

# DM545 – Linear and Integer Programming

## Answers to the Take-home Assignment, Winter 2025

---

In this assignment I will be using Tableau Prinitng function provided on github by Marco always including it as:

```
from util import tableau
```

I will also be using *numpy*, and *fractions* for matrix operations:

```
import numpy as np  
from fractions import Fraction
```

Throughout the Excercise a [website](#) that was showed during lectures will also be used as solver/visualizer

## Task 1

### Subtask 1.a

In the following problem:

$$\begin{aligned} \max \quad & 4x_1 + 5x_2 - 7x_3 \\ & -x_1 - x_2 + x_3 \leq 2, \\ & -5x_1 + 10x_3 \leq 10, \\ & x_1 \in [0, 5], \\ & x_2 \in [-1, 1], \\ & x_3 \in [-2, 2]. \end{aligned}$$

By inspection of the objective function we can deduct the potential values of variables:

$$\begin{aligned} 4x_1 \Rightarrow 4 \text{ is positive} &\Rightarrow x_1 \text{ make as large as possible,} \\ 5x_2 \Rightarrow 5 \text{ is positive} &\Rightarrow x_2 \text{ make as large as possible,} \\ -7x_3 \Rightarrow -7 \text{ is negative} &\Rightarrow x_3 \text{ make as small as possible.} \end{aligned}$$

Check potential solution formed by limits of interval bounds  $[x_1, x_2, x_3] = [5, 1, -2]$  :

$$\begin{aligned} \text{1st constraint: } & -x_1 - x_2 + x_3 \leq 2 \\ & -5 - 1 + (-2) \leq 2 \\ & -8 \leq 2 \quad \text{is feasible} \end{aligned}$$

$$\begin{aligned} \text{2nd constraint: } & -5x_1 + 10x_3 \leq 10 \\ & -25 + 10(-2) \leq 10 \\ & -45 \leq 10 \quad \text{is feasible} \end{aligned}$$

Objective function value:

$$\begin{aligned} 4x_1 + 5x_2 - 7x_3 &= \\ 20 + 5 + 14 &= \\ 39 & \end{aligned}$$

We conclude the optimal solution is  $[x_1, x_2, x_3] = [5, 1, -2]$

**Subtask 1.b**

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{s.t.} \quad & sx_1 + tx_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

We can choose (s) and (t) to get different cases:

**I) Single Optimal Solution**

$$s = 2, \quad t = 1$$

- The slope of the only constraint is different from the slope of the objective function.
- Geometrically, the feasible region intersects the objective function at a single vertex.
- The vertex is our optimal solution

**II) Infinite Optimal Solutions**

$$s = 1, \quad t = 1$$

- The constraint line is parallel to the objective function.
- Objective function “placed” on the face gives us infinitely many optimal solutions.

**III/IV) Infeasible / Unbounded**

- If either  $s$  or  $t$  (or both) are negative, the problem becomes unbounded as each variable balances the other in the constraint allowing the objective function to grow indefinitely.

$$s = -1, \quad t = 1 \text{ is an example of unbounded}$$

- For any  $s, t \geq 0$ , setting  $x_1 = \frac{1}{s}, x_2 = \frac{1}{t}$  gives a feasible solution (except if  $s = 0$  or  $t = 0$ , in which case the value of  $x_1$  or  $x_2$  does not matter as it's multiplied by 0).

It is **impossible** to set  $s, t$  such that the problem becomes *infeasible*, we considered all possible cases

## Task 2

### Subtask 2.a

First, we transform the problem into the standard form:

- Change a minimization into a maximization:  $\min(c^T x)$  into  $-\max -(c^T x)$ .
- Replace equalities with two inequalities.
- Convert all constraints to “ $\leq$ ”.

The result is:

$$\begin{aligned} -\max & -3x_1 - 2x_2 - 7x_3 \\ & -x_1 + x_2 \leq 10, \\ & x_1 - x_2 \leq -10, \\ & -2x_1 + x_2 - x_3 \leq -10. \end{aligned}$$

Then the tableau looks like:

```
T = np.array([[-1,1,0,1,0,0,0,10],
              [1,-1,0,0,1,0,0,-10],
              [-2,1,-1,0,0,1,0,-10],
              [-3,-2,-7,0,0,0,1,0]],dtype=object)
```

```
tableau(T)
```

	x1	x2	x3	x4	x5	x6	-z	b
	-1	1	0	1	0	0	0	10
	1	-1	0	0	1	0	0	-10
	-2	1	-1	0	0	1	0	-10
	-3	-2	-7	0	0	0	1	0

As we can see the tableau is **optimal** (no positive reduced costs) but **infeasible** (two of the  $b_i \leq 0$ ), we could apply Dual Simplex to work towards feasibility

**Subtask 2.b**

Tableau is given:

```
T = np.array([[0,0,0,1,1,0,0,0],
              [0,1,1,2,0,-1,0,30],
              [1,0,1,1,0,-1,0,20],
              [0,0,2,-7,0,5,1,-120]],dtype=object)

tableau(T)
```

	x1	x2	x3	x4	x5	x6	-z	b
	0	0	0	1	1	0	0	0
	0	1	1	2	0	-1	0	30
	1	0	1	1	0	-1	0	20
	0	0	2	-7	0	5	1	-120

It is worth noting that this tableau is *unbounded* as all coefficients of  $x_6$  in  $A$  are negative, but  $x_6$  has a positive reduced cost, however we can still technically perform a change of basis and see the results.

By largest coefficient  $x_6$  would have to enter (and  $x_3$  leave as it's constraint is tighter) and that would make the problem infeasible, and unoptimal

```
# III * -1
T[2] = T[2] * Fraction(-1, 1)

# II + III
T[1] = T[1] + T[2]

# IV - 5*III
T[3] = T[3] - 5*T[2]

tableau(T)
```

	x1	x2	x3	x4	x5	x6	-z	b
	0	0	0	1	1	0	0	0
	-1	1	0	1	0	0	0	10
	-1	0	-1	-1	0	1	0	-20
	5	0	7	-2	0	0	1	-20

By ratio test  $x_3$  enters (as  $x_6$  has only negative coefficients in  $A$ ) and  $x_1$  leaves as  $20/1 < 30/1$  (we ignore first line  $0/0 = ?$ ).

