参赛密码 _____

（由组委会填写）

第十一届华为杯全国研究生数学建模竞赛

学　　校	上海理工大学
参赛队号	**10252192**
队员姓名	**1.**周成明
	2.鲁　俊
	3.刘子华

参赛密码 _____

（由组委会填写）

第十一届华为杯全国研究生数学建模竞赛

题 目 **乘用车物流运输计划问题**

摘 要：

本文依据物流公司的运载要求，通过建立数学模型，为物流公司安排了五次运输计划。此外，本文对同类型运输问题建立了通用运输模型，并设计了相关算法，采用 C 与 C#语言混合编程对模型进行求解与优化。最后，论文还使用了空间利用率、轿运车配比、总行驶路程指标全面了评估了模型效果。

针对问题一二三，论文首先建立了带约束条件的多目标数学模型。论文依据目标优先级分二阶段逐步求解。第一阶段，使用尽量少的轿运车。主要涉及排样算法，轿运车配比策略，基于空间利用率与约束条件的最优装载策略。第二阶段，通过建立结果判断器，优化 1-1 型与 1-2 型轿运车配比，使得在使用相同轿运车情况下，优先使用 1-1 型成本较小车辆。下问题一、二、三的最终求解结果如下：

	问题一	问题二	问题三
1-1 型车使用量	16	12	25
1-2 型车使用量	2	1	5
总空间利用率	96.2%	97.3%	96.6%

对问题四，在问题一二三通用模型的基础上建立运输路径优化算法，模型采用远距离节点优先调度算法，逐步完成各节点配送任务。由于随着物流节点增多时，算法复杂度急剧增加，论文设计了基于遗传算法的启发式算法，寻找复杂物流网络下的较优配送方案。模型求解结果如下：

	A	B	C	D	合计
1-1 型车使用量	7	4	7	3	21
1-2 型车使用量	1	1	1	1	4
空间利用率	94.3%	97.2%	95.2%	93.4%	--
总行驶里程	2880	1400	1888	640	6408

针对问题五，主要分为以下三步求解。第一步，数据预处理。用分类与排序算法过滤掉装载效果一定很差的装载方案。第二步，建模寻优。论文建立了基于禁忌搜索算法的配载优化模型，经数据预处理的较优可行解中利用模型逐步寻优，达到模型终止条件后，将产生的随机解中最优解作为模型输出。第三步，综合优化。建立了基于计算空间利用率的寻未装满轿运车算法，通过调配开往 ABCDE 中未装满轿运车装载方案，进一步优化结果。实验结果表明，该算法的计算效率较高，收敛速度较快，计算结果也较稳定。

目的地	A	B	C	D	E	合计
轿运车数量	28	22	24	21	21	116
空间利用率	93.3%	96.2%	92.1%	90.2%	93.1%	--
总行程数	10080	6160	6608	3360	8064	34272

关键词：多目标　配载优化　　路径优化　启发式算法　禁忌搜索算法　C 编程

目 录

一、 问题重述

物流公司在确保完成运输任务的前提下，物流公司追求降低运输成本。运输中装载具体要求如下：每种轿运车上、下层装载区域均可等价看成长方形，各列乘用车均纵向摆放，相邻乘用车之间纵向及横向的安全车距均至少为 0.1 米，下层力争装满，上层两列力求对称，以保证轿运车行驶平稳。受层高限制，高度超过 1.7 米的乘用车只能装在 1-1、1-2 型下层。轿运车、乘用车规格（第五问见附件）如下：

乘用车型号	长度(米)	宽度(米)	高度(米)
Ⅰ	4.61	1.7	1.51
Ⅱ	3.615	1.605	1.394
Ⅲ	4.63	1.785	1.77

表 1 乘用车规格

轿运车类型	上下层长度(米)	上层宽度(米)	下层宽度(米)
1-1	19	2.7	2.7
1-2	24.3	3.5	2.7

表 2 轿运车规格

整车物流的运输成本计算较为繁杂,这里简化为：影响成本高低的首先是轿运车使用数量；其次，在轿运车使用数量相同情况下，1-1 型轿运车的使用成本较低，2-2 型较高，1-2 型略低于前两者的平均值，但物流公司 1-2 型轿运车拥有量小，为方便后续任务安排，每次 1-2 型轿运车使用量不超过 1-1 型轿运车使用量的 20%;再次,在轿运车使用数量及型号均相同情况下,行驶里程短的成本低，注意因为该物流公司是全国性公司，在各地均会有整车物流业务，所以轿运车到达目的地后原地待命，无须放空返回。最后每次卸车成本几乎可以忽略。

请为物流公司安排以下五次运输，制定详细计划，含所需要各种类型轿运车的数量、每辆轿运车的乘用车装载方案、行车路线。（前三问目的地只有一个，可提供一个通用程序；后两问也要给出启发式算法的程序，优化模型则更佳）：

1． 物流公司要运输Ⅰ车型的乘用车 100 辆及Ⅱ车型的乘用车 68 辆。

2． 物流公司要运输Ⅱ车型的乘用车 72 辆及Ⅲ车型的乘用车 52 辆。

3． 物流公司要运输Ⅰ车型的乘用车 156 辆、Ⅱ车型的乘用车 102 辆及Ⅲ车型的乘用车 39 辆。

4．物流公司要运输 166 辆Ⅰ车型的乘用车（其中目的地是 A、B、C、D 的分别为 42、50、33、41 辆）和 78 辆Ⅱ车型的乘用车（其中目的地是 A、C 的，分别为 31、47 辆），具体路线见图 4，各段长度：OD=160，DC=76，DA=200，DB=120，BE=104，AE=60。

5. 附件的表 1 给出了物流公司需要运输的乘用车类型（含序号)、尺寸大小、数量和目的地，附件的表 2 给出可以调用的轿运车类型（含序号)、数量和装载区域大小（表里数据是下层装载区域的长和宽,1-1 型及 2-2 型轿运车上、下层装载区域相同；1-2 型轿运车上、下层装载区域长度相同，但上层比下层宽 0.8 米。此外 2-2 型轿运车因为层高较低,上、下层均不能装载高度超过 1.7 米的乘用车。

二、模型假设

1、相邻乘用车之间纵向及横向的安全车距均至少为 0.1 米，但是承运车与轿运车前后左右距离无需考虑。
2、针对一二三四四问仅有 1-1 型、1-2 型轿运车可调度。
3、不考虑汽车到站时中间加装或卸货的成本。
4、在满足约束条件下，同一辆轿运车上 I 型、II 型、III 型车可以混装。
5、轿运车载重量大，可行运载方案中不存在超载问题。
6、轿运车长度需严格大于车身长度以及所需最短车身距离。
7、对问题四，汽车允许从中途卸货再开往其他节点，但是不允许中途再上货进行货物调用。
8、问题四中的物流配运起点为 0。
9、对问题五，附件中列出的轿运车为可调用车辆，无需考虑 1-2 型轿运车使用量不超过 1-1 型轿运车使用量的 20%的约束条件。

三、基本符号说明

n_i　需运输汽车中第 i 类汽车的辆数，i=1, 2, 3。1、2、3 类汽车分别对应 I、II、III 型汽车。

N_i　i 型轿运车使用量，i=1, 2, 3。 1、2、3 型轿运车分别对应 1-1, 1-2, 2-2 型轿运车

X_1　I 型承运车需运输数量

X_2　II 型承运车需运输数量

X_3　III 型承运车需运输数量

$X_{ij} = \{X_i|\mathrm{i} = 1,2,3\} = \{X_{ij}|X_{ij} \in X_i; i = 1,2,3; j = 1,2 \cdots, n_i\}$　i 表示汽车所属的类号，j 表示汽车在所属汽车类中的件号

F_1　1-1 型轿运车使用数量

F_2　1-2 型轿运车使用数量

F_3　2-2 型轿运车使用数量

$F_{ij} = \{F_{ij} | i = 1,2,3; j = 1,2,\cdots,m\}$　表示 i 类轿运车中的 j 号轿运车

XL_{ij} 需运输车X_{ij}的长度

XW_{ij} 需运输车辆X_{ij}的宽度

D_{ij}　i 型轿运车中编号为 j 的轿运车行驶距离

Δ_w　　两相邻车辆之间的最小距离

FL_{ij}　　i 型轿运车编号为 n 轿运车的长度

$FW_{ij}mn$　m 型轿运车编号为 n 的轿运车宽度, 该车运载车辆X_{ij}

$FW_{上ijmn}$　m 型轿运车编号为 n 的轿运车上层宽度，该车运载车辆X_{ij}

$FW_{下ijmn}$　m 型轿运车编号为 n 的轿运车下层宽度，该车运载车辆X_{ij}

d_{ij}　　i 与 j 两点间的距离

$$\omega_{ijmn} = \begin{cases} 1(\text{用车辆}F_{mn}\text{承载汽车}X_{ij}\text{时}) \\ 0\text{（否则）} \end{cases}$$

$$\alpha_{ijmn} = \begin{cases} 1(\text{用车辆}F_{mn}\text{承载汽车}X_{ij}\text{时且装在下层时}) \\ 0\text{（否则）} \end{cases}$$

$$\beta_{ijmn} = \begin{cases} 1(\text{用车辆}F_{mn}\text{承载汽车}X_{ij}\text{时且装在上层时}) \\ 0\text{（否则）} \end{cases}$$

$$Y_{ijmn} = \begin{cases} 1, \text{轿运车}F_{mn}\text{由 i 使向 j} \\ 0, \text{轿运车}F_{mn}\text{不由 i 使项 j} \end{cases}$$

$$y_{imn} = \begin{cases} 1，\text{车辆}F_{mn}\text{向 i 客户运车} \\ 0，\text{车辆}F_{mn}\text{不向 i 客户运车} \end{cases}$$

四、问题分析

4.1 问题一二三分析

物流规划的目的是保证在完成配送任务的前提下，使得物流配送的成本最低。影响物流成本因素繁杂，但影响成本高低的首要因素是轿运车使用数量；其次，在轿运车使用数量相同情况下，优先使用 1-1 型成本相对较低轿运车。再次，才是在轿运车使用数量及型号均相同情况下，行驶里程短的成本低。

由于问题一二三的目的地只有一个，前三问的模型目标是在约束条件下使得总轿运车数量最少，以及在轿运车数量一定时，优先使用成本小型轿运车。

针对该问题，可以依据目标优先级分阶段逐步求解，最后再综合优化装载方案。首先，依据不同型号轿运车配比约束，确定轿运车配比安排策略。再在约束条件下，优先使用装载量大的轿运车，即优先装载 1-2 型车。接着，确定承运车装车策略，该步骤又可分为有约束承运车装载策略，无约束车装载策略，通过策略最大化装车后轿运车的空间利用率。最后，再对前三步装载策略进行综合，进一步优化装载策略，逐步达到模型目标。

4.2 问题四分析

对问题四，如果先综合考虑完成 ABCD 点的任务，直接计算完成总任务轿运车最少所需要的类型与数量，会复杂化轿运车的配送安排。会出现一辆轿运车同时配送多个点状况，会使得有些轿运车总行驶距离增大。故采用先分别考虑 A、B、C、D 点任务完成，再综合调节装载与运输方案，使得完成任务前提下轿运车使用数量最少，并且总行驶里程最少。针对该问题可以采取远距离节点优先调度方法设计运载方案，并通过中途卸货方式，减少轿运车数量以及缩短轿运车总行程距离。由于物流公司为全国性公司，随着物流节点增加，对精确求解模型的求解时间开销将增大，故有必要设计启发式算法，解决物流节点和网络增多情况下轿运车装载计划问题。

4.3 问题五分析

对第五问，轿运车类型与数量以及承运车类型与数量相对应前四问有大幅提升。运输计划的装载方案太多，再依据问题一二三四模型，求解最优模型变得不切实际。故应该采用启发式算法，逐步寻优。

具体求解过程可以分为三步。第一步，对原始数据预处理。基于分类与排序算法建立一套可行解过滤规则，过滤掉明显装载效果不好的装载方案。第二步，对 A、B、C、D、E 点单独考虑。采用启发式算法，输入经过优化处理的可行解，从优化数据中随机产生装载方案，经过多次迭代后，选择其中最优解作为装载方案。第三步，综合分析 A、B、C、D、E 点装载方案，通过仍有运载能力的车装满，并中途卸货给其他节点来减少轿运车数量及总行程数。

五、模型的建立与求解

5.1 问题一二三模型的建立与求解

5.1.1 模型建立

对问题一二三，可以调度的轿运车仅有 1-1,与 1-2 两种型号。在确保完成任务的前提下，影响运输成本高低的首要因素是轿运车使用数量；其次，在轿运车使用数量相同情况下，1-1 型轿运车的使用成本较低，2-2 型较高，1-2 型略低于前两者的平均值（由于物流公司 1-2 型轿运车拥有量小，为方便后续任务安排，每次 1-2 型轿运车使用量不超过 1-1 型轿运车使用量的 20%）；再次，在轿运车使用数量及型号均相同情况下，行驶里程短的成本低，因为问题一二三所涉及的目的地一致，故只需考虑前两项成本。并且是优先求出最小轿运车数量，在轿运车数量相同情况下，优先使用 1-1 型轿运车。

故针对问题一二三可建立如下运输成本优化问题的数学模型：

目标函数

第一目标：使用尽量少的轿运车完成任务。

$$\mathrm{Min\ Z} = (F_1 + F_2)$$

第二目标：再使用轿运车数量相同前提下，优先使用 1-1 型轿运车。

$$\begin{cases} F_1 + F_2 = \mathrm{Z} \\ \min(F_2) \\ \max(F_1) \end{cases}$$

约束条件：

（1）1-2 型轿运车不超过 1-1 型轿运车使用量 20%

$$0 \le F_2 \le 0.2F_1$$

（2）轿运车承载车辆长度限制

下层长度限制

$$\sum_{i=1}^{3}\left[\left(\sum_{j=1}^{n_1}\alpha_{\mathrm{ijmn}}\right) * (\mathrm{XL}_{ij} + 0.1)\right] \le F\mathrm{L}_{ij} \quad ,\forall m,n$$

上层长度限制

$$\sum_{i=1}^{3}\left[\left(\sum_{j=1}^{n_1}\beta_{\mathrm{ijmn}}\right) * (\mathrm{XL}_{ij} + 0.1)\right] \le F\mathrm{L}_{ij} \quad ,\forall m,n$$

