**一个测量装置在大规模制造中的标定问题**

组号 02 姓名 周睿文 学号 518021911150

摘 要：近年来，越来越多综合性的电子产品在市场上涌现，而这些大规模制造的电子产品中都包含有带测量功能的模块（测量装置）。任何测量装置在制造时都要经过标定的工序，其中标定点的选择能够从标定成本和产品质量两方面影响生产成本，在实际生产中是尤为需要重视的。本文将标定点的选择问题转化为组合优化问题，提出了一种基于自适应遗传算法的求解框架，并在来自一批具有显著非线性和个体差异性的温度测量装置的数据集上进行了充分的试验。此外，本文还通过数值仿真研究了不同拟合插值方法的影响，并进行了误差分析和讨论。

关键词：自适应遗传算法, 曲线拟合插值, 装置标定, 组合优化, 误差分析

**The Calibration Problem of a Measuring Device**

**in Large-Scale Manufacturing**

**ABSTRACT:** In recent years, more and more synthesized electronic products have appeared on the market, which are manufactured with a lot of modules for measuring (measuring devices). Any measuring device has to be calibrated during manufacturing, where the selection of sampling points is especially important since it can influence the cost of a company from both the cost of sampling and the accuracy of product. In this article, I formulate the selection of sampling point as a combinatorial optimization problem, propose a scheme to select the sampling points based on the self-adaptive genetic algorithm, and conduct extensive experiments on the dataset of a batch of temperature measuring devices with significant non-linearity and individual variance. Furthermore, I study the effect of different interpolation methods through numerical simulation and carry out error analysis to offer some insights.

**Key words:** Adaptive Genetic Algorithm, Curve Fitting and Interpolation, Device Calibration,

Combinatorial Optimization, Error Analysis

**1 引言**

随着电子产品制造业的发展和智能产品市场的扩张，越来越多综合性的电子产品活跃在市场上，而这些大规模制造的电子产品往往都集成有大量的传感设备，也就是本文所提到的测量装置。在测量装置的生产过程当中，标定工序是至关重要的，它能够直接决定装置的精确度，从而影响产品的市场潜力。在标定工序当中，设定更多的标定点必然能够在一定程度上提高标定的准确度，但标定点的设置也会给生产带来额外的成本，这就要求生产者在标定的精确度和生产的低成本之间做出一个权衡——既要避免过大误差造成的产品不达标，又要尽可能降低设置标定点所带来的成本。为此，生产中一般采用的方法是：设定较少的标定点，随后通过曲线的拟合插值得到剩余的待标定值。本文中，标定点的选择问题被建模为一个组合优化问题，即通过选择合适的标定点，达到总生产成本的最小化。

求解整数规划问题的常用方法是分枝定界法，该方法通过将整数规划松弛为线性规划进行分枝，随后剪去低于已知最优解的枝条，以快速求解枚举量为指数级的规划问题。然而，本文中的标定成本为分段阶梯函数，并不具有连续性，因而分枝定界法难以应用于本文研究的问题。对此，本文采取了引入自适应改进的遗传算法，以实现快速、高质量地求解标定点设置的组合优化问题。

此外，在不同的实际应用场景下，采用不同的曲线拟合插值方法对标定点的选取和最终方案的成本有着显著的影响。本文也将以案例一的场景为例，探讨不同拟合插值方法的特点及其影响，并通过误差分析说明不同拟合插值方法的适用性。

**2 标定问题的数学模型**

2.1 标定方案的成本模型

据本课程规定[1]，标定方案的成本包括标定误差成本和单点标定成本。在数据集上,对任一样本*i*，其在所表示的90个温度比较点上的单点和样本个体标定误差成本的计算规则如式（2-1）和式（2-2）所示：

（2-1）

（2-2）

其中为样本*i*在温度比较点*j*的实际温度，为标定后样本*i*在温度比较点*j*的温度估计值，为样本*i*在温度比较点*j*的单点标定误差成本，为样本*i*的个体标定误差成本，为数据集的样本总数。

对所有样本，其单点标定成本*Q*均相同，在本问题中设定*Q* = 50，由此得到样本*i*的个体标定成本如式（2-3）所示：

（2-3）

对某一标定方案，其在数据集上的方案总体成本*C*如式（2-4）所示：

（2-4）

2.2 标定点设置的组合优化问题模型

考虑到生产实际，本文假设在同一标定方案下，所有样本的标定点均设置于相同的温度比较点，因此在式（2-3）中有.

现设一标定方案，其中：

对该标定方案，所有样本在满足*xj* = 1的温度比较点*j*处的温度*T*和电压*V*的关系被认为是已知的，其余温度比较点处仅电压*V*已知。对每一样本*i*，以所有已知的数据为基础进行曲线拟合，并在实际温度未知的比较点处进行插值，即可得到一系列温度估计值，依据式（2-1）至（2-4）即可计算得到该方案的总体成本.