Coincidentally by Bland's Rule  $x_3$  enters and  $x_1$  leaves (we take the lowest index)

We can follow with both of them:

```
T = np.array([[0,0,0,1,1,0,0,0],
              [0,1,1,2,0,-1,0,30],
              [1,0,1,1,0,-1,0,20],
              [0,0,2,-7,0,5,1,-120]],dtype=object)

# II - III
T[1] = T[1] - T[2]

# IV - 2*III
T[3] = T[3] - 2*T[2]

tableau(T)
```

	x1	x2	x3	x4	x5	x6	-z	b
I	0	0	0	1	1	0	0	0
II	-1	1	0	1	0	0	0	10
III	1	0	1	1	0	-1	0	20
IV	-2	0	0	-9	0	7	1	-160

As we can see the tableau is still unbounded (by the case of  $x_6$ )

## Task 3

### Subtask 3.a

Tableau given:

```
T = np.array([[1,0,1,-1,0,5],
              [0,1,-2,3,0,15],
              [0,0,-2,-2,1,-110]],dtype=object)

tableau(T)
```

x1	x2	x3	x4	-z	b
1	0	1	-1	0	5
0	1	-2	3	0	15
0	0	-2	-2	1	-110

From the tableau, we observe the following:

- The solution is  $[x_1, x_2, x_3, x_4] = [5, 15, 0, 0]$  with objective value 110.
- The reduced costs are  $-2, -2$
- The values of dual variables are 2, 2 (negative reduced costs).
- The shadow prices are the same as the dual variables: 2, 2
- There is no over-capacity, as all the constraints are tight (no slacks in the basis).

## Task 4

### Subtask 4.a

We denote original problem as **P** and relaxed original problem as **PR**. Let's start by considering the original problem (with marked constraints by potential dual variables  $\alpha, \beta, \gamma$ )

$$\begin{aligned}
 \min \quad & \sum_{i=1}^n c_i y_i \\
 & \sum_{j=1}^m a_{ij} x_{ij} \leq b_i y_i, \quad i = 1, \dots, n \quad (\alpha) \\
 & \sum_{i=1}^n x_{ij} = 1, \quad j = 1, \dots, m \quad (\beta) \\
 & y_i \leq 1, \quad i = 1, \dots, n \quad (\gamma) \\
 & y_i \geq 0, \quad x_{ij} \geq 0.
 \end{aligned}$$

We denote the potential dual variables:

$$\begin{aligned}
 \alpha &= [\alpha_1, \dots, \alpha_n] \\
 \beta &= [\beta_1, \dots, \beta_m] \\
 \gamma &= [\gamma_1, \dots, \gamma_n]
 \end{aligned}$$

Then measure the violation of constraints (by putting everything to one side and multiplying by corresponding dual variable):

$$\begin{array}{lll}
 \alpha_1 (0 + b_1 y_1 - \sum_{j=1}^m a_{1j} x_{1j}) & \beta_1 (1 - \sum_{i=1}^n x_{i1}) & \gamma_1 (1 - y_1) \\
 \alpha_2 (0 + b_2 y_2 - \sum_{j=1}^m a_{2j} x_{2j}) & \beta_2 (1 - \sum_{i=1}^n x_{i2}) & \gamma_2 (1 - y_2) \\
 \vdots & \vdots & \vdots \\
 \alpha_n (0 + b_n y_n - \sum_{j=1}^m a_{nj} x_{nj}) & \beta_m (1 - \sum_{i=1}^n x_{im}) & \gamma_n (1 - y_n)
 \end{array}$$

Then we denote **PR** relaxed problem:

$$\text{PR}(\alpha, \beta, \gamma) = \min_{\text{by all } y, x \geq 0} \{ \cdot \} =$$



$$= \min_{\text{by all } y, x \geq 0} \left\{ \begin{array}{l} c_1 y_1 + \dots + c_n y_n + \\ \alpha_1 (b_1 y_1 - \sum_{j=1}^m a_{1j} x_{1j}) + \\ \alpha_2 (b_2 y_2 - \sum_{j=1}^m a_{2j} x_{2j}) + \\ \vdots \\ \alpha_n (b_n y_n - \sum_{j=1}^m a_{nj} x_{nj}) + \\ \beta_1 (1 - \sum_{i=1}^n x_{i1}) + \\ \beta_2 (1 - \sum_{i=1}^n x_{i2}) + \\ \vdots \\ \beta_m (1 - \sum_{i=1}^n x_{im}) + \\ \gamma_1 (1 - y_1) + \\ \gamma_2 (1 - y_2) + \\ \vdots \\ \gamma_n (1 - y_n) \end{array} \right\} = \min_{\text{by all } y, x \geq 0} \left\{ \begin{array}{l} y_1 (c_1 + \alpha_1 b - \gamma_1) + \\ y_2 (c_2 + \alpha_2 b - \gamma_2) + \\ \vdots \\ y_1 (c_n + \alpha_n b - \gamma_n) + \\ x_{11} (0 - \alpha_n a_1 - \beta_1) + \\ x_{21} (0 - \alpha_2 a_1 - \beta_2) + \\ \vdots \\ x_{n1} (0 - \alpha_n a_1 - \beta_n) + \\ x_{12} (0 - \alpha_1 a_2 - \beta_1) + \\ x_{22} (0 - \alpha_2 a_2 - \beta_2) + \\ \vdots \\ x_{nm} (0 - \alpha_n a_m - \beta_n) + \\ \sum_{j=1}^m \beta_j + \\ \sum_{i=1}^n \gamma_i \end{array} \right\}$$

