整数规划

分享人：王德民

总结人：从坤

定义

对于限制全部或部分决策变量取离散非负整数值的线性规划，我们称之为整数线性规划，简称为整数规划。

详细分类

类别	定义
纯整数规划	整数规划中，所有决策变量都限制为整数
混合整数规划	整数规划中，一部分决策变量都限制为整数
$0-1$ **整数规划**	整数规划中，决策变量仅限制为 0 或 1

求解方法

方法	求解问题
分枝定界法	可求纯或混合整数线性规划
割平面法	可求纯或混合整数线性规划
隐枚举法	用于求解 $0-1$ 整数规划，有过滤隐枚举法和分枝隐枚举法
匈牙利法	解决指派问题（$0-1$ 整数规划特殊情形）
蒙特卡罗法	求解各种类型规划

指派问题

1.标准指派模型

标准指派问题的一般提法是：拟分派 n 个人 $A_1, A_2, \cdots, A_n$ 去完成 n 项工作 $B_1, B_2, \cdots, B_n$, 要求每项工作需且仅需一个人去完成, 每个人需完成且仅需完成一项工作。已知人 A_i 完成工作 B_j 的时间或费用等成本型指标值为 c_{ij}, 则应如何指派才能使总的工作效率最高?

引入0-1决策变量

$$x_{ij}=\begin{cases}1, & \text{指派 } A_i \text{ 去完成工作 } B_j,\\ 0, & \text{否则},\end{cases} \quad i,j=1,2,\cdots,n.$$

则标准指派问题的数学模型为

$$\min z=\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij},$$

$$\text{s.t.}\begin{cases}\sum_{j=1}^{n}x_{ij}=1, & i=1,2,\cdots,n,\\ \sum_{i=1}^{n}x_{ij}=1, & j=1,2,\cdots,n,\\ x_{ij}=0 \text{ 或 } 1, & i,j=1,2,\cdots,n,\end{cases}$$

这是一个纯 $0-1$ 整数规划模型，可以通过上面通用算法解决，为了提高求解效率，下面介绍匈牙利算法。

匈牙利算法基与以下两个定理：

定理 6.1 设效率矩阵 $C=(c_{ij})_{n\times n}$ 中任何一行 (列) 的各元素都减去一个 常数 k (可正可负) 后得到的新矩阵为 $B=(b_{ij})_{n\times n}$, 则以 $B=(b_{ij})_{n\times n}$ 为效率矩 阵的指派问题与原问题有相同的最优解, 但其最优值比原问题的最优值小 k。

定理 6.2 (独立零元素定理) 若一方阵中的一部分元素为 0 , 一部分 元素为非 0 , 则覆盖方阵内所有 0 元素的最少直线数恰好等于那些位于不同 行、不同列的 0 元素的最多个数。

定理6.1告诉我们如何将效率矩阵中的元素转换为每行每列都有零元素，而定理6.2告诉我们效率矩阵中有多少个独立的零元素。

下面结合具体实例，分析匈牙利算法如何解决任务分配问题。

以 $N=4$ 为实例，下图为 cost 列表和 cost 矩阵。

	Work1	Work2	Work3	Work4
Person1	90	75	75	80
Person2	35	85	55	65
Person3	125	95	90	105
Person4	45	110	95	115

$$\begin{bmatrix} 90 & 75 & 75 & 80 \\ 35 & 85 & 55 & 65 \\ 125 & 95 & 90 & 105 \\ 45 & 110 & 95 & 115 \end{bmatrix}$$

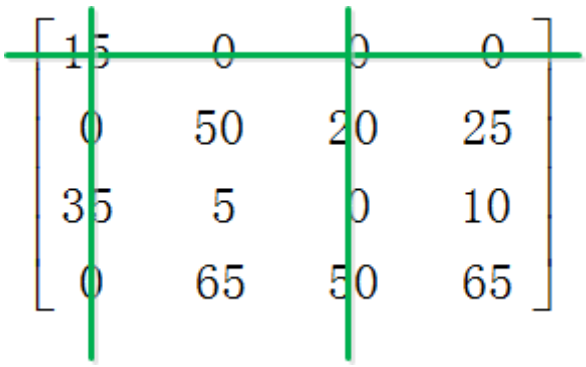
Step1.从第1行减去75，第2行减去35，第3行减去90，第4行减去45。

$$\begin{bmatrix} 90 & \boxed{75} & 75 & 80 \\ \boxed{35} & 85 & 55 & 65 \\ 125 & 95 & \boxed{90} & 105 \\ \boxed{45} & 110 & 95 & 115 \end{bmatrix} \rightarrow \begin{bmatrix} 15 & 0 & 0 & 5 \\ 0 & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{bmatrix}$$

