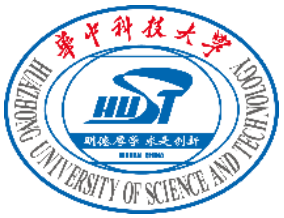


中国研究生创新实践系列大赛

中国光谷·“华为杯”第十九届中国研究生数学建模竞赛

学　　校	安徽大学
参赛队号	22103570053
队员姓名	1.吴建平
	2.李婧然
	3.王芮

中国研究生创新实践系列大赛

中国光谷·“华为杯”第十九届中国研究生数学建模竞赛

题 目　　COVID-19 疫情期间生活物资的科学管理问题

摘　　要：

针对问题一，为了直观地描述并判别分析出长春市实行发放蔬菜包前后的效果，本文先后分析了新增病毒感染人数、累计感染人数和感染人数的变化率，并且在此基础上继续细化，分为总计感染人数、本土确诊人数和无症状感染者这三个部分。通过绘制病毒感染人数及其变化率的图像，可以直观比较得出疫情的发展或被控制扑灭与生活物资发放方式之间的联系。通过使用 **ARIMA** 进行时间序列预测，使用 **ADF** 检验其平稳性，得出如果未实行蔬菜包这一生活物资管理方式的感染人数结果，并结合使用 **SEIR** 传染病数学模型，构建非线性常微分方程，验证 ARIMA 模型预测结果的合理性。

针对问题二，首先分别使用**小区数量、路网密度、隔离人数**以及**蔬菜包接受量**等评价因素对 9 个区域投放点数量进行合理性分析，然后根据问题一求解的不同区域疫情防控效果指标择优选取目标区域，建立**评价因素与投放点数量的多元方程组**对投放点数量进行合理优化，结果见表 5.2.2。接下来使用 **CFLP 多中心选址模型**，并通过**系统聚类**和 **LU 分解**确定最优选址数量和规模大小，结果见表 5.2.3，最后结合各区域小区数量和隔离人口数等因素，确定备用场所位置，进结果见表 5.2.4。

针对问题三，首先以**描述统计、相关性分析、MD-Apriori** 及供求之间的关系及**改进的模糊关联规则挖掘算法**深度分析蔬菜包的需求、发放规律。其次，提取潜在扩散风险、接收反应能力、疫情扩散强度三大指标，采用**层次分析法、熵权-TOPSIS 方法**，求解各地区发放蔬菜包的优先度作为评价标准之一。再构建 **ARIMA 模型**，以过往供应数据结合所分析蔬菜包需求规律对 4 月 10 日至 15 日的需求进行预测得到需求量，以各区人口数占比为权重，得到 9 个区的蔬菜包投放量，再以附件 3 中各区的管辖小区为依据，以各小区人口占比权重为基础，分别计算各小区蔬菜包供应量作为优化方案。

针对问题四，首先将 9 个区域**重心点**作为初始物资来源地，并通过问题二优化的各区域投放点数量设置初始集散地数量，进一步地通过 **k-means 聚类**确定各个集散地所管辖的小区，完成有序网络图的初步搭建。然后构建**图优化模型**，并通过**粒子群算法**和 **k-means 算法**对集散地位置进行优化，通过得出集散地位置信息计算 9 个区域最优集散地数量、工作量和物资来源点位置，结果见表 5.4.1，并以宽城区为例对有序图进行分析，结果见图 5.4.3。接下来考虑街区的实际情况并结合蔬菜的种类和价格变化对居民的生活影响以及减少人员的接触等因素，使用**哈密顿距离**求解**图优化模型**，结果见表 5.4.2。最后，进一步考虑使

用卡车运送物资，对预案进行分析，结果见表 5.4.3。

关键词：CFLP 多中心选址模型；ARIMA 模型；粒子群算法；图优化模型

目　　录

一、问题重述

1.1 问题背景

进入 2022 年以来全国范围内陆续出现了多次较大规模疫情爆发事件[1-2]，无论是 2022 年 3 月初在吉林省长春市大规模爆发的新型冠状病毒疫情（COVID-19），还是 4 月初在上海及 5 月初在北京大规模爆发的疫情，均显示出了一些普遍存在的难点问题。在大规模疫情爆发期间由于我国采用封闭式管理方式来实现疫情的快速清零，从而疫情期间各类人群的科学管理变得尤为重要[3-4]。

图 1.1.1 我国近期疫情发生情况

考虑到近期或未来仍然有部分省份存在潜在的大规模爆发风险，且大部分地区管理者尚未遇见如此大规模爆发的疫情及不同地区疫情爆发时长、人口规模与地理位置等的差异性，众多地区仍未形成较为科学的统一管理模式，使得疫情的清零周期普遍较长。在对大规模的人员采用封闭式管理的情况下，关联地区的经济发展持续放缓，再考虑到全球经济形势及疫情等因素所导致的经济下行压力[5]，我们急需形成规范化的疫情快速清零机制，并努力为有效降低大规模疫情防控成本、缩短疫情防控周期及保障各省经济的稳定可持续发展方面提供可靠的依据。

在疫情期间由于对确诊病例、无症状感染者及密切接触者采用集中治疗或隔离的手段，因此此类人群的科学管理较易实现。然而，对于大量、分散居家隔离人员实现统一化管理成为新的难点问题，目前我们已知的典型问题包括三个方面：

（1）生活物资的科学发放问题；

（2）特殊群体的管理问题；

（3）为援助与医务人员提供高效服务问题。

本文聚焦解决疫情期间生活物资的科学管理问题，在疫情期间由于采用封闭式管理方式，居民的生活物资发放成为所有问题中的焦点。如果发放不当不仅会影响隔离居民的生活，还很有可能在生活物资发放过程中造成疫情的二次传播。因此，在全面提高疫

情防控效率的同时，如何调控好生活物资的大规模流动以及其科学有序发放成为了亟需解决的问题。

1.2 问题提出

本文主要考虑以下几个问题：

问题一：根据长春市疫情期间的感染者历史数据及每日各区蔬菜包投放的数据，分析疫情的发展或受控与生活物资的大规模流动方式之间所存在的关联性。

问题二：第一问：对照长春市的交通路线情况及小区分布，分析各区域投放点数量的合理性；通过数学建模对投放点数量进行优化。第二问：合理规划用于储备物资、分拣物资的场所的位置与数量规模（选址半径、管辖范围内小区个数及人口个数）等，提出最优选项及备用选项。

问题三：根据每日各区蔬菜包相关数据分析蔬菜包的需求及发放规律；结合各小区位置与人口信息，对 4 月 10 日至 4 月 15 日的蔬菜包供应方案进行评价及调整。

问题四：以有序网络图的形式建立特殊时期物资供应的详细预案并分析评价其优势，网络图上游为物资来源，中游为物资集散地，下游为所有小区。完成有序网络图后考虑用大、小卡车运送物资时预案有无显著不同。

二、问题分析

关于问题一，本文选取长春市的 9 个区数据，分别为朝阳区、二道区、经开区、净月区、宽城区、绿园区、南关区、汽开区、长春新区（高新）。为了直观比较疫情的发展或被控制扑灭与生活物资发放方式之间的联系，本文分别统计各地区病毒感染人数数据，包括新增本土感染者和新增无症状感染者，以及各地区蔬菜包的统计数据。再此基础上再分别去统计全市的病毒感染人数数据，包括新增本土感染者和新增无症状感染者，以及全市的蔬菜包数据。通过使用 ARIMA 进行时间序列预测，得出如果未实行蔬菜包这一生活物资管理方式的感染人数结果，并结合使用 SEIR 传染病数学模型，构建非线性常微分方程，验证 ARIMA 模型预测结果的合理性。如何建立和求解非线性常微分方程，使用 SEIR 进行预测是需要解决的问题。

关于问题二，

关于问题三，首先对附件 5 中信息进行整合重组以方便后续使用。分析蔬菜包需求、发放规律，首先我们从蔬菜包需求、发放之间的内在联系着手，再从感染人数、蔬菜价格等各个维度，采用描述性统计、相关性分析、关联信息挖掘等方法，分析影响蔬菜包需求、发放的因素。为了做出合适的评价，我们提取出适宜的评价指标，通过层次分析、熵权-TOPSIS 方法评价供应方案，再运用 ARIMA 模型进行调整，以 9 个区的人口占比为权重，分别计算出各区蔬菜包投放数量，再由各小区人口在管辖区内人口占比为

权重，计算出最终的各小区蔬菜包供应方案。

关于问题四，为了应对疫情突发，方便物资的配送，建立有序网络图来对配送方案进行优化求解；为了尽可能的减少人力成本，要求工作量尽可能的小，(对配送集散地，物资来源地进行选址优化)，同时还需减少人员的直接接触和间接接触，要求居民尽可能减少外出，非必要不在超市采购物资，因此需要进一步结合小区位置对集散地进行选址，并考虑集散地数量的增多可以减少居民外出采购，同时还需考虑蔬菜的种类和价格变化对居民的生活影响。进一步地，将两点欧拉距离替换为实际的街区距离进行有序网络图规划，确定更为合理的居民生活物资发放预案；考虑实际情况，物资运输可能需要多次运输才可以送到集散地再对小区进行发放，因此接下来分别考虑用最大装载量为 10 吨的大卡车运输物资和最大装载量为 4 吨的小卡车运输物资对预案结果的影响。

三、模型假设与符号说明

3.1 模型假设

A 假设疫情期间采取封控措施，无人口流动，不考虑自然出生率、自然死亡率

B 假设需求点和物资储备设施点呈点状分布

C 假设应急设施点的物资供应能力为无限大，即应急需求点能被最近的应急设施点所覆盖

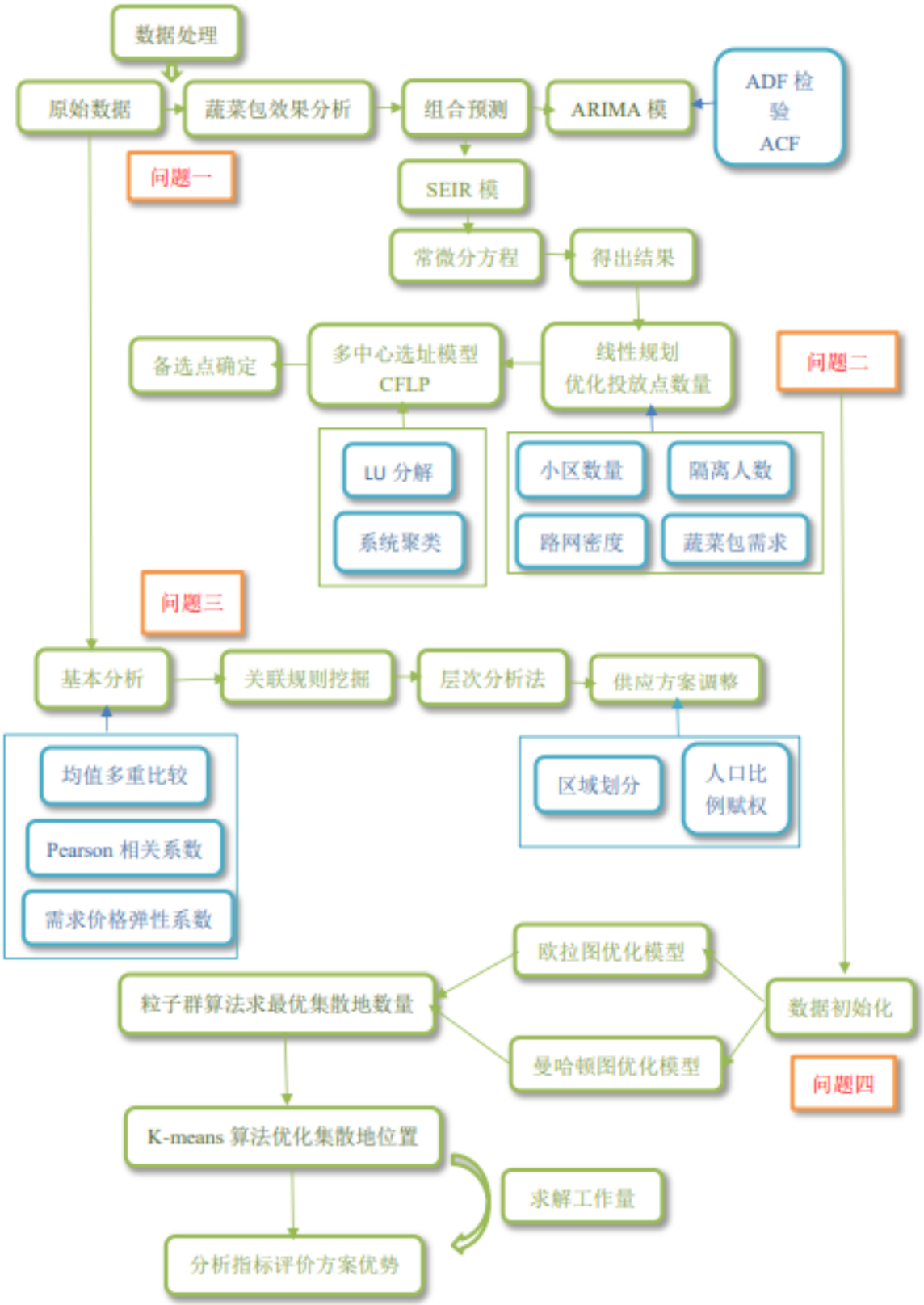
D 假设路网中每条路线无施工、损毁等情况，能够容纳卡车通过

3.2 符号说明

符号	说明
NSD_i	小区数量
RND_i	路网面积
NIP_i	隔离人数
VPA_i	蔬菜包接收量
I_i	平均新增感染率
n	最优分拣中心的数量
m	小区总数
n_0	可建分拣中心的最大个数

d_0	最大投放半径
d_{ij}	第i个分拣中心到第j个小区的距离
ps_i	第i个分拣中心的位置
pe_j	第j个小区的位置
R	人口密度，人口数与地区面积的比值
Q	小区密度，小区数量与地区面积的比值
T	本土感染者占总人数的比率
E	无症状感染者占总人数的比率

四、论文框架图


数据处理
原始数据
蔬菜包效果分析
组合预测
ARIMA 模
ADF 检验
ACF
问题一
SEIR 模
常微分方程
得出结果
备选点确定
多中心选址模型
CFLP
线性规划
优化投放点数量
问题二
LU 分解
系统聚类
小区数量
隔离人数
路网密度
蔬菜包需求
问题三
基本分析
关联规则挖掘
层次分析法
供应方案调整
均值多重比较
Pearson 相关系数
需求价格弹性系数
区域划分
人口比例赋权
欧拉图优化模型
粒子群算法求最优集散地数量
数据初始化
曼哈顿图优化模型
问题四
K-means 算法优化集散地位置
求解工作量
分析指标评价方案优势

五、模型的建立与求解

5.1 问题一的模型建立与求解

5.1.1 问题分析

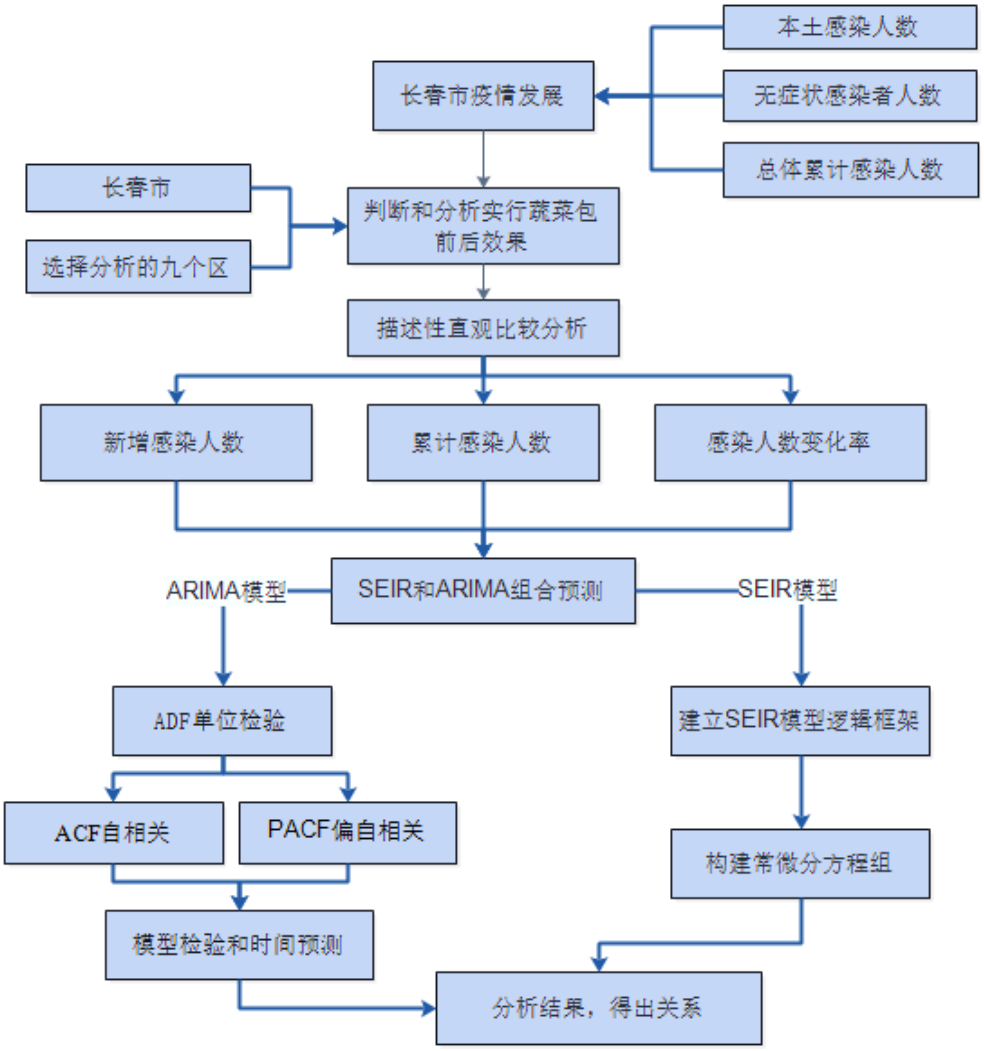


图 5.1.1 问题一总体分析流程图

为了判断和分析实行蔬菜包前后效果，本文先是建立图表直观表现新增病毒感染人数、累计感染人数和感染人数的变化率，并且在此基础上继续细化，分为总计感染人数、本土确诊人数和无症状感染者这三个部分。通过对已有的数据进行分析并绘制相关图表，本文发现疫情的发展或被控制扑灭与生活物资发放方式有关。为了进一步证明分析的可靠性，本文采用 SEIR 模型和 ARIMA 模型组合预测进行若未实行蔬菜包疫情的感染人数。通过对比两组数据得出结果。

5.1.2 数据来源及预处理

5.1.2.1 数据来源

本题中的数据主要来源于附件 1 和附件 5，附件 1 为长春市及其各区每日的新增本土感染者人数以及新增无症状感染者人数，时间跨度为 2022/03/04-2022/05/23/；注意到问题一要研究蔬菜包发放前后疫情控制效果进行分析，故对附件 5 中数据不能直接使用，将其整合成一个完整的时间序列数据集，时间跨度为 2022/03/26-2022/05/01。

5.1.2.2 数据预处理

预处理主要包括对数据中的缺失值、异常值进行处理，对变量指标无量纲化等。具体处理方法及步骤如下：

a）异常数据的挖掘与处理

所谓数据挖掘是指按照既定目标，对大量的统计数据进行探索，揭示其隐藏的规律并将之模型化的一种先进有效的方法，其实质是对海量数据的识别过程。识别统计数据的第一步就是检验异常值的存在，常用的异常值挖掘方法有3σ检测法、聚类分析法和回归分析等。

综合比较几种检验方法后选择效果更好的3σ准则检验异常值。在该准则下，若统计数据没有明显的上升或下降趋势，且都分布在其均值周围，标准差σ就能很好地反映其离散程度。统计数据若是来自某一总体的样本，要求统计数据与其均值的偏差超过3σ的比例不超过 1/9，否则将那些均值之差的绝对值超过3σ的统计数据视为异常数据。常用的检验异常值是否剔除的方法有 t 检验、拉依达准则（3σ准则）、拉格布斯准则和迪克逊准则等，本题采用3σ准则剔除异常值。即根据每个样本点距离样本中心的距离判断是否将可疑数据x_d从该组数据中剔除，至于选择 3 倍的标准差（即3σ）还是 2 倍的标准差与显著性水平α有关。显著性水平α表示的是检验出错的几率为α，或检验的置信度为 1-α，3σ相当于显著性水平为 0.01，2σ相当于显著性水平为 0.05。

b）缺失值的处理

本数据库中缺失数据较多，本题的数据库检测发现，部分时间各区域的变量都有较多的缺失值，而数据缺失很可能对分析发放蔬菜包对疫情控制的效果有较大影响，因此本部分的处理相当重要。处理缺失值的方法主要有删除元组、数据补齐、不处理。例如九台区新增本土感染者人数在 4 月 13 日以后全部缺失，对这样的缺失值数据我们采取的是数据补齐的方式。首先根据别其他相关字段进行预估，如全市新增本土感染者。若无法根据相关字段进行填充的，使用了 K 最近距离邻法，即根据欧式距离或相关分析来确定距离具有缺失数据样本最近的 K 个样本，将这 K 个值加权平均来估计该样本的缺失数据。

5.1.3 数据处理和分析

5.1.3.1 病毒感染人数的数据处理和分析

为了直观地描述并比较判别分析出长春市实行发放蔬菜包前后的效果，本文先后分析了新增病毒感染人数、累计感染人数和感染人数的变化率，并且在此基础上继续细化，分为总计感染人数、本土确诊人数和无症状感染者这三个部分。通过绘制病毒感染

人数及其变化率的图像，可以直观比较得出疫情的发展或被控制扑灭与生活物资发放方式之间的联系。通过使用 ARIMA 进行时间序列预测，得出如果未实行蔬菜包这一生活物资管理方式的感染人数结果，并结合使用 SEIR 传染病数学模型，构建非线性常微分方程，验证 ARIMA 模型预测结果的合理性。

a) 全市病毒感染人数

全市疫情发展时间段为 3 月 4 日到 5 月 23 日，共持续 81 天。5 月 2 日全市新增感染者人数归零，5 月 3 日到 5 月 7 日发生极小范围波动，新增两三例新增感染者病例。5 月 8 日以后，全市无新增感染者。

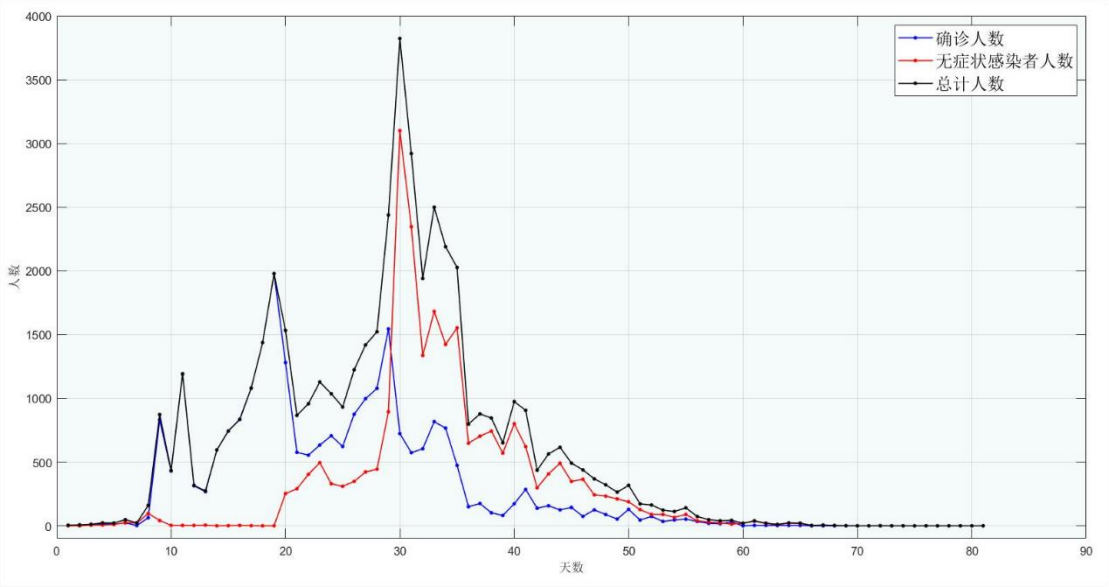


图 5.1.2 全市病毒感染人数

由图 2 得，全市无症状感染者变化可以分为两个阶段，第一阶段时间段为 3 月 6 日到 3 月 12 日，第二阶段时间段为 3 月 23 日到 5 月 10 日，其中全市新增无症状感染者集中爆发时间段为 4 月 1 日到 4 月 13 日，最高新增无症状感染者 2346 例。

b) 各地区病毒感染人数

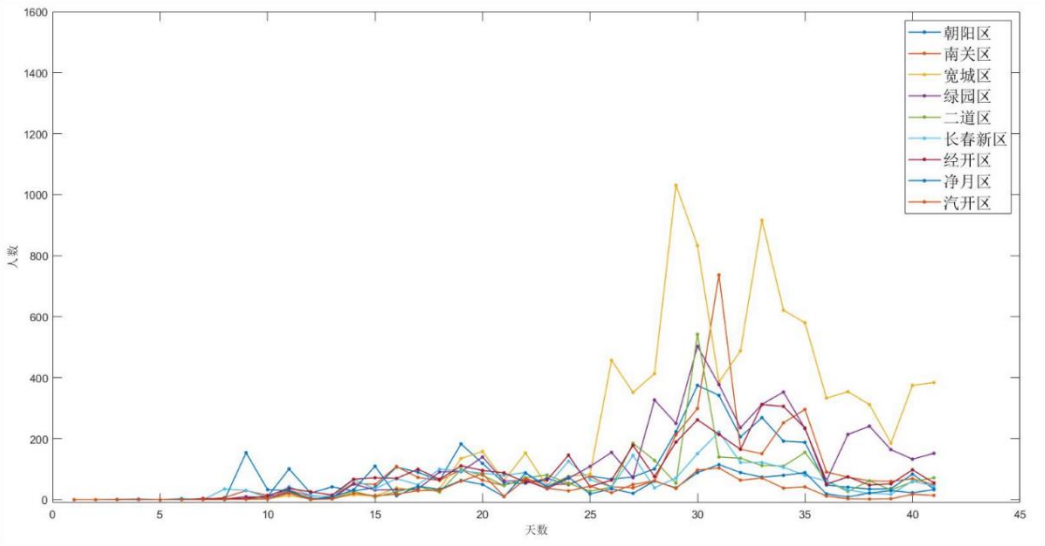


图 5.1.3 各地区感染者总数变化曲线

3 月 4 日到 4 月 13 日，共持续 41 天。统计随着疫情的不断扩散，各个地区每天新增的确诊人数，包括新增的本土感染者和新增的无症状感染者。3 月 12 日到 4 月 13 日为各地区新增感染者的爆发阶段，其中宽城区疫情爆发情况尤为严重，8 月 27 日新增感染者 1000 余例。净月区和汽开区的防疫压力较小。

c) 各地区无症状感者人数

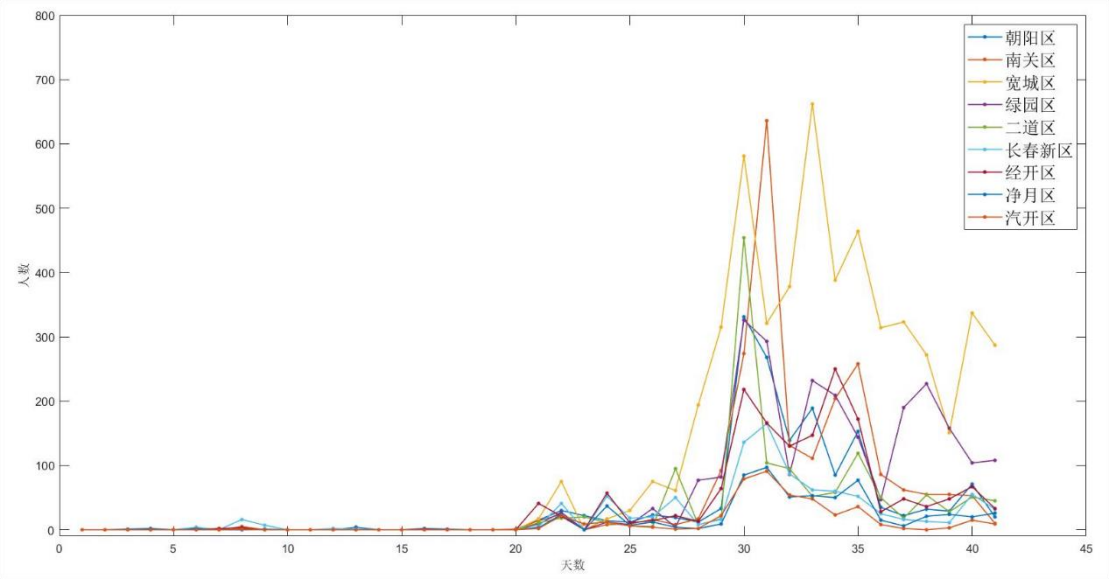


图 5.1.4 各地区无症状感染者变化曲线

3 月 24 日到 4 月 13 日为各地区新增无症状感染者的爆发阶段。其中宽城区的疫情爆发情况尤为严重，4 月 5 日新增无症状感染者 662 人。从 4 月 1 日到 4 月 9 日，宽城区持续 9 天新增无症状感染者超过 300 人。汽开区的防疫压力较小，4 月 3 日新增无症状感染 91 人，为疫情期间的最大值。

d) 各地区确诊人数

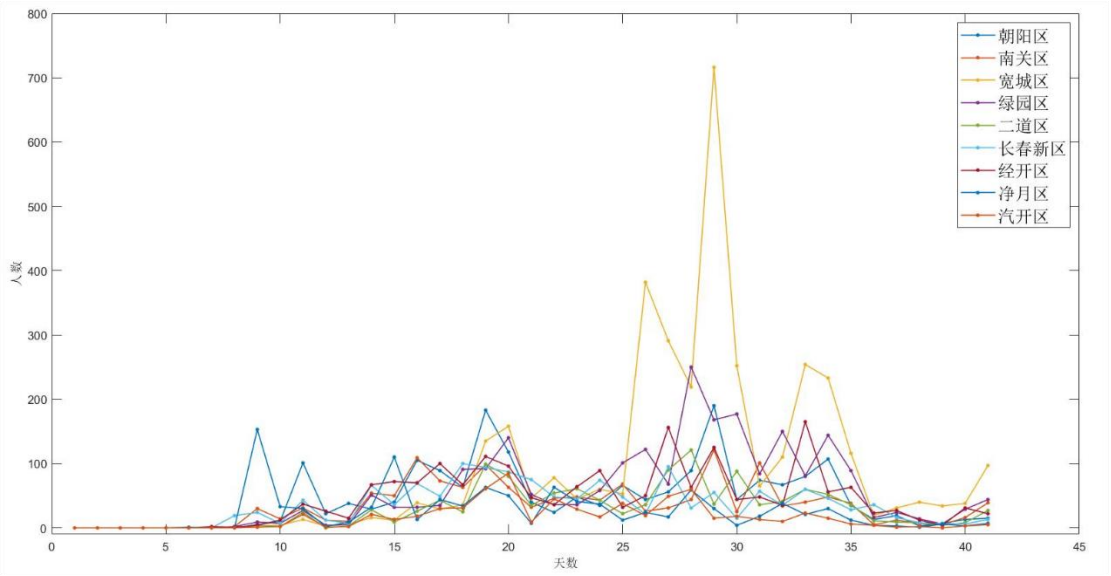


图 5.1.5 各地区确诊人数变化曲线

由图 5.1.5 可知，各地区在疫情开始的前八天无新增确诊病例，第 8 天到第 25 天，各地区确诊人数开始不断增加，其中朝阳区疫情防控形势日渐严峻，最高一天新增确诊人数接近 200 人。第 25 天到第 36 天，疫情继续蔓延，新增确诊人数达到最大值，其中宽城区的最高确诊人数超过了 700 人，防疫工作进入最紧张的阶段。

