Lecture -13
DISCRETE OPTIMIZATION

Reference: Book Chapters 9&14

Introduction

- In **continuous optimization** problems, the design variables were assumed to be continuous, i.e., they can assume any real values within given ranges.
- In many practical engineering problems, the acceptable values of the design variables do not form a continuous set; such problems are referred to as ***discrete optimization problems***.

Example:

- The number of rivets required in a riveted joint has to be an integer (such as 1, 2, 3).

INTRODUCTION

- A generic discrete optimization problem can be defined as

$$\min_{x,y,z} \; f(x,y,z)$$

objective function

design variable vectors

subject to

$$g(x,y,z) \leq 0$$

inequality constraint function

$$h(x,y,z) = 0$$

equality constraint

$$x \in Z^m$$

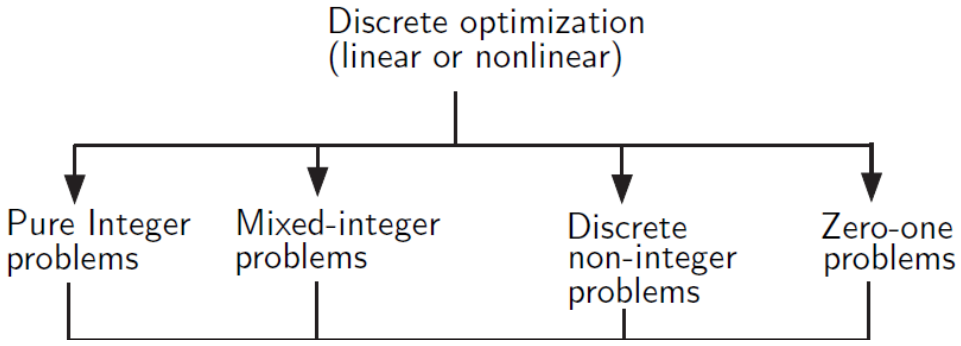
$$y \in R^n$$

$$z \in A$$

Z^m is a set of given feasible integers; R^n is a set of real numbers; and A stands for a combinatorial set resulting from given feasible discrete values.

INTRODUCTION

- The following terms are commonly encountered within the umbrella of discrete optimization literature; and define the main classes of discrete optimization problems.
    - Pure integer programming problems.
    - Mixed-integer programming problems.
    - Discrete non-integer optimization problems.
    - Zero-one (or binary) Programming problems.
    - Combinatorial optimization problems.



INTRODUCTION

- ***Pure integer programming problems***

In these problems, the design variables are allowed to assume only integer values. Example: Number of routes a travelling salesman can take.

- ***Mixed-integer programming problems***

In these problems, some design variables are allowed to assume only integer values, while others are allowed to assume continuous values.

- ***Discrete non-integer optimization problems:***

In these problems, the design variables are allowed to assume only a given set of discrete, not necessarily integer, values.
Example: Standardized sizes of machine components.

INTRODUCTION

- **Zero-one (or binary) Programming problems:**

In these problems, the design variables are allowed to assume values of either 0 or 1.

- **Combinatorial optimization problems:**

In these problems, the set of possible feasible solutions is defined by a combinatorial set resulting from the given feasible discrete values of the design variables.

INTRODUCTION

Example:

1. Knapsack problem:

- This is a classical example of a combinatorial optimization problem.
- A select set of items needs to be packed in a knapsack, or a bag. Each member of the set of items has a cost and a merit associated with it.
- How can items be optimally chosen to be packed, such that the cost is minimized, and the merit maximized?

2. Vehicle routing problem:

- The goal of this problem is to choose the route so as to minimize the distribution cost of the goods to be delivered to a set of customers.
- The goods are located in a central inventory.

INTRODUCTION

Example:

3. Traveling salesman problem:

- A traveling salesman needs to plan a round-trip route.
- The salesman needs to visit a given number of cities, and each city exactly once. Each segment of the journey (from one city to another) has an associated cost.
- Which route yields the least expensive round-trip that visits each city exactly once?

4. Capital budgeting:

- This problem optimizes the allocation of capital budgets among different investment alternatives.
- Suppose a given amount of money in invested, among four alternatives. Each investment alternative has a present value and a minimum required investment.

COMPUTATIONAL COMPLEXITY

- Discrete optimization problems are known to be computationally complex.
- The study of complexity is related to a branch of mathematics and computing, known as complexity theory.
- One of the issues of interest in complexity theory is to quantify the performance of computational algorithms.
- The computational time of an algorithm is usually presented using the O (known as the Big O) notation.

COMPUTATIONAL COMPLEXITY

Example:

-The gradient of a function f(x1, x2, ...xn) using the finite difference method.

-The i-th component of the n × 1 gradient vector can be computed as

$$\left.\frac{\partial f}{\partial x_i}\right|_{x=p} = \frac{f(p+\delta_p) - f(p)}{\delta_p}$$

where δ_p is the chosen step size.

- To compute all of the n components of the gradient vector, we require n+1 function evaluations.
- Therefore the above gradient computation algorithm using finite differences has a complexity of the order of *O(n)* for a problem size of n.

COMPUTATIONAL COMPLEXITY

- An algorithm is said to be of **polynomial time** if its complexity is of the order of $O(n^k)$ for a problem size of n, where k is a finite non-negative integer.
- Another aspect of complexity theory is to categorize computational problems and algorithms into complexity classes.
- One important complexity class is known as the **P class**.
- The problems belonging to P class are regarded as tractable, and easily solvable by algorithms in polynomial time.
- Unfortunately, several practical discrete optimization problems belong to the notoriously difficult complexity class of the so-called **NP-Complete problems**.

BASIC SOLUTION APPROACHES

- From formulation and solution approaches perspectives, discrete problems can be classified as pure-integer problems, mixed-integer problems, discrete non-integer problems, and zero-one problems.

- Discrete optimization (linear or nonlinear) solution approaches include
  - Exhaustive/enumerative search
  - Graphical method
  - Relaxation approach
  - Branch and bound method
  - Cutting plane method
  - Evolutionary algorithms

BRUTE FORCE METHOD: EXHAUSTIVE SEARCH

- **The most straightforward, and computationally expensive, approach** to solving discrete problems is to perform an exhaustive search – where all the possible options are enumerated and evaluated.

- The optimal solution can then be readily selected from the enumerated solutions.

- This approach is a viable option only for small problems, as it can be computationally prohibitive for larger problems.

