Università della Svizzera italiana

Institute of Computing CI

Numerical Computing **2024**

Student: POLAD BAKHISHZADE

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

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

1.1. Minimization Problem

The task is to minimize the function $z = 4x + y$, known as the objective function. This function represents the value to be optimized—in this case, minimized—based on the decision variables x and y. The constraints $x + 2y \leq 40$, $x + y \geq 30$, $2x + 3y \geq 72$, and $x, y \geq 0$ restrict the feasible values of x and y, defining a region on the Cartesian plane where all inequalities are satisfied simultaneously.

The solution is determined graphically by plotting the boundaries of the constraints, which divide the plane into regions. The feasible region is the area where all constraints overlap. According to the Fundamental Theorem of Linear Programming, the optimal solution lies at one of the vertices of this region. The vertices of the feasible region are obtained by solving pairs of constraints as equations. Specifically, the boundary equations $x + 2y = 40$ and $2x + 3y = 72$ intersect at the point $(24, 8)$. Similarly, solving $x + y = 30$ with $2x + 3y = 72$ gives $(36, 0)$, and combining $x + 2y = 40$ with $y = 0$ yields $(40, 0)$.

To determine the optimal value of the objective function, it is evaluated at these vertices. At $(24, 8)$, the value of z is $4(24) + 8 = 104$. For $(36, 0)$ and $(40, 0)$, the values are 144 and 160, respectively. The minimum value is $z = 104$, achieved at $(24, 8)$.

The graphical representation of the feasible region and the optimal solution is shown in Figure 1.

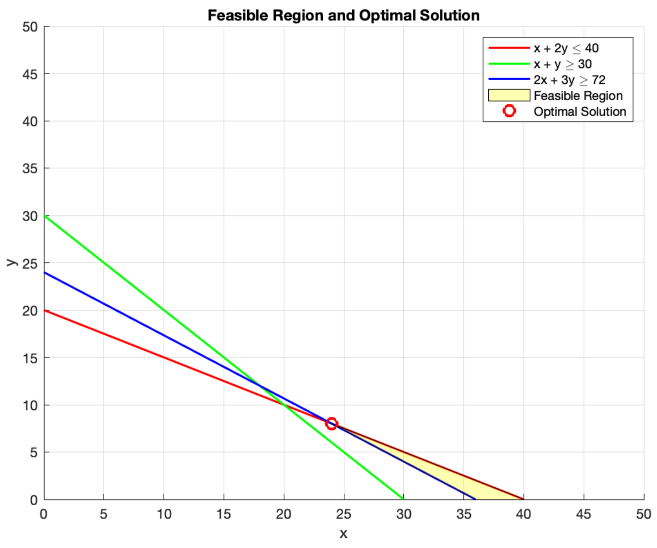

Figure 1: Graphical representation of the feasible region and the optimal solution.

1.2 Maximization Problem for Tailor's Net Profit

This task focuses on formulating and solving a linear programming problem to maximize the net profit of a tailor producing two types of trousers. The problem is defined as follows:

$$\max z = 60x_1 + 70x_2$$
$$\text{subject to: } \begin{cases} 25x_1 + 40x_2 \leq 7000, \\ x_1 + x_2 \leq 265, \\ x_1, x_2 \geq 0, \end{cases}$$

where x_1 and x_2 represent the quantities of trousers of type 1 and type 2, respectively, z denotes the total profit in CHF, and the constraints reflect the budget limitation of CHF 7000 for raw materials and the estimated monthly demand of 265 trousers.

The feasible region defined by the constraints is bounded and was computed graphically. By evaluating the objective function at all vertices of the feasible region, as per the Fundamental Theorem of Linear Programming, the optimal solution was identified. The vertices of the feasible region are as follows:

$$\text{Vertices: } \begin{bmatrix} 240 & 25 \\ 265 & 0 \\ 0 & 175 \end{bmatrix},$$

with corresponding objective function values:

$$\text{Objective function values: } [16150, 15900, 12250] \text{ CHF}.$$

The optimal solution is achieved at the vertex $(240, 25)$, yielding a maximum profit of:

$$z = 16150 \text{ CHF}.$$

The graphical solution is depicted in Figure 2, where the feasible region is shaded in yellow, and the optimal solution is highlighted. The solution is feasible, satisfying all constraints, and demonstrates the utility of linear programming in decision-making for the Tailor's profit maximization.

