# 模拟退火算法解决TSP问题

## 解决该问题注意要点

1. 新解的产生：
   1. 二变换法
      1. 产生两个不相等的随机数，直接调换其位置的值
   2. 三变换法
      1. 产生三个不等的随机数，进行升序排序，将随机数1 2 之间的值移到随机数3之后。
2. 算法其余部分

其余部分使用模拟退火算法原理即可。

1. Matlab代码

clear all;

clc;

%读取数据

A1=xlsread('E:\顾子涵专用文件夹\学习\matlab学习\matlab与数学模型\TSP数据.xlsx',1,'A2:C19');

A2=xlsread('E:\顾子涵专用文件夹\学习\matlab学习\matlab与数学模型\TSP数据.xlsx',1,'D2:F19');

A3=xlsread('E:\顾子涵专用文件夹\学习\matlab学习\matlab与数学模型\TSP数据.xlsx',1,'G2:I17');

POINT=[A1;A2;A3];

%说明：第一列为城市编号，第二列为x坐标，第三列为y坐标

amout=size(POINT,1);%求列的长度

dist\_matrix=zeros(amout,amout);

sol\_first=1:amout;%产生初始解

%for i=1:amout

% for j=1:amout

% dist\_matrix(i,j)=distance(POINT(i,:),POINT(j,:));

% end

%end

POINT(:,1)=[];

x1=POINT(:,1)\*ones(1,amout);

x2=x1';

y1=POINT(:,2)\*ones(1,amout);

y2=y1';

dist\_matrix=sqrt((x1-x2).^2+(y1-y2).^2);

%※※※用向量计算距离矩阵，以减少matlab运算量

%说明距离矩阵POINT(1,2)表示城市1、2之间的距离

%POINT=POINT'

%模拟退火算法参数初始设定

lambda=0.95;%温度衰减系数

T0=100;%初始温度

Tf=3;%终结温度

Markov\_length=10000;%任意温度的迭代次数

%%%%%%%%%%%%%%%%%%%%%%%

T\_now=T0;

sol\_now=sol\_first;%算法执行前的初值赋予

best\_sol\_val=calc\_dist(sol\_first,dist\_matrix);%最好解

%执行模拟退火算法

while T\_now>=Tf

%产生新解，两种方式

for num=1:Markov\_length

if unifrnd(0,1)<0.5 %二变换法

while 1 %产生随机数直到不等为止

rnd(1)=ceil(unifrnd(0.5,amout-0.5));

rnd(2)=ceil(unifrnd(0.5,amout-0.5));

if rnd(1)~=rnd(2)

break;

end

end

%交换数据 即为二变换法

sol\_new=sol\_now;

temp=sol\_new(rnd(1));

sol\_new(rnd(1))=sol\_new(rnd(2));

sol\_new(rnd(2))=temp;

else %三变换法

while 1 %产生随机数，使得rnd1<=rnd2<rnd3

rnd(1)=ceil(unifrnd(0.5,amout-0.5));

rnd(2)=ceil(unifrnd(0.5,amout-0.5));

rnd(3)=ceil(unifrnd(0.5,amout-0.5));

rnd=sort(rnd);

if rnd(2)~=rnd(3)

break;

end

end

%开始三变换法

add=rnd(2)-rnd(1)+1;

%申请新数组用于变换

sol\_new=sol\_now;

temp\_arr1=sol\_new(rnd(1):rnd(2));

temp\_arr2=sol\_new;

sol\_new(rnd(1):rnd(1)+rnd(3)-rnd(2)-1)=temp\_arr2(rnd(2)+1:rnd(3));

sol\_new(rnd(1)+rnd(3)-rnd(2):rnd(1)+rnd(3)-rnd(2)-1+add)=temp\_arr1;

%变换完毕

end

%产生新解完毕

%计算新解的距离

new\_sol\_val=calc\_dist(sol\_new,dist\_matrix);

now\_sol\_val=calc\_dist(sol\_now,dist\_matrix);

if(new\_sol\_val<now\_sol\_val)

%新解的距离较短

%无条件接受

sol\_now=sol\_new;

now\_sol\_val=new\_sol\_val;

%记录此值

if now\_sol\_val<best\_sol\_val

best\_sol\_val=now\_sol\_val;

sol\_best=sol\_now;

end

else

%根据蒙特卡洛方法概率性接受

p=exp(-abs(new\_sol\_val-now\_sol\_val)/T\_now);

temp\_rnd=unifrnd(0,1);

if(temp\_rnd<p)

%概率性接受，但不计入最大值

sol\_now=sol\_new;

now\_sol\_val=new\_sol\_val;

else

sol\_new=sol\_now;

end

end

end

T\_now=T\_now\*lambda;

%T\_now=0;

%%%%%%%%

end

disp('最优方案：');

disp(sol\_best);

disp('距离');

disp(best\_sol\_val);

function sum=calc\_dist(sol,dist\_matrix)

%计算这个解要走的距离的函数

sum=0;

for i=1:size(sol,2)-1

sum=sum+dist\_matrix(sol(i),sol(i+1));

end

sum=sum+dist\_matrix(sol(size(sol,2)),sol(1));

end

function d=distance(p1,p2)

%计算两城市之间距离的函数

d=sqrt((p1(2)-p2(2))^2+(p1(3)-p2(3))^2);

end

1. 结果



结果分析：官方给出的最短距离为7542米。这个值与官方的值相差很小。可以认为是成功计算的。若要算法再精确，就需要更改退火速率和最低温度。但这会使程序变慢。

1. 运行效率

由于matlab运行效率远不如c语言。但编程简单。因此，我们需要了解如何提高matlab运行效率的方法。

1. 如何提高运行效率
   1. 循环矢量化
   2. 尽量避免使用循环
   3. 在使用数组或矩阵之前定义其维数
   4. 对矩阵元素使用下标或者索引操作
   5. 尽量多使用函数文件，少使用脚本文件

**（过一段时间对这个程序进行优化，提高运行速率）**