Foundation Of Operations Research
Exercises

Christian Rossi

Academic Year 2023-2024

Abstract

Operations Research is the branch of applied mathematics dealing with quantitative methods to analyze and solve complex real-world decision-making problems.

The course covers some fundamental concepts and methods of Operations Research pertaining to graph optimization, linear programming and integer linear programming.

The emphasis is on optimization models and efficient algorithms with a wide range of important applications in engineering and management.

Contents

Chapter 1

Exercise session I

1.1 Linear programming modeling

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

1. With an annual revenue of 15% and 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 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 two must not be larger than two times that used to buy stocks of one. At least $\frac{1}{6}$ of C must be invested into one.

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

Solution

- The parameters are:
  - The quantity of available capital C.
- The decision variables are:
  - The amount of money invested in stock of type one x_1.
  - The amount of money invested in stock of type two x_2.

- The objective function is:

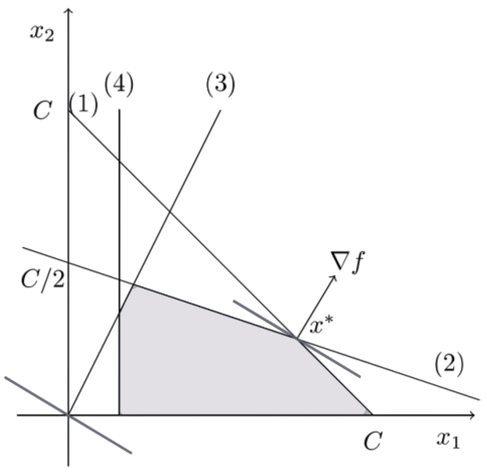
$$\max\,[0.15x_1 + 0.25x_2]$$

- The constraints are:
  - Maximum capital:
  $$x_1 + x_2 \leq C$$
  - Half of the invested capital is risk-free:
  $$\frac{1}{3}x_1 + x_2 \leq \frac{C}{2}$$
  - The amount of money used to buy stocks of two must not be larger than two times that used to buy stocks of one:
  $$x_2 \leq 2x_1$$
  - At least $\frac{1}{6}$ of C must be invested into one:
  $$x_1 \geq \frac{1}{6}C$$
  - Constraint on the variables:
  $$x_1, x_2 \geq 0$$

To solve the problem graphically, we must identify the feasible region in $\mathbb{R}^2$ that satisfies the constraints. 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. There are two possible ways to identify which of the two half planes is the feasible one:

- In the first one, it suffices to pick a random point and checking whether it satisfies the constraint. If it does, the half space to which the point belongs is the feasible one, otherwise the other half space is.
- Alternatively, we can consider the gradient of the constraint and compare it to the direction of the inequality.

The region found with the constraints is the following:



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. 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 $\left(\frac{C}{6}, 0\right)$, and then, increasing z, at any other point, until the intersection is empty. The last feasible point having a nonempty intersection is the maximizer of f over the feasible set. In this problem there is a single maximizer. The maximizer, denoted by x^*, can be found as the solution to the following linear system:

$$\begin{cases} x_1 + x_2 = C \\ \dfrac{1}{3}x_1 + x_2 = \dfrac{C}{2} \end{cases}$$

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

1.2 Linear programming modeling

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	$\leq 40\%$	$\geq 10\%$	-	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	3 000	3
2	2 000	6
3	4 000	4

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

Solution

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

$$\max 5.5y_A + 4.5y_B + 3(x_{1A} + x_{1B}) - 6(x_{2A} + x_{2B}) - 4(x_{3A} + x_{3B})$$

- The constraints are:
  - Availability of 1:
$$x_{1A} + x_{1B} \leq 3000$$
  - Availability of 2:
$$x_{2A} + x_{2B} \leq 2000$$

