems: Past, Present and Future," in *IEEE Computational Intelligence Magazine*, vol. 14, no. 1, pp. 56-68, Feb. 2019. doi: 10.1109/MCI.2018.2881644

[4] S. Kato, N. Wada and T. Kagawa, "Food texture estimation by fuzzy inference," *2017 International Conference on Fuzzy Theory and Its Applications (iFUZZY)*, Pingtung, 2017, pp. 1-6. doi: 10.1109/iFUZZY.2017.8311792

[5] S. Gollapudi and V. Laxmikanth, Practical Machine Learning. Birmingham, UK: Packt Publishing, 2016.

[6] Scikit-learn (2020) Machine Learning in Python. Available at: https://scikit-learn.org/stable/ (accessed 20 April 2020)

[7] J. Patterson and A. Gibson, Deep Learning: A Practitioner's Approach. Ed. O'Reilly, 2017. ISBN 9781491914250.

[8] Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3212-3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.

[9] M. Xu, M. Han, C. L. P. Chen and T. Qiu, "Recurrent Broad Learning Systems for Time Series Prediction," in IEEE Transactions on Cybernetics, vol. 50, no. 4, pp. 1405-1417, April 2020, doi: 10.1109/TCYB.2018.2863020.

[10] A. Gaspar-Cunha, R. Takahashi and C. H. Antunes, Manual de Computação Evolutiva e Meta-heurística, Imprensa da Universidade de Coimbra, 2012. ISBN 9789892605838.

[11] J. H. Holland, Adaptation in Natural and Artificial Systems: An introductory analysis with applications to biology, control, and artificial intelligence, MIT Press, Cambridge, 1992. ISBN 9780262082136.

[12] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Inc., New York, 1989. ISBN 9780201157673.

[13] C. A. Coello Coello, G. B. Lamont and D. A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd ed., Ed. Springer, 2007. ISBN 9780387332543.

[14] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems), MIT Press, 1993. ISBN 9780262111706.

[15] Y. Tan, Y. Shi and B. Niu, Advances in Swarm Intelligence: 10th International Conference, ICSI 2019, Chiang Mai, Thailand, July 26–30, 2019, Proceedings, Part I (Lecture Notes in Computer Science Book 11655) 2019. LNCS 11655.

[16] S. Goss et al. Self-organized shortcuts in the Argentine ant. Naturwissenschaften v. 76, pp. 579–581, May 1989. doi: 10.1007/BF00462870.

[17] B. Walker, Particle Swarm Optimization (PSO): Advances in Research and Applications, Series: Computer Science, Technology and Applications, Nova Science Publishers, 2017. ISBN 9781536108286.

[18] L. N. de Castro and J. Timmis, Artificial Immune Systems: A New Computational Intelligence Approach, Springer-Verlag. 2002. ISBN 9781852335946.

[19] H. Mo, Handbook of Research on Artificial Immune Systems and Natural Computing: Applying Complex Adaptive Technologies, IGI Global, 2009. ISBN 9781605663104.

[20] E. Wirsansky, Hands-On Genetic Algorithms with Python: Applying genetic algorithms to solve real-world deep learning and artificial intelligence problems, Packt Publishing Ltd, 2020. ISBN 9781838559182.

# AG mono-objetivo

O capítulo anterior apresentou uma introdução a computação evolutiva. Um Algoritmo Genético (AG) simples foi implementado e testado. Agora, cabe discutir como melhorar a eficiência e explorar as potencialidades dos AGs.

As ferramentas para melhoria da convergência dos AGs têm dois objetivos principais. O primeiro é evitar a convergência prematura do método. O segundo é acelerar a busca pela solução ótima. Numa primeira análise, tais requisitos parecem ser antagônicos. Mas, na verdade, possuem uma relação de cooperação que determina a eficiência dos AGs.

A convergência prematura é prejudicial porque se pode ficar preso a um mínimo local. Para evitar isto, foram desenvolvidas ferramentas que, à medida em que a diversidade genética diminui e os indivíduos tornam-se muito parecidos, agem causando perturbações na população ou no método de seleção. Surgiram então as técnicas de escalonamento, variação dinâmica de probabilidades, formação de nichos, entre outras.

Acelerar a busca pelo ótimo é necessário, pois os AGs são um método custoso devido ao grande número de avaliações da função de mérito. Para isto, usam-se ferramentas como a redução do espaço de busca e o elitismo, por exemplo.

A Figura 5.1 ilustra um AG mono-objetivo completo. A sequência do texto detalha as sub-rotinas.

## 5.1 AG mono-objetivo completo

```
% Declarações iniciais
G = 1;                          % tipo de AG – SGA= 1 – SSGA = 2 – RGA = 3
ação = 2;                       % ação – minimização = 1 – maximização = 2
função = 'AGMONOPeaks';         % funções – problemas
nbind = 100;     nbgen = 50;    % número de indivíduos e gerações
pcross = 0.90;   pmut = 0.05;   % probabilidade de cruzamento e mutação
limites = [-2 6; -2 6];         % limites do problema
nvar = size(limits,1);          % número de variáveis
seleção = 2;                    % métodos – roleta = 1 – torneio = 2
dap = 1;                        % adaptação dinâmica probabilidades (0=off)
elit = 2;                       % elitismo – simples = 1 – global = 2
pred = 0;                       % redução do espaço de busca (0=off)
fniche = 0;                     % técnica de nicho (0=off)
pniche = 0.1;                   % raio do nicho
indniche = 1;                   % número de indivíduos dentro do nicho
%% Rotina principal – i define o número de vezes que a rotina será executada.
para i = 1:10
    n = 1;                      % geração = 1
    nav = floor(G*nbind);       % número de indivíduos participantes
    Se (nav<=2) nav=2; pcross=1;  fim_se        % se RGA pcross = 1;
    % população inicial
    POPREAL = aleatório(nbind, nvar, limites);  % gera a população inicial
    AVPOP = mérito(POPREAL);                    % avalia população inicial
    SE ação == 1
        amin = 1e6; popAv = amin-popAv;         % minimização
    fim_se
    SE ação == 1
        AV = amin – popAv;
        [Min(n),pos] = min(AV); M = Min;        % melhor indivíduo
        Max(n) = max(AV);                       % pior indivíduo
        Med = sum(AV) / length(AV);             % média
        D(n) = ((1/nbind)*sum((AV-Med).^2))^0.5; % dispersão da população
    Senão ação == 2
        [Max(n),pos] = max(popAv);M=Max;        % melhor indivíduo
        Min(n)=min(popAv);                      % pior indivíduo
        Med=sum(popAv)/length(popAv);           % média
        D(n) = ((1/nbind)*sum((popAv-Med).^2))^0.5; % dispersão população
    fim_se
    Indv(n,:) = popReal(pos,:);                 % salvar o melhor indivíduo
```

```
% processo evolucionário
n=2;
Enquanto n <= nbgen
   Se (ação ==1) mdg=Med/Min(n-1); % medida de diversidade genética
   Senão (ação == 2) mdg=Med/Max(n-1); fim_se
   Se (dap==1) [pcross,pmut] = adapta(mdg,pcross,pmut); fim_se  % DAP
   Se (nav==2) pcross=1; fim_se  % se RGA  pcross=1
   nbpop = size(popReal,1);  popaux = [];      ava = popAv; avaux = [];
   Se (fniche==1)  fp = niche(popReal,popAv,pniche,indniche);    % nicho
   Senão fp = popAv./sum(popAv); fim_se
   Se (seleção == 1)
            [rpopReal,rpopAv] = roleta(popReal,popAv,fp,nav); % roleta
   Senão    [rpopReal,rpopAv] = torneio(popReal,popAv,fp,nav);  fim_se
   rpopReal = cruzamento(rpopReal,pcross,rpopAv);  % cruzamento
   rpopReal = mutação(rpopReal,pmut,limits,n,nbgen); % mutação
   rpopReal = reflexão(rpopReal,pmut,limits,n,nbgen);  % reflexão
   Se (pred ~= 0)  rpopReal = redução(rpopReal, limits); fim_se %redução
   rpopAv = mérito(rpopReal);                          % avalia população
   SE ação == 1
            amin = 1e6; rpopAv = amin-rpopAv;       % minimização
   fim_se
   [popReal,popAv]= ...
   substituição(elit,nbind,rpopReal,rpopAv,Indv,action,Max,Min,amin,n,
   popAv,popReal,fp,fniche); % substituição e elitismo
   SE ação == 1
            AV = amin – rpopAv;
            [Min(n),pos] = min(AV); M = Min;          % melhor indivíduo
            Max(n) = max(AV);                         % pior indivíduo
            Med = sum(AV) / length(AV);               % média
            D(n) = ((1/nbind)*sum((AV-Med).^2))^0.5; % dispersão
   Senão ação == 2
            Max(n),pos] = max(rpopAv);M=Max;        % melhor indivíduo
            Min(n)=min(rpopAv);                       % pior indivíduo
            Med=sum(rpopAv)/length(rpopAv);           % média
            D(n) = ((1/nbind)*sum((rpopAv-Med).^2))^0.5; % dispersão
   fim_se
 fim_enquando
%% fim AG
fim_para   % fim
```

Figura 5.1 AG mono-objetivo completo.

## 5.2 Variação Dinâmica de Probabilidades – Adapta

A variação dinâmica de probabilidades tem como objetivo evitar a convergência prematura. O que esta ferramenta faz é utilizar a medida de diversidade genética da população para medir o grau de semelhança entre os indivíduos. Se o grau de semelhança for alto, alteram-se as probabilidades de cruzamento e mutação (pcross e pmut). Especificamente, reduz-se pcross e aumenta-se pmut, aumentando-se assim a inserção de material genético novo na população. Se a situação for contrária, ou seja, se os indivíduos estiverem muito dispersos, aumenta-se pcross e reduz-se pmut.

Existem inúmeras técnicas para fazer estas variações, como as apresentadas por Srinivas & Patnaik [1]. A Figura 5.2. apresenta o pseudocódigo utilizado.

```
função [pcross,pmut]=adapta(mdg,pcross,pmut)
Km = 1.1; Kc = 1.1;  % constantes de mutação (Km) e de cruzamento (Kc)
Vmin = 0.005;   Vmax = 0.15;
Se (mdg>Vmax)      pmut = Km*pmut;   pcross = pcross/Kc;      fim_Se
Se (mdg<Vmin)      pmut = pmut/Km;   pcross = Kc*pcross;      fim_Se
Se (pcross>1)      pcross=1;         fim_Se
Se (pmut>0.25)     pmut=0.25;        fim_Se
Se (pcross<0.5)    pcross=0.5;       fim_Se
Se (pmut<0.001)    pmut=0.025;       fim_Se
```

Figura 5.2 Pseudocódigo para variação dinâmica de probabilidades.

## 5.3 Formação de Nichos

Na natureza, define-se nicho como uma pequena parte do ambiente onde as populações vivem relativamente isoladas. Por isso, acabam adquirindo características próprias, formando subespécies. Este isolamento pode melhorar o processo de evolução. Nos AGs é possível utilizar o mesmo conceito. Pode-se trabalhar com subpopulações, ocasionando assim o aparecimento e o desenvolvimento de características próprias (e novas). Isto é interessante, pois se estaria

explorando melhor diferentes áreas do universo de busca, aumentando assim o conhecimento a respeito do problema.

Existem técnicas para a implementação desta ferramenta. As mais utilizadas são a função de partilha [2] e o SSS (*Simple Subpopulation Schemes*) [3].

A função de partilha mede o "grau de vizinhança", ou seja, quantifica a proximidade de um indivíduo em relação aos outros no universo de busca. Neste caso, o operador de seleção analisaria o indivíduo por sua aptidão aparente, relativa somente à aptidão de seus vizinhos, ou seja, de uma subpopulação local.

Já o SSS consiste em criar subpopulações desde o início do processo, de modo que cada indivíduo da população receba uma "etiqueta" que indica a qual subpopulação pertence. Da mesma forma que ocorre com a função partilha de Goldberg, os indivíduos são selecionados de acordo com sua aptidão aparente.

A Figura 5.3. apresenta o pseudocódigo utilizado, a qual combina os dois conceitos.

```
função fp = niche(popReal,popAv,pniche,indniche)
[nbpop,nvar] = size(popReal);     qs = pniche;      k = indniche;
norm = abs([max(popReal(:,1))-min(popReal(:,1)) …
        max(popReal(:,2))-min(popReal(:,2))] );
Se (norm(1)==0) norm(1)=1; fim_Se       Se (norm(2)==0) norm(2)=1; fim_Se
fp = popAv./sum(popAv);
Para i=1:nbpop-1
   Se (fp(i)~=0)      nbD=1;
        Para j = i+1:nbpop
                d(i,j) = max( abs(popReal(i,:)-popReal(j,:))./norm );
                Se d(i,j) < qs
                      Se nbD < k  nbD = nbD+1; Senão   fp(j)=0;  fim_Se
                fim_Se
        fim_Para
   fim_Se
fim_Para
```

Figura 5.3 Pseudocódigo para nicho.

## 5.4 Seleção

A seleção forma os pares (pais) que poderão sofrer os demais operadores genéticos. A seleção deve, ao mesmo tempo, garantir a rápida convergência ao ótimo e não conduzir a concentração da população em um só ponto. Assim, o processo de seleção deve trabalhar com dois conceitos 'antagônicos': rápida convergência mantendo diversidade das soluções.

São dois os métodos de seleção aqui utilizados: roleta e torneio [2]. Ambos eficientes, mas que em alguns casos pode conduzir os AGs para a convergência prematura, ou seja, pode-se ficar preso a um máximo (ou mínimo) local.

Isto acontece porque, quando da criação dos indivíduos, geralmente eles possuem um valor de aptidão baixo. Quando entre estes indivíduos aparece um com aptidão muito alta, pode acontecer que muitas cópias dele sejam criadas. Isto é, ele ocupará uma área muito grande na roleta e, consequentemente poderá ser selecionado muitas vezes. Se este indivíduo corresponder a um mínimo ou máximo local, a probabilidade de se ficar preso nesta região será alta.

Para evitar este problema uma saída seria fazer o escalonamento da população, que consiste em limitar o número de cópias de um mesmo indivíduo na próxima geração.

A Figura 5.4. apresenta o pseudocódigo utilizado para (a) roleta e (b) torneiro.

```
função [rpopReal, rpopAv] = roleta(popReal,popAv,fp,nindiv)
nbpop = size(popReal,1);
% pizza
fs(1) = 0;
Para (i=1:nbpop)
    fs(i+1) = fp(i) + fs(i);
fim_Para
fp = fs;
```

```
% seleção
Para n = 1:nindiv
    prob = rand(1);    j = randperm(nindiv);      pos = j(1);
    Para i = 1:nbpop
        Se ((prob >= fp(i)) & (prob < fp(i+1)))   pos = i;  break; fim_Se
    fim_Para
    rpopReal(n,:) = popReal(pos,:);
    rpopAv(n) = popAv(pos);
fim_Para
```

(a) Roleta

```
função  [rpopReal, rpopAv] = torneio(popReal,popAv,fp,nav)
[nbpop,nvar] = size(popReal);
aux = ceil(nbpop/10);       % 10% de nbpop para o torneio
% seleção
Para n = 1:nav
    aux2 = randperm(nbpop);          aux3 = aux2(1:aux);
    av = fp(aux3);                   [a,p] = max(av);
    rpopReal(n,:) = popReal(aux3(p),:);
    rpopAv(n,:) = popAv(aux3(p),:);
fim_Para
```

(b) Torneio

Figura 5.4 Pseudocódigo para seleção.

## 5.5 Cruzamento e Mutação – Codificação Real

Assumindo-se a codificação real como aquela que melhor se adapta à linguagem de programação utilizada e a problemas com grande número de variáveis, pois resulta em vetores e matrizes menores, trabalhou-se no desenvolvimento de operadores genéticos para codificação real que permitam varrer o espaço de busca de maneira mais eficiente.

Para varrer eficientemente o espaço de busca foi desenvolvida uma metodologia modificada para os operadores genéticos cruzamento e mutação. Fundamentalmente, o que se fez foi unir os trabalhos de

Qing et al. [4] e Takahashi et al. [5], além de inserir novos aspectos: a direção para o cruzamento e mutação, bem como a análise do comportamento da população para quantificar a mutação.

No processo evolucionário, agrupam-se os indivíduos em pares e, para cada par, verifica-se a ocorrência do cruzamento ou não, segundo pcross. Se for o caso, a permutação de material genético é feita conforme abaixo:

$$X_{kcross\ldots dir}^{n+1,i} = \alpha_{pol} X_{kcross\ldots dir}^{n,i} + (1-\alpha_{pol}) X_{kcross\ldots dir}^{n,j} \quad , \tag{5.1}$$

$$X_{kcross\ldots dir}^{n+1,j} = (1-\alpha) X_{kcross\ldots dir}^{n,i} + \alpha X_{kcross\ldots dir}^{n,j} \quad , \tag{5.2}$$

onde kcross é um número inteiro aleatório com distribuição uniforme no intervalo $1 \leq kcross \leq nvar$ que define o ponto de corte para a realização do cruzamento; $\alpha_{pol}$ é o coeficiente de multiplicação polarizado, fixado em 0,9 (poderia ser qualquer valor entre 0 e 1); $\alpha$ é o coeficiente de multiplicação aleatório, também com distribuição uniforme entre $-0{,}1 \leq \alpha \leq 1{,}1$ e *dir* é uma variável binária aleatória que indica em qual direção será realizado o cruzamento: se do ponto de corte até a última variável (neste caso, *dir* = *nvar*) ou da primeira variável até o ponto de corte (*dir* = 1). $X_{kcross\ldots dir}^{n,i}$ representa a porção do indivíduo *i* que inclui as variáveis de $X_{kcross}^{n,i}$ até $X_{dir}^{n,i}$ ( ou de *dir* até *kcross* se *dir* < *kcross*). As variáveis que não estão incluídas no intervalo definido acima são copiadas diretamente do progenitor.

Com esta abordagem, que pode ser chamada de cruzamento polarizado modificado, um filho estará muito mais próximo de seu pai de melhor aptidão no universo de busca. Neste caso, para que a população evolua (ou seja, para que o valor de mérito da população aumente a cada geração), é imperativo que o pai deste filho polarizado tenha mérito maior que o segundo pai. Portanto:

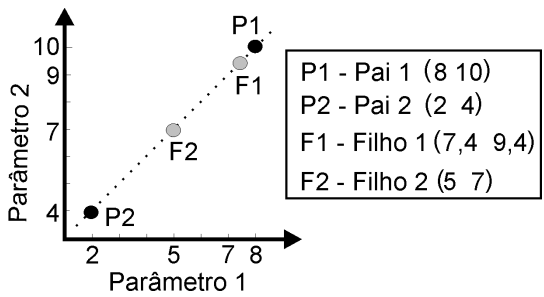
$$f(X^{n,i}) > f(X^{n,j}) \quad , \tag{5.3}$$

Pode-se ilustrar esta operação com um exemplo para melhor compreensão. Sejam os dois indivíduos dados em (5.12). Escolheu-se $\alpha_{pol}$ = 0,9; $\alpha$ = 0,5; *kcross* = 3 (barra vertical) e *dir* = 0, ficando assim o cruzamento determinado pelas duas últimas variáveis de cada indivíduo. Os filhos correspondentes são apresentados em (5.13) e (5.14). Como o cruzamento ocorreu para apenas duas das cinco variáveis, tem-se, portanto, um universo de cruzamento de duas dimensões, conforme Figura 5.5.

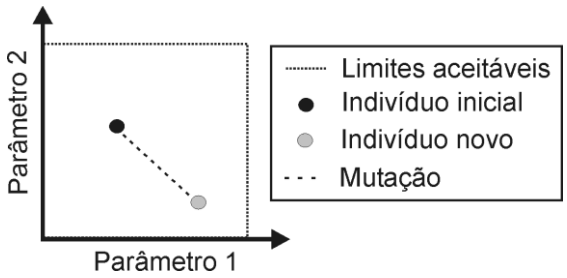
Pai1 = [ 2 4 6 **| 8 10** ] Pai2 = [ 1 3 5 **| 2 4** ] (5.4)

Filho1 = [ 2 4 6 ]∪0,9×[ **8 10** ]+0,1×[ **2 4** ] = [ 2 4 6 **7,4 9,4**] (5.5)

Filho2 = [ 1 3 5 ]∪0,5×[ **8 10** ]+0,5×[ **2 4** ] = [ 1 3 5 **5 7** ] (5.6)




(a) Cruzamento




(b) Mutação

Figura 5.5 Cruzamento real polarizado modificado com duas variáveis.

De maneira similar ao cruzamento faz-se a mutação (que é realizada ou não de acordo com *pmut*). Com codificação real, a mutação consiste em somar ao indivíduo um vetor de perturbação ($\gamma$) dado por:

$$\gamma_{kmut\ldots dir}^{n,i} = 0{,}05\beta\ range_{kmut\ldots dir}^{n,i} \quad , \tag{5.7}$$

onde *kmut* define o ponto de corte para a realização da mutação (determinado aleatoriamente); $\beta$ é um número aleatório com distribuição uniforme no intervalo $0 \leq \beta \leq 1$; *range* é a amplitude da faixa definida pelos limites mínimo e máximo de cada variável e *dir* indica em qual direção a mutação será realizada (do ponto de corte para a direita ou do ponto de corte para a esquerda). Como antes, as variáveis não incluídas no intervalo *kmut...dir* permanecem inalteradas. Desta forma, a mutação corresponde a uma variação máxima de ±5% em cada variável (com relação à amplitude de sua faixa de valores). Esta limitação na amplitude de perturbação garante uma exploração eficiente do universo de busca sem que o processo se torne errático.

No fim do processo evolutivo, o cálculo da mutação é modificado e passa a ser feito conforme:

$$\gamma_{kmut\ldots dir}^{n,i} = 0{,}05\beta \frac{\sum_{i=1}^{nbpop} X_{kmut\ldots dir}^{n,i}}{nbpop} \quad , \tag{5.8}$$

Com isto $\gamma$ passa a depender do valor médio das variáveis que sofrerão mutação. Esta estratégia, que resulta numa diminuição da amplitude das perturbações, permite que a mutação ocorra somente no espaço restrito pela população, condicionando a uma melhor varredura deste espaço. Isto pode permitir a localização mais precisa do ponto de ótimo.

