智能计算第六次实验报告

180111234 金凯

一、实验目的及要求

理解并掌握蚁群算法，并利用蚁群算法求解TSP问题，并能用此寻找指定函数在指定区间内的最大值。

二、实验内容

**1.蚁群算法求解TSP问题**

北京等六城市的距离表如下。请据此表求解TSP（Travelling Salesman Problem） 问题。即给定N个城市，从某一城市开始走遍所有城市但不许重复，最后回到出发点，要求总路径最短。

六城市间距离表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 上海 | 天津 | 石家庄 | 太原 | 呼和浩特 |
| 北京 | 1078 | 119 | 263 | 398 | 401 |
| 上海 | 0 | 963 | 989 | 1096 | 1381 |
| 天津 |  | 0 | 262 | 426 | 504 |
| 石家庄 |  |  | 0 | 171 | 394 |
| 太原 |  |  |  | 0 | 341 |

**2.下面展示了13个城市之间的距离，现要求从城市A出发，找一条最短的旅游顺序，使得游览所有城市后回到城市A。**

**表2 13个城市间距离表（单位km）**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | I | J | K | L | M |
| A | 0 | 2.2 | 18.1 | 4.8 | 9.2 | 8.4 | 5.2 | 0.4 | 9.3 | 11.3 | 10.2 | 0.4 | 10.4 |
| B | 2.2 | 0 | 17.5 | 2.8 | 7.9 | 6.6 | 3.3 | 1.8 | 8.5 | 9.8 | 8.7 | 2.4 | 9.6 |
| C | 18.1 | 17.5 | 0 | 14.7 | 9.6 | 10.9 | 14.2 | 18.2 | 9.1 | 7.6 | 8.7 | 18.3 | 8.0 |
| D | 4.8 | 2.8 | 14.7 | 0 | 5.4 | 3.8 | 0.5 | 4.4 | 6.4 | 7.0 | 6.0 | 5.1 | 7.4 |
| E | 9.2 | 7.9 | 9.6 | 5.4 | 0 | 2.4 | 4.9 | 9.0 | 1.7 | 2.3 | 1.2 | 9.5 | 2.2 |
| F | 8.4 | 6.6 | 10.9 | 3.8 | 2.4 | 0 | 3.3 | 8.1 | 4.1 | 3.3 | 2.5 | 8.7 | 4.6 |
| G | 5.2 | 3.3 | 14.2 | 0.5 | 4.9 | 3.3 | 0 | 4.8 | 6.0 | 6.6 | 5.5 | 5.5 | 6.9 |
| H | 0.4 | 1.8 | 18.2 | 4.4 | 9.0 | 8.1 | 4.8 | 0 | 9.2 | 11.1 | 10.0 | 0.6 | 10.3 |
| I | 9.3 | 8.5 | 9.1 | 6.4 | 1.7 | 4.1 | 6.0 | 9.2 | 0 | 3.3 | 2.5 | 9.6 | 1.1 |
| J | 11.3 | 9.8 | 7.6 | 7.0 | 2.3 | 3.3 | 6.6 | 11.1 | 3.3 | 0 | 1.2 | 11.7 | 2.8 |
| K | 10.2 | 8.7 | 8.7 | 6.0 | 1.2 | 2.5 | 5.5 | 10.0 | 2.5 | 1.2 | 0 | 10.5 | 2.5 |
| L | 0.4 | 2.4 | 18.3 | 5.1 | 9.5 | 8.7 | 5.5 | 0.6 | 9.6 | 11.7 | 10.5 | 0 | 10.7 |
| M | 10.4 | 9.6 | 8.0 | 7.4 | 2.2 | 4.6 | 6.9 | 10.3 | 1.1 | 2.8 | 2.5 | 10.7 | 0 |

**3.****寻找函数在最大值。**

三、实验基本原理

3.1 TSP问题描述

TSP问题是典型的组合优化问题。给定个城市以及各个城市之间的距离，要求找到一条遍历所有城市且每个城市只被访问一次的路线，并使得总路线距离最短。其可用数学式来表达：