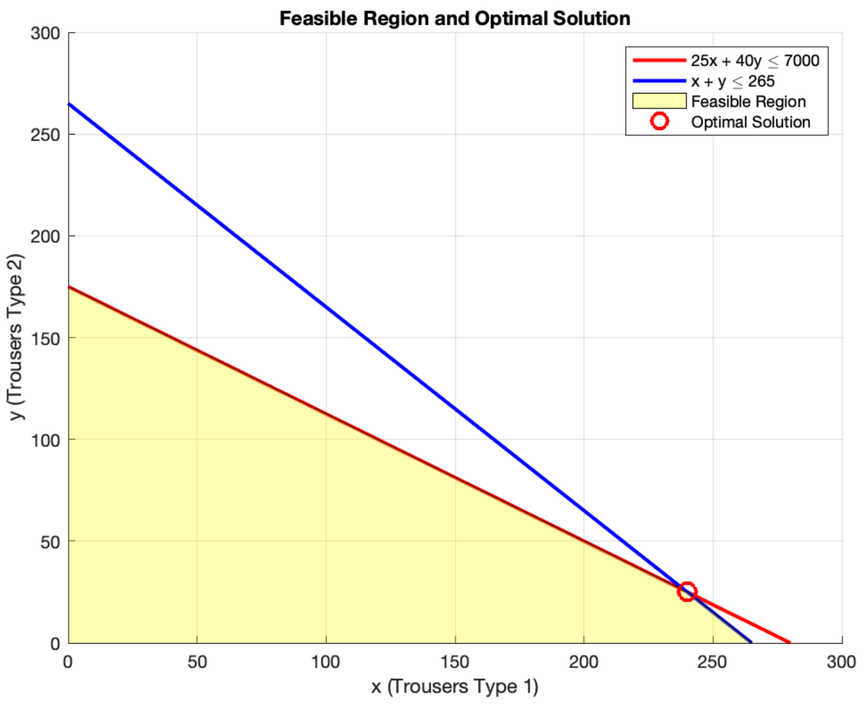

Figure 2: Feasible region and optimal solution for the tailor's maximization problem.

2. Implementation of the simplex method [30 points]

This exercise involved completing the simplex method implementation by addressing key `TODOs` to solve linear programming problems. The simplex algorithm optimizes a linear objective function iteratively while satisfying equality constraints. The main `TODOs` were implemented as follows.

In the `simplexSolve` function, the reduced cost coefficients r_D were computed as:

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

where c_B and c_D are the coefficients of basic and non-basic variables, respectively. Basic variables are those included in the current solution, while non-basic variables remain zero. The matrices B and D represent the columns of the constraint matrix corresponding to the basic and non-basic variables. The reduced costs r_D indicate whether the solution is optimal; for maximization problems, all r_D values must be non-positive, and for minimization, they must be non-negative. If not optimal, the variable with the most violating reduced cost (largest positive for maximization, smallest negative for minimization) was identified as the entering variable.

To maintain feasibility, the exiting variable was determined by evaluating the ratio of $B^{-1}h$ to the column of $B^{-1}D$ corresponding to the entering variable. Only positive ratios were considered, ensuring feasibility, and the smallest ratio identified the exiting variable. Matrices B and D were updated to reflect this exchange, and the basic variable values x_B were recomputed as:

$$x_B = B^{-1}h - B^{-1}D x_D,$$

where h is the right-hand side vector of constraints, and x_D represents the non-basic variables.

The `simplex` function calculates the maximum number of basic feasible solutions as:

$$it_{\max} = \frac{(n+m)!}{n! \cdot m!},$$

where n is the number of decision variables, and m is the number of constraints. To solve the problem using the simplex algorithm, it must first be standardized. This involves converting inequality constraints to equalities by adding slack or surplus variables and ensuring all variables are non-negative. As a result, the total number of variables increases to $n+m$. The formula calculates the total number of possible basic feasible solutions, which is the maximum number of iterations the simplex algorithm may perform.

