基于多目标粒子群算法的太阳影子定位

摘　要

本文针对太阳影子定位问题，应用天文物理知识确定太阳影子变化模型，基于约束条件下的单双目标优化，建立数学模型，分析各个参数的变化规律，最终确定直杆的所处位置、日期。

什么关系？如何建立的？

针对问题一，影响直杆影长的主要因素有杆长、直杆所在地的经纬度，测量的日期和时刻，通过求解太阳高度角与直杆长度间的关系可以建立求解影子长度的数学模型，求解 9:00—15：00 间天安门直杆的影子长度，以分钟为间隔得到影长变化曲线，其中当时刻为 12：15，影长最短 3.6628m；影长最长 7.34m 时，时刻为 9：00。还可得到影长变化率曲线，最初影长呈下降趋势，在 12：12 时后影长呈上升趋势,使用控制变量法，独立分析每个参数对影长的影响。

什么影响？如何呈现？

针对问题二，本质上问题二就是问题一的逆求解，但未知量由问题一中的影长变为直杆的经度和纬度，因此建立约束条件下的单目标优化模型，采用模拟退火算法求解全局最优解，求解出直杆的地理坐标（120E,36W），大概位于西安的东北方向。

谁是优化目标？谁是约束条件？

针对问题三，相较于问题二，对已知量进一步进行限制，基于最小二乘法思想，建立双目标优化模型。对经典的粒子群算法进行改进，使用 Python 实现多目标粒子群算法(MOPSO)，旨在求解模型最优解。因地球公转、自转以及南北半球对称，针对附件二、三，分别求解南北半球两组结果如下表。

数据	所处半球	日期	杆长(米)	位置坐标	
				经度	纬度
附件 2	北半球	6 月 23 日	1.974	79.557°	39.377°
	南半球	11 月 27 日	2.010	79.740°	40.150°
附件 3	北半球	2 月 15 日	3.211	110.105°	36.619°
	南半球	8 月 17 日	3.165	227.973°	33.941°

同时对于多目标粒子群算法进行相对误差分析，结果如下表：

数据	附件 2（北半球）	附件 2（南半球）	附件 3（北半球）	附件 3（南半球）
相对误差	0.007643%	0.0262%	5.04%	2.83%

如何解？

针对问题四，首先从视频中提取出 21 张关键帧图片，进行灰度化，二值腐蚀算法以及 Photoshop 处理，通过影长和杆长的像素比值求得实际影长。在已知时间的情况下，解出拍摄地点为$(111.155^{o}E, 42.913^{o}N)$—内蒙古自治区乌兰察布市境内；在视频拍摄日期未知的情况下，求解出视频拍摄地点为$(114.159^{o}E, 34.878^{o}N)$—河南省郑州市中牟县，拍摄日期为 2015 年 6 月 22 日，另一个可能的拍摄地点为$(114.572^{o}E, 34.336^{o}S)$—澳大利亚的西南海域上，拍摄日期为 2015 年 1 月 1 日。并采用灵敏度检验(见表)模型的鲁棒性。

如何提取？等时间间隔还是其他？

没有表，就简单叙述结论！

关键词：多目标规划模型　模拟退火算法　多目标粒子群算法　二值腐蚀算法

一、 问题重述

1.1 问题背景

古往今来，人类对于太阳的研究一直没有简单，早在古希腊时期，人们就已经开始对太阳进行研究，从太阳围绕地球转，到地球围绕太阳转；从地球是宇宙的中心，到太阳是宇宙的中心。对太阳的了解就在这一次次的推翻中进步，即使在今天，人们对太阳的研究依然没有停止，太阳影子定位技术就是通过分析图像中物体影子的变化，从而确定拍摄的日期和位置等信息。

1.2 问题一的重述

问题一中，题干给出了直杆的高度 3m，直杆所在的经纬度（北纬 39 度 54 分 26 秒,东经 116 度 23 分 29 秒），测量当天的日期，要求作出在北京时间 9：00—15：00 时间段内直杆影子长度的变化曲线，再单独研究各个变量对影长的影响。

1.3 问题二的重述

问题二中为了确定水平地面上固定直杆的拍摄地点，以直杆的太阳影子顶点为坐标数据建立数学模型，并将其应用于附件 1，求解出若干可能的拍摄地点。

1.4 问题三的重述

为了确定在水平地面上固定直杆的拍摄地点和日期，以直杆的太阳影子顶点为坐标数据建立数学模型，并将其建立的模型应用于附件 2 和附件 3 中 2 的影子顶点坐标数据，求解出若干的拍摄地点与日期。

1.5 问题四的重述

问题四中，题干要求从给出的视频中分析拍摄视频的地点，已知直杆的高度，测量杆的影子长度，再确定相应的数学模型，一问是已知时间，求解出若干模型给出的可能地点；另一问要求在时间未知的情况下，求出若干个模型给出的可能的地点与日期。

二、 问题分析

2.1 问题一的分析

问题一要求在给定的一系列前提条件下解决直杆影子的变化问题。为了作出影子长度的变化曲线，首先应建立一个关于影子长度的函数表达式，通过分析影响影子长度的变量以及这些变量间的关系，可以建立起影子长度的数学模型，进而得出适应各变量的影子长度表达式。经过分析和查找资料，初步选取的四个影响影长的变量，分别为杆长、测量的时刻、直杆的位置、以及测量当天的日期。本题只要求计算影子的长度而不要求计算影子的方位，因而可以将三维问题降维，只考虑影子在地面上的长短而忽略影子在地面的移动。影长表达式可通过太阳高度角与直杆间的特殊的运算规律得出，得到影长表达式后，根据控制变量的思想，只需改变当前的时刻就可以作出影长变化曲线。使用控制变量的方法独立的研究每一个变量对影长的影响。

2.2 问题二的分析

问题二本质是在基于影子长度的函数表达式之上，已知量只有日期，时刻以及影子长度，求解直杆的长度以及杆的所在经纬度。由于要检验测量影长与实际影长的误差，所以我们可抽象为以求得影长与实际影长的差值平方和为目标函数的单目标规划模型，可同时进一步采用模拟退火算法进行求解。

2.3 问题三的分析

问题三经过分析，相比问题二日期未知，要求求解直杆长度、位置以及日期，单目标模型误差可能较大，所以需要建立双目标规划模型。为了得到全局最优解，需要发掘内在的数字特征，所以以问题二的目标函数和影长变化率差值平方和为两个目标函数，构造双目标规划模型，同时为求解最优解，对粒子群算法进行改进，让其适应于多目标规划模型。

2.4 问题四的分析

问题四要求从给出的视频中寻找信息，从而确定拍摄的地点。所以首先从视频中选择合适时间间隔提取出一定数量的关键帧图片，接下来为了保证尽可能的减小测量影长的误差，对图片进行灰度化、二值化处理，然后使用 Photoshop 建立合适的坐标系，找到直杆顶点像素坐标和影长顶点像素坐标，最后使用比值之比的方法，求出影长。接下来在已知时间的情况下，我们带入问题二的模型求解拍摄地点。在日期未知的情况下，带入问题三的模型求出可能的拍摄时间和地点。

