统计推断在数模转换系统中的应用

组号：23　　王博远 5130309579 主苏杰 5130309582

摘要：本文将介绍如何处理大量的电子产品的数据，并利用遗传算法和模拟退火算法，分别用多项式拟合法和三次样条插值法来确定最优的取点方案及拟合方式，本文目的在于旨在选取少量数据点推断大量数据的统计特性，来达到最小的总成本，从而合理高效地评估该电子产品的特性。

关键词：遗传算法，模拟退火算法，三次样条差值，定标

The article for the Application of Statistical Inference in Analog-to-Digital Converting System

ABSTRACT: In this paper, there is an introduction about how to deal with large amount of data from the specific kind of electronic device on the basis of genetic algorithm and the simulated annealing algorithm. The polynomial fitting and the cubic spline interpolation are used to choose the best combination of pots and its homologous fitting method. This passage is aimed to choose the less data to infer the statistical characteristics of the whole data in order to get the lowest cost. Hence, this method can be used to assess the performance of the electronic device reasonably and efficiently.

Key words: genetic algorithm, simulated annealing algorithm, cubic spline interpolation, calibration

1 引言

随着现代科技的日益发达，电子仪器已经普及到了各地，被广泛地使用，电子产品的制作过程中仍少不了校准和定标的步骤。要生产电子产品，总是少不了输入数据和输出数据的关系。但在实际运用中，我们很难找到具体两个物理变量的实际函数关系。因此，定标的工序为一般是先生产一定批量的电子产品，测得若干组数据，再进行一定的数据处理得到最佳的函数关系，以便达到最终定标目的。普遍意义上大多数输入输出都呈现非线性关系，即多数图像为曲线，通过数学推导并不可靠，因此在定标的时候要进行一定数量的数据拟合。监测的物理量与相关的变量的选取也是影响最终定标校准的一个不可忽略的因素，也有一定的附加成本。减少测定的数量即可以节约大量生产成本，因此通过算法比较设计出一种成本合理的最优传感特性校准（定标工序）方案显得尤为重要。

2 模型假设与变量建立

2.1 题目解读与分析

本实验中一共提供了 469 个样本，每个样本中有 51 组数据，x 均为 5.0 到 10.0 间且隔为 0.1 时对应不同的取值，y 在 0 到 100 之间随 x 单调递增，且为非线性曲线关系，大致类似于陡缓陡的三段线性关系。本定标实验目的在于选取一种特定的拟合方式，再通过一定的算法找到最优的取点方案，使得总定标成本最小，其中总定标成本包括误差成本和取点成本，

误差成本为对于每一个样本取同样的位置点数，从样本库获取并按选定拟合方式计算该样本所有拟合值，对所有样本逐一完成定标，计算总误差和取点成本，最后对样本个数取平均值。

找到最优的取点方案，即最小成本的取点方案，是采用相对的比较算法得到的，是优化组合的问题。由于从 51 个点数选取若干个点的方案数巨大，无法直接采用费时费力的暴力穷举法，因此采用一定的启发式搜索算法来代替，一方面要取点数尽可能少来降低取点成本，另一方面拟合要越接近越好来降低误差成本，为了找到之间的平衡关系，因此本文决定采用遗传算法和模拟退火算法来选取特定的取点组合方案。再找到两个启发式搜索算法大致得到的最优取点方案数之间再采用其他不同拟合方式进行比较分析，得出最终结论。

2.2 物理变量

X　自变物理量，本题 $X \in [5.0,5.1,...10.0]$

Y　原始的被监测物理量

M　原始样本总个数

$\hat{Y}$　通过传感器自身的功能测算，即通过计算、拟合方式得到的数据

S(i,j)　误差成本，根据 Y 和 $\hat{Y}$ 差值的大小进行分段函数确定，俗称“惩罚”