5.1.3.2 累计病毒感染人数的数据处理和分析

a) 长春市累计病毒感染人数

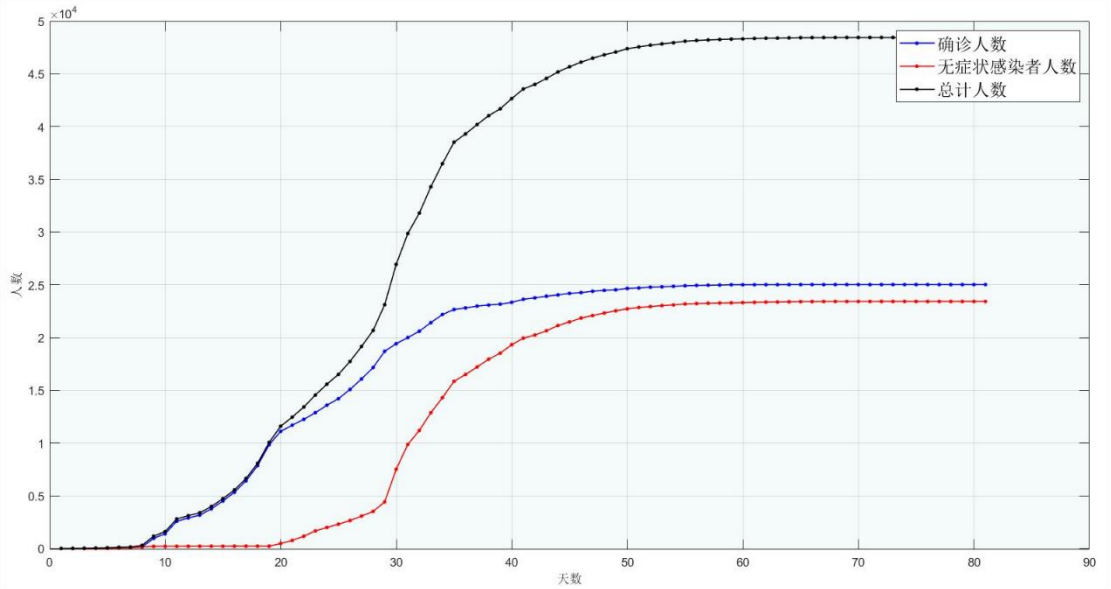


图 5.1.6 全市累计感染人数

从 3 月 4 日到 5 月 23 日，全市疫情累确认人数接近 49000 人，包括新增的本土感染者和新增的无症状感染者。在疫情传播初期，3 月 4 日到 3 月 23 日，全市无症状感染者仅为 100 余人，降低本土确诊病例数目为是控制疫情传播的主要途径。从第 30 天开始，

无症状感染者人数开始快速增长，直到第 50 天疫情状况趋于平缓。

b) 长春市九个地区累计确诊人数

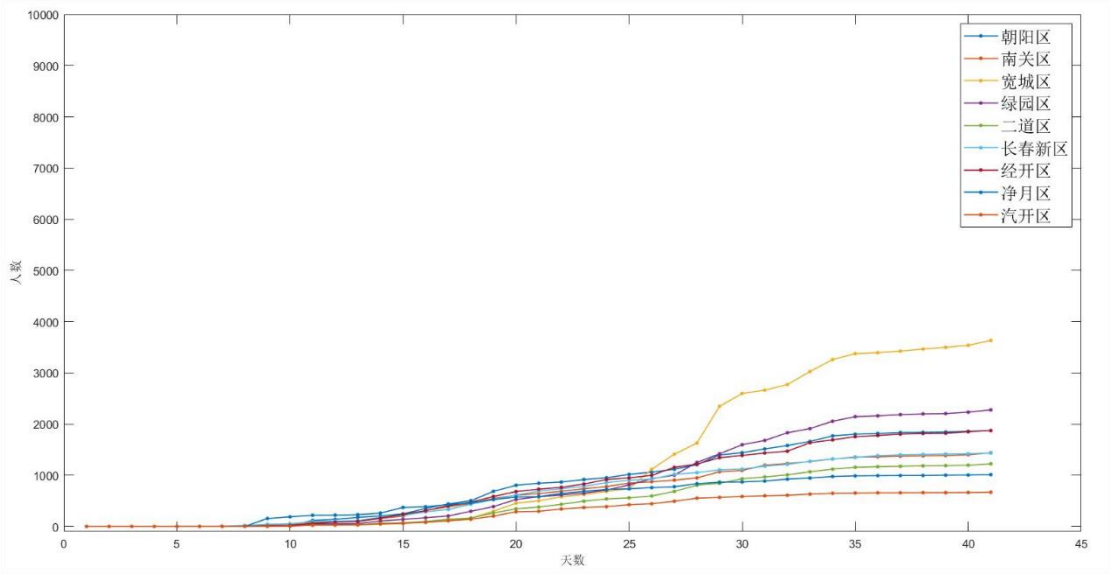


图 5.1.7 各地区累计确诊人数变化曲线

长春市疫情爆发的前 8 天各个地区累计确诊人数较少，从 3 月 13 日开始各个地区的累计确诊人数开始增加。从 4 月 1 日开始，宽城区的确诊人数开始以较大幅度增长，其他地区的确诊人数也开始快速增加。

c) 长春市九个地区累计病毒感染人数

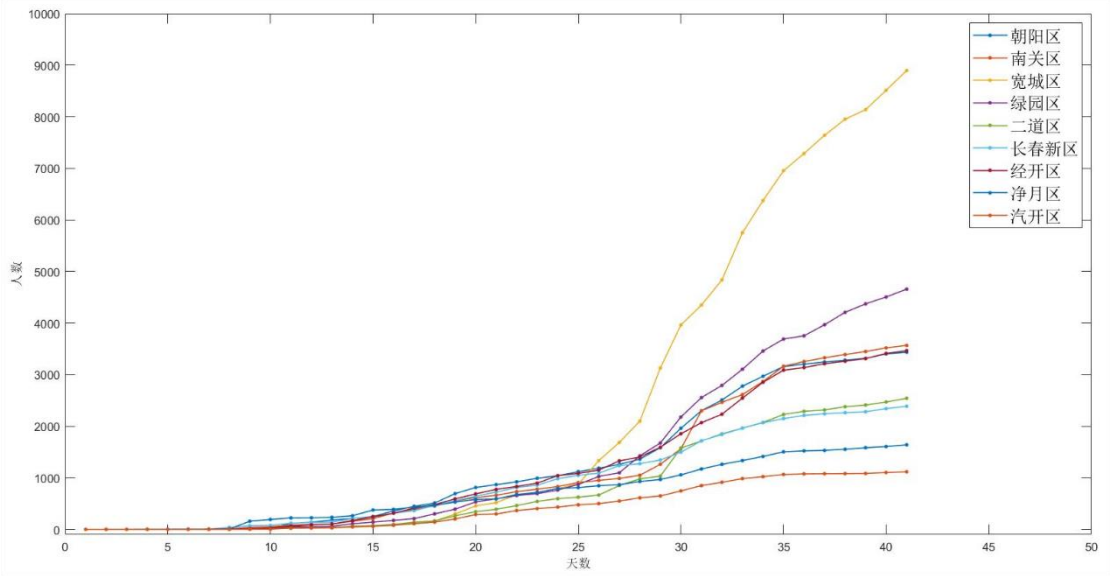


图 5.1.8 各地区累计感染者总数变化曲线

整个疫情爆发期间，各地区新冠肺炎累计感染者总数不断增加，汽开区感染者人数变化较慢，宽城区感染者人数变化较快。南关区、净月区和经开区感染者人数变化程度相似，二道区和长春新区变化程度相似。

d) 长春市九个地区累计无症状感染者人数

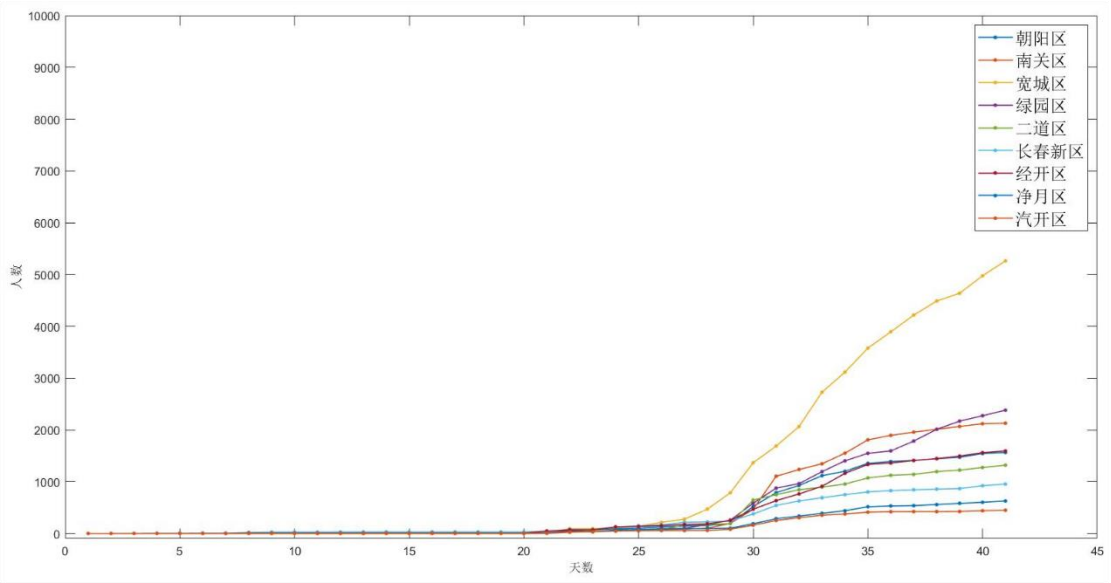


图 5.1.9 各地区累计无症状感染者人数变化曲线

长春市在新冠病毒疫情刚爆发初期时段，无症状感染者并没有呈现较大幅度增加，各个地区的无症状感染者人数都维持在较低的水平，但在疫情爆发的后期，各地区无症状感染者人数不断增加，在宽城区的表现非常明显，从疫情爆发的第 28 天开始，长春市宽城区的累计无症状感染者平均增长率为 43.6%。

5.1.3.3 感染人数变化率的数据处理和分析

a) 长春市全市感染人数变化率曲线

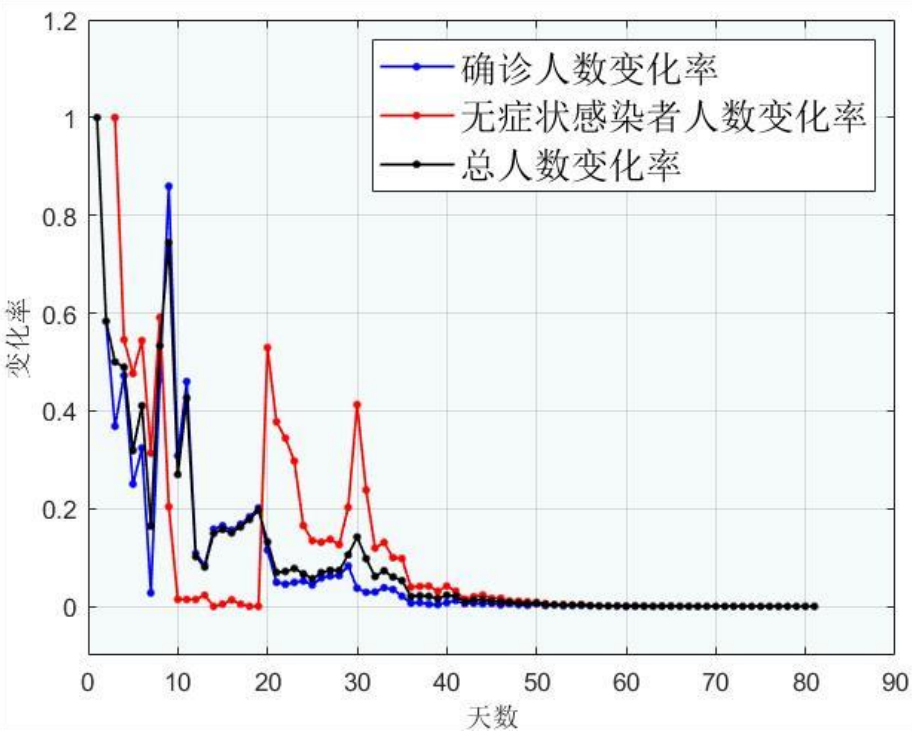


图 5.1.10 全市感染人数变化率曲线

从总体上看，全市感染人数、确诊人数、无症状感染者人数变化率随着时间的推进，由大变小，第 40 天以后变化率逐渐趋于平缓，不在发生较大的变化。3 月 26 日，也就是疫情发生的第 22 天，长春市开始实行发放蔬菜包，发放蔬菜包前，确诊人数变化率的平均值为 47.9%，无症状感染者人数变化率的平均值是 45.8%，总人数变化率的平均值为 39.6%。3 月 26 日开始发放蔬菜包以后，确诊人数、无症状感染者和总人数的变化率都呈现逐渐减少的趋势。3 月 26 日到 4 月 12 日，由于采用了蔬菜包这一新的生活物资科学管理模式，新增新冠肺炎患者人数不断减少。无症状感染者人数变化率的平均值为 11.3%，总人数变化率平均值为 7.9%，确诊人数变化率的平均值为 4.32%。4 月 12 日以后，三者变化率都趋向于 0，疫情形势好转。

b) 长春市九个地区感染人数变化率曲线

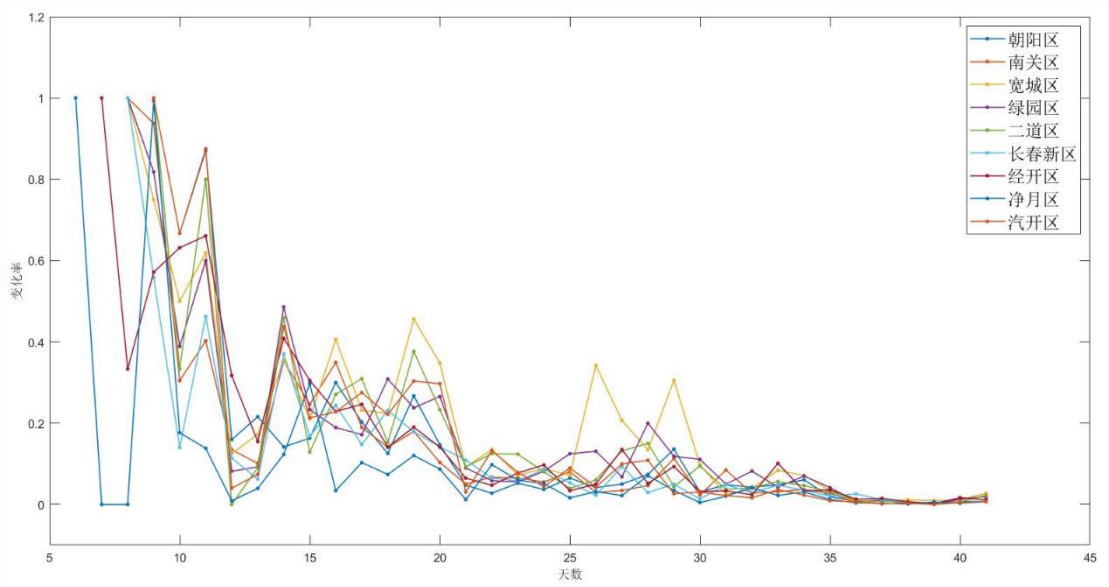


图 5.1.11 长春市 9 个区感染人数变化率曲线

统计长春市 9 个区从 3 月 4 日到 4 月 13 日新增和累计感染人数的计算数据，由上图可知，随着时间的推移，总体上来看，9 个区的感染人数变化率逐渐降低，趋于平缓，最终为变为零。从 3 月 4 日到 3 月 26 日，长春市未采用蔬菜包这一生活物资科学管理模式阶段，各个地区每日新增感染人数变化率与 3 月 26 日以后发放蔬菜包相比较大。其中，长春市朝阳区和经开区的变化率平均值变化较明显，朝阳区变化率平均值由 58.78%变为 17.65%，经开区变化率平均值由 48.8%变为 9.6%。

5.1.4 模型的建立和求解

5.1.4.1 ARIMA 模型的建立

ARIMA 模型是时间序列预测模型之一，其建模思想是，将预测对象随时间推移而形成的数据序列看作一个随机序列，时间序列是一组依赖时间的随机变量，构成该时间序列的单个序列值虽然具有不确定性，但整个序列的变化却是有一定的规律性，可以用数学模型近似描述。通过 ARIMA 模型可以消除趋势性，转化为平稳时间序列进行建模。

为了能够准确预测若长春市未实行发放蔬菜包，疫情期间病毒感染人数的变化，本文建立 ARIMA 模型。该模型要求时间序列必须是平稳的，所以第一步是对原始数据进行平稳性检验。通常的检验方法有很多种，包括 ADF、KPSS 等，本文采用 ADF 单位根平稳型检验。

a) ADF 单位根平稳型检验

ADF 平稳型检验是一种经典的单位根检验方法，用于检验 AR 序列是否平稳。其主要原理是判断序列是否存在单位根。若序列平稳，则不会存在单位根；反之存在。

其 H_0 假设为存在单位根。当得到的显著性检验统计量小于三个置信度（10%，5%，1%）时，对应有（90%，95%，99%）的把握拒绝原假设。

表 5.1.1 ADF 单位根平稳性检验结果

变量	差分阶数	*t*	*p*	AIC	临界值		
					1%	5%	10%
全市总计	0	-2.521	0.110	575.626	-3.563	-2.919	-2.597
	1	-5.43	0.0009	563.653	-3.568	-2.921	-2.599
	2	-5.693	0.0008	557.499	-3.597	-2.933	-2.605

该序列检验的结果显示，基于字段全市病毒感染人数总计:在差分为 0 阶时，显著性 P 值为 0.110，水平上不要呈现显著性，不能拒绝原假设，该序列为不平稳的时间序列。在差分为 1 阶时，显著性 P 值为 0.0009，水平上呈现显著性，拒绝原假设，该序列为平稳的时间序列。在差分为 2 阶时，显著性 P 值为 0.0008，水平上呈现显著性，拒绝原假设，该序列为平稳的时间序列。因此，该序列是平稳的。

b) 确定 ARIMA 模型阶数

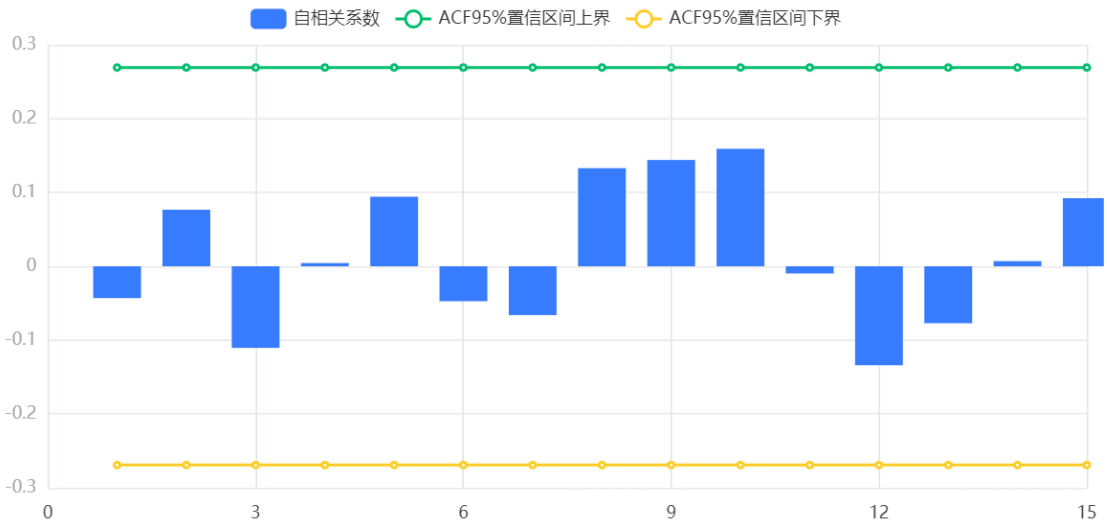


图 5.1.12 模型残差自相关图（ACF）

自相关（ACF）是指序列与其自身经过某些阶数滞后形成的序列之间存在某种程度的相关性，而偏自相关函数（PACF）是在其他序列给定情况下的两序列条件相关性的度量

函数。一般来说（偏）自相关用于时间序列分析 AR、MA 的 p、q 进行定阶。由图可知，一阶和三阶自相关系数很明显地大于 2 倍标准差范围，且自相关系数衰减为小值波动的过程比较缓慢或非常连续，我们可以判断自相关图为拖尾。

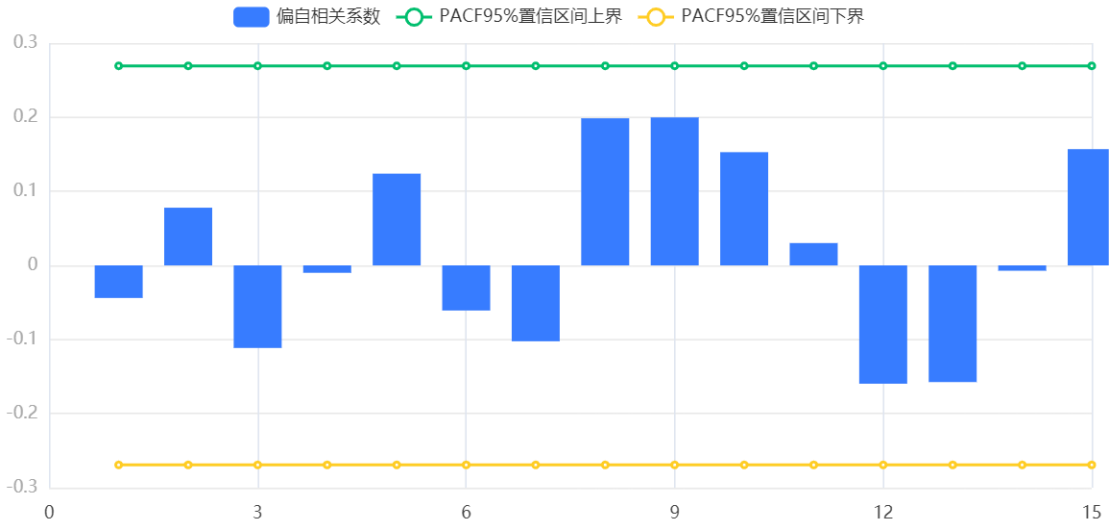


图 5.1.13 模型残差偏自相关图（PACF）

上图展示了模型的残差偏自相关图（PACF），包括系数，置信上限和置信下限。若相关系数均在虚线内，滑动平均模型（MA）残差为白噪声序列，时间序列要求模型残差为白噪声序列。一阶和二阶偏自相关系数很明显地大于 2 倍标准差范围，自一阶偏自相关系数后，其余偏自相关系数都在 2 倍标准差范围以内，且二阶后偏自相关系数衰减为在零附近小值波动的过程非常突然。我们可以判断偏自相关图为截尾。

c) 模型的检验

表 5.1.2 模型检验所得参数结果

	系数	标准差	t	$p>\|t\|$	0.025	0.975
常数	416.159	170.643	2.439	0.015	81.704	750.614
全市感染人数总计	0.776	0.085	9.165	0	0.61	0.942