Step2.从第1列减去0，第2列减去0，第3列减去0，第4列减去5。

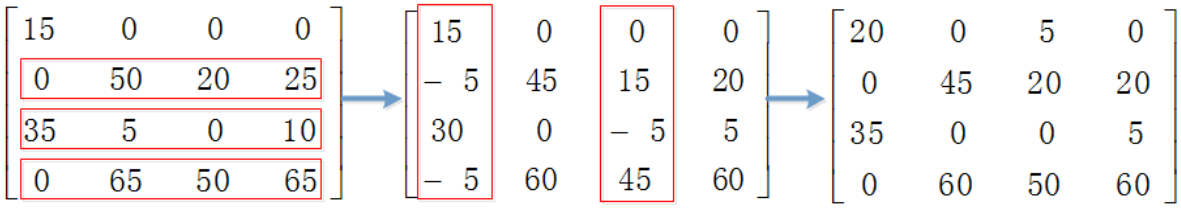
$$\begin{bmatrix} 15 & \boxed{0} & \boxed{0} & \boxed{5} \\ \boxed{0} & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{bmatrix} \rightarrow \begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{bmatrix}$$

Step3.利用最少的水平线或垂直线覆盖所有的0。

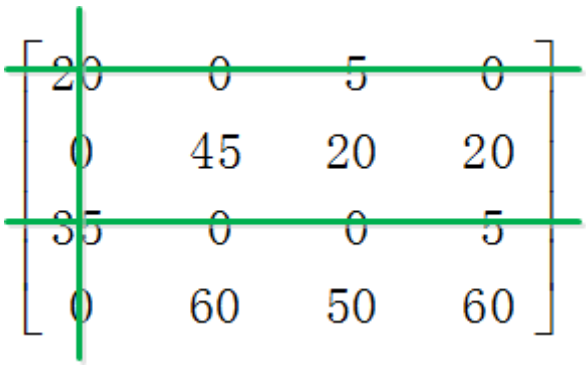


Step4.由于水平线和垂直线的总数是3，少于4，进入Step5。

Step5.没有被覆盖的最小值是5，没有被覆盖的每行减去最小值5，被覆盖的每列加上最小值5，然后跳转到步骤3。

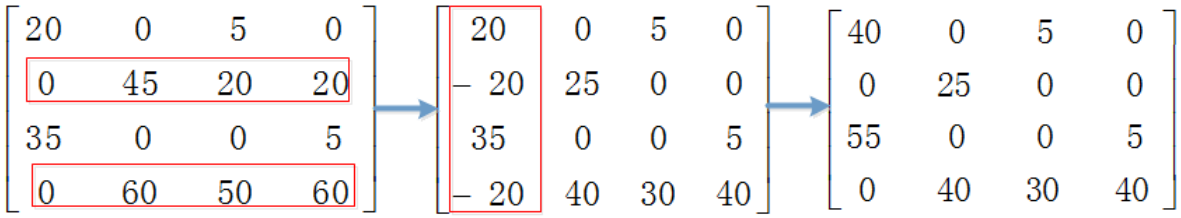


Step3.利用最少的水平线或垂直线覆盖所有的0。

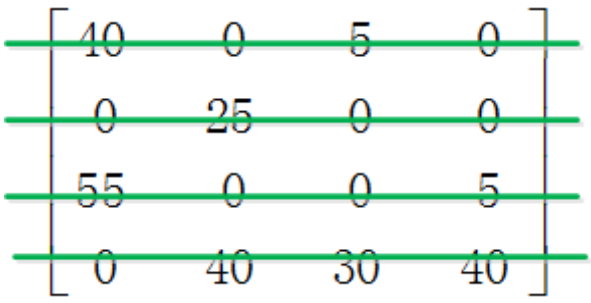


Step4.由于水平线和垂直线的总数是3，少于4，进入Step5。

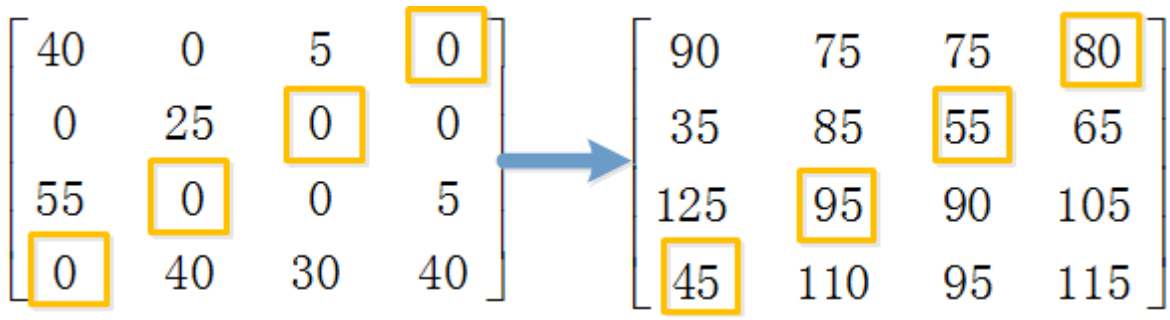
Step5.没有被覆盖的最小值是20，没有被覆盖的每行减去最小值20，被覆盖的每列加上最小值20，然后跳转到步骤3。



Step3.利用最少的水平线或垂直线覆盖所有的0。



Step4.由于水平线和垂直线的总数是4，算法结束，分配结果如下图所示。



其中，黄色框表示分配结果，左边矩阵的最优分配等价于左边矩阵的最优分配。

2.广义指派模型

在实际应用中，常会遇到各种非标准形式的指派问题一广义指派问题。 通常的处理方法是先将它们转化为标准形式, 然后用匈牙利算法求解。

类型	方法
最大化指派问题	用效率矩阵中最大数分别与效率矩阵中每个数作差即化为标准模型
人数和任务数不等的指派问题	添加虚拟人或者虚拟任务
一个人可完成多项任务的指派问题	一人看作多人，只需保证他们完成同一项任务的效率一样
某项任务一定不能由某人完成的指派问题	对应效率取无穷大（足够大）

例子

例 6.1 求解下列整数线性规划问题:

$$\min z = 40x_1 + 90x_2,$$
$$\text{s.t.} \begin{cases} 9x_1 + 7x_2 \leq 56, \\ 7x_1 + 20x_2 \geq 70, \\ x_1, x_2 \geq 0 \text{ 为整数}. \end{cases}$$

解 利用 cvxpy 库, 求得的最优解为 $x_1 = 2, x_2 = 3$; 标函数的最优值为 $z = 350$ 。

