基于太阳辐射规律的光伏板朝向设计

摘　要

对于问题一，大气层对太阳能直射辐射的衰减变化量与其辐射强度、所穿过的大气层厚度成正比。对此，我们通过分析太阳光线变化过程中的几何关系，建立了**太阳辐射衰减模型**。我们利用所给的 5 月 23 日地表太阳直射强度数据，确定了衰减系数的值。然后，我们通过分析太阳光线与倾斜面的关系，建立了**倾斜面上的辐射模型**。通过分析斜面上的瞬时太阳辐射强度变化规律，可以得到相应的最大辐射强度。每日太阳辐射总能量即为所建立的函数对时间的积分。但由于函数关系较为复杂，我们使用**离散型数值积分**方法进行近似计算，得到了比较符合实际的结果。

对于问题二，需要设计出固定安装太阳能光伏板的朝向，使光伏板在晴天条件下受到的太阳直射辐射日均总能量最大。我们分别基于**遗传算法**和**模拟退火算法**建立了优化模型。遗传算法和模拟退火算法都是启发式搜索算法，在处理一些复杂的的优化问题时具有良好的鲁棒性和通用性。基于两种模型的计算结果，通过多维比较，我们确定了在该题背景下基于模拟退火算法的优化模型得出的解更优，确定了太阳能光伏板的水平方位角和倾角分别为 $-0.23°$**,**$27.86°$。

对于问题三，为了解决该多目标优化问题，我们在遗传算法的基础上，设计出**基于精英策略的多目标非支配排序遗传算法** NSGA-II。通过快速非支配排序算法识别出非支配个体，同时通过精英策略，将父代种群和子代种群合并，从合并后的种群中根据非支配排序和拥挤度选择个体组成新的父代种群，确保了优秀个体能够保留到下一代中，从而提高了算法的收敛速度和解的精度。根据 NSGA-II 得出的**最优解分布以及 Pareto 前沿面**，我们采用 **CRITIC 权重法**来确定储电效率和储电量的权重，得到方位角为 $0.84°$**，倾角** $28.26°$, 日均总能量和相应时长为 $4396.54\mathbf{W{\cdot}h}$**，9.27h**。

关键词： 太阳辐射衰减模型　倾斜面上的辐射模型　模拟退火算法　基于精英策略的非支配排序算法

目录

1 问题重述

1.1 问题背景

太阳能电池板也叫光伏板，它利用光伏效应接收太阳辐射能并转化为电能输出，经过充放电控制器储存在蓄电池中。太阳能辐射由直射辐射和散射辐射组成，其中直射辐射对聚集太阳能系统起到了至关重要的影响。受地球运行轨道及太阳光传播的距离影响，大气层外层太阳能辐射强度随时间发生改变。安装光伏板的朝向直接影响到光伏板获得太阳辐射能量的多少。光伏板的朝向包括方位角和水平仰角，当太阳光线和光伏板的法线方向一致时，光伏板瞬时受到的太阳照射能量最大，否则会有太阳辐射能量的损失。当光板受到太阳直射强度过低时，它转换电能的效率也很低；而当光伏板受到太阳直射强度过高时，它转换电能实现储电的效率也会受到限制。现考虑太阳辐射强度及角度随时间变化规律，设计模型确定光伏板固定安装的最优朝向。

1.2 问题要求

根据以上背景信息，建立数学模型对以下具体问题进行讨论：

1. 计算 2025 年每月 15 日，在晴天条件下，该城区一块面积为 $1\mathrm{m}^2$ 的光伏板朝向正南方且水平倾角分别为 $20°, 40°, 60°$ 时受到的最大太阳直射强度和太阳直射辐射总能量.

2. 设计该城区固定安装太阳能光伏板的朝向，使光伏板在晴天条件下受到的太阳直射辐射日均总能量最大；

3. 综合考虑路灯蓄电池的储电效率高和储电量大这两个目标，设计出光伏板固定安装的最优朝向，并计算晴天条件下光伏板受到的太阳直射辐射日均总能量和太阳直射辐射（上午大于 $150\mathrm{W/m}^2$、下午大于 $100\mathrm{W/m}^2$）时长。

图 1: 装载太阳能电池板的路灯

2 问题分析

对于问题一，首先我们假定我们所获得的 5 月 23 日地表太阳直射强度和 1-12 月份大气层外层太阳能辐射强度的数据是可靠的，且每年相同日期的太阳辐射强度规律是相同的。由此我们可以利用题目中所给的两个数据进行分析与计算。由题目背景信息可知，大气层对太阳能直射辐射的衰减变化量与其辐射强度、所穿过的大气层厚度成正比。随着时间的变化，太阳辐射强度和太阳光线所穿过的大气层厚度也会随之改变。对此，我们通过分析太阳光线变化过程中的几何关系，建立了太阳辐射衰减模型。我们利用所给的 5 月 23 日地表太阳直射强度数据，确定了衰减系数的值，并由此计算出特定时间穿过大气层后的太阳辐射强度。然后，我们通过分析太阳光线与倾斜面的关系，建立了倾斜面上的辐射模型。对于问题中给定斜面的倾角，我们得到了任意时刻斜面上的瞬时太阳辐射强度。通过分析斜面上的瞬时太阳辐射强度变化规律，我们可以得到相应的最大辐射强度。而我们发现每日太阳辐射总能量即为所建立的函数对时间的积分。但由于函数关系较为复杂，我们使用离散型数值积分方法进行近似计算，得到了比较符合实际的结果。

对于问题二，需要在问题一建立的模型基础上，设计出固定安装太阳能光伏板的朝向，使光伏板在晴天条件下受到的太阳直射辐射日均总能量最大。依据问题背景，这是一个双变量单目标优化问题，又考虑到此题非线性等实际情况，我们分别基于遗传算法和模拟退火算法建立了优化模型。遗传算法和模拟退火算法都是启发式搜索算法，在处理一些复杂的的优化问题时具有良好的鲁棒性和通用性。基于两种模型的计算结果，通过多维比较，我们确定了在该题背景下基于模拟退火算法的优化模型得出的解更优，从而确定了符合实际的太阳能光伏板的水平方位角和倾角。

对于问题三，需要在前两个问题建立的模型基础上，设计出同时能满足路灯蓄电

池的储电效率高和储电量大两个目标的光伏板固定安装最优朝向。根据问题背景，储电效率与光伏板受到太阳直射强度上午大于 150W/m^2、下午大于 100W/m^2 的时间呈正相关关系，储电量同光伏板受到的太阳直射辐射日均总能量呈正相关关系。因此为了解决该多目标优化问题，我们在遗传算法的基础上，设计出基于精英策略的多目标非支配排序遗传算法 NSGA-II。算法的主要策略是通过快速非支配排序算法识别出非支配个体，即 Pareto 最优解。同时通过精英策略，即通过将父代种群和子代种群合并，然后从合并后的种群中根据非支配排序和拥挤度选择个体组成新的父代种群，确保了优秀个体能够保留到下一代中，从而提高了算法的收敛速度和解的精度。根据 NSGA-II 得出的最优解分布以及 Pareto 前沿面，我们采用 CRITIC 权重法（一种比熵权法和标准离差法处理效果更优的客观赋权法）来确定储电效率和储电量的权重，从而得出比较符合实际问题答案。

3 模型假设

假设 1 我们只考虑太阳直射辐射对太阳能光伏板工作的影响。

解释. 直射辐射是太阳能辐射中最主要的部分，为了简化问题，我们不再考虑其他类型的辐射。

假设 2 我们忽略温度对太阳能光伏板发电效率的影响。

解释. 温度过高会显著影响太阳能光伏板的发电效率，但由于缺乏相关信息且为使问题尽可能简化，我们在此问题中暂不考虑温度的影响。

假设 3 我们仅在晴天的情况下进行分析，计算并由此确定最佳朝向。

解释. 由于阴雨天气及极端天气出现较为随机，难以准确预测未来的天气状况，故我们仅在晴天的前提下进行计算。

假设 4 我们所获得的 5 月 23 日地表太阳直射强度和 1-12 月份大气层外层太阳能辐射强度的数据是可靠的，且每年相同日期的太阳辐射强度规律是相同的。

解释. 参考资料中建立的公式仅与一年中的具体日期与时间点有关，未涉及年份的计算，我们暂且忽略年度差异。

4 符号说明

符号	说明	单位
t	每天时刻	小时
n	从该年第一天起计算的天数	\
m	月份	天
$\omega(t)$	时角	度
ϕ	所处地区纬度	度
$\delta(n)$	赤纬角	度
$\alpha(t,n)$	太阳高度角	度
$A(t,n$	太阳方位角	度
$I_0(m)$	每月太阳辐射衰减前的值	$\mathrm{W/m^2}$
$I(t,n)$	衰减后单位法向平面太阳辐射强度	$\mathrm{W/m^2}$
$I_r(t,n)$	每天单位太阳辐射	$\mathrm{W/m^2}$
K	衰减系数	$\mathrm{W/m^2}\cdot km$
$L(t)$	穿过大气层距离	千米
x	光伏板的方位角（从南向东逆时针旋绕）	度
y	光伏板的水平倾角	度
β	太阳直射辐射日均总能量权重系数	\
γ	太阳辐射直射时长权重系数	\