n　取点数目，即单点测定次数

Q　单点测定成本

S　测定的综合成本

2.3 流程图

为了对本课题展开有效讨论，需建立一个简洁的数学模型，对问题的某些方面进行必要的描述和限定。

2.3.1 成本的计算方法

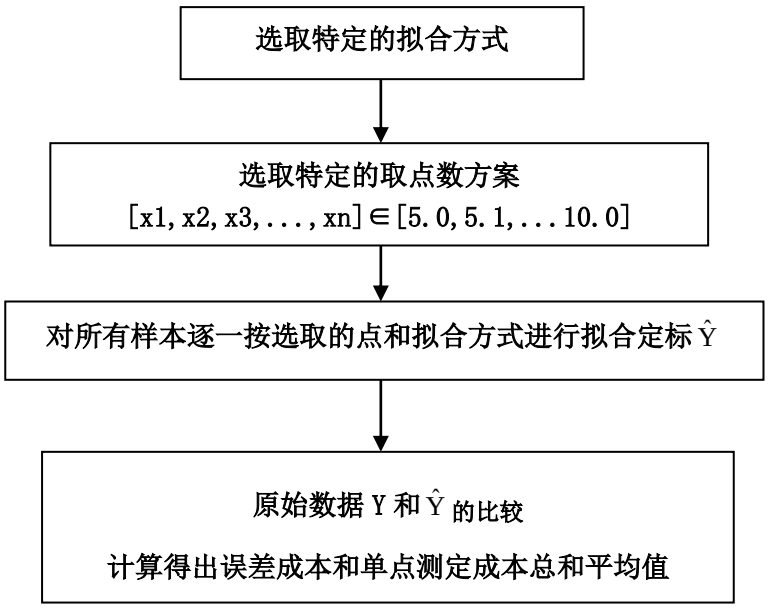

图 2-3-1 综合成本的计算方法

2.3.2 启发式搜索算法的运用

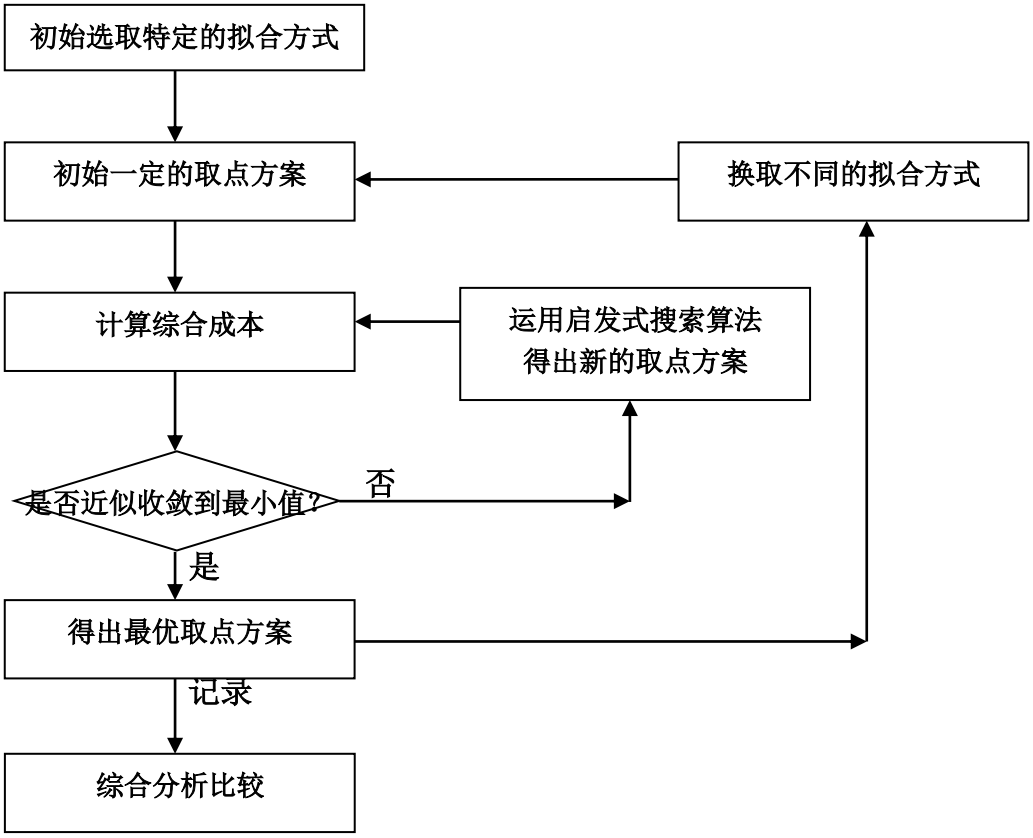

图 2-3-2 运用启发式搜索算法的流程图

2.4 遗传算法介绍

遗传算法（GA）是一类借鉴生物进化规律形成的随机化搜索方法。遗传算法的实现可分为以下几个步骤：首先随机建立初始状态即个体基因第一代，根据实际问题求解函数来设定适应度函数。其次在接下来的若干代中，依次进行：每个个体适应度函数的求取，根据适应度值的大小进行轮盘概率求取，个体间进行交叉和变异，繁殖出具有适应度更好的下一代。

2.5 模拟退火算法介绍

模拟退火（SA）是一种通用概率算法，用来在一个大的搜寻空间内找最优解。可分为如下四个步骤：第一步为初始化温度，解的状态和每个温度对应的迭代次数。第二步为对于每个温度，依照迭代次数依次产生新的解并根据对应的“能量”变化判断是否接受新解，还可以根据一定的小概率误接受新解。中间参加判断，如果满足连续若干个新解都没有被接受时终止算法。不满足终止条件时则降低温度，直到降低到最低温度。

