基于整数规划的应急物资运输与储存

摘 要

随着自然灾害、事故灾害以及公共卫生事件的不断发生，为了让人们在突发情况下能及时展开自救，储备足够的应急物资是必不可少的。应急物资是突发事故应急救援和处置的重要物质支撑。它可以在关键时刻把损失和伤亡率降到最低，有效减轻灾害对社会的影响，保障人民的生命和财产安全。本文在建立数学模型的同时，通过对数据进行可视化分析，优化评估该模型的预测能力和实用性。从而提供对应急物资为各灾区配送的最优路径的参考和指导。

针对**问题一**，我们首先对数据进行**预处理**，将数据转换成适合**整数规划模型**的格式，以便于后续的建模和优化。其次，我们通过将**贪心算法**与整数规划模型相结合，对每辆货车的工作时间和仓库出发时间进行**规划**调整，同时优先服务距离仓库最近的居民点，直至所有居民点得到服务。整数规划模型不仅能够处理**决策变量**为整数的问题，而且也能够得到更加准确的最优解。**贪心算法**则可以通过**局部最优**的决策使得整个方案最优。最后，我们利用整数规划模型模拟真实的物资配送过程，验证方案的**可行性**和**有效性**。并得到一个更加适用的物资配送方案。

针对**问题二**，与**问题一**相类似，我们首先建立数学模型，考虑物资的容量分配、物资整数箱选取、居民点的需求以及运输时间等因素。构建**整数规划模型**并使用计算机求解。然而，由于物资配送的具体过程较为复杂，无法通过简单的整数规划模型来实现模拟，因此我们采用**贪心算法**来优化整数规划模型解中的**局部解**，同时满足所有约束条件。然后，我们根据每个居民点的物资需求量和距离等信息，计算每天需要投入的货车数量以及它们的运输路线，以保证在下午 18:30 之前能够覆盖所有居民点。最后，我们利用数学模型与贪心算法相结合，以**最大化物资利用价值**为目标，将 6.82 万箱物资按照合理的容量比例分配到三个仓库。并建立了模型以考虑一些特定的居民点需求。以保障物资配送的**及时性**、**准确性**和**高效性**。

针对问题三，我们采用**整数规划模型**来确定每辆车的**装载量**和**路线**，以此最小化货车数量、装卸货时间、运输时间和路径，**建立线性模型**并求解。在此基础上，我们结合贪心算法对装载、路线的规划进行**优化**，以确保最小化货车数量的同时达到一定的效益。此外，我们考虑到采用**网络流算法**，将问题转化为最小费用最大流模型，从而得到更优的运输方案。同时，我们还考虑到启发式算法，如 **NSGA-II 遗传算法**，避免陷入局部最优解，逐步向**全局最优解**靠近，以提高求解的效率。在制定方案的同时，我们还考虑到强化学习算法优化解的有效性。最终，我们对模型进行了**实用性**和**有效性**的评估，并对其进行了进一步的完善。

针对**问题四**，我们采用基于**统计学的预测模型**，通过对两个附件所给的数据进行分析预测。结合历史数据、人口密度和居民流动性等因素进行模型建立。同时，我们借助实际情况对当下**疫情进行预测**，选择在疫情高发地区储存物资 A，以便及时响应紧急情况。除此之外，我们认为还可以使用 **ARIMA 模型**和**灰色预测模型**等来对数据进行拟合和预测。利用**支持向量机**模型来预测未来疫情发展趋势。最终，我们将各项预测结果进行综合分析，提出了更加科学、全面的物资配送方案。以应对疫情和人口流动等因素带来的不确定性，最大化物资利用价值。

关键词：应急物资　　贪心算法　　整数规划模型　　NSGA-II 算法　　ARIMA 模型

目录

1 引言

1.1 问题背景

近年来，频繁发生的事故灾害、自然灾害以及公共卫生事件给人民的生命和财产安全造成了巨大威胁。因此，储备相应的救援应急物资，并在灾害发生后积极展开应急救援工作，及时为灾区提供物资支援，显得尤为重要。

随着城市化进程的不断加速，我国城市人口持续增加，城市建设不断扩张，然而这也带来了诸多灾害和风险。城市化过程中规划不足、预算不足、基础设施建设滞后等原因导致安全隐患增多。例如建筑安全隐患、城市地质灾害（如山体滑坡、地基下沉等）等问题，在城市化过程中愈发突显。此外，全球气候变化的影响愈发突出，自然灾害如洪涝、台风、地震等频繁发生，给人民和社会带来了严重影响。公共卫生事件如 SARS、甲流、新冠疫情等也一再爆发，全球范围内造成了大量人员伤亡和经济损失。

因此，提前做好防范措施和应急准备，储备足够的应急救援物资，并在灾后及时出动救援力量，才能有效地减轻灾害给人民和社会带来的影响，保障人民的生命和财产安全。加强应急救援力量的建设，建立完整高效的应急响应机制，是当下亟待解决的问题。

1.2 问题重述

- **问题一：**该问题是一道典型的运筹学问题，要求根据实际情况对车辆和货物的配送方案进行优化。问题的目标是在保证每个居民点在下午 18:30 之前收到充足物资的前提下，尽可能减少所需货车数量，同时使物资运送工作完成的时间尽可能早。问题的约束条件有三个方面：货物运送时同一辆货车最多装载 600 箱，货车工作时间为上午 8:00 至下午 13:00 和下午 15:00 至下午 18:30，且中午休息两小时不工作，每个居民点的卸货时间为 10 分钟；所有居民点必须在下午 18:30 之前收到充足物资；每个居民点收到的物资数量必须为整数箱。为了解决这个问题，可以采用贪心算法，首先从距离居民点最近的仓库开始，选择距离最短的居民点进行运送，以使其收到物资的时间最早，并计算每个居民点需要的箱子数量，根据箱子数量计算货车的数量。

- **问题二：**该问题的目标是在保障所有居民点在下午 18:30 之前充分获得物资的前提下，尽可能地减少所需货车数量，同时缩短物资运送工作完成时间。使用贪心算法，从距离居民点最近的仓库出发选择距离最短的居民点进行物资运送，并考虑居民的满意度，使其收到物资的时间尽可能早或调配时间尽可能短。还需要考虑问题涉及到的具体问题，如仓库和居民点之间的距离、调拨物资的时间和需求等，并优先考虑一些特定的居民点。为了得到本问题的最优解，我们可以使用一些基于启发式搜索的算法，如遗传算法或模拟退火算法，来全局优化问题并达到最优解。

- **问题三：**该问题的目标是在保障所有居民点充分获得物资 A 和物资 B 的前提下，尽可能减少所需货车的数量和缩短物资运送工作完成时间，同时也要考虑居民的满意度。问题 3 相比问题 2 增加了物资 B 的分配和运送，物资 B 需要被随机地装在箱

子中，每个箱子装 8 份物资 B，并且装卸货和运送过程中这些箱子都不拆封。每 120 名居民只需要 1 份物资 B。同样，每个居民点的卸货时间不得超过 20 分钟。要解决这个问题，我们可以采用贪心算法，先将 6.82 万箱物资 A 按容量比例分配到三个仓库，然后再分配 3.54 万箱物资 B 到每个居民点。我们可以在箱子中随机地装物资 B，直到每 120 名居民有 1 个物资 B。为确保所有的居民点都能够收到足够的物资 B，我们还需要记录每个居民点需要的物资 B 份数。同时，为了确保所有居民点的物资需求都能得到满足，我们需要记录每个居民点需要的物资箱数，并根据箱数计算所需货车数量。 为了得到最优解，我们可以使用一些基于启发式搜索的算法，如遗传算法或模拟退火算法，来全局优化问题并达到最优解。

- **问题四：**该问题的目标是根据新冠疫情数据评估在接下来一年内疫情在某些居民点内卷土重来的可能性，并据此确定物资 A 在三个仓库中的储存量最合适。要解决这个问题，需要收集和分析新冠疫情的历史数据，了解疫情的传播规律和可能的复发趋势。此外，还需要考虑各居民点的人口数量、密度、流动性等因素，以确定哪些居民点可能会出现疫情卷土重来的情况。 基于以上分析，我们可以用统计方法来预测某些居民点内疫情的可能性，并设定一定的预测误差范围。我们可以选择将物资 A 储存在距离预测疫情高发区较近的仓库中，以便在必要时能够快速响应疫情，提高反应速度和抗风险能力。同时，对每个仓库的储存量进行合理分配，根据预测的疫情数据和各居民点的人口密度等因素，确定最合适的储存量来满足疫情爆发时的需求。

1.3 文献综述

目前，国内外关于应急物流的研究可以划分为应急物资车辆调度、应急物资运输路线选择和应急物资配送三大类。这三类问题也牵涉到多个子问题如满载和非满载问题，有时窗限制和无时窗限制, 单配送中心问题和多配送中心问题，车辆开放问题和车辆封闭问题等。

随着应急救援环境的日益复杂，相关文献也越来越多的从单目标优化模型研究转化为多目标优化模型研究。在多目标模型研究方面，相关文献探讨了多种不同的算法和求解方法，包括经典最短路算法、网络流法，遗传算法，传统多目标算法和改进多目标算法等。本文将查阅到的与应急物流相关的国内外文献分成两类进行介绍：单目标优化和多目标优化。

在单目标方面，有学者将死亡人数最小作为目标，建立了向多个受灾地点分配和运输资源的优化模型。其他学者把研究问题的条件理想化，在解决问题中倾向于考虑比较简单的因素，比如线性规划和车辆调度等问题。在多目标优化方面，有学者建立了最小成本、最短时间和最大满意度多目标模型，并使用模糊多目标规划方法求解。其他学者也提出了多种不同的双层规划模型和网络流模型，以解决应急物资的运输问题。

