

Modelling Transportation Systems

Integer Programming

Dr Zhiyuan Liu

Email: zhiyuanl@seu.edu.cn

Integer Programming (IP)

□ Learning Objectives

- Understand the application, theory, and computation of IP.
- Able to formulate the mathematical model of IP.
- Understand the procedures of Branch-and-Bound algorithm and Cutting-Plane algorithm.
- Understand the concepts of Upper Bound and Lower Bound
- Master the formulation and solution of 0-1 integer programming and assignment problem.

Formulating an IP

Five projects are being evaluated over a 3-year planning horizon. The following table gives the expected returns for each project and the associated yearly expenditures.

Project	Expenditures (million \$)/yr			Returns (million \$)
	1	2	3	
1	5	1	8	20
2	4	7	10	40
3	3	9	2	20
4	7	4	1	15
5	8	6	10	30
Available funds (million \$)	25	25	25	

Which projects should be selected over the 3-year horizon?

Formulating an IP (con't)

The problem reduces to a “*yes-no*” decision for each project.

Define the binary variable x_j

$$x_j = \begin{cases} 1, & \text{if project } j \text{ is selected} \\ 0, & \text{if project } j \text{ is not selected} \end{cases}$$

The IP model is

$$\text{Max } z = 20x_1 + 40x_2 + 20x_3 + 15x_4 + 30x_5$$

$$5x_1 + 4x_2 + 3x_3 + 7x_4 + 8x_5 \leq 25$$

$$x_1 + 7x_2 + 9x_3 + 4x_4 + 6x_5 \leq 25$$

$$8x_1 + 10x_2 + 2x_3 + x_4 + 10x_5 \leq 25$$

$$x_1, x_2, x_3, x_4, x_5 = 0 \text{ or } 1$$

Formulating an IP (con't)

The optimal integer solution is $x_1 = x_2 = x_3 = x_4 = 1, x_5 = 0$
with $z = 95$ (million \$)

This solution shows that all but project 5 must be selected.

□ **Remarks:** Comparison between **LP solution** and **IP solution**

LP optimum: replace $x_j = (0,1)$ with $0 \leq x_j \leq 1$ for all j

$x_1 = 0.5789, x_2 = x_3 = x_4 = 1, x_5 = 0.7368$, and $z = 108.68$ (million \$)

The solution is meaningless because two of the variables assume fractional values. We may **round** the solution to the closest integer values, which yields $x_1 = x_5 = 1$. However, the resulting solution is infeasible because the constraints are violated. More important, the concept of **rounding** is meaningless here because x_j represents a “yes-no” decision.

Classifications of IP Models

Pure IP Model: Where all variables must take integer values.

$$\begin{array}{ll} \text{Maximize} & z = 3x_1 + 2x_2 \\ \text{subject to} & x_1 + x_2 \leq 6 \\ & x_1, x_2 \geq 0, x_1 \text{ and } x_2 \text{ are integers} \end{array}$$

Mixed IP Model: Where some variables must be integer while others can take real values. (the following one is a mixed linear integer program)

$$\begin{array}{ll} \text{Maximize} & z = 3x_1 + 2x_2 \\ \text{subject to} & x_1 + x_2 \leq 6 \\ & x_1, x_2 \geq 0, x_1 \text{ integer} \end{array}$$

0-1 IP Model: Where all variables must take values 0 or 1 .

$$\begin{array}{ll} \text{Maximize} & z = x_1 - x_2 \\ \text{subject to} & x_1 + 2x_2 \leq 2 \\ & 2x_1 - x_2 \leq 1 \quad x_1, x_2 = 0 \text{ or } 1 \end{array}$$

Model Formulation Techniques: Linearization

The big-M method

❑ Complex logical constraints

❑ In an optimization model, the relation between the constraints is “AND”, which means all of them must be satisfied. Binary variables can formulate more complex problems to be linear constraints.

❑ Example:

❑ Consider three candidate constraints

$$\begin{cases} x + y \leq 2 \\ 2x - y \leq 3 \\ -3x + 4y \leq 4 \end{cases}$$

❑ Requirement: at least one of them must hold.

The big-M method

□ Define M as a large positive number, for example, $M=10^{10}$ (note that M is a constant rather a decision variable), define three binary decision variables z_1, z_2, z_3 , and the requirement can be formulated as:

$$\begin{cases} x + y - 2 \leq Mz_1 \\ 2x - y - 3 \leq Mz_2 \\ -3x + 4y - 4 \leq Mz_3 \\ z_1 + z_2 + z_3 \leq 2 \\ z_1, z_2, z_3 \in \{0, 1\} \end{cases}$$

Reformulation-Linearization Technique (RLT)

y is binary, x is continuous, $\underline{x} \leq x \leq \bar{x}$

$$z = x \cdot y \quad \longleftrightarrow \quad \begin{cases} z - y\underline{x} \geq 0 \\ z - y\bar{x} \leq 0 \\ z - x + \underline{x} - y\underline{x} \leq 0 \\ z - x + \bar{x} - y\bar{x} \geq 0 \end{cases}$$

□ Reference:

Sherali, H. D., & Alameddine, A. (1992). A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization*, 2(4), 379-410.

Linear transformation to linearize a product of binary variables

z_1 and z_2 are binary variables

$$z = z_1 \cdot z_2 \quad \longleftrightarrow \quad \begin{cases} z \leq z_1 \\ z \leq z_2 \\ z \geq z_1 + z_2 - 1 \\ z \geq 0 \end{cases}$$

$$z \geq z_1 \cdot z_2 \quad \longleftrightarrow \quad \begin{cases} z \geq z_1 + z_2 - 1 \\ z \geq 0 \end{cases}$$

Linear transformation to linearize a product of binary variables

y is a binary variable, x is continuous, $0 \leq x \leq M$

$$z = y \cdot x \quad \longleftrightarrow \quad \begin{cases} z \leq x \\ z \leq My \\ z \geq x - M(1 - y) \\ z \geq 0 \end{cases}$$

$$z \geq y \cdot x \quad \longleftrightarrow \quad \begin{cases} z \geq x - M(1 - y) \\ z \geq 0 \end{cases}$$

$$z \leq y \cdot x \quad \longleftrightarrow \quad \begin{cases} z \leq x \\ z \leq My \end{cases}$$

Integer Programming Model

- ❑ The **integer linear programming model** is a natural extension to the linear programming model, which is the same as the linear optimization models except that the decision variables can only take integer values.
- ❑ We also have **mixed-integer linear programming models**, in which some decision variables can only take integer values and the others are continuous.
- ❑ We use “IP models” to refer to integer linear programming models or both integer linear programming models and mixed-integer linear programming models.
- ❑ However, one should keep in mind that integer linear programming models are **not linear**; in other words, integer linear programming models belong to the category of **nonlinear programming model (NPM)**.

The relationship between LP and IP

□ Example

$$\begin{aligned} \text{IP:} \quad & \max Z = x_1 + x_2 \\ & \begin{cases} 14x_1 + 9x_2 \leq 51 \\ -6x_1 + 3x_2 \leq 1 \\ x_1, x_2 \geq 0 \text{ and } x_1, x_2 \text{ are integers} \end{cases} \end{aligned}$$

LP relaxation*:

$$\begin{aligned} & \max Z = x_1 + x_2 \\ & \begin{cases} 14x_1 + 9x_2 \leq 51 \\ -6x_1 + 3x_2 \leq 1 \\ x_1, x_2 \geq 0 \end{cases} \end{aligned}$$

*数学建模中的“松弛”指弱化某个（某些）约束条件。这样就增大了可行域。

求解一个“松弛”了的问题，能给出一个maximization原问题的upper bound; 或一个minimization原问题的lower bound. 这里给了UB和LB的一个初步认识。

The relationship between LP and IP (con't)

LP optimum:

$$x_1 = 3/2, x_2 = 10/3$$

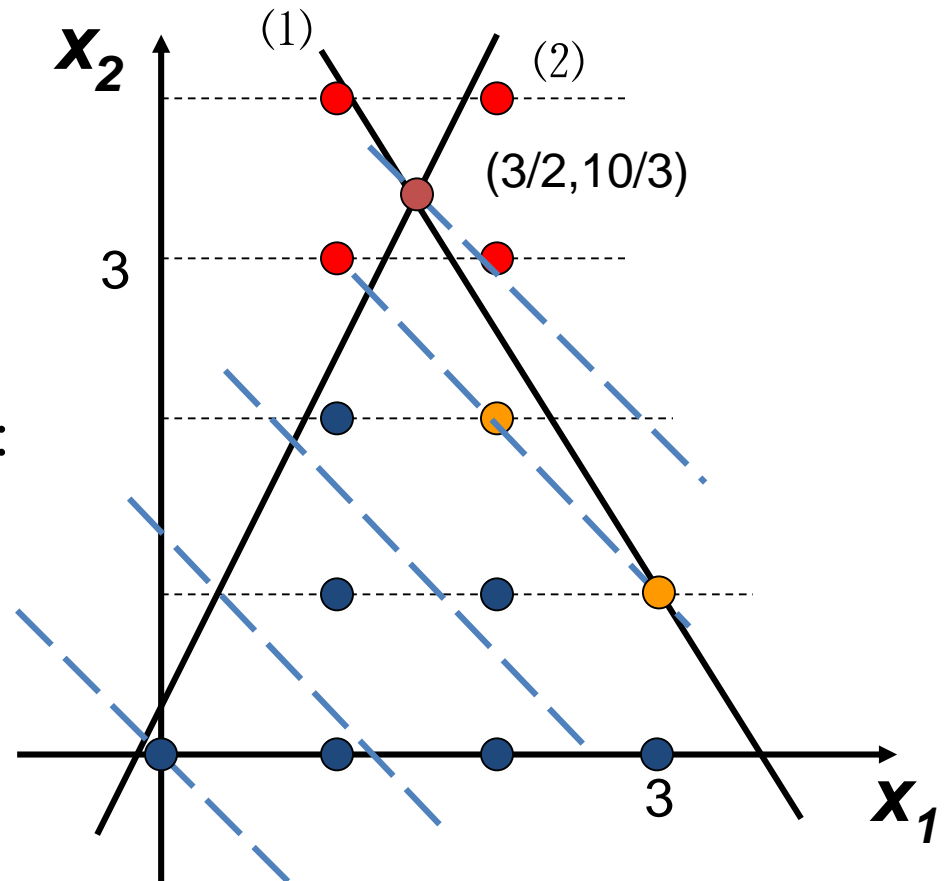
$$z = 29/6$$

Integer solution (by *rounding*):

$$(1,3), (2,3)$$

$$(1,4), (2,4)$$

All of them are not optimum!



The solution of IP is lying in the **feasible region of LP** and it is a **integer point**, which means that the set of IP feasible solution is **finite**.

The relationship between LP and IP (con't)

Approach 1:

- Enumerate all possible solutions
- Determine their objective function values
- Select the solution with the maximum (or, minimum) value.

Any potential difficulty with this approach?

