Decision Aid Methodologies In Transportation
Lecture 3: Integer programming

Chen Jiang Hang

Transportation and Mobility Laboratory

May 13, 2013

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Integer programming

The (linear) integer programming

$$\begin{aligned} \min: \quad & \mathbf{c}'\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \\ & x_i \in \mathbb{Z}, \forall i \in \mathcal{I} \end{aligned}$$

Suppose $\mathbf{x} \in \mathbb{R}^n$. If $|\mathcal{I}| = n$, the integer programming problem is called **pure** integer programming problem; Otherwise, if $|\mathcal{I}| < n$, the problem is called **mixed** integer programming (MIP) problem. MIP models have extremely wide applications in practice!

The relaxation of integer programming problem

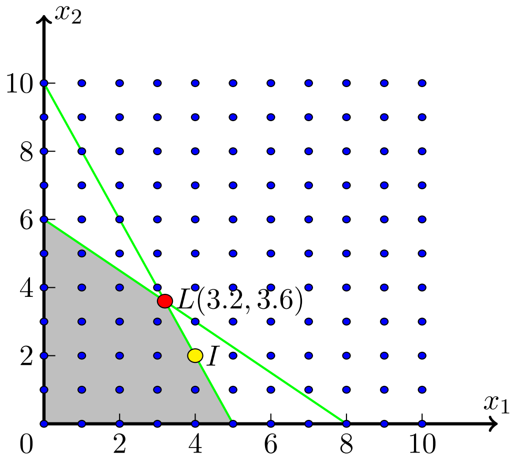
The linear relaxation problem of an integer programming problem is very crucial since it provides useful bound information for the integer one.

$$\begin{aligned} \min: \quad & \mathbf{c}'\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \\ & x_i \in \mathbb{Z}, \forall i \in \mathcal{I} \end{aligned} \qquad \begin{aligned} \min: \quad & \mathbf{c}'\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Question: if the sense of the objective function is minimization, the optimal value of the linear relaxation problem is a/an (lower/upper) bound of the integer programming problem?

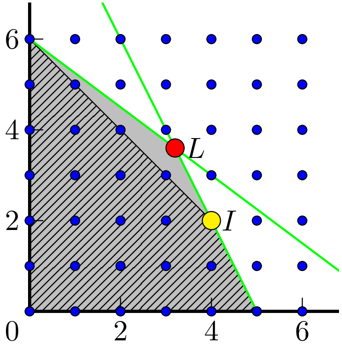
The feasible region of an integer programming problem

$$\max\{7x_1+5x_2 \mid 2x_1+x_2 \le 10, 3x_1+4x_2 \le 24, x_1, x_2 \ge 0, x_1, x_2 \in \mathbb{Z}\}$$



The feasible region of an integer programming problem

The hatched region is called the **convex hull** of the feasible region of the integer programming problem. The convex hull has an very important property: let $P = \{\mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}, x_i \in \mathbb{Z}, \forall i \in \mathcal{I}\}$ and denote the convex hull for P as CH(P); then the optimal solution of an integer programming problem is exactly the same as its linear relaxation problem under the feasible region CH(P).



The feasible region of a general integer programming problem

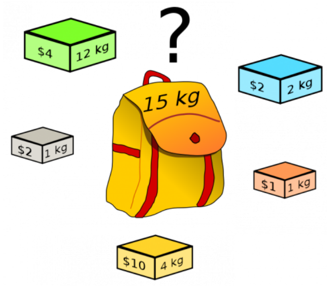
- Unfortunately, how to obtain CH(P) for an integer programming problem is an extremely hard problem (i.e., $\mathcal{NP}$-hard).
- If you can develop an algorithm with polynomial complexity to find out CH(P) (more generally, $\mathcal{P} = \mathcal{NP}$), you will be awarded US$ 1,000,000 by Clay Mathematics Institute.

Integer programming formulation techniques

The power of binary decision variables: **To model the binary choice**.

Knapsack problem

We are given n items. The jth item has weight w_j and its value is c_j. Given a bound K on the weight that can be carried in a knapsack, we would like to select items to maximize the total value.



