**一个测量装置在大规模制造中的标定问题**

——基于插值拟合与禁忌遗传

戴天杰 519021910734

摘 要：本文从工业界测量的装置标定问题出发，将其抽象为一个组合优化问题。在问题求解的初阶过程中，笔者借助建模软件MATLAB，使用三次样条插值法和遗传算法迭代出了贴合实际情况的标定方案。在后续章节，笔者进一步探讨了插值方法对结果的影响，并借助禁忌算法深化了对本问题的认识。

关键词：标定，插值，拟合，遗传算法，禁忌搜索算法

**Calibration of measuring devices**

**in large scale manufacturing**

**ABSTRACT:** In this paper, the calibration problem of industrial measuring devices is abstracted as a combinatorial optimization problem. In the initial stage of the problem-solving process, the author uses the modeling software MATLAB, cubic spline interpolation method and genetic algorithm to iterate out the calibration scheme fitting the actual situation. In the following chapters, the author further discusses the influence of interpolation method on the accuracy of the results, and with the help of tabu algorithm to deepen the understanding of the problem.

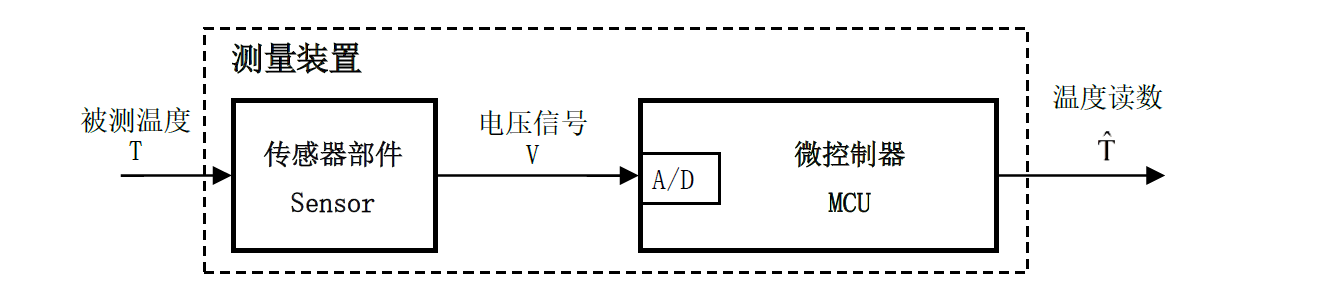
**Keywords:** Calibration, interpolation, fitting, *Genetic Algorithm*, *Tabu Algorithm*

1 引言[1]

1.1 问题背景

许多电子产品中均含有监测某种物理量的测量装置，例如温度、压力或光强等。任何测量装置在出厂验核时都需要经过标定这一步。

在本案例中，老师们提出了一种温度检测装置，量程是-20℃至70℃。其所用的核心传感器的输入-输出特性呈现出非常明显的非线性，且个体差异性显著。因而，本案例要求我们设计出一种协助生产商标定装置的流程，能够适用于大规模、高效率地批量生产与制造。



**图1 装置原理框图**

1.2 问题引出

在本案例中，我们需要对给定数据集中的电压-温度关系进行拟合处理。一方面，我们需要考虑采取少量数据点拟合后的估计值与原来真实值的误差，显然，精度越高越好；另一方面，我们需要控制采集装置的数量，以节约成本。但是，精度高与装置数量少是不可兼得的，我们只能努力去寻找两种评判体系下的最均衡点。为了，统一衡量精度与装置数量，我们给出了包含两者的方案总体成本的概念，具体内容见1.3节。

1.3 方案成本分析

为了评估和比较不同的标定方案，老师们给出了如下成本计算规则。总体成本较低的标定方案，可认定为较优方案。每个种群中的个体，要结合已知的500组数据计算标定成本。

1.3.1 单点测定成本

实施一次单点测定成本为。

1.3.2 标定误差成本



其中表示估计值，即通过拟合后得到的第个样本、第点对应的估计数值；为第个样本、第点对应的真实数值。

由此可得单个样本个体的标定误差成本：



1.3.3 样本个体标定成本



每个样本的标定成本由测定成本与误差成本组成，表示第个样本中标定点的数目。

1.3.4 方案总体成本



方案的总体成本即为所有样本个体标定成本的平均。

2 插值拟合方法

插值与拟合是数据处理中常用手段，在实际中，我们往往利用部分已知数据，借助于一定的规则与算法，来推出整体情况。这一方面有助于节约采样成本，但另一方面会引起误差。MATLAB中常见的插值方法有线性插值、三次样条插值以及Hermite方法等，在课题研究初阶，笔者选用三次样条插值法进行插值操作。[2]