2.6 不同拟合方式的选取

本文选取的是两种拟合运算方法：三次样条插值(spline)、interp1 中的三次拟合函数(polyfit)。

三次样条差值为在 0 到 n 这 n+1 个数据点中每一分段都是是三次多项式函数曲线，且节点达到二阶连续。

多项式函数拟合法为根据若干个数据一次性拟合出一个多项式，根据阶次 n 给定 n+1 个系数，本文采用的是三次多项式拟合法，故返回 4 个系数。

3 思想方法与具体实现

3.1 遗传算法及程序设计思路

（1）初始化：设立初始种群 M 个个体，每个个体的基因长度为 51，每个基因点是 0 或 1 连个状态代表选取该点或不选取该点。

（2）适应度函数：适应度函数包括了适应度的计算和为下一代的配对做准备的两个部分。由选定的拟合方式计算每一个个体的定标成本，包括误差成本和取点成本的两项内容。

（3）适应度选择、交叉及变异：对于适应度的选择评判，我们选择用倒数来刻画，对种群中的每个个体拟合得到的综合成本取倒数，成本小的适应度高，应该被保留，大的适应度低，应当被淘汰，除以倒数之和得到用 0 到 1 的小数来表示的适应度。这样就求出了每个个体所对应的适应度，并且满足误差小的适应度高的条件。为了配对时的方便，在适应度计算完毕后，增加了一个赌轮选择出过渡种群（根据适应度优胜劣汰但不进行配对）的部分：对于适应度矩阵，从第二项开始，进行累加，使得每一项和前一项构成一个区间，形成一块赌轮上的区域。再用随机数生成 M 个（M 为种群大小）的 0~1 之间的随机数矩阵，对于这 M 个数每个都和每一个赌轮区间进行比较，如果落在区间里，就记录对应的父代的位置。

（4）输出该代所有个体中适应度最小的一个，回到步骤（2），依次循环，直到设定的代数。

3.2 模拟退火算法及程序设计思路

（1）初始化：仅设立一个个体，其基因长度为 51，每一个基因点是 0 或 1 两个状态代表选取该点进行拟合或不选取该点。

（2）“能量”函数：由事先选定的拟合函数测定该取点方案下的误差成本和测定成本，相加得到定标成本。

（3）对于每一个降低到的温度，迭代若干次：求原解和新解的能量值，扰动求新解，比较原解和新解对应的“能量”值，如果新解“能量”值低，即成本低，则采取新解，或者以一定的小概率接受新解。

（4）若达到一定的次数没有接受新解则跳出算法，直接结束。否则迭代完次数后降低温度，回到步骤（3），直到降低到最小的温度为止。

4 数据分析与结论

4.1 两种算法的最优取点组合比较

采用的方式为三次样条插值时，经过代码的测试，发现到最优取点组合时取样点的个数稳定在 6 个或者 7 个，且总成本经过启发式搜索算法基本稳定收敛在 90 到 96 间，说明已经达到或接近该种拟合方式下的最小定标成本。

表 4-1-1 遗传算法三次样条插值得到的最优解

取点方案	总成本
[2,10,21,30,41,50]	93.4009
[2,9,20,26,33,43,50]	93.8561
[3,11,21,29,40,49]	94.1034
[2,10,20,28,35,44,50]	94.2665
[2,9,20,26,33,43,49]	94.3571
[2,9,20,28,34,43,50]	94.3945
[2,9,20,26,33,44,49]	94.5512
[2,10,20,29,34,44,50]	95.1716

表 4-1-2 模拟退火算法三次样条插值得到的最优解

取点方案	总成本
[3,12,22,31,43,50]	92.8774
[3,12,22,31,41,50]	92.9595
[3,11,22,31,41,50]	93.0533
[3,11,21,30,41,50]	93.2708
[2,10,21,30,41,50]	93.4009
[2,9,20,26,33,43,50]	93.8561
[2,9,20,27,34,43,50]	93.9495
[2,9,19,26,33,43,50]	93.9947

故在遗传算法中，运用三次样条插值进行拟合时，遗传算法得出的最好解为[2,10,21,30,41,50]，成本 93.4009。模拟退火算法得出的最好解为[3,12,22,31,43,50]，成本 92.8774。

且通过比较观察可知：用模拟退火算法得出最优组合解时往往取点数为 6，且在两个算法的 8 个最优解中，模拟退火算法得出的取样个数为 6 的情况比遗传算法得出取样个数为 6 的概率要大，且模拟退火的方法相比于遗传算法能求解出平均成本更小的取点组合方案。接下来我们又从不同的因素角度用控制变量的方法来进行了下一步的研究。

