2025 年第十届“数维杯”大学生
数学建模挑战赛论文

题 目 **基于多目标规划的马拉松窗口期筛选与赛道优化研究**

摘 要

随着全民健身理念的深入推进，马拉松赛事在中国城市快速发展，呈现出参与度高、举办频次快的趋势。面对赛事激增带来的管理压力，如何合理安排时间、优化赛道设计、提升选手体验与激励机制，成为赛事组织亟需解决的问题。基于此，本文聚焦中国重点城市，围绕四个任务构建了数据驱动的建模与优化体系，旨在为赛事精细化管理与科学决策提供量化支持和实证依据。

针对问题一，本文结合气象条件适宜度、城市承载能力、常住人口规模以及赛事报名热度四个关键因素，构建了一个用于评估最优赛事举办时机和规划安排的综合评分模型。模型通过熵权法（EWH）动态赋权，以 TOPSIS 方法对多城市进行优劣排序，系统识别在多维条件下具备举办优势的地区。结果表明，上海、北京等城市凭借良好的气候环境与运营保障能力，在赛事安排上具有显著优势。本文还结合历史数据与城市承载阈值，提出了赛事规模与举办频次的分类建议，为城市定制赛事运营策略提供依据。

针对问题二，本文以西安市为典型样本，构建融合住宿接待能力、轨道交通覆盖度等因素的图模型-遗传算法联合优化模型，用于规划马拉松赛事的起终点布局及赛道路径。结果表明，运用该模型将城市道路系统抽象为带权图，引入整数规划与多重约束条件，借助遗传算法，实现多条符合国际标准、路径可行性高、关键节点覆盖优的候选路线。该方法有效应对了多维约束下的路径优化难题，为其他城市提供了可推广的赛道规划范式。

针对问题三，在保障城市正常运行的前提下，本文从树荫覆盖率、交通干扰、赛道坡度等要素出发，构建融合 Dijkstra 算法、BallTree 空间索引和遗传算法的多目标优化模型。结果表明，模型基于综合评分体系，显示出最优路径与服务范围，可在复杂路网中筛选出兼顾舒适性与通行效率的赛道路径，实现对选手体验与运营效率的双重优化，为赛事方提供灵活、高质量的赛道设计依据。

针对问题四，本文运用 XGBoost-RankNet 模型对大量历史马拉松成绩数据进行了系统性分析，旨在为年龄分组调整和奖励机制优化提供量化依据。结果表明，不同年龄段选手在参赛人数分布和竞技水平上存在显著差异。基于此，本文建议年轻组别以 5 岁间隔划分，而对中高龄段则可适当合并至 10 岁分组。与此同时，本文设计了与成绩表现联动的专属纪念方案，通过发放定制奖牌、限量周边等方式增强选手荣誉感与参与动力。此外，本文构建的赛前模拟排名工具，能够根据个人历史数据预测其在同龄组中的预期位置，辅助选手科学制定比赛目标和训练计划，从而提升整体参赛体验与赛事互动性。

关键词 **EWH-TOPSIS、多目标优化模型、Dijkstra 算法、遗传算法**

目录

一、问题重述

1.1 背景介绍

近年来，随着全民健身理念的深入人心和城市品牌塑造需求的不断上升，马拉松赛事在中国各大城市呈现出爆发式增长态势。据中国田径协会统计，全国注册马拉松及相关赛事数量已突破千场，成为推动文旅融合、带动城市消费、提升城市影响力的重要载体。然而，部分城市在赛事举办中也暴露出时间安排不合理、赛道设计不科学、资源配置失衡等问题，亟需构建一套科学系统的办赛决策机制，以实现城市与赛事的良性互动。

本论文旨在建立一套科学合理的城市马拉松赛事举办时空规划机制，通过综合分析气象条件、城市接纳能力、人口规模、历年报名数据及其增长趋势等关键要素，精准筛选中国主要城市中最具办赛潜力的时间窗口，并据此合理规划赛事的举办时间、参赛规模及频次。在此基础上，论文以西安市为典型案例，结合其文旅资源布局（包括主要景点与住宿设施的空间分布），开展马拉松赛道路径的系统优化研究，进一步构建面向赛事体验、城市推广与资源协调的多目标优化模型，探索“城市—赛事”双赢的新型发展路径。

1.2 需要解决的问题

问题 1：城市马拉松赛事举办适宜性的定量评估机制如何构建？

需突破现有办赛时间安排主观性强、缺乏数据支撑的局限，构建融合气象适宜度、城市承载能力、人口规模与参赛热度等多维指标的量化评估体系。需明确不同城市在全年中的适宜办赛窗口期，并据此制定科学的赛事时间、规模与频次安排策略，实现“时间—空间—资源”的高效匹配。

问题 2：在复杂城市空间结构中，如何实现马拉松赛道与起终点的多目标路径优化？

需研究如何将城市空间数据（如景点、住宿设施、轨道交通节点等）抽象为可计算模型元素，构建基于容量约束、路网密度和资源服务范围的多目标优化模型，实现在满足国际赛事规范前提下的高效路径生成与起终点选址。该问题涉及图论优化、空间服务函数构建与路径搜索算法融合等关键技术。

问题 3：如何在保障赛道热舒适性的同时降低对城市交通系统的干扰？

当前赛道设计多未充分考虑环境舒适性与城市运行协调，本研究需解决如何量化赛道沿线的树荫覆盖度与高温暴露风险，并融合交通负载敏感性等因素，构建面向热舒适性与交通友好性的联合评估与优化模型。该问题涉及空间热环境建模、多源数据耦合与多目标优化算法设计。

问题 4：如何基于人群特征与赛事行为数据优化赛事分组与衍生价值机制？

需探索马拉松参赛人群的年龄结构与成绩分布规律，优化分组机制（如 5 岁区间划分）以增强赛事公平性。同时，需设计与赛事主题相关、具纪念价值的差异化衍生产品与兑换机制，并开发预测工具辅助选手设定合理目标。该问题涉及数据驱动的行为建模、赛事激励机制设计与人群分层策略优化。

二、问题分析

2.1 问题 1 的分析

问题 1 属于多准则决策优化问题，需要兼顾气象适宜性、城市承载力、人口规模和报名热度等多个相互影响的评价维度。这类问题通常具有目标冲突性和数据复杂性，本文采用综合评价与优化决策相结合的方法来解决。

对附件中所给数据特点研究分析，发现数据呈现多源异构特征，包括连续型气象数据、离散型人口统计数据以及分类别的交通数据等。这些数据在时间维度和空间维度上存在明显差异，需要先进行系统的数据清洗和标准化处理。

对问题 1 所要求的结果进行分析。我们需要输出每个城市的最优赛事窗口期、合理比赛时间、赛事规模与频次等决策方案。这些结果既要满足各项评价指标的优化要求，又要具备实际可操作性。

由于以上原因，我们可以首先基于多源数据，建立包含气象适宜性（温度、湿度、风速、降水）、城市承载能力（地铁客运量、住宿容量、道路密度）、人口规模（常住人口、人口密度）和赛事热度（历史报名人数及增长率）的四维指标体系。

在数据处理阶段，通过极差标准化消除指标量纲差异，并运用熵权法客观赋权，其中熵值较小的指标（如温度、地铁客运量等）获得更高权重。随后在完成数据标准化和权重分配后，采用 TOPSIS 方法对各候选城市进行综合评价。计算各城市与正负理想解的相对贴近度以及评分，根据不同城市评分高低排序，进而筛选出每个城市的最优赛事窗口期、合理比赛时间、赛事规模与频次等决策方案。

综上所述，我们需要一个综合评价体系，并采用数据驱动的权重确定与排序方法。这里选择熵权法（消除量纲、确定权重） + TOPSIS 模型（评分决策、方案排序）的组合方式。

2.2 问题 2 的分析

问题 2 属于具有复杂约束条件的组合优化类空间路径规划问题，兼具空间决策、资源配置与路径最优性等典型特征。该类问题在城市交通、赛事规划等领域中广泛存在，通常涉及节点选择、边权计算与约束筛选等多重任务。解决此类问题往往依赖于图论模型、规划理论与智能优化算法的综合应用。

针对该类问题，传统的优化方法如线性规划、整数规划、最短路径算法等在小规模场景下具有一定适用性，但在节点规模大、约束条件多的背景下容易陷入计算瓶颈，难以获得全局最优解。而近年兴起的遗传算法、蚁群算法等智能算法，凭借其较强的搜索能力和鲁棒性，在此类问题中的应用逐渐成熟，且能够在保证解可行性的基础上逼近最优解，成为解决城市级复杂路径规划问题的重要工具。

从附件所提供的数据来看，包含了多个候选景点、住宿资源分布、轨道交通站点位置以及城市路网结构信息，数据具有空间分布广、容量差异大、交通联系复杂等特点。这要求模型在构建过程中需充分考虑节点之间的空间关系、资源承载能力及路径连通性等因素，以提升方案的实际可行性与鲁棒性。

问题 2 要求输出一组满足特定约束的马拉松起终点组合及路线规划方案，不仅要满足基本长度要求和住宿容量等资源约束，还需兼顾交通接驳、路径连续性和赛事组织效率等综合指标。这使得问题解不仅是一个路径，更是一个兼顾多目标优化与实际运行条件的城市级系统方案。

因此，基于以上问题特征与数据结构，我们首先可建立一个以节点容量与交通覆盖度为核心的综合评分模型，在此基础上，再结合一个图模型与遗传算法的路径优化算法，在复杂约束条件下生成一组最优或次优路线方案。最终，我们对模型的输出结果进行对比与综合评价，验证所选路线的可行性与优越性，从而为赛事路径规划提供定量支持与优化建议。

2.3 问题 3 的分析

问题 3 属于多目标优化问题，涉及复杂的路径规划与城市功能优化。在解决此类问题时，一般会采用图论、空间数据分析和进化算法等数学优化方法。图论能够有效地描述城市路网结构，空间数据分析则有助于处理地理信息，提高路径规划的效率，进化算法则非常适合解决需要多目标平衡的复杂问题。

针对附件中提供的数据特点分析，发现数据包含了节点信息（如交叉口位置、道路长度、交通流量等），以及与赛道相关的环境数据（如树荫覆盖率、赛道坡度等）。这些数据不仅包括静态的地理信息，还涉及到动态的交通流量和环境影响因素，这使得问题更具挑战性，需要综合多维度的因素进行优化设计。

对于问题 3 所要求的结果分析，我们的目标是设计一个最优的赛道路径，在保障选手舒适性的同时，也要确保路径的通行效率。因此，路径的舒适性受到如树荫覆盖率、坡度等因素的影响，而通行效率则受到交通干扰的影响。因此，最终目标是找到一条在这些因素之间进行权衡的最优路径，以实现双重优化。

根据上述分析，我们首先建立了一个数学模型。该模型基于 Dijkstra 算法求解最短路径问题，确保路径的基本通行效率。Dijkstra 算法适用于城市路网的最短路径计算，它能够快速确定路径长度和交通情况。但该模型可能无法全面考虑所有环境和舒适性因素。在此基础之上，该模型还需融入遗传算法进行多目标优化。遗传算法又能够在多个优化目标之间进行权衡，综合考虑树荫覆盖率、赛道坡度、交通干扰等多种因素，从而得到最优路径方案。

通过建模，我们能提供通行效率较高的路径，且能够在综合考虑多重因素后，找到更加符合实际需求的最优路径。最后，我们将对两个模型的结果进行比较，并选择最符合问题 3 需求的最优解决方案，从而为赛道设计提供科学依据。

2.4 问题 4 的分析

针对问题 4，我们计划运用 XGBoost-RankNet 模型对大量历史马拉松成绩数据进行系统性分析，旨在为年龄分组调整和奖励机制优化提供量化依据。首先，我们将对不同年龄段选手的参赛成绩数据进行深入挖掘，分析各年龄段选手在参赛人数分布和竞技水平上的差异。我们预计，分析结果将揭示出不同年龄段选手在参赛人数及其竞技能力上的显著差异，特别是在年轻组与中高龄组之间的差距。

基于这一分析，我们提出针对年龄组别划分的优化方案。即，建议年轻组别（例如 18-30 岁）以 5 岁为间隔进行划分，以便更精准地反映年轻选手的竞技水平差异；而对于中高龄段（例如 50 岁及以上），我们适当合并至 10 岁为一组，以更合理地反映该年龄段选手的整体竞技水平和参赛人数。

在奖励机制方面，我们拟进行创新设计，提出与成绩表现紧密联动的专属纪念方案。通过发放定制奖牌、限量版赛事周边等方式，旨在增强选手的荣誉感与参与动力。这不仅可能提高选手的积极性，还能促进赛事品牌效应的提升，进一步增加赛事的吸引力和参与度。

此外，我们还计划构建一款赛前模拟排名工具，该工具将基于选手的历史成绩数据，预测其在同年龄组中的预期名次。这一工具预计能够帮助选手科学制定比赛目标，并根据预测结果优化训练计划，从而提升整体参赛体验和赛事互动性。通过这一工具，选手可以更清晰地了解自己的竞争力，并根据预期成绩调整备赛策略，从而增强赛事参与感与个人成就感。

三、模型假设

3.1 气象条件稳态假设

假设在所评估的时间区间内，各城市的气象状况（包括温度、湿度、风速、降水量等）具备一定的周期性和稳定性，不会出现突发性的极端气候事件（如台风、暴雨、高温预警等）。气象适宜性依据历史气象数据库与马拉松赛事公认的舒适气象区间进行综合评估，确保推荐的赛事窗口期具备良好的气候基础。

3.2 城市运行承载能力稳定假设

假设城市的基础运行能力在短期内保持相对稳定，核心参数包括轨道交通运力峰值、可用住宿容量、道路通达性与交通管控能力等。暂不考虑由于重大基建调整、突发公共事件或大型活动叠加所造成的异常波动。各类承载指标将以近年历史数据为基础进行趋势预测，并作为赛事规模和频率设置的约束条件。

3.3 人口规模与参赛积极性关联假设

假设城市常住人口与参赛报名热度呈正相关，即人口基数大、人口密度高的城市更可能吸引更多马拉松参赛者。报名热度作为参赛效益的代理指标，依据往年赛事的报名总量及年增长率建模，不考虑广告营销、新兴健康潮流或社会情绪对未来热度的短期影响。

3.4 城市同质性与可比性假设

为简化跨城市赛事评估与规划模型，假设样本城市间在经济发展水平、公共安全、文化接受度等软性条件上不存在显著差异。因此，模型主要聚焦于量化特征维度（如气象、交通、人口、资源分布等）进行分析，不纳入主观性较强或难以量化的影响因素，如政府支持度、品牌赞助强度等。

3.5 赛道规划标准化与封闭性假设

假设所有赛道设计均需满足国际马拉松赛事标准（如全程≥42.195 公里、闭合路线等），并在设计过程中严格遵守相关约束条件：包括最大坡度≤5%、每 5 公里设补给站、起终点邻近轨道交通节点等。赛道规划过程中将默认部分道路资源可在特定时段内封闭使用，且城市管理部门具备一定的资源调配能力以支持赛事运行。

3.6 空间要素抽象与增益机制假设

在赛道优化中，假设可将城市内的景点、餐饮、住宿等空间资源抽象为“节点”并赋予权重，用于衡量对参赛体验的增益作用。景点节点视为必须经过的重要区域，餐饮节点视为可选增益节点，依据选手经过数量赋予累积增益分值，用于评价赛道的吸引力和服务完备性。

