Algoritmos de Busca Aleatória (*Random Search*)

Prof. Dr. Guilherme de Alencar Barreto

Outubro/2015

Departmento de Engenharia de Teleinformática
Programa de Pós-Graduação em Eng. de Teleinformática (PPGETI)
Universidade Federal do Ceará (UFC), Fortaleza-CE

gbarreto@ufc.br

1 Definições Preliminares

Considere uma função escalar (contínua) de p variáveis, $f : \mathbb{R}^p \rightarrow \mathbb{R}$; ou seja, $f(\cdot)$ é uma função que produz uma saída escalar $y \in \mathbb{R}$ e que possui p ($p \geq 1$) variáveis de entrada. Formalmente, podemos escrever

$$y = f(\mathbf{x}) = f(x_1, x_2, \ldots, x_p), \tag{1}$$

em que $\mathbf{x}$ é um vetor cujas componentes são as variáveis x_i, $i = 1, \ldots, p$. Como exemplos, para $p = 1$, temos que a parábola é uma função dada por

$$y = f(x) = (x - a)^2 + b, \tag{2}$$

cujo gráfico para $a = b = 5$ no plano cartesiano está mostrada na Figura 1a. A função equivalente em três dimensões é chamada de parabolóide, sendo escrita matematicamente como

$$z = f(x, y) = (x - a)^2 + (y - b)^2 + c, \tag{3}$$

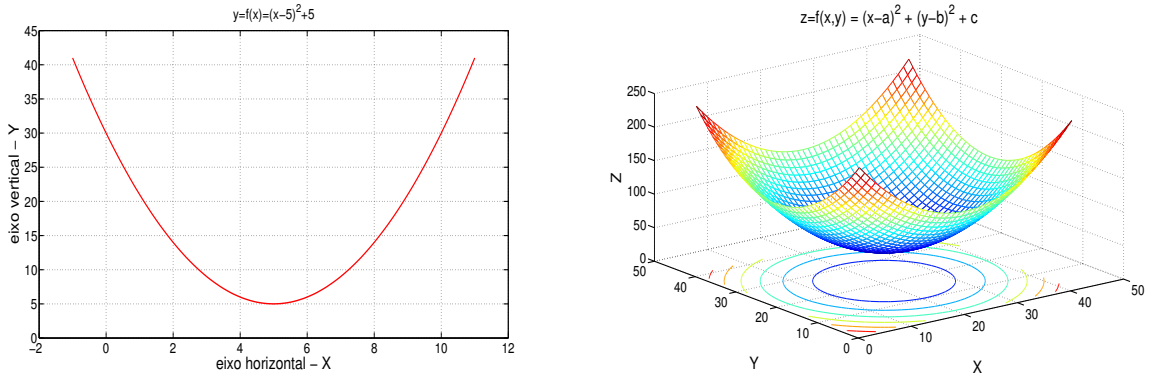
em que a, b e c são constantes reais. O gráfico desta função é uma superfície em 3 dimensões, mostrada na Figura 1b.

Note que as funções acima são contínuas e de variação suaves (i.e. não possuem interrupções ao longo do seu domínio, nem mudanças bruscas em seus gráficos). Além disso, são também funções convexas, ou seja, possuem apenas um ponto extremo, chamado de *mínimo global* neste caso.

Se a função mostrada na Eq. (2) for usada como função-custo ou função-objetivo em algum problema de minimização, o valor da variável x para o qual $y = f(x)$ produz seu maior valor é chamado de valor ótimo de x, simbolizado como x_{opt}. Se o problema envolver duas variáveis, como a função mostrada na Eq. (3), busca-se um vetor ótimo $\mathbf{x}_{opt} = [x_{opt} \quad y_{opt}]^T$, em que T simboliza o vetor-transposto.