BRUTE FORCE METHOD: EXHAUSTIVE SEARCH

Example

$$\min_{x,y,z} \; f(x) = 5(x_1 - 4)^2 + 4x_2x_3$$

Subject to

$$x_1 \in \{1, 3, 5.5\}$$
$$x_2 \in \{2, 3.5, 5.5\}$$
$$x_3 \in \{4, 7\}$$

The number of feasible combinations of the three design variables: $3 \times 3 \times 2 = 18$

Design Variable Combinations and Their Objective Function Values

x_1	x_2	x_3	$f(x)$	x_1	x_2	x_3	$f(x)$
1	2	4	77	1	2	7	101
1	3.5	4	101	1	3.5	7	143
1	5.5	4	133	1	5.5	7	199
3	2	4	37	3	2	7	61
3	3.5	4	61	3	3.5	7	103
3	5.5	4	93	3	5.5	7	159
5.5	2	4	43.25	5.5	2	7	67.25
5.5	3.5	4	67.25	5.5	3.5	7	109.25
5.5	5.5	4	99.25	5.5	5.5	7	165.25

The minimum of f(x) is: 37

The optimal solution is: $x_1 = 3$, $x_2 = 2$, and $x_3 = 4$.

Example:

- We have a set of 6 truss elements. The generic i-th element has a weight w_i.

Element, i	1	2	3	4	5	6
Weight, w_i	1.5	6.4	2.0	3.2	5.7	4.3

We need to choose three elements that yield a minimum total weight.

- This problem can be viewed as a combinatorial optimization problem, and can be stated as

$$\min_{x_i, x_j, x_k} \quad w_i + w_j + w_k$$

Such that

$$x_i \neq x_j \neq x_k$$

$$x_i, x_j, x_k \in \{x_1, x_2, x_3, x_4, x_5, x_6\}$$

$$\{w_1, w_2, w_3, w_4, w_5, w_6\} = \{1.5, 6.4, 2.0, 3.2, 5.7, 4.3\}$$

BRUTE FORCE METHOD: EXHAUSTIVE SEARCH

There are $C^6_3 = \frac{6!}{3!(6-3)!}$ = 20 ways of choosing three elements from six possibilities.

The optimal solution

Combination	Total weight	Combination	Total weight
1,2,3	9.9	2,3,4	11.6
1,2,4	11.1	2,3,5	14.1
1,2,5	13.6	2,3,6	12.7
1,2,6	12.2	2,4,5	15.3
1,3,4	**6.7**	2,4,6	13.9
1,3,5	9.2	2,5,6	16.4
1,3,6	7.8	3,4,5	10.9
1,4,5	10.4	3,4,6	9.5
1,4,6	9	3,5,6	12
1,5,6	11.5	4,5,6	13.2

GRAPHICAL METHOD

- For simple problems where **the objective function and constraints can be visualized**, the feasible region can be plotted to find the optimal discrete solution graphically.
- The graphical solving may not be a viable option for complex problems with a large number of design variables.

GRAPHICAL METHOD

Example:

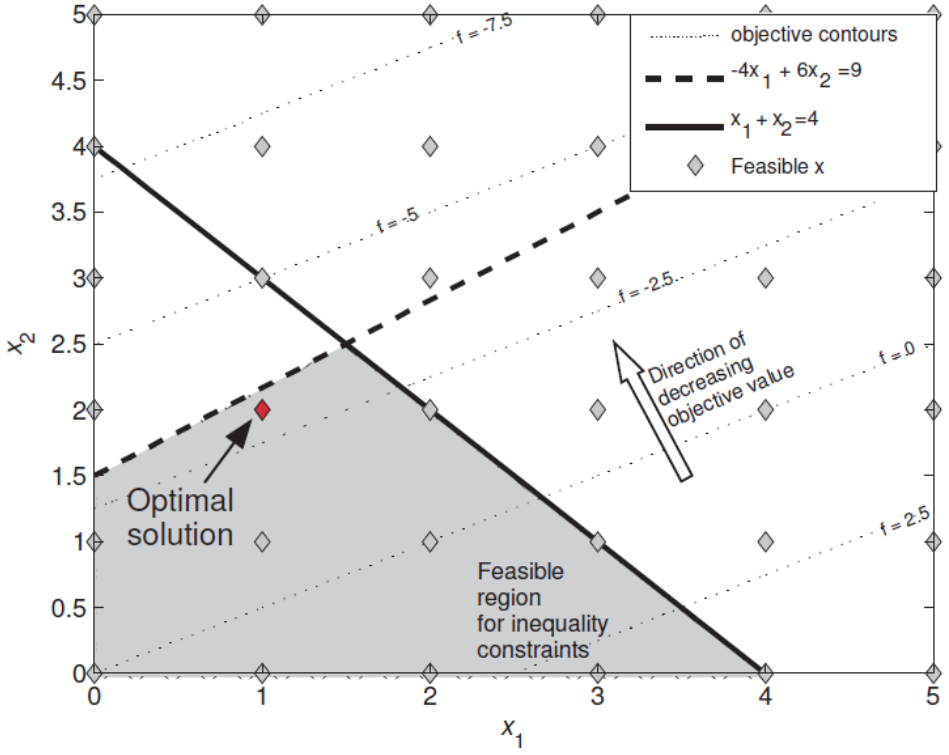
$$\min_{x} \; x_1 - 2x_2$$

such that

$$-4x_1 + 6x_2 \leq 9$$
$$x_1 + x_2 \leq 4$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2 \in Z$$

GRAPHICAL METHOD

- The direction of decreasing objective function value is shown by the block arrow.
- The grey shaded region represents the feasible region as defined by the inequality constraints.
- The diamonds represent feasible integer solutions.



- The optimal solution is $\mathbf{x_1 = 1, x_2 = 2}$, which is the integer solution within the feasible design space with the least objective function value.

RELAXATION APPROACH

- In this method, the discrete variables are assumed to be continuous variables; and the optimization problem is solved using the continuous optimization techniques.
- The real-valued optimum design variables obtained are then **rounded off to the nearest feasible discrete solution**.
- While this technique is used quite often on account of its ease of implementation, the user is warned that the solution obtained can often be sub-optimal.
- Noted that rounding of the optimal solution can result in constraints violations at the approximate discrete solution.

RELAXATION APPROACH

Example

$$\min_{x} \quad -2x_1 + x_2$$

Subject to

$$6x_1 - 4x_2 \leq 15$$
$$x_1 + x_2 \leq 5$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2 \in Z$$

By ignoring the constraint ($x_1, x_2 \in Z$) the problem can be solved as a continuous optimization problem.

MATLAB function ***linprog*** is used to solve the relaxed linear programming problem.

Main Function (to be continued)

```
clear
clc
% objective function
f = [-2; 1];
% Define Linear constraints
A=[6 -4; 1 1]; B=[15; 5];
Aeq=[]; Beq=[];
```


RELAXATION APPROACH

```
% Define bounds constraints
LB=[0 0]; UB=[];
[x, fopt] = linprog(f, A, B, Aeq, Beq, LB, UB)
```

The final solution obtained is: $x_1 = 3.5$ and $x_2 = 1.5$.

Rounding off to the nearest integer yields an integer optimal solution is $x_1 = 4$, $x_2 = 2$.

The solutions **do not satisfy** the inequality constraint equation($x_1 + x_2 \leq 5$)

If we round off the continuous solution to $x_1 = 3$, $x_2 = 1$, we obtain the **correct solution**.

