2020 年武汉理工大学大学生数学建模竞赛

题 目：基于模拟退火算法求解路径规划问题的混合 0-1 整数模型

摘 要：

本文以最优化理论为基础，研究车辆运输路径最优化的问题。通过建立混合 0-1 整数规划模型，并利用**蚁群算法**和**模拟退火算法**进行求解。从而得出运输车效率最高的运输路径。

关于问题一 首先分析得出一辆运输车可以一次运输完所有货物，进而将问题转换为旅行商问题，利用 LINGO 进行求解。最终得出了该运输车耗时最短的路径，共耗时为 9.23 小时，其路径长度为 116 公里，运输费用为 3535.3 元。

关于问题二 该问题增加了对车辆的额定载重和总工作时间的限制，使问题转换为增加了约束的路径规划问题。本文根据实际物流过程，将调度效率最大化转化为运输费用最小化，并以此为目标建立了**混合 0-1 整数规划模型**。在模型求解的过程中，首先利用蚁群算法得到一个上界解，并根据此解缩小了解空间。之后利用改进的模拟退火算法对模型进行求解。最后得出需要运输车的数量为 2，其最小运输费用为 835.00 元，两辆运输车分别用时 2.83 小时和 3.84 小时。

关于问题三 此问题在问题二的基础上增加了车型的选择。首先根据问题二的相关证明，可知运输车的总数为 2 辆，并基于此将问题转化为确定车辆数量的路径规划问题。进而以最小运输费用为目标，建立每一种车型组合的混合整数规划模型，之后利用改进的模拟退火算法计算得到最优解。通过基本的分析，发现最佳方案是采用两辆载重量为 4 吨的货车，其费用为 775.40 元，两辆车分别用时 3.79 小时和 3.95 小时。

本文详细的描述了调度过程中的各约束条件，符合现实情况。并且将模型中的非线性项线性化，提高了模型的性能。在求解过程中，采用了改进的模拟退火算法，保证结果性质良好的前提下，增加了求解速度。

关键词：0-1 整数规划 模拟退火算法 路径规划问题 蚁群算法

目录

一、 问题重述

1.1 问题背景

物流运输是现在供应链不可缺少的一部分，今年来随着经济的快速发展，物流运输行业迅速崛起，配送车辆路径的选择密切关系到了企业的配送成本，因此如何优化配送路线提高企业竞争力成为了许多学者关注、研究的重点问题。物流配送是指在一定范围内，根据用户的需求，对物资进行拣选、加工、包装、分割、组配等作业，并按时送达用户指定地点的物流活动。

1.2 问题概述

问题一：根据题中给定的 19 个站点，用载重 100 吨的运输车，以 40 公里/小时的速度进行配送，其中每个销售点需要 20 分钟的时间下货，空载费用为 0.6 元/公里，并在送完所有食物后返回仓库。求最短时间，并给出运输方案。

问题二：现有一种平均速度为 50km/小时的小型运输车，每个销售点需要 5 分钟的下货时间，载重为六吨，空载费用为 0.4 元/公里；求他们送完所有食物并返回仓库的总体调度效率最高的调度方案。

问题三：现有载重为分别为 6 吨和 4 吨的运输车，空载费用分别为 0.2、0.4 元/公里，其他条件均相同。求送完所有食物的调度效率最高的车辆数和调度方案。

二、 模型的假设

- 考虑到现实物流活动中频繁接收货物导致对接成本过高，对于同一个需求点的货物只允许一次性满足;
- 需求点的信息在运输之前被全部告知，不存在运输过程中新增需求点的情况;
- 不考虑天气、路况等可能对运输速度造成影响的随机因素。

三、 符号说明

符号	意义
d_{ij}	表示站点 i 与站点 j 之间的距离
x_{ij}	站点 i,j 之间是否有连线的判断系数
N	需求点的集合 $i,j,p.$
V	车辆的集合，下标为 $v.$
R	车辆每次出行顺序的集合，下标为 $r.$

四、 问题一的模型建立与求解

4.1 问题分析

问题一要求派一辆 100 吨载重量的货车，尽可能快地将货物送到表一所给的 19 个站点，并求出得出相应方案。

通过对表一数据的分析可以算出 19 个站点的总需求量为 31.15 吨，小于货车的总载重量。说明该货车能一次运送所有货物，并返回原点。可以看出问题一是典型的

TSP（旅行商问题：假设有一个旅行商人要拜访 n 个城市，他必须选择所要走的路径，路径的限制是每个城市只能拜访一次，而且最后要回到原来出发的城市）。并且据题目中所给的假设条件：街道方向均平行于坐标轴，任意两站点间都可以通过一次拐弯到达。此处我们计算两点间的距离应使用曼哈顿距离而非欧氏距离。

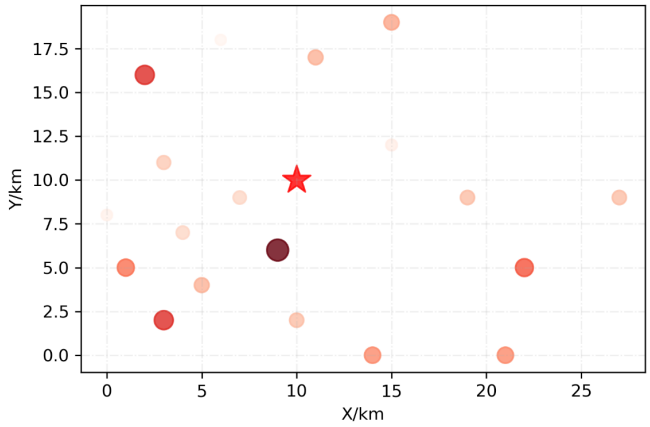

图 1　站点示意图

4.2 预备工作

根据题目给定的街道平行于坐标轴的假设可知，各需求点之间的距离可以用曼哈顿距离进行表示。则需求点 i 和需求点 j 之间的距离计算公式为：

$$d_{ij} = |x_i - x_j| + |y_i - y_j|$$

遍历所有需求点，得到了各需求点之间的距离矩阵

i / j	1	2	3	4	…	18	19	20
1	0	5	4	6	…	31	29	15
2	5	0	4	9	…	30	28	14
3	4	5	0	5	…	27	25	11
4	6	5	4	0	…	27	23	9
…	…	…	…	…	…	…	…	…
18	31	30	27	25	…	0	22	18
19	29	28	25	23	…	22	0	14
20	15	14	11	9	…	18	14	0

4.3 模型的建立

建立整数线性规划（LIP）模型，设每个站点的距离用矩阵 d 来表示，d_{ij} 表示站点 i 与站点 j 之间的距离。设 $(0,1)$ 矩阵用来表示经过的各站点之间的路线。设

$$x_{ij} = \begin{cases} 1, & \text{if 从站点 i 到站点 j;} \\ 0, & \text{if 站点 i 不到站点 j.} \end{cases} \tag{1}$$

并且由于是一个闭环每个站点前后都只有一个城市，则应当满足：

$$\sum_{i=1,i\neq j}^{n} x_{ij} = 1, i = 1, 2, ...n \tag{2}$$

$$\sum_{j=1,j\neq i}^{n} x_{ij} = 1, i = 1, 2, ...n. \tag{3}$$

但以上约束条件不能避免在遍历中产生多个互不连通的闭环。

为此需要额外引入一个变量 $u_i \quad (i = 1, 2, ...n)$ 使得

$$u_i - u_j + nx_{ij} \leq n - 1, 1 < i \neq j \leq n \tag{4}$$

对 (4) 约束的解释：

任意的 i 和 j 不会构成回路, 若出现了子环，则会存在 $x_{ij} = 1, x_{ji} = 1$ 的出现，使得

$$u_i - u_j \leq -1, u_j - u_i \leq -1$$

两个不等式相加则出现了 $0 \leq -2$, 导致矛盾。

任意的 i, j, k 不会构成回路，若构成了子环则会存在 $x_{ij} = 1, x_{jk} = 1, x - ki = 1$ 的出现，使得

$$u_i - u_k \leq -1, u_j - u_k \leq -1, u_k - u_i \leq -1$$

三个不等式相加则出现了 $0 \leq -3$, 导致矛盾。

于是我们可以得出以下模型：

$$\min \sum_{i,j=1}^{n} x_{ij} d_{ij} \tag{5}$$

$$s.t. \begin{cases} \sum_{i=1,i\neq j}^{n} x_{ij} = 1, i = 1, 2, ...n, \\ \sum_{j=1,j\neq i}^{n} x_{ij} = 1, i = 1, 2, ...n, \\ u_i - u_j + nx_{ij} \leq n - 1, 1 < i \neq j \leq n, \\ x_{ij} \in \{0, 1\} \quad i, j = 1, 2, ...n \\ u_{ij} \in \mathbb{R} \qquad i = 1, 2, ...n \end{cases} \tag{6}$$

4.4 模型求解

利用 lingo 软件对上述混合整数线性规划模型进行求解。得出最短路径的路线为：$20 \to 7 \to 4 \to 3 \to 1 \to 2 \to 5 \to 6 \to 11 \to 12 \to 13 \to 19 \to 14 \to 15 \to 18 \to 16 \to 17 \to 10 \to 9 \to 8 \to 20$。最短距离：116 公里，最短时间：9.23 小时，花费金钱：3535.3 元。

