Introduction

1. **Assignment problem**

Given m jobs and m machines, suppose that each job can be executed by any machine and that t_{ij} is the execution time of job J_i on machine M_j.

	M_1	M_2	M_3
J_1	2	6	3
J_2	8	4	9
J_3	5	7	8

Decide which job to assign to each machine so as to minimize the total execution time.

Each job must be assigned to exactly one machine, and each machine to exactly one job.

Number of feasible solutions? $m!$ possible assignments

1.2 Scheme of an O.R. study

$$\boxed{\text{Problem}} \rightarrow \boxed{\text{Model}} \rightarrow \boxed{\text{Algorithm}} \rightarrow \boxed{\text{Implementation}}$$

Main steps:

- define the problem
- build the model
- select or develop an appropriate algorithm
- implement or use and efficient computer program
- analyze the results

with feedbacks (the previous steps are reconsidered whenever useful).

A mathematical model is a simplified representation of a real-world problem.

To define it we must identify the fundamental elements of the problem and the main relationships among them.

Example 1. Production planning

A company produces 3 types of electronic devices: D_1, D_2, D_3

Main phases of the production process: assembly, refinement and quality control.

Time (in minutes) required for each phase and product:

	D_1	D_2	D_3
Assembly	80	70	120
Refinement	70	90	20
Quality control	40	30	20

Available resources whithin the planning horizion (depend on the workforce) in minutes:

Assembly	Refinement	Quality control
30 000	25 000	18 000

Unitary profit (in KEuro):

D_1	D_2	D_3
1.6	1	2

Assumption: the company can sell whathever it produces.

Give a **mathematical model** for determining a production plan which maximizes the total profit.

Decision variables:

$$x_j = \text{ number of devices } D_j \text{ produced, for } j = 1, 2, 3$$

Objective function:

$$\max\ z = 1.6x_1 + 1x_2 + 2x_3$$

Constraints: Production capacity limit for each phase

$$\begin{aligned} 80x_1 + 70x_2 + 120x_3 &\leq 30\,000 && \text{(assembly)} \\ 70x_1 + 90x_2 + 20x_3 &\leq 25\,000 && \text{(refinement)} \\ 40x_1 + 30x_2 + 20x_3 &\leq 18\,000 && \text{(quality control)} \end{aligned}$$

Non-negative variables:

$$x_1, x_2, x_3 \geq 0 \qquad \text{may be fractional (real) values}$$

Example 2. Portfolio selection problem

An insurance company must decide which investments to select out of a given set of possible assets (stocks, bonds, options, gold certificates, real estate, ...).

Investments	area	capital (c_j K€)	expected return (r_j)
A (automotive)	Germany	150	11%
B (automotive)	Italy	150	9%
C (ICT)	U.S.A.	60	13%
D (ICT)	Italy	100	10%
E (real estate)	Italy	125	8%
F (real estate)	France	100	7%
G (short term treasury bonds)	Italy	50	3%
H (long term treasury bonds)	UK	80	5%

Available capital = 600 KEuro

At most 5 investments to avoid excessive fragmentation.

Geographic diversification to limit risk: ≤ 3 investments in Italy and ≤ 3 abroad.

Give a **mathematical model** for deciding which investments to select so as to maximize the expected return while satisfying the constraints.

Decision variables:

$x_j = 1$ if j-th investment is selected and $x_j = 0$ otherwise, for $j = 1, \ldots, 8$

Objective function:

$$\max\ z = \sum_{j=1}^{8} c_j\ r_j\ x_j$$

Constraints:

$$\begin{aligned}
\sum_{j=1}^{8} c_j\ x_j &\leq 600 && \text{(capital)} \\
\sum_{j=1}^{8} x_j &\leq 5 && \text{(max 5 investments)} \\
x_2 + x_4 + x_5 + x_7 &\leq 3 && \text{(max 3 in Italy)} \\
x_1 + x_3 + x_6 + x_8 &\leq 3 && \text{(max 3 abroad)}
\end{aligned}$$

Binary (integer) variables:

$$x_j \in \{0, 1\} \qquad 1 \leq j \leq 8$$

Variant: In order to limit the risk, if any of the ICT investment is selected then at least one of the treasury bond must be selected.

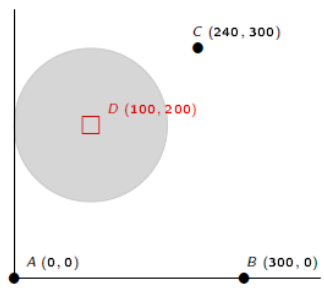
Model:

$$\max\ z = \sum_{j=1}^{8} c_j\ r_j\ x_j$$

$$\begin{array}{rcll}
\text{subject to} \displaystyle\sum_{j=1}^{8} c_j\ x_j & \leq & 600 & \text{(capital)} \\
\displaystyle\sum_{j=1}^{8} x_j & \leq & 5 & \text{(max 5 investments)} \\
x_2 + x_4 + x_5 + x_7 & \leq & 3 & \text{(max 3 in Italy)} \\
x_1 + x_3 + x_6 + x_8 & \leq & 3 & \text{(max 3 abroad)} \\
\dfrac{x_3 + x_4}{2} & \leq & x_7 + x_8 & \text{(investment in treasury bonds)} \\
x_j & \in & \{0,1\} & 1 \leq j \leq 8
\end{array}$$

Example 3. Facility location

Consider 3 oil pits, located in A, B and C, from which oil is extracted.



Connect them to a refinery with pipelines whose cost is proportional to the square of their length.

The refinery must be at least 100 km away from point D but the oil pipelines can cross the corresponding forbidden zone.

Give a **mathematical model** to decide where to locate the refinery so as to minimize the total pipeline cost.

Decision variables:

$$x_1, x_2 \text{ cartesian coordinates of the refinery}$$

Objective function:

$$\min \quad z = \left[(x_1 - 0)^2 + (x_2 - 0)^2\right] + \left[(x_1 - 300)^2 + (x_2 - 0)^2\right] + \left[(x_1 - 240)^2 + (x_2 - 300)^2\right]$$

Constraints:

$$\sqrt{(x_1 - 100)^2 + (x_2 - 200)^2} \geq 100$$

Variables:

$$x_1, x_2 \in \mathbb{R}$$

1.5 Decision-making problems

- Number of decision makers (who decides?)
- Number of objectives (based on which criteria?)
- Level of uncertainty in the parameters (based on which information?)

You may think of each one of these three features as a dimension of an abstract coordinate system...

One decision maker, one objective $\Rightarrow$ Mathematical programming

One decision maker, several objectives $\Rightarrow$ Multi-objective programming

Uncertainty level $> 0 \Rightarrow$ Stochastic programming

Several decision makers $\Rightarrow$ Game theory

1.6 Mathematical Programming/Optimization (MP)

Decision-making problems with a **single decision maker**, a **single objective** and reliable parameter estimates:

$$\min \ f(\mathrm{x}) \quad \text{with} \ \ \mathrm{x} \in X \tag{1}$$

- **decision variables** $\mathrm{x} \in \mathbb{R}^n$: numerical variables whose values identify a solution of the problem
- **feasible region** $X \subseteq \mathbb{R}^n$ distinguishes between feasible and infeasible solutions (via constraints)

$$X = \left\{ \mathrm{x} \in \mathbb{R}^n : g_i(\mathrm{x}) \left\{ \begin{array}{c} = \\ \leq \\ \geq \end{array} \right\} 0, i = 1, \ldots, m \right\}$$

- **objective function** $f : X \to \mathbb{R}$ expresses in quantitative terms the value or cost of each feasible solution.

Observation: $\max\{f(\mathrm{x}) : \ \mathrm{x} \in X\} = -\min\{-f(\mathrm{x}) : \ \mathrm{x} \in X\}$

Global optima

Solving a mathematical programming (MP) problem consists in finding a **global minimum**, i.e., a vector $\mathrm{x}^* \in X$ such that

$$f(\mathrm{x}^*) \leq f(\mathrm{x}) \qquad \forall \mathrm{x} \in X$$

It may happen that the problem

1. is infeasible ($X = \emptyset$),
2. is unbounded ($\forall c \in \mathbb{R}, \exists \mathrm{x}_c \in X$ such that $f(\mathrm{x}_c) \leq c$ or $f(\mathrm{x}_c) \geq c$),
3. has a single optimal solution,
4. has a large number (even infinite) of optimal solutions.

Local optima

When the problem is very hard we must settle for a feasible solution that is a **local minimum**, i.e., a vector $\hat{\mathrm{x}} \in X$ such that

$$f(\hat{\mathrm{x}}) \leq f(\mathrm{x}) \qquad \forall \mathrm{x} \in X \text{ with } ||\mathrm{x} - \hat{\mathrm{x}}|| \leq \epsilon$$

for an appropriate value $\epsilon > 0$.

An optimization problem can have many local optima.

Special cases of Mathematical Programming

- **Linear Programming** (LP)

$$f \text{ linear and } X = \left\{ \mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \left\{ \begin{array}{c} = \\ \leq \\ \geq \end{array} \right\} 0, i = 1, \ldots, m \right\} \text{ with } g_i \text{ linear } \forall i$$

Example: Production planning

- **Integer Linear Programming** (ILP)

$$f \text{ linear and } X = \left\{ \mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \left\{ \begin{array}{c} = \\ \leq \\ \geq \end{array} \right\} 0, i = 1, \ldots, m \right\} \cap \mathbb{Z}^n \text{ with } g_i \text{ linear } \forall i$$

Example: Portfolio selection

- **Nonlinear Programming** (NLP)

$$f \text{ (non) convex/regular and } X = \left\{ \mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \left\{ \begin{array}{c} = \\ \leq \\ \geq \end{array} \right\} 0, i = 1, \ldots, m \right\} \text{ with } g_i$$

(non) convex/regular $\forall i$

Example: Facility location (with g_i convex)

Complexity of algorithms

- Since it is often hard to determine the exact number of elementary operations (as a function of the instance size), we consider the asymptotic number of elementary operations (speed of growth) in the worst case.
- We look for a function $f(n)$ such that, for n sufficiently large,

  the number of elementary operations needed to solve any instance of size at most $n \leq f(n)$.

