

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：

本文对乘用车物流运输计划问题进行了研究，针对不同情境，将问题抽象成不同维度的箱柜装载问题进行建模，并取得了以下几方面的成果。

针对问题 1,2,3: 主要采用约束编程的建模思想，建立了适用于多种尺寸的乘用车、轿运车的 3 维乘用车装载数学模型，并获得了在给定乘用车车型数量的前提下，最优的运载分配策略。

结论 (1): 在题中给定的约束条件下，运载 100 辆 I 车型的乘用车及 68 辆 II 车型的乘用车，至少需 16 辆 1-1 型乘用车及 2 辆 1-2 型乘用车。

结论 (2): 在题中给定的约束条件下，运载 72 辆 II 车型的乘用车及 52 辆 III 车型的乘用车，至少需 12 辆 1-1 型乘用车及 1 辆 1-2 型乘用车。

结论 (3): 在题中给定的约束条件下，运载 156 辆 I 车型的乘用车及 102 辆 II 车型的乘用车，至少需 25 辆 1-1 型乘用车及 5 辆 1-2 型乘用车。

针对问题 4: 进一步优化了问题 1,2,3 中的数学模型，完善并支持了考虑行车路线问题的轿运车运输乘用车问题，并给出了通用算法和程序解决了乘用车装载方案及行车路线方案。

结论：在题中给定的约束条件及运输路线下，运输 166 辆 I 车型的乘用车和 78 辆 II 车型的乘用车至少需 22 辆 1-1 型乘用车及 4 辆 1-2 型乘用车。

针对问题 5：在乘用车、轿运车类型、尺寸、数量、目的地更多样化的情境下，持续优化了问题 4 中建立的 3 维乘用车装载模型，以支持复杂的乘用车装载问题的建模和分析需求，并通过经验分析增加了必要的约束条件以缩小解空间，以保证在更短时间内求得较优解。

本文综合运用数理统计、分支限界等数学方法，多维装箱问题的建模思想，借助 Gecode 系统建立了多种满足约束编程思想的数学模型，并对所提出问题进行了研究，具有很好的实用性与推广性。最后，总结了模型的优点与不足，为后续研究此类问题的学者提供了一个新的思路。

关键字：约束编程，多维装箱问题，物流调度

目录

1	问题重述	4
1.1	提出问题	4
1.2	问题要求	4
2	模型假设	5
3	基本符号说明	5
4	问题分析	6
4.1	问题一、二、三分析	7
4.2	问题四分析	7
4.3	问题五分析	8
5	技术背景	9
5.1	约束编程 (Constraint Programming)	9
5.2	Gecode	10
6	模型的建立与求解	10
6.1	模型建立	10
6.1.1	问题一、二、三通用模型	10
6.1.2	问题四模型	15
6.1.3	问题五模型	17
6.2	模型求解	20
6.2.1	问题一、二、三求解	20
6.2.2	问题四求解	21
6.2.3	问题五求解	22
7	模型的评价、改进及推广	23
A	附录：代码	25

1 问题重述

1.1 提出问题

近年来我国汽车市场每年平均以 25% 的速度快速增长，其发展速度快于国民经济的增长。汽车消费需求的增长，促进了乘用车的整车物流量迅速上升。乘用车生产厂家根据全国客户的购车订单，向物流公司下达运输乘用车到全国各地的任务，物流公司则根据下达的任务制定运输计划并配送这批乘用车。为此，物流公司首先要从他们当时可以调用的“轿运车”中选择出若干辆轿运车，进而给出其中每一辆轿运车上乘用车的装载方案和目的地，以保证运输任务的完成。但因轿运车、乘用车规格的多样性和运输路线的多样性，当前很多物流公司在制定运输计划时主要依赖调度人员的经验。而在运输任务较复杂时，往往会发生运输效率低下，运输成本不理想等问题。物流公司希望能够在确保完成运输任务的前提下寻求降低运输成本的方法。“轿运车”是通过公路来运输乘用车整车的专用运输车。本文涉及的双层轿运车分为三种子型：上下层各装载 1 列乘用车，故记为 1-1 型（图 1）；下、上层分别装载 1、2 列，记为 1-2 型（图 2）；上、下层各装载 2 列，记为 2-2 型（图 3），每辆轿运车可以装载乘用车的最大数量在 6 到 27 辆之间。



图 1: 1-1 型轿运车



图 2: 1-2 型轿运车



图 3: 2-2 型轿运车

1.2 问题要求

建立数学模型，以通用算法和程序解决整车物流成本优化的问题，能够比较准确地模拟整车物流的装载方案及行车路线，更好地降低整车物流运输成本。本次竞赛需要解决的具体问题如下所述：问题一：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。问题二：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。问题三：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。问题四：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。问题五：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。

2 模型假设

1. 题目中所列数据均真实可靠；
2. 乘用车与轿运车两侧的安全距离在说明装载区域大小时已经考虑；
3. 乘用车不可在轿运车上倾斜摆放；
4. 轿运车运输路线为有向图；
5. 运输花费主要由轿运车数量决定，次要由轿运车类型决定，最后由轿运车里程数决定。

3 基本符号说明

- N 乘用车总数
- i 乘用车下标, $i \in N$
- K 为轿运车使用量
- k 轿运车下标, $k \in K$
- j 轿运车层级下标, $j \mapsto \{1, 2\}$
- l 乘用车长度
- l_i 第 i 辆乘用车长度
- w 乘用车宽度