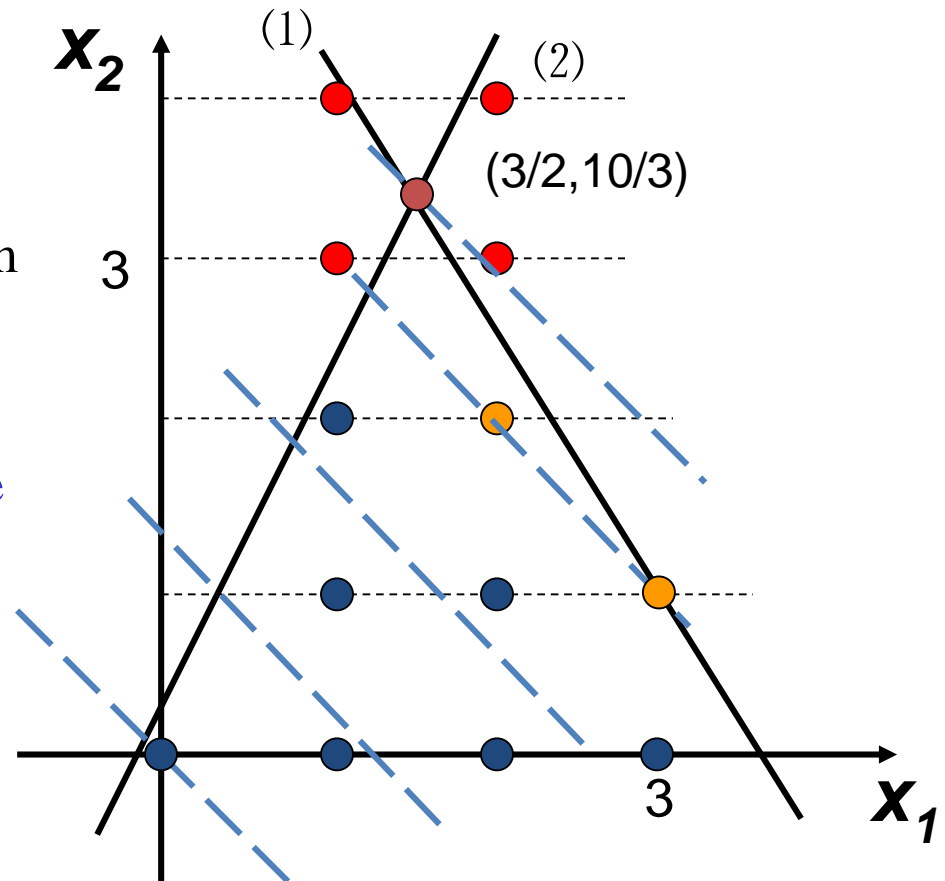
-- may be time-consuming or impossible to do

Approach 2:

- Solve the LP relaxation
- Round-off the solution to the nearest feasible integer solution

Any potential difficulty with this approach?

-- may not be optimal solution to the original IP



Other approaches:

Branch-and-bound Algorithm
Cutting-Plane Algorithm

Solution Algorithms for IPs

- ❑ Most algorithms for solving integer optimization models are based on repeatedly solving linear programming relaxations that are related to the integer optimization models
 - ⑩ Branch and Bound Algorithm
 - ⑩ Cutting Plane Algorithm
 - ⑩ Branch and Cut Algorithm

Branch-and-bound Algorithm

B&B Algorithm

- ❑ Update the lower bound and upper bound $\underline{Z} \leq Z^* \leq \bar{Z}$
- ❑ Branch and bound algorithms are the most popular methods for solving integer programming problems.
- ❑ They enumerate the entire solution space but only implicitly; hence they are called implicit enumeration algorithms.
- ❑ A general-purpose solution technique which must be specialized for individual IP's.
- ❑ Running time grows exponentially with the problem size, but small to moderate size problems can be solved in reasonable time.

Example:

$$\begin{array}{ll} \text{IP:} & \max Z = x_1 + 5x_2 \\ & \left\{ \begin{array}{l} x_1 - x_2 \geq -2 \\ 5x_1 + 6x_2 \leq 30 \\ x_1 \leq 4 \\ x_1, x_2 \geq 0 \text{ and } x_1, x_2 \text{ are integers} \end{array} \right. \end{array}$$

$$\begin{array}{ll} \text{LP:} & \max Z = x_1 + 5x_2 \\ & \left\{ \begin{array}{l} x_1 - x_2 \geq -2 \\ 5x_1 + 6x_2 \leq 30 \\ x_1 \leq 4 \\ x_1, x_2 \geq 0 \end{array} \right. \end{array}$$

Iteration 1

(LP1) optimal solution:

$$x_1=1.64, x_2=3.64$$

$$Z^{(1)} = 218/11 \approx 19.8$$

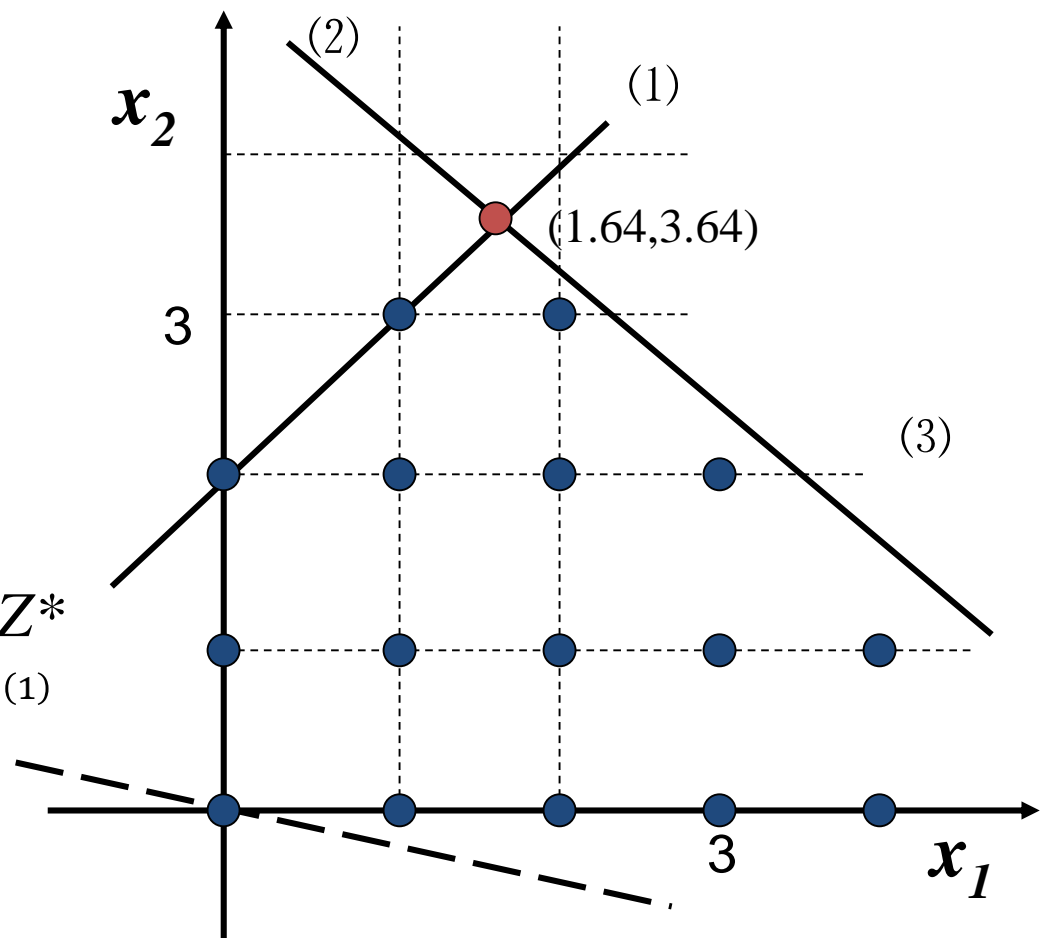
□ Bounding:

Update the **upper bound** of Z^* as current optimal solution $z^{(1)}$

$$\bar{Z} = Z^{(1)} = 19.8$$

Clearly, $x_1=0, x_2=0$ is a feasible solution, while $z=0$.

Thus, $z=0$ is taken as a lower-bound of Z^* , so $\underline{Z} = 0$

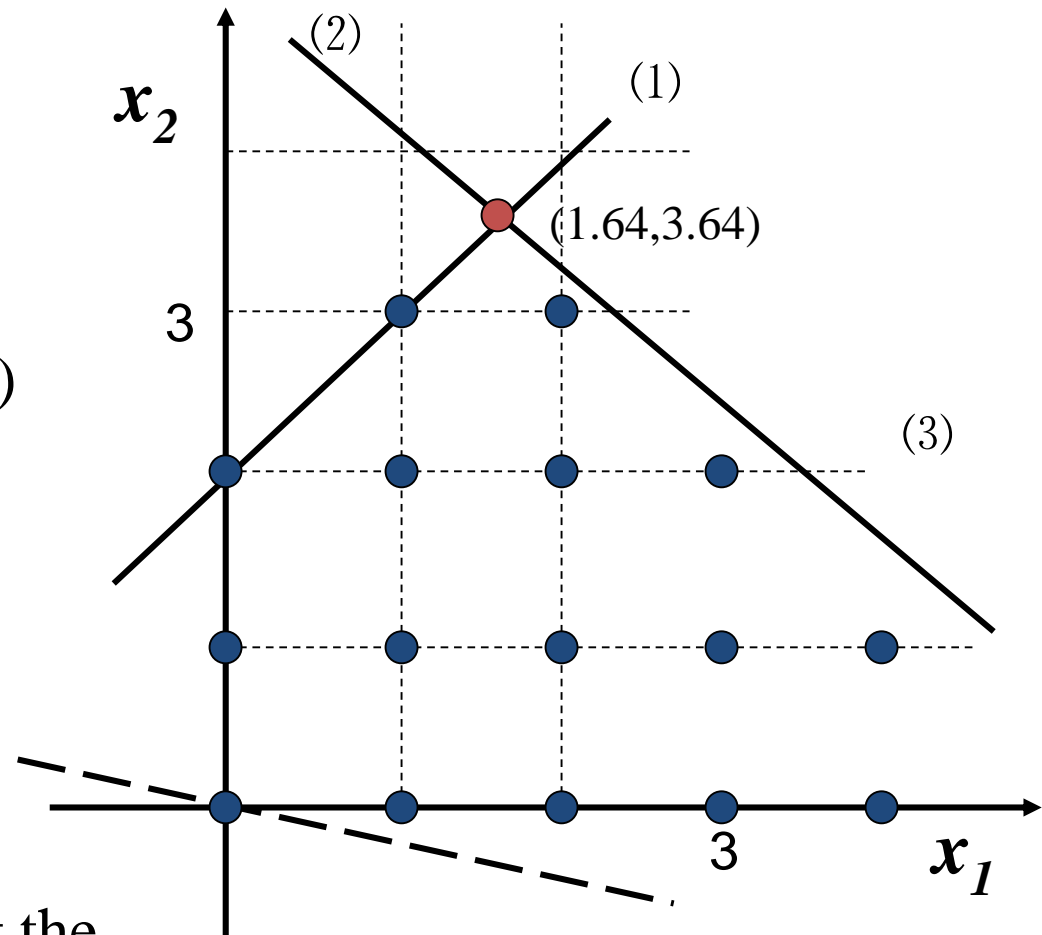


In this iteration, $\underline{Z} \neq \bar{Z}$

Iteration 1

□ Branching:

Select x_1 **arbitrarily**, the region $1 < x_1 < 2$ of the (LP1) solution space contains no integer values of x_1 , and thus can be **pruned**.



*It is equivalent to replacing the original (LP1) with two new LPs:

Iteration 2

(LP2) optimal solution **(Point B)** x_2

$$x_1=1, x_2=3$$

$$Z^{(2)}=16$$

The integer solution is found.

