MATHEMATICAL MODELING

Methods and Application

Man VM. Nguyen and Khuong A. Nguyen

Mahidol University

MATHEMATICAL MODELING

Methods and Application

Man Van Minh Nguyen(†)

Khuong An Nguyen(‡)

(†) Faculty of Science - Mahidol University

(‡) Faculty of Computer Science & Engineering - HCMC University of Technology

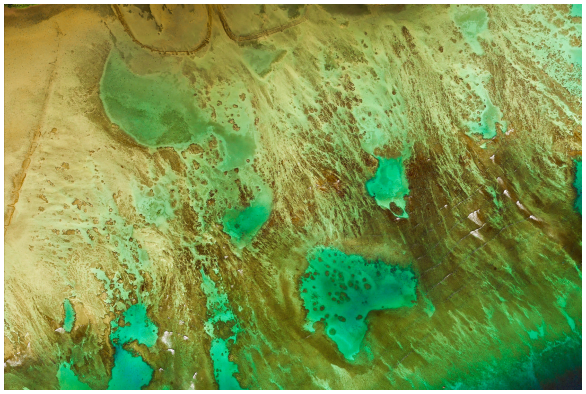
Contents

Index 211

Chapter 3

INTEGER OPTIMIZATION

Models and Practical Usages



Goals and Learning outcomes: To introduce learners how to

a) obtain practical experiences in **modeling** linear constraints/objectives,

b) use a broader class of linear programming models in **various contexts**.

After studying this lecture, you should be capable of:

- **Transferring** a practical optimization problem to a linear program
- **Describing** components and concepts of a linear program
- **Formulating** some specific linear models in

  * Diet problem- Set covering problem

  ** Multi-phase Production and Inventory problem

Lecture's outline- PART I: FUNDAMENTAL THEORY

We will briefly describe some practical instances of LP modeling.

- **A. General Formulation**
- **B. Views of linear programming** - The Simplex method

* Graphical view - Algebraic view - Mathematical foundation * The Simplex method

- C. Diet problem - The Simplex method with LP tableau
- D. Computational Simplex Method

Guidance for team presentations/reports on Week x, Month y, Year z

PART II: MODELS

- E. Practical Modeling in Daily life

- Set covering problem (Model 1), Production and Inventory (Model 2)

- F. Models for Integer Optimization- IO
- G. IP modeling with binary variables- BIP | H. Transportation models with ILP

PART III: MATHEMATICAL TREATMENT

- I. Branch-and-Bound Method
- J. CHAPTER REVIEW: Popular problems revisited
- K. Practical Models in Real World & Demonstration: Solving Sudoku [Extra reading]

PART I: FUNDAMENTAL THEORY

3.1 A. General Formulation

- **History** of linear programming (LP)- Founder
- **Standard form** and Canonical form of a Linear Programming Model
- **Assumptions** of linear programming

B. Views of linear programming - The Simplex method

Geometrical views: convex sets and bounded sets

The feasible region of a LP - *Key theorem*

The Simplex Method (Intuition)

History of LP - Prof. George B. Dantzig

- Found by Prof. George B. Dantzig (1914- 2005), a great American mathematical scientist, his

work in LP is a landmark event in the history of mathematics. Its applications brought our ability to solve general systems of **linear constraints** (including *linear equations, inequalities*) to a state of completion.

- The LP is exploited for modeling real world decision-making problems, in all areas of science and engineering.

Application range of Linear Programming (LP)

Agriculture: planing various crop plants in plantations [Model 1]

Manufacturing and Logistics: transport of raw materials and products among factories

Military science and defense seen from WW2 when the Allies fought the Nazi.

Urban management: distribute district/precinct police stations in cities, see Section 3.5.1

Large-scale Industry: multi-phase Production with Inventory-type constraints

Actuarial and Investment science: Capital Budget planning

Transportation science and traffic engineering , see Section 3.8.

Why LP is so popular? It is the simplicity of both the model's objective function and constraints.

What is a Linear Programming model?

The LP model is suitable for modeling a real world decision-making problem if

- All the **decision variables** $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ are continuous variables.
- There is a single **objective function** $Z = f(\boldsymbol{x})$ that is required to be optimized.
- The objective function Z and all the constraints functions defining the **constraints** in the problem are linear functions of the decision variables.

♦ **EXAMPLE 3.1.** *For a problem with $n = 2$ decision variables $\boldsymbol{x} = (x_1, x_2)$, we want to*

Maximize $Z = f(\boldsymbol{x}) = 2x_1 + 4x_2$ subject to

$$\begin{cases} x_1 + 5x_2 \leq 80; \quad 4x_1 + 2x_2 \geq 20 \\ x_1 + x_2 = 10; \qquad x_1 \geq 0,\ x_2 \geq 0. \text{ [Do we have } m = 5 \text{ constraints?]} \end{cases}$$

The general form of a LP is defined by an objective and linear constraints, given by:

$$P_0 : \text{Maxinize / Minimize } z = f(x_1, x_2, \cdots, x_n) = \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \qquad i \in \{1, 2, \ldots, m\}$$

$$x_j \geq 0 \quad (\text{or } x_j \leq 0 \text{ or unrestricted in sign}), \qquad j \in \{1, 2, \ldots, n\}.$$

- Since LP problems cannot be *solved symbolically* - i.e., the parameters a_{jj}, b_i in constraints and

c_j in objective function must be real numbers)- any computation in software must be done in the real set $\mathbb{R}$.

- Hence for a given LP P_0, any **feasible solution** $\boldsymbol{x}$ is a vector (point) in the n- dimension space $\mathbb{R}^n$; and

  the feasible region is a subset of the space $\mathbb{R}^n$.

- From now on, concepts are generally defined in n-dimensional space $\mathbb{R}^n$.

  E.g. points $\boldsymbol{x} \in \mathbb{R}^n$ and feasible regions $F \subset \mathbb{R}^n,\ n \geq 2$.

Standard form of a LP

- Standard form (some constraints are equality)

$$LP: \quad \min_{\boldsymbol{x}} Z = c^T \cdot \boldsymbol{x} = \sum_{j=1}^{n} c_j\, x_j$$

$$\text{such that (s. t.)} \quad A\boldsymbol{x} \leq \boldsymbol{b}$$

where $A = [a_{ij}]$ is a coefficient matrix, with size $m \times n$, coefficients $a_{ij} \in \mathbb{R}$,

and $\boldsymbol{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n_+$ is the decision vector, $x_j \geq 0$. [1]

Here the linear system $A\boldsymbol{x} \leq \boldsymbol{b}$ consists of m constraints

$$\sum_{j=1}^{n} a_{ij} x_j \leq b, \quad i = 1, 2, \ldots, m.$$

Canonical form of a LP applied when all constraints are equality

$$LP: \quad \min_{\boldsymbol{x}} Z = c^T \cdot \boldsymbol{x} = \sum_{j=1}^{n} c_j \, x_j$$

$$\text{s. t. } A\boldsymbol{x} = \boldsymbol{b}$$

where $A = [a_{ij}]$ is a coefficient matrix, with size $m \times n$, and

$\boldsymbol{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n_+$ is the decision vector [of decision variables $x_j \geq 0$].

The matrix equation $A\boldsymbol{x} = \boldsymbol{b}$ means that all constraints (the rows of equation) are stated as

[1] $\mathbb{R}_+ = [0, +\infty)$, $\mathbb{R}^n_+$ is called the non-negative **orthant**, also a *convex cone*.

equalities $\sum_{j=1} a_{ij}\, x_j = b_i$, except for the non-negativity of decision variables.

Assumptions of linear programming with an objective function

$Z = c^T \cdot \boldsymbol{x} = \sum_{j=1}^{n} c_j\, x_j$ and $A\boldsymbol{x} \leq \boldsymbol{b}$ is a constraint set.

Proportionality assumption:

- The contribution of each activity x_j to the value of Z is proportional to the level of the activity x_j, as represented by the $c_j\, x_j$ term in Z.
- The contribution of each x_j to the left-hand side of each *functional constraint* is proportional to the level of x_j, as represented by the $a_{ij}\, x_j$ term in the constraint.

Additivity assumption: Every function in a linear programming (LP) model is the sum of the individual contributions of the respective activities.

Divisibility assumption: Variables $x_j \geq 0$ are allowed to have any values that satisfy the functional constraints.

NOTE: Degeneracy of canonical form

If $m < n$, [the number of variables is large than the number of constraints] the system $A\boldsymbol{x} = \boldsymbol{b}$ will have an **infinite number of solutions**, see the below.

Lemma 3.1

If *the system* $\mathbf{A}\ \boldsymbol{x}\ =\ \boldsymbol{b}$ *is consistent [i .e., the rank of the coefficient matrix,* $\mathrm{rank}(\boldsymbol{A})$*, is equal to the rank of the augmented matrix,* $\mathrm{rank}(\boldsymbol{A}|\ b)$*]*

and ***if*** $\mathrm{rank}(\boldsymbol{A}) < n$*,*

then *the system has an infinite number of solutions.* ♡

- George B. Dantzig firstly solved this problem using basic solution set with *slack* and *surplus*

variables (see Reminder or Chapter 2).

- Fortunately, MAPLE or LINGO takes care of degeneracy with the `simplex package`, we no need to introduce the artificial slack, surplus.

3.2 B. Geometric views of linear programming

`OUTLINE`

- Mathematical foundation with Geometrical view: From convex sets to bounded sets

  Key point: convexity and bounded-ness (finite-ness) are different concepts.
- The feasible region of a LP - **Theorem 1.1**
- Planes-edges-extreme points, a ternary relation in a feasible region
- **The Simplex Method** (Intuitive version)

3.2.1 *Mathematical foundation*

Definition 3.1

*A set F is said to be **convex** if, for any two points $\boldsymbol{p}_1, \boldsymbol{p}_2 \in F$, the line segment joining the two points lies entirely within the set, i.e., any point*

$$\boldsymbol{u} = \lambda \boldsymbol{p}_1 + (1 - \lambda)\boldsymbol{p}_2 \in F \quad \textit{for } 0 \leq \lambda \leq 1.$$

- Intuitively, a convex set cannot have any "holes" in it and its boundaries are always "flat" or "bent away" from the set, as we will see in the current example. The polygon $OABCD$ is convex.

Is the feasible region of a LP *convex*?

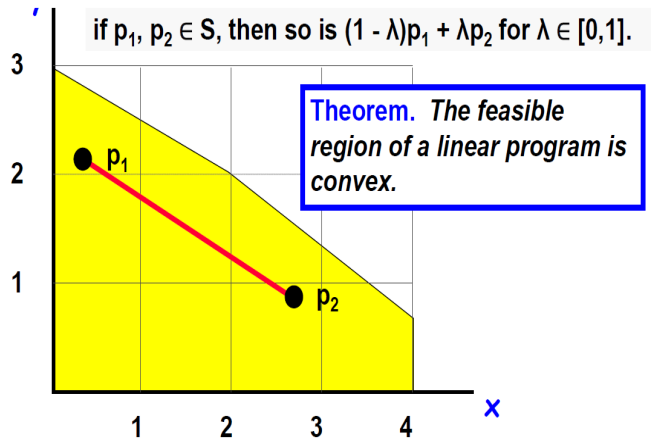


Figure 3.1: *1. What is the solution space S of a feasible LP?*

Theorem 3.1

The feasible region of a given linear programming model is convex. ♡

Proof. If the feasible region $F \neq \emptyset$ then clearly any feasible vector $\boldsymbol{x} \in F$ fulfills the matrix constraint $A\boldsymbol{x} = \boldsymbol{b}$ (Canonical form) [or $A\boldsymbol{x} \leq \boldsymbol{b}$ (Standard form)].

When $\boldsymbol{p}_1, \boldsymbol{p}_2 \in S$- the solution space - they are feasible vectors, we get both $A\boldsymbol{p}_i = \boldsymbol{b}$, $i = 1, 2$. Can you see the point $\boldsymbol{u} = \lambda\boldsymbol{p}_1 + (1 - \lambda)\boldsymbol{p}_2$ also satisfies $A\boldsymbol{u} = \boldsymbol{b}$, if parameter $0 \leq \lambda \leq 1$? Explain in details. □

From convex sets to bounded sets?

Definition 3.2

An optimization problem is unbounded if, *for any feasible solution* $\boldsymbol{x}$, *there is another feasible solution* $\boldsymbol{x}^*$ *whose value is better than the value of* $\boldsymbol{x}$.

- *If the problem is a maximization problem with an objective function* $Z = f(.)$, *we get* $f(\boldsymbol{x}^*) > f(\boldsymbol{x})$: *the function value at* $\boldsymbol{x}^*$ *is greater than the function value at* $\boldsymbol{x}$.
- *if it is a minimization problem, the function value of* $\boldsymbol{x}^*$ *is less than that of* $\boldsymbol{x}$: $f(\boldsymbol{x}^*) < f(\boldsymbol{x})$ ♣

We say that an **unbounded** optimization problem **has no finite optimal solution**.

Proposition 3.1 ♠

1. The feasible region of a linear program (LP) always is **convex**, either bounded convex or unbounded convex. [Figure 3.2 shows a bounded and unbounded set.]
2. Additionally, if the **convex feasible region** corresponding to the linear programming problem is nonempty, it must have at least one extreme (corner) point.

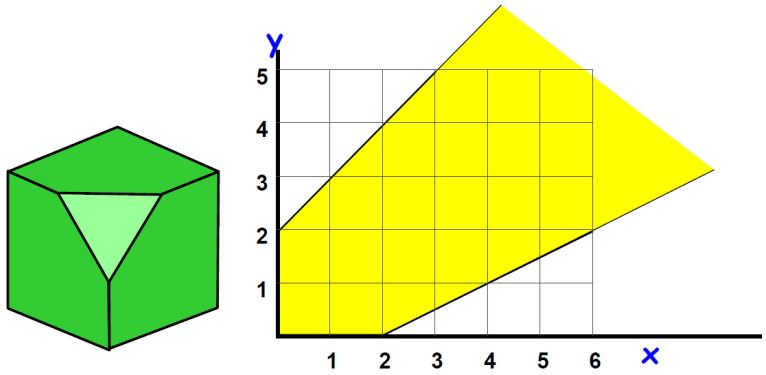


Figure 3.2: *2. Bounded convex and unbounded convex sets*

3. More importantly, optimality implies extreme points exist, that is

   if there is a **finite optimal solution** to the linear program, then this optimal solution must be an extreme point of the feasible region.

   [But, extreme points (corner points) are not necessarily optimal solutions.] [2]

PRACTICE 3.1. *Suppose an LP has a feasible solution. Which of the following is not possible?*

[2] **Ref.** G. Hadley. *Linear Programming*. Addison-Wesley, Massachusetts, 1962.

1. *The LP has no corner point.*
2. *The LP has a corner point that is optimal.*
3. *The LP has a corner point, but there is no optimal solution.*
4. *The LP has corner points and an optimal solution, but no corner point is optimal.*

3.2.2 *Concepts from High-dimensional Geometry*

Planes-edges-extreme points of a feasible region

In 2D an edge of the *feasible region* F of a LP is one of the line segments making up the boundary of the feasible region. It is a (bounded) equality constraint.

In 3D case: We see a a ternary relation, among planes, edges and extreme points, in a feasible region of a LP.

- A **face** P is a flat region of the feasible region, its defining equation is $ax + by + cz + d = 0$.
- The **edges** are where the faces intersect with each other.

  * An **edge** d of the feasible region is one of the line segments making up the framework (skeleton)

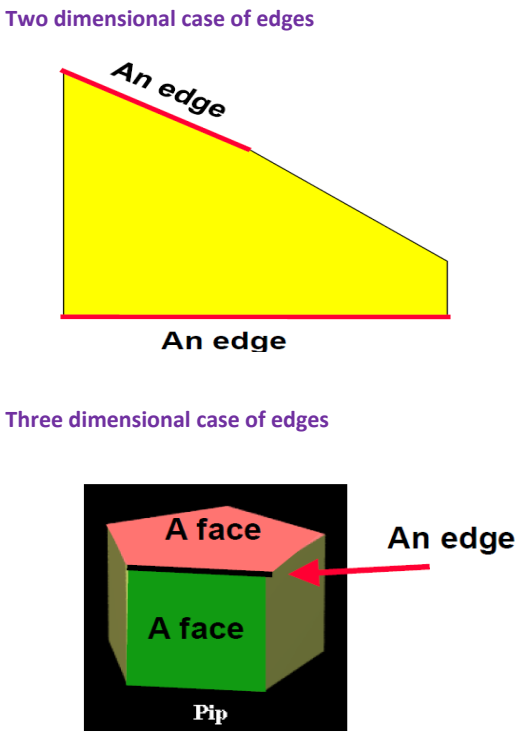


Figure 3.3: *3. Edges in 2D plane and 3D space - [**Courtesy:** MIT OCW]*

of a **polyhedron**: d is formed by a bounded intersection of two faces P_1, P_2 (equality type constraints), as a result it is determined by two defining equations of P_1, P_2.

Plane, Edge and Polyhedron in n-D space, $n \geq 3$

- In 3D, any **plane** has the form $ax + by + cz + d = 0$ where $a, b, c \in \mathbb{R}$, and $a^2 + b^2 + c^2 \neq 0$.

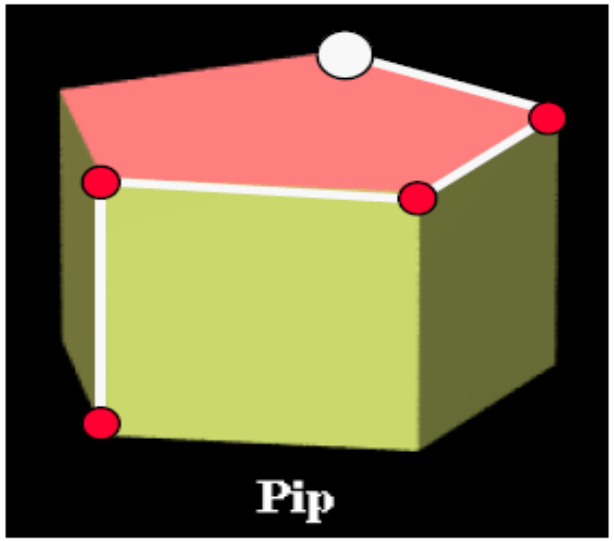
  In **higher-than-3D** space, planes are called **hyper-planes**, having equation

$$a_1x_1 + a_2x_2 + \ldots + a_{n-1}x_{n-1} + a_nx_n + a_0 = 0,\ n > 3.$$

- A **polyhedron** in n-D space (generalized from polygon in 2D, 3D) is a bounded set formed by at least $n + 1$ un-parallel hyper-planes.

  [Un-parallel means linearly independent.]

- An **edge of polyhedron** (polyhedral set) $S \subset \mathbb{R}^n$ is the set of solutions formed by the intersection of $(n - 1)$ linearly independent defining hyperplanes.



In 3D space, a linear constraint with equality gives a **plane**, and two un-parallel planes cutting each other give us a **line**.

A **polygon** is a bounded solid body made by at least 4 planes. An **edge** is a bounded line, and an extreme point (at the corner of the polygon) is the intersection of 2 lines. In other words, each **extreme point** of a convex set is the intersection of 3 planes.

Figure 3.4: *4. Geometric view of the feasible region*

Definition 3.3

`Extreme points - Corner point of a convex set` F ♣

A point $\boldsymbol{x} \in F$ is **corner point** if it is the intersection of 2 lines (edges) [in 2D], or 1 line and 1 face [in 3D].

A point $\boldsymbol{x} \in \mathbb{R}^n$ generally is said to be an **extreme point** or **corner point** of F

if and only if there do not exist other points $\boldsymbol{x}_1, \boldsymbol{x}_2$ $(\boldsymbol{x}_1 \neq \boldsymbol{x}_2)$ in F such that

$$\boldsymbol{x} = \lambda \boldsymbol{x}_1 + (1 - \lambda)\boldsymbol{x}_2 \quad \text{for } 0 \leq \lambda \leq 1.$$

Other words, the line segment joining any two points of a convex set F never passes through an extreme point $\boldsymbol{x}$. Intuitively, an extreme point cannot be “between” any other two points of the set.

QUIZ: Is the claim `"Only convex sets with boundaries formed by` **`linear constraints`** `have extreme points."` TRUE or FALSE?

HINT: **Does a circle in 2D plane have extreme points?**

3.2.3 *The (Basic) Simplex Method*

We start from a linear programming (LP) model, knowing that its feasible region is a convex set F. Therefore, F is identified by a polyhedral set (polyhedron) $S \subset \mathbb{R}^n$, so w.t.l.g. we set $F = S$.

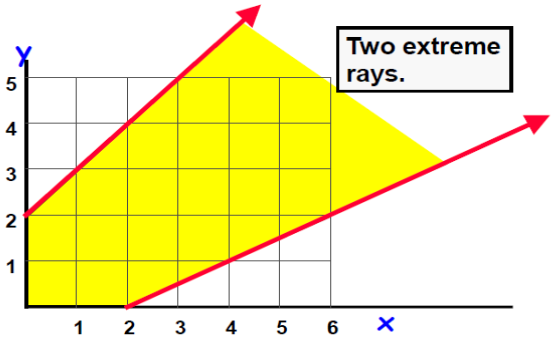
Definition 3.4

Assume further that our feasible region F has some corner (extreme) points. ♣

- `An extreme ray` is like an edge, but it starts at a **extreme point** and goes on infinitely. (Fig. 3.5.)
- `Adjacent Extreme Points` mean closest neighbors in space.

  Mathematically, two extreme points of a polyhedral set $S \subset \mathbb{R}^n$ are **adjacent** if the *line segment* joining them is an edge of S. Equivalently, two extreme points of S are **adjacent** (closest neighbors) if there are $n - 1$ linearly independent defining hyperplanes of S that are **active** at both extreme points (i.e. each constraint equality happens at them).

a) Extreme rays



b) Using extreme rays

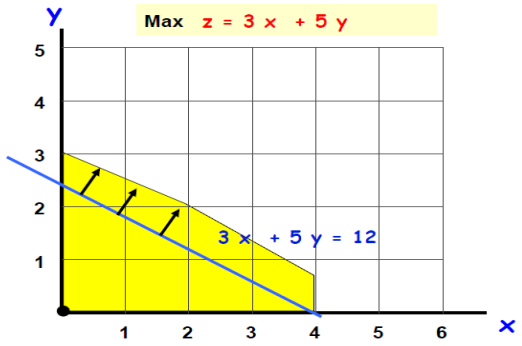


Figure 3.5: *5. Extreme arays and usages*

ELUCIDATION

- In 2D plane $\mathbb{R}^n$, two extreme points are **adjacent**

when there is $n - 1 = 1$ defining line passing through them.

