信陽師範学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

第6章 优化与规划问题

讲授人：牛言涛

日期：2020年3月19日

第6章 优化与规划问题知识结构图

运筹学（operational research）是一门解决一定约束条件下最优解的学科，应用现有的科学技术知识与数学手段，来解决实际生活之中的各种问题，是一门应用学科。运筹学分支还有规划论，排队论，图论，决策论等。

1. 遗传算法

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

- 遗传算法（Genetic Algorithm）是一类借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）演化而来的随机化搜索方法。
- 它是由美国的J.Holland教授1975年首先提出，其主要特点是直接对结构对象进行操作，不存在求导和函数连续性的限定；具有内在的隐并行性和更好的全局寻优能力；采用概率化的寻优方法，能自动获取和指导优化的搜索空间，自适应地调整搜索方向，不需要确定的规则。
- 遗传算法的这些性质，已被人们广泛地应用于组合优化、机器学习、信号处理、自适应控制和人工生命等领域。它是现代有关智能计算中的关键技术。 https://my.oschina.net/u/1412321/blog/192454

用遗传算法寻找函数的最优解：

- [x,fval] = ga(fitnessfcn, nvars, A, b, Aeq, beq, LB, UB, nonlcon, IntCon, options)
- fitnessfc为函数的句柄或者为匿名函数；
- IntCon指定决策变量是否为整数的向量，如[2,3]第二第三变量是整数；
- nvars表示自变量个个数。

1. 遗传算法

信阳师范学院
数学与统计学院

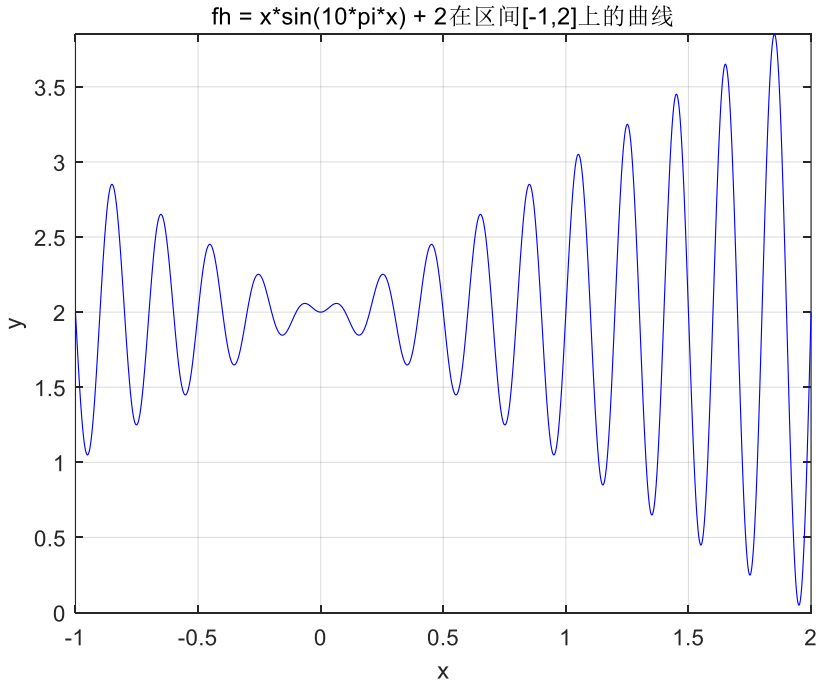
例1：考虑一个简单的一元函数最优化问题$f(x) = xsin10\pi x + 2, x \in (-1, 2)$，试求出$f(x)$取最大值时$x$的值。

从图中可以看出，该曲线为震荡曲线，存在很多极值点。

因为最优化工具箱的搜索函数需要给出初值，对不同的初值可能得出不同的搜索结果。

```
fh = @(x)-x.*sin(10*pi*x) - 2;
v = [];
xi = [-1:0.8:1.5,1.5:0.1:2];
for x0 = xi
    solx = fmincon(fh,x0,[],[],[],[],-1,2);
    v = [v;x0,solx,-fh(solx)];
end
```