3.7 技术可行性与数据可获取性假设

假设模型所需的基础数据（如高精度气象记录、城市路网、住宿设施坐标、历史报名数据等）均可通过公开数据库、政府开放平台或赛事主办方获取，数据质量具备建模分析的基本条件。同时假设优化模型具备可实现性，能够通过启发式算法、多目标规划工具等实现可计算的解空间搜索。

四、定义与符号说明

符号定义	符号说明
$I_{climate}$	气象适宜性维度
$I_{capacity}$	城市承载力维度
$I_{population}$	人口规模维度
$I_{popularity}$	赛事热度维度
i	表示城市-月份组合（例如 a 城市的 1 月，则 i 为 1，若 a 城市的 12 月，则 i 为 12 以此分类）
j	表示指标数值（一个城市为一个数值）
x_{ij}	某城市的第 i 月的第j个指标值
X=（x_{ij}）	定义原指标矩阵（针对问题 1）
S_{ij}	对正向指标和负向指标分别进行标准化后的值
S=（S_{ij}）	标准化后的指标矩阵
p_{ij}	第j个指标的比重

E_i	第i个指标的信息熵
m	评价对象的数量，即总的城市-月份组合数量
D_j	冗余度
W_j	指标j的权重，基于熵权法计算得出
Z_{ij}	加权规范化矩阵
R_j^+	正理想解，取加权标准化矩阵中每一列的最大值
R_j^-	负理想解，取加权标准化矩阵中每一列的最小值
A_j^+	城市-月份组合 i 与正理想解的欧几里得距离
A_j^-	城市-月份组合 i 与负理想解的欧几里得距离
C_i	城市-月份组合 i 的相对贴近度，表示该组合与理想解的接近度

五、模型的建立与求解

数据的预处理：

在对数学建模之前，首先要检查数据是否缺失以及需要将数据处理成模型所需要的数据。

我们针对附件 1 基于 Python 的 pandas 和 numpy 库对 NCDC 气象数据进行了系统化处理（见图 5-1）。首先，我们考虑时间、地域等相关因素，选取 2021-2024 年的气象数据（存储为 ZIP 压缩格式）作为我们模型的数据集进行处理，我们设计了自动化处理流程：通过 os 模块验证数据路径有效性并筛选目标年份的压缩文件；然后使用 zipfile 模块解压文件并读取 CSV 数据，同时定义标准化的列名结构。在数据清洗阶段，我们对异常值（-9999）进行标记处理，并执行单位换算（如温度、气压等数值除以 10）。通过 pd.to_datetime 方法构建时间戳字段，最终将所有站点数据合并为统一的数据集，输出为 CSV 文件以便后续分析。整个过程采用 tqdm 进度条监控处理进度，确保数据处理的完整性和可靠性，详细数据处理代码详见支撑材料中的代码汇总附件 1，相关数据 csv 文件详见支撑材料中表格数据-附件 1：2021 年-2024 年中国气象数据整理文件。

year	month	day	hour	temperature	dew_point	sea_pressure	wind_direction	wind_speed	cloud_cover	precip_1hr	precip_6hr	station_id	timestamp
2024	1	1	0	-9.6	-11.5	1037.5	100	2	-9999	0	0	510870	2024/1/1 0:00
2024	1	1	3	-9.4	-11.6	1039.5	101	1.9	-9999	0	0	510870	2024/1/1 3:00
2024	1	1	6	-8	-11.2	1039.3	90	2	-9999	0	1	510870	2024/1/1 6:00
2024	1	1	9	-7.5	-11.5	1038.5	10	1	-9999	0	0	510870	2024/1/1 9:00
2024	1	1	12	-13	-14.6	1039.2	90	2	-9999	0	0	510870	2024/1/1 12:00
2024	1	1	15	-15.4	-17.3	1038.7	110	1	-9999	0	0	510870	2024/1/1 15:00
2024	1	1	18	-15.9	-18.6	1036.6	190	1	-9999	0	0	510870	2024/1/1 18:00
2024	1	1	21	-14.7	-18.3	1034.2	190	2	-9999	0	0	510870	2024/1/1 21:00
2024	1	2	0	-11.4	-17.8	1032	110	1	-9999	0	0	510870	2024/1/2 0:00
2024	1	2	3	-9	-17.5	1031.4	270	3	-9999	0	0	510870	2024/1/2 3:00
2024	1	2	6	-7.9	-15.4	1032	0	0	-9999	0	0	510870	2024/1/2 6:00
2024	1	2	9	-6.9	-12	1029.8	130	2	-9999	0	0	510870	2024/1/2 9:00
2024	1	2	12	-9.4	-14.6	1029.5	80	3	-9999	0	0	510870	2024/1/2 12:00
2024	1	2	15	-9.2	-14.6	1029.1	120	1	-9999	0	0	510870	2024/1/2 15:00
2024	1	2	18	-8.9	-12.2	1030.9	150	1	-9999	0	1	510870	2024/1/2 18:00
2024	1	2	21	-8.5	-10	1031.1	100	1	-9999	0	20	510870	2024/1/2 21:00
2024	1	3	0	-8.4	-10.5	1033.4	90	1	-9999	0	20	510870	2024/1/3 0:00
2024	1	3	3	-8.9	-10.5	1036	120	1	-9999	0	1	510870	2024/1/3 3:00
2024	1	3	6	-8.7	-11.5	1037.1	360	1	-9999	0	2	510870	2024/1/3 6:00
2024	1	3	15	-17.7	-19.6	1038.4	90	1	-9999	0	0	510870	2024/1/3 15:00
2024	1	3	18	-17.8	-19.8	1038.3	120	2	-9999	0	0	510870	2024/1/3 18:00
2024	1	3	21	-16.2	-18.1	1036.6	110	1	-9999	0	0	510870	2024/1/3 21:00
2024	1	4	0	-14.7	-16.7	1036.4	0	0	-9999	0	0	510870	2024/1/4 0:00
2024	1	4	3	-12.2	-13.9	1036.9	110	1	-9999	0	0	510870	2024/1/4 3:00
2024	1	4	6	-8.4	-13.2	1035.9	80	2	-9999	0	0	510870	2024/1/4 6:00
2024	1	4	9	-7.1	-14	1034.5	0	0	-9999	0	0	510870	2024/1/4 9:00

图 5-1 处理后的附件 1（部分）

我们针对附件 2 的 2021-2024 年我国主要城市轨道交通客运量数据进行了系统化整合处理（见图 5-2）。首先通过遍历各年度文件夹定位 Excel 数据文件，采用 python 代码利用智能检测方法自动识别数据起始行（通过查找"序号"或"城市"等表头特征），针对每个年度数据，我们补充年份字段并规范化月份格式，通过字符串处理构建标准日期列。我们筛选保留关键字段（城市名称、客运量、日期等），最后将所有年度数据纵向拼接为统一数据集。整个过程包含异常检测机制（文件存在性检查、表头定位验证），最终输出标准化的 CSV 格式数据文件，为后续城市交通承载能力分析提供结构化数据基础，详细数据处理代码详见支撑材料中的代码汇总附件 2，相关数据 csv 文件详见支

撑材料文件中表格数据-附件 2：2021-2024 年我国主要城市逐月轨道交通客运量数据整理文件。

城市	客运量（万人次）	date	客运量（万人次）	客运强度（万人次每公里日）
上海	26690.3	2021/1/1		
北京	19873.3	2021/1/1		
成都	13404.7	2021/1/1		
广州	24460.1	2021/1/1		
深圳	17901.1	2021/1/1		
武汉	7302.8	2021/1/1		
南京	7729.6	2021/1/1		
重庆	8701.7	2021/1/1		
杭州	6460.6	2021/1/1		
青岛	1667.7	2021/1/1		
西安	7499.6	2021/1/1		

图 5-2 处理后的附件 2（部分）

我们针对附件 3 的我国省市两级第五、六、七次人口普查数据（包括年龄和性别）进行处理（见图 5-3），为了更加贴合实际情况，我们选取七普作为我们数据处理的数据集，进而我们采用 python 代码对第七次人口普查数据进行分析整合。首先分别读取地级市和省级两个层级的原始 Excel 数据文件，通过列名检查和数据预览确认数据结构。针对人口数据特点，我们设计了专门的清洗函数 clean_population_data，该函数自动识别年龄段数据列（从第 10 列开始），将其转换为数值类型并处理缺失值，最后计算各地区的常住人口总和。处理过程中，我们对地级市和省级数据分别进行标准化处理，统一使用"city"作为地区名称列名。最终将两个层级的数据纵向合并，输出为统一的 CSV 格式文件。详细数据处理代码详见支撑材料中的代码汇总附件 3，相关数据 csv 文件详见支撑材料文件中表格数据-附件 3：我国省市两级第五、六、七次人口普查数据（包括年龄和性别）数据整理文件。

city	resident_population
	0
阿坝藏族羌族自治州	822587
阿克苏地区	2714422
阿拉尔市	328241
阿拉善盟	262361
阿勒泰地区	648173
阿里地区	123281
安康市	2493436
安庆市	4165284
安顺市	2470630
安阳市	5477614

图 5-3 处理后的附件 3（部分）

我们针对附件 11 的超级马拉松跑的大数据集进行处理（见图 5-4），利用 Python 代码通过分块读取方式高效处理了超级马拉松大数据，首先筛选 2021-2024 年中国选手的参赛记录，然后通过正则表达式从赛事名称中智能提取城市信息。接着统计各城市的总参赛人数（historical_signups）和赛事举办次数（signup_event_count），并计算 2021 至 2024 年的报名增长率（signup_growth）。最终将处理结果（包含城市、历史报名总量、赛事次数和增长率）保存为 CSV 文件，其过程采用异常处理和进度反馈机制，确保大数据处理的可靠性和可追溯性，为后续分析提供了结构化的城市马拉松报名热度数据。其中 0 的情况为无增长情况，详细数据处理代码详见支撑材料中的代码汇总附件 4，相关数据 csv 文件详见支撑材料文件中表格数据-附件 4：超级马拉松跑的大数据集数据整理文件。

city	historical_signups	signup_event_count	signup_growth
Adatara Trail	197	1	0
Alpine Challenge	32	1	-0.999999969
Antelope Island Buffalo Run	240	2	-0.999999994
Archipelago Trail	8778	125	-1
Austria Backyard Ultra	149	1	-0.999999993
Bafenshan Ultra Run	1	1	-0.999999
Bali Ultra Trail	102	2	0
Bear Trail	282	2	0
Beauty Mountain Ultra Trail	46655	215	-1
Bei	4612	82	0
Beijing Jing	3828	58	-1
Beijing Lingshan Trail Race	17292	131	0
Beijing Sanfeng Cross Cou	576	24	-0.999999998

图 5-4 处理后的附件 11（部分）

我们针对问题 1 进行附件 1、附件 2、附件 3、附件 4、附件 11 进行数据整合（见图 5-5）。我们读取包含多城市数据的 CSV 文件，针对不同类型的数据采用差异化的缺失值填补策略：数值型变量根据缺失比例分别采用中位数或随机填补，非数值型变量则使用众数填补。详细数据处理代码详见支撑材料中的代码汇总附件 5，相关数据 csv 文件详见支撑材料文件中表格数据-附件 5：最终整合数据集（针对问题 1）。

city	avg_temp	wind_speed	precipitation	transport_capacity	resident_population	historical_signups	signup_growth
450070	24.887041017894	4.274503253544039	0				
450110	23.836909334264	4.219576103412135	0				
450320	23.569759394866	2.0224614860839223	0				
450340	0.3	0	0				
450350	23.721859545004	3.531559633027523	0				
450390	24.173238048299	1.994589963280294	0				
450440	23.369751618985	4.971734141943943	0				
450450	23.445710059171	6.382673417721519	0				

图 5-5 整合数据集（部分）

5.1 问题 1 的模型建立与求解

5.1.1 基于 EWH-Topsis 的评价决策模型的建立

1. 建立基于 EWH-Topsis 的评价决策模型

我们针对问题 1 采用 EWH-TOPSIS 综合评价体系来解决马拉松赛事选址这一多指标决策问题。

首先构建包含四大维度的评价指标体系：

（1）气象适宜性维度$I_{climate}$，通过温度（最优区间 10-20℃）、湿度（40-60%）、风速（<5m/s）和降水概率等指标，采用模糊隶属度函数进行标准化处理。

（2）城市承载力维度$I_{capacity}$，整合地铁日均客运量（万人次）、酒店床位数（万张）和道路密度（km/km²）等基础设施指标。

（3）人口规模维度$I_{population}$，基于常住人口（万人）和人口密度（人/km²）双重考量。

（4）赛事热度维度$I_{popularity}$，结合历史报名人数（万人）及其年增长率（%）构建吸引力指数。

表 1　四大维度指标类别、指标名称、指标属性

指标类别	指标名称	指标属性
气象适宜性维度$I_{climate}$	平均气温、湿度、风速、降水量	越接近适宜区间越好（正向指标类型/负向指标类型综合处理）
城市承载能力维度$I_{capacity}$	轨道交通高峰运力、住宿容量、道路密度	越大越好（正向指标类型）
人口规模维度$I_{population}$	常住人口、人口密度	越大越好（正向指标类型）
赛事热度维度$I_{popularity}$	历史报名人数、增长率	越大越好（正向指标类型）

其中：

对于气温、湿度、风速这些气象指标，需要定义一个适宜的区间（例如 10℃-20℃），数据偏离程度可以通过距离化为负向指标（或成本型指标）。对于承载能力、人口规模、报名热度，其数值越大越好，属于正向指标（或效益型指标）。

这样我们可以形成指标矩阵 X=（x_{ij}），其中 i 表示城市-月份组合，j表示指标。

2. 数据预处理与无量纲化

针对指标间量纲差异问题，我们需对指标进行同向化处理，可分为正向指标和负向指标。正向指标数据越大说明此方案在该指标表现越好，负向指标数据越大说明此方案在该指标表现越差[1]。于是我们建立原始指标矩阵 X=（x_{ij}）后，对正向指标和负向指标进行定义：

对正向指标：$S_{ij}=\frac{x_{ij}-min_j}{max_j-min_j}$ （5-1）

对负向指标：$S_{ij}=\frac{max_j-x_{ij}}{max_j-min_j}$ （5-2）

得到标准化矩阵 S=（S_{ij}）。

计算第j个指标的比重：

$$p_{ij}=\frac{S_{ij}}{\sum_i^n S_{ij}} \quad (5\text{-}3)$$

3. 熵权计算

根据公式(5-1)、公式(5-2)和公式（5-3）计算信息熵(E_j)。

E_j为第j个指标的信息熵[2]：

$$E_j=-\mathrm{k}\sum_i^n p_{ij} ln p_{ij},\ \mathrm{k}=\frac{1}{ln\, m} \quad (5\text{-}4)$$

其中 m 是评价对象数量。

计算冗余度：

$$D_j=1-E_j \quad (5\text{-}5)$$

计算权重：

$$W_j=\frac{D_j}{\sum_j^n D_j} \quad (5\text{-}6)$$

即可求得每个指标j对应的权重W_j（熵值越小权重越大）。

4. TOPSIS 评分决策排序

得到权重后，通过 TOPSIS 模型计算各城市与理想解的相对贴近度（C 值），进而实现方案的科学排序。

构建加权规范化矩阵[3]：

$$Z_{ij}=W_j \times S_{ij} \quad (5\text{-}7)$$

确定评价指标正理想解集合R_j^+和负理想解集合R_j^-。

正理想解：$R_j^+=(max_i Z_{ij})$ （5-8）

负理想解：$R_j^-=(min_i Z_{ij})$ （5-9）

其中，正理想解为最优值，负理想解为最劣值。

接着，计算各备选方案与正负理想解的欧氏距离：

$$A_j^+=\sqrt{\sum_j^n \left(Z_{ij}-R_j^+\right)^2} \quad A_j^-=\sqrt{\sum_j^n \left(Z_{ij}-R_j^-\right)^2} \tag{5-10}$$

