1.3 Knapsack and Assignment Problems

An integer programming problem in which all the variables must equal 0 or 1 is called a 0-1 IP, or a Binary Problem. In practice 0-1 variables are commonly used to capture the logical (TRUE, FALSE) arguments. التعبير المنطقي

The Knapsack Problem

Example StockCo is considering four investments, which have to be invested as a whole if selected. A total of 14, 000 is available.

Investment	1	2	3	4
cost	5,000	7,000	4,000	3,000
NPV	16,000	22,000	12,000	8,000

Formulate an IP whose solution will tell StockCo how to maximize the (net present value) NPV obtained from investment 1-4. (صافي القيمة الحالية)

SOLUTION
Step 1: Definition of the Decision Variables

Let $x_j = \begin{cases} 1, & if\ investment\ \ j\ \ is\ made \\ 0, & otherwise \end{cases}$, j = 1, 2, 3, 4.

Step 2: Define the Constraints
At most 14, 000 can be invested (in thousands of pounds):

$$5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$$
$$x_1, x_2, x_3, x_4 \in \{0, 1\}.$$

Step 3: Define the Objective Function

$$\max \ \ 16x_1 + 22x_2 + 12x_3 + 8x_4$$

A Knapsack Model is a pure integer linear program with a single main constraint. The knapsack name derives from the problem confronted by a hiker packing a backpack. He or she must choose the most valuable collection of items to take subject to a volume or weight limit on the size of the pack.

In general, **a Knapsack Model** is given by

$$\max \sum_{j \in N} v_j x_j$$
$$s.t. \sum_{j \in N} c_j x_j \leq b, \quad \text{(capacity conditions)}$$
$$x \in \{0,1\}^n$$

where v_j denotes the *value* of item *j*; c_j denotes the *"cost"* of including item *j* in pack where a maximum of *b* is allowed.

Remarks
• Knapsack decision variables are 0-1 in most applications. However, if multiple units of each item type can be packed, we have $x \in Z^n$.
• Despite its name, the $0-1$ **Knapsack Model**, can be seen in many applications e.g. the capital budgeting problem.
• Although the Knapsack problem is indeed known as the simplest integer (linear) programs, its solution is not that straightforward. Special purpose solution methods will be discussed in later sections.

The Assignment Problem

Suppose there are *n* machines and *m* (parallel) jobs, with $n \geq m$. A job is only carrying out on a single machine and a machine can do only one job at a time. If the cost of doing job *j* by machine *i* is c_{ij}, then what is the optimal assignment of jobs to machines to complete all jobs while minimizing cost?

Step 1: Definition of the Decision Variables

Let $x_j = \begin{cases} 1, & if\ job\ \ j\ is\ assigned\ to\ machine\ \ i \\ 0, & otherwise \end{cases}$,

$$\forall i = 1, \dots, n, \ \ \forall j = 1, \dots m.$$

Step 2: Define the Constraints
Since exactly one machine is to assign to job *j*:

$$s.t. \quad \sum_{i=1}^{n} x_{ij} = 1, \ \ \forall j = 1, \dots, m \quad (job\ conditions)$$

Also each machine *i* can do no more than one job at a time:

$$\sum_{j=1}^{m} x_{ij} \leq 1, \quad \forall i = 1, \dots, n \qquad (machine\ conditions)$$

Step 3: Define the Objective Function

$$\min \qquad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}$$

Therefore, the entire optimization problem is written as

$$\min \qquad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}$$

$$s.t. \qquad \sum_{i=1}^{n} x_{ij} = 1, \quad \forall j = 1, \dots, m \qquad (job\ conditions)$$

$$\sum_{j=1}^{m} x_{ij} \leq 1, \quad \forall i = 1, \dots, n \qquad (machine\ conditions)$$

$$x \in \{0,1\}^{m \times n}$$

Remarks

• If $m = n$, no machine can go idle in operations, then the machine conditions become:

$$\sum_{j=1}^{m} x_{ij} = 1, \quad \forall i = 1, \dots, n$$

• There are also **Assignment Problems** with nonlinear objectives such as **Quadratic Assignment Problem**.

