报告题目：基于 Matlab 的遗传算法解决 TSP 问题

说明：该文包括了基于 Matlab 的遗传算法解决 TSP 问题的基本说明，并在文后附录了实现该算法的所有源代码。此代码经过本人的运行，没有发现错误，结果比较接近理论最优值，虽然最优路径图有点交叉。

因为本人才疏学浅，本报告及源代码的编译耗费了本人较多的时间与精力，特收取下载积分，还请见谅。若有什么问题，可以私信，我们共同探讨这一问题。

希望能对需要这方面的知识的人有所帮助！

1.问题介绍

旅行商问题(Traveling Salesman Problem，简称TSP)是一个经典的组合优化问题。它可以描述为：一个商品推销员要去若干个城市推销商品，从一个城市出发，需要经过所有城市后，回到出发地，应如何选择行进路线，以使总行程最短。从图论的角度看，该问题实质是在一个带权完全无向图中。找一个权值最小的Hemilton回路。其数学描述为：设有一个城市集合其中每对城市之间的距离 $d\left(c_i,c_j\right)\in R^+$，求一对经过C中每个城市一次的路线 $\left(c_{\Pi_1},c_{\Pi_2},\cdots c_{\Pi n}\right)$ 使

$$\min\sum_{i=1}^{n-1}d\left(c_{\Pi i},c_{\Pi(i+1)}\right)+d\left(c_{\Pi n},c_{\Pi_1}\right)$$

其中 $\Pi_1,\Pi_2,\cdots\Pi_n$ 是 $(1,2\cdots n)$ 的一个置换。

2.遗传算法

2.1 遗传算法基本原理

遗传算法是由美国 J. Holland 教授于 1975 年在他的专著《自然界和人工系统的适应性》中首先提出的，它是一类借鉴生物界自然选择和自然遗传机制的随机化搜索算法。

遗传算法模拟自然选择和自然遗传过程中发生的繁殖、交叉和基因突变现象，在每次迭代中都保留一组候选解，并按某种指标从解群中选取较优的个体，利用遗传算子(选择、交叉和变异)对这些个体进行组合，产生新一代的候选解群，重复此过程，直到满足某种收敛指标为止。

遗传算法，在本质上是一种不依赖具体问题的直接搜索方法，是一种求解问题的高效并行全局搜索方法。遗传算法在模式识别、神经网络、图像处理、机器学习、工业优化控制、自适应控制、负载平衡、电磁系统设计、生物科学、社会科学等方面都得到了应用。在人工智能研究中，现在人们认为“遗传算法、自适应系统、细胞自动控制、混沌理论与人工智能一样，都是对今后十年的计算技术有重大影响的关键技术”。

