所属类别	2024 年“华数杯”全国大学生数学建模竞赛	参赛编号
本科组		CM2404747

基于熵权法-TOPSIS 的目标优化模型在游览路径规划的应用

摘要

本文针对外国游客在中国的短期旅行问题，运用 SPSS、MATLAB 等工具，通过数据预处理、建模分析及启发式算法，为外国游客提供了包括景点评分分析、城市综合评价、个性化游玩路线规划等多方面的解决方案。

针对问题一，先通过 SPSS 软件城市景点数据进行预处理，去除重复数据，并将缺失值和异常值归零处理，并建立描述性统计模型，对数据进行分析。通过 MATLAB 可求解得：352 个城市中所有 35200 个景点评分的最高分为 5 分，全国共有 2392 个景点获评了这个最高评分，获评了这个最高分（BS）景点最多的城市是玉溪，景点个数为 21，前十名城市依次为：玉溪、益阳、大兴安岭、潍坊、烟台、周口、自贡、邢台、保定、万宁。其中，万宁与东营、内江、宁德、庆阳、抚州、武威、沈阳、泸州、雅安并列第十。

针对问题二，在综合评价 352 个城市时，以城市占地面积、空气质量指数、博物馆数量、城市每日的高铁总活动量、年均气温、特色美食综合评分这六个具体数据分别量化指标，通过建立**基于熵权法的 TOPSIS 模型**进行降维和权重分配，计算得到各个城市的综合评价得分并排序，取前 50 个城市作为“最令外国游客向往的 50 个城市”。

针对问题三，对于从广州出发的游客，构建了**基于网络优化的旅行商（TSP）模型**来求解出最优游玩路线，以确保在 144 个小时内尽可能多地游览评分最高的景点，模型中，考虑了总时间限制、城市访问约束、景点开放时间约束、入境城市约束。通过蚁群算法，找到了在有限时间内游览最多最高评分景点的最优路径，路线中游玩时间总共花费了 60 个小时，从广州出发共游玩了 19 个景点，门票和交通共花费了 5499.5 元。

针对问题四，在问题三的基础上，目标变为既要最大化游览城市的数量又要最小化门票和交通的总费用。保持问题三约束条件不变，由于两个目标存在冲突，我们建立了**多目标优化模型**，进一步优化游玩路线。最后通过**$\varepsilon-$约束法**对多目标优问题进行求解，找到了一条使得游客在预算有限的情况下获得最佳的游玩体验的路线，该路线所总的游玩时间为 48.5 小时，从广州出发共游玩了 19 个城市，总花费 2396.5 元。

针对问题五，对于仅对山景感兴趣的外国游客，从 352 个城市中筛选出所有与山景相关的景点与城市，然后，通过综合评价并选出每个城市评分最高的山景景点，再建立**多目标粒子群优化模型**进行路径优化，以规划出既能尽可能多地游览山景，又能花费最少的路线。通过**粒子群算法**等启发式算法求得一条较优的游玩路线。

关键词：基于熵权法的 TOPSIS 模型、蚁群算法、网络优化 TSP 模型、多目标优化模型、粒子群算法、MATLAB、SPSS、短期旅行规划、景点评分分析、个性化路线优化

一、 问题重述

1.1 问题背景

近年来，“city 不 city”这一网络流行语在国际社交媒体上迅速走红，尤其在外国网红的推动下，成为了一个连接中国与世界的新桥梁。随着中国过境免签政策的逐步落实，越来越多的外国游客得以轻松踏入中国，并通过社交媒体等平台分享他们在中国的旅行体验。这一趋势不仅极大地促进了中国旅游业的发展，还通过外国游客的镜头，向世界展示了一个真实、多元且充满活力的中国形象。在此背景下，如何高效、全面地规划旅行路线，以最大化外国游客的游览体验，成为了旅游研究和规划领域的重要课题。根据“每个城市只选择一个评分最高的景点游玩”的“城市最佳景点游览原则”，考虑到游客时间有限（如 144 小时过境停留），如何精准选择城市及景点，确保游客能在有限的时间内尽可能多地领略中国的魅力，同时平衡游玩质量与经济成本，成为了一个亟待解决的挑战。

1.2 问题重述

基于上述背景，我们现拥有包含中国（不含港澳台）352 个城市旅游景点的详尽数据集。每个城市的 csv 文件中记录了该城市内 100 个景点的多方面信息，如名称、地址、评分、游玩时长建议等。请根据附件和实际情况建立相应的数学模型，解决以下问题：

问题一：通过对附件中城市所有景点评分数据进行分析和处理，确定 352 个城市中所有 35200 个景点的最高评分（BS），并统计获评 BS 的景点总数，找出拥有最多 BS 评分的景点的前 10 个城市，按数量进行排序。

问题二：结合城市规模、环境环保、人文底蕴、交通便利性、气候、美食等多方面因素，收集整理相关资料，对 352 个城市进行综合评价，并按评分高低排序选出前 50 名作为“最令外国游客向往的 50 个城市”。

问题三： 在限定时间内（144 小时），为从广州入境的外国游客规划一条游玩体验最佳、尽可能多地游览城市的最优路线。所规划的路线具体需包括总的花费时间、门票和交通的费用以及可以游玩的景点数量，同时，路线需满足“城市最佳景点游览原则”，且城市间交通仅选择高铁，城市必须在“最令外国游客向往的 50 个城市”之中选择。

问题四：在问题三的基础上，保持游玩路线的约束条件不变，修改游览目标为：既要尽可能多地游览城市，又要最小化门票和交通的总费用。请重新调整游玩路线，并给出路线的具体信息，包括总花费时间、门票和交通的费用以及可以游玩的城市数量。

问题五：为只想游览中国山景的外国游客个性化规划一条入境城市不限、尽可能多地游览山景且门票和交通的总费用最低的 144 小时旅游路线。所规划的路线需具体包括游客入境的城市和机场、总花费时间、门票和交通的费用以及可以游玩的景点数量，路线需满足以下要求：遵循每个城市只游玩一座评分最高的山的原则、城市间交

通仅选择高铁、游玩城市拓展到 352 个城市，不局限于“最令外国游客向往的 50 个城市”。

二、 问题分析

2.1 问题一的分析

针对问题一，为了寻找全国最高评分景点及其分布情况，我们首先对 352 个城市的景点评分数据进行试剂库整合与清洗，检查和处理数据的缺失值、异常值和重复值等，确保数据的完整性和一致性；其次，在整合后的数据集中，遍历所有景点的评分字段，使用基本的统计方法（如最大值函数）找出最高的评分值（BS），进一步统计获得 BS 评分的景点总数；最后，绘制柱状图可视化这 352 个城市每个城市所拥有的 BS 景点数量这一数据，再按城市进行分组，用分组统计的方法计算每个城市拥有 BS 景点的数量，并按照数量对城市进行降序排序，列出前 10 个城市及其对应的 BS 评分景点数量。

2.2 问题二的分析

针对问题二，我们首先将城市规模、环境保护、人文底蕴、交通便利、气候、美食这 6 个关键要素进一步细化为可量化的具体指标，如城市占地面积、空气质量指数、博物馆数量、城市每日的高铁总活动量、年均气温、特色美食综合评分等；接着通过官方统计数据、旅游网站、社交媒体等多渠道收集各城市的相关数据，并进行数据整合和数据清洗。其次，由于评价指标数与本身数据量相对较大，为了使得结果更为科学、合理，可采用**基于熵权法的 TOPSIS 模型**对 352 个城市进行综合评价。最后，计算各个城市的相对接近度作为城市综合得分，根据得分大小对城市进行排序，选出前五十名城市作为“最令外国游客向往的 50 个城市”。

2.3 问题三的分析

针对问题三，为了精准地量化各城市的游玩价值，我们依据问题二中求解的“最令外国游客向往的 50 个城市”综合得分，作为城市游玩价值的指标。收集并整理前 50 名城市的之间的高铁运行时间、班次、票价等数据。然后，对数据进行清洗和预处理，去除异常值和重复数据，确保数据的准确性和一致性。由于可能一个城市会存在多个评分最高的景点，则可使用基于熵权法的 TOPSIS 综合评价法评选出游客游玩该城市时游玩的景点。鉴于该优化问题具备直观的图形化表征潜力，我们可利用图论理论和启发式算法相结合的方法来求解。首先将优化问题抽象为一个图模型，建立**基于网络优化的旅行商问题（TSP）模型**，在图中寻找一条满足约束条件的最优路径。最后，可通过蚁群算法等启发式算法进一步求解出最优路径。在确定城市最优路径后，相应选择最佳景点去游玩，并计算行程总花费时间，门票和交通的总费用等，综合上述信息，可形成了一条具体的游玩路线建议。

2.4 问题四的分析

针对问题四，在问题三的基础上，保持所有约束条件不变，增加一个目标，即最小化门票和交通的总费用。由于“最大化游玩城市数量”与“最小化门票和交通的总费用”这两个目标存在冲突，因此可考虑为本题构建一个多目标优化模型。我们首先需要收集并整理各个景点的门票费用和各个城市之间的高铁班次的费用等数据，方便后续的求解。然后将多目标问题的多个目标函数与约束条件列出，再通过**ε – 约束法对多目标模型进行优化求解**。求解时可尝试使用数学规划求解器（如 Gurobi、CPLEX 等）求解转化后的ε – 约束问题。最后，需要对所求得的最优路径结果进行模拟和验证。

2.5 问题五的分析

针对问题五，由于外国游客只想游览中国的山景，每个城市只游玩一座评分最高的山，而且旅游城市范围为 352 个城市，因此我们首先需要考虑从附件中分别提取出每个城市所有评分最高的山景景点数据，再结合景点门票费用、开放时间等因素对该城市这些山景综合评价得出最佳山景景点，最终得到 352 个最佳山景数据集。接着，我们根据最大化游览的山景数量和最小化门票和交通的总费用这两个目标，建立目标函数，并将时间限制、城市访问次数限制、山景景点开放时间约束、访问城市山景次数约束等约束条件表示出来，最后可通过**粒子群算法**等启发式算法求解多目标优化问题。

