遗传算法 GA-最短路问题

模型建立：

（1）编码

采用整数排列编码方法。对于 n 个城市的 TSP 问题，将染色体分为 n 段，其中每一段为对应城市的编号。

（2）种群初始化

在完成染色体编码以后，必须产生一个初始种群作为起始解，所以首先需要决定初始化种群的数目。初始化种群的数目一般根据经验得到，一般情况下种群的数量视城市规模的大小而确定，其取值在 50～200 之间浮动。

（3）适应度函数

设 $k_1|k_2|\cdots|k_i|\cdots|k_n|$ 为一个采用整数编码的染色体，$D_{k_ik_j}$ 为城市 k_i 到城市 k_j 的距离，则该个体的适应度为

$$\text{fitness} = \frac{1}{\sum_{i=1}^{n-1} D_{k_ik_j} + D_{k_nk_1}}$$

即适应度函数为恰好走遍 n 个城市，再回到出发城市的距离的倒数。优化的目标就是选择适应度函数值尽可能大的染色体，适应度函数值越大的染色体越优质，反之越劣质。

（4）选择操作

选择操作即从旧群体中以一定概率选择个体到新群体中，个体被选中的概率跟适应度值有关，个体适应度值越大，被选中的概率越大。

（5）交叉操作

采用部分映射杂交，确定交叉操作的父代，将父代样本两两分组，每组重复以下过程

①产生两个[1，n]区间内的随机整数 r_1 和 r_2，确定两个位置的中间数据进行交叉。

② 交叉后，同一个个体中有重复的城市编号，不重复的数字保留，有冲突的数字（带 * 位置）采用部分映射的方法消除冲突，即利用中间段的对应关系进行映射。

（6）交叉变异

变异策略采用随机选取两个点，将其位置对换。产生两个[1，n]范围内的随机整数 r_1 和 r_2，确定两个位置，将其位置对换。

（7）进化逆转操作

为改善遗传算法的局部搜索能力，在选择、交叉、变异之后引进连续多次的进化逆转操作。这里的“进化”是指逆转算子的单方向性，即只有经逆转后，适应度值有提高的才接受下来，否则逆转无效。

产生两个[1，n]区间内的随机整数 r_1 和 r_2，确定两个位置，将其位置对换。

对每个个体进行交叉变异，然后代入适应度函数进行评估，x 选择出适应值大的个体进行下一代的交叉和变异以及进化逆转操作。循环操作：判断是否满足设定的最大遗传代数 MAXGEN，不满足则跳入适应度值的计算；否则，结束遗传操作。

案例

本案例以 14 个城市为例，假定 14 个城市的位置坐标如表 4－1 所列。寻找出一条最短的遍历 14 个城市的路径。

表 4－1　14 个城市的位置坐标

城市编号	X 坐标	Y 坐标	城市编号	X 坐标	Y 坐标
1	16.47	96.10	8	17.20	96.29
2	16.47	94.44	9	16.30	97.38
3	20.09	92.54	10	14.05	98.12
4	22.39	93.37	11	16.53	97.38
5	25.23	97.24	12	21.52	95.59
6	22.00	96.05	13	19.41	97.13
7	20.47	97.02	14	20.09	92.55