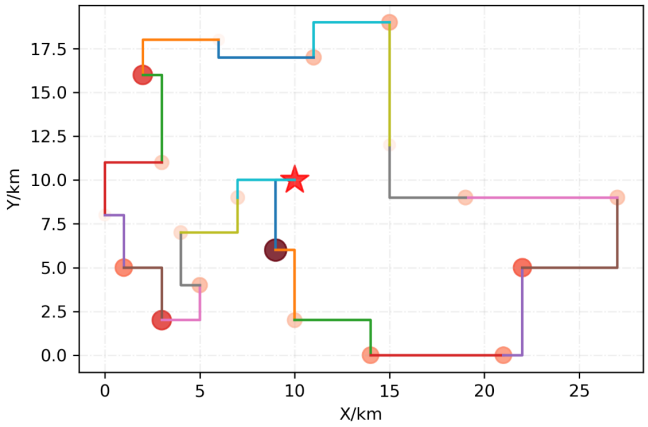

图 2　运输轨迹图

五、 问题二模型的建立与求解

5.1 问题分析

问题二要求用平均速度为 50 公里/小时载重为 6 吨小型运输车，求送完所有食物并返回仓库的总体调度效率最高的调度方案。由于载重仅为 6 吨没有办法一次送完所有食物，所以需要派出多辆车进行多次送货。其中，本问题本质上是一个需要同时考虑到车辆数量不确定 [1]，每辆车可重复使用 [2]，并且车辆具有载重与时间的约束的路径规划问题（VRP，vehicle routing problem）。在实际生产中，提高调度效率本质上是增加运输效率，即在物流活动中尽可能使用最少的成本获取最大的收益。由于此次物流活动的收益并不会因为耗费的时间或到达需求点的顺序对收益产生影响，所以认为收益为固定值，并将求解运输总费用最小化作为本题调度目标。

考虑到实际物流活动中，设备的购置、租赁价格以及后续的保养费用不菲，在对问题进行求解时，应在保证车辆的数量尽可能少的前提下进行。

5.2 预备工作

因为每辆车可以多次使用，所以题目中给出的总工作时间的上限既可以是表达对车辆的约束，也可以是表达对车辆每次出行的约束。若将两种情况的约束都表达出来，会导致大量的无效约束。因此，首先通过计算上限证明：工作时间的上限仅对车辆产生有效约束。首先，可以确定由于额定载重 6 吨的限制，任意一辆车在一次出行时，无法一次性满足 19 个需求点共 31.25 吨的货物。只要证明在额定载重的约束下，一次出行的最长运输时间 Z_{max} 小于四小时，则可完成证明。

5.3 模型建立

首先对本题出现的变量和集合进行说明。

N— 需求点的集合，下标为 i, j, p.

V— 车辆的集合，下标为 v.

R— 车辆每次出行顺序的集合，下标为 r.

式中，L_v 表示车辆 v 的额定载荷，Q_i 表示需求点 i 货物的重量，a_{ij} 表示从 i 出发到达 j 所经过路径的长度。

另外，设 $0-1$ 变量 y_i^{vr} 和 x_{ij}^{vr} 对图中的各顶点和边的状态进行表达。

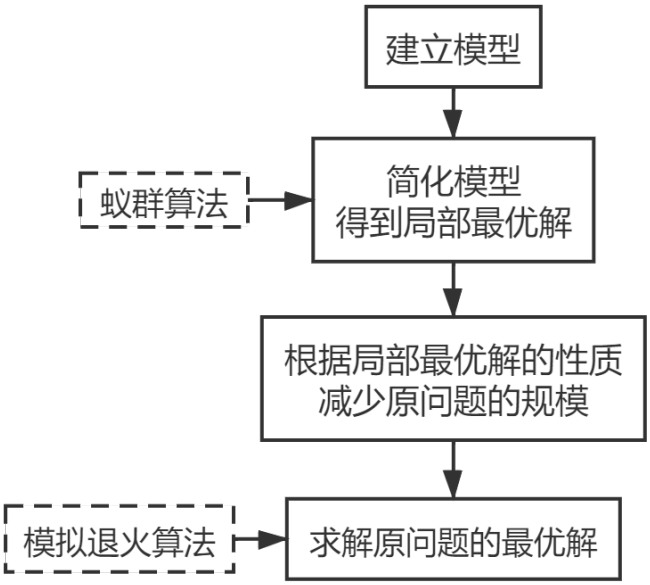

图 3　问题二流程图

$$y_i^{vr}=\begin{cases}1, & \text{if 第 v 辆车在第 r 次出行中经过了 i 点;}\\ 0, & \text{else.}\end{cases} \tag{7}$$

$$x_{ij}^{vr}=\begin{cases}1, & \text{if 第 v 辆车在第 r 次出行由 i 到 j;}\\ 0, & \text{else.}\end{cases} \tag{8}$$

同时引入 k_v 对车的数量进行统计：

$$k_v=\begin{cases}1, & \text{使用了第 v 辆车;}\\ 0, & \text{else.}\end{cases} \tag{9}$$

目标函数：

$$\min Z=f+f'+\sum_{v\in V}k_vM \tag{10}$$

其中 f 为重载运费，f' 为空载运费。由于购置或租赁车辆的费用较高，因此在满足配送需求的同时，需要车的数量尽可能的少，在此令 M 为极大的数，通过设定极大的代价达到目标。

接下来对约束的意义进行说明。

（1）根据问题的要求以及相关假设，每个需求点上的需求都应该被一辆车在一次出行中一次性满足，因此对于除起点外的任意一点，考查每一辆车的每一次出行，应该有且仅有一次到达点 i，故给出下式：

$$\sum_{v\in V}\sum_{r\in R}^{n}y_i^{vr}=1\quad \forall i\in N\backslash\{20\}. \tag{11}$$

（2）在上式的基础上，对于边的属性进行约束。显然，对于车辆的若干次出行形

成若干个闭环，从各顶点的出发的次数和到达各顶点的次数相同。且除起点由于需要多次经过导致顶点访问次数不等于 1，其余各顶点三者的数值相等，故给出如下约束：

$$\sum_{i\in N}^{n} x_{ip}^{v} r = \sum_{j\in N}^{n} x_{pj}^{vr} \quad \forall p \in N\backslash\{20\}, r \in R, v \in V. \tag{12}$$

针对顶点，仅对其出发和到达的次数进行约束：

$$\sum_{v\in V}^{n} x_{i20}^{v} r = \sum_{v\in V}^{n} x_{20j}^{vr} \quad \forall v \in V, r \in R. \tag{13}$$

（3）上文中已经证明，对于车辆的每次出行，在满足车辆时间的约束之前，其实际载重必然先受到额定载重的约束，因此针对每一辆车的每次出行，其对接的需求点的需求之和必然小于额定载重。考虑到问题三中有两辆车进行选择，额定载荷也应该根据车辆型号的不同而进行设置：

$$\sum_{i\in N\backslash\{0\}}^{n} Q_i y_i^{vr} \leq 6 \quad \forall v \in V, r \in R. \tag{14}$$

（4）对于每一辆车，其总工作时间不能超过四小时。同时考虑下货时间和运输时间，给出如下约束：

$$\frac{1}{12}\sum_{r\in R}^{n}\sum_{i\in N}^{n} y_i^{vr} + \sum_{r\in R}^{n}\sum_{(i,j)\in A}^{n} \frac{a_{ij}x_{ij}^{vr}}{50} \leq 4 \quad \forall v \in V. \tag{15}$$

（5）为打破子环，此处采用和问题一相同的思路，即随着前进的次序对某一指标进行增加，若出现子环，则会出现矛盾的情况。

结合问题中变化的运输费用，考虑到虽然每一段路径上的运输费用都会随着车上货物的重量进行改变，但通过对问题进行分解，实际上对于车上每个需求点的货物所付出的费用仅仅取决于车辆到达该需求点前经过的距离。

因此，可以给定随着次序增加的指标具有的实际含义为：对于某辆车的某一次出行，从起点到需求点 i 已经走过的距离。

首先将起点处的距离初始化为 0：$u_{20}^{vr} = 0, \quad v \in V, r \in R$

按访问的顺序，若 $x_{ij}^{vr} = 1$，则对于该车的某次出行，下一个访问的顶点 j，其对应已经走过的距离 u_j^{vr} 为到达上一个顶点已经走过的距离与 a_{ij} 之和；而当 $x_{ij}^{vr} = 0$，则由于任意常数必然大于 $-M$，该约束变为无效约束。

$$u_j^{vr} - u_i^{vr} \geq (a_{ij} + M)\, x_{ij}^{vr} - M \quad \forall i, j \in N, \forall v \in V, r \in R. \tag{16}$$

（6）在此分为载重和空载两种状态，对各自的运费进行求解。而对于空载状态，易知在且仅在进入起点，即到达终点，的最后一段路程上，车辆处于空载状态，因此考虑所有车辆的每次出行进入起点的路径长度及该车型的空载价格，得到空载状态下的运输费用。

$$f' = \sum_{i\in N}\sum_{v\in V}\sum_{r\in R} x_{i20}^{vr} a_{i20} e_v. \tag{17}$$

对于载重状态，根据上述计算运费的思路，对每个需求点的货物单独计算运输费用：

$$f = 2\sum_{i\in N}\sum_{j\in N}\sum_{v\in V}\sum_{r\in R} u_j^{vr} x_{ij}^{vr} Q_j. \tag{18}$$