2.2 遗传算法的流程

标准的遗传算法包括群体的初始化，选择，交叉，变异操作。流程图如图 1 所示，其主要步骤可描述如下：

（1）随机产生一组初始个体构成的初始种群，并评价每一个个体的适配值。

（2）判断算法的收敛准则是否满足。若满足输出搜索结果；否则执行以下步骤。

（3）根据适配值大小以一定方式执行选择操作。
（4）按交叉概率 Pc 执行交叉操作。
（5）按变异概率 Pm 执行变异操作。
（6）返回步骤（2）。

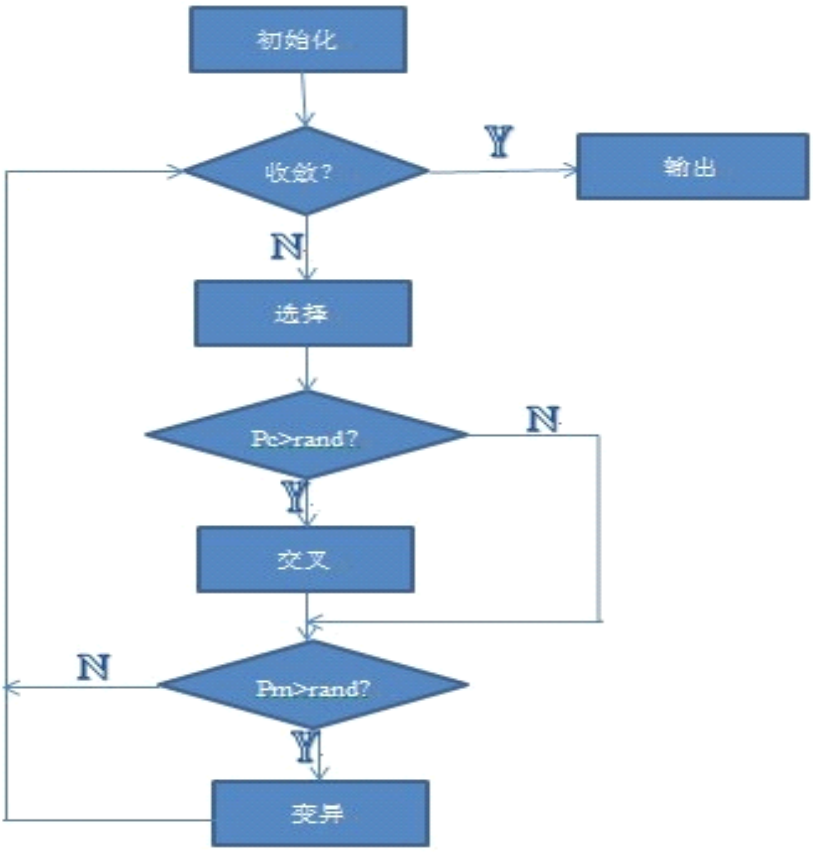

图 1　遗传算法流程图

3.TSP 问题的遗传算法设计与实现

3.1TSP 问题的图论描述

求最短路径问题，用图论术语描述如下：在图 G(V,A)中，V 表示顶点集合，V=(v1,v2,…,vn)对 G 中的某一边(v_i, v_j)，相应的有一个数 d(v_i, v_j)，如果 G 中不存在边(v_i, v_j)，则令 d(v_i, v_j)无穷大，如果把 d(v_i, v_j)认为是边(v_i, v_j)的长度，则通路的长度定义为组成路的各条边的长度总和。顶点v_i, v_j之间是否有边相连，由邻接矩阵来决定。

邻接矩阵 A：对一个具有 v 个顶点，e 条边的图 G 的邻接矩阵 A=[a_{ij}]是一个 v×v 阶方阵，其中 a_{ij}=1，表示 v_i和 v_j邻接， a_{ij}=0 表示 vi 和 vj 不相邻接（或 i=j）。

3.2 读取 txt 文件

标准的测试文件一般都是存储 n*2 的 txt 文件，为此本人编译了一个 readfile.m 的程序，方便了不同文件的使用。此程序返回的是城市的坐标矩阵 pop 和城市的距离矩阵 dista。

3.3 初始种群

对于n个城市的问题，每个个体即每个解的长度为n ，用s行,t列的pop矩阵表示初始群体，s表示初始群体的个数，t为n+1，矩阵的每一行的前n个元素表示城市编码，最后一个元素表示这一路径的长度。这一算法通过start.m程序实现。

3.3 适应度

在TSP的求解中，可以直接用距离总和作为适应度函数。个体的路径长度越小，所得个体优越，以pop矩阵的每一行最后一个元素作为个体适应值。求适应值的qiujuli.m 程序，见附录。

3.4 选择

选择就是从群体中选择优胜个体、淘汰劣质个体的操作，它是建立在群体中个体适应度评估基础上。这里采用方法是最优保存方法。

算法就是首先将群体中适应度最大的k个个体直接替换适应度最小的k个个体。程序为select.m，见附录。

3.5 交叉

受贪婪算法的启发, 本文设计一种有目的使适应值上升的交叉算子。已知两个父代 a1(m11,m12,m13,...,m1n),a2(m21,m22,m23,...,m2n)， 算法产生后代 a1’和 a2’的过程如下：

(1) 随机产生一个城市d作为交叉起点, 把d作为a1’和a2’的起始点

(2) 分别从a1和a2中找出d的右城市dr1和dr2,并计算(d,dr1)和(d,dr2)的距离j1和j2。

(3) 如果j1<j2,则把dr1作为a1’的第二个点，从a1和a2中删除d，并且把当前点改为dr1.转步骤(5)。

(4) 如果j1>j2,则把dr2作为a1’的第二个点，从a1和a2中删除d，并且把当前点改为dr2。

(5) 若此时p1和p2的个数为1，结束，否则回到第二步继续执行。

同理，把第二步中的右城市改成左城市dle1和dle2，通过计算(d,dle1)和(d,dle2) 的距离并比较大小来确定子代a2’。为程序cross.m，见附录。

