TEMA Tend. Mat. Apl. Comput., **4**, No. 2 (2003), 197-206.

Um Método Determinístico para Otimização Global

C. LAVOR[1], L.M. CARVALHO[2], Departamento de Matemática Aplicada, Instituto de Matemática e Estatítica, UERJ, Rua São Francisco Xavier 524, 6º andar, bl. D, 20550-900 Rio de Janeiro, RJ, Brasil

N. MACULAN[3], Programa de Engenharia de Sistemas e Computação, COPPE, UFRJ, Cx.P. 68511, 21945-970 Rio de Janeiro, RJ, Brasil.

Resumo. Empregamos um algoritmo determinístico para otimização global baseado em um método *branch and bound*, que utiliza a aritmética intervalar para o cálculo dos limites inferiores. Para testar o algoritmo, utilizamos uma função em que a quantidade de mínimos locais cresce exponencialmente com o aumento do número de variáveis. Resultados computacionais envolvendo problemas com até 25 variáveis são apresentados. Em todos os casos, o mínimo global foi encontrado.

1. Introdução

Um problema de otimização global pode ser definido da seguinte forma: dadas uma função contínua $f : \mathbb{R}^n \to \mathbb{R}$ e $S \subset \mathbb{R}^n$, a região em que buscamos o(s) ponto(s) x^* onde o valor mínimo de f ocorre, encontre o mínimo global $f^* = \min\{f(x) : x \in S\}$ e o conjunto de todos os minimizadores globais de f, $X^*(f) = \{x^* \in S : f(x^*) = f^*\}$. A maioria dos métodos de programação não-linear [9] encontra apenas um *mínimo local*, ou seja, um ponto $y^* \in S$ tal que existe uma vizinhança N de y^*, onde

$$f(y^*) \leq f(x) \text{ para todo } x \in N \cap S.$$

Entretanto, muitos mínimos locais podem existir e os valores correspondentes da função nesses pontos podem variar substancialmente. Um problema de otimização global requer o desenvolvimento de algoritmos que façam a distinção entre esses mínimos locais e localizem o de menor valor.

Em geral, não é possível encontrar o mínimo global exatamente. Qualquer método de otimização global deve levar em conta que um procedimento numérico produz apenas resultados aproximados. Portanto, um problema de otimização

[1]carlile@ime.uerj.br
[2]luizmc@ime.uerj.br
[3]maculan@cos.ufrj.br

global pode ser considerado resolvido se, para algum $\epsilon > 0$, um elemento de algum dos seguintes conjuntos for identificado:

$$\begin{aligned} A_x(\epsilon) &= \{x \in S : ||x - x^*|| \leq \epsilon\} \\ A_f(\epsilon) &= \{x \in S : ||f(x) - f(x^*)|| \leq \epsilon\}. \end{aligned}$$

Os métodos existentes de otimização global podem ser divididos em duas classes: *métodos estocásticos* e *métodos determinísticos*. Mas, independentemente do método usado, deseja-se algum resultado sobre a sua convergência. Em alguns casos, tudo o que pode ser dito é que o método funciona bem em um sentido empírico. Isso está longe de ser satisfatório. Gostaríamos de obter alguma garantia de que o método encontrará um elemento de $A_x(\epsilon)$ ou $A_f(\epsilon)$ em um número finito de passos.

Neste trabalho, empregamos um algoritmo determinístico baseado em um método *branch and bound,* que utiliza a aritmética intervalar para o cálculo dos limites inferiores. Resultados computacionais são apresentados para testar a otimalidade global do algoritmo. Para isso, utilizamos uma função parametrizada pelo número de variáveis, onde o mínimo global é conhecido e a quantidade de mínimos locais cresce exponencialmente com o aumento das variáveis. Problemas com até 25 variáveis foram considerados. Em todos os casos, o mínimo global foi encontrado.

