五一数学建模竞赛

承　诺　书

我们仔细阅读了五一数学建模竞赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与本队以外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其它公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们愿意承担由此引起的一切后果。

我们授权五一数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

参赛题号（从 A/B/C 中选择一项填写）：B

参赛队号：T33608134044

参赛组别（研究生、本科、专科、高中）：本科

所属学校（学校全称）：中国矿业大学

参赛队员：　队员 1 姓名：丘均濠

队员 2 姓名：周文凯

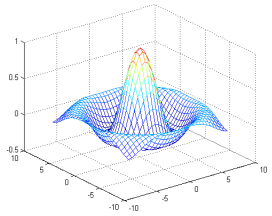
队员 3 姓名：臧召宾

联系方式：　Email：2368126021@qq.com　联系电话：18852182039

日期：　2023　年 4　月 28　日

（除本页外不允许出现学校及个人信息）

五 一 数 学 建 模 竞 赛



题 目：快递需求分析问题

关键词：Topsis 评价法 PageRank 算法 LSTM 神经网络 时间序列预测 Bellman-ford 最短路算法 3sigma 准则 核密度估计

摘 要：

本文通过对某快递公司的实际数据进行分析，建立了数学模型，解决了该公司面临的几大问题。

对于问题一，运用 Python 运算得到各个城市在 2018 年到 2019 年间各自的总发货量与收货量；对各个年份的总值进行逐月差分，得到当月的增长/下降趋势，然后分别对各个城市进行加总取月份的平均值；对于相关性，采取了 PageRank 算法计算各个城市的中心度；最后采用 topsis 模型，得到最终的评价结果为 L、G、H、V、X。

对于问题二， 先运用 Python 提取历史各个城市各天的快递流通量（也就是发货量+收货量），之后运用 SPSS，进行时间序列预测，得到相关的结果。

对于问题三，只需要在问题二的基础上稍作改动即可，构建一矩阵，将某 i 城市通往某 j 城有货物赋值 1，没有货物赋值 0，0 即为不可通行的，对于该矩阵进行模型训练，然后对于相对应可通行的数据进行预测就好。

对于问题四，首先需要对运输网络构建图模型，然后提取各天数据，遍历各天数据，同时将发货城市与收货城市分别作为出发点和终点，代入到带有限制条件的 Bellman-ford 函数中进行最短路探索，找出最短路花费并加到当天的费用中，不断循环，直到求出各天的最小费用。保留两位小数，从 2023 年 4 月 23—27 日每日的最低运输成本分别是 977.07、1181.98、1070.40、950.66、1007.61

对于问题五，运用 3sigma 准则求出固定需求常数。而至于非固定需求的均值与标准差，则是在原有数据的基础上减去相对应的固定需求常数，并运用高斯核函数作为核密度估计，求出目标城市对的运输快递数量的概率密度，再计算相对应的值。

一、问题重述

网络购物作为我国的一种重要的购物方式，在人们的日常生活中发挥着重要的作用。而快递运输是其中重要的一环，为了帮助快递公司布局仓库站点、节约存储成本、规划运输线路，需要解决如下几个问题：

问题一：运用快递公司记录的相关快递运输数据，建立对应的指标，从而建立数学模型，选出前五个最重要的城市。

问题二：运用快递公司记录的相关快递运输数据，建立数学模型，预测 2019 年 4 月 18 日和 2019 年 4 月 19 日各“发货-收货”站点城市之间快递运输数量，以及当日所有“发货-收货”站点城市之间的总快递运输数量，并就个别城市进行深入探究。

问题三：在道路遭到突发情况不可行的情况下，根据快递公司记录的相关快递运输数据，建立数学模型，预测 2023 年 4 月 28 日和 2023 年 4 月 29 日可正常“发货-收货”的站点城市对，并判断指定的站点城市对是否能正常发货以及预测对应的快递运输数量。

问题四：给定所有站点城市间的铁路运输网络，并附加一定的条件限制。建立数学模型，给出该快递公司成本最低的运输方案。利用快递公司给定的相关数据，计算该公司 2023 年 4 月 23—27 日每日的最低运输成本。

问题五：假定快递需求由两部分组成，一部分为固定需求，另一部分为非固定需求。本问题旨在建立相对应的数学模型，求出对应的固定需求常数以及非固定需求的均值与标准差。

二、问题分析

2.1 问题一的分析

问题一要求我们给相关城市排序，因此需要考虑多个指标。首先要对数据进行处理，运用 Python 统计各个城市在 2018 年 4 月 19 日—2019 年 4 月 17 日间各自的总发货量与收货量；然后将二者加在一起，得到各个月份经过某城市的快递数量，之后进行逐月差分，即当月的数据减去前一月的数据的差除以前一月的数据得到当月的增长/下降趋势，然后分别对各个城市进行加总取月份的平均值，可用作评价指标；至于相关性，则是采取了 Python 的 PageRank 算法计算各个城市的中心度；最后，将数据进行汇总，然后相关数据处理，采用 topsis 模型，得到最终的评价结果。

2.2 问题二分析

问题二要求预测 2019 年 4 月 18 日和 2019 年 4 月 19 日各“发货-收货”站点城市之间快递运输数量，以及当日所有“发货-收货”站点城市之间的总快递运输数量。先运用 Python 提取历史各个城市各天的快递流通量（也就是发货量+收货量），之后运用 SPSS，进行时间序列预测。

2.3 问题三分析

本问题主要是探究，在突发情况导致部分节点有时不可达的情况下，预测 2023 年 4 月 28 日和 2023 年 4 月 29 日可正常“发货-收货”的站点城市对。只需要在问题二的基础上稍作改动即可，构建一矩阵，将某 i 城市通往某 j 城有货物赋值 1，没有赋值 0 进行预测，0 即为不可通行的，对这一矩阵数据进行训练预测，然后对于相对应可通行的数据进行预测就好。

2.4 问题四分析

本问题主要探究，在给出了所有站点城市间的铁路运输网络的情况下，结合快递公司的数据，选出当天运输成本最小的路径。首先我们需要对运输网络构建图模型，然后提取各天数据，遍历各天数据，同时将发货城市与收货城市分别作为出发点和终点，代入到带有限制条件的 Bellman-ford 函数中进行最短路探索，找出最短路花费并加到当天的费用中，不断循环，直到求出各天的最小费用。

2.5 问题五分析

本问题主要是为了求出各个季度的固定需求常数以及非固定需求均值以及标准差。对第一问，运用 3sigma 准则求出固定需求常数，对于第二问，则是在原有数据的基础上减去相对应的固定需求常数，并运用高斯核函数作为核密度估计，求出目标城市对的运输快递数量的概率密度，再计算向对应的值。

三、问题一模型的建立与求解

3.1 数据准备

运用附录 1 程序，对于收货量与发货量进行汇总，得到评价指标表对应的数据，算法具体如下：

第一步，先读入相关数据，并创建两个字典，用以存储相对应的发货量与收货量。

第二步，从 Excel 第一行读取数据一直读到最后一行，对于相关城市进行信息存储与读取。

第三步，输出相关数据。

运用附录 2 程序，计算平均趋势值，得到评价指标表对应的数据，算法具体如下：

第一步，读入相关数据，控制相关变量，对于年份进行备份，后筛选去重。

第二步，按照年份索引，对本月数据与下月数据进行计算，并计算其增加/下降趋势。

第三步，存储数据。

最后，运用 Excel 整理数据，得到各个城市的平均趋势值。

之后运用 Excel，对于以上所得数据进行汇总，可得到以下表格。

城市	收货量	发货量	平均趋势值	度中心值
A	51708	31276	0.198140788	0.038213289
B	15624	12458	0.135426576	0.005951913
C	13228	18361	0.067510929	0.011222747
D	52401	31135	0.176568864	0.027753074
E	20055	14643	0.191072915	0.02040195
G	273737	287248	0.192867229	0.135195677
H	47155	45866	1.356805592	0.025749436
I	50855	45100	0.137110585	0.040817905
J	98557	99173	0.160845704	0.059067084
K	71003	74841	0.180551835	0.038401495
L	311289	296846	0.175266169	0.139805271
M	47609	52041	0.167139224	0.031483174
N	44445	62359	0.162471438	0.028934345

O	87091	64986	0.183846933	0.051445718
P	13357	15419	0.170154732	0.009828413
Q	81076	60654	0.202216822	0.060234913
R	91333	109685	0.194074717	0.050282953
S	37140	56714	0.024067543	0.022724309
T	0	32819	0.148375482	0.004166667
U	24339	40548	0.165669725	0.01745005
V	192823	164403	0.190497639	0.095433879
W	77336	69505	0.200734732	0.033799382
X	84303	89437	0.472376235	0.032487126
Y	41811	52758	0.207451325	0.019149231

