中国研究生创新实践系列大赛
“华为杯”第二十二届中国研究生数学建模竞赛

学　　校		杭州电子科技大学
参赛队号		25103360015
队员姓名	1.	吴泰龙
	2.	詹珂昂
	3.	孟乾轲

中国研究生创新实践系列大赛
“华为杯”第二十二届中国研究生数学建模竞赛

题　目：江南古典园林的美学特征建模

摘　要：

江南古典园林是中华文明在空间营造上的极致表达，其运用“小中见大”的哲学与“移步异景”的技法，在极有限的物理空间内，通过叠山、理水、建筑、花木的精密组合，创造出复杂的景观结构。在复杂精妙的园林设计下，构造出的“有法无式”的独特美学无法被正确量化。

针对问题一，本文首先利用园林矢量数据对园林元素坐标数据进行**数据修补**，之后结合外围建筑封闭情况和入园可行性选取多个出入口，并通过园林平面图和网络信息标注了**充足数量的真实景点**，将补充的景点和出入口定义为园林图结构的节点。为了便于使用路径规划算法，对场景进行精细的栅格化处理，之后利用 Prim 算法的逻辑来选取节点对，结合 A* 算法对节点对进行连通性测试和路径规划，最终生成了刻画园林的稀疏连通图。并通过引入景点密度概念在重点区域对图进行**密集化**，形成了稠密的连通网络。通过对每条路径计算**路径长度**、**交叉点数量**、**转折点数量**，作为园林图结构中边的关键特征。为量化“移步异景”，使用**视域分析**方法，在路径的离散观测点上，通过射线追踪法量化视域面积、可见度及景观构成比例等特征，并引入 **Jensen-Shannon（JS）散度**来衡量相邻观测点间景观构成比例的分布差异，实现了每条边对应路径的 **“异景程度”** 的刻画。至此，得到了“路径动态性”和“异景程度”两个关键特征，进一步加权求和来实现园林 **“趣味性”** 的综合量化，最终得到趣味性第一为拙政园（0.941 分）。最后，结合路径重复惩罚，建立了一个基于**多目标价值最大化的关键游线提取模型**，该模型综合考量了路径特征、异景程度、景点覆盖率与**空间探索度**，并利**用基于优先队列的启发式搜索算法**进行求解，为十座园林规划出兼顾趣味性与核心价值的最优游览线路，计算**异景关键点**并在对应园林平面上进行数据可视化。

针对问题二，本文首先通过对园林内所有路径进行离散化采样，并对每个采样点进行视域分析，提取其景观特征向量。随后，采用一种**两阶段聚类方法**：先利用 K-means 算法基于景观特征将观测点进行宏观聚类。利用**凸包算法**生成每个聚类所覆盖的多边形区域，统计内部的景观元素进行**景观分区**。由于园林不同类型的景观分区空间位置并不独立，进

一步使用 DBSCAN 算法基于空间坐标识别出各**景观宏观分区的子空间**。为量化空间的开合变化，利用视域分析得到的视域面积、视觉通透率、遮挡物密度等特征，构建了开阔度的量化标准。在此基础上，我们设计了一个包含主题多样性、开合节律和要素配比三个核心指标的**幻境感加权评分模型**。以寄畅园为 100 分基准，通过**熵权法**客观确定指标权重。将该权重作为模板，对十座园林的幻境感进行了量化评分与排序，得到**幻境感第二名为沈园（90.9 分）**。

针对问题三，本文首先基于园林的多模态数据，选取了**向量**、**集合**、**图结构**、**文本**、**有序序列和边界形状这六种模态**。针对每种模态设计数据并选用对应的相似度族的相似度工具进行计算。具体来说，使用前两问构造的所有数值特征作为向量特征，对应选用**余弦相似度**进行计算；使用路径上视域可见对象的集合和景点名称的连续字符对构建的二元组集合，对应选用**杰卡德相似度**进行计算；使用路径刻画中获得图的拓扑结构，对应选用**谱相似性**进行计算；文字数据使用 n-gram 提取特征，利用 TF-IDF 向量化之后使用余弦相似度计算；使用视域变化序列对应使用动态时间规整算法实现序列相似计算。特别的，针对边界形状相似度，获取不同园林的“山”、“水”、“林”、“楼”的边界数据，利用形状上下文计算相似度。之后参考**相似性网络融合（SNF）算法**实现了多模态相似度的融合。之后基于**统计分析**、**聚类分类**、**层次构建**三种方式提取美学特征。此外，本文的方法成功在**网师园**该新增园林上，运行了前面问题的结果，验证了方法的**泛化性**。并探索了自然度、文化内涵等的园林评价新指标。

关键词：路径特征　异景关键点　启发式搜索　幻境感加权评分　两阶段聚类　相似度

目录

1 问题重述

1.1 问题背景

江南古典园林作为中国传统文化的重要组成部分，以其独特的美学特征和深厚的文化内涵闻名于世。其造园艺术不仅影响了亚洲文化圈内的朝鲜、日本等地，还对欧洲园林艺术的发展产生了深远影响。与西方园林几何式、规则式的美学特征不同，江南古典园林以“不规则性”和“丰富多变”著称，强调“有法无式”的设计理念，展现出“移步异景”“小中见大”等独特的审美体验。随着信息化时代的到来，数字化与智能化方法为研究江南古典园林的美学规律提供了新的可能性。通过数学建模提炼园林设计背后的隐性规律，不仅能够突破传统认知的局限，还能为中国园林美学的传承与创新提供科学依据。本题以江南地区明清时期的十个代表性园林为研究对象，探索如何量化建模江南古典园林的美学特征，并将研究成果应用于其他园林，推动中国传统文化的现代化表达与国际化传播。

1.2 问题提出

围绕江南古典园林“移步异景”的游赏趣味性、“小中见大”的布局幻境感以及“有法无式”的园林相似度，本题依次提出如下问题：

问题一：“移步异景”的游赏“趣味性”建模

江南古典园林通过蜿蜒曲折的游线设计和多样化的景观元素布局，营造出“移步异景”的趣味性体验。造园家通过延长游览路径、增加视角变化，使观赏者在有限空间中感受到丰富多变的景观效果。本问题要求：

1. **路径刻画**：将游园路径转化为图问题，定义路径长度、转折点数量、交叉点数量等关键特征。
2. **异景程度**：建模路径与景观元素的分布组合关系，量化异景程度，并计算游览趣味性。
3. **游线规划**：基于上述分析，规划既满足趣味性又减少重复路径的游园路线，并可视化展示十个园林的游园路线及异景程度分析。

问题二：“小中见大”的布局“幻境感”建模

江南古典园林通过不同景观元素的种类、面积比例及空间布局，营造出“小中见大”的幻境感。园林空间的“开合”变化进一步增强了这种感受。本问题要求：

1. **元素分布**：建模景观元素的分布特征，探索其对园林空间审美的影响。
2. **开合变化**：量化园林空间的“开阔”与“围合”特征，建模空间的开合变化。
3. **幻境评分**：以无锡寄畅园的幻境感为标准（100 分），评价其余九个园林的幻境感表现，给出评分并排序。

问题三：“有法无式”的园林“相似度”建模

江南园林在看似不规则的布局中蕴含一定的普遍性规律，体现了“有法无式”的设计理念。本问题要求：

1. **相似度**：结合问题 1 与问题 2 的建模结果，量化十个园林的相似度，分析其共有的美学特征与规律。
2. **广效用**：将模型应用于其他园林，验证其科学性、有效性与可泛化性。
3. **多元化**：探讨除“趣味性”“幻境感”“相似度”外，是否还有其他重要的美学特征可用于刻画江南古典园林，并提供相关依据。

2 模型的假设

1. 所有假山石均有足够高度遮挡视线；
2. 所有出入口以数据中外围建筑不连接部分且可通行至园林内部的缺口为准；
3. 所有不在道路内部和建筑内部的区域均不可通行；
4. 游客游览过程中可能在可到达的任意位置 360 度观赏园区；

3 符号说明

符号	意义
G_{grid}	栅格化后的园林地图
c_{ij}	地图中的网格单元
d	园林中所有景点间的平均直线距离
$N_{max\_cluster}$	密集区域内的最大景点数
θ_i	路径上连续三点构成的夹角
θ_{thresh}	判断为转折点的角度阈值
L	路径的长度
T	路径的转折点数量
C	路径的交叉点数量
P	路径上的有序观测点集合
d_v	视域分析中的最大视距
A_i	观测点 p_i 的视域多边形面积
S_i	观测点 p_i 的景观构成比例分布
M	两个景观构成比例分布的平均分布
f_{scene}	路径的异景程度得分
f_{path}	路径的路径特征得分
V_e	单条边的综合趣味性价值
r_{cover}	游线的景点覆盖率
$r_{explore}$	游线的空间探索度
V_{path}	游线的累积趣味性
Q	游线的综合评分
I_{open}	开阔度指数
I_{encl}	围合度指数
D_t	主题多样性指标
N_e	有效主题数

表 3.1

符号	意义
P_f	分段连续性因子
R_{oe}	开合节律指标
E_p	要素配比指标
S'_t	主题多样性的原始得分
S'_k	开合节律的原始得分
S'_m	要素配比的原始得分

4 问题一模型建立与求解

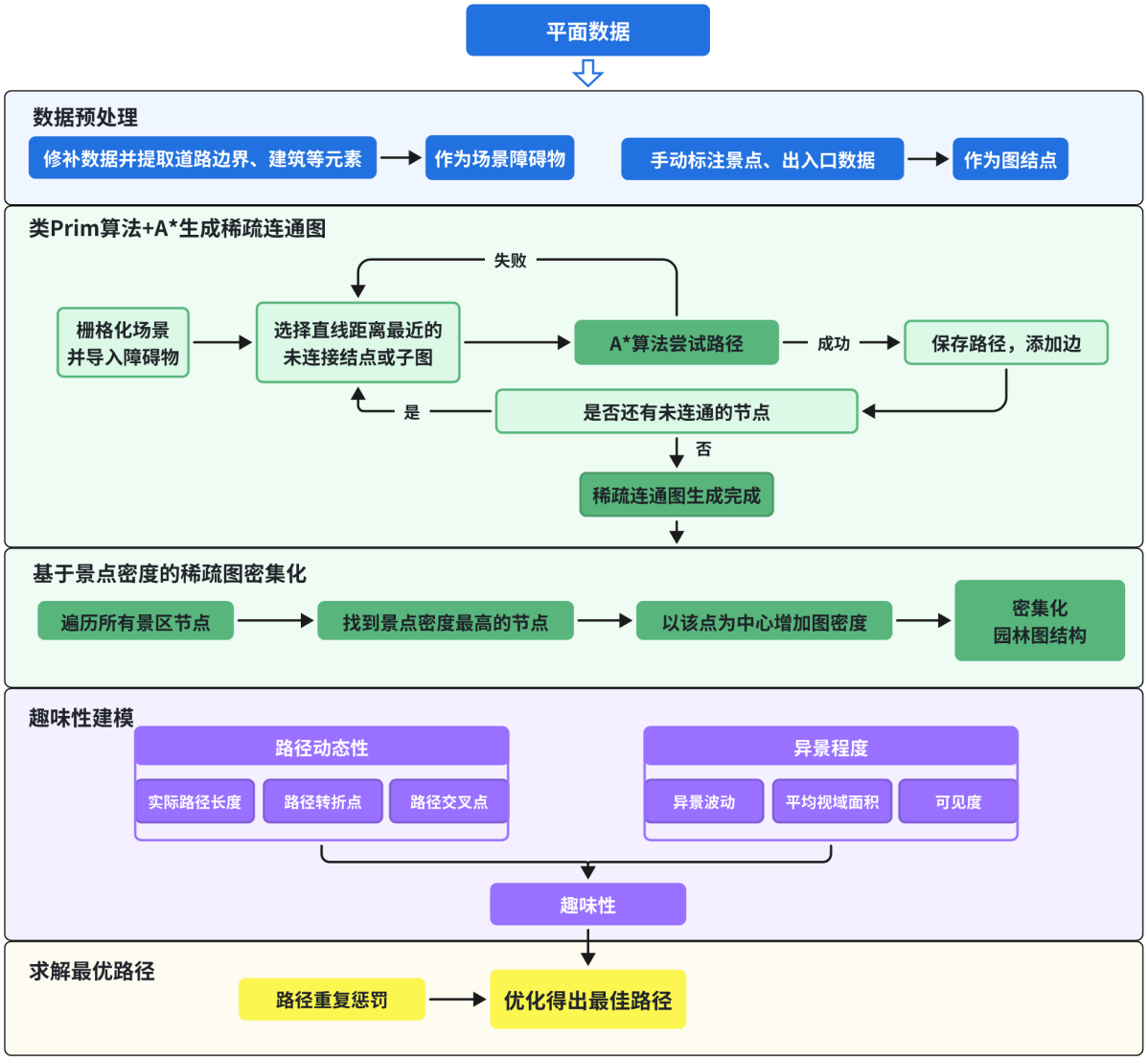

图 4.1　问题一框架图

4.1 问题分析

针对路径刻画，首先需要将复杂的园林场景抽象为可量化的图结构。一个完整的图结构，由节点集和边集组成。针对游览园林的问题场景，本文先进行景点数据和出入口数据的补充，得到了图结构的节点集。之后先以实现一个最简化的可以完整游览的图结构为目标，即在当前点集中构建一个最简的连通图。本文参考了 Prim 算法的思路，并使用节点之间的空间直线距离来确定尝试连接的节点次序。对场景进行栅格化处理之后，利用 A* 算法来测定两个节点之间的连通性，并规划在园林场景下的可达真实行走路径。通过 Prim 结合 A* 的构造算法，得到了一个初步的稀疏连通图。但是针对游园的场景来说，只是一

个最简连通图，并无法良好的表达游园时景点间自由的通行状态。为了解决这个问题，本文进一步引入了景点密度的概念，在景点密度最高的区域半径内对构建的连通图进行密集化，以增加图密度。通过密集化的处理，得到了一个园林的密集连通图。之后利用边对应的真实路径来计算关键特征（路径长度、转折点和交叉点数量）。实现了将园林场景抽象成一个真实可以游览并带有路径动态性特征的图结构。

之后，题目要求在量化园林场景的异景程度。园林的景观相对于在地理的空间位置上是静态的，“异景”是发生人在观测园林时进行移动，造成了相对位置的变化。所以，本文选择在 A* 算法规划出的真实可行走路径上，进行视域分析，得到路径上不同观测点视角、景观的变化指标。并进一步将这些指标结合成异景程度。通过上述对园林场景带有路径动态性特征的图结构的生成和异景程度的量化，游览园林时带来的“趣味性”便可以很自然得通过求和的方式得到。本文将这异景程度和趣味性也作为园林图结构中的边特征。

最后，题目要求在刻画的路径中选择趣味性高并且重复程度低的游园路线。即是在图结构中，增加设定出入口的约束，结合趣味性高重复程度低的目标函数，在图结构中优化出一条游览路径。本文通过引入路径重复度的惩罚项，结合前两问得到的趣味性指标，利用启发式搜索算法求解得到。

问题一框架图见图4.1。

4.2 数据预处理

为了确保模型构建的准确性与数据处理的高效性，对所提供的园林数据进行了系统化的预处理，具体步骤如下：

1. **出入口的标注**：根据园林平面图，首先确定每个园林的入口与出口位置。由于部分园林的平面图未明确区分入口与出口，并且随着时间推移和景区修缮改造等问题，出口和入口均在变动。结合实际情形，本文将入口与出口统一定义为“出入口”。基于园林矢量图文件，选择所有外部围墙未封闭且可通行（未直接进入道路或者建筑）的缺口作为出入口。
2. **景点标注**：园林矢量图中未标注具体景点信息，而后续游园路径转化为图问题时需将景点作为图的节点。因此，结合园林平面图与相关文献资料，对园林中的主要景点进行筛选与标记，并将这些景点位置映射到矢量图中，确保节点信息的准确性与完整性，图**??**展示了出入口与景点标注的最终情况，表4.1为标注数据统计。

表 4.1　园林标注统计表

园林名称	拙政园	留园	寄畅园	瞻园	豫园	秋霞圃	沈园	怡园	耦园	绮园
景点数量	21	20	12	11	19	20	12	15	15	7
出入口数量	4	2	3	3	4	3	2	4	3	5

3. **矢量图格式转换**：由于模型实现基于 Python 语言，而 Python 对 DWG 格式的矢量图文件支持较为有限，为提高数据读取的适配性，将 DWG 文件转换为 DXF 格式。DXF 文件具有更好的兼容性与可读性，能够满足后续数据处理与分析的需求。
4. **坐标数据可视化与修补**：园林矢量图提供了景观元素的坐标数据，但观察可视化结果发现部分数据存在断点或缺失现象。为解决这一问题，首先对坐标数据进行可视化处理，直观展示园林布局与景观元素分布；随后结合矢量图文件对断点数据进行修补，确保数据的连续性与完整性，图4.2为坐标数据可视化与修补数据对比图。

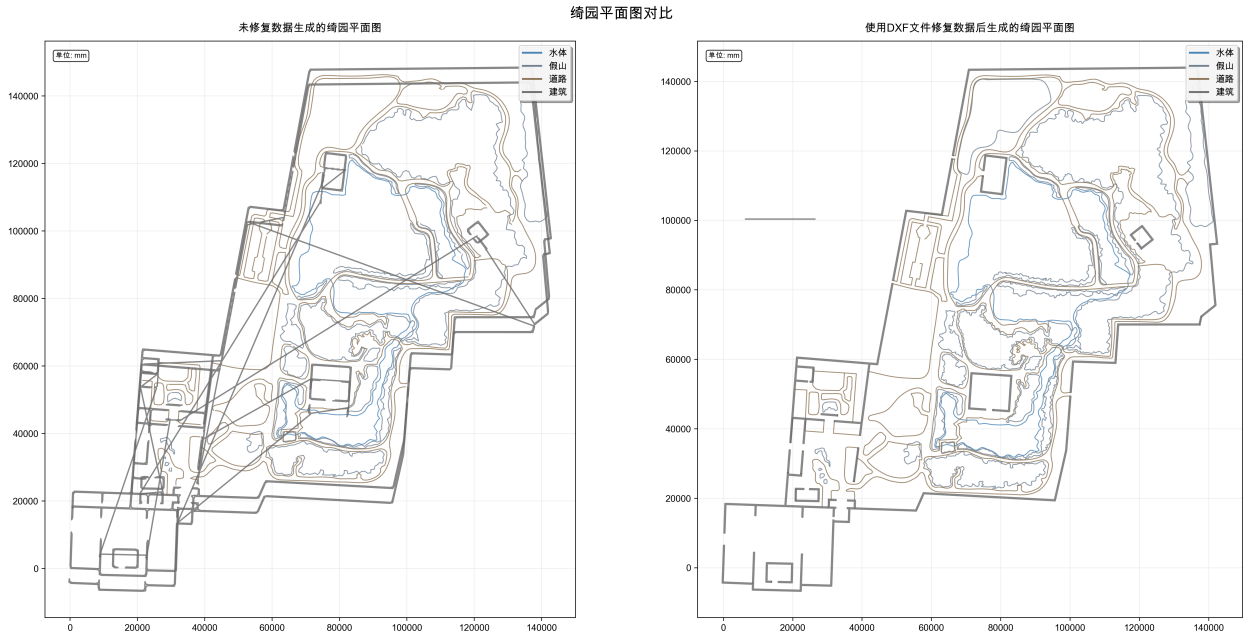

图 4.2　数据可视化与修补数据对比（以绮园为例）

4.3 路径刻画

江南古典园林的游览体验根植于其精心设计的路径网络。为了对这一物理空间结构进行精确的数学描述，需要将复杂的园林布局抽象为一个图论模型。这一过程通过量化分析，构建出既能反映园林基本连通性、又能体现其设计巧思的路径网络。建模的核心思路在于将园林空间离散化，并利用算法生成符合游览逻辑的路径。具体建模过程分为三个步骤：首先，通过数据预处理构建包含障碍物的栅格化地图，为路径规划奠定基础；其次，结合 Prim 与 A* 算法生成连接所有关键节点的稀疏基础路网；最后，通过引入景点密度概念，在重点区域加密线路，形成最终的、结构更为丰富的游园路径网络。且这样的路径构建方法不依赖完善的 dwg 文件，并可以动态适应不同的可行路径规划算法，即使在路径开放区域、室内外路径切换区域均能实现真实可达的细粒度连续路径生成。

4.3.1 障碍图的构建与空间栅格化

在江南古典园林的路径刻画中，首先在数据预处理章节得到了一张包含景点和出入口的图。这一图的建立是路径规划的基础，帮助准确识别园林内的道路和建筑结构。在本问题中，主要关注道路和建筑的障碍物，障碍物的提取规则具体为：保留道路、封闭建筑物和半开放建筑物。这样做的原因在于封闭建筑物完全隔绝了行走路径，而半开放建筑物虽然不遮挡视线，但仍不允许直接通行，因此同样会对游园路径造成限制 [1]。

在经过上述处理后，得到了园林的障碍图4.3。接着对整个园林图进行栅格化。栅格化

图 4.3　简化的园林布局图（以拙政园为例）

的目的是将连续的园林空间转化为离散网格 [2]，这样就可以通过算法对园林进行路径规划。栅格化后的图中，障碍物将被标注为不可通行区域，而其他可通行区域则作为路径计算的基础。这个过程通过下述公式来表示：

$$G_{grid} = \{c_0 1, c_0 2, \ldots, c_n m\} \tag{4.1}$$

其中，G_{grid} 表示栅格化后的园林图，$c_i j$ 为图中每个网格单元，n 为网格单元的列数，m 为网格单元的行数。对于每个网格单元，如果其处于道路或建筑物范围外，则其被标记为障碍物，否则为可通行区域。

4.3.2 路径规划

完成障碍物图的构建后，接下来使用 Prim 算法结合 A* 算法来构建园林的道路联通图。首先，Prim 算法用于节点的选择，构建最小生成树，通过贪心的方式优先选择距离当前已选节点最近的节点，并将其加入集合。

对于每一对节点 u, v，通过欧几里得距离计算它们之间的代价：

$$d_E(u,v) = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2} \tag{4.2}$$

其中，(x_u, y_u) 和 (x_v, y_v) 分别是节点 u 和节点 v 的坐标。

接下来，对于使用 prim 算法选中的两个节点进行路径规划，使用 A* 算法的启发函数 $h(n)$（节点 n 到目标节点的估算代价）来得到估算距离：

$$h(n) = \sqrt{(x_n - x_{goal})^2 + (y_n - y_{goal})^2} \tag{4.3}$$

最终得到的路径即为 $h(n)$ 最小的路径。具体来说，基于栅格化之后的园林图，A* 算法从当前节点所在栅格出发，寻找相邻可通行栅格，计算欧几里得距离，最终综合确定总欧几里得距离最短的路径。

通过 A* 算法的启发函数，能够为每一对节点计算出一条最优路径，从而确保道路网络的高效连接。最后使用以上描述的组合算法，将节点逐步加入节点集合，直到所有节点（包括出入口节点）联通，得到一条稀疏路径，图4.4为最终得到的稀疏路径图，红色线为求得路径。

4.3.3 景点密度与密集线路

在上一节中仅规划了一条联通所有节点的稀疏路径，为了找到更多可选择的路径且提高路径规划的效率，避免遍历整个园林所有路径的高计算复杂度，引入景点密度的概念。具体来说，首先计算所有景点之间的平均直线距离，并将其定义为 d。然后，以这个距离为半径，搜索园林中景点密度最高的区域。密集区域的选择方式是，通过计算每个景点为圆心，以 r 为半径的区域内包含景点的数量，找到景点数最多的区域。

