Determinação de minimizadores globais de funções por meio de métodos híbridos com base no método Luus Jaakola

Mateus Braga Oliveira
Universidade Federal Fluminense - Instituto do Noroeste Fluminense de Educação Superior
Avenida João Jasbick, s/nº - Bairro: Aeroporto - Santo Antônio de Pádua, RJ - CEP: 28470-000
mateusbr.oli@hotmail.com

Joviana Sartori de Souza
Universidade Federal Fluminense - Instituto do Noroeste Fluminense de Educação Superior
Avenida João Jasbick, s/nº - Bairro: Aeroporto - Santo Antônio de Pádua, RJ - CEP: 28470-000
joviana.sartori@gmail.com

RESUMO

Este trabalho tem como principal objetivo a determinação de minimizadores globais de funções e sistemas não lineares por meio de métodos híbridos tendo como base o método Luus Jaakola. Foram utilizados dois sistemas não lineares de dez variáveis e quatro funções não lineares de dez variáveis, com o objetivo de observar o comportamento do método a ser trabalhado para funções de maiores dimensões.Neste trabalho é tratado uma modificação para as hibridizações presentes em [Oliveira, 2016]. Dessa maneira, realiza-se uma troca no local da inserção do método da Busca Coordenada e do método Hooke Jeeves. Os resultados encontrados nos diferentes algoritmos são comparados, a fim de se determinar qual hibridização é mais eficiente.

PALAVRAS CHAVE. Luus Jaakola, Funções Não Lineares, Hibridizações.

PM - Programação Matemática.

ABSTRACT

This study has as main objective to determine global minimizers of nonlinear functions and systems through hybrid methods based on the Luus Jaakola method. We used two non-linear systems ten variables and four non-linear functions of ten variables for the purpose of observing the behavior of the method to be working for larger features. This work dealt with a modification to the hybridizations present in [Oliveira, 2016]. Thus, it carried out an exchange at the site of insertion of the method of coordinate search and Hooke Jeeves method. The results on different algorithms are compared in order to determine which hybridization is more efficient.

KEYWORDS. Luus Jaakola, Nonlinear Functions, Hybridizations.

MP - Mathematical Programming.

1. Introdução

Neste trabalho, serão comparados quatro algoritmos de otimização na busca de minimizadores globais de funções de várias variáveis, com o objetivo de verificar qual dos métodos abordados é mais eficiente na resolução do problema proposto. Os métodos utilizados são as hibridizações Luus Jaakola/Busca Coordenada e Luus Jaakola/Hooke Jeeves, apresentados em [Oliveira, 2016]. Os dois primeiros métodos consistem das hibridizações LJ/BC e LJ/HJ já abordadas na literatura. Os outros dois métodos testados, trazem uma modificação nestas hibridizações, alterando o local da inserção do método da Busca Coordenada e do método Hooke Jeeves no método Luus Jaakola. Esta modificação foi realizada visando analisar se existe alguma influência na escolha do local de inserção da Busca Coordenada e Hooke Jeeves.

Deste modo, o foco deste trabalho é apresentar quatro métodos híbridos de otimização para a determinação dos minimizadores globais para funções não lineares e para a determinação das soluções dos sistemas não lineares de várias variáveis. Com os resultados apresentados, será realizada uma comparação, verificando qual metodologia foi mais robusta na determinação dos pontos mínimos globais para os problemas testados. Assim, pode-se analisar a influência do método da Busca Coordenada e do método Hooke Jeeves, de acordo com sua inserção ao longo das iterações, no desempenho do método Luus Jaakola.

2. Métodos de Otimização

Temos que o problema de interesse exige a determinação dos minimizadores de funções e as soluções de sistemas não lineares. Essa tarefa será realizada por meio do método Luus Jaakola, em conjunto com os métodos da Busca Coordenada e de Hooke Jeeves. Serão apresentadas também as hibridizações mostradas em [Oliveira, 2016], e as modificações que serão mostradas neste trabalho.