Chapter 1

Complexity of the reachability algorithm

At each iteration of the `while` loop:

- Select one node $u \in Q$, extract it from Q and insert it in M.
- For all nodes v directly reachable from u and not already in M or Q, insert v in Q.

Since each node u is inserted in Q at most once and each arc (u, v) is considered at most once, the overall complexity is

$$O(n + m), \text{ where } n = |N|, m = |A|.$$

Note: For dense graphs, $m = O(n^2)$.

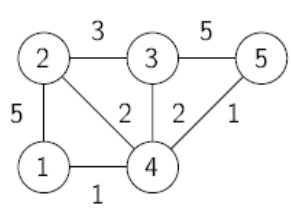
2.2.1 Prim's algorithm

Idea: Iteratively build a spanning tree.

Method

- Start from initial tree (S, T) with $S = \{u\}$ and $T = \emptyset$ (u can be any node in N).
- At each step, add to the current partial tree (S, T) an edge of minimum cost among those which connect a node in S to a node in $N \backslash S$.

Example



Prim's algorithm

- Input: Connected $G = (N, E)$ with edge costs
- Output: Subset $T \subseteq N$ of edges of G such that $G_T = (N, T)$ is a minimum cost spanning tree of G.

Algorithm 1: Prim's algorithm for the minimum cost spanning tree problem

```
S ← {u}
T ← ∅
while |T| < n − 1 do
    {u, v} ← edge in δ(S) of minimum cost /* u ∈ S and v ∈ N\S */
    S ← S ∪ {v}
    T ← T ∪ {{u, v}}
```

If all edges are scanned at each iteration, the complexity order is: $O(nm)$.

Exactness of Prim's algorithm

Definition: An algorithm is exact if it provides an optimal solution for every instance, otherwise it is heuristic.

Proposition 1

Prim's algorithm is exact.

The exactness does not depend on the choice of the first node nor on the selected edge of minimum cost in $\delta(S)$.

We show that each selected edge belongs to a minimum cost spanning tree.

Definition: A greedy algorithm constructs a feasible solution iteratively by making at each step a "locally optimal" choice, without reconsidering previous choices.

Note: Prim's algorithm is greedy.

At each step a minimum cost edge is selected among those in the cut $\delta(S)$ induced by the current set of nodes S.

Note: For most discrete optimization problems greedy-type algorithms yield a feasible solution with no guarantee of optimality.

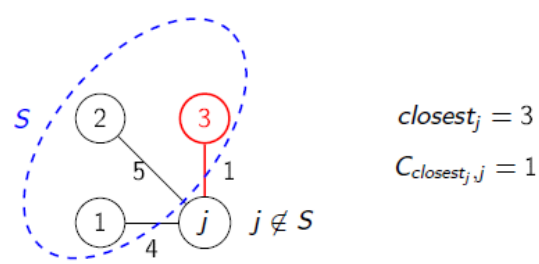
Various greedy algorithms for the minimum cost spanning tree problem:

- Boruvka (1926);
- Kruskal (1956) – Exercise 2.2;
- Prim (1957), ...

Implementation in $O(n^2)$

Data structure

- k: number of edges selected so far
- Subset $S \subseteq N$ of nodes incident to the selected edges
- Subset $T \subseteq E$ of selected edges
- $C_j = \begin{cases} \min\{c_{ij} : \ i \in S\}, & j \notin S \\ +\infty, & \text{otherwise} \end{cases}$
- $closest_j = \begin{cases} \operatorname{argmin}\{c_{ij} : \ i \in S\}, & j \notin S \\ \text{"predecessor" of } j \text{ in the minimum spanning tree}, & j \in S \end{cases}$



$O(n^2)$ **version of Prim's algorithm**

Algorithm 2: Prim's algorithm

```
T ← ∅; S ← {u}                    /* Initialization                          */
for j ∈ N\S do
    C_j ← c_uj                    /* If {u,j} ∉ E then c_uj = +∞             */
    closest_j ← u
for k = 1, ..., n − 1 do
    min ← +∞                      /* Select min edge in δ(S)                 */
    for j = 1, ..., n do
        if j ∉ S and C_j < min then
            min ← C_j; v ← j
    S ← S ∪ {v}; T ← T ∪ {{closest_v, v}}      /* Extend S and T             */
    for j = 1, ..., n do
        if j ∉ S and c_vj < C_j then
            C_j ← c_vj; closest_j ← v   /* Update C_j and closest_j, ∀j ∉ S  */
```

Complexity order of Prim's algorithm

```
for j = 2, ..., n do
    O(1)
for k = 1, ..., n − 1 do
    for j = 2, ..., n do
        O(1)
    for j = 2, ..., n do
        O(1)
```

Overall complexity: $O(n^2)$

For sparse graphs, where $m \ll n(n-1)/2$, a more sophisticated data structure leads to an $O(m \log n)$ complexity.

2.3.2 Dijkstra's algorithm

Definition: A path $\langle (i_1, i_2), (i_2, i_3), \ldots, (i_{k-1}, i_k) \rangle$ is simple if no node is visited more than once.

Property 1

If $c_{ij} \geq 0$ for all $(i,j) \in A$, there is at least one shortest path which is simple.

Assumption: $c_{ij} \geq 0$, for evey $(i,j) \in A$.

Input: $G = (N, A)$ with non-negative arc costs, $s \in N$.

Output: Shortest paths from s to all other nodes of G.

Edsger Dijkstra
(1930-2002)

Data structure

- $S \subseteq N$: subset of nodes whose labels are permanent
- $\overline{S} = (N \setminus S) \subseteq N$: subset of nodes with temporary labels
- $L_j = \begin{cases} \text{cost of a shortest path from } s \text{ to } j, & j \in S \\ \min\{L_i + c_{ij} : (i,j) \in \delta^+(S)\}, & j \notin S \end{cases}$
  (upper bound on the length of a shortest path)

  Given directed G and the current $S \subset N$, consider $\delta^+(S)$ and select $(u,v) \in \delta^+(S)$ such that

  $$L_u + c_{uv} = \min\{L_i + c_{ij} : (i,j) \in \delta^+(S)\}.$$

  Thus, $L_u + c_{uv} \leq L_i + c_{ij}, \quad \forall (i,j) \in \delta^+(S)$.
- $pred_j = \begin{cases} \text{"predecessor" of } j \text{ in the shortest path from } s \text{ to } j, & j \in S \\ u \text{ such that } L_u + c_{uj} = \min\{L_i + c_{ij} : i \in S\}, & j \notin S \end{cases}$

Dijkstra's algorithm

- Input: Graph $G = (N, E)$ with non-negative arc costs, $s \in N$.
- Output: Shortest paths from s to all other nodes of G.

Algorithm 1: Dijkstra's algorithm for the shortest path problem

$S \leftarrow \emptyset$;
for $u \in N$ **do** $L_u \leftarrow +\infty$;
$L_s \leftarrow 0$;
while $|S| \neq n$ **do**
  $v \leftarrow \operatorname{argmin}\{L_i : i \in \overline{S}\}$;
  $S \leftarrow S \cup \{v\}$;
  for $(v, w) \in \delta^+(v)$ *such that* $L_w > L_v + c_{vw}$ **do**
    $L_w \leftarrow L_v + c_{vw}$;
    $pred_w \leftarrow v$;

Complexity order

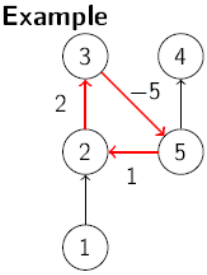
Depends on how the arc (u, v) is selected among those of the current $\delta^+(S)$.

- If all m arcs are scanned, the overall complexity would be $O(nm)$.
- If all labels L_j are determined by appropriate updates the overall complexity is $O(n^2)$.

2.3.3 Shortest path problem with negative costs

Observation: If a graph G contains a circuit of negative cost, the shortest path problem may not be well-defined.

Example



No finite shortest path from s to t.

Circuit's cost: -2

Floyd-Warshall's algorithm:

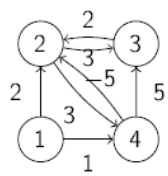
- detects the presence of circuits with negative cost (identifies ill-definedness),
- provides a set of shortest paths between all pairs of nodes, even when there are arcs with negative cost.

Floyd-Warshall's algorithm

- Input: Directed $G = (N, A)$ with an $n \times n$ cost matrix, $C = [c_{ij}]$.
- Output: For each pair of nodes $i, j \in N$, the cost d_{ij} of the shortest path from i to j.
- Data structure: Two $n \times n$ matrices, D, P, whose elements correspond, at the end of the algorithm, to:
  - d_{ij} = cost of a shortest path from i to j;
  - p_{ij} = predecessor of j in that shortest path from i to j.

- **Initialization**

$$d_{ij} = \begin{cases} 0 & i = j \\ c_{ij} & i \neq j \wedge (i,j) \in A, \\ +\infty & \text{otherwise} \end{cases} \qquad p_{ij} = i, i \in N$$

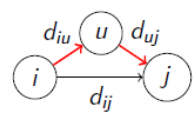


$$D = \begin{bmatrix} 0 & 2 & \infty & 1 \\ \infty & 0 & 3 & 3 \\ \infty & 2 & 0 & \infty \\ \infty & -5 & 5 & 0 \end{bmatrix} \quad P = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

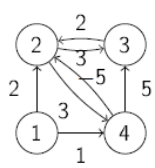
- **Triangular operation**

  For each pair of nodes i, j with $i \neq u$ and $j \neq u$ (including case $i = j$), check whether when going from i to j it is more convenient to go via u:

  if $d_{iu} + d_{uj} < d_{ij}$ then $d_{ij} \leftarrow d_{iu} + d_{uj}$.