三、 模型假设

1. 假设太阳光线经过大气层后仍是平行光线。
2. 题目中的数据全部真实可靠
3. 假设在题目中的测量日期那天全球都为晴天。
4. 假设题目 4 给出的视频中地面为平整地面。

四、 符号说明

符号	说明	单位
L	直杆的影子	m
A	直杆的高度	m
H	太阳高度角	度
ϕ	当前地区的纬度值	度
δ	太阳赤纬	Rad
t	时角	度
b	太阳赤纬计算产生的中间变量	1
T_R	真太阳时	min
N	测量日期距离 1 月 1 日的天数	天
L_l	附件中影长	m
ϑ	杆的经度	度
θ	杆的纬度	度

β_l	影长变化率	无
f	图像颜色矩阵	无
K	二值图像矩阵	无
a	颜色矩阵中的元素	无
C	像素点个数	个

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 模型的建立

针对问题一要得出特点位置下的直杆在一段时间内影子的变化曲线，通过问题分析已经知道影响影子长度的变量有杆长、测量的时刻、直杆的位置、以及测量当天的日期。查阅相关资料[1]后，初步得知直杆影子的长度 L 等于直杆的长度 A 除以太阳高度角 H 的正切值，用数学公式（1）表示即：

$$L = \frac{A}{\tan H} \tag{1}$$

现在已知了直杆的长度 A，只需计算出太阳高度角 H 就可求得影长 L。由于太阳距离地球很远，本文近似的将太阳光线看作平行光线，地面看作平面，太阳高度角 H 的二维图解如下图（图 1 左）所示，三维图解如下图（图 1 右）所示：

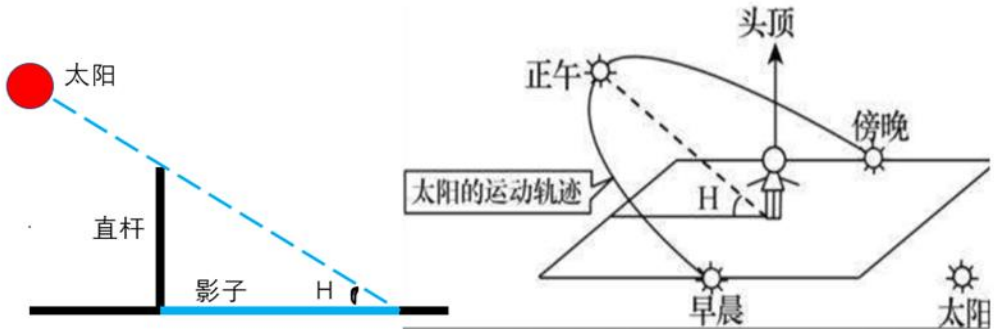


图 1 太阳高度角图解

太阳高度角是太阳光线与某地地面的夹角用符号 H 表示。经查阅资料[2]得知，太阳高度角的求解公式（2）如下所示：

$$\sin H = \sin\phi \times \sin\delta + \cos\phi \times \cos\delta \times \cos t \tag{2}$$

公式 2 中 H 表示太阳高度角（需要计算的值），ϕ 是当前地区的纬度，δ 是太阳赤纬，t 是时角。在公式 2 中，当前地区的纬度是已知量，而剩余的 δ（太阳赤纬）和 t（时角）需要通过下面的公式进行计算。

太阳赤纬角 δ 是地球中心与太阳中心连线和赤道面的夹角（图 2），是确定太阳位置的关键参数。众多专家学者提出了不同的太阳赤纬角的近似算法[3]，本文选用 Spencer 算法对太阳赤纬角进行计算。

公式飘起来了？如何处理？

参考文献不符合规范

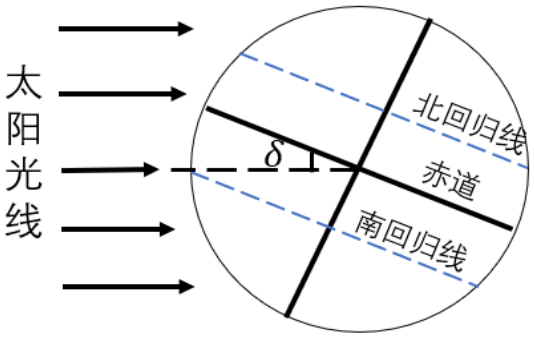


图 2 太阳赤纬角图解

Spencer 太阳赤纬计算公式：

$$\delta（rad）=0.006918-0.39912\cos b+0.070257\sin b-0.006758\cos 2b+0.000907\sin 2b-0.002697\cos 3b+0.00148\sin 3b \tag{3}$$

公式一看就不是自己在 mathtype 中写的！

太阳赤纬计算中的 Spencer 计算公式，相比其他的计算公式如 cooper，stine，bourges 和王炳忠等太阳赤纬近似算法，Spencer 在减小太阳高度角的误差方面有着明显的优势，有实验表明[3]，使用 Spencer 近似算法计算出的太阳高度角与上述算法计算出的太阳高度角进行比较后误差最小。公式 3 中，rad 代表太阳赤纬的单位是弧度，b 是太阳赤纬计算过程中的中间变量，其公式如下所示：

$$b=2\pi\times（N-1）\ 365 \tag{4}$$

公式 4 中的 N 表示的是计算当天的日期距离这一年中的 1 月 1 日的天数，如本题中的测量日期为 10 月 22 日，经查 2015 年日历可得，2015 年的 10 月 22 日距离 2015 年 1 月 1 日相差 295 天，因此 N=295。

