Universidade Estadual de Campinas
Instituto de Matemática, Estatística e Computação Científica
Departamento de Matemática Aplicada

DIRECT, Análise Intervalar e Otimização Global Irrestrita

Douglas Soares Gonçalves *
Mestrado em Matemática Aplicada - Campinas - SP

Orientadora: Profa. Dra. Márcia Ap. Gomes Ruggiero

∗ Este trabalho teve apoio financeiro da CAPES.

DIRECT, Análise Intervalar e Otimização Global Irrestrita

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por **Douglas Soares Gonçalves** e aprovada pela comissão julgadora.

Campinas, 24 de abril de 2009.

Prof. Dra. **Márcia Ap. Gomes Ruggiero**
Orientadora

Banca examinadora:
Prof. Dra. Márcia Ap. Gomes Ruggiero (IMECC/UNICAMP)
Prof. Dr. Aurélio Ribeiro Leite de Oliveira (IMECC/UNICAMP)
Prof. Dr. Paulo Augusto Valente Ferreira (FEE/UNICAMP)

Dissertação apresentada ao Instituto de Matemática Estatística e Computação Científica, UNICAMP, como requisito parcial para a obtenção do título de **Mestre em Matemática Aplicada**.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Fabiana Bezerra Müller – CRB8 / 6162

Gonçalves, Douglas Soares
G586d DIRECT, análise intervalar e otimização global irrestrita/Douglas Soares Gonçalves -- Campinas, [S.P. : s.n.], 2009.

Orientador : Márcia Aparecida Gomes Ruggiero
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1.Otimização global. 2.Análise de intervalos (Matemática). 3.Programação não-linear. 4. Otimização Lipschitziana. . I. Ruggiero, Márcia Aparecida Gomes. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

(mfbm/imecc)

Título em inglês: DIRECT, interval analysis and unconstrained global optimization

Palavras-chave em inglês (Keywords): 1. Global optimization. 2. Interval analysis (Mathematics). 3. Nonlinear programming. 4. Lipschitzian optimization.

Área de concentração: Otimização

Titulação: Mestre em Matemática Aplicada

Banca examinadora: Profa. Dra. Márcia Aparecida Gomes Ruggiero (IMECC-UNICAMP)
Prof. Dr. Aurélio Ribeiro Leite de Oliveira (IMECC-UNICAMP)
Prof. Dr. Paulo Augusto Valente Ferreira (FEE-UNICAMP)

Data da defesa: 24/04/2009

Programa de Pós-Graduação: Mestrado em Matemática Aplicada

Dissertação de Mestrado defendida em 24 de abril de 2009 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.

Prof.(a). Dr(a). MÁRCIA AP. GOMES RUGGIERO

Prof. (a). Dr (a). AURÉLIO RIBEIRO LEITE DE OLIVEIRA

Prof. (a). Dr (a). PAULO AUGUSTO VALENTE FERREIRA

Agradecimentos

Agradeço:

a Deus por iluminar meu caminho,

à minha família o suporte e incentivo,

à CAPES o apoio financeiro,

a todos meus amigos tanto da pós-graduação quanto aos da república que me acolheram em Campinas,

à Ana Carolina o carinho e compreensão ao longo dessa caminhada,

à profa. Márcia a orientação e amizade desde a graduação,

à UNICAMP e ao IMECC a qualidade de ensino e o suporte à pesquisa,

a todos os professores do Departamento de Matemática Aplicada que contribuíram na minha formação e aprimoramento.

Resumo

Neste trabalho analisamos dois métodos para otimização global irrestrita: `DIRECT`, um método tipo branch-and-select, baseado em otimização Lipschitziana, com um critério especial de seleção que balanceia a ênfase entre busca local e global; e um método tipo branch-and-bound empregando as mais recentes técnicas em análise intervalar, junto com *back-boxing* e busca local, para acelerar o processo de convergência. Variações do método branch-and-bound intervalar, e combinações deste com as idéias do `DIRECT` foram formuladas e implementadas. A aplicação a problemas clássicos encontrados na literatura mostrou que as estratégias adotadas contribuíram para melhorar o desempenho dos algoritmos.

Palavras-chave: `DIRECT`, otimização Lipschitziana, branch-and-bound, análise intervalar, otimização global irrestrita, busca local, *back-boxing*.

Abstract

In this work we analyze two unconstrained global optimization methods: `DIRECT`, a branch-and-select method, based on Lipschitzian optimization, with a special selection criterion that balances the emphasis between local and global search; and a branch-and-bound method incorporating the state of art interval analysis techniques, with back-boxing and local search, to speed up the convergence process. Interval branch-and-bound method variations, and combinations of them with the ideas of `DIRECT` were proposed and implemented. Application to classical problems found in literature, shows that the adopted strategies contribute to improve the performance of the algorithms.

Keywords: `DIRECT`, Lipschitzian optimization, branch-and-bound, interval analysis, unconstrained global optimization, local search, back-boxing.

Sumário

Capítulo 1

Introdução

Vários problemas reais podem ser formulados como *problemas de otimização*, isto é, problemas com uma *função objetivo* que depende de *variáveis de decisão*, que deve ser *otimizada* sujeita a um conjunto de *restrições.*A resolução de problemas desse tipo é estímulo de pesquisas em diversas áreas.

De um modo geral, o problema de otimização pode ser definido como encontrar $\bar{x} \in X^0 \subset \mathbb{R}^n$ tal que $f(\bar{x}) \leq f(x)$ para todo $x \in X^0$:

$$\min_{x \in X^0} f(x). \tag{1.1}$$

O ponto $\bar{x}$ é chamado *minimizador global* de f em X^0.

Se f é contínua e X^0 compacto, a existência do minimizador global é assegurada pelo Teorema de Bolzano-Weierstrass, [40].

Em muitos problemas de otimização encontrar uma solução local é suficiente. Nestes casos, estamos interessados em encontrar um *minimizador local* de f em X^0, isto é, $\bar{x} \in X^0$ tal que $f(\bar{x}) \leq f(x)$ para todo $x \in X^0 \cap V(\bar{x})$, onde $V(\bar{x})$ denota uma vizinhança de $\bar{x}$. Para tais problemas existem métodos eficientes e robustos mesmo para o caso de grande porte, isto é, com um grande número de variáveis e/ou restrições (igualdades e desigualdades) que definem o conjunto viável X^0, ver [44], [46], [56].

Porém em alguns problemas, que aparecem por exemplo na Economia, na Engenharia Química [15], etc, há a necessidade de se encontrar o mínimo global.

Neste trabalho, consideramos o problema de encontrar o *mínimo global* de uma função $f : X^0 \to \mathbb{R}$, onde $f \in \mathbb{C}^2$ e $X^0 \subset \mathbb{R}^n$ é um conjunto definido por restrições de canalização:

$$X^0 = \{x \in \mathbb{R}^n \mid l_i \leq x_i \leq u_i, \text{ para } i = 1, 2, \ldots, n\},$$

e chamaremos X^0 de *caixa* ou *retângulo* em $\mathbb{R}^n$.

Seja $\bar{x} \in X^0$, minimizador global de (1.1). Dizemos que o problema de otimização global pode ser considerado resolvido, para uma dada precisão $\varepsilon > 0$, se um elemento dos seguintes conjuntos for identificado [9]:

$$A_x(\varepsilon) = \left\{x \in X^0 \mid \|x - \bar{x}\| \leq \varepsilon\right\}, \tag{1.2}$$

$$A_f(\varepsilon) = \left\{x \in X^0 \mid |f(x) - f(\bar{x})| \leq \varepsilon\right\} \tag{1.3}$$

Uma desvantagem de (1.2) é que pequenas perturbações nos dados do problema podem afetar a localização de $\bar{x}$. Uma terceira possibilidade é definir:

$$\phi(y) = \frac{m(\{z \in X^0 \mid f(z) \leq y\})}{m(X^0)},$$

onde m(.) é a *medida de Lebesgue* e tomar:

$$A_\phi(\varepsilon) = \left\{x \in X^0 \mid \phi(f(x)) \leq \varepsilon\right\}.$$

Podemos notar, entretanto, que esse conjunto pode conter elementos cujos valores de função objetivo diferem consideravelmente de $f(\bar{x})$.

Poucos métodos para otimização global foram desenvolvidos, em comparação com a variedade de métodos para otimização local. A dificuldade da otimização global em relação a otimização local é fácil de ser compreendida. Se assumirmos que f é duas vezes continuamente diferenciável, então tudo que é necessário para testar se um ponto é minimizador local é o conhecimento das derivadas de primeira e segunda ordem em tal ponto. Se o teste falha, a diferenciabilidade contínua da função assegura que em uma vizinhança do ponto podemos encontrar outro ponto com valor de função objetivo menor. Dessa forma, uma sequência de pontos convergindo a um minimizador local pode ser construída, pois as condições de otimalidade estão bem estabelecidas, [44], [46], [56].

Infelizmente testes locais não são suficientes para verificar a otimalidade global, salvo casos em que o problema (1.1) possui alguma estrutura especial (por exemplo f *convexa* em X^0 convexo).

Os métodos desenvolvidos para otimização global podem ser divididos em duas classes, dependendo se incorporam ou não elementos estocásticos [8].

A maioria dos *métodos estocásticos* envolve a avaliação de f em uma amostra aleatória de pontos de X^0 e subsequentes manipulações dessa amostra. Como resultado a garantia absoluta de sucesso é sacrificada. Contudo, sob modestas condições sobre a distribuição da amostra e f, a probabilidade de um elemento de $A_x(\varepsilon)$, $A_f(\varepsilon)$ ou $A_\phi(\varepsilon)$ ser escolhido se aproxima de 1 à medida que o tamanho da amostra aumenta, e o número de iterações tende a infinito.

Métodos determinísticos não envolvem nenhum conceito estocástico. Para garantir o sucesso, tais métodos inevitavelmente envolvem hipóteses adicionais sobre f, porém são capazes de explorar por completo o espaço de busca, verificando assim a otimalidade global.

Outra classificação para os métodos de otimização global é baseada na *filosofia* do método, como em [31]:

1. *Partição e busca*. Nestes métodos X^0 é particionado sucessivamente em subregiões menores nas quais o ótimo global é procurado, no espírito dos métodos branch-and-bound para otimização combinatória [53], [71]. Exemplos: αBB [1], BARON [68], [69], métodos baseados em análise intervalar [25], [35], [61].

2. *Aproximação e busca*. Nessa abordagem, f é substituída por aproximações cada vez mais fáceis de lidar do ponto de vista computacional. Exemplos: Otimização Lipschitziana, DIRECT, [33], [66].

3. *Descenso global*. Esses métodos buscam uma redução permanente nos valores de f, através de heurísticas de diversificação que evitam a estagnação a um mínimo local, culminando na chegada ao ótimo global. Exemplos: *simulated annealing* [36], algoritmos genéticos [20].

4. *Melhoramento do mínimo local*. Explora a disponibilidade de rotinas eficientes para otimização local. Tais métodos buscam gerar uma sequência

de minimizadores locais com valores decrescentes. Claramente, o mínimo global é o último encontrado na sequência. Exemplos: *tunneling method* [42], *filled functions* [19].

5. *Enumeração dos mínimos locais.* A enumeração completa dos mínimos locais é também um caminho para resolver o problema de otimização global. Exemplos: *trajectory methods* [5], *homotopy methods* [70], *multistart* [65].

Neste trabalho consideramos o estudo de dois métodos para otimização global.

Inicialmente analisamos um método do tipo aproximação e busca para otimização global: o método `DIRECT`, [14], [16], [33]. Este método apresenta bons resultados para problemas de otimização global com restrições de caixa para um número pequeno de variáveis, chegando suficientemente próximo dos minimizadores globais com um número modesto de avaliações de função, se comparado a outros métodos [20], [65].

Em seguida analisamos a aplicação da teoria de análise intervalar [25], [49], [50], em um método do tipo branch-and-bound, resultando na elaboração de um algoritmo para otimização global irrestrita. Neste desenvolvimento consideramos técnicas e heurísticas de trabalhos anteriores, [25], [26], [35], [52].

Propusemos também uma *variação* do algoritmo branch-and-bound com análise intervalar, que emprega os critérios de *seleção* e *partição* do algoritmo `DIRECT`.

Em todos os casos, algoritmos foram implementados e aplicados a um conjunto de problemas teste extraídos da literatura [8], [9], [33], [41]. Comparamos os resultados obtidos por cada algoritmo e ressaltamos os pontos fortes, bem como as dificuldades encontradas por cada um.

Esta dissertação encontra-se organizada da seguinte forma.

No Capítulo 2, apresentamos as idéias de otimização global para funções Lipschitz e descrevemos o algoritmo `DIRECT`, bem como um estudo detalhado de suas propriedades. No Capítulo 3, apresentamos os elementos de análise intervalar e ao final, uma breve descrição sobre envelopes convexos e subestimativas convexas. A aplicação da teoria de análise intervalar em um método tipo branch-and-bound é apresentada no Capítulo 4. Neste capítulo também apresentamos algoritmos

para a resolução do problema de otimização global. No Capítulo 5, apresentamos os detalhes da implementação e os resultados computacionais obtidos para um conjunto de problemas teste extraídos da literatura. Por fim as conclusões e trabalhos futuros são expostos no Capítulo 6.

Capítulo 2

Otimização Lipschitz e o algoritmo DIRECT

Neste capítulo apresentamos algoritmos para otimização global irrestrita (ou com restrições de caixa) de funções que atendem a condição Lipschitz. Para tais funções é possível estimar limitantes para o mínimo global em cada subintervalo, o que nos permite definir algoritmos do tipo aproximação e busca (*aproximating and search*). Algoritmos neste sentido foram propostos por Shubert [66] e Jones et al [33], entre outros.

2.1 Funções Lipschitz

Definição 2.1.1. *Seja $f : X \to \mathbb{R}$, $X \subset \mathbb{R}^n$. A função f é dita Lipschitz, se existe uma constante $L > 0$ tal que*

$$|f(x) - f(y)| \leq L \|x - y\| \tag{2.1}$$

para todo $x, y \in X$.

A menor constante L para qual a desigualdade acima é válida é denominada *constante de Lipschitz*. Em (2.1), $\|.\|$ denota uma norma em $\mathbb{R}^n$.

Note que toda função Lipschitz é *contínua*. Seja $f : \mathbb{R}^n \to \mathbb{R}$ Lipschitz, com constante L. Dado $\varepsilon > 0$, existe um δ_ε tal que, se $\|x - y\| < \delta_\varepsilon$ então $|f(x) - f(y)| \leq \varepsilon$. De fato

$$|f(x) - f(y)| \leq L \|x - y\| \leq L\delta_\varepsilon,$$

e tomando $\delta_\varepsilon < \frac{\varepsilon}{L}$ verificamos a afirmação.

Por outro lado nem toda função contínua é Lipschitz. Por exemplo, $f(x) = x^2$ definida em $\mathbb{R}$, não é Lipschitz pois quando $x \to \infty$ temos que $\frac{f(x)-f(y)}{x-y} \to \infty$. A função $f(x) = \sqrt{x}$ definida em $[0, 1]$, também *não* é Lipschitz, pois quando $x \to 0$ não há constante L que limite sua inclinação.

Uma função f *diferenciável* é Lipschitz, se possui derivadas de primeira ordem limitadas

$$\max_i \left| \frac{\partial f}{\partial x_i}(x) \right| = \max_i \left| \lim_{t \to 0} \frac{f(x + te_i) - f(x)}{|t|} \right| \leq L.$$

Porém uma função *não* diferenciável pode ser Lipschitz. É o caso de $f(x) = |x|$, cuja constante de Lipschitz é $L = 1$.

2.2 O Algoritmo de Shubert

Em 1972, Shubert [66] propôs um algoritmo para minimização de funções Lipschitz em uma variável.

Considere $f : \mathbb{R} \to \mathbb{R}$ uma função Lipschitz em um intervalo $[a, b]$. Sendo f Lipschitz em $[a, b]$ temos que:

$$|f(x) - f(y)| \leq L\,|x - y|\,, \tag{2.2}$$

para todo $x, y \in [a, b]$.

Tomando x em (a, b) e substituindo y pelos extremos do intervalo temos

$$f(x) \geq f(a) - L(x - a) \tag{2.3}$$
$$f(x) \geq f(b) + L(x - b). \tag{2.4}$$

A essas duas inequações correspondem as retas com inclinações $-L$ e L conforme mostra a Figura 2.1.

Das desigualdades (2.3) e (2.4) temos que

$$f(x) \geq \frac{f(a) + f(b)}{2} - \frac{L}{2}(b - a) = f_{lb} \tag{2.5}$$

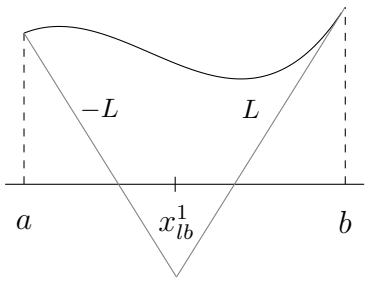

Figura 2.1: Limitante inferior dado pela constante de Lipschitz

que é satisfeita como igualdade no ponto

$$x_{lb} = \frac{a+b}{2} + \frac{f(a) - f(b)}{2L}.$$

Dessa forma, conhecida a constante de Lipschitz L, a relação (2.5) fornece um limitante inferior para a função f no intervalo $[a, b]$.

O algoritmo de Shubert, baseia-se nesse resultado, e procura gerar aproximações cada vez melhores para f em subintervalos cada vez menores.

Tomando como exemplo a Figura 2.1, após obter um limitante inferior para o valor da função no intervalo $[a, b]$, a função é avaliada no ponto x_{lb}^1, e em seguida o intervalo $[a, b]$ é particionado em dois subintervalos $[a, x_{lb}^1]$ e $[x_{lb}^1, b]$, e para cada subintervalo, novos limitantes inferiores são obtidos. A seguir é escolhido o subintervalo com menor limitante inferior para ser particionado e assim sucessivamente, como mostra a Figura 2.2. Este processo continua até que em algum momento $|f(x_{lb}) - f_{lb}| < \xi$, para alguma tolerância $\xi > 0$.

À medida que algoritmo avança, a subestimativa linear por partes aproxima cada vez melhor a função objetivo. É claro que trata-se de uma subestimativa válida pois consideramos o conhecimento da constante de Lipschitz L. Todavia um valor $K > L$ poderia ser usado, e tudo continuaria válido, porém quanto mais distante K estiver da constante de Lipschitz, menos precisos serão os limitantes inferiores, e mais lenta será a convergência do método [27].

As dificuldades intrínsecas ao método de Shubert são a necessidade da constante de Lipschitz L (ou uma estimativa superior para ela), e sua generalização para

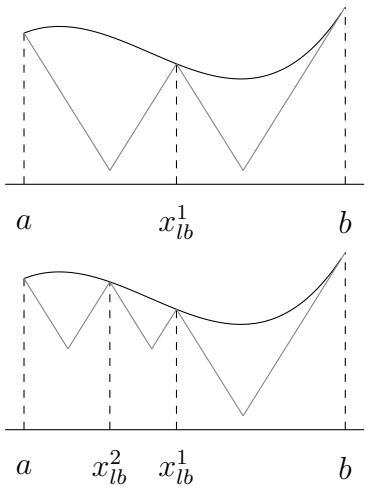

Figura 2.2: Interpretação geométrica do algoritmo de Shubert

várias variáveis pois, como são necessários os extremos do intervalo (ou no caso de $\mathbb{R}^n$, os vértices da caixa), em n variáveis seriam necessárias 2^n avaliações de função apenas para iniciar o método.

Apesar desses inconvenientes alguns autores estudaram a extensão do algoritmo de Shubert para várias variáveis, dentre os quais podemos citar Galperin [18], Mladineo [48] e Pinter [58].

2.3 O Algoritmo DIRECT

Buscando transpor as principais dificuldades encontradas no algoritmo de Shubert, em 1993, Jones, Perttunen e Stuckman [33] propuseram o algoritmo DIRECT (DIviding RECTangles). Para obter um algoritmo viável para dimensões maiores, a forma como o espaço de busca é particionado foi modificada. Ao invés de avaliar a função nos extremos do intervalo, a mesma é avaliada apenas no *centro* do intervalo. Assim com n variáveis não serão necessários 2^n pontos para iniciar o algoritmo, mas apenas um, o centro da caixa em $\mathbb{R}^n$. É claro que isso torna o algoritmo aplicável a dimensões maiores, mas não assegura um bom desempenho em tais espaços.

Estudaremos primeiro o algoritmo DIRECT para uma variável. Em decorrência da mudança para a avaliação da função em pontos centrais dos intervalos, é necessária também uma modificação nos limitantes inferiores.

Seja f uma função Lipschitz em um intervalo $[a, b]$, com $c = (a + b)/2$ o ponto central do intervalo, e constante de Lipschitz L. Então temos que:

$$\begin{aligned} f(x) &\geq f(c) + L(x - c), \qquad \text{para } x \leq c \\ f(x) &\geq f(c) - L(x - c), \qquad \text{para } x \geq c, \end{aligned}$$

como podemos ver na Figura 2.3.

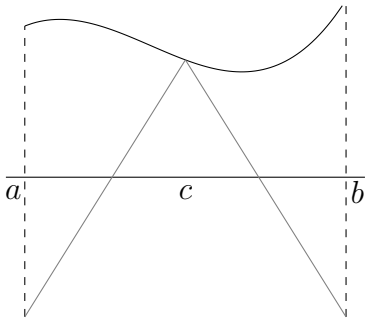

Figura 2.3: Limitante inferior com avaliação no ponto médio

Em ambos os casos temos funções lineares no lado direito, e é fácil ver que o menor valor para elas se dará nos extremos do intervalo, valor este que nos fornece um limitante inferior para f em $[a, b]$

$$f_{lb} = f(c) - L\frac{(b - a)}{2}. \tag{2.6}$$

Como estamos interessados em avaliar a função no centro do intervalo, o processo de subdivisão também é alterado. Considere o intervalo $[a, b]$ com ponto médio c. Ao invés de subdividir $[a, b]$ em dois subintervalos a partir de c, o intervalo é subdividido em três subintervalos de mesmo tamanho de modo que c seja o ponto médio do novo subintervalo central, como mostra a Figura 2.4.

Com esse esquema aproveitamos o ponto médio anterior, e evitamos uma avaliação de função. Além disso, para saber quais serão os próximos pontos a serem

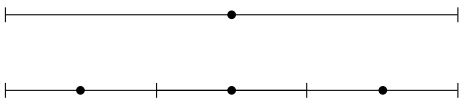

Figura 2.4: Esquema de subdivisão dos intervalos

avaliados, basta somar ou subtrair $\frac{d}{3}$ do ponto médio, onde d é o tamanho do intervalo que foi dividido.

Outro problema a ser resolvido é a necessidade da constante de Lipschitz L. Na prática é difícil obter tal constante, e uma estimativa muito elevada para a mesma pode fornecer limitantes pouco precisos, prejudicando a eficiência do algoritmo.

Para vencer essa dificuldade devemos lembrar do algoritmo de Shubert e entender qual a influência da constante de Lipschitz L.

Como vimos anteriormente o algoritmo de Shubert seleciona os intervalos baseado na comparação de seus limitantes inferiores. De acordo com a desigualdade (2.5), cada limitante inferior é formado pela soma de dois termos

$$\frac{f(a) + f(b)}{2} \tag{2.7}$$

e

$$-\frac{L(b-a)}{2}. \tag{2.8}$$

O primeiro termo é pequeno quando o valor da função nos extremos do intervalo é pequeno, o que motiva a escolha de intervalos com baixos valores na função objetivo, dando uma ênfase maior para a *busca local*.

Por outro lado o segundo termo se torna menor (mais negativo) à medida em que o tamanho do intervalo em questão se torna maior. Este termo motiva a escolha de intervalos de maior amplitude, dando uma ênfase maior para a *busca global*.

Dessa forma a constante de Lipschitz L pode ser interpretada como um *peso relativo* entre busca global e busca local.

Como a constante de Lipschitz é um limitante superior para a taxa de variação da função, normalmente seu valor é alto. Dessa forma, uma ênfase muito grande é dada à busca global, o que resulta em uma convergência lenta do algoritmo de Shubert.

Voltando ao DIRECT, suponha que o espaço de busca já está particionado em intervalos $[a_i, b_i]$, para $i = 1, 2, ..., m$, e que queremos selecionar um desses intervalos para processamento posterior. Na Figura 2.5 representamos os intervalos em questão por pontos, com coordenada horizontal $(b_i - a_i)/2$ e vertical $f(c_i)$.

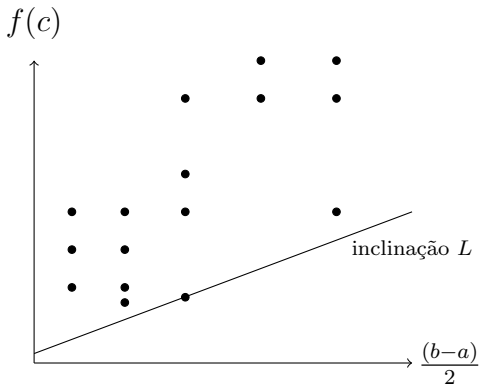

Figura 2.5: Interpretação geométrica da seleção de intervalos

Ao traçar uma reta com inclinação L passando por qualquer ponto na Figura 2.5, a interseção com o eixo vertical será o limitante inferior para o valor da função no intervalo correspondente, f_{lb}, conforme (2.6). Logo, se o critério de seleção de intervalos é baseado no menor limitante inferior, para obtê-lo basta posicionar uma reta de inclinação L abaixo da nuvem de pontos e deslocá-la para cima até tocar um ponto. Este ponto será o escolhido pois terá o menor limitante inferior.

A constante de Lipschitz refletida pela inclinação da reta na Figura 2.5, determina o *peso relativo* entre busca local e busca global. Como já dito, em muitos casos a constante de Lipschitz L apresenta um valor elevado e consequentemente uma ênfase demasiada é dada para a busca global. Na Figura 2.6 vemos que para uma maior constante L o intervalo selecionado estará dentre aqueles de maior amplitude.

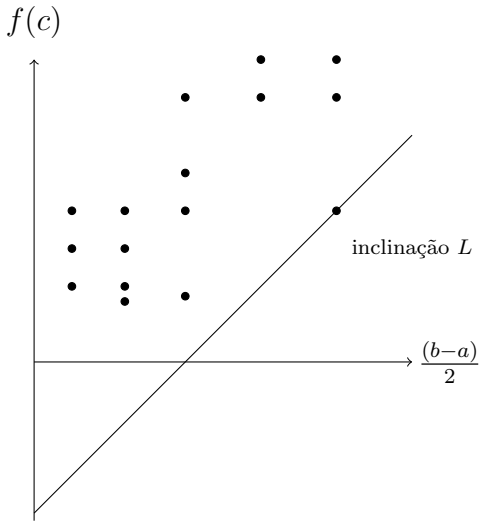

Figura 2.6: Intervalo selecionado para constante L maior

Mas o que aconteceria se ao invés da constante de Lipschitz L, usássemos um outro peso relativo (inclinação) $K > 0$ na escolha dos intervalos? E se ao invés de um único valor para K, atribuíssemos vários valores positivos? Quais intervalos seriam escolhidos?