Integer programming formulation techniques

We define a binary decision variable x_j which is 1 if item j is chosen, and 0 otherwise. The knapsack problem can then be formulated as:

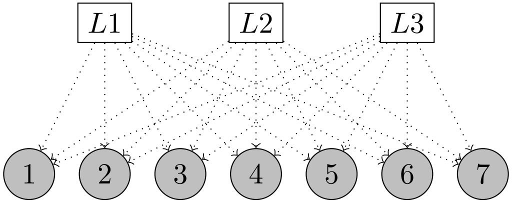
$$\begin{aligned} \max: \quad & \sum_{j=1}^{n} c_j x_j \\ s.t. \quad & \sum_{j=1}^{n} w_j x_j \leq K \\ & x_j \in \{0,1\}, \forall j = 1, \ldots, n \end{aligned}$$

Integer programming formulation techniques

The power of binary decision variables: **To model the dependency between decisions**.

Facility location problem

Suppose we are given n potential facility locations and a list of m clients who need to be serviced from these locations. There is a fixed cost c_j of opening a facility at location j, while there is a cost d_{ij} of serving client i from facility j. The goal is to select a set of facility locations and assign each client to one facility , while minimizing the total cost.



Integer programming formulation techniques

We define a binary decision variable y_j which is 1 if facility j is selected, and 0 otherwise. In addition, we define another binary variable x_{ij}, which is 1 if client i is served by facility j, and 0 otherwise. The facility location problem can be formulated as follows:

$$\begin{aligned} \max: \quad & \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} x_{ij} \\ s.t. \quad & \sum_{j=1}^{n} x_{ij} = 1, \forall i \\ & x_{ij} \leq y_j, \forall i, j \\ & x_{ij}, y_j \in \{0,1\}, \forall i, j \end{aligned}$$

Integer programming formulation techniques

The power of binary decision variables: **To model the requirement of "at least one/exactly one/at most one"**.

At least one: $\sum_{j=1}^{n} x_j \geq 1$
Exactly one: $\sum_{j=1}^{n} x_j = 1$
At most one: $\sum_{j=1}^{n} x_j \leq 1$

In the facility location problem, we have used the "exactly one" type constraints.

Question: in the facility location problem, if we impose another requirement—at most 2 locations can be chosen, then how to model?

Integer programming formulation techniques

The power of binary decision variables: **To model the "if–then" decision**.

If–then decision

If $f(x) \geq 0$ then $g(x) \geq 0$

To mathematically express the above "if–then" decision, we can introduce an auxiliary binary decision variable y. The meaning of y is

$$y = \begin{cases} 1, & \text{if } f(x) \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$

To ensure such a relationship, let M (ϵ) be a sufficiently large (small) positive number.

$$\begin{aligned} f(x) + \epsilon &\leq M \cdot y \\ g(x) &\geq M(y-1) \end{aligned}$$

Integer programming formulation techniques

Then to enforce that if $f(x) \geq 0$ then $g(x) \geq 0$, we also need to add the following constraint:

$$g(x) \geq M(y-1)$$

The above method is usually called **big-M method**.

Pro and Con of the big-M method

Advantage: maintain the linearity of constraints

Disadvantage: adverse effect on the bound quality of the linear relaxation problem if tight M is not used

Advantage of the big-M method

Consider the following question:

- Let x be the amount of useful knowledge that you will receive from this decision-aid methodology course.
- Let $y = 1$ if the lecturer is good; 0, otherwise.
- Let $z = 1$ if you attend the course; 0, otherwise.
- Let $w = 1$ if you attend the laboratory; 0, otherwise.

Write the constraint stating that if the lecturer is good and you attend the course and the laboratory, then the amount of knowledge you will learn from this course should be greater than K.

Advantage of the big-M method

Consider the following question:

- Let x be the amount of useful knowledge that you will receive from this decision-aid methodology course.
- Let $y = 1$ if the lecturer is good; 0, otherwise.
- Let $z = 1$ if you attend the course; 0, otherwise.
- Let $w = 1$ if you attend the laboratory; 0, otherwise.