O restante do artigo está organizado da seguinte forma. Na seção 2, descrevemos o método *branch and bound.* Na seção 3, apresentamos algumas definições e resultados básicos da análise intervalar. Na seção 4, o algoritmo utilizado é descrito. Os resultados computacionais são apresentados na seção 5. Na última seção, finalizamos com algumas conclusões.

2. Branch and Bound

Uma abordagem determinística para resolver um problema de otimização global pode ser dada por meio de uma generalização apropriada do método *branch and bound*, que denotaremos por BB. Esse método é uma técnica determinística bem conhecida na área de otimização combinatória [2].

Escrevamos novamente o problema tratado: dadas uma função contínua $f : \mathbb{R}^n \to \mathbb{R}$ e $S \subset \mathbb{R}^n$ a região em que buscamos o(s) ponto(s) x^* onde o valor mínimo de f ocorre, encontre o mínimo global $f^* = \min\{f(x) : x \in S\}$ e o conjunto de todos os minimizadores globais de f, $X^*(f) = \{x^* \in S : f(x^*) = f^*\}$.

Um método BB alterna entre duas etapas principais: *decomposição*, que faz uma subdivisão recursiva do conjunto S; e *estimação*, que faz o cálculo de limites inferiores e superiores para o menor valor de f numa sub-região de S.

Em cada passo desse procedimento, tem-se uma partição de S em subconjuntos S_α ($\alpha \in A$), um limite inferior $\underline{f}(S_\alpha)$ sobre $\min_{x \in S_\alpha} f(x)$ para todo $\alpha \in A$, e um limite superior $\overline{f}$ sobre $\min_{x \in S} f(x)$, representando o menor valor de f encontrado até o momento. Obviamente, subconjuntos S_α para os quais $\underline{f}(S_\alpha) > \overline{f}$ não podem conter um mínimo global e, portanto, são descartados. Se depois de possíveis melhoramentos de $\overline{f}$, algum subconjunto S_α é mantido com $\underline{f}(S_\alpha) < \overline{f}$, então a partição é

refinada e o procedimento se repete. Existem muitas variações desse esquema, cada uma com resultados de convergência para o mínimo global sob condições diferentes ([6], [11]).

O cálculo dos limites inferiores desempenha um papel fundamental no mecanismo BB. Uma técnica freqüentemente utilizada é usar *envelopes convexos* [5]. Outra possibilidade é usar técnicas de análise intervalar.

3. Análise Intervalar

Um dos principais problemas na teoria e na prática de métodos numéricos é o controle dos erros devidos à representação dos números reais em um sistema de números em *ponto flutuante* [3]. Esse sistema representa somente um conjunto finito de números reais. Além disso, a maioria dos números reais estará entre dois números em ponto flutuante e um deles deverá ser escolhido para representá-los. O erro cometido nesse caso é chamado de *erro de arredondamento*.

A *análise intervalar*, também conhecida por *aritmética intervalar,* começou sendo utilizada para controlar os erros de arredondamento provenientes das operações realizadas em um computador. O seu desenvolvimento moderno iniciou-se a partir da tese de doutorado de Moore [10], em 1962.

A aritmética da análise intervalar opera com intervalos, em vez de números reais. Cada número real x é representado por um par de números em ponto flutuante, $\underline{x}$ e $\overline{x}$, representando um intervalo $X = [\underline{x}, \overline{x}]$ de números reais, tal que $\underline{x} \leq x \leq \overline{x}$. Desse modo, temos não só uma estimativa para o valor de x, dada pelo centro do intervalo, mas também uma medida da qualidade dessa estimativa, dada pelo tamanho do intervalo. Os intervalos são somados, subtraídos, multiplicados, divididos, etc., de tal modo que cada intervalo computado X contenha o valor do número real x correspondente.

