

摘要

本文主要研究电动汽车充电桩的数量与选址问题。电动汽车凭借其节能经济正在逐渐成为当今时代汽车市场重要的一部分，与之伴随而来的是人们对充电桩的需求。合理的充电桩数量与布局能够起到最大的经济效益并且便利人们。

针对问题一，将充电桩分成快慢充两类，利用了 ArcGIS 软件对杭州道路进行仿真得到拓扑结构，以路口为节点将道路转化为无向赋权图。以初始距离矩阵通过弗洛伊德算法求两个节点之间的最短路径，得到 **OD 距离成本矩阵**。我们使用了带承载力限制的 **P-中值模型**确定目标优化函数，以充电桩数量和各点的承载力限制作为约束条件建立整数线性规划模型。使用**对偶单纯形法**利用 GUROBI 求解该数学模型。在快充充电中应用**排队理论**计算排队时间作为额外约束条件，同时考虑成本因素进行**多目标优化**。关于充电桩未来发展的预测，我们通过人口、需求量、财富等因素进行预测，对充电需求量的预测我们转化为了对电动汽车数量增长的预测。考虑到第二问的要求，电动汽车数量增长的预测描述我们写在后面一段中。我们以杭州为例展开我们的算法。

针对问题二，考虑到汽车市场中主要由电动汽车与燃油汽车两个主流车种组成，并且两者数量增长满足 Logistic 规律，我们联想到了**生物种群竞争模型**。该模型中较难得到的指标是发展优势指数 σ ，我们开创性地通过对影响汽车数量的几个主要因素使用**主成分分析法**，得到了以“城市发展程度”为关键词的第一主成分值。接着我们将该主成分值与两种汽车数量分别运用**线性回归**作出拟合，通过得到的拟合函数计算出各自的发展优势指数 σ 。最后使用 MATLAB 软件对该模型进行求解，得到了未来十年电动汽车数量与燃油汽车数量的预测。所得到预测结果基本与主流预测基本接近，且与 2022 年第一季度数量验证后符合。同时根据第一问中对未来五年、十年后的预测结果我们对电动汽车的发展作出了进一步分析。

关键词：**OD 矩阵，P-中值模型，对偶单纯形，排队理论，多目标优化，主成分分析，生物种群竞争模型，线性回归**

一、问题重述

1.1 问题背景

如今，化石燃料和环境保护已成为世界上最热门的问题。世界范围内的主要交通工具以石油为主，造成了严重的环境污染。由于化石燃料的资源有限，而且总是导致污染，因此向更清洁的能源过渡是不可避免的。为了实现这一目标，以新能源汽车为代表的电动汽车被公认为 21 世纪汽车产业转型的主要发展方向。

众所周知，电动汽车效率高、噪音低、污染几乎为零。电动汽车的优势显而易见。因此，它们已在世界各地广泛使用。推动电动汽车的规模化发展，需要完善相应的基础设施。充电站作为电动汽车设施建设的重要组成部分，对整个电动汽车产业的发展至关重要。我们需要在哪里和多少个充电站建设？选择正确的位置和估计充电站的数量非常重要。

随着温室效应和空气污染问题的加剧，各国都在寻找新能源来替代传统燃料，如原油或柴油，以缓解日益严重的空气问题。自混合动力汽车和燃气汽车问世以来，新型清洁汽车的探索仍在不断进行。目前，以特斯拉为首的电动汽车将在更大程度上突破能源和经济的限制，更好地平衡快速增长的汽车需求与环境的关系。充电站数量合适，距离合适，对于电动汽车的普及至关重要。与加油站相比，电动汽车充电站占用的空间更小，具有更高的安全系数，可以更好地分布在街道和社区，让人们更方便、更高效地使用。然而，电动汽车的推广并非一蹴而就。要逐步扩大电动汽车覆盖面，不断完善电动汽车充电站网络，最终完成汽油和柴油汽车的终结运营。另外，不同国家的经济、文化条件不同，需要根据自己的具体情况确定推广时间和推广范围，才能取得更好的效果。要逐步扩大电动汽车覆盖面，不断完善电动汽车充电站网络，最终完成汽油和柴油汽车的。

1.2 具体问题重述

1.2.1 第一问

随着时代的发展，电动汽车正逐渐步入人们的生活，但与此同时充电桩的需求问题逐渐显现。当前时代处于电动汽车发展初期，充电站系统的不完善在一定程度上限制了电动汽车的发展，同时充电桩的分布乱象也对社会资源、交通运输等方面产生了诸多负面影响。问题一要求我们综合考虑多方面因素，对充电站系统建模，开发一个合适的算法解决充电站的数量与布局选址问题。

1.2.2 第二问

电动汽车未来发展的趋势是当下的热点讨论问题，如问题一所言，充电站系统是影响电动汽车整体发展的重要因素。所以在问题一的基础下，要求结合其他模型预测电动汽车未来发展，并结合问题一模型，研究影响电动汽车发展的主要因素。

二、符号说明

表 1: 符号说明

符号	说明
d_{ij}	OD 距离成本矩阵, 表示 i 地到 j 地的距离
q_i	需求列表, 表示 i 地的车辆数量 (即对充电桩的需求量)
p	总充电桩数量
n_i	供应列表, 表示 i 地安装的慢充充电桩数量
ρ_{ij}	分配矩阵, 表示 i 地的需求以 ρ_{ij} 的比例分配给 j 地的慢充充电桩
c_j	容量列表, 表示 i 地能够供应的容量, 等于 i 地建设的慢充充电桩数量
E_k	续航里程
L	年平均行驶里程
F_k	百公里耗电
I	服务强度

三、基本假设

假设 3.1 每个目的地的停车场数量足以供应目的地的车流量 (无需在目的地建造新的停车场)

假设 3.2 慢充不会对电网产生高峰期的负荷, 只需算总耗电。

原因: 慢充总可以智能选择一天当中电网负荷低的时候充电。

假设 3.3 只有电量低于 10% 的电动车会占用慢充车位充电并且错开 (即不会所有车同时电量低于 10%), 所以慢充不考虑排队只考虑总耗电; 同时每天分为白天和晚上两个充电时期。

假设 3.4 所有道路都完全相同，不存在坡度等因素会增加在道路上行驶同等距离时的耗电量。

原因：首先，所有道路导致的耗电量增加可以叠加到道路的距离成本中去。其次，坡度等数据网上没有，需要对实地进行测绘等工作，在短暂的时间内无法完成。

假设 3.5 车辆到达快充充电站接受服务的间隔以及服务时间的分布符合泊松分布。

假设 3.6 所有长途旅行的车辆可以且最多可以在充电站排队等待 1 个小时。

假设 3.7 考虑到数据获取的简便，我们将新能源汽车均归类为电动汽车。

四、模型建立

4.1 充电桩选址模型

4.1.1 充电桩分类

我们将充电站分为两种，分别是短时停靠充电站（简称快充,sc）和长时停靠充点站（简称慢充,lc）

长时停靠充电站用于在人们长时间停车（平均超过 5h，用普通功率，车可以充满），

短时停靠充电站用于在人们长途旅行时临时充电（起到类似于加油站的作用，需在半小时内充满一辆汽车）

4.1.2 建模前的准备：约束条件以及概念

i) 成本

快充成本为地价和充电桩费用的叠加

$$f_{sc} = \sum_i (f_{ci} \cdot S_0 + f_{li}) \cdot n_i \quad (1)$$

慢充建在停车场，价格远低于快充且单价固定，所以对成本的限制转化为对数量的限制

$$f_{lc} \propto p \quad (2)$$

ii) 区域总耗电量 q

汽车的日均总耗电量 q 可由如下公式计算得到：

$$q = \frac{L}{365} \cdot \frac{F_k}{100} \cdot Po \cdot V_p \cdot \alpha_e \quad (3)$$

其中 L 为年平均行驶里程， F_k 为百公里耗电，Po 为地区人口， V_p 为地区人均私家车数量， α_e 为电动车占比，可由电动车发展模型得到。以上数据均可以从互联网中查得。

iii) OD 距离矩阵 \mathbf{d}_{ij}

距离矩阵可以通过道路实际位置和道路拓扑结构通过对每两个节点求最短路径算出。道路数据从互联网上获得。

iv) 电网负荷 \mathbf{Ld}_a

由于中国电网很强大，且现在电动车数量稀少，我们初步模型暂时不考虑这个条件作为约束，但可以根据结果计算对电网造成的负荷。计算方法如下：

慢充考虑一整天的负荷 $Ld_{lc} = \beta q / 24$, β 为损耗系数。除以 24 得到平均每小时负荷

快充考虑高峰时期不间断充电造成的负荷: $Ld_{sc} = P_{sc} = F_k \cdot E_k / t_{sc}$, 其中 E_k 为续航里程, t_{sc} 为快充充满所需时间。以上数据均可以从互联网中查得。

v) 单位承载力 c_0

单个充电桩所能够承载几辆车充电。我们默认 $c_0 = 1$ 。

vi) 单次充电跨越时间 T

依据假设 3.3, 电动车从满电到 10% 电量跨越的半天数

$$T = \left\lfloor 0.9 \frac{E_k}{L/(2 \cdot 365)} \right\rfloor \quad (4)$$

vii) 各节点充电桩数量 n_i

各个节点充电桩数量总和应当等于充电桩总数: $\sum_i n_i = p$

viii) 各节点需求 q_i

每个节点的需求即每个节点在每个半天的所有需要充电（即电量低于 10%）的车的数量

$$q_i = Po_i \cdot V_p \cdot \alpha_e / T \quad (5)$$

ix) 分配矩阵 ρ_{ij}

代表 i 地的需求中 ρ_{ij} 的比例由 j 地满足, 因此有 $\sum_j \rho_{ij} = 1$, 即总体上每个地区的需求量被 100% 地满足

x) 道路流量 \mathbf{F}_i

每条道路的每小时流量。

xi) 排队时间 W_s

指一个顾客在系统中的全部停留时间为期望值, 记为 W_s , 它是等待时间和服务时间的总和 $W_s = W_q + E$ 。

xii) 充电桩总数 p

充电桩所需的总数 p 可以通过总需求加上比例为 β 的冗余求得：

$$p = (1 + \beta) \cdot \sum_i q_i = (1 + \beta) \cdot Po \cdot V_p \cdot \alpha_e / T \quad (6)$$