Iteration $u = 1$: Since no such pair exists, the matrices D, P remain unchanged. However, nodes $i = 1$ and $j = 1$ are skipped.



$$D = \begin{bmatrix} 0 & 2 & \infty & 1 \\ \infty & 0 & 3 & 3 \\ \infty & 2 & 0 & \infty \\ \infty & -5 & 5 & 0 \end{bmatrix} \quad P = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

- $0 = d_{22} < d_{21} + d_{12} = \infty + 2$ (no change)
- $3 = d_{23} < d_{21} + d_{13} = \infty + \infty$ (no change)
- $3 = d_{24} < d_{21} + d_{14} = \infty + 1$ (no change)
- $2 = d_{32} < d_{31} + d_{12} = \infty + 2$ (no change)
- $0 = d_{33} < d_{31} + d_{13} = \infty + \infty$ (no change)
- $\infty = d_{34} < d_{31} + d_{14} = \infty + 1$ (no change)
- $-5 = d_{42} < d_{41} + d_{12} = \infty + 2$ (no change)
- $5 = d_{43} < d_{41} + d_{13} = \infty + \infty$ (no change)
- $0 = d_{44} < d_{41} + d_{14} = \infty + 1$ (no change)

Algorithm 2: Floyd-Warshall's algorithm for the all-pairs shortest path problem

```
for i ∈ N do
    for j ∈ N do
        p_ij ← i; d_ij = { 0      i = j
                           c_ij   i ≠ j ∧ (i,j) ∈ A
                           +∞     otherwise
for u ∈ N        /* Triangular operation                          */
 do
    for i ∈ N\{u} do
        for j ∈ N\{u} do
            if d_iu + d_uj < d_ij   /* Update d_ij if cheaper to go via u  */
             then
                d_ij ← d_iu + d_uj; p_ij ← p_uj;
    for i ∈ N do
        if d_ii < 0   /* There exists a circuit with negative cost  */
         then  Stop
```

Proposition 2

Floyd-Warshall's algorithm is exact.

Proof's idea:

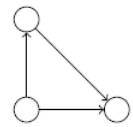
Assume that the nodes of G are numbered from 1 to n.
Verify that, if the node index order is followed, after the u-th cycle the value d_{ij} (for any i and j) corresponds to the cost of a shortest path from i to j with only intermediate nodes in $\{1, \ldots, u\}$.

Complexity

$O(n^3)$ where $n = |N|$

2.3.4 Optimal paths in directed acyclic graphs

Definition: A directed graph $G = (N, A)$ is acyclic (is a DAG) if it contains no circuits.



Problem

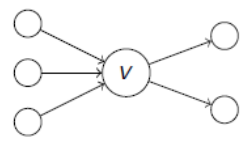
Given a DAG $G = (N, A)$ with a cost $c_{ij} \in \mathbb{R}$ for each $(i, j) \in A$, and nodes s and t, determine a shortest (longest) path from s to t.

Property (Topological order): The nodes of any DAG G can be ordered topologically, that is, indexed so that for each arc $(i, j) \in A$ we have $i < j$.

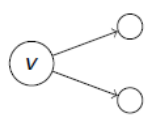
The topological order can be exploited in a very efficient Dynamic Programming algorithm to find shortest (or longest) paths.

Topological ordering method

$G = (N, A)$ is represented via the lists of predecessors $\delta^-(v)$ and successors $\delta^+(v)$ for each node v.



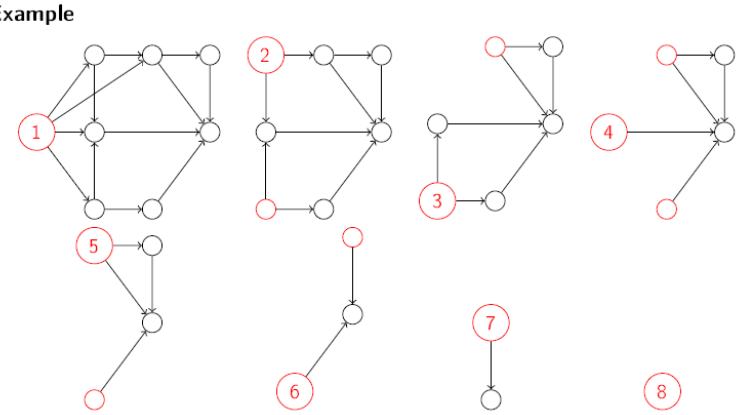
1. Assign the smallest positive integer not yet assigned to a node $v \in N$ with $\delta^-(v) = \emptyset$.



Such a node v exists because G does not contain circuits.

2. Delete the node v with all its incident arcs.
3. Go back to 1. until there are nodes in the current subgraph.

Example



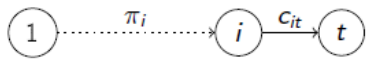
Complexity

$O(n + m)$ where $n = |N|$ and $m = |A|$

because each node/arc is considered at most once.

Dynamic Programming for shortest paths in DAGs

Any shortest path π_t from 1 to t with at least 2 arcs, can be subdivided into two parts: π_i and (i,t), where π_i is a shortest subpath from s to i.



- For each node $i = 1, \ldots, t$, let L_i be the cost of a shortest path from 1 to i. Then
$$L_t = \min_{(i,t)\in\delta^-(t)} \{L_i + c_{it}\},$$
where the minimum is taken over all possible predecessors i of t.
- If G is a DAG and nodes are topologically ordered, the only possible predecessors of t in a shortest path π_t from 1 to t are those with index $i < t$. Thus
$$L_t = \min_{i<t}\{L_i + c_{it}\}.$$
- In a G with circuits, any node different from t can be a predecessor of t in π_t!

Algorithm 3: Dynamic Programming to find shortest paths in DAGs

Sort the nodes of G topologically
$L_1 \leftarrow 0$
for $j = 2, \ldots, n$ /* in topological order */
do
  $L_j \leftarrow \min\{L_i + c_{ij} : (i,j) \in \delta^-(j) \wedge (i < j)\}$
  $pred_j \leftarrow v$ such that $(v,j) = \text{argmin}\{L_i + c_{ij} : (i,j) \in \delta^-(j) \wedge (i < j)\}$

Topological ordering of the nodes: $O(n+m)$ with $m = |A|$

Each node/arc is considered exactly once: $O(n+m)$

Overall: $O(n+m)$

Adaptation of DP for longest paths in DAGs

Straightforward adaptation of the DP method to find longest paths in DAGs:

$$L_t = \max_{i<t}\{L_i + c_{it}\}, \ldots$$

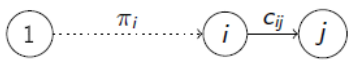
Proposition 3

The Dynamic Programming algorithm for finding shortest (longest) paths in DAGs is exact.

This is due to:

Optimality principle

For any shortest (longest) path π_j from 1 to j there exists $i < j$ such that π_j can be subdivided into two parts: π_i and (i, j), where subpath π_i is of minimum (maximum) length from 1 to i.



Dynamic Programming (DP)

- Proposed in 1953 by Richard Bellman (1920-1984).
- General technique in which an optimal solution, composed of a sequence of elementary decisions, is determined by solving a set of recursive equations.
- DP is applicable to any "sequential decision problem" for which the optimality property is satisfied.
- Wide range of applications: optimal control (rocket launch), equipment maintenance and replacement, selection of inspection points along a production line, ...

Exercise

One machine, 5 year time horizon. Can buy/sell machine at beginning of each year. Cost of a new machine is constant (12 K€).

Goal

Minimize the net total cost (price + maintenance - recover) over 5 year horizon.

Age	Main.	Reco.
0	2	–
1	4	7
2	5	6
3	9	2
4	12	1

If we buy a machine at beginning of 1st year and sell it at the beginning of 2nd year, the net cost is: $12 + 2 - 7 = 7$

If we buy it at the beginning of 1st year and sell it at the beginning of 3rd year, the net cost is: $12 + 2 + 4 - 6 = 12$, etc, ...

Show how the problem of determining an optimal maintenance – replacement policy of minimum total cost (for the next 5 years) can be solved via DP.

Hint

Reduce it to a minimum cost path problem in an appropriate acyclic graph.
Find all the optimal maintenance-replacement policies.

Critical path method (CPM)

Determines:

- a schedule (specifying the order and the alotted time for the activities) that minimizes the overall project duration;
- the slack of each activity, i.e., the amount of time by which its execution can be delayed, without affecting the overall minimum project duration.

Initialization: Construct the graph G representing the project and find a topological order of the nodes.

Step I: Consider the nodes by increasing indices and, for each $h \in N$, find: the earliest time $Tmin_h$ at which the event associated to node h can occur ($Tmin_n$ corresponds to the minimum project duration).

Step II: Consider the nodes by decreasing indices and, for each $h \in N$, find: the latest time $Tmax_h$ at which the event associated to node h can occur without delaying the project completion date beyond $Tmin_n$.

Step III: For each activity $(i,j) \in A$, find the slack: $\sigma_{ij} = Tmax_j - Tmin_i - d_{ij}$.

Critical path method

- Input: Graph $G = (N, A)$, with $n = |N|$, $m = |A|$ and the duration d_{ij} associated to each $(i,j) \in A$.
- Output: $(Tmin_i, Tmax_i)$, for $i = 1, \ldots, n$.

Algorithm 1: CPM

Sort the nodes topologically
$Tmin_1 \leftarrow 0$
for $j = 2, \ldots, n$ **do**
  $Tmin_j \leftarrow \max\{Tmin_i + d_{ij} : (i,j) \in \delta^-(j)\}$
$Tmax_n \leftarrow Tmin_n$
/* minimum project duration */
for $i = n-1, \ldots, 1$ **do**
  $Tmax_i \leftarrow \min\{Tmax_j - d_{ij} : (i,j) \in \delta^+(i)\}$

Complexity order

$O(n+m)$, or simply $O(m)$.