为了解决时角 t 的计算问题，首先需要引入真太阳时的概念。

真太阳时：通过当地经度计算出的当地时间，用T_R表示。

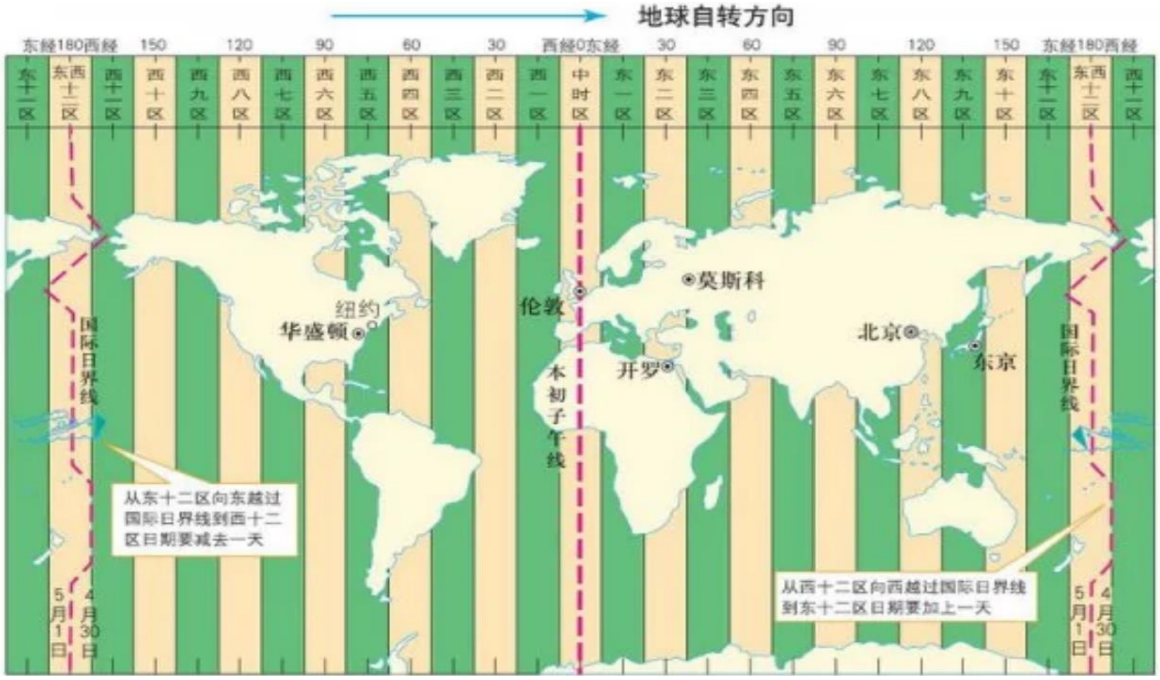


图 3 世界时区图

时角 t 计算公式：

$$t=（T_R-12）\times 15^0 \tag{5}$$

根据时差计算方法，每相差 1 度对应的时间就相差 4 分钟，查阅世界时区图[4]（图 3）得知东经 116 度在北京（东京 120 度）的西边，为了计算当地的真太阳时T_R，因此需要通过计算当地纬度与北京纬度的差值来计算当地时间与北京时间的时差，通过北京时间与当地时差的差值即可得到真太阳时T_R。

将上述五个公式联立，可得到关于影子长度的数学模型，如下公式:

$$\begin{cases} L=\dfrac{A}{\tan H} \\ \sin H=\sin\phi\times\sin\delta+\cos\phi\times\cos\delta\times\cos t \\ \delta（rad）=0.006918-0.39912\cos b+0.070257\sin b-0.006758\cos 2b \\ \qquad +0.000907\sin 2b-0.002697\cos 3b+0.00148\sin 3b \\ b=2\pi\times（N-1）365 \\ t=（T_R-12）\times 15^0 \end{cases} \tag{6}$$

5.1.2 模型的求解

通过 Matlab 数学工具将所有已知值带入函数模型（6）中，即可求得北京时间 9：00—15：00 间直杆影子长度的具体值(代码详见附录 1)。

表 1 影长时刻表

时间（h）	9：00	10：00	11：00	12：00	13：00	14：00	15：00
影子长度（m）	7.34	5.11	4.07	3.67	3.80	4.47	5.99
影长变化率	-0.0554	-0.0247	-0.0112	-0.0021	0.0064	0.0170	0.0365

上表表示每隔一个小时直杆影子长度数值的变化，由表可知直杆影子长度从上午到中午的过程中逐渐减小，从中午到下午的过程中逐渐增加，这也与生活常识相符。使用数学工具 Matlab 可以作出北京时间 9：00—15：00 过程中直杆影子长度的变化曲线（图 4 左）和影长曲线的变化率（图 4 右）。

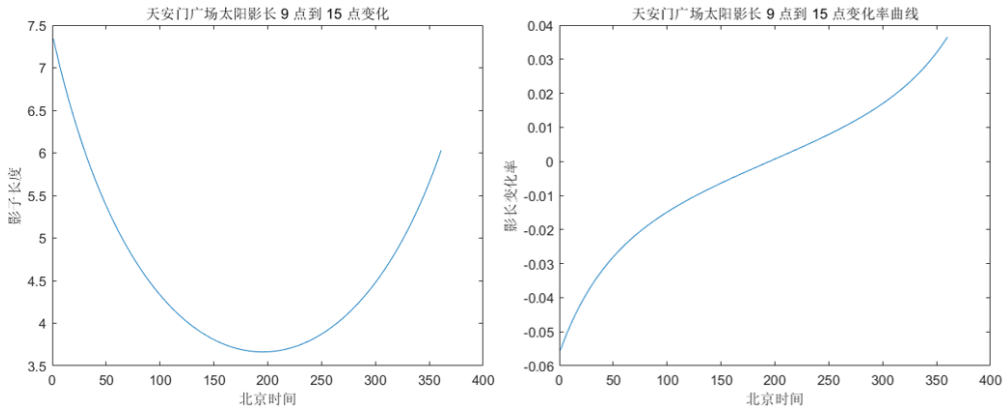


图 4 影长变化曲线

对图 4 进行分析可得到北京时间 9：00—15：00 间当地直杆影子长度的最大值最

小值,以及对应的时刻的影子长度变化率和太阳高度角信息如下表所示：

表 2 影长最值表

	影子长度	对应时刻	影长变化率	太阳高度角 sin 值
影长最短	3.6628m	12:15	0	0.6336
影长最长	7.3402m	9:00	-0.0554	0.3783

表后有说明，对数据的解读，不错！

在北京时间 12：15 时刻的直杆影子长度最短，这时的影子变化率为 0，太阳高度角为一天中最高，其正弦值为 0.6336，在北京时间 9：00 时刻的直杆影子长度最长，这时的影子变化率为-0.0554，影子长度呈下降趋势，太阳高度角的正弦值为 0.3783。

5.1.3 模型的参数分析

本文使用控制变量法独立的研究每个变量对影长的影响，为了得到日期，纬度对影子长度的影响情况，在只改变测量当天日期而不改变其他条件的情况下对影长进行求解（时间设置为 9：00），得到日期影长变化曲线如图所示：

怎么得到的？

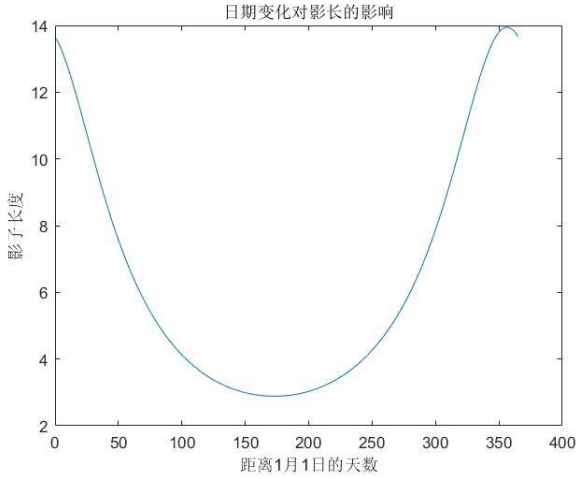


图 5 影长变化曲线（日期改变）

对图 5 进行分析，影子长度在冬季较长，在夏季较短。影子最长的日期为 12 月 22 日长度为 13.994m，影子最短的日期为 6 月 22 日长度为 2.882m。

