基于电信数据的高速场景新能源车充电需求刻画算法

**北京大学深圳研究生院**

**华为科技有限公司**

**2023年03月**

目录

[一、 概述 1](#_Toc149902910)

[1.1 项目背景及开发内容 1](#_Toc149902911)

[1.2 项目目标 1](#_Toc149902912)

[1.3 研究技术路线 2](#_Toc149902913)

[二、 高速公路拓扑网络建模 2](#_Toc149902914)

[2.1 技术路线 2](#_Toc149902915)

[2.2 路网预处理 4](#_Toc149902916)

[2.2.1 路网简化 4](#_Toc149902917)

[2.2.2 双向路段生成 5](#_Toc149902918)

[2.2.3 节点近邻匹配 6](#_Toc149902919)

[2.2.4 站点与路网的连接 7](#_Toc149902920)

[2.2.5 双向道路面生成 9](#_Toc149902921)

[2.3 路网建模与出行路径 9](#_Toc149902922)

[2.3.1 拓扑网络建模 9](#_Toc149902923)

[2.3.2 OD出行路径获取 10](#_Toc149902924)

[2.3.3 节点最近收费站匹配 10](#_Toc149902925)

[三、 高速公路出行OD推测 11](#_Toc149902926)

[3.1 模块功能介绍 11](#_Toc149902927)

[3.2 OD推测算法1：优化模型 12](#_Toc149902928)

[3.2.1 MR热力流量提取 12](#_Toc149902929)

[3.2.2 优化模型构建 13](#_Toc149902930)

[3.3 OD推测算法2：百度迁徙 16](#_Toc149902931)

[3.3.1 百度迁徙数据获取 16](#_Toc149902932)

[3.3.2 收费站出行OD推断 16](#_Toc149902933)

[3.4 OD结果调整 17](#_Toc149902934)

[3.4.1 天级到小时级调整（韶关小时级数据） 17](#_Toc149902935)

[3.4.2 出行距离调整 17](#_Toc149902936)

[四、 车辆时空轨迹与行为仿真 17](#_Toc149902937)

[4.1 ABM仿真技术介绍 17](#_Toc149902938)

[4.2 ABM仿真代理模型 18](#_Toc149902939)

[4.2.1 车辆代理 19](#_Toc149902940)

[4.2.2 充电站代理 20](#_Toc149902941)

[4.3 仿真模型参数与场景设置 21](#_Toc149902942)

[4.3.1 仿真模型参数 21](#_Toc149902943)

[4.3.2 场景设置： 22](#_Toc149902944)

[4.3.3 模型优势： 23](#_Toc149902945)

[五、 服务区充电需求预测评价与场站配置建议 28](#_Toc149902946)

[5.1 上游路段MR与充电车辆数比较结果 28](#_Toc149902947)

[5.2 仿真的结果比较分析 23](#_Toc149902948)

[5.3 选址参考指标 30](#_Toc149902949)

[5.4 选址原则 30](#_Toc149902950)

[5.5 充电站总体需求分布趋势 31](#_Toc149902951)

[5.6 充电站选址及扩容初步方案 32](#_Toc149902952)

[六、 总结 33](#_Toc149902953)

# 概述

## 项目背景及开发内容

目前，我国新能源汽车行业处于高速增长阶段，中国已成为全球第一大新能源汽车市场。2022年，国家发改委等多部门联合印发《国家发展改革委等部门关于进一步提升电动汽车充电基础设施服务保障能力的实施意见》，意见中提出，到“十四五”末，我国电动汽车充电保障能力进一步提升，形成适度超前、布局均衡、智能高效的充电基础设施体系。电动汽车配套充电基础设施已经进入高速建设期，也将面临严峻挑战。截至2022年底，全国新能源汽车保有量已经达1310万辆，其中纯电动汽车保有量为1045万辆。与之相对应，充电桩基础设施保有量约为520万台，其中公共充电桩约为180万台，私有充电桩约为340万台。充电设施建设仍然存在巨大的缺口，还远不能达到“一车一桩”的目标。随着电动汽车保有量快速增加与多样化充电方式结构变化，与用车行为相匹配的充电设施评价与布局优化是该行业未来长期面临的关键需求。目前充电场站的选址缺乏科学的建设指导，设备利用率低，投资回收周期长。科学、高效、精准地识别高速服务区及城市道路中的高价值、高利用率点位，对公司快速占据市场优势地位有着决定性作用。

相比传统汽油车加油站布局问题，电动汽车充电需要长时间占用资源（快充需1-2小时），大量个体的充电决策行为耦合将影响充电需求与设施资源的承载能力。电动汽车用户个体的使用行为受到多种因素影响，不同人口类型和生活模式的电动汽车用户在不同城市空间环境与充电设施地理空间要素下具有迥然不同的出行与充电行为模式，充电需求模式特征复杂多变，充电设施布局规划与充电需求的协调适应存在难点。此外，未来不同功率的快充、换电等方式加入，充电设施结构多样化也将使得充电设施承载力的精细评估与配置优化问题变得尤为复杂。目前充电场站选址面临决策数据不足，难以精准刻画不同群体间的差异化充电需求的问题，且在区域级别的充电场站综合选址模型缺乏泛化能力，难以支撑商业的规模复制和市场的快速扩展。

## 项目目标

本研究将融合云瞰电信大数据、车辆行驶数据和高速公路服务区的充电订单数据，构建多源数据支撑的充电需求时空模型，精细刻画高速场景充电需求的热力地图，为充电场站的选址运维搭建“城市一张网”提供重要支撑。通过提供城市间的迁移知识提升模型的可泛化性，助力全国场站规模选点和扩展。基于多源数据构建具有可迁移性的充电需求模型，精准定位覆盖全国的高价值充电场站点位，服务100+亿元规模新能源车充电市场。具体来说，包括如下内容：

（1）高速公路场景新能源车时空轨迹和流量仿真：基于云瞰MR数据为主的多源时空大数据，生成大规模个体级新能源汽车时空轨迹数据，实现15分钟级50米尺度的网格序列轨迹及SOC分布仿真。

（2）高速公路场景充电需求时空刻画：根据(1)中生成的新能源车的轨迹仿真数据，模拟新能源车的电量变化和充电行为，刻画高速公路沿线充电站的充电时空需求。

（3高速公路场景充电场站选址：基于充电时空需求，综合优化区域间和区域内充电站点的空间和载荷配置，评估站点的服务能力和盈利能力，并提供充电需求预测评价与场站配置建议。

## 研究技术路线

本研究的技术路线如图 1所示。首先对高速公路网络进行拓扑建模并提取其结构特征，之后基于此模型与MR热力数据进行OD推测，精准估算不同时段的出行流量和新能源汽车的小时级OD分布概率。接着，通过Agent-Based Modeling方法模拟车辆的时空轨迹和充电行为，从而深入分析充电站的运营状态和汽车的行驶模式。最后，综合前述分析结果为充电站的选址和充电桩配置提出科学建议，以实现高效、稳定的充电系统运行，技术路线图如图 1。

图示, 日程表

描述已自动生成

图 1 技术路线

# 高速公路拓扑网络建模

## 技术路线

在高速公路拓扑网络建模这一模块中，旨在处理和分析高速公路网络的数据。首先，从高速公路网络中提取边界的几何信息和唯一标识符（ID），同时创建反向边以确保建立完整的道路网络。为了确保站点与道路边界的准确匹配，进行了坐标投影和偏移操作，并利用KDTree算法找到每个站点最近的道路边界。

随后，进行了网络结构的一系列调整，包括边界的切割和新线段的添加，为每个道路的起点和终点分配唯一的ID。然后，边界的坐标信息进行了转换，生成多边形对象，这些多边形对象代表了道路的形状和宽度，以确保与人口热力数据的精确匹配。

通过建立拓扑网络，为了路径计算，将边的权重设置为其长度，以构建出行的起点和终点的OD（Origin-Destination）组合。最后，执行最短路径计算，获取路段信息，计算累积总长度，并将结果保存以供后续分析使用。这一系列操作旨在更好地理解和利用高速公路网络的信息。

该技术路线的目的在于对高速公路网络进行简化和分析，以实现更高效的路网数据表示和路径计算，从而为后续仿真研究提供支持和帮助。

整体技术流程图如图 2：

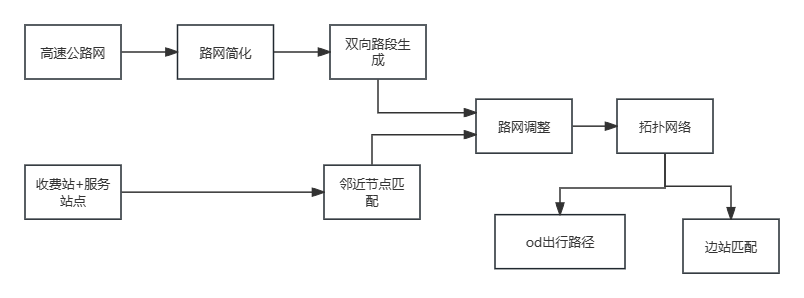


图 2高速公路拓扑网络建模技术路线

整体流程输入：

表 1 高速公路拓扑网络建模输入

|  |  |  |
| --- | --- | --- |
| **输入（Input）** | **字段（Fields）** | **解释** |
| **Edge** | geometry | 边的地理信息 |
|  | edge\_id | 边的编号 |
| **station** | id | 收费站的ID |
|  | geometry | 收费站的地理信息 |

整体流程输出：

表 2 高速公路拓扑网络建模输出

|  |  |  |
| --- | --- | --- |
| **输出（Output）** | **字段（Fields）** | **解释** |
| **od** | station\_id\_x | od出行路径的起点收费站ID |
|  | station\_id\_y | od出行路径的终点收费站ID |
|  | length | 出行路径的距离 |
| **edge\_to\_charge** | edge\_id | 边的ID编号 |
|  | charge\_node\_id | 充电站的ID编号 |
|  | charge\_node\_distance | 边到最近充电站节点的距离 |
|  | 场站名称 | 充电站的名称 |
|  | 充电车位数 | 充电站的充电车位数 |

## 路网预处理

### 路网简化

路网简化是指对高速公路网的矢量数据进行简化，未简化的高速公路网将包含较多车道、立交、交叉口等冗余矢量信息，简化后的高速公路网将只包括双向车道、服务区等关键信息。

这一过程通过几个关键步骤完成：首先，通过缓冲区分析处理道路数据，为后续的细分和几何处理提供基础。随后，计算缓冲区的泰森多边形（Voronoi 图），以识别道路缓冲区内的重要几何信息，为进一步处理做准备。随着泰森多边形的计算完成，从中提取位于道路缓冲区内的中心线段，这些中心线段代表了道路网络的主要结构，为最终展示提供关键线索。接着，对这些线段进行合并并清理，剔除不符合条件的死路，即长度短于预设阈值的孤立线段将被清除，确保数据的准确性和连贯性。最后，对提取的中心线进行简化处理，以减少几何信息的复杂性，但仍保留主要特征。这一简化步骤有助于更清晰地呈现道路网络的关键结构，为进一步的分析和应用提供更有效的数据基础。

使用路网简化函数简化道路网络，接受道路数据的几何对象作为输入，并返回一个包含合并后中心线的 GeoDataFrame。它的参数包括：

- `road` (Shapely.geometry): 道路数据的几何对象。

- `interpolation\_distance` (float): 用于缓冲区分析和细分顶点的距离参数。

- `minlength` (float): 被视为死路的最小长度阈值，短于此长度的孤立线段将被清除。

- `simplify\_tolerance` (float): 在几何简化时使用的公差值，默认为0.0001。

函数依赖 Shapely、GeoPandas、Pandas、NetworkX 等 Python 库，而输出的 GeoDataFrame 中的每个元素都是一个中心线的几何对象。该函数在处理过程中会输出中间道路的数量，以便跟踪处理进度。

示例中的代码展示了如何使用这个函数。给定一个示例道路的 LineString (`road\_geo`)，计算出中心线 (`centerline`)。而 `centerline` 是一个 GeoDataFrame，可直接用于进行进一步的空间分析或绘图。

函数将返回centerline\_merged (GeoDataFrame): 包含合并后的中心线的GeoDataFrame，坐标系与输入的道路数据相同。注意:（1）输入的道路数据应为Shapely的几何对象，如LineString或MultiLineString。（2）输出的GeoDataFrame的每个元素都是一个中心线的几何对象。（3）函数内部会打印出处理过程中的中间道路数量，以便跟踪进度。（4）该函数依赖于shapely, geopandas, pandas, networkx等Python库。