（3）轿运车承载车辆宽度限制。依据 1-1 型、1-2 型、2-2 型轿运汽车分上下层分别考虑承载汽车的宽度限制，分别设定限定条件。

$$\begin{cases} \omega_{\mathrm{ij1n}} * XW_{ij1n} + 0.1 \le \mathrm{FW}_{ij\mathrm{mn}}, \forall \mathrm{i,j,m} \\ \alpha_{\mathrm{ij2n}} * XW_{ij2n} + 0.1 \le \mathrm{FW}_{下ij2\mathrm{n}}, \forall i,j,n \\ \beta_{\mathrm{ij2n}} * XW_{ij2n} + \beta_{\mathrm{ij2n}} * XW_{ij2n} + 0.1 \le \mathrm{FW}_{上ij2\mathrm{n}}, \forall i,j,n \\ \omega_{\mathrm{ij3n}} * XW_{ij3n} + \omega_{\mathrm{ij3n}} * X + W_{ij2n}0.1 \le \mathrm{FW}_{ij3\mathrm{n}}, \forall i,j,n \end{cases}$$

（4）相邻车间横纵向距离

$$\Delta_w \ge 0.1$$

（5）变量之间约束关系

$$\sum_{i=1}^{3} n_i = \sum_{i=1}^{3}\sum_{j=1}^{n_i}\sum_{m=1}^{3}\sum_{n}^{N_i} \omega_{\mathrm{ijmn}}$$

（6）下层力争装满

$$\sum_{i=1}^{3}\sum_{j=1}^{n_i}\sum_{m=1}^{3}\sum_{n}^{N_i} \alpha_{\mathrm{ijmn}} > \sum_{i=1}^{3}\sum_{j=1}^{n_i}\sum_{m=1}^{3}\sum_{n}^{N_i} \beta_{\mathrm{ijmn}}$$

（7）上层尽量保持对称,即保持上层车辆数为偶数

$$(\sum_{\mathrm{i=1}}^{3}\sum_{j=1}^{n_i} \beta_{\mathrm{ijmn}})\%2 = 0\ ,\ \forall \mathrm{m,n}$$

（8）高度约束。高度超 1.7m 的车只能装在下层，即 III 型车只能装在轿运车下层。

$$\begin{cases}\omega_{\mathrm{3jmn}} = 1\\ \alpha_{\mathrm{3jmn}} = 1\end{cases}$$

（9）逻辑约束

$$\omega_{\mathrm{ijmn}} = \alpha_{\mathrm{ijmn}} + \beta_{\mathrm{ijmn}}$$

$$\omega_{\mathrm{ijmn}} \in \{0,1\},\qquad \alpha_{\mathrm{ijmn}} \in \{0,1\}\qquad \beta_{\mathrm{ijmn}} \in \{0,1\}$$

5.1.2 基于排样算法的配载优化算法

5.1.2.1 空间利用率

考虑用轿运车实际装车时,采用双层装载。在高度和宽度上不存在利用的概念。在长度方向上一般可装载 4-6 辆汽车,具体装载辆数主要取决于汽车的类型。因此,在最大化空间利用率时仅考虑长度的充分利用。

假定汽车的长度用 l 表示,i,j 分别表示轿运车上下层汽车的车位号，n_1, n_2 分别表示上下层车位总数,即上下层装载的汽车辆数;l_i, l_j 分别表示车位 i,j 上的汽车长度,设轿运车长度为 L,则其空间利用率可用长度利用率表示为 $\left(\sum_{\mathrm{i=1}}^{n_1} l_i + \sum_{j=1}^{n_2} l_j\right)\Big/_{2L}$。

即轿运车车单车配载问题的目标函数 F 可表示为:

$$Z = \max\left(\sum_{\mathrm{i=1}}^{n_1} l_i + \sum_{j=1}^{n_2} l_j\right)\Big/_{2L}$$

5.1.2.2 基于排样算法的配载方法思路

轿运车装载问题是复杂得组合优化问题,对于多车型多轿运车的配载问题,若采用穷举法，运载方案将随着汽车种类与轿运车种类增加呈几何级增长。因此需要一个行之有效的算法,制定配载方案。

通过对目标函数分析，可以看出对轿运汽车最大运能优化的实质就是要提高轿运车运载长度利用率。这与工程领域中的一维排样[1]问题极为类似。所谓排样是指多个相同或不同的二维零件轮廓图形在原材料平面区域上的合理布置，使得几何图形之间不出现重叠并且材料利用率最高[2]一维排样问题，即线性排样问题，是指在排样时只需考虑一个方向的尺寸。在求解铁路运输汽车专用车的最大运能优化模型时，为避免采用穷举法出现的“组合爆炸”，可采用基于排样算法的配载方法来解决该问题。由于该算法与汽车运输设备的结构尺寸密切相关，因此结合轿运车的结构尺寸，可设计其基于排样算法的汽车配载优化方法如下：

Step1:不同轿运车装载容量不同，依据不同型号轿运车比例约束，确定轿运车配比安排策略，尽量调用装载容量大的轿运车。
Step2:不同类型承运车装载有一定限制，确定有约束承运车装载策略。
step3：通过循环算法，在 1-1 型与 1-2 型所有甲板上穷举出所有承运车的任意组合情况下对应的空间利用率。在约束条件下，尽量使用空间利用率最高的装载组合。
Step4：建立结果判断器，对前三步装载策略进行综合，进一步优化装载策略。

5.1.2.3　运能最大利用方案的制定

（1）　轿运车类型配比优化

对问题一二三，仅有 1-2 型轿运车与 1-1 型轿运车可以调用。1-2 型轿运车运载量大，故应该优先使用 1-2 型轿运车。但是每次 1-2 型轿运车使用量不超过 1-1 型轿运车使用量的 20%。故可以每装载 5 辆 1-1 型轿运车，就装载一辆 1-2 型轿运车。对于末尾车辆，如果刚好装载在 1-2 型车上且数量较少可以装载在 1-1 型轿运车上，可以将该 1-2 型轿运车换成 1-1 型轿运车，从而使得在轿运车使用数量相同情况下，优先使用 1-1 型成本较低轿运车。

（2）　轿运车运能最大化策略

1） 有限制车辆装载策略

首先对承运车分类为有装载限制，和无装载限制车辆。对问题一二三。III 型车为有装载限制车辆。III 车高度大于 1.7m 故只能装载在 1-1 型与 1-2 型轿运车下层。I 型、II 型车无装载限制。

结合最优装载组合，对有限制车优先装载，剩余无限制车用最优组合填上空缺车位。

2） 无限制车辆最优装载组合确定

单独考虑每一类型轿运车上单独某一甲板上可装载承运车的所有组合方式，并计算该组合的空间利用率，则空间利用率最高的组合为轿运车上最优组合装载方式。

由于对问题一二三宽度上不存在限制，并且 1-1 型轿运车与 1-2 型轿运车上下层长度相等。故只需单独考虑 1-1 型，1-2 型轿运车上某一甲板上所有组合空间利用率，就可推知其他甲板上最优组合。以 1-2 型轿运车上层左侧甲板为例，通过循环法，穷举所有组合方式。

Steps1:计算在在保持安全间距条件下 1-2 上层可以放多少量 II 型车。
Steps1:计算在轿运车上层放 1 辆 I 型车，在保持安全间距条件下可以放多少量

II 型车。
Steps1: 计算在轿运车上层 2 辆 I 型车，在保持安全间距条件下可以放多少量
Steps3:依次类推，直至轿运车上层全放 I 型车。
Steps4:计算以上各种方案该甲板的空间利用率，其中空间利用率最高组合即为该甲板装载的最优组合。

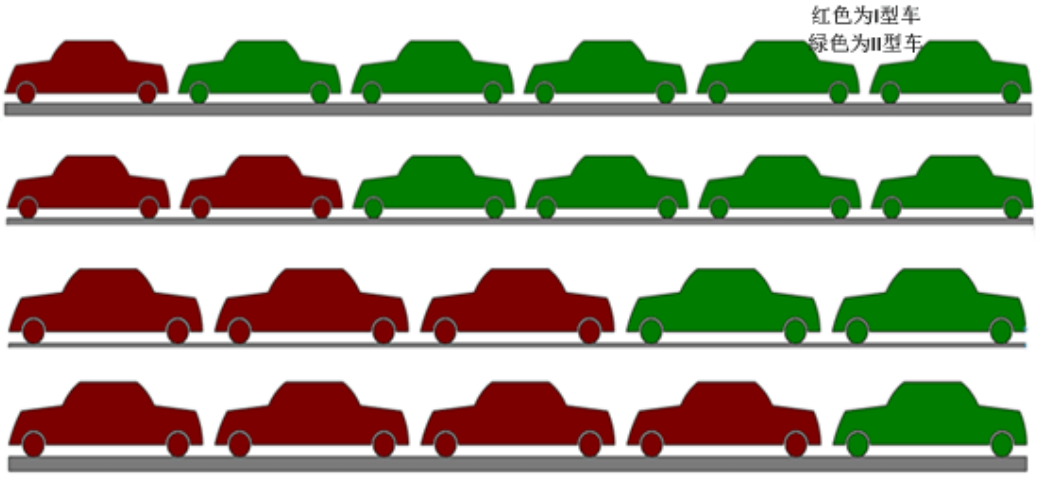


图 5-1　1-2 型车上层左甲板所有各种可能车型组合

3） 最优组合结果
通过计算各种可能组合的空间利用率得到最优装载组合。结果如下：
1-1 型车单甲板上可装 5 辆 II 型车，空间利用率达 95.1%，空间利用率极高。为简化装载，采用该装载方式。
1-2 型车上最优组合为单甲板上放 4 辆 II 型车，2 辆 I 型车，空间利用率达 97.4%。

5.1.2.4　算法求解流程
综合考虑以上优化过程，可以总结为如下流程
STEP1:给变量X_1、X_2、X_3赋值。
STEP2:优先考虑 III车的装配，由于约束条件该车型只能装在轿运车下面，I 型车和 II 型车先装配 I 型车。
STEP3:当装配 I 型车时，轿运车不能装满，同层的空缺将 II 型车看成 I 型装配。
STEP4:判断轿运车数量，每增加 5 辆 1-1 型相应开始考虑增加一辆 1-2 型车。
STEP5:判断一辆 1-1 能否装下余下车辆，不能的话再判断余下的车辆是否一辆 1-2 和一辆 1-1 能装，是的的话再判断两辆 1-1 能否装下余下车辆，如果不能，增加一辆 1-2，能的话增加两辆 1-1；
STEP6:判断轿运车类型是 1-1 型装配 I 型车，若是 1-2 型车而且 I 型车和 II 型车很多，则采用 II 型车与 I 型车的比例是 2:1 装配。
STEP7:最后装配 II 型车。

算法流程图如下：

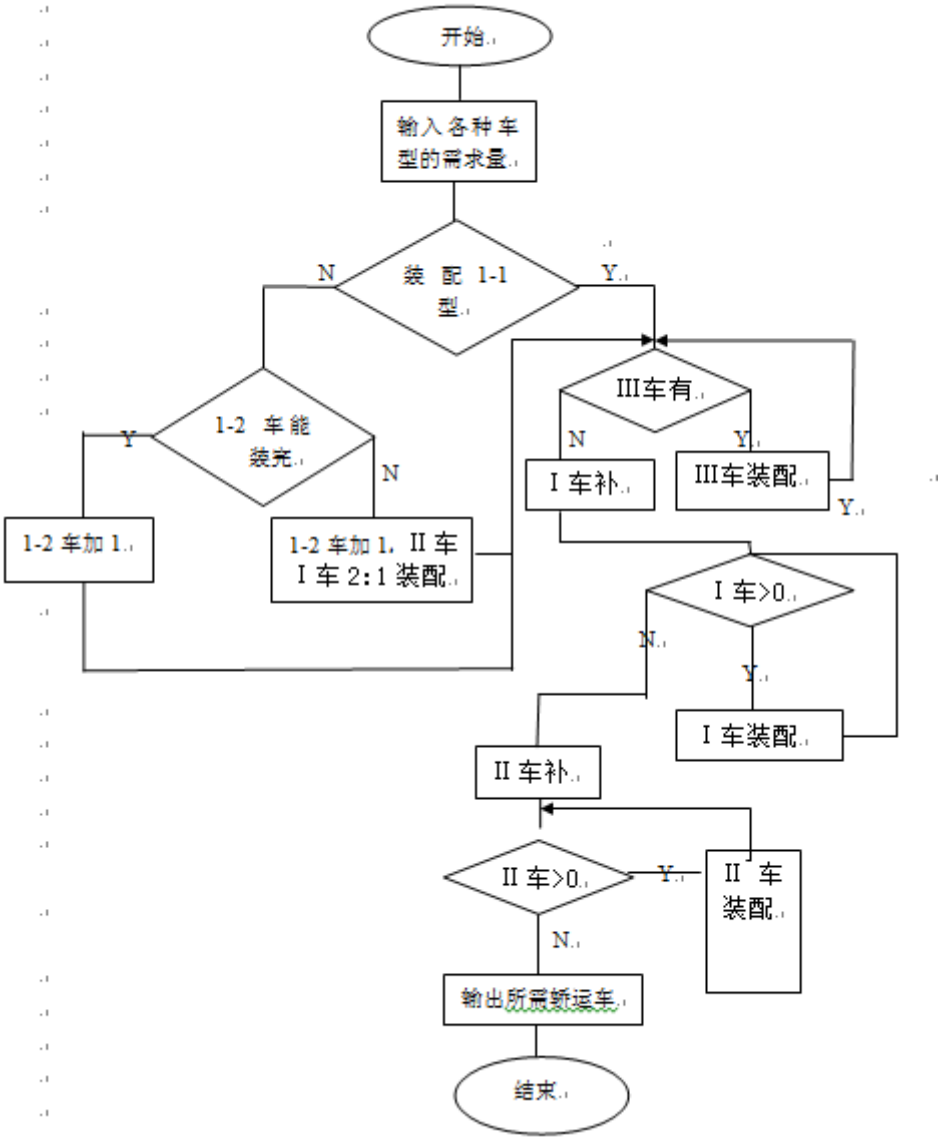


图 5-2　问题一二三算法流程图

5.1.3　基于配载优化算法的模型求解

依据以上算法，通过 visual studio 2010 编程环境，采用 C#与 C 语言进行混合编程对模型进行求解（具体程序见附件），得到下表结果。

表 5-1　问题一装载方案

轿运车类型	相同类型、相同装载方式的车辆数	装在上层序号为 1 乘用车数量	装在上层序号为 2 乘用车数量	装在上层序号为 3 乘用车数量	装在下层序号为 1 乘用车数量	装在下层序号为 2 乘用车数量	装在上层序号为 3 乘用车数量
1-1	11	4	0	0	4	0	0