由上图可知，模型结果参考 ARIMA 模型（1,0,0）检验表且基于 1 差分数据，模型的标准差为 0.085<0.1，表明模型选择具有合理性。

5.1.4.2 ARIMA 模型的求解

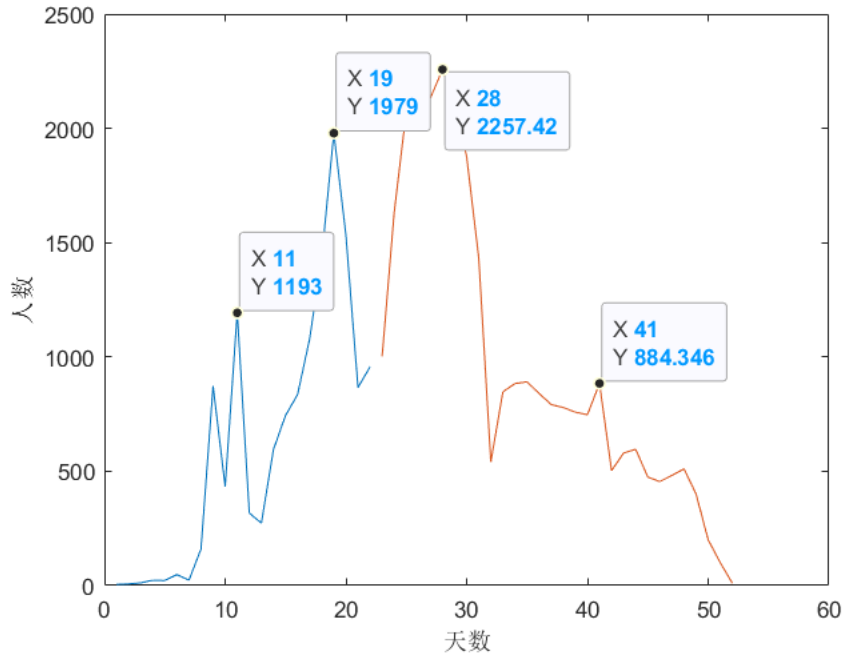


图 5.1.14 模型预测结果值

基于 Matlab 中工具函数求解若不实行发放蔬菜包的效果如上图所示，随着时间的推移，感染人数最终会逐渐变小，最终不存在病毒感染者，疫情结束。但为了提高预测结果的合理可靠性，本文在 ARIMA 的基础上使用 SEIR 模型再次进行预测，综合得出预测结果并进行对比分析。

5.1.4.3 SEIR 传染病模型的建立

在一个简单的 Susceptible(易感者)-Exposed(潜伏者)-Infected(感染者)-Removed(康复者)(SEIR)模型中，Susceptible(易感者)是指没有感染过此传染病，迄今健康的人群，也就是大部分人群，宅在家的我们都属于这一类人群，Exposed(潜伏者)是正在此传染病潜伏期的人群，Infected(感染者)是已经确诊的人群，Removed(康复者)是移除的人群(包括康复的人和死亡的人)。

根据 SEIR 传染病模型的特点，本文搭建相应的逻辑模型框架。

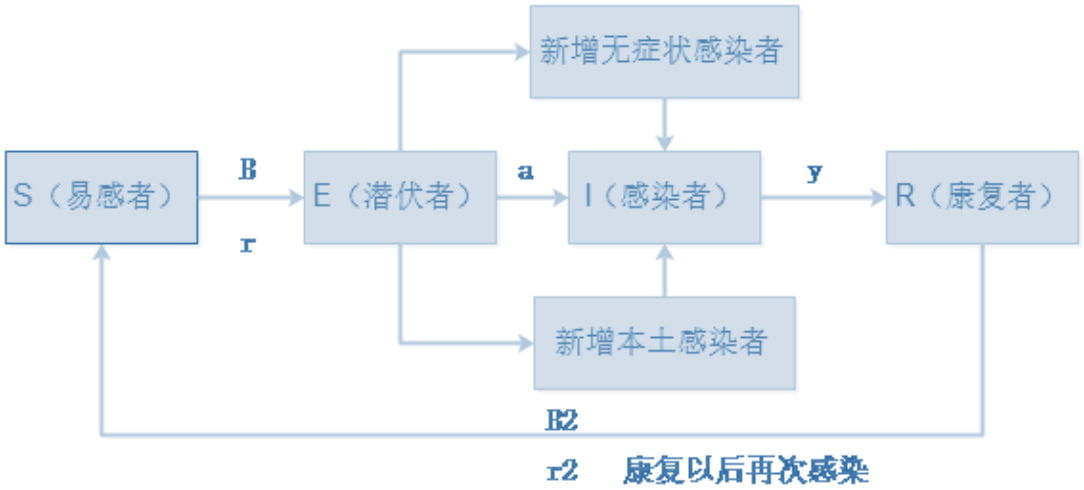


图 5.1.15 SEIR 传染病模型框架

其中 B 为一个易感者和一个感染者接触，他被传染的概率。r 为感染者接触易感者的人数。B_2 为康复者和一个感染者接触，他被再次传染的概率。r_2 为康复者再次被病毒感染变为感染者，然后接触易感者的人数。a 为潜伏者转化为感染者的概率，它可以估计为已知的平均潜伏期 Y 的倒数，即 $a=1/Y$。γ 为感染者康复的概率，可以由平均的康复期 D 的倒数决定，即 $\gamma=1/D$。以上参数均大于 0。其中，

表 5.1.3 参数表

$S=N-I$
$R=0$
$r=20$
$B=0.03$
$a=1.01$
$\gamma=0.99$
$r_2=20$
$B_2=0.03$

5.1.4.4 SEIR 传染病模型的求解

一个病毒在一个群体的传播便可以由非线性常微分方程来描述，由上述模型框架可得一个常微分方程。

$$\begin{cases} S(dx_{i+1})=S(dx_i)-\dfrac{r*B*S(dx_i)*I(dx_i)}{N(i)}-\dfrac{r_2*B_2*S(dx_i)*E(dx_i)}{N(i)} \\ E(dx_{i+1})=E(dx_i)+\dfrac{r*B*S(dx_i)}{N(i)-a*E(dx_i)}+\dfrac{r_2*B_2*S(dx_i)*E(dx_i)}{N(i)} \\ I(dx_{i+1})=I(dx_i)+a*E(dx_i)-y*I(dx_i) \\ R(dx_{i+1})=R(dx_i)+y*I(dx_i) \end{cases} \tag{5.1.1}$$

其中 $S+E+I+R=N$,以上的方程分别表示了 S 、 E 、 I 、 R 随时间的变化率。有了

以上的方程，我们就可以把时间变化量设为 1 天，即可依次推出四种人群的变化曲线。

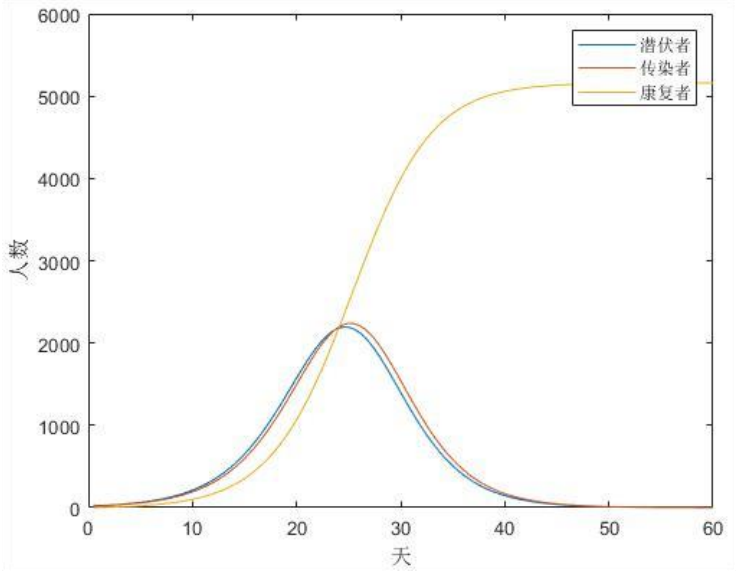


图 5.1.16 四类人群数量随时间的变化曲线

感染者若不受隔离每天出去接触人，潜伏者变为感染者的概率 a 为 1.01，可以看到随时间推移，潜伏者会先达到顶峰，之后随着潜伏者发病成为传染者，传染者后达到增长拐点，随着治疗和自身免疫，潜伏者和传染者人数降低，康复者人数增加。

表 5.1.4 预测感染和实际感染人数

日期	预测感染总数	实际感染人数	增长比例
4 月 26 日	1202	1128	0.065603
4 月 27 日	1632	1036	0.57529
4 月 28 日	2073	932	1.224249
4 月 29 日	2099	1223	0.716271
4 月 30 日	2132	1419	0.502467
5 月 1 日	2257	1522	0.482917
5 月 2 日	2487	2438	0.020098
5 月 3 日	4875	3823	0.275177
5 月 4 日	4433	2920	0.518151
5 月 5 日	3540	1940	0.824742
5 月 6 日	3847	2499	0.539416
5 月 7 日	3783	2189	0.728186
5 月 8 日	2891	2027	0.426246
5 月 9 日	2840	798	2.558897
5 月 10 日	2791	878	2.178815
5 月 11 日	2778	845	2.287574
5 月 12 日	2758	651	3.236559
5 月 13 日	2747	974	1.820329

5 月 14 日	2884	906	2.183223
5 月 15 日	2502	436	4.738532
5 月 16 日	2580	564	3.574468
5 月 17 日	2595	616	3.212662
5 月 18 日	2474	492	4.028455
5 月 19 日	2454	439	4.589977
5 月 20 日	2482	368	5.744565
5 月 21 日	2510	321	6.819315
5 月 22 日	2400	264	8.090909
5 月 23 日	2200	317	5.940063

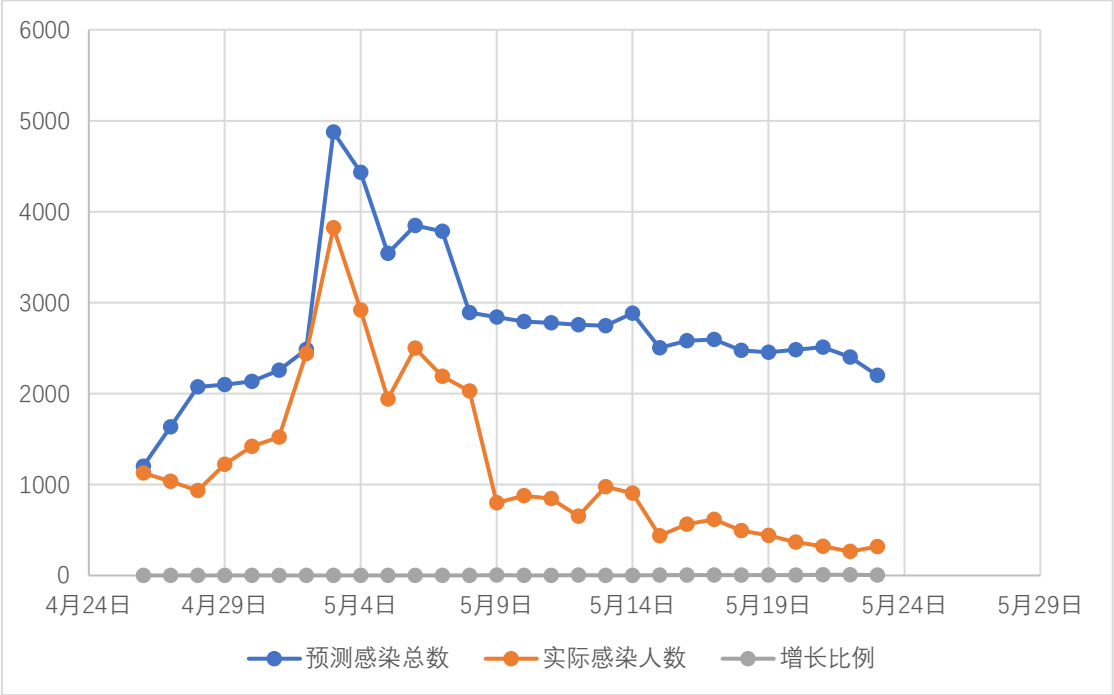


图 5.1.17 预测感染人数和实际感染人数折线图

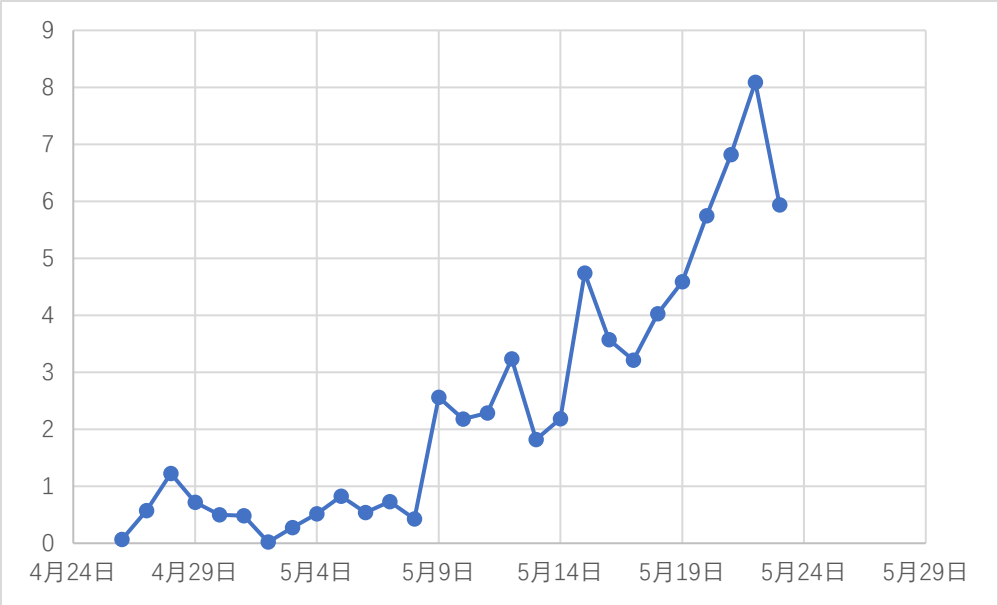


图 5.1.18 预测人数和实际感染人数相差比例

综合 ARIMA 模型和 SEIR 模型进行预测若长春市未实行发放蔬菜包，疫情期间病毒感染人数的变化。由图可知，若未实行发放蔬菜包，预测感染人数比实际感染人数有较大幅度的增加，平均增长率为 24.25%。

5.2 问题二模型建立与求解

5.2.1 问题分析

问题二解决生活物资投放点数量的问题，包括两个目标：首先对不同地区投放点数量合理性进行分析，即投放点数量能否保障隔离区人口的物资保障。为了服务小区居民和医护人员，防疫物资需要很快分发给居民，因此对于不同地区投放点数量需要进行适当优化。第二个目标，考虑到新冠疫情公共卫生事件的突发性，需要确定最优选址数量和规模大小。此外，在选择生活物资投放点时，应该预留具备一定规模和管理能力的现有场所作为储防疫物资和分拣场所的备选点，这些场所在疫情发生时能够迅速被激活，实现对周边患者的快速响应。

针对目标一，需要基于问题一得到长春市不同区域发放蔬菜包前后感染人数，选取感染人数下降率最高的四个地区作为训练集，验证其合理性。根据附加三、附件四，选取小区数量 NSD_i 、路网面积 RND_i 、隔离人数 NIP_i 、蔬菜量包接收量 VPA_i 四个变量，构建最优小区投放点数量模型，并对其余五个地区投放数量进行优化。

针对目标二，需要构建多中心选址模型。首先以最小化生活物资投放点中心到区域内所有小区的距离和最小化投放点数量为目标，构建 CFLP 模型，利用 LU 模型进行优化，对小区进行系统聚类，最后找到最合适投放点中心位置的经纬度坐标和半径。

整体流程图如图 5.2.1 所示。

图 5.2.1 问题二总体分析流程图

5.2.2 合理性分析

基于问题一，本文可以得出蔬菜包发放前后，长春市九个区域感染人数的变化情况，引入平均新增感染率变量 I_i，表达式如下：

$$I_i = \frac{N_i}{A_i} \tag{5.2.1}$$

其中，I_i 为第 i 个区域平均新增感染率，N_i 为第 i 个区域每日新增人数，A_i 为第 i 个区域总感染的人数。九个地区平均新增感染率如下表所示：

表 5.2.1 九个区域平均新增感染率表

区域名称	平均新增感染率
朝阳区	0.075
南关区	0.032
宽城区	0.057
绿园区	0.069
二道区	0.093
长春新区（高新）	0.041
经开区	0.071
净月区	0.025
汽开区	0.084

得到平均新增感染率最低的四个地区为：南关区、长春新区（高新）、净月区、宽城区。本文假设这四个地区投放点数量合理，并作为训练集，构建最优小区投放点数量模型。

5.2.3 最优小区投放点数量模型

5.2.3.1 评价因素的初步划分与解释

首先对影响储备物资的分拣中心数量的四个指标给出具体的定义。

（1）小区数量 NSD_i（Number of subdivisions）

小区数量 NSD_i 表示第 i 个区域内所有小区的总数。小区数量越多，居住人口越多，对人力资源的需求量越大，分拣中心数量越多。

（2）路网密度 RND_i（Road Network Density）

路网密度 RND_i 表示第 i 个区域内所有路线长度 RL_i 与该区域内的总面积 S_i 的比值。路网密度越大，造成交通拥挤的可能性越大，运输给各个投放点的运送成本增加，所以应该设置更多的生活物资投放点。

$$RND_i = \frac{RL_i}{S_i} \tag{5.2.2}$$

（3）隔离人数 NIP_i（Number of Isolated People）

隔离人数 NIP_i 表示第 i 个区域内所有被隔离的人数（见附件二）。隔离人数越多，需要的物资量越多。由于疫情尽可能的减少人员流动与接触。

（4）蔬菜包接受量 VPA_i（Vegetable Package Acceptance）

VPA_i 表示第 i 个区域的蔬菜包投放总量。本文选择附件四 3 月 29 日蔬菜包对接表中的数据作为每个区域的蔬菜包投放总量。

5.2.3.2 最优小区投放点数量模型的构建

本文选择了小区数量、路网密度、隔离人数以及蔬菜包接受量作为自变量，构建了最优小区投放点数量模型，计算公式如下：

$$n_i = \alpha_1 NSD_i + \alpha_2 RND_i + \alpha_3 NIP_i + \alpha_4 VPA_i + \varepsilon_i \tag{5.2.3}$$

其中，ε_i 为第 i 个区域投放点的扰动因子。

5.2.3.3 最优小区投放点数量模型求解

本文选取南关区、长春新区（高新）、净月区、宽城区四个区域为训练集，带入 5-1 式，得到最优小区投放点数量模型参数的最优解。通过数据对比可知，朝阳区、绿园区、二道区、经开区、汽开区五个区域投放点数量不合理。因此，对其进行适当优化，得到优化后的投放点数量如表 5.2.2 所示。

表 5.2.2 九个区域优化前后投放数量表

区域名称	优化前投放点数量	优化后投放点数量
朝阳区	94	285

南关区	261	260
宽城区	181	221
绿园区	470	251
二道区	9	255
长春新区（高新）	215	240
经开区	37	127
净月区	279	198
汽开区	10	139

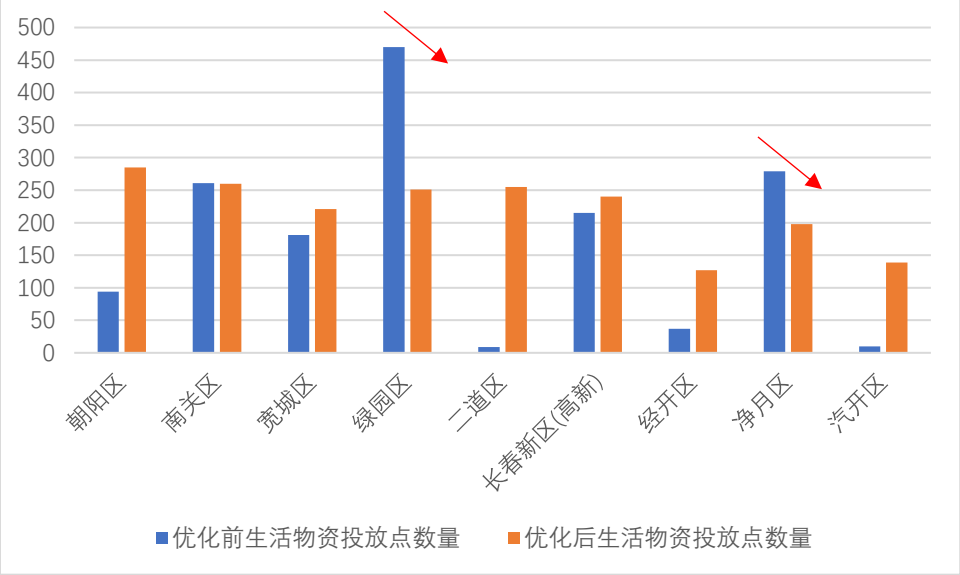


图 5.2.2 九个区域优化前后投放数量对比图

由图 5.2.1 可知，生活物资投放点投放需求在城市中心区域分布紧密、边缘区域分布分散。因此，对于面积大、人口密度低、资源需求点多的隔离区域，如绿园区和净月区，可以考虑减少生活物质投放点的数量。二道口和汽开区的小区分布较为集中，隔离人数较多，需要增加投放点数量。

5.2.4 多分拣中心选址模型

5.2.4.1 CFLP（Capacitated Facilities Location Problem）模型

本文重点是多个分拣中心选址的问题。多个分拣中心选址是在一些已知的投放地点中选出距离相近的一定数目的地点来设置配送中心形成生活物资配送网络系统(如图 5.2.2 所示），使形成的网络总费用最小，即求解使得到各个投放点的距离和分拣中心数量最小的最优解[6,8]。

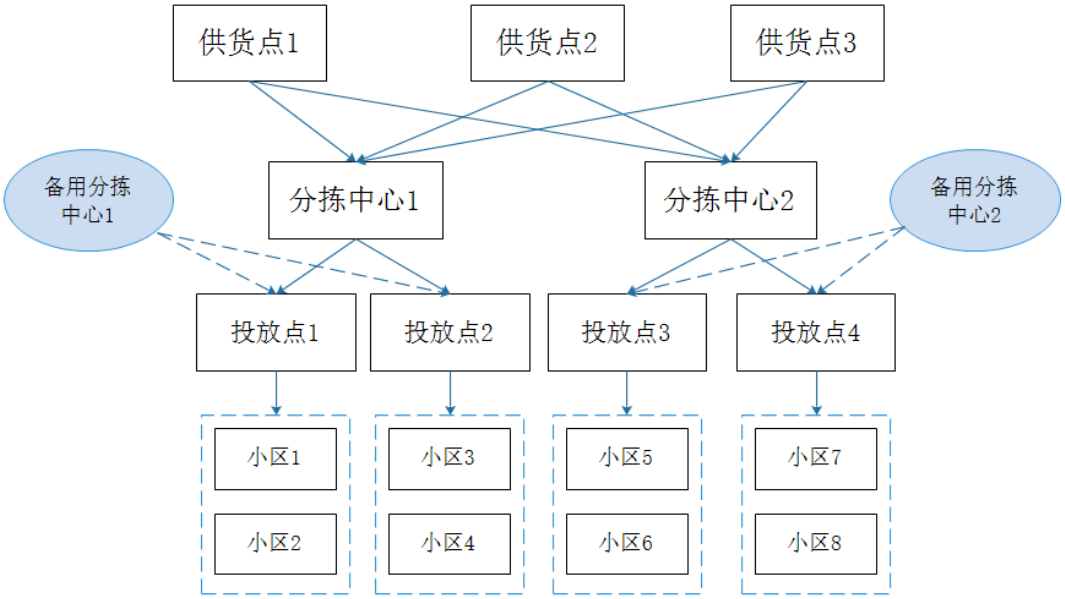


图 5.2.3 物资配送网络图

在一个区域内建立多个分拣中心时，需要更加准确的计算每个分拣中心的位置，并满足合理的分配需要，才能达到降低人力资源以及感染风险率。本文主要采用 CFLP 模型和聚类模型。

CFLP 模型也被称为反町氏法。它用线性规划运输法，确定各分拣中心的占有率，求出它们的重心，然后再采用混合整数规划法的“筹划型”确定分拣中心的位置[4]。

通过问题分析，我们建立以下的数学模型。

1）目标函数：

最小化分拣中心与各小区距离以及投放点数量。

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m} dis(ps_i, pe_j) + L(n) \tag{5.2.4}$$

$$L(n) = k \times n \tag{5.2.5}$$

2）约束条件：

a）分拣中心到小区的距离不应大于其最大投放半径

$$0 \le d_{ij} \le d_0 \pm \xi \tag{5.2.6}$$

b）分拣投放点数量不超过原定的最大限额

$$1 \le n \le n_0 \pm \delta \tag{5.2.7}$$

3）符号说明：

n：最优分拣中心的数量

m：小区总数

n_0：可建分拣中心的最大个数

d_0：最大投放半径

d_{ij}：第 i 个分拣中心到第 j 个小区的距离

ps_i：第 i 个分拣中心的位置

pe_j：第 j 个小区的位置

$dis(a,b):a,b$ 两点之间的距离

k：惩罚项系数，消除量纲影响

ξ：最大投放半径的扰动因子

δ：最大限额的扰动因子

5.2.4.2 小区系统聚类

由于九个区域小区的总数共 1409 个，计算量复杂，数据处理不方便。因此，本文对所有小区进行系统聚类，将聚类后的结果看做新的小区重心。

系统聚类又称为谱系聚类，通过计算两类数据点间的距离，对最为接近的两类数据点进行组合，并反复迭代这一过程，直到将所有数据点合成一类，并生成聚类谱系图。其算法流程如下：

Step 1：将每个对象看作一类，计算两两之间的最小距离；

Step 2：将距离最小的两个类合并成一个新类；

Step 3：重新计算新类与所有类之间的距离；

Step 4：重复二三两步，直到所有类最后合并成一类；

Step 5：得到总体分类。

如何高效的选择聚类个数，对于提高模型的准确度十分重要。本文采用肘部法则–聚类数量选择法。

肘部法则的计算原理是成本函数，即类别畸变程度之和，每个类的畸变程度等于每个变量点到其类别中心的位置距离平方和。

假设将 n 个样本划分到 K 个类中，用 C_k 表示第 k 个类（$k=1,2,\cdots,K$），且该类重心的位置记为 u_k，那么第 k 个类的畸变程度为：

$$\sum_{i\in C_k}\left|x_i-u_k\right|^2 \tag{5.2.8}$$

定义所有类的总畸变程度为：

$$J=\sum_{k=1}^{K}\sum_{i\in C_k}\left|x_i-u_k\right|^2 \tag{5.2.9}$$

其中 J 又称为聚合系数。

类内部的成员彼此越紧凑则类的畸变程度越小，越分散越大。在选择类别数量上，肘部法则会把不同值的成本函数值画出来。随着值的增大，每个类包含的样本数会减少，于是样本离其重心会更近平均畸变程度会减小。随着值继续增大，平均畸变程度的改善效果会不断减低。值增大过程中，畸变程度的改善效果下降幅度最大的位置对应的值就是肘部。在计算过程中，为了简化计算，根据肘部法则，确定聚类个数，将小区密集度高的多个小

区聚类得出新的小区中心，再计算新的小区中心与分拣中心的距离[6]。

5.2.4.3 多分拣中心选址模型求解

考虑到一个分拣中心它所管辖的投放点数量有限。因此，设置最大管辖投点数量 M_0。本文用 matlab 求解得到，最优选址数量和规模如下表 5.2.3 所示，其中位置和选址半径均为虚拟坐标系。

表 5.2.3 最优选址数量、规模（位置和选址半径均为虚拟坐标系）

位置	所属区域	选址半径	管辖范围小区个数	管辖范围内人口数
（35.94,10.84）	长春新区	7.78	28	178451
（43.47,27.82）	长春新区	7.01	29	176214
（36.28,22.43）	长春新区	6.89	28	167841
（44.34,34.98）	朝阳区	5.12	74	164151
（54.41,39.21）	朝阳区	4.97	45	120121
（51.45,30.10）	朝阳区	4.56	28	81451
（50.59,44.53）	朝阳区	4.84	53	143141
（49.29,16.79）	南关区	6.87	44	102474
（61.24,47.70）	南关区	7.21	44	102451
（58.35,27.51）	南关区	6.86	52	120141
（59.88,40.70）	南关区	5.45	62	13412
（58.50,61.44）	宽城区	6.12	66	110415
（50.36,64.42）	宽城区	6.31	45	89451
（57.89,52.92）	宽城区	7.01	57	10104
（45.66,46.70）	绿园区	4.01	77	134511
（46.31,54.66）	绿园区	4.12	45	89451
（37.21,49.37）	绿园区	4.21	49	91021
（29.23,41.85）	绿园区	9.1	13	31420
（69.31,47.07）	二道区	5.01	49	109212
（69.00,57.71）	二道区	5.13	41	94755
（71.38,40.62）	二道区	4.97	39	92421
（66.95,40.83）	二道区	4.89	40	93454
（78.99,42.76）	经开区	6.21	51	84660
（70.48,33.13）	经开区	7.11	63	104580
（68.50,21.58）	净月区	9.81	67	93639
（81.45,9.12）	净月区	7.02	39	54506