附录	运行环境：Matlab2011a

GA_TSP.m（主程序，更改 X 矩阵即可）

```
clear
clc
close all
%% 加载数据
X=[16.47,96.10
      16.47,94.44
      20.09,92.54
      22.39,93.37
      25.23,97.24
      22.00,96.05
      20.47,97.02
      17.20,96.29
      16.30,97.38
      14.05,98.12
      16.53,97.38
      21.52,95.59
      19.41,97.13
      20.09,92.55];
D=Distanse(X);    %生成距离矩阵
N=size(D,1);       %城市个数
%%-------------------遗传参数-------------------------------
NIND=100;            %种群大小
MAXGEN=200;         %最大遗传代数
Pc=0.9;               %交叉概率
Pm=0.05;              %变异概率
GGAP=0.9;             %代沟
%% 初始化种群
Chrom=InitPop(NIND,N);
%% 画出随机解的路径图
DrawPath(Chrom(1,:),X)
pause(0.0001)
%% 输出随机解的路径和总距离
disp('初始种群中的一个随机值:')
OutputPath(Chrom(1,:));
Rlength=PathLength(D,Chrom(1,:));
```

```
disp(['总距离：',num2str(Rlength)]);
disp('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
%% 优化
gen=0;
figure;
hold on;box on
xlim([0,MAXGEN])
title('优化过程')
xlabel('代数')
ylabel('最优值')
ObjV=PathLength(D,Chrom);   %计算路径长度
preObjV=min(ObjV);
while gen<MAXGEN%循环
    %% 计算适应度
    ObjV=PathLength(D,Chrom);   %计算路径长度
    % fprintf('%d     %1.10f\n',gen,min(ObjV))
    line([gen-1,gen],[preObjV,min(ObjV)]);pause(0.0001)
    preObjV=min(ObjV);
    FitnV=Fitness(ObjV);
    %% 选择
    SelCh=Select(Chrom,FitnV,GGAP);
    %% 交叉操作
    SelCh=Recombin(SelCh,Pc);
    %% 变异
    SelCh=Mutate(SelCh,Pm);
    %% 逆转操作
    SelCh=Reverse(SelCh,D);
    %% 重插入子代的新种群
    Chrom=Reins(Chrom,SelCh,ObjV);
    %% 更新迭代次数
    gen=gen+1 ;
end
%% 画出最优解的路径图
ObjV=PathLength(D,Chrom);   %计算路径长度
[minObjV,minInd]=min(ObjV);
DrawPath(Chrom(minInd(1),:),X)
%% 输出最优解的路径和总距离
disp('最优解:')
p=OutputPath(Chrom(minInd(1),:));
disp(['总距离：',num2str(ObjV(minInd(1)))]);
disp('-------------------------------------------------------------')
```

Distanse.m

```
%% 计算两两城市之间的距离
%输入 a   各城市的位置坐标
%输出 D   两两城市之间的距离
function D=Distanse(a)
row=size(a,1);
D=zeros(row,row);
for i=1:row
    for j=i+1:row
        D(i,j)=((a(i,1)-a(j,1))^2+(a(i,2)-a(j,2))^2)^0.5;
        D(j,i)=D(i,j);
    end
end
```

InitPop.m

```
%% 初始化种群
%输入：
% NIND：种群大小
% N：      个体染色体长度（这里为城市的个数）
%输出：
%初始种群
function Chrom=InitPop(NIND,N)
Chrom=zeros(NIND,N);%用于存储种群
for i=1:NIND
    Chrom(i,:)=randperm(N);%随机生成初始种群
End
```

DrawPath.m

```
%% 画路径函数
%输入
% Chrom   待画路径
% X         各城市坐标位置
function DrawPath(Chrom,X)
R=[Chrom(1,:) Chrom(1,1)]; %一个随机解(个体)
figure;
hold on
plot(X(:,1),X(:,2),'o','color',[0.5,0.5,0.5])
plot(X(Chrom(1,1),1),X(Chrom(1,1),2),'rv','MarkerSize',20)
for i=1:size(X,1)
    text(X(i,1)+0.05,X(i,2)+0.05,num2str(i),'color',[1,0,0]);
end
A=X(R,:);
row=size(A,1);
for i=2:row
    [arrowx,arrowy] = dsxy2figxy(gca,A(i-1:i,1),A(i-1:i,2));%坐标转换
    annotation('textarrow',arrowx,arrowy,'HeadWidth',8,'color',[0,0,1]);
end
hold off
xlabel('横坐标')
ylabel('纵坐标')
title('轨迹图')
box on
```

