Algoritmo da Subida da Encosta via Mutação Aleatória

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

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

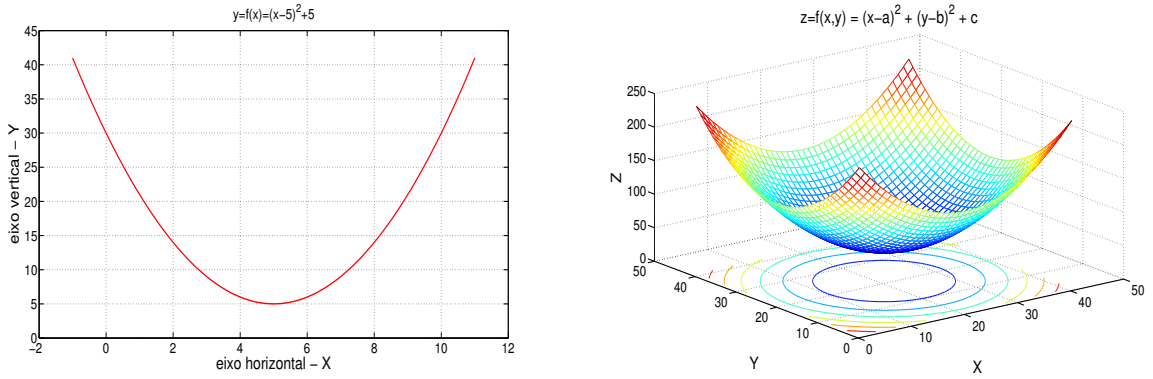
em que a, b e c são constantes reais. O gráfico desta função é uma superfície em 3 dimensões, mostrada na Figura 1b.

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

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

Nas próximas seções apresentarei uma técnica de busca heurística para obter o vetor-solução ótimo para funções convexa. Para funções não-convexas, ou seja, com ótimos locais, o algoritmo a ser apresentado pode 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 Algoritmo da Subida da Encosta

O algoritmo da subida da encosta por mutação aleatória (tradução livre do inglês: *random mutation hill-climbing*, RMHC), doravante simbolizado simplesmente pelo acrônimo RMHC [1], é uma das heurísticas para busca/otimização estocástica mais simples de compreender e implementar, por isso sendo de grande utilização prática.

O algoritmo RMHC 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.

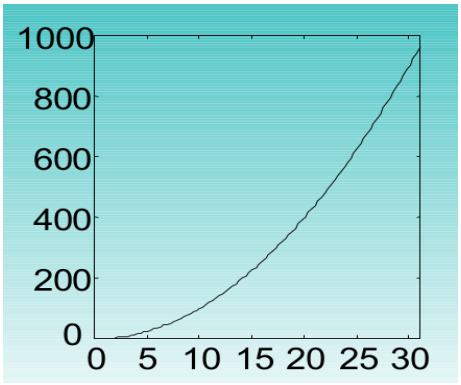


Figura 2: Representação gráfica da função $f(x) = x^2$, no intervalo de 0 a 31.

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 Codificação Binária da Solução

Os vetores-solução no algoritmo RMHC são comumente codificados em formato binário. Neste contexto, é preciso diferenciar o vetor-solução (valor numérico em base decimal), da *string binária* ou *vetor binário* representada em base 2. Isto significa que nos problemas de otimização teremos que transformar da base 2 para base 2. Discuto em mais detalhes duas possibilidades a seguir a partir de problemas de otimização de funções.

• **Caso 1: Vetor-solução é um número inteiro** - Considere o seguinte problema de otimização:

$$\text{Encontrar a solução-ótima que maximiza a função } f(x) = x^2, \quad 0 \leq x \leq 31, \ x \text{ é inteiro.} \quad (6)$$

O gráfico desta função para o domínio especificado está mostrado na Figura 2. Trata-se de uma função convexa no intervalo dado, em que não é preciso muito esforço intelectual para encontrar por simples inspeção visual que a solução-ótima é $x_{opt} = 31$ e que o valor máximo é $f_{max} = f(x_{opt}) = 31^2 = 961$.