- w_i 第 i 辆乘用车的宽度
- h 乘用车高度
- h_i 第 i 辆乘用车的高度
- d 乘用车目的地
- d_i 第 i 辆的目的地 $d_i \mapsto \{A, B, C, D\}$
- D 表示出发地 O 到各目的地的距离
- C 运输成本
- C_k 第 k 辆轿运车的运输成本
- L 轿运车长度
- L_{jk} 第 k 辆轿运车第 j 层的长度
- W 轿运车宽度
- W_{jk} 第 k 辆轿运车第 j 层的宽度
- T 轿运车类型
- T_k 第 k 辆轿运车的类型, $T_k \mapsto \{1, 2, 3 \dots\}$, 其中 1 代表 1-1 型轿运车, 2 代表 1-2 型轿运车, 3 代表 2-2 型轿运车

4 问题分析

本文的基本问题是解决如何分配乘用车到轿运车并使整体运输成本最低的问题。经过对问题本身的分析, 该问题可以视作组合优化问题 (Combinatorial Optimization Problem), 而组合优化问题从计算复杂度上来看属于 NP-困难问题, 即在多项式时间内无法确定能够找到有效算法求解的问题。

本文拟采用约束编程 (Constraint Programming) 的思想进行建模与求解。而对于组合优化的问题, 还需列出对应的最大化或者最小化的目标函数。

4.1 问题一、二、三分析

问题一、二、三要求计算各种类型轿运车的数量和每辆轿运车的乘用车装载方案，各类型乘用车数量已知，所需各类型轿运车数量及各轿运车的装载方案未知，但不涉及运输目的地的差异。并且，问题一、二、三的差异仅为输入参数不同，所以拟建立通用的基于约束编程的模型进行求解。解决该问题一共有三个关键点：

- 采用何种策略将乘用车分配给轿运车；并分配给轿运车的何层。
- 对于已分配的乘用车，如何判断是否满足该轿运车层的长宽约束。
- 如何构造目标函数。
- 采用何种搜索策略能够较快接近并找到最优解。

所以建立的基于约束编程的数学模型，主要是解决以上提出的关键问题。具体的模型建立参考 6.1.1 节。

4.2 问题四分析

问题四要求在考虑行车路线的情况下解决轿运车运输乘用车问题，计算所需各类型轿运车的数量及每辆轿运车的乘用车装载方案。各类型乘用车数量已知，所需各类型轿运车数量及各轿运车的装载方案未知。为解决这一问题，需要在前三题建立的通用模型基础之上，额外考虑运输路线的因素，并添加相关的限制条件。因为我们假设运输路线是一个有向图，所以只会有 $O \rightarrow D \rightarrow C$ 和 $O \rightarrow D \rightarrow B \rightarrow A$ 这两条运输路线。因此，针对路线 $O \rightarrow D \rightarrow C$ ，可以约束只有终点是 D，C 的乘用车可以在一辆轿运车上一次运输。针对路线 $O \rightarrow D \rightarrow B \rightarrow A$ ，可以约束只有终点是 D，B，A 的乘用车可以在一辆轿运车上一次运输。因终点是 D 的乘用车在这两条路线的轿运车里都可以装载，因此可以把装载终点是 D 的乘用车的装载优先级放低。又因行驶路线为有向图，所以在 $O \rightarrow D \rightarrow C$ 这一运输路线中，我们认为终点是 C 的乘用车的运载优先级高于终点是 D 的乘用车。同理可得，终点是 A 的乘用车的运载优先级高于终点是 B 的乘用车，而终点是 B 的乘用车的运载优先级高于终点是 D 的乘用车。各段长度： $OD=160$ ， $DC=76$ ， $DA=200$ ， $DB=120$ 也需要被添加到约束条件里，以将行使里程考虑在行使成本中，半定量化最终的运输成本。因此，对这一问，可以在解决 1-3 问的通用数学模型的基础之上，加上以上这些约束条件以保证满足本题的

运载条件，并获得在给定条件下的最优解。

为了简化模型的求解，我们对该问题做如下处理：

- 在确定分配方案时，限制仅目的地为 $\{C, D\}$ 或 $\{D, B, A\}$ 的组合可以出现在同一轿运车。
- 仅在目的地组合相同的分配方案调整车与目的地的映射关系使得总成本在考虑目的地距离的情况下最低。

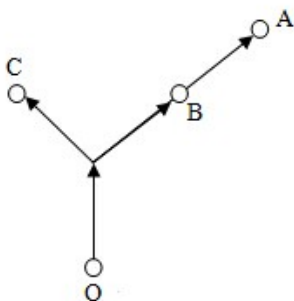


图 4: 问题四涉及目的地及其位置关系

4.3 问题五分析

问题五要求在给定各轿运车、乘用车数量的前提下，综合各车运输路线的情况，解决轿运车运输乘用车的装载问题。。需运输的各类型乘用车数量已知，可使用的轿运车具体型号及数量上限已知，各乘用车运输终点已知，所需各类型轿运车数量及各轿运车的装载方案未知。为解决这一问题，需要在第四题已建立的通用模型基础之上，额外考虑各具体型号轿运车的数量上限，并添加相关的限制条件。为此我们需要扩展轿运车的类型到一个二维变量，以同时包含其类型及具体型号信息。此外，因为我们依然假设运输路线是一个有向图，所以只会有 $O \rightarrow D \rightarrow C$ 和 $O \rightarrow D \rightarrow B \rightarrow A(-\rightarrow E)$ ， $O \rightarrow D \rightarrow B \rightarrow E$ 这三条到达 C, E, A 的运输路线。因此，针对路线 $O \rightarrow D \rightarrow C$ ，可以约束只有终点是 D, C 的乘用车可以在一辆轿运车上一次运输。针对路线 $O \rightarrow D \rightarrow B \rightarrow A(-\rightarrow E)$ 和 $O \rightarrow D \rightarrow B \rightarrow E$ 可以约束只有终点是 D, E, B, A 的乘用车才可以在一辆轿运车上一次运输。因终点是 D 的乘用车在这两条路线的轿运车里都可以装载，因此可以把装载终点是 D 的乘用车的装载优先级放低。又因行驶路线为有向图，所以在 $O \rightarrow D \rightarrow C$ 这一运输路线中，我们认为终点是 C 的乘

