ALGORITMO HEURÍSTICO DE BUSCA DIRETA PARA SOLUÇÃO DE PROBLEMAS DE PROGRAMAÇÃO NÃO LINEAR IRRESTRITA COM MÚLTIPLOS ÓTIMOS.

Maurício Rodrigues Silva
Universidade Federal de Ouro Preto – Departamento Engenharia de Produção
UFOP - DEENP
João Monlevade, MG – Brasil.
dscmauricio@gmail.com

RESUMO

O algoritmo apresentado neste trabalho tem como objetivo a solução de problemas de programação não linear irrestrita com múltiplos ótimos, ou sistemas de equações lineares e não lineares com múltiplas raízes. Para isso, o modelo tem como base, o algoritmo de Luus Jaakola (1973), executado *n* vezes consecutivas através de uma estrutura de repetição, forçando a busca de todas as soluções possíveis dentro do intervalo de busca. Em seguida, todas as soluções geradas são filtradas e pós-otimizadas, resultando em um conjunto de valores distintos, equivalentes aos múltiplos ótimos da função objetivo do problema, ou conjunto de raízes do sistema.

PALAVRAS CHAVE: Otimização, Algoritmos heurísticos, busca direta, pesquisa operacional. Metaheuristicas (MH), Programação Matemática (PM).

ABSTRACT

The algorithm presented in this paper aims to solve unrestricted nonlinear programming problems with multiple optimum, or systems of linear and nonlinear equations with multiple roots. For this, the model has as base, the algorithm of Luus Jaakola (1973), executed *n* times in succession through a structure of repetition, forcing the search of all possible solutions within the range of search. Then all the solutions generated are filtered and post-optimized, resulting in a set of distinct values, equivalent to roots of the objective function of the problem, or set of roots of the system

KEYWORDS: Optimization, heuristic algorithms, direct search, operational research

1. Introdução

Existem atualmente diversos métodos para solução de sistemas de equações lineares e não lineares para otimização de problemas de programação linear e não linear, sejam métodos estocásticos ou determinísticos. Porém à medida que os problemas se tornam mais complexos, as soluções obtidas, se tornam também mais difíceis. Historicamente, as técnicas mais utilizadas para a solução de sistemas não-lineares são os chamados *métodos de Newton* (Rheinboldt, 1986). Kearfott (1987a,b) apresentaram uma técnica denominada método de Newton Intervalar / Bisseção Generalizada (IN/GB) para solução de sistemas de equações não-lineares. Kearfott e Novoa (1990) e Kearfott (1990) ressaltam a capacidade desta metodologia em garantir que todas as raízes (dentro de uma certa tolerância) foram identificadas ou que não há raízes dentro do intervalo considerado. Porém esta metodologia (IN/GB) tem um custo computacional alto, sendo que em alguns casos, a solução do problema não é encontrada.

Desta forma, métodos heurísticos são alternativas mais fáceis de implementar, e com resultados tão próximos da solução exata, que são aplicados cada vez mais. Assim sendo, o presente trabalho tem como objetivo a solução de problemas gerais que sejam caracterizados por modelos não-lineares sem restrições (ou com restrições do tipo "caixa") e possivelmente multimodais. A base deste trabalho é o algoritmo de busca direta aleatória, conhecido como método de Luus-Jaakola (1973), resultando em um novo algoritmo, que em relação aos métodos citados, tem como diferencial, a solução de problemas multimodais. Pois na engenharia, os problemas envolvem freqüentemente sistemas de equações não-lineares, eventualmente com múltiplas soluções. Para que estes sistemas sejam convenientemente resolvidos, será aplicada uma técnica de transformação da estrutura destes sistemas não-lineares em problemas de otimização. Este tipo de problema de otimização pode, alternativamente, representar um sistema de equações algébricas não-lineares (1) com soluções múltiplas.