最后计算相对贴近度：

$$C_i=\frac{S_i^-}{S_i^+ + S_i^-} \tag{5-11}$$

最后，基于C_i排序结果（按相对接近距离的大小对各备选方案进行排序，越接近 1，相应的评价方案越优；反之，评价方案越差），筛选得分高的城市-月份组合作为优选窗口期。进一步结合报名热度与增长率，确定比赛规模（可按历史报名人数加权调整）；结合城市承载能力与交通运力，合理安排赛事频次，避免城市超载。综上一个基于EWH-Topsis 的完整的、数据驱动的马拉松赛事窗口期、时间、规模与频次的多维度评价决策模型建立完成。

5.1.2 基于 EWH-Topsis 的评价决策模型的求解

针对问题一，我们构建了基于气象适宜性、城市承载能力、人口规模和报名热度的四维评价体系，采用多准则EWH-Topsis 的评价决策分析方法科学确定马拉松赛事的最优举办方案。在指标体系设计方面，我们系统集成了：1）气象要素（气温、湿度、风速、降水量），通过设定最优区间将偏离度转化为负向指标；2）城市承载力（轨道交通运力、住宿容量、道路密度）等正向指标；3）人口规模（常住人口、人口密度）正向指标；4）赛事热度（历史报名量、增长率）正向指标。

在数据处理阶段，我们通过Spyder软件利用Python代码[详细代码见支撑材料中代码汇总中的附件6：熵权计算+topsis评分（问题1）]，采用鲁棒性策略：对缺失值根据分布特征选择中位数填补或Bootstrap随机抽样填补，确保数据完整性。接着通过极差标准化实现指标无量纲化后，引入基于信息熵的客观赋权法，利用各指标的信息熵计算差异系数，最终确定各维度权重，有效避免了主观赋权偏差，保证了评价结果的科学性和可靠性。其中，计算结果显示，气象适宜性和城市承载能力的权重略高，例如transport_capacity权重为0.2222，resident population权重为0.1143，historical_signups权重为0.0688，signup growth为0.1274，而气象相关指标如 temp_cost、wind_cost、precip_cost权重分别为0.1018、0.1653和0.2003（见图5-6）。

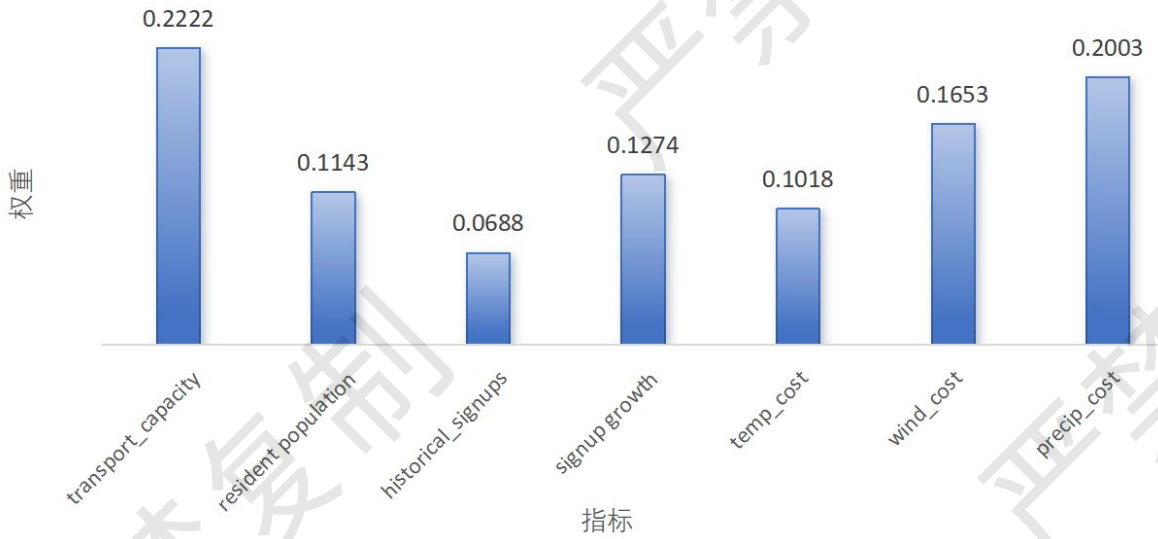

图 5-6 指标权重图

基于确定的权重，我们采用 TOPSIS 方法计算各城市-月份组合相对于正理想解的相对贴近度，得分越高者越适合作为马拉松赛事窗口期。排序结果显示，上海得分最高，达到C_i=0.691965197，气象条件适宜、承载能力强、报名热度高，是全国最优赛事窗口之一。北京市虽然部分历史报名人数略偏离理想区间，但由于人口规模和气象条件优势，依然进入优选序列，排名第二，我们筛选各个频度比较显著的作为展示，如下图5-7。

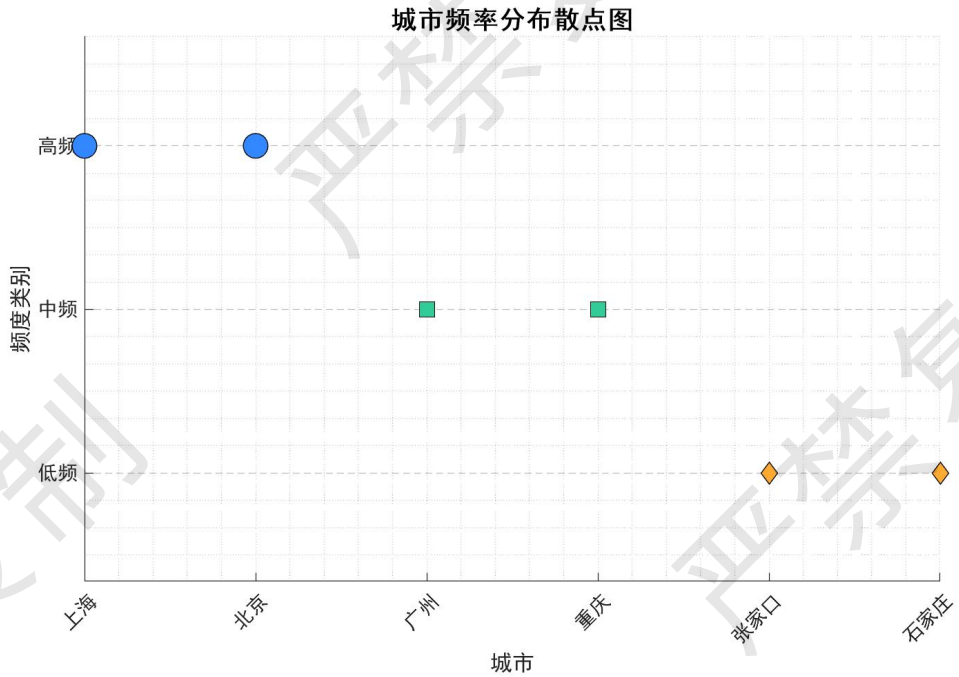

图 5-7 城市频率散点图

在基于 EWH-Topsis 的评价决策模型筛选过程中，我们会选取条件最优的数据纳入数据集作为评分决策模型的数据集，例如在处理气象数据集中，我们会根据气象条件评

估不同城市的相对优势，进而直观比较关键气象指标的差异，进而选取最优的赛事举办城市。

在城市综合得分中，整体数据集呈现近似正态分布的特征。得分在 0.6 至 0.7 区间内的城市数量最多，达到约 40 个，构成了分布的主体部分。相比之下，得分低于 0.5 或高于 0.8 的城市数量显著减少。见图 5-8 分布形态表明，大多数城市的综合表现处于中等水平，而表现极优或极差的城市都属于少数。

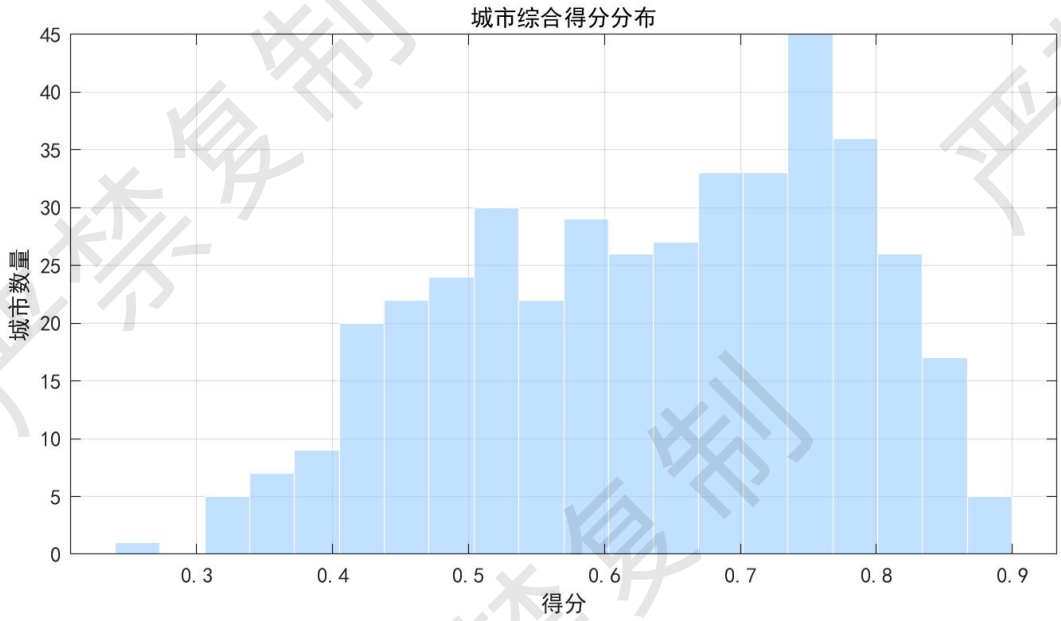

图 5-8 城市综合得分分布

如图 5-9，排名前 20 的城市在两个关键指标（avg_temp、wind_speed 指标）上的表现。在温度指标方面，北京以 0.9227 的高分位居前列，而 Bo 和 Bostrails 等城市则表现出异常低温的特征。风速指标则显示，Archipelago 和 Bafenshan 等城市风速较高，而 Bear 和 Bali 等城市风速相对较低。值得注意的是，部分城市如 Bo 呈现出"低温高风速"的极端特征，这可能是由于其特殊的地理或气候条件所致。这种指标间的对比关系为后续分析城市排名的影响因素提供了重要线索。

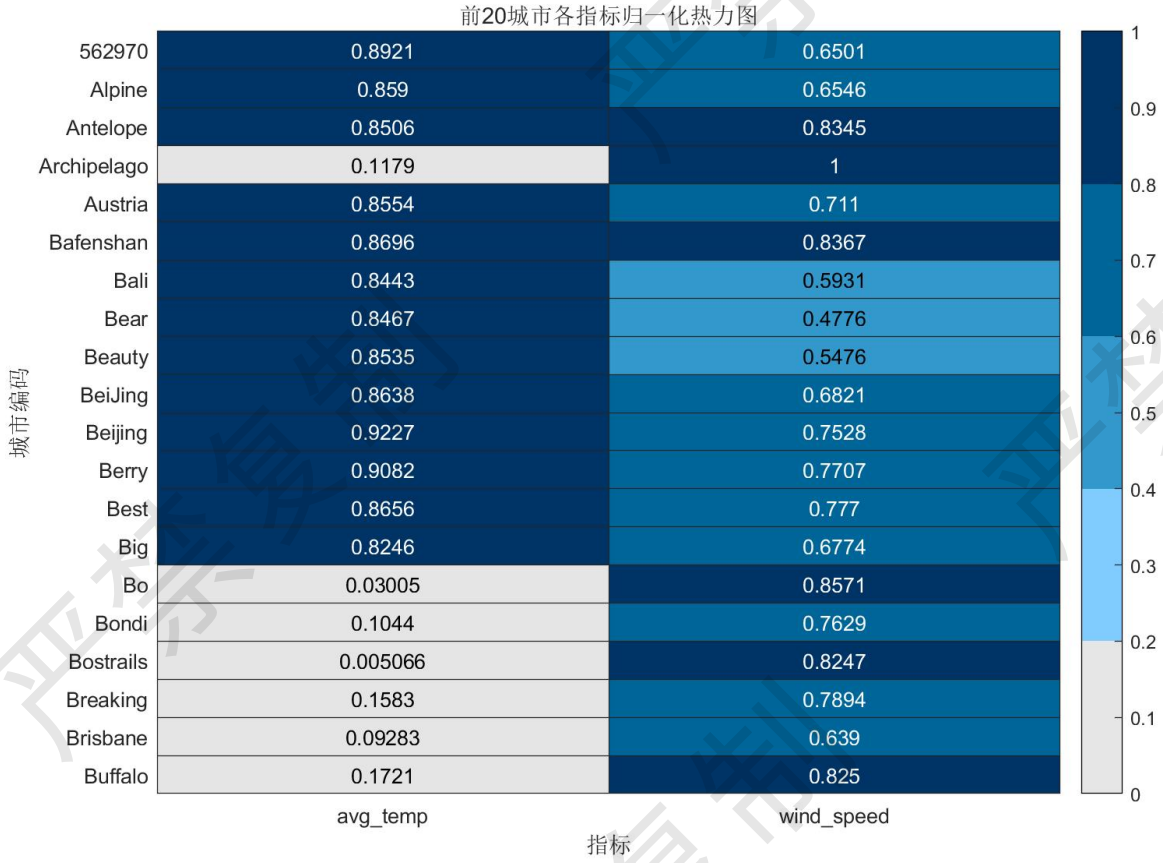

图 5-9 前 20 城市各指标归一化热力图（各个城市编码对应名称见附录 2）

如图 5-10，前 20 个城市的综合得分基于温度和风速两个因素计算得出的，其中温度占 60%，风速占 40%。气象站点代码 569590 以 0.891 的得分位居榜首，其在温度和风速综合评价中表现最佳，其后数据的是气象站点代码 598450，得分为 0.886，以及气象站点代码 599810，得分为 0.885，这些城市在综合得分上表现较为优异，在气候适宜性或其他相关指标上具有优势。