2.1 拟合概念与定义

拟合是指依靠数学方法，将科学或工程上通过实验获得的实际数据用更为密集的离散方

程或连续函数来描述的过程。但不可避免的，拟合会带来一定的误差。在一般情况下，我们常常采用拟合优度来评判拟合好坏，最小二乘法则是其中的精髓。

2.2 三次样条插值[3]

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。

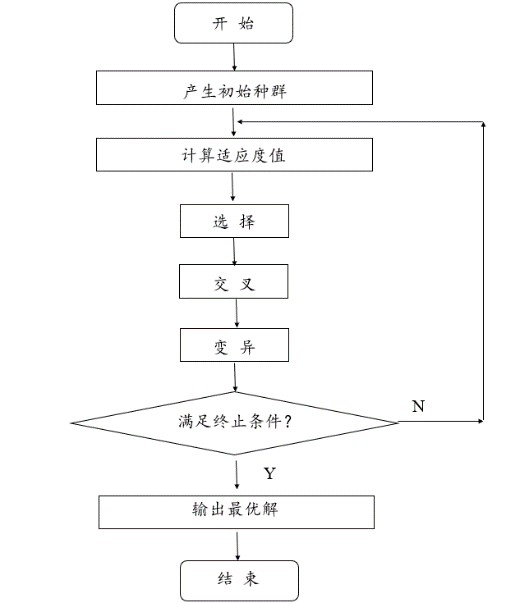
在实际计算时还需要引入边界条件才能完成计算。边界通常有自然边界（边界点的二阶导为0），夹持边界（边界点导数给定），非扭结边界（使两端点的三阶导与这两端点的邻近点的三阶导相等）。一般的计算方法书上都没有说明非扭结边界的定义，但数值计算软件如MATLAB都把非扭结边界条件作为默认的边界条件。

3 基于遗传算法的建模求解

遗传算法（*Genetic Algorithm*，GA）最早是由美国的*John holland*于20世纪70年代提出,该算法是根据大自然中生物体进化规律而设计提出的，是模拟[达尔文](https://baike.baidu.com/item/%E8%BE%BE%E5%B0%94%E6%96%87/23890" \t "_blank)[生物进化论](https://baike.baidu.com/item/%E7%94%9F%E7%89%A9%E8%BF%9B%E5%8C%96%E8%AE%BA/143686)的[自然选择](https://baike.baidu.com/item/%E8%87%AA%E7%84%B6%E9%80%89%E6%8B%A9/1800930)和[遗传学](https://baike.baidu.com/item/%E9%81%97%E4%BC%A0%E5%AD%A6/233918)机理的[生物进化](https://baike.baidu.com/item/%E7%94%9F%E7%89%A9%E8%BF%9B%E5%8C%96/3311428)过程的计算[模型](https://baike.baidu.com/item/%E6%A8%A1%E5%9E%8B/1741186)，是一种通过模拟自然进化过程搜索[最优解](https://baike.baidu.com/item/%E6%9C%80%E4%BC%98%E8%A7%A3/5208902)的方法。遗传算法已被人们广泛地应用于组合优化、机器学习、信号处理、自适应控制和人工生命等领域。[4]对于以本案例为代表的NP-hard问题，相较于可能计算量爆炸的穷举算法，以遗传算法为代表的启发式算法自有它的优势。

3.1 流程概述

遗传算法包括初始化种群、适应度计算、自然选择、交叉互换、基因变异等过程，具体流程如图2所示。



**图2 遗传算法流程图**

3.2 实施细节

在本案例中，笔者采取二进制编码。由于对应数据集给出了90个标定点，染色体的长度即为二进制数的长度90。再按照流程图世代更迭，便可得最终理想解。步骤如下：

3.2.1 初始化种群的操作

初始化种群是开展遗传算法求解工作的第一步。相关参数选择的好坏，对于该模型的适用性、高效性与准确性都有着重要的影响。

笔者首先将种群规模设置为100，由于每个个体具有90个基因，因而种群基因即为一个100行，90列的矩阵，在矩阵中，1代表选取该点参与拟合，0代表不参与。第一代种群的基因利用随机函数获得，从而能满足遗传算法随机性的要求。最大迭代代数设置成了60，鉴于先前测试时算法大概在第50轮后开始较快收敛，故选择60作为参数。

3.2.2 适应度函数的选取

达尔文学说强调，优胜劣汰，用进废退。这句话道出了自然界适者生存的普遍真理。假想上帝在幕后操纵着一切，那么适应度函数就是他择优的一项参考指标。

在本案例中，方案总体成本起着衡量标定方案好坏的作用。因而，适应度函数可以在方案总体成本的基础上进行改编，考虑到成本越高，标定方案越不可靠，我们需要尽可能多的保留成本不高的个体。因而，适应度函数被定义为该方案下总体成本平方的倒数。