dsxy2figxy.m

```
function varargout = dsxy2figxy(varargin)
if length(varargin{1}) == 1 && ishandle(varargin{1}) ...
                                    && strcmp(get(varargin{1},'type'),'axes')
    hAx = varargin{1};
    varargin = varargin(2:end);
else
    hAx = gca;
end;
if length(varargin) == 1
    pos = varargin{1};
else
    [x,y] = deal(varargin{:});
end
axun = get(hAx,'Units');
set(hAx,'Units','normalized');
axpos = get(hAx,'Position');
axlim = axis(hAx);
```

```
axwidth = diff(axlim(1:2));
axheight = diff(axlim(3:4));
if exist('x','var')
    varargout{1} = (x - axlim(1)) * axpos(3) / axwidth + axpos(1);
    varargout{2} = (y - axlim(3)) * axpos(4) / axheight + axpos(2);
else
    pos(1) = (pos(1) - axlim(1)) / axwidth * axpos(3) + axpos(1);
    pos(2) = (pos(2) - axlim(3)) / axheight * axpos(4) + axpos(2);
    pos(3) = pos(3) * axpos(3) / axwidth;
    pos(4) = pos(4) * axpos(4 )/ axheight;
    varargout{1} = pos;
end
set(hAx,'Units',axun)
```

OutputPath.m

```
%% 输出路径函数
%输入：R 路径
function p=OutputPath(R)
R=[R,R(1)];
N=length(R);
p=num2str(R(1));
for i=2:N
    p=[p,'—>',num2str(R(i))];
end
disp(p)
```

PathLength.m

```
%% 计算各个体的路径长度
% 输入：
% D       两两城市之间的距离
% Chrom 个体的轨迹
function len=PathLength(D,Chrom)
[row,col]=size(D);
NIND=size(Chrom,1);
len=zeros(NIND,1);
for i=1:NIND
    p=[Chrom(i,:) Chrom(i,1)];
    i1=p(1:end-1);
    i2=p(2:end);
    len(i,1)=sum(D((i1-1)*col+i2));
end
```

Fitness.m

```
%% 适配值函数
%输入：
%个体的长度（TSP 的距离）
%输出：
%个体的适应度值
function FitnV=Fitness(len)
FitnV=1./len;
```

Select.m

```
%% 选择操作
%输入
%Chrom 种群
%FitnV 适应度值
%GGAP：代沟
%输出
%SelCh   被选择的个体
```

```
function SelCh=Select(Chrom,FitnV,GGAP)
NIND=size(Chrom,1);
NSel=max(floor(NIND*GGAP+.5),2);
ChrIx=Sus(FitnV,NSel);
SelCh=Chrom(ChrIx,:);
```

Sus.m

```
% 输入:
%FitnV    个体的适应度值
%Nsel     被选择个体的数目
% 输出:
%NewChrIx    被选择个体的索引号
function NewChrIx = Sus(FitnV,Nsel)
[Nind,ans] = size(FitnV);
cumfit = cumsum(FitnV);
trials = cumfit(Nind) / Nsel * (rand + (0:Nsel-1)');
Mf = cumfit(:, ones(1, Nsel));
Mt = trials(:, ones(1, Nind))';
[NewChrIx, ans] = find(Mt < Mf & [ zeros(1, Nsel); Mf(1:Nind-1, :) ] <= Mt);
[ans, shuf] = sort(rand(Nsel, 1));
NewChrIx = NewChrIx(shuf);
```

