Universidade Federal de Minas Gerais
Departamento de Matemática

Notas de Aula

Otimização Escalar e Vetorial

Volume 2: Otimização Escalar

Professor: Ricardo H. C. Takahashi

Belo Horizonte, Janeiro de 2007

Conteúdo

Parte II

Otimização Escalar

Capítulo 5

Interpretação Geométrica dos Processos de Otimização

Neste capítulo, iremos discutir, de maneira preliminar, *o que são* os problemas de *Otimização*, com base sempre em funções matemáticas simples, de apenas duas variáveis, que permitem portanto sua representação gráfica em três dimensões. Iremos mostrar como diferentes tipos de funções irão requerer diferentes estratégias de otimização e, de maneira intuitiva, iremos discutir os princípios que se encontram por trás dos métodos de otimização que serão estudados em detalhe nos próximos capítulos.

5.1 O Jogo da Otimização

A *Otimização*, sob o ponto de vista prático, se trata do conjunto de métodos capazes de determinar as melhores configurações possíveis para a construção ou o funcionamento de sistemas de interesse para o ser humano. Estamos falando da aplicação de uma mesma teoria, com um mesmo conjunto de métodos e ferramentas, quando:

- um engenheiro eletricista procura o melhor projeto possível para uma antena ou para um motor elétrico;
- um engenheiro de controle e automação procura o melhor ajuste possível para os controles de um determinado processo industrial;
- um engenheiro de produção busca a melhor configuração possível para encadear as etapas de fabricação de um produto;

- um matemático computacional estuda modelos quantitativos de epidemias, procurando determinar as melhores políticas de vacinação;
- um cientista da computação estuda o desempenho de uma rede de computadores, e tenta estabelecer a melhor estratégia de tráfego de informação possível, visando maximizar o fluxo global de informação nessa rede;
- um economista procura o melhor *portfolio* de investimentos, que maximiza a expectativa de retorno financeiro;
- um veterinário ou zootecnista procura determinar a melhor política de compras e vendas das cabeças de um rebanho de gado.

Apesar dos contextos completamente distintos, todos estes problemas (e muitos outros) uma vez formulados matematicamente, possuem exatamente a mesma estrutura, e sua solução é obtida essencialmente através da utilização do mesmo conjunto de técnicas: a *Otimização*.

5.1.1 Formulação do Problema de Otimização

Evidentemente, em cada contexto distinto, há um conjunto de informações que cada especialista de cada área deve conhecer, que lhe permite obter uma descrição matemática de cada problema, a partir da situação concreta em questão. Uma vez construído o modelo do problema[1], chegamos sempre[2] à formulação característica do problema de otimização:

$$\begin{aligned} \mathbf{x}^* = & \ \arg \min_{\mathbf{x}} \ f(\mathbf{x}) \\ & \\ \text{sujeito a: } & \begin{cases} \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ \\ \mathbf{h}(\mathbf{x}) = \mathbf{0} \end{cases} \end{aligned} \tag{5.1}$$

Vamos primeiro entender o que significa essa expressão. Como convenção que adotaremos ao longo de todo este livro, as letras em negrito significam

[1] O leitor não deve se enganar: a construção do modelo matemático do problema muitas vezes é a parte mais difícil de todo o processo. Estamos saltando esta parte porque a Otimização começa exatamente quando o modelo da situação está pronto.

[2] OK, você está certo: *quase* sempre.

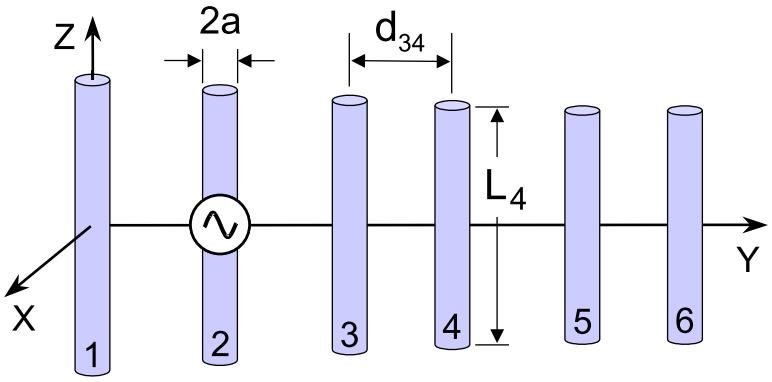


Figura 5.1: Representação esquemática de uma antena de seis elementos. Na prática, uma antena real corresponde a uma estrutura metálica com geometria igual à do diagrama.

grandezas vetoriais (ou seja, que representam conjuntos de vários valores) enquanto as letras sem negrito significam grandezas escalares (que representam um único valor).

O vetor de variáveis de otimização

O vetor $\mathbf{x}$ é o *vetor de variáveis de otimização*, que representa o conjunto das variáveis cujos valores procuramos especificar através do processo de otimização. Por exemplo, suponhamos que um engenheiro está projetando uma antena de seis elementos, como aquela representada na figura 5.1. Projetar essa antena significa escolher os tamanhos dos seis elementos da antena e as cinco distâncias que separam esses elementos entre si. São portanto onze valores a serem representados pelo vetor $\mathbf{x}$, o qual seria dado, nesse caso, por:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{11} \end{bmatrix} \tag{5.2}$$

Uma vez especificados esses onze valores, para construir a antena basta "seguir a receita" implícita em $\mathbf{x}$: cortar hastes de metal com cada um dos seis comprimentos especificados, e prender essas hastes de acordo com cada uma das cinco distâncias especificadas.

Nós optamos pelo exemplo da antena por ser ele bastante simples para ilustrar o fato de que os elementos do vetor $\mathbf{x}$ possuem usualmente um significado bastante concreto, ligado à estrutura do problema que está sendo representado. No caso do economista compondo um *portfolio* de investimentos, outro exemplo também simples, o vetor seria composto pelo montante aplicado em cada um dos tipos de aplicação diferentes (ações, títulos com correção cambial, ouro, etc).

Em ambos os casos, o vetor $\mathbf{x}$ contém as variáveis cujos valores devemos escolher para atingirmos a melhor solução possível do problema. Se supusermos que o capital financeiro é uma grandeza continuamente variável[3], teremos, tanto no caso da antena quanto no do *portfolio* de investimentos, vetores $\mathbf{x}$ contendo variáveis pertencentes ao conjunto dos números reais. De maneira genérica, se o vetor $\mathbf{x}$ possui n variáveis reais, dizemos que $\mathbf{x} \in \mathbb{R}^n$.

Nem sempre o vetor de variáveis de otimização é composto de variáveis reais. Muitas vezes, as variáveis são números inteiros, por exemplo, quando estamos estabelecendo quantas máquinas serão utilizadas para trabalhar em determinada etapa de um processo de fabricação. Outras vezes as variáveis são até mesmo binárias: por exemplo, ao se estudar o problema da formação de uma malha viária ligando diversas cidades, deve-se decidir se determinada estrada ligando diretamente duas cidades será ou não será construída (só existiriam, nesse caso, as opções *sim* ou *não*).

A diferença mais importante entre os problemas de otimização, que conduz a técnicas de resolução com fundamentações bastante distintas, é aquela que separa os problemas em que as variáveis de otimização são reais dos problemas que apresentam variáveis de otimização discretas (binárias ou inteiras). Neste livro, iremos estudar apenas os problemas com variáveis reais.

A função objetivo

A próxima entidade presente na expressão (5.1) que devemos discutir é a chamada *função objetivo*, $f(\cdot)$. Essa entidade representa o índice de desempenho do sistema, cujo valor, por convenção, queremos minimizar para atingirmos o desempenho ótimo.

Um índice que muito freqüentemente desejamos minimizar é o *custo* de fabricação de um equipamento. Dado um certo equipamento que deve ser fabricado e no qual as especificações estão contidas num vetor $\mathbf{x}$, para cada

[3]Evidentemente isso é apenas uma aproximação.

conjunto de diferentes valores que esse vetor assumir, haverá um custo de fabricação diferente envolvido (imagine-se por exemplo a fabricação de um motor: de cada jeito diferente que ele for projetado, ele terá custos de fabricação diferentes). Nesse caso, a *função objetivo* $f(\mathbf{x})$, será uma função que, para cada conjunto de valores que estiver especificado no vetor $\mathbf{x}$, irá fornecer o custo de fabricação do equipamento descrito por esse vetor. Devido a essa interpretação de custo financeiro, muitas vezes a *função objetivo* é chamada, dentro de livros de otimização, de *função custo*.

Outros índices de desempenho de sistemas que muitas vezes queremos minimizar são: consumo de combustível (em automóveis, por exemplo), ruído de funcionamento (em motores), probabilidade de defeitos (em todo tipo de equipamento), etc. Todos eles, claramente, dependem de como o equipamento foi construído, ou seja, são funções do vetor $\mathbf{x}$.

Muitas vezes, entretanto, desejamos *maximizar* e não *minimizar* algum índice de desempenho de um sistema. Queremos, por exemplo, maximizar a expectativa de lucro em um *portfolio* de investimentos, assim como o tempo de vida útil de um equipamento, ou a capacidade de produção de uma fábrica. Para simplificar a tarefa de elaborar a teoria matemática da Otimização, *iremos manter a convenção de sempre formular um problema de otimização como um problema de minimização*. Nos casos em que deseja-se fazer uma maximização, devido ao significado do índice de desempenho escolhido, basta minimizarmos a função que se deseja maximizar multiplicada por -1. Ou seja, se se deseja maximizar a função $p(\mathbf{x})$, basta fazer $f(\mathbf{x}) = -p(\mathbf{x})$, de forma que ao determinarmos o vetor $\mathbf{x}$ que minimiza $f(\cdot)$, este será também, por conseqüência, o vetor que maximiza $p(\cdot)$.

Em linguagem matemática, dizemos que $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$. Isso significa que f é uma função de um vetor de n variáveis reais (pertencente ao *espaço* $\mathbb{R}^n$), e a própria função f retorna um valor que é real. As diferentes características que essa função pode ter, assim como as conseqüências disso para a elaboração de estratégias de otimização são os temas das próximas seções deste capítulo.

A solução ótima

Da maneira como delimitamos nosso problema, supondo que nosso vetor de variáveis de otimização $\mathbf{x}$ seja composto de variáveis reais, existem infinitas maneiras diferentes de especificar o sistema a ser projetado. Diante disso, qual é a melhor especificação possível, $\mathbf{x}^*$, que o sistema pode assumir, ou

seja, qual é a especificação que faz com que ele funcione da melhor forma possível? A resposta a tal pergunta é exatamente aquilo que a *Otimização* procura encontrar, por meio de suas técnicas. Em palavras:

> *O vetor ótimo* $\mathbf{x}^*$ *é igual ao* **argumento** *da função* $f(\cdot)$ *que faz com que essa função atinja seu* **mínimo** *valor.*

Essa é a forma como deve ser lida a primeira linha da expressão (5.1). Posto isso, como encontrar esse vetor $\mathbf{x}^*$? Esse é o assunto deste livro.

As restrições

Para terminarmos de entender a formulação contida na expressão (5.1), ainda falta entendermos o significado da igualdade e da desigualdade a que está *sujeito* o resultado da otimização. Essas são as chamadas *restrições* do problema. Elas significam o conjunto dos requisitos que o resultado do projeto deve atender para ser admissível enquanto solução.

Alguns tipos de restrição têm significado bastante óbvio, por exemplo uma restrição de que o diâmetro de uma engrenagem deva ser positivo. Embora, se substituído na expressão da função objetivo, um valor de diâmetro negativo talvez possa levar a um "melhor valor" para essa função, não é possível no mundo real construir peças com diâmetro negativo.

Outros tipos de restrição, embora não estejam relacionados com a impossibilidade de implementarmos a solução encontrada, igualmente dizem que tal solução não é admissível, se violar a restrição. Um exemplo disso encontra-se no projeto de automóveis: se queremos projetar o veículo de mínimo custo, não podemos entretanto construir um que cause emissão de gases poluentes acima dos limites estabelecidos em lei. Todos os veículos que emitirem poluentes acima de tais limites não serão considerados soluções admissíveis, por mais barata que seja sua construção. O problema de otimização, colocado dessa forma, passa a ser o de encontrar o projeto do veículo mais barato possível *dentre todos os que atenderem à restrição da emissão de poluentes ser menor ou igual ao limite admissível.*

Os dois exemplos anteriormente citados se enquadram na situação da *restrição de desigualdade*, isto é, são representáveis pela expressão:

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{0} \tag{5.3}$$

Em relação à convenção de que as funções de restrição devam ser menores ou iguais a zero, cabem comentários similares àqueles apresentados a respeito

da convenção de estarmos minimizando, sempre, a função objetivo. Para as restrições de desigualdade, caso ocorram situações em que se deseja garantir que certa função seja maior que ou igual a zero, basta garantir que essa função multiplicada por -1 seja menor que ou igual a zero. Caso seja necessário ainda que certa função seja menor ou igual a um número diferente de zero, basta fazer com que essa função menos esse número seja menor que ou igual a zero. Dessa forma, ao construirmos as técnicas de otimização, levaremos sempre em consideração o formato convencionado da desigualdade, assim simplificando a teoria.

Deve-se observar que agora a função $\mathbf{g}(\cdot)$ é, ela própria, vetorial, retornando múltiplos valores, o que quer dizer que na realidade essa expressão sintética, vetorial, contém um conjunto de expressões escalares, cada uma das quais representa uma restrição diferente. Matematicamente, dizemos que $\mathbf{g}(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^m$, o que significa que para cada vetor de variáveis de otimização $\mathbf{x} \in \mathbb{R}^n$ que for utilizado como argumento da função $\mathbf{g}(\cdot)$, esta retorna um conjunto de m valores reais como resultado, ou seja, a expressão (5.3) é o mesmo que:

$$\begin{aligned} g_1(\mathbf{x}) &\leq 0 \\ g_2(\mathbf{x}) &\leq 0 \\ &\vdots \\ g_m(\mathbf{x}) &\leq 0 \end{aligned} \tag{5.4}$$

sendo cada uma das m funções $g_i(\cdot)$ uma função escalar, que retorna um único valor real. Em problemas práticos, usualmente será necessário lidar com diversas restrições simultaneamente. No exemplo do projeto do automóvel, além de atender ao limite legal de emissão de poluentes, provavelmente será necessária também a preocupação com o consumo de combustível (que não pode ultrapassar um máximo aceitável), com a potência do motor (que não deve ser menor que um mínimo aceitável), etc. O veículo a ser projetado não pode violar nenhuma dessas restrições para ser considerado uma solução aceitável.

Resta ainda falar das *restrições de igualdade*, descritas pela expressão:

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \tag{5.5}$$

Esse tipo de restrição ocorre quando é necessário que certas variáveis as-

sumam precisamente certos valores. Por exemplo, se estamos projetando uma peça que deve se encaixar precisamente num certo espaço disponível num equipamento, do qual a peça faz parte, queremos que a peça tenha exatamente o tamanho especificado, nem mais nem menos. A peça pode até ser constituída de diversos sub-componentes, cujos tamanhos poderemos escolher, desde que a soma de todos os tamanhos tenha o tamanho total especificado. Também essa expressão é vetorial: $\mathbf{h}(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^p$, ou seja, a função vetorial representa na realidade p diferentes equações.

Para concluir este tópico, definimos a seguinte nomenclatura, relacionada com as restrições:

Região factível: Conjunto dos pontos do espaço $\mathbb{R}^n$ que satisfazem, simultaneamente, a todas as restrições (tanto de desigualdade quanto de igualdade). Às vezes a *região factível* é chamada de *conjunto factível*, ou de *conjunto viável*.

Região infactível: Conjunto dos pontos do espaço $\mathbb{R}^n$ que deixam de satisfazer (ou seja, *violam*) pelo menos uma das restrições do problema.

Ponto factível: Ponto pertencente à *região factível*.

Ponto infactível: Ponto pertencente à *região infactível*.

Restrição violada: Cada uma das componentes do vetor $\mathbf{g}(\mathbf{x})$ que apresentar valor positivo, ou cada uma das componentes do vetor $\mathbf{h}(\mathbf{x})$ que apresentar valor não-nulo será chamada de *restrição violada* no ponto $\mathbf{x}$.

5.1.2 As Regras do Jogo

O problema da *Otimização* fica em parte definido pela expressão (5.1). Para delinear o que vem a ser o campo de conhecimento da *Otimização Não-Linear*, enunciamos agora um conjunto de *regras* que dizem *como* é abordado esse problema: qual é a informação de que podemos fazer uso durante o processo de otimização, e qual é o custo dessa informação. Iremos supor, ao longo deste livro, que:

Regras de Acesso à Informação

- Não conhecemos expressões matemáticas explícitas que representem a função objetivo $f(\cdot)$ e as funções de restrição $\mathbf{g}(\cdot)$ e $\mathbf{h}(\cdot)$.
- Temos entretanto a possibilidade de descobrir quanto valem as funções objetivo e de restrição em qualquer ponto do espaço de variáveis de otimização. Essa é a única informação que conseguiremos adquirir, ao longo do processo de otimização, para nos guiar em direção à solução desejada.

O leitor já deve estar protestando neste ponto: por quê introduzimos essa premissa aparentemente arbitrária? O que impede que tenhamos em mãos um modelo matemático de um sistema qualquer, formulado em termos de expressões matemáticas explícitas, que seriam nossas funções objetivo e de restrições? Bem, nada impede isso, pelo contrário, muitas vezes é isso que ocorre. Entretanto, nessas situações, quando temos expressões explícitas simples representando o sistema, podemos fazer (e usualmente fazemos) uso de técnicas da chamada *Análise Matemática* para determinar o mínimo da função objetivo, empregando ferramentas que não estão no escopo daquilo que usualmente chamamos *Otimização*. Um procedimento simples que freqüentemente empregamos nesses casos, por exemplo, é o de derivar a função objetivo, e determinar os pontos em que o gradiente se anula. Quando é possível fazer isso, os pontos de mínimo da função são determinados de maneira direta e exata.

Há entretanto situações em que a utilização desse tipo de procedimento é impossível. Suponhamos por exemplo o caso do projeto da estrutura da asa de um avião. Não é possível descrever o comportamento dinâmico dessa asa por meio de expressões simples, envolvendo por exemplo funções trigonométricas ou polinomiais. Tal objeto tem de ser modelado em termos de um sistema de equações diferenciais parciais, cuja solução é provavelmente muito difícil, ou mesmo impossível, de ser determinada analiticamente. Nesse caso, é necessário escrever um algoritmo que realize o cálculo numérico da solução desse sistema de equações. Cada vez que fazemos a avaliação da função objetivo para um determinado vetor de variáveis de otimização (que significa uma determinada especificação para a asa do avião), temos de executar o algoritmo e, com base no resultado do mesmo, fazer o cálculo da função. O mesmo se aplica às funções de restrição. Ora, uma função que

inclui um algoritmo não pode ser, em geral, explicitamente representada por uma expressão matemática simples, nem pode ser por exemplo derivada ou integrada de maneira explícita. A natureza da função objetivo agora deixa de ser a de uma expressão conhecida, que podemos manipular utilizando todas as manipulações matemáticas usuais. Agora, a metáfora mais adequada para compreendermos sua natureza é a de uma *caixa preta*[4], na qual podemos entrar com um vetor $\mathbf{x}$, obtendo como resposta o valor de $f(\mathbf{x})$ associado a esse vetor[5]. Essa é a única informação disponível para ser utilizada pelos métodos de *Otimização*.

Assim, as regras acima enunciadas simplesmente significam que a teoria da *Otimização* é desenvolvida para o contexto dos problemas em que não temos acesso a uma expressão explícita da função objetivo e das funções de restrição. Obviamente, nos casos de problemas em que conhecemos expressões explícitas de todas as funções, as técnicas da *Otimização* continuam sendo aplicáveis, com a ressalva de que possivelmente haveria maneiras mais simples ou mais precisas para a determinação das soluções[6].

Por fim, há ainda a questão de quão difícil, ou quão demorada, é a obtenção da informação dos valores da função objetivo e das funções de restrição: muitas vezes, para calcularmos o valor da função objetivo em um único ponto (ou seja, para um único vetor $\mathbf{x}$) um bom computador de última geração pode demorar horas ou dias. Esse é o caso, por exemplo, de um modelo detalhado da estrutura da asa de um avião; a engenharia, a economia, as ciências naturais, estão repletas de situações assim. Dessa forma, não seria prático prescrever métodos de otimização que dependessem de calcular essa função objetivo alguns milhares ou centenas de milhares de vezes: talvez não seja viável avaliar essas funções mais que algumas dezenas ou centenas de vezes. Uma outra regra então se justifica:

[4]O conceito de *caixa preta*, nas ciências, diz respeito a objetos cujas entradas e saídas podem ser observadas, mas cujo interior é inacessível.

[5]O leitor deve notar que, embora não saibamos qual é a expressão analítica de uma função que corresponde à caixa preta, tal função *existe*. Se o leitor se lembrar de como a Matemática define funções, verá que essa caixa preta atende a todos os requisitos para ser uma função.

[6]Se houver, entretanto, um número muito grande de restrições ou variáveis no problema, é possível que as técnicas de Otimização ainda sejam as mais adequadas para a determinação do ponto de ótimo, mesmo havendo expressões analíticas para as funções objetivo e de restrições.

Regra de Custo da Informação

- Os métodos de otimização serão comparados entre si de acordo com os critérios:
  - Número de avaliações da função objetivo e das funções de restrição que são requeridas para a determinação da solução. Quanto menos avaliações forem necessárias, melhor será considerado o método.
  - Precisão e robustez. Quanto mais a solução fornecida pelo método se aproximar da solução exata do problema, melhor será considerado o método[7].

Agora sabemos o que estaremos fazendo ao longo deste livro: iremos construir *algoritmos*, que serão as implementações práticas dos *métodos de otimização*, cujo objetivo é determinar as soluções do problema (5.1). Esses algoritmos irão chamar sub-rotinas que executam a avaliação das funções objetivo e de restrições, devendo entretanto fazer a chamada dessas sub-rotinas o menor número de vezes que for possível. O diagrama da figura 5.2 ilustra essa idéia.

5.2 Otimização Sem Restrições

Para começar a estudar a interpretação geométrica dos problemas de otimização, começaremos analisando a situação mais simples, do problema de minimização de uma função objetivo sem nenhuma restrição:

$$\mathbf{x}^* = \arg\ \min\ f(\mathbf{x}) \tag{5.6}$$

Para viabilizar a representação gráfica do problema, estaremos supondo a partir deste ponto que o vetor $\mathbf{x}$ possui apenas duas coordenadas, pertencendo, portanto, ao espaço $\mathbb{R}^2$. Evidentemente, na maioria das situações de

[7] O termo *precisão* designa a capacidade de um algoritmo de, estando próximo da solução exata do problema, aproximar ainda mais tal solução exata. O termo *robustez* por sua vez designa a capacidade do algoritmo de, estando distante da solução exata do problema, atingir as proximidades dessa solução. Assim, freqüentemente um algoritmo é mais preciso e ao mesmo tempo menos robusto que outro, e vice-versa.

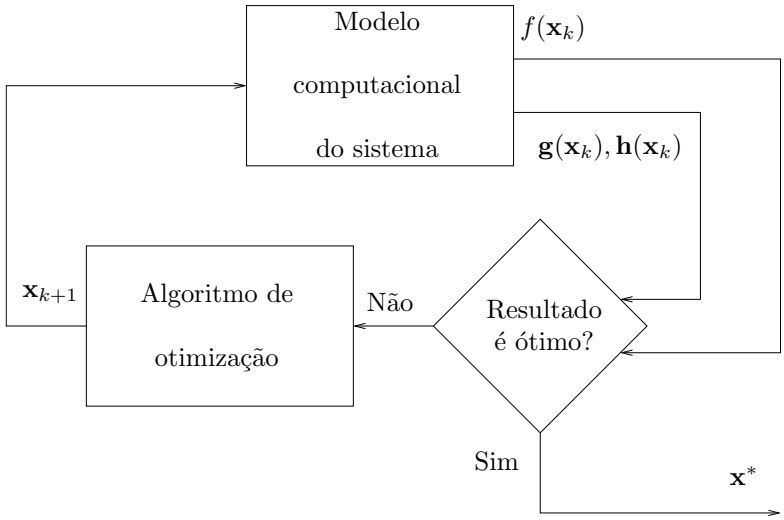


Figura 5.2: Diagrama do processo de otimização. A rotina de otimização fornece o vetor de variáveis de otimização, $\mathbf{x}$, para as rotinas que avaliam a função objetivo e de restrições. Essas rotinas devolvem os valores de $f(\mathbf{x})$, $\mathbf{g}(\mathbf{x})$ e $\mathbf{h}(\mathbf{x})$ para a rotina de otimização. A rotina de otimização, com essas avaliações, calcula um novo vetor de variáveis de otimização a ser avaliado, e assim por diante, até que seja encontrada uma aproximação da solução ótima $\mathbf{x}^*$.

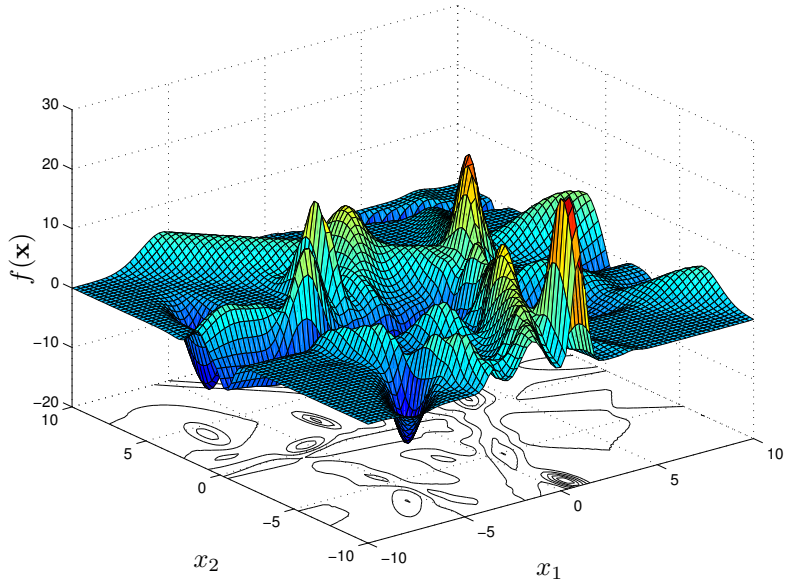


Figura 5.3: Superfície que representa o gráfico de uma função não-linear de duas variáveis reais. Essa superfície poderia representar uma função $f(\mathbf{x})$ cujo mínimo devesse ser determinado por um método de otimização. No "chão" do gráfico, encontram-se representadas as *curvas de nível* da função.

interesse prático o número de coordenadas desse vetor é maior que dois; entretanto, duas variáveis são suficientes para discutirmos a maior parte das questões conceituais que se encontram por detrás da concepção dos métodos de otimização.

Embora estejamos supondo que a função objetivo $f(\cdot)$ não seja conhecida num contexto prático de otimização, essa função é sempre um objeto matemático muito bem definido. Assim, mesmo não sendo possível traçar explicitamente o gráfico de nossa função objetivo (sabemos que isso é impossível devido às *regras da otimização*, anteriormente estabelecidas), podemos afirmar que a superfície correspondente à função existe, e é desta superfície que estaremos colhendo *amostras* durante o processo de otimização, a cada vez que estivermos avaliando a função objetivo. A figura 5.3 mostra uma superfície que corresponde ao gráfico de uma função não-linear de duas variáveis reais. Tal função poderia ser a função objetivo de um problema de otimização.

Uma representação que contém aproximadamente a mesma informação

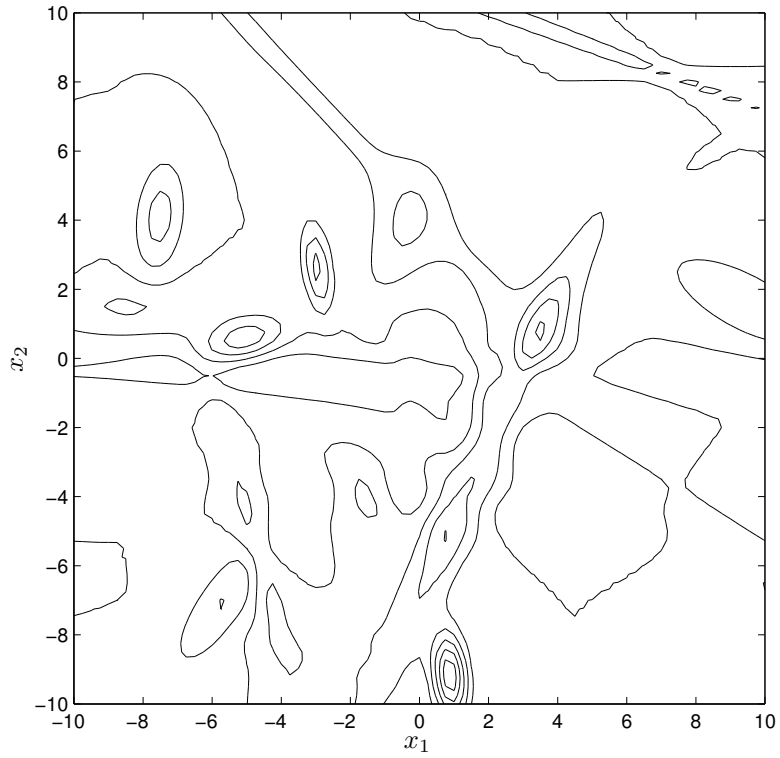


Figura 5.4: Gráfico de *curvas de nível* da mesma função não-linear de duas variáveis reais, $f(\mathbf{x})$, que encontra-se representada na figura 5.3.

que a da figura tridimensional 5.3, mas que utiliza apenas recursos gráficos bidimensionais é a das *curvas de nível* da função. A figura 5.4 mostra as curvas de nível da mesma função representada na figura 5.3. Essa representação, mais fácil de ser manipulada que a representação tridimensional, é normalmente mais útil que esta para ilustrar conceitos relacionados aos métodos de otimização.

Uma metáfora que pode ajudar a compreender o que é o processo de otimização pode ser apresentada da seguinte forma: imaginemos (aqui a imaginação é o mais importante) um ser matemático, o *Otimizador*. Ele vai ser lançado (de pára-quedas) em um ponto qualquer sobre a superfície da figura 5.3, e deverá caminhar sobre essa superfície, em busca do ponto mais baixo da mesma (o *ponto de mínimo*). O Otimizador, entretanto, deverá caminhar com uma venda cobrindo seus olhos, sem poder "olhar" para a superfície; a única informação que ele pode utilizar a respeito da superfície é a altura do ponto no qual ele estiver "pisando". Ele pode, entretanto, se "lembrar" das alturas dos pontos em que ele já tiver pisado anteriormente, fazendo uso dessa informação já adquirida para tomar a decisão de "para onde caminhar". Seu objetivo, além de chegar no ponto de mínima altura sobre a superfície, é fazer isso tendo utilizado o menor número possível de "passos". Essa situação imaginária ilustra bem o que é o problema de otimização. Construir os chamados *métodos de otimização* corresponde, dentro de nossa metáfora, a formular as estratégias a serem utilizadas pelo Otimizador em sua busca pelo ponto de mínimo.

Algumas características da função objetivo (ou seja, da superfície que está associada a essa função) definem que tipos de estratégias seriam efetivas para a otimização dessa função. Por exemplo, a função ser *diferenciável* implica na possibilidade de se tentar sua otimização fazendo uso do cálculo, pelo menos aproximado, de seu gradiente (que pode ser estimado numericamente a partir de amostras de valores da função). Se a função for *unimodal*, ou seja, se tiver um único ponto de mínimo, as estratégias para a determinação desse mínimo serão bem diferentes daquelas que seriam empregadas caso a função fosse *multimodal*, ou seja, caso tivesse vários mínimos locais[8].

Com o objetivo de subsidiar a escolha de métodos adequados para a otimização de funções, podemos definir a seguinte classificação das funções:

[8]Falamos de *mínimos locais* para designar pontos que são de mínimo para uma vizinhança ao seu redor, e de *mínimos globais* para designar o ponto em que a função objetivo atinge seu mínimo valor em todo o domínio considerado.

Modalidade: Unimodal / Multimodal

Diferenciabilidade: Diferenciável / Não-diferenciável

Convexidade: Convexa / *Quasi*-convexa / Não-convexa

Linearidade: Linear / Não-linear

Escala: Uni-escala / Multi-escala

Passamos a mostrar agora algumas superfícies "típicas", que exibem de maneira clara essas propriedades que "fazem a diferença" (o significado dessa classificação deve ficar claro à medida em que essa discussão for apresentada). Com esses exemplos de superfícies, discutiremos de maneira qualitativa possíveis estratégias para a otimização de funções com tais características. Essas estratégias serão depois desdobradas, nos capítulos posteriores, que serão dedicados a discutir em detalhe os *métodos de otimização* correspondentes a essas estratégias.

5.2.1 Estratégias de Direção de Busca

Vamos considerar em primeiro lugar a função cujo gráfico é mostrado na figura 5.5, e cujas curvas de nível estão representadas na figura 5.6.

Para construir essa função, nós utilizamos um esquema bastante simples: o de uma *função quadrática*. A "receita" para a montagem do gráfico da figura 5.5 é dada por:

$$\begin{aligned} f(\mathbf{x}) &= (\mathbf{x}-\mathbf{x}_0)'Q(\mathbf{x}-\mathbf{x}_0) \\ Q &= \begin{bmatrix} 2 & 0.3 \\ 0.3 & 1 \end{bmatrix} \qquad \mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned} \tag{5.7}$$

Claramente, o gráfico dessa função deve ser um parabolóide com mínimo no ponto $\mathbf{x}_0$. O Otimizador, entretanto, como já concordamos, não sabe disso: ele deve descobrir qual é o ponto de mínimo da função objetivo utilizando apenas "amostras" de valores dessa função. Uma estratégia razoável de procedimento para o Otimizador seria:

Método do Gradiente

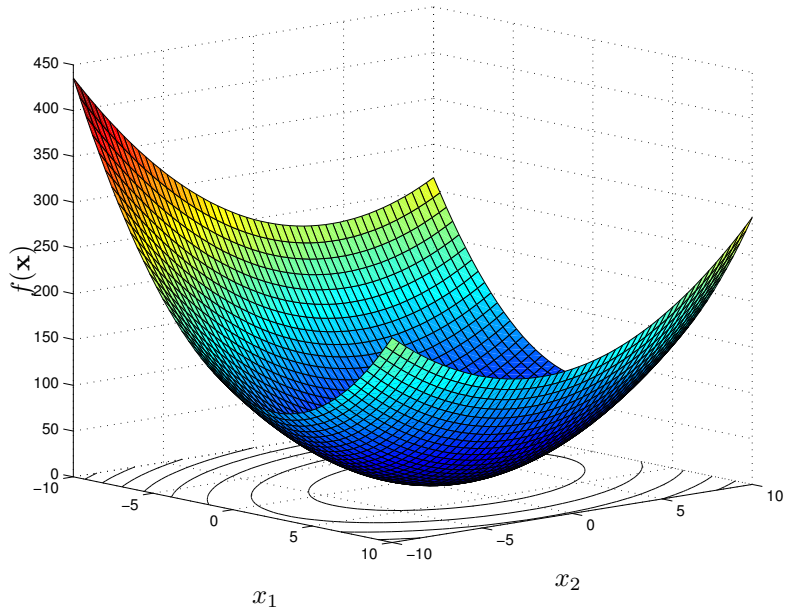


Figura 5.5: Superfície que representa o gráfico de uma função quadrática $f(\mathbf{x})$ de duas variáveis reais. No "chão" do gráfico, encontram-se representadas as *curvas de nível* da função.

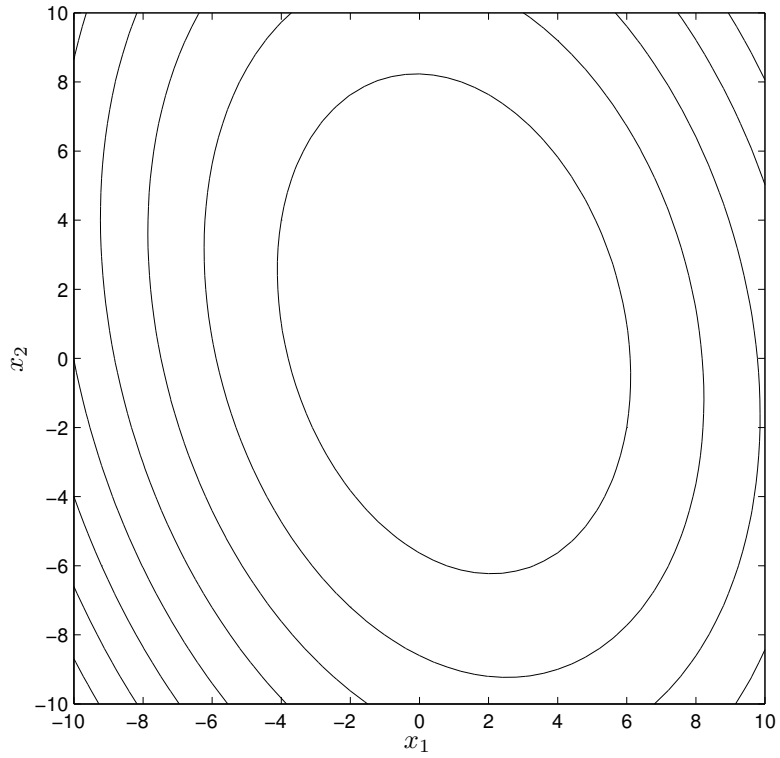


Figura 5.6: Gráfico de *curvas de nível* da mesma função quadrática de duas variáveis reais, $f(\mathbf{x})$, que encontra-se representada na figura 5.5.

Passo 1: O Otimizador, localizado inicialmente em um ponto aleatório sobre o mapa da função, toma amostras da função próximas de onde ele se encontra atualmente. Com essas amostras, ele descobre em qual direção a função decresce mais rapidamente, pelo menos sob o ponto de vista da informação localmente disponível para ele. Em terminologia matemática, o Otimizador calcula uma aproximação numérica do *gradiente* da função no ponto atual (que é o oposto da direção em que a função decresce mais rapidamente).

Passo 2: O Otimizador caminha em linha reta, na direção contrária ao gradiente da função, continuando a andar enquanto estiver sentindo que a função está decrescendo (parando de andar, portanto, assim que percebe que a função volta a crescer nessa direção).

Passo 3: O Otimizador decide agora se ele pára (ou seja, se ele considera que já se encontra suficientemente próximo do ponto de mínimo da função) ou se ele continua a busca, retornando ao Passo 1, para escolher nova direção de caminhada.

O *método do gradiente*, assim esboçado, é um dos métodos de otimização mais primitivos, tendo sido proposto nos primórdios da teoria de otimização, estando hoje obsoleto. Esse método é, entretanto, o protótipo mais simples de toda uma família de métodos, os *métodos de direção de busca*, que incluem importantes métodos hoje utilizados, que sempre têm a estrutura assim descrita:

Métodos de Direção de Busca

Passo 1: O Otimizador, toma amostras da função próximas de onde ele se encontra atualmente. Com essas amostras, ele descobre em qual direção a função decresce mais rapidamente, pelo menos sob o ponto de vista da informação localmente disponível para ele. Em terminologia matemática,

o Otimizador calcula uma aproximação numérica do *gradiente* da função no ponto atual (que é o oposto da direção em que a função decresce mais rapidamente).

Passo 2: Levando em consideração o gradiente calculado no ponto atual, assim como todo o histórico de gradientes anteriormente calculados e de valores de função objetivo amostrados em pontos que o Otimizador visitou anteriormente, ele tenta "adivinhar" qual seria a direção mais provável em que o mínimo da função devesse estar.

Passo 3: O Otimizador caminha em linha reta, na direção em que ele supõe que o mínimo esteja, continuando a andar enquanto estiver sentindo que a função está decrescendo (parando de andar, portanto, assim que percebe que a função volta a crescer nessa direção).

Passo 4: O Otimizador decide agora se ele pára (ou seja, se ele considera que já se encontra suficientemente próximo do ponto de mínimo da função) ou se ele continua a busca, retornando ao Passo 1, para escolher nova direção de caminhada.

Qualquer estratégia de "direção de busca" irá funcionar para determinar o mínimo da função mostrada na figura 5.5, pois esta função é bastante simples. Para esses métodos funcionarem, os requisitos que encontram-se implícitos sobre a função são:

- A função é *unimodal*, ou seja, tem um único mínimo global, no interior de uma única bacia de atração[9]. Dessa forma, o Otimizador não precisa se preocupar com a possível existência de outros mínimos diferentes daquele que ele localizar.

- A função é *diferenciável*, ou seja, não só é possível calcular, de forma significativa, aproximações do gradiente da função em qualquer ponto do

[9]Uma *bacia de atração* é a região ao redor de um mínimo local na qual as curvas de nível da função são fechadas, ou seja, a região na qual um método de direção de busca irá convergir para tal mínimo.

espaço, como, principalmente, o gradiente da função contém informação significativa sobre a forma como a função varia nas vizinhanças do ponto em que tiver sido calculado. Dessa forma, o Otimizador consegue encontrar direções para as quais possa caminhar, nas quais ele consegue observar a diminuição do valor da função objetivo.

Consideremos agora a função mostrada na figura 5.7, que tem suas curvas de nível mostradas na figura 5.8. Essa função, muito menos simples que a função quadrática anteriormente considerada, continua sendo adequadamente otimizada por métodos de direção de busca: ela é unimodal (possui um único mínimo, no ponto $\mathbf{x} = \begin{bmatrix} 1 & 1 \end{bmatrix}'$, o interior de uma única bacia de atração), e é diferenciável (possui gradiente bem definido em todos os pontos).

Essa função já é capaz de "confundir" um Otimizador que utilizar simplesmente uma estratégia de de gradiente: quando o Otimizador chega no fundo do "vale" existente na topografia da função, e tem de encontrar o ponto mais baixo desse vale, o padrão de mudança da direção do gradiente torna o método do gradiente muito ineficiente. Outros métodos de direção de busca, no entanto, não encontram dificuldades para minimizar esta função.

5.2.2 Estratégias de Exclusão de Regiões

Consideremos agora a função $f(\mathbf{x})$, ainda unimodal, porém agora não mais diferenciável, cujo gráfico está mostrado na figura 5.9, e cujas curvas de nível estão representadas na figura 5.10. Este tipo de função em geral traz dificuldades para as estratégias de otimização do tipo *direções de busca*.

Ao contrário do que pode parecer à primeira vista, a dificuldade não está na impossibilidade de calcularmos o gradiente da função: na imensa maioria das vezes, uma função não diferenciável de interesse prático é *diferenciável em quase todo ponto*. Esse é o caso da função representada na figura 5.9: seu gradiente deixa de existir apenas em alguns poucos pontos, que estão situados em algumas linhas sobre o mapa da função. Em todos os outros pontos, o gradiente é bem definido e pode ser calculado. Assim, se um Otimizador estivesse otimizando uma função não diferenciável e encontrasse um ponto no qual fosse impossível calcular o gradiente, bastaria ele se deslocar um pouco do ponto, para outro ponto próximo: lá o gradiente poderia ser calculado, e o processo de otimização poderia prosseguir.

O problema com as funções não diferenciáveis, quando submetidas a

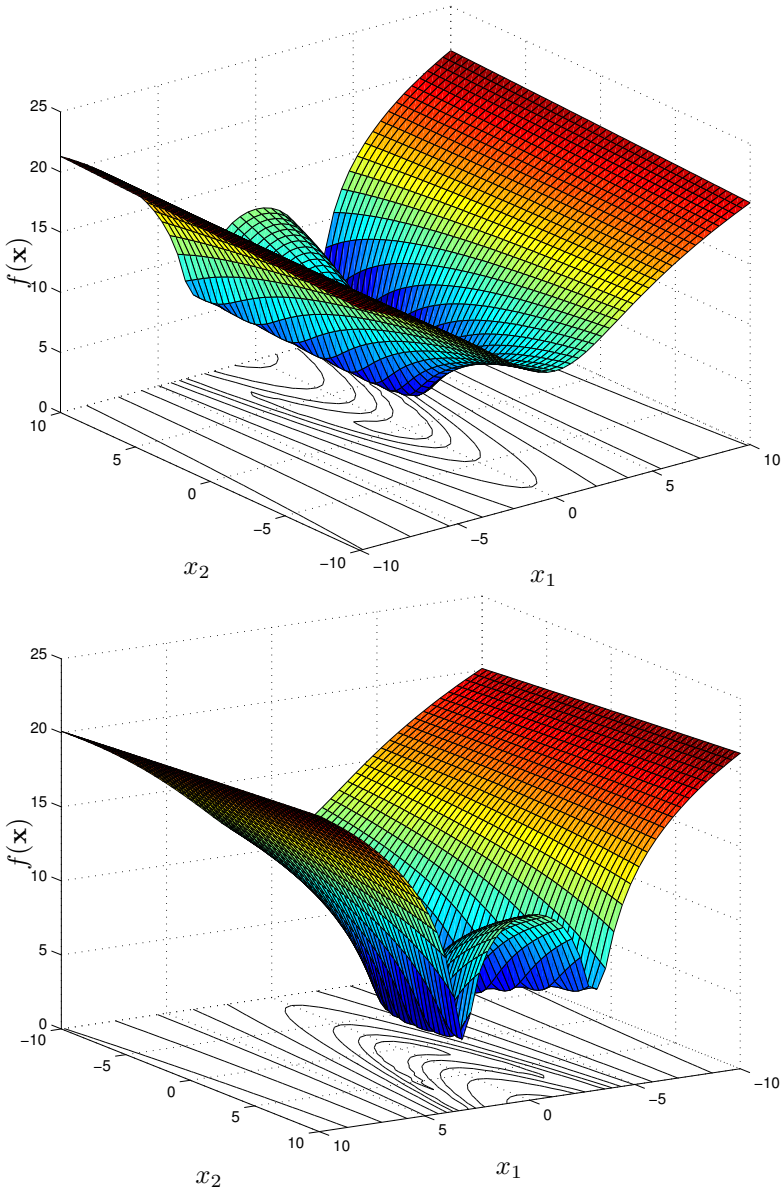


Figura 5.7: Superfície que representa o gráfico de uma função unimodal diferenciável $f(\mathbf{x})$ de duas variáveis reais, mostrada em duas vistas diferentes. No "chão" dos gráficos, encontram-se representadas as *curvas de nível* da função.