Um sistema de equações não-lineares é escrito como:

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix} = 0 \qquad (1)$$

onde $x = (x_1, x_2, ...x_n)$, e $f_1, f_2, ..., f_n$ são funções não-lineares contínuas no domínio

$$\Omega = \prod_{i=1}^{n} [a_1, b_1] \subset \Re^n \qquad (2)$$

Apesar do sistema ser não-linear, algumas equações podem ser lineares, porém não todas (neste caso, o sistema seria linear e as técnicas de solução são bastante difundidas). Encontrar uma solução para um sistema não-linear envolve encontrar uma solução em que todas as equações do sistema não-linear sejam nulas, isto é:

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases} \qquad (3)$$

O sistema (3) pode ser transformado em um problema de minimização não-linear irrestrita. Para isso, cada equação é elevada ao quadrado e convertida em uma parcela de uma soma, compondo uma função-objetivo não-negativa, na forma:

$$F = \sum_{i=1}^{n} [f_i(x)]^2 \qquad (4)$$

O sistema é então transformado em um problema de minimização (5), onde os mínimos globais nulos da função-objetivo (4) representam as soluções do sistema original. Para completar a formulação do problema, os limites dos intervalos de cada variável são considerados. Mínimos não-nulos (locais) não correspondem, naturalmente, a soluções do problema original. O problema assume a seguinte forma:

$$\text{Minimizar } F \text{ , sobre } \Omega \qquad (5)$$

1.1 O Algoritmo Luus Jaakola

A seguir o algoritmo de Luus jaakola (1973) padrão é descrito, o qual é utilizado como núcleo do modelo apresentado neste trabalho. O procedimento que o algoritmo desenvolve se divide em praticamente duas etapas. Na primeira etapa, o algoritmo gera um vetor de números aleatórios para ajustes dos valores iniciais x^T, que dentro de um número de iterações, resultará num conjunto de soluções. Estas soluções estarão sujeitas às restrições do problema, gerando um grupo de soluções viáveis. Nesta etapa, a melhor solução para o problema, será a escolhida.

Na segunda etapa, o espaço de busca é contraído de um fator de redução, e a primeira etapa é novamente executada. Por fim, depois de um número fixo de iterações externas, o resultado obtido será a melhor solução para o problema. Observe que este algoritmo finaliza com apenas uma solução. Em problemas com múltiplas raízes, o algoritmo deve ser executado diversas vezes para observar se o problema converge para mais de uma solução, isto é, para verificar quantas soluções, ou ótimos o problema possui. A Fig. 1 descreve o algoritmo original de Luus jaakola (1973).

Escolha um tamanho inicial de busca $\mathbf{r}(0)$.
Escolha um número externo de iterações nout e um número interno nin.
Escolha um coeficiente de contração ε
Gere uma solução incial $\mathbf{x}^*$.
Para i = 1 até nout
Para j = 1 até nin
$\mathbf{x}(j) = \mathbf{x}^* + R(j)\mathbf{r}(i-1)$ *, onde* $R(j)$ *é um vetor de números aleatórios entre -0.5 e 0.5.*
Se $(\mathbf{x}(j)) < (\mathbf{x}^*)$
$\mathbf{x}^* = \mathbf{x}(j)$
Fim Se
Fim Para
$\mathbf{r}(i) = (1-\varepsilon)\mathbf{r}(i-1)$
Fim Para

Figura 1 - Algoritmo de Luus Jaakola

Para atender casos de problemas não lineares irrestritos com múltiplas soluções, foi sugerido então uma modificação para que este algoritmo continuasse localizando todas as raízes possíveis do problema em questão. Na Fig. 2 o algoritmo modificado é descrito.

1.2 O Algoritmo Modificado

O funcionamento deste algoritmo é bem simples, pois se baseia no Algoritmo Luus Jaakola (1973) original. O diferencial está no fato do algoritmo modificado ter a característica de localizar múltiplos ótimos, ou múltiplas raízes em problemas não lineares irrestritos. Inicialmente, um número de execuções é estabelecido através de um laço externo. Internamente a este laço, o algoritmo Luus Jaakola é executado até atender o número de execuções. A cada execução uma solução é gerada e armazenada em um vetor, contendo os valores das variáveis do problema.

Após o término das execuções, o vetor solução contém um número de soluções equivalente ao número de execuções. Estas soluções formar um conjunto de todas as soluções alcançadas pelo algoritmo de Luus Jaakola dentro dos limites do problema. Através da característica aleatória deste algoritmo, as soluções alternam entre todas as soluções permitidas pelo problema. Para que o algoritmo localize todas, ou o número máximo de soluções distintas possíveis, um número de execuções é imposto ao modelo. Ao final das execuções, o vetor, ou conjunto de todas as soluções possíveis, apresenta valores repetidos distintos.