4.2 不同启发式算法的收敛情况快慢比较

本项比较了遗传算法每一代和模拟退火算法每一温度的最优成本值及平均成本值的收敛快慢情况。

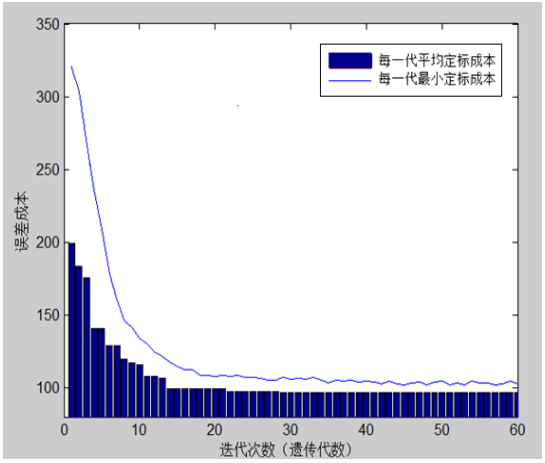

图 4-2-1 遗传算法收敛情况

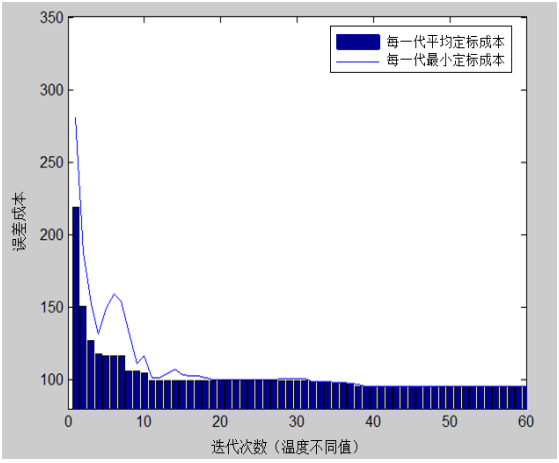

图 4-2-2 模拟退火算法收敛情况

左上图像为遗传算法每一代的最小成本和平均成本的收敛情况，右上图像为模拟退火算法的每一个温度下的最小成本和平均成本收敛情况 ，且均为一般情况。

由此可以看出，模拟退火算法虽然有一些上下波动，出现反弹回跳的情况，但是总体上模拟退火算法比遗传算法在平均成本和最小成本的收敛速度更快。

4.3 不同拟合函数对最后定标成本的影响

三次拟合函数（polyfit 或 cubic）比三次样条差值法（spline）函数能实现减小平均成本。

表 4-3-1 遗传算法三次函数拟合的最优解

取点方案	总成本
[3,12,23,29,39,49]	87.4243
[5,14,26,33,40,49]	88.4115
[3,12,233,31,36,47]	90.5618

表 4-3-2 模拟退火算法三次函数拟合的最优解

取点方案	总成本
[4,13,21,28,36,48]	86.5501
[4,12,21,28,36,48]	86.8113
[3,16,25,31,37,49]	87.0469

换用不同的三次多项式拟合方法，得到的成本更小。遗传算法用 polyfit 拟合得到的最优取点组合为[3,12,23,29,39,49]，其对应成本为 87.4243，而模拟退火算法用其拟合得到的最优取点组合为[4,13,21,28,36,48]，其对应成本为 86.5501.

4.4 变异参数对最后结果的影响

变异的参数一般设为 0.01~0.001，变异的参数变大表示突变的概率变大，也就是有可能产生更优的解，但是同时也可能产生误差更大的个体。我们可以从图中看到由于变异的概率变大，平均误差产生的突变程度就更大。