2.1. O Método de Luus Jaakola

O algoritmo de Luus-Jaakola é um método de busca aleatória que utiliza somente informações da função a ser otimizada. Ele foi apresentado inicialmente pelos autores [Luus e Jaakola, 1973]. A ideia utilizada no algoritmo é a seguinte: a partir de uma ampla região de busca no domínio da função objetivo, geram-se soluções aleatórias enquanto a região de busca é reduzida de tamanho ao longo das iterações. Quando uma melhor solução é encontrada, o valor da função objetivo é calculado para esta solução e é comparado com uma tolerância desejada. Caso essa tolerância seja alcançada, o algoritmo é encerrado e esse valor encontrado é o ponto ótimo.

Algoritmo 1: - Luus Jaakola

Dados $\mathbf{x^0}$ - ponto inicial gerado aleatoriamente; r_a - raio inicial de busca; n_{in} - número máximo de iterações internas por loop externo; n_{out} - número máximo de iterações externas; ε_l - Fator de redução da região de busca $(0 < \varepsilon_l < 1)$; ϵ - tolerância para o critério de parada $(\epsilon > 0)$.
Para $i = 1, 2, ..., n_{out}$ **faça**
  Para $j = 1, 2, ..., n_{in}$ **faça**
    $x^{\mathbf{j}} = \mathbf{x^0} + \mathbf{R_j} \cdot r_a$, onde $\mathbf{R_j}$ é um vetor de números aleatórios entre $-0,5$ e $0,5$, que segue uma distribuição de probabilidade uniforme.
    Se $f(\mathbf{x^j}) < f(\mathbf{x^0})$ **então**
      $\mathbf{x^0} = \mathbf{x^j}$
    fim
    Se $f(\mathbf{x^0}) < \epsilon$ **então**
      Pare.
    fim
  fim
  $r_a = (1 - \varepsilon_l) \cdot r_a$
fim

Assim, o procedimento seguido pelo algoritmo, divide-se em duas partes:

1. Inicialmente, são estabelecidos os valores dos parâmetros do método;
2. Após a escolha dos parâmetros, tem-se dois loops. No loop externo é realizado a contração da região de busca. No loop interno são geradas as configurações aleatórias do método.

2.2. O Método da Busca Coordenada

O método da Busca Coordenada é um dos primeiros exemplos de um método de busca direta usado para resolver problemas de otimização. Os métodos de busca direta são conhecidos deste modo por não utilizarem os valores da função objetivo e suas derivadas em nenhum cálculo do algoritmo, exceto na verificação de qual de dois pontos tem o menor valor da função objetivo.

Por exemplo, para funções reais de duas variáveis, segue-se o procedimento a seguir: tentam-se pontos nas direções paralelas aos eixos coordenados, ou seja, pontos ao norte, sul, leste e oeste do ponto inicial.

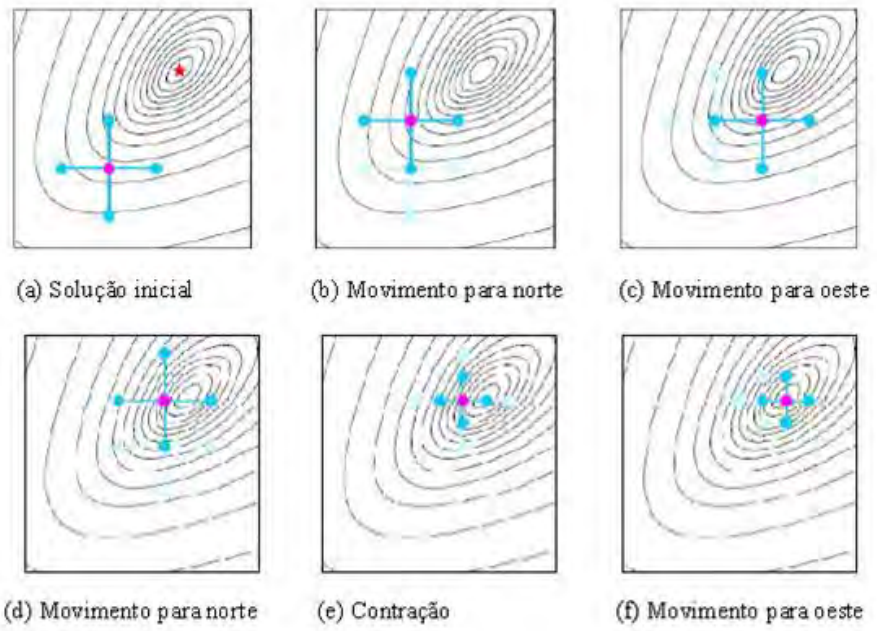

Figura 1: Ilustração do Algoritmo da Busca Coordenada.
Fonte: [Souza, 2010], p.85.