Analisando, vemos que os intervalos selecionados seriam aqueles representados por pontos na parte inferior direita do envoltório convexo da nuvem de pontos, como podemos ver na Figura 2.7.

Note que esse tipo de seleção contempla desde intervalos pequenos com valor baixo de função objetivo até intervalos maiores, com menor valor de função dentre eles. Assim deixamos de usar a constante de Lipschitz L e passamos a considerar todas as inclinações K positivas possíveis, o que tende a balancear a ênfase entre busca local e global.

A definição a seguir estabelece as condições que devem ser satisfeitas para que um intervalo seja considerado potencialmente ótimo, [33], para uma constante $K > 0$.

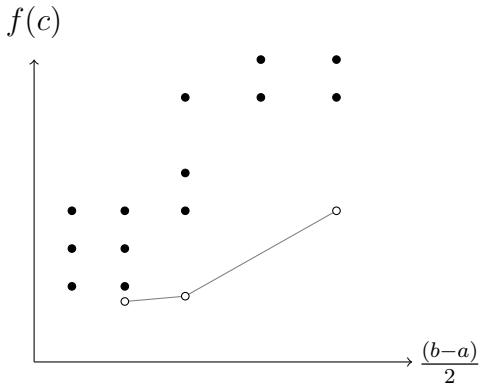

Figura 2.7: Conjunto de intervalos selecionados

Definição 2.3.1. *Suponha que temos o intervalo* $[a, b]$ *particionado em intervalos* $[a_i, b_i]$ *com pontos médios* c_i*, para* $i = 1, 2, ..., m$*. Seja* $\varepsilon > 0$ *uma constante positiva, e* f_{min} *o menor valor de função obtido até o momento. O intervalo* j *é dito potencialmente ótimo se existe uma constante positiva* $K > 0$ *tal que*

$$f(c_j) - K\frac{(b_j - a_j)}{2} \leq f(c_i) - K\frac{(b_i - a_i)}{2}, \quad \text{para todo } i \neq j \tag{2.9}$$

$$f(c_j) - K\frac{(b_j - a_j)}{2} \leq f_{min} - \varepsilon\,|f_{min}|\,. \tag{2.10}$$

A primeira condição seleciona os intervalos que se encontram na parte inferior direita do envoltório convexo da nuvem de pontos, e a segunda garante que o limitante inferior de tais intervalos será menor que a solução atual por uma quantidade não trivial. O parâmetro ε se faz necessário para evitar que o algoritmo se torne muito local, desperdiçando preciosas avaliações de função por melhorias extremamente pequenas.

Resumidamente o algoritmo DIRECT nada mais é que o algoritmo de Shubert modificado para usar o ponto médio como ponto de avaliação e explorar todos os intervalos potencialmente ótimos a cada iteração.

Como em problemas práticos é difícil estimar a constante de Lipschitz L, não temos um limitante inferior para a função e portanto não cabe o critério de parada $|f(x_{lb}) - f_{lb}| < \xi$. Na prática o algoritmo DIRECT é interrompido ao atingir

um certo número pré-estabelecido de iterações e/ou avaliações de função.

2.4 DIRECT em Várias Variáveis

No mesmo trabalho [33], Jones et al., descrevem o algoritmo DIRECT para o caso multidimensional. Inicialmente, assume-se que toda variável está limitada ao intervalo $[0, 1]$ (podemos sempre normalizar nosso problema para que isso seja verdade, mediante a transformação linear $T(x) = M(x - l)$, onde $M = diag(1/(u_i - l_i))$). Assim nosso espaço de busca será o hipercubo unitário em $\mathbb{R}^n$.

À medida que o algoritmo realiza as iterações, o hipercubo unitário será particionado em hiper-retângulos, nos quais a função será avaliada nos pontos centrais.

A principal questão na generalização para o caso multidimensional está em como dividir os hiper-retângulos. Vamos primeiro discutir essa questão para os hipercubos e em seguida para os hiper-retângulos.

Poderíamos optar por dividir um hipercubo selecionando arbitrariamente uma coordenada, e dividindo-o em três ao longo dessa coordenada. Como a seleção arbitrária não é conceitualmente atrativa, usaremos a seguinte abordagem. Primeiro avaliamos a função nos pontos $c \pm \delta e_i$, para $i = 1, 2, ..., n$, onde c é ponto central do hipercubo, δ é um terço do lado do hipercubo e e_i é o i-ésimo vetor canônico. A seguir dividimos o hipercubo em três hiper-retângulos ao longo da coordenada com melhor valor de função objetivo, de modo que os pontos avaliados nessa coordenada sejam pontos centrais dos novos hiper-retângulos.
Depois, o hiper-retângulo central, aquele que contém c, deve ser particionado em três ao longo da coordenada com segundo melhor valor de função objetivo e assim por diante.

No caso de hiper-retângulos, o mesmo processo de divisão é usado, porém é aplicado apenas em relação às coordenadas associadas aos lados mais longos do hiper-retângulo.

A Figura 2.8 ilustra o esquema para duas dimensões. Nesta figura os valores de função em cada ponto são apresentados. Note que a divisão ocorre primeiro ao longo da coordenada com menor valor de função associado.

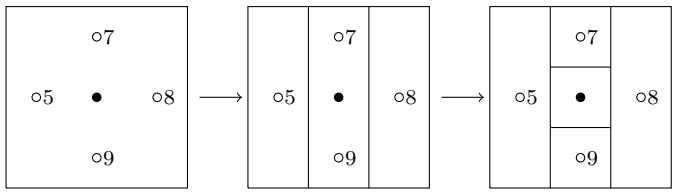

Figura 2.8: Esquema de divisão de hiper-retângulos

O processo de identificação dos hiper-retângulos potencialmente ótimos é o mesmo que no caso unidimensional. Para cada hiper-retângulo, conhecemos o valor da função objetivo em seu ponto médio e a distância d de tal ponto aos vértices.

Definição 2.4.1. *Suponha que temos o hipercubo unitário particionado em m hiper-retângulos. Seja c_i o centro de cada hiper-retângulo, e d_i, a distância desse centro aos vértices. Seja $\varepsilon > 0$ uma constante positiva. Um hiper-retângulo j é dito potencialmente ótimo se existe $K > 0$ tal que*

$$f(c_j) - Kd_j \leq f(c_i) - Kd_i, \quad \text{para todo } i \neq j \tag{2.11}$$

$$f(c_j) - Kd_j \leq f_{min} - \varepsilon |f_{min}| . \tag{2.12}$$

Alguns autores [16], [17], usam outras medidas para os retângulos, por exemplo tomando d_j como o tamanho do maior lado do hiper-retângulo j. No entanto este tipo de modificação pode dar uma ênfase demasiada à busca local.

Decorrem da Definição 2.4.1 algumas observações importantes:

- se um hiper-retângulo j é potencialmente ótimo então $f(c_j) \leq f(c_i)$ para todo hiper-retângulo i de tamanho $d_i = d_j$;
- se $d_j \geq d_i$ para todo i, e além disso $f(c_j) \leq f(c_k)$ para todo k tal que $d_k = d_j$, então o hiper-retângulo j é potencialmente ótimo, isto é, dentre os retângulos de maior tamanho, aquele que tiver o menor valor de função *sempre* será potencialmente ótimo;
- se $d_j \leq d_i$ para todo i, e o hiper-retângulo j for potencialmente ótimo, então $f(c_j) = f_{min}$, isto é, se um dos retângulos de menor tamanho foi selecionado é porque tal retângulo possui o menor valor de função até o momento.

As observações acima são formalizadas no Lema 2.4.2 atribuído a Gablonsky [16]. Este lema fornece uma forma eficaz de identificar os hiper-retângulos potencialmente ótimos.

Lema 2.4.2. *Seja $\varepsilon > 0$ uma constante, e f_{min} o menor valor de função objetivo obtido até o momento. Seja I o conjunto de índices dos hiper-retângulos existentes, $j \in I$, e*

$$\begin{aligned} I_1 &= \{i \in I \mid d_i < d_j\} \\ I_2 &= \{i \in I \mid d_i > d_j\} \\ I_3 &= \{i \in I \mid d_i = d_j\}. \end{aligned}$$

Se j é tal que

i) $f(c_j) \leq f(c_i)$, *para todo* $i \in I_3$

ii) existe uma constante $K > 0$ *tal que*

$$\max_{i \in I_1} \frac{f(c_j) - f(c_i)}{d_j - d_i} \leq K \leq \min_{i \in I_2} \frac{f(c_i) - f(c_j)}{d_i - d_j}$$

iii)

$$\frac{f_{min} - f(c_j)}{|f_{min}|} + \frac{d_j}{|f_{min}|} \min_{i \in I_2} \frac{f(c_i) - f(c_j)}{d_i - d_j} \geq \varepsilon, \quad \text{se } f_{min} \neq 0,$$

ou

$$f(c_j) \leq d_j \min_{i \in I_2} \frac{f(c_i) - f(c_j)}{d_i - d_j}, \qquad \text{se } f_{min} = 0,$$

então o hiper-retângulo j é potencialmente ótimo.

Note que as condições (i) e (ii) do lema equivalem à primeira condição da Definição 2.4.1, enquanto a condição (iii) equivale à segunda condição da Definição 2.4.1.

Na Figura 2.9 temos uma representação gráfica do Lema 2.4.2. Os retângulos 1, 2, 3, 4 e 5 satisfazem a primeira condição do lema, pois são os que possuem menor valor de função para cada amplitude. Porém os retângulos 1 e 4 não satisfazem a segunda condição.

Somente os retângulos 3 e 5 satisfazem a terceira condição, pois existe $K > 0$

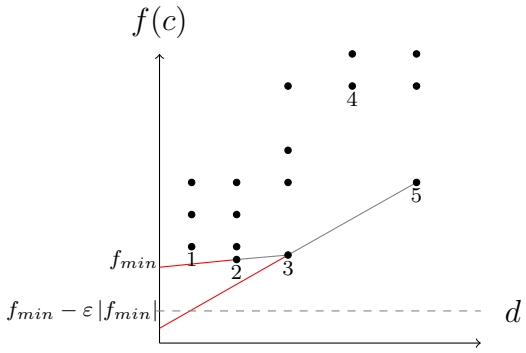

Figura 2.9: Interpretação geométrica do lema

tal que seus limitantes inferiores são suficientemente menores que f_{min}. Note que o limitante para 3 e 5 está abaixo da reta pontilhada, proveniente da terceira condição. Assim 3 e 5 seriam os retângulos potencialmente ótimos.

Com isso temos todos os ingredientes necessários para o algoritmo DIRECT em várias variáveis. Inicializamos a busca pelo centro do hipercubo unitário. A cada iteração identificamos os hiper-retângulos potencialmente ótimos e os subdividimos conforme descrito anteriormente. O processo continua até atingir um certo limite no número de iterações e/ou avaliações de função.

Algoritmo 1: DIRECT

1. Normalize o domínio original X^0 para o hipercubo unitário em $\mathbb{R}^n$.
   Avalie a função no centro c do hipercubo e faça $f_{min} = f(c)$.
   Inicialize a lista de retângulos restantes $L_p = \{X^0\}$.

2. Selecione os retângulos potencialmente ótimos conforme a Definição 2.4.1.

3. Para cada retângulo selecionado $X_i \in L_p$ avalie a função nos pontos $c_i \pm \delta e_j$, ao longo dos j maiores eixos de X_i. Divida o retângulo X_i ao longo dos maiores eixos por ordem crescente de função $f(c_i \pm \delta e_j)$.
   Se $f(c_i \pm \delta e_j) < f_{min}$ para algum j, faça $f_{min} = f(c_i \pm \delta e_j)$.
   Adicione os novos retângulos provenientes de X_i a lista e remova X_i.

4. Se o número de iterações ou de avaliações de função atingiu seu limite, pare. Caso contrário retorne ao passo 2.

2.5 Convergência

Se a função objetivo é contínua o método DIRECT tem convergência garantida ao minimizador global, [33]. Isso decorre do fato de que à medida que o número de iterações vai para infinito, o conjunto de pontos visitados pelo DIRECT forma um conjunto denso no hipercubo unitário. Assim dada uma distância $\delta > 0$, o DIRECT visitará um ponto $\bar{x}$, tal que $\|\bar{x} - x^*\| < \delta$, onde x^* é o minimizador global da função.

Teorema 2.5.1. *À medida que o número de iterações vai para infinito, o conjunto de pontos visitados pelo* DIRECT *forma um conjunto denso no hipercubo unitário.*

Prova. Iniciamos com o hipercubo unitário, no qual cada lado tem tamanho 1. Uma vez que os hiper-retângulos são gerados da divisão de hiper-retângulos já existentes (em três partes), os únicos tamanhos possíveis são 3^{-k}, para k natural. Mais ainda, como os hiper-retângulos são sempre divididos ao longo da maior dimensão, nenhum lado de tamanho $3^{-(k+1)}$ pode ser dividido até que todos os lados de tamanho 3^{-k} tenham sido. Segue que após m divisões, um dado hiper-retângulo terá $j \equiv m \pmod{n}$ lados de tamanho $3^{-(k+1)}$ e $n-j$ lados de tamanho 3^{-k}, onde $k = (m-j)/n$. A distância do centro aos vértices será

$$d = [j3^{-2(k+1)} + (n-j)3^{-2k}]^{1/2}/2.$$

À medida que o número de divisões tende a infinito, a distância do centro aos vértices de um dado hiper-retângulo tende a zero.
Agora, para garantir que todo hiper-retângulo será dividido infinitas vezes, basta lembrar da definição de hiper-retângulos potencialmente ótimos, e das observações seguintes a ela, que a prova segue por contradição. ■

2.6 Clusters ao redor de Minimizadores Globais

Nessa seção mostraremos que as subdivisões do DIRECT se tornam mais refinadas próximo a minimizadores globais, como em [16].

Para isso, considere a iteração k do método DIRECT. Seja I^k o conjunto dos índices dos hiper-retângulos existentes nesta iteração. Considere também o conjunto I^k_{min} definido por:

$$I^k_{min} = \left\{ i \in I^k \mid f(c_i) = f_{min} \right\},$$

onde f_{min} é o menor valor de função encontrado até então.

Seja também

$$w_k = \max_{i \in I^k_{min}} d_i$$

e

$$\hat{I}^k = \left\{ i \in I^k_{min} \mid d_i = w_k \right\}.$$

Pelo Lema 2.4.2 temos que todos os hiper-retângulos em $\hat{I}^k$ são potencialmente ótimos para ε suficientemente pequeno. Então, temos que a cada iteração do `DIRECT` pelo menos um dos seguintes fatos acontece:

- ou o melhor valor de função objetivo diminui $f^{k+1}_{min} < f^k_{min}$;
- ou $w_{k+1} < w_k$;
- ou o número de hiper-retângulos em I^{k+1}_{min} é maior que em I^k_{min}.

O Teorema 2.6.1 formaliza tal observação.

Teorema 2.6.1. *Seja $\varepsilon > 0$ suficientemente pequeno, tal que a segunda condição da Definição 2.4.1 seja satisfeita por todo $i \in \hat{I}^k$. Então para todo k temos que $i \in \hat{I}^k$ é potencialmente ótimo e ao menos uma das desigualdades*

$$f^{k+1}_{min} < f^k_{min} \tag{2.13}$$

$$w_{k+1} \leq w_k/3 \tag{2.14}$$

ou

$$|I^{k+1}_{min}| > |I^k_{min}| \tag{2.15}$$

é verificada.

Prova. Temos que os hiper-retângulos com índices $i \in \hat{I}^k$ sempre se encontram na parte inferior direita do envoltório convexo dos pontos que representam os hiper-retângulos existentes (ver Figura 2.5), logo cumprem a primeira condição de (2.4.1) e são candidatos a potencialmente ótimos. Mas com a hipótese sobre a constante ε, tais hiper-retângulos também cumprirão a segunda condição de 2.4.1, logo serão potencialmente ótimos.

Se $w_{k+1} > w_k/3$ então temos que f^{k+1}_{min} ocorreu em um hiper-retângulo resultante da divisão de um elemento que não estava em $\hat{I}^k$. Daí ou $f^{k+1}_{min} < f^k_{min}$, ou

então o menor valor continua o mesmo, e neste caso um novo hiper-retângulo com valor $f_{min}^{k+1} = f_{min}^{k}$ foi criado, logo $|\hat{I}^{k+1}| > |\hat{I}^{k}|$. Portanto o não cumprimento da desigualdade (2.14) implica no cumprimento da desigualdade (2.13) ou da desigualdade (2.15). ■

Este teorema nos diz que se $\bar{x}$ é minimizador global, então as divisões realizadas pelo `DIRECT` se tornarão mais refinadas próximo a este ponto. Os pontos visitados pelo `DIRECT` formarão um *cluster* ao redor de tal ponto.

De fato, observando os pontos visitados pelo `DIRECT` após algumas iterações, percebemos que o algoritmo é atraído pelos minimizadores globais da função. A Figura 2.10 mostra os pontos visitados pelo `DIRECT` quando aplicado sobre a função *Six Hump Camel Back*, [8], [33], no retângulo $[-1.9, 1.9] \times [-1.1, 1.1]$.

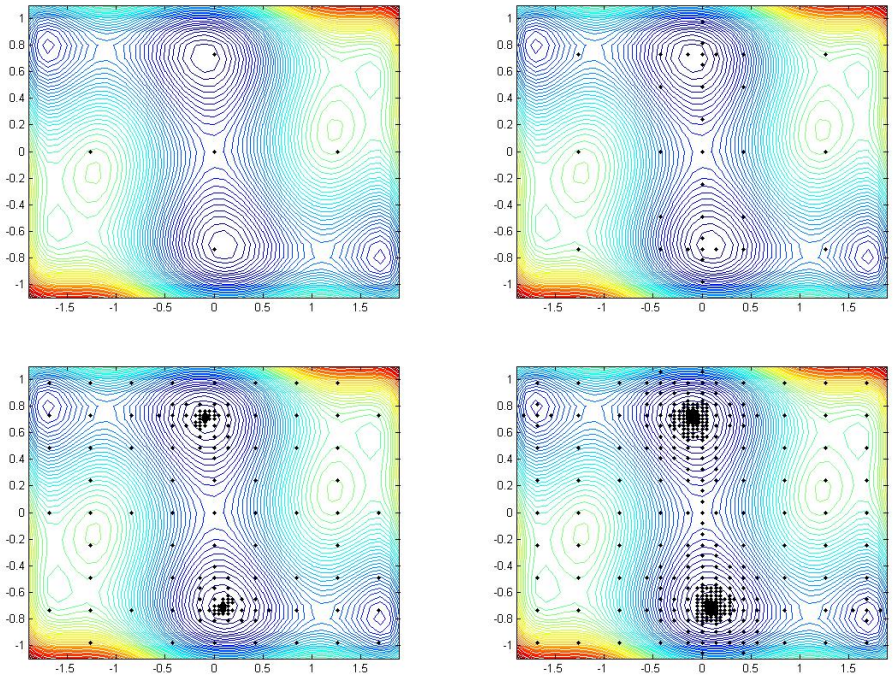

Figura 2.10: Pontos visitados no domínio original após 1, 4, 10 e 20 iterações do `DIRECT`

Note que em poucas iterações os pontos visitados pelo DIRECT se acumulam ao redor dos minimizadores globais da função.

2.7 Cluster cumprindo Condição de Otimalidade

Na seção anterior vimos que o DIRECT cria clusters ao redor de minimizadores globais. Nesta seção veremos que se o DIRECT cria um cluster ao redor de um ponto, tal ponto cumpre uma condição necessária de otimalidade.

Seja S_k o conjunto de pontos visitados pelo DIRECT até a k-ésima iteração. Assim temos que o conjunto de todos os pontos visitados pelo DIRECT é dado por

$$S = \cup_k S_k,$$

lembrando que S é denso no hipercubo unitário como vimos na seção acerca da convergência do DIRECT.

Considere agora o conjunto B_k, como sendo o conjunto dos melhores pontos na k-ésima iteração

$$B_k = \{x \in S_k \mid f(x) \leq f(z), \forall z \in S_k\},$$

e

$$B = \cup_k B_k,$$

o *conjunto dos melhores pontos* gerados pelo DIRECT.

O Teorema 2.7.1, apresentado por Finkel & Kelley [14] mostra que se $\bar{x}$ é um ponto de acumulação (cluster) do conjunto dos melhores pontos gerados pelo DIRECT então não existe direção factível e de descida a partir de $\bar{x}$.

Teorema 2.7.1. *Seja $\bar{x}$ um ponto de acumulação de B, e f Lipschitz contínua de constante L. Então não existe $d \in \mathbb{R}^n$ factível e de descida a partir de $\bar{x}$.*

Prova. Seja $\bar{x}$ ponto de acumulação de B. Suponha que exista $d \in \mathbb{R}^n$ tal que $\bar{y} = \bar{x} + td \in \Omega$ (o hipercubo unitário) para um certo $t \in (0, \delta]$, e $f(\bar{y}) < f(\bar{x})$.

Seja

$$\Delta = \min\left\{\frac{t}{2}, \frac{f(\bar{x}) - f(\bar{y})}{2L}\right\}$$

e

$$V(\bar{x}) = \{x \in \mathbb{R}^n | \, \|\bar{x} - x\| \leq \Delta\} \cap \Omega.$$

Como f é Lipschitz com constante L,

$$|f(x) - f(y)| \leq L \|x - y\|, \quad \forall x, y \in \Omega,$$

em particular temos,

$$|f(\bar{x}) - f(x)| \leq L \|\bar{x} - x\|, \quad \forall x \in V(\bar{x}).$$

Daí:

$$f(x) - f(\bar{y}) \geq f(\bar{x}) - L \|\bar{x} - x\| - f(\bar{y}) \geq f(\bar{x}) - L\Delta - f(\bar{y}),$$

logo,

$$f(x) - f(\bar{y}) \geq \frac{f(\bar{x}) - f(\bar{y})}{2} > 0, \quad \forall x \in V(\bar{x}).$$

Por outro lado S é denso em Ω, logo existe um k_0 tal que $\hat{y} \in S_{k_0}$ e,

$$\|\hat{y} - \bar{y}\| < \Delta/2.$$

Pela continuidade de f temos que $f(\hat{y}) < f(x)$ para todo $x \in V(\bar{x})$. Assim para $k > k_0$ temos que,

$$V(\bar{x}) \cap B_k = \emptyset.$$

Desse modo, temos que $V(\bar{x})$ conterá no máximo um número finito de pontos de B. Portanto existe um $\xi > 0$ tal que $\|\bar{x} - y\| > \xi$ para todo $y \in B$, o que contradiz o fato de $\bar{x}$ ser ponto de acumulação de B. ■

2.8 Sensibilidade ao Parâmetro ε

Inicialmente o parâmetro ε, foi introduzido com a finalidade de evitar uma ênfase demasiada na busca local. Testes foram realizados em problemas clássicos de otimização global, e variações deste parâmetro entre 10^{-7} e 10^{-2} não prejudicaram o desempenho do `DIRECT`. O valor de 10^{-4} foi sugerido em [33], por apresentar os resultados mais robustos na prática.

Todavia trabalhos posteriores [13], mostraram a sensibilidade do `DIRECT` mediante escalamento aditivo, isto é, se uma função em que o `DIRECT` se comporta bem, for modificada, digamos, pela adição de um termo de ordem 10^6, o algoritmo `DIRECT` pode apresentar um desempenho muito ruim.

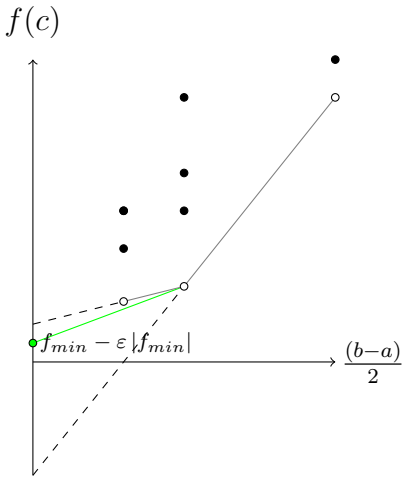

Figura 2.11: Efeito do parâmetro ε

Note pela Figura 2.11, que o menor retângulo, com menor valor de função (no caso f_{min}), não será potencialmente ótimo por não satisfazer a segunda condição, já que seu limitante inferior (onde a linha pontilhada intercepta o eixo vertical) se encontra acima do valor $f_{min} - \varepsilon\,|f_{min}|$.

De fato é esse o comportamento esperado com a segunda condição, que impede a escolha dos menores retângulos com menor valor de função objetivo, evitando assim que o algoritmo se torne muito local. Porém o parâmetro ε aparece multiplicando $|f_{min}|$, e se f_{min} tiver uma magnitude elevada, digamos, de ordem maior que 10^4, então a segunda condição se torna muito forte, reduzindo drasticamente a quantidade de hiper-retângulos potencialmente ótimos por iteração, provocando um desempenho fraco do `DIRECT`.

Para contornar esse problema, Finkel e Kelley [13] propuseram a seguinte alteração na segunda condição da definição (2.4.1)

$$f(c_j) - Kd_j \leq f_{min} - \varepsilon\,|f_{min} - f_{med}|$$

onde f_{med} é a mediana dos valores de função objetivo obtidos até o momento.

2.9 Comentários e Sugestões

Alguns testes computacionais realizados com DIRECT, atestam que este método é eficiente na resolução de problemas clássicos de otimização global, com restrições de canalização, em poucas variáveis, [14], [16], [33]. Nos testes realizados ficou constatado que DIRECT conseguiu obter os minimizadores globais com um número modesto de avaliações de função em relação a outros métodos, [20], [36].

Uma vantagem do DIRECT é que o método faz uso apenas de avaliações de função, sem a necessidade de derivadas ou mesmo do conhecimento de uma expressão explícita para função, o que torna o método atrativo para funções tipo caixa preta (*black-box*).

Mas temos um inconveniente em relação ao critério de parada. Como já comentamos, a ausência da constante de Lipschitz L não nos permite obter limitantes inferiores verdadeiros f_{lb}. Assim não podemos usar um critério de parada como no algoritmo de Shubert, $|f_{min} - f_{lb}| < \xi$, nem eliminar certos retângulos nos quais o mínimo global não pode ocorrer.