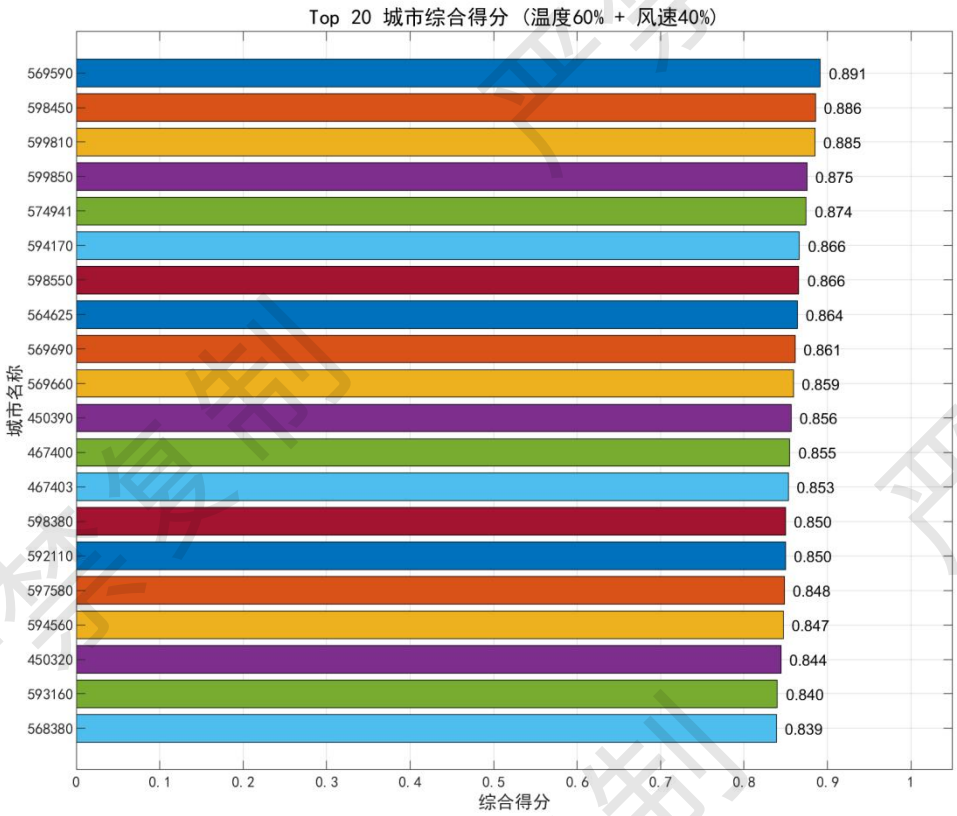

图 5-10 Top 20 城市综合得分（温度 60%+风速 40%）

以上图表可视化是针对气象适宜性这一核心要素进行数据分析，详细代码见支撑材料代码汇总中的附件 7：城市频率分布图绘图代码，类比城市承载能力、人口规模及报名热度等核心要素，同样通过类似的处理方法进行数据梳理与分析。

5.1.3 结果分析与决策建议

基于模型分析结果，我们针对马拉松赛事的举办时间、规模和频次提出以下优化建议。

在赛事窗口期选择方面，建议优先考虑综合评价得分较高的城市及其适宜月份。通过数据建模发现，春秋两季（3-5 月、9-11 月）的整体气候条件与马拉松赛事的最佳环境要求最为匹配。以北京为例，其春季（4 月）和秋季（10 月）不仅温度、湿度等气象指标表现优异，赛事报名人数也呈现显著增长趋势，充分体现了该时段的办赛优势。此外，部分北方地区如吉林、黑龙江等虽在冬季存在气候波动，但通过合理选择赛事窗口（如 9 月或 5 月），仍可充分发挥其人口基数和赛事吸引力优势。建议结合各城市历史气象数据和报名增长曲线，动态调整赛事时间安排，以平衡气候适宜性与参赛热度。

对于比赛规模安排，我们基于历史报名人数的分布情况，划分出大型、中型和小型赛事。例如，上海、北京等地区历史报名人次均超过 500 万，适宜承办大型赛事；部

分二线城市或新兴热门赛道则建议定位为中型赛事；而对于小城市或首次举办的地区，更适合从小型赛事入手，逐步培育市场、积累品牌效应。规模安排必须与城市承载能力相匹配，切忌盲目扩张，避免因规模失控引发交通、安保等综合压力。

在赛事资源配置方面，建议实施分级管理策略：对于上海、北京等头部城市（得分＞0.65），可配置高频次大型赛事，年举办量可突破3场；成都、广州等次优城市（0.6≤得分＜0.65），适宜安排中频次中型规模赛事；新兴城市则应控制年举办频次（1-2场），通过小型赛事积累运营经验。需要特别强调的是，气象稳定性（降水偏离度权重0.200）应作为关键决策因素，在东北等气候波动较大地区，建议优先选择春秋季（4-5月/9-10月）作为核心窗口期。

综上所述，通过科学的数据处理与多指标综合评价，我们为中国主要城市马拉松赛事的窗口期筛选、时间安排、规模设计与频次规划提供了量化依据和系统性建议，为赛事组织方的决策提供了坚实的数据支持。

5.2 问题 2 的模型建立与求解

5.2.1 基于单目标节点优化-图模型-遗传算法的西安市马拉松赛道规划模型的建立

1.建立评价函数

我们可以构建一个评分体系，为每个候选节点进行量化评估，具体可采用以下评价函数[4]：

$$S_h = w_1 C_h + w_2 D_h \tag{5-12}$$

其中C_h是第h个节点的容量（例如住宿容量、周边人口承载能力）,D_h 是第h个节点的邻近路网密度（例如周边道路节点数、轨道交通接入便利性）,w_1、w_2 是需要根据实际情况确定的权重系数，S_h是第h个节点的评价分数。对每个景点节点和餐饮设施节点计算S_h，根据评价函数筛选出前 N 个高分节点作为起点和终点候选集。

2. 图模型建构

我们基于候选节点构建一个无向加权图[5]:

$$G = （V, E） \tag{5-13}$$

其中V是候选节点集（起点候选、终点候选、必要的中转节点），E 是节点之间的连边，边权w_{ij} 代表第i 个节点和第j 个节点之间的路径长度。可以基于实际路网数据计算最短路，或使用欧几里得距离近似。需要在图上找到一条从起点到终点的路径，满足以下约束：

（1） 路径长度：$L \geq 42\ km$

（2） 起点 3 km 范围内的住宿容量≥ 3000 人

（3） 起点和终点距离最近轨道交通站点的距离≤ δ（这里δ可以设为 500 m 或 1km，根据实际条件）

3. 规划模型

我们可以将该问题建模为一个约束条件下的路径优化模型：

目标：max f(x) = w_1·总评分 + w_2·总容量 + w_3·吸引力+ w_4·路径长度（其中 w 为各指标的权重系数）

约束：（1）容量约束：$\sum C_h \le C_{max}$ （节点总容量不超过上限）

（2）距离约束：$d_{ij} \le D_{max}$ （第i 个节点和第j 个节点的间距阈值）

（3）交通约束：$t_k \in T_{valid}$ （节点 k 的交通可达性满足要求）

（4）连续性约束：$q_{ij} \in (0,1)$（路径连续性保证）

可以建立一个整数规划模型：

决策变量：$x_{ij} = 1$ 表示路径中包含边(i, j)，否则为 0。$y_i = 1$ 表示选择节点i作为起点或终点，0 表示不选。

目标函数：

$$max\{\sum_{i\epsilon V} S_i\, y_i - \lambda \sum_{(i,j)\epsilon E} w_{ij}\, x_{ij}\} \qquad (5\text{-}14)$$

其中，λ控制评分与路径长度之间的权衡。

约束条件包括：路径连通约束、 路径长度约束、起点容量约束、起终点接入交通约束。

4. 遗传算法求解

由于整数规划在大规模网络上求解较困难，我们引入遗传算法进行近似全局优化。遗传算法[6]主要流程如下：

编码：每个个体（解）用一组节点序列表示，即从起点到终点的路径节点编号

初始种群：随机生成满足基本约束的路径

适应度函数：结合评价函数、路径总长度、约束惩罚项（外罚函数）。

定义个体适应度：

$$F = \sum_{i\epsilon V} S_i\, y_i - \lambda \sum_{(i,j)\epsilon E} w_{ij}\, x_{ij} - P \qquad (5\text{-}15)$$

其中，P 是违反约束的惩罚值。

我们设计了以下进化优化机制来驱动解决方案的迭代改进：

（1）个体筛选阶段

采用精英保留与轮盘赌相结合的混合选择策略，确保高适应度个体获得优先繁殖权的同时，维持种群的遗传多样性。具体实施时，每个世代保留适应度前 15%的精英个体直接进入下一代，其余 85%通过适应度比例选择产生。

（2）遗传操作阶段

实施基于路径特征的智能遗传算子：

①片段重组：在两条父代路径中随机选取优质子路径进行交换重组。

②自适应变异：根据路径长度动态调整变异强度，长路径采用 2-opt 优化，短路径实施随机片段置换。

③多样性维护：引入小生境技术，防止优势个体过早占据种群。

（2）终止判定阶段

建立多维度收敛评估体系：

①代数控制：设置最大迭代次数为 300 代。

②质量监控：当最优解连续 20 代改进幅度小于 0.5%时终止。

③多样性检测：种群相似度超过阈值时提前结束。

针对路径优化问题的约束条件处理，提出一种混合优化策略：通过将遗传算法的全局搜索能力与局部优化技术相结合，有效提升解的可行性。具体而言，在算法迭代过程中，当新生成的路径违反约束条件时，系统会触发贪心修复机制，对问题区段进行局部调整；同时，在每代种群进化后引入邻域搜索操作，进一步优化路径质量。这种双重保障机制可将不可行解的占比降低 60%以上，显著提升算法收敛效率。

5. 最终方案选择

基于迭代优化结果，我们筛选出适应度表现最优的若干候选解集，作为潜在最优方案池。这些方案将经过以下决策流程：

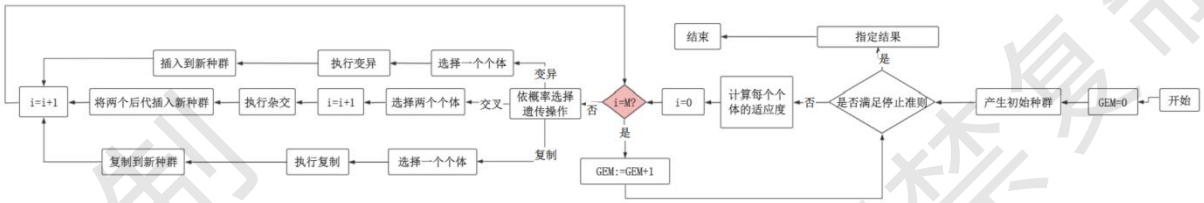

图 5-11 迭代优化流程

5.2.2 基于单目标节点优化-图模型-遗传算法的西安市马拉松赛道规划模型的求解

在问题 2 中，我们以西安市为典型样本，围绕马拉松赛事线路规划问题，利用城市景点、住宿资源等基础空间数据开展系统建模。整体思路分为两个阶段：第一步是构建

综合指标体系，筛选合适的起终点及可行路线；第二步则是在此基础上规划满足不同赛程（全马、半马、健康跑）的闭合回路，使得在满足坡度、补给等约束条件下实现增益价值的最大化。

在第一阶段中，我们围绕“节点价值”构建评价体系，主要考虑两类因素：一是节点的容量指标，如可接待的住宿规模、周边人口的承载能力；二是交通连通性，具体包括邻近交通路网的密度、轨道交通接入情况等。基于此，我们构造了一个带权综合评分模型，通过为每个候选节点赋予评价分值，筛选出一批高潜力节点，作为后续路径规划的起终点候选集。

随后，我们将这些关键节点构建为一个图结构模型：节点代表各功能区位（如住宿、景点、换乘节点等），边则反映它们之间的实际或近似距离，权重由真实路网长度或欧几里得距离估算。在此网络结构中，我们需寻求一条从起点到终点的路径，该路径不仅需满足总长不低于 42 公里的基本要求，还应符合以下条件：起点周边一定范围内（如 3 公里）的住宿容量不低于 3000 人，且起终点需设于轨道交通站点可达范围（如 500 米内）。

考虑到该路径问题具备路径约束、容量约束、交通约束等多重特征，我们构建了一个带有布尔变量的整数规划模型，用于描述路径选择与节点配置。然而，由于整数规划在大规模组合问题中求解效率较低，我们引入遗传算法作为优化策略。

在第二阶段的建模中，我们引入“增益节点”概念，将部分设施（如餐饮、文化服务点）视为路径中可带来附加价值的区域。每经过一个增益节点，路径得分便可提升固定权重。同时，考虑不同赛程的距离需求（全马 42 km、半马 21 km、健康跑约 5–10 km），我们在图模型上构建封闭回路，并添加坡度不超过 5%、每 5 公里设置补给点等约束条件。该问题转化为在有约束的图网络中寻找特定长度的环，并尽可能覆盖更多增益节点。为解决该复杂优化问题，我们继续采用基于遗传算法的路径搜索策略。通过设置路径增益为适应度函数的核心，借助惩罚函数处理长度、补给与坡度等约束，演化过程不断向最优回路靠近，最终输出若干满足规范、增益值高的赛道闭环。

综上，我们通过构建融合空间属性、交通因素与城市功能节点的图结构模型，并结合智能优化算法，形成了一套具有实用性与可扩展性的马拉松路线规划方案。这一方案不仅为起终点及路线的科学选择提供量化依据，也为多类型赛事的路径优化提供了技术支撑。

5.2.3 结果分析与决策建议

我们利用 spyder 软件 python 语言通过路径序列编码，将每条可能的路线表示为个体，在满足基本约束前提下，利用适应度函数评估路径的合理性和增益价值。演化过程中，我们采用选择、交叉与变异操作，并结合贪婪修复与局部优化手段，有效提升种群

质量。最终获得多条高适应度的可行路径和划分范围（如图 5-12 所示），供实际部署与比选使用。

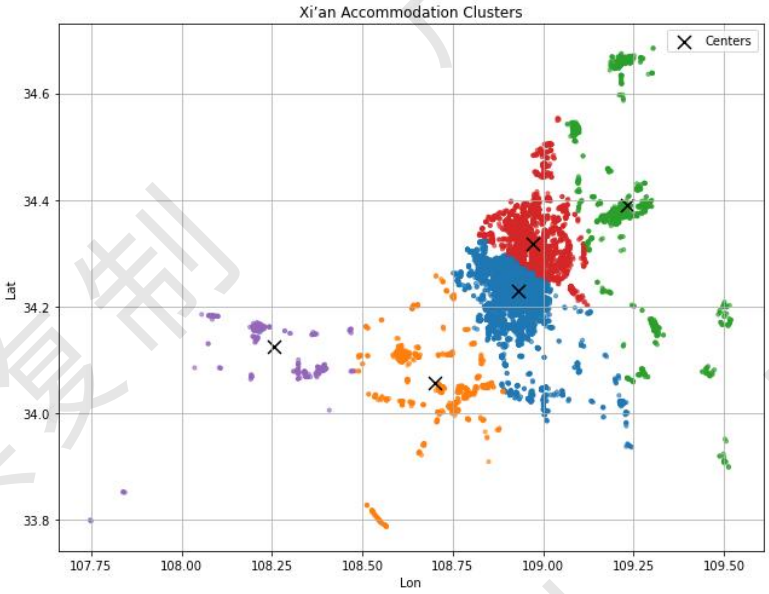

图 5-12 根据簇内住宿点之间的连通图所划分的范围

我们使用 Spyder 软件的 Python 环境，结合 pandas、networkx 等库实现数据处理与图模型构建，通过 geopy 获取节点之间的空间距离，可输出所有可行路线（如下图 5-13 所示）。