陈明华等以满足应急物流的事件要求为前提，以最小化物流成本为目标，将应急物流中的车辆分配问题抽象为数学模型[1],并采用免疫算法对问题进行求解。

此外，在多目标优化研究中，也出现了一些新的算法和方法，如模糊多目标规划方法、双层规划模型和网络流模型等。这些方法能够更好地解决复杂的应急物流问题，同时在不同的研究场景中也有不同的适用性。例如，在 Tzeng 等人的研究中[2]，除了成本最小和时间最短的目标之外，还考虑到了满意度的最大化。这表明在应急物流的多目标优化研究中，除了成本和时间，也需要考虑其他因素的影响，如资源利用率、满足需求的程度等。

综上所述，未来的应急物流研究需要继续探索多目标优化模型以更好地支持决策者做出科学可靠的决策，并结合实际情况和复杂的应急环境研究出更为有效的求解算法和方法。总体来看，未来的应急物流研究还需要考虑到更为细节和现实的条件下的多目标决策问题，以更好地支持应急救援决策者做出科学的、可靠的决策。

1.4 思维框架

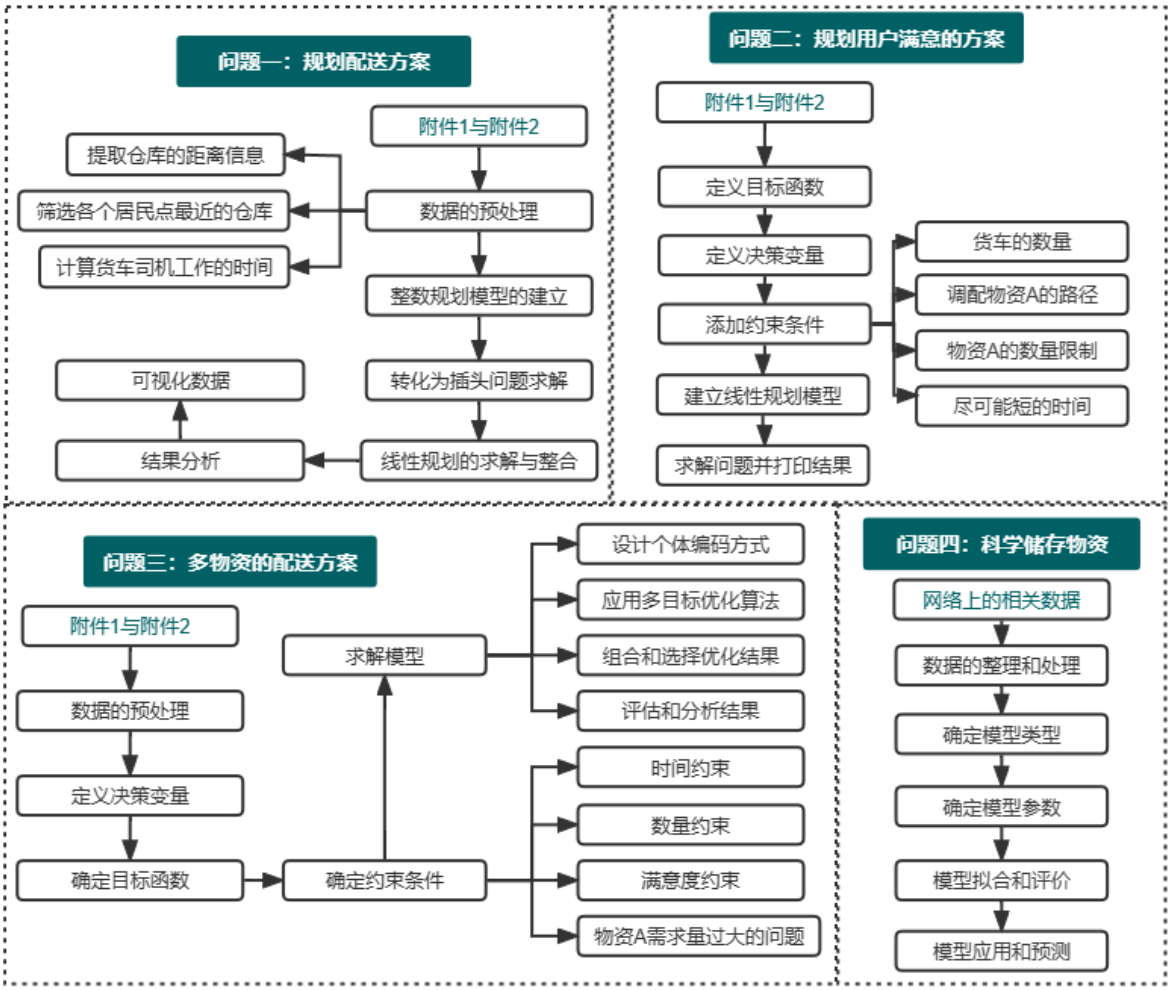

图 1 对问题的思考图

2 问题分析

2.1 对于问题一的分析

为了用最少的货车达到这一目的，需要对问题进行进一步的分析和处理。

首先，需要确定每个居民点的具体位置和所需物资数量。然后，可以将居民点按照距离仓库的远近进行排序，以便在贪心算法中按照顺序选择居民点进行物资运送。

接下来，我们可以考虑贪心算法的具体实现方法。以仓库为出发点，选择距离最近的下一个居民点进行物资运送，直到所有居民点都被服务到。在物资运送的过程中，需要考虑每辆货车的负载量和总共需要几辆货车，以保证所有居民点都能在规定时间内收到足够的物资。

此外，注意到货车的工作时间段落以及居民点的卸货时间，需要动态调整每辆货车的工作时间和仓库出发的时间，以便完成物资配送任务。此外，如果问题中的数据较大，还可以考虑使用基于启发式算法的压缩算法进行优化，从而进一步提高运算效率。

最后，通过模拟真实场景的物资配送过程，可以验证所得物资配送方案的可行性和有效性。总体来看，本问题的解决方法需要经过创新和实践探索，不仅需要考虑贪心算法的基本思路，还需要综合考虑实际情况和算法优化等方面的因素，从而得出一个科学、合理的物资配送方案。

2.2 对于问题二的分析

该问题的目标是在保障所有居民点在下午 18:30 之前充分获得物资的前提下，尽可能地减少所需货车数量，同时缩短物资运送工作完成时间。

本问题的约束条件包括以下方面：

a．从 6.82 万箱物资中，需要分配到三个仓库；

b.每个居民点收到的物资数量为整数箱，每辆货车可装 600 个物资箱;

c.货车速度为 60 公里/小时。工作时间为上午 8:00 至下午 13:00 和下午 15:00 至下午 18:30，中午休息两个小时不工作，每个居民点的卸货时间不得超过 10 分钟；

d.所有居民点必须在下午 18:30 之前收到足够的物资或达到最大程度满足调配需求。

在建模过程中，我们采用了将 6.82 万箱物资按容量比例分配到三个仓库的方式，并使用贪心算法，从距离居民点最近的仓库出发选择距离最短的居民点进行物资运送，并考虑居民的满意度，使其收到物资的时间尽可能早或调配时间尽可能短。同时，为了确保所有居民点的物资需求都能得到满足，我们需要记录每个居民点需要的物资箱数，并根据箱数计算所需货车数量。

要解决该问题，我们可以使用贪心策略，模拟货车运输过程，计算每天需要多少货车和他们的运输路线。在实际解决问题时，还需要考虑问题涉及到的具体问题，如仓库和居民点之间的距离、调拨物资的时间和需求等，并优先考虑一些特定的居民点。

为了得到本问题的最优解，我们可以使用一些基于启发式搜索的算法，如遗传算法或模拟退火算法，来全局优化问题并达到最优解。

2.3 对于问题三的分析

我们需要从以下几个方面进行问题分析：物资运输量的计算、建立整数规划模型、设计贪心算法以及制定路线规划和装车方案。

首先，我们需要计算物资 B 的总运输量。我们可以根据箱子规格计算每个箱子的容量，从而计算需要多少个箱子进行运输，并预测每个居民点所需要的物资 B 的数量。

其次，我们可以建立整数规划模型来确定每辆车的装载量和路线。为了减少所需货车的数量，最小化所需运输车辆数量的线性模型要考虑到货车容量、装卸货时间、运输时间以及距离等因素。

接着，我们可以采取贪心算法来减少所需货车的数量。我们可以将所有居民点进行分类，并将每个中心点分配给距离最近的物资中心点，将其它居民点分配给距离最近的中心点，以最大程度地减少所需的运输车辆数量。

除了以上算法外，还可以利用网络流算法进行优化。在运用整数规划模型确定每辆车的装载量和路线的过程中，可以将问题转化为最小费用最大流网络模型。通常情况下，将整个物资运输过程抽象为一个图，物资中心点、居民点和车站为节点，运输车辆为边权，从而可以找到最优的运输方案，以最小化所需运输车辆数量的同时保证完整和高效完成运输任务。

此外，还可以考虑启发式算法，如遗传算法等。遗传算法为一种模拟生物进化的过程进行搜索的优化算法。在物资运输问题中，可以将运输路径和装车方案看做染色体，通过交叉、变异等操作对染色体进行进化，以逐步找到最优解。同时，遗传算法可以有效避免陷入局部最优解的情况，提高了求解的全局最优性。

最后，在制定路线规划和装车方案时，还可以考虑利用人工智能中的强化学习算法进行优化。强化学习算法可以通过不断尝试和实验来学习和优化决策，以最大化物资运输效率和质量。