在密集区域内，为了增强路径的多样性，增加更多的路径。通过这一方式，我们能够在景点密度较高的区域生成更多的不同路径，确保路径之间的差异性和复杂性。密集区域的最大景点数可以通过如下公式表示：

$$N_{max_cluster} = \sum_{i=1}^{n} I(r_i) \tag{4.4}$$

其中，$I(r_i)$ 表示景点 i 在半径为 d 的圆内的指示函数，若景点 i 在圆内，则 $I(r_i) = 1$，否则为 0。通过在这些密集区域内增加路径，我们能够使得园林的路径更加多样化，图4.5为最终得到的密集路径图，并将其转化为图结构。

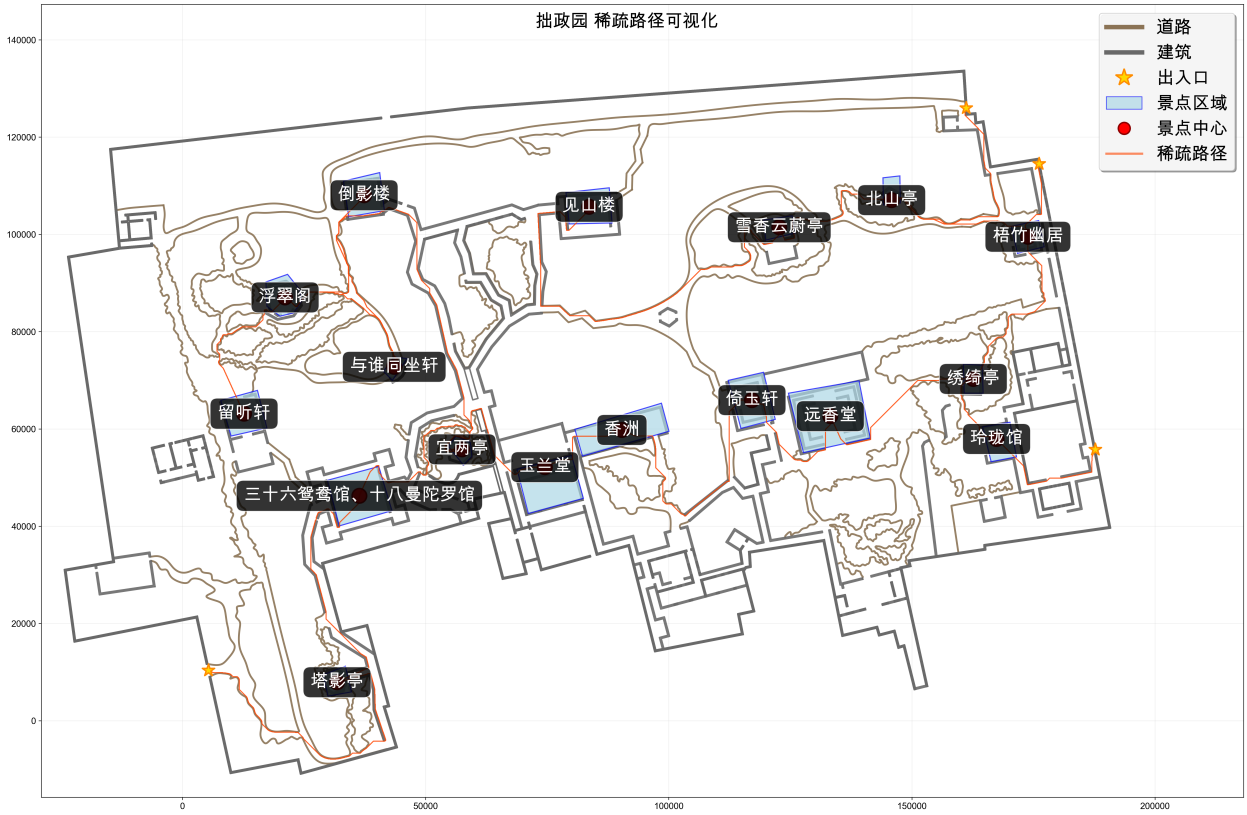

图 4.4　稀疏路径图（以拙政园为例）

图 4.5　密集路径图（以拙政园为例）

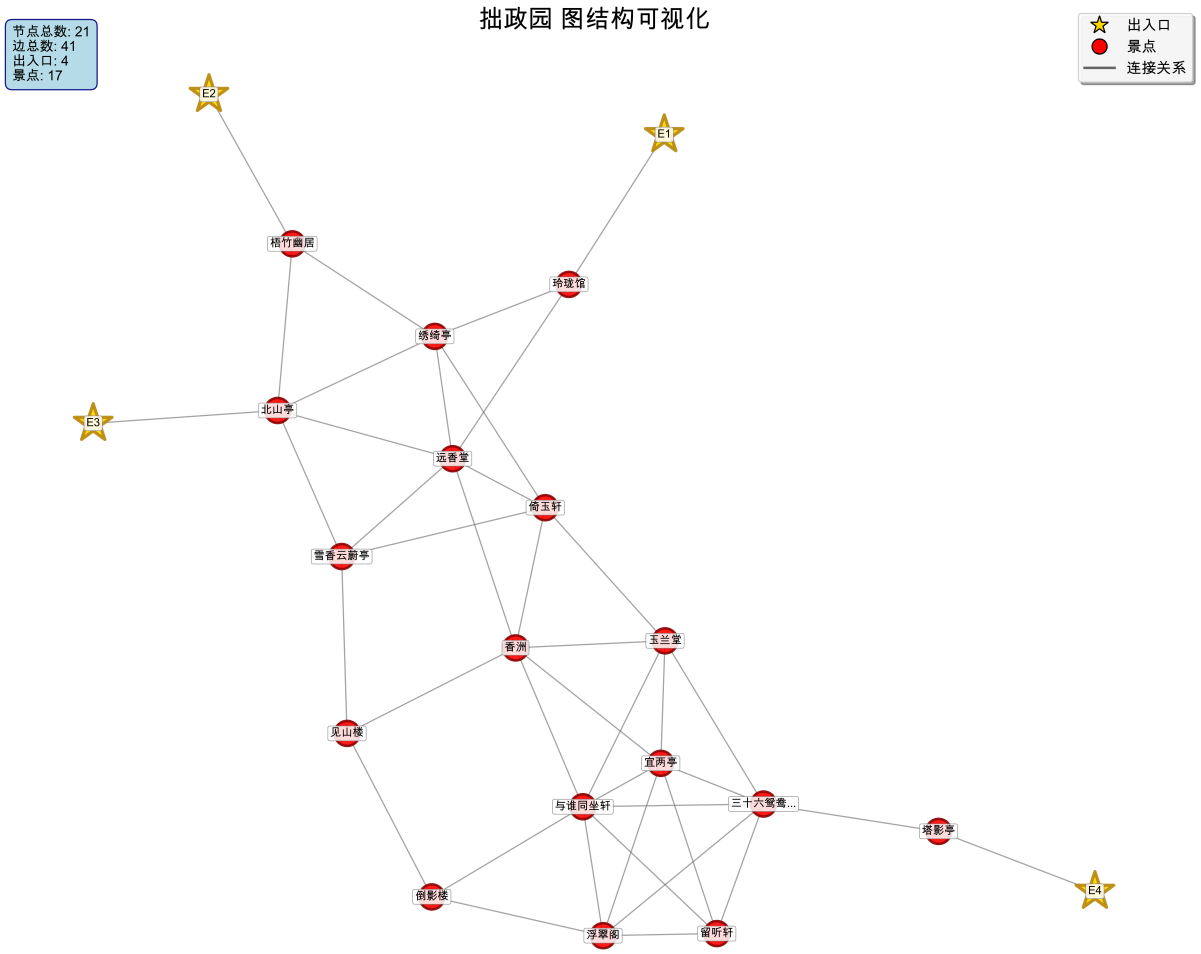

图 4.6　图结构（以拙政园为例）

4.3.4 路径特征的定义

在园林路径规划中，路径的关键特征可以有效地反映路径的复杂度和趣味性。本文定义路径的主要特征包括路径长度、转折点数量和交叉点数量。下面将对这三项特征进行详细定义和计算方法描述。

- **路径长度：**
  在上述进行路径规划时，通过 A* 算法的启发函数 $h(n)$ 得到了路径长度，此路径长度即为两个景点间的路径长度。最终实际路径长度可通过比例尺将实际距离映射回原图尺寸 [3]。
- **转折点数量：**
  转折点是路径中方向发生明显变化的节点。为了定义转折点，我们需要计算路径上每个记录点与前后相邻记录点之间的角度。当每三个连续的记录点 p_{i-1}、p_i、p_{i+1} 构成的线段的夹角大于一个预设的阈值时，p_i 就被认为是一个转折点。

具体地，设两个相邻的线段分别为 $p_{i-1}p_i$ 和 p_ip_{i+1}，这两个线段之间的夹角 α_i 可通过以下公式计算：

$$\cos(\theta_i) = \frac{(p_i - p_{i-1}) \cdot (p_{i+1} - p_i)}{|p_i - p_{i-1}| \cdot |p_{i+1} - p_i|} \tag{4.5}$$

其中，$(p_i - p_{i-1})$ 和 $(p_{i+1} - p_i)$ 分别是两个相邻线段的向量，$\cdot$ 表示向量的点积，$|\cdot|$ 表示向量的模长。夹角 θ_i 即为两条线段之间的夹角，如果 θ_i 大于设定的阈值 θ_{thresh}，则该记录点 p_i 被视为一个转折点。

转折点数量 T 计算公式如下：

$$T = \sum_{i=2}^{n-1} I(\theta_i > \theta_{\text{thresh}}) \tag{4.6}$$

其中，$I(\theta_i > \theta_{\text{thresh}})$ 是指示函数，当 $\theta_i > \theta_{\text{thresh}}$ 时，I 的值为 1，否则为 0。最终，通过遍历路径中的所有记录点，统计转折点的总数量。

- **交叉点数量：**

  交叉点是指路径中两条不同的路径相交的点。为了统计路径中的交叉点，我们需要检测路径上所有线段之间的交点。每一对路径上的线段 (p_ip_{i+1}) 和 (p_jp_{j+1})，如果它们在平面上相交，且不重合，则该交点就是一个交叉点。

  交点的判定可以通过几何计算来实现，具体而言，若两条路径的线段 (p_ip_{i+1}) 和 (p_jp_{j+1}) 的方向向量分别为 $(p_{i+1} - p_i)$ 和 $(p_{j+1} - p_j)$，则它们相交的条件是：

$$\begin{aligned} (p_{i+1} - p_i) \times (p_j - p_i) \cdot (p_{i+1} - p_i) \neq 0 \\ (p_{i+1} - p_i) \times (p_{j+1} - p_j) \cdot (p_{i+1} - p_i) \neq 0 \end{aligned} \tag{4.7}$$

  当两条路径的线段满足这些交叉条件时，我们可以认为它们在交点处相交。统计图中的交叉点数量可以通过计算所有路径对之间的交点来实现。最终，交叉点数量 C 可以通过以下公式得到：

$$C = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} I(l_i \cap l_j) \tag{4.8}$$

  其中，$l_i \cap l_j$ 表示第 i 条和第 j 条路径线段相交的条件，$I(l_i \cap l_j)$ 为指示函数，当两条路径相交时，该值为 1，否则为 0。

通过以上三种关键特征的定义，能够有效地描述路径的长度、复杂度以及交互性，这些特征对于后续路径优化和游园体验的设计有着重要作用。

4.4 异景程度

“移步异景”是中国古典园林营造趣味性的核心手法，其精髓在于观赏者在游览路径上移动时，所见的景观画面不断发生变化。为了对这一动态的游赏体验进行量化建模 [4]，本文使用了视域分析的方法，将“景”定义为观赏者在特定位置所能看到的空间范围及其内容构成，而“异景”则定义为相邻位置之间“景”的变化程度。“异景”的体验源于视觉信息的动态变化，这种变化可以体现在两个层面：一是可见空间范围的大小与形状变化；二是可见景观元素构成的变化。基于此，建模过程分为三个核心步骤：首先，将路径刻画中构成路径的点作为观测点；其次，在每个观测点上通过射线追踪法量化其视域特征；最后，通过比较相邻观测点之间的视域特征差异，计算出量化的异景程度与游览趣味性。

4.4.1 路径离散化与观测点采样

为了对游览过程中的视觉变化进行分析，需要将“路径刻画”章节中生成的游园路径转化为一系列离散的观测点，直接利用 A* 算法在栅格地图上规划出的路径本身。

具体而言，A* 算法的输出结果是一条由相邻栅格单元构成的有序序列。在进行视域分析时，这个栅格序列中的每一个栅格都被视为一个独立的观测点。因此，一条完整的游览路径就被离散化为一个有序的观测点集合 $P = \{p_1, p_2, \ldots, p_m\}$。

采用这种方法，观测点之间的距离不再是固定的，而是由栅格的尺寸以及路径的走向（水平、垂直或对角线移动）所决定。它将视域分析的观测点与路径规划的底层结构（栅格地图）紧密地绑定在了一起，确保了路径上由算法决定的每一个“微观步骤”都被纳入后续的“异景”程度计算中。

4.4.2 单点视域特征量化

对于路径上的任意一个观测点 p_i，需要精确地量化其所能观察到的“景”。此处采用射线追踪法来实现这一目标。从观测点 p_i 出发，向周围 360° 方向均匀发射 N 条虚拟的“视线”射线。每条射线会沿着其方向前进，直至遇到第一个不透明的障碍物或达到预设的最大视距 d_v。

园林中的不同元素对视线的遮挡效果不同。根据其物理特性，园林景观元素（从 DXF 矢量图中提取）被分为两类：

- **视觉不透明障碍物**：包括封闭建筑、假山、植被。当射线与这些物体相交时，射线停止传播。
- **视觉可穿透对象**：包括水体、半开放建筑。射线可以穿过这些对象，不影响视线传播。

对于每条射线 j，记录其传播距离 d_j。所有 N 条射线的终点连接起来，便构成了一个以观测点 p_i 为中心的不规则多边形，称之为视域多边形。这个多边形直观地描绘了观赏者在该点的视野范围。

基于视域多边形，为每个观测点 p_i 定义了以下关键特征指标：

- **视域面积**：视域多边形的面积。若视域多边形由 N 个顶点 $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ 按顺序构成，则其面积 A_i 的计算公式为：

$$A_i = \frac{1}{2} \left| \sum_{k=1}^{N} (x_k y_{k+1} - x_{k+1} y_k) \right| \tag{4.9}$$

其中，约定最后一个顶点与第一个顶点相连，即 $(x_{N+1}, y_{N+1}) = (x_1, y_1)$。该指标反映了视野的开阔程度。面积越大，视野越开阔；面积越小，空间感越围合 [5]。

- **可见度**：未被障碍物在最大视距内阻挡的射线数量占总射线数量的比例：

$$\eta(i) = \frac{N_{\text{rays}} - N_{\text{blocked}}}{N_{\text{rays}}} = 1 - \frac{N_{\text{blocked}}}{N_{\text{rays}}} \tag{4.10}$$

其中 N_{rays} 是从观测点发出的总射线数量，N_{blocked} 是被障碍物阻挡的射线数量

- **景观构成比例**：一个向量，记录了所有被阻挡的射线分别遇到不同类型障碍物的比例。对于任意一种景观类型 t，其在点 p_i 视野中的构成比例 S_i 为：

$$S_i(T) = \frac{N_{i,t}}{N_{\text{rays}}} \tag{4.11}$$

其中 $N_{i,t}$ 是从观测点 p_i 发出并撞击到类型为 t 的障碍物的射线数量。最终，景观构成比例是一个包含所有类型比例的比例分布 S_i：

$$S_i = \{(t, S_i) \mid \forall t \in U\} \tag{4.12}$$

其中，U 为景观类型的全集，该指标描述了当前视野中各种景观元素的构成。

4.4.3 异景程度量化

“移步异景”的核心特征在于游览过程中可视景观的变化，即随观测位置移动而产生的视域特征差异。因此，本文通过量化路径上相邻观测点间的视域特征差异，构建“异景程度”的可计算模型。

首先在量化异景程度前，引入异景波动值，其定义为：对于路径上的观测点 p_i $(i > 1)$，其与前序观测点 p_{i-1} 之间视域特征的综合差异，用于表征两点间“异景”的强弱程度。为简化模型并聚焦核心影响因素，本文选取景观构成比例作为计算异景波动值的核心参数。景观构成比例直接反映观测点视域内各类景观元素（如墙体、植物、水体等）的占比关系，其变化是“移步异景”体验中最直观、最具冲击力的视觉感知来源——例如从“以植物为主”到“以水体为主”的比例转变，会直接引发强烈的景观差异感受，因此该参数能有效捕捉“异景”的本质特征。

为精准衡量相邻观测点景观构成比例的差异，本文引入 Jensen-Shannon 散度（JSD）作为量化指标。如前文所述，当所有射线均撞击某类障碍物时，观测点的景观构成比例集合

S_i 与 S_{i-1} 可视为两个离散概率分布（满足非负性与归一性）；而 JSD 作为一种对称化、平滑化的概率分布差异度量方法，既能避免 Kullback-Leibler（KL）散度的非对称性缺陷，又能通过“平均分布”的引入降低极端值对差异计算的干扰，更适用于景观构成比例这类多类别概率分布的比较场景。针对路径上相邻的两个观测点 p_i 与 p_{i-1}，其景观构成比例分布分别为 S_i 与 S_{i-1}，JSD 的具体计算步骤如下：

$$M = \frac{1}{2}(S_i + S_{i-1}) \tag{4.13}$$

其中，M 的每个维度对应某一类景观类型 t 的平均占比，即 $M(t) = \frac{1}{2}[S_i(t) + S_{i-1}(t)]$，确保后续差异计算的对称性。

之后，计算 Kullback-Leibler (KL) 散度，KL 散度是衡量一个概率分布与另一个概率分布的差异。分别计算和 S_{i-1} 相对于 M 和和 S_i 相对于 M 的 KL 散度。

$$D_{KL}(S_i \| M) = \sum_t S_i(t) \log_2 \left(\frac{S_i(t)}{M(t)} \right) \tag{4.14}$$

$$D_{KL}(S_{i-1} \| M) = \sum_t S_{i-1}(t) \log_2 \left(\frac{S_{i-1}(t)}{M(i)} \right) \tag{4.15}$$

其中 t 代表每一种景观类型。

计算 JSD：JSD 是这两个 KL 散度的平均值。

$$\text{JSD}(S_i \parallel S_{i-1}) = \frac{1}{2} D_{KL}(S_i \parallel M) + \frac{1}{2} D_{KL}(S_{i-1} \parallel M) \tag{4.16}$$

JSD 的取值范围在 $[0, 1]$ 之间（如果使用以 2 为底的对数），值越大表示两个点的景观构成差异越大

$\text{JSD}(S_i \parallel S_{i-1})$ 直接量化了从点 p_{i-1} 移动到点 p_i 所带来的景观变化幅度。JSD 越大，表示景观变化越剧烈，“异景”效果越显著。

在定义完异景波动值JSD后，对异景程度进行量化，“移步异景”的核心特征在于游览过程中可视景观的变化，即随观测位置移动而产生的视域特征差异。因此，本文通过量化路径上相邻观测点间的视域特征差异，构建 **异景程度**指标，它由多个视域分析得出的特征构成，主要包括平均视域面积 A_{avg}、平均可见度 V_{avg}、以及异景波动值JSD。计算公式为：

$$f_{scene} = \frac{1}{3} \left(\frac{\min(A_{avg}, A_{max})}{A_{max}} + V_{avg} + \frac{\min(JSD_{avg}, JSD_{max})}{JSD_{max}} \right) \tag{4.17}$$

其中，A_{avg} 是平均视域面积，V_{avg} 是平均可见度（本身即为比例，无需归一化），JSD是相邻观测点间的平均异景波动值。

4.5 趣味性建模

模型的基础是对路网中的每一条基础路径段（即连接两个节点的边）进行趣味性评估。这一定义是后续路径搜索的核心依据。一条边的趣味性由其“路径特征”和“异景程度”共同决定 [6]。

- **路径特征**: 该指标由4.3.4章节定义，衡量路径本身的几何特征。它由三个归一化的子特征加权构成，它由三个归一化的子特征等权重构成：路径长度 L、转折点数量 T、以及交叉点数量 C。其计算公式如下：

$$f_{path} = \frac{1}{3}\left(\frac{\min(L, L_{max})}{L_{max}} + \frac{\min(T, T_{max})}{T_{max}} + \frac{\min(C, C_{max})}{C_{max}}\right) \tag{4.18}$$

  其中，L, T, C 分别代表边的长度、转折点数和交叉点数。$L_{max}, T_{max}, C_{max}$ 是各自的归一化基准值，用于将不同物理量的特征统一到 [0, 1] 区间。
- **异景程度**: 该指标由4.4.3章节定义，直接量化了“移步异景”的效果，反映了游客在行进中视觉体验的变化程度。

最终，一条边的**趣味性**即综合价值 V_e 通过对上述两大类指标进行加权求和得到，其目标函数可以表示为：

$$V_e = \alpha \cdot f_{path} + \beta \cdot f_{scene} \tag{4.19}$$

其中，α 和 β 分别是路径特征和异景程度的权重系数，其取值反映了模型对不同趣味性来源的侧重。本文选用 α=0.4,β=0.6, 通过该公式，我们将抽象的游览体验量化为了图中每一条边的具体数值

该趣味性指标的定义综合考虑了路径本身的几何特征和视觉体验变化两个维度，这与游客在园林中游览时的实际体验高度契合。路径特征通过长度、转折点和交叉点数量来刻画行走的复杂性和多样性，而异景程度则直接反映了中国传统园林” 步移景异” 的设计理念。其次，模型采用归一化处理将不同量纲的特征统一到 [0,1] 区间，确保了各指标的可比性和可加性。

最终的趣味得分见表4.2

表 4.2　园林趣味性总分表

名称	拙政园	留园	寄畅园	瞻园	瞻园	豫园	秋霞圃	沈园	怡园	耦园	绮园
趣味性	0.94	0.84	0.84	0.80	0.79	0.86	0.88	0.82	0.81	0.75	0.78

根据表4.2所示的趣味性评估结果，拙政园以 0.94 的得分位居首位，这一结果与其作为中国四大名园之一的地位相符。拙政园以其广阔的水域、丰富的景观层次和精巧的路径设计而闻名，路径特征与异景程度均表现优异。秋霞圃（0.88）和寄畅园（0.84）等园林也获得了较高评分，反映了这些园林在营造游览趣味性方面的成功实践。

值得注意的是，趣味性得分与园林的知名度存在较高一致性，但并非完全正比关系。例如，留园（0.84）虽规模不及拙政园，但其精巧的空间组织和" 移步异景" 的设计理念使其获得了优异的评价。这一评估结果验证了本文提出的趣味性量化模型的合理性，表明该模型能够有效捕捉中国传统园林在游览体验设计中的精髓。权重系数设定为 $\alpha = 0.4$、$\beta = 0.6$，强调了" 异景程度" 在游览趣味性中的主导作用，这与园林设计中重视视觉体验变化的理论导向相一致。

