队伍编号	MC2313387
题号	C

电商物流网络包裹应急调运与结构优化问题

摘 要

在互联网飞速发展的今天，各种电商平台飞速发展，已然成为我们生活购物的重要部分。于此同时，伴随而生的物流网络包裹调运与结构优化问题也因此变得尤为重要。然而，传统的包裹调运与结构优化存在资源分配不合理，调运过程复杂，缺乏智能化结构调优等问题。如何高效应急调运网络包裹，优化网络运输结构仍是一个具有挑战性的问题。因此，本文通过对电商物流网络进行研究，分别建立了**基于 LSTM 的 TCN 神经网络模型**和**基于群智能算法的多重粒子群寻优模型。**实验结果表明，我们所建立的 **TCN 神经网络模型**能够很好的依据历史数据来预测未来 31 天内的各线路货量数据，于此同时，**多重粒子群模型**根据每日不同货量来动态智能优化应急时的物流网络结构，极大的提高了网络的鲁棒性与泛化能力。

针对问题一：首先需要对所给历史货量数据进行**预处理**和**相关性分析**，接着我们建立了基于 **LSTM 的 TCN**（Temporal Convolutional Network）时序预测卷积神经网络模型。我们基于清洗后的数据来迭代，训练模型，并用该时间序列预测模型预测未来 2023-01-01 至 2023-01-31 一个月内各线路的货量运输情况，并完成准确度分析与模型评价。验证集上的误差 **MSE 和 MAE** 分别 0.016 和 0.098。

针对问题二：问题二假设 DC5 节点于 2023-01-01 关停，需要对 DC5 相关线路货量进行重新分配。我们基于**群智能算法**提出并建立了**多重粒子群寻优模型**，模型根据基于问题一得到了 2023-01-01 至 2023-01-31 日各线路的货量情况。此外，我们根据多目标优化函数进行拟合寻优得到局部最优解，以此作为我们对于 DC5 停运后的应急处理**货量分配**与物流**网络结构优化**方案。

针对问题三： 基于问题二，我们重新定义了目标优化函数，进一步扩展了粒子群算法，提出了**孪生多重粒子群智能优化**算法。我们加入了可以**动态规划**线路的**线路寻优粒子群**。模型通过迭代更新**分配方案**和**线路规划**这两个粒子群来获得最终的最优方案。基于此模型，我们得出了因 DC9 关停导致货量发生变化的线路数、不能正常流转的货量及网络的负荷情况。

针对问题四：为了不失科学性和全面性，我们提出了一种基于图论相关指标**介数中心性、紧密中心性、度中心性**和线路货量**加权评估**的线路评价方案。通过此方案，我们可以对该网络的不同物流场地及线路的重要性进行**分级**。同时，我们使用了复杂网络理论中的方法例如**攻击性分析**，**容错性分析**和**同步性分析**来对物流网络的鲁棒性进行近一步检验。

关键词：时序预测卷积神经网络 (TCN)，群智能优化算法,孪生多重粒子群算法（SMPSA),线路寻优粒子群 (LOPS),攻击性分析

目 录

一、问题背景

随着电商行业的不断发展，电商物流网络已成为电商企业发展的重要组成部分。电商物流网络旨在实现商品从仓库到消费者的快速、高效、安全的配送服务，但是在电商物流网络的建设和发展过程中，仍然存在着一系列的问题和挑战。

首先，电商物流网络在高峰期或突发情况下，常常出现**物流配送资源不足**、**运输效率低下**等问题。这就需要电商物流企业进行包裹**应急调运**，通过临时调度和调度优化，使物流配送能够顺畅进行。

其次，电商物流网络结构需要不断优化升级，以满足不同消费者需求的多样化和差异化。随着电商行业的快速发展，消费者对于商品的要求也越来越高，电商物流企业需要根据市场需求进行网络结构优化，以提高物流效率和服务质量。

历史上，随着电子商务的兴起和发展，各个电商企业逐渐形成了各自独立的物流配送网络。但是，这种模式存在着物流资源浪费、物流成本高昂等问题。因此，近年来，电商企业开始将物流网络进行整合和优化，以提高物流效率和降低物流成本。同时，也有不少企业开始引入**智能化技术和大数据分析**等手段，对物流网络进行优化和调整。

二、问题分析

问题一：本问题需要对附件所给的各路线的历史货量数据进行分析建模，并由此来预测未来 2023-01-01 到 2023-01-31 间各路线的货量运输情况。考虑到是时间序列预测问题，我们建立**时序预测卷积神经网络 TCN 来进行预测。**

1） 我们首先对数据进行初步清洗，使用**移动中值法**对离群值进行处理，然后将其整合成数据集，方便后续对于每条线路未来的货量情况预测。

2） 对需要预测的线路进行筛选，并对其他线路货量进行相关性分析，以此确定 TCN 模型的协变量。

3） 基于（2）中的数据与协变量，对将数据划分为训练集和验证集，当模型在验证集上表现优异时，证明我们的模型预测效果好，并开始对未来线路货量的预测。

问题二：本问题需要讨论在 DC5 停运后，如何优化物流应急调运与优化物流网络结构，从而使得效率最大化，损失最小化，我们建立多重粒子群寻优模型，来寻找使得满足上述条件的优化方案

1） 基于问题一得到的 2023-01-01 至 2023-01-31 日各节点的数据，找到与 DC5 直接相连的各个节点。

2） 通过改进的粒子群算法，在确定的**多维空间**中**自动搜索和预测分配线路**的全局最优结构。

3） 在搜索寻优的过程中，优化函数的最小值所在的精英粒子即是最佳方案（未正常流转包裹数、线路变化数、网络负荷情况作为决策变量）。

问题三：与问题二相似，本问题需要讨论在 DC9 停运后，如何优化物流应急调运与优化物流

网络结构，从而使得**效率最大化，损失最小化**。但本问题相对来说更加开放，可以自由增减线路，并且每一天都可以动态调整。因此，我们在多重粒子群寻优模型的基础上进一步改良模型，提出了孪生多重粒子群寻优模型，来寻找满足**约束条件**的动态优化方案。

1）基于问题一得到的 2023-01-01 至 2023-01-31 日各节点的数据，进行数据预处理，找到与 DC9 直接相连的各个节点。

2）通过改进的孪生粒子群算法，对每一天的线路增减和货量分配进行动态调整，自动搜寻整个月的最优结构。

3）在寻优过程中，优化函数的最小值所在的**精英粒子**即是最佳方案。

问题四：根据问题描述，我们需要建立一个依据图论相关指标的完整评价体系。

1） 对现有的物流场地和线路进行评价，以确定它们的重要性。

2） 确定新增物流场地的位置和新增线路的路径，并设计其处理能力和运输能力，优化整个物流网络。

3） 考虑到预测结果的随机性，我们还需要对所建网络的鲁棒性进行进一步分析。

三、符号说明

符号	说明	符号	说明
W	$Kendall's\ W$ 检验系数	w_i	L中第i条线路的权重
X	X^2检验统计量	$f(\theta)$	损失函数
$median$	移动中位数	$\nabla f(\theta)$	损失函数的梯度
x	卷积层输入张量	$RMSE$	均方根误差
y	卷积层输出张量	$C_B(v)$	介数中心性
d	卷积层扩张因子	$C_C(v)$	紧密中心性
$ReLU$	激活函数	$k(u,v)$	节点u,v之间的距离
$MAPE$	平均绝对百分比误差	$C_D(v)$	紧密中心性
DC_{in}	线路起点	k_v	节点v的度
DC_{out}	线路终点	N	网络节点数量
$x_{i,j}$	节点i,j之间的货量	$f(p)$	网络的攻击性
$G(x)$	粒子适应度	N_{gaint}	网络最大连通分量
$v(t),x(t)$	粒子t时刻的速度和位置	C	网络容错性
ω	惯性权重	d_{ij}	节点间的耦合强度
L	问题三损失函数	S	同步性
α，β	L中的各部分权重	W_i	评价体系中的权值

四、模型假设

1. 假设所有货运线路不考虑天气，地震等自然因素。
2. 假设所有仓库容量无限，不存在仓库货物溢出问题。
3. 物流场地的关停只影响到其相关的线路，其他线路的运输能力和处理能力不会受到影响。
4. 在物流场地关停时，所有包裹可以被分配到其他线路上，且分配方案需要尽可能减少因关停导致。货量变化的线路数和未能正常流转的包裹数量。
5. 建立线路货量的预测模型时，假设历史货量数据能够较好地预测未来的货量变化。

五、模型的建立与求解

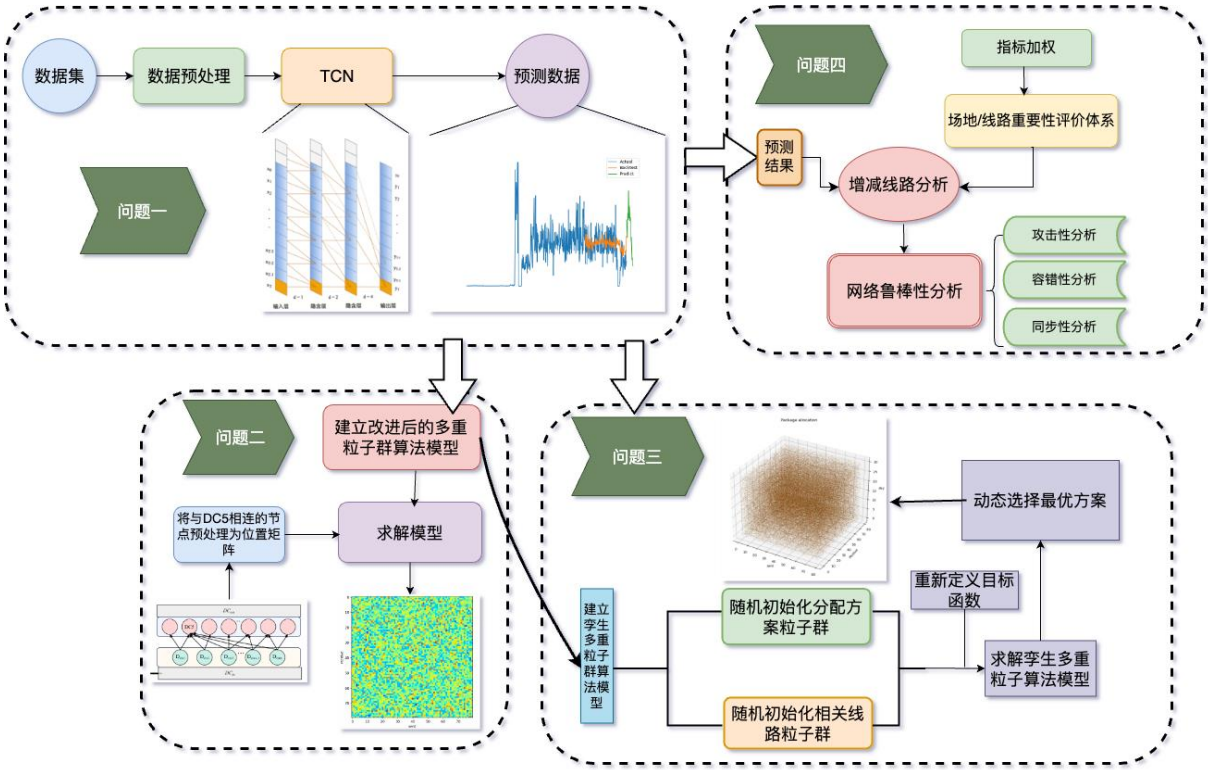


图 5.1 模型整体结构

5.1 问题一建模与求解

5.1.1 数据预处理与分析

我们需要根据历史货量数据预测$DC14-DC10$，$DC20-DC35$，$DC25-62$三条线路的未来一个月内的线路。考虑到单变量预测忽略外部因素影响，对异常值和噪声敏感，难以适应非线性趋势，缺乏泛化性。为了使后续时间序列模型预测更加准确，因此我们将先分析**变量间的相关性**，再进行