To avoid useless lower bounds we set all parts multiplied by  $y, x \geq 0$ , (otherwise corresponding variable, for instance  $x_{i,j} - > \infty$  and  $\min_{\text{by all } y, x \geq 0} = -\infty$ )

That's how we get to the dual constraints:

$$\begin{array}{ll}
(c_1 + \alpha_1 b - \gamma_1) \geq 0 & \alpha_1 b - \gamma_1 \geq -c_1 \\
(c_2 + \alpha_2 b - \gamma_2) \geq 0 & \alpha_2 b - \gamma_2 \geq -c_2 \\
\vdots & \vdots \\
(c_n + \alpha_n b - \gamma_n) \geq 0 & -\alpha_n b - \gamma_n \geq -c_n \\
(0 - \alpha_n a_1 - \beta_1) \geq 0 & -\alpha_n a_1 - \beta_1 \geq 0 \\
(0 - \alpha_2 a_1 - \beta_2) \geq 0 & -\alpha_2 a_1 - \beta_2 \geq 0 \\
\vdots & \vdots \\
(0 - \alpha_n a_1 - \beta_n) \geq 0 & -\alpha_n a_1 - \beta_n \geq 0 \\
(0 - \alpha_1 a_2 - \beta_1) \geq 0 & -\alpha_1 a_2 - \beta_1 \geq 0 \\
(0 - \alpha_2 a_2 - \beta_2) \geq 0 & -\alpha_2 a_2 - \beta_2 \geq 0 \\
\vdots & \vdots \\
(0 - \alpha_n a_m - \beta_n) \geq 0 & -\alpha_n a_m - \beta_n \geq 0
\end{array} \Rightarrow$$

To get new Objective function we take the sums uncorrelated with  $y, x$  in front of the  $\min_{\text{by all } y, x} \{ \cdot \}$ . which are:  $\sum_{j=1}^m \beta_j + \sum_{i=1}^n \gamma_i$  Now we want to:

$$\max_{\alpha, \beta, \gamma} \left\{ \text{PR}(\alpha, \beta, \gamma) = \sum_{j=1}^m \beta_j + \sum_{i=1}^n \gamma_i + \min_{\text{by all } y, x} \{ \cdot \} \right\}.$$

However to keep the:

$$\text{opt}(\text{PR}(\alpha, \beta, \gamma)) \leq \text{opt}(P)$$

We need to penalize breaking the constraints.

That will cause the found  $\min_{\text{by all } y, x \geq 0}$  to not break any constraint.

We have to make it impossible to achieve minimum when breaking the constraint.

And to do that each dual variable must be chosen so that **violating a original constraint increases the value of the objective**.

For the constraints of type  $\alpha \Rightarrow \sum_{j=1}^m a_{ij}x_{ij} \leq b_i y_i$

- The violation measure is  $b_i y_i - \sum_{j=1}^m a_{ij}x_{ij}$ .
- When **broken**, this becomes **negative**.
- To penalize the objective it requires  $\alpha_i \leq 0$ .

For the constraints of type  $\gamma \Rightarrow y_i \leq 1$ :

- The violation measure is  $1 - y_i$ .
- When **breaking** this becomes **negative**.
- To ensure the penalty, we need  $\gamma_i \leq 0$ .

For the constraints of type  $\beta \Rightarrow \sum_{i=1}^n x_{ij} = 1$ :

- Violations can happen **in both ways**.
- Therefore we set:  $\beta_j \in \mathbb{R}$ .

This ensures that always:

$$\text{opt}(\text{PR}(\alpha, \beta, \gamma)) \leq \text{opt}(P)$$

Combinig everything togheter we are left with:

$$\begin{array}{ll}
\max \sum_{j=1}^m \beta_j + \sum_{i=1}^n \gamma_i & \\
\alpha_1 b - \gamma_1 \geq -c_1 & \\
\alpha_2 b - \gamma_2 \geq -c_2 & \\
\vdots & \\
-\alpha_n b - \gamma_n \geq 0 & \\
-\alpha_n a_1 - \beta_1 \geq 0 & \\
-\alpha_2 a_1 - \beta_2 \geq 0 & \\
\vdots & \\
-\alpha_n a_1 - \beta_n \geq 0 & \\
-\alpha_1 a_2 - \beta_1 \geq 0 & \\
-\alpha_2 a_2 - \beta_2 \geq 0 & \\
\vdots & \\
-\alpha_n a_m - \beta_n \geq 0 & \\
\alpha_i \leq 0. \quad i = 1, \dots, n & \\
\beta_j \in \mathbb{R}. \quad j = 1, \dots, m & \\
\gamma_i \leq 0. \quad i = 1, \dots, n &
\end{array}
\quad \Rightarrow \quad
\begin{array}{ll}
\max \sum_{j=1}^m \beta_j + \sum_{i=1}^n \gamma_i & \\
-b\alpha_i + \gamma_i \leq c_i \quad i = 1, \dots, n & \\
\alpha_i a_j + \beta_j \leq 0 \quad i = 1, \dots, n \quad j = 1, \dots, m & \\
\alpha_i \leq 0. \quad i = 1, \dots, n & \\
\beta_j \in \mathbb{R}. \quad j = 1, \dots, m & \\
\gamma_i \leq 0. \quad i = 1, \dots, n &
\end{array}$$

organized as

Q.E.D.