反映了什么？

在只改变纬度而不改变其他条件的情况下对影长进行求解（时间设置为 9：00），得到纬度影长变化曲线如图所示：

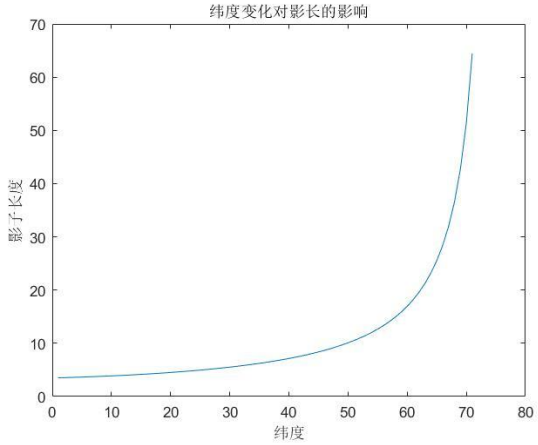


图 6 影长变化曲线（纬度改变）

对图 6 进行分析，影子长度随着直杆所在纬度的增大而增大，纬度小于南北极圈

（66.34^{O}）的纬度时，影子长度随纬度变化呈线性增长，在纬度大于南北极圈（66.34^{O}）后，影子长度随纬度增大呈现指数增长。

5.2 问题二模型的建立与求解

针对问题二，已知量为影长顶点坐标、日期和时刻，通过建立数学模型，求解杆的高度以及杆所在的经纬度。本文将附件 1 的已知数据应用到求解影长的函数公式(6)，得到超定方程组；其次，通过附件 1 分析可知已知条件，同时该问题化简为单目标规划模型；最后采用模拟退火算法，得到杆的高度以及杆所在的经纬度。

5.2.1 单目标规划模型的建立

目标函数是谁？

(1) 确定目标函数

单目标函数为按照附件 1 中的每个时刻，求得影长与对应的实际影长的差值平方和最小，如下公式(7):

$$\min \Delta\lambda_l = \sum_{i=1}^{21} (L_i - L_{li})^2 \tag{7}$$

(2) 确定约束条件

约束条件一：

由于地球自转，地球的每一个都受到太阳光线的照射，所以杆所在的经度范围为：

$$-180° \leqslant \vartheta \leqslant 180° \tag{8}$$

西经度为负，东经度为正。

约束条件二：由于地球公转，南北极圈内会出现极昼或者极夜现象，为排除这种极端情况，式中纬度范围为：

$$-66.34° \leqslant \theta \leqslant 66.34° \tag{9}$$

南纬度为负，北纬度为正。

综上所述，杆所在地点单目标规划模型如下：

$$\min \Delta\lambda_l = \sum_{i=1}^{21} (L_i - L_{li})^2$$

$$s.t \begin{cases} -180° \leqslant \vartheta \leqslant 180° \\ -66.34° \leqslant \theta \leqslant 66.34° \\ L_l = \sqrt{x^2 + y^2} \\ L = f(\theta, \vartheta, d, t, h) \end{cases} \tag{10}$$

整体表述还算是清晰和规范的！

5.2.2 问题二模型的求解

(1)模拟退火算法的概述

模拟退火算法(Simulated Annealing Algorithm,简记为 SSA 或 SA)，它是受物理中固体物质的退火机理与优化问题存在相似性的启发，而建立的一种优化方法，在概率层面，利用随机搜索技术找到目标函数的全局最优解。

(2)算法步骤

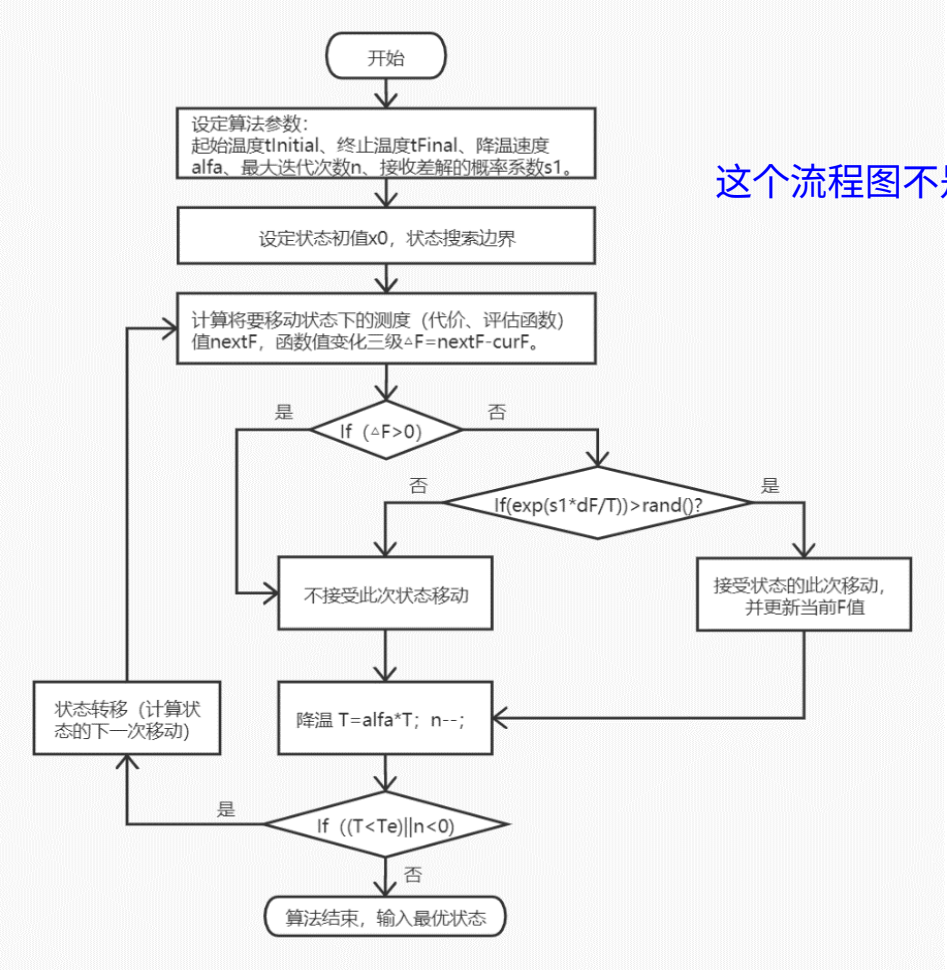
利用 Python 编程语言(源代码见附录 2)编写算法，其中模拟退火算法的核心为搜索最优解的过程可视为随机过程中的马尔科夫过程，根据状态转移概率P^*，由一个候选解转移到另一个候选解。当温度T降为 0 时，x_i的分布为：

$$P_i^* = \begin{cases} \dfrac{1}{|S_{\min}|}, & x_i \in S_{\min} \\ 0, & \text{其他} \end{cases} \tag{11}$$

并且

$$\sum_{x_i \in S_{\min}} P_i^* = 1$$

算法具体流程图如下图 7：



这个流程图不是自己写的吧！

图 7 模拟退火流程图

(3)求解结果

利用公式(10)中的单目标规划模型，并根据附件 1 中的已知数据进行，并采用 Python，最终得到杆的高度以及杆所在的经纬度如下表三：

表 3 附件 1 中杆所在位置

杆高	位置坐标	
	经度	纬度
1.903m	110.791°	18.403°

将位置坐标代入百度卫星地图[5]，得到具体方位如下图 8：

图 8 附件 1 杆具体方位

5.3 问题三模型的建立与求解

针对问题三，已知量为测量影长的时刻、影子顶点坐标，通过建立数学模型，求解杆所在的地理位置、直杆的高度以及测量当天的日期。本文将附件 2 和附件 3 中的数据应用到求解影长的函数公式（6）中，得到超定方程组；其次，通过附件 2 和附件 3 分析已知条件，同时将该问题化简为多目标规划模型；最后本文对经典的粒子群算法进行改进，加入惯性权重，同时应用到多目标规划模型中，得到直杆的高度，直杆所在的经纬度以及测量当天的日期。