表 1 评价指标表

3.2 模型构建+求解

假设有 n 个要评价的对象，m 个评价指标，算法具体如下：

首先第一步，需要对将原始矩阵正向化处理（将所有的指标类型统一转换为极大型指标），但是由于的话此处的值已经全部为极大值，故而这可以忽略不计。

第二步，需要对矩阵进行正向化矩阵标准化，假设有矩阵如下：

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

那么，对标准化后的矩阵记为 Z，对 Z 中的每一个元素：

$$Z = x_{ij} / \sqrt{\sum_{i=1}^{n} x_{ij}^2}$$

第三步，假设有 n 个要评价的对象，m 个评价指标的标准化矩阵：

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

定义最大值：

$$Z^+ = (Z_1^+, Z_2^+, \dots, Z_m^+)$$
$$= (\max\{z_{11}, z_{21}, \dots, z_{n1}\}, \max\{z_{12}, z_{22}, \dots, z_{n2}\}, \dots, \max\{z_{1m}, z_{2m}, \dots, z_{nm}\})$$

定义最小值：

$$Z^- = (Z_1^-, Z_2^-, \dots, Z_m^-)$$
$$= (\min\{z_{11}, z_{21}, \dots, z_{n1}\}, \min\{z_{12}, z_{22}, \dots, z_{n2}\}, \dots, \min\{z_{1m}, z_{2m}, \dots, z_{nm}\})$$

定义第i个评价对象与最大值的距离为：

$$D^+ = \sqrt{\sum_{j=1}^{m} (Z_j^+ - z_{ij})^2}$$

定义第i个评价对象与最小值的距离为：

$$D^- = \sqrt{\sum_{j=1}^{m}(Z_j^- - z_{ij})^2}$$

那么，我们可以计算出第$i(i = 1,2,\ldots.,n)$个评价对象的得分：

$$S_i = D_i^-/(D_i^- + D_i^+)$$

很明显$0 \le S_i \le 1$，并且S_i越大D_i^+越大，即越接近最大值。

最后，再将结果输出到以下表中，并进行可视化处理（而 topsis 对应的程序在附录 4 中）：

城市	得分
L	0.112538404
G	0.109771886
H	0.098769946
V	0.083292557
X	0.057179821
J	0.050810618
R	0.049946965
Q	0.045453059
O	0.043098473
K	0.037566029
W	0.037480787
A	0.030482446
I	0.029721764
N	0.028080986
M	0.028059251
Y	0.026303433
D	0.025976056
S	0.020031296
E	0.019352998
U	0.019251585
P	0.014771335
T	0.013205452
B	0.011553402
C	0.007301449

表 2 topsis 评分表

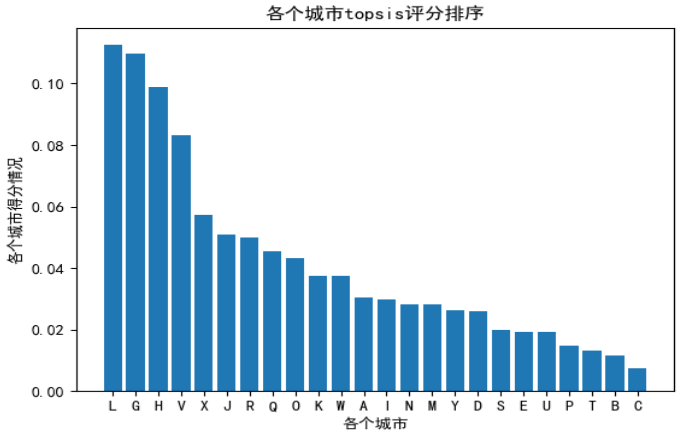


图 1 topsis 可视化图

由表和图的数据可以知道，排前五的城市如下：

排序	1	2	3	4	5
城市名称	L	G	H	V	X

表 3 问题一结果

四、问题二模型的建立与求解

4.1 数据准备

而至于时间序列数据，则要先运用附录 5 程序，提取目标所求各天城市间数据，然后再建立时间序列，由于数据太长，不在此进行展示，随后对其进行差分或者对数变换，以消除数据的季节性和趋势性。

4.2 模型构建+模型求解

在本研究中，选用 SPSSPRO 作为分析工具，该软件可以进行参数自动寻优。可以自动判断是运用，自回归（AR）、移动平均（MA）或者说是 ARIMA 模型，

而对于时间序列预测的处理原理，如下所示：

首先在对数据进行处理之后，根据自相关图和偏自相关图，确定最佳的 ARIMA 模型参数，包括 p 和 q 值，通过模型的训练和评估，确定最终的 ARIMA（p，d，q）模型，并进行预测和分析，由于数据过多，我们采取一个作为例子，来说明如何选用相应的模型参数。而例子的参数说明以及预测结果如下，以下例子为 MU 城市。

对时间序列数据实行 ADF 检验，得到以下表格。

ADF 检验表

变量	差分阶数	t	P	AIC	临界值		
					1%	5%	10%
总快递运输数量	0	-6.35	0.000***	1440.615	-3.474	-2.88	-2.577
	1	-6.194	0.000***	1452.834	-3.476	-2.882	-2.577
	2	-10.5	0.000***	1473.546	-3.473	-2.88	-2.577

表 4 ADF 检验表

注：***、**、*分别代表 1%、5%、10%的显著性水平

而我们可以清楚地看到，差分为 0 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。

而最终最终差分数据自相关图与偏自相关图如下：

图 2 自相关图

图 3 偏自相关图

由上图可见，自相关图与偏相关图均呈现拖尾，最显著的阶数都在 1 阶，故应当采用 ARIMA 模型，p,d,q 取值应当为(1,0,1)，而机器自动寻优，所得参数相同，而得到的相对应的检验参数表如下：

ARIMA 模型（1,0,1）检验表

项	符号	值
样本数量	Df Residuals	159
	N	162
Q 统计量	Q6(P 值)	0.009(0.926)
	Q12(P 值)	1.284(0.973)
	Q18(P 值)	18.53(0.101)
	Q24(P 值)	25.341(0.116)
	Q30(P 值)	38.915(0.028**)
信息准则	AIC	1591.823
	BIC	1604.174
拟合优度	R^2	0.726

表 5 ARIMA 模型（1,0,1）检验表

上表格展示本次模型检验结果，包括样本数、自由度、Q 统计量和信息准则模型的拟合优度。

根据 Q 统计量的 P 值（P 值大于 0.1 为白噪声）对模型白噪声进行检验，可以看到，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 R^2 代表时间序列的拟合程度，此处 R^2为 0.726，模型表现良好，基本符合预测要求。

经过运算，可得到如下模型参数表：

模型参数表

	系数	标准差	t	P>|t|	0.025	0.975
常数	115.333	12.565	9.179	0	90.706	139.961
ar.L1.总快递运输数量	0.722	0.062	11.685	0	0.601	0.844
ma.L1.总快递运输数量	0.413	0.083	4.959	0	0.25	0.576

表 6 ARIMA 模型参数表

最终模型为：y(t)=115.333+0.722*y(t-1)+0.413*ε(t-1)，绘制得到如下时间序列图像：

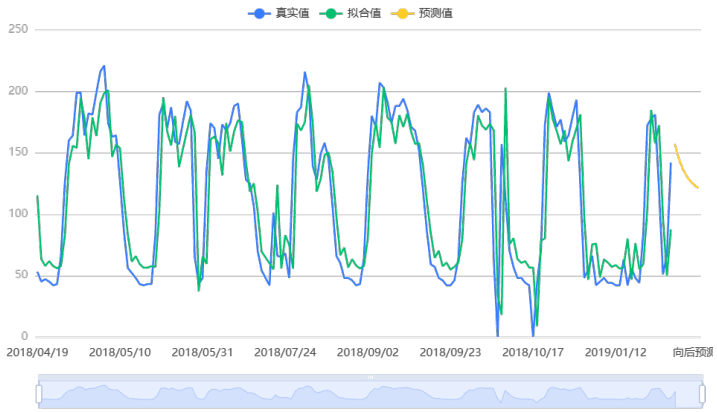


图 4 时间序列拟合结果图

由图像与相关系数 R 方，可得预测结果良好，而预测值如下：

预测值	
阶数（时间）	预测结果
1	157.10289649323198
2	145.50807255741333
3	137.13185018553494
4	131.08077932740704
5	126.70942229980301
6	123.55150815356293
7	121.27019786660861

表 7 预测值表

而阶数则是向后预测的时间，阶数 1 即为 2019 年 4 月 18 日，阶数 2 为 2019 年 4 月 19 日，即为我们所要求的。

而在对 MU 城市进行一次求解，重复应用 SPSS 经历以上过程，可以得到如下答案表格：

日期	“发货-收货”城市之间的快递运输数量		所有“发货-收货”城市之间的总快递运输数量
2019 年 4 月 18 日	M-U	157	10040
	Q-V	48	
	K-L	70	
	G-V	583	
2019 年 4 月 19 日	V-G	507	9533
	A-Q	129	
	D-A	50	
	L-K	294	