（84.81,21.18）	净月区	7.12	40	55911
（42.29,41.74）	汽开区	5.21	48	69856
（25.48,30.60）	汽开区	8.12	39	44508
（33.33,36.74）	汽开区	6.34	54	79098

由于疫情的突发性和不确定性，人们往往无法预测疫情的发生时间与强度，在应急调度初期及应急高峰期由于应急物资生产与储备不足或调度不及时等原因，应急物资短缺现象时有发生，所以需要设置生活物资的备用场所。

本文主要从以下两个方面进行考虑。

首先，对于小区总数和隔离人口数多的区域，传染病感染率很高。传染病应急医疗物资调度过程中，医疗物资短缺和调度延迟会加大疫情扩散的风险。南关区、朝阳区、绿园区等在小区总数排名前三。因此，这些区域应该优先设置备用场所。

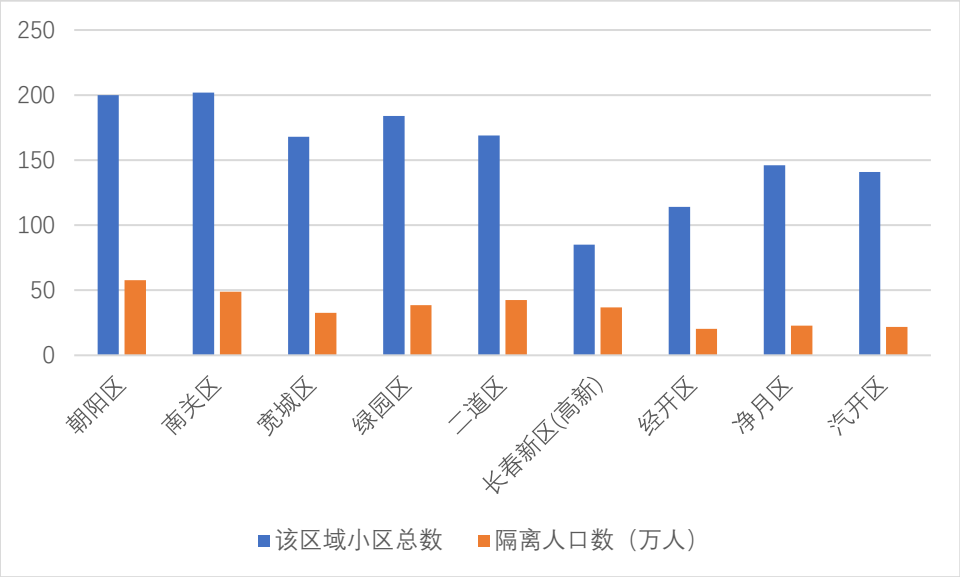


图 5.2.4 九个区域的小区总数与隔离人口数对比图

第二方面，由图 5.2.4 可以看到长春市九个区域小区的分布，其中净月区区域面积较大，小区密度较为分散。南关区、汽开区小区密度分布呈“条带状”，朝阳区、宽城区、二道区小区密度更为集中，呈“团状+点状”。为了降低卡车运输成本以及交通拥堵情况，可以考虑在密度分布为条带状的区域新增备用物资分拣中心。

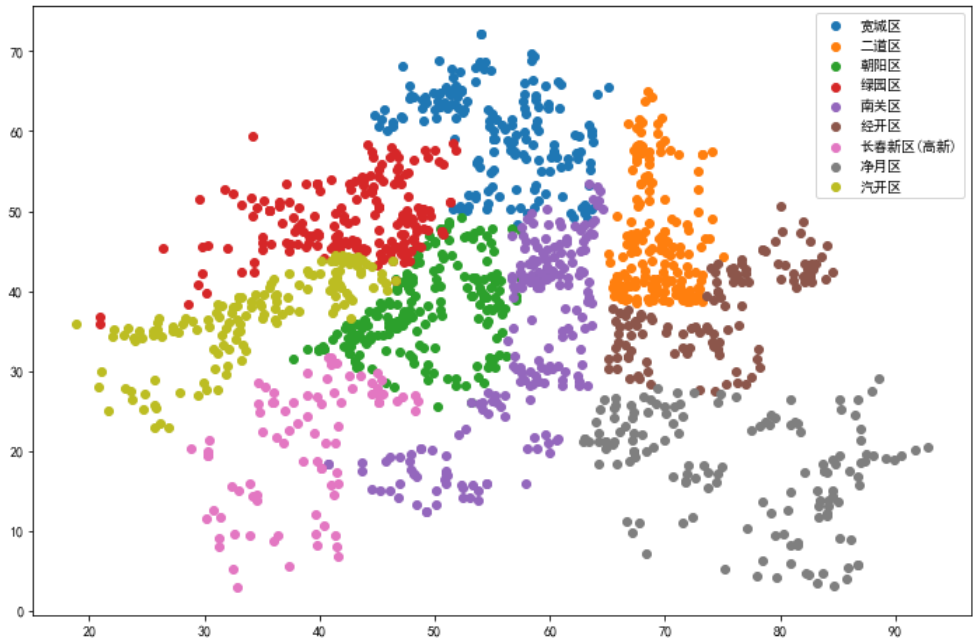


图 5.2.5 长春市小区分布

因此，本文设置了五个备用分拣中心，其位置与所属区域如下表 5.2.4 所示：

表 5.2.4 备用分拣中心的位置与所属区域（位置为虚拟坐标）

备用分拣中心序号	位置	所属区域
1	（40.10，24.12）	长春新区
2	（47.89，37.61）	朝阳区
3	（60.59，40.71）	南关区
4	（42.14，45.32）	绿园区
5	（74.52，40.22）	二道区

最后，分拣中心、备选中心及其选址半径如图 5.2.5 所示，可以看出本文计算出的分拣中心可以实现长春市九个区域内所有小区物资的全覆盖。

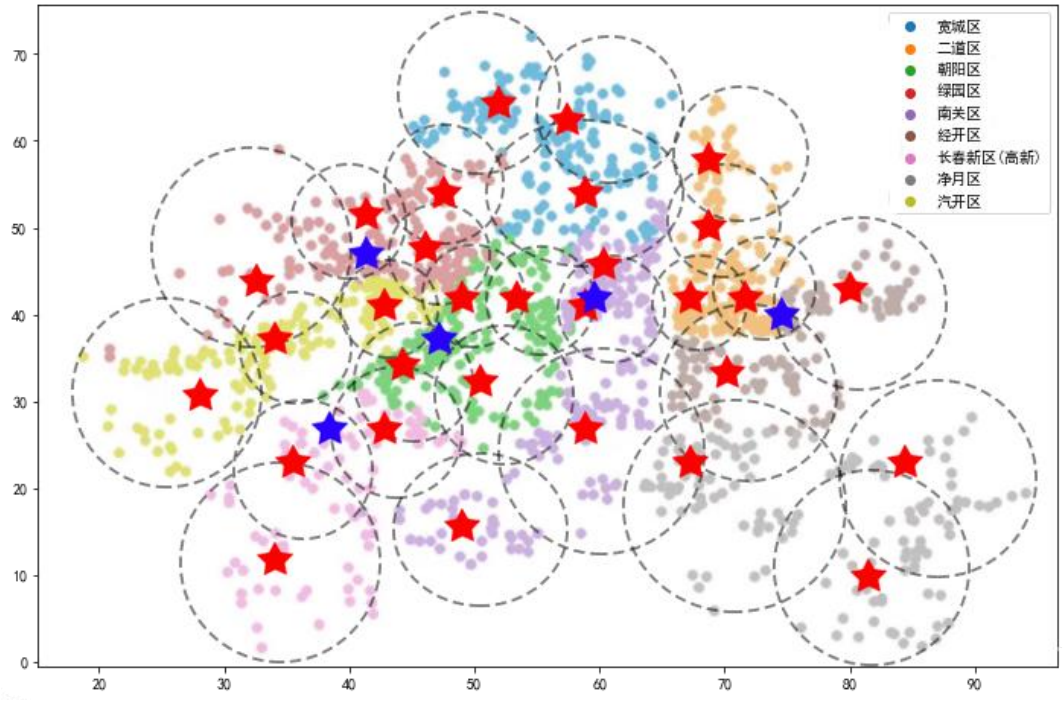


图 5.2.6 分拣中心位置与选址半径示意图

5.3 问题三的模型建立与求解

5.3.1 问题分析

问题三包含三个任务，首先根据蔬菜包的接受发放情况等数据研究蔬菜包的供需规律，再结合小区、人口的信息来评价蔬菜包供应方案，最后对供应方案进行调整。针对任务一，以描述性统计、Pearson 相关系数、需求价格弹性系数、均值多重比较、关联规则挖掘的多种方法考虑多重因素对蔬菜包需求、发放的影响。针对任务二，构建出评价指标体系，主要由潜在扩散风险、接收反应能力、疫情扩散强度三大总指标构成，以这三大指标使用层次分析法、熵权-TOPSIS 方法这两种评价方法统筹做出评价。最后运用 ARIMA 模型对 4 月 10 日-4 月 15 日的需求进行调整，以 9 个区的人口占比为权重，分别计算出各区蔬菜包投放投放数量，再由各小区人口在管辖区内人口占比为权重，最终计算出各小区蔬菜包发放量。问题 3 的具体处理流程如图 5.3.1 所示。

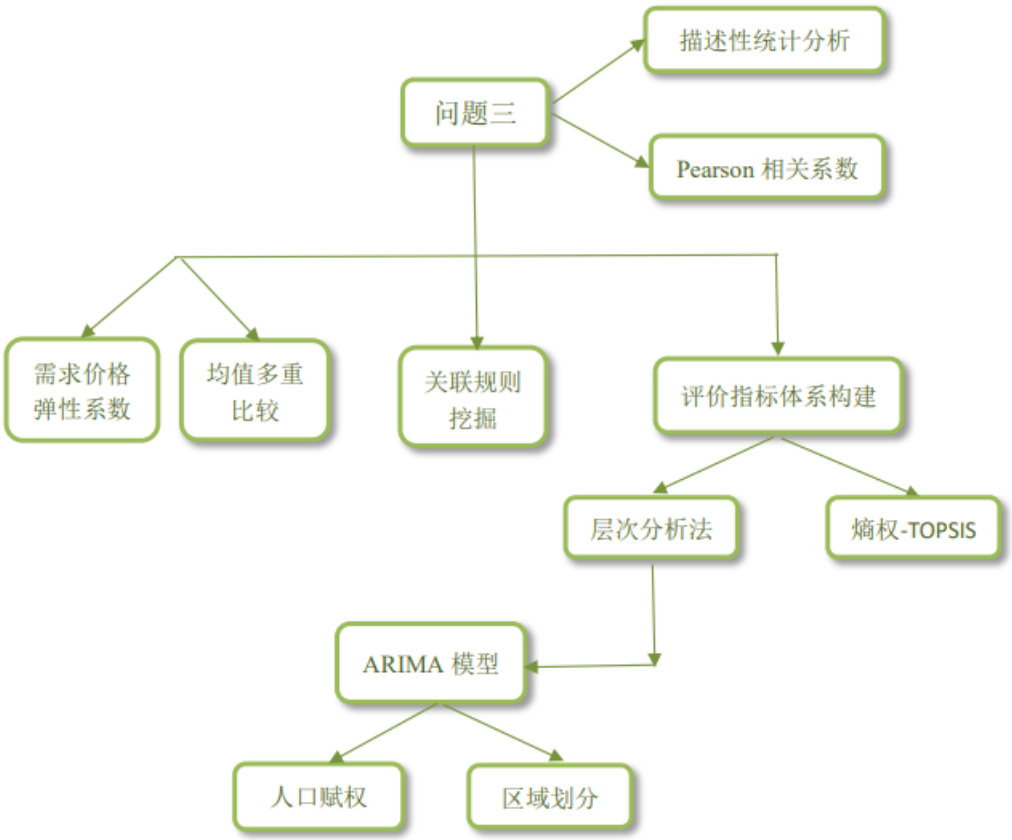


图 5.3.1 问题三总体分析流程图

5.3.2 描述性统计分析

首先我们对各指标的数据分布进行基本的了解，结合附件 5 与查阅相关资料，我们得知 3 月 26 日起至 5 月 1 日长春市开始全面对疫情区居民发放蔬菜包，我们对这段时期内的蔬菜包数据及感染人数数据集先进行了简单的描述统计。

表 5.3.1 描述性统计分析表

单位：吨	样本量	最小值	最大值	平均值	标准偏差	偏度	峰度
市州支援接收	37	0.00	727	145.22	246.50	1.41	0.42
市州支援发放	37	0.00	539.20	135.33	185.45	1.27	0.26
市级直采接收	37	0.00	859.37	185.80	253.00	2.05	3.05
市级直采发放	37	0.00	768.09	174.60	205.66	1.77	2.52
属地自保自采	37	20.00	2009.87	511.22	455.40	1.48	2.43
属地自保发放	37	23.80	1697.12	446.05	410.82	1.48	2.10
共计接收	37	66.00	2820.60	913.48	805.35	1.12	0.28
共计发放	37	64.50	2803.00	910.68	615.88	1.13	0.69
总感染人数/人	37	16	3823	598	787.19	1.63	2.34

表5.3.1的信息反映了各指标的均值、标准差、最大值、最小值、偏度以及峰度情

况。从均值来看，长春市各区通过属地自采获得的蔬菜数量最多，相较来说由市级采摘发放和市州支援的蔬菜数量相对较少，其中市州支援可能考虑到运输成本问题，且大部分时期本地区可以自给。从偏度来看，无论是蔬菜包接收、发放，或是感染人数，都明显呈现出右偏分布，这也说明了蔬菜包的发放与感染人数（隔离人数）变化趋势相似；从峰度来看，所有指标的峰度均大于0，说明蔬菜包的发放与接收在一定时期内存在峰值，感染人数亦存在高峰。

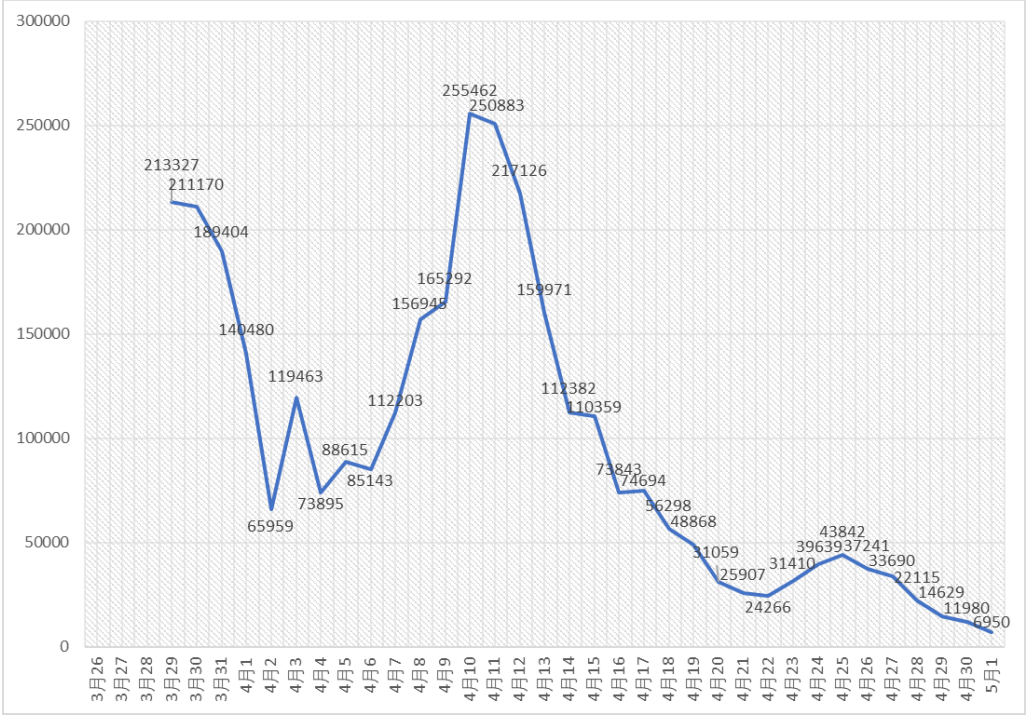


图5.3.2 长春市蔬菜包发放情况

绘制 3 月 26 日至 5 月 1 日的长春市蔬菜包袋数的总体发放情况折线图并对数量进行分析，可发现首次发放蔬菜包为 3 月 29 日，自 29 日至 4 月 2 日发放量逐渐下降。4 月 2 日后整体发放量开始增长，到 10 日为最大值 255462 袋。此后发放量基本呈下降趋势，5 月 1 日为最小值 6950。平均每日发放量为 97191 袋。

5.3.3 相关性定量分析

要分析蔬菜包需求、发放的规律，首先要弄清蔬菜包需求与发放之间存在的内在联系。我们利用两个变量之间的简单相关系数和一个变量与多个变量之间的复相关系数来分析或测定这些变量之间的线性相关程度，并据此进行线性回归分析、预测和控制等。相关系数 r 绝对值愈大(愈接近 1)，表明变量之间的线性相关程度愈高;相关系数绝对值愈小，表明变量之间的线性相关程度愈低。相关系数为零时，表明变量之间不存在线性相关关系。

5.3.3.1 Pearson 相关性分析原理

相关系数(correlation coefficient)是根据数据测定两个变量之间的线性关系强度和相关方向的统计量。若相关系数是根据总体全部数据计算的，称为总体相关系数，记为 ρ；

若是根据样本数据计算的，则称为样本相关系数的计算公式，亦称为皮尔逊相关系数，记为r。其具体计算公式为：

$$r=\frac{\operatorname{cov}(X,Y)}{\sigma_x\sigma_y}=\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2(y_i-\bar{y})^2}} \tag{5.3.1}$$

其中，X,Y为两组n维向量，$X=(x_1,\ldots,x_n),Y=(y_1,\ldots,y_n)$，$\bar{x}$和$\bar{y}$分别为样本向量$X$和$Y$的平均值。可以利用代数推演的方法得到如下的简化式：

$$r=\frac{n\sum xy-\sum x\sum y}{\sqrt{n\sum x^2-(\sum x)^2}*\sqrt{n\sum y^2-(\sum y)^2}} \tag{5.3.2}$$

5.3.3.2 利用相关系数判别线性相关

常用标准如下：

当$|r|$越靠近1，两个变量之间的线性关系越强，当$|r|$越靠近0，两个变量之间的线性关系越弱。根据大量统计数据，通常认为：a）$|r|=0$，表示两个变量之间不存在线性相关；b)$0<|r|\le 0.3$，表示两个变量之间为弱线性相关；c) $0.3<|r|\le 0.5$，表示两个变量之间为低度线性相关；d)$0.5<|r|\le 0.8$，表示两个变量之间为显著线性相关；e) $0.8<|r|\le 1$，表示两个变量之间为高度线性相关；f) $|r|=1$，表示两个变量之间为完全的线性相关。

5.3.3.3 相关系数的显著性检验

一般情况下，总体相关系数ρ是未知的，通常是将样本相关系数r作为ρ的近似估计值。但由于r是根据样本数据计算出来的，因此会受到抽样误差的影响。由于样本是随机抽取的，所以r是一个随机变量。当$|r|$越靠近1，则反映X和Y之间的线性相关性越密切；若是$|r|$越靠近0，则线性相关性程度越小。但在通常情况下，资料都是通过样本得到的，不同的样本其所得结果也因之而异。因此我们采用如下的方法来考察样本相关系数的可靠性。

Step1：提出假设：

$$H_0:\rho=0;H_1:\rho\neq 0; \tag{5.3.3}$$

Step2：预先设定的检验水准为0.05；当检验假设为真，但被错误地拒绝的概率，记作α，通常取$\alpha=0.05$或$\alpha=0.01$;

Step3：计算统计量：

$$t=|r|\sqrt{\frac{n-2}{1-r^2}}\sim t(n-2) \tag{5.3.4}$$

Step4：做出统计推断：根据给定的显著性水平α（本文中我们取$\alpha=0.05$）和自由度

$n-2$。查t分布表，得出$t_{\alpha/2}(n-2)$的临界值。若$t>t_{\alpha/2}$，则拒绝原假设H_0，表明两个变量之间存在显著地线性相关。或根据统计量的大小及其分布确定检验假设成立的可能性P的大小并判断结果。若$P>\alpha$，结论为按α所取水准不显著，不拒绝H_0，即认为差别很可能是由于抽样误差造成的，在统计上不成立；如果$P\le\alpha$，结论为按所取α水准显著，拒绝H_0，接受H_1，则认为此差别不大可能仅由抽样误差所致，很可能是实验因素不同造成的，故在统计上成立。关于P值的具体解释如下。

表 5.3.2 P值的解释

P值	碰巧的概率	对无效假设	统计意义
$P>0.05$	碰巧出现的可能性大于5%	不能否定无效假设	两组差别无显著意义
$P<0.05$	碰巧出现的可能性小于5%	可以否定无效假设	两组差别有显著意义
$P<0.01$	碰巧出现的可能性小于1%	可以否定无效假设	两组差别有非常显著意义

5.3.4 附件 5 数据分析

通过 EViews 软件分析附件 5 中的数据，以及附件 4 中蔬菜价格波动数据，将各种蔬菜价格取平均值，得到“蔬菜价格”指标；我们得到 3 个蔬菜发放指标、总感染人数指标以及蔬菜价格的相关性与检验假设成立的可能性大小P值如下表 5.3.3。

表 5.3.3 相关系数矩阵及检验

Correlation	市州支援发放	市级直采发放	属地自保发放	蔬菜价格	总感染人数
市州支援发放	1.000				
	-				
市级直采发放	0.317*	1.000			
	0.010	-			
属地自保发放	0.404	0.653	1.000		
	0.013	0.000	-		
蔬菜价格	0.442	0.601	0.795	1.000	
	0.006	0.000	0.000	-	
总感染人数	0.400	0.451	0.683	0.759	1.000
	0.011	0.005	0.000	0.000	-

根据表 5.3.3，我们有如下表 5.3.4 的结果及分析。

表 5.3.4 相关系数矩阵及P值与检验水准α的关系

Correlation	市州支援发放	市级直采发放	属地自保发放	蔬菜价格	总感染人数
市州支援发放	1.000				
	-				

市级直采发放	低度线性相关	1.000			
	$P \le \alpha$	-			
属地自保发放	低度线性相关	显著线性相关	1.000		
	$P \le \alpha$	$P \le \alpha$	-		
蔬菜价格	显著线性相关	显著线性相关	显著线性相关	1.000	
	$P \le \alpha$	$P \le \alpha$	$P \le \alpha$	-	
总感染人数	低度线性相关	显著线性相关	显著线性相关	显著线性相关	1.000
	$P \le \alpha$	$P \le \alpha$	$P \le \alpha$	$P \le \alpha$	-

我们有如下结论：

a) 市级直采、市州支援与属地自保之间发放蔬菜量存在低度的线性相关；

b) 总感染人数与 3 种来源的蔬菜发放之间的相关性很强(高度线性相关、显著线性相关)，且强相关的显著性很高($P \le \alpha = 0.05$)。

c) 特别地，我们指出蔬菜价格与总感染人数、3 种来源的蔬菜发放都有显著相关关系，说明疫情感染人数会对蔬菜价格产生影响，感染人数越多，蔬菜价格越高，这可能会进一步影响各区居民的蔬菜需求量。

5.3.5 蔬菜包需求量影响因素分析

5.3.5.1 疫情发展与蔬菜包需求间的联系

通过附件 5 中整理出的数据，将 3 月 4 日-5 月 23 日长春市合计接收蔬菜包总量（单位：吨）作为蔬菜包需求量，与附件 1 中总感染人数(新增本土感染者与无症状感染者之和）在 3 月 26 日到 5 月 1 日的发展情况如下图所示。

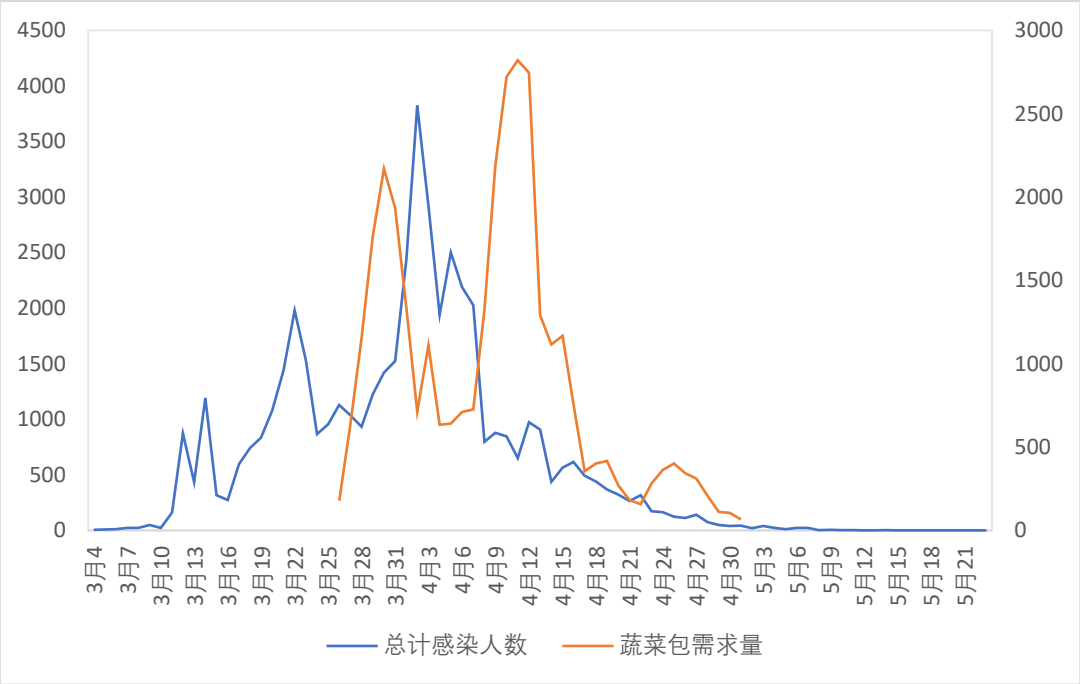


图 5.3.3 由蔬菜包需求与全市感染人数

由上图 5.3.2 可见，

a) 从左向右看，感染人数在 3 月 22 日左右达到了一个小的峰值，蔬菜包由 3 月 26 日开始发放，其需求量在 8 天后（4 月 29 日）也达到了小的峰值，而在疫情峰值过后，随着感染人数下降蔬菜包的需求量也随之减少。这种情况容易理解：疫情的一小波疫情使得长春部分地区需要居家隔离，政府开始采用发放蔬菜包的形式给予居民生活物资上的支持；

b) 由 3 月 28 日开始，感染人数突然呈直线式上升，在 4 月 3 日达到最高峰，而蔬菜包的需求量又迅速增加，在 4 月 11 左右达到最高峰；之后随着疫情感染人数的下降，蔬菜包的需求量也相应地逐渐下降；

c) 通过以上的分析，我们认为蔬菜包的需求量会随着疫情感染人数变动而变动，感染人数增多蔬菜包的需求量就会增大，反应期约为 5~8 日；如要优化改良一段时期内的蔬菜包供给，可以依据感染人数作为依据，运用 ARIMA 模型来预测蔬菜包需求数量，进而准确地得到蔬菜包供给量。