三、 模型假设

1、假设外国游客能在境内逗留 144 小时，且能从任一城市附近的机场出境
2、假设外国游客在每个城市只选择评分最高的景点游玩
3、假设不考虑政策中的入境人员只能在固定范围内逗留
4、假设旅程不存在其他突发状况
5、假设在外国游客旅游时间内交通不发生拥堵现象且高铁线路不停运
6、 假设不考虑不同时间段的景点票价和高铁票价差异（如早鸟票、折扣票等）。

四、 符号说明

符号	说明
y_i	是否选择游玩城市i
t_i	第i个城市的游玩时间
x_{ij}	是否选择从城市i到城市j的游玩路径
T_{ij}	城市i到城市j的高铁行程时间
R_{ij}	考虑夜间行程时的休息补偿时间
c_{ij}	高铁从城市i到城市j的费用

p_i	游玩城市i最佳景点的门票费用
$p_i^{'}$	游玩城市i最佳山景景点的门票费用
C	所有 352 个城市的集合
D	352 个城市间所有高铁线路的集合

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 数据的分析和处理

首先我们对附件中 352 个城市的景点的评分数据进行数据整合和数据清洗，将所有城市中的各个景点的“评分”数据提取出来并整合为一个统一的数据集，然后检查该数据集中数据是否存在缺失值、异常值、重复值等错误数据，接着，检查数值是否存在类型不匹配问题。通过 spss 软件检查数据，我们发现数据存在缺失值、异常值和较多重复值，因此，我们将缺失值、异常值均替换为 0，**去除重复值**，并将字符串数据转为浮点数类型。最后，我们得到一份合理完整的评分数据集用于描述性统计模型的建立与求解。

5.1.2 模型一的建立

根据附件数据，已知 $n = 352$ 个城市的数据集，每个城市 i 有 $m = 100$ 个景点，其中，每个景点$j\ (j=1, 2, \ldots, m)$在城市i（$i=1, 2, \ldots, n$）中的评分记为S_{ij}。以下是针对问题一建立描述性统计模型的步骤：

Step1. 评分统计:

由题意可知，

$$BS = \max\{S_{ij}\} \tag{1}$$

其中，$i = 1,2,\ldots,352, j = 1,2,\ldots,100$。

为了求出 352 个城市中所有 35200 个景点评分的最高分，可以对评分数据集使用基本的统计方法（如 python 中的 max 函数）得出所有景点的最高评分（BS）。

Step2. 统计获得 BS 评分的景点总数：

我们先定义I_{ij}：

$$I_{ij} = \begin{cases} 1, & S_{ij} = BS \\ 0, & S_{ij} \neq BS \end{cases} \tag{2}$$

使用该指示函数，获得 BS 评分的经典总数 T 为：

$$T = \sum_{i=1}^{352} \sum_{j=1}^{100} I_{ij} \tag{3}$$

可通过 MATLAB 编程实现条件筛选（条件为“评分等于 BS”）并求和的算法，计算出获得 BS 的景点数量。

Step3. 分别确定每个城市中获得 BS 的景点数量:

对每个城市 i，定义C_i为城市 i 中获得 BS 的景点数量：

$$C_i = \sum_{j=1}^{m} I_{ij} \tag{4}$$

接着为了直观展示每个城市中获评 BS 的景点数量的差异，可绘制出各个城市的 BS 景点数量分布图进行总体分析。

Step4. 依据拥有最高评分（BS）景点数量的多少排序，并列出前 10 个城市

首先，我们要对上一步所求得的$C_i(\mathrm{i}=1,2,\ldots,352)$进行降序排序,设 R 为程序排序后的索引，使得$C_{R(1)} \geq C_{R(2)} \geq \ldots \geq C_{R(352)}$,然后观察并记录拥有$C_{R(1)}$个 BS 景点的城市名称。最后按照索引对应的城市，选择出排名前十个城市，即：$\mathrm{TOPTEN} = \{\mathrm{R}(1),\mathrm{R}(2),\ldots,\mathrm{R}(10)\}$。

5.1.3 模型一的求解与结果分析

我们通过 MATLAB，实现对模型一结果的正确求解，求解结果如下：

（1）352 个城市中所有 35200 个景点评分的最高分为 5;

（2）全国共有 2392 个景点获评了这个最高评分；

（3）获评了这个最高分（BS）景点最多的城市是玉溪。然后根据 352 个城市中各个城市拥有最高分（BS）景点的数量大小，作出各个城市的 BS 景点数量分布图，如图 1 所示。

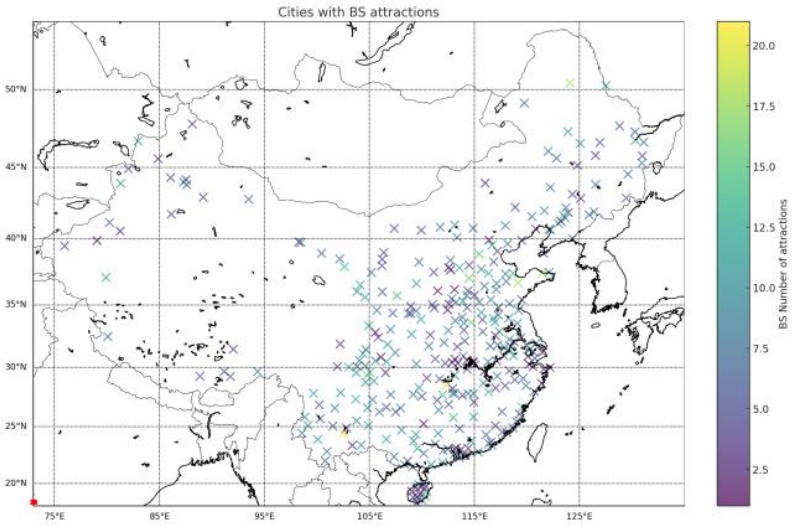

图 1 各城市 BS 景点数量分布图

（4）根据拥有最高评分（BS）的景点数量多少从大到小对城市进行排序后，前十名为：玉溪、益阳、大兴安岭、潍坊、烟台、周口、自贡、邢台、保定、万宁。

其中，万宁与东营、内江、宁德、庆阳、抚州、武威、沈阳、泸州、雅安这些城市并列第十，拥有最高评分（BS）的景点数量均为 14。在图 2 中我们仅展示了万宁作为第十名。

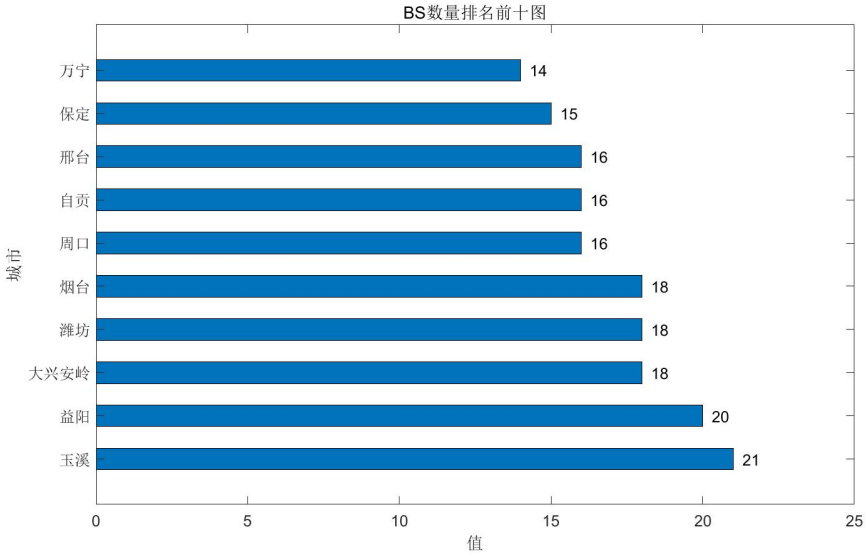

图 2 获得最高评分景点数量最多的前 10 个城市

通过数据分析，我们发现，玉溪和益阳以显著的优势位居前列，这表明它们在保护和提升旅游景点质量方面取得了显著成就，吸引了大量游客并给予高度评价。这两个城市在自然景观、历史文化、旅游设施及服务质量等多方面均表现突出。特别是玉溪市，地理位置独特，以得天独厚的自然风光和丰富的人文历史而著称，素有“滇北明珠”之誉。近年来，玉溪市加快推进旅游景区提档升级，积极开展国家 A 级旅游景区创建工作，新增 6 个国家 A 级旅游景区，截至目前，玉溪市共有国家 A 级旅游景区 39 个[1]。大兴安岭、潍坊、烟台等城市的上榜，进一步证明了自然风光和特色文化资源在吸引游客方面的重要性。这些城市通过充分利用自身独特的自然资源和深厚的文化底蕴，成功打造了具有吸引力的旅游品牌。

5.2 问题二模型的建立与求解

5.2.1 数据的收集和预处理

（1）数据的收集：

首先我们将城市规模、环境保护、人文底蕴、交通便利性、气候及美食六个评价要素量化为城市占地面积、空气质量指数、博物馆数量、城市每日的高铁总活动量、年均气温、特色美食综合评分这六个具体的指标，其中高铁总活动量表示为一天内从该城市出发和到达（或停靠）的高铁数量的加权求和，这里我们认为出发数量的权重α和到达数量的权重β相同，即$\alpha = \beta = 0.5$。接下来我们通过各种渠道收集数据，如政府公开信息、旅游机构报告、社交媒体等，确保数据正确、有效。

（2）数据预处理：

数据整合和数据清洗：我们先检查数据是否有缺失值或异常值。对于缺失值，可以考虑删除对应行或进行插值处理；对于异常值，根据实际情况判断是否需要调整或删除。

5. 2. 2 模型的建立

基于熵权法的 TOPSIS 模型是一种结合了熵权法和 TOPSIS 方法的综合评价技术。该方法通过熵权法客观确定各评价指标的权重，再利用 TOPSIS 方法计算评价对象与理想解之间的接近程度，从而进行排序和评估。由于熵权法是一种使用数据本身来确定权重的方法，它根据各指标数值变化对整体的影响来计算指标的熵值，进而确定权重，具有很强的客观性。因此，采用该方法能对附件中 352 个城市进行更客观、更有效的综合评价。以下是构建基于熵权法的 TOPSIS 综合评价模型的基本步骤：