其中 Po 为当地人口总量。

xiii) 快充站的服务模型

根据假设 3.5，我们选择用 M/M/n 的服务系统模型估计等待时间。[7]

首先，我们根据各个站的充电桩的数量 n_i 、单位时间前来充电的车数期望

$$\lambda_i = \sum_j (\rho_{ji} \cdot F_j) \quad (7)$$

以及单位时间能够充好离开的车数期望 μ 可以求解出每个站服务强度

$$I_i = \frac{\lambda_i}{n_i \mu} = \frac{\sum_j (\rho_{ji} \cdot F_j)}{n_i \mu} \quad (8)$$

因此服务请求队列为 k 的概率，即车辆等待 $k-n$ 辆即可充电的概率：

$$p_k = \begin{cases} p_0 \frac{I^k}{k!}, & 1 \leq k < n \\ p_0 \frac{I^k}{n! n^{k-n}}, & k \geq n \end{cases},$$

其中，

$$p_0 = \left[\sum_{k=0}^{n-1} \frac{I^k}{k!} + \frac{I^n}{n! (1 - \frac{I}{n})} \right]^{-1} \quad (9)$$

由此我们可以得到车辆排队等待的概率 $E_c(I)$ 为：

$$E_c(I) = \sum_{k=n}^{\infty} p_k = \frac{\frac{I^n}{n! (1 - \frac{I}{n})}}{n! (1 - \frac{I}{n})} p_0 \quad (10)$$

最后得到车辆排队等待时间 W_s 为：

$$W_s = \frac{E_c(n, \rho)}{n\mu - \lambda} + \frac{1}{\mu} \quad (11)$$

4.1.3 目标优化模型的建立

对慢充选址的优化： d_{ij} , q_i , p 作为已知量，以 n_j , ρ_{ij} 为变量进行优化，为了使充电站位置便于人们停放，优化目标选为：使所有地区的需求者在停车充电以后所需要步行到达目的地的距离总和最小。因此所需优化的函数为：

$$f(\rho) = \sum_i \sum_j d_{ij} q_j \rho_{ij} \quad (12)$$

因此综合上面的分析可得优化方程为：

$$\min f(\rho) = \sum_i \sum_j d_{ij} q_j \rho_{ij}$$

$$\begin{cases} \sum_i n_i \leq p & (\text{最多有 } p \text{ 个充电桩}) \\ \sum_j \rho_{ij} = 1 & (\text{比例系数和为 } 1) \\ \sum_i q_i \rho_{ij} \leq c_j = n_j \cdot c_0 & (\text{承载力限制}) \end{cases} \quad (13)$$

这是一个整数线性规划模型。

对快充选址的优化：对于一条路上选址的最佳地点在端点 [2,]，因此我们选取道路交叉点作为节点。这样对于快充的选址优化方程框架上与对慢充的优化方程相同，但对成本的约束不能转化为对数量的约束而应成为优化对象：

$$\min \begin{cases} f_{sc} = \sum_i (f_{ci} \cdot S_0 + f_{li}) \cdot n_i \\ f(\rho) = \sum_i \sum_j d_{ij} q_j \rho_{ij} \end{cases}$$

$$\begin{cases} \sum_j \rho_{ij} = 1 & (\text{比例系数和为 } 1) \\ W_{si} \leq W_0 = 1h & (\text{等待时间限制}) \\ dis_{ij} < 0.9E_k & (\text{续航里程限制}) \end{cases} \quad (14)$$

但是，由于快充站的建设成本远高于慢充充电桩，所以快充只用来满足长途旅行的续航问题而不关心短程开车的充电问题（这由慢充来解决）。因此，我们认为城内（地价高）的地方不用设快充充电站，而只需要在城际公路上设置。

城际公路中，高速公路上的快充站设置可以参照服务区的设置（时间先后、地点），因为服务区的设置一般是经过大量计算的。其它道路则可以根据上面的优化方程进行计算。但由于主干道路间隔较大，为了使成本降低，我们只需用快充的捕获半径覆盖全国道路 $n \cdot \frac{\sqrt{3}}{4}R = S \cdot \alpha$ 即可（用正六边形覆盖， $\alpha = 89.26\%$ 为道路覆盖率），而每个快充站的等待时间则与其捕获半径内流量有关。等待时间的限制和流量则规定了每个快充站最少的充电桩数量。

4.2 电动车增长模型

后面我们通过生物种群相互竞争模型来预测未来几年的电动车与燃油车的数量关系。

4.2.1 生物种群相互竞争模型

当某个自然的环境中只有一种生物的群体（种群）生存时，我们常用 Logistic 模型来描述它的数量的演变过程，即：

$$x(t) = rx\left(1 - \frac{x}{N}\right) \quad (15)$$

如果一个自然环境中存在两个种群，它们之间的关系为相互竞争关系，并且数量变化服从 Logistic 规律。

于是我们对其中一种群甲有

$$x_1(t) = r_1 x_1 \left(1 - \frac{x_1}{N_1}\right) \quad (16)$$

这里因子 $(1 - \frac{x_1}{N_1})$ 反映由于甲对有限资源的消耗导致的对其自身增长的阻滞作用， $\frac{x_1}{N_1}$ 可理解为相对于 N_1 而言，数量为 x_1 时供养甲的食物量（设食物总量为单位量 1）。

当考虑到甲乙两个种群在同一自然环境中生存时考虑到乙种群消耗统一资源对甲的增长的影响，我们的因子 $(1 - \frac{x_1}{N_1})$ 再减去一项，该项与种群乙的数量成正比，得到种群甲如下式：

$$x_1(t) = r_1 x_1 \left(1 - \frac{x_1}{N_1} - \sigma_1 \frac{x_2}{N_2}\right) \quad (17)$$

上面 σ_1 表示单位数量的种群乙消耗的供养种群甲的食物量为单位数量甲消耗的供养甲的食物量的 σ_1 倍。 σ_2 则相反同理。

4.2.2 本题模型建立

回到本论文中，我们将电动汽车、燃油汽车与种群甲、乙作如下类比：

$x_1(t)$ 表示电动车在 t 时间的保有量， $x_2(t)$ 表示燃油车在 t 时间的保有量。

r_1 、 r_2 分别表示电动车与燃油车的固有增长率。

N_1 、 N_2 为电动车与燃油车的最大容量。

σ_1 表示电动汽车相对于传统汽车在同等经济发展下的发展优势， σ_2 表示传统汽车对电动汽车同等经济发展下的发展优势。

首先我们假设其数量遵循 Logistic 规律，显然电动汽车与燃油汽车两者之间存在竞争关系，故我们利用生物种群相互竞争模型来预测未来几年的电动车与柴油车的数量关系。

得到如下方程：

$$\begin{cases} x_1(t) = r_1 x_1 \left(1 - \frac{x_1}{N_1} - \sigma_1 \frac{x_2}{N_2}\right) \\ x_2(t) = r_2 x_2 \left(1 - \sigma_2 \frac{x_1}{N_1} - \frac{x_2}{N_2}\right) \end{cases} \quad (18)$$

r_1 与 r_2 我们直接根据近年来两种汽车的增长情况求解得到，其中新能源车近年来固有增长率约为 50%（2022 年中国新能源汽车行业市场前景及投资研究预测报告），燃油车近年来固有增长率约为 4%（杭州统计年鉴 2021）。故取 $r_1 = 0.5, r_2 = 0.04$ 。

N_1 、 N_2 的确定存在不确定性，需要提前估算确定。由于电动汽车与燃油汽车的功能性基本相同，我们这里假设 $N_1 = N_2$ 。对该值的估算方法普遍有三种，分别是直接选

取最大值法、专家判断法和纯粹数学推导法。本研究采用专家判断法确定两种汽车最大保有量，根据文献（《我国到底能承载多少汽车？》——姜靖、戴慧兰），国家信息中心信息资源部主任徐长明认为，汽车保有量最高可达 4.5 亿辆。为了合理地预估我国个人汽车保有量极限，本论文中假设其极限占比为 90%，计算得出我国个人汽车最大保有量约为 4.05 亿辆。

σ_1 与 σ_2 的确定我们开创性地利用主成分分析法来得到主成分数据，以得到的主成分为自变量，汽车数量为因变量通过最小二乘法作线性回归，进而得到 σ_1 与 σ_2 。方法如下。

4.2.3 相关性分析

通过对以往研究电动汽车发展的论文，我们注意到影响我国电动汽车保有量变化的因素有多种，如经济因素，包括人均 GDP、居民收入、经济产业结构、居民消费水平等；社会因素，包括城市人口、城市化率、失业率、拥堵成本等；环境因素，包括公路网规模、基础设施完善度等。考虑长期完整数据的可获得性，我们在这里初步选取了城市人口数、城市化水平、人均 GDP、居民消费水平、公路里程、第一产业占比、第二产业占比、第三产业占比这 8 个代表性的影响因素进行分析。（数据大部分来源于《中国统计年鉴（2021）》[1]）

首先我们将以上八个因素与电动汽车保有量进行了 Spearman 相关系数相关性分析，得到如下相关系数热力图。（电动汽车保有量来自于《2022 年中国新能源汽车行业市场前景及投资研究预测报告》[?]）