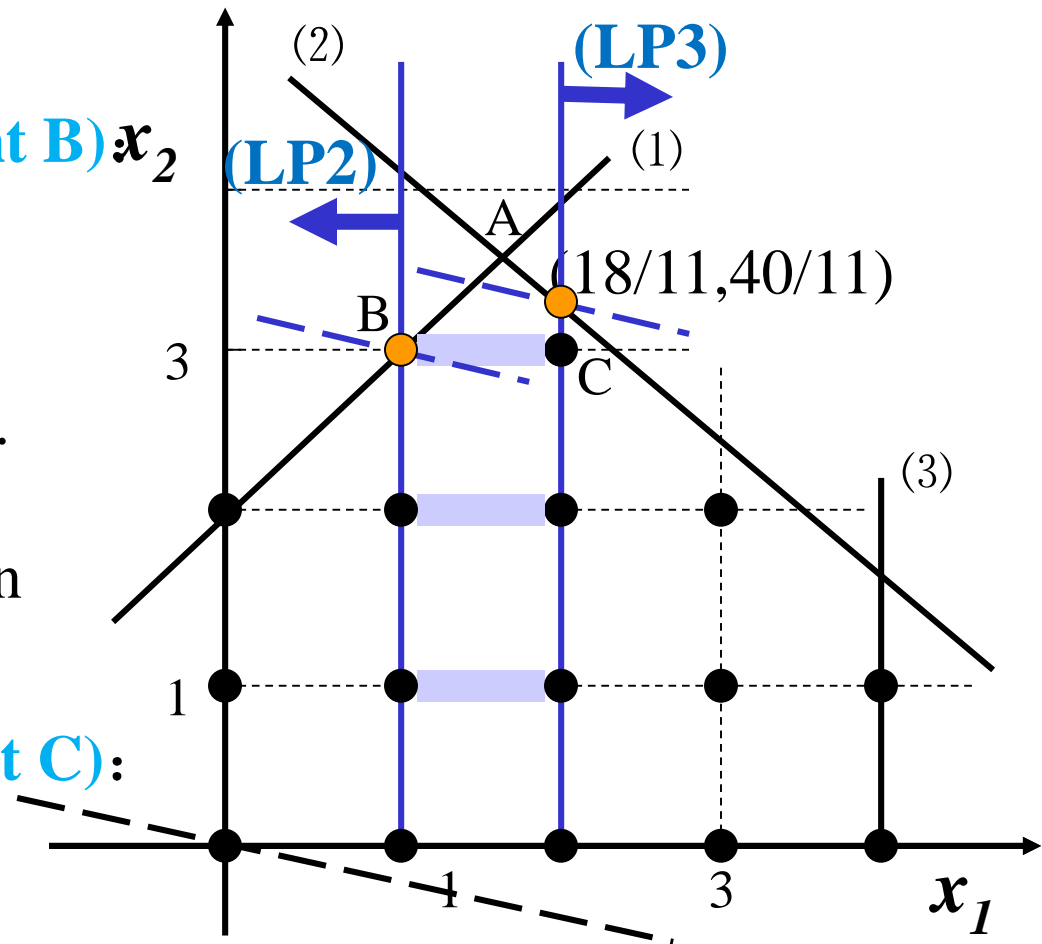
LP2 is said to be *fathomed*.

The fathomed region can then be pruned.

(LP3) optimal solution **(Point C)**:

$$x_1=2, x_2=10/3$$

$$Z^{(3)}=56/3 \approx 18.7$$



Iteration 2 (con't)

□ Bounding:

$$\because Z^{(3)} \geq Z^{(2)} = 16$$

\therefore A solution **better than** $Z^{(2)}$ exists in LP3 space.

The upper-bound of Z^* is updated:

$$\bar{Z} = 18.7$$

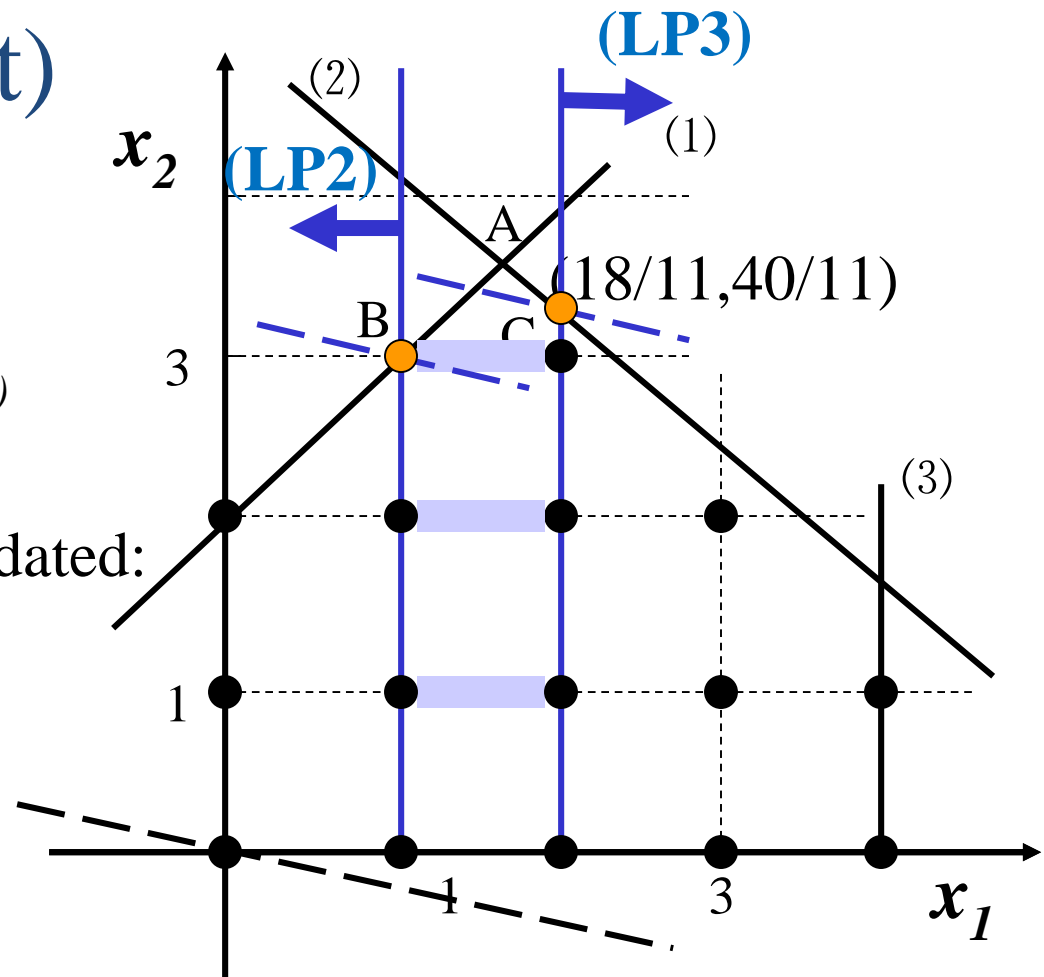
$Z^{(2)} = 16$ is termed as the **lower bound** of Z , $\underline{Z} = 16$

$$\text{Thus, } 16 \leq Z^* \leq 18.7$$

□ Branching:

LP4 space = LP3 space + $(x_2 \leq 3)$

LP5 space = LP3 space + $(x_2 \geq 4)$



Iteration 3

(LP5) has no feasible solution, and hence it is *fathomed*.

(LP4) optimal solution **(Point D)**:

$$x_1 = 12/5 \approx 2.4, x_2 = 3$$

$$Z^{(3)} = 87/5 \approx 17.4$$

□ Bounding:

The upper-bound of Z^* is updated:

$$\bar{Z} = 17.4$$

The lower-bound of Z^* is unchanged

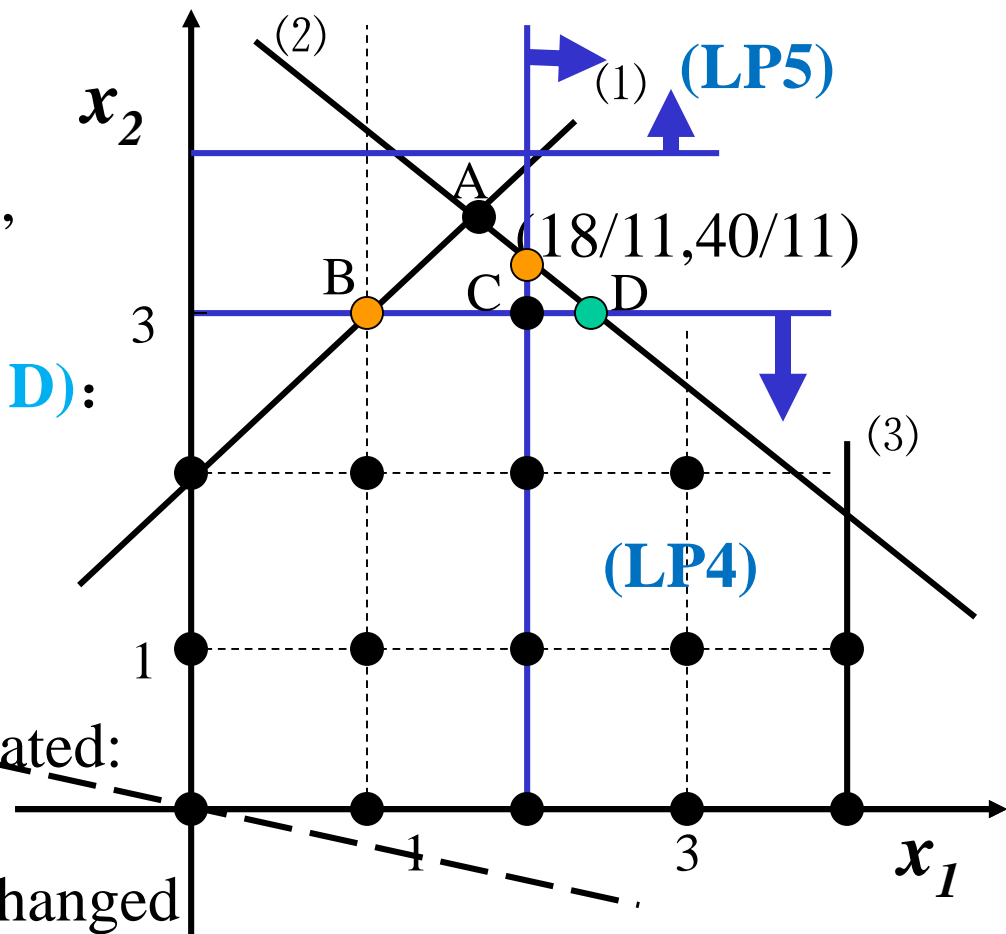
$$\underline{Z} = 16$$

$$\text{Thus, } 16 \leq Z^* \leq 17.4$$

□ Branching:

LP6 space = LP4 space + $(x_1 \leq 2)$

LP7 space = LP4 space + $(x_1 \geq 3)$



Iteration 4

(LP7) optimal solution **(Point F)**:

$$x_1=2, x_2=2.5$$

$$Z^{(7)} = 15.5 < \underline{Z} = 16$$

LP7 is *fathomed*.

(LP6) optimal solution **(Point E)**:

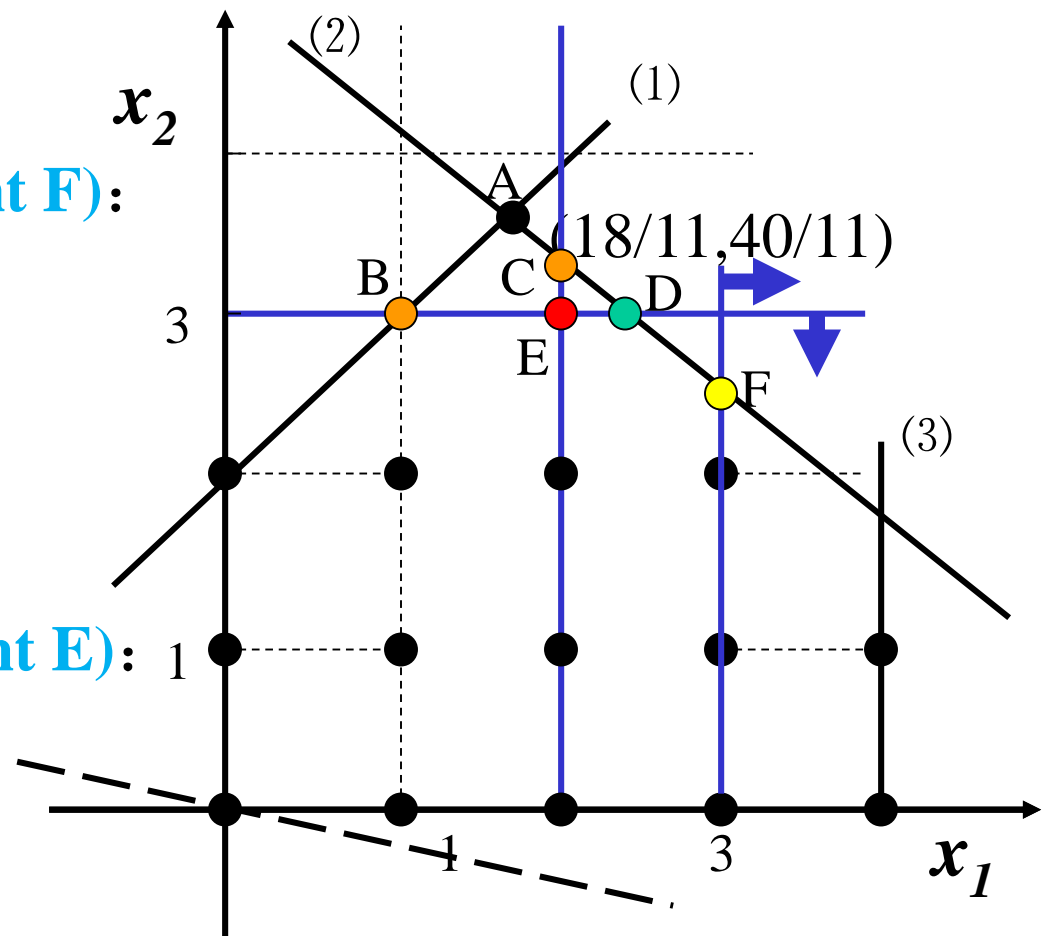
$$x_1=2, x_2=3$$

$$Z^{(6)} = 17$$

□ Bounding:

The two bounds of Z^* is updated:

$$\underline{Z} = 17 \quad \bar{Z} = 17 \quad \text{Thus, } 17 \leq Z^* \leq 17$$

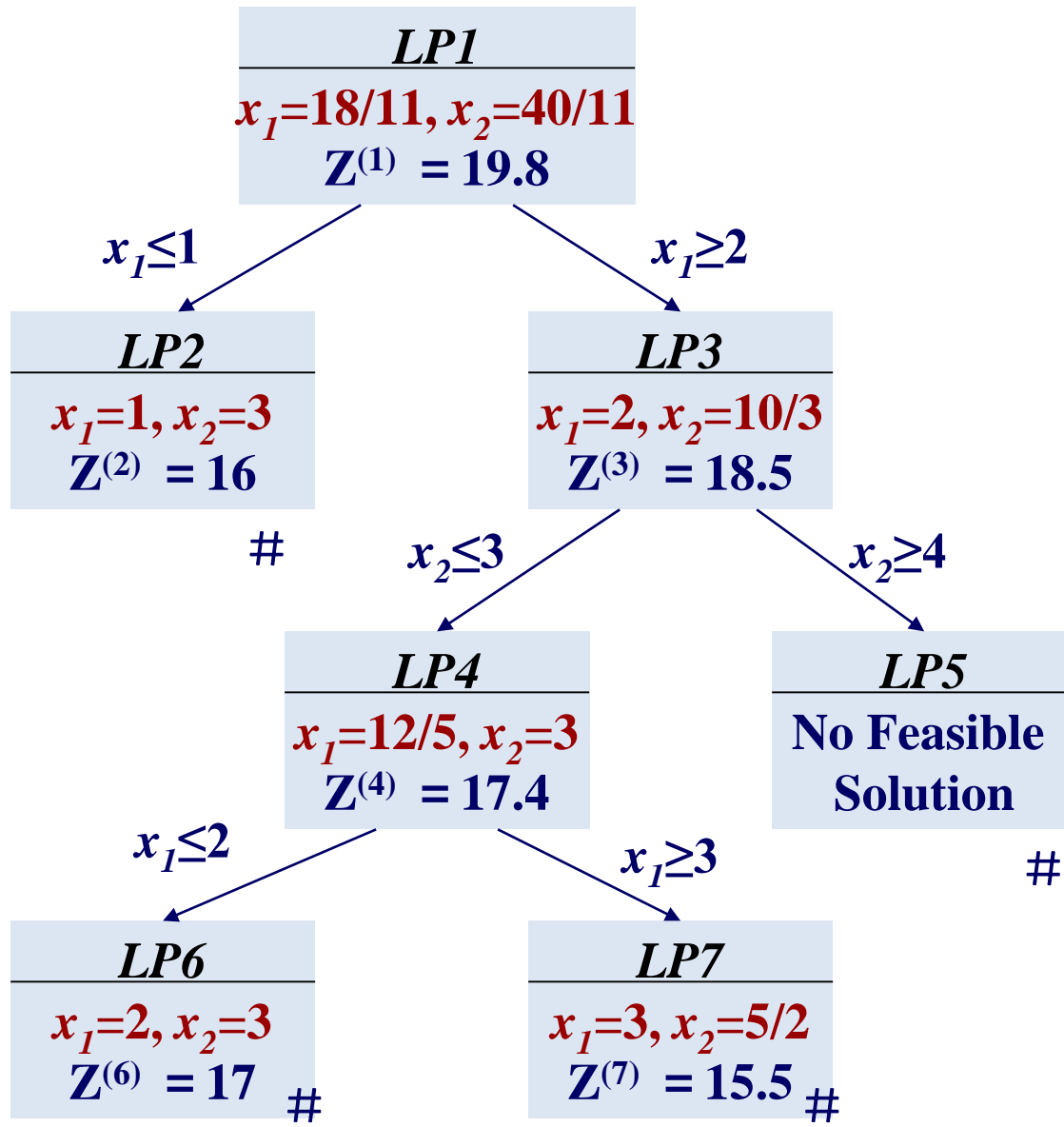


Then the optimal solution of the IP is: $x_1=2, x_2=3$

$$Z^* = \underline{Z} = \bar{Z} = 17$$

B&B tree

Lower and upper bound



$$\underline{Z} = 0, \bar{Z} = 19.8$$

$$\underline{Z} = 16, \bar{Z} = 18.5$$

$$\underline{Z} = 16, \bar{Z} = 17.4$$

$$Z^* = \underline{Z} = \bar{Z} = 17$$

Comments on B&B Algorithm

- ❑ The branch and bound algorithm is a divide and conquer algorithm, where a problem is divided into smaller and various subproblems. Each subproblem is solved separately, and their best solutions are obtained.
- ❑ Lower Bound (LB): Objective function value of the best integer solution found so far.
- ❑ Upper Bound: The optimal (maximal in the example) objective function value of all the subproblems in the current iteration.
- ❑ Branching Strategy : The process of decomposing a subproblem into two or more subproblems is called branching.

Comments on B&B Algorithm (con't)

- ❑ Pruning Strategy: If for a subproblem, it's optimal solution \leq LB, then the subproblem need not be explored further (cutted).
- ❑ Searching Strategy: The order of examining current subproblems. Popular search strategies: LIFO and FIFO.
- ❑ B&B is an exact algorithm, which is much efficient than the brute-force enumeration. (why?)
- ❑ ε -optimal solution. Take a compromised stop criterion

$$Z^* = \underline{Z} \geq \bar{Z}(1 - \varepsilon)$$

B&B Algorithm in Nonlinear IP

□ Mixed-integer nonlinear programming problem (MINLP)

$$\text{MINLP: } \min f(x, y)$$

$$\text{s.t. } g_i(x, y) \leq 0, \quad i = 1, \dots, q$$

$$h_i(x, y) = 0, \quad i = 1, \dots, l$$

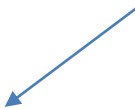
$$x \in X \subseteq \mathbf{R}^n, y \in Y \subseteq \mathbf{Z}^m$$

- The difficulty of developing an efficient method for MINLP lies not only the nonlinearity of the functions involved, but also on the simultaneous presence of both discrete and continuous variables.

B&B Algorithm in Nonlinear IP

□ A MINLP example

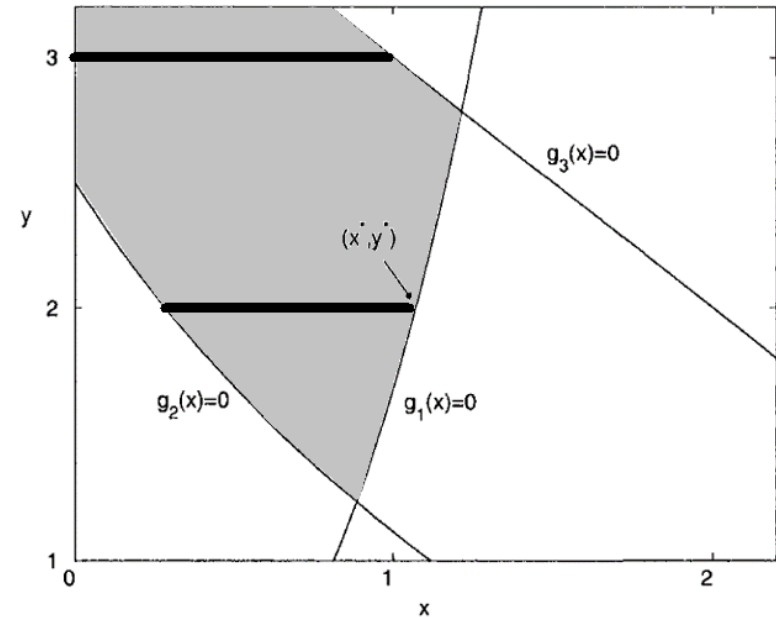
Convex function


$$\begin{aligned} \min \quad & f(x, y) = 5y - 2\ln(x+1) \\ \text{s.t.} \quad & g_1(x, y) = e^{x/2} - (1/2)\sqrt{y} - 1 \leq 0 \\ & g_2(x, y) = -2\ln(x+1) - y + 2.5 \leq 0 \\ & g_3(x, y) = x + y - 4 \leq 0 \\ & x \in [0, 2], \quad y \in [1, 3] \text{ integer.} \end{aligned}$$

B&B Algorithm in Nonlinear IP

□ Solution methods for MINLP

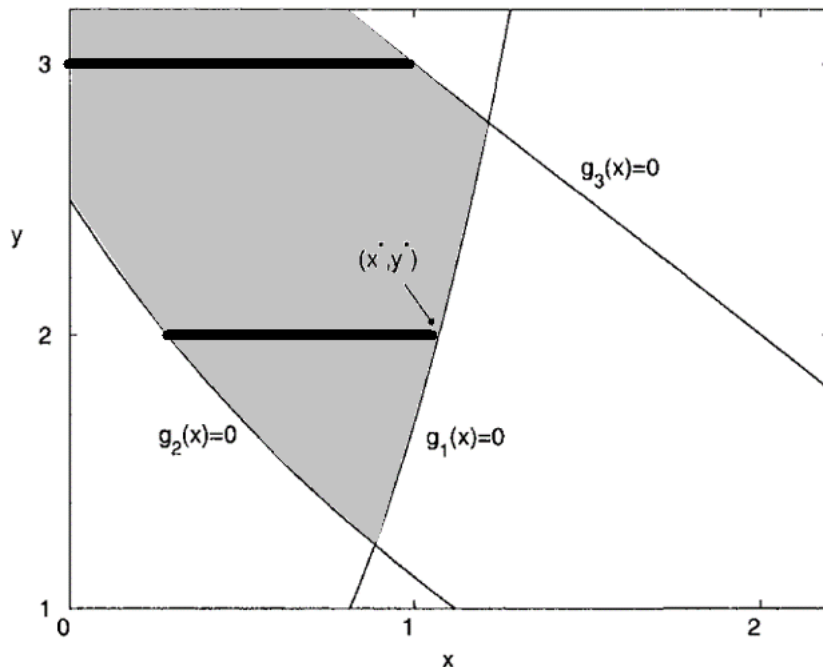
- Feasible region of the MINLP



As we can see from this example, the feasible region of this MINLP is non-connected. A simple way to overcome this difficulty is to fix or to relax the integrality of the discrete variables so as to obtain a continuous relaxation of the MINLP with a **convex feasible region**. This strategy turns out to be one of the basic strategies in various solution methods for solving MINLP.

B&B Algorithm in Nonlinear IP

□ Solution methods for MINLP



Another basic idea underlying the solution methods for MINLP is to separate the nonlinearity from the mixed-integer model so that the primal problem can be reduced to relatively easier subproblems that can be solved by existing solution methods.