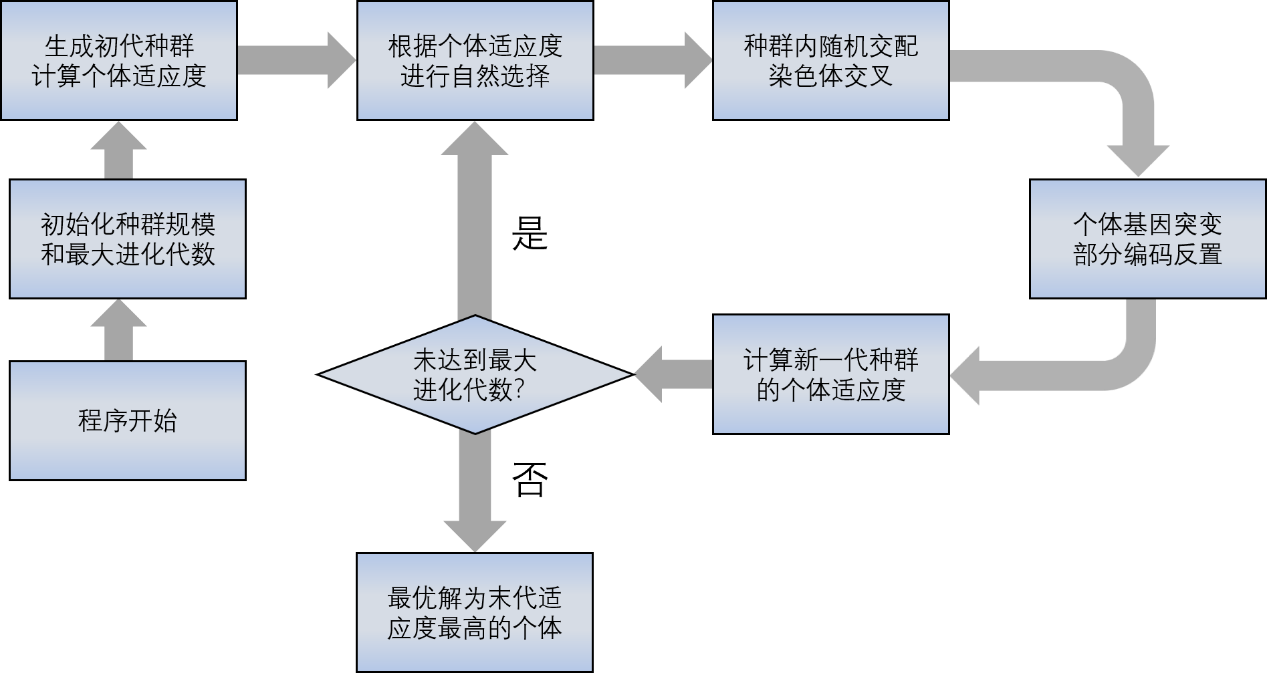
据此，本文将标定点设置问题形式化为如下组合优化问题：给定数据集，找到一个标定方案，使得

（2-5）

**3 遗传算法及其自适应改进**

3.1 标准遗传算法

标准遗传算法 (Simple Genetic Algorithm, SGA)，是美国密歇根大学教授J. Holland于1960年提出的一种借鉴遗传学基本原理、模拟生物自然进化的启发式搜索算法[2]。该算法基于达尔文的进化论，在每一轮迭代当中模拟自然界生物种群的交配、变异和选择行为，从而不断生成优于父代种群的新种群。标准遗传算法的流程如图1所示。



**图1 标准遗传算法流程**

3.1.1 参数及种群初始化

参数及种群初始化，即设定算法的两个重要参数：种群规模*M*和最大进化代数*N*，并随机生成一个初代种群*G*0。种群中个体的染色体一般被编码长度为*d*的二进制串（部分文献中也使用十进制编码），表示目标函数的一组自变量取值。遗传算法中，生成初代种群最常用的方法就是建立一个[0, 1]区间上*M ×* *d*的随机数矩阵，将其中大于0.5的位置赋值为1，小于0.5的位置赋值为0，也即随机生成了种群中的每个由*d*维行向量表示的个体。

初代种群随机产生后，需要选定合适的适应度函数*f*，用以表征该组自变量取值对目标函数优化的程度，并根据*f*计算得到初代种群中个体的适应度，用于后续的自然选择过程。在本文研究中，规定种群规模*M* = 100，最大进化代数*N* = 500，每个个体表示一种标定方案，遗传算法的适应度函数均取为标定方案在训练集上总体成本平方的倒数，即个体适应度为：

