Università della Svizzera italiana
Institute of Computing CI

Numerical Computing **2024**

Student: Matteo Ghilardini Discussed with: Sasha Toscano

Solution for Project 5 **Due date:** Wednesday, December 11, 2024, 11:59 PM

1. Graphical solution of linear programming problems [20 points]

To solve problem (2), I start by extracting all the variables:

- Production cost: $prod_{t1} = 25$ and $prod_{t2} = 40$ CHF
- Sold price: $sold_{t1} = 85$ and $sold_{t2} = 110$ CHF
- Profit: $profit_{t1} = 85 - 25 = 60$ and $profit_{t2} = 110 - 40 = 70$ CHF
- Total monthly demand: $md = 265$
- Max production expense: 7000 CHF

From that, we can extract the following constrains considering that $t_1 = x$ and $t_2 = y$ to match the standard notation:

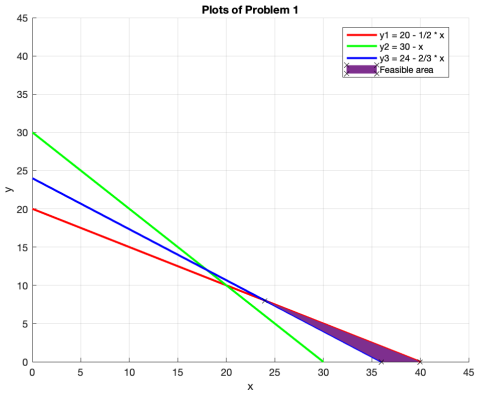
- Total production cost must not exceed 7000 CHF: $25x + 40y \leq 7000$
- Total production must not exceed the monthly demand: $x + y \leq 265$
- Production can't be negative: $x, y \geq 0$

Since the objective is to maximize the net profit, the function can be written as: $z = 60x + 70y$

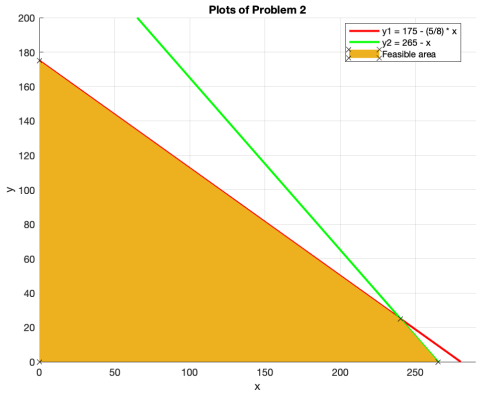
At this point we have problem (2) as a linear programming problem:

$$\begin{array}{ll} \max & z = 60x + 70y \\ \text{s.t.} & 25x + 40y \leq 7000 \\ & x + y \leq 265 \\ & x, y \geq 0 \end{array}$$

- **Solve the system of inequalities:**
  1. $x + 2y \leq 40 \rightarrow 2y \leq 40 - x \rightarrow y \leq 20 - \frac{1}{2}x$
     $x + y \geq 30 \rightarrow y \geq 30 - x$
     $2x + 3y \geq 72 \rightarrow 3y \geq 72 - 2x \rightarrow y \geq 24 - \frac{2}{3}x$
     $x, y \geq 0$
  2. $25x + 40y \leq 7000 \rightarrow 40y \leq 7000 - 25x \rightarrow y \leq 175 - \frac{5}{8}x$
     $x + y \leq 265 \rightarrow y \leq 265 - x$
     $x, y \geq 0$
- **Plot the feasible region identified by the constraints:**



(a) Problem 1



(b) Problem 2

The code used to generate the plot of problem 1 (figure 1a) and problem 2 (figure 1b) is in the Matlab script `ex1.m`.

In both cases, another constraint that is not shown directly with a function is $x, y \geq 0$, since this constraint is given by the x and y axes.

The intersection points of the constraint inequalities of the two problems have been computed and verified using `GeoGebra` since is very simple to draw inequalities and compute the `AND` between all of them to find the **feasible area**.