RELAXATION APPROACH

Example

$$\min_{x} \quad -5x_1 - x_2$$

Subject to

$$10x_1 + x_2 \leq 20$$
$$x_2 \leq 2$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2 \in Z$$

By ignoring the constraint ($x_1, x_2 \in Z$) the problem can be solved as a continuous optimization problem.

MATLAB function ***linprog*** is used to solve the relaxed linear programming problem.

Main Function (to be continued)

```
clear
clc
% objective function
f = [-5; -1];
% Define Linear constraints
A=[10 1; 0 1]; B=[20; 2];
Aeq=[]; Beq=[];
```


RELAXATION APPROACH

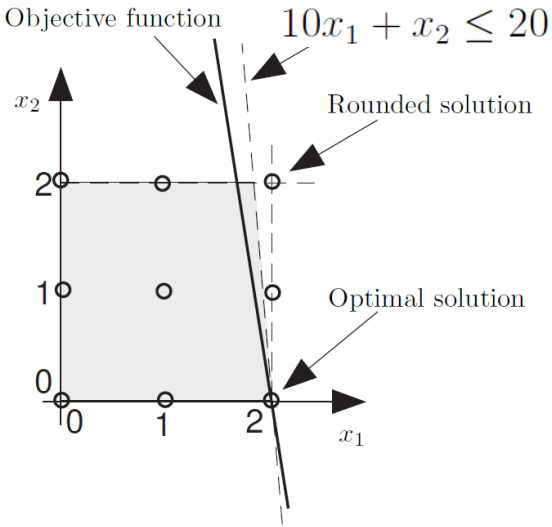
```
% Define bounds constraints
LB=[0 0]; UB=[];
[x, fopt] = linprog(f, A, B, Aeq, Beq, LB, UB))
```

The optimal solution of the relaxed optimization problem is $x_1 = 1.8$ and $x_2 = 2$.

Rounding off to the nearest integer yields an integer optimal solution is $x_1 = 2$ and $x_2 = 2$.

By graphically solving the optimization problem

The optimal solution is $x_1 = 2$ and $x_2 = 0$



This example shows that the relaxation approach can be misleading.

RELAXATION APPROACH

Example:

$$\min_x \; x_1 - 2x_2$$

such that

$$-4x_1 + 6x_2 \leq 9$$
$$x_1 + x_2 \leq 4$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2 \in Z$$

Now solve it as a continuous optimization problem

$$\min_x \; x_1 - 2x_2$$

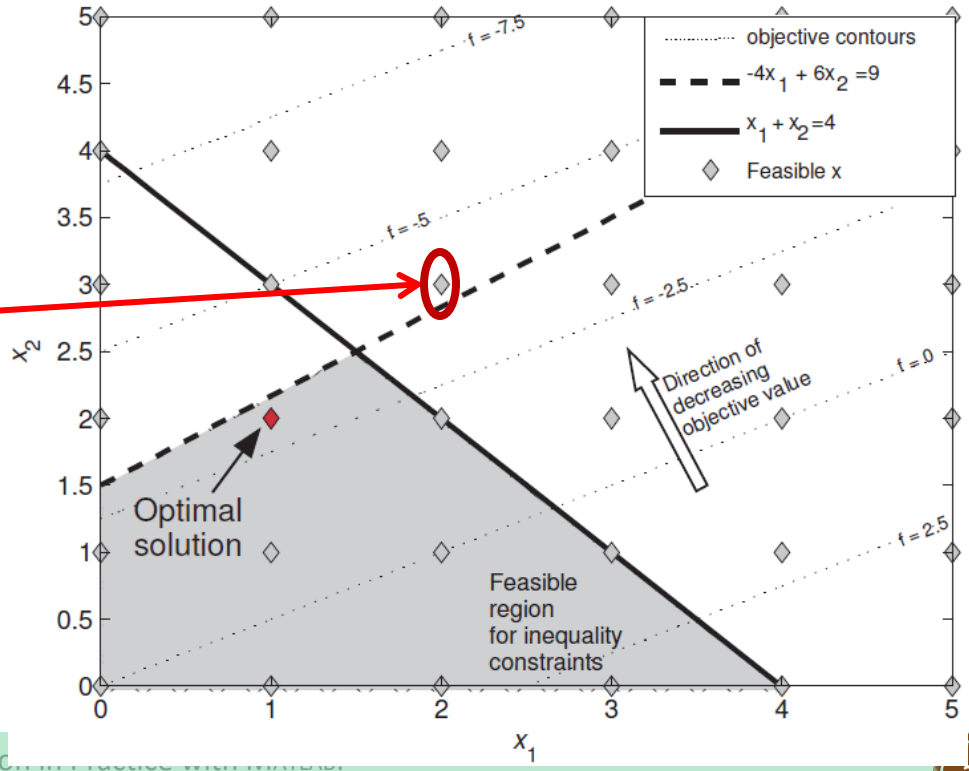
such that

$$-4x_1 + 6x_2 \leq 9$$
$$x_1 + x_2 \leq 4$$
$$x_1, x_2 \geq 0$$

RELAXATION APPROACH

- Solving this optimization problem using a continuous optimization algorithm, such as fmincon, yields the optimal solution as $x_1 = 1.5, x_2 = 2.5$.

- **Rounding off to the nearest integer would yield an integer optimal solution of $x_1 = 2, x_2 = 3$.**



RELAXATION APPROACH

- As seen, the rounded solution lies in the infeasible region of the design space.
- If the continuous solution rounded off to $x_1 = 1$, $x_2 = 2$, we obtain the correct solution.
- The relaxation approach for solving discrete problems must be carefully employed.

BRANCH AND BOUND

- The branch and bound method is a basic technique used to solve discrete programming problems.
- This method is based on the observation that the enumeration of integer solutions has a tree structure.
- It enumerates candidate solutions systematically for a discrete optimization problem.

BRANCH AND BOUND

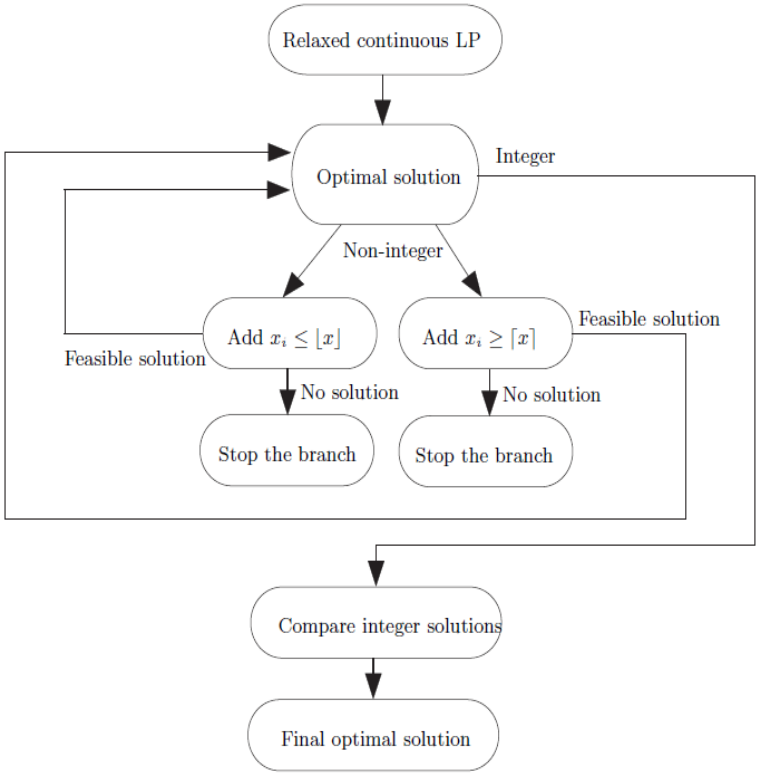
- *The procedure of the branch and bound method used to solve a **linear integer programming** problem*

Relaxed continuous LP:

-A relaxed continuous linear programming (LP) problem is formulated by ignoring the integer constraints.