De [Diniz-Ehrhardt et al., 2010] temos um algoritmo para o Método da Busca Coordenada:

Algoritmo 2: - Busca Coordenada

Dados $\mathbf{x^1} \in \Omega \subseteq \mathbb{R}^n, a_1 > 0$
$m = n^2$, onde n é a dimensão do espaço $\mathbb{R}^n$.
Enquanto *(Critério de parada não for satisfeito)* **faça**
  Para $k = 1, 2, ..., m$ **faça**
    $\mathbf{x}^* = \mathbf{x^k} + a_k \cdot d$, onde d são as direções canônicas do espaço $\mathbb{R}^n$.
    Se $f(\mathbf{x}^*) < f(\mathbf{x^k})$ **então**
      $\mathbf{x^{k+1}} = \mathbf{x}^*$
      $a_{k+1} = a_k$
      Senão
        $\mathbf{x^{k+1}} = \mathbf{x^k}$
        $a_{k+1} = \frac{a_k}{2}$
      fim
    fim
  fim
  $k \leftarrow k + 1$
fim

2.3. O Método de Hooke Jeeves

O Algoritmo proposto por [Hooke e Jeeves, 1961] é um algoritmo determinístico de busca local. Este método promove dois tipos de busca: a exploratória e a padrão. A primeira etapa é a busca exploratória. Para isso, a partir de um ponto inicial, o método explora todas as direções de busca para cada variável, selecionando a melhor (onde a função objetivo tem seu valor diminuído). Esta etapa define um novo ponto com um valor melhor para função objetivo. Depois de explorar todas as direções de busca, o método executa a próxima etapa, a busca padrão, também conhecida como de progressão ou aceleração, avançando na direção definida na última iteração até um valor $\alpha > 0$ (fator de aceleração). A partir deste ponto repete-se a primeira etapa até alcançar o próximo ponto. A Figura 2 ilustra as duas etapas do método.

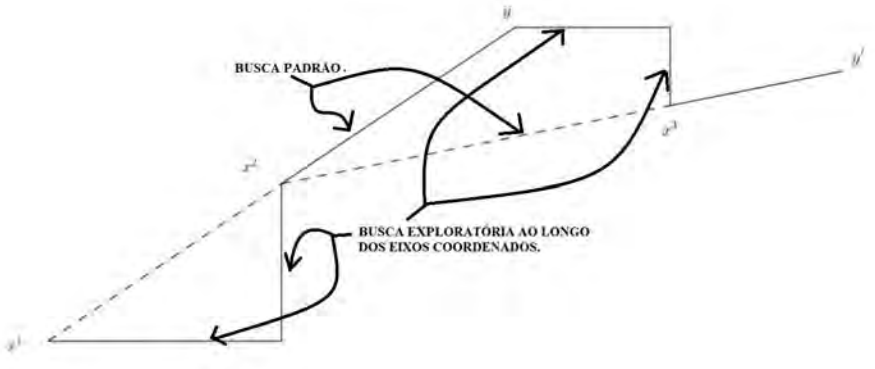

Figura 2: Ilustração dos passos do método de Hooke Jeeves.

Conforme a Figura 2, a partir do ponto inicial $\mathbf{x^1}$, o método executa uma busca exploratória em todas as direções, escolhendo a direção relacionada ao menor valor de $f(x_i)$, $i = 1, 2$, para cada eixo, chegando ao ponto $\mathbf{x^2}$. Neste ponto, o método executa uma busca ao longo da direção $(\mathbf{x^2} - \mathbf{x^1})$ - multiplicada de um fator de aceleração α, alcançando o ponto resultante $\mathbf{y}$. A partir deste ponto, o método repete os passos, alcançando $\mathbf{y}$', e assim sucessivamente até atender ao critério de parada.