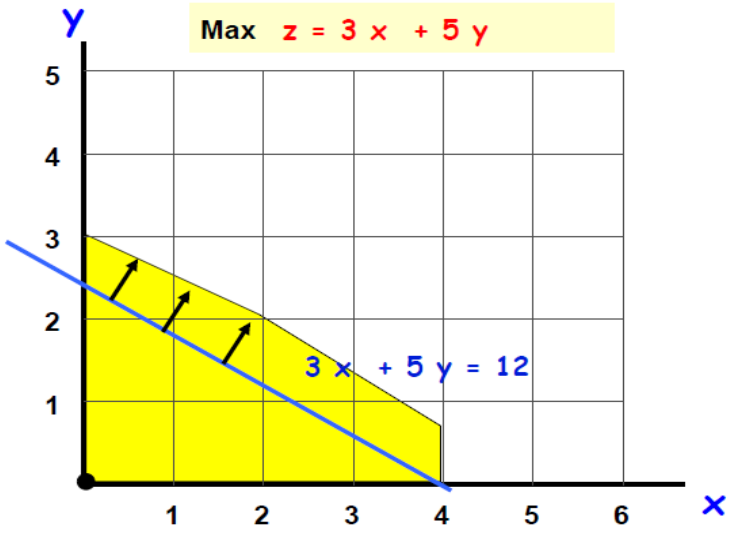
- When $n = 3$, in 3D space, two extreme points are **adjacent**

  when there are $3 - 1 = 2$ **defining planes cutting together to give an edge**,

  and that edge passes through the two extreme points.

- The moving feasible direction of extreme ray make *contours*. The moving direction is determined

  by the normal vector of our **objective function**. (See Figure 3.6.)



The objective function z = 3x + 5y is transformed to the standard form 3x + 5y – z = 0, its graph is called contour.

Hence our contour (**blue line**) has

the slope a = -3/5 or equivalently determined by

the normal vector n = (3,5),

being visualized as black arrows as in the above picture.

Figure 3.6: *6. Iso-objective contour and its feasible moving direction*

Now we take a look at the Intuitive version the Simplex Method.

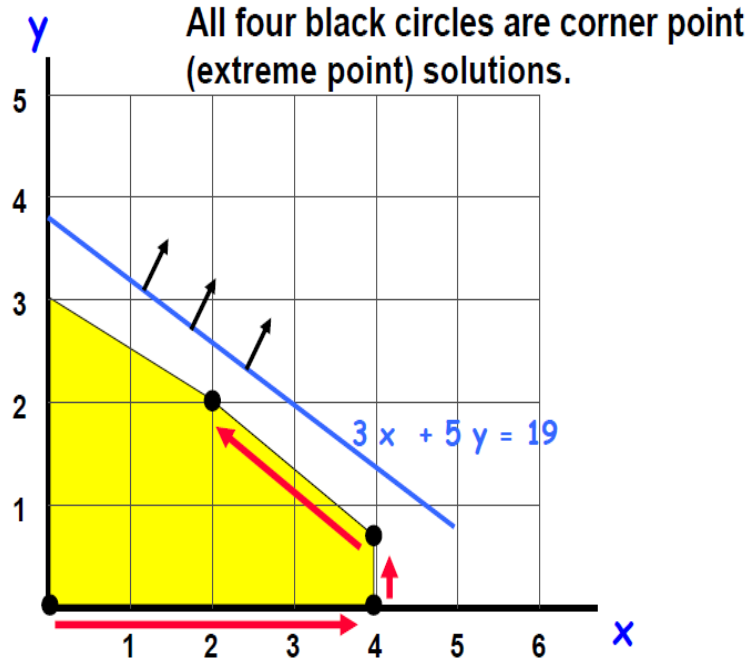
The Simplex Method (Intuition)

1. Start at any feasible **corner point**.
2. Find an edge (or **extreme ray**) at which the objective value is continually improving.

   Go to the **next corner point**.

   (If there is no such corner point, stop. The objective is unbounded.)
3. Continue until **no adjacent corner point** has a better objective value.

- So we start at a corner point. At each iteration, we look for an **adjacent corner point** that is better (called CPF solution). And we stop when there is no improvement. (Figure 3.7.)
- It's one of the nice (but rare) cases in optimization in which you can find **the global optimum** by making *local improvements*.
- But, the algorithm appears more complicated when there are more variables



How many extreme points of the Polygon?

Figure 3.7: *7. Extreme points*

Homework

Indicate graphically whether each of the following linear programs has a *feasible solution*. Determine the *optimal solution*, if one exists, or show that none exists.

- Maximize $z = x + 2y$ subject to

$$\begin{cases} x - 2y \leq 3; & x + y \leq 3; \\ x \geq 0, & y \geq 0. \end{cases}$$

- Maximize $z = x_1 + x_2$ subject to

$$\begin{cases} x_1 - x_2 \leq 2; & x_1 - x_2 \geq -2; \\ x_1 \geq 0, & x_2 \geq 0. \end{cases}$$

3.2.4 *Steps of Mathematical Modeling process*

This guideline of 7 steps will be generally applied for all subsequent parts of the text, for both discrete and continuous mathematical modeling.

1. **Define** the problem.
2. **Collect** and summarize data.
3. **Formulate** a mathematical model, including selection of one or more suitable *decisions*.
4. **Derive** solutions from the model - (usually a *computer-based* procedure).
5. **Verify** the model and refine it as needed.
6. **Prepare** for the ongoing application of the model as prescribed by management, and present the results to the organization.
7. **Implement** the model and update it over time.

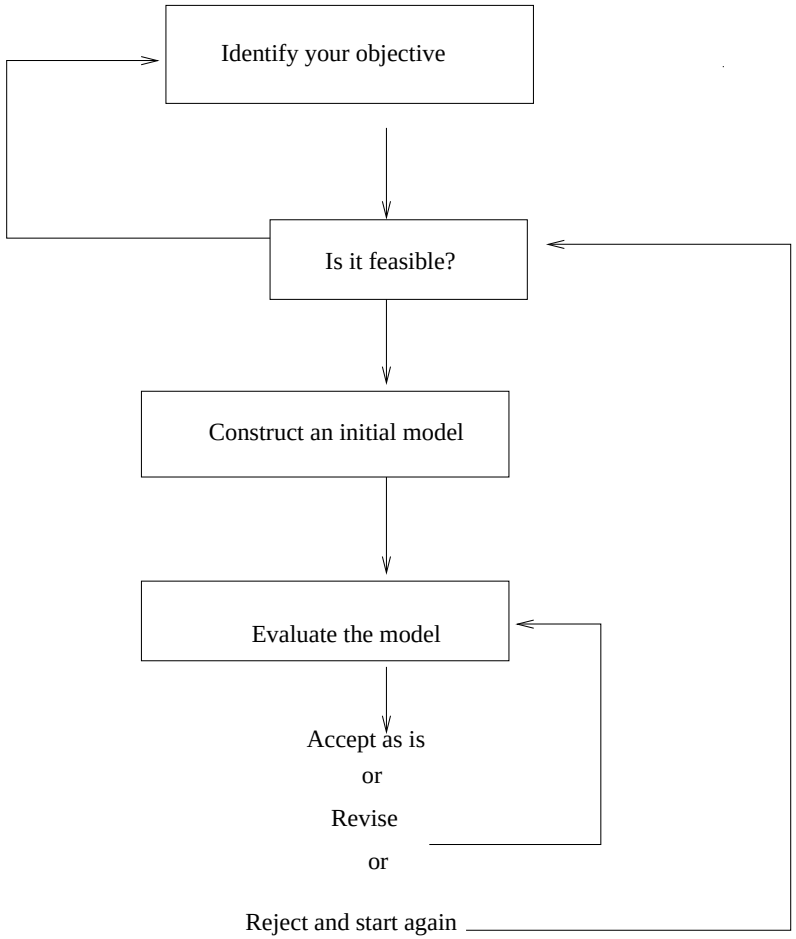


Figure 3.8: *Major steps in Modeling Process*

WHAT NEXT

Part C/ in Section 3.3 shows the basic **Simplex Method** illustrated in details via the Diet problem. Part D/ in 3.4 presents the Computational Simplex Method.

Part E/ describes guidelines for Real-life Modeling with LP and ILP via examples, and Part F/ provides typical problems/models being solvable by the LP- ILP methodology. All such practical problems are designed for self-working individually or in mini-project with programming in small size teams.

3.3 C. Basic Simplex Method via Diet problem

Diet problem - OBSERVED DATA

Table 3.1: *(Prices are approximate)*

	Hamburger H	Big Mac B	McChicken M	Caesar Salad C	French fries R
Total Calories	250	770	360	190	230
Fat Calories	81	360	144	45	99
Protein (grams)	31	44	14	27	3
Sodium (mg)	480	1170	800	580	160
Cost	$1.00	$3.00	$2.50	$3.00	$1.00

OBJECTIVE: **Minimize the cost Z of a meal**

Diet problem- Formulation of Model on dieting

Suppose that we want to design a good 1 week diet at McDonalds.

What would we do? What **data** would we need?

We firstly determine an objective function Z and meaningful constraints.

Minimize the cost Z of a meal with conditions for each burger

- just a few choices listed
- between 600 and 900 calories (a new variable F for calories)

- less than 50% of daily sodium,
- at most 40% of the total calories are from fat (calories)
- at least 30 grams of protein,

- with sodium limit is 1150 mg per day, and
- finally *fractional* meals permitted.

Our original LP : **Diet problem Model**

We need 5 key decision variables $H,\ B,\ M,\ C, R$, one extra variable F to get the standard form below

$$\text{min cost } Z = H + 3\,B + 2.5\,M + 3\,C + R$$

s.t.

$$
\begin{cases}
250H+ & 770B+ & 360M+ & 190C+ & 230R & -F & = 0 \\
 & & & 600 \leq & F & \leq 900 & \\
81H+ & 360B+ & 144M+ & 45C+ & 99R & -0.4\,F & \leq 0 \\
31H+ & 44B+ & 14M+ & 27C+ & 3R & & \geq 30 \\
480H+ & 1170B+ & 800M+ & 580C+ & 160R & & \leq 1150 \\
H, & B, & M, & C, & R & & \geq 0
\end{cases}
$$

Definition 3.5 (*Slack* and *surplus* variables)

Given any inequality constraint $\sum_{j=1}^{n} a_{ij}x_j \leq b$, we can calculate

the difference, or **slack**, s_l between the right- and left-hand sides of the inequality as

$$s_l = b - \sum_{j=1}^{n} a_{ij}x_j.$$

If we have the constraint $\sum_{j=1}^{n} a_{ij}x_j \geq b$, we can calculate

the difference, or **surplus**, s_g between the left- and right-hand sides of the inequality as

$$s_g = \sum_{j=1}^{n} a_{ij}x_j - b.$$

The Simplex method- LP tableau

Since the simplex method starts from **canonical** form (all constraints have '='), we must use slack and surplus variables, and obtain an **LP tableau** (from our original LP) that has properties:

- All variables are non-negative
- All other constraints are "$=$" **constraints**; we write $Z = f()$ to the form

$$Z - f(x_1, \ldots, x_n) = Z - \boldsymbol{c}^t \ \mathbf{x} = 0$$

Finally, we rewrite symbolically the LP in block matrix form, named *LP tableau*

$$\begin{pmatrix} Z & \mathbf{x} & \\ -- & --- & -- \\ 0 & A & b \\ 1 & -\boldsymbol{c}^t & 0 \end{pmatrix}$$

here A is the coefficient matrix, $b \in \mathbb{R}$, and $\boldsymbol{c}^t = (c_1, c_2, \cdots, c_n)$ is the coefficients of $x_1, \ldots, x_n$ in Z.

LP tableau for minimization problem

We will use few new variables:

the **slacks** have form $s_1 = b - \sum_{j=1}^{n} a_{ij}x_j$, $s_3 = b - \sum_{j=1}^{n} a_{ij}x_j$, and

the **surplus** has form $s_2 = \sum_{j=1}^{n} a_{ij}x_j - b$.

min cost $Z = f(H, B, M, C, R, F) = H + 3\,B + 2.5\,M + 3\,C + R$

subject to

$$\begin{cases} 250H + & 770B + & 360M + & 190C + & 230R & -F = 0 \\ & & & 600 \leq & F \leq 900 & \\ 81H + & 360B + & 144M + & 45C + & 99R & -0.4\,F + s_1 = 0 \\ 31H + & 44B + & 14M + & 27C + & 3R & -s_2 = 30 \\ 480H + & 1170B + & 800M + & 580C + & 160R & +s_3 = 1150 \\ H, & B, & M, & C, & R, F & \geq 0 \\ -Z + H + 3\,B + 2.5\,M + 3\,C + R = 0 & & & & & \end{cases}$$

Use Matlab: If we use Matlab (or Maple) soft, the syntax is

```
x = linprog(f, A, b, Aeq, beq, lb, ub)
```

where f is the vector of objective function coefficients,

A is the coefficient matrix with inequalities, b is the vector of constant coefficients.

Aeq is the matrix with equalities, beq is the vector of constant coefficients

If we know lower bounds of variables, set `lb =[0,0,0,0,0, 600]` ... Coding

A = [0 0 0 0 0 1

81 360 144 45 99 -0.4

-31 -44 -14 -27 -3 0

480 1170 800 580 160 0]; b = [900 0 -30 1150];

Aeq=[250 770 360 190 230 -1]; beq = 0

Use the objective function $Z = f(H, B, M, C, R, F) = H + 3\,B + 2.5\,M + 3\,C + R$

c = [1 3 2.5 3 1 0]; lb = [0,0,0,0,0,600]; # Solve the linear program.

`x = linprog(f,A,b,Aeq,beq,lb)` $\longrightarrow$ x = 1.1292 0.4126 0 0 0 600.0000

Use LINGO: If use LINGO for the Diet problem we got Opt LP Solution:

$H = 1.13, B = .41$ (*fractional meals*) and Cost = \$2.37.

Opt Integer LP Solution: $H = 1, R = 2$ and Cost = \$3.

PRACTICE 3.2 (Optimality conditions)**.**

What is the optimal objective value for the following linear program: maximize

$$Z = -3x_1 - 4x_2 - 0x_3 + 13 \text{ subject to } x_1, x_2, x_3 \geq 0$$

A. 0 B. 13 C. 20 D. There is not enough information. ■

SUMMARY 1.

A) In the Simplex Method (Intuition) we start at a corner point. At each iteration, we look for an **adjacent corner point** that is better solution (called CPF solution). And we stops when there is no improvement.

- We find an adjacent corner point by moving contours (planes)

$$P : c_1x_1 + c_2x_2 + \ldots + c_nx_n - Z = 0$$

(get from obj. func. $Z = \sum_{j=1}^{n} c_j x_j$) while fixing its feasible direction.

- The feasible direction is determined by the constant normal vector of P:

$$\boldsymbol{n}_P = (c_1, c_2, c_3, \ldots, c_n).$$

It's one of the nice cases in optimization where you find **the global optimum** by making *local improvements*.

B) Mixed Constraints LP needs slack and surplus variables

LP problems with a mixture of $\leq$, $\geq$ and/ or $=$ constraints require special care before they can be solved with the simplex method. While each less-than-or-equal-to constraint can easily be converted to an equality constraint by the inclusion of a **slack variable**- the other constraint types necessitate additional steps.

Using slack and surplus variables:

Consider a program with two variables

$$\text{Maximize } z = 2x_1 + 4x_2$$

$$\text{subject to } \begin{cases} x_1 + 5x_2 & \leq 80 \\ 4x_1 + 2x_2 & \geq 20 \\ x_1 + x_2 & = 10 \\ x_1 \geq 0,\ x_2 \geq 0 \end{cases}$$

Using the graphical method (**not** the simplex method), this problem has the optimal solution where $x_1 = 10, x_2 = 0$ with the minimum value of the objective $z = 20$.

By introducing

* a **slack** variable $x_3 \geq 0$, we can write the first constraint as $x_1 + 5x_2 + x_3 = 80$;

* similarly, using a **surplus** $x_4 \geq 0$ the second constraint becomes $4x_1 + 2x_2 - x_4 = 20$.

Note that the constraint equations form a system of $m = 3$ equations in $n = 4$ unknowns.

Now our system of constraints is fitted into the form $\mathbf{A}\,\boldsymbol{x} = \boldsymbol{b}$, with

$$\mathbf{A} = [a_{ij}] = \begin{bmatrix} 1 & 5 & 1 & 0 \\ 4 & 2 & 0 & -1 \\ 1 & 1 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{b} = (80, 20, 10)^T.$$

Thus, the optimal solution of the optimization problem must be a solution to this set of linear equations. Since $m = 3 < 4 = n$, this system will have an **infinite number of solutions**.

3.4 D. Computational Simplex Method

Definition 3.6 (Basic Solution and Basic Feasible Solutions of a LP model) ♣

Let $S \subset \mathbb{R}^n$ be a *polyhedral set* defined by linear inequality and equality constraints.

1. **Basic Solution**. Solution $\boldsymbol{x}$ is a basic solution if

   (a) $\boldsymbol{x}$ satisfies all equality constraints of S and

   (b) at least n of the constraints of S are active at $\boldsymbol{x}$ and are linearly independent.

2. **Basic Feasible Solution**. If $\boldsymbol{x}$ also satisfies all constraints of S, then x is a basic feasible solution. ■

Lemma 3.2 (See a proof after Theorem ?? and ??)

We have the following. ♡

a/ Suppose that $S \subset \mathbb{R}^n$ be a polyhedral set. Then

$\boldsymbol{x}$ is an extreme point of S if and only if $\boldsymbol{x}$ is a basic feasible solution.

b/ Assume $\emptyset \neq S \subset (\mathbb{R}_+)^n = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x} \geq \mathbf{0}\}$, then S has at least one extreme point.

Moreover, if there is a **finite optimal solution** to the linear program, then this optimal solution must be an extreme point of the feasible region.

NOTES:

- $\mathbb{R}^n_+$ is the non-negative **orthant**, also a *convex cone*.

  A set K is a convex cone if $x, w \in K$ and for any $\alpha, \beta \geq 0$ then $\alpha x + \beta w \in K$.
- Hence, we have two different ways to describe the same idea - one geometric (*extreme point*) and one algebraic (*basic feasible solution*). We use these terms interchangeably throughout the rest of this text.
- Currently, we know we can describe a polyhedral set by its **defining hyperplanes**, and from this we can describe its *extreme points*. Can we describe the set knowing only its extreme points

and directions?

3.4.1 *Fundamental theorem of Linear programming*

Properties of the feasible region of a LP

From our experience with two-variable linear programs, the feasible region appears as

the convex hull of the extreme points, except when it was unbounded.

Due to representation of bounded polyhedral sets, firstly we can express the followings.

1. The feasible region of a linear program always is convex, [Theorem 3.1], and it is either bounded convex or unbounded convex. [Try yourself, or see Lemma **??** in **Section 3J** (Section **??**)-Mathematical Polyhedron Theory for ILP].

2. More importantly, a nonempty and **bounded** polyhedron is the convex hull of its extreme points. [See Theorem **??** in **Section 3J** (Section **??**) for a mathematical induction proof.]

Definition 3.7 ♣

1. Convex combinations of two points:

   Fix two points $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathbb{R}^n$. Their convex combinations are points

   $$\boldsymbol{p} = \lambda\, \boldsymbol{p}_1 + (1 - \lambda)\, \boldsymbol{p}_2$$

   on the line connecting $\boldsymbol{p}_1, \boldsymbol{p}_2$, where $0 \leq \lambda \leq 1$.

2. Convex combinations of $k \geq 2$ points: Let $F = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_k\}$, $\boldsymbol{p}_i \in \mathbb{R}^n$. The Convex Hull $CH(F)$ of F is the set of all convex combinations of points in F, meaning: $CH(F) = \{\boldsymbol{x} : \boldsymbol{x} = \sum_i\ \lambda_i\, \boldsymbol{p}_i, \}$ where the convex condition hold: $\sum_i\ \lambda_i = 1,\ \lambda_i \geq 0.$

For example, in Figure 3.9 if our LP has constraints of the form $A\, \boldsymbol{x} \leq \boldsymbol{b}$, its visualization is a tetraheron T with 4 extreme points say $F = \{O, A, B, C\}$ (at the vertices of the tetraheron).

What is the convex hull $CH(F)$?

One linear constraint only Ax = b, gives us the plane P.

Non-negativity is x_1, x_2, x_3 >= 0, gives us three planes.

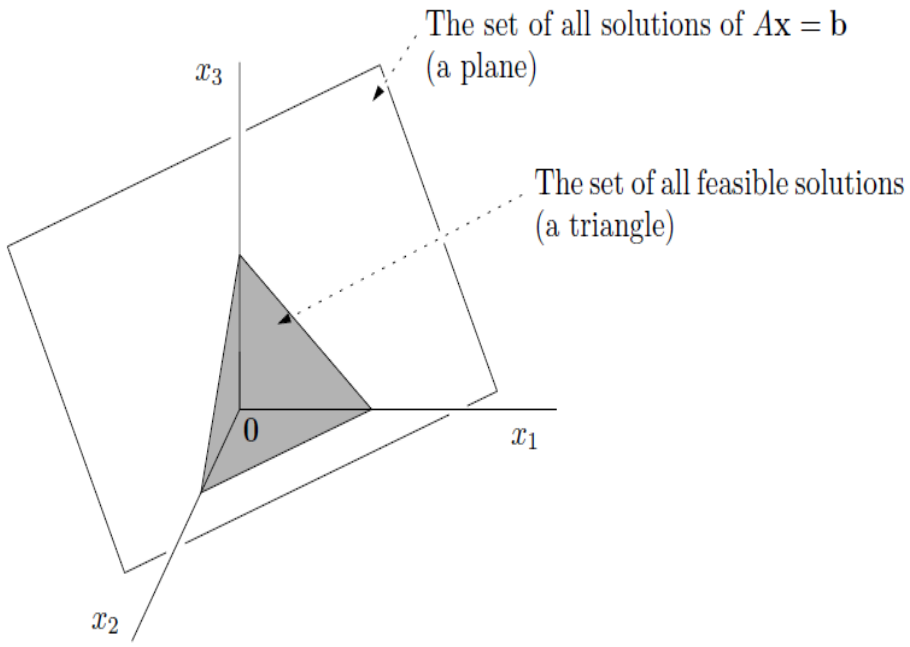


Figure 3.9: *Geometric view of the feasible region when LP has one constraint only*

Representation Theorem

Then the convex hull $CH(F)$ is exactly the **whole tetraheron T (inner and surface)**, consisting of all feasible solutions. We can extend this notion to multidimensional cases through the following.