（3-1）

3.1.2选择过程

对已经计算出适应度的某一代种群*G*，标准遗传算法模仿自然界对生物的选择和淘汰机制，直接以轮盘赌的方式从当前种群中选择适应度较高的个体，其具体过程为：

设第*k*代种群为, 其中表示第*k*代种群中的第*i*个个体。根据已选定的适应度函数*f*，可以得到个体的适应度为

因此，在轮盘赌中，个体被选中的概率为：

（3-2）

在每轮选择中，均进行*M*次个体选择，每次生成一个(0, 1]区间上的随机数*r*.对当前种群，必然存在使得，则个体即为本次选择中被选中保留至下一代的个体。

3.1.3交叉过程

经过选择过程后，新的种群较多的继承了上一代适应度高的个体，但仍然种群中出现的个体种类并没有发生变化，此时通过染色体的交叉可以获得具有新表现型的个体，从而增加个体的多样性，在全局范围内寻找目标函数的更优取值。

标准遗传算法中采取染色体片段随机交叉的策略，从种群中选取两个个体（常为矩阵中的相邻个体），随机生成一个交叉点位，将交配个体在交叉点位一侧的染色体片段互换，得到两个新的个体。

3.1.4变异过程

在选择和交叉过程中，算法保留了适应度较高的个体并执行了染色体的交叉，但对于整个种群中均未出现的基因（即某一编码点位上只有0或1），这两个过程并不能带来改变。因此，算法在这一步模拟自然界中的个体基因突变，改变部分个体的染色体基因型，从而保证每个点位上都可能出现0和1，不会因为初始化过程的随机性影响寻优性能。

以基本位变异为例，设定变异概率*Pmut*，生成一个[0, 1]区间上*M ×* *d*的变异随机数矩阵，将其中值小于*Pmut*的位置赋值为1，其他位置赋值为0。对表示种群的*M ×* *d*矩阵*G*，将变异矩阵中值为1的位置上的编码反置（即1变为0，0变为1），即完成了个体的变异。

经过选择、交叉和变异三个过程，即生成了最终的新一代种群，计算新一代种群中的个体适应度，即可再一次进入选择过程。标准遗传算法将在进化迭代到最大轮数*N*时停止，并将末代种群中适应度最高的个体作为搜索到的最优自变量组合。

3.2 遗传算法的自适应改进

3.2.1 选择过程：保留并复制最优个体

在选择过程中，标准遗传算法采用轮盘赌的方法进行个体筛选，这可能导致新变异产生的优质个体还未在种群中扩张就被淘汰，因此有必要在选择过程中对特别优秀的个体给予一定程度的保护，避免其快速消亡。

本文采用的遗传算法引入并改进了精英选择策略[3]，首先挑选出前代适应度最高的个体，在下一代中复制两个完全相同的子代，随后再挑选出剩下的下一代个体。这样做不仅保留了最优个体，而且在种群趋于收敛时能够促进优质个体的扩张，有利于加速种群的进化。

3.2.2 交叉过程：自适应交叉概率

在交叉过程中，标准遗传算法采用随机交叉，这可能导致优质个体和劣质个体交叉，从而破坏新产生的优质个体。对此，受自然界许多物种繁殖时竞争配偶现象的启发，本文采用的遗传算法根据个体适应度调整不同个体参与交叉过程的策略：在交叉过程进行前，先对整个种群按适应度降序进行重排序，每次取种群矩阵中相邻（适应度接近）的两个个体进行交叉。由此，适应度高的个体可以认为是“竞争”到了优质的配偶，从而更有可能保留自身优质基因、产生优质子代；

此外，考虑到个体参与交叉可能会改变自身基因型，因此根据式（3-3）调整了不同适应度个体参与交叉的概率[4]：

（3-3）

其中为常数，本文中取表示取出的两个待交叉个体中较大的适应度，表示当前种群中的最大适应度，表示当前种群的平均适应度。式（3-3）要求，以保证交叉概率随适应度提高而降低，达到保持优质个体基因型、促进劣质个体交叉的目的。

3.2.3 变异过程：标定点横移

标准遗传算法所采用的基本位变异方法应用于本文改进后的遗传算法时存在一个问题：由于引入了增大竞争压力、加快种群收敛的改进，新变异在种群进化后期往往难以生存，更不用说通过交叉影响种群。因此，考虑到本文研究的问题中横移标定点比单点变异更可能带来改良，我借鉴十进制编码中常用的方法，在每轮基本位变异后采取如下变异策略：

设定标定点横移变异概率为（本文中取），每个个体生成一个随机数，若低于变异概率则表示该个体发生标定点横移——随机选择一个编码为1的点位，将其随机横向移动个位置。通过标定点的随机横移，接近陷入局部最优的种群很可能产生出适应度更高的个体，从而跳出局部最优。