- **Find the optimal solution and the value of the objective function at that point:** We know that the optimal solution belongs to the **feasible basis**, and therefore must be one of the vertex of the **feasible area**;
  - **Problem 1**:
    The vertex of the **Feasible Area** are $v_1 = (24, 8), v_2 = (36, 0), v_3 = (40, 0)$.
    So we insert this values in the **Objective function** $z = 4x + y$ and since we want to minimize the function, we look for the minimal value of z:
    * $z_1 = 4 * 24 + 8 = 104$
    * $z_2 = 4 * 36 + 0 = 144$
    * $z_3 = 4 * 40 + 0 = 160$

    So we see that the **Objective function** is minimized with $v_1 = (24, 8)$, with $z = 104$.
  - **Problem 2**:
    The vertex of the **Feasible Area** are $v_1 = (0, 175), v_2 = (240, 25), v_3 = (265, 0), v_4 = (0, 0)$.
    So we insert this values in the **Objective function** $z = 60x + 70y$ and since we want to maximize the function, we look for the maximal value of z:
    * $z_1 = 60 * 0 + 70 * 175 = 12'250$
    * $z_2 = 60 * 240 + 70 * 25 = 16'150$
    * $z_3 = 60 * 265 + 70 * 0 = 15'900$
    * $z_4 = 60 * 0 + 70 * 0 = 0$

    So we see that the **Objective function** is maximized with $v_2 = (240, 25)$, with $z = 16'150$.

2. Implementation of the simplex method [30 points]

- Implementation of `standardize.m`:
  Since we need that $h \geq 0$, if we have some `h(i)` that is negative, we flip the sign of it, but also the sign of the corresponding line in A and aug_matrix.

  The augmented matrix `A_aug` is constructed by appending A with aug_matrix if is a maximization, or with $-aug\_matrix$ if is a minimization.

- Implementation of `simplexSolve.m`:
  I compute the reduced cost coefficients as `r_D = c_D - c_B * BiD` according to the formula:

  $$r_D = c_D - c_B B^{-1} D$$

  Compute the optimal check as the sum of all r_D that are less-equal than 0 (for maximization) or as the sum of all r_D that are greater-equal than 0 (for minimization).

  In the case of maximization, the entering variable index is the one with the largest reducer cost (`[~, idxIN] = max(r_D)`). Otherwise, in the case of minimization, the entering variable index is the one with the smallest reducer cost (`[~, idxIN] = min(r_D)`).

  The coefficients ratio for the column corresponding to the entering variable is computed according to the formula:

  $$ratio = \frac{B^{-1}h}{B^{-1}D} = \frac{B^{-1}h}{B^{-1}D(:, idxIN)}$$

  In order to avoid an eventual division by 0, i also add `eps` to the denominator.

  The smallest positive ratio, i.e. the minimal between all the ratio, is computed as `idxOUT = find(ratio == min(ratio(ratio >= 0)))`.

  At this point, all the IN and OUT variables are exchanged in B, D, c_B adn c_D.

  Now is computed again the reduced cost coefficients using the same formula as before but using the yet updated matrices.

  At this point, the optimality condition is computed as the sum of all reduced cost coefficients less-equal to 0 for a maximization, or greater-equal to 0 for minimization.

  As last operation in the recursion, the new x_B is computed as `x_B = Bih - BiD * x_D` according to the computation of the basic solution using the basis matrix.

- By running the provided script `testSimplex.m` the the following results are obtained:

Variable	Value
x_1	90.000000
x_2	130.000000
s_1	50.000000

(a) Test 1: Optimal $z = 3600$

Variable	Value
x_1	96.000000
x_2	18.000000

(b) Test 2: Optimal $z = 1062$

Variable	Value
x_1	4.000000
x_3	1.000000
s_3	2.000000

(c) Test 3: Optimal $z = 17$

Variable	Value
x_1	4.000000
x_2	10.000000
s_1	20.000000
s_3	20.000000
s_4	12.000000
s_5	17.000000

(d) Test 4: Optimal $z = 380000$

Variable	Value
x_2	560.000000
x_3	320.000000
s_3	40.000000

(e) Test 5: Optimal $z = 20160$

Variable	Value
x_1	4400.000000
x_2	400.000000
x_6	4000.000000
x_8	2100.000000
x_9	200.000000

(f) Test 6: Optimal $z = 79500$

Accordingly with the output of script `testSimplex.m`, all this tests provide the expected result.