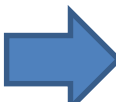
Algorithms based on the above solution strategies include:

- ✓ branch-and-bound (BB) method;
- ✓ generalized Benders decomposition (GBD) method;
- ✓ outer approximation (OA) method.

B&B Algorithm in Nonlinear IP

□ B&B algorithm in MINLP

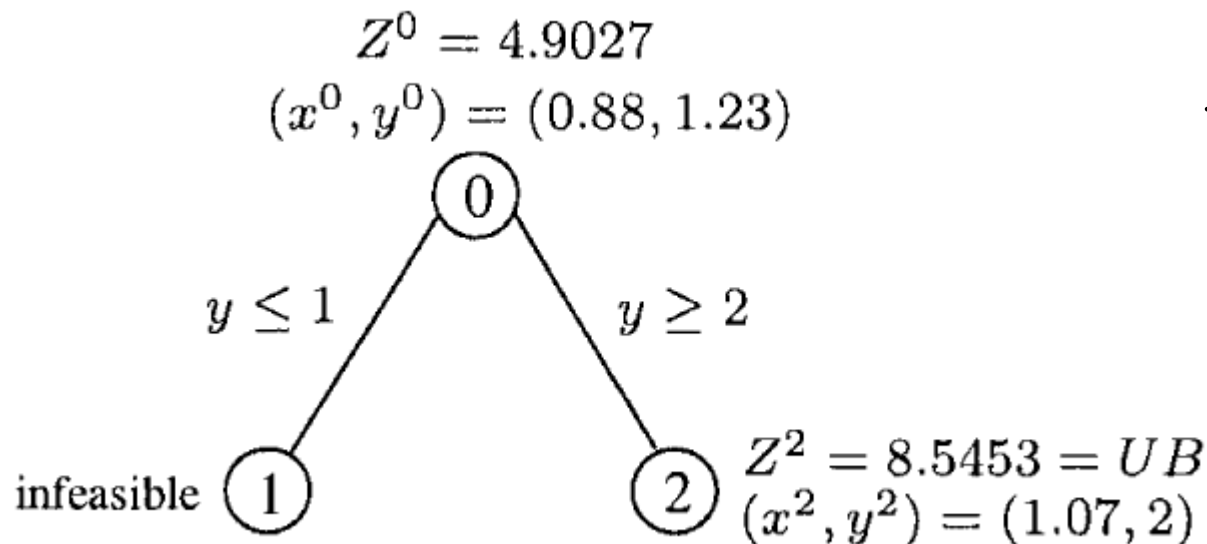
The B&B method for MINLP problem is based on the continuous relaxation of MINLP. By relaxing the integrality of variable y , we obtain the following nonlinear programming problem:

(MINLP)	$\min f(x, y)$		(NLP)	$\min f(x, y)$
s.t.	$g_i(x, y) \leq 0, i = 1, \dots, q$		s.t.	$g_i(x, y) \leq 0, i = 1, \dots, q$
	$h_i(x, y) = 0, i = 1, \dots, l$			$h_i(x, y) = 0, i = 1, \dots, l$
	$x \in X \subseteq \mathbf{R}^n, y \in Y \subseteq \mathbf{Z}^m$			$x \in X \subseteq \mathbf{R}^n, y \in Y \subseteq \mathbf{R}^m$

B&B Algorithm in Nonlinear IP

□ B&B algorithm in MINLP

The search tree is:



(NLP)

$$\min f(x, y)$$

$$\text{s.t. } g_i(x, y) \leq 0, i = 1, \dots, q$$

$$h_i(x, y) = 0, i = 1, \dots, l$$

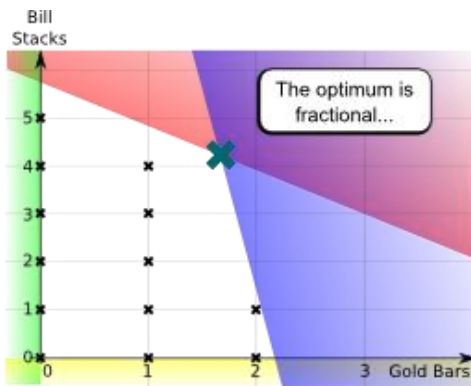
$$x \in X \subseteq \mathbf{R}^n, y \in Y \subseteq \mathbf{R}^m$$

Convex Programming
(solved by Frank-Wolfe)

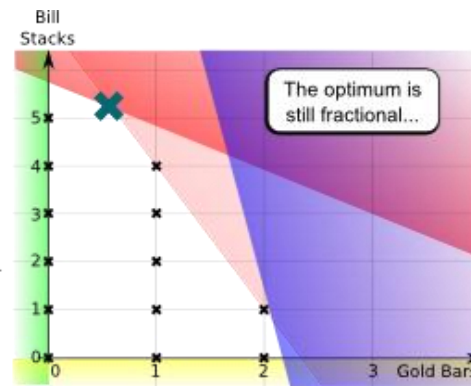
Cutting-Plane Algorithm

Cutting-Plane Algorithm

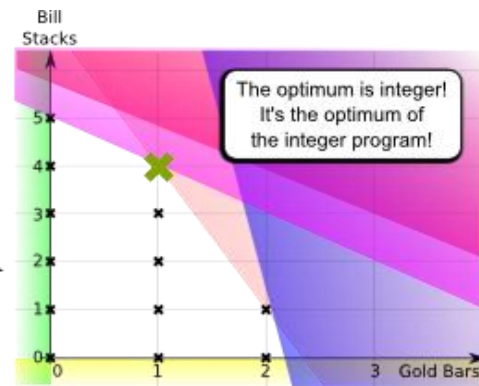
- ❑ The targeted problem in this section is Integer Linear Program, rather than mixed ones.
- ❑ Same as the B&B algorithm, the cutting-plane algorithm also starts at the continuous optimum LP solution.
- ❑ Special constraints (called **cuts**) are added to the solution space in a manner that renders an integer optimum extreme point.



Cut the
Fractional
Optimum



Cut the
Fractional
Optimum



Example

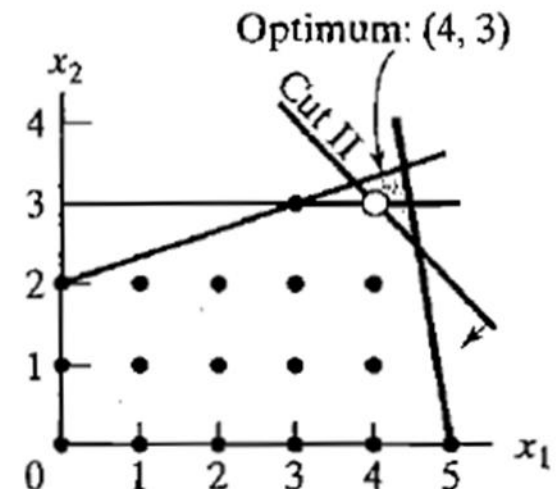
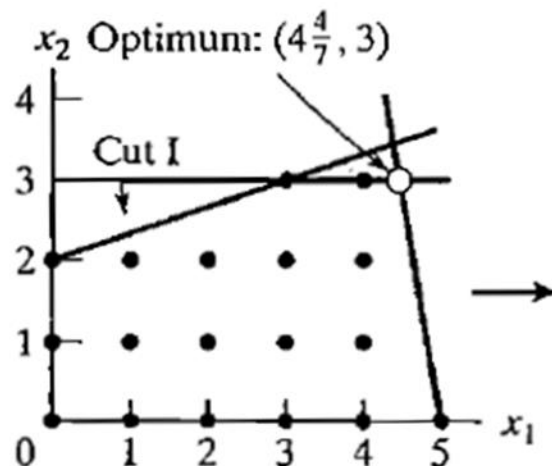
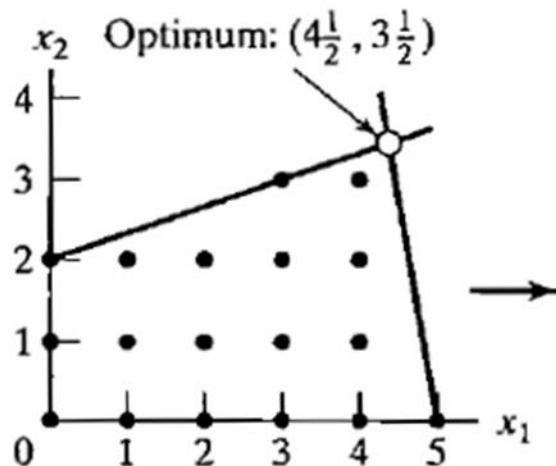
IP: $\max Z = 7x_1 + 10x_2$

$$\begin{cases} -x_1 - 3x_2 \geq -2 \\ 7x_1 + x_2 \leq 35 \\ x_1, x_2 \geq 0 \text{ and integer} \end{cases}$$

What is such a cut in n-dimension space?

Basic requirements of cutting:

- ✓ Do not eliminate any of the original feasible **integer** points.
- ✓ But must pass through at least one feasible or infeasible integer point.
- ✓ Optimal solution to the LP relaxation will be cutted.



Cutting-Plane Algorithm

➤ The optimum LP tableau

Given the slacks x_3 and x_4 for constraints 1 and 2, the optimal LP tableau is given as:

Basic	x_1	x_2	x_3	x_4	RHS
x_2	0	1	$\frac{7}{22}$	$\frac{1}{22}$	$3\frac{1}{2}$
x_1	1	0	$-\frac{1}{22}$	$\frac{3}{22}$	$4\frac{1}{2}$
$-z$	0	0	$-\frac{63}{22}$	$-\frac{31}{22}$	$66\frac{1}{2}$

The optimum of the LP relaxation is:

$$z = 66\frac{1}{2}, x_1 = 4\frac{1}{2}, x_2 = 3\frac{1}{2}, x_3 = 0, x_4 = 0$$

Note that the cut is developed under the assumption that *all* the variables are integer.

Cutting-Plane Algorithm(con't)

➤ Source row

The information in the optimum tableau can be written explicitly as

$$z + \frac{63}{22}x_3 + \frac{31}{22}x_4 = 66\frac{1}{2} \quad (z\text{-equation})$$

$$x_2 + \frac{7}{22}x_3 + \frac{1}{22}x_4 = 3\frac{1}{2} \quad (x_2\text{-equation})$$

$$x_1 - \frac{1}{22}x_3 + \frac{3}{22}x_4 = 4\frac{1}{2} \quad (x_1\text{-equation})$$

Why?



If the right-hand-side (RHS) of an equation is **fractional**, then this equation can be used as a **source row**.

- Note that the z -equation can also be used as a source row.
- Then we will demonstrate how a cut is generated from each of these source rows, starting with the z -equation.

Cutting-Plane Algorithm(con't)

➤ z-equation cutting

Factor out all the non-integer coefficients of the equation into an integer value and a fractional component:

$$(z\text{-equation}) \quad z + \frac{63}{22}x_3 + \frac{31}{22}x_4 = 66\frac{1}{2}$$



$$z + (2 + \frac{19}{22})x_3 + (1 + \frac{9}{22})x_4 = (66 + \frac{1}{2})$$

Moving all the integer components to the left-hand side and all the fractional components to the right-hand side:

$$z + 2x_3 + 1x_4 - 66 = -\frac{19}{22}x_3 - \frac{9}{22}x_4 + \frac{1}{2}$$

Cutting-Plane Algorithm(con't)


➤ z-equation cutting

$$z + 2x_3 + 1x_4 - 66 = -\frac{19}{22}x_3 - \frac{9}{22}x_4 + \frac{1}{2}$$

✓ x_3, x_4 are non-negative

Thus,
$$-\frac{19}{22}x_3 - \frac{9}{22}x_4 + \frac{1}{2} \leq \frac{1}{2}$$

Most important step
in cutting plane



✓ $z + 2x_3 + 1x_4 - 66$ equals an integer value.

Thus, the RHS of the equation must also be integer. $\leq \frac{1}{2} \Leftrightarrow \leq 0$

The inequality above can be replaced by:

$$-\frac{19}{22}x_3 - \frac{9}{22}x_4 + \frac{1}{2} \leq 0$$

x_3 and x_4 are non-basic variable in the LP relaxation, so equal to zero.