5 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 太阳辐射衰减模型

首先，我们由相关资料获取了关于赤纬角、太阳高度角、太阳时角等相关概念及计算公式，说明如下：

时角

时角是以正午 12 点为 0 度开始算，每一小时为 15 度，上午为负下午为正，即 10

点和 14 点分别为-30 度和 30 度。因此，时角的计算公式为

$$\omega = 15(t-12)^\circ$$

其中 t 表示 24 小时制时间。

赤纬角

赤纬角是太阳直射的纬度，其计算公式近似为

$$\delta = 23.45\sin(\frac{2\pi(284+n)}{365})^\circ$$

其中 n 为日期序号 (表示一年中的第 n 天)。

太阳高度角

太阳高度角是太阳相对于地平线的高度角，计算公式为

$$\sin\alpha = \sin\phi\sin\delta + \cos\phi\cos\delta\cos\omega$$

其中 ϕ 表示当地纬度。

太阳方位角

太阳方位角是太阳在方位上的角度，它通常被定义为从北方沿着地平线顺时针量度的角。计算公式为:

$$\cos A = \frac{\sin\delta - \sin\alpha\sin\phi}{\cos\alpha\cos\phi}$$

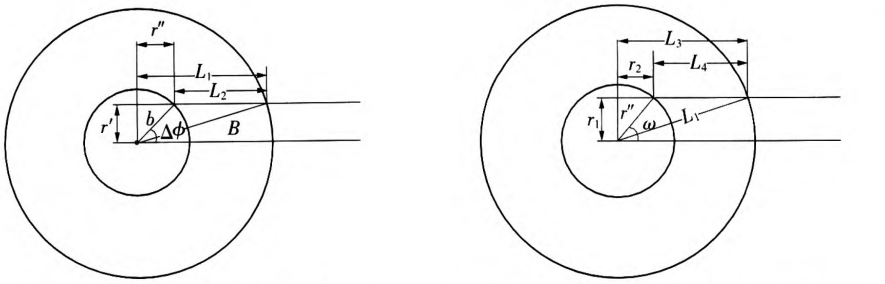
当时角为负值时 (上午), 方位角的角度小于 180°, 时角为正值时 (下午), 方位角应该大于 180°, 即要取补角的值。

由题目背景信息可知，大气层对太阳能直射辐射的衰减变化量与其辐射强度、所穿过的大气层厚度成正比。我们设 $I_0(m)$ 为每月太阳辐射衰减前的值,I 为衰减后单位法向平面太阳辐射强度,K 为衰减系数, L 为穿过大气层距离。则有:

$$\frac{\mathrm{d}I}{\mathrm{d}L} = KI \tag{1}$$

$$I = I_0\mathrm{e}^{-KL} \tag{2}$$

随着时间的变化，太阳光线所穿过的大气层厚度也会随之改变。下面是太阳光线在大气层传播的示意图:



(a) 沿经度圈剖开的大气与地球截面　　(b) 沿纬度圈剖开的大气与地球截面

图 2: 太阳光线的传递分析图

其中 b 为地球半径，B 为大气半径。对此，我们通过分析太阳光线变化过程中的几何关系，发现有:

$$r' = b\sin(\phi - \delta) \tag{3}$$

$$r'' = b\cos(\phi - \delta) \tag{4}$$

$$L_1 = \sqrt{B^2 - r'^2} \tag{5}$$

$$L_2 = L_1 - r'' \tag{6}$$

$$r_1 = r''\sin\omega \tag{7}$$

$$r_2 = r''\cos\omega \tag{8}$$

$$L_3 = \sqrt{L_1^2 - r_1^2} \tag{9}$$

$$L_4 = L_3 - r_2 \tag{10}$$

利用所给的 5 月 23 日地表太阳直射强度数据和公式2确定了衰减系数的值，并由此可以计算出特定时间穿过大气层后的太阳辐射强度。

5.1.2 倾斜面上的辐射模型

为了方便分析斜面上的太阳辐射强度，我们考虑建立空间直角坐标系，分别以正南，正东，垂直于地面向上为 x 轴，y 轴，z 轴正方向。示意图如下:

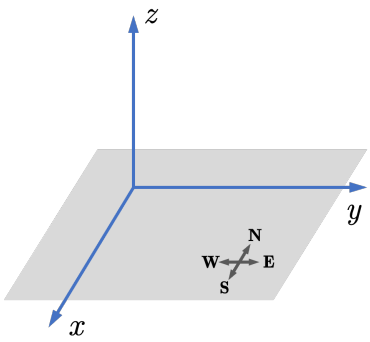


图 3: 空间坐标系的示意图

考虑将太阳直射强度 I 分解为 (x, y, z) 三个方向上的分量, 即有:

$$I_x = -I\cos\alpha\cos A \tag{11}$$

$$I_y = I\cos\alpha\sin A \tag{12}$$

$$I_z = I\sin\alpha \tag{13}$$

用向量符号 $\vec{I} = (I_x, I_y, I_z)^\intercal$ 表示。用 $\vec{d}$ 表示太阳光线的单位向量，有:

$$\vec{d} = (-\cos\alpha\cos A, \cos\alpha\sin A, \sin\alpha)^\intercal$$

对于光伏板的法向量，记为

$$\vec{\gamma} = (\sin\xi\cos\eta, \sin\xi\sin\eta, \cos\xi)^\intercal$$

则斜面实际上接收到的太阳辐射强度为

$$I_r = \vec{\gamma}\cdot\vec{I} = I(t)\cdot R_a$$

$$R_a = \vec{\gamma}\cdot\vec{d}$$

R_a 反映了斜面所接受到的辐射强度与实际太阳辐射强度的比例。

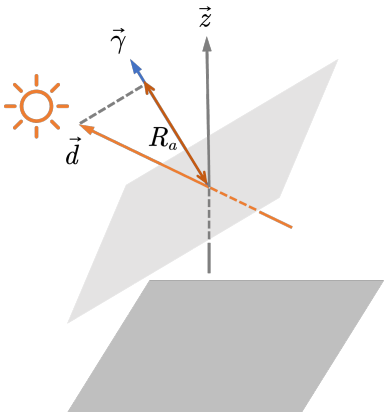
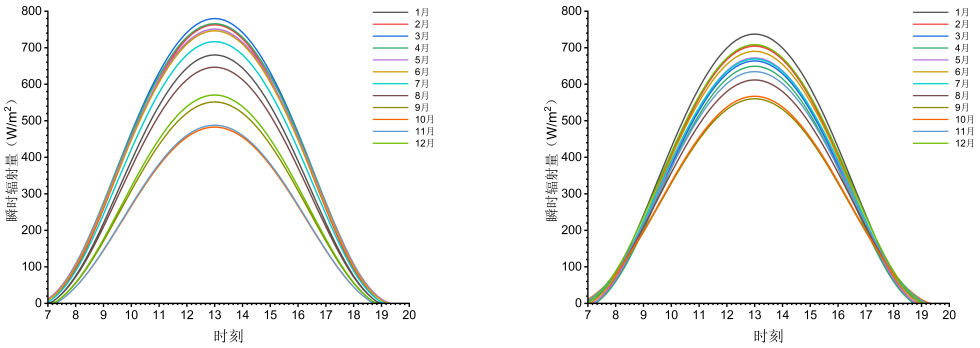
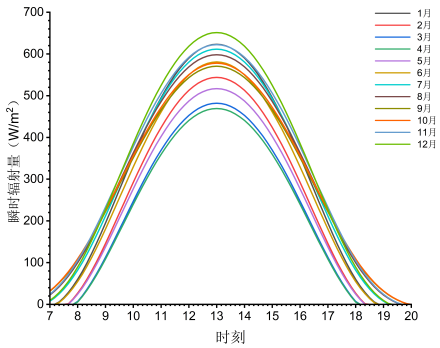


图 4: 斜面受到的辐射强度示意图

对于问题中给定斜面的倾角，我们得到了任意时刻斜面上的瞬时太阳辐射强度。通过分析斜面上的瞬时太阳辐射强度变化规律，我们可以得到相应的最大辐射强度。下面是根据计算结果绘制的三种不同倾角的斜面每月 15 日辐射强度随时间变化的示意图。



(a) 20° 倾角　　(b) 40° 倾角



(c) 60° 倾角

图 5: 20° 倾角斜面每月 15 日辐射强度随时间变化示意图

可以由图直观地看到三种倾角对应的每日最大辐射强度均出现在一天中午左右的时刻，且随着斜面倾角的增大斜面所接受的辐射强度减小 (在早晨和傍晚则略有增加)，这与生活实际情况相符。

