

这种分支约束更易于理解和实现，也更简洁，仅在左右分支节点中各添加了一条约束，并且子问题不用做任何改动。相比第一种基于弧段的分支策略而言，第二种分支策略中左右子树更加平衡。因此，在实际中更建议使用第二种基于弧段的分支策略。

实际上，在第二种基于弧段的分支策略下，子问题也可以做相应的改动，以加快求解。即，在左子节点的子问题中，删去弧 (i_q, i_{q+1}) 。在右子节点的子问题中，删去弧 (i_q, t) , where $t \neq i_{q+1}$ 和 (s, i_{q+1}) , where $s \neq i_q$ 。

15.2.4 界限更新

在分支定界的过程中，采用最基本的界限（Bounds）更新，具体来讲，在每步的迭代中，令

$$UB = z_{RMP}^*$$
 (15.23)

$$LB = z_{RLMP}^* + \sum_{k=1}^K \bar{\sigma}_k^*$$
 (15.24)

其中， z_{RMP}^* 为 RMP 的整数可行解对应的目标值， z_{RLMP}^* 为 RMP 的线性松弛问题 RLMP 的最优解对应的目标值， $\bar{\sigma}_k^*$ 为子问题 k 的最优目标值，即子问题 k 的检验数。

因为篇幅所限，不在此处给出分支定价算法求解 VRPTW 的完整代码。在附配的资源中提供了相关开源代码的链接，请感兴趣的读者自行查阅。

第16章 Dantzig-Wolfe 分解算法

Dantzig-Wolfe 分解算法（Dantzig-Wolfe Decomposition Algorithm）由 George B. Dantzig 和 Philip Wolfe 于 1960 年首次提出，并用于求解大规模线性规划（Dantzig and Wolfe, 1960）。Dantzig-Wolfe 分解算法是一种通过将大规模问题分解成若干较小规模的问题，从而提升求解效率的方法。通常，该算法都是与列生成算法结合使用的。

16.1 引例

在有些线性规划中，约束和变量可以按照如下方式进行分解：

集合 1 中的约束只涉及在变量集合 1 中的变量；

集合 2 中的约束只涉及在变量集合 2 中的变量；

……

集合 k 中的约束只涉及在变量集合 k 中的变量；

集合 $k+1$ 中的约束条件可以涉及任何变量。集合 $k+1$ 中的约束条件称为中心约束群。

下面举个例子进一步理解这一点，该例子参考文献（Winston, 2004; Winston, 2006）。某公司在 2 家工厂（工厂 1 和工厂 2）生产 2 种钢材（钢材 1 和钢材 2）。生产钢材需要铁、煤和鼓风炉。每家工厂都有自己的煤矿，工厂 1 每天可以使用 12 吨煤，工厂 2 每天可以使用 15 吨煤。每家工厂都有自己的鼓风炉，由于类型不同所以生产 1 吨钢材需要的相关资源也不同，如表 16.1 所示。工厂 1 每天可以使用 10 小时的鼓风炉，工厂 2 每天可以使用 4 小时的鼓风炉。铁矿厂位于 2 家工厂的中间，每天一共可以提供 80 吨铁。

表 16.1 公司生产资源需求表

产品/吨	需要的铁/吨	需要的煤/吨	需要的鼓风炉时/吨
工厂 1 生产的钢材 1	8	3	2
工厂 1 生产的钢材 2	6	1	1
工厂 2 生产的钢材 1	7	3	1
工厂 2 生产的钢材 2	5	2	1

设该公司生产的所有钢材都运送给同一个客户，每吨钢材 1 的售价是 170 美元，每吨钢材 2 的售价是 160 美元。从工厂 1 运输 1 吨钢材需要 80 美元，从工厂 2 运输 1 吨钢

450 ◀ 运筹优化常用模型、算法及案例实战——Python+Java实现

材需要 100 美元。假设运输成本是唯一可变的成本，建立使该公司利润最大的线性规划模型并求解。

首先定义如下变量：

x_1 =工厂 1 每天生产的钢材 1 的吨数；

x_2 =工厂 1 每天生产的钢材 2 的吨数；

x_3 =工厂 2 每天生产的钢材 1 的吨数；

x_4 =工厂 2 每天生产的钢材 2 的吨数。

公司的总收入是 $170(x_1 + x_3) + 160(x_2 + x_4)$ ，运输成本是 $80(x_1 + x_2) + 100(x_3 + x_4)$ 。因此，目标函数利润 z 是：

$$\begin{aligned} z &= 170(x_1 + x_3) + 160(x_2 + x_4) - [80(x_1 + x_2) + 100(x_3 + x_4)] \\ &= 90x_1 + 80x_2 + 70x_3 + 60x_4 \end{aligned}$$