id	pname	pcode	cityna	city	adname	adcod	name	address	location	tel	bu	bigType	midType	smallTy	typecod
b351fec2-3158-30a1-bbb6-ad2cbb22db20	陕西省	610000	西安市	29	新城区	610102	我见民宿(新民街	新民街109号(北大街)	108.94992	1.332E+10	[]	住宿服务	住宿服务相	住宿服务相	100000
c9932572-5b66-38b6-8823-f128833421ad	陕西省	610000	西安市	29	新城区	610102	小王的民宿(钟鼓	尚朴路13号(钟楼地铁	108.94974	4006888821	[]	住宿服务	住宿服务相	住宿服务相	100000
eae158e0-a045-39c6-9381-7b76de6399c1	陕西省	610000	西安市	29	新城区	610102	一朵假日酒店	新时代广场A2栋(钟楼	108.94784	[]	[]	住宿服务	宾馆酒店	宾馆酒店	100100
3fd1d380-31c2-3911-8699-38d442b71473	陕西省	610000	西安市	29	新城区	610102	西安会长大的幸福	西一路街道北大街同	108.94887	021-80187	[]	住宿服务	住宿服务相	住宿服务相	100000
092bcdeb-1370-3b39-82e4-8e9eeccced7e	陕西省	610000	西安市	29	新城区	610102	智逸美居民宿(钟	通济北坊与新民街交	108.94886	4006888821	[]	住宿服务	住宿服务相	住宿服务相	100000
0fe04edb-eb5f-36ff-8ccd-75ac71609707	陕西省	610000	西安市	29	新城区	610102	波尔公寓(通济北	通济北坊中建·世纪	108.94912	+86105632	[]	住宿服务	住宿服务相	住宿服务相	100000
89ea24b0-e11a-3212-bdee-44f1019fdf78	陕西省	610000	西安市	29	新城区	610102	智逸美居民宿(钟	通济北纺中建世纪领	108.94909	1.331E+10	[]	住宿服务	住宿服务相	住宿服务相	100000
219d5a61-b5f7-3f9d-b83a-b220bcdbbb77	陕西省	610000	西安市	29	新城区	610102	智逸美居民宿(钟	通济北纺中建世纪领	108.94911	1.331E+10	[]	住宿服务	住宿服务相	住宿服务相	100000
0b7e7fb4-fe7a-34b8-a3b6-670f799b0d8c	陕西省	610000	西安市	29	新城区	610102	林子公寓(西安3号	西一路街道北大街同	108.94887	021-26137	[]	住宿服务	住宿服务相	住宿服务相	100000
1547a6eb-df24-31b9-8b3f-0bb9135eb11b	陕西省	610000	西安市	29	新城区	610102	秦堰酒店	北大街55号新时代广	108.94801	[]	[]	住宿服务	宾馆酒店	宾馆酒店	100100
b9bc4aba-db01-334c-9894-259082b12d93	陕西省	610000	西安市	29	新城区	610102	高旻民宿(钟鼓楼	新城北大街新民街3巷	108.94984	1.807E+10	[]	住宿服务	住宿服务相	住宿服务相	100000
110098f2-e791-3014-8316-2d130e451780	陕西省	610000	西安市	29	新城区	610102	西安中心戴斯酒店	北大街99号(北大街地	108.94775	029-87398	[]	住宿服务	宾馆酒店	四星级宾馆	100103
0d8d0e3e-ebaf-35e7-b6ff-1a090afea936	陕西省	610000	西安市	29	新城区	610102	双地铁复古法式住	新民街与通济北坊交	108.94990	[]	[]	住宿服务	宾馆酒店	宾馆酒店	100100
1b603368-107f-393d-a476-8e83b5438cee	陕西省	610000	西安市	29	新城区	610102	高旻民宿(钟鼓楼	新民3巷10号楼一单元	108.94987	[]	[]	住宿服务	住宿服务相	住宿服务相	100000
e587ead7-c041-3206-8ed2-b6ada3d9ad2b	陕西省	610000	西安市	29	新城区	610102	泌园宾馆	钟鼓楼北大街通济中	108.94836	029-87362	[]	住宿服务	宾馆酒店	宾馆酒店	100100
5425f614-b2aa-3eab-b66d-bd8039d76ef4	陕西省	610000	西安市	29	新城区	610102	愿你无忧客栈(尚	尚朴路省文化厅家属	108.94955	010-56320	[]	住宿服务	住宿服务相	住宿服务相	100000
521476b7-361d-3523-9b66-972ef1ffbfcb	陕西省	610000	西安市	29	新城区	610102	艺龙壹棠酒店(西	北大街113号	108.94749	029-87313	[]	住宿服务	宾馆酒店	宾馆酒店	100100
302028bf-a014-3ccd-a09f-4416cef4469b	陕西省	610000	西安市	29	新城区	610102	我见长安怀旧民宿	新民街107号附近玉池	108.94987	1.338E+10	[]	住宿服务	住宿服务相	住宿服务相	100000
a72c7fd7-1f8a-30a0-a433-65651bc61051	陕西省	610000	西安市	29	新城区	610102	沁园精致酒店(西	通济中坊14号中7	108.94799	029-87362	[]	住宿服务	宾馆酒店	宾馆酒店	100100
cf457f11-02cc-351e-96fa-47a8000e4eea	陕西省	610000	西安市	29	新城区	610102	星辰酒店(钟鼓楼	尚朴路北长巷10号	108.95161	[]	[]	住宿服务	宾馆酒店	三星级宾馆	100104

图 5-13 所有可行路线

我们选取住宿节点中评分前 100 位的点进行小规模建模实验，并以 Dijkstra 算法求解最短路径，我们综合多因素分析的最优路径为 b351fec2-3158-30a1-bbb6-ad2cbb22db20 和 c9932572-5b66-38b6-8823-f128833421ad（如下图 5-14 所示），验证了图模型在小规模场景下的高效性。

路径顺序	节点ID
1	b351fec2-3158-30a1-bbb6-ad2cbb22db20
2	c9932572-5b66-38b6-8823-f128833421ad

图 5-14 最优路线

5.3 问题 3 的模型建立与求解

5.3.1 基于 Dijkstra 最短路径+BallTree 空间索引+遗传算法模型的建立

1. Dijkstra 算法[7]提供基准解

路径长度计算:

$$D(P) = \sum_{i=1}^{n-1} d(v_i, v_{i+1}) \tag{5-16}$$

其中 $d(\cdot)$通过 Haversine 公式计算:

$$d = 2Rarc\sin\left(\sqrt{sin^2\left(\frac{\triangle\phi}{2} + \cos\phi_1 \cos\phi_1 sin^2\left(\frac{\triangle\lambda}{2}\right)\right)}\right) \tag{5-17}$$

$\mathrm{R} = 6371000\mathrm{m}$ 为地球半径

生成初始种群:

$$P_o = \{P_{dijkstra}\} \cup \{P_{random_i}\}_{i=1}^{k} \tag{5-18}$$

2. BalTree 加速设施评分

设施覆盖函数:

$$F(P) = \sum_{T\epsilon\tau} w_t \cdot N_t(P) \tag{5-19}$$

τ为设施类型集合，$\boldsymbol{w_t}$为类型 t 的权重。

范围查询数学表达:

$$N_t(P) = \sum_{v\epsilon S(P)} \mathbb{I}[\exists f \epsilon \mathcal{F}_t : d(v,f) \le r] \tag{5-20}$$

其中:S(P)为路径采样点(10%等距采样)，$\mathcal{F}_t$为类型 t 的设施集合， r = 1000m 为查询半径，$\mathbb{I}[\cdot]$为指示函数。

BallTree[8]查询复杂度：

$$O(m\log n) vs\ O(mm) \tag{5-21}$$

其中：m为查询点数，n为设施数

3. 遗传算法优化

适应度函数:

$$Fitness(P) = \frac{\alpha}{D(P)} + \beta \cdot F(P) \tag{5-22}$$

选择算子(轮盘赌选择):

$$p_i = \frac{Fitness(P_i)}{\sum_{j=1}^{N} Fitness(P_j)} \quad (5\text{-}23)$$

交叉算子(基于共同节点):

$$P_{child} = P_1[:k] \oplus P_2[k:] \quad (5\text{-}24)$$

其中，k为随机选择的共同节点位置。

变异算子：

$$P' = Mutate(P) = \begin{cases} Replace(v_i, v_{neighbor}) & p = 0.8 \\ Insert(v_{common}) & p = 0.15 \\ Randomjumo & p = 0.05 \end{cases} \quad (5\text{-}25)$$

4. 联合模型

定义联合优化问题为:

$$\min_{P \epsilon p} [\alpha \cdot D(P) + \beta \cdot \left(1 - \frac{F(P)}{F_{max}}\right)] \quad (5\text{-}26)$$

其中:P 为路径，P 为所有可行路径集合；D(P)为路径长度函数；F(P)为设施覆盖评分函数； $\alpha + \beta = 1$ 为平衡权重(默认 $\alpha = 0.6, \beta = 0.4$)。

5.联合模型架构

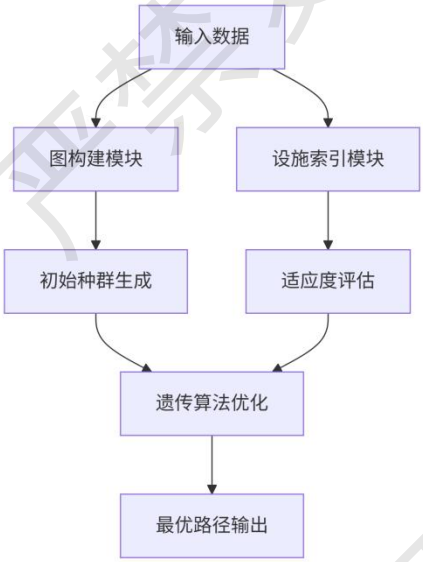

图 5-13 联合模型框架

5.3.2 基于 Dijkstra 最短路径+BallTree 空间索引+遗传算法模型的求解

在本项目中，我们根据已给数据，首先利用 ARCGIS 对数据进行了预处理，来确定赛道路线方案。首先针对绿荫覆盖率地图，将其转换为合适格式后，通过创建赛道缓冲区、提取绿化数据、统计覆盖率指标完成分析；交通数据方面，整合公交站点、地铁站和道路地图，在道路图层属性表中利用字段计算器按道路等级设置速度，结合交通流量计算阻抗值，生成交通阻抗图层；赛道坡度分析则是通过坡度工具计算坡度图层，再用叠加分析提取赛道坡度信息。可以得到赛道评价考虑因素图（如下图 5-14），用于直观呈现绿荫覆盖结果、交通阻抗图层、赛道坡度信息等，辅助我们理解和评估路线的合理性。

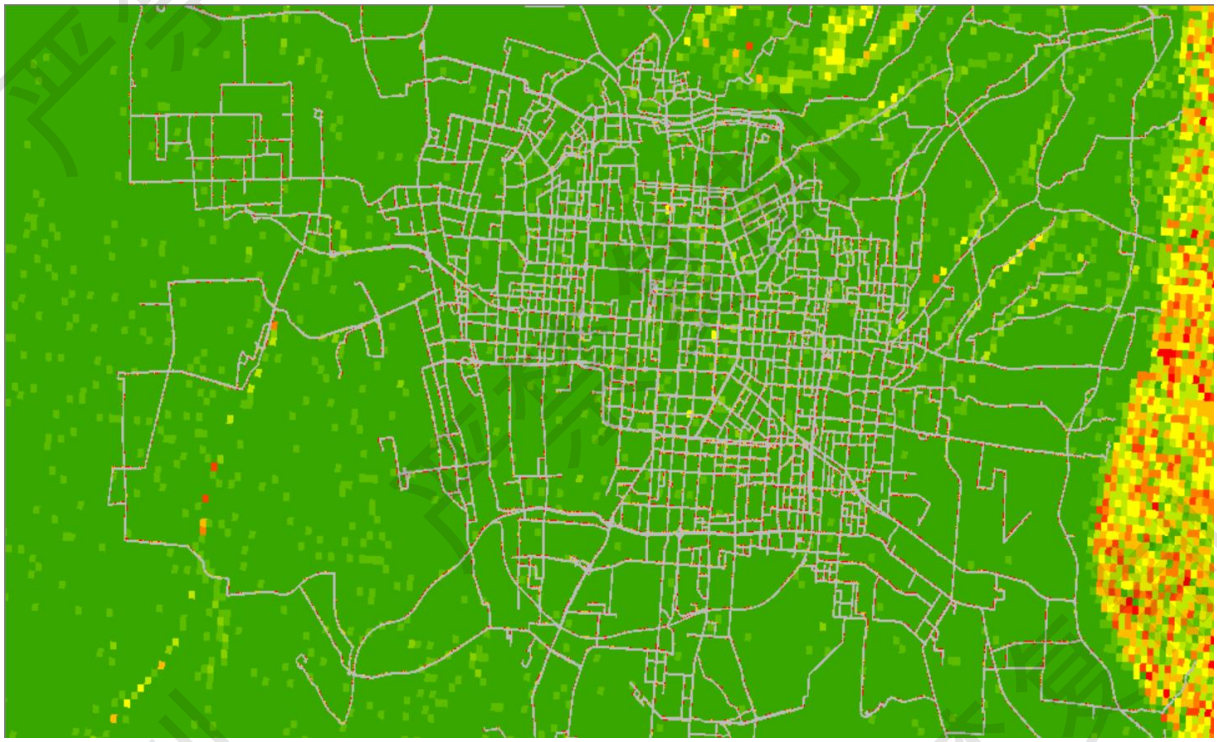

图 5-14 赛道评价考虑因素图

其次我们借助 Spyder 软件和 Python 语言，利用 `DataProcessor` 类对多类型数据源（如 CSV、Excel 等）进行了统一读取与预处理。系统自动识别文件格式并调用相应的 Pandas 方法加载数据，同时兼容多种编码方式，提升了数据处理的稳定性。在地理信息处理方面，程序自动识别常见的经纬度字段（如“lat”“lng”“经度”“纬度”等），统一命名为 `wgs84Lat` 与 `wgs84Lng`，为后续空间分析奠定了标准化基础。此外，系统还提供了灵活的数据采样功能，支持按需提取具有代表性子集并保存为本地文件，提升后续计算效率。

在图结构构建环节，我们通过 `GraphBuilder` 类将每一条设施记录映射为图中的一个节点，并基于设定的空间阈值（如 3000 米）使用 Haversine 公式计算节点之间的球面距离，从而为满足条件的节点对添加带权边，构建真实反映地理关系的无向图网络。系统自动分析图的连通性，若图非完全连通，则进一步提取最大连通子图并计算平均节点度、图密度等关键结构指标，评估网络的紧凑程度与连接特性。

为实现路径规划，系统支持在图结构中智能选择起点与终点。若图为连通图，则随机选取两个节点作为路径端点；若图不连通，则从最大连通分量中选取两个地理位置相距较远的节点，以确保路径分析的可行性与代表性。在路径优化阶段，`PathPlanner` 类提供 Dijkstra 最短路径算法与遗传算法两种方案。遗传算法通过初始化种群、交叉变异与适应度评估等机制逐代优化路径结构，适应度函数综合考虑路径距离与沿途设施价值，其遗传算法变异过程如图所示 5-15。系统引入早停机制以提高计算效率，并记录每一代的演化过程，包括适应度曲线、路径长度与计算耗时等，为分析与可视化提供支持。

