统计推断在数模转换系统中的应用

组号 24 赵双星 5130309589，刘仲康 5130309573

摘要: 本文详细叙述了遗传算法的理论基础及工作原理，并结合某电子产品的校准的实际问题，运用遗传算法在短时间内得到相对优化的解决方案。实践表明遗传算法对表达式没有特殊要求、能高效的随机搜索优化方案、全局性、可操作性等优点是其广为应用的原因。
关键词： 遗传算法、传感器、定标

Statistical inference application in the DAC system

Abstract: The theory of Genetic Algorithm and how it works are introduced in detail. Genetic Algorithm can get an optimization solution in a short time for electronic products' calibration. Practice shows that the Genetic Algorithm has no special requirement for expression. And the high-efficiency, wholeness and manipulability of the Genetic Algorithm account for its extensive applications.
Keywords: Genetic Algorithm, sensor, calibration

1 引言

1.1 背景概述

某电子产品内部有一检测模块需要校准，但其输入输出特性呈明显三段且均为非线性。不同个体间虽曲线形态相同但两两相异，且中段的起始位置有随机性差异，如图 1-1-1。

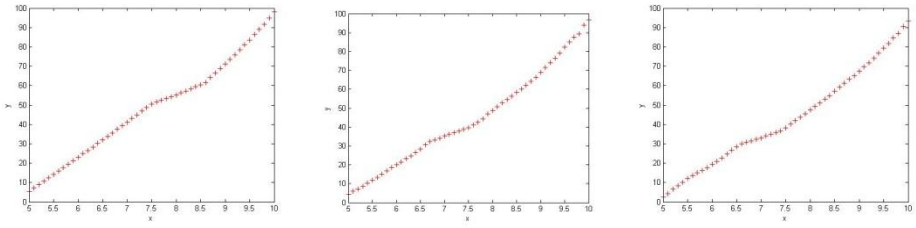

图 1-1-1 三个不同样本的特性曲线

1.2 任务概述

为本电子产品模块的校准建立一个模型。再通过遗传算法求得如何建立模型才能使得总成本趋于合理。

本课题选择测定点的组合方案相当于解一个组合优化问题，但是备选的方案数量巨大，是典型的 NP 难问题。比如只讨论测定 9 个点的方案也大概有 30 亿种情况需要列举，目前的计算机尚难对该问题进行简单穷举（暴力穷举）的方式进行解决。因此解决本课题的任务自然而然地落在了“启发式搜索算法”的身上。在本课题中，我们选取其中的遗传算法进行问题求解。

2 GA 算法的原理

遗传算法是一种宏观意义上的仿生算法，它模拟的机制是一切生命智能的产生与进化的过程。作为一种广为应用的、高效能的随机搜索与优化方法，它对模型的表达式没有特定的

要求，如目标函数用约束函数的连续性、可微性等函数解析性质的限制，还具有全局优化性、稳健性与可操作的简单性等优点[1]。其过程图如下（图 2-1-1）：

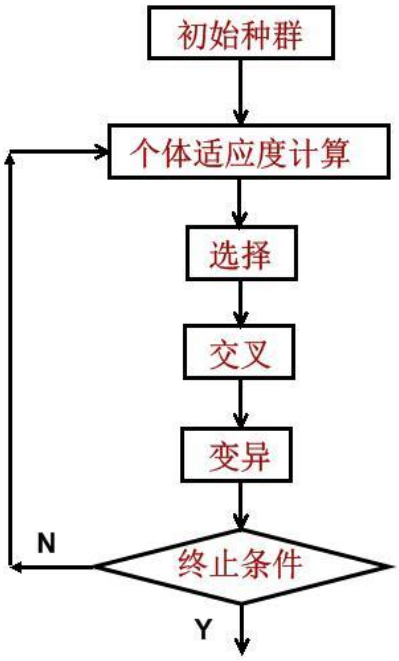

图 2-1-1 遗传算法的基本步骤

2.1 编码原理

遗传算法不是对实际优化变量进行实际操作，而是对表示实际变量可行解的遗传编码进行操作。因此遗传算法中第一步就是将实际变量转化为遗传编码。这一步类似于我们所学的定义域与值域的关系：通过一个对应关系使可行解与遗传编码实现一一对应。通常的编码方式包括：二进制编码、格雷码编码、浮点数编码。