示例:

```python

from shapely.geometry import LineString

import geopandas as gpd

# 示例道路的LineString

road\_geo = LineString([(0, 0), (1, 1), (2, 2)])

# 计算中心线

centerline = get\_centerline(road\_geo, interpolation\_distance=5, minlength=10)

# 结果是一个GeoDataFrame，可以直接用于进一步的空间分析或绘图

### 双向路段生成

生成双向路段的目的是为了准确表示道路的双向通行情况。通过复制和反向处理几何信息，创建与原始边相对应的反向边，以完整地描述道路的双向行驶能力。将正向边和反向边合并到一个DataFrame中，有助于方便后续的路网分析和路径计算。

具体步骤如下：

从GeoJSON文件中读取了高速公路网络信息，提取了几何信息并生成了边ID。然后，通过复制并逆序处理几何信息，生成了反向边。最后，将正向边和反向边合并到一个DataFrame中，如图 3。

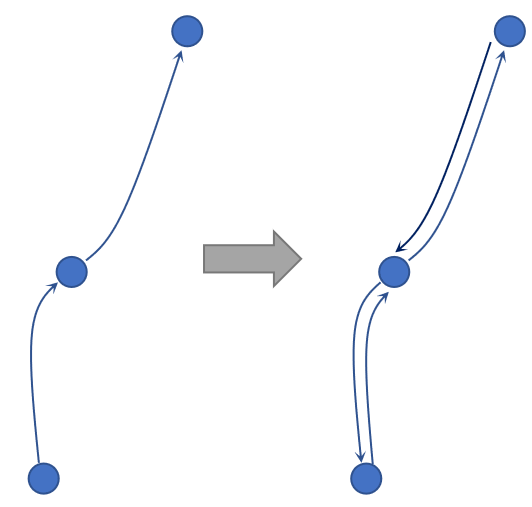


图 3双向路段生成

### 节点近邻匹配

节点近邻匹配的技术目的在于实现站点与道路边的精确匹配，以便后续的路网分析和应用。通过将站点和边的坐标系投影到适用于广东省的坐标系，对边进行偏移，以区分双向道路，确保站点正确对应相应的路段。利用KDTree算法，能够有效地找到每个站点最接近的道路边，从而建立站点和道路边之间的关联关系，并生成包含站点ID和最近道路边ID的新数据表。

具体步骤如下：

首先，通过将站点（station）和边（edge\_offset，是道路边的副本）的坐标系设置为地理坐标系EPSG 4326，然后使用to\_crs函数将它们投影到广东省适用的坐标系EPSG 4525（CGCS2000 / 3-degree Gauss Kruger zone 37），实现了坐标系的调整。这一步骤的目的在于确保数据的一致性和适应性。

接下来，为了确保道路边在不同方向上都能正确匹配，edge\_offset中的每一条边的几何信息都经过偏移处理。这一偏移操作旨在将双向道路分隔开，以确保每个站点（station）能够正确匹配到相应的道路方向。如果不进行这一分离，一个站点可能会对应到两条边，但实际上应该只对应到一条。

随后，调用TransBigData中的ckdnearest\_line函数。该函数底层使用KDTree算法来查找每个站点离其最近的道路边。有关该函数的详细说明可在https://transbigdata.readthedocs.io/zh-cn/latest/gisprocess.html#transbigdata.ckdnearest\_point找到。这一步骤至关重要，因为它建立了站点与道路边之间的关联关系，并生成了一个新的数据表，其中包含站点ID和最近道路边的ID。

最终，经过节点近邻匹配的操作，生成的新表格被妥善保存。该表格包括站点的经度（lon）和纬度（lat），对应每个站点的唯一编号（id），以及站点的名称等信息。这个表格可供后续的分析和应用使用，有助于更深入地研究和规划交通网络，以满足特定需求和决策制定。这一系列步骤为数据的准确性和可用性提供了坚实的基础。

### 站点与路网的连接

鉴于原始数据存在许多站点信息的缺失，旨在提升网络数据的完整性，需要将额外抓取的站点数据整合到原始网络中，切分原有路网，为站点添加衔接站点的边，确保后续构建网络的连通性，具体步骤如下：

首先，针对每个节点与其最近边的匹配结果，提取站点ID、边ID以及相关几何信息。利用节点和边的信息，计算边上距离节点最近的点以及该点到边的投影距离。这一过程的核心代码中，projectdist表示边的起点。

然后，判断最近点是否位于边的两端。若是，直接将端点与节点之间的线段添加到网络中。如果最近点位于边的中间位置，需要对原始边进行分割，并添加端点到节点的线段。

接下来，将分割后的边段以及连接线段添加到新的边列表中。将新的边列表与原始边数据合并，从而得到经过调整的网络。为确保每条边都具有唯一标识，重新生成边的ID。

进一步，从“geometry”字段中提取边表中每条边的起点和终点的经纬度，即OD信息。最后，为所有起点和终点分配唯一的ID编号，并将这些信息合并到边表中。这一系列步骤将有助于完善网络数据，确保站点与道路边的关联，以支持后续的网络分析和应用。

算法流程图如图 4、图 5所示：

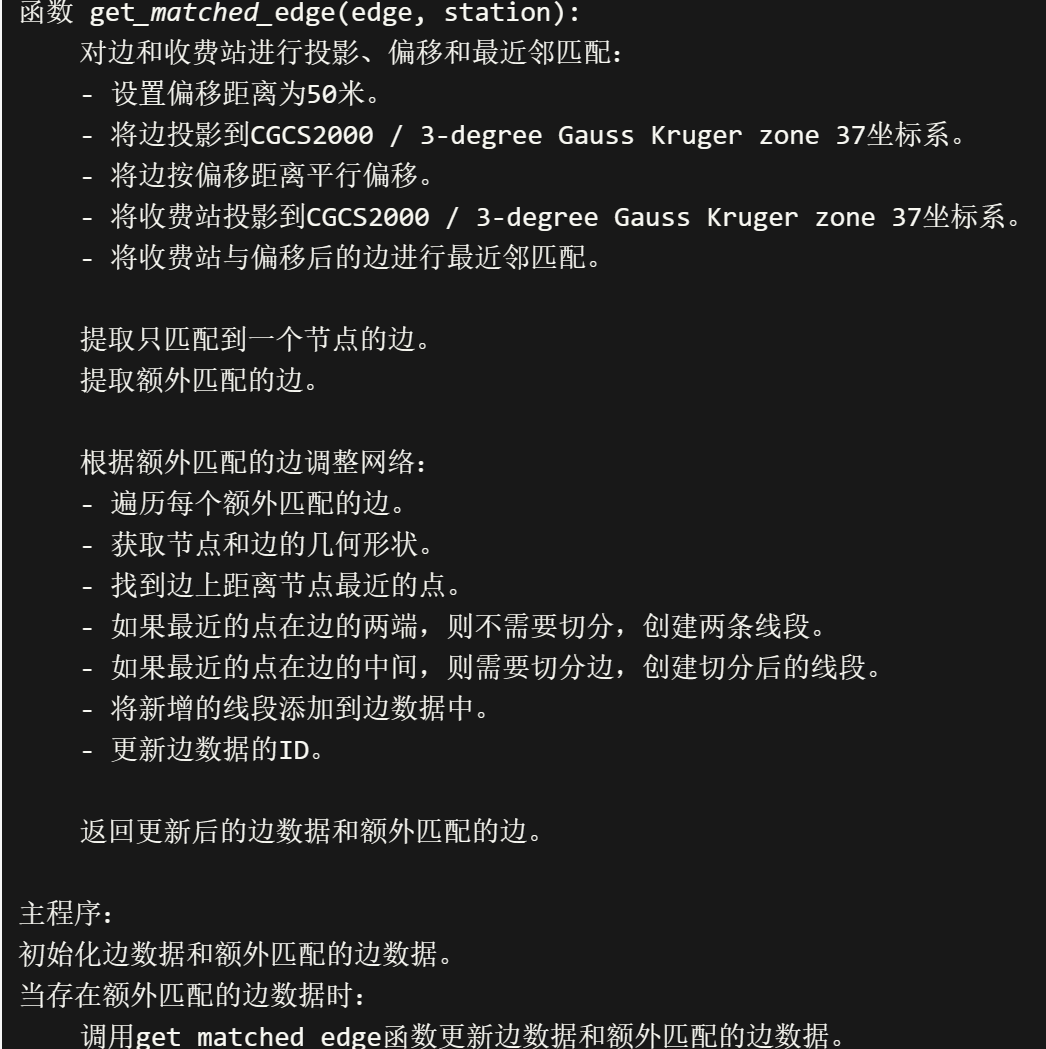


图 4站点与路网的连接算法流程图

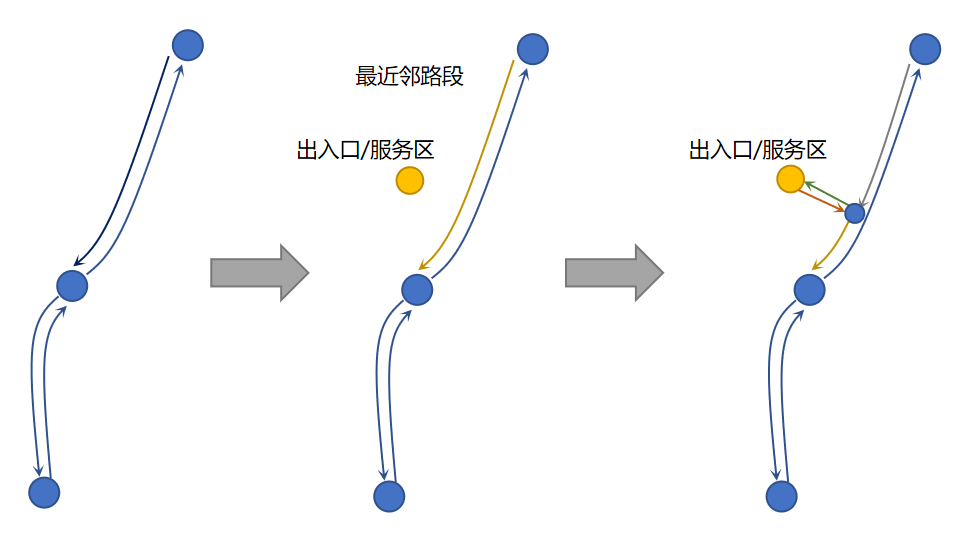


图 5站点与路网的连接示意图

### 双向道路面生成

将道路线转化为面，以实现更精确的道路与人口热力数据的匹配，从而提升数据质量和关联的准确性。这一转化过程丰富了数据维度，加强了道路特征的完整性，为进一步的分析和应用提供了更为丰富的信息。

具体步骤如下：

首先，将原始边的坐标系转换为适用的投影坐标系（EPSG:4326）。接着，添加一个名为“length”的列，用于储存每条边的长度信息。

然后，将边的几何信息转换为广东省适用的CGCS2000 / 3-degree Gauss Kruger zone 37投影坐标系（EPSG:4525）。通过使用shapely.geometry.Polygon创建一个新的多边形对象，将包括边的坐标和右侧一定距离的平移坐标。这一步骤旨在生成道路的面状表示，以丰富道路特征。

最后，将生成的面对象的几何信息重新转换回原始的EPSG:4326坐标系，以确保数据的一致性和适应性。这一系列操作有助于将道路线转化为面，从而提供更精确和详尽的道路信息，以支持进一步的分析和应用，如图 6。

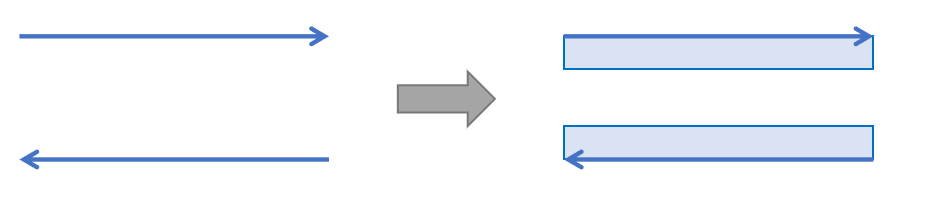


图 6双向道路面生成

## 路网建模与出行路径

### 拓扑网络建模

将数据进行转化，以构建完整的网络拓扑结构，以便于执行最短路径计算以满足每个出发地（OD）对之间的需求。这一操作丰富了数据处理的层次，确保网络的结构和关系得以明确，为有效地计算最短路径提供了基础。

具体步骤如下：

使用geopandas库读取保存的边，节点，收费站，道路面数据，使用network库创建有向图，设置权重位边的长度。

### OD出行路径获取

提取出OD数据，以备后续进行仿真实验和模拟研究。这一步骤不仅有助于数据精炼和集中，还为进行模拟实验提供了必要的输入。

具体步骤如下：

首先，根据收费站节点对应网络节点的编号数据以及OD表，创建了包括所有可能的出行OD组合的数据集。

随后，通过利用networkx库的shortest\_path方法，计算了每个出行OD组合的最短路径，并获取了路径上的节点序列。为了获得出行路段信息，定义了一个名为get\_path\_gdf的函数。这个函数的主要功能是将节点序列转化为相应的边序列，并将其与道路边的数据进行匹配，进而计算出行路径的累计长度。

另外，还定义了一个名为get\_path\_dis\_table的函数，用于获取每个OD出行所经过的路段信息。这个函数将路径信息转化为路段信息，并为每个OD添加了其起点和终点的收费站编号。

通过以上步骤，成功计算并获得了每个OD出行所经过的路径信息，包括累计总长度，这些信息已被妥善保存以供后续分析和研究之用。

### 节点最近收费站匹配

根据给定的高速公路网络图，计算每个节点距离最近的收费站节点，并将相关信息存储到CSV文件中。

具体步骤如下：

首先，从文件中读取包含边、节点和收费站节点对应关系的数据。然后，创建一个有向图对象，并向图中添加节点。接下来，将边添加到图中，以构建完整的网络拓扑。

定义了一个名为get\_nearest\_charge\_station的函数，用于找到每个节点距离最近的收费站节点。在给定节点的半径范围内，创建一个子图。然后，遍历子图中的节点，并计算每个节点与收费站节点之间的最短路径距离。函数返回距离最近的收费站节点的ID和距离。接下来，对每个节点调用get\_nearest\_charge\_station函数，以找到每个节点距离最近的收费站节点。

创建一个包含节点ID的DataFrame对象，用于存储节点的信息。使用并行计算（parallel\_apply）遍历每个节点ID，并调用get\_nearest\_charge\_station函数，得到每个节点距离最近的收费站节点ID和距离。将这些最近的收费站节点ID和距离添加到节点DataFrame中。

将边DataFrame与节点DataFrame合并，以获取边与最近收费站节点的对应关系。将合并后的数据与收费站数据进行合并，获取边与最近收费站节点的详细信息。

按照边ID进行排序，以便更好地组织数据。

# 高速公路出行OD推测

## 模块功能介绍

该模块的主要功能是通过高速网络拓扑建模之后生成的路网的拓扑数据，与MR热力数据进行叠加，通过优化模型对每个OD对（O、D为高速公路出入口）之间的流量进行推测与验证，进而推断OD分布的概率。同时考虑到了不同时段，不同出行距离和出行意愿的影响，并根据相应的数据对新能源汽车的小时级OD分布概率进行了修正。

整体流程输入：

表 3高速公路出行OD推测输入

|  |  |  |
| --- | --- | --- |
| **输入文件（Input）** | **字段（Fields）** | **字段解释** |
| **mr\_edge\_count\_opt.csv** | edge\_id | 高速公路路段编号 |
| MR\_count\_median | MR数据与高速公路叠加后得到的路段热力数据的中值 |
| date | MR数据对应的日期 |
| **od\_dis\_table\_opt.csv** | station\_id\_x | 作为O点收费站的ID |
| station\_id\_y | 作为D点收费站的ID |
| edge\_id | 收费站之间最短路径经过的所有路段的编号 |