A Figura 5.6. apresenta o pseudocódigo utilizado para (a) cruzamento e (b) mutação.

```
função rpopReal = cruzamento(rpopReal,pcross,rpopAv)
[nbpop,nvar] = size(rpopReal); rest = mod(nbpop,2);
Se (rest==1) nbpop=nbpop-1; fim_Se
pai_1 = rpopReal(1:(nbpop/2),:);
pai_2 = rpopReal((nbpop/2+1:nbpop),:);
pai1 = []; pai2 = [];  limalpha=[-0.1 1.1];
Para i = 1:nbpop/2
    B1 = rand(1);
    Se B1 <= pcross
        B2 = rand(1);       alphapol = 0.9; % alpha 90%
        alpha = (limalpha(2)-limalpha(1))*B2;  % alpha
        Se rpopAv(i) < rpopAv(i+nbpop/2)
                pai1(i,:) = pai_2(i,:);  pai2(i,:) = pai_1(i,:);
        Senão
                pai1(i,:) = pai_1(i,:);  pai2(i,:) = pai_2(i,:);
        fim_Se
        Kcross = round( nvar * rand(1));        % ponto de corte
        Se Kcross==0 Kcross = 1; fim_Se
        dir = rand(1);                                  % direção de corte
        Se dir >= 0.5
                Para j = Kcross:nvar
                    rpopReal(i,j) = alphapol*pai1(i,j) + ...
                                    (1-alphapol)*pai2(i,j);          % filho 1
```

```
                    rpopReal(nbpop/2+i,j) = (1-alpha)*pai1(i,j) + ...
                                            alpha*pai2(i,j);          % filho 2
                fim_Para
            Senão
                    Para j = 1:Kcross
                    rpopReal(i,j) = alphapol*pai1(i,j) + ...
                                            (1-alphapol)*pai2(i,j);   % filho 1
                    rpopReal(nbpop/2+i,j) = (1-alpha)*pai1(i,j) + ...
                                            alpha*pai2(i,j);          % filho 2
                fim_Para
            fim_Se
        fim_Se
    fim_Para
```

(a) Cruzamento

```
função rpopReal = mutação(rpopReal,pmut,limites,n,nbgen)
[nbpop,nvar]=size(rpopReal);
Para i = 1:nvar
    range(i) = limites(i,2)-limites(i,1);        % range
    gam(i) = (sum(rpopReal(:,i))/nbpop);         % behavior
fim_para
Para i = 1:nbpop
    prob=rand(1);
    Se prob<pmut
        Kmut = round( nvar * rand(1));           % ponto de corte
        Se Kmut==0 Kmut = 1; fim_Se
        Bi = rand(1);
        Se n <= nbgen/1.5
                gama(i,:) = 0.05*Bi*range;       % 5% máximo
        Senão
                gama(i,:) = 0.05*Bi*gam;         % 5% máximo
        fim_Se
        dir = rand(1);                           % direção de mutação
        aux = rand(1);                           % adição ou redução
        Se (aux>0.5) sinal=1; Senão sinal=-1; fim_Se
        Se dir >= 0.5
           Para (j=Kmut:nvar)
              rpopReal(i,j) = rpopReal(i,j) + sinal*gama(i,j);
           fim_Para
```

```
        Senão
           Para j = 1:Kmut
               rpopReal(i,j) = rpopReal(i,j) + sinal*gama(i,j);
           fim_Para
        fim_Se
    fim_Se
fim_Para
```

(b) Mutação

Figura 5.6. apresenta o pseudocódigo utilizado para (a) cruzamento e (b) mutação.

## 5.6 Redução do Espaço de Busca

À medida que o número de gerações vai sucedendo e que a população vai melhorando, "caminha-se" na direção do objetivo. Para se encurtar este "caminho", utiliza-se a redução do espaço de busca.

Esta redução é feita do seguinte modo: primeiramente seleciona-se o melhor indivíduo da população corrente. A partir deste indivíduo obtém-se uma nova população fazendo pequenas perturbações aleatórias em suas variáveis, gerando assim novos indivíduos. Com isto, passa-se a explorar somente a região onde está inserido o melhor indivíduo.

Deve-se tomar o cuidado de só começar a fazer as reduções do espaço de busca no final do processo de gerações, quando a população já se organizou em torno do objetivo. Se isto não for respeitado, o risco da convergência prematura será grande [6].

A Figura 5.7. apresenta o pseudocódigo utilizado para redução do espaço de busca.

```
função rpopReal = redução(rpopReal, limites);
Se n==ceil(nbgen/2)   % aplicado na metade do processo evolucionário
        range = limits(:,2)-limits(:,1);
        limitsIni = limits;
```

```
        limits = [Indv(end,1)-range(1)*pred     Indv(end,1)+range(1)*pred ; ...
                  Indv(end,2)-range(2)*pred     Indv(end,2)+range(2)*pred];
        Para j=1:nvar
            Se (limits(j,1)<limitsIni(j,1)) limits(j,1)=limitsIni(j,1); fim_Se
            Se limits(j,2)>limitsIni(j,2) limits(j,2)=limitsIni(j,2); fim_Se
        fim_Para
       rpopReal = rand(nbind,nvar).*repmat((limits(:,2) - ...
                  limits(:,1))',nbind,1)+repmat(limits(:,1)',nbind,1);
fim_Se
```

Figura 5.7. Pseudocódigo para redução do espaço de busca.

## 5.7 Elitismo

Os AGs podem em qualquer momento, devido às suas características probabilísticas, localizar o melhor indivíduo ou simplesmente um indivíduo muito bom. O problema é que este indivíduo pode ser perdido ou destruído pelos operadores genéticos durante o processo evolutivo.

A ferramenta de elitismo visa corrigir este problema. Existem dois tipos de elitismo: o elitismo simples e o elitismo global [2][6].

O elitismo simples guarda sempre o melhor indivíduo que surge, ou seja, aparecendo um indivíduo bom, ele é salvo. Depois de uma geração, se não surgiu um indivíduo melhor que ele, o que se faz é inseri-lo novamente na população.

No elitismo global verifica-se a aptidão dos filhos gerados após a ação dos operadores genéticos. No momento de fazer a substituição leva-se em conta o valor da aptidão de cada um, de forma que serão incluídos na população somente aqueles filhos que melhorarem a aptidão média do conjunto. Obviamente os indivíduos excluídos serão os menos aptos da população.

A Figura 5.8. apresenta o pseudocódigo utilizado para elitismo.

```
função [popReal,popAv]=substituição(elit,nbind,rpopReal,rpopAv,Indv, ...
        action,Max,Min,amin,n,popAv,popReal,fp,fniche)
[rnpop nvar] = size(rpopReal);
Se fniche==0
        Se elit == 0                                    % sem elitismo
            a=randperm(nbind);          b=a(1:rnpop);
            popReal(b,:)=rpopReal;      popAv(b,:)=rpopAv;
        Senão elit == 1                                 % com elitismo
            a=randperm(nbind);          b=a(1:rnpop);
            [v,pos] = min(popAv);       popReal(pos,:) = Indv(n-1,:);
            Se (action==1) popAv(pos,:)=amin-Min(n-1);  fim_Se
            Se (action==2) popAv(pos,:)=Max(n-1);       fim_Se
        Senão elit == 2                                 % elitismo global
            Para i = 1:rnpop
                [Min,pos]=min(popAv);
                Se Min < rpopAv(i)
                    popReal(pos,:) = rpopReal(i,:);     popAv(pos) = rpopAv(i);
                fim_Se
            fim_Para
        fim_Se
Senão fniche==1                                         % nicho com elitismo global
        aux=1; POP=[]; AVA=[];
        Para i=1:length(fp)
            Se fp(i)~=0
                POP(aux,:)=popReal(i,:); AVA(aux,:)=popAv(i);   aux=aux+1;
            fim_Se
        fim_Para
        Se nbind < length(AVA)
            P=POP(1:end-nbind,:);   A=AVA(1:end-nbind);
            popReal=[];     popAv=[];     popReal = P;    popAv = A;
        fim_Se
fim_Se
Se nbind > length(AVA)
        Para k=(length(AVA)+1):nbind
            [v,pos]=max(rpopAv);    POP(k,:)=rpopReal(pos,:);
             AVA(k)=rpopAv(pos);    rpopAv(pos)=0;
        fim_Para
        popReal=[]; popAv=[];   popReal=POP;   popAv=AVA;
fim_Se
```

Figura 5.8. Pseudocódigo para elitismo.

O próximo capítulo discute a resolução de diversos problemas teste com o AG aqui apresentado.

**Referências**

[1]. M. Srinivas and L. M. Patnaik, “Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms,” IEEE - Transaction on Systems, Man and Cybernetics, v. 24, n. 4, Apr. 1994. doi: 10.1109/21.286385.

[2]. D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley Longman Inc., New York, 1989. ISBN 9780201157673.

[3]. W. M. Spears, Simple Subpopulation Schemes, Proceedings of Evolutionary Programming Conference, World Scientific, International Edition, 1994.

[4]. Anyong Qing, Ching Kwang Lee and Lang Jen, "Electromagnetic inverse scattering of two-dimensional perfectly conducting objects by real-coded genetic algorithm," in IEEE Transactions on Geoscience and Remote Sensing, vol. 39, no. 3, pp. 665-676, March 2001, doi: 10.1109/36.911123.

[5]. R. H. C. Takahashi, J. A. Vasconcelos, J. A. Ramirez and L. Krahenbuhl, "A multiobjective methodology for evaluating genetic operators," in IEEE Transactions on Magnetics, vol. 39, no. 3, pp. 1321-1324, May 2003, doi: 10.1109/TMAG.2003.810371.

[6]. J. A. Vasconcelos, J. A. Ramirez, R. H. C. Takahashi and R. R. Saldanha, "Improvements in genetic algorithms," in IEEE Transactions on Magnetics, vol. 37, no. 5, pp. 3414-3417, Sept. 2001, doi: 10.1109/20.952626.

# AG mono-objetivo – testes

O capítulo anterior apresentou em detalhes a implementação de um AG completo. Para avaliar sua a eficácia, quatro funções analíticas foram utilizadas: degrau, picos, rastrigin e rastrigin rotacionada.

Os AGs foram implementados com as seguintes condições:

- População = 20 indivíduos;
- Probabilidade de cruzamento inicial = 90%;
- Probabilidade de mutação inicial = 2,5%;
- Número máximo de gerações = 50;
- Utilizou-se o SGA, ou seja, os operadores genéticos trabalham com um número de pares de indivíduos que representa o tamanho total da população;
- Ferramentas: escalonamento, formação de nichos, variação dinâmica de probabilidades, redução do espaço de busca e elitismo global;
- Operadores genéticos codificados com números reais.

Os demais parâmetros variaram para cada função. As funções teste foram otimizadas 100 vezes com o objetivo de assegurar a validade da resposta alcançada pela repetição do processo.

Estes exercícios podem ser encontrados em:

HTTPS://GITHUB.COM/PECCE-IFSC/OTIMIZAÇÃO

## 6.1 Função Degrau

A função degrau é definida aqui por:

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} abs(\text{int}(x_i)) \qquad (6.1)$$

onde $n$ é o número de variáveis do indivíduo [1]. Optou-se por um universo de busca corresponde ao intervalo $x_i \in [-20,0 ; 20,0]$, constituído por duas variáveis. Neste caso, a função degrau é dada por:

$$f(x_1, x_2) = abs[\text{int}(x_1)] + abs[\text{int}(x_2)] \quad . \qquad (6.2)$$

A Figura 6.1 mostra a função degrau onde, para melhor visualização, somente parte do universo de busca (domínio) é mostrado.

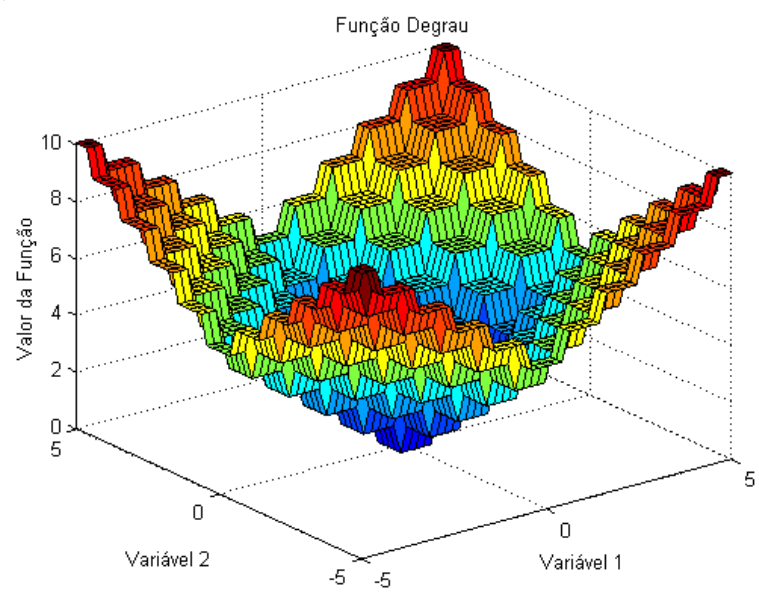



Figura 6.1 Função Degrau para duas variáveis.

Neste caso, o valor ótimo (mínimo global) corresponde não a um único ponto, mas à região definida por $\left(x_1^*\right)^2 + \left(x_2^*\right)^2 < 0{,}5$.

Para a minimização da função degrau, usou-se a seguinte equação de mérito:

$$M = \frac{1}{1 + f\left(x_1, x_2\right)} \quad . \tag{6.3}$$

Pode-se ver a evolução da população para este problema na Figura 6.2, onde a função está representada em suas curvas de nível. Observa-se a concentração dos indivíduos (identificados por x) em torno do mínimo global na sexta geração.

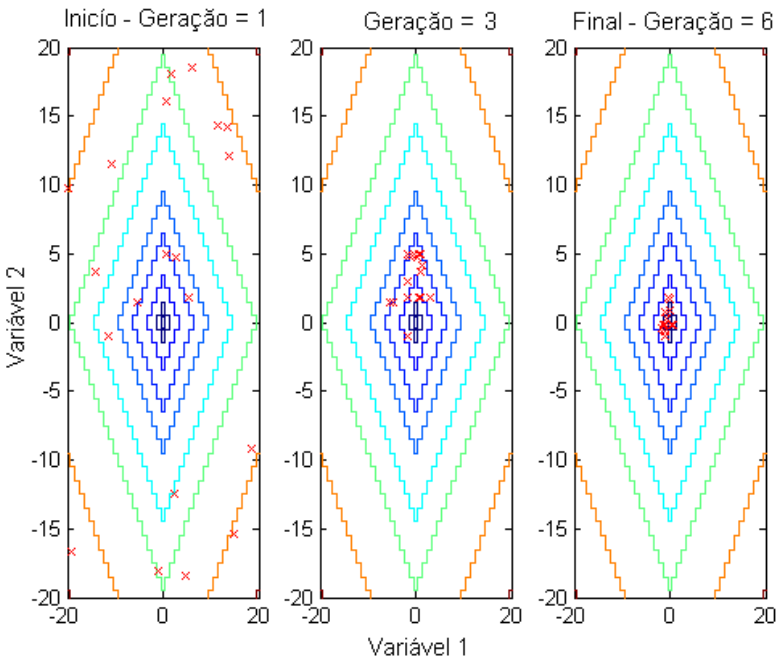



Figura 6.2 Evolução dos AGs para a função Degrau com duas variáveis.

A localização do mínimo da função degrau dificilmente seria obtida com um método determinístico, pois a função é descontínua e possui regiões planas, onde a derivada é igual a zero. Com os AGs, a minimização da função ocorre de maneira eficiente. Foram necessárias apenas cinco gerações, em média, para obter sucesso. Este sucesso é determinado pela localização de um indivíduo que satisfaça o critério de convergência, chamado solução admissível, conforme Tabela 6.1.

Tabela 6.1 Eficiência dos AGs para a função Degrau com 2 variáveis.

| Número de Execuções | Sucesso (%) | Número de Gerações | Solução Admissível |
|---|---|---|---|
| 100 | 100 | $5 \pm 2$ | $\lvert x_i^* \rvert < 0{,}5$ |

Na Figura 6.3 são mostradas todas as soluções encontradas nas 100 execuções. Percebe-se que o algoritmo encontrou a solução ótima em 100% dos casos.

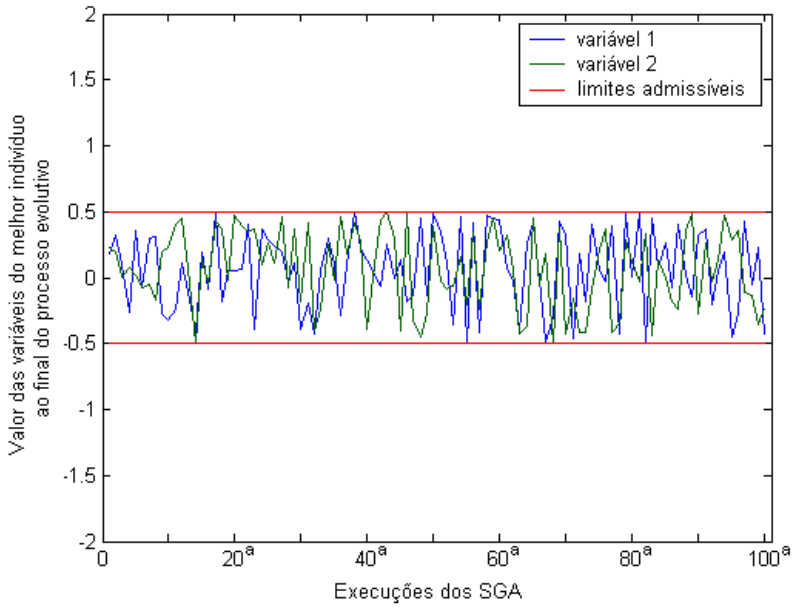



Figura 6.3 Desempenho dos AGs para a função Degrau com duas variáveis.

## 6.2 Função Picos

A função picos de duas variáveis é definida por:

$$f(x_1, x_2) = 3(1-x_1)^2 e^{\left(-x_1^2-(x_2+1)^2\right)} - \left(10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right)e^{\left(-x_1^2-x_2^2\right)} + \frac{1}{3}e^{\left(-(x_1+1)^2-x_2^2\right)}\right)$$

Se o universo de busca for estipulado como: $x_i \in [-3 ; 3]$, conforme Figura 6.4, o ponto máximo global é $(x_1 ; x_2) = (0{,}0094 ; 1{,}5814)$. O que torna essa função como um teste interessante é a sua multimodalidade, tanto se for desejado a sua maximização ou a sua minimização [2]. Sendo o objetivo a sua maximização, a equação de mérito é idêntica à função dada.

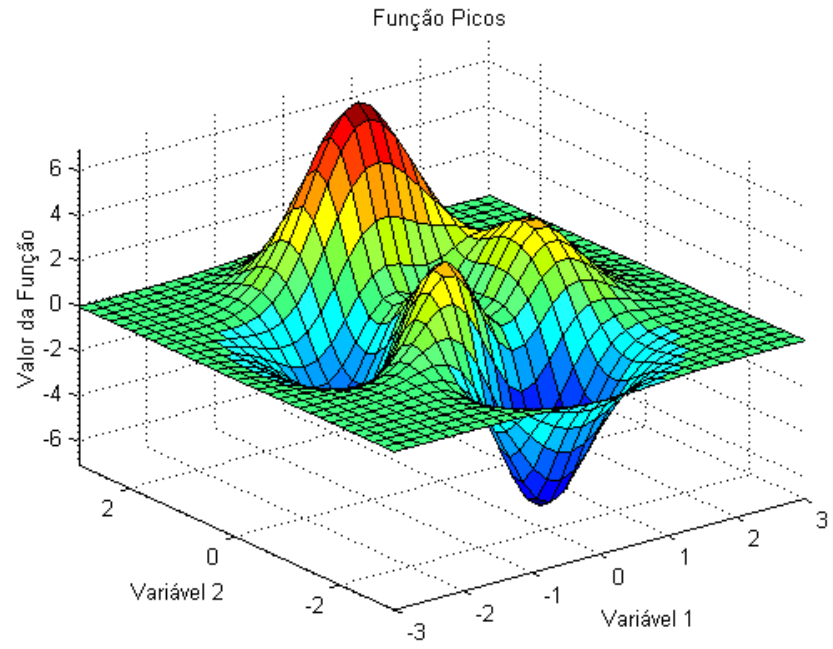



Figura 6.4 Função Picos.

A Figura 6.5 mostra a função em suas curvas de nível assim como a posição dos indivíduos em cada geração. Foram necessárias apenas onze gerações, em média, para obter convergência, conforme Tabela 6.2.

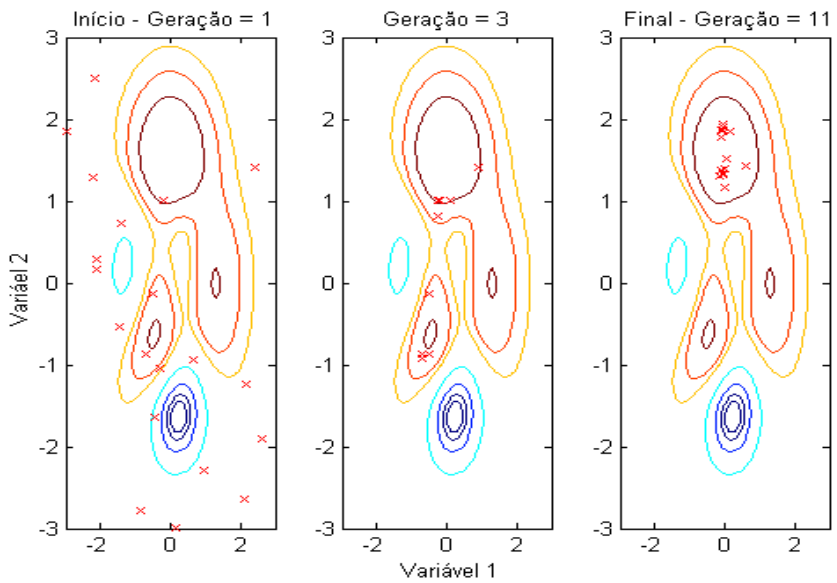



Figura 6.5 Evolução dos AGs para a função Picos.

Tabela 6.2 Eficiência dos AGs para a função Picos.

| Número de Execuções | Sucesso (%) | Número de gerações para convergência | Solução Admissível |
|---|---|---|---|
| 100 | 97 | $11 \pm 2$ | $\sum_{i=1}^{2}\left(x_i - x_i^*\right)^2 < 0{,}02$ |

Na Figura 6.6 são apresentadas todas as soluções encontradas dentro do universo de busca. Percebe-se que o algoritmo encontrou uma solução admissível em 97% dos casos. As três vezes em que não houve convergência ocorreram porque os AGs ficaram presos no segundo máximo da função, localizado em (-0.50 ; -0.64).

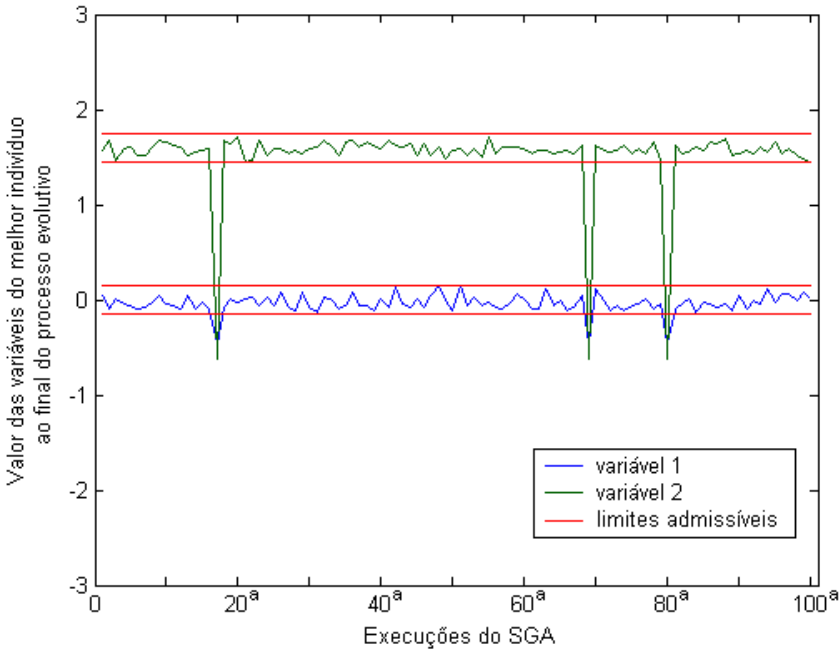



Figura 6.6 Desempenho dos AGs para a função Picos.

