Unit 6. Integer Programming

Jordi Villà i Freixa

Universitat de Vic - Universitat Central de Catalunya
Study Abroad. Operations Research

jordi.villa@uvic.cat
https://mon.uvic.cat/cbbl

last updated: December 11, 2025

December 3rd-10th, 2025

FACULTAT DE CIÈNCIES, TECNOLOGIA I ENGINYERIES
UVIC | UVIC-UCC

Preliminary

This course is strongly based on the monography on Operations Research by Carter, Price and Rabadi [1], and in material obtained from different sources (quoted when needed through the slides).

Learning outcomes

- Getting familiar with the use of integer programming
- Solving integer programming problems

The concept: Summary

The concept

- Problems in which the feasible set is composed of only integer values.
- The feasible set is neither continuous nor convex.
- NP-hard problems in general.
- Integer problems that have a network structure are easy to solve using the Simplex method (assignment and matching problems, transportation and transshipment problems, and network flow problems always produce integer results, provided that the problem bounds are integers).
- Rounding can be effective in some problems and clearly not in others:
  - not the same tires than aircrafts!
  - values 0/1 for variable: zero-one or binary integer programming (produce or not produce cars in this factory)
  - mixed integer programming problems

Integer programming (knapsack problem)

Example: The **knapsack problem** is a classic integer programming problem where items with different weights and values must be selected to maximize total value while staying within a weight capacity constraint.

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i = 1, 2, \ldots, m \\ \text{subject to} \quad & x_j > 0 \quad j = 1, 2, \ldots, n \\ & x_j \quad \text{integer} \quad \text{for some or all} \quad j = 1, 2, \ldots, n \end{aligned}$$

In the 0-1 version of the knapsack problem, each item can either be included (1) or excluded (0) from the knapsack.

FACULTAT DE CIÈNCIES, TECNOLOGIA I ENGINYERIES
UVIC | UVIC-UCC

General integer programming problems I

Example: A manufacturer has 300 person-hours available this week and 1,800 units of raw material. These resources can be used to produce two products, A and B. The requirements and the profit for each item are given in the table below:

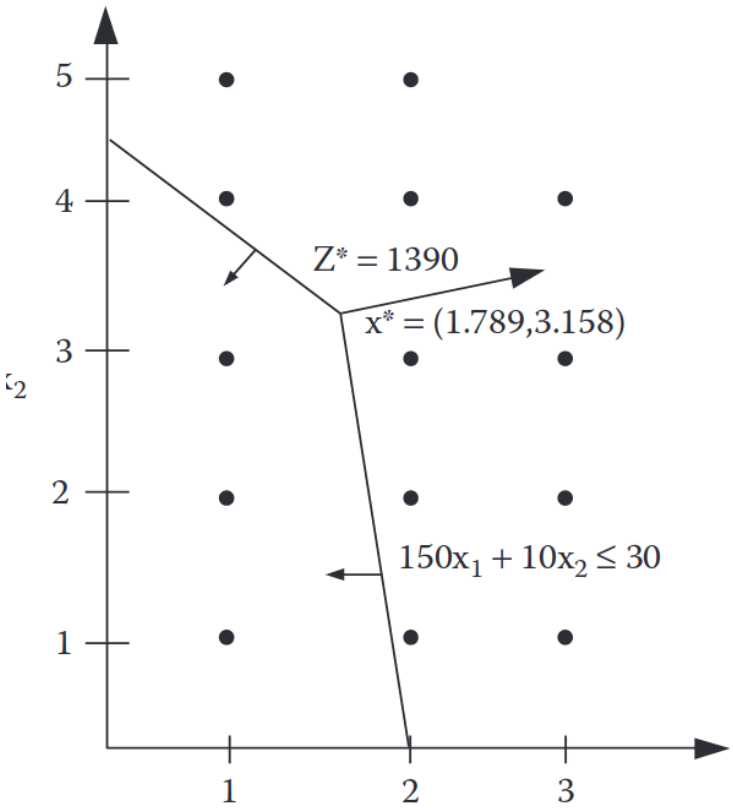
Product	**Person-Hours**	**Raw Material**	**Profit ($)**
A	150	300	600
B	10	400	100