上式（18）中，存在 $u_j^{vr}x_{ij}^{vr}$ 的非线性项，使得线性规划模型变为了混合整数二次规划模型，会对模型的求解造成极大的困难。因此本文通过对其进行线性化，保留了线性规划良好的性质。在此将 $x_j^{vr}u_{ij}^{vr}$ 视为一个变量 z_{ij}^{vr}，并通过（19）-（21）替代约束（18）：

$$f = 2\sum_{i\in N}\sum_{j\in N}\sum_{v\in V}\sum_{r\in R} z_{ij}^{vr} Q_j \tag{19}$$

$$\underline{u}x_{ij}^{vr} \le z_{ij}^{vr} \le \overline{u}x_{ij}^{vr} \quad \forall i,j \in N, \forall v \in V, r \in R \tag{20}$$

$$-\overline{u}\left(1-x_{ij}^{vr}\right) + u_j^{vr} \le z_{ij}^{vr} \le \underline{u}\left(1-x_{ij}^{vr}\right) + u_j^{vr} \quad \forall i,j \in N, \forall v \in V, r \in R \tag{21}$$

虽然增添了约束，使得问题规模在一定程度上增加了，但是由于其线性规划的良好形式，实际求解速度明显提升。

（7）统计实际使用车的数量。

$$\sum_{i\in N}\sum_{j\in J}\sum_{r\in R} x_{ij}^{vr} \le k_v M \quad \forall v \in V, r \in R, i,j \in N. \tag{22}$$

（8）由于 y_i^{vr} 和 x_{ij}^{vr} 为 0-1 变量，故有：

$$x_{ij}^{vr} \in \{0,1\} \qquad \forall v \in V, r \in R, (i,j) \in A \tag{23}$$

$$y_i^{vr} \in \{0,1\} \qquad \forall v \in V, r \in R, i \epsilon N \tag{24}$$

5.4 模型求解

考虑到问题的规模较大，不便于直接进行求解。在此希望通过获得局部最优解，并根据其表现对原问题的解空间进行切割，缩小问题的规模后，再对模型进一步求解。

首先对原问题进行观察发现，其对于车辆数量的并没有上限的约束，仅仅是希望车辆数量越少越好，而下限的约束则来源于每个需求点必须都经过一次的硬性要求，这意味着若将原问题中关于车辆的惩罚项以及相关约束删除，求解得到的解也是原问题的可行解。因此，在求解局部最优解时，不考虑车辆数量，仅以最少运输费用作为目标进行求解。

然而，在实际求解过程中，为了保证局部最优解的表现，本文将车辆数量尽可能少转化为每一次运输的货物尽可能多的近似要求。

基于以上，采用了蚁群算法进行求解，该算法的基本原理如下：

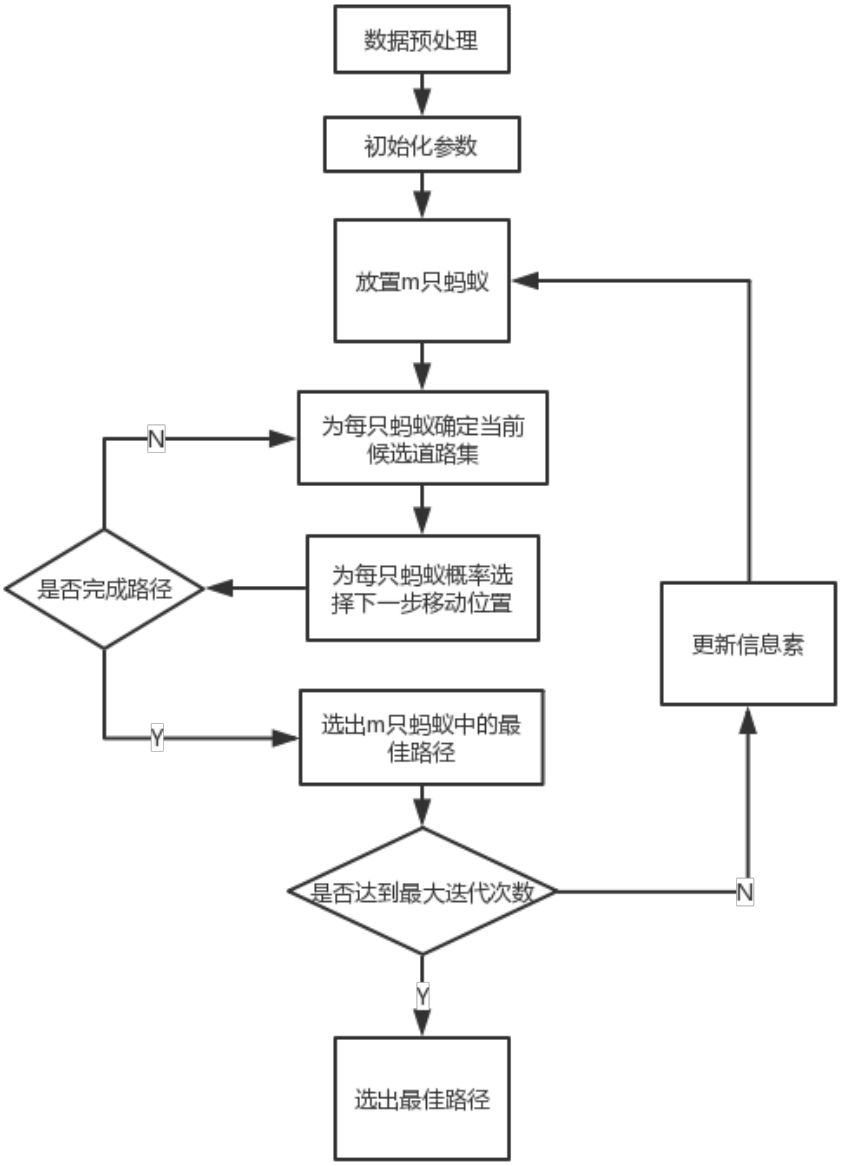

图 4　蚁群算法算法流程图

根据上述说明，并带入到原问题的约束中，发现各约束条件都能够满足，证明此解是原问题的可行解

可以观察到，在此局部最优解中，使用的车的数量为 2。同时，由于优化运输方案改进的费用 $\Delta f < M$，所以可知全局最优解的车辆数量 $V \leq 2$。另外，根据问题一的最短时间运输方案，将其卸货时间从 20 分钟改为目前的 5 分钟，计算得到 1 辆车最少需要 268.8 分钟，大于每辆车的最长使用时间 240 分钟。至此，已经证明使用车的数量 $1 < V \leq 2$，故全局最优解中使用车的数量必然为 2。

通过上述操作，原问题得到了简化。进而通过模拟退火算法对原问题进行求解，该算法的基本原理为：对于某一个初始可行解，生成一个随机的邻解不断寻优，同时根据 Metropolis 准则，有限度的接受恶化解。

Metropolis 准则可以表示如下：

$$p=\begin{cases}1, & \text{if } E(x_{new})<E(x_{old});\\ exp(\frac{E(x_{new})}{E(x_{old})}), & \text{if} E(x_{new})\geq E(x_{old}).\end{cases} \tag{25}$$

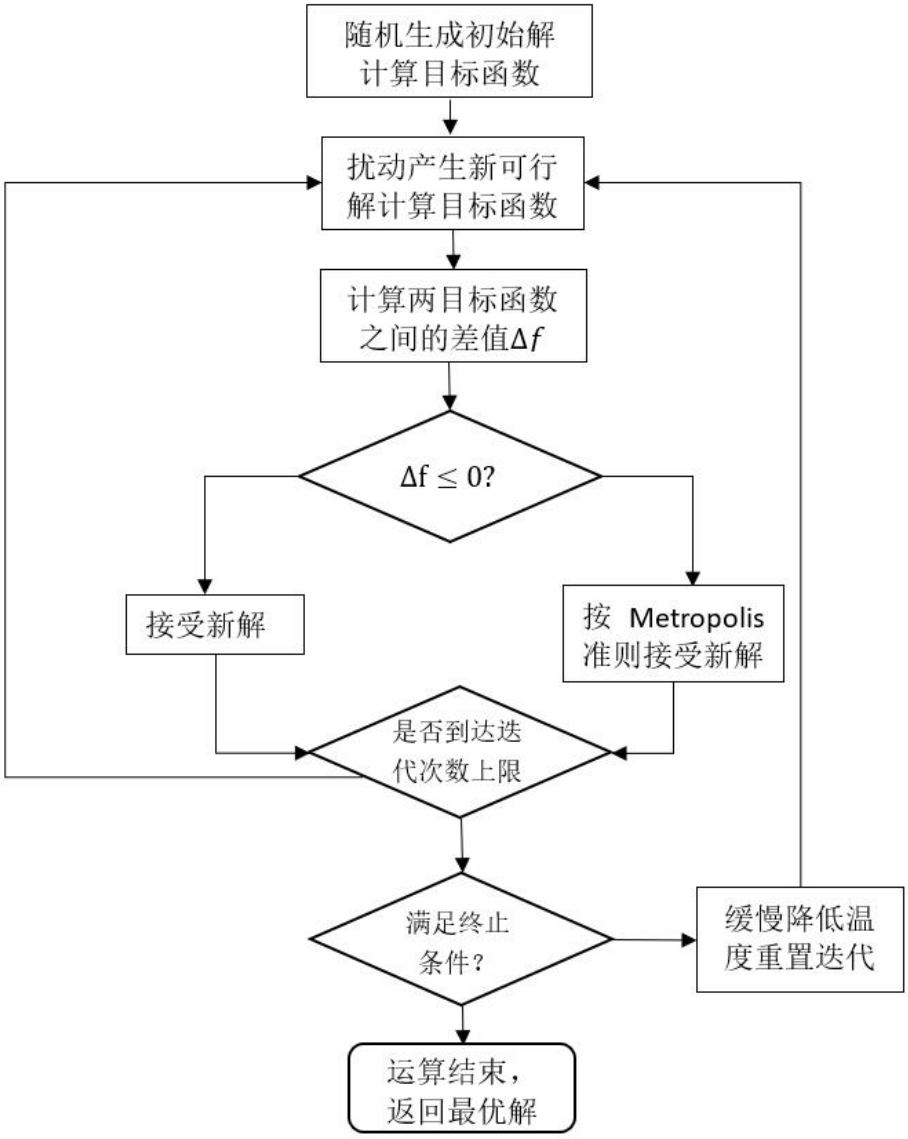

图 5　模拟退火算法流程图

在此根据模型的基本思路，采用模拟退火算法对其进行求解。模拟退火的流程已经在问题二中详细说明，在此仅对模拟退火算法做出的改进进行阐述。本问题中解序列的特殊结构本身可以视为若干个染色体。

