遗传算法

山东理工大学数学学院 周世祥

Contents

思路

$2^7=128$个成员　256个 (2^8)

基本思想

可以求解困难问题。基于遗传科学和自然选择的过程，有学科交叉的特点，在计算机科学领域 也有许多有成效的应用。比如并行计算。 遗传算法作用在一个小的种群上，这个种群可以对应于一个特定的变量值，种群的大小 可能变化，通常与所考虑的问题相关。种群的成员通常为0和1的字符串，即二进制字符串。 如1000010,1110000,1010101,1111001,100001； 实际计算中，种群可能更长。这个01序列就是一个个体，对应于染色体，每个二进制位 理解为基因。跟遗传学一样，有一套流程：基于适应度的选择；交叉；变异。

5个成员

在繁殖阶段，随机选择一组染色体，根据适应度，选择种群繁殖的成员，最适应的具有 更大的繁殖概率，这个概率与它们的适应度值成比例。 **所谓适者生存** 。

配对过程实现简单的交叉想法： 种群的两个成员交换基因。有很多交叉方法：一个交叉点， 或多个交叉点，交叉点随机选择。比如，

交叉点选择（随机）

交

1110|000

1010|101

交叉后给出的新的染色体：

遗传，变异　$P\le 0.1$

子

1110|101

1010|000

将这个过程应用于初始种群，就可以产生新的一代。最后的过程是变异。

变异的想法就是随机在一个特定的染色体上改变一个特定的基因。这样0可以变成1,1可以 变成0，遗传算法中变异的概率很低。

以一个简单的例子为例，说明如何处理优化问题。

设 r为半径的半球，高为2的圆柱合体构成一个容器，希望使得容器的容量尽量大。 $\max v = 2\pi r^3/3 + 2\pi^2$，其中，$2 \le r \le 4$.

如何转化为遗传算法可以直接应用的问题。首先必须生成一组初始的字符串构成的初始 种群，每个字符串中的二进制数的个数，即字符串的长度限制了解的精度。初始化种群 的大小，决定了算法所花费的时间。matlab函数genbin可以用来生成这样的初始种群。

$2^7=128$个染色体　(20 -> 8个)

```
% function chromosome = genbin(bitl,numchrom)
% % Example call: chromosome = genbin(bitl,numchrom)
% % Generates numchrom chromosomes of bitlength bitl.
% % Called by optga.m
%
% % This MATLAB function is to accompany 'Numerical Methods using MATLAB'
% % by GR Lindfield and JET Penny,
% % published by Academic Press, an imprint of Elsevier, 2012.
%
% maxchros = 2^bitl;
% if numchrom>=maxchros
%     numchrom = maxchros;
% end
% chromosome = round(rand(numchrom,bitl));
```

(0,1) 均分布

[0,63] =1

[0,35] = 0

0,1

产生5个长度为6的基因初代种群。

```
chroms=genbin(6,5)
```

6

```
chroms =

     1     0     1     0     0     1
     1     1     1     1     1     0
     0     1     1     1     1     0
     0     1     0     0     0     1
     0     0     1     1     1     1
```

(1)
(2)
(3)
(4)
(5)

因为感兴趣的x值在2到4范围内，所以必须把这些二进制字符串转换为2到4范围内的 值。利用matlab函数binvreal来完成，将一个二进制值转换为所要求的范围内的实值。

[2 : 4]

(0 1 0 1 0 1)

```
% function rval = binvreal(chrom,a,b)
% % Converts binary string chrom to real value in range a to b.
% % Example call rval = binvreal(chrom,a,b)
% % Normally called from optga.
%
% % This MATLAB function is to accompany 'Numerical Methods using MATLAB'
% % by GR Lindfield and JET Penny,
% % published by Academic Press, an imprint of Elsevier, 2012.
%
% [pop bitlength] = size(chrom);
% maxchrom = 2^bitlength-1;
% realel = chrom.*((2*ones(1,bitlength)).^fliplr([0:bitlength-1]));
% tot = sum(realel);
% rval = a+tot*(b-a)/maxchrom;

for i=1:5,rval(i)=binvreal(chroms(i,:),2,4);end
rval
```

```
rval =

    3.3016    3.9683    2.9524    2.5397    2.4762
```

