统计推断在数模转换系统中的应用

——寻求对传感器部件的传感特性校准方法（定标工序）

组号：53　姓名：何晗晓 学号：5130309689，姓名：何俊贤 学号：5130309699

摘要：本文根据提供的关于传感器部件的 469 组样本数据，尝试一些拟合方法，对每一种给定的拟合方法，同时通过遗传算法和模拟退火算法选择出能够使定标成本最低的定标工序。我们也对遗传算法的最优交叉互换方式、遗传算法最优变异方式、模拟退火算法最优的解扰动方式进行了探索和寻找。使用两种算法同时解决课题，一方面可以达到互相验证结果的目的，另一方面我们也从各个角度对两种算法进行了比较，以期得到此课题的最好解决方法。最后我们根据已经得到的结果，加以限制利用枚举法找到了严格的全局最优解。并对相应的误差进行了分析。

关键词：遗传算法，模拟退火算法，比较，枚举，严格最优解，误差分析

1.引言

传感器定标很多地方又名为辐射定标，严格意义上讲，辐射定标是传感器定标的一部分内容。定标是将遥感器所得的测量值变换为绝对亮度或变换为与地表反射率、表面温度等物理量有关的相对值的处理过程。或者说，遥感器定标就是建立遥感器每个探测器输出值与该探测器对应的实际地物辐射亮度之间的定量关系；建立遥感传感器的数字量化输出值 DN 与其所对应视场中辐射亮度值之间的定量关。

我们总结以上的定义，通俗的说法：传感器定标就是将图像的数字量化值（DN）转化为辐射亮度值或者反射率或者表面温度等物理量的处理过程。其中反射率又分为大气外层表观反射率和地表实际反射率，后者又属于大气校正的范畴，有的时候也会将大气校正纳入传感器定标的一种途径。

2.基于遗传算法和模拟退火算法的传感特性校准方案求解

2.1 求解路径

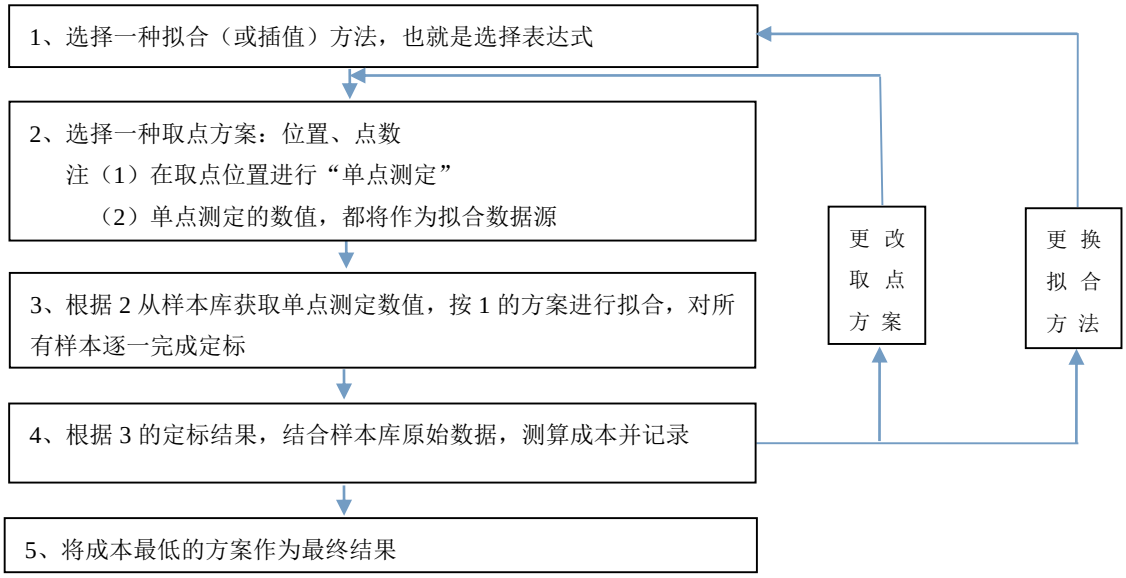

图 2-1 求解流程图

其中我们根据已知的传感器部件特性选择适当的不同拟合方法，因为选点方案的解集总共有 2^{51} 种可能，穷举法肯定不合理。所以更改选点方案依靠启发式搜索算法。

2.2 遗传算法

在遗传算法中，将解集看作是自然界中的个体，依据自然界中的自然选择、优胜劣汰的原理，经过数代的进化，以期尽量得到最佳的选点方案。

这里的“个体”实际上就是一种取点方案。我们用一个长度 51 的二进制向量来表示取点方案，将选点的对应位置置为 1，否则置为 0。一开始算法会指定进化的代数 GenerationNum，以及每一代种群的大小 popsize，算法会生成每一代所有个体的平均成本矩阵 AveCost 以及每一代最优个体的最低成本矩阵 CostBest，在程序的最后会绘制出 AveCost，CostBest 与进化代数之间的关系图像，通过观测两条曲线最后是否趋于平稳来确定是否已经基本达到最优解。

此处说“基本达到最优解”，是因为此算法不可能遍及每一种可能，所以不能保证所找到的解一定是全局最优解，但通过遗传算法可以保证的是，找到的解与最优解之间的差距绝对足够小，通过程序画出的图像以及得到的数据也可以看出，当进化代数足够大时，平均成本以及最优成本都基本不再变化，代与代之间的差距已经很小，这已经可以说明我们的解与最优解之间的差距足够小，于是可以近似认为我们的方案就是当前拟合方式下的最优解。