3.2.4 个体适应度：归一化及动态调整

标准遗传算法一般采用式（3-2）中的概率进行轮盘赌，但在种群逐渐收敛的过程中，不同个体的适应度也将逐渐接近，此时轮盘赌的区分度显著降低，个体选择接近于均等概率的随机选择，这会减缓种群的收敛。因此，本文采用的遗传算法在计算出种群中个体适应度后，首先将适应度归一化，再加入动态调整因子，以提高种群进化后期的竞争压力，加快种群的收敛[5]。适应度的调整方法具体如式（3-4）：

（3-4）

其中表示调整后的适应度，表示当前种群中的最大适应度，表示当前种群的最低适应度，表示当前个体的适应度，和为常数（本文中取），为当前的进化轮数。通过动态线性调整，种群的适应度将分布在上，既保证了相近的不同个体间能够有明显的区分，又使适应度较低的个体仍有一个较小的生存概率。

3.2.5 最优记录：保存历代最优

标准遗传算法选择末代适应度最高的个体作为最优解输出，这与自然界进化的过程是一致的，但从求解最优化问题的角度仍有改进的空间。事实上，即使引入了最佳个体保留，还未占领种群的优质个体仍然有可能在变异过程中消亡，因此可以记录每一代中适应度最高的个体及其适应度，最后“优中取优”，得到整个进化过程中的最优个体。

**4 曲线拟合插值方法**

4.1 分段线性插值

分段线性插值的思想非常简单。对数据集中的*n*个样本点，线性插值区间内的插值函数定义为式（4-1）：

（4-1）

代入待插值的点即可得到目标函数的估计值。

4.2 分段三次Hermite插值

本次研究的案例中，温度传感器的*T*-*V*特性曲线呈现明显的非线性，若将*T*表示为*V*的函数则该函数为典型的凸函数，该情形下分段线性插值并不是一种合适的拟合插值方法，在节中也将通过仿真试验及误差分析证明这一点。因此，本次研究的基准方法为三次样条插值(Spline)，后文也将研究另外两种三次多项式插值方法——Pchip和Makima插值。

4.2.1 Spline插值[6]

三次样条插值 (Spline Interpolation，简称Spline)，是一种分段三次多项式插值方法，该方法要求拟合插值得到的曲线通过所有给定的样本点，并且在这些样本点处曲线的一阶、二阶导数连续，满足两个边界条件。

假设数据集中有*n +* 1个样本点，满足：

样条曲线定义为：

其中可统一写作式（4-2）的形式，并满足式（4-3）和式（4-4）：

（4-2）

（4-3）

（4-4）

式（4-2）中共包含个未知数，式（4-3）提供了个方程，式（4-4）提供了个方程，还需要2个方程才能够求出所有的未知数，这就要求我们对边界条件做出规定。本文中应用的三次样条插值均采取非节点(Not-A-Knot)边界条件：

（4-5）

至此，方程与未知数的个数已经相等，可以求出每一段上三次多项式的系数，代入区间内待插值的点即可得到目标函数的估计值。

4.2.2 Pchip插值与Makima插值

Pchip和Makima插值函数的表达式具有与式（4-2）相同的形式，满足式（4-3）的约束，并且也都要求插值函数满足一阶导数连续性条件，它们主要的不同在于边界条件以及对于函数在某样本点处的导数估计方式，此处不再赘述。从效果上说，插值函数值变化的剧烈程度为：Spline > Makima > Pchip，因此Pchip和Makima比Spline能更好地防止拟合中“过冲”的现象，而Spline在拟合具有振荡趋势的函数时则具有比较明显的优势[7]。

**5仿真试验及结果比较**

5.1 标准遗传算法的试验结果

在未引入任何自适应改进的前提下，直接采用标准遗传算法求解式（2-5）的最优化问题，连续运行3次程序，得到的结果如表1所示：（标定方案一栏中，由序号1~90依次表示温度为-20℃~69℃的标定点）

**表1 采用标准遗传算法的程序运行结果**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 运行时长(s) | 标定方案 | 训练集成本 | 测试集成本 |
| 1 | 791.5 | 1, 10, 18, 40, 73, 87 | 305.3 | 304.6 |
| 2 | 803.7 | 3, 17, 24, 67, 88 | 267.0 | 272.7 |
| 3 | 782.6 | 3, 18, 26, 72, 90 | 271.3 | 271.3 |

标准遗传算法在3次运行中求出的最优标定方案为：总共设置5个标定点，温度分别为-18℃, -4℃, 3℃, 46℃, 67℃. 可以看到，尽管标准遗传算法在2次运行中均给出了质量尚可的方案（对比5.2节），但在接近全局最优时很容易陷入局部最优或收敛缓慢的困境，难以得到非常接近最优的标定方案。除此之外，在调参、测试的过程中可以发现标准遗传算法的结果仍然具有一定的不稳定性，多次运行得出的标定方案并不是非常接近，这会给实际的工程应用带来困难。