用车的运载优先级高于终点是 D 的乘用车。同理可得，终点是 E 的乘用车的运载优先级高于终点是 A 的乘用车，而重点是 A 的乘用车优先级高于终点是 B 的乘用车，终点是 B 的乘用车的运载优先级高于终点是 D 的乘用车。各段长度：OD=160，DC=76，DA=200，DB=120，BE=104，AE=60 也需要被添加到约束条件里，以将行使里程考虑在行使成本中，半定量化最终的运输成本。因此，对这一问，可以在解决第四问的通用数学模型的基础之上，加上以上这些约束条件以保证满足本题的运载条件，并获得在给定条件下的最优解。

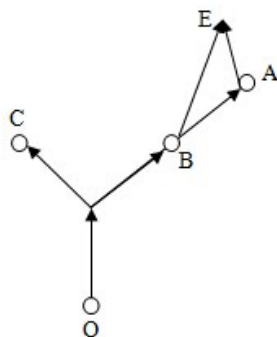


图 5: 问题五涉及目的地及其位置关系

5 技术背景

5.1 约束编程 (Constraint Programming)

约束编程是一种编程范型，在这种编程范型中，变量之间的关系以约束的形式组织。所谓约束，即指的是一个问题的定义域，问题的描述（约束条件）。一般地，约束编程的支持通用的求解程序对约束模型进行求解。过程式语言，如 C、C++ 解决问题的方法是通过给出一组机器指令，一步步完成问题的求解。相比之下，约束编程只需声明问题实体之间的关系，具体求解过程是由系统来自动完成的。约束表示了一组实体之间的关系，约束说明列举了当求解一组变量值时必需满足和维持一致的一组约束。负责把约束说明转化成程序的约束转化体制决定满足约束系统的方式。约束式编程的主要特点有：建模、约束传播和搜索。约束的一个主要功能就是建模，约束是描述许多应用的一种自然语言。约束式语言包含两个方面，定义和行为，可以被分别处理，一旦定义是正确的，就可以得出他们的行为，通过定义和行为的分离，编程人员可以把精力更多地集中于建模。约束传播被

用于两种不同的途径，其一，把一些新的信息加入约束存储中。如加入新约束后，当前的信息和传播结果同以前的信息不一样，那么说明当前约束同已有的约束不一致，应停止当前操作。其二，以自然的方式反映当前决定结果，这种能力常被称为信息传播，传播是对决定立刻反应的重要形式。约束编程最重要的一个应用是用于解决组合问题，在计划、调度、资源分配等复杂问题上的约束编程都干得很出色。组合问题是指那些解是由一系列独立选择的步骤组成，解的正确性不到最后一个决定作出以前是不可知的，通常每次选择都只有有限的备选方案。早期的约束编程应用使用内置的搜索优化方法，如节点一致、弧一致、路径一致算法等，CHIP 中实现的就是弧一致算法。实际证明，该算法能很好地解决调度任务、汽车生产线安排序列等问题。在命令编程语言中，有各种约束编程的类库，如 Choco (Java library, free software: X11 style), Emma (Python library, proprietary), Gecode (C++ library, free software: X11 style) 等。本文涉及的算法及问题求解过程都是基于 Gecode 求解器实现的。

5.2 Gecode

Gecode[1] 是一个基于 c++ 实现的约束编程模型求解器，可用于开发基于约束的系统应用程序，是一个可移植、高效的环境。Gecode 是从根本上进行编程开放，这意味着它可以很容易地与其他系统的接口。它支持新的传播者（如约束的执行情况），分支策略，和搜索引擎编程。新的变量域可以被编程的效率，在有限域和整数集，拿出 Gecode 预定义变量相同的水平。

6 模型的建立与求解

6.1 模型建立

6.1.1 问题一、二、三通用模型

模型参数

- N 乘用车总数, i 表示第 i 辆乘用车, $i \in \{1 \dots N\}$
- K 为轿运车使用量, k 表示第 k 轿运车下标, $k \in \{1 \dots K\}$
- j 轿运车层级, $j \mapsto \{1, 2\}$

- l_i 第 i 辆乘用车长度
- w_i 第 i 辆乘用车的宽度
- h_i 第 i 辆乘用车的高度
- C_k 第 k 辆轿运车的运输成本, $C = \sum_{k=1}^K C_k$
- L_{jk} 第 k 辆轿运车第 j 层的长度
- W_{jk} 第 k 辆轿运车第 j 层的宽度
- T_k 第 k 辆轿运车的类型, $T_k \mapsto \{1, 2, 3\}$

决策变量

- $s_k \mapsto \{0, 1\}$, $k \in K$, 若变量 s_k 为 1, 则表示第 k 辆轿运车已经被使用; 反之, 则第 k 辆轿运车未被使用。
- $x_{ijk} \mapsto \{0, 1\}$, $i \in N$, $k \in K$, $j \in \{1, 2\}$, 若 x_{ijk} 为 1, 则表示第 i 辆乘用车被分配到了第 k 辆轿运车的第 j 层。