General integer programming problems II

The manufacturer wants to determine how many units of each product to produce in order to maximize profit, given the constraints on person-hours and raw material. Since the products cannot be produced in fractional units, this is an integer programming problem. Let x_1 and x_2 represent the *integer number of units* of products A and B, respectively. We can formulate this problem as the following integer linear programming model:

$$\begin{aligned} \text{maximize} \quad & z = 600x_1 + 100x_2 \\ \text{subject to} \quad & \begin{cases} 150x_1 + 10x_2 \leq 300 \\ 300x_1 + 400x_2 \leq 1800 \\ x_1, x_2 \geq 0 \quad \text{and integer} \end{cases} \end{aligned}$$

General integer programming problems III



Graphical representation of a typical 2 dimensional integer programming[1].

IP examples: Summary

Integer problem examples

- Capital budgeting: deciding between a collection of investiments
- Warehouse location: in modelling distribution systems, we should decide about tradeoffs between transportation costs and costs for operating distributions centers
- Scheduling: students-faculty-classrooms allocations, vehicle dispatching, etc (see next slide)

Zero-One (0-1) scheduling problem

Airline crew scheduling problem: The airlines first design a flight schedule composed of a large number of *flight legs* (specific flight on a specific piece of equipment, such as a 747 from New York to Chicago departing at 6:27 a.m.). A flight crew is a complete set of people, including pilots, navigator, and flight attendants who are trained for a specific airplane. A work schedule or rotation is a collection of flight legs that are feasible for a flight crew, and that normally terminate at the point of origin. Variables x_{ij} have value 1 if flight leg i is assigned to crew j. All flight legs should be covered at minimum total cost.

Also called a *set-partitioning problem*

$$\text{minimize} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j = 1 \quad i = 1, 2, \ldots, m$$

$$x_j = 0, 1 \quad j = 1, 2, \ldots, n$$

Travelling salesman problem

Exercise 1

Consider the travelling salesman problem. Starting from his home, a salesman wishes to visit each of $(n-1)$ other cities and return home at minimal cost. He must visit each city exactly once and the cost to travel from city i to j is c_{ij}. Let x_{ij} be 1 or 0 depending on the fact that he goes or not from city i to city j

- formulate the optimization problem
- how to avoid disjoint tours?

Travelling Salesman Problem (TSP): Variables and Objective

Decision variables:

$$x_{ij} = \begin{cases} 1 & \text{if the salesman travels from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

Objective function (minimise total cost):

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

Basic constraints

Each city must be left exactly once and entered exactly once:

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

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

$$x_{ij} \in \{0, 1\} \quad \forall i, j$$

These constraints guarantee cycles, but still allow *subtours* (disconnected tours).

What are subtours?

Even if every city has one incoming and one outgoing arc, the model may produce several disconnected cycles:

Example of an infeasible solution with subtours:

- Tour 1: $1 \to 2 \to 3 \to 1$
- Tour 2: $4 \to 5 \to 6 \to 4$

We must restrict the model so that there is exactly one single tour.

Subtour elimination constraints (SEC)

For any subset of cities $S \subset \{1, \ldots, n\}$:

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1$$

These are known as **subtour elimination constraints (SEC)**.

They prevent a subset of cities from forming an independent cycle.

Drawback: The number of such sets S is exponential.

Alternative formulation: MTZ (Miller–Tucker–Zemlin)

See also this HowTo web page.
Introduce additional continuous variables u_i (visiting order):

$$u_i - u_j + n\,x_{ij} \leq n-1 \quad \forall i \neq j,\ \ i,j = 2,\ldots,n$$

with:

$$1 \leq u_i \leq n \quad \forall i = 2,\ldots,n$$

These constraints **eliminate subtours** by enforcing a consistent visiting order.

Advantage: polynomial number of constraints.

Complete model (summary)

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

subject to:

$$\sum_{j \neq i} x_{ij} = 1 \quad \forall i$$

$$\sum_{i \neq j} x_{ij} = 1 \quad \forall j$$

$$u_i - u_j + n x_{ij} \leq n - 1 \quad \forall i \neq j$$

$$x_{ij} \in \{0, 1\}, \quad u_i \geq 0$$

This formulation guarantees a single tour visiting all cities.