- Input: Graph $G = (N, A)$ with capacity $k_{ij} > 0$,for every $(i, j) \in A$, $s, t \in N$.
- Output: Feasible flow $\underline{x}$ from s to t of maximum value φ^*

Algorithm 1: Ford-Fulkerson's algorithm for the maximum flow problem

```
x ← 0
φ ← 0
optimum ← false
repeat
    Build residual network Ḡ associated to x
    P ← path from s to t in Ḡ
    if P is not defined then optimum ← true
    else
        δ ← min{k̄_ij : (i,j) ∈ P}
        φ ← φ + δ
        for (i,j) ∈ P do
            if (i,j) is a forward arc then x_ij ← x_ij + δ
            else x_ij ← x_ij − δ
until optimum = true
```

Complexity

- Since $\delta > 0$, the value φ increases at each iteration (cycle).
- If all k_{ij} are integer, $\underline{x}$ and $\bar{k}_{ij}$ integer and $\delta \geq 1$, then there are at most φ^* increases.
- Since
$$\varphi^* \leq k(\{s\}) \leq m\, k_{\max},$$
where $m = |A|$ and $k_{\max} = \max\{k_{ij} : (i, j) \in A\}$ and each cycle is $O(m)$, the overall complexity is $O(m^2 k_{\max})$.

Definition 7

The size of an instance I, denoted by $|I|$, is the number of bits needed to describe the instance.

Since $\lceil \log_2 i \rceil + 1$ bits are needed to store integer i, $|I| = O(m \log_2 k_{\max})$.

$O(m^2 k_{\max})$ grows exponentially with $|I|$ because $k_{\max} = 2^{\log_2 k_{\max}}$.

2.4.4 Indirect applications

1) Assigment (matching) problem

Given m engineers, n tasks and for each engineer the list of tasks he/she can perform. Assign the tasks to the engineers such that:

- each engineer is assigned at most one task,
- each task is assigned to at most one engineer,

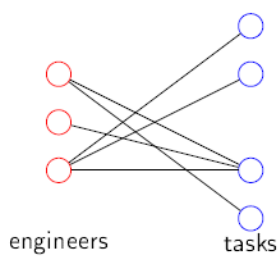
and the number of tasks that are executed (engineers involved) is maximized.

If the competences of the engineers are represented via a bipartite graph, what are we looking for in such a graph?

How can we reduce this problem to the problem of finding a feasible flow of maximum value in an ad hoc network?

Graphical model:

Bipartite graph of competences



Definition 9

Given an undirected bipartite graph $G = (N, E)$, a matching $M \subseteq E$ is a subset of non adjacent edges.

Problem 10

Given a bipartite graph $G = (N, E)$, determine a matching with a maximum number of edges.

2.5 Hard graph optimization problems

Goal of computational complexity

Evaluate the computational requirements (we focus on time) to solve computational problems.

Two major types of issues:

- Evaluate the complexity of a given algorithm A to solve a given problem P.
- Evaluate the inherent difficulty of a given problem P.

The focus here is on discrete optimization problems.

Definition 2

An algorithm is polynomial if it requires, in the worst case, a polynomial number of elementary operations $f(n) = O(n^d)$, where d is a constant and $n = |I|$.

We distinguish between: $O(n^d)$ (polynomial) and $O(2^n)$ (exponential) algorithms.

Polynomial algorithms with $d \geq 6$ are not efficient in practice!

Examples

- Dijkstra's algorithm for the shortest path problem
  Size of an instance: $|I| = O(m \log_2 n + m \log_2 c_{\max})$
  Time complexity: $O(n^2)$, where n is the number of nodes
  Algorithm polynomial w.r.t. $|I|$ ($|I| \geq m \geq n - 1$).
- Basic version of Ford-Fulkerson's algorithm for the maximum flow problem
  Size of an instance: $|I| = O(m \log_2 n + m \log_2 k_{\max})$
  Time complexity: $O(m^2 k_{\max})$, where m is the number of arcs
  Algorithm not polynomial w.r.t. $|I|$.

Chapter 2

Example 1: Diet problem

Given

n aliments $j = 1, \ldots, n$
m nutrients (basic substances) $i = 1, \ldots, m$
a_{ij} amount of i-th nutrient contained in one unit of the j-th aliment
b_i daily requirement of the i-th nutrient
c_j cost of a unit of j-th aliment,

determine a diet that minimizes the total cost while satisfying all daily requirements.

Model:

Decision variables: x_j = amount of j-th aliment in the diet, with $j = 1, \ldots, n$

$$\begin{aligned} \min \quad & z = \sum_{j=1}^{n} c_j x_j \\ \text{s. t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, \ldots, m \\ & x_j \geq 0, \; j = 1, \ldots, n \end{aligned}$$

Example 2: Transportation problem (single product)

Given

m production plants $i = 1, \ldots, m$
n clients $j = 1, \ldots, n$
c_{ij} unit transportation cost from plant i to client j
p_i maximum supply (production capacity) of plant i
d_j demand of client j
q_{ij} maximum amount transportable from plant i to client j

determine a transportation plan that minimizes the total costs while respecting plant capacities and client demands

Assumption: $\sum_{i=1}^{m} p_i \geq \sum_{j=1}^{n} d_j$

Model:

Decision variables: x_{ij} = amount of product transported from i to j, $i = 1, \ldots, m$, $j = 1, \ldots, n$

$$\begin{aligned}
\min \quad & z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s. t.} \quad & \sum_{j=1}^{n} x_{ij} \leq p_i, \quad i = 1, \ldots, m && \text{(plant capacity)} \\
& \sum_{i=1}^{m} x_{ij} \geq d_j, \quad j = 1, \ldots, n && \text{(client demand)} \\
& 0 \leq x_{ij} \leq q_{ij}, \; i = 1, \ldots, m, \; j = 1, \ldots, n && \text{(transport. capacity)}
\end{aligned}$$

Example 3: Production planning problem

Given

m resources $i = 1, \ldots, m$
n products $j = 1, \ldots, n$ which compete for resources
c_j profit (selling price-cost) per unit of j-th product
a_{ij} amount of i-th resource needed to produce one unit of j-th product
b_i maximum available amount of i-th resource

determine a production plan that maximizes the total profit given the available resources.

Model: Decision variables: x_j = amount of j-th product, with $j = 1, \ldots, n$

$$\begin{aligned}
\max \quad & z = \sum_{j=1}^{n} c_j x_j \\
\text{s. t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m \\
& x_j \geq 0, \; j = 1, \ldots, n
\end{aligned}$$

Assumptions of LP models

1. Linearity of the objective function and constraints.

   Proportionality: Contribution of each variable = constant $\times$ variable
   Drawback: does not account for economies of scale.

   Additivity: Contribution of all variables = sum of single contributions
   Drawback: Competing products $\Rightarrow$ profits are not independent.

2. Divisibility: The variables can take fractional (rational) values.

3. Parameters are considered as constants which can be estimated with a sufficient degree of accuracy.

   More complex mathematical programs are needed to account for uncertainty.

LP "sensitivity analysis" allows to evaluate how sensitive an optimal solution is with respect to small changes in the parameter values (see end of Chapter 3).

3.1 Equivalent forms

General form:

$$\begin{array}{lll} \min(\max) & z = \underline{c}^T \underline{x} & \\ \text{s. t.} & A_1 \underline{x} \geq \underline{b_1} & \text{inequality constraints} \\ & A_2 \underline{x} \leq \underline{b_2} & \text{inequality constraints} \\ & A_3 \underline{x} = \underline{b_3} & \text{equality constraints} \\ & x_j \geq 0, \quad \text{for } j \in J \subseteq \{1, \ldots, n\} & \\ & x_j \text{ free}, \quad \text{for } j \in \{1, \ldots, n\} \backslash J & \end{array}$$

Definition 3 (Standard form)

$$\begin{array}{lll} \min & z = \underline{c}^T \underline{x} & \\ \text{s. t.} & A\underline{x} = \underline{b} & \text{only equality constraints and} \\ & \underline{x} \geq \underline{0} & \text{all nonnegative variables} \end{array}$$

- The two forms are equivalent.
- Simple transformation rules allow to pass from one form to the other form.

Warning: The transformation may involve adding/deleting variables and/or constraints.

Transformation rules

- $\max \underline{c}^T \underline{x} = -\min(-\underline{c}^T \underline{x})$
- $\underline{a}^T \underline{x} \leq b \Rightarrow \begin{cases} \underline{a}^T \underline{x} + s = b \\ s \geq 0 \end{cases}$ where s is a slack variable.
- $\underline{a}^T \underline{x} \geq b \Rightarrow \begin{cases} \underline{a}^T \underline{x} - s = b \\ s \geq 0 \end{cases}$ where s is a surplus variable.
- x_j unrestricted in sign $\Rightarrow \begin{cases} x_j = x_j^+ - x_j^- \\ x_j^+, x_j^- \geq 0 \end{cases}$.

  After substituting x_j with $x_j^+ - x_j^-$, we delete x_j from the problem.

3.2 Geometry of Linear Programming

Example Capital budgeting

Capital of 10 000€ and two possible investments A and B with, respectively, 4% and 6% expected return.

Determine a portfolio that maximizes the total expected return, while respecting the diversification constraints:

- at most 75% of the capital is invested in A,
- at most 50% of the capital is invested in B.

Model:

- x_A = amount invested in A
- x_B = amount invested in B

$$\begin{array}{llrcl} \max & z = 0.04x_A & +0.06x_B & & \\ \text{s. t.} & x_A & +x_B & \leq & 10\,000 \\ & x_A & & \leq & 0.75 \cdot 10\,000 \\ & & x_B & \leq & 0.50 \cdot 10\,000 \\ & x_A, & x_B & \geq & 0 \end{array}$$

Algorithm 1: Simplex algorithm (LP with minimization)

```
Let B[1], ..., B[m] be the column indices of the initial feasible basis B.
Construct the initial tableau Ā = {ā[i,j] : 0 ≤ i ≤ m, 0 ≤ j ≤ n} in canonical
  form w.r.t. B.
unbounded ← false
optimal ← false
while optimal = false and unbounded = false do
    if ā[0,j] ≥ 0 ∀j = 1,...,n then  optimal ← true/* for LP with min */

    else
        Select a non basic variable x_s with ā[0,s] < 0 /* negative reduced
            cost                                                          */

        if ā[i,s] ≤ 0 ∀i = 1,...,m then  unbounded ← true
        else
            r ← argmin { ā[i,0]/ā[i,s] with 1 ≤ i ≤ m and ā[i,s] > 0 }
            pivot(r,s) /* update tableau                                  */
            B[r] ← s
```

Economic interpretation

The primal and dual problems correspond to two complementary point of views on the same "market".