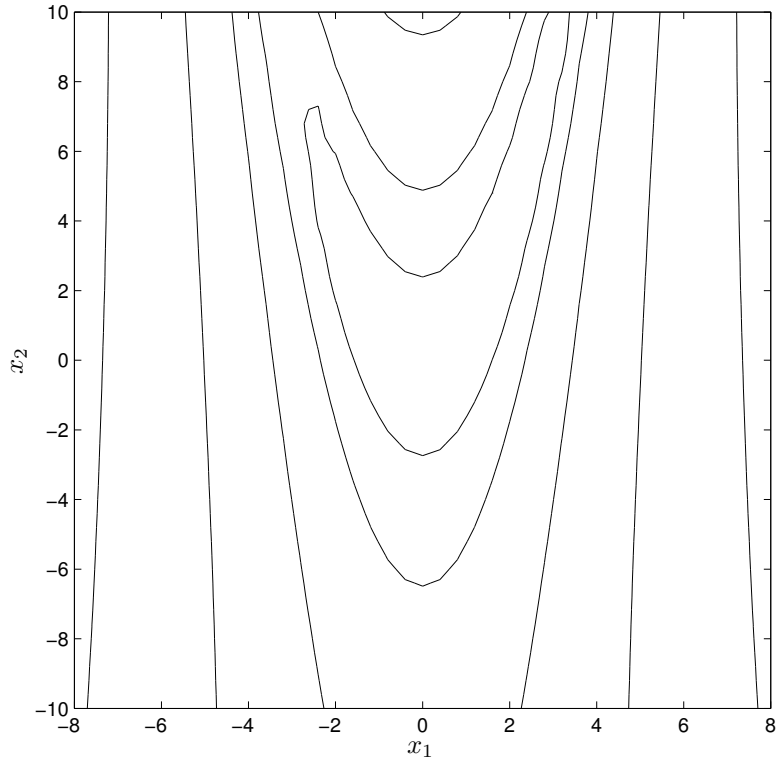


Figura 5.8: Gráfico de *curvas de nível* da mesma função unimodal diferenciável de duas variáveis reais, $f(\mathbf{x})$, que encontra-se representada na figura 5.7.

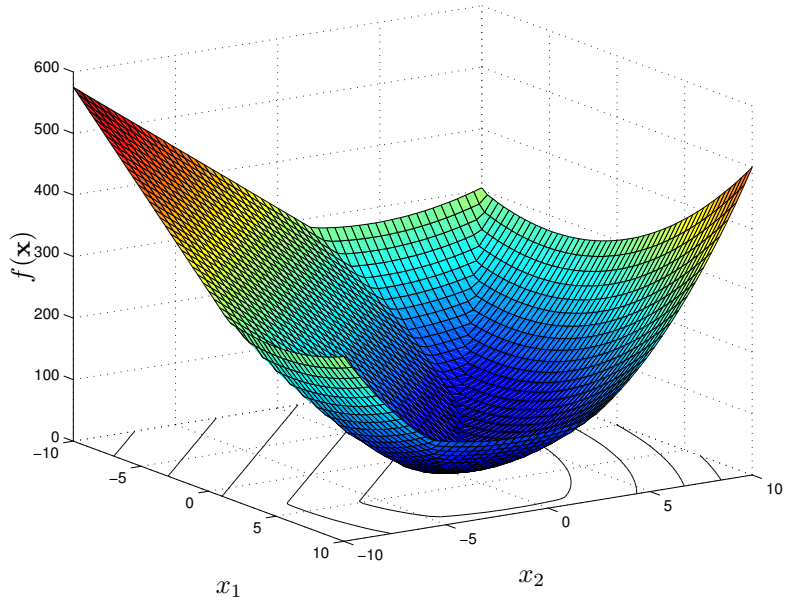


Figura 5.9: Superfície que representa o gráfico de uma função não diferenciável $f(\mathbf{x})$ de duas variáveis. No "chão" do gráfico, encontram-se representadas as *curvas de nível* da função.

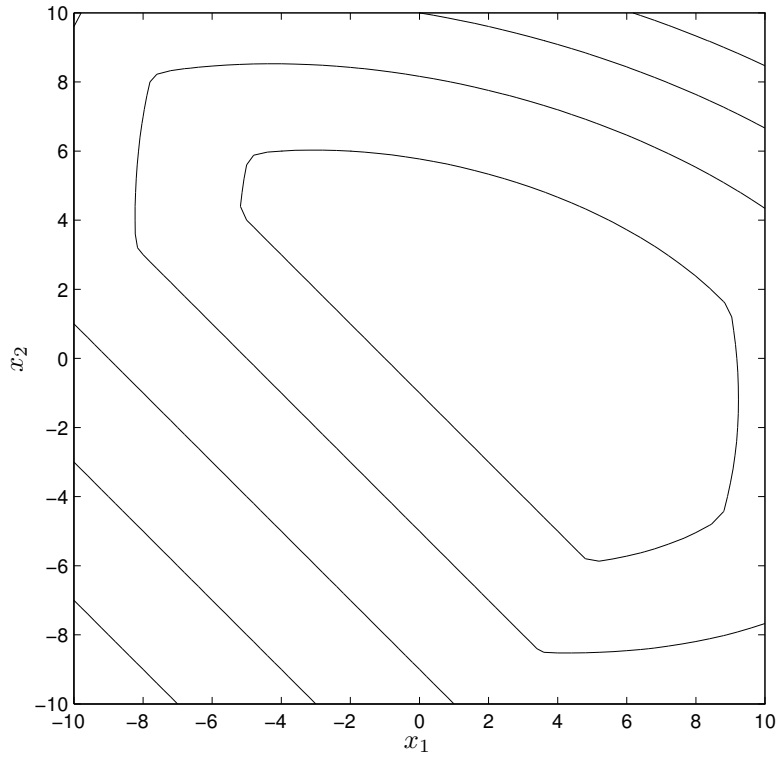


Figura 5.10: Gráfico de *curvas de nível* da mesma função não diferenciável de duas variáveis reais, $f(\mathbf{x})$, que encontra-se representada na figura 5.9.

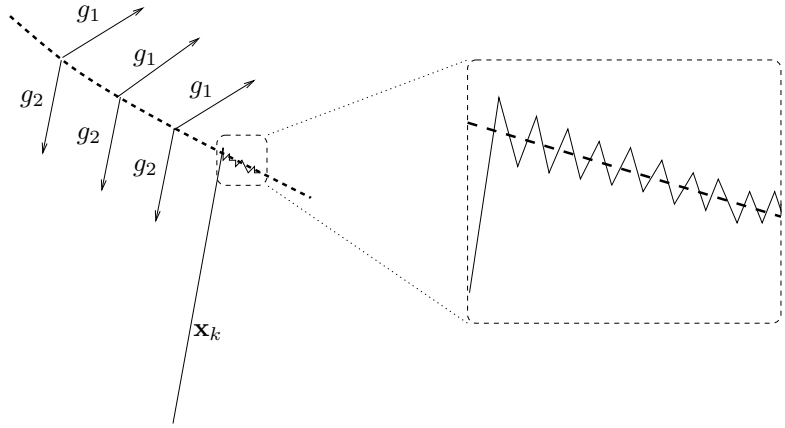


Figura 5.11: Não-diferenciabilidade atratora, representada pela linha tracejada. Acima dessa não-diferenciabilidade, os gradientes da função são representados por $\mathbf{g}_1$, e abaixo por $\mathbf{g}_2$. Exatamente na não-diferenciabilidade, o gradiente da função muda subitamente (ou seja, o gradiente é descontínuo sobre essa linha). A figura mostra ainda a trajetória de um Otimizador que utiliza uma estratégia de direções de busca, percorrendo uma seqüência de pontos $\mathbf{x}_k$. Quando atinge a não-diferenciabilidade atratora, o Otimizador passa a se mover segundo passos muito pequenos. Uma ampliação desse movimento é mostrada na figura à direita.

métodos de direção de busca, é que o cálculo da direção de busca, na qual o Otimizador deve caminhar, é feito a partir da informação obtida pelo cálculo do gradiente (o gradiente atual e o gradiente em pontos anteriores). O Otimizador, ao caminhar nessa direção, espera que a direção tenha validade não apenas pontual: ele espera poder caminhar uma certa distância sobre essa direção, até que a função objetivo pare de decrescer, e ele tenha de mudar de direção. Ora, se a função objetivo muda de comportamento repentinamente nos locais onde a função é não-diferenciável, a informação da direção de busca, obtida com o uso de gradientes pode ser inteiramente inadequada para representar o comportamento da função, mesmo a pequenas distâncias do ponto atual. A otimização por esses métodos pode assim se tornar inviável. Tal dificuldade, por outro lado, não é associada a um ou outro caso específico de método de direção de busca: ela é intrínseca a toda a família dos métodos de direção de busca. A dificuldade é ilustrado na figura 5.11.

Funções não-diferenciáveis estão longe de ser raras, dentro dos modelos

de sistemas que temos interesse em otimizar. Por essa razão, justifica-se a formulação de uma família de métodos diferente, que não esteja sujeita a tal dificuldade: os *métodos de exclusão de regiões*. Para formular a nova estratégia, como estamos abrindo mão da premissa de *diferenciabilidade* da função objetivo, introduzimos em lugar desta a premissa de *convexidade* dessa função[10].

A propriedade associada à convexidade que iremos utilizar na nova estratégia de otimização pode ser entendida da seguinte forma:

- Uma curva de nível de uma função convexa sempre delimita uma região convexa em seu interior.
- O vetor gradiente, por sua vez, é sempre perpendicular à curva de nível que passa pelo ponto onde o vetor foi calculado.
- Assim, a reta perpendicular ao vetor gradiente que passa no ponto onde esse vetor foi calculado é tangente à curva de nível.
- Devido à convexidade da região no interior da curva de nível, esta região sempre fica inteiramente localizada um dos lados apenas dessa reta tangente (essa reta não corta a região no interior da curva de nível): do lado oposto àquele para onde aponta o vetor gradiente.

Isso significa que, se calcularmos o gradiente de uma função convexa num ponto, podemos ter certeza que o ponto de mínimo dessa função, que se localiza necessariamente no interior da curva de nível fechada que passa nesse ponto, está no semi-plano oposto ao do vetor gradiente, delimitado pela reta perpendicular ao vetor gradiente. Esse conceito é ilustrado na figura 5.12.

O procedimento do Otimizador agora é descrito por:

Métodos de Exclusão de Regiões

Passo 1: O Otimizador adquire informação em alguns pontos próximos do atual, e faz uma estimativa do gradiente da função objetivo nesse ponto (se ele estiver exatamente sobre um ponto em que a função é não-diferenciável, admitamos,

[10] É claro que às vezes as funções a serem otimizadas serão convexas e às vezes não serão. Se não forem, os métodos de exclusão de regiões poderão falhar.

para simplificar, que ele se movimenta para algum ponto próximo do atual, em que a função é diferenciável).

Passo 2: Com base no gradiente, o Otimizador descobre qual é a reta tangente à curva de nível que passa pelo ponto atual, e descarta todo o semi-plano que se encontra do lado dessa reta para o qual o vetor gradiente aponta (o Otimizador tem certeza de que o mínimo da função *não está* nesse semi-plano).

Passo 3: O Otimizador se move para algum ponto no interior da região que ainda não está descartada, de preferência para um ponto aproximadamente "no meio" dessa região[11].

Passo 4: O Otimizador decide se existem indícios suficientes de que o novo ponto já esteja suficientemente próximo do mínimo da função, caso em que o processo termina, ou se a otimização deve continuar. Nesse último caso, retorna ao Passo 1.

Deve-se observar que agora a convergência da seqüência de pontos para o ponto de mínimo da função objetivo ocorre em virtude da diminuição sistemática que é feita, a cada iteração do método, da região em que esse ponto de mínimo pode estar localizado. Com o avançar das iterações, a região tem de ficar pequena, e o novo ponto, que é escolhido dentro dessa região, tem de ficar cada vez mais próximo do ponto de mínimo. Não há a possibilidade, agora, de uma não-diferenciabilidade impedir a convergência do método.

Uma seqüência de iterações de um *método de exclusão de região* é ilustrada na figura 5.12.

5.2.3 Estratégias de Populações

Grande parte das funções objetivo que queremos otimizar na prática, infelizmente, não é unimodal. Por conseqüência, tanto as estratégias de direção de busca quanto as estratégias de exclusão de regiões irão falhar em sua otimização[12]. Uma função desse tipo é mostrada na figura 5.13, e suas curvas

[11] A maneira exata de escolher o novo ponto varia de método para método.

[12] Deve-se lembrar que se uma função não é unimodal, ela também não pode ser convexa.

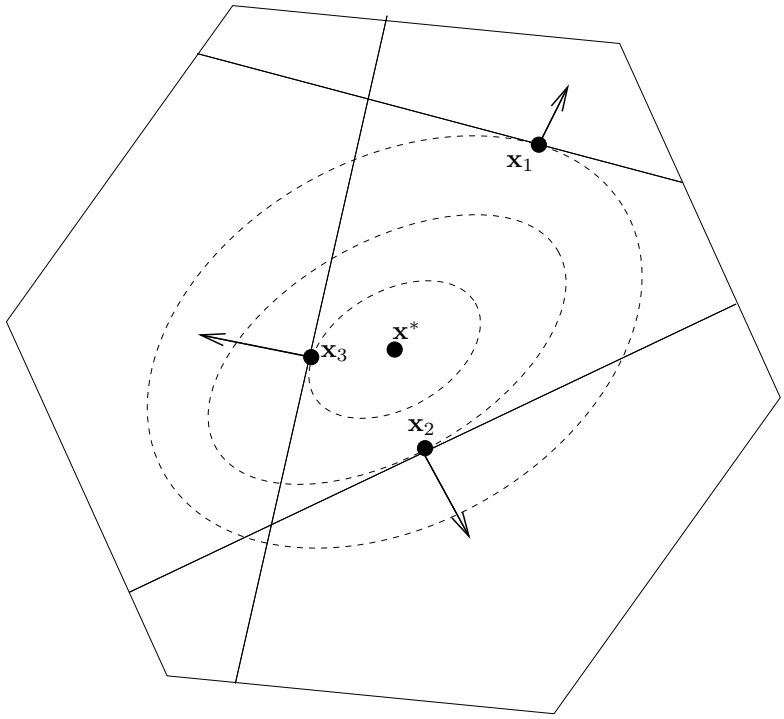


Figura 5.12: Iterações de um método de exclusão de regiões, mostradas sobre as curvas de nível de uma função cujo mínimo exato é $\mathbf{x}^*$. Suponha-se que, *a priori*, se sabe que o mínimo da função se encontra na região delimitada pelo hexágono. Após avaliar o gradiente da função em $\mathbf{x}_1$, o Otimizador pode concluir que o mínimo $\mathbf{x}^*$, cuja localização ainda não é conhecida, encontra-se abaixo da reta perpendicular a esse gradiente, que passa nesse ponto. Um novo ponto $\mathbf{x}_2$ é escolhido no interior da região restante. O gradiente nesse ponto também é calculado, trazendo a informação de que o ponto $\mathbf{x}^*$ não se encontra abaixo da reta perpendicular ao gradiente que passa nesse ponto. A seguir um novo ponto $\mathbf{x}_3$ é escolhido, e o processo se repete, levando à conclusão de que $\mathbf{x}^*$ não se encontra à esquerda da reta que passa por esse ponto. Observa-se que a cada passo vai diminuindo a região onde é possível que $\mathbf{x}$ se encontre. O processo termina quando a região "possível" é suficientemente pequena.

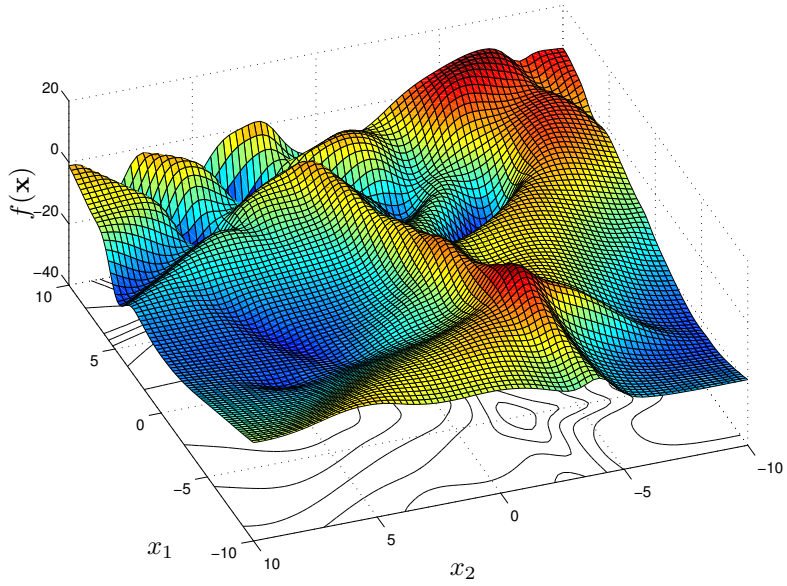


Figura 5.13: Superfície que representa o gráfico de uma função multimodal $f(\mathbf{x})$ de duas variáveis. No "chão" do gráfico, encontram-se representadas as *curvas de nível* da função.

de nível são mostradas na figura 5.14.

De fato, essa função possui diversas bacias de atração diferentes, associadas a diferentes mínimos locais. Na tentativa de se fazer a otimização desta função por meio de um mecanismo de direção de busca, por exemplo, o resultado sempre será o ponto de mínimo local associado à bacia de atração onde a busca tiver sido iniciada. Para se atingir o mínimo global com algum grau de certeza, é necessário "investigar" a função em suas diferentes bacias de atração.

A estratégia a ser adotada envolve agora o trabalho não mais de um único Otimizador sozinho: um grupo de Otimizadores será agora chamado a cooperar, para tentar descobrir a localização do ponto de mínimo da função. Essa estratégia é descrita a seguir:

Métodos de Populações

Passo 1: Um grupo de Otimizadores encontra-se espalhado pela região onde acredita-se que se encontre o ponto de mínimo da

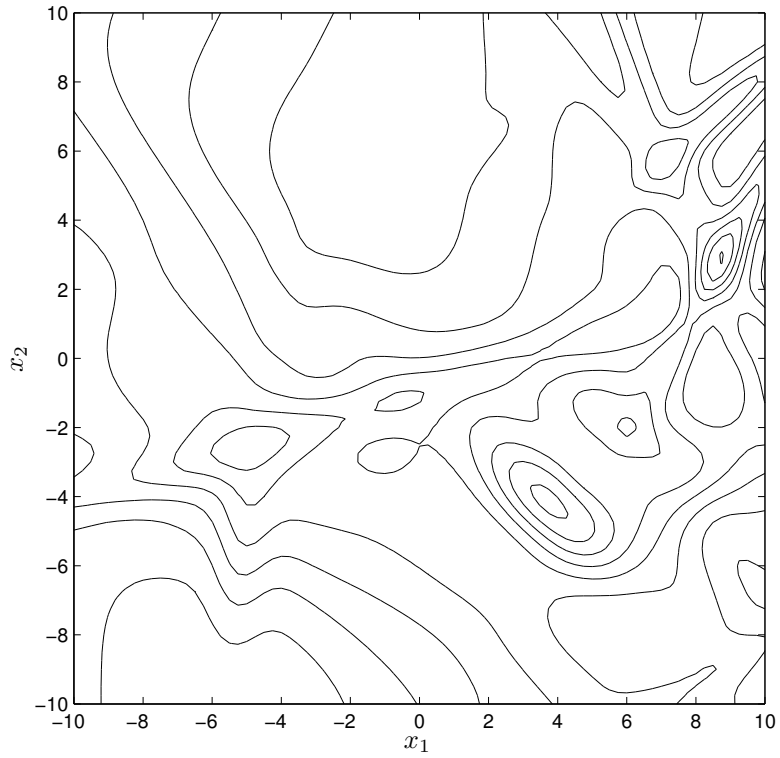


Figura 5.14: Gráfico de *curvas de nível* da mesma função multimodal de duas variáveis reais, $f(\mathbf{x})$, que encontra-se representada na figura 5.13.

função. Cada um dos Otimizadores avalia a função objetivo no ponto onde ele se encontra.

Passo 2: Os Otimizadores se comunicam, e trocam informações a respeito dos valores da função objetivo em cada ponto.

Passo 3: Um pequeno sub-grupo do grupo de Otimizadores, que estiver nas melhores localizações fica parado. Os demais Otimizadores se movimentam, com movimentos que simultaneamente: (i) os façam se aproximarem dos otimizadores melhor localizados; e (ii) os façam explorarem outras regiões, diferentes daquelas já visitadas anteriormente pelo grupo de Otimizadores.

Passo 4: Cada um dos Otimizadores avalia a função objetivo no ponto para onde foi.

Passo 5: Os otimizadores decidem se o processo de otimização já produziu melhoria suficiente na função objetivo, caso em que o processo se interrompe; do contrário, eles retornam ao Passo 2.

Há diferentes maneiras de realizar cada um dos passos do esquema descrito acima. Cada combinação dessas diferentes fórmulas leva a um método específico diferente.

Esse tipo de estratégia pode ser pensado como um jeito de localizar não exatamente o mínimo global da função objetivo, mas sim a bacia de atração no qual este se encontra. Como esse esquema é muito "caro" comparado aos esquemas de direções de busca ou de exclusão de regiões (usualmente os esquemas de "populações" requerem um número muito maior de avaliações da função objetivo até atingirem o ponto de mínimo da função objetivo), a idéia é que o esquema de populações apenas conduza o Otimizador às proximidades do ponto de mínimo global. Uma vez dentro da bacia de atração do mínimo global, o Otimizador passa a adotar uma estratégia por exemplo de direção de busca, que o leva muito mais rapidamente ao mínimo da função. Esse raciocínio funcionaria corretamente, por exemplo, na otimização da função ilustrada na figura 5.13. A figura 5.15 mostra sucessivas aproximações do ponto de mínimo global da função, que terminam por "se parecer" com uma função convexa e unimodal, nas proximidades do ponto

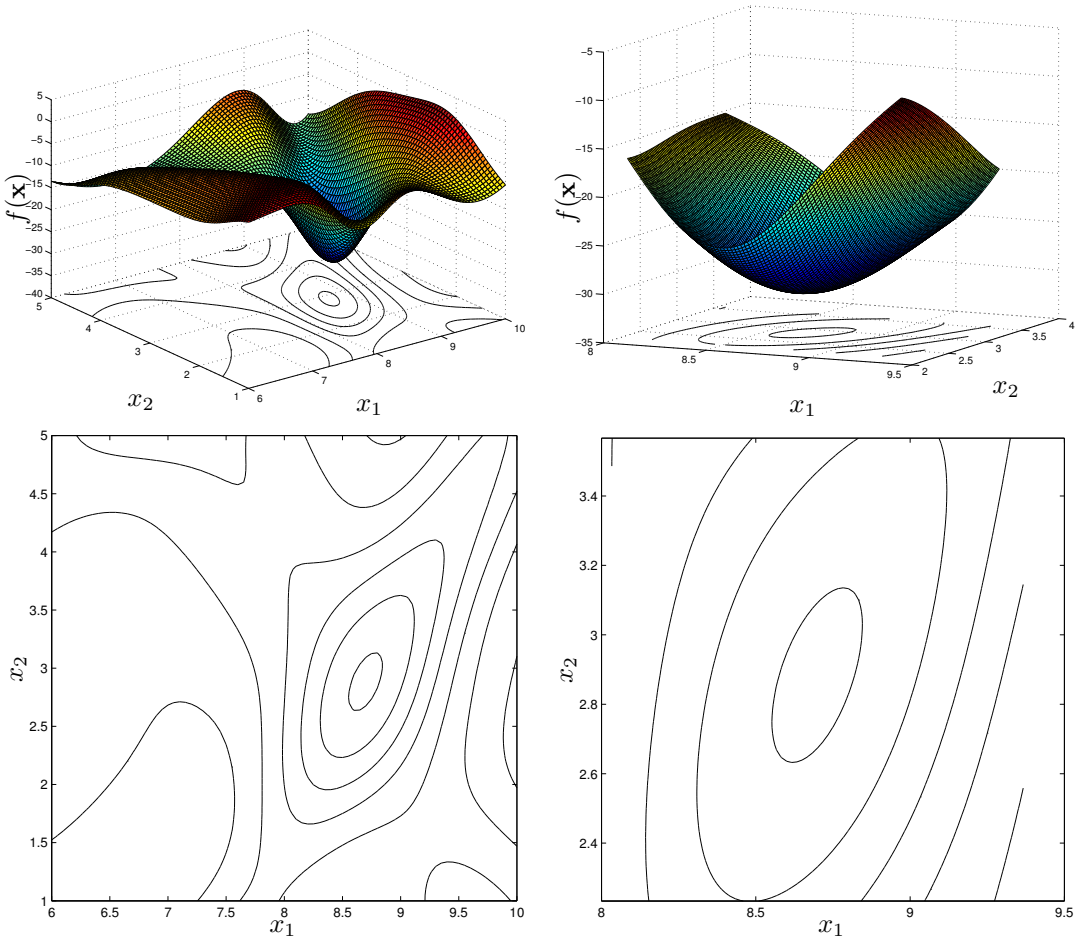


Figura 5.15: Superfície que representa o gráfico da mesma função multimodal $f(\mathbf{x})$ de duas variáveis mostrada na figura 5.13, em sucessivas aproximações da região onde se encontra seu mínimo global. Acima, estão representados os gráficos da superfície, e abaixo as correspondentes curvas de nível na mesma região. Deve-se observar que, na região mais próxima ao mínimo, a função tem a "aparência" de uma função unimodal.

de mínimo. Na região correspondente à última aproximação mostrada na figura, um método de direções de busca ou de exclusão de regiões funcionaria. O método de populações então poderia ser paralisado assim que houvesse indícios suficientes de que determinado ponto se encontra no interior da bacia de atração do mínimo global, sendo iniciado um outro método de otimização nesse ponto.

Essa lógica de mudança de um método de população para outro tipo de método nem sempre funciona. Um exemplo de situação em que tal esquema não funcionaria é a função representada na figura 5.16. Nessa figura, vemos um exemplo de função em que ocorre o fenômeno das *múltiplas escalas*. Essa função, olhada a uma "grande distância", parece ter algumas bacias de atração. Olhada "de perto", ela revela uma estrutura muito mais complexa, com a presença de dezenas pequenas "sub-bacias" onde parecia estar cada uma das bacias de atração inicialmente aparentes. Um método de direção de busca que fosse iniciado no interior dessa "grande bacia" aparente iria quase certamente falhar na busca do mínimo global, ficando provavelmente detido em algum dos múltiplos mínimos locais existentes nessa região. Funções desse tipo vão requerer a utilização de um esquema *de população* para realizar sua otimização, do princípio ao fim, sem a possibilidade de mudança para outro tipo de método.

5.3 Otimização com Restrições de Desigualdade

A próxima situação a ser estudada aqui é aquela em que, na formulação do problema de otimização, aparecem as chamadas *restrições de desigualdade*:

$$\begin{aligned} \mathbf{x}^* = &\ \arg\ \min\ f(\mathbf{x}) \\ &\\ \text{sujeito a:}\ &\ \{\mathbf{g}(\mathbf{x}) \leq \mathbf{0} \end{aligned} \tag{5.8}$$

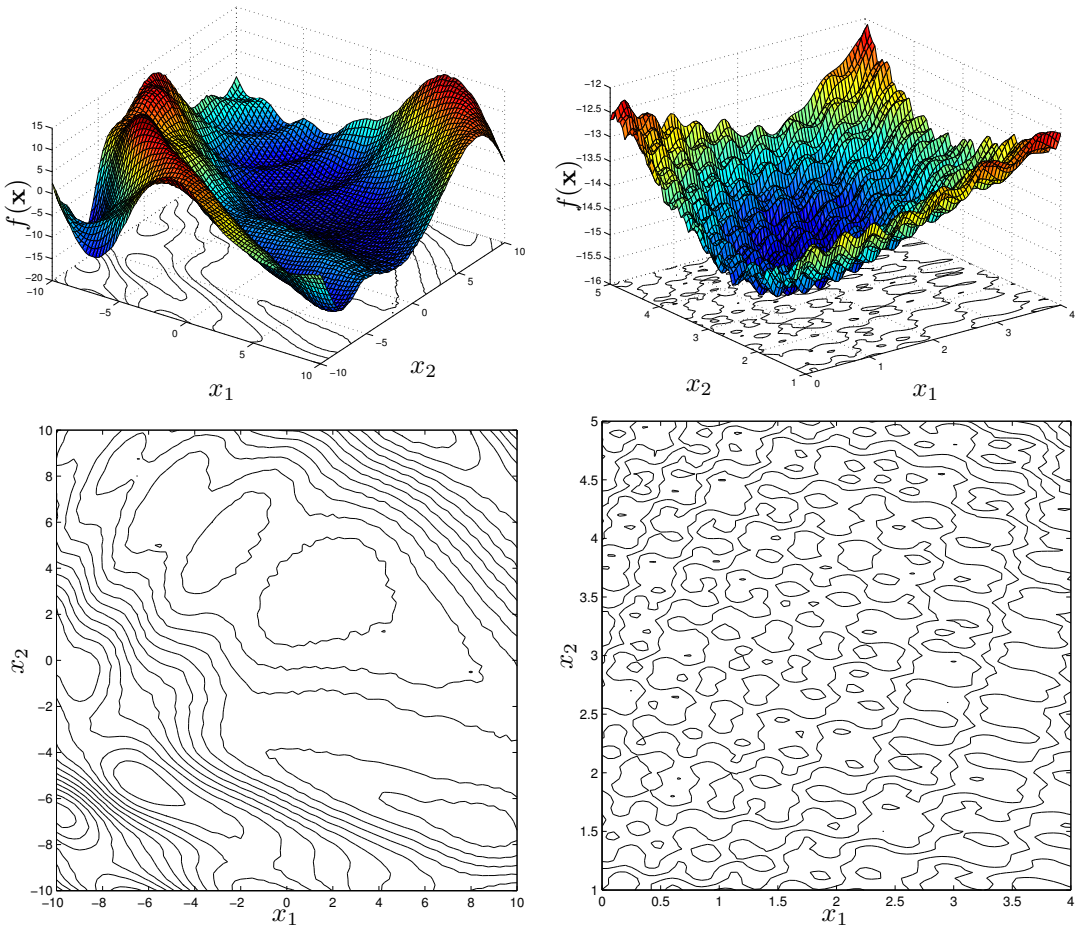


Figura 5.16: Superfície que representa o gráfico de uma função multimodal $f(\mathbf{x})$ de duas variáveis que apresenta a característica de *múltiplas escalas*. Sucessivas aproximações da região onde se encontra seu mínimo global irão revelar sucessivas estruturas de menor escala, que possuem múltiplas bacias de atração dentro de cada bacia de atração maior. Acima, estão representados os gráficos da superfície, e abaixo as correspondentes curvas de nível na mesma região. Deve-se observar que onde, pelo primeiro par de gráficos, esperaríamos encontrar uma única bacia de atração, encontramos, no exame mais detalhado, uma estrutura com múltiplas pequenas "sub-bacias".

Essa descrição do problema significa, conforme já foi visto, que o ponto de ótimo $\mathbf{x}^*$ a ser determinado deve satisfazer às m desigualdades:

$$\begin{aligned} g_1(\mathbf{x}^*) &\leq 0 \\ g_2(\mathbf{x}^*) &\leq 0 \\ &\vdots \\ g_m(\mathbf{x}^*) &\leq 0 \end{aligned} \tag{5.9}$$

5.3.1 Interpretação geométrica de uma restrição de desigualdade

Examinemos primeiro o que significa uma dessas desigualdades apenas, por exemplo a primeira:

$$g_1(\mathbf{x}) \leq 0 \tag{5.10}$$

Admitamos que a função $g_1(\cdot)$ seja contínua. Se isso for verdade, essa função nunca muda "bruscamente" de valor. Por exemplo, para passar de um valor negativo para um valor positivo, necessariamente ela tem de passar pelo valor zero. Isso significa que, se considerarmos todo o espaço $\mathbb{R}^n$ dos $\mathbf{x}$, se houver um subconjunto $\mathcal{P}_1 \subset \mathbb{R}^n$ para cujos pontos $\mathbf{x}$ a função $g_1(\cdot)$ fica positiva, e outro subconjunto $\mathcal{N}_1 \subset \mathbb{R}^n$ para o qual a função $g_1(\cdot)$ fica negativa, então tem de haver um conjunto $\mathcal{G}_1 \subset \mathbb{R}^n$ para o qual a função se anula, e que separa $\mathcal{P}_1$ de $\mathcal{N}_1$.

Matematicamente, definimos o conjunto $\mathcal{P}_1$ da seguinte forma:

$$\mathcal{P}_1 \triangleq \{\mathbf{x} \mid g_1(\mathbf{x}) > 0\} \tag{5.11}$$

Em palavras, essa expressão deve ser lida como: *O conjunto $\mathcal{P}_1$ é definido como ($\triangleq$) o conjunto dos pontos $\mathbf{x}$ tais que ($\mid$) a função $g_1(\cdot)$ avaliada nesses pontos seja maior que zero.* De forma similar, são definidos os conjuntos $\mathcal{G}_1$ e $\mathcal{N}_1$:

$$\begin{aligned} \mathcal{G}_1 &\triangleq \{\mathbf{x} \mid g_1(\mathbf{x}) = 0\} \\ \mathcal{N}_1 &\triangleq \{\mathbf{x} \mid g_1(\mathbf{x}) < 0\} \end{aligned} \tag{5.12}$$

A figura 5.17 ilustra tais conjuntos, para um espaço de duas dimensões.

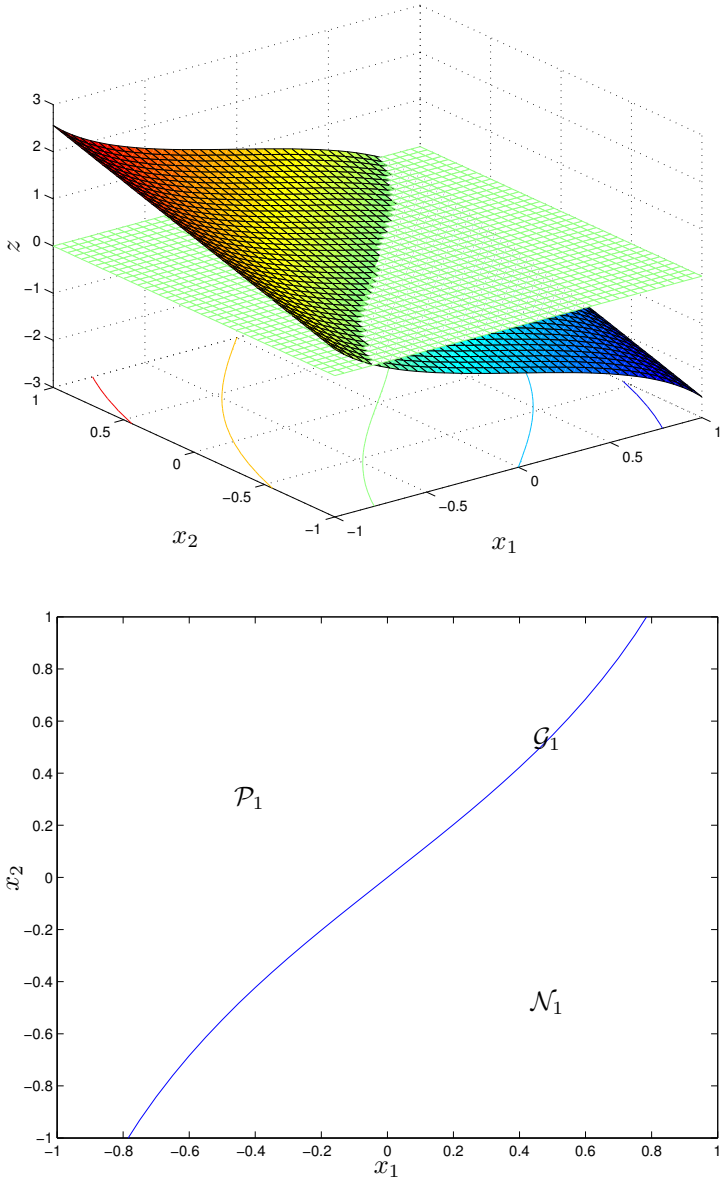


Figura 5.17: Na figura superior, é mostrada a superfície $z = g_1(x)$, com suas curvas de nível, e sua interseção com o plano $z = 0$. Na figura inferior, é mostrado o plano x, sendo mostrada apenas a curva de nível $g_1(x) = 0$. Nesse plano, a região $\mathcal{N}_1$ corresponde aos pontos em que a função $g_1(\cdot)$ é negativa; a região $\mathcal{P}_1$ corresponde aos pontos em que a função $g_1(\cdot)$ é positiva; e a fronteira que separa essas regiões, $\mathcal{G}_1$, corresponde aos pontos em que a função $g_1(\cdot)$ se anula.

Quando inserimos, no problema de otimização, a exigência de que $g_1(\mathbf{x}^*) \leq 0$, queremos dizer que iremos aceitar como soluções do problema de otimização apenas pontos que sejam pertencentes ao conjunto $\mathcal{N}_1$ ou ao conjunto $\mathcal{G}_1$. Não serão admissíveis pontos pertencentes ao conjunto $\mathcal{P}_1$, que será assim denominado *conjunto infactível*, ou *região infactível*. Diz-se então que o *conjunto factível*, ou a *região factível* $\mathcal{F}_1$ é a união de $\mathcal{G}_1$ e $\mathcal{N}_1$:

$$\mathcal{F}_1 = \mathcal{G}_1 \cup \mathcal{N}_1 \tag{5.13}$$

Se aplicarmos agora um dos métodos de otimização já discutidos anteriormente para problemas sem restrições, para a minimização da função $f(\mathbf{x})$, há duas possibilidades para a localização do ponto de mínimo: ele tem de estar em $\mathcal{P}_1$ ou em $\mathcal{F}_1$. Se a última hipótese ocorrer, a solução do problema será o ponto de mínimo encontrado. No entanto, se o mínimo *irrestrito* (ou seja, o mínimo encontrado sem levar em consideração a restrição $g_1(\mathbf{x}^*) \leq 0$) estiver na região *infactível* $\mathcal{P}_1$, alguma modificação deverá ser introduzida no mecanismo de otimização, para que seja possível localizar o ponto de ótimo $\mathbf{x}^*$ que minimiza a função objetivo $f(\cdot)$ nos pontos pertencentes ao conjunto factível $\mathcal{F}_1$.

Esse é, basicamente, o problema da *otimização restrita com restrições de desigualdade*: determinar o ponto $\mathbf{x}^* \in \mathcal{F}$ (ou seja, pertencente à região factível) que minimiza a função $f(\cdot)$ nessa região (ou seja, que produz o menor valor dessa função, quando comparado com os valores da função em todos os demais pontos da região factível).

5.3.2 Interpretação geométrica de várias restrições de desigualdade

Antes de discutirmos como modificar os mecanismos de otimização para lidar com problemas de *otimização restrita*, vamos procurar entender o que significa o sistema de restrições na forma em que o mesmo usualmente aparece: um conjunto de várias desigualdades que devem ser simultaneamente

satisfeitas. Escrevendo novamente o sistema:

$$\begin{aligned} g_1(\mathbf{x}^*) &\leq 0 \\ g_2(\mathbf{x}^*) &\leq 0 \\ &\vdots \\ g_m(\mathbf{x}^*) &\leq 0 \end{aligned} \tag{5.14}$$

A figura 5.18 mostra a situação para duas restrições: a região factível (ou seja, a região dos pontos que simultaneamente atendem às duas restrições) corresponde à interseção da região cujos pontos atendem à primeira restrição com a região cujos pontos atendem à segunda restrição. Em geral, se $\mathcal{F}_i$ designa a região em que a função $g_i(\cdot)$ é menor ou igual a zero[13], temos que a *região factível* $\mathcal{F}$ do problema envolvendo todo o conjunto de restrições (5.14) corresponde à interseção de todas essas regiões:

$$\mathcal{F} = \mathcal{F}_1 \cap \mathcal{F}_2 \cap \ldots \cap \mathcal{F}_m \tag{5.15}$$

O *problema de otimização restrita com restrições de desigualdade*, em sua forma geral, trata da questão de determinação do ponto de mínimo $\mathbf{x}^*$ de uma função, dentro de uma *região factível* $\mathcal{F}$ definida dessa forma. Nas subseções que se seguem, mostraremos algumas formas do nosso Otimizador lidar com tal problema.

5.3.3 Barreiras e Penalidades

A primeira maneira de tentar adaptar os métodos de otimização que foram formulados para problemas de otimização irrestrita para o caso agora em análise, com restrições de desigualdade, é a técnica das *barreiras* e *penalidades*. A idéia é modificar a função-objetivo, acrescentando um termo que, dentro da região factível, afeta pouco a função, mas que nas proximidades da fronteira da região factível (no caso das *barreiras*) ou no exterior da região factível (no caso das *penalidades*) muda bastante a função, "impedindo" o Otimizador (ou seja, o algoritmo de otimização) de sair da região factível.

[13]Observe que essa notação, utilizando o índice i, significa o mesmo que uma enumeração de todas as funções e regiões: $\mathcal{F}_1$ correspondendo à região em que $g_1(\cdot) \leq 0$, $\mathcal{F}_2$ correspondendo à região em que $g_2(\cdot) \leq 0$, e assim por diante.

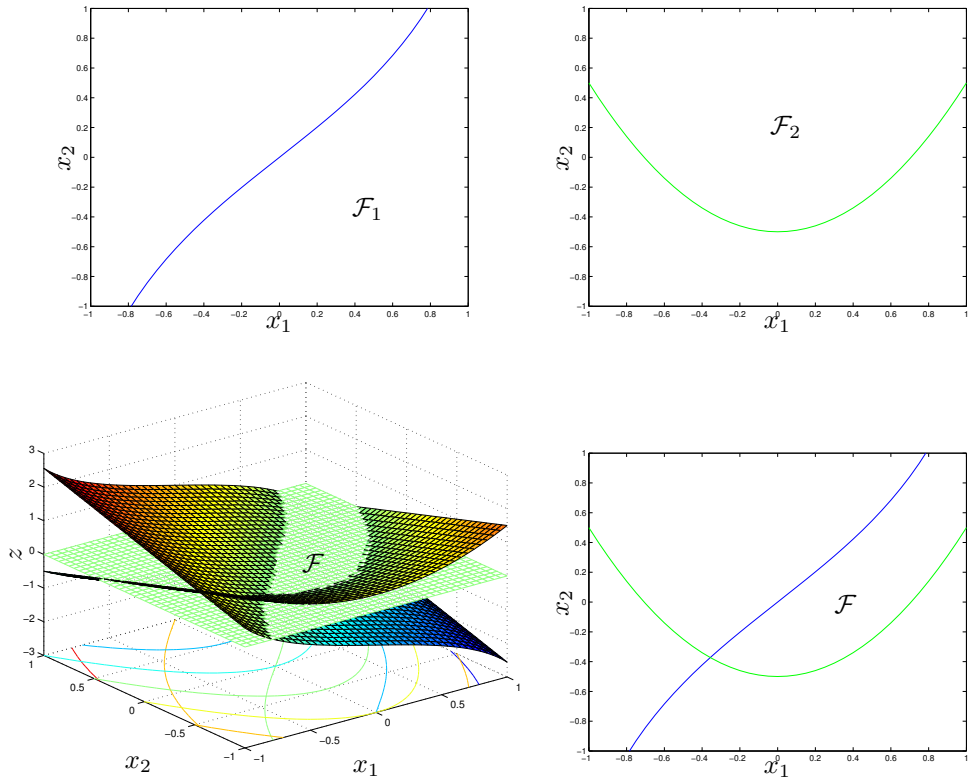


Figura 5.18: A região $\mathcal{F}_1$ corresponde aos pontos em que a função $g_1(\cdot)$ é negativa (figura superior esquerda). A região $\mathcal{F}_2$ corresponde aos pontos em que a função $g_2(\cdot)$ é negativa (figura superior direita). A interseção dessas duas regiões, $\mathcal{F}$, corresponde aos pontos em que ambas as funções são negativas, simultaneamente (figura inferior direita). A figura inferior esquerda mostra as superfícies $z = g_1(x)$, $z = g_2(x)$, assim como sua interseção com o plano $z = 0$ e suas curvas de nível. Pode-se observar também nesta figura a região $\mathcal{F}$.