3.2.3 自然选择的模拟

选择的过程就是上帝依据适应度函数来决定个体去留的过程。显然，适应度越高，对应的基因型就越符合我们的要求，被选中存活的概率就越大。为此，笔者首先将单个个体的适应度除以总的适应度，确定为个体被选中的概率。



其中，为第个个体的适应度，为适应度之和。显然，每个概率唯一对应0~1上的一段范围。通过轮盘赌法，可以在从0至1的范围内产生一个随机数，被选中的个体即表示未被淘汰，可以在下一轮中充当父母，产生子代，实现了使得优秀基因尽可能代代相传的目的。

3.2.3 精英个体的保留

为防止适应度高的基因因为后期的繁育过程而丢失，笔者采取了精英保护机制，即在选择之前，把适应度最高的8个个体直接复制到了下一代。之所以仅仅选择8个最优个体作为父本、母本直接参与下一代，是因为倘若精英个体数目选择过多，将会导致之后个体可能产生的新优势难以凸显出来。

3.2.4 交叉互换

交叉互换是指父本、母本进行繁殖活动，亲本的染色体之间出现交叉从而产生具有新基因型个体的过程。

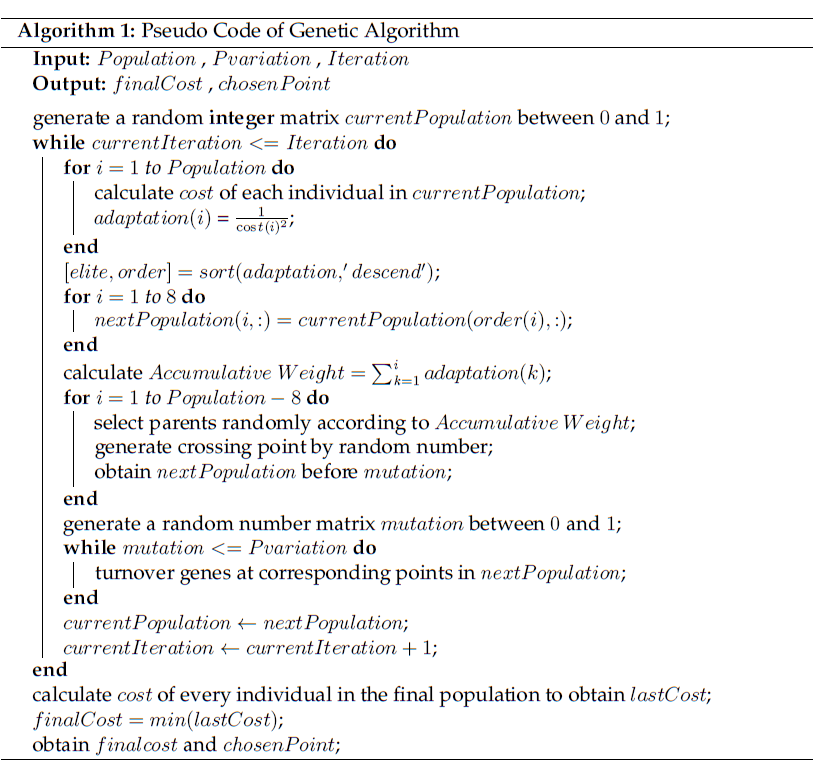
为了尽可能真实地模拟真实环境，笔者将交叉概率设置为1。与此同时，为了简化操作，笔者采取单点交叉法，即在随机产生的交换位点前，两个染色体的基因不发生改变，而之后部分整体交换，希望借此来不断产生具有新基因型的新种群个体。

3.2.5 基因突变

生物体的基因型在某些时刻偶尔也会发生改变，但异变率往往并不高，且往往都是有害变异。在本案例中，笔者将变异率设置为0.001。当随机数矩阵中格子的数值低于该概率，那么该格子对应的基因将会发生翻转，由于采用二进制编码，相关基因的数值将会从0变1，或者1变0。