当变异概率 Rate 为 0.005，取种群数量为 150，用遗传算法取 40 代，得到如下结果。

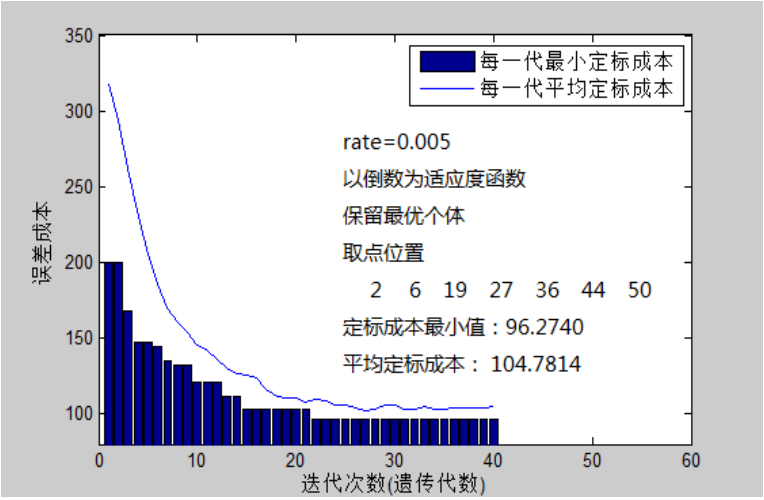

图 4-4-1 遗传算法 Rate=0.005 时收敛情况

当变异概率 Rate 改为 0.01，同样取种群数量为 150，用遗传算法取 40 代，得到如下结果。

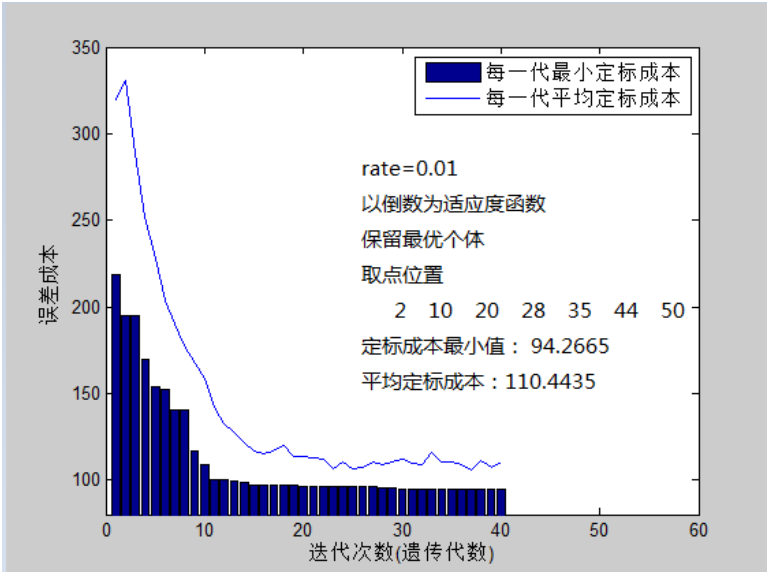

图 4-4-2 遗传算法 Rate=0.01 时收敛情况

4.5 适应度函数对概率的算法对最后收敛结果的影响

适应度函数可以采取两种，一种是取倒数，另一种是先平方再取倒数。两者都能做实现令误差小的个体适应度高，但是二者的区别在于平方的处理将个体间的差异放大，这样虽然能让误差小的个体生存概率大大增加，收敛的速度变快。可是由于赌轮选择会导致种群中的下一代中的大部分都是这一个单独的个体，虽然种群的平均误差会比直接求倒数的低，但是这样会丧失种群的基因多样性，导致所求得的结果只是局部最优解。

直接取倒数时的图见 4-4-1；

反之，同样取种群数量为 150，用遗传算法取 40 代先平方再取倒数时，得到的结果如下：

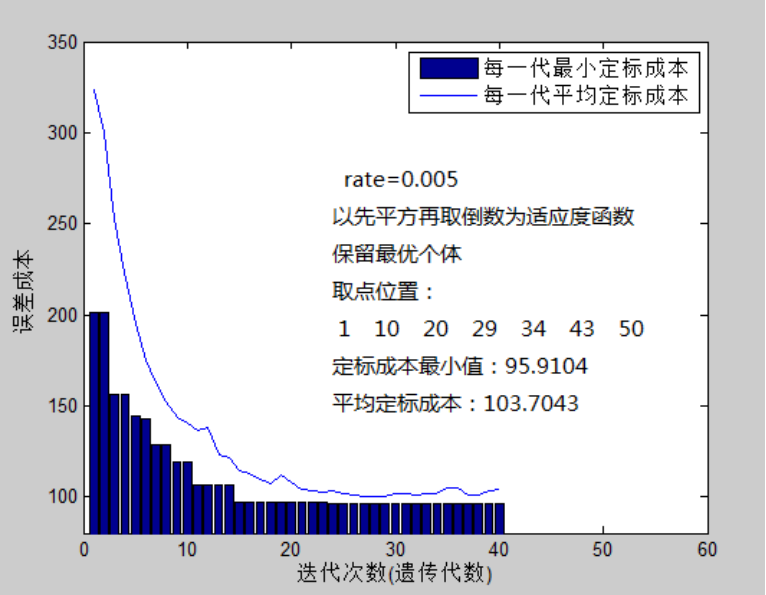

图 4-5 遗传算法适应度平方取倒数时收敛情况

从两幅图中我们可以看到，一开始的平均误差和误差最小值下降速率差别不大，在第十代之后我们可以发现方法一仍在逐渐收敛，最小值在第 17 代左右达到稳定值，而方法二则在第十代之后很快收敛，最小值在第 13 代左右就达到了稳定值。这是由于在种群个体的误差相差不大时，平方能将个体之间的差距放大，收敛的速度就更快。