1-1	4	0	5	0	0	5	0
1-1	1	0	0	0	0	4	0
1-2	2	4	8	0	2	4	0

表 5-2　问题二装载方案

轿运车类型	相同类型、相同装载方式的车辆数	装在上层序号为 1 乘用车数量	装在上层序号为 2 乘用车数量	装在上层序号为 3 乘用车数量	装在下层序号为 1 乘用车数量	装在下层序号为 2 乘用车数量	装在上层序号为 3 乘用车数量
1-1	11	0	5	0	0	0	4
1-1	1	0	5	0	0	0	3
1-2	1	0	12	0	0	0	5

表 5-3　问题三装载方案

轿运车类型	相同类型、相同装载方式的车辆数	装在上层序号为 1 乘用车数量	装在上层序号为 2 乘用车数量	装在上层序号为 3 乘用车数量	装在下层序号为 1 乘用车数量	装在下层序号为 2 乘用车数量	装在上层序号为 3 乘用车数量
1-1	9	4	0	0	0	0	4
1-1	1	4	0	0	1	0	3
1-1	11	4	0	0	4	0	0
1-1	1	0	5	0	3	1	0
1-1	3	0	5	0	0	5	0
1-2	4	4	8	0	2	4	0
1-2	1	0	12	0	0	6	0

采用 CAD 制图软件，对求解结果进行可视化展现，结果如下：

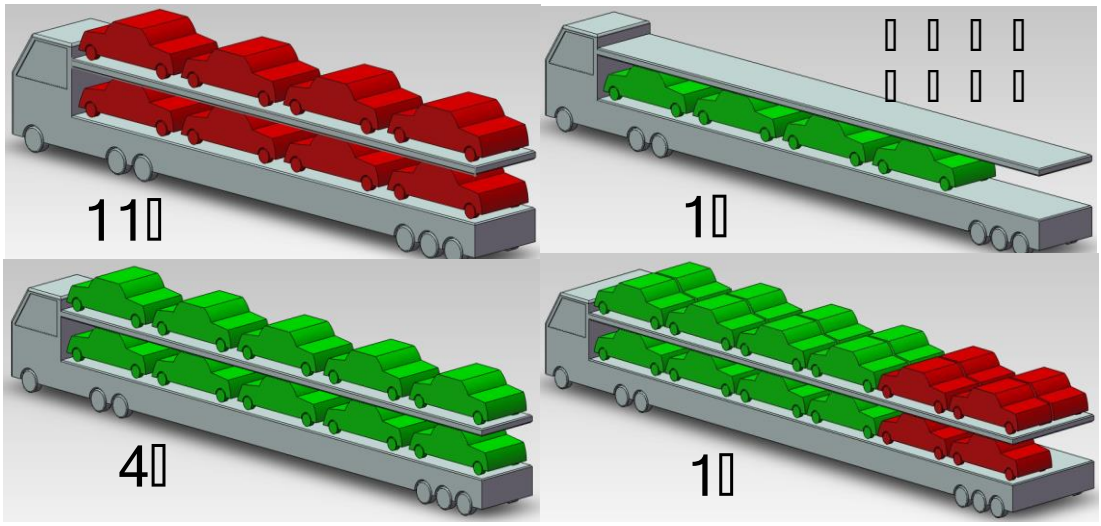


图 5-3 问题一装载方案

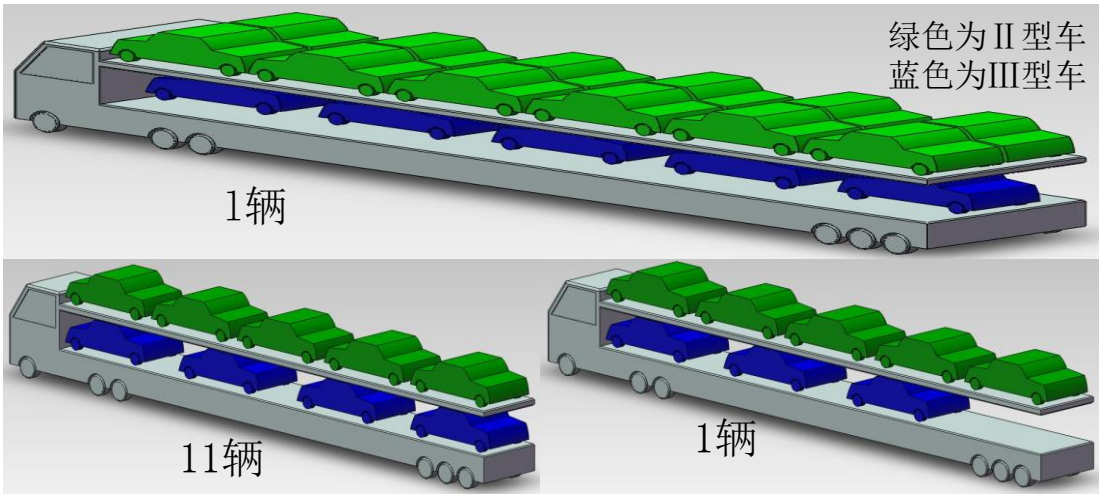


图 5-4　问题二装载方案

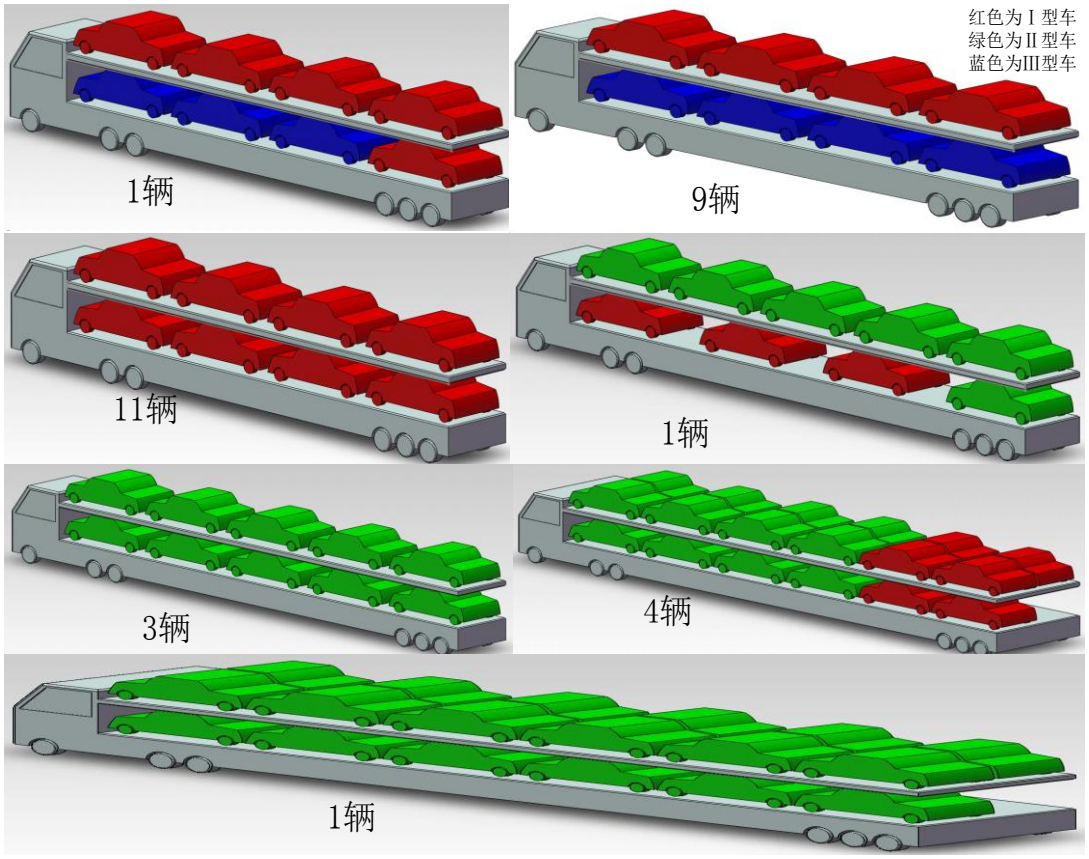


图 5-5 问题三装载方案

5.1.4 结果分析

理论上，依据以上算法，可以运输方案以达到最优。论文计算了各个问题装载方案的总空间利用率，侧面反映装载效果。

表 5-4 问题一二三装载方案总表

	问题一	问题二	问题三
1-1 型车	16	12	25
1-2 型车	2	1	5
总空间利用率	94.1%	95.2%	94.1%

5.2 问题四模型建立与求解

5.2.1 模型建立

问题四的物流配送路径优化问题可以描述为：从物流起点 O 点用多台配送车辆向多个地点配送指定类型及指定数量的车辆。每个地点的位置和货物需求量一定，每台配送车辆的载货量依据装载车辆类型有一定限制。求使得总配送路径最短的方案。

以 0 表示物流中心，以配送总里程最短为目标函数，则可以建立如下物流配送路径优化问题的数学模型：

目标函数：

$$\text{Min Z} = (F_1 + F_2)$$

$$\begin{cases} \max(F_1) \\ \min(F_2) \\ F_1 + F_2 = \text{Z} \end{cases}$$

在此基础上, 使得轿运车总行程数最短

$$\text{Min Z} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{m=1}^{2}\sum_{n=1}^{N_i} d_{ij}Y_{ijmn}$$

约束条件：

$$
\begin{cases}
0 \le X_2 \le 0.2X_1 \\
\sum_{i=1}^{3}\left[\left(\sum_{j=1}^{n_1}\alpha_{\mathrm{ijmn}}\right)*(l_{ij}+0.1\right)] \le F\mathrm{L}_{ij} \quad ,\forall m,n \\
\sum_{i=1}^{3}\left[\left(\sum_{j=1}^{n_1}\beta_{\mathrm{ijmn}}\right)*(l_{ij}+0.1\right)] \le F\mathrm{L}_{ij} \quad ,\forall m,n \\
\Delta_w \ge 0.1 \\
\sum_{i=1}^{3} n_i = \sum_{i=1}^{3}\sum_{j=1}^{n_i}\sum_{m=1}^{3}\sum_{n}^{N_i}\omega_{\mathrm{ijmn}} \\
\sum_{i=1}^{3}\sum_{j=1}^{n_i}\sum_{m=1}^{3}\sum_{n}^{N_i}\alpha_{\mathrm{ijmn}} > \sum_{i=1}^{3}\sum_{j=1}^{n_i}\sum_{m=1}^{3}\sum_{n}^{N_i}\beta_{\mathrm{ijmn}} \\
(\sum_{\mathrm{i}=1}^{3}\sum_{j=1}^{n_i}\beta_{\mathrm{ijmn}})\%2 = 0\ ,\ \forall \mathrm{m,n} \\
\begin{cases}\omega_{3\mathrm{jmn}} = 1 \\ \alpha_{3\mathrm{jmn}} = 1\end{cases} \\
\omega_{\mathrm{ijmn}} = \alpha_{\mathrm{ijmn}} + \beta_{\mathrm{ijmn}} \\
\omega_{\mathrm{ijmn}} \in \{0,1\}\ \alpha_{\mathrm{ijmn}} \in \{0,1\} \quad \beta_{\mathrm{ijmn}} \in \{0,1\} \\
\begin{cases}
\omega_{\mathrm{ij1n}} * XW_{ij1n} + 0.1 \le \mathrm{FW}_{ijmn}, \forall \mathrm{i,j,m} \\
\alpha_{\mathrm{ij2n}} * XW_{ij2n} + 0.1 \le \mathrm{FW}_{下ij2n}, \forall i,j,n \\
\beta_{\mathrm{ij2n}} * XW_{ij2n} + \beta_{\mathrm{ij2n}} * XW_{ij2n} + 0.1 \le \mathrm{FW}_{上ij2n}, \forall i,j,n \\
\omega_{\mathrm{ij3n}} * XW_{ij3n} + \omega_{\mathrm{ij3n}} * X + W_{ij2n}0.1 \le \mathrm{FW}_{ij3n}, \forall i,j,n
\end{cases}
\end{cases}
$$

5.2.2 远距离优先调度算法的设计

5.2.2.1 算法策略

对问题四，对问题四，如果先综合考虑完成 ABCD 点的任务，直接计算完成总任务轿运车最少所需要的类型与数量，会复杂化轿运车的配送安排。会出现一辆轿运车同时配送多个点状况，会使得有些轿运车总行驶距离增大。故采用先分别考虑 A、B、C、D 任务完成，再综合调节装载与运输方案，使得完成任务前提下轿运车使用数量最少，并且总行驶里程最少。

由于该物流公司是全国性公司，在各地均会有整车物流业务，轿运车到达目的地后原地待命，无须放空返回。据此，可以采用远距离优先调度的策略，优先对较远节点进行货物配送。首先将目的地按距离进行降序排序，每辆轿运车依据问题一三三算法满负荷装载选择一条最短路依次给较远目的地配车，对于最后剩余的散装车辆，则中途卸载给在途经路线中距离目的地最近节点。依据此流程，配送完前 N-1 个节点。而对于最后一个节点，则是依据需求，最优组合装载方案运送。

5.2.2.2 算法流程

STEP1:给变量X_1、X_2、X_3赋值。

STEP2:考虑路程对成本的影响，优先配送远距离的目的地。

STEP3:1-2 型车下层优先装配III乘用车，下层多余空缺用Ⅰ型车代替。1-2 型车

上层的装配乘用车Ⅱ车与Ⅰ型车的比例是 2:1 装配。

STEP4:装配时考虑与之相邻途径目的地所需的乘用车数量和种类，优先装配相邻目的地所需乘用车的种类，如果所有种类都有，按照 1-3 问题的通用模型装配。

STEP5:单独考虑某个目的地的时,运用 1-3 问题的通用模型计算所轿运车的种类和数量，考虑到轿运车不能装满的情况。多余运力分配给途径目的地，此途径目的地的相应的X_1、X_2、X_3变量减少对应值。