设有一个城市集合，其中每对城市之间的距离。求一对经过中每个城市一次的路线。使

其中是的一个置换。

3.2 蚁群算法概述

类似于粒子群算法，蚁群算法也是通过对群体智能的模拟而实现的，即模拟蚂蚁群体觅食为主的群智能算法。

在自然界中，对于觅食的蚂蚁群体，其可以在任何和没有提示的情况下找到食物和巢穴之间的最短路径。并且能够根据和环境的变迁，自适应地找到新的最优路径。根据生物学家研究，蚂蚁群体这一行为的根本原因是：蚂蚁在寻找食物的过程中，能在其走过的路径上释放一种特殊的物质----信息素，随着时间的推移，这种信息素会逐渐地挥发，而对于后来的蚂蚁，选择某条路径的概率与该路径上信息素的浓度成正比。当某一条路径上通过的蚂蚁越多的时候，这条路径上的信息素的浓度就会累积越大，后来的蚂蚁选择此路径的概率也就越大。路径上蚂蚁越多，导致信息素浓度越高，从而会吸引更多的蚂蚁，从而形成一种正反馈机制，通过这种机制，最终蚁群可以发现最短路径。

蚁群算法本质上也是概率算法，通过大概率收敛到最佳值。蚁群分泌的信息素存在正反馈，使最优解有大概率被找寻到，当全局都选用较佳的解，既可以得到整体的最优解。

3.3符号说明

|  |  |
| --- | --- |
|  | 蚂蚁数量 |
|  | 蚂蚁编号 |
|  | 时间 |
|  | 城市数 |
|  | 城市（i，j）之间的距离 |
|  | 启发式因子 |
|  | 边上的信息素量 |
|  | 本次迭代边上的信息素增量 |
|  | 第k只蚂蚁在本次迭代中留在边上的信息素量 |
|  | 信息素挥发系数 |
|  | 持久性系数 |
|  | 时刻t蚂蚁k由城市i转移到城市j的概率 |
|  | 蚂蚁k的禁忌表 |

3.4涉及的公式

转移概率公式

启发因子计算公式

3.5蚁群算法步骤

**注：**1.概率选择受信息素浓度和启发函数影响，启发函数为距离的倒数

2.信息素会随时间挥发，故加入了挥发因子使其更贴近现实

Step1：初始化各个参数

此处的参数包括但不限于点的距离，信息素的初始浓度，蚂蚁数量，信息素挥发因子，信息素和启发函数的重要度因子，启发函数，最大迭代次数等。

Step2：迭代

对每只蚂蚁随机指定初始值，再根据概率选择出每只蚂蚁的路径，确定每只蚂蚁的路径总长度。

Step3：更新信息素

对信息素进行更新确定蚁群的最佳路径长度和平均长度。

Step4： 展示结果

展示出最佳路径以及最佳路径对迭代的变化图。

3.4 蚁群算法流程图

图示

描述已自动生成

四、程序简述

4.1 用蚁群算法求解TSP问题

4.1.1问题的求解

Step1：设定参数

编写city.m随机生成6（13）个城市的位置，并将该位置保存到citys.mat方便调取数据。

由于题目中已给出城市间的相互距离表，添加缺少的数据后写出距离矩阵D

Step2：初始化

初始化的参数包括蚁群规模、信息素因子、启发函数因子、信息素挥发因子、信息素常数、最大迭代次数等

Step3

随机将蚂蚁放入不同的出发点，对每个蚂蚁计算其下个访问城市，直到有蚂蚁访问完所有城市。

Step4

计算各蚂蚁经过的路径长度，并记录当前迭代次数最优解，并对路径上信息素浓度进行更新.