De um modo geral, o problema de minimização de funções (sem restrições) pode ser formalizado matematicamente da seguinte maneira. O vetor $\mathbf{x}_{opt} \in \mathbb{R}^p$ é o vetor-ótimo se

$$f(\mathbf{x}_{opt}) < f(\mathbf{x}), \quad \forall \mathbf{x} \neq \mathbf{x}_{opt}, \tag{4}$$



(a) (b)

Figura 1: Representação gráfica de funções no plano cartesiano e em 3 dimensões. (a) $y = f(x) = (x-5)^2 + 5$, (b) $z = (x-20)^2 + (y-20)^2 + 50$.

ou, de maneira alternativa, como

$$\mathbf{x}_{opt} = \arg\min_{\forall \mathbf{x}} f(\mathbf{x}). \tag{5}$$

Restrições do tipo *caixa* são comuns em problemas de otimização. Tais restrições são descritas na forma de intervalos com limites inferiores e superiores para cada uma das p variáveis que compõem o vetor-solução $\mathbf{x} \in \mathbb{R}^p$. Assim, denotando a j-ésima componente de $\mathbf{x}$ como x_j, e seus limites inferior e superior como x_j^l e x_j^u, respectivamente, temos que uma restrição do tipo caixa é representada como

$$x_j^l \leq x_j \leq x_j^u. \tag{6}$$

Vetorialmente, podemos compactar a representação de todas as restrições do tipo caixa através da seguinte notação:

$$\mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u, \tag{7}$$

em que os vetores $\mathbf{x}^l$ e $\mathbf{x}^u$ contêm, respectivamente, os limites inferiores e superiores para todas as componentes do vetor-solução $\mathbf{x}$.

1. A notação de "menor ou igual a" usada na Eq. (7) é uma liberdade poética, pois um vetor em si não pode ser menor ou maior que outro, mas sim as normas (ou comprimentos) podem ser comparados.

2. Para tratar violações de restrições do tipo caixa, pode-se optar por um dos seguintes métodos:

   Método 1 - Gerar um número aleatório uniforme no intervalo $x_j^l \leq x_j \leq x_j^u$; ou seja:

   $$x_j \sim U(x_j^l, x_j^u), \quad \text{se } x_j < x_j^l \text{ ou } x_j > x_j^u, \tag{8}$$

   em que $u \sim U(a,b)$ denota um número aleatório uniformemente distribuído no intervalo (a,b).

   Método 2 - Forçar que a solução que violou os limites assuma um dos limites do intervalo; ou seja:

   $$\text{Se } x_j < x_j^l, \text{ então } x_j = x_j^l. \tag{9}$$
   $$\text{Se } x_j > x_j^u, \text{ então } x_j = x_j^u. \tag{10}$$

Nas próximas seções apresentarei duas técnicas de busca aleatória (uma de natureza global, a outra de natureza local) desenvolvidas com o intuito de obter o vetor-solução ótimo para funções convexa ou não. Para funções não-convexas, ou seja, com ótimos locais, os algoritmos a serem apresentados podem eventualmente encontrar o vetor-solução ótimo, mas não há garantias que isso irá de fato acontecer. Discutiremos como fazer na prática para ter alguma certeza de que a solução encontrar soluções subótimas, que podem ser consideradas boas soluções.

2 Busca Aleatória Global para Otimização Contínua

O algoritmo de busca aleatória global (GRS, do inglês *global random search*) é bem simples, pois consiste basicamente em testar soluções candidatas que são geradas aleatoriamente dentro do domínio da função a ser otimizada. Usaremos este método para encontrar o ponto-ótimo e o valor-ótimo correspondente da função de interesse. Iremos assumir também que as variáveis que compõem o vetor-solução assumem valores reais em um intervalo contínuo.

O algoritmo GRS pode ser caracterizado pelas seguintes propriedades:

1. É uma **heurística**, pois não é derivada a partir de princípios matemáticos formais, mas sim de conhecimento intuitivo ou informal sobre o domínio de um dado problema. Por isso, não há garantias de que a solução ótima seja encontrada, até porque muitas vezes não se conhece a solução ótima. Nestes casos, várias execuções do algoritmo são necessárias, com a solução subótima sendo dada pela solução que ocorre com maior frequência (ou seja, a moda das soluções).

2. É um método **estocástico** por dois motivos:

   - Exige a geração de uma solução inicial aleatória. Em outras palavras, a solução de partida é gerada aleatoriamente.
   - A geração de uma potencial solução a cada iteração do algoritmo é dependente de rotinas em que são gerados números (pseudo)aleatórios.

3. É um método de **solução única**, em que apenas uma solução-candidata é gerada a cada iteração; ao contrário de métodos populacionais como os algoritmos genéticos.