4.6 多目标价值最优的关键游线提取模型

在完成了对园林路径网络和景观变化的量化分析后，接下来的核心问题是如何在复杂的路网中，规划出最具“趣味性”的游览线路 [7]。一条优秀的游线不仅应串联起关键景点，更应在游览过程中提供丰富的动态体验和空间探索感。为此，建立一个基于多目标价值最优的关键游线提取模型，从所有可能的路径中筛选出综合价值最高的游览线路。

该模型的构建并非基于单一指标，而是综合考量了由路径特征和异景程度定义的趣味性以及路径重复性，并通过加权求和的方式构建了一个综合性的评价体系。

4.6.1 游线综合评分

在定义了边的趣味性后，需要建立一个评价函数来评估一条完整游线 P 的全局综合趣味性。一条理想的游线应尽可能多地覆盖景点、探索更广阔的园林空间，并由高价值的路径片段构成。因此，游线的综合评分 Q 被设计为一个包含三个核心指标的加权模型：

- **景点覆盖率** r_{cover}：路径所经过的独特景点数量与园林总景点数量的比值。这是衡量游线核心价值的最基本指标，表示为：

$$r_{cover} = \frac{N_P}{N_{total}} \tag{4.20}$$

  其中，N_P 是路径 P 所经过的景点的数量；N_{total} 是园林中所有景点的总数量。

- **空间探索度** $r_{explore}$：路径所覆盖的空间范围。此处采用路径经过所有节点的坐标点所构成的凸包面积来近似度量，并对其进行归一化处理。较大的覆盖面积意味着游客探索了园林中更广阔的区域，表示为：

$$r_{explore} = \frac{A_P}{A_{total}} \tag{4.21}$$

  其中，A_P 是路径 P 上所有节点构成的凸包面积；A_{total} 是预估的园林总面积，作为一个归一化的基准值。

- **趣味性累积** V_{path}：路径上所有边的价值总和，并进行归一化。它反映了路径在局部趣味性上的整体表现，表示为：

$$V_{path} = \frac{\sum_{e \in P} V_e}{V_{max}} \tag{4.22}$$

  其中，$\sum_{e \in P} V_e$ 是路径 $\mathcal{P}$ 中所有边的价值 V_e 的累加和；V_{max} 是预设的归一化最大期望价值。

最终，游线的**综合评分** Q 是上述三个归一化指标的加权和：

$$Q = \omega_1 \cdot r_{cover} + \omega_2 \cdot r_{explore} + \omega_3 \cdot V_{path} \tag{4.23}$$

其中，$\omega_1, \omega_2, \omega_3$ 是各自的权重系数，在本文，其值分别选用为 0.3, 0.3, 0.4，反映了在最终评价中对景点数量、探索范围和路径本身趣味性的不同侧重。

4.6.2 关键游线搜索算法

寻找最佳的游线路径可以转化为如下最优化问题

$$\begin{aligned}
\max \quad & Q(P) = \omega_1 \cdot r_{cover}(P) + \omega_2 \cdot r_{explore}(P) + \omega_3 \cdot V_{path}(P) \\
\text{s.t.} \quad & v_1 \in E, \quad E = v \in V \mid \text{type}(v) = \text{entrance} \\
& v_n \in E, \quad v_n \neq v_1 \\
& 3 \leq n \leq 12 \\
& |A_P| \geq 2 \\
& \forall v \in V, \quad \sum_{i=1}^{n} 1_{\{v_i = v\}} \leq 2
\end{aligned} \tag{4.24}$$

其中：$P = (v_1, v_2, \ldots, v_n)$ 表示游览路径，$v_i \in V$（节点集合）权重系数：$\omega_1 = 0.3, \omega_2 = 0.5, \omega_3 = 0.2$

由于需要平衡多个目标（如路径总价值、景点数量、不重复性等），传统的 Dijkstra 或 A* 算法难以直接适用。因此，采用一种**基于优先队列的启发式搜索算法**来寻找近似最优解。

该算法从一个指定的入口节点开始，使用一个优先队列来管理待扩展的路径。队列中的每个元素不仅包含当前路径，还存储了该路径的综合评分、已访问的景点集合、路径长度等状态信息。算法的核心思想如下：

1. **初始化**：将仅包含入口节点的路径放入优先队列。
2. **迭代扩展**：循环从队列中取出综合评分最高的路径进行扩展。对于当前路径的末端节点，遍历其所有邻居节点。
3. **动态评估与剪枝**：对于每一个扩展出的新路径，动态计算其临时综合评分。这个评分会给予访问**新景点**额外的奖励，同时对**重复访问**已走过的节点或景点施加轻微惩罚。这种动态奖励与惩罚机制，引导搜索算法倾向于探索更广阔、更多样化的路径。
4. **路径约束**：为了保证路径的合理性，设置了最大路径长度和节点重复访问次数的上限，对搜索空间进行剪枝，避免产生无意义的冗长或循环路径。
5. **终止条件**：当搜索到的满足特定条件（如到达指定出口、访问了足够数量的景点）的路径达到预设数量时，或优先队列为空时，搜索终止。

通过该算法，可以高效地在庞大的路径组合空间中，搜索出若干条在景点覆盖、空间探索和路径趣味性之间达到良好平衡的关键游览线路，为游客提供高质量的游园体验方案。

算法 1 基于优先队列的关键游线搜索算法

输入: 图 G，入口节点 $start$，最大路径长度 $maxLen$，节点重复访问上限 $maxRevisit$，目标路径数量 $targetCount$

输出: 满足条件的路径集合 $solutions$

初始化优先队列 pq（按综合评分降序排列）
初始化空集合 $solutions$
创建初始路径 $path$，仅包含 $start$
计算初始路径的综合评分 $score$
将 $(path, score, \{start\}, 1)$ 加入 pq ▷ 包含路径、评分、已访问景点、长度
while pq 非空且 $|solutions| < targetCount$ **do**
    从 pq 中取出综合评分最高的路径状态 $current$
    if $current.path$ 满足终止条件（如到达出口或访问足够景点） **then**
        将 $current.path$ 加入 $solutions$
        continue ▷ 跳过扩展，继续下一轮循环
    for 每个邻居节点 $neighbor \in G.adjacent(current.lastNode)$ **do**
        创建新路径 $newPath = current.path \cup [neighbor]$
        计算新路径长度 $newLen = current.len + 1$
        检查节点重复访问次数 $visitCount$
        if $newLen > maxLen$ 或 $visitCount > maxRevisit$ **then**
            continue ▷ 路径约束剪枝
        更新已访问景点集合 $newAttractions = current.attractions$
        初始化评分变化 $\Delta score = 0$
        if $neighbor$ 是新景点且未访问过 **then**
            $\Delta score \mathrel{+}= reward$ ▷ 动态奖励新景点
        else if $neighbor$ 已访问过 **then**
            $\Delta score \mathrel{-}= penalty$ ▷ 轻微惩罚重复访问
        计算新综合评分 $newScore = current.score + \Delta score$
        将 $(newPath, newScore, newAttractions, newLen)$ 加入 pq
return $solutions$

4.7 模型求解

在完成关键游线提取模型构建后，得到了模型构建的**游线图4.7-图4.11**，其中带标号的圆形区域代表游线的到达顺序；带标号的菱形区域代表在整条游线上异景程度最高的点，记为异景关键点，其选取规则为：选取游线上异景程度最高的 6 个点，剔除空间位置上相近的点（相近点往往各种指标相似，参考价值较低）。

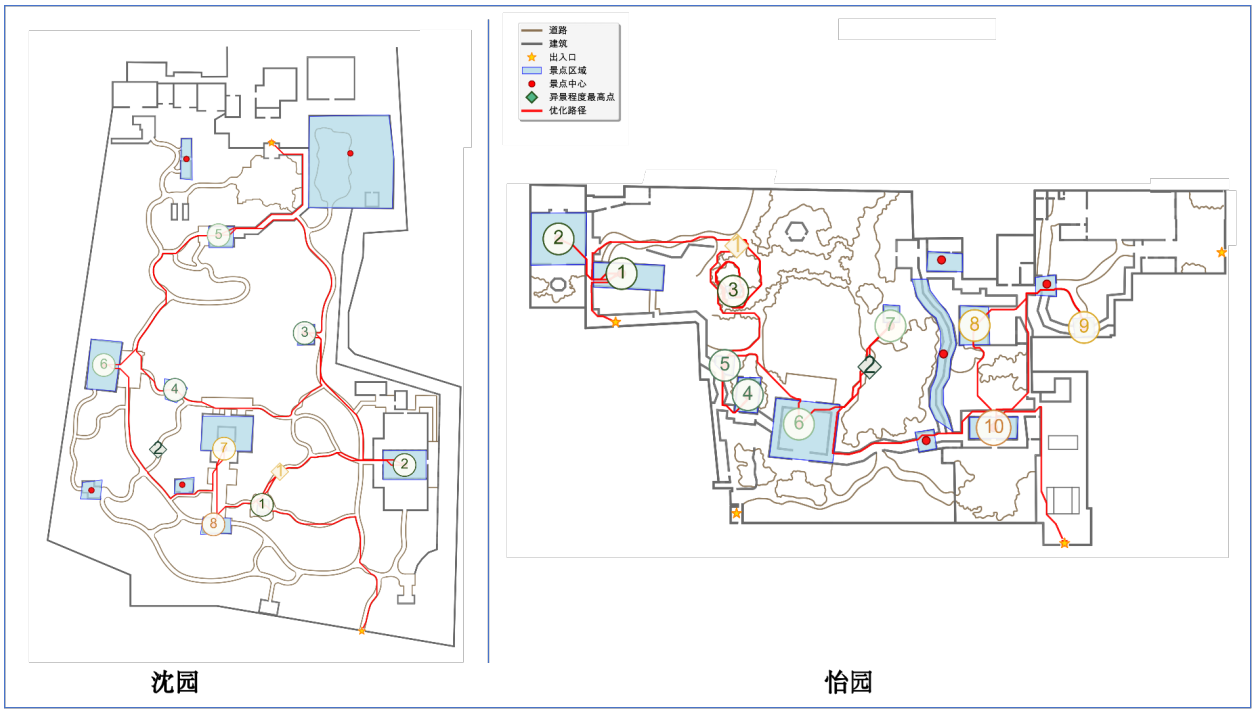

图 4.7　沈园与怡园游线展示

表4.3展示了各游线上所选取的异景程度得分最高的点的平均得分、最高得分以及最低得分，衡量了各园在游线路径上的**异景程度得分**。

表4.4展示了各游线通过计算得到的**路径特征**。

图4.12展示了各园的**游线综合得分**。

5 问题二模型建立与求解

5.1 问题分析

“小中见大”的布局“幻境感”建模

第一问得到密集图所有路径作为园林的可到达区域。对所有路径进行均匀采样，用得到的点作为附近空间的中心点，通过游客在该点上的所看到的景观元素来统计和计算可视元素，进一步作为空间的元素特征。具体来说，在每个点上进行视域分析，得到每个点视域特征。之后点空间位置使用 K-means 算法聚类得到园林的景观分区。由于园林的同一景观

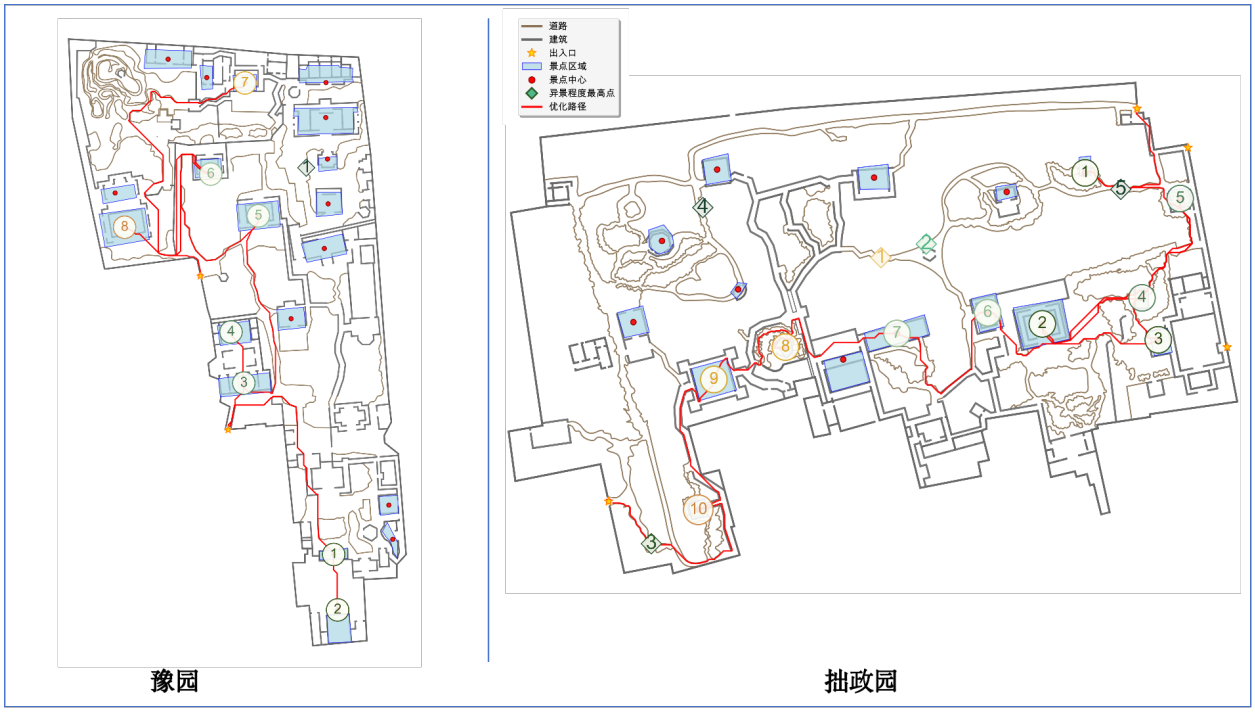

图 4.8　豫园与拙政园游线展示

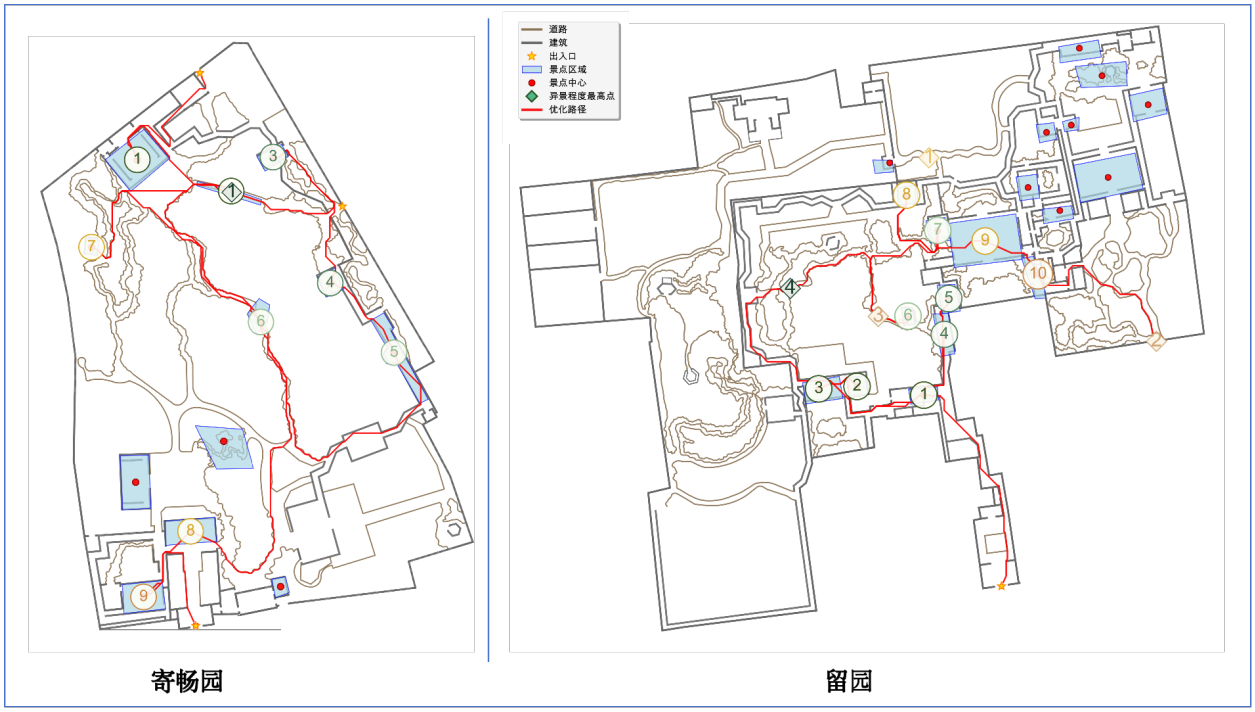

图 4.9　寄畅园与留园游线展示

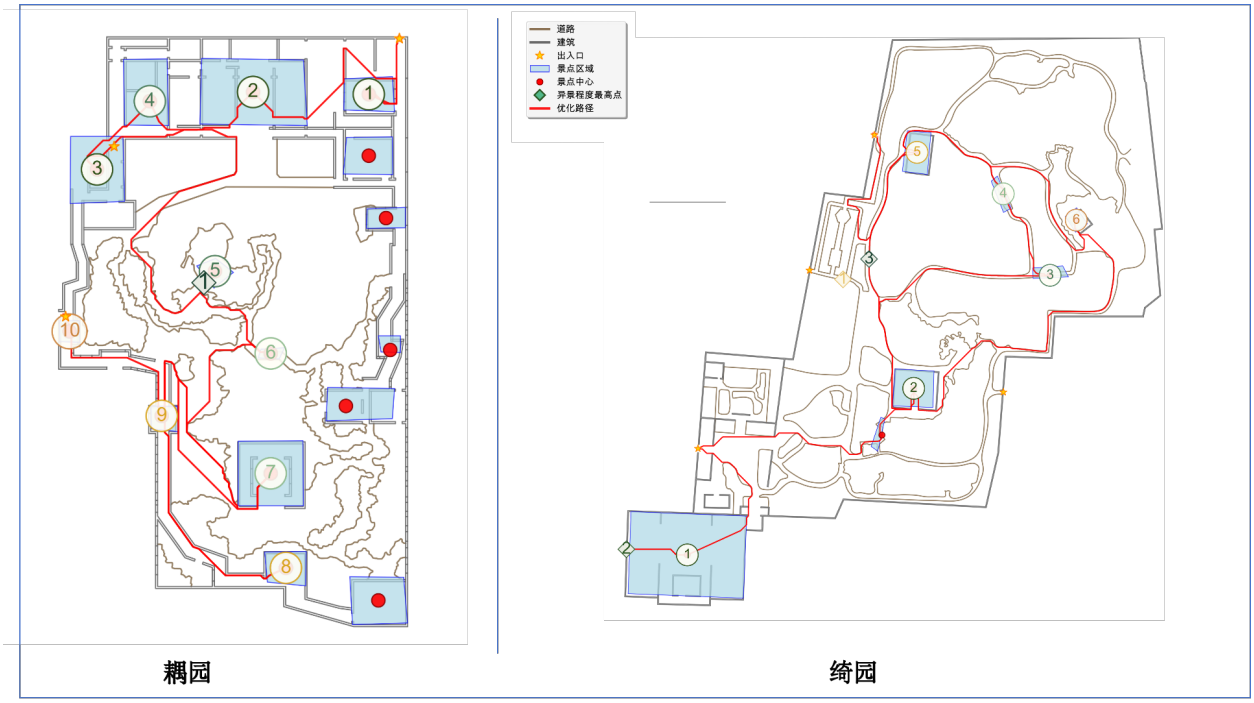

图 4.10　藕园与绮园游线展示

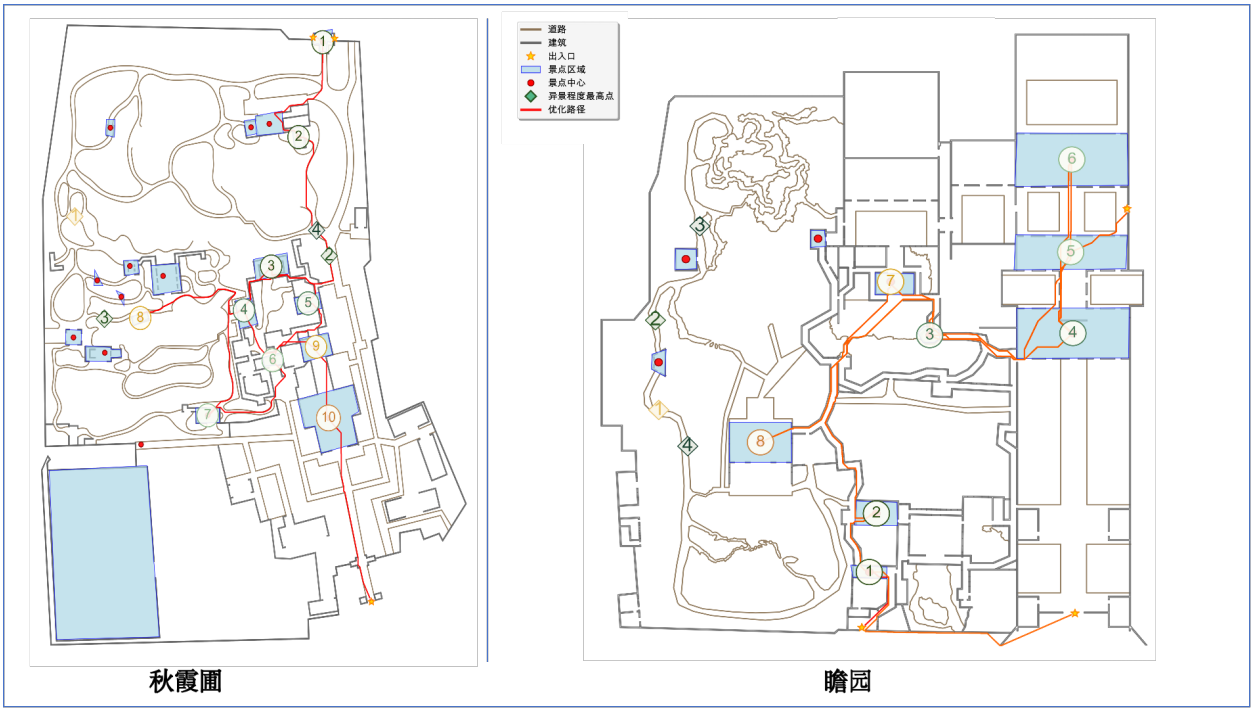

图 4.11　秋霞圃与瞻园游线展示

表 4.3　园林异景程度得分

排名	园林名称	平均异景得分	最高异景得分	最低异景得分	异景关键点数量
1	怡园	0.930	0.936	0.924	2
2	沈园	0.895	0.897	0.892	2
3	留园	0.858	0.874	0.823	4
4	寄畅园	0.857	0.857	0.857	1
5	瞻园	0.844	0.893	0.819	4
6	绮园	0.842	0.856	0.830	3
7	耦园	0.838	0.838	0.838	1
8	豫园	0.838	0.838	0.838	1
9	秋霞圃	0.825	0.843	0.813	4
10	拙政园	0.814	0.832	0.803	5

表 4.4　园林路径特征

园林名称	转折点数	交叉点数	路径长度 (m)
绮园	10	31	690.06
拙政园	7	23	669.67
沈园	10	29	735.14
寄畅园	8	7	639.90
瞻园	6	40	463.69
豫园	10	74	966.06
秋霞圃	6	3	530.90
留园	7	38	581.92
怡园	9	3	531.68
耦园	7	26	377.99