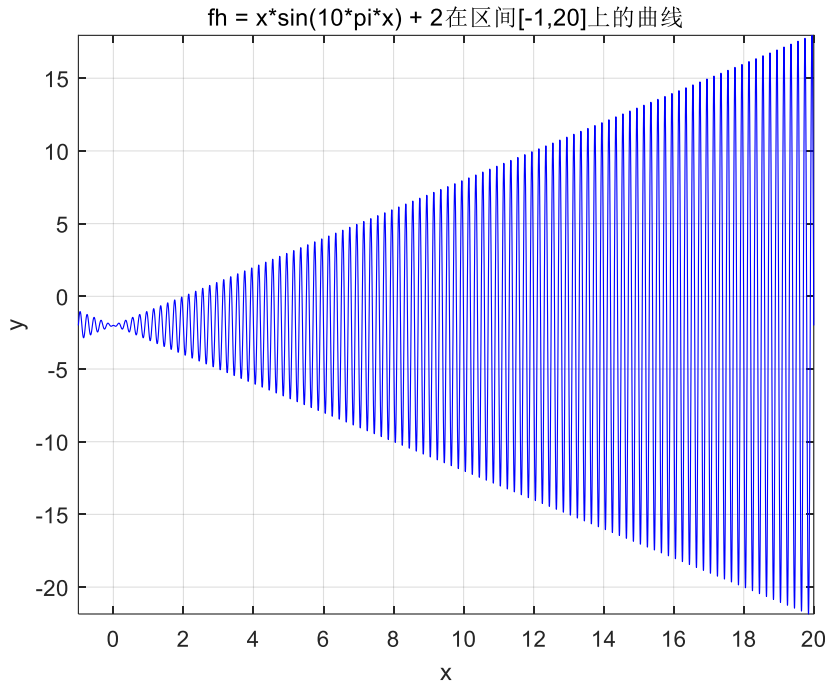
```
v =
   -1.0000   -0.4522    2.4511
   -0.2000   -0.2540    2.2520
    0.6000    0.8512    2.8506
    1.4000    1.6506    3.6503
    1.5000    0.2540    2.2520
    1.6000    1.6506    3.6503
    1.7000    1.0510    3.0505
    1.8000    1.8505    3.8503
    1.9000    0.4522    2.4511
    2.0000    1.2508    3.2504
```



1. 遗传算法

使用遗传算法，迭代64次，得到最优解

```
>> fh = @(x)-x.*sin(10*pi*x) - 2;
>> options = optimoptions('ga','Display','iter');
>> [x,fval] = ga(fh,1,[],[],[],[],-1,2,[],options)
x =
    1.8505
fval =
   -3.8503
```



考虑区间[-1,20]，则fmincon方法很难确定初值，使用ga算法

```
>> [x,fval] = ga(fh,1,[],[],[],[],-1,20,[],options)
x =
   18.4501
fval =
  -20.4500
```

1. 遗传算法

例2：求解混合整数非线性规划问题

$$\min\ f = 5x_4 + 6x_5 + 8x_6 + 10x_1 - 7x_3 - 18\ln(x_2+1) - 19.2\ln(x_1 - x_2 + 1) + 10$$

$$x.y\ \ s.t \begin{cases} 0.8\ln(x_2+1) + 0.96\ln(x_1 - x_2 + 1) - 0.8x_3 \le 0 \\ \ln(x_2+1) + 1.2\ln(x_1 - x_2 + 1) - x_3 - 2x_6 \ge -2 \\ x_2 - x_1 \le 0 \\ x_2 - 2x_4 \le 0 \\ x_1 - x_2 - 2x_5 \le 0 \\ x_4 + x_5 \le 1 \\ 0 \le x \le [2,2,1,1,1,1]^T \end{cases}$$

```
%非线性约束系数不等式定义
function [C,Ceq] = gacon_fun1(x)
   C = [0.8*log(x(2)+1)+0.96*log(x(1)-x(2)+1)-0.8*x(3);
       -log(x(2)+1)-1.2*log(x(1)-x(2)+1)+x(3)+2*x(6)-2];
   Ceq = [];
end
```

1. 遗传算法

```
fmin = @(x)5*x(4)+6*x(5)+8*x(6)+10*x(1)-7*x(3) - 18*log(x(2)+1)-19.2*log(x(1)-
x(2)+1)+10;
nvars = 6;   %决策变量的个数
intcon = [4,5,6];  %取整决策变量下标
ub = [2 2 1 1 1 1]';
lb = [0 0 0 0 0 0]';
A = [-1 1 0 0 0 0;0 1 0 -2 0 0;1 -1 0 0 -2 0;0 0 0 1 1 0];
b = [0 0 0 1]';
options = optimoptions('ga','Display','iter');
[x,fval] = ga(fmin,nvars,A,b,[],[],lb,ub,@gacon_fun1,intcon,options)
x =
    0.8020    0.8010    1.0000    1.0000         0         0
fval =
    5.4107
```