5.3.1 多目标规划模型的建立

（1）确立目标函数

双目标函数为按照附件 2、3 中的每个时刻，求得影长与对应的实际影长的差值平方和最小以及影长变化率差值平方和最小，如下公式(12、13):

$$\min \Delta\lambda_l = \sum_{i=1}^{21} (L_i - L_{li})^2 \tag{12}$$

$$\min \Delta\beta_l = \sum_{i=1}^{21} (\beta_i - \beta_{li})^2 \tag{13}$$

(2)确定约束条件:

约束条件一：

由于地球自转，地球的每一个都受到太阳光线的照射，所以杆所在的经度范围为：

$$-180° \leqslant \vartheta \leqslant 180° \tag{14}$$

西经度为负，东经度为正。

约束条件二：

由于地球公转，南北极圈内会出现极昼或者极夜现象，为排除这种极端情况，式中纬度范围为：

$$-66.34° \leqslant \theta \leqslant 66.34° \tag{15}$$

南纬度为负，北纬度为正。

综上所述，直杆所在位置的双目标规划模型如下：

由于两直杆所在地理位置纬度(ϕ)相反，经度(时角t)相同，季节（太阳赤纬角δ）相反的情况下会出现太阳高度角(H)相等的情况，导致分属南北半球的两直杆的影长也相等。

$$\begin{cases} \min \Delta\lambda_l = \sum_{i=1}^{21} (L_i - L_{li})^2 \\ \min \Delta\beta_l = \sum_{i=1}^{21} (\beta_i - \beta_{li})^2 \end{cases} \tag{16}$$

北半球为：

$$s.t \begin{cases} L = f(\theta, \vartheta, d, t, h) \\ \tan\beta_l = \dfrac{y}{x} \\ -180° \leqslant \vartheta \leqslant 180° \\ 0 \leqslant \theta \leqslant 66.23° \\ L_l = \sqrt{x^2 + y^2} \end{cases} \tag{17}$$

南半球为：

$$s.t \begin{cases} L = f(\theta, \vartheta, d, t, h) \\ \tan\beta_l = \dfrac{y}{x} \\ -180° \leqslant \vartheta \leqslant 180° \\ -66.23° \leqslant \theta \leqslant 0 \\ L_l = \sqrt{x^2 + y^2} \end{cases} \tag{18}$$

5. 3. 2 模型的求解

(1) 基于粒子群算法的多目标规划算法概述

粒子群算法，也称为粒子群优化算法(particle swarm optimization，PSO)，是 Eberhart 等[6]在 1995 年提出的新仿算法。该算法通过模拟鸟群觅食行为在求解空间进行目标函数的最优解的过程。在粒子群算法中，每个优化问题的可能被假定为 D 维搜索空间中的一个多边形的顶点，称为“粒子”[7]。所有粒子具有目标函数确定的适应度值，粒子按照确定飞行方向和飞行距离的速度，跟随搜索空间中当前的最佳粒子进行搜索[8]。粒子经初始化后，确定一个随机解，然后通过不断更新个体极值和全局极值的迭代方式来寻优[9]。

基于经典的粒子群算法，加入惯性权重，同时限制粒子的多个适应值。得到核心公式如下：

$$v_i^d = wv_i^{d-1} + c_1 r_1 (pbest_i^d - x_i^d) + c_2 r_2 (gbest^d - x_i^d)$$

其中c_1:个体加速因子 c_2:社会加速因子 w:惯性权重

v_i^d:第d次迭代时第i个粒子的速度

x_i^d:第d次迭代时第i个粒子的位置 (19)

$pbest_i^d$:到第d次迭代为止，第i个粒子经过的最好的位置

$gbest^d$:到第d次迭代为止，所有粒子经过的最好的位置

r_1，r_2是$[0,1]$上的随机数

(2)算法步骤

同样利用 Python 编程语言(源代码见附录 3)编写算法，算法的具体流程图如下:

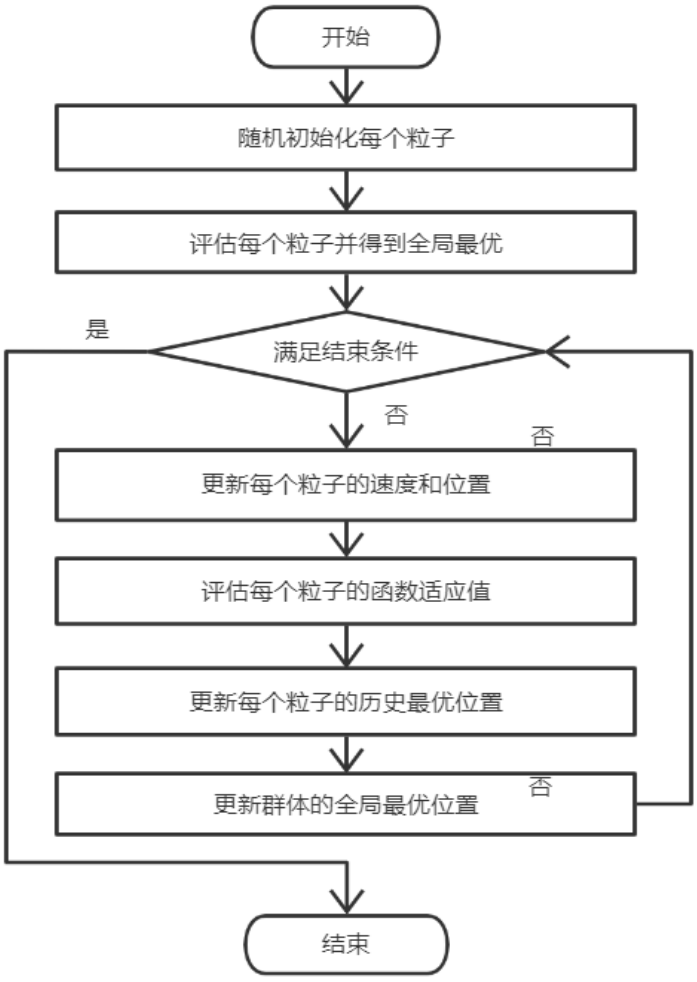


图 9 多目标粒子群算法流程图

(3)求解结果

算法利用公式(16)(17)(18)中的双目标规划模型，并根据附件 2、3 的已知数据进行求解，最终得到杆的经纬度、杆的高度、以及测量日期如下表四所示：

表 4 附件 2、3 中的直杆求解结果

数据	所处半球	日期	杆长(米)	位置坐标	
				经度	纬度
附件 2	北半球	6 月 23 日	1.974	79.557°	39.377°
	南半球	11 月 27 日	2.010	79.740°	-40.150°
附件 3	北半球	2 月 15 日	3.211	110.105°	36.619°
	南半球	8 月 17 日	3.165	227.973°	-33.941°