Recombin.m

```
%% 交叉操作
% 输入
%SelCh    被选择的个体
%Pc        交叉概率
%输出：
% SelCh  交叉后的个体
function SelCh=Recombin(SelCh,Pc)
NSel=size(SelCh,1);
for i=1:2:NSel-mod(NSel,2)
    if Pc>=rand %交叉概率 Pc
        [SelCh(i,:),SelCh(i+1,:)]=intercross(SelCh(i,:),SelCh(i+1,:));
    end
end
%输入：
%a 和 b 为两个待交叉的个体
%输出：
%a 和 b 为交叉后得到的两个个体
function [a,b]=intercross(a,b)
L=length(a);
r1=randsrc(1,1,[1:L]);
r2=randsrc(1,1,[1:L]);
if r1~=r2
    a0=a;b0=b;
    s=min([r1,r2]);
    e=max([r1,r2]);
    for i=s:e
        a1=a;b1=b;
        a(i)=b0(i);
        b(i)=a0(i);
        x=find(a==a(i));
        y=find(b==b(i));
        i1=x(x~=i);
        i2=y(y~=i);
        if ~isempty(i1)
```

```
            a(i1)=a1(i);
        end
        if ~isempty(i2)
            b(i2)=b1(i);
        end
    end
end
```

Mutate.m

```
%% 变异操作
%输入：
%SelCh   被选择的个体
%Pm       变异概率
%输出：
% SelCh  变异后的个体
function SelCh=Mutate(SelCh,Pm)
[NSel,L]=size(SelCh);
for i=1:NSel
    if Pm>=rand
        R=randperm(L);
        SelCh(i,R(1:2))=SelCh(i,R(2:-1:1));
    end
end
```

Reverse.m

```
%% 进化逆转函数
%输入
%SelCh  被选择的个体
%D       个城市的距离矩阵
%输出
%SelCh   进化逆转后的个体
function SelCh=Reverse(SelCh,D)
[row,col]=size(SelCh);
ObjV=PathLength(D,SelCh);   %计算路径长度
SelCh1=SelCh;
for i=1:row
    r1=randsrc(1,1,[1:col]);
    r2=randsrc(1,1,[1:col]);
    mininverse=min([r1 r2]);
    maxinverse=max([r1 r2]);
    SelCh1(i,mininverse:maxinverse)=SelCh1(i,maxinverse:-1:mininverse);
end
ObjV1=PathLength(D,SelCh1);   %计算路径长度
index=ObjV1<ObjV;
SelCh(index,:)=SelCh1(index,:);
```

Reins.m

```
%% 重插入子代的新种群
 %输入：
 %Chrom   父代的种群
 %SelCh   子代种群
 %ObjV    父代适应度
 %输出
 % Chrom   组合父代与子代后得到的新种群
function Chrom=Reins(Chrom,SelCh,ObjV)
NIND=size(Chrom,1);
NSel=size(SelCh,1);
```

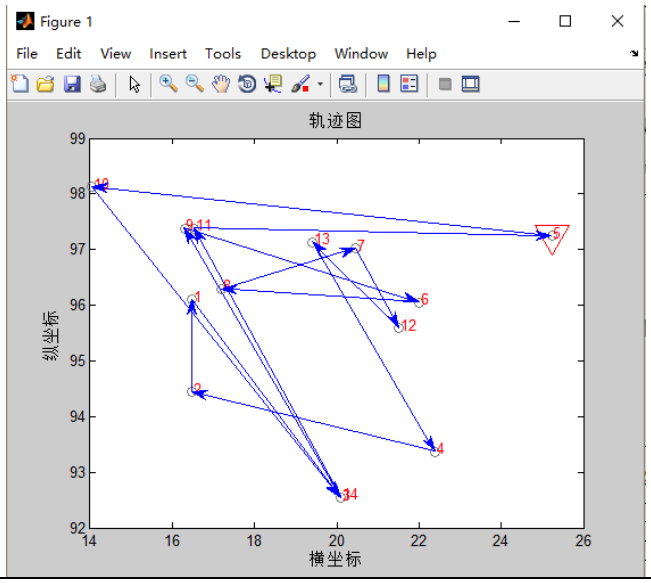
```
[TobjV,index]=sort(ObjV);
Chrom=[Chrom(index(1:NIND-NSel),:);SelCh];
```