5.3.5.2 蔬菜价格与蔬菜包需求间的联系

a) 蔬菜价格变动

由附件 4 的数据包可以得到 3 月 4 日-5 月 23 日以来每日的蔬菜价格如下图 5.3.3 所示。可以看到，部分蔬菜如“土豆”价格相比其他蔬菜价格没有明显变化，但总体也有缓慢增长的趋势，部分蔬菜价格波动较大，如“青椒”，但大部分蔬菜价格总体看不出有何明显趋势。因此，我们进一步寻找蔬菜价格与蔬菜包需求的关系。

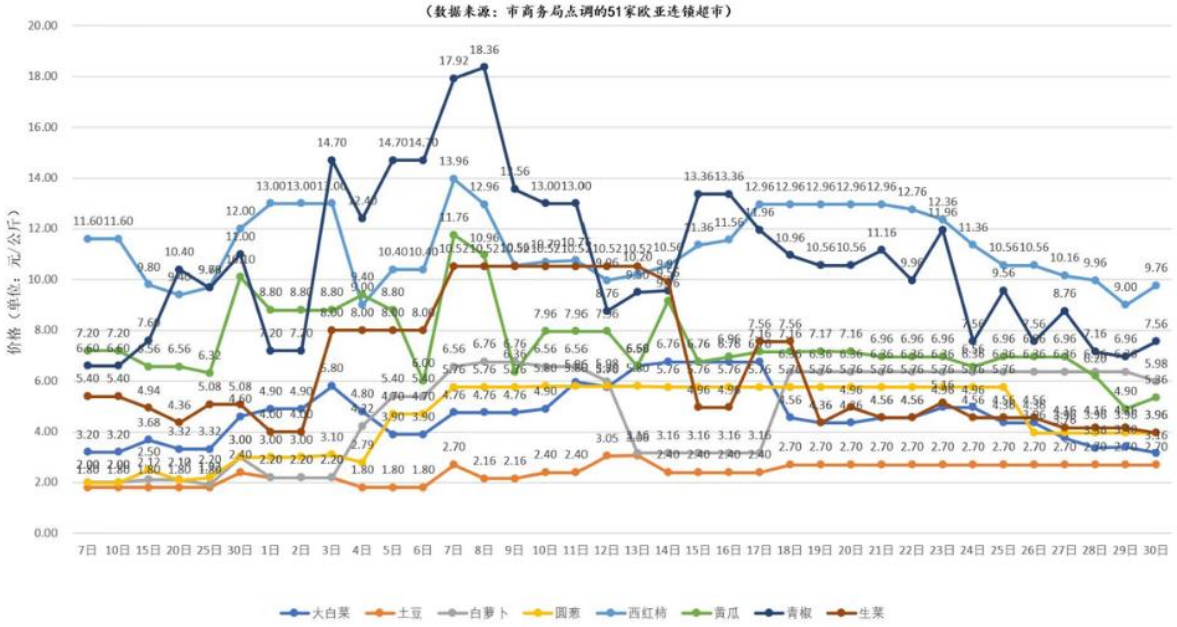


图 5.3.4 点调重点企业蔬菜价格变化图表

b) 蔬菜需求价格弹性系数

我们将 8 种蔬菜每日的价格分别进行算术平均，得到的平均值用 P 来表示；从而将每日蔬菜价格用 P 来表示，用合计接收蔬菜包总量（单位：吨）作为蔬菜包需求量，用 Q 表示，接下来求蔬菜价格的需求弹性系数。

微观经济学中，需求的价格弹性用来表示在一定时期内一种商品需求量的相对变动对于该商品的本身价格相对变动的反应程度。需求的价格弹性简称为需求弹性，需求弹性系数的数值用来表示这种反应程度的大小，即：需求弹性系数=需求量的变动率/价格的变动率。

假定需求函数为 $Q=f(P)$（Q 为商品的需求量，P 为商品的价格；Q 是 P 的函数），ΔQ 和 ΔP 分别表示需求和价格的变动量，E_d 表示需求弹性系数，则：

$$E_d=\frac{\frac{\Delta Q}{Q}}{\frac{\Delta P}{P}}=\frac{\Delta Q}{\Delta P}\bullet\frac{P}{Q} \tag{5.3.5}$$

计算时 E_d 取正数，因此 $\frac{\Delta Q}{\Delta P}$ 取其绝对值 $\left|\frac{\Delta Q}{\Delta P}\right|$。设 TR 代表总收益，则

$$TR=P\bullet Q \tag{5.3.6}$$

当某种商品缺乏弹性，即 $E_d<1$ 时，有 $\frac{\Delta Q}{\Delta P}\bullet\frac{P}{Q}<1$，即 $P\bullet\Delta Q<\Delta P\bullet Q$,此时商品的价格由 P_1 上升为 $P_2=P_1+\Delta P$，需求量相应地由 Q_1 减少为 $Q_2=Q_1-\Delta Q$，则收益的增量表现为：

$$\begin{aligned}\Delta TR &= TR_2 - TR_1 \\ &= P_2 \bullet Q_2 - P_1 \bullet Q_1 \\ &= (P_1 + \Delta P) \bullet (Q_1 - \Delta Q) - P_1 \bullet Q_1 \\ &= \Delta P \bullet Q_1 - \Delta Q \bullet P_1 - \Delta P \bullet \Delta Q > 0\end{aligned} \tag{5.3.7}$$

即$TR_2 > TR_1$，总收益增加。因此可得出结论：对于$E_d < 1$的缺乏弹性的商品而言，提高价格会使商品的总收益增加。

通过计算我们得到每日的需求价格弹性系数如下：

表 5.3.5 蔬菜需求价格弹性系数表

日期	E_d
3月25日	0.561
3月26日	0.691
3月27日	0.869
…	…
4月13日	0.847
…	…
5月21日	0.319
5月22日	0.112
5月23日	0.628

将每日求得的E_d进行算数平均，作为我们需要的需求价格弹数，最终得到：

$$E_d = 0.745 \tag{5.3.8}$$

由于蔬菜是缺乏弹性的商品，因而随着价格的波动蔬菜包的需求量变动也不会过大，但蔬菜价格对蔬菜包的需求量仍存在一定的影响，蔬菜价格越高蔬菜包需求量会相应减少，可以通过E_d对蔬菜包的发放进行数量调整。

5.3.5.3 区域与蔬菜包需求之间的联系

前两部分分别考虑了感染人数、蔬菜价格对蔬菜包需求的影响，下面我们对不同区的蔬菜包需求量进行方差分析终的均值多重比较，看不同区的蔬菜包需求是否有显著差异，篇幅过长这里只选出 4 个代表性的区的比较结果。

表 5.3.6 不同区蔬菜包需求均值多重比较结果

区域(I)	平均值	区域(J)	平均差($I-J$)	标准差	显著性	95%置信区间	
						下限	上限
朝阳区	88.894	经开区	2.268	23.311	0.163	-12.352	27.610
		宽城区	-12.707	32.168	0.000	-43.658	-18.356

		汽开区	59.974	34.765	0.000	21.548	63.184
经开区	86.626	朝阳区	-2.268	23.311	0.163	-27.610	12.352
		宽城区	-14.975	29.325	0.000	-52.819	-29.614
		汽开区	57.706	28.546	0.000	20.761	40.765
宽城区	101.601	朝阳区	12.707	32.168	0.000	18.356	43.658
		经开区	14.975	29.325	0.000	29.614	52.819
		汽开区	72.681	33.731	0.000	18.352	73.879
汽开区	28.920	朝阳区	-59.974	34.765	0.000	-63.184	-21.548
		经开区	-57.706	28.546	0.000	-40.765	-20.761
		宽城区	-72.681	33.731	0.000	-73.879	-18.352

分析表 5.3.6 可知，

a) 汽开区蔬菜包需求量最少，均值仅有 28.920 吨，宽城区蔬菜包需求量最大，均值达 101.601 吨；

b) 通过观察显著性，可以得知除朝阳区与经开区蔬菜包需求量差异不显著外，其他区之间蔬菜包需求量的差异均在 5%水平下显著；

由此可见，不同区的蔬菜包需求量也存在不同，这与各区的感染人数不同有关。

5.3.6 关联规则挖掘

关联规则挖掘以关联规则的分组与比较为基础，可从大量数据中发现项集相互之间的关系，是典型的数据挖掘方法之一。挖掘出的关联规则为先行条件 A 导出后继结果 B，即“$A \to B$”。其统计指标有两个，分别为支持度（support）、置信度（confidence），表达式如下：

$$support(A \to B) = P(A \cup B) \tag{5.3.9}$$

$$confidence(A \to B) = P(B \mid A) = \frac{P(A \cap B)}{P(A)} \tag{5.3.10}$$

$support(A \to B)$ 表示整个数据集中事件 A、事件 B 同时发生的概率，

$confidence(A \to B)$ 表示时间 A 发生后引起事件 B 发生的可能性。

其还有多种评价标准，如提升度（lift），确信度（conviction），卡方系数，Kulc，cosine 距离，Leverage，不平衡因子（IR）等。

其中提升度的表达式为：

$$lift(A \rightarrow B) = \frac{P(B \mid A)}{P(B)} = \frac{P(A \cup B)}{P(A)P(B)} \qquad (5.3.11)$$

$lift(A \rightarrow B)$ 表示事件 A 发生后事件 B 发生的概率是否比事件 B 单独发生的概率高。

同时满足最小支撑度(sup_min）和最小置信度(con_min)的规则被称为强关联规则。

5.3.6.1 以 Apriori 改进的基于数据立方体的多维挖掘算法(MD-apriori)

Apriori 算法是关联规则中最经典最常用的挖掘频繁项集的算法，它采用逐层搜索的迭代方法，核心思想是通过连接产生候选项并得到其支持度，再通过剪枝生成频繁项集。其主要分为连接步和剪枝步两个步骤。连接步首先对 1 项候选集 C_1，筛除小于给定最小支持度阈值的项集得到 1 项频繁集，由此不断循环迭代生成 2 项、3 项直至所需的 K 项频繁集。剪枝步在连接步其后，将不满足频繁项集的所有非空子集也是频繁项集这一性质的项集剔除。实例演算如下：

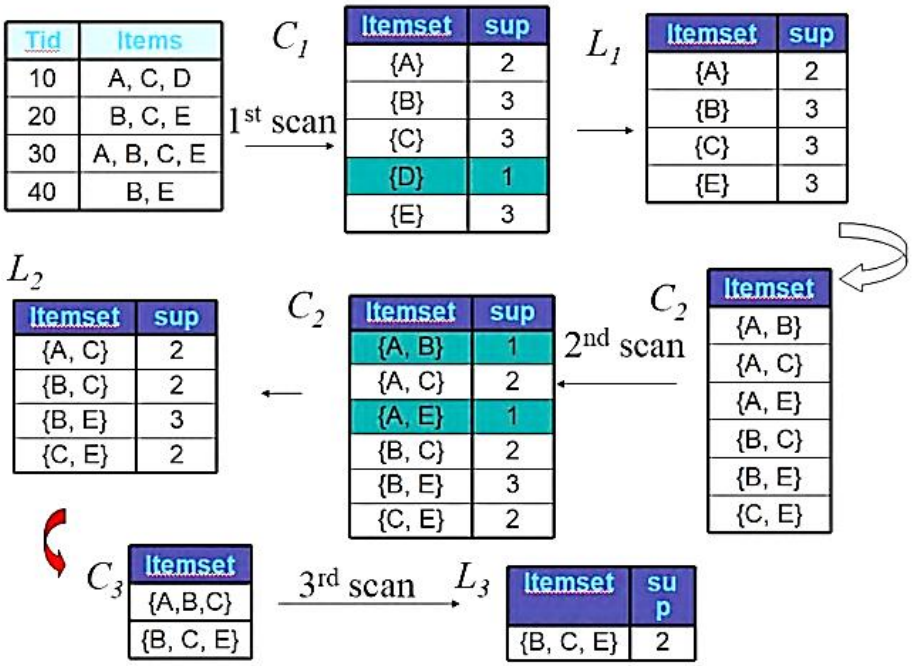


图 5.3.5 Apriori 算法流程

使用 Apriori 算法进行关联规则生成的流程如下图所示。

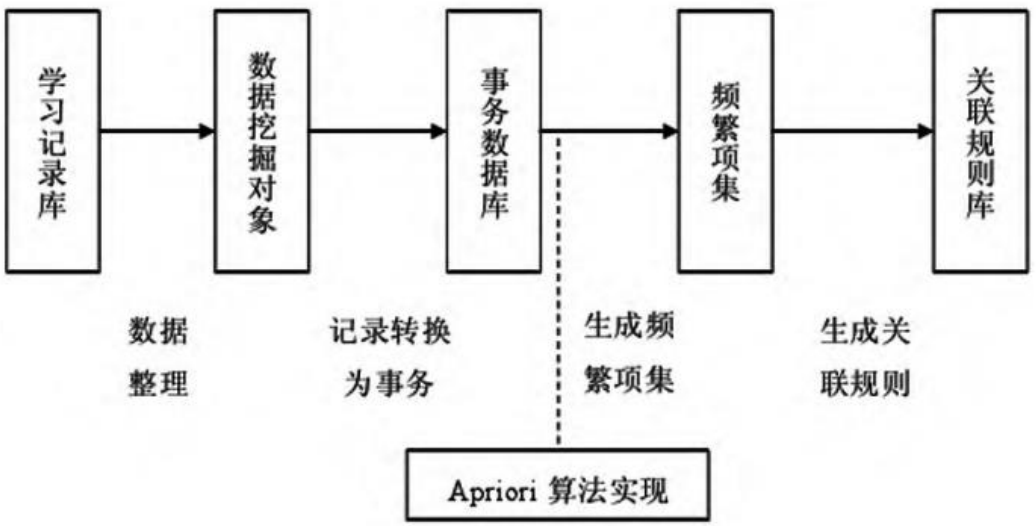


图 5.3.6 关联规则生成流程

以 Apriori 改进的基于数据立方体的多维挖掘算法(MD-apriori)[1]，能够减少对无关频集的生成，降低复杂度，节省时间与空间资源。该算法以 Apriori 算法为基础，对候选集检查其在立方体中相对应的方格，将其与最小支持度比较来进行筛选。

以 MD-apriori 算法进行关联规则挖掘可归结为以下步骤：

Step1：数据离散化。对蔬菜包需求袋数、吨数、蔬菜包投放袋数、吨数、感染人数、隔离人数等采取等宽离散化处理。对关联规则的生成量（模型复杂度）及各数据的范围规模、可区分性进行综合考虑。以净月区为例，将蔬菜包袋数以 2000 为间隔、吨数以 20 为间隔、感染人数以 10 为间隔、隔离人数以 10 为间隔、投放点以 50 为间隔进行等宽离散化。

Step2：以联机分析处理（OLAP）技术进一步组织和处理数据，将每日市场支援、市场直采、属地自保及总和的蔬菜包需求袋数、吨数、投放袋数、吨数与感染人数组织处理为一组多维数据立方体，将每个区的蔬菜包平均每日需求袋数、吨数、平均每日投放袋数、吨数、隔离人数、投放点数组织处理为另一组多维数据立方体。

Step3：采用 MD-apriori 算法，其算法步骤为：

输入：a.一个 n 维数据立方体 $CB[d_1,\ldots,d_n]$ b.最小支持度：sup_min
输出：n 维间的频繁项目集 L
Step1： $k=1; L=\varpi;$ Step2：对每维生成 1 项候选集 $C_{1,di}=\{di\text{维中所有互不相同的取值}\}$， $C_1=U_{i=1}^{n}C_{1,di}$； Step3：生成 1 项频繁项目集 $L_1=gen-frequent(1,C_1)$； Step4：Repeat

k=k+1;

生成 k 项候选集 $C_k = gen-candidate(k, L_{k-1})$；

生成 k 项频繁集 $L_k = gen- frequent(k, C_k)$；

$L = L \bigcup L_k$ ；

Until $L_k = \varphi$；

其中 $gen- frequent(k, C_k)$ 函数用于从候选集 C_k 中生成频繁项目集 L_k，$gen-candidate(k, L_{k-1})$ 函数用于从频繁项目集 L_{k-1} 中生成 k 项候选集 C_k。

Function $gen- frequent(k, C_k)$

$L_k = \varphi$;

for each candidate $I = \{i_1, i_2 \ldots, i_k\} \in C$ do{

frequency=k 维立方体空间中的方格 $(i_1, \ldots, i_k)$ 中的 count 值

support=frequency/totalcount;

if(support>sup_min)then

$L_k = L_k \bigcup \{I\}$

}

Function $gen-candidate(k, L_{k-1})$

$C_k = \varphi$;

for each item $I_1 \in L_{k-1}$ {

for each item $I_2 \in L_{k-1}$ {

if(I_1 与 I_2 有 k-2 个相同项目，且最后一个项目来自不同维)then{

if c 有非频繁的（k-1）子集，then{

$c = I_1 \Theta I_2$

}

if c 有非频繁的（k-1）子集，then{

删除 c

}

else 将 c 加入到 C_k 中

}

}

}

return C_k

5.3.5.2 改进的模糊关联规则挖掘算法[2]

传统 Apriori 算法只能处理布尔型数据，如蔬菜包发放量需要转换成离散值才能够用

于计算。而模糊关联规则挖掘算法模糊化了多值数据并改进支持度、置信度等概念，更适合用于多值数据信息挖掘，可以更好的提取出连续值中的信息。

传统的模糊关联规则挖掘算法以选定的阈值进行划分，小于阈值的值设为“0”，大于阈值的值设为“1”。这种区间划分方式会使区间边界尖锐，可能导致区间边界临近处信息的丢失。在这里引入模糊集合理论，以隶属函数表征，值域为$[0,1]$，能将属性转化为模糊属性，减弱区间边界的尖锐程度，从而保留边界数据所包含的信息。对事务I_i进行模糊化的过程可表示为：

$$I_{i\_fuz} = F_i(I) \tag{5.3.12}$$

其中$F_i(x)$表示模糊函数，I_{i_fuz}表示模糊化处理后的事务I_i。

经过模糊化后的数据集D_{fuz}即为模糊集。

模糊化后支持度和置信度的概念也进行了相应改变以适应取值范围在值域[0,1]中的模糊数。对数据集D，模糊项集$X=\{x_1,x_2,\ldots,x_p\}$，计算项集X对D的支持度为：

$$Support(X)_D = \frac{x_1^i \times x_2^i \ldots x_p^i}{|D|} \tag{5.3.13}$$

模糊关联规则挖掘算法计算可归结为以下步骤：

*Step1：*整合概念集，分别以每个日期和每个地区作为关联规则挖掘中的事务进行亮相关联规则挖掘，以表 5.3.7，5.3.8 中所示概念作为关联规则挖掘中的项目。

表 5.3.7 按日期进行关联规则挖掘的概念集

标签					
蔬菜包需求袋数	蔬菜需求吨数	蔬菜包投放袋数	蔬菜投放吨数	无症状感染者人数	本土感染者人数

表 5.3.8 按地区进行关联规则挖掘的概念集

标签					
蔬菜包平均每日需求袋数	蔬菜平均每日需求吨数	蔬菜包平均每日投放袋数	蔬菜平均每日投放吨数	地区隔离人数	地区投放点个数

*Step2：*对附录所给数据进行整合，得到数据集，以净月区按日期的数据集其中十条为例，如表 5.3.9 中所示。

表 5.3.9 净月区 4 月 4 日至 4 月 13 日数据集

蔬菜包需求袋数	蔬菜需求吨数	蔬菜包投放袋数	蔬菜投放吨数	无症状感染者人数	本土感染者人数
10744	123	4934	54	51	38

1955	20	7329	87.2	53	21
0	0	5083	51	50	30
0	0	8871	88	77	12
10451	104	10451	104	15	4
8980	89	8980	89	6	3
11594	110	11594	110	21	1
7462	74.62	7462	74.62	24	6
23333	222	19333	186	20	3
23126	185	24522	200	26	7

Step3：构造隶属函数。由于数据集所跨时间较短，选取短时间的小尺度下的参数来构造隶属函数。此处采用偏大梯形隶属函数，其函数表达式为：

$$\mu_A=\begin{cases}0, x\le a\\ \dfrac{x-a}{b-a}, a\le x\le b\\ 1, x>b\end{cases} \tag{5.3.14}$$

a 和 b 的数值根据前述数据统计分析结果来确定，μ_A 为隶属函数值。

Step4：将概念中的各项所含多值型数据转换为模糊值，模糊化得到模糊集。

表 5.3.10 净月区 4 月 4 日至 4 月 13 日模糊集

蔬菜包需求袋数	蔬菜需求吨数	蔬菜包投放袋数	蔬菜投放吨数	无症状感染者人数	本土感染者人数
0.451	0.554	0.197	0.265	0.918	0.988
0.082	0.090	0.293	0.427	0.954	0.546
0	0	0.203	0.250	0.917	0.780
0	0	0.355	0.431	1	0.312
0.439	0.468	0.418	0.509	0.272	0.104
0.377	0.401	0.359	0.436	0.108	0.078
0.487	0.495	0.464	0.539	0.378	0.026
0.313	0.336	0.298	0.366	0.432	0.156
0.979	0.978	0.773	0.911	0.362	0.078
0.971	0.833	0.981	0.980	0.468	0.203

Step5：分析蔬菜包的投放、需求与其他概念间的支持度、置信度数值。以 $X_1, X_2 \ldots, X_6$ 表示如上六个概念，部分实验结果如表 5.3.11 所示。

表 5.3.11 净月区蔬菜包投放袋数与其他概念的支持度、置信度

$X_2 \to X_1(88.769\%, 80.457\%)$
$X_3 \to X_1(66.337\%, 68.397\%)$

$X_4 \rightarrow X_1(62.476\%, 53.457\%)$
$X_5 \rightarrow X_1(36.568\%, 24.464\%)$
$X_6 \rightarrow X_1(33.636\%, 38.477\%)$

可以看出蔬菜包的投放量与蔬菜投放吨数关联性最强，与需求量、需求吨数关联性也较强,与无症状感染人数、本土感染人数也呈一定的关联性。

同理求得按地区进行关联规则挖掘的六个概念间的支持度、置信度数值，以 $S_1, S_2 \ldots, S_6$ 表示蔬菜包平均每日需求袋数、蔬菜平均每日需求吨数、蔬菜包平均每日投放袋数、蔬菜平均每日投放吨数、地区隔离人数、地区投放点个数七个概念，部分实验结果如表 5.3.12 所示。

表 5.3.12 净月区蔬菜包平均每日投放袋数与其他概念的支持度、置信度

$S_2 \rightarrow S_1(83.247\%, 75.394\%)$
$S_3 \rightarrow S_1(73.579\%, 64.378\%)$
$S_4 \rightarrow S_1(70.423\%, 56.436\%)$
$S_5 \rightarrow S_1(25.547\%, 20.458\%)$
$S_6 \rightarrow S_1(44.436\%, 36.422\%)$

可看出各个地区的蔬菜包投放量与隔离人数关联较弱，与投放点个数关联稍强。

5.3.7 蔬菜包供应方案评价

5.3.7.1 蔬菜包需求-供应对比

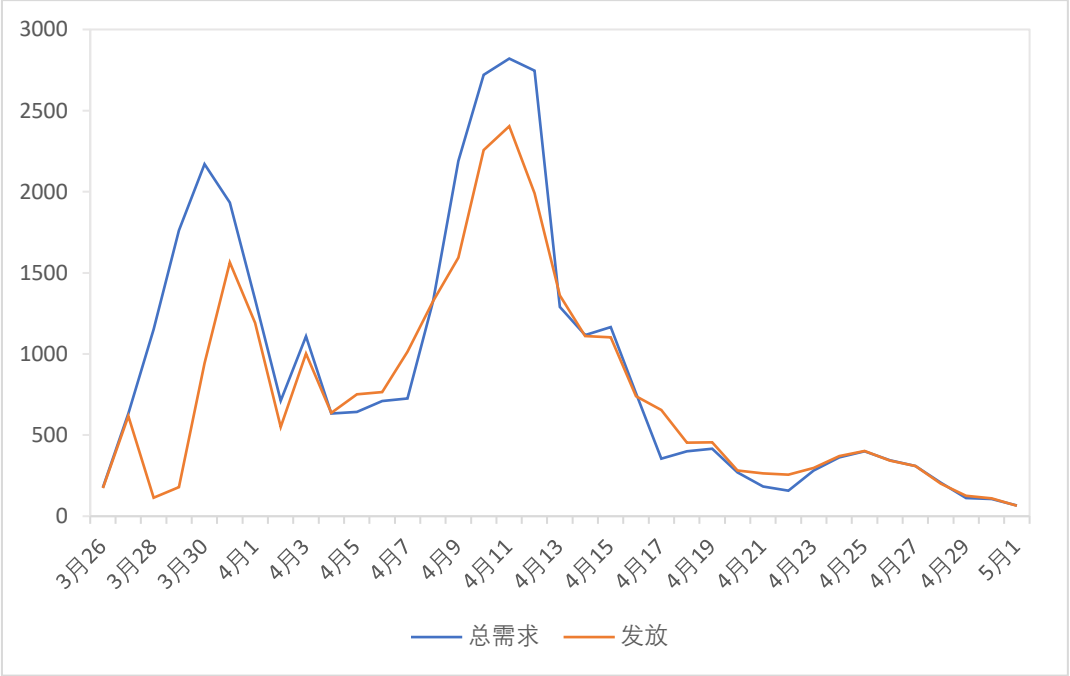


图 5.3.7 蔬菜包需求-供给对比

图 5.3.6 为全市 3 月 26 日—5 月 1 日蔬菜包需求-供给对比折线图，可以看到在 3 月 26 日看蔬菜包供应开始，至 4 月 1 日处于供不应求的状态；到 4 月 9 日左右，疫情感染人数又一轮爆发，蔬菜包的供应又处于供不应求的状态直至 4 月 13 日左右，之后供求趋于稳定。

综上所述，4 月 10 日—4 月 15 日之间蔬菜包的供应存在供不应求的情况，为此，我们结合人数与位置两个要素，对期间蔬菜包的供应进行调整改善。

5.3.7.2 蔬菜包供应量评价指标体系构建

在了解现有文献研究及分析题给数据集的基础上，选取疫情潜在扩散风险、疫情应急反应能力以及疫情扩散强度这三点作为评价指标体系中的一级指标，在一级指标下继续西安则多个指标作为二级指标，共同组成完整的评价指标体系。对于疫情扩散强度这一指标，由于每日的感染者人数不断发生变化，这里选用动态指标来刻画疫情扩散强度。所得到的评价指标体系如下表所示。

表 5.3.13 蔬菜包供应量评价指标体系构建

一级指标	二级指标	指标说明
潜在扩散风险/人群蔬菜需求	人口总量 *NIP*	各个区的隔离人口总数，人口越多则感染风险越大
	人口密度 *R*	人口总数与面积的比值，密度越大感染概率越大
接收反应能力	小区个数 *NSD*	各个区的小区个数，反映地区向下管理的难易程度
	小区密度 *Q*	反映地区内小区接收蔬菜包的难易程度
疫情扩散强度（动态指标）	本土感染者占比 *T*	本土感染者占总人数的比率，反映疫情传染率
	无症状感染者占比 *E*	无症状感染者占总人数的比率，反映疫情的潜在传染率

a) 疫情潜在扩散风险。COVID-19 疫情的扩散可能性与所在区域的人口分布情况密切相关，其中通过隔离人口总量 *NIP* 可以大致获得人群中的易感者人数，而人口密度 *R* 则能够间接反映该区域个人能够接触到的人数和传染概率。同时，人口总量 *NIP* 越多，需要的蔬菜包数量也就越多。因此人口总量 *NIP* 越大、人口密度 *R* 越高，该地区获得蔬菜包供应的优先级就越高。

b) 区域接收反应能力。接收反应能力反映了该地区接收蔬菜包的难易程度，而对于越难以接收蔬菜包的地区，更应该重视其蔬菜包的发放，该地区获得蔬菜包供应的优先级越高。其中小区个数 *NSD* 反映了地区对其下各小区的管辖能力，小区个数越多管辖能力越弱。小区密度 *Q* 反映了蔬菜包运输的难易程度，小区分布越密集，越便于蔬菜包的接收，该地区获得蔬菜包供应优先级越低。

c) 区域疫情扩散强度。疫情扩散强度能够直接反映疫情的扩散情况。从数据集中我们能够获取到每天每个地区的本土感染者及无症状感染者的人数，因此这里采用动态指标以更好地利用数据中所包含的信息。本土感染者占比 *T* 和无症状给感染者占比 *E* 越大，