5.1.3 问题一的解答

根据计算的结果，我们得到面积为 $1\mathrm{m}^2$ 的光伏板每月 15 日最大辐射强度为：

	1 月 15 日	2 月 15 日	3 月 15 日	4 月 15 日	5 月 15 日	6 月 15 日
20	762.9534154	780.020176	765.931	751.2645	746.4988	716.7908
40	737.5593119	704.460841	663.977	649.4217	672.2648	690.4788
60	623.2046703	543.933132	481.9376	469.249	516.9458	580.8849
	7 月 15 日	8 月 15 日	9 月 15 日	10 月 15 日	11 月 15 日	12 月 15 日
20	646.6947661	551.590445	482.7924	488.0374	570.6356	680.4197
40	669.4852955	611.831965	560.4816	567.2448	634.8158	708.5982
60	611.5260177	598.277519	570.5683	578.0341	622.4278	651.3092

表 1: 三种倾角对应的每月 15 日最大辐射强度　单位：W

每日太阳辐射总能量即为上述曲线下方与坐标轴围成的面积，也即对时间的积分

$$\int I_r(t,m,n)\mathrm{d}t$$

但由于函数关系较为复杂，我们考虑使用离散型数值积分方法进行近似计算。

对具有 $N+1$ 个均匀分布的点的积分，使用梯形法计算的近似值为

$$\begin{aligned}\int_a^b f(x)\mathrm{d}x &\approx \frac{b-a}{2N}\sum_{n=1}^{N}\left(f(x_n)+f(x_{n+1})\right)\\ &= \frac{b-a}{2N}\left[f\left(x_1\right)+2f\left(x_2\right)+\ldots+2f\left(x_N\right)+f\left(x_{N+1}\right)\right]\end{aligned}$$

其中，各点之间的间距等于 $\frac{b-a}{N}$。

由此计算方法我们得到面积为 $1\mathrm{m}^2$ 的光伏板每月 15 日辐射强度总量为：

	1 月 15 日	2 月 15 日	3 月 15 日	4 月 15 日	5 月 15 日	6 月 15 日
20°	5053.299912	5241.78224	5186.61	5089.829	5019.329	4751.404
40°	4821.954967	4559.61507	4267.496	4171.781	4349.225	4512.04
60°	4010.659035	3344.07492	2867.081	2784.787	3171.164	3730.386
	7 月 15 日	8 月 15 日	9 月 15 日	10 月 15 日	11 月 15 日	12 月 15 日
20°	4213.067841	3532.08557	3059.155	3091.582	3651.474	4426.191
40°	4413.273515	4066.58758	3751.001	3797.045	4221.015	4674.292
60°	4082.029595	4117.30789	4003.478	4057.966	4288.788	4359.857

表 2: 三种倾角对应的每月 15 日辐射强度总量　　单位：W·h

5.2　问题二模型的建立与求解

5.2.1　倾斜面上的太阳辐射强度变化分析

对于问题二，我们可以继续使用问题一中建立的太阳辐射衰减模型计算任意一天特定时刻衰减后的太阳辐射强度，并使用倾斜面上的辐射模型计算不同朝向及倾角的倾斜面所接受的辐射强度。例如，我们计算求得了在方位角 0°、倾斜角 20° 情况下，8 月 15 日 Ra 随时间变化的数据。

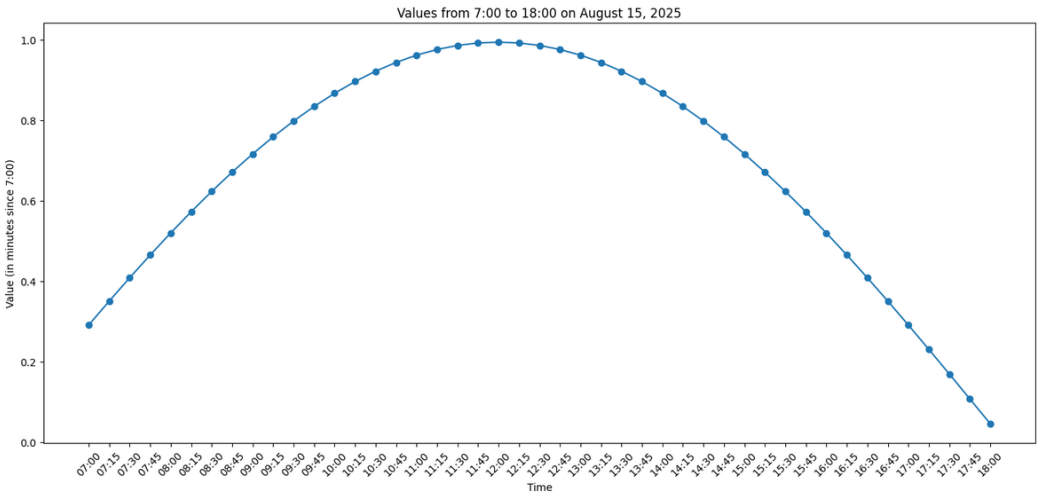


图 6: 在方位角 0°、倾斜角 20° 情况下，8 月 15 日 Ra 变化图

为了确定光伏板在晴天条件下受到的太阳直射辐射日均总能量最大时对应的倾角及朝向，我们考虑使用算法随机选取两个角度，并由此计算每天所接受到的辐射总量，

求得对应的年辐射总量。我们的目标是使年辐射总量尽可能大，以此来确定问题二的答案。为此我们考虑使用遗传算法求解。

5.2.2 遗传算法的构建

遗传算法（Genetic Algorithm,GA）是一种模拟自然选择过程中的生物进化来解决优化问题的算法。在这种算法中，候选解被看作是“个体”，并被编码为一串“基因”来形成“染色体”。遗传算法通过迭代的方式应用“选择”，“交叉”（杂交），和“变异”等遗传操作来生成新的种群，这些新种群的适应度会逐渐向最优解靠近。下面是遗传算法的大致流程图：

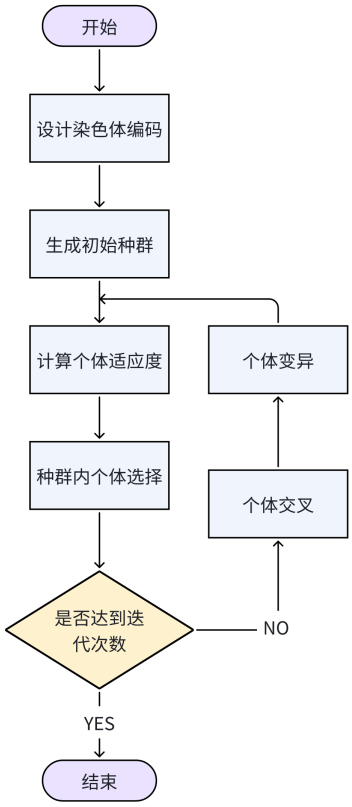


图 7: 遗传算法流程图

将优化对象光伏板方位角 x、水平倾角 y 转化为二进制表示并分别放在奇数位和偶数位来进行 DNA 编码为染色体串。太阳方位角范围约束在 $[-90°, 90°]$，水平倾角范围约束在 $[0, 90°]$.

步骤 1 将编码长度设为 24，种群数量设为 10000，交叉概率为 0.8，变异概率为 0.005，迭代次数为 100，光伏板方位角范围为-90° 至 90°，水平倾角为 0° 至 90°。随机生成一个初始种群，包含多个个体。每个个体都是一个染色体串，即一个编码了

x 和 y 的二进制字符串。

例如，我们随机生成一个种群的个体，则每一列代表一个个体的染色体串，初始种群共生成 10000 行数据，每一行的数据中，共有 48 列 01 字符，奇数列字符组合表示 x, 偶数列字符组合表示 y。

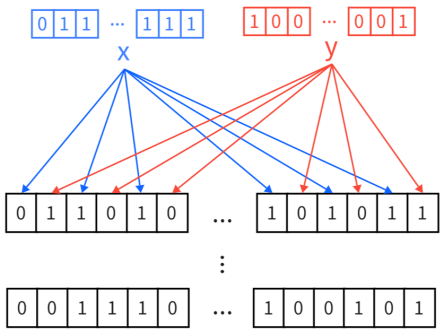


图 8: 遗传算法编码过程

步骤 2 计算种群个体的给定的 x 和 y 组合在一年中可以接收到的总太阳直射辐射强度来评估种群个体适应度，高适应度意味着该个体的 x 和 y 组合能够接收更多的太阳能。个体适应度评估函数即平均每日太阳辐射总能量如下：

$$F = \frac{1}{365} \sum_{n=1}^{365} \int I_r(t, m, n) \mathrm{d}t$$

步骤 3 对种群进行轮盘赌选择法根据个体的适应度来选择将要进行交叉和变异的个体，适应度较高的个体被选中的概率更大。随机选择两个个体作为父母，根据交叉率来决定是否进行交叉操作。如果进行交叉，则从父母的染色体中随机选择一个点，交换这个点之后的基因。对于每个新产生的个体，根据变异率随机改变其染色体中的某些基因（即染色体上的某个位点从 0 变为 1，或从 1 变为 0）。重复选择、交叉和变异操作，得到新一代种群。

步骤 4 判断是否满足迭代次数条件。若是，则解码最优染色体 DNA 编码，输出最优解；否则循环步骤 3 至步骤 4，直到达到迭代次数。

5.2.3 模拟退火算法的构建

模拟退火算法（Simulated Annealing, SA）是受物理中退火过程启发的一种随机搜索方法，用于求解全局优化问题。这种算法通过模拟物质加热后再慢慢冷却的过