公司面临如下 5 个约束条件：

约束 1：工厂 1 每天使用的煤不能超过 12 吨；

约束 2：工厂 1 每天使用的鼓风炉时间不能超过 10 小时；

约束 3：工厂 2 每天使用的煤不能超过 15 吨；

约束 4：工厂 2 每天使用的鼓风炉时间不能超过 4 小时；

约束 5：工厂 1 和工厂 2 每天使用的铁矿石不能超过 80 吨。

上述 5 个约束条件对应如下 5 条约束：

$$\begin{aligned} 3x_1 + x_2 &\leq 12 \\ 2x_1 + x_2 &\leq 10 \\ 3x_3 + 2x_4 &\leq 15 \\ x_3 + x_4 &\leq 4 \\ 8x_1 + 6x_2 + 7x_3 + 5x_4 &\leq 80 \end{aligned}$$

再加上变量取值范围约束 $x_i \geq 0$ ，我们得到该问题的线性规划模型如下：

$$\begin{array}{ll} \max & z = 90x_1 + 80x_2 + 70x_3 + 60x_4 \\ \text{s.t.} & \begin{aligned} 3x_1 + x_2 &\leq 12 \\ 2x_1 + x_2 &\leq 10 \\ 3x_3 + 2x_4 &\leq 15 \\ x_3 + x_4 &\leq 4 \\ 8x_1 + 6x_2 + 7x_3 + 5x_4 &\leq 80 \\ x_1, x_2, x_3, x_4 &\geq 0 \end{aligned} \end{array} \quad (16.1)$$

上述模型约束结构可以用图 16.1 表示。

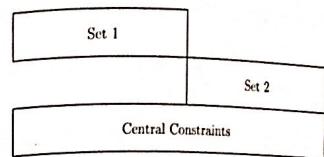


图 16.1 模型约束结构示意图

约束条件集合 1 和变量集合 1 涉及工厂 1 的活动，而不涉及 x_3 和 x_4 ，约束条件集合 2 和变量集合 2 涉及工厂 2 的活动，而不涉及 x_1 和 x_2 ，约束条件集合 3 可以看作一个中心约束条件，它和两个变量集合都有耦合。

对于上述问题模型结构，Dantzig 和 Wolfe 开发了 Dantzig-Wolfe 分解算法。利用 Dantzig-Wolfe 分解算法通常可以求解按照这种方式分解的 LP。

分解算法依赖于如下定理的结论。

假设一个 LP 的可行域是有界的，并且这个 LP 可行域的极点（或基本可行解）是 P_1, P_2, \dots, P_k 。那么这个 LP 可行域中的任意一个点 x 都可以表示为 P_1, P_2, \dots, P_k 的线性组合。也就是说，存在一组权重 $\mu_1, \mu_2, \dots, \mu_k$ 满足：

$$x = \mu_1 P_1 + \mu_2 P_2 + \dots + \mu_k P_k \quad (16.2)$$

同时，这组权重必须满足下面的条件：

$$\mu_1 + \mu_2 + \dots + \mu_k = 1, \quad \forall i = 1, 2, \dots, k \quad (16.3)$$

$$\mu_i \geq 0 \quad (16.4)$$

满足 (16.3) 的权重向量的任意组合称为凸组合。因此该定理表明，如果一个 LP 的可行域是有界的，那么其中的点就可以被表示为该 LP 可行域中极点的凸组合。

为了解释 Dantzig-Wolfe 分解算法的基本思想，假设变量集合被分解成集合 1 和集合 2。Dantzig-Wolfe 分解算法的步骤如下。

1. 第 1 步

设变量集合 1 中的变量是 x_1, x_2, \dots, x_{n_1} 。我们把变量表示成约束条件集合 1（只涉及变量集合 1 中变量的约束条件）的可行域中极点的凸组合。如果设 P_1, P_2, \dots, P_k 是这个可行域的极点，那么可以把约束条件集合 1 中的可行域中的点

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_1} \end{bmatrix} \quad (16.5)$$

记作如下形式：

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_1} \end{bmatrix} = \mu_1 P_1 + \mu_2 P_2 + \cdots + \mu_k P_k \quad (16.6)$$

其中， $\mu_1 + \mu_2 + \cdots + \mu_k = 1$ 且 $\mu_i \geq 0, i = 1, 2, \dots, k$ 。

2. 第2步

把变量集合 2 中的变量 $x_{n_1+1}, x_{n_2+1}, \dots, x_n$ 表示成约束条件集合 2 的可行域中极点的凸组合。如果设这个可行域中的极点为 Q_1, Q_2, \dots, Q_m ，那么可以把约束条件集合 2 的可行域中的点记作

$$\begin{bmatrix} x_{n_1+1} \\ x_{n_2+2} \\ \vdots \\ x_n \end{bmatrix} = \lambda_1 Q_1 + \lambda_2 Q_2 + \cdots + \lambda_m Q_m \quad (16.7)$$