Percebe-se por inspeção do enunciado que o domínio do problema está bem definido e possui 32 possíveis soluções. Assim, podemos codificar uma solução qualquer para este problema por uma string binária de 5 bits, porque teremos exatamente $2^5 = 32$ possíveis soluções candidatas. Por exemplo, o valor $x = 0$ seria codificada como a string $\mathbf{s} = [0 \ \ 0 \ \ 0 \ \ 0 \ \ 0]$, enquanto o valor $x = 31$ seria codificado como $\mathbf{s} = [1 \ \ 1 \ \ 1 \ \ 1 \ \ 1]$.

• **Caso 2: Vetor-solução é um número real** - Neste caso, o número de bits vai ser usado para definir em quantos níveis vamos discretizar (i.e. dividir) o intervalo que define o domínio do meu problema. Seja b o número de bits escolhido. Considere que o domínio da variável x é definido pelo intervalo fechado $[x_{min}, x_{max}]$, em que $x_{max} > x_{min}$. Assim, o número de níveis em que o intervalo $[x_{min}, x_{max}]$ é dividido é dado por 2^b. No jargão da área, costumamos dizer que o intervalo $[x_{min}, x_{max}]$ foi discretizado em 2^b níveis.

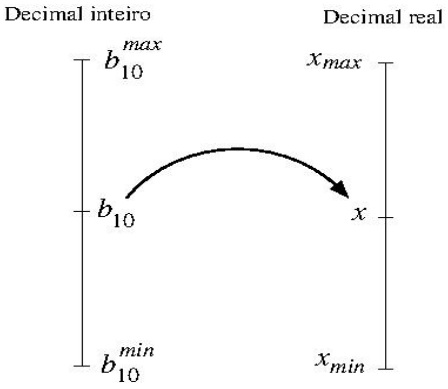


Figura 3: Transformação de escalas: de inteiro em base decimal para real na mesma base.

O menor número inteiro (na base decimal) que podemos representar com b bits é o próprio zero ($b_{10}^{min} = 0$), enquanto o maior número decimal inteiro que é representado com b bits é $b_{10}^{max} = 2^b - 1$. Logo, para converter qualquer número inteiro na base 10, representado por b_{10}, para um número real no intervalo $[x_{min}, x_{max}]$ desejado, devemos aplicar a seguinte regra de conversão de escala:

$$x = x_{min} + \left(\frac{x_{max} - x_{min}}{2^b - 1} \right) b_{10}. \tag{7}$$

A Eq. (7) é obtida a partir de uma operação de mudança de escala, tal qual aquelas usadas para transformação entre diferentes escalas de temperatura (e.g. de Celsius para Fahrenheit). Tomando como base a Figura 3, chegamos à seguinte igualdade:

$$\frac{x - x_{min}}{x_{max} - x_{min}} = \frac{b_{10} - b_{10}^{min}}{b_{10}^{max} - b_{10}^{min}}, \tag{8}$$

de modo que ao substituirmos os valores de $b_{10}^{min} = 0$, $b_{10}^{max} = 2^b - 1$ e isolarmos para x, chegamos à Eq. (7).

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

$$f(x) = x \cdot \sin(10\pi x) + 1, \tag{9}$$

tal que $-1 \leq x \leq 2, x \in \mathbb{R}$. O gráfico desta função para o domínio especificado está mostrado na Figura 4. Trata-se não mais de uma função convexa no intervalo dado, pois a muitos ótimos (máximos) locais. O máximo global é $x_{opt} = 1,85055$ e que o valor máximo é $f_{max} = f(x_{opt}) = 2,85027$.