Write the constraint stating that if the lecturer is good and you attend the course and the laboratory, then the amount of knowledge you will learn from this course should be greater than K.

1. Some people will choose the constraint: $x \geq K \cdot y \cdot z \cdot w$

Advantage of the big-M method

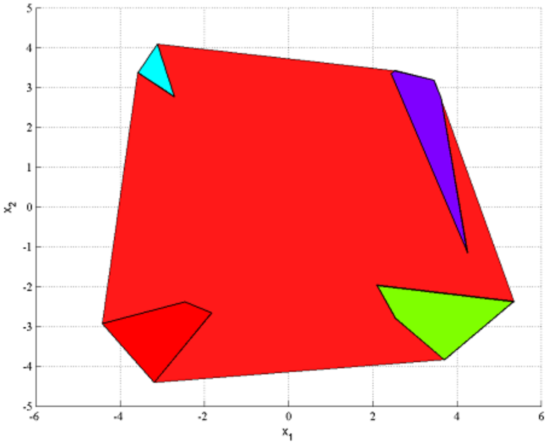
Consider the following question:

- Let x be the amount of useful knowledge that you will receive from this decision-aid methodology course.
- Let $y = 1$ if the lecturer is good; 0, otherwise.
- Let $z = 1$ if you attend the course; 0, otherwise.
- Let $w = 1$ if you attend the laboratory; 0, otherwise.

Write the constraint stating that if the lecturer is good and you attend the course and the laboratory, then the amount of knowledge you will learn from this course should be greater than K.

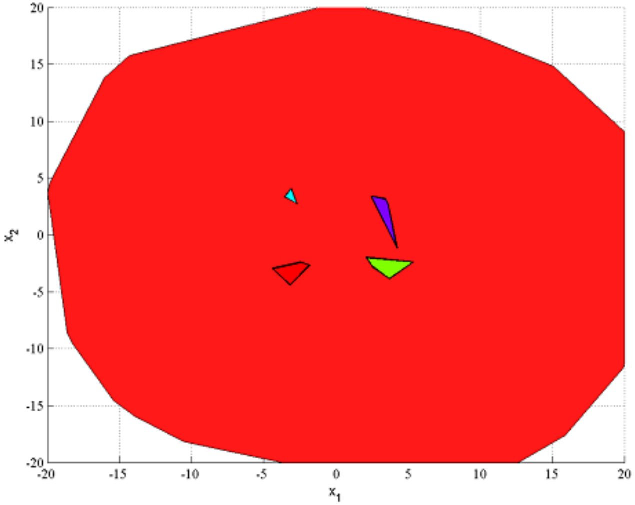
1. Some people will choose the constraint: $x \geq K \cdot y \cdot z \cdot w$
2. By using big-M method, $x \geq K - M(3 - y - z - w)$

Disadvantage of the big-M method



note: the picture is from YALMIP website.

Disadvantage of the big-M method



note: the picture is from YALMIP website.

Integer programming applications

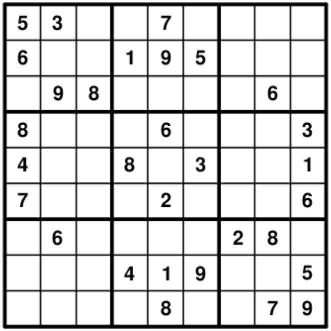
N queens problem

Consider a $N \times N$ chessboard (a generalized version of a 8×8 chessboard). What is the maximum number of queens that can be placed on the board such that no queen is in a confronting position with any other. Here a confronting position is defined if two queens are either in the same row, same column, or the same diagonal.