Step5：

判断是否达到最大迭代次数，若否返回Step3

Step6：绘图和结果显示

绘出城市之间的路径连线图，并标出最短路径，出发起点和终点，为使分析清晰可辨，还绘出了各个迭代图显示的最短路径方便分析使用。

4.2用遗传算法求解TSP问题

4.2.1问题的求解

算法的流程与3.5一致，其与4.1属于同一问题，不做过多解释。

4.3寻找函数在特定区间的最大值

4.3.1问题的求解

已知这样的一个函数

Step1：绘图

先画出其在区间[0,4]内的图像，即

图表, 折线图

描述已自动生成

此时需要寻找这个函数在定义域内的最高点（最大值）。这里使用的是蚁群算法解决该问题。

Step1：初始化参数。

Step2: 初始化蚁群，第一代蚁群随机分布在可行域中**。**

Step3: 初始化信息素，第一代信息素采用第一代蚁群的函数值表示，函数值越大，信息素越多。

Step4: 状态转移，计算状态转移概率，根据状态转移概率进行局部搜索或全局搜索。

Step5：约束边界。

Step6:选择，根据目标函数值在原始蚁群和状态转移之后的蚁群之间进行选择。

Step7：更新信息素

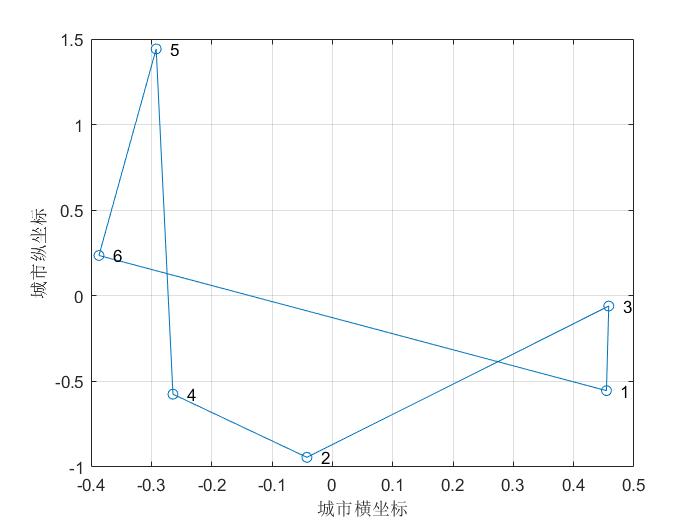
Step8：对选择后的蚁群重复进行状态转移、约束边界和更新信息素3步，直至结束。

五、实验结果与分析

5.1用蚁群算法求解TSP问题

对于city.mat文件中的城市序列，参数在代码注释中写的较为详细，在此不再赘述。

运行代码后得到如下结果：



上图为粒子群算法优化路径示意图，横坐标和纵坐标是随机生成的城市位置。

图表

描述已自动生成

上图为经过迭代后形成的最短路径长度，可以发现一开始就得出了最短路径长度。

图表

中度可信度描述已自动生成

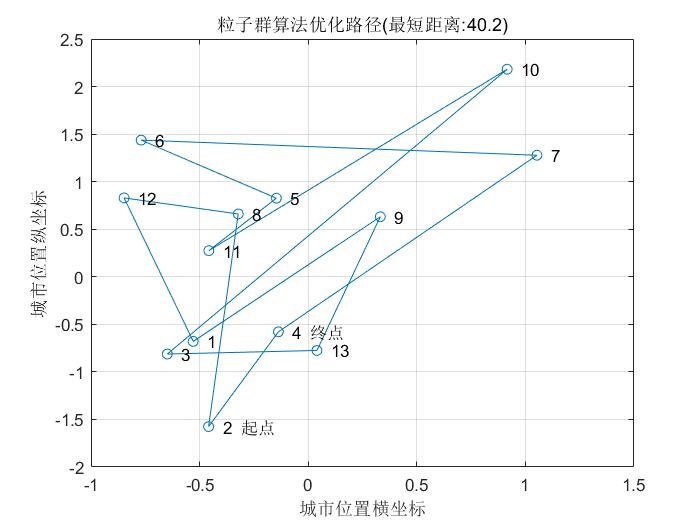
最短路径为2984

最终路径：4,5,6,1,3,2，（3）[4为起点，2为终点]