5.2 改进遗传算法的试验结果

在引入3.2节所述自适应改进的前提下，采用改进后的遗传算法求解式（2-5）的最优化问题，连续运行3次程序，得到的结果如表2所示：

**表2 采用改进遗传算法的程序运行结果**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 运行时长(s) | 标定方案 | 训练集成本 | 测试集成本 |
| 1 | 820.1 | 4, 19, 27, 72, 88 | 261.6 | 261.3 |
| 2 | 826.7 | 4, 20, 27, 72, 87 | 260.8 | 261.4 |
| 3 | 815.9 | 4, 20, 26, 72, 87 | 261.5 | 261.4 |

改进后的遗传算法在3次运行中求出的最优标定方案为：总共设置5个标定点，温度分别为-17℃, -2℃, 6℃, 51℃, 69℃. 对比表1，可以观察到：改进后的遗传算法在程序计算开销几乎没有增加的前提下，显著地改善了寻优的质量和稳定性。改进算法的程序3次运行得到的最优标定方案几乎完全一致，且成本相较于标准遗传算法求出的方案要更低一些，这大大提高了求出方案的实际应用价值，也证明了本文对遗传算法引入的改进在标定问题上是非常有效的。

**6 拓展讨论：不同拟合插值方法的影响**

不同的拟合插值方法选择的边界条件和拟合模型都有所差异，因此我在Spline插值以外又选取了三种比较常用的插值方法：Pchip，Makima和分段线性插值（Linear），并通过横向对比尝试得到不同插值方法对最优标定方案及成本以及插值误差的影响。

首先我将探讨不同插值方法求得结果的差异性。基于改进后的遗传算法，在其框架内分别应用4种插值方法进行求解，每种插值方法进行3次试验，得到的结果如表3所示：

**表3 采用不同拟合插值方法的程序运行结果**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 插值方法 | 运行时长(s) | 标定方案 | 训练集成本 | 测试集成本 |
| 1 | Spline | 820.1 | 4, 19, 27, 72, 88 | 261.6 | 261.3 |
| 2 | 826.7 | 4, 20, 27, 72, 87 | 260.8 | 261.4 |
| 3 | 815.9 | 4, 20, 26, 72, 87 | 261.5 | 261.4 |
| 4 | Pchip | 821.6 | 4, 15, 68, 77 | 219.8 | 219.8 |
| 5 | 807.7 | 4, 16, 70, 79 | 219.3 | 219.1 |
| 6 | 810.4 | 4, 16, 69, 78 | 219.4 | 219.4 |
| 7 | Makima | 805.9 | 3, 15, 37, 62, 89 | 255.5 | 255.4 |
| 8 | 799.2 | 3, 15, 37, 62, 89 | 255.5 | 255.4 |
| 9 | 810.7 | 3, 15, 39, 61, 88 | 255.8 | 255.8 |
| 10 | Linear | 813.3 | 1, 9, 19, 37, 57, 73, 90 | 393.4 | 393.3 |
| 11 | 821.1 | 1, 9, 20, 38, 58, 74, 90 | 393.2 | 393.2 |
| 12 | 820.6 | 1, 8, 19, 37, 51, 73, 90 | 395.0 | 395.0 |

由表3可以看出，采用不同的拟合插值方法，在程序的计算开销上几乎没有差异，但得到的最优标定方案及其成本却有着显著的不同。可以明显地看出，分段线性插值（Linear）的效果最糟糕，使用了7个标定点，来自误差的成本仍然是其他方案的2倍以上，考虑到被拟合函数的凸性，这一现象并不难理解。Spline和Makima的效果相近，后者稍占优势。它们选择的中部标定点有一定的不同，这可能与它们在估计导数和约束边界时的不同行为有关。比较出乎我意料的是Pchip在试验中表现出了很高的优越性，仅需4个点就可以达到足够的插值精度以避免误差超限的巨额成本。对此我给出的解释是：根据查阅的资料，Pchip的插值曲线对振荡的捕捉少，更倾向于平滑地连接样本点，而在本案例中*T*-*V*关系为单调凸函数，Pchip的拟合插值行为最为合适。

接下来我将从拟合插值误差的角度分析各种插值方法在本问题中应用的效果与合理性。根据四种插值方法下求出的最优标定方案进行标定，然后计算每个温度比较点的平均误差，得到的结果如图2所示：