将染色体上的每一个基因看作一个站点，一条染色体视作一辆车经过的站点序列，那么对于题目中的 n 辆货车，利用 n 条染色体来进行表示，那么这样即可以表示 n 辆货车的任务分配以及站点的先后顺序。

那么利用模拟退火算法，一开始随机生成一个初始解，再不断产生新解。产生新解的方式包括站点号的随机倒序与相邻染色体间不定长站点序列的交叉。

交叉：对于每条染色体，都可随机与一条非自身的染色体进行某一段序列的交换，即改变两辆货车经过需求点的数量及交换序列对应的需求点。交叉操作可以保证各种

顺序的随机组合都可以出现。交叉概率根据遗传算法中的变异概率设置为 0.2，目的是让每辆车经过的站保持不变时，保证车经过的站的排列顺序尽可能多，保证局部搜索能力。

倒序：对于每一条染色体上的序列，可以随机选取其中的一小段进行倒序操作。

利用以上两种方式产生新的可行解之后，通过计算该可行解下总成本，进而用模拟退火算法的原理对解的最优性进行判别，最后选择输出结果或继续迭代。

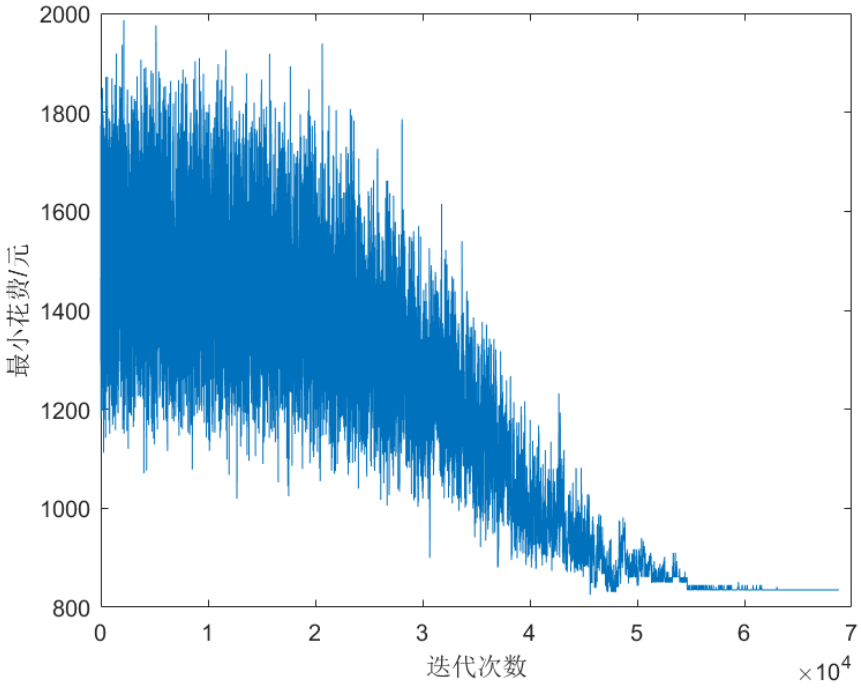

图 6　成本收敛图

结果呈现：最小花费为 835.00 元。第一辆车的用时为 2.83 小时。第二辆车的用时为 3.84 小时。第一辆车的总路程为 108.00 公里。第二辆车的总路程为 146.00 公里。第一辆车路径为:20 → 7 → 4 → 2 → 5 → 20 → 9 → 10 → 17 → 20 → 18 → 20；第二辆车路径为:20 → 6 → 11 → 12 → 20 → 13 → 19 → 14 → 20 → 8 → 3 → 20 → 15 → 16 → 20 → 1 → 20。

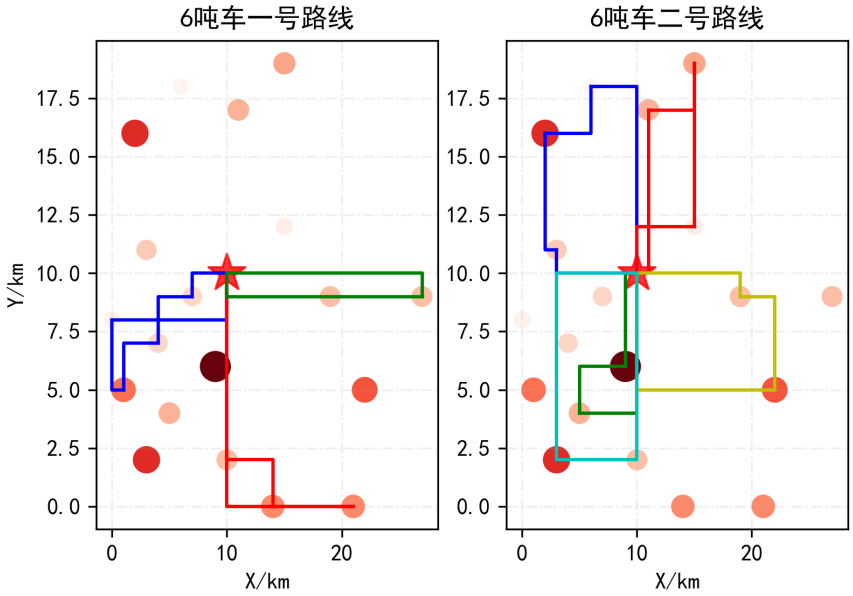

图 7　运输示意图

六、 问题三模型建立与求解

6.1 问题分析

问题三要求现有载重为分别为 6 吨和 4 吨的运输车，空载费用分别为 0.2、0.4 元/公里，其他条件均相同。求送完所有食物的调度效率最高的车辆数和调度方案。。此问题的具体约束与问题二几乎一样，仅仅是增加了车型的选择。又因为在问题二已经证明了车辆数量的必然为 2，所以对于问题三，只需要依照问题二的模型结构，分别讨论载重 4 吨和载重 6 吨的运输车各一辆，两辆载重 4 吨的运输车以及两辆载重 6 吨的运输车三种情况即可。其中，问题二已经对两辆载重 6 吨运输车的情况进行了计算，接下来只需要对其余两者情况计算即可。

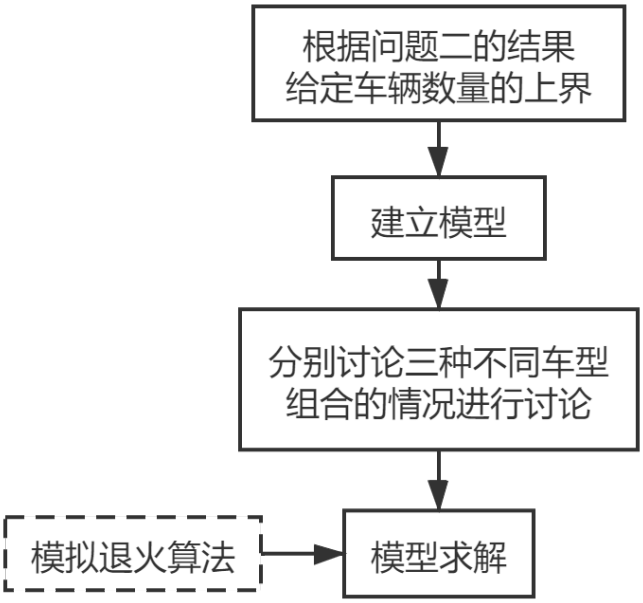

图 8　问题三流程图

6.2 模型建立

由于在本题中，对于车辆每次出行最大承重、车辆最大使用时间以及车辆每次出行访问点的顺序都具有有效的约束。同时由于可以将问题二的结果作为问题三解的上界，所以对于车辆数量以及每辆车的出行次数都有了明确的上界，在此建立了更详细的四参数模型对该过程进行描述。其中，由于车辆的数量已经确定，所以删除了车辆数量的惩罚项及车辆状态的约束方程。另外对于第 v 辆车的额定载重 L_v，在此只需要讨论 $L_1=4, L_2=6;\ L_1=6, L_2=6; L_1=4, L_2=4$ 三种情况。其余各变量及约束的意义与问题二中基本相同。

