Short Course on Column Generation - Part II

Ana Flávia U. S. Macambira
ana.macambira@academico.ufpb.br

Alain Faye
alain.faye@ensiie.fr

Departamento de Estatística
Universidade Federal da Paraíba
École nationale supérieure d'informatique pour l'industrie et l'entreprise

Overview

1. Column Generation for integer programming
2. Branch and Price
3. Branch and Price for the Generalized Assignment Problem (GAP)
   - Numerical example GAP
4. Branch and Price for Cutting stock
   - Numerical example cutting stock
5. Integer Programming - revision - Branch and Cut
6. Basic Julia JuMP

Just remembering

- Last class we talked about the relationship between tighter formulations, smaller gaps and smaller branch and bound tree;
- Column generation works better for integer programming;
- One example is the generalized assignment problem (GAP) for which the relaxed restricted master problem (RMP) is tighter than the relaxation of the original problem, [1], p 317.

Branch and Price

- Branch and Price is a combination of Branch and Bound with column generation, [2];
- But there are some difficulties when applying the column generation, which is originally developed for linear programming problems, in integer programming problems;
- One of these problems is that the usual branching may destroy the structure of the pricing problem [1];
- If you want to read more about that subject, the suggestion is reading [3].

Branch and Price

- At Branch and Price we relax the RMP and solve the integer Auxiliary Problem;
- As the Auxiliary Problem is as difficult to solve as the original problem, there are strategies used to solve many auxiliary problems of the literature, as the knapsack problem that will appear at our examples;
- The consequence is that there are specific Branch and Price algorithms adapted to many problems of the literature;
- Once we generate integer columns, we add these columns to the master problem until there is no column to add anymore;
- If we are in the case of no more columns to add and the solution of the master problem is not integer, we run a Branch and Bound at the master problem.

Branch and Price - Generalized Assignment Problem (GAP)

- We have m tasks to be assigned to n machines, $|m| \geq |n|$;

Branch and Price - Generalized Assignment Problem (GAP)

- We have m tasks to be assigned to n machines, $|m| \geq |n|$;
- Each task is assigned to exactly one machine;

Branch and Price - Generalized Assignment Problem (GAP)

- We have m tasks to be assigned to n machines, $|m| \geq |n|$;
- Each task is assigned to exactly one machine;
- Each machine has its capacity constraint;

Branch and Price - Generalized Assignment Problem (GAP)

- We have m tasks to be assigned to n machines, $|m| \geq |n|$;
- Each task is assigned to exactly one machine;
- Each machine has its capacity constraint;
- Objective: maximize the profit assignment of the m tasks to the n machines.

Branch and Price - GAP

- We are going to use p_{ij} for the profit of assigning task i to machine j;
- w_{ij} is the amount of resource consumption of task i at machine j;
- d_j is the total resource of machine j;
- x_{ij} is the binary variable which indicates whether task i is assigned to machine j.

Branch and Price - GAP

Traditional formulation. Just remembering, index i is related to the tasks and index j is related to the machines.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{1 \le i \le m} \sum_{1 \le j \le n} p_{ij} x_{ij} \\
\text{subject to:} \quad & \sum_{i \le j \le n} x_{ij} = 1, \qquad i = 1, ..., m \\
& \sum_{1 \le i \le m} w_{ij} x_{ij} \le d_j, \qquad j = 1, ..., n \\
& x_{ij} \in \{0, 1\}, \;\; i = 1, ..., m, \;\; j = 1, .., n
\end{aligned}
$$

$x_{ij} = 1$ if task i is assigned to machine j. The first constraint states that each task is assigned to one machine and the second constraint means that the sum of the resource consumption of the tasks has to be less or equal the total resource of each machine.

Branch and Price - GAP

Knapsack Problem

- Given a set of items, each item with a related weight (w) and value (p);
- Given a knapsack with a total weight capacity (d);
- The objective is to determine which items to put at the knapsack respecting the total weight capacity of the knapsack;

$$\begin{aligned} \text{maximize} \quad & \sum_{1 \le i \le n} p_i x_i \\ \text{subject to:} \quad & \sum_{1 \le i \le n} w_i x_i \le d, \\ & x_i \in \{0,1\}, \ i = 1, \ldots, n \end{aligned}$$

Branch and Price - GAP

- In order to rewrite GAP problem applying the Dantzig-Wolfe reformulation;
- Consider that the m entries (related to the tasks) of a column

$$y_j^k = (y_{1j}^k, y_{2j}^k, ..., y_{mj}^k) \tag{1}$$

  satisfy the knapsack constraint and also binary constraints of GAP problem.
- We are going to rewrite the GAP problem using these columns.

Branch and Price - GAP

Master Problem

$$
\begin{aligned}
\text{maximize} \quad & \sum_{1\leq j\leq n}\sum_{1\leq k\leq K_j}\Big(\sum_{1\leq i\leq m} p_{ij}y_{ij}^k\Big)\lambda_j^k \\
\text{subject to:} \quad & \sum_{1\leq j\leq n}\sum_{1\leq k\leq K_j} y_{ij}^k\lambda_j^k = 1, \quad i = 1, ..., m \\
& \sum_{1\leq k\leq K_j} \lambda_j^k = 1, \; j = 1, ..., n \\
& \lambda_j^k \in \{0, 1\}, \; j = 1, ..., n, k = 1, ..., K_j
\end{aligned}
$$

Branch and Price - GAP

Auxiliary Problem

$$\begin{aligned}\text{maximize} \quad & \sum_i (p_{ij} - \pi_i) x_{ij} - \nu_j \\ \text{subject to:} \quad & \sum_i w_{ij} x_{ij} \leq d \\ & x_{ij} \in \{0,1\}, \ i = 1, ..., m\end{aligned}$$

Numerical example GAP

This example is at [4].

$m = 3,\ n = 2,\ (d_1, d_2) = (11, 18)$

$$p_{ij} = \begin{pmatrix} 10 & 6 \\ 7 & 8 \\ 5 & 11 \end{pmatrix},\ w_{ij} = \begin{pmatrix} 9 & 5 \\ 6 & 7 \\ 3 & 9 \end{pmatrix}$$

Traditional Formulation

$$\begin{array}{rl} \text{maximize} & 10x_{11} + 7x_{21} + 5x_{31} + 6x_{12} + 8x_{22} + 11x_{32} \\ \text{subject to:} & 9x_{11} + 6x_{21} + 3x_{31} \le 11 \\ & 5x_{12} + 7x_{22} + 9x_{32} \le 18 \\ & x_{11} + x_{12} = 1 \\ & x_{21} + x_{22} = 1 \\ & x_{31} + x_{32} = 1 \\ & x_{ij} \in \{0,1\},\ i = 1,\dots,3,\ j = 1,2. \end{array}$$

Branch and Price - GAP

Traditional Formulation

$$\begin{aligned}
\text{maximize} \quad & 10x_{11} + 7x_{21} + 5x_{31} + 6x_{12} + 8x_{22} + 11x_{32} \\
\text{subject to:} \quad & 9x_{11} + 6x_{21} + 3x_{31} \leq 11 \\
& 5x_{12} + 7x_{22} + 9x_{32} \leq 18 \\
& x_{11} + x_{12} = 1 \\
& x_{21} + x_{22} = 1 \\
& x_{31} + x_{32} = 1 \\
& x_{ij} \in \{0,1\},\ i = 1, ..., 3,\ j = 1, 2.
\end{aligned}$$

As we have seen in Class 1, we are going to find the vertices of $\{x_{ij} \in \mathbb{R}^2_+ \mid A_2 x \leq b_2,\ x_{ij} \in \{0,1\}\}$ and then rewrite the problem using this representation.

Numerical example GAP

- From constraint $9x_{11} + 6x_{21} + 3x_{31} \leq 11$ related to machine one, we can see that the sum of the resource consumption of the three tasks $(9 + 6 + 3)$ exceeds the 11 units of total resource of machine 1. Therefore, machine 1 cannot process the 3 tasks;
- From constraint $5x_{12} + 7x_{22} + 9x_{32} \leq 18$ we also see that machine 2 cannot process all the 3 tasks for the same reason.