运行结果

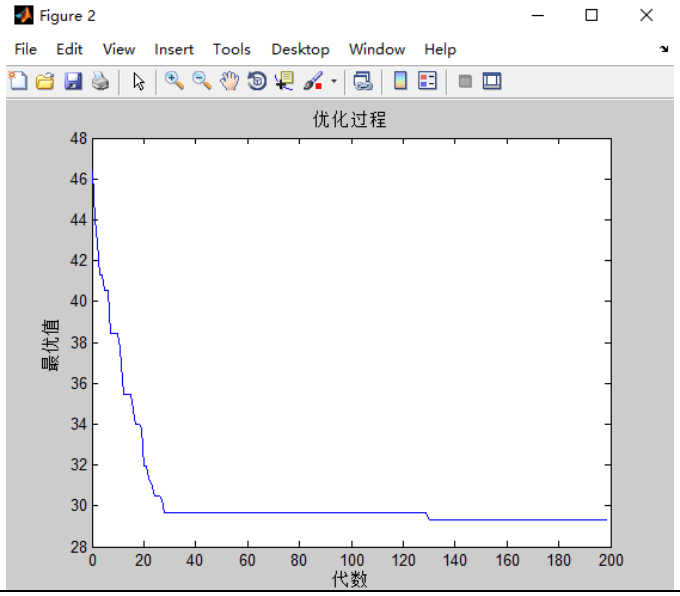
```
初始种群中的一个随机值:
5->10->3->9->6->8->7->12->13->4->2->1->14->11->5
总距离: 76.226
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

最优解:
5->4->3->14->2->1->10->9->11->8->13->7->12->6->5
总距离: 29.3405
```

初始随机路线



遗传过程



最优路线

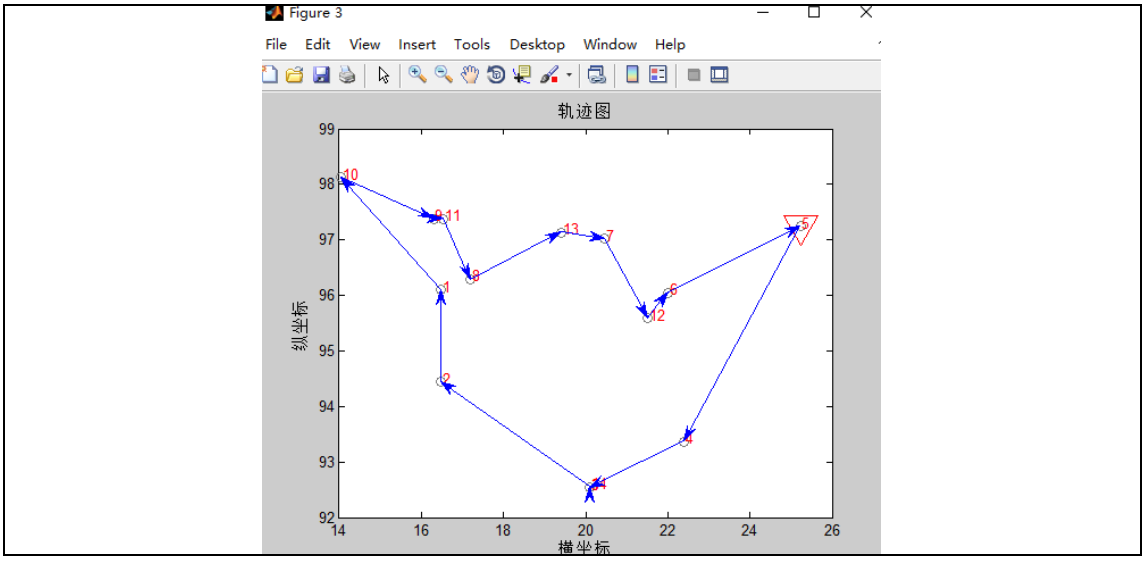


Figure 3
File Edit View Insert Tools Desktop Window Help
轨迹图
纵坐标
横坐标
99
98
97
96
95
94
93
92
14
16
18
20
22
24
26
10
11
8
13
7
5
1
6
12
2
4
3