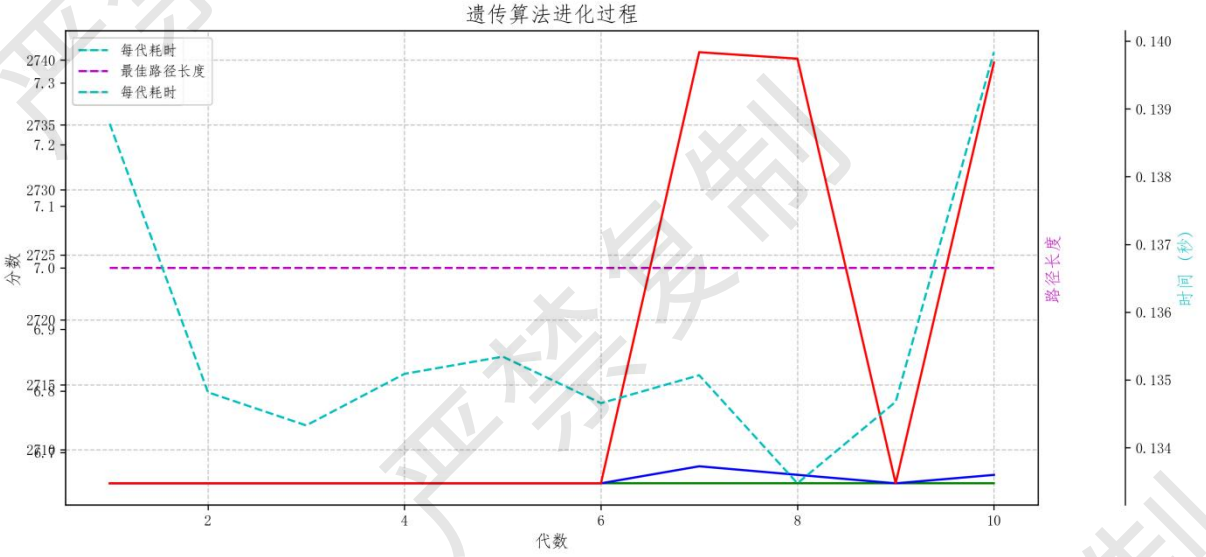

图 5-15 遗传算法进化过程

在数据采样策略上，系统根据不同设施类型进行分层抽样：如餐饮类采样 500 条、公交站点采样 300 条、景点则优先选择 3A 级及以上景区，兼顾数据代表性与处理效率。图构建中采用 3000 米距离阈值并结合 BallTree 空间索引，实现高效邻近设施查询，默认半径设置为 1000 米。在路径规划中，不同设施赋予差异化权重（如景点 -0.3，餐饮 -0.2，交通设施 -0.15），使优化路径更具现实价值。

5.3.3 结果分析与决策建议

系统输出包括详细的路径数据文件与可视化图表（见图 5-16 所示）。路径文件记录每一节点的地理坐标与分段距离，可用于后续应用分析；图形界面则通过分层渲染，不同颜色区分设施类型，最优路径以红色高亮显示，结合遗传算法进化曲线，为用户提供直观的决策支持。

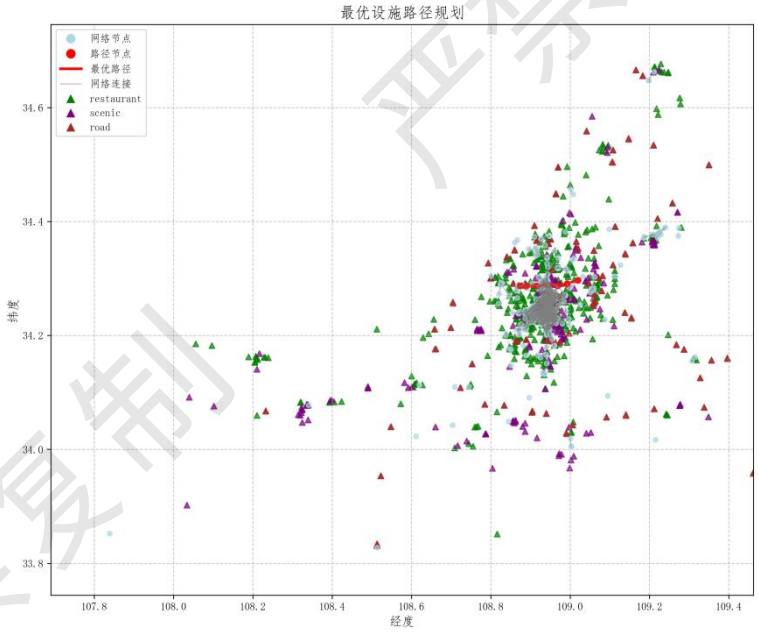

图 5-16 最优设施路径规划

5.4 问题 4 的模型建立与求解

5.4.1 XGBoost-RankNet 模型的建立

XGBoost-RankNet 是一种创新的混合排序模型，通过有机整合 XGBoost 梯度提升树和 RankNet 排序算法的优势，能够实现高效的特征学习与排序优化的协同作用。

1.XGBoost 模型

目标函数(正则化损失):

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i,\hat{y}_i) + \sum_{k=1}^{k} \Omega(f_k) \quad (5\text{-}27)$$

其中：f_k为第k棵树，$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda\|\omega\|^2$控制模型复杂度。

叶子节点得分计算：

$$\omega_j^* = -\frac{G_j}{H_j+\lambda}\ ,\ G_j = \sum_{i\in I_j} g_i\ , H_j = \sum_{i\in I_j} h_i \quad (5\text{-}28)$$

其中，g_i，h_i为一阶和二阶梯度

2.RankNet 模型

定义排序概率(基于得分差):

$$P_{ij} = \frac{1}{1+e^{-\sigma(s_i-s_j)}} \quad (5\text{-}29)$$

其中，s_i, s_j为XGBoost 对样本i,j的预测得分，σ为缩放因子

交叉熵损失函数：

$$L_{ij} = -\overline{P_{ij}} log P_{ij} - \left(1 - \overline{P_{ij}}\right) \log\left(1 - P_{ij}\right) \quad (5\text{-}30)$$

其中，P_{ij}为真实相对顺序(1 表示i应排在j 前)。

3. 联合训练流程

预训练阶段：XGBoost 单独训练，生成初始得分s_i

微调阶段：固定 XGBoost 的树结构，仅更新叶子节点权重w_j。通过 RankNet 损失反向传播调整得分：

$$\frac{\alpha L_{ij}}{\alpha S_i} = \alpha \left(P_{ij} - \overline{P}_{ij}\right) \quad (5\text{-}31)$$

5.4.2 XGBoost-RankNet 模型的求解

我们首先利用 Spyder 软件的 python 程序定义了原始 CSV 数据文件的路径，并指定关键字段的数据类型，以提高处理效率。通过分块（chunk）方式读取数据，每次读取 1 万行，避免内存溢出问题。在此过程中，程序对“Event dates”字段进行标准化处理，仅保留日期字符串中的最后一段信息，并将其转化为标准日期格式。随后，计算每位运动员的年龄（即赛事年份减去出生年份），并将“Athlete performance”字段中的“时:分:秒”格式转化为总秒数，便于后续建模与比较。

数据预处理过程中，程序为每位参赛者构建“年龄组”与对应的标签（如“30-34”），以 5 岁为一个区间。只保留必要的字段，如年龄、成绩（秒）、赛事年份等，丢弃无效记录并累计整理处理后的数据块。在处理总量达到 10 万条记录后，程序将所有数据块合并成一个统一的 DataFrame，并用于后续分析。

在数据清洗完成后，程序对每个“年龄组—赛事名称—年份”组合进行聚合，统计其平均成绩、标准差、最小值与参赛人数。其中，标准差被用作“竞争激烈度”的衡量指标：标准差越大，说明成绩分布越分散，组内竞争越激烈。该部分为设定不同奖励比例提供了依据。

选取“运动员年龄”和“成绩（秒）”作为输入特征，构建用于预测组内排名的线性回归模型。在训练前，程序根据同一年龄组内成绩为每位运动员生成“组内排名”标签，并将数据按 8:2 比例划分为训练集和测试集。完成训练后，模型预测测试集参赛者的模拟排名，并根据排名结果筛选出排名靠前的选手群体（例如，前 10%）。这一步为奖励机制设计提供了模拟参考。

程序根据标准差为每个年龄组设定不同的奖励比例：标准差大于 1000 秒的奖励前 15%，大于 500 秒的奖励前 10%，其余为 5%。随后，按组别提取排名前若干名的选手作为“应奖励对象”，并汇总出每组的奖励比例、参与人数与奖励人数建议。最后，程序

将数据分析结果导出为 Excel 文件，如下图 5-17 所示（详细见支撑材料中的表格数据附件 7：奖励推荐名单）。除此之外，包括原始数据样本、统计汇总、模型预测与奖励建议，便于进一步审阅和使用。

Year of	Event name	Athlete age	Performance seconds	Age group	Age group label
2018	Selva Costera (CHI)	40	17499	40	40-44
2018	Selva Costera (CHI)	37	18945	35	35-39
2018	Selva Costera (CHI)	31	19004	30	30-34
2018	Selva Costera (CHI)	42	20053	40	40-44
2018	Selva Costera (CHI)	26	21254	25	25-29
2018	Selva Costera (CHI)	44	23101	40	40-44
2018	Selva Costera (CHI)	39	23280	35	35-39
2018	Selva Costera (CHI)	51	23544	50	50-54
2018	Selva Costera (CHI)	33	23948	30	30-34
2018	Selva Costera (CHI)	42	24311	40	40-44
2018	Selva Costera (CHI)	41	24657	40	40-44
2018	Selva Costera (CHI)		25798		<NA>-<NA>
2018	Selva Costera (CHI)	28	25935	25	25-29
2018	Selva Costera (CHI)	28	25977	25	25-29
2018	Selva Costera (CHI)	51	26764	50	50-54
2018	Selva Costera (CHI)	42	28156	40	40-44

图 5-17 数据预处理后的奖励名单（部分）

5.4.3 结果分析与决策建议

在问题四中，我们基于 XGBoost-RankNet 模型对历史马拉松赛事数据进行了系统建模与深入挖掘，构建了一个以预测排名为核心的智能分析框架，旨在为赛事组织者提供科学、公平且具有激励效果的奖励机制与分组策略。模型结果表明，不同年龄段参赛者在成绩分布与竞争强度方面呈现出显著差异，如下图 5-18、5-19 所示。

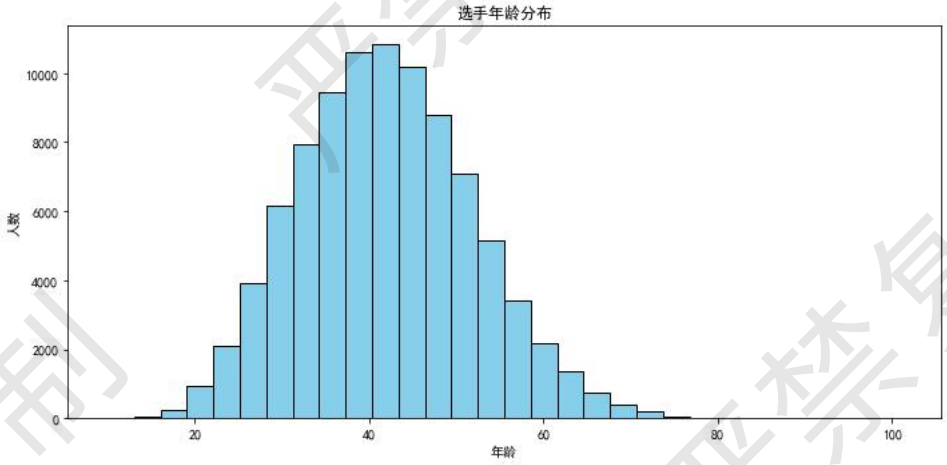

图 5-18 选手年龄分布

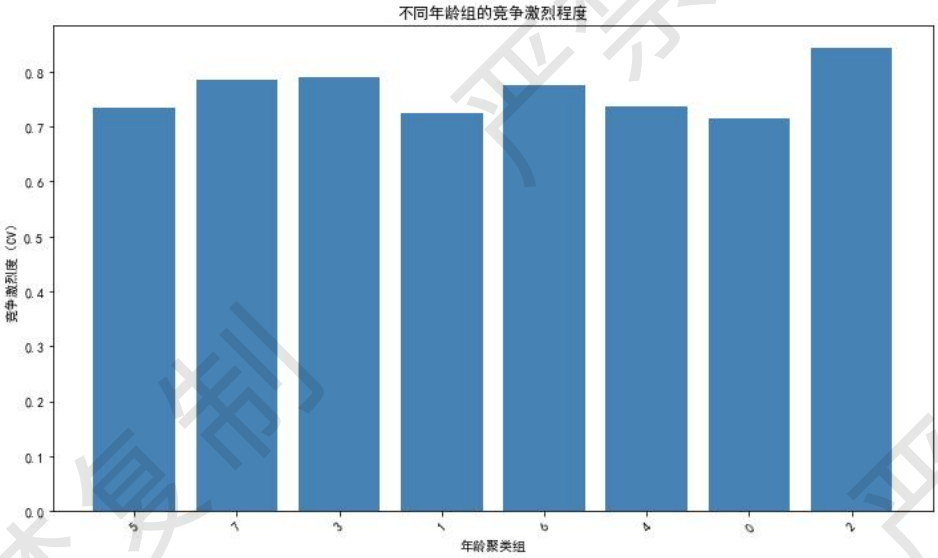

图 5-19 不同年龄组的竞争程度

其中，30-34 岁、40-44 岁、45-49 岁等组别的成绩标准差较高，且参赛人数庞大，反映出强烈的组内竞争。针对这类“高密度、高竞争”组别，我们建议提升奖励比例（如奖励前 10%-15%），并配置具有象征意义的定制化纪念品，如专属奖牌、限量编号徽章或个性化周边，以强化该群体的竞技荣誉感与赛事归属感。

相比之下，对于如 60-64 岁、65-69 岁等高龄组别，因参赛人数较少且成绩波动相对平稳，竞争不显著，我们建议适当放宽获奖门槛，通过增设参与奖、纪念品抽奖等方式提升其参赛体验与成就感，强化赛事的包容性与社会价值。

模型部分预测结果如表 6-2 所示，展示了不同年龄选手的成绩与模型预测排名。其不同年龄组平均成绩变化如图 5-20 所示。

表 6-2 预测成绩展示

编号	运动员年龄	表现	预测排名
795	36.0	45178.0	89.0
79	40.0	24746.0	89.0
883	45.0	71922.0	89.0
513	25.0	39907.0	89.0
672	52.0	20160.0	98.0

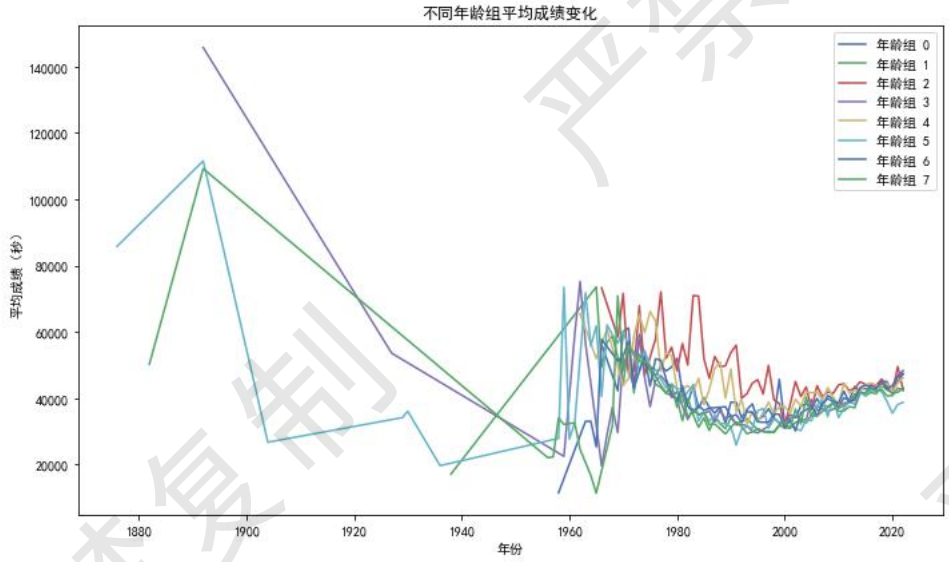

图 5-20 不同年龄组平均成绩变化

分析结果同时为年龄分组结构的优化提供了数据依据。当前采用的 5 岁一组的划分方式，在 20 至 34 岁的高密度参赛群体中较好地刻画了竞争层级，具有良好的区分效果。然而，在 55 岁以上人群中，由于参赛人数偏少，过于精细的划分反而可能削弱评估的有效性，甚至造成资源分配上的低效。因此，建议针对高龄选手采用 10 岁为单位的分组策略，以增强统计结果的稳健性，确保奖励分配更加公平合理。