A seguir, de [Bazaraa, 2013], tem-se um algoritmo para o método de Hooke Jeeves:

Algoritmo 3: - Hooke Jeeves

Passo de Inicialização:
Defina $d_1, ..., d_n$ como as direções coordenadas. Escolher um escalar $\epsilon > 0$ para determinar a parada do algoritmo. Escolher o tamanho do passo inicial $\Delta \geq \epsilon$, e o fator de aceleração $\alpha > 0$. Escolha o ponto de partida $\mathbf{x^1}$, faça $\mathbf{y^1} = \mathbf{x^1}$ e $k = 1$ e vá ao passo principal.
Passo Principal:
Enquanto $\Delta > \epsilon$ **faça**
  Para $i = 1, ..., n$ **faça**
    Se $f(\mathbf{y^i} + \Delta d_i) < f(\mathbf{y^i})$ **então**
      $\mathbf{y^{i+1}} = \mathbf{y^i} + \Delta d_i$;
    fim
    Senão se $f(\mathbf{y^i} - \Delta d_i) < f(\mathbf{y^i})$ **então**
      $\mathbf{y^{i+1}} = \mathbf{y^i} - \Delta d_i$;
    fim
    Senão
      $\mathbf{y^{i+1}} = \mathbf{y^i}$;
    fim
  fim
  Se $f(\mathbf{y^{n+1}}) < f(\mathbf{x^k})$ **então**
    $\mathbf{x^{k+1}} = \mathbf{y^{n+1}}$;
    $\mathbf{y^1} = \mathbf{x^{k+1}} + \alpha(\mathbf{x^{k+1}} - \mathbf{x^k})$;
    Senão
      $\Delta \leftarrow \frac{\Delta}{2}$;
      $\mathbf{x^{k+1}} \leftarrow \mathbf{x^k}$; $\mathbf{y^1} \leftarrow \mathbf{x^k}$;
    fim
  fim
  $k \leftarrow k + 1$;
fim

3. Métodos Híbridos de Otimização

A partir de agora, são apresentados as hibridizações testadas no presente trabalho.

3.1. Método 1 - Luus Jaakola/ Busca Coordenada

O algoritmo Luus Jaakola/Busca Coordenada 1 (LJ/BC - 1) é um método híbrido que adiciona o método da Busca Coordenada ao fim das iterações internas do algoritmo Luus Jaakola. Esse método funciona da seguinte forma: realizam-se os primeiros passos do Luus Jaakola. Quando uma solução é encontrada, ele atualiza este ponto e realiza uma busca coordenada ao seu redor, tentando melhorar este novo ponto, ou não. Com o término da busca coordenada, o método retorna para o Luus Jaakola, verifica se a solução encontrada satisfaz a tolerância desejada. Caso essa tolerância seja alcançada, o algoritmo é encerrado e esse valor de é o ponto ótimo.

A seguir, segue um pseudocódigo para o algoritmo híbrido de LJ/BC - 1:

Algoritmo 4: - Luus Jaakola/Busca Coordenada 1

Dados $\mathbf{x^0}$ - ponto inicial gerado aleatoriamente; r_a - raio inicial de busca; n_{in} - número máximo de iterações internas por loop externo; n_{out} - número máximo de iterações externas; ε_l - Fator de redução da região de busca $(0 < \varepsilon_l < 1)$; ϵ - tolerância para o critério de parada $(\epsilon > 0)$.
Para $i = 1, 2, ..., n_{out}$ **faça**
  Para $j = 1, 2, ..., n_{in}$ **faça**
    $\mathbf{x^j} = \mathbf{x^0} + \mathbf{R_j} \cdot r_a$, onde $\mathbf{R_j}$ é um vetor de números aleatórios entre $-0,5$ e $0,5$, que segue uma distribuição de probabilidade uniforme.
    Se $f(\mathbf{x^j}) < f(\mathbf{x^0})$ **então**
      $\mathbf{x^0} = \mathbf{x^j}$
    fim
    $\mathbf{x^0} = BuscaCoordenada(\mathbf{x^0})$ - Método Busca Coord. é realizado atualizando o ponto $\mathbf{x^0}$.
    Se $f(\mathbf{x^0}) < \epsilon$ **então**
      Pare.
    fim
  fim
  $r_a = (1 - \varepsilon_l) \cdot r_a$
fim

Neste primeiro método, a inserção da Busca Coordenada é feita após as iterações internas do método Luus Jaakola, como é possível perceber no pseudocódigo.