Integer programming applications

A Sudoku Problem

- Within any of the 9 individual 3×3 boxes, each of the numbers 1 to 9 must be found
- With any column (row) of the 9×9 grid, each of the numbers 1 to 9 must be found



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Cutting plane method

To illustrate how cutting plane method works, let's consider **pure** integer programming problem.

$$\begin{aligned}[\mathcal{IP}]\min: \quad & \mathbf{c}'\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \\ & x_i \in \mathbb{Z}, \forall i\end{aligned} \qquad \begin{aligned}[\mathcal{LP}]\min: \quad & \mathbf{c}'\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}\end{aligned}$$

The main idea in cutting plane method is to solve $[\mathcal{IP}]$ by solving a sequence of linear programming problems. We first solve $[\mathcal{LP}]$ and find an optimal solution $\mathbf{x}^*$. If $\mathbf{x}^*$ is integer, then it is an optimal solution to $[\mathcal{IP}]$. If not, **we find an inequality that all integer solutions to $[\mathcal{IP}]$ satisfy, but $\mathbf{x}^*$ does not**. We add this inequality to the linear programming problem $[\mathcal{LP}]$ to obtain a tighter relaxation.

Cutting plane method

A generic cutting plane algorithm

1. Solving the linear programming relaxation $[\mathcal{LP}]$. Let $\mathbf{x}^*$ be an optimal solution.
2. If $\mathbf{x}^*$ is integer, stop.
3. If not, add a linear inequality constraint (called a cut or a valid inequality) to $[\mathcal{LP}]$ that all integer solutions to $[\mathcal{IP}]$ satisfy, but $\mathbf{x}^*$ does not; go to Step 1.

Cutting plane method

Here we will introduce Gomory cuts for solving pure integer programming problem.
Let $\hat{\mathbf{x}}$ be an optimal basic feasible solution for $[\mathcal{LP}]$ and let $\hat{\mathbf{B}}$ be the associated optimal basis and $\hat{\mathbf{N}}$ be the rest of $\mathbf{A}$. We partition $\mathbf{x}$ (a general feasible solution) into a subvector $\mathbf{x}_B$ (associated with a basis $\mathbf{B}$) of basic variables and subvector $\mathbf{x}_N$ of nonbasic variables. In $[\mathcal{LP}]$, we have $\mathbf{Ax} = \mathbf{b}$. Multiply both sides $\hat{\mathbf{B}}^{-1}$.

$$\hat{\mathbf{B}}^{-1}\mathbf{Ax} = \hat{\mathbf{B}}^{-1}\mathbf{b}$$

$$\mathbf{x}_{\hat{B}} + \hat{\mathbf{B}}^{-1}\hat{\mathbf{N}}\mathbf{x}_{\hat{N}} = \hat{\mathbf{B}}^{-1}\mathbf{b}$$

here the sets $\hat{B}$ and $\hat{N}$ are the sets of indices associated with $\hat{\mathbf{B}}$ and $\hat{\mathbf{N}}$.

Cutting plane method

$$\boxed{\mathbf{x}_{\hat{B}} + \hat{\mathbf{B}}^{-1}\hat{\mathbf{N}}\mathbf{x}_{\hat{N}} = \hat{\mathbf{B}}^{-1}\mathbf{b}}$$

Denote $\hat{a}_{ij} = (\hat{\mathbf{B}}^{-1}\hat{\mathbf{N}}_j)_i$ and $\hat{a}_{i0} = (\hat{\mathbf{B}}^{-1}\mathbf{b})_i$. Let's examine the ith row of the above linear system **whose $\hat{a}_{i0}$ is fractional**:

$$x_i + \sum_{j \in \hat{N}} \hat{a}_{ij} x_j = \hat{a}_{i0}$$

Since $x_j \geq 0$ for all j, we have

$$x_i + \sum_{j \in \hat{N}} \lfloor \hat{a}_{ij} \rfloor x_j \leq x_i + \sum_{j \in \hat{N}} \hat{a}_{ij} x_j = \hat{a}_{i0}$$

Since x_j should be integer in $\mathcal{IP}$, we have the following Gomory cut

$$x_i + \sum_{j \in \hat{N}} \lfloor \hat{a}_{ij} \rfloor x_j \leq \lfloor \hat{a}_{i0} \rfloor$$