| **edge\_to\_charge.csv** | edge\_id | 建设有充电站，需要进行充电判断的路段编号 |

整体流程输出：

表 4 高速公路出行OD推测输出

|  |  |  |
| --- | --- | --- |
| **输出（Output）** | **字段（Fields）** | **字段解释** |
| **OD\_flow\_{date}.csv** | O | 作为O点收费站的ID |
| D | 作为D点收费站的ID |
| count/prop | 根据OD推测得到的收费站之间的流量/概率 |
| date | 对应日期 |

## OD推测算法1：优化模型

大规模的矩阵求解对于内存和cpu都有较高的要求，并且通常需要极大的的时间成本，直接求解困难较大。因此，该OD推测算法的核心思路是将大规模的OD-edge矩阵拆分成若干个小矩阵，并利用最小二乘和约束最小二乘法进行求解，运用到的数据主要是表征每个OD对（O、D为高速公路出入口）需要经过的路径的OD-edge矩阵，以及表征每条路段上车流量的的MR热力数据的中值MR\_count\_median。

具体的步骤包括：

（1）对MR热力数据提取到高速公路路段上，得到每条路段的的流量。

（2）在优化模型中输入OD—edge的稀疏矩阵和路段edge的流量表。

（3）将矩阵进行分割，然后计算分割后数据的比例，删除数据矩阵和流量数据中的无效信息。

（4）利用带有正则项的最小二乘法进行求解，迭代删除OD对和edge，直到满足相对误差阈值sigma或OD数小于edge数。

（5）利用第一次最小二乘法处理后的数据，使用约束最小二乘法进行求解，得到每对OD之间的流量。

### MR热力流量提取

在路网拓扑建模的基础上，利用geopandas库将每个路段与MR热力数据的网格在空间上叠加，得到每个路段上对应的多个热力值，对所有的热力值求中位数来代表每个路段上的车流量。具体步骤如下：

1、读入MR热力数据和高速公路路网的拓扑数据并进行预处理，利用geopandas库将栅格的经纬度坐标转化为点要素信息，以便进行后续分析统计。

2、将点要素与高速公路面数据在空间上进行叠加，分别统计每条edge上的点的数量MR\_count\_grids，MR的总和MR\_count\_sum以及中值MR\_count\_median。

3、读入之前处理得到的记录OD对途径路径的文件od\_dis\_table.csv，筛选出长度大于1000m的路径并记录编号，根据编号对之前得到路段edge的热力数据进行筛选，去除不必要的路段数据，导出具有大于1000m的路段的信息的文件od\_dis\_table\_opt.csv和MR\_edge\_count\_opt.csv。

### 优化模型构建

在data\_processing.py程序中，程序首先实现数据的预处理工作，输入数据文件为od\_dis\_table\_opt.csv和mr\_edge\_count\_opt.csv。由于部分路段在某些天可能存在缺失MR热力值的问题，因此首先对od\_dis\_table文件和mr\_edge\_count文件中的路径edge\_id求交集，并删除存在缺失值的路段。进一步根据OD及其经过的edge构建矩阵，矩阵每行代表一条路段edge，矩阵每列代表一个OD对，进一步利用 库的 sparse函数，将该矩阵转化为稀疏矩阵，并保存为 .npz。将筛选后的mr\_edge\_count按照edge\_id进行排序，并导出保存为 .csv。

表 5 优化模型基础设置

|  |
| --- |
| **# 程序说明**  # 运行此脚本的文件夹结构如下：  # ----/scaling\_factor\_estimation  # |----/scripts  # | |----main\_scaling\_factor.py <-- 本文件  # |----bw\_data\_matrix.npz <-- 来自 preprocessing.py 的结果  # |----scenario\_table.csv <-- 来自 preprocessing.py 的结果  # |----/results |
| **#数据输入**  # 程序输入 1：OD\_edge\_matrix.npz 是一个具有 n 行和 r 列的稀疏矩阵。其中，n 表示高速公路收费站OD对的数量，r表示高速公路路段的数量。  # 程序输入 2：edge\_flow.csv 是一个具有 r 行和 1 列的数组。  # 程序输出：'ODresults\_{train\_line}\_{start\_day}\_{start\_hour}\_{start\_minute}.csv' 包含所有 OD对的流量数据，排序方式与OD\_edge\_matrix.npz矩阵相同。 |
| **# 参数设置**  # 设置容差参数sigma。如果相对误差小于 'sigma'，则算法终止。  # sigma = 0.1 # 浮点数 sigma >= 0：当 sigma 增加时，程序运行时间减少  # num\_edge\_delete = 5 # 整数 num\_edge\_delete >= 2：当 num\_edge\_delete 增加时，程序运行时间增加  # num\_od\_delete = 5 # 整数 num\_od\_delete >= 1：当 num\_od\_delete 增加时，程序运行时间增加  # max\_iter\_num = 100 # 整数 max\_iter\_num >= 5：当 max\_iter\_num 增加时，程序运行时间增加，并提高精度 |

在main\_FlowEstimation程序中，主要实现了利用最小二乘法求解OD流量。具体的步骤如下：

1. 导入所需的库和模块，包括：time、pandas、numpy、scipy、argparse等。
2. 定义一个FlowEstimator类，该类包含了一些参数和方法。首先在类的初始化方法中，设置了一些默认参数，包括sigma（相对误差的阈值）、num\_scenario\_delete（要删除的场景数）、num\_user\_delete（要删除的用户数）和max\_iter\_num（约束最小二乘法的最大迭代次数）。
3. FlowEstimator类的fitting方法是主要的计算逻辑。它接受两个输入参数：matrix\_data和edge\_data,分别读取输入的数据文件OD\_data\_matrix.npz和edge\_flow.csv，这些数据分别代表OD和edge的稀疏矩阵数据和每个路段的车流量数据。同时该方法还包括三个默认参数division\_base（分割后矩阵的尺寸）、train\_line（选择需要训练的数据，这里是日期），mode（决定是否要保存输出结果，当等于0时则输出结果）。
4. 对数据进行预处理，包括加载稀疏矩阵和路段的车流量数据，提取相关数据并对其进行处理：

表 6 数据预处理算法流程

|  |
| --- |
| 根据train\_line参数选择路段的车流量数据需要处理的数据edge\_ |
| 转换稀疏矩阵的数据类型并对其进行转置，得到处理后的矩阵od\_matrix\_ |
| 计算od\_matrix\_中非零元素的和od\_sum\_total。 |
| 根据给定的division\_base参数将矩阵od\_matrix\_切分成若干小矩阵，计算迭代次数 |
| **对于每次迭代：** |
| 根据division\_base切分od\_matrix\_，计算切分后的od\_matrix的比例od\_ratio。 |
| 根据od\_ratio计算edge\_的比例edge |
| **删除s\_matrix中信息较少的OD和路径**： |
| 计算每列的和od\_matrix\_sum\_index，并找到大于0.5的列索引mid1\_index |
| 根据mid1\_index筛选出s\_matrix的子矩阵od\_matrix\_p\_resolve1 |
| 计算od\_matrix\_p\_resolve1的每行的和，并找到大于0.5的行索引mid2\_index |
| 根据mid2\_index筛选出od\_matrix\_p\_resolve1的子矩阵od\_matrix\_p\_resolve2 |
| 根据mid2\_index筛选出edge的子矩阵edge\_p\_resolve2 |
| 打印被删除的OD和edge的信息 |

1. 根据Tikhonov正则化方法来对数据进行处理，去除不必要的OD和edge。具体是将矩阵正则化后使用一般最小二乘法进行计算，并将预测的结果与真实结果进行对比，计算其绝对平均误差。按照信息量从少到多的顺序来对OD和路径进行删除，直到满足绝对平均误差小于给定的相对误差sigma或OD数小于路段数，如图 7。

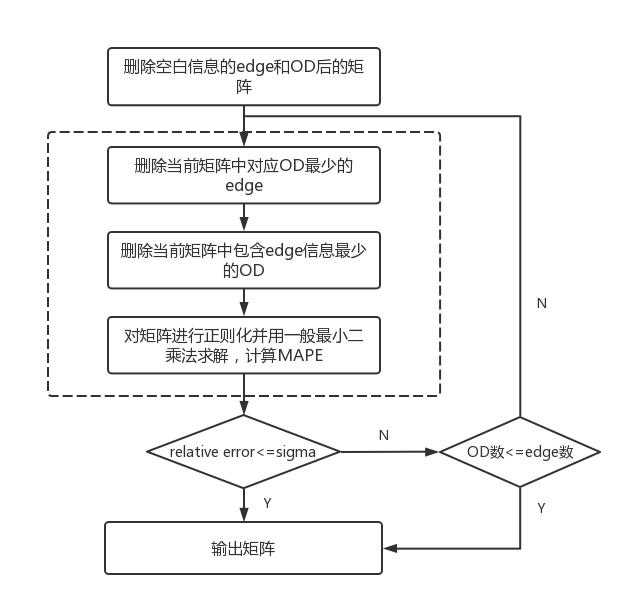


图 7 优化模型算法流程

1. 将满足求解条件的数据利用scipy库的线性求解库中的scipy.optimize.lsq\_linear命令来实现约束最小二乘法的求解，设置每个OD对的流量的上下限和模型的迭代次数max\_iter\_num,以保证模型收敛并且每个OD对的流量数据在合理的范围内，求解得到满足约束的每个OD对的流量计算结果。
2. 导出计算结果到文件，文件中的结果数值的排列顺序与输入的稀疏矩阵中的排列顺序一致：ODresult\_{train\_line}{division\_base}{start\_day}{start\_hour}{start\_minute}.csv。
3. 使用argparse模块解析命令行参数，并根据参数运行FlowEstimator类的fitting方法。
4. 打印程序执行完成的提示信息,输出程序运行的时间。

最后通过match.py程序对输出的结果数据进行处理，利用pandas库的merge命令为结果添加O、D以及date信息，输出包含OD和日期信息的结果文件。