3.2. Método 2 - Luus Jaakola/ Busca Coordenada

O algoritmo Luus Jaakola/Busca Coordenada 2 (LJ/BC - 2) é um método híbrido que adiciona o método da Busca Coordenada ao início das iterações internas do algoritmo Luus Jaakola.

Este método híbrido funciona da seguinte forma: realiza-se a inicialização dos passos do Luus Jaakola. A solução inicial do método Luus Jaakola é utilizada para iniciar a Busca Coordenada, realiza-se uma busca coordenada ao seu redor, tentando melhorar este ponto, ou não. Com o término da busca coordenada, o método retorna para o Luus Jaakola, realizando as iterações internas, prosseguindo com o método Luus Jaakola. Após estas iterações, se uma solução é encontrada, encerra-se os procedimentos, encontrando a solução ótima. Caso contrário, os procedimentos são reiniciados.

Neste segundo método, a inserção da Busca Coordenada é feita após a inicialização do método Luus Jaakola, antecedendo as suas iterações internas, como é possível perceber no pseudocódigo que segue.

A seguir, encontra-se o pseudocódigo para o algoritmo híbrido de LJ/BC - 2:

Algoritmo 5: - Luus Jaakola/Busca Coordenada 2

Dados $\mathbf{x^0}$ - ponto inicial gerado aleatoriamente; r_a - raio inicial de busca; n_{in} - número máximo de iterações internas por loop externo; n_{out} - número máximo de iterações externas; ε_l - Fator de redução da região de busca $(0 < \varepsilon_l < 1)$; ϵ - tolerância para o critério de parada $(\epsilon > 0)$.
Para $i = 1, 2, ..., n_{out}$ **faça**
  $\mathbf{x^0} = BuscaCoordenada(\mathbf{x^0})$ - Método Busca Coord. é realizado atualizando o ponto $\mathbf{x^0}$.
  Para $j = 1, 2, ..., n_{in}$ **faça**
    $\mathbf{x^j} = \mathbf{x^0} + \mathbf{R_j} \cdot r_a$, onde $\mathbf{R_j}$ é um vetor de números aleatórios entre $-0,5$ e $0,5$, que segue uma distribuição de probabilidade uniforme.
    Se $f(\mathbf{x^j}) < f(\mathbf{x^0})$ **então**
      $\mathbf{x^0} = \mathbf{x^j}$
    fim
    Se $f(\mathbf{x^0}) < \epsilon$ **então**
      Pare.
    fim
  fim
  $r_a = (1 - \varepsilon_l) \cdot r_a$
fim

3.3. Método 3 - Luus Jaakola/Hooke Jeeves

O método Luus Jaakola/Hooke Jeeves 3 (LJ/HJ - 3) é um método híbrido que adiciona o método de Hooke Jeeves ao fim das iterações internas do algoritmo Luus Jaakola, funcionando da seguinte forma: realizam-se os primeiros passos do Luus Jaakola. Quando uma solução é encontrada, este ponto é atualizado e realiza-se os passos do método de Hooke Jeeves, tentando melhorar este novo ponto, ou não. Com o término do Hooke Jeeves, o método retorna para o Luus Jaakola, verificando se a solução encontrada satisfaz a tolerância desejada. Caso essa tolerância seja alcançada, o algoritmo é encerrado e esse valor de é o ponto ótimo.

A seguir, segue um pseudocódigo para o algoritmo híbrido de LJ/HJ - 3:

Algoritmo 6: - Luus Jaakola/Hooke Jeeves 3

Dados $\mathbf{x^0}$ - ponto inicial gerado aleatoriamente; r_a - raio inicial de busca; n_{in} - número máximo de iterações internas por loop externo; n_{out} - número máximo de iterações externas; ε_l - Fator de redução da região de busca $(0 < \varepsilon_l < 1)$; ϵ - tolerância para o critério de parada $(\epsilon > 0)$.
Para $i = 1, 2, ..., n_{out}$ **faça**
  Para $j = 1, 2, ..., n_{in}$ **faça**
    $\mathbf{x^j} = \mathbf{x^0} + \mathbf{R_j} \cdot r_a$, onde $\mathbf{R_j}$ é um vetor de números aleatórios entre $-0,5$ e $0,5$, que segue uma distribuição de probabilidade uniforme.
    Se $f(\mathbf{x^j}) < f(\mathbf{x^0})$ **então**
      $\mathbf{x^0} = \mathbf{x^j}$
    fim
    $\mathbf{x^0} = HookeJeeves(\mathbf{x^0})$ - Método Hooke Jeeves é realizado atualizando o ponto $\mathbf{x^0}$.
    Se $f(\mathbf{x^0}) < \epsilon$ **então**
      Pare.
    fim
  fim
  $r_a = (1 - \varepsilon_l) \cdot r_a$
fim