适应度信息

适应度值的计算

定义目标函数：

```
g=@(x) pi*(0.66667*x+2).*x.^2;
% 适应度
fit=g(rval)
% 这一阶段的总的适应度为
sum(fit)

% 可以看出最适应的值是哪一个成员或染色体。
```

```
fit =

  143.8650   229.8173   108.6664    74.8348    70.3244

ans =

  627.5079
```

上述过程可以封装为一个函数fitness()

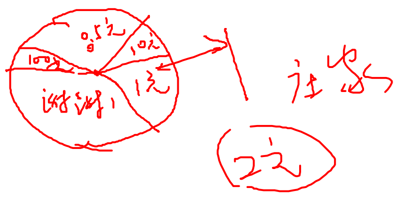
```
% function [fit,fitot] = fitness(criteria,chrom,a,b)
% % Example call: [fit,fitot] = fitness(criteria,chrom,a,b)
% % Calculates fitness of set of chromosomes chrom in range a to b.
% % using the fitness criterion, criteria.
% % Called by optga.
% % Calculates fitness of set of chromosomes in range a to b.
%
% % This MATLAB function is to accompany 'Numerical Methods using MATLAB'
% % by GR Lindfield and JET Penny,
% % published by Academic Press, an imprint of Elsevier, 2012.
%
% [pop bitl] = size(chrom);
% for k = 1:pop
%     v(k) = binvreal(chrom(k,:),a,b);
%     fit(k) = feval(criteria,v(k));
% end
% fitot = sum(fit);
% 执行
[fit,sum_fit]=fitness(g,chroms,2,4)
```

```
fit =

  143.8650   229.8173   108.6664    74.8348    70.3244

sum_fit =

  627.5079
```



繁殖阶段

按照适应度复制字符串，因此交配库中的最适应的染色体具有更高的概率。这个选择的过程 比较复杂，基于模拟轮盘赌的过程。分配给一个特定字符串的轮盘的百分比正比与 字符串的适应度。对于适应度向量fit，这个百分比计算如下：

```
percent=100*fit/sum_fit

sum(percent)

% 概率越大，染色体被选中的几率就越大。
```

```
percent =

   22.9264   36.6238   17.3171   11.9257   11.2069

ans =

   100
```

algorithm

由selectga()函数实现

```
% function newchrom = selectga_g(criteria,chrom,a,b)
% % Example call: newchrom = selectga_g(criteria,chrom,a,b)
% % Selects best chromosomes from chrom for next generation
% % using function criteria in range a to b.
% % Called by function optga_g.
% % Selects best chromosomes for next generation using criteria
%
% % This MATLAB function is to accompany 'Numerical Methods using MATLAB'
% % by GR Lindfield and JET Penny,
% % published by Academic Press, an imprint of Elsevier, 2012.
%
% [pop bitlength] = size(chrom);
% fit = [ ];
% %calculate fitness
% [fit,fitot] = fitness_g(criteria,chrom,a,b);
% for chromnum = 1:pop
%     sval(chromnum) = sum(fit(1,1:chromnum));
% end
% %select according to fitness
% parname = [ ];
% for i = 1:pop
%     rval = floor(fitot*rand);
%     if rval<sval(1)
%         parname = [parname 1];
%     else
%         for j = 1:pop-1
%             sl = sval(j);
%             su = sval(j)+fit(j+1);
%             if (rval>=sl) & (rval<=su)
%                 parname = [parname j+1];
%             end
%         end
%     end
% end
% newchrom(1:pop,:) = chrom(parname,:);

% 上述函数实现选择如下:
matepool =selectga(g,chroms,2,4)

% 由于选择的随机性，我们发现，有的成员没有被选中，有的成员尽管适应度较高，
% 也不一定被选中，接下来重新评估新种群的适应度。
```