Se optarmos por representar uma potencial solução por uma string binária de $b = 22$ bits, por exemplo, então o intervalo fechado e contínuo $[-1, 2]$ será discretizado em 2^b níveis. Para exemplificar o processo de conversão de uma string binária para um número inteiro binário e, então para o número real correspondente usando a fórmula mostrada na Eq. (7) é preciso executar a seguinte sequência de conversões:

Primeira conversão: Binário $\Rightarrow$ Inteiro Decimal

$$\mathbf{s} = [1000101110110101000111] \qquad \Rightarrow \qquad b_{10} = 2288967 \tag{10}$$

Segunda conversão: Inteiro Decimal $\Rightarrow$ Real $\in [x_{min} = -1, x_{max} = 2]$

$$b_{10} = 2288967 \qquad \Rightarrow \qquad x = -1 + \left(\frac{2 - (-1)}{2^{22} - 1} \right) \cdot 2288967 = 0,637197 \tag{11}$$

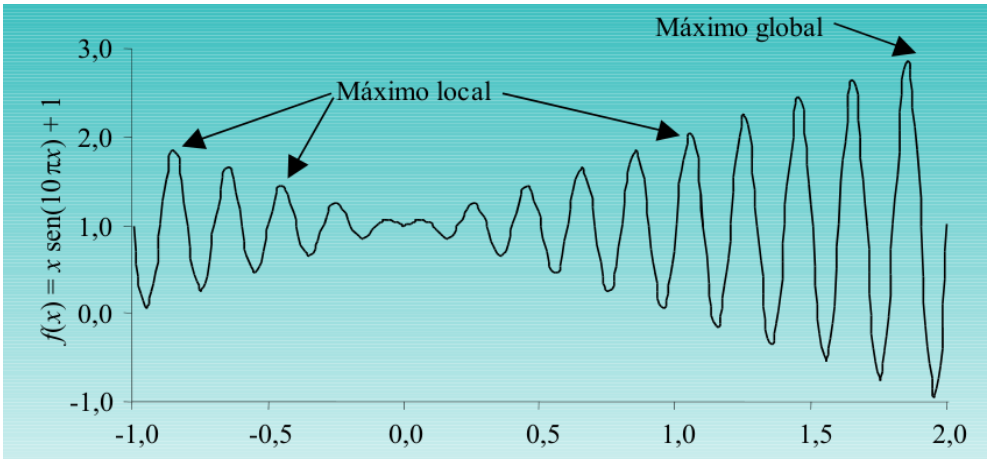


Figura 4: Representação gráfica da função $f(x) = x \cdot \sin(10\pi x) + 1$, no intervalo de -1 a 2.

Observação Importante: Em problemas de otimização com várias variáveis podemos compor os dois tipos de codificação descritos acima de forma independente para cada variável. Por exemplo, se o problema duas variáveis x e y, ambas reais. Neste caso, a primeira coisa a se fazer é determinar os intervalos do domínio de cada variável de modo a usar a Eq. (7) corretamente. Em seguida, escolhemos o número de bits da discretização de cada intervalo, que por simplicidade escolhemos o mesmo número (e.g. $b = 15$ bits). Portanto, a string binária para tal problema teria 30 bits, sendo que os primeiros 15 bits (da esquerda para a direita) codificam a variável x, enquanto os 15 bits restantes codificam a variável y.

Tendo explicado a questão da codificação binária do vetor-solução podemos seguir em frente e apresentar o pseudocódigo do algoritmo RMHC na próxima seção.

2.2 Pseudocódigo do Algoritmo RMHC e Implementação no Matlab/Octave

O pseudocódigo do algoritmo RMHC 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 $F_{opt}(\mathbf{s})$. Além disso, é preciso especificar o número de bits da string binária e o número máximo de iterações do algoritmo (N_{max}). Para facilitar o entendimento dos passos do algoritmo RMHC, incluo também a implementação correspondente em Matlab/Octave.