Neste primeiro método, a inserção do Hooke Jeeves é feita após as iterações internas do método Luus Jaakola, como é possível perceber no pseudocódigo.

3.4. Método 4 - Luus Jaakola/Hooke Jeeves

O algoritmo Luus Jaakola/Hooke Jeeves 4 (LJ/HJ - 4) é um método híbrido que adiciona o método de Hooke Jeeves ao início das iterações internas do algoritmo Luus Jaakola. O funcionamento deste método ocorre da seguinte forma: realiza-se a inicialização do método Luus Jaakola. A solução inicial do método Luus Jaakola é utilizada para iniciar o método Hooke Jeeves, realizando-se seus passos, tentando melhorar este ponto, ou não. Com o término do método Hooke Jeeves, o método retorna para o Luus Jaakola, realizando as iterações internas, prosseguindo com o método Luus Jaakola. Após estas iterações, se uma solução é encontrada, encerra-se os procedimentos, encontrando a solução ótima. Caso contrário, os procedimentos são reiniciados.

Neste segundo método, a inserção do Hooke Jeeves é feita após a inicialização do método Luus Jaakola, antecedendo as suas iterações internas, como é possível perceber no pseudocódigo que segue.

A seguir, encontra-se o pseudocódigo para o algoritmo híbrido de LJ/HJ - 4:

Algoritmo 7: - Luus Jaakola/Hooke Jeeves 4

Dados $\mathbf{x^0}$ - ponto inicial gerado aleatoriamente; r_a - raio inicial de busca; n_{in} - número máximo de iterações internas por loop externo; n_{out} - número máximo de iterações externas; ε_l - Fator de redução da região de busca $(0 < \varepsilon_l < 1)$; ϵ - tolerância para o critério de parada $(\epsilon > 0)$.
Para $i = 1, 2, ..., n_{out}$ **faça**
  $\mathbf{x^0} = HookeJeeves(\mathbf{x^0})$ - Método Hooke Jeeves é realizado atualizando o ponto $\mathbf{x^0}$.
  Para $j = 1, 2, ..., n_{in}$ **faça**
    $\mathbf{x^j} = \mathbf{x^0} + \mathbf{R_j} \cdot r_a$, onde $\mathbf{R_j}$ é um vetor de números aleatórios entre $-0,5$ e $0,5$. que segue uma distribuição de probabilidade uniforme.
    Se $f(\mathbf{x^j}) < f(\mathbf{x^0})$ **então**
      $\mathbf{x^0} = \mathbf{x^j}$
    fim
    Se $f(\mathbf{x^0}) < \epsilon$ **então**
      Pare.
    fim
  fim
  $r_a = (1 - \varepsilon_l) \cdot r_a$
fim

4. Sistemas e Funções Não Lineares

Para testar a eficiência das duas hibridizações apresentadas, serão realizados testes para 2 sistemas de equações não lineares e 4 funções não lineares. Estas funções e sistemas são apresentados a seguir. Os dois sistemas estão presentes em [Ruggiero e Lopes, 1996]. As funções não lineares encontram-se em [Laguna e Martí, 2005].

4.1. Sistema 1 - Broyden Tridiagonal

$$\begin{cases} f_1(x) &= (3 - 2x_1)x_1 - 2x_2 + 1 &= 0 \\ f_i(x) &= (3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1 &= 0 \qquad i = 2, ..., 9 \\ f_{10}(x) &= (3 - 2x_{10})x_{10} - x_9 + 1 &= 0 \end{cases} \tag{1}$$

onde $F(x) = f_1^2 + f_2^2 + ... + f_{10}^2$

Mínimo global: $f(x^*) = 0$ em $x^* = (-0,5707 \quad -0,6818 \quad -0,7022 \quad -0,7055 \quad -0,7049 \quad -0,7015 \quad -0,6919 \quad -0,6658 \quad -0,5960 \quad 0,4164)$.