数据预处理，方便模型后续进行**多变量预测。**

I.相关性分析

考虑到仓库**每日处理货量能力有限**，我们进一步假设，其他变量如其他线路到目的地/始发地的每日运载货量与我们所预测的线路运输货量有一定相关性。经过分析，我们认为目标线路每日运载货量与始发地/目的地的物流场地同其他物流场地线路上的运载货量存在相关性。例如，如果研究$DC14 — DC10$这条线路的货量，则应该考虑$DC14/DC10$运往其他场地的货量、其他场地运往DC14/DC10的货量。

为了验证这一假设，我们使用了$Kendall$一致性检验算法[1] [2]来探究这五个变量之间的相关性。$Kendall's\ W$检验，即$Kendall$和谐性分析，W统计量称和谐性系数或一致性系数($coefficient\ of\ concordance$)，用于度量一致性好坏。对同一份数据作分析，$Kendall's\ W$检验拒绝H0与否和$Friedman$检验完全相同，但它们所检验的H0不相同，$Kendall's\ W$是$Friedman$统计量正态化的结果，取值在0~1之间，用以度量不同测量之间的一致性，系数越接近1。一致性越高。计算公式如下:

$$W = \frac{12S}{K^2(N^3 - N)} \tag{5.1.1}$$

式中，S为秩和与其平均值之差的平方和，K为秩评定的组数，N为被评秩的对象数。用卡方检验$X^2 = K(N-1)rw$，构造统计量$X^2 = K(N-1)rw$,其中自由度为$N-1$。如果

$$X^2 < X^2(df)\alpha \tag{5.1.2}$$

那么有$100(1-\alpha)\%$的把握可以断定K个变量总体不存在相关。如果

$$X^2 \geq X^2(df)a \tag{5.1.3}$$

那么有$100(1-\alpha)\%$的把握可以断定K个变量总体存在相关。

下表呈现的是使用$Kendall'sW$系数对$DC14-DC10$，$DC14-其他$，$其他-DC14$，$DC10-其他$，$其他-DC10$这五个变量的相关性的分析结果，结果显示，总体数据的显著性P值为0.000***，水平上呈现显著性，拒绝原假设α，因此数据呈现一致性，同时模型的$Kendall$协调系数W值为0.788，因此相关性的程度为高度的一致性。

表5.1.1：$Kendall'sW$系数相关性分析结果

$Kendall'sW$分析结果					
名称	秩平均值	中位数	$Kendall'sW$系数	X2	P
y	1.183	34213	0.788	2300.897	0.000***
x1	3.828	166269.5			
x2	3.441	138690.5			
x3	1.942	53516.5			
x4	4.605	235437.5			

注: ***、*、*分别代表1%、5%、10%的显著性水平

验证假设之后，我们对附件 1 中的数据进行整理和提取，建立了数据集。

II.数据预处理

为了清理数据集中的离群数据，我们采用了以移动中位数为基准的方法对数据进行了处理。选择移动中位数的原因是考虑到有些线路的运载货量是在以年为单位的尺度上有较大变化的，例如$DC14 — DC10$这条线路在2021年1月1日的货量是273，在2022年1月1日增长到了34918，这一年的变化很大。我们选择了大小为30的窗口（30正好是大多数月份的天数）计算移动中位数，移动中位数的计算公式如下：

设数据序列为 $x_1, x_2, x_3, \dots x_n$，窗口大小为K，则第i个移动中位数M_i为：

$$M_i = median\left(x_{i-\left[\frac{k-1}{2}\right]}, x_{i-\left[\frac{k-1}{2}\right]+1}, \cdots, x_{i+\left[\frac{k-1}{2}\right]}\right) \tag{5.1.4}$$

其中，$\left[\frac{k-1}{2}\right]$表示$k-1$除以 2 向下取整，$median$表示求中位数。找到离群值后，用中心值插入到其原本的位置。

下面的子图呈现的是$DC14 — DC10$这条线路的货量、$DC14/DC10$运往其他场地的货量和其他场地运往$DC14/DC10$的货量这五个时间序列的处理。

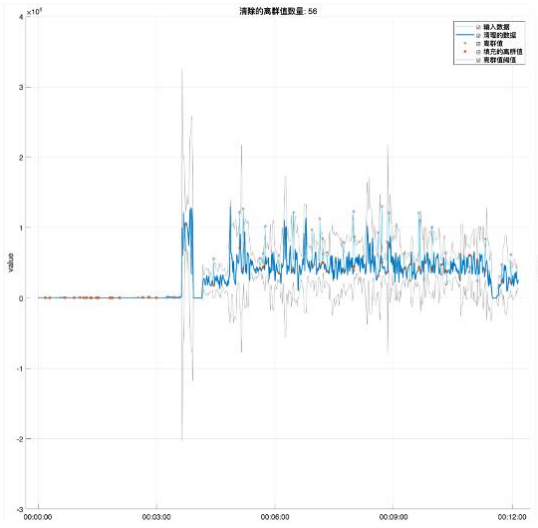


图 5.1.1 DC14-DC10 线路货量预处理

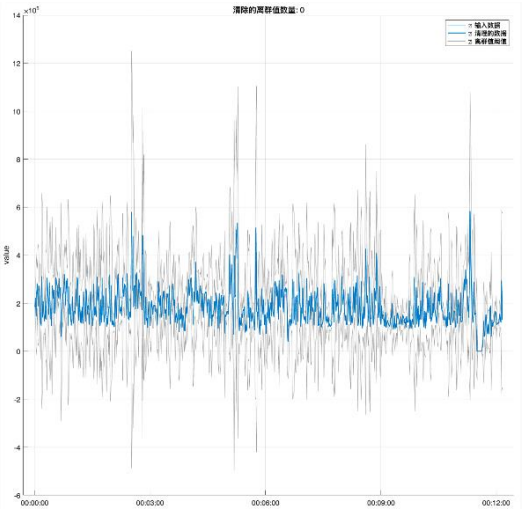


图 5.1.2 DC14-其他所有线路货量预处理

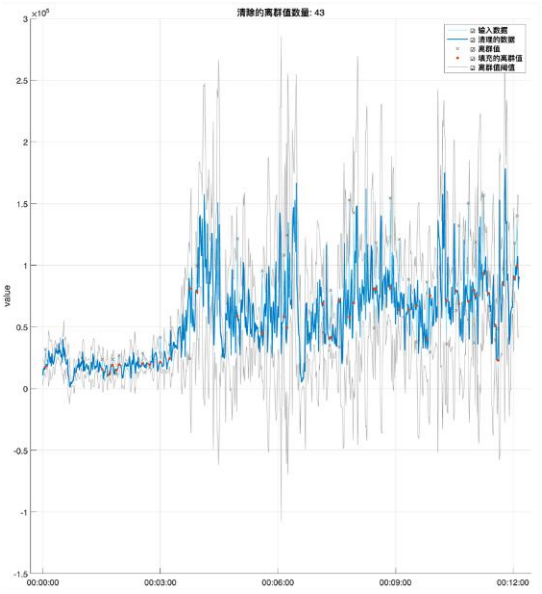


图 5.1.3 DC10-其他所有线路货量预处理

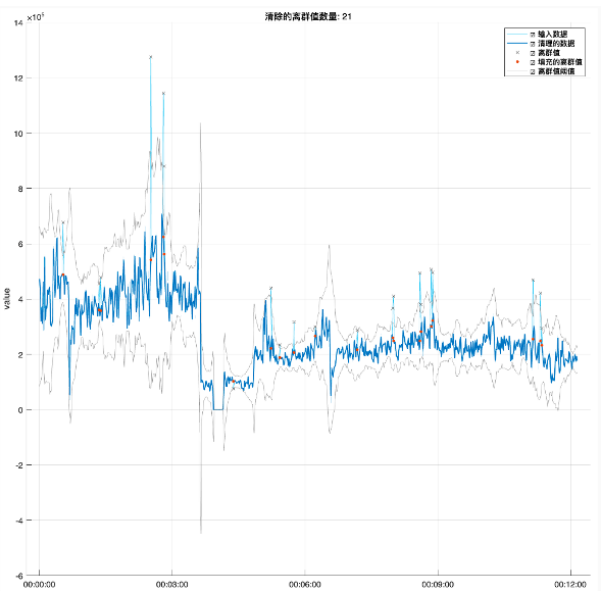


图 5.1.4 其他所有线路-DC10 货量预处理

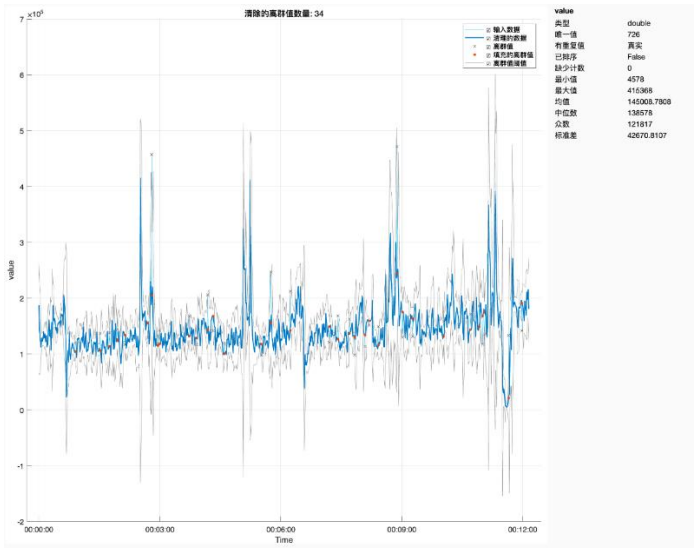


图 5.1.5 其他所有线路-DC14 货量预处理

其他两条线路也是按照上述方法进行预处理。值得注意的是，$DC25-DC62$的线路在$2021-01-01$至$2021-12-31$时，货量运输情况基本为 0，也是从 2022-01-01 开始逐渐有运货量，我们推测其是在$2022-01-01$才正式投入使用。