4. Não é um método de **inspiração biológica**, pois sua formulação teve motivação puramente computacional, sem apelo biológico, como os algoritmos genéticos.

5. É um método **livre de gradiente**, ou seja, não requer cálculo de gradientes para determinar direções de atualização de soluções. Por isso, pode ser usado para otimização de funções descontínuas.

2.1 Pseudocódigo do Algoritmo GRS

O pseudocódigo do algoritmo GRS consiste basicamente de poucos passos que são descritos a seguir. Para a execução destes passos, considere que a função de avaliação da solução, que normalmente é a própria função a ser otimizada, é representada por

$$f(\mathbf{x}) = f(x_1, x_2, \ldots, x_d). \tag{11}$$

Passo 1 - Especificar as restrições do tipo caixa para todas as d variáveis; ou seja,

$$x_j^l \leq x_j \leq x_j^u, \qquad j = 1, \ldots, d. \tag{12}$$

Passo 2 - Iniciar a melhor solução corrente $\mathbf{x}_{best}$ com valores aleatórios uniformes:

$$\mathbf{x}_{best} \sim U(\mathbf{x}^l, \mathbf{x}^u), \tag{13}$$

em que $\mathbf{x}^l$ e $\mathbf{x}^u$ são os vetores contendo, respectivamente, os limites inferiores (*lower limits*) e superiores (*upper limits*) das restrições do tipo caixa. A letra U denota a densidade de probabilidade uniforme.

Passo 3 - Avaliar a melhor solução corrente $\mathbf{x}_{best}$, i.e. calcular $f(\mathbf{x}_{best})$.

Passo 4 - Gerar uma solução candidata $\mathbf{x}_{cand}$ com valores aleatórios uniformes:

$$\mathbf{x}_{cand} \sim U(\mathbf{x}^l, \mathbf{x}^u). \tag{14}$$

OBS - Caso haja violações das restrições, aplicar um dos métodos descritos na Seção 1.

Passo 5 - Avaliar a solução candidata $\mathbf{x}_{cand}$, i.e. calcular $f(\mathbf{x}_{cand})$.

Passo 6 - Verificar como a solução candidata se sai em relação à melhor solução corrente.

Se $f(\mathbf{x}_{cand}) > f(\mathbf{x}_{best})$,
Então $\mathbf{x}_{best} = \mathbf{x}_{cand}$ e $f(\mathbf{x}_{best}) = f(\mathbf{x}_{cand})$.

Passo 7 - Retorne ao Passo 4 até que o algoritmo tenha convergido (i.e. permaneça inalterado por um certo número de iterações) ou até que o número máximo de iterações (N_{max}) tenha sido alcançado.

Passo 8 - Se uma das condições do Passo 7 seja verdadeira, encerre a execução do algoritmo e retorne $\mathbf{x}_{best}$ e $f(\mathbf{x}_{best})$.

Vale ressaltar que os passos do pseudocódigo acima foram desenvolvidso tendo-se em mente um problema de maximização. Para problemas de minimização trocar o sinal de "maior que" ($>$) no Passo 6 por um sinal de "menor que" ($<$).

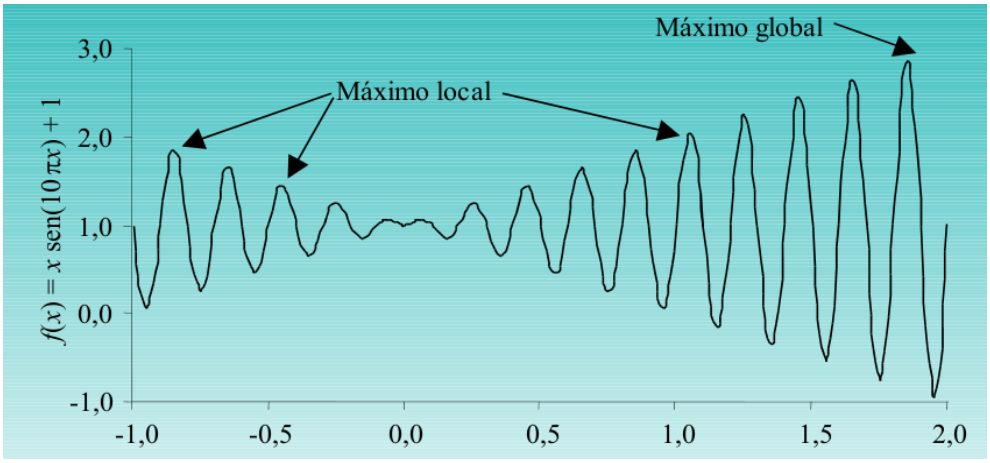


Figura 2: Representação gráfica da função $f(x) = x \cdot \sin(10\pi x) + 1$, no intervalo de -1 a 2.