内存 池

```
matepool =

     0     1     0     0     0     1
     1     1     1     1     1     0
     0     0     1     1     1     1
     0     1     1     1     1     0
     1     0     1     0     0     1
```

```
fitness(g,matepool,2,4)
sum(ans)
% 会发现整体的适应度显著增加
```

```
ans =

   74.8348  229.8173   70.3244  108.6664  143.8650


ans =

  627.5079
```

交叉

只配对其中的一部分个体，比如60%的个体，本例为3个，因为配对为偶数，向下取整，所以取2， 随机选择种群的2个成员进行配对，实现函数为matesome.

```
% function chrom1 = matesome(chrom,matenum)
% % Example call: chrom1 = matesome(chrom,matenum)
% % Mates a proportion, matenum, of chromosomes, chrom.
%
% % This MATLAB function is to accompany 'Numerical Methods using MATLAB'
% % by GR Lindfield and JET Penny,
% % published by Academic Press, an imprint of Elsevier, 2012.
%
% mateind = [ ];
% chrom1 = chrom;
% [pop bitlength] = size(chrom);
% ind = 1:pop;
% u = floor(pop*matenum);
% if floor(u/2)~=u/2
%     u = u-1;
% end
% % select percentage to mate randomly
% while length(mateind)~=u
%     i = round(rand*pop);
%     if i==0
%         i = 1;
%     end
%     if ind(i)~=-1
%         mateind = [mateind i];
%         ind(i) = -1;
%     end
% end
% %perform single point crossover
% for i = 1:2:u-1
%     splitpos = floor(rand*bitlength);
%     if splitpos==0
%         splitpos = 1;
```

```
%     end
%     i1 = mateind(i);
%     i2 = mateind(i+1);
%     tempgene = chrom(i1,splitpos+1:bitlength);
%     chrom1(i1,splitpos+1:bitlength) = chrom(i2,splitpos+1:bitlength);
%     chrom1(i2,splitpos+1:bitlength) = tempgene;
% end

newgen=matesome(matepool,0.6);


% matepool =
%
%      1     1     1     1     1     0
%      1     1     0     1     1     1
%      1     0     1     0     0     1
%      1     1     1     1     1     0
%      1     0     0     0     1     1

% newgen =
%
%      1     1     1     1     1     0  %原种群继承下来的个体
%      1     1     0     1     1     0  % 第2个和第4个被选中交配
%      1     0     1     0     0     1 %原种群继承下来的个体
%      1     1     1     1     1     1  % 第2个和第4个被选中交配
%      1     0     0     0     1     1 %原种群继承下来的个体
```

计算新种群的适应度，有

```
fitness(g,newgen,2,4)

sum(ans)

% 注意到这个阶段，总适应度并不一定总能得到改善，当然不能期望每次都会改善。
```

```
ans =

   74.8348   229.8173    68.1355   111.6079   143.8650


ans =

  628.2606
```

陷入局部最优

基因突变

最后一个阶段执行基因突变，然后再重复相同的循环步骤，实现变异的函数为 mutate():

```
% function chrom = mutate(chrom,mu)
% % Example call: chrom = mutate(chrom,mu)
% % mutates chrom at rate given by mu
% % Called by optga
%
% % This MATLAB function is to accompany 'Numerical Methods using MATLAB'
% % by GR Lindfield and JET Penny,
% % published by Academic Press, an imprint of Elsevier, 2012.
%
% [pop bitlength] = size(chrom);
% for i = 1:pop
```

```
%     for j = 1:bitlength
%         if rand<=mu
%             % Neater method of mutation compared with that given in text.
%             chrom(i, j)~=chrom(i, j);
%         end
%     end
% end

% mu为变异率，一般取的非常小，这种规模的种群不可能在一代中就发生改变，该函数调用如下：

mutate(newgen, 0.05);
```

```
*   1   1   1   1   1   0
*   1   1   0   1   1   0
*   1   0   1   0   0   1
*   1   1   1   1   1   1
*   1   0   0   0   1   1
```

```
% 突变时基因的某个位发生改变，从1变成0，从0变成1.有时也可能不会发生变异，因为
% 变异率很低。
```