BNB20_new自定义函数求解结果：

```
fm =
    5.4198
x =
    0.8000
    0.8000
    1.0000
    1.0000
         0
         0
```

2. 粒子群算法

粒子群算法（Particle Swarm Optimization，PSO）是20世纪90年代兴起的一门学科，因其概念简明、实现方便、收敛速度快而为人所知。

粒子群算法的基本思想是模拟鸟群随机搜寻食物的捕食行为，鸟群通过自身经验和种群之间的交流调整自己的搜寻路径，从而找到食物最多的地点。其中每只鸟的位置/路径则为自变量组合，每次到达的地点的食物密度即函数值。每次搜寻都会根据自身经验（自身历史搜寻的最优地点）和种群交流（种群历史搜寻的最优地点）调整自身搜寻方向和速度，这个称为跟踪极值，从而找到最优解。

粒子群算法是一门新兴算法，此算法与遗传算法有很多相似之处，其收敛于全局最优解的概率很大。

①相较于传统算法计算速度非常快，全局搜索能力也很强；

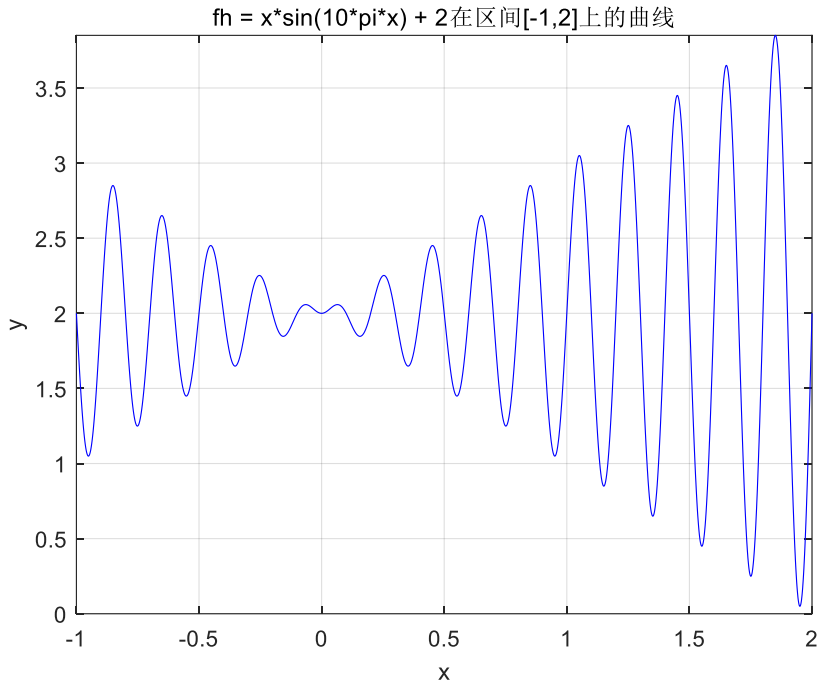
②PSO对于种群大小不十分敏感，所以初始种群设为500-1000，速度影响也不大；

③粒子群算法适用于连续函数极值问题，对于非线性、多峰问题均有较强的全局搜索能力。

2. 粒子群算法

求解函数：[x,fval,exitflag,output] = particleswarm(fun,nvars,lb,ub,options)

```
fh = @(x)-x.*sin(10*pi*x) - 2;
[x,fval,exitflag,output] = particleswarm(fh,1,-1,2)
  x = 1.8505
  fval = -3.8503
  exitflag = 1
  output =
    包含以下字段的 struct:
        rngstate: [1×1 struct]
        iterations: 41
        funccount: 420
         message: 'Optimization ended: relative change in the objective value ...'
```