2.2 Exemplo Ilustrativo - Algoritmo GRS

Considere o seguinte problema de otimização. Encontrar a solução-ótima que maximiza a função:

$$f(x) = x \cdot \sin(10\pi x) + 1, \tag{15}$$

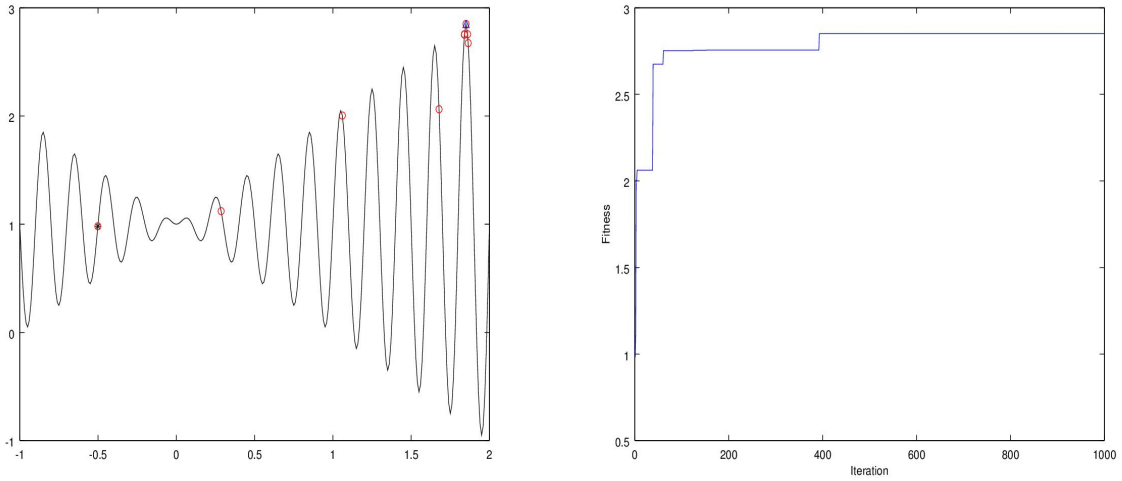
tal que $-1 \leq x \leq 2, x \in \mathbb{R}$. O gráfico desta função para o domínio especificado está mostrado na Figura 2. Trata-se não mais de uma função convexa no intervalo dado, pois a muitos ótimos (máximos) locais. O máximo global é $x_{opt} = 1,85$ e que o valor máximo é $f_{max} = f(x_{opt}) = 2,85$.

Os gráficos das soluções intermediárias de x_{best} e os valores correspondentes de $f(x_{best})$ ao longo de 1000 iterações estão mostrados na Figura 3. Na Figura 3(a), o asterisco denota a solução inicial, os círculos vermelhos denotam as soluções intermediárias e o triângulo azul denota a melhor solução ao final das iterações.

Ao término das 1000 iterações, a solução-ótima obtida foi $x_{best} = 1,8506$, que resultou em $f(x_{best}) = 2,8503$. O código Matlab/Octave usado para gerar os resultados desta seção estão listados nos apêndices ao final desta nota de aula.

3 Busca Aleatória Local para Otimização Contínua

O algoritmo de busca aleatória local (LRS, do inglês *local random search*) também é bem simples e muito similar ao algoritmo GRS. A principal diferença é que no LRS as soluções candidatas são geradas na vizinhança da melhor solução corrente $\mathbf{x}_{best}$. O algoritmo LRS, em suas propriedades gerais, é uma heurística estocástica uma vez que dependente de rotinas que geram números (pseudo)aleatórios. Além disso, o algoritmo LRS produz uma única solução a cada iteração, não é um método bioinspirado, e não depende do cálculo de gradientes podendo assim ser usado para otimização de funções descontínuas.