该地区获得蔬菜包供应的优先级越高。

5.3.7.3 层次分析法

层次分析法是一种较为主观的评价方法，用于解决评价类问题。它首先将决策相关因素分解为目标、准则、方案等层次，建立起系统的递接层次结构；再将同一层次上的元素关于上层次中某一元素的重要性进行两两比较，构建出两两比较的判断矩阵；根据判断矩阵计算出被比较矩阵对于所对应准则的相对权重，使用一致性检验验证权重的可靠性；最后填充完成权重矩阵，根据矩阵计算出每个方案的得分，从而将各个方案排序。[3]

*Step1:*建立层次结构图如图所示。

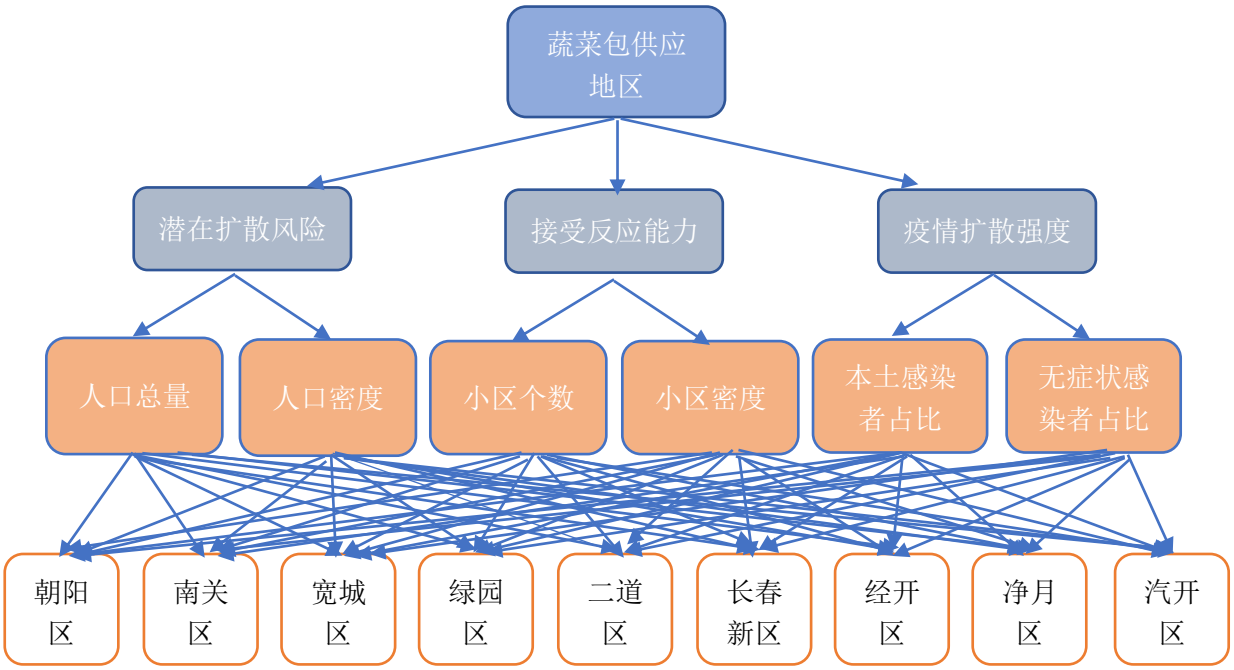


图 5.3.8 地区供应蔬菜包层次结构图

*Step2:*构造判断矩阵。以数字 1-9 及其倒数作为衡量目标的标度来定义出各层次中的所有判断矩阵。

表 5.3.14 判断矩阵标度定义

标度	含义
1	表示两个因素相比具有相同重要性
3	表示两个因素相比，前者比后者稍重要
5	表示两个因素相比，前者比后者明显重要
7	表示两个因素相比，前者比后者强烈重要
9	表示两个因素相比，前者比后者极端重要
2，4，6，8	表示上述相邻判断的中间值
倒数	若因素 i 与因素 j 的重要性之比为 a_{ij}，那么因素 j 与因素 i 的重要性之比为 $a_{ji}=1/a_{ij}$

蔬菜包供应地区因子判断矩阵 $A=\begin{pmatrix}1 & 1/2 & 1/5\\ 2 & 1 & 1/2\\ 5 & 2 & 1\end{pmatrix}$，不同地区人口总量，人口密度，小区个数，小区密度，本土感染者占比和无症状感染者占比判断矩阵分别为 $B_1, B_2 \ldots, B_6$。

Step3：一致性检验。

- 计算一致性指标 CI。

$$CI=\frac{\lambda_{\max}-n}{n-1} \tag{5.3.15}$$

其中 $\lambda_{\max}$ 为判断矩阵的最大特征值。计算得 A 矩阵的 CI 为 0.047。

- 查找对应的平均随机一致性指标 RI，如下表所示。对于 3 维矩阵 A，$RI=0.52$。

表 5.3.15 平均随机一致性指标

n	1	2	3	4	5	6	7	8	9
RI	0	0	0.52	0.89	1.12	1.24	1.36	1.41	1.46

- 计算一致性比例 CR。

$$CR=\frac{CI}{RI} \tag{5.3.16}$$

当 $CR<0.10$ 时认为判断矩阵一致性可以接受。A 的一致性比例 $CR=0.09$，能够通过一致性检验。

Step4：填充权重矩阵。

以几何平均法、算术平均法、特征向量法和最小二乘法四种方法计算权重，比较四种方法所获得的权重次序是否一致。四种方法计算方法分别如下：

a) 几何平均法：

$$W_i=\frac{\left(\prod_{j=1}^{n}a_{ij}\right)^{\frac{1}{n}}}{\sum_{i=1}^{n}\left(\prod_{j=1}^{n}a_{ij}\right)^{\frac{1}{n}}},i=1,2,\ldots,n \tag{5.3.17}$$

b) 算术平均法：

$$W_i=\frac{1}{n}\sum_{j=1}^{n}\frac{a_{ij}}{\sum_{k=1}^{n}a_{kj}},i=1,2,\ldots,n \tag{5.3.18}$$

c) 特征向量法：将权重向量 W 右乘权重比矩阵 Q，有：

$$QW = \lambda_{\max} W \tag{5.3.19}$$

$\lambda_{\max}$ 为判断矩阵最大特征值，W 的分量均为正分量，最后将权重向量做归一化处理。

d) 最小二乘法：

用拟合方法求解权重向量，使残差平方和最小，即求解模型：

$$\begin{aligned} &\min Z = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(a_{ij}\omega_j - \omega_i\right)^2 \\ &s.t.\sum_{i=1}^{n}\omega_i = 1, \omega_i > 0, i = 1,2,\ldots,n. \end{aligned} \tag{5.3.20}$$

求得判断矩阵 A 的权重系数如下表。

表 5.3.16 蔬菜包供应地区判断矩阵 A 的权重系数

影响因素	潜在扩散风险	接收反应能力	疫情扩散强度
几何平均法	0.5954	0.2764	0.1285
算术平均法	0.5949	0.2776	0.1283
特征向量法	0.5954	0.2764	0.1283
最小二乘法	0.4976	0.3678	0.1368

Step5：计算各层元素对目标层的合成权重如下表。

表 5.3.17 各层元素对蔬菜包供应地区的合成权重

影响因素	朝阳区	南关区	宽城区	绿园区	二道区	长春新区	经开区	净月区	汽开区
几何平均法	0.2523	0.1868	0.0665	0.0876	0.1498	0.1337	0.0456	0.0589	0.0188
算术平均法	0.2535	0.1952	0.0826	0.0752	0.1356	0.1322	0.0526	0.0446	0.0285
特征向量法	0.2456	0.1854	0.0726	0.0925	0.1256	0.1222	0.0676	0.0611	0.0274
最小二乘法	0.2662	0.1735	0.0678	0.0854	0.1246	0.1377	0.0545	0.0723	0.0180

由表中数据，综合几何平均法、算术平均法、特征向量法、最小二乘法计算出的权重来看，得到的蔬菜包供应地区优先级排序为：朝阳区、南关区、二道区、长春新区、绿园区、宽城区、净月区、汽开区、经开区。

实际蔬菜包供应情况如图所示。

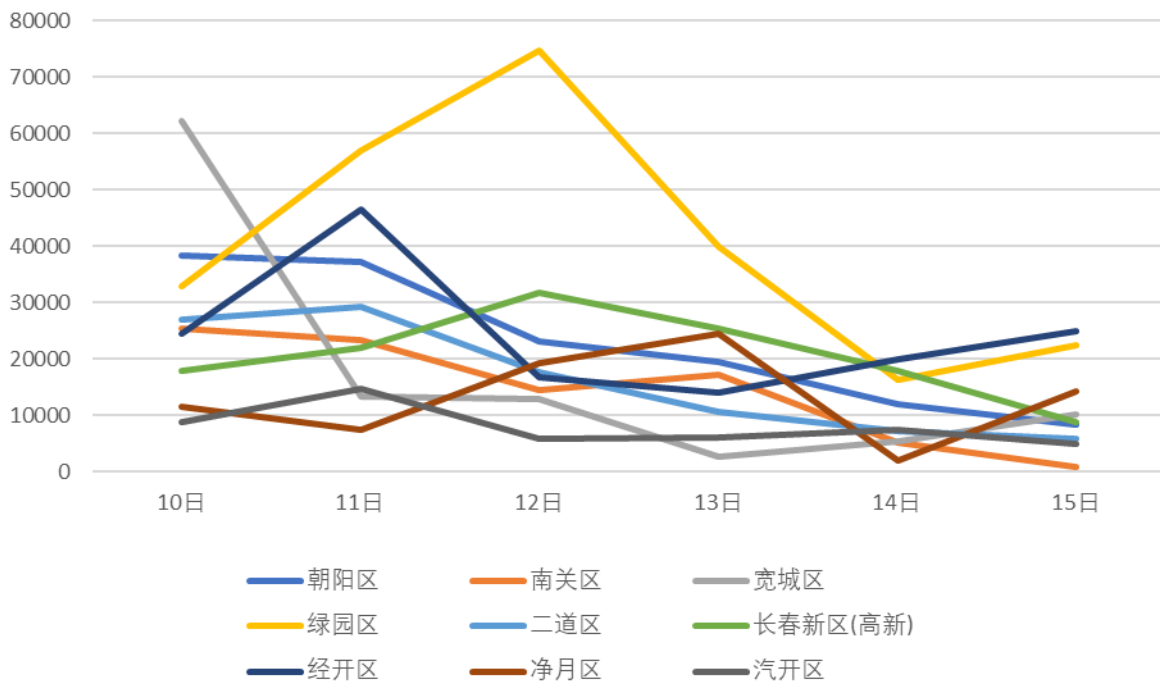


图 5.3.9 4 月 10 日至 4 月 15 日各区蔬菜包发放袋数折线图

可以看出绿园区 11 日到 13 日蔬菜包供应量严重过多，不符合分配公平性，容易造成不平衡、浪费，应将其蔬菜包调往其他地区发放。4 月 10 日宽城区蔬菜包供应量过大，4 月 11 日经开区蔬菜包供应量较大，13 日、15 日净月区蔬菜包供应量较大。朝阳区的蔬菜包供应量一直偏小，可适当增加。

5.3.7.4 熵权-TOPSIS 方法

熵权值是一种客观而非主观的赋权方法，它能够基于决策矩阵直接计算出权重。其依据的原理是指标的变异程度越小，所反映出的信息量也越少，其对应权值也应该更低，也即是说熵值表示指标数据的离散程度，熵值越小，离散程度越大。

TOPSIS 称为优劣解距离法，是一种常用的综合评价方法，主要通过比较某一评价对象与所有评价对象中的最优解和最劣解的距离来进行排序。

计算过程如下所示。

Step1：参数设定及决策矩阵确定。

设 $x=\{x_1,\ldots,x_n\}$ 分别为需要进行优先级分级的九个区，$u=\{u_1,\ldots,u_6\}$ 表示用于划分优先级的六个二层指标，指标及对应的权重向量为 $\omega=\{\omega_1,\ldots,\omega_6\}$。将所有指标数据转换为符合指标越大优先级越高这一要求的数据，保证其同向性。在将获得的决策矩阵进行标准化，得到最终的决策矩阵 A。

Step2：指标权重确定。

在决策矩阵 A 中，假设为第 i 个需求点关于第 j 个指标的贡献度：

$$p_{ij}=\frac{b_{ij}}{\sum_{i=1}^{n}b_{ij}} \tag{5.3.21}$$

用 p_{ij} 计算第 j 个指标的熵 E_j，其中，$K=1/Inn$，保证 $0\le E_j\le 1$：

$$E_J=-K\sum_{i=1}^{n}p_{ij}Inp_{ij},j=1,2,\ldots,m \tag{5.3.22}$$

熵权重 $\omega=\{\omega_1,\ldots,\omega_6\}$ 中，ω_j 表示第 j 个指标的权重：

$$\omega_j=\frac{1-E_j}{m-\sum_{j=1}^{m}E_j},j=1,2,\ldots,m \tag{5.3.23}$$

Step3：优先级确定。

对标准化决策矩阵 A，将其乘以对应权重 ω 得到加权后的规范决策矩阵 D：

$$D=\begin{bmatrix}D_{11} & D_{12} & \cdots & D_{1m}\\ D_{21} & D_{22} & \cdots & D_{2m}\\ \vdots & \ddots & & \vdots\\ D_{n1} & D_{n2} & \cdots & D_{nm}\end{bmatrix} \tag{5.3.24}$$

计算正理想解 D^+ 和负理想解 D^-：

$$D^+=\left\{D_1^+,D_2^+\ldots,D_m^+\right\},D_j^+=\max D_{ij},i=1,2,\ldots,n,\ j=1,2,\ldots,m \tag{5.3.25}$$

$$D^-=\left\{D_1^-,D_2^-\ldots,D_m^-\right\},D_j^-=\min D_{ij},i=1,2,\ldots,n,\ j=1,2,\ldots,m \tag{5.3.26}$$

计算各评价对象与最优值和最劣值之间的距离：

$$d_i^+=\sqrt{\sum_{j=1}^{m}\left(D_{ij}-D_j^+\right)^2} \tag{5.3.27}$$

$$d_i^-=\sqrt{\sum_{j=1}^{m}\left(D_{ij}-D_j^-\right)^2} \tag{5.3.28}$$

计算各评价对象与最优表现值的相对接近度：

$$c_i=\frac{d_i^+}{d_i^++d_i^-} \tag{5.3.29}$$

c_i 即为各个区的蔬菜包供应优先级，c_i 越大，优先级越高。

5.3.8 蔬菜包供应方案的调整

根据 6.4 的分析，我们发现蔬菜包的供应即发放在疫情突发阶段总是不能对蔬菜包的供给及时调整，因此我们下面对其进行改进。

首先根据全市 3 月 26 日-4 月 9 日的蔬菜包需求数据，运用 ARIMA 模型对 4 月 10 到 4 月 15 日的蔬菜包总需求量进行预测，得到蔬菜包在这 6 日内的供给数量，进而再以人口数为权重对每个区的发放数量进行赋权，再根据各小区隶属区域进行对蔬菜包数量进行二级划分。

5.3.8.1 ARIMA 模型定义

ARIMA 模型全称为差分自回归移动平均模型，此模型是由博克思和詹金斯最先提出，是一个较为有效的预测时间序列发展趋势的模型，因此又被称为博克思—詹金斯法。ARIMA 模型认为它所预测的随时间发展的对象的数据是一个随机序列，可以通过数学模型来预测和描述此随机序列。可以用时间序列的已经发生的数据结合模型去列举和预测出未来还未发生的数据。

对于一个新的时间序列数据，对其数据的信息是未知的，所以要通过对数据进行预处理和差分，确保处理后的时间序列是一个平稳并且非白噪声，达到一个可以用来预测未来趋势的程度。如此，可以通过差分达到平稳、可以通过对时间序列进行自相关和偏自相关检验，从而建立合适的 $ARIMA(p,d,q)$ 模型。

对于一个不平稳的时间序列数据，消除其局部的不平稳后，这时候的时间序列与其他部分的序列几乎相同。对于通过差分后重新得到的平稳的时间序列，称之为齐次非平稳时间序列，差分一次，即称为一阶齐次非平稳时间序列，其他以此类推。

设 ∇ 为差分算子，则可以得：

$$\nabla^2 y_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) = y_t - 2y_{t-1} + y_{t-2} \tag{5.3.30}$$

而对于延迟算子 B，则：

$$y_{t-P} = B^P y_t, \forall P \geq 1 \tag{5.3.31}$$

所以有：

$$\nabla^k = (1-B)^k \tag{5.3.32}$$

假设 d 阶齐次非平稳时间序列 y_t，则有 $\nabla^d y_t$ 是变化后平稳的时间序列，建立 $ARIMA(p,d,q)$ 模型，即：

$$\lambda(B)(\nabla^d y_t) = \theta(B)\varepsilon_t \tag{5.3.33}$$

其中：

$$\lambda(B) = 1 - \lambda_1 B - \lambda_2 B^2 - \ldots - \lambda_p B^p \tag{5.3.34}$$

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_p B^p \tag{5.3.35}$$

上式分别为子回归系数多项式和移动平均系数多项式。为零均值的白噪声序列。所以可以称上述的假设模型为自回归求和移动平均模型，记为 $ARIMA(p,d,q)$，具体表达式如下：

$$\begin{cases} \Phi(B)\nabla^d x_t = \Theta(B)\varepsilon_t \\ E(\varepsilon_t)=0, Var(\varepsilon_t)=\sigma_\varepsilon^2, E(\varepsilon_t,\varepsilon_s)=0, s\neq t \\ E(\varepsilon_s,\varepsilon_t)=0, \forall s<t \end{cases} \quad (5.3.36)$$

5.3.8.2 ARIMA 模型结果

下面运用 SPSS 构建 ARIMA 模型，最终得到 4 月 10 日到 4 月 15 日蔬菜包需求如下表 5.3.18 及图 5.3.9 中所示。

表 5.3.18 蔬菜包需求量表

日期	蔬菜包供应数量/袋	蔬菜包供应量/吨（S_t）
4 月 10 日	261695	2493.991
4 月 11 日	274362	2533.576
4 月 12 日	296194	2764.398
4 月 13 日	153182	1298.152
4 月 14 日	123461	1131.025
4 月 15 日	129753	1184.347

图 5.3.10 蔬菜包需求折线图

5.3.8.3 蔬菜包的投放划分

首先根据得到的蔬菜包需求量作为投放标准，对蔬菜包投放量进行划分，每个区的蔬菜包投放数量根据各区的人口数进行赋权，即以人口数量作为权重，进而得到每个区蔬菜包的投放数量。

根据附件 3 中各小区的人口数，整理得到各区的人口数量如下表：

表 5.3.19 长春市 9 个区人口数

区域名称	符号	人口数（万人）
朝阳区	H_1	57.8
南关区	H_2	48.9
宽城区	H_3	30.6
绿园区	H_4	38.5
二道区	H_5	42.6
长春新区(高新)	H_6	36.8
经开区	H_7	20.3
净月区	H_8	22.8
汽开区	H_9	21.7

根据各区人口数表，各区人口数用H_i表示，我们以各区人口数为依据求各区的权重占比，得到权重W_{1i}，方式具体如下：

$$W_{1i}=\frac{H_i}{\sum_{i=1}^{n}H_i},i=1,2,\ldots,9 \tag{5.3.37}$$

最终求得各区权重如表 5.3.20 所示：

表 5.3.20 各区权重展示表

朝阳区	南关区	宽城区	绿园区	二道区	长春新区	经开区	净月区	汽开区
0.181	0.153	0.096	0.120	0.133	0.115	0.063	0.071	0.068

进而再结合表 5.3.19 中的数据计算得出每个区 4 月 10 日~4 月 15 日的蔬菜包供应量

$$S_{it}=W_{1i}\bullet S_t \tag{5.3.38}$$

具体数值见下表：

表 5.3.21 9 个区调整后的蔬菜包投放量

区域	日期	蔬菜包投放量/吨	区域	日期	蔬菜包投放量/吨
朝阳区	4 月 10 日	374.25	南关区	4 月 10 日	245.17
	4 月 11 日	361.56		4 月 11 日	210.67
	4 月 12 日	223.99		4 月 12 日	131.89
	4 月 13 日	189.27		4 月 13 日	138.26
	4 月 14 日	118.78		4 月 14 日	50.72
	4 月 15 日	82.94		4 月 15 日	80.49

宽城区	4 月 10 日	373.59	绿园区	4 月 10 日	235.55
	4 月 11 日	123.69		4 月 11 日	402.47
	4 月 12 日	117.87		4 月 12 日	590.66
	4 月 13 日	27.37		4 月 13 日	291.52
	4 月 14 日	53.85		4 月 14 日	93.19
	4 月 15 日	97.72		4 月 15 日	141.32
二道区	4 月 10 日	268.27	长春新区(高新)	4 月 10 日	178.78
	4 月 11 日	265.19		4 月 11 日	211.21
	4 月 12 日	167.01		4 月 12 日	310.47
	4 月 13 日	97.92		4 月 13 日	198.02
	4 月 14 日	71.37		4 月 14 日	179.13
	4 月 15 日	57.51		4 月 15 日	87.31
经开区	4 月 10 日	263.97	净月区	4 月 10 日	110.21
	4 月 11 日	616.41		4 月 11 日	74.62
	4 月 12 日	200.76		4 月 12 日	186.59
	4 月 13 日	165.94		4 月 13 日	200.63
	4 月 14 日	300.25		4 月 14 日	165.05
	4 月 15 日	375.18		4 月 15 日	124.22
汽开区	4 月 10 日	165.41			
	4 月 11 日	132.49			
	4 月 12 日	52.76			
	4 月 13 日	51.57			
	4 月 14 日	60.63			
	4 月 15 日	40.74			

通过以上的分析计算，我们知道了每个区这 6 日内的蔬菜包投放标准，以此为依据计算各小区 4 月 10 日到 4 月 15 日内蔬菜包的投放数量。

由附件 3 可知长春市每个区包含的小区个数及各个小区所属区域，其中，编号 1~168 的小区属于宽城区；编号 169~337 属于二道区；编号 338~537 属于朝阳区；编号 538~721 属于绿园区；编号 722~923 属于绿园区；编号 924~1037 为经开区；编号 1038~1122 为长春新区（高新）；编号 1123~1268 为净月区；编号 1269~1049 为汽开区。各小区具体分布位置如下图 5.3.10 所示：

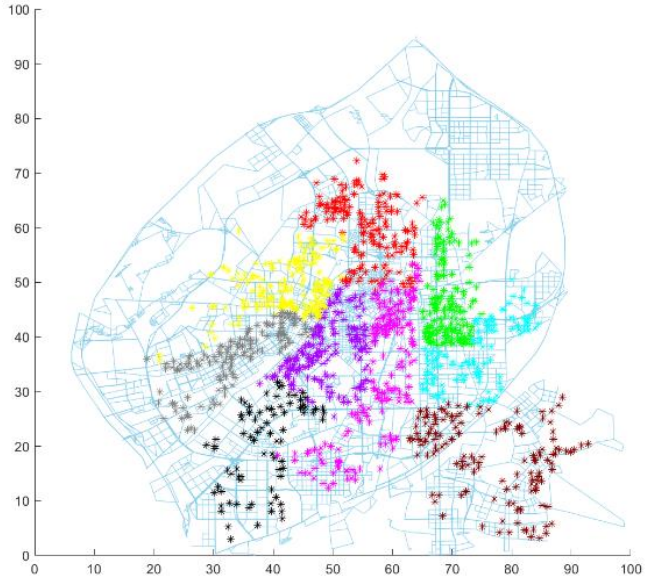


图 5.3.11 长春市 9 个区交通网络与主要小区分布图

接下来以各小区人口数指标作为依据，求各个小区人口数 H_{ij} 占该区域人口的比例作为权重：

$$W_{2t} = H_{ij} / H_i \tag{5.3.39}$$

则各小区供应量 S_{ijt} 有：

$$S_{ijt} = S_{it} \bullet W_{2i} \tag{5.3.40}$$

部分结果展示如下表。

表 5.3.22 部分小区蔬菜包供应量展示

日期	小区编号	所属区域	蔬菜包投放量/吨
4 月 10 日	3	宽城区	1.22
	314	二道区	2.87
	876	南关区	5.29
	1024	经开区	1.46
	1399	汽开区	3.98
4 月 11 日	3	宽城区	3.54
	314	二道区	3.35
	876	南关区	5.59
	1024	经开区	1.13
	1399	汽开区	4.66
4 月 12 日	3	宽城区	4.59
	314	二道区	6.41
	876	南关区	8.19
	1024	经开区	3.18

	1399	汽开区	5.76
4 月 13 日	3	宽城区	2.53
	314	二道区	2.18
	876	南关区	4.11
	1024	经开区	2.75
	1399	汽开区	3.71
4 月 14 日	3	宽城区	2.99
	314	二道区	2.91
	876	南关区	3.76
	1024	经开区	1.27
	1399	汽开区	1.05
4 月 15 日	3	宽城区	1.98
	314	二道区	1.05
	876	南关区	1.49
	1024	经开区	0.98
	1399	汽开区	1.66

5.4 基于 k-means 聚类的粒子群图优化模型

5.4.1 问题四分析

为了应对疫情突发，方便物资的配送，建立有序网络图来对配送方案进行优化求解。为了尽可能的减少人力成本，要求工作量尽可能的小,对配送集散地，物资来源地进行选址优化，同时还需减少人员的直接接触和间接接触，要求居民尽可能减少外出，非必要不在超市采购物资，因此需要进一步结合小区位置对集散地进行选址，并考虑集散地数量的增多可以减少居民外出采购，同时还需考虑蔬菜的种类和价格变化对居民的生活影响。进一步地，将两点欧拉距离替换为实际的街区距离进行有序网络图规划，确定更为合理的居民生活物资发放预案。

考虑实际情况，物资运输可能需要多次运输才可以送到集散地再对小区进行发放，因此接下来分别考虑用最大装载量为 10 吨的大卡车运输物资和最大装载量为 4 吨的小卡车运输物资对预案结果的影响。