其中， $\lambda_1 + \lambda_2 + \cdots + \lambda_m = 1$ 且 $\lambda_i \geq 0 (i = 1, 2, \dots, m)$ 。

3. 第3步

利用 (16.6) 和 (16.7) 把 LP 的目标函数和约束条件表示为关于 μ_i 和 λ_i 的方程。在增加了约束条件（称作凸性约束条件） $\mu_1 + \mu_2 + \cdots + \mu_k = 1$ 和 $\lambda_1 + \lambda_2 + \cdots + \lambda_m = 1$ 以及符号约束条件 $\mu_i \geq 0 (i = 1, 2, \dots, k)$ 和 $\lambda_i \geq 0, i = 1, 2, \dots, m$ 后，我们可以得到如下通常称作限制主问题的 LP (RMP)：

$$\begin{array}{ll} \max(\text{or } \min) & [\text{关于 } \mu_i \text{ 和 } \lambda_i \text{ 的目标函数}] \\ \text{s.t.} & [\text{关于 } \mu_i \text{ 和 } \lambda_i \text{ 的中心约束条件}] \\ & \mu_1 + \mu_2 + \cdots + \mu_k = 1 \\ & \lambda_1 + \lambda_2 + \cdots + \lambda_m = 1 \\ & \mu_i \geq 0, \quad \forall i = 1, 2, \dots, k \\ & \lambda_i \geq 0, \quad \forall i = 1, 2, \dots, m \end{array}$$

在大规模线性规划问题中，限制主问题 (RMP) 可能有很多个变量（对应于每组约束条件中的很多个基本可行解）。一般情况下，我们只需要产生特定的 μ_i 或 λ_i 对应的列，而不需要写出完整的限制主问题。

4. 第4步

假设限制主问题 (RMP) 基本可行解可以通过启发式算法或者求解器比较容易得到。接下来使用前文介绍的列生成方法就可以求解限制主问题 (RMP)。

5. 第5步

将第4步中求出的 μ_i 或 λ_i 的值代入 (16.6) 和 (16.7)，我们就可以得到 x_1, x_2, \dots, x_n 的最优值。

我们给出 Dantzig-Wolfe 分解算法的伪代码如下。

Algorithm 18 Dantzig-Wolfe 分解算法

初始化：设置 $x_i \leftarrow$ 极点的凸组合

while RMP 的解不是最优 do

用列生成法求解 RMP

end while

return x_i

16.2 块角模型与 Dantzig-Wolfe 分解

上面我们用了一个比较简单的例子引入了 DW 分解算法的具体步骤，本节我们以一个通用的角度来解释 DW 分解（参考文献 [kalvelagen, 2003]）。

16.2.1 块角模型

我们考虑下面的线性规划问题：

$$\begin{array}{ll} \min & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} & \mathbf{A} \mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^n \end{array}$$

其中，约束矩阵 \mathbf{A} 具有如下特定结构：

$$\mathbf{A} \mathbf{x} = \begin{pmatrix} \mathbf{B}_0 & \mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{B}_K \\ & \mathbf{A}_1 & & & \\ & & \mathbf{A}_2 & & \\ & & & \ddots & \\ & & & & \mathbf{A}_K \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_K \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_K \end{pmatrix}$$

即

$$\sum_{k=0}^K \mathbf{B}_k x_k = b_0$$

对应于上述约束矩阵的第一行，称作中心约束。

Dantzig-Wolfe 分解算法的思想就是将上述模型进行分解，而不是考虑所有约束一起求解。

将问题分解为主问题和若干子问题进行求解，主问题考虑中心约束条件，子问题进行

分块求解，从而只有一系列更小规模的问题需要我们求解。

16.2.2 Minkowski 表示定理

考虑线性规划问题的可行域:

$$P = \{x | Ax = b, x \geq 0\}$$

如果 P 是有界的, 那么可以将点 $x \in P$ 表示成 P 的极点 $x^{(j)}$ 的线性组合形式, 如下:

$$\begin{aligned} x &= \sum_j \lambda_j x^{(j)} \\ \sum_j \lambda_j &= 1 \\ \lambda_j &\geq 0 \end{aligned}$$

如果可行域不是有界的情况, 那需要引入极射线, 如下:

$$\begin{aligned} x &= \sum_j \lambda_j x^{(j)} + \sum_i \mu_i r^{(i)} \\ \sum_j \lambda_j &= 1 \\ \lambda_j &\geq 0 \\ \mu_i &\geq 0 \end{aligned}$$

其中, $r^{(i)}$ 是 P 的极射线。上述表述被称作 Minkowski 表示定理。约束 $\sum_j \lambda_j = 1$ 即上文提到的凸约束。