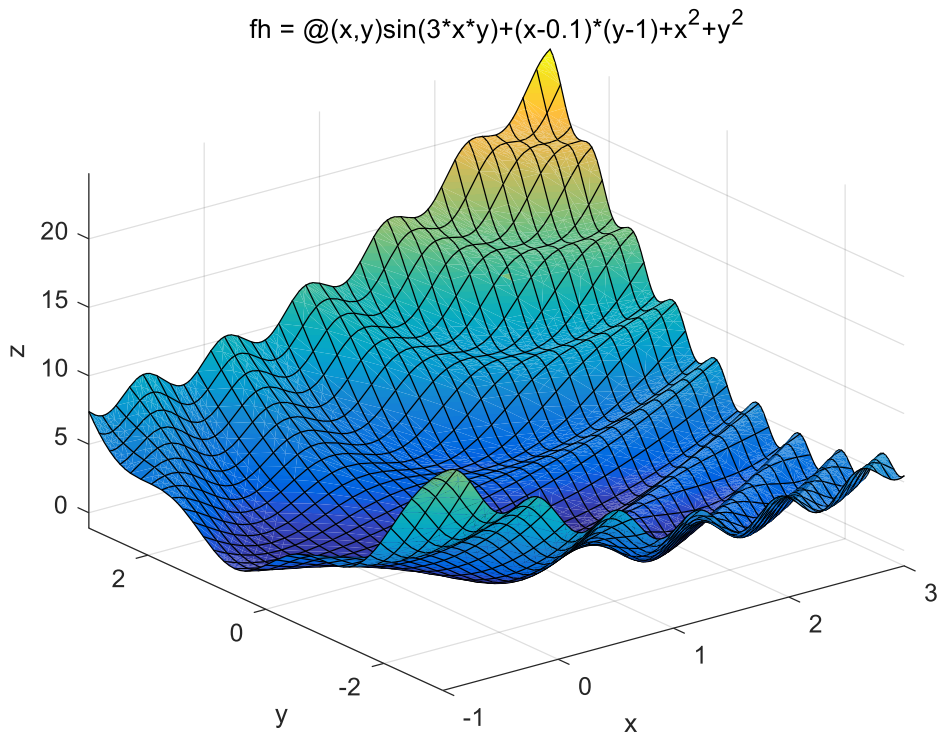


2. 粒子群算法

例3：试求下面最优化问题的全局最优解

$$f(x,y)=\sin(3xy)+(x-0.1)(y-1)+x^2+y^2,\ -1\le x\le 3,\ -3\le y\le 3,$$

```
>> fh = @(x)sin(3*x(1)*x(2))+(x(1)-0.1)*(x(2)-1)+x(1)^2+x(2)^2;
>> [x,fval,exitflag,output] = particleswarm(fh,2,[-1;-3],[3;3])
x =
    0.8768   -0.5503
fval =  -1.1251
exitflag = 1
output =
  包含以下字段的 struct:
      rngstate: [1×1 struct]
    iterations: 40
     funccount: 820
       message: 'Optimization ended: relative change in the objective value ...'
```



3. 模拟退火算法

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

模拟退火算法(Simulated Annealing，SA)最早的思想是由N. Metropolis [1] 等人于1953年提出。1983 年,S. Kirkpatrick 等成功地将退火思想引入到组合优化领域。

它是基于Monte-Carlo迭代求解策略的一种随机寻优算法，其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。模拟退火算法从某一较高初温出发，伴随温度参数的不断下降，结合概率突跳特性在解空间中随机寻找目标函数的全局最优解，即在局部最优解能概率性地跳出并最终趋于全局最优。模拟退火算法是一种通用的优化算法，理论上算法具有概率的全局优化性能，目前已在工程中得到了广泛应用，诸如VLSI、生产调度、控制工程、机器学习、神经网络、信号处理等领域。

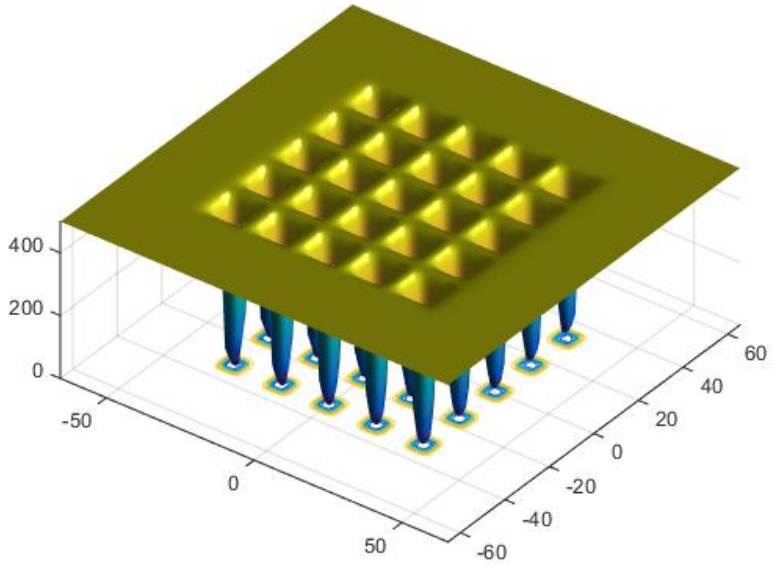
模拟退火算法是通过赋予搜索过程一种时变且最终趋于零的概率突跳性，从而可有效避免陷入局部极小并最终趋于全局最优的串行结构的优化算法。