3.3 代码流程

在上述思想的指导下，笔者进行了MATLAB代码实现。主函数代码的流程与结构如图3所示。



**图3 遗传算法伪代码**

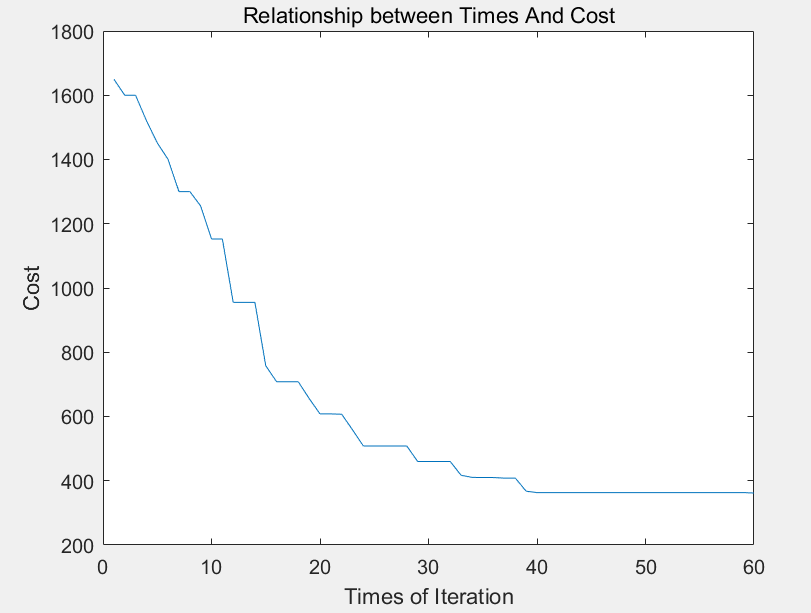
3.4 结果探讨

代码实现后，笔者进行了多次仿真实验，仿真结果如表1所示。显然，在多次仿真后，基于三次样条插值的遗传算法求解出的最小成本约为360，取点为-16、-2、13、22、42、61、70。

**表1 多次仿真不同取样点及成本计算**

|  |  |  |  |
| --- | --- | --- | --- |
| 次 数 | 1 | 2 | 3 |
| 取 点 | -17、6、15、29、32、59、67 | -17、6、15、29、32、53、69 | -16、-2、13、22、42、61、70 |
| 成 本 | 361.7 | 362.148 | 360.9060 |

随着迭代次数的增高，成本逐渐下降，这符合我们的预期。经过大概四十余次的迭代，成本的变化量趋近为0。表1中第三次实验时迭代次数与成本的可视化如图4所示。



**图4 迭代次数与成本的关系**

4 拓展一：不同插值方式对最低成本的影响[5]

在第2部分，笔者已经提及，MATLAB中常见的插值方法有线性插值方法、三次样条插值方法以及Hermite方法等。不同种方法的拟合插值方式各异，能够带来不一样的运算求解结果。在课题研究初阶，笔者选用了三次样条插值法。在本章节，笔者依次选用较为常见的线性插值与Hermite插值方式进行对比实验。

4.1 线性插值

在数值计算中，线性插值是上手容易，对新手较为友好的一种插值方法。其优点在于公式表现形式较为简单，比较易于理解。但美中不足的是，该方法拟合出来的曲线形式是分段折线，而在本案例的工程背景中，电压-温度关系呈现出极为明显的非线性，因而误差较其他拟合插值方式而言较大。具体数据如表2第1列所示。

4.2 Hermite插值

在数值计算中，线性插值是上手容易，对新手较为友好的一种插值方法。其优点在于公式表现形式较为简单，比较易于理解。但美中不足的是，该方法拟合出来的曲线形式是分段折线，而在本案例的工程背景中，电压-温度关系呈现出极为明显的非线性，因而误差较其他拟合插值方式而言较大。具体数据如表2第2列所示。

**表2 三种插值方式结果比对**

|  |  |  |  |
| --- | --- | --- | --- |
| 方 法 | *Linear* | *Hermite* | *Spline* |
| 取 点 | -19、-13、-5、8、12、25、44、59、70 | -19、-13、9、25、37、66 | -16、-2、13、22、42、61、70 |
| 成 本 | 464.6680 | 314.7200 | 360.9060 |

4.3 插值方法小结

从结果上看，Hermite法表现出了较为明显的优势，成本显著低于其他插值方式，但这并不意味着其他插值方法一无是处。对于数据量较少的常规任务，线性插值仍是友好度与性价比最高的一种方法；如果要求插值拟合后结果的总体平滑度，那么Hermite插值与三次样条插值是首选。但它们又各有明显的缺点，分段三次Hermite插值要求每个拐点处函数斜率已知，而三次样条插值求解代价往往较大、精确度会受到端点条件的影响。由于MATLAB默认了一些边界条件，才使得求解显得如此容易，在实际生活与工作中，我们需要根据不同的情况选择适用度最好的插值方式进行拟合。