Os números a e b de um intervalo $[a, b]$ de números reais podem não ser representados em um dado computador. Nesse caso, arredonda-se a para o maior número em ponto flutuante menor ou igual a a e arredonda-se b para o menor número em ponto flutuante maior ou igual a b. Dessa forma, o intervalo obtido ainda contém $[a, b]$. Esse procedimento é chamado de *arredondamento externo.* Todas as máquinas que suportam o padrão de ponto flutuante IEEE [13], como a maioria dos PC's e as estações de trabalho, permitem o arredondamento externo.

3.1. Definições e operações básicas

Sejam $X = [\underline{x}, \overline{x}]$ e $Y = [\underline{y}, \overline{y}]$ dois intervalos quaisquer. Para qualquer operação binária $\circ$ ($+, -, *$ ou $/$) entre números reais, define-se:

$$X \circ Y = [\min_{\substack{x \in X \\ y \in Y}} (x \circ y), \max_{\substack{x \in X \\ y \in Y}} (x \circ y)].$$

Usando a definição acima, obtém-se:

$$\begin{aligned}
X+Y &= [\underline{x}+\underline{y}, \overline{x}+\overline{y}] \\
X-Y &= [\underline{x}-\overline{y}, \overline{x}-\underline{y}] \\
X*Y &= [a,b] \text{ (onde } a=\min\{\underline{x}\underline{y}, \overline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\overline{y}\} \text{ e } b=\max\{\underline{x}\underline{y}, \overline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\overline{y}\}) \\
\frac{1}{Y} &= \left[\frac{1}{\overline{y}}, \frac{1}{\underline{y}}\right] \text{ (se } 0 \notin Y) \\
\frac{X}{Y} &= X * \frac{1}{Y} \text{ (se } 0 \notin Y).
\end{aligned}$$

O *centro* c de X é definido por $c(X) = \frac{\underline{x}+\overline{x}}{2}$ e o *comprimento* w de X é definido por $w(X) = \overline{x} - \underline{x}$.

Um intervalo degenerado $[x, x]$ será representado simplesmente por x. Para maiores detalhes sobre essas definições, ver [4].

Nas regras dadas acima, excluímos a possibilidade de divisão por um intervalo contendo zero. A *aritmética intervalar estendida* considera esse caso [4].

3.2. Funções intervalares

Uma *função intervalar* é uma função que retorna um intervalo tendo um ou mais intervalos como argumentos. Considere uma função real $f(x_1, ..., x_n)$ com n variáveis reais e uma função intervalar $F(X_1, ..., X_n)$ com n variáveis intervalares. A função intervalar F é uma *extensão intervalar* de f, se

$$F(x_1, ..., x_n) = f(x_1, ..., x_n) \text{ para todo } x_i \in X_i,\ i = 1, ..., n.$$

Ou seja, se os argumentos de F são intervalos degenerados, então $F(X_1, ..., X_n)$ é um intervalo degenerado igual a $f(x_1, ..., x_n)$.

Uma função intervalar $F(X_1, ..., X_n)$ é *monótona inclusiva,* se

$$X_i \subset Y_i,\ i = 1, ..., n \Rightarrow F(X_1, ..., X_n) \subset F(Y_1, ..., Y_n).$$

O teorema abaixo é o resultado mais importante da análise intervalar e é conhecido como *Teorema Fundamental da Análise Intervalar* [4]. Uma de suas importantes conseqüências é que ele permite a obtenção de limites inferiores para serem usados em um método BB.

Teorema 3.1. *Se $F(X_1, ..., X_n)$ é uma extensão intervalar monótona inclusiva de uma função real $f(x_1, ..., x_n)$, então $f(x_1, ..., x_n) \in F(X_1, ..., X_n)$ para todo $x_i \in X_i$, $i = 1, ..., n$.*

Demonstração. Ver [4], p. 16. □

O intervalo obtido quando avaliamos uma função intervalar depende da forma como a função é representada. Por exemplo, embora

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

sejam extensões intervalares para

$$f(x) = x^2 - x \quad (x \in \mathbb{R}),$$

F_1 e F_2 podem não produzir o mesmo resultado quando avaliadas:

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

O resultado gerado por F_2 é o valor exato da imagem de f sobre $[0,2]$.