Numerical example GAP

$$9x_{11} + 6x_{21} + 3x_{31} \leq 11$$

This constraint is related to machine 1, and we have four feasible solutions:

- $(1,0,0) = y_{i1}^1$ means task one at machine one, consuming 9 units of 11 units of resource available. y_{i1}^1 is associated to λ_1^1
- $(0,1,0) = y_{i1}^2$ means task two at machine one, consuming 6 units of 11 units of resource available. y_{i1}^2 is associated to λ_1^2
- $(0,0,1) = y_{i1}^3$ means task three at machine one, consuming 3 units of 11 units of resource available. y_{i1}^3 is associated to λ_1^3
- $(0,1,1) = y_{i1}^4$ means task 2 and 3 at machine one, consuming 9 (9=6+3) units of 11 units of resource available. y_{i1}^4 is associated to λ_1^4

Numerical example GAP

$$5x_{12} + 7x_{22} + 9x_{32} \leq 18$$

This constraint is related to machine 2, and we have six feasible solutions:

- $(1,0,0) = y_{i2}^1$ means task one at machine two, consuming 5 units of 18 units of resource available. y_{i2}^1 is associated to λ_2^1
- $(0,1,0) = y_{i2}^2$ means task two at machine two, consuming 7 units of 18 units of resource available. y_{i2}^2 is associated to λ_2^2
- $(0,0,1) = y_{i2}^3$ means task three at machine two, consuming 9 units of 18 units of resource available. y_{i2}^3 is associated to λ_2^3
- $(1,1,0) = y_{i2}^4$ means task 1 and 2 at machine two, consuming 12 (12= 5+7) units of 18 units of resource available. y_{i2}^4 is associated to λ_2^4
- $(0,1,1) = y_{i2}^5$ means task 2 and 3 at machine two, consuming 16 (16 = 7+9) units of 18 units of resource available. y_{i2}^5 is associated to λ_2^5
- $(1,0,1) = y_{i2}^6$ means task 1 and 3 at machine two, consuming 14 (14 = 5 + 9) units of 18 units of resource available. y_{i2}^6 is associated to λ_2^6

Numerical example GAP

So, for the first constraints related to the tasks,

$$\sum_{1\le j\le n}\sum_{1\le k\le K_j} y_{ij}^k\lambda_j^k = 1, \qquad i = 1,\dots,m$$

for $i = 1,\dots,3,\ j = 1,2$ and we know that $K_1 = 1,\dots,4$ and $K_2 = 1,..,6$, writing literally each constraint we have:

$$y_{11}^1\lambda_1^1 + y_{11}^2\lambda_1^2 + y_{11}^3\lambda_1^3 + y_{11}^4\lambda_1^4 + y_{12}^1\lambda_2^1 + y_{12}^2\lambda_2^2 + y_{12}^3\lambda_2^3 + y_{12}^4\lambda_2^4 + y_{12}^5\lambda_2^5 + y_{12}^6\lambda_2^6 = 1 \tag{2}$$

$$y_{21}^1\lambda_1^1 + y_{21}^2\lambda_1^2 + y_{21}^3\lambda_1^3 + y_{21}^4\lambda_1^4 + y_{22}^1\lambda_2^1 + y_{22}^2\lambda_2^2 + y_{22}^3\lambda_2^3 + y_{22}^4\lambda_2^4 + y_{22}^5\lambda_2^5 + y_{22}^6\lambda_2^6 = 1 \tag{3}$$

$$y_{31}^1\lambda_1^1 + y_{31}^2\lambda_1^2 + y_{31}^3\lambda_1^3 + y_{31}^4\lambda_1^4 + y_{32}^1\lambda_2^1 + y_{32}^2\lambda_2^2 + y_{32}^3\lambda_2^3 + y_{32}^4\lambda_2^4 + y_{32}^5\lambda_2^5 + y_{32}^6\lambda_2^6 = 1 \tag{4}$$

Numerical example - GAP

Rewriting constraint $x_{11} + x_{12} = 1$ related to task 1, we can see that task 1 appears in

- $(1, 0, 0) = y_{i1}^{1}$ which is associated to λ_1^1;
- $(1, 0, 0) = y_{i2}^{1}$ which is associated to λ_2^1;
- $(1, 1, 0) = y_{i2}^{4}$ which is associated to λ_2^4;
- $(1, 0, 1) = y_{i2}^{6}$ which is associated to λ_2^6;

Therefore, this constraint is rewritten as:

$$\lambda_1^1 + \lambda_2^1 + \lambda_2^4 + \lambda_2^6 = 1.$$

Numerical example - GAP

Rewriting constraint $x_{21} + x_{22} = 1$ related to task 2, we can see that task 2 appears in

- $(0, 1, 0) = y_{i1}^2$ which is associated to λ_1^2;
- $(0, 1, 1) = y_{i1}^4$ which is associated to λ_1^4;
- $(0, 1, 0) = y_{i2}^2$ which is associated to λ_2^2;
- $(1, 1, 0) = y_{i2}^4$ which is associated to λ_2^4;
- $(0, 1, 1) = y_{i2}^5$ which is associated to λ_2^5;

Therefore, this constraint is rewritten as:

$$\lambda_1^2 + \lambda_1^4 + \lambda_2^2 + \lambda_2^4 + \lambda_2^5 = 1.$$

Numerical example - GAP

Rewriting constraint $x_{31} + x_{32} = 1$ related to task 3, we can see that task 3 appears in

- $(0,1,0) = y_{i1}^3$ which is associated to λ_1^3;
- $(0,1,1) = y_{i1}^4$ which is associated to λ_1^4;
- $(0,0,1) = y_{i2}^3$ which is associated to λ_2^3;
- $(0,1,1) = y_{i2}^5$ which is associated to λ_2^5;
- $(1,0,1) = y_{i2}^6$ which is associated to λ_2^6;

Therefore, this constraint is rewritten as:

$$\lambda_1^3 + \lambda_1^4 + \lambda_2^3 + \lambda_2^5 + \lambda_2^6 = 1.$$

Numerical example GAP

Now we are going to write the constraints related to the convexity property

$$\sum_{1\leq k\leq K_j} \lambda_j^k = 1,\ j = 1, ..., n$$

We have $k_1 = 1, ..., 4,\ \ k_2 = 1, ..., 6$ and $j = 1, 2$, so we have:

$$\lambda_1^1 + \lambda_1^2 + \lambda_1^3 + \lambda_1^4 = 1$$

$$\lambda_2^1 + \lambda_2^2 + \lambda_2^3 + \lambda_2^4 + \lambda_2^5 + \lambda_2^6 = 1$$

Numerical example GAP

Now we have to calculate the coefficients of the objective function. For machine 1 $(1,0,0),(0,1,0),(0,0,1),(0,1,1)$ the first feasible solution has a cost of 10, which is the cost of task 1 at machine 1, solution 2 has a cost of 7, solution 3 has a cost 5 and solution 4 has a cost of 12 related to 7+5.
For machine 2 we have $(1,0,0),(0,1,0),(0,0,1),(1,1,0),(0,1,1),(1,0,1)$ and solution 1 has a cost of 6, solution 2 has a cost 8, solution 3 has a cost 11, solution 4 has a cost 14, solution 5 has a cost 19 and solution 6 has a cost 17.
So, the expression for the objective function is:

$$10\lambda_1^1 + 7\lambda_1^2 + 5\lambda_1^3 + 12\lambda_1^4 + 6\lambda_2^1 + 8\lambda_2^2 + 11\lambda_2^3 + 14\lambda_2^4 + 19\lambda_2^5 + 17\lambda_2^6.$$

Numerical example GAP

So, the full master problem is given by:

$$\begin{aligned}
\text{maximize} \quad & 10\lambda_1^1 + 7\lambda_1^2 + 5\lambda_1^3 + 12\lambda_1^4 + 6\lambda_2^1 + 8\lambda_2^2 + 11\lambda_2^3 + 14\lambda_2^4 + 19\lambda_2^5 + 17\lambda_2^6 \\
\text{subject to:} \quad & \lambda_1^1 + \lambda_2^1 + \lambda_2^4 + \lambda_2^6 = 1 \\
& \lambda_1^2 + \lambda_1^4 + \lambda_2^2 + \lambda_2^4 + \lambda_2^5 = 1 \\
& \lambda_1^3 + \lambda_1^4 + \lambda_2^3 + \lambda_2^5 + \lambda_2^6 = 1 \\
& \lambda_1^1 + \lambda_1^2 + \lambda_1^3 + \lambda_1^4 = 1 \\
& \lambda_2^1 + \lambda_2^2 + \lambda_2^3 + \lambda_2^4 + \lambda_2^5 + \lambda_2^6 = 1 \\
& \lambda_j^k \geq 0
\end{aligned}$$

But as we want to start the column generation, we just need a restricted master problem that has a basic feasible solution. It can be found using two phase method.