3. 模拟退火算法

- 模拟退火算法MATLAB函数：
  - [x,fval,exitflag,output] = simulannealbnd(fun, x0, lb, ub, options)
- 例4：函数dejong5fcn

```
>> dejong5fcn
>> x0 = [0 0];
>>[x,fval] =
simulannealbnd(@dejong5fcn,x0)
x =
    0.0106  -31.9657
fval =
    2.9821
```

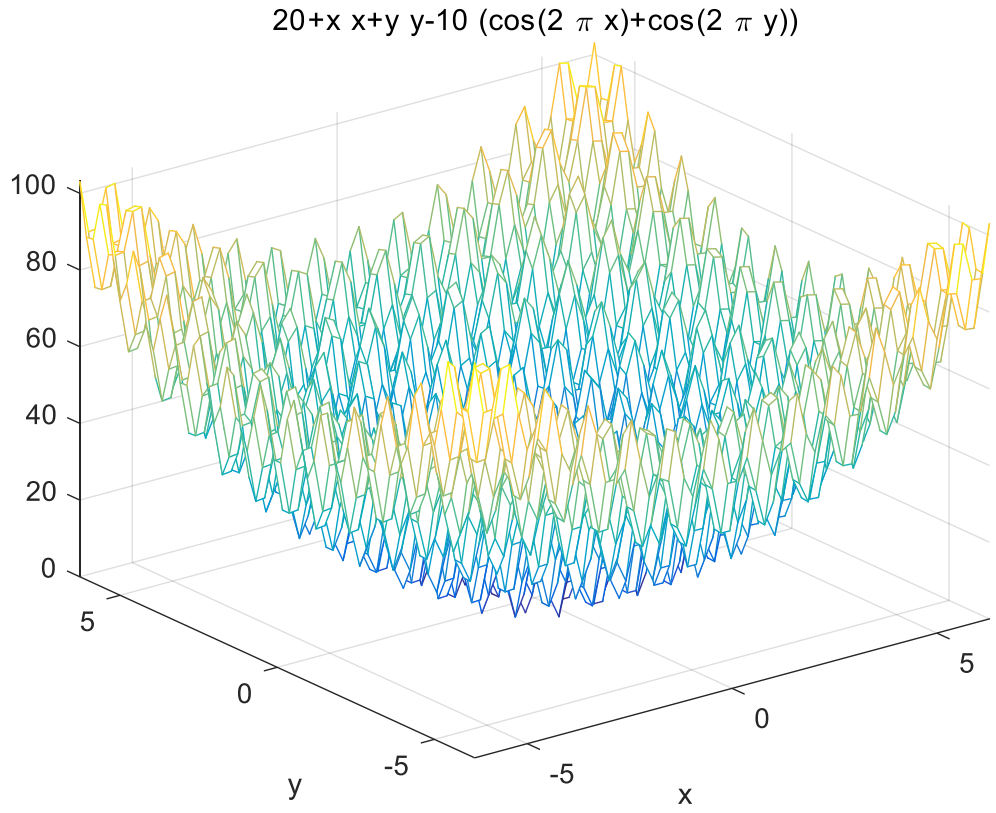
```
%限定变量上下限
>> x0 = [0 0];
>> lb = [-64 -64];
>> ub = [64 64];
>>[x,fval] =
simulannealbnd(@dejong5fcn,x0,lb,ub)
x =
  -31.9641  -32.0145
fval =
    0.9980
```



3. 模拟退火算法

例5：求函数最小值 $f(x,y)=20+x^2+y^2-10(\cos 2\pi x+\cos 2\pi y)$

```
>> ezmesh('20+x*x+y*y-10*(cos(2*pi*x)+cos(2*pi*y))')
>> fh = @(x)(20 + x(1)^2 + x(2)^2 - 10*(cos(2*pi*x(1)) + cos(2*pi*x(2))));
>> format long
>> [x,fval,exitflag] = simulannealbnd(fh,rand(1,2))
x =
  -0.000016202744096   0.994987808032280
fval =
   0.994959277907039
exitflag =
     1
```




01
IDEA
02
LEARNING
03
CREATIVE
INSPIRATION
EDUCATION
BUSINESS
PHOTO BOOK

感谢聆听