重复上述步骤，用optga函数封装上述循环

```
% function [xval,maxf] = optga(fun,range,bits,pop,gens,mu,matenum)
% % Determines maximum of a function using the Genetic algorithm.
% % Example call: [xval,maxf] = optga(fun,range,bits,pop,gens,mu,matenum)
% % fun is name of a one variable user defined positive valued function.
% % range is 2 element row vector giving lower and upper limits for x.
% % bits is number of bits for the variable, pop is population size.
% % gens is number of generations, mu is mutation rate,
% % matenum is proportion mated in range 0 to 1.
% % WARNING. Method is not guaranteed to find global optima.
%
% % This MATLAB function is to accompany 'Numerical Methods using MATLAB'
% % by GR Lindfield and JET Penny,
% % published by Academic Press, an imprint of Elsevier, 2012.
%
% newpop = [ ];
% a = range(1);
% b = range(2);
% newpop = genbin(bits,pop);
% for i = 1:gens
%     selpop = selectga(fun,newpop,a,b);
%     newgen = matesome(selpop,matenum);
%     newgen1 = mutate(newgen,mu);
%     newpop = newgen1;
% end
% [fit,fitot] = fitness(fun,newpop,a,b);
% [maxf,mostfit] = max(fit);
% xval = binvreal(newpop(mostfit,:),a,b);
```

想想最初的问题，指定自变量的范围从2到4，使用8位染色体，且初始种群数为10 这个过程连续进行20代，变异概率为0.005，交叉概率matenum为0.6，需要在0到1之间。 使用8位长的染色体，就给出了2^8=256个划分，区间从2到4被划分为256和分区，每个长 为0.0078125.

二进制编码

```
[x f]=optga(g,[2 4],8,10,20,0.005,0.6);
% x =
%     3.8980
% f =
```

```
%    219.5219

% 精确解我们可以看出为 x=4,所以这个结果是一个合理的结果。
```

遗传算法背后的理论基础

遗传算法不同于简单的直接搜索过程的原因是，它具有两个特点：交叉和变异。从初始 种群开始，算法发展了新的世代，从而迅速探索到所关注的区域，这对复杂的优化问题 非常有用。特别是那种有很多局部极大和极小，又希望找到函数的全局最大或最小的情况。 共轭梯度法，只能找到局部解，而遗传算法，却有可能找到全局最优解，避免了 卡在某个局部极小位置。

发明者Holland模式定理解释了遗传算法的作用机理：11***00,***0001等结构的 字符串，比如以11开头，00结尾的，就是一种模式，繁殖过程中，有些具有较高的适应度， 他证明了，在世代繁殖中低适应度的模式会呈指数速度消亡。理论是复杂的，我们这里不谈。

求函数在x=0到x=1范围内的最大值。 $f(x) = e^x + sin(3\pi x)$.

定义h函数为：

```
h=@(x) exp(x)+sin(3*pi*x);
% 调用optga函数
[x f]=optga(h,[0,1],8,40,50,0.005,0.6);
% x =
%       0.8784
% f =
%       3.3181
```

matlab的fminsearch函数也可以解决该问题。

```
h1=@(x) -(exp(x)+sin(3*pi*x));  %注意要加一个负号，变为极小化
fminsearch(h1,0,1);

h1(ans);
% ans =
%       0.1802
% ans =
%      -2.1893
% 很显然只找到局部最优解。
```

试一试

画出函数 $f(x) = 10 + \frac{1}{(x-0.16)^2+0.1}\sin(1/x)$的图像，并求其最大值。

x范围从0.001到0.3之间。参考结果大约为x=0.1288,f(x)=19.8631.