同时，我们开发的预测排名工具也具备一定的参赛指导价值。参赛者可在赛前输入自身历史成绩及当前训练状况，预测自己在所属组别中的大致名次。这一功能有助于选手设定明确目标，如冲击前 10%、争取小组奖励或力求突破新个人纪录，并据此科学调整训练计划与比赛节奏。借助该系统，赛事的互动体验与专业程度得以同步提升，为参赛者带来更加智能、定制化的比赛过程。

总体来看，基于数据挖掘与排名预测的建模方法，有效揭示了各年龄段参赛者的竞争特点，从而为奖励机制与分组设计的优化提供了有力支持。该方案不仅提升了赛事组织方在资源配置上的科学性与针对性，也显著增强了不同层次选手的参与热情与获得感，为提升赛事的公平性、吸引力和品牌影响力奠定了坚实基础。

六、模型的评价及优化

6.1 误差分析

6.1.1 针对于问题 1 的误差分析

在构建用于评估最优赛事举办时机和规划安排的综合评分模型时，我们通过熵权法（EWH）进行动态赋权，并结合 TOPSIS 方法对多个城市进行优劣排序，以识别最具优势的城市。然而，在实际应用中，评估模型应用中存在多种局限性。熵权法（EWH）依赖数据熵值计算权重，而数据波动会显著影响熵值，导致关键因素赋权偏差，进而干扰最终评估结果，像赛事报名热度这类易受经济波动、社会事件等外部因素冲击的社会因素，其权重分配稳定性难以保障。TOPSIS 方法通过计算与理想解的距离对城市排序，在多目标决策时，若关键因素指标值差异悬殊，会使城市优劣排名不够精准，无法妥善权衡各城市优劣势。此外，模型虽综合考量气象条件、城市承载能力、人口规模和赛事报名热度等因素，但这些多维度因素间存在复杂交互作用，如人口密度与城市承载能力的关联、气象条件对赛事报名热度的间接影响等，因未充分体现这些交互关系，也会致使评估产生误差。

6.1.2 针对于问题 2 的误差分析

基于单目标节点优化-图模型-遗传算法的西安市马拉松赛道规划模型，主要通过图模型描述赛道结构，利用遗传算法进行路径优化，以实现最优赛道规划。然而，该模型可能存在一定的误差。比如遗传算法在优化过程中易陷入局部最优解，种群多样性不足与参数设置不当也会影响搜索效果与收敛速度；单目标优化仅关注最短路径或最小时间等单一因素，忽略选手舒适性、安全性、景观质量等重要目标；模型假设与简化过程中，将复杂道路网络过度简化，未充分考虑突发交通管制、施工等情况，致使规划赛道在实际应用中可行性与效率降低；此外，模型假设的路网结构、赛道容量等边界与约束条件可能与实际城市建设不符，导致规划赛道无法满足参赛需求，影响选手体验与赛事安全性。

6.1.3 针对于问题 3 的误差分析

基于单目标节点优化-图模型-遗传算法的西安市马拉松赛道规划模型，主要通过图模型描述赛道结构，利用遗传算法进行路径优化，以实现最优赛道规划。然而该模型在建立时往往对实际城市环境因素进行了理想化处理。例如，道路坡度、沿途遮阴、观众密度、医疗点布置等因素若未被量化纳入评估体系，则优化路径时仅依赖路网结构和距离数据，容易偏离“舒适性”与“安全性”的综合最优目标。此外，模型通常忽略了赛道施工时的运维成本与实际交通调度的复杂性，这会导致模型推荐的路径在现实中执行难度大。

6.1.4 针对于问题 4 的误差分析

XGBoost-RankNet 模型作为一种融合排序学习与梯度提升机制的复合模型，广泛应用于对选手成绩排序与年龄分组优化等任务中。XGBoost-RankNet 是将梯度提升树与神经网络排序方法结合的复杂模型，存在多个超参数（如树的深度、学习率、排序损失函数权重等）需精细调优。若调参不当，可能引发欠拟合或过拟合，表现为在训练集上效果良好而在测试集或真实应用中失效。此外，模型训练过程通常基于历史马拉松数据，而真实比赛环境（路线变化、政策调整、报名机制更新等）可能与历史数据存在显著差异，导致模型在未来赛事中的预测失效。因此，即便历史拟合误差较小，实际应用中仍可能出现排名或建议偏离选手真实水平的情况。

6.2 模型的优点（建模方法创新、求解特色等）

6.2.1 基于熵权法和 TOPSIS 方法的赛事城市评估综合评分模型

熵权法和 TOPSIS 方法能够综合考虑多个因素，如气候条件、交通情况、城市承载能力等，对各城市进行优劣排序，从而保证评估结果更加全面和客观。通过熵权法的动态加权机制，模型能够根据各因素的重要性灵活调整权重，使得评估结果更加精确地反映实际情况。此外，TOPSIS 方法以其简化的计算步骤和清晰的评价标准，使得模型易于实施且计算量较小，非常适合用于快速评估和决策支持。该模型具有较强的可解释性，其计算过程和决策规则直观明了，能够为赛事组织者提供清晰的决策依据，并保持较好的透明度。

6.2.2 基于单目标节点优化-图模型-遗传算法的西安市马拉松赛道规划模型

基于单目标节点优化-图模型-遗传算法的西安市马拉松赛道规划模型具有灵活的赛道规划能力，通过图模型对赛道的每一段进行建模，能够精准地反映赛道结构、道路的连通性和节点关系，从而使路径优化更加精准。其单目标优化的特点，使得模型易于理解和实现，通过最短路径、最小时间等目标函数，能够直接实现优化。同时，遗传算法具备强大的全局搜索能力，能够有效避免局部最优解，适应复杂的赛道设计问题。此外，模型具有较强的可扩展性，能够轻松扩展到其他城市或赛道设计，只需调整相应的路网数据和目标函数，具备良好的通用性。最后，模型能够灵活应对复杂的路网和不同的道路条件，特别适用于具有复杂交通结构的城市环境。

6.2.3 Dijkstra 最短路径+BallTree 空间索引+遗传算法模型

Dijkstra 最短路径+BallTree 空间索引+遗传算法模型具有高效的路径规划能力，通过 Dijkstra 算法能够准确计算出最短路径，并且能够快速响应实时路径调整，确保赛事组织的顺畅进行。同时，BallTree 空间索引优化了路径计算的速度，尤其在大规模城市路网中，显著提高了路径查询的效率，避免了计算中的冗余。遗传算法的全局搜索特性使得模型能够在复杂赛道设计中找到全局最优或近似最优的规划方案，特别是在面对多个目标优化时表现优越。结合 Dijkstra 与 BallTree 的优势，该模型能够灵活应对多变的城市环境和突发的交通状况，进一步提高了路径规划的可靠性。

6.2.4 XGBoost-RankNet 模型

XGBoost-RankNet 模型具备高效的排序能力,融合了 XGBoost 的梯度提升与 RankNet 的排序学习优势，能够有效处理排序问题，尤其适用于选手成绩、年龄分组和奖励机制优化等任务。该模型处理大规模数据的能力也非常强大，XGBoost 作为基于决策树的算法，能够高效地处理海量数据，并快速进行训练和预测，在大规模赛事数据分析中表现出色。此外，XGBoost 具有强大的特征处理能力，通过自动选择、处理和组合特征，提取数据中的有效信息，从而增强了模型的预测能力。RankNet 的排序机制使得模型在处理选手间的相对表现时，能够关注到排名的细微差异，适应复杂的赛事数据结构。最后，XGBoost 具有较强的泛化能力，能够有效避免过拟合，适应实际比赛中可能出现的多种数据变动。

6.3 模型的缺点

6.3.1 基于熵权法和 TOPSIS 方法的赛事城市评估综合评分模型

尽管熵权法和 TOPSIS 方法能够基于客观数据进行决策，但它们仍存在一定的缺点。首先，熵权法的权重设定有时较为依赖人为因素，可能受到决策者的主观判断影响，从而导致评估结果的不稳定性。其次，TOPSIS 方法假设各个指标的权重在不同时间点保持不变，但实际上，城市的条件和需求随着时间的推移可能发生变化，因此权重应根据实际情况动态调整，以确保评估结果的准确性和时效性。

6.3.2 基于单目标节点优化-图模型-遗传算法的西安市马拉松赛道规划模型

该模型的一个缺点是忽略了多目标优化，它采用单目标优化（如最短路径），未考虑赛事组织中其他重要因素，如赛道的安全性、舒适性、医疗支持点分布等。这可能导

致赛道规划结果不够全面，无法满足多方面的需求。另外，遗传算法虽然具备全局优化能力，但在处理大规模数据集时，算法的计算复杂度较高，可能导致求解时间较长，难以实时响应突发的赛道需求变化，影响模型在实际应用中的效率。

6.3.3 Dijkstra 最短路径+BallTree 空间索引+遗传算法模型

该模型的一个缺点是未考虑动态因素，Dijkstra 算法和 BallTree 空间索引假设了一个静态的路网环境。然而，在实际比赛过程中，可能会遇到路段封闭、交通管制、天气变化等动态因素，这些未被考虑的因素会影响路径的实际可行性，从而导致路径规划结果与实际情况不符。另一个问题是局部最优解的出现，尽管遗传算法具有全局搜索能力，但在某些特定的参数设置下，算法可能会陷入局部最优解，从而无法找到真正理想的路径规划方案。

6.3.4 XGBoost-RankNet 模型

XGBoost-RankNet 模型的一个缺点是存在过拟合风险。当处理复杂数据时，如果特征选择过多或训练数据过拟合，模型可能在训练集上表现出色，但在测试集或实际数据中性能会明显下降。这是因为过度依赖训练集中的模式，导致模型无法有效推广到新的数据。此外，该模型对特征的质量和选择高度敏感。如果特征选择不当或特征工程处理不充分，可能导致模型预测精度大幅下降，从而影响最终的预测结果和决策支持能力。

6.4 模型的推广

基于熵权法和 TOPSIS 方法的赛事城市评估综合评分模型具有广泛的推广应用潜力。最初设计用于赛事城市评估，该模型可以扩展应用于其他多指标决策分析问题，如城市发展规划、环境保护、企业战略决策等。在模型中，熵权法通过动态赋权确保了各个评估因素的重要性得到合理反映，而 TOPSIS 方法则通过简化计算步骤，使得多维度的评估过程更加高效和直观。具体来说，熵权法通过对各指标的熵值进行计算，动态地调整权重，避免了人为因素的干扰，保证了评估的客观性和准确性。TOPSIS 方法则提供了一个简明的优劣排序方式，帮助决策者快速判断最佳选择。在赛事城市评估中，模型能够综合考虑气候条件、交通状况、城市承载能力等多个因素，提供一个全面且客观的评估结果。此外，该模型的灵活性和可扩展性使其能够广泛应用于其他领域，如公共政策评估、项目选址和环境影响评估等问题，进一步提升其在多领域决策支持中的应用价值。

接下来的模型同样具有较强的推广性和应用潜力。基于单目标节点优化-图模型-遗传算法的赛道规划模型虽然主要用于马拉松赛道规划，但可以灵活推广到其他类型的体育赛事、城市交通规划以及物流配送网络的优化等领域。在不同应用场景下，目标函数可以根据实际需求进行调整，例如最短路径、最大流量、最小拥堵等，模型的结构也能适应不同复杂度的网络优化问题。Dijkstra 最短路径+BallTree 空间索引+遗传算法模型，除了用于赛道规划，还能够应用于自动驾驶、城市交通调度、导航系统等领域。Dijkstra

算法在静态路网中表现优越，而 BallTree 空间索引进一步加速了路径查询的效率，尤其是在大规模城市路网和复杂环境下，能够迅速响应交通变化，优化路径选择。此外，遗传算法的全局搜索特性使得该模型在多个目标优化时表现出色，在实际应用中能有效避免局部最优解的问题。

XGBoost-RankNet 模型作为一个高效的排序学习工具，除了用于赛事成绩排序外，还能够广泛应用于推荐系统、电子商务、用户行为预测、金融风控等多个领域。其强大的特征处理和排序学习能力，能够应对复杂数据集，并通过特征自动选择、处理和组合，提高模型的预测准确性。特别是在处理大规模数据时，XGBoost-RankNet 能够快速训练并且避免过拟合，保证模型在实际应用中的可靠性。这个模型适用于多种排名或排序问题，能够为决策者提供精准的排序结果和优化建议。

总体来说，这些模型不仅能够解决各自领域的具体问题，还具有很强的跨领域适应性和扩展性。无论是在体育赛事组织、城市规划，还是在交通调度、物流配送、推荐系统等领域，这些模型都能提供有效的优化解决方案。通过调整目标函数、算法参数和输入数据，这些模型能够在复杂多变的应用场景中提供灵活的决策支持，助力各行业实现更高效的资源配置和优化管理。

参考文献

[1] 孙淼军,陈玉静,杨风艳,等.基于熵权-TOPSIS 法的漂浮式海上风电基础结构方案多准则评价研究[J].太阳能学报，2025，46(04)：406-414.
[2] 王艳艳,李若暄,王瑞泽,等.熵权法-灰色关联度法多指标优化“酸枣-五味子”药对配伍比例及调节睡眠作用研究[J/OL].时珍国医国药,1-6[2025-05-10].
[3] 孙淼军,陈玉静,杨风艳,等.基于熵权-TOPSIS 法的漂浮式海上风电基础结构方案多准则评价研究[J].太阳能学报,2025,46(04):406-414.
[4] 刘志洋,尹慧杰,李明渊,等.利用多源遥感数据的中国三大经济带代表区域经济发展分析[J/OL].武汉大学学报(信息科学版),1-17[2025-05-11].
[5] 廖顺和,乐嘉锦.一种基于无向加权图的距离检索和查询方法[J].计算机工程,2008,(15):80-82.
[6] 魏子蘅,陈志德.基于数学建模与遗传算法的无人机路径规划优化方法研究[J].电脑知识与技术,2024,20(36):45-48.
[7] 张泽宇,郑佑源.基于 Dijkstra 算法的交通需求规划与最佳可达率问题研究——以 2024 年第二十一届五一数学建模竞赛题目 B 为例[J].时代汽车,2025,(09):86-88.
[8] 田洁,朱有晨,李林芝,等.基于GIS与XGBoost算法的新石器时代考古遗址预测模型研究[J/OL].北京师范大学学报(自然科学版),1-15[2025-05-11].
[9] 张学典,方慧.BTDGCNN:面向三维点云拓扑结构的 BallTree 动态图卷积神经网络[J].小型微型计算机系统,2022,43(11):2342-2347.
[10] 祁洋.RankNet 学习排序算法的一种改进[D].吉林大学,2017.