Em termos matemáticos, o problema de otimização original, definido por:

$$\begin{aligned} \mathbf{x}^* &= \arg\min_{\mathbf{x}} f(\mathbf{x}) \\ \text{sujeito a: } &\{\mathbf{g}(\mathbf{x}) \leq 0 \end{aligned} \tag{5.16}$$

é substituído pelo problema:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}) + F(\mathbf{x}) \tag{5.17}$$

A função $F(\cdot)$ deve ser muito pequena (ou zero) no interior da região factível, de tal forma que o ponto de mínimo de $f(\cdot)$ seja muito parecido com o ponto de mínimo de $f(\cdot) + F(\cdot)$.

No caso de métodos de *barreiras*, a função $F(\cdot)$ deve crescer muito rapidamente quando estamos perto da fronteira da região factível. A idéia é que o Otimizador, ao se aproximar dessa fronteira, verifique um súbito aumento da função $f(\mathbf{x}) + F(\mathbf{x})$ (que é a função que ele está otimizando), de forma que ele não caminha em direção a essa fronteira. O Otimizador, se tiver iniciado a busca no interior da região factível, irá sempre ficar nessa região, portanto[14]. Esse tipo de método é denominado de *barreira* porque a função $F(\cdot)$ cria uma espécie de "barreira", que impede que o Otimizador atinja a fronteira da região factível. A figura 5.19 ilustra uma função modificada com uma barreira, para uma situação de otimização em uma única variável.

Os métodos de *penalidades*, por outro lado, são obtidos se se faz a função $F(\cdot)$ crescer rapidamente do lado de fora da região factível, para valores que aumentam à medida em que nos afastamos dessa região. A idéia, neste caso, é fazer com que o Otimizador, ao sair da região factível, encontre um crescimento da função $f(\mathbf{x}) + F(\mathbf{x})$ que ele está otimizando, de forma que ele tende a voltar ao interior da região. Esse tipo de método é denominado de *penalidade* porque a função $F(\cdot)$ faz com que o Otimizador seja apenado (ou seja, sofra uma penalidade) caso ultrapasse a fronteira da região factível, sendo tanto maior a penalidade quanto mais o Otimizador se afastar dessa região. A figura 5.20 ilustra uma função de penalidade.

A figura 5.21 sobrepõe os gráficos das figuras 5.19 e 5.20, que mostram uma função barreira e uma função penalidade para o tratamento da mesma restrição.

[14]Deve-se tomar o cuidado, ao utilizar um método de barreira, para que o ponto inicial já esteja no interior da região factível.

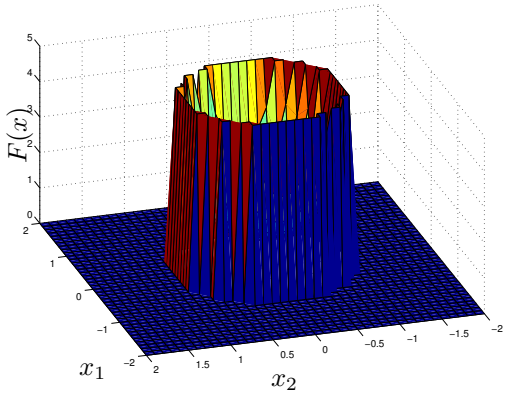


Figura 5.19: Ilustração de uma função de barreira, construída para garantir a restrição de que a otimização deva ocorrer no interior de um círculo de raio igual a 1, que seria a região factível de um problema de otimização. Essa função, somada à função objetivo, teria o papel de "impedir" a saída de um Otimizador do interior desse círculo de raio 1 que corresponde à região factível.

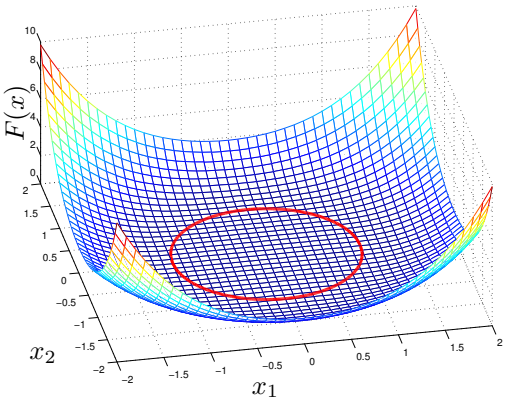


Figura 5.20: Ilustração de uma função de penalidade. A região factível corresponde ao interior do círculo indicado em vermelho. A função de penalidade é igual a zero no interior da região factível, e cresce rapidamente à medida em que o ponto se afasta dessa região.

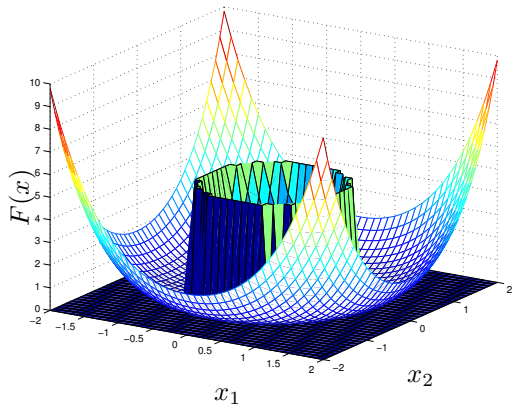


Figura 5.21: Sobreposição dos gráficos das figuras 5.19 e 5.20, de forma a mostrar uma função barreira e uma função penalidade para a mesma restrição. No caso, a restrição define como região factível o interior do círculo de raio 1 centrado na origem.

Deve-se notar que, uma vez que a função objetivo esteja modificada, seja por uma função de barreira, seja por uma de penalidade, a resultante função modificada pode ser otimizada utilizando os mesmos métodos que foram desenvolvidos para o caso da otimização sem restrições. Tipicamente, serão empregados métodos de direções de busca para resolver problemas formulados dessa maneira[15].

5.3.4 Composição pelo Máximo

Embora seja possível utilizar as funções de penalidade para lidar com as restrições de problemas de otimização nos casos em que o mecanismo de otimização a ser empregado é do tipo *exclusão de regiões*, há uma forma mais natural de tratar as restrições nesse caso. Considera-se, primeiro, a seguinte função:

$$G(\mathbf{x}) = \max(g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_m(\mathbf{x})) \tag{5.18}$$

[15]Deve-se notar que, em particular, as funções de barreira não seriam funcionais se empregados nem junto com métodos de exclusão de regiões nem junto com métodos de populações. Já as funções de penalidade não causariam essas dificuldades, e poderiam ser empregadas com qualquer sistema de otimização. O leitor é convidado a explicar por quê isso ocorre.

A função $G(\cdot)$ é a chamada *composição pelo máximo* das funções $g_i(\cdot)$. O leitor é convidado a examinar a curva de nível $G(\mathbf{x}) = 0$. Essa curva de nível corresponde exatamente à fronteira da região factível do problema. Cada curva de nível $G(\mathbf{x}) = \alpha$, para $\alpha > 0$, corresponde a uma curva (ou hipersuperfície, em dimensões maiores que dois) fechada que é exterior às curvas correspondentes a valores menores de α, e todas têm em seu interior a região factível do problema (a curva correspondente a $\alpha = 0$)[16].

Imagine-se agora a aplicação de uma técnica de otimização por exclusão de regiões à função $G(\cdot)$. Se o Otimizador começar, nesse caso, em um ponto fora da região factível, a primeira exclusão será de um semi-espaço que garantidamente não contém a região factível, ficando para continuar a ser examinado o semi-espaço que contém a região factível. O processo continua até que, certamente, o Otimizador finalmente cai dentro da região factível.

Para fechar o procedimento a ser aplicado, uma vez dentro da região factível do problema, aplica-se um passo convencional de "exclusão de região", utilizando a função objetivo $f(\cdot)$ para determinar a exclusão. O significado desse passo é: após esse corte, o Otimizador permanece com o semi-espaço que contém a parcela da região factível na qual o ponto de ótimo do problema se encontra (ou seja, elimina-se a parcela da região factível em que o ponto de ótimo não se encontra). Essas operações são ilustradas na figura 5.22.

O algoritmo resultante da seqüência dessas operações pode oscilar, levando o Otimizador sucessivamente para dentro e para fora da região factível. No entanto, como no caso irrestrito, o volume da região considerada necessariamente diminui a cada passo, sendo que o ponto de ótimo permanece nessa região. O Otimizador, assim, termina arbitrariamente próximo do ótimo.

5.4 Otimização com Restrições de Igualdade

Consideremos agora o problema de otimização com *restrições de igualdade*:

$$\begin{aligned} \mathbf{x}^* = &\ \arg\ \min\ f(\mathbf{x}) \\ &\\ \text{sujeito a:}\ &\ \{\mathbf{h}(\mathbf{x}) = \mathbf{0} \end{aligned} \tag{5.19}$$

[16]Para fazermos essa afirmativa, na verdade, estamos assumindo que as funções $g_i(\cdot)$ sejam todas *convexas* ou, pelo menos, *quasi-convexas*.

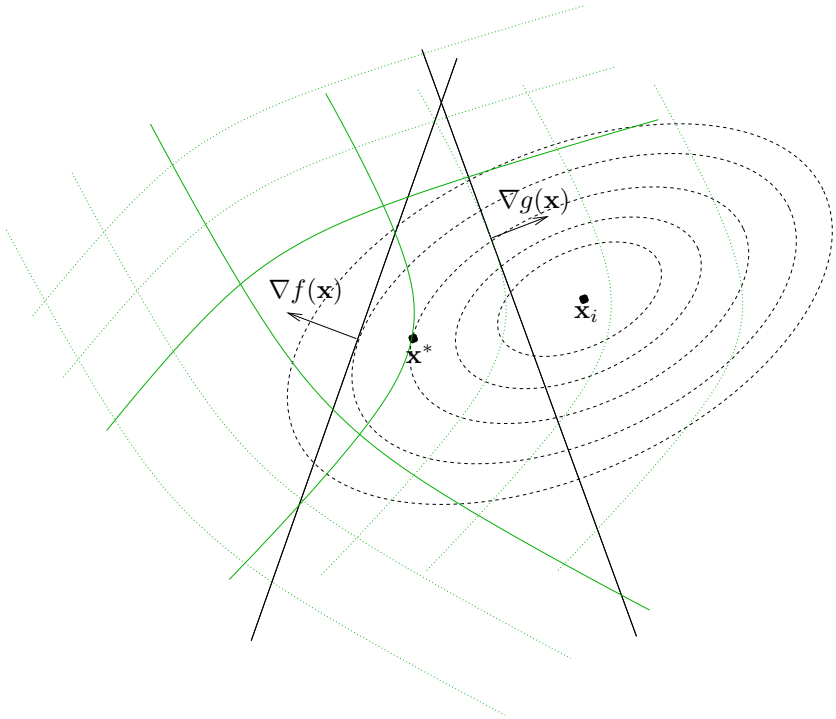


Figura 5.22: Ilustração da aplicação do processo de exclusão de região em um problema de otimização restrita. São mostradas, na figura, as curvas de nível da função objetivo $f(\mathbf{x})$, ao redor do mínimo irrestrito $\mathbf{x}_i$, e as curvas de nível das restrições $\mathbf{g}(\mathbf{x})$. Estas são mostradas no exterior da região factível, sendo mostradas, em traço mais grosso, as curvas correspondentes a $g_i(\mathbf{x}) = 0$ (ou seja, as curvas que definem as fronteiras da região factível). O ponto de ótimo do problema é representado por $\mathbf{x}^*$. São mostrados os vetores gradientes da função objetivo, $\nabla f(\mathbf{x})$, em um ponto factível, e gradiente de uma restrição violada, $\nabla g(\mathbf{x})$, em um ponto factível. Deve-se observar que as retas normais a ambos os vetores gradiente definem cortes do plano tais que o semi-plano oposto ao vetor gradiente, em ambos os casos, necessariamente contém a solução $\mathbf{x}^*$. (No caso do corte feito no ponto infactível, o semi-plano oposto ao gradiente contém de fato toda a região factível).

Essa descrição do problema significa, conforme já foi visto, que o ponto de ótimo $\mathbf{x}^*$ a ser determinado deve satisfazer às p equações:

$$\begin{aligned} h_1(\mathbf{x}^*) &= 0 \\ h_2(\mathbf{x}^*) &= 0 \\ &\vdots \\ h_p(\mathbf{x}^*) &= 0 \end{aligned} \tag{5.20}$$

Num espaço de n dimensões, cada uma dessas equações pode ser interpretada como uma descrição de um conjunto de pontos (os pontos $\mathbf{x}$ que a satisfazem) que fazem parte de uma superfície de dimensão $n-1$. Por exemplo, num espaço de dimensão 3, uma equação dessas significa uma superfície no sentido convencional, dotada de duas dimensões (algo como uma "folha" curvada). Essa superfície corresponde ao conjunto dos pontos factíveis do problema de otimização, se ele envolver apenas uma restrição de igualdade. No caso de p restrições de igualdade, o conjunto factível corresponde à interseção de todas as superfícies (cada uma associada a uma das restrições de igualdade).

O espaço que estamos considerando, na série de exemplos que vem sendo apresentada neste capítulo, possui apenas duas dimensões. Assim, o lugar geométrico definido por uma equação do tipo:

$$h_1(\mathbf{x}) = 0 \tag{5.21}$$

corresponde a um objeto de dimensão um, ou seja, uma linha (possivelmente curva). Este será o conjunto factível de um problema de otimização que tiver (5.21) como restrição. A figura 5.23 mostra um exemplo dessa situação.

Das técnicas mostradas anteriormente para tratar de problemas de otimização com restrições de desigualdade, duas simplesmente não funcionam para o caso de restrições de igualdade: o método de barreiras e o método de composição pelo máximo. A razão disso é que ambas as técnicas dependem da existência de pontos que sejam *interiores* à região factível do problema para funcionarem, e as regiões factíveis de restrições de igualdade não possuem pontos interiores[17]. A técnica de penalidades, por sua vez, pode ser empregada.

[17]Pontos interiores a uma região são pontos que pertencem a essa região e não estão

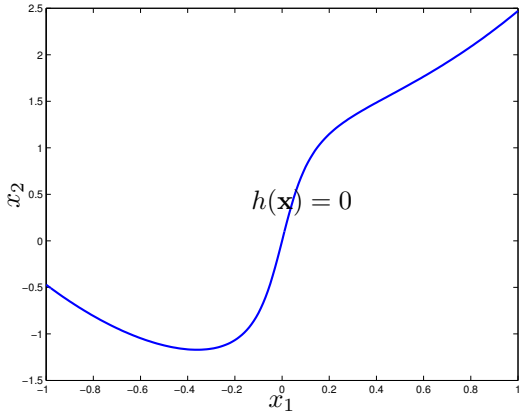


Figura 5.23: A linha corresponde ao lugar geométrico dos pontos que satisfazem $h(\mathbf{x}) = 0$. Essa linha é a região factível de um problema de otimização com essa restrição.

5.5 Otimização Linear

Um caso especial particularmente importante do problema de otimização ocorre quando tanto a função objetivo quanto as funções de restrição são lineares[18]. Esse é o chamado problema de *otimização linear*:

$$\begin{aligned} \mathbf{x}^* &= \arg\min \mathbf{c}'\mathbf{x} \\ \text{sujeito a: } &\{A\mathbf{x} \leq \mathbf{b} \end{aligned} \tag{5.22}$$

sendo $\mathbf{c}$ um vetor de dimensão n (mesmo tamanho que $\mathbf{x}$), A uma matriz $\mathbb{R}^{m \times n}$ e $\mathbf{b}$ um vetor de dimensão m. Claramente, a função objetivo desse problema é a função linear:

$$f(x) = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \tag{5.23}$$

em sua fronteira. Claramente, todos os pontos factíveis de problemas de otimização com restrições de igualdade estão na fronteira da região factível, isto é, possuem algum ponto vizinho fora dessa região.

[18]No caso das restrições, uma terminologia mais precisa iria dizer que são *afins* e não *lineares*.

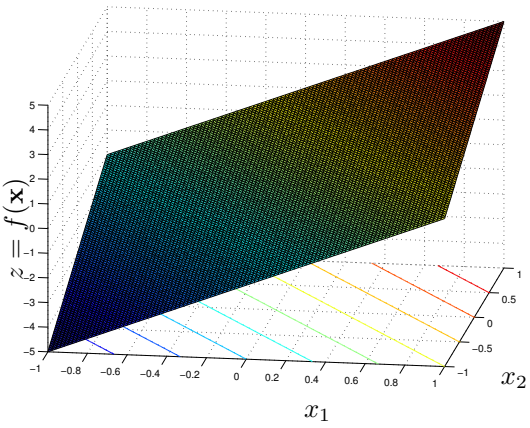


Figura 5.24: Superfície correspondente à função objetivo linear $f(\mathbf{x}) = \mathbf{c}'\mathbf{x}$. Na figura, estão representadas também as curvas de nível da função, que são retas paralelas.

e o conjunto de restrições corresponde às m desigualdades:

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2 \\ &\vdots \\ &a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m \end{aligned} \tag{5.24}$$

A otimização linear é particularmente importante por duas razões: Primeiro, um número muito grande de situações práticas é modelado pela formulação linear. Segundo, devido à sua estrutura peculiar, problemas de otimização linear podem ser resolvidos muito mais rapidamente que problemas de otimização não-linear com o mesmo número de variáveis e o mesmo número de restrições. Assim, algoritmos especializados para resolver apenas problemas lineares são capazes de lidar com problemas muito grandes (muito maiores que aqueles que poderiam ser resolvidos no caso não-linear geral).

Vamos examinar essa estrutura peculiar que torna tão favorável a otimização linear. No caso de duas variáveis de otimização, a superfície representativa da função linear é simplesmente um plano, e suas curvas de nível são retas paralelas. Isso é mostrado na figura 5.24.

O problema de otimização de uma função linear não faz sentido se não estiver acompanhado de restrições, pois o ponto que minimiza tal função

objetivo encontra-se no infinito[19]. Examinemos o que são as restrições do problema de otimização linear. Num espaço de n dimensões, a desigualdade:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \tag{5.25}$$

representa um semi-espaço. A fronteira que separa a região factível da infactível corresponde a um hiperplano nesse espaço. No caso de duas dimensões, a desigualdade:

$$a_{11}x_1 + a_{12}x_2 \leq b_1 \tag{5.26}$$

define um semi-plano como região factível, e a fronteira dessa região factível corresponde à reta $a_{11}x_1 + a_{12}x_2 = b_1$. Consideremos agora várias restrições de desigualdade em duas dimensões:

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 \leq b_1 \\ &a_{21}x_1 + a_{22}x_2 \leq b_2 \\ &\vdots \\ &a_{m1}x_1 + a_{m2}x_2 \leq b_m \end{aligned} \tag{5.27}$$

Como cada uma dessas restrições de desigualdade define um semi-plano, as várias restrições de desigualdade correspondem à interseção de vários semi-planos, o que define um poliedro. Isso é mostrado na figura 5.25.

Observemos agora, na figura 5.26, a superposição das curvas de nível de uma função objetivo linear com uma região factível linear. O dado relevante a ser observado é que, num problema linear, o ponto de ótimo *necessariamente se encontra sobre um vértice do poliedro factível.*

O leitor deve se convencer de que seria impossível, num problema linear, que o mínimo da função objetivo estivesse no interior da região factível. Seria também impossível que esse mínimo estivesse em um ponto da fronteira da região factível sem estar em um dos vértices dessa fronteira[20]. Assim, uma possível estratégia para resolver problemas lineares, seria fazer o Otimizador percorrer apenas o conjunto dos vértices da região factível, escolhendo dentre esses vértices aquele com menor valor de função objetivo. É possível implementar métodos bastante eficientes de otimização com base em tal estratégia:

[19]Em outras palavras, não existe nenhum mínimo local irrestrito de uma função objetivo linear.

[20]No entanto, seria possível que houvesse múltiplos mínimos, incluindo pontos diversos da fronteira, dentre esses necessariamente pelo menos um dos vértices.

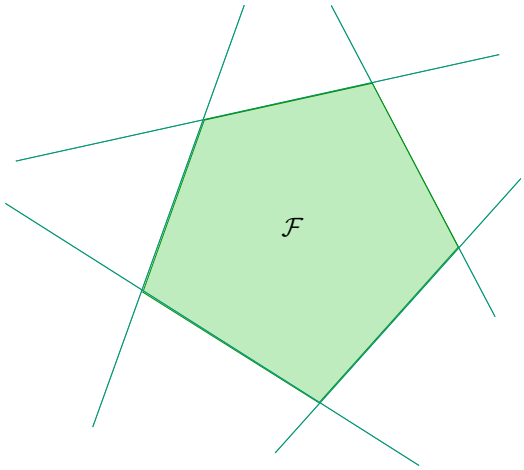


Figura 5.25: Região factível $\mathcal{F}$, correspondente a várias restrições lineares de desigualdade. Cada reta que contém um dos lados do poliedro factível corresponde à fronteira de uma restrição de desigualdade.

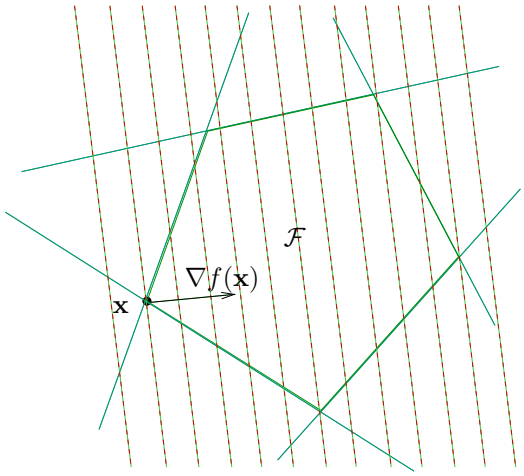


Figura 5.26: O vetor gradiente da função objetivo, $\nabla f(\mathbf{x})$, mostrado no ponto $\mathbf{x}$, é constante em todo o espaço, pois a função objetivo é linear. As linhas tracejadas correspondem às curvas de nível da função objetivo, sendo que elas correspondem a valores cada vez menores de função objetivo quando se caminha da direita para a esquerda. Dessa forma, o ponto $\mathbf{x}$ indicado na figura é o de menor valor de função objetivo dentro da região factível $\mathcal{F}$, correspondendo ao ponto em que a curva de nível de menor valor toca a região factível.

esses são os chamados métodos Simplex. Esse tipo de lógica, largamente empregada no contexto da otimização linear, é fundamentalmente diferente dos procedimentos que podem ser utilizados na otimização não-linear[21].

[21]Devemos entretanto informar o leitor que, recentemente, outras estratégias de otimização linear, denominadas *métodos de pontos interiores*, vêm ganhando a preferência dos usuários, estratégias essas que têm semelhança com métodos de otimização não-linear.

Capítulo 6

Métodos de Direções de Busca

Os primeiros métodos de otimização (minimização) de funcionais não-lineares foram desenvolvidos a partir da idéia básica de fazer o algoritmo evoluir encontrando novos pontos situados em direções para as quais o funcional decresça, em relação ao ponto corrente.

A versão mais primitiva dessa família de métodos vem a ser o "Algoritmo do Gradiente": dado um ponto inicial do espaço de busca, obtém-se um novo ponto situado sobre a reta definda por esse ponto e pelo gradiente da função-objetivo. Essa é a direção para a qual, localmente, a função mais rapidamente decresce (no sentido contrário ao do gradiente). Determina-se o novo ponto como sendo aquele em que a função objetivo atinge o mínimo sobre essa reta (note-se que este é um problema de minimização de uma função de uma única variável). A partir desse novo ponto, repete-se o processo, até que seja satisfeito um critério de convergência.

Ao longo das décadas de 50 e 60 do século XX, tal método básico foi aperfeiçoado, para permitir que a direção de busca na qual era feita a busca unidimensional sofresse uma "correção". Tal correção levava em conta mais informações a respeito da função objetivo, além do valor de seu gradiente no ponto corrente: procurava-se também levar em consideração a curvatura da função. Aproximações de segunda ordem, por exemplo, levando em consideração estimativas da Hessiana da função objetivo, permitiram significativa aceleração de convergência dos métodos.

Os métodos aqui agrupados sob a denominação "de direção de busca" têm essa raiz, e possuem em comum as seguintes características:

- Cada novo ponto é obtido de um processo de otimização unidimensional que tem como ponto de partida o ponto anterior.

- A direção na qual é feita a busca unidimensional é uma função das avaliações anteriores da função objetivo.

O objetivo deste capítulo é o estudo dessa classe de métodos, sendo o conteúdo selecionado para fundamentar sua compreensão geral. Não houve a intenção de esgotar a apresentação de todas as variações já desenvolvidas de métodos classificados nesta categoria.

6.1 Estrutura Básica

Dado o problema (mono-objetivo) irrestrito:

$$x^* = \arg\min_x f(x) \tag{6.1}$$

e dado um ponto inicial $x_0 \neq x^*$, obtém-se uma seqüência x_k tal que $x_k \to x^*$ a partir do algoritmo de otimização. A família dos algoritmos de direção de busca possui a estrutura:

Algoritmo de Direção de Busca

$k \leftarrow 0$

enquanto (não critério de parada)

$d_k \leftarrow h(x_1, \ldots, x_k, f(x_1), \ldots, f(x_k))$

$\alpha_k \leftarrow \arg\min_\alpha f(x_k + \alpha d_k)$

$x_{k+1} \leftarrow x_k + \alpha_k d_k$

$k \leftarrow k + 1$

fim-enquanto

Nessa estrutura, $h(\cdot, \ldots, \cdot)$ é uma função que em geral será recursiva, isto é, não irá depender explicitamente dos pontos anteriores, mas irá armazenar sua influência em variáveis intermediárias. Um algoritmo irá diferir do outro essencialmente pela maneira como é calculada a direção de busca d_k, ou

seja, na escolha dessa função. No caso do Algoritmo do Gradiente, tem-se simplesmente que:

$$d_k = -\nabla f(x_k)$$

No caso do Algoritmo de Newton, tem-se que:

$$F_k = \text{hessiana}(f(x_k))$$

e

$$d_k = -F_k^{-1}\nabla f(x_k)$$

Tanto o gradiente quanto a Hessiana são determinados por meio de diversas avaliações da função $f(x)$, tendo em vista a regra básica de que este é o único tipo de informação disponível. A justificativa para a utilização dessas direções de busca será vista neste capítulo. Os métodos chamados de *quasi-Newton* substituem a avaliação da Hessiana da função objetivo pela construção de uma estimativa para essa Hessiana.

Os elementos para a construção de algoritmos de direções de busca são, portanto: (i) um método de cálculo de direções de busca, possivelmente envolvendo o cálculo de estimativas para o gradiente e para a Hessiana da função objetivo; (ii) um método de minimização de funções de uma única variável; (iii) um critério de decisão que permita afirmar que o algoritmo convergiu para uma solução satisfatória, podendo ser terminada sua execução. Esses elementos serão examinados a seguir. Antes disso, a natureza do processo de convergência intrínseco aos métodos de direção de busca é estudada, através do exame da convergência de um algoritmo de interesse apenas conceitual: o algoritmo de busca em direções aleatórias.

6.2 Busca em Direções Aleatórias

Considere-se o algoritmo definido por:

Algoritmo de Busca em Direções Aleatórias

$k \leftarrow 0$

enquanto (não critério de parada)

$d_k \leftarrow \text{rand}(n, 1)$

$\alpha_k \leftarrow \arg\ \min_\alpha f(x_k + \alpha d_k)$
$x_{k+1} \leftarrow x_k + \alpha_k d_k$
$k \leftarrow k + 1$

fim-enquanto

A função rand$(n, 1)$ é definida tal que sua saída é um vetor de n componentes aleatórias com distribuição Gaussiana, média 0 e variância 1. A convergência desse algoritmo para o ponto de mínimo de uma função unimodal é estabelecida no teorema a seguir.

Teorema 6.1 *Seja $f(x) : \mathbb{R}^n \mapsto \mathbb{R}$ uma função estritamente unimodal, e seja x_0 um ponto qualquer em seu domínio. A aplicação do* algoritmo de busca em direções aleatórias *a essa função, partindo desse ponto, produz uma seqüência $[f(x_k)]$ que se aproxima de forma monotônica do valor mínimo da função, $f(x^*)$.* □

DEMONSTRAÇÃO: A subrotina de minimização unidimensional embutida no algoritmo implica que, qualquer que seja a direção d_k escolhida:

$$f(x_k) \leq f(x_{k-1})$$

o que demonstra a monotonicidade da seqüência. A unimodalidade estrita de $f(x)$ implica que para todo ponto $x_k \neq x^*$ haverá possíveis direções d_k para as quais ocorra:

$$f(x_k + \alpha_k d_k) < f(x_k)$$

para algum valor de α_k. Se uma dessas direções não for escolhida, ocorrerá:

$$x_{k+1} = x_k$$

Do contrário:

$$x_{k+1} \neq x_k$$

$$f(x_{k+1}) < f(x_k)$$

Pela construção da função aleatória geradora do vetor d_k, há uma probabilidade não-nula de geração de direções em que ocorre a diminuição do valor da função, de forma que a aproximação fica demonstrada, ou seja:

$$\forall\ x_k \neq x^*\ \exists\ N \mid f(x_{k+N}) < f(x_k)$$

■

Note-se que o teorema 6.1 mostra que ocorre a *aproximação*, mas não a *convergência* para o ponto de mínimo x^*. De qualquer forma, este é um algoritmo que efetivamente funcionaria para a minimização de funções. A questão a ser observada é que uma escolha adequada da direção de busca d_k, em substituição à escolha aleatória, pode aumentar em muito a eficiência do algoritmo de minimização. Os diversos algoritmos de direções de busca surgem precisamente quando se propõem diferentes formas de se fazer tal escolha de uma direção.

6.3 Algoritmo do Gradiente

A primeira escolha razoável para uma direção de busca d_k é a da direção contrária à do gradiente da função no ponto corrente x_k. Essa escolha se justifica com a observação de que, localmente, essa é a direção na qual a função decresce mais rapidamente. Isso define o *Algoritmo do Gradiente*, esquematizado a seguir:

Algoritmo do Gradiente

$k \leftarrow 0$

enquanto (não critério de parada)

$g_k \leftarrow \text{gradiente}(f(\cdot), x_k)$
$d_k \leftarrow -g_k$
$\alpha_k \leftarrow \arg\ \min_\alpha f(x_k + \alpha d_k)$
$x_{k+1} \leftarrow x_k + \alpha_k d_k$
$k \leftarrow k + 1$

fim-enquanto

Esse algoritmo baseia-se apenas na informação local a respeito da variação da função em todas as direções do espaço (sintetizada no gradiente da

função). A única suposição implícita na aplicação desse algoritmo é a de que a função $f(x)$ seja diferenciável.

Os elementos construtivos desse algoritmo são examinados a seguir.

6.3.1 Cálculo do Gradiente

No contexto da teoria de otimização, a suposição mais geral a respeito da informação sobre o sistema sendo otimizado é: dispõe-se *apenas* de um algoritmo que, recebendo como entradas o vetor de variáveis de otimização, fornece o valor da função-objetivo para tal vetor. Não se dispõe portanto, em geral, de funções que explicitamente forneçam o gradiente da função objetivo para certa especificação do vetor de variáveis de otimização. Dessa forma, é necessário construir um algoritmo que forneça o gradiente.

O algoritmo mais simples que se pode imaginar para o cálculo numérico (aproximado) do gradiente de uma função é decorrência imediata da definição de gradiente, sendo substituída a fórmula diferencial por diferenças finitas. Seja $x \in \mathbb{R}^n$ o vetor de variáveis de otimização, e seja e_i o vetor definido por:

$$e_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \rightarrow \; i - \text{ésima posição} \tag{6.2}$$

Considere-se um certo $\delta > 0$, tal que $\delta \approx 0$. O algoritmo de cálculo do vetor gradiente g no ponto x pode ser definido como:

Algoritmo de Cálculo do Gradiente

para $i \leftarrow 1$ até n faça

$$g_i \leftarrow \frac{f(x + \delta e_i) - f(x)}{\delta}$$

fim-para

$$g \leftarrow \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}$$

NOTA 6.1 *Deve-se observar que o* Algoritmo de Cálculo do Gradiente *é* exato *para funções lineares, ou seja, para funções cuja série de Taylor termina no termo de primeira ordem. Nesse caso, o valor de δ pode ser qualquer: o cálculo será exato mesmo para δ grande.*

$\diamondsuit$

EXEMPLO 6.1 *Seja a função de duas variáveis $f(x)$, definida por:*

$$f(x) = 2x_1^2 + x_2^2 + 2x_1x_2 + x_1 - 2x_2 + 3$$

Analiticamente, o gradiente dessa função é dado por:

$$\nabla f(x) = \begin{bmatrix} 4x_1 + 2x_2 + 1 \\ 2x_1 + 2x_2 - 2 \end{bmatrix}$$

Por essa fórmula analítica, sabe-se que no ponto $x_o = [\ 0 \quad 0\ ]'$ o gradiente fica igual a:

$$\nabla f(x_o) = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

Utilizando-se o algoritmo de diferenças finitas, para $\delta = 0.0001$, obtém-se a estimativa do gradiente igual a:

$$\hat{\nabla} f(x_o) = \begin{bmatrix} 1.0002 \\ -1.9999 \end{bmatrix}$$

Deve-se notar que o Algoritmo do Gradiente não utiliza nenhuma informação analítica a respeito da função. A única informação utilizada é proveniente de avaliações da função em pontos. $\diamondsuit$

6.3.2 Otimização Unidimensional

A seguinte linha do algoritmo do gradiente agora é examinada:

$$\alpha_k \ \leftarrow \ \arg \min_{\alpha} f(x_k + \alpha d_k)$$

O cálculo de α_k é feito fixando-se o ponto atual x_k e uma direção de busca, d_k. Isso faz com que a função objetivo, $f(x)$, que originalmente seria de n variáveis (ou seja, dependeria de um vetor x de dimensão n) torne-se agora uma função de uma única variável real, α.

A otimização de funções de uma única variável (em uma única dimensão, portanto) é tarefa substancialmente mais simples que a otimização em diversas dimensões. Podem-se construir algoritmos diversos para resolver o problema, baseados em premissas diversas a respeito da função a ser otimizada. Uma premissa comum, que necessariamente possui validade local em alguma vizinhança de um ponto de ótimo estrito, é a de que a função-objetivo possua um único mínimo local no domínio em questão. Um algoritmo de otimização unidimensional pode ser construído com fundamento nessa premissa, a partir do teorema a seguir.

Teorema 6.2 *Sejam uma função $f(\cdot) : \mathbb{R} \mapsto \mathbb{R}$. Seja um domínio $[a, b] \subset \mathbb{R}$, no qual f possui um único mínimo local x^*. Sejam ainda dois pontos x_a e x_b tais que*

$$a < x_a < x_b < b \tag{6.3}$$

Se ocorrer

$$f(x_a) < f(x_b) \tag{6.4}$$

então a solução minimizante x^* não se encontra *no intervalo $[x_b, b]$, e se ocorrer*

$$f(x_a) > f(x_b) \tag{6.5}$$

então a solução minimizante x^* não se encontra *no intervalo $[a, x_a]$* □

DEMONSTRAÇÃO: Tome-se o intervalo $[a, x_b]$. Nesse intervalo, há algum ponto x_o para o qual $f(x_o) \leq f(x)\ \forall\ x \in [a, x_b]$ e $x_o \neq x_b$, pela hipótese (6.4). Logo, x_o é um mínimo local no segmento $[a, x_b]$. Como $x_o \neq x_b$, tem-se que no intervalo $[a, b]$ x_o permanece sendo mínimo local. Acrescentando-se agora a hipótese de que só há um único mínimo local em $[a, b]$, obtém-se que $x^* = x_o$, que é o resultado pretendido. Para o outro lado do segmento, o argumento é análogo. ■

Com esse teorema, é possível construir um algoritmo que se fundamenta na lógica de excluir, a cada passo, um trecho do segmento considerado, de forma a fazê-lo contrair-se. Quando o segmento estiver suficientemente "pequeno", pode-se considerar que ocorreu a convergência para o ponto de

mínimo. A precisão dessa convergência (ou seja, o erro máximo cometido) será igual à metade do comprimento remanescente.

Existem, claramente, formas de escolher os pontos x_a e x_b dentro do segmento, de forma a maximizar (em média) o comprimento do intervalo a ser excluído a cada passo, assim minimizando o número de iterações necessário para se atingir determinada precisão. Uma escolha freqüentemente adotada é definida pela "seção áurea": escolhem-se x_a e x_b de forma que:

$$\begin{aligned} x_b - a &= 0.618(b-a) \\ b - x_a &= 0.618(b-a) \end{aligned} \tag{6.6}$$

O fator 0.618 corresponde à "razão áurea", utilizada pelos antigos gregos para definir a razão dos lados adjacentes de um retângulo que seria "perfeita" sob o ponto de vista estético.

Com esta escolha, o algoritmo de minimização de uma função real no intervalo $[a, b]$ para se atingir uma precisão $\epsilon/2$ pode ser definido como:

Algoritmo da Seção Áurea

$x_a \leftarrow b - 0.618(b-a)$

$x_b \leftarrow a + 0.618(b-a)$

$f_a \leftarrow f(x_a)$

$f_b \leftarrow f(x_b)$

enquanto $b - a > \epsilon$

    se $f_a < f_b$ então

        $b \leftarrow x_b$

        $x_b \leftarrow x_a$

        $x_a \leftarrow b - 0.618(b-a)$

        $f_b \leftarrow f_a$

        $f_a \leftarrow f(x_a)$

    senão

        $a \leftarrow x_a$

$x_a \leftarrow x_b$
$x_b \leftarrow a + 0.618(b - a)$
$f_a \leftarrow f_b$
$f_b \leftarrow f(x_b)$

fim-se

fim-enquanto

$$x \leftarrow \frac{a+b}{2}$$

Claramente, a cada passo do algoritmo o comprimento do intervalo $[a, b]$ é multiplicado por um fator menor ou igual a 0.618, de forma que pode-se calcular o número esperado máximo de passos para atingir a precisão desejada:

$$d_k \leq 0.618^{k-1} d_1 \tag{6.7}$$

sendo d_k o comprimento do intervalo $[a, b]$ no passo k.

O leitor deve estar ciente de que é possível construir outros algoritmos, mais eficientes, para a otimização de funções de uma única variável, que levam em consideração aproximações diversas para a função. Para maiores detalhes, pode-se consultar a referência (Luenberger 1984).

EXEMPLO 6.2 *Tome-se a mesma função do Exemplo 6.1.*

$$f(x) = 2x_1^2 + x_2^2 + 2x_1x_2 + x_1 - 2x_2 + 3$$

O gradiente da função é dado por:

$$\nabla f(x) = \begin{bmatrix} 4x_1 + 2x_2 + 1 \\ 2x_1 + 2x_2 - 2 \end{bmatrix}$$

No ponto $x_o = [\ -1 \quad 1\ ]'$*, o gradiente fica igual a:*

$$\nabla f(x_o) = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

A função $f(x)$*, tomada a partir do ponto* x_o *na direção de* $-\nabla f(x_o)$ *pode ser encontrada analiticamente por substituição da variável vetorial* x *pela variável escalar* α*, feita segundo:*

$$x = x_o - \alpha \nabla f(x_o)$$

ou:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} - \alpha \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} \alpha - 1 \\ 2\alpha + 1 \end{bmatrix}$$

A função unidimensional fica:

$$f(\alpha) = 2(\alpha-1)^2+(2\alpha+1)^2+2(\alpha-1)(2\alpha+1)+(\alpha-1)-2(2\alpha+1)+3 = 10\alpha^2-5\alpha+1$$

Essa função possui mínimo para:

$$\frac{df(\alpha)}{d\alpha} = 20\alpha - 5 = 0$$

ou seja, para $\alpha = \frac{1}{4}$*. Para esse valor de* α*, o ponto no espaço dos vetores* x *fica:*

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} \frac{-3}{4} \\ \frac{3}{2} \end{bmatrix} = \begin{bmatrix} -0.75 \\ 1.5 \end{bmatrix}$$

Com o algoritmo da seção áurea, obtém-se uma estimativa do ponto de mínimo igual a:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -0.75010 \\ 1.49980 \end{bmatrix}$$

para uma precisão estabelecida de $\epsilon = 0.001$*. Deve-se notar que o ponto determinado não é o ponto de mínimo global da função* $f(x)$*, nem corresponde a um mínimo local dessa função, pois o gradiente não se anula nesse ponto. O mínimo apenas diz respeito à minimização da função sobre a reta definida pelo ponto* x_o *e pelo vetor* $\nabla f(x_o)$*.* ♢

6.3.3 Critérios de Parada

Após produzir uma seqüência de estimativas da função objetivo, avaliadas para uma seqüência de pontos do espaço de variáveis de otimização, o algoritmo de otimização eventualmente deverá se aproximar de um ponto de mínimo local (ótimo local) da função. Como a aproximação para o ótimo ocorre de forma assintótica, é necessário em algum momento tomar a decisão de interromper o algoritmo, sendo a aproximação obtida considerada o valor ótimo alcançado.

Alguns critérios possíveis, para a tomada dessa decisão, seriam:

Estabilização do valor da função-objetivo

Caso o valor da função-objetivo, em um certo número de iterações, não varie mais que certo percentual da diferença entre seu valor máximo ocorrido em todo o processo de otimização e seu valor mínimo verificado também em todo o processo, é possível interromper o algoritmo supondo que dificilmente viriam a ocorrer melhorias significativas da função-objetivo com essa continuidade.

A seguir é apresentado um trecho de algoritmo que exemplifica a construção desse critério, o qual considera como estabilizado um algoritmo que varia, nas últimas cinco iterações, menos de 0.1% da "amplitude" Δ_f da função objetivo, sendo f_{max} e f_{min} respectivamente o máximo e o mínimo valor ocorrido para a função objetivo durante toda a execução.

Critério de Parada: Função Objetivo

$\Delta_f \leftarrow f_{max} - f_{min}$

$f_{5+} \leftarrow \max\{f(x_k), f(x_{k-1}), f(x_{k-2}), f(x_{k-3}), f(x_{k-4}), f(x_{k-5})\}$

$f_{5-} \leftarrow \min\{f(x_k), f(x_{k-1}), f(x_{k-2}), f(x_{k-3}), f(x_{k-4}), f(x_{k-5})\}$

$\delta_f \leftarrow f_{5+} - f_{5-}$

se $\delta_f < 0.001\Delta_f$

então *parada* ← `true`

senão *parada* ← `false`

NOTA 6.2 *O leitor deve estar atento para o fato de que é necessário calcular o valor Δ_f, não sendo recomendável utilizar, em seu lugar, nem f_{min} nem f_{max}. Fica para o leitor a tarefa de explicar que problemas ocorreriam caso fossem feitas tais escolhas.*

♢

NOTA 6.3 *Seria entretanto possível utilizar, para Δ_f, alguma definição um pouco mais sofisticada, que por exemplo excluísse alguns dos máximos valores ocorridos para a função objetivo antes do cálculo de f_{max}. Tal procedimento aumenta a complexidade do algoritmo, mas pode torná-lo mais estável.*

♢

Estabilização do vetor de variáveis de otimização

Outra alternativa para o problema de formulação de critérios de parada de algoritmos de otimização seria a constatação de que o vetor de variáveis se estabilizou em algum ponto do espaço.

A seguir é apresentado um trecho de algoritmo que exemplifica a construção desse critério, o qual considera como estabilizado um algoritmo cujo vetor de variáveis varia, nas últimas cinco iterações, menos de 0.1% da "faixa de variação" verificada do vetor de variáveis ao longo de toda a execução. Sejam x_{max} o vetor cujas componentes são o máximo valor ocorrido para cada componente do vetor de variáveis durante toda a execução, e x_{min} o vetor cujas componentes são o mínimo valor ocorrido para cada componente do vetor de variáveis durante toda a execução do algoritmo. As operações com vetores são entendidas como operações realizadas sobre cada uma das componentes dos operandos. A comparação entre dois vetores será verdadeira se cada uma das comparações de componentes for verdadeira.