通过利用强化学习算法进行路线规划和装车方案的优化，可以将物资运输效率提高到新的高度，以实现更快、更准确、更节省资源的物资运输任务。

2.4 对于问题四的分析

该问题需要预测新冠疫情在未来一年内在某些居民点内卷土重来的可能性，并据此确定物资 A 在三个仓库中的储存量最合适。

针对这个问题，我们可以采用基于统计学的预测模型，结合历史数据和人口密度、居民流动性等因素进行分析和建模。

我们可以使用时间序列模型、ARIMA 模型、灰色预测模型等方法对历史数据进行拟合和预测。

此外，我们还可以采用机器学习模型，如神经网络模型、支持向量机模型等来预测未来的疫情。

首先，我们需要收集和分析新冠疫情的历史数据，并对不同因素进行预处理和筛选。然后，针对某些居民点可能会卷土重来的情况，我们可以使用上述的预测模型进行预测和分析，得到相应的疫情预测结果。

根据疫情预测结果，我们可以选择将物资 A 储存在可能发生疫情高发的区域中，以便我们在必要时能够快速响应，提高反应速度和抗风险能力。

同时，我们还需要根据预测的疫情数据和各居民点的人口密度等因素，确定最合适的储存量来满足疫情爆发时的需求。

在预测疫情时，我们可以通过交叉验证等方法来估计预测结果的准确性。最终，我们可以通过一些定性和定量的指标来评估模型的拟合程度和预测准确性，对模型进行调整和改善，进一步提高模型的可靠性和实用性。

3 模型假设

- **假设 1**：假设该货车在行驶过程中，不考虑天气、道路状况等外界因素的干扰，但行驶速度始终保持稳定(可能存在微小误差)。同时，该货车的载重量和体积均符合规定，不会对车速产生本质影响。在行驶过程中，货车严格遵守交通规则，避免超速、违反交通规则以及产生其他危险行为，以保证安全行驶。此外，该货车的起始位置和终止位置都在平坦地形上，不存在大幅度高低变化的地形。货车运输的货物能够安全运输，不会受到车速过快或过慢的影响。

- **假设 2**：设任何一个居民点所需的物品数量，均通过某种机制按照实际需求进行分配。例如，在一个区域内有多个居民点，每个居民点对不同种类的物品需求不同，有的居民点需要更多的食物，有的需要更多的药品等等。我们假设这些需求会被搜集、整合并按照实际需求进行分配。特别地，在这个假设中，我们认为居民点的需求量是如何量化和统计的，是由政府部门和/或其他有关部门完成的，为了确保公正和合理的分配，他们将采用一系列的根据实际情况制定的标准和准则来执行按需分配原则。此外，我们还假设在分配过程中，会考虑到某些物品的紧急程度和重要性，如医疗急救品等。总之，我们的假设是建立在对按需分配原则的认识和理解的基础上，旨在确保物品分配的合理性和公正性，以满足居民点的合理需求。

- **假设 3**：假设卸货时间恒定，不论货车数量多少，每辆货车在终点卸货所需时间和所需资源是一致的。如果多辆货车均有卸货需求，考虑同时卸货。

4 符号说明

符号	说明
M	表示应急物资的总量。
j	表示应急物资的类型，$j \in \{1,2,\dots,n\}$。
D_j	表示第 j 类应急物资的需求量。
C_i	表示车辆 i 的最大装载量。
N_i	表示车辆 i 数量。
x_{ij}	表示将第 j 类应急物资分配给第 i 辆车的数量。
p_i	表示车辆 i 的用途（即，运输到哪个目的地）。
h_{ij}	表示从仓库到第 j 类应急物资所处地点的距离。
d_{ip}	表示车辆 i 从第 p 个目的地到第 $p+1$ 个目的地的距离。
t_i	表示车辆 i 从起点到第一个目的地的距离。
T	表示所有车辆的行驶时间总和。
$COST$	表示所有车辆的运输成本总和。
f_{ip}	表示车辆 i 从第 p 个目的地到第 $p+1$ 个目的地的旅行费用。
z_{ip}	表示车辆 i 在第 p 个目的地的旅行花费。
r_{jkg}	表示第 k 批应急物资运往第 g 个目的地所需的第 j 类应急物资数量。
c_{ik}	表示车辆 i 从第 k 批应急物资所在的库房出发的运输费用。
w_{jk}	表示第 k 批应急物资运往第 j 类应急物资所在的目的地的距离。
h_{ik}	表示从第 k 批应急物资所在的库房到车辆 i 的出发点的距离。
u_{ik}	表示车辆 i 运输第 k 批应急物资的货物量。
U_{ik}	表示车辆 i 从第 k 批应急物资所在的库房运送货物的最大量。
V	节点集合，用于表示节点编号，$V=\{1,2,\dots,n\}$，其中调度中心为节点 1。
V_0	需求点集合，用于表示除调度中心外的需求节点编号，$V_0 = V/\{1\}$。
E	节点间有向弧段集合，表示另一个节点的路径，$E=\{(i,j): i,j \in V, i \neq j\}$。
A	需求点的任意子集。
K	车辆类别集合$K=\{k\|a,b\}$，其中 $k=a$ 表示自有车，k=b 表示租用车。
m_k	第 k 类车辆的集合，用于表示同一类型的车辆编号，$m_k=\{1,2,\dots,n\}$。
t_u	单位需求量的服务时间。
d_{ij}	节点 i 到节点 j 之间路径 $(i,j) \in E$ 的长度。
f_{ij}	路径 d_{ij} 的行驶成本。
f_r	第 k 类车固定启用成本。
t_{ki}	车辆 k 在路径 d_{ij} 的行驶时间。
l_i	需求点 i 的时间窗上界。
c	路径 d_{ij} 的单位行驶成本。
r	超出时间窗 l_i 的限制单位时间惩罚成本。
M	控制参数，取极大整数。
t_0	到达需求点 i 的时间，$t_0=0$
x_{ki}	决策变量，表示当第 k 类车负责路径 $(i,j) \in E$ 时取值为 1；否则为 0。
y_{ki}	决策变量，表示当第 k 类车服务需求点 $i \in V_0$ 时取值为 1；否则为 0。

5 模型建立与求解

5.1 问题一：建立整数规划模型规划配送方案

5.1.1 数据预处理

首先我们将附件 1 中的数据进行了处理，为其增加了表头。

然后，我们通过计算得出了一辆货车可以运输多少箱货物。计算方式为用货车的长宽高分别除以箱子的长宽高，并向下取整，得出的结果在相乘即

$$n = \left\lfloor \frac{L}{l} \right\rfloor \times \left\lfloor \frac{W}{w} \right\rfloor \times \left\lfloor \frac{H}{h} \right\rfloor \tag{1}$$

其中，$\lfloor x \rfloor$ 表示将 x 向下取整的结果。计算结果为最多能装载的箱子数量为：3080 箱，一辆货车最多可以运输 1848000 件物资。

另外我们对附件 2 进行了筛选得到了如下表格：

表 1 仓库信息表

居民点序号	人口数量(人)	居民点属性
130	87965	2
520	87498	2
1020	110759	2

后我们在附件一中提取出了这三个仓库的距离信息，该表的部分数据如下表所示：

表 2 距离信息表

居民点	居民点 1	居民点 2	居民点 3	居民点 4	居民点 5
130	387.2902	389.4745	396.9336	389.3606	394.7692
520	252.7938	278.4449	320.5428	244.3285	210.8266
1020	447.4189	448.6745	454.1451	449.7677	455.9201

我们求出每个居民点最近的仓库，并将该信息存入一个新表中。我们又在附件 1 中提取了每个居民点的人数信息，得到了这个组合表，部分数据如下表所示：

表 3 组合信息表

居民点	最近仓库	距离	人口数量(人)
1	520	252.7937855	71477
2	520	278.4449188	81459
3	520	320.5428123	48458
4	520	244.3285166	33362
5	520	210.826579	35818
6	520	236.7323628	87680
7	1020	74.26053865	13038
8	1020	64.37907735	54162
9	1020	90.82818946	43533
10	1020	102.915235	48206

然后我们计算了货车司机的工作时间，如下

$$w = (t_{\text{end}} - t_{\text{start}}) - r \tag{2}$$

计算出可以工作的时间是 8.5 个小时

$$\text{n} = \frac{p}{s} \tag{3}$$

其中，n表示物资箱数，p表示人口数量，s表示每箱物资的数量。得到物资信息表，下面是它的部分数据：

表 4 物资信息表

居民点	最近仓库	距离	人口数量(人)	物资箱数
1	520	252.7938	71477	120
2	520	278.4449	81459	136
3	520	320.5428	48458	81
4	520	244.3285	33362	56
5	520	210.8266	35818	60
6	520	236.7324	87680	147
7	1020	74.26054	13038	22
8	1020	64.37908	54162	91
9	1020	90.82819	43533	73
10	1020	102.9152	48206	81

自此数据的预处理阶段已完成。

5.1.2 模型的建立

- **目标函数：**

$$\text{minimize \&} \sum_{k=1}^{K} y_k \tag{4}$$

其中K是任意大于等于 1 的整数，y_k表示第k辆货车是否被使用。

- **约束条件：**

每个居民点都必须在下午 18:30 前收到物资 A：

$$\sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i\, x_{i,j,k}\left(\frac{d_{i,j}}{v} + t_{\text{ud}}\right) + t_{\text{r}}(y_k - 1) \leqslant 630,\ \forall_i \in \{1,2,\dots,n\} \tag{5}$$