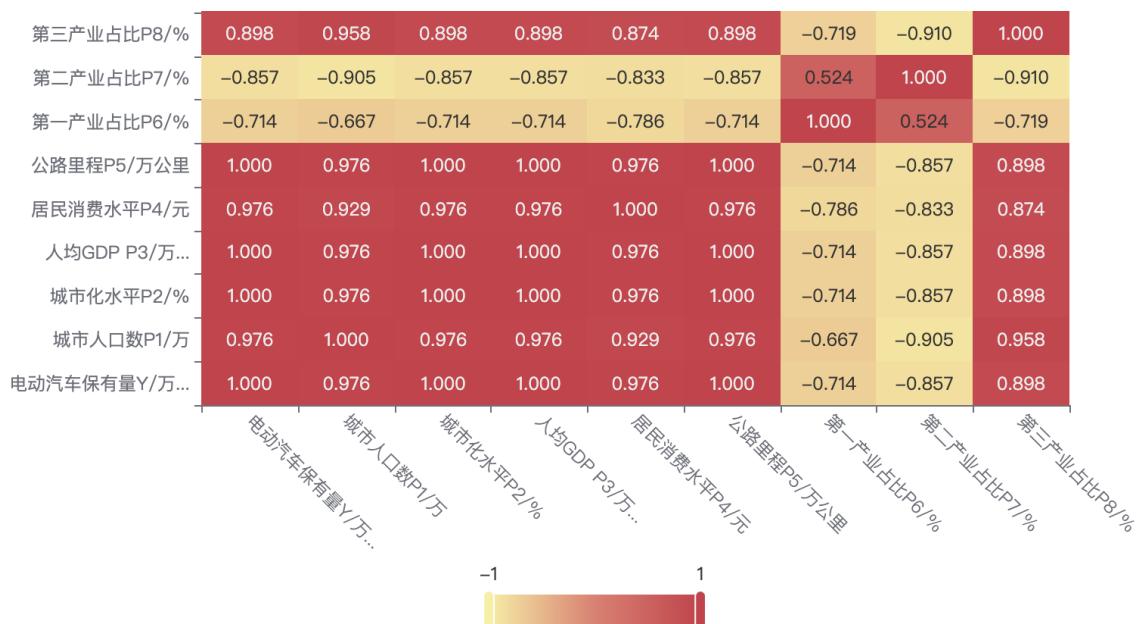


图 1：相关系数热力图

注意到第一产业占比、第二产业占比与电动汽车保有量相关性较低，所以我们在后

续的主成分分析时略去这两种因素。

4.2.4 主成分分析

主成分分析法是运用“降维”思想，把多个指标变换为少数综合指标的多元统计方法，这里的综合指标就是主成分。每个主成分都是原始变量的线性组合，彼此相互独立，并保留了原始变量绝大部分信息。其本质是通过原始变量的相关性，寻求相关变量的综合替代对象，并且保证了转化过程中的信息损失最小。

我们对城市人口数、城市化水平、人均 GDP、居民消费水平、公路里程、第三产业占比六个因素通过 SPSSPRO 进行主成分分析，来综合评价每年度的社会经济发展水平。

KMO 检验是 Kaiser, Meyer 和 Olkin 提出的抽样适合性检验 (Measure of Sampling Adequacy)。该检验是对原始变量之间的简相关系数和偏相关系数的相对大小进行检验。Bartlett 球形度检验用于检验相关阵中各变量间的相关性，是否为单位阵，即检验各个变量是否各自独立。

我们得到 KMO 值 =0.58，显著值 $p<0.001$ ，水平上呈现显著性，拒绝原假设，各变量间具有相关性，主成分分析有效，效度虽然在 0.6 以下，但尚能接受。

成分	特征根		
	特征根	方差百分比	累积
1	5.657	94.286%	94.286%
2	0.278	4.636%	98.922%
3	0.056	0.927%	99.849%
4	0.007	0.111%	99.96%
5	0.002	0.039%	99.999%
6	0.000	0.001%	100.0%

表 2: 总方差解释

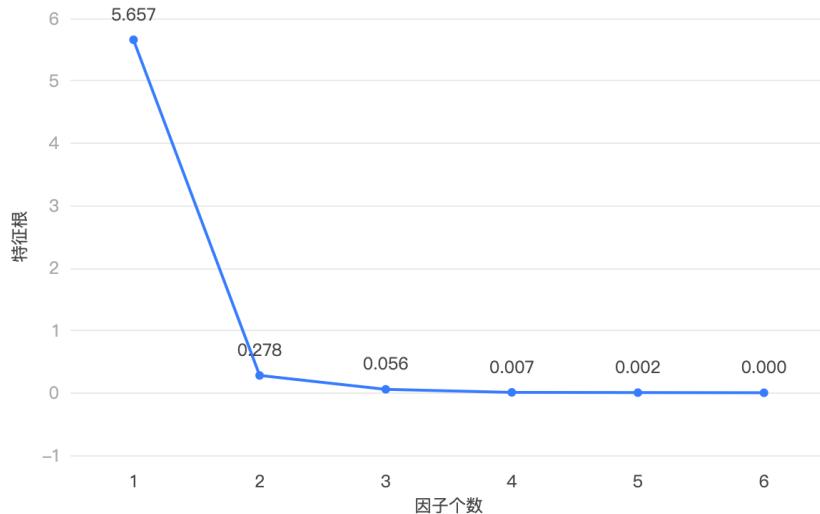


图 2: 因子分析碎石图

上面方差解释表中，我们发现第一主成分的特征值 $=5.657 > 1$ ，且远大于其他特征值，累积方差贡献率为 94.286%，即可以解释总体数据的 94.286%。由碎石图可知，在第二个主成分开始特征值下降坡度变缓，且前两个主成分累积解释贡献率超过 90%，因此我们选择两个主成分进行分析。

从下面的因子载荷矩阵热力图中可以分析每个主成分中隐变量的重要性，其中第一个主成分与前五个因素都具有较大的相关性（0.9 以上），我们概括为“城市总体发展程度”。主成分 2 与第三产业占比的相关程度较大，可以概括为“服务业发展程度”。

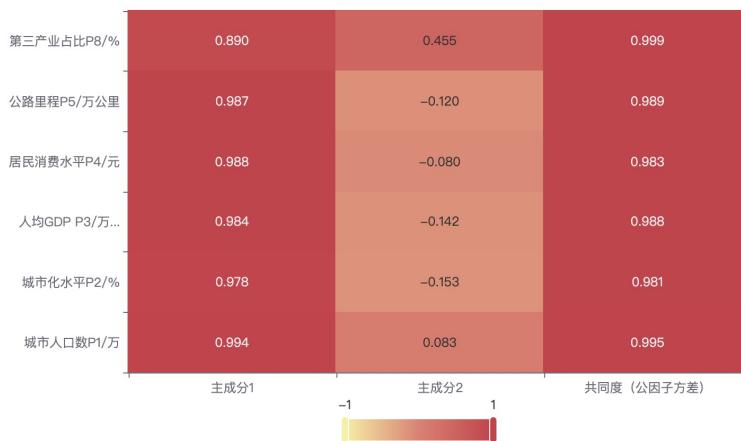


图 3: 相关系数热力图

根据表 2 我们得到模型的公式：

$$F1 = 0.176 \times \text{城市人口数} P1 + 0.173 \times \text{城市化水平} P2 + 0.174 \times \text{人均 GDP} P3 + 0.175 \times \text{居民消费水平} P4 + 0.174 \times \text{公路里程} P5 + 0.157 \times \text{第三产业占比} P8 \quad (19)$$

$$F2 = 0.299 \times \text{城市人口数} P1 - 0.549 \times \text{城市化水平} P2 - 0.511 \times \text{人均 GDP} P3 - 0.289 \times \text{居民消费水平} P4 - 0.433 \times \text{公路里程} P5 + 1.635 \times \text{第三产业占比} P8 \quad (20)$$

由上可以得到:

$$F = (0.943/0.989) \times F1 + (0.046/0.989) \times F2 \quad (21)$$

名称	成分	
	成分 1	成分 2
城市人口数 P1/万	0.176	0.299
城市化水平 P2/%	0.173	-0.549
人均 GDP P3/万元	0.174	-0.511
居民消费水平 P4/元	0.175	-0.289
公路里程 P5/万公里	0.174	-0.433
第三产业占比 P8/%	0.157	1.635

表 3: 成分矩阵表

由下图可知: 第一主成分值与年份呈显著的线性正相关, 故利用线性回归模型得出第一主成分 (y) 与年份 (x) 的关系, 如式:

$$y = 0.4071x - 820.4 \quad (22)$$

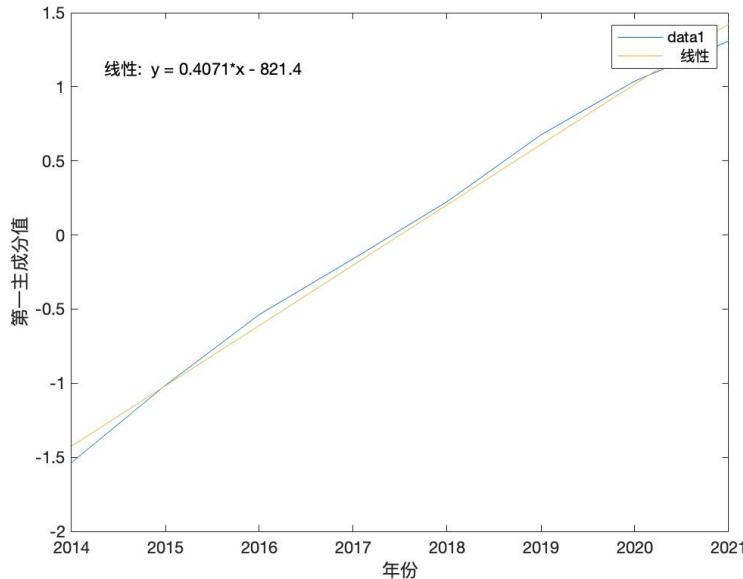


图 4: 第一主成分值与年份呈显著的线性正相关

第一主成分与电动汽车、燃油汽车保有量均存在明显的非线性关系, 故考虑选取第一主成分为基准得到后面生物种群相互竞争模型中的 σ_1 、 σ_2 值。

4.2.5 线性回归（最小二乘法）

我们使用 SPSSPRO，将得到的第一主成分值分别与电动汽车数量、燃油汽车数量作线性回归，得到如下拟合效果图。

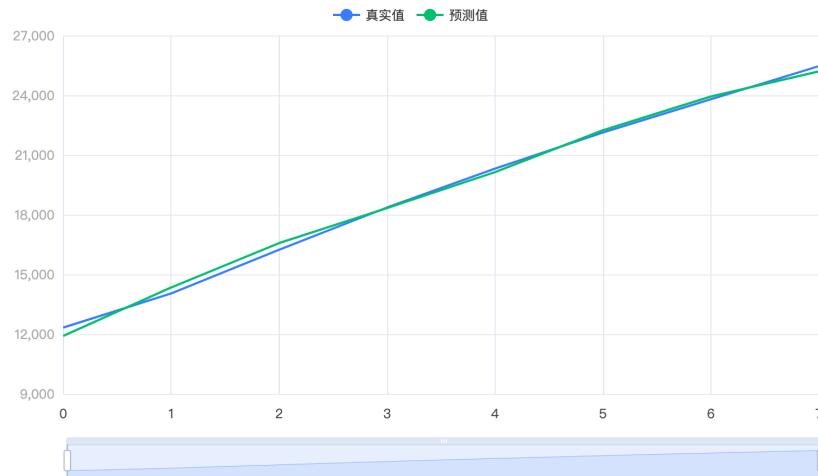


图 5: 拟合效果图

线性回归分析结果 n=8									
	非标准化系数		标准化系数		t	p	VIF	R ²	调整R ²
	B	标准误	Beta						
常数	19083.71	103.596	-		184.213	0.000***	-	0.997	0.996
主成分	4673.396	110.748	0.998		42.198	0.000***	1.000		F=1780.695 P=0.000***