表 8 最终结果表

五、问题三模型的建立与求解

5.1 数据准备

利用附件 x 中的程序对附件 2 中的数据进行处理，得到一个指示每个线路在每个日期运输的二维数组 d，以及每个线路在每个日期的货量的二维数组 d1。

对于每个线路，将其货量时间序列数据按照一定的时间窗口划分为训练集和测试

集，并对数据进行归一化处理。基于训练集建立一个 LSTM 神经网络模型，而后基于模型预测，将预测结果存入 Excel 表格中作为项目报告的输出。

5.2 模型建立与求解

对于该问题，建立的 LSTM 模型，LSTM 的输入门 i, 遗忘门 f 和输出门 o，以及单元状态 Ct 的更新公式如下：

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$\boldsymbol{C_t = f_t * C_{t-1} + i_t * tanh(W_C[h_{t-1}, x_t] + b_C)}$$

$$h_t = o_t * \tanh(C_t)$$

以上公式中，x_t是输入向量，h_t 是 LSTM 的隐状态向量，W 和 b 是权重和偏移量，σ 和 tanh 分别是 sigmoid 函数和 tanh 函数。对于具体实现过程，首先对每个线路的历史数据进行预处理，将数据按照是否工作日进行表示，实际运载数量使用 0 填充没有数据的部分。

然后为每个线路建立一个 LSTM 模型，使用历史数据进行训练。训练时需要将数据进行归一化处理，以便提高模型的鲁棒性。对于每一个数据点，LSTM 模型将其与前面的数据（前几个历史数据）一起作为输入，输出该数据点的预测值。

每个线路的预测值与是否开通该线路的信息相乘，得到该线路的预测货运量，即为该线路 28 号和 29 号的货运量预测值。最后将所有线路的预测结果存入 Excel 表格中作为第三问的输出。

具体而言，该模型的实现过程包括以下步骤：

1. 读入 Excel 表格数据，提取出所有线路编号和日期。
2. 将数据进行预处理，使用 0 和 1 表示每个线路在每天是否有服务，以及每天的实际货运量。
3. 为每个线路建立一个 LSTM 模型，使用历史数据进行训练，训练时需要将数据进行归一化处理。
4. 使用训练好的 LSTM 模型进行预测，并将预测结果与是否开通该线路的信息相乘，得到每个线路的预测货运量。
5. 将所有线路的预测结果存入 Excel 表格中，作为第三问的输出。

此外，在整个过程中，还需要对模型进行评估，以确定其预测能力和泛化能力。评估可以使用多种指标，例如均方误差（Mean Squared Error，MSE）、平均绝对误差（Mean Absolute Error，MAE）和 R 方系数等。具体而言，在该项目中，利用 MSE 和 R 方系数对模型进行评估。MSE 是用来评估模型预测结果和实际值之间的误差大小的指标，计算公式如下：

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

其中，n 为样本个数， y_i为实际值，$\hat{y}_i$为预测值。MSE 越小，代表模型的预测能力越好。

R 方系数是用来衡量模型对样本数据的拟合程度的指标，计算公式如下：

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(yi - \hat{y}_i)^2}{\sum_{i=1}^{n}(yi - \bar{y})^2}$$

其中，$\bar{y}$ 为实际值的平均值。R 方系数越接近 1，代表模型对样本数据的拟合程度越好。

对于该项目，使用历史数据进行训练，然后使用训练好的 LSTM 模型对测试数据进行预测，并计算 MSE 和 R 方系数。通过不断调整 LSTM 模型的超参数，例如 LSTM 层的数量、神经元数量、Epoch 等，可以提高模型的预测能力和泛化能力，使其在新的数据上具有更好的表现。

而以上思路即为附录 6 程序的思路，其结果如下（只输出问题要求的结果，其它城市的能否正常发货及其预测情况，则跟附录 6 程序放在一起）：

日期	“发货-收货”站点城市对	是否能正常发货(填写“是”或“否”)	快递运输数量
2023 年 4 月 28 日	I-S	否	X
	M-G	是	44
	S-Q	是	99
	V-A	是	46
	Y-L	是	36
2023 年 4 月 29 日	D-R	否	X
	J-K	是	210
	Q-O	否	X
	U-O	否	X
	Y-W	否	X

表 9 最终结果表（X 表示不存在）

六、问题四模型的建立与求解

6.1 数据准备

首先运用 Python，将该公司 2023 年 4 月 23—27 日各日的数据提取出来，分别命名为“….日数据.xlsx”，方便后续构图以及处理，数据较多不在此进行展示。

之后，逐步读入每日的数据，对于边数以及结点分别进行标号，固定都从 0 开始，分日进行运算，读入一次处理一次。

而在此之前，需要先建立好铁路模型，在这里我们运用附录 7 的程序，运用 Python 的 networkx 库，构建了下图模型（由于运用了不同的图像生成方式，与原图有所区别）：

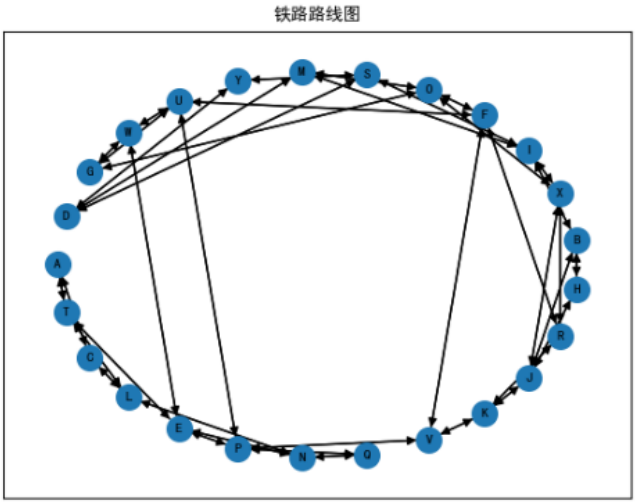


图 5 铁路路线图

6.2 模型构建与求解

假设当天总共有 n 个结点，有 wegnum 条路径，beg 为开始节点，end 为结束节点。

本研究采用 Bellman-ford 算法构建最短路径模型，该算法通过不断地松弛操作，计算起点到各节点的最短距离，从而得到最短路径。首先构建一个函数，内部包含算法原型，具体实现过程如下：

首先，对起点的距离赋值为 0，其他节点的距离值设为无穷大，即为：

$$dist_i = INF(i = 1,2,\dots,n)$$

$$dist_{beg} = 0$$

第二步然后不断进行松弛操作，即比较当前节点通过经过目前节点到其他节点的距离和起点到其他节点的距离之和，更新当前节点的最短路径和距离值。

$$dist_i^{(k+1)} = min_j\left\{dist_j^{(k)} + w_{j,i}\right\},$$

$$i = 1,2,\dots,wegnum-1, j = 1,2,\dots,wegnum-1$$

第三步，如果更新了当前节点的最短路径和距离，同时也需要更新当前节点到下一个节点的距离和路径值。如此反复进行松弛操作，直到到达目标节点，从而得到最短路径。而$dist_{end}$即为目标值。

而对于主函数，则是由以下几步构成：

第一步，运用 for 循环语句，按天读入，将其出发点、结束点以及快递进行分开存储。

第二步，对于这一天的每个点进行遍历，将出发点和结束点以及待发快递重量传入上述函数中

第三步，打印相关结果，回到第一步继续循环直到结束。

运用附录 7 的程序,可以得到以下结果值：

日期	最低运输成本
2023 年 4 月 23 日	977
2023 年 4 月 24 日	1182
2023 年 4 月 25 日	1070
2023 年 4 月 26 日	951
2023 年 4 月 27 日	1008

表 10 最低运输成本表

七、问题五模型的建立与求解

7.1 数据准备

首先，先将附件 3 中的数据提取出来，划分为 2022 年 7—9 月以及 2023 年 1—3 月的数据，分别命名为“22 年季度数据.xlsx”和“23 年季度数据.xlsx”