也可以把上述表述写作如下统一的形式:

$$\begin{aligned} x &= \sum_j \lambda_j x^j \\ \sum_j \delta_j \lambda_j &= 1 \\ \lambda_j &\geq 0 \\ \mu_i &\geq 0 \end{aligned} \tag{16.8}$$

其中

$$\delta_j = \begin{cases} 1, & x^j \text{ 是一个极点} \\ 0, & x^j \text{ 是一条极射线} \end{cases}$$

通过上述形式就可以把原问题关于变量 x 的形式转换为关于变量 λ 的形式。实际上, 随着变量 λ_j 的数量增多, 此模型逐渐变得不能直接求解。

16.2.3 模型分解

下面将模型分解成主问题和 K 个子问题, K 个子问题有如下约束形式:

主问题形式如下:

$$\begin{aligned} A_k x_k &= b_k \\ x_k &\geq 0 \end{aligned}$$

我们将 (16.8) 代入上述主问题, 得到

$$\begin{aligned} \min & c_0^T x_0 + \sum_{k=1}^K \sum_{j=1}^{p_k} \left(c_k^T x_k^{(j)} \right) \lambda_{k,j} \\ \text{s.t.} & B_0 x_0 + \sum_{k=1}^K \sum_{j=1}^{p_k} \left(B_k x_k^{(j)} \right) \lambda_{k,j} = b_0 \\ & \sum_{j=1}^{p_k} \delta_{k,j} \lambda_{k,j} = 1, \forall k = 1, 2, \dots, K \\ & x_0 \geq 0 \\ & \lambda_{k,j} \geq 0 \end{aligned}$$

其中, p_k 是第 k 个子问题的极点个数。这仍是一个大规模的线性规划问题。虽然行的数量很多, 但是每个子问题的极点和极射线的数量是很多的, 这就导致变量 $\lambda_{k,j}$ 的数量很多。一开始很多变量是非基变量, 我们就不需要把它们纳入进来。我们的想法是应用列生的思想, 只考虑那些检验数有希望为负的变量。

由此得到限制性主问题 (Restricted Master Problem), 模型如下:

$$\begin{aligned} \min & c_0^T x_0 + c^T \lambda' \\ \text{s.t.} & B_0 x_0 + B \lambda' = b_0 \\ & \Delta \lambda' = 1 \\ & x_0 \geq 0 \\ & \lambda' \geq 0 \end{aligned}$$

缺失的变量定义为 0。限制主问题的列是不固定的, 随着列生成, 会有新的列加入该模型中。变量 $\lambda_{k,j}$ 的潜力可以由其检验数来衡量。我们把中心约束 $B_0 x_0 + B \lambda' = b_0$ 对应的对偶变量记作 π_1 , 把凸约束 $\sum_j \delta_{k,j} \lambda_{k,j}' = 1$ 的对偶变量记作 π_2^k , 则主问题中变量 $\lambda_{k,j}'$ 的检验数可以写作

$$\sigma_{k,j} = (c_k^T x_k^{(j)}) - \pi^T \begin{pmatrix} B_k x_k^{(j)} \\ \delta_{k,j} \end{pmatrix} = (c_k^T - \pi_1^T B_k) x_k^{(j)} - \pi_2^k \delta_{k,j} \quad (16.9)$$

如果子问题是有界的，那么最有潜力的加入主问题的基本可行解 x_k 可以通过求解下面以检验数为目标函数的模型得到：

$$\begin{aligned} \min_{x_k} \quad & \sigma_k = (c_k^T - \pi_1^T B_k) x_k - \pi_2^k \\ \text{s.t.} \quad & A_k x_k = b_k \\ & x_k \geq 0 \end{aligned}$$

寻找这些检验数的过程通常称作 pricing。如果 $\sigma_k^* \leq 0$ ，就可以将新列 $\lambda_{k,j}$ 加入主问题中，该列的成本系数是 $c_k^T x_k^*$ 。