2.2 适应度计算

遗传算法是一种概率性搜索算法，但是它并非等概率无目的地任意搜索，而是通过适应度函数来进行选择。适应度大的个体存活进行下一轮遗传。所谓的适应度大就是此方案较符合问题解决方案的标准。因此适应度函数的设计直接影响遗传收敛性与收敛速度。若收敛速度过小，则收敛性较弱；若收敛速度过大，则过早收敛，种群缺少多样性，即满足条件进入下一轮遗传的可行解数目过少。

2.3 选择

选择操作决定如何从当前种群中选取个体作为下一代种群的父代个体。不同的选择策略导致的选择压力也不一样。选择压力即最佳个体被选中的概率与平均中概率的比值。压力越大，选中最佳个体的概率越高，但收敛速度较快，容易过早收敛。较小的选择压力则会导致时间上的损失。

常用的选择方法有基于比例的适应度分配方法和基于排名的适应度分配方法。本课题中我们采用的轮盘赌选择法就是基于比例的适应度分配方法中的一种。

2.4 交叉

交叉是指将父代个体的部分结构加以替换重组而产生新的个体。其目的是在下一代获得优良个体，提高遗传算法的搜索能力。一般交叉不能破坏太多个体编码，同时又要产生一些较好的新编码。另外交叉算法要结合个体编码一起考虑。

常用的交叉算法有实值重组、离散重组、二进制交叉、单点交叉、多点交叉、洗牌交叉等，本课题中采用单点交叉作为交叉算法。

2.5 变异

变异是遗传算法中保持生物多样性的一种方法。它以一定概率选择某一基因进行改变（对于二进制编码来说就是 1 变 0 或 0 变 1）。变异的概率可以给定，也可以采取自适应取得。通过这种方法可以更广泛地进行基因选择，而且又不过多改变优良的基因编码，是提供多种基因选择的一种基础途径。

3 基于遗传算法的传感特性校准方案

从数据库中提取一定数量的样本，使用遗传算法随机生成定标点的个数与位置，并用二进制矩阵来表示。通过遗传算法的优化求解，高效地找到成本合理的传感特性校准方案。

3.1 编码方案的设计

因为每个定标点只有两种状态：被选作定标点与未被选作定标点。这与二进制序列拥有极大的相似性，因此采用二进制序列对每一种校准方法进行编码。因为一个样本的实测点数只有 51 个，所以个体的基因个数自然为 51 个。而对于初始的个体数量设置为 200，这样是为了保证初始个体的多样性。由此便生成了初始的二进制序列矩阵，规模为 200×51.每行代表一个个体，每个数据代表一个基因。

需要特别指出的是，在分析该问题时，我们发现：取点位置在 1,11,21,31,41,51，拟合方法采用三次样条插值得到的成本为 96.41。故我们可以断定最小成本一定小于 100。所以，在编码产生初始种群时我们控制了 1 的个数产生的概率（等于 0.3）。这样做的优点是：1、有效的增加了问题求解时的效率，具体来说，即在产生初始种群时提前过滤掉适应度较低的个体（取 15 个点，定标成本已经达到了 180，肯定不是最低成本）。2、产生 1 的个数概率控制在 0.3 可以有效地为该算法后面的变异环节提供多样的变异资源，不至于导致种群变异速率过慢，只找到了局部最优解的问题。

3.2 适应度函数设计

适应度函数的设计要遵循目标值函数的规律。在本例中目标值即成本。成本包括两部分：误差成本与定标成本。成本的计算公式如下：

单点误差成本：$$s_{i,j}=\begin{cases}0 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|\le 0.5\\ 0.5 & \text{if } 0.5<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 1\\ 1.5 & \text{if } 1<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 2\\ 6 & \text{if } 2<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 3\\ 12 & \text{if } 3<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 5\\ 25 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|>5\end{cases} \quad (1)$$

单点定标成本：$s_n=n\,q$（q 取值为 12） (2)