(a) (b)

Figura 3: Resultados da aplicação do algoritmo GRS na maximização da função mostrada na Figura 2. (a) Ilustração das soluções encontradas, desde a solução inicial (asterisco) até a solução final (triângulo). (b) Evolução do valor da função ao longo das iterações.

3.1 Pseudocódigo do Algoritmo LRS

O pseudocódigo do algoritmo LRS consiste basicamente de poucos passos que são descritos a seguir.

Passo 1 - Este passo está dividido em dois:

- **Passo 1.1** - Especificar as restrições do tipo caixa para todas as d variáveis de interesse ao problema; ou seja,

$$x_j^l \leq x_j \leq x_j^u, \qquad j = 1, \ldots, d. \tag{16}$$

- **Passo 1.2** - Especificar o desvio-padrão σ do ruído (perturbação aleatória) a ser adicionado à melhor solução corrente para gerar uma solução candidata.

Passo 2 - Iniciar a melhor solução corrente $\mathbf{x}_{best}$ com valores aleatórios uniformes:

$$\mathbf{x}_{best} \sim U(\mathbf{x}^l, \mathbf{x}^u), \tag{17}$$

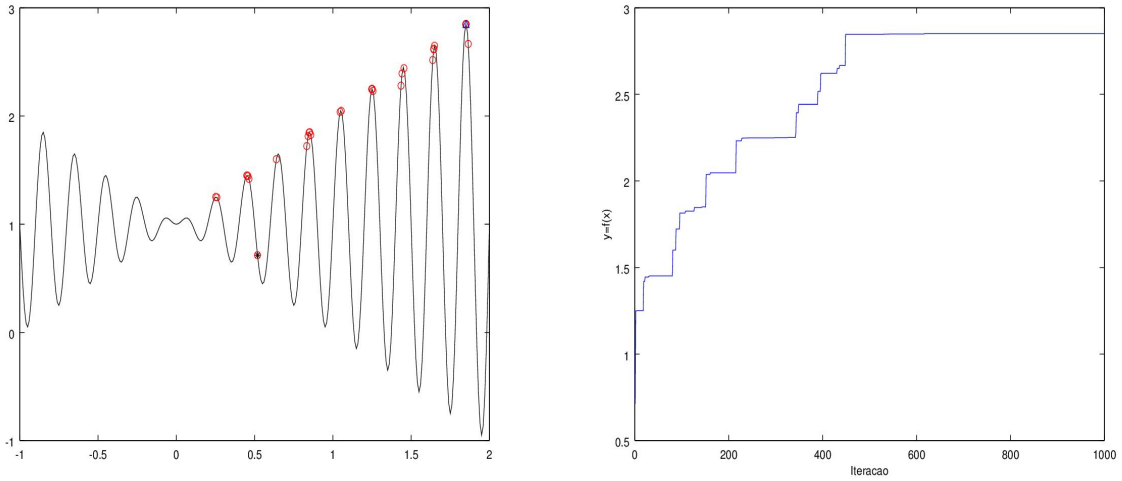
em que $\mathbf{x}^l$ e $\mathbf{x}^u$ são os vetores contendo, respectivamente, os limites inferiores (*lower limits*) e superiores (*upper limits*) das restrições do tipo caixa. A letra U denota a densidade de probabilidade uniforme.

Passo 3 - Avaliar a melhor solução corrente $\mathbf{x}_{best}$, i.e. calcular $f(\mathbf{x}_{best})$.

Passo 4 - Gerar uma solução candidata $\mathbf{x}_{cand}$ com valores aleatórios uniformes:

$$\mathbf{x}_{cand} = \mathbf{x}_{best} + \mathbf{n}, \tag{18}$$

em que $\mathbf{n} \in \mathbb{R}^d$ é um vetor aleatório que segue uma distribuição normal d-variada, de vetor-médio nulo e matriz de covariância $\sigma^2\mathbf{I}_d$; ou seja, $\mathbf{n} \sim N(\mathbf{0}_d, \sigma^2\mathbf{I}_d)$. Caso haja violações das restrições do tipo caixa, aplicar um dos métodos descritos na Seção 1.