Estes valores necessitam de uma seleção a fim de apresentar somente as soluções distintas sem repetição. Neste momento, o algoritmo aciona um filtro para eliminar as soluções repetidas, resultando num conjunto final contendo somente valores distintos. Este conjunto final é o conjunto de soluções do problema de múltiplos ótimos, ou raízes que o algoritmo alcançou. A seguir alguns testes serão executadas a fim de ilustrar o algoritmo e seu desempenho.

Defina um numero de iterações n
Defina R = Vetor raízes
Defina S = Vetor Solução
Para k =1 até n faça
    Execute a rotina Luus Jaakola
    Faça $r_k = x_k$ *(*$r_k \in R$*), onde* x_k *é a solução corrente da rotina Luus Jaakola*
Fim para
Elimine as raízes repetidas do vetor R
Copie as raízes distintas de R em S
Imprima o vetor solução S

Figura 2 - Algoritmo de Luus Jaakola Modificado

Uma característica essencial do algoritmo está na geração dos números pseudo-aleatórios que define a solução inicial. O algoritmo foi implementado em linguagem C, que utiliza uma rotina para definir um número pseudo-aleatório no intervalo de -0,5 e 0,5 através de uma função geradora de valores para este fim. A semente obtida é exibida no arquivo de saída junto com a solução, para sua reprodução a qualquer momento. Para isso, esta semente pode ser introduzida em um arquivo (.h) da função que contém o problema em teste, quando se desejar a reprodução de seus resultados.

2. Testes e Resultados

Para testar o modelo e verificar seu desempenho, alguns exemplos foram rodados, e seus resultados analisados numérica e graficamente, e por fim será comparado ao método IN/GB (Tabela 5).

2.1 Exemplo 1

Neste exemplo, procura-se a solução do sistema não-linear (6)

$$\begin{cases} x_1 - x_2 = 0 \\ {x_1}^2 + {x_2}^2 = 1 \end{cases} \tag{6}$$

Para isto, o sistema (6) foi convertido em um problema de otimização irrestrita, representado pela função-objetivo (7) (a ser minimizada, com mínimo nulo):

$$Minimizar\ f(x_1, x_2) = (x_1 - x_2)^2 + ({x_1}^2 + {x_2}^2 - 1)^2 \tag{7}$$

Graficamente, a solução do sistema não-linear corresponde à determinação das interseções entre a reta e a circunferência das equações (6). A Figura 3 ilustra as soluções deste problema simples, com solução analítica.

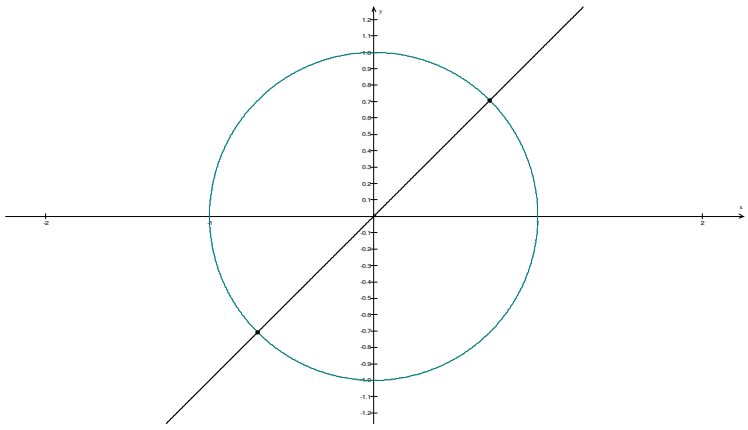

Figura 3 – Gráfico das funções do Exemplo 1

Analiticamente as raízes da Fig. 3 e Fig. 4 são:

Raiz 1: $x_1 =$ *0.70710678118661*, $x_2 =$ *0.70710678118648*
Raiz 2: $x_1 =$ *- 0.70710678118661*, $x_2 =$ *- 0.70710678118648*

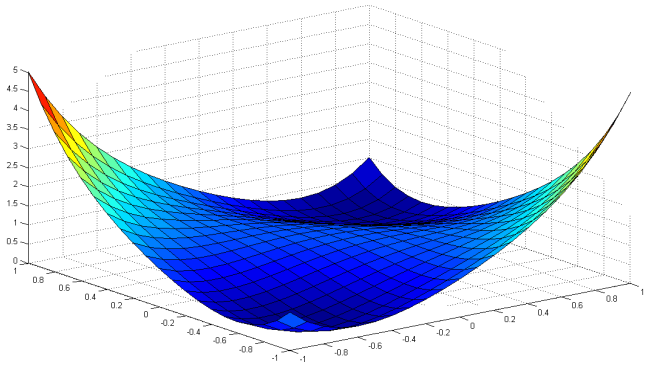

Figura 4 - Gráfico 3-D da Função Objetivo do Exemplo 1

Tabela 1 – Soluções do algoritmo Luus Jaakola para o Exemplo 1 com 100 iterações internas, 50 externas e 100 execuções. Tempo = 1 segundo.