因变量: 燃油汽车保有量

注: ***、**、*分别代表1%、5%、10%的显著性水平

图 6: 线性回归分析结果

$$\text{燃油汽车保有量} = 19083.71 + 4673.396 * \text{主成分}$$

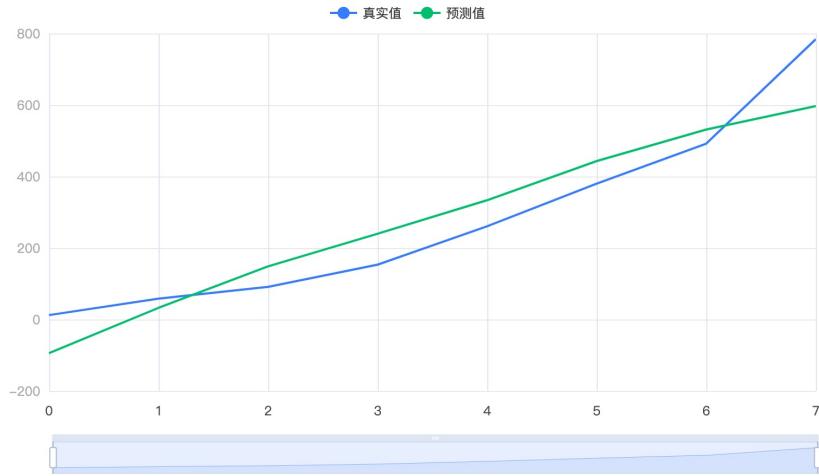


图 7: 拟合效果图

线性回归分析结果 n=8										
	非标准化系数		标准化系数		t	p	VIF	R ²	调整R ²	F
	B	标准误	Beta							
常数	279.04	37.801	-		7.382	0.000***	-	0.858	0.834	F=36.149 P=0.001***
主成分	242.963	40.41	0.926		6.012	0.001***	1.000			

因变量: 电动汽车保有量

注: ***、**、*分别代表1%、5%、10%的显著性水平

图 8: 线性回归分析结果

$$\text{电动汽车保有量} = 279.04 + 242.963 * \text{主成分}$$

以上回归分析结果中, 从 F 检验的结果分析可以得到, 显著性 P 值均为 0.000***, 水平上呈现显著性, 拒绝回归系数为 0 的原假设, 因此模型基本满足要求对于变量共线性表现, VIF 全部小于 10, 模型没有多重共线性问题, 模型构建良好。

$$\sigma_1 = \frac{4673.396}{19083.71} \approx 0.24 \quad (23)$$

$$\sigma_2 = \frac{242.963}{279.04} \approx 0.87 \quad (24)$$

4.3 充电桩增长模型

在初次建设充电桩时, 我们时常无法一次性建设满足要求数量的充电桩。这时就需要根据人口、需求量、财富等因素得到优先级 $Pr_i = \alpha_1 * Po_i + \alpha_2 * q_i + \alpha_3 * E$ 逐步建立充电桩, (对应于 p 较小的时刻)。随着 p 的增长充电桩的数量得以逐步满足需求和冗余。

对于慢充来说，随着电池技术的发展，单次充电跨越时间 T 也会增长，同时随着经济增长，人均电动车数量也会同步增加，这两个因素的预测共同决定了 q_i 的发展。而对快充来说，随着科技的发展，充电速度则会提升（即平均服务时间减少），同时车辆的续航里程也会提升。我们依据电动车的增长模型所算得的数据再代入选址模型可以得到对充电桩选址的动态预测模型。

五、模型求解

5.1 问题一的求解

5.1.1 充电桩选址

我们从网上搜集到了 L 、 F_k 以及 V_p 数据，使用最短路径算法得到道路拓扑数据并将数据导入 python（见附录 A.1）。然后再从网络上得到杭州人口密度的热力图，通过 matlab 程序将两者结合（见附录 A.2.2），如图 9所示：

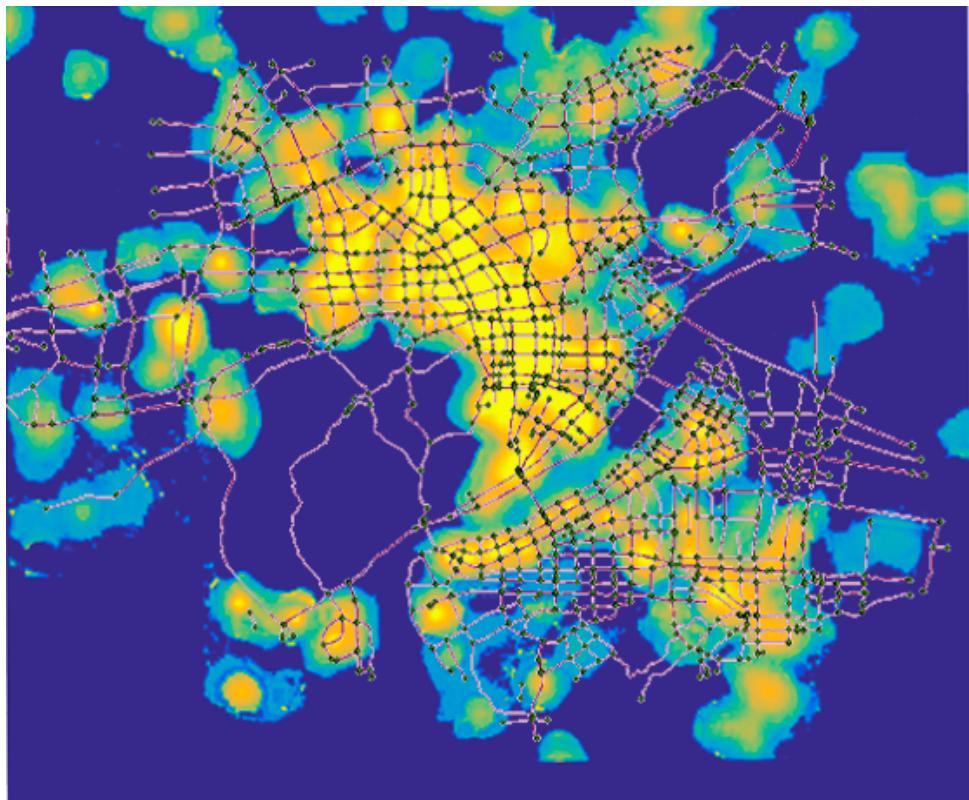


图 9: 杭州道路-人口热力图

接着我们再对线性规划方程 (13) 选用对偶单纯形 (dual simplex) 算法（见附录??）求解得到结果如图所示。

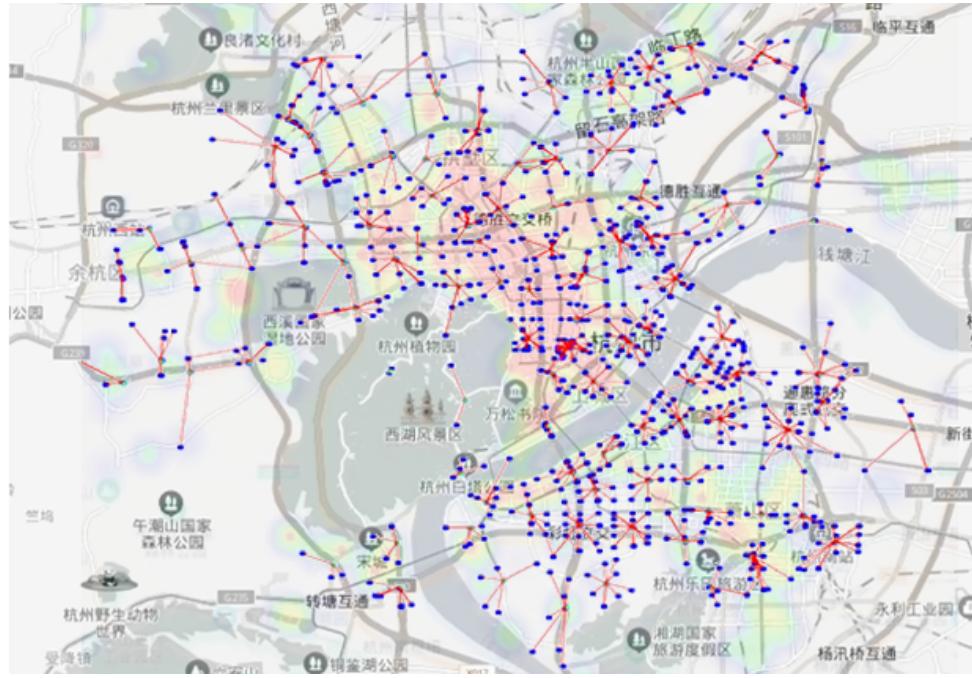


图 10: 优化结果

其中红线代表两个节点间有需求传递。

5.1.2 充电桩未来发展趋势

我们认为充电桩未来发展影响因素有三个：人口因素、财富因素以及技术因素。

人口预测:

通过对近年人口预测论文的阅读，我们发现主流的人口预测模型为灰色模型。因此我们在 SPSSPRO 中使用灰色预测模型 $GM(1,1)$ 通过 1964—2021 年我国总人口数对未来十年我国总人口数。

得到灰色模型构建结果如下：

发展系数 a	灰色作用量 b	后验差比 C 值
-0.011	554.104	0.035

表 4: 灰色模型构建

从上表分析可以得到，后验差比值为 0.035，模型精度高。

模型拟合预测图：

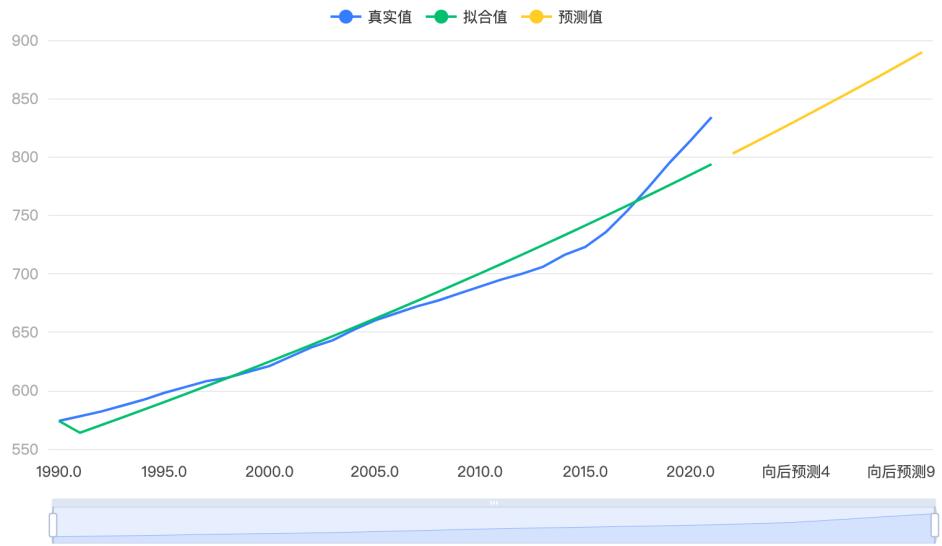


图 11: 模型拟合预测图

预测得 2027 年杭州总人口数为 840.3 万人，2032 年杭州总人口数为 889.6 万人。

经济预测：

我们以 1990-2021 年杭州市人均 GDP 作为杭州市经济的衡量指标。我们在 SPSSPRO 中使用 ARIMA 模型对未来十年杭州市人均 GDP 做预测。

ADF 检验表						
变量	差分阶数	t	p	AIC	临界值	
1%	5%	10%				
人均 gdp	0	0.239	0.974	452.928	-3.661	-2.961
1	-6.73	0.000***	451.419	-3.67	-2.964	-2.621
2	-2.027	0.275	435.42	-3.809	-3.022	-2.651

表 5: ADF 检验表

在差分为 0 阶、2 阶时，显著性 P 值为 0.974、0.275，水平上不呈现显著性，不能拒绝原假设，该序列均为不平稳的时间序列。在差分为 1 阶时，显著性 P 值为 0.000***，水平上呈现显著性，拒绝原假设，该序列为平稳的时间序列。

1 阶差分时序图：

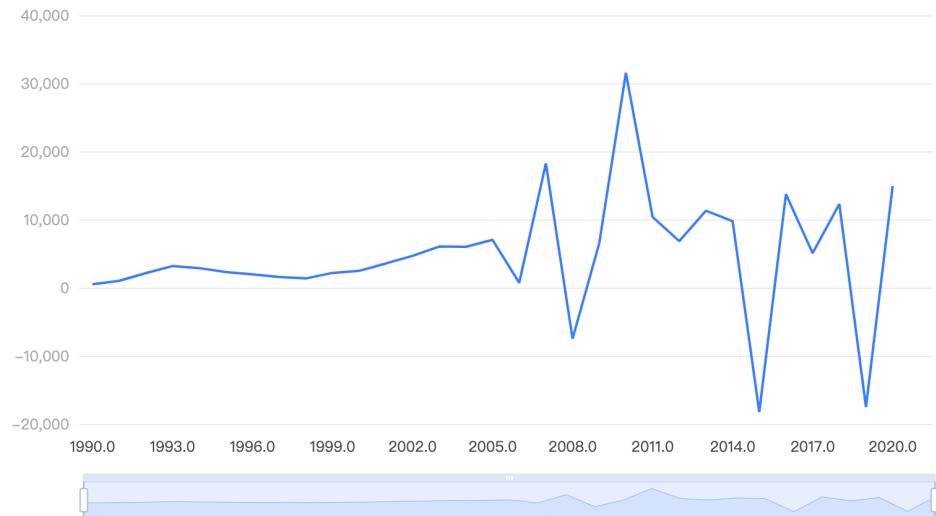


图 12: 最佳差分序列图

最终差分数据自相关图 (ACF) :



图 13: 最终差分数据自相关图 (ACF)

ARIMA 模型 $(0,1,0)$ 检验表如下，得到模型结果为 ARIMA 模型 $(0,1,0)$ 检验表，从 Q 统计量结果分析可以得到：Q6 在水平上不呈现显著性，不能拒绝模型的残差为白噪声序列的假设，同时模型的拟合优度 R2 为 0.968，模型表现优秀，模型基本满足要求。

ARIMA 模型 (0,1,0) 检验表

项	符号	值
	Df Residuals	30
样本数量	N	32
Q 统计量	Q6(p 值)	2.026(0.155)
Q12(p 值)	9.435(0.151)	
Q18(p 值)	16.005(0.191)	
Q24(p 值)	16.657(0.547)	
Q30(p 值)	17.06(0.846)	
信息准则	AIC	656.864
BIC	659.732	
拟合优度	R ²	0.968

表 6: ARIMA 模型 (0,1,0) 检验表

得到时间序列图：

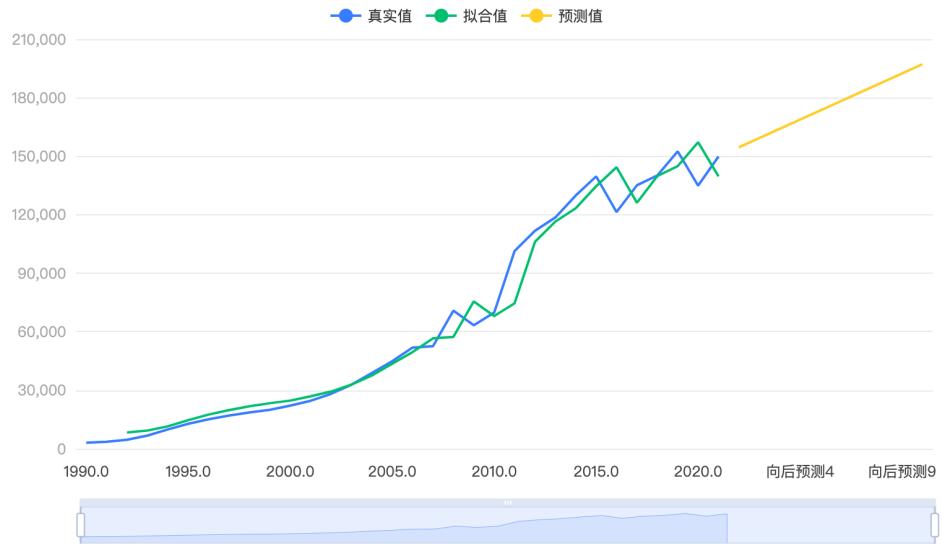


图 14: 时间序列图

预测 2027 年杭州市人均 GDP 为 173526 人民币，2032 年杭州市人均 GDP 为 197195 万元。

技术发展

技术的发展可以使汽车的续航里程和充电速度增加，我们假设这两者同步增加（即充电服务时间保持不变），并只对续航里程进行一个预测（由于没有人收集续航里程的数据，这里只能根据不同年份的报道做一个粗略的预测），如图所示：