（a）应用Spline插值 （b）应用Makima插值



（c）应用Pchip插值 （d）应用Linear插值

**图2 不同拟合插值方法下最优标定方案的误差分析对比**

可以看到，正如上文所提到的，*T*-*V*关系的凸性使得Linear插值的标定误差全部为负，而且这是该插值方法使用了7个标定点才达到的水准，因此可以认为它并不适合本次待测装置的标定。Spline、Makima和Pchip三种方式在误差的绝对大小上比较接近，但需要注意的是Pchip只使用了4个标定点，因此在本课程规定的成本计算方式下可以认为Pchip是最适合本案例的插值方式。在实际工程应用中，Pchip在左侧边界上的插值误差非常高，如果采用Pchip插值4点标定的策略，可以在左侧边界的标定上特别做一些优化，以期得到更好的标定准确性。或者，可以舍弃只追求成本最低的想法——采用Pchip插值5点标定，试验证明精度将显著提高，并且总体成本可以控制在251以内，仍优于其他插值方法。

从模型合理性上说，Linear方法从理论上已经可以被否定，而其他三种插值方式虽然都具备低成本将误差控制在±1℃范围内的能力，但可以看到其标定误差在很大范围内都超出了±2倍标准差的范围，因此可以推测三次多项式的模型和本次研究的待测装置特性并不完全符合。如果能从物理上得到装置特性的准确模型，对标定的准确性将会是大有裨益的。

**7 结论**

本文以一种具有显著非线性和个体差异性的温度测量装置为例，研究了大规模制造中测量装置的标定问题。基于量化的误差成本和标定成本，本文将选择标定点形式化为数学上的组合优化问题，并通过引入了自适应改进的遗传算法进行求解。试验结果表明，改进后的遗传算法相较于标准遗传算法具有更强的寻优能力和稳定性，求出采用Spline插值的最优标定方案为：设置5个标定点，温度分别为-17℃, -2℃, 6℃, 51℃, 69℃.

同时，本文中我还研究了不同拟合插值方法对最优标定方案的影响，并通过误差分析探讨了拟合插值所建立模型的合理性，结果表明通过Pchip插值方法建立的模型最符合待标定装置的特性，该插值方法下的最优标定方案仅需设置4个标定点。在将来的工作中，我希望能够系统研究遗传算法中超参数选取对寻优的影响，并应用其他的启发式搜索算法（如模拟退火算法、粒子群算法等）进行分析和试验。

**8 致谢**

在本次标定问题的研究中，非常荣幸地得到了老师和同学们的帮助，我对此表示衷心的感谢：袁焱和李安琪两位老师在授课过程中就问题的研究思路、程序的设计思路以及论文的写作技巧提供了大量富有指导性的观点和建议，李安琪老师在工作进度汇报中提出了关于对比实验的建议，都为我整个研究的构建和报告的写作提供了莫大的帮助；2018级王灿浩同学在工作进度汇报中提出了十进制编码的思路，这是本次研究改进变异过程思路的主要来源，这一改进显著提高了算法的性能。

**9 参考文献**

[1] 上海交通大学电子工程系. 工程问题建模与仿真课程讲义[EB/OL]. ftp://202.120.39.248.

[2] J. Holland. Adaptation in natural and artificial systems: an introductory analysis with application to biology[J]. Control and artificial intelligence, 1975.

[3] K. De Jong. Analysis of the behavior of a class of genetic adaptive systems[D]. 1975.

[4] M. Sriniva, L. M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms[J]. IEEE Transactions on Systems, Man, and Cybernetics,1994,24(4):656-667.

[5] 周明, 孙树栋. 遗传算法原理及应用[M]. 国防工业出版社, 1999.

[6] Wikipedia. Spline interpolation[Z/OL]. https://en.wikipedia.org/wiki/Spline\_interpolation.

[7] MathWorks. Cubic spline data interpolation (spline)[DB/OL]. https://ww2.mathworks.cn/help/matlab/ref/spline.html.

**附 录**

代码清单(仅包含本文改进后的遗传算法)

main.m：遗传算法主程序，控制整个算法流程

clear;

% Load the provided Datasets

train\_set = load('dataform\_train.csv');

test\_set = load('dataform\_testA.csv');

% Define max rounds of iteration and the number of individuals

max\_iter = 500;

num\_features = 90;

num\_ind = 100;

mean\_cost = zeros(num\_ind, max\_iter + 1);

tic;

init\_groups; % Implement group initiation

for k = 2 : max\_iter + 1

% Resort the group

[f, I] = sort(f, 'descend');

groups = groups(I, :);

p = cumsum(f) / sum(f);

groups = select(p, groups);

% Implement cross between pairs

for m = 1 : num\_ind / 2

[groups(2 \* m - 1, :), groups(2 \* m, :)] = match(groups(2 \* m - 1, :), groups(2 \* m, :), max(f(2 \* m - 1), f(2 \* m)), f);

end

% Implement single mutation

groups = mutate(groups);

% Implement point transverse shifting

groups = mutate\_pro(groups);