7.2 模型构建与求解

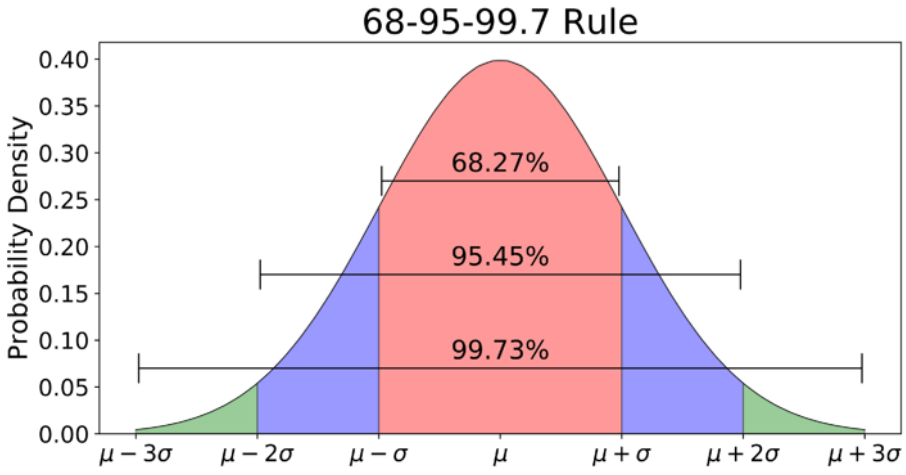


图 6 3 Sigma 示意图

根据 3sigma 准则，通常我们认为，在满足一定条件下，如果一个事件在平均值附近的概率非常高，在 3 个标准差内的数据概率为约 68.27%，在 6 个标准差内的数据概率为约 99.73%，而对于不同城市的固定需求而言，其通常符合正态分布。并且由于我们是多个城市多天的数据，可以看成是从多个城市或者是同一城市多天中抽样，故而满足中心极限定理，也满足正态分布要求。于是我们取均值减去两个标准差的量为固定需求常数（只有 5%是的极为特殊的）。

而对于非固定需求常数，我们需要用对应的原数据减去对应城市对的固定需求常数从而得到各个城市各天的非固定需求值。随后运用核密度估计法，求得整个数据集的概率密度估计。

而本问题总共有两个程序，分别在附录 8 和附录 9 中，具体算法如下：

第一步，读入 2022 年份或者 2023 年份数据（分开处理），然后从中挑出对应的程序对。

第二步，读取相应的时间序列数据，然后将时间与数据分开来，计算均值以及最小值，计算当前城市对的固定需求常数。

第三步，让原数据所有值减去对应的固定需求常数，计算其标准差以及均值，

输出结果，导出表格。

第四步，定义好变量中想要得到的城市对的二维数组

第五步，求得其非固定需求的值，随后采用高斯核的非参数估计方法，对其进行拟合。同时指定评估点，对其进行评测

第六步，绘制目标城市对的直方图以及折线图，输出结果，而绘制的图形如下：

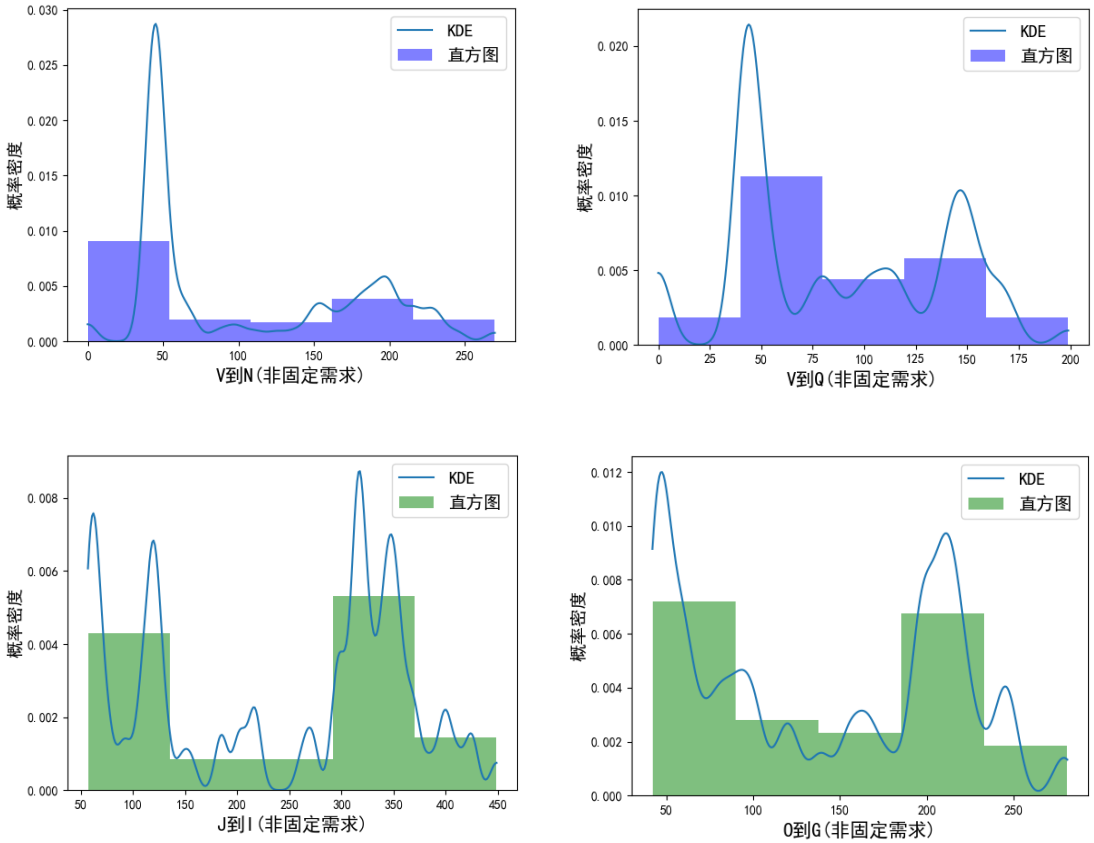


图 7 3 对应城市非固定需求拟合组图

以上便是算法执行的流程，相关年份的固定需求常数由于过长，故而放在附录 10 中，而对应要求月份的相关值如下所示：

季度	2022 年第三季度 (7—9 月)		2023 年第一季度 (1—3 月)	
“发货-收货”站点城市对	V-N	V-Q	J-I	O-G
固定需求常数	0	0	4	0
非固定需求均值	135	99.5	252.7	161.5
非固定需求标准差	0.0036	0.0048	0.0025	0.004
固定需求常数总和	0		4	
非固定需求均值总和	234.5		414.2	

非固定需求标准差总和	0.0084	0.0065

表 11 相关需求系数表

八、模型评价与推广

8.1 模型的优点

对于问题一的模型，主要有两个优点，一是采用 pagerank 算法计算相关性，通过 pagerank 算法计算各个城市的中心度，得到城市之间的相关性。帮助识别具有最高中心度的城市，具有一定的科学性和可靠性。二是采用 topsis 模型，得出评价结果，该模型是一种常用的综合评价模型，它可以将多个因素综合起来，得出最终的评价结果。该模型在管理决策中应用广泛，可以对城市之间的快递服务进行全面综合评价，并得到最优方案，具有一定的实用价值。

对于问题二的模型，通过 SPSS 进行时间序列预测，可以自动考虑时间序列数据的趋势、季节性、周期性和随机性，更全面地反映时间序列数据的变化趋势；同时可以分析时间序列数据中的长期和短期波动，具有较高的准确性，并且模型的建立和应用相对较简单，有现成的工具包可以直接使用，操作方便。

对于问题三的模型，使用矩阵来表示能够方便而直观地观察和把握网络中各个节点和路径的关系，有利于运输网络的优化和管理。且这个模型是基于机器学习方法来设计的，能够自动学习数据的特征和规律，并且可以不断地优化和改进。这样可以提高预测的准确性和可靠性。

对于问题四的模型，其能够结合快递公司的数据来计算当天不同站点之间的最小成本路径，实现较为准确地成本计算，有利于快递公司的成本控制，在确定铁路运输网络中各项运输路线的最小成本路径方面具有一定的应用价值

对于问题五的模型，使用 3sigma 准则求解固定需求常数的方法非常实用和稳健。通过该方法可以有效地剔除异常值和季节性变动等因素对需求的影响，从而得到较为准确的固定需求常数。同时采用高斯核函数作为核密度估计的方法，可以更好地反映出数据分布的特征，提高分析结果的可靠性和准确性

8.2 模型的改进

对于问题一：判断一个站点的货运可达性，可以采用类似网络中的连通性指标，即通过计算该站点能到达的其他站点数量来衡量。快递运输网络的密度对构建一个高效且快速的物流系统至关重要，因此可以通过计算网络中实际边数和可能边数的比值来衡量其网络密度。除了计算节点的度中心度和众数中心度之外，可以考虑引入直接度中心度和中介度中心度这两个指标，来更好地评价站点的重要程度。

直接度中心度指的是一个节点直接与其他节点相连的程度，中介度中心度指的是一个节点在所有其他节点间的中间位置程度。这两个指标可以更全面地评价站点在快递运输网络中的节点重要性。

8.3 模型的推广

本文建立的模型可以对快递运输网络建模，对快递运输网络进行评价，预测未来的快递运输量，计算铁路运输成本，求解最大流问题，预测非固定需求等，能够为物流领域提供有价值的科学方法和技术手段。