Critério de Parada: Vetor de Variáveis

$\Delta_x \leftarrow x_{max} - x_{min}$

$x_{5+} \leftarrow \max\{x_k, x_{k-1}, x_{k-2}, x_{k-3}, x_{k-4}, x_{k-5}\}$

$x_{5-} \leftarrow \min\{x_k, x_{k-1}, x_{k-2}, x_{k-3}, x_{k-4}, x_{k-5}\}$

$\delta_x \leftarrow x_{5+} - x_{5-}$

se $\delta_x < 0.001\Delta_x$

então *parada* ← `true`

senão *parada* ← `false`

NOTA 6.4 *Novamente, observa-se que não é recomendável utilizar, para construir esse critério de parada, nem x_{max}, nem x_{min}, nem x_{k-1} (embora essa última alternativa seja freqüentemente usada na literatura) em substituição a Δ_x. Fica para o leitor a tarefa de explicar que problemas ocorreriam nesses casos.*

♢

Anulação do Vetor Gradiente

Por fim, é possível ainda determinar o final de um processo de otimização com uma informação a respeito do vetor gradiente da função objetivo. Sabe-se que, se a função-objetivo é diferenciável, seu gradiente será nulo em seus pontos de mínimos locais. Pode-se portanto detectar a ocorrência desses mínimos pela monitoração do valor da norma do vetor gradiente.

No trecho de algoritmo a seguir, é mostrada uma implementação desse teste sobre o vetor $g(x)$, que é o gradiente da função-objetivo $f(x)$. A base de comparação adotada é o máximo valor da norma do gradiente ocorrido ao longo de toda a execução, denotado por M_{max}.

Critério de Parada: Vetor Gradiente

$M_g = \max\{\|g(x_k)\|, \|g(x_{k-1}\|, \|g(x_{k-2})\|\}$

se $M_g < 0.001 M_{max}$

então *parada* ← `true`

senão *parada* ← `false`

EXEMPLO 6.3 *Considere-se novamente a função $f(x)$, utilizada nos Exemplos 6.1 e 6.2:*

$$f(x) = 2x_1^2 + x_2^2 + 2x_1x_2 + x_1 - 2x_2 + 3 \tag{6.8}$$

O gradiente da função é dado por:

$$\nabla f(x) = \begin{bmatrix} 4x_1 + 2x_2 + 1 \\ 2x_1 + 2x_2 - 2 \end{bmatrix}$$

O mínimo global dessa função ocorre no ponto em que o gradiente se anula, ou seja:

$$x^* = \begin{bmatrix} -1.5 \\ 2.5 \end{bmatrix}$$

Nesse ponto, a função exibe o valor $f(x^*) = -0.25$. *A minimização dessa função será feita pelo Algoritmo do Gradiente, com o ponto inicial fixado em:*

$$x(1) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

O índice entre parêntesis denota o número da iteração. A seqüência de pontos encontrada durante a execução do Algoritmo do Gradiente é mostrada na Tabela 6.1 (o cálculo do gradiente foi feito, neste caso, utilizando a fórmula analítica acima). Deve-se observar que o último critério de convergência a ser satisfeito, nesse caso, foi o da estabilização do vetor de variáveis de otimização. Pelos outros dois critérios, o processo de otimização teria sido interrompido antes. O ponto de ótimo estimado, ao final de 27 iterações do algoritmo, é dado por:

$$\hat{x} = \begin{bmatrix} -1.4999 \\ 2.4998 \end{bmatrix}$$

A função objetivo, nesse ponto, vale $f(\hat{x}) = -0.25000$ *(valor igual ao ótimo exato até pelo menos a quinta casa decimal). A evolução dos três critérios de parada é mostrada na figura 6.1.*

$\diamondsuit$

6.3.4 Convergência

Pode-se mostrar, usando o *teorema da convergência global*, que o Algoritmo do Gradiente converge para a solução dos problemas de otimização mediante as condições formuladas na proposição a seguir.

Proposição 6.1 *Seja o problema de otimização irrestrito definido por:*

$$x^* = \arg \min_x f(x) \tag{6.9}$$

sendo $x \in \mathbb{R}^n$, *com* $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ *uma função contínua. Então o* Algoritmo do Gradiente *irá convergir para* x^* *para todo ponto inicial* x_0 *situado na bacia de atração de* x^*. $\square$

Tabela 6.1: Seqüência de valores das coordenadas do vetor de otimização, x_1 e x_2, da função objetivo, $f(x)$ e das coordenadas do vetor gradiente, g_1 e g_2, para a otimização da função descrita na equação (6.8). O índice da iteração é dado por k.

k	x_1	x_2	$f(x)$	g_1	g_2
1	-1.0000	1.0000	1.0000	-1.0000e-00	-2.0000e+00
2	-0.7500	1.5000	0.3750	1.0000e+00	-4.9997e-01
3	-1.2500	1.7500	0.0625	-4.9996e-01	-1.0000e+00
4	-1.1250	2.0000	-0.0937	5.0006e-01	-2.4998e-01
5	-1.3750	2.1250	-0.17186	-2.4994e-01	-5.0001e-01
6	-1.3125	2.2500	-0.21093	2.5006e-01	-1.2500e-01
7	-1.4375	2.3125	-0.23046	-1.2496e-01	-2.5002e-01
8	-1.4062	2.3750	-0.24023	1.2507e-01	-6.2489e-02
9	-1.4687	2.4062	-0.24511	-6.2441e-02	-1.2502e-01
10	-1.4531	2.4375	-0.24756	6.2530e-02	-3.1278e-02
11	-1.4844	2.4531	-0.24878	-3.1277e-02	-6.2538e-02
12	-1.4765	2.4687	-0.29390	3.1283e-02	-1.5620e-02
13	-1.4922	2.4765	-0.24969	-1.5600e-02	-3.1262e-02
14	-1.4883	2.4844	-0.24985	1.5676e-02	-7.7977e-03
15	-1.4961	2.4882	-0.24992	-7.7493e-03	-1.5633e-02
16	-1.4941	2.4922	-0.24996	7.8649e-03	-3.9051e-03
17	-1.4980	2.4941	-0.24998	-3.9196e-03	-7.8515e-03
18	-1.4971	2.4961	-0.24999	3.9571e-03	-1.9424e-03
19	-1.4990	2.4970	-0.25000	-1.9426e-03	-3.9328e-03
20	-1.4985	2.4980	-0.25000	1.9744e-03	-9.8908e-04
21	-1.4995	2.4985	-0.25000	-9.7109e-04	-1.9697e-03
22	-1.4995	2.4985	-0.25000	-9.7109e-04	-1.9697e-03
23	-1.4993	2.4990	-0.25000	1.0396e-03	-4.5807e-04
24	-1.4997	2.4992	-0.25000	-4.2714e-04	-9.8425e-04
25	-1.4996	2.4995	-0.25000	5.4816e-04	-2.3553e-04
26	-1.4998	2.4996	-0.25000	-2.0023e-04	-5.0734e-04
27	-1.4998	2.4997	-0.25000	3.0737e-04	-1.1170e-04
28	-1.4999	2.4998	-0.25000	-8.4129e-05	-2.6398e-04

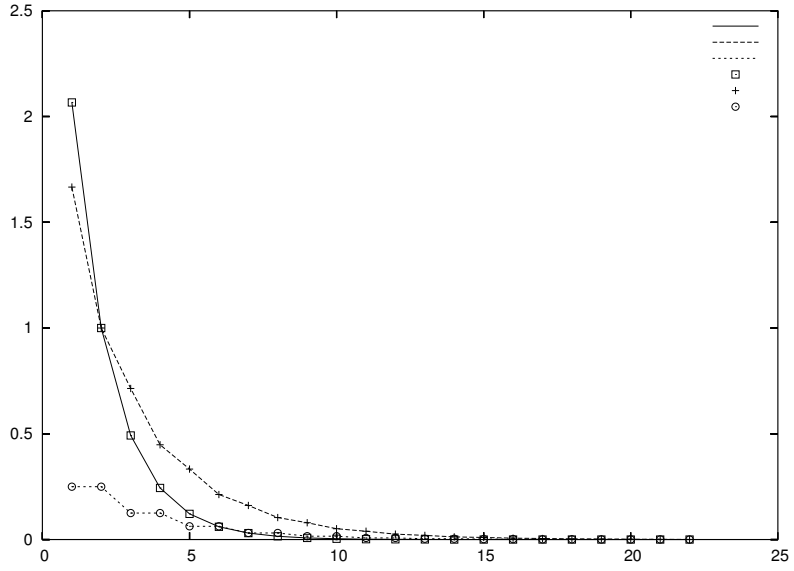


Figura 6.1: Evolução dos valores dos critérios de parada: Valor da diferença da função objetivo (□), valor da norma da diferença do vetor de variáveis de otimização (+) e valor da norma do gradiente (○). Em todos os casos, a normalização foi feita em relação ao padrão pertinente.

DEMONSTRAÇÃO: As condições para a validade do teorema da convergência global se completam quando se restringe o domínio da função à bacia de atração do ponto de mínimo. Nessa situação, a iteração do Algoritmo do Gradiente se torna descendente. As demais condições não dependem do domínio. ■

Corolário 6.1 *Caso o* Algoritmo do Gradiente *seja iniciado em um ponto x_0 não situado na bacia de atração do mínimo global x^*, podem ocorrer duas situações:*

i. O Algoritmo do Gradiente *converge para o mínimo local associado à bacia de atração em que estiver localizado seu ponto inicial x_0.*

ii. Caso o ponto inicial não esteja localizado em nenhuma bacia de atração, o Algoritmo do Gradiente *não converge.*

□

6.4 Aproximações Quadráticas

Suponha-se agora que, conhecendo-se *a priori* a natureza da função objetivo, saiba-se que é razoável admitir que essa função corresponda, de maneira aproximada, a uma função quadrática, dentro de algum domínio que contenha o ponto de mínimo x^*. A aproximação é feita ao redor de um ponto x_o, também contido nesse domínio:

$$f(x) \approx c_0 + c_1 \cdot (x - x_o) + (x - x_o)' C_2 (x - x_o) \tag{6.10}$$

sendo $c_0 \in \mathbb{R}$, $c_1 \in \mathbb{R}^n$ e $C_2 \in \mathbb{R}^{n \times n}$. Essa suposição, de fato, corresponde à suposição de que a função $f(x)$ seja de classe $\mathcal{C}^\infty$, pois toda função dessa classe pode ser escrita em termos de uma série de Taylor:

$$f(x) = f(x_o) + \nabla f(x_o)'(x - x_o) + \frac{1}{2}(x - x_o)' F(x_o)(x - x_o) + \mathcal{O}(3) \tag{6.11}$$

onde o vetor $\nabla f(x_o)$ é o gradiente da função no ponto x_o, a matriz $F(x_o)$ é a Hessiana da função em x_o, e $\mathcal{O}(3)$ é o conjunto das contribuições dos termos de ordem maior ou igual a três. O gradiente da função $f(x)$ dada por (6.11) é:

$$\nabla f(x) = \nabla f(x_o) + F(x_o)(x - x_o) \tag{6.12}$$

Sabe-se entretanto que no ponto de mínimo local x^*, o gradiente se anula, de forma que:

$$\nabla f(x^*) = \nabla f(x_o) + F(x_o)(x^* - x_o) = 0 \tag{6.13}$$

de onde se obtém a fórmula de determinação do ponto de mínimo:

$$x^* = x_o - (F(x_o))^{-1}\nabla f(x_o) \tag{6.14}$$

Ou seja, se a função a ser otimizada for exatamente quadrática, basta se conhecer o gradiente e a Hessiana em um ponto *qualquer* x_o para se determinar, em uma única iteração, o ponto de mínimo x^*, através da fórmula (6.14). Se a função for aproximadamente quadrática num certo domínio, a fórmula (6.14) pode ainda ser empregada para produzir estimativas do ponto de mínimo que convergem muito mais rapidamente que aquelas produzidas pelo Algoritmo do Gradiente.

EXEMPLO 6.4 *Considere-se a seguinte função quadrática de três variáveis reais:*

$$f(x) = x_1^2 + x_2^2 + 2x_3^2 + x_1x_2 + x_1x_3 - 7x_1 - 5x_2 - 3x_3 + 13$$

O gradiente dessa função é dado por:

$$\nabla f(x) = \begin{bmatrix} 2x_1 + x_2 + x_3 - 7 \\ x_1 + 2x_2 - 5 \\ x_1 + 4x_3 - 3 \end{bmatrix}$$

A Hessiana é dada por:

$$F(x) = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 4 \end{bmatrix}$$

O ponto de mínimo da função, x^*, no qual o gradiente se anula, é dado por:

$$x^* = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}$$

Essa função pode ser reescrita, de maneira exata, como:

$$f(x) = 3 + \frac{1}{2}\begin{bmatrix} x_1 - 3 & x_2 - 1 & x_3 \end{bmatrix}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 4 \end{bmatrix}\begin{bmatrix} x_1 - 3 \\ x_2 - 1 \\ x_3 \end{bmatrix}$$

O que se pretende mostrar neste exemplo é que, conhecendo-se o valor do gradiente e da Hessiana da função em um ponto qualquer, por exemplo $x_o = [\ 1 \quad 1 \quad 1\ ]$, *é possível utilizar tal informação para determinar a localização de* x^*, *através da fórmula (6.14). Para tal* x_o, *o valor do gradiente seria:*

$$\nabla f(x_o) = \begin{bmatrix} -3 \\ -2 \\ 2 \end{bmatrix}$$

Por (6.14):

$$x^* = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 4 \end{bmatrix}^{-1} \begin{bmatrix} -3 \\ -2 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}$$

que é o resultado esperado. ♢

EXEMPLO 6.5 *Seja a função de duas variáveis:*

$$f(x) = x_2^2(1 - \ cos(x_1)) + 1 - \ cos(x_2) + \ e^{x_1^2}$$

O gradiente dessa função é dado por:

$$\nabla f(x) = \begin{bmatrix} x_2^2 \ sen(x_1) + 2x_1 \ e^{x_1^2} \\ 2x_2(1 - \ cos(x_1)) + \ sen(x_2) \end{bmatrix}$$

A Hessiana é dada por:

$$F(x) = \begin{bmatrix} x_2^2 \ cos(x_1) + 2 \ e^{x_1^2} + 4x_1^2 \ e^{x_1^2} & 2x_2 \ sen(x_1) \\ 2x_2 \ sen(x_1) & cos(x_2) \end{bmatrix}$$

Um mínimo local dessa função ocorre para $x^* = [\ 0 \quad 0\ ]'$, *ponto para o qual o gradiente se anula. Nesse ponto, a Hessiana fica igual a:*

$$F(x^*) = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

Tome-se um ponto próximo a tal mínimo local, por exemplo $x_o = [\ 0.1 \quad 0.2\ ]'$. *Nesse ponto, o gradiente é igual a:*

$$\nabla f(x_o) = \begin{bmatrix} 0.20600 \\ 0.20067 \end{bmatrix}$$

A Hessiana é igual a:

$$F(x_o) = \begin{bmatrix} 2.100303 & 0.039933 \\ 0.039933 & 0.980067 \end{bmatrix}$$

A estimativa do ponto de ótimo, aqui denominada $\hat{x}$, fica:

$$\hat{x} = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} - \begin{bmatrix} 2.100303 & 0.039933 \\ 0.039933 & 0.980067 \end{bmatrix}^{-1} \begin{bmatrix} 0.20600 \\ 0.20067 \end{bmatrix} = \begin{bmatrix} 5.7372 \times 10^{-3} \\ -9.0822 \times 10^{-4} \end{bmatrix}$$

Embora $\hat{x}$ não seja exatamente igual a x^*, pode-se observar que a aproximação ocorreu rapidamente. Para fins de comparação, o valor da estimativa do ótimo produzida pela minimização da função na direção do gradiente, $\bar{x}$, a partir do mesmo ponto x_o, seria:

$$\bar{x} = \begin{bmatrix} -0.033814 \\ 0.069652 \end{bmatrix}$$

A aproximação quadrática, em uma iteração, produziu convergência uma ordem de grandeza mais veloz, no caso apresentado neste exemplo. ♢

6.4.1 Algoritmo de Newton

A partir da iteração definida pela equação (6.14), pode-se construir um algoritmo de minimização de funções que, em sua forma mais simples, envolve a aplicação seqüencial de (6.14) para a busca do ótimo:

Algoritmo de Newton

$k \leftarrow 0$

enquanto (não critério de parada)

$g_k \leftarrow$ gradiente$(f(\cdot), x_k)$
$F_k \leftarrow$ Hessiana$(f(\cdot), x_k)$
$x_{k+1} \leftarrow x_k - F_k^{-1} g_k$
$k \leftarrow k + 1$

fim-enquanto

Convergência

No caso da otimização de funções com forma precisamente quadrática, o Algoritmo de Newton não apenas converge para a solução exata do problema, como também o faz de maneira não-iterativa, em um único passo. Essa não é entretanto a situação geral: as funções a serem otimizadas, embora freqüentemente sejam duas vezes diferenciáveis, o que é necessário para a aplicabilidade desse método, na maioria dos casos não serão quadráticas.

Nessa última situação, o Algoritmo de Newton, na formulação apresentada, pode até mesmo não convergir. Observando os requisitos arrolados entre as hipóteses do *teorema da convergência global*, verifica-se que o Algoritmo de Newton não satisfaz à exigência de que a iteração deva ser *descendente*, ou seja, de que o valor da função objetivo necessariamente decresça a cada iteração. De fato, nada garante que o cálculo analítico da solução que seria a exata para um problema quadrático, se aplicado a um problema que não é quadrático, não venha a levar até mesmo a um aumento no valor da função objetivo.

6.4.2 Método de Newton Modificado

Para garantir que o algoritmo produza a diminuição monotônica do valor da função objetivo, mesmo para funções não-lineares que tenham comportamento significativamente diferente da função quadrática, é empregada uma variação do Algoritmo de Newton que incorpora um aspecto crucial das características de convergência do Algoritmo do Gradiente: a execução de uma minimização unidimensional em cada iteração.

Algoritmo de Newton Modificado

$k \leftarrow 0$

enquanto (não critério de parada)

$g_k \leftarrow \text{gradiente}(f(\cdot), x_k)$

$F_k \leftarrow \text{Hessiana}(f(\cdot), x_k)$

$d_k \leftarrow -F_k^{-1} g_k$

$\alpha_k \leftarrow \arg\ \min_\alpha f(x_k + \alpha d_k)$

$x_{k+1} \leftarrow x_k + \alpha_k d_k$
$k \leftarrow k + 1$

fim-enquanto

Com exceção da rotina de cálculo da Hessiana, todas as subrotinas envolvidas na construção desses algoritmos já foram apresentadas por ocasião da construção do Algoritmo do Gradiente, e são reaproveitadas aqui.

Convergência

O algoritmo modificado é exatamente equivalente ao Algoritmo de Newton original, no sentido de que ambos produzem a mesma seqüência de pontos, caso a função a ser otimizada seja exatamente quadrática. Agora, no entanto, há a garantia de decrescimento monotônico da função objetivo a cada iteração, qualquer que seja a estrutura da função objetivo. Dessa forma, garante-se o atendimento de todos os requisitos do teorema da convergência global num sentido similar ao da convergência do Algoritmo do Gradiente. Agora, para estabelecer a completa equivalência da região de convergência do Algoritmo de Newton Modificado com a do Algoritmo do Gradiente, basta mostrar que o primeiro é bem definido na mesma região de convergência do último, ou seja, na bacia de atração. Isto é assegurado pela proposição a seguir.

Proposição 6.2 *Seja $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ uma função contínua infinitas vezes diferenciável. Seja x^* um mínimo local estrito dessa função. Sob tais condições, a Hessiana de $f(\cdot)$ é definida positiva na bacia de atração de x^*.* □

Há a necessidade de diferenciabilidade infinita de $f(\cdot)$ neste caso, ao contrário das funções otimizadas com o algoritmo do gradiente, que precisam apenas ser diferenciáveis uma vez. Isso decorre da possibilidade que haveria, se não se colocasse tal exigência, de se concatenar trechos de hiperplanos por meio de funções suaves, que podem ser diferenciáveis até alguma ordem finita, formando bacias de atração suaves nas quais a Hessiana é nula em diversos trechos. O método de Newton simplesmente não seria definido para tais funções. O mínimo local, agora, ainda deve ser estrito, pois do contrário

a Hessiana poderia ter posto incompleto, também invalidando a iteração de Newton.

Definidas essas exigências para a aplicabilidade do método, é possível estabelecer a região de convergência.

Proposição 6.3 *Seja o problema de otimização irrestrito definido por:*

$$x^* = \arg \min_x f(x) \tag{6.15}$$

sendo $x \in \mathbb{R}^n$*, com* $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ *uma função contínua infinitas vezes diferenciável, e* x^* *um mínimo estrito. Então o* Algoritmo de Newton Modificado *irá convergir para* x^* *para todo ponto inicial* x_0 *situado na bacia de atração de* x^**.* □

Corolário 6.2 *Garantidas as condições da proposição 6.3, caso o* Algoritmo de Newton Modificado *seja iniciado em um ponto* x_0 *não situado na bacia de atração do mínimo global* x^**, podem ocorrer três situações:*

i. *O* Algoritmo de Newton Modificado *converge para o mínimo local estrito associado à bacia de atração em que estiver localizado seu ponto inicial* x_0*.*

ii. *Caso o ponto inicial esteja localizado em uma bacia de atração de um mínimo local não estrito, o Algoritmo de Newton Modificado pode ficar indefinido (ou seja, a Hessiana pode ser não inversível). Se isso não ocorrer, ocorrerá convergência para o mínimo local.*

iii. *Caso o ponto inicial não esteja localizado em nenhuma bacia de atração, o Algoritmo de Newton Modificado não converge, podendo ainda ficar indefinido.*

□

NOTA 6.5 *O leitor deve estar ciente de que existem procedimentos* ad-hoc *para evitar que a "Hessiana" utilizada pelo algoritmo fique não inversível, ao custo da perda de sua fidelidade para representar a verdadeira Hessiana da função (porém garantindo as propriedades de convergência do algoritmo). Para maiores informações, ver (Luenberger 1984).*

♢

6.4.3 Determinação Numérica da Hessiana

Para a implementação do método de Newton é necessário o cálculo numérico da Hessiana. Por meio de um hipotético método de diferenças finitas, seria necessário avaliar o *gradiente* da função objetivo em $n+1$ pontos, no caso de uma função de n variáveis. Sendo $g(x)$ o gradiente da função objetivo, avaliado numericamente por meio de diferenças finitas, como já visto, o método de cálculo da Hessiana por diferenças finitas poderia ser formulado como:

Cálculo da Hessiana por Diferenças Finitas

para $i \leftarrow 1$ até n faça

$$F_i \leftarrow \frac{g(x + \delta e_i) - g(x)}{\delta}$$

fim-para

$$F \leftarrow \left[\begin{array}{ccc} F_1 & \ldots & F_n \end{array} \right]$$

Cada uma das avaliações de gradiente por sua vez envolve, como já se viu, a avaliação da função objetivo em $n+1$ pontos, de forma que o número total de avaliações da função objetivo seria igual a $(n+1)^2$.

6.4.4 Construção da Hessiana

Examine-se novamente a equação (6.12), reproduzida abaixo por conveniência:

$$\nabla f(x) = \nabla f(x_o) + F(x_o)(x - x_o) \tag{6.16}$$

Essa equação foi o ponto de partida para a construção do método de Newton. Ela pode também ser usada para construir um método para estimar a própria Hessiana da função. Reescrevendo a equação, para dois pontos x_1 e x_2 (tomar cuidado: o índice subscrito não se refere, aqui, às coordenadas de um vetor, mas a vetores diferentes), e supondo que a Hessiana seja constante em todo o espaço:

$$F(x_1 - x_2) = \nabla f(x_1) - \nabla f(x_2) \tag{6.17}$$

Essa mesma fórmula pode ser repetida para a seguinte seqüência de vetores:

$$\begin{array}{l} F(x_1 - x_2) = \nabla f(x_1) - \nabla f(x_2) \\ F(x_2 - x_3) = \nabla f(x_2) - \nabla f(x_3) \\ \vdots \\ F(x_{n-1} - x_n) = \nabla f(x_{n-1}) - \nabla f(x_n) \\ F(x_n - x_{n+1}) = \nabla f(x_n) - \nabla f(x_{n+1}) \end{array} \tag{6.18}$$

Definindo os vetores v_i e r_i como:

$$\begin{array}{l} v_i = x_i - x_{i+1} \\ \\ r_i = \nabla f(x_i) - \nabla f(x_{i+1}) \end{array} \tag{6.19}$$

tem-se que:

$$F \left[\begin{array}{cccc} v_1 & v_2 & \dots & v_n \end{array} \right] = \left[\begin{array}{cccc} r_1 & r_2 & \dots & r_n \end{array} \right] \tag{6.20}$$

Definindo $V = \left[\begin{array}{cccc} v_1 & v_2 & \dots & v_n \end{array} \right]$ e $R = \left[\begin{array}{cccc} r_1 & r_2 & \dots & r_n \end{array} \right]$:

$$FV = R \tag{6.21}$$

Observando agora que os vetores v_i tratam-se de *escolhas*, nota-se que é possível escolhê-los de tal forma que V seja inversível, o que permite fazer:

$$F = V^{-1}R \tag{6.22}$$

Isso significa que, avaliando o gradiente da função $f(x)$ em $n + 1$ pontos adequadamente escolhidos do espaço, é possível determinar a Hessiana dessa função.

NOTA 6.6 *Examinando-se o resultado obtido, verifica-se que a equação (6.21) é uma generalização do* cálculo da Hessiana por diferenças finitas. *De fato, fazendo-se $V = \delta I$ tem-se de (6.21) que $F = \frac{1}{\delta}R$.*

♢

NOTA 6.7 *Da mesma forma como o cálculo do gradiente por diferenças finitas é exato para funções polinomiais de grau 1, o cálculo da Hessiana por (6.22) é exato para funções polinomiais de grau 2 (desde que se disponha, no entanto, de avaliações exatas do gradiente). Pelo mesmo motivo que no caso da avaliação exata do gradiente em funções lineares, caso a função seja quadrática, não é necessário que os pontos em que se avalia o gradiente estejam próximos entre si para que o cálculo da Hessiana permaneça exato.*

♢

Diversos métodos de otimização baseiam-se na equação (6.22), variando-se, de método para método, a escolha dos pontos (o que implica na variação da escolha de V).

6.4.5 Correção de Posto 1

Conforme foi visto, há certa arbitrariedade na escolha dos vetores v_i (a única condição necessária é de que sejam n vetores linearmente independentes). Dessa forma, é possível acrescentar restrições ao problema de forma a obter fórmulas recursivas particularmente interessantes.

A idéia a ser explorada aqui é a de que deve ser possível fazer a construção recursiva da estimativa da Hessiana (ou de sua inversa), durante o decorrer de um processo de otimização. A estimativa parcial da Hessiana deve poder ser utilizada no decorrer desse processo. Isso é particularmente útil na otimização de funções não-quadráticas, em que a Hessiana não é constante: esse procedimento permite a adaptação contínua da estimativa da Hessiana ao seu valor localmente válido.

É mostrado inicialmente o algoritmo mais simples possível para realizar o procedimento pretendido, que será aqui denominado *Algoritmo de Correção de Posto 1*.

Seja $H_k = F_k^{-1}$. A idéia é construir um método recursivo que produza uma seqüência de estimativas $[H_k]$, a partir de novas avaliações da função e de seu gradiente em novos pontos. Observa-se inicialmente que a Hessiana de toda função é simétrica, de forma que a recursão deve gerar uma matriz simétrica. A recursão proposta é da forma:

$$H_{k+1} = H_k + \alpha_k z_k z_k' \tag{6.23}$$

sendo $z_k \in \mathbb{R}^n$ e $\alpha_k \in \mathbb{R}$. Claramente, o termo $\alpha_k z_k z_k'$ é uma matriz $n \times n$ com posto no máximo igual a 1, de onde vem o nome do algoritmo. Supondo, para fins de desenvolvimento da fórmula de recursão, que a função objetivo fosse exatamente quadrática, é preciso definir α_k e z_k em função dos valores conhecidos (os vetores $[x_k]$ e $[\nabla f(x_k)]$), de forma a garantir que seja satisfeita a relação:

$$H_{k+1} r_i = v_i \quad \forall\ i = 1, \ldots, k \tag{6.24}$$

Essa relação é quase a mesma que (6.21), mas exige a igualdade apenas para os pontos já avaliados (até o índice k). Em primeiro lugar, desenvolve-se a

fórmula para $i = k$. Substituindo-se (6.23) em (6.24), obtém-se:

$$\begin{aligned} &\alpha_k z_k z_k' r_k = v_k - H_k r_k \\ &\therefore (v_k - H_k r_k)(v_k - H_k r_k)' = \alpha_k z_k z_k' r_k \alpha_k r_k' z_k z_k' \\ &\therefore (v_k - H_k r_k)(v_k - H_k r_k)' = \alpha_k (z_k' r_k)^2 \alpha_k z_k z_k' \end{aligned} \tag{6.25}$$

Com isso, quase se tem uma fórmula para o termo de correção $\alpha_k z_k z_k'$ em função de H_k, v_k e r_k, a menos da quantidade escalar $\alpha_k (z_k' r_k)^2$. Para se determinar essa constante, faz-se:

$$\begin{aligned} &r_k' \alpha_k z_k z_k' r_k = r_k' (v_k - H_k r_k) \\ &\therefore \alpha (z_k' r_k)^2 = r_k' v_k - r_k' H_k r_k \end{aligned} \tag{6.26}$$

Substituindo-se (6.26) em (6.25) obtém-se:

$$\alpha_k z_k z_k' = \frac{1}{r_k' v_k - r_k' H_k r_k} (v_k - H_k r_k)(v_k - H_k r_k)' \tag{6.27}$$

ou, voltando agora à fórmula recursiva para cálculo de H_{k+1}:

$$H_{k+1} = H_k + \frac{1}{r_k' v_k - r_k' H_k r_k} (v_k - H_k r_k)(v_k - H_k r_k)' \tag{6.28}$$

Essa fórmula, por construção, vale para $i = k$. Resta provar que ela é válida para $i < k$.

Teorema 6.3 *Seja F uma matriz simétrica fixa, e suponha-se que $v_0, v_1, \ldots, v_k$ sejam vetores dados. Definam-se os vetores $r_i = F v_i$, para $i = 0, 1, \ldots, k$. Seja ainda H_0 uma matriz simétrica qualquer. Se:*

$$H_{i+1} = H_i + \frac{1}{r_i' v_i - r_i' H_i r_i} (v_i - H_i r_i)(v_i - H_i r_i)' \tag{6.29}$$

então:

$$v_i = H_{k+1} r_i \quad \forall\ i = 1, \ldots, k \tag{6.30}$$

□

DEMONSTRAÇÃO: Por construção, a relação é válida para $i = k$. Tome-se algum r_i para $i < k$, e aplique-se esse vetor em H_{k+1}:

$$H_{k+1}r_i = H_k r_i + \frac{1}{r'_k v_k - r'_k H_k r_k}(v_k - H_k r_k)(v'_k r_i - r'_k H'_k r_i)$$

Note-se que H_k é simétrica, de forma que:

$$H_{k+1}r_i = H_k r_i + \frac{1}{r'_k v_k - r'_k H_k r_k}(v_k - H_k r_k)(v'_k r_i - r'_k H_k r_i)$$

Adota-se neste ponto, como hipótese de indução, que:

$$v_i = H_k r_i$$

seja verdade. Isso implica que:

$$H_{k+1}r_i = v_i + \frac{1}{r'_k v_k - r'_k H_k r_k}(v_k - H_k r_k)(v'_k r_i - r'_k v_i)$$

Entretanto:

$$r'_k v_i = v'_k F' v_i = v'_k F v_i = v'_k r_i$$

de forma que:

$$v'_k r_i - r'_k v_i = 0$$

ou:

$$H_{k+1}r_i = v_i$$

Isso completa a prova. ■

Sabe-se então que, usando-se a fórmula (6.28), obtém-se o valor exato da inversa da Hessiana de uma função quadrática, a partir de $n + 1$ valores de pontos do espaço com as respectivas avaliações de gradientes da função nesses pontos.

Algoritmo de Correção de Posto 1

Com esse resultado, é possível construir um algoritmo de otimização utilizando a estrutura básica da "direção de busca", tomando H_k como aproximação da inversa da Hessiana. O algoritmo se inicia em um ponto x_0 qualquer:

Algoritmo de Correção de Posto 1

$k \leftarrow 0$

$H_k \leftarrow I$

$g_k \leftarrow \text{gradiente}(f(\cdot), x_k)$

enquanto (não critério de parada)

$d_k \leftarrow -H_k g_k$

$\alpha_k \leftarrow \arg\ \min_\alpha f(x_k + \alpha d_k)$

$x_{k+1} \leftarrow x_k + \alpha_k d_k$

$g_{k+1} \leftarrow \text{gradiente}(f(\cdot), x_{k+1})$

$v_k \leftarrow x_k - x_{k+1}$

$r_k \leftarrow g_k - g_{k+1}$

$H_{k+1} \leftarrow H_k + \frac{1}{r_k' v_k - r_k' H_k r_k}(v_k - H_k r_k)(v_k - H_k r_k)'$

$k \leftarrow k + 1$

fim-enquanto

Deve-se notar que, de maneira arbitrária, a estimativa H_0 foi inicializada com a matriz identidade. Qualquer outra matriz simétrica poderia ter sido utilizada, de acordo com o teorema 6.3. Esse teorema, juntamente com o resultado anteriormente conhecido a respeito de aproximações quadráticas em geral, afirma que se a função objetivo for quadrática, a convergência exata do algoritmo para o mínimo global da função necessariamente ocorrerá, e o número de passos para tal convergência será menor ou igual a n. Note-se que, ao invés de serem tomados pontos quaiquer que gerem vetores v_i linearmente independentes, estão sendo tomados exatamente aqueles pontos gerados pelo processo de otimização. Estes *geram* vetores v_i linearmente independentes necessariamente, caso a função seja exatamente quadrática.

Sob o ponto de vista da otimização de uma função a priori sabida ser quadrática, não há vantagem computacional em se utilizar o *Algoritmo de Correção de Posto 1* em lugar da fórmula exata (6.14) junto com (6.22). A aplicação destas envolveria exatamente $n + 1$ avaliações de gradiente, enquanto a aplicação do algoritmo de correção envolveria um número menor

ou igual a este de iterações, cada uma envolvendo uma avaliação de gradiente, mas envolvendo também uma otimização unidimensional. Esta última poderia tornar o algoritmo de correção mais caro sob o ponto de vista computacional.

No entanto, sabe-se que no caso geral da otimização de funções não-lineares não necessariamente quadráticas, a Hessiana da função objetivo não será em geral constante. Não ocorrerá, de qualquer forma, a convergência em n iterações. O Algoritmo de Correção de Posto 1 torna-se nesse caso vantajoso, pois a estimativa da Hessiana vai mudando dinamicamente, de forma a acompanhar a variação dessa Hessiana. A cada passo, uma nova estimativa da Hessiana está disponível, para ser utilizada no processo de otimização. Essas são características gerais da categoria de métodos conhecidos como quasi-Newton, que será vista a seguir.

NOTA 6.8 *Deve-se notar que a primeira iteração do* Algoritmo de Correção de Posto 1*, no formato acima definido, corresponde exatamente a uma iteração do* Algoritmo do Gradiente*. Isso ocorre porque, com a matriz H_0 sendo inicializada igual à identidade, no primeiro passo a direção de busca fica sendo igual à do gradiente. A partir do segundo passo, a direção começa a mudar gradativamente, até que no n-ésimo passo a direção passa a coincidir com a do* Algoritmo de Newton*, caso a Hessiana seja constante. Caso a Hessiana não seja constante, a estimativa do* Algoritmo de Correção de Posto 1 *será sempre inexata, e este algoritmo não chegará a convergir para o comportamento hipotético do* Algoritmo de Newton*. No entanto, como a avaliação direta da Hessiana, exigida pelo* Algoritmo de Newton*, é inconveniente, na prática usualmente se opta pela utilização de algoritmos quasi-Newton.*

♢

Convergência do Algoritmo de Correção de Posto 1

O *Algoritmo de Correção de Posto 1* possui propriedades de convergência que são intermediárias entre as do Algoritmo do Gradiente e as do Algoritmo de Newton. Este último simplesmente não se aplica quando a Hessiana não é definida positiva. Já o Algoritmo do Gradiente exige apenas a existência de uma bacia de atração de uma função diferenciável. O Algoritmo de Correção de Posto 1 não pode ficar indefinido em nenhum ponto, uma vez que não envolve inversões de matrizes. No entanto, sua formulação permite que a matriz H_{k+1} venha a eventualmente perder a propriedade de ser positiva

definida, caso ocorra:

$$r'_k v_k - r'_k H_k r_k < 0 \tag{6.31}$$

Não há nada que impeça essa condição de ocorrer. Isso significa que a fórmula de correção pode eventualmente vir a ficar comprometida. Isso pode fazer com que o algoritmo fique estacionado em pontos que não correspondem à solução do problema. Pode-se evitar tal situação incluindo-se uma verificação dos autovalores de H_{k+1} a cada passo, fazendo-se a substituição dessa matriz pela identidade sempre que for detectado um autovalor negativo. Isso iria restaurar as condições do *teorema da convergência global*, e o algoritmo passaria a convergir exatamente na mesma região que o Algoritmo do Gradiente.

6.4.6 Métodos Quasi-Newton

Os métodos de otimização conhecidos como *quasi-Newton* são desenvolvidos de acordo com a mesma lógica que foi usada na elaboração do *Algoritmo de Correção de Posto 1* (de fato, este algoritmo é o exemplo mais simples de um algoritmo quasi-Newton). Constrói-se uma regra recursiva que permite a construção gradativa de uma matriz H_k que corresponde a uma estimativa da inversa da Hessiana da função objetivo. Como deve ter sido observado na seção anterior, diversas escolhas arbitrárias de regras foram realizadas, de forma que outras escolhas teriam sido possíveis para garantir as propriedades desejadas de H_k. Com os graus de liberdade ainda remanescentes, podem-se produzir métodos que evitem as dificuldades de convergência do *Algoritmo de Correção de Posto 1*: essencialmente, deve-se garantir que a matriz H_k permaneça sempre definida positiva, e preferencialmente bem condicionada (ou seja, com autovalores não muito distanciados entre si).

Dois métodos particularmente eficientes foram desenvolvidos para produzir estimativas recursivas para H_k com as propriedades requeridas: o método DFP (Davidon-Fletcher-Powell) e o método BFGS (Broyden-Fletcher-Goldfarb-Shanno), assim batizados em homenagem aos seus formuladores. Verificando-se, *a posteriori*, as conexões entre esses métodos, estes foram agrupados em uma estrutura mais geral, a *família de Broyden*. Esses métodos são apresentados a seguir.

Método DFP

A correção proposta pelo método DFP é dada por:

$$C_k^{DFP} = \frac{v_k v_k'}{v_k' r_k} - \frac{H_k r_k r_k' H_k}{r_k' H_k r_k} \tag{6.32}$$

Método BFGS

A correção proposta pelo método BFGS é dada por:

$$C_k^{BFGS} = \left(1 + \frac{r_k' H_k r_k}{r_k' v_k}\right) \frac{v_k v_k'}{v_k' r_k} - \frac{v_k r_k' H_k + H_k r_k v_k'}{r_k' v_k} \tag{6.33}$$

Família de Broyden

A correção genérica utilizada pelos métodos conhecidos como *família de Broyden* é dada por:

$$C_k = (1 - \alpha) C_k^{DFP} + \alpha C_k^{BFGS} \tag{6.34}$$

Em todos os casos da família de Broyden, incluindo os casos extremos BFGS e DFP, a fórmula de atualização para a estimativa da inversa da Hessiana fica:

$$H_{k+1} = H_k + C_k(\alpha) \tag{6.35}$$

Para $\alpha = 0$, obtém-se o método DFP, e para $\alpha = 1$ o método BFGS.

Alguns fatos devem ser citados a respeito dessa correção da família de Broyden (embora não sejam apresentadas aqui as respectivas provas):

- A correção realizada a cada passo é de posto possivelmente dois (isso é facilmente verificável por inspeção).
- A correção é sempre definida positiva, de forma que a matriz H_k preservará sua propriedade de ser definida positiva. A prova disto é apresentada em (Luenberger 1984).
- Dados i e j tais que $0 \leq i < j \leq k$, então $v_i' F v_j = 0$, ou seja, v_i e v_j são F-ortogonais. Ver a prova em (Luenberger 1984).
- Dado i tal que $0 \leq i \leq k$, então $H_{k+1} F v_i = v_i$. Ver a prova em (Luenberger 1984).

Algoritmos Quasi-Newton

Os algoritmos obtidos a partir da família de Broyden, aqui denominados *Algoritmos Quasi-Newton*, são estruturados da seguinte forma, dados um x_0 e um α:

Algoritmo de Quasi-Newton

$k \leftarrow 0$

$H_k \leftarrow I$

$g_k \leftarrow \text{gradiente}(f(\cdot), x_k)$

enquanto (não critério de parada)

$d_k \leftarrow -H_k g_k$

$\alpha_k \leftarrow \arg\ \min_\alpha f(x_k + \alpha d_k)$

$x_{k+1} \leftarrow x_k + \alpha_k d_k$

$g_{k+1} \leftarrow \text{gradiente}(f(\cdot), x_{k+1})$

$v_k \leftarrow x_k - x_{k+1}$

$r_k \leftarrow g_k - g_{k+1}$

$C_k^{DFP} \leftarrow \frac{v_k v_k'}{v_k' r_k} - \frac{H_k r_k r_k' H_k}{r_k' H_k r_k}$

$C_k^{BFGS} \leftarrow \left(1 + \frac{r_k' H_k r_k}{r_k' v_k}\right) \frac{v_k v_k'}{v_k' r_k} - \frac{v_k r_k' H_k + H_k r_k v_k'}{r_k' v_k}$

$C_k \leftarrow (1-\alpha) C_k^{DFP} + \alpha C_k^{BFGS}$

$H_{k+1} \leftarrow H_k + C_k(\alpha)$

$k \leftarrow k + 1$

fim-enquanto

Evidentemente, para a implementação pura do Algoritmo DFP ou do Algoritmo BFGS, não haveria necessidade do cálculo intermediário de C_k, sendo possível simplificar o programa, para o cálculo direto de H_k com a correção correspondente.

Convergência da Família de Broyden

A maneira mais fácil de provar a convergência dos algoritmos da família de Broyden seria introduzindo uma modificação nos mesmos: se se faz com que a matriz H_k seja periodicamente reinicializada, sendo igualada à identidade, torna-se possível a aplicação direta do teorema da convergência global. Os algoritmos passam a convergir exatamente como o Algoritmo do Gradiente.

É possível, sem introduzir tal modificação, ainda assim provar a convergência dos algoritmos, sendo necessárias entretanto algumas condições adicionais sobre a função a ser otimizada. Para maiores informações, ver (Luenberger 1984).

6.5 Tratamento de Restrições

Considere-se agora o problema restrito:

$$\begin{aligned} x^* = & \arg \min_x f(x) \\ & \text{sujeito a: } \{g(x) \leq 0 \end{aligned} \tag{6.36}$$

sendo $x \in \mathbb{R}^n$, $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ e $g(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^p$.

Pretende-se resolver este problema utilizando como mecanismo de otimização um algoritmo de otimização irrestrita do tipo "direção de busca". Para isso, são definidos os métodos de *barreira* e os métodos de *penalidades*, que transformam o problema restrito em problemas irrestritos aproximadamente (ou assintoticamente) equivalentes. A discussão apresentada a seguir pode ser vista, em maior detalhe, em (Luenberger 1984).

6.5.1 Método de Barreira

A forma geral dos métodos de barreira é dada pelo problema irrestrito, que é aproximadamente equivalente a (6.36):

$$x^* = \arg \min_{x,\gamma} F(x, \gamma) \tag{6.37}$$

sendo $F(x, \gamma)$ uma função com as propriedades:

- $\lim_{g(x)\to 0^-} F(x, \gamma) = +\infty$

- $F(x,\gamma) \approx f(x)\ \forall\ g(x) < -\epsilon$
- $\lim_{\gamma \to 0^+} F(x,\gamma) = f(x)\ ,\ \forall\ g(x) < -\epsilon$

Supõem-se que $\epsilon > 0$ e $\gamma > 0$ são escalares pequenos. Necessariamente, deve-se ter $g(x) < 0$ para que a "função barreira" esteja definida.

Um exemplo de função barreira que pode ser construída é:

$$\begin{aligned} x^* &= \arg \min_{x,\alpha} F(x,\alpha) \\ F(x,\alpha) &= f(x) + \sum_{i=1}^{p} G_i(x,\alpha_i) \\ G_i(x,\alpha_i) &= -\frac{\alpha_i}{g_i(x)} \\ \alpha &= \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_p \end{bmatrix}' \end{aligned} \tag{6.38}$$

sendo $1 \gg \alpha_i > 0$. A solução desse problema, evidentemente, tem de se aproximar da solução do problema original. A função $G(\cdot)$ é uma função do tipo "barreira", que possui a propriedade de se aproximar de infinito para pontos factíveis próximos da fronteira da região factível. Dessa forma, a solução do problema (6.38) estará sempre estritamente dentro da região factível definida para o problema (6.36).