并不分布在空间的相邻区域，而是可能离散的出现在不同的空间位置。本文使用 DBSCAN 在 K-means 聚类的景观分区的基础上，基于点的空间密度，进一步实现景观的子空间区域的划分。使用上述提出的两阶段聚类算法，得到了园林主要景观类型的分区和每个景观类型的子空间区域，并且拥有每个区域内观测点所看到元素集合。

开阔与围合是此消彼长的两个概念，故使用开阔度一个指标统一来衡量。在上一小问的基础上，得到了每个采样点的视域特征，并且通过二次聚类得到空间的划分。将开阔度定义为相关视域特征的加权和，之后使用视域特征计算每个景区子空间的开阔度。

根据题目对于幻境感的描述，幻境感主要受到两个关键因素影响：路径上的开阔围合的变化（开阔围合的交替出现增加幻境感）和路径上景观变换（景观的不断变化增加幻境

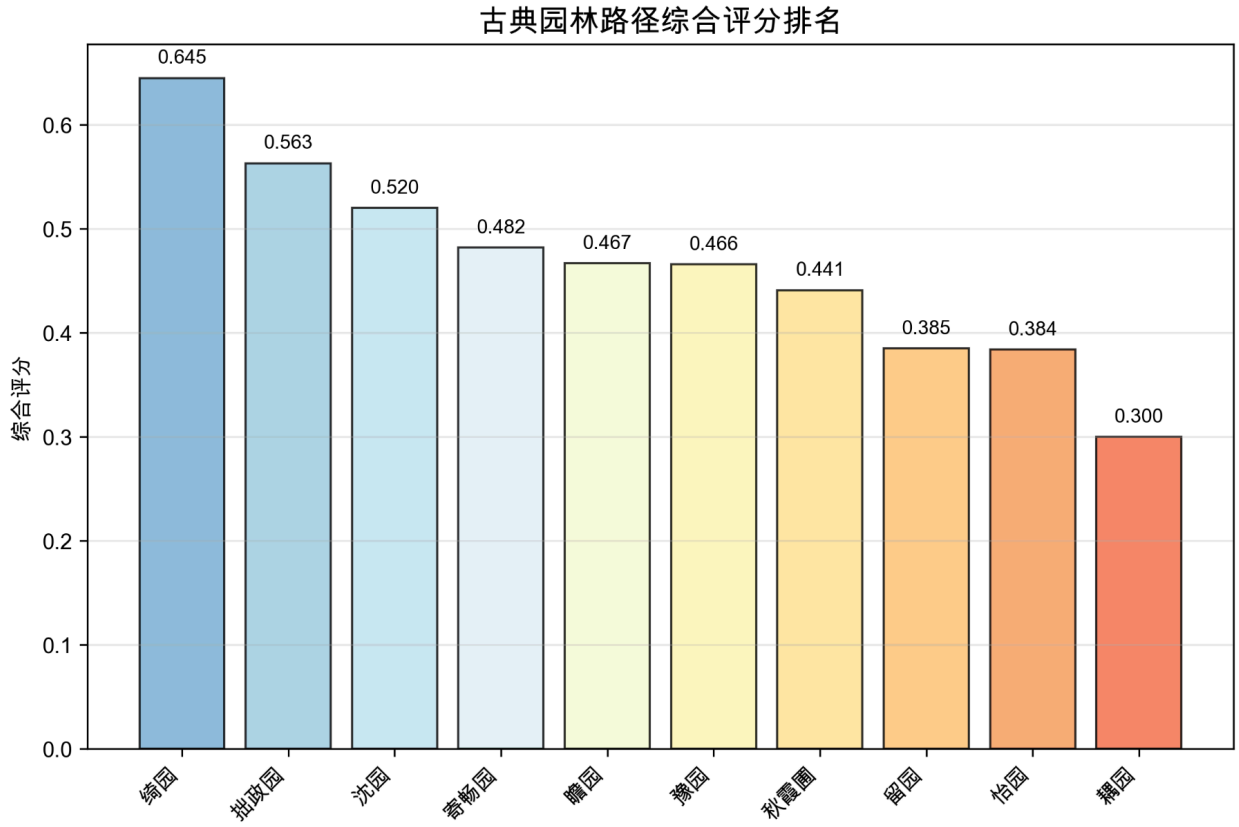

图 4.12　游线综合评分对比

感）。针对这两个关键因素，本文设计了路径的主题多样性、开合节律变化和要素配比三个指标，三个指标共同构成了幻境感的量化表达。同时，将第一的寄畅园作为标准，使用熵权法构建计算三个指标之间的参考权重，并在其他园林进行得分计算。

问题二框架图见图5.1。

5.2 元素分布

为客观描述园林的空间布局结构，采用以游赏者感知视角为基础的分析方法，通过量化其在园内不同位置的视觉体验，构建景观元素的分布模型。此过程主要分为视域特征提取、基于景观特征的初步聚类和基于空间位置的二次聚类三个步骤。

5.2.1 视域特征的量化提取

对于观测点与视域分析的定义参考章节4.4.2, 最终得到一个遍布全园的观测点集合 $P = \{p_1, p_2, \ldots, p_N\}$，其中每个观测点 p_i 的坐标为 (x_i, y_i)。

基于视域多边形 V_i 为每个观测点 p_i 提取一个多维特征向量 $\mathbf{f}_i$，其构成如下：

$$\mathbf{f}_i = [A(V_i), R_{\text{vis}}(i), C_{\text{wall}}(i), C_{\text{plant}}(i), C_{\text{rock}}(i), C_{\text{water}}(i), \ldots] \tag{5.1}$$

其中 $A(V_i)$ 为可视多边形 V_i 的面积，反映视野的开阔程度；$R_{\text{vis}}(i)$ 为视域通透率，定义为

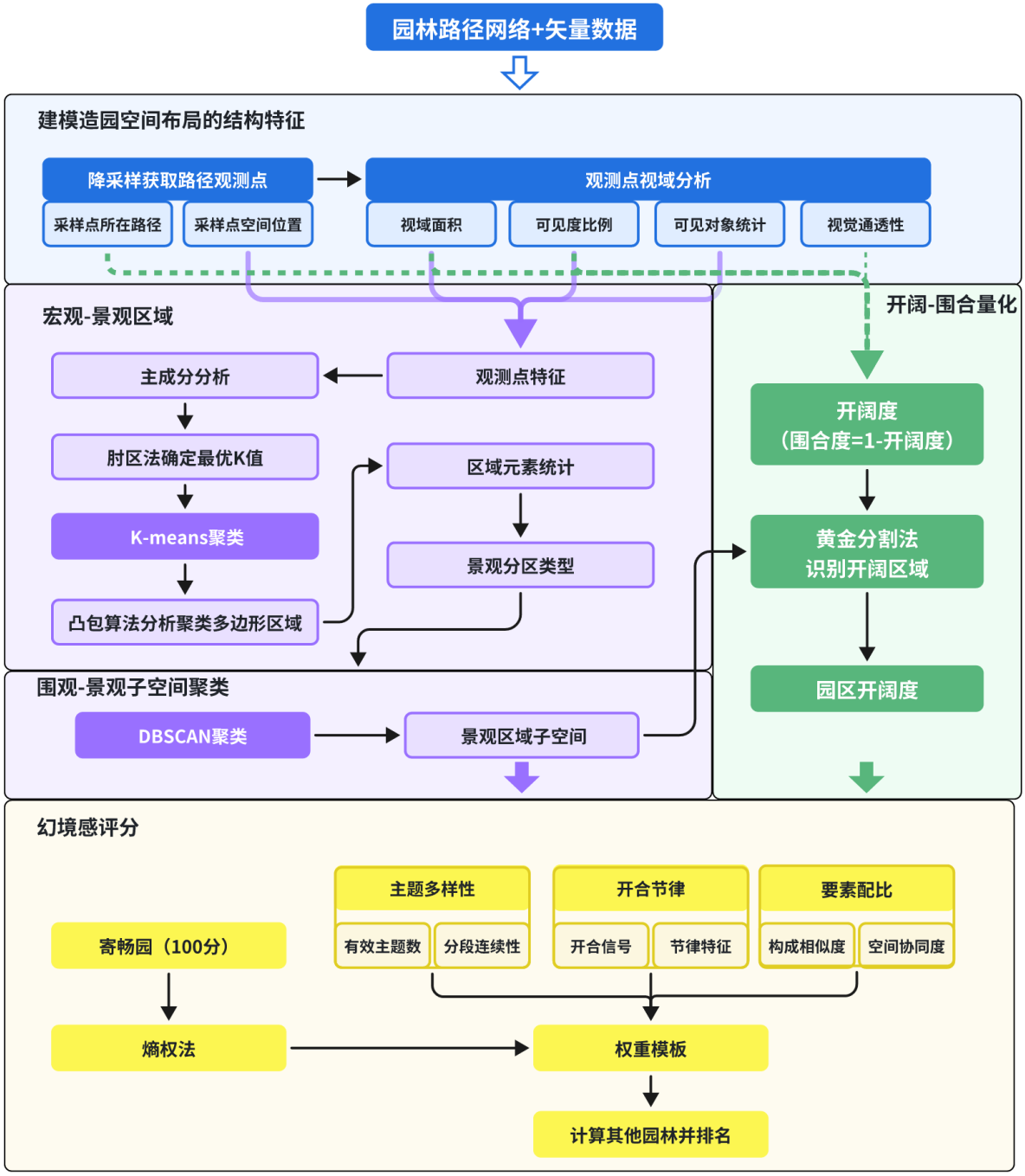

图 5.1　问题二框架图

未被遮挡的射线数量与总射线数量 K 的比值，衡量视野的连续性；$C_{\text{type}}(i)$ 表示在可视多边形 V_i 范围内，可见的某类型景观元素的数量。通过此方法，将每个观测点的抽象视觉体验，转化为了一个可供计算的、标准化的特征向量。

视域分析可视化结果呈现在图5.2中，图中的每个圆形范围即是以圆心为观测点的可视范围。

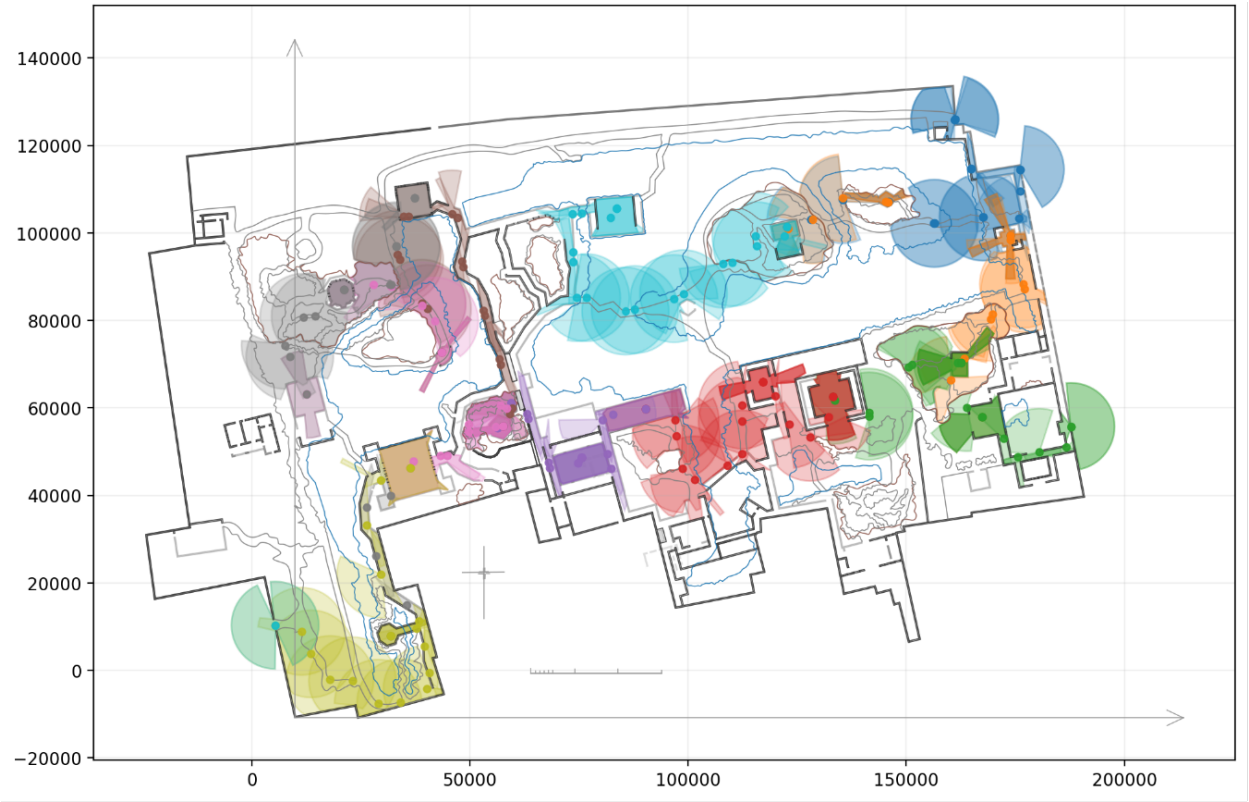

图 5.2　拙政园所有路径视域分析结果

5.2.2 基于 K-means 的景观分区提取

在获得所有观测点的特征向量集合 $F = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_N\}$ 后，采用 K-means 聚类算法对这些点进行分类，旨在将具有相似视觉体验的观测点归为一类，从而形成具有不同审美主题的宏观景观分区。

由于 K-means 算法对特征的尺度敏感，首先对特征矩阵进行标准化处理。对于每个特征维度 j，其标准化后的值 $f'_{i,j}$ 计算如下：

$$f'_{i,j} = \frac{f_{i,j} - \bar{f}^{(j)}}{\sigma^{(j)}} \tag{5.2}$$

其中 $\bar{f}^{(j)}$ 和 $\sigma^{(j)}$ 分别是第 j 个特征在所有观测点上的均值和标准差。

聚类的目标是找到 K 个聚类中心 $\{\mu_1, \mu_2, \ldots, \mu_M\}$，使得所有点到其所属聚类中心的平方误差和最小化。目标函数应用于标准化后的特征向量 $\mathbf{f}'_i$：

$$\min \sum_{j=1}^{K} \sum_{\mathbf{f}'_i \in S_j} ||\mathbf{f}'_i - \mu_j||^2 \tag{5.3}$$

其中，S_j 是第 j 个聚类簇的集合，μ_j 是该簇的质心。通过肘部法则确定最优聚类数 M。最终，每个观测点 p_i 被赋予一个景观聚类标签 $L_i \in \{1, 2, \ldots, M\}$。通过分析每个聚类簇的

质心 μ_j（即各类景观特征的平均值），为每个簇赋予语义化名称，如将“可见水体”特征值显著高的簇命名为“水景区域”。

在通过 K-means 算法将观测点划分到不同的景观聚类 S_j 后，为了在物理空间上界定这些功能分区的范围，为每个聚类引入了凸包算法来生成其几何边界。

对于一个给定的观测点集合 S_j，其凸包 $H(S_j)$ 是包含该集合中所有点的最小凸多边形。其数学定义为点集 S_j 的所有凸组合的集合：

$$H(S_j) = \left\{ \sum_{i=1}^{m} \alpha_i p_i \mid p_i \in S_j, \alpha_i \geq 0, \sum_{i=1}^{m} \alpha_i = 1 \right\} \tag{5.4}$$

其中，α_i 是分配给点 p_i 的非负权重系数，其总和为 1。采用高效的 Quickhull 算法来确定构成凸包的顶点，并构建一个代表景观区域边界的多边形。然而，当聚类中的点数过少（$m < 3$）时，无法形成多边形。针对这些特殊情况，采用以下策略：

- **单点簇 ($m = 1$):** 将该点视为区域中心，生成一个预设半径为 r 的圆形缓冲区 $B(\mathbf{p}_1, r)$ 作为其影响范围。
- **两点簇 ($m = 2$):** 将两点连成的线段作为轴心，生成一个预设半径为 r 的缓冲区，形成一个胶囊状区域。

通过此方法，每个景观聚类 S_j 都对应一个确定的空间多边形。这些多边形在地图上直观地展示了各类景观的分布范围。图5.3表示 K-means 聚类以及凸包算法的结果。

5.2.3 基于 DBSCAN 的景观子空间提取

初步聚类仅依据景观特征，未考虑空间分布。然而，同一景观主题（如水景）在园林中可能并非是单一连续的区域，而是散布于不同位置。为识别这些空间上独立的子区域，在初步聚类的基础上，进一步采用基于密度的 DBSCAN（Density-Based Spatial Clustering of Applications with Noise）算法。

DBSCAN 算法根据样本的分布密度进行聚类，它需要两个核心参数：邻域半径 ϵ 和构成核心点的最小邻域样本数 n。与第一步不同，此阶段的聚类对象是观测点的物理坐标 (x_i, y_i)，而非其景观特征向量。算法从任意点开始，如果其 ϵ-邻域内的点数不少于 n，则将其标记为核心点并创建一个新的簇。然后，递归地将所有从该核心点密度可达的点加入此簇。

通过 DBSCAN，每个观测点 p_i 被赋予一个空间聚类标签 $S_i \in \{-1, 0, 1, \ldots, Q\}$，其中 Q 是识别出的空间子区域数量，标签值为-1 的点被视为不属于任何簇的噪声点。

最终，通过结合上述两阶段聚类的结果，每个观测点 p_i 同时拥有一个景观聚类标签 L_i 和一个空间聚类标签 S_i。这能够精确地描述园林的布局结构：不仅识别出园内存在哪些审美主题，还能定位到每个主题在空间上具体分布的、一个或多个独立的物理区域。图5.4表示 DBSCAN 聚类的结果。

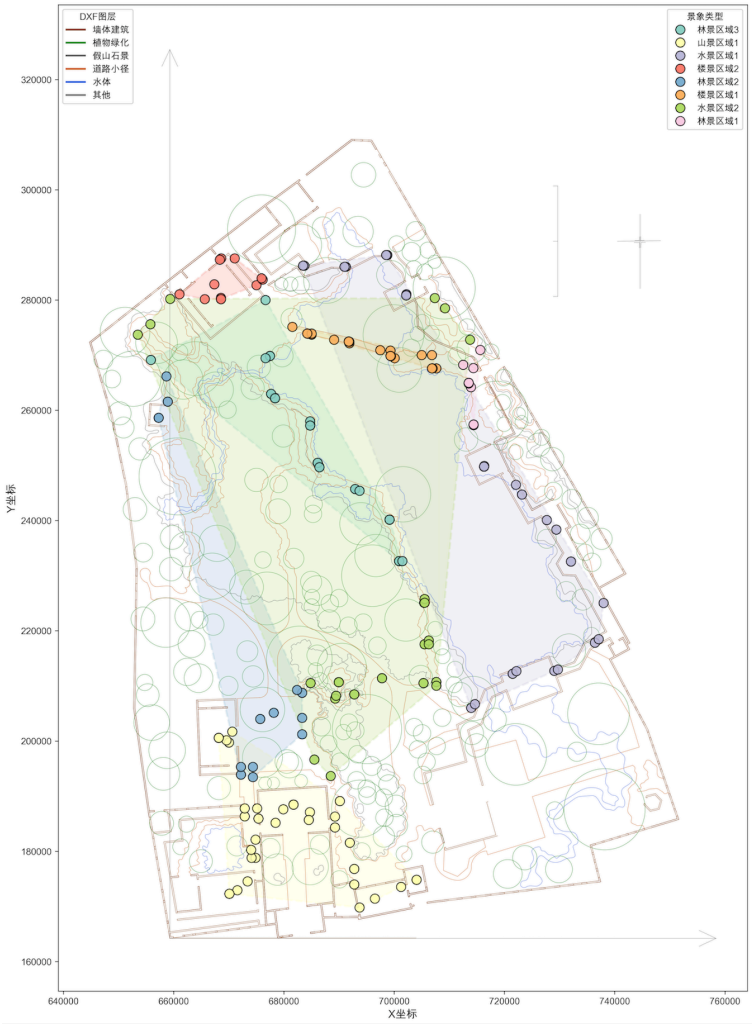

图 5.3　K-means 聚类结果（以寄畅园为例）

5.2.4 景观空间分布可视化

5.3 开合变化

园林空间的“开合变化”是营造“幻境感”的关键手法，它指的是在游赏过程中，空间在视觉感受上的开放与封闭之间的交替与对比。为对此进行量化，建立一个统一的评价体系来描述任意空间点的开合程度。鉴于开阔与围合是同一空间感知谱系的两端，故可构建“开阔度”与“围合度”两个核心指标来综合评价 [8]。

5.3.1 开阔度与围合度的量化标准

基于前文提取的视域特征，可为每个观测点 p_i 定义其开阔度与围合度。

1. **核心指标定义**:
   开阔感主要源于广阔的视野和较少的视觉阻碍，而围合感则来自于清晰的边界和较高的视觉封闭性。基于此，从视域特征中选取以下关键指标 [9]:

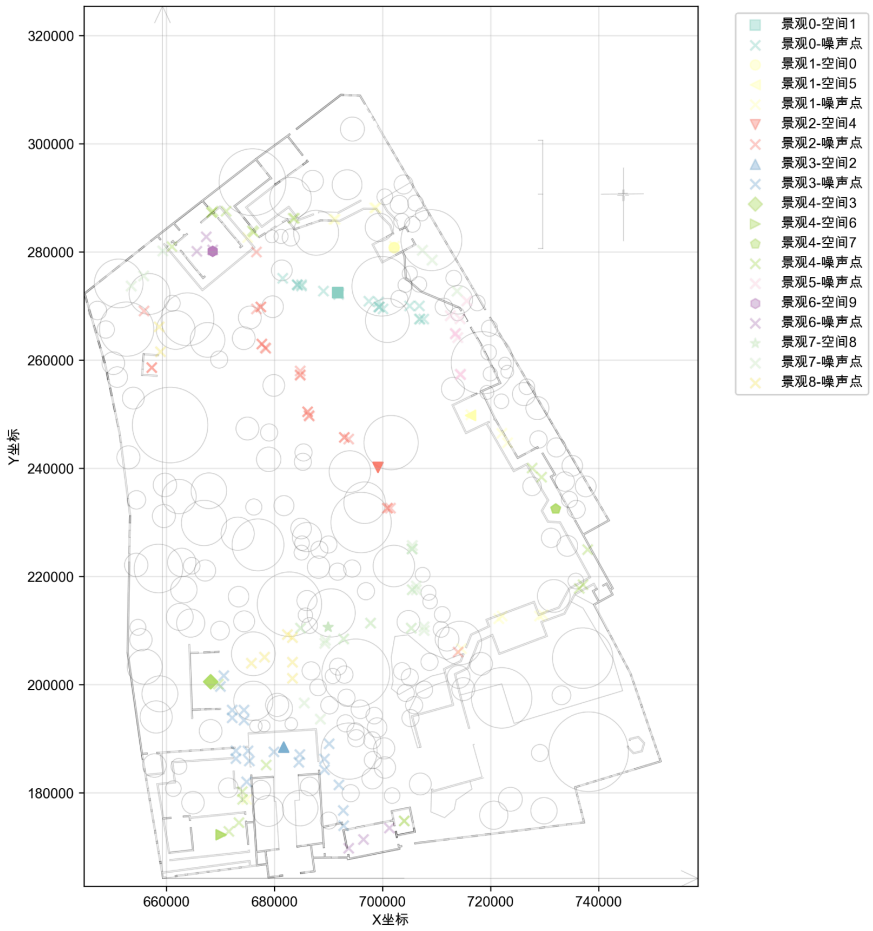

图 5.4　DBSCAN 聚类结果（以寄畅园为例）

- **视域面积 (A_i)**: 观测点可视多边形的面积。面积越大，物理空间越开阔。
- **视觉通透率 (R_v)**: 未被遮挡的视线光线比例。比率越高，视线穿越性越好，感觉越开阔。
- **视觉渗透性 (P_v)**: 定义为视域面积与可见对象总数的比值，即 $P_v = A_i/(C_{total} + 1)$，其中 C_{total} 为可见对象总数。该指标衡量单位景观元素的空间贡献，值越大表示空间越疏朗。
- **视线阻碍度 (O_v)**: 定义为 $1 - R_v$，与视觉通透率互补，直接反映视线的受阻程度。
- **遮挡物密度 (D_b)**: 定义为主要遮挡物（建筑、山石、植物）数量与视域面积的比值，即 $D_b = (C_{wall} + C_{rock} + C_{plant})/(A_i + 1)$。该值越高，围合感越强。
- **边界限定度 (D_{bound})**: 定义为主要边界元素（建筑、植物）在所有可见对象中的占比，即 $D_{bound} = (C_{wall} + C_{plant})/(C_{total} + 1)$。该值越高，空间的边界感越清晰，围合感越强。