此外，本模型可以作为引导物流领域管理者和决策者采取更科学的方法和技术手段解决物流问题，提高物流系统的效率和质量。

将本模型和方法推广到其他领域，如公共安全、城市交通管理、能源供应等领

域，可以为解决其他大规模的复杂问题提供科学的思路和方法。

本模型采用了多种数据分析和建模技术，如时间序列分析、网络流算法等，可以为数据科学和人工智能领域的研究提供参考和借鉴，促进数据科学和人工智能技术的发展。

附录：

附录 1：收货量+发货量汇总程序.py

```
import xlrd
import numpy as np
from openpyxl import Workbook

wb=xlrd.open_workbook("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\附件 1.xlsx")
ws=wb.sheet_by_index(0)
start,end,num=[],[],[]

##去掉头部的标签
start=ws.col_values(1)
end=ws.col_values(2)
num=ws.col_values(3)
start.pop(0);end.pop(0);num.pop(0)


##deli 为发货量，rece 为收货量
deli=dict()
rece=dict()
for i in range(0,len(start)):
    ##以下两行代码的意思是如果城市号不在对应的字典的键中，那么就说明是第一次读到，进行赋值，否则加上即可
    if(start[i] not in deli.keys()):
        deli[start[i]]=num[i]
    else :
        deli[start[i]]+=num[i]
    if(end[i] not in rece.keys()):
        rece[end[i]]=num[i]
    else :
        rece[end[i]]+=num[i]

##以下存储发货量
res1=Workbook()
ws1=res1.active

##城市标签行赋值
ws1.cell(1,1).value="城市"
```

```
ws1.cell(1,2).value="发货量"

##以下逐行进行赋值
j=2
for name,tot in zip(list(deli.keys()),list(deli.values())):
    ws1.cell(j,1).value=name
    ws1.cell(j,2).value=tot
    j+=1
res1.save("发货量总结表.xlsx")

##以下为存储收货量
res2=Workbook()
ws2=res2.active
ws2.cell(1,1).value="城市"
ws2.cell(1,2).value="收货量"
j=2
for name,tot in zip(list(rece.keys()),list(rece.values())):
    ws2.cell(j,1).value=name
    ws2.cell(j,2).value=tot
    j+=1
res2.save("收货量总结表.xlsx")
```

附录 2：趋势汇总程序.py

```
import pandas as pd
import numpy as np
from openpyxl import Workbook
data=pd.read_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\附件1.xlsx",parse_dates=["日期(年/月/日) (Date Y/M/D)"])

month=data['日期(年/月/日) (Date Y/M/D)'].dt.strftime('%Y-%m')
month_ori=month
##下列代码用来计算不重复的时间列表
month=month.drop_duplicates()

month.index=range(0,10)

nowline,nextline=0,0
needdata=Workbook()
des=needdata.active

##代表对其进行从头到尾遍历
for i in range(0,9):
    ##nowcity 表示现在城市的快递收发货情况，nextcity 表示下一个时间段城市的收发货情况，change 表示相比于上个月的差异平均差异
```

```
    nowcity,nextcity,change=dict(),dict(),dict()
    ##下列两个变量分别用来判断该城市有没有进入过上述字典中
    jud1,jud2=dict(),dict()
    ##nowline 表示现在遍历到的年份，只要符合现在的就可以继续往下遍历，一直遍历到下一年终止
    while(month_ori.iloc[nowline]==month[i]):
        bgi,end,num=data.iloc[nowline,1],data.iloc[nowline,2],data.iloc[nowline,3]
        if(bgi not in jud1.keys()):
            nowcity[bgi]=num
            jud1[bgi]=1
        else :
            nowcity[bgi]+=num
        if(end not in jud1.keys()):
            nowcity[end]=num
            jud1[end]=1
        else :
            nowcity[end]+=num
        nowline+=1

    ##然后遍历下一年的，同上，nextline 表示其遍历到的位置
    nextline=nowline
    while(month_ori.iloc[nextline]==month[i+1]):
        bgi,end,num=data.iloc[nextline,1],data.iloc[nextline,2],data.iloc[nextline,3]
        if(bgi not in jud2.keys()):
            nextcity[bgi]=num
            jud2[bgi]=1
        else :
            nextcity[bgi]+=num
        if(end not in jud2.keys()):
            nextcity[end]=num
            jud2[end]=1
        else :
            nextcity[end]+=num
        nextline+=1
        if(nextline==16962):       ##表明这是数据的最后一行
            break

    same=list(set(list(nextcity.keys())+list(nowcity.keys())))##表明当前月份出现的城市列

    temp=2
    for j in sorted(same):
        if((j in nowcity.keys())&(j in nextcity.keys())):  ##如果都出现了，就计算其增长率
```

```
                change[j]=(nextcity[j]-nowcity[j])/nowcity[j]       ##计算某月的增加或下降趋势，后面两个是为了避免 0
            elif(j not in nowcity.keys()):      ##如果上一年没有出现过，就赋值为 1，否则为 0
                change[j]=1
            else :
                change[j]=-1
            des.cell(1+i*2,temp).value=j
            temp+=1

        des.cell(2+i*2,1).value=month.iloc[i+1]
        temp=1
        for j in sorted(same):
            des.cell(2+i*2,temp+1).value=change[j]
            temp+=1
needdata.save("趋势数据.xlsx")
```

附录 3：各个节点的中心度计算程序.py

```
import xlrd
from openpyxl import Workbook
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

wb=xlrd.open_workbook("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\附件 1.xlsx")
ws=wb.sheet_by_index(0)
left,right,num=[],[],[]
left=ws.col_values(1)
right=ws.col_values(2)
num=ws.col_values(3)
left.pop(0);right.pop(0);num.pop(0)
##上述代码为读入数据，然后去掉标签


jud=dict()

g=nx.DiGraph()##创建一个图

for i in range(0,len(num)):
    ##以下则是判断相应的点或边是否在图中，没有则加入
    if(left[i] not in jud.keys()):
        jud[left[i]]=1
        g.add_node(left[i])
    if(right[i] not in jud.keys()):
```

```
            jud[left[i]]=1
            g.add_node(right[i])
        if(g.has_edge(left[i],right[i])==0):
            g.add_edge(left[i],right[i],weight=num[i])
        else :

g.edges[left[i],right[i]].update({"weight":num[i]+g.get_edge_data(left[i],right[i])["weight
"]})

nx.draw_networkx(g,pos=nx.shell_layout(g),node_size=200,node_shape='o',width=1,styl
e='solid',font_size=8)##进行可视化
plt.show()

des=Workbook()
ws=des.active
ws.cell(1,1).value="节点"
ws.cell(1,2).value="度中心值排序"
j=2

np=nx.pagerank(g,alpha=0.9)
top_k = sorted(np.items(), key=lambda x: x[1], reverse=True)
for node, value in top_k:
        ws.cell(j,1).value=node
        ws.cell(j,2).value=value
        j+=1
des.save("中心值的值.xlsx")
```

附录 4：topisis 评价.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from openpyxl import Workbook
plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['SimHei']

df=pd.read_excel("C:\\Users\\Administrator\\Desktop\\vs工作界面\\各个指标整理.xlsx")
df=df.dropna(axis=0,how="all")
data=df.values
##提取城市标签
X=data[:,:1]
##提取城市评价指标
data=data[:,1:]

##进行标准化
```

```
data=data/np.sum(data*data,axis=0)**0.5
max_score=np.max(data,axis=0)
min_score=np.min(data,axis=0)
##求出其对应与最大值与最小值的距离
max_dist=np.sum((max_score-data)*(max_score-data),axis=1)**0.5
min_dist=np.sum((min_score-data)*(min_score-data),axis=1)**0.5

##进行得分计算
final_score=(min_dist/(max_dist+min_dist))
final_score/=np.sum(final_score)

final_score=final_score.tolist()
X=X.tolist()

x=dict()
j=0
##将其变成一个字典，方便之后从大到小排序
for i in X:
    x[i[0]]=final_score[j]
    j+=1
res=sorted(x.items(),key=lambda x:x[1],reverse=True)
x,final_score=[],[]



##以下皆为输出+可视化
wb=Workbook()
ws=wb.active

ws.cell(1,1).value="城市"
ws.cell(1,2).value="得分"

for num,sco in res:
    x.append(num)
    final_score.append(sco)


for i in range(len(x)):
    ws.cell(i+2,1).value=x[i]
    ws.cell(i+2,2).value=final_score[i]
wb.save("topsis 评分表.xlsx")

ax=plt.bar(x=x,height=final_score)##绘制条形图
plt.title("各个城市 topsis 评分排序")
```

```
plt.xlabel("各个城市")
plt.ylabel("各个城市得分情况")
plt.show()
```

附录 5: 互动提取各天各城快递运输数量程序.py