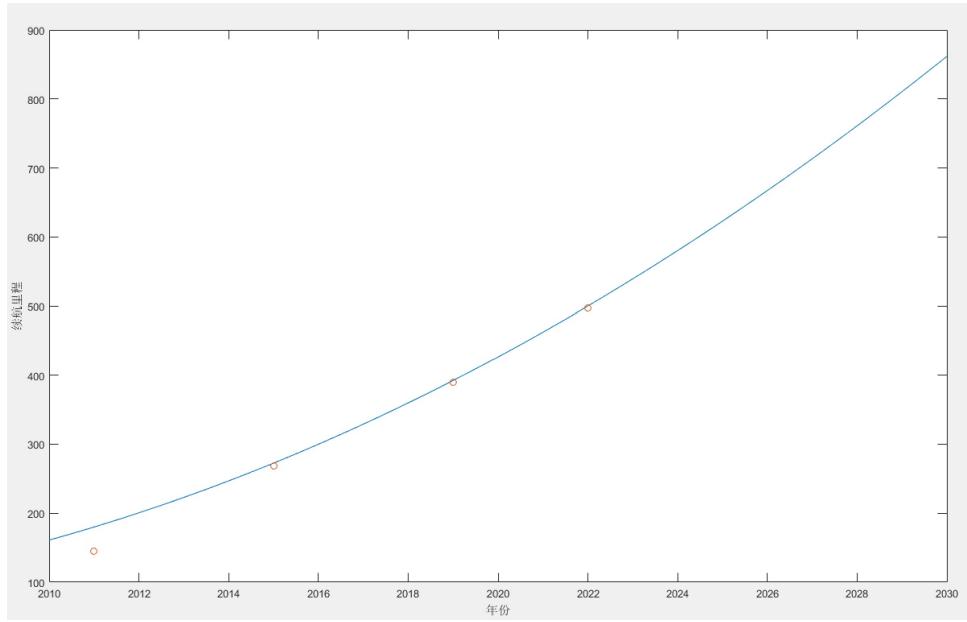


图 15: 续航里程随年份变化

充电桩增长预测根据我们对于电动车增长的预测模型，代入重新计算了充电桩的选址，可以得到以下充电桩发展图：

首先是初次建立充电桩时的逐步选址图，以下分别列出了 p 只有当前需求 25%、50%、75% 和 100% 时的优先级选址：

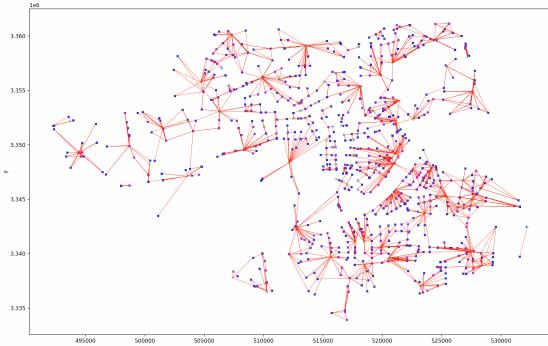


图 16: 3% 时

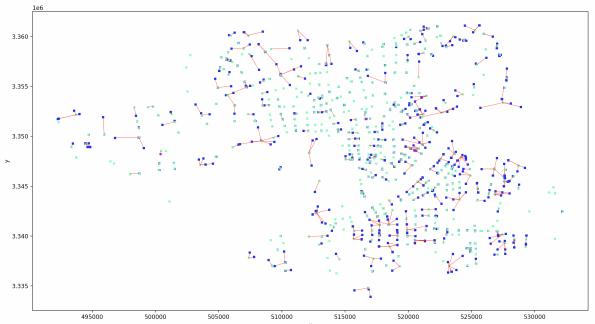


图 17: 5% 时

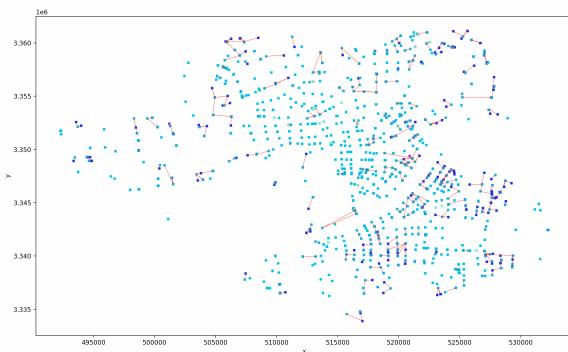


图 18: 10% 时

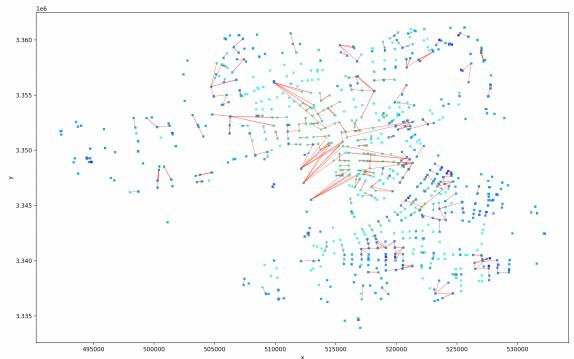


图 19: 20% 时

如图, 图中连接的红线(代表节点对外部的需求)随着建设的完成度增加(可用点越来越多)越来越少, 也就是说人们不用去节点外部充电, 便利性逐步趋向于理论最高值; 此外充电桩的选址随着建设的完成度增加, 会逐渐趋于使每个节点都有充电桩, 因此对于以后的预测将只对充电桩数量进行预测。

接着是随着电动车发展而导致的充电桩建设需求数量预测(假定政府会根据预测提前开始建造充电设施, 因此无需关心优先级), 如图所示:

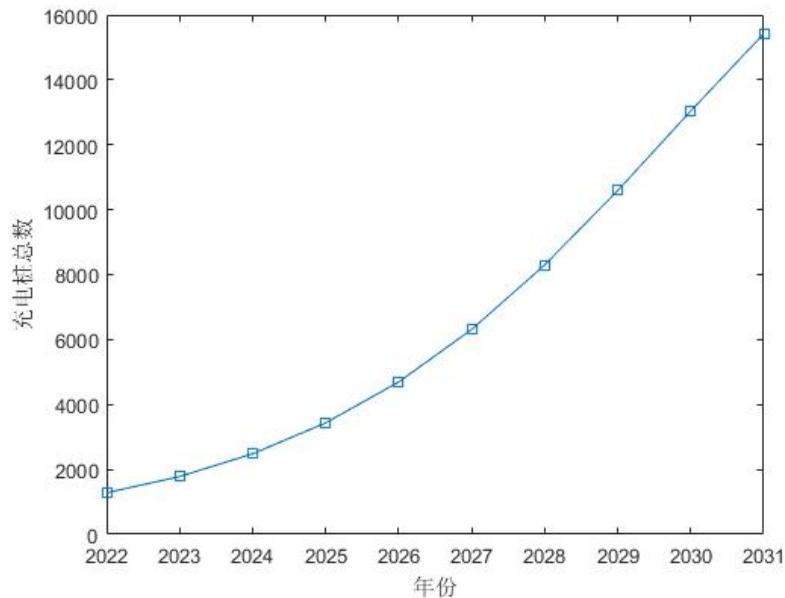


图 20: 充电桩数量预测

图中可以看出, 如果续航里程如预测那样发展, 并不能抵消电动车数量增加带来的对充电桩数量增加, p 值随着时间推移会先以一个越来越快的速率增加, 然后增加速率会趋于平稳。

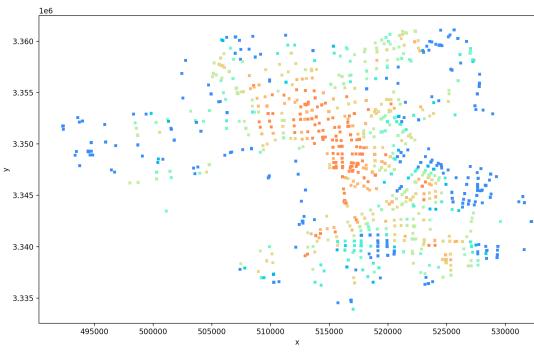


图 21: 5 年后



图 22: 十年后

如图所示，随着电动车的增加， p 值不断增加，市中心需求量大的地区红色加深，即每个节点充电桩数量增加。此外，红色趋于向外移动，但速度缓慢，也就是说郊区充电桩逐渐缓慢地被充电桩完全覆盖。

5.2 问题二的求解

5.2.1 模型的求解

我们使用 matlab 对该生物种群相互竞争模型进行求解，得到如下结果：

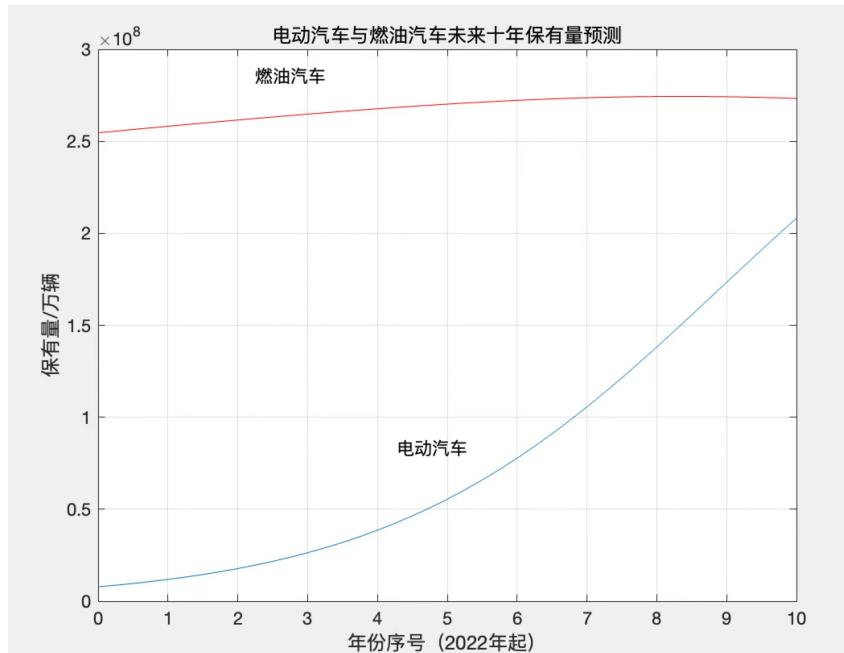


图 23: 电动汽车与燃油汽车未来十年保有量预测

根据最新统计数据，2022 年第一季度电动汽车保有量为 891.5 万辆，与我们预测的结果 873.9 万辆基本接近。

六、模型结论分析

从我们预测曲线中注意到，传统燃油汽车未来十年内的保有量依然增长缓慢，并且与 2030 年开始出现拐点，保有量出现减少趋势。电动汽车增长迅速，预计到五年后保有量达到 50 万辆以上，未来十年内增长趋势都将变快。且从主成分分析中可以看出，城市发展程度，尤其是人均 GDP、公路里程数对电动汽车数量有显著影响。

参考文献

- [1] 中华人民共和国统计局. 中国统计年鉴 (2021)[M]. 北京：中国统计出版社,2022.
- [2] 徐浩. 面向规模化电动汽车入网的充电站布局优化及有序充电策略研究 [D]. 湖北: 华中科技大学,2015. DOI:10.7666/d.D731664.
- [3] 刘斌, 赵天舒, 张冉霞. 基于改进 PCA-Logistic 模型对个人汽车保有量预测 [J]. 公路交通科技,2020,37(08):136-143.
- [4] 张兰怡, 胡喜生, 陈清耀, 邱荣祖. 基于 PCA-Logistic 回归的汽车保有量预测研究 [J]. 重庆交通大学学报 (自然科学版),2017,36(05):104-109.
- [5] 姜靖, 戴慧兰. 我国到底能承载多少汽车? [N]. 科技日报,2011-09-26(011).DOI:10.28502/n.cnki.nkjrb.2011.004243.
- [6] 何蔼琳, 王豹. 基于改进 LSTM 算法的城市轨道交通 OD 客流预测方法 [J]. 综合运输,2021,43(04):67-72+87.
- [7] 何洪磊. 使用 M/M/n 模型优化服务器队列性能 [J]. 现代信息科技,2021,5(23):80-83,87. DOI:10.19850/j.cnki.2096-4706.2021.23.021.