2. **指标归一化**:

为消除不同指标间的量纲差异，需对部分指标进行最小-最大值归一化处理。对于任意

指标 X，其归一化后的值 X_{norm} 计算如下：

$$X_{norm} = \frac{X - X_{\min}}{X_{\max} - X_{\min} + \epsilon} \tag{5.5}$$

其中 $X_{\min}$ 和 $X_{\max}$ 分别为该指标在所有观测点中的最小值和最大值，ϵ 是一个极小的正数以避免分母为零。本文对 A_i, P_v, D_b 和 D_{bound} 进行归一化。

3. **综合指标构建**:

通过对上述核心指标进行加权求和，构建“开阔度指数” (I_{open}) 和“围合度指数” (I_{encl})。权重系数依据各指标对开阔或围合感知的贡献度设定。

开阔度指数主要由积极影响开阔感的因素构成：

$$I_{open} = \omega_1 \cdot A_{vs,norm} + \omega_2 \cdot R_{vis} + \omega_3 \cdot P_{vis,norm} \tag{5.6}$$

其中，$\omega_1, \omega_2, \omega_3$ 为权重系数，取值为 $\omega_1 = 0.4, \omega_2 = 0.3, \omega_3 = 0.3$。

围合度指数则由积极影响围合感的因素构成：

$$I_{encl} = v_1 \cdot O_{vis} + v_2 \cdot D_{block,norm} + v_3 \cdot D_{bound,norm} \tag{5.7}$$

其中，v_1, v_2, v_3 为权重系数，取值为 $v_1 = 0.4, v_2 = 0.3, v_3 = 0.3$。

通过上述公式，可以计算出园内任意一个观测点 p_i 的开阔度与围合度得分，这两个得分共同构成了对该点空间感受的量化描述。

5.3.2 空间类型的划分

为了从宏观上理解园林的空间布局，可根据开阔度与围合度指数对空间进行分类。利用所有观测点得分的分位数来确定分类阈值。例如，将所有得分按从低到高排序，以 33% 和 67% 分位点作为“低-中”和“中-高”的分割线。由此，可定义四种典型的空间类型 [10]：

- **开敞型空间**: 高开阔度 (I_{open} > 67% 分位数) 且低围合度 (I_{encl} < 33% 分位数)。
- **围合型空间**: 低开阔度 (I_{open} < 33% 分位数) 且高围合度 (I_{encl} > 67% 分位数)。
- **半开敞/半围合空间**: 中等开阔度与中等围合度。
- **复合型空间**: 其他组合，如高开阔度与高围合度并存，可能代表视线虽远但边界感强的特殊空间。

5.3.3 景区子空间的开合评价

在对园林进行景观分区后，每个景区子空间 Z_k 包含了若干观测点。该子空间的整体开合感受，可由其内部所有观测点开阔度/围合度指数的统计平均值来表示。

$$\bar{I}_{open}(Z_k) = \frac{1}{|Z_k|} \sum_{p_i \in Z_k} I_{open}(p_i) \tag{5.8}$$

$$\bar{I}_{encl}(Z_k) = \frac{1}{|Z_k|} \sum_{p_i \in Z_k} I_{encl}(p_i) \tag{5.9}$$

其中 $|Z_k|$ 是子空间 Z_k 内的观测点数量。这两个平均值 $\bar{I}_{open}(Z_k)$ 和 $\bar{I}_{encl}(Z_k)$ 共同量化了第 k 个景区子空间的整体空间氛围，为后续分析园林整体的“开合变化”节奏与“幻境感”评分提供了基础。

5.4 幻境感的量化评分模型

根据题意，“幻境感”源于在有限空间内通过场景的巧妙组织与转换，营造出超越物理空间的感知体验。这种感受主要受到两个关键因素影响：一是游赏路径上“开”与“合”的空间节奏变化，二是沿途景观主题的丰富性与变换频率。

为对“幻境感”进行量化，本文设计了一个综合评分模型。该模型包含三个核心评价维度：**主题多样性**、**开合节律**和**要素配比**。以寄畅园作为“江南第一园”的完美范例，将其幻境感设定为 100 分基准。通过熵权法确定各指标的客观权重，并建立一个统一的、可泛化的评分体系。

5.4.1 幻境感评价指标体系

对于任一园林，首先沿其游览路径进行采样，获得一个包含空间与景观特征的有序观测点序列 $S = \{p_1, p_2, \ldots, p_N\}$。基于此序列，计算以下三个指标：

- **主题多样性 (D_t)**

  该指标衡量游赏过程中所经历的景观主题的丰富程度与变换的流畅性。

  - **有效主题数 (N_e)**: 首先统计路径序列 S 中各个景观主题（由 2.1 节聚类得到）出现的频率，计算其概率分布 $\{q_1, q_2, \ldots, q_M\}$。利用香农熵计算主题分布的信息熵 H：

    $$H = -\sum_{j=1}^{M} q_j \log_2(q_j) \tag{5.10}$$

    有效主题数定义为 $N_e = e^H$。N_e 越高，表明游客能体验到的景观主题种类越均衡和丰富。

  - **分段连续性 (P_f)**: 为避免主题过于频繁切换导致的破碎感，引入分段连续性因子。该因子奖励长度适中的连续主题片段。计算路径上所有连续主题片段的平均长度 $\bar{L}$，并定义连续性因子为：

    $$P_f = \frac{\bar{L}}{\bar{L} + L_0} \tag{5.11}$$

    其中 L_0 为一个常数（如 10），用于调节曲线的饱和度。

  最终，主题多样性指标的原始分由两者相乘得到：

  $$S'_t = N_e \cdot P_f \tag{5.12}$$

- **开合节律 (R_{rhy})**

  该指标量化路径上开阔与围合交替变化的节奏感与强度。

– **开合信号 ($K(t)$)**: 定义在路径点 p_t 上的开合信号为其开阔度指数与围合度指数之差：

$$K(t) = I_{open}(p_t) - I_{encl}(p_t) \tag{5.13}$$

正值表示偏开阔，负值表示偏围合。

– **节律特征**: 对开合信号序列 K 提取四个核心特征：

1. **振幅 (A)**: 信号的标准差 $\sigma(K)$，反映开合变化的整体强度。
2. **过渡频率 (T)**: 信号穿越零点的次数，反映开合状态切换的频繁程度。
3. **总能量 (E)**: 信号一阶差分的绝对值之和，$\sum |K(t) - K(t-1)|$，反映变化的剧烈程度。
4. **节律规整度 (R)**: 过渡间隔时长的变异系数的倒数，反映开合变化的周期性。

– **评分函数**: 以寄畅园的各项节律特征值 $\{A_{ref}, T_{ref}, E_{ref}, R_{ref}\}$ 为最优峰值，采用高斯响应函数对其他园林的特征值进行评分。例如，对于振幅 A，其得分为：

$$f_A = \exp\left(-\frac{(A - A_{ref})^2}{2\sigma_A^2}\right) \tag{5.14}$$

其中 σ_A 是控制容忍度的宽度参数。开合节律的原始分为各特征得分的加权和：

$$S'_k = w_A \cdot f_A + w_T \cdot f_T + w_E \cdot f_E + w_R \cdot f_R \tag{5.15}$$

本文将其权重 w 统一定义为 0.25。

- **要素配比 ($C_{element}$)**

该指标评价景观元素（山、水、植物、建筑）的构成比例是否符合江南园林的普遍审美范式，以及元素类型是否与所在空间的开合感受相协调。

– **构成相似度 (S_{sim})**: 计算园林路径上可见的各类元素（水、石、植物、建筑）的总体比例向量 $\mathbf{q}$。以寄畅园的元素比例向量 $\mathbf{q}_{ref}$ 为参考，使用 L1 距离计算相似度：

$$S_{sim} = \max\left(0, 1 - \frac{1}{2}\sum_j |q_j - q_{ref,j}|\right) \tag{5.16}$$

– **空间协同度 (S_{co})**: 评价元素与空间氛围的匹配程度。在开阔空间（$I_{open} > I_{encl}$）中，水、路等引导性元素的出现应被奖励；在围合空间（$I_{encl} > I_{open}$）中，墙、石等限定性元素的出现应被奖励。协同度是全路径上这种匹配得分的平均值。

要素配比的原始分 S'_m 是构成相似度与空间协同度的加权和：

$$S'_m = \alpha \cdot S_{sim} + \beta \cdot S_{co} \tag{5.17}$$

本文取 α=0.7,β=0.3。

5.4.2 权重确定与幻境感综合评分

1. **熵权法确定权重**:
   为客观地确定三个指标在最终评分中的重要性，采用熵权法（Entropy Weight Method）。该方法基于指标数据的变异程度来分配权重：一个指标在不同园林（或园林内不同分段）间差异越大，其包含的信息量越多，权重也应越高。基于对十个园林数据的分析，计算得到各指标的信息熵，进而得到权重向量 $\mathbf{w} = \{w_t, w_k, w_m\}$。根据计算，权重近似为：$w_t = 0.203, w_k = 0.540, w_m = 0.257$。
2. **归一化与幻境感总分计算**:
   首先，对所有园林的各项原始分 $\{S'_t, S'_k, S'_m\}$ 进行全局最小-最大值归一化，将其映射到 $[0, 1]$ 区间。然后，以寄畅园的归一化得分为基准，进行比例缩放，使得寄畅园的各项分指标得分均为 1.0。对于任意园林 g，其某项指标的最终得分 $S_{g,j}$ 计算如下：

$$S_{g,j} = \frac{(S'_{g,j} - S'_{\text{min},j})/(S'_{\text{max},j} - S'_{\text{min},j})}{(S'_{y,j} - S'_{\text{min},j})/(S'_{\text{max},j} - S'_{\text{min},j})} \tag{5.18}$$

   最后，通过加权求和得到园林 g 的**最终幻境感评分** $Score_g$ 由5.12、5.15、5.17式共同计算得出：

$$Score_g = (w_t \cdot S_{g,t} + w_k \cdot S_{g,k} + w_m \cdot S_{g,m}) \times 100 \tag{5.19}$$

   通过此模型，可以对所有园林的“幻境感”进行量化评价与排序，并能深入分析其在主题、节奏和构成上的具体表现。
3. **幻境感模型合理性分析**: 幻境感量化评分模型的合理性在于，它精准地将一种主观的“感觉”分解为可测量的客观维度。模型紧扣幻境感的核心——即“空间节奏”与“主题变换”，并由此设立了“开合节律”、“主题多样性”和“要素配比”三个关键指标。通过科学的熵权法确定权重，避免了主观臆断，使得最重要的“开合节律”获得了最高权重，这与园林体验中空间收放对人的情绪影响最为直接的理论相符。最终，模型以寄畅园为基准进行校准，其计算结果与这些历史名园的实际声誉和空间特点高度一致（如沈园、瞻园得分高，而布局相对规整的留园得分较低），证明了该模型不仅理论框架严谨，其结果也真实可信，成功实现了对抽象美学体验的客观量化。

5.5 模型求解

表5.1展示了古典园林通过**幻境感**计算出的得分排名，它由主题多样性、开合节律和要素配比综合计算得出。

表 5.1　古典园林空间分析最终排名结果

排名	园林名称	幻境综合评分	主题多样性得分	开合节律得分	要素配比得分
1	寄畅园	100.0	1.000	1.000	1.000
2	沈园	90.9	0.994	0.973	0.707
3	瞻园	90.2	0.766	0.984	0.837
4	耦园	85.5	0.896	0.756	1.030
5	秋霞圃	80.8	0.627	0.863	0.833
6	拙政园	79.1	0.626	0.746	1.014
7	绮园	77.3	0.831	0.882	0.500
8	怡园	77.0	0.500	0.795	0.930
9	豫园	66.3	0.996	0.500	0.743
10	留园	54.8	0.505	0.500	0.683

6 问题三模型建立与求解

6.1 问题分析

江南古典园林” 有法无式” 的美学特质构成了一个复杂的多维度分析问题。所谓” 有法”，指园林设计存在内在的普遍规律和美学法则；而” 无式” 则意味着这些规律不表现为固定的模式，每个园林都有独特的表达方式。这种辩证统一的美学特性要求我们建立能够同时量化共性规律和个性差异的分析框架 [11]。结合前面两问和多模态的数据出发，本文从向量、集合、图结构、文本、有序序列、边界轮廓（建筑、水体、树木、石头）六种模态出发，分别利用余弦相似度、杰卡德相似度、谱相似度、TF-IDF 文本相似度、动态时间规整（DTW）算法、形状上下文算法计算相似度。为每个相似度设计合理的数据结构，具体在6.2中框架图见图6.1。采用改进的 SNF（相似性网络融合）方法将六个视角的相似度矩阵进行非线性融合。通过 KNN 修剪保持局部结构特征，通过迭代式矩阵传播促进全局信息交互，最终生成一个既能反映共性规律又能体现个性差异的复合相似度矩阵。这种融合方法避免了简单加权平均的信息损失，更好地保持了原始数据的结构特性。从融合相似度出发，通过三重分析路径提取美学规律：统计分析识别高一致性维度作为普遍法则；聚类分析揭示园林群体的自然分组模式；层次结构分析评估各特征维度的稳定性。最终归纳出江南园林” 有法无式” 的具体表现形式，为园林保护、修复和创新设计提供科学依据。

6.2 相似性度量

我们构建了包含六种相似度类型的多模态建模框架：

(1) 向量相似度：基于园林级别的数值特征向量的余弦相似度

(2) 集合相似度：基于景点类型等离散特征的杰卡德相似度

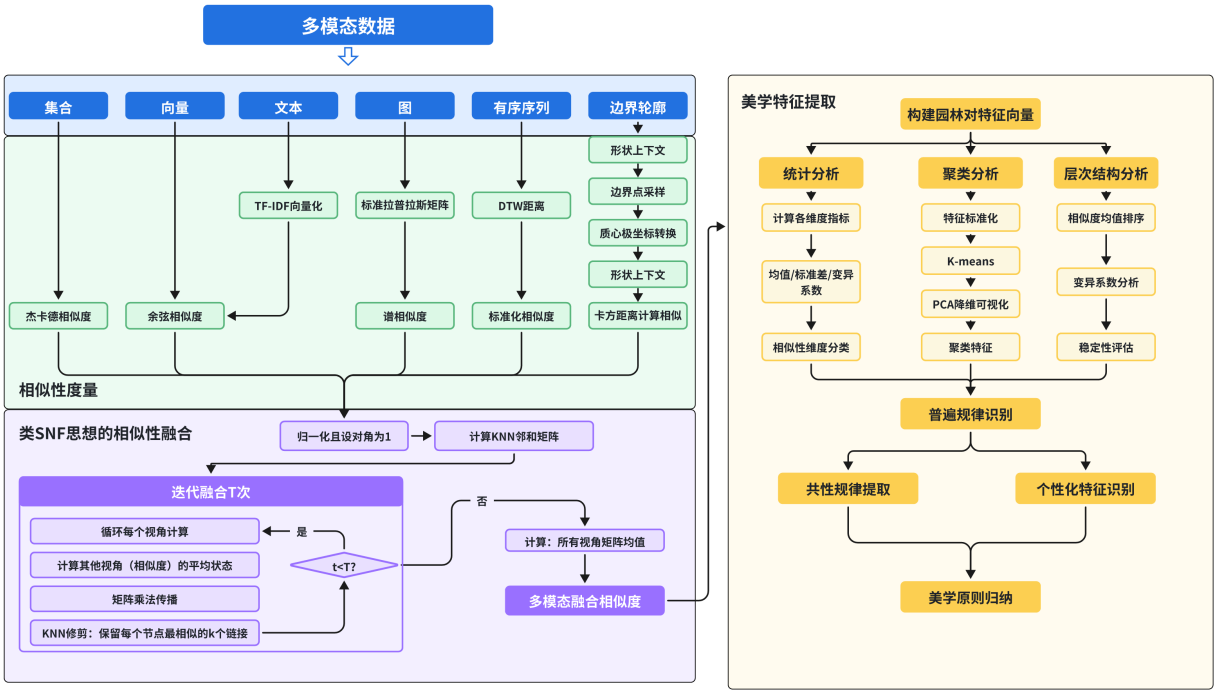

图 6.1　相似度建模与美学共性归纳框架图

(3) 谱相似度：基于图结构拉普拉斯矩阵特征值的相似度

(4) 文本相似度：基于园林名称和景点名称的 TF-IDF 余弦相似度

(5) 序列相似度：基于最佳趣味性路径视域变化序列的 DTW 相似度

(6) 形状相似度：基于边界数据形状上下文的相似性 [12]

这六种相似度从不同角度刻画园林特征：向量相似度反映整体量化特征，文本相似度体现语义内涵，谱相似度揭示结构特性，序列相似度捕捉空间体验变化，形状相似度描述几何特征，集合相似度表征离散属性组合。所有相似度的结果在

6.2.1 多模态相似度计算

1. **向量相似度**:

园林 i 和 j 之间的向量相似度定义为余弦相似度：

$$S_{vec}(i,j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{|\mathbf{v}_i||\mathbf{v}_j|} = \frac{\sum_{k=1}^{d} v_{i,k} \cdot v_{j,k}}{\sqrt{\sum_{k=1}^{d} v_{i,k}^2} \cdot \sqrt{\sum_{k=1}^{d} v_{j,k}^2}}$$

为了处理不同量纲的特征，在计算相似度前需要对特征向量进行标准化：

$$\tilde{v}_{i,k} = \frac{v_{i,k} - \mu_k}{\sigma_k}$$

其中 μ_k 和 σ_k 分别为第 k 维特征在所有园林中的均值和标准差。

特征说明：对图6.2的向量相似度热力图进行分析，可以发现一个显著现象：所有园林间的向量相似度均值高达 0.99，且标准差极低。

这一结果表明尽管江南古典园林在平面布局、规模大小和具体景致上千差万别，呈现出“无式”的表象，但它们在底层的设计逻辑和美学追求上遵循着高度一致的法则，即共同的设计范式。这种内在的一致性，正是对“有法无式”中“有法”的有力数据证明。

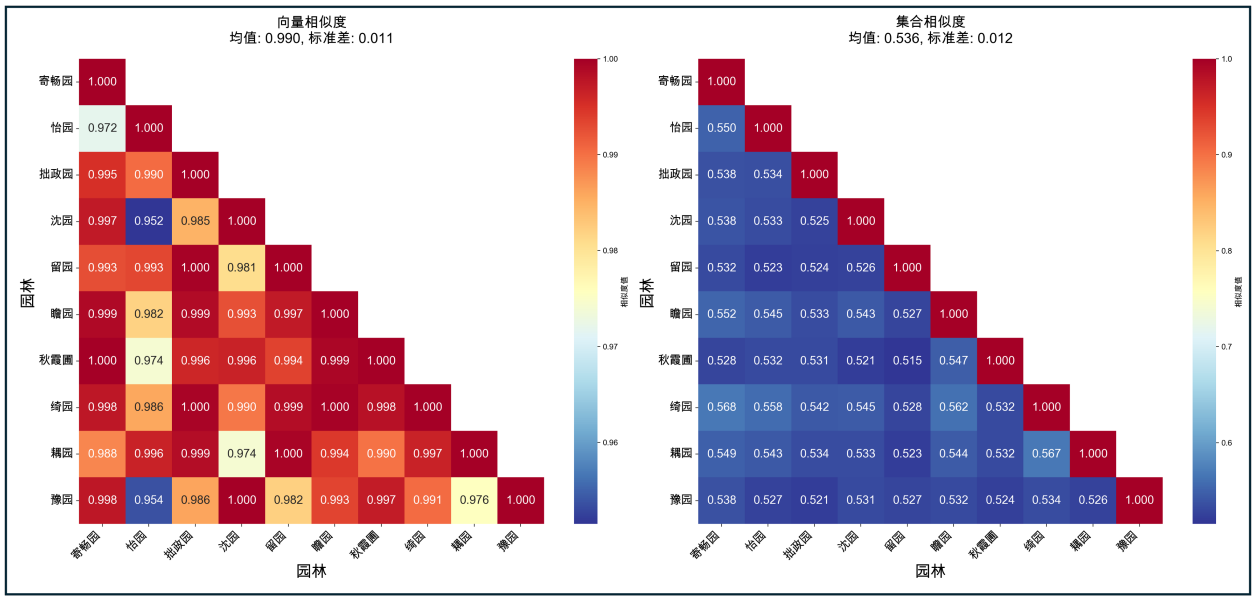

图 6.2　向量相似度与集合相似度热力图

2. **集合相似度建模:**

集合特征构建 园林的集合特征包括两类：

视域可见对象类型集合：

基于视域分析结果，构建园林 i 的可见对象类型集合：

$$\mathcal{C}_i = \{c \mid \exists \text{采样点} p \in P_i, \text{可见对象数}(p, c) > 0\}$$

其中 $c \in \{\text{walls, plants, rocks, roads, water, others}\}$ 为对象类型。名称字符二元组集合：

基于景点名称，构建字符二元组集合：

$$\mathcal{N}_i = \bigcup_{n \in \text{景点名}_i} \text{bigrams}(n)$$

其中 $\text{bigrams}(n)$ 表示从名称 n 中提取的所有连续字符对。

杰卡德相似度 对于两个集合 $\mathcal{A}$ 和 $\mathcal{B}$，杰卡德相似度定义为：

$$\text{Jaccard}(\mathcal{A}, \mathcal{B}) = \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}$$

综合集合相似度 综合两类集合特征的相似度：

$$S_{set}(i,j) = \frac{1}{2}[\text{Jaccard}(\mathcal{C}_i, \mathcal{C}_j) + \text{Jaccard}(\mathcal{N}_i, \mathcal{N}_j)]$$

特征说明：对图6.2的集合相似度进行分析可知，各园林间的相似度处于中等水平，均值为 0.536。该指标综合了两个方面：一是园林中可见景观元素的种类丰富度（通过视域分析），二是景点命名的文化基因（通过名称字符二元组）。

这个中等数值揭示了两个层面的信息。首先，它表明江南园林在造园元素的构成上具有高度的共性。几乎所有园林都包含了建筑、山石、水体、植物这几大核心要素，导致其可见对象类型集合的交集较大。其次相似度中等也这主要源于景点命名体系的差异性。如前文文本相似度分析所述，每个园林的景点命名都承载着独特的文化意涵，其用词和构词方式各不相同，导致名称字符集的交集有限。