Numerical example GAP

Using two phase method for finding an initial basis, we have the fist basis given by $\lambda_B = (\lambda_1^2, \lambda_1^3, \lambda_1^4, \lambda_2^1, \lambda_2^3)$. So, the first iteration Restricted Master Problem (because it has just a subset of all columns) is:

$$\begin{aligned} \text{maximize} \quad & 7\lambda_1^2 + 5\lambda_1^3 + 12\lambda_1^4 + 6\lambda_2^1 + 11\lambda_2^3 \\ \text{subject to:} \quad & \lambda_2^1 = 1 \\ & \lambda_1^2 + \lambda_1^4 = 1 \\ & \lambda_1^3 + \lambda_1^4 + \lambda_2^3 = 1 \\ & \lambda_1^2 + \lambda_1^3 + \lambda_1^4 = 1 \\ & \lambda_2^1 + \lambda_2^3 = 1 \\ & \lambda_j^k \geq 0 \end{aligned}$$

and the optimal solution is $\lambda_1^4 = \lambda_2^1 = 1,\ \lambda_1^2 = \lambda_1^3 = \lambda_2^3 = 0$ with $OF = 18$ and the value of the dual variables are: $\pi_1 = 0,\ \pi_2 = 7,\ \pi_3 = 5,\ \nu_1 = 0,\ \nu_2 = 6.$

Numerical example GAP

The Auxiliary problem for machine 1 of the first iteration is:

$$\begin{array}{rl} \text{maximize} & (10-0)x_{11}+(7-7)x_{21}+(5-5)x_{31}-0 \\ \text{subject to:} & 9x_{11}+6x_{21}+3x_{31}\leq 11 \\ & x_{11},x_{21},x_{31}\in\{0,1\}. \end{array}$$

The auxiliary problem for machine 2 of the first phase is

$$\begin{array}{rl} \text{maximize} & (6-0)x_{12}+(8-7)x_{22}+(11-5)x_{32}-6 \\ \text{subject to:} & 5x_{12}+7x_{22}+9x_{32}\leq 18 \\ & x_{12},x_{22},x_{32}\in\{0,1\}. \end{array}$$

Numerical example GAP

The Auxiliary problem for machine 1 of the first iteration is:

$$\begin{aligned} \text{maximize} \quad & 10x_{11} \\ \text{subject to:} \quad & 9x_{11} + 6x_{21} + 3x_{31} \leq 11 \\ & x_{11}, x_{21}, x_{31} \in \{0, 1\}. \end{aligned}$$

$x_{11} = 1, x_{21} = x_{31} = 0,\ OF = 10$. Solution (1,0,0) means λ_1^1.
The auxiliary problem for machine 2 of the first iteration is

$$\begin{aligned} \text{maximize} \quad & 6x_{12} + x_{22} + 6x_{32} - 6 \\ \text{subject to:} \quad & 5x_{12} + 7x_{22} + 9x_{32} \leq 18 \\ & x_{12}, x_{22}, x_{32} \in \{0, 1\}. \end{aligned}$$

$x_{12} = 1, x_{22} = 0,\ x_{32} = 1,\ OF = 6$. Solution (1,0,1) means λ_2^6.

Numerical example GAP

We are going to insert columns λ_1^1 and λ_2^6 at the master problem. So, the second iteration Restricted Master Problem is:

$$\begin{aligned}
\text{maximize} \quad & 10\lambda_1^1 + 7\lambda_1^2 + 5\lambda_1^3 + 12\lambda_1^4 + 6\lambda_2^1 + 11\lambda_2^3 + 17\lambda_2^6 \\
\text{subject to:} \quad & \lambda_1^1 + \lambda_2^1 + \lambda_2^6 = 1 \\
& \lambda_1^2 + \lambda_1^4 = 1 \\
& \lambda_1^3 + \lambda_1^4 + \lambda_2^3 + \lambda_2^6 = 1 \\
& \lambda_1^1 + \lambda_1^2 + \lambda_1^3 + \lambda_1^4 = 1 \\
& \lambda_2^1 + \lambda_2^3 + \lambda_2^6 = 1 \\
& \lambda_j^k \geq 0
\end{aligned}$$

The optimal solution is: $\lambda_1^1 = 0, \lambda_1^2 = 1, \lambda_1^3 = \lambda_1^4 = \lambda_2^1 = \lambda_2^3 = 0, \lambda_2^6 = 1$.
$\pi_1 = 6, \pi_2 = 3, \pi_3 = 11, \nu_1 = 4, \nu_2 = 0$.

Numerical example GAP

The Auxiliary problem for machine 1 of the second iteration is:

$$\begin{aligned} \text{maximize} \quad & (10-6)x_{11} + (7-3)x_{21} + (5-11)x_{31} - 4 \\ \text{subject to:} \quad & 9x_{11} + 6x_{21} + 3x_{31} \leq 11 \\ & x_{11}, x_{21}, x_{31} \in \{0,1\}. \end{aligned}$$

The auxiliary problem for machine 2 of the second iteration is:

$$\begin{aligned} \text{maximize} \quad & (6-6)x_{12} + (8-3)x_{22} + (11-11)x_{32} - 0 \\ \text{subject to:} \quad & 5x_{12} + 7x_{22} + 9x_{32} \leq 18 \\ & x_{12}, x_{22}, x_{32} \in \{0,1\}. \end{aligned}$$

Numerical example GAP

The Auxiliary problem for machine 1 of the second iteration is:

$$\begin{aligned} \text{maximize} \quad & 4x_{11} + 4x_{21} - 6x_{31} - 4 \\ \text{subject to:} \quad & 9x_{11} + 6x_{21} + 3x_{31} \leq 11 \\ & x_{11}, x_{21}, x_{31} \in \{0, 1\}. \end{aligned}$$

$x_{11} = 1, x_{21} = x_{31} = 0,\ OF = 0$. Solution (1,0,0) means λ_1^1 which already is at the solution thus OF=0.
The auxiliary problem for machine 2 of the second iteration is:

$$\begin{aligned} \text{maximize} \quad & 5x_{22} \\ \text{subject to:} \quad & 5x_{12} + 7x_{22} + 9x_{32} \leq 18 \\ & x_{12}, x_{22}, x_{32} \in \{0, 1\}. \end{aligned}$$

$x_{12} = 0, x_{22} = 1,\ x_{32} = 0,\ OF = 5$. Solution (0,1,0) means λ_2^2.