Diet problem:

n aliments $j = 1, \dots, n$
m nutrients $i = 1, \dots, m$ (vitamines,...)
a_{ij} quantity of i-th nutrient in one unit of j-th aliment
b_i requirement of i-th nutrient
c_j cost of one unit of j-th aliment

$$(P) \quad \begin{array}{ll} \min & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, \dots, m \\ & x_j \geq 0, \quad j = 1, \dots, n \end{array} \qquad (D) \quad \begin{array}{ll} \max & \sum_{i=1}^{m} b_i y_i \\ \text{s.t.} & \sum_{i=1}^{m} a_{ij} y_i \leq c_j, \quad j = 1, \dots, n \\ & y_i \geq 0, \quad i = 1, \dots, m \end{array}$$

Interpretation of the dual problem:

A company that produces pills of the m nutrients needs to decide the nutrient unit prices y_i so as to maximize income.

- If the costumer buys nutrient pills, he will buy b_i units for each i, $i = 1 \dots m$.
- The price of the nutrient pills must be competitive:

$$\sum_{i=1}^{m} a_{ij} y_i \leq c_j, \quad j = 1, \dots, n$$

(cost of the pills that are equivalent to 1 unit of j-th aliment)

If both (P) and (D) admit a feasible solution, the strong duality theorem implies that $z^* = w^*$.

An "equilibrium" exists (two alternatives with the same cost).

Note: Strong connection with Game theory (zero-sum games).

3.6 Sensitivity analysis in Linear Programming

Evaluate the "sensitivity" of an optimal solution with respect to variations in the model parameters.

Example: Production planning

$$\begin{array}{lll} \max & \sum_{j=1}^{n} p_j x_j & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, & i = 1, \dots, m \\ & x_j \geq 0, & j = 1, \dots, n \end{array}$$

- p_j is the profit of one unit of j-th product
- b_i is the availability of i-th resource

Chapter 3- Final Chapter

Example 1: Knapsack problem

Given

n objects $j = 1, \ldots, n$

p_j profit (value) of object j

w_j weight of object j

b maximum knapsack capacity.

determine a subset of objects that maximizes the total profit, while respecting the knapsack capacity.

Variables: $x_j = \begin{cases} 1 & \text{if } j\text{-th object is selected} \\ 0 & \text{otherwise} \end{cases}$

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} p_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} w_j x_j \leq b \\ & x_j \in \{0,1\}, \quad j = 1, \ldots, n \end{aligned}$$

Example 2: Assignment problem

Given

m machines, $i = 1, \ldots, m$

n jobs, $j = 1, \ldots, n$ (Assuming that $n > m$)

c_{ij} cost of assigning job j to machine i

determine an assignment of jobs to the machines so as to minimize the total cost, while assigning at least one job per machine and at most one machine for each job.

Variables: $x_{ij} = \begin{cases} 1 & \text{if machine } i \text{ executes job } j \\ 0, & \text{otherwise} \end{cases}$

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{m} x_{ij} \leq 1, \quad j = 1, \ldots, n \quad \text{(at most one machine for each job)} \\ & \sum_{j=1}^{n} x_{ij} \geq 1, \quad i = 1, \ldots, m \quad \text{(at least one job for each machine)} \\ & x_{ij} \in \{0,1\}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \end{aligned}$$

Example 3: Transportation problem

Given

m production plants, $i = 1, \ldots, m$

n clients, $j = 1, \ldots, n$ (assumption: $n > m$)

c_{ij} transportation cost of one unit of product from plant i to client j

p_i production capacity of plant i

d_j demand of client j

q_{ij} maximum amount to be transported from plant i to client j

determine a transportation plan that minimizes total costs while satisfying plant capacitiy and client demands.

Assumption $\sum_{i=1}^{m} p_i \geq \sum_{j=1}^{n} d_j$

Variables: x_{ij} = amount transported from plant i to client j

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} \leq p_i, \quad i = 1, \ldots, m \quad \text{(plant capacity)} \\
& \sum_{i=1}^{m} x_{ij} \geq d_j, \quad j = 1, \ldots, n \quad \text{(client demand)} \\
& 0 \leq x_{ij} \leq q_{ij}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \quad \text{(transportation capacity)}
\end{aligned}
$$

Note: the assignment problem is a special case.

Property of transportation and assignment problems:

Optimal solution of the linear relaxation $\equiv$ optimal solution of the ILP!

In fact, efficient (polynomial-time) ad hoc algorithms are known for both.

Example 4: Scheduling problem

Given

m machines, $k = 1, \ldots, m$

n jobs, $j = 1, \ldots, n$

d_j deadline for job j, $j = 1, \ldots, n$

p_{jk} processing time of job j on machine k (may be $= 0$)

Assumption: Each job must be processed once on each machine following the order of the machine indices $1, 2, \ldots, m$.

Determine an optimal sequence in which to process the jobs so as to minimize the total completion time while satisfying the deadlines.

Variables

t_{jk} = time at which the processing of job j starts on machine k

t = upper bound on the completion time of all jobs

$$y_{ijk} = \begin{cases} 1, & \text{if job } i \text{ preceds job } j \text{ on machine } k \\ 0, & \text{otherwise} \end{cases}$$

Parameter $M := \max_j d_j$

$$\begin{array}{ll}
\min & t \\
\text{s.t.} & t_{jm} + p_{jm} \leq t, \quad j = 1, \ldots, n \quad (t \text{ is upper bound on overall completion time}) \\
& t_{jm} + p_{jm} \leq d_j, \quad j = 1, \ldots, n \quad (\text{satisfy deadlines}) \\
& t_{jk} + p_{jk} \leq t_{j,k+1}, \quad j, k = 1, \ldots, m-1 \\
& t_{ik} + p_{ik} \leq t_{jk} + M(1 - y_{ijk}), \quad \forall i, j, k \quad i < j \quad (1) \\
& t_{jk} + p_{jk} \leq t_{ik} + M\, y_{ijk}, \quad \forall i, j, k \quad i < j \quad (2) \\
& t \geq 0, \quad t_{jk} \geq 0, \quad j, k = 1, \ldots, m \\
& y_{ijk} \in \{0, 1\}, \quad i, j, k = 1, \ldots, m
\end{array}$$

Mixed ILP!

- (1) and (2) make sure that 2 jobs are not simultaneously processed on the same machine
- (1) active when $y_{ijk} = 1$ (i preceeds j on machine k) and ensures that i is completed before j starts (on k)
- (2) active when $y_{ijk} = 0$ (j preceeds i on machine k) and ensures that j is completed before i starts (on k).

ILP formulation can be extended to the case where each job j must be processed on (a subset of) the m machines according to a different order.

5.1.2 Branch-and-Bound for a combinatorial optimization problem

Example: Scheduling problem on a single machine (NP-hard)

Given n jobs with deadlines, and a single machine with processing time for each job, determine a sequence that minimizes the total delay.

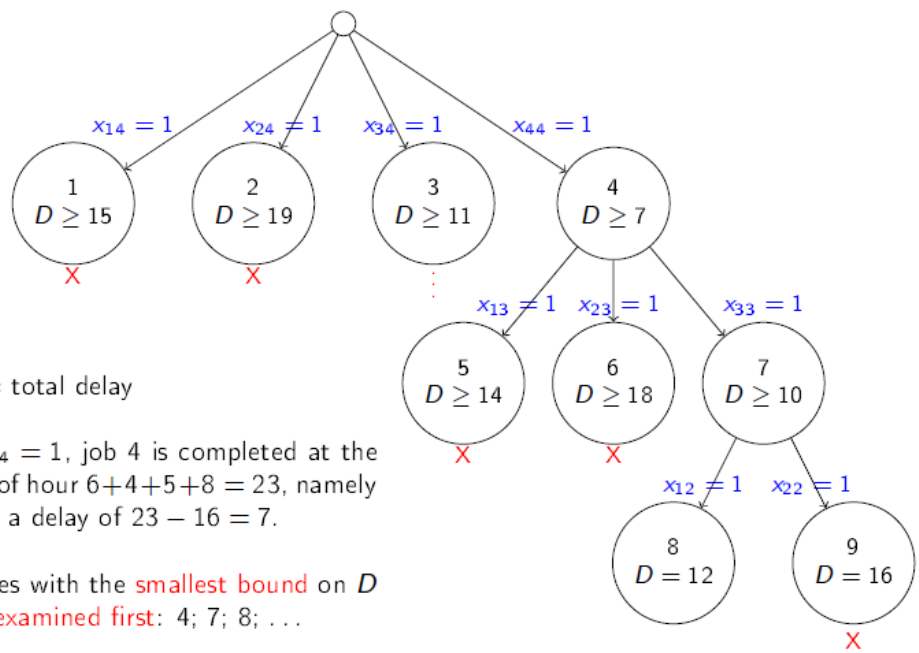
jobs	processing time (h)	deadline (completed by hour)
1	6	8
2	4	4
3	5	12
4	8	16

The sequence 1–2–3–4, has a total delay of $0 + 6 + 3 + 7 = 16$ hours.

Define $x_{ij} = \begin{cases} 1 \text{ if job } i \text{ is processed in } j\text{-th position} \\ 0 \text{ otherwise} \end{cases}$

Idea: Partition the set of all feasible solutions based on the last job that is processed.

Clearly $x_{14} = 1$ or $x_{24} = 1$ or $x_{34} = 1$ or $x_{44} = 1$.