但是经过多次的实验后，没有出现我们所预想的局部最优解的情况，原因可能为由于变异函数的存在，基因的多样性得以保留，并不会出现局部最优解。

所以可以得出将适应度函数替换成先平方再取倒数，是对适应度函数的有效改进的结论。

4.6 是否保留最优个体对最后结果的影响

在不保留最优个体的情况下，每一代中最优的个体的基因会在配对和变异中被破坏和重组，这样不能使种群的最小误差满足单调减小的趋势，从而导致最小值发生变化，收敛的速度与保留最优解相比相差很大。从图中可以看出，即使将遗传代数改为 80 代，最小值还是维持在 170 左右，最小值一直在发生变化，种群的平均误差的大致方向是在逐渐减小，但是收敛的速度过于缓慢，还会出现先突变增加再逐渐减小的情况。可见保留最优个体能使最小值满足单调递减的趋势，也使种群的平均误差收敛更快。

故保留最优个体能达到较快收敛的效果，并且能够算出较小的定标成本。

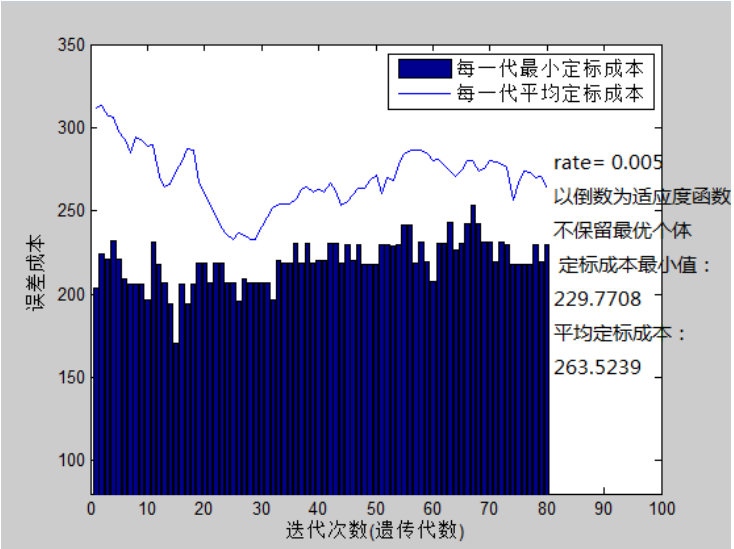

图 4-6 遗传算法不保留最优个体时收敛情况

5 参考文献

[1] 上海交大电子工程系. 统计推断在数模转换系统中的应用课程讲义 [EB/OL].ftp://202.120.39.248.

6 附录 程序清单

一 、遗传算法

初始化时，先设立初始种群 M 个个体，即为 M╳51 的由 0 和 1 随机生成的矩阵代表基因。每个个体代表基因的一行，“1”代表选取该点对应的 x 数字，“0”代表不选。

适应度函数包括了适应度的计算和为下一代的配对做准备的两个部分。先读取表格文件所有的数据记录到 x 和 originY 中，对于每一个个体，由随机生成的基因行数“1”代表取该点，调出该行“1”的个数及取点所对应的 x 值，分别对 469 行中每一行分别进行独立的拟合，得到一个新的拟合数据矩阵 calculateY，再进行与 originY 比较调用误差生成函数，取平均值，再加上取点的个数决定的单点测定误差随得到每一个个体的成本。最后 M 个个体的所有成本值记录到 Result 矩阵中，完成适应度函数的计算。

接下来再通过适应度选择，交叉和变异得到下一代的个体，即新的 M╳51 矩阵，往复循环依次迭代直到设定的代数。

main.m 文件

```
tic;
clear all;clc;
%fid = fopen('output.txt','wt');                    %输出文件
minput=dlmread('20141010dataform.csv');             %从xlsx中读入数据
M0=938;N=51;
samplenum=M0/2;npoint=N;                            %设置样本数量及点数规模
global x;
x=zeros(1,npoint);
global originY;
global tempMaxChosenY;                              %记录每一代中最小成本对应的取点方案
global tempMax;                                     %记录每一代中最小的成本
global average;
originY=zeros(samplenum,npoint);
calculateY=zeros(469,51);                           %开辟一个用于计算拟合结果的数据
M=200;                                              %设定种群数量
G=60;                                               %设定遗传的代数
Result=zeros(1,M);
for i=1:samplenum
    x(1,:)=minput(i*2-1,:);
    originY(i,:)=minput(2*i,:);
end                                                 %把xlsx表格中的数据全部读入x,originY矩阵中
RecordtempMax=[];
Recordaverage=[];
Gene=round(1*rand(M,51));     %随机生成M行51列的0、1矩阵代表群体基因初始化
for generation=1:G                                  %开始遗传算法
    Result = fitness( M,Result ,Gene);              %计算成本
    son = shiyindu( M,Result ,Gene);                %根据成本进行概率选择
    sonMutate = mutate(son ,M);                     %交叉
```

```
    sonbianyi = bianyi(sonMutate,M);          %变异
    Gene=sonbianyi;                                %更替新的基因作为下一代的基因
    generation
    tempMaxChosenY
    tempMax
    RecordtempMax=[RecordtempMax,tempMax];
    Recordaverage=[Recordaverage,average];
    fprintf(fid,'generation:%d \n ',generation);
    fprintf(fid,'min_cost:%.4f \n',tempMax);
    average
end
bar(RecordtempMax)
axis([0 60 80 350]);
hold on;
plot(Recordaverage)
axis([0 60 80 350]);
toc;
```

