a = 0.99; %温度衰减函数的参数

t0 = 97; %初始温度

tf = 3; %终止温度

t = t0;

Markov\_length = 10000; %Markov链长度

coordinates = [

1 565.0 575.0; 2 25.0 185.0; 3 345.0 750.0;

4 945.0 685.0; 5 845.0 655.0; 6 880.0 660.0;

7 25.0 230.0; 8 525.0 1000.0; 9 580.0 1175.0;

10 650.0 1130.0; 11 1605.0 620.0; 12 1220.0 580.0;

13 1465.0 200.0; 14 1530.0 5.0; 15 845.0 680.0;

16 725.0 370.0; 17 145.0 665.0; 18 415.0 635.0;

19 510.0 875.0; 20 560.0 365.0; 21 300.0 465.0;

22 520.0 585.0; 23 480.0 415.0; 24 835.0 625.0;

25 975.0 580.0; 26 1215.0 245.0; 27 1320.0 315.0;

28 1250.0 400.0; 29 660.0 180.0; 30 410.0 250.0;

31 420.0 555.0; 32 575.0 665.0; 33 1150.0 1160.0;

34 700.0 580.0; 35 685.0 595.0; 36 685.0 610.0;

37 770.0 610.0; 38 795.0 645.0; 39 720.0 635.0;

40 760.0 650.0; 41 475.0 960.0; 42 95.0 260.0;

43 875.0 920.0; 44 700.0 500.0; 45 555.0 815.0;

46 830.0 485.0; 47 1170.0 65.0; 48 830.0 610.0;

49 605.0 625.0; 50 595.0 360.0; 51 1340.0 725.0;

52 1740.0 245.0;

];

coordinates(:,1) = []; %一个城市一行

amount = size(coordinates,1); %城市的数目

%通过向量化的方法计算距离矩阵

dist\_matrix = zeros(amount,amount);

coor\_x\_tmp1 = coordinates(:,1) \* ones(1,amount);

coor\_x\_tmp2 = coor\_x\_tmp1';

coor\_y\_tmp1 = coordinates(:,2) \* ones(1,amount);

coor\_y\_tmp2 = coor\_y\_tmp1';

dist\_matrix = sqrt((coor\_x\_tmp1 - coor\_x\_tmp2).^2 + (coor\_y\_tmp1 - coor\_y\_tmp2).^2);

sol\_new = 1:amount; %产生初始解，sol\_new是每次产生的新解

sol\_current = sol\_new; %sol\_current是当前解

sol\_best = sol\_new; %sol\_best是冷却中的最好解

E\_current = inf; %E\_current是当前解对应的回路距离

E\_best = inf; %E\_best是最优解

p = 1;

while t >= tf

for r = 1:Markov\_length %Markov链长度

%产生随机扰动

if(rand < 0.5)

%两交换

ind1 = 0;

ind2 = 0;

while(ind1 == ind2)

ind1 = ceil(rand \* amount);

ind2 = ceil(rand \* amount);

end

tmp1 = sol\_new(ind1);

sol\_new(ind1) = sol\_new(ind2);

sol\_new(ind2) = tmp1;

else

%三交换

ind1 = 0;

ind2 = 0;

ind3 = 0;

while( (ind1 == ind2) || (ind1 == ind3) || (ind2 == ind3) || (abs(ind1 -ind2) == 1) )

ind1 = ceil(rand \* amount);

ind2 = ceil(rand \* amount);

ind3 = ceil(rand \* amount);

end

tmp1 = ind1;

tmp2 = ind2;

tmp3 = ind3;

%确保 ind1 < ind2 < ind3

if(ind1 < ind2) && (ind2 < ind3);

elseif(ind1 < ind3) && (ind3 < ind2)

ind1 = tmp1; ind2 = tmp3; ind3 = tmp2;

elseif(ind2 < ind1) && (ind1 < ind3)

ind1 = tmp2; ind2 = tmp1; ind3 = tmp3;

elseif(ind2 < ind3) && (ind3 < ind1)

ind1 = tmp2; ind2 = tmp3; ind3 = tmp1;

elseif(ind3 < ind1) && (ind1 < ind2)

ind1 = tmp3; ind2 = tmp1; ind3 = tmp2;

elseif(ind3 < ind2) && (ind2 < ind1)

ind1 = tmp3; ind2 = tmp2; ind3 = tmp1;

end

tmplist1 = sol\_new((ind1 + 1):(ind2 - 1));

sol\_new((ind1 + 1):(ind1 + (ind3 - ind2 + 1) )) = sol\_new((ind2):(ind3));

sol\_new((ind1 + (ind3 - ind2 + 1) + 1):(ind3)) = tmplist1;

end

%检查是否满足约束

%计算目标函数值（即内能）

E\_new = 0;

for i = 1:(amount - 1)

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

end

%再算上从最后一个城市到第一个城市的距离

E\_new = E\_new + dist\_matrix(sol\_new(amount),sol\_new(1));

if E\_new < E\_current

E\_current = E\_new;

sol\_current = sol\_new;

if E\_new < E\_best

E\_best = E\_new;

sol\_best = sol\_new;

end

else

%若新解的目标函数值大于当前解，

%则仅以一定概率接受新解

if rand < exp(-(E\_new - E\_current) / t)

E\_current = E\_new;

sol\_current = sol\_new;

else

sol\_new = sol\_current;

end

end

end

t = t \* a; %控制参数t（温度）减少为原来的a倍

end

disp('最优解为:');

disp(sol\_best);

disp('最短距离:');

disp(E\_best);