下面是我们使用的遗传算法的详细流程图：

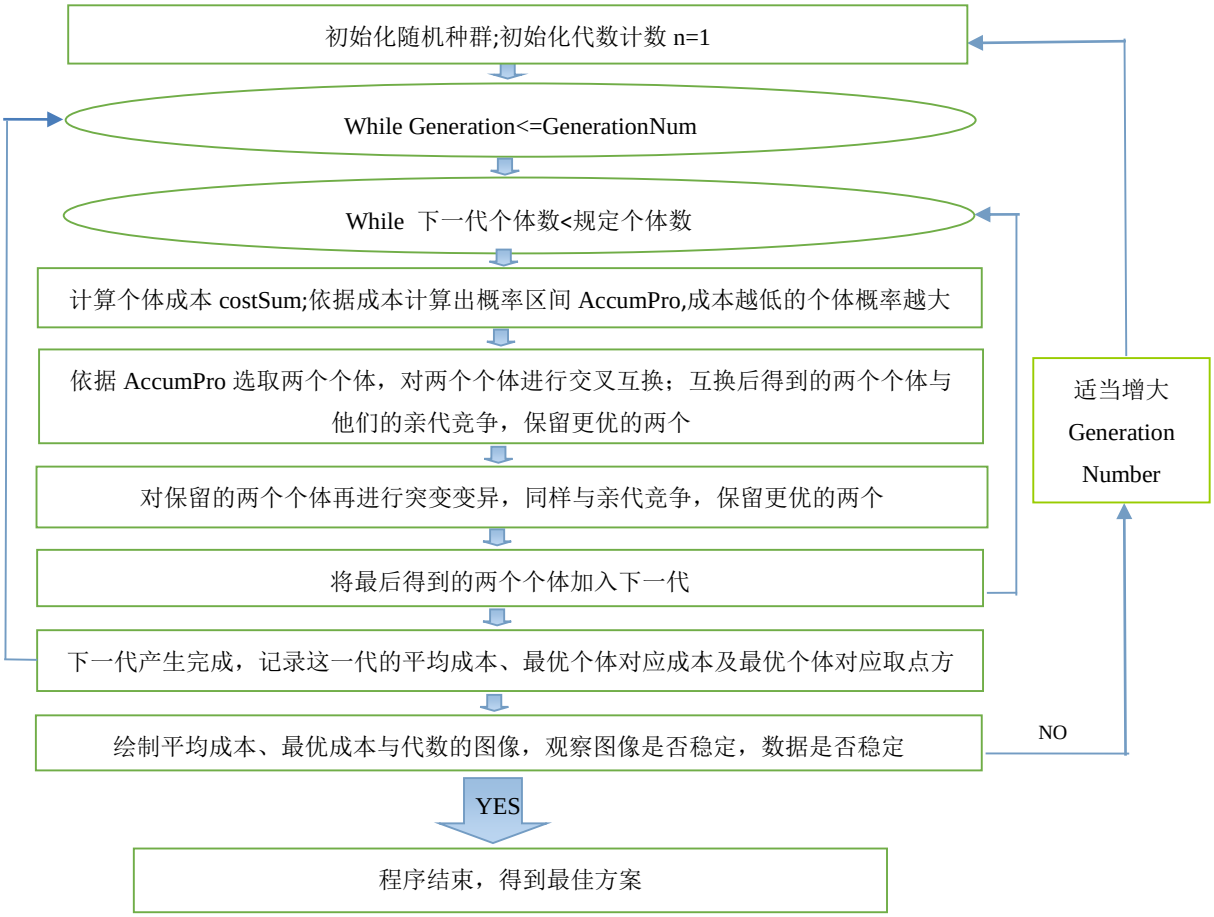

图 2-2 遗传算法流程图

其中对于选取两个体的过程，我们的程序中实现了上一代最优的两个个体，在下一代开始产生之初一定会被选出来交叉互换和变异。这样做保证了良好的基因一定会被继承下来，使每一代的最优成本一定是单调递减的，且这样更好的继承好基因会使整个种群进化更快。

遗传算法的交叉变异、基因突变以及适应度计算的方式都非常重要，它们的优劣会极大的影响种群的进化速度以及进化程度，我们在做的过程中对这三种方式也进行了多次尝试和对比，下文会详细讨论。

算法中产生最终结果使用的交叉变异和基因突变方式均为我们经过分析对比后认为的最好的方式。

适应度的计算采用倒数的平方。这样可使优的个体优势体现得更加明显。

2.3 模拟退火算法

模拟退火算法来源于固体退火原理，将固体加温至充分高，再让其徐徐冷却，加温时，固体内部粒子随温升变为无序状，内能增大，而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。根据 Metropolis 准则，粒子在温度 T 时趋于平衡的概率为 e(-ΔE/(kT))，其中 E 为温度 T 时的内能，ΔE 为其改变量，k 为 Boltzmann 常数。由初始解 i 和控制参数初值 t 开始，对当前解重复“产生新解→计算目标函数差→接受或舍弃”的迭代，并逐步衰减 t 值。

“固体的内部粒子状态”就是此课题中的解，我们依旧用长度为 51 的二进制向量来表示取点方案，取点的对应位置为 1，否则为 0。在初始时随机生成一个状态，选择足够高的初始温度和足够低的末态温度，设置好每个温度下的迭代次数。我们用 recordValue 来记录每个温度下产生的最优解，最后可以通过它获取整个过程中出现的最优解。程序结束后绘制出每个温度下的最优解随序数的变化图像，当图像平稳时认为“内能”已经最小。已经基本达到最优。下面是程序流程图：

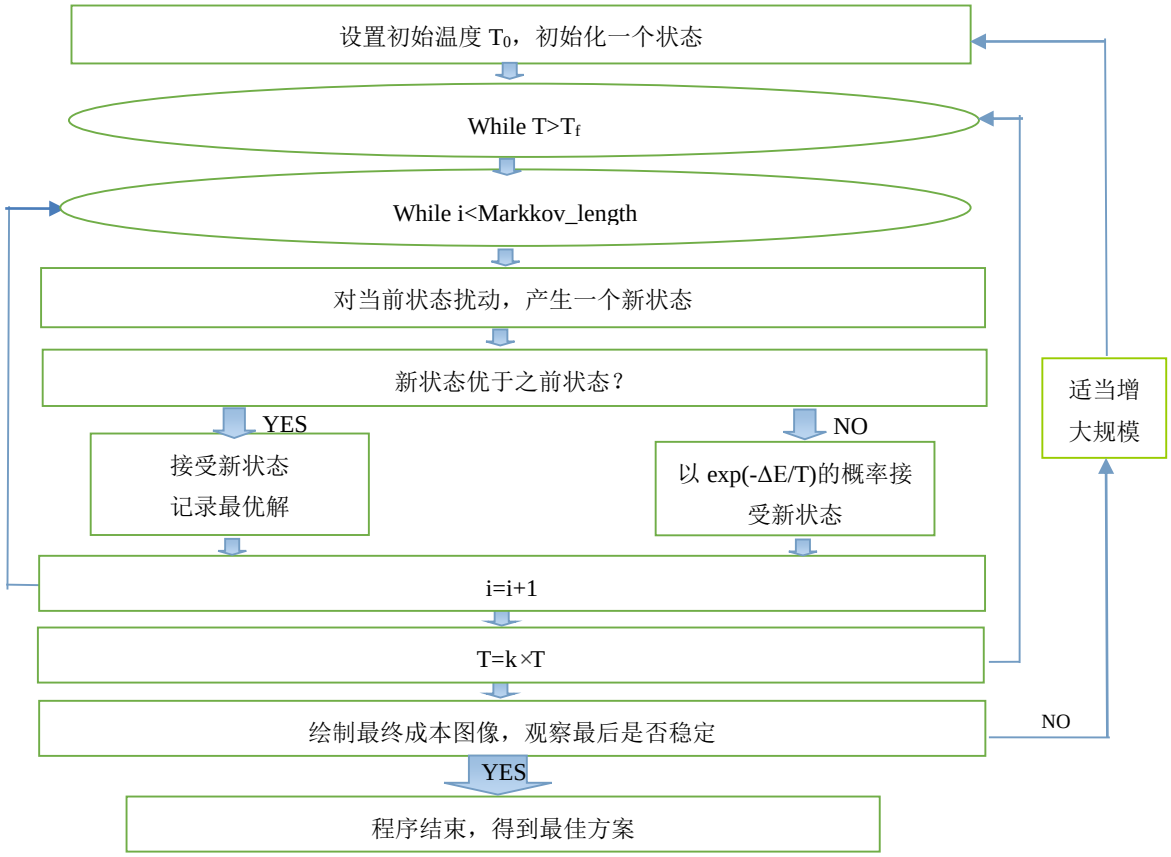

图 2-3 模拟退火算法流程图

和遗传算法一样，退火算法解的波动方式也相当重要，下文同样也会对不同的波动方式进行详细讨论。

退火算法产生最终结果使用的波动方式为我们经过分析讨论后认为的最好的波动方式。

2.4 结论

2.4.1 遗传算法

2.4.1.1 结果的数据说明

我们选取进化代数为 50 代，每代种群由 500 个个体组成，首先使用三次多项式拟合方法，我们规定选的数不得低于 5 个（我们有理由限制最小取点数，因为取点数过少必然是不合理的）。程序在运行了 25000s 后得到结果，最后得到的最低成本 BestValue 为 111.7004，最佳选点方案为选取 4,15,27,38,49 这 5 个点。

然后我们将方法改为三次样条插值，同样规定选点数不低于 5 个，最后得到的最低成本 BestValue 为 92.8774，最佳选点方案 BestMethod 为选取 3,12,22,31,43,50 这 6 个点。

最后我们将方法改为三次多项式插值，规定选点数不低于 5 个，得到的结果是最低成本 BestValue 为 82.7655，最佳选点方案 BestMethod 为选取 4,16,26,35,48 这 5 个点。

因此，在经过多次尝试后，基于我们目前所做的工作，我们认为最佳的拟合方式为三次多项式插值，最低成本为 82.7655，最佳选点方案为选取 4,16,26,35,48 这 5 个点。

从进化的监测数据上看，虽然平均成本还呈减小趋势，种群还在不断进化，但进化幅度已经相当微小，而且每一代的最小成本在最后几代中也不再减少，学生以为可以认为种群已经达到最优，我们得到的解即为此种拟合方式下的最优解。