$$\min Z = f + f' \tag{26}$$

$$s.t.\begin{cases}
\sum_{v\in V}^{n}\sum_{r\in R}^{n} y_i^{vr} = 1 \quad \forall i \in N\backslash\{20\}. \\
\sum_{i\in N}^{n} x_{ip}^{v} r = \sum_{j\in N}^{n} x_{pj}^{vr} \quad \forall p \in N\backslash\{20\}, r \in R, v \in V. \\
\sum_{v\in V}^{n} x_{i20}^{v} r = \sum_{v\in V}^{n} x_{20j}^{vr} \quad \forall v \in V, r \in R. \\
\sum_{i\in N\backslash\{0\}}^{n} Q_i y_i^{vr} \le L_v \quad \forall v \in V, r \in R. \\
\frac{1}{12}\sum_{r\in R}^{n}\sum_{i\in N}^{n} y_i^{vr} + \sum_{r\in R}^{n}\sum_{(i,j)\in A}^{n} \frac{a_{ij}x_{ij}^{vr}}{50} \le 4 \quad \forall v \in V. \\
u_j^{vr} - u_i^{vr} \ge (a_{ij}+M)\, x_{ij}^{vr} - M \quad \forall i,j \in N, \forall v \in V, r \in R. \\
f' = \sum_{i\in N}\sum_{v\in V}\sum_{r\in R} x_{i20}^{vr} a_{i20} e_v \\
f = 2\sum_{i\in N}\sum_{j\in N}\sum_{v\in V}\sum_{r\in R} u_j^{vr} x_{ij}^{vr} Q_j. \\
x_{ij}^{vr} \in \{0,1\} \qquad \forall v \in V, r \in R, (i,j) \in A \\
y_i^{vr} \in \{0,1\} \qquad \forall v \in V, r \in R, i \epsilon N
\end{cases} \tag{27}$$

6.3 模型求解

与第二题的求解方法相似，使用模拟退火算法求解。

分别有如下四种解：

一辆 4 吨运输车与一辆 6 吨运输车；两辆 6 吨运输车；两辆 4 吨运输车。

一辆 6 吨运输车一辆 4 吨运输车的最小花费为 808.80 元。第一辆车的用时为 3.07 小时。第二辆车的用时为 3.76 小时。第一辆车的总路程为 120.00 公里。第二辆车的总路程为 142.00 公里。第一辆车路径为:20 → 6 → 11 → 20 → 13 → 12 → 20 → 8 → 20 → 14 → 19 → 20 → 2 → 20。第二辆车路径为:20 → 7 → 3 → 1 → 20 → 9 → 10 → 17 → 20 → 15 → 16 → 18 → 20 → 4 → 5 → 20。

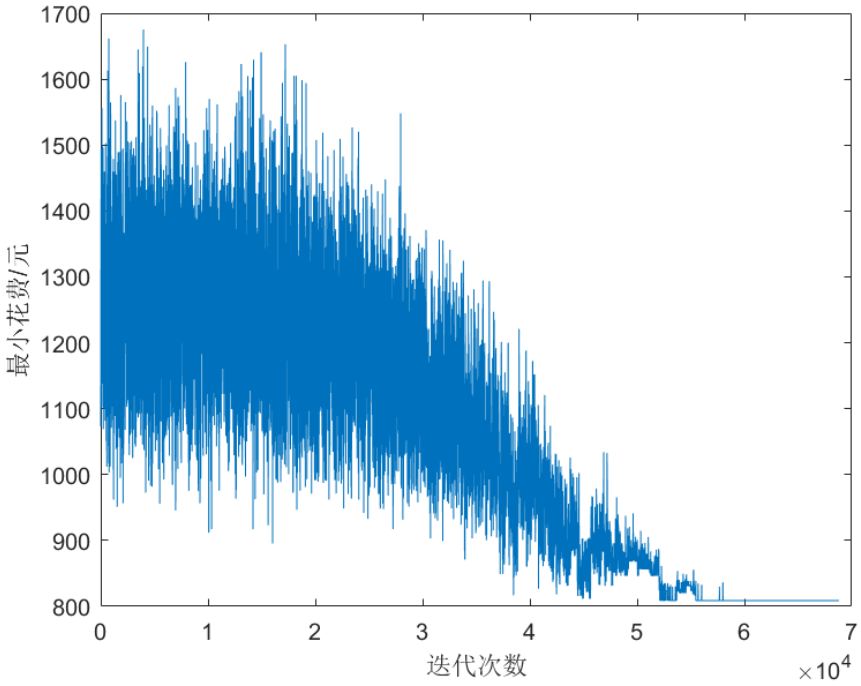

图 9　最小花费收敛折线图

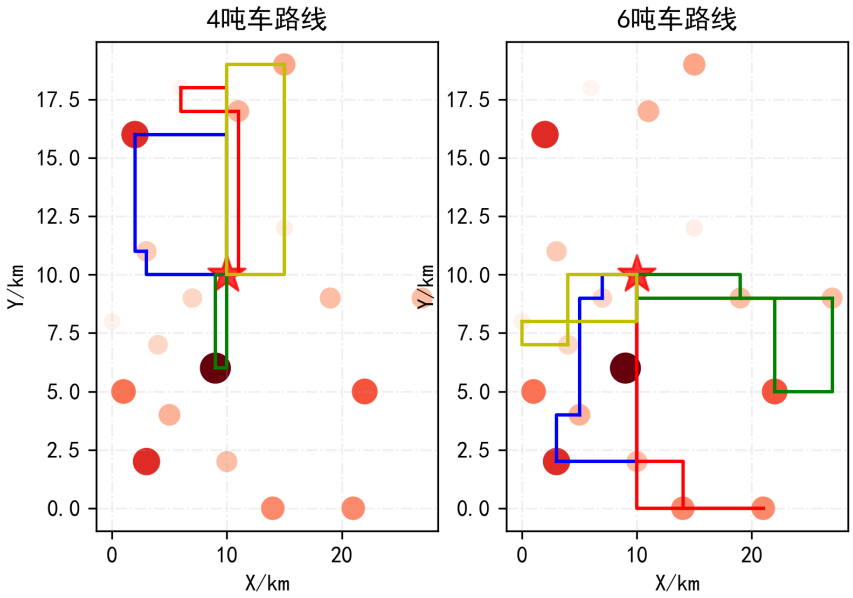

图 10　运输示意图

两辆 6 吨运输车具体方案由问题二所得

两辆 4 吨运输车最小花费为 775.40 元。第一辆车的用时为 3.79 小时。第二辆车的用时为 3.95 小时。第一辆车的总路程为 152.00 公里。第二辆车的总路程为 156.00 公里。第一辆车路径为:20 → 6 → 11 → 20 → 4 → 2 → 20 → 15 → 18 → 20 → 13 → 12 → 20 → 16 → 20 第二辆车路径为:20 → 14 → 19 → 20 → 8 → 20 → 7 → 3 → 20 → 10 → 17 → 20 → 9 → 1 → 20 → 5 → 20

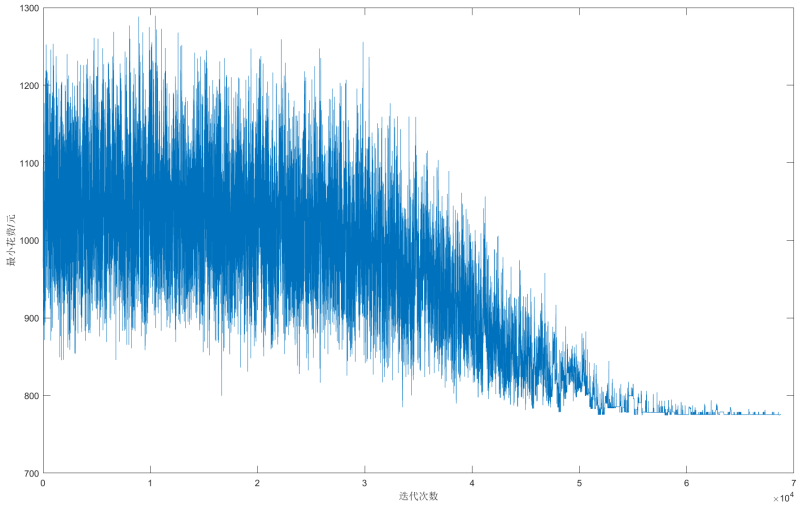

图 11　最小花费收敛折线图

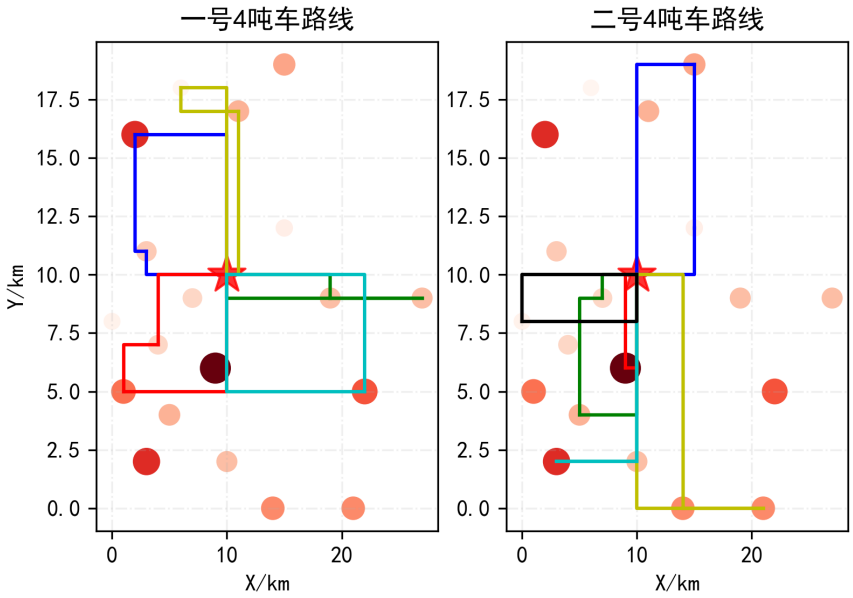

图 12　运输示意图

根据比较，两辆 4 吨运输车的运输成本最低，而且 4 小时内的利用率最高。成本为 775.40 元。

上述结果整理如下表所示：

表 1　运输策略比较表

方案	总成本（元）	总路程（公里）	分别用时（小时）
一辆 4 吨运输车与一辆 6 吨运输车	808.8	242	3.07；3.76
两辆 6 吨运输车	835.00	254	2.83；3.84
两辆 4 吨运输车	775.40	308	3.95；3.79