某样本总成本：$$S_i=\sum_{j=1}^{51}s_{i,j}+q\cdot n_i \quad (3)$$

依据以上三个公式，我们可以得到目标值函数（详见附录 calobjvalue 函数）。通过目标值函数的计算，我们发现绝大部分较优的成本值落在 100 左右，依据成本越小，方案越合理这一简单思路，我们不难发现适应度函数应与目标值函数成负相关。而当成本小较小时，不同方案的成本相差很小，但我们也要通过适应度函数扩大其差距，从而选择出最好的方案。延续如上思路，我们发现反比函数可以很好地符合要求，所以我们将适应度函数确定如下（其中 Su 为适应度，Si 为目标值）：

$$Su_i=100/Si \qquad (4)$$

3.3 选择参数与方法的选定

计算出适应度函数之后就可以进行选择了。本例采用的轮盘赌方法进行选择（代码详见附录）。轮盘赌方法由赌桌上的轮盘得到思路：将一整块圆盘按不同比例分为不同大小，随机投掷小球，通过小球落入的区域进行选择。在本课题中，所有适应度之和(Su)就是整个赌桌，而每个个体的适应度(Su_i)就是被选择的面积。

$$Su=\sum_{i=0}^{n} Su_i \qquad (5)$$

我们将个体适应度与所有适应度之和的比作为该个体被选择的概率，易知其总和为1。在此我们采用累计概率的办法，在整体 1 中按比例分配不同的值域给每个个体，再产生一个 1 中的随机数（相当于随机投掷小球），该随机数落在某个个体的区域中，便选择该个体。

3.4 交叉的实现

在选择出新种群父代后，就在父代中进行交叉（交配）。本课题中我们采用单点交叉(详见附录 crossover 函数)，即在基因段中从随机一点断裂，并与另一基因进行重组。查询相关资料后[2]，我们将交叉概率 pc 定为 0.9。该概率的实现可以由简单的随机数实现：随机产生 0 到 1 之间的随机数，若该数落在了 0 到 0.9 之间，则进行交叉，否则不交叉。先将父代随机排列，并进行编号。将奇数号基因和相邻的偶数号基因结为一对。若判定二者进行交叉，则随机产生一个基因断裂点，将二者基因重组后产生新的种群。若判定二者不进行交叉，则直接将二者输入新的种群。

3.5 变异的实现

变异比较容易实现(详见附录 mutation 函数)，在本例中，变异就是指将基因中 0、1 进行互换。首先我们给定基因变异的概率。因为变异概率只是提供多样化的选择，不能因为变异而改变基因整体的优化性，因此变异的概率应该较小，查阅相关资料后[2]，我们将变异概率 pm 定为 0.01。种群中每一个个体的每一个基因均有可能发生突变。

3.6 结果展示

通过多次模拟，种群在第 50 代左右趋于稳定，最终的模拟结果如表 3-6-1 展示。由表 3-6-1 我们不难发现，通过遗传算法求出的定标最小成本为 93.19，取点位置为 3,11,22,31,42,50。

表 3-6-1 遗传算法求解结果

（pc=0.9，pm=0.01，采用三次样条插值（spline））

次数	第一代最小值	第一代最小值取点位置	稳定后最小值	稳定后最小值取点位置
1	104.02	5 11 17 26 34 45 51	93.82	3 11 22 31 44 50
2	108.00	4 7 10 20 26 34 45 51	93.86	3 9 21 31 43 50
3	110.45	2 9 19 22 26 29 45 48	94.05	3 9 21 30 40 50
4	110.18	2 4 8 23 33 45 48	93.19	3 11 22 31 42 50
5	112.50	1 4 20 21 26 33 46 48	93.75	3 11 22 32 42 50
6	105.31	2 11 21 26 31 44 45 50	93.82	3 11 22 31 44 50
7	105.25	1 9 19 27 35 44 45 50	93.97	3 11 22 30 42 50
8	105.65	2 14 24 26 34 45 51	94.32	3 9 22 31 42 50
9	111.65	4 8 26 30 41 50	94.01	2 10 20 26 34 43 50
10	111.04	3 14 22 28 31 42 47 49	94.02	3 10 20 26 33 43 50

4 思考与拓展

4.1 轮盘赌选择法的改进

轮盘赌选择法：通过各个个体的选择概率$Ps(a_j)$，计算其累计概率。第k个个体的累计概率为$P_x(a_k)$。

$$P_x(a_k)=\sum_{j=1}^{k} p_s(a_j) \quad (6)$$

然后产生0到1之间的随机数e与$P_x(a_k)$进行比较来决定选择的个体。若$a_{k-1}<e<a_k$，则选择第k个个体。通过重复n轮来产生n个子代个体。

改进后的选择方法：直接对各个个体的生存期望数目$P(a_i)$. 向下取整得到$x(a_i)$。对并$x(a_i)$求和得到m.