## OD推测算法2：百度迁徙

该算法主要基于高速公路出入口的MR热力数据和广东省内的地级市之间的百度迁徙数据，通过出入口的MR热力数据计算出每两个城市之间各高速公路出入口的权重，并将地级市之间的人口流动的规模指数分配到每个OD上，计算出每个OD的分布概率。该算法较为简单，计算效率高，但受限于百度迁徙数据，结果仅考虑了广东省省内的出行，相较于现实数据可能存在较大的出入，主要用于描述和表征总体的迁移的趋势。

### 百度迁徙数据获取

收费站出行OD推断主要依据地级市之间的百度迁徙数据，结合MR热力数据来进行计算。

1、首先需要将广东省各个高速公路的出入口分配到地级市。利用python的Geopandas库，通过将高速公路出入口点的坐标与各市的面域数据叠加，得到并统计各个出入口所在的地级市信息。依据高速公路的出入口数据生成所有可能的OD对，并将每个OD对的OD对应到地级市层级。

2、基于高速公路的出入口数据，所有可能的OD（Origin-Destination）对将被生成。每个OD对将映射到地级市层级，以建立高速公路交通流动的起点和终点。同时为了确定每个OD对之间的流量，华为云瞰数据中的MR热力数据将被叠加在各高速公路出入口上，得到出口O\_count和入口D\_count的热力数据，并根据这些热力数据来计算每个高速公路入口O和高速公路出口D的概率prop，将其相乘可以得到每个OD的分布概率。

3、爬取广东省各市之间的百度迁徙数据，利用百度迁徙数据的结果将可以得到地级市之间的人流移动的规模指数。将这些指数与各市之间对应的OD的权重相乘，计算出每个OD对所对应的人流移动的规模指数，通过将每个OD之间的规模指数除以所有OD规模指数的总和，可以计算得出每个OD对的分布概率。

4、爬取周边省市到广东省内的百度迁徙数据，利用跨省百度迁徙数据对出行OD进行了修正，具体来说，为了模拟省界附近的道路流量及服务区充电情况，在省界附近道路的北部端点两侧创建了收费站出入口，以模拟跨省流量，并根据缓冲区结果确定站点其所属的省份。进一步利用百度迁徙数据的迁徙指数，计算从各省份到广东省的流量与广东省内流量的比值，并根据收费站所属的省，分别计算以这些收费站为O/D的出行路径流量占省内总流量的比例P，最后计算扩样系数，并修正跨省流量。

## OD结果调整

### 天级到小时级调整（韶关小时级数据）

为了更具体和详细地模拟用户的出行行为，并与真实数据进行对比，可以根据不同时段用户选择出行的概率，将天级的OD出行分布概率调整到小时级别。我们依据韶关市的小时级数据，具体的操作步骤如下：

1. 基于韶关市的小时级数据，计算多天数据的均值，并计算每小时的MR热力值占单日内的热力值的比例。利用得到一天内各小时内用户选择出行的分布概率。
2. 利用得到的单日内每小时的概率分布数据，将其与OD推断得到的单日的分布概率进行相乘，得到每天内每小时OD分布的概率。

### 出行距离调整

由于电动车和燃油车在最大出行距离等方面存在显著差异，特别是在长途出行时，燃油车的比例可能更高。因此，不同出行距离可能会对新能源汽车在总出行车辆中的占比产生影响，应该依据不同的出行距离对新能源汽车的OD分布概率进行修正。当前计划按照每50公里为一个区间，将出行距离划分成多个区段，统计每个区段内新能源汽车占总车辆数的比例，以生成一份出行距离与新能源汽车占比之间的对应数值表。（具体数据尚未确定，之后会进行调整）

在此基础上，可以根据每个OD对之间的最短路径来确定其对应的出行距离，并将其对应到相应的出行距离范围。即将小时级的新能源汽车的OD概率分布，乘以根据出行距离确定的新能源汽车占比，以获得通过出行距离修正后的新能源汽车OD分布概率。

### 新能源车辆占比修正

由于基于MR和百度迁徙数据推测出的车流量及OD实际上是全部类型车的车流量，而新能源汽车可能只占其中的一部分。同时不同城市的新能源车的保有量及其在所有车辆中的占比不同，因此根据不同城市的新能源车保有量的占比对出行分布概率进行了修正。

# 车辆时空轨迹与行为仿真

## ABM仿真技术介绍

Agent-Based Modeling (ABM) 是一种用于模拟和研究复杂系统的计算模型技术。在 ABM 中，系统中的个体被建模为独立的代理(agent)，每个代理有自己的状态、行为规则和与其他代理的交互方式。通过模拟代理之间的相互作用和行为规则，ABM 可以用于研究整个系统的宏观行为。

ABM仿真模型中的关键概念包括代理、状态、行为规则、环境和相互作用。代理是系统中的个体，具有自己的状态和行为规则。状态表示代理在特定时间点的属性或特征，而行为规则定义了代理如何根据当前状态和环境信息做出决策或执行动作。环境是代理交互和行为的背景，可以包括其他代理、物理空间和资源分布等。代理之间通过相互作用影响彼此的状态和行为。通过考虑这些概念，ABM仿真模型可以模拟和分析复杂系统的行为和动态变化，为各个领域的研究提供帮助，例如交通规划、城市规划和生态系统研究等。

ABM仿真过程通常包括初始化、更新代理状态、代理交互和时间推进等步骤。首先，初始化阶段设置仿真环境和代理的初始状态。然后，根据代理的当前状态和环境信息，应用行为规则来更新代理的状态。如果代理之间可以相互作用，执行代理之间的交互过程。接下来，将仿真时间推进到下一个时间步，更新环境和代理的状态。重复执行更新代理状态、代理交互和时间推进的步骤，直到达到预定的仿真结束条件。通过这些步骤的循环迭代，ABM仿真模型可以模拟系统的动态演化和行为变化。

ABM 可以在各种领域中应用，例如交通流模拟、城市规划、经济学、生态学等。在车辆时空轨迹和行为仿真中，ABM 可以用于模拟和研究车辆的行驶轨迹、交通流量、交通拥堵、驾驶行为等方面的问题。通过构建每辆车作为一个代理，定义其状态（如位置、速度、目的地等）、行为规则（如遵循交通规则、路径选择策略等）和与其他车辆的交互方式（如避让、协同行驶等），可以模拟并观察整个道路网络中车辆的运行情况和交通流动态。

ABM 的优势在于它能够考虑个体之间的异质性、非线性交互和新兴的群体行为。它可以帮助理解和预测复杂系统的整体行为，并为政策制定和规划提供支持和决策依据。然而，ABM 也面临一些挑战，包括数据收集和验证、计算复杂性、参数设置和灵敏性分析等。在进行 ABM 仿真时，需要合理选择和校准模型参数，并进行敏感性分析以评估模型的稳定性和可靠性。总的来说，ABM 是一种强大的模拟技术，可以用于研究车辆时空轨迹和行为，并为交通规划、交通管理和智能交通系统等领域提供决策支持。它能够考虑到个体的异质性和交互作用，从而更好地模拟真实世界中的复杂系统行为。

## ABM仿真代理模型

本研究所构建的ABM模型包含两类代理：汽车代理和充电站代理。汽车代理具有唯一的ID、起点和终点等自身属性，同时记录行驶状态。充电站代理负责维护正在充电和等待充电的汽车队列。模型使用step作为时间单位，表示模拟的每个时间步，通过更新代理状态、执行行动和决策来模拟充电站的运行和汽车的行驶过程。这样的模型可用于充电站的仿真模拟和性能评估，为优化电动车充电系统提供帮助。

### 车辆代理

当车辆进行行驶时，每个车辆代理根据给定的车速和耗电参数，在事先提取好的路网上进行移动。车辆代理记录并跟踪自身的当前位置和电量等信息，以便进行有效的管理和监控。行驶行为的实现涉及几个关键步骤。首先，车辆代理根据预先定义的路网拓扑结构，在每个步骤中选择下一个节点进行移动，从而保证行驶路径的合理性。其次，根据给定的车速参数，车辆代理能够计算出每个步骤中车辆移动的距离和消耗的电量，从而为行驶过程提供准确的能源需求。此外，车辆代理会记录自身的当前位置、电量等信息，并根据行驶状态进行及时的状态更新，以便进行实时监测和调度。

充电作为行驶过程中重要的环节之一，也需要得到适当的处理。车辆代理需要根据一定的规则和条件寻找充电站，并进行充电操作。为此，车辆代理会根据当前位置和预设的寻找充电站距离，搜索附近的充电站，并使用一定的策略选择最近的充电站作为充电目标。进一步，车辆代理会根据当前电量和充电需求，判断是否需要进行充电，并根据预设的充电速度参数计算所需的充电电量和充电时长。通过这样的行驶和充电机制，车辆代理能够在路网上高效地行驶，并及时满足充电需求，从而实现可靠的运行和管理。

以下是车辆初始化的示例，如图 8：

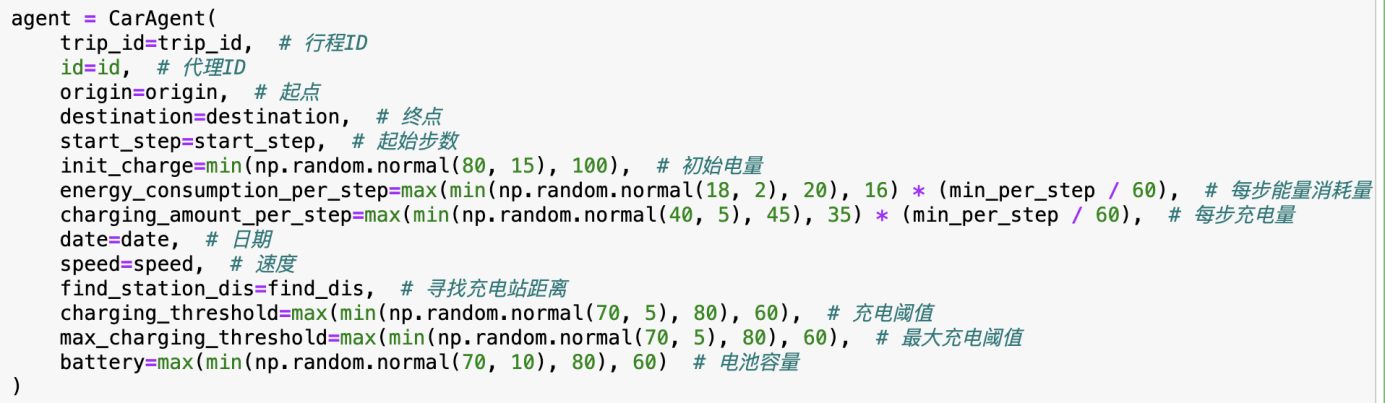


图 8 车辆初始化代码示例

下面是对`CarAgent`类和`StationAgent`类中各个函数的说明：

CarAgent

1. `run(self)`: 当线程开始运行时，该函数被执行。它表示车辆代理的主要逻辑。如果当前节点不是目的节点，车辆代理会更新状态，移动到下一个节点，并可能在附近的充电站充电。如果当前节点是目的节点，车辆代理会被标记为待删除。

2. `charging\_per\_step(self, station\_agent)`: 根据每一步的充电策略，该函数用于在每一步中给车辆充电。如果车辆的电量低于阈值并且处于行驶状态，它会增加电量，并相应地更新充电站的充电需求。

3. `append\_car\_charging\_df(self, station\_id)`: 该函数用于将车辆的充电记录追加到车辆的充电数据帧中。它记录了车辆在特定充电站充电的相关信息。

4. `create\_car\_dataframe(self)`: 该函数用于创建当前车辆状态的数据帧。它收集了车辆的行驶信息，包括当前节点、电量、行驶距离等。

5. `find\_nearby\_charging\_stations(self)`: 返回满足距离要求的当前边附近的充电站。它根据给定的条件筛选出符合要求的充电站，并返回相应的数据。

6. `charge\_at\_nearest\_station(self, nearby\_charging\_stations)`: 在附近充电站中找到距离最近的充电站，并进行充电操作。该函数更新车辆的当前节点，计算需要充电的电量，并调用充电站的相关方法进行充电。

7. `check\_charging\_station(self, mode)`: 该函数用于判断是否需要充电。如果车辆处于行驶状态且当前节点不是目的节点，它会根据给定的模式检查附近是否有可用的充电站，并决定是否进行充电。