遗传算法的改进

关于编码，可以用Gray格雷码代替二进制编码；可以用多种不同的方式实现轮盘选择； 交叉可改为多点交叉或其他，人们常常觉得遗传算法运行缓慢，但记住，遗传算法 最好应用在难题上，比如有多个优化解但需要求全局最优解的那些问题，标准算法 经常失效，用遗传算法费点时间是值得的。

格雷码：格雷码是一种二进制数系统，其中相邻的代码只有一个二进制数不同，是格雷开发的。 定义汉明距离，汉明距离是两个二进制向量之间对应位不同的数量。

1, 2

$h(m_1, m_2) = 2$

$h(m_1, m_3) = 1$

- 十进制 0 1 2 3 4 5 6 7
- 二进制 000 001 010 011 100 101 110 111
- 格雷码 000 001 011 010 110 111 101 100

对于遗传算法，须把格雷码转换成十进制，为此，需采取两个阶段：先把格雷码转换为 二进制，然后再把二进制转换为十进制，把格雷码转换为二进制，算法为：

- 对第一位(最高位)b(1)=g(1)
- 对第i位，其中i=2,...,n,

若b(i-1)=g(i),则b(i)=0,否则b(i)=1

其中b(i)是二进制，g(i)是等价的格雷码，该算法可用函数grayvreal来实现。

```
%  function rval = greyvreal(grey,a,b)
% % Converts grey string to real value in range a to b.
% % Example call rval = greyvreal(grey,a,b)
% % Normally called from optga_g.
%
% % This MATLAB function is to accompany 'Numerical Methods using MATLAB'
% % by GR Lindfield and JET Penny,
% % published by Academic Press, an imprint of Elsevier, 2012.
%
% [pop bitlength] = size(grey);
% maxchrom = 2^bitlength-1;
% % Converts grey to binary
% bin(1) = grey(1);
% for i = 2:bitlength
%     if bin(i-1) == grey(i)
%         bin(i) = 0;
%     else
%         bin(i) = 1;
%     end
% end
% % Converts binary to real
% reale1 = bin.*((2*ones(1,bitlength)).^fliplr([0:bitlength-1]));
% tot = sum(reale1);
% rval = a+tot*(b-a)/maxchrom;

% 这个函数可以替代函数fitness（重命名为fitness_g）中的binvreal，在selectga（重命名为selectga_g）
% 中使用，最后，函数optga_g中会使用fitness_g和select_g。

g=@(x)exp(x)+sin(3*pi*x);
[x,f]=optga_g(g,[0 1],8,40,50,0.005,0.6)

% 研究人员声称用格雷码有显著的优势。
```

```
x =

    0.8039


f =

    3.1961
```

[2 , 4]　　8bit

x_1 , x_2 , x_3 , x_4　　32

连续遗传算法

有时候初始种群是一组随机实数而非二进制数，初始种群值，可以是感兴趣区域内 任意连续的集合，而不是在二进制形式算法中使用的二进制值的离散集。

若假设待优化的函数有四个变量，则初始的每条染色体都是由四个随机产生的十进制数 构成的向量，每个数都位于变量的搜索范围内，如果选择有20条染色体的种群，那么对应于 适应标准，每条染色体都可以计算出它们的适应度，选择若干最适应的进行配

对过程，例如从 20条中选择最适应的8条染色体构成一个组作为交配库，从这组中，选择随机对用来 交叉和配对。

配对过程大致类似于二进制形式的遗传算法，即选择一个交叉的随机点，简单交换两个 染色体上的实变量值，使得父母染色体在该点出混合，但这种方式没有产生该区域内的新值， 因此一般建议组合公式为：设有两个配对的染色体：$r_1 = [u_1, v_1, w_1, x_1]$, $r_2 = [u_2, v_2, w_2, x_2]$

在随机点处用一对染色体的线性随机组合，创建出两个新值，取代原染色体交叉点处的值。

$x_a = x_1 - \beta(x_1 - x_2), x_a = x_1 - \beta(x_1 - x_2)$

类似的公式可以应用到变量 u, v, w 上，在每一代 β，交叉点以及先前种群中 成对的适应成员都可以重新选择。 交叉点可以在随机点1,2,3,4处随机出现，根据交叉点的选择，配对后的新染色体为：

