<http://hi.baidu.com/ycdoit/item/12d890df72bd5f4cdcf9be7f>

Matlab模拟退火算法——走过数模

模拟退火算法  
  
模拟退火算法来源于固体退火原理，将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。根据Metropolis准则，粒子在温度T时趋于平衡的概率为e-ΔE/(kT)，其中E为温度T时的内能，ΔE为其改变量，k为Boltzmann常数。用固体退火模拟组合优化问题，将内能E模拟为目标函数值f，温度T演化成控制参数t，即得到解组合优化问题的模拟退火算法：由初始解i和控制参数初值t开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减t值，算法终止时的当前解即为所得近似最优解，这是基于蒙特卡罗迭代求解法的一种启发式随机搜索过程。退火过程由冷却进度表(Cooling Schedule)控制，包括控制参数的初值t及其衰减因子Δt、每个t值时的迭代次数L和停止条件S。  
  
         模拟退火算法的模型  
  
模拟退火算法可以分解为解空间、目标函数和初始解三部分。  
  
              　模拟退火的基本思想:  
  
              　　(1) 初始化：初始温度T(充分大)，初始解状态S(是算法迭代的起点)， 每个T值的迭代次数L  
  
              　　(2) 对k=1，……，L做第(3)至第6步：  
  
              　　(3) 产生新解S′  
  
              　　(4) 计算增量Δt′=C(S′)-C(S)，其中C(S)为评价函数  
  
              　　(5) 若Δt′<0则接受S′作为新的当前解，否则以概率exp(-Δt′/T)接受S′作为新的当前解.  
  
              　　(6) 如果满足终止条件则输出当前解作为最优解，结束程序。  
  
    终止条件通常取为连续若干个新解都没有被接受时终止算法。  
  
              　　(7) T逐渐减少，且T->0，然后转第2步。  
  
              算法对应动态演示图：  
  
              模拟退火算法新解的产生和接受可分为如下四个步骤：  
  
              　　第一步是由一个产生函数从当前解产生一个位于解空间的新解；为便于后续的计算和接受，减少算法耗时，通常选择由当前新解经过简单地变换即可产生新解的方法，如对构成新解的全部或部分元素进行置换、互换等，注意到产生新解的变换方法决定了当前新解的邻域结构，因而对冷却进度表的选取有一定的影响。  
  
              　　第二步是计算与新解所对应的目标函数差。因为目标函数差仅由变换部分产生，所以目标函数差的计算最好按增量计算。事实表明，对大多数应用而言，这是计算目标函数差的最快方法。  
  
              　　第三步是判断新解是否被接受,判断的依据是一个接受准则，最常用的接受准则是Metropo1is准则:  
  
              若Δt′<0则接受S′作为新的当前解S，否则以概率exp(-Δt′/T)接受S′作为新的当前解S。  
  
              　　第四步是当新解被确定接受时，用新解代替当前解，这只需将当前解中对应于产生新解时的变换部分予以实现，同时修正目标函数值即可。此时，当前解实现了一次迭代。可在此基础上开始下一轮试验。而当新解被判定为舍弃时，则在原当前解的基础上继续下一轮试验。  
  
              　　模拟退火算法与初始值无关，算法求得的解与初始解状态S(是算法迭代的起点)无关；模拟退火算法具有渐近收敛性，已在理论上被证明是一种以概率l  
  
              收敛于全局最优解的全局优化算法；模拟退火算法具有并行性。  
  
              模拟退火算法的简单应用  
  
              　　作为模拟退火算法应用，讨论货郎担问题(Travelling Salesman Problem，简记为TSP)：  
  
设有n个城市，用数码1,…,n代表。城市i和城市j之间的距离为d(i，j)  
  
i, j=1,…,n．TSP问题是要找遍访每个域市恰好一次的一条回路，且其路径总长度为最短.。  
  
              　　求解TSP的模拟退火算法模型可描述如下：  
  
              　　解空间 解空间S是遍访每个城市恰好一次的所有回路，是{1，……，n}的所有循环排列的集合，S中的成员记为(w1,w2 ,……，wn)，并记wn+1= w1。初始解可选为(1，……，n)目标函数 此时的目标函数即为访问所有城市的路径总长度或称为代价函数： 　　我们要求此代价函数的最小值。  
  
              　　新解的产生 随机产生1和n之间的两相异数k和m，若k<m，则将  
  
              　　(w1, w2 ,…，wk , wk+1 ,…，wm ,…，wn)  
  
              　　变为：  
  
              　　(w1, w2 ,…，wm , wm-1 ,…，wk+1 , wk ,…，wn).  
  
              　　如果是k>m，则将  
  
              　　(w1, w2 ,…，wk , wk+1 ,…，wm ,…，wn)  
  
              　　变为：  
  
              　　(wm, wm-1 ,…，w1 , wm+1 ,…，wk-1 ,wn , wn-1 ,…，wk).  
  