Em geral, quando uma dada variável ocorre mais de uma vez em um cálculo intervalar, ela é considerada uma variável distinta em cada ocorrência, o que dificulta a obtenção de intervalos mais estreitos. Esse problema é conhecido como *problema da dependência* [12].

4. O Algoritmo

Apresentaremos agora um algoritmo de otimização global baseado em um método BB que utiliza técnicas de análise intervalar. Esse algoritmo é baseado no algoritmo de Hansen [4].

O problema é

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

onde $f : \mathbb{R}^n \to \mathbb{R}$ é uma função de classe C^2 e X é uma caixa no $\mathbb{R}^n$, ou seja, $X = X_1 \times ... \times X_n \subset \mathbb{R}^n$, onde $X_i = [\underline{x_i}, \overline{x_i}]$ $(i = 1, ..., n)$. Sejam f^* o menor valor de f em X e x^* um ponto onde esse valor é atingido, ou seja, $f^* = f(x^*)$.

O centro da caixa X é definido por $c(X) = (c(X_1), ..., c(X_n))$ e o comprimento de X é definido por $w(X) = \max\{w(X_i) : i = 1, ..., n\}$.

Para aplicar o algoritmo, não é necessário que a função f seja de classe C^2. Entretanto, com essa hipótese, podemos utilizar as informações provenientes do gradiente e da Hessiana de f para eliminar regiões que não contenham o mínimo global.

Se x^* está no interior de X, então $g(x^*) = 0$, onde g é o gradiente de f. Consideremos uma subcaixa B de X. Se existir algum $i = 1, ..., n$ tal que $0 \notin g_i(B)$, então g não se anula em nenhum ponto de B. Portanto, podemos descartar a caixa B.

Novamente, se x^* está no interior de X, então a Hessiana H de f é semidefinida positiva em x^*. Uma condição necessária para isso é que, para $i = 1, ..., n$, $H_{ii}(x^*) \geq 0$, onde $H_{ii}(x^*)$ são os elementos da diagonal de $H(x^*)$. Consideremos, então, uma subcaixa B de X. Se existir algum $i = 1, ..., n$ tal que $H_{ii}(B) < 0$, então $H_{ii}(x) < 0$ para todo $x \in B$. Portanto, H não pode ser semidefinida positiva em nenhum ponto de B. Por esse motivo, podemos descartar a caixa B.

Com o decorrer do algoritmo, a caixa X será dividida em várias outras. A divisão de caixas pode ser feita simplesmente particionando a caixa em duas, escolhendo a variável intervalar X_i de maior tamanho. Em [1], propõe-se uma divisão diferente.

As novas caixas geradas são colocadas em duas listas. A primeira lista L_1 é formada por caixas B, que não satisfazem a pelo menos um dos critérios abaixo:

$$\begin{aligned} w(B) &\leq \varepsilon_X \\ w(f(B)) &\leq \varepsilon_f, \end{aligned}$$

onde ε_X e ε_f são as tolerâncias dadas para a dimensão das caixas resultantes no final do algoritmo e para o tamanho do intervalo que contém f^*, respectivamente.

A segunda lista L_2 é formada por caixas que satisfazem a esses dois critérios.

Seja $[\underline{f}(B), \overline{f}(B)]$ o intervalo obtido quando se avalia $f(B)$ (note que os valores $\underline{f}(B)$ e $\overline{f}(B)$ são apenas limitantes para a imagem de f sobre B). A caixa B escolhida de L_1 para ser processada é aquela com o menor valor de $\underline{f}(B)$. Se B é pequena, pode-se obter um bom limite superior para f^*. Por outro lado, se B é grande, $\underline{f}(B)$ tende a ser bem menor que o menor valor de f em B. Nesse caso, seleciona-se uma caixa que ainda foi pouco explorada e que deve ser reduzida para se obter melhores informações sobre f.