3. Applications to a real-life example: Cargo aircraft [25 points]

1. Formulate the problem above as a linear program

Taking into account $x_{C,S}$ as the number of tons x that is loaded into cargo C, compartment S, the profits of each compartment S can be calculated as

$$P_S = (135x_{C,S} + 200x_{C,S} + 410xC, S + 520x_{C,S}) * (100 + bonus)\%$$

, so we have:

- $P_{S_1}(no\_bonus) = 135x_{1,1} + 200x_{2,1} + 410x3, 1 + 520x_{4,1}$
- $P_{S_2}(10\%) = (135x_{1,2} + 200x_{2,2} + 410x3, 2 + 520x_{4,2}) * (100 + 10)\%$
- $P_{S_3}(20\%) = (135x_{1,3} + 200x_{2,3} + 410x3, 3 + 520x_{4,3}) * (100 + 20)\%$
- $P_{S_4}(30\%) = (135x_{1,4} + 200x_{2,4} + 410x3, 4 + 520x_{4,4}) * (100 + 30)\%$

Since our objective is to maximize the profit, the **Objective function** can be written as:

$$(max)\ z = P_{S_1} + P_{S_2} + P_{S_3} + P_{S_4}$$

After this, we have to consider the weight and volume constrains of the problem, since each compartment S has a weight and volume capacity that can't be exceeded:

- **Weight constrains:**
  - $W_{S_1} = 16x_{1,1} + 32x_{2,1} + 40x_{3,1} + 28x_{4,1} \leq 18$
  - $W_{S_2} = 16x_{1,2} + 32x_{2,2} + 40x_{3,2} + 28x_{4,2} \leq 32$
  - $W_{S_3} = 16x_{1,3} + 32x_{2,3} + 40x_{3,3} + 28x_{4,3} \leq 25$
  - $W_{S_4} = 16x_{1,4} + 32x_{2,4} + 40x_{3,4} + 28x_{4,4} \leq 17$
- **Volume constrains:**
  - $V_{S_1} = 320x_{1,1} + 500x_{2,1} + 630x_{3,1} + 125x_{4,1} \leq 11830$
  - $V_{S_2} = 320x_{1,2} + 500x_{2,2} + 630x_{3,2} + 125x_{4,2} \leq 22552$
  - $V_{S_3} = 320x_{1,3} + 500x_{2,3} + 630x_{3,3} + 125x_{4,3} \leq 11209$
  - $V_{S_4} = 320x_{1,4} + 500x_{2,4} + 630x_{3,4} + 125x_{4,4} \leq 5870$

After all, there is also the **non-negativity** constrain since we are dealing with quantities and it is not possible to have negative quantities of tons. This constraint is simply defined as the fact that any ton of cargo placed in any compartment must be greater than or equal to 0, and this can be expressed with the following formula:

$$x_{C,S} \geq 0\ ;\ \forall\ C, S \in \{1, 2, 3, 4\}$$

.

So at the end, the **Linear problem** can be expressed as:

$$\begin{aligned}
\max \quad & z = P_{S_1} + P_{S_2} + P_{S_3} + P_{S_4} \\
\text{s.t.} \quad & 16x_{1,1} + 32x_{2,1} + 40x_{3,1} + 28x_{4,1} \leq 18 \\
& 16x_{1,2} + 32x_{2,2} + 40x_{3,2} + 28x_{4,2} \leq 32 \\
& 16x_{1,3} + 32x_{2,3} + 40x_{3,3} + 28x_{4,3} \leq 25 \\
& 16x_{1,4} + 32x_{2,4} + 40x_{3,4} + 28x_{4,4} \leq 17 \\
& 320x_{1,1} + 500x_{2,1} + 630x_{3,1} + 125x_{4,1} \leq 11830 \\
& 320x_{1,2} + 500x_{2,2} + 630x_{3,2} + 125x_{4,2} \leq 22552 \\
& 320x_{1,3} + 500x_{2,3} + 630x_{3,3} + 125x_{4,3} \leq 11209 \\
& 320x_{1,4} + 500x_{2,4} + 630x_{3,4} + 125x_{4,4} \leq 5870 \\
& x_{C,S} \geq 0\ ;\ \forall\ C, S \in \{1, 2, 3, 4\}
\end{aligned}$$

2. Create a script exercise2.m which uses the simplex method implemented in the previous exercise to solve the problem

That said, the code implementation for solving the linear problem is in the file `exercise2.m` and follows the following key points:

- **Define cargo constrains:**
  each cargo type has some weights (`cargo_weights = [16, 32, 40, 28]`) and each compartment has some max-weight (`max_weights = [18, 32, 25, 17]`). Besides this, each cargo type has a volume (`cargo_volumes = [320, 500, 630, 125]`) and each compartment has a max-volume (`max_volumes = [11830, 22552, 11209, 5870]`).