$$m=\sum_{j=1}^{n} x(a_j) \quad (7)$$

此时只需要再生成(n—m)个体就可以形成完整的子代。选择前(n—m)个个体赋值为l，然后与原有对应序号向下取整得到的整数值相加，便得到该个个体实际选取个数。比如，如果第k个个体a_k在前(n—m)个体中，则第k个个体在子代中实际选择的个数为$(x(a_k)+1)$[3]。

两种方法的比较：(1)轮盘赌选择法完全依靠每次的随机数进行选择，这增加了选择的不确定性。可以说，本文的方法是轮盘赌方法的最理想情况的表现；(2)对于种群规模n=200而言，平均选择概率仅为0. 005。最佳个体被选择的概率会很低，故对于轮盘赌而言，由于其随机性，可能导致了优良个体丢失。而本文的方法则可以将优良个体保留，防止优良解的丢失；(3)轮盘赌方法使子代个体并没有完全按父代个体适应度多少进行选择，本文方法则可以。这充分体现了适应者更多繁殖，不适者少量繁殖或者不繁殖的情形，从而加快收敛（改进后代码见附录selection改进后函数）。改进后的结果如下表：

表4-1-1 轮盘赌与改进后的结果对比

（pc=0.9,pm=0.01,采用三次样条插值的拟合方法）

次数	轮盘赌最小值	轮盘赌最小值位置	改进后方法最小值	改进后方法最小值位置
1	93.82	3 11 22 31 44 50	93.42	3 10 21 30 41 49
2	93.86	3 9 21 31 43 50	92.88	3 12 22 31 43 50
3	94.05	3 9 21 30 40 50	93.26	3 12 22 32 43 50
4	93.19	3 11 22 31 42 50	93.03	3 11 22 31 43 50
5	93.75	3 11 22 32 42 50	93.18	2 11 22 31 43 50
6	93.82	3 11 22 31 44 50	92.88	3 12 22 31 43 50
7	93.97	3 11 22 30 42 50	93.03	3 11 22 31 43 50
8	94.32	3 9 22 31 42 50	92.88	3 12 22 31 43 50
9	94.01	2 10 20 26 34 43 50	93.03	3 11 22 31 43 50
10	94.02	3 10 20 26 33 43 50	92.88	3 12 22 31 43 50

4.2 多种拟合方法的比较

4.2.1 三次样条插值（spline 插值）

三次样条插值（简称 spline 插值）：是通过一系列形值点的一条光滑曲线，数学上通过求解三弯矩方程组得出曲线函数组的过程。

由三次样条插值进行拟合的结果见表 3-6-1;

4.2.2 三次多项式拟合（polyfit 拟合）

多项式直接拟合的优点在于：运算简单，计算速度快，短时间内搜索范围广；

其明显缺点在于：由少数点确定的多项式在整体的适应度上表现不佳，曲线走向与实际数据点偏差往往较大。

拟合结果如下表（表 4-2-1）

表 4-2-1 三次多项式拟合结果

次数	最小值	最小值位置
1	113.04	4 18 29 41 49
2	113.30	4 16 29 39 49
3	113.83	4 18 29 39 49
4	113.30	4 16 29 39 49
5	113.47	3 15 26 39 49

4.2.3 三次多项式插值（pchip 插值）

常用拟合中较为精确的一种，本课题中拟合结果见下表（表 4-2-2）

表 4-2-2 三次多项式插值拟合结果

次数	最小值	最小值位置
1	83.49	4 15 26 35 48
2	83.61	5 15 26 35 48
3	83.62	3 15 26 35 48
4	84.16	4 15 26 36 48
5	84.27	4 15 26 35 47

4.2.4 三种不同拟合方法的比较

表 4-2-3 三种不同拟合方法的对比

拟合方法	最小成本	最小成本对应位置
三次多项式拟合（polyfit）	113.04	4 18 29 41 49
三次样条插值（spline）	93.19	3 11 22 31 42 50
三次多项式插值（pchip）	83.49	4 15 26 35 48