(a) (b)

Figura 4: Resultados da aplicação do algoritmo LRS ($\sigma = 0,1$) na maximização da função mostrada na Figura 2. (a) Ilustração das soluções encontradas, desde a solução inicial (asterisco) até a solução final (triângulo). (b) Evolução do valor da função ao longo das iterações.

Passo 5 - Avaliar a solução candidata $\mathbf{x}_{cand}$, i.e. calcular $f(\mathbf{x}_{cand})$.

Passo 6 - Verificar como a solução candidata se sai em relação à melhor solução corrente.

Se $f(\mathbf{x}_{cand}) > f(\mathbf{x}_{best})$,
Então $\mathbf{x}_{best} = \mathbf{x}_{cand}$ e $f(\mathbf{x}_{best}) = f(\mathbf{x}_{cand})$.

Passo 7 - Retorne ao Passo 4 até que o algoritmo tenha convergido (i.e. permaneça inalterado por um certo número de iterações) ou até que o número máximo de iterações (N_{max}) tenha sido alcançado.

Passo 8 - Se uma das condições do Passo 7 seja verdadeira, encerre a execução do algoritmo e retorne $\mathbf{x}_{best}$ e $f(\mathbf{x}_{best})$.

OBS: Para problemas de minimização trocar o sinal de "maior que" ($>$) no Passo 6 por um sinal de "menor que" ($<$).

3.2 Exemplo Ilustrativo - Algoritmo LRS

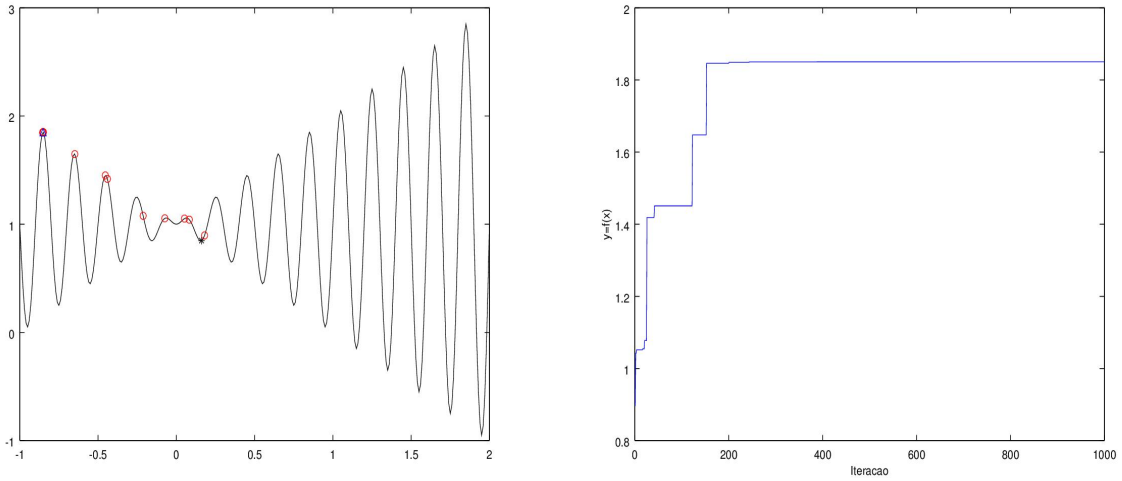
Considere o mesmo problema de otimização usado para testar o algoritmo GRS.

Os gráficos das soluções intermediárias de x_{best} e os valores correspondentes de $f(x_{best})$ ao longo de 1000 iterações estão mostrados na Figura 4. O valor escolhido para o desvio-padrão do ruído foi $\sigma = 0,1$.

Para esta rodada do algoritmo, a solução-ótima obtida foi $x_{best} = 1,8504$, que resultou em $f(x_{best}) = 2,8501$. O código Matlab/Octave usado para gerar os resultados desta seção estão listados no final desta nota de aula.

Em uma outra execução do algoritmo LRS, para o mesmo valor de desvio-padrão ($\sigma = 0,1$), obtivemos os resultados mostrados na Figura 5.

É possível notar que desta vez o algoritmo LRS convergiu para um máximo local. Este tipo de situação ocorre com frequência caso o valor do desvio-padrão do ruído seja muito pequeno.