目标函数

本问题的目标是最小化已使用的轿运车的数量, 而考虑到题设中 当所使用的轿运车数量相同的, 优先使用耗油量较少的车, 所以在考虑目标函数时, 我们采用对每种不同的车型赋不同的成本值, 即在考虑 1-1 型, 1-2 型和 2-2 型轿运车的情况下, $C_{1-1}=1$, $C_{1-2}=1.0002$, $C_{2-2}=1.0004$ 。通过微调不同车型的对应成本值, 可以保证在优化过程中能够首先考虑轿运车的使用数目, 同时在使用数目相同的情况下使用耗油量更少的轿运车。

$$C = \sum_{k=1}^K C_k \cdot s_k \quad (1)$$

约束条件

- 只有当轿运车确定为使用状态时才能为其分配乘用车

$$x_{ijk} \leq s_k \quad \forall i, j \quad (2)$$

- 所有乘用车必须被分配给某一轿运车

$$\sum_{k=1}^K \sum_{j=1}^2 x_{ijk} = 1 \quad \forall i \quad (3)$$

- 1-2 的使用数目不得超过 1-1 使用数目的 20%

$$\sum_{k=1}^K s_k[T_k = 2] \leq 20\% \cdot \sum_{k=1}^K s_k[T_k = 1] \quad (4)$$

- 对于高度超过 1.7 米的乘用车，只能放在 1-1 或 1-2 型轿运车的下层

$$\forall i \quad h_i > 1.7 \implies \sum_{k=1}^K x_{i1k} = 0 \quad (5)$$

- 对于任一轿运车的任一层，所包含的乘用车必须能够满足该区域的长宽要求

$$\forall k, j \quad x_{ijk} = 1 \implies$$

$$cumulatives(W_{jk}, \{xcoord_i | x_{ijk} = 1\}, \{w_i | x_{ijk} = 1\}, \{xcoord_i | x_{ijk} = 1\}, \{l_i | x_{ijk} = 1\}, L_{jk}) \quad (6)$$

cumulative 约束

在约束编程的范型中，cumulative 约束是一类可以解决在一定资源情限制清况下任务分配的约束。任务用 4-元组 (起始时间, 持续时间, 终止时间, 占用资源) 表示，而资源用 2-元组 (资源, 限制) 表示。对于给定的任务集合 \mathcal{T} ，在单资源的情况下，cumulative 约束强制在任一时间点，任务所占的资源数不超过所限制的阈值，单资源情况下 cumulative 约束所适用的问题如图 6 所示。

而对于多资源的情况，即对一组资源 \mathcal{R} 和一组任务 \mathcal{T} ，任务必须被分配到可用的资源并且对各资源的使用不超过资源的限制。定义多资源情况下的 cumulative 约束原型如下：

$$cumulatives(resource, start, duration, end, height, limit)$$

根据 [2] 中所述思想，我们采用了基于交换的 cumulatives 约束实现，具体算法如图 7 和图 8 所示：

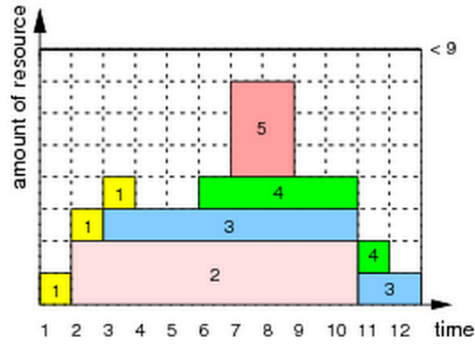


图 6: cumulative 约束

```

1  Set  $nb\_task_r$ ,  $sum\_height_r$ , and  $top\_prune_r$  to 0.
2  Extract the next event  $\&type, task, date, increment\&$ 
3  Set  $d$  to  $date$ .
4  while  $\&type, task, date, increment\& \neq NULL$  do
5      if  $type \neq pruning$  then
6          if  $d \neq date$  then
7              if  $nb\_task_r > 0$  and  $sum\_height_r < Limit[r]$  then fail.
8              Set  $d$  to  $date$ .
9              if  $type = check$  then Add increment to  $nb\_task_r$ , else Add increment to
                 $sum\_height_r$ 
10         else Set  $top\_prune_r$  to  $top\_prune_r + 1$  and set  $stack\_prune_r[top\_prune_r]$  to  $task$ .
11         Extract the next event  $\&type, task, date, increment\&$ 
12     if  $nb\_task_r > 0$  and  $sum\_height_r < Limit[r]$  then fail.

```

图 7: 交换算法 1

搜索策略

为了在解空间的搜索过程中不断最小化目标函数寻找最优解，本问题采用分支定界法 (Branch and Bound) [3] 来寻找模型的最优解。

它是由 R.J. 达金和兰德—多伊格在 20 世纪 60 年代初提出的。分支定界法的基本思想是对有约束条件的最优化问题的所有可行解（数目有限）空间进行搜索。所谓“分支”是采用广度优先的策略，依次生成扩展结点的所有分支（即：儿子结点）。所谓“限界”是在结点扩展过程中，计算结点的上界（或下界），边搜索边减掉搜索树的某些分支，从而提高搜索效率。该算法在具体执行时，把全部可行的解空间不断分割为越来越小的子集（称

```

1  if  $nb\_task_r \neq 0$  and  $sum\_height_r - contribution[t] < Limit[r]$  then.
2      if FIXVAR( $Machine[t], r$ )=fail
3      or ADJUSTMINVAR( $Origin[t], up - \max(Duration[t]) + 1$ )=fail
4      or ADJUSTMAXVAR( $Origin[t], low$ )=fail
5      or ADJUSTMAXVAR( $End[t], low + \max(Duration[t])$ )=fail
6      or ADJUSTMINVAR( $End[t], up + 1$ )=fail
7      or ADJUSTMINVAR( $Duration[t], \min(up - \max(Origin[t]) + 1,$ 
 $\min(End[t] - low))$ )=fail then fail.