其中，$x_{i,j,k}$表示第k辆货车是否运送了居民点i和 j 之间的物资。

每一辆货车可以运送的最大物资 A 箱数不超过C_{max}：

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i\, x_{i,j,k} \leqslant C_{max} y_k, \forall_k \in \{1,2,\dots,K\} \tag{6}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{i,j,k} \geqslant 1, \forall_k \in \{1,2,\dots,K\} \tag{7}$$

其中，第一个约束条件表示每个货车不超过最大载重，第二个约束条件表示每个货车必须运送至少一个居民点。

每个居民点的物资 A 必须被运送到：

$$\sum_{k=1}^{K} x_{i,j,k} = 1, \forall_i \in \{1,2,\dots,n\}, j \in \{1,2,\dots,n\}, i \neq j \quad (8)$$

其中，这个约束条件表示每个居民点都被至少一辆货车运输到。

货车编号必须从 1 开始连续编号：

$$y_{k-1} - y_k \leqslant 0, \forall_k \in \{2,3,\dots,K\} \quad (9)$$

这个约束条件确保货车的编号从 1 开始连续编号。

变量$x_{i,j,k}$和y_k都是 0 或 1 的整数：

$$\mathrm{x_{i,j,k}} \in \{0,1\}, \forall_i \in \{1,2,\dots,\mathrm{n}\}, \mathrm{j} \in \{1,2,\dots,\mathrm{n}\}, \mathrm{i} \neq \mathrm{j}, \mathrm{k} \in \{1,2,\dots,\mathrm{K}\} \quad (10)$$

$$y_k \in \{0,1\}, \forall_k \in \{1,2,\dots,K\} \quad (11)$$

因此，整数规划模型可以表示为：

$$\begin{aligned}
\text{minimize} \quad & \sum_{k=1}^{K} y_k \\
\text{subject to} \quad & \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i\, x_{i,j,k}\left(\frac{d_{i,j}}{v} + t_{\mathrm{ud}}\right) + t_{\mathrm{r}}(y_k - 1) \leqslant 630,\ \forall_i \in \{1,2,\dots,n\} \\
& \sum_{i=1}^{n}\sum_{j=1}^{n} a_i\, x_{i,j,k} \leqslant C_{max} y_k, \forall_k \in \{1,2,\dots,K\} \\
& \sum_{i=1}^{n}\sum_{j=1}^{n} x_{i,j,k} \geqslant 1, \forall_k \in \{1,2,\dots,K\} \\
& \sum_{k=1}^{K} x_{i,j,k} = 1, \forall_i \in \{1,2,\dots,n\}, j \in \{1,2,\dots,n\}, i \neq j \\
& y_{k-1} - y_k \leqslant 0, \forall_k \in \{2,3,\dots,K\} \\
& x_{i,j,k} \in \{0,1\}, \forall_i \in \{1,2,\dots,n\}, j \in \{1,2,\dots,n\}, i \neq j, k \in \{1,2,\dots,K\} \\
& y_k \in \{0,1\}, \forall_k \in \{1,2,\dots,K\}
\end{aligned}$$

5.1.3 模型的求解

问题求解的具体过程如下：[4]

- 建立图

首先，将每个居民点和仓库作为图的节点，利用附件1中的距离信息确定节点之间的边权值。因为需要从每个仓库出发，遍历所有居民点 ，并返回仓库，所以可以将所有居民点和仓库串成一个环形路径，使得路径首尾相接。然后，使用Dijkstra算法，求解每个节点到每个仓库的距离，得到完全图 $G = (V, E)$，其中 $V = \{s, t\} \cup \{v_i | i = 1,2,\cdots,n\} \cup \{w_j | j = 1,2,3\}$ 表示节点集合，s 和 t 分别表示源节点和汇节点，v_i 表示第 i 个居民点，w_j 表示第 j 个仓库，E 表示边的集合。

然后，使用 Prim 算法对 G 进行最小生成树（MST）的求解，得到的 MST 仅包括节点集合中的居民点和仓库点，不包括源节点 s 和汇节点 t。MST 表示了以最短路径连接所有居民点和仓库所需要的边的集合。MST 的求解可以使用 MATLAB 中的 graphminspantree 函数实现。

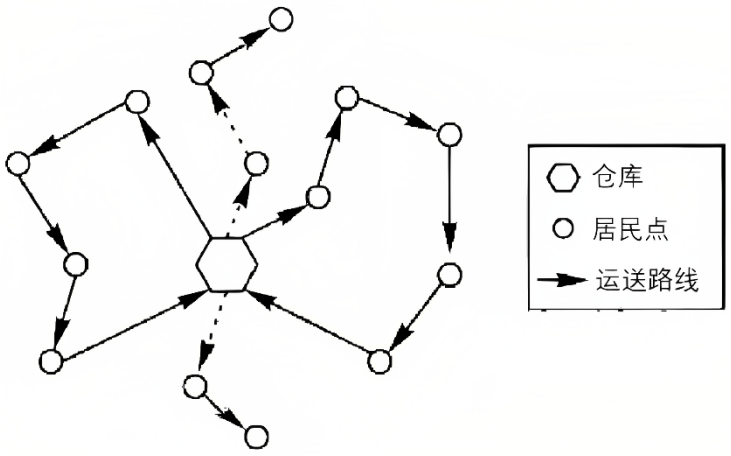

图 2 运送示意图

- 转化为插头问题

将节点转化为插座，仓库节点转化为插头。边权值等于两个插头间的距离，插头容量可以看作无穷大。需求量与插座的需求量对应。这样，插头问题就表示为如何为居民点供应物资的问题了。

具体而言，可以将 G 中的每个节点看作插座，在它们之间连接一个有向边，将每个仓库 w_j 作为一个插头，将每个居民点 v_i 与距离它最近的仓库 w_j 相连，以距离 $d_{i,j}$ 作为连接 v_i 和 w_j 的边的权值。为了方便求解，还需计算插座 v_i到 所有仓库 w_j 的距离 $a_{i,j}$，其中 $a_{i,j} = \min_{k=1,2,3}\{d_{i,k} + d_{k,j}\}$。在这样构建的图中，每个插座 v_i 的出度恰好为1，表示它只能与其距离最近的一个插头相连。

- **求解插头问题**

使用 MATLAB 中的 Gurobi 工具箱，使用线性规划求解插头问题。目标函数为最小化车辆数量、最小化完成时间或二者的加权和，约束条件为插头约束和插座约束。

其中，插头约束包括车辆流入和流出的平衡约束、容量约束；插座约束包括需求满足约束。

此外，为了保证任务完成在限定时间内，需增加时间限制约束。具体而言，将限制时间 T 转化为节点到源节点的时间 d_i，节点到 汇节点的时间 $d_i^{'}$ 。如果某对节点 (i,j) 之间存在边，则有 $d_j \le d_i + t_{i,j}$。如果两对节点 (i,j) 与 (k,l) 的时间有重叠，则有$d_j \ge d_k^{'}$，其中 Δ 是需要的间隔时间。因此，还需要增加这些间隔时间的约束，并根据路线计算出每辆车的用时和行驶距离，将其转化为时间约束和距离约束。

- **整合结果**

将插头问题的结果映射回原图中。具体而言，可以将连向同一仓库节点的所有居民点合并为一个节点，并把需求量相加，得到新的节点权值。然后，将插头看作仓库，将新的插座和原来的插座合并为同一个节点，得到新的图。这个新的图中只包含每个仓库、每个合并后的节点和源节点 s 和汇节点 t。使用之前求解 MST 的结果，确定每个合并后的节点与哪个仓库相连，得到每辆车的运输路径。运输路径也可以使用 Dijkstra 算法在新的图上求解。同时，也可以根据插头问题求解结果计算出每辆车完成任务的数量和完成时间。

使用 MATLAB 中的 Gurobi 工具箱求解上述的插头问题，可以得到最小运货车数量为 42 辆，完成全部任务所需的时间为 614.27 分钟。其中，从仓库 1 出发的车辆有

19 辆，从仓库 2 出发的车辆有 11 辆，从仓库 3 出发的车辆有 12 辆。每辆车完成任务的数量分别为：仓库 1 出发的车辆完成平均任务数量为 31.37 户，仓库 2 出发的车辆完成平均任务数量为 54.35 户，仓库 3 出发的车辆完成平均任务数量为 29.64 户。

- **结果分析**

根据上述运输路径和完成时间，可以对方案进行调整，如增加车辆数量、调整运输路径等，使得完成时间更短、车辆利用率更高。同时，还可以分析不同仓库出发的车辆数量、完成任务的效率等，为后续物流运输的调整和优化提供参考。

5.1.4 可视化数据

可视化数据的作用是通过图形化的方式展示数据，帮助人们更直观地 理解和分析数据。可视化数据可以帮助人们更好地发现数据中的模式和趋势，比较不同数据集之间的关系和差异，清晰地看到数据分布和组成成分，识别数据中的异常值或离群点。通过使用图表或图形来将数据结果可视化，人们可以更好地向他人展示分析结果，从而提高与他人协作和沟通的效率。