谱相似度建模: 本文选用谱相似度对园林间的图结构相似度进行建模。

领接矩阵定义 园林的空间连接关系可建模为无向图 $G_i = (V_i, E_i)$，其中 V_i 为节点集（包括入口和景点），E_i 为边集（表示路径连接）。图的邻接矩阵 A_i 定义为：

$$A_i[u,v] = \begin{cases} w_{uv}, & \text{if } (u,v) \in E_i \\ 0, & \text{otherwise} \end{cases}$$

其中 w_{uv} 为边权重，可以是距离的倒数或其他相关度量。

拉普拉斯矩阵 图的拉普拉斯矩阵定义为：

$$L_i = D_i - A_i$$

其中 D_i 为度矩阵，$D_i[u,u] = \sum_v A_i[u,v]$。

标准化拉普拉斯矩阵为：

$$\mathcal{L}_i = D_i^{-1/2} L_i D_i^{-1/2}$$

标准化拉普拉斯矩阵具有更好的数值稳定性和谱性质。

谱特征提取 计算标准化拉普拉斯矩阵 $\mathcal{L}_i$ 的特征值谱 $\{\lambda_{i,1}, \lambda_{i,2}, \ldots, \lambda_{i,k}\}$（按升序排列）。由于图的连通性质，最小特征值 $\lambda_{i,1} = 0$，因此选取前 k 个特征值作为谱特征向量：

$$\boldsymbol{\lambda}_i = [\lambda_{i,1}, \lambda_{i,2}, \ldots, \lambda_{i,k}]^T$$

对于不连通的图，分别处理每个连通分量，或选择最大连通分量进行分析。

谱相似度计算 谱相似度通过特征值向量的相似性度量。采用均方误差的倒数作为相似度指标：

$$S_{spec}(i,j) = \frac{1}{1 + \text{MSE}(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j)}$$

其中：

$$\text{MSE}(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j) = \frac{1}{k}\sum_{l=1}^{k}(\lambda_{i,l} - \lambda_{j,l})^2$$

特征说明：对图6.3的谱相似度进行分析可知，各个园林之间谱相似度较高，均值为0.904。谱相似度衡量的是园林路径网络的拓扑结构特征。这一高相似度结果表明，尽管各园林布局各异，但其景点与路径构成的网络在宏观结构上遵循着相似的组织范式。

然而，一个显著的例外是怡园，其与其他所有园林的谱相似度均明显偏低。这揭示了怡园在布局上的独特性。其路径网络可能比其他园林更为紧凑、交错，拥有更高的节点密度和更复杂的环路结构。这种独特的、非典型的网络拓扑导致其拉普拉斯谱与其他园林产生显著差异。

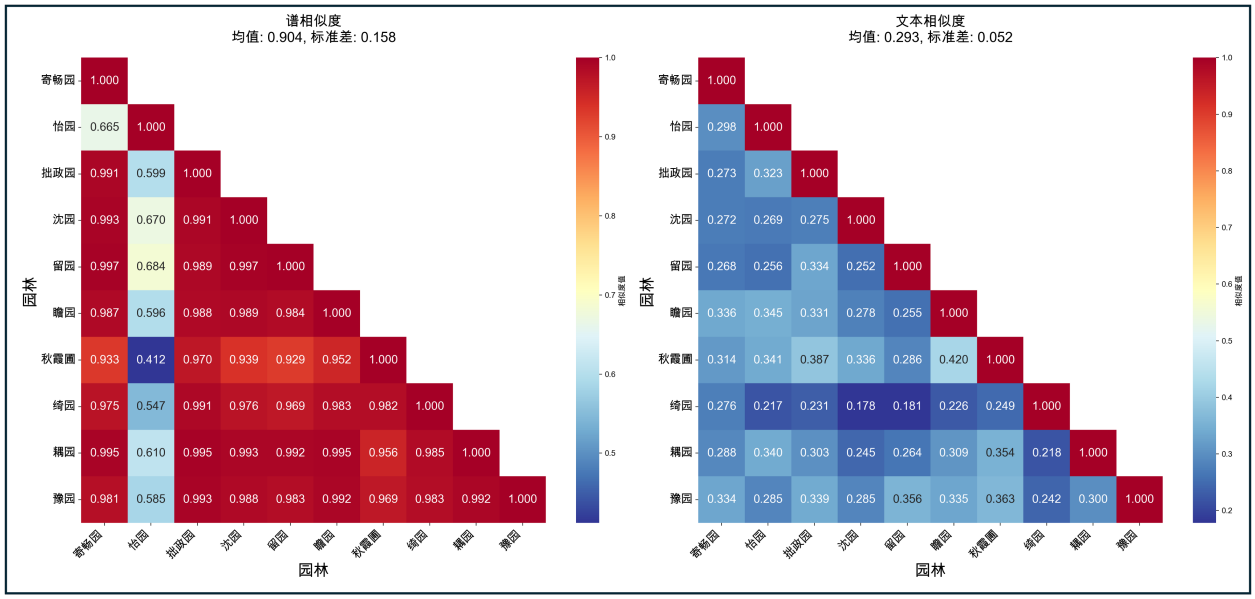

图 6.3　谱相似度与文本相似度热力图

文本相似度：

文本特征构建 园林的文本特征由园林名称和景点名称构成。设园林 i 的文本文档为 D_i：

$$D_i = \{\text{园林名}_i\} \cup \{\text{景点名}_{i,1}, \text{景点名}_{i,2}, \ldots, \text{景点名}_{i,n_i}\}$$

考虑到中文文本的特点，采用字符级 n-gram 方法进行特征提取。对于文档 D_i，构建字符级 bi-gram 和 uni-gram 特征。

TF-IDF 向量化 采用 TF-IDF（Term Frequency-Inverse Document Frequency）方法将文档转换为向量表示。对于词汇表 $\mathcal{V}$ 中的词项 t，其在文档 D_i 中的 TF-IDF 值为：

$$\text{TF-IDF}(t, D_i) = \text{TF}(t, D_i) \times \text{IDF}(t)$$

其中：

$$\text{TF}(t, D_i) = \frac{\text{count}(t, D_i)}{\sum_{t' \in D_i} \text{count}(t', D_i)}$$

$$\text{IDF}(t) = \log \frac{N}{\text{DF}(t)}$$

这里 N 为总文档数，$\text{DF}(t)$ 为包含词项 t 的文档数。

文本相似度计算 文本相似度通过 TF-IDF 向量的余弦相似度计算：

$$S_{text}(i, j) = \cos(\text{TF-IDF}_i, \text{TF-IDF}_j) = \frac{\text{TF-IDF}_i \cdot \text{TF-IDF}_j}{|\text{TF-IDF}_i||\text{TF-IDF}_j|}$$

特征说明：与向量相似度形成鲜明对比，图6.3显示，基于园林名和景点名的文本相似度普遍较低，均值仅为 0.293。这一指标衡量的是园林在文化符号与意境表达上的独特性。低相似度首先说明了每个园林在命名体系上的高度个性化与创造性。其次，这也反映了“移步异景”在文化层面的体现，每个景点名称都力求独特，共同构建一个丰富多元、不可复制的意境序列。

序列相似度建模:

视域变化序列构建

基于最佳趣味性路径构建视域变化序列。设园林 i 的最佳路径为节点序列 $(n_1, n_2, \ldots, n_{m_i})$，对应的边序列为 $(e_1, e_2, \ldots, e_{m_i-1})$，其中 $e_l = (n_l, n_{l+1})$。

通过视域分析数据，获取每条边上采样点的视域面积。设边 e_l 上的采样点视域面积为 $\{a_{l,1}, a_{l,2}, \ldots, a_{l,p_l}\}$，则该边的代表视域面积为：

$$x_l = \frac{1}{p_l} \sum_{k=1}^{p_l} a_{l,k}$$

由此构建园林 i 的视域变化序列：

$$\mathbf{x}_i = (x_1, x_2, \ldots, x_{m_i-1})$$

动态时间规整（DTW）算法

不同园林的路径长度可能不同，因此采用动态时间规整（DTW）算法计算序列距离。对于序列 $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,m})$ 和 $\mathbf{x}_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,n})$，DTW 距离的递推关系为：

$$D(p,q) = |x_{i,p} - x_{j,q}| + \min \begin{cases} D(p-1,q-1) & \text{（对角线）} \\ D(p-1,q) & \text{（垂直）} \\ D(p,q-1) & \text{（水平）} \end{cases}$$

初始条件：

$$D(0,0) = 0, \quad D(p,0) = D(0,q) = \infty \quad \text{for } p,q > 0$$

最终的 DTW 距离为 $D(m,n)$。

序列相似度计算

为了将距离转换为相似度，并考虑不同园林序列幅度的差异，采用标准化的相似度计算：

$$S_{seq}(i,j) = \frac{1}{1 + \frac{D(m_i,m_j)}{(\bar{x}_i + \bar{x}_j)/2}}$$

其中 $\bar{x}_i = \frac{1}{m_i}\sum_{l=1}^{m_i} x_{i,l}$ 为序列 $\mathbf{x}_i$ 的均值，用于标准化距离量纲。

特征说明：对图6.4的序列相似度进行分析可知，各园林间的相似度极低，均值仅为0.101。该指标通过 DTW 算法，比较了沿最佳游线行进时“视域面积”变化序列的相似性，它本质上量化了园林空间“开合变化”的节奏与韵律。

这一极低的相似度表明：虽然园林在设计准则（向量相似度）和网络拓扑（谱相似度）上可能相似，但如何引导游客穿梭于开阔与围合之间，营造出富有节奏感和叙事性的游览体验上则完全不同。这正是“移步异景”在更高维度层面的量化体现。

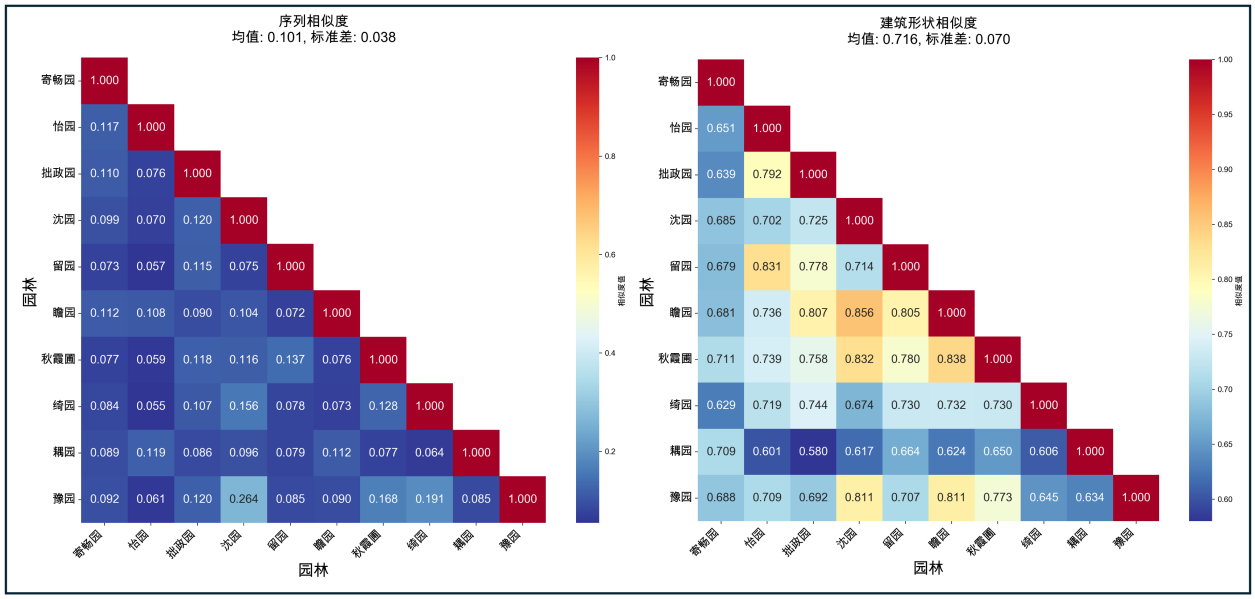

图 6.4　序列相似度与建筑形状相似度热力图

形状相似度建模:

边界数据处理 园林的形状特征通过边界数据体现。根据园林要素的不同，分别处理四类边界：

- 建筑边界：实体建筑 + 半开放建筑
- 水体边界：水体要素边界点
- 植物边界：植物要素边界点
- 假山边界：假山要素边界点

每类边界由一组二维点 $\mathcal{P}_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{n_i}, y_{n_i})\}$ 表示。

形状上下文（Shape Context） 形状上下文是一种有效的形状描述子。对于园林 i 的边界点集 $\mathcal{P}_i$，首先计算其质心：

$$(\bar{x}_i, \bar{y}_i) = \left(\frac{1}{n_i} \sum_{k=1}^{n_i} x_k, \frac{1}{n_i} \sum_{k=1}^{n_i} y_k \right)$$

将每个边界点转换为相对于质心的极坐标 (r, θ)：

$$r_k = \sqrt{(x_k - \bar{x}_i)^2 + (y_k - \bar{y}_i)^2}$$

$$\theta_k = \arctan 2(y_k - \bar{y}_i, x_k - \bar{x}_i)$$

形状上下文直方图 构建二维直方图 $H_i \in \mathbb{R}^{K \times L}$，其中 K 为角度分箱数，L 为对数径向分箱数：

$$H_i(k, l) = \sum_{p=1}^{n_i} \mathbf{1}_{[\theta_p \in \text{bin}_k] \cap [\log r_p \in \text{bin}_l]}$$

其中 $\mathbf{1}[\cdot]$ 为示性函数。角度均匀分箱：

$$\text{bin}_k = \left[\frac{2\pi(k-1)}{K}, \frac{2\pi k}{K} \right), \quad k = 1, 2, \ldots, K$$

对数径向分箱：

$$\log r_p^{norm} = \frac{\log r_p - \log \bar{r}_i}{\sigma_{\log r}}$$

其中 $\bar{r}_i$ 为平均半径，$\sigma_{\log r}$ 为对数半径的标准差。

标准化直方图：

$$\tilde{H}_i(k, l) = \frac{H_i(k, l)}{\sum_{k'=1}^{K} \sum_{l'=1}^{L} H_i(k', l')}$$

形状相似度计算 采用卡方距离度量直方图之间的差异：

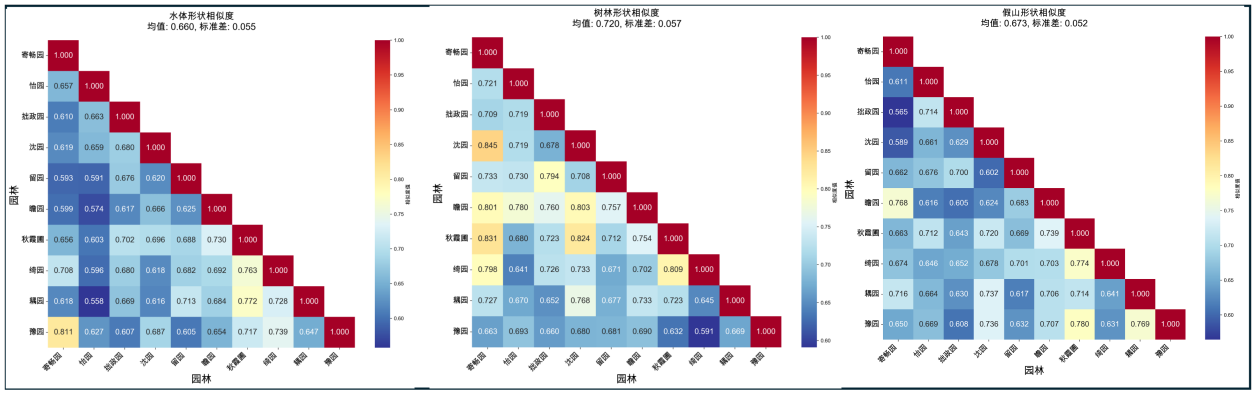

图 6.5　水体树林假山热力图

$$\chi^2(\tilde{H}_i, \tilde{H}_j) = \frac{1}{2}\sum_{k=1}^{K}\sum_{l=1}^{L}\frac{(\tilde{H}_i(k,l) - \tilde{H}_j(k,l))^2}{\tilde{H}_i(k,l) + \tilde{H}_j(k,l) + \epsilon}$$

其中 ϵ 为小正数，防止分母为零。

形状相似度通过卡方距离的倒数计算：

$$S_{shape}(i,j) = \frac{1}{1 + \chi^2(\tilde{H}_i, \tilde{H}_j)}$$

其中建筑类型边界的相似度热力图可视化在图6.4的右侧，另外水体、树林、假山类型边界的相似度热力图可视化在图6.5。

特征说明：对图6.4的建筑形状相似度进行分析可知，各园林间的相似度处于中等水平，均值为 0.718。该指标量化了园林中建筑群落的宏观空间分布形态。

这个中等水平的相似度恰如其分地反映了江南园林建筑布局的特点：既有共性，又有个性。一方面，较高的均值说明建筑布局存在一定的范式。另一方面，相似度中等也表明每个园林在具体的建筑组合、朝向和密度上又各有侧重，形成了自己独特的风格。

6.2.2 相似性结果分析

6.2.3 多模态相似性融合

本文参考相似性网络融合（Similarity Network Fusion, SNF）方法将多种相似度整合为复合相似度，流程可见图6.1的类 SNF 思想的相似性融合部分。首先，定义 $\mathbf{S}^{(v)} \in \mathbb{R}^{n\times n}$ 为第 v 种相似度矩阵，其中 $v \in \{\text{vec}, \text{text}, \text{spec}, \text{seq}, \text{shape}, \text{set}\}$，$n$ 为园林数量。

k 近邻亲和矩阵构建 首先构建 k 近邻亲和矩阵。对于相似度矩阵 $\mathbf{S}^{(v)}$，保留每行的前 k 个最大值，其余置零：

$$W_{ij}^{(v)} = \begin{cases} S_{ij}^{(v)}, & \text{if } j \in \text{TopK}(S_{i,:}^{(v)}) \\ 0, & \text{otherwise} \end{cases}$$

对称化处理：

$$\tilde{W}^{(v)} = \frac{W^{(v)} + (W^{(v)})^T}{2}$$

行归一化得到转移矩阵：

$$P_{ij}^{(v)} = \frac{\tilde{W}_{ij}^{(v)}}{\sum_{k=1}^{n} \tilde{W}_{ik}^{(v)}}$$

迭代网络融合 SNF 算法的核心是迭代更新过程。在第 t 次迭代中，第 v 种视角的亲和矩阵更新为：

$$W_{t+1}^{(v)} = P^{(v)} \cdot \left(\frac{1}{m-1} \sum_{u \neq v} W_t^{(u)} \right) \cdot (P^{(v)})^T$$

其中 m 为视角总数。该更新规则的含义是：每个视角的亲和矩阵通过其他视角的平均信息进行更新，同时保持自身的局部结构（通过 $P^{(v)}$）。

收敛与融合 重复迭代直至收敛或达到最大迭代次数 T。收敛后的复合相似度矩阵为：

$$\mathbf{S}_{composite} = \frac{1}{m} \sum_{v=1}^{m} W_{\infty}^{(v)}$$

最终进行归一化处理，确保对角线元素为 1：

$$\tilde{S}_{composite}[i,j] = \frac{S_{composite}[i,j] - \min_{k,l} S_{composite}[k,l]}{\max_{k,l} S_{composite}[k,l] - \min_{k,l} S_{composite}[k,l]}$$

$$\tilde{S}_{composite}[i,i] = 1, \quad \forall i$$

融合后的相似度矩阵的热力图可视化在图6.6中。

6.3 美学特征提取模型

6.3.1 基于统计分析的美学规律发现

对每个相似度维度计算：
均值：$\mu = \frac{2}{n(n-1)} \sum_{i<j} S_{ij}$
标准差：$\sigma = \sqrt{\frac{2}{n(n-1)} \sum_{i<j} (S_{ij} - \mu)^2}$
变异系数：$CV = \sigma / \mu$

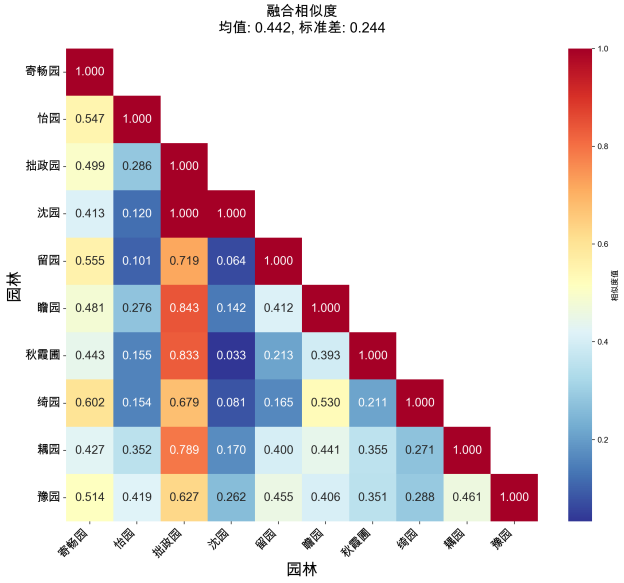

图 6.6　SNF 算法融合相似度矩阵热力图

表 6.1　不同特征相似度指标的计算结果

相似度类型	平均相似度	变异系数
余弦相似度	**0.99**	0.01
谱相似度	**0.90**	0.18
树林边界相似度	0.72	0.08
建筑边界相似度	0.72	0.10
假山边界相似度	0.67	0.08
水体边界相似度	0.66	0.08
集合相似度	0.54	0.02
SNF 融合相似度	0.44	0.49
文本相似度	**0.29**	0.18
序列相似度	**0.10**	0.38

数据结果： 表6.1展示了统计分析的具体数据结果。

将均值大于 0.8 的归类为高度一致性特征，均值小于 0.3 的归类为个性化表达特征。

6.3.2 基于聚类分析的美学群体识别

对园林对 (i, j) 构建 6 维特征向量：$\vec{f}ij = [sij^{vector}, s_{ij}^{text}, s_{ij}^{spectral}, s_{ij}^{sequence}, s_{ij}^{shape}, s_{ij}^{set}]$ 之后使用 Z-score 归一化进行特征标准化，之后使用 K-means 进行聚类，PCA 降维之后可视化在图6.7中。