-The resulting optimal solution may have some non-integer variable values.

-If the resulting LP solution has only integer values, the obtained solution is the integer optimal solution.

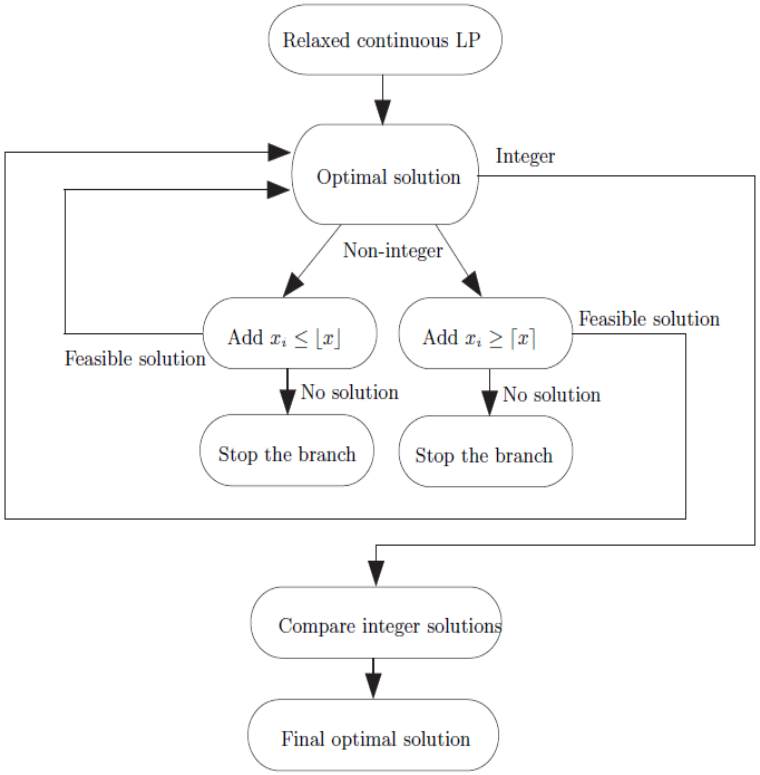


BRANCH AND BOUND

- *The procedure of the branch and bound method used to solve a linear integer programming problem*

Ceil:

The notation of [x] is defined as the ceiling function. This function returns the smallest integer value which is greater than or equal to x. For example,
[5.14] = 6, [10] = 10, and [−8.6] = −8.



Floor:

The floor function is denoted ⌊x⌋, which returns the largest integer that is less than or equal to x.
For example,
⌊5.14⌋ = 5, ⌊10⌋ = 10, and ⌊−8.6⌋ = −9.

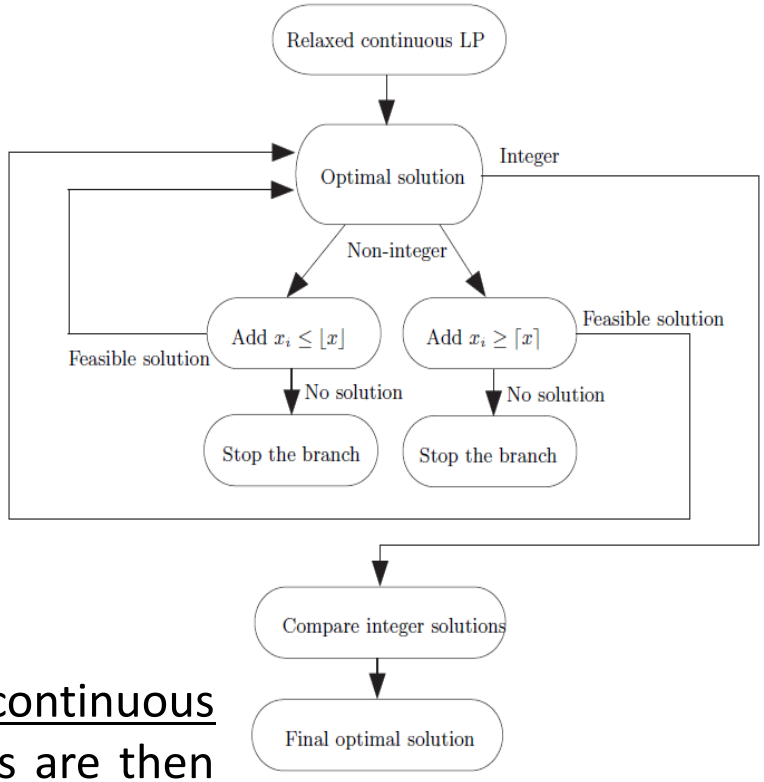
BRANCH AND BOUND

➢ *The procedure of the branch and bound method used to solve a linear integer programming problem*

A ceil or a floor

For those design variables with decimal parts in the optimal result, two sub-problems are created by imposing a ceil or a floor on the design variable values.

- The first sub-problem is formulated by adding the constraint $x_i \leq \lfloor x \rfloor$.
- The second sub-problem is formulated by adding the constraint $x_i \geq \lceil x \rceil$.



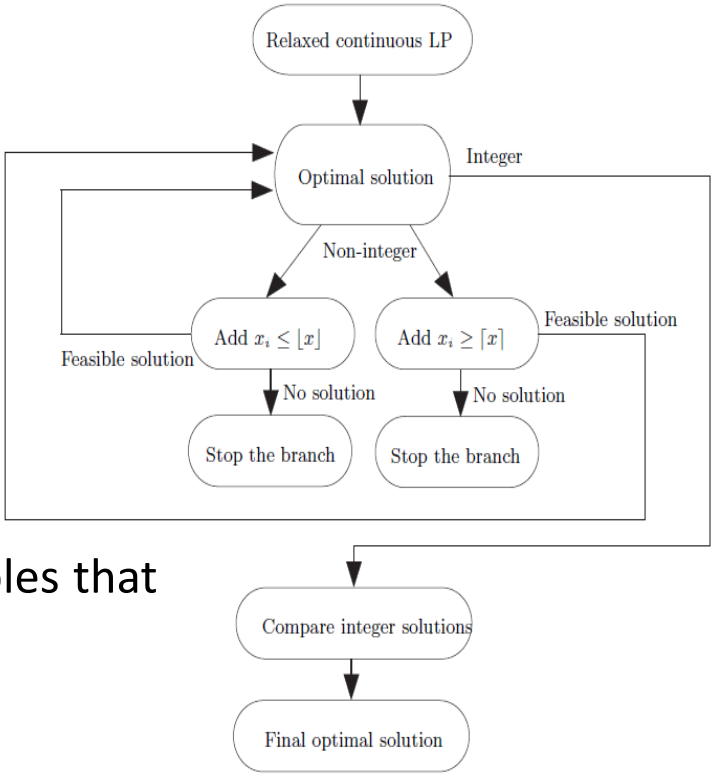
The two sub-problems are then solved as continuous problems. The solutions of the two sub-problems are then examined for fractional parts, and the process is repeated.