Theorem 3.2

Representation Theorem for generic linear programs ♡

Let $S \subseteq \mathbb{R}^n$ be a nonempty polyhedral set in the non-negative orthant $(\mathbb{R}_+)^n$.

Then S has at least one extreme point.

Hence, a polyhedral set $\emptyset \neq S \subseteq \{\boldsymbol{x} : \boldsymbol{x} \geq 0\}$ has a finite number of extreme points.

Proof. See details in Theorem **??**. □

E.g., consider the polyhedral set

$$S = \{(x, y) : 1 \leq x + y \leq 3\}.$$

The set S is the region bounded by two parallel lines in $\mathbb{R}^2$ and has no extreme points.

Theorem 3.3 (Fundamental Theorem of Linear Programming (Dantzig)) ♡

Suppose $S \subseteq \{\boldsymbol{x} : \boldsymbol{x} \geq 0\}$ be the nonempty feasible region of some linear program, where

$$z = \boldsymbol{c}^T \boldsymbol{x} = \sum_{j=1}^{n} c_j x_j$$

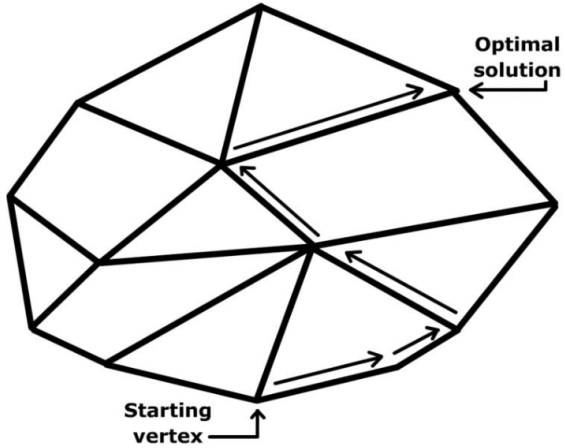
is its objective function. Then

- either z attains its optimal value at some extreme point of S,
- or the linear program is unbounded.

Hence: **If** an optimal solution exists, and there are extreme points, **then** at least one of the optimal solutions must be at an extreme point.

REMINDER [3]

1. **Canonical form of an LP** has obj function $\min_{\mathbf{x}} c^T \mathbf{x}$ and the constraints have the form $A\mathbf{x} = b,\ \mathbf{x} \in \mathbb{R}^n_+$.
2. **Geometry of Simplex Method**: Geometrically, the simplex method **moves from one extreme point (of the LP polyhedron) to one of its *adjacent* extreme point.** Since there are only a finite number of extreme points, the method terminates finitely at an *optimal solution* or detects that the problem is *infeasible* or it is *unbounded*.



[3] COURTESY: Some material in next section of Computational Simplex Method is borrowed from Dr. Le Hong Trang and the staff of CSE faculty (HCMUT-VNUHCM). Readers can see more at `https://math.libretexts.org/`.

3. **In the simplex method** only a finite number of these solutions (known as the basic solution set) will be of interest, and this depends on the coefficient matrix.

Knowledge box 1 (The rank of the coefficient matrix).

If the system $\mathbf{A}\,\boldsymbol{x} = \boldsymbol{b}$ is *consistent* (i .e., the rank of the coefficient matrix, $r(\mathbf{A}) = r(\mathbf{A}|\, b)$, the rank of the augmented matrix) , and

if $r(\mathbf{A}) < n$, then the system has an *infinite* number of solutions.

Computational Simplex Method with a mathematical base

1. Let $S = \{\mathbf{x} \in \mathbb{R}^n |\ A\,\mathbf{x} = b,\ \mathbf{x} \geq 0\}$, the feasible set of LP.

   Since A is full row rank, if $S \neq \emptyset$, then we must have $m \leq n$.

   Without loss of generality, we assume that $m < n$ (the system has an infinite number of solutions).

2. Let decompose matrix $A = (B, N)$, where B is an $m \times m$ matrix with full rank, i.e., $det(B) \neq 0$.

   Then, B is called a *basis*. Let $\mathbf{x} = \begin{pmatrix} \mathbf{x}_B \\ \\ \mathbf{x}_N \end{pmatrix}$. We have reformulation of constraints

   $B\ \mathbf{x}_B + N\ \mathbf{x}_N = b$, where $\mathbf{x}_N$ is called *nonbasic variables*. Setting $\mathbf{x}_N = 0$ gives $\mathbf{x}_B = B^{-1}b$.

   Vector $\mathbf{x} = \begin{pmatrix} B^{-1}b \\ \\ 0 \end{pmatrix}$ is called a *basic solution*. Vector $\mathbf{x}_B$ is called *basic variables*.

   - If the basic solution is also feasible, i.e., $B^{-1}b \geq 0$, $\mathbf{x} = \begin{pmatrix} B^{-1}b \\ \\ 0 \end{pmatrix}$ is said to be a *basic feasible solution*.
   - $\hat{x} \in S$ is an extreme point of S *if and only if* $\hat{x}$ is a basic feasible solution. [Lemma 3.4 a/]
   - Two extreme points are *adjacent* if they differ in only one basic variable.

3. The Fundamental Theorem 3.3 for the LP

$$\min\{z = c^T \mathbf{x} \mid A\mathbf{x} = b,\ \mathbf{x} \geq 0\}$$

means: If S has at least one extreme point and there exists an optimal solution, then z attains its optimal value at some extreme point of S.

4. The feasible set S has at least one extreme point [by Lemma 3.4 b/]. We so claim that the optimal value is either $-\infty$ (unbounded case), or is attained an extreme point (basic feasible solution) of the feasible set.

3.4.2 *A Naive Algorithm for Solving Linear Program*

- Let $\min\{c^T \mathbf{x} \mid A\mathbf{x} = b,\ \mathbf{x} \geq 0\}$ be a bounded (canonical) linear program.
- Enumerate all bases $\mathcal{B} \in \{1, 2, \ldots, n\}$, $\binom{m}{n} = O(n^m)$, too many.

- Compute associated basic solution $\mathbf{x} = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$.

- Return the one which has smallest objective function value among the feasible basic solutions.

Running time is $O(n^m \cdot m^3)$. **Are there more efficient algorithms? The Simplex Method**.

HOW TO DO? First of all, we rewrite symbolically the LP in block matrix form, named *LP tableau* or the initial simplex tableau

$$\begin{pmatrix} \mathbf{x} & Z & \\ -- & --- & -- \\ A & 0 & b \\ -\boldsymbol{c}^t & 1 & 0 \end{pmatrix}$$

where A is the coefficient matrix, $b \in \mathbb{R}$, and

the row vector $\boldsymbol{c}^t = (c_1, c_2, \cdots, c_n)$ keeps the coefficients of variables $x_1, \ldots, x_n$ in cost function Z.

The Simplex Method- Using the LP tableau for Minimization

1. From the initial simplex tableau we will construct a sequence of tableaus of a similar form, by gradually rewriting them according to certain rules.

   * Each tableau will contain the same information about the linear program, only written differently.

   * The procedure terminates with a tableau that represents the information so that the desired optimal solution can be read off directly.

2. Write the objective function as the bottom row.

   Question: Why do we choose the most negative entry in the bottom row?

   The most negative entry in the bottom row identifies the pivot column.

3. Calculate the quotients. The smallest quotient identifies a row.

The element in the intersection of the column identified in above step and the row identified in this step is identified as the pivot element.

4. This process of rewriting one simplex tableau into another is called a **pivot step**.

   In each pivot step some *nonbasic* variable enters the basis, while some basic variable leaves the basis.

   Perform pivoting to make all other entries in this column (leaving column) zero. This is done the same way as we did with the *Gauss-Jordan* method.

Algorithm 3.1 Simplex Algorithm: Steps with objective function $Z - \boldsymbol{c}^t \mathbf{x} = 0$

Simplex Algorithm($\boldsymbol{c}, \mathbf{x}, A, \boldsymbol{b}$)

1. **Initialization**:

   Identify a basic feasible solution $\widehat{x} = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$, with matrix $A = (B, N)$.

   Let $\mathbf{x} \in S$ be any feasible solution of the LP. Let $\mathbf{x} = \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{pmatrix}$ and $c = \begin{pmatrix} c_B \\ c_N \end{pmatrix}$.

2. **Model transformation**:

   The model becomes $\min\ c_B^T \mathbf{x}_B + c_N^T \mathbf{x}_N$, the canonical constraints

   $A\ \mathbf{x} = B\,\mathbf{x}_B + N\,\mathbf{x}_N = b$ and so $\mathbf{x}_B = B^{-1}b - B^{-1}N\,\mathbf{x}_N$. We have

$$\begin{aligned} c^T \mathbf{x} &= c_B^T \mathbf{x}_B + c_N^T \mathbf{x}_N = c_B^T B^{-1} b - \ c_B^T B^{-1} N\,\mathbf{x}_N + c_N^T \mathbf{x}_N \\ &= c^T\,\widehat{\mathbf{x}} + (c_N^T - c_B^T\ B^{-1}\ N)\ \mathbf{x}_N\,. \end{aligned}$$

3. **Constructing reduced cost vector**:

   Let $r_N = (c_N^T - c_B^T B^{-1} N)$, called vector of *reduced costs*,

   giving reduced value $r_N\ \mathbf{x}_N$ (associated with the nonbasic $\mathbf{x}_N$).

4. **Optimality check**:

   If $r_N \geq 0$,

   then $c^T\,\mathbf{x} \geq c^T\,\widehat{\mathbf{x}}$ and the current extreme point $\widehat{\mathbf{x}}$ is optimal with obj. value $Z = c^T\,\widehat{\mathbf{x}}$.

   Otherwise, there must exist an entry $r_j < 0$ in vector r_N,

   giving an improving direction in $\min$ problem, so we can

   let corresponding **nonbasic** variable x_j become a **basic** one $x_j > 0$ (*entering variable*).

5. **Updating basic vector:**

♣ **OBSERVATION 1.** *In theory, it does not matter since the algorithm will stop only when either an optimal solution or an unbounded improving direction is found.*

I) Dantzig's Rule: George Dantzig so suggested a greedy rule for selecting our improving simplex direction.

- If we have a *maximization* problem, we choose the simplex direction whose reduced cost is most positive; if we have a *minimization* problem, we choose the simplex direction whose reduced cost is most negative.
- If the simplex method **does not identify a simplex direction** that is improving, then the current solution is a global optimal solution to the linear program.

Simplex Direction: A simplex direction $\boldsymbol{d}_j$ corresponding to the nonbasic variable x_j is a direction vector where (1) $\boldsymbol{d}_j[j] = 1$, (2) $\boldsymbol{d}_j[k] = 0$ for all other nonbasic variables $x_k \neq x_j$, and (3) the other entries $\boldsymbol{d}_j[i]$ corresponding to the basic variables x_i is (uniquely) determined by $B^{-1}\ \mathbf{a}_j$. [Note $\mathbf{a}_j$ is column j of matrix $A = [B,\ N]$].

The feasible improving direction associated with nonbasic x_j is rewritten as

$$\boldsymbol{d}_j = \begin{pmatrix} -B^{-1}\ \mathbf{a}_j \\ \boldsymbol{e}_j \end{pmatrix},\ \boldsymbol{e}_j \text{ is a unit vector in } \mathbb{R}^m?$$

II) Determining the Optimum Step Size: Once we determined the simplex direction $\boldsymbol{d}_j$ to improve the current **x**, we move in $\boldsymbol{d}_j$ with a step size $\lambda \geq 0$ to the solution $\mathbf{x} + \lambda\ \boldsymbol{d}_j$, so long as all constraints remain satisfied. To the $\max$ problem, select largest λ, to the $\min$ problem, select smallest λ, as $\lambda = \min\left\{\frac{\bar{b}_i}{\bar{a}_{ij}} | \bar{a}_{ij} > 0\right\}$.

3.4.3 *Computational Simplex Method- Step by step*

- Step 0: Compute an initial basis B and the basic feasible $B^{-1}b$ solution $\mathbf{x} = \begin{pmatrix} B^{-1}b \\ 0 \end{pmatrix}$.
- Step 1: if $r_N = (c_N^T - c_B^T B^{-1} N) \geq 0$, **stop**, **x** is an optimal solution. Otherwise **goto** Step 2.
- Step 2: Choose index j satisfying $c_N^T - c_B^T B^{-1}\ \mathbf{a}_j < 0$, if vector $\bar{a}_j = B^{-1}\ \mathbf{a}_j \leq 0$, **stop**, the LP is infeasible.

Otherwise, m-vector $\bar{a}_j = B^{-1}\ \mathbf{a}_j > 0$ **goto** Step 3. []

- Step 3: compute the step size [where $\bar{b}$ is the RHS vector of system]

$$\lambda = \min\left\{ \frac{\bar{b}_i}{\bar{a}_{ij}} |\ \bar{a}_{ij} > 0, \quad i = 1, \ldots, m \right\} = \frac{\bar{b}_r}{\bar{a}_{rj}}$$

Let $\mathbf{x} := \mathbf{x} + \lambda \boldsymbol{d}_j$, where $\boldsymbol{d}_j = \begin{pmatrix} -B^{-1}\ \mathbf{a}_j \\ \boldsymbol{e}_j \end{pmatrix}$. **goto** Step 1.

Simplex Tableau

x_B	x_N	RHS
B	N	b
c_B^T	c_N^T	0

It implies that

x_B	x_N	RHS
I	$B^{-1}N$	$B^{-1}b$
0	$r_N = c_N^T - c_B^T B^{-1} N$	$-c_B^T B^{-1} b = -Z$

3.4.4 *Simplex Method: An example*

- Consider the following LP

$$\min_{x} \quad -7x_1 - 2x_2$$

s.t.

$$-x_1 + 2x_2 + x_3 = 4,$$

$$5x_1 + x_2 + x_4 = 20,$$

$$2x_1 + 2x_2 - x_5 = 7,$$

$$\mathbf{x} \geq 0.$$

- The initial tableau should be

x_1	x_2	x_3	x_4	x_5	RHS
-1	2	1	0	0	4
5	1	0	1	0	20
2	2	0	0	-1	7
-7	-2	0	0	0	0

Simplex Method: example (cont.)

- Choose the initial basis to be $B = (a_1, a_3, a_4)$, we have basic $x_B = (x_1, x_3, x_4)^T$. The simplex tableau is then

x_1	x_2	x_3	x_4	x_5	RHS
0	3	1	0	$-\frac{1}{2}$	$7\frac{1}{2}$
0	-4	0	1	$2\frac{1}{2}$	$2\frac{1}{2}$
1	1	0	0	$-\frac{1}{2}$	$3\frac{1}{2}$
0	5	0	0	$-\frac{7}{2}$	$24\frac{1}{2}$

- The basic feasible solution is $x_B = (x_1, x_3, x_4)^T = (3\frac{1}{2}, 7\frac{1}{2}, 2\frac{1}{2})^T$.
- Since $r_5 = -\frac{7}{2} < 0$, x_5 is chosen entering variable. $\lambda = \frac{2\frac{1}{2}}{2\frac{1}{2}} = 1$, then x_4 is leaving variable. The new basic variable should be $x_B = (x_1, x_3, x_5)^T$.

  The new tableau is obtained as follows.

x_1	x_2	x_3	x_4	x_5	RHS
0	$\frac{11}{5}$	1	$\frac{1}{5}$	0	8
0	$-\frac{8}{5}$	0	$\frac{2}{5}$	1	1
1	$\frac{1}{5}$	0	$\frac{1}{5}$	0	4
0	$-\frac{5}{3}$	0	$\frac{7}{5}$	0	28

- Similarly, choose x_2 as the entering variable, then $\lambda = \min\{\frac{8}{11/5}, \frac{4}{1/5}\} = \frac{40}{11}$. Hence x_3 is leaving variable. The new basic variable should be $x_B = (x_1, x_2, x_5)^T$. Therefore, the tableau is

x_1	x_2	x_3	x_4	x_5	RHS
0	1	$\frac{5}{11}$	$\frac{1}{11}$	0	$\frac{40}{11}$
0	0	$\frac{8}{11}$	$\frac{6}{11}$	1	$\frac{75}{11}$
1	0	$-\frac{1}{11}$	$\frac{2}{11}$	0	$\frac{36}{11}$
0	0	$\frac{3}{11}$	$\frac{16}{11}$	0	$30\frac{2}{11}$

- Since $r_N = \left(\frac{3}{11}, \frac{16}{11}\right) \geq 0$, the current basic feasible solution $x = \left(\frac{36}{11}, \frac{40}{11}, 0, 0, \frac{75}{11}\right)^T$ is optimal with the optimal value is $-30\frac{2}{11}$.

The Simplex Method- Summarized Points for Minimization

1. Convert the inequalities into equations (by adding slack variables).
2. Construct the initial simplex tableau. Write the objective function as the bottom row.
3. **Question**: Why do we choose the most negative entry in the bottom row? The most negative entry in the bottom row identifies the pivot column j (non-basic variable).

4. Calculate the quotients. The smallest quotient identifies a row.

   The element in the intersection of the column identified in step 3 and the row identified in this step is identified as the pivot element.

   Question Why do we find quotients, and why does the smallest quotient identify a row?

   Using the quotients to identify the pivot element guarantees that we do not violate the constraints.

5. **Question** Why do we identify the pivot element?

   The simplex method begins with a corner point and then moves to the next corner point always improving the value of the objective function. After each simplex iteration,

   * the nonbasic variable corresponding to the chosen simplex direction enters the basis and becomes *basic*,

   * any one of the (possibly several) basic variables that define the minimum step size will leave the basis and become *nonbasic*.

   The variable is added is called the *entering* variable, and the variable is being replaced is called

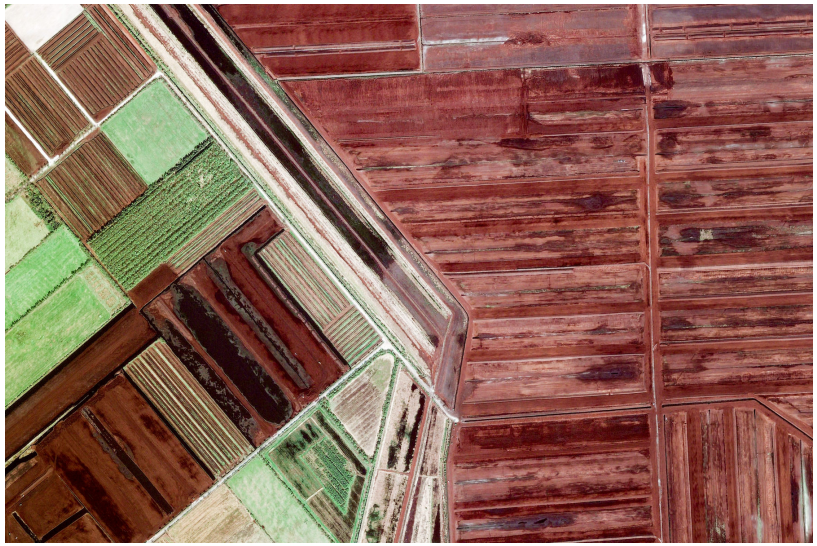
the *departing* variable (identified by the lowest of all quotients.).

6. **Perform pivoting to make all other entries in this column zero**. This is done the same way as we did with the Gauss-Jordan method.

   When there are no more negative entries in the bottom row, we are finished; otherwise, we start again from step 3.

7. **Question**: Why we finish when there are *no negative* entries in the bottom row? Since, all variables are non-negative, the highest value Z is found.

8. Read off your answers, we determine the **basic solution** associated with the *final simplex tableau*. Get the variables using the columns with 1 and 0s. The minimum value appears in the bottom right hand corner.



Guidance for self-reviews

For the year 2022-202x ($x > 2$) the learners of MM text could deal with many questions taken from INTEGER OPTIMIZATION in their midterms and also final exam.

In traditional class (on-site teaching and leaning) students prepare a case study, work out suggested models 1,2,3 provided in next part (and 6 others in Section 3.11). In the last two weeks of the semester, they then will either show a presentation or submit their team report with coding in LINGO, MATLAB, R , Python (your choice).

Software and solvers for new topic of **Stochastic Programming- Stochastic Optimization** in Chapter **??** include GAMS/DECIS, GAMSPy, Stochastic Modeling Interface ... see Section **??** for more info.

3.5 E. Practical Modeling in Daily life

A Blue print

Set covering problem how to serve dweller's safety at cities, **illustrated** in Model 1

Production and Inventory will be illustrated next in Model 2

Resource Allocation in Section 3.10.1

Model 1 and 2 were solved as hints, the readers follow LIY and DIY principles for others.

3.5.1 *Model 1: Set Covering problem- Role of Constraints*

- Selecting a few from a collection of many is a common situation in engineering and services. For example, if we are **forming a committee** to determine recent financial decisions on employee morale, it seems reasonable **to choose at least one employee** from each department.
- **Set Covering Problems**: where the set is the collection of choices from which at least one must be chosen.

Binary variables and covering constraints

- **A binary variable is one that can get only two values: 0 and 1**:

$$x = \begin{cases} 1 & \text{if a choice } k \text{ is selected} \\ 0 & \text{otherwise,} \end{cases}$$

here k belongs to a finite set.

- When we model with **binary variables**, we often need to ensure that at least one variable from a collection has value 1.
- Each of these binary cases appear so often that they have specific names associated with them:

(1) **covering** constraints,

(2) **packing** constraints, and

(3) **partitioning** constraints.

Now we apply to the Set Covering problem.