```
import pandas as pd
import numpy as np
from openpyxl import Workbook
data=pd.read_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\附件1.xlsx",parse_dates=["日期(年/月/日) (Date Y/M/D)"])

tim=data['日期(年/月/日) (Date Y/M/D)'].dt.strftime("%Y/%m/%d")
tim_ori=tim
tim=tim.drop_duplicates()
tim=tim.values

b=input("请输入你要查找数据的路径起点：")
e=input("请输入你要查找数据的路径终点：")

nowline=0
needdata=Workbook()
des=needdata.active
daydata=dict()

for i in range(0,len(tim)):

    while(tim_ori.iloc[nowline]==tim[i]):
        bgi,end,num=data.iloc[nowline,1],data.iloc[nowline,2],data.iloc[nowline,3]
        if(bgi==b) & (end==e):
            if(tim[i] not in daydata.keys()):
                daydata[tim[i]]=num
            else :
                daydata[tim[i]]+=num
        nowline+=1
        if(nowline==16962):
            break

wb=Workbook()
ws=wb.active
ws.cell(1,1).value="时间"
ws.cell(1,2).value="总快递运输数量"
i=2
for temp1,temp2 in zip(list(daydata.keys()),list(daydata.values())):
    ws.cell(i,1).value=temp1
```

```
        ws.cell(i,2).value=temp2
        i+=1
wb.save("各天{}城到{}城快递运输数量提取收据.xlsx".format(b,e))
```

附录 6：求可达及其路径.py

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import datetime

data = pd.read_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\附件2.xlsx").values


##求出不重复的城市对
lines = np.unique(np.array([i[0]+i[1]for i in data[:,[1,2]]]))
lines = np.array([[i[0],i[1]] for i in lines])
date = np.unique(data[:,0])

d = np.zeros(shape=(lines.shape[0],date.shape[0]))
d1 = np.zeros(shape=(lines.shape[0],date.shape[0]))
for i in range(d.shape[0]):
        data1 = data[np.logical_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])]
        for j in range(d.shape[1]):
                data2 = data1[data1[:,0]==date[j]]
                if data2.shape[0]==0:
                        d[i,j]=0
                        d1[i,j]=0
                else:
                        d[i,j]=1
                        d1[i,j]=data2[0,-1]

all_pre1 = []
def create_dataset(data, look_back=1):
        X, Y = [], []
        for i in range(len(data) - look_back):
                X.append(data[i:(i + look_back), 0])
                Y.append(data[i + look_back, 0])
        return np.array(X), np.array(Y)
for i in range(lines.shape[0]):
        y = d[i]
        # 将数据划分为训练集和测试集
        train_size = int(len(y) * 0.8)
```

```
train, test = y[:train_size], y[train_size:]

# 数据归一化
scaler = MinMaxScaler()
train = scaler.fit_transform(train.reshape(-1, 1))
test = scaler.transform(test.reshape(-1, 1))

# 创建数据生成器
look_back = 7
X_train, y_train = create_dataset(train, look_back)
X_test, y_test = create_dataset(test, look_back)

# 重塑数据以适应 LSTM 模型的输入格式
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# 创建 LSTM 模型
model = Sequential()
model.add(LSTM(50, input_shape=(look_back, 1)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')

# 训练模型
model.fit(X_train, y_train, epochs=20, batch_size=1, verbose=1)

# 预测
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

# 将预测结果转换回原始尺度
train_predict = scaler.inverse_transform(train_predict)
y_train = scaler.inverse_transform(y_train.reshape(-1, 1))
test_predict = scaler.inverse_transform(test_predict)
y_test = scaler.inverse_transform(y_test.reshape(-1, 1))

# 计算预测准确性，例如使用均方误差（MSE）作为评估指标
train_mse = np.mean((train_predict - y_train) ** 2)
test_mse = np.mean((test_predict - y_test) ** 2)

# 计算预测日期与最后一个训练日期之间的天数
last_train_date = datetime.date(2019, 4, 17)
start_pred_date = datetime.date(2019, 4, 18)
end_pred_date = datetime.date(2019, 4, 20)
days_to_predict = (end_pred_date-start_pred_date).days
```

```
        # 使用训练数据的最后一部分来开始预测
        input_data = train[-look_back:]

        predictions = []

        # 预测每一天的货量
        for i in range(days_to_predict):
            input_data_reshaped = input_data.reshape(1, look_back, 1)
            pred = model.predict(input_data_reshaped)
            predictions.append(pred[0, 0])

            # 更新输入数据，用预测值替换最早的值
            input_data = np.roll(input_data, -1)
            input_data[-1] = pred

        # 将预测值转换回原始尺度
        predictions = scaler.inverse_transform(np.array(predictions).reshape(-1, 1))

        tdays = []
        # 打印预测结果
        for i, pred in enumerate(predictions, start=1):
            pred_date = start_pred_date + datetime.timedelta(days=i - 1)
            tdays += [pred[0]]
        all_pre1+=[tdays]
all_pre1 = np.array(all_pre1)

all_pre = []
def create_dataset(data, look_back=1):
        X, Y = [], []
        for i in range(len(data) - look_back):
            X.append(data[i:(i + look_back), 0])
            Y.append(data[i + look_back, 0])
        return np.array(X), np.array(Y)
for i in range(lines.shape[0]):
        y = d1[i]
        y[y==0]=y.mean()
        # 将数据划分为训练集和测试集
        train_size = int(len(y) * 0.8)
        train, test = y[:train_size], y[train_size:]

        # 数据归一化
        scaler = MinMaxScaler()
        train = scaler.fit_transform(train.reshape(-1, 1))
```

```
test = scaler.transform(test.reshape(-1, 1))

# 创建数据生成器

look_back = 7
X_train, y_train = create_dataset(train, look_back)
X_test, y_test = create_dataset(test, look_back)

# 重塑数据以适应 LSTM 模型的输入格式
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

# 创建 LSTM 模型
model = Sequential()
model.add(LSTM(50, input_shape=(look_back, 1)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')

# 训练模型
model.fit(X_train, y_train, epochs=20, batch_size=1, verbose=1)

# 预测
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

# 将预测结果转换回原始尺度
train_predict = scaler.inverse_transform(train_predict)
y_train = scaler.inverse_transform(y_train.reshape(-1, 1))
test_predict = scaler.inverse_transform(test_predict)
y_test = scaler.inverse_transform(y_test.reshape(-1, 1))

# 计算预测准确性，例如使用均方误差（MSE）作为评估指标
train_mse = np.mean((train_predict - y_train) ** 2)
test_mse = np.mean((test_predict - y_test) ** 2)

# 计算预测日期与最后一个训练日期之间的天数
last_train_date = datetime.date(2019, 4, 17)
start_pred_date = datetime.date(2019, 4, 18)
end_pred_date = datetime.date(2019, 4, 20)
days_to_predict = (end_pred_date-start_pred_date).days

# 使用训练数据的最后一部分来开始预测
input_data = train[-look_back:]
```

```
        predictions = []

        # 预测每一天的货量
        for i in range(days_to_predict):
            input_data_reshaped = input_data.reshape(1, look_back, 1)
            pred = model.predict(input_data_reshaped)
            predictions.append(pred[0, 0])

            # 更新输入数据，用预测值替换最早的值
            input_data = np.roll(input_data, -1)
            input_data[-1] = pred

        # 将预测值转换回原始尺度
        predictions = scaler.inverse_transform(np.array(predictions).reshape(-1, 1))

        tdays = []
        # 打印预测结果
        for i, pred in enumerate(predictions, start=1):
            pred_date = start_pred_date + datetime.timedelta(days=i - 1)
            tdays += [pred[0]]
        all_pre+=[tdays]

all_pre = np.array(all_pre)
all_pre1[all_pre1>0.5]=1
all_pre1[all_pre1<=0.5]=0
p2 = all_pre*all_pre1


temp=pd.DataFrame(np.c_["1",lines,all_pre1,p2],columns=["起点","终点","28 号是否开通","29 号是否开通","28 号预测值","29 号预测值"])
temp.to_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\第三问结果.xlsx")
```

预测结果如下（1 表示开通，0 表示未开通）：

起点	终点	28 号是否开通	29 号是否开通	28 号预测值
A	O	0.0	0.0	0.0
A	Q	1.0	1.0	63.801426
C	M	1.0	1.0	59.20709
C	N	1.0	1.0	41.939297
C	U	0.0	0.0	0.0
C	V	1.0	1.0	50.05312
D	A	0.0	0.0	0.0
D	E	0.0	0.0	0.0
D	L	1.0	1.0	40.27782
D	R	0.0	0.0	0.0