- **Construct constrains matrices:**
  In this process, I use the `kron` product to construct the constraint matrices for weight, volume, and availability. The `kron` function allows us to create block matrices that replicate a set of coefficients for each cargo type across all compartments. Specifically, it generates a matrix where each cargo type's constraint (weight, volume, availability) is applied to each compartment. After it, all this "individual" matrices are combined into a unique constrains matrix (parameter `A` in function `simplex(...)`).

- **Build the RHS vector for constrains inequality:**
  this vector `rhs_constraints = [max_weights, max_volumes, cargo_weights]'` contains the coefficients of the constraints inequalities and also the cargo weights for each type. The vector is transposed in order to match the expected format in the linear programming solver.

- **Compute the profit and profit multiplier:**
  After setted the cargo profits for each compartment as `cargo_profits = [135, 200, 410, 520]` compute the actual profit as `bonus_multipliers = kron(cargo_profits, [1, 1.1, 1.2, 1.3])` resulting in a matrix with the profit values adjusted with the corresponding multiplier (depending of the compartment).

- **Constrain sign:**
  the constraint sign is computed as `constraint_sign = -ones(1, size(constraint_matrix, 1))` corresponding to a vector of -1 (meaning sign $\leq$) of the same size as the number of rows in the matrix.

- **Call `simplex(...)`** and solve the optimization problem using simplex method.

- **Plot the results:**
  In order to plot the distribution of the cargo across the different compartments I've just written the code that populates the `distribution_matrix` based on the basic solution and indices (computed by `simplex(...)  function`. Since the actual compartment indexes are between 1 and 4, i use the `mod(...)` function to retrieve the actual compartment index across the indexes computed by `simplex(...)`.

The results of the implementation are the following:

idx	*Cargo*	*Compartment*	Tons
x5	*2*	*1*	18
x6	*2*	*2*	6
x10	*3*	*2*	26
x11	*3*	*3*	14
x15	*4*	*3*	11
x16	*4*	*4*	17

Table 1: Optimal value of `z = 41'890 CHF`

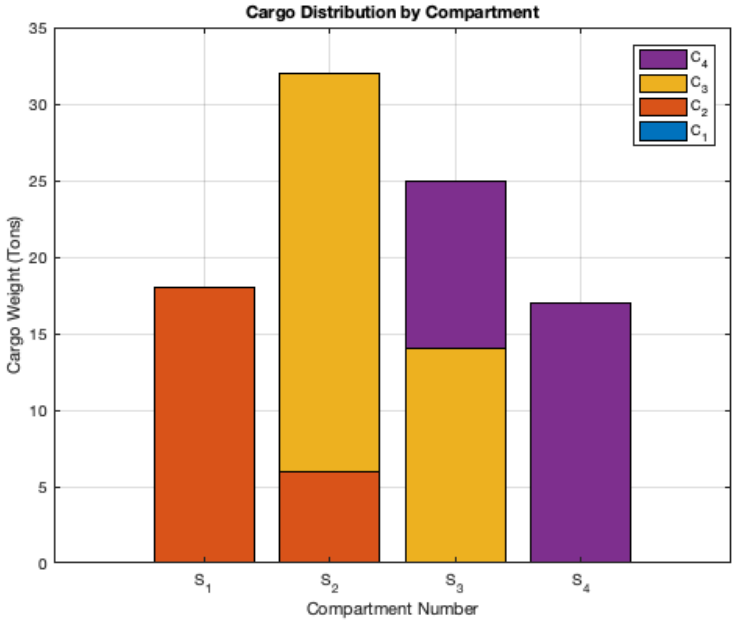


Figure 3: Cargo Distribution by Compartment

By comparing the expectation with the results (plot in figure 3) we can say that **the results align with expectations**: compartments with higher bonus profits are filled first with the cargo offering the highest profit per ton (e.g., Cargo 4 fills Compartment 4).

Cargo 1 is unused (shown by the fact that we don't have it in figure 3) due to its low profit and weight limitations.