BRANCH AND BOUND

➢ *The procedure of the branch and bound method used to solve a linear integer programming problem*

Branching:

- The process of adding a ceil or a floor is called branching.
- For a given variable, the branching process is repeated until the relaxed continuous problem with the additional constraints yields either an integer solution or an infeasible solution.



- The branching process is repeated for all variables that have fractional solutions.

Bounds:

The added ceil and floor constraints are called bounds on the variable values.

BRANCH AND BOUND

- *Implementation of the branch and bound method for linear integer programming problems*

Step 1. Formulate a relaxed continuous linear programming problem by ignoring the integer constraints. The relaxed problem comprises continuous variables.

Step 2. If the solution to the above relaxed continuous linear optimization problem only has integers, it is the optimal solution. If the solution has non-integer variables, go to the next step.

Step 3. Select one non-integer variable, and generate two sub-problems (two branches). For one of them, add a ceil constraint to the selected non-integer variable, and for the other branch, add a floor constraint to it.

Step 4. If a branch has no feasible solutions, stop this branch. If the solution only has integers, it becomes a candidate for the final optimal solution. If the solution has non-integer values, go to Step 3.

BRANCH AND BOUND

- *Implementation of the branch and bound method for linear integer programming problems*

Step 5. Once the branching process is completed for all variables, compare all the integer solutions. The best solution is considered the final optimal solution..

BRANCH AND BOUND

Example

$$\min_x \; -2x_1 + x_2$$

Subject to

$$6x_1 - 4x_2 \leq 15$$
$$x_1 + x_2 \leq 5$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2 \in Z$$

Formulate a relaxed continuous linear programming problem

$$\min_x \; -2x_1 + x_2$$
subject to
$$6x_1 - 4x_2 \leq 15$$
$$x_1 + x_2 \leq 5$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2 \in Z$$

solve as a continuous problem by ignoring integer constraints.

BRANCH AND BOUND

$$\begin{aligned} \min_{x} \quad & -2x_1 + x_2 \\ \text{subject to} \quad & \\ & 6x_1 - 4x_2 \leq 15 \\ & x_1 + x_2 \leq 5 \\ & x_1, x_2 \geq 0 \\ & x_1, x_2 \in Z \end{aligned}$$

$$\begin{aligned} \min_{x} \quad & -2x_1 + x_2 \\ \text{subject to} \quad & \\ & 6x_1 - 4x_2 \leq 15 \\ & x_1 + x_2 \leq 5 \\ & x_1, x_2 \geq 0 \end{aligned}$$

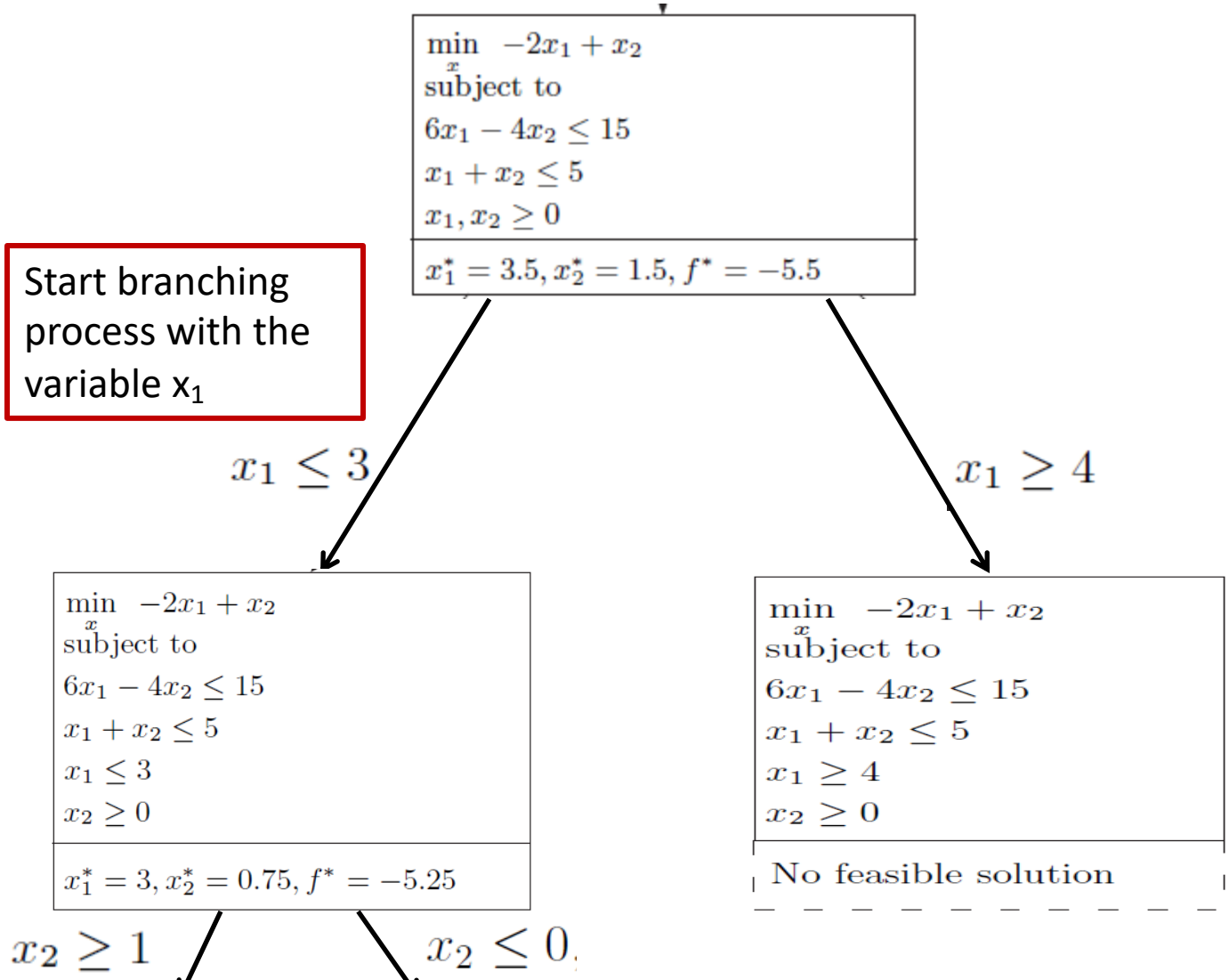
$$x_1^* = 3.5, x_2^* = 1.5, f^* = -5.5$$