(a) (b)

Figura 5: Outra execução do algoritmo LRS com $\sigma = 0,1$ para maximização da função mostrada na Figura 2. (a) Ilustração das soluções encontradas, desde a solução inicial (asterisco) até a solução final (triângulo). (b) Evolução do valor da função ao longo das iterações.

Neste caso, sugiro testar vários valores na faixa entre 0,1 e 0,5. Na Figura 6 são apresentados os resultados de uma execução do algoritmo LRS com o desvio-padrão aumentado para $\sigma = 0,5$. A função a ser otimizada é a mesma.

Com o aumento do desvio-padrão, o algoritmo LRS encontra com frequência o ponto de máximo global da função. A solução-ótima obtida para esta rodada foi $x_{best} = 1,8506$, que resultou em $f(x_{best}) = 2,8503$. O código Matlab/Octave usado para gerar os resultados desta seção estão listados no final desta nota de aula.

4 Sugestões de Exercícios Computacionais

Resolver as questões abaixo usando os algoritmo GRS e LRS.

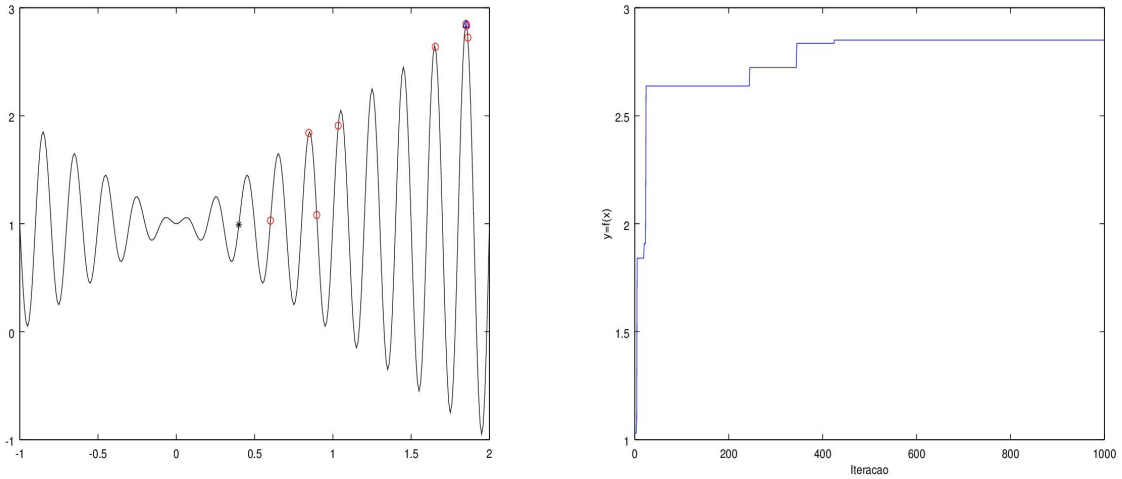
1. Quais as dimensões de uma caixa retangular sem tampa com volume 4 m e com a menor área de superfície possível?

2. Uma indústria produz dois produtos denotados por A e B. O lucro da indústria pela venda de x unidades do produto A e y unidades do produto B é dado por:

$$L(x, y) = 60x + 100y - \frac{3}{2}x^2 - \frac{3}{2}y^2 - xy. \tag{19}$$

3. Encontrar o ponto de mínimo e o valor mínimo da função de Ackley de 1 variável:

$$f(x) = -20e^{-0.2|x|} - e^{\cos(2\pi x)} + 20 + e. \tag{20}$$



(a) (b)