因此，对于 TCN 预测模型而言，2021 年的 0 数据训练意义不强，**不具有参考性和代表性**，为了防止其对模型的训练与预测造成较大影响，因此我们仅取了 2022 年的数据进行训练和预测。

5.1.2 TCN 神经网络模型建立

循环神经网络（RNN）是一种基于序列数据建模的神经网络，但是由于它的设计存在梯度消失和梯度爆炸等问题，因此长期依赖关系的建模存在一定的挑战。为了解决这个问题，

$Kalchbrenner$等人在 2016 年提出了一种全卷积神经网络，即时间卷积网络（TCN）。

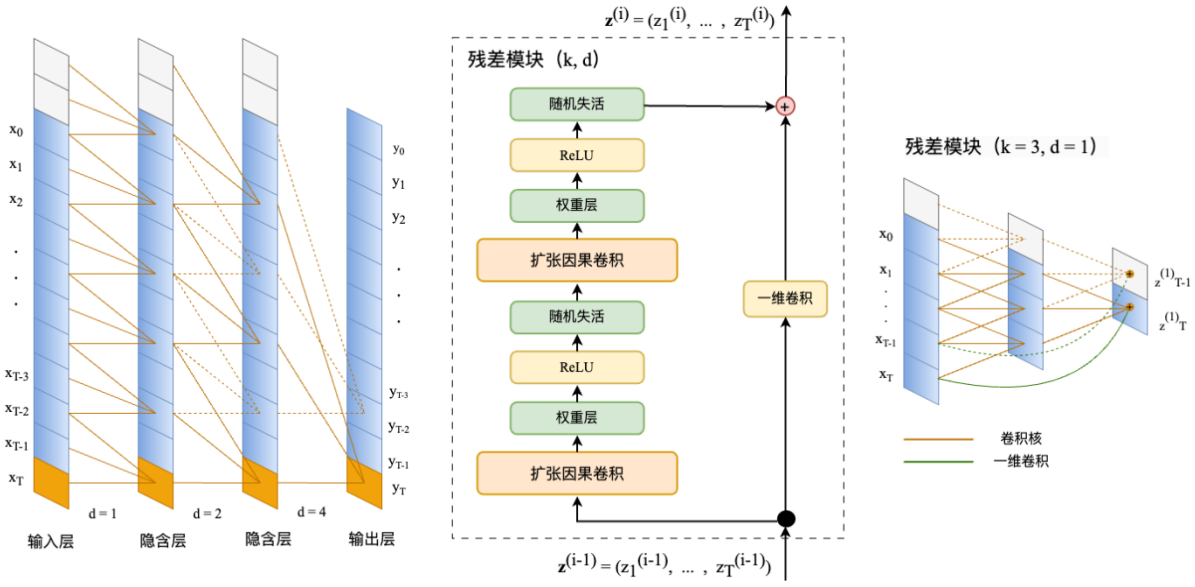


图 5.1.6 TCN 网络结构

时序卷积网络（TCN）是一种基于卷积神经网络（CNN）的序列建模方法，它能够很好的处理时间序列数据并做出较为精确的预测，并已被证实在许多应用领域表现优异。

TCN模型（见图一）由**多层级联残差模块**经**残差连接**而成，每个残差模块由一维卷积层、标准化层、非线性激活层和可选的下采样层串联组成。残差模块作为TCN的核心模块，在**特征学习**时有着至关重要的作用，下面我们将详细介绍残差模块中的**扩张因果卷积，残差连接，以及最后的评价指标。**

I.扩张的、因果的一维卷积层

如上所述，TCN 模型的输入和输出的一维张量具有相同的维度。为了完成这一点，TCN使用了一维全卷机网络（FCN）【3】架构，每个隐含层的长度与输入层相同，使用0补充以保持后续层和先前曾的长度相同。为了实现因果性，TCN采用了因果卷积，即t时刻的输出仅与时刻t和前一层中更早的元素进行卷积。可以简单概括为如下公式：

$$TCN = 一维FCN + 因果卷积 \tag{5.1.5}$$

简单的因果卷积只能回顾历史信息，其输出大小和网络深度呈线性关系，加入扩张卷积【4】可以实现更大的感受野，提高模型处理时间序列的能力，一般来说，对一维序列s 输入n维向量 $x \in \mathbb{R}^n$和一维滤波器$f: 0, \dots, k-1 \to \mathbb{R}$,对于张量$s$的扩张卷积可以表示为如下公式：

$$F(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s-d \cdot i} \tag{5.1.6}$$

其中d是膨胀因子，k是卷积核大小，$s - d * i$表示向过去时间进行卷积。扩张卷积等价于在相邻两个滤波器抽头之间引入固定步长。其中当$d = 1$时，扩张卷积就变成了常规卷积。更大的扩张可以让输出代表更广泛的输入，从而扩展卷积层的感受野。

II.残差连接

残差模块是从输出产生一条分支，该分支经过一系列的F变换，之后被添加到输入向量 x 上，残差模块的加入有利于深层网络的工作，可以有效解决梯度消失问题。为了使TCN不仅仅是一个过于复杂的线性回归模型，需要在卷积层的顶部添加激活函数来引入非线性。$ReLU$激活被添加到两个卷积层之后的残差模块中。权值规范化应用于每一个卷积层，可以让规范化隐含层的输入(抵消了梯度爆发的问题)，为了防止过拟合，在每个剩余块的每个卷积层之后通过随机失活引入正则化。

III.评价指标

$MSE(MeanSquaredError)$是一种常用的预测误差度量指标，用于评估回归模型的准确度。MSE表示平均平方误差，是预测值和实际值之间差值的平方的平均值。MSE的公式如下：

$$MSE(y,\hat{y}) = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - f(x_i)\right)^2 \tag{5.1.7}$$

$MAE(MeanAbsoluteError)$是一种常用的预测误差度量指标，用于评估回归模型的准确度。MAE 表示平均绝对误差，是预测值和实际值之间差值的绝对值的平均值。MAE的公式如下：

$$MAE(y,\hat{y}) = \frac{1}{m}\sum_{i=1}^{m}(|y_i - f(x_i)|) \tag{5.1.8}$$

$RMSE(RootMeanSquaredError)$是一种常用的预测误差度量指标，用于评估回归模型的准确度。$RMSE$表示平均平方根误差，是预测值和实际值之间差值的平方的平均值的平方根。$RMSE$的公式如下：

$$RMSE(y,\hat{y}) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y_i - f(x_i)\right)^2} \tag{5.1.9}$$

$MAPE(MeanAbsolutePercentageError)$是一种常用的预测误差度量指标，用于评估时间序列预测模型的准确度。$MAPE$表示平均绝对百分比误差，是预测值和实际值之间的百分比误差的平均值。$MAPE$的公式如下：

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right| \tag{5.1.10}$$

其中，A_i是实际值，F_i是预测值，n 是样本数量。

5.1.3 TCN 神经网络模型求解

I.参数设置及变量设置

以下为我们训练TCN网络时设置的参数：

表 5.1.2 TCN 模型参数设置

TCN	$output_chunk_length$	$Droupout\ Rate$	$weight_norm$	$Epochs$	$DropOut$
	31	0.1	$True$	500	0.1
	$input_chunk_length$	$dilation_base$	$kernel_size$	$num_filters$	$random_state$
	35	2	5	3	0

我们将$output_chunk_length$设置为题目要求的31，即预测未来31天的货量情况。

II.结果分析

下面给出题目要求的三条线路的未来 31 天的货量预测情况，纵轴代表实际**运货量**，横轴代表时间，蓝色曲线（$Actual$）代表$2021-01-01$ 至$2022-12-31$每天的实际**总货量**，橘色曲线（$Backtest$）代表以$2022-06-25$开始的历史**回溯预测**曲线，绿色曲线（$Predict$）则代表未来$2023-01-01$至$2023-1-31$每天的**货量预测**情况。

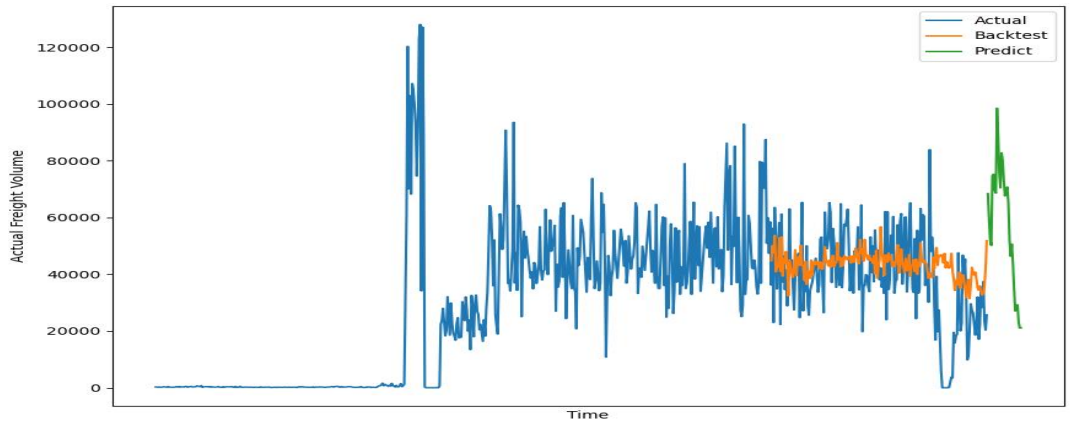


图 5.1.7 DC14→DC10 货量预测情况

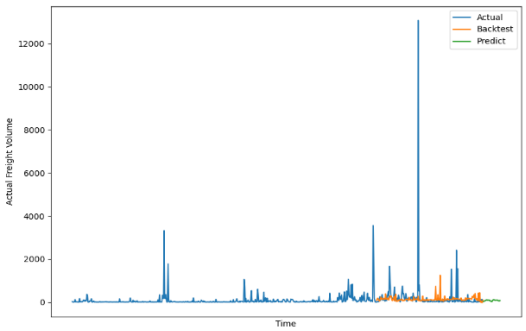


图 5.1.8 DC20→DC35 货量预测情况

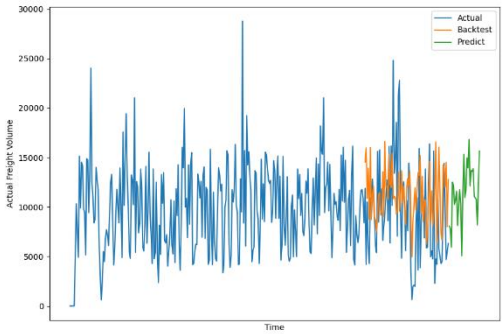


图 5.1.9 DC25→DC62 货量预测情况

表 5.1.3 三条线路各项评价指标

	$DC14-DC10$	$DC20-DC35$	$DC25-DC62$
MSE	0.016	0.005	0.036
$RMSE$	0.127	0.075	0.189
MAE	0.098	0.016	0.146
$MAPE$	NAN	NAN	81.45%