The optimal value of the objective function is computed as:

$$z = c_B x_B,$$

where c_B are the coefficients of the basic variables, and x_B are their values in the solution.

The implementation was validated using the provided `testSimplex` script, which included six test cases. All tests passed, confirming the correctness of the implementation.

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

3.1 Linear Program Formulation

This section formulates the problem of maximizing the profit from a cargo aircraft as a linear programming problem. The task involves distributing four types of cargos across four compartments while respecting weight, volume, and availability constraints.

The objective is to maximize the total profit z generated by transporting the cargo. The profit depends on both the type of cargo and the compartment where it is stored. Compartments with better storage conditions provide higher profits.

$$\begin{aligned}\max z = {} & 135x_{11} + 200x_{12} + 410x_{13} + 520x_{14} \\ & + 1.1(135x_{21} + 200x_{22} + 410x_{23} + 520x_{24}) \\ & + 1.2(135x_{31} + 200x_{32} + 410x_{33} + 520x_{34}) \\ & + 1.3(135x_{41} + 200x_{42} + 410x_{43} + 520x_{44}).\end{aligned}$$

We can also rewrite it in a compressed form:

$$\max z = \sum_{i=1}^{4} \sum_{j=1}^{4} p_{ij} x_{ij},$$

where x_{ij} represents the tons of cargo j allocated to compartment i, and p_{ij} is the profit per ton for storing cargo j in compartment i. The profit p_{ij} is calculated using the formula:

$$p_{ij} = p_j \cdot (1 + 0.1(i - 1)),$$

where p_j represents the base profit per ton for cargo j. The values of p_j for each cargo type are as follows:

$$p_j = \begin{cases} 135, & j = 1 \\ 200, & j = 2 \\ 410, & j = 3 \\ 520, & j = 4 \end{cases}$$

The term $1 + 0.1(i - 1)$ adjusts the profit based on the compartment i, which reflects better storage conditions as we move from $S1$ to $S4$. Specifically:

$$\begin{aligned} &\text{S1 } (i = 1) : \text{No adjustment, } p_{ij} = p_j, \\ &\text{S2 } (i = 2) : 10\% \text{ increase, } p_{ij} = 1.1 \cdot p_j, \\ &\text{S3 } (i = 3) : 20\% \text{ increase, } p_{ij} = 1.2 \cdot p_j, \\ &\text{S4 } (i = 4) : 30\% \text{ increase, } p_{ij} = 1.3 \cdot p_j. \end{aligned}$$

Here, $S1$, $S2$, $S3$, and $S4$ represent the four compartments in the cargo aircraft. $S1$ offers standard storage conditions with no profit adjustment, while $S2$, $S3$, and $S4$ provide increasingly better storage conditions, leading to higher profits for cargos stored in those compartments.

This objective function ensures that both the type of cargo and the economic advantage of better compartments are considered, aiming to maximize the total profit while satisfying all constraints.

Constraints:

1. *Weight Capacity of Compartments:* The total weight of all cargos in each compartment must not exceed its capacity:

$$\sum_{j=1}^{4} x_{ij} \leq W_i, \quad i = 1, 2, 3, 4$$

where $W = [18, 32, 25, 17]$ tons specifies the maximum allowable weight for compartments S1, S2, S3, and S4, respectively. This ensures that no compartment is overloaded.

2. *Volume Capacity of Compartments:* The total volume of all cargos in each compartment must not exceed its storage capacity:

$$\sum_{j=1}^{4} v_j x_{ij} \leq V_i, \quad i = 1, 2, 3, 4$$

where $V = [11930, 22552, 11209, 5870]$ m^3 specifies the maximum allowable volume for each compartment, and $v = [320, 510, 630, 125]$ m^3/ton is the volume per ton of each cargo type. This constraint ensures that the spatial limits of each compartment are respected.

3. *Cargo Availability:* The total allocation of each cargo across all compartments must not exceed its available quantity:

$$\sum_{i=1}^{4} x_{ij} \leq A_j, \quad j = 1, 2, 3, 4$$

where $A = [16, 32, 40, 28]$ tons specifies the total availability for each cargo type. This ensures that the allocation does not exceed the available supply.