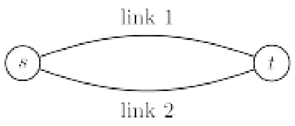
- Availability of 3:
$$x_{3A} + x_{3B} \leq 4000$$
- Conservation of A:
$$y_A = x_{1A} + x_{2A} + x_{3A}$$
- Conservation of B:
$$y_B = x_{1B} + x_{2B} + x_{3B}$$
- Minimum quantity of A:
$$x_{1A} \leq 0.3y_A$$
- Minimum quantity of B:
$$x_{1B} \leq 0.5y_B$$
- Maximum quantity of A:
$$x_{2A} \geq 0.4y_A$$
- Maximum quantity of B:
$$x_{2B} \geq 0.1y_B$$
- The variable must be non-negative:
$$x_{1A}, x_{2A}, x_{3A}, x_{1B}, x_{2B}, x_{3B}, y_A, y_B \geq 0$$

Chapter 2

Exercise session II

2.1 Linear programming modeling

Assume that 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, $i \in l, \dots, n$, and the cost to route it on link 1.

Packet	**Consumed capacity**	**Cost on link one**
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 programming formulation for the more general case where m links are available.

Solution

The 2-link case can be formulated as the following integer linear program.

- The sets are:
  - The set of packets $I = \{1, \ldots, n\}$.
  - The set of links $J = \{1, \ldots, m\}$.
- The parameters are:
  - The capacity consumed by packet i, for $i \in I$ a_i.
  - The routing cost for packet i on link j, for $i \in I, j \in J$ c_{ij}.
  - The capacity for link j and $j \in J$ k_j.
- The decision variables are:
  - $x_i j$: 1 if packet i is routed on link j, or 0 otherwise, for $i \in I, j \in J$
- The objective function is:

$$\min \sum_{i \in I} c_{ij} x_{ij}$$

- The constraints are:
  - The assignment:

$$\sum_{j \in J} x_{ij} = 1$$

  - The capacity:

$$\sum_{j \in J} a_i x_{ij} \leq k_j$$

  - The variables must be binary:

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

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.

2.2 Linear programming modeling

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 determining 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.

Solution

1. • The sets are:
     - The set of months $T = \{1, 2, 3\}$.
   • The parameters are:
     - The production capacity of A b.
     - The production capacity of B $b^{'}$.
     - The unit production cost for A c.
     - The unit production cost for B $c^{'}$.
     - The inventory cost per unit and month m.
     - The sales forecast for month t, for $t \in T$ d_t.
   • The decision variables are:
     - The units produced by A in month t, $t \in T$ x_t.
     - The units bought from B in month t, for $t \in T$ x_t $x_t^{'}$.
     - The units in inventory at the end of month t, for $t \in T \cup \{0\}$ z_t.
   • The objective function is:

$$\min \sum_{t \in T} cx_t + c^{'}x_t^{'} + mz_t$$

- The constraints are:
  - The capacity of A:
$$x_t \leq b$$
  - The capacity of A:
$$x_t^{'} \leq b^{'}$$
  - The demand:
$$x_{t-1} + x_t + x_t^{'} \geq d_t$$
  - The inventory balance:
$$x_{t-1} + x_t + x_t^{'} - d_t = z_t t$$
  - The starting condition:
$$z_0 = 0$$
  - The non-negative variables:
$$x_t, x_t^{'}, z_t \geq 0$$

2. To take into account the minimum lot size, we add the binary variables y_t, that is 1 if production is active at month t, or 0 otherwise, for $t \in T$ and the constraints:

   - The minimum lot size:
$$x_t \geq l_{y_t}$$
   - The activation:
$$x_t \leq M_{y_t}$$

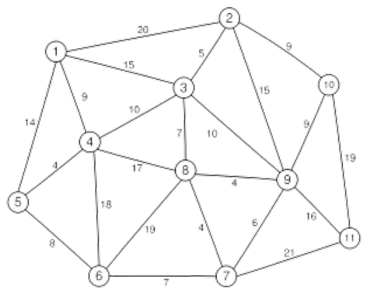
   where $l = 15$ is the minimum lot size, and M is a large enough value, such that constraint $x_t \leq M_{y_t}$ is redundant when $y_t = 1$. For instance, we can choose $M = 110$. Such constraints are usually called big-M constraints.