DW 分解中主问题和子问题的交互可以参见图 16.2。

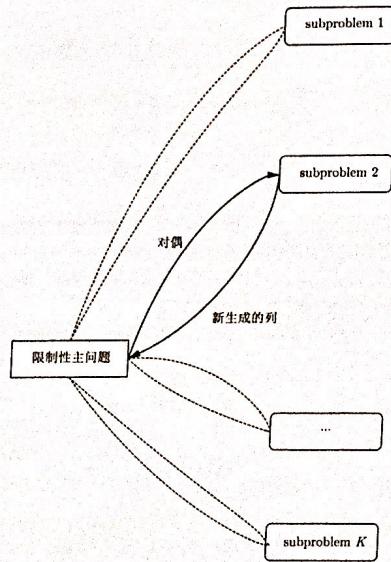


图 16.2 限制性主问题和子问题之间的交互

DW 分解的伪代码如下 (kalvelagen, 2003)。

```

Algorithm 19 Dantzig-Wolfe 分解算法
1 {初始化}: 选择初始变量的子集
2 while true do
3   [Master problem]
4     求解 RMP.
5      $\pi_1 \leftarrow$  中央约束的对偶变量
6      $\pi_2^k \leftarrow$  第  $k$  个凸组合约束的对偶变量
7     {Sub problems}
8     for  $k = 1, \dots, K$  do
9       将  $\pi_1$  和  $\pi_2^k$  代入 subproblem  $k$ 
10      求解 subproblem  $k$ 
11      if  $\sigma_k^* \leq 0$  then
12        将新的候选极点  $x_k^*$  加入 RMP
13      end if
14    end for
15    if 没有新的候选极点产生 then
16      Stop:optimal
17    end if
18 end while

```

关于分解算法的初始化需要根据不同的问题具体实际地分析。在求解子问题过程中，如果子问题不可行，那么原问题即是不可行的。

16.2.4 两阶段法

有时候，一开始的解可能不满足中心约束条件。我们可以通过 Phase I/II Algorithm 来解决（实际上就是单纯形法中的两阶段法）。首先，通过引入人工变量并且最小化它来形成第一阶段问题。关于人工变量的解释本书不再赘述，不熟悉的读者可参考线性规划的相关教材。需要注意的是，第一阶段问题的检验数和第二阶段问题的检验数是有些差别的。

比如说，我们考虑如下中心约束条件：

$$\sum_j x_j \leq b$$

引入人工变量 $x_a \geq 0$ ，得

$$\sum_j x_j - x_a \leq b$$

这时第一阶段的目标函数是

$$\min x_a$$

变量 x_j 的检验数如 (16.9) 所示，但是需要 $c_j^T = 0$ 。

需要提醒的是，在第二阶段开始时需要将人工变量移除。在下面的例子中采取的做法是使该人工变量为 0。

16.3 详细案例

接下来继续求解本章开头提到的例子。

和上文定义的集合 1 和集合 2 一致，可以得到约束条件集合 1 和约束条件集合 2 的可行域，如图 16.3 所示，其中阴影部分是可行域。

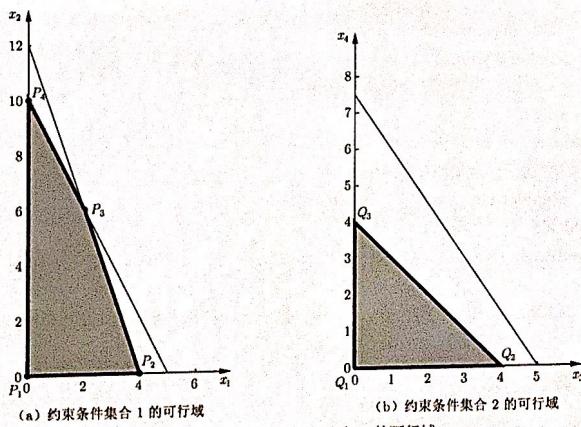


图 16.3 约束条件集合 1 和 2 的可行域

于是，可以把约束条件集合 1 的极点记作

$$P_1 = (0, 0), P_2 = (4, 0), P_3 = (2, 6), P_4 = (0, 10)$$

把约束条件集合 2 的极点记作

$$Q_1 = (0, 0), Q_2 = (4, 0), Q_3 = (0, 4)$$

已知变量集合 $1=\{x_1, x_2\}$ ，则有

$$\text{约束条件集合 } 1 = \begin{cases} 3x_1 + x_2 \leq 12 \\ 2x_1 + x_2 \leq 10 \end{cases}$$

接下来可以得到约束条件集合 1 的可行点，记作

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mu_1 \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \mu_2 \begin{bmatrix} 4 \\ 0 \end{bmatrix} + \mu_3 \begin{bmatrix} 2 \\ 6 \end{bmatrix} + \mu_4 \begin{bmatrix} 0 \\ 10 \end{bmatrix} = \begin{bmatrix} 4\mu_2 + 2\mu_3 \\ 6\mu_3 + 10\mu_4 \end{bmatrix} \quad (16.10)$$

且 $\mu_1 + \mu_2 + \mu_3 + \mu_4 = 1, \mu_i \geq 0, i = 1, 2, 3, 4$ 。

变量集合 $2=\{x_3, x_4\}$ ，则有

$$\text{约束条件集合 } 2 = \begin{cases} 3x_3 + 2x_4 \leq 15 \\ x_3 + x_4 \leq 4 \end{cases}$$