We can also notice that the volume constraint is nearly irrelevant in this case, as compartments reach their weight capacity before the volume limit start to being considered.

4. Cycling and degeneracy [10 points]

1. Create a script `exercise3.m` which uses the simplex method implemented above to solve this problem

The implementation of `exercise3.m` is quite simple since I simply follow the steps mentioned in exercise 3 (related with script `exercise2.m`) but using simpler constrains and coefficients.

By running the script, the following error is thrown:

Error using simplexSolve (line 84)
Incorrect loop, more iterations than the number of basic solutions

From theory, we know that *the algorithm would achieve the optimal solution in the maximum possible number of iterations, which is exactly equal to the number of basic solutions.* (quoted from Project 5) accordingly to the formula:

$$N = \frac{(m+n)!}{m!n!}$$

That is exactly the formula used to compute `itMax` (i.e. the maximum number of iterations) in `simplex(...)` function as `itMax = factorial(m + n) / (factorial(m) * factorial(n));` where m and n means the size of the constrains matrix ($3 \times 2 - matrix$, that means $itMax = 10$).

Just for the purpose of testing, I've tried to add a print statement in `simplexSolve(...)` to see the current iteration over the max iterations, confirming that the we exceed `itMax`. This proves that there are some loop (or cyclic) probably due to multiple variables that repeatedly swap the same pairs of values.

2. Represent the constraints graphically and try to use this information to explain the behavior of your solver in the previous point

Looking at the constrains we notice that we have 3 constrains (4 if we also consider the one related with non-negativity) and 2 unknowns, and this means that the system is **over-constrained** and this may cause some issues. This is correct because we notice that for any positive value of x_1 and x_2 (non-negativity constrain), if $4x_1 + x_2 \leq 8$ is satisfied, also $4x_1 + 2x_2 \leq 8$ will be satisfied since the left-hand-side of the second one is bigger and the right-hand-side of the two is the same.

So the constrain $4x_1 + x_2 \leq 8$ is redundant since don't provide any "more specific constrain" (i.e. don't reduce) to the Feasible area.

By trying to plot the constrains of the linear problem (implemented in file `exercise3Plot.m`, we obtain the following figure:

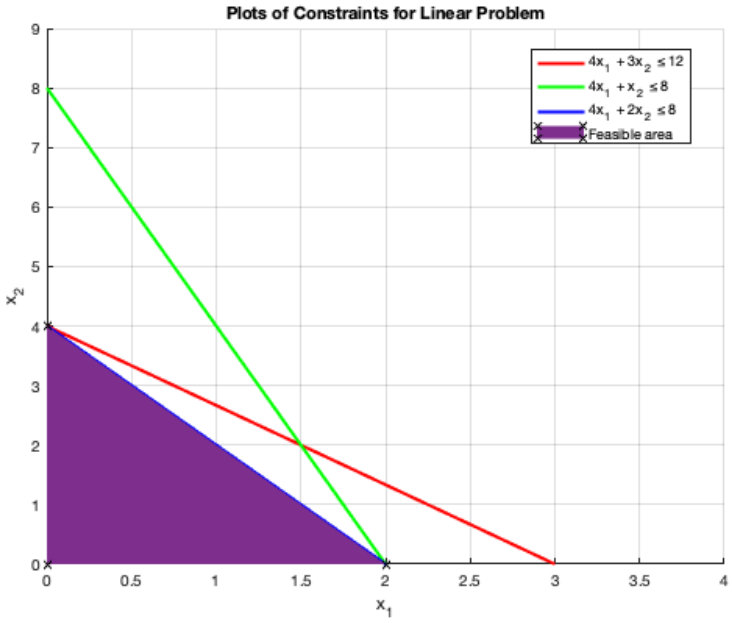


Figure 4: Ex4 constrains and Feasible Area

From the figure 4 we see exactly what is said before; the Feasible area correspond to the constrain $4x_1 + 2x_2 \leq 8$ and the green constrain ($4x_1 + x_2 \leq 8$) is redundant.

If a constrain is redoundant means that certain vertices of the feasible region are shared by more than two active constraints, introducing **degeneracy** (the solver might cycle and/or exceed the expected number of iterations).

Degeneracy can lead the solver to perform unnecessary swaps of variables without improving the solution. This increases the number of iterations in a useless way exceeding the number of max iterations and so, producing the error seen in the previous point.