```

图 8: 交换算法 2

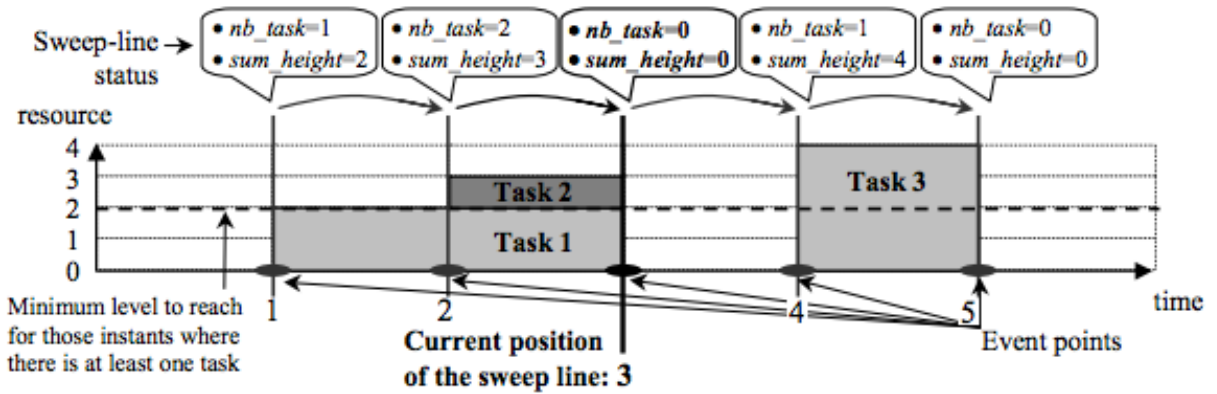


图 9: 交换算法示例

为分支)，并为每个子集内的解的值计算一个下界或上界（称为定界）。在每次分支后，对凡是界限超出已知可行解值的那些子集不再做进一步分支。这样，解的许多子集（即搜索树上的许多结点）就可以不予考虑了，从而缩小了搜索范围。这一过程一直进行到找出可行解为止，该可行解的值不大于任何子集的界限，因此这种算法一般可以求得最优解。

分枝界限法是组合优化问题的有效求解方法，其步骤如下所述：

- 步骤一：如果问题的目标为最小化，则设定目前最优解的值 $Z = \infty$
- 步骤二：根据分枝法则（Branching rule），从尚未被洞悉（Fathomed）节点（局部解）中选择一个节点，并在此节点的下一阶层中分为几个新的节点。
- 步骤三：计算每一个新分枝出来的节点的下限值（Lower bound, LB）

- 步骤四：对每一节点进行洞悉条件测试，若节点满足以下任意一个条件，则此节点可洞悉而不再被考虑：此节点的下限值大于等于 Z 值。已找到在此节点中，具最小下限值的可行解；若此条件成立，则需比较此可行解与 Z 值，若前者较小，则需更新 Z 值，以此为可行解的值。此节点不可能包含可行解。
- 步骤五：判断是否仍有尚未被洞悉的节点，如果有，则进行步骤二，如果已无尚未被洞悉的节点，则演算停止，并得到最优解。

6.1.2 问题四模型

相较于问题一的模型参数，由于问题四引入了目的地和距离信息，所以问题四的模型基于问题一、二、三的通用模型做了在考虑目的地和距离信息情况的扩展。

在模型参数部分加入了 d_i 以表示第 i 辆乘用车的目的地。根据题设已知 $OD=160$, $DC=76$, $DA=200$, $DB=120$ ，且该题所设计始发地均为点 O ，由此可以计算出始发地 O 到各目的地 A 、 B 、 C 、 D 的距离，距离集合 $D = \{D_A = 360, D_B = 280, D_C = 216, D_D = 160\}$ 也作为问题四模型的扩展参数。

增加决策变量 $dest_k$ 表示第 k 辆车所去的目的地。

问题分析中所提及的两个关键点：

1. 在确定分配方案时，限制仅目的地为 $\{C, D\}$ 或 $\{D, B, A\}$ 的组合可以出现在同一辆运输车。
2. 仅在目的地组合相同的分配方案调整车与目的地的映射关系使得总成本在考虑目的地距离的情况下最低。

分别采用增加约束和调整目标函数的方式来实现。

模型参数

- N 乘用车总数， i 表示第 i 辆乘用车， $i \in \{1 \dots N\}$
- K 为运输车使用量， k 表示第 k 运输车下标， $k \in \{1 \dots K\}$

- j 轿运车层级, $j \mapsto \{1, 2\}$
- l_i 第 i 辆乘用车长度
- w_i 第 i 辆乘用车的宽度
- h_i 第 i 辆乘用车的高度
- C_k 第 k 辆轿运车的运输成本, $C = \sum_{k=1}^K C_k$
- L_{jk} 第 k 辆轿运车第 j 层的长度
- W_{jk} 第 k 辆轿运车第 j 层的宽度
- T_k 第 k 辆轿运车的类型, $T_k \mapsto \{1, 2, 3\}$
- d_i 第 i 辆车的目的地, $d_i \mapsto \{A, B, C, D\}$
- 出发地 O 到各目的地的距离所在集合为 $D = \{D_A = 360, D_B = 280, D_C = 216, D_D = 160\}$

决策变量

- $s_k \mapsto \{0, 1\}$, $k \in K$, 若变量 s_k 为 1, 则表示第 k 辆轿运车已经被使用; 反之, 则第 k 辆轿运车未被使用。
- $x_{ijk} \mapsto \{0, 1\}$, $i \in N$, $k \in K$, $j \in \{1, 2\}$, 若 x_{ijk} 为 1, 则表示第 i 辆乘用车被分配到了第 k 辆轿运车的第 j 层。
- $dest_k \mapsto \{A, B, C, D\}$ $k \in K$ $dest_k$ 表示第 k 辆轿运车被分配的目的地。