程，在高温时系统具有较高的能量状态（能够跨越能量壁垒），随着温度的降低，系统冷却并逐渐稳定在最低能量状态。模拟退火算法在优化中利用这一机制来逃离局部最优，并最终找到一个全局最优解。

算法流程

模拟退火过程中，每一个 “状态” 对应一对光伏板的方位角 x 和倾斜角 y。类似物理中的能量，但在优化中通常指成本或损失，这里我们用负的太阳辐射强度表示：使用 $-F(x,y)$ 作为能量函数，其中 $F(x,y)$ 为计算给定方位角和倾斜角下的年太阳辐射强度总和的函数。目标是最大化 $F(x,y)$，相当于最小化 $-F(x,y)$。

步骤 1 设置较高的初始温度 $T_0 = 100$, 允许算法在开始时能够接受较差的移动，以增加搜索空间的覆盖率. 终止温度 Tf 设为 0，内循环迭代次数 L 设为 80，随机选择一个初始状态，产生一个初始解 $(x_0\ y_0)$, 并计算对应的目标函数值 $F(x_0\ y_0)$。令 $T = \alpha T$, 其中 α 设为 0.97，为温度下降速率。

步骤 2 对当前解 $(x_t\ y_t)$ 施加随机扰动，在其邻域内产生一个新解 (x_{t+1}, y_{t+1})，并计算对应的评估函数值 $F(x_{t+1}, y_{t+1})$, 计算

$$\Delta F = F(x_{t+1}, y_{t+1}) - F(x_t, y_t)$$

若 $\Delta F < 0$, 即新状态的能量低于当前状态，则接受新状态。接受新解作为当前解，否则按照概率

$$P = \exp(-\frac{\Delta F}{kT}), \quad \Delta F \geq 0$$

判断是否接受新解；

步骤 3 在温度 T 下，重复 L=80 次扰动和接受过程，并更新当前温度的最优解；

步骤 4 重复步骤 2 和步骤 3 的过程继续降温，直到降低到终止温度，算出最优光伏板方位角 x 和倾斜角 y 的组合

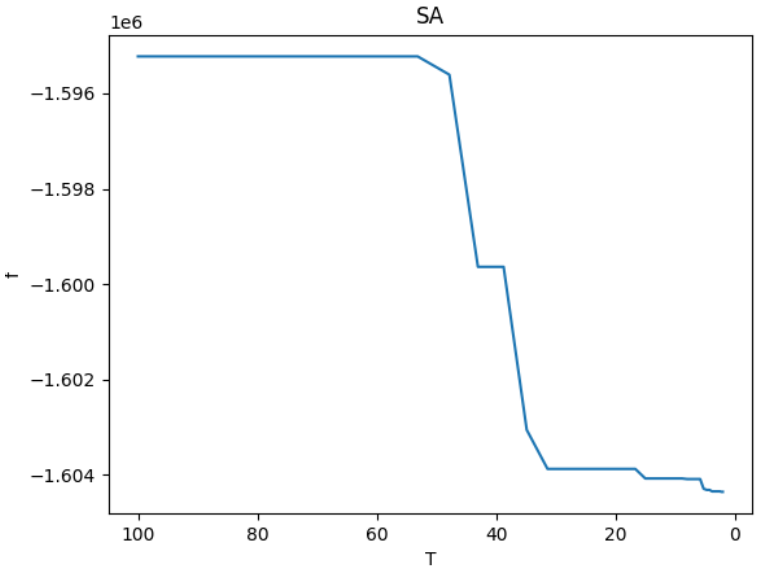
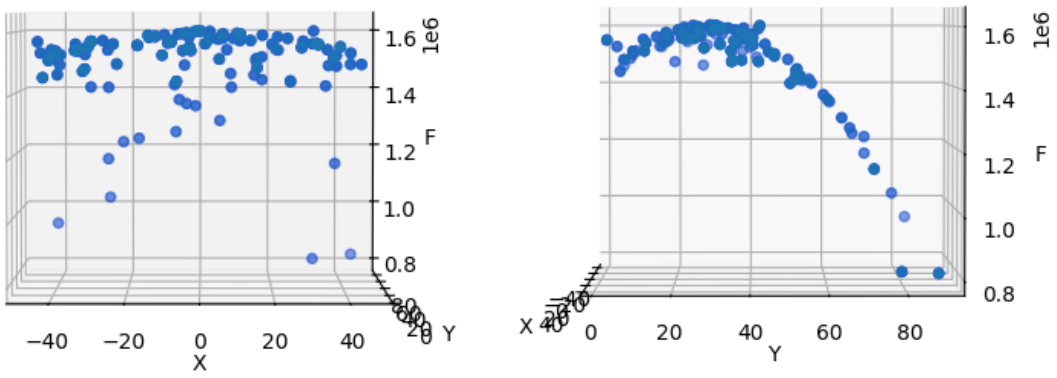


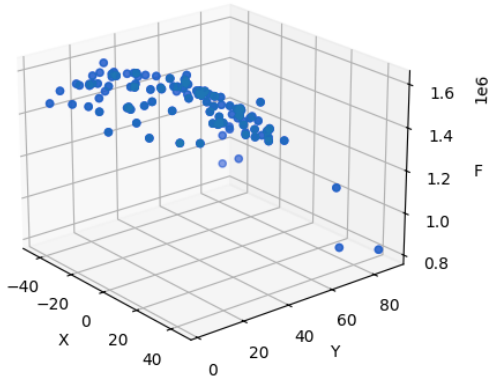
图 9: 模拟退火算法的求解过程

5.2.4 问题二的解答

使用遗传算法优化光伏板的方位角和水平斜角后得到了以方位角为 x 轴，水平倾角为 y 轴，适应性为 z 轴的三维图像，图像的 xOz 平面视图、yOz 平面视图和三维视图如图10所示，可以看出遗传算法的优化过程中，在水平倾角近似的情况下，数据的坡度并不明显，说明方位角对个体适应度的影响并不显著。



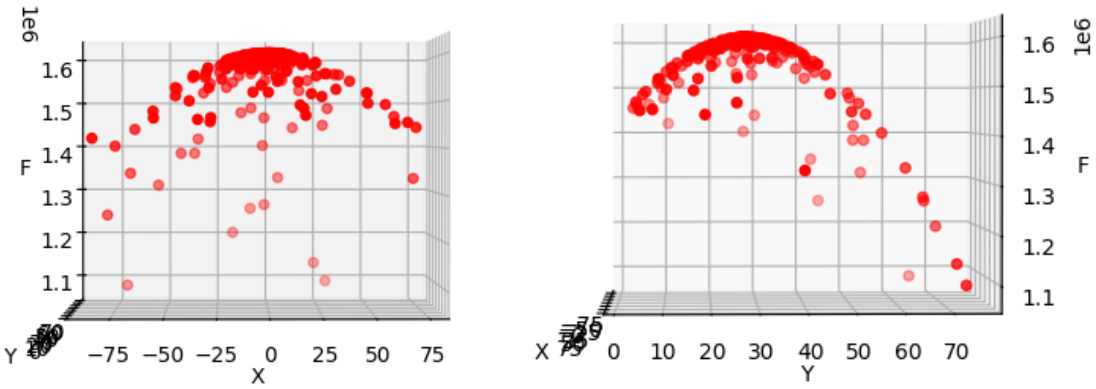
(a) x 方向　　(b) y 方向



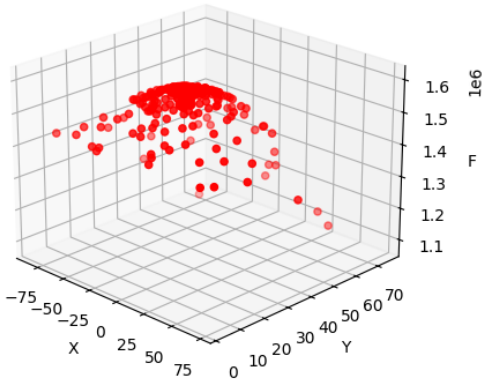
(c) 总图

图 10: 使用遗传算法求解结果示意图

图11为使用模拟退火算法优化光伏板的方位角和水平倾角后得到的数据三维视图，相较于在遗传算法中方位角对个体适应度的影响并不明显，在模拟退火的优化过程中，在水平倾角近似的情况下，数据的坡度更大，方位角对个体适应度的影响比较明显。



(a) x 方向　　(b) y 方向



(c) 总图

图 11: 使用模拟退火算法求解结果示意图

通过图12展示，我们发现模拟退火算法所达到的峰值明显高于遗传算法所得到的峰值。这一观察结果表明，在针对太阳直射辐射日均总能量最大问题的全局最优解搜索中，模拟退火算法可能展现出了更高的搜索效率或更强的寻优能力。