Thus, only the IP can fulfill the "cut", but not the LP relaxation.

Cutting-Plane Algorithm(con't)

➤ z-equation cutting

$$-\frac{19}{22}x_3 - \frac{9}{22}x_4 + \frac{1}{2} \leq 0$$

- This inequality is a desired cut and it represents a *necessary* (but not sufficient) condition for the optimal integer solution. It is also referred to as the **fractional cut** because all its coefficients are fractions.
- Because $x_3 = x_4 = 0$ in the optimum continuous LP tableau given before, the current continuous solution violates the above cut. Thus, if we add this cut to the optimum tableau, the resulting optimum extreme point moves the solution toward satisfying the integer requirements.

Mathematical procedure (con't)

➤ Constraints cutting

Before showing how a cut is implemented in the optimal tableau, we will demonstrate how cuts can also be constructed from the constraint equations. Consider the x_1 -row and x_2 -row:

$$x_1\text{-row: } x_1 - \frac{1}{22}x_3 + \frac{3}{22}x_4 = 4\frac{1}{2} \quad x_2\text{-row: } x_2 + \frac{7}{22}x_3 + \frac{1}{22}x_4 = 3\frac{1}{2}$$

$$x_1 + \left(-1 + \frac{21}{22}\right)x_3 + \left(0 + \frac{3}{22}\right)x_4 = \left(4 + \frac{1}{2}\right) \quad \Downarrow$$

$$-\frac{21}{22}x_2 - \frac{3}{22}x_4 + \frac{1}{2} \leq 0$$

$$x_2 + \left(0 + \frac{7}{22}\right)x_3 + \left(0 + \frac{1}{22}\right)x_4 = \left(3 + \frac{1}{2}\right) \quad \Downarrow$$

$$-\frac{7}{22}x_3 - \frac{1}{22}x_4 + \frac{1}{2} \leq 0$$

Note that any one of three cuts can be used in the first iteration of the cutting-plane algorithm. But it is not necessary to generate all three cuts before selecting one.

Cutting-Plane Algorithm(con't)

➤ Adding the constraint

Arbitrary selecting the cut generate from the x_2 -row

$$-\frac{7}{22}x_3 - \frac{1}{22}x_4 + \frac{1}{2} \leq 0 \quad \Rightarrow \quad -\frac{7}{22}x_3 - \frac{1}{22}x_4 + s_1 = -\frac{1}{2}, \quad s_1 \geq 0 \quad (\text{Cut I})$$

The constraint is added to the LP optimum tableau as follows:

Basic	x_1	x_2	x_3	x_4	s_1	RHS
x_2	0	1	$\frac{7}{22}$	$\frac{1}{22}$	0	$3\frac{1}{2}$
x_1	1	0	$-\frac{1}{22}$	$\frac{3}{22}$	0	$4\frac{1}{2}$
s_1	0	0	$-\frac{7}{22}$	$-\frac{3}{22}$	1	$-\frac{1}{2}$
$-Z$	0	0	$-\frac{63}{22}$	$-\frac{31}{22}$	0	$66\frac{1}{2}$

The solution is optimal but infeasible.

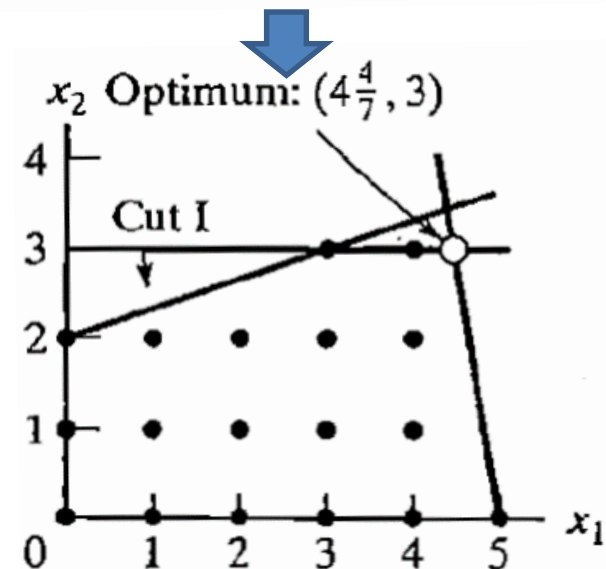
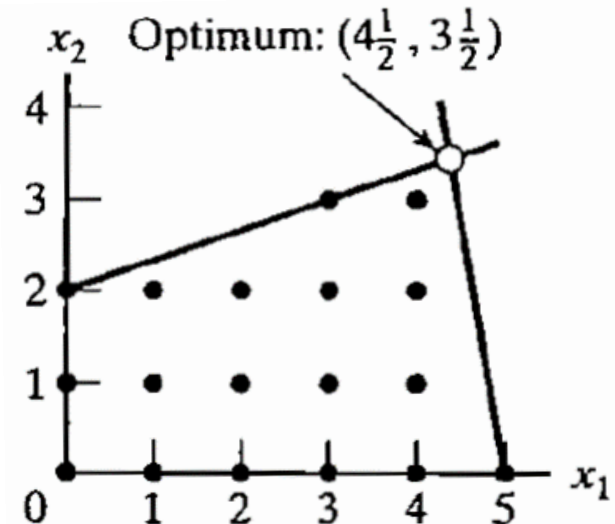
Cutting-Plane Algorithm(con't)

➤ Simplex method

The dual simplex method is applied to recover feasibility, which yields

Basic	x_1	x_2	x_3	x_4	s_1	RHS
x_2	0	1	0	0	1	3
x_1	1	0	0	$\frac{1}{7}$	$-\frac{1}{7}$	$4\frac{4}{7}$
x_3	0	0	1	$\frac{1}{7}$	$-\frac{22}{7}$	$1\frac{4}{7}$
$-z$	0	0	0	-1	-9	62

x_1 and x_3 are still not integer. Let us arbitrarily select x_1 as the next source row.



Cutting-Plane Algorithm(con't)

➤ Cutting

$$x_1 + (0 + \frac{1}{7})x_4 + (-1 + \frac{6}{7})s_1 = 4 + \frac{4}{7} \Rightarrow$$

The associated cut is

$$-\frac{1}{7}x_4 - \frac{6}{7}s_1 \leq -\frac{4}{7}$$

$$-\frac{1}{7}x_4 - \frac{6}{7}s_1 + s_2 = -\frac{4}{7}, s_2 \geq 0 \text{ (Cut II)}$$

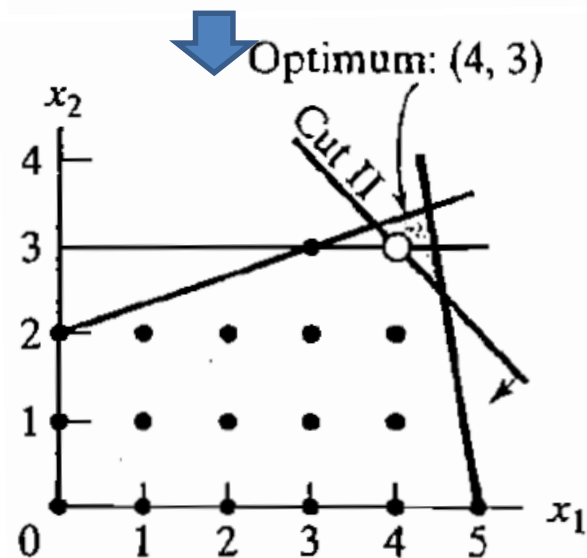
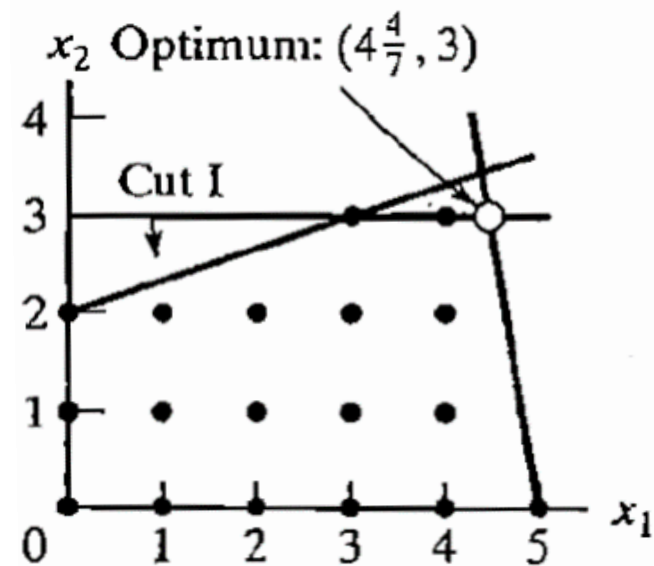
Basic	x_1	x_2	x_3	x_4	s_1	s_2	RHS
x_2	0	1	0	0	1	0	3
x_1	1	0	0	$\frac{1}{7}$	$-\frac{1}{7}$	0	$4\frac{4}{7}$
x_3	0	0	1	$\frac{1}{7}$	$-\frac{22}{7}$	0	$1\frac{4}{7}$
s_2	0	0	0	$-\frac{1}{7}$	$-\frac{6}{7}$	1	$-\frac{4}{7}$
-z	0	0	0	-1	-9	0	62

Cutting-Plane Algorithm(con't)

➤ The dual simplex method yields the following tableau:

Basic	x_1	x_2	x_3	x_4	s_1	s_2	RHS
x_2	0	1	0	0	1	0	3
x_1	1	0	0	0	-1	1	4
x_3	0	0	1	0	-4	1	1
x_4	0	0	0	1	6	-7	4
$-z$	0	0	0	-1	-9	0	58

The optimal solution is $x_1=4$, $x_2=3$, $z=58$



Branch-and-cut Algorithm

Branch-and-cut Algorithm

□ What is B&C Algorithm (分支-切割法)?

Branch and cut is a method of combinatorial optimization for solving *integer linear programs (ILPs)*, that is, linear programming (LP) problems where some or all the unknowns are restricted to integer values. Branch and cut involves running a *branch and bound algorithm* and using *cutting planes* to tighten the linear programming relaxations.

Note that if cuts are only used to tighten the initial LP relaxation, the algorithm is called *cut and branch*.

Branch-and-cut Algorithm

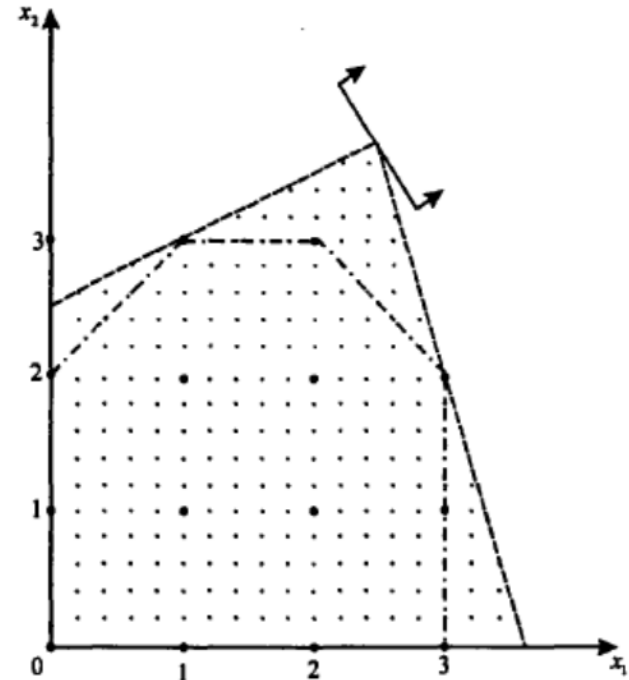
□ Description of the Algorithm

In the initial step, the *branch and bound* part of the algorithm is started. The problem is split into multiple (usually two) versions. The new linear programs are then solved using the *simplex method* and the process repeats. During the branch and bound process, non-integral solutions to **LP relaxations** serve as **upper bounds** and **integral solutions** serve as **lower bounds** (same as **B&B method**). *A subtree can be pruned if its optimal solution is lower than an existing lower bound. (Why?)* Further, when solving the LP relaxations, additional cutting planes may be generated, which may be either global cuts, i.e., valid for all feasible integer solutions, or local cuts, meaning that they are satisfied by all solutions fulfilling the side constraints from the currently considered branch and bound subtree.