由图 5.1.7 至 5.1.9 可知，$DC14-DC10$的线路可以看出，$2021-01-01$至$2021-08-07$，该线路货量一直稳定在2000以内，在$2021-08-08$后开始激增，我们推测可能由于客观条件原因，可能是交通更加方便亦或者某地突然爆红带来的流量。而$DC20-DC35$的线路则出现间歇性高峰的情况，我们推测可能是存在促销活动或者用于附近线路故障时用于紧急调配使用。

5.2 问题二建模与求解

由问题一我们得到了$2023-01-01$至$2023-01-31$的各节点数据，为了建立线路**重新分配**的调整模型，我们首先需要找到所有直接运往$DC5$的节点DC_{in}，再对DC_{in}中每个节点找到可以运往其他节点的线路，组成DC_{out}，再采用粒子群算法求解最优的从DC_{in}到DC_{out}的分配方案。粒子群算法具有收敛速度快、参数少、算法简单易实现的优点（对高维度优化问题，比遗传算法更快收敛于最优解）。

5.2.1 数据预处理

我们需要确定$DC5$场地与其他场地之间的关系，以便在后续的建模处理中使用。具体而言，我们通过对数据进行处理，得到了$DC5$作为场地1时与多少个场地2相连，以及$DC5$作为场地2时与多少个场地 1 相连的信息。这些信息是建立物流网络模型所必需的，并可用于计算不同场地之间的物流成本和物流时间等参数。最终，我们编写代码得到了一组完整的物流网络数据，为后续的物流规划和优化提供了有力的支持。其中$DC5$作为场地1时与67个场地2相连，$DC5$作为场地2时与23个场地1相连。

5.2.2 多重粒子群的模型建立

I.粒子群算法

粒子群优化（PSO）是$Eberhart$和$Kennedy$引入的基于种群的优化技术。PSO 受到群体行为的启发，例如鸟群和鱼群等。在不互相碰撞的情况下，一群鸟或一群鱼能够寻找食物和栖息地。成员在组内共享信息,每个成员都使用自己的发现和小组信息来更新其方向。模仿这种社会行为，在 S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.中开发了 PSO 算法。在 PSO 算法中，群体中的每个粒子代表了给定问题的潜在解决方案。粒子通过使用自己和其他群体成员的最佳经验来调整“飞行方向”，从而找到问题的最佳位置。

在问题二中，我们对粒子群优化算法进行了改进，由于DC_{in}到DC_{out}之间并不是全连接的，而DC_{out}又取决于DC_{in}中各个节点如何分配自己的货物。所以我们并不能对DC_{out}中分配的货物量进行

任意的调整。因此，我们从DC_{in}入手，把DC_{in}中每个节点的货物向DC_{out}分配的情况作为粒子的位置，定义$x_{i,j}$为节点DC_{in}i 到节点DC_{out}j 的货物数量，由于DC_{in}有多个节点，那么粒子的位置 X（t）也就成为了一个矩阵，实验证明，这种处理方法收敛速度快、参数少、模拟效果好。我们将这种方法命名为**多重粒子群寻优算法。**

II.多重粒子群寻优算法

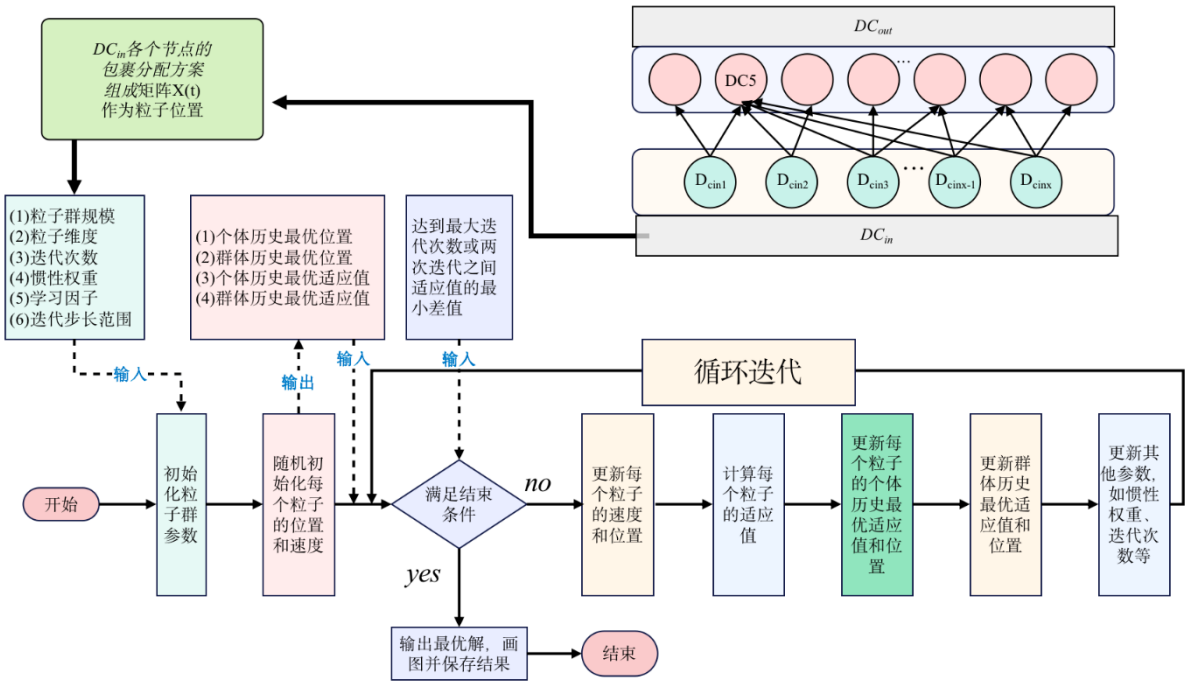


图 5.2.1 多重粒子群寻优算法

（1）初始化：由问题一中得到的结果确定粒子群的规模，在问题解空间中随机生成粒子的位置和速度。其中，粒子的位置代表DC_{in}中每个节点的货物向DC_{out}分配的情况，粒子的速度代表DC_{in}中每个节点的货物向DC_{out}中各个节点分配货物的比例调整情况。

定义h_i为处理能力最大值，f_j为运输能力最大值，建立约束条件：

$$\sum_{\mathrm{j}} x_{i,j} \le h_j, \forall i \tag{5.2.1}$$

$$\sum_{\mathrm{i}} x_{i,j} \le \mathrm{f}_j, \forall \mathrm{j} \tag{5.2.2}$$

$$x_{i,j\ge 0}, \forall i,j \tag{5.2.3}$$

对于粒子的位置，由于DC_{in}和DC_{out}之间并不是全连接的，因此在构建粒子的位置矩阵时，我们须将矩阵加上和掩蔽矩阵，以防止某些DC_{in}中节点的包裹分配到并无线路连接的DC_{out}中。

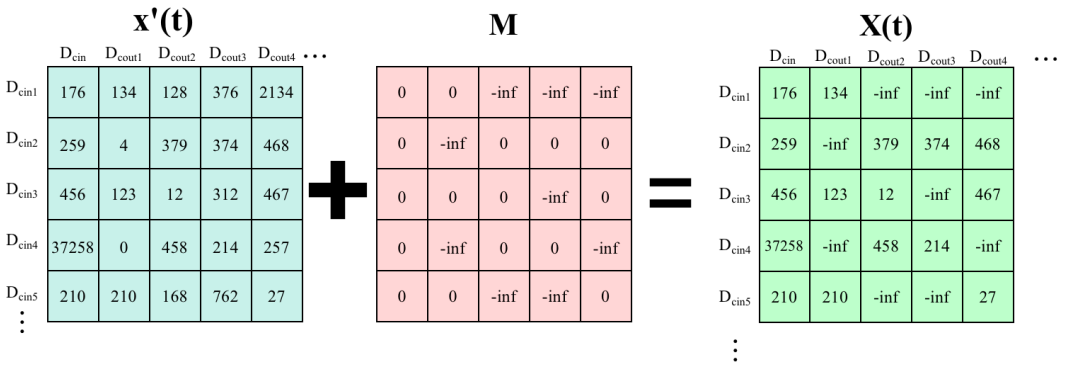


图 5.2.2 经过掩蔽矩阵处理后的粒子位置矩阵 X(t)

（2）评价：计算每个粒子的适应度（即目标函数值）：

$$G(x) = C1 * n + C2 * m + C3 * \sigma 2 \tag{5.2.4}$$

其中n代表了改变的线路数，m代表未正常流转的包裹总数，$\sigma 2$代表的是网络负荷的方差。将其与该粒子的历史最优解（个体最优解$pbest$）和整个群体的历史最优解（全局最优解$gbest$）进行比较。如果当前适应度更优，则更新相应的$pbest$和$gbest$。在计算当中，n即是位置矩阵 X 中有值的列，m为矩阵的第一列的总和，$\sigma 2$则可以通过计算矩阵每列总和/最大运输量，再求方差得到。

（3）更新：根据$pbest$和$gbest$更新每个粒子的速度和位置。速度更新的公式为：

$$v(\mathrm{t}+1) = \omega * v(\mathrm{t}) + c1 * r1 * \left(\mathrm{pbest} - \mathrm{x(t)}\right) + c2 * r2 * \left(\mathrm{gbest} - \mathrm{x(t)}\right) \tag{5.2.5}$$