```
import cvxpy as cp
from numpy import array
c=array([40,90])  #定义目标向量
a=array([[9,7],[-7,-20]])  #定义约束矩阵
b=array([56,-70])  #定义约束条件的右边向量
x=cp.Variable(2,integer=True)  #定义两个整数决策变量
obj=cp.Minimize(c*x)  #构造目标函数
cons=[a*x<=b, x>=0]    #构造约束条件
prob=cp.Problem(obj, cons)  #构建问题模型
prob.solve(solver='GLPK_MI',verbose =True)  #求解问题
print("最优值为:",prob.value)
print("最优解为: \n",x.value)
```

例 6.2 某商业公司计划开办 5 家新商店, 决定由 5 家建筑公司分别承建。已知建筑公司 $A_i(i = 1, 2, \cdots, 5)$ 对新商店 $B_j(j = 1, 2, \cdots, 5)$ 的建造费用报价 (万元) 为 $c_{ij}(i, j = 1, 2, \cdots, 5)$, 见表 6.1 。为节省费用, 商业公司应当对 5 家建筑公司怎样分配建造任务, 才能使总的建造费用最少?

表6.1 建造费用报价数据

	$\mathbf{B}_1$	$\mathbf{B}_2$	$\mathbf{B}_3$	$\mathbf{B}_4$	$\mathbf{B}_5$
$\mathbf{A}_1$	4	8	7	15	12
$\mathbf{A}_2$	7	9	17	14	10
$\mathbf{A}_3$	6	9	12	8	7
$\mathbf{A}_4$	6	7	14	6	10
$\mathbf{A}_5$	6	9	12	10	6

解 这是一个标准的指派问题。引进 0-1 变量

$$x_{ij}=\begin{cases}1, \text{ 当 } \mathrm{A}_i \text{ 承建 } \mathrm{B}_j \text{ 时}\\ 0, \text{ 当 } \mathrm{A}_i \text{ 不承建 } \mathrm{B}_j \text{ 时}\end{cases}, i,j=1,2,\cdots,5.$$

则问题的数学模型为

$$\min z=\sum_{i=1}^{5}\sum_{j=1}^{5}c_{ij}x_{ij},$$

$$\text{s.t.}\begin{cases}\sum_{j=1}^{5}x_{ij}=1, & i=1,2,\cdots,5,\\ \sum_{i=1}^{5}x_{ij}=1, & j=1,2,\cdots,5,\\ x_{ij}=0 \text{ 或 } 1, & i,j=1,2,\cdots,5.\end{cases}$$