2.4.1.2 结果的图像说明

以下是每一代的最低成本 CostBest，平均成本 AveCost 与进化代数之间的关系图像：

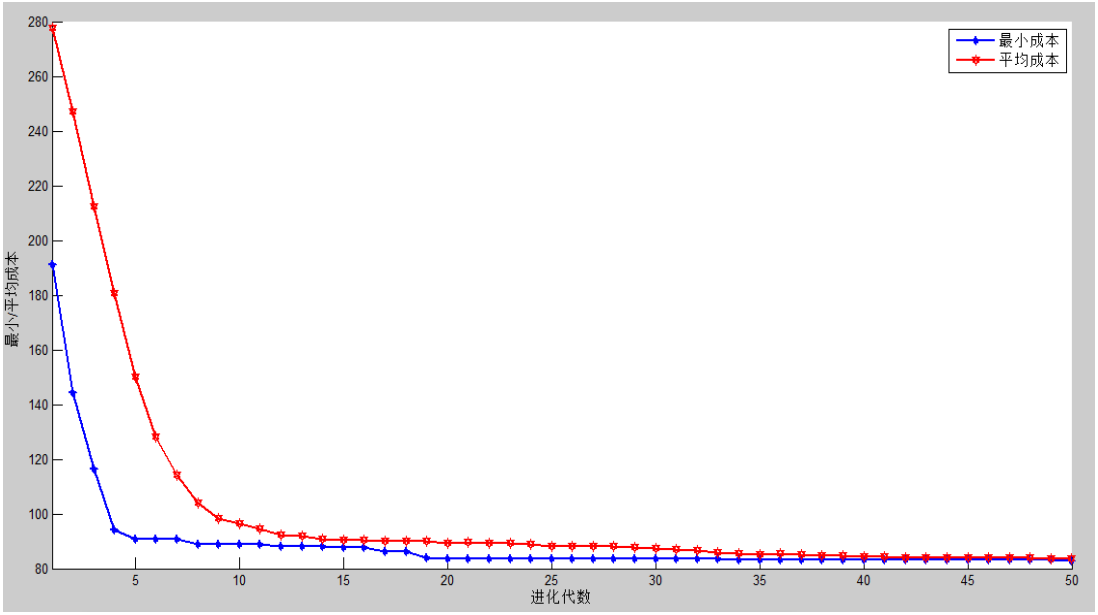

图 2-4 遗传算法运行结果图

从图中可以看出，在进化的最后，最小成本与平均成本均趋于稳定，平均成本也越来越趋于最小成本。这也从图像的角度说明了我们最后得到的种群基本已经最优，可以认为我们得到的解为最优解。

2.4.2 模拟退火算法

2.4.2.1 结果的数据说明

我们选取初始温度为 250，末态温度为 0.01，Boltzmann 常数为 0.9，每一个温度下的迭代次数为 100 次。

因为遗传算法已经得到最佳拟合（插值）方式为三次多项式插值，所以这里我们不必再尝试多种拟合方式，而直接使用三次多项式插值。

在程序运行之后，我们得到最低成本为 82.7655，最佳取点方案为选取 4,16,26,35,48 这 5 个点。

得到的结果与遗传算法得到结果完全一致，两者相互验证。说明结果是十分可靠的。

2.4.2.2 结果的图像说明

下面是每个温度下的最优解随温度改变次数的变化图像：

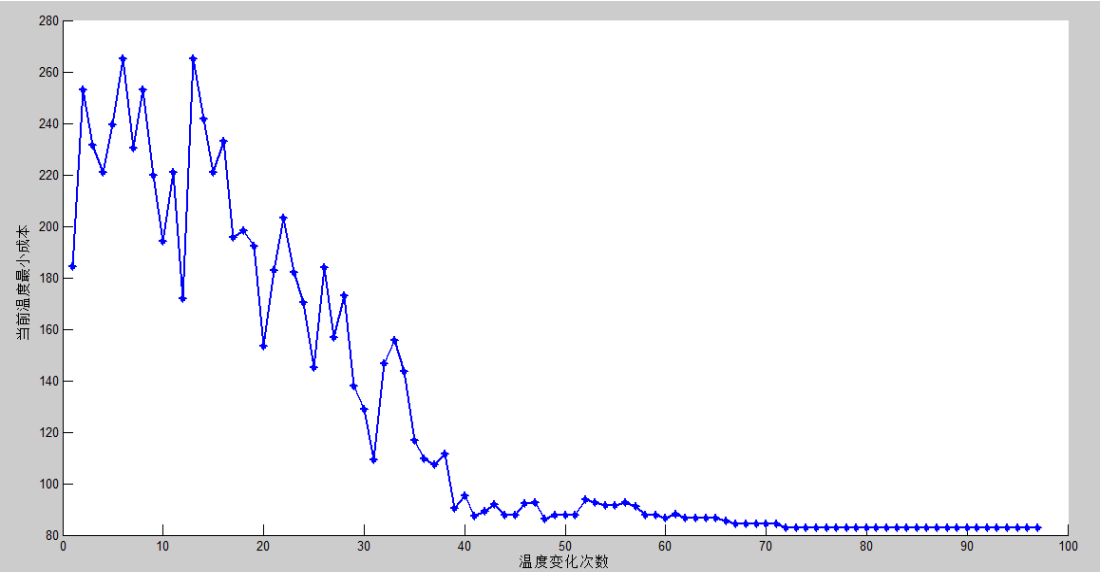

图 2-5 模拟退火算法运行结果图

由图像看出，在温度较高时波动很大，温度较低时基本不再波动，趋于平稳。这是由算法本身决定的。因为温度高时接受差解的概率很大，所以看到的图像是波动的。温度低时接受差解的概率已经趋于 0。这个波动虽然占用了很多时间，但它又是十分必要的。因为这使局部最优解能够概率性的跳出并趋向全局最优，有效的避免了算法陷入全局最优解。

3.严格全局最优解的产生——枚举法

此题要遍及解集内的所有解在短时间内当然是不可能的，不过其实容易想到，在枚举的过程中，大量的情况都是可以排除的。于是我们希望基于之前的遗传算法和退火算法得到的方案，在短时间内得到此课题在三次多项式插值方法下的严格全局最优解。

注意到遗传算法和退火算法得到的解方案为取点 4，16，26，35，48 。这五个点分别分布在区间 1-10,11-20，21-30,31-40,41-51 内，于是我们有理由相信，全局最优解的取点数一定为 5 个，并且这 5 个点一定分别分布在这 5 个区间内。所以我们利用 5 层 for 循环，分别在这 5 个区间内遍历，这样很大程度上减少了我们需要遍历的规模，在短时间内得到了结果。

巧合的是，我们最后得到的结果刚好是 82.7655，最佳方案是取点 4，16，26，35，48。这个结果与遗传算法和退火算法得到的结果完全一致！这说明前面两种算法都在短时间内非

常成功的找到了严格全局最优解。

三种方法，得到的结果均完全相同，彼此验证。进一步说明结果的正确性。

4.最优解下的误差分析

最优解已经使成本最低，我们为了进一步验证这种插值方式和选点方案与实际数据的符合程度。对 469 组样本的平均误差和最大误差进行了统计，统计结果如下图所示：

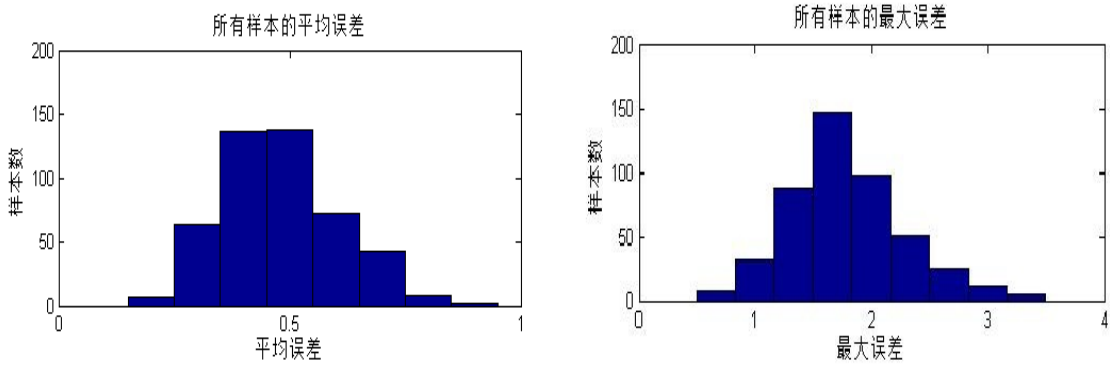

图 4-1 最优解误差直方图

从图中可以看出，样本的平均误差主要集中在 0.4、0.5 附近；最大误差主要集中在 1.5 附近。这说明这样的插值方式和选点方案与实际数据之间误差较小，符合程度较好。