通过表 4-2-3，不难看出，采用三次多项式拟合（polyfit）得到的结果精度最差，而采用三次样条插值（spline）得到的结果次之，采用三次多项式插值的拟合方法（pchip）可以得到结果最精确。故在实际定标中，应尽量采用三次多项式插值（pchip）的拟合方法。

5 结论

本课题是为某投入批量生产的电子产品的检测模块进行成本合理的传感特性校准。由于传感器特性存在个体上的差异，因此暴力列举方法是不可行的，这就将我们引向了启发式搜索算法——遗传算法。通过遗传算法的拟合，我们得出结论如下：在对传感器定标时选取点的位置为 4,15,26,35,48 并采用三次多项式插值作为拟合方法时可以得到平均最低定标成本 83.49。而该方法可以高效、精确地解决许多同类问题。

6 参考文献

[1]春花,王海珍. 遗传算法的原理及组成分析[J]. 内蒙古民族大学学报(自然科学版),2009,06:632-634.
[2]陈昆,石国桢. 浮点数编码遗传算法变异概率的选取[J]. 武汉理工大学学报(交通科学与工程版),2001,04:496-499.
[3]梁宇宏,张欣. 对遗传算法的轮盘赌选择方式的改进[J]. 信息技术,2009,12:127-129.

7 附录

7.1 best 函数

```
function [bestgene,bestobjvalue,bestfit] = best( pop,fitvalue)
%BEST 取出种群中最好的基因，及其目标函数值和适应度
[px,py]=size(pop);
bestgene=zeros(1,py);
bestfit=max(fitvalue);
for i=1:px
    if fitvalue(i)==bestfit
        bestgene=pop(i,:);
        bestfit=fitvalue(i);
    end
end
bestobjvalue=100/bestfit;
end
```

7.2 calfitvalue 函数

```
function [ fitvalue ] = calfitvalue(objvalue)
%CALFITVALUE 根据目标函数计算适应度
%这里采用 100 除以目标函数值的方法
m=size(objvalue,1);
fitvalue=zeros(m,1);
fitvalue=100./objvalue;
end
```

7.3 calobjvalue 函数

```
% 标准样本原始数据读入
minput=dlmread('20141010dataform.csv');
[M,N]=size(minput);
nsample=M/2; npoint=N;
x=zeros(nsample,npoint);
y0=zeros(nsample,npoint);
y1=zeros(nsample,npoint);
for i=1:nsample
    x(i,:)=minput(2*i-1,:);
    y0(i,:)=minput(2*i,:);
end

for k=1:px
    gene=pop(k,:);%取出一个个体的基因
```

```
    n_gene=size(gene(logical(gene)),2);%测定点的个数
    % 定标计算
    index_temp=logical(gene);%记录基因中值为 1 的位置
    x_optimal=x(:,index_temp);%取出在 index_temp 中值为 1 的位置
    y0_optimal=y0(:,index_temp);%取出在 index_temp 中值为 1 的位置
    for j=1:nsample
        % 调用模拟函数 mycurvefitting
        y1(j,:)=mycurvefitting(x_optimal(j,:),y0_optimal(j,:));
    end

    % 成本计算
    Q=12;
    errabs=abs(y0-y1);
    le0_5=(errabs<=0.5);
    le1_0=(errabs<=1);
    le2_0=(errabs<=2);
    le3_0=(errabs<=3);
    le5_0=(errabs<=5);
    g5_0=(errabs>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)+Q*ones(nsample,1)*n_gene;
    cost=sum(si)/nsample;
    objvalue(k,1)=cost;
end
end
```

7.4 crossover 函数

```
function [ newpop ] = crossover( pop,pc )
%CROSSOVER 交叉产生新种群，pc 为交叉概率
[px,py]=size(pop);

%交换各行的顺序
pop1=zeros(px,py);
randmm=randperm(px);
for i=1:px
    pop1(i,:)=pop(randmm(i),:);
end

%完成各基因交换，采取单点交叉。
newpop=zeros(px,py);
pxx=floor(px/2);
for i=1:pxx
    if(rand()<pc)
        randnn=floor(rand(1)*(py-1)+1);
```

```
            newpop(2*i-1,1:randnn)=pop1(2*i-1,1:randnn);
            newpop(2*i,1:randnn)=pop1(2*i,1:randnn);
            newpop(2*i-1,(randnn+1):py)=pop1(2*i,(randnn+1):py);
            newpop(2*i,(randnn+1):py)=pop1(2*i-1,(randnn+1):py);
        else
            newpop(2*i-1,:)=pop1(2*i-1,:);
            newpop(2*i,:)=pop1(2*i,:);
        end
end

end
```