6.5.2 Método de Penalidades

Métodos de penalidades também alteram o problema original (6.36) para um formato irrestrito assintoticamente equivalente:

$$x^* = \arg \min_{x,\gamma}\ F(x,\gamma) \tag{6.39}$$

sendo $F(x,\gamma)$ uma função com as propriedades:

- $F(x,\gamma) >> f(x)\ \forall\ g(x) > 0$
- $(g(x_1) - g(x_2) > 0; g(x_2) > 0) \Rightarrow F(x_1,\gamma) > F(x_2,\gamma)$
- $F(x,\gamma) \approx f(x)\ \forall\ g(x) < \epsilon$

- $\lim_{\gamma \to 0^+} F(x, \gamma) = f(x)$

Novamente, $\epsilon > 0$ e $\gamma > 0$ são escalares pequenos.

Empregando um possível método de penalidade, obtém-se o seguinte formato irrestrito:

$$
\begin{aligned}
x^* = & \ \arg \min_{x,\alpha} F(x, \alpha) \\
& \\
F(x, \alpha) = & \ f(x) + \sum_{i=1}^{p} G_i(x, \alpha_i) \\
& \\
G_i(x, \alpha_i) = & \begin{cases} g_i(x) \quad , & \text{se } g_i(x) \leq 0 \\ \\ \alpha_i (g_i(x) + g_i(x)^2) \quad , & \text{se } g_i(x) > 0 \end{cases} \\
& \\
\alpha = & \left[\begin{array}{cccc} \alpha_1 & \alpha_2 & \cdots & \alpha_p \end{array} \right]'
\end{aligned}
\tag{6.40}
$$

sendo também neste caso $1 \gg \alpha_i > 0$. Neste caso, a função $G(\cdot)$, que é uma função de "penalidade", tem a propriedade de assumir valores elevados para pontos que violem fortemente as restrições, e valores nulos para pontos que não violem as restrições. Diferentemente do método de barreira, o ponto-solução a cada iteração pode, neste caso, caminhar dentro ou fora da região factível, sendo que a solução de (6.40) pode eventualmente estar fora da região factível do problema original (6.36). Observe-se ainda que, na maioria das vezes, a função de "penalidade" será não-diferenciável, como no exemplo acima.

Computacionalmente, os problemas (6.38) e (6.40) têm a vantagem de serem irrestritos, de forma que é possível aplicar a eles métodos irrestritos de otimização. A solução é produzida fixando-se valores para o vetor α e calculando soluções em x. Heuristicamente, adota-se o mínimo de uma seqüência de soluções, para α's gradativamente maiores, como a solução do problema restrito original (6.36).

NOTA 6.9 *Observe-se que, caso o mínimo do problema restrito se encontre na fronteira da região factível, existe uma tendência para que a sequência de soluções do método da barreira aproximar esse mínimo, sempre pelo lado de dentro da região factível, enquanto que a sequência gerada por aplicações sucessivas do método da penalidade tende a aproximar o mínimo pelo lado de fora da região factível.*

♢

6.6 Comportamento dos Métodos de Direção de Busca

Nesta seção, alguns aspectos a respeito de como os métodos de direções de busca se comportam diante de funções objetivo com diferentes características são discutidos.

6.6.1 Não-Diferenciabilidade

A existência de não-diferenciabilidades nas funções pode causar problemas para a execução de métodos baseados em direções de busca. Esses problemas seriam de duas naturezas.

Primeiro, o cálculo do gradiente, do qual normalmente depende a execução do método, poderá ser inviabilizado em algumas regiões do espaço de parâmetros. Tal dificuldade, entretanto, em princípio pode ser contornada com a simples substituição do ponto de singularidade por outro ponto não-singular arbitrariamente próximo.

No entanto, uma dificuldade maior reside na possibilidade de haver descontinuidades da função gradiente que "atraiam" as trajetórias da seqüência de estimativas. Nesse caso, uma vez que uma trajetória caia em uma dessas descontinuidades "atratoras", o método pode ter problemas de convergência, uma vez que a direção de busca pode estar apontando para a superfície numa inclinação tal que o próximo mínimo unidimensional fique muito próximo do anterior. O efeito é muito similar ao do mal-condicionamento numérico de uma função diferenciável.

Essa situação é ilustrada na Figura 6.2. A superfície de descontinuidade do gradiente, $\mathcal{G}$, possui dimensão $n-1$, dividindo o espaço em duas regiões disjuntas. O vetor "menos gradiente" da função em todos os pontos situados próximos a $\mathcal{G}$ aponta para esta superfície, independente do lado em que o ponto se situa. Essa superfície irá tender a "atrair" as trajetórias de busca.

EXEMPLO 6.6 *Considere-se a função* $f(x) : \mathbb{R}^2 \mapsto \mathbb{R}$ *definida por:*

$$f(x) = \max\{f_1(x), f_2(x)\}$$

$$f_1(x) = (x - c_1)'Q(x - c_1)$$

$$f_2(x) = (x - c_2)'Q(x - c_2)$$

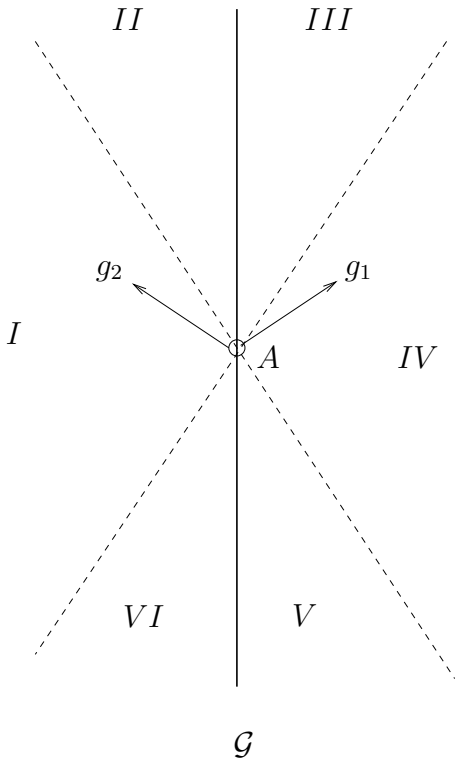


Figura 6.2: Representação de uma descontinuidade "atratora". A superfície $\mathcal{G}$ possui dimensão $n-1$, dividindo o espaço em duas regiões disjuntas. Tome-se um ponto A situado sobre a superfície de descontinuidade $\mathcal{G}$. Infinitesimalmente à esquerda de A, o gradiente da função objetivo é g_2. Infinitesimalmente à direita do mesmo ponto, o gradiente é g_1. Esses vetores gradiente definem hiperplanos normais que dividem o espaço em regiões para as quais a função objetivo aumenta (no sentido dos gradientes) e regiões para as quais a função objetivo diminui (sentido contrário ao do gradiente). Pela figura pode-se notar que: (i) a função objetivo diminui apenas na direção do cone formado pelas regiões V e VI da figura. Direções de busca baseadas apenas no gradiente g_1 ou g_2, entretanto, iriam indicar a busca dentro dos cones I e IV, respectivamente, deixando o método "estacionado".

sendo:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

e

$$c_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad c_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

Um mapa de curvas de nível da função $f(x)$ pode ser visto na Figura 6.3. Pode-se perceber claramente que a função é não diferenciável, e a região de não-diferenciabilidade corresponde à linha $x_1 = 0$.

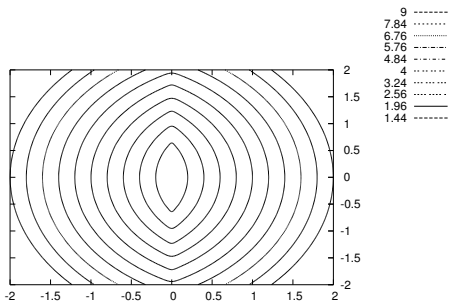


Figura 6.3: Curvas de nível da função $f(x)$ do Exemplo 6.6.

O mínimo desta função ocorre para $x^* = [\ 0 \quad 0\ ]'$. O Algoritmo do Gradiente, aplicado à otimização da função, com ponto inicial $x_o = [\ 1 \quad 2\ ]$, pára no ponto: $x = [\ 0 \quad 0.359\ ]'$. Já o Algoritmo Quasi-Newton DFP converge para o valor correto do mínimo da função. ♢

EXEMPLO 6.7 *Considere-se agora a função $f(x) : \mathbb{R}^2 \mapsto \mathbb{R}$ definida por:*

$$f(x) = \max\{|x_1|, |x_2|\}$$

Um diagrama de curvas de nível desta função é mostrado na Figura 6.4. A região de não-diferenciabilidade agora é formada pelas retas $x_1 = x_2$ e $x_1 = -x_2$.

Após 6 iterações, iniciando do ponto $x_o = [\ 1 \quad 1.01\ ]$, o Algoritmo do Gradiente vai para o ponto $x = [\ -0.0477 \quad 0.0477\ ]$, onde fica estacionado indefinidamente.

Neste exemplo, diferentemente do anterior, o Algoritmo Quasi-Newton DFP não é aplicável, uma vez que as avaliações de gradientes produzidas são linearmente dependentes, o que impede a construção da Hessiana. ♢

6.6.2 Não-Convexidade

Caso uma função a ser otimizada seja não-convexa, sendo garantidas entretanto a unimodalidade e a diferenciabilidade da mesma, os algoritmos de

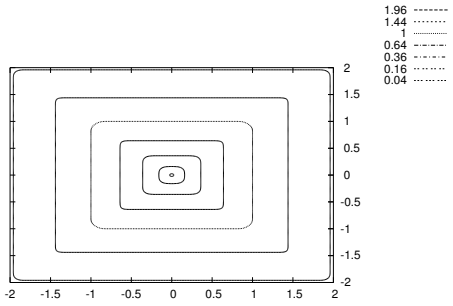


Figura 6.4: Curvas de nível da função $f(x)$, do Exemplo 6.7.

direções de busca deverão convergir (exceto por problemas de condicionamento numérico) para o ponto de ótimo da função.

6.6.3 Multimodalidade

Excluindo-se agora a unimodalidade dos funcionais, sendo mantida sua diferenciabilidade, os algoritmos da categoria de "direções de busca" ainda irão convergir para mínimos desses funcionais, mas agora possivelmente mínimos apenas locais. Esses algoritmos não estão munidos de nenhum instrumental para procurar mínimos globais em funções multimodais.

Capítulo 7

Métodos de Exclusão de Semi-Espaços

Os métodos aqui denominados "de exclusão de semi-espaços" são aqueles que empregam a propriedade dos subgradientes de funcionais de definir um plano que divide o espaço em dois semi-espaços, sendo que o funcional necessariamente decresce em um desses semi-espaços. Por trabalhar com subgradientes, esses métodos admitem funcionais não diferenciáveis. Dentro dessa categoria, encontram-se os diversos métodos de "planos de corte", e também o método "elipsoidal" com suas variantes. A fórmula básica que caracteriza a evolução desses métodos é:

Métodos de Exclusão de Semi-Espaços

Passo 1: Calcula-se o subgradiente do funcional a ser minimizado em um ponto.

Passo 2: Com esse subgradiente, particiona-se uma região de busca em duas novas regiões, sendo excluída uma delas.

Passo 3: Nessa região, procura-se uma nova estimativa do ponto de mínimo. Caso esta não seja ainda adequada, retorna-se ao primeiro passo.

Os primeiros algoritmos da família de "exclusão de semi-espaços" a serem propostos foram os chamados *algoritmos de planos de corte*. O capítulo 13 da referência (Luenberger 1984) trata desses métodos.

Em 1979, foi proposto um novo método que se enquadrava na categoria de "exclusão de semi-espaços": o *algoritmo elipsoidal*. Esse novo método veio a possuir grande importância teórica: foi o primeiro método a partir do qual se demonstrava a resolução em tempo polinomial dos problemas de otimização linear. Posteriormente, o método elipsoidal foi suplantado, em problemas lineares, pelos métodos de pontos interiores. No entanto, na otimização não-linear, o método elipsoidal permanece sendo uma importante alternativa. As referências (Bland, Goldfarb & Todd 1981, Akgül 1984) fornecem detalhes da história do desenvolvimento do método elipsoidal e de sua aplicação em contextos diversos.

Este capítulo faz um estudo breve dos métodos de *planos de corte*, que objetiva apenas a compreensão geral de seu mecanismo, e um estudo mais detalhado do *método elipsoidal*, em seu formato clássico e com algumas variações.

7.1 Formulação Geral

Dado o problema (mono-objetivo) irrestrito:

$$x^* = \arg\min_x f(x) \tag{7.1}$$

e dado um ponto inicial $x_0 \neq x^*$, obtém-se uma seqüência x_k tal que $x_k \to x^*$ a partir do algoritmo de otimização. Define-se ainda uma região inicial $\mathcal{Q}_0$ na qual se assume que deve estar contido o ponto x^*. Define-se ainda $\mathcal{H}(g, x)$ como o semi-espaço definido pelo vetor g que não o contém e que passa pelo ponto x. A família dos algoritmos de exclusão de semi-espaço possui a estrutura:

Algoritmo de Exclusão de Semi-Espaço

$k \leftarrow 1$

$\mathcal{Q}_k \leftarrow \mathcal{Q}_0$

$x_k \leftarrow x_0$

enquanto (não critério de parada)

$g_k \leftarrow$ subgradiente$(f(\cdot), x_k)$

$\mathcal{Q}_k \leftarrow V(\mathcal{Q}_{k-1} \cap \mathcal{H}(g_k, x_k))$

$x_{k+1} \leftarrow T(\mathcal{Q}_k)$

$k \leftarrow k + 1$

fim-enquanto

A especificidade do algoritmo em questão é definida, portanto, pela função $T(\cdot)$, que define a lei de determinação da nova estimativa de solução, e pela função $V(\cdot)$, que define a lei de construção de uma nova região, após a realização da exclusão de uma parcela da região anterior. Em particular, os algoritmos do tipo "elipsoidal" sempre construirão regiões elipsoidais, enquanto os algoritmos de "planos de corte" irão construir regiões poliedrais.

NOTA 7.1 *Os algoritmos de* exclusão de semi-espaços *não possuem a função objetivo como função descendente: a função $T(\cdot)$ não é construída para assegurar tal propriedade. De fato, usualmente ocorre oscilação no valor da função objetivo ao longo da execução do algoritmo. Outra grandeza, no entanto, é descendente com esses algoritmos: o* volume *da região onde está contido o ótimo.*

$\diamondsuit$

7.2 Métodos de Planos de Corte

Os métodos denominados de *planos de corte*[1] se aplicam a problemas cuja forma geral é:

$$x^* = \arg \min_x \ c^T x$$
$$\text{sujeito a: } x \in \mathcal{S} \tag{7.2}$$

sendo $\mathcal{S}$ um conjunto convexo fechado. A função objetivo, portanto, é um funcional linear. Problemas de formato geral envolvendo a minimização de

[1] O conteúdo desta seção segue predominantemente a apresentação em (Luenberger 1984).

funcionais convexos podem ser colocados nesta forma. Considere-se o problema:

$$\begin{gathered} x^* = \arg \min_x \ f(x) \\ \text{sujeito a: } x \in \mathcal{R} \end{gathered} \tag{7.3}$$

sendo $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ uma função convexa e $\mathcal{R} \subset \mathbb{R}^n$ um conjunto convexo compacto. Define-se uma nova variável:

$$\bar{x} = \begin{bmatrix} r \\ x \end{bmatrix} \tag{7.4}$$

de forma que $\bar{x} \in \mathbb{R}^{n+1}$. Com essa variável, o problema (7.3) pode ser escrito como:

$$\begin{gathered} \bar{x}^* = \arg \min \ r \\ \text{sujeito a: } \begin{cases} f(x) - r \leq 0 \\ \\ x \in \mathcal{R} \end{cases} \end{gathered} \tag{7.5}$$

Observando-se que o conjunto $\mathcal{S}$ definido por

$$\mathcal{S} = \{\bar{x} \mid f(x) - r \leq 0 \ , \ x \in \mathcal{R}\} \tag{7.6}$$

é um conjunto convexo no espaço $\mathbb{R}^{n+1}$ e que

$$r = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \end{bmatrix} \bar{x} \tag{7.7}$$

conclui-se que o problema no formato (7.5) está na forma (7.2), podendo portanto ser a ele aplicado um método de planos de corte.

Seja dado um politopo P_0 tal que $P_0 \supset \mathcal{S}$. Os métodos de planos de corte são dados pelo algoritmo geral:

Algoritmo de Planos de Corte

$k \leftarrow 0$

Minimizar $c^T x$ sobre P_k, obtendo x_k

enquanto $\text{dist}(x_k, \mathcal{S}) > \epsilon$

Determinar um hiperplano H_k que separa x_k de $\mathcal{S}$

Colocar este novo hiperplano como suporte de uma nova face para o politopo, que passa a ser o politopo P_{k+1}

Minimizar $c^T x$ sobre P_{k+1}, obtendo x_{k+1}

$k \leftarrow k + 1$

fim-enquanto

Deve-se observar a respeito desse algoritmo que:

- O mesmo depende de uma etapa que corresponde a uma otimização linear convencional (funcional objetivo linear e conjunto de restrições em formato de politopo);
- Dessa forma, sabe-se que a solução de tal etapa sempre corresponderá a um vértice do politopo;
- Como o politopo é exterior ao conjunto $\mathcal{S}$, e por construção preserva a cada passo tal propriedade, o ponto x_k sempre será exterior a $\mathcal{S}$;
- A aproximação da solução ocorre portanto assintoticamente, e necessariamente pelo lado de fora do conjunto factível. Note-se que, pela linearidade da função objetivo, a solução necessariamente deverá estar exatamente na fronteira de $\mathcal{S}$.

NOTA 7.2 *A principal distinção entre diferentes algoritmos de planos de corte está na maneira como cada algoritmo faz a determinação do plano H_k que irá separar o ponto x_k do conjunto $\mathcal{S}$. Diferentes escolhas produzem algoritmos que geram seqüências inteiramente diferentes de pontos, dadas as mesmas condições iniciais, e podem levar a velocidades de convergência muito distintas.*

♢

NOTA 7.3 *Um problema que afeta a eficiência computacional dos algoritmos de planos de corte é o fato de que os mesmos envolvem otimizações lineares cuja estrutura de restrições cresce a cada iteração, com o acréscimo de uma restrição por iteração. Para os padrões atuais, considera-se que os algoritmos de planos de corte em geral são pouco eficientes, de forma que os mesmos não são usualmente empregados a menos que a estrutura do problema em questão seja particularmente favorável à sua aplicação.*

♢

7.2.1 Algoritmo de Planos de Corte de Kelley

Nesta subseção é mostrada uma possível realização para a escolha dos planos de corte, proposta por Kelley. Considera-se o problema:

$$\begin{aligned} x^* = & \text{ arg } \min_x c^T x \\ & \text{sujeito a: } g(x) \leq 0 \end{aligned} \tag{7.8}$$

onde $x \in \mathbb{R}^n$ e $g(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^p$ é convexa. Com essas premissas, vale a desigualdade:

$$g_i(x) \geq g_i(w) + \nabla g_i(w)(x - w) \quad \forall\ x, w \tag{7.9}$$

onde $\nabla g_i(\cdot)$ é o gradiente da função $g_i(\cdot)$ no ponto (ou o subgradiente, no caso de se tratar de ponto onde a função é não-diferenciável). Sendo i o índice que indica a restrição com maior valor (a "restrição mais violada"), o hiperplano separador é definido por:

$$H_k = \{g_i(x_k) + \nabla g_i(x_k)(x - x_k) = 0\} \tag{7.10}$$

Esse hiperplano pode agora ser utilizado no *Algoritmo de Planos de Corte* genérico anteriormente apresentado.

7.3 Algoritmo Elipsoidal

Como exemplo de método da família dos métodos de exclusão de semi-espaço, é apresentado aqui o método elipsoidal, em suas variantes básica e com "deep-cuts". Para maiores detalhes, ver (Bland et al. 1981, Akgül 1984, Dziuban, Ecker & Kupferschmid 1985, Saldanha, Takahashi, Vasconcelos & Ramirez 1999).

Considere-se o problema de minimização do funcional $f_0(\cdot)$ numa região $\mathcal{S}$:

$$\begin{aligned} x^* = & \arg \min_x f_0(x) \\ & \text{sujeito a: } \{x \in \mathcal{S} \end{aligned} \tag{7.11}$$

A região factível é dada por:

$$\mathcal{S} \triangleq \{x \in \mathcal{S} \mid f_i(x) \leq 0\ \forall\ i = 1, \ldots, m\} \tag{7.12}$$

O vetor de variáveis de otimização é $x \in \mathbb{R}^n$.

O algoritmo elipsoidal básico é descrito pelas seguintes fórmulas recursivas que geram uma seqüência de pontos x_k:

$$\begin{aligned} x_{k+1} &= x_k - \beta_1 \frac{Q_k g_k}{(g_k^T Q_k g_k)^{\frac{1}{2}}} \\ Q_{k+1} &= \beta_2 \left(Q_k - \frac{\beta_3 (Q_k g_k)(Q_k g_k)^T}{g_k^T Q_k g_k} \right) \end{aligned} \tag{7.13}$$

com:

$$\beta_1 = \frac{1}{n+1} \qquad \beta_2 = \frac{n^2}{n^2-1} \qquad \beta_3 = \frac{2}{n+1}$$

O vetor g_k é um subgradiente da restrição mais violada $f_i(x_k)$, $i \geq 1$, ou, no caso de x_k estar na região factível, um subgradiente da função objetivo $f_0(x_k)$ naquele ponto. A explicação para tal tratamento das restrições é apresentada na Seção 7.4 a seguir. O algoritmo começa com um elipsóide E_0, centrado num ponto inicial x_0:

$$E_0 \triangleq \left\{ x \mid (x - x_0)^T Q_0^{-1} (x - x_0) \leq 1 \right\} \tag{7.14}$$

O ótimo x^*, no caso de problemas convexos (f_i convexo $\forall\ i = 0, \ldots, m$), garantidamente pertence ao semi-espaço H_k dado por:

$$H_k \triangleq \left\{ x \mid g_k^T (x - x_k) \leq 0 \right\} \tag{7.15}$$

A equação 7.13 é interpretada como uma seqüência de novos elipsóides E_k, cada um sendo o elipsóide de menor volume que contém o semi-elipsóide $E_{k-1} \cap H_{k-1}$. Como o elipsóide inicial contém, por premissa, a solução e como cada novo elipsóide também o contém, a seqüência de elipsóides converge para um elipsóide de volume zero que contém x^*. Se este objeto for um único ponto, este será o ponto x^*. A figura 7.1 mostra uma iteração do método elipsoidal.

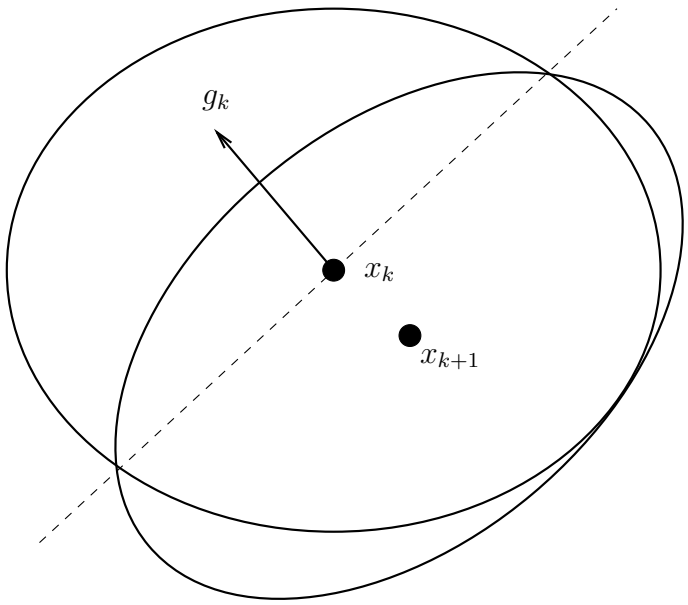


Figura 7.1: Ilustração de uma iteração do método elipsoidal. O elipsóide inicial (a elipse maior) tem centro no ponto x_k. O subgradiente da função objetivo, calculado no ponto x_k, resulta no vetor g_k. Esse vetor determina o semi-elipsóide que deve estar contido no novo elipsóide, cujo centro, após calculado pela fórmula de recorrência, é o ponto x_{k+1}. O novo elipsóide é também mostrado.

7.3.1 Algoritmo Elipsoidal com "Deep Cut"

O procedimento convencional de "deep cut" é descrito pelas fórmulas:

$$\begin{aligned} x_{k+1} &= x_k - \beta_1 \frac{Q_k g_k}{(g_k^T Q_k g_k)^{\frac{1}{2}}} \\ Q_{k+1} &= \beta_2 \left(Q_k - \beta_3 \frac{Q_k g_k (Q_k g_k)^T}{g_k^T Q_k g_k} \right) \end{aligned} \tag{7.16}$$

nas quais:

$$\beta_1 = \tfrac{1+n\alpha}{n+1} \quad \beta_2 = \tfrac{n^2(1-\alpha^2)}{n^2-1} \quad \beta_3 = \tfrac{2(1-n\alpha)}{(n+1)(1+\alpha)}$$

Este novo conjunto de fórmulas de atualização gera uma seqüência de elipsóides que contém a interseção de cada elipsóide anterior com um semi-espaço H_k que não passa mais pelo centro do elipsóide anterior:

$$H_k \triangleq \left\{ x \mid g_k^T (x - x_k) \leq -\alpha [g_k^T Q_k g_k]^{\frac{1}{2}} \right\} \tag{7.17}$$

A interseção $H_k \cap E_k$ é menor que a metade de E_k. O parâmetro α é chamado de *profundidade* do corte (cut), e representa a distância de x_k para o semi-espaço H_k na métrica correspondente à matriz Q_k. A equação 7.16 é adequada para determinar E_{k+1} com $-1/n \leq \alpha \leq 1$. Na faixa $0 \leq \alpha \leq 1/n$, o procedimento leva a uma convergência acelerada.

A convergência do algoritmo básico (sem "deep cut") para a solução x^*, que ocorre para problemas convexos, é devida à propriedade dos subgradientes de definirem um semi-espaço que garantidamente contém o mínimo do funcional[2]. Como o algoritmo de "deep cut" emprega um subconjunto desse semi-espaço, a solução x^* pode ser perdida neste caso, mesmo para problemas convexos.

Conjectura 7.1 *Sabe-se que, além da porção do elipsóide anterior definida pelo corte, cada novo elipsóide também contém uma "porção extra" do espaço. Essa "porção extra" permite ao algoritmo elipsoidal encontrar até mesmo pontos de mínimo que estejam fora do elipsóide inicial. Seria possível definir "deep cuts" tais que, explorando-se essa propriedade, permanecesse sendo garantida a propriedade de convergência para o ótimo global, mesmo havendo a perda do ponto de ótimo em alguma iteração?* ○

Conjectura 7.2 *Seria possível utilizar cortes com profundidade negativa, para permitir o corte de elipsóides com base em subgradientes avaliados fora do centro? (Isso seria potencialmente útil para restaurar o condicionamento numérico da matriz geradora do elipsóide durante o processo de otimização).* ○

Conjectura 7.3 *O padrão de convergência dos centros dos elipsóides quando o ótimo não está contido nos mesmos deve ser diferente daquele quando o ótimo está contido. Deve ser possível portanto elaborar um teste de "perda do ponto de ótimo".* ○

7.4 Tratamento de Restrições

A questão do tratamento de restrições não-lineares genéricas é uma antiga dificuldade que desafia os diferentes métodos de otimização conhecidos

[2]De fato, esta é a *definição* de subgradiente.

(Luenberger 1984). Métodos que se baseiam em funções de barreira ou penalidade, por exemplo, podem estar sujeitos a mal-condicionamento numérico. Outros métodos, tais como aqueles baseados em programação quadrática, necessitam aproximar as suas restrições por funções afins, o que pode impedir a sua aplicação em determinadas classes de problemas. Em particular, restrições não-diferenciáveis constituem uma classe de problemas para os quais existem poucos métodos disponíveis.

Os métodos da categoria de "exclusão de semi-espaços" têm em comum ainda a maneira imediata como tratam a questão das restrições. O mecanismo de tratamento de restrições se baseia no fato de que, dada uma restrição convexa (ou seja, que define uma região factível convexa), o subgradiente dessa restrição num ponto sempre define um semi-espaço no qual deverá estar contida a região factível. Dessa forma, não é necessária nenhuma alteração estrutural no algoritmo para tratar as restrições, basta substituir o subgradiente da função objetivo pelos subgradientes dos funcionais que representam as restrições violadas.

A estrutura lógica da abordagem de "exclusão de semi-espaços" para o tratamento de múltiplas restrições decorre do fato de que a interseção de conjuntos convexos é um conjunto convexo. Para ilustrar isso, considere-se o sistema com uma função-objetivo e duas funções de restrição, sendo as três funções convexas:

$$x^* = \arg \min_x f(x)$$

$$\text{sujeito a: } \begin{cases} g_1(x) \leq 0 \\ \\ g_2(x) \leq 0 \end{cases} \tag{7.18}$$

com $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$, $g_1(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ e $g_2(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$. A região factível do problema é dada pela interseção das regiões sub-nível das duas funções, para o nível zero:

$$\mathcal{F} = R(g_1, 0) \cap R(g_1, 0) \tag{7.19}$$

Esta região factível pode ainda ser interpretada como sendo a região sub-nível para o nível zero de uma outra função, $r(\cdot)$, construída como:

$$r(x) = \max\{g_1(x), g_2(x)\} \tag{7.20}$$

Dessa forma:

$$\mathcal{F} = R(r, 0) \tag{7.21}$$

Claramente, a partir da convexidade de $g_1(\cdot)$ e $g_2(\cdot)$, tem-se a convexidade de $r(\cdot)$ e de $\mathcal{F}$. Dessa forma, no caso geral, pode-se definir:

$$r(x) = \max\{g_1(x), g_2(x), \cdots, g_m(x)\} \tag{7.22}$$

A otimização de $r(x)$, claramente, resulta em um ponto factível, se houver tal ponto. A cada passo antes de se atingir a factibilidade, exclui-se apenas um semi-espaço infactível. Defina-se por fim a função $s(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ dada por:

$$s(x) = \begin{cases} f(x) \quad , & \text{se } r(x) \leq 0 \\ \\ r(x) \quad , & \text{se } r(x) > 0 \end{cases} \tag{7.23}$$

A otimização de $s(x)$ irá produzir a cada passo:

- a exclusão do semi-espaço em que o ponto de ótimo não se encontra, se o ponto corrente for factível;
- a exclusão do semi-espaço que é totalmente infactível, se o ponto corrente for infactível.

A otimização de $s(\cdot)$ leva, portanto, à convergência para o ótimo restrito x^*.

Devido a tal estrutura de tratamento de restrições, o método elipsoidal é conhecido como um procedimento particularmente flexível de otimização, que possui convergência garantida para o ótimo sob a condição de convexidade da função objetivo e das restrições (Akgül 1984). A simplicidade de se tratar com com restrições não-lineares neste método é uma das principais razões de sua aplicabilidade.

7.5 Características de Comportamento

O comportamento dessa categoria de métodos frente a algumas dificuldades que irão surgir com freqüência em problemas de otimização de tipo geral é agora discutido.

7.5.1 Descontinuidades e Não-Diferenciabilidade

As regiões de descontinuidade e não-diferenciabilidade de funcionais no espaço de parâmetros não constituem problema para a execução dos métodos de "exclusão de semi-espaços". Tais singularidades não obstruem a determinação de

subgradientes, que são o elemento utilizado para a evolução desses métodos. Não há também problemas com singularidades do tipo "atrativa", uma vez que o cálculo do próximo ponto estimado não é feito sobre uma trajetória que parte do ponto atual, como no caso dos algoritmos de busca em direções (isso faz com que que tais algoritmos fiquem presos nessas singularidades). O próximo ponto, ao invés disso, é obtido por construção a uma distância finita do ponto anterior, o que impede que a seqüência de estimativas fique "aprisionada".

7.5.2 Não-Convexidade

A principal premissa sobre a qual se assenta a categoria dos métodos de "exclusão de semi-espaços" é a convexidade de todos os funcionais envolvidos. Se tal premissa for violada, o processo de exclusão passa a ser feito "às cegas", de forma que a evolução do método torna-se imprevisível. É possível que, nessas circunstâncias, o método não convirja para o ponto de ótimo.

7.5.3 Multimodalidade

A questão da multimodalidade pode ser entendida como um caso particular da não-convexidade. Isso poderia levar a análise a conclusões semelhantes. No entanto, deve-se observar que uma função multimodal pode ser localmente convexa, de forma que os algoritmos de "exclusão de semi-espaços" podem convergir para mínimos locais. É possível, entretanto, que não ocorra convergência para nenhum mínimo.

7.5.4 Velocidade de Convergência

Deve-se notar inicialmente que as características de convergência tanto dos métodos de "direção de busca" quanto dos de "exclusão de semi-espaços" podem ser muito diferentes dependendo do método específico que estiver sendo referido em cada caso. De uma forma geral, entretanto, é possível afirmar que há uma tendência para que em problemas nos quais são aplicáveis tanto os métodos do tipo "direção de busca" quanto os métodos de "exclusão de semi-espaços", os primeiros apresentem maior velocidade de convergência. Os segundos devem ser aplicados, portanto, especificamente nos problemas em que houver não-diferenciabilidades que impediriam a convergência dos

primeiros. Há ainda o caso de problemas de factibilidade (que podem constituir a etapa inicial de grande parte dos problemas de otimização), em que a eficiência de alguns métodos (especificamente os métodos do tipo "elipsoidal") de "exclusão de semi-espaços" pode ser muito aumentada com a aplicação de "deep cuts".

7.6 Algoritmo Cone-Elipsoidal

Esta seção e as próximas do presente capítulo pretendem mostrar que a interpretação das condições de Karush-Kuhn-Tucker para otimalidade enquanto uma condição de "inexistência de um cone", e seu complemento como uma condição de "existência" de um cone de direções factíveis pode fornecer "insights" a respeito de problemas de otimização em geral. Esses cones oriundos da condição KKTE serão utilizados para aperfeiçoar as propriedades de convergência do algoritmo elipsoidal.

Nas seções que se seguem, é mostrado que: (i) em algumas situações, a configuração das restrições pode levar a uma convergência arbitrariamente lenta do método elipsoidal clássico para a solução do problema; e (ii) uma modificação do método elipsoidal, denominada *Método Cone-Elipsoidal* (MCE), proposta por este autor e colaboradores (Takahashi, Saldanha, Dias-Filho & Ramirez 2003), restaura as propriedades de convergência do método, obtendo a mesma taxa de convergência atingida quando o mesmo é aplicado a problemas irrestritos. O algoritmo MCE proposto preserva as condições formais de convergência para o ótimo em problemas convexos, possibilitando, ainda, tratar uma classe de problemas não-convexos.

Os dados apresentados a seguir foram, em grande parte, publicados na referência (Takahashi, Saldanha, Dias-Filho & Ramirez 2003) e na tese de doutorado, orientada pelo presente autor, (Dias-Filho 2003). Uma abordagem diferente da que está sendo proposta aqui foi empregada na referência (Shah, Mitchell & Kupferschmid 2001) para lidar com a mesma dificuldade de convergência do método elipsoidal em problemas com restrições de igualdade. Entretanto, tal referência se limita a abordar problemas com restrições lineares, fazendo uso de projeções do elipsóide dentro da variedade linear que define o conjunto factível.

7.7 Definição do Problema

Considere-se o problema de otimização com restrições de igualdade[3] e de desigualdade, definido no espaço dos parâmetros $x \in \mathbb{R}^n$:

$$x^* = \arg \min_x f(x)$$

$$\text{sujeito a: } \begin{cases} q_j(x) \leq 0 \ ; \ j = 1, \ldots, r \\ h_l(x) \leq 0 \ ; \ l = 1, \ldots, s \\ -h_l(x) \leq 0 \ ; \ l = 1, \ldots, s \end{cases} \tag{7.24}$$

Assume-se que as funções $f(\cdot)$, $q(\cdot)$ e $h(\cdot)$ devam ser convexas, para permitir a construção de provas formais de convergência para o ótimo global. No caso de funções arbitrárias, o método pode convergir para mínimos locais ou pode divergir (assim como qualquer outro método de otimização). O vetor de restrições é reescrito da seguinte forma:

$$g(x) = \left[\begin{array}{ccc} q^T(x) & h^T(x) & -h^T(x) \end{array} \right]^T \tag{7.25}$$

Seja $\mathcal{F}$ o conjunto de soluções factíveis do problema (7.24), ou seja, o conjunto dos pontos para os quais $g(x) \leq 0$.

7.8 Método Elipsoidal Convencional

O processamento de restrições no método elipsoidal clássico é realizado através do processamento de uma restrição por vez (no caso, a restrição mais violada):

$$g_{max} = \max_i g_i(x) \tag{7.26}$$

Defina-se:

$$m(\cdot) = \begin{cases} \nabla g_{max}(x) & \text{if } g_{max} < 0 \\ \nabla f(x) & \text{if } g_{max} \geq 0 \end{cases} \tag{7.27}$$

[3]Nas equações (7.24), cada restrição de igualdade foi substituída por duas restrições de desigualdade, ou seja: $h_l(x) = 0 \ \Leftrightarrow \ h_l(x) \leq 0$ e $-h_l(x) \leq 0$.

onde $\nabla(\cdot)$ significa ou o gradiente ou qualquer subgradiente do argumento. O algoritmo elipsoidal básico, neste caso, é descrito pelas equações recursivas (7.13) com o vetor m_k definido em (7.27).

7.8.1 Problemas Difíceis para o Método Convencional

Um problema formulado como (7.24), de fato, não pode ser diretamente resolvido pelo método elipsoidal clássico, uma vez que as restrições de igualdade, se existirem, estarão sempre ativas. Alguma relaxação nas mesmas é necessária, dentro da formulação clássica, para tais restrições:

$$\begin{aligned} x^* = & \ \arg \ \min_x \ f(x) \\ & \\ \text{sujeito a: } & \begin{cases} q_j(x) \leq 0 \ \ ; \ j = 1, \ldots, r \\ \\ h_l(x) \leq \epsilon \ \ ; \ l = 1, \ldots, s \\ \\ -h_l(x) \leq -\epsilon \ \ ; \ l = 1, \ldots, s \end{cases} \end{aligned} \tag{7.28}$$

Com essa reformulação, pode haver algum x tal que as restrições fiquem satisfeitas. O algoritmo clássico, neste caso, irá funcionar dirigindo o centro do elipsóide para a região factível, só então ativando a função objetivo enquanto "guia" do processo de otimização.

À medida em que $\epsilon \to 0$, o problema se torna mais difícil, terminando por recair na completa não-convergência do método para $\epsilon = 0$. Esse é um "caso-limite artificial" que, entretanto, expressa o que acontece quando o conjunto factível é muito "estreito". Note-se que: (i) isso pode ocorrer mesmo que haja apenas restrições de desigualdade; e (ii) isso, de fato, não significa que o conjunto factível é "pequeno": ele pode ser "grande", no sentido de conter pontos localizados a grande distância uns dos outros.

EXEMPLO 7.1 *Para ilustrar como o método convencional pode falhar, considere-se o problema:*

$$\begin{aligned} x^* = & \ \arg \ \min \ (x-c)^T(x-c) \\ & \\ & \textit{subject to: } \{ax + b = 0 \end{aligned} \tag{7.29}$$

com $x \in \mathbb{R}^2$ *e:*

$$a = \begin{bmatrix} 1 & -1 \end{bmatrix} \quad b = 1 \quad c = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$$

Esse problema possui como solução analítica $x^* = [\ 0 \quad 1\ ]^T$. *Fazendo* $\epsilon = 1 \times 10^{-5}$, *o método "converge" para* $x = [\ 0.5 \quad 1.5\ ]^T$, *que é o ponto mais próximo do ponto inicial, dentro do conjunto factível, mas que está distante do ótimo analítico. Nesse caso, a evolução do algoritmo foi guiada apenas pela função de restrição, a função objetivo nunca foi ativada (ou seja, seu gradiente não foi utilizado em nenhuma iteração pelo método para determinar o corte do elipsóide). A seqüência de pontos gerados pelo algoritmo é mostrada na Figura 7.2.* ♢

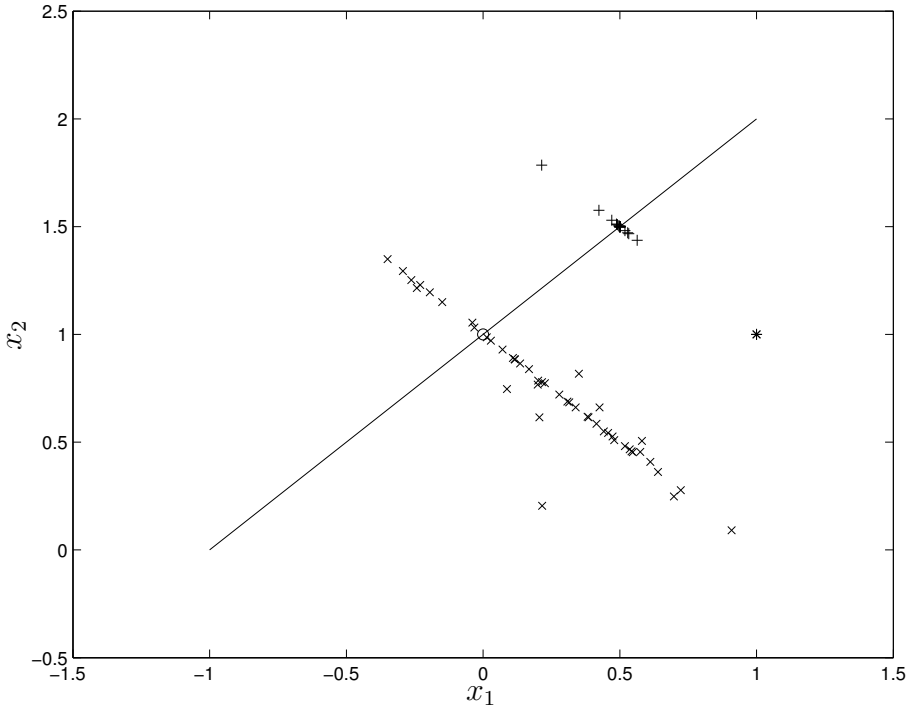


Figura 7.2: Seqüência dos centros dos elipsóides para o algoritmo elipsoidal convencional (+), e para o algoritmo cone-elipsoidal (MCE) (×). O conjunto factível é a linha, o ótimo restrito é representado por (○) e o ponto inicial de ambos os algoritmos é representado por (∗).

7.9 Cones das Direções Factibilizantes

Considere-se o problema de otimização, definido sobre o espaço de parâmetros $x \in \mathbb{R}^n$, com um conjunto de restrições:

$$\begin{aligned} \tilde{x} = \ & \arg \min_x \ f(x) \\ & \\ \text{sujeito a: } & \{g_j(x) \leq 0 \ ; \ j = 1, \ldots, r \end{aligned} \tag{7.30}$$

Seja $\mathcal{F}$ o conjunto dos pontos *factíveis* do problema (7.30), ou seja, dos pontos que satisfazem à desigualdade vetorial $g(x) \leq 0$.

Uma situação em que as condições KKT se encontram satisfeitas está exemplificada na figura 7.3. Nessa figura, estão representados os vetores gradiente da funções-objetivo e de duas restrições ativas, que atendem às condições.

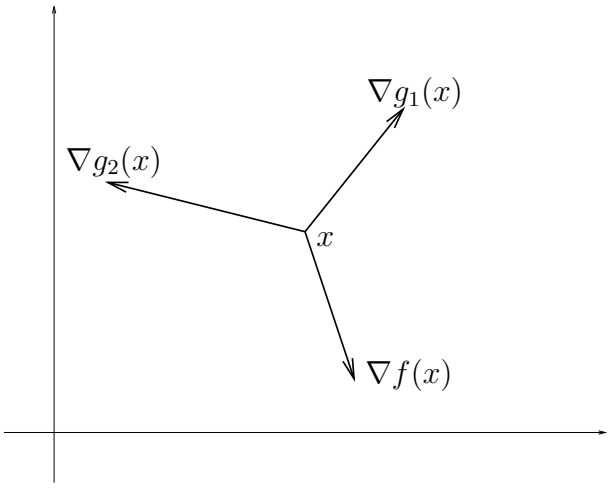


Figura 7.3: Exemplo de situação na qual estão atendidas as condições de Karush-Kuhn-Tucker (KKT). Nesse caso, existem multiplicadores positivos que fazem com que a soma dos vetores gradiente da função-objetivo, $\nabla f(x)$ e de duas restrições ativas, $\nabla g_1(x)$ e $\nabla g_2(x)$, no ponto x se anule.