5.关于遗传算法交换、突变的进一步探究

5.1 交叉互换方式的探究

遗传算法交叉互换方式的不同会很大程度上影响到其进化速度以及进化能力。不光是影响程序的效率，较坏的交换方式甚至会对最后得到的最低成本数值有直接的影响。所以我们通过一定的标准来比较不同的交叉互换方式，以期得到一种较好的交叉互换方式。下面是对参与比较的三种交换方式的说明：

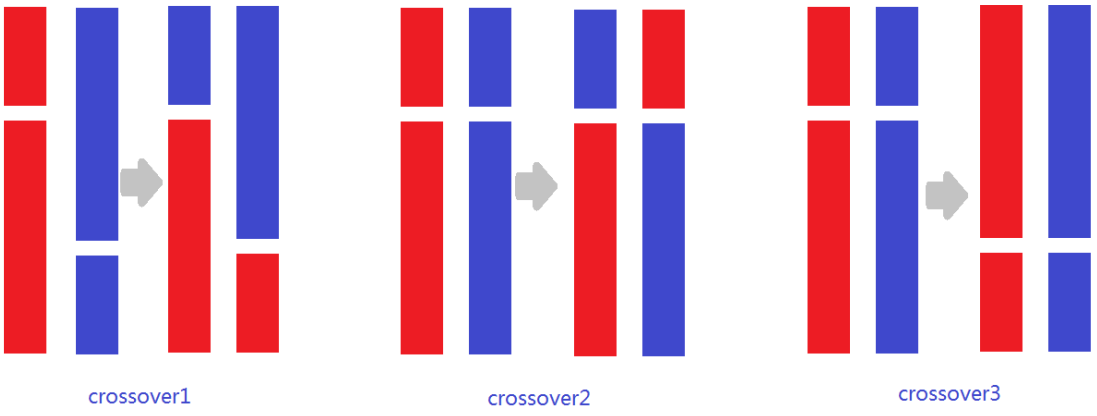

图 5-1 三种不同 crossover 示意图

三种交换方式均是随机选择一个交叉位点。crossover1 是第一个个体的“头”与第二个个体的“尾”交换；crossover2 是第一个个体的“头”与第二个个体的“头”交换；crossover3 是第两个个体自己的“头”与自己的“尾”交换。

对于比较的“标准”，我们进行了多次讨论和多次尝试。最初我们是使用当前时间得到的

最优解随时间的变化关系作为标准。后来发现随机性很大，它并不能反映一种方式对种群的进化能力，因为就算整个种群都较差，但是由于随机性，种群中可能有一个个体非常的好，但这不能说明这种交换方式就是好的。因为最优个体的产生有很大的随机性，每次运行都会有不一样的情况。

基于这一点，我们决定选取一代种群中所有个体的平均成本随时间的变化关系作为标准来比较。这样是将一个种群作为一个整体来考虑，在种群规模较大的情况下，可以很好的反应随机背后的统计规律。这样的平均成本可以较好的反应在某种交换方式下种群的进化能力。

当然，这样得到的进化能力最强的交换方式并不一定在每次运行中都会较其他方式更快的到达最优解，也不一定得到的最优解更优。但我们有理由说，这样得到的最好的交换方式较其他交换方式有更大的概率更快达到最优解，也有更大的概率产生更优的最优解。

比较需要控制无关变量，首先是种群规模和进化代数一致；然后是基因突变的方式要一致，这里我们以最简单的基因突变方式——随机选择一个突变位 0 变为 1 或者 1 变为 0；适应度的计算也要一致，我们这里选取 $1/(costSum)^2$ 作为适应度(costSum)为个体的成本；还有一点容易忽略的是，要控制初始种群是相同的，因为只有在初始种群完全一致的前提下，不同时刻的种群优劣才能反应进化能力的大小。

综合以上进行统计，我们得到了在三种交换方式下种群的平均成本随时间的变化图像：

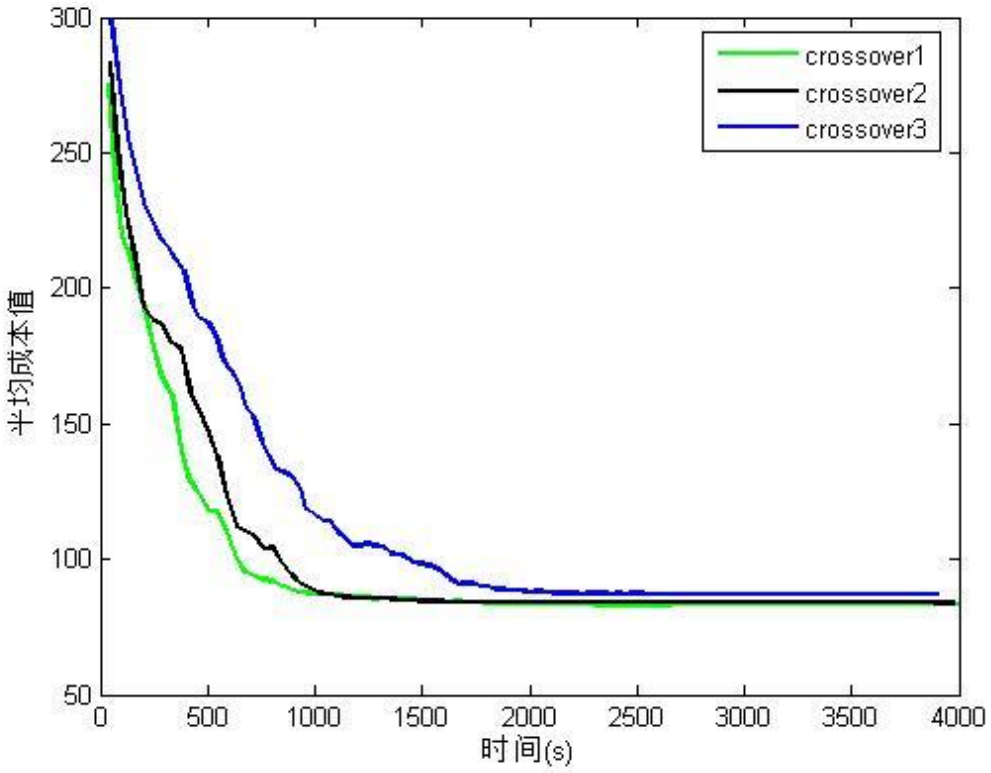

图 5-2 三种不同 crossover 运行结果图

分析上面图像我们可以得到：

(1) 在三者都不稳定的区间内，同一时刻下的种群平均成本值是 crossover1<crossover2<crossover3，且差异较明显；

(2) crossover1 和 crossover2 代表的平均成本呢几乎同时达到基本稳定，差异不大，而 crossover3 达到稳定所需的时间明显更长；

(3) 最后的平均成本稳定值，crossover1 为 82.966，crossover2 为 83.856，crossover3 为 86.951，crossover1<crossover2<crossover3;

综合以上三点，我们知道 crossover1 使种群进化得更快，且最后稳定时进化程度也更高，所以我们认为在这三种方式中，crossover1 是最好的交叉互换方式。