• If we are pairing objects in ONE set instead of two distinct sets, we end up with the so called **Matching Problem**:

$$\min \qquad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$s.t. \qquad \sum_{i<j} x_{ij} + \sum_{k<i} x_{ki} = 1, \quad \forall j = 1, \dots, m$$

$$x_{ij} = 0 \ or \ 1, \quad \forall i < j$$

1.4 Set Covering, Packing and Partitioning Problems (p107- Applied-Integer-programming)

Example
There are six cities (cities 1-6) in a county. The county must determine at which city/cities to build fire stations. The county wants to build the minimum number of fire stations needed to ensure that at least one fire station is within 15 minutes

(driving time) of each city. The times (in minutes) required to drive between the cities are shown in the table below.

	city1	city2	city3	city4	city5	city6
city1	0	10	20	30	30	20
city2	10	0	25	35	20	10
city3	20	25	0	15	30	20
city4	30	35	15	0	15	25
city5	30	20	30	15	0	14
city6	20	10	20	25	14	0

Formulate an IP that will tell how many fire stations should be built and where they should be located.

SOLUTION

Step 1: Definition of the Decision Variables
For each city j, we need to decide whether to build a fire station there:

Let $x_j = \begin{cases} 1, & if\ a\ fire\ station\ is\ build\ at\ city\ \ j \\ 0, & otherwise \end{cases}$, $j = 1,\ldots,6$

Step 2: Define the Constraints
To ensure that there is a fire station within 15 minutes of each city:

$x_1 + x_2 \geq 1,$ $(city\ 1\ constraint)$
$x_1 + x_2 + x_6 \geq 1,$ $(city\ 2\ constraint)$

$$x_3 + x_4 \geq 1, \quad (city\ 3\ constraint)$$
$$x_3 + x_4 + x_5 \geq 1, \quad (city\ 4\ constraint)$$
$$x_4 + x_5 + x_6 \geq 1, \quad (city\ 5\ constraint)$$
$$x_2 + x_5 + x_6 \geq 1, \quad (city\ 6\ constraint)$$

Step 3: Define the Objective Function
We want to build as less number of fire stations as possible:

$$\min \quad x_1 + x_2 + x_3 + x_4 + x_5 + x_6.$$

Set-covering Problem
In such problems, each member of a given set $N = \{1, \dots, n\}$ must be "covered" by a member/some members of $\{S_1, \dots, S_m\}$, where
$S_j \subseteq N, j = 1, \dots, \text{m}$.
If cost c_j is associated with set S_j , the objective is to find a minimum cost cover for the original set *N*, i.e., to have each $i \in N$ in at least one S_j .
In general a **Set-covering Model** is given by

$$\min \quad \sum_{j=1}^{m} c_j x_j$$
$$s.t. \quad \sum_{j=1}^{m} a_{ij} x_j \geq 1, \ \forall i \in N \quad (all - inclusive\ \ condition)$$
$$x \in \{0,1\}^m$$

Note that in above model, a_{ij} states for the 0-1 variable representing the TRUE/FALSE argument of whether element *i* belongs to set S_j .
For instance, the matrix $A = \{a_{ij}\}$ as defined in the fire station problem is:

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Set-packing Problem

Similarly (and conversely), if we are considering the packing of set N with the selection of set S_j to find a maximum-value collection of subsets which "packs" as many of the members of N as possible but allows no overlap:

$$\begin{aligned} \max \quad & \sum_{j=1}^{m} c_j x_j \\ s.t. \quad & \sum_{j=1}^{m} a_{ij}\, x_j \leq 1, \qquad (non\ overlapping\ condition) \\ & x \in \{0,1\}^m \end{aligned}$$

Set-partitioning Problem

Furthermore, if we consider covering and packing jointly, then the duty becomes to cover the set N fully and non-overlappingly, at a minimum cost:

$$\begin{aligned} \min \quad & \sum_{j=1}^{m} c_j x_j \\ s.t. \quad & \sum_{j=1}^{m} a_{ij}\, x_j = 1, \qquad (all - inclusive\ and\ non - overlapping) \\ & x \in \{0,1\}^m \end{aligned}$$

Exercise

A university is acquiring mathematical programming software for use in operational research classes. There are four available codes. Types of optimization problems they are able to solve are indicated by stars in the following table.

	code			
problems	1	2	3	4
LP	*	*	*	*
IP	-	*	-	*
NLP	-	-	*	*
objective	3	4	6	14

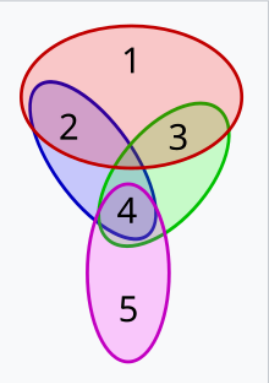
1. Taking objective function coefficients as code costs, formulate a set covering

model to acquire a minimum cost collection of codes that can solve all types of problem.