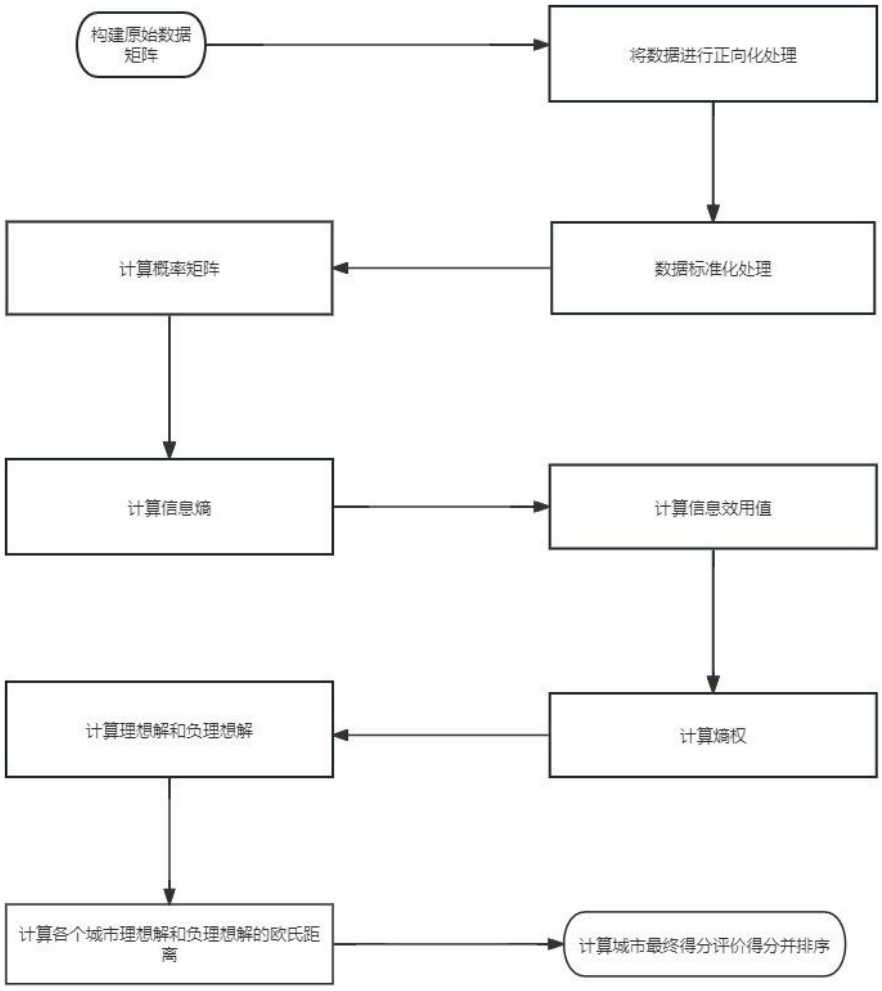

图 3 基于熵权法的 TOPSIS 综合评价法

Step1.构建原始数据矩阵：本题中共有 352 个城市，即 352 个评价对象，6 个评价指标，可形成原始数据矩阵：

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

其中$n = 352, m = 6$。

Step2.将数据进行正向化处理：先判断数据类型，确定数据是否需要进行正向化处理。显然，空气质量指数指数越低越好，说明空气质量这一指标的数据为极小型数据，而年均气温在某个区间范围内最好，气温值越靠近这个区间越好，说明该数据是区间型数据，二者都需进行正向化处理，进一步得到正向化矩阵X。

通过查阅文献可知，当环境温度在 18℃到 25℃，相对湿度在 40%到 70%时，人

体感觉最舒适[1]。因此，针对本题气温数据，我们定义最佳气温区间为[18,25]。
Step3.数据标准化处理：判断输入的正向化矩阵$\overline{\mathrm{X}}$，并对$\overline{\mathrm{X}}$进行标准化，得到标准化后的矩阵 Z。

$$z_{ij} = \frac{\overline{x_{ij}}}{\sqrt{\sum_{i=1}^{n} \overline{x_{ij}}^2}} \tag{5}$$

其中，$i = 1,2,\ldots,n, j = 1,2,\ldots,m$

若 Z 存在负数，则对 X 使用另一种标准化方法：

$$\widetilde{z_{ij}} = \frac{x_{ij} - min\{x_{1j}, x_{2j}, \ldots, x_{nj}\}}{max\{x_{1j}, x_{2j}, \ldots, x_{nj}\} - min\{x_{1j}, x_{2j}, \ldots, x_{nj}\}} \tag{6}$$

Step4.计算概率矩阵 P:分别计算第j项指标下第i个评价对象（城市）的数值比重p_{ij},计算公式如下。

$$p_{ij} = \frac{\widetilde{z_{ij}}}{\sum_{i=1}^{n} \widetilde{z_{ij}}} \tag{7}$$

Step5.计算信息熵：根据信息熵的定义，计算第j项指标的熵值e_j。熵值越大，表示该指标的离散程度越小，对综合评价的影响（即权重）也越小。计算公式为：

$$e_j = -\frac{1}{lnn} \sum_{i=1}^{n} (p_{ij} ln(p_{ij})) \tag{8}$$

其中，k为常数，通常取$k = \frac{1}{lnn}$；当$p_{ij} = 0$时，令$p_{ij} ln(p_{ij}) = 0$。

Step6.计算信息效用值：信息效用值越大，说明该指标在这 6 个指标中的重要程度越高。计算公式为：

$$d_j = 1 - e_j \tag{9}$$

Step7.计算熵权:将信息效用值进行归一化，就可以得到每个指标的权重。权重越大，表示该指标在对城市进行综合评价中的重要性越高。计算公式为：

$$w_j = \frac{d_j}{\sum_{j=1}^{m} d_j} \tag{10}$$

其中，$m = 6, m$为指标个数。

Step8.确定理想解和负理想解：其中，理想解（理想最优解向量）是各指标的最优值，负理想解（理想最劣解向量）是各指标的最小值。

$$Z^+ = [max(z_{11}, z_{21}, \ldots, z_{n1}), max(z_{12}, z_{22}, \ldots, z_{n2}), \ldots, max(z_{1m}, z_{2m}, \ldots, z_{nm})] \tag{11}$$

$$Z^- = [min(z_{11}, z_{21}, \ldots, z_{n1}), min(z_{12}, z_{22}, \ldots, z_{n2}), \ldots, min(z_{1m}, z_{2m}, \ldots, z_{nm})] \quad (12)$$

Step9. 计算各个城市与理想解和负理想解的欧氏距离：结合熵权法所得各指标的权重结果$w_j(j = 1,2,\ldots,m)$，分别计算出每个城市与最大值、最小值的欧氏距离。

定义第$i(i = 1,2,\ldots,n)$个评价对象与最大值的距离D_i^+：

$$D_i^+ = \sqrt{\sum_{j=1}^{m} w_i * (z_j^+ - z_{ij})} \quad (13)$$

定义第$i(i = 1,2,\ldots,n)$个评价对象与最小值的距离D_i^-：

$$D_i^- = \sqrt{\sum_{j=1}^{m} w_i * (z_j^- - z_{ij})} \quad (14)$$

Step10. 计算城市最终得分评价得分并排序：进一步计算出第$i(i = 1,2,\ldots,n)$个城市未归一化的得分：

$$S_i = \frac{D_i^-}{D_i^+ + D_i^-} \quad (15)$$

显然，$0 <= S_i <= 1$,且S_i越大，D_i^+越小，即越接近最大值。将得分归一化，可得各个城市的最终评价得分：

$$\widetilde{S_i} = \frac{S_i}{\sum_{i=1}^{n} S_i} \quad (16)$$

最后，根据得分$\widetilde{S_i}$的大小，我们可对 352 个城市进行排序，并选出前 50 名作为“最令外国游客向往的 50 个城市”。

5.2.3 模型二的求解与结果分析

通过 MATLAB 对模型进行求解，可得求解得到前 50 名城市，其中排名前十的是：北京、上海、广州、成都、巴音郭楞、深圳、重庆、大兴安岭、西安、呼伦贝尔。综合排名前 50 名的城市，如下表所示（按列依次读，每列从上到下升序排名）为：

表 1 排名前 50 的城市

1	北京	11	青岛	21	喀什	31	阿勒泰	41	赤峰
2	上海	12	武汉	22	郑州	32	海口	42	厦门
3	广州	13	巴音郭楞	23	兰州	33	合肥	43	温州
4	成都	14	天津	24	甘孜	34	临沂	44	太原
5	长沙	15	哈尔滨	25	济南	35	大连	45	石家庄
6	深圳	16	杭州	26	哈密	36	林芝	46	贵阳
7	重庆	17	南京	27	淄博	37	南昌	47	珠海

8	大兴安岭	18	昆明	28	洛阳	38	鄂尔多斯	48	苏州
9	西安	19	日喀则	29	福州	39	黄山	49	南宁
10	呼伦贝尔	20	宁波	30	沈阳	40	济宁	50	阿坝

根据国内各个城市旅游业的发展情况，查阅相关资料收集关于全国各个城市旅游的推荐指数，可知以上排名与现实情况比较吻合。

（1）北京、上海、广州、深圳作为中国的四大一线城市，在排名中稳居前列，这符合普遍的经济、文化和政治地位认知。

（2）排名前十的城市中，既有沿海经济发达城市（如上海、广州、深圳），也有内陆重要城市（如成都、重庆、西安）。此结果较好地验证了模型不仅考虑了经济因素，还涵盖了其他如文化、环境、政策等多方面的综合评价。

（3）虽然整体排名反映了城市间的综合实力差异，但相邻排名的城市间差距可能并不显著。例如，福州与沈阳、海口与合肥等城市的排名相近，说明这些城市在综合评估中的表现较为接近。

5.3 问题三模型的建立与求解

5.3.1 数据的收集与预处理

基于问题二收集的数据和求解的结果，结合游客要求我们将只考虑在排名前 50 的城市中规划路线，而且由于假定“每个城市只选择一个评分最高的景点游玩”和交通方式只考虑高铁，因此交通出行时间我们只计算城市间的高铁的运行时间。为此，我们需要先收集并整理 50 个城市之间的高铁运行时间、班次数据并进行数据预处理。而关于城市最佳景点的选择，我们将直接使用附件提供的各个城市的景点评分、门票费用、游览时间数据对景点进行综合评价，选择出最佳景点。