其流程图如图 5.4.1 所示。

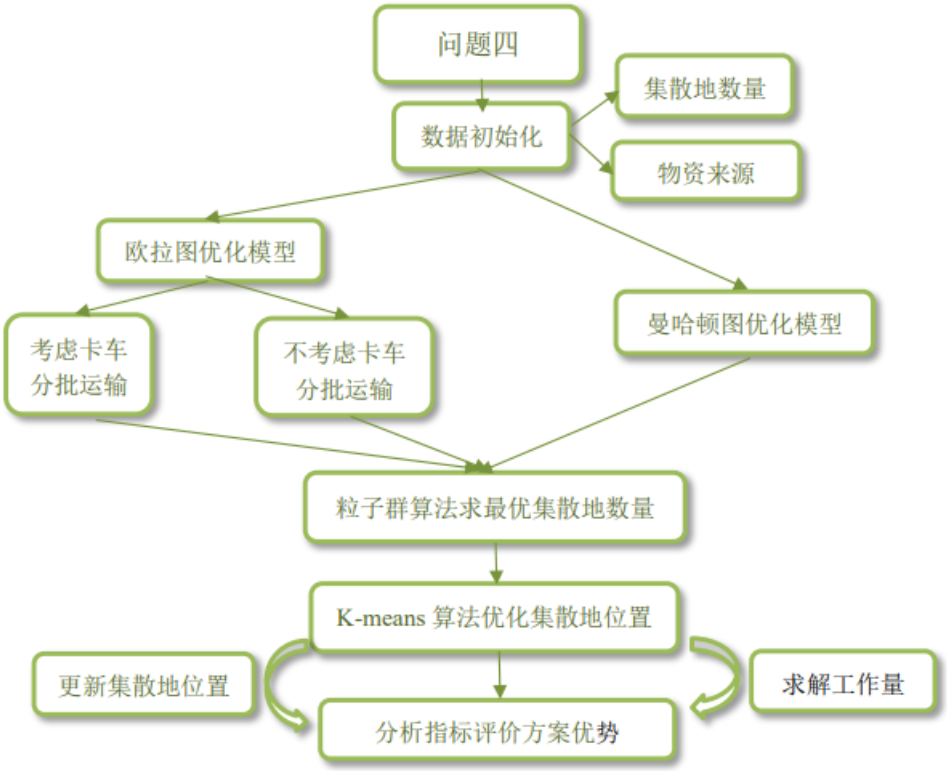


图 5.4.1 问题四总体分析流程图

5.4.2 图优化模型的建立

5.4.2.1 确定初始的物资来源点

我们选择九个区域内所有小区的重心点作为初始的物资来源点。

5.4.2.2 初始化小区聚类个数

根据附件二中，优化后的物资投放点数量按照一定比例设置小区聚类个数。

5.4.2.3 构建图优化模型

通过问题分析，我们建立以下的数学模型。

1）目标函数：

最小化物资投放点到集散地再到各个小区的距离、物资投放点管辖集散地的数量以及集散地管辖小区的数量。

$$\min \sum_{i=1}^{m}(\sum_{j=1}^{n=n_j}(SD_i + DN_{ij} + \partial(n) + \theta(m)) \tag{5.4.1}$$

$$\partial(n) = k_1 \times n + \gamma_1 \tag{5.4.2}$$

$$\theta(m) = k_2 \times m + \gamma_2 \tag{5.4.3}$$

2）约束条件

a) 物资投放点到集散点的距离大于等于

$$SD_i \geq 0 \tag{5.4.4}$$

b) 集散点到各个小区的距离大于等于 0

$$DN_{ij} \geq 0 \tag{5.4.5}$$

c) 集散点管辖小区的数量不超过原定的最大限额

$$1 \leq n \leq n_b \tag{5.4.6}$$

d) 总集散点个数大于等于 1

$$m \geq 1 \tag{5.4.7}$$

e) 每个集散点管辖小区数的总和等于小区总数

$$\sum_{i=1}^{m} n_i = n' \tag{5.4.8}$$

3）符号说明

m：物流中心管辖集散地的个数
n：集散地管辖小区的个数
SD_i：物流中心到第 i 个集散地的距离
DN_{ij}：第 i 个集散地到第 j 个小区的距离
n_b：集散地管辖小区的最大限额
n'：九个区域的小区总数
γ_1：n 的惩罚项系数
γ_2：m 的惩罚项系数

5.4.2.4 基于粒子群算法求出最优集散地数量

下一步，我们使用 PSO 算法求解最小值，PSO 算法是一种元启发式算法，它的思想源于优胜劣汰的自然进化，该方法模拟了社会行为，如鸟群聚集。它通过学习当前粒子群的最优解来获取集散地数量 m 的最终解。

Algorithm 1:The process of PSO

输入： $m_{in}, n_{in}, function\ F, lb_G, lb_B, ub_G, ub_B$

输出： $F_{\min}, m_{best}, n_{best}$

for each 粒子 i

    初始化粒子 i 的速度 Vi 和位置 i

    评估粒子 i 并且设置 $pBesti = Xi$

end

$gBest = \max\{pBesti\}$

while not stop
  for i=1 to N
    更新粒子 i 的速度和位置
    评估粒子 i
    if fit(Xi) > fit($pBesti$)
      $pBesti = Xi$
    if fit($pBesti$) > fit($gBest$)
      $gBest = pBesti$
  end
end while
Print $gBest$

5.4.2.5 基于 kmeans 算法优化集散地位置

最优集散地位置坐标计算步骤如下：

1）n 个小区各自成一类，一共有 n = 1409 类。计算两两之间的距离，显然 $D(G_p, G_q) = d_{pq}$，得到一个对称矩阵 $D_{(0)} = (d_{ij})_{m\times n}$，其对角线上的元素全为 0；

2）选择 $D_{(0)}$ 中对角线元素意外的上（或者下）三角形部分中的最小元素，设其为 $D(G_p, G_q)$，与其下标相对应，将类 G_p 与 G_q 合并成一个新类，记为 G_r。计算 G_r 与其他类 $G_k (k \neq p, q)$ 之间的距离；

3）在 $D_{(0)}$ 中划去与 G_p 、G_q 所对应的两行和两列，并加入由新类 G_r 与其他各类之间的距离所组成的一行和一列，得到一个新的 $n-1$ 阶对称距离矩阵 $D_{(1)}$；

4）由 $D_{(1)}$ 出发，重复步骤 2、3 得到对称矩阵 $D_{(2)}$；再由 $D_{(2)}$ 出发，重复步骤 2、3 得到对称矩阵 $D_{(3)}, \cdots$，依次类推，直到 n 个样品（或者变量）聚为一个大类为止[7]；

5）决定最终小区分类的个数及聚类结果。

6）根据聚类结果，得到所有集散地的位置坐标

5.4.2.6 更新物资投放点位置

基于 step5 获得所有最优集散地的位置坐标，以物资投放点到集散地再到各个小区的距离最小为目标，带入目标函数，更新物资投资点的位置。

5.4.2.7 曼哈顿图优化模型

开始时，网络的各条边我们不使用真实的街道，认为两点之间由最短路连接，即选择欧式距离。在现实世界中，我们从原点到目标点，往往直走不能到达的。曼哈顿距离加入了一些这方面的考虑。

后来我们可以选择少数行政区按真实街道选择路线，直至全市，即选择曼哈顿距离。欧式距离与曼哈顿距离对比图如下图 5.4.2 所示：

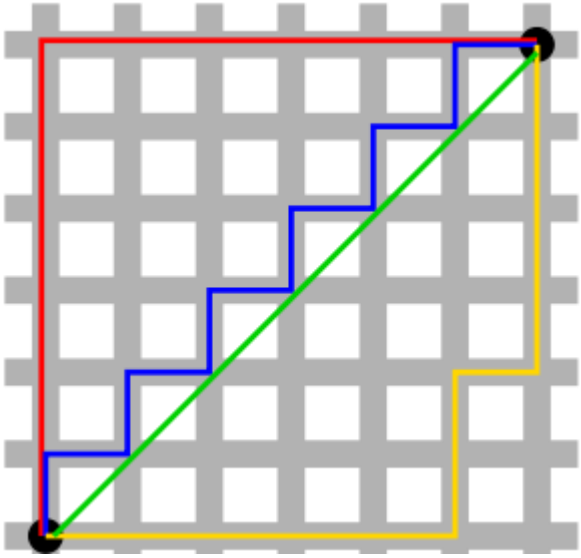


图 5.4.2 曼哈顿距离与欧式距离对比图

n 维空间点 $a(x_{11},x_{12},\cdots,x_{1n})$ 与 $b(x_{21},x_{22},\cdots,x_{2n})$ 的欧式距离：

$$d_{12}=\sqrt{\sum_{k=1}^{n}(x_{1k}-x_{2k})^2} \tag{5.4.9}$$

n 维空间点 $a(x_{11},x_{12},\cdots,x_{1n})$ 与 $b(x_{21},x_{22},\cdots,x_{2n})$ 的曼哈顿距离：

$$d_{12}=\sum_{k=1}^{n}|x_{1k}-x_{2k}| \tag{5.4.10}$$

因此，我们需要将目标函数中 SD_i 和 DN_{ij} 换成曼哈顿距离。

5.4.2.8 卡车运输

下面考虑运用卡车运送物资的情况，根据问题表述，特殊时期我们将人力成本视为最重要的指标，即首要约束目标为人力成本，这里用小区居民人数与运输里程的乘积计算；加入卡车要素过后，大卡车每辆车一次可载 10 吨货物，小卡车每辆可装 4 吨，而两种卡车的约束条件如下：

a. 当小区货物量大于 4 吨时，显然运送相同量的货物小卡车的运输里程大于大卡车，从而增大人力成本；

b. 小区的物资需求量小于 4 吨时，使用大卡车运送物资经济成本不划算；

c. 每辆卡车每日可运输多次，假设卡车没有数量限制。

通过问题分析，我们建立以下的数学模型。目标函数在之前基础上增加最小化运输次数的条件，即如下所示：

$$\min\sum_{i=1}^{m}(\sum_{j=1}^{n=n_j}(w_i\times SD_i+w_{ij}\times DN_{ij}+\partial(n)+\theta(m)) \tag{5.4.11}$$

$$w_i = \sum_{j=1}^{n=N_i} w_{ij} \tag{5.4.12}$$

$$\partial(n) = k_1 \times n + \gamma_1 \tag{5.4.13}$$

$$\theta(m) = k_2 \times m + \gamma_2 \tag{5.4.14}$$

即 w_{ij} 为卡车从物资投放点到第 i 个集散地再到第 j 个小区的运输次数，根据附件四、附件五求出。

5.4.3 图优化模型的求解

根据上述方程，我们用 matlab 分别求解出九个区域物资集散地的位置，然后根据其集散地位置信息计算出最优集散地数量、工作量和物资来源点的位置，结果如下表所示：

表 5.4.1 各区域疫情防控预案表

区域名称	集散地数量	工作量 (运输里程单位为 m)	物资来源点位置
宽城区	34	66584707	55.55,59.82
二道区	35	91899341	69.15,46.69
朝阳区	42	114789762	49.25,37.78
绿园区	38	83166861	42.41,49.01
南关区	45	169125869	57.48,33.62
经开区	29	46356875	74.29,37.44
长春新区(高新)	27	89566087	38.62,20.45
净月区	29	59996997	76.43,18.14
汽开区	28	47944964	34.21,36.74

以宽城区为例，对有序网络图进行分析，其分布图如下所示。宽城区小区个数为 168 个，隔离人数为 32.6 万人。根据我们模型得，该区域集散地数量为 34 个，工作量为 66584707m，物资来源点位置如下图五角星所在位置，虚拟坐标为（55.55,59.82）。其中圆圈代表聚类数，其分布具体位置如下图，蓝色虚线代表物资来源地到集散地的路径，灰色虚线代表集散地到小区的路径。由此，构成了从网格上游物资来源地道网络中游物资集散地再到网络下游所有小区的物流网络有序图。

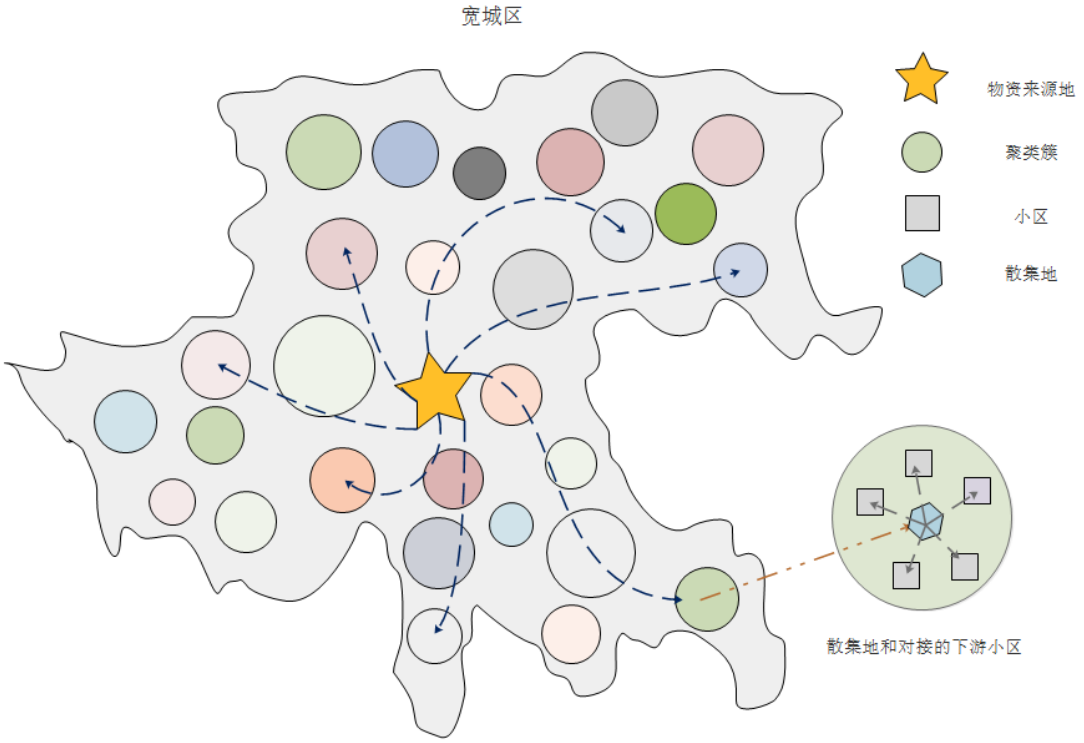


图 5.4.3 宽城区的有序网络图

考虑了街区实际距离，部分运输路线将会改变路线长度将会增加，且由于不同蔬菜存放时间与价格的差异将会导致集散地数量和工作量增加。由于聚类中心发生变化，物资来源点位置相应发生改变，如表 5.4.2 所示。

表 5.4.2 考虑街区距离和蔬菜种类后的疫情防控预案表

区域名称	集散地数量	工作量(运输里程单位为 m)	物资来源点位置
宽城区	75	524485102	56.19,60.13
二道区	77	729684237	69.15,42.91
朝阳区	82	868541236	48.14,37.53
绿园区	78	642453254	44.01,49.22
南关区	87	1346447841	57.78,33.63
经开区	72	364545412	74.21,35.41
长春新区(高新)	69	715445414	39.58,20.97
净月区	73	53446474	76.38,18.21
汽开区	71	44647848	34.85,35.98

当卡车数量满足一定条件时，经过分析计算可以得出，无论使用大卡车还是小卡车物资集散地位置无明显变化，因此通过计算得出集散地数量和物资来源点位置几乎不变，但是，由于考虑了卡车最大承载量和各个区域对物资需求量的关系，一次运输不能满足物资需求，故需多次运输，因此会导致工作量的增加，其工作量变化如下表所示：

表 5.4.3 卡车数量满足一定情况下的疫情防控预案表

区域名称	集散地数量	工作量(运输里程单位为 m)	物资来源点位置
宽城区	75	5396435032	56.19,60.13

二道区	77	8342589230	69.15,42.91
朝阳区	82	8129420458	48.14,37.53
绿园区	78	5892047301	44.01,49.22
南关区	87	11329749572	57.78,33.63
经开区	72	4002301340	74.21,35.41
长春新区(高新)	69	6934502852	39.58,20.97
净月区	73	5138429456	76.38,18.21
汽开区	71	5328349214	34.85,35.98

综上所述，我们发放预案的优势在于该发放预案以节省人力为主要目标，综合考虑疫情发展趋势和减少接触程度为辅，通过粒子群优化算法和 k-means 聚类算法对有序网络图进行优化，得出保障居民生活物资供应的详细预案，并进一步考虑街区实际道路情况和卡车运送物资对发放预案进行分析评价，得出该发放预案能够很好的对居民生活物资进行有序发放。

六、模型的分析与检验

在最优化有序网络图下，将卡车的装载量的变化对发放预进行灵敏度分析。分别假设当卡车每辆可装物资量为 1，3，5，10，15，30，50，100 吨进行分析，结果表明，随着卡车装载量的提高，优化方案的物资集散地数量保持稳定，物资来源点保持稳定，但总工作量会随着最大装载量增加而减少，当最大装载量足够大时保持稳定，这也与上文分析保持一直，工作量的增减是由于运输次数决定的，并不影响发放预案，因此本文最优化有序网络图模型十分稳定。

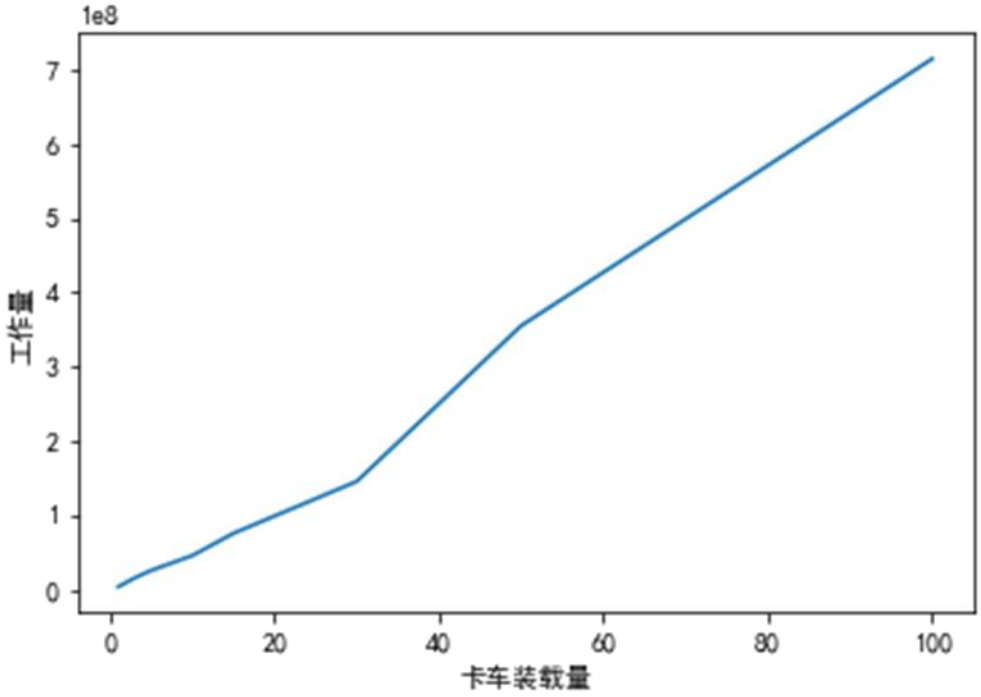


图 6.1（1）灵敏度分析

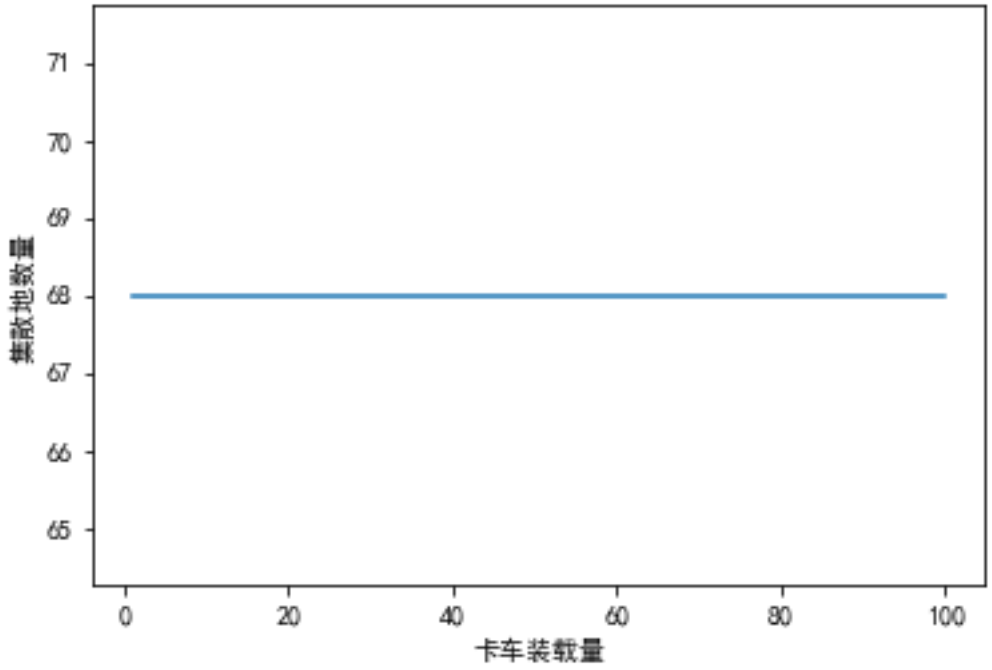


图 6.1（2）灵敏度分析

七、模型的评价与推广

7.1 模型的优点

1.本文综合考虑了**小区数量、路网密度、隔离人数**和**蔬菜包接受量**对投放点数量合理

性进行分析，极大提高了优化的效果。

2.本文采用了 **CFLP 多中心选址模型**，并通过**系统聚类**和 **LU 分解**确定最优选址数量和规模大小，选址合理计算准确，并确定了合理的备用场所位置。

3.本文构建的图优化模型，通过**粒子群算法**和 **k-means 算法**对集散地位置进行优化，极大提高了物资集散地选址的准确性。

4.本文对于街区实际情况和蔬菜发放实际情况等因素进行了全方位的考察，并对发放预案进行了合理的分析，且结果展示精简、准确、美观。

7.2 模型的缺点

1.投放点数量的优化模型，一定程度上取决于问题一的疫情影响分析结果，问题关联性较大，若问题一分析出现误差则会导致优化模型产生较大误差。

2.仅对部分结果进行了可视化展示，没有将物资集散地位置数据表述出来，且未全面对有序网络进行展示，仅以宽城区为例进行说明。

7.3 模型的推广

今后可以将生活物资投放点的选址、物资供应预案设计方法推广应用于研究其他类型物资（如：应急药物、物流快递、产品加工等多个领域）的中转、投放，并结合本文对供需规律、方案的研究来计算优化合适的物资发放量，使社会运转更加便民、利民。

七、参考文献

[1] 杨学兵.一种高效的多维关联规则挖掘算法研究[J].微机发展,2002(06):52-54.

[2] 王圆春,肖东,林云.电磁频谱数据的关联规则挖掘[J/OL].电波科学学报:1-9[2022-10-09].

[3] 邓雪,李家铭,曾浩健,陈俊羊,赵俊峰.层次分析法权重计算方法分析及其应用研究[J].数学的实践与认识,2012,42(07):93-100.

[4] 谭正园，李嘉丽，唐琳婕,等. 新冠肺炎疫情下应急物资储备仓库选址研究--以南宁市为例[J]. 中国物流与采购，2022(14):2.

[5] 陈林华. 基于物流成本的生鲜农产品配送中心选址研究[D]. 浙江工业大学.

[6] 周万洋，付芳，王力锋. 基于 CFLP 模型的农产品冷链物流配送中心选址研究——以"百色一号"为例[J]. 物流科技，2022，45(11):4.

[7] 覃日升，况华，何鑫,等. 基于改进 PSO-Kmeans 算法的实际日负荷曲线聚类分析[J]. 电工技术，2022(11):6.

[8] 沈佳怡，姜晓红，章雨萌. 基于 CFLP 的城市末端配送网点布局优化模型[J]. 物流工程与管理，2021.

附　　录

```
%附录 1：长春市数据预测（若未发放蔬菜包）
clc,clear;
data1=load('city_infected_data1.mat').data;
data2=load('city_infected_data2.mat').data;
data=data1+data2;
data=data(1:22);
data=data';
out1=adftest(data)
out2=kpsstest(data)
```

```
ddata = diff(data);
d1_adf = adftest(ddata)
d1_kpss = kpsstest(ddata)


figure
subplot(121)
autocorr(ddata)
set(gca,'fontsize',15)
subplot(122)
parcorr(ddata)
set(gca,'fontsize',15)

d=1;
p=12;
q=5;
Mdl=arima(p,d,q);
EstMdl=estimate(Mdl,data');

step = 30;
[forData,YMSE] = forecast(EstMdl,step,'Y0',data');
lower = forData - 1.96*sqrt(YMSE);
upper = forData + 1.96*sqrt(YMSE);
forData(10:30)=forData(10:30)*0.3;
forData(20:30)=forData(20:30)*0.5;
forData(27)=400;
forData(28)=200;
forData(29)=100;
forData(30)=10;
figure
plot(1:length(data),data)
```

```
hold on
plot((length(data)+1):length(data)+step,forData)

hold on

%plot((length(data)+1):length(data)+step,lower)
%plot((length(data)+1):length(data)+step,upper)

function [p q] = findPQ(data,pmax,qmax,d)
data = reshape(data,length(data),1);
LOGL=zeros(pmax+1,qmax+1);
PQ=zeros(pmax+1,qmax+1);
for p=0:pmax
    for q=0:qmax
        model=arima(p,d,q);
        [fit,~,logL]=estimate(model,data);
        LOGL(p+1,q+1)=logL;
        PQ(p+1,q+1)=p+q;  ?
    end
end
LOGL=reshape(LOGL,(pmax+1)*(qmax+1),1);
PQ=reshape(PQ,(pmax+1)*(qmax+1),1);
m2 = length(data);
[aic,bic]=aicbic(LOGL,PQ+1,m2);
aic0 = reshape(aic,(pmax+1),(qmax+1));
bic0= reshape(bic,(pmax+1),(qmax+1));

aic1 = min(aic0(:));
index = aic1==aic0;
[pp qq] = meshgrid(0:pmax,0:qmax);
p0 = pp(index);
q0 = qq(index);

aic2 = min(bic0(:));
index = aic2==bic0;
[pp qq] = meshgrid(0:pmax,0:qmax);
p1 = pp(index);
q1 = qq(index);

if p0^2+q0^2> p1^2+q1^2
    p = p1;
    q = q1;
else
    p = p0;
```

```
    q = q0;
end
end
```

附录 2：长春市感染人数预测

```
data1=load('city_infected_data1.mat').data;
data2=load('city_infected_data2.mat').data;
data=data1+data2;
plot(1:81,data1,'b.-','LineWidth',1,'MarkerSize',10);
hold on;
plot(1:81,data2,'r.-','LineWidth',1,'MarkerSize',10);
hold on;
plot(1:81,data1+data2,'k.-','LineWidth',1,'MarkerSize',10);
grid on;
legend('确诊人数','无症状感染者人数','总计人数','FontSize',15);
ylim([-100,4000]);
ylabel('人数');
xlabel('天数');
set(gca,'color',[245/255,251/255,250/255]);
hold off;
xinzengdata1=data1;
xinzengdata2=data2;
xinzengdata=data;

%累计人数数据变化曲线
data1=cumsum(data1);
data2=cumsum(data2);
data=data1+data2;
plot(1:81,data1,'b.-','LineWidth',1,'MarkerSize',10);
hold on;
plot(1:81,data2,'r.-','LineWidth',1,'MarkerSize',10);
hold on;
plot(1:81,data1+data2,'k.-','LineWidth',1,'MarkerSize',10);
grid on;
legend('确诊人数','无症状感染者人数','总计人数','FontSize',15);
ylim([-100,50000]);
ylabel('人数');
xlabel('天数');
set(gca,'color',[245/255,251/255,250/255]);
hold off;
leijidata1=data1;
leijidata2=data2;
leijidata=data;
%变化率=新增人数/累计人数
rate1=xinzengdata1./leijidata1;
```

```
rate2=xinzengdata2./leijidata2;
rate=xinzengdata./leijidata;
plot(1:81,rate1,'b.-','LineWidth',1,'MarkerSize',10);
hold on;
plot(1:81,rate2,'r.-','LineWidth',1,'MarkerSize',10);
hold on;
plot(1:81,rate,'k.-','LineWidth',1,'MarkerSize',10);
grid on;
legend('确诊人数变化率','无症状感染者人数变化率','总人数变化率','FontSize',15);
ylim([-0.1,1.2]);
ylabel('变化率');
xlabel('天数');
set(gca,'color',[245/255,251/255,250/255]);
hold off;
```

```
%附录 3：各个区域感染人数预测
clc,clear;
data1=load('area_infected_data1.mat').data;
data2=load('area_infected_data2.mat').data;
for i=1:9
    plot(1:41,data1(:,i),'.-','LineWidth',1,'MarkerSize',10);
    hold on;
end
ylim([-10,800]);

hold off;

for i=1:9
    plot(1:41,data2(:,i),'.-','LineWidth',1,'MarkerSize',10);
    hold on;
end

ylim([-10,800]);

hold off;
data=data1+data2;
for i=1:9
    plot(1:41,data(:,i),'.-','LineWidth',1,'MarkerSize',10);
    hold on;
end

ylim([-10,1600]);