5.4 问题四模型的建立与求解

5.4.1 图像预处理

重要信息要在摘要中呈现出来！

首先将拍摄到的视频转化为 AVI 格式，用 MATLAB，以 3 分钟为间隔截取视频图片，共提取到 21 张图片，每张图记为$f(i),i=1,2,3,...,21$。为了得到影长更加清晰的图像，因此需要将$f(i)$先化为灰度图，再进行二值腐蚀。

二值腐蚀公式[11]如下：

$$E\left(f,K\right)=f\ominus K=\bigcap_{b\in K}\{a-b|a\in f\} \tag{20}$$

经过预处理后得到的影子图像的轮廓已经逐渐清晰，如下图所示：

图 10 彩色图像

图 11 灰度处理图像

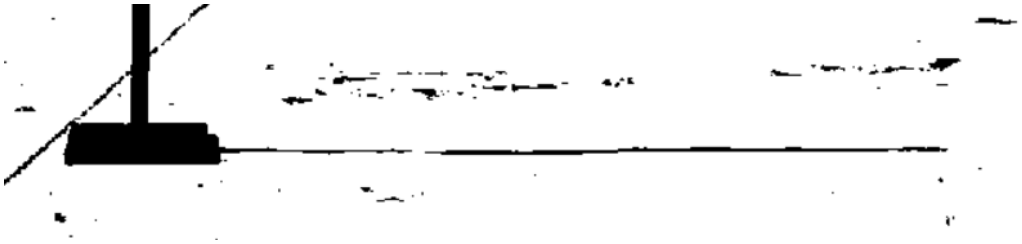


图 12 二值腐蚀图像

使用 PhotoshopCS 软件读取到直杆底端和影子顶点的像素点坐标数据，通过直杆像素与影长像素之比可求得影长。

$$L_l = \frac{C_1}{C_2} h \tag{21}$$

通过实验得出影长的像素相对值，通过 PS 处理，分别得出 21 个影子对应的实际影子长度。

5.4.2 第一问模型建立与求解

建立与问题二相同的单目标最优解模型，将已知量带入之前问题二使用的模拟退火算法算法中，可得到如下表所示：

表 5 直杆位置表

经度（度）	纬度（度）
111.155	42.912

在百度卫星地图中输入直杆的经纬度，查找直杆的位置，大概在西安的东北方向，卫星图(图 13)如下所示：

图 13 直杆卫星定位图

5.4.3 第二问模型建立与求解

建立与问题三相同的多目标最优解模型，将已知量带入之前问题三使用的多目标粒子群算法中，得到两种可能的结果，如下表所示：

表 6 直杆位置表

日期	经度（度）	纬度（度）
6 月 13 日	**114.159**	**34.878**
1 月 1 日	**114.57**	**-34.33**

在百度卫星地图中输入直杆的经纬度，查找直杆的位置，位于北半球的直杆大概在三门峡的东北方向(图 14)，位于南半球的直杆大概在澳大利亚曼吉马普的西边海岸(图 14)卫星图如下图所示：

图 14 直杆位置图

六、 模型的分析与检验

6.1 灵敏度分析

灵敏度分析是研究与分析一个系统(或模型)的状态或输出变化对系统参数或周围条件变化的敏感程度的方法，在本文中，主要对问题四进行灵敏度分析，因为问题四相对于前三问增加了视频数据的提取与预处理的过程，且模型也相对复杂。

主要步骤是：控制其他参数不变的情况下，改变提取视频帧的时间，一开始以每 2 分钟一帧的速度，从 8：54 开始对视频提取 21 组数据，然后将起始时刻进行变换，从 8：55 开始还是以每 2 分钟提取一帧的速度提取视频帧，在已知日期的情况下带入模型进行求解，从而检验视频数据的改变，对模型结果产生的影响，得到的结果如下：

表 7 灵敏度检验对比表

	经度（度）	维度（度）
原始数据	**111.155**	**42.912**
检验数据	**110.931**	**40.737**

对结果进行比较发现，视频拍摄地点坐标并没有因为数据采集点的不同而发生明显波动，对同一视频得出基本相同的地点坐标，说明本模型具有很良好的稳健性。

6.2 误差分析

6.2.1 影长误差分析

本文根据误差公式对双目标优化模型求得的影长与附件 2、3 中的实际影长相对比；即：

$$e = \frac{|L - L_l|}{L_l} \tag{22}$$

上式中L—计算值，L_l—实际值

利用 python 求得双目标优化模型影长与附件 2、3 中实际影长的相对误差如下表：

表 8 相对误差表

数据	附件 2（北半球）	附件 2（南半球）	附件 3（北半球）	附件 3（南半球）
相对误差	**0.007643%**	**0.0262%**	**5.04%**	**2.83%**

由上表所知，双目标优化模型的影长那个与附件 2 实际影长相比，南北半球误差分别为 0.007643%和 0.0262%，双目标优化模型影长与附件 3 实际影长相对比，南北半球误差为 5.04% 和 2.83%，误差都不超过 6%，因此，本文可以较为清晰的得知，使用双目标优化模型所求的影长是合理的。

6.3 视频数据预处理过程中的误差：

采用 Photoshop 软件对二值化后的图片建立坐标系求出影长后并没有考虑视频畸变的情况，所以实际影长可能与视频中的影长有一些差别，造成计算结果的微小偏差。

6.4 局部最优解造成的地理坐标误差：

粒子群算法的本身原理会导致其收敛在局部的最优解。为了减少运算量，也为了避免模型陷入局部最优，在程序运行过程中，有时候需要人为干预来筛去一些的地区，可能造成冗余解或缺解。

七、 模型的评价、改进与推广

7.1 模型的优点　　这个部分不要逐项写这么多，言多必失！！！

7.1.1 问题一模型的优点

问题一中建立的数学模型，将三维的问题降维成二维，忽略影子角度的偏差，更注重计算影子的长度。这样不仅计算时方便简洁而且对比利用三维坐标求解影长的方法求出的影长，在二维空间中使用三角函数求出的影长更精准。

7.1.2 问题二模型的优点

问题二中采用了模拟退火算法建立的数学模型，本文的模拟退火算法[13]可通过调节多个参数来控制在局部寻优以及在全局寻优的能力，算法有着较强的鲁棒性和推广性,算法通过赋予搜索过程一种时变且最终趋于零的概率突跳性，从而可有效避免陷入局部极小并最终趋于全局最优的串行结构的优化算法[11]。

7.1.3 问题三模型的优点

问题三中采用了粒子群算法建立的数学模型，使用多个目标函数，使得最终拟合的结果有着更多的约束条件从而提高了拟合结果与实际数据的误差。PSO 算法具有记忆性，粒子群中的最优解的位置可以记录下来，并且传递给其他粒子，搜索速度快[12]。

7.2 模型的缺点

7.2.1 问题一模型的缺点

问题一建立的模型基于地理学公式，无明显缺点。

7.2.2 问题二模型的缺点

问题二建立的数学模型主要基于模拟退火算法。由于模拟退火算法需要控制多个参数来控制在局部和全局的寻优能力，因此在操作时，随着目标函数的改动需要实时对算法参数进行改动才能保持一个较为精准的拟合结果[12]。