A função Picos apresenta três máximos locais bem distintos. Talvez seja interessante a quem otimiza, descobrir os três. Num problema prático, talvez a segunda melhor solução seja mais fácil de implementar ou conceber do que ótimo global.

Remete-se aqui ao conceito chamado Formação de Nichos, discutido na seção 5.3. O AG pode trabalhar com subpopulações, ocasionando assim o aparecimento e o desenvolvimento de características próprias. A Figura 6.7 apresenta os resultados do AG mono-objetivo com formação de nichos para a função Picos.

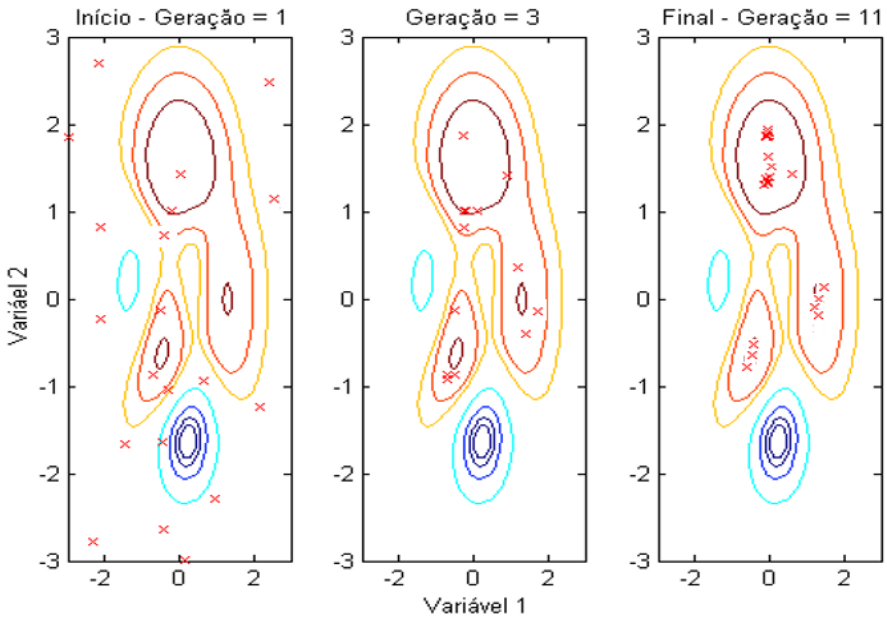



Figura 6.7 Evolução dos AGs com nichos para a função Picos.

### 7.3 Função Rastrigin

A função Rastrigin é definida por:

$$f(x_1, x_2, \ldots, x_n) = 10n + \sum_{i=1}^{n} \left[ x_i^{\,2} - 10\cos(2\pi x_i) \right] \quad , \qquad (6.5)$$

onde $n$ é a dimensão do espaço de busca onde a função é definida [3]. O mínimo global é $x_i^* = 0$. Sendo o objetivo a minimização da função, usa-se uma equação de mérito semelhante a (6.2). Esta função teste tem sua importância devido ao caráter multimodal, não convexa e não linear. Por exemplo, se a amplitude de faixa (range) de cada variável corresponder a dez unidades ($x_{imáx} - x_{imin} = 10$), têm-se $10^n$ mínimos, sendo apenas um global. Aqui, a função Rastrigin foi formada por duas variáveis ($n = 2$) definidas no intervalo [-5,12 ; 5,12], o que corresponde a $10^2$ mínimos, conforme Figura 6.7.

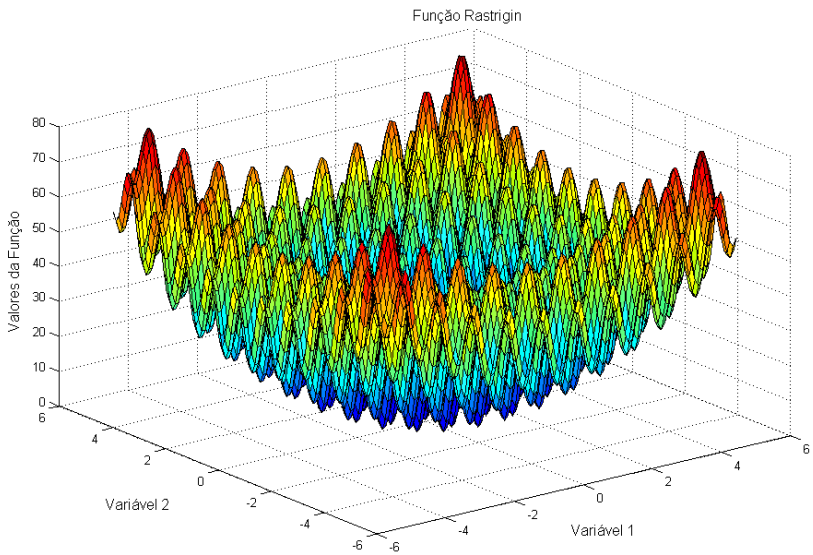



Figura 6.7 Função Rastrigin para duas variáveis.

Pode-se ver a evolução da população para este problema na Figura 6.8, onde a função está representada em suas curvas de nível.

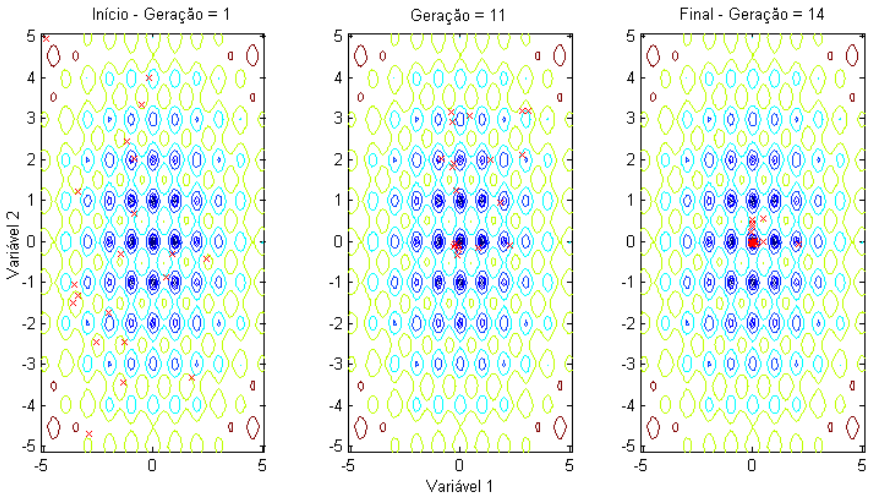



Figura 6.8 Evolução dos AGs para a Rastrigin com 2 variáveis.

Foram necessárias apenas quatorze gerações, em média, para obter sucesso, conforme Tabela 6.3. Na Figura 6.9 são apresentadas todas as soluções encontradas dentro do universo de busca. A percentagem de sucesso é 98%.

Tabela 6.3 Eficiência dos AGs para a Rastrigin com 2 variáveis.

| Número de Execuções | Sucesso (%) | Número de gerações para convergência | Solução Admissível |
|---|---|---|---|
| 100 | 98 | $14 \pm 2$ | $\sum_{i=1}^{2} x_i^2 < 0{,}02$ |

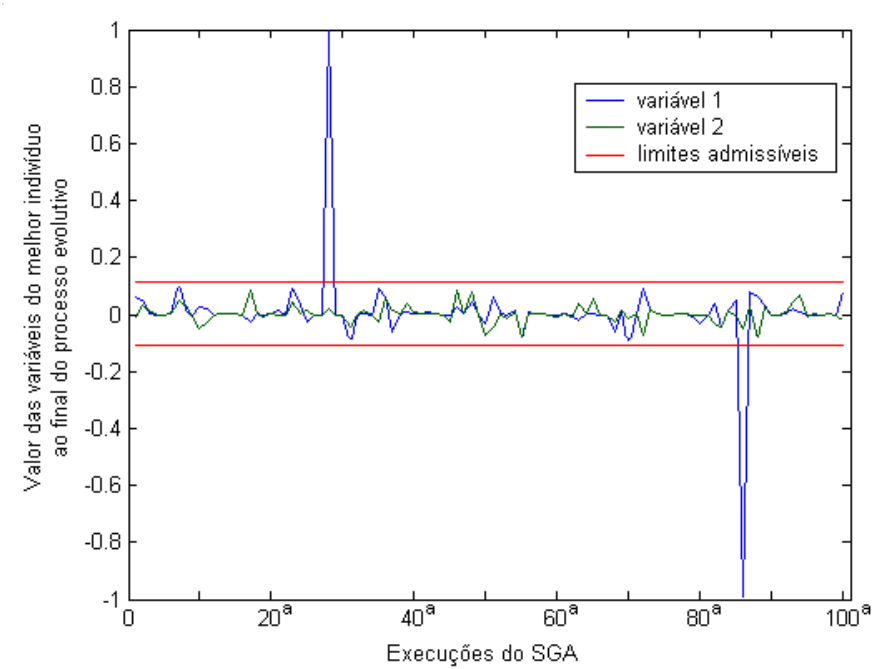



Figura 6.9 Desempenho dos AGs para a função Rastrigin.

## 6.4 Função Rastrigin Rotacionada

Analisando a Figura 6.7, correspondente à função Rastrigin, percebe-se um alinhamento dos mínimos em relação aos eixos das variáveis. Este tipo de disposição facilita o trabalho dos AGs, pois existe a possibilidade do uso da informação dos mínimos adjacentes. Em outras palavras, seria possível localizar uma das coordenadas do mínimo global, mantê-la fixa e ficar procurando pela outra coordenada.

Para que isto não ocorra, pode-se rotacionar a função original, fazendo com que os mínimos saiam dos eixos paralelos aos eixos das variáveis. Com isto introduz-se uma dependência entre os mínimos, o que é mais próximo do que ocorre em problemas reais. Esta rotação pode ser feita através da multiplicação da função Rastrigin por uma matriz de rotação, conforme:

$$\begin{bmatrix} a_x{}^2(1-\cos\theta)+\cos\theta & a_y a_x(1-\cos\theta)-a_z\,\mathrm{sen}\,\theta & a_z a_x(1-\cos\theta)+a_y\,\mathrm{sen}\,\theta \\ a_x a_y(1-\cos\theta)+a_z\,\mathrm{sen}\,\theta & a_Y{}^2(1-\cos\theta)+\cos\theta & a_z a_y(1-\cos\theta)-a_x\,\mathrm{sen}\,\theta \\ a_x a_z(1-\cos\theta)-a_y\,\mathrm{sen}\,\theta & a_y a_z(1-\cos\theta)+a_x\,\mathrm{sen}\,\theta & a_Z{}^2(1-\cos\theta)+\cos\theta \end{bmatrix}$$

onde $\theta$ é o ângulo de rotação desejado e $a = [a_x\ a_y\ a_z]$ é um vetor que representa os eixos de rotação. Por exemplo, quando $a$ = [0 0 1], a rotação ocorre em torno do eixo *z*.

É preciso mencionar que a matriz proposta é válida para um indivíduo contendo duas variáveis, ou seja, para um universo de busca de duas dimensões. Neste caso, o eixo *x* e *y* correspondem às *Variáveis* 1 e 2, respectivamente, e a terceira dimensão (*z*) corresponde ao valor da função (ver Figura 6.7). Para a função Rastrigin com 30 variáveis, a matriz de rotação é 31 × 31 e o vetor de rotação tem a seguinte forma: a = [0 0 0 ... 0 0 1] (31 coeficientes).

Uma comparação entre a função Rastrigin normal e a rotacionada com duas variáveis pode ser vista na Figura 6.10.

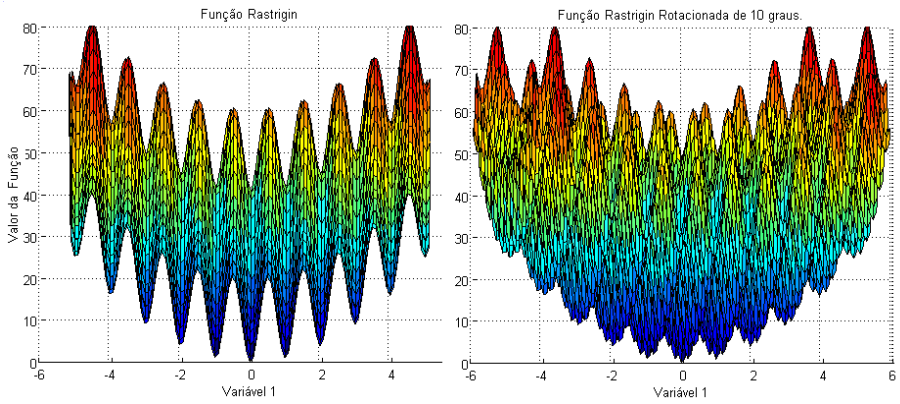



Figura 6.10 Comparação entre a função Rastrigin e Rastrigin rotacionada.

Para dificultar ainda mais o processo de otimização, será utilizada uma função Rastrigin rotacionada com trinta variáveis, ou seja, $10^{30}$ mínimos, sendo apenas um global. Obviamente, não é possível representar graficamente esta função já que o universo de busca das variáveis corresponde a um hiperplano de 30 dimensões.

Os AGs demonstraram ser bastante efetivos também na minimização desta função. Como este problema é muito mais complexo que os demais, é necessário aumentar o número de gerações e de indivíduos. Faz-se isto com o intuito de dar condições aos AGs de poder explorar o universo de busca adequadamente. Para tanto, fixou-se o número de indivíduos em 200 e o número de gerações em 100, o que implica em 20 000 ($200 \times 100$) avaliações da equação de mérito.

A Figura 6.11 mostra a evolução da aptidão do melhor indivíduo e a evolução da aptidão média da população a cada geração. Estas aptidões foram normalizadas em relação a aptidão do melhor indivíduo encontrado ao final do processo evolutivo.

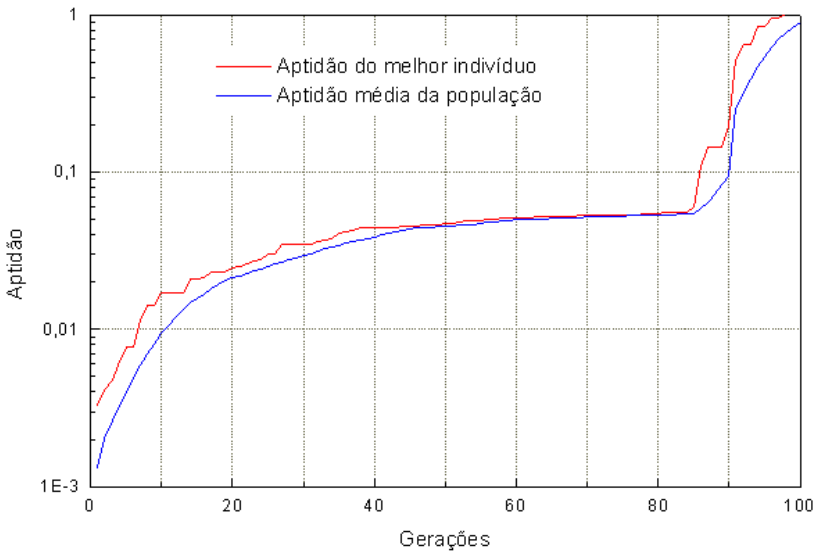



Figura 6.11 Evolução da aptidão dos indivíduos ao longo do processo evolutivo.

Com as curvas apresentadas na Figura 6.11 pode-se visualizar claramente a evolução da aptidão do melhor indivíduo e o aumento gradual da aptidão média da população. Percebe-se que no início do processo evolutivo os valores de aptidão são bem baixos e crescem com a ação dos operadores genéticos (seleção, cruzamento e mutação) a cada geração. Com o passar das gerações a tendência é que a aptidão média da população se aproxime da aptidão do melhor indivíduo, ou seja, toda a população tende a se concentrar na região em torno deste indivíduo.

Ao final do processo evolutivo tem-se a importante atuação da estratégia de redução do espaço de busca. Esta ferramenta para melhoria da convergência reduz o universo de busca a partir do melhor indivíduo encontrado, ou seja, permite explorar melhor uma pequena região em torno deste indivíduo. Isto pode ser melhor visualizado na Figura 6.12, que mostra a distância geométrica no espaço de busca entre o ponto correspondente ao melhor indivíduo e o ponto de ótimo.

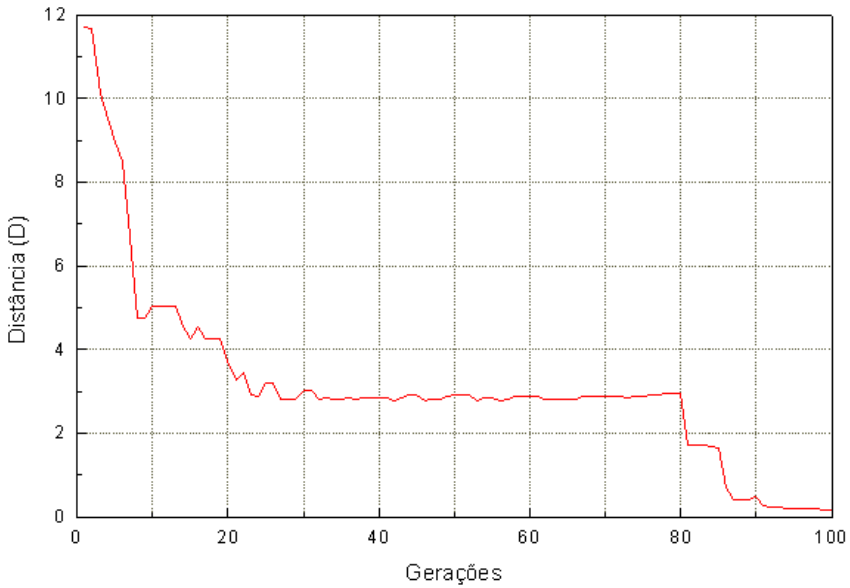



Figura 6.12 Distância entre o ponto correspondente ao melhor indivíduo ao final de cada geração e o ponto de ótimo.

Como o ponto de ótimo possui todas as variáveis iguais a zero ($x_i^* = 0$, $i$=1, ... , 30), a distância no espaço de busca de um indivíduo ao ponto de ótimo pode ser calculada por:

$$D = \sqrt{x_1^2 + x_2^2 + x_3^2 + \ldots + x_{30}^2} \quad , \qquad (6.6)$$

Na Figura 6.12, assim como na Figura 6.11, percebe-se a aproximação do melhor indivíduo do ponto de ótimo. Isto acontece de uma maneira rápida no início do processo evolutivo. A partir de um certo momento (aproximadamente na geração 30, neste caso) o processo tende a se estagnar. Isto ocorre devido à grande complexidade do problema.

No final do processo evolutivo aplicou-se a redução do espaço de busca nas gerações 80, 85 e 90. Esta ferramenta de melhoria na convergência só deve ser aplicada num estágio avançado do processo evolutivo, devido ao perigo da convergência prematura. Assim, é necessário dar tempo aos AGs de varrer o universo de busca e de utilizar as demais ferramentas para melhoria da convergência antes de aplicar a redução do espaço de busca.

Após as reduções do espaço de busca, obtém-se um indivíduo muito próximo ao indivíduo ótimo. Como já mencionado, o número total de avaliações da equação de mérito para a obtenção de uma boa solução foi de 20 000. Não é um número alto se comparado aos $10^{30}$ mínimos da função Rastrigin Rotacionada utilizada neste exemplo.

## 6.5 Conclusões sobre os AGs mono-objetivo

Os AGs, conforme demonstrado aqui em todos os exemplos com funções teste, apresentaram um ótimo percentual de convergência. Isto demonstra a grande eficiência dos AGs, desde que implementados apropriadamente.

Os AGs demonstraram ser uma ferramenta robusta, de fácil entendimento e implementação e bastante versátil, ou seja, de fácil adaptação a uma larga classe de problemas.

Uma importante característica dos AGs vem do fato de trabalhar com uma população de possíveis soluções, o que permite uma certa liberdade quanto à escolha da melhor solução, como por exemplo a formação de nichos.

A principal desvantagem dos AGs é o alto número de avaliações da equação de mérito. Isto fica bastante evidente quando o problema exige, por exemplo, a análise eletromagnética de um dispositivo. Como ilustração, seja a otimização de um problema eletromagnético onde a avaliação da função de mérito envolva a modelagem da estrutura, correspondendo a um tempo de cálculo de um minuto. Considerando

uma população com 20 indivíduos e um processo evolutivo de 30 gerações, o tempo total de cálculo é de 10 horas (sem levar em conta o tempo de processamento dos AGs). Já para as funções teste apresentadas neste capítulo o tempo de avaliação das funções de mérito é ínfimo e a resposta final é quase imediata.

**Referências**

[1] J. A. Vasconcelos, J. A. Ramirez, R. H. C. Takahashi and R. R. Saldanha, "Improvements in genetic algorithms," in IEEE Transactions on Magnetics, vol. 37, no. 5, pp. 3414-3417, Sept. 2001, doi: 10.1109/20.952626.

[2] https://www.mathworks.com/help/matlab/ref/peaks.html

[3] J. G. Digalakis and K. G. Margaritis, "An experimental study of benchmarking functions for genetic algorithms," Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. (cat. no.0, Nashville, TN, 2000, pp. 3810-3815 vol.5, doi: 10.1109/ICSMC.2000.886604.

# AG multiobjetivos

Entre as vantagens dos AGs, podem-se citar: a facilidade para trabalhar com parâmetros discretos ou contínuos (ou com os dois tipos de variáveis simultaneamente); a não necessidade da informação do gradiente da função (as possíveis descontinuidades presentes na função objetivo têm pequeno efeito sobre o desempenho destes algoritmos); eles são resistentes a ficarem presos em ótimos locais; podem lidar com um grande número de parâmetros e são bem apropriados à computação paralela; e, por trabalharem com uma população de soluções possíveis, eles geram uma lista de soluções "semi ótimas" em lugar de uma única solução. Esta última talvez seja a vantagem mais relevante no trato de problemas multiobjetivos, pois a fronteira Pareto vai se formando naturalmente.