4. *Non-Negativity:* The allocation x_{ij} must be non-negative:

$$x_{ij} \geq 0, \quad 0 \leq i, j \leq 4$$

This guarantees that all allocations are physically meaningful, as negative weights are not possible.

3.2 Optimal Allocation and Results

The optimal cargo allocation problem was solved using the simplex method to maximize profit. The compartments (S1, S2, S3, S4) and cargos (C1, C2, C3, C4) were evaluated based on their respective profit adjustments and capacities. The optimal allocation matrix is presented in Table 1.

Table 1: Optimal Allocation of Cargos to Compartments (tons)

Cargos	**S1**	**S2**	**S3**	**S4**	**Total Cargo (tons)**
C1	0	0	0	0	0
C2	18	6	0	0	24
C3	0	26	14	0	40
C4	0	0	11	17	28
Total Compartment (tons)	18	32	25	17	

The resulting allocation yields a maximum profit of 41,890 CHF. The visualization of cargo allocation across compartments is shown in Figure 3.

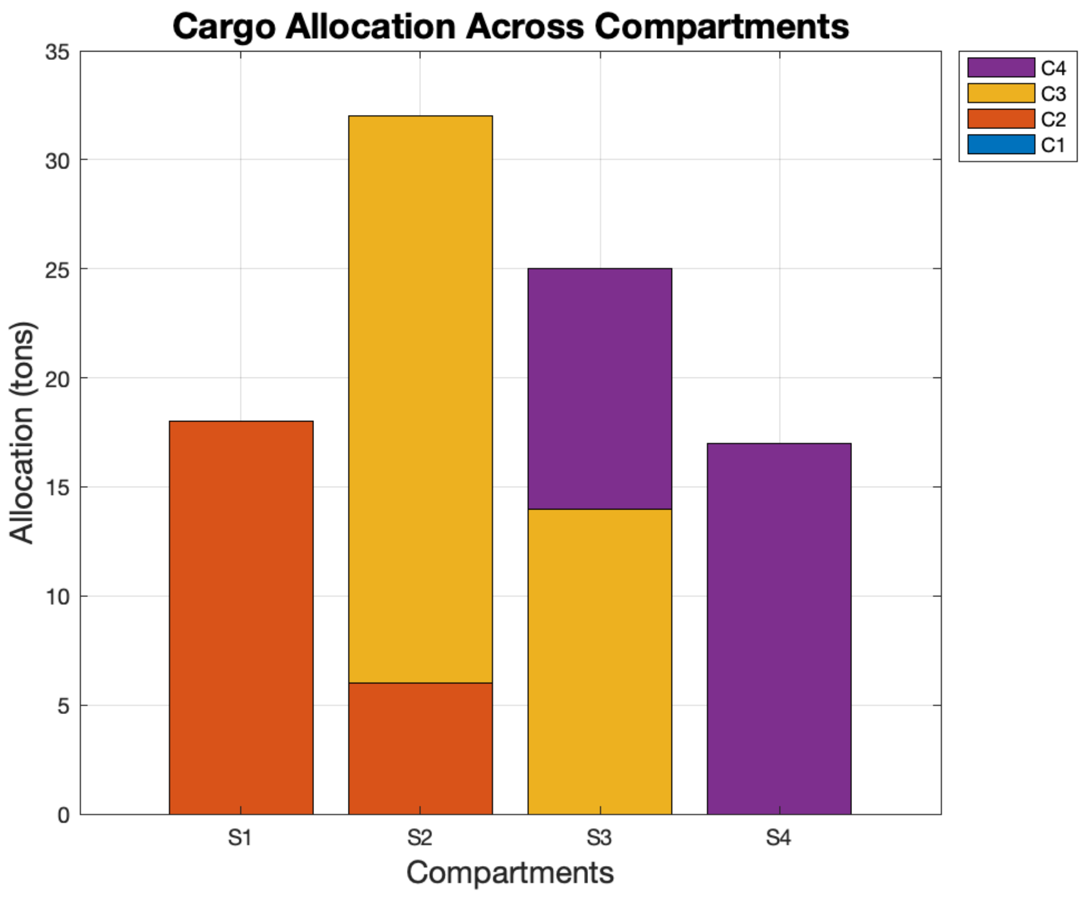

Figure 3: Cargo Allocation Across Compartments

The results demonstrate that higher-profit cargos, such as C3 (410 CHF/ton) and C4 (520 CHF/ton), are prioritized and allocated to compartments with higher profit multipliers, specifically