STEP6: 轿运车运输乘用车时，在与之相邻的途径目的地卸载为之装配乘用车。

STEP7:去除应经装配好的终点目的地与途径目的地的联系，这样途径目的地变为终点目的地。重复 STEP1- STEP6。

STEP8:根据各目的地轿运车的数量与路程的乘总和为最短路程。

算法流程图如下

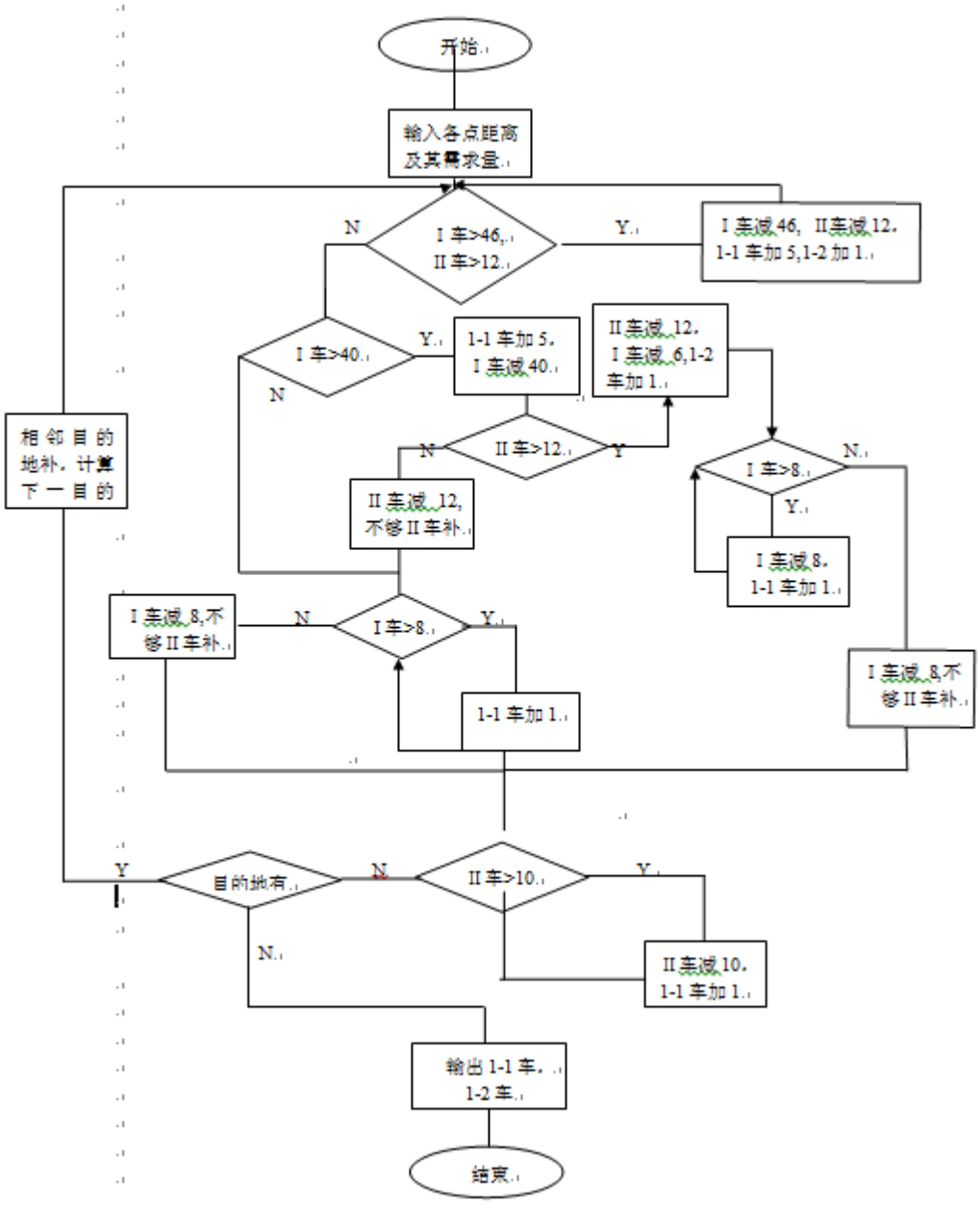


图 5-6　问题四算法流程图

5.2.3 基于远距离优先调度算法的模型求解

依据问题四求解算法，在 visual studio 2010 编程环境下，采用 C#与 C 语言混合编程，对模型进行编程求解（程序见附录），求解结果如下：

表 5-5 A 点配送方案

轿运车类型	相同类型、相同装载方式的车辆数	装在上层序号为1乘用车数量	装在上层序号为2乘用车数量	装在上层序号为3乘用车数量	装在下层序号为1乘用车数量	装在下层序号为2乘用车数量	装在上层序号为3乘用车数量	中间停靠地	目的地
1-1	5	4	0	0	4	0	0	B、D	A
1-1	1	0	5	0	0	5	0	B、D	A
1-1	1	0	5	0	0	4	0	B、D	A
1-2	1	4	8	0	2	4	0	B、D	A

表 5-6 B 点配送方案

轿运车类型	相同类型、相同装载方式的车辆数	装在上层序号为1乘用车数量	装在上层序号为2乘用车数量	装在上层序号为3乘用车数量	装在下层序号为1乘用车数量	装在下层序号为2乘用车数量	装在上层序号为3乘用车数量	中间停靠地	目的地
1-1	4	4	0	0	4	0	0	D	B
1-2	1	10	0	0	5	0	0	D	B

表 5-7 C 点配送方案

轿运车类型	相同类型、相同装载方式的车辆数	装在上层序号为1乘用车数量	装在上层序号为2乘用车数量	装在上层序号为3乘用车数量	装在下层序号为1乘用车数量	装在下层序号为2乘用车数量	装在上层序号为3乘用车数量	中间停靠地	目的地
1-1	3	4	0	0	4	0	0	D	C
1-1	3	0	5	0	0	5	0	D	C
1-1	1	4	0	0	5	0	0	D	C
1-2	1	4	8	0	2	4	0	D	C

表 5-8　D 点配送方案

轿运-车类型	相同类型、相同装载方式的车辆数	装在上层序号为 1 乘用车数量	装在上层序号为 2 乘用车数量	装在上层序号为 3 乘用车数量	装在下层序号为 1 乘用车数量	装在下层序号为 2 乘用车数量	装在上层序号为 3 乘用车数量	中间停靠地	目的地
1-1	3	4	0	0	4	0	0	无	C
1-2	1	10	0	0	5	0	0	无	C

采用 CAD 与 Visio 制图软件，对求解结果进行可视化展现，结果如下：

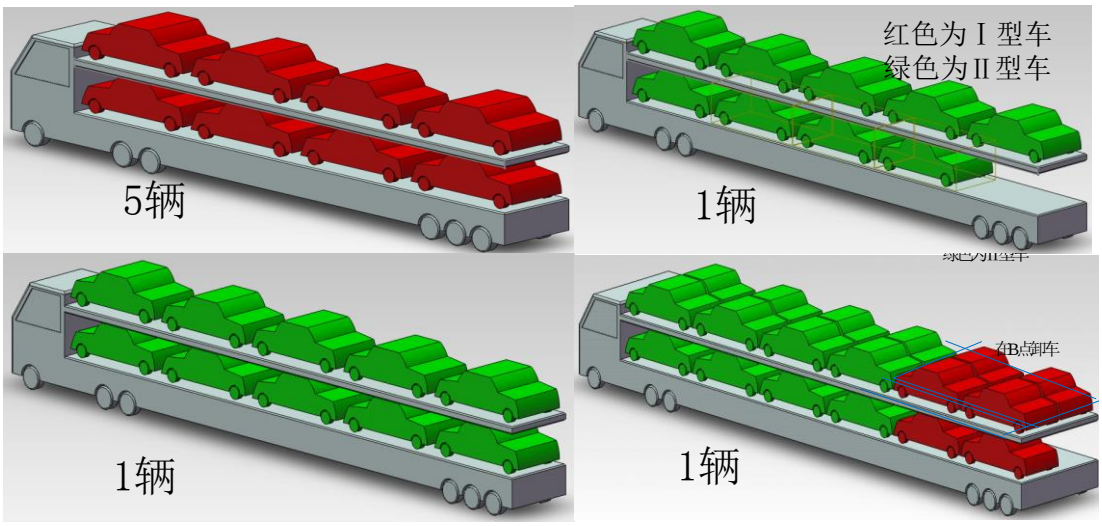


图 5-7　A 点装载方案

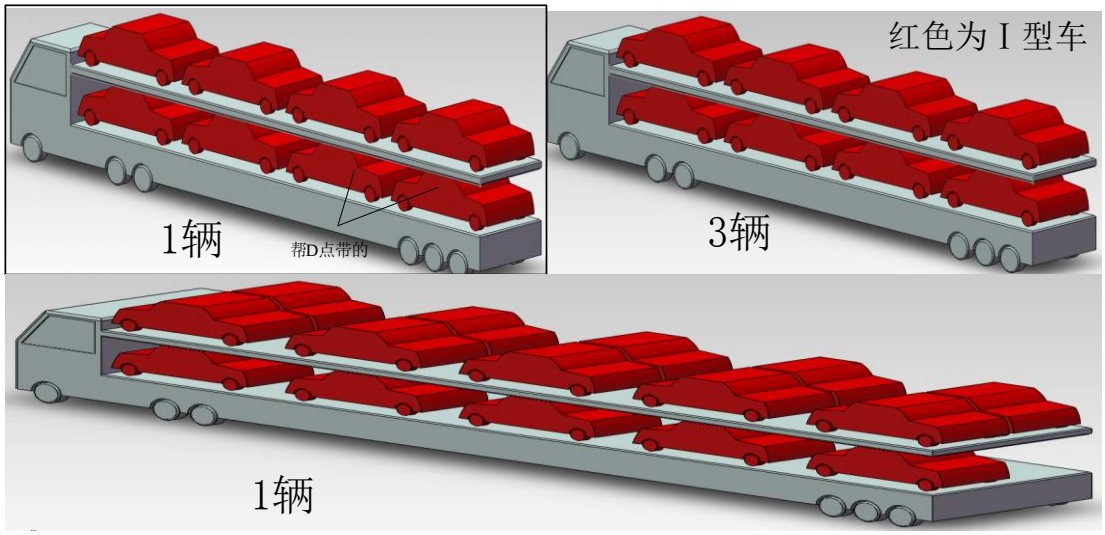


图 5-8　B 点装载方案

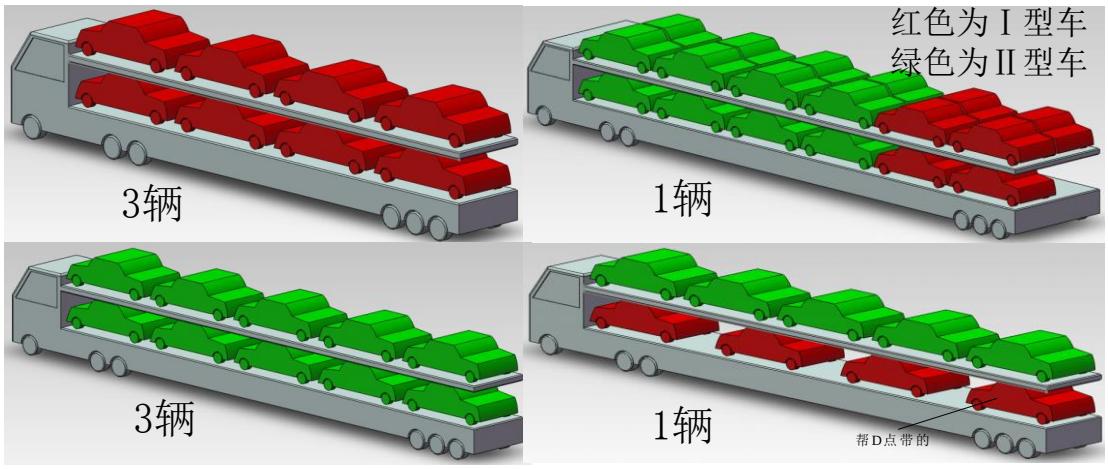


图 5-9　C 装载方案

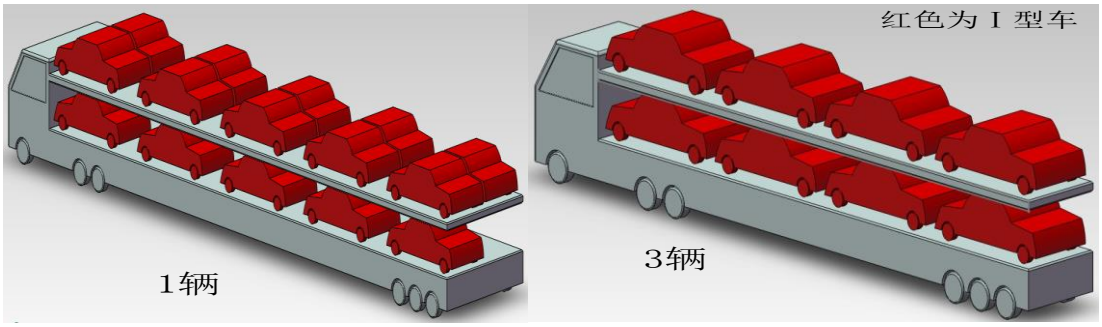
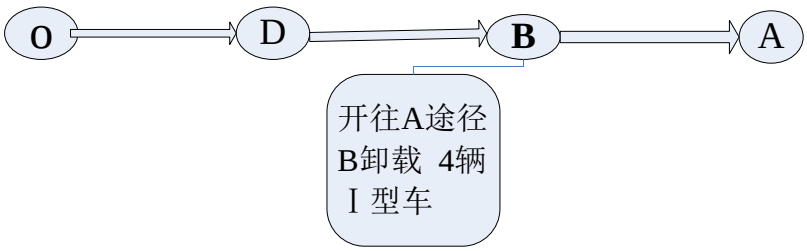