8. `get\_charging\_probability(self)`: 根据车辆的电量情况，计算车辆在当前状态下进行充电的概率。该函数根据车辆的电量调整基础概率，并返回调整后的充电概率。

### 充电站代理

充电站代理负责为车辆提供充电服务，并管理车辆的充电需求和状态。充电服务是充电站代理的核心职责之一。充电站代理根据车辆的充电需求和充电速度参数，为车辆提供相应的充电能力。通过考虑充电桩的功率和车辆的充电需求，充电站代理能够计算出充电时长并进行充电操作。在某些情况下，充电站代理可能面临车辆排队等待的情况。当充电站的充电能力有限时，充电站代理需要管理车辆的排队等待状态。为此，在添加车辆到队列时，充电站代理会检查车辆是否已经在等待队列或正在充电队列中，以避免重复添加。若当前充电队列有空余位置，充电站代理会将等待队列中的车辆移到充电队列中，并将其充电状态标记为非行驶状态。通过这样的充电站代理操作，车辆能够有序地进行充电，并且充电站能够合理地管理充电排队等待的情况，以提供高效和可靠的充电服务。

StationAgen

1. `\_\_init\_\_(self, charge\_node\_id, max\_capacity)`: 初始化充电站代理对象。设置充电站的节点ID、最大容量、剩余容量、等待队列、当前队列、充电需求、车辆数量和数据帧等属性。

2. `if\_not\_exit\_in\_queues(self, car\_agent)`: 检查车辆代理是否不存在于当前队列和等待队列中。返回布尔值，表示车辆是否不存在于队列中。

3. `append\_car\_charging\_df(self, car)`: 将车辆的充电记录追加到充电站的充电数据帧中。创建一个包含车辆充电信息的字典，并将其转换为数据帧格式返回。

4. `append\_to\_stationDf(self)`: 将当前充电站的信息追加到充电站数据帧中。创建一个包含充电站相关信息的数据帧，并将其与之前的数据帧合并。

5. `has\_capacity(self)`: 检查当前充电站是否有剩余容量。返回布尔值，表示当前充电站是否还有空余位置。

6. `charge\_cars(self, car\_agent)`: 对当前队列中的车辆进行充电。如果车辆在当前队列中，调用车辆代理的`charging\_per\_step`方法进行充电。

7. `add\_to\_queue(self, car\_agent)`: 将车辆代理添加到充电站的队列中。根据车辆代理是否存在于队列中和充电站是否有空余位置，将车辆添加到当前队列或等待队列，并更新相应的数据帧和车辆数量。

8. `move\_cars\_from\_waiting\_to\_current(self)`: 将等待队列中的车辆移动到当前队列。当当前队列有空余位置且等待队列不为空时，从等待队列中取出车辆并添加到当前队列中，并更新相应的数据帧和车辆数量。

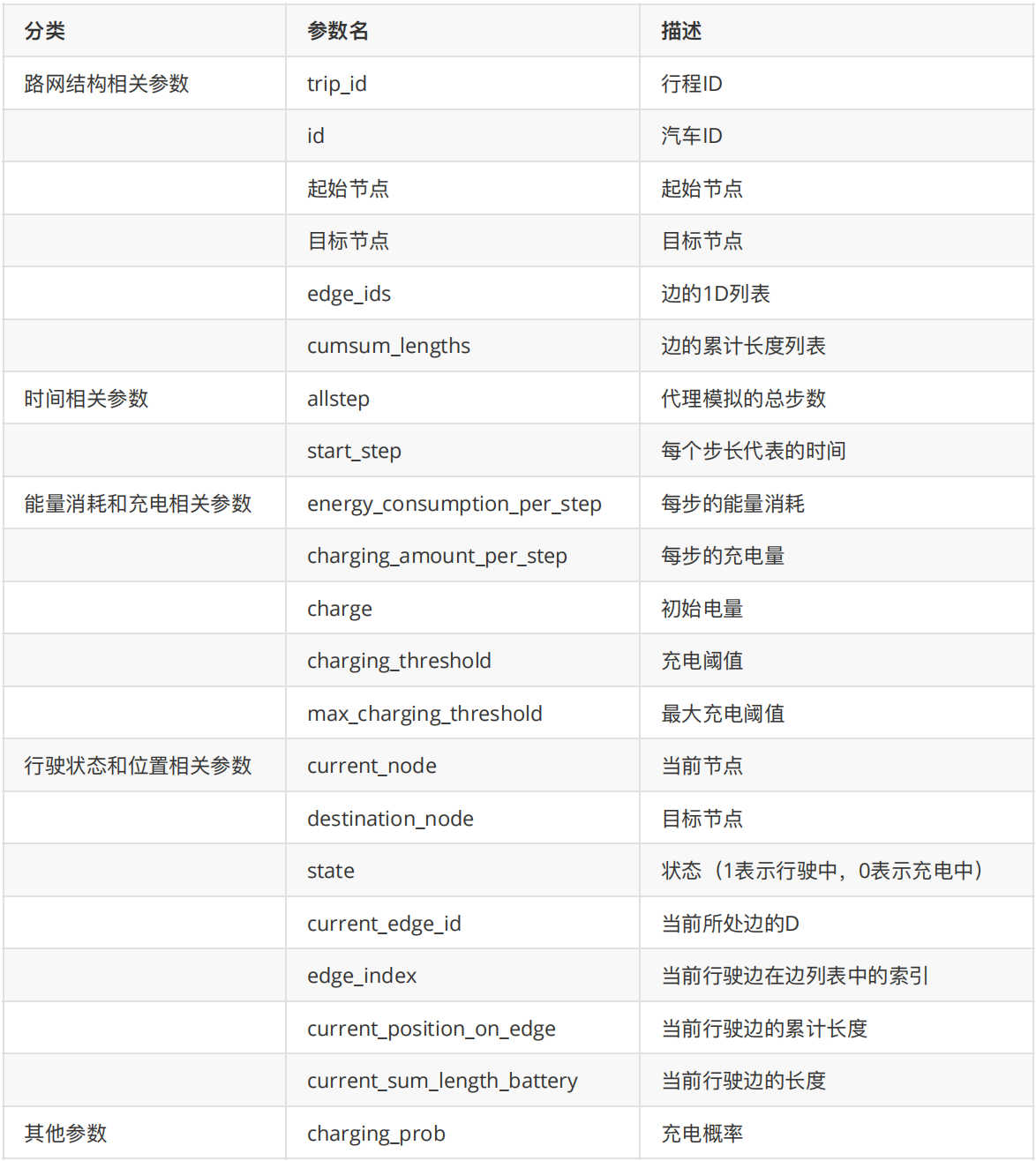
9. `station\_run(self, car\_agent)`: 充电站的执行逻辑封装。该函数接受一个car\_agent参数，表示要处理的车辆代理。首先，将车辆代理的状态设置为非行驶状态（state=0）。然后，调用add\_to\_queue方法将车辆代理添加到充电站的队列中。接下来，调用charge\_cars方法对当前队列中的车辆进行充电操作。如果充电站有空余位置，会从等待队列中将车辆移动到当前队列中，直到当前队列达到最大容量或等待队列为空。

## 仿真模型参数与场景设置

### 仿真模型参数

ABM模型的可调整参数如表 7所示

表 7 仿真模型参数



### 场景设置

在建立充电站仿真模型时，还需要考虑以下几个方面。对于高速场景下的路网结构比较单一的情况，可以选择使用最短路径算法来确定车辆的行驶路径。这意味着车辆在仿真中选择的路径将是最短的路线。充电逻辑的决策可以基于电量作为考量因素。例如，可以使用以电量为均值的正态分布作为充电结束的阈值。当车辆的电量达到或超过阈值时，仿真模型可以将其安排去充电站进行充电。在确定车辆数量和分布时，可以采用一天一万辆车左右的数量。为了更真实地模拟车辆的分布情况，可以根据求解得到的广东省内起点和终点的车流量数据，计算概率分布，并进行采样来确定车辆的起点和目的地。充电站的设置可以基于真实的充电站情况进行。这包括充电车位数和充电站的位置。可以根据实际充电站的信息，确定充电站的数量和位置分布，以便在仿真中准确地模拟充电站的布局和可用资源。通过综合考虑路网结构、充电逻辑、车辆数量和分布以及充电站的设置，可以建立一个更真实和可靠的充电站仿真模型，以便进行相关研究和分析。

### 模型优势

根据高速公路场景的特点，与城市场景相比，充电决策更为简化。在高速场景中，充电决策主要是为了确保车辆能够到达目的地，而不受工作或其他需求的影响。因此，模型将充电决策仅与当前电量以及最近的充电站之间的距离相关联，从而简化了逻辑。

这种简化的逻辑有助于提高模型的运行效率，并使其更加适应高速行驶的要求。车辆需要快速做出充电决策以保证行程的顺利进行，而复杂的决策逻辑可能导致计算和操作的延迟，从而影响到车辆的性能和可靠性。

简化后的ABM仿真模型更加注重于刻画每个时间步的车辆和充电站信息，并从原始数据中提取更多有用的信息。例如，可以获取**上下游路段**的车辆信息以及**高位、低位**充电车辆的比例等。这些数据为充电站选址提供了数据支持。通过考虑上下游路段的车辆信息，能够更准确地了解充电站周围的交通状况和需求情况。这为确定充电站的最佳位置提供了重要的参考依据，以满足车辆充电需求并确保充电站的可达性和利用率；通过分析高、低位充电车辆占比等信息。这些数据能够揭示充电需求的特点和趋势，例如高位充电需求更高的区域或时间段。基于这些信息，能够更精确地确定充电站的数量、位置和配置，以最大程度地满足高速场景下的充电需求，并优化充电站的利用率和效益。

## 仿真的结果评价

仿真模型的结果评价将与真实的充电订单数据进行对比，比较的指标为各充电站在各小时的订单数量。在比较仿真结果之前，需要进行归一化处理，仿真模型所反映的是现实场景的趋势，由于车辆规模设置的差异，仿真结果和实际订单数据可能具有不同的量纲或单位，归一化通过消除这些量纲差异，也确保所有数据在相同的尺度上，避免了某些较大值的数据对结果产生不成比例的影响。

具体来说，归一化的方法如下：

为归一化后数值，为需进行归一化的仿真数据及真实数据。

选取平均绝对百分误差 (mean absolute percentage error, MAPE)评价精度，它计算每个时间段的平均绝对误差百分比。由于单位缩放为百分比，因此广泛用于预测误差。当数据中没有异常值时，效果较好。MAPE损失是通过将所有误差标准化为百分比来计算的，同时也可以避免正数抵消负数的问题。

为了比较仿真结果效果，选用MAPE指标进行计算，衡量仿真结果与真实结果间的差异，并选用订单数这一指标进行衡量，订单数能够较好地反映充电桩充电需求，并从站点小时级订单数、站点总订单数、城市级订单数、高速公路级订单数等多维度地进行精度的比较，评估仿真结果，衡量仿真结果的趋势与真实结果趋势的一致性，计算公式如下：

计算站点小时级订单量时， 代表计算的站点小时级总样本数， 代表第个站点的小时级订单数真实结果 , 代表仿真结果中，第个站点的仿真结果，使用的数据均为归一化后数据。

计算站点总订单量时， 代表计算的站点总样本数， 代表第个站点的总订单数真实结果 , 代表仿真结果中，第个站点的仿真结果的总订单数，使用的数据均为归一化后数据。

计算城市级仿真精度时，代表计算的城市的总样本数， 代表第个城市的总订单数真实结果 , 代表仿真结果中，第个城市的仿真结果的总订单数，使用的数据均为归一化后数据。

计算高速公路仿真精度时， 代表计算的高速公路的总样本数， 代表第个高速公路的总订单数真实结果 , 代表仿真结果中，第个高速公路的仿真结果的总订单数，使用的数据均为归一化后数据。

（1）小时级订单量

在时间分布方面，选取2023年1月27日与2023年1月16日为典型日，对比仿真结果和真实结果的小时级总订单量曲线时，发现二者呈现出较高的一致性，如图 11。尤其在夜间，充电订单数量相对较少，而在12点至15点之间出现了充电高峰。表明仿真结果与现实场景拟合较好。仿真结果的夜间的充电需求较低，但在白天特别是中午到下午，出现了显著的充电高峰期，与真实结果十分相似。小时级充电订单总数平均绝对百分比误差（MAPE）为7.93%和12.65%

图片包含 图示

描述已自动生成 图表

描述已自动生成

图 11 小时级订单量对比

（2）站点总订单量空间分布

对比一天内仿真结果和真实结果的站点级总订单量空间分布时，发现总体的空间分布趋势较为一致，如图 12。在粤西地区，例如云浮、肇庆和阳江等地的服务区，充电订单数量较为集中。相对而言，在粤东地区，河源至潮汕方向的服务区充电订单数量较多。而大湾区外围区域，充电订单数量相对较少。经归一化后，站点小时级充电订单量平均绝对百分比误差（MAPE）为13.64%和13.32%，对应预测精度为86.36%和86.68%

图表, 散点图

描述已自动生成

图 12 站点总订单空间分布对比

（3）站点总订单量拟合程度

对比站点级的总订单量仿真结果与真实结果，绘制拟合曲线，如图 13。部分站点的真实订单数量相对较小，仿真结果的订单数也较小，说明模型对这些站点的仿真较为准确。然而，也可能存在一些站点，其仿真订单数量低于真实数据。可能由于仿真所使用的车辆数与真实情况存在较大的量级差异，导致仿真的这些站点的订单量较少，这些站点在散点图的上方。

图表, 散点图

描述已自动生成图表, 散点图

描述已自动生成

图 13 站点总订单拟合程度

（4）订单充电量分布

对比仿真与真实数据的电量分布和充电时长呈现出总体相似的趋势，如图 14，其中订单电量均值集中在20-30 kWh之间，充电时长主要分布在20-30分钟的区间。这种相似性表明仿真模型在模拟电动车充电过程中的整体趋势方面表现良好。

图表, 直方图

描述已自动生成

图 14 订单充电量、时长分布对比

（5）城市、高速公路级仿真精度