得到约束条件集合 2 的可行点，记作

$$\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \lambda_1 \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 4 \\ 0 \end{bmatrix} + \lambda_3 \begin{bmatrix} 0 \\ 4 \end{bmatrix} = \begin{bmatrix} 4\lambda_2 \\ 4\lambda_3 \end{bmatrix} \quad (16.11)$$

且 $\lambda_1 + \lambda_2 + \lambda_3 = 1, \lambda_i \geq 0, i = 1, 2, 3$ 。

将 (16.10) 和 (16.11) 代入目标函数和中心约束条件中，得到限制主问题。目标函数整

数

$$360\mu_2 + 660\mu_3 + 800\mu_4 + 280\lambda_2 + 240\lambda_3$$

和约束条件整理为

$$32\mu_2 + 52\mu_3 + 60\mu_4 + 28\lambda_2 + 20\lambda_3 \leq 80$$

将约束条件加入松弛变量 s_1 ，得到限制主问题模型如下：

$$\begin{aligned} \max z = & 360\mu_2 + 660\mu_3 + 800\mu_4 + 280\lambda_2 + 240\lambda_3 & (16.12) \\ \text{s.t.} & 32\mu_2 + 52\mu_3 + 60\mu_4 + 28\lambda_2 + 20\lambda_3 + s_1 = 80 \\ & \mu_1 + \mu_2 + \mu_3 + \mu_4 = 1 \\ & \lambda_1 + \lambda_2 + \lambda_3 = 1 \\ & \mu_1, \mu_2, \mu_3, \mu_4, \lambda_1, \lambda_2, \lambda_3, s_1 \geq 0 \end{aligned}$$

这一步转化模型之后，我们需要解释一下。模型 (16.12) 和模型 (16.11) 已经是完全等价了。直接求解 (16.12) 得到的解，和 (16.11) 的解是等价的。目标函数相同，只是变量不同。但是原始变量可以通过 (16.12) 中的变量转化得到。

接下来使用改进单纯形法和列生成法求解限制主问题。把初始单纯形表称作单纯形表 0. $BV(0)=\{s_1, \mu_1, \lambda_1\}$ 。

$$B_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, B_0^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

由于 s_1, μ_1, λ_1 没有出现在限制主问题目标函数中，所以 $c_{BV} = [0 \ 0 \ 0]$ ，单纯形表 0 的影子价格是

$$c_{BV}^T B_0^{-1} = [0 \ 0 \ 0] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [0 \ 0 \ 0]$$

我们分两个阶段应用列生成的方法。首先确定是否存在与约束条件集合 1 相关且价格有利的 μ_i （由于求解的是最大化问题，第 0 行中系数为负的列是不利的）。与约束条件集合 1 的极点 $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ 相关的系数 μ_i 在限制主问题中具有如下列：

$$\mu_i \text{ 的目标函数系数 } = 90x_1 + 80x_2$$

$$\mu_i \text{ 在约束条件中的列 } = \begin{bmatrix} 8x_1 + 6x_2 \\ 1 \\ 0 \end{bmatrix}$$

由以上信息可知，在单纯形表 0 中，对应于 $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ 的 μ_i 的列具有如下价格：

$$c_{BV}^T B_0^{-1} \begin{bmatrix} 8x_1 + 6x_2 \\ 1 \\ 0 \end{bmatrix} - (90x_1 + 80x_2) = -90x_1 - 80x_2$$

由于 $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ 需要满足约束条件集合 1（或工厂 1 的约束条件），所以具有最大负价格的加权 μ_i 是与作为下面 LP 最优解的极点相关的加权。

工厂 1 子问题如下：

$$\begin{aligned} \min z &= -90x_1 - 80x_2 \\ \text{s.t. } &3x_1 + x_2 \leq 12 \\ &2x_1 + x_2 \leq 10 \\ &x_1, x_2 \geq 0 \end{aligned}$$

求解工厂 1 子问题，得解 $z = -800, x_1 = 0, x_2 = 10$ 。这说明，与极点 $\begin{bmatrix} 0 \\ 10 \end{bmatrix}$ 相关的加权 μ_i 将有最大负价格。由于 $P_4 = \begin{bmatrix} 0 \\ 10 \end{bmatrix}$ ，这意味着 μ_i 在限制主问题中具有系数 -800 。

现在分析和约束条件集合 2 相关的加权，并设法确定具有最大负价格的加权 λ_i 。对于约束条件集合 2 的极点 $\begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$ 的 λ_i 在限制主问题中具有以下列：

$$\lambda_i \text{ 的目标函数系数 } = 70x_3 + 60x_4$$

$$\lambda_i \text{ 在约束条件中的列 } = \begin{bmatrix} 7x_3 + 5x_4 \\ 0 \\ 1 \end{bmatrix}$$