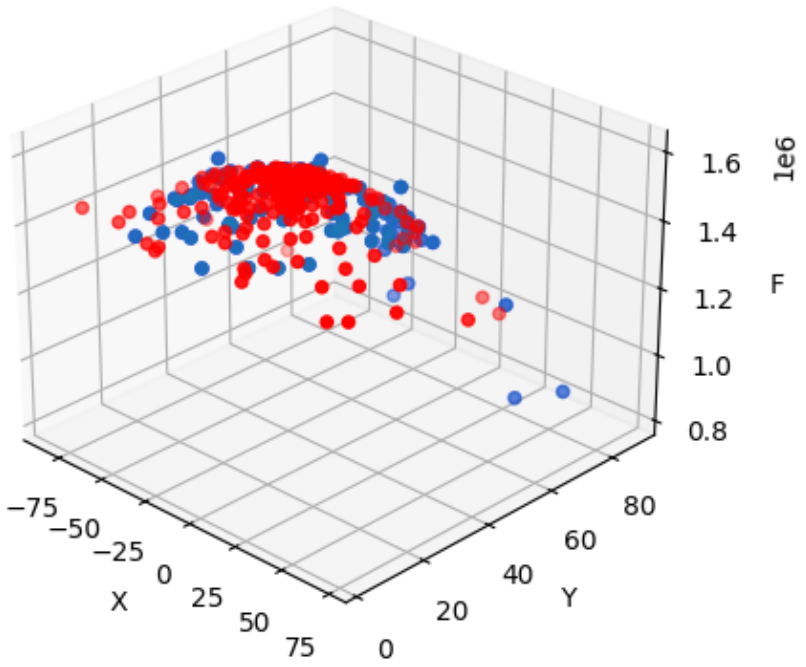


图 12: 两种算法求解结果的对比图

综上，我们选用模拟退火算法作为我们的优化模型，其运行结果如表3，我们在多次优化后对结果取平均值，得到最佳朝向组合为 **(-0.2358, 27.8555)**。

最佳方位角	水平倾角
-0.48612	26.8346
-0.2479	27.9192
0.3452	29.8145
-0.5544	26.8537

表 3: 模拟退火算法优化结果

5.3 问题三模型的建立与求解

非支配排序遗传算法 NSGA (Non-dominated Sorting Genetic Algorithms) 是由 Srinivas 和 Deb 于 1995 年提出的。该算法是在基本遗传算法的基础上，对选择再生方法进行改进: 将每个个体按照它们的支配与非支配关系进行分层，再做选择操作，从而使得该算法在多目标优化方面得到非常满意的结果。但是该算法非支配排序时间复杂度太高，且缺少精英保留策略，为此我们使用 NSGA 的改进算法：基于精英策略的多目标非支配排序遗传算法 NSGA-II，下面是 NSGA-II 算法的大致流程图：

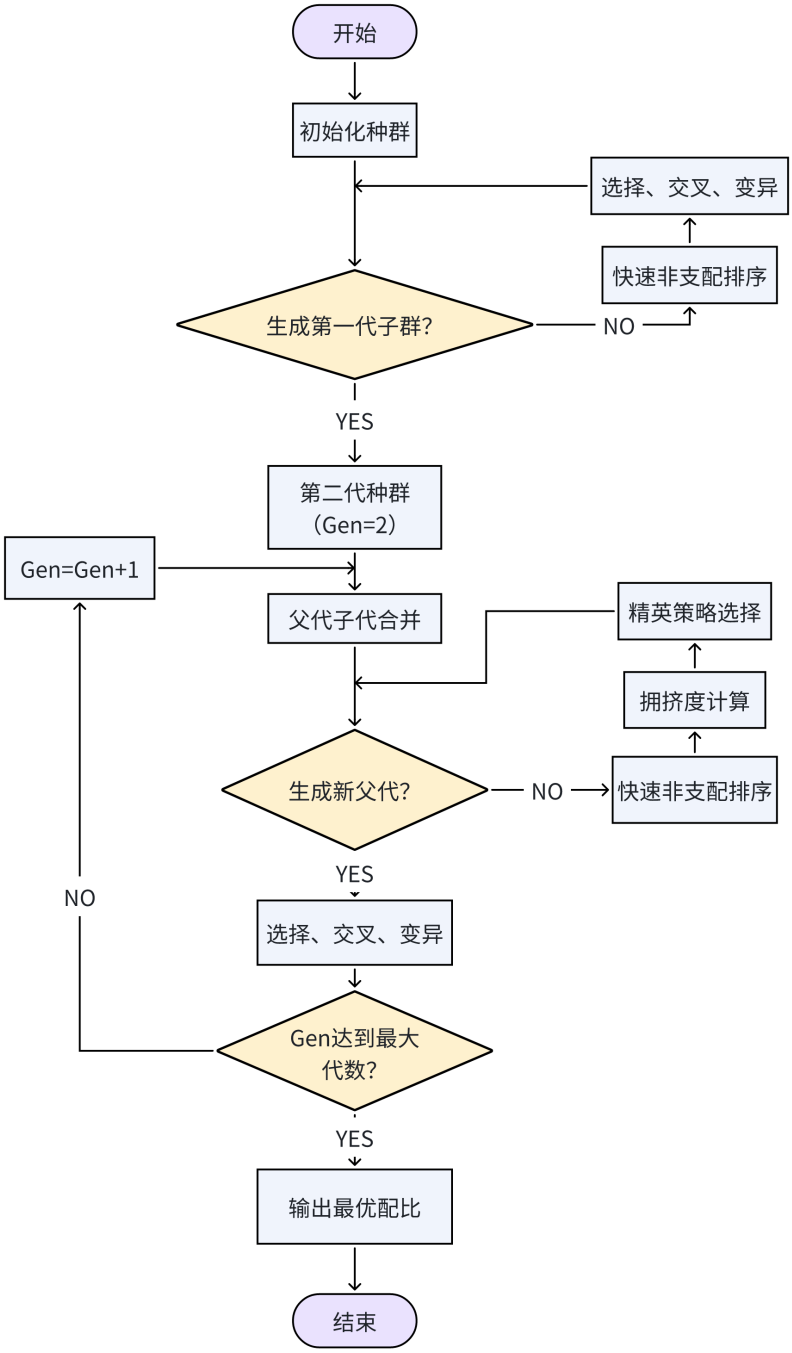


图 13: NSGA-II 算法的流程图

在本题中，假设任何二解 $(x1, y1)$ 及 $(x2, y2)$ 对所有目标而言，$(x1, y1)$ 均优于 $(x2, y2)$，则我们称 $(x1, y1)$ 支配 $(x2, y2)$；若 $(x1, y1)$ 的解没有被其他解所支配，则 $(x1, y1)$ 称为非支配解，也称 Pareto 解。这些非支配解的集合即所谓的帕累托前沿 (Pareto front)。非支配排序就是将种群分成多个不同水平的帕累托前沿。

步骤 1 随机产生规模为 N 的初始种群 P_t，进行快速非支配排序:

1）对于每个个体都设有以下两个参数 $n(x, y)$ 和 $s(x, y)$。$n(x, y)$ 指在种群中支

配个体 (x,y) 的解个体的数量，$s(x,y)$ 指支配 (x,y) 的解个体的集合。

2）对于所有 $n(x,y)=0$ 的个体，将它们存入当前集合 $\mathcal{F}(1)$，考察支配它的个体集合 $s(x,y)$，将 $s(x,y)$ 中的每个个体 (x',y') 的 $n(x',y')$ 减去 1。

3）标记 $\mathcal{F}(1)$ 为第一级非支配个体集合，并赋予该集合内个体一个相同的非支配序 $\mathrm{rank}(x,y)$，继续作上述分级操作并赋予相应的非支配序，直到所有的个体都被分级。

步骤 2 对每个非支配层中的个体进行拥挤距离计算：我们沿着对应目标计算这个解两侧的两点间的平均距离，这个值 $dist(x,y)$ 是对用最近的邻居作为顶点形成的长方体的周长的估计。某个解的拥挤距离越小，这个解被其他解拥挤的程度越高。接下来通过拥挤度比较算子来选择解实现更广的帕累托最优解分布：对于不同非支配等级的两个解，倾向于选择 rank 值更低的解，如果两个解的等级相同，倾向于选择拥挤距离更大或者说拥挤区域更小的解。

步骤 3 依据路灯蓄电池的储电效率和储电量这两个评估函数，通过选择、交叉和变异产生新的子代种群 Q_{t+1}，注意此时选择要依据选择标准是基于拥挤度比较算子。采用精英策略防止优秀个体的流失，将父代和子代所有个体合并为新的种群 R_t 后进行非支配排序，重复以上操作，直到满足程序结束的条件。

5.3.1 非支配排序遗传算法的构建

根据问题背景，储电效率与光伏板受到太阳直射强度上午大于 $150\mathrm{W/m^2}$、下午大于 $100\mathrm{W/m^2}$ 的时间 tr 呈正相关关系，储电量同光伏板受到的太阳直射辐射日均总能量呈正相关关系。即有储电效率 $\eta \propto t_r$，储电量 $E_s \propto I_r$ 且有

$$I_r = t_r \cdot \int I_r(t,n)\mathrm{d}t$$

我们首先对依据上文所述模型计算得到的数据进行**归一化处理**:

目的 本问中储电效率和储电量的值量纲差距较大，带入算法后可能会导致分析时的重心偏移，即易忽略较小数据的变化影响，从而降低最终预测结果的精准程度。为了消除这些不同单位或量纲对算法的影响，使用 Z-score 归一化将原始数据转换为均值为 0、标准差为 1 的分布。