E	F	0.0	0.0	0.0
E	I	0.0	0.0	0.0
G	L	1.0	1.0	145.2668
G	N	1.0	1.0	44.77389
G	O	0.0	0.0	0.0
G	Q	1.0	1.0	60.370255
G	R	1.0	1.0	64.37284
G	V	1.0	1.0	614.3894
G	X	1.0	1.0	50.60406
H	J	1.0	1.0	71.468765
H	K	1.0	1.0	59.009796
H	L	1.0	1.0	50.36455
I	E	0.0	0.0	0.0
I	F	0.0	0.0	0.0
I	J	0.0	0.0	0.0
I	S	0.0	0.0	0.0
J	H	1.0	1.0	45.556595
J	I	0.0	0.0	0.0
J	K	1.0	1.0	229.14053
J	L	1.0	1.0	105.05712
K	H	1.0	1.0	157.5657
K	J	1.0	1.0	49.22154
K	L	1.0	1.0	192.38072
L	D	1.0	1.0	53.90667
L	G	1.0	1.0	72.8111
L	H	1.0	1.0	53.832863
L	J	1.0	1.0	226.62184
L	K	1.0	1.0	73.30573
L	O	0.0	0.0	0.0
L	P	0.0	0.0	0.0
L	R	1.0	1.0	97.96739
L	W	0.0	0.0	0.0
L	X	1.0	1.0	314.9342
M	C	0.0	0.0	0.0
M	G	1.0	1.0	44.36205
M	N	1.0	1.0	46.74587
M	U	0.0	0.0	0.0
M	V	1.0	1.0	54.781918
N	G	1.0	1.0	48.86266
N	M	0.0	0.0	0.0
N	V	1.0	1.0	130.27242
O	G	0.0	0.0	0.0
O	Q	0.0	0.0	0.0

O	R	0.0	0.0	0.0
P	D	0.0	0.0	0.0
Q	A	1.0	1.0	56.036922
Q	M	0.0	0.0	0.0
Q	N	0.0	0.0	0.0
Q	O	0.0	0.0	0.0
Q	V	1.0	1.0	77.9601
R	D	1.0	1.0	48.006897
R	G	1.0	1.0	55.370132
R	L	1.0	1.0	84.83743
R	O	0.0	0.0	0.0
R	S	1.0	1.0	149.41547
S	D	1.0	1.0	62.07857
S	I	0.0	0.0	0.0
S	L	1.0	1.0	72.767914
S	Q	1.0	1.0	98.74985
S	R	1.0	1.0	59.3365
U	A	0.0	0.0	0.0
U	G	0.0	0.0	0.0
U	O	0.0	0.0	0.0
U	V	0.0	0.0	0.0
V	A	1.0	1.0	121.24717
V	C	1.0	1.0	88.54199
V	G	1.0	1.0	568.9954
V	M	1.0	1.0	42.345474
V	N	1.0	1.0	55.636276
V	Q	1.0	1.0	63.99519
W	L	0.0	0.0	0.0
W	X	0.0	0.0	0.0
W	Y	0.0	0.0	0.0
X	G	1.0	1.0	85.85304
X	L	1.0	1.0	338.79156
X	W	0.0	0.0	0.0
X	Y	1.0	1.0	286.15942
Y	L	1.0	1.0	36.0969
Y	W	0.0	0.0	0.0
Y	X	1.0	1.0	380.5135

附录 7：最短路计算代码.py

```
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd
from copy import deepcopy
plt.rcParams['font.sans-serif']=['SimHei']
```

```
plt.rcParams['axes.unicode_minus'] = False


fixeddata=pd.read_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\附件 3.xlsx")

##以下为提取相对应的数据
sta=fixeddata.loc[:,"起点  (Start)"]
end=fixeddata.loc[:,"终点  (End)"]
fixcos=fixeddata.loc[:,"固定成本  (Fixed cost)"]
dot_inf=dict()
ori=nx.DiGraph()


##以下表示给相对应的路径编码，并插入到图中
wegnum=0
weg=dict()
for i in range(len(sta)):
    weg[wegnum]=(sta[i],end[i])
    wegnum+=1
    dot_inf[(sta[i],end[i])]=fixcos[i]
    ori.add_edge(sta[i],end[i],weight=fixcos[i])

pos=nx.shell_layout(ori)##表示画图的格式，下面是画图相关代码
nx.draw_networkx(ori,pos=pos,node_size=200,node_shape='o',width=1,style='solid',font
_size=8)

node_labels=nx.get_node_attributes(ori,"des")
nx.draw_networkx_labels(ori,pos=pos,labels=node_labels)
plt.title("铁路路线图",fontsize=10)
plt.show()

nodes=[]
for i in ori.nodes.data():
    nodes.append(i[0])

node=dict()
letters=dict()
for i in range(0,len(nodes)):##表示对于点进行编码
    node[nodes[i]]=i
    letters[i]=nodes[i]

dist=dict()
pre=dict()
INF=9999999
```

```
def bellman_ford(bgi,end,weight):##最短路算法
    backup=dict()
    for i in range(0,len(node)):
        dist[i]=INF

    dist[bgi]=0
    for i in range(0,5):##限制次数为 5
        backup=deepcopy(dist)
        for j in range(0,wegnum): ##代表走 wegnum 条路
            wega=node[weg[j][0]]
            wegb=node[weg[j][1]]
            w=dot_inf[weg[j]]*(1+(weight/200)**3)##计算对应的成本
            if(dist[wegb]>backup[wega]+w):
                dist[wegb]=backup[wega]+w
    return dist[end]

for j in range(23,28):##代表逐天读取，并且进行逐天预测
    g=nx.DiGraph()
    data=pd.read_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\{}日数据.xlsx".format(str(j)))
    dci=data["发货城市 (Delivering city)"]
    rci=data["收货城市 (Receiving city)"]
    num=data["快递运输数量(件) (Express delivery quantity (PCS))"]


    today=0
    for i in range(0,len(dci)):
            start=node[dci[i]]
            end=node[rci[i]]
            today+=bellman_ford(start,end,num[i])
    print("4 月{}日的最低运输成本是{}".format(j,today))
```

附录 8：第五问 22 年代码.py

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KernelDensity
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams["font.sans-serif"]=["SimHei"]
mpl.rcParams["axes.unicode_minus"]=False

data = pd.read_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\22 年季度数据.xlsx").values
```

```
date = np.unique(data[:,0])        ##表示全部数据

lines = np.unique([i[0]+i[1]for i in data[:,[1,2]]])
lines = np.array([[i[0],i[1]] for i in lines])       ##表示总共的城市对

line_mean = []
line_min = []
xu = []
f_mean = []
f_std = []
mt = np.zeros(shape=(lines.shape[0],date.shape[0]))
print(mt)
for i in range(lines.shape[0]):
    d1 = data[np.logical_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])][:,-1]   ##d1 为相对应的月份的数据
    rq = data[np.logical_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])][:,0]    ##rq 为相对应的时间序列的数据
    line_mean+=[d1.mean()]        ##表示每个城市对的均值
    line_min +=[d1.min()]         ##表示每个城市对的最小值
    fixed =d1.mean()-2*d1.std()     ##表示 3sigma 定则
    if(fixed<0):
        fixed=0
    xu +=[fixed]
    notfixed = d1-(fixed)          ##代表各个城市对的非固定需求

    ##以下则为计算均值与标准差
    f_mean += [notfixed.mean()]
    f_std += [notfixed.std()]
    for j in range(rq.shape[0]):
        mt[i,date==rq[j]] = notfixed[j]
xu = np.array(xu)
print("22 年固定需求",np.c_["1",lines,np.round(xu)])
print("22 年非固定需求均值标准差\n",np.c_["1",lines,np.round(f_mean,4),np.round(f_std,4)])

temp=pd.DataFrame(np.c_["1",lines,np.round(xu)],columns=["发货城市","收货城市","固定需求"])
temp.to_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\固定需求(22 年).xlsx")

temp=pd.DataFrame(np.c_["1",lines,np.round(f_mean,2),np.round(f_std,2)],columns=["发货城市","收货城市","均值","标准差"])
temp.to_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\非固定需求(均值与标准差)(22 年).xlsx")
```

```
temp=pd.DataFrame(np.c_["1",lines,mt],columns=np.r_["0",["发货城市","收货城市"],date])
temp.to_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\非固定需求(22年).xlsx")



sc = np.array([["V","N"],["V","Q"]])
i=0
for i in range(2):
    d1 = data[np.logical_and(data[:,1]==sc[i,0],data[:,2]==sc[i,1])][:,-1]
    fixed =d1.mean()-2*d1.std()          ##求出固定需求常数，即 3sigma 准则
    if(fixed<0):
        fixed=0
    d2 = d1-fixed
    sample_data = d2.reshape(-1, 1)

    # KDE 模型实例化,高斯核函数是一个常用的核函数，可以对连续变量进行估计，带宽参数影响核函数的宽度，决定了估计出的概率密度函数的平滑度。
    kde = KernelDensity(kernel='gaussian', bandwidth=6).fit(sample_data)