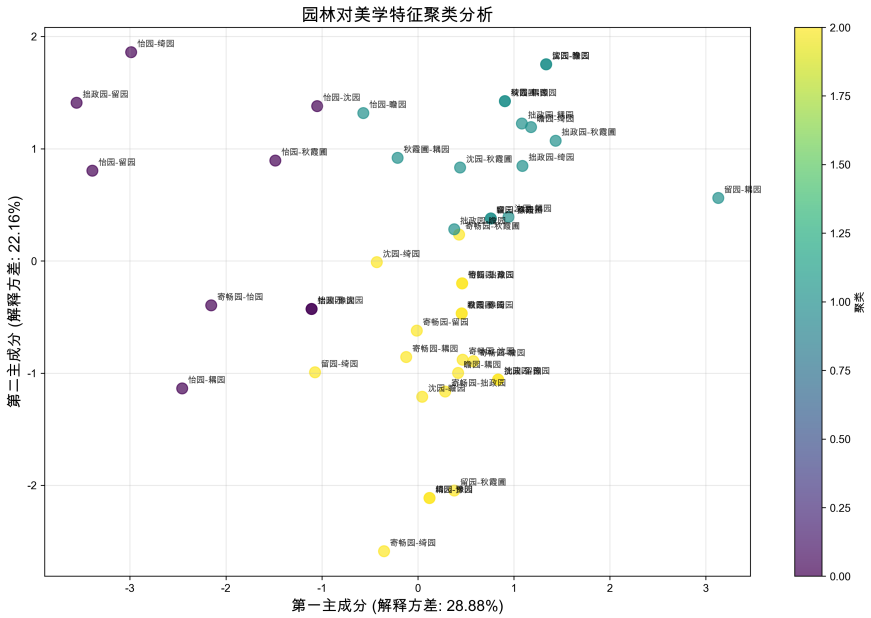

图 6.7　美学特征聚类结果

6.3.3 基于层次结构的美学稳定性评估

层次构建方法: 先按相似度均值降序排列各维度，之后计算每个维度的变异系数 (CV)。

根据定义，CV 值越小，特征越稳定。从表6.1中可以发现，基本上所有指标的 CV 值都很小，大部分特征都高度稳定。

6.3.4 美学特征归纳

多模态相似度计算结果分析: 通过对 10 个代表性江南古典园林的全面分析，本研究获得了丰富的量化数据支持。分析结果显示，各相似度维度呈现出明显的层次结构特征，为理解” 有法无式” 的美学特质提供了实证基础。

高度一致性特征表明园林在空间布局和拓扑结构上存在显著共性。向量相似度均值达到 0.990（CV=0.011），谱相似度均值为 0.904（CV=0.175），这说明江南古典园林在整体空间组织和网络拓扑方面遵循着高度一致的设计法则，体现了” 有法” 的深刻内涵。这种一致性反映了中国传统造园技艺的规范性和传承性。

中等相似度特征主要集中在几何形态和元素配置维度。形状相似度均值为 0.716（CV=0.070），集合相似度均值为 0.536（CV=0.023），表明园林在具体形态设计和元素选择上既存在一定的共性基础，又保留了适度的灵活性。这种中等程度的相似性恰恰体现了” 有法无式” 中法则与自由的平衡。

低相似度特征突出表现在文本语义和游览序列方面。文本相似度均值仅为 0.293（CV=0.176），序列相似度均值低至 0.101（CV=0.376），这充分说明各个园林在文化表达和游览体验设计上具有强烈的个性化特征，是” 无式” 的具体体现。

聚类分析揭示的美学群体特征基于多维相似度特征的聚类分析将 45 个园林对划分为三个具有明显差异的美学群体，这一发现具有重要的理论价值。

聚类 0（9 个园林对）以中等谱相似度（0.588）和形状相似度（0.694）为特征，代表了一种相对均衡的美学表达方式。这一群体的园林在保持基本设计法则的同时，在空间组织和形态设计上表现出适度的个性化。

聚类 1（16 个园林对）显示出极高的谱相似度（0.954）和较高的形状相似度（0.793），表明这一群体在空间拓扑结构和几何形态上具有高度一致性。这些园林可能代表了江南古典园林中较为传统和规范的设计流派。

聚类 2（20 个园林对）虽然谱相似度最高（0.985），但形状相似度相对较低（0.685），文本相似度也仅为 0.265。这种特征组合表明该群体园林在保持基本空间结构规范的同时，在形态设计和文化表达上具有较大的创新空间。

三个聚类群体的分布验证了江南古典园林并非单一模式，而是在共同法则指导下形成了多样化的美学表达方式，这正是" 有法无式" 精髓的量化体现。

6.4 广效用验证：网师园

在以上对江南园林的“趣味性”、“幻境感”、“相似度”的建模中，广效用具有重要的意义，它能够确保模型不仅仅在特定的园林样本上有效，而是在不同类型和风格的园林中也能保持准确性和可靠性。中国古典园林在设计上具有地域性和时代性的多样性，因此验证方法的广效用有助于测试模型是否具备跨区域、跨时代的适用能力，能够适应不同园林的空间布局、文化元素以及设计需求。

“苏州最精致、最完整的私家园林”：鉴赏家玛丽安娜 · 鲍榭蒂女士在其《中国园林》一书中给予网师园此赞誉。网师园始建于南宋，虽历史悠久，但现存的规模、景物建筑多为清乾隆年间宋宗元重建以及乾隆末年瞿远村修复增建后的遗物，其布局、建筑风格等体现了明清时期园林的特点，因此本文选择网师园作为广效用的验证对象。

6.4.1 问题一模型验证结果

将网师园的数据输入问题一的模型后，首先得到了网师园的整体布局图（见图6.8）。该图展示了园林的整体结构和路径分布，是后续路径网络生成的基础。通过应用 Prim 算法结合 A* 算法，生成了网师园的稀疏游线网络图，并通过密集化处理得到了更加精细的稠密游线网络图（见图6.9）。图6.10展示了游线网络图的结构，揭示了园林内部路径之间的连接关系和节点布局，这为后续的路径优化提供了重要的结构数据。

最终，模型计算得出了网师园的最优游线路径，结果如图6.11所示。这个路径满足了园林的趣味性要求，同时减少了重复路径，体现了路径的多样性。然而，路径网络的构建并未覆盖园林内所有路径，导致某些“冤枉路”的产生。为进一步优化路径生成，未来可以考虑增加路径网络的覆盖范围，从而让模型生成的路径更加合理。

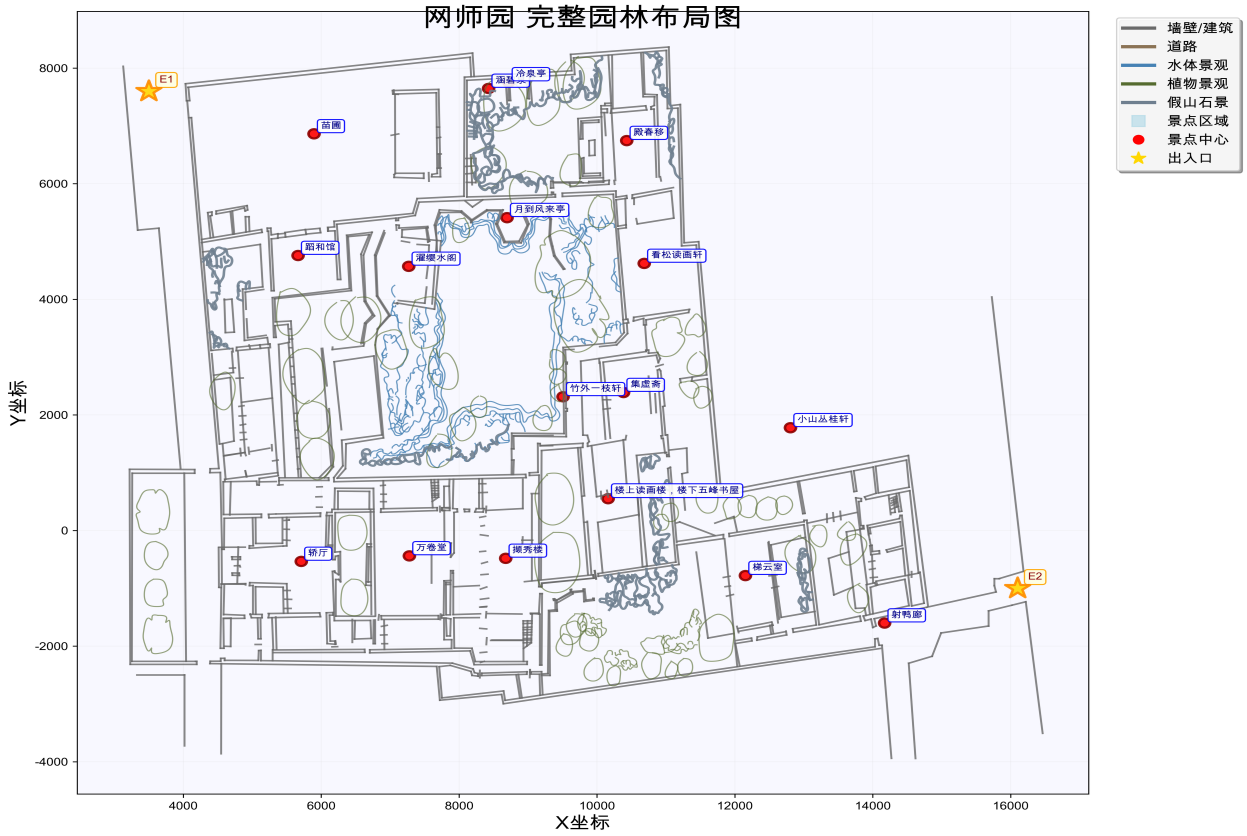

图 6.8　网师园整体布局图

6.4.2 问题二模型验证结果

在第二问的模型验证中，使用第一问得到的游线路径文件和原始 DWG 文件数据进行处理，得出了网师园的开阔度评价图（见图6.12），展示了园林空间开阔区域的分布情况。开阔度分析对评估园林空间感和游客的游园体验有着重要意义。图6.13展示了网师园在模型评价中的最终得分和排名，反映了其在多个维度下的表现。

通过对数据的分析，我们发现，模型对于大尺度、多层次园林的评价效果较好，能够较为准确地刻画园林的主题和幻境感。然而，网师园作为一座较小的园林，具有许多围合性质的空间，因此在某些绝对评价指标上处于劣势。模型对网师园的评分较低，显示出当前模型在精巧型园林评价中的局限性。因此，未来的研究需要考虑如何对不同类型的园林，尤其是精巧型园林，进行更加合理的评价。

6.5 多元化探索：其他美学特征的讨论

在江南古典园林的美学特征研究中，除了本文已建模的” 趣味性”” 幻境感” 和” 相似度” 外，还存在其他重要的美学特征，这些特征同样源于中国古典园林的造园理念和文化内涵。基于园林的多模态数据（如平面图、矢量图、景观元素坐标），我们进一步探讨了” 自然度”” 文化内涵” 和” 空间节奏” 这三个美学特征，并提供了量化思路和依据。这些特征

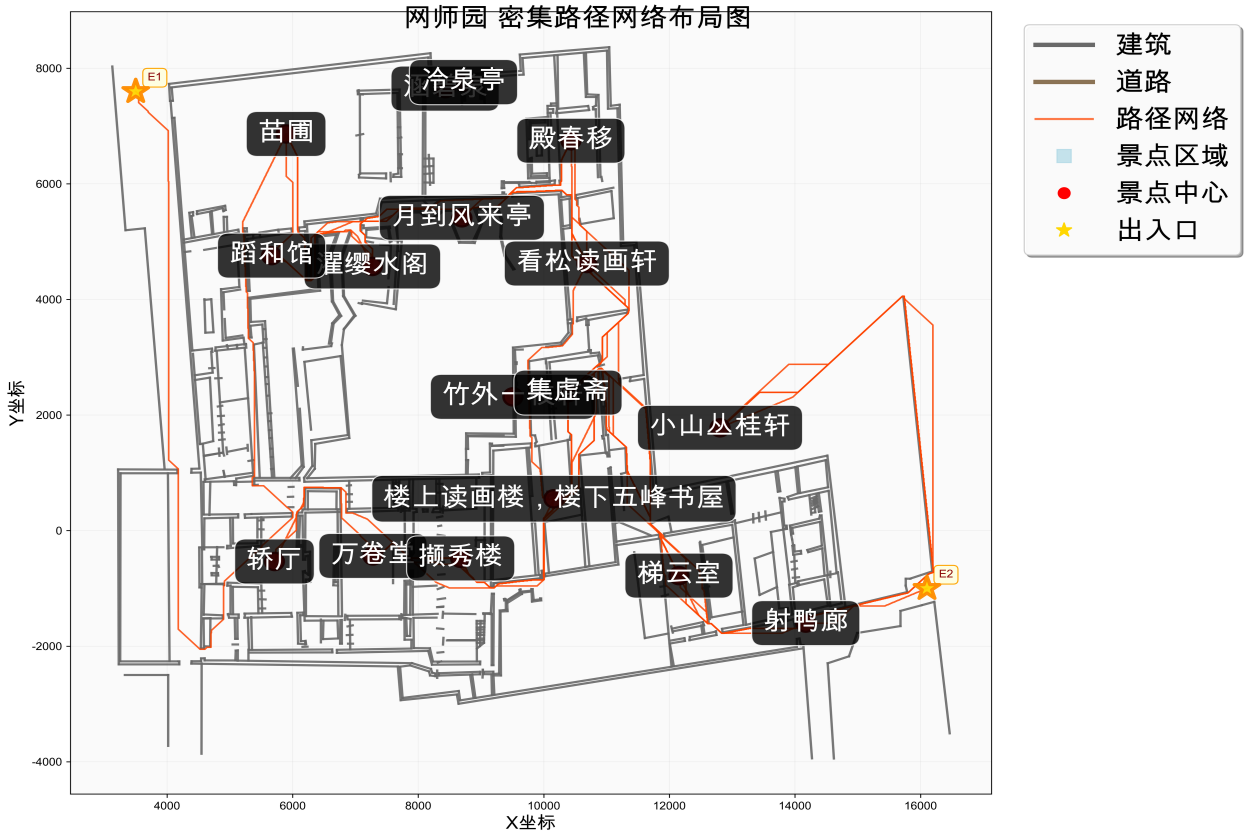

图 6.9　网师园稠密游线网络图

不仅丰富了园林美学的刻画维度，也为数字化传承提供了新视角。

6.5.1 自然度

重要性依据：江南古典园林追求”虽由人作，宛自天开”的艺术境界，强调人工景观与自然生成景观的相似性。自然度反映了园林模仿自然山水形态、结构和演变规律的能力，是评价园林美学价值的关键指标。如计成在《园冶》中强调”有真为假，做假成真”，即通过人工手段再现自然景观的本质特征。

量化思路：自然度可通过园林景观与自然生成景观在形态、结构和动态特征上的相似性来量化。具体方法包括：

- **地形形态相似性**：利用数字高程模型（DEM）分析技术，比较园林地形与自然山水的形态特征：
  - **曲率分析**：计算地表曲率，自然地形通常具有连续变化的曲率分布。采用高斯曲率和平均曲率量化地形复杂性：

$$K = \frac{\partial^2 z}{\partial x^2} \cdot \frac{\partial^2 z}{\partial y^2} - \left(\frac{\partial^2 z}{\partial x \partial y} \right)^2$$

  - **分形维数**：采用盒计数法计算地形边界的分形维数，自然地形通常具有较高的分形

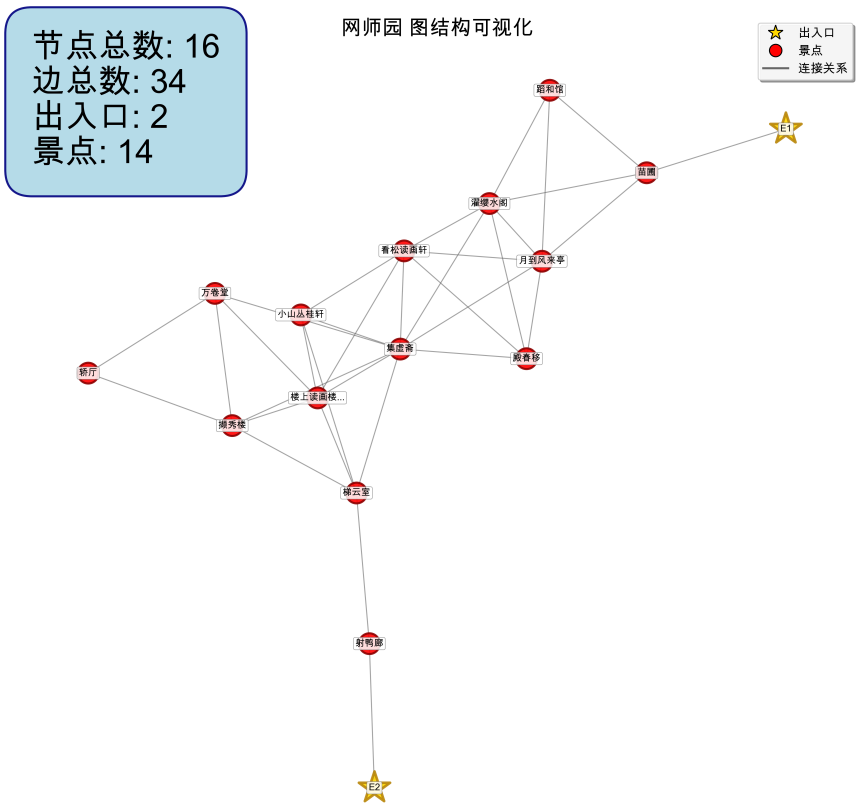

图 6.10　网师园游线网络图结构

维数（1.2-1.5）：

$$D = \lim_{\varepsilon \to 0} \frac{\log N(\varepsilon)}{\log(1/\varepsilon)}$$

  - **水文网络相似性**：模拟自然流域的水文网络，比较园林水系的拓扑结构（如斯特拉勒序、分支比）与自然河流的相似度。

- **生态结构匹配度**：基于景观生态学原理，评估园林植物配置与自然群落的相似性：
  - **种间关系**：利用植物坐标数据，计算最近邻指数（NNI）评估植物分布模式（随机、聚集或规则）：

$$R = \frac{\bar{r}_{\text{obs}}}{\bar{r}_{\text{exp}}}$$

  - **层次结构**：分析植物群落垂直分层（乔木层、灌木层、草本层）与自然森林群落的匹配程度。
  - **年龄结构**：通过树冠大小分布推断年龄结构，与自然群落的年龄分布进行比较。
- **动态演化特征**：模拟景观的自然演化过程，评估园林设计的” 自然感”：
  - **侵蚀模拟**：使用元胞自动机模拟水蚀过程，比较园林地形稳定性与自然地形；
  - **群落演替**：建立植物群落演替模型，评估当前配置与潜在顶极群落的相似性；

图 6.11　网师园第一问最优游线路径结果

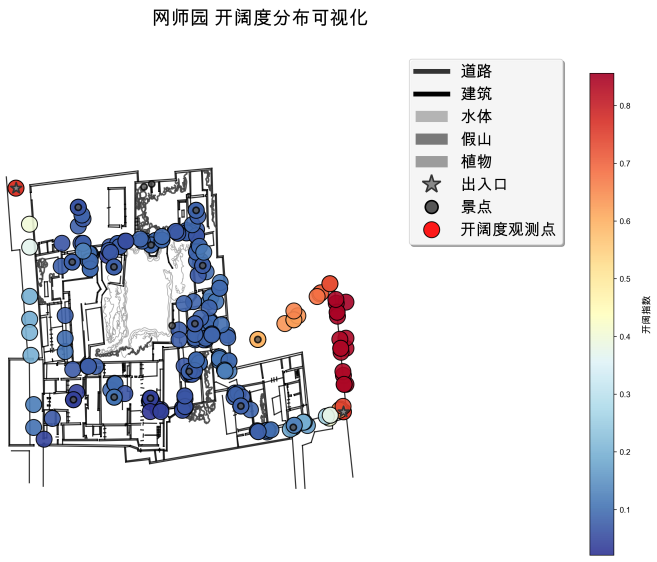

图 6.12　网师园开阔度分析

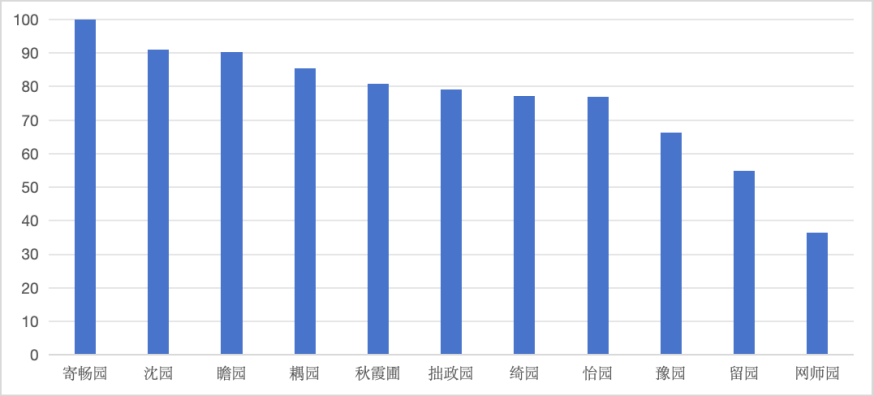

图 6.13　网师园第二问模型评价结果

  - **随机性检验**：使用 L 系统（Lindenmayer System）生成自然景观模式，与园林布局进行模式匹配。

- **视觉感知自然性**：基于视觉感知理论，量化景观的自然印象：
  - **纹理分析**：使用 Gabor 滤波器提取景观图像的纹理特征，与自然景观纹理库进行相似性匹配；
  - **色彩协调性**：分析景观色彩分布的熵值与自然景观的相似度；
  - **视觉复杂性**：基于视觉搜索理论，评估场景的视觉信息量与自然景观的匹配程度。

  综合自然度评分模型：

$$N = w_1 \cdot S_{\text{terrain}} + w_2 \cdot S_{\text{ecology}} + w_3 \cdot S_{\text{dynamics}} + w_4 \cdot S_{\text{visual}}$$

其中，S_{terrain} 为地形相似度，S_{ecology} 为生态结构匹配度，S_{dynamics} 为动态演化相似度，S_{visual} 为视觉自然度，权重 w_i 通过专家调查法确定。

6.5.2 文化内涵

重要性依据：江南园林是文人雅士的精神寄托，常融入诗词、绘画、哲学思想，如陆游与沈园的关联、拙政园的” 远香堂” 取自周敦颐《爱莲说》。文化内涵提升了园林的艺术品位，是美学特征的核心。董豫赣在《江南园林志》中指出，园林是” 立体诗画”，文化元素赋予景观深意。

量化思路：文化内涵的量化需结合文本和空间数据，方法包括：

- **文本分析**：从园林简介、历史文献或景点名称中提取文化关键词（如诗词典故、哲学概念）。使用自然语言处理技术，计算文本的文化密度。例如，TF-IDF（Term Frequency-Inverse Document Frequency）加权后，构建文化词频向量，并计算其模长作为文化内涵得分。
- **元素象征性**：特定景观元素具有文化象征，如梅兰竹菊代表君子品格。根据元素类型赋予文化权重（如竹子权重高于普通树木），加权求和得文化价值。

- **历史事件关联**：园林与历史事件或人物的关联度可通过网络资源量化。如沈园因陆游故事闻名，可基于百度百科或学术数据库检索关联词频。

应用示例：沈园有”钗头凤”碑刻，文化词频高；而绮园以山水为主，文化元素较少。文化内涵评分可补充”相似度”模型，区分园林的人文特质。

6.5.3 总结与讨论

上述个特征——自然度、文化内涵和空间节奏——从不同维度刻画了江南古典园林的美学多样性。自然度体现了园林的生态智慧，文化内涵承载了历史传承，空间节奏控制了游览体验。这些特征均可基于现有数据量化：

- 自然度利用矢量图的空间计算。
- 文化内涵结合文本挖掘和符号学。
- 空间节奏依赖于路径序列分析。

在后续研究中，可将这些特征融入相似度模型，形成多维度美学指标体系。此外，季节变化、声景美学等特征也值得探索，但需更多数据支持。

7 模型评价

针对问题一的模型，当前版本能够生成完整且合理的游园路径，在保证趣味性的同时有效减少了重复路线，表明该模型较好地量化了路径特征与异景程度。网师园的计算结果进一步验证了模型具有较好的鲁棒性。然而，由于所构建的路径网络未能覆盖园林中全部可行走路径，可能导致游客在实际游览中绕行部分“冤枉路”。后续可探索生成覆盖率更高的路径网络方案，以进一步提升生成路径的合理性与实际可行性。

针对问题二的模型，最终输出结果呈现了各园林的排名。分析表明，该模型的评价指标并非简单依赖于主题或景观种类的数量，而是较好地捕捉了题目所要求的 “幻境感”。但在引入网师园数据进行验证时，发现其评分显著偏低。进一步分析显示，网师园作为一座精巧型园林，在空间尺度、围合性等方面与大型园林存在差异，导致其在若干绝对指标上处于劣势。这说明当前模型更适用于评价大尺度、多层次的传统园林，而对小型精巧型园林可能存在系统性偏差，未来需进一步优化评价体系的适用性。

针对问题三的模型，能够有效量化“有法无式”的设计理念，并在多次测试中表现出良好的鲁棒性，说明模型具备一定的理论合理性与实践稳定性。

参考文献

[1] 严佳, 游观视角下的江南园林假山游径平面解析, 东南大学, 2023.