An example to illustrate Gomory cutting plane method

$$\begin{aligned} \min: \quad & x_1 - 2x_2 \\ s.t. \quad & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \\ & x_i \in \mathbb{Z}^+, \forall i = 1, 2 \end{aligned}$$

First of all, we convert it to standard form by introducing slack **integer** decision variables x_3 and x_4:

$$\begin{aligned} \min: \quad & x_1 - 2x_2 \\ s.t. \quad & -4x_1 + 6x_2 + x_3 = 9 \\ & x_1 + x_2 + x_4 = 4 \\ & x_i \in \mathbb{Z}^+, \forall i = 1, \cdots, 4 \end{aligned}$$

An example to illustrate Gomory cutting plane method

$$\begin{aligned}[\mathcal{LP}_1]\ \min: \quad & x_1 - 2x_2 \\ s.t. \quad & -4x_1 + 6x_2 + x_3 = 9 \\ & x_1 + x_2 + x_4 = 4, x_i \geq 0, \forall i\end{aligned}$$

Iteration 1: Solve $[\mathcal{LP}_1]$. The optimal solution is $\mathbf{x} = (1.5, 2.5, 0, 0)$,

$$\mathbf{B} = \begin{bmatrix} -4 & 6 \\ 1 & 1 \end{bmatrix}, \mathbf{B}^{-1} = \begin{bmatrix} -0.1 & 0.6 \\ 0.1 & 0.4 \end{bmatrix}$$

$$\mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 1.5 \\ 2.5 \end{bmatrix}, \mathbf{B}^{-1}\mathbf{N} = \begin{bmatrix} -0.1 & 0.6 \\ 0.1 & 0.4 \end{bmatrix}$$

$$x_2 + 0.1x_3 + 0.4x_4 = 2.5$$

The Gomory cut is: $x_2 \leq 2$

An example to illustrate Gomory cutting plane method

$$\begin{aligned}
[\mathcal{LP}_2]\,\min: \quad & x_1 - 2x_2 \\
s.t. \quad & -4x_1 + 6x_2 + x_3 = 9 \\
& x_1 + x_2 + x_4 = 4 \\
& x_2 + x_5 = 2, x_i \geq 0, \forall i
\end{aligned}$$

Iteration 2: Solve $[\mathcal{LP}_2]$. The optimal solution is $\mathbf{x} = (0.75, 2, 0, 1.25, 0)$,

$$\mathbf{B} = \begin{bmatrix} -4 & 6 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \mathbf{B}^{-1} = \begin{bmatrix} -0.25 & 0 & 1.5 \\ 0 & 0 & 1 \\ 0.25 & 1 & -0.25 \end{bmatrix}$$

$$\mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} 0.75 \\ 2 \\ 1.25 \end{bmatrix}, \mathbf{B}^{-1}\mathbf{N} = \begin{bmatrix} -0.25 & 1.5 \\ 0 & 1 \\ 0.25 & -2.5 \end{bmatrix}$$

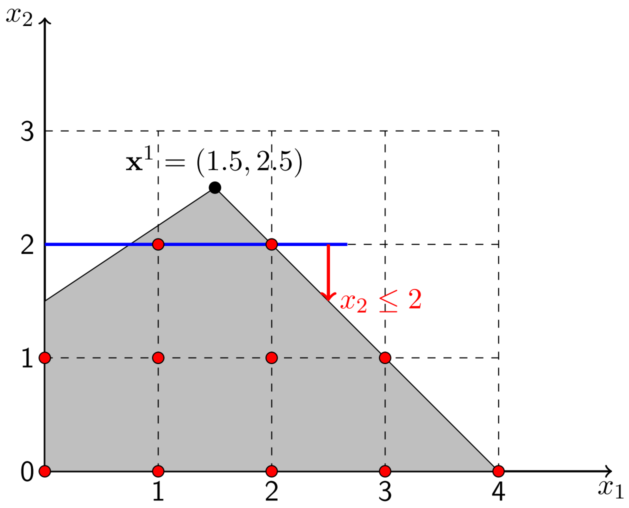
$x_1 - 0.25x_3 + 1.5x_5 = 0.75,$ *The Gomory cut is:* $x_1 - x_3 + x_5 \leq 0$