## Task 5

### Subtask 5.a

First write original problem with changed  $\min() \Rightarrow -\max -()$  (and with slacks):

$$c = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad A = \begin{bmatrix} -3 & 7 & 1 & 0 \\ 1 & -4 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} -21 \\ 4 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

So far  $B = 3, 4$  Let's prepare for Revised Simplex:

$$B = 1, 2$$

$$A_B = \begin{bmatrix} -3 & 7 \\ 1 & -4 \end{bmatrix}$$

$$A_B^{-1} = \frac{1}{\det(A_B)} \begin{bmatrix} -4 & -7 \\ -1 & -3 \end{bmatrix} = \begin{bmatrix} -\frac{4}{5} & -\frac{7}{5} \\ -\frac{1}{5} & -\frac{3}{5} \end{bmatrix}$$

(by using the formula for the inverse of 2x2 matrix, and  $\det()$  of 2x2 matrix)

$$A_N = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$c_B^T = [-1 \quad -1]$$

$$c_N^T = [0 \quad 0]$$

To check whether or not the solution with this basis is optimal we just need to compute  $c_n^T - c_B^T A_B^{-1} A_N$  (new reduced costs), which in our case can be easily done by hand:

$$\begin{aligned} [0 \quad 0] - [-1 \quad -1] \begin{bmatrix} -\frac{4}{5} & -\frac{7}{5} \\ -\frac{1}{5} & -\frac{3}{5} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} &= \\ &= [0 \quad 0] - \left[ \left( \frac{4}{5} + \frac{1}{5} \right) \quad \left( \frac{7}{5} + \frac{3}{5} \right) \right] = \\ &= [0 \quad 0] - [1 \quad 2] = \\ &= [-1 \quad -2] \end{aligned}$$

As we can see all reduced costs  $\leq 0$ , therefore solution is optimal

**Subtask 5.b**

Firstly we need to calculate the values  $x_1, x_2, x_3, x_4$  in the optimal solution this can be done by calculating  $A_B^{-1}b$ :

$$A_B^{-1}b = \begin{bmatrix} -\frac{4}{5} & -\frac{7}{5} \\ -\frac{1}{5} & -\frac{3}{5} \end{bmatrix} \begin{bmatrix} -21 \\ 4 \end{bmatrix} = \begin{bmatrix} \frac{56}{5} \\ \frac{9}{5} \end{bmatrix}, \quad \text{optimal basis was } B = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Therefore  $x_1 = 11\frac{1}{5}$  and  $x_2 = 1\frac{4}{5}$ . Let's calculate new non basic  $A$

$$A_{N_{new}} = A_B^{-1}A_N = \begin{bmatrix} -\frac{4}{5} & -\frac{7}{5} \\ -\frac{1}{5} & -\frac{3}{5} \end{bmatrix}$$

Both of the solution  $x_1, x_2$  fractional, we provide Gomory cuts for both:

row  $u = 1, 2$

$$\sum_{j \in N_{new}} (\bar{a}_{uj} - \lfloor \bar{a}_{uj} \rfloor) x_j \geq \bar{b}_u - \lfloor \bar{b}_u \rfloor$$

for convinience not that  $x_3 = s_1, x_4 = s_2$  - slacks

Gomory cut for  $u = 1$

$$(-\frac{4}{5} - \lfloor -\frac{4}{5} \rfloor) s_1 + (-\frac{9}{5} - \lfloor -\frac{9}{5} \rfloor) s_2 \geq \frac{56}{5} - \lfloor \frac{56}{5} \rfloor$$

$$\frac{1}{5} s_1 + \frac{3}{5} s_2 \geq \frac{1}{5}$$

$$s_1 + 3s_2 \geq 1$$

Gomory cut for  $u = 2$

$$(-\frac{1}{5} - \lfloor -\frac{1}{5} \rfloor) s_1 + (-\frac{3}{5} - \lfloor -\frac{3}{5} \rfloor) s_2 \geq \frac{9}{5} - \lfloor \frac{9}{5} \rfloor$$

$$\frac{4}{5} s_1 + \frac{2}{5} s_2 \geq \frac{4}{5}$$

$$4s_1 + 2s_2 \geq 4$$

We can express them in terms of the original variables by substitution, using the equalities from standard form:

$$-3x_1 + 7x_2 + s_1 = -21$$

$$x_1 - 4x_2 + s_2 = 4$$

$$s_1 = -21 + 3x_1 - 7x_2$$

$$s_2 = 4 - x_1 + 4x_2$$

Gomory cut for  $u = 1$ 

$$s_1 + 3s_2 \geq 1$$

$$\begin{aligned} & -21 + 3x_1 - 7x_2 \\ & + 12 - 3x_1 + 12x_2 \geq 1 \end{aligned}$$

$$-9 + 5x_2 \geq 1$$

$$x_2 \geq -2$$

Gomory cut for  $u = 2$ 

$$4s_1 + 2s_2 \geq 4$$

$$\begin{aligned} & -84 + 12x_1 - 28x_2 \\ & + 8 - 2x_1 + 8x_2 \geq 4 \end{aligned}$$

$$-76 + 10x_1 - 20x_2 \geq 4$$

$$x_1 - 2x_2 \geq 8$$

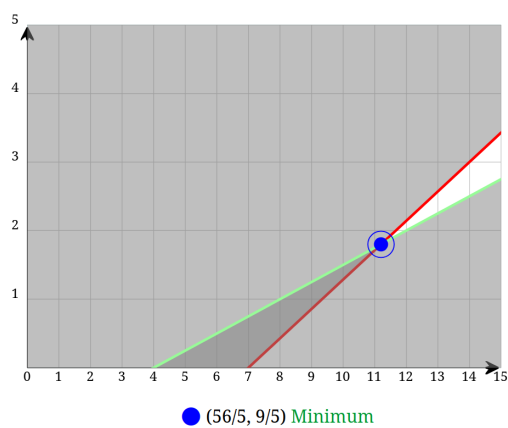
**Subtask 5.c**

[Website](#) was used as a visualizer

$x_2 \geq -2$  is not visible as it's weaker than base constraints  $x_1, x_2 \geq 0$