Applications of the Traveling Salesman Problem (TSP)

- **Logistics and Supply Chain:** Delivery route optimization for couriers (e.g., FedEx, UPS), and minimizing travel distances for food delivery services (e.g., Glovo).
- **Manufacturing:** Efficient routing of robotic arms in assembly lines, and optimal sequencing of drilling or cutting operations in circuit board manufacturing.
- **Telecommunications:** Designing optimal fiber optic or cable routing networks, and planning routes for signal maintenance teams.
- **Travel and Tourism:** Planning shortest tour routes for sightseeing in cities, and organizing efficient itineraries for travel agencies.
- **Biology and Genetics:** DNA sequencing, where TSP helps align fragments of genetic material.
- **Other Examples:** School bus routing, and urban planning and traffic management.

Solving integer problems: Summary

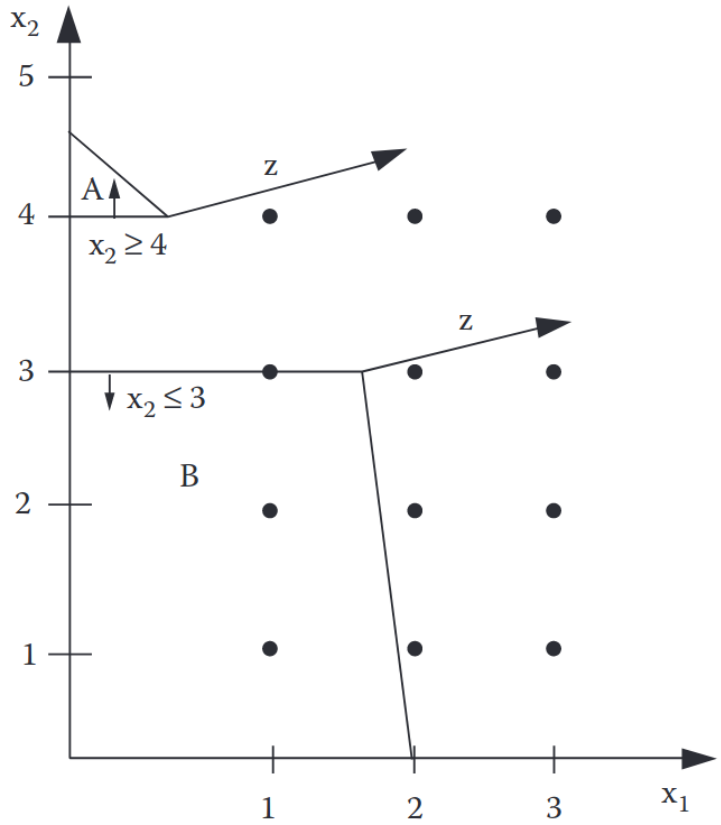
1. The concept
   - example
2. IP examples
   - Zero-One Scheduling Problem
   - Travelling Salesman Problem
3. Solving integer problems
   - Branch and Bound

Solving Integer problems

Simplex is not useful to solve, in general, integer problems. Instead, many other techniques have been proposed:

- Enumeration techniques, including the branch-and-bound procedure;
- cutting plane techniques; and
- group-theoretic techniques,

as well as several composite techniques



Separation into two subproblems in the Branch-and-Bound method[1].

Exercise 2

Using the graphical representation and the branch-and-bound procedure, solve this integer program:

$$
\begin{aligned}
\text{maximize} \quad & z = x_1 + 5x_2 \\
& -4x_1 + 3x_2 \leq 6 \\
\text{subject to} \quad & 3x_1 + 2x_2 \leq 18 \\
& x_1, x_2 \geq 0 \quad \text{and integer}
\end{aligned}
$$

LP Relaxation (Graphical Solution)

Relax the integrality constraints: $x_1, x_2 \geq 0$ continuous.