Como visto anteriormente, a convergência do método aos minimizadores globais é garantida, pois à medida que o número de iterações tende a infinito, o DIRECT visita um conjunto denso de pontos no hipercubo unitário. Porém na prática estabelecemos um limite para o número de iterações ou avaliações de função, e esperamos que dentro deste limite o DIRECT seja capaz de encontrar um ponto suficientemente próximo do minimizador global.

Vimos também que os clusters criados pelo DIRECT cumprem uma certa condição de otimalidade. Assim se nos contentamos com uma solução local, podemos interromper o DIRECT logo que detectada a formação de um cluster. Isso pode ser feito monitorando as distâncias d_i do centro ao vértice dos retângulos. Se para algum i, $d_i < \varepsilon_X$ então interrompemos a execução do DIRECT e retornamos o menor valor f_{min} encontrado até o momento.

Com isso o DIRECT pode ser utilizado como um gerador de pontos iniciais, identificando regiões promissoras para utilização de um método de busca local.

Capítulo 3

Análise Intervalar e Envelopes Convexos

Inicialmente o propósito da análise intervalar foi fornecer um meio de estimar limitantes inferiores e superiores sobre o efeito causado por erros de arredondamento, incerteza nos dados ou aproximações, ao se calcular uma certa quantidade. Vários pesquisadores tiveram independentemente a idéia de limitar os erros de arredondamento através do cálculo de intervalos [11], [67]. Uma forte referência para análise intervalar foi o trabalho de R. E. Moore, *Interval Analysis* em 1966 [50], que transformou essa simples idéia em uma valiosa ferramenta para análise de erros. Ao invés de meramente tratar erros de arredondamento, ele estendeu o uso da análise intervalar para tratar o efeito de erros de todos os tipos, inclusive erros de aproximação e incerteza nos dados.

Porém a análise intervalar tem outras aplicações além da análise de erros, e uma das mais interessantes para a otimização global [25] é a possibilidade de se calcular limitantes sobre um número infinito de dados (intervalo).

Por exemplo, considere uma função $f : \mathbb{R} \to \mathbb{R}$, $f \in \mathbb{C}^1$. Pelo Teorema do Valor Médio

$$f(x+h) = f(x) + f'(\xi)h \qquad \text{, com } \xi \in [x, x+h].$$

Como obter $f(x+h)$ desconhecendo ξ?

A análise intervalar auxilia esta e outras questões interessantes, como veremos a seguir.

3.1 Aritmética Intervalar

A análise intervalar baseia-se na extensão do conceito de *número real*, encarando um *intervalo* de números reais como um *novo tipo de número* e para combiná-los, uma nova aritmética, que é uma generalização da aritmética usual: *a aritmética intervalar*.

3.1.1 Números Intervalares e Notações

Neste seção apresentamos as definições e a notação que iremos utilizar. Para mais referências sobre aritmética intervalar ver [25], [50].

Definição 3.1.1 (Número intervalar)**.** *Considere um intervalo real fechado* $X = [a, b]$*. Um número intervalar* X *é tal intervalo fechado, ou seja, consiste do conjunto* $\{x : a \leq x \leq b\}$ *dos números reais que se encontram entre* a *e* b*.*

Um *número intervalar* é também chamado simplesmente de *intervalo*. Um número real x equivale ao intervalo *degenerado* $[x, x]$.

Note contudo, que os extremos de um intervalo $[a, b]$ podem não ser representáveis pela máquina, neste caso o substituímos por $[\bar{a}, \bar{b}]$, onde $\bar{a}$ é o maior número representável pela máquina menor ou igual a a e $\bar{b}$ o menor número representável pela máquina maior ou igual a b.

Com relação à notação usaremos letras maiúsculas para intervalos (números intervalares), e o super-índice I para indicar que se trata de uma quantidade intervalar. Por exemplo, quando V denotar uma quantidade real, V^I denotará um intervalo. Quando for claro que a quantidade em questão é um intervalo suprimimos o uso de I. Por exemplo, se $f(x)$ é uma função real de uma variável real x, denotaremos f com argumento intervalar X por $f(X)$ no lugar de $f^I(X)$, pois a presença do argumento intervalar é suficiente para indicar que $f(X)$ é um intervalo, como veremos mais adiante.

Um intervalo $X = [a, b]$ é dito *positivo* se $a > 0$ e *negativo* se $b < 0$. Dois intervalos $[a, b]$ e $[c, d]$ são iguais se e só se $a = c$ e $b = d$. Os números intervalares são parcialmente ordenados, isto é, temos que $[a, b] < [c, d]$ se e somente se $b < c$. Denotamos por $\mathbb{I}$ o conjunto de todos os intervalos reais.

Também definimos *vetores intervalares* como sendo vetores cujas componentes são intervalos. Neste caso se Y denota um vetor intervalar (ou simplesmente *caixa*), então cada componente Y_i é um intervalo, e ainda dizemos que $Y \in \mathbb{I}^n$, onde $\mathbb{I}^n = \mathbb{I} \times \mathbb{I} \times ... \times \mathbb{I}$.

Analogamente trabalhamos também com *matrizes intervalares*. Se A denota uma matriz intervalar, então A_{ij} é um intervalo. Dizemos também que $\mathbb{I}^{m\times n}$ é o espaço das matrizes intervalares de ordem $m \times n$.

Seja $S \subset \mathbb{R}^n$. Denotamos por $I(S)$, o conjunto de todas as caixas $Y \in \mathbb{I}^n$ tais que $Y \subset S$.

Por fim, seja $f : S \to \mathbb{R}$, com $S \subset \mathbb{R}^n$. Seja Y um intervalo tal que $Y \subset S$. Então a imagem de f sobre Y será denotada por $\square f(Y)$, isto é:

$$\square f(Y) = \{f(x),\ x \in Y\}.$$

Há também algumas funções reais de intervalos, que serão utilizadas ao longo desse texto.

Seja $X = [a, b]$ um intervalo real. O *ponto médio* ou *centro* de X é dado por:

$$m(X) = \frac{a+b}{2},$$

a *largura* de X é:

$$w(X) = b - a,$$

e o *valor absoluto* de X é:

$$|X| = \max\{|a|, |b|\}.$$

O valor absoluto também é chamado de *magnitude*. Por outro lado o *menor* valor de $|x|$ para todo $x \in X$ é definido por:

$$mig(X) = \begin{cases} a & \text{se } a > 0, \\ -b & \text{se } b < 0, \\ 0 & \text{caso contrário.} \end{cases}$$

Para um dado intervalo X, também podemos denotar seus limitantes inferiores e superiores por $\underline{X}$ e $\overline{X}$ respectivamente, isto é, $X = [\underline{X}, \overline{X}]$.

3.1.2 Aritmética Intervalar

Se o símbolo $\circ$ denota alguma das operações aritméticas elementares para números reais, e X e Y números intervalares, a operação correspondente na aritmética intervalar é definida por:

$$X \circ Y = \{x \circ y : x \in X, y \in Y\}.$$

Assim, o intervalo resultante da operação $X \circ Y$ contém todo número (real) obtido por $x \circ y$, para cada $x \in X$ e $y \in Y$.

Dessa definição temos as seguintes regras para obter os extremos do intervalo $X \circ Y$ de dois intervalos $X = [a, b]$ e $Y = [c, d]$:

$$\begin{aligned}
X + Y &= [a + c, b + d] \\
X - Y &= [a - d, b - c] \\
X * Y &= \begin{cases}
[ac, bd] & \text{se } a \geq 0 \text{ e } c \geq 0 \\
[bc, bd] & \text{se } a \geq 0 \text{ e } c < 0 < d \\
[bc, ad] & \text{se } a \geq 0 \text{ e } d \leq 0 \\
[ad, bd] & \text{se } a < 0 < b \text{ e } c \geq 0 \\
[bc, ac] & \text{se } a < 0 < b \text{ e } d \leq 0 \\
[ad, bc] & \text{se } b \leq 0 \text{ e } c \geq 0 \\
[ad, ac] & \text{se } b \leq 0 \text{ e } c < 0 < d \\
[bd, ac] & \text{se } b \leq 0 \text{ e } d \leq 0 \\
[\min(bc, ad), \max(ac, bd)] & \text{se } a < 0 < b \text{ e } c < 0 < d
\end{cases} \\
\frac{1}{Y} &= \left[\frac{1}{d}, \frac{1}{c}\right] \qquad (0 \notin Y) \\
\frac{X}{Y} &= X * \left(\frac{1}{Y}\right) \qquad (0 \notin Y)
\end{aligned}$$

Também definimos

$$X^n = \begin{cases}
[1, 1] & \text{se } n = 0 \\
[a^n, b^n] & \text{se } a \geq 0 \text{ ou se } a \leq 0 \leq b \text{ e } n \text{ é ímpar} \\
[b^n, a^n] & \text{se } b \leq 0 \\
[0, \max(a^n, b^n)] & \text{se } a \leq 0 \leq b \text{ e } n \text{ é par}
\end{cases}$$

Nas regras acima, excluímos a divisão por um intervalo contendo zero. Mas em certos casos é útil remover tal restrição, e temos como resultado uma *aritmética*

intervalar estendida, [28], [34].

Sejam a, b, c e d valores reais finitos, com $c \leq 0 \leq d$ e $c < d$, $X = [a, b]$ e $Y = [c, d]$. As regras de divisão por um intervalo contendo zero são as seguintes:

$$\frac{X}{Y} = \begin{cases} [b/c, \infty) & \text{se } b \leq 0 \text{ e } d = 0 \\ (-\infty, b/d] \cup [b/c, \infty) & \text{se } b \leq 0 \text{ e } c < 0 < d \\ (-\infty, b/d] & \text{se } b \leq 0 \text{ e } c = 0 \\ (-\infty, \infty) & \text{se } a < 0 < b \text{ e } c < 0 < d \\ (-\infty, a/c] & \text{se } a \geq 0 \text{ e } d = 0 \\ (-\infty, a/c] \cup [a/d, \infty) & \text{se } a \geq 0 \text{ e } c < 0 < d \\ [a/d, \infty) & \text{se } a \geq 0 \text{ e } c = 0 \end{cases}$$

3.1.3 O Problema de Dependência

Temos alguns inconvenientes ao trabalhar com a aritmética intervalar. Suponha por exemplo, a operação de subtrair o intervalo $X = [a, b]$ dele mesmo. Pela aritmética intervalar definida, teremos como resultado $[a-b, b-a]$. Isto é, a menos que $a = b$, o resultado dessa subtração será diferente de $[0, 0]$, como seria desejável. Na verdade o resultado é o conjunto $\{x - y : x \in X, y \in Y\}$ ao invés de $\{x - x : x \in X\}$.

Em geral, quando uma variável aparece mais de uma vez em uma expressão intervalar, esta é tratada como uma variável diferente a cada ocorrência. Assim $X - X$ é o mesmo que $X - Y$, com Y igual mas independente de X. Isso pode causar a chamada *largura excessiva* nos intervalos calculados e dificultar a obtenção de bons resultados nos cálculos intervalares.

Para ilustrar esse fato, considere $X = [-1, 2]$. Segundo a definição de potência dada anteriormente temos que $X^2 = [-1, 2]^2 = [0, 4]$, enquanto $X * X = [-1, 2] * [-1, 2] = [-2, 4]$.

Note que, se uma dada variável intervalar ocorre apenas uma vez na expressão então o problema de dependência não ocorre. Logo, o problema de dependência pode ocorrer por exemplo em $(X - Y)/(X + Y)$, mas não se reescrevermos a expressão como $1 - 2/(1 + X/Y)$.

Estudos em análise intervalar foram realizados com o intuito de reduzir o efeito

do problema de dependência nos cálculos intervalares. Dentre eles podemos citar a *Aritmética Intervalar Generalizada* proposta por Hansen [23] e *Aritmética Afim* proposta por Stolfi et al [12].

3.2 Funções Intervalares

Definição 3.2.1 (Função Intervalar)**.** *Uma função é dita intervalar quando um ou mais argumentos são intervalos, e cuja imagem é também um intervalo.*

Considere uma *função intervalar* $F : \mathbb{I}^n \to \mathbb{I}$, onde $\mathbb{I}$ denota o conjunto dos intervalos reais e $\mathbb{I}^n = \mathbb{I} \times \mathbb{I} \times ... \times \mathbb{I}$. Desse modo, F é função das variáveis intervalares (intervalos) $X_1, X_2, ..., X_n$, e $F(X)$ é também um intervalo.

3.2.1 Extensões Intervalares e Monotonicidade Inclusiva

Seja agora $f : S \to \mathbb{R}$ uma função real, com $S \subset \mathbb{R}^n$.

Definição 3.2.2 (Extensão Intervalar)**.** *Dizemos que F é uma extensão intervalar de f se:*

$$f(x_1, x_2, ..., x_n) = F(x_1, x_2, ..., x_n).$$

Isso significa que se os argumentos de F são intervalos degenerados ($X_i = [x_i, x_i]$), então $F(X_1, X_2, ..., X_n)$ é um intervalo degenerado que coincide com $f(x_1, x_2, ..., x_n)$.

Dada uma função racional real f, para obter uma extensão intervalar, basta substituir as variáveis reais por variáveis intervalares, e a aritmética real pela aritmética intervalar. A essa extensão intervalar denominamos *extensão intervalar natural*.

Lembrando que $I(S)$ denota todas as caixas $X \in \mathbb{I}^n$ tais que $X \subset S$, temos outra definição importante:

Definição 3.2.3 (Função de Inclusão)**.** *Uma função intervalar F é dita função de inclusão para $f : S \to \mathbb{R}$, se:*

$$\square f(X) \subset F(X)$$

para todo $X \in I(S)$.

Definição 3.2.4. *Dizemos que a função intervalar F, é monótona inclusiva, se $X_i \subset Y_i$ para todo i, implicar que:*

$$F(X_1, X_2, ..., X_n) \subset F(Y_1, Y_2, ..., Y_n).$$

Segue da definição que as operações na *aritmética intervalar* são monótonas inclusivas, ou seja, se $X_1 \subset Y_1$ e $X_2 \subset Y_2$, e $\circ$ denota uma das operações + , - , * , /, então:

$$(X_1 \circ X_2) \subset (Y_1 \circ Y_2).$$

Usaremos $f(X)$ para denotar a *extensão intervalar natural* de uma função de variáveis reais f.

Teorema 3.2.5. *Seja $F(X_1, X_2, ..., X_n)$ uma função intervalar racional, e assuma que F é avaliada de uma forma fixada. Então F é monótona inclusiva.*

A prova deste teorema será omitida, mas decorre do fato das operações em aritmética intervalar serem monótonas inclusivas. Além disso a hipótese da função ser avaliada de forma fixada, significa que $f(x) = x(1 - x)$ é diferente de $f(x) = c(1-c)+(1-2c)(x-c)-(x-c)^2$ quanto ao resultado na avaliação de suas extensões intervalares. Para mais detalhes e demonstração completa ver [25].

Funções irracionais são tratadas da seguinte maneira. Seja $f : S \to \mathbb{R}$ uma função real irracional nas variáveis $x_1, x_2, ..., x_n$. Considere uma aproximação racional $r(x)$ para $f(x)$ tal que

$$|f(x) - r(x)| < \varepsilon$$

para todo $x \in S$. Então

$$f(X_1, X_2, ..., X_n) \subset r(X_1, X_2, ..., X_n) + [-\varepsilon, \varepsilon]$$

para todo intervalo $X \subset S$. Desse modo a imagem de f sobre um intervalo qualquer $X \subset S$ pode ser limitada pela avaliação de $r(X_1, ..., X_n)$ mais um limitante para erro de aproximação $[-\varepsilon, \varepsilon]$.

Já para funções monótonas, limitantes precisos podem ser obtidos através da avaliação da função real nos extremos do intervalo em questão. Por exemplo se $X = [a, b]$ e f é não decrescente, então $\square f(X) \subset [f(a), f(b)]$.

O próximo teorema é um dos mais importantes em análise intervalar [59], pois assegura que a imagem de uma função real sobre um dado intervalo pode ser limitada pela imagem de sua extensão intervalar. Este fato é relevante num contexto de otimização global pois permite estimar limitantes para os valores que a função objetivo pode assumir em um dado intervalo.

Teorema 3.2.6. *Seja $F(X_1, ..., X_n)$ uma extensão intervalar monótona inclusiva de um função real $f(x_1, ..., x_n)$, que toma valores em uma caixa $S \subset \mathbb{R}^n$. Então $F(X_1, ..., X_n)$ é uma função de inclusão para f sobre S.*

Prova. Seja $x \in S$. Pela monotonicidade inclusiva temos que $F([x_1, x_1], ..., [x_n, x_n]) \subset F(X_1, ..., X_n)$, pelo fato de F ser uma extensão intervalar para f, $f(x_1, ..., x_n) \in F([x_1, x_1], ..., [x_n, x_n])$. Como tomamos x arbitrário em S, temos que para todo $x \in S$, $f(x_1, ..., x_n) \in F(X_1, ..., X_n)$. ■

3.2.2 Formas de Funções Intervalares

O intervalo obtido quando avaliamos uma função intervalar depende da *forma* da função (devido ao problema de dependência). Por exemplo, enquanto

$$F_1(X) = X^2 - X \quad \text{e} \quad F_2(X) = (X - \frac{1}{2})^2 - \frac{1}{4}$$

são ambas funções de inclusão para

$$f(x) = x^2 - x = (x - \frac{1}{2})^2 - \frac{1}{4}$$

elas não produzem o mesmo resultado quando avaliadas. Neste caso

$$F_1([0, 2]) = [-2, 4] \quad \text{e} \quad F_2([0, 2]) = [-\frac{1}{4}, 2].$$

Ambas contêm a imagem de f sobre $[0, 2]$ que é $[-\frac{1}{4}, 2]$, como esperávamos pelo Teorema 3.2.6. Porém o resultado de F_2 é exatamente a imagem de f sobre $[0, 2]$, enquanto o resultado de F_1 não é tão preciso assim.

Uma medida de quão boa é uma função de inclusão para uma função f em uma certa caixa X é dada pelo que chamamos de *largura excessiva*:

$$w(F(X)) - w(\square f(X)),$$

onde $F(X)$ denota uma função de inclusão para f, e $\square f(X)$ denota a imagem de f sobre o intervalo X. A função largura, $w(.)$, para um vetor intervalar $X = ([a_1, b_1], ..., [a_n, b_n])$ significa:

$$w(X) = \max_{1 \leq i \leq n} (b_i - a_i).$$

Definição 3.2.7 (Ordem de convergência para funções de inclusão)**.** *Dizemos que uma função de inclusão F para f tem ordem de convergência $\alpha > 0$ se*

$$w(F(X)) - w(\square f(X)) = O(w(X)^\alpha).$$

A ordem de convergência expressa a velocidade com que a largura excessiva vai a zero à medida em que a largura do intervalo é reduzida.

Em [50] está demonstrado que a extensão intervalar natural de uma função real racional f é de ordem 1, isto é:

$$w(f(X)) - w(\square f(X)) = O(w(X)).$$

3.2.3 Formas Centradas

Em [50] temos também que dada $f(x_1, x_2, ..., x_n)$ a mesma pode ser escrita como:

$$f_c(x_1, ..., x_n) = f(c_1, ..., c_n) + g(x_1 - c_1, ..., x_n - c_n)$$

para alguma função racional g, onde $c_i = m(X_i)$ (ponto médio do intervalo X_i). Neste caso temos que:

$$w(f_c(X)) - w(\square f(X)) = O(w(X)^2).$$

A forma f_c, denominada *forma centrada*, tem largura excessiva de *segunda ordem* em relação à largura do intervalo X.

Várias formas centradas podem ser derivadas por meio de expansões apropriadas. As mais úteis para nossos fins são as formas centradas *do Valor Médio* e de *Taylor*.

Seja $f : S \to \mathbb{R}$, com $S \subset \mathbb{R}^n$, $f \in \mathbb{C}^1$, e $F' : I(S) \to \mathbb{I}^n$ uma função de inclusão para ∇f. Então $T_1 : I(S) \to \mathbb{I}$ definida por:

$$T_1(X) = f(c) + F'(X)^t (X - c),$$

onde $c = m(X)$, é chamada *forma centrada do Valor Médio.*

Note que $F'(X)$ pode ser obtida pela extensão intervalar das derivadas parciais, ou ainda por diferenciação automática, como citado em [25]. Relacionando com o Teorema do Valor Médio de análise em $\mathbb{R}^n$, dado $x \in X$ tal que $X \subset I(S)$ temos que:

$$f(x) = f(c) + \nabla f(\xi)^t(x - c) \in f(c) + F'(X)^t(x - c),$$

ou ainda

$$\square f(X) \subset f(c) + F'(X)^t(X - c) = T_1(X).$$

Além disso, sob certas hipóteses adicionais temos que T_1 é convergente de ordem 2.

Definição 3.2.8 (Função Intervalar Lipschitz)**.** *Uma função intervalar $F : I(S) \to \mathbb{I}$, é dita Lipschitz, se para todo $X \in I(S)$ existe uma constante $L > 0$ tal que:*

$$w(F(X)) \leq Lw(X).$$

Teorema 3.2.9. *Considerando as hipóteses anteriores da definição de T_1, se F' é Lipschitz, então T_1 é convergente de ordem 2.*

Prova. ver [39], [50].

Seja agora $f : S \to \mathbb{R}$, $S \subset \mathbb{R}^n$, $f \in \mathbb{C}^2$ e $F'' : I(S) \to \mathbb{I}^{n\times n}$ uma função de inclusão para a $\nabla^2 f$. Então $T_2 : I(S) \to \mathbb{I}$ definida por:

$$T_2(X) = f(c) + \nabla f(c)^t(X - c) + \frac{1}{2}(X - c)^t F''(X)(X - c),$$

é denominada *forma centrada de Taylor* (de ordem 2).

Dizemos que F'' é limitada se $\exists\, A \in \mathbb{I}^{n\times n}$ tal que $F''(X) \subset A$, para todo $X \in I(S)$.

Teorema 3.2.10. *Se além das hipóteses da definição de T_2, F'' for limitada, então T_2 é convergente de ordem 2.*

Prova. ver [50].

3.2.4 Forma Inclinação

Além das formas centradas de Taylor e do Valor Médio, existem funções de inclusão que remetem à idéia de aproximar as derivadas por diferenças finitas, [38].

Considere $f : \mathbb{R} \to \mathbb{R}$ racional, e a seguinte identidade:

$$f(x) - f(c) = g(c, x)(x - c).$$

Resolvendo para $g(c, x)$ vemos que

$$g(c, x) = \frac{f(x) - f(c)}{x - c}.$$

Denominamos $g(c, x)$ de *função inclinação*, e seu limite quando $x \to c$, aproxima a inclinação de f em c. É claro que, sendo f racional, podemos dividir analiticamente $f(x) - f(c)$ por $x - c$ e obter $g(c, x)$ explicitamente.

Considere agora um intervalo real X, tal que $c \in X$, por exemplo $c = m(X)$.

Então para qualquer $x \in X$ temos que

$$f(x) \in f(c) + g(c, X)(x - c).$$

Comparando com a forma do valor médio:

$$f(x) \in f(c) + f'(X)(x - c),$$

onde $f'(X)$ é a extensão intervalar de $f'(x)$ em X, vemos que a forma inclinação desempenha um papel de uma expansão de primeira ordem.

Segundo Hansen, [25], algumas ocorrências do intervalo X na avaliação de $f'(X)$ podem ser substituídas pelo intervalo degenerado c quando avaliamos $g(c, X)$.

Com isso podemos reduzir o problema de dependência e obter melhores limitantes usando a forma inclinação.

3.2.5 Calculando as Formas Centradas

Seja f uma função real de n variáveis. Vimos pela forma centrada do Valor Médio que:

$$f(x) \in f(c) + (x - c)^T g(X_1, X_2, ..., X_n),$$

onde $g(X_1, X_2, ..., X_n)$ é uma função de inclusão para ∇f.

Note que os argumentos de g são todos intervalares. Alguns pesquisadores [22], [63], mostraram que alguns desses argumentos podem ser trocados por quantidades reais, e que isso pode gerar limitantes mais precisos.

Ilustraremos essa idéia para $n = 3$. Primeiro pensamos em $f(x_1, x_2, x_3)$ como uma função de x_3 apenas. Daí,

$$f(x_1, x_2, x_3) = f(x_1, x_2, c_3) + (x_3 - c_3)g_3(x_1, x_2, \xi_3).$$

Expandindo agora $f(x_1, x_2, c_3)$ como função de x_2 temos:

$$f(x_1, x_2, c_3) = f(x_1, c_2, c_3) + (x_2 - c_2)g_2(x_1, \xi_2, c_3).$$

E por fim, expandindo $f(x_1, c_2, c_3)$ em função de x_1:

$$f(x_1, c_2, c_3) = f(c_1, c_2, c_3) + (x_1 - c_1)g_1(\xi_1, c_2, c_3).$$

Desse modo temos que:

$$\begin{aligned} f(x_1, x_2, x_3) &= f(c_1, c_2, c_3) + (x_1 - c_1)g_1(\xi_1, c_2, c_3) + \\ &\quad (x_2 - c_2)g_2(x_1, \xi_2, c_3) + (x_3 - c_3)g_3(x_1, x_2, \xi_3), \end{aligned}$$

com $\xi_i \in X_i$ para $i = 1, 2, 3$. Então

$$\begin{aligned} f(x) &\in f(c) + (x_1 - c_1)g_1(X_1, c_2, c_3) + (x_2 - c_2)g_2(X_1, X_2, c_3) + \\ &\quad (x_3 - c_3)g_3(X_1, X_2, X_3). \end{aligned}$$

Para n variáveis temos a expressão

$$f(x) \in f(c) + \sum_{i=1}^{n}(x_i - c_i)g_i(X_1, X_2, \ldots, X_i, c_{i+1}, \ldots, c_n).$$

Considere agora a forma de Taylor de ordem 2

$$f(x) \in f(c)(x - c)^T g(c) + \frac{(x - c)^T H(x, X)(x - c)}{2},$$

onde $H(c, X)$ é uma função de inclusão para a Hessiana.