5.3.2 模型三的建立

针对问题三，我们需要考虑在满足外国游客要求和逗留时间限制的情况下，规划出一条最优的游玩路线，以最大化所游玩城市的数量和综合游玩体验。鉴于该优化问题具备直观的图形化表征潜力，我们巧妙地采用图论作为理论基石，将每个城市抽象为图中的一个顶点，而城市间的高铁连接则映射为连接这些顶点的边。由此，原问题自然转化为一个在多重约束条件下，于无向网络图中寻找最优路径的复杂决策过程。为了求解这一复杂的路径优化问题，我们进一步构建了一个**基于网络优化的旅行商（TSP）模型**来求解这个最优路径问题，此模型充分纳入时间限制、交通方式（限定为高铁）、以及最大化城市与景点游览质量的综合考量，但并不要求所规划的路径必须是回路，即不要求最终回到游玩路线的出发地。

由于游客要求只在排名前 50 的城市中游玩，因此我们利用这 50 个城市中所有景点的相关数据，首先选出每个城市所有评分最高景点，接着对这些景点结合门票费用、游玩时长限制、开放时间等数据，确定各指标的权重，并通过**基于熵权法的 TOPSIS 综合评价模型**评选出该城市最适合外国游客游玩的景点。基于这 50 个最佳景点的数据，下面可建立基于网络优化的旅行商（TSP）模型求解出最优路径。

以下是**基于网络优化的旅行商（TSP）模型**的建立步骤：

（1）城市访问约束：需要确保每个城市最多访问一次。

$$\sum_{j\in V} x_{ij} \le 1 \quad (\forall i \in V) \tag{19}$$

$$\sum_{i\in V} x_{ij} \le 1 \quad (\forall j \in V) \tag{20}$$

（2）景点开放时间约束：确保游玩时间在景点开放时间内。设每个城市i的游玩的最佳景点的开放时间是$\mathrm{T_{i.open}}$到$\mathrm{T_{i.close}}$，设游客从城市i到城市j的高铁到达时间为A_{ij}。

（3）城市访问约束：需要确保每个城市最多访问一次。

$$\sum_{j\in V} x_{ij} \le 1 \quad (\forall i \in V) \tag{19}$$

$$\sum_{i\in V} x_{ij} \le 1 \quad (\forall j \in V) \tag{20}$$

（4）城市访问约束：需要确保每个城市最多访问一次。

$$\sum_{j\in V} x_{ij} \le 1 \quad (\forall i \in V) \tag{19}$$

$$\sum_{i\in V} x_{ij} \le 1 \quad (\forall j \in V) \tag{20}$$

（5）景点开放时间约束：确保游玩时间在景点开放时间内。设每个城市i的游玩的最佳景点的开放时间是$\mathrm{T_{i.open}}$到$\mathrm{T_{i.close}}$，设游客从城市i到城市j的高铁到达时间为A_{ij}。

$$A_{ij} + t_i \le T_{i.close} \tag{21}$$

$$A_{ij} + T_0 \ge T_{i.open} \tag{22}$$

$$t_i \cdot y_i \le T_{i.close} - T_{i.open} \tag{23}$$

（6）入境城市约束：由于该名外国游客从广州入境，因此外国游客必定会从广州坐高铁出发前往某地，由问题二结果可知，广州排名第三，因此V集中序号为 3。

$$\sum_{j\in V} x_{3j} = 1 \quad (\forall j \in V) \tag{24}$$

最后，我们可以通过蚁群算法进行模型的求解。蚁群算法（ACO）是一种基于自然蚂蚁寻找食物路径行为的优化算法，它通过模拟蚂蚁在路径上释放信息素（如氨基胺）并根据信息素浓度选择路径的机制，来解决目标优化问题。以下是通过**蚁群算法求解网络优化问题**的具体步骤：

Step1.问题定义与参数初始化

（1）定义问题：明确网络优化问题的具体目标为在多重约束条件下寻找最优的旅游路线。

（2）初始化参数:设蚁群中的蚂蚁数量为 m,城市的数量为 n,城市 i 与城市 j 之间的欧式距离$d_{ij}(t)(i,j=1,2,\ldots,n)$，t 时刻城市 i 和城市 j 之间的路径上信息素浓度为$\tau_{ij}(t)$,信息素权重因子$\alpha=1$，启发函数因子为$\beta=5$，信息素常量为 Q，Q 为常数，信息素挥发因子为ρ。

Step2.蚂蚁路径构建

（1）假设初始时各路径的信息素浓度相同，浓度为常数$\tau_{ij}(0)=\tau_0$。

（2）随机放置蚂蚁：将一定数量的蚂蚁随机放置某个节点上。

（3）路径选择：每只蚂蚁从当前节点出发，根据路径上的信息素浓度和启发式信息（如距离的倒数）计算转移到其他节点的概率，并使用轮盘赌算法选择下一个节点。这个过程重复进行，直到蚂蚁访问完所有节点并回到起点。

设P_{ij}^{k}为 t 时刻第 k 只蚂蚁从城市 i 转移到城市 j 的概率，s 表示暂未访问的城市集

合（$allowed_k$）中的某一个城市，$allowed_k$:第$k(k=1,2,\ldots,m)$只蚂蚁暂未访问的城市集合。则计算概率的公式：

$$P_{ij}^{k}=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\times[\dfrac{1}{d_{ij}(t)}]^{\beta}}{\sum_{s\in allowed_k}[\tau_{is}(t)]^{\alpha}\times[\dfrac{1}{d_{is}(t)}]^{\beta}}, & j\in allowed_k\\ 0, & j\notin allowed_k\end{cases} \tag{25}$$

P_{ij}^{k}值越大，前往城市 j 的概率越大，但并不是绝对会前往概率值最大的城市。然后，计算每只蚂蚁所走路径的适应度(如路径总长度、总成本)。

Step3.更新信息素浓度

根据路径质量更新路径上的信息素。通常，较优路径上的信息素会得到增强，而较差路径上的信息素会逐渐挥发。我们需要将每条路径上的信息素浓度进行修改，更新后的信息素浓度为：

$$\begin{cases}\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\sum_{k=1}^{m}\Delta\tau_{ij}^{k},\ 0<\rho<1\\ \Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{第 k 只蚂蚁曾经到过路径 i 到 j}\\ 0, & \text{其他}\end{cases}\end{cases} \tag{26}$$

其中，L_k为k只蚂蚁走过的总路径长度。接着，求出本轮每只蚂蚁走过的路径的最小值，与上一轮的最优解相比，最小的即为当前最优解。

Step4.迭代求解

重复路径构建、路径评估和信息素更新过程即重复步骤 2 到 4，直到达到最大迭代次数或解的变化满足收敛标准。在迭代过程中记录并更新最优解，算法终止后，输出最优路径或近似最优路径。

5.3.3 模型三的求解和结果分析

通过 MATLAB 实现优化算法，求解出游玩的最优路径后，我们借助 MATLAB 的地图绘制功能 Mapping Toolbox，绘制了详细的游玩路线规划图来直观展示这条最优路径。通过地图，游客可以清晰地看到从起点到终点，乃至沿途重要景点的完整游览路线。如下图所示：

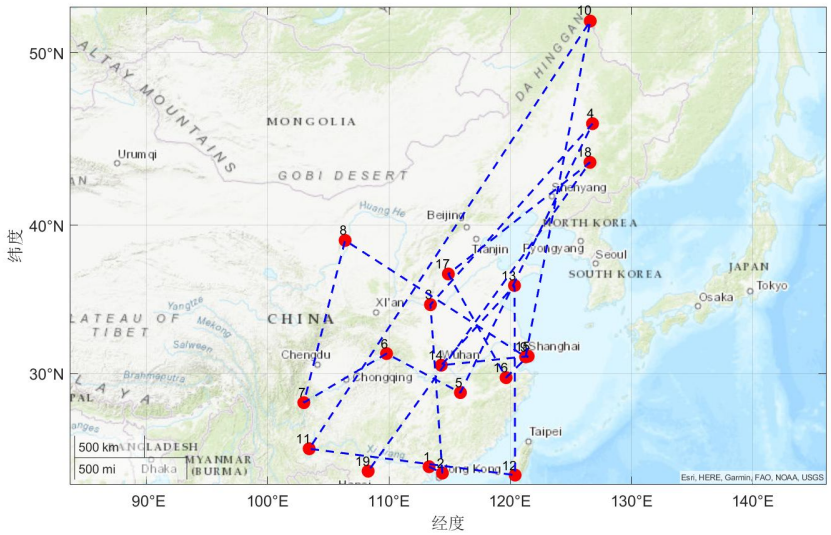

图 4 最优游玩路线图

从图中直观上可以看出，所规划的外国游客的游玩路线地域覆盖广泛，路线覆盖了从南到北、从东到西的多个城市，包括广州、深圳、厦门等，一直到北部的哈尔滨和大兴安岭，以及西部的西安和兰州。这样的设计让游客能够全面体验中国的地理多样性和文化差异。下面是详细的旅游行程表：

表 2 旅游行程表 1

日期	时间	游玩城市	游玩景点	游玩时长（/h）	门票费用（/元）	高铁交通费用（/元）	总交通时间（/h）
	9:30-11:00	广州	熊猫馆	2	35	64	1.5
第一天	13:00-15:30	深圳	金龟村	2.5	94	228	4.5
	20:00-23:00	厦门	乐玩街 Lewan Street	3	100	541	9
	8:00-11:00	长沙	沈家大屋	2	免费	265	2.5
第二天	12:30-15:30	南昌	南昌华侨城	3	100	578	2.5
	18:00-6:00	重庆	兰英大峡谷 Lanyingda	12	免费	164	1.5
	7:30-10:30	成都	俄尔则俄	3	153	252	4.5
第三天	15:00-16:00	北京	长城公园 Changchen Park	1	35	55	1
	17:00-5:00	天津	卢家峪五盆沟 Wupeng	12	免费	210	3