- 交叉点1：$r_1 = [u_a, v_2, w_2, x_2], r_2 = [u_b, v_1, w_1, x_1]$
- 交叉点2：$r_1 = [u_1, v_a, w_2, x_2], r_2 = [u_2, v_b, w_1, x_1]$
- 交叉点3：$r_1 = [u_1, v_1, w_a, x_2], r_2 = [u_2, v_2, w_b, x_1]$
- 交叉点4：$r_1 = [u_1, v_1, w_1, x_a], r_2 = [u_2, v_2, w_2, x_b]$

注意到一维问题就无法使用这个特定的配对算法求解。

变异过程也类似于二进制遗传算法，选择一个变异率，则变异数可通过染色体数目 和染色体上元素的数目计算出来。然后随机选择染色体上的位置，用某区域内随机选出 的值替换染色体上的这些值。用contgaf函数实现连续遗传算法。

```
% function [x,f] = contgaf(func,nv,range,pop,gens,mu,matenum)
% % function for continuous genetic algorithm
% % func is the multivariable function to be optimised
% % nv is the number of variables in the function (minimum = 2)
% % range is row vector with 2 elements. i.e [lower bound upper bound]
% % pop is the number of chromosomes, gens is the number of generations
% % mu is the mutation rate in range 0 to 1.
% % matenum is the proportion of the population mated in range 0 to 1.
%
% % This MATLAB function is to accompany 'Numerical Methods using MATLAB'
% % by GR Lindfield and JET Penny,
% % published by Academic Press, an imprint of Elsevier, 2012.
%
% pops = [ ];
% fitv = [ ];
% nc = pop;
% % Generate chromosomes as uniformly distributed sets of random decimal
% % numbers in the range 0 to 1
% chrom = rand(nc,nv);
% % Generate the initial population in the range a to b
% a = range(1);
% b = range(2);
% pops = (b-a)*chrom+a;
% for MainIter = 1:gens
%     % Calculate fitness values
%     for i = 1:nc
%         fitv(i) = feval(func, pops(i,:));
%     end
%     % Sort fitness values
%     [sfit,indexf] = sort(fitv);
%     % Select only the best matnum values for mating
%     % ensure an even number of pairs is produced
%     nb = round(matenum*nc);
%     if nb/2~=round(nb/2)
%         nb = round(matenum*nc)+1;
%     end
%     fitbest = sfit(1:nb);
```

```
%     % Choose mating pairs use rank weighting
%     prob = @(n) (nb-n+1)/sum(1:nb);
%     rankv = prob([1:nb]);
%     for i = 1:nb
%         cumprob(i) = sum(rankv(1:i));
%     end
%     % Choose two sets of mating pairs
%     mp = round(nb/2);
%     randpm = rand(1,mp);
%     randpd = rand(1,mp);
%     mm = [ ];
%     for j = 1:mp
%         if randpm(j)<cumprob(1)
%             mm = [mm,1];
%         else
%             for i = 1:nb-1
%                 if (randpm(j)>cumprob(i)) && (randpm(j)<cumprob(i+1))
%                     mm = [mm i+1];
%                 end
%             end
%         end
%     end
%     % The remaining elements of nb = [1 2 3,...] are the other ptnrs
%     md = [ ];
%     md = setdiff([1:nb],mm);
%     % Mating between mm and md. Choose crossover
%     xp = ceil(rand*nv);
%     addpops = [ ];
%     for i = 1:mp
%         % Generate new value
%         pd = pops(indexf(md(i)),:);
%         pm = pops(indexf(mm(i)),:);
%         % Generate random beta
%         beta = rand;
%         popm(xp) = pm(xp)-beta*(pm(xp)-pd(xp));
%         popd(xp) = pd(xp)+beta*(pm(xp)-pd(xp));
%         if xp==nv
%             % Swop only to left
%             ch1 = [pm(1:nv-1),pd(nv)];
%             ch2 = [pd(1:nv-1),pm(nv)];
%         else
%             ch1 = [pd(1:xp),pm(xp+1:nv)];
%             ch2 = [pm(1:xp),pd(xp+1:nv)];
%         end
%         % New values introduced
%         ch1(xp) = popm(xp);
%         ch2(xp) = popd(xp);
%         addpops = [addpops;ch1;ch2];
%     end
%     % Add these ofspring to the best to obtain a new population
%     newpops = [ ];
%     newpops = [pops(indexf(1:nc-nb),:); addpops];
%     % Calculate number of mutations, mutation rate mu
%     Nmut = ceil(mu*nv*(nc-1));
%     % Choose location of variables to mutate
%     for k = 1:Nmut
%         mui = ceil(rand*nc);
%         muj = ceil(rand*nv);
%         if mui~=indexf(1)
%             newpops(mui,muj) = (b-a)*rand+a;
%         end
%     end
%     pops = newpops;
% end
```

```
% f = sfit(1);
% x = pops(indexf(1),:);
```