Se um limite superior u de f^* é conhecido, fazemos $\overline{f} = u$. Caso contrário, $\overline{f} = \infty$. Se é conhecida uma aproximação $\overline{x}$ de x^*, avaliamos f nesse ponto e definimos $\overline{f}$ como o menor valor entre u e $f(\overline{x})$.

A seguir, descrevemos os passos do algoritmo.

Inicialização: Particione X em duas subcaixas e coloque-as na lista L_1.

1: Se a lista L_1 está vazia, vá para **6**.
Remova uma caixa B da lista L_1 tal que $\underline{f}(B) = \min_{Y \in L_1} \underline{f}(Y)$.
2: Se existe algum i tal que $0 \notin g_i(B)$, descarte B e vá para **1**.
3: Se existe algum i tal que $H_{ii}(B) < 0$, descarte B e vá para **1**.
4: Avalie f no centro de B e atualize $\overline{f}$.
Remova qualquer caixa Y da lista L_1 com $\underline{f}(Y) > \overline{f}$.
5: Se $w(B) \leq \varepsilon_X$ e $w(f(B)) \leq \varepsilon_f$, coloque B na lista L_2 e vá para **1**.
Particione B em duas subcaixas, coloque-as na lista L_1 e vá para **1**.
6: Remova qualquer caixa Y da lista L_2 com $\underline{f}(Y) > \overline{f}$ e saia com a lista L_2.

Sejam $C_1, ..., C_p$ as caixas restantes em L_2. Calcule $\underline{f} = \min_{1 \leq i \leq p} \underline{f}(C_i)$. Temos então que

$$\underline{f} \leq f^* \leq \overline{f}$$

e

$$w(C_i) \leq \varepsilon_X$$

para toda caixa $C_i \in L_2$ (ver [4], p. 127). A lista L_2 contém os mínimos globais do problema.

5. Resultados Computacionais

Os testes computacionais foram realizados em um Pentium III 700 MHz com 256 MB de memória RAM. O código, em Fortran 90, é uma adaptação do código de R. Kearfott [7].

Para testarmos computacionalmente o algoritmo, seria interessante usarmos uma função onde o mínimo global fosse conhecido. Para isso, usaremos uma função que apresenta duas características principais: possui um único mínimo global conhecido e a quantidade de mínimos locais cresce de forma exponencial com o tamanho do problema.

A função teste f tem a seguinte expressão:

$$f = \sum_{i=1}^{n} \left(1 + \cos(3x_i) + \frac{(-1)^i}{\sqrt{10.60099896 - 4.141720682(\cos x_i)}} \right),$$

onde $x_i \in [0, 5]$. Em [8], demonstra-se que a quantidade de mínimos locais dessa função é 2^n e que existe apenas um minimizador global, dado pela seqüência

$$a, b, a, b, a, b, ...,$$

onde $a = 1,039195303$ e $b = 3,141592654$.

Com 25 variáveis (o maior problema em que aplicamos o algoritmo), a quantidade de mínimos locais é:

$$\left(2^{25}\right) > 30 \times 10^6.$$

Ou seja, mais de 30 milhões de mínimos locais!

A seguir, damos o valor do mínimo global para $n = 5, ..., 25$ ($a = 1,039195303$ e $b = 3,141592654$):

$$\begin{aligned}
f(a,b,a,b,a) &= -0,50715193 \\
f(a,b,a,b,a,b) &= -0,24670982 \\
f(a,b,a,b,a,b,a) &= -0,58938853 \\
f(a,b,a,b,a,b,a,b) &= -0,32894643 \\
f(a,b,a,b,a,b,a,b,a) &= -0,67162514 \\
f(a,b,a,b,a,b,a,b,a,b) &= -0,41118303 \\
f(a,b,a,b,a,b,a,b,a,b,a) &= -0,75386174 \\
f(a,b,a,b,a,b,a,b,a,b,a,b) &= -0,49341964 \\
f(a,b,a,b,a,b,a,b,a,b,a,b,a) &= -0,83609836 \\
f(a,b,a,b,a,b,a,b,a,b,a,b,a,b) &= -0,57565625 \\
f(a,b,a,b,a,b,a,b,a,b,a,b,a,b,a) &= -0,91833496 \\
f(a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b) &= -0,65789287
\end{aligned}$$