    # 指定评估点（根据实际需求调整范围和间隔）
    ep = np.linspace(d2.min(), d2.max(), num=300).reshape(-1, 1)

    # 评估 KDE 模型
    ld = kde.score_samples(ep)
    ds = np.exp(ld)

    # 绘制 KDE 结果和直方图
    fig, ax = plt.subplots()
    ax.plot(ep, ds, label='KDE')
    ax.hist(sample_data, bins=5, density=True, alpha=0.5, color='blue', label='直方图')

    ax.set_xlabel('{}到{}(非固定需求)'.format(sc[i,0],sc[i,1]),fontsize=15)
    ax.set_ylabel('概率密度',fontsize=14)
    ax.legend(fontsize=14)
    plt.savefig('{}到{}(非固定需求)分布图.png'.format(sc[i,0],sc[i,1]))
    plt.show()
    temp=pd.DataFrame(np.c_["1",ep,ds[:,None]],columns=["非固定需求量","概率密度"])
    temp.to_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\{}_{}(非固定需求).xlsx".format(sc[i,0],sc[i,1]))
```

附录 9：第五问 23 年代码.py

```
    import numpy as np
```

```
import pandas as pd
from sklearn.neighbors import KernelDensity
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams["font.sans-serif"]=["SimHei"]
mpl.rcParams["axes.unicode_minus"]=False

data = pd.read_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\23 年季度数据.xlsx").values

date = np.unique(data[:,0])


lines = np.unique([i[0]+i[1]for i in data[:,[1,2]]])
lines = np.array([[i[0],i[1]] for i in lines])

line_mean = []
line_min = []
xu = []
f_mean = []
f_std = []
mt = np.zeros(shape=(lines.shape[0],date.shape[0]))
for i in range(lines.shape[0]):
    d1 = data[np.logical_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])][:,-1]
    rq = data[np.logical_and(data[:,1]==lines[i,0],data[:,2]==lines[i,1])][:,0]
    line_mean+=[d1.mean()]
    line_min +=[d1.min()]
    fixed =d1.mean()-2*d1.std()
    if fixed<0 :
        fixed=0
    xu +=[fixed]
    notfixed = d1-(fixed)          ##这一部分是非固定需求
    f_mean += [notfixed.mean()]
    f_std += [notfixed.std()]
    for j in range(rq.shape[0]):
        mt[i,date==rq[j]] = notfixed[j]
xu = np.array(xu)
print("23 年固定需求",np.c_["1",lines,np.round(xu)])
print("23 年 非 固 定 需 求 均 值 标 准 差 \n",np.c_["1",lines,np.round(f_mean,4),np.round(f_std,4)])


temp=pd.DataFrame(np.c_["1",lines,np.round(xu)],columns=["发货城市","收货城市","固定需求"])
```

```
    temp.to_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\固定需求(23年).xlsx")

    temp=pd.DataFrame(np.c_["1",lines,np.round(f_mean,2),np.round(f_std,2)],columns=["发货城市","收货城市","均值","标准差"])
    temp.to_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\非固定需求(均值与标准差)(23 年).xlsx")

    temp=pd.DataFrame(np.c_["1",lines,mt],columns=np.r_["0",["发货城市","收货城市"],date])
    temp.to_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\非固定需求(23年).xlsx")

    sc = np.array([["J","I"],["O","G"]])
    i=0
    for i in range(2):
        d1 = data[np.logical_and(data[:,1]==sc[i,0],data[:,2]==sc[i,1])][:,-1]
        fixed =d1.mean()-2*d1.std()          ##求出固定需求常数，即 3sigma 准则
        if(fixed<0):
            fixed=0
        d2 = d1-fixed
        sample_data = d2.reshape(-1, 1)

        # KDE 模型实例化,高斯核函数是一个常用的核函数，可以对连续变量进行估计，带宽参数影响核函数的宽度，决定了估计出的概率密度函数的平滑度。
        kde = KernelDensity(kernel='gaussian', bandwidth=6).fit(sample_data)

        # 指定评估点（根据实际需求调整范围和间隔）
        ep = np.linspace(d2.min(), d2.max(), num=300).reshape(-1, 1)

        # 评估 KDE 模型
        ld = kde.score_samples(ep)
        ds = np.exp(ld)

        # 绘制 KDE 结果和直方图
        fig, ax = plt.subplots()
        ax.plot(ep, ds, label='KDE')
        ax.hist(sample_data, bins=5, density=True, alpha=0.5, color='green', label='直方图')

        ax.set_xlabel('{}到{}(非固定需求)'.format(sc[i,0],sc[i,1]),fontsize=15)
        ax.set_ylabel('概率密度',fontsize=14)
        ax.legend(fontsize=14)
        plt.savefig('{}到{}(非固定需求)分布图.png'.format(sc[i,0],sc[i,1]))
```

```
plt.show()
temp=pd.DataFrame(np.c_["1",ep,ds[:,None]],columns=["非固定需求量","概率密度"])
temp.to_excel("C:\\Users\\Administrator\\Desktop\\vs 工作界面\\{}_{}(非固定需求).xlsx".format(sc[i,0],sc[i,1]))
```

附录 10：22 年固定需求（第一幅图）+23 年固定需求（第二幅图）

发货城市	收货城市	固定需求
A	O	24.0
A	Q	0.0
C	M	6.0
C	N	0.0
C	U	10.0
C	V	34.0
D	A	14.0
D	E	16.0
D	L	4.0
D	R	9.0
E	F	14.0
E	I	4.0
G	L	15.0
G	N	5.0
G	O	0.0
G	Q	16.0
G	R	6.0
G	V	0.0
G	X	40.0
H	J	0.0
H	K	14.0
H	L	36.0
I	E	4.0
I	F	31.0
I	J	3.0
I	S	28.0
J	H	22.0
J	I	5.0
J	K	0.0
J	L	14.0
K	H	0.0
K	J	14.0
K	L	0.0
L	D	12.0
L	G	42.0
L	H	19.0

L	J	0.0
L	K	0.0
L	O	24.0
L	P	27.0
L	R	0.0
L	W	0.0
L	X	0.0
M	C	0.0
M	G	17.0
M	N	9.0
M	U	0.0
M	V	12.0
N	G	22.0
N	M	0.0
N	V	3.0
O	G	0.0
O	Q	0.0
O	R	0.0
P	D	16.0
Q	A	0.0
Q	M	14.0
Q	N	0.0
Q	O	22.0
Q	V	0.0
R	D	6.0
R	G	0.0
R	L	0.0
R	O	0.0
R	S	0.0
S	D	0.0
S	I	0.0
S	L	36.0
S	Q	0.0
S	R	0.0
U	A	30.0
U	G	16.0
U	O	9.0
U	V	17.0
V	A	0.0
V	C	41.0
V	G	0.0
V	M	0.0
V	N	0.0

V	Q	0.0
W	L	0.0
W	X	1.0
W	Y	0.0
X	G	28.0
X	L	0.0
X	W	0.0
X	Y	0.0
Y	L	0.0
Y	W	0.0
Y	X	0.0

以上为 22 年固定需求。

发货城市	收货城市	固定需求
A	O	21.0
A	Q	0.0
C	M	28.0
C	N	29.0
C	U	20.0
C	V	22.0
D	A	13.0
D	E	9.0
D	L	16.0
D	R	16.0
E	F	14.0
E	I	2.0
G	L	15.0
G	N	29.0
G	O	0.0
G	Q	21.0
G	R	0.0
G	V	0.0
G	X	39.0
H	J	0.0
H	K	17.0
H	L	37.0
I	E	6.0
I	F	26.0
I	J	0.0
I	S	24.0
J	H	32.0
J	I	4.0
J	K	0.0
J	L	14.0

K	H	4.0
K	J	5.0
K	L	0.0
L	D	6.0
L	G	41.0
L	H	18.0
L	J	10.0
L	K	10.0
L	O	20.0
L	P	28.0
L	R	22.0
L	W	0.0
L	X	41.0
M	C	17.0
M	G	26.0
M	N	0.0
M	U	16.0
M	V	27.0
N	G	28.0
N	M	0.0
N	V	6.0
O	G	0.0
O	Q	0.0
O	R	0.0
P	D	15.0
Q	A	14.0
Q	M	1.0
Q	N	0.0
Q	O	22.0
Q	V	0.0
R	D	19.0
R	G	20.0
R	L	2.0
R	O	0.0
R	S	0.0
S	D	24.0
S	I	25.0
S	L	38.0
S	Q	0.0
S	R	0.0
U	A	8.0
U	G	22.0
U	O	10.0

U	V	15.0
V	A	13.0
V	C	38.0
V	G	0.0
V	M	11.0
V	N	10.0
V	Q	7.0
W	L	0.0
W	X	13.0
W	Y	0.0
X	G	37.0
X	L	4.0
X	W	0.0
X	Y	0.0
Y	L	0.0
Y	W	0.0
Y	X	0.0

以上为 23 年固定需求