3.6 变异

变异操作是以变异概率Pm对群体中个体串某些基因位上的基因值作变动，若变异后子代的适应度值更加优异，则保留子代染色体，否则，仍保留父代染色体。这里采用的方法是倒置变异法。

假设当前个体X为（1 3 7 4 8 0 5 9 6 2）。如果Pm>rand，那么随机选择来自同一个体的两个点mutatepoint(1)和mutatepoint(2)，比如说3和7，倒置P1和P之间的部分，产生下面的子体X’为(1 3 7 5 0 8 4 9 6 2)。为mutate.m程序，见附录。

4.试验与结果分析

试验采用TSPLib标准库的eil51，eil76，eil101作为测试实例。每个实例分别测试五次，求出平均值和最优值作为比较依据。

问题	求解次数	最优理论解	最优解	最差解	平均值
eil51	5	426	432.3981	445.9899	441.2703
eil76	5	538	561.5283	576.1645	570.2783
eil101	5	629	678.3685	699.7653	693.5415

下面给出五次运行中的最优路径图：

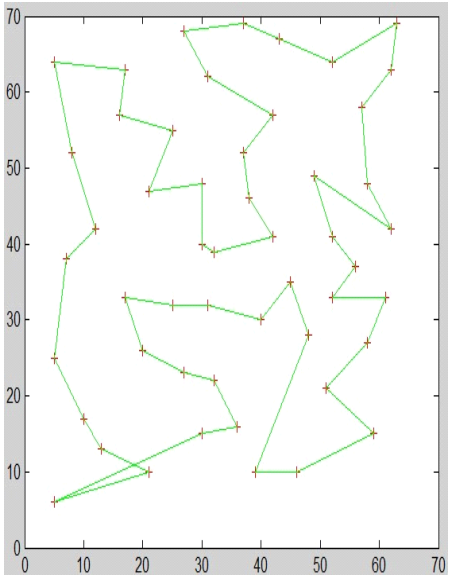

图1 eil51最优路径图

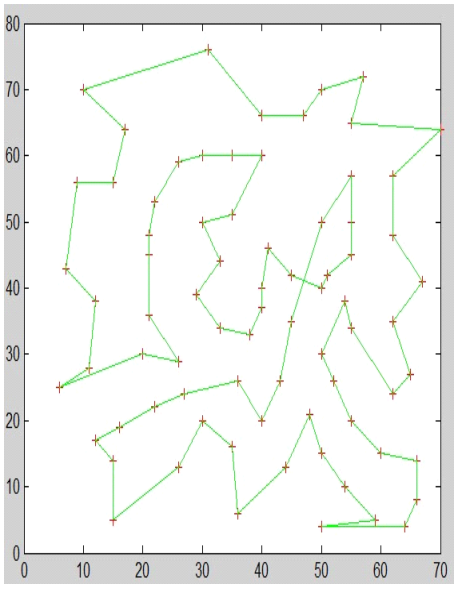

图2 eil76最优路径图

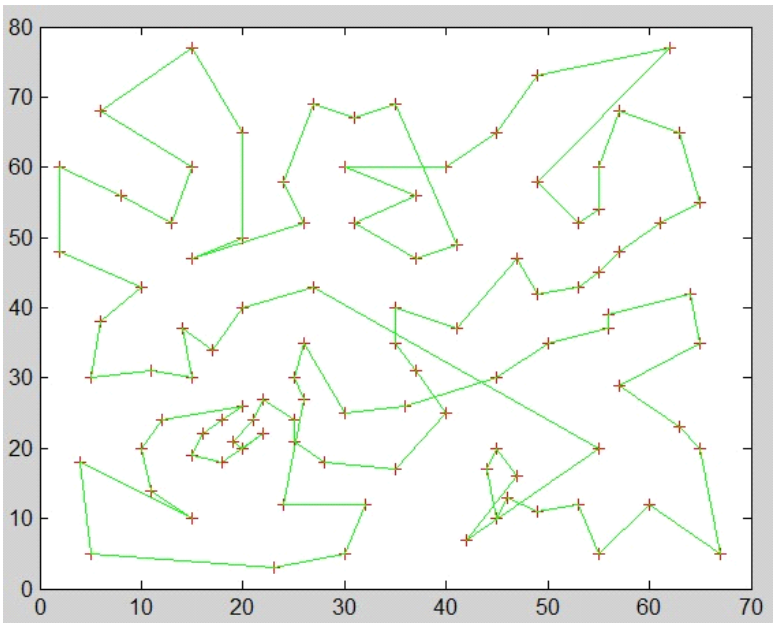

图3 eil101最优路径图

从图中可以看出，所获得路线都有交叉，明显不是最优路径。对于51个城市和76个城市来说，都只有一个交叉，而对于101个城市来说，交叉比较多，求得最优路径的结果也不是太理想。