Conforme Veldhuizen & Lamont em *Analyzing the State-of-the Art* [1], o primeiro trabalho de AG para resolver problemas multiobjetivos foi proposto por Schaffer [2] em 1984. Desde então, o número de publicações nesta área tem crescido exponencialmente. Coello [3] apresenta uma revisão classificando e avaliando inúmeras técnicas de otimização multiobjetivos. Este estudo concluiu que a maioria dos métodos é baseada nos AGs mono objetivo. A diferença está na proposta de seleção dos indivíduos. Os principais procedimentos, segundo uma ordem cronológica de publicação, são:

AG baseado em Vetor de Avaliação (*Vector Evaluated Genetic Algorithm* – VEGA) [4]. Modifica-se o operador genético seleção de um AG mono-objetivo de modo a criar populações separadas para cada objetivo. Isto acaba gerando 'especializações', ou seja, cada população tenderá ao

ponto ótimo para aquele objetivo e não para a fronteira Pareto-ótima associada ao problema;

AG Multiobjetivo (*Multiobjective Genetic Algorithm* – MOGA) [5]. A ideia do MOGA é estabelecer uma ordem dos indivíduos: os não-dominados têm classificação igual; já os dominados são penalizados de acordo com sua dominância. A dificuldade está em encontrar uma forma de interpolar estes dois grupos de modo a permitir uma boa conformação da fronteira Pareto-ótima;

AG baseado em Ordenação Não-Dominada (*Nondominated Sorting Genetic Algorithm* – NSGA [6]. Somente as soluções não-dominadas são selecionadas. Por todas serem eficientes, terão a mesma probabilidade para se reproduzir. Aqui, a desvantagem é a dificuldade em conseguir manter a diversidade da população. A falta de diversidade pode gerar uma fronteira incompleta, ou seja, a concentração de soluções em algumas regiões;

AG baseado em Pareto Dominante (*Niched Pareto Genetic Algorithm* – NPGA)[7]. Uma das técnicas de seleção para AG monoobjetivo é o 'torneio' entre os indivíduos. Horn & Nafpliotis implementaram um torneio em que a regra de competição é a ideia de dominância por Pareto. Neste método existe a dificuldade de determinar quais e quantos indivíduos participarão do torneio;

Método das Populações Intermediárias [8]. O método segue três passos básicos: a determinação dos pontos mínimos de cada objetivo; a procura de uma população intermediária (baseada na escolha de n indivíduos para cada objetivo); e, a partir desta população, a definição da fronteira Pareto-ótima. A principal restrição desta metodologia é que cada objetivo deve haver um único ótimo no espaço de estudo (função unimodal);

Algoritmo evolucionário baseado na 'força' de Pareto (*Strength Pareto Evolutionary Algorithm* – SPEA) [9]. Um escalar indicando uma medida de 'força' para cada indivíduo é criado para o processo de seleção. Os indivíduos não-dominados devem possuir maior 'força'.

## 7.1 Características para um AG multiobjetivo eficiente

De maneira geral, um algoritmo evolucionário é caracterizado por:

(1) um conjunto de soluções candidatas é submetido;

(2) a um processo de seleção; e

(3) as soluções escolhidas são manipuladas por operadores genéticos com a intenção de melhorar este conjunto.

Devido ao seu inerente paralelismo, os algoritmos evolucionários têm o potencial de encontrar múltiplas soluções Pareto em uma única iteração. Entretanto, em aplicações complexas, nem sempre é possível obter soluções ótimas, muito menos um conjunto Pareto-ótimo completo. Lembrando que a noção de ótimo é uma idealização na maioria dos problemas reais.

Consequentemente, o principal alvo da otimização de problemas multicritério pode ser reformulado, consistindo de três preocupações:

- A distância entre a fronteira não-dominada resultante e a fronteira Pareto-ótima deve ser minimizada;

- Uma boa distribuição (uniformidade) das soluções encontradas é desejável. A ausência de soluções em partes da fronteira pode dificultar a escolha final; e

- O espalhamento da fronteira não-dominada deve ser maximizado. Isto é, valores extremos para cada objetivo devem ser alcançados, permitindo assim, um melhor entendimento do compromisso entre os objetivos.

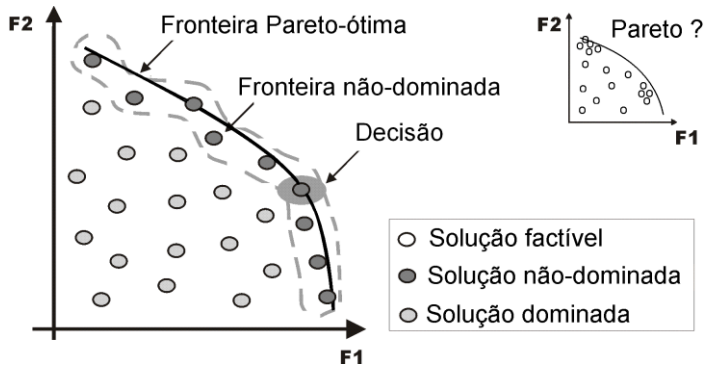



Figura 7.1. Princípios da otimização multiobjetivo.

Além destes três preceitos, ilustrados na Figura 7.1, existem duas preocupações principais quando se aplica um algoritmo evolucionário na resolução de um problema de otimização multiobjetivo:

- Como intermediar a informação dos méritos de cada indivíduo com o processo de seleção de maneira a guiar a busca pelo conjunto Pareto-ótimo; e
- Como manter a diversidade da população de modo a prevenir uma convergência prematura, e assim garantir a obtenção de uma fronteira não-dominada ampla e uniformemente distribuída.

Para dirimir estas dificuldades, procedimentos (operadores genéticos) são empregados. Os mais utilizados e aqui implementados são:

- Seleção: forma os pares (pais) que poderão sofrer os demais operadores genéticos. A seleção deve, ao mesmo tempo, garantir a rápida definição da fronteira Pareto e não conduzir a concentração da população em um só ponto. Assim, o processo de seleção deve trabalhar com dois conceitos 'antagônicos': rápida convergência mantendo diversidade das soluções;
- Cruzamento: geração de novos indivíduos por permutação de 'informações' dos pares. Responsável maior pela exploração do espaço de busca;

- Mutação: inserção aleatória de 'informações' novas nos indivíduos, o que aumenta a diversidade da população;
- Nicho: capacidade de explorar simultaneamente regiões distintas, descobrindo ótimos locais e/ou globais;
- Espaçamento: permite eliminar da fronteira não-dominada as soluções muito próximas umas das outras;
- Reflexão: as variáveis devem a respeitar os limites impostos;
- Elitismo: manutenção das 'boas' soluções no processo evolutivo;
- Redução do Espaço de Busca: diminuição do espaço dos parâmetros de acordo com informações obtidas da fronteira não-dominada.

A seguir é apresentada uma implementação particular de um Algoritmo Genético Multiobjetivo (AGMO). Buscou-se um maior equilíbrio entre o espaço de parâmetros e o espaço de objetivos, a fim de melhorar a eficiência do método e facilitar os estudos de sensibilidade das soluções. Este equilíbrio é explicado no detalhamento de cada procedimento, após a apresentação da estrutura do AGMO. Obviamente, aumentar a confiabilidade da fronteira Pareto e diminuir o tempo para convergência são também preocupações.

## 7.2 Algoritmo Genético Multiobjetivo: Três populações correntes

O AGMO proposto aqui é baseado em três populações correntes. A Figura 7.2 apresenta esta metodologia. O algoritmo inicia como o AG mono-objetivo, ou seja, são estipuladas as especificações iniciais do processo (probabilidades de cruzamento e mutação, tamanho da população e número máximo de gerações). Devem ser previamente conhecidos: o número de variáveis de cada indivíduo, os limites aceitáveis de cada variável e o número de objetivos que serão abordados. Todos os procedimentos aqui apresentados foram desenvolvidos para a codificação real. Isto significa que os parâmetros (variáveis de otimização) são números reais, não necessitando de qualquer codificação, como por exemplo, a transformação para números binários.

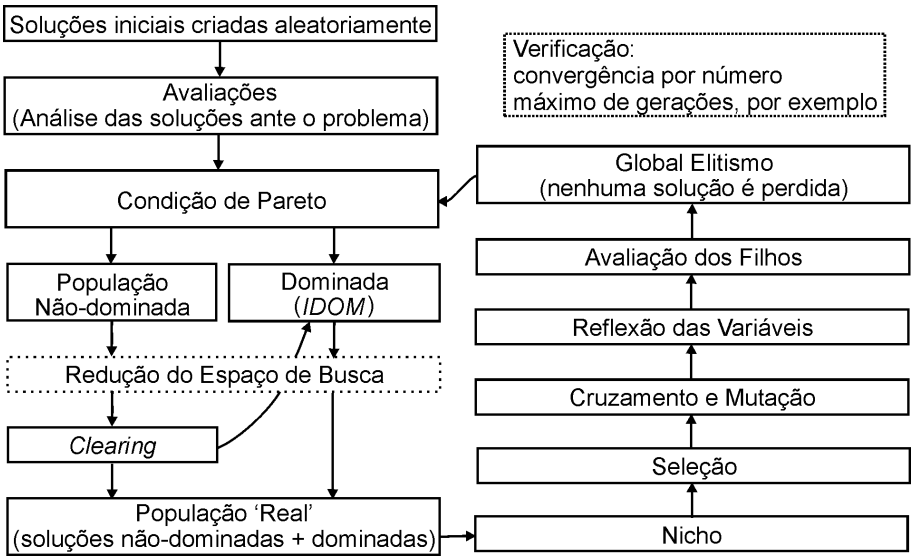



Figura 7.2. AG Multiobjetivo: três populações correntes.

Conhecidas essas informações iniciais, o primeiro passo é criar a população inicial de possíveis soluções do problema. Estes indivíduos são gerados aleatoriamente dentro dos limites pré-estabelecidos. Estas soluções são avaliadas perante o problema e a condição Pareto-ótima é testada, conforme capítulo 2 e 3.

Resumidamente:

$$P = \{ \quad \textit{não deve existir } \vec{x} \in X_f \ |$$

$$\text{(1) exista } i \text{ tal que } f_i(\vec{x}) > f_i(\vec{x}^*) \text{ e} \qquad (7.1)$$

$$\text{(2) } \forall\ j \neq i \quad f_j(\vec{x}) \geq f_j(\vec{x}^*) \quad \}$$

onde $X_f$ é o espaço factível. O conjunto $P$ contém as soluções eficientes ($\vec{x}^*$) do problema.

A verificação (ou a não verificação) da condição de Pareto (7.1) separa a população em dois grupos de soluções: um formado pelas soluções não-dominadas (*POPNDOM*); e outro por soluções dominadas (*POPDOM*).

Um índice (*IDOM*) que indica por quantas vezes cada solução é dominada por outras é criado. Isto é importante para que o processo de seleção trabalhe somente com as soluções que estão na região próxima ao conjunto Pareto, o que pode acelerar a convergência.

Após esta verificação, pode-se aplicar uma técnica de 'espaçamento' (*clearing*), cujo propósito é obter uma repartição regular dos indivíduos sobre a fronteira Pareto. Se similaridades entre os indivíduos são detectadas (no espaço de parâmetros e/ou no espaço de objetivos), algumas destas soluções podem ser retiradas de POPNDOM.

Para melhor controle de todo o processo evolucionário, um número fixo de indivíduos (nbind) é usado nas ações de cruzamento e mutação. Por óbvio, o relevante é o número de avaliações do problema. Ou seja, se o número de indivíduos for baixo, deve-se compensar aumentando o número de gerações. O contrário também é válido.

Este grupo de tamanho mínimo fixo é chamado de 'população real' (*POPREAL*), o qual é recriado a cada geração. *POPREAL* é sempre composto por todas as soluções de *POPNDOM* (após *clearing*) somadas a *nbind*/4 soluções de *POPDOM* (escolhem-se as que possuem os menores índices *IDOM*), de maneira a manter certa diversidade. Se o número de indivíduos de *POPREAL* for ainda menor que *nbind*, ela é completada com mais indivíduos de *POPDOM* (escolhidas outras que tiverem os menores índices). O contrário, ou seja, o caso em que o número de indivíduos de *POPREAL* for maior que *nbind*, será tratado no processo de seleção.

Quando for necessário, uma técnica de nicho pode ser executada após a montagem de POPREAL. Esta técnica permite a exploração de regiões distintas contendo ótimos locais. Isto é possível através de uma transformação da função objetivo: os méritos são trocados por índices de semelhanças, nos espaços de objetivos e no espaço de parâmetros.

O número de indivíduos selecionados de POPREAL é sempre nbind. A seleção é feita pela ação conjunta de dois procedimentos. Os pais são em parte escolhidos por amostragem determinística (baseada na média dos méritos da população, o que aumenta a possibilidade de seleção de indivíduos da parte central da fronteira não-dominada) e em parte por Torneio (baseado em cada objetivo individualmente, facilitando a escolha de indivíduos dos extremos da fronteira não-dominada).

Após a seleção, os operadores de cruzamento e mutação são executados. Geração após geração, estes operadores criam novos indivíduos (filhos) baseados nas informações contidas nos indivíduos correntes (pais), de maneira a explorar eficientemente o espaço de busca. A geração destes novos indivíduos pode não respeitar as restrições sobre as variáveis (dimensões construtivas máximas, por exemplo). Neste caso, é necessário ajustar: ou os novos indivíduos são modificados de modo a respeitar os limites ou estes são redefinidos (quando isto for possível).

Os novos indivíduos são avaliados e diretamente inseridos em *POPREAL*. Então, *todas* as soluções (correntes e novas) são submetidas à condição de Pareto (7.1). Isto resulta em uma *POPNDOM* modificada, cujo tamanho varia a cada geração (aumenta e diminui), enquanto que *POPDOM* pode somente aumentar. Como nos AGs mono objetivo, novas 'boas' soluções podem aparecer em qualquer momento do processo. Entretanto, estes indivíduos podem ser perdidos durante o processo evolucionário. É o conceito de elitismo global quem garante a permanência das soluções eficientes. No AGMO apresentado aqui, o elitismo global é implicitamente incorporado devido ao uso de todas as soluções não-dominadas (*POPNDOM*) para compor *POPREAL*. Finalmente, o processo evolucionário é reiniciado com novas populações *POPDOM* e *POPNDOM*.

Para reduzir o custo computacional necessário para determinar a tabela de dominância IDOM, um tamanho máximo de POPDOM é estipulado, ou seja, é fixado um número máximo de soluções dominadas. As soluções que possuírem os piores índices IDOM são transferidas para uma população externa ao processo evolutivo (POPDOMold).

Ainda com o intuito de reduzir o custo computacional, pode-se pensar em diminuir o espaço de busca a ser explorado. De posse da fronteira não-dominada a cada geração, o engenheiro/projetista pode estabelecer valores extremos mais interessantes para os objetivos e, assim, restringir os limites dos parâmetros. Esta ação caracteriza um método de otimização com decisão progressiva.

O processo evolutivo pode ser finalizado por um número máximo de gerações, por um número máximo de soluções não-dominadas, por verificação da não melhoria na fronteira ou por qualquer outra decisão do usuário.

A Figura 7.3 apresenta um esquema da rotina principal. O pseudocódigo mostra passo a passo o algoritmo proposto, sendo possível observar a ordenação e entender a dinâmica da metodologia. Na sequência, cada procedimento é explicado em detalhes.

```
% Declarando as variáveis iniciais:
nbind  = 30;              % número de indivíduos - tamanho da população;
pcross = 0,9;             % probabilidade de cruzamento;
pmut   = 0,025;           % probabilidade de mutação;
nbgen  = 50;              % número máximo de gerações;
nvar   = 5;               % número de variáveis;
limites = [1 1,2; 10 50]; % limites das variáveis [mínimo máximo];
nvar   = size(limites,1); % número de variáveis;
fniche = 0;               % técnica de nicho    - 0:off 1:on;
fclear = 1;               % técnica de espaçamento - 0:off 1:on;
freduc = 0;               % redução do espaço de busca - 0:off 1:on;

%% Rotina principal – i define o número de vezes que a rotina será executada.
para i = 1:10
% AVDOM: avaliação de POPDOM: população de dominados;
% AVDOMold: avaliação de POPDOMold: população de dominados por muitos;
% AVNDOM: avaliação de POPNDOM: população de não-dominados;
% AVPOP: avaliação de POPREAL: população submetida aos operadores;
% rPOPREAL: auxiliar de POPREAL; % rAVPOP: auxiliar de AVPOP;
   POPREAL = aleatório(nbind, nvar, limites); % Gera a População Inicial
   AVPOP = mérito(POPREAL);  % Avalia População Inicial
```

```
   n=1; % % Início do processo evolutivo
   POPDOM=[ ];POPNDOM=[ ]; POPDOMold=[ ];
   AVDOM=[ ];AVNDOM=[ ]; AVDOMold = [ ];     % inicialização matrizes
   Enquanto n < nbgen
     IDOM = [ ]; % Verificando Pareto
     [POPDOM, POPNDOM, AVDOM, AVNDOM, IDOM] = ...
        pareto(POPREAL, AVPOP);
     Se ( freduc == 1 ) % Redução do Espaço de Busca
        [POPDOM, POPNDOM, AVDOM, AVNDOM, AVDOMold, ...
                POPDOMold, limites] = redução(POPREAL, AVPOP, ...
                POPDOM, AVDOM, AVDOMold, POPDOMold, limites);
     fim_se
     Se  ( fclear == 1 ) % Técnica de Espaçamento
        [POPNDOM,AVNDOM, POPDOM, AVDOM, IDOM]= ...
                clear( POPNDOM, AVNDOM, POPDOM, AVDOM, IDOM);
     fim_se
     POPREAL=[ ]; AVPOP=[ ]; % Criando População POPREAL
     [POPRREAL, AVPOP, AVDOMold, POPDOMold, AVDOM, POPDOM]= ...
     criar(POPNDOM, POPDOM, AVNDOM, AVDOM, IDOM, nbind, ...
     AVDOMold, POPDOMold);
     AV=[ ]; % Técnica de Nicho
     Se  ( fniche == 1 )
        [NicheSig]=niche( POPREAL, AVPOP); AV=NicheSig;
     Senão AV=AVPOP; fim_se
     rPOPREAL = [ ];
     rPOPREAL = seleção( POPREAL, AV, nbind);          % Seleção
     rPOPREAL = cruzamento( rPOPREAL, pcross);         % Cruzamento
     rPOPREAL = mutação( rPOPREAL, pmut);              % Mutação
     rPOPREAL = reflexão( rPOPREAL, limites);          % Reflexão
     rAVPOP = Mérito(rPOPREAL);        % Avaliando os novos indivíduos
     % Montando nova POPREAL – Elitismo Global
     POPREAL=[POPREAL; rPOPREAL; POPDOM];
     AVPOP = [AVPOP ; rAVPOP ; AVDOM];
     n=n+1;
   fim_enquanto
fim_para    % fim
```

Figura 7.3. Pseudocódigo: Algoritmo Genético Multiobjetivo (AGMO).

## 7.3 Extração das soluções não-dominadas

Para distinguir as soluções não-dominadas, é necessário testar uma a uma todas as soluções que compõem *POPREAL*, usando (7.1). O índice indicativo de quantas vezes uma solução é dominada por outras (*IDOM*) é calculado durante esta verificação. O conhecimento deste índice permite completar *POPREAL* com indivíduos pouco dominados, o que acelera o processo de convergência. Em contrapartida, o número de verificações da condição de Pareto aumenta com a obtenção deste índice. Entretanto, o tempo para este cálculo pode ser insignificante quando comparado ao tempo de cálculo necessário para a avaliação da solução. A Figura 7.4 mostra o procedimento de verificação de Pareto (a) com e (b) sem o cálculo de *IDOM*.

```
% identificando dominância - minimização
[nbp,nvar]=size(POPREAL); IDOM=zeros(nbp,1);
para i=1:nbp
   para j=1:nbp
        se AVPOP(i,:) < = AVPOP(j,:)
             se AVPOP(i,:) == AVPOP(j,:)
                  IDOM(j) = IDOM(j) + 0;
             senao  IDOM(j) = IDOM(j) + 1; fim_se
        fim_se
   fim_para
fim_para
% separação
auxN=1; aux=1; AVNDOM=[ ]; POPNDOM=[ ]; AVDOM=[ ];POPDOM=[ ];
se IDOM(i)==0
     AVNDOM(auxN,:)=AVPOP(i,:); POPNDOM(auxN,:)=POPREAL(i,:);
     auxN=auxN+1;
senão AVDOM(aux,:)=AVPOP(i,:);
     IDOM(aux)=b(i); POPDOM(aux,:)=POPREAL(i,:);
     aux=aux+1;
fim_se
```

(a) pseudocódigo de identificação de dominância com IDOM

```
% identificando dominância - minimização
[nbp,nvar]=size(POPREAL); D=zeros(nbp,1);
para i=1:nbp-1
   para j=i+1:nbp
         se D(j) == 0
               se AVPOP(i,:) < = AVPOP(j,:)
                     se AVPOP(i,:) == AVPOP(j,:)          D(j) = 0;
                     senao  D(j) = 1; fim_se
               fim_se
         fim_se
   fim_para
fim_para
% separação
Mesmo código, troca-se IDOM(i) por D(i)
```

(b) pseudocódigo de identificação de dominância sem IDOM

Figura 7.4. Verificação da condição de dominância de Pareto.

Outro aspecto importante consiste em verificar a repetição de soluções, que deve ser evitada. Pode acontecer que um par de indivíduos não sofra nem cruzamento nem mutação. Essas soluções retornariam à verificação de Pareto e seriam idênticas a soluções já existentes. Tal verificação deve ser feita comparando os parâmetros da solução, e não seus méritos. Méritos idênticos para soluções diferentes podem existir num problema multimodal.

Para reduzir o custo computacional necessário para calcular IDOM, estipula-se um tamanho máximo para POPDOM. As soluções com os piores índices de dominância são movidas para uma população externa ao processo evolutivo (POPDOMold).

## 7.4 Redução do Espaço de Busca

Pode-se pensar em reduzir o espaço dos parâmetros como procedimento para minimizar o custo computacional (algo semelhante ao proposto para a otimização mono-objetivo. Isto porque, diminuindo-se a complexidade da procura, o número de gerações necessário para a obtenção da fronteira Pareto-ótima pode ser menor.

Esta ação funciona como um decisor progressivo. Na medida em que uma aproximação da fronteira Pareto é conhecida, o engenheiro/usuário pode analisar e, se desejar, estipular níveis mínimos (ou máximos) para determinados objetivos. Assim, as soluções que não atenderem a estas especificações são eliminadas (POPDOMold) e os limites de busca redefinidos.

A Figura 7.5 ilustra este procedimento num problema de maximização de desempenho e minimização de custo. Uma vez estipulados um valor mínimo para o desempenho e um valor máximo para o custo, pode-se excluir grande parte da região factível. Seguem no processo evolutivo somente as soluções que estão dentro destes novos limites. Para melhor exploração desta região restrita, uma população nova pode ser criada dentro destes limites redefinidos, juntando-se às soluções remanescentes.

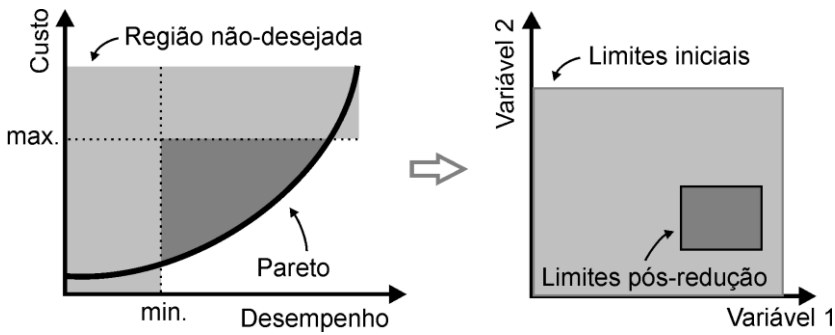



Figura 7.5. Redução de espaço de busca.

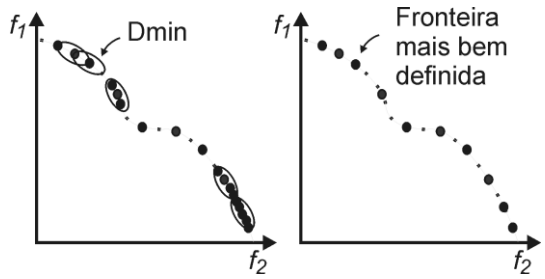
A ilustração 7.5. corresponde a um caso particularmente simples: poder-se-ia obter no espaço de parâmetros uma forma qualquer, sem limitação clara para os valores extremos dos parâmetros ou mesmo múltiplas zonas separadas. Em problemas simples, estas decisões podem ser feitas no início do processo de otimização, como um decisor a priori. Entretanto, em problemas complexos, a localização das melhores soluções pode não ser rápida, mas sim dependente de uma

evolução lenta. Restrições fortes desde o início podem dificultar ou até mesmo inviabilizar o processo evolutivo. Assim como em problemas mono objetivo, a redução do espaço de busca deve ser usada com cautela. Com a má utilização da redução, existe o risco de perda de informações, como por exemplo, possíveis ótimos locais.