Procedendo de forma análoga para que $H(c, X)$ contenha um número menor de entradas intervalares, tomamos para $n = 3$,

$$H(c,X) = \begin{bmatrix} h_{11}(X_1, c_2, c_3) & 0 & 0 \\ h_{21}(X_1, c_2, c_3) & h_{22}(X_1, X_2, c_3) & 0 \\ h_{31}(X_1, c_2, c_3) & h_{32}(X_1, X_2, c_3) & h_{33}(X_1, X_2, X_3) \end{bmatrix},$$

onde

$$h_{ij} = \begin{cases} \partial^2 f/\partial x_i^2 & \text{para } j = i \\ 2\partial^2 f/\partial x_i \partial x_j & \text{para } j < i \\ 0 & \text{caso contrário.} \end{cases}$$

3.3 Sistemas de Equações Lineares Intervalares

Denotemos por $A^I \in \mathbb{I}^{m\times n}$ uma matriz intervalar e $b^I \in \mathbb{I}^m$ um vetor intervalar. A solução do sistema linear intervalar

$$A^I x = b^I \tag{3.1}$$

é definida pelo conjunto

$$S = \left\{ x \in \mathbb{R}^n : Ax = b, A \in A^I, b \in b^I \right\}.$$

Normalmente tal conjunto *não* é um vetor intervalar. De fato, S pode ser difícil de descrever, e quando resolvemos (3.1) por métodos intervalares, estamos interessados na verdade em obter a menor caixa X que contém S.

Note que A^I pode conter alguma matriz A singular. Logo o sistema linear $Ax = b$ pode não ter solução, ou ainda ter infinitas soluções. Neste último caso temos que o conjunto solução do sistema linear intervalar $A^I x = b^I$ é ilimitado.

Definição 3.3.1. *Uma matriz intervalar A^I é dita regular, se toda matriz $A \in A^I$ é não singular.*

Note que se A^I é regular então temos que $S \subset A^H b^I$, onde A^H é a matriz intervalar que contém as inversas A^{-1} para toda $A \in A^I$.

3.3.1 Eliminação Gaussiana

Podemos aplicar eliminação Gaussiana na resolução de (3.1). Na versão intervalar da eliminação Gaussiana substituímos a aritmética real pela aritmética intervalar. Porém o método pode falhar caso tenhamos uma divisão por um intervalo contendo zero.

Esse método não é muito utilizado na prática, já que os erros de arredondamento e problemas de dependência produzem limitantes muito grandes.

3.3.2 Precondicionamento

Uma idéia para contornar o mal desempenho na eliminação Gaussiana é usar como matriz de precondicionamento

$$B \approx m(A^I)^{-1},$$

isto é, uma aproximação para a inversa do ponto médio de A^I. Em seguida resolvemos o sistema

$$M^I x = r^I, \tag{3.2}$$

onde $M^I = BA^I$ e $r^I = Bb^I$.

Caso $m(A^I)$ seja singular possivelmente o sistema (3.1) pode ser ilimitado. Mas se for de real interesse obter uma matriz de precondicionamento, podemos perturbar $m(A^I)$ para que a mesma se torne não singular.

A eliminação Gaussiana intervalar com precondicionamento chega a ser seis vezes mais cara (computacionalmente) que a eliminação Gaussiana convencional. Todavia este é o preço pago por limitantes mais precisos.

3.3.3 Método de Krawczyk

Seja $A \in A^I$ e $b \in b^I$. O método de Krawczyk baseia-se na seguinte equação de *ponto fixo*:

$$x \;=\; Bb + (I - BA)x, \tag{3.3}$$

onde B é uma matriz de precondicionamento, normalmente tomada como $m(A^I)^{-1}$.

Note que se A é não singular então $x = A^{-1}b$ é ponto fixo da equação (3.3).

Daí segue que, se A^I é regular, e $X \supset A^H b^I$ então:

$$A^H b^I \subset Bb^I + (I - BA^I)X \cap X.$$

Assim dada uma estimativa inicial X^0 que contenha o conjunto solução do sistema linear intervalar, como em [7], definimos o passo iterativo:

$$\begin{aligned} Z^k &= Bb^I + (I - BA^I)X^k \\ X^{k+1} &= X^k \cap Z^k. \end{aligned}$$

Uma das vantagens do método de Krawczyk é que cada iteração tem como custo apenas a multiplicação intervalar de matriz por vetor.

3.3.4 Método de Gauss-Seidel

Caso tenhamos uma boa aproximação para a solução de (3.1) e uma matriz de precondicionamento, podemos resolver (3.2) de maneira mais eficiente do que com a eliminação Gaussiana.

A versão intervalar do método de Gauss-Seidel foi estudada inicialmente por Alefeld e Herzberger, [2]. Hansen e Sengupta,[24], estenderam o método para casos em que existem elementos da diagonal contendo zero.

Na i-ésima iteração calculamos

$$Y_i = \frac{1}{m_{ii}^I}(r_i^I - \sum_{j \neq i} m_{ij}^I X_j) \tag{3.4}$$

Em seguida fazemos a interseção

$$X_i' = X_i \cap Y_i \tag{3.5}$$

e usamos $X_i = X_i'$ para a próxima iteração analogamente ao método convencional.

Note que m_{ii}^I pode conter zero. Caso isso ocorra, a divisão em (3.4) pode resultar em Y_i não finito. Nestes casos procedemos da seguinte forma:

1. se o numerador também contém zero, então temos que $Y_i = (-\infty, \infty)$, logo $X_i' = X_i$. Ao detectar zero tanto no numerador quanto no denominador, não tentamos calcular Y_i, simplesmente passamos para o próximo i;

2. se o zero encontra-se somente no denominador, então Y_i pode ser composto de um intervalo semi-infinito (caso zero seja um extremo de m_{ii}^I), ou composto por dois intervalos semi-infinitos (caso zero esteja no interior de m_{ii}^I).

Analisando a segunda opção, se $X_i \cap Y_i$ é vazio, provamos que a equação (3.2) não tem solução, logo a equação original (3.1) também não. Consideremos de agora em diante que $X_i \cap Y_i \neq \emptyset$. Suponha que X_i' é composto por dois intervalos. Então X_i' é na verdade o intervalo original X_i com um *gap*, isto é $X_i \setminus Y_i^C$. Para não ser obrigado a carregar informações sobre o *gap*, e como estamos interessados apenas em uma caixa que contenha o conjunto solução S, ignoramos o *gap* e usamos X_i ao invés de X_i'.

Não usamos os procedimentos (3.4) e (3.5) em uma ordem sequencial, para $i = 1, 2, ..., n$, como no método convencional. Se $0 \notin m_{ii}^I$ então X_i' pode estar estritamente contido em X_i. Se este for o caso então melhores resultados podem ser obtidos de (3.4) para outros subsequentes valores de i.

Portanto, primeiro resolvemos Y_i para os valores de i tais que $0 \notin m_{ii}^I$, e só então resolvemos para os valores restantes de i.

Definição 3.3.2 (Matriz intervalar diagonalmente dominante)**.** *Uma matriz intervalar $A^I \in \mathbb{I}^{n \times n}$ é diagonalmente dominante se*

$$mig(a_{ii}^I) \geq \sum_{j \neq i}^{n} \left| a_{ij}^I \right|$$

para todo $i = 1, 2, ..., n$.

Se os elementos fora da diagonal de M^I são intervalos grandes em relação aos elementos da diagonal, uma única iteração do método Gauss-Seidel não dará bons resultados. Todavia neste caso o próprio conjunto solução será grande e há pouco a se ganhar obtendo limitantes justos na solução.

Por outro lado, se os elementos fora da diagonal forem intervalos pequenos em

relação aos elementos da diagonal, um passo do método de Gauss-Seidel intervalar gera resultados relativamente bons. É claro que mais de uma iteração do método pode ser feita. Na verdade podemos iterar até que não ocorra mais melhoria nos limitantes. O método pode parar após um número finito de iterações.

Seja λ_i denotando um autovalor de uma matriz real M de ordem n. O raio espectral de M é definido por:

$$\rho(M) = \max_{1 \leq i \leq n} |\lambda_i| .$$

O raio espectral de uma matriz intervalar M^I é:

$$\rho(M^I) = \max_{M \in M^I} \rho(M).$$

O método iterativo de Gauss-Seidel em sua versão intervalar converge se $\rho(\left|M^I\right|) < 1$, onde $\left|M^I\right|$ denota a matriz cujos elementos são valores absolutos dos elementos de M^I. Veja [3].

Se o vetor original X usado em (3.4) contém a solução de (3.2), então o novo vetor obtido após uma iteração também conterá a solução. Por outro lado, se para algum i a interseção $X_i' \cap Y_i$ for vazia, então isso prova que o vetor original X não continha a solução.

Uma escolha clássica para a matriz de precondicionamento como já vimos é tomar $B = m(A^I)^{-1}$, a inversa do ponto médio de A^I. Embora esta seja a escolha *ótima* para o método de Krawczyk, como discutido em [35], o método de Gauss-Seidel fornece sempre melhores resultados, ver [7]. Portanto este resultado de otimalidade deve ser interpretado de maneira cuidadosa. Em [35], Kearfott apresenta uma teoria sobre precondicionadores ótimos para o método de Gauss-Seidel obtidos através da resolução de problemas de programação linear.

3.4 Sistemas de Equações Não Lineares

Suponha que estamos interessados em encontrar *todos* os zeros de uma função continuamente diferenciável $f : \mathbb{R} \to \mathbb{R}$ em um intervalo fechado X^0. Para tanto estudaremos o *método de Newton Intervalar*, que posteriormente estenderemos para a resolução de um sistema de equações não-lineares.

A seguir deduziremos o método de Newton Intervalar como em [25].

Considere o Teorema do Valor Médio:

$$f(x) - f(\bar{x}) = (x - \bar{x})f'(\xi),$$

onde ξ é algum ponto entre x e $\bar{x}$. Se $f(\bar{x}) = 0$, e $f'(\xi) \neq 0$ então temos que:

$$\bar{x} = x - \frac{f(x)}{f'(\xi)}.$$

Seja X um intervalo contendo x e $\bar{x}$, e consequentemente ξ. Temos que $f'(\xi) \in f'(X)$, para $f'(X)$ uma função de inclusão para $f'(x)$ em X. Então:

$$\bar{x} = x - \frac{f(x)}{f'(\xi)} \quad \in \quad x - \frac{f(x)}{f'(X)} = N(x, X),$$

onde $N(x, X)$ é denominado *operador de Newton Intervalar*.

Uma vez que qualquer zero de f em X está em $N(x, X)$, o mesmo também estará em $X \cap N(x, X)$. Desse fato definimos a iteração do método de Newton Intervalar. Seja X^0 um intervalo contendo $\bar{x}$, para $k = 0, 1, 2, ...$, definimos:

$$x^k = m(X^k) \tag{3.6}$$

$$N(x^k, X^k) = x^k - \frac{f(x^k)}{f'(X^k)} \tag{3.7}$$

$$X^{k+1} = X^k \cap N(x^k, X^k). \tag{3.8}$$

Optamos por tomar x^k como o ponto médio de X^k, mas isso não é necessário, poderíamos tomar qualquer outro ponto em X^k.

Mesmo quando $0 \in f'(X^k)$ a iteração (3.7) está bem-definida mediante o uso da aritmética intervalar estendida. É claro que devemos ter em mente o caso patológico quando $0 \in f^I(x)$ e $0 \in f'(X)$, e 0 é um ponto interior de $f^I(x)$ ou de $f'(X)$, então $N(x, X) = (-\infty, \infty)$, um resultado inútil.

3.4.1 Critérios de Parada

Há dois critérios de parada simples que podemos usar:

1. $w(X) < \varepsilon_X$ para um dado $\varepsilon_X > 0$
2. $|f(X)| < \varepsilon_F$ para um dado $\varepsilon_F > 0$

É razoável pedir que ambos os critérios sejam satisfeitos antes de interromper as iterações sobre um intervalo X.

O primeiro critério assegura que a localização de qualquer zero de f está suficientemente limitada, enquanto o segundo assegura que para todo x no intervalo resultante, $|f(x)| < \varepsilon_F$.

Contudo se escolhermos ε_X menor do que $w(\bar{X})$ (onde $\bar{X}$ é a menor caixa contendo todos os zeros de f em X^0) então o algoritmo precisará *particionar* o intervalo atual até que *cada* intervalo restante tenha largura menor que ε_X.

Neste último caso, devemos manter uma lista de intervalos ainda não processados L_p, e aplicar o método a cada $X \in L_p$, até que cumpram os critérios de parada. Outros critérios de parada podem ser vistos em [25].

3.4.2 Propriedades do Método de Newton Intervalar

Listamos agora algumas das propriedades mais interessantes do método de Newton Intervalar. Para uma exposição formal de tais propriedades como teoremas, ver [25] e [49], [50].

1. Todo zero de f no intervalo inicial X^0, será encontrado e corretamente limitado.
2. Se não houver zero de f em X^0, o algoritmo provará automaticamente esse fato, em um número finito de iterações.
3. A existência ou não de um zero de f em X^0 será provada automaticamente pelo algoritmo.
4. Se $0 \notin f'(X^k)$, ao menos metade de X^k será eliminado para a próxima iteração. Isto é, a convergência pode ser rápida mesmo quando $w(X^k)$ for grande.

5. Se $0 \notin f'(X^k)$ para algum $k \geq 0$, então a taxa de convergência a um zero de f em X^k é quadrática.

3.4.3 Newton Intervalar em Várias Variáveis

Vamos agora estender as idéias do Método de Newton Intervalar para encontrar todos os zeros de uma função continuamente diferenciável $f : \mathbb{R}^n \to \mathbb{R}^n$.

Considere a expansão de primeira ordem de Taylor (com resto de Lagrange) em torno de $\bar{x}$:

$$f(x) = f(\bar{x}) + J_f(\xi)(x - \bar{x}),$$

onde ξ é uma combinação convexa de x e $\bar{x}$, e J_f denota a Jacobiana de f. Podemos escrever:

$$f(\bar{x}) = f(x) + J_f(\xi)(\bar{x} - x).$$

Seja agora X uma caixa em $\mathbb{R}^n$ contendo x e $\bar{x}$, e portanto contendo ξ. Seja $J(x, X)$ uma função de inclusão para J_f em X, daí:

$$f(\bar{x}) \in f(x) + J(x, X)(\bar{x} - x),$$

e se $\bar{x}$ é um zero de f em X então:

$$0 \in f(x) + J(x, X)(\bar{x} - x).$$

Logo, nos interessa a solução do sistema linear intervalar:

$$J(x, X)(y - x) = -f(x),$$

que como vimos anteriormente é dada pelo conjunto:

$$S = \{y \in \mathbb{R}^n : f(x) + J(x, x')(y - x) = 0,\ J(x, x') \in J(x, X)\}.$$

Proposição 3.4.1. *O conjunto S contém todo ponto $y \in X$ tal que $f(y) = 0$.*

Prova. Seja $y \in X$ tal que $f(y) = 0$. Então para um dado $x \in X$ temos que

$$0 = f(y) = f(x) + J_f(\xi)(y - x),$$

sendo ξ combinação convexa de x e y. Mas, $J(x, X)$ é um função de inclusão para J_f em X. Logo, para todo $z \in X$, tem-se que $J_f(z) \in J(x, X)$. Assim, como $\xi \in X$ tem-se que $J_f(\xi) \in J(x, X)$. Denotando por $J(x, \xi) = J_f(\xi)$ temos que

$$f(x) + J(x, \xi)(y - x) = 0$$

e portanto $y \in S$. ■

O conjunto S pode ser difícil de descrever, conforme destacado na Seção 3.3. Na verdade queremos uma caixa Y que contenha S (de preferência a menor possível). Daí a sugestiva equação intervalar:

$$J(x, X)(Y - x) = -f(x).$$

Mas, estamos interessados nos zeros de f em X, e como $Y \supset S$, e S contém todos os zeros de f em X, basta buscar em $X \cap Y$.

Para Y, usaremos a notação de $N(x, X)$ e definimos a iteração do método de Newton Intervalar:

$$J(x^k, X^k)[N(x^k, X^k) - x^k] = -f(x^k) \tag{3.9}$$

$$X^{k+1} = X^k \cap N(x^k, X^k), \tag{3.10}$$

onde normalmente tomamos $x^k = m(X^k)$, mas basta que $x^k \in X^k$.

Para resolver (3.9) utilizamos um dos métodos vistos na seção 3.3. Conforme a estratégia adotada para se resolver (3.9) temos variações do Método de Newton Intervalar. Ver [24], [37].

Se empregamos Gauss-Seidel na resolução de (3.9), tão logo as componentes de $N(x^k, X^k)_i$ sejam calculadas, efetuamos (3.10) $X_i^{k+1} = X_i^k \cap N(x^k, X^k)_i$. Trabalhar sempre com as componentes atualizadas nos permite obter melhores limitantes.

Se $J(x, X)$ é regular, então existe uma matriz $V(x, X)$ que contém as inversas das matrizes em $J(x, X)$, e a solução para (3.9) está contida no vetor intervalar

$$N(x, X) = x - V(x, X)f(x).$$

Mas, ao resolver o sistema linear (3.9), podemos nos deparar com $J(x, X)$ contendo alguma matriz singular. Neste caso procedemos como descrito na seção anterior, apenas com uma modificação. Caso a interseção $X \cap N(x, X)$, seja composto por subintervalos, então criamos uma lista de subintervalos, e aplicamos o método de Newton Intervalar a cada intervalo da lista.

3.4.4 Algumas Propriedades

A seguir apresentamos os principais teoremas acerca do Método de Newton Intervalar. Para a demonstração dos mesmos ver [51].

Teorema 3.4.2. *Se existe um zero $\bar{x}$ de f em X, e se $x \in X$, então $\bar{x} \in N(x, X)$.*

Teorema 3.4.3. *Se $X \cap N(x, X) = \emptyset$, então não há zero de f em X.*

Teorema 3.4.4. *Seja x um ponto interior de X. Assuma que $N(x, X)$ foi calculado através de eliminação Gaussiana ou Gauss-Seidel Intervalar. Se $N(x, X)$ é interior a X, então existe um único zero de f em X (e consequentemente em $N(x, X)$).*

Teorema 3.4.5. *Seja f continuamente diferenciável na caixa inicial X^0 contendo um zero $\bar{x}$ de f e a matriz intervalar $M(x^0, X^0)$ sendo regular. Se a sequência de caixas geradas pelo Método de Newton Intervalar converge então:*

$$w(X^{k+1}) \leq \gamma[w(X^k)]^2,$$

para alguma constante $\gamma \geq 0$.

3.5 Desigualdade Intervalar Linear

Em muitos casos nos deparamos com o problema de encontrar os pontos em uma dada caixa X que satisfazem uma determinada inequação intervalar, [25], [26].

Nesta seção trataremos do caso linear, que frequentemente aparece num contexto de otimização global após a linearização de determinadas funções.

Sejam $U = [a, b]$ e $V = [c, d]$ intervalos dados. Estamos interessados em resolver a desigualdade:

$$U + Vt \leq 0, \tag{3.11}$$

isto é, determinar todos os valores de t para os quais exista ao menos um $u \in U$ e um $v \in V$ tais que $u + vt \leq 0$. Logo, queremos encontrar o conjunto:

$$T = \{t \mid \exists\ u \in U,\ \exists\ v \in V,\ u + vt \leq 0\}.$$

Uma maneira simples de resolver a inequação intervalar (3.11) é reescrevê-la como a seguinte equação intervalar:

$$U + Vt = (-\infty, 0].$$

Desse modo, temos que:

$$T = \frac{(-\infty, -a]}{[c, d]},$$

que podemos resolver com o uso da aritmética intervalar estendida.

Com isso obtemos as seguintes regras:

$$T = \begin{cases} [-a/d, \infty) & \text{se } a \leq 0 \text{ e } d < 0 \\ [-a/c, \infty) & \text{se } a > 0,\ c < 0 \text{ e } d \leq 0 \\ (-\infty, \infty) & \text{se } a \leq 0 \text{ e } c \leq 0 \leq d \\ (-\infty, -a/d] \cup [-a/c, \infty) & \text{se } a > 0 \text{ e } c < 0 < d \\ (-\infty, -a/c] & \text{se } a \leq 0 \text{ e } c > 0 \\ (-\infty, -a/d] & \text{se } a > 0,\ c \geq 0 \text{ e } d > 0 \\ \emptyset & \text{se } a > 0 \text{ e } c = d = 0 \end{cases}$$

Na Seção 4.5.3 veremos como este resultado pode contribuir para a resolução do problema de otimização global.

3.6 Box Consistency

Nesta seção discutiremos sobre a técnica *box consistency*, apresentada em [26].

Considere uma equação do tipo $f(x, y) = 0$, com x pertencente a um intervalo X e y pertencente a um intervalo Y. Dizemos que $\hat{x} \in X$ e $\hat{y} \in Y$ são *consistentes* com relação a equação anterior, se fixado $\hat{x}$, existe $y \in Y$ tal que $f(\hat{x}, y) = 0$ e, se fixado $\hat{y}$ existe $x \in X$ tal que $f(x, \hat{y}) = 0$.

Assim, se existem valores de $x \in X$ para os quais não existe $y \in Y$ tal que a equação se cumpra, então tais valores de x podem ser descartados, já que nosso objetivo é encontrar as soluções de $f(x, y) = 0$. Tal conceito pode ser estendido a mais variáveis.

Suponha que procuramos em uma caixa $X \subset \mathbb{R}^n$ por soluções da equação:

$$f(x_1, x_2, \ldots, x_n) = 0.$$

Se considerarmos f apenas como função de x_i e fixarmos as demais variáveis temos:

$$q(x_i) = f(X_1, X_2, \ldots, X_{i-1}, x_i, X_{i+1}, \ldots, X_n) = 0. \tag{3.12}$$

Logo, se f possui derivadas de primeira ordem contínuas, podemos reduzir a caixa X em relação à i-ésima coordenada aplicando o método de Newton intervalar à equação (3.12):

$$N(\hat{x}_i, X_i) = \hat{x}_i - \frac{q(\hat{x}_i)}{q'(X_i)}, \tag{3.13}$$

e então atualizamos X_i, tomando $X_i' = X_i \cap N(\hat{x}_i, X_i)$. Note que, se a interseção for vazia, então demonstramos que não existe zero de $q(x_i)$ em X_i e portanto não há zeros de $f(x_1, x_2, \ldots, x_n)$ em X.

Normalmente o ponto de expansão $\hat{x}_i$ é tomado como $m(X_i)$. Todavia seguindo o esquema apresentado em [26], tomaremos $\hat{x}_i = a$, onde $X_i = [a, b]$. Com isso procuramos aumentar o limitante inferior em X_i. Note que se $0 \notin q(a)$ então $a \notin N(a, X_i)$, e consequentemente haverá redução em X_i.

O mesmo esquema pode ser aplicado em b, buscando reduzir o limitante superior de X_i.

Além disso, claramente podemos aplicar este esquema a mais de uma coordenada na caixa X, e à medida que as componentes X_i são atualizadas, estas são utilizadas na redução das próximas coordenadas.

Agora, se estamos interessados em encontrar as soluções de $F(x) = 0$, para x pertencente a uma caixa $X \subset \mathbb{R}^n$ e $F : \mathbb{R}^n \to \mathbb{R}^n$, podemos aplicar o mesmo processo a cada componente $F_i : \mathbb{R}^n \to \mathbb{R}$ por vez.

Este processo é denominado *box consistency*, e segundo [26] apresenta melhores resultados que o método de Newton intervalar quando aplicado a caixas *grandes*. O uso desta técnica para otimização global é apresentada no Capítulo 4.

Em [26], Hansen e Walster, introduzem também o conceito de *hull consistency*, o qual comentaremos brevemente com um exemplo unidimensional.

Suponha que queremos obter as soluções de $f(x) = 0$, para $x \in X$. A idéia de *hull consistency* é muito semelhante a de *box consistency*, exceto pelo fato

que *hull consistency* não faz uso das derivadas de f. No entanto, é exigida uma estrutura especial de f:

$$f(x) = h(x) - g(x),$$

e que a inversa de h seja facilmente obtida.

Então para que $f(x) = 0$ devemos ter $h(x) - g(x) = 0$, ou ainda:

$$\begin{aligned} h(x) &= g(x) \\ x &= h^{-1}[g(x)]. \end{aligned}$$

Logo, se existe zero de f em X então pela monotonicidade inclusiva o mesmo deverá estar em:

$$X' = h^{-1}[g(X)].$$

Este procedimento pode ser aplicado a funções de várias variáveis assim como feito com *box consistency*, escolhendo uma variável x_i como dependente e fixando as demais.

3.7 Envelopes Convexos

Em um algoritmo do tipo branch-and-bound para otimização global, uma etapa fundamental é obter limitantes inferiores para a função objetivo f em uma dada caixa $X \in \mathbb{R}^n$. Até aqui, vimos duas formas de obter tais limitantes.

Primeiro, se temos conhecimento da constante de Lipschitz L (ou uma estimativa superior para a mesma), vimos na Seção 2.2 que um limitante inferior para f em $X = [a, b]$ é dado por

$$f(x) \geq \frac{f(a) + f(b)}{2} - \frac{L}{2}(b - a) = f_{lb}.$$

Vimos também que a Análise Intervalar, pode ser usada para obter limitantes, seja pela *extensão intervalar* de f em X, ou pelo uso de alguma *forma centrada*. Porém essas não são as únicas formas de se obter limitantes inferiores para a função objetivo. Na verdade, o problema de otimização global poderia ser facilmente resolvido, se pudéssemos obter o *envelope convexo* de f em X, isto é, a maior função convexa que subestima f em X.

Definição 3.7.1 (Envelope Convexo)**.** *Seja* $f : [a, b] \to \mathbb{R}$*. Dizemos que* $cof(x)$ *é o envelope convexo de* f *em* X *se:*

$$cof(x) = \sup \{g(x) \mid g : X \to \mathbb{R} \ \ convexa\ ,\ g(x) \leq f(x),\ \forall x \in X\}\,.$$

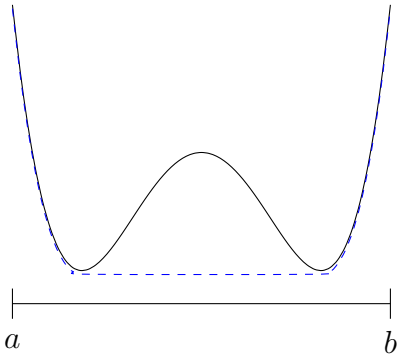

Figura 3.1: Envelope convexo

A Figura 3.1 ilustra o envelope convexo de uma função não convexa qualquer.

Note que o envelope convexo de um função f, pode ser visto como a parte inferior do *envoltório convexo* (conjunto de todas as combinações convexas) do *epígrafo* de f.

Além disso o envelope convexo também pode ser descrito como:

$$cof(x) = \sup \left\{s^t x - b \mid s^t y - b \leq f(y),\ \forall y \in X\right\},$$

ou seja, é o supremo das funções lineares que minoram f em X.

Das definições anteriores temos o seguinte resultado.

Teorema 3.7.2. *$\bar{x}$ é minimizador global de f em X se e somente se $\bar{x}$ é minimizador local de cof e $cof(\bar{x}) = f(\bar{x})$.*

Logo se conhecemos o envelope convexo de f, então para encontrar o mínimo global de f basta minimizar o cof, que por sua vez é uma função convexa.