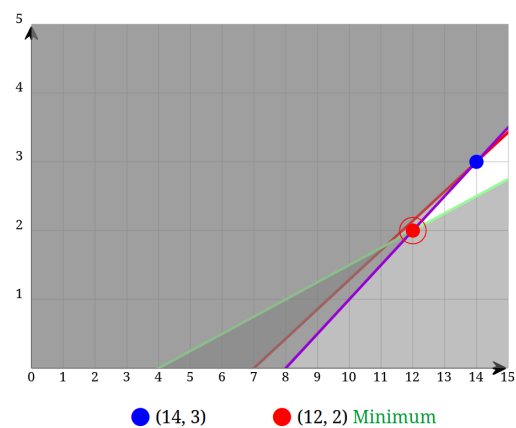
---

without Gomory cuts



---

with Gomory cuts

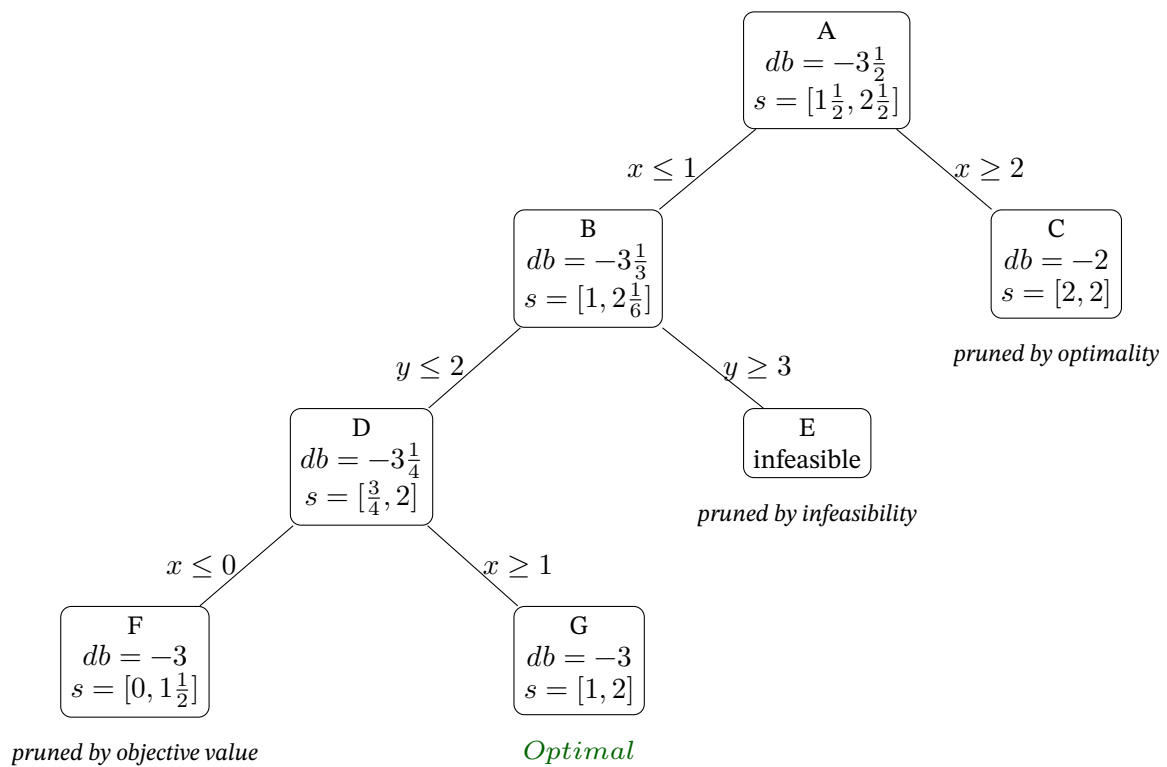


## Task 6

### Subtask 6.a

$$\begin{aligned}
 \min \quad & x - 2y \\
 \text{s.t.} \quad & -4x + 6y \leq 9 \\
 & x + y \leq 4 \\
 & x, y \in \mathbb{N}
 \end{aligned}$$

As the problem has only 2 variables [website](#) was used as a visualizer and solver.  
 Nodes were opened in alphabetical order (i.e.  $A \rightarrow B \rightarrow C \dots$ )





## Task 7

### Subtask 7.a

Firstly let us note *Hamming Distance*  $H_d$  between two words  $w_i, w_j$  of length  $d$  i.e.  $w_i, w_j \in \{0, 1\}^d$

$$H_d(w_i, w_j) = \sum_{k=1}^d \mathbf{1}(w_i[k] \neq w_j[k])$$

$k$  being the index of a word being checked and  $\mathbf{1}$  indicator function

Our problem can then be expressed as:

$$\max \left\{ \min \left\{ H_d(w_i, w_j) \right\} \right\} \text{ for } i, j \in \{1 \dots N\}$$

across all  $w_i \neq w_j$  of  $|w_i| = |w_j|$

We transform it into *Mathematical Programming form* by first denoting:

$$M_d = \min \left\{ H_d(w_i, w_j) \right\} \text{ for } i, j \in \{1 \dots N\}$$

And subsequently transforming the formulation into:

$$\begin{aligned} &\max M_d \\ &\sum_{k=1}^d \mathbf{1}(w_i[k] \neq w_j[k]) \geq M_d \\ &\text{for } 1 \leq i < j \leq N \end{aligned}$$

To make the constraints more workable:

$$\begin{aligned} &\max M_d \\ &\sum_{k=1}^d |w_i[k] - w_j[k]| \geq M_d \\ &\text{for } 1 \leq i < j \leq N \\ &\forall_{j,k,i} w_i[k], w_j[k] \in \{0, 1\} \end{aligned}$$

This formulation has exactly  $\frac{N(N-1)}{2}$  constraints (all unique pairs) and  $N \cdot d$  variables ( $N$  words of length  $d$ )