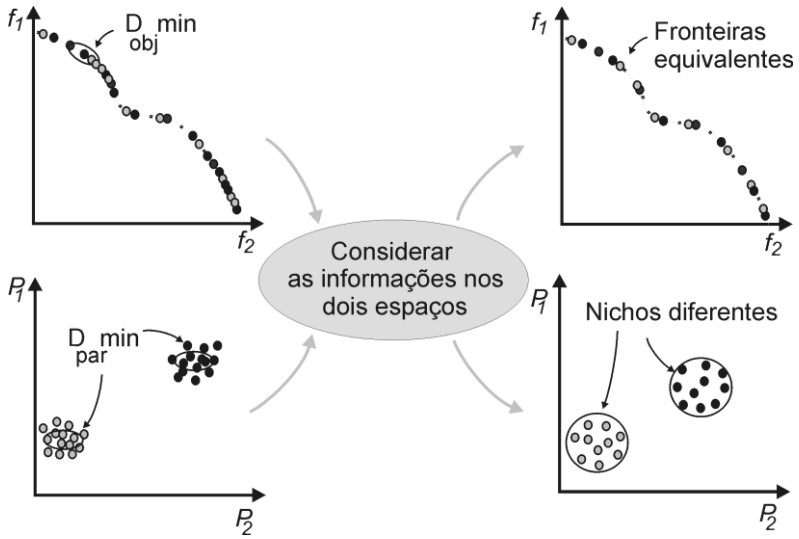
## 7.5 Espaçamento entre Soluções Não-Dominadas

Em certos problemas, o conjunto Pareto-ótimo pode ser extremamente amplo ou, ainda, conter um número infinito de soluções (sobretudo para os problemas com variáveis contínuas). A permanência de um número excessivo destas soluções no processo evolutivo leva à diminuição da diversidade, podendo acarretar em uma convergência prematura. Isto porque, a grande concentração de soluções não-dominadas reduz a importância ('pressão') do processo de seleção, diminuindo a exploração do espaço de busca.

O procedimento 'espaçamento', também chamado de *clearing*, tem como princípio melhorar o estabelecimento da fronteira Pareto, evitando uma possível convergência prematura ou ainda aglomerações de soluções em determinadas regiões. A Figura 7.6a ilustra o procedimento mais comum. A Figura 7.6b mostra a ação mais adequada.




(a) Espaçamento realizado somente no espaço dos objetivos




(b) maior equilíbrio – espaçamento realizado sobre os dois espaços: objetivos (f) e parâmetros (p)

Figura 7.6. Espaçamento entre soluções não-dominadas.

Com a intenção de melhorar a repartição dos indivíduos sobre a fronteira Pareto, pode-se estipular uma distância mínima (*Dmin*) entre os indivíduos não-dominados. Este procedimento é chamado de 'preservação da diversidade' e o cálculo de *Dmin* é realizado somente no espaço de objetivos (ver Figura 7.6a) [6]. Esta ação funciona bem em problemas com apenas uma fronteira ótima.

No caso de problemas com múltiplas fronteiras, conforme Figura 7.6b, é conveniente também observar a distância entre os indivíduos no espaço de parâmetros, evitando que soluções de nichos diferentes sejam penalizadas por terem objetivos parecidos ou mesmo idênticos.

Pode-se pensar em fazer o cálculo de *Dmin* somente sobre o espaço de parâmetros, mas isto poderia resultar em efeitos indesejáveis. Por exemplo, poderia ocorrer a eliminação de soluções que estão próximas no espaço de parâmetros, mas que possuem desempenhos bastante diferentes devido a descontinuidades. Uma vez detectada a

semelhança entre soluções não-dominadas nos dois espaços, indivíduos são penalizados. A penalidade consiste em mover o indivíduo punido para *POPDOM*, alterando seu *IDOM* de 0 para um índice de dominância aleatório (0 para não-dominado e 1,2,...$k$ para um indivíduo dominado $k$ vezes).

A Figura 7.7 apresenta o pseudocódigo para o 'espaçamento'. Este método necessita da experiência do projetista/usuário para a definição das distâncias mínimas no domínio dos objetivos ($D_{obj}min$) e no domínio dos parâmetros ($D_{par}min$).

```
% Espaçamento  [nbp,nvar]=size(POPNDOM); aux=size(POPDOM,1)+1;
para i=1:nbp-1
   para j=i+1:nbp
        se AVNDOM(j,1) ~= 1e8
              se abs(POPNDOM(i,:)-POPNDOM(j,:)) < Dparmin
              & se abs(AVNDOM(i,:)-AVNDOM(j,:)) < Dobjmin
                    POPDOM(aux,:) = POPNDOM(j,:);
                    AVDOM(aux,:) = AVNDOM(j,:);
                    IDOM(aux) = round(10*rand(1));
                    AVNDOM(j,1) = 1e8; aux=aux+1;
              fim_se
        fim_se
   fim_para
fim_para % segue 'limpeza' de AVNDOM (1e8).
```

Figura 7.7. Espaçamento entre soluções não-dominadas.

Foi aqui testada uma adaptação da técnica de nicho para a ação dinâmica do 'espaçamento'. Na técnica de nicho, as duas informações (distâncias entre objetivos e parâmetros) são utilizadas. Optou-se por manter o processo de distâncias explícitas proposto aqui, isto porque o projetista tem maior controle definindo ele mesmo as distâncias, o que permite também um aprendizado sobre o comportamento do problema. O método dinâmico ainda tem como aspecto negativo o aumento do custo computacional. É preciso atenção ao fato de que é muito importante manter a solução extrema de cada objetivo, que pode vir a ser indesejavelmente excluída se não obedecer às distâncias

mínimas. Outro aspecto a ser observado, *Dparmin* e *Dobjmin* são vetores com variáveis condizentes aos parâmetros [*Dpar1, Dpar2, ... nvar*] ou objetivos [*Dobj1, Dobj2, ... nobj*], respectivamente.

## 7.6 Construção da população de trabalho *POPREAL*

A população de trabalho ou 'real', chamada de POPREAL, é a maneira aqui utilizada para relacionar o grupo de soluções dominadas e não-dominadas. A partir desta população será feita a seleção dos indivíduos que sofrerão os demais operadores genéticos.

*POPREAL* possui um tamanho mínimo definido por *nbind*, mas não um máximo. A intenção é representar bem a diversidade de soluções, porém buscando a convergência ao conjunto Pareto. Desta forma, *POPREAL* é montada com todos os indivíduos de *POPNDOM* (após *clearing*); somados a *nbind*/4 soluções de *POPDOM* de maneira a manter certa diversidade. Estes indivíduos dominados são escolhidos entre os que possuírem os menores índices *IDOM* (quando houver soluções com índices de dominância iguais, a escolha é feita aleatoriamente, escolhendo a solução primeira encontrada). Se o número de indivíduos de *POPREAL* for ainda menor que *nbind*, ela é completada com mais indivíduos de *POPDOM* (escolhidas outras que tiverem os menores índices). A Figura 7.8 ilustra um exemplo de população de trabalho.

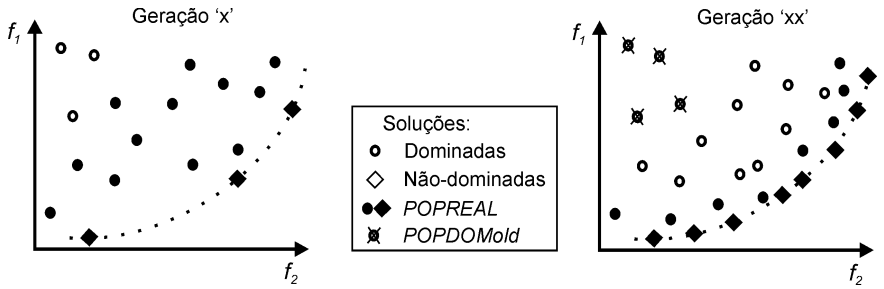



Figura 7.8 População de trabalho (POPREAL) baseada no índice de dominância IDOM.

Nesta operação, limita-se também o tamanho de *POPDOM* ao dobro de *nbind*, de modo a reduzir o custo computacional envolvido no cálculo do índice de dominância *IDOM*. As soluções excluídas de *POPDOM* são armazenadas em *POPDOMold*, população esta que não participa do processo evolutivo.

Este controle é feito em duas etapas: primeiro são excluídas as soluções que possuem *IDOM* demasiadamente alto, ou seja, aquelas que se encontram distantes da fronteira Pareto (*IDOM* > 200, por exemplo); se o tamanho de *POPDOM* for ainda superior a $2 \times nbind$ , retiram-se as soluções de maior *IDOM* até restar o número desejado.

```
% Completando POPREAL
POPREAL=POPNDOM;  AVPOP=AVNDOM;   % todas as não-dominadas
para i=1:round(0.25*nbind)                % 1/4 dominados
      [a,p]=min(IDOM);  IDOM(p)=1e10;      % por dominância
      POPREAL(end+1,:)=POPDOM(p,:);      AVPOP(end+1,:)=AVDOM(p,:);
fim_para
aux=size(POPREAL,1);
se aux<nbind                          % ainda menor que nbind ?
     para i=(aux+1):nbind
        [a,p]=min(IDOM);      % por dominância
        IDOM(p)=1e10; POPREAL(i,:)=POPDOM(p,:); AVPOP(i,:)=AVDOM(p,:);
     fim_para
fim_se
```

Figura 7.9. Montando POPREAL.

## 7.7 Técnica de Nicho

O procedimento para a detecção de nichos em problemas multiobjetivos parte do conceito adaptado da mesma técnica para o AG mono objetivo. Esta técnica permite a exploração de regiões distintas que constituem ótimos locais, conforme ilustrado na Figura 7.10 e na Figura 7.6b. Em projetos práticos, por exemplo, a detecção de soluções diferentes proporciona ao projetista a possibilidade de escolha final não só pelos objetivos pré-definidos, mas também pela facilidade de construção desta ou daquela solução.

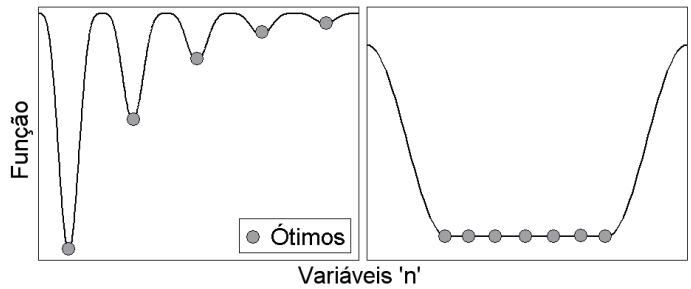



Figura 7.10 Nicho em problemas mono-objetivo.

Proposição: calcular dois índices de nicho (ou de semelhança), no domínio dos objetivos (*Nobj*) e dos parâmetros (*Npar*), de maneira a respeitar os dois domínios.

Estes índices são as distâncias entre indivíduos subsequentes segundo a ordem estabelecida pelos valores de cada objetivo. O operador genético de seleção irá trabalhar com estes índices e não com as avaliações do problema. O que se deseja é detectar nichos distintos (espaço de parâmetros), ou seja, a detecção de ótimos locais e/ou globais num problema multimodal, conforme ilustrado na Figura 7.10.

O processo é constituído de duas etapas. Primeiro, para cada objetivo $k$, dispõe-se a população em ordem crescente (ou decrescente) segundo o objetivo em análise (é necessário guardar um ponteiro que indique a ordem original). Na sequência são calculadas as distâncias entre os indivíduos nos dois espaços, $Nobj_k$ e $Npar_k$, conforme a ordem estabelecida. A segunda etapa consiste na junção dos dois índices de semelhança através de uma função de transferência.

**Índice de Nicho no domínio dos objetivos**

Este índice é calculado conforme Deb *et al.* [6] e Sareni *et al.* [10]:

$$Nobj_k(\vec{x}_i) = f_k(\vec{x}_{i+1}) - f_k(\vec{x}_{i-1}) \ ,$$

onde $\vec{x}_{i:nbp}$ é a solução em análise e *nbp* é o tamanho de *POPREAL.* É preciso atribuir o maior índice aos indivíduos situados nos extremos da fronteira ($\vec{x}_1$ e $\vec{x}_{nbp}$), para manter estas soluções no processo.

**Índice de Nicho no domínio dos parâmetros**

Este índice é calculado conforme:

$$Npar_k(\vec{x}_i) = \left\| \vec{x}_{i\,k} - \vec{x}_{i-1\,k} \right\| + \left\| \vec{x}_{i+1\,k} - \vec{x}_{i\,k} \right\| .$$

É preciso lembrar que $\vec{x}$ pode ser constituído por *nvar* variáveis, de modo que é imperativo fazer a normalização das variáveis, pois estas são somadas entre si.

**Função de Transferência**

Com a intenção de somar as distâncias ($Nobj_k$ *e* $Npar_k$), deve-se primeiro tornar as grandezas correspondentes, para que elas variem de maneira equivalente. Para tal, usa-se aqui uma função sigmóide, como ilustra a Figura 7.11, onde *Nmin* e *Nmax* são os mínimos e máximos de cada índice.

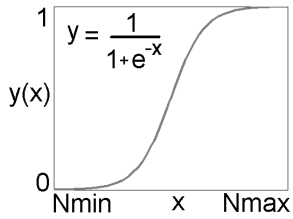



Figura 7.11. Função de transferência – técnica de Nicho.

Realizada esta transformação, pode-se construir para objetivo *k* um índice de semelhança único, levando em conta os dois espaços (como por exemplo, pelo produto dos dois índices: $Nobj_k \times Npar_k$). Quando a técnica de nicho é necessária, o processo de seleção é realizado com base estes índices no lugar das funções *fitness*.

```
% técnica de Nicho
nbobj = size(AVPOP,2); [nbp,nvar] = size(POPREAL); AVaux  = AVPOP;
Paux=[ ];  %normalizando os parâmetros
para i=1:nvar
        Paux(:,i) = POPREAL(:,i) / max(POPREAL(:,i));
fim_para
Sigpar = [ ];      Sigobj =[ ];
para k=1:nbobj
   % colocando em ordem crescente
   AV=[ ]; POP=[ ]; Pos=[ ];
   para w=1:nbp
         [v, p] = min(AVaux(:,k));
         AV(w) = AVaux(p,k);  POP(w,:) = Paux(p,:);
         AVaux(p,k) = 1e10;  Pos(w) = p;
   fim_para
   % cálculo da distância
   Nobjk=zeros(nbp,1);    Npark=zeros(nbp,1);
   para i=2:(nbp-1)
      Nobjk(Pos(i)) = (AV (i+1)-AV(i-1));
      Npark(Pos(i)) = sqrt(sum((POP(i,:)-POP(i-1,:)).^2)) + ...
         sqrt(sum((POP(i+1,:)-POP(i,:)).^2));
   fim_para
   % salvando extremos
   Nobjk(Pos(end)) = max(Nobjk(2:end-1))+.1;
   Nobjk(Pos(1)) = Nobjk(Pos(end))+.1;
   Npark(Pos(end)) = max(Npark(2:end-1))+.1;
   Npark(Pos(1)) = Npark(Pos(end))+.1;
   % função sigmóide   Lmax=7; Lmin=-7;
   %objetivos
   Dmax = max(Nobjk);                Dmin = min(Nobjk);
   a = (Lmax-Lmin)/(Dmax-Dmin);      b = Lmin-a*Dmin;
   Domo = a.*Nobjk+b;                Sigobj(:,k) = 1./(1+exp(-Domo));
   %parâmetros
   Dmax = max(Npark);                Dmin = min(Npark);
   a = (Lmax-Lmin)/(Dmax-Dmin);      b = Lmin-a*Dmin;
   Domp = a.*Npark+b;                Sigpar(:,k) = 1./(1+exp(-Domp));
fim_para
% Indice de Nicho   Nicho = Sigobj.*Sigpar;
```

Figura 7.12 Técnica de Nicho.

Outros cuidados devem ser tomados para a correta resolução de problemas multimodais, sendo o mais importante: as soluções não-dominadas com avaliações iguais, porém com parâmetros diferentes, devem ser consideradas como Pareto.

A própria montagem da população de trabalho *POPREAL* e a maneira como foi proposto o *clearing* contribuem para a determinação de múltiplas fronteiras. Ainda sobre o *clearing*, cabe mencionar que a formulação apresentada aqui na técnica de nicho pode ser utilizada para a exclusão de soluções não-dominadas. As soluções com menores índices seriam excluídas automaticamente até que um número mínimo de soluções não-dominadas fosse atingido. Para isto, a cada exclusão seria necessário recalcular todos os índices, pois a disposição das soluções foi alterada. Como já dito, o custo computacional deste procedimento seria muito elevado em comparação à técnica de *clearing* adotada.

## 7.8 Processo de Seleção

A seleção é responsável pela escolha dos casais que sofrerão os operadores genéticos de cruzamento e mutação. O importante é, através das soluções escolhidas, amostrar satisfatoriamente a população corrente, permitindo a estes operadores a possibilidade de explorar bem o espaço de busca.

O número de indivíduos selecionados de *POPREAL* é sempre *nbind*. A seleção é feita pela ação conjunta de dois procedimentos. Os pais são em parte escolhidos por amostragem determinística (*deterministic sampling – AMDET* – baseada na média dos méritos da população, a qual dá ênfase à parte central da fronteira Pareto) e em parte por Torneio (baseado em cada objetivo individualmente, o qual enfatiza os extremos da fronteira Pareto), ambos propostos para o AG mono-objetivo [11].

Desta forma, em um problema com dois objetivos por exemplo, a população de soluções selecionadas será constituída por três grupos:

pela média dos dois objetivos, por torneio para o primeiro objetivo e por torneio para o segundo objetivo. Este procedimento facilita a obtenção de um conjunto Pareto bem distribuído. A Figura 7.13 ilustra processos de seleção. A Figura 7.14 apresenta o pseudocódigo deste procedimento.

Apenas uma preocupação: para a amostragem determinística, deve-se calcular a média das avaliações com cuidado, pois qualquer tipo de ponderação é desaconselhado em problemas conflitantes. Este cálculo é feito somente sobre as avaliações da população que sofrerá a seleção, o que acelera o processo de convergência.

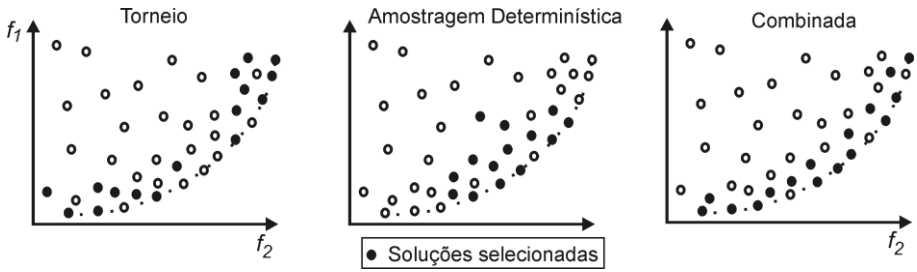



Figura 7.13. Ilustração do processo de seleção.

```
% Seleção
rPOPREAL=[ ];
[nbp,nvar]=size(POPREAL);
[nbp,nbobj]=size(AVPOP);
MED=sum(popAV)/size(AVPOP,1);
% número de soluções por sub-população
nn=ceil(nbind/(nbobj+1));

% seleção por Torneio
rpopTorneio =[ ];
aux = ceil(nbp/20);          % 10% de nbp a cada torneio
para b=1:nn
    aux2=b;
```

```
    para i=1:nbobj          % uma sub-população por objetivo
        ind1=randperm(nbp);  ind2=ind1(1:aux);
        av=[ ]; av=AVPOP(ind2,i); [a,p]=min(av);
        rpopTorneio(aux2,:)=POPREAL(ind2(p),:);
        aux2=aux2+nn;
    fim_para
fim_para

% seleção por Amostragem Determinística
aux=1; rpopAMDET=[ ];
para i = 1:size(POPREAL,1)
    se AVPOP(i,:)<MED
        rpopAMDET(aux,:)=POPREAL(i,:);
        aux=aux+1;
    fim_se
fim_para

% verificando o tamanho de rpopAMDET
nbrpop=size(rpopAMDET,1);
se nbrpop<nn
    para x=1:(nn-nbrpop)
        ind=ceil(nbp*rand(1));
        rpopAMDET(nbrpop+x,:)=POPREAL(ind,:);
    fim_para
fim_se
se nbrpop>nn
    rrpop=rpopAMDET(1:nn,:);
    rpopAMDET=[ ];    rpopAMDET=rrpop;
fim_se

rPOPREAL=[rpopTorneio; rpopAMDET];
```

Figura 7.14 Seleção.

## 7.9 Cruzamento e Mutação

Com a seleção feita, os operadores que geram novos indivíduos são aplicados: cruzamento e mutação, os mesmos utilizados no AG mono-objetivo, conforme capítulo 5.

Para explorar eficientemente o espaço de busca de um problema multiobjetivo, e para que a população evolua rapidamente, é imperativo que o pai deste filho polarizado tenha mérito maior que o segundo pai, ou seja:

$$f(X^{n,i}) > f(X^{n,j}) \ ,$$

onde *f*(.) representa a função *fitness* na otimização mono objetivo. Para um problema multiobjetivo, utiliza-se a dominância como indicativo. O indivíduo $i$ será sempre o indivíduo dominante. No caso em que os indivíduos são ambos não-dominados, a posição é indiferente.

## 7.10 Considerações sobre restrições aos parâmetros

Pode ocorrer que a geração de novos indivíduos não respeite os limites pré-determinados das variáveis, ou seja, a busca das soluções Pareto-ótimas durante o processo evolutivo pode gerar indivíduos que possuem uma ou mais variáveis fora de suas faixas de valores permitidos. Nestes casos, é necessário ajustar ou os novos indivíduos para dentro dos limites ou permitir a redefinição destes. Esta ação pode ser feita de muitas maneiras. A mais simples e interessante consiste na saturação (Figura 7.15), isto é, atribuem-se às variáveis que extrapolam seus limites os valores máximos correspondentes.

Este procedimento é interessante porque o projetista pode aprender com o problema, pois o comportamento de cada variável que toca os limites poderá ser observado. Isto pode ser feito, por exemplo, reiniciando o processo evolutivo com as mesmas populações, mas adaptando os limites. Na prática, entretanto, esta variação de limites nem sempre pode ser feita (por exemplo, quando os valores máximos das variáveis são definidos por aspectos construtivos imutáveis).

```
% reflexão por saturação
 [rnbp,rnvar]=size(rPOPREAL);
 para i=1:rnbp
       para j=1:rnvar
```

```
            se rPOPREAL(i,j) < limites(j,1)  rPOPREAL(i,j) = limites(j,1);
            fim_se
            se rPOPREAL(i,j) > limites(j,2)  rPOPREAL(i,j) = limites(j,2);
            fim_se
      fim_para
fim_para
```

Figura 7.15 Reflexão por saturação.