表明对应于极点 $\begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$ 的 λ_i 的价格是

$$c_{BV}^T B_0^{-1} \begin{bmatrix} 7x_3 + 5x_4 \\ 0 \\ 1 \end{bmatrix} - (70x_3 + 60x_4) = -70x_3 - 60x_4$$

由于 $\begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$ 需要满足约束条件集合 2（或工厂 2 的约束条件），所以具有最大负价格的加权与下列 LP 最优解的极点相关的加权。

工厂 2 的子问题如下：

$$\begin{aligned} \min z &= -70x_3 - 60x_4 \\ \text{s.t. } &3x_3 + 2x_4 \leq 15 \\ &x_3 + x_4 \leq 4 \\ &x_3, x_4 \geq 0 \end{aligned}$$

求解工厂 2 的子问题，得解 $z = -280, x_3 = 4, x_4 = 0$ 。这意味着，与极点 $P_2 = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$ 相关的 λ_2 在所有 λ_i 中具有最大负价格。由于 μ_i 的价格比 λ_2 还要负，所以把 μ_i 换入基。

μ_i 在单纯形表 0 中的列是

$$B_0^{-1} \begin{bmatrix} 8x_1 + 6x_2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 60 \\ 1 \\ 0 \end{bmatrix}$$

单纯形表 0 的右端项是

$$B_0^{-1} b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 80 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 80 \\ 1 \\ 1 \end{bmatrix}$$

通过测试表明， μ_i 应在第二个约束条件中换入基。因此 $BV(1) = \{s_1, \mu_i, \lambda_1\}$ ，由于

$$E_0 = \begin{bmatrix} 1 & -60 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, B_1^{-1} = E_0 B_0^{-1} = \begin{bmatrix} 1 & -60 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

所以 μ_4 的目标函数系数是 800。根据

$$c_{BV}^T B_1^{-1} = \begin{bmatrix} 0 & 800 & 0 \end{bmatrix} \begin{bmatrix} 1 & -60 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 800 & 0 \end{bmatrix}$$

可以求出影子价格的新集合。接下来求当前单纯形表中具有最大负价格的加权。和前面一样，先求当前单纯形表的工厂 1 和工厂 2 的子问题，不断求出新的影子价格，直至得不到一个 μ_i 和 λ_i 具有利的价格（子问题目标函数值大于或等于 0）。

最终得到限制主问题的最优解是 $\lambda_3 = 1, \mu_4 = 1, \lambda_1 = 0$ ，其他加权值都为 0。

下面把约束条件集合 1 可行域的表达式作为其极点的凸组合，来确定 $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ 的最优值

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0P_1 + 0P_2 + 0P_3 + P_4 = \begin{bmatrix} 0 \\ 10 \end{bmatrix}$$

同样地，把约束条件集合 2 可行域的表达式作为其极点的凸组合，来确定 $\begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$ 的最优值

$$\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = 0Q_1 + 0Q_2 + Q_3 = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$$

所以该问题的最优解是 $x_2 = 10, x_4 = 4, x_1 = x_3 = 0$ ，即在工厂 1 生产 10 吨钢材 2，在工厂 2 生产 4 吨钢材 2，该公司能达到最大利润值 1040。

这里用 Python 调用 Gurobi 来验证上面的解，代码如下：

PythonGurobiDWExample

```

1 DW_model = Model('DW Decomposition')
2 mu = {}
3 lam = {}
4 s = {}
5 s[1] = DW_model.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name='s_1')
6 for i in range(4):
7     mu[i+1] = DW_model.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS,
8         name='mu_' + str(i+1))
9     lam[i+1] = DW_model.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS,
10        name='lam_' + str(i+1))
11 obj = LinExpr(0)

```

```

obj = 360 * mu[2] + 660 * mu[3] + 800 * mu[4] + 280 * lam[2] + 240 * lam[3]
DW_model.setObjective(obj, GRB.MAXIMIZE)
DW_model.addConstr(32*mu[2] + 52*mu[3] + 60*mu[4] + 27*lam[2] + 20*lam[3] +
    s[1] == 80, name = 'c1')
DW_model.addConstr(mu[1] + mu[2] + mu[3] + mu[4] == 1, name = 'c2')
DW_model.addConstr(lam[1] + lam[2] + lam[3] == 1, name = 'c3')
DW_model.optimize()

for var in DW_model.getVars():
    if(var.x > 0):
        print(var.varName, '\t', var.x)

```

得到的解如下。

PythonGurobiDWExampleOutput

```

Solved in 3 iterations and 0.02 seconds
Optimal objective 1.040000000e+03
mu_4 1.0
mu_3 1.0