De maneira similar, é possível definir uma condição necessária para a infactibilidade de um problema de otimização:

Teorema 7.1 (Condição Necessária para a Infactibilidade (CNI)) *Seja o problema de otimização (7.30) com todos g_i diferenciáveis e quasi-*

convexos. Se esse problema é infactível então existe pelo menos um ponto $x^{\emptyset}$ *tal que:*

i. $g_i(x^{\emptyset}) > 0$ *para algum* $i = 1, \ldots, r$*;*

ii. existe um vetor multiplicador $\mu^{\emptyset} \geq 0$*, com pelo menos uma desigualdade estrita* $\mu_i^{\emptyset} > 0$*, tal que:*

$$\begin{aligned} & g_k(x^{\emptyset}) < 0 \quad \Rightarrow \quad \mu_k^{\emptyset} g_k(x^{\emptyset}) = 0 \quad ; \quad k = 1, \ldots, r \\ & \sum_{k=1}^{r} \mu_k^{\emptyset} \nabla g_k(x^{\emptyset}) = 0 \end{aligned} \tag{7.31}$$

□

Uma situação em que as CNI se encontram satisfeitas está exemplificada na figura 7.4. Nessa figura estão representados os vetores gradiente de três restrições ativas que atendem às condições.

A idéia central neste trabalho é utilizar o complemento da condição CNI para gerar "direções de busca". Parte-se da observação de que um ponto x_k que não satisfaz às CNI possui a ele associado um cone de *direções factibilizantes*, ou seja, um cone de direções nas quais existem outros pontos x_{k+1} capazes de gerar soluções tais que $g(x_{k+1}) \leq g(x_k)$.

Definição 7.1 (Cone de direções factibilizantes (CDF)) *Seja* $\mathcal{C}_k$ *um cone com vértice num ponto* x_k *infactível. Esse cone é um* cone de direções factibilizantes *do problema de otimização (7.30) se:*

$$\forall x \in \mathcal{C}_k \exists \epsilon > 0 \mid g(x_k + \epsilon(x - x_k)) \leq g(x_k) \tag{7.32}$$

□

O teorema a seguir permite a determinação desse cone, para pontos infactíveis.

Teorema 7.2 *Considere-se o problema (7.30). Seja um ponto* x_k *tal que* $g_i(x_k) \geq 0$ *para algum* $i = 1, \ldots, r$*. Seja* $\mathcal{I} \triangleq \{i \mid g_i(x_k) \geq 0\}$*, com* v *elementos. Suponha-se que as funções* $g_i(\cdot)$ *tais que* $i \in \mathcal{I}$ *sejam diferenciáveis em* x_k*. Seja definida a matriz* G_f*:*

$$G_f = - \left[\begin{array}{ccc} \nabla g_{\mathcal{I}(1)}(x_k) & \ldots & \nabla g_{\mathcal{I}(v)}(x_k) \end{array} \right] \tag{7.33}$$

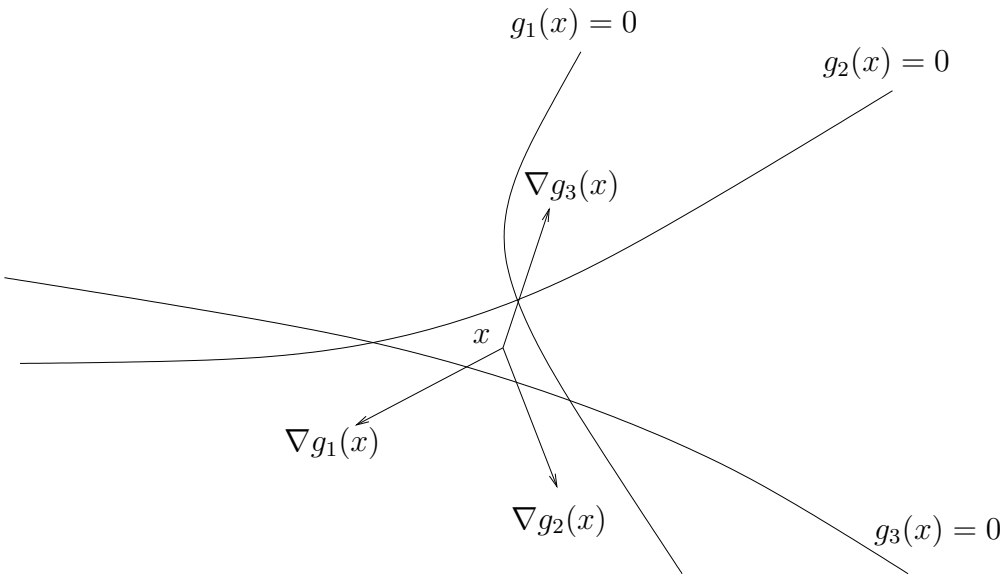


Figura 7.4: Exemplo de situação na qual estão atendidas as condições necessárias de infactibilidade (CNI). Nesse caso, existem multiplicadores positivos que fazem com que a soma dos vetores gradiente de três restrições ativas no ponto x, $\nabla g_1(x)$, $\nabla g_2(x)$ e $\nabla g_3(x)$, se anulem.

O cone de direções factibilizantes $\mathcal{C}_k^f$ associado ao ponto x_k é dado por:

$$x \in \mathcal{C}_k^f \Leftrightarrow G'_f(x - x_k) > 0 \tag{7.34}$$

□

Corolário 7.1 *Como corolário desse teorema, obtém-se que, caso o cone $\mathcal{C}^f$ não exista em nenhum ponto do espaço de parâmetros, então o problema é infactível.* □

O conceito de Cone de Direções Factibilizantes (CDF) é ilustrado nas figuras 7.5 e 7.6.

7.10 O Método Cone-Elipsoidal

O *Algoritmo Cone-Elipsoidal* utiliza o *cone CDF*, para o tratamento simultâneo de todas as restrições ativas e da função objetivo.

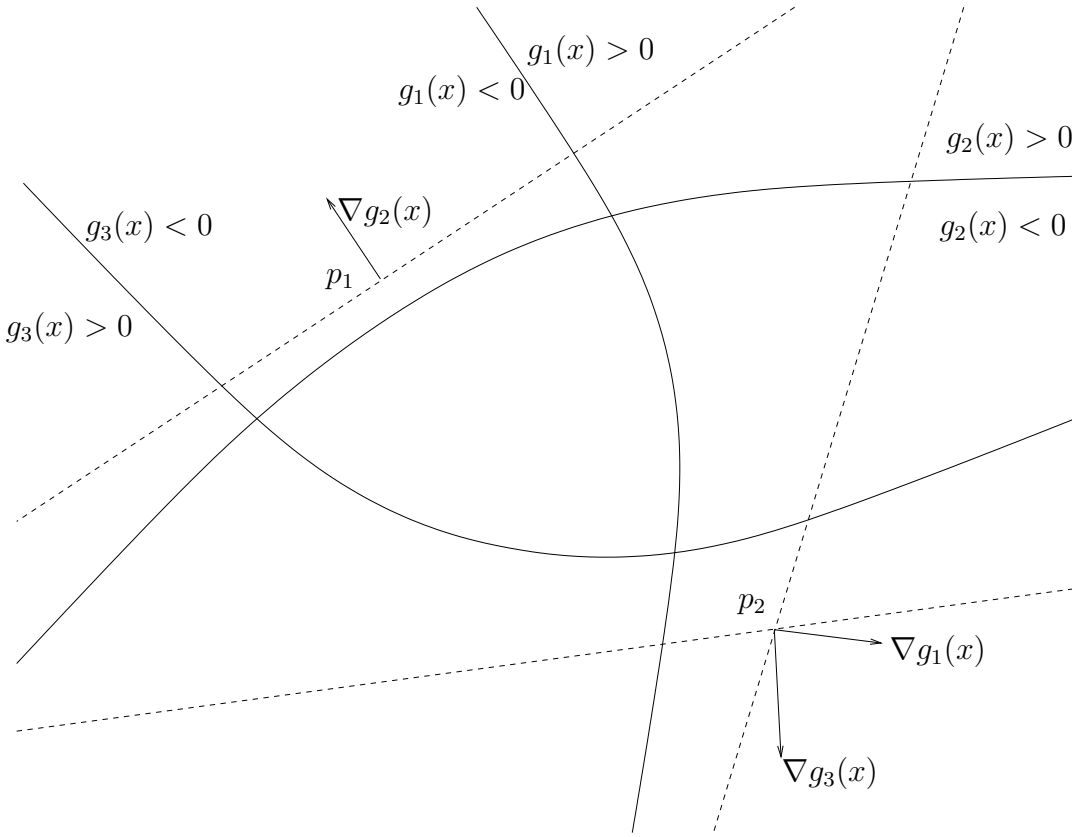


Figura 7.5: Nesta figura são estabelecidos os elementos para ilustrar o conceito de Cone de Direções Factibilizantes (CDF). São mostradas as variedades $g_1(x) = 0$, $g_2(x) = 0$, $g_3(x) = 0$, que determinam os limites da região factível do problema. São tomados dois pontos infactíveis, p_1 e p_2, sobre os quais são calculados os gradientes das restrições ativas em cada caso, $\nabla g_2(x)$ em p_1, e $\nabla g_1(x)$ e $\nabla g_3(x)$ em p_2. Para o ponto p_1, o Cone das Direções Factibilizantes (CDF) é todo o semi-espaço "atrás" do plano definido por $\nabla g_2(x)$. Pode-se constatar que, de fato, a região factível está toda contida nesse semi-espaço. O CDF associado ao ponto p_2 é mostrado na próxima figura.

7.10.1 Primeira Reformulação do Problema

Inicialmente, deve-se observar que é possível, sem perda de generalidade, transformar o problema com restrições de igualdade num problema com restrições estritamente de desigualdade, sendo possível, ainda, eliminar algumas restrições de desigualdade do problema, desde que se conheça "a priori" qual

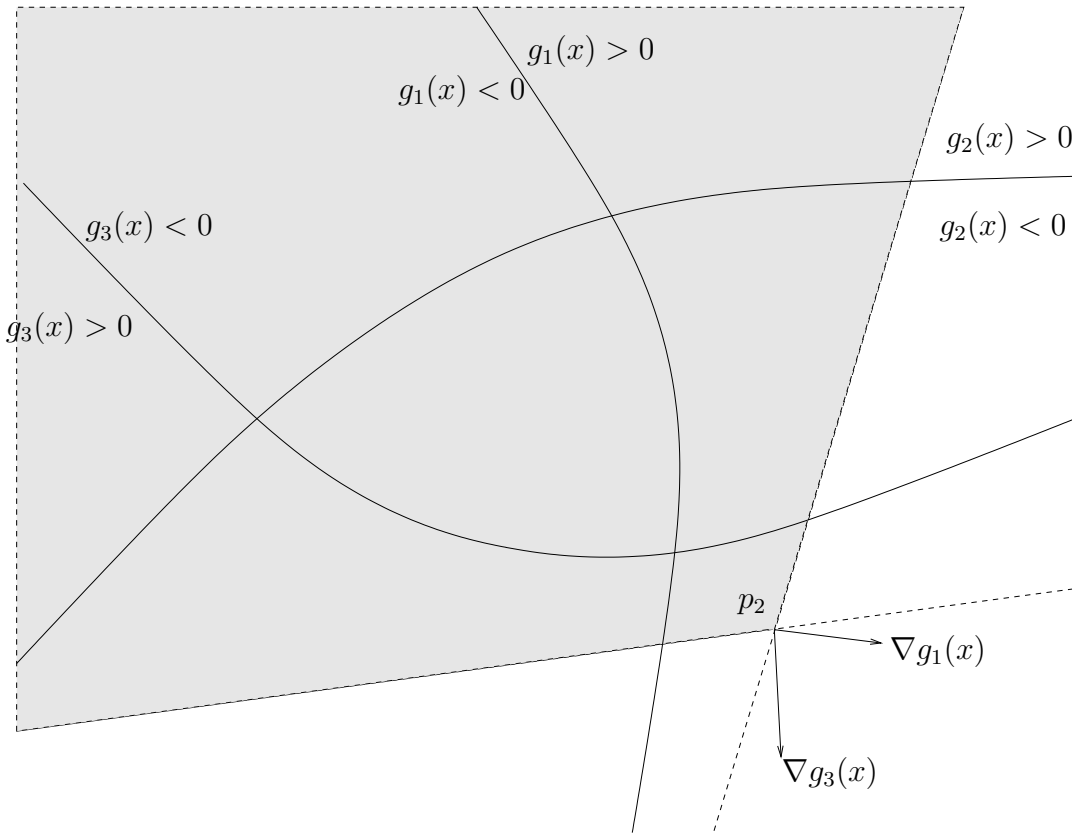


Figura 7.6: Nesta figura são mostrados os mesmos elementos definidos na figura anterior. O Cone das Direções Factibilizantes (CDF) associado ao ponto p_2 é indicado pela região hachurada. Deve-se observar que toda a região factível está de fato contida nesse cone.

é o ponto de ótimo *irrestrito* do problema, ou, pelo menos, quais restrições estariam ativas nesse ponto. Claramente, basta manter essas restrições, sendo possível simplesmente descartar aquelas que estivessem inativas no ótimo irrestrito. Tendo em vista essa observação, o problema pode ser reformulado como:

$$\begin{aligned} x^* = \ & \arg \min_x \ f(x) \\ & \text{sujeito a: } \left\{ \ g_j(x) \leq 0 \ \ ; \ j = 1, \ldots, m \right. \end{aligned} \tag{7.35}$$

sendo $m \leq r + s$ e sendo $g_i(x)$ o conjunto de restrições de desigualdade do problema (7.24) que se encontram ativas no ponto x_o definido por:

$$x_o = \arg \min_x f(x) \tag{7.36}$$

ou seja:

$$\begin{array}{l} g_j(x_o) = h_i(x_o) > 0 \\ \text{ou} \\ g_j(x_o) = q_l(x_o) > 0 \end{array} \tag{7.37}$$

Note-se que, em alguns problemas, é possível saber de antemão, a partir do significado físico das variáveis, quais restrições deveriam estar ativas num hipotético "ótimo irrestrito". Se não for esse o caso, é sempre possível executar um algoritmo de otimização sem restrições, para determinação prévia desse "ótimo irrestrito".

A simples utilização de (7.35) em lugar de (7.24) já permitiria tratar da dificuldade acima descrita, permitindo a convergência do método elipsoidal convencional para o ponto de ótimo de problemas com restrições de igualdade. Entretanto, é possível melhorar ainda mais as propriedades de convergência do método, utilizando a idéia de "cones", o que permitirá aplicar a informação simultânea de mais de uma restrição e da função objetivo em um único corte.

7.10.2 Segunda Reformulação do Problema

A idéia fundamental do método proposto neste capítulo encontra-se na observação de que, a cada vez que o algoritmo determina um novo ponto factível x_k do problema modificado (7.35), pode-se introduzir a informação de que o ponto de ótimo do problema, x^*, terá função-objetivo com valor menor ou igual ao de x_k. Essa informação pode ser entendida como uma nova "restrição", e a interseção dessa seqüência de restrições definidas pela seqüência dos x_k é dada por:

$$f(x_k) - \min_{i \in \mathcal{Z}} f(x_i) \leq 0 \tag{7.38}$$

onde $\mathcal{Z}$ designa o conjunto dos índices para os quais o ponto tornava o problema (7.35) factível, ou seja:

$$g(x_i) \leq 0 \quad \forall\, i \in \mathcal{Z} \tag{7.39}$$

Dessa forma, o problema (7.24) pode sofrer nova reformulação, ficando:

$$x^* = \arg\min_x f(x)$$

$$\text{sujeito a: } \begin{cases} q_j(x) < 0 \ ; \ j = 1, \ldots, r \\ h_l(x) < 0 \ ; \ l = 1, \ldots, s \\ -h_l(x) < 0 \ ; \ l = 1, \ldots, s \\ f(x) - \min_{i \in \mathcal{Z}} f(x_i) < 0 \end{cases} \tag{7.40}$$

O problema (7.40) terá sempre no mínimo uma restrição ativa (observe-se que as desigualdades foram substituídas por desigualdades estritas, para impedir a ocorrência, mesmo hipotética, de factibilidade). Dessa forma, o problema de determinação de x^* é substituído pelo problema de factibilidade de (7.40), o qual pode sempre ser tratado por meio dos cones CDF.

A interpretação desse resultado é apresentada na figura 7.7.

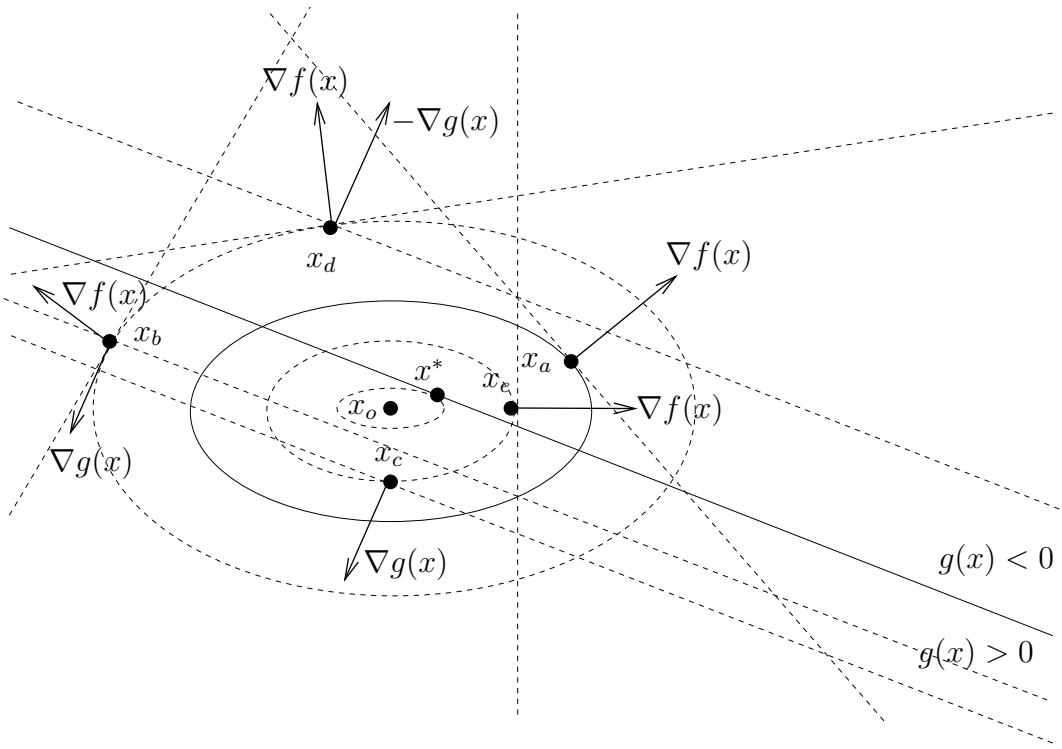


Figura 7.7: Nesta figura são mostradas todas as possíveis situações que podem ocorrer para a evolução do algoritmo MCE. O ponto x^* corresponde ao ponto de ótimo do problema, e o ponto x_o corresponde ao ponto de "ótimo irrestrito" do problema. São esboçadas algumas curvas de nível da função objetivo $f(x)$, com formato aproximadamente elipsoidal. É também mostrada a variedade convexa $g(x) = 0$, que passa por x^*, que corresponde à restrição de igualdade do problema. Suponha-se que se verificou, após a obtenção do ponto x_a, que este constituía a melhor estimativa até aquele instante do ótimo do problema, ocorrido do lado oposto ao de x_o. Isso faz com que a restrição $f(x) < f(x_a)$ passe a valer para os próximos passos do algoritmo, até que se encontre outro ponto do lado oposto ao de x_o que permita reduzir ainda mais a região onde possivelmente se encontraria x^*. Haveria quatro possibilidades para a evolução do algoritmo. O próximo ponto poderia ser tal que: (x_b) o valor da função objetivo aumentasse e o ponto estivesse do mesmo lado que x_o, o que faria com que o cone fosse definido pelos gradientes da restrição e da função objetivo; (x_c) o valor da função objetivo diminuísse, estando o novo ponto do mesmo lado que x_o, o que faria com que o cone fosse definido apenas pelo gradiente da restrição; (x_d) o valor da função objetivo aumentasse, estando o ponto do lado oposto ao de x_o, sendo, nesse caso, o cone definido tanto pelo gradiente da restrição quanto pelo da função objetivo; e (x_e) o valor da função objetivo diminuisse, estando o ponto do lado oposto ao de x_o, sendo que nesse caso o cone seria definido apenas pelo gradiente da função objetivo, e a restrição associada à função objetivo seria atualizada para $f(x) < f(x_e)$. O leitor deve observar que, em todas as situações, o cone contém o ponto x^*.

7.11 O Algoritmo MCE

Dado esse resultado, o tradicional algoritmo elipsoidal pode ser empregado agora com qualquer vetor d_k tal que:

$$\langle d_k, c_k \rangle \ \ \forall c_k \in \mathcal{C}_k. \tag{7.41}$$

No algoritmo elipsoidal, d_k irá substituir m_k, o (sub-)gradiente da função objetivo ou da restrição mais violada, na equação (7.13). A idéia é que cada corte do elipsóide irá manter no novo semi-elipsóide o *cone de direções factibilizantes* $\mathcal{C}_k$. O algoritmo irá gerar uma seqüência de elipsóides que irá convergir para uma solução factível do problema.

Uma adaptação do método, é obtida empregando a definição de *op-factibilidade* apresentada a seguir:

Definição 7.2 (Op-Factibilidade) *Um ponto x_k é op-factível se atender simultaneamente às condições:*

$$g_i(x_k) \leq 0 \tag{7.42}$$

e

$$f(x_k) \leq f(x_i) \ \ \forall \ x_i \mid g(x_i) \leq 0 \ , \ i < k \tag{7.43}$$

□

O conceito de pontos *op-factíveis* permite escrever o problema de encontrar pontos "melhores" (sob o ponto de vista da função objetivo) e simultaneamente "factíveis" como um problema de factibilidade. Assim, a otimização utilizando a idéia de manter os *cones de direções factíveis* no interior da sequência de elipsóides gerada pelo método elipsoidal é capaz de levar em consideração simultaneamente as restrições e a função objetivo para a determinação da solução ótima do problema.

Um algoritmo para encontrar o vetor d_k pode ser estabelecido a partir de qualquer combinação convexa dos vetores gradiente das restrições violadas $\nabla g_i(x)$ (incluindo a "op-restrição" $f(x_k) \leq \min f(x_i)$):

$$\begin{aligned} d_k &= \sum_{i=1}^{m} \nabla \alpha_i g_i(x_k) \\ \alpha_i &\geq 0 \\ g_i(x_k) < 0 &\Rightarrow \alpha_i = 0 \end{aligned} \tag{7.44}$$

Para que o algoritmo funcione corretamente, a "op-restrição" só pode ser incluída após ter sido encontrado pelo menos um ponto factível do problema.

Note-se que, na abordagem proposta, não há nenhuma perda da propriedade de convergência garantida, que ocorreria em outros tipos de tentativa de se ponderar o funcional objetivo e as funções de restrição. Isso é particularmente importante quando existem restrições de igualdade que, de outra forma, poderiam impedir a função objetivo de ser considerada.

EXEMPLO 7.2 *Considere-se o mesmo problema analítico (7.29) mostrado difícil para o método elipsoidal tradicional, dependendo do tamanho da relaxação que for empregada. Agora, o problema é considerado como ele é,* sem qualquer relaxação, *dentro da formulação (7.24). O* Método Cone-Elipsoidal *(MCE), começando do mesmo ponto inicial com o mesmo elipsóide inicial, converge para o ótimo, como mostrado na Figura 7.2. Claramente, o gradiente da função objetivo dirige o algoritmo para o ótimo restrito. O algoritmo converge para este ótimo, ao mesmo tempo se aproximando do conjunto factível.* ♢

7.12 Não-Convexidade de Restrições de Igualdade

A principal premissa sobre a qual se assenta a categoria dos métodos de "exclusão de semi-espaços" é a convexidade da função-objetivo e da região factível. Entretanto, sabe-se que o único tipo de restrição de igualdade que produz regiões factíveis convexas é o "afim" (ou seja, constituído por funcionais lineares).

A metodologia aqui desenvolvida, entretanto, permite tratar um caso específico de restrição de igualdade não-linear, preservando a propriedade de convergência garantida do método. A situação "tratável" se estabelece quando a variedade não-linear $g(x) = 0$ divide o espaço em duas regiões, sendo uma convexa e a outra côncava, ocorrendo ainda que o ponto de ótimo irrestrito x_o se encontre no interior da região côncava. Nesse caso, é imediato observar que o problema (7.35) é convexo. Pode-se então empregar uma nova

reformulação do problema:

$$x^* = \arg\min_x f(x)$$

$$\text{sujeito a: } \begin{cases} g_j(x) < 0 \ ; \ j = 1, \ldots, m \\ f(x) - \min_{i \in \mathcal{Z}} f(x_i) < 0 \end{cases} \tag{7.45}$$

na qual, novamente, se resolve o problema de factibilidade utilizando os cones CDF. A diferença desse problema em relação ao problema (7.40) é que, para pontos no interior da região convexa, apenas o gradiente da função-objetivo é empregado para produzir cortes. O gradiente da restrição é utilizado somente na região côncava. Essa situação é ilustrada na figura 7.8. O leitor é convidado a comparar essa figura com a figura 7.7.

7.13 Conclusões

Foi mostrada a adaptação dos Cones de Direções Factibilizantes para agregar, em um único corte, a informação proveniente de várias restrições e da função objetivo, simultaneamente. Além disso, foi apresentada uma forma de tratar restrições de igualdade, eventualmente sendo possível a transformação de problemas com restrições de igualdade não-lineares em problemas convexos.

Tais questões foram abordadas dentro do contexto da otimização baseada no algoritmo elipsoidal. Foi mostrado, por meio de um exemplo simples, que os procedimentos propostos podem acelerar a convergência do método elipsoidal, ou, até mesmo, permitir tal convergência em situações nas quais a mesma não ocorreria.

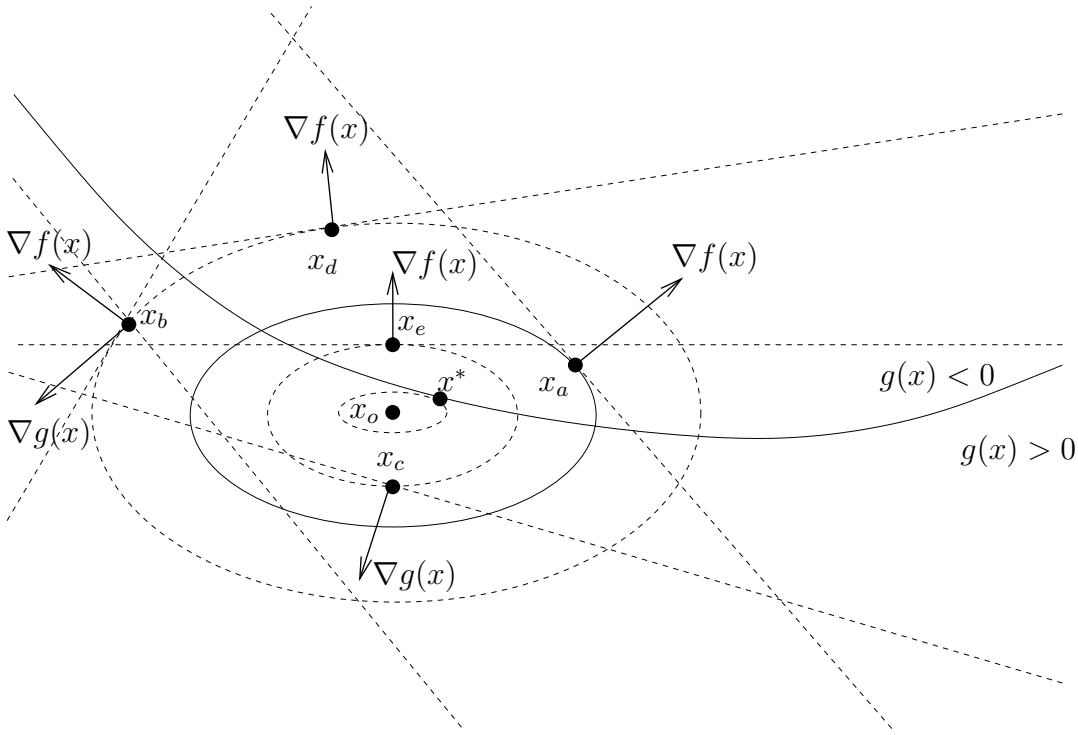


Figura 7.8: Nesta figura são mostradas todas as possíveis situações que podem ocorrer para a evolução do algoritmo MCE, agora para uma restrição de igualdade $g(x) = 0$ não convexa. A variedade não-convexa $g(x) = 0$, que passa por x^*, define uma região convexa $g(x) < 0$ e outra região côncava $g(x) > 0$ no espaço de parâmetros. O ponto x^* corresponde ao ponto de ótimo do problema, e o ponto x_o corresponde ao ponto de "ótimo irrestrito" do problema (é importante notar que esse ponto encontra-se na região côncava definida por $g(x) > 0$). São esboçadas algumas curvas de nível da função objetivo $f(x)$, com formato aproximadamente elipsoidal. Suponha-se que se verificou, após a obtenção do ponto x_a, que este constituía a melhor estimativa até aquele instante do ótimo do problema, ocorrido do lado oposto ao de x_o. Isso faz com que a restrição $f(x) < f(x_a)$ passe a valer para os próximos passos do algoritmo, até se encontrar outro ponto do lado oposto ao de x_o que permita reduzir, ainda mais, a região onde possivelmente se encontraria x^*. Haveria quatro possibilidades para a evolução do algoritmo. O próximo ponto poderia ser tal que: (x_b) o valor da função objetivo aumentasse e o ponto estivesse do mesmo lado que x_o, o que faria com que o cone fosse definido pelos gradientes da restrição e da função objetivo; (x_c) o valor da função objetivo diminuísse, estando o novo ponto do mesmo lado que x_o, o que faria com que o cone fosse definido apenas pelo gradiente da restrição; (x_d) o valor da função objetivo aumentasse, estando o ponto do lado oposto ao de x_o, o cone neste caso, diferentemente do caso da restrição convexa, seria definido apenas pelo gradiente da função objetivo; e (x_e) o valor da função objetivo diminuisse, estando o ponto do lado oposto ao de x_o, sendo que o cone seria definido apenas pelo gradiente da função objetivo, e a restrição associada à função objetivo seria atualizada para $f(x) < f(x_e)$. O leitor deve observar que, em todas as situações, o cone contém o ponto x^*. O leitor ainda é convidado a comparar esta figura com aquela do caso da restrição convexa.

Capítulo 8

Métodos de Otimização por Populações

Os métodos de otimização por exclusão de semi-espaços e os métodos de otimização por direções de busca trabalham sempre com:

- apenas uma estimativa corrente da solução;
- informações obtidas apenas localmente, sobre a estimativa corrente, para definir a próxima estimativa de solução.

Diferentemente disso, os métodos aqui denominados de "otimização por populações" poderão trabalhar com:

- informação a respeito de mais de um ponto, que é tratada como "informação corrente"; e/ou
- informações obtidas em mais de um ponto do espaço de soluções que é utilizada para definir a transição do estado corrente para o próximo estado da seqüência de soluções.

Um método primitivo de otimização utilizando a lógica das "populações" é o chamado *Algoritmo Evolucionário* (AE). Os principais algoritmos atualmente utilizados, representativos dos métodos de populações, são os *Algoritmos Genéticos* (AG's) e o *Algoritmo de Simulated Annealing* (SA). Esses três tipos de algoritmos serão abordados neste capítulo, sendo os últimos (os AG's) estudados em maior detalhe. Existem outros métodos que se enquadram na definição de métodos de populações, e que não serão tratados

aqui, tais como: *Otimização por Colônia de Formigas* (Dorigo, Maniezzo & Colorni 1996), *Máquinas de Boltzman* (Ackley, Hinton & Sejnowski 1985), etc.

A formulação geral dos métodos de populações pode ser sistematizada conforme o esquema a seguir. Defina-se $\mathcal{P}_i$ como um certo conjunto de pontos pertencentes ao espaço de objetivos, e define-se Φ como um vetor de variáveis aleatórias com determinada distribuição de probabilidades. O formato básico dos métodos dessa família é:

Algoritmo de População

- $k \leftarrow 1$
- $\mathcal{P}_k \leftarrow \mathcal{P}_0$
- enquanto (não critério de parada)
- $A_k \leftarrow$ avaliação$(f(\mathcal{P}_k))$
- $\mathcal{P}_{k+1} \leftarrow G(A_k, \mathcal{P}_k, \Phi_k)$
- $k \leftarrow k + 1$
- fim-enquanto

Por utilizar o vetor Φ de variáveis aleatórias para definir a transição de uma população para a seguinte, os métodos de populações são freqüentemente denominados também *métodos estocásticos*, ou ainda *algoritmos evolucionários* (esta é uma denominação geral, para toda a categoria de métodos deste tipo, que não deve ser confundida com a denominação *algoritmo evolucionário simples*, que se refere a um método dessa família, bastante simples, que será mostrado neste capítulo).

Note-se que não é feito uso de nenhuma informação a respeito de gradientes de funções. Cada instância particular do algoritmo de população é construída definindo-se a função $G(\cdot, \cdot, \cdot)$, que define como é feita a transição de uma população para a população seguinte. A evolução da população ao longo das iterações conduz a uma convergência do conjunto dos "indivíduos" para uma região próxima de um ponto de ótimo. Um subconjunto da população efetivamente converge para uma vizinhança do ponto de ótimo.

8.1 Algoritmo Evolucionário Simples

O modelo mais simples de algoritmo de populações é baseado apenas em uma busca aleatória cuja "média" vai sendo deslocada à medida em que são geradas novas avaliações da função objetivo. Este tipo de algoritmo é aqui denominado *Algoritmo Evolucionário Simples*. Uma possível versão desse algoritmo é fornecida a seguir.

Considere-se o problema de otimização irrestrita:

$$x^* = \arg \min_x f(x) \tag{8.1}$$

sendo $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$. Supõe-se que, embora o problema seja irrestrito, sabe-se *a priori* que as soluções de interesse provavelmente estarão no interior de uma esfera de raio ρ e centro x_0.

Denomina-se uma *população* um conjunto $X = \left[\begin{array}{cccc} x_1 & x_2 & \dots & x_N \end{array} \right]$, com $x_i \in \mathbb{R}^n$. A avaliação da função objetivo nos pontos correspondentes aos elementos da população é organizada no vetor $F = \left[\begin{array}{cccc} f(x_1) & f(x_2) & \dots & f(x_N) \end{array} \right]$. $F(i)$ irá designar o escalar $f(x_i)$ e $X(i)$ irá designar o vetor x_i. As variáveis X e F receberão índices para designar a iteração à qual correspondem.

Supõem-se disponíveis as funções:

- randn(p, q): gera uma matriz de p linhas e q colunas, sendo cada entrada da matriz um número aleatório com distribuição Gaussiana, média zero e variância um.
- logspace(p, q, r): gera um vetor de r componentes, logaritmicamente espaçados, começando em 10^p e terminando em 10^q.

O *Algoritmo Evolucionário Simples* é mostrado a seguir.

Algoritmo Evolucionário Simples

Entradas do algoritmo: ρ, x_0, N

$V \leftarrow \rho \cdot \text{logspace}(-3, 0, N)$

$k \leftarrow 0$

$x_{min} \leftarrow x_0$

enquanto (não critério de parada)

para $i \leftarrow 1$ até N faça
  $\Gamma \leftarrow \text{randn}(n, 1)$
  $X_k(i) \leftarrow x_{min} + V(i)\Gamma$
  $F_k(i) \leftarrow f(X_k(i))$
fim-para
$j \leftarrow$ arg $\min_i$ $F_k(i)$
$f_{min} \leftarrow F_k(j)$
$x_{min} \leftarrow X_k(j)$
fim-enquanto

Claramente, o *Algoritmo Evolucionário Simples* produz sucessivas populações, sempre centradas no "melhor" elemento obtido na população da iteração anterior, sendo que cada elemento da nova população é gerado a distâncias que em média irão desde $\rho \times 10^{-3}$ até ρ. Dessa forma, garante-se que o algoritmo faça sempre uma busca em regiões próximas à melhor estimativa disponível, realizando também a procura em regiões distantes, na tentativa de se escapar de mínimos locais.

NOTA 8.1 *Esse algoritmo, na forma apresentada, serve para a determinação de uma única bacia de atração. Para o propósito de se localizarem mais bacias de atração, é possível executar novamente o algoritmo, excluindo os pontos gerados que se encontrem a distâncias menores que um certo valor d em relação ao(s) mínimo(s) anteriormente encontrado(s). O valor de d depende do tamanho que se espera que possuam as bacias de atração da função $f(\cdot)$.*

$\diamondsuit$

8.2 Algoritmo de Simulated Annealing

O algoritmo de *Simulated Annealing* (ou, em português, *Recozimento Simulado*) emprega um princípio de funcionamento parecido com o do *Algoritmo Evolucionário*. No entanto, agora há a previsão de que eventualmente ocorram transições de uma solução corrente para outra pior que a corrente,

segundo certa probabilidade. Essa probabilidade foi estabelecida para mimetizar as probabilidades de mudança de estados em sólidos submetidos a processos de "recozimento", daí a denominação do algoritmo.

Considere-se o problema de otimização irrestrita:

$$x^* = \arg \min_x f(x) \tag{8.2}$$

sendo $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$. Supõe-se que, embora o problema seja irrestrito, sabe-se *a priori* que as soluções de interesse provavelmente estarão no interior de uma esfera de raio ρ e centro x_0.

Denomina-se uma *população* um conjunto $X = \left[\begin{array}{cccc} x_1 & x_2 & \ldots & x_N \end{array} \right]$, com $x_i \in \mathbb{R}^n$. A avaliação da função objetivo nos pontos correspondentes aos elementos da população é organizada no vetor $F = \left[\begin{array}{cccc} f(x_1) & f(x_2) & \ldots & f(x_N) \end{array} \right]$. $F(i)$ irá designar o escalar $f(x_i)$ e $X(i)$ irá designar o vetor x_i. As variáveis X e F receberão índices para designar a iteração à qual correspondem.

Supõem-se disponíveis as funções:

- $\underline{\text{randn}}(p, q)$: gera uma matriz de p linhas e q colunas, sendo cada entrada da matriz um número aleatório com distribuição Gaussiana, média zero e variância um.
- $\underline{\text{rand}}(p, q)$: gera uma matriz de p linhas e q colunas, sendo cada entrada da matriz um número aleatório com distribuição uniforme entre 0 e 1.

O *Algoritmo de Simulated Annealing* é mostrado a seguir.

Algoritmo de Simulated Annealing

Entradas do algoritmo: ρ, x_0, N

$\alpha \leftarrow 0.9$

$\underline{\text{para}}\ i \leftarrow 1\ \underline{\text{até}}\ N\ \underline{\text{faça}}$

$\quad \Gamma \leftarrow \underline{\text{randn}}(n, 1)$

$\quad X_0(i) \leftarrow x_0 + \rho\Gamma$

$\quad F_0(i) \leftarrow f(X_k(i))$

$\underline{\text{fim-para}}$

$H \leftarrow \max(|F_0(1) - F_0(2)|, |F_0(2) - F_0(3)|, \ldots, |F_0(N-1) - F_0(N)|)$

$j \leftarrow$ arg $\min_i$ $F_k(i)$

$x_{min} \leftarrow X_k(j)$

$y \leftarrow x_{min}$

$q \leftarrow 1$, $p \leftarrow 1$

enquanto (não critério de parada)

$T \leftarrow 3\alpha^q H$

$x \leftarrow y + \alpha^q \cdot \rho \cdot \underline{\text{randn}}(n, 1)$

se $f(x) \leq f(y)$

$y \leftarrow x$

se $f(x) < f(x_{min})$

$x_{min} \leftarrow x$

fim-se

senão

$\psi = \exp\left(\frac{f(y)-f(x)}{T}\right)$

$\zeta = \underline{\text{rand}}(1, 1)$

se $\zeta \leq \psi$

$y \leftarrow x$

fim-se

fim-se

$p \leftarrow p + 1$

se $p > 100$

$p \leftarrow 1$

$q \leftarrow q + 1$

fim-se

fim-enquanto

A variável T dentro do algoritmo faz a analogia com a "temperatura" de um processo de recozimento, estando positivamente correlacionada com a probabilidade de ocorrência de transições de estado.

8.3 Algoritmos Genéticos

Os *Algoritmos Genéticos* (AG's) utilizam um mecanismo de funcionamento bastante distinto dos anteriormente apresentados. Tal mecanismo baseia-se na idéia de mimetizar a evolução natural dos seres vivos. Os AG's são caracterizados pela "evolução" de um conjunto de soluções-tentativas (*população*), segundo regras estocásticas de busca e combinação que, levam de uma população à seguinte, numa seqüência de *gerações*. A existência de três regras básicas, ou *operadores genéticos* básicos, define um Algoritmo Genético:

i. Um operador de *cruzamento*, que combina a informação contida em dois ou mais *indivíduos* (ou seja, duas ou mais soluções-tentativas), assim gerando outros indivíduos;

ii. Um operador de *mutação* que, utilizando a informação contida em um indivíduo, estocasticamente gera outro indivíduo; e

iii. Um operador de *seleção* que, utilizando a avaliação da função objetivo sobre todos os indivíduos da população, produz réplicas de alguns desses indivíduos e elimina outros indivíduos, assim gerando a próxima população.

Um Algoritmo Genético pode ser construído a partir dessas três regras somente, ou pode conter outros tipos de regras, tais como: *nicho*, *busca local*, etc.

Como referências para o estudo dos Algoritmos Genéticos, são indicados os artigos tutoriais (Tanomaru 1995, K. F. Man & Kwong 1996, Johnson & Ramat-Semii 1997). O presente autor e colaboradores estudaram a questão da eficiência dos AG's, sendo os resultados desses estudos relatados em (Vasconcelos, Ramirez, Takahashi & Saldanha 2001, Takahashi, Vasconcelos, Ramirez & Krahenbuhl 2003).

8.3.1 Algoritmo Genético - Codificação Binária

Como exemplo de método pertencente à família dos Algoritmos Genéticos, é aqui apresentada em primeiro lugar uma formulação comumente aplicada. Por possuir codificação binária das variáveis de otimização, tal algoritmo será aqui denominado AG-B. A forma específica como encontra-se implementado o AG-B, descrita a seguir, segue a referência (Tanomaru 1995).

Algoritmo Genético Binário (AG-B)

- Cada parâmetro de projeto é codificado segundo uma codificação binária com 16 bits de comprimento, correspondendo à faixa de variação desse parâmetro. O código de um determinado *indivíduo* é obtido pela concatenação das codificações correspondentes a cada um dos parâmetros.
- O algoritmo se inicia com a geração aleatória de um número N, (usualmente grande) de conjuntos de parâmetros (ou *indivíduos*) dentro da faixa admissível.
- São realizadas em seqüência as operações de: *cruzamento*, *mutação*, *avaliação*, *cálculo da função de ajuste* ("fitness function"), *seleção* e *elitização*, sendo gerada nova população com mesmo número de indivíduos.
- O algoritmo termina seja atingindo determinada condição de término, seja excedendo o número máximo permitido de iterações.

As operações realizadas são definidas da seguinte forma:

Cruzamento: Divide-se a população em duas metades. Para cada par formado, verifica-se se vai ou não ocorrer cruzamento, com probabilidade de ocorrência de $0,6$. Caso vá ocorrer cruzamento, determina-se para cada bit dos cromossomas, com probabilidade $0,5$, se esse bit será trocado ou não. Caso seja trocado, os indivíduos em questão ficam cada um com o bit correspondente do outro indivíduo.

Mutação: Determina-se para cada indivíduo se o mesmo sofrerá ou não mutação, com probabilidade igual a $0,02$. Caso vá ocorrer mutação, escolhe-se com um gene (com igual probabilidade para todos os genes) que será trocado por seu complemento.

Avaliação: Cada indivíduo é retornado à codificação no formato de um vetor de parâmetros reais, e então avaliado na função objetivo.

Função de Ajuste: A função objetivo é injetada na *função de ajuste*, dada por (8.3), sendo obtido para cada indivíduo um valor de função de ajuste. Foi adotado um valor de $\gamma = 1,8$.

Seleção: É realizada uma seleção de N indivíduos dentre os N indivíduos existentes, sendo que cada indivíduo pode ser selecionado mais de uma vez. A probabilidade de um indivíduo ser selecionado a cada vez é igual ao valor da fração de sua função de ajuste em relação à soma das funções de ajuste de todos os indivíduos.

Elitização: Caso o melhor indivíduo não tenha sido selecionado para a nova população, ele é nela introduzido, com a exclusão de um elemento qualquer, escolhido aleatoriamente.