% Compute the fitness and cost

for i = 1 : num\_ind

[mean\_cost(i, k), f(i)] = fitness(groups(i, :), train\_set);

end

% Record the best individual in current generation

[best(k, 1), idx] = max(f);

best\_ind(k, :) = groups(idx, :);

% Dynamically adjust the fitness of individuals

f = (f - min(f)) / (max(f) - min(f)) + c;

c = ceta \* c;

end

% Find the historically best individual and get the best solution

[~, idx] = max(best);

sample\_points = find(best\_ind(idx, :));

fprintf('The best sampling points are:');

for k = 1 : length(sample\_points)

fprintf(' %d', sample\_points(k));

end

% Preparation for data analysis

main\_cost = mode(mean\_cost, 1);

least\_cost = min(mean\_cost, [ ], 1);

T\_error = compute\_error(best\_ind(idx, :), train\_set);

mean\_error = mean(T\_error, 1);

var\_error = var(T\_error, [ ], 1);

% Print some related information

fprintf('\nThe best solution is found in Generation %d.', idx);

fprintf('\nThe execution time is: %.1f s.\n', toc);

fprintf('The least average cost on training set is: %.2f\n', min(least\_cost));

% Examine the best solution on the test set

temp = false(1, num\_features);

temp(1, sample\_points) = 1;

[real\_cost, ~] = fitness(temp, test\_set);

fprintf('The average cost on testing set is: %.2f\n', real\_cost);

% Plot the evolving picture

hold on;

plot(1 : max\_iter + 1, least\_cost, 'LineWidth', 2);

plot(1 : max\_iter + 1, main\_cost, 'LineWidth', 2);

legend('Least Cost', 'Mode Cost', 'Location', 'northeast');

xlim([1, max\_iter + 1]);

set(gca, 'Fontsize', 16);

xlabel('Round of Evolving', 'Fontsize', 20);

ylabel('Average Cost', 'Fontsize', 20);

title('Trend of Least Average Cost During Group Evolving', 'Fontsize', 28);

grid on; box on;

hold off;

% Conduct error analysis

error\_analysis;

init\_groups.m：种群初始化

% Matrices for best records

best = zeros(max\_iter + 1, 1);

best\_ind = false(max\_iter + 1, num\_features);

% Form a stochastic population

groups = rand(num\_ind, num\_features) >= 0.5;

% Initiate the dynamic adjust parameters of fitness

c = 0.1;

ceta = 0.995;

f = zeros(num\_ind, 1);

% Fix those individuals with vital errors

for i = 1 : num\_ind

while sum(groups(i, :)) < 4

groups(i, :) = rand(1, num\_features) >= 0.5;

end

[mean\_cost(i, 1), f(i)] = fitness(groups(i, :), train\_set);

end

% Record the best individual and compute individual fitness

[best(1, 1), idx] = max(f);

best\_ind(1, :) = groups(idx, :);

f = (f - min(f)) / (max(f) - min(f)) + c;

c = ceta \* c;

fitness.m：计算个体适应度

function [mean\_cost, f] = fitness(ind, dataset)

% Conduct calibration with the solution related to the individual

% Use the reciprocal of squared total cost as the fitness

num\_points = sum(ind); % The number of sampling points

num\_samples = size(dataset, 1) / 2; % The number of samples

cost = zeros(num\_samples, 1);

T\_inter = zeros(num\_samples, size(dataset, 2));

% Real statistics

T\_all = dataset(1, :);

V\_all = dataset(2 : 2 : num\_samples \* 2, :);

% Take out the sampling points of the individual

V\_taken = V\_all(:, ind);

T\_taken = T\_all(:, ind);

for i = 1 : size(V\_all, 1)

% Implement interpolation, the last parameter can be replaced by others to test different methods

T\_inter(i, :) = interp1(V\_taken(i, :), T\_taken, V\_all(i, :), 'spline');

% Compute the cost for a single sample

cost(i) = total\_cost(T\_inter(i, :), T\_all, num\_points);

end

% Compute the average cost and fitness of the individual

mean\_cost = mean(cost);

f = 1 / mean\_cost^2;

end

select.m：改进后的选择过程，基于轮盘赌

function next\_gen = select(p, groups)

% Select individuals of high fitness

% Improved by adding protection for the best individual

[num\_ind, num\_features] = size(groups);

temp = rand(num\_ind, 1); % Stochastic number for roulette

next\_gen = false(num\_ind, num\_features);

% Preserve the best individual of last generation and copy it

next\_gen(1, :) = groups(1, :);

next\_gen(2, :) = groups(1, :);

% Select individuals according to the rand matrix

for k = 3 : num\_ind

for m = 1 : num\_ind

if temp(k) <= p(m)

next\_gen(k, :) = groups(m, :);

break;

end

end

end

end

match.m：改进后的个体交叉

function [new\_1, new\_2] = match(ind\_1, ind\_2, fm, f)