6.4 结果分析

考虑到实际物流活动中，客户的需求、需求点的坐标等都会发生改变，所以灵敏度分析对此问题的实际意义重大。然而，由于混合 0-1 整数规划的部分变量为离散值，导致无法使用普通的基于单纯形法的灵敏度分析。在此，本文仅讨论在上述最优解的基础上增加一辆车对调度问题的影响，拟通过车辆利用率的变化考查调度系统的稳定性。

对于增加一辆额定载重为 4 吨的情况：

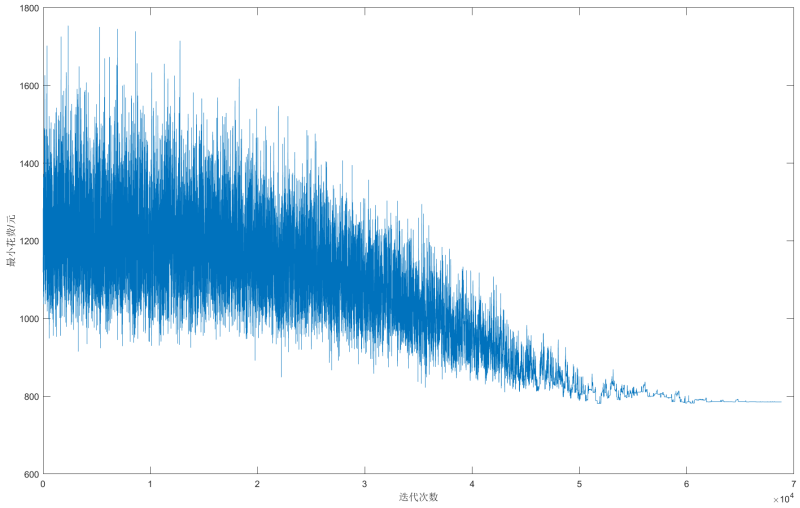

图 13　最小花费收敛折线图

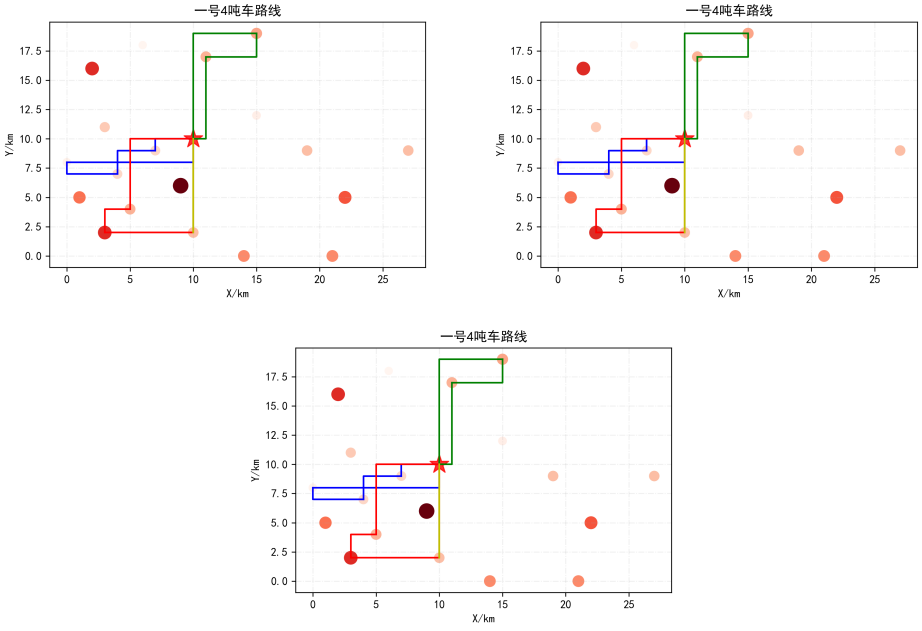

图 14　运输示意图

最小花费为 785.40 元。第一辆车的用时为 2.67 小时。第二辆车的用时为 3.30 小时。第三辆车的用时为 1.93 小时。第一辆车的总路程为 100.00 公里。第二辆车的总路程为 136.00 公里。第三辆车的总路程为 80.00 公里。第一辆车路径为:20 → 17 → 14 → 15 → 120 → 13 → 11 → 120 → 113 → 119 → 120 → 19 → 120; 第二辆车路径为:20 → 16 → 111 → 120 → 115 → 118 → 120 → 12 → 120 → 116 → 120 → 18 → 120; 第三辆车路径为:20 → 114 → 120 → 110 → 117 → 120 → 112 → 120。

对于增加一辆额定载重为 6 吨的情况。

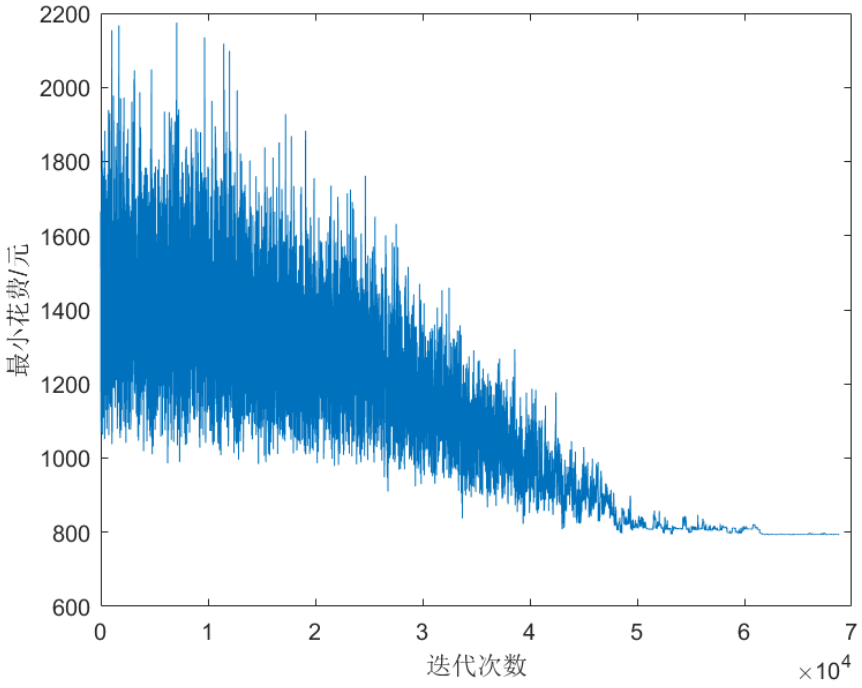

图 15　最小花费收敛折线图

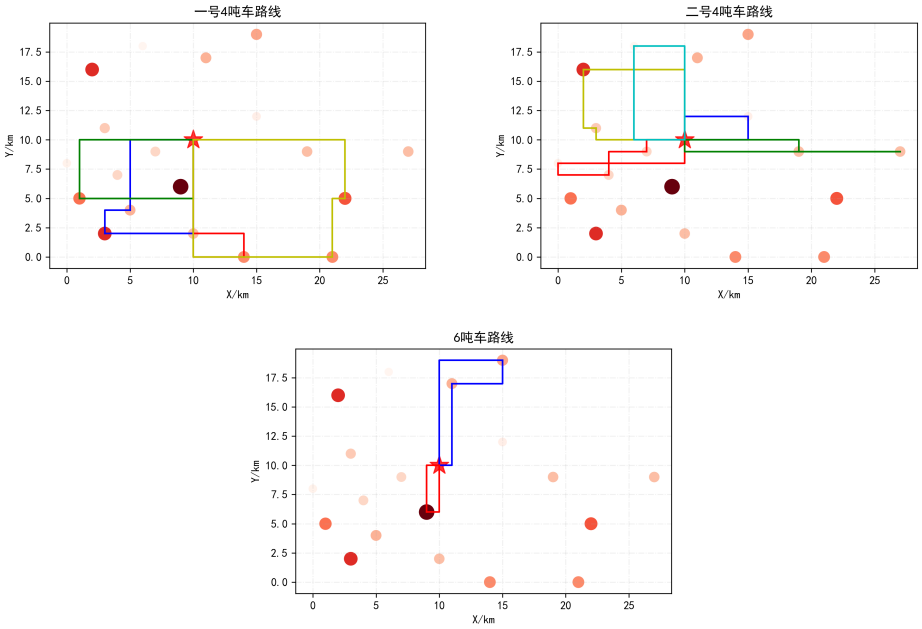

图 16　运输示意图

两辆 4 吨运输车与一辆 6 吨运输车最小花费为 794.20. 第一辆车的用时为 3.18 小时。第二辆车的用时为 1.01 小时。第三辆车的用时为 3.31 小时。第一辆车的总路程为 130.00 公里。第二辆车的总路程为 38.00 公里。第三辆车的总路程为 128.00 公里。第一辆车路径为:20 → 3 → 1 → 20 → 9 → 10 → 20 → 2 → 20 → 16 → 17 → 20。第二辆车路径为:20 → 13 → 19 → 20 → 8 → 20。第三辆车路径为:20 → 14 → 20 → 7 → 4 → 5 → 20 → 15 → 18 → 20 → 6 → 11 → 20 → 12 → 20。