We would like however to get rid of not explicitly linear absolute value in the constraints. We can do that by introducing another variable that will replace it, let's call it  $\gamma$ :

$$\gamma_{ijk} \Leftarrow |w_i[k] - w_j[k]|$$

We have to bound it so that in all possible situations it is forced to hold a correct value:

$$\begin{aligned}\gamma_{ijk} &\leq 2 - (w_i[k] + w_j[k]) \\ \gamma_{ijk} &\leq w_i[k] + w_j[k] \\ \gamma_{ijk} &\geq w_i[k] - w_j[k] \\ \gamma_{ijk} &\geq w_j[k] - w_i[k]\end{aligned}$$

We see that using previously mentioned bounds we can replace absolute value in all cases:

$w_i[k], w_j[k] = (0, 0)$	$w_i[k], w_j[k] = (1, 1)$
$\gamma_{ijk} \leq 2$	$\gamma_{ijk} \leq 0$
$\gamma_{ijk} \leq 0$	$\gamma_{ijk} \leq 2$
$\gamma_{ijk} \geq 0$	$\gamma_{ijk} \geq 0$
$\gamma_{ijk} \geq 0$	$\gamma_{ijk} \geq 0$
$\Rightarrow \gamma_{ijk} = 0$	$\Rightarrow \gamma_{ijk} = 0$
$w_i[k], w_j[k] = (0, 1)$	$w_i[k], w_j[k] = (1, 0)$
$\gamma_{ijk} \leq 1$	$\gamma_{ijk} \leq 1$
$\gamma_{ijk} \leq 1$	$\gamma_{ijk} \leq 1$
$\gamma_{ijk} \geq -1$	$\gamma_{ijk} \geq 1$
$\gamma_{ijk} \geq 1$	$\gamma_{ijk} \geq -1$
$\Rightarrow \gamma_{ijk} = 1$	$\Rightarrow \gamma_{ijk} = 1$

Combinig that our new formulation looks as follows:

$$\begin{aligned}&\max M_d \\&\left\{ \sum_{k=1}^d \gamma_{ijk} \geq M_d \right\} \quad 1 \leq i < j \leq N \\&\left\{ \begin{array}{l} \gamma_{ijk} \leq 2 - (w_i[k] + w_j[k]) \\ \gamma_{ijk} \leq w_i[k] + w_j[k] \\ \gamma_{ijk} \geq w_i[k] - w_j[k] \\ \gamma_{ijk} \geq w_j[k] - w_i[k] \end{array} \right\} \quad \begin{array}{l} 1 \leq i < j \leq N \\ 1 \leq k \leq d \end{array} \\&\forall_{j,k,i} \gamma_{ijk} \in \mathbb{R} \\&\forall_{j,k,i} w_i[k], w_j[k] \in \{\mathbf{0}, \mathbf{1}\}\end{aligned}$$

That yields a total of  $(\frac{N(N-1)}{2} + 4d\frac{N(N-1)}{2})$  constraints and  $(N \cdot d + d\frac{N(N-1)}{2})$  variables  $(w_i[k] + \gamma_{ijk})$

**Subtask 7.b**

$$\max M_d$$

$$\begin{aligned} \sum_{k=1}^d \gamma_{ijk} &\geq M_d & 1 \leq i < j \leq N \\ \gamma_{ijk} &\leq 2 - (w_i[k] + w_j[k]) & 1 \leq i < j \leq N \\ \gamma_{ijk} &\leq w_i[k] + w_j[k] \\ \gamma_{ijk} &\geq w_i[k] - w_j[k] & 1 \leq k \leq d \\ \gamma_{ijk} &\geq w_j[k] - w_i[k] \\ \forall_{j,k,i} \gamma_{ijk} &\in \mathbb{R} \\ \forall_{j,k,i} w_i[k], w_j[k] &\in \{0, 1\} \end{aligned}$$

In the final formulation the problem belongs to the **Mixed Integer Linear Programming** family as its constraints are all linear and some of the variables ( $\gamma$ ) can be real, the other ( $w_i[k]$ ) integer. It's worth noting however that all integer variables here are **binary** therefore it's more of a *Mixed Binary Linear Programming*, which *might* be a bit easier to solve than regular *Integer* programming.

## Task 8

### Subtask 8.a

Let's first denote:

- $j \in N$  = set of available assets to invest in
- $t \in T$  = set of all months where return data is given
- $r_{jt}$  = value of historical return on investment  $j$  from month  $t$  to month  $t + 1$
- $\epsilon$  = parameter determining minimum reward
- $x_j$  = variable representing fraction of total resources put into asset  $j$
- $\tau_t$  = variable that will replace the absolute value in objective function

Then our variables used in model will be:

$$\hat{R}_j = \frac{1}{T} \sum_{t=1}^T r_{jt}$$

$$\alpha = \max \left( 0, \min_j \hat{R}_j \right)$$

$$\beta = \max_j \hat{R}_j$$

$$B = \alpha + \epsilon(\beta - \alpha) =$$

= minimal reward

$$\hat{R} = \sum_{j=1}^N x_j R_j =$$

= reward

$$\widehat{\text{MAD}} = \frac{1}{T} \sum_{t=1}^T \left[ \left| \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \right| \right]$$

↓ (not a formal equality, just notation)

$$= \frac{1}{T} \sum_{t=1}^T \tau_t$$

= risk

$$\tau_t \approx \left| \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \right|$$

= auxiliary variable to remove  
absolute value from the formulation

We want to replace the absolute value with something more linear, let's introduce a set of additional variables responsible for that,  $\tau_t$ , and 2 successive inequalities for each, to make up for the possible negativity before applying absolute value. Assuming that all the fractions have to sum up to 1, i.e. we *have to* invest all of our resources, the model is as follows:

$$\begin{aligned} \min \quad & \frac{1}{T} \sum_{t=1}^T \tau_t \\ & \sum_{j=1}^N x_j \hat{R}_j \geq \alpha + \epsilon(\beta - \alpha), \\ & \sum_{j=1}^N x_j = 1, \\ & \tau_t \geq \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \quad t = 1, \dots, T \\ & \tau_t \geq - \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \quad t = 1, \dots, T \\ & \forall_j x_j \geq 0 \end{aligned}$$