分析城市级、高速公路级仿真精度，如图 15。部分城市如肇庆、云浮、清远等城市精度较高、汕头、茂名等城市精度较低。分析道路级别的仿真进度，汕湛高速、乐广高速的精度较高，揭惠高速、兰海高速的精度较低。归一化后，大部分地级市小时级订单精度在85%以上（MAPE小于15%）。高速公路小时级订单精度在85%左右，部分充电站数量较少、省界附近的地级市与高速误差较大。

图表, 直方图

描述已自动生成

图 15 城市、高速公路级仿真精度

（6）部分站点小时级订单量

选取部分站点小时级仿真结果与真实订单进行对比，如图 16，仿真结果的小时级订单总体趋势与真实订单较为相较，夜间订单数量较少，高峰小时的订单量趋势也较为一致。在特定的时间段内或特定的站点上，实际结果和仿真结果可能会有一些细微的差异，但总体上能够看出仿真结果与实际情况之间的相似性。

图表

描述已自动生成

图 16 部分站点小时级订单对比

# 服务区充电需求预测评价与场站配置建议

## 上游路段MR与充电车辆数比较结果

将站点上游路段MR与站点小时级充电车辆数进行对比分析，站点小时级充电车辆数和站点上游路段的MR显示出一定的相关性，如图 9。通过分析不同日期的数据，均呈现出一定的相关性，相关系数达到0.6。小时级内较多的MR数据可能意味着较多的车流量，车流量较大时，充电需求也将较大，MR能够一定程度反映高速道路运行状况，基于MR数据推测出行OD具有可行性。同时表明，考虑充电站选址及扩容时，可考虑能反映车流量、上游路段MR等的指标。

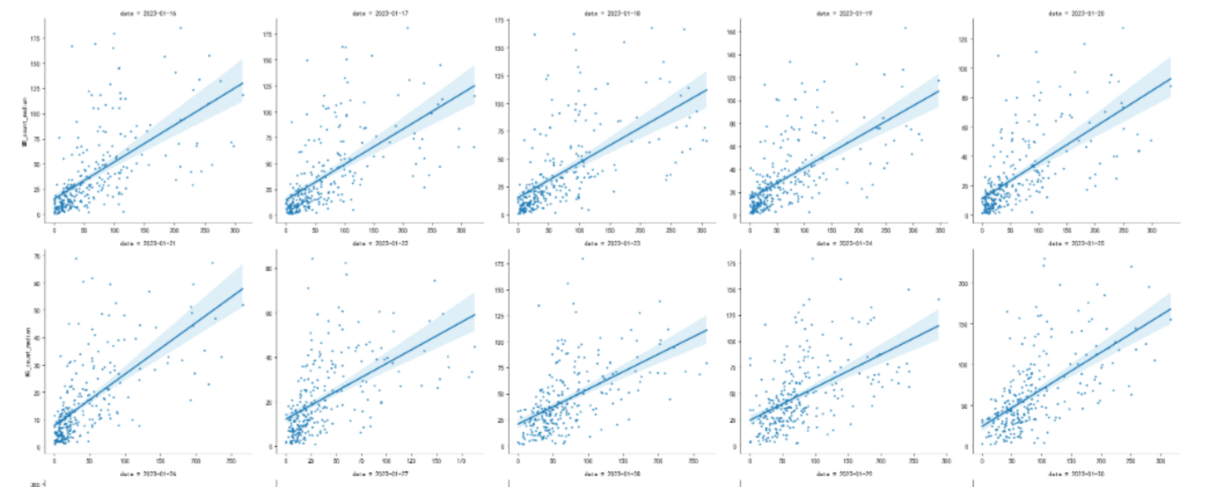


图 9 上游路段MR与充电车辆数相关程度

分析不同上游路段范围内（100米至5000米）的MR与站点小时级充电订单的相关性，发现不同距离范围对小时级充电订单的相关性影响明显，如图 10。总体而言，相关系数保持在0.58至0.61之间。200米范围内的上游路段与小时级充电订单的相关性最为显著，相关系数达到0.61。表明距离较近的路段的MR对充电需求的影响更加显著。随着距离范围的增加，相关系数逐渐减少，意味着随着距离的增加，上游路段MR对充电订单的影响逐渐减弱。在使用MR数据时，可考虑上游路段200米范围的MR作为分析指标。

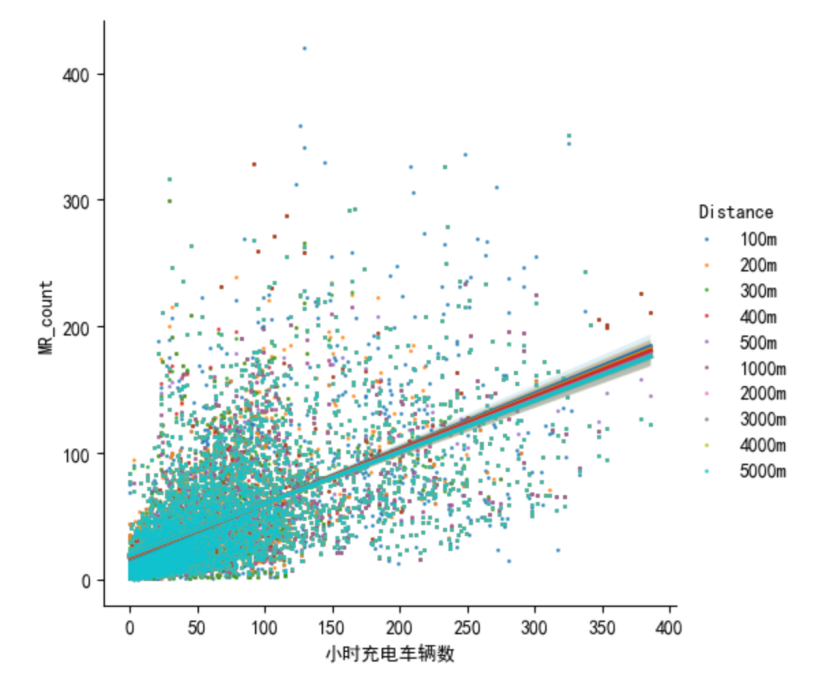


图 10 不同上游路段范围MR与小时级充电订单相关性

## 选址参考指标

1、站点整体特征：

（1）充电站订单量： 站点一天内小时级的充电订单量和一天的订单总量。这个指标揭示了站点的充电需求，利用仿真结果数据和真实订单数据进行分析，可以识别时间上的周期性的需求变化。选址时，高订单量的站点可能需要更多的充电桩来满足需求。

（2）充电需求： 站点一天的总充电需求（kwh），为该站点一天订单的充电量总和。该指标将评估站点的整体充电需求量，有助于确定站点的承载能力。对于选址，需要考虑站点是否有足够的能力满足现有的和潜在的充电需求。

（3）充电桩需求缺口：仿真充电需求与真实充电需求间差值。衡量实际充电需求与当前充电桩数量之间的差距，一方面可以确定建设充电桩的紧迫性，另一方面可以指导充电桩布设数量。

（4）充电桩利用率： 充电桩一天内的使用率，为充电桩使用时间占一天24小时时间的比值。分析充电桩的实际利用情况有助于了解是否存在充电桩利用不足或过度的情况。确定是否需要重新调整资源配置，以更好地满足充电需求。

2、车辆个体特征：

（1）进入充电站时电量分布： 分析车辆进入充电站时的电量（kwh）分布。反映了不同电量需求的车辆数量及车辆充电需求的分布特征，以确定适当的充电桩类型。

（2）服务区充电时长： 车辆在服务区充电的平均时长。评估车辆在充电站的停留时间。

（3）服务区排队时长：车辆在充电站的平均排队时长。可以确定充电桩是否需要进行扩容，并对充电桩布设前后排队时长进行分析以评估选址方案。

（4）订单的充电电量分布：站点级的充电订单充电量分布。

## 选址原则

1. 交通流量和车辆密度： 选择人流密集或车流量大的区域，比如城际关键走廊、主要高速公路、交叉口等，以满足更多车辆的充电需求。

2. 充电需求：选址应在在充电需求热点区域，比如特定的服务区域、高车流量、人流量的服务区、加油站等区域。

3.停车场设施：充电桩在停车场设施内的部署更为便利，比如服务区、收费站、停车场等区域。

4. 便利设施周边环境：确保充电桩周边有便利设施，如便利店、餐厅、公共卫生间等，提供用户更好的充电体验。

5. 供电设施和技术支持： 选择在电力供应充足、网络连接良好的区域，同时需要有良好的技术支持以确保充电桩的运行。

6. 场地和用地政策：确保选址符合土地政策和法规，避免后期出现法律问题。

7. 用户可达性：选址需考虑用户的可达性，确保对车辆驾驶员来说是方便和易达的。

8. 充电桩覆盖范围：充电桩的分布需要均匀，以确保服务区域内的全面覆盖，而不仅限于某个地区。

## 充电站总体需求分布趋势

粤西云浮、肇庆和阳江地区部分服务区充电需求高，这些服务区显示出较高且集中的充电需求，尤其是在粤西地区，如云浮、肇庆和阳江等区域。这表明在这些城市群的服务区内，充电需求集中并较为显著。

粤东河源至潮汕地区部分服务区充电需求高，相对而言，河源至潮汕方向的服务区，展示出充电需求较大，可能与该地区的特定交通或城市发展模式相关。

粤北韶关区域，乐广高速走廊服务区充电需求高，在广州至韶关方向的乐广高速走廊，部分服务区呈现出相对较高的充电需求。这表明在该走廊的特定服务区内，充电需求相对较为显著。

汕湛高速走廊充电需求相对较低，汕湛高速东西走廊方向的服务区显示出相对较低的充电需求。这表明这些区域的出行量较低，车流量稀少，相应地充电设施需求较为有限，如图 17。

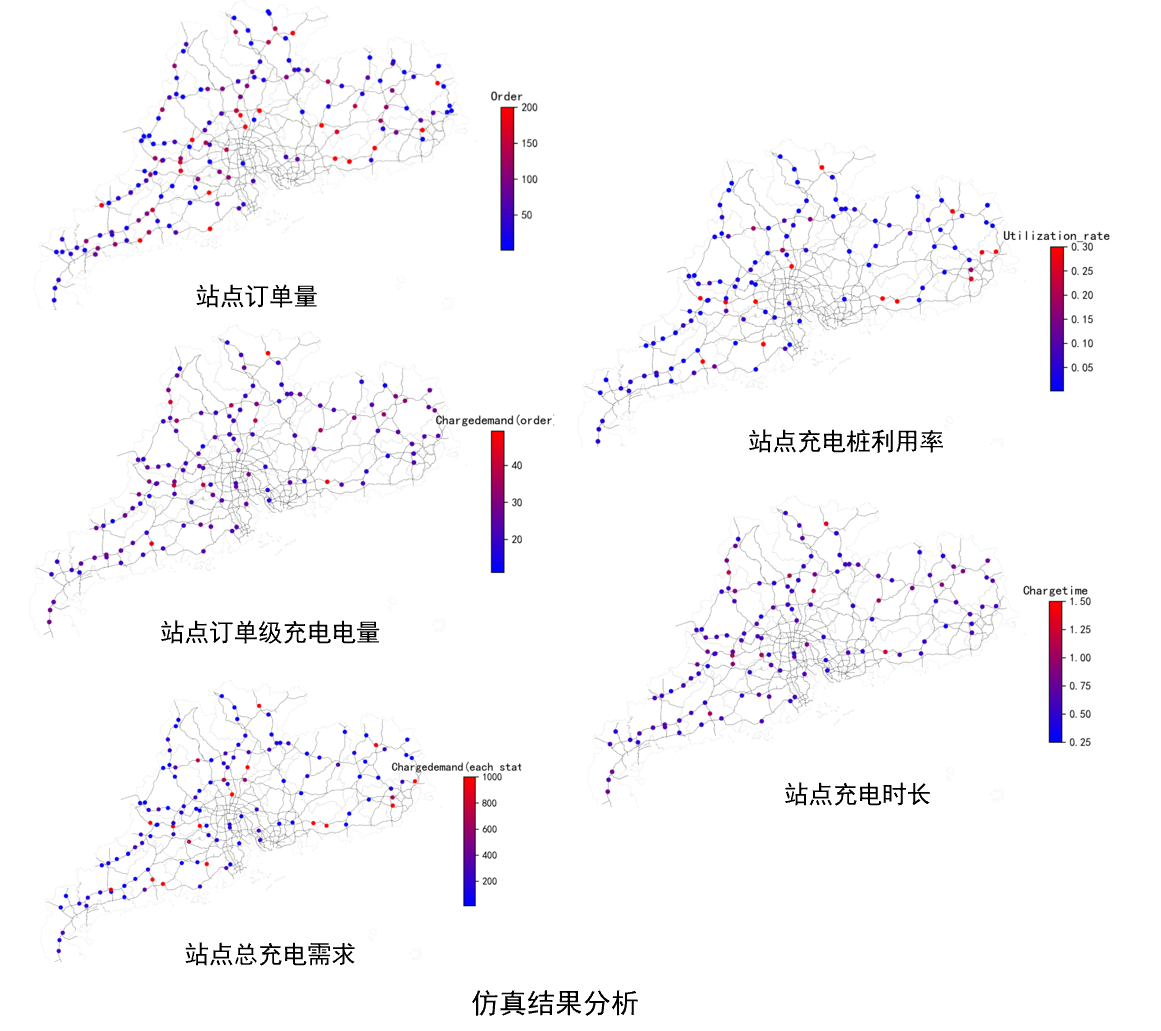


图 17 站点需求空间分布趋势

## 充电站选址及扩容初步方案

考虑站点整体特征，充电订单量、充电总量、充电桩需求缺口、充电桩利用率，车辆个体特征，进入充电站时电量分布、在服务区充电+排队时长、订单的充电电量分布等指标，综合分析充电桩充电需求，确定潜力较高、选址及扩容优先级较高的站点为初步选址方案，推荐阳茂高速新虚东行，阳茂高速新虚西行，沈海高速梁金山北行，河莞惠高速蓝塘南行，揭博高速石坝西行，潮惠高速陆河东行等服务区进行选址及扩容，如图 18，该类型站点的充电需求较高，充电桩利用率较高，表明仍有一定的充电需求未被满足，可在这些站点布设充电桩，以更好满足电动车充电需求。

表 8 初步方案部分指标

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **服务区** | **订单数（仿真）** | **充电需求（仿真）** | **利用率（仿真）** | **订单数** | **充电需求** | **利用率** |
| 阳茂高速新虚东行 | 81 | 1590 | 0.25 | 211 | 6453 | 0.9 |
| 阳茂高速新虚西行 | 128 | 4420 | 0.9 | 119 | 3010 | 0.74 |
| 沈海高速梁金山北行 | 238 | 8544 | 0.9 | 301 | 7877 | 0.89 |
| 河莞惠高速蓝塘南行 | 387 | 12588 | 0.9 | 386 | 9034 | 0.81 |
| 揭博高速石坝西行 | 291 | 9066 | 0.6 | 341 | 7716 | 0.76 |
| 潮惠高速陆河东行 | 101 | 1685 | 0.27 | 187 | 4548 | 0.38 |

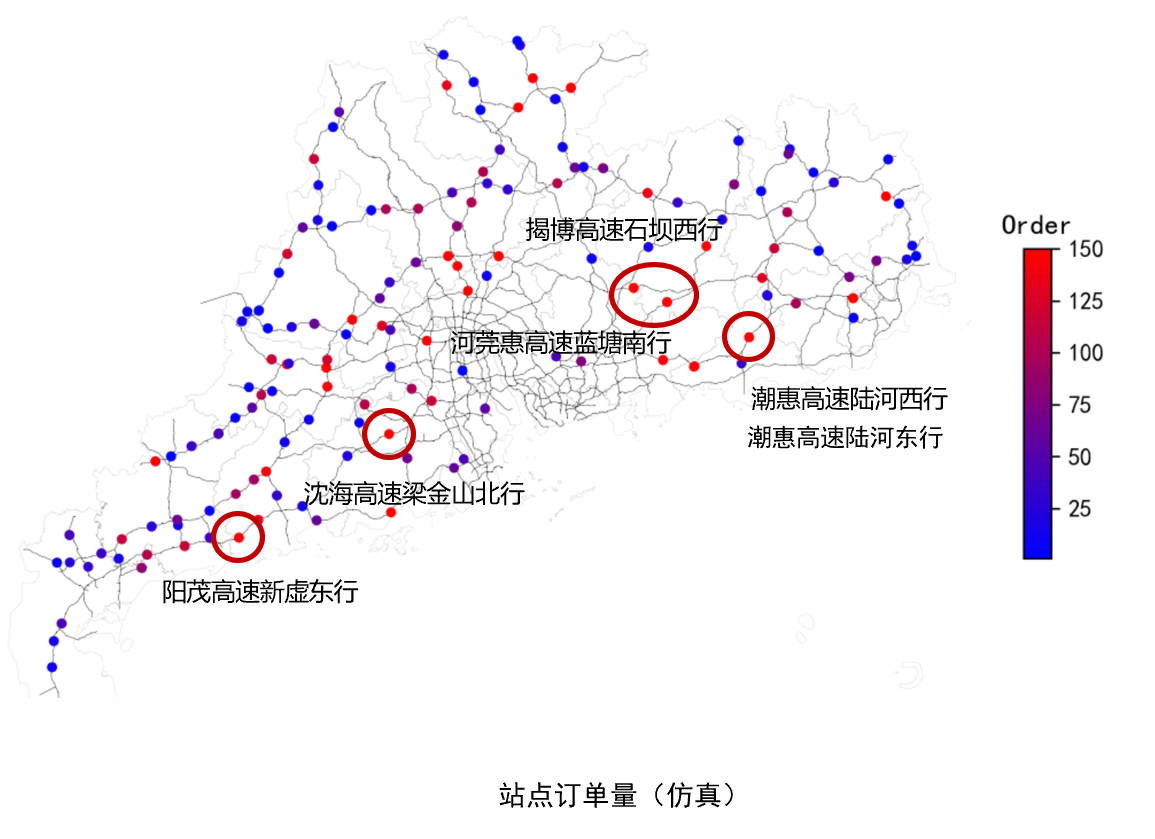


图 18 初步推荐选址及扩容方案

# 总结

针对充电场站选址面临决策数据不足，难以精准刻画不同群体间的差异化充电需求等现实问题，阶段一通过融合云瞰电信大数据和高速公路服务区的充电订单数据，形成了高速公路场景下的出行OD推测-新能源车辆行为仿真-充电桩选址及评估为一体的技术路线，精细刻画高速场景充电需求，为充电场站的选址运维搭建“城市一张网”提供重要支撑。

高速场景新能源车充电需求刻画共分为四个模块，包括拓扑网络建模、OD推测、基于Agent-based的电动车行为仿真及充电桩评估及选址建议。高速公路拓扑网络建模中，通过地理信息处理、近邻匹配、识别最短路径等方法，对高速公路网和服务区进行简化，建立了网络模型，获取了高速公路OD间出行路径。OD推测部分，一方面使用MR及路网数据，利用最优化算法对OD进行推测，另一方面是使用百度迁徙数据结合韶关小时级MR数据推断出小时级出行OD。基于OD推测结果，建立了基于Agent-based的电动车充电行为仿真，通过车辆代理和站点代理实现对高速公路场景下的电动车行为、轨迹仿真。最后，基于仿真结果，考虑站点综合指标，提出充电桩配置及扩容建议。

# 附录：代码函数说明

|  |  |  |  |
| --- | --- | --- | --- |
| Highway\_model模块函数说明 | | | |
| 函数 | 说明 | 参数 | 输出 |
| reconstruct\_highway | 提取简化高速公路边信息并构建相关网络 | highway\_centerline (GeoDataFrame): 包含高速公路中心线的地理空间数据框（GeoDataFrame），其中应至少包含以下列： - 'geometry': 包含道路中心线几何形状的列 station (GeoDataFrame): 包含收费站信息的地理空间数据框（GeoDataFrame），其中应至少包含以下列： - 'geometry': 包含收费站位置几何形状的列 - 'id': 收费站的唯一标识 | edge (GeoDataFrame): 包含生成的道路网络边的地理空间数据框（GeoDataFrame），其中包括以下列：  - 'geometry': 包含道路边界几何形状的列 - 'edge\_id': 道路边的唯一标识 - 'attr': 道路属性（默认为 'road'）  node (DataFrame): 包含生成的道路网络节点的数据框（DataFrame），其中包括以下列：  - 'lon': 节点经度 - 'lat': 节点纬度 - 'geometry': 包含节点位置几何形状的列 - 'id': 节点的唯一标识  station2node (DataFrame): 包含收费站节点与网络节点的对应关系的数据框（DataFrame），其中包括以下列： - 'station\_id': 收费站的唯一标识 - 'node\_id': 对应的网络节点的唯一标识 |