If given a collection C of choices, where we define variable

$$x_k = \begin{cases} 1 & \text{if choice } k \text{ is selected} \\ 0 & \text{otherwise,} \end{cases}$$

a **covering constraint** is of the form $\sum_{i \in C} x_i \geq 1;$

a **packing constraint** is of the form $\sum_{i \in C} x_i \leq 1;$

and a **partitioning constraint** is of the form

$$\sum_{i \in C} x_i = 1.$$

Example - police stations of town

- A local community is looking to put precinct police stations throughout the city. The administration has divided the city into **10 districts** and has designated a list of *6 candidate locations* for **the stations**. Each station can serve a small number of the districts, as described in the table below.

Station	Districts Served
1	1, 3, 5, 6
2	1, 4, 7, 9
3	2, 5, 8
4	2, 3, 4, 7
5	4, 6, 10
6	8, 9, 10

Apply to police stations of a city we have a variable x_i defined for each station i with values

$$x_i = \begin{cases} 1 & \text{if station } i \text{ is selected} \\ 0 & \text{otherwise,} \end{cases}$$

Once we have these variables, the constraints seem straightforward. For example, to service

district 1, either station 1 or station 2 must be open, giving the constraint $x_1 + x_2 \geq 1$.

District 4 can be serviced by station 2, 4, or 5, giving the constraint $x_2 + x_4 + x_5 \geq 1$.

Each of these constraints indicate that at least one of the variables must have value 1.

Proposed solution for Model 1: Set covering problem

Want to determine the fewest number of stations to open while still serving all 10 districts. Hence, our objective function is to **minimize the number N of stations opened**: $\min_{\boldsymbol{x}} \sum_j c_j x_j$.

This gives the set covering integer program as follows.

$$x_j = \begin{cases} 1, \text{ if station } j \text{ is selected.} \\ 0, \text{ otherwise.} \end{cases}$$

$$\min_{\boldsymbol{x}} \quad N = c^T \cdot \boldsymbol{x} = \sum_j c_j x_j$$

$$\text{s. t.} \qquad a_j \, x_j \geq b \ \textbf{(covering type constraint)}$$

$$\boldsymbol{x} \in \{0,1\}^6 \text{ or each } x_j \in \{0,1\}, \text{ for } j = 1, 2, \ldots, 6.$$

$$\text{min cost } N = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \; s.t.$$

$$\begin{cases}
x_1 + x_2 \geq 1 & \textbf{District 1} \\
x_3 + x_4 \geq 1 & \textbf{District 2} \\
x_1 + x_4 \geq 1 & \textbf{District 3} \\
x_2 + x_4 + x_5 \geq 1 & \textbf{District 4} \\
x_1 + x_3 \geq 1 & \textbf{District 5 ...} \\
x_2 + x_6 \geq 1 & \textbf{District 9} \\
x_5 + x_6 \geq 1 & \textbf{District 10} \\
x_i \in \{0,1\}, \quad i \in \{1,2,\ldots,6\}
\end{cases}$$

SOLUTION by LINGO: The minimum number of stations to open is 3, where each district is

covered by at least one of the stations 1, 4, or 6.

3.5.2 *Model 2: Production and Inventory*

HINO Co. (Hi) produces two engines, one for trucks and one for cars. They anticipate the following demands in next 3 months:

	Month 1	Month 2	Month 3
Truck engines	400	300	500
Car engines	800	500	600

How can we meet monthly demand at a minimum total cost?

Our total cost T in this problem are the sum of

(1) H : holding costs associated with inventory levels and

(2) S : costs for supplies.

REQUEST: We want to identify, for each month,

* how many of each engine is produced and

* how many engines are placed in inventory, to get min cost T.

Specific conditions

x) Each month, max 1000 engines (combined) can be produced.

y) Each truck engine requires 10 hours of labor to produce and costs $\$2000$ in supplies, while each car engine requires 8 hours of labor and costs $\$1500$ in supplies.

z) At most 9000 hours are available each month.

u) At the end of the third month, management wants to have at least 100 of each engine in inventory.

(a) At the beginning of month 1, 100 truck engines and 200 car engines are in inventory.

(b) No backlogging is allowed, which means that each month's demand must be fully satisfied.

and (c) At the end of each month, a holding cost of $\$150$ per engine is assigned to any engine in inventory.

We define the following **decision variables** for each month $i \in \{1, 2, 3\}$:

- **the number of truck engines** T_i produced during month i,
- **the number of car engines** C_i produced during month i,
- the number of truck engines IT_i **in inventory** at the end of month i,
- number of car engines IC_i **in inventory** at the end of month i.

Define by Condition (a), the values $IT_0 = 100$ and $IC_0 = 200$, that is the starting inventory levels for each engine type.

Conditions (x), (y) and (z)

The constraints on the number of engines produced and the labor used in month by Condition (x)

$$T_i + C_i \leq 1000 \qquad \text{(the max manufacturing capacity);}$$

plus Conditions (y) and (z):

$$10\, T_i + 8\, C_i \leq 9000 \qquad \text{(the max labor capacity).}$$

- Condition (u) means $IT_3 \geq 100,\ IC_3 \geq 100$.
- Condition (b) means inventory at end of month i must non-negative, i.e. $IT_i,\ IC_i \geq 0$ for each month $i \in \{1, 2, 3\}$.

Holding Costs and Costs for supplies

- Condition (c) on holding cost

  $H = 150 * [$ no. of truck and car engine engines in 3 months$]$; so

$$H = \mathbf{150} * [\sum_{j=1}^{3} IT_j + \sum_{j=1}^{3} IC_j]$$

- Condition (y) on supply cost

  $S =$ **2000** $*$ (no. of truck engines produced in 3 months)

  $+$**1500** $*$ (no. of car engines produced in 3 months) gives us

  $S = 2000 * (T_1 + T_2 + T_3) + 1500 * (C_1 + C_2 + C_3)$

  [costs $2000 for each truck, $1500 for each car when HINO buys from suppliers,]

Multi-months Holding and Supplies costs

- Holding cost

$H = \mathbf{150} * (\sum_{j=1}^{3} IT_j + \sum_{j=1}^{3} IC_j)$ and on supply cost

$S = \mathbf{2000} * (T_1 + T_2 + T_3) + \mathbf{1500} * (C_1 + C_2 + C_3)$

- How about the total cost? $T = H + S$, but in how many decision variables?

Inventory- type constraints for multi-period manufacturing

Most importantly and new in this production type, we pay attention to the **inventory constraints** that have the form

$$I_t = I_{t-1} + C_t - D_t$$

for each time period t, where

- I_t is the inventory level for period t,
- D_t is the demand for period t,
- C_t is the amount of product generated during period t.

Result of Inventory- type constraints

Now due to Condition (a): $IT_0 = 100$, $IC_0 = 200$, and moreover

$$IT_1 = 100 + T_1 - 400,\ IT_2 = IT_1 + T_2 - 300,$$

$$IT_3 = IT_2 + T_3 - 500,\ IC_1 = 200 + C_1 - 800,$$

$$IC_2 = IC_1 + C_2 - 500,\ IC_3 = IC_2 + C_3 - 600.$$

FINAL RESULT: DIY on LINGO

to get a LP in 12 decision variables, and an optimal solution has total cost of $5,190,000.

PART II: SPECIFIC MODELS

Part II: Learning outcomes

After studying this lecture, you should be capable of:

- Express key concepts of Integer Optimization
- Understand and use the practical (Computational) Simplex Method for LP
- Understand and use the Branch-and-Bound Method for ILP
- Discuss why discrete optimization problems are difficult to solve in general

Brief contents of Part II

1. F. Models for Integer Optimization

   LP (Linear Optimization)

   IP, IO (Integer Optimization)

and Mixed ILP (MILP)

2. G. Capital Budgeting using the Knapsack Problem Formulation

   Optimal binary decision vector: when does it exist?

   ♠ Capital Budgeting via the knapsack problem

3. H.Transportation model with integer variables

3.6 F. Models for Integer Optimization- IO

QUESTION. *What is Integer Optimization or Integer Programming (IP)?*

- Integer programming (IP) is a mathematical programming problem in which **some or all of the variables** are restricted to assume only integer or discrete values.
- Hence the **optimal values of decision variables** must be integer.

For example, when a decision problem requires finding

the **optimal number of employees** to be assigned to jobs or

the **number of aircraft** to purchase at Thai Airways, ...

the decision variables must be obviously limited to integer values.

$$LO: \quad \max_{\boldsymbol{y}} \quad h^T \cdot \boldsymbol{y}$$

$$\text{s. t.} \quad B\boldsymbol{y} \leq \boldsymbol{b}$$

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n_+$$

- $\mathbb{R}_+ = \{y_i \in \mathbb{R} : \ y_i \geq 0\}$- the set of non-negative real numbers.
- $\mathbb{R}^n_+$ the product of n times $\mathbb{R}_+$. Moreover, $\mathbb{Z}^n_+ = \mathbb{N}^n = \mathbb{N} \times \mathbb{N} \times \cdots \times \mathbb{N}$ is the Cartesian product n times of the natural set $\mathbb{N} = \{0, 1, 2, \ldots, k, k+1, \ldots\}$

3.6.1 Pure IO and Mixed IO

Pure IO (PILP- Pure Integer Linear Programming)

$$IO: \quad \max_{\boldsymbol{x}} \quad c^T \cdot \boldsymbol{x}$$

$$\text{s. t.} \quad A\boldsymbol{x} \leq \boldsymbol{b}$$

$$\boldsymbol{x} \in \mathbb{Z}_+^n$$

Unlike LP, where problems with millions of variables and thousands of constraints can be solved in a reasonable time, **computational experience** with

- Integer LP (ILP= Pure IO- Pure Integer Optimization = PILP);
- Mixed IO or MILP (Mixed Integer LP, some decision variables are real, some are integer)

  remains elusive (i.e. very difficult).

Binary Optimization- BIP An important special case is Binary IO or BIP.

$$Binary\ IO: \quad \max_{\boldsymbol{x}} \quad c^T \cdot \boldsymbol{x}$$

$$\text{s. t.} \quad A\boldsymbol{x} \leq \boldsymbol{b} \ \text{ or } A\boldsymbol{x} \geq \boldsymbol{b}$$

$$\boldsymbol{x} = (x_1, x_2, x_3, \cdots, x_{n-1}, x_n) \ \in \{0,1\}^n$$

here all decision variables x_j have two choices only, $\{0,1\}^n = \{0,1\} \times \{0,1\} \times \cdots \times \{0,1\}$ (in n times). E.g, we recall the **Set Covering Problem** (see Model 1, in Lecture 3. Part I).

Mixed IO - Mixed ILP

$$MIO: \quad \max_{x,y} \quad c^T \cdot \boldsymbol{x} + h^T \cdot \boldsymbol{y}$$

$$\text{s. t.} \quad A\boldsymbol{x} + B\boldsymbol{y} \leq \boldsymbol{b}$$

$$\boldsymbol{x} \in \mathbb{Z}_+^n \quad (\boldsymbol{x} \geq 0, \text{ integer})$$

$$\boldsymbol{y} \in \mathbb{R}_+^n \quad (\boldsymbol{y} \geq 0, \text{ real})$$

Key bond between LP and Pure ILP

How to deal with integrality of solutions in an ILP using the well-known basic LP? Many ways, but a simple idea is called **LP relaxation**.

If all the integer restrictions on all the variables are omitted (that means we allow an integer variable to assume **continuous values**), an ILP becomes an LP, called its **LP relaxation**.

FACT: The *feasible region* of an ILP is obviously a subset of the feasible region of its LP relaxation, and therefore, the optimal objective value of an ILP is always **no better** than that of its LP relaxation.

LP relaxation of an Binary ILP

mathematically just means we impose the lower and upper bounds $0 \leq x \leq 1$ (**viewed as real numbers**) onto the binary variable x [originally with only two values 0 or 1].

Or to the general ILP, we impose the lower and upper bounds

$$a \leq x \leq b$$

(**viewed as real numbers**) onto the integer variable x viewed receiving only integer values $a, a+1, a+2, \cdots, b$.

How to make IP model with binary variables?

3.6.2 IP modeling with binary variables- BIP

Generic steps for modeling relations with Binary Choice

1. Define variables: For an interest event in practical usage, let

$$x = \begin{cases} 1, & \text{if event occurs} \\ 0, & \text{otherwise} \end{cases}$$

2. Modeling relations with binary choices:

   - Multiple-choice constraint (packaging constraint): at most one event occurs

$$\sum_i x_i \leq 1$$

   - Neither or both events occur means $x_2 - x_1 = 0$.
   - Contingency constraint (cause-effect constraint): If one event occurs then another occurs

$$0 \leq x_i \leq x_j.$$

   [In Project Management it states that ...

Modeling relations/constraints with binary choices

In Project Management it states that $x_i = 1 \Rightarrow x_j = 1$:

if $x_i = 1$ and project i is accepted, **then** necessarily $x_j = 1$ and project j

must be accepted (j like construction of a new plant for project i).

- Possibility constraint

If $x = 0$ then $y = 0$;

if $x = 1$ then y is unconstrained as $0 \leq y \leq Ux, \quad x \in \{0, 1\}$ for some $U > 0$.

Optimal binary decision vector: when does it exist

Let's look back at the standard Binary IP.

$$\textbf{Model BIP} \; : \quad \max_{\boldsymbol{x}} \quad Z = \boldsymbol{c}^T \cdot \boldsymbol{x}$$

$$\text{s. t.} \quad A\boldsymbol{x} \leq \boldsymbol{b}$$

$$\boldsymbol{x} = (x_1, x_2, x_3, \cdots, x_{n-1}, x_n) \; \in \{0, 1\}^n$$

where the constraint set $A\boldsymbol{x} \leq \boldsymbol{b}$ has m constraints, the constraint matrix $A = [a_{ij}]$ has size

$m \times n$. Obviously the feasible solution set F is a subset of the vertex $V = \{0,1\}^n$ of the hypercube

$$H_n = (V, E), \quad \text{in which the edge set} \quad E = \{(\boldsymbol{u}, \boldsymbol{v}) : d(\boldsymbol{u}, \boldsymbol{v}) = 1, \boldsymbol{u}, \boldsymbol{v} \in V\}.$$

Here, $d(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^{n} |u_i - v_i|$ is the *Hamming distance* between nodes $\boldsymbol{u} = u_1, u_2, \cdots, u_n$, $\boldsymbol{v} = v_1, v_2, \cdots, v_n \in V$.

■ CONCEPT 1.

Assume that matrix A is positive, that is every $a_{ij} \geq 0$.

Consider any $i = 1, 2, \ldots, m$.

- The i-th constraint of $A\boldsymbol{x} \leq \boldsymbol{b}$ is said to be 'perfect' iff $b_i \geq \sum_{j=1}^{n} a_{ij}$.
- The i-th constraint of $A\boldsymbol{x} \leq \boldsymbol{b}$ is said to be 'practical' (not perfect) if and only if $b_i < \sum_{j=1}^{n} a_{ij}$.
- We say that the constraint set $A\boldsymbol{x} \leq \boldsymbol{b}$ is **perfect** if all of its constraints are perfect.

Otherwise, if there exists a practical constraint, $A\boldsymbol{x} \leq \boldsymbol{b}$ is said to be **practical**.

Theorem 3.4 (General Model- no source-sink balance) ♡

Assume that constraint matrix A of **Model BIP** is positive .

1. If $A\boldsymbol{x} \leq \boldsymbol{b}$ is perfect and the cost vector $\boldsymbol{c} = (c_1, c_2, \cdots, c_n)$ is positive then the unique optimal vector of **Model BIP** is the vector $\mathbf{1}$ in H_n.

   Otherwise, if there exists a practical constraint in $A\boldsymbol{x} \leq \boldsymbol{b}$, the optimal solution can not be the vector $\mathbf{1}$.

2. Moreover, let $P = \{j_0 \in \{1, 2, \ldots, n\} : \exists i : a_{ij_0} > b_i\}$

   (set P consists of variables inducing practical constraints of $A\boldsymbol{x} \leq \boldsymbol{b}$).

   Clearly, $0 \leq |P| \leq n$, and the dimension of the search space is reduced from n to $n - |P|$.

   The size of the feasible set F now is $2^{n-|P|}$.

PROOF: We show only brief ideas for the 1st claims. DIY for the others.

(1) HINT: For any row $i = 1, 2, \ldots, m$ of $A\boldsymbol{x} \leq \boldsymbol{b}$, if there exists a column j_0 such that $a_{ij_0} > b_i$ then that i-th constraint is practical. Then any vector

$$\boldsymbol{u} = u_1, u_2, \cdots, u_n \in H_n$$

with coordinate $u_{j_0} = 1$ will be rejected from the feasible set F.

As a result, any feasible solution $\boldsymbol{v} = v_1, v_2, \cdots, v_n$ now must have coordinate $v_{j_0} = 0$; and the dimension of the search space is reduced from n to $n - 1$.

Now denote the variable indexes by $I_n = \{1, 2, \ldots, n\}$, and let any $K \subset I_n$.

Write $\boldsymbol{x}_K$ be the subset of decision variables x_j with index $j \in K$, and $\boldsymbol{c}_K$ be associated cost coefficients in the objective function Z. Then clearly

$$Z_K = \max_{\boldsymbol{x}_K} \boldsymbol{c}_K^T \cdot \boldsymbol{x}_K = \sum_{j \in K} c_j x_j \leq \sum_{j \in I_n} c_j x_j \leq \sum_{j \in I_n} c_j.$$

The last equality occurs only at the vector $\mathbf{1}$. By contradiction, the vector $\mathbf{1}$ is not optimal solution when we find a practical constraint.

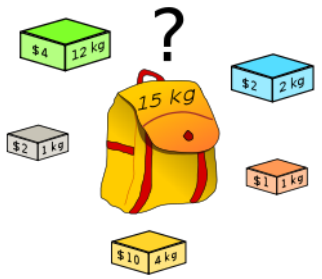
(2) HINT: Put the problem in terms of hypercube structure, as a graph of 2^n vertices if you model n variables.

3.7 G. Capital Budgeting with the Knapsack Formulation

3.7.1 Concepts and Setting

What is a knapsack?

*A soldier's or hiker's **bag** with shoulder straps, carried on the back, and typically made of canvas or other weatherproof material.*



The 0-1 knapsack problem is a *decision making* problem when you prepare to travel and bring n items with you. Since your bag has a fix volume (or weight) b, you have to selected items with

volumes a_j so that:

$$\sum_{j=1}^{n} a_j x_j \leq b$$

(the total volume is less than or equal to a given limit b), where

$$x_j = \begin{cases} 1, & \text{if item } j \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases}$$

is a knapsack problem in which we have

- $m > 1$ bags, each has fix volume (or weight) b_i, for $i = 1, \cdots, m$; and
- each item j now has **large volume** and can **be cut into smaller pieces** and distribute into different bags i with volume a_{ij}.

Formally, for each ith knapsack or bag, you have to selected those items j so that the total volume of that bag is less than or equal to a given limit b_i:

$$\sum_{j=1}^{n} a_{ij}\, x_j \leq b_i, \ \forall i = 1, \cdots, m$$

In a typical capital-budgeting problem,

decisions involve the selection of a number of *potential investments*.

The investment decisions might be

- to choose among possible plant locations, to select a configuration of capital components, or
- to settle upon a set of research-and-development projects.

3.7.2 Capital Budgeting- Modeling Steps

Let define binary variables

$$x_j = \begin{cases} 1, \text{ if project } j \text{ is selected.} \\ 0, \text{ otherwise.} \end{cases}$$

The variables are binary, indicating that the j-th investment is rejected or accepted.

A) The general knapsack problem with BIP

The objective is to maximize total contribution from all investments without exceeding the limited availability b_i of any resource. With assumptions that

- c_j is the contribution (value) of investment, resulting from the jth investment and
- a_{ij} is the cost or amount of resource i (as manpower or money) used on or spent for investment j,

we can state the capital-budgeting formally as **the general knapsack problem with BIP**.

Denote

- $m =$ # the number of resources, like time or financial budget with total budget

$$b = \sum_{i=1}^{m} b_i$$

- $n =$ # the number of projects/ investments,

then the general knapsack is

$$\max_{x} \quad c^T \cdot \boldsymbol{x} = \sum_{j=1}^{n} c_j \, x_j$$

$$\text{s. t.} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad \forall \, i = 1, \cdots, m$$

$$\boldsymbol{x} \in \{0, 1\}^n$$

meaning $x_j = 0$ or 1, $\quad \forall \, j = 1, \cdots, n.$

B/ The simplest capital-budgeting model

The 0-1 knapsack problem with BIP with notation

$m = 1$: only one resource

n: # projects, with total budget b

a_j: the portion or cost of the resource spent for project j

c_j: value of project j

is described as

$$\max_x \quad c^T \cdot \boldsymbol{x} = \sum_{j=1}^{n} c_j \, x_j$$

$$\text{s. t.} \quad \sum_{j=1}^{n} a_j x_j \leq b,$$

$$x_j = 0 \text{ or } 1, \quad (j = 1, \cdots, n)$$

Theorem 3.5 (2. Roots of BILP)

Suppose: $c_j, a_j\ (j = 1, \ldots, n)$*-all positive, and satisfies*

$$\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \cdots \geq \frac{c_n}{a_n}. \tag{3.1}$$

Then there is an index $p\ (1 \leq p \leq n)$ *and an optimal sol.* $\mathbf{x}^*$ *s.t.*

$$x_1^* = x_2^* = \cdots = x_{p-1}^* = 1,\ x_{p+1}^* = x_{p+2}^* = \cdots = x_{p+1}^* = 0.$$

♡

Proof HW!

Hint: set n small like $n = 3$, set c_j to be specific numbers 3, 4, 3 and a_j to be specific numbers 1, 2, 3 to get 'utility' ratio $u_j = \frac{c_j}{a_j}$ and think from there.

C/ The m-resources capital-budgeting model

$m > 1$: # the number of resources, (time or budget) with total budget

$$b = \sum_{i=1}^{m} b_i$$

n: # the number of projects/ investments,

$$\max_{x} \quad c^T \cdot \boldsymbol{x} = \sum_{j=1}^{n} c_j \, x_j$$

$$\text{s. t.} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad \forall \, i = 1, \cdots, m$$

$$\boldsymbol{x} \in \{0, 1\}^n$$

meaning $x_j = 0$ or 1, $\quad \forall \, j = 1, \cdots, n.$

Now consider a specific case with $m = 3$ in the Capital Budgeting.

D/ Capital Budgeting problem with data

Four projects are being considered for execution over the next $m = 3$ years.