测试函数：$f(x_1,x_2)=(x_1^4-16x_1^2+5x_1)/2+(x_2^4-16x_2^2+5x_2)/2$

```
tf=@(x) 0.5*(x(1).^4-16*x(1).^2+5*x(1))+0.5*(x(2).^4-16*x(2).^2+5*x(2));

% 这个函数有若干局部极小，但全局最优解是(-2.9035,-2.9035).执行三次连续遗传算法：
[x,f]=contgaf(tf,2,[-4,4],50,50,0.2,0.6)
[x,f]=contgaf(tf,2,[-4,4],50,50,0.2,0.6)
[x,f]=contgaf(tf,2,[-4,4],50,50,0.2,0.6)
% 注意x值之间的差别，因为这个方法涉及随机元素，所以每次不会产生完全一样的结果。
```

```
x =

   -2.9035   -2.9035


f =

  -78.3323


x =

   -2.9035   -2.9035


f =

  -78.3323


x =

   -2.9035   -2.9035


f =

  -78.3323
```

求下述函数的最小值：$f(x)=\sum_{n=1}^{4}[100(x_{n+1}-x_n^2)^2+(1-x_n)^2$ 显然这个函数的最小值为0，在各个变量都为1处取得.

```
ff=@(x)(1-x(4))^2+(1-x(3))^2+(1-x(2))^2+(1-x(1))^2+...
    100*((x(5)-x(4)^2)^2+(x(4)-x(3)^2)^2+(x(3)-x(2)^2)^2+(x(2)-x(1)^2)^2);

[x,f]=contgaf(ff,5,[-5,5],20,100,0.15,0.6)
```

```
x =

    0.8012    0.8937    0.8998    0.9905    0.9509


f =
```

10.7871

模拟退火算法

由于问题的规模大，难度高，且其他方法都不太适合，可以考虑基于模拟退火的优化方法。 如果允许金属冷却得足够慢，它的冶金结构自然能够找到系统的最小能量态。寻找自然过程的 能量最小态过程，可以用来找既定的非线性函数的全局最优解。

缓慢冷却过程可以用能量态的玻尔兹曼概率分布来体现，这个分布函数在热力学中起着 重要作用，形式如下：

$P(E) = exp(-E/kT)$, 其中，E是指定的能量态，k是玻尔兹曼常数，T是温度，这个函数反映了冷却过程中能量水平的变化，最终会找到全局最小能量态。

设$f(x)$是需要最小化的非线性函数，其中x具有n个分量的向量，则

- 步骤1：令$k = 0, p = 0$，选择初始解x^k和任意初始温度T_p,
- 步骤2：x的新值x^{k+1}引起的变化为$\Delta f = f(x^{k+1}) - f(x^k)$; 则，若$\Delta f < 0$,以概率1接受这个变化，用$x^{k+1}$代替$x^k$，$k = k + 1$; 若$\Delta f \not> 0$,以概率$exp(-\Delta f/T_p)$接受这个变化，用$x^{k+1}$代替$x^k$，$k = k + 1$;
- 步骤3： 从步骤2开始重复，直到函数值没有明显变化。
- 步骤4： 用适当的降温过程$T_{p+1} = g(T_p)$降低温度，令$p = p + 1$，从步骤2 开始重复，直到函数值对降温没有明显的变化。