Figura 6: Execução do algoritmo LRS com $\sigma = 0,5$ para maximização da função mostrada na Figura 2. (a) Ilustração das soluções encontradas, desde a solução inicial (asterisco) até a solução final (triângulo). (b) Evolução do valor da função ao longo das iterações.

```
% Implementacao do algoritmo de busca aleatoria GLOBAL (GRS, global random search)
% para encontrar o maximo/minimo de funcoes de 1 variavel
%       f(x) = x^2,   0 < x < 31;
%       f(x) = x*sin(10*pi*x)+1,  -1 < x < 2.
%
% Autor: Guilherme A. Barreto
% Data: 15/11/2017

clear; clc; close all;

%%% Parametros do GRS
Ng=1000;   % Numero de iteracoes

funcao=2;  % Escolhe funcao-objetivo

% Escolhe limites do dominio de x
if funcao==1,
        limites=[0 31];   % Limites do intervalo para funcao: F=x^2
else    limites=[-1 2];   % Limites do intervalo de busca para funcao: F=x*sin(10*pi*x)+1;
end

x_best=unifrnd(limites(1),limites(2));  % Gera solucao inicial dentro do intervalo permitido

Fbest=func_objetivo1D(x_best,funcao);  % Avalia solucao inicial

%%% Roda GRS por Ng geracoes
for t=1:Ng,
    iteracao=t;

    x_cand=unifrnd(limites(1),limites(2));              % Gera solucao candidata

    Fcand=func_objetivo1D(x_cand,funcao);  % Avalia solucao candidata

    if Fcand>Fbest,     % Se x_cand produz melhor resultado que x_best
        x_best=x_cand;  % x_cand vira "melhor solucao ate o momento"
        Fbest=Fcand;
    end

    aptidao(t)=Fbest;

end

x_best, Fbest

figure; plot(aptidao);
xlabel('Iteration');
ylabel('Fitness');
```

```
% Implementacao do algoritmo de busca aleatoria LOCAL (LRS, local random search)
% para encontrar o maximo/minimo de funcoes de 1 variavel
%       f(x) = x^2,   0 < x < 31;
%       f(x) = x*sin(10*pi*x)+1,  -1 < x < 2.
%
% Autor: Guilherme A. Barreto
% Data: 29/11/2017

clear; clc; close all;

%%% Parametros do AG
Ng=1000;   % Numero de iteracoes

dp=0.5;    % Desvio-padrao do ruido aleatorio

funcao=2;  % Escolhe funcao-objetivo

% Escolhe limites do dominio de x
if funcao==1,
        limites=[0 31];   % Limites do intervalo para funcao: F=x^2
else    limites=[-1 2];   % Limites do intervalo de busca para funcao: F=x*sin(10*pi*x)+1;
end

x_best=unifrnd(limites(1),limites(2));  % Gera solucao inicial dentro do intervalo permitido

Fbest=func_objetivo1D(x_best,funcao);  % Avalia solucao inicial

%%%%%%%%%%% Grafico da funcao a ser otimizada
x=limites(1):0.01:limites(2);  % Dominio de x discretizado em incrementos de 0.1
f_opt=func_objetivo1D(x,funcao);  % Avalia solucao inicial
figure; plot(x,f_opt,'k-');
hold on
%%%%%%%%%%%

plot(x_best,Fbest,'k*');  %%% Plota solucao inicial sobre a funcao a ser otimizada

%%% Roda algoritmo LRS por Ng iteracoes
for t=1:Ng,
    iteracao=t;

    ruido=normrnd(0,dp);     % Perturbacao aleatorio na melhor solucao

    x_cand = x_best + ruido;    % Gera solucao candidata

    % Verifica se solucao candidata estah fora do limite superior
    if x_cand > limites(2),
        x_cand = limites(2);
    end

    % Verifica se solucao candidata estah fora do limite inferior
    if x_cand < limites(1),
        x_cand = limites(1);
    end

    Fcand=func_objetivo1D(x_cand,funcao);   % Avalia solucao candidata

    if Fcand>Fbest,     % Se x_cand produz melhor resultado que x_best,
        x_best=x_cand;  % x_cand vira "melhor solucao ate o momento"
        Fbest=Fcand;
    end

    plot(x_best,Fbest,'ro');  %%% Plota melhor solucao (solucoes intermediarias)

    aptidao(t)=Fbest;

end

x_best, Fbest

plot(x_best,Fbest,'b^');  %%% Plota melhor solucao
hold off

figure; plot(aptidao);
xlabel('Iteracao');
```

```
ylabel('y=f(x)');
```

```
function F=func_objetivo1D(x,funcao)
%
% Funcao objetivo 1D (1 variavel)
%
% Entradas:
%       x: valor a ser avaliado
%       funcao: 1, se f(x)=x^2
%               2, se f(x)=x*sin(10*pi*x)+1;
%
% Saida:
%       F: valor da funcao-objetivo em x
%
% Data: 22/11/2017
% Autor: Guilherme Barreto

if funcao==1,
       F=x.*x;
else   F=x.*sin(10*pi*x)+1;
end
```