The expected returns in *net present value* c_j and yearly funds (resources) a_{ij} of each project j

available per year i are tabulated below (unit is 1 *million dollar*):

Year	**Project**				**Available Funds**
$\downarrow$	1	2	3	4	
Year 1	$a_{11} = 5$	9	3	4	$b_1 = 21$
Year 2	2	6	5	3	$b_2 = 16$
Year 3	4	5	4	$a_{34} = 4$	$b_3 = 17$
Returns	$c_1 = 20$	$c_2 = 45$	$c_3 = 24$	$c_4 = 30$	**Max return = ?**

Hence $n = 4$ projects,

being performed in $m = 3$ years;

with total resource fund b_1 in year 1 is 21 million dollars,

resource b_2 in year 2 is \$16 million,

b_3 in year 3 is $ 17 million, and

c_i is the return of project i given in the last row.

Modeling phase

GOAL: to determine **which projects should be executed** over the next 3 years

such that at the end of the 3-year period, the total returns of the executed projects are at the maximum, subject to the availability of funds each year.

3-STEP Modeling.

1. Identify the decision variables
2. Formulate the constraints
3. Formulate the objective function

FIGURE OUT in CLASS these steps?

1. Identify the decision variables

* Brainstorming first:

Each project is either executed or rejected. Therefore, the problem reduces to a 'yes-no' decision for each project. Such a decision can be represented as a binary variable, where the value **1 means 'yes' and 0 means 'no.'**

* Delivering your choice: Should we define decision variables by

Option 1: For $j = 1, 2, 3, 4$ let

$$y_j = \begin{cases} 1, & \text{if project } j \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

or use **Option 2**: For $i = 1, 2, 3$ and $j = 1, 2, 3, 4$ let

$$x_{ij} = \begin{cases} 1, & \text{if project } j \text{ is selected to implement in year } i \\ 0, & \text{otherwise?} \end{cases}$$

2. Formulate the constraints Now we follow **Option 1** [why not Option 2?]

The constraints say that total annual expenditures of the selected projects **do not exceed** funds available for each year. Mathematically we express them as in matrix form $A\,\boldsymbol{y} \leq \boldsymbol{b}$, or explicitly

$$\sum_{j=1}^{n} a_{ij} y_j \leq b_i, \ i = 1, 2, 3.$$

For year 1, 2, 3 the constraints respectively are :

$$5y_1 + 9y_2 + 3y_3 + 4y_4 \leq 21,$$

$$2y_1 + 6y_2 + 5y_3 + 3y_4 \leq 16,$$

$$4y_1 + 5y_2 + 4y_3 + 4y_4 \leq 17.$$

Remind the data

c_j = expected returns of each project j, and a_{ij}= fund for project j in year i

Year	**Project**				**Available Funds**
$\downarrow$	1	2	3	4	
Year 1	$a_{11} = 5$	9	3	4	$b_1 = 21$
Year 2	2	6	5	3	$b_2 = 16$
Year 3	4	5	4	$a_{34} = 4$	$b_3 = 17$
Returns	$c_1 = 20$	$c_2 = 45$	$c_3 = 24$	$c_4 = 30$	**Max return = ?**

3. Formulate the objective function

AIM

To maximize the *total returns* of the selected projects at the end of the 3-year planning period.

The (optimal) total of returns is

$$Z = \max_{\boldsymbol{y}} \quad c^T \cdot \boldsymbol{y} = \sum_{j=1}^{n=4} c_j \, y_j = 20y_1 + 45y_2 + 24y_3 + 30y_4$$

Remark:

If we employ Option 2, hope to get better understanding with 12 variables

$\boldsymbol{x} = (x_{11}, x_{12}, x_{13}, x_{14}, \cdots, x_{33}, x_{34})$ and then get the objective

$$T = \max_{\boldsymbol{x}} \quad c^T \cdot \boldsymbol{x} = \sum_{i=1}^{m=3} \sum_{j=1}^{n=4} c_{ij} \, x_{ij}?$$

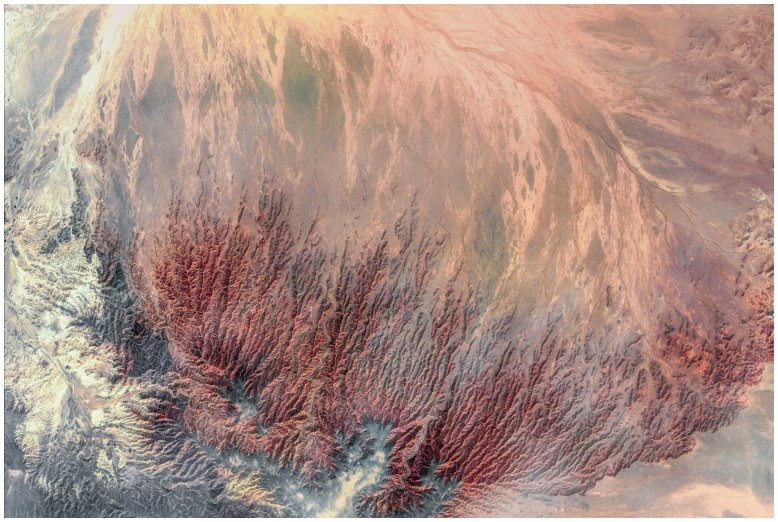
But what are the coefficients c_{ij} then?

The Pure Binary LP model for this capital budgeting problem is

$$\text{maximize } Z = \sum_{j=1}^{n=4} c_j \, y_j$$
$$= 20y_1 + 45y_2 + 24y_3 + 30y_4$$

s.t.

$$\begin{cases} 5y_1 + 9y_2 + 3y_3 + 4y_4 \leq 21 & \text{for year 1} \\ ? & \text{for year 2} \\ ? & \text{for year 3} \\ y_j \in \{0,1\}, \quad j = 1,2,3,4. \end{cases}$$



What next?

A student team could work carry out a case study using few transportation models:

1. Problems without balancing of source and sink
2. Problems with a balance of source and sink.

3.8 H. Transportation models with integer variables

Overview

We give brief introduction to Transportation Models to which team can use for assignment [due in Week 15 or Week 16]. Teams can also use models suggested in Section L 3.11 of this text to prepare your team work. This year 2023 we focus on using INTEGER OPTIMIZATION (IP), STOCHASTIC OPTIMIZATION or Programming- SP and Applications in Smart Modern Urban Management, with few problems in Evacuation planning in Disaster response, Logistics, Facility location, Industrial manufacturing ... being discussed in Chapter **??**.

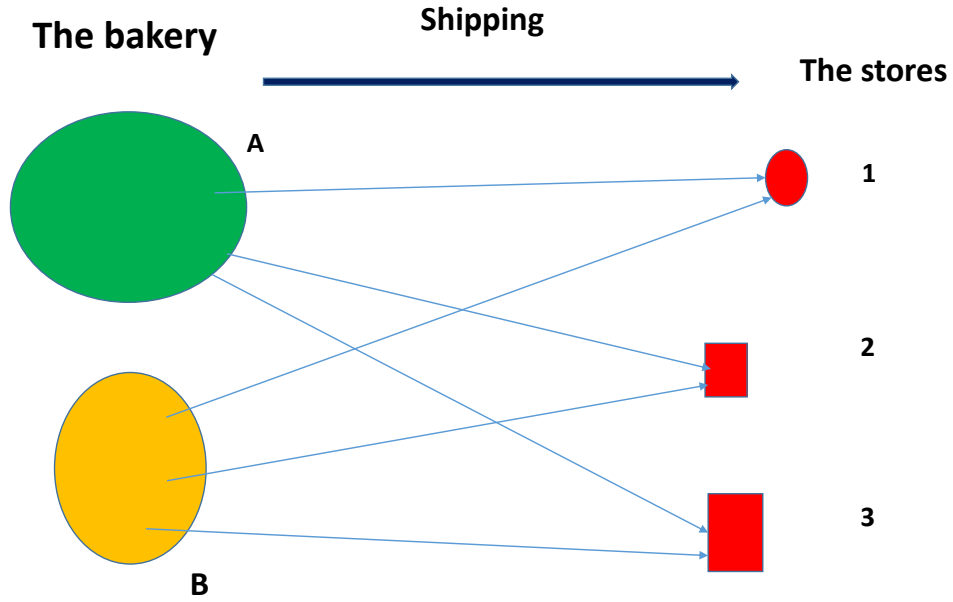
The reasons come from key facts below:

- (0) Having right 'good models' summarzing all key factors that impact on urban life before rare and bad event's occurences is an urgent need.

- (1) Modern Urbans are lively places only with efficient and safe transportation systems, and at least without traffic jam,
- (2) Successful Logistics is an essential factor for daily live of dwellers and sustainable development,
- and (3) Smooth Logistics (accepting a lot of uncertainty) is understood in broad sense, so be helpful in normal times, extremely helpful in hard times and/or other relevant sectors like military science (war time) and medical science (epidemic, pandemic as Covid19)...

PROBLEM H1: Transportation without balancing of source and sink

Context: A local baked goods company has two bakeries where they bake their goods, which they then ship to three different stores to sell. Here the bakeries would be our supply nodes and our stores would be the demand nodes. We would have arcs (i, j) going from each bakery i to each store j, where $i = 1, 2$ and $j = 1, 2, 3$.



- **Each bakery can produce** up to 50 truckloads of baked goods per week, while the stores anticipate the following weekly demands, in number of truckloads.

Table 3.2: *Bakery transportation*

Store	Demand
1	30
2	25
3	40

- **Each bakery can supply** any of the stores, and the unit **cost per truckload** c_{ij} of shipping from a bakery to a given store is given in the table below.

Table 3.3: *Cost per truckload of shipping from bakeries i to stores j*

	Store 1	**Store 2**	**Store 3**
Bakery A	$c_{11} = \$20$	\$45	\$35
Bakery B	\$35	\$35	$c_{23} = \$50$

QUESTION. *How much truckloads should be sent from each bakery to each store so as to minimize the* ***total shipping cost****?*

GOAL: to determine **truckloads should be made** from each bakery to each store so that the total cost of transportation is minimized subject to the above conditions.

3-STEP Modeling.

1. Identify the decision variables
2. Formulate the constraints
3. Formulate the objective function

FIGURE OUT in CLASS these steps?

Our decision variables [for $i = 1, 2;\ j = 1, 2, 3$] are

$x_{ij} =$ the amount of truckloads of baked goods sent from bakery i to store j.

The constraints

a/ Constraints at supply nodes: (bakery factories)

Using these variables, the number of truckloads sent from *bakery* i *(supply node)* to all stores (demand nodes) is

$$\sum_{j=1}^{3} x_{ij} = x_{i1} + x_{i2} + x_{i3}. \tag{3.2}$$

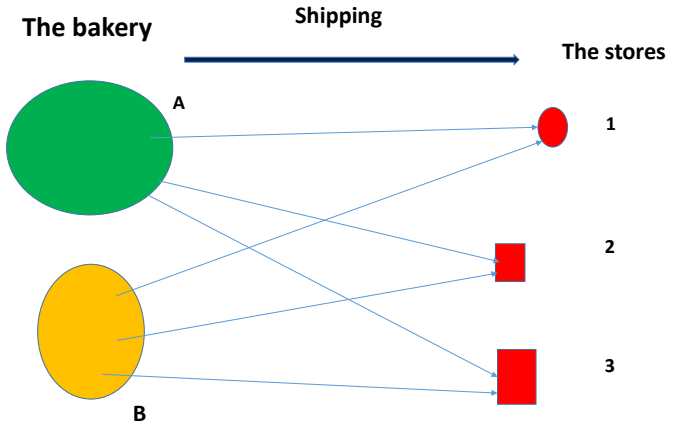
Since each bakery can send at most 50 truckloads out, we have

$$\sum_{j=1}^{3} x_{ij} \leq 50, \qquad \forall\, i = 1, 2. \tag{3.3}$$

b/ Constraints at demand nodes (stores):

Now consider the constraints being associated with the demand nodes. For each store j *(demand node)*, the number of truckloads sent to store j is

$$\sum_{i=1}^{2} x_{ij} = x_{1j} + x_{2j},\ \forall j \in \{1, 2, \ldots, n\}. \tag{3.4}$$



Given the demand that each store has, can you write all the constraints?

Finally, the objective is to choose the values of these 6 decision variables x_{ij} so as to minimize

$$Z = \sum_{i}^{2} \sum_{j}^{3} c_{ij}\ x_{ij} = 20x_{11} + 45x_{12} + \cdots + 35x_{22} + 50\ x_{23}.$$

Table 3.4: *Cost per truckload of shipping from bakeries i to stores j*

	Store 1	Store 2	Store 3
Bakery A	$c_{11} = \$20$	\$45	\$35
Bakery B	\$35	\$35	$c_{23} = \$50$

In general, without balancing of source and sink, what is the transportation model?

If there are

m supply nodes, each with supplies r_i , and

n demand nodes, each with demand d_j, and

the **cost associated with shipping items** from supply node i to demand node j is c_{ij} then we have the following model for optimizing total cost Z.

ILP :
$$\min_{\boldsymbol{x}} Z = \boldsymbol{c}^T \cdot \boldsymbol{x} = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} \le r_i, \quad i \in \{1, 2, \ldots, m\}$$

$$\sum_{i=1}^{m} x_{ij} \le d_j, \quad j \in \{1, 2, \ldots, n\}$$

and $x_{ij} \in \mathbb{N}, \quad \forall i, j.$

$$\textbf{ILP}: \quad \min_{\boldsymbol{x}} Z = \sum_{i}^{2}\sum_{j}^{3} c_{ij}\, x_{ij} = 20x_{11} + 45x_{12} + \cdots + 35x_{22} + 50\, x_{23}$$

$$\sum_{j=1}^{3} x_{1j} = \quad \ldots \quad \leq r_1 = 50,$$

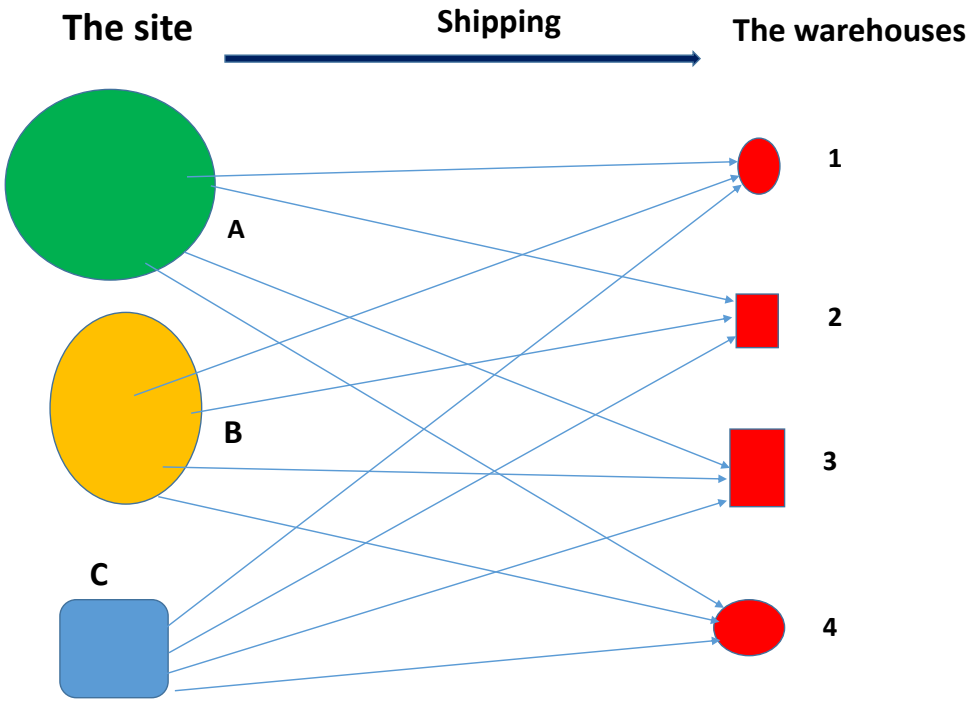
$$\sum_{j=1}^{3} x_{2j} = \quad \ldots \quad \leq r_2 = 50,$$

$$\sum_{i=1}^{2} x_{i1} = \quad \ldots \quad \leq d_1 = 30,$$

$$\sum_{i=1}^{2} x_{i2} = \quad \ldots \quad \leq d_2 = 25$$

$$\sum_{i=1}^{2} x_{i3} = \quad \ldots \quad \leq d_3 = 40$$

PROBLEM H2: Transportation with a balance of source and sink



Denote by x_{ij} the number of items (cars, PC, tons of steel...)

to be brought/ shipped from producer (site, farm ...) P_i to warehouse F_j.

A balance of commodities *delivering from sources* and *receiving at sinks* [source - sink balance]

assumes that:

- the i-th producer (production site) supplies daily r_i quantities (goods), for $i = 1, 2, ..., m$, and
- the jth warehouse (retail site, consumer) need d_j quantities, for $j = 1, 2, ..., n$,
- and **the total supply agrees with (equals) the total demand** assumption must be satisfied:

$$r_1 + r_2 + \ldots + r_m = d_1 + d_2 + \ldots + d_n \tag{3.5}$$

for any given vector $\mathbf{r} \in \mathbb{N}^m$ and $\mathbf{d} \in \mathbb{N}^n$.

Transportation problems where $\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} d_j$ [the total supply equals the total demand] are said to be *balanced*.

A transportation plan is a matrix of size $m \times n$

$$X = (x_{ij}) \in \mathbb{N}_*^{m \times n},$$

here x_{ij} is the number of items to be brought from producer P_i to warehouse F_j.

Modeling: mathematically, let $C = (c_{ij}) \in \mathbb{R}_*^{m \times n}$ be an $m \times n$ matrix of non-negative real

numbers, representing the transportation costs.

The aim: find an optimal plan to transport goods from producers to warehouses in the sense of minimizing the cost of transporting.

To be precise, we determine which plan for assigning these shipments to the various *producer-factory* combinations that would minimize the total shipping cost $Z = \sum_i^m \sum_j^n c_{ij}\, x_{ij}$.

Minimize $Z = \sum_i^m \sum_j^n c_{ij}\, x_{ij}$ subject to 3 types of constraints:

- **The Source (Sites) constraints**: For each site i (source node), the number of truckloads sent

$$\sum_{j=1}^{n} x_{ij} = r_i, \qquad \forall\, i = 1, 2, \ldots, m; \tag{3.6}$$

- **The Sink (Warehouse) constraints**: For each warehouse j (demand node), the number of truckloads received

$$\sum_{i=1}^{m} x_{ij} = x_{1j} + x_{2j} + \cdots + x_{mj} = d_j, \ \forall\, j = 1, 2, \ldots, n; \tag{3.7}$$

- **The Source-sink Balance constraint** $r_1 + r_2 + \ldots + r_m = d_1 + d_2 + \ldots + d_n$

  says that the total supply agrees with the total demand.

  See MODEL **??** where we combine

- transportation - **shipping costs**
- and **costs for operating** distribution centers in one model.

PART II SUMMARY

I) Consider a linear program with two variables and three constraints: Maximize $z = 2x_1 + 4x_2$

subject to

$$\begin{cases} x_1 + 5x_2 & \leq 80 \\ 4x_1 + 2x_2 & \geq 20 \\ x_1 + x_2 & = 10 \\ x_1 \geq 0,\ x_2 \geq 0 \end{cases}$$

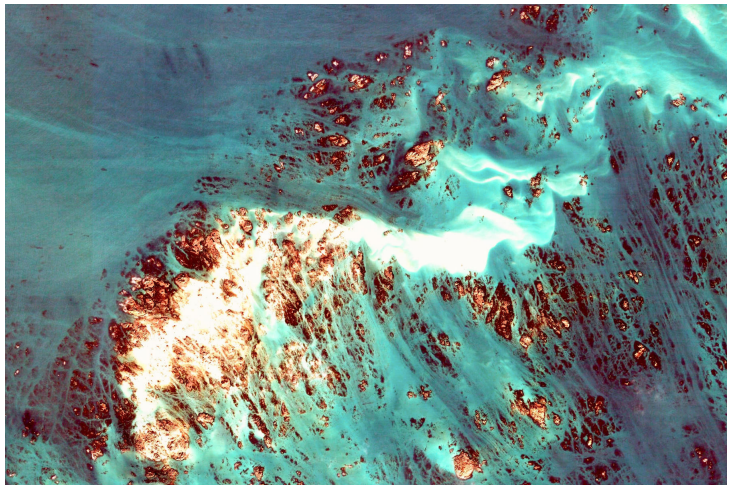
The number of slack and surplus variables needed to transform the constraint to the **Canonical form** of our LP model is 0, 1, 2 or 3?

II) Consider a program with two variables and constraints given as:

$$\begin{cases} x_1 + 5x_2 \leq 80 \\ x_1 \geq 0,\ x_2 \geq 0 \end{cases}$$

The feasible region which are associated with the constraints is

a line, a triangle, a pentagon, or a quadrilateral?



PART III- Branch and Bound Method -

Mathematical Polyhedron Theory for ILP

The tools include geometrical thinking, **algebraic formulation**, **algorithmic ideas**, linear algebra's background, **mathematical logic**, programming skill, and sometime **management viewpoints**.

The lecture promotes the T learning principle to students in which the lecturer's aims are both telling interesting and apparently complex problems (in R & D) and emphasizing the essential role of **mathematical modeling with various tools**.

I. Branch-and-Bound Method [4]

J. CHAPTER 3' REVIEW: Popular problems revisited

[4] **Courtesy** Dr. Trang H. Le and the staff of CSE faculty (HCMUT-VNUHCM) for material used to design **Section I**.

K. Modeling of real world problems with LP and ILP methodology for team works.

* **Mathematical Polyhedron Theory for ILP and Stochastic Programming** moved to next chapter.

KEY REFERENCES

I have used the following texts when preparing the note, among other resources.

[0] My lecture note on the class SCMA 621 - OPERATIONS RESEARCH in MUSC

[1] Canvas paintings by Australian artists of ethnic minorities, Australian National Museum

[2] Deterministic Operations Research, David J. Rader, 2010, John Wiley & Sons

[3] Operations research and management science handbook, A. R. Ravindran, 2008, CRC

[4] Practical Optimization: a Gentle Introduction, John W. Chinneck, 2000

[5] Introduction to Linear Optimization, Dimitris Bertsimas & John Tsitsiklis, Athena, 1997

3.9 I. Branch-and-Bound (BB) Method for ILP

3.9.1 WHAT, WHY & HOW Questions? Key Ideas

Branch-and-Bound Method generally attempts to solve discrete optimization problems by dividing the feasible region and examining each subregion for the best integer solution. It often uses a relaxed version of the problem to obtain bounds on the integer solution as a means of determining whether a subregion is worth investigating.