S2 (+10%), S3 (+20%), and S4 (+30%). Cargo C3 is allocated 26 tons in S2 and 14 tons in S3, while C4 is allocated 11 tons in S3 and 17 tons in S4, as shown in Table 1 and Figure 3. In contrast, the lower-profit cargo C1 (135 CHF/ton) is excluded entirely due to its minimal contribution to overall profit. The solution strictly adheres to all constraints, ensuring that the total allocation in each compartment does not exceed its respective weight capacity W (e.g., 32 tons in S2, 25 tons in S3). This allocation achieves optimal resource utilization within the defined limits.

4. Cycling and degeneracy [10 points]

4.1 Analysis of the Problem

Cycling in the simplex method occurs when the algorithm revisits the same basic feasible solution, leading to an infinite loop without finding the optimal solution. This issue arises primarily in cases of degeneracy, where multiple basic feasible solutions have the same objective function value. To demonstrate this phenomenon, the following linear programming problem was analyzed:

$$\text{Objective:} \quad \max z = 3x_1 + 4x_2,$$

$$\text{Subject to:} \quad \begin{cases} 4x_1 + 3x_2 \leq 12, \\ 4x_1 + x_2 \leq 8, \\ 4x_1 + 2x_2 \leq 8, \\ x_1, x_2 \geq 0. \end{cases}$$

In matrix form, this problem can be expressed as:

$$\begin{aligned} \text{Maximize:} \quad & z = \mathbf{c}^\top \mathbf{x}, \\ \text{Subject to:} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{h}, \quad \mathbf{x} \geq 0, \end{aligned}$$

where

$$\mathbf{c} = \begin{bmatrix} 3 \\ 4 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 4 & 3 & 1 & 0 & 0 \\ 4 & 1 & 0 & 1 & 0 \\ 4 & 2 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} 12 \\ 8 \\ 8 \end{bmatrix}.$$

Here, $\mathbf{c}$ represents the coefficients of the objective function, $z = 3x_1 + 4x_2$, with additional zeros for slack variables. The matrix $\mathbf{A}$ contains the coefficients of the constraints, with slack variables introduced to transform inequalities into equalities:

$$4x_1 + 3x_2 + x_3 = 12, \quad 4x_1 + x_2 + x_4 = 8, \quad 4x_1 + 2x_2 + x_5 = 8.$$

Finally, $\mathbf{h}$ is the right-hand side vector of the constraints, corresponding to the upper bounds 12, 8, and 8 from the original problem.

Initially, upon running exercise3.m file, the simplex algorithm encountered cycling and exceeded the maximum iteration limit, as shown by the error message: "Incorrect loop, more iterations than the number of basic solutions." This confirmed that the problem exhibits degeneracy and cycling, preventing the algorithm from converging.

To address this issue, the error condition in the simplexSolve function was modified to instead generate a warning when the iteration limit is reached. This change allowed the algorithm to terminate gracefully, avoiding infinite looping. The updated code of simplexSolve.m for handling this scenario is shown below:

```
nIter = nIter + 1;
if nIter > itMax
    warning("Iteration limit exceeded. Stopping to avoid infinite loop.");
    return;
end
```

The final solution was returned with basic variables $x_1 = 2$, $x_3 = 4$, $x_4 = 0$, and the maximum profit $z = 6$. The simplex algorithm identified $x_1 = 2$ and $x_2 = 0$ as the optimal solution, which satisfies all constraints and maximizes the objective function $z = 3x_1 + 4x_2$. Degeneracy in the problem is evident as basic variable $x_2 = 0$. This can lead to repeated iterations over degenerate vertices, causing cycling. Despite the iteration limit warning:

```
Warning: Iteration limit exceeded. Stopping to avoid infinite loop.
```

the algorithm successfully returned the correct optimal solution:

$$x_1 = 2, \quad x_2 = 0, \quad z = 6.$$

This solution adheres to all constraints, confirming the correctness of the result despite degeneracy.