Explained line by line:

- minimization of risk denoted by  $\widehat{MAD}$
- constraint on the reward  $\geq$  minimal reward  $B$
- constraint on full investment
- 2 sets of constraints replacing absolute value, There is combined  $2T$  of them, 2 constraints for each  $\tau_t$
- general bounds on  $x_j$

**Subtask 8.b**

Model:

$$\min \quad \frac{1}{T} \sum_{t=1}^T \tau_t$$

$$\sum_{j=1}^N x_j \hat{R}_j \geq \alpha + \epsilon(\beta - \alpha),$$

$$\sum_{j=1}^N x_j = 1,$$

$$\tau_t \geq \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \quad t = 1, \dots, T$$

$$\tau_t \geq - \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \quad t = 1, \dots, T$$

$$\forall_j x_j \geq 0$$

$$\forall_t \tau_t \in \mathbb{R}$$

Has exactly  $N + T$  variables and  $2T + 2$  (or  $2T + N + 2$  if we include  $x_j \geq 0$ ) constraints

**Subtask 8.c**

As gurobi was available only with license I used **mip** solver that is free. The solution is based on provided template, adjusted to use **mip** and my model:

```
from mip import Model, xsum, CONTINUOUS, MINIMIZE
import matplotlib.pyplot as plt
import sys
import numpy as np

SEE_DETAILED_ALLOCATIONS = False

class Data:
    def __init__(self, filename):
        with open(filename, "r") as filehandle:
            lines=filehandle.readlines()

        self.N = int(lines[0].strip("\r\n"))
        self.T = int(lines[1].strip("\r\n"))
        self.r = np.zeros([self.N, self.T])

        for line in lines[2:]:
            line=line.strip("\r\n");
            parts=line.split(" ");
            j=int(parts[0])-1
            t=int(parts[1])-1
            val=float(parts[2])
            self.r[j,t]=val

        self.mean_r = np.mean(self.r, axis=1)
        self.min_r = np.max([0, np.min(self.mean_r)])
        self.max_r = np.max(self.mean_r)

        print("min_r:", self.min_r, "max_r:", self.max_r)

def solve(data, epsilon):
    m = Model("portfolio", sense=MINIMIZE)

    N = data.N
    T = data.T
    R_hat = data.mean_r
    r = data.r

    # The Minimum Return
    B = data.min_r + (epsilon / 100.0) * (data.max_r - data.min_r)

    # Decision variables
    x = [m.add_var(var_type=CONTINUOUS, lb=0) for _ in range(N)]
    tau = [m.add_var(var_type=CONTINUOUS) for _ in range(T)]
```

```

# Objective function
m.objective = (1 / T) * xsum(tau[t] for t in range(T))

# Minimum Return
m += xsum(R_hat[j] * x[j] for j in range(N)) >= B

# All money
m += xsum(x[j] for j in range(N)) == 1

# Tau constraints
for t in range(T):
    expr = xsum(x[j] * (r[j, t] - R_hat[j]) for j in range(N))
    m += (tau[t] >= expr)
    m += (tau[t] >= -expr)

status = m.optimize()

if status is None:
    return B, 0, None

x_values = [x[j].x for j in range(N)]
return B, m.objective_value, x_values

def main(argv):
    if len(argv) != 1:
        usage()
    instance = Data(argv[0])
    portfolio_info = []

    epsilons = np.arange(0, 101, 10)
    rewards = np.zeros_like(epsilons, dtype=np.float64)
    risks = np.zeros_like(epsilons, dtype=np.float64)

    for i, epsilon in enumerate(epsilons):
        print(f"=== Solve for epsilon = {epsilon}")
        B_val, obj_val, x_values = solve(instance, epsilon)

        rewards[i] = B_val
        risks[i] = obj_val
        positive_x = [(j+1, x) for j, x in enumerate(x_values) if x > 0]
        if not SEE_DETAILED_ALLOCATIONS:
            positive_x = len(positive_x)

    portfolio_info.append({
        "epsilon": epsilon,
        "reward": B_val,
        "risk": obj_val,
        "allocations": positive_x
    })

```



```

    })

    plt.figure(figsize=(10,6))
    plt.plot(rewards, risks, '-o')
    plt.xlabel("Min Reward")
    plt.ylabel("Associated Min Risk")
    plt.title("Portfolio")
    plt.grid(True)

    plt.figure(figsize=(6, 10))
    plt.axis("off")
    text_str = ""
    for info in portfolio_info:
        epsilon = info['epsilon']
        positive_x = info['allocations']
        if(SEE_DETAILED_ALLOCATIONS):
            alloc_str = ", ".join([f"x_{j} = {x:.4f}" for j, x in positive_x])
        else:
            alloc_str = f"different assets: {positive_x}"
        text_str += f"\n = {epsilon}: \n{alloc_str}\n"

    plt.text(0, 1, text_str, fontsize=12, va='top', ha='left', wrap=True)

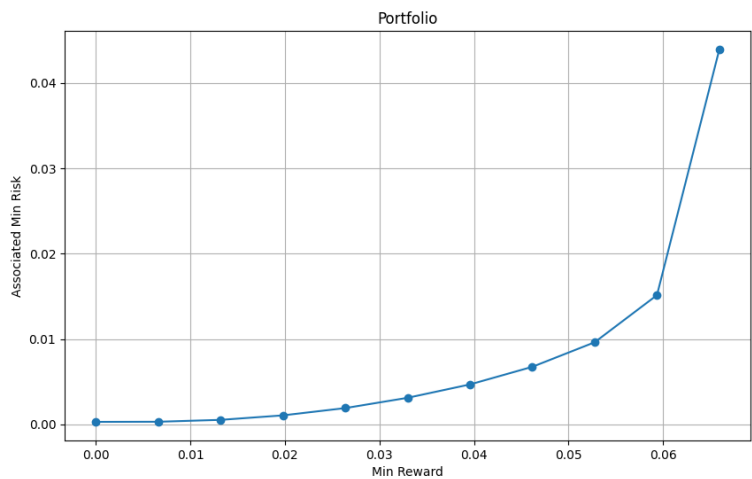
    plt.tight_layout()
    plt.show()

def usage():
    print("Reads data from datafilename")
    print("Usage: [\"datafilename\"]\n")
    raise SystemExit

if __name__ == "__main__":
    main(sys.argv[1:])