说明：crossover1 似乎违背了生物交叉互换的原理，按理说应该是 crossover2 更贴近自然界的实际情况，应该是 crossover2 更好才对。但图像以及数据证明确实是 crossover1 更好。我们认为这与本课题的实际情况有关。crossover1 看似没有任何道理，看似是一种没有理由的变异方式，其实不然。因为此题目的数据分布曲线本来就具有对称性，也就是说我们的取点的序号关于 25 一定是基本对称的，所以这说明在最优解附近的个体不光相同段“基因”相似度很高，头部与尾部“基因”相似度也很高，所以我们完全有理由选择两个个体的“头”与“尾”互换或者“头”与“头”互换。实际证明“头”与“尾”互换更好，即 crossover1 更好。

得到这样的结论后，我们最后产生最终结果的遗传算法选取的交叉互换方式就是 crossover1。

5.2 基因突变方式的探究

和交叉互换一样，基因突变的方式对种群的进化速度和进化程度也有很大的影响。这里我们选取了三种基因突变的方式进行比较。三种方式的说明如下：

(1) mutation1:以 0.7 的概率随机选择一个位置变异;0.3 的概率随机选择两个位置变异；

(2) mutation2:以 0.5 的概率随机选择一个位置变异;0.5 的概率随机“丢弃”一个位置，即舍弃一个位置的基因，后段所有基因整体前移一位，第 51 位用 0 补齐；

(3) mutation3:随机选择一个位置变异。

比较标准和交叉互换一样，选择种群的平均成本随时间的变化曲线作为标准。无关变量的控制也和交叉互换基本一致，只是这里交叉互换方式是作为无关变量，我们选择的交叉互换方式是上面得到的 crossover2。

综合以上，我们得到了三种基因突变方式下种群的平均成本呢随时间的变化曲线：

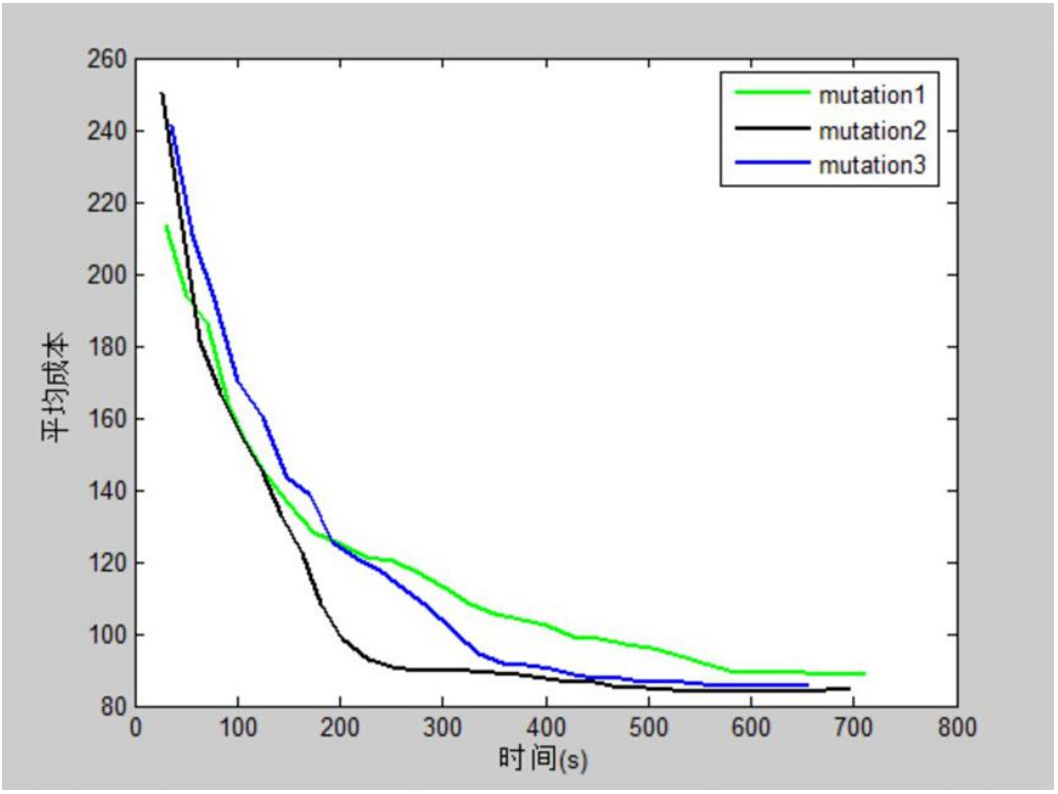

图 5-3 三种不同 mutation 运行结果图

由图像我们得到：

(1) 在进化之初，也就是 150s 以前，三条曲线相接近，差别不大；

(2) 150s 左右开始，三条曲线开始出现明显差别，在之后三条曲线都还没有稳定的区间内，同一时刻下三条曲线代表的平均成本的大小关系为：mutation2<mutation3<mutation1；

(3) 比较达到稳定所需的时间，mutation2<mutation3<mutation1；

(4) 比较最后的稳定值，mutation2<mutation3<mutation1；

从以上四点可以看出，种群在 mutation2 的方式下，进化的更快，稳定时进化程度也更高，所以我们认为 mutation2 是这三种方式中最好的一种突变方式。得到这样的结论后，我们最后产生最终结果的遗传算法选取的基因突变方式就是 mutation2。

说明：和 crossover1 一样，这里最好的 mutation2 看起来也是有点“无厘头”的，其实不然。因为整体前移，相当于对尾部的每个取点在原本的位置上左移了一位。在取点不多的情况下，这样的突变方式其实相当于只对原来的方案造成了微小的改变，可以很好地实现在局部内的搜索。

6.关于模拟退火算法解扰动方式的进一步探究

对于退火算法解扰动的方式的重要性就像变异方式对于遗传算法的重要性一样，可以很大程度上的去影响程序的效率以及最后解的优劣。这里我们选取了三种扰动方式进行比较，三种方式说明如下：

(1) flux1：以 0.5 的概率随机选择一位变化，0.45 的概率随机选择两位变化，0.05 的概率随机选择三位变化；

(2) flux2：随机选择一位变化；

(3) flux3：0.5 的概率随机选择一位变化，0.5 的概率随机选择一位舍弃。即舍弃一个位置的基因，后段所有基因整体前移一位，第 51 位用 0 补齐；

对于比较我们同样控制了变量，即控制退火算法的所有参数相同。也控制了它们在程开始时随机产生的第一个状态相同。

对于比较的标准，我们选用每个温度下的最后一个成本随温度变化次数的图像。这样做是因为最好成本具有很大随机性。我们需要的是能够反应整体受到影响的一个量。退火算法每个温度下迭代 100 次，每次扰动以一定的概率接受更差的解跳出局部，实际上 100 次后最后一个值是有统计上的意义的，从退火算法的原理来说，它代表了这个温度下固体内部粒子稳定的状态，所以它是可以反映整体情况的。综合以上，下面是我们得到的图像：