[2] 杨静, 模拟尺度的选择及栅格地图相似性评价方法的应用, 昆明理工大学, 2015.

[3] 杨萍, 园林景观中尺度的合理性分析: 第 009 册.

[4] 孙萌, 苏州园林小空间大氛围设计手法探究及其在现代建筑中的应用, 南京工业大学, 2018.

[5] 叶佳琪, 基于空间句法的苏州古典园林廊空间量化研究, 集美大学, 2024.

[6] 彭江林, 中国古代园林阁廊组合的空间及路径趣味性, 大舞台(04):150, 2012.

[7] 曹珊珊, 基于粒子群和蚁群混合算法的城市园林景观植物群落优化配置方法, 佳木斯大学学报 (自然科学版), 42(11):37-39+95, 2024.

[8] 夏德美, 基于粒度分层的风景园林植物配置及空间布局研究, 陇东学院学报, 36(02): 76-81, 2025.

[9] Zhang Y, Quantitative analysis of surface landscape elements for landscape environmental design - based on high resolution remote sensing image technology, Systems and Soft Computing, 7200325-200325, 2025.

[10] 吴涛, 陶欣, 王晓春, 扬州小盘谷园林造景要素的量化分析研究, 园林(06):57-63, 2020.

[11] 孟兆祯, 中日韩园林的相似性与独特性, 风景园林人居环境小康社会——中国风景园林学会第四次全国会员代表大会论文选集（下册）, 158-160, 2008.

[12] 沈国沅, 李胜, 田茵茵, 杭州传统园林假山皴纹形态量化研究——以黄龙洞、蒋庄假山为例, 园林, 42(08):98-106, 2025.

[13] Martin B, The Machinic Garden of Forking Paths: Time, Tempo and Tango, Architectural Design, 94(6):40-47, 2024.

[14] 张谦, 江南园林中的迷宫趣味性, 中国建筑装饰装修(13):106-108, 2022.

[15] 曹苑亭, 移步异景理念在现代南方园林景观设计中的应用, 住宅与房地产(12):26, 2019.

[16] 朱子墨, 江南私家园林假山路径量化研究, 南京林业大学, 2020.

[17] (DeepSeek) 深度求索, DeepSeek - R1 模型, 深度求索 (DeepSeek) 公司, 2025.

附录 A　AI 辅助说明

AI 辅助说明

在本论文的撰写过程中，我们使用了人工智能（AI）工具以辅助研究。具体信息如下：

1. AI 软件平台:
   名称: DeepSeek
   版本: R1
   制造商: 深度求索
2. AI 使用方式详述: 我们在论文的以下部分借助了 AI 工具：
   模型构建与概念化阶段: 使用 AI 辅助构思，对不同的数学建模方法和思路进行初步探索，并对模型的基本假设和框架进行构思。
   方法论部分: AI 被用于协助生成和优化核心数学方程，验证公式的推导逻辑，并为模型的参数选择提供了参考建议。此外，在编写模型算法的伪代码时，也借助了 AI 的辅助。
   数据分析与模拟部分: 使用 AI 辅助编写了用于模型模拟和数据分析的 Python 代码，并对模拟结果的可视化方案进行了探讨。
   结果与讨论部分: AI 协助我们对模型输出的数据进行初步解读，并对结果讨论部分的语言表达和结构组织进行了润色和优化。
3. 作者责任确认: 本论文的所有作者已对 AI 生成的内容进行了严格的审查、验证和修订。我们确认，手稿的最终内容（包括所有数据、分析、结论和文字表述）的准确性和完整性由全体作者承担全部责任。AI 工具的使用仅限于辅助性质，所有关键的学术判断和最终成果均由作者独立完成。

注：本声明不涵盖使用 AI 工具进行常规的语法、拼写检查或文献格式排版等用途。

附录 B　Python 源程序

main.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
江南古典园林美学特征建模系统 - 核心算法精简版

核心数学模型:
- 多目标优化: F = √(P × V) × U
- 幻境感评分: 四维评价体系
- 相似性度量: 频谱相似性 + 形状相似性

分析11个经典园林: 拙政园、留园、寄畅园、瞻园、豫园、秋霞圃、沈园、怡
    园、耦园、绮园、网师园
"""
import json
import math
from typing import Dict, List, Tuple, Optional
from statistics import mean, pstdev


# A1. DXF文件处理核心算法
class DXFReader:
    """DXF文件读取器"""

    def __init__(self):
        self.layer_keywords = {
            'walls': ['墙', '建筑', 'wall', 'building'],
            'roads': ['路', '道路', 'road', 'path'],
            'water': ['水', 'water', '池', '湖'],
            'plants': ['植', 'plant', 'tree', '树'],
            'rocks': ['山', 'rock', '石', '假山']
        }

    def read_dxf_file(self, file_path: str) -> Optional[Dict]:
        """读取DXF文件并提取几何数据"""
        try:
            # 简化的DXF读取逻辑
```

```
            return {
                'lines': [],
                'polylines': [],
                'circles': [],
                'layer_names': []
            }
        except Exception as e:
            print(f"DXF读取失败: {e}")
            return None

    def classify_elements(self, dxf_data: Dict) -> Dict:
        """根据图层名称分类几何元素"""
        classified = {'walls': [], 'roads': [], 'water': [], 'plants':
            [], 'rocks': []}

        for element_type in ['lines', 'polylines', 'circles']:
            for element in dxf_data.get(element_type, []):
                layer = element.get('layer_name', '').lower()
                classified_flag = False

                for category, keywords in self.layer_keywords.items():
                    if any(kw in layer for kw in keywords):
                        classified[category].append(element)
                        classified_flag = True
                        break

                if not classified_flag:
                    classified.setdefault('others', []).append(element)

        return classified


# A2. 多目标路径优化模型
class MultiObjectivePathOptimizer:
    """多目标路径优化模型: F = √(P × V) × U"""

    def __init__(self, graph_data: Dict):
        self.nodes = graph_data.get('nodes', {})
        self.edges = graph_data.get('edges', {})
```

```
def calculate_path_dynamics_score(self, edge_id: str) -> float:
    """计算路径动态性评分 P = (L'×T'×C')"""
    edge_data = self.edges.get(edge_id, {})

    # 简化计算: 基于路径长度的高斯函数
    length = edge_data.get('length', 60.0)
    length_score = math.exp(-((length - 60.0) ** 2) / 1800.0)

    # 转折点对数函数
    turns = edge_data.get('turns', 0)
    turn_score = math.log(1 + turns) / math.log(26)

    # 交叉点负指数
    crossings = edge_data.get('crossings', 0)
    cross_score = math.exp(-4 * crossings / 6)

    return (length_score * turn_score * cross_score) ** (1/3)

def calculate_scenic_variety_score(self, edge_id: str) -> float:
    """计算异景程度评分 V = √(VS'×VR'×PF'×BR'×OS')"""
    edge_data = self.edges.get(edge_id, {})

    # 视域面积反正切函数
    area = edge_data.get('viewshed_area', 500000)
    area_score = math.atan(area / 500000) / math.atan(3)

    # 可见度比例
    visibility = edge_data.get('visibility_ratio', 0.3)
    vis_score = max(0, visibility)

    return (area_score * vis_score * 0.5 * 0.7 * 0.8) ** 0.2

def evaluate_multiobjective_function(self, path_edges: List[str])
   -> Tuple[float, Dict]:
    """计算多目标优化函数 F = √(P' × V') × U"""
    if not path_edges:
        return 0.0, {'error': 'Empty path'}
```

```
        P_scores = [self.calculate_path_dynamics_score(e) for e in
            path_edges]
        V_scores = [self.calculate_scenic_variety_score(e) for e in
            path_edges]

        P_avg = mean(P_scores)
        V_avg = mean(V_scores)

        # 唯一性系数
        unique_edges = len(set(path_edges))
        U = unique_edges / len(path_edges) if path_edges else 1.0

        F = math.sqrt(P_avg * V_avg) * U

        return F, {'P': P_avg, 'V': V_avg, 'U': U, 'F': F}


# A3. 视域分析算法
class ViewshedAnalyzer:
    """视域分析器, 360度射线投射算法"""

    def __init__(self, ray_count: int = 360, max_distance: float =
        100.0):
        self.ray_count = ray_count
        self.max_distance = max_distance

    def analyze_viewpoint(self, viewpoint: Tuple[float, float],
        obstacles: List[Dict]) -> Dict:
        """分析观察点的视域"""
        vx, vy = viewpoint
        visible_points = []
        blocked_count = 0

        # 射线投射算法
        for i in range(self.ray_count):
            angle = 2 * math.pi * i / self.ray_count
            dx, dy = math.cos(angle), math.sin(angle)

            # 简化的障碍物检测
```

```
            hit_distance = self.max_distance
            for obstacle in obstacles:
                # 简化计算：假设与障碍物的交点
                if self._ray_intersects_obstacle(vx, vy, dx, dy,
                    obstacle):
                    hit_distance = min(hit_distance, obstacle.get('
                        distance', 50))
                    blocked_count += 1
                    break

            # 记录可见点
            visible_x = vx + hit_distance * dx
            visible_y = vy + hit_distance * dy
            visible_points.append((visible_x, visible_y))

        # 计算视域统计
        viewshed_area = self._calculate_polygon_area(visible_points)
        visibility_ratio = 1.0 - (blocked_count / self.ray_count)

        return {
            'viewshed_area': viewshed_area,
            'visibility_ratio': visibility_ratio,
            'blocked_rays': blocked_count,
            'visible_points': visible_points
        }

    def _ray_intersects_obstacle(self, vx, vy, dx, dy, obstacle) ->
        bool:
        """简化的射线与障碍物交点检测"""
        # 简化实现
        return obstacle.get('type') in ['building', 'plant'] and abs(
            obstacle.get('x', 0) - vx) < 50

    def _calculate_polygon_area(self, points: List[Tuple[float, float
        ]]) -> float:
        """计算多边形面积"""
        if len(points) < 3:
            return 0.0
```

```
        # 简化的面积计算
        n = len(points)
        area = 0.0
        for i in range(n):
            j = (i + 1) % n
            area += points[i][0] * points[j][1]
            area -= points[j][0] * points[i][1]
        return abs(area) / 2.0


# A4. 开阔围合量化算法
class OpennessEnclosureQuantifier:
    """开阔围合量化器，基于黄金分割阈值0.618"""

    def __init__(self, golden_ratio_threshold: float = 0.618):
        self.golden_ratio_threshold = golden_ratio_threshold

    def calculate_openness_index(self, spatial_features: Dict) -> float
        :
        """计算开阔度指数"""
        viewshed_area = spatial_features.get('viewshed_area', 0)
        visibility_ratio = spatial_features.get('visibility_ratio', 0)
        visual_permeability = spatial_features.get('visual_permeability
            ', 0)

        # 加权计算开阔度
        openness = (
            0.4 * self._normalize(viewshed_area, 100000) +
            0.3 * visibility_ratio +
            0.3 * visual_permeability
        )

        return min(1.0, max(0.0, openness))

    def calculate_enclosure_index(self, spatial_features: Dict) ->
        float:
        """计算围合度指数"""
        visual_obstruction = spatial_features.get('visual_obstruction',
            0)
```

```
        blocking_density = spatial_features.get('blocking_density', 0)
        boundary_definition = spatial_features.get('boundary_definition
            ', 0)

        # 加权计算围合度
        enclosure = (
            0.4 * visual_obstruction +
            0.3 * self._normalize(blocking_density, 0.1) +
            0.3 * boundary_definition
        )

        return min(1.0, max(0.0, enclosure))

    def classify_spatial_type(self, openness: float, enclosure: float)
        -> str:
        """基于黄金分割阈值分类空间类型"""
        if openness >= self.golden_ratio_threshold and enclosure < self
            .golden_ratio_threshold:
            return '开阔型'
        elif openness < self.golden_ratio_threshold and enclosure >=
            self.golden_ratio_threshold:
            return '围合型'
        elif openness >= self.golden_ratio_threshold and enclosure >=
            self.golden_ratio_threshold:
            return '平衡型'
        else:
            return '过渡型'

    def _normalize(self, value: float, max_value: float) -> float:
        """归一化到[0,1]区间"""
        return min(1.0, value / max_value) if max_value > 0 else 0.0


# A5. 幻境感评分模型
class PhantasmScoringModel:
    """幻境感评分模型，四维评价体系"""

    def __init__(self, baseline_score: float = 100.0):
        self.baseline_score = baseline_score
```

```
        # 四维权重配置
        self.weights = {
            'theme_diversity': 0.25,        # 主题多样性
            'rhythm_variation': 0.35,       # 节奏变化
            'theme_matching': 0.25,         # 主题匹配
            'element_proportion': 0.15      # 元素比例
        }

    def calculate_theme_diversity_score(self, spatial_points: List[Dict
        ]) -> float:
        """计算主题多样性评分（Shannon熵）"""
        if not spatial_points:
            return 0.0

        # 统计主题分布
        theme_counts = {}
        for point in spatial_points:
            theme_id = point.get('theme_id', 0)
            theme_counts[theme_id] = theme_counts.get(theme_id, 0) + 1

        # Shannon多样性指数
        total = len(spatial_points)
        shannon_entropy = 0.0
        for count in theme_counts.values():
            if count > 0:
                p = count / total
                shannon_entropy -= p * math.log2(p)

        # 归一化
        max_entropy = math.log2(len(theme_counts)) if len(theme_counts)
            > 1 else 1.0
        return shannon_entropy / max_entropy if max_entropy > 1e-6 else
            0.0

    def calculate_rhythm_variation_score(self, spatial_points: List[
        Dict]) -> float:
        """计算开阔围合节奏变化评分"""
        if len(spatial_points) < 2:
```

```
            return 0.0

        # 计算相邻点间的开阔围合变化
        variations = []
        for i in range(len(spatial_points) - 1):
            curr = spatial_points[i]
            next_point = spatial_points[i + 1]

            openness_var = abs(next_point.get('openness', 0) - curr.get
                ('openness', 0))
            enclosure_var = abs(next_point.get('enclosure', 0) - curr.
                get('enclosure', 0))

            combined_var = math.sqrt(openness_var**2 + enclosure_var
                **2)
            variations.append(combined_var)

        # 计算变化的标准差作为节奏丰富性
        if variations:
            rhythm_score = pstdev(variations) if len(variations) > 1
                else 0.0
            return min(1.0, rhythm_score * 2.0)

        return 0.0

    def calculate_comprehensive_score(self, garden_data: Dict) -> Tuple
        [float, Dict]:
        """计算综合幻境感评分"""
        spatial_points = garden_data.get('spatial_points', [])

        # 四维评分
        theme_diversity = self.calculate_theme_diversity_score(
            spatial_points)
        rhythm_variation = self.calculate_rhythm_variation_score(
            spatial_points)
        theme_matching = 0.7  # 简化计算
        element_proportion = 0.8  # 简化计算

        # 加权综合评分
```

```
        weighted_score = (
            self.weights['theme_diversity'] * theme_diversity +
            self.weights['rhythm_variation'] * rhythm_variation +
            self.weights['theme_matching'] * theme_matching +
            self.weights['element_proportion'] * element_proportion
        )

        final_score = weighted_score * self.baseline_score

        details = {
            'final_score': final_score,
            'theme_diversity': theme_diversity,
            'rhythm_variation': rhythm_variation,
            'theme_matching': theme_matching,
            'element_proportion': element_proportion
        }

        return final_score, details


# A6. 频谱相似性分析
class SpectralSimilarityAnalyzer:
    """频谱相似性分析器，基于图拉普拉斯矩阵特征值"""

    def __init__(self, eigenvalue_dim: int = 20):
        self.eigenvalue_dim = eigenvalue_dim

    def extract_spectral_features(self, graph_data: Dict) -> List[float
        ]:
        """提取图的频谱特征（简化版）"""
        nodes = graph_data.get('nodes', {})
        edges = graph_data.get('edges', {})

        if len(nodes) == 0:
            return [0.0] * self.eigenvalue_dim

        # 简化的频谱特征计算
        node_count = len(nodes)
        edge_count = len(edges)
```

```
        # 基于图的基本统计构建简化频谱特征
        features = []
        for i in range(self.eigenvalue_dim):
            # 简化的特征值近似
            feature_val = (i + 1) / self.eigenvalue_dim * (edge_count /
                max(node_count, 1))
            features.append(feature_val)

        return features

    def calculate_similarity(self, features1: List[float], features2:
        List[float]) -> float:
        """计算频谱相似性"""
        if len(features1) != len(features2):
            return 0.0

        # 简化的欧氏距离计算
        distance = math.sqrt(sum((a - b)**2 for a, b in zip(features1,
            features2)))

        # 转换为相似性
        similarity = math.exp(-distance / 10.0)
        return min(1.0, max(0.0, similarity))


# A7. 形状相似性分析
class ShapeSimilarityAnalyzer:
    """形状相似性分析器，基于凸包几何特征"""

    def __init__(self):
        self.feature_weights = {
            'area_ratio': 0.25,
            'circularity': 0.25,
            'aspect_ratio': 0.25,
            'compactness': 0.25
        }

    def extract_shape_features(self, spatial_coords: List[Tuple[float,
```

```
        float]]) -> Dict:
        """提取形状特征"""
        if len(spatial_coords) < 3:
            return {'error': 'Insufficient points'}

        # 简化的形状特征计算
        # 计算边界框
        x_coords = [p[0] for p in spatial_coords]
        y_coords = [p[1] for p in spatial_coords]

        min_x, max_x = min(x_coords), max(x_coords)
        min_y, max_y = min(y_coords), max(y_coords)

        width = max_x - min_x
        height = max_y - min_y

        # 简化的特征计算
        area = width * height
        perimeter = 2 * (width + height)

        circularity = (4 * math.pi * area) / (perimeter ** 2) if
            perimeter > 0 else 0
        aspect_ratio = max(width, height) / min(width, height) if min(
            width, height) > 0 else 1
        compactness = (perimeter ** 2) / (4 * math.pi * area) if area >
            0 else 1

        return {
            'area': area,
            'circularity': circularity,
            'aspect_ratio': aspect_ratio,
            'compactness': compactness
        }

    def calculate_similarity(self, features1: Dict, features2: Dict) ->
        float:
        """计算形状相似性"""
        if 'error' in features1 or 'error' in features2:
            return 0.0
```

```
        # 计算各维度相似性
        area_sim = min(features1['area'], features2['area']) / max(
            features1['area'], features2['area'])
        circ_sim = 1.0 - abs(features1['circularity'] - features2['
            circularity'])
        aspect_sim = min(features1['aspect_ratio'], features2['
            aspect_ratio']) / max(features1['aspect_ratio'], features2[
            'aspect_ratio'])
        comp_sim = min(features1['compactness'], features2['compactness
            ']) / max(features1['compactness'], features2['compactness'
            ])

        # 加权综合相似性
        similarity = (
            self.feature_weights['area_ratio'] * area_sim +
            self.feature_weights['circularity'] * circ_sim +
            self.feature_weights['aspect_ratio'] * aspect_sim +
            self.feature_weights['compactness'] * comp_sim
        )

        return min(1.0, max(0.0, similarity))


# 使用示例和测试代码
if __name__ == "__main__":
    print("江南古典园林美学特征建模系统 - 核心算法演示")
    print("="*60)

    # 示例图数据
    sample_graph = {
        'nodes': {
            'entrance_1': {'type': 'entrance', 'x': 0, 'y': 0},
            'attraction_1': {'type': 'attraction', 'x': 100, 'y': 100},
            'attraction_2': {'type': 'attraction', 'x': 200, 'y': 200}
        },
        'edges': {
            'path_1': {'length': 60.0, 'turns': 3, 'crossings': 0,
                       'viewshed_area': 800000, 'visibility_ratio': 0.7}
```

```
    }
}

# 1. 多目标路径优化演示
print("\n1. 多目标路径优化模型:")
optimizer = MultiObjectivePathOptimizer(sample_graph)
F, metrics = optimizer.evaluate_multiobjective_function(['path_1'])
print(f"   优化函数值: {F:.4f}")
print(f"   路径动态性: {metrics['P']:.4f}")
print(f"   异景程度: {metrics['V']:.4f}")
print(f"   路径唯一性: {metrics['U']:.4f}")

# 2. 视域分析演示
print("\n2. 视域分析算法:")
viewshed = ViewshedAnalyzer()
obstacles = [{'type': 'building', 'x': 50, 'y': 50, 'distance':
    30}]
view_result = viewshed.analyze_viewpoint((0, 0), obstacles)
print(f"   视域面积: {view_result['viewshed_area']:.2f}")
print(f"   可见性比例: {view_result['visibility_ratio']:.3f}")

# 3. 幻境感评分演示
print("\n3. 幻境感评分模型:")
phantasm = PhantasmScoringModel()
sample_garden = {
    'spatial_points': [
        {'theme_id': 0, 'openness': 0.7, 'enclosure': 0.3},
        {'theme_id': 1, 'openness': 0.4, 'enclosure': 0.6},
        {'theme_id': 0, 'openness': 0.8, 'enclosure': 0.2}
    ]
}
score, details = phantasm.calculate_comprehensive_score(
    sample_garden)
print(f"   综合评分: {score:.2f}分")
print(f"   主题多样性: {details['theme_diversity']:.3f}")
print(f"   节奏变化: {details['rhythm_variation']:.3f}")

# 4. 频谱相似性演示
print("\n4. 频谱相似性分析:")
```

```
    spectral = SpectralSimilarityAnalyzer()
    features1 = spectral.extract_spectral_features(sample_graph)
    features2 = spectral.extract_spectral_features(sample_graph)
    similarity = spectral.calculate_similarity(features1, features2)
    print(f"   频谱相似性: {similarity:.4f}")

    # 5. 形状相似性演示
    print("\n5. 形状相似性分析:")
    shape = ShapeSimilarityAnalyzer()
    coords1 = [(0, 0), (100, 0), (100, 100), (0, 100)]
    coords2 = [(0, 0), (150, 0), (75, 130)]
    features1 = shape.extract_shape_features(coords1)
    features2 = shape.extract_shape_features(coords2)
    similarity = shape.calculate_similarity(features1, features2)
    print(f"   形状相似性: {similarity:.4f}")

    print(f"\n系统信息:")
    print(f"- 核心算法模块: 7个")
    print(f"- 数学模型: F = √(P × V) × U + 四维幻境评分")
    print(f"- 分析园林: 11个经典中国园林")
    print("="*60)
```