更新后的位置计算为：

$$x(\mathrm{t}+1) = x(\mathrm{t}) + v(\mathrm{t}+1) \tag{5.2.6}$$

（4）终止条件：当满足预设的最大迭代次数或达到预期精度时，算法终止。输出全局最优解$gbest$

（5）输出结果：根据全局最优解，如果正常流转，给出因$DC5$关停导致货量发生变化的线路数及网络负荷情况；如果不能正常流转，给出因$DC5$关停导致货量发生变化的线路数、不能正常流转的货量及网络的负荷情况。

5.2.3 多重粒子群的模型求解

我们根据模型预测的结果，画出了以下热力图：

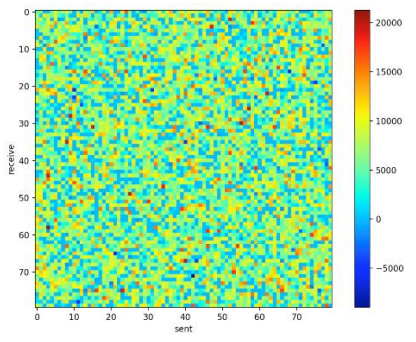


图 5.2.3 货物分配情况

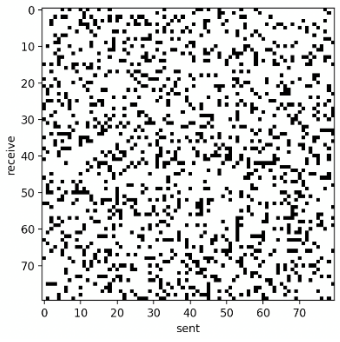


图 5.2.4 线路改变情况

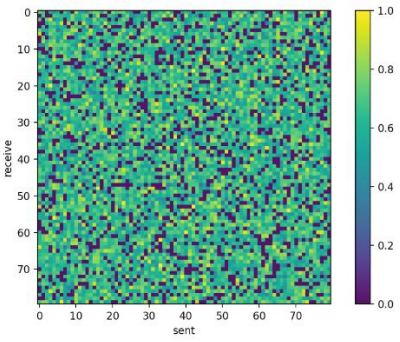


图 5.2.5 网络负荷情况

根据以上三图所示的数据分析，可以看出网络的总体负荷分配相对较为均衡，各个节点的负荷承载能力得到了充分利用，同时货物分配也比较合理，没有出现明显的堆积和滞留情况。此外，网络的线路**改变频率较低**，表明路线规划和调度方案相对稳定和有效。这样的结果表明多重粒子群模型在网络负荷均衡、货物分配和线路规划等方面取得了比较优秀的表现，并且在这三者之间找到了一种较为合理的平衡。

5.3 问题 3 分析与建模

在问题（2）的基础上，我们需要调整多重粒子群的寻优算法，使其能够根据不同时间网络的**差异负荷情况**，对物流网络线路进行动态增减，分配，使得物流分配效率最大化，成本最小化，因此我们提出了可学习的**孪生多重粒子群模型。**

5.3.1 数据预处理

此处预处理同问题 2 一致，只需将停运结点更换为$DC9$，按照相同方法进行数据处理即可，详情参照 5.2.1。

5.3.2 孪生多重粒子群模型建立

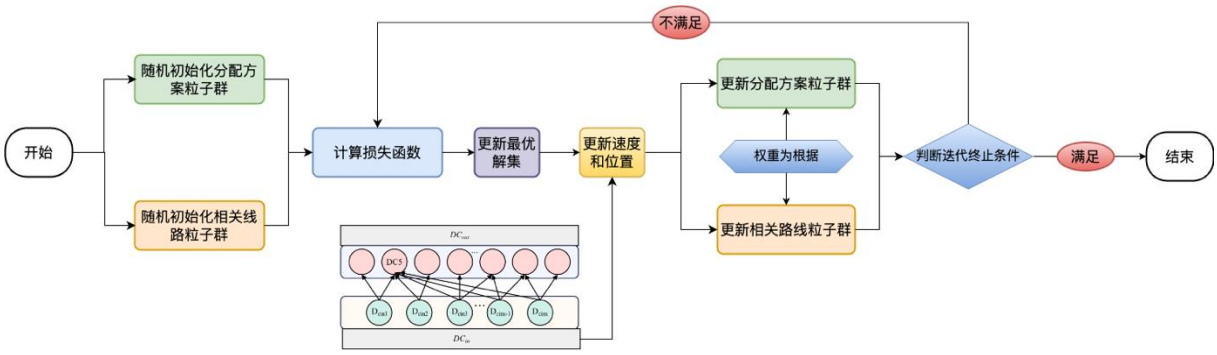


图 5.3.1 孪生多重粒子群智能优化算法流程

孪生多重粒子群的模型是基于问题二的模型进行改进，主要创新改进点在于**动态调整**与**线路的动态更新**，下面将详细介绍这两个模块。

I.损失函数优化

考虑到尽可能减少线路的变化，我们需要对损失函数进一步优化。我们将原有的损失函数分为两部分：**线路变动的损失和货量分配的损失。**其中，线路变动的损失考虑了线路的增减和变更；货量分配的损失则考虑了未正常流转的货量以及线路工作负载的均衡程度。我们将这两部分损失加权求和，作为新的损失函数。

假设有n条线路，$x_{i,j}$表示第i条线路在第j天的货量，$y_{i,j}$表示第i条线路在第j天的状态（1 表示存在，0 表示不存在），w_i表示第i条线路的权重（考虑工作负载均衡的因素），则问题 3 的损失函数

可以表示为：

$$L = \alpha \sum_{i=1}^{n} \sum_{j=1}^{T} w_i \left(y_{i,j} - y_{i,j-1}\right)^2 + \beta \sum_{i=1}^{n} \sum_{j=1}^{T} \left(x_{i,j} - \frac{1}{n} \sum_{k=1}^{n} x_{k,j}\right)^2 \tag{5.3.1}$$

其中，α和β是损失函数中各部分的权重，T是考虑的时间段。第一部分表示线路变动的损失，其中$\left(y_{i,j} - y_{i,j-1}\right)^2$表示第$i$条线路在第$j$天与前一天状态的变化，$w_i$表示第$i$条线路的权重，用于考虑工作负载均衡的因素。第二部分表示货量分配的损失，其中$\left(x_{i,j} - \frac{1}{n}\sum_{k=1}^{n} x_{k,j}\right)^2$表示第$i$条线路在第$j$天的货量与当天所有线路货量的平均值之间的差距。

我们可以通过调整α和β的值，来平衡线路变动和货量分配的重要性。同时，我们也可以根据具体问题的情况，调整w_i的值，以考虑工作负载均衡的因素。

II.线路动态更新

在多重粒子群优化算法中，我们引入了新的操作：动态增减线路。我们将每条线路表示为一个粒子，当需要增减线路时，我们可以将当前的粒子群进行扩展或缩减。具体而言，我们可以通过引入一定的随机性，让算法自主地增加或删除一定数量的粒子。增减的粒子根据前一天的历史数据进行初始化，从而保证操作的有效性和可行性。

在线路动态更新后，我们需要重新计算损失函数并进行优化。我们采用了一种**基于梯度下降的学习算法**来进行模型优化。具体而言，我们首先对模型进行初始化，并对每个粒子设置一定的学习率。然后，我们通过对损失函数的梯度进行计算，对粒子进行更新，以使损失函数尽可能地减小。

梯度下降是一种常见的优化算法，用于更新模型参数，使其逼近最优解。其核心思想是在参数空间中沿着负梯度方向进行迭代，以使损失函数最小化。

假设有一个函数$f(x)$（通常表示成损失函数），它的目标是最小化这个函数。对于一个函数的参数 θ，梯度下降算法的目标是找到θ的最优值，使得$f(\theta)$最小。梯度下降通过**反复迭代更新** θ 来实现这个目标。

具体来说，梯度下降的更新公式如下：

$$\theta = \theta - \alpha * \nabla f(\theta) \tag{5.3.2}$$

其中，α是学习率（learning rate），$\nabla f(\theta)$是损失函数$f(x)$对θ的梯度，表示损失函数 f(x)在 θ 处的变化率，即$f(x)$对θ的导数。这个导数可以通过求偏导数的方式得到。更新后的 θ 会沿着损失函数$f(x)$下降最快的方向进行调整，从而使损失函数逐步减小。

在粒子更新后，我们需要进行一定的后验处理，以保证线路的合法性和可行性。通过上述改进，我们可以实现对物流网络结构的动态更新，以求得最佳方案。我们将这种改进后的算法命名为**孪生多重粒子群**智能优化算法。

5.3.3 孪生多重粒子群模型求解

I.参数设置

表 5.3.1 孪生多重粒子群参数

$SMPSA$	$num_particles$	$num_iterations$	$inertia_weight$	$cognitive_coefficient$	$social_coefficient$
	30	100	0.7	2	2

II.结果分析

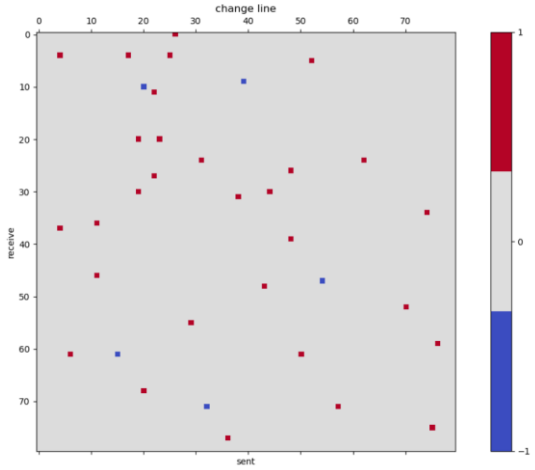


图 5.3.2 线路改变情况

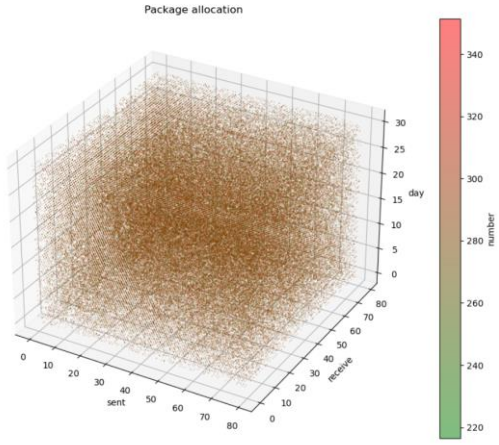


图 5.3.3 2023 年 1 月货物分配情况

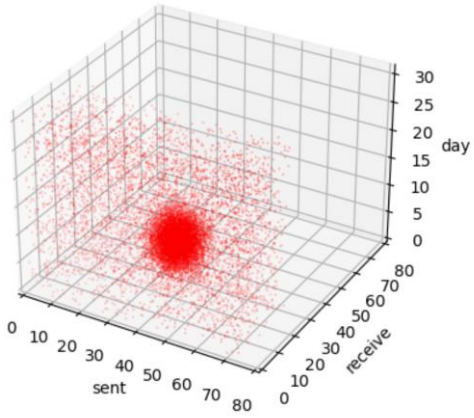


图 5.3.4 网络负荷情况

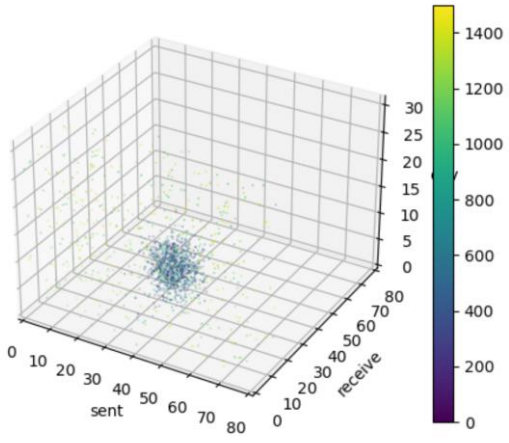


图 5.3.5 未能正常流转的包裹数量

其中，图 1 是2023年 1 月份的货物分配情况，X轴代表场地1，作为发送方，Y轴代表场地2，作为接收方，Z轴表示日期，红色表示增加，灰色表示不变，蓝色表示减少，从1号至31号。