4.2. Sistema 2 - Função Trigexp de Toint

$$\begin{cases} f_1(x) = 3x_1^3 + 2x_2 - 5 + sen(x_1 - x_2)sen(x_1 + x_2) = 0 \\ f_i(x) = -x_{i-1}e^{x_{i-1}-x_i} + x_i(4 + 3x_i^2) + 2x_{i+1} + sen(x_i - x_{i+1})sen(x_i + x_{i+1}) - 8 = 0 \\ f_{10}(x) = -x_9 e^{x_9 - x_{10}} + 4x_{10} - 3 = 0 \end{cases} \quad (2)$$

onde $i = 2, ..., 9$ e $F(x) = f_1^2 + f_2^2 + ... + f_{10}^2$.

Mínimo global: $f(x^*) = 0$ em $x^* = (1, ..., 1)$.

4.3. Função 1 - Ackley

$$f(x) = -a \cdot exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} cos(cx_i)}\right) + a + exp(1), \quad (3)$$

onde $a = 20$, $b = 0,2$ e $c = 2\pi$, d- Dimensão da função.

Mínimo global: $f(x^*) = 0$ em $x^* = (0, ..., 0)$.

4.4. Função 2 - Griewank

$$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1. \quad (4)$$

Mínimo global: $f(x^*) = 0$ em $x^* = (0, ..., 0)$.

4.5. Função 3 - Levy

$$f(x) = sen^2(\pi w_i) + \sum_{i=1}^{d-1}(w_i - 1)^2[1 + 10sen^2(\pi w_i + 1)] + (w_d - 1)^2[1 + sen^2(2\pi w_d)], \quad (5)$$

onde $w_i = 1 + \frac{x_i - 1}{4}$.

Mínimo global: $f(x^*) = 0$ em $x^* = (1, ..., 1)$.

4.6. Função 4 - Rastrigin

$$f(x) = 10d + \sum_{i=1}^{d}[x_i^2 - 10cos(2\pi x_i)]. \quad (6)$$

Mínimo global: $f(x^*) = 0$ em $x^* = (0, ..., 0)$.

5. Resultados Numéricos

A partir das funções apresentadas, todos os testes foram realizados considerando $d = 10$, onde d indica o número de variáveis, compondo funções de 10 variáveis, assim como os sistemas de equações não lineares também testados. Para implementação dos métodos apresentados foi utilizado o software MATLAB. Os resultados foram gerados em um notebook LENOVO, munido de um processador Intel Core $i5$, $4,00$ GB de memória RAM.

A Tabela 1 traz os parâmetros utilizados para o método Luus Jaakola em todas as hibridizações testadas. O valor inicial para o método x_0 foi gerado de maneira aleatória considerando o intervalo para os componentes deste vetor no intervalo entre -1 e 1.

Tabela 1: Parâmetros - Método Luus Jaakola.

Parâmetros - Luus Jaakola e Hibridizações	
ε_l	$0,95$
ϵ	10^{-9}
n_{in}	800
n_{out}	500
r_a	$1,0$

Para os Métodos da Busca Coordenada e Hooke Jeeves inseridos no método Luus Jaakola foi considerada a tolerância 10^{-7} para melhora ou não do ponto gerado pelo método Luus Jaakola. Essa tolerância utilizada para realização de cada iteração dos métodos Busca Coordenada e Hooke Jeeves, para então ter prosseguimento o método Luus Jaakola.

Os resultados numéricos obtidos pela metodologia proposta são apresentados nas Tabelas 2 e 3. Nestas Tabelas, mostram-se o tempo gasto e também o número de iterações internas e externas (It_{In}/It_{Ex}) realizado pelos métodos LJ/BC - 1, LJ/BC - 2, LJ/HJ - 3 e LJ/HJ - 4. Como os mínimos encontrados já foram mostrados para cada sistema e função, omite-se sua apresentação nas Tabelas 2 e 3.

Tabela 2: Resultados Numéricos.