## 7.11 Elitismo Global

Com a geração e posterior controle de novos indivíduos (*rPOPREAL*), estes são avaliados perante o problema. De modo a não perder nenhuma solução, todo o conjunto formado pela união de *rPOPREAL* e *POPDOM* é adicionado à população corrente *POPREAL* (lembrando que esta é constituída por todos os elementos de *POPNDOM*).

Este é o conceito de elitismo global proposto para os AG mono-objetivos: nenhum indivíduo se perde durante o processo evolutivo e os filhos podem ocupar os lugares de qualquer indivíduo corrente, se possuírem melhores méritos.

A nova *POPREAL* é submetida à condição de Pareto (7.1). Isto resulta em uma *POPNDOM* modificada, cujo tamanho que aumenta e diminui continuamente, enquanto que *POPDOM* aumenta sempre. No AGMO descrito aqui, o conceito de elitismo global é utilizado, portanto, quando da junção das três populações a cada geração.

Por fim, o processo evolucionário é reiniciado com novas *POPDOM* e *POPNDOM*.

O código completo do AGMULTI aqui descrito pode ser encontrado em HTTPS://GITHUB.COM/PECCE-IFSC/OTIMIZAÇÃO

O próximo capítulo apresenta funções teste que validam a eficácia do algoritmo proposto.

**Referências**

[1] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithms: analyzing the state-of-the-Art," Evolutionary Computation, MIT Press, v. 8, n. 2, pp. 125-147, 2000. doi: 10.1162/106365600568158.

[2] J. D. Schaffer, Some Experiments in Machine Learning using Vector Evaluated Genetic Algorithms, Thesis (Doctor of Philosophy) – Vanderbilt University, Nashville, 1984.

[3] C. A. Coello Coello, "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques," Laboratorio Nacional de Informatica Avanzada, Mexico, 1999.

[4] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," In: First International Conference on Genetic Algorithm, Lawrence Erlbaum, New Jersey, pp. 93-100, 1985.

[5] C. M. Fonseca and P. J. Fleming, "Genetic Algorithm for Multiobjective Optimization: Formulation, Discussion and Generalization," In: 5th Conference on Genetic Algorithm, San Mateo, California, pp. 416-423, Aug. 1993.

[6] K. Deb, S. Agrawal, A. Pratab, T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: NSGA-II," IEEE – Transactions on Evolutionary Computation, v. 6, n. 2, pp. 182–197, Apr. 2002.

[7] J. Horn and N. Nafpliotis, "Multiobjective Optimization using the Niched Pareto Genetic Algorithm," Technical report, University of Illinois at Urbana Champaign, Illinois, 1993.

[8] C. F. Viennet and I. Marc, "Multicretira Optimization using a Genetic Algorithm for Determining a Pareto Set," International Journal of Systems Science, v. 27, n. 2, pp. 255-260, 1996.

[9] E. Zitzler, M. Laumanns and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," In: EUROGEN 2001,

Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, pp. 12-21, Sept. 2001.

[10] J. A. Vasconcelos, J. A. Ramirez, R. H. C. Takahashi and R. R. Saldanha, "Improvements in genetic algorithms," in IEEE Transactions on Magnetics, vol. 37, no. 5, pp. 3414-3417, Sept. 2001, doi: 10.1109/20.952626.

[11] B. Sareni, J. Regnier and X. Roboam, "Recombination and self-adaptation in multi-objective genetic algorithms," Lecture Notes in Computer Science, Springer, v. 2933, pp 115-126, 2004.

[12] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley Longman Inc., New York, 1989. ISBN 9780201157673.

# AG multiobjetivos – testes

O capítulo anterior apresentou em detalhes a implementação de um AGMO completo. Para avaliar sua a eficácia, três funções analíticas foram utilizadas: Schaeffer F3, 3 parábolas e Himmelblau.

O AGMO foi implementado conforme Figura 7.3 e com as seguintes condições:

- População = 20 indivíduos;
- Probabilidade de cruzamento inicial = 90%;
- Probabilidade de mutação inicial = 2,5%;
- Número máximo de gerações = 50;
- Utilizou-se o SGA, ou seja, os operadores genéticos trabalham com um número de pares de indivíduos que representa o tamanho total da população;
- Ferramentas: escalonamento, formação de nichos, *clearing*, variação dinâmica de probabilidades, redução do espaço de busca e elitismo global;
- Operadores genéticos codificados com números reais;

Os demais parâmetros variaram para cada função. As funções teste foram otimizadas 100 vezes com o objetivo de assegurar a validade da resposta alcançada pela repetição do processo.

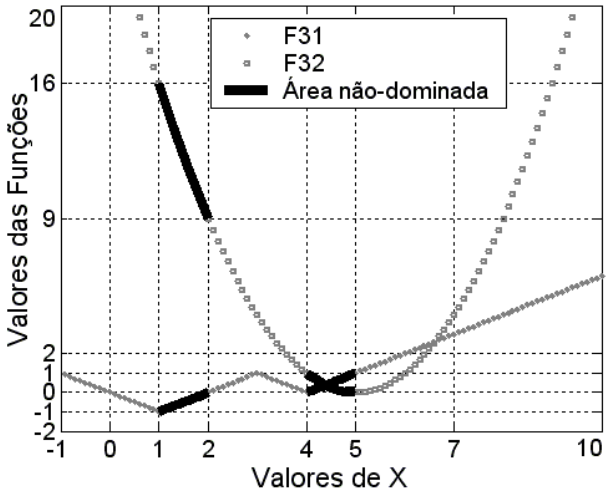
## 8.1 Problema Schaffer $F_3$

Apresentado por Schaffer [1], o conjunto $F_3$ é um teste interessante devido à descontinuidade na sua fronteira Pareto-ótima. O problema consiste em minimizar as seguintes funções:

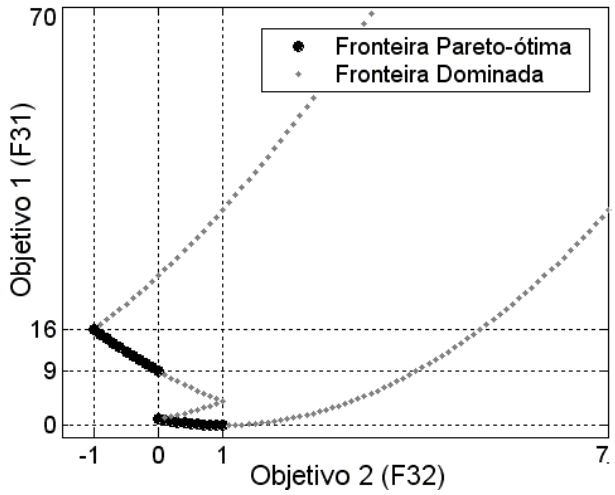
$$F_{31}(x) = \begin{cases} -x & se \quad x \le 1 \\ -2 + x & se \quad 1 < x \le 3 \\ 4 - x & se \quad 3 < x \le 4 \\ -4 + x & se \quad x > 4 \end{cases} \quad \text{e} \quad F_{32}(x) = (x-5)^2 \quad , \quad (8.1)$$

sendo $x$ pertencente ao intervalo [-1 ; 10]. O indivíduo é, portanto, formado por uma única variável ($x$). Já o vetor objetivo é constituído por dois valores ($F_{31}$ e $F_{32}$).

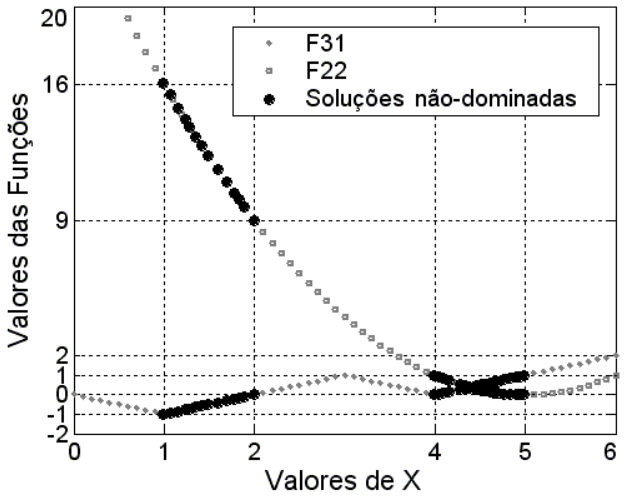
As Figuras 8.1a e 8.2b apresentam as curvas das funções. Nestas Figuras são mostradas também as regiões de soluções não-dominadas e a fronteira Pareto-ótima.



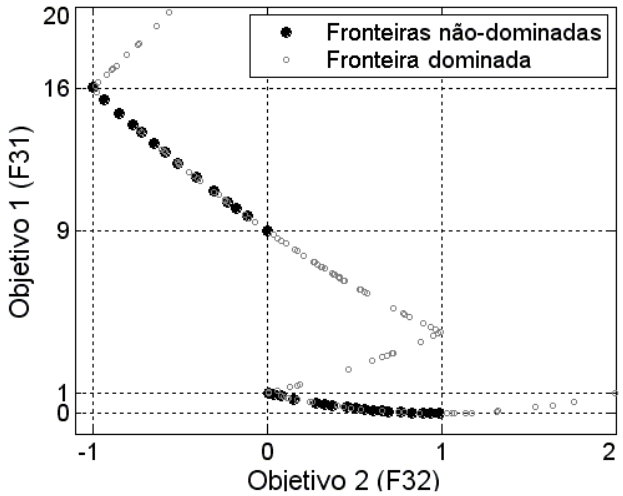

(a) Funções $F_{31}$ e $F_{32}$




(b) Fronteiras




(c) Soluções não-dominadas encontradas pelo AGMO




(d) Fronteira não-dominada obtida

Figura 8.1 – Problema Schaffer F3.

A Figura 8.1c mostra as soluções não-dominadas encontradas na otimização. Na Figura 8.1d é possível perceber a fronteira determinada por estas soluções. Para tal resultado foram utilizados 100 indivíduos correntes e uma distância mínima entre indivíduos (*clearing*) de 0,5% da faixa permitida da variável (neste caso igual a 11, já que a variável pertence ao intervalo [-1 ; 10]).

O critério de convergência foi a obtenção de um número máximo de soluções não-dominadas (aqui igual a 30). A convergência ocorreu na terceira geração, com 32 indivíduos não-dominados e 239 indivíduos dominados. Pode-se observar a boa qualidade dos resultados obtidos.

Neste caso, uma cobertura do espaço com 200 indivíduos, respeitando o critério de espaçamento das soluções, é suficiente para determinar a fronteira Pareto-ótima com aproximadamente 36 indivíduos não dominados (Figuras 3.18 (a) e (b)); Assim, este exercício pode ser resolvido com menos cálculos que utilizando o AGMO. Este exemplo serve, então, unicamente para mostrar que o algoritmo funciona em um caso muito simples e verificável, mas não para demonstrar que o AGMO é mais eficaz que qualquer outro método.

É importante ressaltar que, alterando o valor do *clearing* e do critério de convergência, os números finais poderão ser diferentes. Um aumento da distância mínima entre os indivíduos leva a um acréscimo na dificuldade de obtenção das soluções não-dominadas, o que pode aumentar sensivelmente o número de gerações para a convergência.

É claro que se o valor de espaçamento for muito grande, possivelmente não se conseguirá atingir o número de indivíduos necessário para satisfazer o critério de convergência. Os 'pontos claros' na Figura 8.1d que aparecem dentro da fronteira não-dominada são também soluções eficientes, mas que foram penalizadas pela técnica de *clearing*.

## 8.2 Problema Parábolas

Apresentado por Dias [2], este problema consiste em minimizar simultaneamente três parábolas. O problema é interessante pela dificuldade de se obter os vértices da fronteira, pois os objetivos são todos conflitantes. As funções que definem as parábolas são:

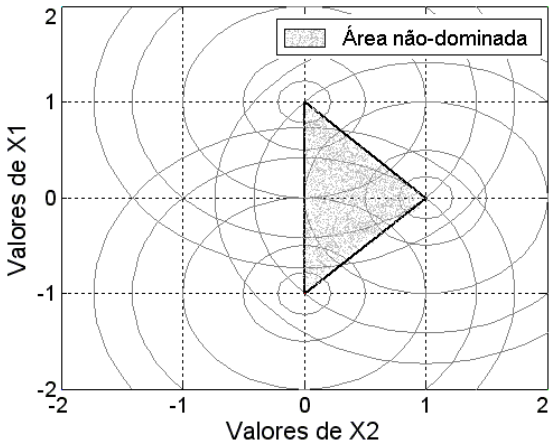
$$F_1(x) = x_1^2 + (x_2 - 1)^2 \ ,$$

$$F_2(x) = x_1^2 + (x_2 + 1)^2 + 1 \ \text{e} \qquad (8.2)$$

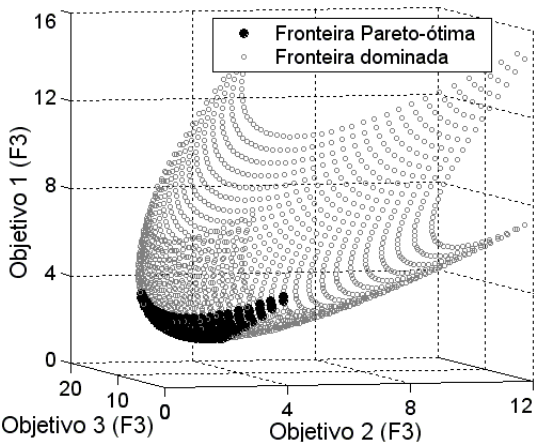
$$F_3(x) = (x_1 - 1)^2 + x_2^2 + 2 \ .$$

Aqui, cada indivíduo é formado por duas variáveis ($x_1$ e $x_2$) pertencentes ao intervalo [-2 ; 2]. Já o vetor objetivo é constituído por três valores ($F_1$, $F_2$ e $F_3$). Nas Figuras 8.2a e 8.2b estão desenhadas as curvas de nível das funções $F_1$, $F_2$ e $F_3$. Mostram-se também a região das soluções não-dominadas e a fronteira Pareto-ótima. A Figura 8.2c mostra as soluções não-dominadas encontradas na otimização. Na Figura 8.2d é possível perceber a fronteira não-dominada formada por estas soluções.

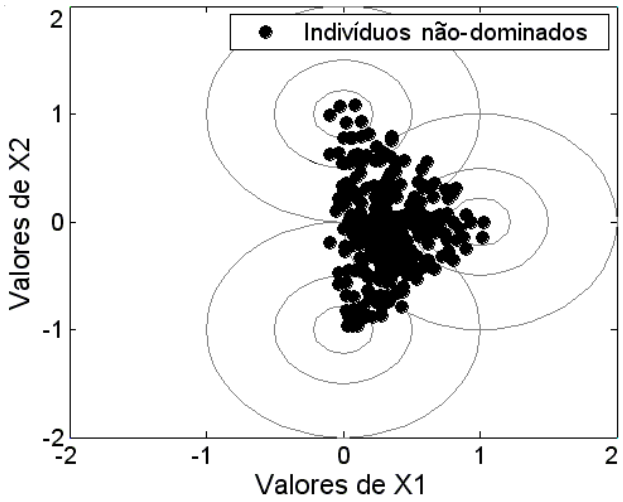
Para tal resultado foram utilizados 100 indivíduos correntes e uma distância mínima entre indivíduos de 0,1% da faixa permitida das variáveis (neste caso igual a 4, já que cada variável pertence ao intervalo [-2 ; 2]).



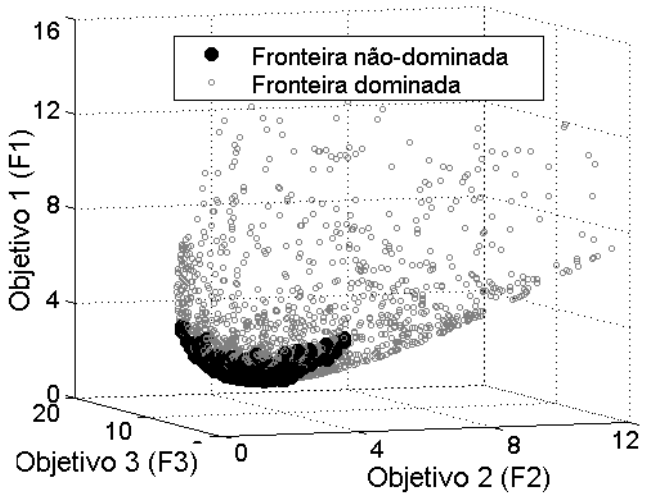

(a) Funções $F_1$, $F_2$ e $F_3$




(b) Fronteiras




(c) Soluções não-dominadas encontradas.




(d) Fronteira não-dominada.

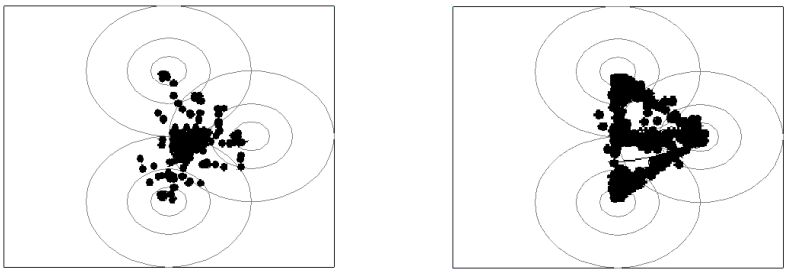
8.2 – Problema Parábolas.

O critério de convergência foi a obtenção de um número máximo de soluções não-dominadas (neste exemplo igual a 100). A convergência ocorreu na décima quarta geração, com 103 indivíduos não-dominados e 1112 indivíduos dominados. As considerações a respeito da técnica de espaçamento e do critério de convergência feitas para a função Schaffer F3 também são válidas aqui.

As Figuras 8.3a e 8.3b são apresentadas com a intenção de analisar o procedimento de seleção e a importância do espaçamento entre as soluções não dominadas. A correta determinação da fronteira Pareto (Figura 8.3b) é facilitada com o auxílio da técnica de *clearing* e com seleção utilizada (AMDET + Torneio).

Como já dito, a técnica AMDET concentra os filhos no centro da fronteira (Figura 8.3a) e o Torneio nas extremidades (Figura 8.3b). É importante ressaltar que nas três Figuras citadas (8.2c, 8.3a e 8.3b) o processo foi parado quando se atingiu um número de soluções não-dominadas acima de 100.

Neste problema, o comportamento das soluções no espaço de parâmetros e no dos objetivos é idêntico. Sendo assim, a percepção dos efeitos das técnicas em ambos os domínios é a mesma. Escolheu-se mostrar no espaço de parâmetros por maior facilidade de visualização.



(a) AMDET sem *clearing* (b) Torneio sem *clearing*

Figura 8.3 – Problema Parábolas – verificando dois tipos de seleção com ausência de 'espaçamento'.

## 8.3 Himmelblau transformada em um problema multiobjetivo

Problemas com um ou múltiplos objetivos e *n* parâmetros podem ser construídos com a ajuda da metodologia proposta por Deb [3]. Para uma dada função, o problema multiobjetivo pode ser escrito:

Minimizar

$$f_1(x) = x_1 \quad \text{e} \quad f_2(x) = g(x_2,x_3) * h(\, g(x_2,,x_3), f(x_1)) \,,$$

onde

$$h(x) = \left(1 - \sqrt{\frac{f(x_1)}{g(x_2,x_3)}}\right), \tag{8.3}$$

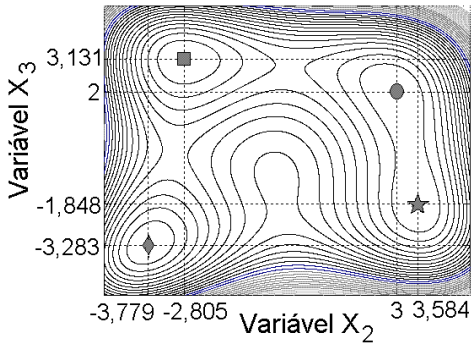
Sendo

$$g(x_2, x_3) = 1 + ({x_2}^2 + x_3 - 11)^2 + (x_2 + {x_3}^2 - 7)^2.$$

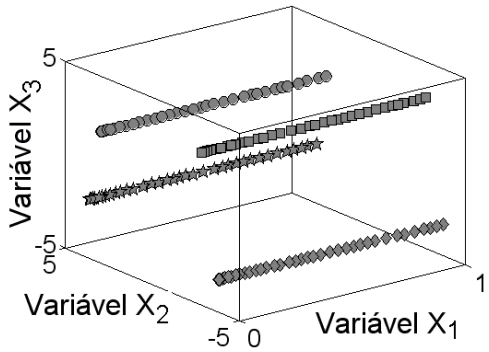
Este conjunto de equações transforma a função *g* (no caso a Himmelblau, mostrada em curvas de nível na Figura 8.4a) em um problema multiobjetivo, multimodal e com fronteira convexa.

A função Himmelblau possui quatro mínimos nos pontos ($x_2^*$, $x_3^*$) de coordenadas (3 , 2), (3,584 , -1,848), (-3,779 , -3,283) e (-2,805 , 3,131). Sendo os limites possíveis para $x_1 \in [0 \;\; 1]$ e $x_{2,3} \in [-5 \;\; 5]$, o problema de otimização multiobjetivo definido por (8.3) apresentará então quatro fronteiras Pareto iguais, semelhante à Figura 8.4c.

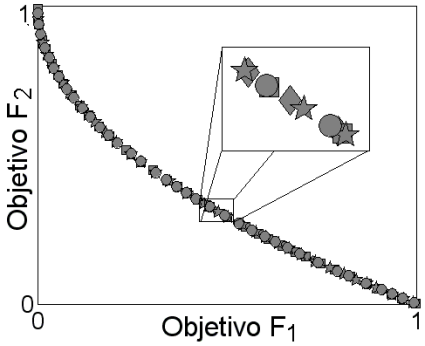
A técnica de nicho foi a responsável por permitir a obtenção das quatro fronteiras, ou seja, os quatro grupos de soluções não-dominadas. Contribuem também para o sucesso na resolução deste problema a técnica de *clearing* e a montagem de *POPREAL* através de *IDOM*, conforme mostrado na Figura 8.4.




(a) Função Himmelblau em curvas de nível.



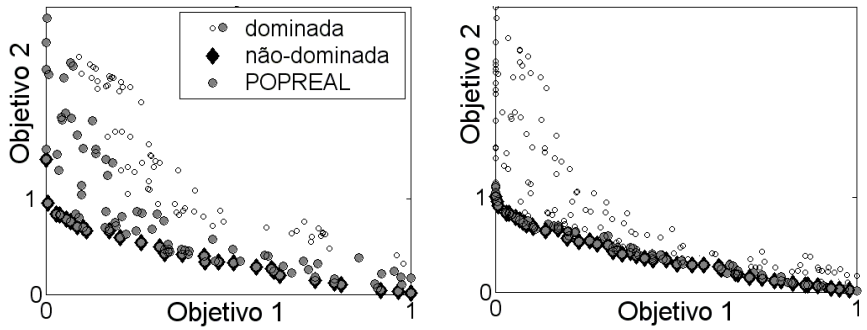

(b) Soluções não-dominadas encontradas AGMO.




(c) Fronteiras não-dominadas

Figura 8.4 – Himmelblau transformada em problema multiobjetivo.

Percebe-se, na Figura 8.5, que a população *POPREAL* é constituída pelas soluções que estão em torno da fronteira não-dominada (lembrando que são escolhidas aquelas que possuem os índices de dominância *IDOM* mais baixos). Isto acelera e facilita a convergência do processo de otimização na direção de uma fronteira Pareto bem definida.