$$f(a,b,a,b,a,b,a,b,a,b,a,b,a,b,a) = -1,00057157$$
$$f(a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b) = -0,74012947$$
$$f(a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a) = -1,08280818$$
$$f(a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b) = -0,82236608$$
$$f(a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a) = -1,16504479$$
$$f(a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b) = -0,90460269$$
$$f(a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a) = -1,24728141$$
$$f(a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b) = -0,98683929$$
$$f(a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a,b,a) = -1,32951801.$$

5.1. Resultados numéricos

Na tabela 1, apresentamos os resultados numéricos obtidos aplicando o algoritmo em problemas com $n = 5, ..., 25$ ($\varepsilon_X = \varepsilon_f = 10^{-4}$). A coluna **N** apresenta o número de variáveis, a coluna **f** apresenta o número de avaliações da função, a coluna ∇**f** apresenta o número de avaliações do gradiente da função, a coluna ∇^2**f** apresenta o número de avaliações da Hessiana da função, a coluna **NÓS** apresenta o número de caixas processadas pelo algoritmo, a coluna $\mathbf{f}^*$ apresenta o menor valor da função obtido, a coluna **CPU**(**s**) apresenta o tempo em segundos gasto para a obtenção do mínimo global e a coluna **CPU**(**h**) apresenta o tempo em horas gasto para a obtenção do mínimo global.

N	**f**	∇**f**	∇^2**f**	**NÓS**	$\mathbf{f}^*$	**CPU**(**s**)	**CPU**(**h**)
5	90	42	18	17	-0,5072	0,83	0,0002
6	113	58	24	18	-0,2467	1,44	0,0004
7	168	63	30	19	-0,5894	2,47	0,0007
8	215	96	37	21	-0,3289	4,32	0,0012
9	350	114	46	24	-0,6716	6,70	0,0019
10	497	160	57	32	-0,4112	12,24	0,0034
11	896	298	68	61	-0,7539	22,47	0,0062
12	1154	380	105	96	-0,4934	34,56	0,0096
13	1600	474	185	142	-0,8361	50,53	0,0140
14	2964	1354	294	254	-0,5757	97,20	0,0270
15	5333	2073	423	536	-0,9183	218,4	0,0607
16	10852	4258	734	1105	-0,6579	448,2	0,1245
17	21949	9485	2509	3377	-1,0006	1167	0,3242
18	45789	15485	5421	7128	-0,7401	3075	0,8542
19	100026	36053	10078	12563	-1,0828	6365	1,768

20	174857	59125	16585	19587	-0,8224	10126	2,813
21	253358	91908	25441	33060	-1,1650	19480	5,411
22	385745	148522	41525	56123	-0,9046	34657	9,627
23	662161	250506	67139	92950	-1,2473	62730	17,425
24	1019868	425781	107285	154212	-0,9868	123232	34,231
25	1779637	704548	192735	258393	-1,3295	297100	82,528

Tabela 1: Resultados com $n = 5, ..., 25$ ($\varepsilon_X = \varepsilon_f = 10^{-4}$)

É importante destacar que, neste trabalho, não estamos considerando a eficiência do método utilizado. Qualquer algoritmo baseado em um método *branch and bound* tem, no pior caso, complexidade exponencial. Isso ficou evidente nos resultados computacionais. Em contrapartida, pode-se garantir que o mínimo global estará em um dado intervalo, tão pequeno quanto se queira, algo que nenhum método estocástico ou heurístico pode fazer. Isso explica porque, na tabela acima, os valores da coluna $\mathbf{f}^*$ estão representados com apenas 4 casas decimais. Na verdade, esse valor é o ponto médio do intervalo obtido pelo algoritmo, com comprimento dado por $\varepsilon_f = 10^{-4}$. Para obter resultados com uma precisão maior, basta diminuir o valor de ε_f.