- Because any bounded pure integer programming (pure IP) problem has only a finite number of feasible solutions, it is natural to consider using some kind of enumeration procedure for finding an optimal solution.

  Unfortunately, as we discussed in the preceding section, this finite number can be, and usually is, **very large**.
- Since the original "large" problem is too difficult to be solved directly, it is divided into smaller and

smaller sub-problems until these sub-problems can be conquered. Therefore, any enumeration procedure should be cleverly structured so that only **a tiny fraction of the feasible solutions** actually need be examined.

This is the *divide and conquer* method. We divide a large problem into a few smaller ones.

1. The *dividing or branching* part is done by

   * partitioning the entire set of feasible solutions into smaller and smaller subsets,

   * using integer values of certain variable x_j.

2. The *conquering (fathoming)* part is done partially by bounding how good the best solution in the subsets (smaller problems) can be. To do so, we may have to divide the problem further, until we get a problem that we can handle, that is the *"bounding"* part.

   If further branching on a sub-problem will yield no useful information, then we can **fathom** (**dismiss**) the sub-problem.

3. The *relaxation* uses the so-called *linear programming relaxation* to estimate the optimal solution

of an IP.

Reminder: For an integer programming model $\mathcal{P}$, the linear programming model we get by dropping the requirement that all variables must be integers is called **the linear programming relaxation of $\mathcal{P}$.** [5]

- **Branching and Branching Variables**: When we divide the current subregion into smaller ones in a branch and bound algorithm, we are said to be branching, and the variable with which we are decomposing the region is known as the branching variable.
- **Branch-and-Bound Tree** is a binary tree describing this decomposition.

  Each node in this tree corresponds to a relaxed sub-problem obtained by the decomposition, while each edge corresponds to how this sub-problem was transformed into the new sub-problem. Often, this is due to the addition of a new constraint.
- **Fathom a Node**: To fathom a subproblem (node) or to prune the branch-and-bound tree means we **do not branch** from that subproblem due to either its *infeasibility*, or because a bound on

[5] Omitting all the integer restrictions on all variables of $\mathcal{P}$ means all **integer variables** will get *continuous values*, the initial ILP becomes an LP, called its **LP relaxation**.

its optimal integer value is not better than the value of the current best-known integer solution to the original problem.

Active Nodes: Those nodes that we have **not** fathomed yet [**not** conquered] or not branched upon.

Property 3.1.

Consider a maximization IP: $Z(\mathbf{x}) = \max\ \boldsymbol{c}^T\ \mathbf{x}$ with constraints $A\ \mathbf{x}\ \leq \mathbf{b}$.

1. If LP-relaxation has *integral* optimal solution $\mathbf{x}_{OPT}$, then $\mathbf{x}_{OPT}$ is optimal for the IP too.
2. Fix an *optimal* solution $\mathbf{x}_{OPT}$ of that max IP, and

   call $\mathbf{x}_{LP}$ the optimal solution of its LP-relaxation,

   then $Z(\mathbf{x}_{LP}) \geq Z(\mathbf{x}_{OPT})$. That is, the optimal value of the LP-relaxation is an upper bound for the optimal value of the integer program.
3. The best current integer solution found so far is stored as **incumbent** $\mathbf{x}^*$, [6]

[6] Incumbent solution refers to the best known feasible solution in the branching tree.

its value is denoted by $Z^* = Z(\mathbf{x}^*)$, and $Z^* = Z(\mathbf{x}^*) \leq Z(\mathbf{x}_{OPT})$ - it is a lower bound for the optimal.

3.9.2 Brief Branch-and-Bound Algorithm for a MAX IP

Consider a maximization IP: $Z_{IP} = \max\ \boldsymbol{c}^T\ \mathbf{x}$ with constraints $A\ \mathbf{x}\ \leq \mathbf{b}$.

0. Initialization: Set $Z^* = -\infty$.

If **not** conquered, we perform full steps below (`branching, bounding, conquering, and optimality test`) for each iteration.

1. Branching: Among the remaining un-fathomed sub-problems, select the one that was created most recently. Precisely, we choose from among those variables x_i, that **do not** have integral values at this node, one variable to be the branching variable x_k.

Formulation of New Nodes: Branch from the **active node** x_k with value v_k for this sub-problem to create two new sub-problems

* by fixing the next branching variable at either 0 or 1, [for the Binary IP, $v_k \in (0, 1)$],

** for the general IP, creating two new mixed-integer sub-problems represented by the node x_k.

The left and right branches adds respectively the constraints

$$x_k \leq \lfloor v_k \rfloor \qquad \text{and} \qquad x_k \geq \lfloor v_k \rfloor + 1. \tag{3.8}$$

NOTE: the LP-relaxation is **not** the only choice, but also is the one most often used, since we can form the linear programming relaxation very easily by removing the integrality constraints, and we can solve efficiently linear programs by the Simplex Method.

2. **Bounding:** For each new sub-problem, obtain its bound $Z(\mathbf{x}_{LP})$ by applying the **simplex method** to its LP relaxation and rounding down the integer optimal value $Z(\mathbf{x}_{OPT})$.

3. **Conquering (fathoming):** For each new sub-problem, apply the three fathoming tests summarized below, and discard those sub-problems that are fathomed by any of the tests.

   - Test 1: Its bound $Z(\mathbf{x}_{OPT}) \leq Z^*$, or
   - Test 2: Its LP relaxation has **no** feasible solutions, or
   - Test 3: The optimal solution for its LP relaxation is *integer*.

If this solution is better than the *incumbent*, it becomes the new incumbent,

and Test 1 is reapplied to all unfathomed sub-problems with the new larger Z^*.

Optimality test: Stop when there are no remaining sub-problems; the current incumbent is optimal. If there is **no** incumbent, conclude that the sub-problem has no feasible solutions.

Otherwise, return to Step 2. to perform another iteration. ■

Knowledge box 2. *We informally utilize the following facts in Branch-and-Bound Algorithm.*

- The key point in Step 1 - Branching is to choose **active node** and

  to branch on a branching **fractional variable**.

- Solve the linear relaxation of the problem. If the solution is integer, then we are done.

  Otherwise create **two new sub-problems** by branching on a *fractional* variable.

- A node (sub-problem) is **not active** when any of the following occurs:

  1. The node is being branched on (the current subproblem yields a relaxation bound that is not

better than the value of the incumbent solution);

2. The solution is *integral* (when the relaxed problem is solved);

3. The sub-problem is *infeasible*;

- **Choose an active node** and branch on a fractional variable.

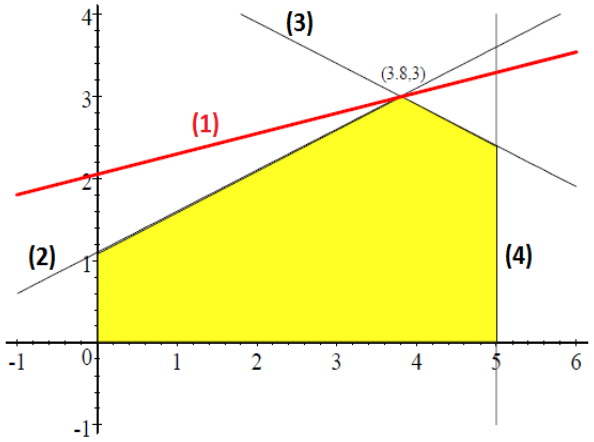
Repeat until there are no active sub-problems.

♦ **EXAMPLE 3.2** (BB for Integer Programming)**.**

$$\max(Z = -x_1 + 4x_2) \tag{3.9}$$

Subject to

$$\begin{cases} -10x_1 + 20x_2 \leq 22 \\ 5x_1 + 10x_2 \leq 49 \\ x_1 \leq 5,\ x_i \geq 0, x_i \in \mathbb{Z}\ \ \forall i \in \{1, 2\} \end{cases}$$

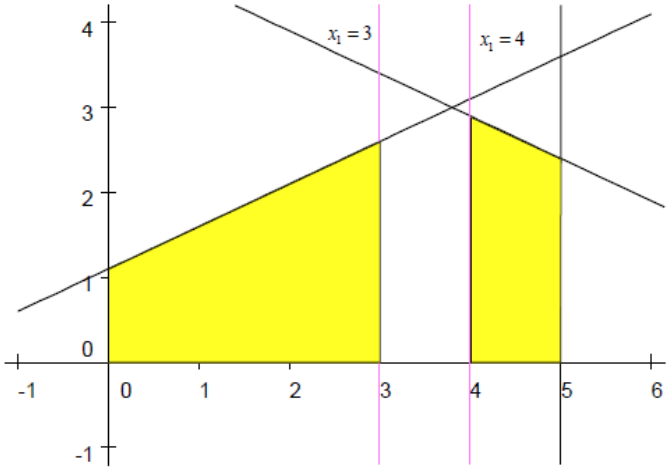


With linear programming relaxation, we drop $x_i \in \mathbb{Z}$ so $x_i \geq 0\ \ \forall i \in \{1, 2\}$.

Optimal solution of relaxation is (3.8, 3) with Z = 8.2.

The *initial relaxed problem* is often referred to root problem, corresponding to the **root node** x_1.

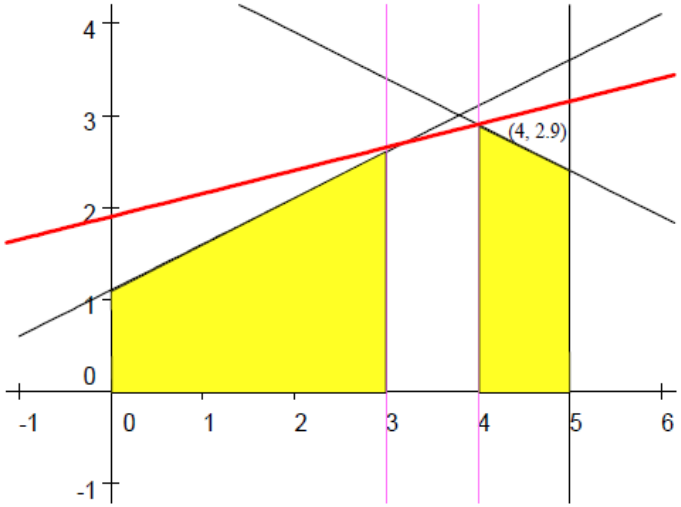
Since optimal solution of relaxation is (3.8, 3), we consider two cases: $x_1 \geq 4$ and $x_1 \leq 3$.



CASE 1: The right search tree branches at variable x_1, say $x_1 \geq 4$

$$\begin{cases} -10x_1 + 20x_2 \leq 22 \\ 5x_1 + 10x_2 \leq 49 \\ x_1 \leq 5 \\ x_1 \geq 4, x_2 \geq 0, x_i \in \mathbb{Z} \ \ \forall i \in \{1, 2\} \end{cases}$$

has optimal solution at (4, 2.9) with $Z = 7.6$. Then we consider two cases: $x_2 \geq 3$ and $x_2 \leq 2$.



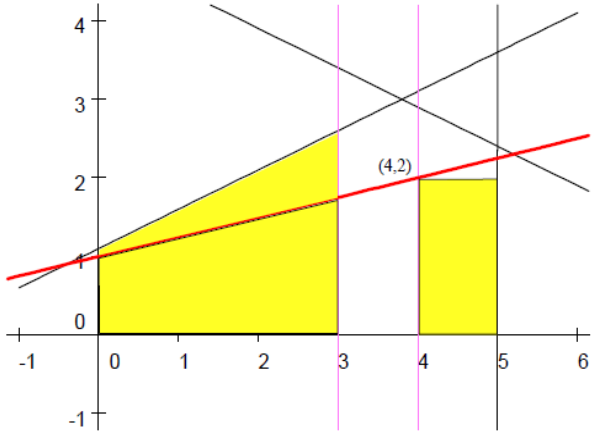
Case 1A: $x_1 \geq 4$, and $x_2 \geq 3$: Now $\max(Z = -x_1 + 4x_2)$ subject to

$$\begin{cases} -10x_1 + 20x_2 \leq 22;\ 5x_1 + 10x_2 \leq 49 \\ x_2 \geq 0, x_2 \in \mathbb{Z} \\ 4 \leq x_1 \leq 5\ AND\ x_2 \geq 3 \end{cases}$$

has **no feasible** solution (since $5x_1 + 10x_2 \geq 50$) so the IP has no feasible solution either.

Case 1B: $x_1 \geq 4$, $x_2 \leq 2$: now $4 \leq x_1 \leq 5\ AND\ 2 \geq x_2 \geq 0$

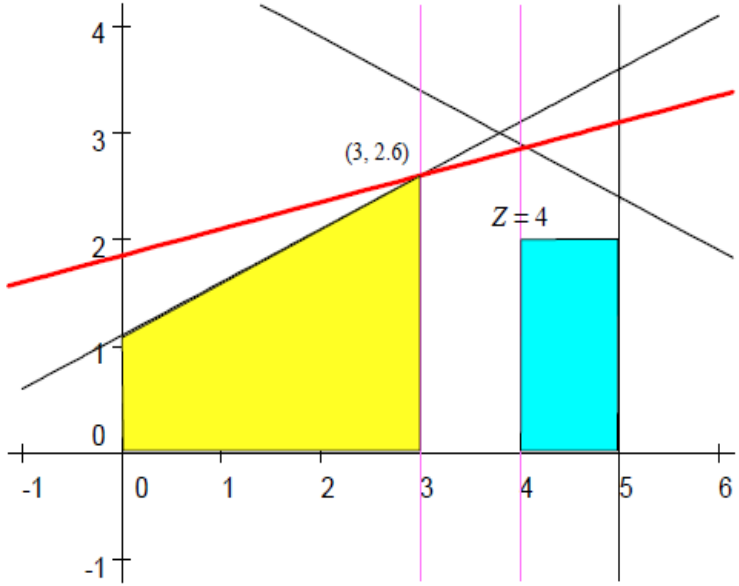
has an optimal solution at (4, 2) with Z = 4. This is the optimal solution of the IP as well. Currently, the best value of Z for the original IP is Z = 4.



CASE 2: Now we back to the left search tree branching at variable x_1:

$$\begin{cases} -10x_1 + 20x_2 \leq 22 \\ 5x_1 + 10x_2 \leq 49 \\ x_1 \leq 5 \; AND \; x_1 \leq 3 \rightarrow 0 \leq x_1 \leq 3 \\ x_2 \geq 0, x_i \in \mathbb{Z} \;\; \forall i \in \{1, 2\} \end{cases}$$

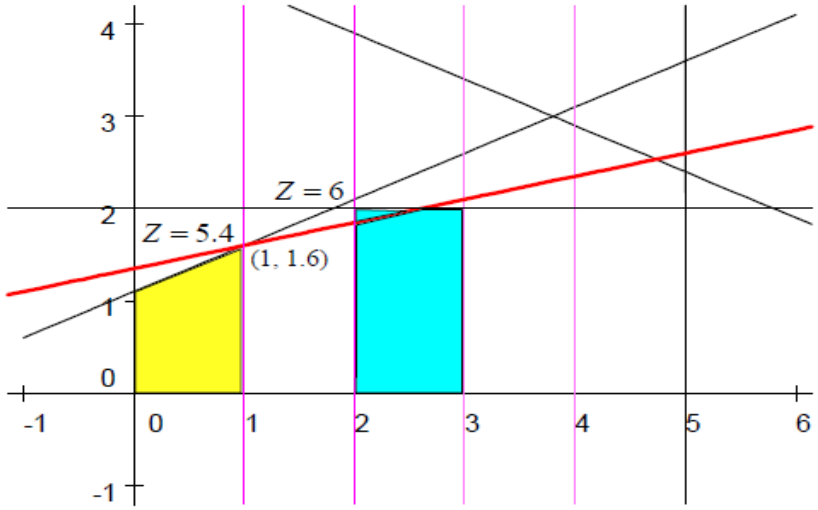
has an optimal solution at (3, **2.6**) with Z = 7.4. We branch out further to two cases: $x_2 \geq 3$ and $x_2 \leq 2$.

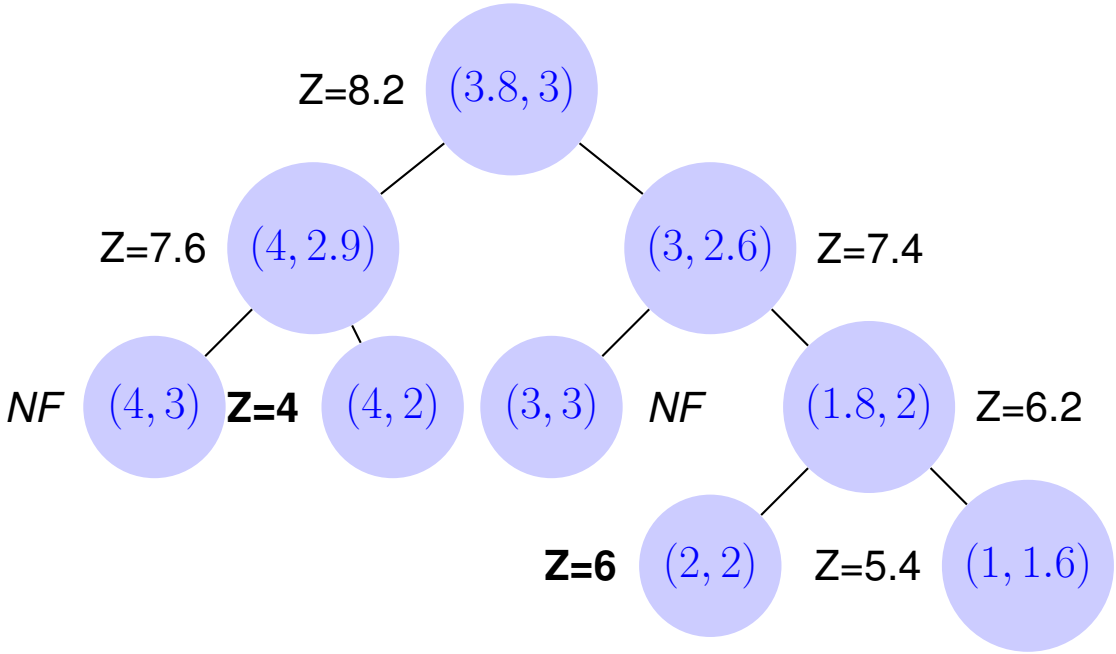


Continue the same way $\cdots$

LP relaxation: New bounds $x_1 \leq 3$, and $x_2 \leq 2$ return an optimal at (1, 1.6) with Z = 5.4.

Then any integer solution in this region can not give us a solution with the value of Z greater than 5.4. This branch is fathomed. We get the complete search tree.


4
3
2
1
0
-1
-1
0
1
2
3
4
5
6
Z = 6
Z = 5.4
(1, 1.6)


Z=8.2
(3.8, 3)
Z=7.6
(4, 2.9)
(3, 2.6)
Z=7.4
NF
(4, 3)
Z=4
(4, 2)
(3, 3)
NF
(1.8, 2)
Z=6.2
Z=6
(2, 2)
Z=5.4
(1, 1.6)

REMARKS: Best-First or Depth-First?

1. **Best-First Search**: We choose the subproblem whose relaxed solution has the best value (largest if a maximization problem, smallest if a minimization problem) among all those subproblems we have not branched from. This is because if we were to obtain a feasible solution in this subregion, we hope that its value would be very close to the generated upper bound. However, one **potential drawback** to this approach is that we may need to store a large number of subproblems at any one ti
2. **Depth-First Search**. We explore each subregion/node as if we were doing a depth-first search on the tree. In terms of the sub-problems, we examine one region completely before we proceed to examine any other. But, this tends to be the approach that takes the longest in terms of computational time. [7] ■

[7] However, this works well in combination with best-first search because we can use depth-first search to find an initial feasible point (which allows for pruning), and then use best-first search to complete the optimization.

Algorithm 3.2 Branch-and-Bound Algorithm ($\boldsymbol{c}, \mathbf{x}, A, \boldsymbol{b}$)

1. **Build LP relaxation and find its solution**: Omit the integrality constraints from the original problem's variables in order to obtain a **LP relaxation**, a LP problem is obtained and we solve it.
2. **Divide** a problem into sub-problems by branching at a good `active node` (does not have integral value solution and have not fathomed yet), see Equation (3.8)
3. **Calculate the LP relaxation** of a sub-problem
   - The LP problem has no feasible solution, **done**;
   - The LP problem has an integer optimal solution; **done**.
   - Compare the optimal solution with the **best solution we know (the incumbent)**.
   - Case A: The LP problem has an optimal solution that is worse than the incumbent, done.

     In all the cases above, we know all we need to know about that sub-problem. We say that **sub-problem is fathomed**.
   - Case B: The LP problem has an optimal solution that are **not all integer**, better than the incumbent. In this case we would have to go back Step 1, to divide this sub-problem further and repeat.

Practice an 0-1 Knapsack Problem Instance

- Consider the problem

$$\max_x \quad 8x_1 + 11x_2 + 6x_3 + 4x_4$$

$$\text{s.t.} \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14,$$

$$x \in \{0, 1\}^4.$$

- The linear relaxation solution is $x = (1, 1, \underline{0.5}, 0)$ with a value of 22.