Passo 1 - Gerar uma string binária aleatória, chamá-la de $\mathbf{s}_{best}$ e avaliá-la.

```
>> s_best = round(rand(1,b));
>> Fbest=Fopt(s_best);
```

Passo 2 - Escolher aleatoriamente uma posição na string binária $\mathbf{s}_{best}$.

```
>> pos = floor(b*rand)+1;
```

Passo 3 - Gerar uma solução candidata $\mathbf{s}_{cand}$ pela operação de *negação binária* (NOT) do bit na posição "*pos*" em $\mathbf{s}_{best}$.

```
>> s_cand = s_best; % Copia s_best em s_cand
>> s_cand(pos) =  s_cand(pos); % Inverte bit na posicao ‘‘pos’’
```

Passo 4 - Avaliar a solução gerada pela string candidata $\mathbf{s}_{cand}$.

```
>> Fcand = Fopt(s_cand);
```

Passo 5 - Verifica se a string candidata gerada no Passo 3 ($\mathbf{s}_{cand}$) produz melhor solução numérica ou não que a melhor solução até o presente momento ($\mathbf{s}_{best}$).

```
>> if Fcand > Fbest, % se s_cand eh melhor que s_best
s_best = s_cand; % Substitui s_best por s_cand
Fbest = Fcand; % Substitui Fbest por Fcand
end
```

Passo 6 - Retorne ao Passo 2 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 7 - Se uma das condições do Passo 6 seja verdadeira, encerre a execução do algoritmo e retorne $\mathbf{s}_{best}$ e `Fbest`.

Vale ressaltar que o pseudocódigo acima foi desenvolvido tendo-se em mente um problema de maximização. Para problemas de minimização trocar o sinal de “maior que” ($>$) no Passo 4 por um sinal de “menor que” ($<$).

2.3 Implementação Completa em Matlab/Octave

A Figura 5 traz a listagem de uma implementação completa do algoritmo RMHC para o problema da Figura 3. Na Figura 5, apresento as rotinas auxiliares `aptidao1.m` e `bina2deci.m` que são usadas inernamente ao código da Figura 5.

2.4 Exercícios

Resolver as questões abaixo usando o algoritmo RMHC.

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:

$$L(x, y) = 60x + 100y - \frac{3}{2}x^2 - \frac{3}{2}y^2 - xy. \tag{12}$$

Referências

[1] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 5th edition, 1999.

```
clear; clc; close all;

%%% Parametros do algoritmo RMHC
N=22;    % Tamanho da string
Nmax=500;   % Numero de iteracoes

s_best=round(rand(1,N)); % Gera string inicial, tornando-a "melhor string"
[x_best Fbest]=aptidao1(sbest);  % Avalia string inicial

%%% Roda algoritmo por Nmax iteracoes
for t=1:Nmax,
    iteracao=t,

    Aptidao(t)=Fbest;

    %%%%%%%%%%%%%%%
    %%% Passo 2: Gera nova string por mutacao em posicao aleatoria
    %%%%%%%%%%%%%%%
    pos=floor(N*rand)+1;  % Seleciona posicao aleatoria na string
    s_cand=s_best;
    s_cand(pos)=~s_cand(pos); % negacao do bit na posicao "pos"

    [x_cand Fcand]=aptidao1(s_cand);  % Avalia nova string

    if Fcand>Fbest, % Se aptidao da nova string eh maior
        s_best=s_cand; % ela vira "melhor string" ate o momento
        Fbest=Fcand;
        x_best=x_cand;
    end
end

x_best, s_best, Fbest

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

Figura 5: Código Matlab/octave completo para otimização da função $f(x) = x^2$, no intervalo de 0 a 31, usando o algoritmo RMHC.

```
function [X Fopt]=aptidao1(str);
% Calcula aptidao dos individuos da populacao P
% Entrada:
%    str: string binaria
% Saida:
%    X - numero correspondente em base decimal
%    F - Valor correspondente a avaliacao de X pela funcao de otimizacao

X=bina2deci(str);  % Converte de binario para decimal
Fopt=X*X;       % Avalia X usando a funcao de otimizacao Fopt(x)=x^2
```

(a)

```
function Y=bina2deci(X)
% rotina simples para converter de binario para decimal

n=length(X);
ex=fliplr(cumsum(ones(1,n)))-1;
aux=2.^ex;
Y=dot(X,aux);
```

(b)

Figura 6: Rotinas auxiliares ao código mostrado na Figura 5.