An example to illustrate Gomory cutting plane method

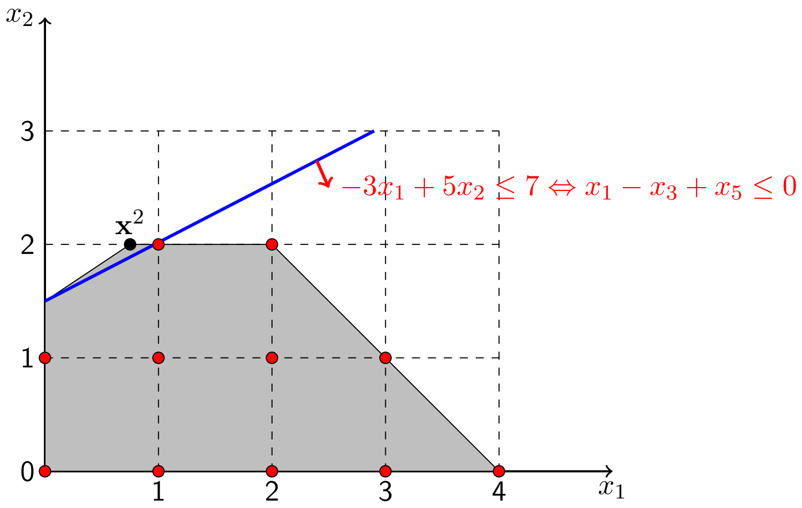
$$\begin{aligned}
[\mathcal{LP}_3]\,\min: &\quad x_1 - 2x_2 \\
s.t. &\quad -4x_1 + 6x_2 + x_3 = 9 \\
&\quad x_1 + x_2 + x_4 = 4 \\
&\quad x_2 + x_5 = 2 \\
&\quad x_1 - x_3 + x_5 + x_6 = 0 \\
&\quad x_i \geq 0, \forall i
\end{aligned}$$

Iteration 3: Solve $[\mathcal{LP}_3]$. The optimal solution is $\mathbf{x} = (1, 2, 1, 1, 0, 0)$ which is an integer vector. We can terminate the algorithm.

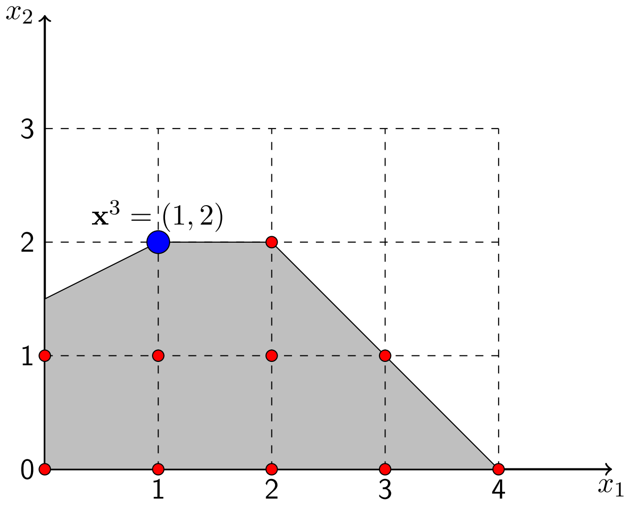
An example to illustrate Gomory cutting plane method



An example to illustrate Gomory cutting plane method

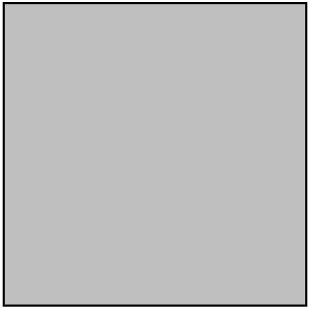

x_2
3
2
1
0
1
2
3
4
x_1
$\mathbf{x}^2$
$-3x_1 + 5x_2 \leq 7 \Leftrightarrow x_1 - x_3 + x_5 \leq 0$

An example to illustrate Gomory cutting plane method



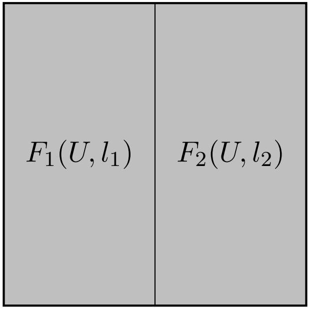
Branch and Bound

Branch and Bound is a “divide and conquer” approach to solve the integer programming problem.