Raízes:	x_1	x_2
1	0,7071068	0,7071068
2	- 0,7071068	- 0,7071068

2.2. Exemplo 2.

Neste exemplo, o problema a ser executado é encontrar as raízes da função objetivo, representado pelo seguinte sistema de equações:

$$\begin{cases} x_1^{\,3} - 3x_1^{\,2} - x_2 = -2 \\ (x_1 - 1)^2 + x_2^{\,2} = 4 \end{cases} \tag{8}$$

O sistema de equações (8) pode então ser convertido em uma função-objetivo (9) não-negativa, representada por:

$$f(x_1, x_2) = (x_1^{\,3} - 3x_1^{\,2} - x_2 + 2)^2 + ((x_1 - 1)^2 + x_2^{\,2} - 4)^2 \tag{9}$$

Então, busca-se o mínimo da função-objetivo representada por (9). A Figura 5 ilustra graficamente as seis raízes das equações (8).

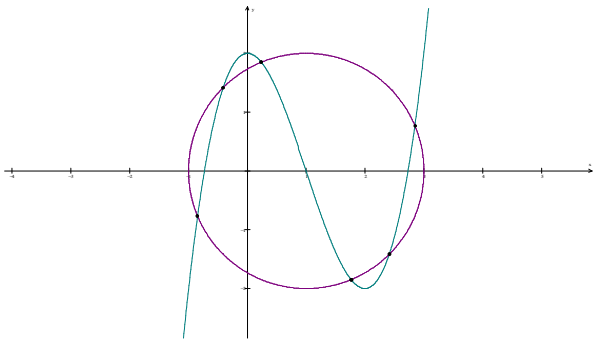

Figura 5 - Gráfico das funções do Exemplo 2

Valores exatos das raízes da Fig. 5 e Fig. 6:

Raiz 1: $x_1 =$ *0.23463313526969,* $x_2 =$ *1.84775906502252*
Raiz 2: $x_1 =$ *1.76536686473050,* $x_2 =$ *-1.84775906502244*
Raiz 3: $x_1 =$ *2.84775906502256,* $x_2 =$ *0.76536686473020*
Raiz 4: $x_1 =$ *-0.84775906502248,* $x_2 =$ *-0.76536686473041*
Raiz 5: $x_1 =$ *1.76536686473050,* $x_2 =$ *-1.84775906502244*
Raiz 6: $x_1 =$ *2.41421356237272,* $x_2 =$ *-1.41421356237347*

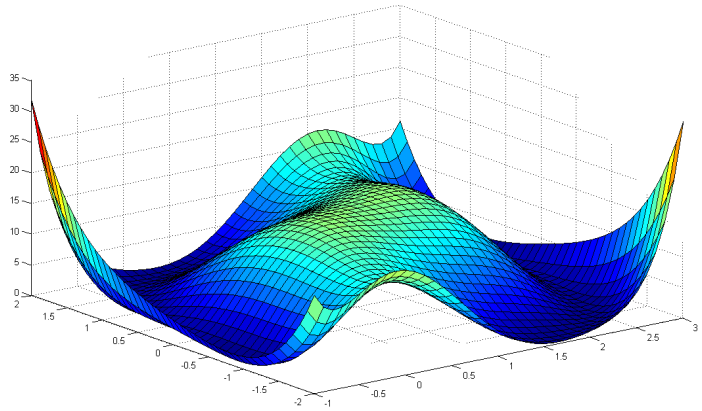

Figura 6 **-** Gráfico 3-D da Função Objetivo do Exemplo 2

Tabela 2 – Soluções do algoritmo Luus Jaakola para o Exemplo 2 com 100 iterações internas, 50 externas e 100 execuções. Tempo = 2 segundos.