图 2 是 2023 年 1 月货物分配情况，X 轴代表场地 1，作为发送方，Y 轴代表场地 2，作为接收方，Z轴表示日期。

图 3 是2023年一月的网络负荷情况，X 轴代表场地 1，作为发送方，Y 轴代表场地 2，作为接收方，Z轴表示日期，我们将网络负荷超过90%的线路标为红色，得到图上结果。

图 4 是未正常流转的包裹的情况，X 轴代表场地 1，作为发送方，Y 轴代表场地 2，作为接收方，Z 轴表示日期，从 1 号至 31 号。

根据以上四图的数据分析，我们可以观察到网络总体负担分布相当平衡，并且线路分配根据每天货量不同动态调整。各节点的承载能力得到了充分发挥，货物分配也比较合理。没有出现明显的拥堵和过多滞留现象。另外，网络线路调整频率较低，说明路线规划和调度方案较为稳定且高效。

5.4 问题 4 的模型建立与求解

根据问题描述，我们需要建立一个能够对物流网络进行优化的数学模型。具体来说，我们需要对现有的物流场地和线路进行评价，以确定它们的重要性；然后，我们需要确定新增物流场地的位置和新增线路的路径，并设计其处理能力和运输能力，以优化整个物流网络。此外，考虑到预测结果的**随机性**，我们还需要对所建网络的**鲁棒性**进行进一步分析。

5.4.1 物流场地和线路重要性优先级排序模型建立

为了对物流场地和线路进行评价，不失去科学性和合理性，我们使用基于网络分析的方法来评价不同物流场地和线路的重要性：**介数中心性**（Betweenness centrality）、**紧密中心性**（$Closeness\ centrality$）和**度中心性**（$Degree\ centrality$）等图论指标。

介数中心性是指网络中一个节点在所有最短路径中出现的次数，通常用于衡量节点在网络中的“桥梁”作用，即连接不同社区或子图的节点的重要程度。

介数中心性可以用以下公式计算

$$\boldsymbol{C_B(v) = \sum_{s\neq v\neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}} \tag{5.4.1}$$

其中, v是网络中的某个节点, σ_{st} 是节点s和节点t之间的最短路径数量, $\sigma_{st}(v)$是节点v在最短路径中作为桥梁出现的次数。

紧密中心性是指一个节点与其他节点之间距离的倒数之和，通常用于衡量节点在网络中的交互频率和信息传递效率，即越容易被其他节点到达的节点越重要。

紧密中心性可以用以下公式计算：

$$C_C(v) = \frac{N-1}{\sum_{u\neq v} d(u,v)} \tag{5.4.2}$$

其中,v是网络中的某个节点, N是网络中节点的数量, $d(u,v)$是节点u和节点 v之间的距离。

度中心性是指一个节点在网络中连接的数量，即节点的度，通常用于衡量节点在网络中的影响力和重要程度，即连接更多节点的节点越重要。

度中心性可以用以下公式计算：

$$C_D(v) = \frac{k_v}{N-1} \tag{5.4.3}$$

其中,v 是网络中的某个节点,k_v 是节点 v 的度,即与节点 v 相邻的节点数量,N 是网络中节点的数量。

我们对以上方法进行了加权,为不失全面性，根据前两问的求解和分析，我们引入了线路货量作为评价标准。由其他类型的网络，例如计算机网络、生物血液网络的先验知识可知，网络负荷量大的线路和吞吐量大的结点往往是关键线路和结点，据此，我们加入了货量大小的权重进行线路评价。

5.4.2 新增线路情况与网络鲁棒性模型建立

为了对物流网络的鲁棒性进行进一步分析，我们使用了复杂网络理论中的方法来分析网络的鲁棒性，例如攻击性分析、容错性分析和同步性分析等。

攻击性分析：攻击性分析是指在网络中随机或有目的地去掉一些节点或边，研究网络在攻击下的鲁棒性。 在复杂网络理论中,我们通常使用网络破坏程度（f) 来描述网络的攻击性。网络破坏程度是指在网络中去掉一定比例的节点或边之后,网络的连通性发生变化的程度。一般情况 下,我们将去掉的节点或边的比例定义为p。因此,网络破坏程度可以表示为:

$$f(p)=\frac{N_{\text{giant}}(p)-N_{\text{giant}}(0)}{N_{\text{giant}}(0)} \tag{5.4.4}$$

其中$N_{giant}(p)$是去掉 p 比例节点或边之后网络的最大连通分量的大小,$N_{giant}(0)$是原始网络的最大连通分量的大小。

容错性分析：容错性分析是指在网络中出现随机或有目的的节点或边失效的情况下，网络仍能保持其功能 和性能。容错性分析可以通过研究网络的鲁棒性来评估网络的容错性。在复杂网络理论中,我们通常使用网络连通性的变化来描述网络的容错性。容错性可以表示为：

$$\boldsymbol{C}=\frac{\boldsymbol{N}_{\text{giant}}-\boldsymbol{N}'_{\text{giant}}}{\boldsymbol{N}_{\text{giant}}} \tag{5.4.5}$$

其中N_{giant}是原始网络的最大连通分量的大小，N_{giant}是去掉一些节点或边之后网络的最大连通分量的大小。

同步性分析：同步性分析是指研究网络中节点之间的同步现象。在复杂网络理论中,我们通常使用节点间 的距离（d_{ij}）和节点间的耦合强度

$$\boldsymbol{S}=\frac{\boldsymbol{1}}{\boldsymbol{N(N-1)}}\sum_{\boldsymbol{i\neq j}}\frac{\boldsymbol{1}}{\boldsymbol{1+d_{ij}}}\left|\frac{\boldsymbol{k_i}}{\sqrt{\langle \boldsymbol{k^2}\rangle}}-\frac{\boldsymbol{k_j}}{\sqrt{\langle \boldsymbol{k^2}\rangle}}\right| \tag{5.4.6}$$

其中N是网络中节点的数量，$\langle k^2\rangle$是节点的平均度数的平方。

5.4.3 重要性排序模型求解

我们使用介数中心性、紧密中心性和度中心性等图论指标来对不同物流场地和线路的重要性进行评价。其中，介数中心性被用于衡量节点在网络中的“桥梁”作用，即连接不同社区或子图的节点的重要程度。

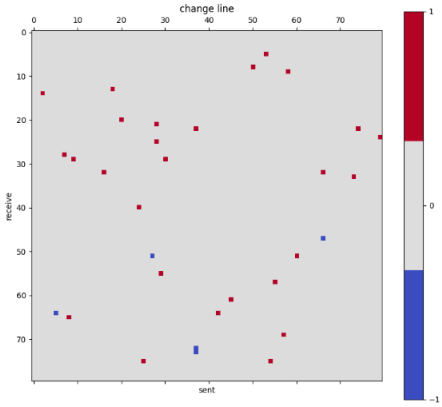
在介数中心性的计算公式中，分子部分是节点在所有最短路径中作为桥梁出现的次数之和，分母部分是所有节点对之间的最短路径数量之和。介数中心性越高的节点，在连接不同社区或子图时，所处的位置越关键，其重要性也越高。

据此，我们可以得出一个有各条线路的评分表。每条线路的评分值反映了该线路在物流网络中的重要性，评分越高表示该线路的贡献越大，对于整个物流网络的连通性和稳定性也具有更高的保障作用。在实际应用中，可以根据这些评分值来对物流网络进行优化调整，以提高物流效率和降低物流成本。

表 5.4.1 线性重要性得分情况

线路	*得分*
$DC17-DC4$	0.979907586
$DC14-DC8$	0.97985583
$DC23-DC4$	0.979812027
…	…
$DC31-DC4$	0.31029299
$DC50-DC14$	0.310196757
$DC46-DC35$	0.310118092

5.4.4 新增线路情况与网络鲁棒性研究求解



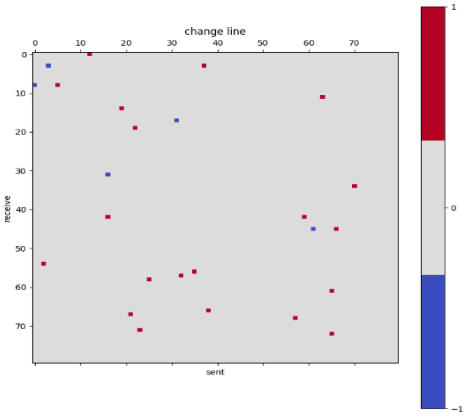


表 5.4.1 新增线路情况　　**表 5.4.2 攻击 DC72 节点后线路动态调整情况**

在网络鲁棒性分析中，我们进行了**攻击性分析**，即在网络中去掉了一个节点，观察网络结构的**差异性变化**。我们以 D72 节点为例，去掉这个节点，再运用问题 3 点孪生多重粒子群模型进行动态线路的分配。得到**表 5.4.2** 。根据我们的实验结果，我们的模型在去掉一个节点后，仍然能够对网络进行动态线路的分配，以保证物流网络的高效性和鲁棒性。这再次说明我们所建立的模型在**应急情况**下具有较好的**动态调整能力**。

六、灵敏度分析

根据我们对问题的分析，下面列出的系数将对我们构建的模型产生影响，并最终影响我们的判断。 将模型迁移到其他团簇时，有必要了解以下参数的变化对模型结果的影响，以便对模型的准确性进行分析。