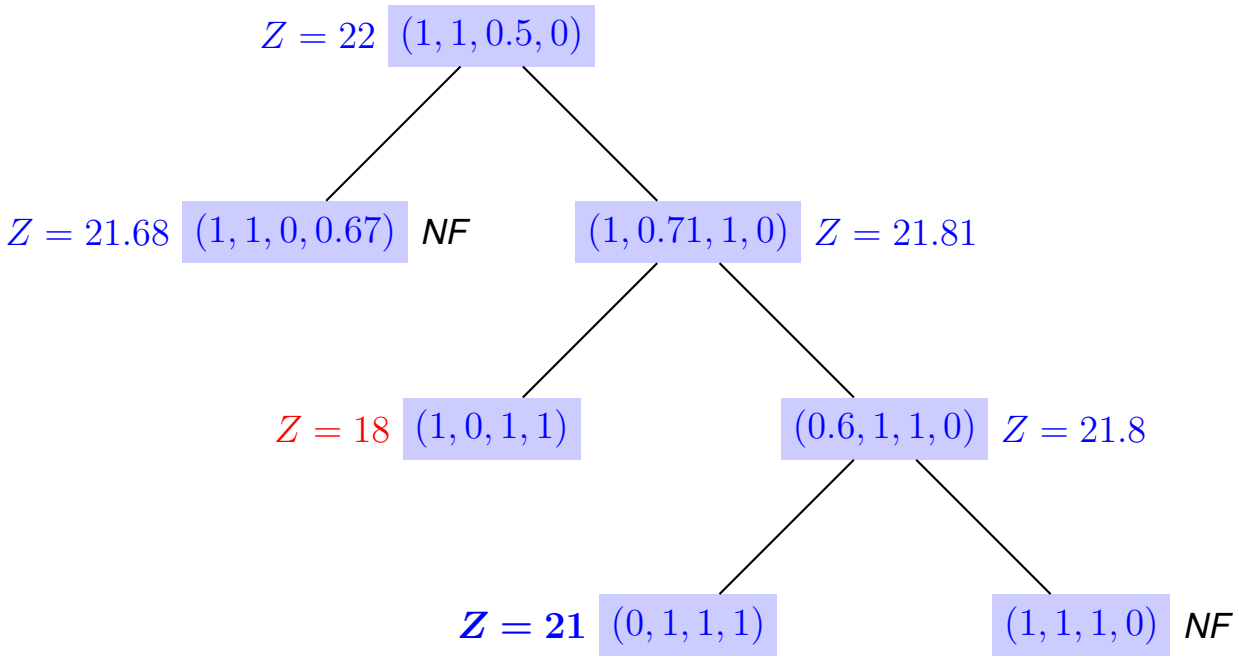
  The solution is not integral.

- Choose x_3 to branch. The next two sub-problems will have $\underline{x_3 = 0}$ and $\underline{x_3 = 1}$, respectively.

- ...

  Now we build up the tree search (a binary tree in fact!)

- The tree search:



- The optimal solution is $\boldsymbol{x = (0, 1, 1, 1)}$.

3.9.3 Summarized Key Points

A/ Rule of Fathoming: A sub-problem **is fathomed** when

1. The relaxation of the sub-problem has an optimal solution with $z < z^*$ where z^* is the current best solution;
2. The relaxation of the sub-problem has no feasible solution;
3. The relaxation of the sub-problem has an optimal solution that has all integer values (or all binary if it is an BIP).

B/ How to Branch? Find a fractional variable

- We want to divide the current problem into two or more sub-problems that are easier than the original. A commonly used branching method to get two sub-trees:

  Left $x_i \leq \lfloor x_i^* \rfloor$, and Right $x_i \geq \lceil x_i^* \rceil$, $\quad x_i^*$ is a fractional variable.

- Which variable to branch? A commonly used branching rule: Branch the most fractional variable.
- How do we know the time will the BB take to solve each sub-problem?
  - Answer: We don't know.

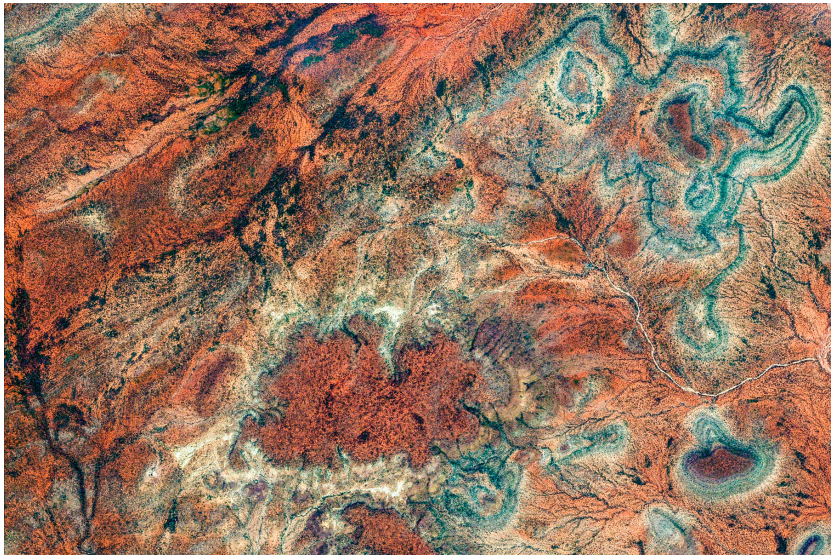
- Idea: Try to predict the difficulty of a sub-problem.

C/ New way of using CUTTING PLANES?

We can find better formulations of the feasible integer solutions, using CUTTING PLANES. To do this, we need to determine *additional constraints* called Valid Inequality.

Valid Inequality: for a set of solutions $\mathcal{S}$ is an inequality that is satisfied by all solutions in $\mathcal{S}$.

COFFEE BREAK



Question 1.

What possibly are good discrete mathematical modeling tools?

How does it work? How do we use it?

Mathematical Theory of Polyhedra with a few illustration models is discussed later in Section **??** of next chapter. The next two sections

J. CHAPTER 3' REVIEW and K. PRACTICAL ILP MODELS IN REAL WORLD

provide a grand review of Chapter 3, and the guidelines for preparing the midterm.

3.10 J. CHAPTER REVIEW: Popular problems revisited

3.10.1 Resource Allocation

We consider the allocation of scarce resources to optimize some function, typically the profit margin. A production company **C** designs three types of steel doors: *Standard*, *High Security*, and *Maximum Security*.

Each door requires different amounts of **machine** and **labor time**, and has different profit margins; this information is given in the following table:

Resource Allocation- Questions

a/ Each door must go through both machine 1 and machine 2 before it can be sold.

b/ Each worker is assigned to work on only one of the doors, which means they work on both machines.

c/ Firm **C** has available only 120 hours per week on machine 1 and 100 hours on machine 2 before

Table 3.5: *A specific resource allocation model in manufacturing*

	Machine 1 $\Longrightarrow$		Machine 2		
	Hours	**Manpower**	**Hours**	**Manpower**	**Profit margin**
Standard	3.5	5	4	6	**$ 35**
High Security	6	8	5	7	**$ 45**
Maximum Security	8	11	6	9	**$ 65**

required maintenance, and 280 hours of manpower available per week.

d/ In addition, management has decided not to sell more Maximum Security (best quality) doors than the combined total of Standard and High Security doors sold, in order **to keep supply high** for Standard and High Security doors in market.

If we assume that we can sell every door that we make, **how many of each door should be produced** each week in order to maximize profits?

Modeling

Our decisions are the number of doors to produce of each type, hence our variables are

$$x_1 = \text{the number of Standard doors produced,}$$

$$x_2 = \text{the number of High Security doors produced,}$$

$$x_3 = \text{the number of Maximum Security doors produced.}$$

Our profit margin then can be formulated as

$$\text{profit } P = f(x_1, x_2, x_3) = 35x_1 + 45x_2 + 60x_3.$$

Constraints: We determine the number of hours used by machine 1 and 2, and labor.

For machine 1, we need 3.5 hours for each Standard door, 6 hours for each High Security door, and 8 hours for each Maximum Security door. This implies a total time for machine 1 of $3.5\ x_1 + 6x_2 + 8x_3$. Since there are only 120 hours per week available on machine 1, due to Condition c/

we have the constraint

$$3.5\ x_1 + 6x_2 + 8x_3 \leq 120.$$

For machine 2, similarly reasoning, we have the 2nd constraint

$$4\ x_1 + 5x_2 + 6x_3 \leq 100.$$

Labor We quantify the total labor hour - manpower - to produce all door types

Each worker will work at both machines (by Condition b/), so the amount of labor time is calculated first for each machine and then added together (by Condition a/). Thus, the total labor time used per week to produce x_1 Standard doors, x_2 High Security doors, and x_3 Maximum Security doors is

$$(5+6)\ x_1 + (8+7)x_2 + (11+9)x_3 = 11\ x_1 + 15x_2 + 20x_3.$$

Since only 280 hours of labor are available per week, this gives us the constraint

$$11\ x_1 + 15x_2 + 20x_3 \leq 280.$$

Demand and supply trade-off Finally, due to Condition d/ we want to restrict the number of Maximum Security doors sold to market no more than the total combined number of Standard and High Security doors, leading to the constraint

$$x_3 \leq x_1 + \ x_2.$$

Comment: Note that the last constraint has variables on both the left- and right-hand sides. This is perfectly acceptable, since simple **algebra transforms** the constraint into one that has all the variables on the left-hand side. Since we must produce a non-negative amount of each type of door, we can combine all the above constraints and the profit calculations to produce the linear program given in Linear Program 1.

▼ Linear Program 1 (Resource Allocation Model).

$$\text{max profit } P = 35x_1 + 45x_2 + 60x_3$$

s.t.

$$\begin{cases} 3.5x_1 + 6x_2 + 8x_3 \leq 120 & (\text{machine 1}) \\ 4x_1 + 5x_2 + 6x_3 \leq 100 & (\text{machine 2}) \\ 11\,x_1 + 15x_2 + 20x_3 \leq 280 & (\text{total labor}) \\ x_3 \leq x_1 + x_2 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

Solution

Using the Simplex method gives the optimal solution of $x_1 = 22.85$ Standard and $x_3 = 1.42$ Maximum Security doors for a profit of $P_0 = \$885.7143$. Of course, it may seem that this situation does not satisfy all the assumptions needed to formulate it as a linear program, namely that the variables should, perhaps, have only **integer variables**.

Using Branch-and-Bound method [see Section 3.9.2]: If we add the constraint that the variables must be integers, the optimal solution is to produce 23 of Standard and 1 Maximum Security doors for an optimal profit of $P_1 = \$865$.

COMMENTS: Our optimal (*fractional*) solution uses all available labor hours, but more than 16 hours (out of the 120 hours) are still available on machine 1, and almost 10 hours are available on machine 2. These quantities can be calculated by taking **the differences** between the right-hand side and left-hand side of each constraint; for example, the slack amount of hours still available on

machine 1 can be found by

$$s_{machine\ 1} = 120 - (3.5\ x_1 + 6x_2 + 8x_3).$$

This tells us that our labor availability is truly constraining us from generating greater profit, while we are completely under- utilizing our machine usage.

3.10.2 Capital-Budgeting with m-resources

$m > 1$: # the number of resources, (time or budget) with total budget

$$b = \sum_{i=1}^{m} b_i$$

n: # the number of projects/ investments,

$$\max_{x} \qquad c^T \cdot \boldsymbol{x} = \sum_{j=1}^{n} c_j \, x_j$$

$$\text{s. t.} \qquad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad \forall \, i = 1, \cdots, m$$

$$\boldsymbol{x} \in \{0,1\}^n \ \text{\# meaning } x_j = 0 \ \text{ or } 1, \quad \forall \, j = 1, \cdots, n.$$

Now consider a specific case with $m = 3$ in the Capital Budgeting.

Capital Budgeting problem with data [see larger data in Model 4]

Four projects are being considered for execution over the next $m = \ 3$ years.

The expected returns in *net present value* c_j and yearly funds (resources) a_{ij} of each project j available per year i are tabulated below (unit is 1 *million dollar*):

Year	**Project**				**Available Funds**
$\downarrow$	1	2	3	4	
Year 1	$a_{11} = 5$	9	3	4	$b_1 = 21$
Year 2	2	6	5	3	$b_2 = 16$
Year 3	4	5	4	$a_{34} = 4$	$b_3 = 17$
Returns	$c_1 = 20$	$c_2 = 45$	$c_3 = 24$	$c_4 = 30$	**Max return = ?**

Hence $n = 4$ projects, $m = 3$; with total resource fund b_1 in year 1 is 21 million dollars, resource b_2 in year 2 is \$16 million, b_3 in year 3 is \$ 17 million,

c_i is the return of project i given in the last row.

Modeling phase in 3-STEPS

GOAL: to determine **which projects should be executed** over the next 3 years

such that at the end of the 3-year period, the total returns of the executed projects are at the maximum, subject to the availability of funds each year.

1. Identify the decision variables
2. Formulate the constraints
3. Formulate the objective function

1. Identify the decision variables

* Brainstorming first: Each project is either executed or rejected. Therefore, the problem reduces to a 'yes-no' decision for each project. Such a decision can be represented as a binary variable, where the value **1 means 'yes' and 0 means 'no.'**

* Delivering your choice: Should we define decision variables by

Option 1: For $j = 1, 2, 3, 4$ let

$$y_j = \begin{cases} 1, \text{ if project } j \text{ is selected} \\ 0, \text{ otherwise} \end{cases}$$

or use **Option 2**: For $i = 1, 2, 3$ and $j = 1, 2, 3, 4$ let

$$x_{ij} = \begin{cases} 1, \text{ if project } j \text{ is selected to implement in year } i \\ 0, \text{ otherwise?} \end{cases}$$

2. Formulate the constraints [We follow **Option 1**, why not Option 2?]

The constraints say that total annual expenditures of the selected projects **do not exceed** funds available for each year. Mathematically we express them as in matrix form $A\,\boldsymbol{y} \leq \boldsymbol{b}$, or explicitly

$$\sum_{j=1}^{n} a_{ij} y_j \leq b_i, \ i = 1, 2, 3.$$

For year 1, 2, 3 the constraints respectively are :

$$5y_1 + 9y_2 + 3y_3 + 4y_4 \leq 21,$$

$$2y_1 + 6y_2 + 5y_3 + 3y_4 \leq 16,$$

$$4y_1 + 5y_2 + 4y_3 + 4y_4 \leq 17.$$

Remind the data: c_j = expected returns of each project j, and

a_{ij}= fund for project j in year i

Year	**Project**				**Available Funds**
$\downarrow$	1	2	3	4	
Year 1	$a_{11} = 5$	9	3	4	$b_1 = 21$
Year 2	2	6	5	3	$b_2 = 16$
Year 3	4	5	4	$a_{34} = 4$	$b_3 = 17$
Returns	$c_1 = 20$	$c_2 = 45$	$c_3 = 24$	$c_4 = 30$	**Max return = ?**

3. Formulate the objective function

Aim: To maximize the *total returns* of the selected projects at the end of the 3-year planning period. The (optimal) total of returns is

$$Z = \max_{\boldsymbol{y}} \quad c^T \cdot \boldsymbol{y} = \sum_{j=1}^{n=4} c_j \, y_j = 20y_1 + 45y_2 + 24y_3 + 30y_4$$

Remark: If we employ Option 2, hope to get better understanding with 12 variables $\boldsymbol{x} =$

$(x_{11}, x_{12}, x_{13}, x_{14}, \cdots, x_{33}, x_{34})$ and then get the objective

$$T = \max_{\boldsymbol{x}} \quad c^T \cdot \boldsymbol{x} = \sum_{i=1}^{m=3} \sum_{j=1}^{n=4} c_{ij}\, x_{ij}?$$

But what are the coefficients c_{ij} then?

Capital budgeting problem gives the Pure Binary LP model

$$\text{maximize } Z = \sum_{j=1}^{n=4} c_j\, y_j$$

$$= 20y_1 + 45y_2 + 24y_3 + 30y_4$$

s.t.

$$\begin{cases} 5y_1 + 9y_2 + 3y_3 + 4y_4 \leq 21 & \text{for year 1} \\ ? & \text{for year 2} \\ ? & \text{for year 3} \\ y_j \in \{0,1\}, \quad j = 1,2,3,4. \end{cases}$$

Use the Branch-and-Bound method [see Section 3.9.2] to solve this BIP: DIY.

3.10.3 Summary of various constraints in ILP

1. **Constraints in combinatorial types**:

   These indicate that at least one of the variables must have value 1. Each of these cases appear so often that they have specific names associated with them:

   (1) covering constraints, (2) packing constraints, and (3) partitioning constraints.

   Given a collection C of choices, where we define variable

$$x_k = \begin{cases} 1 & \text{if choice } k \text{ is selected} \\ 0 & \text{otherwise,} \end{cases}$$

   a **covering constraint** is of the form

$$\sum_{i \in C} x_i \geq 1;$$

   a **packing constraint** is of the form $\sum_{i \in C} x_i \leq 1;$

and a **partitioning constraint** is of the form $\sum_{i \in C} x_i = 1.$

2. **Inventory- type constraints when time t involves**

Most importantly and new in this production type, we pay attention to the **inventory constraints** that have the recursive form

$$I_t = I_{t-1} + C_t - D_t$$

for each time period t, where

- I_t, D_t respectively are the inventory level and the demand for period t,
- C_t is the amount of product generated during period t, and

Care must be taken for the initial time period, however; we typically define the initial inventory level I_0, even though there is no 'real' period 0.

See MODEL 3 (Production and Inventory) for Inventory- type constraints.

3.11 K. PRACTICAL ILP MODELS IN REAL WORLD

We suggest further case studies for team presentations or self-study as follows [e.g. ref. [**Ravindran**] and [**Chinneck**] for Models 1-5] .

1. MODEL 1 **Regional Planning for Agriculture**
2. MODEL 2 **Task Assignment in College**
3. MODEL 3 **Production and Inventory**

4. MODEL 4 (Capital Budgeting problem)

MODEL 1 (Regional Planning for Agriculture).

The SOUTHERN CONFEDERATION OF KIBBUTZIM (SCK) is a group of three kibbutzim (*communal farming communities*) in **Israel**. Overall planning for this group is done by planning agricultural production for the coming year. The agricultural output of each **kibbutz** is limited by both the amount of available irrigable land and the quantity of water allocated for irrigation by the Water Commissioner (a national government official). These data are given in Table 3.6.

Table 3.6: *Resource data for the Southern Confederation of Kibbutzim*

Kibbutz	Usable Land (hectare)	Water Allocation (m^3)
1	400	600
2	600	800
3	300	375

The crops suited for this region include sugar beets, cotton, and sorghum, and these are the three being considered for the upcoming season. These crops differ primarily in their expected net return per acre and their consumption of water. In addition, the Ministry of Agriculture has set

a maximum quota for the total acreage that can be devoted to each of these crops by the SCK, see Table 3.7

Table 3.7: *Crop data for the Southern Confederation of Kibbutzim*

Crop	**Maximum quota** (Acres)	Water consumption (m^3/ hectare)	Net return ($/ hectare)
Sugar beets	600	3	1,000
Cotton	500	2	750
Sorghum	325	1	250

Because of the limited water available for irrigation, the Southern Confederation of Kibbutzim will not be able to use all its irrigable land for planting crops in the upcoming season. To ensure equity between the three kibbutzim, it has been agreed that every kibbutz will plant the same proportion of its available irrigable land. [8] The job facing the Coordinating Technical Office is to plan how many acres to devote to each crop at the respective kibbutzim while satisfying the given

[8] For example, if kibbutz 1 plants 200 of its available 400 acres, then kibbutz 2 must plant 300 of its 600 acres, while kibbutz 3 plants 150 acres of its 300 acres. However, any combination of the crops may be grown at any of the kibbutzim.

restrictions.

The objective is to maximize the total net return to the Southern Confederation of Kibbutzim as a whole, use a Linear Programming model.

MODEL 2 (Task Assignment in College).

Senior design students are trying to determine which person will take primary responsibility for the remaining tasks the team must complete. Below is a table of the amount of time each student would need to complete the corresponding task, in hours.

Table 3.8: *Task assignment*

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
Student 1	12	5	**8**	9	6	**11**
Student 2	14	8	**7**	11	10	**5**
Student 3	10	9	**9**	8	7	**8**
Student 4	11	8	**10**	10	9	**10**

If each student must do at least one but no more than two tasks, how should the tasks be divided so as to minimize the total amount of time required to finish all the tasks? Hint: formulates this problem as a Linear / Integer Programming.

GUIDANCE for solving. How to define decision variables?

$n = 6$: # tasks, $m = 4$: # students,

c_{ij}: amount of time student i needs to complete task j

$$x_{ij} = \begin{cases} 1, \text{ if student } i \text{ is assigned to task } j, \\ 0, \text{ otherwise.} \end{cases}$$

Each of variables can have only the values 0 or 1. The objective function is

$$Z = \sum_{i=1} \sum_{j=1} c_{ij}\, x_{ij}.$$

With these decision variables, we have the constraints that

(1) each student can be assigned to either one or two jobs, and

(2) each job must be assigned to at least one student.

Constraint (1) implies that

$$1 <= \sum_{j=1} x_{ij} \leq 2, \quad i?$$

Constraint (2) implies that

$$\sum_{i=1} x_{ij} \geq 1, \quad j?$$

MODEL 3 (Production and Inventory).

American Engine Co. (AEC) produces two engines, one for trucks and one for cars. During the next 3 months, they anticipate the following demands for their engines:

	Month 1	Month 2	Month 3
Truck engines	400	300	500
Car engines	800	500	600

No backlogging is allowed, which means that each month's demand must be fully satisfied. During each month, at most 1000 engines (combined) can be produced. Each truck engine requires 10 hours of labor to produce and costs $\$2000$ in supplies, while each car engine requires 8 hours of labor and costs $\$1500$ in supplies. At most 9000 hours are available each month.

At the beginning of month 1, 100 truck engines and 200 car engines are in inventory. At the end of each month, a holding cost of $\$150$ per engine is assigned to any engine in inventory. At the

end of the third month, management wants to have at least 100 of each engine in inventory.

How can we meet monthly demand at a minimum cost? Each month, how many of each engine is produced and how many engines are placed in inventory?

MODEL 4 (Capital Budgeting problem).

Six projects are being considered for execution over the next $m = 3$ years. The expected returns in *net present value* and yearly expenditures (funds) of each project available per year are tabulated below (all units are *million dollars*):

Project	1	2	3	4	5	6	**Available Funds**
Year 1	5	9	3	4	3	2	26
Year 2	2	6	5	3	3	3	22
Year 3	4	5	4	4	5	3	25
Returns	20	45	24	30	12	32	**Max return = ?**
(Contributions)							

Hence $n = 6$ projects, with total fund b_1 in year 1 is 26 million dollars, b_2 in year 2 is 22 million dollars, b_3 in year 3 is 25 million dollars, $m = 3$; c_j is the return of project j given in the last row, for $j = 1, 2, \cdots, 6$.

QUESTIONS: The problem seeks to determine which projects should be executed over the next 3 years such that at the end of the 3-year period, **the total returns of the executed projects are at the maximum**, subject to the availability of funds each year. Determine the **ILP** model for this

capital budgeting problem? And solve it by Maple ...