Chapter 3

Exercise session III

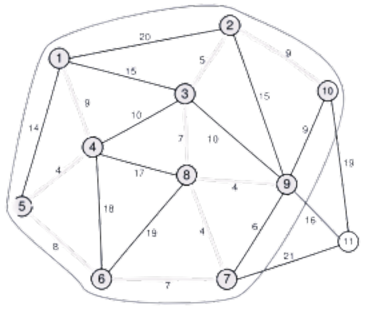
3.1 Minimum cost spanning tree

Find the minimum-cost spanning tree in the graph given in the figure by using Prim's algorithm, starting from the node three.



Solution

We apply Prim's algorithm, starting from node three and in the end we obtain:



3.2 Kruskal's algorithm

In 1956 Joseph Kruskal proposed the following greedy algorithm to find a minimum cost spanning tree in an arbitrary connected undirected graph $G = (N, E)$ with a cost c_e attached to each edge $e \in E$.

Algorithm 1 Kruskal's algorithm

```
 1: sort the edges of $E$ as $\{e_1, \ldots, e_m\}$ where $c_{e_1} \leq c_{e_2} \leq \cdots \leq c_{e_m}$
 2: $i \leftarrow 1$
 3: initialize the sub graph $G' = (N, E)$ of $G$ with $F = \varnothing$
 4: while $|F| < n - 1$ do
 5:     if the two endpoints of the edge $e_i$ belong to different connected
    components of the current sub graph $G^{'}$ then
 6:         $F \leftarrow F \cup \{e_i\}$
 7:         merge the two connected components
 8:     end if
 9:     $i \leftarrow i + 1$
10: end while
11: return the spanning tree $G^{'} = (N, E)$
```

In other words, we order the edges by increasing (non-decreasing) cost, we consider the edges in that order and, at each step, we select the current edge (which is one of the cheapest edges still available) only if it does not create a cycle with the previously selected edges. The algorithm terminates when $n - 1$ edges have been selected.

1. Describe an efficient way to identify/keep track of the connected components of the sub graph $G^{'}$ and to check that a new edge is creating a cycle with the previously selected edges (is connecting two distinct connected components of $G^{''}$). Determine the overall computational complexity of this implementation of Kruskal's algorithm.

2. By invoking the optimality condition for minimum-cost spanning trees, verify that Kruskal's algorithm is exact.

3. Find the maximum-cost spanning tree in the graph of the previous exercise by using a straightforward adaptation of Kruskal's algorithm.

Solution

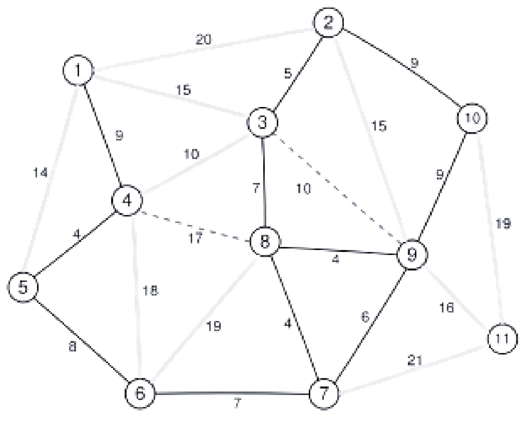
1. To identify/keep track of the connected components (subtrees) of the sub graph G, we use a vector v with as many components as vertices

in the graph, where v_i indicates the index of the connected component containing node i. At the beginning of the algorithm, we start with $v_i = i$, for $i = 1, \ldots, n$. When an edge $e = i, j$ is considered for addition to G'', we compare the values v_i and v_j. If $v_i \neq v_j$, then we can add the edge e to G' because it does not create a cycle. Since the two connected components of indices v_i and v_j are merged, the indices are updated as follows: in the vector v we substitute each occurrence of the index of node i with that of node j. If $v_i = v_j$, edge e is skipped because it would create a cycle.

2. The m edges can be ordered by non-decreasing cost in $O(m \log m)$, which is $O(m \log n)$ since $m \log m \leq m \log n^2 = 2m \log n$. At most m edges are considered for addition to the current sub graph G'. At each iteration, an edge $e = i, j$ is considered and the vector v is updated (in $O(n)$) only if $v_i \neq v_j$. Since a merging operation occurs exactly$n - 1$ times (a spanning tree contains $n - 1$ edges), the overall complexity is $O(m \log n + m + n^2) = O(mlogn + n^2)$.