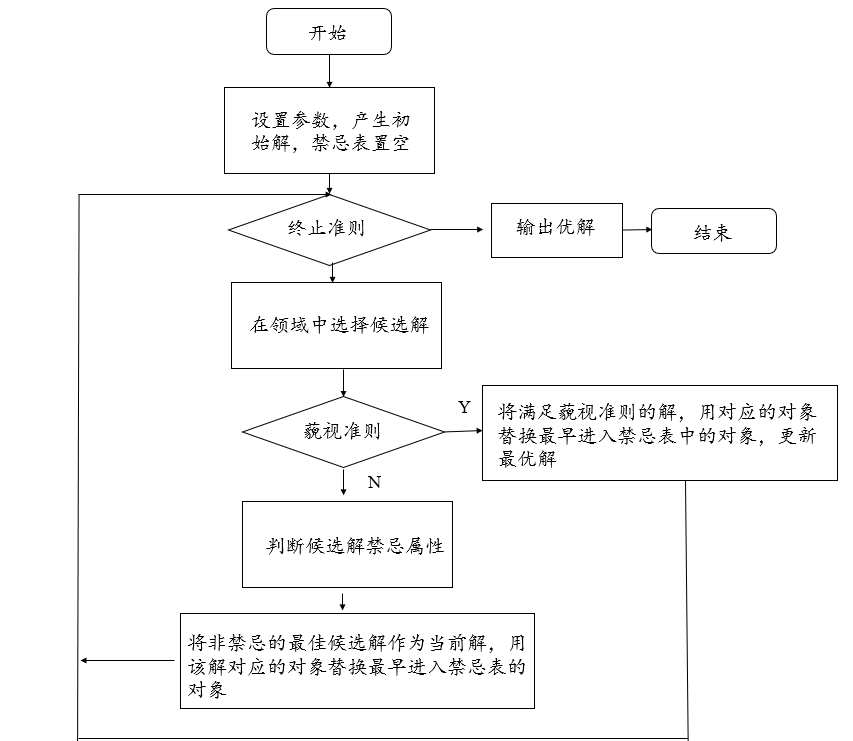
5 拓展二：禁忌搜索在本案例中的延伸应用

当然，不同的插值方式仅仅是浅层的试探性研究，更为重要的是算法上的优化。启发式算法包括很多，例如模拟退火算法、蚁群算法，但这些算法的运算量与耗时往往较大。笔者在此选用了禁忌搜索法。

禁忌算法最大的优势在于能够“跳出舒适圈，放眼望世界”，也就是说，该算法相对传统的遗传算法而言，更有可能突破局部最优解，取得全局意义上的最优。这是禁忌算法最令人称道的地方。

禁忌算法之所以更有希望取得全局最优解，其精髓在于“刀刃向内”。对于暂时寻找的局部最优解，该算法能够暂时忽略之，将目标投向更为广阔的搜索空间，从而更能保证探索的有效性。

5.1 实施细节[6]