Start branching process with the variable x_1

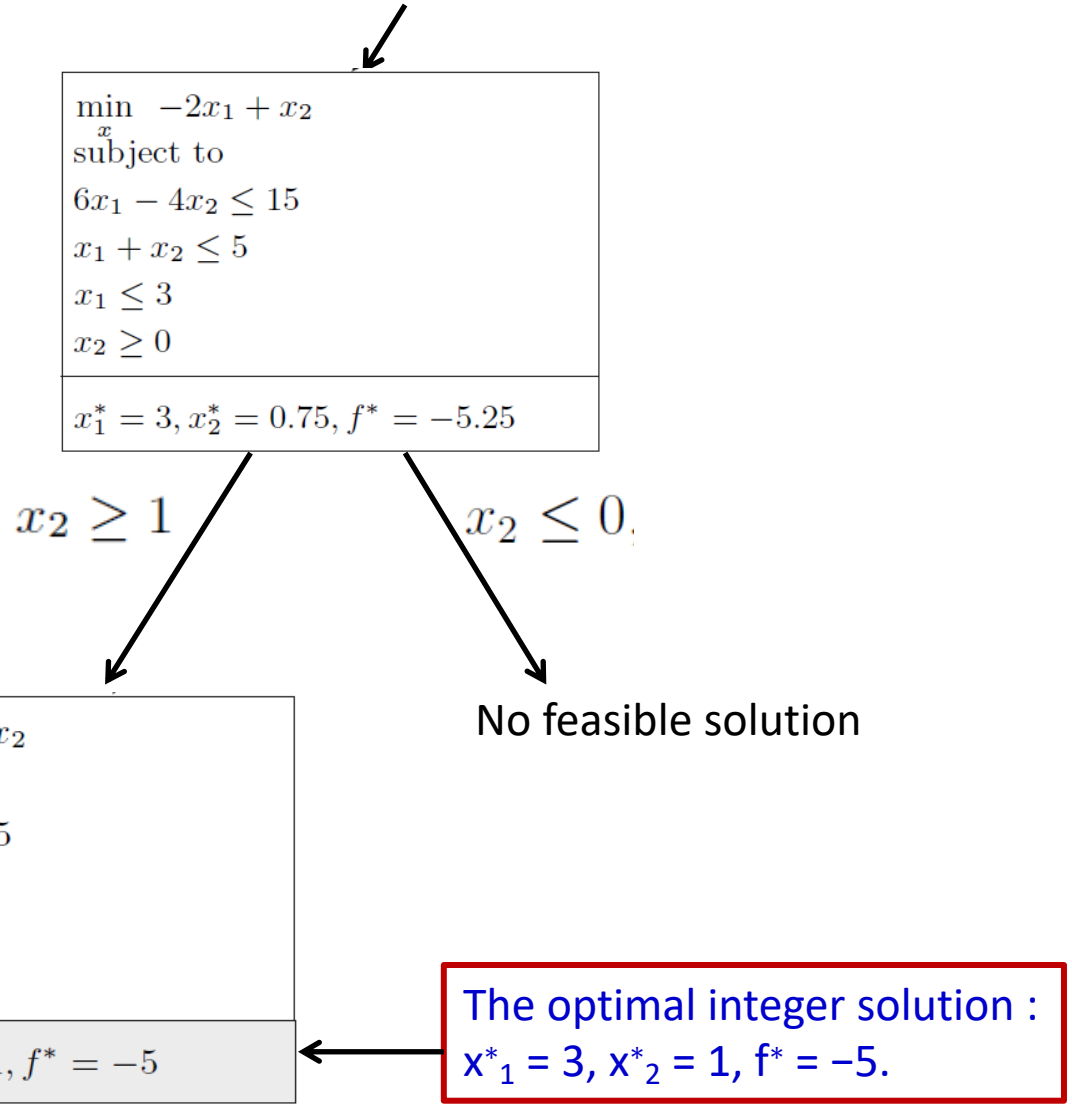
$x_1 \leq 3$

$x_1 \geq 4$

BRANCH AND BOUND



BRANCH AND BOUND



CUTTING PLANE METHOD

- The basic idea in the cutting plane method is to add inequalities, also known as cuts, to the existing integer problem.
- The purpose of adding the inequality is to cut off the non-integer solutions without eliminating integer points of the feasible region.
- The resulting formulation is then solved as a non-integer continuous problem, and is tested if the obtained solution is an integer. If not, a new cut is added and the procedure is repeated until an optimal integer solution is found.
- There are different approaches for finding effective inequalities or cuts that are added to the initial set of constraints.
- These effective inequalities are usually taken from pre-defined families, one of which is the **Gomory cut**.

CUTTING PLANE METHOD

- In linear integer programming problem, A Gomory cut is a linear inequality constraint that does not exclude any feasible integer solutions from the integer problem.
- The following is a generic discussion of the Gomory's cutting plane method by using the **Simplex method** for **linear integer programming** problems.

CUTTING PLANE METHOD

Step 1. Initial Simplex Tableau:

- Relax the integer programming problem by ignoring the integer constraints on the design variables.
- The linear programming (LP) formulation can be written as

$$\min_{x} \; c^T x$$

such that

$$Ax \leq b$$

$$A_{eq}x = b_{eq}$$

$$x \geq 0$$

CUTTING PLANE METHOD

Step 1. Initial Simplex Tableau:

- Add slack variables, denoted by the vector ***s***, to the inequality constraints for the Simplex method, obtain the following formulation.

$$\min_{x} \; c^T x$$

such that

$$Ax + s = b$$
$$A_{eq}x = b_{eq}$$
$$x, s \geq 0$$

- The above relaxed linear programming formulation is solved using the Simplex method.
  - If the resulting optimal solution consists of only integer values, the solution procedure can be stopped.
  - If that is not the case, the following procedure is used to obtain integer solutions.

Step 2. Generating the Gomory Cut:

- Examine the current basic variables in the Simplex tableau.
- Choose arbitrarily one basic variable with a fractional value, say $\mathbf{x_i}$.
- The following equation corresponds to the row of $\mathbf{x_i}$ in the Simplex tableau.

$$x_i = b_i - \{a_{i1}x_1 + \ldots + a_{it}x_t\}$$

where ***t*** is the total number of variables: n original variables and ***m*** slack variables. In the above equation, $\boldsymbol{x_i}$ and $\boldsymbol{b_i}$ are fractional.

CUTTING PLANE METHOD

Step 2. Generating the Gomory Cut:

- Separate each of the above coefficients into their respective integer and fractional parts.

Let $\mathbf{b_i = b_{Z:i} + b_{f:i}}$,

where $b_{Z:i}$ is an integer, and $b_{f:i}$ is a positive fraction such that $0 \leq b_{f:i} \leq 1$.