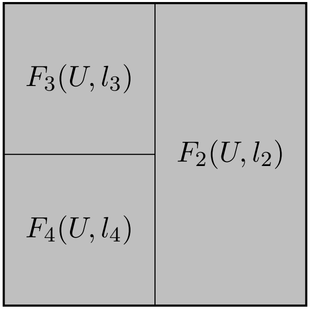
Branch and Bound

Branch and Bound is a “divide and conquer” approach to solve the integer programming problem.



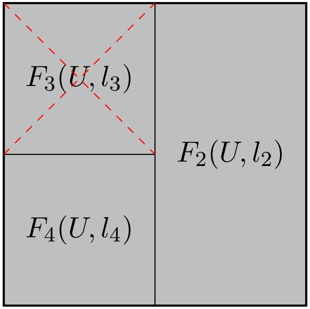
Branch and Bound

Branch and Bound is a “divide and conquer” approach to solve the integer programming problem.



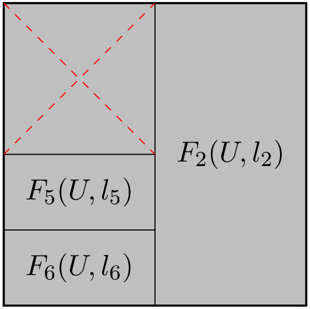
Branch and Bound

Branch and Bound is a “divide and conquer” approach to solve the integer programming problem.



Branch and Bound

Branch and Bound is a “divide and conquer” approach to solve the integer programming problem.



Branch and Bound

Essential steps for Branch and Bound method

1. Select an active subproblem F_i.
2. If the subproblem is infeasible, delete it; otherwise, compute l_i for the corresponding subproblem.
3. If $l_i \geq U$, delete the subproblem.
4. If $l_i < U$, break the corresponding subproblems into further subproblems, which are added to the list of active subproblems.

There are three key questions for the Branch and Bound method:

1. **How to obtain tight lowers bound for subproblems?**
2. **How to branch?**
3. **Which node to branch?**

How to obtain tight lowers bound for subproblems?

One way of doing it is to relax the integrity constraints of subproblems and try to solve their linear counterparts. Most of the MIP solvers adopt such a strategy. It explains why we need to use big-M method with caution. Actually, there are 3 more smart ways to improve the performance of the algorithm:

- **Branch and Cut**: after solving the linear relaxation problem to optimality, you can derive the Gomory cuts and add them to the original linear relaxation problem and resolve it.
- **Branch and Price**: if the linear relaxation problem contains a large number of decision variables, you can adopt column generation method to solve it.
- **Branch and Price and Cut**: you can also mix the previous two methods. First use column generation to solve the linear relaxation problem and then add Gomory cuts (or other valid inequalities) to the original linear relaxation problem.

How to branch?

There is no standard on how to branch. However, the common practice is: 1) after solving the linear relaxation problem, suppose we have an optimal solution $\mathbf{x}^*$, identify a component x_i^* for which $\mathbf{x}^*$ is not integer; 2) Create two subproblems by adding either of the constraints

$$\begin{aligned} x_i &\leq \lfloor x_i^* \rfloor \\ x_i &\geq \lfloor x_i^* \rfloor \end{aligned}$$