fitness.m 文件

```
function    Result= fitness( M,Result ,Gene)
global originY;
global x;
global tempMax;                                    %tempMax记录每一代中最小的成本
tempMax=12*51+469*51*25;
global tempMaxChosenY;        %tempMaxChosenY记录每一代中最小成本对应的取点方案
global average;
average=0;
for m=1:M
    [Xtemp,Ytemp]=find(Gene(m,:)~=0);
    count=length(Xtemp);
    tempChosenX=Ytemp.*0.1+4.9;
    ChosenY=originY(:,Ytemp);
    calculateY=spline(tempChosenX,ChosenY,x);        %三次样条差值拟合计算
    errorCount=abs(calculateY-originY);                %误差成本计算
    le0_5=(errorCount<=0.5);
    le1_0=(errorCount<=1);
    le2_0=(errorCount<=2);
    le3_0=(errorCount<=3);
    le5_0=(errorCount<=5);
    g5_0=(errorCount>5);

    sij=0.5*(le1_0-le0_5)+1.5*(le2_0-le1_0)+6*(le3_0-le2_0)+
    12*(le5_0-le3_0)+25*g5_0;
```

```
    si=sum(sij,2);
    Result(1,m)=sum(si)/469+12*count;                %计算平均成本
    if Result(1,m)<tempMax                           %比较是否是该代中的最小成本
        tempMaxChosenx=tempChosenX;
        tempMaxChosenY=Ytemp;
        tempMax=Result(1,m);
    end
end
average=sum(Result)/M;
out=Result;
end
```

shiyindu.m 文件

```
function son = shiyindu( num,m ,parent)
%适应度函数，包括了适应度的计算和自然选择的过程
% 保留最优的个体，num表示种群大小，m表示fitness的误差结果，parent表示
% m=m.^2;
%sum1 = sum(m);
m=1./m;% 取倒数
%m=sum1*m;
sum2=sum(m);% 计算矩阵的和以使适应度用小数表示，并且保证适应度的和为1
m = 1/sum2 *m;%得到适应度矩阵
adp = zeros(1,num)+1;%用来记录产生的过渡种群中的个体在原种群中的位置
son = zeros(num,51);% 产生的过渡种群
b= m(1);%用于找最小的误差。一开始选取第一个个体，之后再逐个比较
e=1;% 记录误差最小值也即是适应度最小的个体的位置
for i=2:1:num% 对于每个适应度进行累加，得到赌轮的区域
    if (m(i)>b) % 比较法寻找适应度最大的个体
        e = i;
        b=m(i);
    end
    m(i)= m(i-1)+m(i);%累加
end
for i=2:1:num %从子代第二个开始记录，第一个为最优的个体
    for j=1:1:num %与赌轮各个区域逐个比较
        c=rand; %产生随机数
    if (c < m(j)    && j==1) adp(i) = j ; %落在第一个区域
    continue;
    end
    if (c < m(j)    && c >=m(j-1)) adp(i) = j ; %落在第j个区域，记录位置j，并跳出循环
    continue;
    end
```

```
    end
end
son(1,:) = parent(e,:); %生成的过渡矩阵中第一个个体为上一代中的最优个体
fori=2:1:num
    son(i,:) = parent(adp(i),:); %取出赌轮结果中对应的父代个体，构成过渡代
end
out= son; %返回过渡代

end
```

mutate.m 文件

```
function m = mutate( m ,num)
%  过渡代的配对，产生新的一代
%     num 为种群大小，m 为适应度产生的过渡矩阵
%  采用生成 num/2*1 的随机数用来决定交换基因的位置
k = num/2; %两两配对，k 为个数的一半
pos = randi(51,k,1); %生成范围为 1~51 的整数随机数的 k*1 的矩阵
for i = 1:1:k %配对 k 次
    if (pos(i)==51)   %如果是最后一位，交换后面的无法进行，就不交换
continue
end
    tmp1 = [m(i+1,1:pos(i)),m(num-i+1,(pos(i)+1):51) ]; %  交换基因段，产生新的个体
    tmp2 = [m(num-i+1,1:pos(i)),m(i+1,(pos(i)+1):51) ];
    if (i~= 1) m(i,:) = tmp1; %如果位置不等于 1，就变换成新的基因，为了保留最优个体在位置 1，并且能让最优个体的基因段能参与配对
end
    m(num-i+1,:) = tmp2; %新的个体
end
out = m;   %返回新的种群
end
```

bianyi.m 文件

```
function m2 = bianyi( m2 ,num)
%变异函数，对于每个基因点 0.01 的概率变异
% m2 为新的种群，num 为种群大小

for i=1:51                                  %每个基因点
    for j=2:num                             %对于每个个体，第一个最优个体保留，不变异
        if (rand<0.01)                      %变异的概率为 0.01
            m2(j,i) = 1-m2(j,i);            %0 变为 1，1 变为 0
        end
    end
end
```

```
out = m2;%返回 m2，得到最终的种群
end
```