Example

ILP: $\max z = 6x_1 + 5x_2$

$$\begin{cases} 3x_1 + x_2 \leq 11 \\ -x_1 + 2x_2 \leq 5 \\ x_1, x_2 \geq 0 \text{ and integer} \end{cases}$$



□ Same as the B&B method and cutting plane method, the first step is to solve the LP relaxation problem and obtain a initial optimal solution:

$$x_1 = 2\frac{3}{7}, x_2 = 3\frac{5}{7}, z = 33\frac{1}{7}$$

$$\bar{z} = z^{(1)} = 33\frac{1}{7}$$

□ Now, we have two choices:

- branching of one variable (B&B method)
- adding a cutting plane (cutting plane method)

Example

□ We now arbitrarily branch x_1 .

Subproblem 1:

$$\max z = 6x_1 + 5x_2$$

$$\begin{cases} 3x_1 + x_2 \leq 11 \\ -x_1 + 2x_2 \leq 5 \\ x_1 \geq 3 \\ x_1, x_2 \geq 0 \end{cases}$$

The optimal solution is:

$$x_1=3, x_2=2, z^{(2)} = 28$$

$$\underline{z} = z^{(2)} = 28$$

$$28 \leq z^* \leq 29.5$$

Subproblem 2:

$$\max z = 6x_1 + 5x_2$$

$$\begin{cases} 3x_1 + x_2 \leq 11 \\ -x_1 + 2x_2 \leq 5 \\ x_1 \leq 2 \\ x_1, x_2 \geq 0 \end{cases}$$

The optimal solution is:

$$x_1=2, x_2=3.5, z^{(3)} = 29.5$$

$$\bar{z} = \max(z^{(2)}, z^{(3)}) = 29.5$$

Example

□ Adding a cutting plane in subproblem 2

Subproblem 2:

$$\max z = 6x_1 + 5x_2$$

$$\begin{cases} 3x_1 + x_2 \leq 11 \\ -x_1 + 2x_2 \leq 5 \\ x_1 \leq 2 \\ x_1, x_2 \geq 0 \end{cases}$$

Final Tableau:

Basic	x_1	x_2	x_3	x_4	x_5	RHS
x_3	0	0	1	$-\frac{1}{2}$	$-\frac{7}{2}$	$\frac{3}{2}$
x_2	0	1	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{7}{2}$
x_1	1	0	0	0	1	2
$-z$	0	0	0	$-\frac{5}{2}$	$-\frac{17}{2}$	$-\frac{35}{2}$

The optimal solution is:

$$x_1=2, x_2=3.5, z^{(3)} = 29.5$$

Example

□ Adding a cutting plane in subproblem 2

□ Recall the construction of cut plane:

□ Constraint 2:

$$x_2 + \frac{1}{2}x_4 + \frac{1}{2}x_5 = \frac{7}{2} = 3 + \frac{1}{2}$$



$$x_2 - 3 = \frac{1}{2} - \frac{1}{2}x_4 - \frac{1}{2}x_5 \leq 0$$



Cutting plane: $1 - x_4 - x_5 \leq 0 \iff x_2 \leq 3$

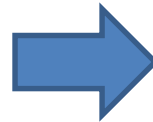
Example

□ Adding a cutting plane in subproblem 2

Subproblem 2:

$$\max z = 6x_1 + 5x_2$$

$$\begin{cases} 3x_1 + x_2 \leq 11 \\ -x_1 + 2x_2 \leq 5 \\ x_1 \leq 2 \\ x_1, x_2 \geq 0 \end{cases}$$



Subproblem 3:

$$\max z = 6x_1 + 5x_2$$

$$\begin{cases} 3x_1 + x_2 \leq 11 \\ -x_1 + 2x_2 \leq 5 \\ x_2 \leq 3 \\ x_1 \leq 2 \\ x_1, x_2 \geq 0 \end{cases}$$

The optimal solution of Subproblem 3

is:

$$x_1 = 2, x_2 = 3, z^{(4)} = 27$$

$$\bar{z} = z^{(4)} = 27$$

Example

□ Discussion

We can find that the current value of upper-bound is $\bar{z} = z^{(4)} = 27$

However, the range of the optimal solution before subproblem 3 is

$$28 \leq z^* \leq 29.5$$

That is to say, the new upper-bound obtained by the LP relaxation is lower than the existing lower-bound. The optimal integer solution of subproblem 3 would be no larger than the optimal solution of the LP relaxation. Thus, this branch is pruned.

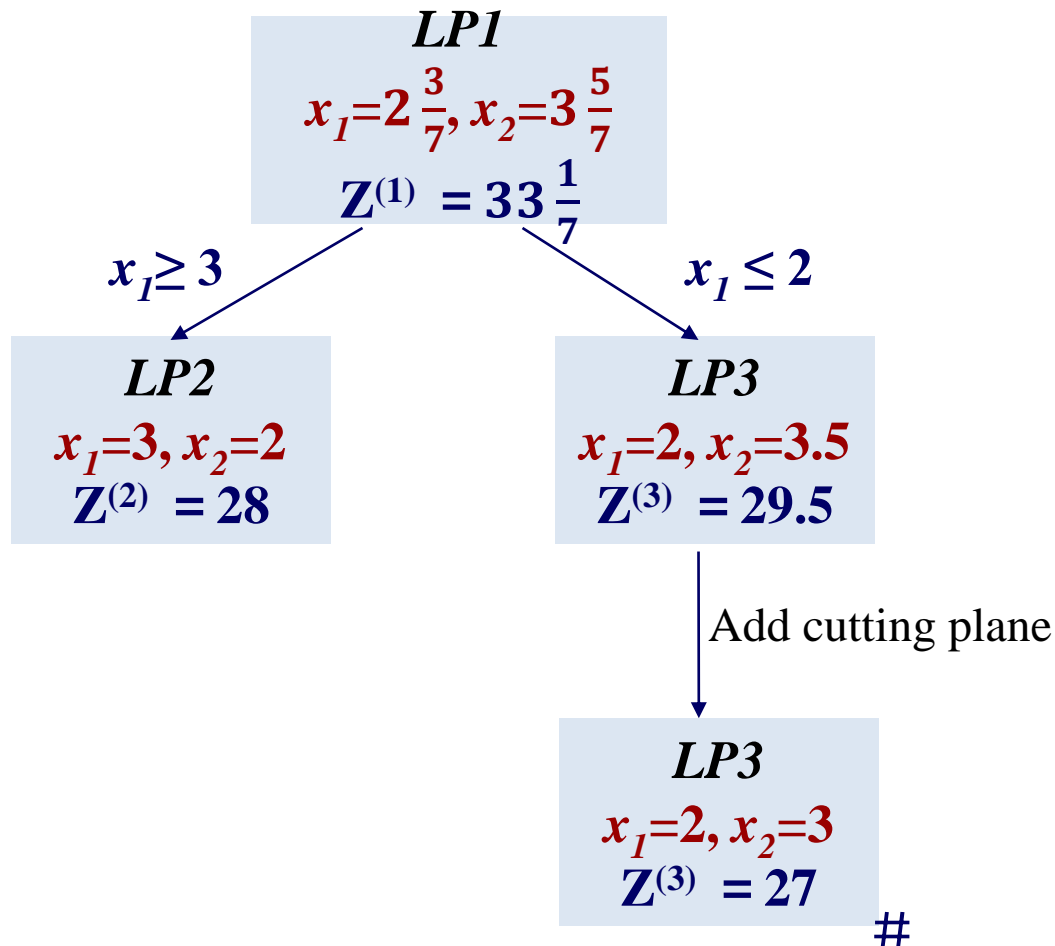
The optimal integer solution of the original ILP is

$$x_1 = 3, x_2 = 2, z^* = 28$$

Branch-and-cut Algorithm

□ B&C search tree

Lower- and upper-bound



$$\underline{z} = 0, \bar{z} = 33\frac{1}{7}$$

$$\underline{z} = 28, \bar{z} = 29.5$$

$$z^* = \underline{z} = 28$$

Summary

□ B&B algorithm

- It recursively splits the search space into smaller spaces; the splitting is called branching.
- Branching alone would amount to brute-force enumeration of candidate solution and testing them all. To improve on the performance of brute-force search, a B&B algorithm keeps track of bounds on the optimum that is trying to find, and uses these bounds to “prune” the search space, eliminating candidate solutions that it can prove will not contain an optimal solution.

□ Cutting plane algorithm

- It refines the feasible set or objective function by means of linear inequalities, termed cuts, which uses the thought of the Simplex Method to solve IPs.

□ B&C algorithm

- It can be termed as a hybrid between B&B and the cutting plane methods, which reduce the feasible region for the LP relaxation which does not reduce the feasible solution.

0-1 Integer Programming

0-1 Integer Programming

- The 0-1 integer programming is widely used to describe the decision problem, which should answer “**yes-no**” questions.
- The 0-1 variables are usually represented as follows
$$x_j = \begin{cases} 1 & x_j \text{ is yes} \\ 0 & x_j \text{ is no} \end{cases}$$
- The mathematical model of 0-1 IP is:

$$\max z = \sum_{j=1}^n c_j x_j$$

$$\begin{cases} \sum_{j=1}^n a_{ij} x_j = b_i & (i = 1, 2, \dots, m) \\ x_j = 0 \text{ or } 1 & (j = 1, 2, \dots, n) \end{cases}$$

The IP model of the shortest-path problem

- Let us consider a directed graph $G = (V, A)$ with the set of nodes V and the set of arcs A . Let n and m be the cardinality of V and A , respectively. A path is a sequence of nodes v_1, \dots, v_k , and is said to be elementary if no node appears in the path more than once. A cycle, or tour, is a path with $v_1 = v_k$. We denote by $\delta^+(i)$ and $\delta^-(i)$ the set of outgoing and incoming arcs of node i , by $\delta^+(S)$ and $\delta^-(S)$ the arcs leaving/entering the set $S \subseteq V$, and by $A(S)$ the set of arcs with both ends in $S \subseteq V$.
- Let us also define: $V_i := V \setminus \{i\}$, $V_{ij} := V \setminus \{i, j\}$, and, for any arc set $B \subseteq A$, we define $x(B) := \sum_{b \in B} x_b$.
- In all the formulations, it is assumed w.l.o.g. that

$$|\delta^-(s)| = |\delta^+(t)| = 0$$

The IP model of the shortest-path problem

- A standard integer programming formulation to determine a shortest path from node s to node t is the following:

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

Flow conservation
constraint

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t, \forall i \in V \\ 0 & \text{else} \end{cases}$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} \leq 1, \forall i \in V$$

Ensure that the
outgoing degree of
each node is at
most one.

$$x_{ij} \in \{0,1\}, \forall (i,j) \in A$$

0-1 Integer Programming

□ Conversion of a pure IP to 0-1 IP.

□ Suppose the variable x_i is an integer. Let n be the smallest integer such that we can be sure that $x_i < 2^{n+1}$. Then x_i can be (uniquely) expressed as

$$x_i = u_n 2^n + u_{n-1} 2^{n-1} + \cdots + u_2 2^2 + 2u_1 + u_0$$

where $u_i = 0$ or 1

To convert the original IP to a 0-1 IP, replace such occurrence of x_i by the right hand side of the upper equation. If $x_i \leq 100$, then $x_i < 2^{6+1} = 128$.

$$x_i = 64u_6 + 32u_5 + 16u_4 + 8u_3 + 4u_2 + 2u_1 + u_0 \leq 100$$

□ 0-1 IPs are generally easier to solve than other pure IPs. Many problem (shortest path problem) naturally yield 0-1 problem, so it's necessary to learn how to solve 0-1 IPs.