O problema está exatamente em como obter cof. Conforme a definição acima, o cof coincide exatamente com o biconjugado de Legendre-Fenchel $cof = f^{**}$, onde $f^* : X^* \to \mathbb{R}$

$$f^*(s) = \sup_{x \in X}[s^t x - f(x)].$$

Existem algoritmos para calcular f^{**}, como em [43]. Porém tais procedimentos são viáveis apenas para uma variável.

3.7.1 Subestimativas Convexas

Para determinados tipos de função é fácil de se obter uma subestimativa convexa para a mesma. Em certos casos tal subestimativa pode coincidir com o envelope convexo da função.

Por exemplo, seja f uma função côncava em um intervalo $[a, b]$. Daí uma subestimativa convexa é facilmente obtida por

$$f(a) - \frac{f(a) - f(b)}{b - a}(x - a),$$

que corresponde a reta que une os extremos do intervalo, ver Figura 3.2.

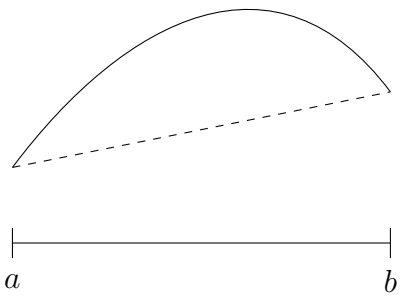

Figura 3.2: Envelope convexo para uma função côncava

No caso de termos bilineares, na forma xy, Al-Khayyal e Falk, [4], mostraram que a subestimativa convexa mais precisa no domínio $[x^L, x^U] \times [y^L, y^U]$ é obtida

substituindo xy por uma variável w_b, satisfazendo:

$$w_B = \max\left\{x^L y + xy^L - x^L y^L \ , \ x^U y + xy^U - x^U y^U\right\}.$$

Podemos introduzir a relação acima em nosso problema através das desigualdades:

$$\begin{aligned} w_B &\geq x^L y + xy^L - x^L y^L \\ w_B &\geq x^U y + xy^U - x^U y^U. \end{aligned}$$

E mais, segundo McCormick, [47], impondo um limitante superior para w_B podemos obter uma melhor aproximação para o problema inicial. Isso é feito incluindo as desigualdades:

$$\begin{aligned} w_B &\leq x^U y + xy^L - x^U y^L \\ w_B &\leq x^L y + xy^U - x^L y^U. \end{aligned}$$

Uma subestimativa para termos fracionários x/y pode ser obtida de maneira análoga, segundo Maranas e Floudas (1995), [45]. Uma nova variável w_F é introduzida e limitada pelas seguintes desigualdades

$$\begin{aligned} w_F &\geq \begin{cases} x^L/y + x/y^U - x^L/y^U, & \text{se } x^L \geq 0 \\ x/y^U - x^L y/y^L y^U + x^L/y^L, & \text{se } x^L \leq 0 \end{cases} \\ w_F &\geq \begin{cases} x^U/y + x/y^L - x^U/y^L, & \text{se } x^U \geq 0 \\ x/y^L - x^U y/y^L y^U + x^U/y^U, & \text{se } x^U \leq 0 \end{cases} \end{aligned}$$

Além dessas regras, outras para termos trilineares xyz e trilineares fracionários xy/z são apresentadas em [1].

Uma função $f : [a, b] \to \mathbb{R}$ é dita côncavo-convexa se para algum $x_m \in (a, b)$ a função é côncava em $[a, x_m]$ e convexa em $[x_m, b]$. Assim, pela definição de envelope convexo, espera-se que o mesmo coincida com f em sua parte convexa e se altere para uma reta em sua parte côncava, ver Figura 3.1. Desse modo, o que nos interessa é determinar o ponto $\bar{x}$ em que o envelope convexo muda de representação. O ponto $\bar{x}$ é aquele onde a inclinação da reta secante aos pontos a e $\bar{x}$ coincide com a inclinação da curva da função, que equivale a solução de

$$f(x) - f(a) - f'(x)(x - a) = 0,$$

que pode ser obtida pelo Método de Newton ou Secante.

Para funções convexo-côncavas, o envelope convexo pode ser determinado de maneira similar.

Note que todas as regras apresentadas nessa seção, são aplicáveis a funções de uma variável, ou a termos bilineares/trilineares. Porém tais técnicas são empregadas na prática para se determinar uma subestimativa convexa para funções de várias variáveis que podem ser escritas em função dos termos tratados anteriormente. Para isso essas regras são aplicadas aos termos que compõe a função original.

3.7.2 Envelopes α

Para funções não convexas mais gerais, sem nenhuma das particularidades citadas na seção anterior, uma subestimativa convexa pode ser obtida por meio de um *envelope* α, como descrito em [1].

Seja $f : X \to \mathbb{R}$, com X sendo uma caixa em $\mathbb{R}^n$, $X = [x_1^L, x_1^U] \times \cdots \times [x_n^L, x_n^U]$ e $f \in \mathbb{C}^2$. Uma subestimativa convexa é dada pela função abaixo

$$\phi(x) = f(x) + \frac{1}{2}\sum_{i=1}^{n} \alpha_i (x_i^L - x_i)(x_i^U - x_i),$$

onde α_i são escalares positivos.

Note que o segundo termo de $\phi(x)$ é sempre negativo para $x \in X$, assim $\phi(x) \leq f(x)$ para todo $x \in X$. Além disso este termo é quadrático e convexo, logo todas as não convexidades na função original podem ser eliminadas tomando-se α_i's suficientemente grandes. Portanto $\phi(x)$ é de fato uma subestimativa convexa válida para f em X.

Como ϕ também é continuamente diferenciável duas vezes, temos que ϕ será convexa se e somente se $\nabla^2\phi(x)$ for semi-definida positiva. Note que:

$$\nabla^2\phi(x) = \nabla^2 f(x) + \Delta,$$

onde Δ é uma matriz diagonal cujos elementos são os α_i's.

Dado que a $f \in \mathbb{C}^2$ e o conjunto X é compacto, a matriz $\nabla^2 f(x)$ é limitada e portanto, existe uma matriz δ tal que a matriz $\nabla^2\phi(x)$ é definida positiva.

Uma forma de determinar os valores α_i's de modo que $\nabla^2\phi(x)$ seja semi-definida positiva para todo $x \in X$, é usar uma versão intervalar do Teorema dos Discos de Gerschgorin.

Teorema 3.7.3. *Seja $A^I \in \mathbb{I}^{n\times n}$ uma matriz intervalar, $A^I_{ij} = [a^L_{ij}, a^U_{ij}]$. Um limitante inferior para o menor autovalor é dado por*

$$\lambda_{min} \geq \min_i \left\{ a^L_{ii} - \sum_{i\neq j} mag(a_{ij}) \right\},$$

onde $mag(a_{ij}) = \max\left\{\left|a^L_{ij}\right|, \left|a^U_{ij}\right|\right\}$, e λ_{min} é o menor dentre os menores autovalores das matrizes $A \in A^I$.

Outras formas de se obter os coeficientes α_i são tratadas em [1].

É claro que quanto menores forem os valores α_i's, mais preciso será o limitante inferior dado por $\phi(x)$. Todavia após obter os α_i's, conhecemos uma subestimativa convexa ϕ para f em X, mas não sabemos qual o mínimo de ϕ em X. Logo para obter o limitante inferior para f em X é necessário resolver o subproblema

$$\min_{x\in X} \phi(x),$$

o qual esperamos resolver de forma eficiente já que ϕ é convexa.

Capítulo 4

Análise Intervalar aplicada à Otimização Global Irrestrita

Nesta capítulo veremos como a análise intervalar pode contribuir na resolução do problema de otimização global (1.1).

O minimizador global poderia ocorrer na fronteira de X^0. Assumiremos que tal minimizador ocorre no interior de X^0.

Poderíamos usar o método de Newton Intervalar, descrito no Capítulo 3, para tentar limitar por pequenas caixas todos os pontos em que $\nabla f(x) = 0$ para $x \in X^0$. Todavia, f pode ter um número arbitrariamente alto de pontos em X^0 onde o gradiente se anula, sendo que estes por sua vez podem ser maximizadores ou pontos de sela, e não minimizadores.

As ferramentas que a análise intervalar nos fornece são mais úteis num contexto de algoritmos do tipo *branch-and-bound (BB)* como veremos a seguir.

4.1 Princípio Branch-and-Bound

Em um método branch-and-bound (BB), inicialmente a região viável X^0 é *relaxada*, isto é, obtemos um conjunto $S_0 \subset \mathbb{R}^n$ (mais simples que X^0) tal que $X^0 \subset S_0$. Em seguida S_0 é particionado (*branching*) em um número finito de subconjuntos S_i, dando origem a subproblemas:

$$P_i\colon \ \min_{x \in S_i} f(x). \tag{4.1}$$

É claro que a *melhor solução dentre as soluções globais dos subproblemas* P_i é também solução global de (1.1). Portanto o próximo passo é resolver os subproblemas.

Para cada subproblema P_i são determinados limitantes inferiores (*lower bounds*) e (se possível) superiores (*upper bounds*) para a solução, que denotaremos por $\beta(S_i)$ e $\alpha(S_i)$ respectivamente (*bounding*).

Além disso definimos

$$\beta = \min_{i \in I} \beta(S_i)$$

e

$$\alpha = \min_{i \in I} \alpha(S_i),$$

onde I é o conjunto de índices dos subproblemas P_i que ainda não foram *processados* (isto é, *resolvidos* ou *descartados*, como veremos adiante).

Em seguida *selecionamos* alguns (ou todos) subproblemas P_i, e cada um destes é novamente particionado em subproblemas ainda menores, e assim sucessivamente.

Note que

$$\beta \leq \min_{x \in X^0} f(x) \leq \alpha,$$

portanto um critério prático de parada é $\alpha - \beta < \varepsilon$.

Um fato importante é que ao longo do processo BB, alguns subproblemas podem ser *descartados*, isto é, não precisam ser explorados (particionados), simplesmente porque o mínimo global não pode ser alcançado em tais subproblemas. A esse processo damos o nome de *pruning* (poda).

Essa denominação decorre do fato de que o processo de BB pode ser visto como uma árvore onde S_0 é o nó raiz e cada particionamento (ramificação) dá origem a novos nós (subproblemas). Assim se temos certeza de que em um dado nó o mínimo global não pode ocorrer, então podamos tal nó (descartamos tal subproblema).

As regras mais comuns de pruning são:

- *Prune by bound.* Se para um dado subproblema P_i, seu limitante inferior $\beta(S_i)$ é maior que α (o menor limitante superior no momento) então P_i pode ser descartado. A lógica dessa regra é fácil de ser entendida, pois se:

$$\beta(S_i) > \alpha = \min_{i \in I} \alpha(S_i) = \alpha(S_k),$$

  então o melhor valor possível que pode ser alcançado em P_i não é melhor do que o pior valor que pode ser alcançado em P_k.

- *Prune by Infeasibility.* Se $S_i \cap X^0 = \emptyset$ então a região viável do subproblema P_i não é viável para o problema P, logo P_i pode ser descartado.

- *Prune by Optimality.* Se o mínimo global de um dado subproblema P_i foi encontrado não há porque explorar mais tal subproblema. Se tal solução for menor que α, então ela se torna uma das candidatas a mínimo global.

4.2 Um Protótipo de Algoritmo BB

Um protótipo de algoritmo branch-and-bound segundo Horst & Tuy, [31], é descrito a seguir:

Algoritmo 2: branch-and-bound

1. obtenha uma relaxação S_0 para a região factível X^0. Obtenha um limitante superior $\alpha(S_0)$ e inferior $\beta(S_0)$ para f em S_0. Encontre x^0 factível em S_0 e faça $\bar{x} = x^0$. Se $f(x^0) < \alpha$, então $\alpha = f(x^0)$. Inicialize a lista $L_p = \{P_0\}$.

2. se $\alpha - \beta < \varepsilon$ ou $L_p = \emptyset$ pare, retorne $\bar{x}$ como solução. Senão, escolha alguns subproblemas que ainda estão na lista L_p (*selection*). Remova tais subproblemas de L_p.

3. para cada subproblema P_k escolhido, particione-o (*branching*) em m subproblemas menores, tais que $\cup_{i=1}^{m} S_{k_i} = S_k$, e adicione-os a lista L_p.

4. para cada subproblema *filho* P_{k_i}, se $S_{k_i} \cap X^0 = \emptyset$, então remova-o da lista L_p (*prune by infeasibility*).

5. caso contrário, obtenha limitantes $\alpha(S_{k_i})$ e $\beta(S_{k_i})$ (*bounding*).

6. se $\beta(S_{k_i}) > \alpha$ então remova P_{k_i} de L_p (*prune by bound*).

7. caso $\alpha(S_{k_i}) = \beta(S_{k_i})$, então remova-o da lista (*prune by optimality*).

8. faça $\alpha = \min_{i \in L_p} \alpha(S_i)$, $\beta = \min_{i \in L_p} \beta(S_i)$, e $\bar{x}$ igual a algum ponto factível, dentro de alguma região S_j tal que $\alpha(S_j) = \min_{i \in L_p} \alpha(S_i)$.
   Retorne ao passo 2.

O protótipo descrito anteriormente é uma das versões mais simples de algoritmo BB para otimização global, e é a base para algoritmos que empregam técnicas mais avançadas como envelopes convexos [1] ou análise intervalar [25], [35], para eliminar regiões onde o minimizador global não pode estar.

Como nosso propósito é obter um algoritmo para a resolução do problema de otimização global irrestrito, não nos preocuparemos como no protótipo acima, em verificar a viabilidade.

Além disso, iremos supor que uma caixa inicial $X^0 \subset \mathbb{R}^n$ que contém os minimizadores globais de f é conhecida previamente. De fato, se f é coerciva, isto é, se $f(x) \to \infty$ à medida em que $\|x\| \to \infty$, então existe uma caixa $X^0 \subset \mathbb{R}^n$, ainda que muito grande, tal que todos os minimizadores globais de f se encontram em X^0.

4.3 Critérios de Seleção e Partição

As operações de seleção, partição e obtenção de limitantes são cruciais para eficiência e convergência dos métodos branch-and-bound. Não trataremos de todos os detalhes aqui. Para mais informações, ver [31], [52], [62].

Um esquema de obtenção de limitantes é dito *consistente* se todo elemento de partição S_k não descartado pode ser refinado posteriormente e, se para qualquer sequência infinita decrescente de elementos de partição, a diferença entre os limitantes superiores e inferiores tende a zero. A maioria dos algoritmos assegura a consistência através de um esquema exaustivo de partição, isto é, os elementos de partição convergem a pontos ou a conjuntos onde o subproblema é facilmente resolvido.

Um critério de seleção é dito *bound-improving* se, sempre após um número finito de iterações é selecionado o elemento de partição associado ao menor limitante inferior atual.

Se o esquema de obtenção de limitantes é consistente e o critério de seleção é bound-improving então o processo branch-and-bound é convergente, ver [31].

No passo 2 do Algoritmo 2, selecionaremos sempre o subproblema que possui *menor limitante inferior* $\beta(S_i)$. Isso assegura que o critério de seleção é bound-improving. A lista L_p será mantida em ordem crescente de limitante inferior, e desse modo, sempre escolheremos o primeiro elemento da lista.

No passo 3 do Algoritmo 2, o problema selecionado será subdivido em dois novos subproblemas. Para isso realizaremos a *bisseção* da caixa S_i ao longo de alguma coordenada, gerando duas novas caixas menores.

Alguns autores [30], [31], selecionam a coordenada na qual se dará a bisseção como sendo aquela de maior tamanho, isto é, se $X = S_i$, então a bisseção se dará na coordenada j tal que:

$$j = \text{argmax}_k \left\{ w(X_k) \right\}.$$

Em nosso algoritmo tomaremos como coordenada para bisseção:

$$j = \text{argmax}_k \left\{ w(\nabla f(X)_k.(X_k - x_k)) \right\}, \tag{4.2}$$

onde $x_k = m(X_k)$.

A idéia por trás dessa regra é a de minimizar a largura da função de inclusão:

$$\begin{aligned} w(F(X)) &= w(F(X) - f(x)) \\ &\approx w(\nabla f(X)^T (X - x)) \\ &= w(\sum_{k=1}^{n} \nabla f(X)_k.(X_k - x_k)) \\ &= \sum_{k=1}^{n} w(\nabla f(X)_k.(X_k - x_k)). \end{aligned}$$

Para mais detalhes sobre esta e outras regras, ver [62].

Há outros esquemas além da bisseção para a partição dos intervalos. Alguns autores sugerem a trisseção ou a multiseção da caixa X. O leitor interessado pode consultar: [29], [30], [31], [32], [52] e [62].

Para obtenção de limitantes usaremos extensões intervalares e formas centradas. Como a largura excessiva das extensões intervalares e formas centradas tende a zero à medida que a amplitude do intervalo é reduzido e, como a cada iteração uma caixa será descartada, reduzida ou particionada, temos que o esquema de obtenção de limitantes é consistente.

Como o critério de seleção adotado é bound-improving e, o esquema de obtenção de limitantes é consistente, temos então que o algoritmo branch-and-bound com as escolhas anteriores é convergente.

4.4 Obtendo Limitantes

Podemos usar a análise intervalar para obter limitantes inferiores e superiores de f para os subproblemas gerados, pois suas regiões factíveis serão também caixas. Para tanto, podemos tomar simplesmente a *extensão intervalar natural* de f, ou usar as *formas centradas* de *Taylor* ou do *Valor Médio*, ou ainda a *forma inclinação*.

Lembrando, quando utilizamos a notação $f(X)$ estamos nos referindo a extensão intervalar natural de f em X, e como os limitantes inferiores e superiores de f em X serão denotados por $\underline{f(X)}$ e $\overline{f(X)}$ respectivamente, temos que $f(X) = [\underline{f(X)}, \overline{f(X)}]$.

Há outras formas para obter limitantes para um dado subproblema, ou em um dado subintervalo $X \subset X^0$. Por exemplo, se f é Lipschitz em X e temos conhecimento de sua constante L, então podemos obter um limitante inferior de maneira barata, como vimos na Seção 2.2.

Temos também alternativas mais caras computacionalmente, como calcular uma *subestimativa convexa* da função no subintervalo [1], ou ainda o *envelope convexo* [68], como descrito na Seção 3.7, e em seguida obter um limitante inferior encontrando o mínimo de tal *relaxação*.

Em nosso algoritmo optamos por utilizar a extensão intervalar natural da função f em um dado intervalo, pela simplicidade na implementação. Os limitantes obtidos dessa forma podem ser melhorados após o *teste de monotonicidade*.

4.5 Análise Intervalar e Branch-and-Bound

Podemos tirar proveito das ferramentas de análise intervalar principalmente para incrementar as regras de *descarte* durante o processo de BB.

4.5.1 Teste de Monotonicidade

Seja $X \subset X^0$ uma caixa no interior de X^0. Note que se não existe $x \in X$ tal que $\nabla f(x) = 0$, então X não pode conter um minimizador global. Para realizar este teste basta considerar uma extensão intervalar $\nabla f(X)$ do gradiente da f em X. Realizamos o teste componente a componente, e se $0 \notin \frac{\partial f(x)}{\partial x_i}(X)$ para algum i, então $\nabla f(x)$ não possui nenhum zero em X, logo X pode ser descartada. Esse teste é chamado *teste de monotonicidade*.

Se a caixa X não foi eliminada pelo teste de monotonicidade, aproveitamos a avaliação de $\nabla f(X)$ para dois propósitos.

Primeiro, avaliamos a forma do Valor Médio:

$$f(x) + \nabla f(X)^T (X - x),$$

onde $x = m(X)$, e realizamos a interseção dos limitantes obtidos aqui, com aqueles obtidos anteriormente pela extensão intervalar natural. Isso pode melhorar os limitantes de f em X.

Além disso, a informação de $\nabla f(X)$ pode ser utilizada posteriormente para determinar em qual coordenada realizar a bisseção, de acordo com a regra estabelecida em (4.2).

4.5.2 Teste de Não Convexidade

Se $\bar{x}$ é um minimizador irrestrito de f, seja local ou global, temos que $\nabla^2 f(x) \geq 0$ para todo $x \in V(\bar{x})$. Por outro lado uma condição necessária para que $\nabla^2 f(x)$ seja semi-definida positiva é que os elementos da diagonal $\nabla^2 f(x)_{ii}$ sejam não negativos para todo i. Consideremos novamente uma caixa $X \subset X^0$, e $H(X)$ uma extensão intervalar para a Hessiana da f em X. Se $H_{ii}(X) < 0$ para algum i, então $\nabla^2 f(x)_{ii} < 0$ para todo $x \in X$, e portanto $\nabla^2 f(x)$ não pode ser semi-definida positiva em nenhum ponto de X. Logo podemos descartar X. Este é chamado *teste de não convexidade*.

4.5.3 Desigualdade Intervalar

Tanto o teste de monotonicidade, como o de não convexidade são capazes de eliminar, ou não, *toda* a caixa X em questão. A partir daqui apresentaremos técnicas capazes de eliminar *partes* da caixa X.

Assim que uma nova caixa X é gerada pelo algoritmo BB, limitantes inferiores e superiores para f em X são avaliados pela extensão intervalar natural. Neste momento verificamos se $\underline{f(X)} > \bar{f}$, onde $\bar{f}$ é o menor valor de função encontrado até o momento. Se sim, toda a caixa X pode ser eliminada.

Agora, considere a forma centrada do Valor Médio para f em uma dada caixa X de centro em x:

$$\begin{aligned}
\square f(X) &\subset f(x) + \nabla f(X)^T (X - x) \\
&= f(x) + \sum_{j=1}^{n} \nabla f(X)_j . (X_j - x_j) \\
&= f(x) + \sum_{j \neq i} \nabla f(X)_j . (X_j - x_j) + \nabla f(X)_i (X_i - x_i).
\end{aligned}$$

Nos interessam na caixa X apenas os pontos x nos quais $f(x) \leq \bar{f}$, ou seja:

$$f(x) + \sum_{j \neq i} \nabla f(X)_j . (X_j - x_j) + \nabla f(X)_i (X_i - x_i) \leq \bar{f}.$$

Assim, tomando:

$$\begin{aligned}
U &= f(x) + \sum_{j \neq i} \nabla f(X)_j . (X_j - x_j) - \bar{f}, \\
V &= \nabla f(X)_i, \text{ e} \\
t &= X_i - x_i,
\end{aligned}$$

obtemos uma desigualdade intervalar linear $U + Vt \leq 0$.

Note que a desigualdade intervalar pode ser aplicada às n coordenadas da caixa X.

A Seção 3.5 mostra como calcular o conjunto solução T para este tipo de desigualdade. Note que nem todos os casos precisam ser considerados no cálculo

de T. Isto porque o teste de desigualdade intervalar é efetuado após o teste de monotonicidade. Logo, se $0 \notin \nabla f(X)_i = V$, a caixa X já terá sido eliminada pelo teste de monotonicidade.

Após obter T realizamos a seguinte atualização:

$$\begin{aligned} X_i^{'} &= T + x_i \\ X_i &= X_i^{'} \cap X_i \end{aligned}$$

Se a interseção acima resultar em um conjunto vazio, mostramos que não há ponto $x \in X$ tal que $f(x) \leq \bar{f}$, logo toda a caixa X pode ser descartada. Caso T seja composto por dois intervalos semi-infinitos, a interseção acima pode resultar em um *gap* na caixa X. Se isso ocorrer, armazenamos a informação do *gap* para ser usada em uma futura bisseção.

Poderíamos realizar a bisseção em mais de uma coordenada a medida em que os gaps vão aparecendo. Todavia eliminar os gaps em cada coordenada pode gerar um número excessivo de novas caixas, a medida que a dimensão aumenta.

Optamos por armazenar as informações de *gap* em cada coordenada, para decidir posteriormente em qual coordenada será feita a bisseção.

4.5.4 Box Consistency

A linearização utilizada na seção anterior é feita em relação a todas as variáveis. Podemos utilizar também o conceito de *box consistency* apresentado na Seção 3.6 para verificar se $f(x) \leq \bar{f}$ em uma determinada caixa X. Ao utilizar este conceito, linearizamos f com respeito a uma variável por vez, fixando as demais em seus intervalos:

$$q(x_i) = f(X_1, X_2, \ldots, X_{i-1}, x_i, X_{i+1}, \ldots, X_n) \leq \bar{f}.$$

Podemos reescrever a desigualdade acima como:

$$f(X_1, X_2, \ldots, X_{i-1}, x_i, X_{i+1}, \ldots, X_n) - \bar{f} = (-\infty, 0],$$

ou ainda:

$$f(X_1, X_2, \ldots, X_{i-1}, x_i, X_{i+1}, \ldots, X_n) + [-\bar{f}, \infty) = 0. \tag{4.3}$$

Assim podemos aplicar a esta equação o procedimento descrito na Seção 3.6.

Para cada coordenada serão necessárias duas novas avaliações de f, uma em cada extremo do intervalo X_i. Precisaremos também de $\frac{\partial f}{\partial x_i}(X)$. Mas essa informação já temos armazenada pois o teste de monotonicidade já foi realizado.

Além de usar *box consistency* para verificar os pontos $x \in X$ que verificam $f(x) \leq \bar{f}$, podemos ainda aplicá-la sobre a equação $\nabla f(x) = 0$, para eliminar regiões de X que não contêm pontos estacionários de f.

Este também é o propósito do método de Newton Intervalar como veremos a seguir. Todavia, segundo [26], *box consistency* funciona melhor em caixas *grandes*, enquanto o Newton intervalar funciona melhor em caixas *pequenas*.

Perceba que $\nabla f(x) = 0$ é um sistema de equações não lineares. Assim, aplicamos *box consistency* a cada uma das equações $\nabla f(x)_j = 0$, com respeito a uma, algumas, ou todas as variáveis x_i.

Optamos por aplicar *box consistency* com respeito à variável x_i sobre a equação $\nabla f(x)_i = 0$, para $i = 1, 2, \ldots, n$. Outros esquemas são discutidos em [26].

Note que para cada coordenada i, precisaremos de duas avaliações de gradiente, em relação aos extremos de X_i, e também de um elemento da diagonal da Hessiana $\nabla^2 f(X)_{ii}$. Se o teste de não convexidade foi realizado previamente, podemos aproveitar o cálculo de $\nabla^2 f(X)_{ii}$ já realizado. Caso contrário à medida que as componentes $\nabla^2 f(X)_{ii}$ forem calculadas, podemos verificar se $\nabla^2 f(X)_{ii} < 0$.

4.5.5 Newton Intervalar

Poderíamos aplicar o método de Newton Intervalar para localizar todos os pontos em que $\nabla f(x) = 0$ em X^0, mas como já mencionado anteriormente, tal procedimento é inviável.