这个算法的关键难点是，选择初始温度和降温策略。

matlab函数asaq实现了上述算法的一个改进，带有某种淬火的指数冷却策略，加速算法 收敛，关键参数如，qf,tinit,maxstep以及变量的上下限都可以调整。

```
% function [fnew,xnew] = asaq(func,x,maxstep,qf,lb,ub,tinit)
% % Determines optimum of a function using simulated annealing.
% % Example call: [fnew,xnew] = asaq(func,x,maxstep,qf,lb,ub,tinit)
% % func is the function to be minimized, x the initial approx.
% % given as a column vector, maxstep the maximum number of main
% % iterations, qf the quenching factor in range 0 to 1,
% % Note: small value gives slow convergence, value close to 1 gives
% % fast convergence, but may not supply global optimum,
% % lb and ub are lower and upper bounds for the variables,
% % tinit is the intial temperature value.
% % Suggested values for maxstep = 200, tinit = 100, qf = 0.9.
%
% % This MATLAB function is to accompany 'Numerical Methods using MATLAB'
% % by GR Lindfield and JET Penny,
% % published by Academic Press, an imprint of Elsevier, 2012.
%
% xold = x;
% fold = feval(func,x);
% n = length(x);
% lk = n*10;
% % Quenching factor q
% q = qf*n;
% % c values estimated
% nv = log(maxstep*ones(n,1));
% mv = 2*ones(n,1);
% c = mv.*exp(-nv/n);
% % Set values for tk
% t0 = tinit*ones(n,1);
% tk = t0;
% % upper and lower bounds on x variables
% % variables assumed to lie between -100 and 100
% a = lb*ones(n,1);
% b = ub*ones(n,1);
```

```
% k = 1;
% % Main loop
% for mloop = 1:maxstep
%     for tempkloop = 1:lk
%         % Choose xnew as random neighbour
%         fold = feval(func,xold);
%         u = rand(n,1);
%         y = sign(u-0.5).*tk.*((1+ones(n,1)./tk).^(abs((2*u-1))-1));
%         xnew = xold+y.*(b-a);
%         fnew = feval(func,xnew);
%         % Test for improvement
%         if fnew <= fold
%             xold = xnew;
%         elseif exp((fold-fnew)/norm(tk))>rand
%             xold = xnew;
%         end
%     end
%     % Update tk values
%     tk = t0.*exp(-c.*k^(q/n));
%     k = k+1;
% end
% tf = tk;
```

解优化问题：$f(x_1,x_2)=(x_1^4-16x_1^2+5x_1)/2+(x_2^4-16x_2^2+5x_2)/2$

```
fv=@(x) 0.5*(x(1)^4-16*x(1)^2+5*x(1))+0.5*(x(2)^4-16*x(2)^2+5*x(2));
[fnew,xnew]=asaq(fv,[0,0].',200,0.9,-10,10,100)

% 每次运行都给出不同的结果
```

```
fnew =

  -64.1956


xnew =

   -2.9024
    2.7473
```

练习

1、利用函数optga在x=0到2内，最大化函数$y=1/[(x-1)^2+2]$.

2、利用模拟退火算法asaq最小化双变量函数$f=(x_1-1)^2+4(x_2+3)^2$,多运行几次 试试，看能够找出最优解。

3、利用模拟退火算法计算函数全局最小点
$f=0.5(x_1^4-16x_1^2+5x_1)+0.5(x_2^4-16x_2^2+5x_2)-10\cos[4(x_1+2.9035)]\cos[4(x_2+2.9035)]$正确结果应是$x_1=-2.9035,x_2=-2.9035$.

参考文献

1. Numerical Methods Using MATLAB，Third Edition，George Lindfield著，2018年