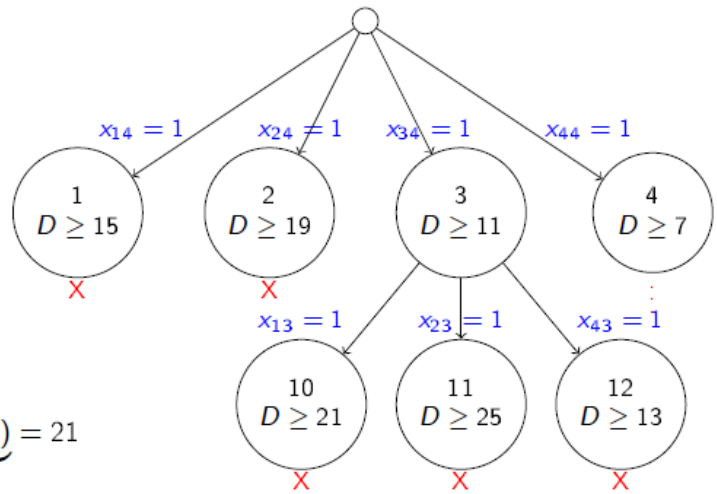
D = total delay

If $x_{44} = 1$, job 4 is completed at the end of hour $6+4+5+8 = 23$, namely with a delay of $23 - 16 = 7$.

Nodes with the smallest bound on D are examined first: 4; 7; 8; ...

- For node 7:
  - Job 4 is the last one with a delay of 7 hours
  - Job 3 is the next to last one with a delay of $6+4+5-12=15-12=3$ hours $\Rightarrow D \geq 7+3=10$
- Node 8: Sequence 2–1–3–4 is a feasible solution with delay 12
- Node 9: Sequence 1–2–3–4 is a feasible solution with delay 16

Note: Nodes 1, 2, 5 and 6 can be "pruned"!



$$11+\underbrace{(6+4+8-8)}_{\text{job 1 next to last}}=21$$

$$11+\underbrace{(6+4+8-16)}_{\text{job 4 next to last}}=13$$

$\Rightarrow$ Optimal sequence: 2–1–3–4 with $D=12$

4.2.3 Cutting plane method with fractional Gomory cuts

Algorithm 1: Cutting plane method with fractional Gomory cuts

```
Solve the linear relaxation min{c^T x : Ax = b, x ≥ 0}
Let x*_LP' be an optimal basic feasible solution
while x*_LP has fractional components do
    Select a basic variable with a fractional value
    Generate the corresponding Gomory cut
    Add constraint to the optimal tableau of the linear relaxation
    Perform one iteration of the dual simplex algorithm
```

Theorem 5

If the ILP has a finite optimal solution, the cutting plane method finds one after adding a finite number of Gomory cuts.

(But often very large.)

Exercises' models

1.1 Portfolio optimization

A bank has a capital of C billions of Euro and two available stocks:

1. with an annual revenue of 15% and a risk factor of $\frac{1}{3}$,
2. with an annual revenue of 25% and risk factor of 1.

The risk factor represents the maximum fraction of the stock value that can be lost. A risk factor of 0.25 implies that, if stocks are bought for 100 Euro up to 25 Euro can be lost. It is required that at least half of C is risk-free. The amount of money used to buy stocks of (2) must not be larger than two times that used to buy stocks of (1). At least $\frac{1}{6}$ of C must be invested into (1).

Give a Linear Programming formulation for the problem of determining an optimal portfolio for which the profit is maximized. Solve the problem graphically.

Parameters

- C: available capital

Decision variables

- x_1: amount of money invested in stock of type 1
- x_2: amount of money invested in stock of type 2

s.t.

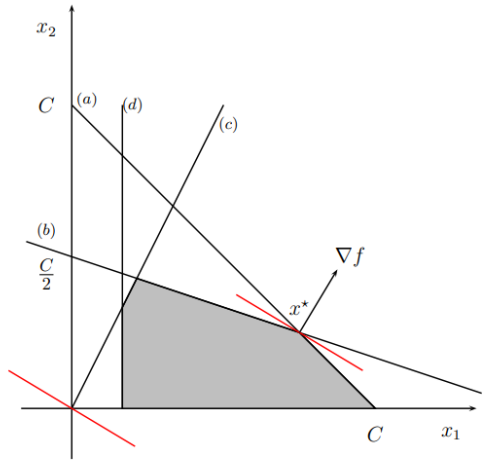
$$\begin{aligned} x_1 + x_2 &\leq C \\ \frac{1}{3}x_1 + x_2 &\leq \frac{1}{2}C \\ x_2 &\leq 2x_1 \\ x_1 &\geq \frac{1}{6}C \\ x_1, x_2 &\geq 0 \end{aligned}$$

To solve the problem graphically, we must identify the feasible region in $\mathbb{R}^2$ that satisfies the constraints (a) $x_2 \leq -x_1 + C$, (b) $x_2 \leq -\frac{1}{3}x_1 + \frac{C}{2}$, (c) $x_2 \leq 2x_1$, (d) $x_1 \geq \frac{1}{6}C$, (e) $x_1 \geq 0$, and (f) $x_2 \geq 0$. For any two variables problem, any linear inequality constraint divides $\mathbb{R}^2$ in two halfplanes.

To draw a constraint, it suffices to find any two points that satisfy it with equality (as an equation). The "border" of the constraint is then represented by the only line containing such points. For simplicity, it is convenient to choose the origin $(0, 0)$ as the first point, if feasible, and if it is not, the points $(x_1, 0)$ and $(0, x_2)$, where x_1 and x_2 are two unknowns that are to be determined w.r.t. the constraint. To identify which of the two halfplanes is the feasible one, two ways are possible. In the first one, it suffices to pick a random point and checking whether it satisfies the constraint (i.e., the inequality). If it does, the halfspace to which the point belongs is the feasible one, otherwise the other halfspace is. Alternatively, we can consider the gradient of the constraint and compare it to the direction of the inequality. For $\geq$ inequalities, the feasible halfplane is that in the direction of the gradient, while for $\leq$ inequalities, it is the other one.

The feasible region is as shown in the picture. To find the feasible point where the objective function attains its maximal value, we can draw the *level curves* $f(x_1, x_2) = 0.15x_1 + 0.25x_2 = z$, where each level curve is the set of points whose objective function value is equal to z, for any constant z.

Since f is linear, the level curve $f(x_1, x_2) = z$ is a line, orthogonal to its gradient, and parametric in z. Clearly, when z is increased, we obtain parallel level lines that move towards the direction of the gradient $\nabla f(x_1, x_2)$.



Note that, by starting with $z = 0$ and by increasing it in a continuous way, the level lines of f will first intersect the feasible region at $(\frac{C}{6}, 0)$, and then, increasing z, at any other point, until the intersection is empty. The last feasible point(s) having a nonempty intersection is (are) the maximizer(s) of f over the feasible set. In this problem, since lines (a) and (b) are not parallel and the level lines are not parallel to lines (a) or (b), there is a single maximizer that corresponds to the intersection of the lines (a) and (b). The maximizer, denoted by x^*, can be found as the solution to the following (trivial) linear system

$$\begin{cases} x_2 = -x_1 + C \\ x_2 = -\frac{1}{3}x_1 + \frac{1}{2}C, \end{cases}$$

which yields $\underline{x}^* = (\frac{3C}{4}, \frac{C}{4})$, where $f(\underline{x}^*) = \frac{7C}{40}$.

1.2 Gasoline mixture

A refinery produces two types of gasoline, mixing three basic oils according to the following gasoline mixture rules:

	Oil 1	Oil 2	Oil 3	Revenue
Gasoline A	$\leq 30\%$	$\geq 40\%$	-	5.5
Gasoline B	$\leq 50\%$	$\geq 10\%$	-	4.5

The last column of the previous table indicates the profit (Euro/barrel). The availability of each type of oil (in barrel) and the cost (Euro/barrel) are as follows:

Oil	Availability	Cost
1	3000	3
2	2000	6
3	4000	4

Give a Linear Programming formulation for the problem of determining a mixture that maximizes the profit (difference between revenues and costs).

Decision variables

- x_{ij}: amount of the i-th oil used to produce the j-th gasoline, for $i \in \{1, 2, 3\}$ and $j \in \{A, B\}$
- y_j: amount of gasoline of type j-th that is produced, for $j \in \{A, B\}$

The total revenue is $5.5y_A + 4.5y_B$, the total cost, due to the amount of oil 1, 2, and 3 that is used is, respectively, $3\sum_{j\in\{A,B\}} x_{1j}$, $6\sum_{j\in\{A,B\}} x_{2j}$, and $4\sum_{j\in\{A,B\}} x_{3j}$.

Model

$$
\begin{array}{lll}
\max & 5.5y_A + 4.5y_B - 3(x_{1A} + x_{1B}) & \\
 & \quad - 6(x_{2A} + x_{2B}) - 4(x_{3A} + x_{3B}) & \text{(profit)} \\
\text{s.t.} & & \\
 & x_{1A} + x_{1B} \leq 3000 & \text{(availability 1)} \\
 & x_{2A} + x_{2B} \leq 2000 & \text{(availability 2)} \\
 & x_{3A} + x_{3B} \leq 4000 & \text{(availability 3)} \\
 & y_A = x_{1A} + x_{2A} + x_{3A} & \text{(conservation A)} \\
 & y_B = x_{1B} + x_{2B} + x_{3B} & \text{(conservation B)} \\
 & x_{1A} \leq 0.3y_A & \text{(min. quantity A)} \\
 & x_{1B} \leq 0.5y_B & \text{(min. quantity B)} \\
 & x_{2A} \geq 0.4y_A & \text{(max. quantity A)} \\
 & x_{2B} \geq 0.1y_B & \text{(max. quantity B)} \\
 & x_{1A}, x_{2A}, x_{3A}, x_{1B}, x_{2B}, x_{3B}, y_A, y_B \geq 0 & \text{(nonnegative variables)}
\end{array}
$$

The variables y_A and y_B can be removed from the formulation by substitution: they are replaced in the objective function and in the constraints by the right-hand side terms of equations (*conservation A*) and (*conservation B*). This gives the following more compact formulation:

$$
\begin{array}{lll}
\max & 5.5(x_{1A} + x_{2A} + x_{3A}) + 4.5(x_{1B} + x_{2B} + x_{3B}) & \\
 & \quad - 3(x_{1A} + x_{1B}) - 6(x_{2A} + x_{2B}) - 4(x_{3A} + x_{3B}) & \text{(profit)} \\
\text{s.t.} & & \\
 & x_{1A} + x_{1B} \leq 3000 & \text{(availability 1)} \\
 & x_{2A} + x_{2B} \leq 2000 & \text{(availability 2)} \\
 & x_{3A} + x_{3B} \leq 4000 & \text{(availability 3)} \\
 & x_{1A} \leq 0.3(x_{1A} + x_{2A} + x_{3A}) & \text{(min. quantity A)} \\
 & x_{1B} \leq 0.5(x_{1B} + x_{2B} + x_{3B}) & \text{(min. quantity B)} \\
 & x_{2A} \geq 0.4(x_{1A} + x_{2A} + x_{3A}) & \text{(max. quantity A)} \\
 & x_{2B} \geq 0.1(x_{1B} + x_{2B} + x_{3B}) & \text{(max. quantity B)} \\
 & x_{1A}, x_{2A}, x_{3A}, x_{1B}, x_{2B}, x_{3B}, y_A, y_B \geq 0 & \text{(nonnegative variables)}
\end{array}
$$

Both formulations are valid. The former one contains two additional variables and two additional constraints with respect to the latter one. In the latter formulation, the constraints are less sparse (they involve more variables due to the substitutions).