6. Considerações Finais

A maioria dos métodos computacionais de otimização global é de natureza estocástica ou heurística. Portanto, não garantem que a solução encontrada seja o mínimo global.

Empregamos um algoritmo determinístico baseado em um método *branch and bound,* que utiliza a aritmética intervalar para o cálculo dos limites inferiores. Este algoritmo produz um intervalo, tão pequeno quanto se queira, que contém o mínimo global. Para testar o algoritmo, usamos uma função onde o mínimo global é conhecido e a quantidade de mínimos locais cresce exponencialmente com o aumento do número de variáveis. Problemas com até 25 variáveis foram considerados. Em todos eles, o mínimo global foi encontrado.

Como prosseguimento deste trabalho, além da paralelização do método *branch and bound,* pretendemos desenvolver um método híbrido que incorpore as características dos métodos estocásticos, a fim de reduzir o tempo computacional. Nesse sentido, os resultados apresentados neste artigo tornam-se um passo importante para atingir esse objetivo.

Abstract. We use a deterministic algorithm for global optimization problems. The algorithm is based on a *branch and bound* method that uses the interval arithmetics for computing lower bounds. The algorithm was applied to a scalable test function with known global minimum and an exponentially growing number of local minima. Problems with up to 25 variables were successfully solved.

Agradecimentos. Gostaríamos de agradecer as valiosas sugestões dos revisores, que ajudaram a melhorar a qualidade desse artigo. Agradecemos também o suporte financeiro da FAPERJ e do CNPq.

Referências

[1] A.E. Csallner, T. Csendes & M.C. Markót, Multisection in interval branch and bound methods for global optimization I. Theoretical results, *J. Global Optim.*, **16** (2000), 371-392.

[2] G. Nemhauser & L. Wolsey, "Integer and Combinatorial Optimization", Wiley, New York, 1988.

[3] D. Goldberg, What every computer scientist should know about floating-point arithmetic, *ACM Computing Surveys*, **23** (1991), 5-48.

[4] E.R. Hansen, "Global Optimization using Interval Analysis", Springer-Verlag, Berlin, 1993.

[5] R. Horst, On the convexification of nonconvex programming problems, *European Journal of Operational Research*, **15** (1984), 382-392.

[6] R. Horst & H. Tuy, On the convergence of global methods in multiextremal optimization, *Journal of Optimization Theory and Applications*, **54** (1987), 253-271.

[7] R.B. Kearfott, A Fortran 90 environment for research and prototyping of enclosure algorithms for nonlinear equations and global optimization, *ACM Trans. Math. Software*, **21** (1995), 63-78.

[8] C. Lavor & N. Maculan, A function to test methods applied to global minimization of potential energy of molecules, *Numerical Algorithms* (aceito para publicação).

[9] J.M. Martinez & S.A. Santos, Métodos Computacionais de Otimização, em "20º Colóquio Brasileiro de Matemática", IMPA, Rio de Janeiro, 1995.

[10] R.E. Moore,"Interval arithmetic and automatic error analysis in digital computation", Ph.D. Dissertation, Stanford University (1962).

[11] J. Pintér, Branch and bound algorithms for solving global optimization problems with lipschitzian structure, *Optimization*, **19** (1988), 101-110.

[12] J.G. Rokne, Low complexity k-dimensional centered forms, *Computing*, **37** (1986), 247-253.

[13] D. Stevenson, *IEEE standard for binary floating point arithmetic* (IEEE/ANSI 754-1985), Technical Report, IEEE, 1985.