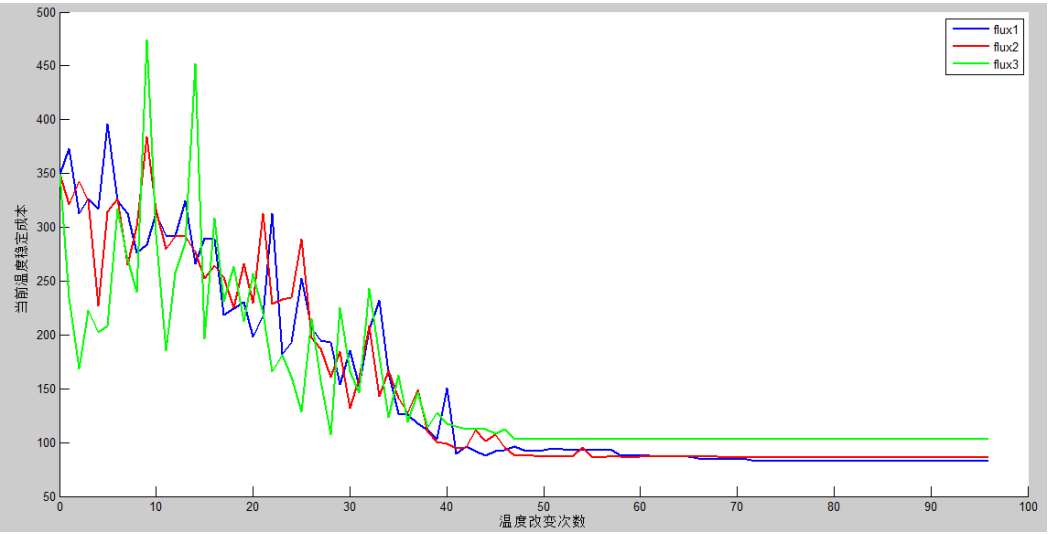

图 6-1 三种不同 flux 运行结果图

从图像可以看出：

(1) 从波动的幅度来看，前期 flux3 波动程度较大，flux1 与 flux2 波动程度较小，且它们之间相差不大；

(2) 从稳定的时间来看，三者几乎同时稳定。其实这是肯定的，因为它们在相同的温度条件下变化，温度高时接受差解概率更大，温度低时概率更小，所以就越稳定；

(3) 从后期的稳定值来看，flux1<flux2<flux3。具体数据是：flux1 为 82.9659，flux2 为 86.5011，flux3 为 102.9360。

从以上三点我们知道，最好的方式为 flux1。因为它在较小的波动下达到了更低的成本，而 flux3 则较差，它在前期大幅度的波动可最后并没有成功到达全局最优解所在局部。

因此我们在用模拟退火算法产生最终结果时，用的扰动方式为 flux1。

7.遗传算法和模拟退火算法的比较

接下来我们从多个角度来比较遗传算法和模拟退火算法，希望能够较为全面的分析这两种算法，得到解决此课题的最好方法。

7.1 从解的角度

这是两种完全不一样的算法，但我们希望寻求某种标准，可以从解随时间的变化关系来比较这两种算法。对此，我们进行了多次讨论。首先，是因变量的选取，也就是这里解的选取，前文已经说过最优解的产生随机性太大，所以我们不能去比较最优解，而是希望寻求一个随机性小且可以代表一定统计规律的值。于是对于遗传算法我们选取的是种群的平均成本；对于退火算法我们选取的是每个温度下的最后一个值，这里需要说明一下，退火算法每个温度下迭代 100 次，每次扰动以一定的概率接受更差的解跳出局部，实际上 100 次后最后一个值是有统计上的意义的，从退火算法的原理来说，它代表了这个温度下固体内部粒子稳定的状态，所以它是可以反映整体情况的。

接下来一个非常棘手的问题就是两者规模的控制。作为无关变量我们要控制两者的规模一致，但是对于两种不同的算法，并没有一个规模相同的标准。由于程序运行的时间几乎全部用在计算成本的函数 costSum 上，所以容易想到，规模相同首先应该满足 costSum 的执行次数相同，可这样仍旧有很多的组合。进一步，我们将退火算法类比遗传算法。退火算法的温度就相当于遗传算法的当前代数；退火算法的每个温度下的迭代就相当于是每一代中种群所有个体的变异。

基于这个思想，我们定义两者的规模相等条件如下：

$$\begin{cases} popSize = \dfrac{Markov\_length}{2} \\ GenerationMax = \log_k^{\frac{T_f}{T_0}} \end{cases} \tag{7-1}$$

其中 popSize 为种群规模，GenerationMax 为进化代数，T_0 为初始温度，T_f 为末态温度，Markov_length 为每个温度下的迭代次数，k 为 Boltzmann 常数。Markov_length 之所以除以 2 是因为退火算法每迭代一次只计算一次 costSum,而遗传算法每个选择个体到下一代要经过交叉互换和基因突变，要计算两次 costSum。

温度从 250 到 0.01 改变了 97 次，所以设置 GenerationMax=97,迭代次数为 100 次，所以设置 popSize=50。交换和突变方式我们选择前面得到的 crossover1 与 mutation2，解的扰动方式我们选择前面得到的 flux1, 在这样的条件下，得到的结果如下图：

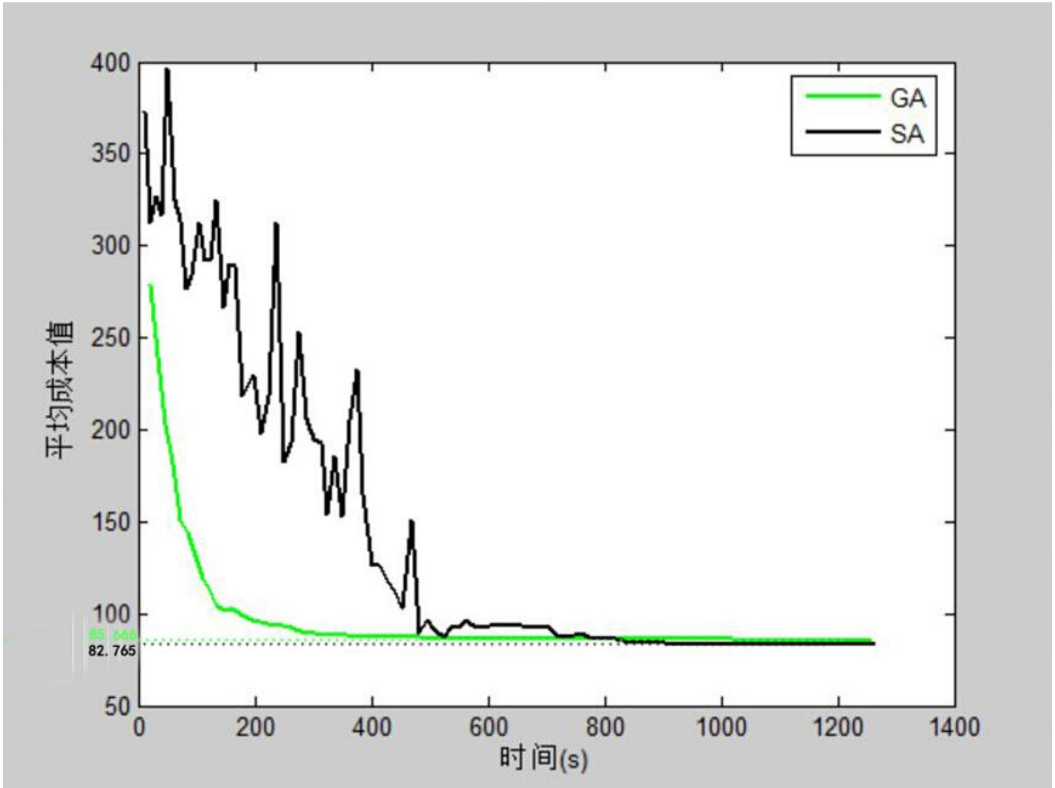

图 7-1 遗传算法与模拟退火算法比较结果图

其中 GA 代表遗传算法，SA 代表退火算法。

分析图像可以得到：

(1) 在不稳定时，相同时刻下，平均成本值 GA<SA；

(2) 比较到达稳定的时间，同样是 GA<SA；

(3) 比较稳定时两者的稳定平均成本，SA<GA；

由以上三点和实验数据我们知道，GA 可以更快的趋于稳定，且在不稳定阶段 GA 代表的成本也更低。但是在稳定后，在平均成本到达 85 时，GA 的进化变得非常困难，但是 SA 仍在不断变优。于是 800s 以后 SA 曲线位于 GA 曲线的下方，并且 SA 到达了之前计算出的全局最优解 82.7655，而 GA 仍旧在 85 附近。