			Canyon				
第四天	8:00-9:00	大兴安岭	鹿鼎山 Humaxian Luding Mountain	1	50	230	4
	13:00-14:00	呼伦贝尔	月亮小镇	1	90	96	2
	16:00-18:00	哈尔滨	香林园	2	100	224	2
	20:00-22:00	青岛	青岛啤酒博物馆 THE WORLD OF TSINGTAO	2	60	658	9
第五天	7:00-8:00	武汉	武汉梨园景区	1	92	315	4
	12:00-14:00	上海	德莱蒙德住宅 Drummond Residence	2	免费	24	1.5
	15:30-18:00	杭州	富春江 Fuchun River	2.5	115	173.5	8
第六天	5:00-7:00	西安	小寨公园	2	免费	93	4
	11:00-13:30	兰州	文庙	2.5	10	236	3
	16:30-20:00	苏州	江南采珠游	3.5	59		

由于游玩路线安排上，苏州是最后一个游玩的城市，外国游客可在苏州体验完特色活动“江南采珠游”后直接出发去杭州机场，乘坐飞机出境，结束旅程。

观察以上行程安排不难看出，每个城市的停留时间都经过了精心规划，既有足够的时间游览主要景点，又避免了行程过于紧凑导致的疲劳。例如，在深圳的金龟村游览后，有充分的时间休息并前往厦门继续下一段旅程（高铁上可以休息 4.5 个小时）。同时，所有的游玩的景点均为拥有最高评分的景点。除了传统的旅游景点如长城公园、沈阳故宫等，路线还包含了特色活动如江南采珠游、青岛啤酒博物馆参观等，增加了行程的趣味性和互动性，让游客能够更深入地了解当地文化和生活方式。整个旅游路线相对经济实惠，部分景点提供了免费或相对较低的票价，如长沙的沈家大屋、上海的德莱蒙德住宅等。

表 3 路线 1 信息统计

游玩城市数量（/个）	19
游玩景点数量（/个）	19
游玩时长（/h）	60
门票和高铁总费用（/元）	5499.5

5.4 问题四模型的建立与求解

5.4.1 数据收集与预处理

结合问题三中所求解的前 50 名的各个城市的最佳景点，首先我们需要收集这些代表性景点的门票价格及开放时间，并查询这些景点所在城市之间的高铁运行时间和票价。接着，检查数据是否有缺失值和异常值，并作出相应的处理。

5.4.2 模型四的建立

针对问题四，我们在保持问题三约束条件不变的情况下，多增加一个目标函数，即最小化门票和交通的费用。但实际情况下，如果要尽可能的游览更多的城市，就会增大门票和交通的花销，因此，这两个目标之间存在着冲突，我们可建立多目标优化模型求解出在本问题条件下的最优路径。

设c_{ij}为通过高铁从城市i到城市j的费用，p_i为游玩城市i最佳景点的门票费用。

可为根据题意，为该多目标优化模型构建以下目标函数：

$$\begin{cases} \text{Maximize} \sum_{\text{i}\epsilon\text{V}} y_i \\ Minimize \sum_{i\epsilon V} (p_i \cdot y_i) + \sum_{(i,j)\in E} (c_{ij} \cdot x_{ij}) \end{cases} \tag{27}$$

保持问题三中总时间约束、城市访问约束、景点开放时间约束、入境城市约束这四个条件不变，接着我们对这个多目标优化模型运用$\varepsilon-$约束法进行求解。

$\varepsilon-$约束法通常涉及在处理多目标优化问题时，将一个或多个目标函数转化为约束条件，并通过引入一个小的正数ε来控制这些约束的松紧程度。以下是$\varepsilon-$约束法的求解步骤：

（1）定义原问题

首先，明确原多目标优化问题中包含多个目标函数和一系列约束条件，可以表示为：

$$\begin{gathered} min\{f_1(x), f_2(x), \ldots, f_n(x)\} \\ \text{st. h(x)} = 0 \\ \text{g(x)} \le 0 \end{gathered} \tag{28}$$

本题中，$f_1(x) = -\sum_{\text{i}\epsilon\text{V}} (W_i \cdot y_i)$，$f_2(x) = \sum_{i\epsilon V} (p_i \cdot y_i) + \sum_{(i,j)\in E} (c_{ij} \cdot x_{ij})$，$\text{h(x)} = \sum_{j\in V} x_{3j} - 1\ (\forall j \in V)$。$\text{h(x)} = 0$ 是等式约束，$\text{g(x)} \le 0$ 是不等式约束，且此不等式约束条件与公式（18）-公式（24）相同。

（2）选取主目标函数：

从多个目标函数中选取一个作为主要目标函数，通常这个选择基于决策者的偏好或问题的实际需求。假设选取$f_1(x)$为主目标函数。

（3） 转化为$\varepsilon-$约束问题

将除主目标函数外的其他目标函数转化为约束条件，并引入ε参数来控制这些约

束的松紧程度。转化后的问题形式为：

$$min\{f_1(x)\}$$

$$\begin{aligned} &\text{st. } f_2(x) \le \varepsilon_2, \ldots, f_n(x) \le \varepsilon_n \\ &\mathrm{h(x)} = 0 \\ &\mathrm{g(x)} \le 0 \end{aligned} \tag{29}$$

其中ε_2需要通过特定方式确定的参数。本题中$\varepsilon-$约束问题表示为：

$$min\{-\sum_{\mathrm{i}\epsilon \mathrm{V}}(W_i \cdot y_i)\}$$

$$\begin{aligned} &\text{st. } \sum_{i\epsilon V}(p_i \cdot y_i) + \sum_{(i,j)\in E}(c_{ij} \cdot x_{ij}) \le \varepsilon_2 \\ &\sum_{j\in V} x_{3j} = 1 \ (\forall j \in V) \\ &\mathrm{g(x)} \le 0 \end{aligned} \tag{30}$$

（4）确定ε值

确定ε值的方法有多种，其中一种常用的是基于 payoff 矩阵的方法。首先，分别求解每个目标函数的单目标优化问题，得到每个目标函数的最优解和对应的最优值。然后，将这些最优解代入其他目标函数，得到 payoff 矩阵。通过 payoff 矩阵，可以计算出每个目标函数的最优值、最劣值以及网格化分数，进而确定每个ε的具体值。

（5）求解$\varepsilon-$约束问题

使用数学规划求解器（如 Gurobi、CPLEX 等）求解转化后的$\varepsilon-$约束问题。求解器将根据给定的目标函数、约束条件和ε值，寻找满足所有约束条件的最优解或近似最优解。

（6）评估解集

评估求解得到的解集，检查是否满足问题的实际需求。如果解集不满足要求，可能需要调整ε值或重新选择主目标函数，并重复上述步骤。

（7） 选择最优解

从解集中选择一个最优解或根据自身的偏好与实际情况选择一个满意的解。

5.4.3 模型四的求解与结果分析

通过 MATLAB 使用$\varepsilon-$约束法进行求解，可得到如下结果：

表 4 旅游行程表 2

日期	出发时间	游玩城市	游玩景点	游玩时长（/h）	门票费用（/元）	高铁交通费用（/元）	总交通时间（/h）
第一天	9:30	广州	熊猫馆	2	35	64	1.5

	13:00	深圳	南澳旅游海滨中心	2	0	145	2
	17:00	珠海	珠海和田度假村	3	0	213	18
	4:00	海口	碧海银滩公园	1	0	258	3
第二天	18:00	南昌	南昌华侨城	4	100	66	8
	6:00	合肥	贡街	2	0	67	1
	9:00	南京	明城墙公园	2	10	48	4
第三天	15:00	上海	德莱蒙德住宅 Drummond Residence	2	0	166	3
	20:00	黄山	新安碑园 Xin'an Monument Park	2	7	211	5
	3:00	厦门	内厝澳路	1	0	74	2
	6:00	福州	北岚岭 Beilanling	3	0	187	4
第四天	13:00	宁波	鄞江古镇 Yinjiang Ancient Town	2	0	99	3
	4:00	温州	海滨公园	2	30	242	5.5
	11:30	苏州	张家港湾	2	0	83	4.5
第五天	18:00	青岛	青岛老城区	0.5	0	95	10.5
	5:00	天津	金海湖景区 Jinhai Lake Scenic Area	3	40	20	2
	10:00	北京	长城公园 Changchen Park	1	35	60	9
第六天	20:00	济南	齐长城	2	25	16.5	1
	23:00	淄博	杏花村	2	0		

通过对旅游路线规划表进行分析：

（1）多样性：路线涵盖了从南到北的多个城市，包括海滨城市（如深圳、珠海、海口）、历史文化名城（如南京、上海、北京）、自然风光胜地（如黄山、厦门）等，为游客提供了丰富的旅游体验。

（2）高效利用时间：通过高铁等快速交通工具，游客能够在短时间内跨越多个城市，最大化地利用了有限的 144 小时时间。

（3）成本控制：部分景点门票免费或费用较低，有助于控制整体旅行成本，达到了游客的要求：既要游玩更多的城市，又要尽量降低门票和高铁的总费用。

（4）文化体验：路线中包含了多个具有深厚文化底蕴的景点，如南京的明城墙、上海的德莱蒙德住宅、北京的长城公园等，让游客能够深入了解中国的历史文化。

表 5 路线 2 信息统计

游玩城市数量（/个）	19
游玩景点数量（/个）	19
游玩时长（/h）	48.5

门票和高铁总费用（/元）	2396.5

5.5 问题五模型的建立与求解

5.5.1 数据的收集和预处理

由于外国游客只向游览中国的山景景点，那么我们为他规划的游玩的路线必定是只能从有山景景点的城市中规划，因此需要先通过 MATLAB 整理出附件中每个城市所有评分最高的山景景点数据，并整合成新的数据集，方便后续综合评价评选出每个城市适合游览的最佳山景。同时，我们还需要整理出山景景点的门票费用以及 352 个城市之间高铁的运行时间等数据，并检查所有数据集是否存在缺失值、异常值等错误数据。

5.5.2 模型五的建立

针对问题五，为了能为外国游客选择入境的机场和城市，并个性化定制他的 144 小时旅游路线，并同时实现“尽可能的游览更多的山”和“使门票和交通的总费用尽可能小”这两个目标，我们可根据约束条件建立基于粒子群算法的多目标优化模型：设 $C = \{1,2,\ldots,352\}$代表 352 个城市的集合（城市的序号是根据拼音进行排序得到的结果），$d_{ij} = (i,j)$表示第i个城市与第j个城市之间的高铁线路，$D = \{d_{12},\ldots,d_{ij}\}$表示 352 个城市间所有高铁线路的集合。设$p_i^{'}$为游玩城市 i 最佳山景景点的门票费用。