- Similarly, separate each of the coefficients, a_{ij} , j = {1, ..., t}, into an integer part, $a_{Z:j}$, and non-negative fractional part, $a_{f:j}$,

$$\mathbf{a_{ij} = a_{Z:j} + a_{f:j}}$$

CUTTING PLANE METHOD

Example: Consider the following equation:

$$x_2 = \frac{53}{11} - \frac{2}{11}x_3 + \frac{4}{11}x_4$$

It can be re-written as

$$x_2 = (4 + \frac{9}{11}) - [(0 + \frac{2}{11})x_3 + (-1 + \frac{7}{11})x_4]$$

$$x_i = b_i - \{a_{i1}x_1 + \ldots + a_{it}x_t\}$$

$$x_i = b_{Z:i} + b_{f:i} - [\{a_{Z:1} + a_{f:1}\}x_1 + \ldots + \{a_{Z:t} + a_{f:t}\}x_t]$$

This equation can further be written

$$b_{f:i} - [a_{f:1}x_1 + \ldots + a_{f:t}x_t] = x_i - b_{Z:i} + [a_{Z:1}x_1 + \ldots + a_{Z:t}x_t]$$

The right hand side of the above equation is always an integer. Therefore the left hand side should also be an integer.

CUTTING PLANE METHOD

$$b_{f:i} - [a_{f:1}x_1 + \ldots + a_{f:t}x_t] \in Z$$

Since $a_{f:i}$ are non-negative fractions and $x \geq 0$, the quantity $[a_{f:1}x_1 + \ldots + a_{f:t}x_t]$ is non-negative. Also, note that $0 < b_{f:l} < 1$. Therefore, the following equation holds.

$$b_{f:i} - [a_{f:1}x_1 + \ldots + a_{f:t}x_t] \leq b_{f:i} \leq 1$$

This implies that

$$b_{f:i} - [a_{f:1}x_1 + \ldots + a_{f:t}x_t] \leq 0$$

This condition yields the **Gomory constraint**, given as

CUTTING PLANE METHOD

Step 3. Solve problem with Gomory constraint:

- Adding a slack variable to $b_{f:i} - [a_{f:1}x_1 + \ldots + a_{f:t}x_t] \leq 0$

  we obtain

$$b_{f:i} - [a_{f:1}x_1 + \ldots + a_{f:t}x_t] + s_{t+1} = 0$$

- The above constraint is then added to the linear programming problem from STEP 1.
- The optimal solution obtained in STEP 1 (initial Simplex tableau) does not satisfy the Gomory constraint generated above.
- In this case, the basic Simplex method used in STEP 1 cannot be used to solve the new problem.
- Use the dual Simplex method for solving the new problem with the Gomory constraint.

CUTTING PLANE METHOD

Step 4. Repeat

The process of generating the Gomory constraint and solving the new formulation with the dual Simplex method is repeated until integer solutions are obtained.

CUTTING PLANE METHOD

Some shortcomings of this method are:

(1) As the number of variables grows, the size and the complexity of the problem also grows, since an additional slack variable and an additional constraint are added for each non-integer variable.

(2) Implementing this method in a computer using decimal representation can result in rounding errors as the algorithm proceeds, and might result in a wrong integer optimal solution.

CUTTING PLANE METHOD

Example:

The linear integer programming problem is given as follows.

$$\min_{x} \; x_1 - 2x_2$$

such that

$$-4x_1 + 6x_2 \leq 9$$

$$x_1 + x_2 \leq 4$$

$$x_1, x_2 \geq 0$$

$$x_1, x_2 \text{ are integers}$$

CUTTING PLANE METHOD

1. Initial Simplex Tableau:

Temporarily ignore the integer constraints on x_1 and x_2. The standard linear programming formulation for Simplex method can be written as follows.

$$\min_x \; x_1 - 2x_2$$

such that

$$-4x_1 + 6x_2 + s_1 = 9$$
$$x_1 + x_2 + s_2 = 4$$
$$x_1, x_2, s_1, s_2 \geq 0$$

CUTTING PLANE METHOD

- Initial Simplex Tableau Before Adding Cutting Planes

	x_1	x_2	s_1	s_2	b	$\frac{b_i}{a_{ij}}$
R_1	-4	**6**	1	0	9	$\mathbf{\frac{3}{2}}$
R_2	1	1	0	1	4	4
f	1	**-2**	0	0	f	

- Final Simplex Tableau Before Adding Cutting Planes

	x_1	x_2	s_1	s_2	b
R_1	0	1	$\frac{1}{10}$	$\frac{2}{5}$	$\frac{5}{2}$
R_2	1	0	$-\frac{1}{10}$	$\frac{3}{5}$	$\frac{3}{2}$
f	0	0	$\frac{3}{10}$	$\frac{1}{5}$	$f + \frac{7}{2}$

CUTTING PLANE METHOD

2. Generating the Gomory Cut:

A constraint is added using the theory of Gomory cut algorithm. From the Final Simplex Tableau, arbitrarily choose a basic variable that has a fractional value, say x_2 (from row R_1). The equation in R_1 can be written as follows.

$$x_2 + \frac{1}{10}s_1 + \frac{2}{5}s_2 = \frac{5}{2}$$

or

$$x_2 = \frac{5}{2} - \frac{1}{10}s_1 - \frac{2}{5}s_2$$

CUTTING PLANE METHOD

- separating the integer and fractional parts.

$$x_2 = 2 + \frac{1}{2} - (0 + \frac{1}{10})s_1 - (0 + \frac{2}{5})s_2$$

$$\downarrow$$

$$\frac{1}{2} - \frac{1}{10}s_1 - \frac{2}{5}s_2 = x_2 - 2$$

- The Gomory cut for the above case can be written as follows.

$$-\frac{1}{10}s_1 - \frac{2}{5}s_2 + s_3 = -\frac{1}{2}$$

$$-\frac{1}{10}s_1 - \frac{2}{5}s_2 \leq -\frac{1}{2}$$

CUTTING PLANE METHOD

3.Solve problem with Gomory constraint (Iteration 1):

- The updated tableau can be written as

	x_1	x_2	s_1	s_2	s_3	b
R_1	0	1	$\frac{1}{10}$	$\frac{2}{5}$	0	$\frac{5}{2}$
R_2	1	0	$-\frac{1}{10}$	$\frac{3}{5}$	0	$\frac{3}{2}$
R_3	**0**	**0**	$-\frac{1}{10}$	$-\frac{2}{5}$	**1**	$-\frac{1}{2}$
f	0	0	$\frac{3}{10}$	$\frac{1}{5}$	0	$f+\frac{7}{2}$
$\frac{c_i}{-a_i}$	N/A	N/A	3	$\frac{1}{2}$	N/A	

- According to the Initial Simplex Tableau After Adding the First Cutting Plane, The linear programming problem will now be solved using dual Simplex method.