所以我们认为对于此课题，模拟退火算法要优于遗传算法。因为：

(1) 模拟退火算法的本身决定了它在之前不稳定时是波动较大的，所以比较这一段它与遗传算法的优劣是没有意义的；

(2) 最后的结果说明了退火算法的这种波动是很有意义的，正是因为这样随机的波动，一直的跳出局部，才使得退火算法最后成功收敛到了全局最优解所在邻域，成功找到了严格的全局最优解。

(3) 因为执行规模较小，所以我们并不能说遗传算法就陷入了局部最优。但是正是在这样的小规模下，退火算法找到了严格的全局最优，我们多次执行过这样小规模的退火算法，每一次都可以找到严格最优的 82.7655，但是遗传算法则不然，遗传算法在小规模下很难达到 82.7655。

综合以上，我们认为在解的角度上，退火算法要优于遗传算法。

7.2 从简洁的角度

遗传算法需要考虑的因素太多，交叉互换的方式，基因突变的方式，以及适应度的计算方式，这三种方式任一种选择不当都会对遗传算法产生很大影响。但是退火算法则不然，退火算法只需要考虑解扰动的方式即可，更加直观和简洁。

也正是因为如此，就像附录中的代码一样，退火算法的代码不管是从长度还是简洁程度，都大大的由于遗传算法。

所以，我们认为从简洁的角度上，同样是退火算法由于遗传算法。

8.评价

8.1 优点

(1) 我们在本课题中很好的应用了遗传算法和退火算法，并通过对两种算法一系列的纵向比较，找到了遗传算法较好的变异方式以及退火算法较好的扰动方式，使两种算法都能够在有限时间内找到我们用枚举法产生的严格全局最优解。找到的结果无疑是较好的。

(2) 我们还对两种算法进行横向的比较分析得到了合理的结论：退火算法比遗传算法更适合此课题的求解。

(3) 也就是说我们在课题的要求下利用两种解法很好的解决了课题的问题，并且还对解决课题的方法进行了分析、讨论和探究，找到了较好的解决此课题的方法。这两项成果都是极富意义的。

8.2 缺点

(1) 因为时间有限我们并没有尝试其他更为复杂的拟合方式，比如对数据点利用分段线性拟合得到的最低成本可能会更好。

(2) 在算法的纵向比较时，我们虽然控制了初始的个体(状态)完全相同，但更为科学的做法应该是针对大量不同的初始个体，分别绘制曲线，最后再对所绘制曲线加以平均，这样得到的结论更加可靠。因为时间有限，我们并没有这样去做。

(3) 在两种算法的横向比较时，因为遗传算法初始是一个种群，退火算法初始只有一个状态，所以我们没法控制这两个初值相同，于是让他们自由随机产生。但是更为科学的做法是应该用一个衡量的函数制定一个标准来衡量他们的初值相同。

9.参考文献

[1]赵英时.遥感应用分析原理与方法[M].北京:科学出版社,2003:1.

[2]卓金武.Matlab 在数学建模中的应用[M].北京:北京航空航天大学出版社,2014:37-59.

附录

Matlab 源代码

1. 遗传算法

main.m

```
%主文件
clear all;
close all;
BitLength=51;                                   %每个样本的数据数
popsize=30;                                     %每一代种群的大小
Generationnmax=50;                              %最大代数
global data;
data=csvread('data.csv');                       %读取文件
%计算积累概率时选取的上限
%随机产生初始种群
population=zeros(popsize,51);
for i=1:popsize
    population(i,:)=RandomProduce();
end
%计算适应度函数，返回适应度 Fitvalue 和积累概率 accumPro
[Fitvalue,accumPro]=fitness(population);
AfterCross=zeros(popsize,BitLength);            %初始化每次交换后整个种群的矩阵
AfterMutation=zeros(popsize,BitLength);         %初始化每次变异后整个种群的矩阵
PointsBest=zeros(Generationnmax,BitLength);     %每一代最优个体组成的矩阵
CostBest=zeros(Generationnmax,1);               %每一代最优个体对应成本组成的矩阵
AveCost=zeros(Generationnmax,1);        %每一代所有种群的总成本的平均值组成矩阵，用来观测代与代之间的进化程度
Generation=1;
while Generation < Generationnmax+1
    disp(Generation);
    [maxtmp,postmp]=max(Fitvalue);
    Fitvalue(postmp)=0;
    [maxtmp2,postmp2]=max(Fitvalue);
    select(1)=postmp;
    select(2)=postmp2;
    flag=false;
    Fitvalue(postmp)=maxtmp;
    for j=1:2:popsize

        if flag
            select=selection(population,accumPro);      %选择操作,依一定概率选择两个体
        end
```

```
        [selectPoints,z1,z2]=CrossOver(population,select,ProCrossOver);        %交换操作，
z1,z2 分别表示两个个体交换是否成功
        %选择交叉后父代和子代中较优的，作为新的父代
        if z1==0
            cost1=Fitvalue(select(1));
        else cost1=1./((costSum(selectPoints(1,:)).^2));
        end
        if z2==0
            cost2=Fitvalue(select(2));
        else cost2=1./((costSum(selectPoints(2,:))).^2);
        end
        if Fitvalue(select(1))>=cost1
            AfterCross(j,:)=population(select(1),:);
            cost1=Fitvalue(select(1));
        else AfterCross(j,:)=selectPoints(1,:);
        end
        if Fitvalue(select(2))>=cost2
            AfterCross(j+1,:)=population(select(2),:);
            cost2=Fitvalue(select(2));
        else AfterCross(j+1,:)=selectPoints(2,:);
        end
        %变异操作
        [IndividualA,z3]=mutation(AfterCross(j,:));
        [IndividualB,z4]=mutation(AfterCross(j+1,:));
    %选择变异后父代和子代中较优的，作为新的父代,z3,z4 分别表示两个个体是否变异成功
        if z3==1
            tmp=1./((costSum(IndividualA)).^2);
        else tmp=cost1;
        end
        if cost1>=tmp
            AfterMutation(j,:)=AfterCross(j,:);
        else
            cost1=tmp;
            AfterMutation(j,:)=IndividualA;
        end
        if z4==1
            tmp=1./((costSum(IndividualB)).^2);
        else tmp=cost2;
        end
        if cost2>=tmp
            AfterMutation(j+1,:)=AfterCross(j+1,:);
        else
            cost2=tmp;
            AfterMutation(j+1,:)=IndividualB;
```

```
            end
            Fitvalue(j)=cost1;
            Fitvalue(j+1)=cost2;
            flag=true;
        end
        ProCrossOver=ProCrossOver*k;
        population=AfterMutation;              %产生新种群
        %计算新种群的积累概率
        SumAccumCost=sum(Fitvalue);
        ProEvery=Fitvalue/SumAccumCost;                %计算选择概率
        accumPro(1)=ProEvery(1);                       %计算累积概率
        for i=2:popsize
            accumPro(i)=accumPro(i-1)+ProEvery(i);
        end
        %记录当前代最好的适应度和平均适应度
        [value,pos]=max(Fitvalue);              %第一个是最大值，第二个是最大值所在位置

        CostBest(Generation)=1./((value).^0.5);
        disp(CostBest(Generation));
        AveCost(Generation)=mean(1./((Fitvalue).^0.5));

        xx=population(pos,:);                          %记录当前代的最佳染色体个体
        PointsBest(Generation,:)=xx;
        Generation=Generation+1;
    end

    [Bestvalue,minPos]=min(CostBest);              %最佳成本
    BestMethod=PointsBest(minPos,:);
    %绘制平均适应度和最大适应度的曲线，通过图像判断种群是否成熟。
    figure(1)
    hand1=plot(1:Generationnmax,CostBest);
    set(hand1,'linestyle','-','linewidth',1.8,'marker','*','markersize',6)
    hold on;
    hand2 = plot(1:Generationnmax,AveCost);
    set(hand2,'color','r','linestyle','-','linewidth',1.8,'marker','h','markersize',6)
    xlabel('进化代数');ylabel('最小/平均成本');xlim([1 Generationnmax]);
    legend('最小成本','平均成本');
    box off; hold off;
```