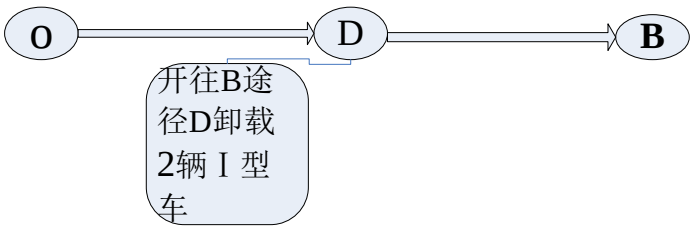
图 5-10　D 点装载方案

具体配送过程如下：

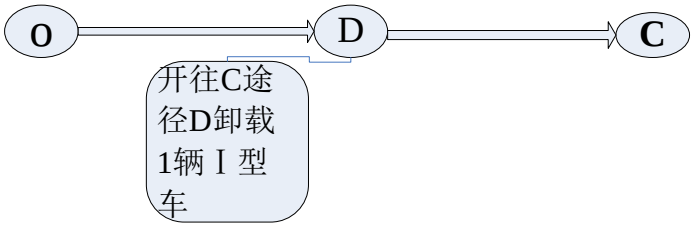
A 点的配送方案：7 辆 1-1 型轿运车, 1 辆 1-2 型轿运车



B 点的配送方案：4 辆 1-1 型轿运车, 1 辆 1-2 型轿运车



C 点的配送方案：7 辆 1-1 型轿运车, 1 辆 1-2 型轿运车



D 点的配送方案：3 辆 1-1 型轿运车,1 辆 1-2 型轿运车

5.2.4 结果分析

针对问题四,基于远距离优先调度的算法取得了满意的结果,具体见表 5-9。可以看到 1-1 型车与 1-2 型车配比达到最优,且轿运车空间利用率达到了 93.4%以上，总行驶里程也较为 6408 单位。

表 5-9 问题四结果总表

	A	B	C	D	合计
1-1 型车使用量	7	4	7	3	21
1-2 型车使用量	1	1	1	1	4
空间利用率	94.3%	97.2%	95.2%	93.4%	-
总行驶里程	2880	1400	1888	640	6408

5.2.5 基于遗传算法的运输方案优化模型

上文算法的优点是求解精度高，但是当物流节点增加，路径交错复杂时，该类型算法时间复杂度将剧增。为了对企业物流物流调度实现实时调度，可以采用启发式算，寻找相对最优解。常用的启发式算法有：禁忌搜索算法(TS)、模拟退火算法(SA)、遗传算法等(GA)[3]。

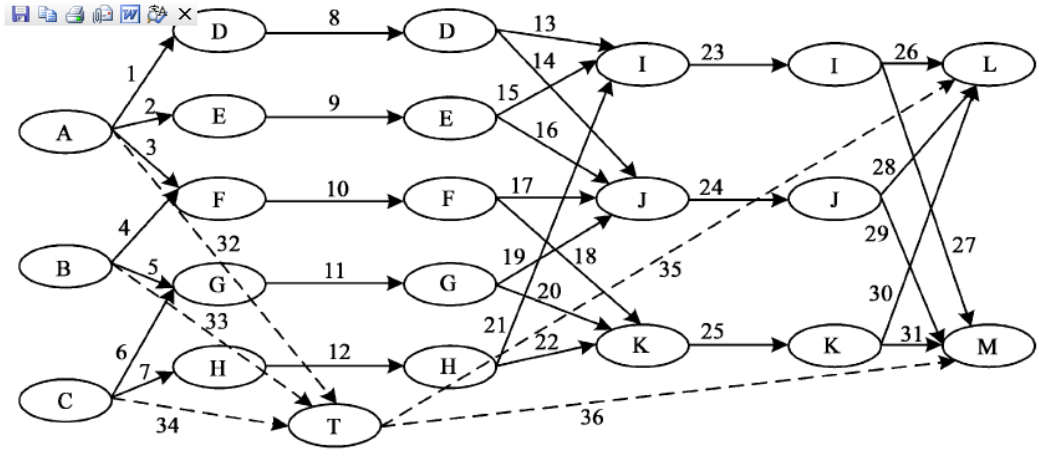


图 5-11 复杂物流配送网络示例

5.2.6 GA 算法基本思想

GA 算法具有较好的逼近最优解和使运算时间大大缩短的优点,所以能够兼顾运算时间和效率两个方面，是具有较好的发展前途的优化方法[4]。。但是遗传算法容易陷入局部最优解，出现“早熟收敛”等现象。本文针对此情况，设计了改进的遗传算法，并应用解决其

求解车辆优化调度问题。

为了安排线路，需要预先对需要的车辆数进行估计。可按照下式确定车辆数[5-6]。

$$\mathrm{m}=\left[\sum_{i=1}^{n} g_i laq\right]+1$$

式中：[]表示取整；a（0< a<1）是车辆载重量的弹性系数，根据装车的复杂性程度及约束多少估计得出，一般来讲，装车越复杂，约束越多，应越小，表示一辆车真正能容纳的货物越少。该算法通过人机对话调整 a 的大小。

以C_{ij}表示为从点 i 到点 j 的运输成本(距离)，在此采用客户到客户之间的运距 d_{ij}(i= 1, 2,⋯,n ； j= 1,2,⋯,n，当 i= 0 时表示配送中心，即配送中心和各个客户之间的距离)目标为使车辆的总运输距离最短。

先定义变量如下：

$$\mathrm{Y_{ijmn}}=\begin{cases} 1,\text{轿运车}F_{mn}\text{由 i 使向 j} \\ 0,\text{轿运车}F_{mn}\text{不由 i 使项 j} \end{cases}$$

$$y_{i\mathrm{mn}}=\begin{cases} 1，\text{车辆}F_{mn}\text{向 i 客户运车} \\ 0，\text{车辆}F_{mn}\text{不向 i 客户运车} \end{cases}$$

得到数学模型如下：

目标函数为

$$\mathrm{Min\ Z}=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{m=1}^{2}\sum_{n=1}^{N_i} d_{ij}Y_{ijmn}$$

约束条件为

$$\sum_{i=0}^{n} x_{ijk}=y_{\mathrm{ik}}\quad (\mathrm{j}=0,1,\cdots,\mathrm{n};\mathrm{k}=1,2,\cdots,\mathrm{m})$$

$$Y_{ijmn}=0\text{ 或者 }1\quad (\mathrm{i,j}=0,1,\cdots,\mathrm{n};\mathrm{m}=0,1;\mathrm{n}=1,2,\cdots,N_i)$$

$$y_{imn}=0\text{ 或 }1\qquad (\mathrm{i}=0,1,\cdots,\mathrm{n};\mathrm{m}=0,1;\mathrm{n}=1,2,\cdots,N_i)$$

5.2.6.1　遗传算法的设计

（1）　遗传算法的编码设计

本文采用基于自然数的编码方式。车辆数 m 事先估计，自然数 n 表示客户的个数，进化群体中的染色体结构表示为：（$O,i_{11,}\ i_{12}\cdots i_{1s},0,i_{21}\cdots,i_{2t,}\cdots$，$0,i_{m1}\cdots$，$i_{mt,}$）染色体的总长为 n+m。其中，自然数$i_{kj}$表示第 K 条路线中的第 j 个客户,自然是 0 表示配送起点，m 个 0 把染色体分为 m 段，代表 m 条路径。该染色体可以解释为第一辆车从配送起点 0 出发，依次经过客户

$O, i_{11}, i_{12} \cdots i_{1s}$形成子路径 1；第二辆车从配送起点 0 出发，依次经过客户 $i_{21} \cdots i_{2t,}$，形成子路径 2；以此类推，直到形成子路径 m 。这样，染色体具有子路径内部有序，各个子路径间无序。

（2） 种群的初始化

产生初始群体的步骤如下：

步骤 1 根据配送中心和客户的编号，产生以自然数串(0,1,2,3,…, n)0 表示配送中心，1,2,…,n 表示 m 个客户点；

步骤 2 将 m-1 个 0 随机插入到以上自然数串中，构成长度为 m+n 的一个染色体；为了防止产生无效的染色体，插入 0 后满足任意两个 0 不相邻；

步骤 3 在染色体中随机选择基因值不全为 0 的两个点，进行交换，当交换次数达到 k 次后，产生另外一条染色体，K 为一随机数；

步骤 4 重复步骤 3，直至满足种群规模，即 popsize。

（3） 种群变异策略

1） 约束处理与适应度函数

应用轮转法选择法，要求适应度值不为负，因此通过变化将目标函数转化为适应度函数

$$f_k = \frac{b * z_1}{z_k}$$

式中：f_k为第 k 个染色体的适应度，b 为常数，z_1为初始群体中最优染色体的总运输成本的估计值，z_k为当前染色体所对应的运输成本。

对于 VRP 问题，我们采用如下公式对目标函数进行约束：

$$\text{sumdd} = \begin{cases} pathlen, carrycase = 0 \\ pathlen + C, carrycase = 1 \end{cases}$$

公式中，sumdd 表示运输的总路径长度，carrycase 为 0 表示行车方案满足车载量要求时，目标函数值为实际的总路径长度，carrycase 为 1 表示行车方案超载时，目标函数值为实际的总路径长度加上惩罚的路径长度，C 为惩罚的路径长度。

2） 选择算子

选择算子采用改进的轮转法。轮转法是 GA 常用的选择方法，它使各染色体的选择概率与其适应度值成正比例，对于适应度为f_i的第 i 个染色体，被选择的概率为

$$\text{p(i)} = \frac{f_i}{\sum_{j=1}^{popsize} f_i}$$

即 p(i)等于第 i 个染色体的适应度在整个群体的适应度总和中所占的比例，染色体适应度越大，它被选择的概率越高。改进的方法是，将目前为止的最好染色体直接复制到下一代群体中。

交叉算子：依据轮转法策略选择适应度高的个体，进行遗传操作。

非满载 VRP 问题具有组间无序、组内有序的特征，如果单纯的使用一般的交叉算子往往会使一些优秀的子路径被破坏，并且在两父个体相同时，无法再

产生新的个体，所以采用对顺序交叉(OX)方式进行改进的一种交叉算子[7]首先通过随机函数产生两个数作为交叉点，两个父个体交叉时，通过选择父个体 1 的一部分，然后保存父个体 2 中的客户码的相对顺序生成。

例如：对下面两个父个体，随机选择两个数(4,9)作为交叉点：

p1：(0 8 5 ｜ 7 0 1 6 0 ｜ 2 4 3)

p2：(0 1 3 ｜ 4 5 0 7 8 ｜ 0 2 6)

首先，两个交叉点之间的中间段和第一位的基因码保持不变，得到：

01：(0 * *｜ 7 0 1 6 0 ｜* * *)

02：(0 * *｜ 4 5 0 7 8 ｜* * *)

然后,取父个体 2 从第 2 个交叉点开始客户码的排列顺序,当到达表尾时，返回表头继续记录客户码，直至到达第 2 个交叉点结束，这样便获得了父个体 2 从第 2 个交叉点开始的客户码排列顺序为 0-2-6-0-1-3-4-5-0-7-8。对于父个体 1 而言，已有客户码 0，7，0，1，6，0，将它们从父个体 2 的客户码排列顺序中去掉，得到排列顺序 2-3-4-5-8，再将这个排列顺序复制给父个体 1，复制的起点也是从第 2 个交叉点开始,以此决定子个体 1 对应位置的未知码*，这样子个体 1 表示为：01：(0 5 8 7 0 1 60 2 3 4)。

同样，可以产生子个体 2 为：02：(0 6 0 4 5 0 7 8 2 3 1)。

为了使子代继承更多的父带基因信息，本文采用了两种变异算子：一种是改进现有的逆转变异算子得到；另一种是 k 对基因换位变异算子。

（4） 种群更新策略和终止条件

1） 种群更新策略

传统遗传算法在遗传过程中控制参数是不变的，这样，选取合适的控制参数对算法设计就非常重要。本文采取了自适应调整遗传概率的策略。基本思想是，在进化的初期应使用较大的遗传概率，从而使群体内具有足够的多样性，这将有助于找到全局最优解；而在进化的后期，较小的遗传概率将使算法具有良好的收敛性，且使搜索在某些可能有最优解的局部邻域内进行。

同时，在每一代染色体的求解后，都保留一定数量的较优染色体，同时替换掉较差的若干染色体。这样可以避免进化的过程中随着交叉变异操作的进行，失去最优解的可能性，从而提高了问题求解的收敛速度，增加算法自适应环境的能力。

2） 终止条件

循环次数达到规定值 maxgen 时终止算法。

5.2.7 基于遗传算法的模型求解

首先为算法准备数据：

表 5-10 各目的地间距离及各目的地所需要的车型类型及数量：

C_{ij}	0	A	B	C	D	E
0	0					
A	360	0				
B	280	80	0			
C	236	276	196	0		
D	160	200	120	76	0	
E	384	60	104	300	224	0
I 型车需求量	-	42	50	33	41	0

II 型车需求量	-	31	0	47	0	0

对模型进行求解，具体结果见 5-11 表：

表 5-11　遗传算法运行结果

运行次数	路径长度	运行次数	路径长度
1	7036	6	7012
2	6812	7	6452
3	6976	8	6702
4	6652	9	6204
5	6902	10	6904

由表 5-11 数据可知当物流网络复杂时，模型求解结果虽然不是最优，但是通过多次求解，仍然能找到较佳的解。

5.2.8　结果分析

在一般算法中，个体的编码采用数目不稳定的编码方式，即当问题需要 m 辆车时，个体的编码将使用 n*m 个基因的编码方式，这样随着问题的不断复杂，以往的算法体现出了明显的缺陷，搜索速度将成倍的变慢。而随着问题的不断复杂，本算法的优越性越发明显，由于算法的编码方式始终是采用固定的 n+m 个基因，搜索时间不会随问题的复杂而有明显变化。

实验结果表明，改进后的遗传算法对 VRP 问题的求解不失为一种更优的搜索方法，能够快速的求到最优解，尤其是在最初几代变换中明显的进化速度表明本文构造的算法在较小的种群规模下可以很快的逼近最优解。

5.3 问题五模型建立与求解

5.3.1　模型建立

目标函数

第一目标：使用尽量少的轿运车完成任务。

$\mathrm{Z} = \mathrm{Min}\sum_{\mathrm{i}=1}^{3} F_i$

第二目标：在使用轿运车数量相同前提下，依据各类轿运车运输成本，优化各类轿运车的数量。

$$\begin{cases} Z = F_1 + F_2 + F_3 \\ \mathrm{Min}(F_1 + 1.2F_2 + 2F_3) \end{cases}$$

第三目标：轿运车使用数量以及各类型配比一致的前提下，使得轿运车总行程里程最短。

$$\mathrm{Min}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{m=1}^{2}\sum_{n=1}^{N_i} d_{ij}Y_{ijmn}\right)$$