表 2　运输策略比较表

方案	总成本（元）	总路程（公里）	分别用时（小时）	利用率
三辆 4 吨	785.40	316	2.67	66.75%
			3.30	82.5%
			1.93	48.25%
两辆 4 吨和一辆 6 吨	794.20	296	3.18	79.5%
			1.01	79.5%
			3.31	82.75%

相较于原问题两辆车的利用率 98.75% 和 94.75%，增加了一辆车后利用率明显降低，能够应对需求变化较大的情况。

七、 模型总结与评价

7.1 模型优点

1. 模型在尽可能保证问题规模合适的前提下，几乎没有进行省略，详细的描述了调度过程中的各约束条件。

2. 问题二的模型中，将消除子环的约束与计算运费的公式相结合，减少了额外的约束。

3. 问题二的模型中通过上下界的约束，将非线性项转化为线性项，将混合整数非线性规划转化为混合整数线性规划，增加了求解速度。

4. 采用了改进的模拟退火算法，保证结果性质良好的前提下，极大提升了求解速度。

7.2 模型缺点

1. 问题规模较大，计算成本较高。

2. 智能算法所求得的结果具有一定随机性，但由于时间问题，未能获得有效下界保证其精度。

7.3 模型的改进与展望

本模型通过作为复杂约束条件下的路径规划问题，对多种不同约束条件的下的路径规划问题都有着较好的适应性，推广性较强。另外，还可以继续研究应对突发事件的调度模型，使得更贴近生产实际。

本模型是以 Toth 和 Vigo 等人针对多次出行的路径规划问题（VRPMT，Vehicle routing problems with multiple trips）所构建的四参数规划（4-index formulations）模型 [3] 为基础，综合考虑车辆数量不确定等约束进行建立的。而实际上，Azi 等人已经提出了忽略车辆参数的三参数规划模型 [4]；Juan Carlos Rivera 等人提出了更精炼的二参数规划模型 [5]，这些模型对问题的简化程度会更好。但由于本问题中含有对不同车型容量不同以及对每次出行的额定载重限制，需要对不同的车辆和不同的出行次数给予下标，并且由于时间关系，未能想到将其运用在这些简化模型上的方法。若时间允许，可以基于此对模型进一步简化。

参考文献

[1] 张涛, 张玥杰, 王梦光. 不确定车辆数的车辆路径问题模型和混合算法 [J]. 系统管理学报, 2002, 011(002):121-124,130

[2] Cattaruzza, Diego, Feillet,Vehicle routing problems with multiple trips[J]. 4or Quarterly Journal of the Belgian French Italian Operations Research Societies, 2016.

[3] Golden B , Assad A , Levy L , et al. The fleet size and mix vehicle routing problem[J]. Computers Operations Research, 1984, 11(1):49-66.

[4] Azi N , Gendreau M , Potvin J Y . An exact algorithm for a single-vehicle routing problem with time windows and multiple routes[J]. European Journal of Operational Research, 2007, 178(3):755-766.

[5] Juan Carlos Rivera, H. Murat Afsar, Christian Prins. Multistart Evolutionary Local Search for a Disaster Relief Problem[M]// Artificial Evolution. Springer International Publishing, 2013.

附录 A 代码

A.1 规划解决程序 –lingo 源代码

```
MODEL:
SETS:
city/A1..A20/:U;
links(city,city):distance,X;
ENDSETS
DATA:
distance=
0  5  4  6  9  9  11 10 7  13 15 19 23 22 23 22 20 31 29 15
5  0  5  5  4  8  10 9  12 18 12 18 22 21 22 21 25 30 28 14
4  5  0  4  9  9  7  6  7  13 15 15 19 18 19 18 20 27 25 11
6  5  4  0  5  5  5  6  11 17 11 13 17 16 17 20 24 25 23 9
9  4  9  5  0  6  8  11 16 22 10 16 20 19 20 25 29 28 26 12
9  8  9  5  6  0  6  11 16 22 6  10 14 13 18 25 29 26 20 8
11 10 7  5  8  6  0  5  10 16 12 10 12 11 12 19 23 20 18 4
10 9  6  6  11 11 5  0  5  11 17 15 13 12 13 14 18 21 19 5
7  12 7  11 16 16 10 5  0  6  22 20 16 15 16 15 13 24 22 8
13 18 13 17 22 22 16 11 6  0  28 26 20 13 14 13 7  22 20 14
15 12 15 11 10 6  12 17 22 28 0  6  10 17 24 31 35 32 16 14
19 18 15 13 16 10 10 15 20 26 6  0  6  15 22 29 33 30 10 12
23 22 19 17 20 14 12 13 16 20 10 6  0  9  16 23 27 24 6  8
22 21 18 16 19 13 11 12 15 13 17 15 9  0  7  14 18 15 7  7
23 22 19 17 20 18 12 13 16 14 24 22 16 7  0  7  11 8  14 10
22 21 18 20 25 25 19 14 15 13 31 29 23 14 7  0  6  9  21 17
20 25 20 24 29 29 23 18 13 7  35 33 27 18 11 6  0  15 25 21
31 30 27 25 28 26 20 21 24 22 32 30 24 15 8  9  15 0  22 18
29 28 25 23 26 20 18 19 22 20 16 10 6  7  14 21 25 22 0  14
15 14 11 9  12 8  4  5  8  14 14 12 8  7  10 17 21 18 14 0;
ENDDATA
n=@SIZE(city);
MIN=@SUM(links:distance*X);
@FOR(city(k):
@SUM(city(i)|i#ne#k:x(i,k))=1;
@SUM(city(j)|j#ne#k:x(k,j))=1;
@FOR(city(j)|j#gt#1 #and#
    j#ne#k:U(j)>=U(k)+X(k,j)-(N-2)*(1-X(k,j))+(N-3)*X(j,k)););
@FOR(links:@BIN(x));
@FOR(city(k)|k#gt#1:
U(k)<=N-1-(N-2)*X(1,k);
U(k)>=1+(N-2)*X(k,1));
```