Metodologia	Método 1 - LJ/BC		Método 2 - LJ/BC	
Sistemas/Funções	Tempo (s)	It_{In}/It_{Ex}	Tempo (s)	It_{In}/It_{Ex}
Broyden Tridiagonal	0,4220	3480/5	0,2190	3392/5
Trigexp de Toint	1,0320	4017/6	0,4220	4005/6
Ackley	0,7820	5740/8	0,4530	5718/8
Griewank	0,4380	2849/4	0,2190	2643/4
Levy	0,8280	3305/5	0,3900	3238/5
Rastrigin	0,3750	3540/5	0,2660	4000/6

Tabela 3: Resultados Numéricos.

Metodologia	Método 3 - LJ/HJ		Método 4 - LJ/HJ	
Sistemas/Funções	Tempo (s)	It_{In}/It_{Ex}	Tempo (s)	It_{In}/It_{Ex}
Broyden Tridiagonal	0,4120	3376/5	0,2080	3360/5
Trigexp de Toint	0,9290	4057/6	0,3820	4041/6
Ackley	0,6720	5759/8	0,4200	5695/8
Griewank	0,3600	2743/4	0,1960	2569/4
Levy	0,7920	3240/5	0,3600	3239/5
Rastrigin	0,3840	4048/6	0,2340	4014/6

6. Conclusões

A partir da análise dos resultados apresentados nas Tabelas 2 e 3, é possível perceber que as quatro hibridizações foram eficientes encontrando o minimizador global de cada função testada e a solução dos dois sistemas também testados neste trabalho. No entanto, também é possível

perceber que, para os quatro métodos apresentados, os métodos 2 e 4 foram melhores, considerando-se o tempo gasto na determinação de todos os mínimizadores dos exemplos mostrados. O tempo sofreu uma redução próxima de 50%.

Além disso, quando foi utilizado o método Hooke Jeeves em conjunto com o método Luus Jaakola, os resultados encontrados, em sua maioria, foram mais vantajosos em comparação aos resultados obtidos com a utilização do método da Busca Coordenada.

Assim, percebe-se que para este problema apresentado, considerando funções e sistemas não lineares de 10 variáveis, a escolha do local de inserção do método Busca Coordenada ou Hooke Jeeves, em conjunto com o método Luus Jaakola, influenciou no tempo computacional gasto, onde a segunda escolha, acrescentando o algoritmo da Busca Coordenada ou Hooke Jeeves antes do início das iterações internas do método Luus Jaakola, forneceu melhores resultados quando comparado com a primeira escolha de inserção destes dois métodos.

Portanto, a metodologia aqui apresentada constitui uma boa ferramenta de resolução para o problema proposto, e também, a utilização do método Hooke Jeeves trouxe melhores resultados para este problema trabalhado, sendo assim, o metodo 4, que acrescenta o método Hooke Jeeves no início das iterações internas do método Luus Jaakola, foi o método mais eficiente, conseguindo encontrar os minimizadores globais das funções testadas em um menor tempo, quando comparado aos outros métodos testados.

*Referências Bibliográficas

Bazaraa, M. S. (2013). *Nonlinear Programming: Theory and Algorithms*. Wiley Publishing, 3rd edition. ISBN 1118857569, 9781118857564.

Diniz-Ehrhardt, M. A., Lopes, V. L. R., e Pedroso, L. G. (2010). Métodos sem derivadas para minimização irrestrita. `http://www.sbmac.org.br/notas.php`. Acessado: 12-01-2015.

Hooke, R. e Jeeves, T. A. (1961). Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, v. 8:212–229.

Laguna, M. e Martí, R. (2005). Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *Journal of Global Optimization*, 33(2):235–255.

Luus, R. e Jaakola, T. (1973). Optimization by direct search and systematic reduction of the size of search region. *AIChE Journal*, v. 19:760–766.

Oliveira, M. B. (2016). *Aplicação do Método Luus Jaakola em um Problema Termodinâmico*. Trabalho de Conclusão de Curso, INFES-UFF.

Ruggiero, M. A. G. e Lopes, V. L. R. (1996). *Cálculo Numérico - Aspectos Teóricos e Computacionais*. Pearson Education (Makron Books), São Paulo.

Souza, J. S. (2010). *Análise global da estabilidade termodinâmica de misturas: um estudo com o método do conjunto gerador*. Tese de Doutorado, IPRJ-UERJ.