**图5 禁忌搜索流程图**

如图5所示，禁忌搜索包括给定参数、置空表、终止准则判断、藐视准则判定等步骤。兼顾之前的遗传算法，禁忌搜索算法的具体步骤如下：

5.1.1 解空间的构建

显然，解空间中包含90个点。取值为1，则代表选中作为测量用点进行插值拟合计算成本，否则代表不被选中，初始解空间可以自由定义，可以用来验证先前的运算结果。

5.1.2 邻域的选取

在一条染色体上，随机选择出和两个不同的点位，然后进行交换。与此同时，遗传算法中的变异依旧适用，异变概率置为0.001，发生变异的位置基因翻转。

5.1.3 禁忌表的构建

邻域对象为，禁忌表长度取，禁忌表被设置为循环队列的数据结构。

我们最初需要初始化列表，即。在候选集合中挑出向量，若该解之前不在列表中，且队列未满，则进队；否则，最早进入的解出队，留出空间给进队。

5.1.4 特赦原则

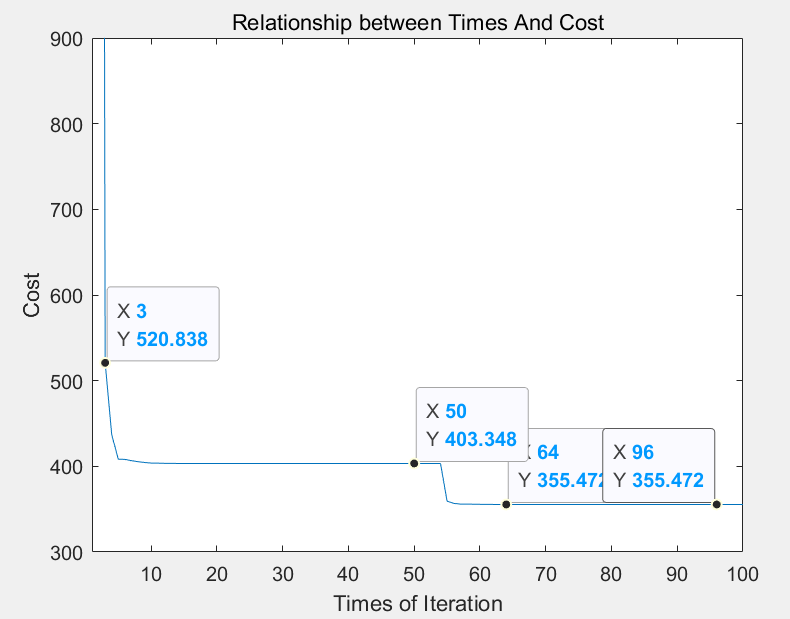
若偏偏候选集合中的所有内容都已经被循环队列所存储，此时，我们需要调用“特赦”。调用时，最优解进队，排在最末端的解则出来。

5.2 结果探讨

如图6所示，禁忌搜索算法的优势还是较为明显的。函数自身运行时间非常短，100代仅需60多秒。与此同时，如表3所示，多次任意改变初始解空间，经过运行，代码最终运算结果均一致，这说明了禁忌遗传算法取得的结果有很大把握可以确定为是全局最优解，该算法的稳定性非常高。MATLAB在进行迭代运算求解时，并没有被禁锢在局部最优的范畴之内，恰恰相反，借助于禁忌搜索的遗传算法较传统的启发式算法而言跳跃性更强了，更不容易“一叶蔽目，不见泰山”，能够既快又好地取得全局最优。

**表3 多次仿真不同取样点及成本计算**

|  |  |  |  |
| --- | --- | --- | --- |
| 次 数 | 1 | 2 | 3 |
| 取 点 | -16、6、13、24、39、57、69 | -16、6、13、24、39、57、69 | -16、6、13、24、39、57、69 |
| 成 本 | 355.4720 | 355.4720 | 355.4720 |



**图6 禁忌搜索算法迭代次数与成本的关系**

6 致谢

首先感谢袁焱、李安琪老师组织的MATLAB软件讲解课程以及面谈活动，这为我们自主开展MATLAB实验奠定了基础；感谢李安琪老师对初稿的评论，为我修改终稿格式提供了很多参考；感谢于刚、李林果学长的工作思路，为笔者实现初稿的基本要求指明了方向；感谢冯大虎学长、参加面谈的李则骞同学，他们对于笔者后期拓宽眼界，了解禁忌搜索与模拟退火算法很有帮助。

参考资料

1. 上海交通大学电子工程系. 工程问题建模与仿真课程案例一说明.
2. 拟合-百度百科.
3. 三次样条插值-百度百科.
4. 遗传算法-百度百科.
5. 五种插值法的对比研究.百度文库https://wenku.baidu.com/view/538e3c52793e0912a21614791711cc7931b7788c.html
6. 禁忌搜索与Python实现. https://blog.csdn.net/adkjb/article/details/81712969

附 录

**Appendix A costofallsample.m（**基本成本计算函数**）**

**———————————————————————————————————————————**

function c1 = costofallsample(Matrix)

table = dlmread('dataform\_train.csv');

Content= table(1,:);

Number = table(2:2:1000,:);

equal1 = find(Matrix);

Numberchosen = Number(:, equal1);

Contentchosen = Content(:, equal1);

residual = zeros(500,90);

costsingle = zeros(500,90);

n = 1;

for j = 1:90

if ( Matrix(1,j) == 1)

n = n + 1;

end

end

for i = 1:500

residual(i,:) = abs(Content-interp1(Numberchosen(i,:),Contentchosen,Number(i,:),'spline'));

end

for i = 1:500

for j = 1:90

if (residual(i,j) <= 0.5)

costsingle(i,j) = 0;

end

if (residual(i,j) > 0.5 && residual(i,j) <= 1.0)

costsingle(i,j) = 1;

end

if (residual(i,j) >1.0 && residual(i,j) <= 1.5)

costsingle(i,j) = 6;

end

if (residual(i,j) >1.5 && residual(i,j) <=2.0)

costsingle(i,j) = 20;

end

if (residual(i,j) >2.0)

costsingle(i,j) = 10000;

end

end

end

c1 = sum(costsingle(:))/500 + 50 \* n;

return;

end

**———————————————————————————————————————————**

**Appendix B main.m（**基本功能主函数**）**

**———————————————————————————————————————————**

clc;

clear;

tic;

Population = 100;

Pvariation = 0.001;