RandomProduce.m

```
function  Individual= RandomProduce()
%个体随机生成函数，并在此过程中限定选中观测点数不小于 5
s=0;
while s<4
```

```
Individual = round(rand(1,51));
s=sum(Individual);
end

end
```

selection.m

```
function select= selection(population,accumPro)
%子程序：新种群选择操作，函数名称 selection
%从种群里面选择两个个体
select=zeros(2);
for i=1:2
   r=rand;              %产生随机数
   %disp(cumsump);
   prand=accumPro-r;
   j=1;
   while prand(j)<0
      j=j+1;
   end
   select(i)=j;          %选中个体的序号
end
end
```

fitness.m

```
function [Fitvalue,accumPro]= fitness(population)
%子程序：计算适应度函数，函数名称 fitnessfun
popsize1=size(population,1);
Fitvalue=zeros(1,popsize1);
accumPro=zeros(1,popsize1);
for i=1:popsize1
   x=population(i,:);
   Fitvalue(i)=costSum(x);
   %disp(i);
end
Fitvalue=1./((Fitvalue).^2);       %转化成合适的值
fsum=sum(Fitvalue);          %计算选择概率
Pperpopulation=Fitvalue/fsum;
accumPro(1)=Pperpopulation(1);       %计算累积概率
for i=2:popsize1
   accumPro(i)=accumPro(i-1)+Pperpopulation(i);
   %disp(cumsump1(i));
end
Fitvalue=Fitvalue';
accumPro=accumPro';
```

```
end
```

CrossOver.m

```
function    [selectPoints,z1,z2]=CrossOver(population,select,possibility)
%新种群交叉互换操作
%z1,z2 判断用于是否交换，1 代表已交换
BitLength1=size(population,2);
z1=1;
z2=1;
selectPoints=zeros(2,51);
while sum(selectPoints(1,:))<5 || sum(selectPoints(2,:))<5
    chb=round(rand*(BitLength1-2))+1;          %在[1，Bitlength-1]范围随机产生一个交叉位
    selectPoints(1,:)=[population(select(1),1:chb) population(select(2),1:51-chb)];
    selectPoints(2,:)=[population(select(1),chb+1:51) population(select(2),51-chb+1:51)];
end

end
```

mutation.m

```
function [AfterMutation,z] = mutation( snew)
%子程序：突变变异操作，z 用于判断是否变异成功
BitLength=size(snew,2);
AfterMutation=zeros(1,51);
pmm=1;
z=1;
if rand<0.7
    while sum(AfterMutation)<4
        chb=round(rand*(BitLength-1))+1;%在[1，BitLength]范围内随机产生一个变异位
        snew(chb)=1-snew(chb);
        AfterMutation=snew;
    end
else
    while sum(AfterMutation)<4
        chb=round(rand(1,2)*(BitLength-1))+1;
        snew(chb)=1-snew(chb);
        AfterMutation=snew;
end

end
```

costSum.m

```
function [ average ] = costSum(    points )
```

```
%计算一个个体的成本，points 为选点方案
global data;
currentLength=1;
for i=1:2:937
    x=data(i,:);
    y=data(i+1,:);
    c(currentLength)=costOne(x,y,points);
    currentLength=currentLength+1;
end
    average=mean(c);
end
```

costOne.m

```
function [ cost] = costOne( x,y,points )
%计算一个样本的成本.x,y 为样本数据,points 为选点方案
pos=find(points==1);
x2=x(pos);
y2=y(pos);

%p=polyfit(x2,y2,3);
%q=polyval(p,x);
q=interp1(x2,y2,x,'cubic');
difference=abs(q-y);

sum=0;
for i=1:length(difference)
    if difference(i)>0.5 && difference(i)<=1
        sum=sum+0.5;
    end

    if difference(i)>1 && difference(i)<=2
        sum=sum+1.5;
    end

    if difference(i)>2 && difference(i)<=3
        sum=sum+6;
    end

    if difference(i)>3 && difference(i)<=5
        sum=sum+12;
    end

    if difference(i)>5
        sum=sum+25;
```

```
    end
end
cost=sum+12*length(pos);

end
```

2. 模拟退火算法

main.m

```
k=0.9;
T0=250;Tf=0.01;
global T;
T=T0;
Markov_length=100;
global data;
data=csvread('data.csv');

individual=RandomProduce();
purpose=costSum(individual);
current=0;
while T>Tf
    current=current+1;
    disp(current);
    recordValue(current)=purpose;
    AveValue1(current)=purpose;
    recordMethod(current,:)=individual;
    for i=1:Markov_length
        individualNew=flux(individual);
        purposeNew=costSum(individualNew);
        change=purposeNew-purpose;
        if change<=0
            individual=individualNew;
            purpose=purposeNew;
            if purpose<recordValue(current)
                recordValue(current)=purpose;
                recordMethod(current,:)=individual;
            end

        else if rand<exp(-change/T)
                individual=individualNew;
                purpose=purposeNew;
            end
        end
    end
    %individual=recordMethod(current,:);
```

```
    %purpose=recordValue(current);
    disp(recordValue(current));

    T=k*T;
end

[BestValue,pos]=min(recordValue);
BestMethod=recordMethod(pos,:);
Xaxis=0:current-1;
figure(1)
hand1=plot(Xaxis,recordValue);
set(hand1,'linestyle','-','linewidth',1.8)
hold on;
xlabel('温度变化次数');ylabel('当前温度最小成本');
box off; hold off;
```

flux.m

```
function [ individualNew ] = flux(individual)
%扰动解的函数
global T;
individualNew=zeros(1,51);
while sum(individualNew)<5
    if rand<0.5
        pos1=round(rand*50)+1;
        individual(pos1)=1-individual(pos1);
        individualNew=individual;
    else if rand<0.9
            pos1=round(rand*50)+1;
            pos2=round(rand*50)+1;
            individual([pos1 pos2])=ones(1,2)-individual([pos1 pos2]);
            individualNew=individual;
        else
            pos=round(rand(1,3)*50)+1;
            individual(pos)=1-individual(pos);
            individualNew=individual;
        end
    end
end
end
```

RandomProduce.m

```
function    Individual= RandomProduce()
%个体随机生成函数，并在此过程中限定选中观测点数不小于 5,函数名为 RandomProduce
s=0;
```

```
while s<4
Individual = round(rand(1,51));
s=sum(Individual);
end

end
```

costSum.m

```
function [ average ] = costSum(    points )
%计算一个个体的成本，points 为选点方案
global data;
currentLength=1;
for i=1:2:937
    x=data(i,:);
    y=data(i+1,:);
    c(currentLength)=costOne(x,y,points);
    currentLength=currentLength+1;
end
    average=mean(c);
end
```

costOne.m

```
function [ cost] = costOne( x,y,points )
%计算一个样本的成本.x,y 为样本数据,points 为选点方案
pos=find(points==1);
x2=x(pos);
y2=y(pos);

%p=polyfit(x2,y2,3);
%q=polyval(p,x);
q=interp1(x2,y2,x,'cubic');
difference=abs(q-y);

sum=0;
for i=1:length(difference)
    if difference(i)>0.5 && difference(i)<=1
        sum=sum+0.5;
    end

    if difference(i)>1 && difference(i)<=2
        sum=sum+1.5;
    end

    if difference(i)>2 && difference(i)<=3
```

```
        sum=sum+6;
    end

    if difference(i)>3 && difference(i)<=5
        sum=sum+12;
    end

    if difference(i)>5
        sum=sum+25;
    end
end
cost=sum+12*length(pos);

end
```

3. 枚举法

main.m

```
global data;
data=csvread('data.csv');
BestValue=200;

for i=1:9
    for j=10:19
        for k=20:29
            for m=30:39
                for n=40:51
                    Methods=zeros(51,1);
                    Methods([i j k m n])=1;
                    tmp=costSum(Methods);
                    if tmp<BestValue
                        BestValue=tmp;
                        disp(BestValue);
                        BestMethod=Methods;
                    end
                end
            end
        end
    end
end
```