目标函数

$$C = \sum_{k=1}^K C_k \cdot s_k \cdot D_{dest_k} \quad (7)$$

约束条件

- 只有当轿运车确定为使用状态时才能为其分配乘用车

$$x_{ijk} \leq s_k \quad \forall i, j \quad (8)$$

- 所有乘用车必须被分配给某一轿运车

$$\sum_{k=1}^K \sum_{j=1}^2 x_{ijk} = 1 \quad \forall i \quad (9)$$

- 1-2 的使用数目不得超过 1-1 使用数目的 20%

$$\sum_{k=1}^K s_k[T_k = 2] \leq 20\% \cdot \sum_{k=1}^K s_k[T_k = 1] \quad (10)$$

- 对于高度超过 1.7 米的乘用车，只能放在 1-1 或 1-2 型轿运车的下层

$$\forall i \quad h_i > 1.7 \implies \sum_{k=1}^K x_{i1k} = 0 \quad (11)$$

- 对于任一轿运车的任一层，所包含的乘用车必须能够满足该区域的长宽要求

$$\forall k, j \quad x_{ijk} = 1 \implies$$

$$cumulatives(W_{jk}, \{x_{coord_i} | x_{ijk} = 1\}, \{w_i | x_{ijk} = 1\}, \{x_{coord_i} | x_{ijk} = 1\}, \{l_i | x_{ijk} = 1\}, L_{jk}) \quad (12)$$

- 限制仅目的地为 $\{C, D\}$ 或 $\{D, B, A\}$ 的组合可以出现在同一轿运车，优先级为 $D >_{priority} C, A >_{priority} B >_{priority} C$ ¹

$$\forall i, k \quad x_{ijk} = 1 \Leftrightarrow dest_k \geq_{priority} d_i \quad (13)$$

6.1.3 问题五模型

较之于问题四，问题五主要问题在于复杂的轿运车和乘用车类型以及轿运车的可使用量限制。容易看出乘用车部分的模型能够适应问题五的需求，而由于问题五中涉及的轿运车种类无法用 T 唯一确定，所以增加模型参数 X 用以明确轿运车的序号。同时，移除 1-2 的使用数目不得超过 1-1 使用数目的 20% 的约束条件。

同时，问题五新增了目的地 E，所以根据 $BE=104, AE=60$ ，可以得到新的出发到各目的地距离集合 $D = \{D_A = 360, D_B = 280, D_C = 216, D_D = 160, D_E = 384\}$

¹此次所指优先级理解为若一辆轿运车同时载有去目的地 C 和 D，则将轿运车的目的地视为 C，对于 A, B, C 亦同

模型参数

- N 乘用车总数, i 表示第 i 辆乘用车, $i \in \{1 \dots N\}$
- K 为轿运车使用量, k 表示第 k 轿运车下标, $k \in \{1 \dots K\}$
- j 轿运车层级, $j \mapsto \{1, 2\}$
- l_i 第 i 辆乘用车长度
- w_i 第 i 辆乘用车的宽度
- h_i 第 i 辆乘用车的高度
- C_k 第 k 辆轿运车的运输成本, $C = \sum_{k=1}^K C_k$
- L_{jk} 第 k 辆轿运车第 j 层的长度
- W_{jk} 第 k 辆轿运车第 j 层的宽度
- T_k 第 k 辆轿运车的类型, $T_k \mapsto \{1, 2, 3\}$
- d_i 第 i 辆车的目的地, $d_i \mapsto \{A, B, C, D, E\}$
- 出发地 O 到各目的地的距离所在集合为 $D = \{D_A = 360, D_B = 280, D_C = 216, D_D = 160, D_E = 384\}$
- idx 表示第 idx 序的轿运车, $idx \mapsto \{1, 2, \dots, 10\}$
- a_{idx} 表示第 idx 序的轿运车累计可使用的数目。

决策变量

- $s_k \mapsto \{0, 1\}$, $k \in K$, 若变量 s_k 为 1, 则表示第 k 辆轿运车已经被使用; 反之, 则第 k 辆轿运车未被使用。
- $x_{ijk} \mapsto \{0, 1\}$, $i \in N$, $k \in K$, $j \in \{1, 2\}$, 若 x_{ijk} 为 1, 则表示第 i 辆乘用车被分配到了第 k 辆轿运车的第 j 层。
- $dest_k \mapsto \{A, B, C, D, E\}$ $k \in K$ $dest_k$ 表示第 k 辆轿运车被分配的目的地。
- I_k 为第 k 辆车所分配轿运车对应的序号

目标函数

问题五所需最小化的目标函数与问题四相同。

$$C = \sum_{k=1}^K C_k \cdot s_k \cdot D_{dest_k} \quad (14)$$

约束条件

- 只有当轿运车确定为使用状态时才能为其分配乘用车

$$x_{ijk} \leq s_k \quad \forall i, j \quad (15)$$

- 所有乘用车必须被分配给某一轿运车

$$\sum_{k=1}^K \sum_{j=1}^2 x_{ijk} = 1 \quad \forall i \quad (16)$$

- 1-2 的使用数目不得超过 1-1 使用数目的 20%

$$\sum_{k=1}^K s_k [T_k = 2] \leq 20\% \cdot \sum_{k=1}^K s_k [T_k = 1] \quad (17)$$