```

Risk/Reward plot



$\epsilon = 0$ :  
different assets: 244

$\epsilon = 10$ :  
different assets: 237

$\epsilon = 20$ :  
different assets: 228

$\epsilon = 30$ :  
different assets: 188

$\epsilon = 40$ :  
different assets: 143

$\epsilon = 50$ :  
different assets: 111

$\epsilon = 60$ :  
different assets: 72

$\epsilon = 70$ :  
different assets: 50

$\epsilon = 80$ :  
different assets: 27

$\epsilon = 90$ :  
different assets: 14

$\epsilon = 100$ :  
different assets: 1

**Subtask 8.d**

**Worst-period return** objective:

$$\max \min_{t=1, \dots, T} \sum_{j=1}^n x_j r_{jt}$$

We can try to transform it:

$$\begin{aligned} &\min \gamma \\ &\min_{t \in T} \left\{ \sum_{j=1}^n x_j r_{jt} \right\} \leq \gamma \end{aligned}$$

This formulation should work (because we need to make  $\gamma$  only bigger than the smallest one)

We can try to remove the min part again by introducing another variable  $\omega$ . We make  $\omega$  negative in the objective to penalize it being less than minimum across all  $T$

$$\begin{aligned} &\min \gamma - \omega \\ &\gamma \geq \omega \\ &\omega \leq \sum_{j=1}^n x_j r_{jt} \quad t = 1, \dots, T \end{aligned}$$

Therefore *Worst-period return* could be used, but the objective function would have to be corrected by the value of  $\omega$  and in general this formulation is harder to work with.

**Mean-variance:**

$$\hat{\sigma}^2 = \sum_i \sum_j x_i x_j \hat{\sigma}_i \hat{\sigma}_j \rho_{ij}$$

Cannot be used in a linear programming problem, as it involves multiplication of decision variables  $x_i x_j$ . This would make it a quadratic problem.

**Subtask 8.e**

We can model this feature by adding additional variable  $\theta_j \in \{0, 1\}$  indicator-type for  $x_j$ . We know that upper bound of each  $x_j$  is 1 (you cannot spend more than 100% of your resources),  $\theta_j$  can be used by introducing a new constraint for each  $x_j$ :

$$\begin{aligned}
 \min \quad & \frac{1}{T} \sum_{t=1}^T \tau_t \\
 & \sum_{j=1}^N x_j \hat{R}_j \geq \alpha + \epsilon(\beta - \alpha), \\
 & \sum_{j=1}^N x_j = 1, \\
 & \sum_{j=1}^N \theta_j \leq M, \\
 & \tau_t \geq \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \quad t = 1, \dots, T \\
 & \tau_t \geq - \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \quad t = 1, \dots, T \\
 & x_j \leq \theta_j \quad j = 1, \dots, N \\
 & \forall_j x_j \geq 0 \\
 & \forall_j \theta_j \in \{0, 1\} \\
 & \forall_t \tau_t \in \mathbb{R}
 \end{aligned}$$

Here for each  $x_j$ ,  $\theta_j$  can be called a *choosing* variable for asset  $j$ :

- *Not Chosen* :  $\theta_j = 0$  makes corresponding  $x_j = 0$
- *Chosen* :  $\theta_j = 1$  makes corresponding  $x_j \in [0, 1]$

Sum of all  $\theta_j$  ensures that amount of different assets bought is  $\leq M$

**Subtask 8.f**

We can reuse model from previous task:

$$\min \quad \frac{1}{T} \sum_{t=1}^T \tau_t$$

$$\sum_{j=1}^N x_j \hat{R}_j \geq \alpha + \epsilon(\beta - \alpha),$$

$$\sum_{j=1}^N x_j = 1,$$

$$\sum_{j=1}^N \theta_j \leq M,$$

$$\tau_t \geq \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \quad t = 1, \dots, T$$

$$\tau_t \geq - \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \quad t = 1, \dots, T$$

$$x_j \leq \theta_j \quad j = 1, \dots, N$$

$$\forall_j x_j \geq 0$$

$$\forall_j \theta_j \in \{\mathbf{0}, \mathbf{1}\}$$

$$\forall_t \tau_t \in \mathbb{R}$$

By removing maximum different assets constraint and adding a new one with corresponding variable  $v$  that will be responsible for minimum  $x$  value. We can do that by adding another constraint that, when corresponding  $\theta_j = 1$  will enforce  $x_j \in [v, 1]$

$$\min \quad \frac{1}{T} \sum_{t=1}^T \tau_t$$

$$\sum_{j=1}^N x_j \hat{R}_j \geq \alpha + \epsilon(\beta - \alpha),$$

$$\sum_{j=1}^N x_j = 1,$$

$$\tau_t \geq \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \quad t = 1, \dots, T$$

$$\tau_t \geq - \sum_{j=1}^N x_j (r_{jt} - \hat{R}_j) \quad t = 1, \dots, T$$

$$x_j \leq \theta_j \quad j = 1, \dots, N$$

$$\theta_j v \leq x_j \quad j = 1, \dots, N$$

$$\forall_j x_j \geq 0$$

$$\forall_j \theta_j \in \{\mathbf{0}, \mathbf{1}\}$$

$$\forall_t \tau_t \in \mathbb{R}$$

This ensures that all  $x$  stay within set bounds  $0 \cup [v, 1]$ .