(a) POPREAL no início do processo evolutivo

(b) POPREAL no final do processo evolutivo

Figura 8.5 – Himmelblau – verificando montagem de POPREAL.

## 8.4 Conclusões sobre o AGMO

As funções analíticas apresentados neste capítulo tiveram por objetivo não só testar o algoritmo proposto, mas também explicar mais concretamente os diferentes procedimentos utilizados.

O AGMO mostrou-se uma ferramenta extremamente eficiente, sendo sua principal vantagem às inúmeras possibilidades que ele oferece para compreensão do problema.

Como já dito, o inconveniente desta metodologia é o custo computacional necessário para a descoberta das soluções eficientes, o qual, em muitos problemas, pode ser superior às metodologias de otimização tradicionais (quando estas são capazes de os resolverem).

Este custo computacional excedente é devido principalmente a maior generalidade do AGMO. Outras considerações são:

O acompanhamento do processo de otimização deve ser constante, feito pelo projetista ou dinamicamente pela própria rotina computacional, se possível. A aquisição de informações deve ser a prioridade no início de qualquer processo quando não se conhece o problema. Somente compreendendo plenamente o problema será possível utilizar toda a potencialidade do método de otimização, seja ele qual for, obtendo assim soluções mais próximas das ideais.

Em problemas reais, quando se trabalha com objetivos muito concorrentes, é possível que os extremos da fronteira Pareto-ótima não tenham valor prático: eventualmente, valores muito baixos ou muito altos dos objetivos podem não ser interessantes. A Figura 8.6 ilustra tal situação. Isto pode ser contornado de duas maneiras:

- por medidas restritivas durante a avaliação da solução; se a avaliação for menor que um valor mínimo, a solução é penalizada;

- pela redução do espaço de busca: novos limites são estipulados a partir das soluções que estão dentro da região de interesse. O processo evolutivo é reiniciado com uma nova população criada aleatoriamente dentro desses novos limites, mais aquelas soluções pré-selecionadas.

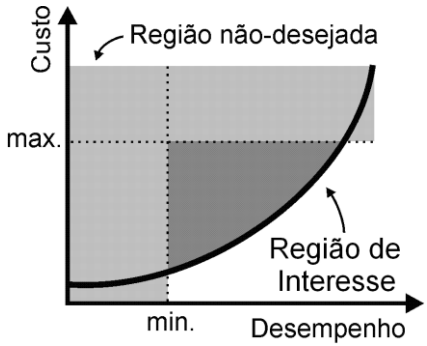



Figura 8.5. Redução do espaço de busca.

Na verdade, estes procedimentos funcionam como critérios de decisão durante o processo evolutivo, conforme discutido no capítulo 2. Estes artifícios podem ser utilizados em conjunto quando forem necessários.

Como complemento ao parágrafo anterior, pode-se acrescentar ainda que os limites de cada variável são determinados pela mínima experiência do projetista sobre o problema (por exemplo, limites construtivos). Se não existe experiência sobre o problema tratado, deve-se optar por limites amplos. Obviamente isso aumenta a dificuldade de resolução do problema.

Um controle com saturação nos limites das variáveis é recomendável na otimização de problemas reais. Este controle é entendido como o procedimento que corrige indivíduos que escapam dos limites propostos. A saturação seria o método que recoloca o indivíduo exatamente no limite imposto. Desta maneira, tem-se um melhor entendimento e um maior controle do processo de otimização. Estudar a evolução das variáveis é importante.

Pequenas mudanças eventuais na função objetivo ajudam na evolução do processo. A intenção é introduzir uma perturbação que force uma reorganização (adaptação) da população. Isto é similar ao que ocorre na natureza, em que perturbações no ecossistema podem gerar indivíduos mais bem adaptados às dificuldades. Evidentemente essas mudanças devem ser mínimas e, de certa maneira, complementares.

A variação dinâmica das probabilidades de cruzamento e mutação também tem por objetivo causar perturbações na população corrente através do aumento da 'pressão' dos operadores de cruzamento e mutação. Este procedimento age sobre as variáveis de otimização e não sobre os objetivos, como proposto no parágrafo anterior.

É sabido que os métodos estocásticos têm como principal desvantagem a necessidade de inúmeras avaliações do problema para a sua resolução. Aumentando-se sua complexidade, maior será o tempo necessário para a sua otimização. Os AGs podem levar vantagem neste aspecto se forem equacionados para trabalhar usando

processamento paralelo. Como suas operações são repetidas, ou seja, como cada indivíduo é avaliado de maneira independente, o paralelismo é direto.

**Referências**

[1] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," In: First International Conference on Genetic Algorithm, Lawrence Erlbaum, New Jersey, pp. 93-100, 1985.

[2] A. H. F. Dias, Algoritmos Genéticos Aplicados a Problemas com Múltiplos Objetivos, Dissertação (Mestrado em Engenharia Elétrica) – UFMG, Belo Horizonte, 2000.

[3] K. Deb, Multi-objective genetic algorithm: problem difficulties and construction of test functions. Technical Report n. CI-49/98, Department of Computer Science/XI, University of Dortmund, Dortmund, 1998.

# *Análise de sensibilidade*

Na concepção de dispositivos, o papel da análise de sensibilidade é verificar quão estável é o desempenho de soluções otimizadas quando as mesmas sofrem perturbações inevitáveis em projetos reais.

Essas variações podem ser erros inerentes à construção e montagem ou ainda mudanças nas condições de operação. Por exemplo, a obtenção de uma asa de avião que produza um ganho fenomenal em eficiência aerodinâmica pode não significar grande vantagem se, com a variação de alguns milímetros em sua posição (causada por trepidação, por exemplo), este ganho seja completamente diferente.

A Figura 9.1 ilustra a necessidade de buscar soluções robustas. Neste exemplo, a solução $S_2$ é mais estável que $S_1$, pois, quando sujeita aos mesmos valores de perturbações, tem seus objetivos menos alterados.

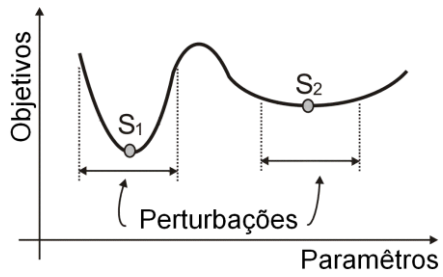



Figura 9.1. Análise de Sensibilidade buscando soluções robustas.

Como já visto nos capítulos anteriores, o AGMO produz um vasto conjunto de soluções, o qual reflete o comportamento das funções objetivo. Após o processo evolutivo, o engenheiro escolhe a solução final dentre as inúmeras possibilidades. A ideia é utilizar estudos complementares de maneira a qualificar as soluções perante sua sensibilidade às perturbações, facilitando a decisão final e o aprendizado do comportamento do problema.

São duas abordagens apresentadas neste texto: análise de sensibilidade baseada em uma métrica que relaciona objetivos e parâmetros; e análise de sensibilidade sobre especificações fixas do projeto, verificando a confiabilidade da arquitetura ou projeto.

## 9.1 Análise de Sensibilidade sobre os parâmetros de otimização

Este procedimento é um estudo sobre o desempenho do dispositivo quando suas variáveis de otimização são submetidas a perturbações decorridas da imprecisão na construção, por exemplo.

Entre as várias metodologias de análise de sensibilidade existentes, a constante de Lipschitz pode ser utilizada [1]. Para uma dada solução, esta constante é definida como a taxa de variação máxima dos objetivos dentro de um domínio finito de perturbações.

Outro procedimento, baseado no método de elipsoides interiores, é utilizado para aproveitar a grande quantidade de soluções geradas pelos métodos estocásticos [2].

Nos dois procedimentos citados, uma avaliação precisa da sensibilidade exige grande esforço computacional, além da necessidade do gradiente da função objetivo em algumas situações.

De forma concisa, uma explanação sobre os fundamentos das metodologias clássicas para este tipo de análise de sensibilidade é:

### 9.1.1 Metodologias clássicas para análise de sensibilidade

Os procedimentos clássicos de análise de sensibilidade, como já dito, exprimem a informação sobre o comportamento de um dispositivo (função) quando suas variáveis são desviadas dos valores projetados [1][3]. Uma maneira bem intuitiva e muito utilizada para a medição de sensibilidade é o cálculo da máxima variação de uma função dentro de um domínio (*D*). Esta medida de sensibilidade é definida como:

$$V_D(F) = \underset{\vec{a},\vec{b}\in D}{max}\left|F(\vec{a}) - F\left(\vec{b}\right)\right| \quad . \tag{9.1}$$

Tal variação responde uma questão frequentemente feita pelo projetista: qual o desvio máximo nos objetivos de um dispositivo otimizado quando as suas variáveis de otimização permanecem dentro de um dado domínio de perturbações? Note que esta métrica é intrinsecamente ligada ao tamanho do domínio.

De maneira a obter uma métrica de sensibilidade com o significado de variação por perturbação, a variação do funcional pode ser normalizada por uma métrica do próprio domínio *D*:

$$\mu_D(F) = \underset{\vec{a},\vec{b}\in D}{max}\frac{\left|F(\vec{a})-F(\vec{b})\right|}{N_D} \quad . \tag{9.2}$$

Outra possibilidade é a 'taxa de máxima variação média':

$$\delta_D(F) = \underset{\vec{a},\vec{b}\in D}{max}\frac{\left|F(\vec{a})-F(\vec{b})\right|}{\left\|\vec{a}-\vec{b}\right\|} \quad , \tag{9.3}$$

a qual é conhecida como constante de Lipschitz de uma função em *D*.

Esta métrica tem uma característica muito interessante. Em ao menos um ponto (quando $\vec{a} \to \vec{b}$), ela é igual à máxima norma do gradiente da função dentro do domínio. Uma definição alternativa da mesma métrica de sensibilidade, para funções $C^1$ (que possuem primeira derivada), é dada por:

$$\delta_D(F) = \max_{\vec{a} \in D} \left\| \nabla F(\vec{x})|_{\vec{x}=\vec{a}} \right\| \quad , \tag{9.4}$$

Considerando uma função contínua $C^1$ em um domínio infinitesimal $D$ em torno do ponto $\vec{p}$, a $\delta_D(F)$ torna-se $\left\| \nabla F(\vec{x})|_{\vec{x}=\vec{p}} \right\|$, o qual é provavelmente a métrica para sensibilidade mais utilizada. Para uma função contínua $C^2$ (com derivadas até a segunda ordem) em um domínio infinitesimal $D$ em torno do ponto $\vec{p}$, $\delta_D(F)$ é proporcional ao máximo autovalor da Hessiana associada ao ponto $\vec{p}$. Desta maneira, a métrica $\delta_D(F)$ é uma generalização de medidas de sensibilidade infinitesimais convencionais. Em todos os casos aqui discutidos, as variáveis de otimização devem ser normalizadas de acordo com as suas precisões. A métrica $\delta_D(F)$ caracteriza sensibilidade em domínios finitos, no sentido que, sendo conhecido o domínio, ela pode ser usada para definir os limites para $V_D(F)$. O cálculo de qualquer uma das medidas apresentadas pode ser definido como um problema de otimização que, para ser resolvido corretamente, requer a determinação dos valores de máximo e mínimo da função dentro do domínio finito estipulado, ou como alternativa, o cálculo do gradiente máximo da função.

Métodos clássicos de otimização são usualmente empregados para a determinação desses máximos e mínimos, já que o cálculo do gradiente pode não ser evidente. Assim, tem-se um processo de otimização dentro do outro: o principal, que visa à maximização ou minimização dos objetivos principais; e, quando desejado, uma otimização 'inversa' para a determinação da informação dos valores mínimos ou máximos, respectivamente, para análise de sensibilidade. Este procedimento torna o estudo de sensibilidade durante o processo de otimização bastante custoso computacionalmente.

### 9.1.2 Análise de sensibilidade por dados intrínsecos ao AGs.

É possível um estudo de sensibilidade baseado em uma métrica diretamente calculada dos dados gerados pelo AGMO, não necessitando, entretanto, de qualquer esforço computacional adicional no que diz respeito a novas avaliações das soluções ou cálculos de derivadas.

Como mostrado na Figura 9.2, ao final do processo evolutivo ocorre uma concentração de soluções em volta da fronteira Pareto, o que torna razoável o uso destes dados para a realização de análises referentes ao comportamento dos valores de objetivos em relação às variáveis de otimização.

O método proposto aqui também permite a identificação de qual parâmetro é o mais importante para a manutenção da 'otimalidade' do dispositivo. A efetividade deste procedimento está condicionada a um espaço de busca apropriadamente representado no entorno da solução estudada, ou seja, não deve ocorrer uma convergência prematura do processo evolutivo; utiliza-se todo o conjunto de indivíduos gerado desde a primeira geração.

A metodologia proposta tem os seguintes passos de execução:

- Primeiramente, determina-se os valores máximos para perturbações nos parâmetros de otimização, definindo o tamanho de um domínio finito de estudo no espaço dos parâmetros. As amplitudes destas variações são especificadas pelo projetista, sendo estipuladas segundo os possíveis valores de desvios que podem ocorre no dispositivo devido às imperfeições na construção, montagem, dilatações térmicas, etc;

- Escolher algumas soluções (V*) segundo algum interesse particular (ou um grupo de soluções amostradas na fronteira Pareto de maneira a estudar a sensibilidade do conjunto). No entorno de cada solução V* é estabelecido um domínio de estudo (D*) no espaço de parâmetros (onde se encontram os indivíduos),

no qual se pode estudar os desempenhos de cada objetivo dessas soluções constituintes, e os comparar com os objetivos de V*. O tamanho de D* é definido pelas amplitudes das perturbações determinadas anteriormente;

- Para conhecer a maior mudança de desempenho no domínio D*, faz-se necessário encontrar a maior degradação de cada objetivo no interior do domínio D*. Isto configura uma procura critério a critério, que busca a pior solução para cada objetivo;

- Se para cada objetivo o pior valor da alteração for aceitável, pode-se afirmar que a solução é estável para as exigências do projetista;

- De modo a comparar soluções ditas estáveis, pode-se calcular a 'distância' dos piores casos à solução em estudo. Quanto maior esta distância, ou seja, quanto mais longe ocorre o pior caso, mais estável é a solução.

A procura critério a critério pode ser feita utilizando as ferramentas tradicionais de otimização, como métodos baseados em derivadas ou mesmo o AG mono-objetivo, entretanto, isto pode ser muito custoso computacionalmente.

De maneira a reduzir este custo (eliminar a necessidade de novas avaliações de soluções), é possível utilizar as soluções geradas pelo processo evolutivo do AGMO.

A partir das amostras de Pareto ou qualquer outra solução em análise (*V**) e com base nos limites máximos para perturbações preestabelecidos, é possível verificar quais soluções dentre todas aquelas geradas pelo AGMO (*POPNDOM* + *POPDOM* + *POPDOMold*) estão contidas em *D**. Definido o grupo de soluções a ser estudado, basta procurar dentro deste conjunto os indivíduos com os piores valores de objetivos. A ideia é simples e resulta em uma noção satisfatória de sensibilidade, como será comprovado na resolução dos exercícios a seguir.

### 9.1.3 Teste – funções analíticas – parâmetros de otimização

A Figura 9.2a mostra as soluções geradas pelo AGMO na minimização simultânea de duas funções quadráticas ($f_1$ e $f_2$) com duas variáveis de otimização cada ($x_1$ e $x_2$):

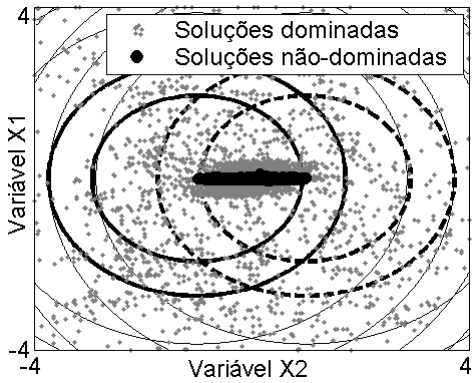
$$f_1 = (x_1 - 1)^2 + x_2^2 \qquad \text{e} \qquad f_2 = (x_1 + 1)^2 + x_2^2. \qquad (9.5)$$

A região com soluções ótimas é uma linha que conecta os vértices das duas parábolas. A Figura 9.2b mostra a fronteira não dominada deste problema encontrada pelo AGMO.

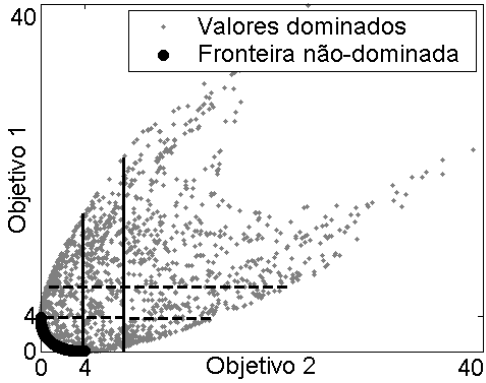
Neste exercício analítico, a solução escolhida para o teste de sensibilidade é a $V^*$ = [$x_1$ = 0,003387 , $x_2$ = -0,004788], a qual é a solução mais próxima dos objetivos [$f_1$ = 1 , $f_2$ = 1] encontradas pelo AGMO e que corresponde ao meio da fronteira.

A 'transformação geométrica' de um espaço para outro nem sempre é simples. De modo a explicar esta transformação, podem-se traçar sobre as Figuras 9.2(b) as linhas transpostas de mesmo valor das Figuras 9.2(a). A comparação das figuras permite compreender como se passa esta transformação. Sobre a fronteira Pareto deste exercício, verifica-se que existem sempre dois pontos de (a) que possuem a mesma imagem em (b).

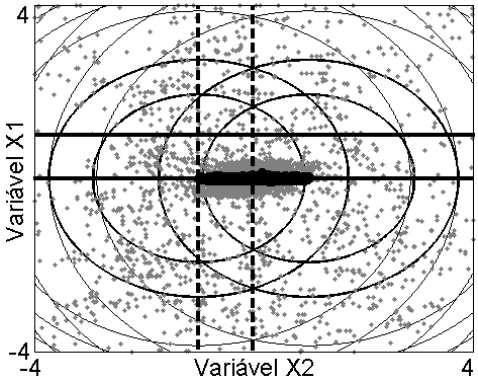
O domínio $D^*$ foi definido como sendo o desvio entorno de $V^*$ de intervalo [-0,2 , 0,2] para cada uma das variáveis. Determinados estes limites, buscou-se dentre as soluções constituintes deste domínio a pior solução para cada objetivo. Neste exercício, as piores soluções dos dois objetivos estão nas 'quinas' do domínio estipulado (Figuras 9.2c e 9.2c'). Isto porque o comportamento de ambos os espaços, objetivos e parâmetros, é uniforme. O projetista considera a solução como estável se estas piores avaliações forem aceitáveis.



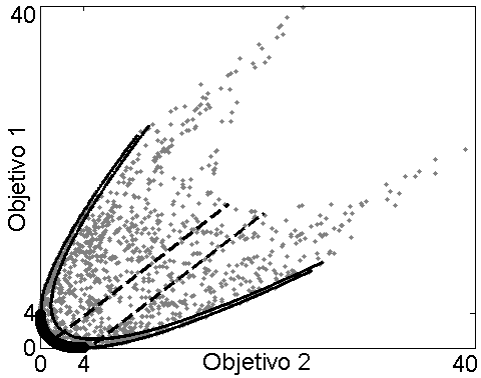

(a) espaço de parâmetros



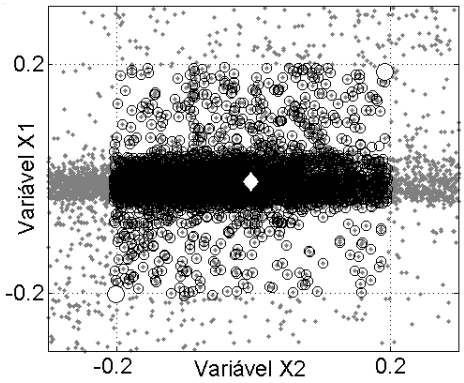

(b) fronteira não-dominada – espaço de objetivos



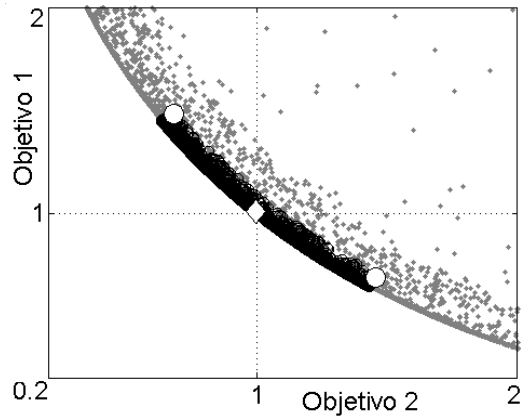

(a') espaço de parâmetros




(b') fronteira não-dominada – espaço dos objetivos




(a') espaço de parâmetros – detalhe




(b') espaço de objetivos – detalhe

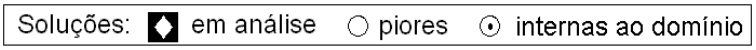



Figura 9.2. Duas parábolas – funções analíticas para o estudo de análise de sensibilidade.

Para comparar as soluções 'estáveis', buscando saber qual é a menos sensível entre elas, pode-se associar as variações nos objetivos com a distância correspondente no espaço dos parâmetros ($B_k(.)$): a solução analisada é estável se a distância é grande no domínio dos parâmetros e se ela é pequena no domínio dos objetivos. Este comportamento $B_k(.)$ associado ao ponto ótimo $V^*$ para cada função objetivo $f_k(.)$, dentro de um domínio de perturbação $D^*$, pode ser dado por:

$$B_k(V^*) = \max_{V_i, V_j \in D^*, i \neq j} \frac{|f_k(V_i) - f_k(V_j)|}{\|V_i - V_j\|} \quad , \tag{9.6}$$

onde $V_i$ e $V_j$ são soluções que estão dentro de $D^*$ (este cálculo é feito comparando duas à duas todas as soluções de $D^*$ ). Note que $D^*$ é definido por uma variação possível preestabelecida sobre $V^*$ ($n$ variáveis de otimização).

O cálculo de $B_k$ é simples – pois ele é feito com soluções já conhecidas; e confiável, desde que o AGMO tenha feito uma boa exploração do espaço $D^*$ (como disposto na Figura 9.2.a). Ademais, $B_k$ é um valor assimptoticamente similar à constante de Lipschitz quando o número de soluções dentro do domínio $D^*$ tende ao infinito.

Isto significa que $B_k$ é uma aproximação da taxa de variação máxima dos objetivos dentro de um domínio finito de perturbações. Resumindo, quanto maior $B_k$, mais instável será a função $f_k$ dentro do domínio $D^*$. De modo a obter $B_k$ com valor significativo, deve-se ter o cuidado de normalizar cada parâmetro por seu valor máximo em $V^*$ e cada objetivo por $|f_k(V*)|$.

### 9.1.4 Considerações sobre parâmetros de otimização

Para melhorar a confiança dos resultados obtidos com os métodos estocásticos (para o AGMO, é desejável confirmar e/ou melhorar a aproximação para a fronteira Pareto), existe a necessidade de repetir os processos evolutivos. Todas as soluções geradas durante estas execuções sucessivas podem ser utilizadas para a análise de sensibilidade, o que também melhora a qualidade deste estudo.