过程 分别计算 t_r 和 $I_r\eta$ 的均值和标准差 $\mu_1,\sigma_1;\mu_2,\sigma_2$，计算公式为：

$$\mu_1 = \frac{1}{365}\sum_{n=1}^{365} t_r(n)$$

$$\sigma_1 = \sqrt{\frac{1}{365}\sum_{n=1}^{365}(t_r(n) - \mu_1)^2}$$

类似地，可计算 $I_r\eta$ 的均值和标准差。使用 Z-score 得到归一化数据，有

$$X' = \frac{X - \mu}{\sigma}$$

然后我们采用评估函数

$$F_1 = (t_r - \mu_1)/\sigma_1$$

$$F_2 = (I_r - \mu_2)/\sigma 2$$

综上所述，我们建立的双目标优化模型为：

$$\max \quad F_1, F_2 \tag{14}$$

$$s.t. \quad \begin{cases} -90^\circ \leq \quad x \leq 90^\circ \\ 0 \leq \quad y \leq 90^\circ \end{cases} \tag{15}$$

权重系数的计算：

对太阳辐射日均总能量和太阳直射辐射时长权重系数的取值我们采取了 CRITIC 权重法。CRITIC 是一种比较熵权法和标准离差法更好的客观赋权法。它基于评价指标的对比强度和指标之间的冲突性来综合衡量指标的客观权重。考虑指标变异性大小的同时兼顾指标之间的相关性，利用数据自身的客观属性进行科学评价赋权。

- **对比强度**是指同一个指标各个评价方案之间取值差距的大小，以标准差的形式来表现。标准差越大，说明波动越大，即各方案之间的取值差距越大，权重会越高；
- **指标之间的冲突性**，用相关系数进行表示，若两个指标之间具有较强的正相关，说明其冲突性越小，权重会越低。

5.3.2 问题三的解答

结合 2023 年 5 月 23 日的太阳辐射数据与问题二中算出的最佳太阳能光伏板朝向，使用 CRITIC 权重法确定出晴天条件下光伏板受到的太阳辐射日均总能量的权重系数 β 为 0.7855，太阳直射辐射时长的权重系数 γ 为 0.2145.

通过 NSGA 模型的计算，我们得到了最优解空间及前沿面部分集, 具体数据如下表：

方位角 x	倾角 y	太阳直射时长	辐射日均总能量
1.451	28.231	9.286	4396.233
0.044	31.322	9.326	4388.763
-0.054	31.322	9.326	4388.763
0.281	30.314	9.312	4392.563
-0.369	31.059	9.325	4389.865
0.776	28.357	9.283	4396.319
1.392	29.156	9.298	4395.296
0.866	29.606	9.303	4394.498
-0.168	30.810	9.320	4390.835
0.018	30.553	9.316	4391.741

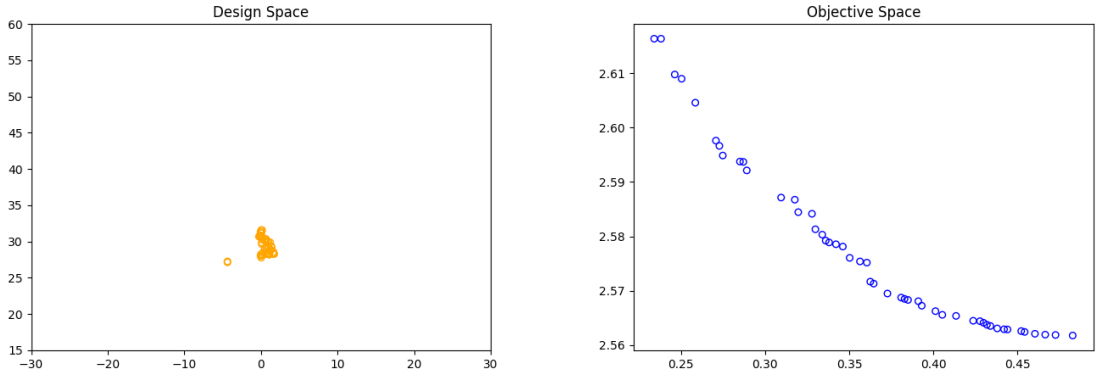
表 4: 问题三最优解空间及前沿面部分集

由此我们绘制得到了图14，左图为优化后得到的 Pareto 最优解集，表明优化后的最优解均集中分布于点（0,30）周围，而右图为优化后的 Pareto 前沿面。

结合使用 CRITIC 权重法得出的权重系数，带入公式

$$F_a = I_r\beta + t_r\gamma$$

我们计算出最优化方位角和水平倾角组合为**（0.84040338，28.26848105）**，晴天条件下光伏板受到的太阳辐射日均总能量为 **4396.54382819KJ**，太阳直射辐射时长为 **9.27351598h**。



(a) 1　　(b) 2

图 14: NSGA 模型优化结果

6 模型的分析、评价与推广

6.1 模型优点

一、我们通过分析太阳光线变化过程中的几何关系，建立了太阳辐射衰减模型；通过分析太阳光线与倾斜面的关系，建立了倾斜面上的辐射模型。这两个模型基于数学和物理实际，具有准确性。

二、此问根据题目中给出“如果光伏板受到的太阳直射辐射总能量最大时，可使路灯蓄电池储电量最大”的这一条件。我们采用遗传算法和模拟退火算法来有效地求解本问题，这两种算法的通用性、鲁棒性强，尤其是模拟退火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，能够有效避免陷入局部极小并最终趋于全局最优，因此可以很好地求解本问题。

三、此问不仅要求储电效率高，而且要求了储电量大。针对该多目标多变量优化问题，我们采用了 NSGA-II 算法。NSGA-II 算法在处理多目标优化问题时表现出色，它通过引入快速非支配排序和拥挤度比较，降低了计算复杂度，提高了算法的收敛速度。同时，算法中的精英保留策略保证了优秀个体能够传递到下一代，从而加速收敛过程。

6.2 模型缺点

一、遗传算法和模拟退火算法对初始值和参数的选择非常敏感。不恰当的参数设置可能导致算法陷入局部最优解或收敛速度过慢，需要我们对参数进行调优。但基于算法参数的多样性和复杂性，寻找最优参数是一个较为复杂的问题。

二、我们通过 NSGA-II 算法得出了一组解，它们组成了 Pareto 前沿面，但对于对于两个目标需要进行权重分析。我们使用了 CRITIC 权重法，CRITIC 权重法对于数据的稳定性和指标之间的关联关系有一定的要求，因此需要我们对数据进行充分的预处理和分析。基于数据关联的复杂性，求解最优权重是一个困难的问题。

6.3 模型推广

本文为解决太阳能光伏板朝向问题，巧妙运用遗传算法、模拟退火算法和 NSGA-II 算法建立了优化模型，在解决难以用精确算法计算的目标优化问题方面有较好的前景，为光伏板朝向设置提供了合理的方案。通过改变输入数据中的地理位置信息，模型还可以应用于不同地点的光伏板安装优化问题。

参考文献

[1] 梅晓妍, 王民权, 邹琴梅, 等. 任意朝向的光伏电池板最佳安装倾角的研究 [J]. 电源技术,2014,38(04):687-690+733.

[2] Liu Yifan,YanChunping,NiHengxin,et al.Multi-objective optimization decision of high-speed dry cutting gear hobbing pro cess parameters based on GABP and improved NSGA- [J].

[3] CHIN C S, BABU A, MCBRIDE W. Design, modeling and testing of a standalone single axis active solar tracker using MATLAB / Simulink[J]. Renewable Energy, 201 l, 36: 3075 · 3090.

[4] 朱超群, 廖静明. 我国最佳倾角的计算及其变化 [J]. 太阳能学报, 1992, 13(1): 38–44

附录

表 5: 利用太阳辐射衰减模型得到的 5 月 23 日数据与原始数据对比表

北京时间	所给数据的辐射量 W/m2	太阳辐射穿越的大气厚度 (km)	衰减系数	计算得到的辐射量 W/m2
6:00	21.132	4274.704849	0.000969692	40.36
6:30	64.906	3445.473165	0.000877382	85.55
7:00	155.472	2789.588143	0.000770534	179.65
7:30	265.660	2294.759387	0.00070322	302.59
8:00	372.830	1930.56696	0.000660332	410.96
8:30	499.623	1664.292405	0.000590091	530.16
9:00	587.170	1469.091549	0.000558592	617.56
9:30	643.019	1325.445124	0.00055058	659.66
10:00	688.302	1219.888168	0.000542435	692.19
10:30	727.547	1143.334864	0.000530255	716.28
11:00	736.604	1089.739502	0.000544981	732.86
11:30	747.170	1055.178125	0.000549333	742.55
12:00	750.189	1037.26495	0.000554933	745.74
12:30	754.717	1034.809767	0.000550434	742.55
13:00	762.264	1047.651952	0.000534189	732.86
13:30	744.151	1076.638369	0.000542144	716.28
14:00	733.585	1123.740718	0.000532146	692.19
14:30	698.868	1192.335651	0.000542192	659.66
15:00	653.585	1287.701477	0.000554061	617.56
15:30	605.283	1417.817086	0.000557365	564.63
16:00	546.415	1594.560458	0.000559752	500.83
16:30	464.906	1835.309563	0.000574346	423.16
17:00	377.358	2164.499795	0.000583388	337.03
17:30	271.698	2613.437274	0.000608871	247.84
18:00	125.283	3214.724872	0.00073579	155.81
18:30	48.302	3988.421718	0.000832024	78.78
19:00	10.566	4926.175227	0.000982161	36.59