七、附录

A 代码

A.1 导入数据到 python

A.1.1

Listing 1: load.py

```
1 import numpy as np
2 with open ("OD矩阵表.txt", mode="r") as f:
3     str = f.readlines()
4 size=int((str[-1].split(',') )[2])
5 data = np.zeros((size, size), dtype = float)
6 a=str[1].split(',')
7 for line in str[1:]:
8     list=line.split(',')
9     data[int(list[2])-1][int(list[3])-1]=list[5].strip("\n")
10 np.save("data",data)
```

A.1.2

Listing 2: loadxy.py

```
1 import numpy as np
2 with open ("杭州xy.txt", mode="r") as f:
3     str = f.readlines()
4 size=int((str[-1].split(',') )[0])+1
5 data = np.zeros((size,2), dtype = float)
6 a=str[1].split(',')
7 for line in str[1:]:
8     list=line.split(',')
9     data[int(list[0])][0]=list[1].strip("\n")
10    data[int(list[0])][1]=list[2].strip("\n")
11 np.save("杭州xy",data)
12 demandCoordinates=[(data[i][0],data[i][1]) \
13     for i in range(data.shape[0])]
14 print(data)
```

```
15 #np.save("xy坐标",data)
```

A.2 提取热力图数据的 matlab 代码

A.2.1

Listing 3: **heatcolortrans.m**

```
1 %<< 2      f << I , ' << , heat2data - % + 
2 
3 workingdtr='C:\Users\heyiyang\Desktop\%g\% \paper\pic\' ;
4 % % L%
5 dataim=imread([workingdtr,'zh .png']);
6 workim=rgb2hsv(dataim);
7 [x,y,z]=size(workim);
8 datax=1:x;%x +
8 datay=1:y;%y +
```

A.2.2

Listing 4: **heat2data.m**

```
1 
2 workingdtr='';
3 workim=imread([workingdtr,'人口热力图-work.png']);
4 cpim=imread([workingdtr,'人口热力图-ovf.png']);
5 
6 %-----对准坐标系
7 %由于坐标系均为直角坐标系，且y轴方向均为北方，所以只有四个
    自由度X'=SX+V，也就是说只需要两个二维点即可确定坐标系变换
8 %注意选的点不能太近
9 syms sx sy vx vy
10 s=[sx,0;0,sy];%伸缩
11 v=[vx,vy];%平移
12 p1=[498471.0 3342937.8483];%基准点[57]
13 p2=[532472.0 3341447.9522];%基准点[-1]
14 p3=[520953.0 3335486.6781];%检验点，用来判断误差多大[2189]
15 imshow(cpim)%比较，选出坐标点
```

```

16 x1=[24.3947,288.7591];
17 x2=[542.7264,310.7756];
18 x3=[366.5943,398.2127];
19 [Sx,Sy,Vx,Vy]=solve([x1*s+v==p1,x2*s+v==p2],[sx,sy,vx,vy]);
20 V=double([Vx,Vy]);
21 S=double([Sx,0;0,Sy]);
22 Error=(x3*S+V-p3)/sum(p3-p2);
23 if sum(abs(Error))>0.01
24     warning('误差超过了1%!')
25     Error
26 end
27
28 data=readNPY("杭州mergedXY.npy");
29
30 dataxy=data;%输入要读取的点的坐标数据
31 dataheat=[];
32
33 for i=1:length(dataxy)
34     X=(dataxy(i,:)-V)*inv(S)
35     if X(1)<0|X(2)<0
36         dataheat=[dataheat;0];
37         continue
38     end
39     dataheat=[dataheat;255-workim(round(X(2)),round(X(1))
40 ,1)];
40 end
41 %输出数据
42 writeNPY(dataheat,"dataheat.npy")

```

A.3 热力图数据换算人口密度的 python 代码

Listing 5: calculateq.py

```

1 import matplotlib.pyplot as plt
2 import matplotlib.colors as color
3 import numpy as np
4

```

```

5 heat=np.load( "dataheat.npy" )
6 pop_MAX=1.1*10**4 # 1.1 万人每平方公里
7 pop_MIN=2000
8 size=len(heat)
9 pop_density=np.zeros(size)
10 for i in range(size):
11     cof=heat[i]/255
12     print(cof)
13     pop_density[i]=pop_MIN+(pop_MAX-pop_MIN)*cof
14     print(pop_density[i])
15 np.save("pop_density",pop_density)

```

A.4 合并冗余数据的 matlab 代码

Listing 6: **merge.m**

```

1
2 data=readNPY('nyod.npy');
3 xy=readNPY('nyxy.npy');
4 tol_dis=40;
5
6 workmat=[data,xy];
7 [l,bin]=size(workmat);
8 judge=find(sum(workmat==0)>l/2);
9 workmat(judge,:)=[];
10 workmat(:,judge)=[];
11 p=workmat<tol_dis;
12 i=1;
13 j=1;
14 while i<=l
15     j=i+1;
16     while j <= l
17         if p(j,i)
18             workmat(j,:)=[];
19             workmat(:,j)=[];
20         end
21     [l,bin]=size(workmat);

```

```

22         j=j+1;
23     end
24     i=i+1;
25 end
26 writeNPY(workmat(:,1:end-2), 'nymerged0D.npy')
27 writeNPY(workmat(:,end-1:end), 'nymergedXY.npy')

```

A.5 生物竞争模型的 matlab 代码

Listing 7: 生物竞争.m

```

1 t0=0;tf=10;
2 [t,y]=ode45('logistic',[t0 tf],[784e4 25462e4]);%取初始条件
    一时，求微分方程数值解
3 subplot(1 ,1,1);
4 plot(t,y(:,1),t,y(:,2),'r');%画出x1(t),x2(t)曲线图
5 xlabel('年份序号（2022年起）');
6 ylabel('保有量/万辆');
7 gtext('电动汽车');
8 gtext('燃油汽车');%作标记
9 title('电动汽车与燃油汽车未来十年保有量预测');
10 grid on;

```

A.6 logistic 的 matlab 代码

Listing 8: logistic.m

```

1 function dy=logistic(t,y)
2 p1=0.24; p2=0.87; r1=0.5; r2=0.04; N1=40500e4; N2=40500e4;
3 dy=zeros(2,1);
4 dy(1)=r1 *y(1).*(1-y(1)./N1-p1*y(2)./N2);
5 dy(2)=r2 *y(2).*(1-y(2)./N2-p2*y(1)./N1);

```

A.7 求解优化问题以及画图的 python 代码

Listing 9: P_location.py

```

1
2 import matplotlib.pyplot as plt
3 import matplotlib.colors as color
4 import numpy as np
5 from gurobipy import Model, GRB, quicksum
6 # data=np.load("杭州xyn.npy")
7 data=np.load("杭州mergedXY.npy")
8
9 demandCoordinates=[(data[i][0],data[i][1]) \
10     for i in range(data.shape[0])]
11 size=data.shape[0]
12 A=[(i,j) for i in range(size) for j in range(size)]
13 N=[i for i in range(size)]
14 pop_density=np.load("pop_density.npy")
15 for xy in demandCoordinates:
16     plt.plot(xy[0],xy[1],c="b",marker="s")
17 pop_sum=np.sum(pop_density)
18 pop_average=pop_sum/size
19
20 cof=1.2
21 q=[pop_density[i] for i in N]
22 p=1000
23 ablty=cof*pop_sum/p
24
25 geo_dis={(i,j):np.hypot(demandCoordinates[i][0]\
26     -demandCoordinates[j][0],
27     demandCoordinates[i][1]\
28         -demandCoordinates[j][1]) for i,j in A} # 几何距离
29 # od=np.load("杭州ODN.npy")
30 od=np.load("杭州mergedOD.npy") # np格式的od矩阵
31 od_dis={(i,j):od[i][j] for i,j in A} # 字典格式的od矩阵
32 dis=od_dis
33
34 mdl=Model("p-center")
35 x=mdl.addVars(A,vtype=GRB.CONTINUOUS,ub=1,lb=0)
36 # x=mdl.addVars(A,vtype=GRB.BINARY)
37 n=mdl.addVars(N,vtype=GRB.INTEGER)

```

```

38 mdl.update()
39
40 mdl.ModelSense=GRB.MINIMIZE
41 mdl.setObjective(quicksum(x[i,j]*dis[i,j]*q[j]\
42     for i,j in A))
43
44 mdl.addConstrs(quicksum(x[i,j] for j in N)== 1 for i in N);
45
46 mdl.addConstr(quicksum(n[i] for i in N )== p);
47
48 mdl.addConstrs((quicksum(q[i]*x[i,j] for i in N)<=n[j]*ablty )\
49     for j in N);
50
51 mdl.optimize()
52
53 # 画所有点及标注需求
54 markersize=3
55 linewidth=0.5
56 for i in N:
57     xy=demandCoordinates[i]
58     plt.plot(xy[0],xy[1],c="b",\
59             marker="s",markersize=markersize)
60     #plt.text(xy[0],xy[1],f"{q[i]}")
61 norm=color.Normalize(vmax=13,vmin=0)
62 # 画需求量
63 for i in N:
64     if (n[i].x>0):
65         xy=demandCoordinates[i]
66         col=plt.cm.rainbow
67         nMax=0.0
68         for ni in N:
69             # print(type(n[ni].x),n[ni].x,type(nMax))
70             if n[ni].x>nMax:
71                 nMax=n[ni].x
72             # norm=color.Normalize(vmax=round(nMax)+1,vmin=0)
73             plt.plot(xy[0],xy[1],marker="s",\
74                     color=col(norm(n[i].x)),markersize=markersize)

```

```

75         # plt.text(xy[0]-5,xy[1]-5,f'{n[i].x}',color='b')
76
77
78 # 画供应线
79 for i,j in A:
80     if x[i,j].x>0.0:
81         demandCoordinates[i][0],demandCoordinates[i][1]
82         demandCoordinates[j][0],demandCoordinates[j][1]
83         plt.plot([demandCoordinates[i][0],\
84                 demandCoordinates[j][0]]\
85                 ,[demandCoordinates[i][1],\
86                 demandCoordinates[j][1]],\
87                 color='r',linewidth=linewidth)
88
89 plt.xlabel("x")
90 plt.ylabel("y")
91 plt.show()
92
93 array=np.array([])
94 for i in N:
95     array=np.append(array,n[i].x)
96 array
97 np.save("n.npy",array)
98
99 array2=np.zeros((size,size))
100 for i,j in A:
101     array2[i,j]=x[i,j].x
102 np.save("x.npy",array2)