Na verdade o método de Newton Intervalar pode fornecer bons resultados quando aplicado dentro de um algoritmo do tipo BB. Por exemplo, se um dado subintervalo $X \subset X^0$, não foi descartado pelo teste de *limitante inferior*, passou pelo teste de *monotonicidade* e de *não-convexidade*, então podemos aplicar *uma iteração* do Método de Newton Intervalar sobre ∇f em X.

Ao realizar uma iteração de Newton Intervalar, se $N(x, X)$ estiver no interior de X o Teorema 3.4.4 garante a *existência* e *unicidade* de um zero de $\nabla f(x)$ em X e consequentemente em $N(x, X)$. Como o intervalo X não foi barrado pelo teste de não convexidade e tem um único ponto estacionário, temos boas chances de tal ponto ser o minimizador global de f em X, e então podemos nos sentir encorajados a iniciar um algoritmo de busca local em X. Se tal algoritmo local convergiu a um ponto $\hat{x} \in X$ que cumpre as *condições suficientes de segunda ordem* então podemos descartar X *por otimalidade*.

Por outro lado o Teorema 3.4.3 nos diz que se $N(x, X) \cap X = \emptyset$ então não existe ponto $x \in X$ no qual o gradiente se anula. Portanto X não pode conter um minimizador global de f, e podemos descartar X.

No caso em que a iteração do método de Newton Intervalar não conseguiu provar a unicidade ou a não existência de zeros de ∇f em X, adicionamos o novo intervalo $N(x, X) \cap X$ a lista de intervalos a serem processados no lugar de X.

Algoritmo 3: Iteração de Newton Intervalar

1. Obtenha X, $J(x, X)$ e $\nabla f(x)$.
2. Calcule $m(J(x, X))$ e se possível $B = m(J(x, X))^{-1}$. Senão, tome $B = I$.
3. Faça $M = BJ(x, X)$, $r = -B\nabla f(x)$ e $y = N(x, X) - x$.
4. Se M for diagonalmente dominante, aplique eliminação Gaussiana sobre $My = r$.
5. Caso M não seja diagonalmente dominante ou a eliminação Gaussiana falhe, aplique o método de Gauss-Seidel sobre $My = r$, enquanto houver redução na caixa X.
6. Se alguma componente X_i for vazia, significa que X não possui zeros de $\nabla f(x)$, logo descarte X.
   Se alguma componente X_i for composta por dois intervalos, guarde a informação do *gap*.
   Se constatada a unicidade de um zero de $\nabla f(x)$ em X, guarde esta informação.

No Algoritmo 3, ao resolver o sistema linear $J(x, X)[N(x, X) - x] = -\nabla f(x)$, primeiro precondicionamos o sistema. A seguir, se a matriz intervalar do sistema

precondicionado $M = BJ(x, X)$ for intervalarmente diagonalmente dominante, então aplicamos eliminação Gaussiana antes do método de Gauss-Seidel intervalar. Apesar de mais cara computacionalmente, os limitantes fornecidos pela eliminação Gaussiana Intervalar costumam ser mais precisos, permitindo uma redução mais efetiva na caixa X.

4.5.6 Back-Boxing

Em 1996, Iwaarden [32] propôs uma estratégia para a resolução de problemas de otimização global irrestrita. Denominada *Back-Boxing*, a idéia consiste em tirar proveito da eficiência dos métodos locais de otimização, e após a localização de um minimizador local $\bar{x}$, determinar a maior região possível ao redor de $\bar{x}$, tal que $\bar{x}$ seja o único minimizador nessa região. Alguns autores, como [35], se referem a este procedimento como *ε-inflation*.

Para tanto, novamente é empregada a análise intervalar.

Uma vez determinado um minimizador local $\bar{x} \in X$, buscamos a maior caixa B_g ao redor de $\bar{x}$ para qual $0 \notin \nabla f(B_g)$. Para isso é usada uma extensão intervalar natural de ∇f. Enquanto $0 \notin \nabla f(B_g)$ aumentamos B_g.

Em seguida, buscamos a maior caixa B_h a partir de B_g, na qual $\nabla^2 f(x) > 0$ para todo $x \in B_h$. Para verificar se $\nabla^2 f(x) > 0$ para todo $x \in B_h$, verificamos se $\nabla^2 f(B_h)$ é diagonalmente dominante com elementos positivos na diagonal. Enquanto $\nabla^2 f(B_h)$ for diagonalmente dominante aumentamos B_h.

Por fim, buscamos a maior caixa B_{hs} a partir de B_h, na qual $\bar{x}$ seja o único ponto que anula o $\nabla f(x)$. Para tanto usamos o operador de Newton Intervalar $N(\bar{x}, B_{hs})$, e aumentamos B_{hs} enquanto $N(\bar{x}, B_{hs}) \subset B_{hs}$.

Resumindo, após localizar um minimizador local $\bar{x} \in X$ (através de um método de busca local restrito a X), obtemos a caixa B_g, que em seguida é aumentada para B_h, e por fim aumentada para B_{hs}. A essa caixa final chamaremos de B. Sabemos daí que $\bar{x}$ é o único minimizador global de f em B, e consequentemente de $B \cap X$, portanto tal região não precisa mais ser explorada em X. A seguir, particionamos o espaço restante $X \setminus (B \cap X)$ de maneira conveniente e continuamos com o processo de branch-and-bound.

Algoritmo 4: Crescer a caixa

1. Dados $\bar{x} \in X,\ \alpha \in (0,1)$, $0 < a_0 < \alpha X$ e $\theta > 1$. Faça $a = a_0$ e $B = \bar{x}$;
2. Faça $B_g = (\bar{x} + [-a, a]e) \cap X$
3. Se $0 \notin \nabla f(B_g)$, faça $B = B_g$ e,
   se $w(B) > \alpha X$, pare, senão faça $a = \theta a$ e volte ao passo 2.
4. Faça $B_h = (\bar{x} + [-a, a]e) \cap X$
5. Se $\nabla^2 f(B_h)$ for diagonalmente dominante, faça $B = B_h$ e,
   se $w(B) > \alpha X$, pare; senão faça $a = \theta a$ e volte ao passo 4.
6. Faça $B_{hs} = (\bar{x} + [-a, a]e) \cap X$
7. Aplique uma iteração do método de Newton e obtenha $N(\bar{x}, B_{hs})$.
   Se $N(\bar{x}, B_{hs}) \subset B_{hs}$, faça $B = B_{hs}$ e,
   se $w(B) > \alpha X$, pare; senão faça $a = \theta a$ e volte ao passo 6.

O parâmetro a_0 é o raio inicial da caixa de *Back-Boxing* B e o parâmetro θ define o fator de expansão desta caixa. Já o parâmetro α serve para limitar o tamanho máximo da caixa B a uma porcentagem da caixa original X.

Em [32], a caixa B pode crescer até se tornar a própria caixa X. Introduzimos um limite no crescimento da caixa B porque acreditamos que eliminar as novas caixas provenientes da divisão do espaço restante através de técnicas mais simples seja mais eficiente do que tentar crescer demasiadamente a caixa B.

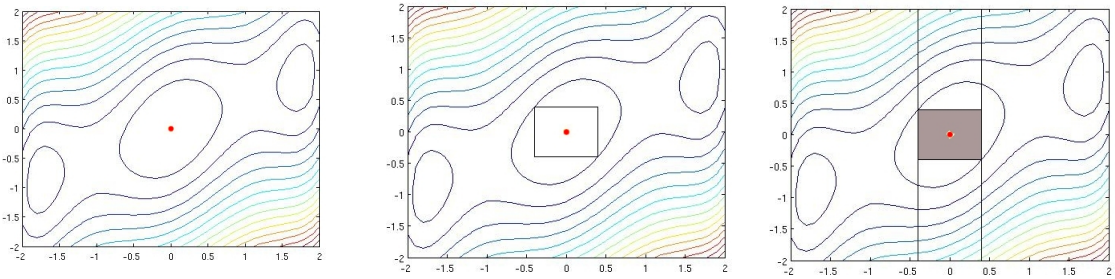

Figura 4.1: Obtendo Back-Box e particionando o espaço restante

Após o crescimento da caixa B o espaço restante é subdividido conforme o algoritmo a seguir.

Algoritmo 5: Subdivisão do espaço restante

1. Faça $X^d = X$, $B = B \cap X$.
2. Efetue a divisão das coordenadas em ordem decrescente de tamanho.
3. Para a j-ésima coordenada, faça:
   $X^1 = X^2 = X^d$,
   $X_j^1 = [\underline{X_j^1}, \underline{B_j}]$, $X_j^2 = [\overline{B_j}, \overline{X_j^2}]$
4. Adicione X^1 e X^2 à lista L_p.
5. Atualize $X_j^d = B_j$, faça $j = j + 1$ e retorne ao passo 3.

A Figura 4.1 ilustra o processo aplicado a função *Tree Hump Camel Back*. Um minimizador local é localizado no ponto $(0, 0)^T$. A seguir, enquanto a unicidade de tal minimizador é verificada, uma caixa ao redor do mesmo é ampliada. Então, tal caixa é eliminada e o restante do espaço de busca particionado.

4.6 Busca Local

A busca local desempenha um papel importante em algumas partes de nosso algoritmo.

Lembrando que o primeiro teste capaz de descartar uma caixa X é $\underline{f(X)} > \bar{f}$, quanto menor o valor de $\bar{f}$ maior é a probabilidade de uma caixa ser eliminada por esse critério. Além disso, $\bar{f}$ também é utilizado para eliminar toda ou partes da caixa X que não cumprem a desigualdade $f(x) \leq \bar{f}$.

Logo é de interesse melhorar o valor $\bar{f}$ o quanto antes. Portanto a cada nova caixa X selecionada para processamento, avaliaremos $f(m(X))$ e se $f(m(X)) < \bar{f}$ atualizamos este valor. Porém, seguindo a filosofia de que quanto menor $\bar{f}$ melhor, sempre que ocorrer $f(m(X)) < \bar{f}$, não nos contentaremos apenas com o valor de f no ponto médio de X, e aplicaremos um método de busca local à partir de $m(X)$.

Com isso esperamos melhorar o valor de $\bar{f}$ quão logo possível, alcançando assim de forma mais rápida o minimizador global. Lembrando que uma vez localizado um minimizador local $\bar{x}$, podemos utilizar *Back-Boxing* para delimitar uma área

na qual $\bar{x}$ é o único minimizador global, eliminando tal região de X e subdividindo o espaço restante.

Além disso, se ao aplicar o método de Newton intervalar constatarmos a unicidade de um zero de $\nabla f(x)$ em X, então novamente usamos um método local para localizar tal ponto estacionário.

4.7 Um Algoritmo para Otimização Global Irrestrita

Nesta seção propomos um algoritmo BB com as técnicas de análise intervalar apresentadas anteriormente para resolução do problema de otimização global irrestrita.

Como já dito anteriormente, supomos conhecida uma caixa inicial X^0 na qual o minimizador global da função esteja em seu interior. Consideramos também que as derivadas de primeira e segunda ordem são conhecidas explicitamente e informadas como dados de entrada. Isso pode ser evitado mediante uso de *diferenciação automática*, ver [21], [56].

No Algoritmo 6, a lista L_p deve estar ordenada por limitante inferior. Logo, sempre que uma caixa for adicionada à lista ela já deve entrar em sua posição correta. Deste modo, para selecionar a caixa com menor limitante inferior no passo 4 basta tomar a primeira caixa da lista.

O menor limitante superior α será tomado como $\alpha = \bar{f}$, onde $\bar{f}$ denota o menor valor de função objetivo até o momento.

Em vários pontos do algoritmo usamos o termo *progresso suficiente*. Consideramos que uma dada caixa X sofreu uma redução suficiente ou teve um progresso suficiente quando:

$$\max_i \{w(X_i') - w(X_i)\} < \gamma w(X),$$

para $\gamma \in (0, 1)$, onde X' denota a caixa reduzida.

Procuramos no Algoritmo 6 empregar as técnicas de análise intervalar em ordem crescente de custo computacional, onde a primeira é a avaliação de $f(X)$ e o

teste de limitante inferior $\underline{f(X)} > \bar{f}$ e a última é o método de Newton intervalar. Assim, caso a caixa X já tenha sofrido uma redução suficiente, dispensamos as técnicas mais caras e retornamos X a lista L_p para processamento posterior.

Não fazemos uso de *box consistency* em todas as caixas processadas. Em [26], Hansen e Walster afirmam que *box consistency* funciona melhor que o método de Newton intervalar em caixas grandes. A técnica de *box consistency* só é realizada em uma determinada caixa X, se este procedimento foi bem sucedido na caixa pai de X. Bem sucedido significa que:

$$\max_i \{w(X_i') - w(X_i)\} < \gamma' w(X),$$

para $\gamma' \in (0, \gamma)$, onde X' denota a caixa reduzida após *box consistency*.

Também não aplicamos o método de Newton Intervalar a todas as caixas. Em [25], [26], é observado que o método de Newton Intervalar funciona melhor em caixas pequenas. Para decidir se uma caixa é pequena adotamos, em nosso algoritmo, a estratégia a seguir.

Definindo w_I como o tamanho da menor caixa para a qual o método de Newton intervalar não foi capaz de realizar redução alguma, e w_R como o tamanho da maior caixa para a qual o método de Newton intervalar foi capaz de realizar alguma redução. Iniciamos com $w_I = w(X^0)$ e $w_R = 0$, e este valores são atualizados após cada aplicação do método de Newton intervalar. Daí, para verificar se o método de Newton intervalar deve ser utilizado em uma determinada caixa X, verificamos se:

$$w(X) < (1 - \lambda) w_R + \lambda w_I,$$

com $\lambda \in [0, 1]$.

Em [26], é tomado $\lambda = 0.5$. Assim, o método de Newton intervalar é aplicado a caixa X apenas se esta tiver largura inferior a média aritmética de w_R e w_I. Quanto mais próximo de 1 estiver λ, mais ênfase será dada ao método de Newton Intervalar, já que a tolerância com relação ao tamanho da caixa será maior.

O Algoritmo 6 retorna uma lista de caixas L_r na união das quais se encontra um minimizador global de f em X^0. Note que o algoritmo pode ser finalizado antes de encontrar, ou delimitar por pequenas caixas, todos os minimizadores globais, consequência do critério de parada $\bar{f} - \beta < \varepsilon_f$. Caso seja de interesse

encontrar todos os minimizadores globais de f em X^0, basta remover este critério de parada do passo 3.

Algoritmo 6: branch-and-bound com técnicas de análise intervalar

1. Dados f,∇f, $\nabla^2 f$, X^0, ε_f, ε_X, γ, γ'
2. Calcule $f(X^0)$ e faça $\bar{f} = \overline{f(X^0)}$ e $\beta = \underline{f(X^0)}$.
   Inicialize $L_p = X^0$, $L_r = \emptyset$, $w_I = w(X^0)$, $w_R = 0$.
3. Faça $\beta = \min_{X \in L_p} \underline{f(X)}$.
   Remova de L_r as caixas X tais que $\underline{f(X)} > \bar{f}$.
   Se $L_p = \emptyset$ ou $\bar{f} - \beta < \varepsilon_f$ pare. Retorne a lista L_r.
4. Faça X igual a primeira caixa na lista L_p (aquela que possui o *menor limitante inferior*). Remova tal caixa de L_p.
5. Se $w(X) < \varepsilon_X$ ou $\overline{f(X)} - \underline{f(X)} < \varepsilon_f$, coloque X em L_r e retorne ao passo 3.
6. Avalie $f(m(X))$ e se $f(m(X)) < \bar{f}$, faça $\bar{f} = f(m(X))$.
7. Se no passo anterior $f(m(X)) < \bar{f}$, aplique um *método de busca local* restrito a X a partir de $m(X)$. Atualize $\bar{f}$ se necessário.
8. Remova da lista L_p todas as caixas Y tais que $\underline{f(Y)} > \bar{f}$.
9. Aplique o *teste de monotonicidade*, avaliando as componentes de $\nabla f(X)$. Se para algum i, $0 \notin \nabla f(X)_i$, descarte X, retorne ao passo 3. Caso contrário, armazene $\nabla f(X)$.
10. Se o método local foi acionado no passo 7, aplique *Back-Boxing* como no Algoritmo 4. Adicione a caixa B a lista L_r, forçando $\underline{f(B)} = \bar{f}$. Subdivida o espaço restante conforme o Algoritmo 5. Calcule $f(X^i)$ para as novas caixas geradas, e elimine aquelas tais que $\underline{f(X^i)} > \bar{f}$. Adicione as novas caixas restantes à lista L_p por ordem de limitante inferior. Retorne ao passo 3.
11. Avalie a *forma do Valor Médio*, $f_g(X) = f(m(X)) + \nabla f(X)^T (X - m(X))$ e faça interseção de $f_g(X)$ com $f(X)$. Atualize os limitantes $\underline{f(X)}$ e $\overline{f(X)}$.
    Se $\underline{f(X)} > \bar{f}$ remova X e retorne ao passo 3.
    Se $\overline{f(X)} - \underline{f(X)} < \varepsilon_f$, coloque X na lista L_r, e retorne ao passo 3.

12. Aplique *desigualdade intervalar* sobre $f(x) \leq \bar{f}$, como na Seção 4.5.3, para eliminar toda ou parte da caixa X. Caso haja *gap*, armazene essa informação. Se toda caixa X foi eliminada, retorne ao passo 3, senão atualize a caixa X. Se houve progresso suficiente, recalcule $f(X)$ para atualizar os limitantes. Se $\underline{f(X)} > \bar{f}$, elimine X, senão coloque X em L_p e retorne ao passo 3.

13. Se na caixa pai de X, *box consistency* aplicado a *desigualdade* $f(x) \leq \bar{f}$ teve bons resultados, aplique *box consistency* sobre $f(x) \leq \bar{f}$ em X, conforme Seção 4.5.4. Se a caixa X foi eliminada, retorne ao passo 3, senão atualize X e guarde informação sobre a qualidade na redução.
    Se houve progresso suficiente, recalcule $f(X)$. Se $\underline{f(X)} > \bar{f}$, elimine X, senão coloque X em L_p e retorne ao passo 3.

14. Se na caixa pai de X, *box consistency* aplicado a $\nabla f(x) = 0$ teve bons resultados, aplique *box consistency* sobre $\nabla f(x) = 0$ em X, conforme Seção 4.5.4. Se a caixa X foi eliminada, retorne ao passo 3, senão atualize X e guarde informação sobre a qualidade na redução.
    Se houve progresso suficiente, recalcule $f(X)$. Se $\underline{f(X)} > \bar{f}$, elimine X, senão coloque X em L_p e retorne ao passo 3.

15. Se no passo anterior, *box consistency* foi aplicada, aproveite as avaliações de $\nabla^2 f(X)_{ii}$ e aplique o *teste de não convexidade*. Se para algum i, $\nabla^2 f(X)_{ii} < 0$, elimine a caixa X e retorne ao passo 3.

16. Se $w(X) < (1-\lambda)w_R + \lambda w_I$, então avalie $J(x, X) = \nabla^2 f(X)$, e aplique o *método de Newton Intervalar* sobre $\nabla f(x) = 0$ em X, como no Algoritmo 3. Se o teste de não convexidade ainda não foi realizado, efetue-o durante o método de Newton.
    Se o método de Newton comprovou não existência, elimine X e retorne ao passo 3.
    Se o método de Newton comprovou unicidade, então aplique um *método de busca local* restrito a X a partir de $m(X)$, e caso necessário atualize $\bar{f}$. Remova da lista L_p todas as caixas Y tais que $\underline{f(Y)} > \bar{f}$. Se $\bar{f}$ foi atualizado, faça $\underline{f(X)} = \bar{f}$. Coloque X na lista L_r e retorne ao passo 3.
    Por fim, se houve redução, atualize X. Se houve *gap*, guarde essa informação. Se a redução gerou um progresso suficiente, recalcule $f(X)$.Se $\underline{f(X)} > \bar{f}$, elimine X, senão coloque X em L_p e retorne ao passo 3.

17. Caso haja informação de *gap* proveniente dos passos 12 e 16, sobreponha os gaps e escolha uma das coordenadas com *gap* para efetuar a *bisseção*. Removendo o *gap* escolhido de X, são geradas duas novas caixas X^1 e X^2. Calcule $f(X^1)$ e $f(X^2)$. Se $\underline{f(X^1)} > \bar{f}$ elimine X^1, senão adicione X^1 a lista L_p (em ordem de limitante inferior). Se $\underline{f(X^2)} > \bar{f}$ elimine X^2, senão adicione X^2 a lista L_p. Retorne ao passo 3.

18. Faça a *bisseção* de X ao longo da coordenada determinada pela Regra (4.2). Para as duas caixas geradas X^1 e X^2, calcule $f(X^1)$ e $f(X^2)$, efetue o teste de limitante inferior, e em seguida adicione-as a lista L_p.
Retorne ao passo 3.

4.8 Branch-and-bound e DIRECT

No Capítulo 2 estudamos o método DIRECT e suas principais propriedades. O DIRECT pode ser abordado como um algoritmo do tipo *branch-and-select*, com o esquema de partição como definido na Seção 2.3, e o critério de seleção baseado na definição de retângulos potencialmente ótimos.

Vimos que um dos pontos fortes do DIRECT é a forma como o método explora o espaço de busca, balanceando a ênfase entre busca local e busca global.

Por outro lado fazendo uso apenas de avaliações de função e, sem o conhecimento da constante de Lipschitz propriamente dita, o DIRECT não é capaz de realizar uma busca completa, descartando retângulos não promissores, obtendo limitantes para a função objetivo em um determinado retângulo e por fim verificando a otimalidade global.

Já um algoritmo do tipo branch-and-bound com técnicas de análise intervalar, como o Algoritmo 6, contempla todas essas características.

Propomos uma variação no Algoritmo 6 quanto ao critério de seleção e ao esquema de partição.

Ao invés de selecionar a caixa X associada ao menor limitante inferior, passo 4 do Algoritmo 6, usaremos o critério de retângulos (caixas) potencialmente ótimos do DIRECT.

Além disso ao invés da bisseção da caixa X, passo 18 do Algoritmo 6, adotaremos o esquema de partição do DIRECT, dividindo a caixa X ao longo das maiores coordenadas por ordem de função objetivo, como na Seção 2.3.

A esta variação do Algoritmo 6, denominaremos **Algoritmo 7**, mas não apresentaremos uma listagem explícita devido às semelhanças entre este algoritmos.

Como o critério de retângulos potencialmente ótimos requer o valor de função no ponto médio das caixas, tal informação deve ser armazenada para cada caixa da lista L_p. Como tal avaliação ocorre no esquema de partição, o passo 6 do Algoritmo 6 pode ser omitido no Algoritmo 7.

Na Seção 4.3, vimos que se o esquema de obtenção de limitantes é consistente e o critério de seleção é bound-improving, então o processo branch-and-bound converge.

O esquema de obtenção de limitantes do Algoritmo 7 é o mesmo daquela seção.

Logo, nos resta provar que o critério de seleção através de retângulos potencialmente ótimos é bound-improving, provando assim que o Algoritmo 7 é convergente.

Proposição 4.8.1. *O critério de seleção baseado em retângulos potencialmente ótimos, como no* DIRECT*, é bound-improving.*

Prova. Seja X_{lb}^k a caixa associada ao menor limitante inferior f_{lb}^k na iteração k. Se X_{lb}^k é potencialmente ótima, será selecionada na própria iteração k.
Caso contrário, pelas observações na Seção 2.4 decorrentes da Definição 2.4.1 de retângulos potencialmente ótimos, temos que a cada iteração sempre será selecionada dentre as maiores caixas aquela com menor valor de função objetivo. Assim, após um número finito de iterações, teremos que:

$$w(X_{lb}^k) > w(X), \forall X \in L_p,\ X \neq X_{lb}^k,$$

então X_{lb}^k será potencialmente ótima e portanto selecionada. ∎

Desse modo, temos que o Algoritmo 7 é um algoritmo branch-and-bound convergente.

Apesar do possível maior número de avaliações de função do Algoritmo 7 em relação ao Algoritmo 6, devido ao esquema de divisão inspirado no DIRECT, esperamos que o Algoritmo 7 herde as boas características do DIRECT e explore o espaço de busca de maneira mais efetiva.

Capítulo 5

Resultados Computacionais

Neste capítulo apresentamos os detalhes na implementação do algoritmo DIRECT e dos algoritmos propostos nas Seções 4.7 e 4.8. A seguir são apresentados os problemas teste extraídos da literatura [8], [9], [30], [33], [41]. Uma comparação entre os algoritmos é apresentada e os resultados computacionais expostos e analisados.

5.1 Detalhes da Implementação

Para a implementação do algoritmo DIRECT e dos demais algoritmos propostos usamos o software MATLAB®, em sua versão 2007a. Os testes computacionais foram realizados em um Intel Core 2 Duo, com 2GB de RAM, e sistema operacional Microsoft Windows XP®.

Para utilizar a aritmética intervalar no MATLAB utilizamos o pacote INTLAB, [64], em sua versão 5.5, disponível na internet:

http://www.ti3.tu-harburg.de/~rump/intlab.

Infelizmente tal pacote não implementa aritmética intervalar estendida, ou seja, divisão por um intervalo contendo zero. As rotinas desse pacote associadas a divisão na aritmética intervalar foram alteradas para tratar de alguns casos da divisão por um intervalo contendo zero. Para o caso em que a divisão contendo zero resulta em dois intervalos semi-infinitos o tratamento foi realizado à parte nas rotinas onde tal resultado é relevante, como na iteração de Gauss-Seidel Intervalar e Newton Intervalar.
O pacote INTLAB fornece ainda, suporte para diferenciação automática, em modo

avançado, [56], [60], [64]. Assim, na implementação dos algoritmos propostos, pode-se evitar o fornecimento das expressões para as derivadas de primeira e segunda ordem.

Nos Algoritmos 6 e 7, quando precisamos de um *solver local* (com restrições de caixa) utilizamos o comando `fmincon` do `MATLAB`. Para o problema de minimização com restrições de canalização `fmincon` utiliza um método tipo Região de Confiança, [6].

5.2 Problemas Teste

Testamos os algoritmos implementados sobre um conjunto de problemas clássicos da literatura em otimização global, que podem ser encontrados em [8], [9], [30] [33]. Incluímos também ao conjunto de problemas teste um problema específico relacionado a minimização de energia potencial, de grande interesse em Engenharia Química, ver [41].