HINT:

Identify the decision variables. Each project is either executed or rejected. Therefore, the problem reduces to a 'yes-no' decision for each project. Such a decision can be represented as a binary variable, where the value **1 means 'yes' and 0 means 'no.'** Should we define

$$y_j = \begin{cases} 1, \text{ if project } j \text{ is executed} \\ 0, \text{ if project } j \text{ is rejected} \end{cases}$$

or define

$$x_{ij} = \begin{cases} 1, \text{ if project } j \text{ is selected to implement in year } i \\ 0, \text{ otherwise?} \end{cases}$$

Formulate the constraints. These constraints can be expressed as follows, for year 1:

$$5y_1 + 9y_2 + 3y_3 + 4y_4 + 3y_5 + 2y_6 \leq 26 = b_1$$

WHY? How about year 2 and 3?

Formulate the objective function. The objective of this problem is to maximize the total returns of the selected projects at the end of the 3-year planning period.

The total returns are given mathematically as

$$Z = \max_{\boldsymbol{y}} \quad c^T \cdot \boldsymbol{y} = \sum_{j=1}^{n=6} c_j \, y_j = 20y_1 + 45y_2 + \cdots + 32y_6.$$

The ILP model for this capital budgeting problem is

$$\begin{aligned} \text{maximize } Z &= \sum_{j=1}^{n=6} c_j \, y_j = 20y_1 + 45y_1 + \cdots + 32y_6 \\ &= 12 * x_{11} + 5 * x_{12} + ... + 10 * x_{46} \end{aligned}$$

$$
s.t. \begin{cases}
5y_1 + \; 9y_2 + \; 3y_3 + \; 4y_4 + \; 3y_5 + \; 2y_6 \leq 26 & \text{for year 1} \\
? & \text{for year 2} \\
? & \text{for year 3} \\
y_j \in \; \{0,1\}, \; j \in \{1,2,3,4,5,6\} & .
\end{cases}
$$

You have to use the **LP relaxation model** in Lingo or Maple ... by imposing the upper bounds $x_j \leq 1$ (of original decision variables y_j), for all j.

COMMENTS: The optimal solution is

$y_1 = y_2 = y_5 = 1, y_3 = 0.7796$, $y_4 = 0.7627$, and $y_6 = 0.0678$

with an objective value of $101.61 million. This solution has no meaning to the **ILP**, and rounding to the closest integer values leads to an **infeasible solution**. The optimal integer solution is $y_1 = y_2 = y_3 = y_4 = 1$ and $y_5 = y_6 = 0$ with $Z = \$96$ million.

3.12 Demonstration: Solving Sudoku [Extra reading]

3.12.1 Sudoku Game

Problem statement

Given a grid of 9×9, the grid is divided into 9 subgrids of 3×3. Some cells of subgrids are filled by digits in $\{1, 2, \ldots, 9\}$. The objective is to fill remaining cells such that

- each column, each row, and each subgrid that compose the grid, contain all of the digits from 1 to 9.

A Sudoku Grid and Variables

$$\text{Variables } V = \{X_{ijk} \mid 1 \leq i, j, k \leq 9\}$$

- X_{ijk} true iff cell at row i column j equals k.
- $|V| = 9^3 = 729$
- X_{726} is true

- X_{72k} is false for $k \neq 6$

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

Sudoku

Constraining exactly one variable to be true

Variables = $\{p, q, r, s\}$

- At least one is true: (use the OR operator)

$$\alpha = p \vee q \vee r \vee s$$

- No more than one is true:

(now combine with the AND operator, pairwise)

$$\beta = (\bar{p} \vee \bar{q}) \wedge (\bar{p} \vee \bar{r}) \wedge (\bar{p} \vee \bar{s}) \wedge$$

$$(\bar{q} \vee \bar{r}) \wedge (\bar{q} \vee \bar{s}) \wedge (\bar{r} \vee \bar{s})$$

- Exactly one is true (now the AND operator again)

$$\psi = \alpha \wedge \beta$$

Sudoku row 2 contains exactly one 8

Remember:

$X_{2j8} = true$ if the cell at row 2, column j receives number 8.

$\overline{X}_{2j8} = true \iff X_{2j8} = ?$

- Row 2 must contain at least one 8

$$\alpha_{2,8} = \bigvee_{1 \leq j \leq 9} X_{2j8}$$

- Row 2 has at most one 8

$$\beta_{2,8} = \bigwedge_{\substack{1 \leq j,m \leq 9 \\ j \neq m}} \left(\bar{X}_{2j8} \vee \bar{X}_{2m8} \right)$$

- Row 2 has exactly one 8

$$\psi_{2,8} = \alpha_{2,8} \wedge \beta_{2,8}$$

Sudoku row constraints

Remember:

$\psi_{2,k} = true$ if and only if row 2 has exactly one number k.

- Row 2 contains all 9 values exactly once

$$\gamma_2 = \bigwedge_{1 \le k \le 9} \psi_{2,k}$$

- All 9 rows contain all 9 values exactly once

$$\begin{aligned} R &= \bigwedge_{1 \le i \le 9} \gamma_i = \bigwedge_{1 \le i \le 9} \bigwedge_{1 \le k \le 9} \psi_{i,k} \\ &= \bigwedge_{1 \le i \le 9} \bigwedge_{1 \le k \le 9} (\alpha_{i,k} \wedge \beta_{i,k}) \\ &= \bigwedge_{1 \le i \le 9} \bigwedge_{1 \le k \le 9} \left[\left(\bigvee_{1 \le j \le 9} X_{ijk} \right) \wedge \left(\bigwedge_{\substack{1 \le j,m \le 9 \\ j \ne m}} (\bar{X}_{ijk} \vee \bar{X}_{imk}) \right) \right] \end{aligned}$$

Column constraints

- All 9 columns contain all 9 values exactly once

$$C = \bigwedge_{1\le j\le 9}\bigwedge_{1\le k\le 9}\left[\left(\bigvee_{1\le i\le 9} X_{ijk}\right)\wedge\left(\bigwedge_{\substack{1\le i,m\le 9\\ i\neq m}}(\bar{X}_{ijk}\vee\bar{X}_{mjk})\right)\right]$$

3×3 box constraints

- 3×3 box containing cell $(4,7)$ has at least one 5

$$\xi_{475} = \bigvee_{\substack{i=4,5,6\\ j=7,8,9}} X_{ij5}$$

- 3×3 box containing cell $(4,7)$ has at most one 5

$$\zeta_{475} = \bigwedge_{\substack{i,m=4,5,6\\ j,n=7,8,9\\ i\neq m\vee j\neq n}} (\bar{X}_{ij5}\vee\bar{X}_{mn5})$$

3×3 box constraints, cont. . .

- 3×3 box containing cell $(4, 7)$ has exactly one 5

$$\theta_{475} = \xi_{475} \wedge \zeta_{475} ??$$

- All 9 3×3 boxes contains exactly one 5

$$\mu_5 = \bigwedge_{\substack{i=1,4,7 \\ j=1,4,7}} \theta_{ij5}$$

$$\begin{aligned} \mu_5 \;=\; & \theta_{1,1,5} \wedge \theta_{1,4,5} \wedge \theta_{1,7,5} \\ & \theta_{4,1,5} \wedge \theta_{4,4,5} \wedge \theta_{4,7,5} \\ & \theta_{7,1,5} \wedge \theta_{7,4,5} \wedge \theta_{7,7,5} \end{aligned}$$

- All 9 3×3 boxes contain all 9 values

$$B = \bigwedge_{1 \le k \le 9} \mu_k = \bigwedge_{1 \le k \le 9} \bigwedge_{\substack{i=1,4,7 \\ j=1,4,7}} \theta_{ijk}$$

Initial predefined values $I = X_{115} \wedge X_{123} \wedge \cdots \wedge X_{999}$

3.12.2 Illustration Sudoku Game by ILP model

INPUT

Given a grid of 9×9, the grid is divided into 9 subgrids of 3×3.

Some cells of subgrids are filled by digits in $\{1, 2, \ldots, 9\}$.

OUTPUT - REQUEST

The objective is to fill remaining cells such that

- each column, each row, and
- each subgrid (that compose the grid) all contain all of the digits from 1 to 9.

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

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

Sudoku Game: An IP Model

$$\text{Variable } x_{ijk} = \begin{cases} 1 & \text{if cell } (i,j) \text{ contains number } k; \\ 0 & \text{otherwise.} \end{cases}$$

$$\min_x \quad \mathbf{0}^T \boldsymbol{x}$$

subject to

$$\sum_{i=1}^{9} x_{ijk} = 1, \quad j,k = 1,\ldots,9,$$

$$\sum_{j=1}^{9} x_{ijk} = 1, \quad i,k = 1,\ldots,9,$$

$$\sum_{k=1}^{9} x_{ijk} = 1, \quad i,j = 1,\ldots,9,$$

$$\sum_{j=3q+2}^{3q} \sum_{i=3p+2}^{3p} x_{ijk} = 1, \quad k = 1,\ldots,9; p,q = 1,2,3,$$

$$x_{ijk} \in \{0,1\}, \quad x_{ijk} = 1, \quad \forall (i,j,k) \in G.$$

3.12.3 Software for Integer Programming

- Matlab optimization toolbox: `bintprog` – a built-in function for mixed integer linear programming.

- Some of softwares which are proprietary but *free for academic use*:
  - CPLEX: www-03.ibm.com/.../ibmilogcpleoptistud
  - Gurobi: gurobi.com
  - MOSEK: www.mosek.com
  - SCIP: scip.zib.de

  These softwares can also be used as solvers in some modeling tools.
- Many open-source solvers were developed in the literature.

Software for ILP: Modeling Tools

- CVX: cvxr.com/cvx
- YALMIP: yalmip.github.io
- TOMLAB: tomopt.com/tomlab

SUMMARY

MODELING: Students should try to solve the following problems in 3-steps:

(1) Modeling [transforming the given practical problem to a LP or ILP]

(2) Using the Simplex method or **Branch-and-Bound method** to solve

(3) Writing the conclusion properly, with comments if possible.

There are few practical problems in **K**. with hints/guidelines to modeling,

(I) Resource allocation [denoted Model 3 in Lecture **D**]

(II) Capital Budgeting [general knapsack problem with BIP in Lecture **G**]

(III) Summary of constraints in ILP

SELF-SOLVING EXERCISES

- Read carefully and do all Exercises in Chapter 7 of [4]: F.R. Giordano, W.P. Fox and S.B. Horton, A First Course in Mathematical Modeling, 5th ed., Cengage, 2014. Do all Exercises in the file named "BT Chuong 3.pdf".
- More problems are given in next section **L.**, without hints.

3.13 COMPLEMENT: Programming Language LINGO

Introduction to LINGO, a language for optimization

A comprehensive modeling language and solvers for linear, non-linear, and integer programming. It is a comprehensive tool designed to make building and solving

- Linear, Nonlinear (convex & non-convex/ Global),

- Quadratic, Quadratically Constrained, Second Order Cone,

- Semi-Definite, Stochastic, and Integer optimization models

fast, easy and efficient.

- LINGO provides a completely integrated package that includes a powerful language for expressing optimization models, a full featured environment for building and editing problems, and a set of fast built-in solvers.
- LINGO is designed mainly for efficiently formulating **very large models** by simultaneously dealing with all constraints or variables of the same type. See next section for more.

ILLUSTRATION

Simple problems are entered into LINGO in a fairly natural fashion. To illustrate, consider the following linear programming problem.

Resource allocation with mathematical modeling phase.

STEP 1: Problem A production company **C** designs three types of steel doors: *Standard*, *High Security*, and *Maximum Security*. Each door requires different amounts of **machine** and **labor time**, and has different profit margins.

STEP 2: Data this information is given in the following table:

Table 3.9: *A specific resource allocation model in manufacturing*

	Machine 1 $\Longrightarrow$		**Machine 2**		
	Hours	**Manpower**	**Hours**	**Manpower**	**Profit margin**
Standard	3.5	5	4	6	**$ 35**
High Security	6	8	5	7	**$ 45**
Maximum Security	8	11	6	9	**$ 65**

CONDITIONS:

a/ Each door must go through both machine 1 and machine 2 before it can be sold.

b/ Each worker is assigned to work on only one of the doors, which means they work on both machines.

c/ Firm **C** has available only 120 hours per week on machine 1 and 100 hours on machine 2 before required maintenance, and 280 hours of manpower available per week.

d/ In addition, management has decided not to sell more Maximum Security (best quality) doors than the combined total of Standard and High Security doors sold, in order **to keep supply high** for Standard and High Security doors in market.

STEP 3: Modeling Decisions are the number of doors to produce of each type, hence our variables are

$$x_1 = \text{the number of Standard doors produced,}$$

$$x_2 = \text{the number of High Security doors produced,}$$

$$x_3 = \text{the number of Maximum Security doors produced.}$$

Our profit margin then can be formulated as

$$\text{profit } P = f(x_1, x_2, x_3) = 35x_1 + 45x_2 + 60x_3.$$

Constraints: We determine the number of hours used by machine 1 and 2, and labor.

For machine 1, we need 3.5 hours for each Standard door, 6 hours for each High Security door, and 8 hours for each Maximum Security door. This implies a total time for machine 1 of $3.5\ x_1 + 6x_2 + 8x_3$. Since there are only 120 hours per week available on machine 1, due to Condition c/ we have the constraint $3.5\ x_1 + 6x_2 + 8x_3 \leq 120.$

For machine 2, similarly reasoning, we have the 2nd constraint

$$4\ x_1 + 5x_2 + 6x_3 \leq 100.$$

Labor We quantify the total labor hour - manpower - to produce all door types

Each worker will work at both machines (by Condition b/), so the amount of labor time is calculated first for each machine and then added together (by Condition a/). The total labor time used per week to produce x_1 Standard doors, x_2 High Security doors, and x_3 Maximum Security doors so

is

$$(5+6)\,x_1+(8+7)x_2+(11+9)x_3=11\,x_1+15x_2+20x_3.$$

Since only 280 hours of labor are available per week, this gives us the constraint

$$11\,x_1+15x_2+20x_3\le 280.$$

Demand and supply trade-off Finally, due to Condition d/ we want to restrict the number of Maximum Security doors sold to market no more than the total combined number of Standard and High Security doors, leading to the constraint

$$x_3\le x_1+\,x_2.$$

Resource Allocation Model

$$\text{max profit } P=35x_1+45x_2+60x_3$$

s.t.

$$\begin{cases} 3.5x_1+ & 6x_2+ & 8x_3 & \le & 120 & (\text{ machine 1})\\ 4x_1+ & 5x_2+ & 6x_3 & \le & 100 & (\text{ machine 2})\\ 11\,x_1+ & 15x_2+ & 20x_3 & \le & 280 & (\text{ total labor})\\ & & x_3 & \le & x_1+x_2 & \\ & & x_1,x_2,x_3 & \ge & 0 & \end{cases}$$

STEP 4: Computation with Lingo

Sentences after the exclamation ! is for engineers, for human-being. The software skips all information after exclamations !, but you must use ; in those lines. Hence Lingo will execute commands in lines 3, and 5 to 9.

```
!EXAMPLE on Resource Allocation;
!Here is the profit objective function of firm C;
MAX = 35 * Standard + 45 * HighSecurity + 60 * MaxSecurity;

!Constraints on machines and workers:
3.5 * Standard + 6 * HighSecurity + 8* MaxSecurity <= 120;
4 * Standard + 5 * HighSecurity + 6* MaxSecurity <= 100;
11 * Standard + 15 * HighSecurity + 20* MaxSecurity <= 280;

MaxSecurity <= Standard + HighSecurity;
Standard >= 0; HighSecurity >= 0; MaxSecurity >=0;
```

Key features and working mechanism of LINGO

LINGO has 4 solvers used to solve different types of models. These solvers are:

· a direct solver,

· a linear solver,

· a nonlinear solver, and

· a branch-and-bound manager.

Working mechanism:

- When you solve a model, the direct solver first computes the values for as many variables as possible. If the direct solver finds an equality constraint with only one unknown variable, it determines a value for the variable that satisfies the constraint.
- The direct solver stops when it runs out of unknown variables or there are no longer any equality constraints with a single remaining unknown variable.
- Once the direct solver is finished, if all variables have been computed, LINGO displays the solution report.

1. If **unknown variables** remain, LINGO determines which solvers to use on a model by examining its structure and mathematical content.
2. For a **continuous** linear model, LINGO calls the linear solver.
3. If the model contains one or more **nonlinear constraints**, LINGO calls the nonlinear solver.
4. When the model contains any **integer restrictions**, the branch-and-bound manager is invoked to enforce them. The **branch-and-bound** manager will call either the linear or nonlinear solver depending upon the nature of the model.

LINGO output of our example

Global optimal solution found.

Objective value: 885.7143

Infeasibilities: 0.000000

Total solver iterations: 3

Elapsed runtime seconds: 0.50

Model Class: LP

Total variables: 3

Nonlinear variables: 0

Integer variables: 0

Total constraints: 8

Nonlinear constraints: 0

Total nonzeros: 18

Nonlinear nonzeros: 0

===

Variable Value Reduced Cost

STANDARD 22.85714 0.000000

HIGHSECURITY 0.000000 1.428571

MAXSECURITY 1.428571 0.000000

Row Slack or Surplus Dual Price

1 885.7143 1.000000

2 28.57143 0.000000

3 0.000000 2.857143

4 0.000000 2.142857 ...

The optimal value is $P = \$885.7143$, corresponding with the optimal solution (but not integers) $(x_1, x_2, x_3) = (22.85, 0, 1.42)$.

Here variable STANDARD= $x_1 = 22.85$ stands at row 6, ... the optimal value is P has **Dual Price** 1 USD.

List of Figures

List of Tables

Bibliography

[1] ALEXANDER HOLMES, Introductory Business Statistics, OpenStax, Rice University, 2017

[2] Annette J. Dobson and Adrian G. Barnett,
An Introduction to Generalized Linear Models, Third Edition, CRC (2008)

[3] Antal Kozak, Robert A. Kozak, Christina L. Staudhammer, Susan B. Watts *Introductory Probability and Statistics Applications for Forestry and Natural Sciences*, CAB (2008)

[4] *Canvas paintings* by Australian artists of ethnic minorities, Australian National Museum

[5] John J. Borkowski's Home Page, *www.math.montana.edu/ jobo/courses.html/*

[6] David S. Moore, George P. McCabe and Bruce A. Craig, 2009. *Introduction to the Practice of Statistics*, 6th edition, W. Freeman Company, New York

[7] Man Nguyen, 2018. *Statistical Data Analysis I*, 1st edition, Mahidol University

[8] S.R. Dalai and al., *Factor-covering designs for Testing Software, Technometrics* 40(3), 234-243, American Statistical Association and the American Society for Quality, 1998.

[9] Douglas C. Montgomery, George C. Runger,
Applied Statistics and Probability for Engineers, Sixth Edition, (2014) John Wiley & Sons

[10] Jay L. Devore and Kenneth N. Berk,
Modern Mathematical Statistics with Applications, 2nd Edition, Springer (2012)

[11] M. F. Fecko and al., *Combinatorial designs in Multiple faults localization for Battlefield networks, IEEE Military Communications* Conf., Vienna, 2001.

[12] Glonek G.F.V. and Solomon P.J. *Factorial and time course designs for cDNA microarray experiments, Biostatistics* **5**, 89-111, 2004.

[13] Hedayat, A. S., Sloane, N. J. A. and Stufken, J. *Orthogonal Arrays, Springer-Verlag*, 1999.

[14] Robert V. Hogg, Joseph W. McKean, Allen T. Craig *Introduction to Mathematical Statistics,* Seventh Edition *Pearson*, 2013.

[15] Paul Mac Berthouex, Linfield C. Brown, *Statistics for Environmental Engineers*, 2nd Edition, LEWIS PUBLISHERS, CRC Press, 2002

[16] Michael Baron, *Probability and Statistics for Computer Scientists*, 2nd Edition (2014), CRC Press, Taylor & Francis Group

[17] R. H. Myers, Douglas C. Montgomery and Christine M. Anderson-Cook
Response Surface Methodology : Process and Product Optimization Using Designed Experiments, Wiley, 2009.

[18] Man Nguyen, Tran Vinh Tan and Phan Phuc Doan,
Statistical Clustering and Time Series Analysis for Bridge Monitoring Data, Recent Progress in Data Engineering and Internet Technology,
Lecture Notes in Electrical Engineering 156, (2013) pp. 61 - 72, Springer-Verlag

[19] Man Nguyen and Le Ba Trong Khang.
Maximum Likelihood For Some Stock Price Models, Journal of Science and Technology, Vol. 51, no. 4B, (2013) pp. 70- 81, VAST, Vietnam

[20] Nguyen Van Minh Man,
Computer-Algebraic Methods for the Construction of Designs of Experiments, Ph.D. thesis

[21] Nguyen, Man V. M. *Some New Constructions of strength 3 Orthogonal Arrays*,
the Memphis 2005 Design Conference Special Issue of the **Journal of Statistical Planning and Inference**, Vol 138, Issue 1 (Jan 2008) pp. 220-233.

[22] Nathabandu T. Kottegoda, Renzo Rosso.
Applied Statistics for Civil and Environmental Engineers, 2nd edition (2008), Blackwell Publishing Ltd and The McGraw-Hill Inc

[23] Man Nguyen (2005) *Computer-algebraic Methods for the Construction of Design of Experiments*, Ph.D thesis, Eindhoven Univ. Press, 2005

[24] Paul Mac Berthouex. L. C. Brown. *Statistics for Environmental Engineers*; 2nd edition (2002), CRC Press

[25] Ron S. Kenett, Shelemyahu Zacks.
Modern Industrial Statistics with applications in R, MINITAB, 2nd edition, (2014), Wiley

[26] Sheldon M. Ross. *Introduction to probability models*, 10th edition, (2010), Elsevier Inc.

[27] Sloane N.J.A., `http://neilsloane.com/hadamard/index.html/`

[28] Google Earth, Digital Globe, 2014- 2019

[29] Vo Ngoc Thien An, Design of Experiment for Statistical Quality Control, Master thesis, LHU, Vietnam (2011)

[30] Larry Wasserman, *All of Statistics- A Concise Course in Statistical Inference*, **Springer**, (2003)

[31] C.F. Jeff Wu, Michael Hamada *Experiments: Planning, Analysis and Parameter Design Optimization,* Wiley, 2000.