约束条件：

（1）轿运车承载车辆长度限制

下层长度限制

$$\sum_{i=1}^{3}\left[\left(\sum_{j=1}^{n_1}\alpha_{\mathrm{ijmn}}\right)*\left(\mathrm{XL}_{ij}+0.1\right)\right]\leq FL_{ij}\quad,\forall m,n$$

上层长度限制

$$\sum_{i=1}^{3}\left[\left(\sum_{j=1}^{n_1}\beta_{\mathrm{ijmn}}\right)*\left(\mathrm{XL}_{ij}+0.1\right)\right]\leq FL_{ij}\quad,\forall m,n$$

（2）轿运车承载车辆宽度限制。依据 1-1 型、1-2 型、2-2 型轿运汽车分上下层分别考虑承载汽车的宽度限制，分别设定限定条件。

$$\begin{cases}\omega_{\mathrm{ij1n}}*XW_{ij1n}+0.1\leq \mathrm{FW}_{ij\mathrm{mn}},\forall \mathrm{i,j,m}\\ \alpha_{\mathrm{ij2n}}*XW_{ij2n}+0.1\leq \mathrm{FW}_{下ij2\mathrm{n}},\forall i,j,n\\ \beta_{\mathrm{ij2n}}*XW_{ij2n}+\beta_{\mathrm{ij2n}}*XW_{ij2n}+0.1\leq \mathrm{FW}_{上ij2\mathrm{n}},\forall i,j,n\\ \omega_{\mathrm{ij3n}}*XW_{ij3n}+\omega_{\mathrm{ij3n}}*X+W_{ij2n}0.1\leq \mathrm{FW}_{ij3\mathrm{n}},\forall i,j,n\end{cases}$$

（3）相邻车间横纵向距离

$$\Delta_w\geq 0.1$$

（4）变量之间约束关系

$$\sum_{i=1}^{3}n_i=\sum_{i=1}^{3}\sum_{j=1}^{n_i}\sum_{m=1}^{3}\sum_{n}^{N_i}\omega_{\mathrm{ijmn}}$$

（5）下层力争装满

$$\sum_{i=1}^{3}\sum_{j=1}^{n_i}\sum_{m=1}^{3}\sum_{n}^{N_i}\alpha_{\mathrm{ijmn}}>\sum_{i=1}^{3}\sum_{j=1}^{n_i}\sum_{m=1}^{3}\sum_{n}^{N_i}\beta_{\mathrm{ijmn}}$$

(6)上层车辆尽量保持对称，即使上层车辆总数为偶数

$$(\sum_{\mathrm{i}=1}^{3}\sum_{j=1}^{n_i}\beta_{\mathrm{ijmn}})\%2=0\ ,\ \forall \mathrm{m,n}$$

（7）高度约束。高度超 1.7m 的车只能装在下层，即 III 型车只能装在轿运车下层。对 2-2 型轿运车因为层高较低，上、下层均不能装载高度超过 1.7 米的乘用车。

$$\begin{cases}\omega_{\mathrm{3jmn}}=1\\ \alpha_{\mathrm{3jmn}}=1\\ \omega_{\mathrm{3j3n}}=0\end{cases}$$

（8）逻辑约束

$$\omega_{\mathrm{ijmn}}=\alpha_{\mathrm{ijmn}}+\beta_{\mathrm{ijmn}}$$

$$\omega_{\mathrm{ijmn}}\in\{0,1\},\qquad \alpha_{\mathrm{ijmn}}\in\{0,1\}\qquad \beta_{\mathrm{ijmn}}\in\{0,1\}$$

5.3.2　禁忌搜索算法思想及流程

5.3.2.1　　基本思想

禁忌搜索算法[8]是一种元启发式(Meta-heuristic)优化算法,其思想最早

于 1986 年由美国工程院院士 Fred Glover 提出。它是对局部邻域搜索的一种扩展,是一种全局逐步寻优算法。TS 算法通过引入一个灵活的存储结构和相应的禁忌准则来避免迂回搜索,并通过特赦准则来赦免一些被禁忌的优良状态,进而保证多样化的有效探索以最终实现全局优化。

在禁忌搜索算法中,首先采用经典启发式算法或随机方法产生一个初始解作为当前解,然后按照预先设置的解的邻域生成方式,生成邻域候选解集,并取其中的最好解作为新的当前解。为了避免陷入局部最优解,TS 算法要求在搜索过程中允许一定的下山操作,即允许接受劣解。另外,为了避免对已搜索过的局部最优解进行重复搜索,TS 算法采用禁忌表来记录已搜索的局部最优解的历史信息,这在一定程度上可使搜索过程避开局部极值点,从而开辟新的搜索区域。

5.3.2.2　算法流程

根据 TS 算法的基本思想,可将其用如下步骤进行描述:

(1)设定算法的参数,随机给出初始解 X,把禁忌表设置为空集;

(2)根据当前解的情况,判断计算过程是否满足算法的终止条件。若满足,则结束算法并输出优化结果;否则,继续执行以下步骤;

(3)用邻域函数产生当前解 x 的邻域解,并且选取若干邻域解作为候选解,从中选出候选最优解;

(4)对候选解判断特赦准则是否成立。若成立,则用满足特赦准则的最佳状态 y 替代 x 成为新的当前解,即 x = y,并用与对应的禁忌对象替换最早进入禁忌表的禁忌对象,同时用 y 替换当前最优解,然后转步骤 6; 否则,继续以下步骤;

(5)判断候选解对应的各对象的禁忌属性,选择候选解集中非禁忌对象对应的最佳状态作为新的当前解,同时用与之对应的禁忌对象替换最早进入禁忌表的禁忌对象元素;

(6)转步骤(2)。

上述算法可用图 5-12 所示的流程图更直观地描述。

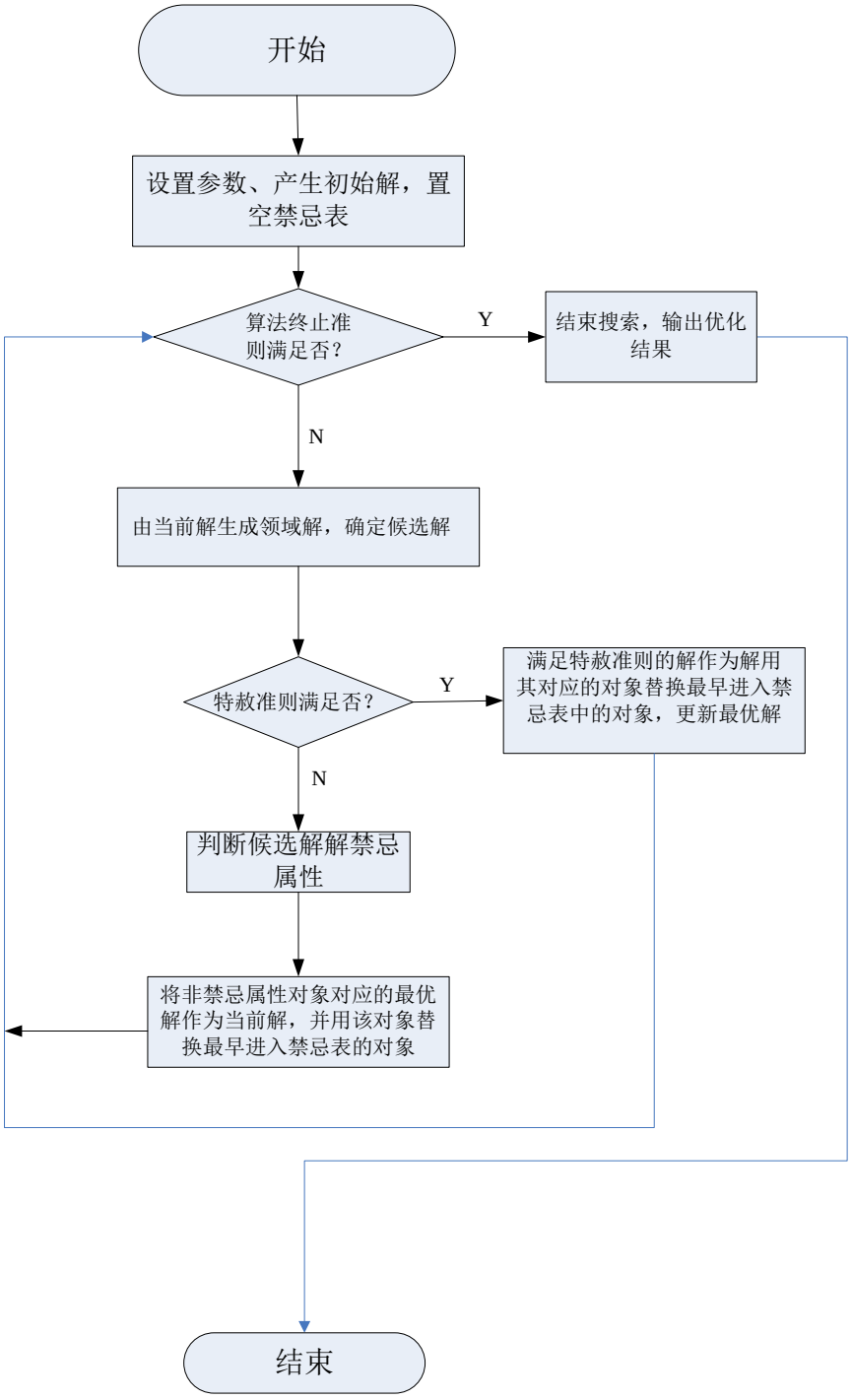


图 5-12 禁忌搜索算法流程图

5.3.2.3 基于禁忌搜索算法的配载优化模型求解

（1） 基于分类与排序的数据预处理

问题五所涉及数量很庞大，主要体现在轿运车型号与数量、乘用车型号与数量相对于前四问大幅提升。如果直接将数据作为禁忌搜索算法的输入，则会算法程序开销大，而且由于解空间太庞大，会使得算法结果稳定性不高。故可以在输入数据之前设定一个分类器，依据一定的规则过滤掉一些明显效果不好的可行解，这样不仅能降低程序开销，同时能提升算法稳定性。如图，直接将数

据输入禁忌搜索算法类似从 A ~ D 中随机寻找最优解,不仅程序开销大,而且容易陷入局部最优解,如 E。而使用基于过滤规则的优化输入后的禁忌搜索算法类似从 B~C 中寻找最优算法，不仅程序开销小，而且不会得到太差的解，能提升模型求解的稳定性。

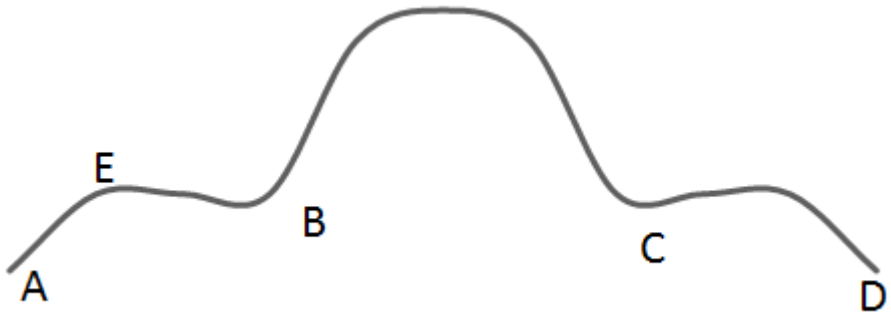


图 5-13　基于过滤规则的禁忌搜索算法效果图

为了优化输入数据，可以建立以下分类规则，将要运载的汽车分为以下三类。同时参考模型约束规则与附件 1 与附件 2 中数据，可以得到各个类型车辆可以装载进去的轿运车。假设 h 代表车高，w 代表车宽。

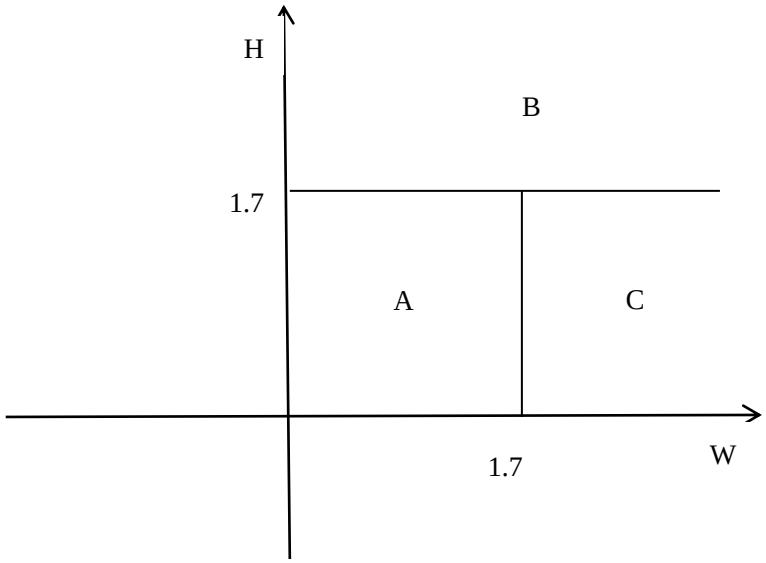


图 5-14　承运汽车分类示意图

A 类：h<1.7,w<1.7 无限制乘用车，可以装载在任意轿运车上。
B 类：h>1.7　有限制乘用车，仅可以装载在 1-1 型车下层，1-2 型车下层
C 类：w>1.7 h<1.7 有限制乘用车，仅可以装载在 1-1 型车，1-2 型车下面

可以通过建立优先装载规则,过滤掉一些比较差的装载情况,从而优化输入。优先装载规则（过滤规则）为：

1） 优先装载有限制车辆：优先装载 B 类车，将 B 类车放在 1-2 型车下层，1-2 型车不够时再将车装载在 1-1 下层。再装载 C 类车，放在 1-2 型车下层，

同样 1-2 型车不够再装载在 1-1 型车上。再装载 A 类车，由于 B、C 类车基本装载在 1-2 型、1-1 型车下层，故 A 类车装载在这些上层有空位的 1-2 型，1-1 型车。车不够再依次装载在 2-2 型，1-2 型，1-1 型车上。