7.2.3 问题三模型的缺点

问题三建立的数学模型主要基于粒子群算法。粒子群本身具有记忆性，且粒子间可以相互传递信息，因此算法在约束条件范围较大的情况下可能会陷入局部最优解[13]。

7.3 模型的改进

7.3.1 问题二模型的改进

模拟退火算法需要调整的参数较多导致调参的困难，因此可以通过预先设定参数进行多次实验的方法减小调参带来的不便。

7.3.2 问题三模型的改进

粒子群算法在约束条件范围较大的情况下可能会陷入局部最优解，这是由 PSO 算法本身特性决定的，但可以通过人为的缩小约束条件的范围来避免，如本题采用南北半球分别计算的方法。

7.3.3 模型的推广

本文所建立的模型通过调整模型内部参数的方法，可适用于多种不同场景，如股票的预测问题、天气的预测问题、传染病的传播速度预测等问题。

八、 参考文献　　每一篇都有引用么？！！

[1]王浩璇,王一鸣.基于 MATLAB 软件的太阳影子定位[J].科学技术创新,2019(13):57-58.
[2]张旭彦.“正午太阳高度角及其应用”教学设计[J].中学地理教学参考,2021(01):47-50.
[3]丁艳,袁隆基,赵培涛,仝军令.太阳视日轨迹跟踪算法研究[J].节能,2020,39(11):95-97.
[4]百度百科‘世界时区图’
[5]百度卫星地图 https://www.earthol.com/bd/
[6]EBERHARTRC,SHIY,Particle swarm optimization: developments,applications and resources［C］// Proceedings of the IEEE congress on evolutionary computation．Piscataway,NJ: IEEE Service Center,2001: 81－86．
[7]SHIY，LIUHC，GAOL，etal.Cellular particle swarm optimization［J］.Information sciences，2011,181(20) : 4460－4493．
[8]DONGY,TANGJF,XUBD,etal．An application of swarm optimization to nonlinear programming［J］Computers＆mathematics with applications,2005,49(11/12) : 1655－1668.
[9]ZHANGY,GONGDW,DINGZH.A bare-bones multi-objective particle swarm optimization algorithm for environmental/economic dispatch［J］.Information sciences，2012,192: 213－227．
[10]李俊．一种基于迁移学习的印章识别方法[P]．上海市：CN113159015A,2021-07-23.
[11]牛军霞.代价敏感属性中模拟退火算法和信息增益算法的比较[J].湖北农机化,2020(16):112-113.
[12]张佳仁,张婧瑜,王鹏,张建忠.用模拟退火算法拟合铂电阻的热特性曲线[J].上海计量测试,2015,42(03):20-22.
[13]左右宇.基于粒子群算法的 RFID 天线优化设计[J].中国设备工程,2020(17):114-116.

该篇文章整体上还不错，基本掌握了写作的套路。建议后期重点关注一下几点

1.论文格式的规范性
2.摘要要好好写，可能体现你想法的要素是否写清楚了
3.关注模型建立的合理性
4.求解等细节应该更丰富一些

如果都是自己的工作，目前大概是省一水平！

附录

前面加一个小表格，介绍每个附录！

附录 1

介绍：Matlab，影子长度函数

```
clear
b = 2 * pi * (295-1)/365
p = 0.006918 - 0.399912*cos(b) + 0.070257*sin(b) -
0.006758*cos(2*b) + 0.000907*sin(2*b) - 0.002697*cos(3*b) +
0.00148*sin(3*b)
jingdu = dms2degrees([116 23 29])
shicha = (120-jingdu)*4
st = [540-shicha:900-shicha]
t = 15*(st-720)/60
d =dms2degrees([39 54 26])
H = sind(d) * sin(p) + cosd(d) * cos(p) * cosd(t)
x = asin(H)
finish = tan(x)
plot(finish)
L = 3./finish
plot(L)
xlabel('北京时间')
ylabel('影子长度')
max(L)
[m,n]=find(L==max(L))
D = diff(L)
title('天安门广场太阳影长 9 点到 15 点变化率曲线');
xlabel ('北京时间');
ylabel ('影长变化率');
newlie=[]
for i=1:60:360
    data=D(1:i)
    newlie=[newlie;data]
end
```