              　　上述变换方法可简单说成是“逆转中间或者逆转两端”。  
  
              　　也可以采用其他的变换方法，有些变换有独特的优越性，有时也将它们交替使用，得到一种更好方法。  
  
              　　代价函数差 设将(w1, w2 ,……，wn)变换为(u1, u2 ,……，un), 则代价函数差为：  
  
             根据上述分析，可写出用模拟退火算法求解TSP问题的伪程序：  
  
              Procedure TSPSA:  
  
              　begin  
  
              　　init-of-T; { T为初始温度}  
  
              　　S={1，……，n}; {S为初始值}  
  
              　　termination=false;  
  
              　　while termination=false  
  
              　　　begin  
  
              　　　　for i=1 to L do  
  
              　　　　　　begin  
  
              　　　　　　　　generate(S′form S); { 从当前回路S产生新回路S′}  
  
              　　　　　　　　Δt:=f(S′))-f(S);{f(S)为路径总长}  
  
              　　　　　　　　IF(Δt<0) OR (EXP(-Δt/T)>Random-of-[0,1])  
  
              　　　　　　　　S=S′;  
  
              　　　　　　　　IF the-halt-condition-is-TRUE THEN  
  
              　　　　　　　　termination=true;  
  
              　　　　　　End;  
  
              　　　　T\_lower;  
  
              　　　End;  
  
              　End  
  
              　　模拟退火算法的应用很广泛，可以较高的效率求解最大截问题(Max Cut Problem)、0-1背包问题(Zero One  
  
              Knapsack Problem)、图着色问题(Graph Colouring Problem)、调度问题(Scheduling  
  
              Problem)等等。  
  
  
  
              ----------------------------------------------  
  
  
  
                              
  
              模拟退火算法的参数控制问题  
  
              　　模拟退火算法的应用很广泛，可以求解NP完全问题，但其参数难以控制，其主要问题有以下三点：  
  
              　　(1) 温度T的初始值设置问题。  
  
              　　温度T的初始值设置是影响模拟退火算法全局搜索性能的重要因素之一、初始温度高，则搜索到全局最优解的可能性大，但因此要花费大量的计算时间；反之，则可节约计算时间，但全局搜索性能可能受到影响。实际应用过程中，初始温度一般需要依据实验结果进行若干次调整。  
  
(2) 退火速度问题。  
  
模拟退火算法的全局搜索性能也与退火速度密切相关。一般来说，同一温度下的“充分”搜索(退火)是相当必要的，但这需要计算时间。实际应用中，要针对具体问题的性质和特征设置合理的退火平衡条件。  
  
(3) 温度管理问题。  
  
     　温度管理问题也是模拟退火算法难以处理的问题之一。实际应用中，由于必须考虑计算复杂度的切实可行性等问题，常采用如下所示的降温方式：  
  
T(t+1)＝k×T(t)  
  
              式中k为正的略小于1.00的常数，t为降温的次数。  
**Matlab源代码：**  
main.m  
zuobiao=[0.37 0.75 0.45 0.76 0.71 0.07 0.42 0.59 0.32 0.6 0.3 0.67 0.62 0.67 0.20 ...  
    0.35 0.27 0.94 0.82 0.37 0.61 0.42 0.6 0.39 0.53 0.4 0.63 0.5 0.98 0.68;  
    0.91 0.87 0.85 0.75 0.72 0.74 0.71 0.69 0.64 0.64 0.59 0.59 0.55 0.55 0.5...  
    0.45 0.43 0.42 0.38 0.27 0.26 0.25 0.23 0.19 0.19 0.13 0.08 0.04 0.02 0.85]  
plot(zuobiao(1,:),zuobiao(2,:),'g\*'),hold on  
plot(zuobiao(1,:),zuobiao(2,:))  
length=max(size(zuobiao));  
%求初始距离..  
  
zhixu=randperm(length)      %随机生成一个路线经过点的顺序  
temp=zuobiao(1,:);  
newzuobiao(1,:)=temp(zhixu);  
temp=zuobiao(2,:);  
newzuobiao(2,:)=temp(zhixu);  
newzuobiao  
f=juli(newzuobiao)  
  
  
%参数定义区--------------------------------------  
%初始温度为10000  
tmax=100;  
tmin=0.001;  
%温度下降速率  
down=0.95;  
  
  
%退火算法的函数..  
figure  
t=tmax;  
while t>tmin  
      
    for n=1:500  
        newzuobiao=newpath(zuobiao,length);  
        newf=juli(newzuobiao);  
        if newf<f  
           zuobiao=newzuobiao;  
           f=newf;  
        elseif   rand<exp(-(newf-f)/t)  
           zuobiao=newzuobiao;  
           f=newf;  
        end  
    end  
    huatu=[zuobiao,zuobiao(:,1)];  
    plot(huatu(1,:),huatu(2,:)),hold on  
    plot(huatu(1,:),huatu(2,:),'ro'),hold off  
    pause(0.00001)  
    t=t\*down  
    f  
end  
  
  
  
newpath.m  
function zuobiao=newpath(zuobiao,length)  
%随机交换两个点的坐标..  
a=ceil(rand(1,2)\*length);  
qian=a(1);  
hou=a(2);  
temp=zuobiao(:,qian);  
zuobiao(:,qian)=zuobiao(:,hou);  
zuobiao(:,hou)=temp;  
  
juli.m  
function lucheng=juli(zuobiao)  
length=max(size(zuobiao));  
s=0;  
for i=2:length  
    s=s+sqrt(sum((zuobiao(:,i)-zuobiao(:,i-1)).^2));  
end  
if length~=2  
    s=s+sqrt(sum((zuobiao(:,1)-zuobiao(:,length)).^2));  
end  
    lucheng=s;