Numerical example GAP

We are going to insert column λ_2^2 at the master problem. So, the third iteration Restricted Master Problem is:

$$\begin{aligned}
\text{maximize} \quad & 10\lambda_1^1 + 7\lambda_1^2 + 5\lambda_1^3 + 12\lambda_1^4 + 6\lambda_2^1 + 8\lambda_2^2 + 11\lambda_2^3 + 17\lambda_2^6 \\
\text{subject to:} \quad & \lambda_1^1 + \lambda_2^1 + \lambda_2^6 = 1 \\
& \lambda_1^2 + \lambda_1^4 + \lambda_2^2 = 1 \\
& \lambda_1^3 + \lambda_1^4 + \lambda_2^3 + \lambda_2^6 = 1 \\
& \lambda_1^1 + \lambda_1^2 + \lambda_1^3 + \lambda_1^4 = 1 \\
& \lambda_2^1 + \lambda_2^2 + \lambda_2^3 + \lambda_2^6 = 1 \\
& \lambda_j^k \geq 0
\end{aligned}$$

The optimal solution is: $\lambda_1^1 = 0, \lambda_1^2 = 1, \lambda_1^3 = \lambda_1^4 = \lambda_2^1 = \lambda_2^3 = 0, \lambda_2^6 = 1$.
$\pi_1 = 0, \pi_2 = -3, \pi_3 = 5, \nu_1 = 10, \nu_2 = 12$.

Numerical example GAP

The Auxiliary problem for machine 1 of the third iteration is:

$$\begin{aligned} \text{maximize} \quad & (10-0)x_{11} + (7+3)x_{21} + (5-5)x_{31} - 10 \\ \text{subject to:} \quad & 9x_{11} + 6x_{21} + 3x_{31} \leq 11 \\ & x_{11}, x_{21}, x_{31} \in \{0,1\}. \end{aligned}$$

The auxiliary problem for machine 2 of the third iteration is:

$$\begin{aligned} \text{maximize} \quad & (6-0)x_{12} + (8+3)x_{22} + (11-5)x_{32} - 12 \\ \text{subject to:} \quad & 5x_{12} + 7x_{22} + 9x_{32} \leq 18 \\ & x_{12}, x_{22}, x_{32} \in \{0,1\}. \end{aligned}$$

Numerical example GAP

The Auxiliary problem for machine 1 of the third iteration is:

$$\begin{array}{rl} \text{maximize} & 10x_{11} + 10x_{21} - 10 \\ \text{subject to:} & 9x_{11} + 6x_{21} + 3x_{31} \leq 11 \\ & x_{11}, x_{21}, x_{31} \in \{0,1\}. \end{array}$$

$x_{11} = 1, x_{21} = x_{31} = 0$, $OF = 0$. Solution (1,0,0) means λ_1^1.
The auxiliary problem for machine 2 of the third iteration is:

$$\begin{array}{rl} \text{maximize} & 6x_{12} + 11x_{22} + 6x_{32} - 12 \\ \text{subject to:} & 5x_{12} + 7x_{22} + 9x_{32} \leq 18 \\ & x_{12}, x_{22}, x_{32} \in \{0,1\}. \end{array}$$

$x_{12} = 1, x_{22} = 1$, $x_{32} = 0$, $OF = 5$. Solution (1,1,0) means λ_2^4.

Numerical example GAP

We are going to insert column λ_2^4 at the master problem. So, the fourth iteration Restricted Master Problem is:

$$\begin{aligned}
\text{maximize} \quad & 10\lambda_1^1 + 7\lambda_1^2 + 5\lambda_1^3 + 12\lambda_1^4 + 6\lambda_2^1 + 8\lambda_2^2 + 11\lambda_2^3 + 14\lambda_2^4 + 17\lambda_2^6 \\
\text{subject to:} \quad & \lambda_1^1 + \lambda_2^1 + \lambda_2^4 + \lambda_2^6 = 1 \\
& \lambda_1^2 + \lambda_1^4 + \lambda_2^2 + \lambda_2^4 = 1 \\
& \lambda_1^3 + \lambda_1^4 + \lambda_2^3 + \lambda_2^6 = 1 \\
& \lambda_1^1 + \lambda_1^2 + \lambda_1^3 + \lambda_1^4 = 1 \\
& \lambda_2^1 + \lambda_2^2 + \lambda_2^3 + \lambda_2^6 = 1 \\
& \lambda_j^k \geq 0
\end{aligned}$$

The optimal solution is: $\lambda_1^1 = 0, \lambda_1^2 = 1, \lambda_1^3 = \lambda_1^4 = \lambda_2^1 = \lambda_2^3 = 0, \lambda_2^6 = 1$.
$\pi_1 = 5, \pi_2 = 2, \pi_3 = 5, \nu_1 = 5, \nu_2 = 7$.

Numerical example GAP

The Auxiliary problem for machine 1 of the fourth iteration is:

$$\begin{aligned} \text{maximize} \quad & (10-5)x_{11} + (7-2)x_{21} + (5-5)x_{31} - 5 \\ \text{subject to:} \quad & 9x_{11} + 6x_{21} + 3x_{31} \leq 11 \\ & x_{11}, x_{21}, x_{31} \in \{0,1\}. \end{aligned}$$

The auxiliary problem for machine 2 of the fourth iteration is:

$$\begin{aligned} \text{maximize} \quad & (6-5)x_{12} + (8-2)x_{22} + (11-5)x_{32} - 7 \\ \text{subject to:} \quad & 5x_{12} + 7x_{22} + 9x_{32} \leq 18 \\ & x_{12}, x_{22}, x_{32} \in \{0,1\}. \end{aligned}$$

Numerical example GAP

The Auxiliary problem for machine 1 of the fourth iteration is:

$$\begin{aligned} \text{maximize} \quad & 5x_{11} + 5x_{21} - 5 \\ \text{subject to:} \quad & 9x_{11} + 6x_{21} + 3x_{31} \leq 11 \\ & x_{11}, x_{21}, x_{31} \in \{0, 1\}. \end{aligned}$$

$x_{11} = 1, x_{21} = x_{31} = 0,\ OF = 0$. Solution (1,0,0) means λ_1^1.
The auxiliary problem for machine 2 of the fourth iteration is:

$$\begin{aligned} \text{maximize} \quad & x_{12} + 6x_{22} + 6x_{32} - 7 \\ \text{subject to:} \quad & 5x_{12} + 7x_{22} + 9x_{32} \leq 18 \\ & x_{12}, x_{22}, x_{32} \in \{0, 1\}. \end{aligned}$$

$x_{12} = 0, x_{22} = 1,\ x_{32} = 1,\ OF = 5$. Solution (0,1,1) means λ_2^5.

Numerical example GAP

We are going to insert column λ_2^4 at the master problem. So, the fifth iteration Restricted Master Problem is:

$$
\begin{array}{rl}
\text{maximize} & 10\lambda_1^1 + 7\lambda_1^2 + 5\lambda_1^3 + 12\lambda_1^4 + 6\lambda_2^1 + 8\lambda_2^2 + 11\lambda_2^3 + 14\lambda_2^4 + 19\lambda_2^5 + 17\lambda_2^6 \\
\text{subject to:} & \lambda_1^1 + \lambda_2^1 + \lambda_2^4 + \lambda_2^6 = 1 \\
& \lambda_1^2 + \lambda_1^4 + \lambda_2^2 + \lambda_2^4 + \lambda_2^5 = 1 \\
& \lambda_1^3 + \lambda_1^4 + \lambda_2^3 + \lambda_2^5 + \lambda_2^6 = 1 \\
& \lambda_1^1 + \lambda_1^2 + \lambda_1^3 + \lambda_1^4 = 1 \\
& \lambda_2^1 + \lambda_2^2 + \lambda_2^3 + \lambda_2^5 + \lambda_2^6 = 1 \\
& \lambda_j^k \geq 0
\end{array}
$$

The optimal solution is: $\lambda_1^1 = 1, \lambda_1^2 = \lambda_1^3 = \lambda_1^4 = \lambda_2^1 = \lambda_2^3 = \lambda_2^4 = 0$ $\lambda_2^5 = 1,\ \lambda_2^6 = 0$. $\pi_1 = -2, \pi_2 = 0, \pi_3 = 0, \nu_1 = 12, \nu_2 = 19$.