Seja J o vetor das avaliações da função objetivo para os N indivíduos da população. A equação da função de ajuste (FT) é dada por:

$$\bar{J} = \text{média}(J)$$

$$J_M = \max(J)$$

$$J_m = \min(J)$$

$$v = \frac{(\gamma J_m - J_M)}{(\gamma - 1)}$$

$$
J_m \geq v \Rightarrow \begin{cases} \alpha = \bar{J}\dfrac{(\gamma - 1)}{(J_M - \bar{J})} \\ \\ \beta = \bar{J}\dfrac{(J_M - \gamma\bar{J})}{(J_M - \bar{J})} \end{cases}
$$

$$
J_m < v \Rightarrow \begin{cases} \alpha = \dfrac{\bar{J}}{(J_M - J_m)} \\ \\ \beta = -\dfrac{\bar{J}J_m}{(J_M - J_m)} \end{cases} \tag{8.3}
$$

$$
FT = \alpha J + \beta
$$

Exceto pelo formato da função de ajuste (8.3) (que foi proposta em (Goldberg 1989)), o restante do algoritmo é constituído de operações-padrão, encontradas com grande freqüência na literatura. Esse algoritmo, conforme mostrado acima, foi utilizado em (Takahashi, Peres & Ferreira 1997).

8.3.2 Algoritmo Genético - Codificação Real - Polarizado

Outro algoritmo genético, o qual foi proposto e vem sendo pesquisado pelo presente autor, com bons resultados, é também apresentado a seguir: o *Algoritmo Genético Real-Polarizado* (AG-RP).

O algoritmo AG-RP adota uma codificação real dos parâmetros de otimização, e realiza ainda a operação de "cruzamento polarizado", que produz como resultado da operação de cruzamento entre dois indivíduos (ancestrais) dois novos indivíduos, sendo que um desses tem maior probabilidade de se encontrar próximo ao ancestral com melhor valor da função objetivo.

Algoritmo Genético Real Polarizado (AG-RP)

- Cada parâmetro de projeto é descrito por uma variável real, sendo o conjunto de parâmetros armazenado em um vetor no espaço $\mathbb{R}^n$. Cada indivíduo corresponde a um vetor nesse espaço.

- Existe uma faixa admissível para cada um dos parâmetros (ou seja, para cada coordenada do vetor de parâmetros), dentro da qual estarão localizados os respectivos componentes de todos os indivíduos.
- O algoritmo se inicia com a geração aleatória de um número N, (usualmente grande) de vetores (indivíduos) dentro das faixas admissíveis.
- São realizadas em seqüência as operações de: *cruzamento*, *mutação*, *avaliação*, *cálculo da função de ajuste* ("fitness function"), *seleção* e *elitização*, sendo gerada nova população com mesmo número de indivíduos.
- O algoritmo termina seja atingindo determinada condição de término, seja excedendo o número máximo permitido de iterações.

As operações realizadas são definidas da seguinte forma:

Cruzamento: Divide-se a população em duas metades. Para cada par formado, verifica-se se vai ou não ocorrer cruzamento, com probabilidade de ocorrência de $0,5$. Caso vá ocorrer cruzamento, são gerados dois novos indivíduos segundo a lei:

$$x_g = \alpha x_1 + (1-\alpha)x_2$$

$$-0,1 < \alpha < 1,1$$

sendo x_g o novo indivíduo gerado, x_1 e x_2 os indivíduos ancestrais. Deve-se observar neste caso a restrição de que:

$$J(x_2) < J(x_1)$$

sendo $J(\cdot)$ a função objetivo a ser minimizada. Para a geração de α, verifica-se se o cruzamento será *polarizado* ou *não-polarizado*, sendo que a probabilidade de ser polarizado é de 0,3. Caso não seja polarizado, adota-se α com distribuição uniforme de probabilidade dentro do intervalo de valores possíveis para ambos os novos indivíduos gerados. Caso seja polarizado, para um dos novos indivíduos escolhe-se:

$$\alpha = 1,4\beta_1\beta_2 - 0,2 \tag{8.4}$$

sendo β_1 e β_2 escolhidas aleatoriamente e independentemente, com distribuição de probabilidade uniforme no intervalo $[0, 1]$. O outro indivíduo sempre será escolhido sem polarização. Cada novo indivíduo gerado está portanto necessariamente sobre o segmento de reta que contém x_1 a x_2, com extremos localizados de forma a ultrapassar em $0, 1$ os ancestrais. Esta operação está mostrada, em duas dimensões, na figura 8.1. Sendo definida dessa forma a operação de cruzamento,

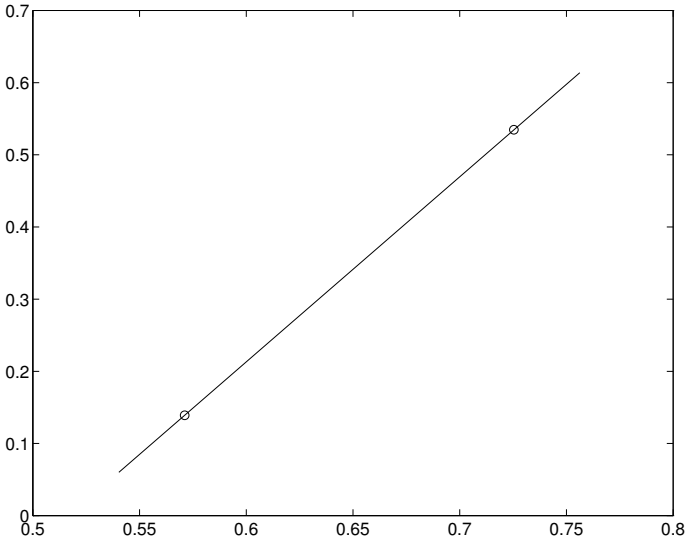


Figura 8.1: Operação de cruzamento no AG Real. Os indivíduos ancestrais são marcados com (o). Os novos indivíduos gerados podem estar em qualquer ponto sobre o segmento de reta representado na figura, com equiprobabilidade.

torna-se possível que os indivíduos gerados estejam localizados fora das faixas admissíveis de parâmetros. Caso isso ocorra, ainda é realizada uma operação de *reflexão* do indivíduo para o interior da região admissível. Essa operação é definida como:

$$x_r = x_L + |x - x_L|$$

para reflexão no limite inferior, sendo x o indivíduo que violava a restrição, x_r o resultado da reflexão, e x_L o vetor de limites inferiores.

Para a reflexão no limite superior, a operação é definida por:

$$x_r = x_U - |x_U - x|$$

para x_U o vetor de limites superiores, e as demais variáveis com mesmo significado que anteriormente. A operação de *reflexão* é mostrada, em duas dimensões, na figura 8.2.

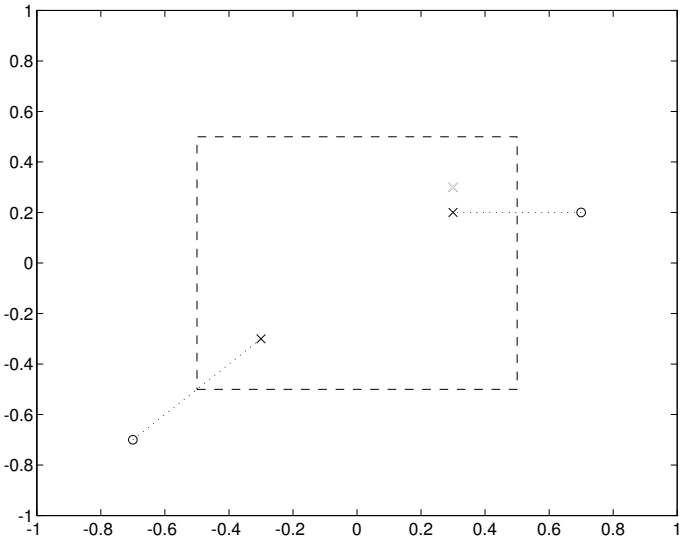


Figura 8.2: Operação de reflexão no AG-RP. O indivíduo inicial é marcado com (o), e o indivíduo resultante é marcado com (x).

Mutação: Determina-se para cada indivíduo se o mesmo sofrerá ou não mutação, com probabilidade igual a $0,02$. Caso vá ocorrer mutação, é somado ao indivíduo x um vetor δ cujas componentes são dadas por:

$$\delta_i = 0,05\beta_i(x_R)_i$$

sendo β_i um número aleatório com distribuição gaussiana, média zero e variância um, e x_R o vetor de diferença entre os máximos e mínimos dos parâmetros:

$$x_R = (x_U - x_L)$$

Avaliação: Cada indivíduo é avaliado na função objetivo.

Função de Ajuste: A função objetivo é injetada na *função de ajuste*, dada por (8.3), sendo obtido para cada indivíduo um valor de função de ajuste. Foi adotado um valor de $\gamma = 1,8$.

Seleção: É realizada uma seleção de N indivíduos dentre os N indivíduos existentes, sendo que cada indivíduo pode ser selecionado mais de uma vez. A probabilidade de um indivíduo ser selecionado a cada vez é igual ao valor da fração de sua função de ajuste em relação à soma das funções de ajuste de todos os indivíduos.

Elitização: Caso o melhor indivíduo não tenha sido selecionado para a nova população, ele é nela introduzido, com a exclusão de um elemento qualquer, escolhido aleatoriamente.

A operação de *cruzamento polarizado* imita uma busca seguindo a direção de uma "tendência" (com uma informação semelhante à fornecida pelo gradiente), o que não é feito por nenhum operador genético convencional, ao mesmo tempo em que mantém a vantagem do AG de avaliar apenas a função objetivo (sem nenhum cálculo de derivadas). No caso de ancestrais localizados próximos um ao outro, é realizado um passo semelhante ao do algoritmo do gradiente. Isso acelera a convergência local para o ótimo. No caso de indivíduos que se encontram distantes (possivelmente em bacias de atração diferentes), a operação pode ser interpretada como a utilização de uma informação de tendência de longa distância. A utilização de tal tipo de informação não possui nenhum tipo de correlato nem em outros operadores genéticos nem em algoritmos determinísticos.

O AG-RP foi utilizado nos trabalhos de pós-graduação (Meneguim 1999, Miranda 2000), e no artigo (Takahashi, Ramirez, Vasconcelos & Saldanha 2001), e teve seu desempenho avaliado em detalhe no artigo (Takahashi, Vasconcelos, Ramirez & Krahenbuhl 2003). Esse algoritmo é ainda a base para a formulação do Algoritmo Genético Multiobjetivo, que será apresentado no Capítulo 12.

8.4 Sobre a Estrutura do AG-B e do AG-RP

Nesta seção[1], uma série de estudos são realizados, na tentativa de elucidar quais são os mecanismos segundo os quais os algoritmos AG-B e AG-RP produzem a convergência de suas populações para os pontos de ótimo de problemas de otimização.

Para o propósito de testar o comportamento dos algoritmos analisados, foram empregadas duas funções-padrão, uma função quadrática convexa e a função Rastrigin. A primeira se presta a simular o comportamento que seria esperado em regiões próximas a mínimos locais de funções diferenciáveis. A segunda consiste em uma função multimodal, que serve para avaliar o comportamento do algoritmo diante da multiplicidade de mínimos locais. A expressão dessas funções é:

Função Quadrática:

$$\begin{aligned} f &= x' * Q * x \\ Q &> \mathbf{0} \end{aligned} \tag{8.5}$$

Função Rastrigin:

$$f = \sum_{i=1}^{n} (x_i^2 - 10\,\cos(2\pi x_i)) \tag{8.6}$$

Os testes são inicialmente realizados em um espaço de dimensão 2, de forma que é possível a visualização gráfica direta dos resultados (note-se que a informação relevante, neste caso, é a distribuição da população de indivíduos gerados no espaço de soluções). Para validar os resultados, são realizados testes também em espaços de dimensão maior.

8.4.1 Resultados para o AG-B

Função Quadrática

O AG-B foi executado para populações de 100 indivíduos, e 70 iterações (gerações), para a função quadrática acima definida com duas variáveis, sendo

[1]Os resultados de avaliações apresentados nesta seção foram mostrados na dissertação de mestrado (Meneguim 1999).

a matriz Q dada por:

$$Q = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 3 \end{bmatrix}$$

Foi considerando o espaço de soluções inicial com ambas as variáveis entre -1.0 e $+1.0$. Os resultados da execução do AG-B são mostrados na figura 8.3.

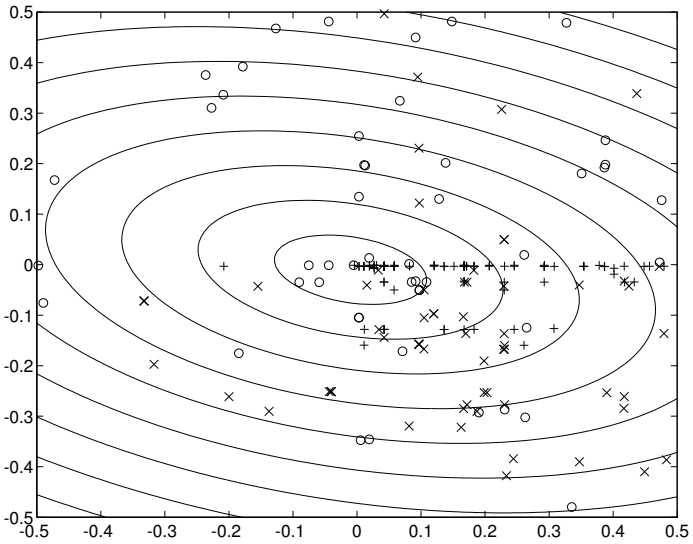


Figura 8.3: Resultado da aplicação do AG-B à função quadrática de duas dimensões. São mostradas as curvas de nível da função, e a população de soluções nas iterações 10 (o), 30 (x) e 70 (+).

Pode-se observar um curioso fenômeno: à medida em que prosseguem as iterações, a população tende a se concentrar segundo determinados eixos, paralelos aos eixos coordenados das variáveis de otimização.

Para testar a conjectura de que tal concentração da população ocorre efetivamente segundo eixos paralelos aos eixos coordenados das variáveis de otimização, foi realizada a rotação da mesma função objetivo quadrática, e realizada nova otimização com o AG-B. Para relizar tal rotação, foi adotada em lugar da matriz Q anteriormente definida a matriz:

$$Q = \begin{bmatrix} 3 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

A otimização foi realizada utilizando as mesmas condições definidas para o experimento anterior. Os resultados são mostrados na figura 8.4.

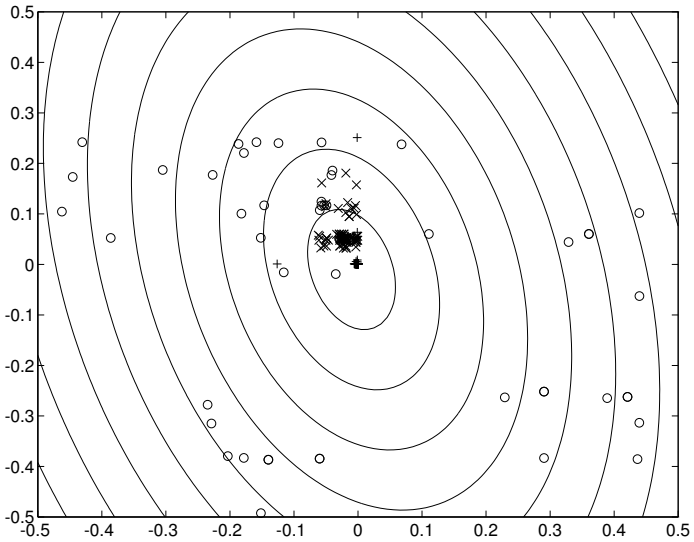


Figura 8.4: Resultado da aplicação do AG-B à função quadrática de duas dimensões com rotação de eixos. São mostradas as curvas de nível da função, e a população de soluções nas iterações 10 (o), 30 (x) e 70 (+).

Nesta figura, é necessária a visualização de um detalhe próximo ao ponto de ótimo, para que seja possível discernir a distribuição da população final. Isso é mostrado na figura 8.5.

Foi a seguir definida mais uma função quadrática, agora com curvas de nível que são elipses com semi-eixos paralelos aos eixos coordenados das variáveis de otimização. Isso é obtido com:

$$Q = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

Foi realizada novamente a otimização utilizando o AG-B nas mesmas condições anteriores. O resultado é mostrado na figura 8.6.

O fenômeno de concentração da população segundo eixos paralelos aos eixos coordenados das variáveis de otimização se repete em ambos os casos. Dessa forma, fica corroborada a hipótese que as direções segundo as quais a

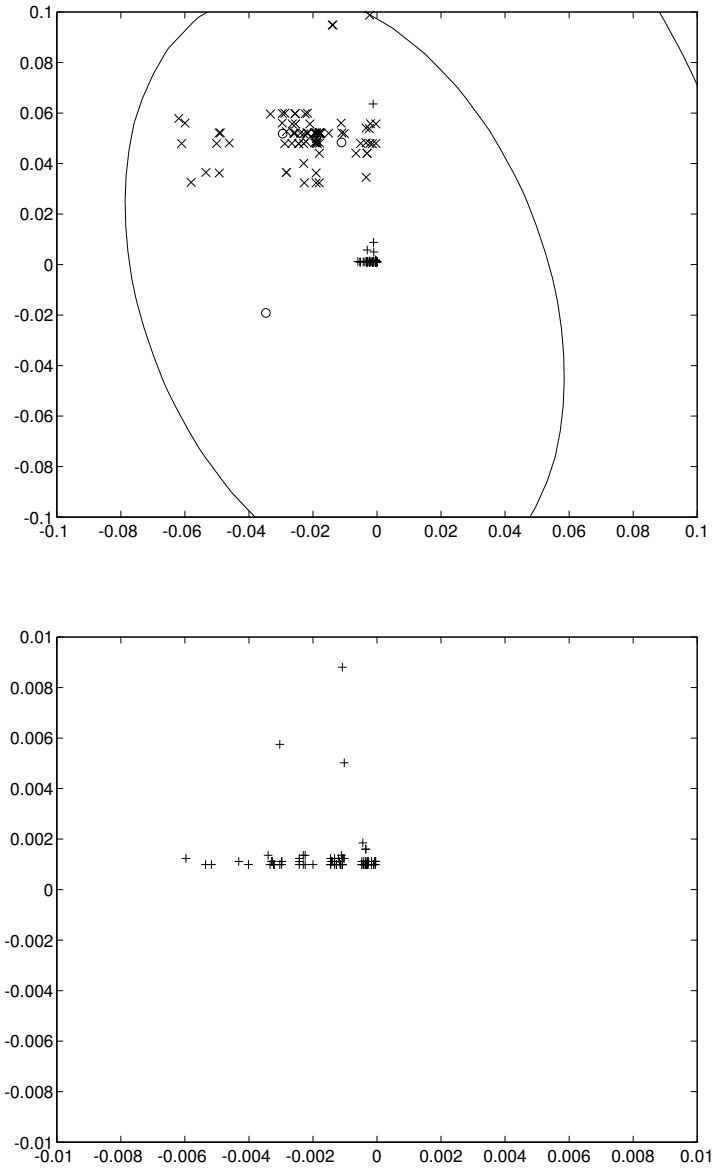


Figura 8.5: Detalhe ampliado da figura anterior. Os pontos são marcados segundo a geração a que pertencem: 10 (o), 30 (x) e 70 (+).

população se aglutina *não dependem* da função objetivo, dependendo apenas das coordenadas utilizadas para representar tal função.

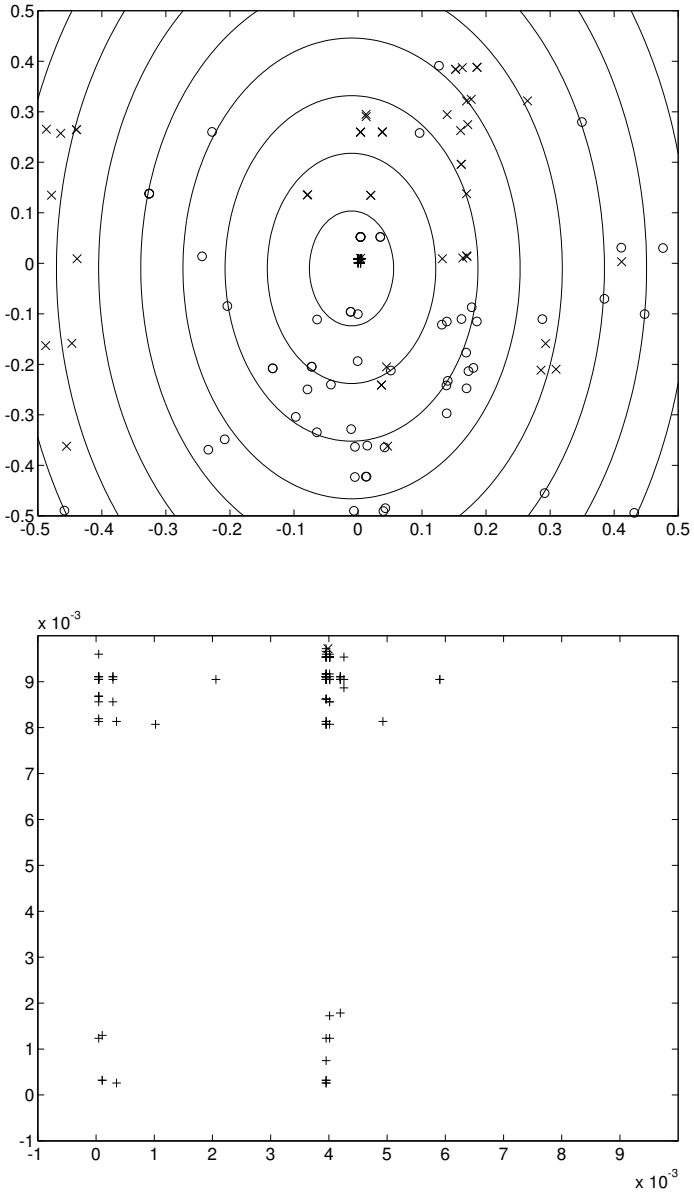


Figura 8.6: Resultado da aplicação do AG-B à função quadrática de duas dimensões com semi-eixos paralelos aos eixos coordenados. São mostradas as curvas de nível da função, e a população de soluções nas iterações 10 (o), 30 (x) e 70 (+). No quadro de baixo, é mostrado um detalhe da figura de cima.

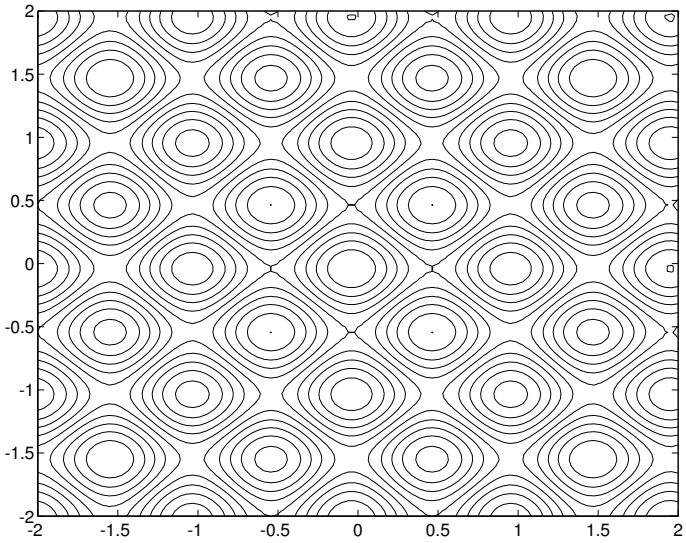


Figura 8.7: Curvas de nível da função Rastrigin.

Para verificar a dependência desse fenômeno em relação à natureza da função objetivo, é realizado um experimento com a função Rastrigin acima definida, em duas dimensões. As curvas de nível dessa função estão mostradas na figura 8.7. Os limites das variáveis são definidos em -4 e $+4$, sendo utilizadas populações de 100 indivíduos e executadas 70 iterações.

Os resultados desse experimento estão mostrados na figura 8.8. No gráfico do alto, pode-se perceber por exemplo diversos indivíduos da geração 10 localizados próximos do eixo $x_1 = -1$, para valores diversos da variável x_2. É significativo o fato de que isto ocorre para indivíduos localizados em regiões de atração de mínimos locais diferentes, o que representa mais um elemento a indicar que esta ordenação da população não depende realmente da estrutura da função objetivo, sendo causada pela estrutura do próprio algoritmo de otimização. No gráfico de baixo, são visualizados indivíduos pertencentes às gerações 30 e 70, todos localizados em uma região de atração de um único mínimo local. O padrão de ordenamento dos indivíduos novamente se verifica aqui.

Pode-se notar, no conjunto de testes realizados, a ocorrência de um fenômeno de concentração da população de indivíduos em torno de alguns eixos (com valores de coordenadas aproximadamente constantes). Tal fenômeno

não foi particularidade de uma função específica, tendo ocorrido para todos os casos, mesmo sendo as funções de naturezas bastante diversas.

Para tomar um exemplo em 3 dimensões, foi executado ainda o AG-B na otimização de uma norma $\mathcal{H}_2$ de um sistema dinâmico de ordem 3. As variáveis de otimização são os três valores de ganhos da matriz do controlador por realimentação estática de estados. Para não introduzir neste momento informações que desviariam o leitor do cerne do problema aqui estudado, limitamo-nos a informar que o problema em questão possui características bastante distintas dos anteriores. O detalhamento do problema aqui apresentado como exemplo será apresentado no capítulo 4 desta dissertação. A figura 8.9 mostra o espalhamento da população correspondente à geração 70 obtida pelo AG-B, num gráfico tridimensional.

O mesmo padrão de aglutinação das populações segundo eixos paralelos aos eixos coordenados é verificado nesse gráfico. Existe aglutinação segundo todas as três direções coordenadas.

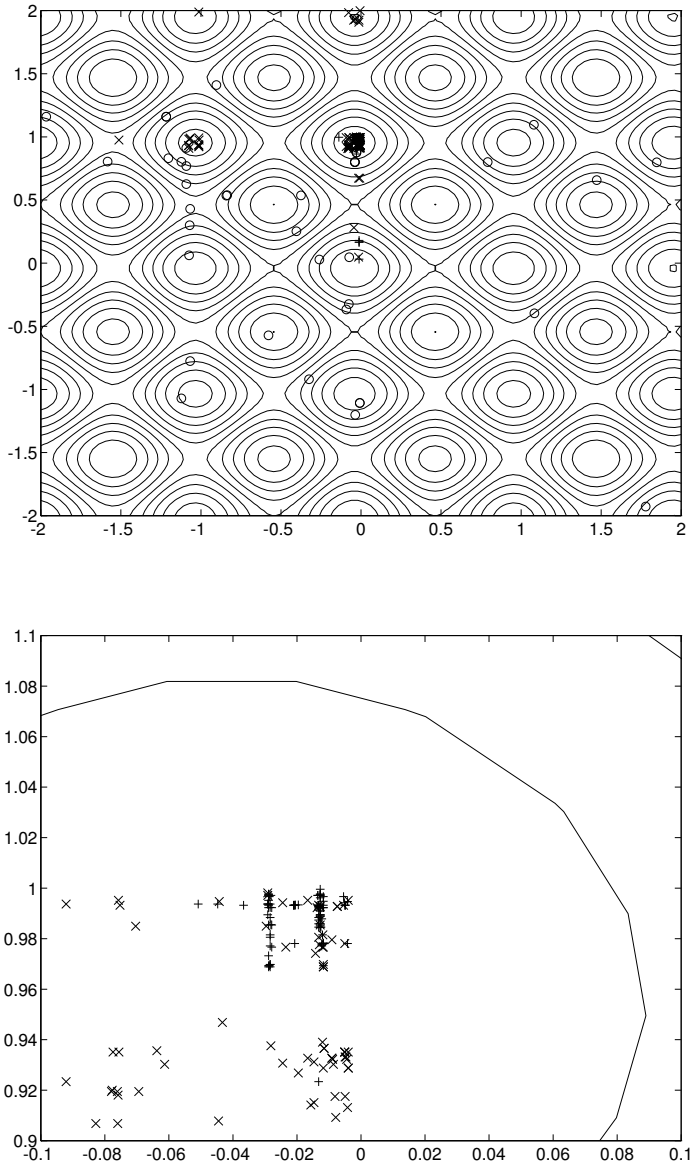


Figura 8.8: Resultado da aplicação do AG-B à função Rastrigin de duas dimensões. São mostradas as curvas de nível da função, e a população de soluções nas iterações 10 (o), 30 (x) e 70 (+). No quadro de baixo, é mostrado um detalhe da figura de cima.

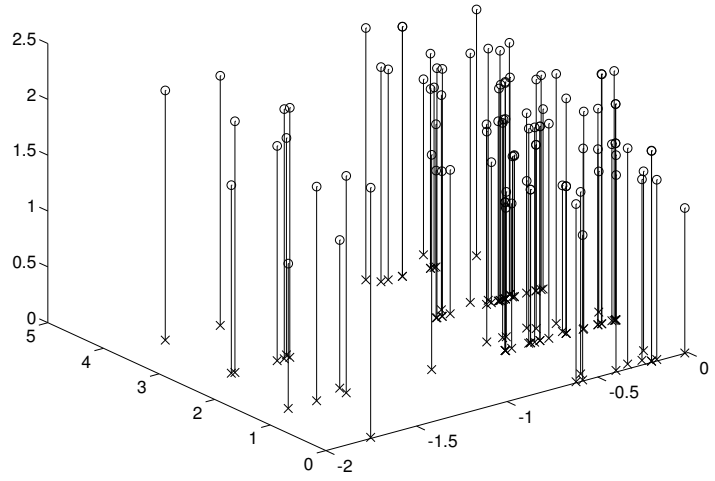


Figura 8.9: Resultado da aplicação do AG-B à otimização da norma $\mathcal{H}_2$ de um sistema dinâmico. O gráfico tridimensional mostra as três variáveis do controlador por realimentação completa de estados (o) e a projeção de cada elemento desses no plano $K_3 = 0$ (x).

Parece possível, com base nos dados levantados, propor a seguinte conjectura:

Conjectura 8.1 *À medida em que ocorre a sequência de gerações em um algoritmo genético com codificação binária, as sucessivas populações tendem a se distribuir segundo padrões que vão se aproximando de eixos paralelos aos eixos em que estão representadas as variáveis de otimização.* ∘

Se essa conjectura estiver correta, será possível concluir que o AG-B encontra condições favoráveis à sua execução no caso de funções com parâmetros aproximadamente desacoplados, isto é, funções em que a otimização de cada um dos parâmetros possa ser feita independentemente da otimização dos demais. Sendo particularmente adequado para tal classe de funções, o AG-B certamente tenderia a ter um desempenho pior quando essa hipótese de desacoplamento de parâmetros não fosse verdadeira. Nesse caso, algoritmos que não fizessem buscas privilegiando as direções paralelas aos eixos coordenados tenderiam a ter um comportamento melhor.

Na tentativa de isolar as causas para o fenômeno observado, foram realizados dois testes adicionais: o AG-B foi executado, para otimização da mesma funçã quadràtica de duas variáveis inicialmente empregada, com a matriz Q dada por:

$$Q = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 3 \end{bmatrix}$$

O AG-B foi executado para uma população de 100 indivíduos e 70 iterações (gerações), sendo considerado o espaço de soluções inicial com ambas as variáveis entre -1.0 e $+1.0$. Em uma das execuções, foi apenas desabilitada a operação de mutação, sendo os resultados desse experimento mostrados na figura 8.10.

Na outra execução foi desabilitada a operação de cruzamento, sendo também aumentada a probabilidade de mutação para 0.8. Os resultados desse experimento são mostrados na figura 8.11.

Pode-se verificar que em ambos os casos continuou a ocorrer o fenômeno de concentração da população segundo eixos paralelos às coordenadas. Parece que a causa do mesmo está associada à própria realização de operações sobre a representação binária dos parâmetros, que implica na transmissão das características assim codificadas de um único indivíduo para vários outros.

A operação de "cruzamento" seria então, nesse caso, responsável pela produção de novos indivíduos que estarão provavelmente localizados em posições

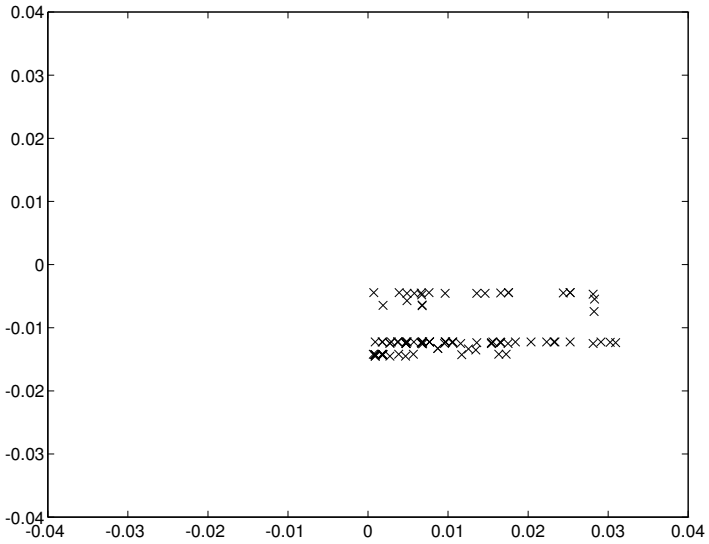


Figura 8.10: Resultado da aplicação do AG-B à otimização da função quadrática, com a operação de mutação desabilitada.

coordenadas iguais às de "ancestrais" (em pelo menos uma coordenada). Como a operação de "seleção" faz o papel de reduzir a diversidade de indivíduos, possivelmente replicando várias cópias de indivíduos com "melhores características", existe uma grande probabilidade de surgimento de grupos de indivíduos compartilhando determinadas coordenadas. Por outro lado a mutação acima definida iria sempre produzir necessariamente novos indivíduos que possuem pelo menos uma coordenada igual à de um indivíduo preexistente (que por sua vez possivelmente possui réplicas), causando o mesmo efeito.

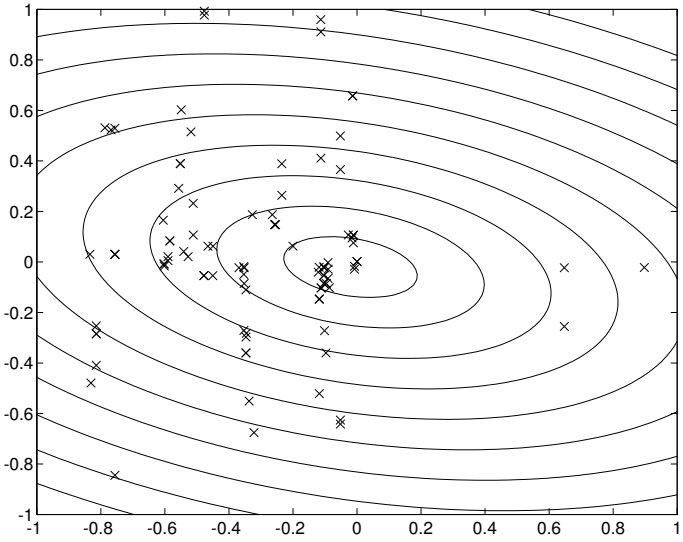


Figura 8.11: Resultado da aplicação do AG-B à otimização da função quadrática, com a operação de cruzamento desabilitada.

8.4.2 Resultados para o AG-RP

O algoritmo AG-RP foi desenvolvido ao longo do ano de 1998 exatamente a partir da observação do fenômeno verificado no algoritmo AG-B, de existência de uma premissa implícita em sua formulação de desacoplamento de variáveis. Tal premissa implícita seria indesejável, em um algoritmo que tivesse o propósito de otimizar funções não-lineares genéricas. A observação de que o problema estaria associado intrinsecamente ao fato dos operadores genéticos do AG-B serem construídos com base numa codificação binária levou à tentativa de se trabalhar com uma codificação real. Considerações de velocidade de convergência e de robustez (confiabilidade) levaram à concepção do operador de cruzamento real-polarizado, que veio a ser um elemento inovador que determinou as propriedades mais importantes desse algoritmo. A seguir é apresentada a bateria de testes que foi originalmente empregada para a avaliação do desempenho do AG-RP.

Função Quadrática

O AG-RP foi executado para populações de 100 indivíduos, e 70 iterações (gerações), para a função quadrática acima definida com duas variáveis, sendo a matriz Q dada por:

$$Q = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 3 \end{bmatrix}$$

Foi considerando o espaço de soluções inicial com ambas as variáveis entre -1.0 e $+1.0$. Os resultados da execução do AG-RP são mostrados na figura 8.12.

Pode-se observar que neste caso a população tende a ter distribuição não exatamente uniforme, porém ainda aproximadamente aleatória, como no caso do AG-R. Também aqui a distribuição dos indivíduos tende a acompanhar as curvas de nível da função (isso é particularmente visível na população da geração 10). Parece novamente que a distribuição dos indivíduos está fortemente correlacionada com a estrutura geométrica da função objetivo que está sendo otimizada, sendo aparentemente independente das coordenadas em que essa função se encontra representada. Existem portanto indícios de que seria plausível afirmar que a população gerada fornece uma descrição da geometria da função objetivo na região em que essa população está localizada, podendo ser utilizada para análises aproximadas dessa geometria.

Função Rastrigin

Foi também realizado um experimento com a função Rastrigin acima definida, em duas dimensões. Os limites das variáveis são definidos em -4 e $+4$, sendo utilizadas populações de 100 indivíduos e executadas 70 iterações.

Os resultados desse experimento estão mostrados na figura 8.13. Também neste caso se confirma a tendência das populações consecutivas acompanharem a geometria das curvas de nível, ocupando curvas correspondentes a níveis menores (menor valor da função objetivo) à medida em que avançam as gerações. Não fica aparente, também neste caso, nenhuma dependência da distribuição de indivíduos nem relação aos eixos coordenados nem em relação à estrutura do próprio algoritmo de otimização. Deve-se observar que o AG-RP convergiu para o ótimo global da função.

Desses experimentos, pode-se inferir que a propriedade de dispersão espacial aproximadamente uniforme das populações aparentemente se preservou. Pode-se inferir que é possível utilizar o AG-RP, nas mesmas condições que o AG-R, para análise da geometria da função objetivo em regiões próximas ao ponto de mínimo. A questão relevante que estabelece a escolha entre o AG-RP e o AG-R é portanto a comparação entre as propriedades de convergência dos dois algoritmos.

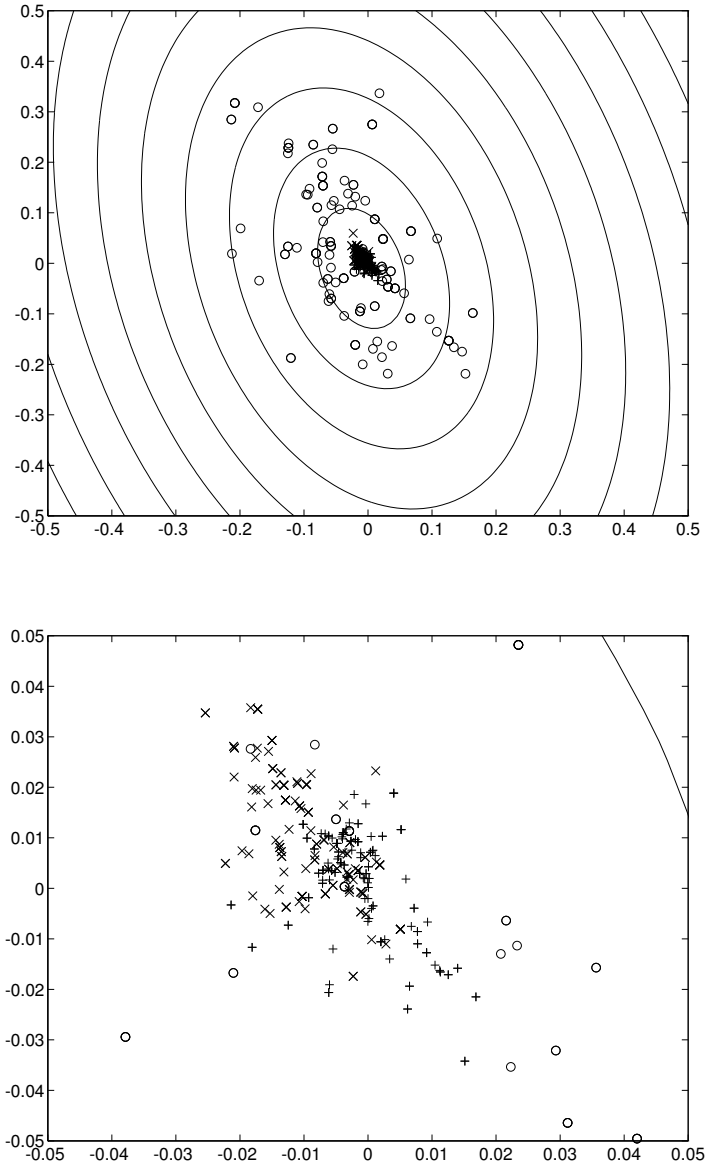


Figura 8.12: Resultado da aplicação do AG-RP à função quadrática de duas dimensões. São mostradas as curvas de nível da função, e a população de soluções nas iterações 10 (o), 30 (x) e 70 (+). No quadro de baixo, é mostrado um detalhe do gráfico acima, em região próxima ao ponto de ótimo.

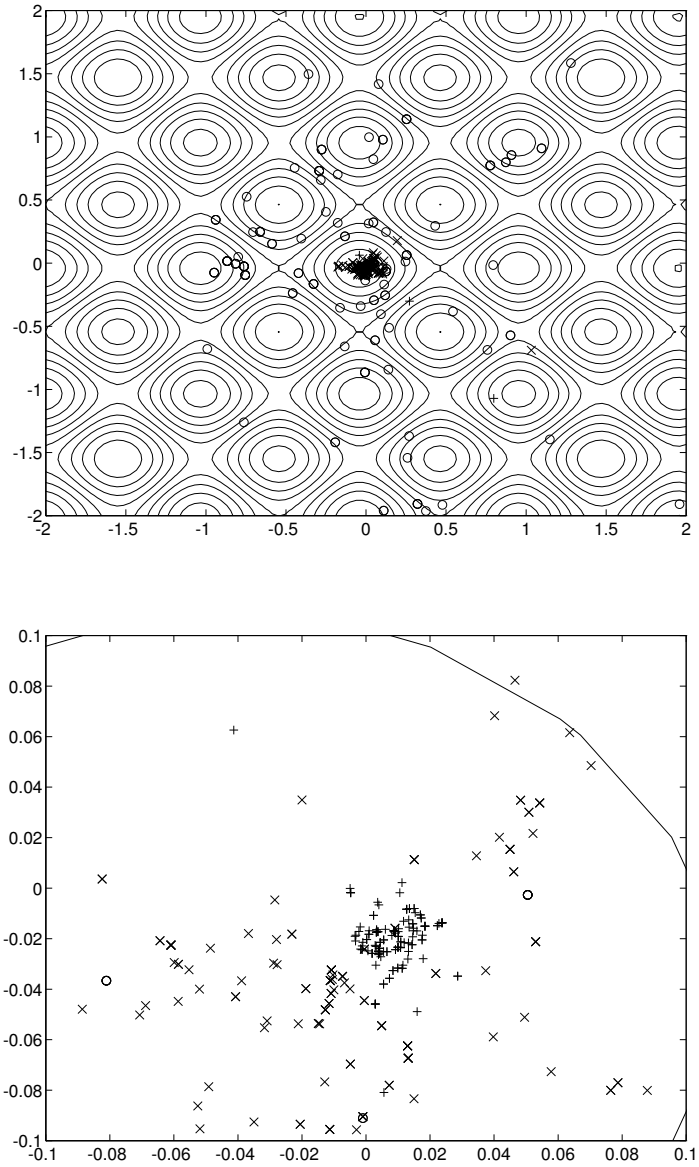


Figura 8.13: Resultado da aplicação do AG-RP à função Rastrigin de duas dimensões. São mostradas as curvas de nível da função, e a população de soluções nas iterações 10 (o), 30 (x) e 70 (+). No quadro de baixo, é mostrado um detalhe da figura de cima.

8.4.3 Teste das Propriedades de Convergência

Os algoritmos AG-B e AG-RP foram testados finalmente para a avaliação de suas propriedades de convergência. Para se testar o exato papel do operador de *cruzamento real polarizado*, foi feito um teste em que esse operador se encontrava habilitado e outro em que este se encontrava desabilitado[2], dentro do AG-RP. Para designar o caso em que o operador real-polarizado se encontrava desabilitado, o algoritmo resutante será aqui designado como *Algoritmo Genético Real*, e referido pela sigla AG-R.

Por propriedades de convergência entende-se: (i) o fato de um algoritmo convergir ou não para o mínimo global de uma função, e (ii) o número de iterações necessárias para a convergência, nos casos em que esta ocorre. Para realizar a comparação das propriedades de convergência dos algoritmos, foram empregadas as mesmas funções quadrática e Rastrigin anteriormente definidas, a primeira em duas e cinco dimensões, a última apenas em duas dimensões.