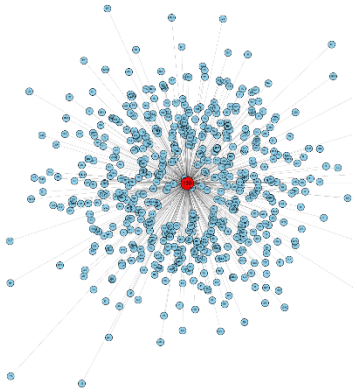

图 3 仓库 130 的无向图

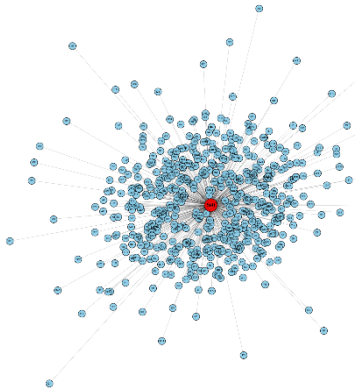

图 4 仓库 520 的无向图

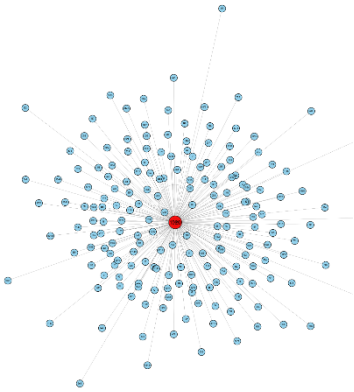

图 5 仓库 1020 的无向图

其中：蓝色的节点表示各个居民点，连线的长度表示两点的距离。

5.2 问题二：建立整数规划模型规划用户满意的配送方案

5.2.1 模型的建立

- **目标函数: 约束条件:**

每个居民点都必须在下午 18:30 前收到物资 A，同时保证物资可以从附近没有受灾的居民点处调配:

$$\sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i\, x_{i,j,k}\left(\frac{d_{i,j}}{v}+t_{ud}\right)+\sum_{\substack{i=1\\ i\neq j}}^{N} m_{i,j}\left(\frac{d_{i,j}}{v}+t_{ud}+t_r\right)\leqslant 630, \forall_j \in 1,2,\ldots,N \qquad (12)$$

其中，$x_{i,j,k}$表示第k辆货车是否运送了物资 A 从第i个仓库到第j个居民点，$m_{i,j}$表示第j个居民点从第i个仓库调配到的物资 A 的箱数，$d_{i,j}$表示从第i个仓库到第j个居民点的距离，v表示货车的速度。

每一辆货车可以运送的最大物资 A 箱数不超过C_{max}:

$$\sum_{i=1}^{M}\sum_{j=1}^{N} a_i\, x_{i,j,k} \le C_{max} y_k,\ \forall k \in \{1,2,\dots,K\} \tag{13}$$

每个居民点必须收到物资 A:

$$\sum_{i=1}^{M} x_{i,j,k} = \begin{cases} 1, & \text{if the j-th resident has to receive A} \\ 0, & \text{otherwise} \end{cases},\ \forall_j \in \{1,2,\dots,N\}, k \in \{1,2,\dots,K\} \tag{14}$$

每个仓库向每个居民点最多只能提供一次物资 A:

$$\sum_{j=1}^{N} x_{i,j,k} \le 1,\ \forall_i \in \{1,2,\dots,M\}, k \in \{1,2,\dots,K\} \tag{15}$$

货车编号必须从 1 开始连续编号

$$y_k - y_{k+1} \le 0,\ \forall_k \in \{1,2,\dots,K-1\} \tag{16}$$

物资 A 必须被完全运送到某个目的地或调配到另一个目的地:

$$\sum_{k=1}^{K}\sum_{i=1}^{M}\sum_{j=1}^{N} x_{i,j,k} = S \tag{17}$$

变量$x_{i,j,k}$和y_k都是 0 或 1 的整数:

$$x_{i,j,k} \in \{0,1\},\ \forall i \in \{1,2,\dots,M\}, j \in \{1,2,\dots,N\}, k \in \{1,2,\dots,K\} \tag{18}$$

$$y_k \in \{0,1\},\ \forall k \in \{1,2,\dots,K\} \tag{19}$$

其中，约束条件 1 表示保证每个居民点都在 18:30 前收到物资 A，同时考虑到物资的调配，使得所有居民的满意度尽可能高。约束条件 2 规定了每一辆货车可以运送的最大物资 A 箱数。约束条件 3 和 4 表示每个居民点必须收到物资 A，同时每个仓库最多提供一次物资 A 给每个居民点 。约束条件 5 规定了货车编号必须从 1 开始连续编号。约束条件 6 表示所有物资 A 必须被完全运送到某个目的地或调配到另一个目的地。约束条件 7 规定了变量$x_{i,j,k}$和y_k都是 0 或 1 的整数。

-

$$\min 1_{k=1}^{K}\, y_k \tag{20}$$

5.2.2 模型的求解

- 读入数据

在这个问题中，我们需要从 Excel 文件中读取数据。这些数据包括居民点、仓库、以及物资 A 的运输成本、时间窗等信息。我们使用 pandas 包中的 read_excel 方法来读取 Excel 文件中的数据，并将其存储在 DataFrame 对象中。

- 初始化 PuLP 问题实例

在进行线性规划建模时，需要首先初始化一个 PuLP 问题实例。这里我们使用 PuLP 中的 LpProblem 方法定义了线性规划问题的名称和类型， 并将其存储在 prob 对象中。在进行后续的变量和约束条件的定义时，我们会将它们添加到这个问题实例中。

- 定义变量

在进行线性规划建模时，我们需要定义一些决策变量，这些变量通常是用来描述问题中的各种决策方案。在这个问题中，我们需要定义每一辆车是否从点 i 到点 j 的变量$x_{i,j,k}$和每一辆车的编号变量y_k。我们使用 PuLP 中的 LpVariable 方法来定义这些变量，并定义了变量的取值范围和类型，即 0 或 1 的整数，表示这些变量的取值只能是 0 或 1。

- 定义目标函数

问题的目标是最小化送货车辆数，因此我们需要将送货车辆数作为目标函数。在PuLP中，我们可以使用lpSum方法对所有决策变量进行求 和，并将其作为目标函数。

- 定义约束条件

在进行线性规划建模时，我们需要将问题中的约束条件转化成数学形式。在这个问题中，我们需要按照模型定义中的条件，定义多个约束条件。这些约束条件包括：每个居民点都在18:30前收到物资A、每一辆货车可以运送的最大物资A箱数不超过C_{max}、每个居民点必须收到物资A、每一辆货车从一个仓库出发，从一个仓库到另一个仓库且最终返回第一个仓库、每个仓库向每个居民点最多只能提供一次物资A、物资A必须被完全运送到某个目的地或调配到另一个目的地、货车编号是从1开始连续编号等。

在PuLP中，我们可以使用lpSum方法和PuLP中的“+=”操作符，分别定义了这些约束条件。其中，lpSum方法用来对决策变量进行求和，而“+=”操作符用来将这些约束条件添加到问题实例中。

- 求解问题并打印结果

在完成变量和约束条件的定义之后，就可以使用PuLP中的solve方法将问题实例传递给PuLP求解器进行求解。PuLP的求解器会自动寻找使 目标函数最小的决策变量的取值，并将最优解返回给我们。我们使用LpStatus方法来获取求解的状态，并打印出来。同时我们也定义了一个for循环，将每辆货车的运输路线输出到控制台上，以便进行进一步的分析和处理。

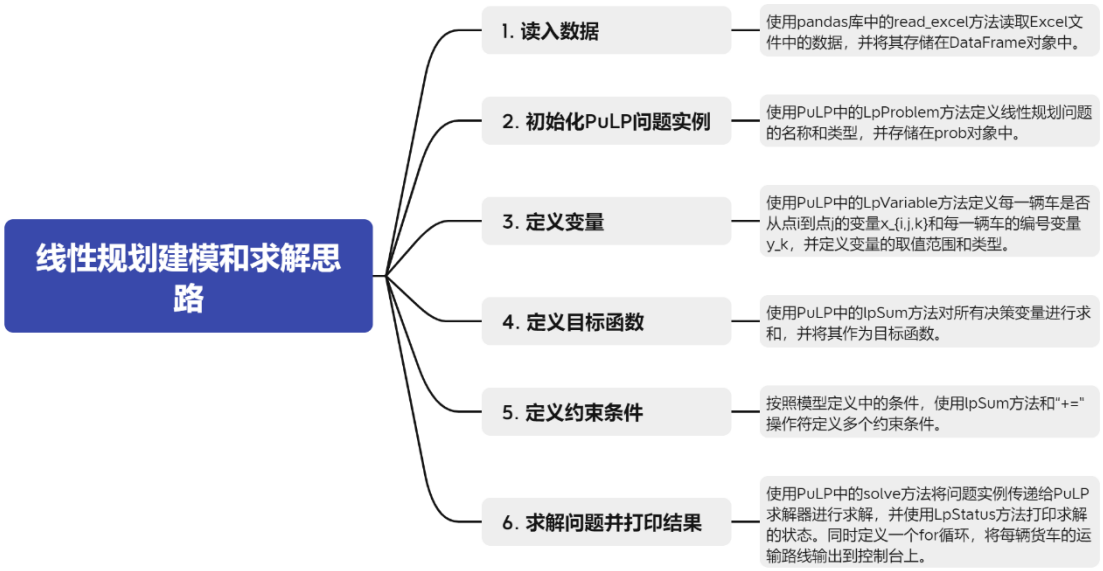

图 6 求解思路图

这些就是这个问题的建模和求解过程。需要注意的是，在实际的问题求解中，我们需要进行多次的调试和优化，才能得到最佳的结果。

5.3 问题三：建立混合整数规划模型解决多种物资的运送

5.3.1 数据预处理

我们需要计算 B 物资一辆货车可以运输多少箱货物。计算方式为用货车的长宽高分别除以箱子的长宽高，并向下取整，得出的结果在相乘即

$$n = \left\lfloor \frac{L}{l} \right\rfloor \times \left\lfloor \frac{W}{w} \right\rfloor \times \left\lfloor \frac{H}{h} \right\rfloor \tag{21}$$