Solution Algorithms for 0-1 IP

□ Solution algorithms for 0-1 IPs

- ⑩ Complete Enumeration (枚举、穷举、遍历)
- ⑩ Implicit Enumeration (隐枚举法)
- ⑩ General algorithms for pure IPs (e.g., B&B; B&C)

Implicit Enumeration

□ Example:

$$\begin{aligned} \max z &= 4x_1 + 3x_2 + 2x_3 \\ s.t. \quad &\begin{cases} 2x_1 - 5x_2 + 3x_3 \leq 4 \\ 4x_1 + x_2 + 3x_3 \geq 3 \\ x_2 + x_3 \geq 1 \\ x_1, x_2, x_3 = 0 \text{ or } 1 \end{cases} \end{aligned}$$

□ First, get a feasible solution, say $x_1 = x_2 = 0$; $z_0 = 2$.

□ Then, take this feasible solution to the objective function, we build a filtering constraint (过滤条件) as an additional constraint

$$4x_1 + 3x_2 + 2x_3 \geq 2$$

(3) 求解过程如下表所示:

点	过滤条件	约束				z值
		④	①	②	③	
	$4x_1 + 3x_2 + 2x_3 \geq 2$					
$(0, 0, 0)^T$		×				
$(0, 0, 1)^T$		√	√	√	√	2
$(0, 1, 0)^T$		√	√	×		
$(0, 1, 1)^T$		√	√	√	√	5
	$4x_1 + 3x_2 + 2x_3 \geq 5$					
$(1, 0, 0)^T$		×				
$(1, 0, 1)^T$		√	×			
$(1, 1, 0)^T$		√	√	√	√	7
	$4x_1 + 3x_2 + 2x_3 \geq 7$					
$(1, 1, 1)^T$		√	√	√	√	9

Appendix : Formulations of Several Transport Problems

Applications

➤ 1.相互排斥的计划 (Mutually exclusive planning)

例1 某公司拟在市东、西、南三区中建立门市部，有例7个点 A_i ($i=1, 2, \dots, 7$) 可供选择，要求满足以下条件：

- 1) 在东区，在 A_1, A_2, A_3 三个点中至多选两个；
- 2) 在西区， A_4, A_5 两个点中至少选一个；
- 3) 在南区， A_6, A_7 两个点为互斥点。
- 4) 选 A_2 点必选 A_5 点。

若 A_i 点投资为 b_i 万元，每年可获利润为 c_i 万元，投资总额为 B 万元，试建立利润最大化的0—1规划模型。

➤ 1.相互排斥的计划 (Mutually exclusive planning)

设决策变量为

$$x_i = \begin{cases} 1, & \text{当} A_i \text{点被选用} \\ 0, & \text{当} A_i \text{点未被选用} \end{cases} \quad i = 1, 2, \dots, 7$$

建立0 - 1规划模型如下:

$$\max z = c_1 x_1 + c_2 x_2 + \dots + c_7 x_7$$

$$s.t \left\{ \begin{array}{l} \sum_{i=1}^7 b_i x_i \leq B \\ x_1 + x_2 + x_3 \leq 2 \\ x_4 + x_5 \geq 1 \\ x_6 + x_7 = 1 \\ x_2 - x_5 \leq 0 \\ x_i = 0 \text{或} 1, \quad i = 1, 2, \dots, 7 \end{array} \right.$$

➤ 2.互排斥的约束条件(Mutually exclusive constraints)

例2 某产品有 A_1 和 A_2 两种型号，需要经过 B_1 、 B_2 、 B_3 三道工序，单位工时和利润、各工序每周工时限制见表所示，问工厂如何安排生产，才能使总利润最大？（ B_3 工序有两种加工方式 B_{31} 和 B_{32} ，产品为整数）。

型号 \ 工序	B_1	B_2	B_3		利润 (元/件)
			B_{31}	B_{32}	
A_1	0.3	0.2	0.3	0.2	25
A_2	0.7	0.1	0.5	0.4	40
每周工时(小时/月)	250	100	150	120	

➤ 2.互排斥的约束条件(Mutually exclusive constraints)

设 A_1 、 A_2 产品的生产数量分别为 x_1 、 x_2 件，在不考虑 B_{31} 和 B_{32} 相互排斥的情况下，问题的数学模型为

$$\begin{aligned} \max z &= 25x_1 + 40x_2 \\ s.t. \left\{ \begin{array}{l} 0.3x_1 + 0.7x_2 \leq 250 \\ 0.2x_1 + 0.1x_2 \leq 100 \\ 0.3x_1 + 0.5x_2 \leq 150 \\ 0.2x_1 + 0.4x_2 \leq 120 \\ x_1, x_2 \geq 0, \text{且为整数} \end{array} \right. \end{aligned} \quad \begin{array}{l} (1) \\ (2) \end{array}$$

➤ 2.互排斥的约束条件(Mutually exclusive constraints)

工序 B_3 只能从两种加工方式中选择一种，那么约束条件（1）和（2）就成为相互排斥的约束条件。为了统一在一个问题中，引入0-1变量

$$y_i = \begin{cases} 1, & B_3 \text{采用} B_{3i} \text{加工方式} \\ 0, & B_3 \text{不采用} B_{3i} \text{加工方式} \end{cases} \quad i=1,2$$

$$\max z = 25x_1 + 40x_2$$

则数学模型为

$$s.t. \begin{cases} 0.3x_1 + 0.7x_2 \leq 250 \\ 0.2x_1 + 0.1x_2 \leq 100 \\ 0.3x_1 + 0.5x_2 \leq 150 + M(1 - y_1) \\ 0.2x_1 + 0.4x_2 \leq 120 + M(1 - y_2) \\ y_1 + y_2 = 1 \\ x_1, x_2 \geq 0, \text{且为整数} \\ y_1, y_2 = 0 \text{或} 1 \end{cases}$$

➤ 3.固定成本问题 (Fixed cost problem)

例3 某公司制造小、中、大三种尺寸的容器，所需资源为金属板、劳动力和机器设备，制造一个容器所需的各种资源的数量如下表所示：不考虑固定费用，小、中、大号容器每售出一个其利润分别为4万元、5万元、6万元，可使用的金属板有500吨，劳动力有300人/月，机器有100台/月，另外若生产，不管每种容器生产多少，都需要支付一笔固定费用：小号为100万元，中号为150万元，大号为200万元。问如何制定生产计划使获得的利润对大？

➤ 3.固定成本问题 (Fixed cost problem)

资源	小号容器	中号容器	大号容器
金属板 (吨)	2	4	8
劳动力 (人/月)	2	3	4
机器设备 (台/月)	1	2	3

设 x_1 、 x_2 、 x_3 分别为小号容器、中号容器、大号容器的生产数量。不考虑固定费用，则问题的数学模型为

$$\begin{aligned} \max z &= 4x_1 + 5x_2 + 6x_3 \\ s.t. \quad &\begin{cases} 2x_1 + 4x_2 + 8x_3 \leq 500 \\ 2x_1 + 3x_2 + 4x_3 \leq 300 \\ x_1 + 2x_2 + 3x_3 \leq 100 \\ x_1, x_2, x_3 \geq 0 \end{cases} \end{aligned}$$

➤ 3.固定成本问题 (Fixed cost problem)

若考虑固定费用就必须引入0—1变量:

$$y_i = \begin{cases} 1, & \text{当生产第} i \text{ 种容器即} x_i > 0 \text{ 时} \\ 0, & \text{当不生产第} i \text{ 种容器时即 } x_i = 0 \end{cases} \quad i = 1, 2, 3$$

则该问题的
数学模型为

$$\begin{aligned} \max z &= 4x_1 + 5x_2 + 6x_3 - 100y_1 - 150y_2 - 200y_3 \\ s.t. \quad &\begin{cases} 2x_1 + 4x_2 + 8x_3 \leq 500 \\ 2x_1 + 3x_2 + 4x_3 \leq 300 \\ x_1 + 2x_2 + 3x_3 \leq 100 \\ x_1 - My_1 \leq 0 \\ x_2 - My_2 \leq 0 \\ x_3 - My_3 \leq 0 \\ x_1, x_2, x_3 \geq 0 \\ y_1, y_2, y_3 = 0 \text{ or } 1 \end{cases} \end{aligned}$$

➤ 4.布点问题 (Location Problem)

例4 某城市消防队布点问题。该城市共有6个区，每个区都可以建消防站，市政府希望设置的消防站最少，但必须满足在城市任何地区发生火警时，消防车要在15 分钟内赶到现场。据实地测定，各区之间消防车行驶的时间见下表，请帮助该市制定一个布点最少的计划。

	地区1	地区2	地区3	地区4	地区5	地区6
地区1	0	10	16	28	27	20
地区2	10	0	24	32	17	10
地区3	16	24	0	12	27	21
地区4	28	32	12	0	15	25
地区5	27	17	27	15	0	14
地区6	20	10	21	25	14	0

单位：min

➤ 4.布点问题 (Location Problem)

引入0 - 1变量 x_i 作决策变量, 令

$$x_i = \begin{cases} 1, & \text{表示在地区} i \text{设消防站} \\ 0, & \text{表示在地区} i \text{不设消防站} \end{cases}$$

$i = 1, 2, \dots, 6$

- 本问题的约束方程是要保证每个地区都有一个消防站在15分钟行程内。如地区1, 由上表可知, 在地区1及地区2内设消防站都能达到此要求, 即 $x_1 + x_2 \geq 1$
- 因此本问题的数学模型为:

$$\min z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6$$

$$x_1 + x_2 \geq 1$$

$$x_1 + x_2 + x_6 \geq 1$$

$$x_3 + x_4 \geq 1$$

$$x_3 + x_4 + x_5 \geq 1$$

$$x_4 + x_5 + x_6 \geq 1$$

$$x_2 + x_5 + x_6 \geq 1$$

$$x_i = 1 \text{ 或 } 0 \quad (i = 1, \dots, 6)$$

Assignment problem

Assignment problem

- 指派问题是一种特殊的整数规划问题。在实践中经常会遇到一种问题：某单位有 m 项任务要 m 个人去完成（每人只完成一项工作），在分配过程中要充分考虑各人的知识、能力、经验等，应如何分配才能使工作效率最高或消耗的资源最少？这类问题就属于指派问题。引入0—1变量 x_{ij}

$$x_{ij} = \begin{cases} 1 & \text{指派第 } i \text{ 个人完成第 } j \text{ 项任务} \\ 0 & \text{不指派第 } i \text{ 个人完成第 } j \text{ 项任务} \end{cases}$$

$$\min z = \sum_{i=1}^m \sum_{j=1}^m c_{ij} x_{ij} \quad s.t. \begin{cases} \sum_{j=1}^m x_{ij} = 1 & i = 1, 2, \dots, m \\ \sum_{i=1}^m x_{ij} = 1 & j = 1, 2, \dots, m \\ x_{ij} = 0 \text{ 或 } 1 \end{cases}$$

Example

- 福尔市场营销调查公司有3个新客户需要进行市场调查，目前正好有3个人没有其他工作，由于他们的对不同市场的经验和能力不同，估计他们完成不同任务所需时间如下表。公司面临的问题是如何给每个客户指派一个项目主管（代理商），使他们完成市场调查的时间最短。

预计完成时间

项目主管 \ 客户			
	1	2	3
1	10	15	9
2	9	18	5
3	6	14	3

Example

设 $x_{ij}=1$ 表示指派主管 i 完成第 j 项市场调查,
否则 $x_{ij}=0$

则问题的数学模型为:

$$\min f = 10x_{11} + 15x_{12} + 9x_{13} + 9x_{21} + 18x_{22} + 5x_{23} + 6x_{31} + 14x_{32} + 3x_{33}$$

$$x_{11} + x_{12} + x_{13} = 1$$

$$x_{21} + x_{22} + x_{23} = 1$$

$$x_{31} + x_{32} + x_{33} = 1$$

$$x_{11} + x_{21} + x_{31} = 1$$

$$x_{12} + x_{22} + x_{32} = 1$$

$$x_{13} + x_{23} + x_{33} = 1$$

$$x_{ij} \geq 0, \quad i=1, 2, 3; \quad j=1, 2, 3$$