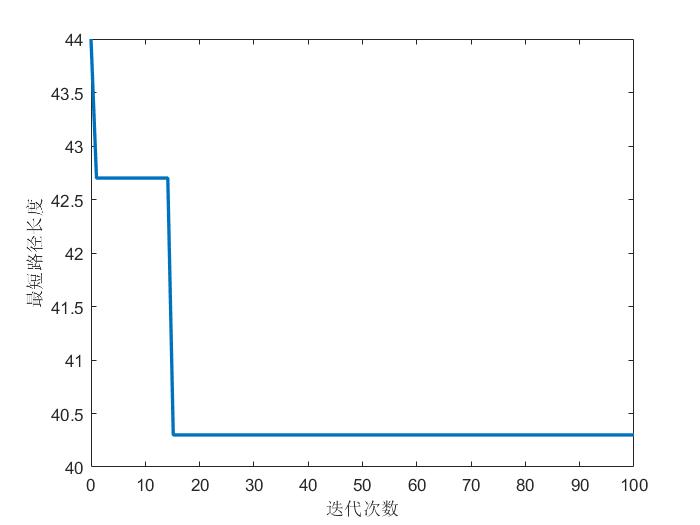
注：1-北京，2-上海，3-天津，4-石家庄，5-太原，6-呼和浩特

5.2用粒子群解决TSP问题（13个城市）

继续沿用上图代码，并加做参数上的修改，



上图为蚁群算法优化路径示意图，横坐标和纵坐标是随机生成的城市位置。



上图为经过各代最短距离与平均距离的对比图，可以清晰的发现在迭代10多次左右趋于平稳

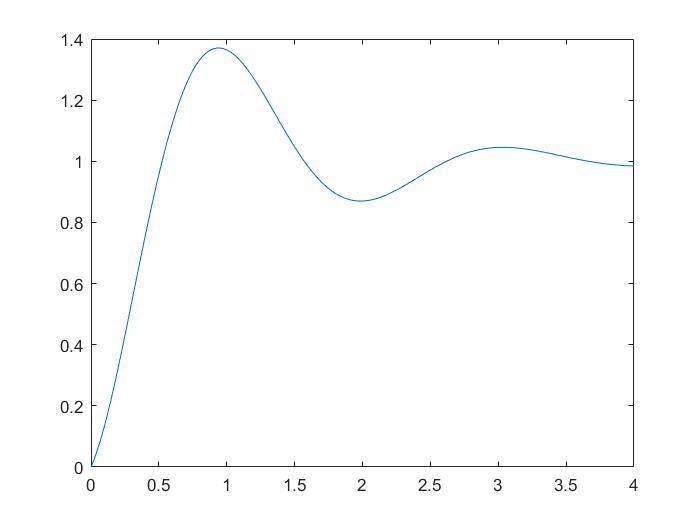
图表, 散点图

描述已自动生成

最优路径解3 10 11 5 6 7 4 2 8 1 12 9 13，得到的最优路程解的最小路程为40.3

5.3寻求函数在特定区间的最大值

该函数在区间[0,4]之间的图像：



运行6.3的实验代码后，得到如下结果

图表, 折线图

描述已自动生成

图片包含 图形用户界面

描述已自动生成

可以得出当x=1时得最大值1.3642

六、实验代码

**6.1用**蚁群**算法求解TSP问题**

**Citys.m**

N=6; %%城市的个数

citys=randn(N,2); %%随机生成城市位置

save citys.mat %%将城市位置保存到citys.mat

clc

clear all

load citys.mat; %坐标矩阵

n=6; %n表示城市个数

D= [0,1078,119,263,398,401;

1078,0,963,989,1096,1381;

119,963,0,262,426,504;

263,989,262,0,171,394;

398,1096,426,171,0,341;

401,1381,504,394,341,0];

**%%==================蚁群算法实现过程======================================================**

**%%============== 第一步 变量初始化==============**

**iter\_max=100; %最大迭代次数**

m=30; % 蚂蚁个数

Alpha=1; % 表征信息素重要程度的参数

Beta=5; % 表征启发式因子重要程度的参数

Rho=0.8; % 信息素蒸发系数

Q=10; % 信息素增加强度系数

Eta=1./D; % Eta为能见度因数，这里设为距离的倒数

Tau=ones(n,n); % Tau为信息素矩阵，初始化全为1

Tabu=zeros(m,n); % 存储并记录路径的生成

nC=1; % 迭代计数器

R\_best=zeros(iter\_max,n); %各代最短路线，行为最大迭代次数，列为城市个数

L\_best=inf.\*ones(iter\_max,1);%%各代最短路线的长度，inf为无穷大

L\_ave=zeros(iter\_max,1); % 各代平均路线长度

**%%============== 第二步 将m只蚂蚁放到城市上==============**

while nC<=iter\_max %停止条件之一：达到最大迭代次数

Randpos=[];

for i=1:(ceil(m/n)) %ceil表示向无穷方向取整

Randpos=[Randpos,randperm(n)]; %randperm(n)：表示随机产生一个整数排列

end

Tabu(:,1)=(Randpos(1,1:m))'; %每只蚂蚁（m只）都对应有一个位置，Tabu(:,1)为每只蚂蚁走过的第一个城市

**%% ============== 第三步 m只蚂蚁按概率函数选择下一座城市，完成各自的周游==============**

for j=2:n %城市从第二个开始

for i=1:m

visited=Tabu(i,1:(j-1)); %已访问的城市

J=zeros(1,(n-j+1)); %待访问的城市

P=J; %待访问城市的选择概率分布（初始化）

Jc=1; %循环下标

for k=1:n %利用循环求解待访问城市，如果第k个城市不属于已访问城市，则其为待访问城市

if length(find(visited==k))==0

J(Jc)=k;