A.2 蚁群算法 –matlab 源程序

```
function [R_best,L_best,L_ave,Shortest_Route,Shortest_Length]=ANT_VRP1

(D,Demand,Cap,iter_max,m,Alpha,Beta,Rho,Q)
```

```
%% R_best 各代最佳路线
%% L_best 各代最佳路线的长度
%% L_ave 各代平均距离
%% Shortest_Route 最短路径
%% Shortest_Length 最短路径长度
%% D 城市间之间的距离矩阵，为对称矩阵
%% Demand 客户需求量
%% Cap 车辆最大载重
%% iter_max 最大迭代次数
%% m 蚂蚁个数
%% Alpha 表征信息素重要程度的参数
%% Beta 表征启发式因子重要程度的参数
%% Rho 信息素蒸发系数
%% Q 信息素增加强度系数

n=size(D,1);
T=zeros(m,2*n);          %装载距离
Eta=ones(m,2*n);         %启发因子
Tau=ones(n,n);           %信息素
Tabu=zeros(m,n);         %禁忌表
Route=zeros(m,2*n);      %路径
L=zeros(m,1);            %总路程
L_best=zeros(iter_max,1); %各代最佳路线长度
R_best=zeros(iter_max,2*n); %各代最佳路线
nC=1;

while nC<=iter_max               %停止条件
   Eta=zeros(m,2*n);
   T=zeros(m,2*n);
   Tabu=zeros(m,n);
   Route=zeros(m,2*n);
   L=zeros(m,1);
   Cost = zeros(m,1);
   %初始化起点城市（禁忌表）%%%%%==================================
   for i=1:m
      Cap_1=Cap;    %最大装载量
      j=1;
      j_r=1;
      while Tabu(i,n)==0
         T=zeros(m,2*n); %装载量加载矩阵
         Tabu(i,1)=20;     %禁忌表起点位置为1
         Route(i,1)=20;    %路径起点位置为1
         visited=find(Tabu(i,:)>0); %已访问城市
         num_v=length(visited);    %已访问城市个数
         J=zeros(1,(n-num_v));     %待访问城市加载表
         P=J;                      %待访问城市选择概率分布
         Jc=1;                     %待访问城市选择指针
         for k=1:n                 %城市
            if length(find(Tabu(i,:)==k))==0 %如果不是已访问城市代号，就将加入
                矩阵中kkJ
               J(Jc)=k;
               Jc=Jc+1;
```

```
            end
        end

        %每只蚂蚁按照选择概率遍历所有城市%%%%%%==============================

        for k=1:n-num_v           %待访问城市
            if Cap_1-Demand(J(k))>=0 %如果车辆装载量大于待访问城市需求量

                if Route(i,j_r)==20       %如果每只蚂蚁在起点城市
                    T(i,k)=D(20,J(1,k));
                    P(k)=(Tau(20,J(1,k))^Alpha)*((1/T(i,k))^Beta); %概率计算公式
                        中的分子
                else                      %如果每只蚂蚁在不在起点城市
                    T(i,k)=D(Tabu(i,j),J(1,k));
                    P(k)=(Tau(Tabu(i,visited(end)),J(1,k))^Alpha)*((1/T(i,k))^Beta);
                     %概率计算公式中的分子
                end

            else            %如果车辆装载量小于待访问城市需求量
                T(i,k)=0;
                P(k)=0;
            end
        end


        if length(find(T(i,:)>0))==0 %当车辆装载量小于待访问城市时，选择起点
            为%%1
            Cap_1=Cap;
            j_r=j_r+1;
            Route(i,j_r)=20;
            L(i)=L(i)+D(20,Tabu(i,visited(end)));
        else
            P=P/(sum(P));             %按照概率原则选取下一个城市
            Pcum=cumsum(P);           %求累积概率和：（cumsum[1 2 3])=1 3 目的
                在于使得6,的值总有大于的数Pcumrand
            Select=find(Pcum>rand);   %按概率选取下一个城市：当累积概率和大于给
                定的随机数，则选择求和被加上的最后一个城市作为即将访问的城市
            o_visit=J(1,Select(1));   %待访问城市
            j=j+1;
            j_r=j_r+1;
            Tabu(i,j)=o_visit;         %待访问城市
            Route(i,j_r)=o_visit;
            Cap_1=Cap_1-Demand(o_visit); %车辆装载剩余量
            L(i)=L(i)+T(i,Select(1));   %路径长度
        end
    end
    L(i)=L(i)+D(Tabu(i,n),20);          %路径长度%

    % 求解当前路径的费用
    temp_route = Route(i,find(Route(i,:))>0);
    len = 0;
    for kk = 1:length(temp_route)
```

```
        if temp_route(kk) == 20
           Cost(i) = Cost(i) +
               2*D(temp_route(kk+1),20)*Demand(temp_route(kk+1));
           len = len + D(20,temp_route(kk+1));
           continue;
        end
        if kk == length(temp_route)
           Cost(i) = Cost(i) + 0.4*D(temp_route(kk),20);
           continue;
        end
        if temp_route(kk+1) == 20
           len = 0;
           Cost(i) = Cost(i) + 0.4*D(temp_route(kk),20);
           continue;
        end
        len = len + D(temp_route(kk),temp_route(kk+1));
        Cost(i) = Cost(i) + 2*len*Demand(temp_route(kk+1));
      end
   end

   L_best(nC)=min(Cost);          %最优路径为距离最短的路径
   pos=find(Cost==min(Cost));      %找出最优路径对应的位置：即为哪只蚂蚁
   R_best(nC,:)=Route(pos(1),:); %确定最优路径对应的城市顺序

   Delta_Tau=zeros(n,n);        %Delta_Tau(i,j)表示所有蚂蚁留在第个城市到第个城
       市路径上的信息素增量ij
   L_zan=L_best(1:nC,1);
   post=find(L_zan==min(L_zan));
   Cities=find(R_best(nC,:)>0);
   num_R=length(Cities);

   for k=1:num_R-1       %建立了完整路径后在释放信息素
      Delta_Tau(R_best(nC,k),R_best(nC,k+1))=Delta_Tau(R_best(nC,k),
      R_best(nC,k+1))+Q/L_best(nC);
   end
   Delta_Tau(R_best(nC,num_R),1)=Delta_Tau(R_best(nC,num_R),1)+Q/L_best(nC);
   Tau=Rho*Tau+Delta_Tau;

   nC=nC+1;
end
[Shortest_Length,pos] = min(L_best)
Shortest_Route=zeros(1,2*n);      %提取最短路径
Shortest_Route(1,:)=R_best(pos,:);
Shortest_Route=Shortest_Route(Shortest_Route>0);
Shortest_Route=[Shortest_Route Shortest_Route(1,1)];
%Shortest_Length=L_best(iter_max); 提取最低费用    %
```

A.3 模拟退火算法 –matlab 源程序

```
% 求解的主函数
```

```
clear; clc;
load('distancematrix')
dis = dist;
temperature = 1000; % 初始温度
cooling_rate = 0.99; % 冷却率

% 给出初始解
route{1} = [1:7];
route{2} = [8:19];
[previous_where20,previous_distance] = totaldistance(route, dis, z);
[~,previous_croute1]=completeroute(route{1},previous_where20{1});
[~,previous_croute2]=completeroute(route{2},previous_where20{2});
[~,previous_time1]=computeTime(previous_croute1,dis,1);
[~,previous_time2]=computeTime(previous_croute2,dis,2);
previous_time=previous_time1+previous_time2;
temperature_iterations = 1;
no = 0;
no1 = 0;
iteration_cnt = 1;

while 1.0 < temperature
   temp_route{1} = perturb(route{1},'reverse', dis, previous_where20{1}, 1);
   temp_route{2} = perturb(route{2}, 'reverse', dis, previous_where20{2}, 2);
   [current_where20,current_distance] = totaldistance(temp_route, dis, z);
   [route_new11, routenew12]=completeroute(temp_route{1},current_where20{1});
   [route_new21, routenew22]=completeroute(temp_route{2},current_where20{2});
   [k11,k12]=computeTime(routenew12,dis,1);
   [k21,k22]=computeTime(routenew22,dis,2);
   % 判断路径是否满足小于小时的约束4
   if k11 ==1 && k21==1
      diff = current_distance - previous_distance;
      fprintf('diff=%d\ntemp=%d', diff, temperature);
      if (diff < 0) || (rand < exp(-diff/(temperature)))
         route = temp_route;
         previous_distance = current_distance;
         previous_time1=k12;
         previous_time2=k22;
         previous_time=k12+k22;
         low_cost(iteration_cnt) = previous_distance;
         iteration_cnt = iteration_cnt + 1;
         previous_where20 = current_where20;
         [xxx,nnew_route{1}]=completeroute(route{1},previous_where20{1});
         [xxx,nnew_route{2}]=completeroute(route{2},previous_where20{2});
         for i =1:2
            [x,y]=computeTime(nnew_route{i},dis,i)
            if x == 0
               no = no + 1;
            end
         end
         temperature_iterations = temperature_iterations + 1;
      end
```

```
    end
    if rand > 0.8
       temp_route1 = crossover(route);
       [current_where20,current_distance] = totaldistance(temp_route1, dis,
          z);
       [route_new11,
          routenew12]=completeroute(temp_route1{1},current_where20{1});
       [route_new21,
          routenew22]=completeroute(temp_route1{2},current_where20{2});
       [k11,k12]=computeTime(routenew12,dis,1);
       [k21,k22]=computeTime(routenew22,dis,2);
       % 判断路径是否满足小于小时的约束4
       if computeTime(routenew12,dis,1)~=0 && computeTime(routenew22,dis,2)~=0
          diff = current_distance - previous_distance;
             if (diff < 0) || (rand < exp(-diff/(temperature)))
                route = temp_route1;
                previous_time1=k12;
                previous_time2=k22;
                previous_time=k12+k22;
                previous_distance = current_distance;
                low_cost(iteration_cnt) = previous_distance;
                iteration_cnt = iteration_cnt + 1;
                previous_where20 = current_where20;
                [xxx,nnew_route{1}]=completeroute(route{1},previous_where20{1});
                [xxx,nnew_route{2}]=completeroute(route{2},previous_where20{2});
                for i =1:2
                   [x,y]=computeTime(nnew_route{i},dis,i)
                   if x == 0
                      no1 = no1 + 1;
                   end
                end
                temperature_iterations = temperature_iterations + 1;
             end
       end
    end
    if temperature_iterations >= 100
       temperature = cooling_rate * temperature;
       temperature_iterations = 0;
    end
end

[xxx,new_route{1}]=completeroute(route{1},previous_where20{1});
[xxx,new_route{2}]=completeroute(route{2},previous_where20{2});

%new_route{1}
%new_route{2}
%作费用收敛折线图%
plot(1:length(low_cost),low_cost);
xlabel('迭代次数'); ylabel('最小花费');
for i =1:2
    [x(i),y(i)]=computeTime(new_route{i},dis,i);
```

```
end
fprintf最小花费为("%.2f\n",previous_distance);

fprintf第一辆车的用时为("%.2f\n",y(1));
fprintf第二辆车的用时为("%.2f\n",y(2));
len = zeros(1,2);
for k = 1:2
   for i=1:length(new_route{k})-1
      len(k) = len(k) + dis(new_route{k}(i),new_route{k}(i+1));
   end
end
fprintf第一辆车的总路程为("%.2f\n",len(1));
fprintf第二辆车的总路程为("%.2f\n",len(2));
fprintf第一辆车路径为(":");
for i = 1:length(new_route{1})-1
   fprintf("%d\rightarrow1",new_route{1}(i));
end
fprintf("%d\n",new_route{1}(end));

fprintf第二辆车路径为(":");
for i = 1:length(new_route{2})-1
   fprintf("%d\rightarrow1",new_route{2}(i));
end
fprintf("%d\n",new_route{2}(end));
```