An example to illustrate Branch and Bound method

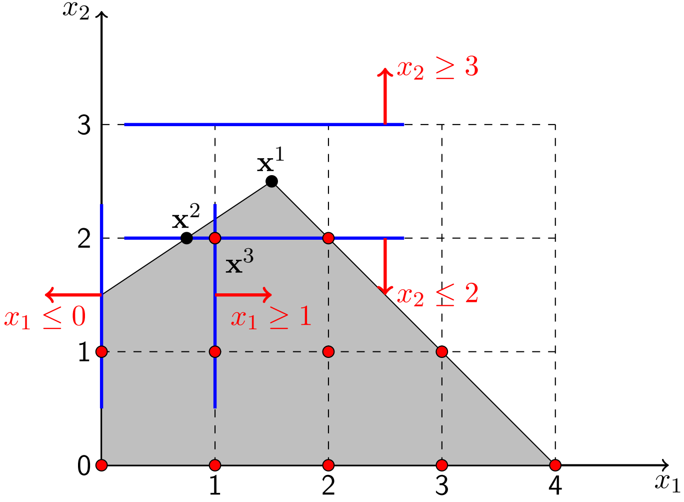
$$\begin{aligned}[\mathcal{IP}]\min: \quad & x_1 - 2x_2 \\ s.t. \quad & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \\ & x_i \in \mathbb{Z}^+, \forall i = 1, 2\end{aligned} \qquad \begin{aligned}[\mathcal{LP}]\min: \quad & x_1 - 2x_2 \\ s.t. \quad & -4x_1 + 6x_2 \leq 9 \\ & x_1 + x_2 \leq 4 \\ & x_i \geq 0, \forall i = 1, 2\end{aligned}$$

Since now we don't have any found feasible solution, let upper bound $U = +\infty$. After solving $\mathcal{LP}$, you can obtain $(x_1, x_2) = (1.5, 2.5)$ and the corresponding optimal value is -3.5.

An example to illustrate Branch and Bound method

$$\begin{tikzcd}
 & F(\frac{U}{L_1} = \frac{+\infty}{-3.5}, \frac{x_1}{x_2} = \frac{1.5}{2.5}) \arrow[dl, "x_2 \geq 3"'] \arrow[dr, "x_2 \leq 2"] & & \\
F_1(\frac{U}{L_1} = \frac{+\infty}{\times}, \frac{x_1}{x_2} = \frac{\times}{\times}) & & F_2(\frac{U}{L_1} = \frac{+\infty}{-3.25}, \frac{x_1}{x_2} = \frac{0.75}{2}) \arrow[dl, "x_1 \geq 1"'] \arrow[dr, "x_1 \leq 0"] & \\
 & F_3(\frac{U}{L_1} = \frac{-3}{-3}, \frac{x_1}{x_2} = \frac{1}{2}) & & F_4(\frac{U}{L_1} = \frac{-3}{-3}, \frac{x_1}{x_2} = \frac{0}{1.5})
\end{tikzcd}$$

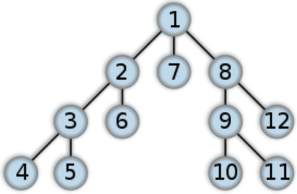
An example to illustrate Branch and Bound method

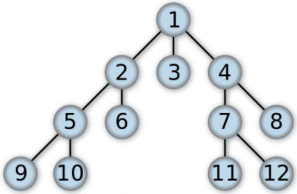


Which node to branch?

Given a set of unpruned nodes (or active nodes), the Branch and Bound method needs to decide which node to branch (i.e., the search strategy).

- **Depth first search:** Deep dive on a single node to several child generations till you reach the last leaf node
- **Breadth first search:** Search all the sibling nodes of the Branch and Bound tree first





Branch and Bound algorithm complexity

Assuming that a Branch and Bound algorithm is used on an integer programming problem with n integer decision variables. What is the maximum number of nodes (linear relaxation problems) that need to be solved? By adopting the aforementioned branching rule, in the **worse case**:

- 1 variable $\Rightarrow$ 2 LPs
- 2 variables $\Rightarrow$ 4 LPs
- 10 variables $\Rightarrow$ 1024 LPs
- $\vdots$
- n variables $\Rightarrow$ 2^n LPs

Therefore, if you branch on the "unimportant" integer decision variables or your linear relaxation problem cannot provide tight lower bound, then you will get into trouble.