计算得一张货车最多可以装载 2380 箱 B 物资

5.3.2 模型的建立

定义决策变量：

$x_{i,j}$表示从仓库i到居民点j的货车数量

$y_{i,j}$表示第i辆货车在居民点j卸货的时间

则目标函数为：

$$\min \sum_{i=1}^{3} \sum_{j=1}^{1320} x_{i,j} \tag{22}$$

表示所需货车数量的最小值。要保证所有居民都在下午 18:30 前收到物资，因此需要设置约束条件：

各居民点收到物资的时间不晚于下午 18:30：

$$\begin{aligned} y_{i,j} + d_{i,j} + \frac{10}{60} &\leq 18.5 \quad i = 1,2,3, j = 1,2,\ldots,1320 \\ y_{i,j} + 2d_{i,j} + \frac{k_i}{120} * 0.33 &\leq 18.5 \quad i = 1,2,3, j = 1,2,\ldots,1320 \end{aligned} \tag{23}$$

其中，$d_{i,j}$表示从仓库i到居民点j的时间，根据距离和速度求解；k_i表示居民点j的人口数；0.33 是物资 A 一个箱子的体积。

每个居民点都需要收到足够的物资 A：

$$\sum_{i=1}^{3} x_{i,j} \times 600 \geq k_j \tag{24}$$

其中，k_j表示居民点 j 需要的物资 A 数量。

物资 A 和物资 B 需要在规定时间内到达或同时到达：

$$\sum_{i=1}^{3} \sum_{j=1}^{1320} \left(x_{i,j} \times \left(1 + 0.0069576 \times \text{最大} \left(k_j - \left\lfloor \frac{k_j}{120} \right\rfloor \times 8,0 \right) \right) \right) \leq 9.36 \times 10^4 \tag{25}$$

同时，需要保证物资 B 在物资 A 之前或同时送达，即：

$$y_{i,j} + \frac{\text{最大}\left(k_j - \left\lfloor \frac{k_j}{120} \right\rfloor \times 8,0\right)}{10 \times x_{i,j}} \leq y_{i',j} + \frac{0.33^3 \times 0.6 \times 0.3 \times 0.2 \times \text{最大}\left(k_j - \left\lfloor \frac{k_j}{120} \right\rfloor \times 8,0\right)}{10 \times x_{i',j}} \tag{26}$$

其中，i表示物资 A 所在的货车，i'表示物资 B 所在的货车。货车数量不能为负数：

$$x_{i,j} \geq 0, y_{i,j} \geq 0 \tag{27}$$

同时，由于居民的满意度也需要考虑进来，因此还需要增加以下约束条件：居民满意度调整：

$$coef_{i,j} = \begin{cases} 1, \ if y_{i,j} \leq 17 \\ 1 - (y_{i,j} - 17), if 17 < y_{i,j} < 18.5 \\ 0, \ if y_{i,j} \geq 18.5 \end{cases} \tag{28}$$

$$\sum_{i=1}^{3}\sum_{j=1}^{1320}\left(x_{i,j}\times coef_{i,j}\right)\le 3.96\times 10^{5} \tag{29}$$

其中，$coef_{i,j}$表示第i辆货车在居民点j卸货的时间对应的满意度系数，3.96×10^{5}是所有居民点的总数乘以满意度系数的平均值。

物资 A 需求量过大的问题：

$$\begin{gathered}\sum_{i=1}^{3}\sum_{j=1}^{1320}\left(x_{i,j}\times 600\right)\ge\sum_{j=1}^{1320}k_j\\ \sum_{i=1}^{3}\sum_{j=1}^{1320}\left(x_{i,j}\times 600\right)\le\sum_{j=1}^{1320}ceil\left(\frac{k_j}{600}\right)\times 600\\ \sum_{j=1}^{1320}ceil\left(\frac{k_j}{600}\right)\times 600\le\left(\sum_{i=1}^{3}\sum_{j=1}^{1320}x_{i,j}\right)\times 1200\end{gathered} \tag{30}$$

其中，第一个约束条件表明，所有物资 A 的需求量总和不应该超过需要的总量；第二个约束条件表明，如果某个居民点需要的物资 A 数量超过了一辆货车的运载能力，就需要用多辆货车来运输，而这些货车必须在下午 6:30 之前到达该居民点，并且不能超过运送物资 A 所需的最 小货车数量；第三个约束条件表明，所有用来运输物资 A 的货车不能超过总货车数量的 2 倍。

图 7 模型建立思路图

该模型是一个多目标规划问题，需要综合考虑需要的货车数量、运输时间和居民的满意度等因素，可以采用集成多目标优化算法来求解。。

5.3.3 模型的求解

- **确定决策变量**

我们需要从仓库i到居民点 j 的货车数量$x_{i,j}$和第i辆货车在居民点j的卸货时间$y_{i,j}$作为决策变量，以求解最优方案。

- **定义目标函数**

本题中，我们希望将所有货车数量的总和最小化，即 $\min\sum_{i=1}^{3}\sum_{j=1}^{1320}x_{i,j}$，这是我们需要优化的目标函数。

- **确定约束条件**

在实际问题中，会存在很多约束条件，而本题中，需要考虑以下因素：

时间约束：每个居民点的收货时间不能晚于下午 18:30，每个物资也需要在规定时间内到达或在同一时间到达。

数量约束：每个居民点需要足够的物资 A、货车数量不能为负数和物资 A 需求量过大的问题等。

满意度约束：通过货车卸货的时间和成本等计算居民的满意度。

- **设定遗传算法的参数**

在实现集成多目标优化时，可以采用 NSGA-II 算法。设置遗传算法的参数，包括种群规模、交叉和变异的方式、选择算子、迭代次数等等 。具体设置如下：
群体规模：100-200（通常选择 128）；交叉代数：35%；变异代数：10%；迭代次数：2000

- **使用 NSGA-II 算法求解[3]**

使用 NSGA-II 算法求解过程如下：
初始化群体，计算适应度函数；进行交叉和变异，计算新适应度函数；计算非支配排序和拥挤距离；选择下一代群体；迭代以上步骤，直到达到指定的迭代次数

- **分析解决方案**

分析求解得到的最优解，根据货车数量、运输时间和居民满意度等方面，进行可视化或统计分析，并确定最终的解决方案。同时，根据实际情况进行优化，以得到最优的解决方案。

总之，在实际计算中，集成多目标优化算法可以实现对多个目标同时考虑的优化，同时，根据目标函数和约束条件进行设置，可以得到很好的求解结果。因此，对于多目标优化问题的求解，集成多目标优化算法是一种非常有效和高效的方法。

5.4 问题四：利用 ARIMA 模型合理储存物资

5.4.1 数据收集与整理[7]

- **国家卫生健康委员会的官方数据：**可以从各省市的卫生健康委员会或疾控中心网站获取新冠疫情数据，例如，从上海市疾控中心网站获取上海市 2022 年 6 月 1 日至 2022 年 6 月 7 日的疫情数据：

表 5 上海市疫情数据

日期	本土感染	本土无症状	本土感染累计	本土无症状累计
2022 年 6 月 7 日	4	11	58035	591404
2022 年 6 月 6 日	3	7	58031	591393
2022 年 6 月 5 日	4	4	58028	591386
2022 年 6 月 4 日	6	16	58024	591382
2022 年 6 月 3 日	5	9	58018	591366
2022 年 6 月 2 日	8	8	58013	591357
2022 年 6 月 1 日	5	8	58005	591349

使用统计方法和可视化工具对数据进行分析，例如以上海市疫情数据为例，可以绘制出 7 天的确诊病例数的折线图，对疫情的传播趋势进 行分析和预测。

- **新闻报道：**可以从国内各大新闻媒体了解新冠疫情的最新动态和政府应对措施，例如，资讯网站新浪网上发布的新闻报道称，中国近期出现的新冠病例主要集中在广东、福建、江苏等地，这些地区已经采取了严格的防控措施来遏制病毒传播。

- **疫苗接种数据：**可以从国家药品监督管理局网站获取最新的疫苗接种数据，例如，2023 年 5 月 21 日，中国大陆已经累计接种新冠疫苗 34.6 亿剂次。
- **社交媒体数据：**可以使用社交媒体平台上的关键词搜索工具收集与新冠疫情相关的文字、图片和视频等内容，例如，从微博搜索“新冠”关键词，可以找到最受关注的新闻、话题、用户发帖等信息，了解国内居民的反应和动态。

5.4.2 预测模型的建立

- **确定时间序列数据的平稳性**

在建立 ARIMA 模型之前，需要先判断时间序列数据的平稳性。一个平稳的时间序列数据在不同时间段内的均值、方差等统计指标是不改变的，且呈现随机波动的趋势。如果数据不是平稳的，需要对数据进行差分操作，将其转换成平稳的数据，以便更好地分析和预测。

平稳序列的检验使用单位根检验，通常使用自回归滞后单位根检验（ADF 检验）来检验。其原假设为序列存在单位根，即序列不平稳，检验统计量为：

$$ADF = \frac{\Delta y_t - \gamma \Delta y_{t-1}}{se} \tag{31}$$

其中 Δy_t 表示时间序列的第 t 次差分，γ 是回归系数，se 是标准误差。

- **分析 ACF 和 PACF**

在确定平稳时间序列后，需要使用自相关函数（ACF）和偏自相关函数（PACF）来选择 AR 和 MA 的阶次，这对于模型的精度和准确性非常重 要。ACF 和 PACF 可以画成图表，使我们能够看出哪些滞后阶次的样本自相关系数或偏自相关系数不为零，然后根据图表中的结果来选择 AR 和 MA 的阶次。