Jc=Jc+1; %下表加1，便于下一步存储待访问的城市

end

end

for k=1:length(J) % 下面计算待访问城市的概率分布，length(J)表示待访问城市个数

P(k)=(Tau(visited(end),J(k))^Alpha)\*(Eta(visited(end),J(k))^Beta); %概率计算公式中的分子

end

P=P/(sum(P)); %概率分布：长度为待访问城市个数

Pcum=cumsum(P); %求累积概率和：cumsum（[1 2 3])=1 3 6,目的在于使得Pcum的值总有大于rand的数

Select=find(Pcum>=rand); %按概率选取下一个城市：当累积概率和大于给定的随机数，则选择求和被加上的最后一个城市作为即将访问的城市

if isempty(Select) %若选择城市为空集，则随机将任一城市加入禁忌表中

Tabu(i,j)=round(1+(n-1)\*rand);

else

next\_visit=J(Select(1)); %next\_visit表示即将访问的城市

Tabu(i,j)=next\_visit; %将访问过的城市加入禁忌表中

end

end

end

if nC>=2;Tabu(1,:)=R\_best(nC-1,:);end %若迭代次数大于等于2，则将上一次迭代的最佳路线存入到Tabu的第一行中

**%% ==============第四步 记录本次迭代最佳路线==============**

L=zeros(m,1);

for i=1:m;

R=Tabu(i,:);

for j=1:(n-1)

L(i)=L(i)+D(R(j),R(j+1)); %求路径距离

end

L(i)=L(i)+D(R(1),R(n)); %加上最后一个城市与第一个城市之间的距离

end

L\_best(nC)=min(L); %最优路径为距离最短的路径

pos=find(L==L\_best(nC)); %找出最优路径对应的位置：即为哪只蚂蚁

R\_best(nC,:)=Tabu(pos(1),:); %确定最优路径对应的城市顺序

L\_ave(nC)=mean(L); %求第k次迭代的平均距离

nC=nC+1;

**%% ==============第五步 更新信息素，此处蚁周系统==============**

Delta\_Tau=zeros(n,n); %Delta\_Tau(i,j)表示所有蚂蚁留在第i个城市到第j个城市路径上的信息素增量

for i=1:m

for j=1:(n-1) %建立了完整路径后在释放信息素

Delta\_Tau(Tabu(i,j),Tabu(i,j+1))=Delta\_Tau(Tabu(i,j),Tabu(i,j+1))+Q/L(i);

end

Delta\_Tau(Tabu(i,n),Tabu(i,1))=Delta\_Tau(Tabu(i,n),Tabu(i,1))+Q/L(i);

end

Tau=(1-Rho).\*Tau+Delta\_Tau; %信息素更新公式

**%% ==============第六步 禁忌表清零==============**

Tabu=zeros(m,n);

end

**%% ==============第七步 输出结果==============**

Pos=find(L\_best==min(L\_best)); %找到L\_best中最小值所在的位置

Shortest\_Route=R\_best(Pos(1),:) %提取最短路径

Shortest\_Length=L\_best(Pos(1)) %提取最短路径长度

**%% ==============作图==============**

figure(1) %作迭代收敛曲线图

x=linspace(0,iter\_max,iter\_max);

y=L\_best(:,1);

plot(x,y,'-','LineWidth',2);

xlabel('迭代次数'); ylabel('最短路径长度');

figure(2) %作最短路径图

Shortest\_Route=[Shortest\_Route Shortest\_Route(1)];

plot([citys(Shortest\_Route,1)],[citys(Shortest\_Route,2)],'o-');

grid on

for i = 1:size(citys,1)

text(citys(i,1),citys(i,2),[' ' num2str(i)]);

end

xlabel('城市横坐标'); ylabel('城市纵坐标');

**6.2求解TSP问题，13个城市**

**代码同上。改了6.1的参数与距离矩阵**

D=[ 0 2.2 18.1 4.8 9.2 8.4 5.2 0.4 9.3 11.3 10.2 0.4 10.4;