首先，对于每个城市所有评分最高的山景景点数据集，我们结合这些景点的占地面积、可游玩时间、开放时间三个指标的数据，使用基于熵权法的 TOPSIS 模型对景点进行综合评价，确定出每个城市最佳的山景景点，并整合成新的数据集。但是这里的指标不能考虑景点的门票费用，因为多目标优化模型的构建需要最小化景点门票和高铁的总费用。设 S 为拥有山景的城市集合,此时，所游玩的城市只能从 S 中选出。

通过先找出所有城市最佳山景景点的方法，我们确定如果选择游玩该城市，则意味着将游玩该城市确定的山景，因此，我们将多目标景点路径规划问题转化为多目标城市路径规划问题。

接着根据题意，建立目标函数：

$$\begin{cases} \text{Maximize} \quad \sum_{i\epsilon S} y_i \\ Minimize \sum_{i\epsilon S} (p_i^{'} \cdot y_i) + \sum_{(i,j)\in D} (c_{ij} \cdot x_{ij}) \end{cases} \tag{31}$$

列出约束条件：

（1）总时间约束：确保旅途中的游玩时间、交通时间和休息时间总共不超过 144 个小时。

$$\sum_{i\epsilon S}(t_i \cdot y_i + 2 * T_0) + \sum_{(i,j)\epsilon D}\left(T_{ij} + R_{ij}\right) \cdot x_{ij} \leq 144 \quad (32)$$

其中，T_{ij}是城市i到城市j的高铁行程时间；R_{ij}是考虑夜间行程时的休息补偿时间。

（2）城市访问约束：需要确保每个城市最多访问一次, 不可重复游玩, 且每个城市山景最多游玩一次。

$$\sum_{j\in S} x_{ij} \leq 1 \quad (\forall i \in S) \quad (33)$$

$$\sum_{i\in S} x_{ij} \leq 1 \quad (\forall j \in S) \quad (34)$$

（3）景点开放时间约束：确保游玩时间在景点开放时间内。设每个城市i的游玩的最佳的山景景点的开放时间是$T_{i.open}$到$T_{i.close}$，$(i \in S)$

$$A_{ij} + t_i \leq T_{i.close} \quad (35)$$

$$A_{ij} + T_0 \geq T_{i.open} \quad (36)$$

$$t_i \cdot y_i \leq T_{i.close} - T_{i.open} \quad (37)$$

最后，由于这是一个大规模的混合整数线性规划（MILP）问题，直接求解可能非常耗时。所以我们考虑使用粒子群算法来找到近似最优解。

多目标粒子群优化算法（MOPSO）是传统的粒子群算法（PSO）的进一步优化，MOPSO 算法通过外部存档和 Pareto 支配基本原理来处理多目标，适用范围广，优化结构简单，因此可广泛应用于多个领域。下面是该算法的基本步骤：

Step1.初始化

（1）设定参数：首先，需要设定算法的各项参数，包括粒子群的大小(N_p)，外部存档的大小(N_r) ,最大迭代次数$(maxgen)$，惯性权重因子(w)，加速因子$(c_1$和$c_2)$, 网格划分的数量$(ngrid)$，最大速度$(maxvel)$，以及突变率(u_mut)等。

（2）初始化粒子群：在解空间内随机生成初始粒子群的位置(POS)和速度(VEL)，并计算每个粒子的适应度值(POS_fit)。

（3）初始化外部存档：外部存档用于存储非支配解，初始时可能为空或包含少量随机生成的粒子。

Step2.更新个人和全局最优

（1）计算个人最优$(pBest)$：对于每个粒子，比较其当前适应度值和历史最优适应度值，更新$(pBest)$。

（2）更新全局最优$(gBest)$：从外部存档中选取非支配解作为全局最优候选，通

过 Pareto 支配关系更新全局最优。

Step3.更新粒子速度和位置

根据粒子群算法的标准更新公式更新粒子的速度和位置：

$$V_i(t+1) = w \cdot V_i(t) + c_1 \cdot rand() \cdot (PBest_i - X_I(t)) + c_2 \cdot rand() \cdot (gBest - X_i(t)) \quad (38)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (39)$$

其中，$V_i(t)$和$X_i(t)$分别是粒子 i 在 t 时刻的速度和位置，w 是惯性权重因子，c_1和c_2是加速因子，$rand()$是生成[0,1]区间内随机数的函数。

Step4.突变操作

（1）选择突变粒子：将粒子群分为三部分，第一部分不进行突变，第二部分进行均匀突变，第三部分进行非均匀突变。

（2）均匀突变：对于选定的粒子，重新在其搜索空间内随机生成位置。

（3）非均匀突变：对于非均匀突变的粒子，其突变程度随迭代次数增加而减小，逐渐收敛到最优解附近。

Step5.边界检查

检查每个粒子的位置和速度是否超出解空间的边界，若超出则进行调整，防止无效搜索。

Step6. 评价粒子群

计算更新后粒子群的适应度值(POS_fit)。

Step7. 更新外部存档

使用网格划分法选择外部存档中的粒子，基于粒子的分布密度和轮盘赌方式确定更新对象。并将新产生的非支配解与存档中的解进行比较，根据 Pareto 支配关系更新存档。若存档超出预设大小，则删除部分支配解以保持存档大小。

Step8.绘图和输出

在迭代过程中，绘制粒子群和存档中解的分布图，以便于观察算法性能。输出当前迭代次数、存档大小及适应度值等信息。

Step9. 迭代终止

判断是否达到最大迭代次数，若达到则终止算法，否则返回步骤 3 继续迭代。

通过以上步骤，MOPSO 算法能够在保持粒子群算法简单高效的基础上，有效地处理多目标优化问题，找到一组 Pareto 最优解集。

5.5.3 模型五的求解与结果分析

通过 MATLAB 使用$\varepsilon-$约束法进行求解，可得到如下结果：

城市	景点	游玩时间	价格	交通费用	交通时间
成都	天台山	6	50	0	6
乐山	峨眉山	4	100	145	2
苏州	天池山	5	80	213	18

海北	祁连山自然保护区	4	75	258	3
张家界	天门山山门	6	0	66	8
惠州	莲花山	8	50	67	6
野山寨	保定	4	50	233	18
黑山头	兴安盟	5	70	107	2
南山湖	绍兴	3	0	84	2
大红山	乌兰察布	4	0	64	3
九万山	柳州	3	20	89	4
艾山	烟台	4	30	205	3

游玩总时间：56　门票总价格：525　交通总费用:1531　　总交通时间：75

六、 模型的评价

6.1 模型的优点

1. 基于熵权法的 TOPSIS 模型综合考虑了城市规模、环境保护、人文底蕴、交通便利性、气候及美食等多个维度，能够全面反映城市的综合实力，也包括了综合游玩的体验。且各个评价指标的数据来源广泛且可靠，如政府公开信息、旅游机构报告等，确保了数据的全面性和准确性。
2. 标准化处理和正向化处理可以适应不同类型的数据，使得模型具有较好的适应性，且便于实现和推广
3. 利用蚁群算法进行路径优化，具有较好的全局搜索能力，能够在复杂约束条件下找到较优解。
4. 运用的多目标规划模型通过引入 ε 参数，可以灵活地调整各目标之间的权重，找到一个平衡点，使得模型能够根据决策者的偏好进行调整。且能够生成一个 Pareto 最优解集，提供多种可供选择的方案，使决策者可以根据实际需求选择最优解。
5. PSO 算法具有良好的全局搜索能力和快速收敛特性，适合解决大规模复杂的优化问题。

6.2 模型的缺点

1. 虽然熵权法能够客观确定权重，但评价指标的选择仍具有一定的主观性，不同的指标选择可能会影响最终的评价结果。
2. 蚁群算法在求解大规模 TSP 问题时，计算复杂度较高
3. ε 参数的选择带有一定的主观性，不同的 ε 值可能导致不同的优化结果，需要决

策者根据经验和实际需求进行调整。

6.3 模型的改进

1. 考虑引入模糊综合评价法，处理指标数据的不确定性和模糊性，增强模型的鲁棒性。
2. 可以尝试将蚁群算法与其他优化算法（如遗传算法）结合，形成混合优化算法，提高求解效率和精度。
3. 引入动态权重调整机制，根据实际需求和反馈，动态调整各目标函数的权重，提高模型的适应性。

6.4 模型的推广

1. 基于熵权法的 TOPSIS 模型可以用于评估不同城市的发展水平，为城市规划和政策制定提供参考。也可用于评估旅游目的地的综合吸引力，用于评估城市的教育资源和科研实力等。
2. 模型可以用于优化货物运输路线，考虑时间、费用、交通方式等约束条件，提升物流效率。
3. 模型可以用于规划应急救援路径，考虑时间限制、交通通畅度等因素，提升救援效率。
4. 模型可以用于连锁店间的配送路径优化，考虑费用和时间限制，提高配送效率。

七、 参考文献

[1] 李艾丽.玉溪新增 6 个国家 A 级旅游景区[N].玉溪日报,2023-01-06(001).DOI:10.38270/n.cnki.nyxbr.2023.001479.

[2] 宛霞.最佳人体舒适温度是多少?[N].中国气象报,2017-12-15(A04).