7.5 go 函数

```
%统计推断 4.0 程序，24 组 Might。
%设定参数
popsize=200;genelength=51;pc=0.9;pm=0.01;n=100;%种群大小 200，基因长度 51，交叉概率 0.9，突变概率 0.01，繁衍 100 代

%初始化种群
pop=initpop(popsize,genelength);

%结果显示
fprintf('%s\t\t','G');fprintf('%s\t\t','mincost');fprintf('%s\n','position');
for i=1:n
    objvalue=calobjvalue(pop);%计算成本，即目标函数值
    fitvalue=calfitvalue(objvalue);%计算适应度
    [bestgene,bestobjvalue,bestfit]=best(pop,fitvalue);%获取种群中最好的基因，成本，适应度
    pop=selection(pop,fitvalue);%选择产生新种群
    pop=crossover(pop,pc);%交叉产生新种群
    pop=mutation(pop,pm);%变异产生新种群

    %结果输出
    fprintf('%d\t\t',i);
    fprintf('%5.2f \t\t',bestobjvalue);
    fprintf('%d \t',find(bestgene==1));
    fprintf('\n');
end
```

7.6 initpop 函数

```
function [pop] = initpop(popsize,genelength)
%INITPOP 产生初始种群
%输入参数：popsize 为种群大小，genelength 为基因长度
%输出：pop 为 popsize*genelength 的矩阵，该矩阵中均为 0 和 1（二进制表示）
pop=round(rand(popsize,genelength)-0.3);
end
```

7.7 mutation 函数

```
function [ newpop ] = mutation( pop,pm )
%MUTATION  突变函数，由原始种群及突变概率产生新种群。pm 为突变概率
%     基因中每一位都以 pm 的概率变异
[px,py]=size(pop);
newpop=ones(size(pop));
for i=1:px
    if(rand<pm)
        mpoint=round(rand(1,py));
        newpop(i,:)=pop(i,:);
        for j=1:py
            if (mpoint(j)==1)
                if(newpop(i,j)==1)
                    newpop(i,j)=0;
                else
                    newpop(i,j)=1;
                end
            end
        end
    else
        newpop(i,:)=pop(i,:);
    end
end
end
```

7.8 mycurvefitting 函数

```
function y1 = mycurvefitting( x_premea,y0_premea )

x=[5.0:0.1:10.0];

% 根据定标计算方法，产生每个点的估算值
y1=interp1(x_premea,y0_premea,x,'spline');

end
```

7.9 selection 函数

```
function [newpop] = selection(pop,fitvalue)
%SELECTION  按照适应度选择优秀个体形成新种群
[px,py]=size(pop);
totalfit=sum(fitvalue);%求适应值之和
fitvalue1=zeros(px,1);
fitvalue1=fitvalue/totalfit;%单个个体被选择的概率
fitvalue1=cumsum(fitvalue1);%累积概率

ms=sort(rand(px,1)); %从小到大排列，将"rand(px,1)"产生的一列随机数变成轮盘赌形式的表示方法，由小到大排列
fitin=1;%表示位置
```

```
newin=1;%表示位置

while newin<=px
    if (ms(newin))<fitvalue1(fitin)
        newpop(newin,:)=pop(fitin,:);newin=newin+1;
    else
        fitin=fitin+1;
    end
end
bestfitvalue1=max(fitvalue);
for i=1:px
    if fitvalue(i,1)>=bestfitvalue1
        newpop(i,:)=pop(i,:);
    end
end
end
```

7.10 selection 改进后函数

```
function [newpop] = selection(pop,fitvalue)
%SELECTION  按照适应度选择优秀个体形成新种群
[px,py]=size(pop);
newpop=zeros(px,py);
fitvalue1=floor(fitvalue./sum(fitvalue).*px);
for i=1:(px-sum(fitvalue1))
    fitvalue1(i,1)=fitvalue1(i,1)+1;
end
n=1;
for m=1:px
    i=fitvalue1(m,1);
    while i>=1
        newpop(n,:)=pop(m,:);
        n=n+1;i=i-1;
    end
end
end
```