2. Taking objective function coefficients as code costs, formulate a set partitioning model to acquire a minimum cost collection of codes with exactly one code solves each type of problem.

3. Taking objective function coefficients as code quality, formulate a set packing model to acquire a maximum quality collection of codes with each type of problem can be solved by at most one code.

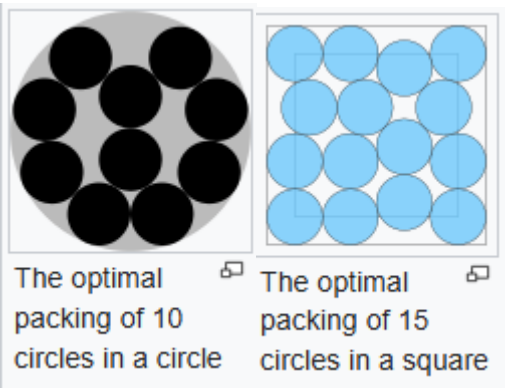
Examples Set covering: consider the universe, $U = \{1, 2, 3, 4, 5\}$ and the collection of sets $S = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}\}$. In this example, m is equal to 4, as there are four subsets that comprise this collection. The union of S is equal to U. However, we can cover all elements with only two sets: $\{\{1, 2, 3\}, \{4, 5\}\}$, see picture, but not with only one set. Therefore, the solution to the set cover problem for this U and S has size 2.



N = $U = \{1, 2, 3, 4, 5\}$, i=1,…,5

$S = \{S1=\{1, 2, 3\}, S2=\{2, 4\}, S3=\{3, 4\}, S4=\{4, 5\}\}$, j=1,…4

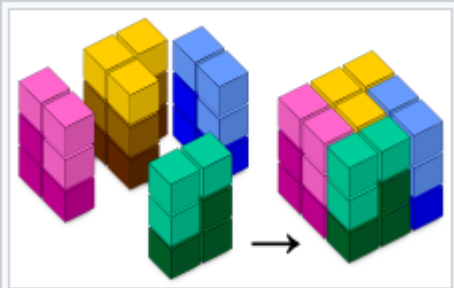
S1⊆ N, …..


The optimal packing of 10 circles in a circle

The optimal packing of 15 circles in a square

Example: Different cuboids into a cuboid

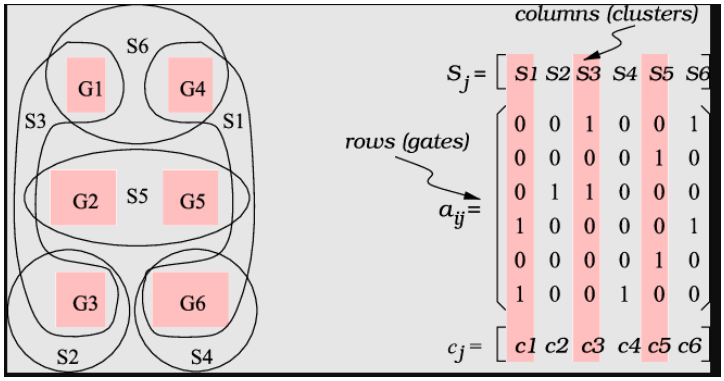
Determine the minimum number of cuboid containers (bins) that are required to pack a given set of item cuboids. The rectangular cuboids to be packed can be rotated by 90 degrees on each axis.



Packing nine L tricubes into a cube

Example: Given S = {3,1,1,2,2,1}, a valid solution to the partition problem is the two sets S_1 = {1,1,1,2} and S_2 = {2,3}. Both sets sum to 5, and they partition S. Note that this solution is not unique. S_1 = {3,1,1} and S_2 = {2,2,1} is another solution.

Example:



S1={G4, G6}, S3={G1, G3}, S5={G2, G5} the sets covers the elements and not overlapping.

https://youtu.be/Bo83F82Wd6U?si=HiDdN07AkMJCVdrN

https://youtu.be/PiryT9N7yeI?si=UfS6iJvIGqpv50J_