部分源代码

Listing 1: **问题一.py**

```python
import numpy as np
import math

K_legal = 0.0005624701263657791
I0=1334
b=6371
B=7371
phi=30.35
pi=3.1415926
standard_meridian=120

#计算光线直射穿过大气层长度
def calculate_length(phi, delta, omega, b, B):
    phi_rad = np.radians(phi)
    delta_rad = np.radians(delta)
    omega_rad = np.radians(omega)

    r_prime = b * np.sin(phi_rad - delta_rad)
    r_double_prime = b * np.cos(phi_rad - delta_rad)

    L_1 = np.sqrt(B**2 - r_prime**2)

    r_1 = r_double_prime * np.sin(omega_rad)
    r_2 = r_double_prime * np.cos(omega_rad)

    L_3 = np.sqrt(L_1**2 - r_1**2)

    L_4 = L_3 - r_2
    return L_4

#计算赤纬角
def calculate_declination(local_time):
    day_of_year = local_time.timetuple().tm_yday
    earth_tilt = 23.45
    day_angle = 2 * np.pi * day_of_year / 365

    sin_declination = np.sin(day_angle) * np.sin(np.radians(earth_tilt))
    declination = np.degrees(np.arcsin(sin_declination))

    return declination
```

```
41
42 #计算时角
43 def calculate_hour_angle(local_time):
44
45     time_difference = local_time - local_time.replace(hour=12, minute=0, second=0, microsecond=0)
46     time_difference_in_hours = time_difference.total_seconds() / 3600
47     hour_angle = time_difference_in_hours * 15  # degrees
48     return hour_angle
49
50 #函数计算出日均太阳直射辐射强度比值
51 def Ra(x, y):
52     angle_x = np.radians(x)  # 将角度转换为弧度
53     angle_y = np.radians(y)  # 将角度转换为弧度
54     omega = calculate_hour_angle(local_time)
55     delta=calculate_declination(local_time)
56     phi1=math.radians(phi)
57     delta1=math.radians(delta)
58     omega1=math.radians(omega)
59     alpha=math.asin(math.sin(phi1)*math.sin(delta1)+math.cos(phi1)*math.cos(delta1)*math.cos(
           omega1))
60
61     if local_time.hour<12:
62         A = math.acos((math.sin(delta1) - math.sin(phi1) * math.sin(alpha)) / (math.cos(alpha) *
               math.cos(phi1)))
63     else:
64         A = math.acos((math.sin(delta1) - math.sin(phi1) * math.sin(alpha) ) / (math.cos(alpha)*
               math.cos(phi1)))
65         A=2*pi-A
66
67     Ra=-math.cos(alpha)*math.cos(A)*np.sin(angle_y)*np.cos(angle_x)+math.cos(alpha)*math.sin(A)*np
           .sin(angle_x)*np.sin(angle_y)+np.cos(angle_y)*math.sin(alpha)
68
69     if Ra<0:
70         Ra=0
71     return Ra
```

Listing 2: **问题二.py**

```
1 import math
2 from random import random
3 import matplotlib.pyplot as plt
4 from datetime import datetime, timedelta
5 import numpy as np
6
```

```
K_legal = 0.0005624701263657791
b=6371
B=7371
phi=30.35
pi=3.1415926
standard_meridian=120

#计算光线直射穿过大气层长度
def calculate_length(phi, delta, omega, b, B):
    phi_rad = np.radians(phi)
    delta_rad = np.radians(delta)
    omega_rad = np.radians(omega)

    r_prime = b * np.sin(phi_rad - delta_rad)
    r_double_prime = b * np.cos(phi_rad - delta_rad)

    L_1 = np.sqrt(B**2 - r_prime**2)

    r_1 = r_double_prime * np.sin(omega_rad)
    r_2 = r_double_prime * np.cos(omega_rad)

    L_3 = np.sqrt(L_1**2 - r_1**2)

    L_4 = L_3 - r_2

    return L_4

#计算赤纬角
def calculate_declination(local_time):
    day_of_year = local_time.timetuple().tm_yday
    earth_tilt = 23.45  # Earth's tilt in degrees

    # Day angle in radians
    day_angle = 2 * np.pi * day_of_year / 365

    # Calculating sin of declination
    sin_declination = np.sin(day_angle) * np.sin(np.radians(earth_tilt))

    # Convert sin of declination to declination angle in degrees
    declination = np.degrees(np.arcsin(sin_declination))

    return declination

```

```python
#计算时角
def calculate_hour_angle(local_time):

    time_difference = local_time - local_time.replace(hour=12, minute=0, second=0, microsecond=0)
    time_difference_in_hours = time_difference.total_seconds() / 3600

    hour_angle = time_difference_in_hours * 15  # degrees
    return hour_angle

def func(x,y):
 # 设置起始日期和结束日期
 sum2 = 0
 angle_x = np.radians(x)  # 将角度转换为弧度
 angle_y = np.radians(y)  # 将角度转换为弧度

 start_date = datetime(2024, 1, 1)
 end_date = datetime(2024, 12, 31)
 # 设置每天的起始时间和结束时间
 start_time_of_day = datetime(2024, 1, 1, 6, 00)
 end_time_of_day = datetime(2024, 1, 1, 19, 00)

 # 初始化一个列表来保存每一天的日期
 dates = []
 # 遍历每一天
 local_time = start_date
 while local_time <= end_date:
     dates.append(local_time.date())  # 只保存日期部分
     local_time += timedelta(days=1)
 # 现在，我们遍历每一天，并为每一天生成从早上8点到下午17点的每30分钟的时间点
 for date in dates:
     sum1 = 0
     local_time = datetime.combine(date, start_time_of_day.time())
     # print(local_time)
     while local_time <= datetime.combine(date, end_time_of_day.time()):
         # print(local_time)
         month_to_I0 = {
             1: 1405,
             2: 1394,
             3: 1378,
             4: 1353,
             5: 1334,
             6: 1316,
             7: 1308,
```

```
            8: 1315,
            9: 1330,
            10: 1350,
            11: 1372,
            12: 1392
        }
        omega = calculate_hour_angle(local_time)
        delta = calculate_declination(local_time)
        phi1 = math.radians(phi)
        delta1 = math.radians(delta)
        omega1 = math.radians(omega)
        alpha = math.asin(
            math.sin(phi1) * math.sin(delta1) + math.cos(phi1) * math.cos(delta1) * math.cos(
                omega1))

        if local_time.hour < 12:
            A = math.acos(
                (math.sin(delta1) - math.sin(phi1) * math.sin(alpha)) / (math.cos(alpha) * math.
                    cos(phi1)))
        else:
            domain = (math.sin(delta1) - math.sin(phi1) * math.sin(alpha)) / (math.cos(alpha) *
                math.cos(phi1))
            if domain > 1:
                domain = 1
            if domain < -1:
                domain = -1
            A = math.acos(domain)
            A = 2 * pi - A
        Ra1 = -math.cos(alpha) * math.cos(A) * np.sin(angle_y) * np.cos(angle_x) + math.cos(
            alpha) * math.sin(
            A) * np.sin(angle_x) * np.sin(angle_y) + np.cos(angle_y) * math.sin(alpha)
        if Ra1 < 0:
            Ra1 = 0
        if Ra1 > 1:
            Ra1 = 1
        I0 = month_to_I0.get(local_time.month)
        L = calculate_length(phi, delta, omega, b, B)
        I_values = I0 * np.exp(-K_legal * L) * Ra1
        # print(I_values)
        sum1 += I_values /2
        # 将当前时间增加30分钟
        local_time += timedelta(minutes=30)
    sum2 += sum1
```

```
133  return -sum2/365
134
135  class SA:
136      def __init__(self, func,  iter=10, T0=100, Tf=2, alpha=0.9):
137          self.func = func
138          self. iter =  iter  # 内循环迭代次数,即为L =100
139          self.alpha = alpha  # 降温系数，alpha=0.99
140          self.T0 = T0  # 初始温度T0为100
141          self.Tf = Tf  # 温度终值Tf为0.01
142          self.T = T0  # 当前温度
143          self.x = [random() * 180 - 90 for i in  range( iter)]  # 随机生成100个x的值
144          self.y = [random() * 90  for i in  range( iter)]  # 随机生成100个y的值
145          self.most_best = []
146          self.history = {'f': [], 'T': []}
147
148      def generate_new(self, x, y):  # 扰动产生新解的过程
149          while True:
150              x_new = x + self.T * (random() - random())
151              y_new = y + self.T * (random() - random())
152              if (-90 <= x_new <= 90) & (0 <= y_new <= 90):
153                  break  # 重复得到新解，直到产生的新解满足约束条件
154          return x_new, y_new
155
156      def Metrospolis(self, f, f_new):  # Metropolis准则
157          if f_new <= f:
158              return 1
159          else:
160              p = math.exp((f - f_new) / self.T)
161              if random() < p:
162                  return 1
163              else:
164                  return 0
165      def best(self):  # 获取最优目标函数值
166          f_list = []  # f_list数组保存每次迭代之后的值
167          for i in  range(self. iter):
168              f = self.func(self.x[i], self.y[i])
169              f_list.append(f)
170          f_best =  min(f_list)
171
172          idx = f_list.index(f_best)
173          return f_best, idx  # f_best,idx分别为在该温度下，迭代L次之后目标函数的最优解和最优解的下
                 标
174
```

```
    def run(self):
        count = 0
        # 外循环迭代，当前温度小于终止温度的阈值
        while self.T > self.Tf:
            # 内循环迭代100次
            for i in  range(self. iter):
                f = self.func(self.x[i], self.y[i])  # f为迭代一次后的值
                x_new, y_new = self.generate_new(self.x[i], self.y[i])  # 产生新解
                f_new = self.func(x_new, y_new)  # 产生新值
                if self.Metrospolis(f, f_new):  # 判断是否接受新值
                    self.x[i] = x_new  # 如果接受新值，则把新值的x,y存入x数组和y数组
                    self.y[i] = y_new
            # 迭代L次记录在该温度下最优解
            ft, _ = self.best()
            self.history['f'].append(ft)
            self.history['T'].append(self.T)
            # 温度按照一定的比例下降（冷却）
            self.T = self.T * self.alpha
            count += 1
            # 得到最优解
            print(self.T)
        f_best, idx = self.best()
        print(f"F={f_best}, x={self.x[idx]}, y={self.y[idx]}")

sa = SA(func)
sa.run()

plt.plot(sa.history['T'], sa.history['f'])
plt.title('SA')
plt.xlabel('T')
plt.ylabel('f')
plt.gca().invert_xaxis()
plt.show()
```