The two formulations describe a very particular instance of the gasoline mixture problem in which only two types of gasoline are produced from only three oils, for some specific values of the parameters. The models are, in a sense, not general.

When describing a problem (even though we are interested in solving a particular instance), it is advisable to give an abstract description of it, which does not depend on the data of the instance at hand.

To obtain a general formulation of the gasoline mixture problem, it suffices to define the variables and parameters of the problem as vectors and matrices, with indices belonging to well-defined sets.

Sets

- I: types of oil
- J: types of gasoline

Parameters

- c_i: cost per barrel of the i-th type of oil, for $i \in I$
- b_i: maximum availability of the i-th type of oil, for $i \in I$
- r_j: revenue per barrel of the j-th type of gasoline, for $j \in J$
- $q_{ij}^{\max}$: maximum quantity (as fraction of the unit) of the i-th type of oil in the j-th type of gasoline, for $i \in I$ and $j \in J$
- $q_{ij}^{\min}$: minimum quantity (as fraction of the unit) of the i-th type of oil in the j-th type of gasoline, for $i \in I$ and $j \in J$

Decisions variables

- x_{ij}: quantity of the i-th type of oil in the j-th type of gasoline, for $i \in I$ and $j \in J$
- y_j: quantity of the j-th type of gasoline produced, for $j \in J$

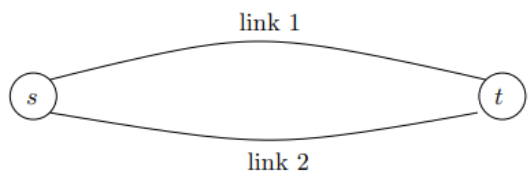
Model

$$
\begin{array}{llll}
\max & \displaystyle\sum_{j \in J} r_j y_j - \sum_{i \in I, j \in J} c_i x_{ij} & & \text{(profit)} \\
\text{s.t.} & & & \\
& \displaystyle\sum_{j \in J} x_{ij} \leq b_i & i \in I & \text{(availability)} \\
& y_j = \displaystyle\sum_{i \in I} x_{ij} & j \in J & \text{(conservation)} \\
& x_{ij} \leq q_{ij}^{\max} y_j & i \in I, j \in J & \text{(min. quantity)} \\
& x_{ij} \geq q_{ij}^{\min} y_j & i \in I, j \in J & \text{(max. quantity)} \\
& x_{ij}, y_j \geq 0 & i \in I, j \in J & \text{(non negative variables)}
\end{array}
$$

Note the differences between this formulation and the previous ones. Every "row" of this model describes a group, or family, of constraints, via the mathematical quantifiers. Observe that some constraints are defined for each combination of type of oil and type of gasoline, while, in the statement of the problem, some of those combinations where not expressed (e.g., no minimum or maximum quantities were given for oil 3). In this case, the parameters will be chosen so that these constraints are always satisfied for the specific combination of indices, e.g, by letting the minimum quantity equal to 0 and the maximum quantity equal to 1. As a simple exercise, remove each variable y_j from the formulation.

1.3 Packet routing

n packets of data must be routed from node s to node t, along one of two available links, with capacity (bandwidth) $k_1 = 1$ Mbps and $k_2 = 2$ Mbps.



The cost per unit of capacity of link 2 is 30% larger than that of link 1. The following table indicates the quantity of capacity consumed by each packet i, $i \in \{1 \dots n\}$, and the cost to route it on link 1.

Packed	Consumed capacity	Cost on link 1
1	0.3	200
2	0.2	200
3	0.4	250
4	0.1	150
5	0.2	200
6	0.2	200
7	0.5	700
8	0.1	150
9	0.1	150
10	0.6	900

Give an integer linear programming formulation for the problem of minimizing the total cost of routing all the packets. Give also an integer linear programing formulation for the more general case where m links are available.

1.3 **Packet routing.**

The integer linear programming formulation for the 2-link case is as follows

Sets

- $I = \{1 \dots n\}$: set of packets

Parameters

- a_i: capacity consumed by packet i, for $i \in I$
- c_{i1}, c_{i2}: routing cost for packet i on link 1 and 2, for $i \in I$
- k_1, k_2: capacity for link 1 and 2

Variables

- x_i: 1 if packet i is routed on link 1, 0 if routed on link 2, for $i \in I$

Model

$$\begin{array}{lll} \min & \sum_{i \in I} c_{i1} x_i + c_{i2}(1 - x_i) & & \text{(cost)} \\ \text{s.t.} & & & \\ & \sum_{i \in I} a_i x_i \leq k_1 & & \text{(capacity 1)} \\ & \sum_{i \in I} a_i (1 - x_i) \leq k_2 & & \text{(capacity 2)} \\ & x_i \in \{0, 1\} & i \in I & \text{(binary variables)} \end{array}$$

The m-link formulation requires a new set of binary variables, one for each packet and link. The packet-to-link assignment is also to be explicitly introduced.

Sets

- $I = \{1 \dots n\}$: set of packets
- $J = \{1 \dots m\}$: set of links

Parameters

- a_i: capacity consumed by packet i, for $i \in I$
- c_{ij}: routing cost for packet i on link j, for $i \in I, j \in J$
- k_j: capacity of link j, for $j \in J$

Variables

- x_{ij}: 1 if packet i is routed on link j, 0 otherwise, for $i \in I, j \in J$

Model

$$\begin{array}{lll} \min & \sum_{i \in I, j \in J} c_{ij} x_{ij} & & \text{(cost)} \\ \text{s.t.} & & & \\ & \sum_{j \in J} x_{ij} = 1 & i \in I & \text{(assignment)} \\ & \sum_{i \in I} a_i x_{ij} \leq k_j & j \in J & \text{(capacity)} \\ & x_{ij} \in \{0, 1\} & i \in I, j \in J & \text{(binary variables)} \end{array}$$

1.4 Multi-period production planning (uncapacitated lot sizing)

A company A, which produces one type of high-precision measuring instrument, has to plan the production for the next 3 months. Each month, A can produce at most 110 units, at a unit cost of 300 Euro. Moreover, each month, up to 60 additional units produced by another company B can be bought at a unit cost of 330 Euro. Unsold units can be stored. The inventory cost is of 10 Euro per unit of product, per month. Sales forecasts indicate a demand of 100, 130, and 150 units of product for the next 3 months.

1. Give a linear programming formulation for the problem of determinig a production plan (direct or indirect) which minimizes the total costs, while satisfying the monthly demands.
2. Give a mixed integer linear programming formulation for the variant of the problem where production lots have a minimum size. In particular, if any strictly positive quantity is produced in a given month, this quantity cannot be smaller than 15 units.

Sets

- $T = \{1 \dots 3\}$: set of months

Parameters

- b: production capacity of A
- b': production capacity of B
- c: unit production cost for A
- c': unit production cost for B
- m: inventory cost per unit and month
- d_t: sales forecasts for month t, for $t \in T$

Variables

- x_t: units produced by A in month t, for $t \in T$
- x'_t: units bought from B in month t, for $t \in T$
- z_t: units in inventory at the end of month t, for $t \in T \cup \{0\}$

Model

$$
\begin{array}{lcll}
\min & \sum_{t \in T}(cx_t + c'x'_t + mz_t) & & \text{(cost)} \\
\text{s.t.} & & & \\
 & x_t \leq b & t \in T & \text{(capacity of A)} \\
 & x'_t \leq b' & t \in T & \text{(capacity of B)} \\
 & z_{t-1} + x_t + x'_t \geq d_t & t \in T & \text{(demand)} \\
 & z_{t-1} + x_t + x'_t - d_t = z_t & t \in T & \text{(inventory balance)} \\
 & z_0 = 0 & & \text{(starting condition)} \\
 & x_t, x'_t, z_t \geq 0 & t \in T & \text{(nonnegative variables)}
\end{array}
$$

Observe that the (*demand*) constraint is redundant, as it is implied by $z_{t-1}+x_t+x'_t-d_t = z_t \geq 0$.

To take into account the minimum lot size, we add the binary variables

- y_t: 1 if production is active at month t, 0 otherwise, for $t \in T$

and the constraints

$$
\begin{array}{rlll}
[rlll] & x_t \geq l y_t & t \in T & \text{(minimum lot size)} \\
 & x_t \leq M y_t & t \in T & \text{(activation)},
\end{array}
$$

where $l = 15$ is the minimum lot size, and M is a large enough value, such that constraint $x_t \leq M y_t$ is redundant when $y_t = 1$. For instance, we can choose $M = 110$, i.e., equal to the monthly productive capacity. Such constraints are usually called *big-M* constraints.

1.5 Computer rental planning

A company must rent computers in order to face the requirements during the next four months:

Month	January	February	March	April
Requirements	9	5	7	9

The rental cost depends on the length of the rental as follows:

Length	1 month	2 months	3 months
Cost (Euro)	200	350	450

Give an integer linear programming formulation for the problem of finding a rental plan of minimum total cost.