Raízes:	x_1	x_2
1	2.4142137	-1.4142134
2	1.7653669	-1.8477590
3	0.2346331	1.8477590
4	2.8477590	0.7653669
5	-0.4142136	1.4142135
6	-0.8477591	-0.7653669

2.3. Exemplo 3.

Neste exemplo foi utilizado um problema (Bini e Mourrain, 2004), envolvendo três equações não-lineares e apresentando oito raízes. Este problema tem sua origem semelhante ao Exemplo 2, onde o sistema em questão é dado por:

$$\begin{cases} -x_2^{\,2}x_3^{\,2} - x_2^{\,2} + 24x_2x_3 - x_3^{\,2} - 13 = 0 \\ -x_1^{\,2}x_3^{\,2} - x_1^{\,2} + 24x_1x_3 - x_3^{\,2} - 13 = 0 \\ -x_1^{\,2}x_2^{\,2} - x_1^{\,2} + 24x_1x_2 - x_2^{\,2} - 13 = 0 \end{cases} \tag{10}$$

$$0 \le x_1, x_2, x_3 \le 20$$

A solução obtida por este problema é a seguinte (Bini e Mourrain, 2004):

Raízes:	$\mathbf{x}_T$
1	(0,7796, 0,7796, 0,7796)
2	(10,8580, 0,7796, 0,7796)
3	(4,6252, 0,3321, 4,6252)
4	(4,6252, 4,6252, 0,3321)
5	(0,7796, 10,8580, 0,7796)
6	(0,3321, 4,6252, 4,6252)
7	(0,7796, 0,7796, 10,8580)
8	(4,6252, 4,6252, 4,6252)

Tabela 3 – Soluções do algoritmo Luus Jaakola para o Exemplo 3 com 500 iterações internas, 200 externas e 1000 execuções. Tempo = 147 segundos.

Raízes:	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
1	0.7795482	10.8577042	0.7795482	0.0000000
2	10.8577042	0.7795482	0.7795482	0.0000000
3	0.7795480	0.7795480	0.7795480	0.0000000
4	0.7795480	0.7795480	10.8577032	0.0000000
5	0.3320762	4.6121612	4.6382346	0.0017992
6	4.5452123	0.3321903	4.7062817	0.0686255
7	4.6474214	4.6030293	0.3320819	0.0052153
8	4.6251817	4.6251817	4.6251817	0.0000000

A seguir as soluções do problema serão mostradas, variando-se alguns parâmetros, como número de iterações internas, externas e execuções. O objetivo deste teste é evidenciar o desempenho do algoritmo Luus Jaakola, pois neste exemplo, devido ao aumento de uma variável em relação aos exemplos anteriores, o grau de complexidade do problema aumenta para cada solução, requerendo mais iterações e execuções para alcançar todas as raízes, quando possível (Tabela 3 e Tabela 4).

Tabela 4 – Variações do número de execuções do algoritmo Luus Jaakola para o Exemplo 3 com 500 iterações internas, 200 externas, tempo e número de raízes encontradas.

Execuções	***Tempo (S)***	***N° de Raízes***
100	22	5
200	44	5
500	103	7
1000	147	8

O gráfico da Fig. 7 ilustra o crescimento do tempo de execução do algoritmo em função do número de execuções. Para isso o eixo vertical expressa o tempo de execução e o eixo horizontal o número de execuções.

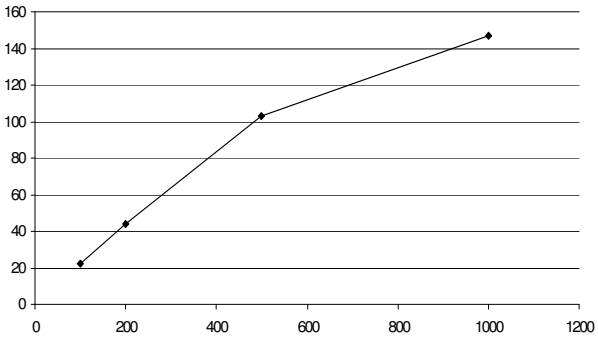

Figura 7 - Gráfico do crescimento do tempo de execução do Exemplo 3

Por fim, os resultados obtidos nos testes 2.1, 2.2 e 2.3 neste trabalho foram comparados com uma metodologia determinística, para determinação de todas as raízes de sistemas de equações não-lineares, baseada em análise intervalar. O algoritmo, representado usualmente pela sigla IN/GB (*Interval Newton/Generalized Bisection*), apresenta garantias matemáticas e computacionais de localização de todas as raízes de sistemas não-lineares, embora seu custo computacional limite sua aplicação para problemas com pequeno número de variáveis.