```

可以看到，目标函数是相同的，也是 1040。并且最优解是 $\mu_4 = 1, \lambda_3 = 1$ 。跟上面的解是相同的。

16.4 Dantzig-Wolfe 分解求解大规模混合整数规划

DW 分解算法的主要优点是将原问题分解成几个规模较小的子问题。而求解几个较小的 LP 问题通常比求解一个大型 LP 问题要容易得多。模型 (16.1) 是一个非常小的算例。如果说模型规模比较大的时候，比如对于 100 个客户点的 TSP，其变量个数要成千上万个，因此问题的可行域的极点数量就非常多。要想一开始就全部穷举出所有的极点就不太现实。我们观察到，模型 (16.12) 非常有规律，每一列都对应一个极点，比如 μ_1 就对应极点 $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ 而 μ_2 就对应极点 $\begin{bmatrix} 4 \\ 0 \end{bmatrix}$ 。每一个极点，例如 $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ 和 $\begin{bmatrix} 4 \\ 0 \end{bmatrix}$ 其实就是子问题的一个基可行解。所以只要找到每个子问题的可行域的所有极点（也就是该子问题的所有基可行解），DW 分解之后的完整的等价形式就可以写出来。规模较大时，不用完整地列举出所有的极点，也能找到问题的等价形式。该方法就是使用列生成算法。我们之前介绍过，列生成算法其实就是要找到将要进基的变量（一个进基变量就对应一个极点）。因此我们用列生成的办法，将那些检验数为正（max 问题）的极点（也就是列）添加进限制性主问题 RMP，这就可以实现求解大规模线性规划或者整数规划的目的。

我们就以模型 (16.1) 为例，一步一步详细地展示这个过程。

16.4.1 两阶段法实现 Dantzig-Wolfe 分解算法介绍

首先将问题初始化，一般情况下我们需要用启发式算法生成一些初始的列。但是如果我们将初始的列怎么办呢？文献（kalvelagen, 2003）提供了一种用两阶段法求解的思路。两阶段法是线性规划求解中为人熟知的方法。

由于有凸组合约束和中心约束，因此我们刚开始添加的列可能会违反中心约束。因此我们可以在第1阶段引入人工变量，目标函数是最小化人工变量的和。引入人工变量的目的是判断该问题是否有可行解，如果第1阶段最小化人工变量的和的目标函数到达了0，就说明该问题有解。我们就可以删去人工变量，进行第2阶段的算法，最终得到原问题的最优解。

本问题中，中心约束为

$$\sum_{i \in I} \alpha_i \mu_i + \sum_{j \in J} \beta_j \lambda_j \leq b \quad (16.13)$$

的形式，我们可以引入一个单独的人工变量 $x_a \geq 0$ ，并且将约束改写为

$$\sum_{i \in I} \alpha_i \mu_i + \sum_{j \in J} \beta_j \lambda_j - x_a \leq b \quad (16.14)$$

设置第1阶段的目标函数为

$$\min x_a \quad (16.15)$$

此时，子问题产生的变量在目标函数中的系数全部为0。当第1阶段的目标函数（16.15）为0时，我们开启第2阶段的算法。将人工变量的值固定为0，继续迭代算法即可。

16.4.2 第1阶段

初始化问题的时候模型是空列。我们添加人工变量 s ，将主问题初始化为

$$\max z = s \quad (16.16)$$

$$\text{s.t. } -s \leq 80 \rightarrow \pi_1 \quad (16.17)$$

$$\text{Null} = \text{Null} \rightarrow \pi_2 \quad (16.18)$$

$$\text{Null} = \text{Null} \rightarrow \pi_3 \quad (16.19)$$

$$s \geq 0 \quad (16.20)$$

$$(16.21)$$

1. Iteration 0

模型（16.16）中，人工变量 s 就是为了算法能够进行下去才引入的，2个凸组合的约束也需要初始化。我们先将其初始化为 Null。相应的初始化代码如下。

```

DWInstancePhase1Step0

# Model('DW Master Problem')
DP.setParam("OutputFlag", 0)
----- start initialize RMP -----
# initialize column : Null column
RMPnum = 3
; because vars in RMP is added into RMP dynamically, thus we creat an array
; to store the variable set
RMPvars = []
RMPvars.append(RMP.addVar(lb=0.0
                           , ub=1000
                           , obj=1
                           , vtype=GRB.CONTINUOUS # GRB.BINARY, GRB.INTEGER
                           , name='s_1'
                           , column=None
                           ))
RMPcons = []
RMPnames = []
col = [-1, 0.001, 0.001]
RMPnames.append('s_0')

; column to add constraints into RMP
RMPcons.append(RMP.addConstr(lhs = RMPvars[0] * col[0]
                               , sense= "<="
                               , rhs= 80
                               , name='rmp_con_centeral'
                               ))
RMPcons.append(RMP.addConstr(lhs = RMPvars[0] * col[1]
                               , sense= "="
                               , rhs= 1
                               , name='rmp_con_convex_combine_mu'
                               ))

```