2.2 0 17.5 2.8 7.9 6.6 3.3 1.8 8.5 9.8 8.7 2.4 9.6;

18.1 17.5 0 14.7 9.6 10.9 14.2 18.2 9.1 7.6 8.7 18.3 8.0;

4.8 2.8 14.7 0 5.4 3.8 0.5 4.4 6.4 7.0 6.0 5.1 7.4;

9.2 7.9 9.6 5.4 0 2.4 4.9 9.0 1.7 2.3 1.2 9.5 2.2;

8.4 6.6 10.9 3.8 2.4 0 3.3 8.1 4.1 3.3 2.5 8.7 4.6;

5.2 3.3 14.2 0.5 4.9 3.3 0 4.8 6.0 6.6 5.5 5.5 6.9;

0.4 1.8 18.2 4.4 9.0 8.1 4.8 0 9.2 11.1 10.0 0.6 10.3;

9.3 8.5 9.1 6.4 1.7 4.1 6.0 9.2 0 3.3 2.5 9.6 1.1;

11.3 9.8 7.6 7.0 2.3 3.3 6.6 11.1 3.3 0 1.2 11.7 2.8;

10.2 8.7 8.7 6.0 1.2 2.5 5.5 10.0 2.5 1.2 0 10.5 2.5;

0.4 2.4 18.3 5.1 9.5 8.7 5.5 0.6 9.6 11.7 10.5 0 10.7;

10.4 9.6 8.0 7.4 2.2 4.6 6.9 10.3 1.1 2.8 2.5 10.7 0];

**6.3求解已知函数在指定区间的最大值**

**%% i. 绘制目标函数曲线图**

x=0:0.01:4;

y=1-cos(3\*x).\*exp(-x);

figure

plot(x, y)

hold on

function sants = edgeselection(ants, tau, P0, lamda, xl, xu, yl, yu)

sants = ants;

% 计算状态转移概率

[taubest, ~] = max(tau);

p = abs((taubest - tau) / taubest);

lsindex = find(p < P0);

gsindex = find(p >= P0);

% 局部搜索

r = rand(length(lsindex), 2);

sants(lsindex, :) = sants(lsindex, :) + (2 .\* r - 1) .\* lamda;

% 全局搜索

r = rand(length(gsindex), 2);

gedge = repmat([xu-xl, yu-yl], length(gsindex), 1);

sants(gsindex,:) = sants(gsindex,:) + gedge .\* (r - 0.5);

% 约束边界

sants(sants(:, 1) < xl, 1) = xl;

sants(sants(:, 1) > xu, 1) = xu;

sants(sants(:, 2) < yl, 2) = yl;

sants(sants(:, 2) > yu, 2) = yu;

% 选择

objvalue = calObjFun(ants);

sobjvalue = calObjFun(sants);

tindex = find(sobjvalue < objvalue);

sants(tindex, :) = ants(tindex, :);

function objval = calObjFun(X)

% 计算目标函数值

% X input 点输入 [x, y] nx2

% objvalue output 输出 nx1

objval = 1-cos(3\*x(:,1)).\*exp(-x(:,2));

function ants = initant(num, xl, xu, yl, yu)

% 初始化蚁群

ants = rand(num, 2);

ants(:,1) = xl + (xu - xl) .\* ants(:,1);

ants(:,2) = yl + (yu - yl) .\* ants(:,2);

主函数

clear;

clc;

Ant = 300; % 蚂蚁数量

Times = 80; % 蚂蚁移动次数

Rho = 0.9; % 信息素挥发系数

P0 = 0.2; % 转移概率常数

xl = 0; xu = 4; % 设置搜索范围

yl = -1; yu = 3;

step = 0.05;

ants = initant(Ant, xl, xu, yl, yu); % 初始化蚁群

tau = calObjFun(ants); % 计算初代信息素

firstants = ants;

for t = 1:Times

ants = edgeselection(ants, P0, tau, 1/t, xl, xu, yl, yu); % 转移+约束

tau = (1 - Rho) .\* tau + calObjFun(ants); % 更新信息素

end

figure(1);

plotobjfun(xl, xu, yl, yu, step);

hold on;

plot3(firstants(:,1), firstants(:,2), calObjFun(firstants), 'b\*');

hold on;

plot3(ants(:,1), ants(:,2), calObjFun(ants), 'r\*');

hold off;