6.1 TCN灵敏度分析

表 6.1.1 孪生多重粒子群参数

	η	β_1	β_1
参数值 1	0.8	0.001	0.92
TCN 网络收敛时间-预测精度	51.5s-1.6%		
参数值 2	0.9	0.01	0.5
TCN 网络收敛时间-预测精度	40.3s-8.9%		
参数值 3	0.7	0.0001	1.5
TCN 网络收敛时间-预测精度	210.3s-6.9%		

可以发现，η=0.8，β_1=0.001，β_1=0.92 为 TCN 模型最优参数。

6.2 孪生多重粒子群寻优灵敏度分析

对于数据的训练过程，我们使用多重粒子群网络。网络的超参数调整如下表所示。可以发现，在调整前后，多重粒子群网络在设置 lr=0.01，Hs=200 是具有最佳的预测性能和收敛速度。

表 5.3.1 孪生多重粒子群参数

	LR	Hs
参数值 1	0.01	200
SMPSA 网络收敛时间-预测精度	51.5s-1.6%	
参数值 2	0.001	1000
SMPSA 网络收敛时间-预测精度	511.5s-16.5%	
参数值 3	0.1	100
SMPSA 网络收敛时间-预测精度	22.5s-9.5%	

6.3 异构网络参数：P

在模型的解中，我们将p设置为 0.5，然后将p依次设置为 0.3、0.35、0.4、0.45、0.6、0.65、0.7。通过分析更改 P 值之前和之后，分析了调整前后的误差百分比(与给定的标准型对比)。可以发现，当 P=0.35 时，异构网络模型具有最小误差。

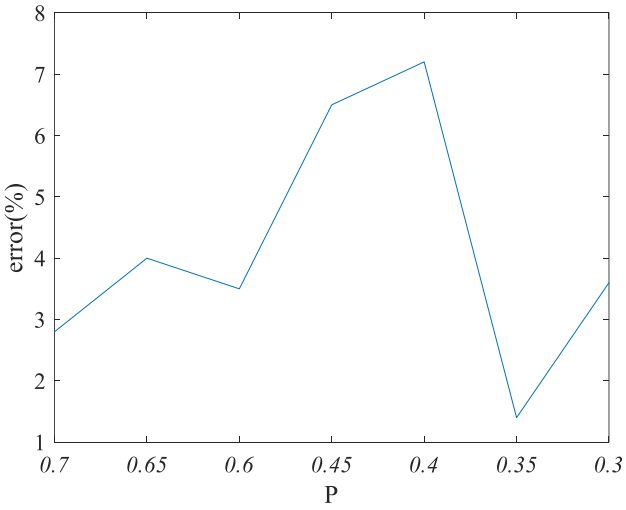


图 6.1 异构网络参数

七、模型评价与改进

7.1 模型评价

7.1.1模型的优点

1） 问题一建立了 **TCN** 时序预测卷积神经网络。相比于传统的循环神经网络（RNN），TCN 能够更好地捕捉时间序列中的长程依赖关系，并且可以并行化处理输入序列，提高了计算效率。TCN 模型可以处理等长或不等长的序列，并且可以自适应地调整模型深度和复杂度以适应不同的任务。相较于常用于时间序列预测任务的长短时记忆网络（LSTM），TCN 可以更好地捕捉时间序列数据中的长程依赖关系，并且具有更快的训练速度和更少的参数量，所以在时序预测任务上具有更高的计算效率和准确性。

2）问题二，三建立了**多重粒子群寻优算法**解决物流线路规划问题。对于物流线路规划这种**高维度优化问题**，多重粒子群寻优算法的表现要比遗传算法更优，因为遗传算法需要维护大量的基因和染色体，而多重粒子群寻优算法只需要维护每个粒子的位置和速度，对于高维度优化问题，多重粒子群寻优算法的收敛速度更快，更容易找到最优解。

3）**参数少，收敛速度快**。多重粒子群寻优算法可以快速地找到最优解，这是因为该算法利用了多个粒子的经验，并能够在搜索空间中不断地调整粒子的位置和速度，以快速收敛到最优解。

7.1.2 模型的缺点

1）**可扩展性有限**：我们设计的模型目前仅适用于特定的场景，无法完美适应于其他领域或问题。如果要将该模型应用于其他场景，需要进行调整和重新设计。

2）粒子群算法缺陷：粒子群算法为启发式算法，启发式算法本身就存在陷入局部最优解的问题，所以对于模型二，三同样如此，问题可能无法得到全局最优解

7.2 模型改进

1）对于问题一的 TCN 模型，可以加入注意力机制或门控卷积层，将门控机制应用于卷积层中，可以让模型更好地控制信息的流动，从而提高模型的精度和稳定性；引入多个不同尺度的卷积核来学习不同时间尺度的特征，从而提高模型对时间序列数据的建模能力。

2）对于多重粒子群寻优模型，可以引入自适应惯性权重，以提高 PSO 算法的收敛速度和全局搜索能力。自适应惯性权重方法的基本思想是通过调整惯性权重，实现粒子在搜索过程中的探索和开发的平衡，从而更好地适应不同阶段的搜索需求。

参考文献

【1】Scientific Platform Serving for Statistics Professional 2021. SPSSPRO. (Version 1.0.11)[Online Application Software].

【2】李娟生,李江红,刘小宁,申希平,米友军.Kendall’s W 分析方法在医学数据处理中的应用及在 SPSS 中的实现方法[J].现代预防医学,2008(01):33+42.

【3】Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

【4】van den Oord, Aaron, Dieleman, Sander, Zen, Heiga, Simonyan, ¨Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew W., and Kavukcuoglu, Koray. WaveNet: A generative model for rawaudio.arXiv:1609.03499,2016.

附录

问题一代码：

```
from utils import fix_pythonpath_if_working_locally
fix_pythonpath_if_working_locally()
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from scipy.io import wavfile
from scipy.fftpack import dct
import warnings
warnings.filterwarnings('ignore')
from darts.metrics import mape, mse, mae
from darts import TimeSeries
from darts.models import TCNModel, RNNModel
from darts.dataprocessing.transformers import Scaler
from darts.utils.timeseries_generation import datetime_attribute_timeseries
from darts.metrics import mape, r2_score
from darts.utils.missing_values import fill_missing_values
from darts.datasets import AirPassengersDataset, SunspotsDataset, EnergyDataset
from sklearn.metrics import mean_squared_error
import xlrd
import csv
import os
import xlsxwriter
from xlutils.copy import copy
import openpyxl
from openpyxl.utils import get_column_letter,column_index_from_string
from openpyxl import Workbook

# Read data:
# We'll use the month as a covariate
""" month_series = datetime_attribute_timeseries(ts, attribute="month", one_hot=True)
scaler_month = Scaler()
month_series = scaler_month.fit_transform(month_series)
 """
train_path="/home/jack/Project/TCN_new/data_14-10/"
test_path="/home/jack/Project/TCN_new/data_25-62/25to62/"

# 读取 Excel 文件中的特定工作表
def get_data(file_path):
    df = pd.read_excel(train_path+file_path,index_col="timestamp")
    df.index = pd.to_datetime(df.index)
```

```
    ts=df['value']
    ts=TimeSeries.from_series(ts)
    scaler=Scaler()
    ts=scaler.fit_transform(ts)
    return ts, scaler
def cor_data(file_path):
    df = pd.read_excel(train_path+file_path,index_col="timestamp")
    df.index = pd.to_datetime(df.index)
    ts=df['value']
    ts=TimeSeries.from_series(ts)
    return ts

ts,ts_scaler=get_data("14-10.xlsx")
cor1=cor_data("10-all.xlsx")
cor2=cor_data("14-all.xlsx")
cor3=cor_data("all-14.xlsx")
cor4=cor_data("all-10.xlsx")
cor_all=cor1.stack(cor2)
cor_all=cor_all.stack(cor3)
cor_all=cor_all.stack(cor4)
cor_scaler=Scaler()
cor_all=cor_scaler.fit_transform(cor_all)
train,test_val=ts.split_before(pd.Timestamp("20220910"))
train_cor,test_cor=cor_all.split_before(pd.Timestamp("20220910"))
""" train_14_o=get_data("train","14-other_train.xlsx")
train_o_10=get_data("train","other-10_train.xlsx")
train_o_14=get_data("train","other-14_train.xlsx") """

"""
test_14_10=get_data("test","14-10_test.xlsx")
test_10_o=get_data("test","10-other_test.xlsx") """
""" test_14_o=get_data("test","14-other_test.xlsx")
test_o_14=get_data("test","other-14_test.xlsx")
test_o_10=get_data("test","other-10_test.xlsx") """


""" train_multivariate = merge_time_series([ts, train_10_o, ])
test_multivariate = merge_time_series([test_14_10,tst_o_14, test_10_o,test_14_o,test_o_10]) """
# Create training and validation sets:
""" train, val = ts.split_after(pd.Timestamp("19580801"))
train_month, val_month = month_series.split_after(pd.Timestamp("19580801")) """
model_air = TCNModel(
    input_chunk_length=35,
    output_chunk_length=31,
    n_epochs=500,
```

```
    dropout=0.1,
    dilation_base=2,
    weight_norm=True,
    kernel_size=5,
    num_filters=3,
    random_state=0,
)
model_air.fit(
    series=train,
    past_covariates=train_cor,
    val_series=test_val,
    val_past_covariates=test_cor,
    verbose=True,
)
backtest = model_air.historical_forecasts(
    series=ts,
    past_covariates=cor_all,
    start=0.7,
    forecast_horizon=31,
    retrain=False,
    verbose=True,
)
pred=model_air.predict(series=ts,past_covariates=cor_all,n=30)
Mse=mse(ts,backtest)
Rmse=math.sqrt(Mse)
Mae=mae(ts,backtest)
#Mape=mape(ts,backtest)
print("Mse: ", Mse)
print("Rmse: ", Rmse)
print("Mae: ", Mae)
#print("Mape: ", Mape)

ts.plot(label="actual")
backtest.plot(label="backtest")
pred.plot(label="pred(H=30days)")
plt.legend()
fig = plt.figure(figsize=(10, 8))
plt.savefig('/home/jack/Project/TCN_new/25-62 归一化.png')


ts=ts_scaler.inverse_transform(ts)
pred=ts_scaler.inverse_transform(pred)
backtest=ts_scaler.inverse_transform(backtest)
""" ts.plot(label="actual")
backtest.plot(label="backtest")
```

```
pred.plot(label="pred(H=31days)")
plt.savefig('/home/jack/Project/TCN_new/14-10 实际运载量.png')
 """
""" ts.plot(label="actual")
backtest.plot(label="backtest")
pred.plot(label="pred(H=30days)")
plt.legend()
plt.savefig('/home/jack/Project/TCN_new/14-10 归一化.png')
 """
# plot unnormalized data
""" ts_unscaled = ts_scaler.inverse_transform(ts)
ts_unscaled=np.array(ts_unscaled.values).reshape(-1,1).flatten()
pred_unscaled = ts_scaler.inverse_transform(pred)
pred_unscaled = np.array(pred_unscaled.values).reshape(-1,1).flatten()
backtest_unscaled = ts_scaler.inverse_transform(backtest)
backtest_unscaled=np.array(backtest_unscaled.values).reshape(-1,1).flatten() """

#放弃，输出成数在另外一个数据画！
""" fig, ax = plt.subplots()
ax.plot(ts_unscaled, label="actual")
ax.plot(backtest_unscaled, label="backtest")
ax.plot(pred_unscaled, label="pred(H=31days)")
ax.legend()
plt.savefig('/home/jack/Project/TCN_new/14-10 实际运载量.png') """
pred=pred.pd_dataframe()
ts=ts.pd_dataframe()
backtest=backtest.pd_dataframe()
p_values=pred['value']
p_dates=pred.index
t_values=ts['value']
t_dates=ts.index
b_values=backtest['value']
b_dates=backtest.index
with open("./TCN_new/data_25-62/25to62/pred.txt", 'a+') as fpa:
    len=pred.__len__()
    for i in range(len):
        fpa.write(str(p_dates[i])+"\t"+str(p_values[i])+"\n")
with open("./TCN_new/data_25-62/25to62/ts.txt", 'a+') as fpa:
    len=ts.__len__()
    for i in range(len):
        fpa.write(str(t_dates[i])+"\t"+str(t_values[i])+"\n")
with open("./TCN_new/data_25-62/25to62/back.txt", 'a+') as fpa:
    len=backtest.__len__()
    for i in range(len):
```

```
        fpa.write(str(b_dates[i])+"\t"+str(b_values[i])+"\n")
```