从参数设置来说，对于51个城市和76个城市，参数：群体总数s=400，交叉概率Pc=0.9，变异概率Pm=0.2，最大迭代次数C=100。对于不断调整试验参数中可以得出，最大迭代次数越多，结果比较接近理论最优解。

对于101个城市，参数：群体总数s=500，交叉概率Pc=0.9，变异概率Pm=0.1，最大迭代次数C=200。对于不断调整试验参数中可以得出，对于较大数目的城市数，变异概率小一些，得到结果也会相应好一些。

5.结语

本文运用Matlab软件，利用遗传算法解决了小规模的TSP问题。文章首先介绍了TSP问题，并给出TSP问题的数学定义，然后介绍了遗传算法的原理以及算法的基本过程，最后通过对标准TSPLib中的51、76和101个城市分别进行了测试，根据试验对参数进行分析。本文程序解决小规模的TSP问题还可以，随着城市数目的增大，计算精度有所下降，计算时间增长很快，效率较低较快，这也是下一步需要改进的地方。

6.附录

%此为主程序代码，将下面各个程序分别保存在同一文件下运行，即可得出比较%

好的解，只是会有点交叉，但是最小路径值很接近最优理论值

```
function ga
```

```
s=500;%群体中个体数目
k=100;%选择优化个数
Pc=0.9;%交叉概率
Pm=0.1;%变异概率
C=200;%最大循环次数
[M,dista]=readfile('tsp76.txt');%读取城市坐标文件
[Ncities,b]=size(M);
t=Ncities+1;
farm=start(s,t);      %随机初始化种群
farm=qiujuli(farm,dista);%求出种群的适应度
counter=0;
while counter<=C
    counter=counter+1;
    farm=select(farm,k);%选择
    if(Pc>rand)
        farm=cross(farm,dista);%交叉
    end
    if(Pm>rand)
        farm=mutate(farm,dista);%变异
    end
end
[a,b]=size(farm); %求出总路径中的最小值
A=zeros(1,a);
for i=1:a
    A(1,i)=farm(i,b);
end
shortest_path=min(A)%最短路径值
[c,d]=find(A==shortest_path);
e=c(1);%画出最短路径图
for i=1:Ncities
    plot(M(i,1),M(i,2),'ro');% ‘ro’ 可改
    hold on
end

for i=1:t-2
    plot_ga(farm(e,1),farm(e,t-1),M);
    plot_ga(farm(e,i),farm(e,i+1),M);
end

-------------------------------------------------
-------------------------------------------------
%读取城市坐标的txt文件，格式为n*2的坐标，n为城市数目，%filename为文件

名，返回n*2的坐标pop，以及城市距离矩阵%city_distance，为n*n
```

```
function [pop,city_distance]=readfile(filename)
fid=fopen(filename,'r');
pop=fscanf(fid,'%d',[2 inf]);
pop=pop';
[a,b]=size(pop);
city_distance=zeros(a,a);
for i=1:a
    for j=i:a
        city_distance(i,j)=sqrt((pop(i,1)-pop(j,1))^2+(pop(i,2)-pop

(j,2))^2);
        city_distance(j,i)=city_distance(i,j);
    end
end
fclose(fid)
```

```
%生成s*t列的种群，其中每一行的前t-1个数为1到t-1的随机不重复排列，用于
%显示随机行走的路径，最后一列记录这样走的总距离，一般t等于城市数目+1,
%s为初始设置的群体中个体数目
function pop=start(s,t)
pop=zeros(s,t);
for i=1:s
pop(i,1:t-1)=randperm(t-1);
end
```

```
function [a]=bianma(k,u)  %编码，生成k个小于u的整数数组
a=zeros(1,k);
aa=0:u-1;
for i=1:k
    point=round(rand*(u-i))+1;
    a(i)=aa(point);
    aa(point)=[];
end
```

```
%该算法主要实现pop矩阵中最后一列的值，即总距离
function [pop]=qiujuli(pop,D)  %D为城市的距离矩阵，pop为种群
[s,t]=size(pop);
for i=1:s
    dd=0;
    for j=1:t-2
        dd=D(pop(i,j),pop(i,j+1))+dd;
    end
    dd=dd+D(pop(i,1),pop(i,t-1));                %dd为每个访遍城市路径
的

适应度
    pop(i,t)=dd;                                 %存储适应度
end
```

```
%该算法是将群体中适应度最大的k个个体直接替换适应度最小的k个%个体