Iteration = 60;

currentIteration = 1;

currentPopulation = randi([0,1], Population, 90);

costRecord = zeros(Iteration, 1);

while(currentIteration <= Iteration)

Adaptation = zeros(Population, 1);

for i = 1:Population

Adaptation(i) = costofallsample(currentPopulation(i,:));

end

costRecord(currentIteration) = min(Adaptation);

Adaptation = 1./ Adaptation.^2;

[elite,order] = sort(Adaptation,'descend');

nextPopulation = zeros(Population, 90);

nextPopulation(1, :) = currentPopulation(order(1), :);

nextPopulation(2, :) = currentPopulation(order(2), :);

nextPopulation(3, :) = currentPopulation(order(3), :);

nextPopulation(4, :) = currentPopulation(order(4), :);

nextPopulation(5, :) = currentPopulation(order(5), :);

nextPopulation(6, :) = currentPopulation(order(6), :);

nextPopulation(7, :) = currentPopulation(order(7), :);

nextPopulation(8, :) = currentPopulation(order(8), :);

for i = 2:Population

Adaptation(i) = Adaptation(i) + Adaptation(i - 1);

end

select = zeros(Population, 1);

for i = 1:Population - 8

tmp = find((Adaptation / Adaptation(Population))>=rand());

select(tmp(1)) = select(tmp(1)) + 1;

end

startrow = 9;

while (sum(select) ~= 0)

a = find(select);

b = randi(length(a));

dad = a(b);

b = randi(length(a));

mom = a(b);

select(dad) = select(dad) - 1;

select(mom) = select(mom) - 1;

location = randi(89);

nextPopulation(startrow, 1:location) = currentPopulation(dad, 1:location);

nextPopulation(startrow, (location + 1):90) = currentPopulation(mom, (location + 1):90);

nextPopulation(startrow + 1, 1:location) = currentPopulation(mom, 1:location);

nextPopulation(startrow + 1, (location + 1):90) = currentPopulation(dad, (location + 1):90);

startrow = startrow + 2;

end

mutation = rand(Population, 90) < Pvariation;

nextPopulation = abs(mutation - nextPopulation);

currentPopulation = nextPopulation;

currentIteration

currentIteration = currentIteration + 1;

end

lastCost = zeros(Population, 1);

for i = 1:Population

lastCost(i) = costofallsample(currentPopulation(i, :));

end

finalCost = min(lastCost)

chosenIndividual = currentPopulation(lastCost == finalCost,:);

chosenPoint = find(chosenIndividual(1, :)) - 20

toc;

figure;

plot(1:currentIteration-1, costRecord);

title('Relationship between Times And Cost');

xlabel('Times of Iteration');

ylabel('Cost');

**———————————————————————————————————————————**

**Appendix C costofallsample.m（**拓展部分成本计算**）**

**———————————————————————————————————————————**

function costofallsample = costofallsample(Input)

global Content;

x = [];

y = [];

for count = -20 : 69

if(Input(count+21) == 1)

x = [x, count];

end

end

Long = size(Content, 1);

costofallsample = 0;

for count = 1 : (Long/2)

y = [];

for j = -20 : 69

if(Input(j+21) == 1)

y = [y, Content(count\*2,j+21)];

end

end

yNihe = Content(count\*2,:);

x;

xNihe = interp1(y,x,yNihe,'spline');

for j = 1 : size(yNihe,2)

if(abs(Content(count\*2-1,j)-xNihe(j))<=0.5)

elseif(abs(Content(count\*2-1,j)-xNihe(j))<=1)

costofallsample = costofallsample + 1;

elseif(abs(Content(count\*2-1,j)-xNihe(j))<=1.5)

costofallsample = costofallsample + 6;

elseif(abs(Content(count\*2-1,j)-xNihe(j))<=2)

costofallsample = costofallsample + 20;

else

costofallsample = costofallsample + 10000;

end

end

end

costofallsample = costofallsample/(Long/2) + size(x,2) \* 50;

end

**———————————————————————————————————————————**

**Appendix D main2.m（**拓展功能主函数**）**

**———————————————————————————————————————————**

clear

clc

tic

global Content;

Content = dlmread( 'dataform\_train.csv' ) ;

currentIteration = 0;

rand('state',sum(clock));

Dad = zeros (1,90) ;

Tabu = zeros(60,90) ;

TabuNumber = 0;

Solution = zeros(1,90);

chosenPointCost = 1e7;

Cost = [];

for i = 1:90

if(rand()<0.1)

Dad(1,i)=1;

end

end

if (sum(Dad(1,:))<3)

Pin = randperm(90);

Dad(1,Pin(1)) = 1;

Dad(1,Pin(2)) = 1;

Dad(1,Pin(3)) = 1;