CUTTING PLANE METHOD

- Select R_3 from Tableau because it has a negative b value.
- For each negative coefficient in R_3, find the corresponding cost coefficient (entries of the row f), c_i , and compute the following.

$$\min_{a_i<0}\left(\frac{c_i}{-a_i}\right)$$

For example, for s_1 column

$$\left(\frac{c_3}{-a_3}\right)=\frac{\frac{3}{10}}{-\left(\frac{-1}{10}\right)}=3$$

- Since the column for s_2 satisfied the above condition, we pivot on the element −2/5 in R_3

CUTTING PLANE METHOD

- Then the final Simplex tableau can be shown

	x_1	x_2	s_1	s_2	s_3	b
R_1	0	1	0	0	1	2
R_2	1	0	$-\frac{1}{4}$	0	$\frac{3}{2}$	$\frac{3}{4}$
R_3	0	0	$\frac{1}{4}$	1	$-\frac{5}{2}$	$\frac{5}{4}$
f	0	0	$\frac{1}{4}$	0	$\frac{1}{2}$	$f + \frac{13}{4}$

- This is the final Simplex tableau because all the elements in the b vector are positive.
- The optimal values are $x^*_1 = 3/4$, $x^*_2 = 2$, $f = -13/4$.
- Since the optimal solution has fractional parts, another Gomory cutting plane will be generated from tableau .

4. Solve problem with Gomory constraint (Iteration 2):

- Choose x_1 as the variable for which a Gomory constraint is generated, using R_1.
- Separation of the integer and fractional parts of the coefficients, are shown below.

$$x_1 - \frac{1}{4}s_1 + \frac{3}{2}s_3 = \frac{3}{4}$$

$$x_1 + (-1 + \frac{3}{4})s_1 + (1 + \frac{1}{2})s_3 = (0 + \frac{3}{4})$$

$$\frac{3}{4} - \frac{3}{4}s_1 - \frac{1}{2}s_3 = x_1 - s_1 + s_3$$

$$\frac{3}{4} - \frac{3}{4}s_1 - \frac{1}{2}s_3 \leq 0$$

CUTTING PLANE METHOD

- The Gomory constraint can be written as

$$-\frac{3}{4}s_1 - \frac{1}{2}s_3 + s_4 = -\frac{3}{4}$$

- The initial simplex tableau after adding second cutting plane for the above problem can be shown

	x_1	x_2	s_1	s_2	s_3	s_4	b
R_1	0	1	0	0	1	0	2
R_2	1	0	$-\frac{1}{4}$	0	$\frac{3}{2}$	0	$\frac{3}{4}$
R_3	0	0	$\frac{1}{4}$	1	$-\frac{5}{2}$	0	$\frac{5}{4}$
R_4	**0**	**0**	$-\frac{3}{4}$	0	$-\frac{1}{2}$	**1**	$-\frac{3}{4}$
f	0	0	$\frac{1}{4}$	0	$\frac{1}{2}$	0	$f + \frac{13}{4}$
$\frac{c_i}{-a_i}$	N/A	N/A	$\frac{1}{3}$	N/A	1	N/A	N/A

CUTTING PLANE METHOD

- By Selecting the row R_4 with the negative b value. Computing the $c_i/-a_i$ ratio for the negative elements in the row leads to the choice of s_1 column.
- Pivoting on −3/4 in row R_4, obtain the final Simplex tableau

	x_1	x_2	s_1	s_2	s_3	s_4	b
R_1	0	1	0	0	1	0	2
R_2	1	0	0	0	0	$\frac{5}{3}$	1
R_3	0	0	0	1	$-\frac{8}{3}$	$\frac{1}{3}$	1
R_4	0	0	1	0	$\frac{2}{3}$	$-\frac{4}{3}$	1
f	0	0	0	0	$\frac{4}{3}$	$\frac{1}{3}$	$f+3$

- **The optimal values : $x^*_1 = 1$, $x^*_2 = 2$, $f^* = -3$.**

Evolutionary Algorithms

- ***Evolutionary algorithms*** are a popular choice for discrete optimization because of their ability to work directly with discrete search spaces.
- The application of Evolutionary algorithms is presented in **Chapter 19.**

GENETIC ALGORITHMS

- Genetic Algorithms (GAs) are a family of computational algorithms inspired by the principles of evolution described in Darwin's theory.

- GAs were first conceived by J.H. Holland in 1975.

- The GA uses a population of solutions, whose individuals are represented in forms of chromosome. The individuals in the population go through a process of simulated evolution to obtain the global optimum.

- GAs repeatedly modify a set of solutions or individuals in the course of its entire run. At each iteration, the genetic algorithm selects individuals from the current population to be parents based on certain criteria. The parents are then used to produce the next generation of individuals, called children. Over successive generations, the population evolves toward an optimal solution.

- ***Binary coded genetic algorithms are particularly popular choices for discrete optimization because of their ability to deal directly with discrete search spaces.***

SIMULATED ANNEALING

- Simulated annealing is an effective algorithm that can be used to solve problems in which the number of candidates increases exponentially simulated annealing.

- The concept is based on the manner in which liquids freeze or metals recrystallize in the process of annealing.

- The algorithm mimics the metallurgical process of annealing: heating a material and slowly lowering the temperature to decrease defects, thus minimizing the system energy.

- At the beginning of this method, the initial state is similar to a thermodynamic system with its energy and temperature. At each iteration of the algorithm, a new point is randomly generated.

- The distance of the new point from the current point, or the extent of the search, is based on a probability distribution. The scale of the distribution is proportional to the temperature.

SIMULATED ANNEALING

- The algorithm accepts all new points that lower the energy, but also, with a certain probability, points that raise the energy.
- By accepting points that raise the energy, the algorithm avoids being trapped in local minima, and is capable of global exploration for better solutions.
- An annealing schedule is selected to systematically decrease the temperature as the algorithm proceeds. As the temperature decreases, the algorithm reduces the extent of its search to progressively converge to a minimum.
- Simulated annealing has been used in various combinatorial optimization problems and has been particularly successful in circuit design problems.

Software Options

- Some of the popular software tools available for solving integer programming problems are:

1. The **XPRESS** suite of optimization algorithms is distributed by Dash Optimization.
The *XPRESS MIP* tool provides capability to solve mixed-integer programming problems by using sophisticated implementations of branch and bound and cutting plane algorithms.

2. **CPLEX solvers**, created by CPLEX Optimization, Inc., are designed to handle large scale problems with mixed-integer programming features. The solver uses a combination of algorithms and heuristics.

Basic Solution Approaches

3. **Excel and Quattro Pro Solvers**, developed by Frontline systems, are available to solve small scale integer programming problems using Excel spreadsheets.
The integer programming method employs a branch and bound technique, which uses a nonlinear programming tool known as GRG2 .

4. **The NEOS Server** is a website hosted by Argonne National Laboratory, and is free to use. several state-of-the-art algorithms in optimization software, including those in integer programming, are available from this website (http://neos.mcs.anl.gov/neos/solvers/index.html).