A formulação de cada um dos problemas é dada a seguir:

1. **Branin**

$$\min_x \quad \left(x_2 - \frac{5x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right) cos(x_1) + 10$$

$$\begin{array}{rrcccl} \text{s.a} & -5 & \leq & x_1 & \leq & 10 \\ & 0 & \leq & x_2 & \leq & 15 \end{array}$$

   Possui 3 minimizadores locais que são globais, $f^* = 0.3979$.

2. **Booth**

$$\min_x \quad (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

$$\text{s.a} \quad -10 \leq x_i \leq 10 \text{ para } i = 1, 2.$$

   O mínimo global é $f^* = 0$.

3. **Rastrigin**

$$\min_x \qquad 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$$

$$\text{s.a} \quad -5.12 \;\leq\; x_i \;\leq\; 5.12 \;\text{ para }\; i = 1, 2, \ldots, n.$$

Possui vários mínimos locais. O mínimo global é $f^* = 0$. Neste problema usamos $n = 2$.

4. O mesmo que o Problema 3, com $n = 5$.

5. **Michalewicz**

$$\min_x \qquad -\sum_{i=1}^{n} sin(x_i)(sin(\frac{i}{\pi}x_i^2))^{2m}$$

$$\text{s.a} \quad 0 \;\leq\; x_i \;\leq\; \pi \;\text{ para }\; i = 1, 2, \ldots, n.$$

Possui vários mínimos locais. Neste problema usamos $m = 10$ e $n = 2$. O mínimo global é $f^* = -1.8013$.

6. O mesmo que o Problema 5, com $m = 10$ e n=5. O mínimo global é $f^* = -4.687658$

7. **Griewank**

$$\min_x \qquad \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$\text{s.a} \quad -600 \;\leq\; x_i \;\leq\; 600 \;\text{ para }\; i = 1, 2, \ldots, n.$$

Possui vários mínimos locais. O mínimo global é $f^* = 0$. Neste problema usamos $n = 2$.

8. **Easom**

$$\min_x \quad -cos(x_1)cos(x_2)exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$$

$$\text{s.a} \quad -100 \;\leq\; x_i \;\leq\; 100 \;\text{ para }\; i = 1, 2, \ldots, n.$$

O mínimo global é $f^* = -1$.

9. **Three Hump Camel Back**

$$\begin{array}{ll} \min\limits_{x} & 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 - x_1x_2 + x_2^2 \\ \\ \text{s.a} & \begin{array}{rcccl} -3 & \leq & x_1 & \leq & 3 \\ -2 & \leq & x_2 & \leq & 2 \end{array} \end{array}$$

Possui 3 mínimos locais dos quais 1 é global, com $f^* = 0$.

10. **Six Hump Camel Back**

$$\begin{array}{ll} \min\limits_{x} & (4 - 2.1x_1^2 + \frac{1}{3}x_1^4)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \\ \\ \text{s.a} & \begin{array}{rcccl} -3 & \leq & x_1 & \leq & 3 \\ -2 & \leq & x_2 & \leq & 2 \end{array} \end{array}$$

Possui 6 mínimos locais, dos quais 2 são globais, com $f^* = -1.0316$.

11. **Hartman 3**

$$\begin{array}{ll} \min\limits_{x} & -\sum\limits_{i=1}^{m} c_i \exp\left(-\sum\limits_{j=1}^{m} a_{ij}(x_j - p_{ij})^2\right) \\ \\ \text{s.a} & 0 \leq x_j \leq 1 \qquad \text{para } j = 1, 2, \ldots, n \end{array}$$

Neste problema $n = 3$, e A, c e P são dados por

$$A = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}, \; c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}, \; P = \begin{pmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{pmatrix},$$

Possui 4 mínimos locais, sendo 1 global com $f^* = -3.8627$.

12. **Hartman 6**. O mesmo que o Problema 11, com $n = 6$ e A, c, P dados por

$$A = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}, \; c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix},$$

$$P = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix}$$

Possui 4 mínimos locais, dos quais 1 é global com $f^* = -3.3223$.

13. **Shekel 5**

$$\min_x \qquad -\sum_{i=1}^{m} \frac{1}{(x-a_i)^T(x-a_i)+c_i}$$

$$\text{s.a} \quad 0 \leq x_j \leq 10 \qquad \text{para } j = 1,2,3,4,$$

onde a_i é a i-ésima linha de A e c_i é o i-ésimo elemento de c. A e c são dados por

$$A = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 4 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{pmatrix}, \; c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.5 \end{pmatrix}.$$

Neste problema usamos $m = 5$. Possui 5 mínimos locais dos quais apenas um é global, com $f^* = -10.1531$.

14. **Shekel 7**. O mesmo que o Problema 13, com $m = 7$. Possui 7 mínimos locais dos quais apenas um é global, com $f^* = -10.4029$.

15. **Shekel 10**. O mesmo que o Problema 13, com $m = 10$. Possui 10 mínimos locais dos quais apenas um é global, com $f^* = -10.5364$.

16. **2D Shubert**

$$\min_x \quad \left(\sum_{i=1}^{5} i\cos[(i+1)x_1 + i]\right)\left(\sum_{i=1}^{5} i\cos[(i+1)x_2 + i]\right)$$

$$\text{s.a} \quad -10 \leq x_j \leq 10 \qquad \text{para } j = 1, 2.$$

Possui 760 mínimos locais, sendo 18 globais, com $f^* = -186.7309$.

17. **Rosenbrock**

$$\min_x \qquad \sum_{i=1}^{n-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$$

$$\text{s.a} \quad -5 \leq x_j \leq 10 \qquad \text{para } j = 1, 2, \ldots, n.$$

O mínimo global é $f^* = 0$.

18. **Levy**

$$\min_x \quad sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2(1 + 10sin^2(\pi y_{i+1})) + (y_n - 1)^2$$

$$\text{s.a} \quad -10 \leq x_j \leq 10 \qquad \text{para } j = 1, 2, \ldots, n.$$

onde $y_i = 1 + \frac{x_i - 1}{4}$, para $i = 1, 2, \ldots, n$.
Possui vários mínimos locais. O mínimo global é $f^* = 0$. Neste problema, $n = 3$.

19. O mesmo que o Problema 18, com $n = 4$.

20. O mesmo que o Problema 18, com $n = 5$.

21. O mesmo que o Problema 18, com $n = 8$.

22. O mesmo que o Problema 18, com $n = 10$.

23. Problema extraído de [41].

$$\min_x \qquad \sum_{i=1}^{n} 1 + cos(3x_i) + \frac{(-1)^i}{\sqrt{b + acos(x_i)}}$$

$$\text{s.a} \quad 0 \leq x_j \leq 5 \qquad \text{para } j = 1, 2, \ldots, n.$$

onde $a = -4.141720682$ e $b = 10.60099896$. Possui vários mínimos locais, da ordem de 2^n. Neste problema usamos $n = 5$. O mínimo global é $f^* = -0.5072$.

24. O mesmo que o Problema 23, com $n = 8$. O mínimo global é $f^* = -0.3289$.

25. O mesmo que o Problema 23, com $n = 10$. O mínimo global é $f^* = -0.4112$.

5.3 Resultados Computacionais

Nesta seção apresentamos os resultados computacionais dos algoritmos implementados quando aplicados ao conjunto de problemas teste da Seção 5.2.

Em nossos experimentos computacionais testamos o desempenho dos seguintes algoritmos:

- `DIRECT`: implementação do algoritmo `DIRECT` discutido na Seção 2.3.
- `BBIA1`: implementação do Algoritmo 6, da Seção 4.7, sem o uso de *Back-Boxing*.
- `BBIA2`: implementação do Algoritmo 6, da Seção 4.7, com o uso de *Back-Boxing*.
- `DIRIA1`: implementação do Algoritmo 7, da Seção 4.8, sem o uso de *Back-Boxing*.
- `DIRIA2`: implementação do Algoritmo 7, da Seção 4.8, com o uso de *Back-Boxing*.

Nas tabelas apresentadas nas próximas seções, adotamos a seguinte notação:

#	número do problema teste
n	dimensão do problema teste
FE	número de avaliações de função
GE	número de avaliações de gradiente
HE	número de avaliações de Hessiana
Tempo(s)	tempo em segundos.

Além disso, quando comparamos os quatro últimos algoritmos, estamos nos referindo à busca completa, isto é, todo espaço de busca é verificado e *todos* os mínimos globais devidamente localizados. Nestes testes tomamos $\varepsilon_x = \varepsilon_f = 10^{-4}$.

5.3.1 Parâmetros Envolvidos

Tanto o algoritmo `DIRECT` quanto os algoritmos branch-and-bound propostos nas Seções 4.7 e 4.8 requerem certos parâmetros.

No caso do `DIRECT`, o único parâmetro que precisa ser informado é o parâmetro ε empregado na seleção de retângulos potencialmente ótimos. Nos experimentos computacionais adotamos o valor de $\varepsilon = 10^{-4}$ sugerido em [33].

Já os Algoritmos 6 e 7, da Seção 4.7, requerem uma quantidade maior de parâmetros. Os valores para os parâmetros apresentados a seguir foram os mesmos nas variações `BBIA1`, `BBIA2`, `DIRIA1` e `DIRIA2`.

Com relação ao parâmetro γ empregado no critério de progresso suficiente, tomamos $\gamma = 0.25$, como em [26]. Assim, se uma caixa X sofrer uma redução equivalente a 25% de seu tamanho inicial, consideramos que houve progresso suficiente.

Para γ' relacionado ao critério de aplicação de *box consistency* adotamos $\gamma' = 0.2$. Desse modo, se após a aplicação das técnicas de *box consistency* houve redução de 20% na caixa X, consideramos que *box consistency* foi bem sucedido e portanto será utilizado nas caixas geradas a partir de uma possível subdivisão de X.

O parâmetro λ usado para verificar se uma caixa X é pequena o suficiente para a aplicação do método de Newton Intervalar foi tomado como $\lambda = 1$, isto é, sempre que $w(X) < w_I$, aplicaremos o método de Newton Intervalar.

Temos também os parâmetros empregados nos procedimentos de *Back-Boxing*. Para a_0, o diâmetro inicial da caixa de *Back-Boxing*, utilizamos:

$$a_0 = \begin{cases} 0.01X & \text{se } w(X) \leq 100, \\ 0.001X & \text{caso contrário.} \end{cases}$$

Para α, a porcentagem com relação a caixa inicial X, usamos:

$$\alpha = \begin{cases} 0.05 & \text{se } w(X) > 1, \\ 0.25 & \text{caso contrário.} \end{cases}$$

e para o fator de expansão da *Back-Boxing* tomamos $\theta = 2$.

5.3.2 Perfil de Desempenho

Para a análise e comparação dos métodos usaremos a ferramenta denominada *perfil de desempenho*.

Introduzido por Dolan e Moré [10], o perfil de desempenho é uma ferramenta para comparar o desempenho de n_s algoritmos quando aplicados na resolução de n_p problemas, usando uma *medida de desempenho* tal como o número de avaliações de função ou tempo de execução.

O valor denotado por m_{sp} representa a medida de desempenho para resolver o problema p através do algoritmo s. Seja S o conjunto de algoritmos e P o conjunto de problemas. Para cada problema p e algoritmo s a *taxa de desempenho* é calculada por:

$$r_{sp} = \frac{m_{sp}}{min\left\{m_{sp}\ ,\ \forall s \in S\right\}}$$

se o algoritmo s tem sucesso na resolução do problema p, caso contrário:

$$r_{sp} = r_M$$

onde r_M é um parâmetro pré-fixado, suficientemente grande.

Para cada s, a função de distribuição acumulativa $\rho_s : \mathbb{R} \to [0, 1]$ para a taxa de desempenho r_{sp} é construída:

$$\rho_s(\tau) = \frac{1}{n_p} \#\left\{p \in P \mid r_{sp} \leq \tau\right\},$$

onde $\#A$ denota a cardinalidade de um conjunto A.

Esta função representa o desempenho do algoritmo s na resolução de p problemas e é não decrescente e constante por partes. Na análise do algoritmo s, dois valores nos dão informações importantes: a eficiência deste algoritmo, que é indicada pela porcentagem de problemas resolvidos em menor tempo dada pelo valor $\rho_s(1)$, e sua robustez, que é representada pelo valor $\bar{\tau}$ para o qual $\rho_s(\bar{\tau}) = 1$, se existe tal valor para $\tau < r_M$.

Seja $\bar{s}$ o algoritmo que dá o máximo valor para $\rho_s(1)$. Isto significa que o algoritmo $\bar{s}$ resolve o maior número de problemas com o menor valor para a medida m e portanto é considerado o algoritmo mais eficiente. E, o melhor algoritmo, em termos de robustez, será $\hat{s}$ para o qual $\bar{\tau}_{\hat{s}} = \min_{s \in S} \bar{\tau}_s$.

5.3.3 Comparação entre os algoritmos BB

Nesta seção apresentamos uma comparação entre os algoritmos `BBIA1` e `BBIA2`. Ambos compartilham das mesmas técnicas e estratégias, porém `BBIA2` faz uso de *Back-Boxing* enquanto `BBIA1` não. Com isso poderemos determinar o impacto do uso de *Back-Boxing* na resolução do problema de otimização global via métodos intervalares do tipo branch-and-bound.

As Tabelas 5.1 e 5.2 apresentam o número de avaliações e o tempo exigido em cada algoritmo na resolução dos problemas teste apresentados.

#	n	FE	GE	HE	Tempo(s)
1	2	67	56	16	0.81
2	2	54	33	4	0.42
3	2	56	33	13	0.65
4	5	496	300	125	15.62
5	2	59	45	19	1.59
6	5	540	361	161	58.18
7	2	124	79	37	2.18
8	2	106	85	21	3.21
9	2	84	70	26	1.11
10	2	194	135	36	3.02
11	3	160	90	23	4.99
12	6	965	440	93	66.31
13	4	106	73	30	3.52
14	4	112	78	30	4.18
15	4	118	80	30	5.58
16	2	2578	1350	410	92.62
17	2	119	81	28	1.41
18	3	72	52	20	2.24
19	4	94	66	26	3.92
20	5	112	76	32	6.17
21	8	171	111	50	16.85
22	10	205	130	60	25.59
23	5	96	63	24	4.74
24	8	248	162	68	25.88
25	10	422	274	123	68.26

Tabela 5.1: Resultados `BBIA1`

#	n	FE	GE	HE	Tempo(s)
1	2	88	51	18	1.08
2	2	78	29	7	0.56
3	2	7	3	4	0.06
4	5	13	3	4	0.12
5	2	52	32	13	1.89
6	5	545	364	165	59.83
7	2	7	3	2	0.09
8	2	84	55	17	2.64
9	2	77	63	26	0.99
10	2	200	137	47	3.18
11	3	191	103	31	6.37
12	6	1789	691	84	105.41
13	4	129	55	14	3.26
14	4	232	100	24	7.21
15	4	232	101	24	8.94
16	2	2559	1333	408	92.11
17	2	119	82	29	0.99
18	3	18	12	3	0.29
19	4	22	14	3	0.43
20	5	21	11	3	0.57
21	8	29	13	3	1.16
22	10	32	12	3	1.63
23	5	95	63	25	4.61
24	8	259	176	67	26.33
25	10	435	287	122	68.64

Tabela 5.2: Resultados `BBIA2`

A Figura 5.1 apresenta o gráfico de perfil de desempenho para `BBIA1` e `BBIA2`, Utilizando o tempo (em segundos), como medida de desempenho.

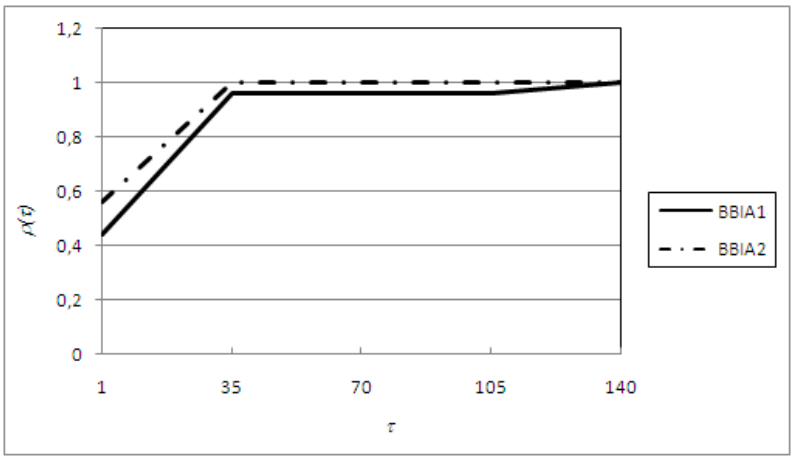

Figura 5.1: Perfil de desempenho para `BBIA1` e `BBIA2`

Podemos ver que o algoritmo `BBIA2` foi mais eficiente que `BBIA1`, resolvendo em um tempo menor, 58% dos problemas. Além disso, `BBIA2` também foi o mais robusto.

O melhor desempenho de `BBIA2` em relação a `BBIA1` se reflete principalmente nos problemas 3, 4, 7 e nos problemas 18 a 22, conforme podemos verificar nas Tabelas 5.1 e 5.2.

Nos problemas 3, 4 e 7 isso ocorre, pois a *Back-Boxing* evita o *problema de cluster*, isto é, quando as subdivisões se dão sobre ou, muito próximo, do minimizador global, fazendo com que o algoritmo perca muito tempo tentando eliminar caixas pequenas próximas ao ótimo global.

Nos problemas de 18 a 22, a primeira busca local já leva ao único ótimo global o qual é devidamente isolado pela *Back-Boxing*, permitindo que o restante do espaço de busca seja facilmente eliminado pelo algoritmo já que para estas funções as extensões intervalares são bem precisas.

Observamos a dificuldade de ambos os algoritmos em resolver o problema 16, que apesar de ter dimensão 2, possui vários mínimos locais, pois a função envolve produto de cossenos. Além disso os limitantes da análise intervalar são prejudicados pela presença das variáveis de decisão neste produto de cossenos, devido ao problema de dependência.

O problema de dependência dificultou também a resolução do problema 12. Apesar da função do problema 12 ser bem comportada, no sentido de possuir apenas 4 mínimos locais, o somatório de termos exponenciais envolvendo todas as variáveis em cada termo, aumentou o problema de dependência e consequentemente reduziu o desempenho das ferramentas intervalares empregadas nos algoritmos.

5.3.4 Comparando BB com BB-DIRECT

Nesta seção apresentamos os resultados obtidos pelos algoritmos DIRIA1 e DIRIA2, que são similares a BBIA1 e BBIA2 exceto pelo critério de seleção e esquema de partição que são os mesmos utilizados no DIRECT.

#	n	FE	GE	HE	Tempo(s)
1	2	76	60	19	0.98
2	2	54	33	4	0.44
3	2	38	15	5	0.24
4	5	80	15	5	0.55
5	2	36	25	7	1.08
6	5	1368	574	260	98.42
7	2	80	42	14	0.83
8	2	108	57	12	1.95
9	2	83	80	23	1.01
10	2	132	109	32	2.28
11	3	246	123	35	7.94
12	6	2586	702	87	123.43
13	4	127	48	16	2.29
14	4	123	60	21	3.27
15	4	133	64	23	4.32
16	2	3361	1070	311	100.41
17	2	273	103	20	1.56
18	3	73	39	13	1.58
19	4	94	49	17	2.83
20	5	126	57	21	4.68
21	8	241	66	26	11.63
22	10	347	71	29	18.65
23	5	260	160	68	13.06
24	8	419	265	114	40.47
25	10	631	389	176	92.26

Tabela 5.3: Resultados para DIRIA1

#	n	FE	GE	HE	Tempo(s)
1	2	101	63	22	1.18
2	2	78	29	7	0.56
3	2	7	3	4	0.06
4	5	13	3	4	0.13
5	2	52	32	13	1.89
6	5	1369	537	256	102.23
7	2	7	3	2	0.11
8	2	81	45	13	2.06
9	2	77	63	26	1.01
10	2	133	108	36	2.37
11	3	254	133	43	8.89
12	6	3254	854	45	146.22
13	4	161	45	12	2.94
14	4	241	80	24	5.95
15	4	235	74	20	6.91
16	2	3514	901	90	99.81
17	2	273	104	21	1.55
18	3	18	12	3	0.31
19	4	22	14	3	0.43
20	5	21	11	3	0.57
21	8	29	13	3	1.17
22	10	32	12	3	1.62
23	5	267	163	76	13.61
24	8	477	345	139	51.91
25	10	825	512	297	120.61

Tabela 5.4: Resultados para DIRIA2

As Tabelas 5.3 e 5.4 apresentam o número de avaliações e o tempo exigido em cada algoritmo na resolução dos problemas teste apresentados.

Utilizando o tempo (em segundos), como medida de desempenho, temos o seguinte perfil de desempenho para DIRIA1 e DIRIA2, apresentado na Figura 5.2.

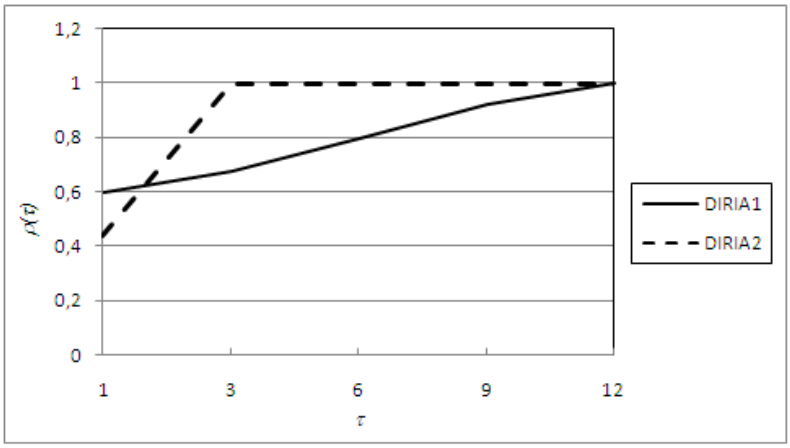

Figura 5.2: Perfil de desempenho para DIRIA1 e DIRIA2

Apesar de DIRIA1 ser mais eficiente que DIRIA2, vemos que o segundo algoritmo é bem mais robusto que o primeiro. Note ainda que a diferença nos tempos de DIRIA1 e DIRIA2 nos problemas 3, 4 e 7 são menores que a diferença entre BBIA1 e BBIA2, para os mesmos problemas. Isso ocorre graças ao esquema de subdivisão baseado no DIRECT, que contribui para a redução do *problema de cluster*.

Apresentaremos agora uma comparação dos quatro algoritmos propostos. Na Figura 5.3 apresentamos o perfil de desempenho, tomando o tempo como medida de desempenho, para os algoritmos BBIA1, BBIA2, DIRIA1 e DIRIA2.

Analisando este perfil de desempenho verificamos que BBIA2 é o algoritmo mais eficiente e robusto dentre os algoritmos propostos, sendo seguido pelo DIRIA2 em termos de robustez.

No perfil de desempenho da Figura 5.3, é interessante notar que DIRIA1 foi mais robusto que BBIA1. Isso indica que a modificação no algoritmo branch-and-bound intervalar BBIA1 para utilizar o critério de seleção e o esquema de partição do DIRECT contribuiu para aumentar a robustez do algoritmo.

Traçando agora um perfil de desempenho entre BBIA2 e DIRIA2, verificamos pela

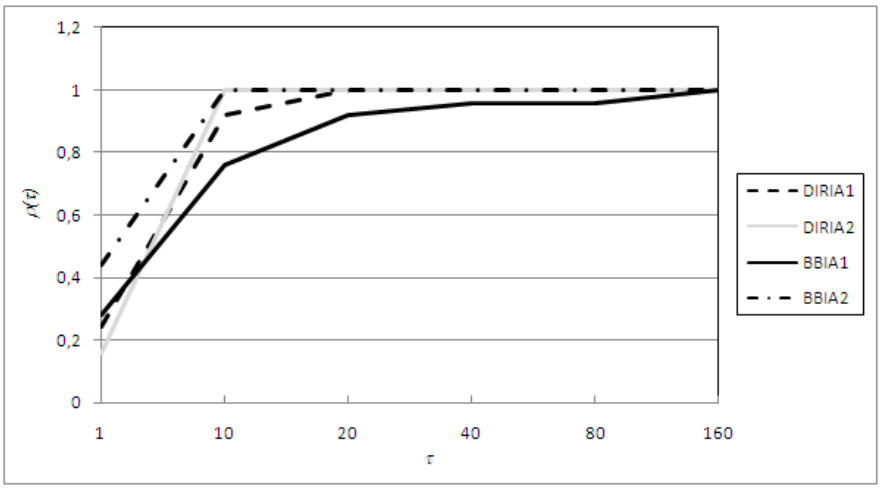

Figura 5.3: Perfil de desempenho para os algoritmos propostos

Figura 5.4 que de fato, `BBIA2` é mais eficiente e robusto que `DIRIA2`.

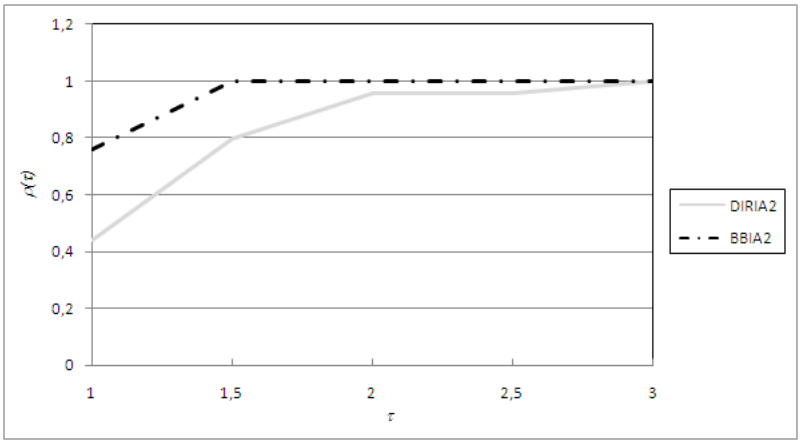

Figura 5.4: Perfil de desempenho entre `BBIA2` e `DIRIA2`

O esquema de divisão do `DIRECT` realiza a trisseção de uma dada caixa ao longo das maiores coordenadas. Assim, se todas os lados da caixa forem de mesmo tamanho o `DIRECT` realizará $2n$ subdivisões, gerando $2n + 1$ novas caixas (não devemos esquecer da caixa central). Por outro lado o esquema de divisão adotado no Algoritmo 6 e na maioria dos métodos branch-and-bound, realiza a bisseção ao longo de apenas uma coordenada, gerando apenas duas novas caixas.