Dad(1,Pin(4)) = 1;

end

Iteration = 200;

Dad = zeros (1,90) ;

Dad([1,10,20,40,60,70,85])=1;

Nearby = zeros(50,90) ;

for k = 1 : Iteration

for i = 1 : 30

fir = 1 + floor(90\*rand(1));

sec = 1 + floor(90\*rand(1));

if(fir < sec)

Nearby(i,1:fir) = Dad(1,1:fir);

Nearby(i,1+fir:sec-1) = Dad(1,sec-1:-1:fir+1);

Nearby(i,sec) = Dad(1,sec);

if(rand(1) < 0.001)

for count = 1 : floor(rand()\*3+1)

PinI = floor(rand(1)\*90)+1;

if(Nearby(i,PinI) == 1)

Nearby(i,PinI) = 0;

else

Nearby(i,PinI) = 1;

end

end

end

if(sum(Nearby(i,:)) < 4)

PinII = randperm(90);

Nearby(i,PinII(1)) = 1;

Nearby(i,PinII(2)) = 1;

Nearby(i,PinII(3)) = 1;

Nearby(i,PinII(4)) = 1;

end

if(sum(Nearby(i,:)) > 7)

Location = find(Nearby(i,:) == 1);

PinI = floor(rand(1)\*7) + 1;

PinII = floor(rand(1)\*7) + 1;

Nearby(i,PinI) = 0;

Nearby(i,PinII) = 0;

end

else

tmp = fir;

fir = sec;

sec = tmp;

Nearby(i,1:fir) = Dad(1,1:fir);

Nearby(i,1+fir:sec-1) = Dad(1,sec-1:-1:fir+1);

Nearby(i,sec) = Dad(1,sec);

if(rand(1) < 0.001)

for count = 1 : floor(rand()\*3+1)

PinI = floor(rand(1)\*90)+1;

if(Nearby(i,PinI) == 1)

Nearby(i,PinI) = 0;

else

Nearby(i,PinI) = 1;

end

end

end

if(sum(Nearby(i,:)) < 4)

PinII=randperm(90);

Nearby(i,PinII(1)) = 1;

Nearby(i,PinII(2)) = 1;

Nearby(i,PinII(3)) = 1;

Nearby(i,PinII(4)) = 1;

end

if(sum(Nearby(i,:)) > 7)

Location = find(Nearby(i,:)==1);

PinI = floor(rand(1)\*7)+1;

PinII = floor(rand(1)\*7)+1;

Nearby(i,PinI) = 0;

Nearby(i,PinII) = 0;

end

end

end

for i = 31 : 50

Location = find(Dad==1);

Nearby(i,:) = zeros(1,90);

for count = 1 : size(Location,2)

RandomNumber = rand();

if((RandomNumber < 0.1) && Location(count) > 1)

Nearby(i,Location(count)-1) = 1;

end

if((RandomNumber > 0.9) && Location(count) < 90)

Nearby(i,Location(count)+1) = 1;

end

if((RandomNumber <= 0.9) &&(RandomNumber > 0.1))

Nearby(i,Location(count)) = 1;

end

end

end

for j = 1 : 50

TemResult = costofallsample(Nearby(j,:));

Temp(j) = TemResult;

end

[Row,Col] = sort(Temp);

currentIteration = 1;

flag = 0;

for i = 1 : TabuNumber

if(Tabu(i,:) == Nearby(Col(currentIteration),:))

flag = 1;

break;

end

end

while(flag == 1)

flag = 0;

currentIteration = currentIteration + 1;

for i = 1 : TabuNumber

if(Tabu(i,:) == Nearby(Col(currentIteration),:))

flag = 1;

break;

end

end

if currentIteration == 50

flag = 0;

currentIteration = 1;

end

end

if(TabuNumber < 60)

Tabu(TabuNumber+1,:) = Nearby(Col(currentIteration));

else

for i = 1:59

Tabu(i,:) = Tabu(i+1,:);

end

Tabu(60,:) = Nearby(Col(currentIteration),:);

end

Dad = Nearby(Col(currentIteration),:);

if (chosenPointCost > costofallsample(Nearby(Col(currentIteration),:)))

chosenPointCost = costofallsample(Nearby(Col(currentIteration),:));

Solution = Nearby(Col(currentIteration),:);

end

Cost = [Cost,costofallsample(Dad(1,:))];

end

toc

chosenPoint = find(Solution(1, :)) - 20

chosenPointCost

figure;

plot(1:Iteration,Cost);

title('Relationship between Times And Cost');

xlabel('Times of Iteration');

ylabel('Cost');

axis([1 100 300 900])

**———————————————————————————————————————————**