- 对于高度超过 1.7 米的乘用车，只能放在 1-1 或 1-2 型轿运车的下层

$$\forall i \ h_i > 1.7 \implies \sum_{k=1}^K x_{i1k} = 0 \quad (18)$$

- 对于任一轿运车的任一层，所包含的乘用车必须能够满足该区域的长宽要求

$$\forall k, j \quad x_{ijk} = 1 \implies$$

$$cumulatives(W_{jk}, \{xcoord_i | x_{ijk} = 1\}, \{w_i | x_{ijk} = 1\}, \{xcoord_i | x_{ijk} = 1\}, \{l_i | x_{ijk} = 1\}, L_{jk}) \quad (19)$$

- 限制仅目的地为 $\{C, D\}$ 或 $\{D, E, B, A\}$ 的组合可以出现在同一轿运车, 优先级为 $D >_{priority} C, A >_{priority} E >_{priority} B >_{priority} C$

$$\forall i, k \ x_{ijk} = 1 \Leftrightarrow dest_k \geq_{priority} d_i \quad (20)$$

- 各序号的轿运车使用量不得超过其累计可使用量

$$\forall idx, \sum_{k=1}^K s_k [I_k = idx] \leq a_{idx} \quad (21)$$

6.2 模型求解

本文所有涉及模型均采用 Gecode[1] 求解

基于上文所介绍的数学模型，我们使用 C++ 建立了数学模型，并对问题进行了求解，本章我们将对问题的答案进行详细的解释说明。

6.2.1 问题一、二、三求解

问题一求解

如表 1所示，针对问题一，运送 100 辆 I 型乘用车以及 68 辆 II 型乘用车，我们模型求解得到的最优结果是，使用 16 辆 1-1 型轿运车以及 2 辆 1-2 型轿运车。相应的装车方案见表 1。实际输出的 csv 文件格式是横向排列的，由于文档格式的限制，本文档对所有的输出答案进行了格式的更改，我们保证答案是一致的，只是将横向表格更改为了纵向表格，此处理适用于本章的所有输出答案表格，以下将不再对格式更改进行说明。表 1中，第一行展示了运输使用的轿运车类型，第二行展示了使用相同类型，相同装载方式的轿运车数量，其中，相同装载方式意味着在乘用车数量以及位置在这些轿运车中是相同的。从第三行到第八行，展示了乘用车在一辆轿运车中的分布情况，包括在轿运车上层的乘用车数量以及在轿运车下层的乘用车数量。针对 1-2 型的轿运车上层，装载方式满足对称装载的需求，以左一辆，右一辆的方式进行装载。第九行以及第十行主要是为了第四问而设计的，为保证程序的通用性，虽然前三问没有用到这两项，我们还是对他们进行了输出，在前三问中这两项对于所有的轿运车都采取置零处理。

表 1: 问题一分配方案详解

轿用车类型	1-1	1-2	1-1	1-2	1-1	1-1
相同类型、相同装载方式的车辆数	5	1	5	1	5	1
装在上层序号为 1 乘用车数量	4	10	4	0	0	0
装在上层序号为 2 乘用车数量	0	0	0	12	5	1
装在上层序号为 3 乘用车数量	0	0	0	0	0	0
装在下层序号为 1 乘用车数量	4	5	4	5	0	0
装在下层序号为 2 乘用车数量	0	0	0	0	5	5
装在下层序号为 3 乘用车数量	0	0	0	0	0	0
中间停靠地	0	0	0	0	0	0
目的地	0	0	0	0	0	0

问题二求解

针对问题二，运送 72 辆 II 型乘用车以及 52 辆 III 型乘用车，我们模型求解得到的最优结果是，使用 12 辆 1-1 型轿运车以及 1 辆 1-2 型轿运车来执行此次运输。具体乘用车在轿运车的分布情况详见表 2。表中各项指标均与问题一中的相同，此处不再赘述。

表 2: 问题二分配方案详解

轿用车类型	1-1	1-2	1-1	1-1	1-1
相同类型、相同装载方式的车辆数	5	1	5	1	1
装在上层序号为 1 乘用车数量	0	0	0	0	0
装在上层序号为 2 乘用车数量	5	12	5	5	4
装在上层序号为 3 乘用车数量	0	0	0	0	0
装在下层序号为 1 乘用车数量	0	0	0	0	0
装在下层序号为 2 乘用车数量	0	0	0	0	1
装在下层序号为 3 乘用车数量	4	5	4	4	3
中间停靠地	0	0	0	0	0
目的地	0	0	0	0	0

问题三求解

针对问题三，运送 156 辆 I 型乘用车、102 辆 II 型乘用车以及 39 辆 III 型乘用车，我们模型求解得到的最优结果是，使用 25 辆 1-1 型轿运车以及 5 辆 1-2 型轿运车来完成这个运输任务。表 3 详细介绍了乘用车在轿运车中的分配方案，表中各项指标均与问题一中的相同，如表中有不明确的地方，请参照问题一中的表中的指标说明。

6.2.2 问题四求解

针对问题四，运送 42 辆 I 型乘用车以及 31 辆 II 型乘用车到 A 地，50 辆 I 型乘用车到 B 地，33 辆 I 型乘用车以及 47 辆 II 型乘用车到 C 地，41 辆 I 型乘用车到 D 地的问题，我们模型求解的最优结果是，使用 6 辆直接开往 A 地的 1-1 型轿运车，1 辆直接开往 A 地的 1-2 型轿运车以及 1 辆中途停留 B 地，最终开往 A 地的 1-1 型轿运车；使用 3 辆直接开往 B 地的 1-1 型轿运车，1 辆直接开往 B 地的 1-2 型轿运车以及 1 辆中途停留 D 地，最终目的地 B 地的 1-1 型轿运车；使用 7 辆直接开往 C 地的 1-1 型轿运车以及 1 辆直接开往 C 地 1-2 型轿运车；使用 4 辆直接开往 D 地的 1-1 型轿运车以及 1 辆直接开往