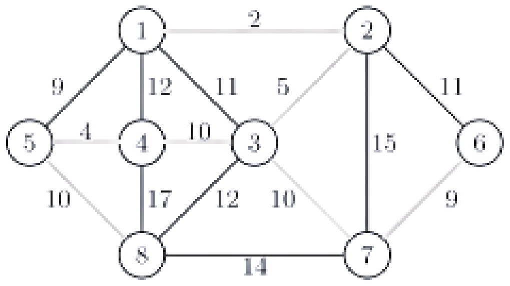
3. To verify that Kruskal's algorithm is exact, we just need to recall that the edges are considered in order of non-decreasing cost and to invoke the optimality condition for minimum cost spanning tree. Since each edge e that has been discarded (not added to F) has a cost c_e which is at least as large as the cost of all the previously selected edges, it is not a cost-decreasing edge. According to the optimality condition for minimum cost spanning trees, the resulting spanning tree is of minimum total cost because no cost-decreasing edge exists.

The execution of the algorithm gives the following result.



3.3 Optimality check

Without applying any one of Prim's and Kruskal's algorithms, verify whether the following spanning tree is of minimum cost.



Solution

It suffices to verify that there exists a cost decreasing edge. By inspection, we observe that, by adding edge $\{1, 5\}$ to the tree, the cycle $\langle 1, 5, 4, 3, 2, 1\rangle$ is introduced. In such cycle, edge $\{4, 3\}$ has a strictly larger cost than $\{1, 5\}$. Therefore, by removing edge $\{4, 3\}$ and adding edge $\{1, 5\}$, a spanning tree of strictly smaller total cost is obtained.

3.4 Compact storage of similar sequences

Consider the problem of storing a large set of strings. We assume that the strings have many similar entries (they differ only in a small number of positions) and we wish to store them in a compact way. This problem arises in several contexts such as when storing DNA sequences, where the characters correspond to the four DNA bases. In this exercise, we consider the simplified version of the problem with only two characters. Given a set of k sequences of M bits, we compute for each pair i, j, with $1 < i, j < k$, the Hamming distance between the sequences i and j, i.e., the number of bits that need to be flipped in sequence i to obtain sequence j. This function clearly satisfies the three usual properties of a distance: non-negativity, symmetry and triangle inequality. Consider the following set of 6 sequences and the corresponding matrix $D = d_{ij}$ of Hamming distances

```
1) 011100011101
2) 101101011001
3) 110100111001
4) 101001111101
5) 100100111101
6) 010101011100
```

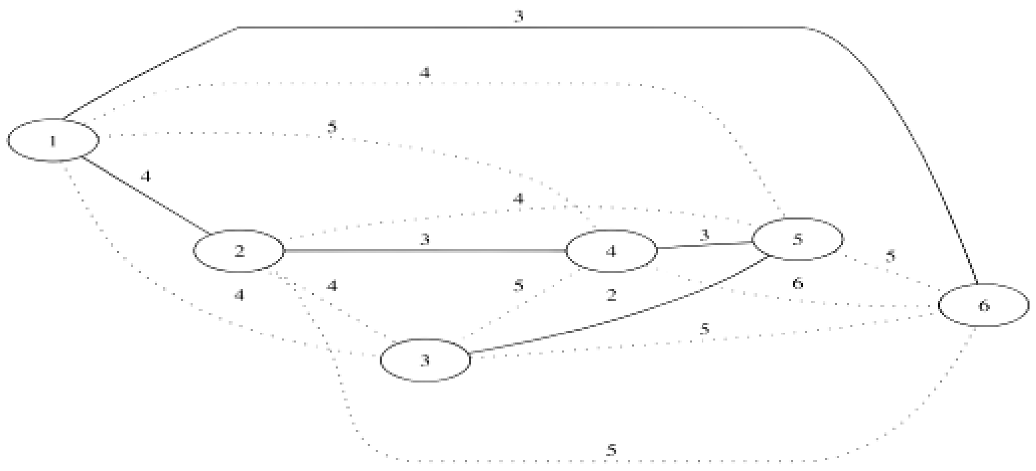
	1	2	3	4	5	6
1	0	4	4	5	4	3
2		0	4	3	4	5
3			0	5	2	5
4				0	3	6
5					0	5
6						0

where, due to symmetry, only the upper triangle of the matrix is shown. In order to exploit redundancies between sequences and to save memory, we can store: one of the sequences, called the reference sequence, completely and for every other sequence, only the set of bit flips that allow us to retrieve it either directly from the reference sequence or from another sequence.