Essa seja talvez a razão pela qual o esquema de subdivisão clássico de BBIA2 foi mais efetivo do que o esquema inspirado no DIRECT de DIRIA2. O esquema de divisão do DIRECT gera bem mais divisões que o esquema clássico de bisseção. Por um lado isso parece bom, pois mais divisões significam caixas menores reduzindo as sobreestimativas nas funções de inclusão. Por outro lado, talvez não sejam necessárias tantas divisões para que as estimativas intervalares se tornem boas, e além disso à medida em que a dimensão n cresce, o número de subdivisões pelo esquema DIRECT se torna bastante elevado.

5.3.5 Comparação com o DIRECT

Apresentamos aqui, uma comparação entre DIRECT, BBIA2 e DIRIA2. O DIRECT não tem a capacidade de verificar se alcançou um minimizador global, logo a comparação cabível entre DIRECT e os outros dois algoritmos é com relação ao custo necessário para atingir o mínimo global, sem verificar este fato.

Como os valores de mínimo global f_{min} são conhecidos para todos os problemas teste, forçamos o critério de parada:

$$\frac{\bar{f} - f_{min}}{1 + f_{min}} < \varepsilon_f,$$

onde $\bar{f}$ denota o menor valor de f encontrado até o momento.

Nos teste realizados, pedimos a precisão $\varepsilon_f = 10^{-2}$ e consideramos que um algoritmo obteve sucesso se o mínimo global foi obtido, dentro da precisão desejada, em até 20000 avaliações de função.

Além disso, DIRECT faz uso apenas de avaliações de função, enquanto BBIA2 e DIRIA2 fazem uso de avaliações de função, gradiente e Hessiana. Logo, definimos a seguinte medida de avaliações para este algoritmos:

$$FE + nGE + \left(\frac{n^2 + n}{2}\right) HE.$$

Nos problemas onde o ponto central do domínio inicial é o minimizador global, realizamos uma pequena perturbação nos limitantes das variáveis, para evitar que os métodos parem logo no primeiro ponto visitado.

A Tabela 5.5 apresenta o número de avaliações necessário em cada algoritmo para alcançar o mínimo global. O símbolo † indica que o algoritmo não foi capaz de alcançar com a precisão desejada o mínimo global dentro do limite estabelecido de avaliações.

	1	2	3	4	5	6	7	8	9
DIRECT	163	239	149	559	67	15241	163	7025	65
BBIA2	43	40	25	88	49	4632	19	245	25
DIRIA2	23	20	11	20	52	7880	17	210	38
	10	11	12	13	14	15	16	17	
DIRECT	207	199	571	151	145	145	2599	1267	
BBIA2	409	76	145	83	98	98	1426	80	
DIRIA2	144	82	157	88	104	103	2581	62	
	18	19	20	21	22	23	24	25	
DIRECT	213	387	691	2155	3843	3543	†	†	
BBIA2	72	108	121	241	317	465	4047	9972	
DIRIA2	42	62	56	101	112	2204	8139	16243	

Tabela 5.5: Avaliações necessárias até alcançar o mínimo global

Porém o número de avaliações de função não reflete de fato o desempenho dos algoritmos, pois no caso de BBIA2 e DIRIA2 estão envolvidas no processo, operações de precondicionamento, resolução de sistemas lineares, etc.

Logo, para o perfil de desempenho utilizaremos como medida de desempenho o tempo em segundos necessário para alcançar o mínimo global.

A Tabela 5.6 apresenta o tempo em segundos gasto pelos algoritmos. A Figura 5.5 apresenta o gráfico de perfil de desempenho para os algoritmos.

Pelo perfil de desempenho verificamos que o DIRECT mostrou-se o mais eficiente dos algoritmos, no sentido de encontrar mais rapidamente o ótimo global, o que ocorreu em 64% dos problemas. No entanto, o DIRECT falhou na resolução de dois problemas, não atingindo o mínimo global dentro do limite de avaliações de função pré-estabelecido.

	1	2	3	4	5	6	7	8	9
DIRECT	0.05	0.06	0.03	0.12	0.01	17.11	0.03	29.67	0.01
BBIA2	0.16	0.07	0.05	0.11	0.41	57.11	0.08	2.66	0.07
DIRIA2	0.05	0.04	0.02	0.03	0.41	102.34	0.03	2.21	0.17

	10	11	12	13	14	15	16	17
DIRECT	0.05	0.05	0.15	0.05	0.05	0.05	1.38	0.32
BBIA2	2.17	0.41	1.21	0.25	0.31	0.36	19.22	0.11
DIRIA2	0.71	0.41	1.21	0.25	0.31	0.36	52.59	0.05

	18	19	20	21	22	23	24	25
DIRECT	0.04	0.07	0.15	0.71	1.38	1.16	†	†
BBIA2	0.27	0.41	0.53	1.09	1.55	2.61	26.11	67.92
DIRIA2	0.04	0.06	0.06	0.09	0.11	13.87	50.05	117.31

Tabela 5.6: Tempo gasto para alcançar o mínimo global

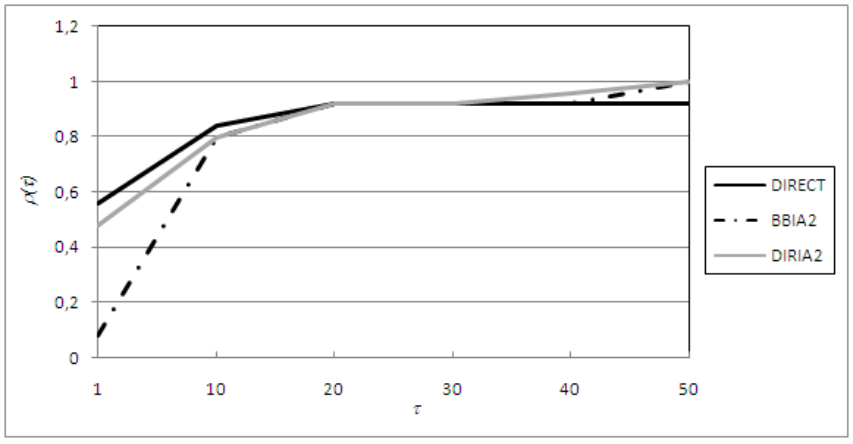

Figura 5.5: Perfil de desempenho DIRECT, BBIA2 e DIRIA2

Os outros dois algoritmos foram capazes de atingir a precisão desejada em todos os problemas, porém exigindo quase sempre um tempo bem maior que DIRECT.

Assim, apesar do DIRECT não conseguir verificar a otimalidade global, ou mesmo ter uma garantia de que uma aproximação para o mínimo global será alcançada em um número finito de iterações, o método explora de maneira eficiente o espaço de busca, usando apenas avaliações de função.

Essa característica do DIRECT é importante, pois permite que o mesmo seja utilizado como um gerador de pontos iniciais, identificando regiões promissoras no espaço de busca, onde um método local possa convergir mais rapidamente a solução e/ou atingir a precisão desejada.

Além disso o DIRECT também é um atrativo em problemas onde as funções envolvidas são do tipo caixa-preta (*black-box*), isto é, quando não se conhece uma expressão para a função, tampouco informações sobre suas derivadas.

5.3.6 Comparação com Trabalhos Anteriores

Nesta seção comparamos nossos algoritmos BBIA2 e DIRIA2, com um algoritmo baseado em branch-and-bound intervalar, também implementado em MATLAB com o pacote INTLAB.

Denotaremos por INTGLOBAL, o algoritmo implementado em MATLAB, discutido no trabalho de Pál e Csendes, [57]. Neste trabalho é proposto um algoritmo tipo branch-and-bound que faz uso de algumas técnicas de análise intervalar tratadas no Capítulo 3. Além disso, os autores fornecem um link para download do código fonte em MATLAB.

No trabalho de Pál e Csendes é realizada também uma comparação entre a implementação feita em MATLAB e outra feita em linguagem C. Tal comparação foi muito importante para nós, pois forneceu uma idéia do quanto o tempo de processamento pode variar de um algoritmo implementado em C e do mesmo algoritmo implementado em MATLAB. Pál e Csendes relatam que apesar do número de avaliações de função, gradiente e Hessiana permanecerem praticamente inalterados da versão em C para a versão em MATLAB, o tempo de processamento é em média 600 vezes maior no MATLAB.

As principais diferenças entre nossos algoritmos e INTGLOBAL, é que INTGLOBAL não faz uso de busca local e nem do processo de *Back-Boxing*. No algoritmo INTGLOBAL é empregado um esquema de trisseção ao invés da bisseção.

Além disso, no método de Newton Intervalar a caixa é dividida ao longo de todas as coordenadas que contêm *gap*, ao invés de armazenar tais informações e decidir posteriormente uma coordenada na qual se usará o *gap*.

#	n	FE	GE	HE	Tempo(s)
1	2	210	147	11	2.97
2	2	49	33	2	0.38
3	2	171	118	15	1.38
4	5	2859	1849	169	26.25
5	2	60	40	3	1.48
6	5	1657	1128	44	96.22
7	2	381	261	33	5.16
8	2	89	62	7	1.11
9	2	303	211	19	4.3
10	2	612	393	19	10.61
11	3	168	123	1	5.95
12	6	976	710	20	61.63
13	4	126	86	7	5.83
14	4	121	78	6	7.48
15	4	123	78	6	10.66
16	2	3098	2090	131	120.33
17	2	142	94	9	1.02
18	3	72	49	4	2.14
19	4	107	71	5	4.11
20	5	135	91	7	6.69
21	8	217	146	9	16.55
22	10	274	188	11	26.42
23	5	606	406	8	25.36
24	8	5043	3130	17	321.47
25	10	21576	12853	48	1793.69

Tabela 5.7: Resultados para `INTGLOBAL`

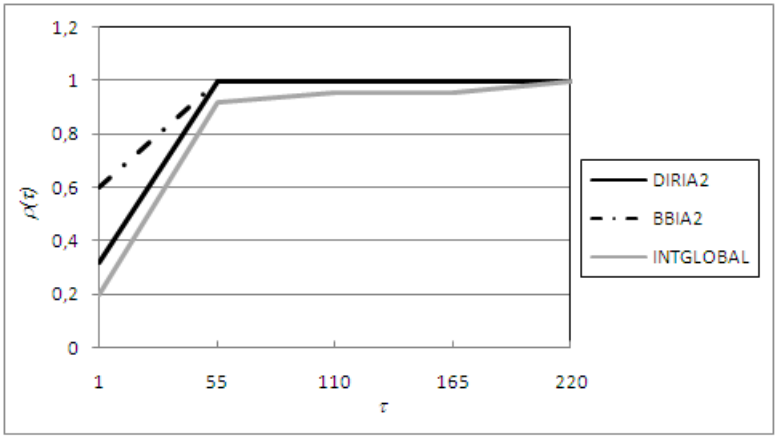

Figura 5.6: Perfil de desempenho `BBIA2`, `DIRIA2` e `INTGLOBAL`

Os resultados para o algoritmo `INTGLOBAL` são listados na Tabela 5.7. A Figura 5.6 apresenta o perfil de desempenho para os algoritmos `BBIA2`, `DIRIA2` e `INTGLOBAL`.

Pelo perfil de desempenho verificamos que tanto `BBIA2` quanto `DIRIA2` foram mais eficientes e robustos que `INTGLOBAL`.

Acreditamos que esse melhor desempenho de nossos algoritmos se deve principalmente à busca local, capaz de melhorar o quanto antes a solução f_{min} potencializando os descartes por limitantes $\underline{f(X)} > f_{min}$ e desigualdade intervalar $f < f_{min}$, e ao processo de *Back-Boxing* capaz de evitar o problema de clusters isolando os minimizadores locais à medida que são encontrados.

Capítulo 6

Conclusões e Trabalhos Futuros

Neste capítulo apresentamos nossas conclusões acerca dos métodos estudados, dos algoritmos propostos e resultados computacionais, e comentamos sobre os trabalhos que podem ser realizados futuramente.

6.1 Conclusões

Sem dúvida o problema de otimização global, ainda que de funções continuamente diferenciáveis e sem restrições, é um dos problemas mais difíceis em programação matemática. No entanto há uma busca constante por métodos ou heurísticas eficientes para a resolução dos mesmos, devido a sua grande aplicabilidade em problemas reais.

O método `DIRECT` estudado em detalhes no Capítulo 2, apresenta propriedades interessantes. Explora de maneira inteligente o espaço de busca, balanceando a ênfase entre busca local e busca global. Os clusters criados pelo `DIRECT` atendem uma condição de otimalidade, indicando regiões onde se encontram minimizadores locais e globais. Faz uso apenas de avaliações de função, não exigindo o conhecimento da expressão da função ou de suas derivadas, o que é bom quando as funções envolvidas são do tipo caixa-preta.

Porém a convergência aos mínimos globais do problema só é garantida quando o número de iterações e/ou avaliações de função tende a infinito. Desse modo o `DIRECT` não é capaz de verificar que chegou à solução, e o critério de parada adotado é um limite no número de iterações e/ou avaliações de função.

Em nosso conjunto de problemas teste o DIRECT foi capaz de alcançar o mínimo global em 92% dos problemas, e em 64% dos problemas, mostrou-se mais eficiente que os métodos branch-and-bound intervalares.

Por essas características acreditamos que o DIRECT deva ser usado em conjunto com um método de busca local em um algoritmo tipo duas fases. Primeiro, o DIRECT explora o espaço de busca até um número pré-estabelecido de avaliações de função, identificando regiões promissoras, nas quais podemos acionar um minimizador local.

Já os métodos do tipo branch-and-bound que empregam técnicas de análise intervalar, como apresentado nos Capítulos 3 e 4, são capazes de realizar uma busca completa, verificando todo o espaço de busca e localizando todos os minimizadores globais do problema. É claro que para isso exigem um custo computacional bem maior que o DIRECT além de hipóteses adicionais sobre as funções, porém este é o preço pago pela verificação da otimalidade global.

De fato, comparando os resultados computacionais, verificamos que na maioria dos problemas os algoritmos branch-and-bound intervalares empregam mais tempo verificando a otimalidade global que propriamente encontrando os minimizadores globais do problema.

Quanto às variações dos Algoritmo 6 e 7, percebemos que o uso de *Back-Boxing* torna os algoritmos mais robustos, como foi o caso entre DIRIA1 e DIRIA2, e entre BBIA1 e BBIA2. Neste último, BBIA2 também tornou-se mais eficiente.

Isso ocorre pois, como já observado em [32] e [35], a *Back-Boxing* é capaz de evitar o problema de cluster, fazendo com que o algoritmo não perca tempo explorando regiões numa pequena vizinhança do minimizador global.

Numa comparação entre BBIA1 e DIRIA1, notamos que introduzir os critérios de seleção e partição do DIRECT em um algoritmo branch-and-bound, tornou DIRIA1 mais robusto que BBIA1.

Os algoritmos BBIA1, BBIA2, DIRIA1 e DIRIA2 foram capazes de resolver os problemas testes em sua totalidade. Dentre os quatro, o mais eficiente e robusto foi o algoritmo BBIA2.

Em comparação com trabalhos anteriores [57], verificamos que nossos algoritmos mostram-se bastante competitivos, o que nos motiva a estender nossos estudos para o caso com restrições gerais do problema de otimização global.

6.2 Trabalhos Futuros

Como nosso trabalho abrangeu o estudo de temas bastante extensos, temos várias possibilidades para trabalhos futuros.

Com relação ao algoritmo `DIRECT` podemos estudar sua aplicação em problemas com restrições de igualdade e desigualdade, mediante o uso de uma função de penalização exata. Mesmo tomando uma função de penalização exata não diferenciável, como a l_1, [44], [56], podemos aplicar o `DIRECT` pois o mesmo não requer a diferenciabilidade em momento algum.

Em relação às técnicas de análise intervalar, podemos realizar um estudo de precondicionadores ótimos na resolução de sistemas lineares intervalares, [7], [35], para melhorar o desempenho do método de Newton Intervalar. As técnicas de *box consistency* e *hull consistency*, citadas na Seção 3.6, também podem ser estudadas com mais profundidade.

Quanto aos métodos branch-and-bound intervalares, várias escolhas para os critérios de seleção e esquema de partição podem ser tomadas, [62]. Um estudo mais detalhado do impacto dessas escolhas no desempenho do BB pode ser feito.

O uso de envelopes convexos e subestimativas convexas, [1], [68], apresentados na Seção 3.7, para obtenção de limitantes pode ser estudado mais a fundo e implementado.

Por fim o estudo de algoritmos branch-and-bound intervalares aplicados ao problema de otimização global com restrições gerais deve ser realizado, bem como a implementação dos algoritmos propostos neste trabalho em uma linguagem mais eficiente como `C` ou `Fortran`.

Referências Bibliográficas

[1] C. S. Adjiman, I. P. Androulakis, C. D. Maranas & C. A. Floudas, A global optimisation method, α BB for process design. *Computers and Chemical Engineering*, **20**, pp. 419–424, 1996.

[2] G. Alefeld & J. Herzberger, Über die berechnung der inversen matrix mit hilfe der intervallrechnung, *Elektronische Rechenanlagen*, **12**(5), pp. 259–261, 1970.

[3] G. Alefeld & J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983.

[4] F. A. Al-Khayyal, J. E. Falk, Jointly constrained biconvex programming, *Mathematics of Operations Research*, **8**(2), pp. 273-286, 1983.

[5] F. H. Branin & S. K. Hoo, A method for finding multiple extrema of a function of n variables, in *Numerical Methods of Nonlinear Optimization*, ed. F.A. Lootsma, Academic Press, London, 1972.

[6] T. F. Coleman & Y. Li, An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, **6**, pp. 418–445, 1996.

[7] S. Corsano & M. Marion, Interval linear systems: state of art, *Computational Statistics*, **21**, pp. 365-383, 2006.

[8] L. C. W. Dixon & G. P. Szegö, *Towards Global Optimization*, North-Holland, 1975.

[9] L. C. W. Dixon & G. P. Szegö, *Towards Global Optimization 2*, North-Holland, 1978.

[10] E. D. Dolan & J. J. Moré, Benchmarking optimization software with performance profiles, *Mathematical Programming*, **91**, pp. 201–213, 2002.

[11] P. S. Dwyer, *Linear Computations*, J. Wiley, N.Y., 1951.

[12] L. H. Figueiredo & J. Stolfi, Affine arithmetic: concepts and applications, *Numerical Algorithms*, **37**, pp. 147–58, 2004.

[13] D. E. Finkel & C. T. Kelley, Additive scaling and the DIRECT algorithm, *Journal of Global Optimization*, **36**, pp. 597–608, 2006.

[14] D. E. Finkel & C. T. Kelley, Convergence analysis of DIRECT algorithm. N. C. State University Center for Research in Scientific Computation, Tech Report Number CRSC-TR04-28, 2004.

[15] C. A. Floudas, P. M. Pardalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Berlin, Springer-Verlag, 1990.

[16] J. M. Gablonsky, *Modifications of the DIRECT algorithm*. PhD Thesis. North Carolina State University, Raleigh, North Carolina, 2001.

[17] J. M. Gablonsky & C.T. Kelley, A locally-biased form of the DIRECT algorithm, *Journal of Global Optimization*, **21**, 27–37, 2001.

[18] E. A. Galperin, The cubic algorithm. *Journal of Mathematical Analysis and Applications*, **112**(2), pp. 155–160, 1984.

[19] R. P. Ge, A filled function method for finding a global minimizer of a function of several variables, *Mathematical Programming*, **46**, pp. 191–204, 1990.

[20] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Kluwer Academic Publishers, Boston, 1989.

[21] A. Griewank, On automatic differentiation, in *Mathematical Programming: Recent Developments and Applications*, eds. M. Iri, K. Tanabe, Kluwer Academic Publishers, pp. 83–108, 1989.

[22] E. R. Hansen, On solving systems of equations using interval arithmetic, *Mathematics of Computation*, **22**(102), pp. 374–384, 1968.

[23] E. R. Hansen, A generalized interval arithmetic, in *Interval Mathematics*, ed. K. L. Nickel, Springer-Verlag, New York, pp. 7–18, 1975.

[24] E. R. Hansen & S. Sengupta, Bounding solutions of systems of equations using interval analysis, *BIT*, **21**, pp. 203–211, 1981.

[25] E. R. Hansen, *Global optimization using interval analysis*, M. Dekker, 1992.

[26] E. R. Hansen & G. W. Walster, *Global optimization using interval analysis*, 2nd Edition, M. Dekker, 2004.

[27] P. Hansen & B. Jaumard, Lipschitz optimization, in *Handbook of Global Optimization*, eds. R. Horst & P. M. Pardalos, Kluwer Academic Publishers, pp. 407-495, 1995.

[28] R. J. Hanson, Interval arithmetic as a closed arithmetic system on a computer. Tech. Rep. 197. Jet Propulsion Laboratory, 1968.

[29] R. Horst, P. M. Pardalos & N. V. Thoai, *Introduction to Global Optimization*, Kluwer Academic Publishers, 1995.

[30] R. Horst & P. M. Pardalos, *Handbook of Global Optimization*, Kluwer Academic Publishers, 1995.

[31] R. Horst & H. Tuy, *Global optimization: deterministc approaches*, Springer-Verlag, 1993.

[32] R. V. Iwaarden, *An improved uncostrained global optimization algorithm*, PhD Thesis, Department of Mathematics, University of Colorado at Denver, 1996.

[33] D. R. Jones, C. D. Perttunen & B. E. Stuckman, Lipschitzian optimization without Lipschitz constant, *Journal of Optimization Theory and Applications*, **79**, 1, pp. 157–181, 1993.

[34] W. M. Kahan, A more complete interval arithmetic. Tech. rep. Univ. Toronto, Toronto, Ont., Canada, 1968.

[35] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, Dordrecht, 1996.

[36] S. Kirkpatrick, C. D. Gelatt & M. P. Vecchi. Optimization by simulated annealing, *Science*, **220**, 4598, pp. 671–680, 1983.

[37] R. Krawczyk, Newton-Algorithmen zur bestimmung von nullstellen mit fehlerschranken, *Computing*, **4**, pp. 187–201, 1969.

[38] R. Krawczyk & A. Neumaier, Interval slopes for rational functions and associated centered form. *SIAM Journal of Numerical Analysis*, **22**, pp. 604–616, 1985.

[39] R. Krawczyk & K. Nickel, Die zentrische form in der intervallarithmetik, ihre quadratische konvergenz und ihre inklusionsisotonie, *Computing*, **28**(2), pp. 117–137, 1982.

[40] E. Kreyszig, *Introductory functional analysis with application*, New York, J. Wiley, 1978.

[41] C. Lavor & N. Maculan, A function to test methods applied to global minimization of potential energy of molecules. *Numerical Algorithms*, **35**, pp. 287–300, 2004.

[42] A. Levy & A. Montalvo, The Tunneling algorithm for the global minimisation of functions, *SIAM Journal on Scientific and Statistical Computing*, **6**(1), pp. 15–29, 1985.

[43] Y. Lucet, Faster than fast Legendre transform, linear-time Legendre transform. *Numerical Algorithms*, **16**, pp. 171–185, 1997.

[44] D. G. Luenberger, *Linear and Nonlinear Programming*, Springer, 2nd Edition, 2003.

[45] C. D. Maranas & C. A. Floudas., Finding all solutions of nonlinear system of equations. *Journal of Global Optimization.* **7**(2), pp. 143–182, 1995.

[46] J. M. Martínez & S. A. Santos, *Métodos Computacionais de Otimização*, IMPA, 20^o Colóquio Brasileiro de Matemática, Rio de Janeiro, SBM, 1995.

[47] G. P. McCormick, Computability of global solutions to factorable nonconvex programs: Part I - Convex underestimating problems, *Mathematical Programming*, **10**, pp. 147–175, 1976.

[48] R. H. Mladineo, An algorithm for finding the maximum of multimodal, multivariate function. *Mathematical Programming*, **34**, pp. 188–200, 1986.

[49] R. E. Moore, *Interval Analysis*. Englewood, Prentice-Hall, 1966.

[50] R. E. Moore, *Methods and Applications of Interval Analysis*, Philadelphia, SIAM, 1979.

[51] A. Neumaier. Interval iteration for zeros of system of equations, *BIT*, **25**, pp. 256–273, 1985.

[52] A. Neumaier, Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, **13**, pp. 271–369, 2004.

[53] G. L. Nemhauser & L. A. Wolsey, *Integer and Combinatorial Optimization*, Wiley Interscience, Series in Discrete Mathematics and Optimization, 1988.

[54] K. L. Nickel, *Interval Mathematics*, Springer-Verlag, New York, 1975.

[55] K. L. Nickel, *Interval Mathematics*, Academic Press, New York, 1980.

[56] J. Nocedal & S. J. Wright, *Numerical Optimization*, Springer, 1999.

[57] L. Pál & T. Csendes, INTLAB implementation of an interval global optimization algorithm, *Optimization Methods and Software*, to appear.

[58] J. Pinter, Globally convergent methods for n-dimensional multiextremal optimization, *Optimization*. **17**. pp. 187–202, 1986.

[59] L. B. Rall, *Computational Solution of Nonlinear Operator Equations*, Wiley, New York, 1969.

[60] L. B. Rall, *Automatic Differentiation: Techniques and Applications*, Lecture Notes in Computer Science 120, Springer, 1981.

[61] H. Ratschek, Inclusion functions and global optimization, *Mathematical Programming* **33**, pp. 300–317, 1985.

[62] D. Ratz & T. Csendes, On the selection of subdivision directions in Interval branch-and-bound methods for global optimization, *Journal of Global Optimization* **7**, pp. 183-207, 1995.

[63] J. Rokne & P. Bao, The number of centered forms for a polynomial, *BIT*, **28**, pp. 852–866, 1988.

[64] S. M. Rump, *INTLAB - INTerval LABoratory*, in *Developments in Reliable Computing*, ed. T. Csendes, Kluwer Academic Publishers, pp. 77–104, 1999.

[65] F. Schoen, Stochastic techniques for global optimization: a survey of recent advances. *Journal of Global Optimization*, **1**, pp. 207–228, 1991.

[66] B. O. Shubert, A sequential method seeking the global maximum of a function, *SIAM Journal on Numerical Analysis*, **9**, 3, pp. 379–388, 1972.

[67] T. Sunaga, Theory of an interval algebra and its application to numerical analysis, *RAAG Memoirs*, **2**, pp. 29–46, 1958.

[68] M. Tawarmalani & N. V. Sahinidis, Convex estimations and envelopes of lower semi-continuous functions. *Mathematical Programming*, **93**, pp. 247–263, 2002.

[69] M. Tawarmalani & N. V. Sahinidis, Global optimization of mixed-integer nonlinear programs: A theoretical and computational study, *Mathematical Programming*, **99**(3), pp. 563–591, 2004.

[70] L. T. Watson & R. T. Haftka, Modern homotopy methods in optimization, *Computer Methods in Applied Mechanics and Engineering*, **74**, pp. 289–304, 1989.

[71] L. A. Wolsey, *Integer Programming*, New York, Wiley, 1998.