Para que a comparação fosse realizada em bases comuns para os três algoritmos, foi realizada uma bateria de testes sendo cada algoritmo empregado para otimizar a mesma função diversas vezes. A seqüência de testes envolveu:

1. gerar uma população inicial;
2. utilizar essa mesma população para inicializar os três algoritmos;
3. para cada algoritmo, determinar se houve convergência para o ótimo global, e qual o número de iterações necessário para se atingir a convergência.

Essa seqüência foi realizada diversas vezes para cada caso estudado, sendo gerado um conjunto de dados que se pretende que permita comparar a capacidade de convergência dos algoritmos, em uma base estatística.

Como critério de convergência, foi exigida uma distância δ menor que ou igual a $0,001$, para os problemas de duas dimensões, e $0,01$ para os problemas de cinco dimensões, em relação ao ponto de ótimo x^* (que corresponde em todos os casos à origem das coordenadas), sendo tal distância definida como:

$$\delta(x) = \max_i |x_i - x_i^*| = \max_i |x_i| \tag{8.7}$$

[2]Desabilitar o operador real-polarizado significa realizar sempre apenas o cruzamento não-polarizado, com α sendo escolhido segundo uma probabilidade uniforme; ver esse conceito na definição do **cruzamento**.

Como critério de não-convergência, foi estabelecido que o algoritmo não teria convergido caso tivessem ocorrido 1000 iterações (gerações) sem que fosse satisfeito o critério de convergência.

Os resultados dos testes realizados são apresentados a seguir.

Função Quadrática 2-D

A função quadrática em duas dimensões é dada por:

$$J(x) = x'Qx \tag{8.8}$$

sendo a matriz Q dada por:

$$Q = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 3 \end{bmatrix}$$

Foi empregada uma população de 100 indivíduos. A região de busca foi definida pelos extremos:

$$X_{ub} = \begin{bmatrix} 10 \\ 10 \end{bmatrix} \quad X_{lb} = \begin{bmatrix} -10 \\ -10 \end{bmatrix}$$

Foram realizadas 30 execuções de cada algoritmo de otimização. A tabela 8.1 apresenta o número de iterações necessário para cada algoritmo convergir em cada caso executado.

Os dados da tabela 8.1 mostram que para o conjunto de execuções dos algoritmos, apenas o algoritmo genético binário não convergiu, uma única vez. Os algoritmos genéticos reais convergiram em todos os casos. Excluindo o caso em que o AG-B não convergiu, são obtidas as seguintes médias do número de iterações para a convergência:

AG-B: $155,52$ iterações

AG-R: $154,17$ iterações

AG-RP: $53,97$ iterações

Esses resultados sugerem que, uma vez que a população já se encontra em uma região localmente convexa próxima ao ponto de mínimo, o AG-B e o AG-R possuem propriedades de convergência aproximadamente equivalentes. Já o AG-RP, nessas circunstâncias, converge muito mais rapidamente que os anteriores.

AG-B	AG-R	AG-RP
240	46	28
182	48	66
75	174	82
116	25	76
110	209	51
51	89	72
71	17	49
84	123	89
183	124	54
77	171	35
92	113	78
1001	182	50
136	113	62
101	153	33
97	374	56
104	117	53
67	221	92
897	303	36
659	174	26
132	51	56
91	431	41
93	84	44
128	21	45
98	114	42
75	196	51
43	348	41
159	155	45
86	64	51
116	92	63
147	321	48

Tabela 8.1: Número de iterações que foram necessárias para cada algoritmo atingir a convergência, para a função quadrática de duas dimensões e população de 100 indivíduos. As colunas correspondem aos dados de: AG-B (algoritmo genético binário), AG-R (algoritmo genético real) e AG-RP (algoritmo genético real polarizado). Cada linha corresponde à execução dos três algoritmos, para uma mesma população inicial gerada aleatoriamente.

Função Quadrática 5-D

Para estudar a interferência da dimensionalidade do problema em questão, foi utilizada uma função quadrática de cinco variáveis. Tal função é dada por:

$$J(x) = x'Qx \tag{8.9}$$

sendo a matriz Q dada por:

$$Q = \begin{bmatrix} 1.6695 & -0.2023 & 0.8054 & 0.4637 & 0.3353 \\ -0.2023 & 2.5152 & -0.0935 & 0.6921 & -0.0628 \\ 0.8054 & -0.0935 & 2.9959 & 0.5379 & 0.5159 \\ 0.4637 & 0.6921 & 0.5379 & 1.8253 & 0.3957 \\ 0.3353 & -0.0628 & 0.5159 & 0.3957 & 2.5741 \end{bmatrix}$$

Essa última matriz possui autovalores iguais a:

$$\sigma = \begin{Bmatrix} 2.8900 & 2.2500 & 1.4399 & 1.0000 & 4.0000 \end{Bmatrix}$$

A otimização dessa função é apresentada na tabela 8.2. Os dados da tabela 8.2 mostram que, também neste caso, para o conjunto de execuções dos algoritmos, apenas o algoritmo genético binário não convergiu, uma única vez. Os algoritmos genéticos reais convergiram em todos os casos. Excluindo o caso em que o AG-B não convergiu, são obtidas as seguintes médias do número de iterações para a convergência:

AG-B: $385,03$ iterações

AG-R: $244,79$ iterações

AG-RP: $183,45$ iterações

Esses resultados sugerem que, uma vez que a população já se encontra em uma região localmente convexa próxima ao ponto de mínimo, o AG-R possui convergência um pouco mais acelerada que o AG-B. Já o AG-RP, também nessas circunstâncias, converge muito mais rapidamente que os anteriores.

AG-B	AG-R	AG-RP
359	256	149
512	308	155
272	306	165
591	193	203
367	237	143
294	238	199
239	264	261
459	91	158
201	246	200
230	245	144
397	204	275
234	250	118
482	236	126
587	304	212
556	387	235
567	282	169
407	289	158
309	216	208
1001	183	193
386	178	127
364	215	220
134	219	209
370	236	138
177	139	157
376	351	164
253	304	238
760	290	190
407	178	127
558	167	251
318	270	221

Tabela 8.2: Número de iterações que foram necessárias para cada algoritmo atingir a convergência, para a função quadrática de cinco dimensões e população de 40 indivíduos. As colunas correspondem aos dados de: AG-B (algoritmo genético binário), AG-R (algoritmo genético real) e AG-RP (algoritmo genético real polarizado). Cada linha corresponde à execução dos três algoritmos, para uma mesma população inicial gerada aleatoriamente.

Função Rastrigin 2-D

Para estudar um problema com características de multimodalidade, foi utilizada a função Rastrigin em duas dimensões.

Neste conjunto de testes, pode-se observar que houve um número significativo de casos em que os três algoritmos não convergiram para a solução ótima global. O desempenho dos três algoritmos foi:

AG-B: não-convergência: 17 vezes (57%)

AG-R: não-convergência: 11 vezes (37%)

AG-RP: não-convergência: 5 vezes (17%)

Esses resultados mostram que houve neste teste uma clara superioridade do AG-RP sobre os demais algoritmos, e uma pequena superioridade do AG-R sobre o AG-B.

Considerando a média de iterações despendida por cada algoritmo nos casos em que convergiu, obtêm-se os seguintes resultados:

AG-B: $143,69$ iterações

AG-R: $125,47$ iterações

AG-RP: $318,44$ iterações

Entretanto, considerando apenas os casos em que todos os três algoritmos convergiram (apenas 08 casos), obtêm-se:

AG-B: $98,38$ iterações

AG-R: $72,38$ iterações

AG-RP: $256,38$ iterações

Por alguma razão o AG-RP é o algoritmo de convergência mais lenta dentre os três na presente situação, quando ocorre convergência. O algoritmo mais veloz parece ser o AG-R, com pequena diferença de velocidade de convergência em relação ao AG-B. Entretanto, o AG-RP converge com muito maior segurança que os outros dois algoritmos.

Esse resultado talvez possa ser explicado pelo fato de os algoritmos genéticos reais possuírem uma capacidade de extrapolação embutida na operação

de cruzamento a qual, combinada com o operador de polarização, permite a consideração implícita de tendências globais (de longo alcance) da função. Essa propriedade não está presente no AG-B devido à ausência da capacidade de extrapolação, nem no AG-R, devido à total aleatoriedade da extrapolação nele realizada.

8.5 Metodologia de Avaliação da Eficiência de AG's

Existe [3] um grande número de operadores genéticos potencialmente plausíveis que poderiam dar origem a Algoritmos Genéticos que funcionassem adequadamente. Os operadores genético básicos podem ser implementados através de uma crescente variedade de fórmulas, ver, por exemplo, (Choi & Oh 2000, Vasconcelos et al. 2001); e existe ainda um número crescente de operadores adicionais, ver, por exemplo, (J. C. Potts & Yadav 1994, Sareni & Krahenbuhl 1998). Essa situação decorre do fato de que o estudo de estruturas de operadores é hoje em dia uma área de pesquisa fértil.

Infelizmente, na maioria dos casos, não existe uma justificativa analítica para a escolha de uma estrutura de operador específica. A referência (Naudts & Kallel 2000) discute algumas dificuldades associadas à tarefa de prever analiticamente o desempenho de um AG específico. Sabe-se, entretanto, que estruturas específicas de operadores, com os respectivos ajustes dos parâmetros desses operadores, deveriam ser empregados para as diferentes classes de problemas, para preservar a eficiência computacionaL do algoritmo quando aplicado a tais classes (Wolpert & Macready 1997). As escolhas de cada instância de operador e dos valores para seus parâmetros deveria ser realizada, portanto, com base em avaliações empíricas prévias (Z. Michalewicz & Stidsen 2000, Vasconcelos et al. 2001).

Não existe firmemente estabelecida, até hoje (início de 2003), nenhuma fonte de informação integrativa para guiar tal escolha pelo usuário. O tipo de informação usual que se encontra disponível na literatura cai em uma das seguintes categorias:

i. *Tutorial*: Algumas alternativas comuns de operadores são apresentadas e alguns valores para seus parâmetros são recomendados, sob a forma

[3] A discussão apresentada nesta seção segue essencialmente a referência (Takahashi, Vasconcelos, Ramirez & Krahenbuhl 2003).

de largas faixas de valores. Esse é o caso, por exemplo, de (Tanomaru 1995, K. F. Man & Kwong 1996, Johnson & Ramat-Semii 1997).

ii. *Avaliação de Algoritmo Único*: Segue o esquema básico de apresentar um novo algoritmo e avaliar esse algoritmo contra algum algoritmo que seja considerado "clássico" ou "usual". Nesta categoria se encontra, por exemplo, o trabalho (J. C. Potts & Yadav 1994).

iii. *Comparação de Operador Único*: Compara diferentes alternativas de implementação de um certo operador, mantendo fixados os demais operadores. Em geral, o objetivo é fazer a apresentação de uma nova estrutura de operador proposta. Um exemplo de trabalho nessa categoria é (Schell & Wegenkittl 2001), que estuda especificamente operadores de seleção.

iv. *Estudos em Aplicações Específicas*: Faz comparações e recomendações, em geral avaliando algoritmos distintos, especificamente relacionados com uma certa aplicação. Um exemplo de tal tipo de trabalho é (Jain & Zongker 1997).

Algumas questões sobre o quê deveria ser considerado uma "fonte de informação integrada" para a escolha de AG's são:

i. As mais recentes e complexas estruturas de operadores deveriam ser consideradas enquanto alternativas a serem analisadas, seja contra estruturas convencionais, seja umas contra as outras.

ii. Deve-se reconhecer também que alguns problemas podem apresentar forte sensibilidade ao ajuste de parâmetros, o que impede a "usabilidade" de largas faixas de valores de referência enquanto guia para o ajuste (Jain & Zongker 1997).

iii. Deve-se reconhecer também, por fim, que os diferentes operadores não são independentes entre si. Isso significa que, por exemplo, um operador de cruzamento que apresenta o melhor desempenho quando aplicado junto com determinado operador de mutação pode ser eventualmente suplantado por outro operador de cruzamento quando o operador de mutação for mudado (Belmont-Moreno 2001).

A escolha de AG's adequados para aplicações específicas é uma tarefa complexa que deveria ser estruturada de forma sistemática, com os objetivos de

permitir o uso do conhecimento mais recente para a construção de AG's específicos, ao mesmo tempo mantendo o esforço de desenvolvimento desses AG's em um patamar razoável.

8.5.1 Metodologia de Avaliação

Um dos problemas a serem abordados aqui é definido como:

Definição 8.1 (Problema de Avaliação de Algoritmos (PAA)) *Dada uma classe de problemas a serem tratados com um conjunto de operadores genéticos, determinar os melhores algoritmos, considerando tanto o critério de máxima taxa de convergência e mínima taxa de falha de convergência.* □

Como o *Problema de Avaliação de Algoritmos* é definido em termos de dois objetivos, sua solução é definida como a solução de um problema de otimização vetorial, e possui a forma de um *conjunto de Pareto*. Este conjunto é definido a partir do conceito de *dominância*: uma solução é dita ser dominada se ela é pior que alguma outra solução em pelo menos um dos objetivos, sem ser melhor em nenhum outro objetivo. Este conceito será mais precisamente discutido no próximo capítulo. O conjunto de Pareto é o conjunto que não contém nenhuma solução dominada.

A definição do outro problema a ser abordado é dada por:

Definição 8.2 (Problema de Avaliação de Novo Algoritmo (PANA)) *Dado um novo algoritmo e uma classe de problemas, comparar tal algoritmo com os melhores algoritmos oriundos do PAA para tal classe, atribuindo ao algoritmo o status de uma* solução melhorada *ou de uma* solução dominada*, possivelmente atualizando o conjunto de "melhores soluções" disponível.* □

O novo algoritmo, no PANA, deve ser comparado apenas com os algoritmos pertencentes ao conjunto de Pareto que tiver sido encontrado através do PAA. Isto significa que a base de dados de comparações pode descartar a maioria das alternativas que tiverem sido consideradas no PAA, e manter apenas as pertencentes ao conjunto de Pareto. Após a execução do PANA, três coisas podem ocorrer:

i. O novo algoritmo não é melhor que os antigos. Ele é então descartado, e o conjunto de Pareto permanece com sua composição anterior.

ii. O novo algoritmo revela-se não-dominado, mas também não domina nenhum dos antigos algoritmos que já se encontravam no conjunto de Pareto. Neste caso, o novo algoritmo é incluído no conjunto de Pareto, que cresce.

iii. O novo algoritmo domina uma ou mais soluções que se encontravam anteriormente no conjunto de Pareto. Ele é então incluído no conjunto de Pareto, e os algoritmos dominados são excluídos.

O esforço computacional no PANA é associado apenas à avaliação das funções de teste realizadas pelo novo algoritmo. O esforço associado à construção do conjunto de Pareto no PAA é re-utilizado (não é necessário executar novamente nenhum algoritmo já testado).

A seguir são discutidos em detalhe os passos da metodologia proposta.

Seleção de Funções Analíticas Representativas

A classe de problemas de interesse é possivelmente constituída de funções que não são expressas na forma de funções analíticas mas, ao contrário, são dadas por modelos de simulação que são difíceis de avaliar. Este é o caso, por exemplo, de modelos de dispositivos eletromagnéticos (K. Rashid & Freeman 2000). A questão de avaliar a usabilidade de GA's em uma classe específica de problemas, entretanto, não depende da utilização factual de alguma função da classe para avaliar o algoritmo: é necessário apenas que se utilize alguma função que preserve algumas das características das funções de interesse prático (Z. Michalewicz & Stidsen 2000). Com esse procedimento, uma função que é fácil de avaliar pode ser usada, o que torna factível executar um grande número de execuções do algoritmo para o propósito de realizar o teste. O trabalho (Z. Michalewicz & Stidsen 2000) apresenta procedimentos para a construção de funções analíticas com algumas propriedades especificadas. Outra possível maneira de se construirem modelos analíticos que apresentem as propriedades de sistemas mais complexos seria através de alguma técnica de aproximação (K. Rashid & Freeman 2000).

No presente trabalho, a *função de Rastrigin rotacionada* é empregada, com o propósito de exemplificar a metodologia aqui proposta:

$$f(x) = x^T A^T A x - 10 \left[\begin{array}{ccc} 1 & \cdots & 1 \end{array} \right] cos(2\pi A x) \tag{8.10}$$

sendo $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$, $A \in \mathbb{R}^{n \times n}$ uma matriz definida positiva e as demais variáveis tais que se mantenha a compatibilidade de dimensões. Este é um

protótipo simples de função multimodal com tendências de larga escala e coordenadas acopladas.

Base de Dados de AG's

Um conjunto de alternativas de operadores é definido na Tabela 8.4. As alternativas são essencialmente aquelas descritas na referência (Vasconcelos et al. 2001). Cada versão do AG é testada com todas as funções do conjunto representativo, com duas figuras de mérito:

i. O número médio de avaliações da função objetivo necessárias para se atingir o mínimo global; e

ii. A fração das execuções do algoritmo que termina por encontrar o mínimo global.

Cada uma dessas figuras de mérito é avaliada para um número (no caso do exemplo aqui mostrado, 100) execuções do algoritmo.

Existem 864 diferentes instâncias de AG's que podem ser formados por combinações desses parâmetros. As figuras de mérito são determinadas para cada algoritmo e para cada função representativa. Uma base de dados é então composta, com a estrutura:

i_1	i_2	$\ldots$	i_k	f	m_1	m_2

Os índices $i_1, \ldots, i_k$ apontam para diferentes alternativas, dentre as k alternativas que se encontram sob estudo. O índice f identifica uma "função representativa", e m_1 e m_2 são as resultantes figuras de mérito que vêm da aplicação do algoritmo definido por $i_1, \ldots, i_m$ à otimização da função. No caso estudado aqui, o índice i_2, por exemplo, significa o operador de mutação, e $i_2 = 3$ significa a opção de mutação do tipo "um bit por indivíduo".

Definindo m_1 e m_2 para todas as instâncias de f como as funções objetivo de um problema de análise multiobjetivo, e $I = [i_1, \ldots, i_n]$ o vetor de índices que identifica uma instância do AG, a base de dados é "podada" de acordo com a lei:

$$\begin{aligned}\mathcal{X}^* = \{I^* \mid \ \nexists I \neq I^* \text{ tal que } (m_1, f)(I) < (m_1, f)(I^*) \text{ e} \\ (m_2, f)(I) < (m_2, f)(I^*)\}\end{aligned} \tag{8.11}$$

Após tal procedimento, a base de dados do *conjunto de Pareto* resultante, $\mathcal{X}^*$, se torna muito menor que a base de dados inicial. Apenas os algoritmos *não dominados* são mantidos nesta base de dados.

A função Rastrigin rotacionada foi testada para todas as 864 instâncias de AG's diferentes. As figuras de mérito resultantes encontram-se plotadas na Figura 8.14. Essa figura mostra que, dentre os algoritmos que empregam esse conjunto de operadores "comuns", há relativamente poucos que podem ser considerados "bons algoritmos", no sentido em que por exemplo, eles venham a falhar em menos que 20% das execuções, e de que necessitem menos de 1000 avaliações da função objetivo para encontrar o ponto de mínimo.

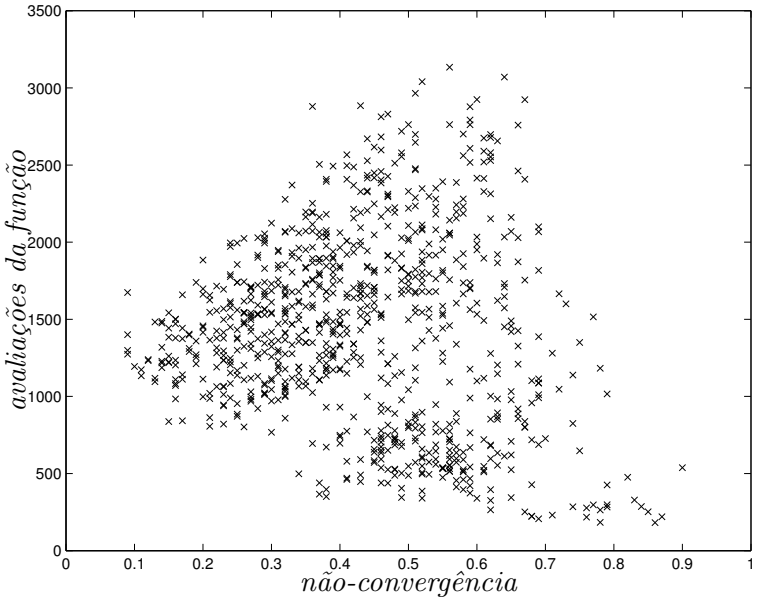


Figura 8.14: Gráfico das figuras de mérito para os algoritmos compostos por combinações de operadores "comuns": número de avaliações da função objetivo para a convergência (vertical) versus a fração de execuções que não convergem (horizontal). A função é a Rastrigin rotacionada.

Por outro lado, existe um grande número de algoritmos que podem ser muito ruins, no sentido em que, por exemplo, necessitem de mais de 2000 avaliações da função objetivo, ou falhem em mais de 50% das execuções. Este resultado mostra que, se um algoritmo é "comum", isso não significa que o mesmo pode ser considerado um bom padrão de comparação para avaliar novos algoritmos.

O conjunto de Pareto é extraído desses dados, sendo mostrado na Figura 8.15. Esse conjunto é constituído de apenas 25 algoritmos que são "não-dominados". Esses 25 algoritmos são suficientes para a comparação com qualquer novo algoritmo sobre a mesma função de teste.

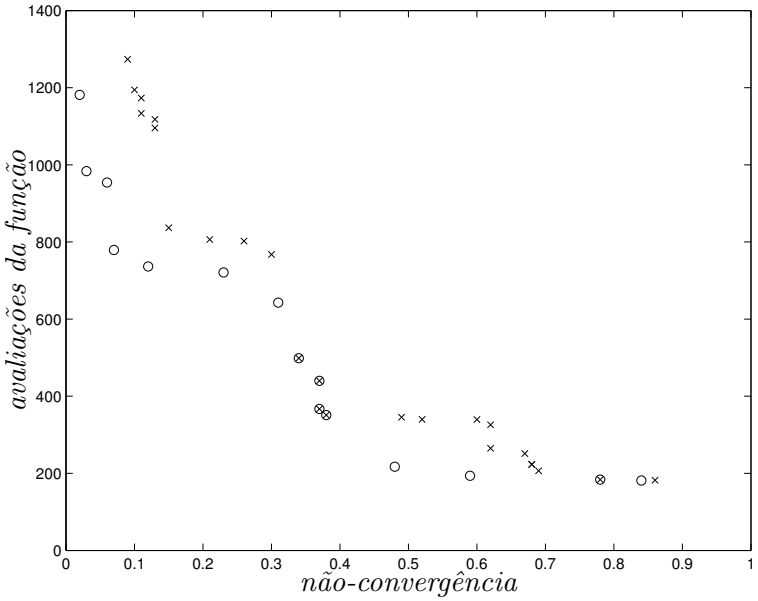


Figura 8.15: Dois conjuntos de Pareto, formados apenas por operadores "comuns" (×), e de sua combinação incluindo o novo operador de "cruzamento real polarizado" (○). Figuras de mérito: número de avaliações da função objetivo até a convergência (vertical) versus a fração de execuções que não convergem (horizontal). A função é a Rastrigin rotacionada.

Procedimentos de Avaliação: Novo Operador e Novo Algoritmo

Para o teste de novos *algoritmos* (isto é, uma combinação específica de operadores definidos), o procedimento a ser seguido é direto: avaliar o algoritmo sobre a mesma função de teste, computando-se as duas figuras de mérito. Feito isso, reavaliar o conjunto de Pareto, utilizando a equação (12.3) para incluir a informação a respeito do novo algoritmo.

Para o teste de novos *operadores*, a questão se torna mais sutil: um operador (de cruzamento, por exemplo) deve ser combinado com outros operadores (mutação e seleção, no mínimo) para dar origem a um AG executável.

Pergunta-se: será razoável a heurística de realizar o teste escolhendo "bons" algoritmos anteriormente testados (que pertençam, portanto, ao conjunto de Pareto anteriormente determinado), apenas substituindo o operador a ser testado nesses algoritmos? Isto significaria, de alguma forma, que os operadores seriam "independentes" uns dos outros. Ou, ao contrário, seria necessário testar o novo operador dentro de outras combinações de operadores que não pertencessem ao conjunto de Pareto anterior? Essa questão é respondida a seguir, no contexto da avaliação do novo operador de "cruzamento real-polarizado".

O operador de *cruzamento real-polarizado* foi testado em todas as combinações com os demais operadores da Tabela 8.4, o que significa 240 combinações. O conjunto de Pareto extraído desses dados é mostrado na Figura 8.15, superposto ao conjunto de Pareto anteriormente obtido. O conjunto de Pareto agora é constituído de apenas 15 algoritmos que são "não-dominados". Comparando agora os dois conjuntos de Pareto: o algoritmo mais confiável (que atinge mais vezes o ótimo global), dentre os algoritmos "comuns", falha em 9% das execuções, e necessita de cerca de 1270 avaliações de função para atingir o ótimo. No conjunto do operador de "cruzamento real polarizado", o melhor algoritmo falha em apenas 2% das execuções, e necessita de menos de 1200 avaliações da função objetivo. Existe um algoritmo no conjunto do operador "real polarizado" que necessita de menos de 800 avaliações de função, e falha em menos de 10% das execuções. O melhor algoritmo que necessita de menos de 800 avaliações de função, no outro conjunto, falha em cerca de 30% das execuções. O operador de *cruzamento real-polarizado*, para a função de teste em questão, mostrou representar uma melhoria bastante significativa de desempenho, conduzindo tanto ao algoritmo mais rápido quanto ao mais confiável, e gerando também a maioria das soluções intermediárias de Pareto (10 dos 15 algoritmos).

Infelizmente, a resposta à questão de serem os operadores independentes entre si é: *não*. Inspecionando os 10 novos algoritmos que utilizam o operador de cruzamento real-polarizado e compõem o conjunto de Pareto atualizado, constata-se que apenas dois deles puderam ser gerados pela substituição do operador de cruzamento em um dos 25 algoritmos que inicialmente constituíam o conjunto de Pareto. Quase todos os novo algoritmos que passaram a compor o conjunto de Pareto, neste caso, utilizam combinações de operadores que não haviam anteriormente conduzido a soluções de Pareto.

Isso não significa que um operador não possa ser avaliado: os dados acima claramente mostram que o operador de *cruzamento real polarizado* constitui

uma melhoria em relação às alternativas anteriormente conhecidas (pelo menos para a classe de problemas que compartilhar das características gerais da função Rastrigin rotacionada). Entretanto, a tarefa de "avaliação de operadores" deveria ser mais bem formulada que o procedimento padrão, relatado na maioria dos artigos sobre o assunto, que implicitamente depende da independência entre operadores.

8.6 Tratamento de Restrições

Métodos de busca por populações podem lidar de duas formas básicas com restrições: (i) através de "penalidades"; ou (ii) impedindo a incorporação à população de indivíduos que violam restrições. No primeiro caso, o problema de otimização pode envolver a busca de regiões factíveis em seu início (problema de factibilidade). No segundo caso, parte-se da premissa que a factibilidade pode ser resolvida *a priori*, seja analiticamente, seja por meio de uma busca aleatória que conduza com alta probabilidade a pontos factíveis.

8.7 Características de Comportamento

O comportamento dessa categoria de métodos frente a algumas dificuldades que irão surgir com freqüência em problemas de otimização de tipo geral é agora discutido.

8.7.1 Descontinuidades e Não-Diferenciabilidade

Esta família de métodos não faz uso de informação nem de gradiente nem de subgradiente. Não há, portanto, nenhuma dificuldade associada à possível não-diferenciabilidade dos funcionais que definem o problema.

8.7.2 Multimodalidade

A principal justificativa para o desenvolvimento dos métodos de "populações" é precisamente a possibilidade de tratar de problemas multimodais. De fato, esta família de métodos é a que mais se aproxima do conceito de "algoritmo de otimização para problemas genéricos". Há entretanto algumas limitações que são fundamentais:

- Os algoritmos tendem a convergir para ótimos que são "globais" na região em que foi distribuída a população inicial de pontos. Os mecanismos de otimização envolvidos, embora possam conduzir à convergência para pontos distantes da região em que se encontrava a população inicial, fazem isso à custa de grande esforço computacional e dependendo de alguma "sorte" (ou seja, com baixa probabilidade de acerto).

- Mesmo na própria região em que estava distribuída a população inicial, podem existir ótimos globais com baixa probabilidade de serem encontrados, ou seja, que estejam em uma "bacia de atração" muito pequena.

- Ainda que os ótimos globais sejam "fáceis de ser encontrados", ou seja, estejam localizados dentro da região em que se encontra a população inicial e estejam dentro de uma grande "bacia de atração", é possível (embora com baixa probabilidade) que os algoritmos de populações venham a não convergir para esses ótimos.

8.7.3 Velocidade de Convergência

Em problemas nos quais são aplicáveis todas as três famílias de métodos de otimização, os métodos de "populações" são os mais ineficientes, isto é, são aqueles que necessitam de maior esforço computacional para atingir os mesmos resultados. Este é o preço da maior generalidade desses métodos.

AG-B	AG-R	AG-RP
596	1001	707
109	25	875
30	76	32
70	48	38
125	1001	1001
1001	147	841
1001	1001	875
182	1001	1001
1001	29	42
1001	1001	763
1001	135	1001
1001	1001	426
1001	109	104
49	186	68
97	1001	45
1001	98	36
1001	122	40
97	115	236
1001	37	853
96	32	476
1001	1001	48
1001	1001	25
1001	68	1001
1001	107	1001
1001	143	405
31	68	49
1001	1001	81
81	1001	24
305	29	277
1001	810	595

Tabela 8.3: Número de iterações que foram necessárias para cada algoritmo atingir a convergência, para a função Rastrigin de duas dimensões e população de 40 indivíduos. As colunas correspondem aos dados de: AG-B (algoritmo genético binário), AG-R (algoritmo genético real) e AG-RP (algoritmo genético real polarizado). Cada linha corresponde à execução dos três algoritmos, para uma mesma população inicial gerada aleatoriamente.

Tabela 8.4: Alternativas de Operadores.

fator ssga	1	1
	2	0.8
	3	0.5
	4	0
mutação	1	cada bit
	2	bit por variável
	3	bit por indivíduo
cruzamento	1	um ponto
	2	dois pontos
	3	um ponto por variável
	4	uniforme
seleção	1	roleta
	2	amostragem determinística
	3	amostr. resto estocástico
elitismo	1	simples
	2	global
tamanho população	1	30
	2	100
	3	300

Capítulo 9

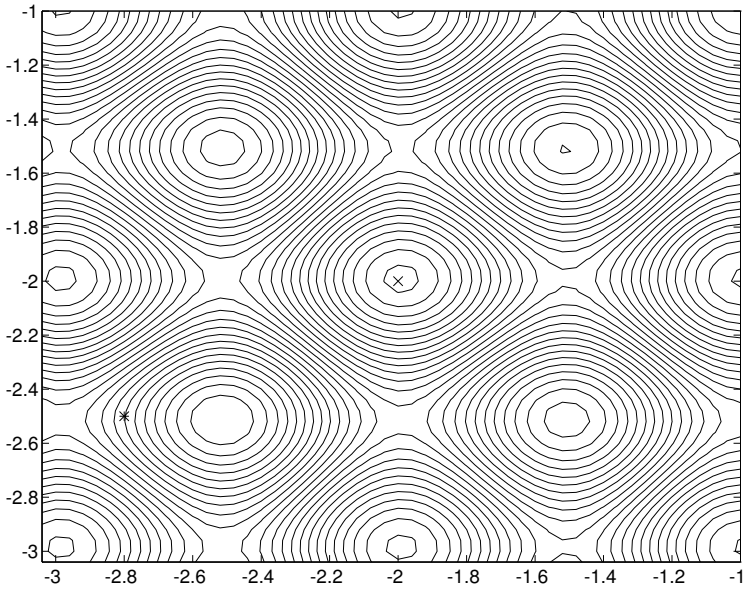
Exercícios - Otimização Escalar

1. Estabeleça algumas condições sob as quais o método de cálculo do gradiente por diferenças finitas irá fornecer direções (contrárias à do gradiente calculado) que se dirigirão estritamente para o interior da região de sub-nível corrente da função. (Ou seja, fixe premissas, tais como de diferenciabilidade, convexidade, quasi-convexidade, Lipschitz, etc, e relacione as mesmas com a propriedade desejada).

2. Mostre a execução de 1 passo do algoritmo de busca pelo gradiente na função:
$$f(x) = 2x_1^2 + x_2^2$$
a partir do ponto:
$$x[k] = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

3. Mostre a execução de 1 passo do algoritmo de busca pelo gradiente na função:
$$f(x) = 2x_1^2 + |x_2|$$
a partir do ponto:
$$x[k] = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

4. Dada uma função cujas curvas de nível são mostradas na figura a seguir, esboçar graficamente dois passos do algoritmo de busca pelo gradiente iniciando no ponto marcado com '*'.



5. Considere o problema (irrestrito) de otimização:

$$f(x) = x'Ax$$

sendo:

$$A = \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

(a) Calcule o gradiente $\nabla f(x)$ no ponto:

$$x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

(b) Calcule o mínimo obtido em uma busca unidimensional na direção $-\nabla f$ a partir do ponto x_0.

(c) Calcule o vetor:

$$d = -A^{-1}\nabla f(x_0)$$

(d) Determine o mínimo obtido em uma busca unidimensional na direção $-\nabla f$ a partir do ponto x_0.

6. Escreva o algoritmo (em pseudo-código) de busca unidimensional do mínimo de uma função de uma única variável, utilizando o método da bisseção ou o método da seção áurea.

7. Escreva o algoritmo (em pseudo-código) de determinação do gradiente de uma função através de diferenças finitas.

8. Escreva o algoritmo (em pseudo-código) de otimização pelo método do gradiente, supondo que estão disponíveis as rotinas:

   - gradiente - rotina de determinação do gradiente de uma função;
   - unidir - rotina de minimização unidimensional de uma função.

9. Escreva o algoritmo (em pseudo-código) de otimização pelo método de Newton, supondo que estão disponíveis as rotinas:

   - gradiente - rotina de determinação do gradiente de uma função;
   - hessiana - rotina de determinação da hessiana de uma função;
   - unidir - rotina de minimização unidimensional de uma função.

10. Discuta em que condições podem falhar:

    (a) O método da seção áurea de minimização unidimensional;

    (b) O método do gradiente de minimização de funções;

    (c) O método de diferenças finitas de cálculo de gradientes.

    Para cada situação acima mostre um exemplo de caso onde ocorre a falha.

11. Analise o algoritmo abaixo de minimização de uma função $f(x)$:

$$
\begin{array}{l}
\text{início} \\
k \leftarrow 0 \\
x_k \leftarrow x_0 \\
\text{enquanto não parada} \\
\qquad d \leftarrow \text{vetor aleatório} \\
\qquad \alpha \leftarrow \arg\min_{\alpha} f(x_k - \alpha d) \\
\qquad x_k \leftarrow x_k - \alpha d \\
\qquad k \leftarrow k + 1 \\
\text{fim-enquanto} \\
\text{fim}
\end{array}
\tag{9.1}
$$

    Discuta se esse algoritmo é capaz de determinar o mínimo da função.

12. Considere o problema de otimização:

$$f(x) = -x_1^2 - x_2^2$$

    com as restrições:

$$
\begin{cases}
x_1 + x_2 < 3 \\
x_1 < 2 \\
x_1 > 0 \\
x_2 > 0
\end{cases}
$$

(a) Esboce a região factível e algumas curvas de nível da função objetivo.

(b) Marque no esboço o ponto solução do problema.

(c) Mostre graficamente que as condições de Kuhn-Tucker são satisfeitas no ponto de ótimo.

13. Seja o problema definido por:

$$x^* = \arg \min_x 5x_1$$

$$\text{sujeito a: } \begin{cases} x_1 + x_2 \leq 0 \\ x_1^2 + x_2^2 - 4 \leq 0 \end{cases}$$

(a) Esboçar a região factível do problema.

(b) Determinar qual é o ponto de mínimo do problema.

(c) Escrever uma função de barreira que poderia ser utilizada para a determinação numérica do ótimo do problema.

(d) Escrever uma função de penalidade que poderia ser utilizada para a determinação numérica do ótimo do problema.

(e) Verificar graficamente a condição de Kuhn-Tucker no ponto de ótimo.

(f) Verificar, em outro ponto da fronteira da região factível, que a condição de Kuhn-Tucker não se verifica.

14. Seja o problema definido por:

$$x^* = \arg \min_x x_1 + x_2$$

$$\text{sujeito a: } \begin{cases} (x_1 - 3)^2 + (x_2 - 3)^2 - 9 \leq 0 \\ x_1^2 + (x_2 - 3)^2 - 9 \leq 0 \\ 3 - x_2 \leq 0 \end{cases}$$

(a) Esboçar a região factível do problema e algumas curvas de nível da função objetivo.

(b) Determinar qual é o ponto de mínimo do problema.

(c) Escrever uma função de penalidade que poderia ser utilizada para a determinação numérica do ótimo do problema.

(d) Verificar graficamente a condição de Kuhn-Tucker no ponto de ótimo.

(e) Verificar, em outro ponto da fronteira da região factível, que a condição de Kuhn-Tucker não se verifica.

Bibliografia

Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. (1985). A learning algorithm for Boltzman Machines, *Cognitive Science* **9**: 147–169.

Akgül, M. (1984). *Topics in Relaxation and Ellipsoidal Methods*, number 97 in *Research Notes in Mathematics*, Pitman Publishing Inc., London, UK.

Belmont-Moreno, E. (2001). The role of mutation and population size in genetic algorithms applied to physics problems, *International Journal of Modern Physics C* **12**(9): 1345–1355.

Bland, R. G., Goldfarb, D. & Todd, M. J. (1981). The ellipsoid method: a survey, *Operations Research* **29**(6): 1039–1091.

Cao, Y.-Y., Lam, J. & Sun, Y.-X. (1998). Static output feedback stabilization: an ILMI approach, *Automatica* **34**(12): 1641–1645.

Chankong, V. & Haimes, Y. Y. (1983). *Multiobjective Decision Making: Theory and Methodology*, North-Holland (Elsevier), New York.

Chen, B. S., Cheng, Y. M. & Lee, C. H. (1995). A genetic approach to mixed $\mathcal{H}_\infty/\mathcal{H}_2$ optimal PID control, *IEEE Control Systems Magazine* **15**(5): 51–60.

Chen, C. T. (1984). *Linear System Theory and Design*, Hartcourt Brace College Pub.

Choi, D. H. & Oh, S. Y. (2000). A new mutation rule for evolutionary programming motivated from backpropagation learning, *IEEE Transactions on Evolutionary Computation* **4**(2): 188–190.

Dias-Filho, W. (2003). *Algoritmos Cone-Elipsoidais para Geração de Soluções Eficientes: Construção e Aplicações*, PhD thesis, Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, Belo Horizonte, MG.

Dorigo, M., Maniezzo, V. & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents, *IEEE Trans. Sys. Man Cyb. - Part B* **26**(1): 29–41.

Dziuban, S. T., Ecker, J. G. & Kupferschmid, M. (1985). Using deep cuts in an ellipsoidal algorithm for nonlinear programming, *Math. Program. Study* **25**: 93–107.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.

J. C. Potts, T. D. G. & Yadav, S. B. (1994). The development and evaluation of an improved genetic algorithm based on migration and artificial selection, *IEEE Transactions on Systems, Man and Cybernetics* **24**(1): 73–86.

Jain, A. & Zongker, D. (1997). Feature selection: evaluation, application, and small sample performance, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(2): 153–158.

Johnson, J. M. & Ramat-Semii, Y. (1997). Genetic algorithms in engineering electromagnetics, *IEEE Antennas and Propagation Magazine* **39**(4): 7–25.

K. F. Man, K. S. T. & Kwong, S. (1996). Genetic algorithms: concepts and applications, *IEEE Transactions on Industrial Electronics* **43**(5): 519–534.

K. Rashid, J. A. R. & Freeman, E. M. (2000). A general approach for extracting sensitivity analysis from a neuro-fuzzy model, *IEEE Transactions on Magnetics* **36**(4): 1066–1070.

Khargonekar, P. P. & Rotea, M. A. (1991). Mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control: a convex optimization approach, *IEEE Transactions on Automatic Control* **36**(7): 824–837.

Luenberger, D. G. (1984). *Linear and Nonlinear Programming*, Addison-Wesley.

Meneguim, R. A. (1999). *Análise da sensibilidade de soluções em otimização através de elipsóides mínimos*, Master's thesis, Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, Belo Horizonte, MG.

Miranda, M. F. (2000). *Controle Multivariável na Presença de Incertezas*, PhD thesis, Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, Belo Horizonte, MG.

Naudts, B. & Kallel, L. (2000). A comparison of predictive measures of problem difficulty in evolutionary algorithms, *IEEE Transactions on Evolutionary Computation* **4**(1): 1–15.

Popper, K. R. (1974). *A Lógica da Pesquisa Científica (trad.)*, Ed. Cultrix. (Logic der Forschung, 5a. ed. 1973; 1a. ed. 1934).

Rudin, W. (1991). *Functional Analysis*, McGraw-Hill.

Saldanha, R. R., Takahashi, R. H. C., Vasconcelos, J. A. & Ramirez, J. A. (1999). Adaptive deep-cut method in ellipsoidal optimization for electromagnetic design, *IEEE Transactions on Magnetics, Part I* **35**(3): 1746–1749.

Sareni, B. & Krahenbuhl, L. (1998). Fitness sharing and niching methods revisited, *IEEE Transactions on Evolutionary Computation* **2**(3): 97–106.

Schell, T. & Wegenkittl, S. (2001). Looking beyond selection probabilities: adaptation of the χ^2 measure for the performance analysis of selection methods in GA's, *Evolutionary Computation* **9**(2): 243–256.

Scherer, C., Gahinet, P. & Chilali, M. (1997). Multiobjective output-feedback control via LMI optimization, *IEEE Transactions on Automatic Control* **42**(7): 896–911.

Shah, S., Mitchell, J. E. & Kupferschmid, M. (2001). An ellipsoid algorithm for equality-constrained nonlinear programs, *Computers and Operations Research* **28**: 85–92.

Shimomura, T. & Fujii, T. (2000). Multiobjective control design via sucessive over-bounding of quadratic terms, *Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, pp. 2763–2768.

Takahashi, R. H. C., Palhares, R. M., Dutra, D. A. & Gonçalves, L. P. S. (2001). Synthesis and characterization of Pareto-optimal solutions for the mixed $\mathcal{H}_2/\mathcal{H}_\infty$ control problem, *Proceedings of the 40th IEEE Conference on Decision and Control*, Orlando, FL, USA, pp. 3997–4002.

Takahashi, R. H. C., Peres, P. L. D. & Ferreira, P. A. V. (1997). Multiobjective $\mathcal{H}_2/\mathcal{H}_\infty$ guaranteed cost PID design, *IEEE Control Systems Magazine* **17**(5): 37–47.

Takahashi, R. H. C., Ramirez, J. A., Vasconcelos, J. A. & Saldanha, R. R. (2001). Sensitivity analysis for optimization problems solved by stochastic methods, *IEEE Transactions on Magnetics* **37**(5): 3414–3417.

Takahashi, R. H. C., Saldanha, R. R., Dias-Filho, W. & Ramirez, J. A. (2003). A new constrained ellipsoidal algorithm for nonlinear optimization with equality constraints, *IEEE Transactions on Magnetics* **39**(3): 1289–1292.

Takahashi, R. H. C., Vasconcelos, J. A., Ramirez, J. A. & Krahenbuhl, L. (2003). A multiobjective methodology for evaluating genetic operators, *IEEE Transactions on Magnetics* **39**(3): 1321–1324.

Tanomaru, J. (1995). Motivação, fundamentos e aplicações de algoritmos genéticos, *Anais do II Congr. Bras. de Redes Neurais*, Vol. 1, Curitiba, PR, Brasil.

Vasconcelos, J. A., Ramirez, J. A., Takahashi, R. H. C. & Saldanha, R. R. (2001). Improvements in genetic algorithms, *IEEE Transactions on Magnetics* **37**(5): 3414–3417.

Vidyasagar, M. (1993). *Nonlinear Systems Analysis*, Prentice-Hall.

Viennet, R., Fonteix, C. & Marc, I. (1996). Multicriteria optimization using a genetic algorithm for determining a Pareto set, *Int. J. Sys. Sci.* **27**(2): 255–260.

Wolpert, D. H. & Macready, W. G. (1997). No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation* **1**(1): 67–82.

Z. Michalewicz, K. Deb, M. S. & Stidsen, T. (2000). Test-case generator for nonlinear continuous parameter optimization techniques, *IEEE Transactions on Evolutionary Computation* **4**(3): 197–215.