二 、模拟退火算法

初始化时，设立 1╳51 的由 0 和 1 随机生成的矩阵代表基因。“1”代表取该点对应的 x 值，“0”代表不选。正式进入算法后，先读取表格文件所有的数据记录到 x 和 originY 中，调出该行“1”的个数及取点所对应的 x 值，分别对 469 行中每一行分别进行独立的拟合，得到一个新的拟合数据矩阵 calculateY，再进行与 originY 比较调用误差生成函数，取平均值，再加上取点的个数决定的单点测定误差随得到综合成本，该数字带入到 `Result` 中。再通过变异进行解附近的扰动，比较与原来的定标成本，若优于原来的值或有一定的小概率接受新解，不停地迭代直到降低到最小的温度。记录并比较每一个温度下的最小值和平均值。

main.m 文件

```
tic;
clear all;
clc;
%fid = fopen('output.txt','wt');
minput=dlmread('20141010dataform.csv');
M0=938;N=51;
samplenum=M0/2;npoint=N;
global x;
x=zeros(1,npoint);
global originY;
global tempMaxChosenY;
average=0;
originY=zeros(samplenum,npoint);
calculateY=zeros(469,51);
M=1;
Result1=zeros(1,1);
Result2=zeros(1,1);
for i=1:samplenum
    x(1,:)=minput(i*2-1,:);
    originY(i,:)=minput(2*i,:);   %从原始xlsx中读入数据
end
tempMin=51*12+51*25;
Gene1=round(1*rand(M,51));
T=100000;                  %初始设定的温度
generation=1;
Record1=[];Record2=[];Record3=[];
while T>0.1                    %直到退火到的最小温度
    flag2=1;k=0;Record2=[];
    for t=1:90
        flag = 0;
```

```
            Gene2 = bianyi(Gene1);
            Result1 = fitness( 1,Result1 ,Gene1);
            Record2=[Record2,sum(Result1)];
            Result2 = fitness( 1,Result2 ,Gene2);
            t=Result2-Result1;
            if t<0|rand<exp(-t/(T*0.0001))       %新解优于原解或以一定的小概率接受新解
                Gene1 = Gene2;
                flag=1;
            if Result2<tempMin
                tempMin=Result2;
                Recorder=tempMaxChosenY;
            end
            end
            if flag==1
                k = 0;
            else
                k = k+1;
            end
            if k>40
                flag2=0;
            end
            if k>40
                break;
            end
        end
        average=sum(Record2)/90;
        if flag2==0 break;
        end
        T
        T=T*0.9;
        tempMin
        Record1=[Record1,tempMin];            % Record1记录每一个温度中的最小定标值
        Record3=[Record3,average];              % Record3记录每一个温度的平均值
        Recorder
        generation
        generation=generation+1;
end
bar(Record1)                                    %画出最小定标值和平均值的图形
axis([0 60 80 150]);
hold on;
plot(Record3)
axis([0 60 90 350]);
toc;
```

fitness.m 文件

```
function    Result= fitness( M,Result ,Gene)
global originY;
global x;
global tempMaxChosenY;
tempMax=12*51+469*51*25;
m=1;
[Xtemp,Ytemp]=find(Gene(m,:)~=0);
count=length(Xtemp);
tempChosenX=Ytemp.*0.1+4.9;
ChosenY=originY(:,Ytemp);
calculateY=spline(tempChosenX,ChosenY,x);       %采用三次样条插值方法
errorCount=abs(calculateY-originY);
le0_5=(errorCount<=0.5);
le1_0=(errorCount<=1);
le2_0=(errorCount<=2);
le3_0=(errorCount<=3);
le5_0=(errorCount<=5);
g5_0=(errorCount>5);
sij=0.5*(le1_0-le0_5)+1.5*(le2_0-le1_0)+6*(le3_0-le2_0)+12*(le5_0-le3_0)+25*g5_0;
si=sum(sij,2);
Result(1,m)=sum(si)/469+12*count;         %计算误差成本和测定成本
tempMaxChosenx=tempChosenX;
tempMaxChosenY=Ytemp;
tempMaxChosenY;
out=Result;
end
```

bianyi.m 文件

```
function m2 = bianyi( m2 )
global num;
for i=1:51
      if (rand<0.03)
            m2(1,i) = 1-m2(1,i);
      end
end
out = m2;
end
```