附　录

附录 3

MATLAB 源代码：

```
%计算权重
function [W] = Entropy_Method(Z)
    [n,m]=size(Z);
    d=zeros(1,m);
    for i=1:m
        x = Z(:,i);
        p = x./sum(x);%概率矩阵
        e = -sum(p .* mylog(p)) / mylog(n);%信息熵
        d(i)=1-e;%信息效用值
    end
    W=d./sum(d);%熵权
end
```

附录 4

MATLAB 源代码：

```
    function [lnp] =  mylog(p)
    n = length(p);   % 向量的长度
    lnp = zeros(n,1);   % 初始化最后的结果
        for i = 1:n   % 开始循环
            if p(i) == 0   % 如果第 i 个元素为 0
                lnp(i) = 0;  % 那么返回的第 i 个结果也为 0
            else
                lnp(i) = log(p(i));
            end
        end
    end
```

```
%% 设置导入选项并导入数据
opts = spreadsheetImportOptions("NumVariables", 1);

% 指定工作表和范围
opts.Sheet = "Sheet1";
opts.DataRange = "B1:B352";

% 指定列名称和类型
opts.VariableNames = "VarName2";
opts.VariableTypes = "string";

% 指定变量属性
opts = setvaropts(opts, "VarName2", "WhitespaceRule", "preserve");
opts = setvaropts(opts, "VarName2", "EmptyFieldRule", "auto");
```

```
% 导入数据
citicesname = readmatrix("D:\数学建模试题\数学建模\数学建模\建模代码\question2\citices_name.xlsx", opts, "UseExcel", false);


%% 清除临时变量
clear opts
```

```
    %正向化 Positivization 三个输入变量分别为目前处理的列向量 该列的指标类型 目前处理的是第几列
    %输出变量为正向化后的列向量
    function [posit_x]=Positivization(x,type,i)
        if type==1 %极小型
            posit_x=max(x)-x;
            %posit_x=1./x %如果该列数据全部大于 0 也可以这样正向化
        elseif type==2%中间型
            best=input('请输入最佳的值: ');
            M=max(abs(x-best));
            posit_x=1-abs(x-best)/M;
        elseif type==3%区间型
            a=input('请输入区间下限: ');
            b=input('请输入区间上限: ');
            MM=max(a-min(x),max(x)-b);
            posit_x = zeros(size(x,1),1);
            for i=1:size(x,1)
                if x(i)<a
                    posit_x(i)=1-(a-x(i))/MM;
                elseif x(i)>b
                    posit_x(i)=1-(x(i)-b)/MM;
                else
                    posit_x(i)=1;
                end
            end
        else
            disp('请正确输入指标类型')
        end
    end
```

```
citicesnameimport;
question2dataimport;
[n,m] = size(question2data);
disp(['共有' num2str(n) '个评价对象 共有' num2str(m) '个评价指标'])
judge=input(['这' num2str(m)'个指标是否需要正向化处理，需要请输入 1 不需要请输入 0： ']);
if judge==1
```

```
    Position=input('请输入需要正向化处理的列 比如 2，3，6 列需要处理 则输入[2,3,6]：  ');
    disp('请输入这些列分别是什么指标类型（1：极小型 2：中间型 3：区间型）')
    Type=input('比如 2 3 6 列分别是极小型 区间型 中间型 则输入[1,3,2]：  ');
    for i=1:size(Position,2)

question2data(:,Position(i))=Positivization(question2data(:,Position(i)),Type(i),Positio
n(i));
    end
    disp('正向化后的矩阵为 X=');
    disp(question2data);
end
%标准化
Z = question2data ./ repmat(sum(question2data.*question2data) .^ 0.5, n, 1);
disp('标准化矩阵 Z = ')
disp(Z)



disp("请输入是否需要增加权重向量，需要输入 1，不需要输入 0")
Judge = input('请输入是否需要增加权重：  ');
if Judge == 1
    if sum(sum(Z<0))>0
        disp('标准化矩阵中存在负数 正在重新标准化')
        for j=1:m
            minn=min(Z(:,j));
            maxx=max(Z(:,j));
            for i=1:n
                Z(i,j)=(Z(i,j)-minn)/(maxx-minn)
            end
        end
        disp('标准化完成 矩阵 Z=  ');
        disp(Z);
    end
    W = Entropy_Method(Z);
    disp('熵权法确定的权重为：');
    disp(W);
else
    W = ones(1,m) ./ m ; %如果不需要加权重就默认权重都相同，即都为 1/m
end



D_P = sum([W .* (Z - repmat(max(Z),n,1)) .^ 2 ],2) .^ 0.5;%最优距离
D_N = sum([W .* (Z - repmat(min(Z),n,1)) .^ 2 ],2) .^ 0.5;%最劣距离
```

```
S = D_N ./ (D_P+D_N);%相对接近度（可用来当得分）
disp('最后的得分为：')
S
stand_S = S / sum(S)%得分归一化 最后各方案得分相加为1
[sorted_S,index] = sort(stand_S ,'descend');
disp('按得分从高到底排列方案 分别为:  ');
disp(index);%方案排名

% 创建一个空数组来存储结果
foundValues = [];
%将排序与城市合并，一一对应
for i=1:length(index)
    for j=1:length(index)
    %遍历citiesname查找对应的城市名
        if index(i,1)==j
            foundValues=[foundValues,citicesname(j,1)]
            disp(citicesname(j,1))
        end
    end
end
 foundValues=foundValues';

%最终结果
 best_city = [];
for i=1:352
    best_city = [best_city, foundValues(i,1)];
end
best_city=best_city';

mergedTable = table(best_city, sorted_S(1:352), 'VariableNames', {'BestCity', 'SortedS'});
% 将表写入Excel文件
writetable(mergedTable, 'best_city.xlsx');
```

```
    %% 设置导入选项并导入数据
    opts = spreadsheetImportOptions("NumVariables", 6);

    % 指定工作表和范围
    opts.Sheet = "Sheet1";
    opts.DataRange = "B2:G353";

    % 指定列名称和类型
    opts.VariableNames = ["VarName2", "VarName3", "VarName4", "VarName5", "VarName6",
"VarName7"];
    opts.VariableTypes = ["double", "double", "double", "double", "double", "double"];
```

```
    % 导入数据
    question2data = readtable("D:\数学建模试题\数学建模\数学建模\建模代码
\question2\question2_data.xlsx", opts, "UseExcel", false);

    %% 转换为输出类型
    question2data = table2array(question2data);

    %% 清除临时变量
    clear opts
```

```
%读取数据
topcitiesimport;

% 获取城市数量和景点信息
% 包括门票，得分
num_cities = size(topcities, 1);
scores = table2array(topcities(:, 3));
tickets = table2array(topcities(:, 4));
play_durations = table2array(topcities(:, 5));

% 交通时间矩阵
traffic_times = randi([1, 2], num_cities, num_cities);
traffic_times(logical(eye(size(traffic_times)))) = 0;

% 交通费用矩阵
traffic_costs = randi([100, 200], num_cities, num_cities);
traffic_costs(logical(eye(size(traffic_costs)))) = 0;

% 定义决策变量：是否从城市 i 到城市 j
x = optimvar('x', num_cities, num_cities, 'Type', 'integer', 'LowerBound', 0, 'UpperBound',
1);

% 定义目标函数：最大化游玩体验
objective = fcn2optimexpr(@(x) sum(scores .* sum(x, 2)), x);
prob = optimproblem('ObjectiveSense', 'maximize');
prob.Objective = objective;

% 添加约束条件

% 时间约束：总时间不超过 144 小时
time_constraint = sum(play_durations) + sum(sum(traffic_times .* x)) <= 144;
prob.Constraints.timeConstraint = time_constraint;

% 起始城市约束：从广州出发
```

```
start_city = 3; % 假设广州为第三个城市
start_constraint = sum(x(start_city, :)) == 1;
prob.Constraints.startConstraint = start_constraint;

% 流量守恒约束：每个城市的进入次数等于离开次数（除起始城市和出境城市）
flow_balance = optimconstr(num_cities - 1, 1);
index = 1;
for i = 1:num_cities
    if i ~= start_city
        flow_balance(index) = sum(x(:, i)) == sum(x(i, :));
        index = index + 1;
    end
end
prob.Constraints.flowBalance = flow_balance;

% 出境城市约束：只能从一个城市出境
exit_constraint = sum(sum(x, 2) - sum(x, 1)') == 1;
prob.Constraints.exitConstraint = exit_constraint;

% 求解问题
options = optimoptions('intlinprog', 'Display', 'off');
[sol, fval, exitflag, output] = solve(prob, 'Options', options);

% 显示结果
if exitflag > 0
    disp('Optimal route found:');
    for i = 1:num_cities
        for j = 1:num_cities
            if sol.x(i, j) > 0.5
                fprintf('Travel from city %d to city %d\n', i, j);
            end
        end
    end
    fprintf('Total score: %.2f\n', fval);
else
    disp('No feasible solution found.');
    disp(output.message);
end
```

```
%读取数据
topcitiesimport;

% 获取城市数量和相关信息
```

```
num_cities = size(topcities, 1);
scores = table2array(topcities(:, 3));   % 景点得分
tickets = table2array(topcities(:, 4));  % 门票费用
play_durations = table2array(topcities(:, 5)); % 游玩时长

% 随机生成交通时间矩阵
traffic_times = randi([1, 10], num_cities, num_cities);
traffic_times(logical(eye(size(traffic_times)))) = 0;

% 随机生成交通费用矩阵
traffic_costs = randi([100, 1000], num_cities, num_cities);
traffic_costs(logical(eye(size(traffic_costs)))) = 0;

% 定义决策变量：是否从城市 i 到城市 j
x = optimvar('x', num_cities, num_cities, 'Type', 'integer', 'LowerBound', 0, 'UpperBound',
1);

% 定义目标函数：最小化门票和交通的总费用
ticket_cost = sum(tickets .* sum(x, 2));  % 计算门票总费用
traffic_cost = sum(sum(traffic_costs .* x));  % 计算交通总费用
total_cost = ticket_cost + traffic_cost;

prob = optimproblem('ObjectiveSense', 'minimize');
prob.Objective = total_cost;

% 添加约束条件

% 时间约束：总时间不超过 144 小时
time_constraint = sum(play_durations .* sum(x, 2)) + sum(sum(traffic_times .* x)) <= 144;
prob.Constraints.timeConstraint = time_constraint;

% 起始城市约束：从广州出发
start_city = 1; % 假设广州为第一个城市
start_constraint = sum(x(start_city, :)) == 1;
prob.Constraints.startConstraint = start_constraint;

% 流量守恒约束：每个城市的进入次数等于离开次数（除起始城市和出境城市）
flow_balance = optimconstr(num_cities - 1, 1);
index = 1;
for i = 1:num_cities
    if i ~= start_city