利用 cvxpy 库, 求得的最优解为

$$\begin{bmatrix}1 & 2 & 3 & 4 & 5\\ 3 & 2 & 1 & 4 & 5\end{bmatrix}$$

也就是说，最优指派方案是，让 A_1 承建 B_3, A_2 承建 B_2, A_3 承建 B_1, A_4 承建 B_4, A_5 承建 B_5 。这样安排能使总的建造费用最少, 最小费用为 34 万元。

```
import cvxpy as cp
import numpy as np
c=np.array([[4, 8, 7, 15, 12],
            [7, 9, 17, 14, 10],
            [6, 9, 12, 8, 7],
            [6, 7, 14, 6, 10],
            [6, 9, 12, 10, 6]])
x = cp.Variable((5,5),integer=True)
obj = cp.Minimize(cp.sum(cp.multiply(c,x)))
con= [0 <= x, x <= 1, cp.sum(x, axis=0, keepdims=True)==1,
               cp.sum(x, axis=1, keepdims=True)==1]
prob = cp.Problem(obj, con)
prob.solve(solver='GLPK_MI')
print("最优值为:",prob.value)
print("最优解为：\n",x.value)
```

例 6.3 (装箱问题) 有 7 种规格的包装箱要装到两辆铁路平板车上去。 包装箱的宽和高是一样的, 但厚度 $l(\mathrm{cm})$ 及重量 $w(\mathrm{kg})$ 是不同的, 表 6.2 给出了每种包装箱的厚度、重量以及数量, 每辆平板车有 $10.2\ \mathrm{m}$ 长的地方来装包装箱, 载重量为 $40\mathrm{t}$, 由于当地货运的限制, 对 C_5, C_6, C_7 类的包装箱的总数有一个特别的限制: 这类箱子所占的空间 (厚度) 不能超过 $302.7\ \mathrm{cm}$ 。要 求给出最好的装运方式。

表 6.2 各类包装箱数据

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
l/cm	48.7	52.0	61.3	72.0	48.7	52.0	64.0
w/kg	2000	3000	1000	500	4000	2000	1000
件数	8	7	9	6	6	4	8

解 1.装箱总厚度最大的模型

$$\max z_1 = \sum_{j=1}^{7} l_j \left(x_{1j} + x_{2j}\right),$$

$$\text{s.t.} \begin{cases} \sum_{i=1}^{2} x_{ij} \le a_j, \quad j = 1, 2, \cdots, 7, \\ \sum_{j=1}^{7} l_j x_{ij} \le 1020, \quad i = 1, 2, \\ \sum_{j=1}^{7} w_j x_{ij} \le 40000, \quad i = 1, 2, \\ \sum_{j=5}^{7} l_j \left(x_{1j} + x_{2j}\right) \le 302.7, \\ x_{ij} \ge 0 \text{ 且为整数}, i = 1, 2; j = 1, 2, \cdots, 7. \end{cases}$$

利用 cvxpy 库，可得到问题付最优解：

$$x^* = \left(x_{ij}\right)_{2\times 7} = \begin{bmatrix} 4 & 1 & 5 & 3 & 3 & 2 & 0 \\ 4 & 6 & 4 & 3 & 0 & 1 & 0 \end{bmatrix}, \quad z_1 = 2039.4.$$

```
import cvxpy as cp
import numpy as np
L=np.array([48.7,52.0,61.3,72.0,48.7,52.0,64.0])
w=np.array([2000,3000,1000,500,4000,2000,1000])
a=np.array([8,7,9,6,6,4,8])
x=cp.Variable((2,7), integer=True)
obj=cp.Maximize(cp.sum(x*L))
con=[cp.sum(x,axis=0,keepdims=True)<=a.reshape(1,7),
     x*L<=1020, x*w<=40000, cp.sum(x[:,4:]*L[4:])<=302.7, x>=0]
prob = cp.Problem(obj, con)
prob.solve(solver='GLPK_MI',verbose =True)
print("最优值为:",prob.value)
print("最优解为：\n",x.value)
```

2.装箱总重量最大的模型

要使两辆平板车的装箱总重量之和最大，目标函数为 $\max z_2 = \sum_{j=1}^{7} w_j \left(x_{1j} + x_{2j}\right)$，约束条件与前述模型相同。利用 cvxpy 库，可得到问题的最优解：

$$x^* = \left(x_{ij}\right)_{2\times 7} = \begin{bmatrix} 6 & 0 & 0 & 6 & 6 & 0 & 0 \\ 2 & 7 & 9 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad z_2 = 73000.$$