```

A.8 遗传算法解规划的 python 代码

Listing 10: p-中值.py

```

1 import math
2 import random
3 import pandas as pd
4 import matplotlib.pyplot as plt

```

```
5  from matplotlib.pyplot import mpl
6  # 分隔
7
8  # 计算路径距离，即评价函数
9
10
11 def calFitness(chrom, dis_matrix, cnum, d, c):
12     # cnum:备选点数量
13     # d:需求量列表，对应demandCoordinates
14     # c:
15     dis_sum = 0 # 总距离
16     dis = 0 # 距离
17     declist = [] # 位置坐标
18     dec_chrom = chrom.copy() # 对应坐标
19     d_list = [[] for i in range(cnum)] # 列出每个
20     weight = [0 for i in range(cnum)]
21     demand = [0 for i in range(cnum)]
22     for i in range(cnum):
23         if chrom[i] >= 1:
24             for j in range(chrom[i]):
25                 declist.append(i)
26     for i in range(cnum, len(dec_chrom)):
27         dec_chrom[i] = declist[dec_chrom[i]-1]
28         d_list[dec_chrom[i]].append(i-cnum)
29         weight[dec_chrom[i]] += \
30             dis_matrix.loc[i-cnum, dec_chrom[i]]*c[i-cnum]
31         demand[dec_chrom[i]] += d[i-cnum]
32
33     dis_sum = 0
34     for i in range(len(demand)):
35         if demand[i] > c[i]:
36             dis_sum += (demand[i]-c[i])*10000
37     dis_sum += sum(weight)
38     return round(dis_sum, 1)
39
40
41 def traversal_search(chrom, dis_matrix,\
```

```

42     tabu_list , cnum , d , c , p):
43     # 邻域随机遍历搜索，成对交换+单点变异
44     traversal = 0 # 搜索次数
45     traversal_list = [] # 存储局部搜索生成的解充当局部禁忌表
46     traversal_value = [] # 存储局部解对应路径距离
47     while traversal <= traversalMax:
48         new_chrom = chrom.copy() # 复制当前路径
49         # 并交换生成新路径
50         pos1 , pos2 = random.randint(
51             0 , cnum-1) , random.randint(0 , cnum-1) # 交换点
52         while(new_chrom[ pos2 ] == 0):
53             pos2 = random.randint(0 , cnum-1)
54         new_chrom[ pos1 ] = new_chrom[ pos1 ]+1
55         new_chrom[ pos2 ] = new_chrom[ pos2 ]-1
56
57         for i in range(cnum , len(chrom)):
58             if random.random() > 0.8: # 一定概率改变
59                 new_chrom[ i ] = random.randint(1 , p)
60
61         new_value=calFitness(new_chrom , dis_matrix ,cnum ,d ,c)
# 当前路径距离
62         # 新生成路径不在全局禁忌表和局部禁忌表中，为有效搜索
63         if (new_chrom not in traversal_list)\&(new_chrom not in tabu_list):
64             traversal_list.append(new_chrom)
65             traversal_value.append(new_value)
66             traversal += 1
67
68
69         return min(traversal_value) , traversal_list[\
70             traversal_value.index(min(traversal_value))]

71
72
73 def initialize(dnum , cnum , p):
74     """
75     in :dnum-需求点数量， cnum-备选点数量 ,
76     out :染色体
77     """

```

```

78     clist = [random.randint(0, cnum-1) \
79             for i in range(p)]
80     cchrom = [clist.count(i) for i in range(cnum)]
81     dchrom = [random.choice(range(1, p+1)) \
82                 for i in range(dnum)]
83     chrom = cchrom + dchrom
84
85     return chrom
86
87
88 # 画散点图
89 def draw_sca(Coordinates1, Coordinates2, d):
90     x, y = [], []
91     pad = 2
92     x = [i[0] for i in Coordinates1]
93     y = [i[1] for i in Coordinates1]
94     plt.scatter(x, y, color='#ff69E1', marker='o')
95
96     x = [i[0] for i in Coordinates2]
97     y = [i[1] for i in Coordinates2]
98     plt.scatter(x, y, color='#4169E1', marker='*')
99     for i in range(len(d)):
100         plt.text(Coordinates1[i][0]-pad, \
101                   Coordinates1[i][1]-pad, \
102                   str(d[i]), color='red')
103
104
105     plt.xlabel('x')
106     plt.ylabel('y')
107     plt.show()
108
109
110 # 画分布图
111 def draw_path(chrom, demandCoordinates, \
112               centerCoordinates, p, cnum, d):
113     pad = 0.2
114     centerlist = []

```

```

115     clist = []
116     for i in range(cnum):
117         if chrom[i] >= 1:
118             for j in range(chrom[i]):
119                 centerlist.append(centerCoordinates[i])
120                 clist.append(i)
121     print(chrom)
122     print(centerlist)
123     for i in range(cnum, len(chrom)):
124         if chrom[i] >= 1:
125             plt.plot([centerlist[chrom[i]-1][0], \
126                     demandCoordinates[i-cnum][0]], \
127                     [centerlist[chrom[i]-1][1], \
128                      demandCoordinates[i-cnum][1]], 'r-', \
129                     color='#4169E1', alpha=0.8, linewidth=0.8)
130             plt.text(centerlist[chrom[i]-1][0]+pad, \
131                     centerlist[chrom[i]-1][1]+pad, \
132                     str(chrom[clist[chrom[i]-1]]))
133
134     draw_sca(demandCoordinates, centerCoordinates, d)
135
136
137 if __name__ == '__main__':
138     # 参数
139     CityNum = 50    # 城市数量
140     MinCoordinate = 0    # 二维坐标最小值
141     MaxCoordinate = 100    # 二维坐标最大值
142
143     # TS参数
144     tabu_limit = 2000    # 禁忌长度
145     iterMax = 2000    # 迭代次数
146     traversalMax = 100    # 每一代局部搜索次数
147
148     tabu_list = []    # 禁忌表
149     tabu_time = []    # 禁忌次数
150     best_value = math.pow(10, 10)
151     # 较大的初始值，存储最优解

```

```

152 best_line = [] # 存储最优路径
153
154 ablty = 4
155 allow=3
156
157 # 需求点位置及需求量，备选中心位置及能力
158 demandCoordinates = [(88, 16), (25, 76), (69, 13), \
159     (73, 56),(80, 100), (22, 92), (32, 84), (73, 46) \
160     , (29, 10),(92, 32), (44, 44), \
161     (55, 26), (71, 27), (51, 91), (89, 54), \
162     (43, 28), (40, 78), (60, 66)]
163 print(len(demandCoordinates))
164 centerCoordinates = demandCoordinates
165 d = [1, 2, 1, 3, 5, 3, 4, 6, 2, 1, 3, 4, 1, \
166     2, 6, 1, 4, 2] # 需求量，对应demandCoordinates
167 c = [ablty for i in range(len(d))]
168     # 能力都设置为25，对应centerCoordinates
169 draw_sca(demandCoordinates, \
170     centerCoordinates, d) # 位置图
171
172 p = round(sum(d)/ablty)+allow # 待决策物流中心数量
173 dnum = len(demandCoordinates) # 需求点数量
174 cnum = len(centerCoordinates) # 备选中心数量
175
176 # 计算中心与需求点之间的距离
177 dis_matrix = pd.DataFrame(
178     data=None, columns=range(cnum), index=range(dnum))
179 for i in range(dnum):
180     xi, yi = demandCoordinates[i][0], \
181         demandCoordinates[i][1]
182     for j in range(len(centerCoordinates)):
183         xj, yj = centerCoordinates[j][0], \
184             centerCoordinates[j][1]
185         dis_matrix.iloc[i, j] = round(\
186             math.sqrt((xi-xj)**2+(yi-yj)**2), 2)
187
188 # 初始化，随机构造

```

```

189     num = 50
190     chroms = [ initialize(dnum, cnum, p) \
191         for i in range(num)]
192     chroms = [[0, 1, 2, 1, 1, 1, 0, 1, 1, 0,\
193         1, 1, 1, 1, 0, 1, 0, 1, 3, 12, 3, 13, \
194         5, 6, 1, 4, 8, 11, 7, 14, 13, 12, 9,\\
195             8, 10, 2]]
196     values = [ calFitness(chrom, dis_matrix,\
197         cnum, d, c) for chrom in chroms]
198     print(values)
199     best_value = min(values)
200     best_chrom = chroms[values.index(best_value)]
201     chrom, value = best_chrom, best_value
202
203     # 存储当前最优
204     print('初代最优点 %.1f' % (best_value))
205
206     best_value_list = []
207     best_value_list.append(best_value)
208     # 更新禁忌表
209     tabu_list.append(best_chrom)
210     tabu_time.append(tabu_limit)
211
212     draw_path(chrom, demandCoordinates, \
213         centerCoordinates, p, cnum, d)
214
215     itera = 0
216     while itera <= iterMax:
217         new_value, new_chrom = traversal_search(
218             chrom, dis_matrix, tabu_list, cnum, d, c, p)
219         if new_value < best_value: # 优于最优解
220             best_value, best_chrom = new_value, new_chrom
221             best_value_list.append(best_value)
222             print('第%.d代最优点 %.1f' %(itera, best_value))
223             chrom, value = new_chrom, new_value # 更新当前解
224
225             # 更新禁忌表

```

```
226     tabu_time = [x-1 for x in tabu_time]
227     if 0 in tabu_time:
228         tabu_list.remove(tabu_list[tabu_time.index(0)])
229         tabu_time.remove(0)
230
231     tabu_list.append(chrom)
232     tabu_time.append(tabu_limit)
233     itera += 1
234 # 画图
235 draw_path(best_chrom, demandCoordinates,
236             centerCoordinates, p, cnum, d)
```