function [pop]=select(pop,k)
[s,t]=size(pop);
m11=(pop(:,t));
m11=m11';
mmax=zeros(1,k);
mmin=zeros(1,k);
num=1;
while num<k+1;
[a,mmax(num)]=max(m11);
m11(mmax(num))=0;
num=num+1;
end
num=1;
while num<k+1;
[b,mmin(num)]=min(m11);
m11(mmin(num))=a;
num=num+1;
end
for i=1:k
pop(mmax(i),:)=pop(mmin(i),:);
end
```

```
-------------------------------------------------------------------
-


-------------------------------------------------
%交叉算法，主要思想是：每两行进行交叉的操作，使每行的适应度%减少
function [pop]=cross(pop,D)
[s,t]=size(pop);
pop1=pop;%保存原群体矩阵
m=zeros(1,t);%初始化交叉后第一个个体
n=zeros(1,t);%初始化交叉后第二个个体
for i=1:2:s %每两行进行交叉操作
    x1=pop(i,:);
    y1=pop(i+1,:);
    x2=pop(i,:);
    y2=pop(i+1,:);
    c1=round(rand*(t-2))+1;   %生成1到t-1之间的随机整数
    c2=c1;
    m(1)=c1;
    n(1)=c2;
    j=2;
    while size(x1,2)>2        %判断是否继续
        l=find(x1==c1);
        h=find(x2==c2);
        if l==t+1-j
            lr=1;
        else
            lr=l+1;
        end
        if h==1
            hle=t+1-j;
        else
            hle=h-1;
        end
        q=find(y1==c1);
        z=find(y2==c2);
        if q==t+1-j
            qr=1;
        else qr=q+1;
        end
        if z==1
            zle=t+1-j;
        else zle=z-1;
        end
        if D(c1,x1(lr))<D(c1,y1(qr))  %根据比较两点间距离进行交换
```

```
            m(j)=x1(lr);
            c1=x1(lr);
        else
            m(j)=y1(qr);c1=y1(qr);
        end
        x1(l)=[];                       %删除父节点
        y1(q)=[];                       %删除父节点
        if D(c2,x2(hle))<D(c2,y2(zle))%根据比较两点间距离进行交换
            n(j)=x2(hle);
            c2=x2(hle);
        else
            n(j)=y2(zle);c2=y2(zle);
        end
        x2(h)=[];%删除父节点
        y2(z)=[];%删除父节点
        j=j+1;
    end
    pop1(i,:)=m;
    pop1(i+1,:)=n;
end
pop1=qiujuli(pop1,D);%求出交叉后的路径矩阵
for i=1:s             %选择适应度变小的交叉，进行更新
    if pop1(i,t)<pop(i,t)
        pop(i,:)=pop1(i,:);
    end
end
```

```
%变异，D为城市的距离矩阵，进行变异
function [pop]=mutate(pop,D)
[s,t]=size(pop);
pop1=pop;
for i=1:s
mutatepoint=bianma(2);
b=round((mutatepoint(2)-mutatepoint(1))/2-0.5);
for j=1:b
zhong=pop1(i,mutatepoint(1)+j);
pop1(i,mutatepoint(1)+j)=pop1(i,mutatepoint(2)-j);
pop1(i,mutatepoint(2)-j)=zhong;
end
```

```
end
pop1=qiujuli(pop,D);
for i=1:s
if pop1(i,t)<pop(i,t)
pop(i,:)=pop1(i,:)
end
end
```

```
%画图
function plot_ga(a,b,V)
P=[V(a,1) V(a,2)];
Q=[V(b,1) V(b,2)];
c=P(1,2);
P(1,2)=Q(1,1);
Q(1,1)=c;
plot(P,Q,'g-')              %g表示颜色，-表示连线的形状，可改。
hold on
```