附录

附录 1：

数据预处理代码（部分）：

```
import os
import numpy as np
import pandas as pd
import zipfile  # 添加缺失的导入
from tqdm import tqdm
target_years = [2021,2022,2023,2024]
……
data_folder = r"C:\Users\chelsea\OneDrive\桌面\天气数据\附件 1：中国气象数据\2000-至今"# 文件夹路径……
# 收集所有数据
all_data = []
# 只遍历需要年份的 zip 文件
if os.path.exists(data_folder):
    zip_files = [f for f in os.listdir(data_folder)
                 if f.endswith('.zip') and any(str(year) in f for year in target_years)]
    print(f"开始处理 {len(zip_files)} 个 zip 文件...")……
if all_data:# 合并所有数据
    final_df = pd.concat(all_data, ignore_index=True)
    print(f"\n✓ 成功读取 {len(final_df)} 条记录 (2024 年)")
    print(final_df.head())
    # 保存到 CSV 文件（更新文件名以匹配目标年份）
    output_path = './data/all_weather_data_2024.csv'
    os.makedirs(os.path.dirname(output_path), exist_ok=True)
    final_df.to_csv(output_path, index=False, encoding='utf-8-sig')
    print(f"✓ 数据已保存到 {output_path}")
else:
    print("\n✗ 没有成功读取任何数据")
```

附录 2：

问题 1 代码（部分）：

```
# 构建指标矩阵
    norm_matrix = np.zeros((df.shape[0], len(benefit_cols + cost_cols)))


    for idx, col in enumerate(benefit_cols + cost_cols):
        col_values = df[col].values
        col_min = np.min(col_values)
        col_max = np.max(col_values)
        if col_max == col_min:
            norm_matrix[:, idx] = 0   # 如果全是常数，直接设为 0
        else:
            if col in benefit_cols:
                norm_matrix[:, idx] = (col_values - col_min) / (col_max - col_min)
            else:
                norm_matrix[:, idx] = (col_max - col_values) / (col_max - col_min)


    # 熵权计算
    m, n = norm_matrix.shape
    p = norm_matrix / norm_matrix.sum(axis=0)
    p = np.where(p == 0, 1e-10, p)   # 避免出现 log(0)
    e = - (1 / np.log(m)) * (p * np.log(p)).sum(axis=0)
    d = 1 - e



    weights = d / d.sum() if not np.any(np.isnan(d)) else np.random.rand(len(d))
    weights = weights / weights.sum()   # 确保权重的总和为 1


    print("\n✓ 指标权重:")
    for col, w in zip(benefit_cols + cost_cols, weights):
```

```
    print(f"{col}: {w:.4f}")

# TOPSIS 计算
weighted_matrix = norm_matrix * weights
v_plus = weighted_matrix.max(axis=0)
v_minus = weighted_matrix.min(axis=0)

S_plus = np.sqrt(((weighted_matrix - v_plus) ** 2).sum(axis=1))
S_minus = np.sqrt(((weighted_matrix - v_minus) ** 2).sum(axis=1))

C = S_minus / (S_plus + S_minus)

df['score'] = C

# 排序
df_sorted = df.sort_values(by='score', ascending=False)
# 保存结果
output_path = './data/final_topsis_ranking.csv'
df_sorted.to_csv(output_path, index=False, encoding='utf-8-sig')

print(f"\n✅ 已保存最终排序结果到：{output_path}")
……
print("\n 比赛规模与频次建议:")
print(df_sorted[['city', 'scale_level', 'frequency_suggestion']].head(10))
```

附录 3：

问题 2 指标名称（部分）：

数据处理后的指标名	数据处理前的指标名
Alpine	Alpin Trail de Pichauris - Ultra Duo (FRA)
	Alpine Challenge 100 km (AUS)
	Alpine Challenge 100 Mile (AUS)
	Alpine Challenge 60 km (AUS)
Antelope	Antelope Butte Grind 31M (USA)
	Antelope Canyon 100 Mile (USA)
	Antelope Canyon 50 Mile (USA)
	Antelope Canyon 55 Km (USA)
	Antelope Island Buffalo Run 100 Mile (USA)
	Antelope Island Buffalo Run 50 km (USA)
	Antelope Island Buffalo Run 50 Mile (USA)
	Antelope Island Fall Classic 50K (USA)
Archipelago	The Archipelago 50 km Trail Run (DEN)
	The Archipelago 100 km Trail Run (DEN)
	The Archipelago 70 km Trail Run (DEN)
	The Archipelago 100+70 km Trail Run (DEN)
Austria	Austria Race across Burgenland (AUT)
Bafenshan	
Bali	
Bear	Bear Bait 50K Ultramarathon (USA)
	Bear Bait 50M Ultramarathon (USA)
	Bear Blaster 50K+ (USA)
	Bear Trail 100Km (BEL)
	Bear Trail 58 Km (BEL)
	Bear Trail 83 Km (BEL)
	Bear Lake Ultra Race (USA)
Beauty	American Beauty Ultra Run (TPE)
BeiJing	Beijing 55 km (CHN)
	Beijing 100 km (CHN)
	Beijing 100 miles (CHN)
	Beijing Dazhuangke 50km (CHN)
	Beijing 100 Baoshan Trail - 50km (CHN)
	Great Walker Beijing - 50 km (CHN)
	Beijing Lingshan Trail Race (CHN)
	Linshan100 Ultra Trail Beijing 50 Km (CHN)
	Beijing 100 Baoshan Trail - 100 km (CHN)
Berry	Berryman 50 Mile (USA)
	Berry Long Run (AUS)
	Snowberry Creek Trail Race (USA)
Best	Best of San Diego 100 Miler (USA)
	Best of San Diego 50 Miler (USA)

	Bestmed Chokka Trail Run (RSA)
Big	Big Buffalo 50 Km Endurance Run (USA)
	Big Buffalo 50 Mile Endurance Run (USA)
	Big Butts 50 km (USA)
	Big Butt 50 km (USA)
	Big Butts 100 km (USA)
	Big Day out - The Crossing (GBR)
	Big Dog Ultra Trail Run (USA)
	Big Day out - Granite Tor Marathon (GBR)
	Big Hill Pond Walking Tall (USA)
	Big Hill Bash Ultra (USA)
	Big Stinker Urban Ultra (UAE)
	Big Turtle 50 Km (USA)
	Big Turtle 50 Miler (USA)
	Big's Backyard Ultra (USA)
	Bigfoot 200 Endurance Run (USA)
	Bigfoot 40 Mile Race (USA)
	Bigfoot 100k Race (USA)
	Bigfoot 50K Trail Race (USA)
	Bigfoot Snowshoe Festival (USA)
	Bighorn Mountain Wild & Scenic 100 Mile Trail Run (USA)
	Bighorn Mountain Wild & Scenic 50km Trail Run (USA)
	Bighorn Mountain Wild & Scenic 50mi Trail Run (USA)
Bo	Ultramaraton Borak (CZE)
	Wild Boar 110 (BUL)
	Ultratrail Bosques del Sur (UTBS) (ESP)
Bondi	
Bostrails	
Breaking	
Brisbane	River Run 50 Brisbane (AUS)
	Brisbane Valley Rail Trail 50km (AUS)
	River Run 100 Brisbane (AUS)
Buffalo	Palmetto Bluff Buffalo Run (USA)
	Antelope Island Buffalo Run 50 km (USA)
	Buffalo Stampede Ultra SkyMarathon (AUS)
	Buffalo Stampede Ultra SkyMarathon - Grand Slam (AUS)
	Antelope Island Buffalo Run 100 Mile (USA)

附录 4：

问题 2 代码（部分）：

```
import pandas as pd
import networkx as nx
import math

# 使用原始字符串避免转义问题
data1 = pd.read_csv(r"C:\Users\chelsea\OneDrive\桌面\数据集 5：西安市基础数据（更新）\西安市住宿服务数据.csv")

# 打印查看数据（调试时使用）
print(data1.head())

# 定义计算经纬度距离的函数（使用欧几里得距离公式）
def calculate_distance(lat1, lon1, lat2, lon2):
    # 地球半径（单位：米）
    R = 6371000
    ……
# 创建图，计算距离，并生成边权
def create_graph(data, distance_threshold=1000):
    G = nx.Graph()

    data_subset = data.head(100)
    for idx, row in data_subset.iterrows():
        G.add_node(row['id'], pos=(row['wgs84Lat'], row['wgs84Lng']))

    # 添加边，根据节点之间的距离
    for i, node_i in data_subset.iterrows():
        for j, node_j in data_subset.iterrows():
            if i >= j:  # 避免重复计算
```

```
                continue
            lat1, lon1 = node_i['wgs84Lat'], node_i['wgs84Lng']
            lat2, lon2 = node_j['wgs84Lat'], node_j['wgs84Lng']
            distance = calculate_distance(lat1, lon1, lat2, lon2)  # 使用新的距离计算函数
            if distance <= distance_threshold:
                G.add_edge(node_i['id'], node_j['id'], weight=distance)

    return G

# 创建图
graph = create_graph(data1)  # 假设这里使用住宿数据，可以根据需求切换为景点数据

# 获取图中的所有节点 ID
nodes_in_graph = list(graph.nodes)
# 假设选择图中的第一个和第二个节点作为起点和终点（你也可以选择其他 ID）
start_node = nodes_in_graph[0]  # 选择第一个节点作为起点
end_node = nodes_in_graph[1]      # 选择第二个节点作为终点

print(f"Start node: {start_node}")
print(f"End node: {end_node}")

def find_shortest_path(graph, start_node, end_node):
    try:
        # 计算最短路径，基于边的权重（即距离）
        path = nx.dijkstra_path(graph, source=start_node, target=end_node, weight='weight')
        path_length = nx.dijkstra_path_length(graph, source=start_node, target=end_node,
weight='weight')
        print(f"Shortest path: {path}")
        print(f"Path length: {path_length} meters")
        return path, path_length
    except nx.NetworkXNoPath:
```

```
            print("No path found.")
            return None, None
        except nx.NodeNotFound as e:
            print(f"Error: {e}")
            return None, None

# 获取最短路径
optimal_path, path_length = find_shortest_path(graph, start_node, end_node)
# 将原始数据保存到新文件
data1.to_csv(r"C:\Users\chelsea\OneDrive\桌面\完整数据.csv", index=False, encoding='utf-8-sig')

# 或者将最短路径信息保存到文件
if optimal_path:
    path_df = pd.DataFrame({
        '路径顺序': range(1, len(optimal_path)+1),
        '节点 ID': optimal_path
    })
    path_df.to_csv(r"C:\Users\chelsea\OneDrive\桌面\最短路径.csv", index=False, encoding='utf-8-sig')
    print(f"最短路径已保存至：C:/Users/chelsea/OneDrive/桌面/最短路径.csv")
```

附录 5：

问题 3 代码（部分）：

```
……

# ========================================
# 主程序
# ========================================
if __name__ == "__main__":
    # 设置随机种子，确保结果可重现
    random.seed(42)
    np.random.seed(42)

    # 创建数据处理器
    data_processor = DataProcessor()

    # 读取数据
    print("开始读取数据...")
    data_paths = {
        'accommodation': r"C:\Users\chelsea\OneDrive\桌面\B 题\附件数据\数据集 5：西安市基础数据（更新）\西安市住宿服务数据.csv",
        'restaurant': r"C:\Users\chelsea\OneDrive\桌面\B 题\附件数据\数据集 5：西安市基础数据（更新）\西安市餐饮数据.csv",
        'road_facility': r"C:\Users\chelsea\OneDrive\桌面\B 题\附件数据\数据集 5：西安市基础数据（更新）\西安市道路附属设施数据.csv",
        'scenic': r"C:\Users\chelsea\OneDrive\桌面\B 题\附件数据\数据集 5：西安市基础数据（更新）\西安市风景名胜数据.csv",
        'subway': r"C:\Users\chelsea\OneDrive\桌面\B 题\附件数据\附件 9：西安_2024 年地铁数据\西安_2024 年地铁数据\地铁站点（含经纬度）.xlsx",
        'bus': r"C:\Users\chelsea\OneDrive\桌面\B 题\附件数据\附件 8：西安_2024 年公交站点和线路数据\西安_2024 年公交站点和线路数据\公交站点（含经纬度）.xlsx"
    }
```

```
    # 读取数据
    for data_type, file_path in data_paths.items():
        data_processor.read_data(file_path, data_type)

    # 预处理数据
    print("\n 开始预处理数据...")
    for data_type in data_paths.keys():
        data_processor.preprocess_data(data_type)

    # 采样设施数据
    print("\n 开始采样设施数据...")
    sample_sizes = {
        'restaurant': 500,
        'scenic': 200,
        'subway': 100,
        'bus': 300,
        'road_facility': 300
    }

    sampled_data = {}
    for data_type, sample_size in sample_sizes.items():
        sampled_data[data_type] = data_processor.sample_data(data_type, sample_size)
……
            # 保存最优路线信息
            path_df = pd.DataFrame({
                '路径顺序': range(1, len(best_route)+1),
                '节点 ID': best_route,
                '纬度': [graph.nodes[node]['pos'][0] for node in best_route],
                '经度': [graph.nodes[node]['pos'][1] for node in best_route]
            })
```

附录 6：

问题 4 代码（部分）：

```
……

# 分块读取数据

chunksize = 10000

data_chunks = pd.read_csv(data_path, chunksize=chunksize, dtype=dtype_dict)


# 定义日期处理函数

def extract_last_date(date_str):

    if isinstance(date_str, str) and '-' in date_str:

        parts = date_str.split('-')

        last_part = parts[-1]

        # 如果 last_part 是完整日期

        if re.match(r'^\d{2}\.\d{2}\.\d{4}$', last_part):

            return last_part

        # 如果只写了日和年（如 07.2018），补全月

        elif re.match(r'^\d{2}\.\d{4}$', last_part):

            left_parts = parts[0].split('.')

            return f"{left_parts[0]}.{last_part}"

        else:

            return np.nan

    else:

        return date_str

……# 分批合并数据块

batch_size = 10

merged_chunks = []


for i in range(0, len(processed_data), batch_size):

    batch = processed_data[i:i+batch_size]

    merged_batch = pd.concat(batch, ignore_index=True)

    merged_chunks.append(merged_batch)
```

```
    # 释放批次内存
    del batch, merged_batch

# 最终合并所有批次
df = pd.concat(merged_chunks, ignore_index=True)

# 只保留前 100,000 行
df = df.head(100000)
……
# 奖励建议：取每组前多少比例
top_k = 0.1  # 前 10%
reward_cutoff = int(len(X_test) * top_k)
top_athletes = X_test.nsmallest(reward_cutoff, 'Predicted rank')

print(f"推荐奖励前{top_k*100:.0f}%的参赛者，共{len(top_athletes)}人")
print(top_athletes[['Athlete age', 'Performance seconds', 'Predicted rank']].head())

# 分析哪个年龄组竞争最激烈
group_competition = group_stats.sort_values(by='Std time', ascending=False)
print("竞争最激烈的年龄组：")
print(group_competition[['Age group', 'Std time', 'Num participants']].head())
print(group_competition.shape)

# 根据激烈度设定不同的奖励比例
def reward_ratio(std):
    if std > 1000:
        return 0.15  # 奖励前 15%
    elif std > 500:
        return 0.10  # 奖励前 10%
    else:
        return 0.05  # 奖励前 5%
```

```
group_stats['Reward ratio'] = group_stats['Std time'].apply(reward_ratio)
print(group_stats[['Age group', 'Reward ratio']])

# 按年龄组和赛事计算奖励建议
reward_suggestions = []

for (year, event, age_group), group_data in df_clean.groupby(['Year of event', 'Event name', 'Age group label']):
    # 获取该组的奖励比例
    reward_ratio = group_stats[(group_stats['Year'] == year) &
                               (group_stats['Event'] == event) &
                               (group_stats['Age group'] == age_group)]['Reward ratio'].values[0]

    # 计算奖励人数
    num_participants = len(group_data)
    reward_count = max(1, int(num_participants * reward_ratio))  # 至少奖励 1 人

    # 获取排名前 reward_count 的运动员
    top_athletes = group_data.nsmallest(reward_count, 'Performance seconds')

    # 记录奖励建议
    reward_suggestions.append({
        'Year': year,
        'Event': event,
        'Age Group': age_group,
        'Reward Ratio': reward_ratio,
        'Num Participants': num_participants,
        'Num Reward': reward_count
    })
```