Listing 3: **问题三.py**

```
import numpy as np
import math
from datetime import datetime, timedelta
from pymoo.core.problem import ElementwiseProblem
K_legal = 0.0005624701263657791
b=6371
B=7371
phi=30.35
```

```
pi=3.1415926
standard_meridian=120
mean1=9.38167
dev1=0.22401
mean2=4816.70238
dev2=164.0126
def calculate_length(phi, delta, omega, b, B):
    phi_rad = np.radians(phi)
    delta_rad = np.radians(delta)
    omega_rad = np.radians(omega)

    r_prime = b * np.sin(phi_rad - delta_rad)
    r_double_prime = b * np.cos(phi_rad - delta_rad)

    L_1 = np.sqrt(B**2 - r_prime**2)

    r_1 = r_double_prime * np.sin(omega_rad)
    r_2 = r_double_prime * np.cos(omega_rad)

    L_3 = np.sqrt(L_1**2 - r_1**2)

    L_4 = L_3 - r_2
    return L_4

#计算赤纬角
def calculate_declination(local_time):
    day_of_year = local_time.timetuple().tm_yday
    earth_tilt = 23.45
    day_angle = 2 * np.pi * day_of_year / 365
    sin_declination = np.sin(day_angle) * np.sin(np.radians(earth_tilt))
    declination = np.degrees(np.arcsin(sin_declination))
    return declination

#计算时角
def calculate_hour_angle(local_time):

    time_difference = local_time - local_time.replace(hour=12, minute=0, second=0, microsecond=0)
    time_difference_in_hours = time_difference.total_seconds() / 3600

    hour_angle = time_difference_in_hours * 15  # degrees
    return hour_angle

class MyProblem(ElementwiseProblem):
```

```
52              def __init__(self):
53                              super().__init__(
54                                              n_var=2,
55                                              n_obj=2,
56                                              n_ieq_constr=0,
57                                              xl=np.array([-30, 15]),
58                                              xu=np.array([30, 60])#上下限
59                              )
60
61              def _evaluate(self, x1, out, *args, **kwargs):
62                              x=x1[0]
63                              y=x1[1]
64                              sum2 = 0
65                              angle_x = np.radians(x)  # 将角度转换为弧度
66                              angle_y = np.radians(y)  # 将角度转换为弧度
67
68                              start_date = datetime(2024, 1, 1)
69                              end_date = datetime(2024, 12, 31)
70                              # 设置每天的起始时间和结束时间
71                              start_time_of_day = datetime(2024, 1, 1, 6, 00)
72                              end_time_of_day = datetime(2024, 1, 1, 19, 00)
73                              # 初始化一个列表来保存每一天的日期
74                              dates = []
75                              tr=0
76                              # 遍历每一天
77                              local_time = start_date
78                              while local_time <= end_date:
79                                      dates.append(local_time.date())  # 只保存日期部分
80                                      local_time += timedelta(days=1)
81                              # 现在，我们遍历每一天，并为每一天生成从早上8点到下午17点的每30分
                                    钟的时间点
82                              for date in dates:
83                                      sum1=0
84                                      local_time = datetime.combine(date, start_time_of_day.time
                                            ())
85                                      while local_time <=datetime.combine(date, end_time_of_day.
                                            time()):
86                                              month_to_I0 = {
87                                                      1: 1405,
88                                                      2: 1394,
89                                                      3: 1378,
90                                                      4: 1353,
91                                                      5: 1334,
```

```
            6: 1316,
            7: 1308,
            8: 1315,
            9: 1330,
            10: 1350,
            11: 1372,
            12: 1392
    }
    omega = calculate_hour_angle(local_time)
    delta = calculate_declination(local_time)
    phi1 = math.radians(phi)
    delta1 = math.radians(delta)
    omega1 = math.radians(omega)
    alpha = math.asin(
    math.sin(phi1) * math.sin(delta1) + math.cos(phi1)
         * math.cos(delta1) * math.cos(omega1))

    if local_time.hour < 12:
            A = math.acos(
                    (math.sin(delta1) - math.sin(phi1)
                         * math.sin(alpha)) / (math.
                        cos(alpha) * math.cos(phi1)))
    else:
            domain=(math.sin(delta1) - math.sin(phi1)
                * math.sin(alpha)) / (math.cos(alpha)
                * math.cos(phi1))
            if domain>1:
                    domain=1
            if domain<-1:
                    domain=-1
            A = math.acos(domain)
            A = 2 * pi - A
    Ra1 = -math.cos(alpha) * math.cos(A) * np.sin(
        angle_y) * np.cos(angle_x) + math.cos(
            alpha) * math.sin(
            A) * np.sin(angle_x) * np.sin(angle_y) +
                np.cos(angle_y) * math.sin(alpha)
    if Ra1<0:
            Ra1=0
    if Ra1>1:
            Ra1=1
    I0 = month_to_I0.get(local_time.month)
    L = calculate_length(phi, delta, omega, b, B)
```

```
                                                I_values = I0 * np.exp(-K_legal * L) * Ra1
                                                sum1 += I_values/60
                                                if local_time.hour < 12 and I_values > 150:
                                                        tr+= 1
                                                if local_time.hour >= 12 and I_values > 100:
                                                        tr+= 1
                                                local_time += timedelta(minutes=1)
                                        sum2 += sum1
                                print(sum2/365,(tr)/365/60)
                                f2 = -(sum2/365-mean2)/dev2
                                f1 = -((tr)/365/60-mean1)/dev1
                                out["F"] = [f1, f2]
problem = MyProblem()

from pymoo.algorithms.moo.nsga2 import NSGA2
from pymoo.operators.crossover.sbx import SBX
from pymoo.operators.mutation.pm import PM
from pymoo.operators.sampling.rnd import FloatRandomSampling

algorithm = NSGA2(
                pop_size=100,
                n_offsprings=3,
                sampling=FloatRandomSampling(),
                crossover=SBX(prob=0.9, eta=20),
                mutation=PM(eta=10),
                eliminate_duplicates=True
)

from pymoo.termination import get_termination
termination = get_termination("n_gen", 3)

from pymoo.optimize import minimize
res = minimize(
                problem,
                algorithm,
                save_hastory=True,
                verbose=True
)

X = res.X               # 最优解集
F = res.F               # 最优解

import matplotlib.pyplot as plt
```

```
# 获得自变量而区间
xl, xu = problem.bounds()
# 绘制结果
plt.figure(figsize=(7, 5))
# 绘制最优解集的散点图
print(X[:,0])
print(X[:,1])
plt.scatter(X[:,0], X[:,1], s=30, facecolors="none", edgecolors="orange")
# 设置x、y周现实区间
plt.xlim(xl[0], xu[0])
plt.ylim(xl[1], xu[1])
plt.title("Design Space")
plt.show()

plt.figure(figsize=(7, 5))
print((-F[:,0]*dev1+mean1))
print((-F[:,1]*dev2+mean2))
plt.scatter(F[:, 0], F[:, 1], s=30, facecolors='none', edgecolors='blue')
plt.title("Objective Space")
plt.show()
```