1.5 Computer rental planning

Using the variables

- g_1, g_2, g_3: number of computers rented in January for 1, 2, and 3 months
- f_1, f_2, f_3: number of computers rented in February for 1, 2, and 3 months
- m_1, m_2: number of computers rented in March for 1 and 2 months
- a_1: number of computers rented in April for 1 month

the problem can be formulated as the following integer linear program:

$$
\begin{aligned}
\min \quad & 200(g_1 + f_1 + m_1 + a_1) && && (1)\\
& + 350(g_2 + f_2 + m_2) + 450(g_3 + f_3) && && (2)\\
\text{s.t.} \quad & && && (3)\\
& g_1 + g_2 + g_3 \geq 9 && \text{(January)} && (4)\\
& f_1 + f_2 + f_3 + g_2 + g_3 \geq 5 && \text{(February)} && (5)\\
& m_1 + m_2 + f_2 + f_3 + g_3 \geq 7 && \text{(March)} && (6)\\
& a_1 + m_2 + f_3 \geq 9 && \text{(April)} && (7)\\
& g_1, g_2, g_3, f_1, f_2, f_3, m_1, m_2, a_1 \in \mathbb{Z}^+ && \text{(integer nonnegative variables)} && (8)
\end{aligned}
$$

By introducing the following index sets, parameters and variables:

Sets

- $I = \{1, \ldots, m\}$: months
- $J = \{1, \ldots, l\}$: rental durations (in months)

Parameters

- d_i: demand for month i, $i \in I$
- c_j: cost for rental of duration j, $j \in J$

Variables

- x_{ij}: number of computers rented in month i for j months, $i \in I, j \in J$

the model can be generalized as

$$\min \quad \sum_{i \in I, j \in J} c_j x_{ij} \qquad \text{(cost)} \tag{9}$$

$$\text{s.t.} \tag{10}$$

$$\sum_{k=\max\{1,i-l+1\}}^{i} \sum_{j=i-k+1}^{l} x_{kj} \geq d_i \qquad i \in I \quad \text{(demand)} \tag{11}$$

$$x_{ij} \in \mathbb{Z}^+ \qquad i \in I, j \in J. \qquad \text{(nonnegative integer variables)} \tag{12}$$

Constraints (demand) work as follows. The initial value for k is such that, if we can rent for, e.g., at most $l = 3$ months, when, e.g., $i = 7$, k will start from $7 - 3 + 1 = 5$. The min operator guarantees that $k \geq 1$ (consider, for instance, $i = 1, l = 3$, where we would have $i - l + 1 = 1 - 3 + 1 = -1$). With the second summation we take into account the fact that if, e.g., $i = 3$, $k = 2$, and $l = 3$, then j must be at least 2, otherwise, with $j = 1$, we would consider the variable x_{21} in the summation, which is wrong. Indeed, it amounts to consider, in month 3, machines which have been rented for one month in month 2, which are no longer available in month 3.

1.6 Workforce planning

A ICT services company has estimated the following demand for maintenance and consultancy for the next 5 months:

Month	1	2	3	4	5
Demand (hours)	6000	7000	8000	9500	11000

At the beginning of the planning period, 50 technicians are available. Each technician works at most 160 hours per month. To satisfy the demand, new technicians must be hired and trained. During the training period, which lasts one month, an expert technician must coach the newly hired one for 50 hours.

Expert technicians are payed 2000 Euro per month, while the newly hired ones are payed 1000 Euro, during the training month. We suppose that, at the end of each month, 5% of the expert technicians leave the company to join competitors.

Give a mathematical programming formulation for the problem of minimizing the total costs.

1.6 Workforce planning

Sets

- $I = \{1, \ldots, 5\}$: months

Parameters

- d_i: demand for month i, $i \in I$

Variables

- x_i: number of expert technicians available in month i, $i \in I$
- y_i: number of technicians training during month i, $i \in I$

Model

$$
\begin{array}{llll}
\min & 2000\,(x_1 + x_2 + x_3 + x_4 + x_5) + 1000\,(y_1 + y_2 + y_3 + y_4 + y_5) & & (13) \\
\text{s.t.} & & & (14) \\
& 160x_i - 50y_i \geq d_i & i \in I \quad \text{(demand)} & (15) \\
& \lfloor 0.95x_i \rfloor + y_i = x_{i+1} & i \in I \quad \text{(number)} & (16) \\
& x_1 = 50 & & (17) \\
& x_i, y_i \in \mathbb{Z}^+ \quad i \in I \quad \text{(nonnegative integer variables)} & & (18)
\end{array}
$$

Obviously, Constraint (number) is nonlinear. By dropping the $\lfloor \cdot \rfloor$ operator, we obtain the constraint

$$0.95x_i + y_i = x_{i+1} \quad i \in I$$

which is not correct since, due to the integrality requirements on the variables, it forces x_i to be a multiple of 100 (such that $0.95x_i$ will always be an integer value).

Constraint (number) can be formulated in a linear way by introducing for each $i \in I$ an auxiliary integer variable z_i that represents $\lfloor 0.95x_i \rfloor$, and by adding the auxiliary constraints:

$$
\begin{array}{llll}
z_i + y_i = x_{i+1} & i \in I & \text{(number2)} & (19) \\
100z_i \leq 95x_i \leq 100z_i + 100 & i \in I & \text{(floor)} & (20) \\
z_i \in \mathbb{Z}^+ \quad i \in I. \quad \text{(nonnegative integer variables)} & & & (21)
\end{array}
$$

Then we clearly have $z_i = \lfloor 0.95x_i \rfloor$.

1.7 Facility location and transportation

A distribution company has to supply a single type of good to n clients. Let d_j denote the demand of client j, where $j \in J = \{1, \dots n\}$. The goods must be stocked in warehouses before being delivered to the clients. The warehouses can be opened (activated) in m candidate sites (locations). Let f_i denote the cost for opening a warehouse at the candidate site $i \in I$, with $i \in I = \{1, \dots, m\}$, and b_i denote its maximum capacity. Let c_{ij} be the unit transportation cost from the warehouse in candidate site i to client j, and q_{ij} be the maximum quantity to be transported from the warehouse in candidate site i to client j. To make sure that the problem admits a feasible solution, assume that $\sum_{i \in I} b_i \geq \sum_{i \in I} d_j$.

1.7 Facility location and transportation

The problem is an important extension of the *transportation problem* (see beginning of Chapter 4) where we have to decide not only the transportation plan for supplying the goods from the wharehouses to the clients but also the candidate sites (locations) in which to open (activate) the warehouses.

Variables

- x_{ij}: amount of good to be transported from the warehouse in candidate site $i \in I$ to client $j \in J$
- y_i: 1 if the warehouse in candidate site $i \in I$ is opened, and 0 otherwise

Model

$$\min \sum_{i \in I, j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \qquad \text{(total cost)} \tag{22}$$

$$\text{s.t.} \sum_{j \in J} x_{ij} \leq b_i \qquad i \in I \quad \text{(availability 2)} \tag{23}$$

$$\sum_{i \in I} x_{ij} \geq d_j \qquad j \in J \quad \text{(demand)} \tag{24}$$

$$0 \leq x_{ij} \leq q_{ij} y_i \qquad i \in I, j \in J \quad \text{(bounds 2)} \tag{25}$$

$$y_i \in \{0, 1\} \qquad i \in I \quad \text{(binary vars)} \tag{26}$$

where the Constraints (bounds 2) link the x_{ij} variables and the corresponding y_i variables. In particular, they make sure that: i) if $y_i = 0$ for a given i then $x_{ij} = 0$ for every $j \in J$, and ii) if for a given i there exists at least one j such that $x_{ij} > 0$ then the corresponding y_i is equal to 1.

Clearly an alternative formulation is as follows:

$$\min \sum_{i \in I, j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \qquad \text{(total cost)} \tag{27}$$

$$\text{s.t.} \sum_{j \in J} x_{ij} \leq b_i y_i \qquad i \in I \quad \text{(availability)} \tag{28}$$

$$\sum_{i \in I} x_{ij} \geq d_j \qquad j \in J \quad \text{(demand)} \tag{29}$$

$$0 \leq x_{ij} \leq q_{ij} \qquad i \in I, j \in J \quad \text{(bounds)} \tag{30}$$

$$y_i \in \{0, 1\} \qquad i \in I \quad \text{(binary vars)} \tag{31}$$

which may seem at first preferable because the variables y_i appear in a smaller number of constraints (m instead of nm).

Although both formulations are correct it is easy to verify that, by summing up all the Constraints (quantity 2) for $j \in J$ with a unit multiplier, we obtain

$$\sum_{j \in J} x_{ij} \leq (\sum_{j \in J} q_{ij}) y_i$$

which is tighter than the Constraints (availability), whenever $\sum_{j \in J} q_{ij} < b_i$.

If we take the Constraints (availability) for some $i \in I$, fix $j \in J$, and sum them up with $-x_{ij'} \leq 0$ for all $j' \in J \setminus \{j\}$, we obtain

$$x_{ij} \leq b_i y_i$$

which is weaker than the Constraints (quantity) whenever $b_i > q_{ij}$.

1.8 Modem production

A company A, producing a single type of modem, is planning the production for the next 4 months. The production capacity is of 2500 units per month, at a cost of 7 Euro per unit. A second company B can be payed to produce extra units, at a cost of 9 Euro per unit, for a maximum of 700 units per month.

Inventory can be used, at a cost of 1 Euro per month per unit. 70 units are stored at the beginning of the planning period, and 300 units must be stored at the end of the period.

For technical reasons, the units must be produced in lots of, at least, 80 units. Sales forecasts for the next 4 months indicate the following demand

Month	1	2	3	4
Demand	2600	3000	2350	2700

Give an Integer Linear Programming (ILP) formulation for the problem of determining a production plan of minimum total cost.

Suppose that fixed charges f_A and f_B must be payed each month that production takes place in factory A and, respectively, B. How can we extend the ILP formulation to account for fixed charges?