Tabela 5 - Comparação entre o método de Newton Intervalar/Bisseção Generalizada (IN/GB) e o Algoritmo Luus Jaakola Modificado.

Problemas testes			***IN/GB***		***Luus-Jaakola Modificado***	
Exemplo	***Nº de variáveis***	***Nº de raízes***	***Nº de raízes localizadas***	***Tempo de execução (s)***	***Nº de raízes localizadas***	***Tempo de execução (s)***
2.1	2	2	2	1,428	2	1,0
2.2	2	6	6	9,685	6	2,0
2.3	3	8	8	3759,703	8	147,0

3. Conclusões

O algoritmo Luus Jaakola (1973), é eficiente, e de fácil implementação devido a sua simplicidade. Porém, este algoritmo é aplicado na busca de apenas um ótimo por execução. Em situações de múltiplos ótimos, a cada execução, há uma convergência para apenas uma solução. Portanto, para que ele tenha um melhor desempenho, ou seja, alcance um número máximo de raízes, ou um conjunto de ótimos, um número de execuções é efetuado. Este diferencial faz deste modelo modificado, uma nova opção em métodos de localização de múltiplas raízes para problemas de otimização não lineares irrestritos. Um detalhe importante está no fato deste modelo modificado apresentar um conjunto de soluções, (Tabelas 1,2,3 e 4), desta forma em alguns casos, dentro deste conjunto, pode-se encontrar o ótimo global. Desta forma o presente trabalho contribui para novas aplicações, que agregado a novas técnicas pode ser integrado em outros modelos na criação de um modelo híbrido mais robusto.

4. Referências

Bini, D.A.; Mourrain, B. "Cyclo", from polynomial test suite. Disponível em: <http://wwwsop.inria.fr/saga/POL/ BASE/2.multipol/cyclohexan.html>. Acesso em: 2004.
Coelho, A. C. R.; Henderson N.; Sacco, W. F. (2007), "Comparison of the Luus-Jaakola algorithm and particle swarm optimization applied to a multimodal test function". *Anais do X Encontro de Modelagem Computacional, Nova Friburgo-Brasil.*
Floudas, C. A.; Pardalos, P. M. (1987), "*A collection of test problems for constrained global optimization algorithms Lecture. Notes in Computer Science"*, Springer-Verlag.
Goldbarg, M. C.; Luna, H. P. L. (2000), "*Otimização combinatória e programação linear: modelos e algoritmos"*, Editora Campus, Rio de Janeiro, Brasil.
Kearfott, R. B. Abstract generalized bisection and a cost bound. Math. Comput. v. 49, n.179, 1987a.
Kearfott, R. B. Some tests of generalized bisection. ACM Trans. Math. Softw., v. 13, n. 3, p. 197-220, 1987b.
Kearfott, R. B.; NOVOA, M. III. Algorithm 681: INTBIS, a portable interval-Newton/bisection package. ACM Transactions on Mathematical Software, v. 16, p. 152 – 157, 1990.
Kearfott, R. B. Preconditioners for the interval Gauss-Seidel method. SIAM J. Numer. Anal., v. 27, n. 3, p. 804-822, 1990.
Klepeis, J. L., Pieja, M. J.; Floudas, C. A. (2003), "Hybrid Global Optimization Algorithms for Protein Structure Prediction: Alternating Hybrids". *Biophysical Journal*.
Knupp, D. C.; Silva Neto, A. J.; Sacco W. F. (2007), "*Estimation of Radiative Properties with The Luus-Jaakola Method"*. Anais do X Encontro de Modelagem Computacional, Nova Friburgo-Brasil.
Lobato, F. S.; V. Steffen Jr. (2008), "*Luus-Jaakola algorithm applied to engineering systems design"*. School of Mechanical Engineering, Federal University of Uberlândia, Uberlândia – Brazil.
Luenberger, D. G. (1984), "*Linear and Nonlinear Programming"*, Second Edition, Standford University.
Luus, R.; Jaakola, T. (1973), "*Optimization by direct search and systematic reduction of the size of search region"*. AIChE Journal, vol. 19.
Rheinboldt, W.C. Methods for solving systems of nonlinear equations. 3rd. ed. Philadelphia: SIAM, 1986.