Show how the problem of deciding which differences to memorize, to minimize the total number of bits used for storage, can be reduced to the problem of finding a minimum-cost spanning tree in an appropriate graph. Solve the problem for the given instance.

Solution

We construct a complete graph G with a node for each sequence and an edge for each pair of sequences. Moreover, each edge $\{i, j\}$ is assigned a cost $d_{ij} \log_2(M)$, which corresponds to the number of bits needed to store the indices of the positions in which the i-th and j-th sequences differ. The problem is then to look for a sub graph $G^{'}$ of G of minimum total cost. Since only one sequence is completely stored, $G^{'}$ must be connected to be able to retrieve any sequence. Since a sub graph of minimal cost is sought, $G^{'}$ will be acyclic. Therefore, a minimum cost spanning tree in G provides an optimal solution to the problem under consideration. The following minimum cost spanning tree has been found using Prim's algorithm.



Chapter 4

Exercise session IV

Chapter 5

Laboratory session I

Linear programming modeling

A canteen has to plan the composition of the meals that it provides. A meal can be composed of the types of food indicated in the following table. Costs, in Euro per hg, and availabilities, in hg, are also indicated.

Food	**Cost**	**Availability**
Bread	0.1	4
Milk	0.5	3
Eggs	0.12	1
Meat	0.9	2
Cake	1.3	2

A meal must contain at least the following amount of each nutrient:

Nutrient	Minimal quantity
Calories	600 cal
Proteins	50 g
Calcium	0.7 g

Each hg of each type of food contains to following amount of nutrients:

Food	**Calories**	**Proteins**	**Calcium**
Bread	30 cal	15 g	0.02 g
Milk	50 cal	15 g	0.15 g
Eggs	150 cal	30 g	0.05 g
Meat	180 cal	90 g	0.08 g
Cake	400 cal	70 g	0.01 g

Give a linear programming formulation for the problem of finding a meal of minimum total cost which satisfies the minimum nutrient requirements.

Solution

```
# Import the package mip
!pip install mip
import mip
# Food
I = {'Bread', 'Milk', 'Eggs', 'Meat', 'Cake'}
# Nutrients
J = {'Calories', 'Proteins', 'Calcium'}
# Cost in Euro per hg of food
c = {'Bread':0.1, 'Milk':0.5, 'Eggs':0.12, 'Meat':0.9, 'Cake':1.3}
# Availability per hg of food
q = {'Bread':4, 'Milk':3, 'Eggs':1, 'Meat':2, 'Cake':2}
# minum nutrients
b = {'Calories':600, 'Proteins':50, 'Calcium':0.7}
# Nutrients per hf of food
a = {  ('Bread','Calories'):30,
       ('Milk','Calories'):50,
       ('Eggs','Calories'):150,
       ('Meat','Calories'):180,
       ('Cake','Calories'):400,
       ('Bread','Proteins'):5,
       ('Milk','Proteins'):15,
       ('Eggs','Proteins'):30,
       ('Meat','Proteins'):90,
       ('Cake','Proteins'):70,
       ('Bread','Calcium'):0.02,
       ('Milk','Calcium'):0.15,
       ('Eggs','Calcium'):0.05,
       ('Meat','Calcium'):0.08,
       ('Cake','Calcium'):0.01}
# Define a empty model
model = mip.Model()
# Define variables
x = [model.add_var(name = i,lb=0) for i in I]
# Define the objective function
model.objective = mip.minimize(mip.xsum())
# Availability constraint
for i,food in enumerate(I):
```

```
model.add_constr()
# Minum nutrients constraint
for j in J:
model.add_constr(mip.xsum()>=)
# Optimizing command
model.optimize()
# Optimal objective function value
model.objective.x
# Printing the variables values
for i in model.vars:
print(i.name)
print(i.x)
```