2） 优先将车装载在容量大轿运车上。在满足约束规则条件下，装车顺序为 2-2 型、1-2 型、1-1 型

3） 优先将车装载在较长轿运车上。依据排序规则，将轿运车按长度排序，优先使用较长车辆运载。

4） 优先将车调度给较远节点，方便后期优化，将可多运载车中途卸载，减少轿运车使用量。

5） 综合以上过滤规则，形成过滤差解的过滤器。

（2） 算法策略

用禁忌搜索算法求解轿运车面向多车型的优化配载问题时,确定解的表示方法是一项非常关键的基础工作,因为它将直接决定算法实现的难易程度和算法性能的优劣。为方便算法的实现,可采用汽车分类后直接排列的表示方法。

汽车直接排列的表示方法具体步骤为：将所有汽车编号，建立汽车编号与汽车类型字典后。对汽车依据分类规则分为 A、B、C 三类。先将 B 类汽车编号随机打乱，编号顺序即为装载在 1-2 型车、1-1 型车下层顺序。再依据同样方法，依次打乱 C 类、A 类车中编号，并依据编号顺序进行装载，据此形成不错装载方案的解

例如：对于一个,4 辆汽车装载 22 辆汽车至 A 点的配载优化问题。先将 22 辆汽车编号为 1~22，并对其依据分类规则进行分类。假设 A 类车编号为 1,3,6,10,13,14,17,18,21 。B 类车编号为：2,5,7,11,15,20 。C 类车编号为 4,8,9,12,16,19,22 。用随机函数首先将 B 类车中编号打乱，比如为 7,2,5,15,11,20 则得到相应配载方案为：首先将编号为 7 的汽车装载在 1-2 型车重最长的车下层，再将编号为 2 汽车装载在 1-2 型车下层（7 号车后面）依次类推。B 类车装完再依据算法依次将 C 类、A 类汽车中序号随机打乱，装载在相应位置。据此形成一种可行解。

随后对可行解不断采用换位法[9]形成其他可行解，多次迭代算法后将其中最优解及相关答案输出，作为装载方案。

5.3.3 基于禁忌搜索算法的模型求解

通过 visual studio 2010 编程环境，用 C 与 C#语言进行混合编程对，迭代步数为 400，每次迭代共搜索当前解的 50 个邻居，禁忌长度取 50，对不能装载的汽车惩罚权重取 100.程序结束后得到如下结果（程序输出接口为 EXCEL，具体程序见附录）：

	A	B	C	D
1	问题五求解程序运行结果			
2	较用车序号	较用车编号	承载的汽车	目的地
3	3	0	,942,943,944,12,12,329,330,589,590	C
4	3	1	,592,605,606,591，,604,621,757,758,	C
5	3	2	5,325,326,327,4,,608,609,610,607	C
6	3	3	,623,624，,604,621,757,758,	C
7	3	4	587,588,602,,626,763,764,765,625	C
8	3	5	,582,583,584,945,946,947,948,949	C
9	3	6	,951,952,953,954,955,950，45	C
10	3	7	,957,958,959,960,961,956，123	C
11	3	8	,963,964,84,84,6,7,8,593	D
12	3	9	612,622,,595,613,614,594	D
13	3	10	,628,629,630,631，781,328	D
14	3	11	,633,767,768,769,632,,2376,777,77	D
15	3	12	,771,772,773,774,770,876,653	D
16	3	13	600,267,268,2,78,965,775,12	D
17	3	14	,967,968,969,970,971567	D
18	3	15	,15,15,16,9,439,440,59678	E
19	3	16	,601,10,10,88,289,,615,616,617,59723	E

图 5-15　问题五程序输出结果

问题五的求解程序输入输出接口均为 EXEL。根据图 5-14，可以清楚展现，为了完成 C 点任务需求，采用 3 序号轿运车进行运载。其中 3 序号车中 0 编号轿运车承载汽车编号为,942,943,944,12,12,329,330,589,590。序号 1 编号轿运车上承载汽车编号为 592,605,606,591,604,621,757,758。依次类推得到所有装载方案。最后查询承运车型号字典，得到表 5-12 格式的装载计划表。

5.3.4　结果分析与优化

5.3.4.1　　结果分析

模型求解结果如下：

表 5-12　　A 点运输方案

承载车辆	装载的汽车	具体装载计划	
		下层	上层
十九位双桥双轮框架 2-2 型　　第 1 辆	497-499,703-705,104-112,647-648	路宝、派力奥 F0	F0、自由舰
十九位双桥双轮框架 1-2 型　第 1 辆	649,1016-1020，216-219,414-418，	赛拉图	自由舰、威志三厢、利亚纳
十九位双桥双轮框架 1-2 型　第 2 辆	419，429，792-794	赛拉、图朗逸	利亚纳、骏逸
十七位双桥双轮框架 1-2 型　第 1 辆	795-798,220,374-380	朗逸	雅绅特
十七位双桥双轮框架 1-2 型　第 2 辆	799-803,22-31	朗逸	雅绅特、马自 2、劲翔得利卡
十七位双桥双轮框架 1-2 型　第 3 辆	804-807,32,33,237-241,441,442	朗逸	得利卡、CITY 锋范
十二位双桥双轮厢式	443，446-449,450-457	嘉誉	嘉誉、福美来

1-1 型　第 1 辆			
十二位双桥双轮厢式 1-1 型　第 2 辆	320-325,533-536	凯越	凯越、福美来
十二位双桥双轮厢式 1-1 型　第 3 辆	890-898,413	天语 SX4	天语 SX4、思域
十二位双桥双轮厢式 1-1 型　第 4 辆	289-294,332-335	思域	思域
十二位双桥双轮厢式 1-1 型　第 5 辆	336-345	思域	速腾
``	```	```	```

备注：完整运输方案见附录

经实验计算可知,在不同禁忌长度条件下(禁忌长度取值 30-80),算法均能收敛至相同的目标值,只是具体装载方案及运算时间有一定区别,说明算法具有较好的稳定性。图 5-16 所示为迭代步数 400,禁忌长度 50 时的收敛过程示意图

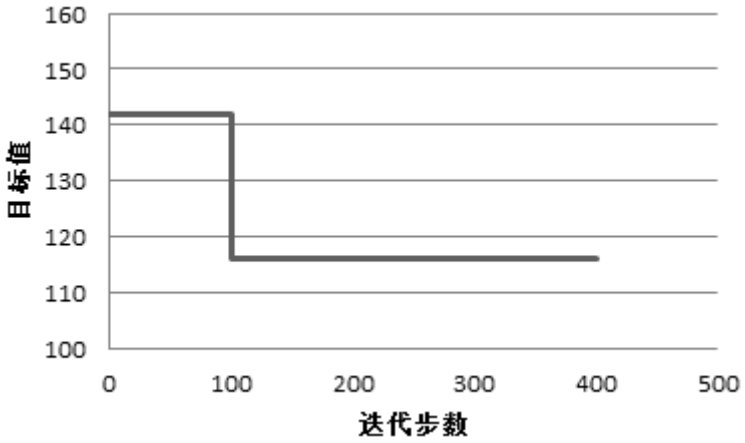


图 5-16　禁忌搜索收敛过程示意图

5.3.4.2　　结果优化

禁忌搜索算法的优化方案，得出了独立完成各个点任务时较优的方案。但是仍然可以进一步综合考虑 A、B、C、D、E 点运送方案，将为离配送起点较远目的地配送的轿运车中仍有装载能力车装满，为他途经的目的地捎带车辆，据此减少轿运车总数以及行驶路程。

首先,依据第 5.1.2 章介绍的空间利用率的计算方法,可计算出为各个点配送车辆的最后一辆轿运车空间利用率

表 5-13　各个配送点最后一辆车空间利用率

目的地	A	B	C	D	E	合计
轿运车数量数	28	22	24	23	21	118
最后一辆配送车空间利用率	62.3%	93.2%	88.6%	34．2%	20．6%	--

由表 5-13 可知，B 点最后一辆车空间利用率高，不好为它途经的节点带车。A、E、C 节点最后一辆车空间率稍低，存在装车空间，可以为比它们途经的 D

点带车。经计算为 A、C、E 配送的最后一辆轿运车可以分别为 D 点带 4 辆、8 辆、5 辆承运车，使为 D 点配送的轿运车降低了两辆 1-1 型轿运车。

表 5-14　优化后的配送方案总表

目的地	A	B	C	D	E	合计
轿运车数量	28	22	24	21	21	116
空间利用率	93.3%	96.2%	92.1%	90.2%	93.1%	--
总行程数	10080	6160	6608	3360	8064	34272

可见基于各个配送点最后一辆车空间利用率计算结果，综合 A、B、C、D、E 点配送方案，能进一步改善模型效果。

六 模型评价与拓展

6.1 模型评价

1.精确性：问题一二三四建立的数学模型具有一定通用性与健壮性，可以精确求解出最优配送方案。而问题五也通过数据预处理与优化，使得配载方案几乎达到最优解。

2.效率高：对问题一二三四问题规模不大，不存在效率问题。对问题五，模型首先对输入数据进行预处理，优化了输入解空间。

3.可测量性：模型结果明确，模型效果优劣可以通过方案的调用轿运车总数、轿运车空间利用率、总行程数进行评价。

4.层次性：模型抓住主要问题。首先优化总轿运车使用量，再优化 1-2 型轿运车与 1-1 型轿运车配比，最后再优化总行驶里程数。层层递进，分步骤优化，逐步达到最优解，并对最终结果进行了综合优化。

缺点：

5.稳定性。对问题五，通过过滤掉明显效果不佳的可行解，减少程序输入量与开销，并且使得模型结果具有稳定性，一定得到一个不太差的较优解。

6.2 模型拓展

对问题一二三，论文建立了通用数学模型，对同类型运输问题具有一定实用性，将模型应用在其他场景尚需要修改一些参数。对问题四，先建立了针对问题的精确求解模型。随后针对现实状况的复杂性建立了基于遗传算法的推广模型。对问题五，在数据预处理与基于禁忌搜索启发式算法求解基础上，通过计算各个节点最后一辆轿运车空间利用率，综合局部微调运载方案。模型优化思路有效，但是仍然有些许笨拙，仍有进一步研究空间。

七 参考文献

[1] 易衍孜. 二维不规则件排样系统的研究与实现[D]. 华南理工大学 2012
[2]黄川.集装箱装载优化算法研究[D].福建师范人学,2005
[3]周必水, 李旭东. 有色 Petri 网在系统分析中的应用 [J].系统仿真学报,2003,15(增刊):102-111.
[4] 田乃硕.休假随机服务系统[M].北京:北京大学出版社,2001.
[5] 肖兵, 瞿坦, 王明哲. 着色 Petri 网及其在系统建模与仿真中的应用[J]. 计算机工程,2001,27(1):30-32.
[6] 江志斌. Petri 网及其在制造系统建模与控制中的应用 [M]. 北京:机械工业出版社,2004.
[7] 朱连章,张红霞. 基于着色 Petri 网的电子商务工作流建模[J].中国石油大学报,2006,30(4):140-144.
[8]丁华福,姜晓伟,王丽雪. 基于禁忌搜索的自适应粒子群算法[J]. 计算机技术与发展. 2010(04)
[9]George J A,Robinson D F.A heuristic for packing boxes into a container. Computers and Operations Research . 2008