Numerical example GAP

The Auxiliary problem for machine 1 of the fifth iteration is:

$$\begin{aligned} \text{maximize} \quad & (10+2)x_{11} + (7-0)x_{21} + (5-0)x_{31} - 12 \\ \text{subject to:} \quad & 9x_{11} + 6x_{21} + 3x_{31} \leq 11 \\ & x_{11}, x_{21}, x_{31} \in \{0,1\}. \end{aligned}$$

The auxiliary problem for machine 2 of the fifth iteration is:

$$\begin{aligned} \text{maximize} \quad & (6+2)x_{12} + (8-0)x_{22} + (11-0)x_{32} - 19 \\ \text{subject to:} \quad & 5x_{12} + 7x_{22} + 9x_{32} \leq 18 \\ & x_{12}, x_{22}, x_{32} \in \{0,1\}. \end{aligned}$$

Numerical example GAP

The Auxiliary problem for machine 1 of the fourth iteration is:

$$\begin{aligned} \text{maximize} \quad & 12x_{11} + 7x_{21} + 5x_{31} - 12 \\ \text{subject to:} \quad & 9x_{11} + 6x_{21} + 3x_{31} \leq 11 \\ & x_{11}, x_{21}, x_{31} \in \{0, 1\}. \end{aligned}$$

$x_{11} = 0, x_{21} = x_{31} = 1$, $OF = 0$. Solution (0,1,1) means λ_1^4.
The auxiliary problem for machine 2 of the fourth iteration is:

$$\begin{aligned} \text{maximize} \quad & 8x_{12} + 8x_{22} + 11x_{32} - 19 \\ \text{subject to:} \quad & 5x_{12} + 7x_{22} + 9x_{32} \leq 18 \\ & x_{12}, x_{22}, x_{32} \in \{0, 1\}. \end{aligned}$$

$x_{12} = 1, x_{22} = 0$, $x_{32} = 1$, $OF = 0$. Solution (1,0,1) means λ_2^6.
Thus, there is no more column to add to the Master Problem, so the current solution is optimal.

Numerical example GAP

Optimal solution: $\lambda_1^1 = \lambda_2^5 = 1$ and this means $(1, 0, 0), (0, 1, 1)$, thus task one at machine 1, tasks 2 and 3 at machine 2.

Example - cutting stock

Traditional formulation for cutting stock problem - parameters and variables

- A company produces bars of total length L;
- The customers needs bars of lengths $l_i, \ i = 1, ..., n$;
- the right side $b_i, \ i = 1, ...n$ of the constraints is related to the number of bars of size l_i the costumers need (demand);
- the variables x_{ij} represent be the number of times item i is cut on bar j;
- y_j is 1 if the bar j is cut and 0 otherwise;

Example - cutting stock

Traditional formulation for cutting stock problem (Kantarovich)

$$
\begin{aligned}
\text{minimize} \quad & z = \sum_{j=1}^{m} y_j \\
\text{subject to:} \quad & \sum_{j=1}^{m} x_{ij} \geq b_i \quad \text{for } i = 1, ..., n \\
& \sum_{i=1}^{n} l_i x_{ij} \leq L y_j \ \text{ for } j = 1, ..., m \\
& x_{ij} \geq 0, \text{integer}, \ \ i = 1, ..., n, \ \ j = 1, ..., m. \\
& y_j, \ \text{binary}, j = 1, ..., m.
\end{aligned}
$$

Example - cutting stock

The traditional formulation is not good because (see [5])

- when we find integer values for one bar, the fractional values that might be found previously can appear again as the solution for another bar.
- fractional values are easily found because we can have unused parts of each bar.

Example cutting stock

- In the column generation formulation, the columns are related to the patterns;
- Imagine a bar of total length L$= 5$ and the costumer wants bars of $l_1 = 1$ and $l_2 = 3$ and $l_3 = 4$;
- One possible pattern would be to cut the bar in 1 of l_1 and one of l_3 and the column would be $\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$;
- Another possible pattern would be to cut the bar in 2 of $l_1 = 1$ and 1 of $l_2 = 3$ and and the column would be $\begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}$;

Example cutting stock

Column generation formulation.

- very large number of patterns P;
- For each $p \in P$, $a_{ip} \in \mathbb{Z}_+$ denote the number of pieces of length l_i in a pattern p;
- λ_p is the number of bars cut in pattern p.

Example cutting stock

Restricted Master Problem (Gilmore-Gomory 1960)

$$\begin{aligned}(RMP)\ \text{minimize}\quad & z = \sum_{p \in P} c_p \lambda_p \\ \text{subject to:}\quad & \sum_{p \in P} a_{ip} \lambda_p \geq b_i,\ \ i = 1, \ldots n \\ & \lambda_p \geq 0, \text{integer},\ \ p \in P.\end{aligned}$$

Example cutting stock

We solve a relaxed *RMP* with a small subset of patterns $P' \subset P$ and generate new patterns as needed.

$$\begin{aligned}(RMP)\ \text{minimize}\quad & z = \sum_{p \in P'} c_p \lambda_p \\ \text{subject to:}\quad & \sum_{p \in P'} a_{ip} \lambda_p \geq b_i,\ \ i = 1, \ldots m \\ & \lambda_p \geq 0,\ \ p \in P'.\end{aligned}$$

In this relaxed problem we don't see the convexity constraint $\sum_{p \in P'} \lambda_p = 1$ because Gilmore and Gomory don't apply Dantzig-Wolfe reformulation.

Example - cutting stock

General auxiliary problem

$$\begin{aligned} (AP) \text{ minimize} \quad & z = (c^\top - \pi * A)v \\ \text{subject to:} \quad & Av \leq b \\ & v \geq 0. \end{aligned}$$

which can be written as

$$\begin{aligned} (AP) \text{ minimize} \quad & z = (c_N^\top - \pi * A_N)x_N \\ \text{subject to:} \quad & Ax \leq b \\ & x \geq 0. \end{aligned}$$

where N denotes the set of non basic variables. But this auxiliary problem, given in our first class is a general auxiliary problem.

Example - cutting stock

In cutting stock problem, the variables of auxiliary problem are the columns. So, we can write our auxiliary problem as:

$$\begin{aligned} (AP)\ \text{minimize} \quad & z = 1 - \sum_{i=1}^{n} \pi_i * a_{ij} \\ \text{subject to:} \quad & \sum_{i=1}^{n} l_i a_{ij} \leq L \\ & a_{ij} \geq 0, \text{integer}. \end{aligned}$$

As we have that min z = - max - z, (AP) can be rewritten as:

$$\begin{aligned} 1 - (AP)\ \text{maximize} \quad & z = \sum_{i=1}^{n} \pi_i * a_{ij} \\ \text{subject to:} \quad & \sum_{i=1}^{n} l_i a_{ij} \leq L \\ & a_{ij} \geq 0, \text{integer}. \end{aligned}$$

So, our auxiliary problem is a knapsack problem. And the reduced cost is given by

Example - cutting Stock

A company produces steel bars with $L = 100m$ and cuts the bars for the costumers according to their necessities. Now, the company has to satisfy the following demand:

l_i	number of pieces needed
22	45
42	38
52	25
53	11
78	12

The costs c_p are all equal to one.
We want to minimize the number of steel bars we need to cut in order to satisfy the demand.