$$-4x_1 + 3x_2 \leq 6 \quad \Rightarrow \quad x_2 \leq 2 + \frac{4}{3}x_1$$

$$3x_1 + 2x_2 \leq 18 \quad \Rightarrow \quad x_2 \leq 9 - \frac{3}{2}x_1$$

Intersection of the two lines:

$$\begin{cases} -4x_1 + 3x_2 = 6 \\ 3x_1 + 2x_2 = 18 \end{cases} \Rightarrow x_1 = \frac{42}{17},\ x_2 = \frac{90}{17}$$

Evaluate the objective function:

$$z = x_1 + 5x_2 = \frac{492}{17} \approx 28.94$$

$$\Rightarrow \textbf{LP optimum: } \left(\frac{42}{17}, \frac{90}{17}\right)$$

This solution is **not integer** $\Rightarrow$ Branch and Bound is needed.

Branch and Bound – First Branching

We branch on the fractional variable:

$$x_2 = \frac{90}{17} \approx 5.29$$

Two subproblems:

$$\text{(A) } x_2 \leq 5 \qquad \text{(B) } x_2 \geq 6$$

Branch B: If $x_2 = 6$ then

$$3x_1 + 12 \leq 18 \Rightarrow x_1 \leq 2$$

But also

$$-4x_1 + 18 \leq 6 \Rightarrow x_1 \geq 3$$

Contradiction $\Rightarrow$ **Branch B is infeasible**. Proceed with **Branch A:** $x_2 \leq 5$

Branch A: LP relaxation

Add constraint $x_2 \leq 5$.

$$3x_1 + 2(5) = 18 \Rightarrow x_1 = \frac{8}{3}$$

LP solution:

$$(x_1, x_2) = \left(\frac{8}{3}, 5\right)$$

$$z = \frac{8}{3} + 25 = \frac{83}{3} \approx 27.67$$

Still fractional $\rightarrow$ branch on x_1:

$$\text{(A1) } x_1 \leq 2 \qquad \text{(A2) } x_1 \geq 3$$

Node A1: $x_1 \leq 2$

Try point $(x_1, x_2) = (2, 4)$ (largest integer feasible): Check constraints:

$$-4(2) + 3(4) = 4 \leq 6 \quad \checkmark$$

$$3(2) + 2(4) = 14 \leq 18 \quad \checkmark$$

$$z = 2 + 5(4) = 22$$

$\Rightarrow$ Feasible integer solution found:

$$z = 22 \Rightarrow \textbf{Current best (incumbent)}$$

Node A2: $x_1 \geq 3$

For $x_1 = 3$:

$$3(3) + 2x_2 = 18 \Rightarrow x_2 = 4.5$$

LP solution:

$$(x_1, x_2) = (3,\ 4.5) \quad \Rightarrow \quad z = 3 + 5(4.5) = 25.5$$

This is fractional $\rightarrow$ branch on x_2:

$$\text{(A2a) } x_2 \leq 4 \qquad \text{(A2b) } x_2 \geq 5$$

Node A2b: $x_2 \geq 5$

Earlier we saw:

$$x_2 \geq 5 \Rightarrow x_1 \leq \frac{8}{3}$$

But here we have:

$$x_1 \geq 3$$

Contradiction $\Rightarrow$ **Node A2b is infeasible (pruned)**

Node A2a: $x_2 \leq 4$

Try the best integer point $(x_1, x_2) = (3, 4)$:

$$-4(3) + 3(4) = 0 \leq 6 \quad \checkmark$$

$$3(3) + 2(4) = 17 \leq 18 \quad \checkmark$$

$$z = 3 + 5(4) = \boxed{23}$$

Update:

$$\text{Best solution } = (3, 4), \quad z = 23$$

All other branches are either infeasible or have lower bounds STOP.

Final Answer

$$\boxed{x_1 = 3, \quad x_2 = 4}$$

$$\boxed{z_{\max} = 23}$$

Summary:

- LP relaxation value: $z = 28.94$ (not integer)
- Branch-and-bound explored feasible regions
- Best integer point: $(3, 4)$

$$\boxed{\text{Optimal integer solution: } z = 23}$$

References I

[1] Michael Carter, Camille C Price, and Ghaith Rabadi. *Operations Research. A Practical Introduction. Second Edition*. CRC Press, 2019. ISBN: 978-1-4987-8010-0.