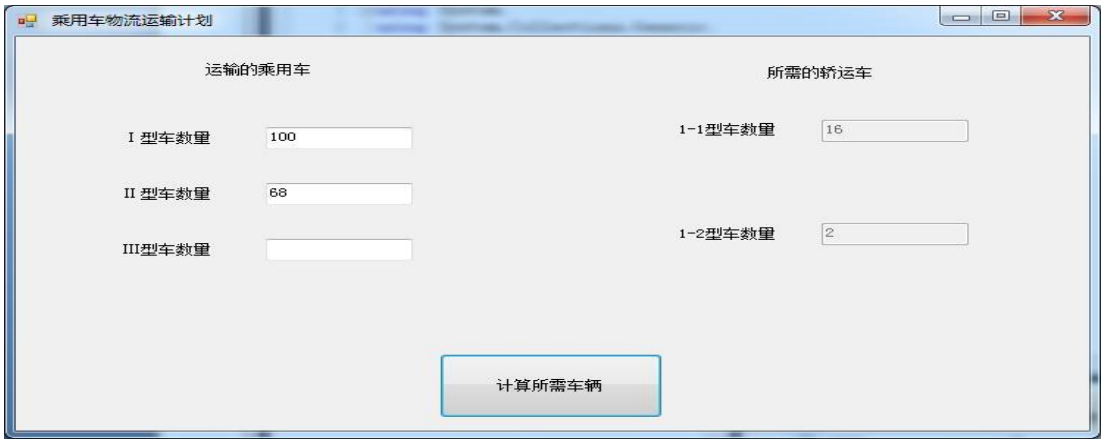
八 附录

附录 I 程序运行界面与结果

1）主程序运行界面

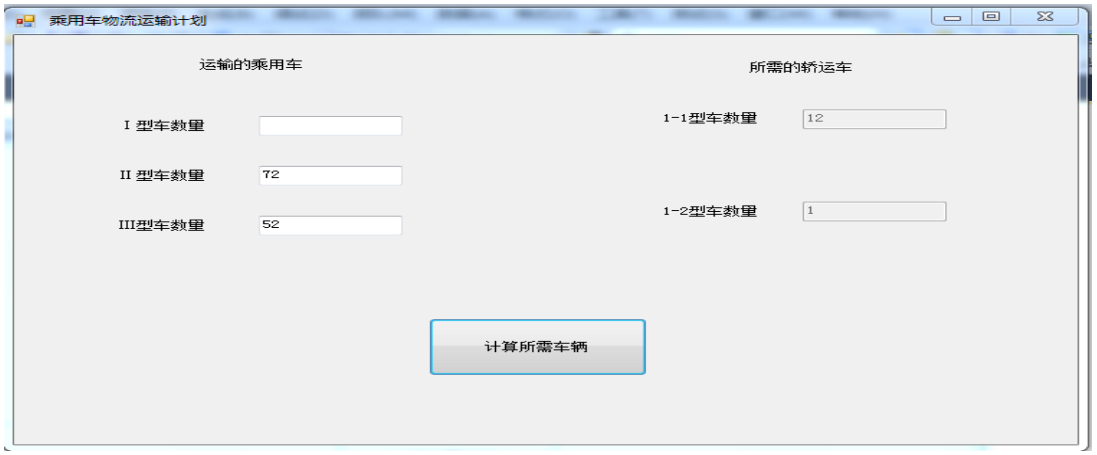
1-1 图 主程序运行界面

2）第一题运行界面与结果



1-2 图 第一题运行结果

3）第二题运行界面与结果

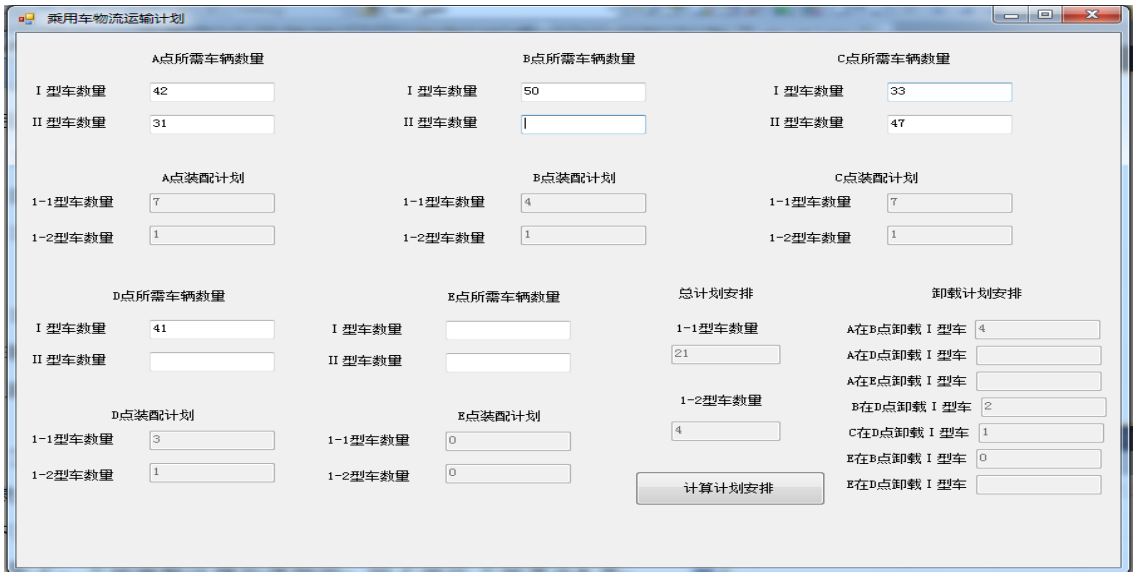


1-3 图 第二题运行结果

4）第三题运行界面与结果

1-4 图 第三题运行结果

5）第四题运行界面与结果

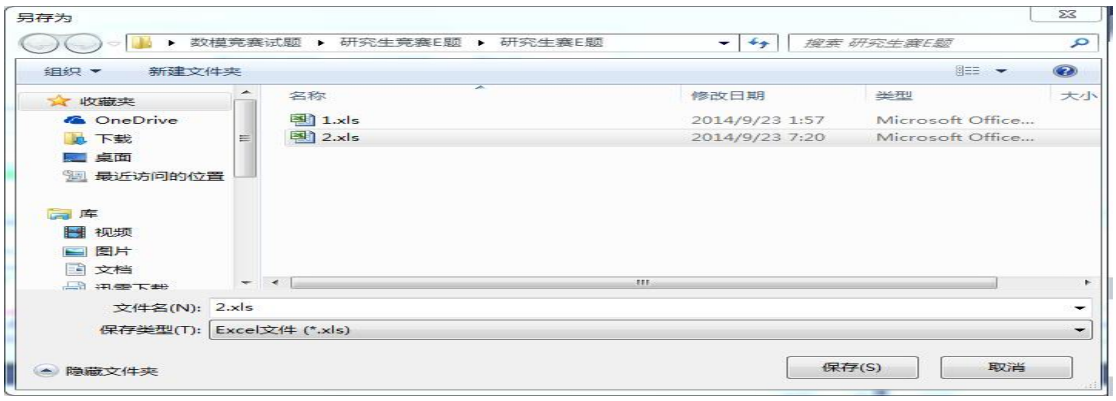


1-5 图 第四题运行结果

6） 第五题运行界面与结果

启动程序

导入数据



运行结果

	A	B	C	D
1			问题五求解程序运行结果	
2	较用车序号	较用车编号	承载的汽车	目的地
3	3	0	,942,943,944,12,12,329,330,589,590	C
4	3	1	,592,605,606,591，,604,621,757,758,	C
5	3	2	5,325,326,327,4,,608,609,610,607	C
6	3	3	,623,624，,604,621,757,758,	C
7	3	4	587,588,602,,626,763,764,765,625	C
8	3	5	,582,583,584,945,946,947,948,949	C
9	3	6	,951,952,953,954,955,950，45	C
10	3	7	,957,958,959,960,961,956，123	C
11	3	8	,963,964,84,84,6,7,8,593	D
12	3	9	612,622,,595,613,614,594	D
13	3	10	,628,629,630,631，781,328	D
14	3	11	,633,767,768,769,632,,2376,777,77	D
15	3	12	,771,772,773,774,770,876,653	D
16	3	13	600,267,268,2,78,965,775,12	D
17	3	14	,967,968,969,970,971567	D
18	3	15	,15,15,16,9,439,440,59678	E
19	3	16	,601,10,10,88,289,,615,616,617,59723	E
20	3	17	,619,620,634,635	E

1-6 图 第五题运行结果

附录II主要代码（完整代码参加附录）

1）1-3问主要代码

```
if (z - 4 > 0) { c2 = c2 + 1; flag = 1; count2 = count2 + 1; }
                    else if ((x - (4 - z) - 4) > 0) { c2 = c2 + 1; flag = 1; count2 = count2 + 1; }
else if ((y - (4 - x) - 5) > 0) { c2 = c2 + 1; flag = 1; count2 = count2 + 1; }
                    else { c1 = c1 + 1; x = 0; y = 0; z = 0; }
                    if (flag == 1)
                    {
                        // if (z < 9 && z > 4)
                        // {
                        if (x == 0 && y < 11) { c2 = c2 - 1; c1 = c1 + 2; x = y = z = 0; break; }
                        else if (y == 0 && x < 9) { c2 = c2 - 1; c1 = c1 + 2; x = y = z = 0;break; }
                        else if (x > 0 && y > 0 && (x + y) < 9) { c2 = c2 - 1; c1 = c1 + 2; x = y = z = 0;
break; }
                        if (z > 5)
                        {
                            z = z - 5;
                            if (x <= 0) { y = y - 12; x = 0; }
```

```
            else if (y <= 0) { x = x - 10; y = 0; }
            else if (x >= 4 && y >= 8) { x = x - 4; y = y - 8; }
            else if (x < 4) { y = y - (5 - x) - 6; x = 0; }
            else if (y < 5) { x = x - (5 - y) - 6; y = 0; }
            else if (y < 7) { x = x - 5; y = 0; }
            else if (y < 9) { x = 4 - x; y = 0; }
        }
        else if (z > 0 && z < 5)
        {
            if (x <= 0) { y = y - (5 - z) - 12; x = 0; }
            if (y <= 0) { x = x - (4 - z) - 10; y = 0; }
            if (x > 0 && y > 0)
            {
                x = x - (5 - z);
                if (x >= 4 && y >= 8) { x = x - 4; y = y - 8; }
                else if (x < 4) { y = y - (5 - x) - 6; x = 0; }
                else if (y < 5) { x = x - (5 - y) - 6; y = 0; }
                else if (y < 7) { x = x - 5; y = 0; }
                else if (y < 9) { x = 4 - x; y = 0; }
            }
        }
```

2）问题 4 主要代码

```
int allc = Convert.ToInt16(cc1) + Convert.ToInt16(cc2);
                //int alla=Convert.ToInt16(ac1)+ Convert.ToInt16(bc1)+ Convert.ToInt16(ac2)+
Convert.ToInt16(bc2);
                int alla = Convert.ToInt16(ac1) + Convert.ToInt16(bc1) + Convert.ToInt16(cc1) +
Convert.ToInt16(dc1) + Convert.ToInt16(ec1);
                int allb = Convert.ToInt16(ac2) + Convert.ToInt16(bc2) + Convert.ToInt16(cc2) +
Convert.ToInt16(dc2) + Convert.ToInt16(ec2);

                tba11.Text = ac1.ToString(); tbb11.Text = bc1.ToString(); tbc11.Text = cc1.ToString();
tbd11.Text = dc1.ToString(); tbe11.Text = ec1.ToString();
                tba22.Text = ac2.ToString(); tbb22.Text = bc2.ToString(); tbc22.Text = cc2.ToString();
tbd22.Text = dc2.ToString(); tbe22.Text = ec2.ToString();
                sum1.Text = alla.ToString(); sum2.Text = allb.ToString();
                tba.Text = ya.ToString(); tbb.Text = yb.ToString(); tbc.Text = yc.ToString(); tbe.Text =
ye.ToString();
            }
    public int car(int x, int y)
            {
                int x1 = x;
                int y1 = y;
```

```
int count = 0, flag = 0,left=0,r1=0,r2=0;
while (x > 0 || y > 0)
{
    if (count == 5)
    {
        if (x - 8 > 0) { r2 = r2 + 1; flag = 1; }
        else if ((y - (8 - x)) > 0) { r2 = r2 + 1; flag = 1; }
        if (flag == 1) { x = x - 6; y = y - 12; r1 = r1 + 1; }
        count = 0; flag = 0;
    }
    else if (count < 5)
    {
        if (x - 8 > 0)
        {
            x = x - 8; r1 = r1 + 1; count = count + 1;
                                        }
        else
        {
            y = y - (8 - x);
            if (y > 0)
            { y = y - 10; r1 = r1 + 1; count = count + 1; x = 0; }
            else if (x > 0) { r1 = r1 + 1; x = 0; }
```

3）问题五主要代码

```
for (a1 = 0; a1 < num; a1++)
        {int p = destination[a1];
            if (place[p] == 0)
            {
                if (flag[p] == 2)
                {
                    place[p] = 2;
                    if (fs(p, 0, 0, des) == 0)
                        ff(p, 0, 0, des);
}}}
if (three_ii == 1)
    {for (a1 = 0; a1 < num; a1++)
            {int p = destination[a1];
                if (place[p] == 0)
                {if (flag[p] == 3)
                    {if (two_ii <= count_two_ii)
                        {if (ff(p, 0, 1, des) == 1)
                            { place[p] = 1; }
                            else if (ff(p, 0, 1, des) == 0)
                            { double_ff(p, des); }}
                        else double_ff(p, des) ;
```

```
                    }}}}
            for (a1 = 0; a1 < num; a1++)
            {int p = destination[a1];
                if (place[p] == 0)
                {if (flag[p] == 1)
                    {if (two_ii <= count_two_ii)
                        {if (ff(p, 0, 1, des) == 1)
                            { place[p] = 1; }
                            else if (ff(p, 0, 1, des) == 0)
                            { double_tt(p, des); }
                        }
                        else if (double_tt(p, des) == 0)
                        {double_ff(p, des);
                                double_ft(p, des);
                                double_tt(p, des);
                                if (double_ff(p, des) == 0)
                                    double_ft(p, des);
        }}}}
```

附录III　问题五详细装载方案

A点

承载车辆	装载的汽车	具体装载计划	
		下层	上层
十九位双桥双轮框架2-2型　　第1辆	497-499,703-705,104-112,647-648	路宝、派力奥F0	F0、自由舰
十九位双桥双轮框架 1-2型　第1辆	649,1016-1020，216-219,414-418，	赛拉图	自由舰、威志三厢、利亚纳
十九位双桥双轮框架 1-2型　第2辆	419，429，792-794	赛拉、图朗逸	利亚纳、骏逸
十七位双桥双轮框架 1-2型　第1辆	795-798,220,374-380	朗逸	雅绅特
十七位双桥双轮框架 1-2型　第2辆	799-803,22-31	朗逸	雅绅特、马自达2、劲翔得利卡
十七位双桥双轮框架 1-2型　第3辆	804-807,32,33,237-241,441,442	朗逸	得利卡、CITY锋范
十二位双桥双轮厢式 1-1型　第1辆	443，446-449,450-457	嘉誉	嘉誉、福美来
十二位双桥双轮厢式 1-1型　第2辆	320-325,533-536	凯越	凯越、福美来
十二位双桥双轮厢式 1-1型　第3辆	890-898,413	天语SX4	天语SX4、思域
十二位双桥双轮厢式 1-1型　第4辆	289-294,332-335	思域	思域
十二位双桥双轮厢	336-345	思域	速腾

式 1-1型　第5辆			
十二位双桥双轮厢式 1-1型　第6辆	346,1101-1103	速腾	标致307两厢
十二位双桥双轮厢式 1-1型　第7辆	984-990,834-836	标致307两厢	标致307两厢、帕萨特
十二位双桥双轮厢式 1-1型　第8辆	837-843,626-628	帕萨特	帕萨特、领航者CUV
十二位双桥双轮厢式 1-1型　第9辆	629,158-166	领航者CUV、F8	F8
十二位双桥双轮厢式 1-1型　第10辆	167，918-922,1076-1099	F8	F8科鲁兹、皇冠
十二位双桥双轮厢式 1-1型　第11辆	1100-1104,1154-1158	皇冠	马自达6
十二位双桥双轮厢式 1-1型　第12辆	1159-1168	马自达6	马自达6
十二位双桥双轮厢式 1-1型　第13辆	401-404,300-305	天籁志翔	志翔
十二位双桥双轮厢式 1-1型　第14辆	305-311,564-567	志翔	志翔、尊驰
十二位双桥双轮厢式 1-1型　第15辆	569,671-678	尊驰、江淮宾悦	江淮宾悦
十二位双桥双轮厢式 1-1型　第16辆	679-672,551-554	江淮宾悦	325i
十二位双桥双轮厢式 1-1型　第17辆	62-71	索纳塔	索纳塔
十二位双桥双轮厢式 1-1型　第18辆	72-76,1-4,226	索纳塔	大切诺基、福克斯三厢
十二位双桥双轮厢式 1-1型　第19辆	227-269,483-486,1026-1028	福克斯三厢、雅阁	雅阁、奥迪A6
十二位双桥双轮厢式 1-1型　第20辆	1029-1037	奥迪A6	奥迪A6
十二位双桥双轮厢式 1-1型　第21辆	11-12,585-589	克莱斯勒300C	华翔驭虎
十二位双桥双轮厢式 1-1型　第22辆	590-592,1151-1152	华翔驭虎	红旗旗舰加长豪华型
28			

注：其他BCDE点装载方案见电子版论文