Example - cutting stock

In order to find an initial solution, we compute how many pieces of each length fits in one bar.

$$\left\lfloor \frac{L}{l_i} \right\rfloor$$

So we make:

$$\left\lfloor \frac{100}{22} \right\rfloor = 4, \left\lfloor \frac{100}{42} \right\rfloor = 2, \left\lfloor \frac{100}{52} \right\rfloor = 1, \left\lfloor \frac{100}{53} \right\rfloor = 1, \left\lfloor \frac{100}{78} \right\rfloor = 1.$$

so our matrix A at the first iteration is:

$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Example - cutting stock

Our first *RMP* is:

$$\begin{aligned} (RMP1)\ \text{minimize} \quad & z = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 \\ \text{subject to:} \quad & 4\lambda_1 \geq 45 \\ & 2\lambda_2 \geq 38 \\ & \lambda_3 \geq 25 \\ & \lambda_4 \geq 11 \\ & \lambda_5 \geq 12 \\ & \lambda_1, \ldots, \lambda_5 \geq 0. \end{aligned}$$

The solution is: $\lambda_1 = 11.25,\ \lambda_2 = 19,\ \lambda_3 = 25, \lambda_4 = 11,\ \lambda_5 = 12.$

The value of the dual variables are:

$\pi_1 = 0.25,\ \pi_2 = 0.5,\ \pi_3 = \pi_4 = \pi_5 = 1.$

Example - cutting stock

In our auxiliary problem we want to check if there exists a new column with negative reduced cost.

$$\begin{aligned} (AP1)\ \text{maximize} \quad & z = 0.25x_1 + 0.5x_2 + 1x_3 + 1x_4 + 1x_5 \\ \text{subject to:} \quad & 22x_1 + 42x_2 + 52x_3 + 53x_4 + 78x_5 \leq 100 \\ & x_1, ..., x_5 \geq 0 \text{ and integer.} \end{aligned}$$

The solution of $(AP1)$ is $x_1 = x_3 = x_5 = 0, x_2 = x_4 = 1$. Hence, the reduced cost is $1 - 1.5 = -0.5$ and we have a new column a_6,

$$a_6 = \begin{pmatrix} a_{16} \\ a_{26} \\ a_{36} \\ a_{46} \\ a_{56} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Example - cutting stock

$$\begin{aligned}(RMP2)\ \text{minimize}\quad & z = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6\\ \text{subject to:}\quad & 4\lambda_1 \geq 45\\ & 2\lambda_2 + \lambda_6 \geq 38\\ & \lambda_3 \geq 25\\ & \lambda_4 + \lambda_6 \geq 11\\ & \lambda_5 \geq 12\\ & \lambda_1, ..., \lambda_6 \geq 0.\end{aligned}$$

The solution is:
$\lambda_1 = 11.25,\ \lambda_2 = 13.5,\ \lambda_3 = 25, \lambda_4 = 0,\ \lambda_5 = 12, \lambda_6 = 11.$
The value of the dual variables are:
$\pi_1 = 0.25,\ \pi_2 = \pi_4 = 0.5,\ \pi_3 = \pi_5 = 1.$

Example - cutting stock

$$\begin{aligned}(AP2)\ \text{maximize}\quad & z = 0.25x_1 + 0.5x_2 + 1x_3 + 0.5x_4 + 1x_5 \\ \text{subject to:}\quad & 22x_1 + 42x_2 + 52x_3 + 53x_4 + 78x_5 \leq 100 \\ & x_1, ..., x_5 \geq 0 \text{ and integer.}\end{aligned}$$

The solution of $(AP2)$ is $x_1 = x_4 = x_5 = 0, x_2 = x_3 = 1$. Hence, the reduced cost is $1 - 1.5 = -0.5$ so we have a new column a_7,

$$a_7 = \begin{pmatrix} a_{17} \\ a_{27} \\ a_{37} \\ a_{47} \\ a_{57} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Example - cutting stock

$$
\begin{array}{rl}
(RMP3) \text{ minimize} & z = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 + \lambda_7 \\
\text{subject to:} & 4\lambda_1 \geq 45 \\
& 2\lambda_2 + \lambda_6 + \lambda_7 \geq 38 \\
& \lambda_3 + \lambda_7 \geq 25 \\
& \lambda_4 + \lambda_6 \geq 11 \\
& \lambda_5 \geq 12 \\
& \lambda_1, ..., \lambda_7 \geq 0.
\end{array}
$$

The solution is:
$\lambda_1 = 11.25,\ \lambda_2 = 1,\ \lambda_3 = \lambda_4 = 0,\ \lambda_5 = 12, \lambda_6 = 11, \lambda_7 = 25.$
The value of the dual variables are:
$\pi_1 = 0.25,\ \pi_2 = \pi_3 = \pi_4 = 0.5, \pi_5 = 1.$

Example - cutting stock

$$\begin{aligned}(AP3)\ \text{maximize} \quad & z = 0.25x_1 + 0.5x_2 + 0.5x_3 + 0.5x_4 + 1x_5 \\ \text{subject to:} \quad & 22x_1 + 42x_2 + 52x_3 + 53x_4 + 78x_5 \leq 100 \\ & x_1, ..., x_5 \geq 0 \text{ and integer.}\end{aligned}$$

The solution of $(AP3)$ is $x_1 = x_5 = 1, x_2 = x_3 = x_4 = 0$. Hence, the reduced cost is $1 - 1.25 = -0.25$ so we have a new column a_8,

$$a_8 = \begin{pmatrix} a_{18} \\ a_{28} \\ a_{38} \\ a_{48} \\ a_{58} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Example - cutting stock

$$\begin{aligned} (RMP4)\ \text{minimize} \quad & z = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6 + \lambda_7 + \lambda_8 \\ \text{subject to:} \quad & 4\lambda_1 + \lambda_8 \geq 45 \\ & 2\lambda_2 + \lambda_6 + \lambda_7 \geq 38 \\ & \lambda_3 + \lambda_7 \geq 25 \\ & \lambda_4 + \lambda_6 \geq 11 \\ & \lambda_5 + \lambda_8 \geq 12 \\ & \lambda_1, ..., \lambda_8 \geq 0. \end{aligned}$$

The solution is:
$\lambda_1 = 8.25,\ \lambda_2 = 1,\ \lambda_3 = \lambda_4 == \lambda_5 = 0, \lambda_6 = 11, \lambda_7 = 25, \lambda_8 = 12.$
The value of the dual variables are:
$\pi_1 = 0.25,\ \pi_2 = \pi_3 = \pi_4 = 0.5,\ \pi_5 = 0.75.$

Example - cutting stock

$$
\begin{aligned}
(AP4)\ \text{maximize} \quad & z = 0.25x_1 + 0.5x_2 + 0.5x_3 + 0.5x_4 + 0.75x_5 \\
\text{subject to:} \quad & 22x_1 + 42x_2 + 52x_3 + 53x_4 + 78x_5 \leq 100 \\
& x_1, ..., x_5 \geq 0 \text{ and integer.}
\end{aligned}
$$

The solution of $(AP4)$ is $x_1 = 4,\ x_2 = x_3 = x_4 = x_5 = 0$. In this case, the reduced cost is 0, thus we don't have a new column and the optimal solution was found at $(RMP4)$.

Example - cutting stock

The correct thing to do would be to run a B&B in the solution of RMP4 but we are going to just round it. Rounding the solution of RMP4 we have the following.

- cut 9 entire bars in pattern 1, which means 4 pieces of 22m;
- cut 1 entire bar in pattern 2, which means 2 pieces of 42m;
- cut 11 entire bars in pattern 6, which means 1 piece of 42m and 1 piece of 53m;
- cut 25 entire bars in pattern 7, which means 1 piece of 42m and 1 piece of 52m;
- 12 entire bars in pattern 8, which means 1 piece of 22m and 1 piece of 78m.

This solution gives us 48 pieces of 22m, 38 pieces of 42m, 25 pieces of 52m, 11 pieces of 53m and 12 pieces of 78m.

Integer Programming - revision - Branch and Cut

- The Branch and Cut is a method for solving integer linear programming problems;
- It is composed by two other methods: cutting planes and branch and bound;
    - Cutting planes are found by solving separation problems, which means finding violated valid inequalities for the integer problem that we want to solve.
    - In the cutting planes method, we solve the relaxation of the integer linear programming problem and obtain a fractional solution x^*;
    - We look for a new inequality $\alpha^\top x \leq \beta$ that is valid for the integer problem (every $x \in \mathbb{Z}^n$ that satisfies $Ax \leq b$ also satisfies the new inequality) and cuts off the fractional solution x^* (which means $\alpha^\top x^* > \beta$);
    - The cuts are made in the hope of obtaining integer solutions to the problem;
    - The separation problem also can be solved heuristically.

Gomory Cuts

Let x^* be an optimal solution of the linear relaxation of the current formulation $min\ c^\top x : Ax = b, x \geq 0$. Let $x^*_{B_r}$ be a basic variable of the problem with fractional optimal value.
The corresponding row of the optimal tableau:

$$x_{B_r} + \sum_{j: x_j \in N} \bar{a}_{rj} x_j = \bar{b}_r$$

where $\bar{b}_r$ is fractional.