问题二、三代码：

```
import numpy as np
import pandas as pd
from random import random, uniform

import matplotlib.pyplot as plt

def create_max_load_matrix(data, max_loads):
 source_nodes = data["场地 1"].unique()
 target_nodes = data["场地 2"].unique()
 matrix = pd.DataFrame(columns=target_nodes, index=source_nodes).fillna(0)
 for idx, row in max_loads.iterrows():
  source, target = row["Route"].split("_")
  matrix.at[source, target] = row["货量"]
  matrix.at[target, source] = row["货量"]
 return matrix



# 根据输入数据创建初始分配矩阵
def create_initial_matrix(data):
 source_nodes = data["场地 1"].unique()
 target_nodes = data["场地 2"].unique()
 matrix = pd.DataFrame(columns=target_nodes, index=source_nodes).fillna(0)
 for idx, row in data.iterrows():
  matrix.at[row["场地 1"], row["场地 2"]] = row["货量"]
 return matrix



# 初始化粒子群
def initialize_particles(num_particles, matrix, mask_matrix):
 particles = []
 for _ in range(num_particles):
  particle_position = matrix.copy()
  for i in range(particle_position.shape[0]):
   for j in range(particle_position.shape[1]):
    if particle_position.iloc[i, j] != 0:
     particle_position.iloc[i, j] = particle_position.iloc[i, j] * uniform(0, 1)

  redistribute_goods(particle_position, matrix)
  particle_position = particle_position * mask_matrix
  particle_velocity = matrix.applymap(lambda x: uniform(-1, 1))
```

```
  particles.append({"position": particle_position, "velocity": particle_velocity, "best_position":
particle_position.copy()})
 return particles



# 计算目标函数
def objective_function(position_matrix):
 # 定义系数
 c1 = 3
 c2 = 20
 c3 = 10000

 # 计算货量发生变化的线路数
 changed_lines = np.sum(position_matrix != initial_matrix)

 # 计算不能正常流转的货量
 stuck_goods = np.sum(position_matrix.iloc[:, 0])

 # 计算网络负荷情况
 target_loads = position_matrix.sum(axis=0)
 load_variance = np.var(target_loads) / max_load_variance

 # 计算目标函数值
 objective_value = c1 * changed_lines + c2 * stuck_goods + c3 * load_variance

 return objective_value

def redistribute_goods(new_position, initial_matrix):
 for i in range(new_position.shape[0]):
  for j in range(new_position.shape[1]):
   if new_position.iloc[i, j] == 0 and initial_matrix.iloc[i, j] != 0:
    remaining_goods = initial_matrix.iloc[i, j]
    non_zero_positions = np.where(new_position.iloc[i, :] != 0)[0]
    if len(non_zero_positions) > 0:
     redistribute_amount = int(remaining_goods / len(non_zero_positions))
     remaining_goods -= redistribute_amount * len(non_zero_positions)
     for k in non_zero_positions:
      new_position.iloc[i, k] += redistribute_amount
     new_position.iloc[i, j] = remaining_goods


# 粒子群优化算法
```

```
def particle_swarm_optimization(particles, num_iterations, inertia_weight, cognitive_coefficient, social_coefficient,
mask_matrix):
 global_best_position = particles[0]["best_position"]
 global_best_value = objective_function(global_best_position)
 global_best_values = [global_best_value]  # 添加一个列表以存储每次迭代的全局最优解

 for _ in range(num_iterations):
  for particle in particles:
   position = particle["position"]
   velocity = particle["velocity"]
   best_position = particle["best_position"]

   # 更新速度
   r1, r2 = random(), random()
   cognitive_term = cognitive_coefficient * r1 * (best_position - position)
   social_term = social_coefficient * r2 * (global_best_position - position)
   new_velocity = inertia_weight * velocity + cognitive_term + social_term
   new_velocity = np.round(new_velocity * mask_matrix)
   particle["velocity"] = new_velocity

   # 更新位置
   new_position = position + new_velocity
   new_position = np.round(new_position * mask_matrix)
   redistribute_goods(new_position, initial_matrix)
   particle["position"] = new_position

   # 更新个体最优位置
   current_value = objective_function(new_position)
   if np.all(current_value < objective_function(best_position)):
    particle["best_position"] = new_position.copy()

   # 更新全局最优位置
   if np.all(current_value < global_best_value):
    global_best_position = new_position.copy()
    global_best_value = current_value

  global_best_values.append(global_best_value)  # 将当前迭代的全局最优解添加到列表中

 return global_best_position, global_best_value, global_best_values  # 返回全局最优解列表



# 读取 Excel 文件
file_path = "附件 1：物流网络历史货量数据.xlsx"  # 请将此路径替换为您的 Excel 文件路径
df = pd.read_excel(file_path, engine="openpyxl")
```

```
# 将起点和终点的组合作为新的一列，注意需要对每一条线路的起点和终点进行排序
df['Route'] = df[['场地 1', '场地 2']].apply(lambda x: '_'.join(sorted(x)), axis=1)

# 按照线路进行分组，并计算历史最高货量
result = df.groupby('Route')['货量'].max().reset_index()

dc5_as_site1 = set()
dc5_as_site2 = set()

# 遍历表格数据
for _, row in df.iterrows():
 site1, site2 = row["场地 1"], row["场地 2"]
 if site1 == "DC5":
  dc5_as_site1.add(site2)
 if site2 == "DC5":
  dc5_as_site2.add(site1)

# 使用集合去除重复场地
unique_dc5_as_site1 = set(dc5_as_site1)
unique_dc5_as_site2 = set(dc5_as_site2)

print(f"DC5 作为场地 1 与 {len(unique_dc5_as_site1)} 个场地 2 相连（去除重复）")
print(f"DC5 作为场地 2 与 {len(unique_dc5_as_site2)} 个场地 1 相连（去除重复）")
print(unique_dc5_as_site1)



# 筛选出 DC5 作为场地 2 的数据
dc5_site2_data = df[df["场地 2"] == "DC5"]

# 获取与 DC5 相连的其他场地
connected_sites = dc5_site2_data["场地 1"].unique()

# 筛选出这些场地作为场地 1 的所有数据
data = df[df["场地 1"].isin(connected_sites)]

# 去除与 DC5 相关的数据
data = data[data["场地 2"] != "DC5"]

# 重置数据索引
data.reset_index(drop=True, inplace=True)

print(data)
```

```
# 创建初始矩阵
initial_matrix = create_initial_matrix(data)

# 修改后的掩蔽矩阵创建代码
mask_matrix = initial_matrix.copy()
mask_matrix[mask_matrix != 0] = 1
mask_matrix[mask_matrix == 0] = 0

# 算法参数
num_particles = 30
num_iterations = 100
inertia_weight = 0.7
cognitive_coefficient = 2
social_coefficient = 2

# 初始化粒子群
particles = initialize_particles(num_particles, initial_matrix, mask_matrix)

total_goods = initial_matrix.sum().sum()
max_load_variance = np.var(np.ones(initial_matrix.shape[1]) * total_goods / initial_matrix.shape[1])

# 执行粒子群优化算法
best_position, best_value, global_best_values = particle_swarm_optimization(particles, num_iterations,
inertia_weight, cognitive_coefficient, social_coefficient, mask_matrix)
# 输出结果
print("最佳分配方案：\n", best_position)
print("最佳目标函数值：", best_value)

# 使用最佳分配方案计算货量发生变化的线路数、不能正常流转的货量及网络的负荷情况
changed_lines = np.sum(best_position != initial_matrix)
stuck_goods = np.sum(best_position.iloc[:, 0])
target_loads = best_position.sum(axis=0)
load_variance = np.var(target_loads)

print("货量发生变化的线路数：", changed_lines)
print("不能正常流转的货量：", stuck_goods)
print("网络负荷情况（方差）：", load_variance)

# 可视化结果
plt.plot(global_best_values)
plt.xlabel("Iteration")
plt.ylabel("Global Best Value")
```

```
plt.title("PSO Convergence")
plt.show()

# 计算每个目标场地的总货量
target_loads = best_position.sum(axis=0)

# 计算方差
load_variance = np.var(target_loads)

# 绘制负荷情况图
plt.figure(figsize=(10, 6))
plt.bar(target_loads.index, target_loads.values)
plt.axhline(y=target_loads.mean(), color='r', linestyle='--', label='Average Load')
plt.xlabel('Target Nodes')
plt.ylabel('Load')
plt.title(f'Load Distribution (Variance: {load_variance:.2f})')
plt.legend()
plt.show()
```