| get\_matched\_edge | 将收费站匹配到最近的道路边并调整网络 | edge (GeoDataFrame): 包含道路边界的地理空间数据框（GeoDataFrame），其中应至少包含以下列： - 'geometry': 包含道路边界几何形状的列 - 'edge\_id': 道路边的唯一标识 station (GeoDataFrame): 包含收费站信息的地理空间数据框（GeoDataFrame），其中应至少包含以下列： - 'geometry': 包含收费站位置几何形状的列 - 'id': 收费站的唯一标识 | edge (GeoDataFrame): 更新后的道路边界的地理空间数据框（GeoDataFrame），包含匹配到的收费站和额外的节点信息。 matched\_table\_2 (DataFrame): 包含未匹配到的收费站信息的数据框（DataFrame），其中包括以下列  - 'edge\_id': 道路边的唯一标识  - 'id': 未匹配到的收费站的唯一标识 |
| generate\_plane | 高速公路边向右平移一定距离形成面 | edge (GeoDataFrame): 包含高速公路边界的地理空间数据框（GeoDataFrame），其中应至少包含以下列： - 'geometry': 包含道路几何形状的列 - 'crs': 坐标参考系统信息，应为 'EPSG:4326'。  dist (float): 高速公路边界向右平移的距离，默认为 100 米。 | edge\_plane (GeoDataFrame): 包含生成的平移后道路面的地理空间数据框（GeoDataFrame），其中包括以下列： - 'geometry': 包含生成的道路面几何形状的列 - 'crs': 坐标参考系统信息，为 'EPSG:4326'。 |
| get\_od\_path | 获取OD路径和长度信息的函数。 | - edge: 包含路段信息的DataFrame，包括 'u'（起点节点ID）、'v'（终点节点ID）、'length'（路段长度）、'edge\_id'（路段ID）等列。 - node: 包含节点信息的DataFrame，至少包括 'id'（节点ID）列。 - station2node\_dict: 字典，将站点ID映射到节点ID的关系。 - gdtollnode: 包含站点信息的DataFrame，至少包括 'id' 列。 | - od\_dis\_table: 包含每个OD对应的路段信息的DataFrame，包括 'station\_id\_x'（起点站点ID）、'station\_id\_y'（终点站点ID）、'edge\_id'（路段ID）、'cumsumlength'（经过路段的累计长度）等列。 - od\_length: 包含每个OD对应的路径总长度的DataFrame，包括 'station\_id\_x'（起点站点ID）、'station\_id\_y'（终点站点ID）、'length'（路径总长度）等列。 |
| od\_merge\_table\_reconstruct | 重新构造OD合并表格的函数。 | od\_dis\_table: 包含每个OD对应的路段信息的DataFrame，至少包括 'station\_id\_x'（起点站点ID）、'station\_id\_y'（终点站点ID）、'edge\_id'（路段ID）、'cumsumlength'（经过路段的累计长度）等列。 | od\_merge\_table: 重新构造的OD合并表格的DataFrame，包括 'station\_id\_x'（起点站点ID）、'station\_id\_y'（终点站点ID）、'edge\_id'（路段ID列表）、'cumsumlength'（经过路段的累计长度差值列表）、'cumsumlength2'（经过路段的原始累计长度列表）等列。 |
| get\_edge\_to\_charge | 查找最近充电站点信息的函数。 | - edge: 包含路段信息的DataFrame，至少包括 'u'（起点节点ID）、'v'（终点节点ID）、'length'（路段长度）、'edge\_id'（路段ID）等列。 - node: 包含节点信息的DataFrame，至少包括 'id'（节点ID）列。 - charge\_station: 包含充电站点信息的DataFrame，至少包括 '场站名称'、'充电车位数'、'node\_id' 等列。 | - edge\_to\_charge: 包含路段到最近充电站点的关联信息的DataFrame，包括 'edge\_id'（路段ID）、'charge\_node\_id'（最近充电站点ID）、'charge\_node\_distance'（距离最近充电站点的距离）、'场站名称'、'充电车位数'等列。 |

|  |  |  |  |
| --- | --- | --- | --- |
| centerline模块函数说明 | | | |
| 函数 | 说明 | 参数 | 输出 |
| get\_centerline | 根据提供的道路数据生成中心线。该函数执行以下步骤：  1. 对道路数据进行缓冲区分析。  2. 计算缓冲区的泰森多边形（Voronoi图）。  3. 从泰森多边形中提取位于道路缓冲区内的中心线段。  4. 合并这些线段，并清除不符合条件的死路。  5. 对结果中的中心线进行简化。 | - road (Shapely.geometry): 道路数据的几何对象。  - interpolation\_distance (float): 缓冲区分析和细分顶点时的距离参数。  - minlength (float): 被视为死路的最小长度阈值，短于此长度的孤立线段将被清除。  - simplify\_tolerance (float): 几何简化时的公差值，默认为0.0001。 | - centerline\_merged (GeoDataFrame): 包含合并后的中心线的GeoDataFrame，坐标系与输入的道路数据相同。 |
| get\_start\_end\_point | 提取道路中心线的起点和终点坐标，并为每条中心线增加起终点的x和y坐标字段。 | - centerline\_merged (GeoDataFrame): 包含道路中心线几何数据的GeoDataFrame。 | - centerline\_merged (GeoDataFrame): 经过处理，新增了起点和终点坐标字段的GeoDataFrame。 |
| clean\_endlane | 清除孤立且长度不足的道路末端，以维护道路网络的连通性。此函数的目标是剔除那些只连接一个节点且长度小于指定阈值的道路末端。 | - centerline\_merged (GeoDataFrame): 含有几何对象的GeoDataFrame，表示待处理的道路中心线。  - minlength (float): 用于判定道路末端是否足够长而应被保留的长度阈值。 | - centerline\_merged (GeoDataFrame): 清除了孤立且短的道路末端后的中心线GeoDataFrame。 |
| clean\_duplicate\_paths | 清除重复的路径段，以生成更加简洁的中心线。此函数的目标是识别并移除长度低于特定阈值的重复线段，  保留较长的线段，并合并其余的路径以形成连续的中心线。 | - centerline\_merged (GeoDataFrame): 含有几何对象的GeoDataFrame，表示当前所有中心线段。  - minlength (float): 用于识别短路径的长度阈值，小于或等于此长度的重复路径会被移除。 | centerline\_merged (GeoDataFrame): 清除了重复路径之后的中心线GeoDataFrame。 |
| get\_spanning\_tree | 构建道路中心线的最小生成树，忽略长度小于等于指定阈值的边。 | - centerline\_merged (GeoDataFrame): 包含道路中心线几何数据的GeoDataFrame。  - minlength (float): 长度阈值，小于等于此阈值的边将不会被包含在生成的最小生成树中。 | - centerline\_merged (GeoDataFrame): 构建完成的最小生成树，其中包含了所有满足长度条件的中心线。 |

|  |  |  |  |
| --- | --- | --- | --- |
| FlowEstimator模块函数说明 | | | |
| 函数 | 说明 | 参数 | 输出 |
| FlowEstimator | 估算OD流量和生成结果。 | 属性： - `sigma`：误差限制，默认为0.1，需要根据实际更改 - `num\_od\_delete`：要删除的OD对数量，默认为5，需要根据实际更改 - `num\_edge\_delete`：要删除的路段数量，默认为5，需要根据实际更改 - `max\_iter\_num`：最大迭代次数，默认为100，需要根据实际更改  输入参数： - matrix\_data：OD矩阵数据的文件路径，默认为'OD\_edge\_matrix.npz'，需要根据实际更改 - edge\_data：边缘流量数据的文件路径，默认为'edge\_flow.csv'，需要根据实际更改 - division\_base：分割基数，默认为10000，需要根据实际更改 - train\_line：培训日期，需要根据实际更改 - mode：模式，如果为1，则不会保存结果；如果为0，则会保存结果，默认为0。 | 如果mode为0，则将流量推断结果导出到CSV文件。 |