Gomory Cuts

The Gomory cut with respect to the fractional basic variable x_{B_r}

$$\sum_{j:x_j \in N} (\bar{a}_{rj} - \lfloor \bar{a}_{rj} \rfloor) x_j \geq (\bar{b}_r - \lfloor \bar{b}_r \rfloor)$$

is a cutting plane with respect to the fractional x^*.

Gomory cuts - Example

$$\begin{aligned}\text{maximize}\quad & 8x_1 + 5x_2\\ \text{subject to:}\quad & x_1 + x_2 \leq 6\\ & 9x_1 + 5x_2 \leq 45\\ & x_1, x_2 \geq 0, \text{integer}\end{aligned}$$

Considering the slack variables s_1 and s_2, he optimal tableau is:

	x_1	x_2	s_1	s_2	
	0	0	-1.25	-0.75	41.25
x_1	1	0	-1.25	0.25	3.75
x_2	0	1	2.25	-0.25	2.25

Gomory cuts - Example

Choosing the basic variable x_1, which has fractional optimal value, we have:

$$x_1 - 1.25s_1 + 0.25s_2 = 3.75$$

So, using

$$\sum_{j:x_j \in N} (\bar{a}_{rj} - \lfloor \bar{a}_{rj} \rfloor) x_j \geq (\bar{b}_r - \lfloor \bar{b}_r \rfloor)$$

the Gomory cut will be

$$(-1.25 - (-2))s_1 + (0.25 - 0)s_2 \geq (3.75 - 3)$$

$$0.75s_1 + 0.25s_2 \geq 0.75$$

With a new slack variable s_3 we have

$$0.75s_1 + 0.25s_2 - s_3 = 0.75$$

We also can write as

$$-0.75s_1 - 0.25s_2 + s_3 = -0.75$$

Gomory cuts - Example

The new tableau is

	x_1	x_2	s_1	s_2	s_3	
	0	0	-1.25	-0.75	0	41.25
x_1	1	0	-1.25	0.25	0	3.75
x_2	0	1	2.25	-0.25	0	2.25
s_3	0	0	-0.75	-0.25	1	-0.75

The optimal tableau is:

	x_1	x_2	s_1	s_2	s_3	
	0	0	0	-0.33	-1.67	40
x_1	1	0	0	0.67	-1.67	5
x_2	0	1	0	-1	3	0
s_1	0	0	1	0.33	-1.33	1

Reaching the integer optimal solution $x_1 = 5$, $x_2 = 0$, $s_1 = 1$, $s_2 = s_3 = 0$ and $z^* = 40$.

Gomory cuts - Example

We can also rewrite the cut using the decision variables. The cut is

$$0.75s_1 + 0.25s_2 \geq 0.75$$

and we have

$$s_1 = 6 - x_1 - x_2$$

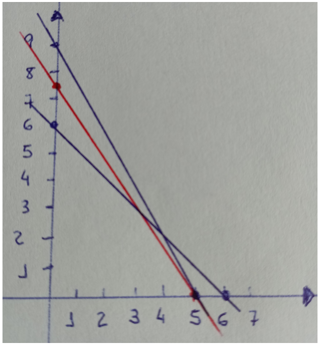
$$s_2 = 45 - 9x_1 - 5x_2.$$

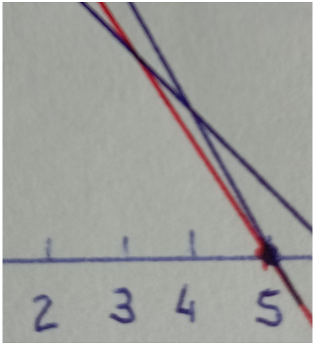
Making the substitutions we have the cut rewritten as:

$$3x_1 + 2x_2 \leq 15.$$

Gomory cut - Exanple

The figure at the left presents the feasible region of the problem obtained with the two original constraints (in blue color) , $x_1 + x_2 \leq 6$ and $9x_1 + 5x_2 \leq 45$ and the Gomory cut $3x_1 + 2x_2 \leq 15$ (in red color). The figure at the right is a zoom of the other figure, showing better that the red constraint really cuts the optimal solution of the relaxed problem.



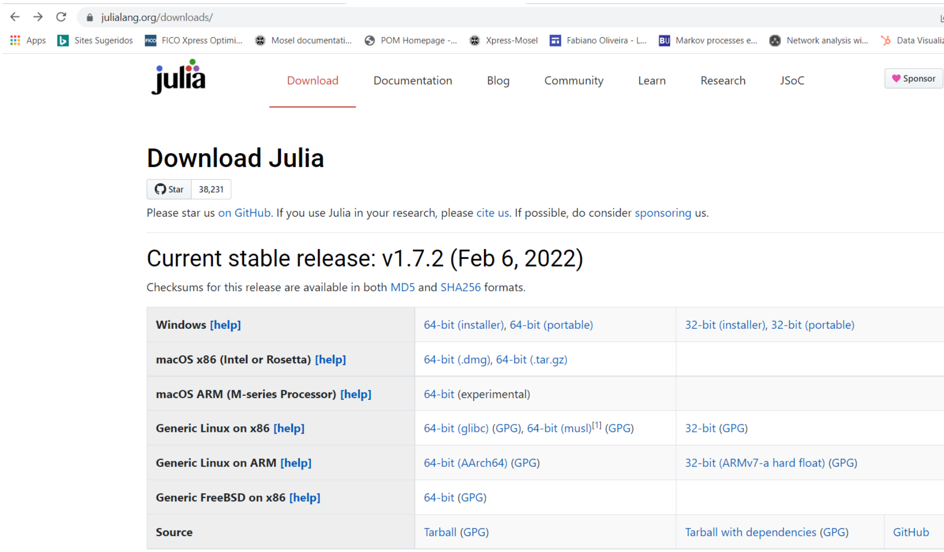


Branch and Cut

- The Gomory cuts are not strong cuts, therefore, its use causes a slow convergence for the algorithm;
- The development of the polyhedral theory led to stronger cutting planes;
- Strong cuts corresponds to facets of the convex hull of integral feasible points of the problem;
- As each problem has its convex hull related to its integer feasible solutions, it's natural that we have a branch and cut algorithm for each problem.

For more details, see [6].

Basic Julia language - JuMP package



Download Julia

Star 38,231

Please star us on GitHub. If you use Julia in your research, please cite us. If possible, do consider sponsoring us.

Current stable release: v1.7.2 (Feb 6, 2022)

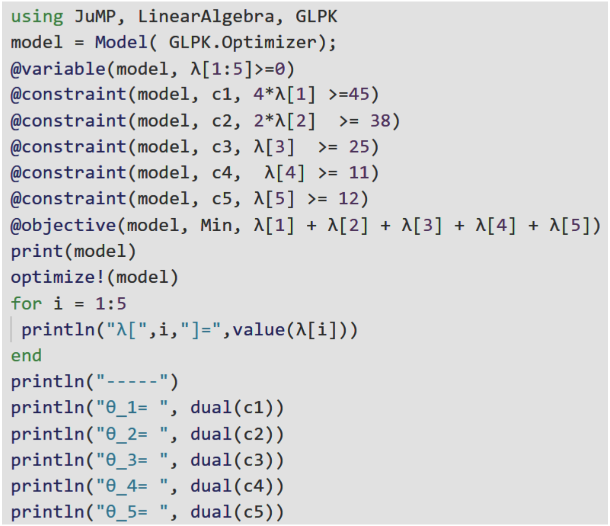
Checksums for this release are available in both MD5 and SHA256 formats.