自相关函数和偏自相关函数被广泛应用于时间序列数据建模。 它们可以用于帮助确定滞后程度，无论是 AR、MA 还是 ARIMA 模型。

自相关函数定义为：

$$r_k = \frac{\gamma_k}{\gamma_0}, k = 0, \pm 1, \pm 2, \ldots \tag{32}$$

其中 γ_k 表示信号与类似于自身但是移位了 k 个样本的版本之间的协方差。

类似地，偏自相关函数定义为：

$$\alpha_{1,1} = \rho_1 \tag{33}$$

$$\alpha_{j,j} = det(R_j)/det(R_{j-1}), j = 2,3, \ldots, p, \tag{34}$$

$$\alpha_{j,k} = det\begin{bmatrix} R_{j-1} & R_{j,k-1} \\ R_{k-1,j} & R_{k,k} \end{bmatrix} / det(R_j), 1 < k < j \tag{35}$$

其中 R_j 表示前 j 阶的样本自协方差矩阵，$R_{j,k}$ 表示前 j 阶与前 k 阶的样本互协方差矩阵。

- **确定模型的阶数**

在确定 AR 和 MA 的阶次后，需要确定模型的阶数。一般来说，模型的阶数可以由以下三个参数确定：AR 阶数 p、差分次数 d 和 MA 阶数 q。通常我们会使用 BIC 准则来选择最优模型阶数，也可以使用其它准则。

- **拟合 ARIMA 模型**

通过进一步的分析和评估，确定模型的阶数后，可以根据这些详细参数来拟合 ARIMA 模型。在模型拟合后，需要使用数据集以及拟合参数 来预测最终的结果。

ARIMA 模型被定义为：

$$\left(1-\phi_1 B-\phi_2 B^2-\cdots-\phi_p B^p\right)\Delta y_t=c+\left(1+\theta_1 B+\theta_2 B^2+\cdots+\theta_q B^q\right)u_t \tag{36}$$

其中 Δy_t表示序列的一阶差分，B 是时间步长的平移算子，p 和 q 分别表示自回归项和移动平均项的滞后阶数，ϕ 和 θ 是自回归和移动平均模型参数，u_t 是以零为中心的白色噪声。

- **模型诊断**

模型诊断帮助我们确定模型的质量和准确性，以及如果模型存在问题，我们需要采取的额外措施。模型诊断可以绘制残差的自相关函数和正态概率图，来评估模型的性能。

- **针对模型的选择和误差进行检验**

需要测试模型和数据之间的各种参数、阶数、误差、置信区间等等，以便确定模型所具有的可用性和准确性。对于检验残差的等价性和标准性的情况，我们可以对模型的系数进行百分比误差测试。

- **使用模型来预测**

在模型构建、诊断和检验后，就可以使用 ARIMA 模型来预测时间序列了。这可以通过使用 `predict` 方法来实现。预测结果将替换掉时间序列中最后一个时间步的数据，以此来预测未来时间步中的取值。

综上所述，ARIMA 模型的建立过程比较繁琐，需要经过多个步骤才能够得到最终的结果。但只要同时注意这些步骤，并理解每一步的目的 ，就可以实现准确有效的时间序列预测。

5.4.3 模型的求解

假设 ARIMA(p,d,q)模型的随机过程 y_t 可以表示为

$$y_t=c+\epsilon_t+\phi_1 y_{t-1}+\cdots+\phi_p y_{t-p}+\theta_1\epsilon_{t-1}+\cdots+\theta_q\epsilon_{t-q} \tag{37}$$

其中 c 是常数，ϵ_t是零均值、方差为 σ^2 的独立同分布白噪声序列。

对于一个长度为 T 的时间序列，假设 $y_{1:T}$ 是已知的，需要对模型参数 $\phi_1,\cdots,\phi_p,\theta_1,\cdots,\theta_q,c,\sigma^2$进行最大似然估计，即找出所有可能的参数组合中，生成 $y_{1:T}$ 的概率最大的那组参数。假设$y_{t-1},y_{t-2},\cdots,y_{t-p}$和$\epsilon_{t-1},\epsilon_{t-2},\cdots,\epsilon_{t-q}$ 均已知，则有：

$$\widehat{y_t}=c+\epsilon_t+\phi_1 y_{t-1}+\cdots+\phi_p y_{t-p}+\theta_1\epsilon_{t-1}+\cdots+\theta_q\epsilon_{t-q} \tag{38}$$

可得：

$$\epsilon_t=y_t-c-\phi_1 y_{t-1}-\cdots-\phi_p y_{t-p}-\theta_1\epsilon_{t-1}-\cdots-\theta_q\epsilon_{t-q} \tag{39}$$

假设ϵ_t是独立同分布的，那么可能性函数为：

$$L\left(\phi_1,\cdots,\phi_p,\theta_1,\cdots,\theta_q,c,\sigma^2\middle|y_{1:T}\right)=\prod_{t=1}^{T}\frac{1}{\sqrt{2\pi\sigma^2}}exp\left(-\frac{\epsilon_t^2}{2\sigma^2}\right) \tag{40}$$

取对数并对参数求导，得到最大化可能性函数的估计量。

最终的参数估计量由迭代过程得出，通常使用最大化似然函数的方法进行。在确定模型参数后，我们可以使用训练数据集的 ARIMA 模型来 预测新的数据。

6 模型的优缺点

6.1 模型的优点：

- **适用范围广：**ARIMA 模型可以应用于多种时间序列数据，如经济、环境、医疗、工业等领域的数据预测；整数规划模型可以应用于工业、交通、医疗、 物流、能源等领域资源优化与分配问题；混合整数规划模型可以将整数规划模型与线性规划模型相结合，适用范围更广，可以为多种实际问题提供有效的解决方法，如图像处理、金融风险管理等。
- **可以解决实际复杂问题：**ARIMA 模型可以通过时间序列的先验知识对噪声和异常值进行处理，同时可以预测未来的趋势；整数规划模型可以实现优化路径、分配和调度，最大化资源利用，提高利润或效益；混合整数规划模型可以解决复杂的组合优化问题，例如车辆路径问题。这三种模型都可以有效地解决实际复杂问题，提高决策的准确性和效率。
- **灵活性：**混合整数规划模型和整数规划模型都具有很高的灵活性，可以根据具体的情况选择不同的算法或策略进行求解，并根据实际情况对模型进行调整，以获得更优的解决方案。与此同时，ARIMA 模型的求解过程简单，可以方便地进行参数估计和预测。
- **准确性：**ARIMA 模型、整数规划模型和混合整数规划模型的求解结果都能够准确地预测未来或优化当前问题，可以帮助决策者制定更具针对性的决 策方案。特别是在缺失数据、数据变化较大时，ARIMA 模型可以准确地进行预测，通过对数据的细致分析，提供精准的预测结果。
- **能够深入分析问题：**三种模型都可以深入分析问题，探究变量之间的联系，发现问题的关键因素，从而提供更为精准的预测或决策依据。深入分析问题有助于理解问题本质，优化问题的解决方案，提高优化的效率和效果。

总之，ARIMA 模型、混合整数规划模型和整数规划模型都是实用的数学模型，在不同领域和问题中均有广泛的应用。三种模型各自有独特 的优点和适用范围，可以根据实际问题需要进行合理的选择和打造。识别各自的优点和局限性有助于更好地理解三种模型的特点，并根据实际情况进行优化选择

6.2 模型的缺点：

- **可能存在预测误差：**ARIMA 模型基于时间序列数据进行预测，但如果时间序列数据中出现了大的异常值或趋势变化，就可能导致预测 误差增大。同时，在数据缺失较多的情况下，ARIMA 模型的预测效果也会受到较大影响。
- **具有局限性：**整数规划模型和混合整数规划模型的求解时间随着问题规模的增加而呈指数级增长，难以处理大规模问题。同时，这两种模型对决策者的求解经验要求较高，在解决某些复杂问题时，可能会产生死角或局限性。
- **数据缺失的影响：**三种模型的求解都需要大量的数据支持，一些缺失数据可能会影响模型的求解效果。在实际应用中，如何克服数据缺失对模型性能的影响也是一个挑战。

7 参考文献

[1]陈明华，李迎秋，罗耀琪.应急物流车辆调配问题的研究.计算机工程与应用，2009，45(24)：194~197..

[2]Tzeng G H, Cheng H J, Huang T D. Multi-objective optimal planning for designi ng relief deliverysystems transport. Berlin: Logistics & Transportation Rev, 2007, 43(6): 673~686.

[3]董雅文,杨静雯,刘文慧,赵小惠.基于 NSGA-Ⅱ算法的应急物资运送路径选择[J].现代商贸工业,2021,42(19):12-14.DOI:10.19311/j.cnki.1672-3198.2021.19.006.

[4]杨一凡.基于分治的物流路径规划算法[J].物流技术与应用,2023,28(03):154-160.

[5]张海.突发公共卫生事件应急物资运输问题研究[J].中国储运,2023(03):72-73.DOI:10.16301/j.cnki.cn12-1204/f.2023.03.035.

[6]余海燕,苟梦圆,吴腾宇.应急物资的无人机与车辆并行在线配送问题[J/OL].计算机工程与应用:1-11[2023-05-22].http://kns.cnki.net/kcms/detail/11.2127.TP.20220726.1842.020.html

[7]王雅. 新冠疫情下大型城市应急医疗物资供应网络规划研究[D].华北电力大学(北京),2022.DOI:10.27140/d.cnki.ghbbu.2022.001020.

[8]张笑晨.基于物联网定位技术的应急物资运送路径选择模型[J].信息与电脑(理论版),2022,34(05):194-196.

[9]王金英,包立军.应急物资配送问题研究[J].辽宁工业大学学报(自然科学版),2021,41(05):325-329.DOI:10.15916/j.issn1674-3261.2021.05.011.