附录 2

介绍：Python 编写 模拟退火算法 完整代码见支撑材料

```
import math                          #
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
from datetime import datetime

# 原始函数
def fuction1(X):
    h,j,w = 2,X[0],X[1]  # h 为 2，最优
    # b 太阳高度角中的中间变量
    # p 太阳赤角
    day=108
    b=2*math.pi*(day-1)/365
    p=0.006918-0.399912*math.cos(b)+0.070257*math.sin(b)-
0.006758*math.cos(2*b) +0.000907*math.sin(2*b)-
0.002697*math.cos(3*b)+0.00148*math.sin(3*b)
    # print(p)
    jindu=j#当前经度
    shicha=(120-jindu)*4
# 当前位置和北京时间的时差
    rfinish=[]
    # # 882-942
    for i in range(882,943,3):
        st=i-shicha
    # %北京时间的 9-15 点在当地时间的时刻
        # 882-942
        # t=[i for i in range(60)]
        # for i in range(60):
        #     t[i] = 15 * (st[i] - 720) / 60
        t=15*(st-720)/60
        # 时角
        d=w
    #    %当地纬度
        d=math.radians(d)
        t=math.radians(t)

        H = math.sin(d) * math.sin(p) + math.cos(d) * math.cos(p) *
math.cos(t)

        x = math.asin(H)
        finish=math.tan(x)
        # % 太阳高度角的真切值

        L=h/finish
        rfinish.append(L)
    return rfinish

def cal_Energy(X, nVar, mk):
```

```
    x_coord =
[1.0365,1.0699,1.1038,1.1383,1.1732,1.2087,1.2448,1.2815,1.3189,1.3568,
1.3955,1.4349,1.4751,1.516,1.5577,1.6003,1.6438,1.6882,1.7337,1.7801,1.
8277]
    y_coord =
[0.4973,0.5029,0.5085,0.5142,0.5198,0.5255,0.5311,0.5368,0.5426,0.5483,
0.5541,0.5598,0.5657,0.5715,0.5774,0.5833,0.5892,0.5952,0.6013,0.6074,0
.6135]
    Lb = []
    for i in range(len(x_coord)):
        Lb.append( (x_coord[i]**2+y_coord[i]**2)**0.5 )

    Li = fuction1(X)

    fx = 0
    for i in range(len(Li)):
        fx += (Li[i]-Lb[i])**2
    return fx
def ParameterSetting():
    cName = "funcOpt"
    nVar = 2
    xMin = [ -180, -66.34]
    xMax = [ 180, 66.34]

    tInitial = 1000.0
    tFinal  = 0.1
    alfa    = 0.98
    meanMarkov = 500
    scale   = 0.2
    return cName, nVar, xMin, xMax, tInitial, tFinal, alfa, meanMarkov,
scale

def
OptimizationSSA(nVar,xMin,xMax,tInitial,tFinal,alfa,meanMarkov,scale):
    randseed = random.randint(1, 100)
    random.seed(randseed)
    xInitial = np.zeros((nVar))
    for v in range(nVar):
        xInitial[v] = random.uniform(xMin[v], xMax[v])
    fxInitial = cal_Energy(xInitial, nVar, 1)

    xNew = np.zeros((nVar))
    xNow = np.zeros((nVar))
    xBest = np.zeros((nVar))
```

```
    xNow[:]  = xInitial[:]
    xBest[:] = xInitial[:]
    fxNow  = fxInitial
    fxBest = fxInitial

print('x_Initial:{:.6f},{:.6f},\tf(x_Initial):{:.6f}'.format(xInitial[0
], xInitial[1], fxInitial))

    recordIter = []
    recordFxNow = []
    recordFxBest = []
    recordPBad = []
    kIter = 0
    totalMar = 0
    totalImprove = 0
    nMarkov = meanMarkov

    tNow = tInitial
    while tNow >= tFinal:

        kBetter = 0                #
        kBadAccept = 0             #
        kBadRefuse = 0
        for k in range(nMarkov):
            totalMar += 1
            xNew[:] = xNow[:]
            v = random.randint(0, nVar-1)
            xNew[v] = xNow[v] + scale * (xMax[v]-xMin[v]) *
random.normalvariate(0, 1)
            xNew[v] = max(min(xNew[v], xMax[v]), xMin[v])
            fxNew = cal_Energy(xNew, nVar, kIter)
            deltaE = fxNew - fxNow

            if fxNew < fxNow:
                accept = True
                kBetter += 1
            else:
                pAccept = math.exp(-deltaE / tNow)
                if pAccept > random.random():
                    accept = True
                    kBadAccept += 1
                else:
                    accept = False
                    kBadRefuse += 1
```

```
            if accept == True:  # 如果接受新解，则将新解保存为当前解
                xNow[:] = xNew[:]
                fxNow = fxNew
                if fxNew < fxBest:  # 如果新解的目标函数好于最优解，则将新解
保存为最优解
                    fxBest = fxNew
                    xBest[:] = xNew[:]
                    totalImprove += 1
                    scale = scale*0.99
        pBadAccept = kBadAccept / (kBadAccept + kBadRefuse)
        recordIter.append(kIter)  # 当前外循环次数
        recordFxNow.append(round(fxNow, 4))  # 当前解的目标函数值
        recordFxBest.append(round(fxBest, 4))  # 最佳解的目标函数值
        recordPBad.append(round(pBadAccept, 4))  # 最佳解的目标函数值

        if kIter%10 == 0:                           # 模运算，商的余数
            print('i:{},t(i):{:.2f}, badAccept:{:.6f},
f(x)_best:{:.6f}'.\
                format(kIter, tNow, pBadAccept, fxBest))

        # 缓慢降温至新的温度，降温曲线：T(k)=alfa*T(k-1)
        tNow = tNow * alfa
        kIter = kIter + 1
        fxBest = cal_Energy(xBest, nVar, kIter)  # 由于迭代后惩罚因子增
大，需随之重构增广目标函数
        # ====== 结束模拟退火过程 ======

    print('improve:{:d}'.format(totalImprove))
    return
kIter,xBest,fxBest,fxNow,recordIter,recordFxNow,recordFxBest,recordPBad

# 结果校验与输出
def
ResultOutput(cName,nVar,xBest,fxBest,kIter,recordFxNow,recordFxBest,rec
ordPBad,recordIter):
    fxCheck = cal_Energy(xBest, nVar, kIter)
    if abs(fxBest - fxCheck)>1e-3:
        print("Error 2: Wrong total millage!")
        return
    else:
        print("\nOptimization by simulated annealing algorithm:")
        for i in range(nVar):
```

```
            print('\tx[{}] = {:.6f}'.format(i,xBest[i]))
        print('\n\tf(x):{:.6f}'.format(cal_Energy(xBest,nVar,0)))
    return


def main():

    [cName, nVar, xMin, xMax, tInitial, tFinal, alfa, meanMarkov,
scale] = ParameterSetting()

[kIter,xBest,fxBest,fxNow,recordIter,recordFxNow,recordFxBest,recordPBa
d] =
OptimizationSSA(nVar,xMin,xMax,tInitial,tFinal,alfa,meanMarkov,scale)
    ResultOutput(cName,
nVar,xBest,fxBest,kIter,recordFxNow,recordFxBest,recordPBad,recordIter)

if __name__ == '__main__':
    main()
```

附录 3

介绍：Python 多目标粒子群算法　完整代码见支撑材料

```
import numpy as np
from fitness_funs import *
import init
import update
import plot
class Mopso:

def __init__(self,particals,w,c1,c2,max_,min_,thresh,mesh_div=10):
                self.w,self.c1,self.c2 = w,c1,c2
                self.mesh_div = mesh_div
                self.particals = particals
                self.thresh = thresh
                self.max_ = max_
                self.min_ = min_
                self.max_v = (max_-min_)*0.05  #速度下限
                self.min_v = (max_-min_)*0.05*(-1) #速度上限
                #self.plot_ = plot.Plot_pareto()
```

```
        def evaluation_fitness(self):
                #计算适应值
                fitness_curr = []
                for i in range((self.in_).shape[0]):
                        fitness_curr.append(fitness_(self.in_[i]))
                self.fitness_ = np.array(fitness_curr) #适应值
        def initialize(self):
                #初始化粒子坐标
                self.in_ =
init.init_designparams(self.particals,self.min_,self.max_)
                #初始化粒子速度
                self.v_ =
init.init_v(self.particals,self.min_v,self.max_v)
                #计算适应值
                self.evaluation_fitness()
                #初始化个体最优
                self.in_p,self.fitness_p =
init.init_pbest(self.in_,self.fitness_)
                #初始化外部存档
                self.archive_in,self.archive_fitness =
init.init_archive(self.in_,self.fitness_)
                #初始化全局最优
                self.in_g,self.fitness_g =
init.init_gbest(self.archive_in,self.archive_fitness,self.mesh_div,self
.min_,self.max_,self.particals)
        def update_(self):
                #更新粒子坐标、粒子速度、适应值、个体最优、外部存档、全局最优
                self.v_ =
update.update_v(self.v_,self.min_v,self.max_v,self.in_,self.in_p,self.i
n_g,self.w,self.c1,self.c2)
                self.in_ =
update.update_in(self.in_,self.v_,self.min_,self.max_)
                self.evaluation_fitness()
                self.in_p,self.fitness_p =
update.update_pbest(self.in_,self.fitness_,self.in_p,self.fitness_p)
                self.archive_in,self.archive_fitness =
update.update_archive(self.in_,self.fitness_,self.archive_in,self.archi
ve_fitness,self.thresh,self.mesh_div,self.min_,self.max_,self.particals
)
                self.in_g,self.fitness_g =
update.update_gbest(self.archive_in,self.archive_fitness,self.mesh_div,
self.min_,self.max_,self.particals)
        def done(self,cycle_):
                self.initialize()
```

```
        #self.plot_.show(self.in_,self.fitness_,self.archive_in,self.ar
chive_fitness,-1)
                for i in range(cycle_):
                        self.update_()

        #self.plot_.show(self.in_,self.fitness_,self.archive_in,self.ar
chive_fitness,i)
                return self.archive_in,self.archive_fitness
```