% Cross the chromosomes of different individuals

% Improved by self-adaptive probability of cross according to individual fitness

f\_avg = mean(f);

f\_max = max(f);

% Self-adaptive probability of cross

if fm < f\_avg

p\_match = 1;

else

p\_match = 0.6 \* (f\_max - fm) / (f\_max - f\_avg) + 0.1;

end

% If cross happens, swap a fragment stochastically

% Else just keep the two individuals unchanged

if rand() <= p\_match

temp = rand(2, 1);

num\_features = size(ind\_1, 2);

front = ceil(min(temp) \* num\_features);

back = ceil(max(temp) \* num\_features);

new\_1 = ind\_1;

new\_2 = ind\_2;

new\_1(1, front : back) = ind\_2(1, front : back);

new\_2(1, front : back) = ind\_1(1, front : back);

else

new\_1 = ind\_1;

new\_2 = ind\_2;

end

end

mutate.m：个体基本位变异

function new\_groups = mutate(groups)

% Stochastic mutation on single points

p\_mut = 0.005; % The probability of mutation

new\_groups = groups;

% Determine if the mutation happens and implement mutation

if\_mut = rand(size(groups)) <= p\_mut;

new\_groups(if\_mut) = 1 - new\_groups(if\_mut);

% 3-order polynomial interpolation needs at least 4 points

% Re-mutate those individuals with vital errors

for m = 1 : size(groups, 1)

while sum(new\_groups(m, :)) < 4

new\_groups(m, :) = groups(m, :);

temp = rand(1, size(groups, 2)) <= p\_mut;

new\_groups(m, temp) = 1 - new\_groups(m, temp);

end

end

end

mutate\_pro.m：标定点横移变异

function new\_groups = mutate\_pro(groups)

% Stochastically shift some calibration points

p\_mut = 0.05; % The probability of shifting

new\_groups = groups;

% Determine if shifting happens on an individual

if\_mut = rand(size(groups, 1), 1) <= p\_mut;

for m = 1 : size(groups, 1)

if if\_mut(m)

idx = find(groups(m, :));

% Get the shifting point

[~, mut\_pos] = max(rand(size(idx)));

temp = idx;

temp(mut\_pos) = idx(mut\_pos) + ceil(6 \* rand() - 3.5);

% If the shift is illegal, try again

while temp(mut\_pos) < 1 || temp(mut\_pos) > 90

temp(mut\_pos) = idx(mut\_pos) + ceil(6\*rand() - 3.5);

end

idx = temp;

new\_groups(m, :) = false;

new\_groups(m, idx) = true;

end

end

end

compute\_error.m：计算方案的标定误差，用于最优个体的分析

function error = compute\_error(ind, dataset)

num\_samples = size(dataset, 1) / 2;

T\_inter = zeros(num\_samples, size(dataset, 2));

T\_all = dataset(1, :);

V\_all = dataset(2 : 2 : num\_samples \* 2, :);

V\_taken = V\_all(:, ind);

T\_taken = T\_all(:, ind);

for i = 1 : size(V\_all, 1)

T\_inter(i, :) = interp1(V\_taken(i, :), T\_taken, V\_all(i, :), 'spline');

end

error = T\_inter - T\_all;

end

error\_analysis.m：误差分析，绘制误差针状图及标准差包络线

% Plot the stem figure of calibration error

figure();

stem(1: 90, mean\_error, 'o');

xlim([0 91]); ylim([-1.5 1.5]);

% Plot the 2 times standard error range

hold on; plot(1: 90, 2 \* sqrt(var\_error), '--');

plot(1: 90, 2 \* sqrt(var\_error), '--', 'linewidth', 1.5);

plot(1: 90, -2 \* sqrt(var\_error), '--', 'linewidth', 1.5);

legend('Average error', '2 times of standard error', '-2 times of standard error');

set(gca, 'fontsize', 16);

xlabel('Comparison temperature points','fontsize', 20); ylabel('Calibration Error','fontsize', 20);

title('Error Analysis of spline interpolation','fontsize', 28);

grid on; box on;

total\_cost.m：计算一个方案在一个样本上的成本

function cost = total\_cost(T\_inter, T\_real, num\_points)

cost = 0;

for i = 1 : size(T\_real, 2)

cost = cost + single\_cost(T\_inter(i), T\_real(i));

end

cost = cost + 50 \* num\_points;

end

single\_cost.m：计算单点误差成本

function cost = single\_cost(T1, T2)

if abs(T1 - T2) <= 0.5

cost = 0;

elseif abs(T1 - T2) <= 1

cost = 1;

elseif abs(T1 - T2) <= 1.5

cost = 5;

elseif abs(T1 - T2) <= 2

cost = 10;

else

cost = 10000;

end

end