| get\_matrixdata | 对需要处理数据求取edge\_id的交集，生成OD推断所需的矩阵数据文件 | od\_dis\_table: 包含OD和edge对应信息的数据表 mr\_edge\_counts: 包含路段流量信息的数据表 edge\_to\_charge: 包含路段充电设施信息的数据表 | 生成OD——edge的稀疏矩阵和记录edge流量的数据表。 |
| result\_match | 根据指定日期范围内的数据，处理和匹配不同数据源的信息，生成特定格式的输出文件。 | - date: 研究的日期 - OD\_pairs: 流量推断的OD对 | 生成多个CSV文件,每个文件包含匹配后的数据。 |

|  |  |  |  |
| --- | --- | --- | --- |
| Abm模块函数说明 | | | |
| 函数 | 说明 | 参数 | 输出 |
| initialize\_agent | 初始化代理信息并生成代理数据 | od\_prob (DataFrame): 包含OD概率信息的数据框，必须包含 'O'（起始站点）、'D'（目标站点）、'count'（代理计数）列。  travel\_hour\_prob (DataFrame): 包含旅行小时概率信息的数据框，必须包含 'hour'（小时）和 'prob'（概率）列。  od\_cnt (int): 生成的代理数量。  current\_date (str): 当前日期。  min\_per\_step (int): 每步代表的分钟数。 | DataFrame: 包含代理信息的数据框，包括 'trip\_id'、'ids'、'O'、'D'、'date' 和 'start\_step' 列。 |
| get\_agent\_path\_table | 根据代理数据和新的OD距离信息创建代理路径表格 | agent\_data (DataFrame): 包含代理信息的数据框，必须包含 'trip\_id'、'ids'、'O'（起始站点）、'D'（目标站点）、'date' 和 'start\_step' 列。  new\_od\_dis (DataFrame): 包含新的OD距离信息的数据框，必须包含 'station\_id\_x'、'station\_id\_y'、'edge\_id' 和 'cumsumlength' 列。 | DataFrame: 包含代理路径信息的数据框，包括 'trip\_id'、'ids'、'O'、'D'、'date'、'start\_step'、'path'（代理路径）和 'cumsumlength'（路径累计长度）列。 |
| get\_agent\_path\_network | 获取代理路径网络信息 | agent\_data (DataFrame): 包含代理信息的数据框，必须包含 'O'（起始站点）和 'D'（目标站点）列。  edge (DataFrame): 包含边信息的数据框，必须包含 'u'（起始节点）、'v'（结束节点）和 'length'（边长度）列。  node (DataFrame): 包含节点信息的数据框，必须包含 'id'（节点ID）列。  station2node (DataFrame): 包含站点到节点映射信息的数据框，必须包含 'station\_id'（站点ID）和 'node\_id'（节点ID）列。 | DataFrame: 包含代理路径信息的数据框，增加了 'path'（代理路径）和 'cumsumlength'（路径累计长度）列。 |
| simulation | 模拟仿真函数，用于模拟车辆在高速公路上的运行和充电行为。 | agent\_data: DataFrame，包含了车辆行程的初始数据，如起始点、终点、路径和出发时间等。 edge\_to\_charge: DataFrame，包含道路和充电站的对应关系以及充电站的充电位数。 min\_per\_step: int，表示仿真中每个时间步长代表的实际分钟数。 cnt\_step: int，表示仿真将运行的总步数。 | agents: list，包含所有模拟车辆代理的列表。  station\_agents: list，包含所有充电站代理的列表。 |
| save\_cars | 将多个代理（agents）中的车辆数据合并为一个DataFrame，并设置适当的索引。 | agents: 一个代理列表，每个代理持有车辆的运行数据和充电数据 | all\_car: 包含所有车辆运行数据的DataFrame，'step'列作为索引被重置  all\_car\_df: 包含所有车辆充电数据的DataFrame，未设置索引 |
| save\_stations | 此函数用于处理充电站代理的数据，将每个代理的车辆充电数据和充电站数据合并，计算并转换时间戳。 | station\_agents: 充电站代理的列表，每个代理包含有关充电站和车辆充电情况的数据。  min\_per\_step: 模拟中每步表示的实际分钟数。  start\_date: 模拟的起始日期字符串，格式为 'YYYY-MM-DD'。 | car\_charging\_df: 包含所有充电站车辆充电数据的DataFrame。 all\_station\_df: 包含所有充电站数据的DataFrame，并计算了基于模拟步骤的时间戳。 |
| save\_car\_vis | 为车辆数据添加真实步骤计数，时间戳，并通过插值计算车辆的经纬度位置。 | all\_car: DataFrame, 包含车辆的运行数据。  edge: DataFrame, 包含道路网络边的数据，用于插值计算位置。min\_per\_step: 每个模拟步骤对应的分钟数。 start\_date: 模拟的起始日期，格式为 'YYYY-MM-DD'。 | sorted\_df: 经过处理的DataFrame, 包含车辆ID, 行程ID, 边ID, 当前电量，经纬度和时间戳，按车辆ID和时间排序。 |
| save\_station\_vis | 根据每一时刻的数据，计算充电站的车辆统计信息，并创建对应的时间戳，用于可视化充电站状态。 | stations\_step\_day: DataFrame, 包含每个时间步骤内充电站的状态信息。 stations\_node: DataFrame, 包含充电站的节点信息，例如位置坐标。 min\_per\_step: int, 每个模拟步骤对应的分钟数。 start\_date: str, 模拟的起始日期，格式为 'YYYY-MM-DD'。 | DataFrame, 包含充电节点ID、经纬度、时间、充电需求、当前车辆数和等待车辆数。 |
| cars\_to\_order | 根据车辆的充电状态变化生成等待、充电和放电的订单。 | cars: DataFrame，包含车辆ID、行程ID、当前电量、日期和步骤等信息。 edge\_to\_charge: DataFrame，包含道路边缘ID与充电站节点ID的对应关系以及充电站名称。 | wait\_orders: DataFrame，包含车辆在不同时间段电量不变时的等待订单。 charge\_orders: DataFrame，包含车辆在不同时间段电量增加时的充电订单。 discharge\_orders: DataFrame，包含车辆在不同时间段电量减少时的放电订单。 |
| cal\_station\_demand | 计算充电站每个时间步长的车辆数和充电需求的变化量。 | station\_output: DataFrame，包含充电站的输出信息，具体包括每个时间步长的车辆总数和充电需求。  cnt\_step: int，表示一天内的时间步长数量。 min\_per\_step: int，每个步长代表的分钟数。 | stations\_demand: DataFrame，包含每个时间步长的车辆数和充电需求的变化量，以及对应的小时和分钟数。 |
| plot\_station\_demand | 根据每小时的车辆数和充电需求绘制充电站需求变化曲线图。 | stations\_demand: DataFrame，包含每个时间步长的车辆数和充电需求变化量，以及对应的小时数。  output: str，指定输出图像文件的保存路径。 | 该函数会生成一张图像，图中展示了一天中不同小时充电需求和车辆数的变化情况，并将此图像保存在指定的路径。 |
| mape | 这个函数计算给定实际值和模拟值的MAPE，它用于衡量模拟值与实际值之间的差异。 | real (实际值), simulation (模拟值) | 返回计算得到的MAPE值（百分比），保留两位小数。 |
| mapefunc | 这个函数用于计算按小时分组的实际值和模拟值的MAPE。 | df (包含 'hour', 'real\_num', 'simulation\_num' 列的DataFrame) | MAPE (平均绝对百分比误差) 值，保留两位小数 |
| plot\_lines | 这个函数用于绘制两个曲线，其中一个使用第一个坐标轴，另一个使用第二个坐标轴。 | ordercomparehour (包含 'hour', col1, col2 列的DataFrame), col1 (要绘制的第一个曲线的列名), col2 (要绘制的第二个曲线的列名) | 返回,绘制两个曲线并显示图形。 |

|  |  |  |  |
| --- | --- | --- | --- |
| plot\_stations模块函数说明 | | | |
| 函数 | 说明 | 参数 | 输出 |
| get\_centerline | 绘制充电站点分布图的函数。 | - data: 包含充电站点信息的DataFrame，至少包括 'lon'（经度）、'lat'（纬度）和指定的列 'column'（用于颜色映射的数据）等列。 - column: 用于颜色映射的列名，默认为 'num'。 - vmin: 颜色映射的最小值，默认为 0。 - vmax: 颜色映射的最大值，默认为 50。 - cax\_title: 颜色映射图例的标题，默认为 'Charge\_value'。 - cmap: 颜色映射的颜色图，默认为 'BuPu'。 - save: 可选，如果提供了文件路径，则将图形保存到指定路径，否则显示图形。 | 充电站点分布图 |