表 3: 问题三分配方案详解

轿用车类型	1-1	1-2	1-1	1-1	1-2	1-1	1-2	1-1	1-2	1-1	1-1	1-1	1-2
相同类型、相同装载方式的车辆数	5	1	4	1	1	5	1	5	1	1	1	3	1
装在上层序号为 1 乘用车数量	4	4	4	4	4	4	4	4	4	4	0	0	0
装在上层序号为 2 乘用车数量	0	8	0	0	8	0	8	0	8	0	5	5	12
装在上层序号为 3 乘用车数量	0	0	0	0	0	0	0	0	0	0	0	0	0
装在下层序号为 1 乘用车数量	0	0	0	3	2	4	2	4	2	4	3	0	0
装在下层序号为 2 乘用车数量	0	4	0	0	4	0	4	0	4	0	1	5	6
装在下层序号为 3 乘用车数量	4	2	4	1	0	0	0	0	0	0	0	0	0
中间停靠地	0	0	0	0	0	0	0	0	0	0	0	0	0
目的地	0	0	0	0	0	0	0	0	0	0	0	0	0

D 地的 1-2 型轿运车。所有轿运车运输的总里程数为 6968，其中 1-1 型轿运车的里程为 5932,1-2 型轿运车的里程为 1036。详细的轿运车装载乘用车运输方案见表 4。第四文中，输出项，最后两项中间停靠地以及目的地均被使用，中间停靠地意味着该轿运车需要在此地卸一部分货，如果轿运车不需要在中间地卸货，我们就将中间停靠地置为零，虽然轿运车仍然途径其他一些地方，但是这些轿运车不需要在中途卸货。中间停靠地即为中间卸货地。

表 4: 问题四分配方案详解

轿用车类型	1-1	1-1	1-2	1-1	1-1	1-2	1-1	1-1	1-2	1-1	1-1	1-2	1-1
相同类型、相同装载方式的车辆数	5	1	1	1	3	1	1	4	1	3	3	1	1
装在上层序号为 1 乘用车数量	4	0	2	4	4	10	4	4	1	0	4	10	0
装在上层序号为 2 乘用车数量	0	5	10	0	0	0	0	0	11	5	0	0	0
装在下层序号为 1 乘用车数量	4	0	0	0	4	5	4	4	0	0	4	5	1
装在下层序号为 2 乘用车数量	0	5	6	5	0	0	0	0	6	5	0	0	0
中间停靠地	0	0	0	B	0	0	D	0	0	0	0	0	0
目的地	A	A	A	A	B	B	B	C	C	C	D	D	D

6.2.3 问题五求解

本模型是基于约束编程的思想所建立, 由于第五题问题相对复杂, 需要使用较多的约束, 编写这些复杂的约束也是一件复杂的任务, 由于本次比赛时间所限, 约束编程是一个较新的, 较复杂的建模, 编程思想, 我们接触的时间也不是很长, 对此并不是十分精通,

因此，在提交最终作品时，我们并没有完成全部约束条件的编写，我们不能解决如第五题一样复杂的乘用车分配以及运输问题。这是一个非常有趣的问题，我们会继续完善此编程。基于对约束编程的研究，我们有理由认为，使用约束编程来解决此类问题，可以保证在最短的时间内得到问题的最优解，其求解效率高于一般普通算法对此类问题的求解。约束编程在建模以及解模型领域有广阔的前景。

7 模型的评价、改进及推广

本文分别对乘用车、轿运车装载方式进行了建模求解，在数据简单解空间较小的情况下计算获得了问题的最优解；在数据复杂解空间过大无法求解的情况下，也根据实际情况为模型设置了约束条件，缩小了解空间。该模型若能继续完善，深入推广，将极大程度上改善物流公司的调度机制，为公司缩减物流成本，提高运输效率提供有力支持。在建模过程中，我们的模型得到了验证，但也不能忽视所存在需要改进的问题，现分析如下：

- 本模型借助了 Gecode 求解器通用模型框架实现，该框架内的算法性能虽然已经较优，但整体上受框架约束较大。目前的约束实现是基于 Gecode 提供的通用传播者（propagator）和分支程序（brancher），所以该模型的求解效率可以通过自定义领域相关的传播者和分支程序来进一步提高。而对于多资源的情况，即对一组资源 R 和一组任务 T ，任务必须被分配到可用的资源并且对各资源的使用不超过资源的限制。如希望进一步优化提高运算效率，需改造 Gecode 求解器内部算法，但这样在具体实现上将较为繁琐。
- 在问题四中，目前得到的结果可能是一个最优解也可能只是一个较优解。因目前的模型中的约束条件还未全部完善起来，会导致解空间较大的问题产生。在固定时间内仅能得到一个较优解。后续我们也将继续完善这一部分模型，力求获得更优质的装配方案推荐。
- 在问题五中，现虽已针对题中提供的信息完成建模，但很遗憾因时间关系暂未完全实现。今后努力方向：

我们将继续深化对乘用车、轿运车装载问题的研究，自定义领域相关的传播者和分支程序来进一步提高现有的运算效率，在一定解空间范围内更快得到最优结果。希望能够做出更加深入、扎实、优质的研究成果，以解决物流公司在乘用车运载调配方面的难题。

参考文献

- [1] Gecode Team. *Gecode: Generic Constraint Development Environment*, 2006.
- [2] Beldiceanu, Nicolas, and Mats Carlsson. "A new multi-resource cumulatives constraint with negative heights." *Principles and Practice of Constraint Programming-CP 2002*. Springer Berlin Heidelberg, 2002.
- [3] Land, Ailsa H., and Alison G. Doig. "An automatic method of solving discrete programming problems." *Econometrica: Journal of the Econometric Society* (1960): 497-520.