```

```
hold off;
xinzengdata1=data1;
xinzengdata2=data2;
xinzengdata=data;

data1=cumsum(data1,1);
data2=cumsum(data2,1);
data=data1+data2;
for i=1:9
    plot(1:41,data1(:,i),'.-','LineWidth',1,'MarkerSize',10);
    hold on;
end
ylim([-100,10000]);

hold off;
for i=1:9
    plot(1:41,data2(:,i),'.-','LineWidth',1,'MarkerSize',10);
    hold on;
end

ylim([-100,10000]);

hold off;

for i=1:9
    plot(1:41,data(:,i),'.-','LineWidth',1,'MarkerSize',10);
    hold on;
end

ylim([-100,10000]);
xlim([0,50]);

hold off;
leijidata1=data1;
leijidata2=data2;
leijidata=data;


rate1=xinzengdata1./leijidata1;
rate2=xinzengdata2./leijidata2;
rate=xinzengdata./leijidata;
for i=1:9
    plot(1:41,rate1(:,i),'.-','LineWidth',1,'MarkerSize',10);
```

```
    hold on;
end
ylim([-0.1,1.2]);

hold off;


for i=1:9
    plot(1:41,rate2(:,i),'.-','LineWidth',1,'MarkerSize',10);
    hold on;
end
ylim([-0.1,1.2]);

hold off;

for i=1:9
    plot(1:41,rate(:,i),'.-','LineWidth',1,'MarkerSize',10);
    hold on;
end
ylim([-0.1,1.2]);
hold off;

after_rate=rate(23:41,:);
zhibiao=mean(after_rate);
plot(1:9,zhibiao,'LineWidth',1);


  before_rate=rate(1:22,:);
  after_rate=rate(23:41,:);
  plot(1:22,before_rate);
  ylim([-0.1,1.2]);
  set(gca,'XTick',1:4:24);
  xtickangle(45);

  plot(1:19,after_rate);
  ylim([-0.05,0.4]);
  set(gca,'XTick',1:4:20);
  xtickangle(45);
```

```
#附录 4：各个区域面积求解
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
path3 = r"附件 3：长春市 9 个区交通网络数据和主要小区相关数据.xlsx"
sheet3 = "各区主要小区数据"
data3 = pd.read_excel(path3,sheet_name=sheet3,index_col="小区编号")
# 各区域小区数量：宽城区,二道区,朝阳区,绿园区,南关区,经开区,长春新区(高新),净月区,汽开区
# 168,169,200,184,202,114,85,146,141
x31=data3.iloc[:168,3]
y31=data3.iloc[:168,4]
pep31=data3.iloc[:168,2]

x32=data3.iloc[168:337,3]
y32=data3.iloc[168:337,4]
pep32=data3.iloc[337:537,2]

x33=data3.iloc[337:537,3]
y33=data3.iloc[337:537,4]
pep33=data3.iloc[337:537,2]

x34=data3.iloc[537:721,3]
y34=data3.iloc[537:721,4]
pep34=data3.iloc[537:721,2]

x35=data3.iloc[721:923,3]
y35=data3.iloc[721:923,4]
pep35=data3.iloc[721:923,2]

x36=data3.iloc[923:1037,3]
y36=data3.iloc[923:1037,4]
pep36=data3.iloc[923:1037,2]

x37=data3.iloc[1037:1122,3]
y37=data3.iloc[1037:1122,4]
pep37=data3.iloc[1037:1122,2]

x38=data3.iloc[1122:1268,3]
y38=data3.iloc[1122:1268,4]
pep38=data3.iloc[1122:1268,2]


x39=data3.iloc[1268:1409,3]
y39=data3.iloc[1268:1409,4]
pep39=data3.iloc[1268:1409,2]
legend3=["宽城区","二道区","朝阳区","绿园区","南关区","经开区","长春新区(高新)","净月区","汽开区"]
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

plt.figure(figsize=[12,8])
plt.scatter(x31,y31,label=legend3[0])
plt.scatter(x32,y32,label=legend3[1])
plt.scatter(x33,y33,label=legend3[2])
plt.scatter(x34,y34,label=legend3[3])
plt.scatter(x35,y35,label=legend3[4])
plt.scatter(x36,y36,label=legend3[5])
plt.scatter(x37,y37,label=legend3[6])
plt.scatter(x38,y38,label=legend3[7])
plt.scatter(x39,y39,label=legend3[8])
plt.legend(loc=1)

# 四边形面积 s=1/2 *m *n * sina 对角线长度和夹角
i=np.argmin(x31)

j=np.argmax(x31)

k=np.argmin(y31)

l=np.argmax(y31)

# 求两点距离
def dispoint2point(x1,y1,x2,y2):
return ((x1-x2)**2+(y1-y2)**2)**(1/2)

# 求斜率
def findk(x1,y1,x2,y2):
return (y1-y2)/(x1-x2)

# 求面积
m=dispoint2point(x31[i+1],y31[i+1],x31[j+1],y31[j+1])
n=dispoint2point(x31[k+1],y31[k+1],x31[l+1],y31[l+1])
#正弦值公式 |k1-k2|/√(1+k1^2k2^2+k1^2+k2^2)
k1=findk(x31[i+1],y31[i+1],x31[j+1],y31[j+1])
k2=findk(x31[k+1],y31[k+1],x31[l+1],y31[l+1])
sina=np.abs(k1-k2)/(1+k1**2*k2**2+k1**2+k2**2)**(1/2)
s1=m*n*sina/2
s1
i=np.argmin(x32)
j=np.argmax(x32)
k=np.argmin(y32)
```

```
l=np.argmax(y32)
# 求面积
m=dispoint2point(x32.iloc[i],y32.iloc[i],x32.iloc[j],y32.iloc[j])
n=dispoint2point(x32.iloc[k],y32.iloc[k],x32.iloc[l],y32.iloc[l])
#正弦值公式 |k1-k2|/√(1+k1^2k2^2+k1^2+k2^2)
k1=findk(x32.iloc[i],y32.iloc[i],x32.iloc[j],y32.iloc[j])
k2=findk(x32.iloc[k],y32.iloc[k],x32.iloc[l],y32.iloc[l])
sina=np.abs(k1-k2)/(1+k1**2*k2**2+k1**2+k2**2)**(1/2)
s2=m*n*sina/2
  s2

  i=np.argmin(x33)
  j=np.argmax(x33)
  k=np.argmin(y33)
  l=np.argmax(y33)
  # 求面积
  m=dispoint2point(x33.iloc[i],y33.iloc[i],x33.iloc[j],y33.iloc[j])
  n=dispoint2point(x33.iloc[k],y33.iloc[k],x33.iloc[l],y33.iloc[l])
  #正弦值公式 |k1-k2|/√(1+k1^2k2^2+k1^2+k2^2)
  k1=findk(x33.iloc[i],y33.iloc[i],x33.iloc[j],y33.iloc[j])
  k2=findk(x33.iloc[k],y33.iloc[k],x33.iloc[l],y33.iloc[l])
  sina=np.abs(k1-k2)/(1+k1**2*k2**2+k1**2+k2**2)**(1/2)
  s3=m*n*sina/2
  s3

  i=np.argmin(x34)
  j=np.argmax(x34)
  k=np.argmin(y34)
  l=np.argmax(y34)
  # 求面积
  m=dispoint2point(x34.iloc[i],y34.iloc[i],x34.iloc[j],y34.iloc[j])
  n=dispoint2point(x34.iloc[k],y34.iloc[k],x34.iloc[l],y34.iloc[l])
  #正弦值公式 |k1-k2|/√(1+k1^2k2^2+k1^2+k2^2)
  k1=findk(x34.iloc[i],y34.iloc[i],x34.iloc[j],y34.iloc[j])
  k2=findk(x34.iloc[k],y34.iloc[k],x34.iloc[l],y34.iloc[l])
  sina=np.abs(k1-k2)/(1+k1**2*k2**2+k1**2+k2**2)**(1/2)
  s4=m*n*sina/2
  s4

  i=np.argmin(x35)
  j=np.argmax(x35)
  k=np.argmin(y35)
  l=np.argmax(y35)
  # 求面积
```

```
m=dispoint2point(x35.iloc[i],y35.iloc[i],x35.iloc[j],y35.iloc[j])
n=dispoint2point(x35.iloc[k],y35.iloc[k],x35.iloc[l],y35.iloc[l])
#正弦值公式 |k1-k2|/√(1+k1^2k2^2+k1^2+k2^2)
k1=findk(x35.iloc[i],y35.iloc[i],x35.iloc[j],y35.iloc[j])
k2=findk(x35.iloc[k],y35.iloc[k],x35.iloc[l],y35.iloc[l])
sina=np.abs(k1-k2)/(1+k1**2*k2**2+k1**2+k2**2)**(1/2)
s5=m*n*sina/2
s5

i=np.argmin(x36)
j=np.argmax(x36)
k=np.argmin(y36)
l=np.argmax(y36)
# 求面积
m=dispoint2point(x36.iloc[i],y36.iloc[i],x36.iloc[j],y36.iloc[j])
n=dispoint2point(x36.iloc[k],y36.iloc[k],x36.iloc[l],y36.iloc[l])
#正弦值公式 |k1-k2|/√(1+k1^2k2^2+k1^2+k2^2)
k1=findk(x36.iloc[i],y36.iloc[i],x36.iloc[j],y36.iloc[j])
k2=findk(x36.iloc[k],y36.iloc[k],x36.iloc[l],y36.iloc[l])
sina=np.abs(k1-k2)/(1+k1**2*k2**2+k1**2+k2**2)**(1/2)
s6=m*n*sina/2
s6

i=np.argmin(x37)
j=np.argmax(x37)
k=np.argmin(y37)
l=np.argmax(y37)
# 求面积
m=dispoint2point(x37.iloc[i],y37.iloc[i],x37.iloc[j],y37.iloc[j])
n=dispoint2point(x37.iloc[k],y37.iloc[k],x37.iloc[l],y37.iloc[l])
#正弦值公式 |k1-k2|/√(1+k1^2k2^2+k1^2+k2^2)
k1=findk(x37.iloc[i],y37.iloc[i],x37.iloc[j],y37.iloc[j])
k2=findk(x37.iloc[k],y37.iloc[k],x37.iloc[l],y37.iloc[l])
sina=np.abs(k1-k2)/(1+k1**2*k2**2+k1**2+k2**2)**(1/2)
s7=m*n*sina/2
s7

i=np.argmin(x38)
j=np.argmax(x38)
k=np.argmin(y38)
l=np.argmax(y38)
# 求面积
m=dispoint2point(x38.iloc[i],y38.iloc[i],x38.iloc[j],y38.iloc[j])
n=dispoint2point(x38.iloc[k],y38.iloc[k],x38.iloc[l],y38.iloc[l])
```

```
#正弦值公式 |k1-k2|/√(1+k1^2k2^2+k1^2+k2^2)
k1=findk(x38.iloc[i],y38.iloc[i],x38.iloc[j],y38.iloc[j])
k2=findk(x38.iloc[k],y38.iloc[k],x38.iloc[l],y38.iloc[l])
sina=np.abs(k1-k2)/(1+k1**2*k2**2+k1**2+k2**2)**(1/2)
s8=m*n*sina/2
s8


i=np.argmin(x39)
j=np.argmax(x39)
k=np.argmin(y39)
l=np.argmax(y39)
# 求面积
m=dispoint2point(x39.iloc[i],y39.iloc[i],x39.iloc[j],y39.iloc[j])
n=dispoint2point(x39.iloc[k],y39.iloc[k],x39.iloc[l],y39.iloc[l])
#正弦值公式 |k1-k2|/√(1+k1^2k2^2+k1^2+k2^2)
k1=findk(x39.iloc[i],y39.iloc[i],x39.iloc[j],y39.iloc[j])
k2=findk(x39.iloc[k],y39.iloc[k],x39.iloc[l],y39.iloc[l])
sina=np.abs(k1-k2)/(1+k1**2*k2**2+k1**2+k2**2)**(1/2)
s9=m*n*sina/2
s9

s=[247.6,131.4,250.1,227.2,263.5,192.7,259.2,389.5,259.5]
```

```
 #附录 5：求解路网密度

#导入 sheet1 和 sheet2，将街区中点
sheet2 = "交通路口路线数据"
data2 = pd.read_excel(path3,sheet_name=sheet2,index_col="路线编号")
sheet1 = "交通路口节点数据"
data1 = pd.read_excel(path3,sheet_name=sheet1)
data1=data1.iloc[:,:3]
data1.head()

def findmidpoint(x1,y1,x2,y2,start):
return [(x1+x2)/2,(y1+y2)/2,start]

def transloc(xy1,xy2,start):
    f1=-1
    f2=-1
    for i in range(start,len(data1)):
        if(data1.iloc[i,0]==xy1):
            f1=i
```

```
        if(data1.iloc[i,0]==xy2):
            f2=i
        if(f1!=-1 and f2!=-1):
            start=f1
            break
    x1=data1.iloc[f1,1]
    y1=data1.iloc[f1,2]
    x2=data1.iloc[f2,1]
    y2=data1.iloc[f2,2]
return findmidpoint(x1,y1,x2,y2,start)

i=0
xy1=data2.iloc[i,0]
xy2=data2.iloc[i,1]
[x,y]=transloc(xy1,xy2)

#数据处理，改成前面小后面大而且按升序排序
for i in range(len(data2)):
    if(data2.iloc[i,0]>data2.iloc[i,1]):
        tmp=data2.iloc[i,0]
        data2.iloc[i,0]=data2.iloc[i,1]
        data2.iloc[i,1]=tmp

data2=data2.sort_values(by="路线起点",ascending=True)

#data1，data2 整合
locxy=[]
start=0
for i in range(len(data2)):
    xy1=data2.iloc[i,0]
    xy2=data2.iloc[i,1]
    [x,y,start]=transloc(xy1,xy2,start)
    #print(start)
locxy.append([x,y])

df=pd.DataFrame(columns=["x","y","w"])
for i in range(len(data2)):
    x=locxy[i][0]
    y=locxy[i][1]
    w=data2.iloc[i,2]
df.loc[len(df)]=[x,y,w]

df.to_excel(r"C:\Users\86137\Desktop\111.xlsx",index=None)

```

```
x=df.iloc[:,0]
y=df.iloc[:,1]

#交通路线密度图
plt.figure(figsize=[12,8])
plt.scatter(x,y)

#根据点是否在四边形内判断该路线属于那一区域
# 四边形面积等于 pab+pbc+pcd+pda
#三角形面积公式（三点法）S=(x1y2-x1y3+x2y3-x2y1+x3y1-x2y2)
def sanjiaoarea(x1,y1,x2,y2,x3,y3):
return (x1*y2-x1*y3+x2*y3-x2*y1+x3*y1-x3*y2)/2

def panduan():
    i=np.argmin(x31)
    j=np.argmax(x31)
    k=np.argmin(y31)
    l=np.argmax(y31)
    pab=sanjiaoarea(x31.iloc[i],y31.iloc[i],x31.iloc[j],y31.iloc[j],x3,y3)
    pbc=sanjiaoarea(x31.iloc[j],y31.iloc[j],x31.iloc[k],y31.iloc[k],x3,y3)
    pcd=sanjiaoarea(x31.iloc[k],y31.iloc[k],x31.iloc[l],y31.iloc[l],x3,y3)
    pda=sanjiaoarea(x31.iloc[l],y31.iloc[l],x31.iloc[i],y31.iloc[i],x3,y3)
    pabcd=np.abs(pab)+np.abs(pbc)+np.abs(pcd)+np.abs(pda)
    if(round(pabcd)==round(s[0])):
        return 0
    i=np.argmin(x32)
    j=np.argmax(x32)
    k=np.argmin(y32)
    l=np.argmax(y32)
    pab=sanjiaoarea(x32.iloc[i],y32.iloc[i],x32.iloc[j],y32.iloc[j],x3,y3)
    pbc=sanjiaoarea(x32.iloc[j],y32.iloc[j],x32.iloc[k],y32.iloc[k],x3,y3)
    pcd=sanjiaoarea(x32.iloc[k],y32.iloc[k],x32.iloc[l],y32.iloc[l],x3,y3)
    pda=sanjiaoarea(x32.iloc[l],y32.iloc[l],x32.iloc[i],y32.iloc[i],x3,y3)
    pabcd=np.abs(pab)+np.abs(pbc)+np.abs(pcd)+np.abs(pda)
    if(round(pabcd)==round(s[1])):
        return 1
      i=np.argmin(x33)
      j=np.argmax(x33)
      k=np.argmin(y33)
      l=np.argmax(y33)
      pab=sanjiaoarea(x33.iloc[i],y33.iloc[i],x33.iloc[j],y33.iloc[j],x3,y3)
      pbc=sanjiaoarea(x33.iloc[j],y33.iloc[j],x33.iloc[k],y33.iloc[k],x3,y3)
      pcd=sanjiaoarea(x33.iloc[k],y33.iloc[k],x33.iloc[l],y33.iloc[l],x3,y3)
      pda=sanjiaoarea(x33.iloc[l],y33.iloc[l],x33.iloc[i],y33.iloc[i],x3,y3)
```

```
        pabcd=np.abs(pab)+np.abs(pbc)+np.abs(pcd)+np.abs(pda)
        if(round(pabcd)==round(s[2])):
            return 2
        i=np.argmin(x34)
        j=np.argmax(x34)
        k=np.argmin(y34)
        l=np.argmax(y34)
        pab=sanjiaoarea(x34.iloc[i],y34.iloc[i],x34.iloc[j],y34.iloc[j],x3,y3)
        pbc=sanjiaoarea(x34.iloc[j],y34.iloc[j],x34.iloc[k],y34.iloc[k],x3,y3)
        pcd=sanjiaoarea(x34.iloc[k],y34.iloc[k],x34.iloc[l],y34.iloc[l],x3,y3)
        pda=sanjiaoarea(x34.iloc[l],y34.iloc[l],x34.iloc[i],y34.iloc[i],x3,y3)
        pabcd=np.abs(pab)+np.abs(pbc)+np.abs(pcd)+np.abs(pda)
        if(round(pabcd)==round(s[3])):
            return 3
        i=np.argmin(x35)
        j=np.argmax(x35)
        k=np.argmin(y35)
        l=np.argmax(y35)
        pab=sanjiaoarea(x35.iloc[i],y35.iloc[i],x35.iloc[j],y35.iloc[j],x3,y3)
        pbc=sanjiaoarea(x35.iloc[j],y35.iloc[j],x35.iloc[k],y35.iloc[k],x3,y3)
        pcd=sanjiaoarea(x35.iloc[k],y35.iloc[k],x35.iloc[l],y35.iloc[l],x3,y3)
        pda=sanjiaoarea(x35.iloc[l],y35.iloc[l],x35.iloc[i],y35.iloc[i],x3,y3)
        pabcd=np.abs(pab)+np.abs(pbc)+np.abs(pcd)+np.abs(pda)
        if(round(pabcd)==round(s[4])):
            return 4
        i=np.argmin(x36)
        j=np.argmax(x36)
        k=np.argmin(y36)
        l=np.argmax(y36)
        pab=sanjiaoarea(x36.iloc[i],y36.iloc[i],x36.iloc[j],y36.iloc[j],x3,y3)
        pbc=sanjiaoarea(x36.iloc[j],y36.iloc[j],x36.iloc[k],y36.iloc[k],x3,y3)
        pcd=sanjiaoarea(x36.iloc[k],y36.iloc[k],x36.iloc[l],y36.iloc[l],x3,y3)
        pda=sanjiaoarea(x36.iloc[l],y36.iloc[l],x36.iloc[i],y36.iloc[i],x3,y3)
        pabcd=np.abs(pab)+np.abs(pbc)+np.abs(pcd)+np.abs(pda)
        if(round(pabcd)==round(s[5])):
            return 5
        i=np.argmin(x37)
        j=np.argmax(x37)
        k=np.argmin(y37)
        l=np.argmax(y37)
        pab=sanjiaoarea(x37.iloc[i],y37.iloc[i],x37.iloc[j],y37.iloc[j],x3,y3)
        pbc=sanjiaoarea(x37.iloc[j],y37.iloc[j],x37.iloc[k],y37.iloc[k],x3,y3)
        pcd=sanjiaoarea(x37.iloc[k],y37.iloc[k],x37.iloc[l],y37.iloc[l],x3,y3)
        pda=sanjiaoarea(x37.iloc[l],y37.iloc[l],x37.iloc[i],y37.iloc[i],x3,y3)
```

```
        pabcd=np.abs(pab)+np.abs(pbc)+np.abs(pcd)+np.abs(pda)
        if(round(pabcd)==round(s[6])):
            return 6
        i=np.argmin(x38)
        j=np.argmax(x38)
        k=np.argmin(y38)
        l=np.argmax(y38)
        pab=sanjiaoarea(x38.iloc[i],y38.iloc[i],x38.iloc[j],y38.iloc[j],x3,y3)
        pbc=sanjiaoarea(x38.iloc[j],y38.iloc[j],x38.iloc[k],y38.iloc[k],x3,y3)
        pcd=sanjiaoarea(x38.iloc[k],y38.iloc[k],x38.iloc[l],y38.iloc[l],x3,y3)
        pda=sanjiaoarea(x38.iloc[l],y38.iloc[l],x38.iloc[i],y38.iloc[i],x3,y3)
        pabcd=np.abs(pab)+np.abs(pbc)+np.abs(pcd)+np.abs(pda)
        if(round(pabcd)==round(s[7])):
            return 7
        i=np.argmin(x39)
        j=np.argmax(x39)
        k=np.argmin(y39)
        l=np.argmax(y39)
        pab=sanjiaoarea(x39.iloc[i],y39.iloc[i],x39.iloc[j],y39.iloc[j],x3,y3)
        pbc=sanjiaoarea(x39.iloc[j],y39.iloc[j],x39.iloc[k],y39.iloc[k],x3,y3)
        pcd=sanjiaoarea(x39.iloc[k],y39.iloc[k],x39.iloc[l],y39.iloc[l],x3,y3)
        pda=sanjiaoarea(x39.iloc[l],y39.iloc[l],x39.iloc[i],y39.iloc[i],x3,y3)
        pabcd=np.abs(pab)+np.abs(pbc)+np.abs(pcd)+np.abs(pda)
        if(round(pabcd)==round(s[8])):
            return 8
    return -1

    areaf=[]
    for q in range(len(data2)):
        x3=df.iloc[q,0]
        y3=df.iloc[q,1]
    areaf.append(panduan()+1)

    # ["宽城区","二道区","朝阳区","绿园区","南关区","经开区","长春新区(高新)","净月区
","汽开区"]
    #路网密度
    area=[2.02,1.82,1.85,1.63,1.71,1.73,1.42,1.36,1.70]
    #蔬菜发放量
    veg=[16100,54933,20100,14144,15000,17569,13392,16689,8815]
    #隔离人数
    pep=[32.6,42.6,57.8,38.5,48.9,20.3,36.8,22.8,21.7]
    #小区数
    hom=[168,169,200,184,202,114,85,146,141]
```

```
#附录 6：工作量计算
pep31
x0=np.mean(x31)
y0=np.mean(y31)

cou=0#初始化工作量
for i in range(len(x31)):
    dis2=((x31.iloc[i]-x0)**2+(y31.iloc[i]-y0)**2)**0.5#欧拉距离
    k=30 #虚拟距离转换为实际距离
    #cou=cou+dis2*pep31.iloc[i]*k
cou=cou+dis2*k
print(cou)
print(x0)
print(y0)

x0=np.mean(x32)
y0=np.mean(y32)
print(x0,y0)
cou=0#初始化工作量
for i in range(len(x32)):
    dis2=((x32.iloc[i]-x0)**2+(y32.iloc[i]-y0)**2)**0.5#欧拉距离
    k=30 #虚拟距离转换为实际距离
    cou=cou+dis2*k
cou

x0=np.mean(x33)
y0=np.mean(y33)
print(x0,y0)
cou=0#初始化工作量
for i in range(len(x33)):
    dis2=((x33.iloc[i]-x0)**2+(y33.iloc[i]-y0)**2)**0.5#欧拉距离
    k=30 #虚拟距离转换为实际距离
    cou=cou+dis2*k
cou

x0=np.mean(x34)
y0=np.mean(y34)
print(x0,y0)
cou=0#初始化工作量
for i in range(len(x34)):
    dis2=((x34.iloc[i]-x0)**2+(y34.iloc[i]-y0)**2)**0.5#欧拉距离
    k=30 #虚拟距离转换为实际距离
    cou=cou+dis2*k
```

```
cou
x0=np.mean(x35)
y0=np.mean(y35)
print(x0,y0)
cou=0#初始化工作量
for i in range(len(x35)):
    dis2=((x35.iloc[i]-x0)**2+(y35.iloc[i]-y0)**2)**0.5#欧拉距离
    k=30 #虚拟距离转换为实际距离
    cou=cou+dis2*k
cou

x0=np.mean(x36)
y0=np.mean(y36)
print(x0,y0)
cou=0#初始化工作量
for i in range(len(x36)):
    dis2=((x36.iloc[i]-x0)**2+(y36.iloc[i]-y0)**2)**0.5#欧拉距离
    k=30 #虚拟距离转换为实际距离
    cou=cou+dis2*k
cou

x0=np.mean(x37)
y0=np.mean(y37)
print(x0,y0)
cou=0#初始化工作量
for i in range(len(x37)):
    dis2=((x37.iloc[i]-x0)**2+(y37.iloc[i]-y0)**2)**0.5#欧拉距离
    k=30 #虚拟距离转换为实际距离
    cou=cou+dis2*k
cou

x0=np.mean(x38)
y0=np.mean(y38)
print(x0,y0)
cou=0#初始化工作量
for i in range(len(x38)):
    dis2=((x38.iloc[i]-x0)**2+(y38.iloc[i]-y0)**2)**0.5#欧拉距离
    k=30 #虚拟距离转换为实际距离
    cou=cou+dis2*k
cou
x0=np.mean(x39)
y0=np.mean(y39)
print(x0,y0)
cou=0#初始化工作量
```

```
for i in range(len(x39)):
    dis2=((x39.iloc[i]-x0)**2+(y39.iloc[i]-y0)**2)**0.5#欧拉距离
    k=30 #虚拟距离转换为实际距离
    cou=cou+dis2*k
cou
```

```
% 附录 7：优化模型
function youhua()
%["宽城区","二道区","朝阳区","绿园区","南关区","经开区","长春新区(高新)","净月区","汽开区"]
%路网密度
area=[2.02,1.82,1.85,1.63,1.71,1.73,1.42,1.36,1.70];
%蔬菜发放量
veg=[16100,54933,20100,14144,15000,17569,13392,16689,8815];
%隔离人数
pep=[32.6,42.6,57.8,38.5,48.9,20.3,36.8,22.8,21.7];
%小区数
hom=[168,169,200,184,202,114,85,146,141];
%蔬菜包发放后各个区新增变化率
rate=[0.122,0.082,0.066,0.096,0.077,0.071,0.054,0.045,0.056];
%选取朝阳区，高新区，净月区，汽开区作为最优投放方案方程变量编号为 3，7，8，9
%投放点数量
nums=[181,9,94,470,261,37,215,279,10];
%建立方程
% area[3]*x1+veg[3]*x2+pep[3]*x3+hom[3]*x4=nums[3]
% area[7]*x1+veg[7]*x2+pep[7]*x3+hom[7]*x4=nums[7]
% area[8]*x1+veg[8]*x2+pep[8]*x3+hom[8]*x4=nums[8]
% area[9]*x1+veg[9]*x2+pep[9]*x3+hom[9]*x4=nums[9]
A = [area(3) veg(3) pep(3) hom(3)
    area(7) veg(7) pep(7) hom(7)
    area(8) veg(8) pep(8) hom(8)
    area(9) veg(9) pep(9) hom(9)];
b = [nums(3) nums(7) nums(8) nums(9)]';
[L,U]=lu(A);
x=U\(L\b)
area.*x(1)+veg.*x(2)+pep.*x(3)+hom.*x(4)
end
```