Windows [help]	64-bit (installer), 64-bit (portable)	32-bit (installer), 32-bit (portable)	
macOS x86 (Intel or Rosetta) [help]	64-bit (.dmg), 64-bit (.tar.gz)		
macOS ARM (M-series Processor) [help]	64-bit (experimental)		
Generic Linux on x86 [help]	64-bit (glibc) (GPG), 64-bit (musl)[1] (GPG)	32-bit (GPG)	
Generic Linux on ARM [help]	64-bit (AArch64) (GPG)	32-bit (ARMv7-a hard float) (GPG)	
Generic FreeBSD on x86 [help]	64-bit (GPG)		
Source	Tarball (GPG)	Tarball with dependencies (GPG)	GitHub

Basic Julia language - JuMP package

- When you click at the link [help] that is at the right side of Windows, it gives you some steps and instructions which really helps the installation;
- After downloading and installing Julia, you have to get package JuMP. Open the Julia terminal and write
  import Pkg
  Pkg.add("JuMP");
- The same thing must be made to solver GLPK
  import Pkg
  Pkg.add("GLPK");
- If you use Windows, you can use Anaconda `https://www.anaconda.com/products/individual`, which is free.
- within Anaconda you can program in Julia using Microsoft Visual Studio or Jupyter Notebook, for example.

Basic Julia language - JuMP package

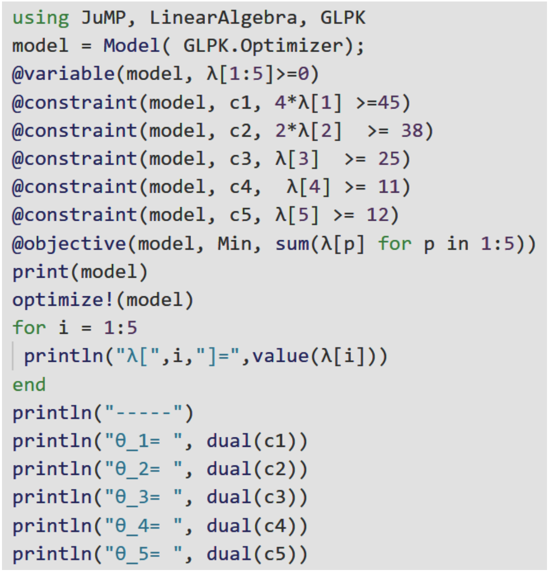


```
using JuMP, LinearAlgebra, GLPK
model = Model( GLPK.Optimizer);
@variable(model, λ[1:5]>=0)
@constraint(model, c1, 4*λ[1] >=45)
@constraint(model, c2, 2*λ[2]  >= 38)
@constraint(model, c3, λ[3]  >= 25)
@constraint(model, c4,  λ[4] >= 11)
@constraint(model, c5, λ[5] >= 12)
@objective(model, Min, λ[1] + λ[2] + λ[3] + λ[4] + λ[5])
print(model)
optimize!(model)
for i = 1:5
  println("λ[",i,"]=",value(λ[i]))
end
println("-----")
println("θ_1= ", dual(c1))
println("θ_2= ", dual(c2))
println("θ_3= ", dual(c3))
println("θ_4= ", dual(c4))
println("θ_5= ", dual(c5))
```

Julia language - JuMP package

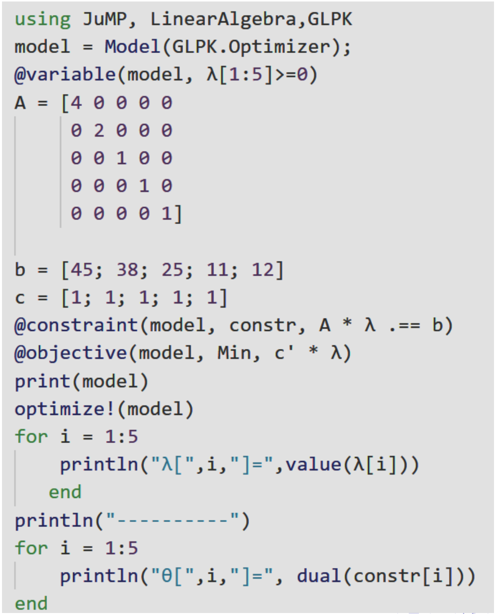
- At line 2 you can see that model was the name I chose to this restricted master problem, but it could be RMP, for example;
- at line 3 we are adding the variables to our model, which is named model;
- 4-8 are related to the constraints, one by one;
- line 9 is related to the objective function;
- at line 10 we print the model;
- at line 11 we solve the model;
- at the next lines we print the values of the primal variables and dual variables respectively;
- I chose to call θ the dual variables because π in Julia is the mathematical constant 3.1415...

Basic Julia language - JuMP package



```julia
using JuMP, LinearAlgebra, GLPK
model = Model( GLPK.Optimizer);
@variable(model, λ[1:5]>=0)
@constraint(model, c1, 4*λ[1] >=45)
@constraint(model, c2, 2*λ[2]  >= 38)
@constraint(model, c3, λ[3]  >= 25)
@constraint(model, c4,  λ[4] >= 11)
@constraint(model, c5, λ[5] >= 12)
@objective(model, Min, sum(λ[p] for p in 1:5))
print(model)
optimize!(model)
for i = 1:5
 println("λ[",i,"]=",value(λ[i]))
end
println("-----")
println("θ_1= ", dual(c1))
println("θ_2= ", dual(c2))
println("θ_3= ", dual(c3))
println("θ_4= ", dual(c4))
println("θ_5= ", dual(c5))
```

Basic Julia language - JuMP package



```
using JuMP, LinearAlgebra,GLPK
model = Model(GLPK.Optimizer);
@variable(model, λ[1:5]>=0)
A = [4 0 0 0 0
     0 2 0 0 0
     0 0 1 0 0
     0 0 0 1 0
     0 0 0 0 1]

b = [45; 38; 25; 11; 12]
c = [1; 1; 1; 1; 1]
@constraint(model, constr, A * λ .== b)
@objective(model, Min, c' * λ)
print(model)
optimize!(model)
for i = 1:5
    println("λ[",i,"]=",value(λ[i]))
   end
println("----------")
for i = 1:5
    println("θ[",i,"]=", dual(constr[i]))
end
```

Basic Julia language - JuMP package

- if you want your variables to be integer
  @variable(model, x[1:5]>=0, Int);
- if you want your variables to be binary
  @variable(model, x[1:5], Bin);
- creating a free (unbounded) variable
  @variable(model, free_x);
- if you want to use Gurobi as solver
  model = Model(Gurobi.Optimizer);
- if some variable has lower and upper bounds, for example
  @variable(model, 0 <= x <= 30) or @variable(model, x, lower_bound = 0, upper_bound = 30)
- if you want the value of the objective function
  objective_value(model);

Basic Julia language - JuMP package

Maybe you will find these functions helpful to your implementation:

- hcat;
- push! ;
- set_objective_coefficient;
- set_normalized_coefficient.

Thank you

I would like to thank you all.
The files of the classes are at `https://github.com/anauzeda/ENSIIE-2022`
Please, you can contact me in case of any doubt at my e-mail af.macambira@gmail.com.br

Bibliography I

[1] C. Barnhart e et al., "Branch-and-Price: Column Generation for Solving Huge Integer Programs.," *Operations Research, JSTOR*, v. 46, n. 3, pp. 316–329, 1998.

[2] N. Maculan, M. M. Passini, J. A. M. Brito e I. Loiseau, "Column-generation in integer linear programming," en, *RAIRO - Operations Research - Recherche Opérationnelle*, v. 37, n. 2, pp. 67–83, 2003. DOI: `10.1051/ro:2003014`. endereço: `www.numdam.org/item/RO_2003__37_2_67_0/`.

[3] E. L. Johnson, "Modeling and Strong Linear Programs for Mixed Integer Programming," em *Algorithms and Model Formulations in Mathematical Programming*, S. W. Wallace, ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 1–43, ISBN: 978-3-642-83724-1.

[4] Accessed:2021-02-7. endereço: `https://users.mai.liu.se/torla64/MAI0127/ch13.3-13.5.pdf`.

[5] H. Ben Amor e J. Carvalho, "Cutting Stock Problems," em jan. de 1970, pp. 131–161, ISBN: 0-387-25485-4. DOI: `10.1007/0-387-25486-2_5`.

[6] J. E. Mitchell, "Branch-and-Cut Algorithms for Combinatorial Optimization Problems,", 1988.