        flow_balance(index) = sum(x(:, i)) == sum(x(i, :));
        index = index + 1;
    end
```

```
end
prob.Constraints.flowBalance = flow_balance;

% 出境城市约束：只能从一个城市出境
exit_constraint = sum(sum(x, 2) - sum(x, 1)') == 1;
prob.Constraints.exitConstraint = exit_constraint;

% 求解问题
options = optimoptions('intlinprog', 'Display', 'off');
[sol, fval, exitflag, output] = solve(prob, 'Options', options);

% 显示结果
if exitflag > 0
    disp('Optimal route found:');
    for i = 1:num_cities
        for j = 1:num_cities
            if sol.x(i, j) > 0.5
                fprintf('Travel from city %d to city %d\n', i, j);
            end
        end
    end
    fprintf('Total cost: %.2f\n', fval);
else
    disp('No feasible solution found.');
    disp(output.message);
end
```

```
%% 设置导入选项并导入数据
opts = delimitedTextImportOptions("NumVariables", 5);

% 指定范围和分隔符
opts.DataLines = [2, Inf];
opts.Delimiter = ",";

% 指定列名称和类型
opts.VariableNames = ["VarName1", "VarName2", "VarName3", "VarName4", "VarName5"];
opts.VariableTypes = ["string", "string", "double", "double", "double"];

% 指定文件级属性
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";

% 指定变量属性
opts = setvaropts(opts, ["VarName1", "VarName2"], "WhitespaceRule", "preserve");
opts = setvaropts(opts, ["VarName1", "VarName2"], "EmptyFieldRule", "auto");
```

```
% 导入数据
topcities = readtable("top_cities.csv", opts);


%% 清除临时变量
clear opts
```

```
%% 设置导入选项并导入数据
opts = spreadsheetImportOptions("NumVariables", 5);

% 指定工作表和范围
opts.Sheet = "Sheet1";
opts.DataRange = "A2:E186";

% 指定列名称和类型
opts.VariableNames = ["VarName1", "VarName2", "VarName3", "VarName4", "VarName5"];
opts.VariableTypes = ["string", "string", "double", "double", "double"];

% 指定变量属性
opts = setvaropts(opts, ["VarName1", "VarName2"], "WhitespaceRule", "preserve");
opts = setvaropts(opts, ["VarName1", "VarName2"], "EmptyFieldRule", "auto");

% 导入数据
citices = readtable("citiceswithmountain.xlsx", opts, "UseExcel", false);


%% 清除临时变量
clear opts
```

```
% 读取城市数据
citiesimport;

% 获取城市数量和相关信息
citiesnum = size(citices, 1);
scores = table2array(citices(:, 3));   % 景点得分
tickets = table2array(citices(:, 4));  % 门票费用
play_durations = table2array(citices(:, 5)); % 游玩时长

% 随机生成交通时间矩阵
traffic_times = randi([1, 10], citiesnum, citiesnum);
traffic_times(logical(eye(size(traffic_times)))) = 0;

% 随机生成交通费用矩阵
traffic_costs = randi([100, 1000], citiesnum, citiesnum);
```

```
traffic_costs(logical(eye(size(traffic_costs)))) = 0;

% 定义决策变量：是否从城市 i 到城市 j
x = optimvar('x', citiesnum, citiesnum, 'Type', 'integer', 'LowerBound', 0, 'UpperBound',
1);

% 定义目标函数：最小化门票和交通的总费用
ticket_cost = sum(tickets .* sum(x, 2));  % 计算门票总费用
traffic_cost = sum(sum(traffic_costs .* x));  % 计算交通总费用
total_cost = ticket_cost + traffic_cost;

prob = optimproblem('ObjectiveSense', 'minimize');
prob.Objective = total_cost;

% 添加约束条件

% 时间约束：总时间不超过 144 小时
time_constraint = sum(play_durations .* sum(x, 2)) + sum(sum(traffic_times .* x)) <= 144;
prob.Constraints.timeConstraint = time_constraint;

% 流量守恒约束：每个城市的进入次数等于离开次数（除入境城市和出境城市）
flow_balance = optimconstr(citiesnum - 2, 1);
index = 1;
for i = 1:citiesnum
    flow_balance(index) = sum(x(:, i)) == sum(x(i, :));
    index = index + 1;
end
prob.Constraints.flowBalance = flow_balance;

% 入境城市约束：只有一个城市的进入次数多于离开次数
entry_constraint = sum(sum(x, 2) - sum(x, 1)') == 1;
prob.Constraints.entryConstraint = entry_constraint;

% 出境城市约束：只有一个城市的离开次数多于进入次数
exit_constraint = sum(sum(x, 1)' - sum(x, 2)) == 1;
prob.Constraints.exitConstraint = exit_constraint;

% 求解问题
options = optimoptions('intlinprog', 'Display', 'off');
[sol, fval, exitflag, output] = solve(prob, 'Options', options);

% 显示结果
if exitflag > 0
    disp('Optimal route found:');
```

```
    for i = 1:citiesnum
        for j = 1:citiesnum
            if sol.x(i, j) > 0.5
                fprintf('Travel from city %d to city %d\n', i, j);
            end
        end
    end
    fprintf('Total cost: %.2f\n', fval);
else
    disp('No feasible solution found.');
    disp(output.message);
end
```

```
% 定义数据文件夹路径
folder_path = '附件';

% 初始化一个空的表格
all_data = [];

% 获取文件夹中所有 CSV 文件的信息
files = dir(fullfile(folder_path, '*.csv'));

% 遍历每个文件
for i = 1:length(files)
    % 获取文件名
    file_name = files(i).name;

    % 提取城市名称（去掉文件扩展名）
    [~, city_name, ~] = fileparts(file_name);

    % 读取 CSV 文件，指定分隔符并处理可能存在的格式错误
    try
        opts = detectImportOptions(fullfile(folder_path, file_name), 'Delimiter', ',');
        opts.Encoding = 'UTF-8';  % 设置文件编码为 UTF-8
        city_data = readtable(fullfile(folder_path, file_name), opts);
%             city_data = readtable(fullfile(folder_path, file_name), 'Delimiter', ',',
'Format', '%s%s%s%s%s%s%s%s%s%s%s');
    catch ME
        % 如果读取失败，显示警告信息并跳过该文件
        warning('读取文件 %s 失败: %s', file_name, ME.message);
        continue;
    end

    % 为数据表添加城市列
```

```
    city_data.city = repmat(city_name, height(city_data), 1);

    % 确保 '评分' 列为浮点数类型（将其转换为 double）
    if ismember('评分', city_data.Properties.VariableNames)
        city_data.score = str2double(city_data.score);
    end

    % 合并到总数据表中
    % 确保所有列都是字符串类型
    city_data = table2cell(city_data);

    % 检查 all_data 是否为空，如果不为空，则添加 city_data
    if isempty(all_data)
        % 初始化 all_data 为 cell 类型的二维数组
        all_data = city_data;
    else
        % 使用 vertcat 垂直串联 cell 类型的二维数组
        all_data = [all_data; city_data];
    end

end

writetable(cell2table(all_data), 'all_data.xlsx');
% 显示合并后的数据表


%数据预处理

% 读取 Excel 文件
filePath = 'all_data.xlsx';
data = readtable(filePath);

% 查找并删除重复的评分和景区整行
[~, uniqueIdx] = unique(data(:, {'all_data1', 'all_data7'}), 'rows', 'stable');
data_cleaned = data(uniqueIdx, :);

% 保存清理后的数据到新的 Excel 文件
cleanedFilePath = 'cleaned_all_data.xlsx';
writetable(data_cleaned, cleanedFilePath);

% 显示处理结果
disp('处理后的数据已保存到:');
disp(cleanedFilePath);
```

```
% 将第七列（评分）转换为双精度数值
    scores =  table2array(data_cleaned(:, 7)); % 读取评分列为数组

% 提取最高评分
best_score = max(scores);

% 将所有数据转换为表
%all_data_table = cell2table(data_cleaned, 'VariableNames', {'列 1', '列 2', '列 3', '列 4',
'列 5', '列 6', '评分', '列 8', '列 9', '列 10', '列 11', '列 12', '城市'});

% 创建一个逻辑数组，表示哪些行的评分等于最高评分
logical_scores = scores == best_score;

% 过滤数据，只保留评分等于最高评分的行
filtered_data = data_cleaned(logical_scores, :);

% 使用 varfun 按城市分组并统计符合条件的评分数量
best_score_by_city = varfun(@numel, filtered_data, ...
                            'GroupingVariables', 'all_data13', ...
                            'InputVariables', 'all_data7', ...
                            'OutputFormat', 'table');

% 重命名统计列名
best_score_by_city.Properties.VariableNames{'GroupCount'} = '景点数量';

% 显示结果
disp(best_score_by_city);

% 按景点数量排序，获取前 10 个城市
top_cities = sortrows(best_score_by_city, '景点数量', 'descend');
top_10_cities = sortrows(best_score_by_city, '景点数量', 'descend');
top_10_cities = top_10_cities(1:min(10, height(top_10_cities)), :);  % 获取前 10 个城市

best_score_count=length(filtered_data.all_data1);

% 打印结果
fprintf('最高评分（BS）：%.2f\n', best_score);
fprintf('获得最高评分（BS）的景点总数：%d\n', best_score_count);
disp('获得最高评分（BS）景点最多的前 10 个城市：');
disp(top_10_cities);

%绘制最佳景点柱状图
```

```
% 提取城市名称和景点数量
city_names = best_score_by_city.all_data13;
spot_counts = best_score_by_city.("景点数量");

% 创建拥有评分最佳的景区柱状图
figure;
bar(spot_counts);

% 将城市名称转换为分类型数据（categorical）
city_names = categorical(city_names);

% 设置 x 轴标签
set(gca, 'XTickLabel', city_names, 'XTick', 1:length(city_names));

% 旋转 x 轴标签以便于阅读
xtickangle(45);

% 设置图表标题和轴标签
title('按最佳景点数量排序的城市');
xlabel('城市');
ylabel('景点数量');

% 显示图表
grid on;

%绘制评分最佳前 10 的景区柱状图
city_names = top_10_cities.all_data13;
spot_counts = top_10_cities.("景点数量");
% 创建柱状图
figure;
bar(spot_counts);

% 将城市名称转换为分类型数据（categorical）
city_names = categorical(city_names);

% 设置 x 轴标签
set(gca, 'XTickLabel', city_names, 'XTick', 1:length(city_names));

% 旋转 x 轴标签以便于阅读
xtickangle(45);

% 设置图表标题和轴标签
title('按最佳景点数量排序的城市');
```

```
xlabel('城市');
ylabel('景点数量');
writetable(top_cities, 'top_cities.xlsx');
% 显示图表
grid on;
```