A análise de sensibilidade aqui apresentada foi tratada como um decisor *a posteriori*. Entretanto, este procedimento pode ser aplicado durante o processo evolutivo, obedecendo obviamente à necessidade de uma boa varredura do espaço de busca. De modo a acelerar o processo de otimização na direção de soluções estáveis, a informação de sensibilidade poderia ser utilizada para descobrir soluções instáveis e, então, puni-las de alguma maneira.

O método de análise descrito usa variáveis 'diretas' (comprimento, largura, ângulo, peso, valores de corrente ou tensão elétrica, etc.). No caso de parâmetros de otimização 'indiretos', como por exemplo os coeficientes de séries de expansão de um polinômio, o procedimento apresentado pode não ter um significado real. Nestes casos, é necessário criar uma parametrização para as perturbações, relacionando os possíveis desvios às variáveis utilizadas no processo.

## 9.2 Sensibilidade sobre os parâmetros predeterminados – modelo

A otimização multiobjetivo tem como principal ação encontrar as soluções ótimas de um problema segundo uma série de objetivos que o estabelecem. Os objetivos, os parâmetros a ajustar e as especificações fixas (demais características) constituem o modelo criado pelo projetista que representa o problema a ser otimizado. Saber caracterizar o problema e entender mudanças não previstas (em virtude de desvios em parâmetros pré-fixados) ajuda a compreender o problema.

O estudo do comportamento das soluções ótimas quando aplicadas às mudanças no modelo pode ajudar na escolha da solução final, bem como pode levar a um melhor entendimento do problema. Régnier [4] elabora um procedimento deste tipo, entretanto, os parâmetros fixos de seu modelo global de sistema são os diferentes modelos elementares do sistema. Ele relata, então, uma análise de sensibilidade pertinente ao modelo, já que ele testa a sensibilidade de uma solução ótima quando submetida às modificações de um modelo particular de subsistema.

Transportando esta ideia, a análise de sensibilidade sobre parâmetros fixos será aqui discutida. Ao contrário da seção 9.1, esta análise não diz respeito diretamente aos parâmetros de otimização, mesmo se a intenção é de ajudar nas escolhas de quais variáveis são mais importantes para a inserção no processo de otimização.

Como exemplo, cita-se o problema da antena embarcada em satélite. O projetista concebe o satélite para operar em um ponto fixo na órbita geoestacionária, mesmo sabendo que o satélite está sujeito a deslocamentos em sua posição, e esta deve ser corrigida de tempos em tempos: a posição orbital não faz parte dos parâmetros de otimização, mas estes parâmetros devem ser escolhidos para que uma variação orbital tenha a menor influência possível.

Outro exemplo, o engenheiro que otimiza as dimensões de um motor elétrico buscando maximizar seu desempenho, mas que, por razões

quaisquer, ocorrem variações constantes de tensão e corrente na sua alimentação. Estas são alterações que o projetista não controla, mas que ele pode estimar. Conhecendo as degradações resultantes nos objetivos, pode-se escolher como solução final aquela que sofre menos perturbação com mudanças em parâmetros ditos fixos.

Nas especificações do problema, é necessário atenção para distinguir aquelas que são dados, chamados aqui de parâmetros fixos, daquelas que são especificações que devem ser atendidas, ou seja, os objetivos ou restrições. A análise de perturbações sobre os parâmetros fixos ajuda na escolha da solução final.

Pode-se definir como objetivo suplementar de otimização a minimização das perturbações dos objetivos principais, associadas as variações das especificações fixas, mais isto aumenta a dificuldade de resolução do problema. Quanto maior o número de objetivos, mais difícil pode ser encontrar uma boa discretização da fronteira Pareto.

O procedimento é simples: todas as soluções não-dominadas e algumas dominadas (aquelas mais 'próximas' da fronteira – menor IDOM) são reavaliadas com as mudanças no modelo do problema. Isto resultará em um reposicionamento destas soluções no espaço dos objetivos. De modo a estimar a estabilidade da solução basta verificar:

$$d_{1\ldots n} = (x* - x_{1\ldots n}\ ,\ y* - y_{1\ldots n})\ ,\ \ D = \sum_{i=1}^{n} |d_i|\ \ , \tag{9.7}$$

onde $D$ é o desvio ocorrido por $n$ perturbações, $x^*$ e $y^*$ são dois objetivos da solução em estudo. Se as $n$ perturbações são correlatas, constituindo na realidade uma apenas, pode-se encontrar a sensibilidade aplicando (9.7) com $n$ = 1.

Se as perturbações são não correlatas, e é desejo obter um só valor de sensibilidade, então a expressão (9.7) é conveniente (quanto mais baixo for o valor $D$ calculado, mais estável é a solução).

## 9.3 Análise de sensibilidade de uma antena refletora embarcada em satélite – exemplo

Para esclarecer as duas metodologias de análise de sensibilidade propostas neste capítulo, usar-se-á o exercício de otimização de uma antena refletora embarcada em satélite com reuso de frequências. Aqui, optou-se por trabalhar com dois objetivos, a saber, a maximização do ganho diretivo em duas áreas de cobertura distintas (GE e GI). A Figura 9.3 apresenta as soluções encontradas.

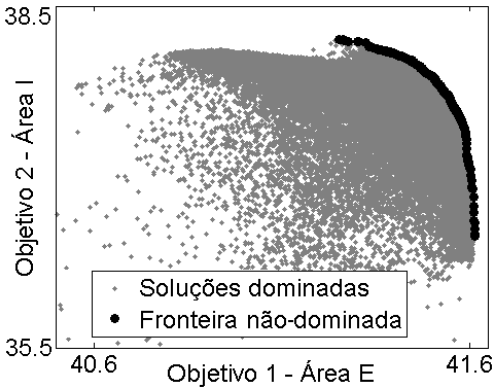



Figura 9.3. Soluções encontradas – Antena para satélite [dBi].

Após muitas simulações para garantir a fronteira não-dominada e uma vez entendido o comprometimento entre os objetivos, neste caso: o aumento do ganho diretivo em uma área implica na diminuição do ganho na outra área; podem ser aplicados os estudos complementares abordados neste capítulo, auxiliando o entendimento do problema e facilitando a escolha da solução 'ótima' na decisão final.

### 9.3.1 Sensibilidade – parâmetros – antena refletora embarcada

Para este tipo de estudo, foram escolhidas duas soluções não-dominadas: $S_1$, com GE = 41,605dBi e GI = 36,901dBi; e $S_2$, com GE = 41,4770dBi e GI = 37,9340dBi, conforme Figura 9.4.

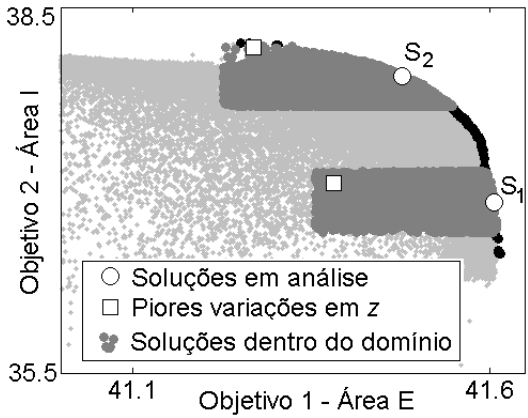



Figura 9.4. Análise dos parâmetros – antena para satélite [dBi].

A conformação da superfície refletora (curvatura) é dada por um polinômio com coeficientes de expansão, portanto, não são variáveis diretas. Isto significa que variações em alguns parâmetros não se traduzem em mudanças facilmente previsíveis na forma da superfície refletora uma vez que a série de expansão é global. Isto exige, para o estudo de sensibilidade em curso, uma parametrização paralela capaz de simular possíveis desvios decorridos de imperfeições na construção da superfície refletora e/ou dilatações térmicas, por exemplo.

Para estudar perturbações reais nas ondulações do refletor, é possível trabalhar de maneira reversa: aceitando uma variação de ±0,25dBi em cada objetivo como tolerável, um domínio finito de soluções ($D^*$) é definido em torno de cada solução selecionada $S_1$ e $S_2$ (Figura 9.7).

Cabe um esclarecimento: na proposta inicial, as perturbações ocorrem nos parâmetros de otimização, portanto, D* é estabelecido no espaço de parâmetros; na proposta reversa, D* é definido no espaço de objetivos.

Dentro de cada domínio é procurada a maior disparidade nas superfícies entre a solução selecionada e as demais em termos da diferença medida na coordenada $z$ (profundidade do refletor). No domínio onde ocorrer a maior diferença, com as mesmas dimensões de $D^*$ obviamente, a solução é considerada a mais estável.

Neste exercício, a solução $S_2$ (com uma variação máxima de 0,582λ) é mais estável que $S_1$ (variação máxima de 0,438λ) pois, com uma diferença maior de sua forma, respeita os limites de perturbações nos objetivos propostos. Note que nos dois casos estudados, as piores soluções não estão nos extremos dos domínios, como o acontecido no problema das funções analíticas. Neste exemplo, o comportamento dos espaços de parâmetros e objetivos não é idêntico. Analisar os dois espaços de busca (parâmetros e objetivos) é, portanto, fundamental.

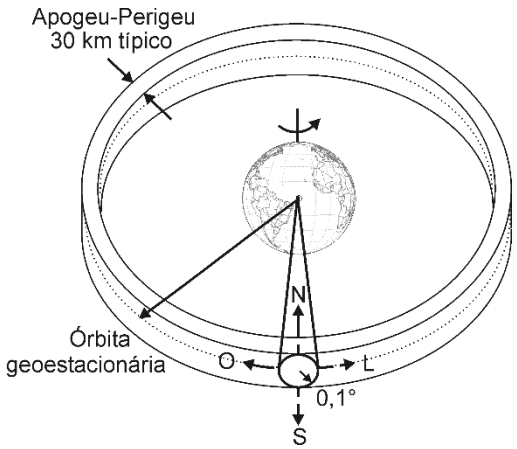
### 9.3.2 Sensibilidade – especificações fixas – antena refletora

É sabido que a posição do satélite na órbita geoestacionária não é fixa. Ela sofre variações que necessitam ser corrigidas de tempos em tempos de modo a manter o satélite entre limites pré-especificados. As oscilações orbitais factíveis são ilustradas na Figura 9.5a.

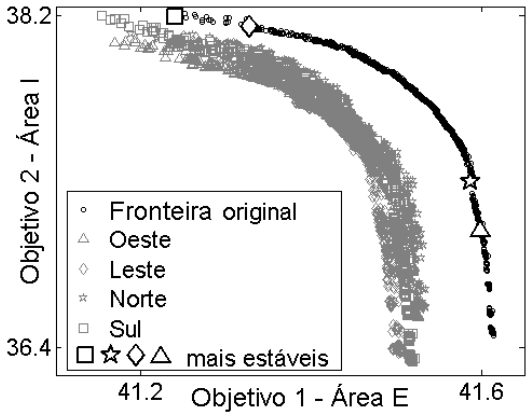
Na parte final do período de utilização do satélite, estes limites podem ser estendidos para economizar combustível e prolongar a vida útil do satélite, contudo aceitando uma degradação na sua cobertura. A análise de sensibilidade concernente ao modelo será aplicada com a intenção de estimar quanto estas perturbações degradam as avaliações das soluções otimizadas.

Neste exercício, optou-se por perturbações no valor de deslocamento angular de $\Delta\phi = +0{,}1°$, de modo que o satélite não apontará mais exatamente para o centro da região de interesse. Todas as especificações do sistema permanecem iguais, menos a posição orbital angular, como mostra a Figura 9.5a.

Isto significa que o diagrama de radiação não estará mais centrado sobre a área de cobertura, e eventualmente, as bordas das áreas serão prejudicadas. A variação em distância (apogeu – perigeu) não será contemplada neste caso. Para efeito de predição do ganho diretivo, a perturbação assumida implica em um recálculo dos pontos em azimute × elevação do mapa de amostragem existente.




a) oscilações orbitais de um satélite.




(b) variação dos objetivos devido às oscilações

Figura 9.5. Fronteiras não-dominadas após mudanças na especificação do modelo: projeto reuso de frequências [em dBi].

A Figura 9.5b apresenta a mudança das avaliações de um grupo de soluções (todas as soluções não-dominadas mais aquelas dominadas com *IDOM* até 10). Com as novas fronteiras conhecidas, é possível calcular o desvio (9.7) de modo a encontrar a solução mais estável, a saber, aquela que possuir as menores variações nos objetivos. A

Tabela 9.1 mostra o mínimo desvio dentre aquelas soluções avaliadas para cada perturbação.

Tabela 9.1. Análise de sensibilidade do modelo – antena refletora.

| Perturbações ($\Delta\phi$) | | | | Mínimo | Mínimo ganho diretivo | |
|---|---|---|---|---|---|---|
| Norte | Sul | Leste | Oeste | Desvio | GE | GI |
| 0,1 | 0 | 0 | 0 | 0,0130 | 41,599 | 37,028 |
| 0 | 0,1 | 0 | 0 | 0,0155 | 41,327 | 38,140 |
| 0 | 0 | 0,1 | 0 | 0,0109 | 41,587 | 37,300 |
| 0 | 0 | 0 | 0,1 | 0,0094 | 41,240 | 38,195 |
| [°] | | | | (adimensional) | [dBi] | |

Com todos os dados apresentados, é possível concluir que a mudança na posição do satélite altera a 'otimalidade' das soluções encontradas. Baseado nesta análise, concebe-se que em um projeto real seria possível recomeçar o processo de otimização considerando esta oscilação orbital como uma restrição ou como um objetivo a minimizar. Evidentemente, isto aumenta a dificuldade de resolução do problema, como já discutido.

### 9.3.3 Solução final

De modo a considerar as duas análises de sensibilidade (parâmetros e modelo) na escolha da solução final, o seguinte procedimento foi utilizado: o cálculo das novas avaliações para o 'estudo do modelo' (mantendo $\Delta\phi = 0{,}1°$) foi feito somente para as soluções que, para uma variação máxima na superfície de 0,25λ, tiveram uma perturbação menor ou igual a 0,15dBi em cada objetivo (sensibilidade sobre os parâmetros).

Sendo assim, a solução mais estável encontrada possui GE = 41,525dBi e GI = 37,773dBi. É importante mencionar que esta solução não é uma não-dominada.

A solução 'ótima' escolhida, segundo preferências do projetista e dos estudos de sensibilidade, faz parte do conjunto de soluções dominadas, e ela teria sido rejeitada sem os estudos de sensibilidade apresentados neste capítulo. Isto evidencia ainda mais a importância dos estudos de sensibilidade realizados.

Conhecendo a fronteira Pareto (ou seja, o comprometimento entre os objetivos) e todos estes estudos complementares, o engenheiro pode escolher a solução final que ele julga 'ótima', ou reiniciar o processo de otimização com diferentes especificações, para usar como vantagem seu melhor entendimento do problema.

Este capítulo mostrou que, quando são considerados aspectos de ordem prática, é necessário verificar se a qualidade de uma solução permanece aceitável quando os parâmetros a ela associados sofrem variações. Conforme os exercícios realizados, fica evidente a importância dos estudos aqui apresentados.

Fica também evidente as inúmeras possibilidades de análises, estudos e considerações que podem ser realizadas com o grande número de soluções encontradas pelo AGMO. Esta é a grande vantagem deste método estocástico em comparação as clássicas metodologias de otimização. Saber lidar com esta vasta gama de informações é importante e necessário para a obtenção da melhor solução possível.

**Referências**

[1] A. C. Lisboa, Análise de estabilidade em otimização para domínios finitos de funcionais de Lipschitz, Dissertação (Mestrado em Engenharia Elétrica) – UFMG, Belo Horizonte, 2003.

[2] R. H. C. Takahashi, J. A. Ramirez, J. A. Vasconcelos, et al., "Sensitivity analysis for optimization problems solved by stochastic methods," IEEE – Transactions on Magnetics, v. 37, n. 5, pp. 3566–3569, Set. 2001.

[3] D. A. G. Vieira, A. C. Lisboa, R. R. Saldanha, J. A. Vasconcelos and R. H. C. Takahashi, "Multi-objective sensitivity analysis in finite domain of a Yagi-Uda antenna optimal design," In: IEEE – 11th CEFC – Conference on Electromagnetic Field Computation International Symposium, Seoul, Korea, Jun. 2004.

[4] J. Régnier, Conception de Systèmes Hétérogènes en Génie Electrique par Optimisation Evolutionnaire Multicritère, Tese (Docteur de l'INPT) – Institut National Polytechnique de Toulouse, Toulouse, 2003.

# *OPTIMAL – software para o Ensino de Otimização em Engenharia*

A utilização de programas iterativos para o ensino de todo tipo de assuntos em engenharia vem ganhando destaque. Os principais objetivos dessas ferramentas multimídia são facilitar e tornar mais interessante o processo de aprendizagem ao estudante. Entretanto, os programas didáticos consagrados à otimização são poucos.

Aqui se apresenta uma ferramenta interativa para o ensino de otimização em engenharia, chamada *OPTIMAL*, cuja principal ideia é mostrar como diferentes metodologias funcionam além de permitir comparações entre elas. Somente a programação não-linear é considerada (métodos determinísticos e estocásticos).

OPTIMAL está disponível na integra em:

HTTPS://GITHUB.COM/PECCE-IFSC/OTIMIZAÇÃO

As características principais da ferramenta são:

- Liberdade total: o usuário pode alterar todos os parâmetros do programa, seja por opções na tela, seja incluindo novos problemas por arquivos ou ainda modificando o código fonte;

- Possibilidade de salvar todos os dados e todas as variações das simulações, o que permite aos educadores demonstrar em sala de aula as características de cada método. Por exemplo, pode-se optar por apresentar alguma simulação particular ou casos onde a

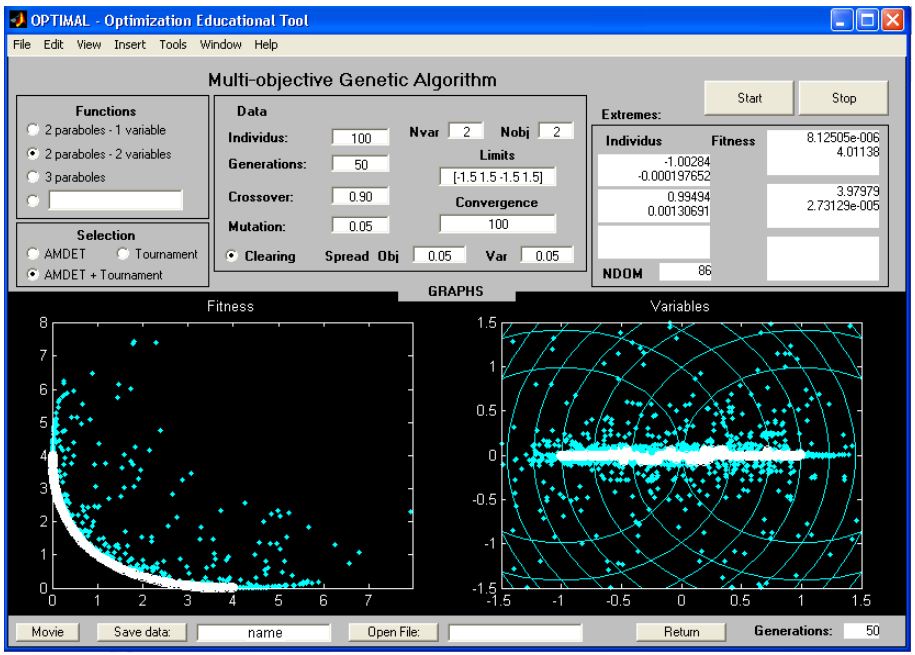
metodologia falhou (este procedimento tem especial importância para os procedimentos probabilísticos);

- Comparação entre métodos: o estudo de diferenças entre as famílias de métodos determinísticos e estocásticos é possível. Conhecer em que tipo de problema cada método é mais eficiente, suas vantagens e desvantagens, são aspectos importantes no momento da escolha de qual método utilizar;

- Otimização multiobjetivo: esclarece a importância de se considerar vários objetivos conflitantes presentes nos problemas de otimização reais;

- Análise de sensibilidade: permite aos estudantes obter uma ideia geral da ação de perturbações em projetos práticos.

A Figura 10.1 mostra a minimização de um problema com duas funções quadráticas de modo a ilustrar o OPTIMAL.



# *Continuação dos estudos*

Este é um livro de introdução à otimização com computação evolutiva, com maior destaque para a técnica dos Algoritmos Genéticos. Ao longo dos capítulos estão citadas dezenas de referências para um aprofundamento nos temas, conforme interesse.

Otimização está inserida num contexto que pode ser chamado de computação científica, que desenvolve modelos matemáticos e técnicas de soluções numéricas para analisar e resolver problemas científicos e de engenharia.

Computação científica é uma área em franco desenvolvimento, devido a dois motivos principais: (i) a crescente capacidade de processamento computacional; e (ii) a crescente quantidade de dados disponíveis (sensoriamento cada vez mais barato e a sua integração pela internet).

O desafio sempre será obter a modelagem matemática mais próxima possível da realidade, conforme discutido no início deste livro. Junta-se a isto agora, a grande quantidade de dados a serem interpretados. Transformar dados em informação útil para uma tomada de decisão mais assertiva é o estado da arte hoje.

Neste contexto, sugere-se ao leitor continuar seus estudos por:

- **Pesquisa operacional** é o uso de modelos matemáticos, estatística e algoritmos para ajudar na tomada de decisões. Muito utilizada dentro da administração e da engenharia de produção, a pesquisa operacional envolve técnicas como: teoria da dualidade e análise de sensibilidade, teoria de jogos, processo de decisão de Markov e teoria de filas [1][2].

- **Inteligência artificial ou aprendizado de máquina** (*machine learning*) são algoritmos construídos de forma a reconhecer padrões, aprender e agir a partir de dados, sem necessitar de instrução humana explícita [3]-[5]. De forma geral, o aprendizado pode ser supervisionado ou não supervisionado. Técnicas para aprendizado supervisionado buscam responder um objetivo, ou seja, há uma variável explícita a ser respondida (*K-nearest neighbors* e *support vector machine* são exemplos de técnicas). Métodos não supervisionados buscam identificar grupos ou padrões a partir dos dados, sem um objetivo específico a ser alcançado (mapas de Kohonen talvez seja a estratégia mais difundida). As chamadas redes neurais artificiais (RNA) podem ser implementadas com as duas orientações.

O Grupo de Pesquisa em Computação Científica para Engenharia (PECCE), vinculado ao Instituto Federal de Santa Catarina (IFSC), do qual o autor deste livro é integrante, desenvolve aplicações para a indústria em todos os temas aqui citados. Entre em contato.

Bons estudos!

[1] F. S. Hiller & G. J. Leiberman, Introdução à Pesquisa Operacional, Mc Graw Hill, 9th Ed., 2017. ISBN 9788580551181.

[2] M. Arenales & V. Armentano, Pesquisa Operacional para Cursos de Engenharia, Elsevier, 2015. ISBN 9788535271614.

[3] G. Huang, G. B. Huang, S. Song, K. You, Trends in extreme learning machines: A review, Neural Networks 61 (2015) pp. 32-48. doi:10.1016/j.neunet.2014.10.001.

[4] V. M. S. Raschka, Python Machine Learning: Machine Learning and Deep Learning with Python, 1st Ed., Packt Publishing, 2017. ISBN 9781789955750.

[5] Khaled Bayoudh, From Machine Learning to Deep Learning, 1st Ed., 2017, Ebook, ISBN: 9781387465606.

INSTITUTO
FEDERAL
Santa Catarina