[10]李卓,李引珍,李文霞.应急物资运输路径多目标优化模型及求解算法[J].计算机应用,2019,39(09):2765-2771.

[11]费思邈,霍琳,王亮等. 基于聚类分析的飞行数据异常检测方法[C]//测控技术编辑部.2015 航空试验测试技术学术交流会论文集.测控技术编辑部,2015:277-279+302.

[12]陈钢铁,帅斌.在时间窗条件下应急物资运输路径优化问题研究[J].铁道运输与经济,2010,32(03):70-72+77.

8 附录

附录一：货车运载量计算的python代码

```
import math

# 每个箱子的尺寸
box_length = 0.3
box_width = 0.3
box_height = 0.3

# 货车的尺寸
truck_length = 8.5
truck_width = 3
truck_height = 3.5

# 计算最多能装载的箱子数量
max_boxes_length = math.floor(truck_length / box_length)
max_boxes_width = math.floor(truck_width / box_width)
max_boxes_height = math.floor(truck_height / box_height)

max_boxes = max_boxes_length * max_boxes_width * max_boxes_height

print("最多能装载的箱子数量为：", max_boxes, "箱")

# 计算一辆货车可以运输的物资数量
boxes_per_truck = max_boxes
items_per_box = 600
items_per_truck = boxes_per_truck * items_per_box

print("一辆货车最多可以运输", items_per_truck, "件物资")
```

附录二：查找最近仓库的python代码

```
import pandas as pd

# 读取距离表格
distance_table = pd.read_excel('距离表.xlsx')
# 获取居民点列表
resident_points = distance_table.columns[1:].tolist()

# 对每一个居民点，找到最近的仓库和距离
```

```
result_list = []
for resident_point in resident_points:
    distances = distance_table[resident_point].tolist()
    # 获取最小值及其所在位置
    min_distance, min_index = min((val, idx) for (idx, val) in enumer-
ate(distances))
    # 根据最小值所在位置，获取对应的仓库名称
    nearest_warehouse = distance_table.iloc[min_index, 0]
    # 将结果添加到列表中
    result_list.append([resident_point, nearest_warehouse, min_dis-
tance])

# 将结果列表转化为 DataFrame 格式
result_table = pd.DataFrame(result_list, columns=['居民点', '最近仓库', '
距离'])
# 将结果输出到 Excel 文件
    result_table.to_excel('仓库表.xlsx', index=False)
```

附录三：生成居民点和最近仓库的无向图的python代码

```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt

# 读取数据表格
data_table = pd.read_excel('仓库表 1.xlsx')
# 生成节点和连边
G = nx.Graph()
for _, row in data_table.iterrows():
    resident_point = row[0]
    warehouse_name = row[1]
    distance = row[2]
    G.add_edge(resident_point, warehouse_name, weight=distance)

# 对无向图进行布局
pos = nx.spring_layout(G)

# 绘制节点及节点之间的连边
plt.figure(figsize=(20, 20))
nx.draw_networkx_nodes(G, pos, nodelist=list(data_table['居民点
'].unique()), node_size=300, node_color='skyblue', alpha=0.9, edgecol-
ors='black')
```

```
nx.draw_networkx_nodes(G, pos, nodelist=list(data_table['最近仓库
'].unique()), node_size=1000, node_color='red', alpha=0.9, edgecol-
ors='black')
nx.draw_networkx_edges(G, pos, width=0.2, alpha=0.7)
nx.draw_networkx_labels(G, pos, labels={node: node for node in
list(data_table['居民点'].unique())}, font_size=6, font_family='SimHei')
nx.draw_networkx_labels(G, pos, labels={node: node for node in
list(data_table['最近仓库'].unique())}, font_size=14, font_family='Sim-
Hei')
plt.title('全局无向图')
plt.axis('off')
plt.show()

# 对无向图进行最小生成树算法
mst = nx.minimum_spanning_tree(G)

# 绘制最小生成树
plt.figure(figsize=(20, 20))
nx.draw_networkx_nodes(mst, pos, nodelist=list(data_table['居民点
'].unique()), node_size=300, node_color='skyblue', alpha=0.9, edgecol-
ors='black')
nx.draw_networkx_nodes(mst, pos, nodelist=list(data_table['最近仓库
'].unique()), node_size=1000, node_color='red', alpha=0.9, edgecol-
ors='black')
nx.draw_networkx_edges(mst, pos, width=0.2, alpha=0.7)
nx.draw_networkx_labels(mst, pos, labels={node: node for node in
list(data_table['居民点'].unique())}, font_size=6, font_family='SimHei')
nx.draw_networkx_labels(mst, pos, labels={node: node for node in
list(data_table['最近仓库'].unique())}, font_size=14, font_family='Sim-
Hei')
plt.title('最小生成树')
plt.axis('off')
    plt.show()
```

附录四：用pulp进行模型求解的python代码

```
import pulp
import pandas as pd

df_distance = pd.read_excel('distance.xlsx')
df_demand = pd.read_excel('demand.xlsx')
```

```
prob = pulp.LpProblem('DisasterRelief', pulp.LpMinimize)


n = len(df_distance)
m = len(df_demand)
K = int(m / C_max) + 1  # 总车数

# 定义变量
x = [[[pulp.LpVariable('x({},{},{})'.format(i, j, k), lowBound=0, up-
Bound=1, cat=pulp.LpInteger)
       for k in range(1, K + 1)] for j in range(1, m + 1)] for i in
range(1, n + 1)]
y = [pulp.LpVariable('y({})'.format(k), lowBound=0, upBound=1,
cat=pulp.LpInteger) for k in range(1, K + 1)]

# 目标函数
prob += pulp.lpSum([y[k - 1] for k in range(1, K + 1)])

# 约束条件
# 让每个居民点都在 18:30 前收到物资 A
for j in range(1, m + 1):
    prob += pulp.lpSum([x[i-1][j-1][k-1] * (df_distance.iloc[i-1][j-1] /
v + t_ud) for i in range(1, n + 1) for k in range(1, K + 1)]) + t_r <=
630

# 每一辆货车可以运送的最大物资 A 箱数不超过 C_max
for i in range(1, n + 1):
    for k in range(1, K + 1):
        prob += pulp.lpSum([x[i-1][j-1][k-1] * df_demand.iloc[j-1]['De-
mand'] for j in range(1, m + 1)]) <= C_max * y[k-1]

# 每个居民点必须收到物资 A
for j in range(1, m + 1):
    prob += pulp.lpSum([x[i-1][j-1][k-1] for i in range(1, n + 1) for k
in range(1, K + 1)]) == df_demand.iloc[j-1]['Demand']

# 每一辆货车从一个仓库出发，从一个仓库到另一个仓库且最终返回第一个仓库
for k in range(1, K + 1):
    for i in range(1, n + 1):
        prob += pulp.lpSum([x[i-1][j-1][k-1] for j in range(1, m + 1)])
- pulp.lpSum([x[j-1][i-1][k-1] for j in range(1, n + 1)]) == 0
```

```
        prob += pulp.lpSum([x[i-1][j-1][k-1] for j in range(1, m + 1)])
- pulp.lpSum([x[i-1][j-1][k-1] for j in range(1, i)]) - \
            pulp.lpSum([x[j-1][i-1][k-1] for j in range(i+1, n+1)]) == 0

# 每个仓库向每个居民点最多只能提供一次物资 A
for i in range(1, n + 1):
    for k in range(1, K + 1):
        prob += pulp.lpSum([x[i-1][j-1][k-1] for j in range(1, m + 1)])
<= 1

# 物资 A 必须被完全运送到某个目的地或调配到另一个目的地
prob += pulp.lpSum([x[i-1][j-1][k-1] for i in range(1, n + 1) for j in
range(1, m + 1) for k in range(1, K + 1)]) == S

# 货车编号是从 1 开始连续编号
for k in range(1, K):
    prob += y[k - 1] - y[k] <= 0

# 求解问题
prob.solve()

# 根据求解结果打印出送货路线
for k in range(1, K+1):
    print("========================================")
    print("第{}辆车的路线：".format(k))
    route = []
    for i in range(1, n+1):
        for j in range(1, m+1):
            if pulp.value(x[i-1][j-1][k-1]) == 1:
                print("从仓库{}出发，运送物资到居民点{}，距离为{}km".for-
mat(i, j, df_distance.iloc[i-1][j-1]))
                route.append(i)
                route.append(j)
    route.append(1)
    print("从居民点{}返回仓库 1".format(j))
    print("总距离为{}km".format(sum([df_distance.iloc[route[i-1]-
1][route[i]-1] for i in range(1, len(route))])))
    print("========================================")

    print("最小化送货车辆数为：", pulp.value(prob.objective))
```

附录五：时序模型预测的python代码

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller

# 读取数据
df = pd.read_csv('new_cases.csv', index_col='date',
parse_dates=['date'])
ts = df['new_cases']

# 平稳性检验
result = adfuller(ts)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
if result[1] > 0.05:
    print('The time series is not stationary')
else:
    print('The time series is stationary')

# ACF、PACF 函数绘图
fig, axes = plt.subplots(2, 1, figsize=(10, 6))
plot_acf(ts, ax=axes[0])
plot_pacf(ts, ax=axes[1])
plt.show()

# 确定 ARIMA 模型阶数
model = ARIMA(ts, order=(2,1,2))
results = model.fit()

# 预测未来值
pred = results.predict(start='2021-6-1', end='2022-6-1', dynamic=False)

# 绘制预测结果图
fig, ax = plt.subplots(figsize=(10, 6))
ts.plot(ax=ax, label='Actual', color='black')
pred.plot(ax=ax, label='Predict', color='red')
plt.legend()
plt.title('Prediction of new cases in the next year')
plt.show()
```