4.2 Visualization of Constraints and the Feasible Region

To provide further clarity, the constraints and feasible region were visualized. The feasible region is defined by the intersection of the constraints:

$$\begin{aligned} 4x_1 + 3x_2 &\leq 12, \\ 4x_1 + x_2 &\leq 8, \\ 4x_1 + 2x_2 &\leq 8. \end{aligned}$$

The plot below illustrates the feasible region and the optimal solution:

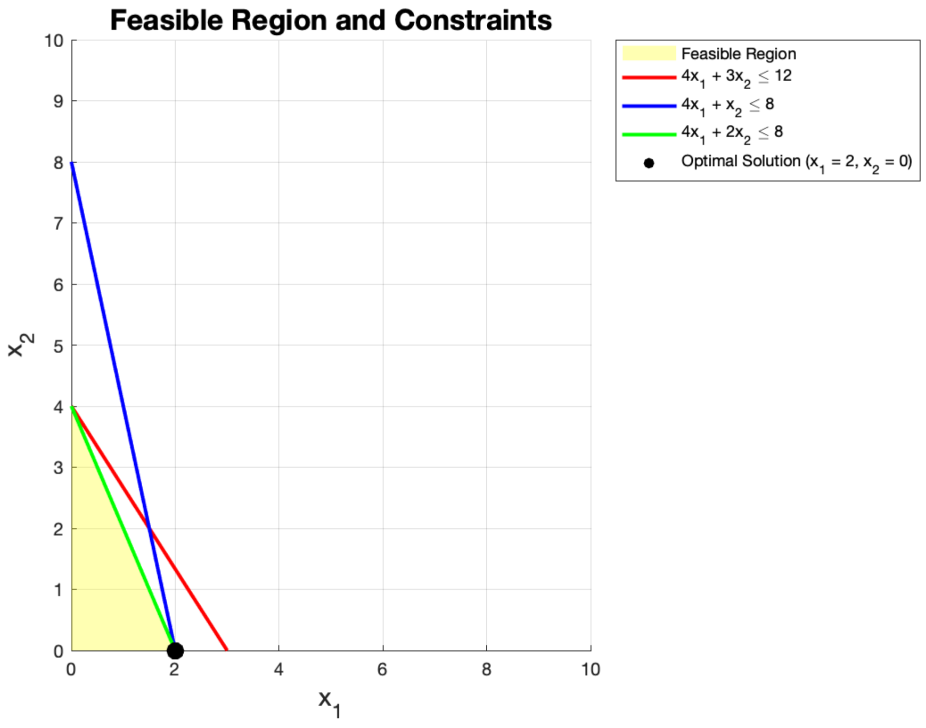

Figure 4: Visualization of the Feasible Region and Constraints.

The system has two unknowns (x_1, x_2) and three constraints, resulting in a feasible region defined by their intersection. The simplex algorithm identified the optimal solution at $x_1 = 2, x_2 = 0$, marked as a black dot in Figure 4, where the objective function $z = 3x_1 + 4x_2$ is maximized, giving $z = 6$.

Degeneracy arises because the optimal solution lies at a vertex where all three constraints intersect ($4x_1 + 3x_2 = 12$, $4x_1 + x_2 = 8$, and $4x_1 + 2x_2 = 8$), even though there are only two variables (x_1, x_2).

As a result, the solution can be represented by different sets of basic and non-basic variables. While the numerical values of x_1, x_2, and the slack variables remain unchanged, the simplex method treats these representations as distinct solutions because it tracks which variables are basic or non-basic, not the actual values.

Cycling occurs because the simplex method alternates between these representations, swapping basic and non-basic variables without improving the objective function. For instance, x_2 (a non-basic variable) might enter the basis, replacing x_3 (a basic variable), only for x_3 to re-enter in the next iteration. This repeated swapping keeps the solution at the same vertex ($x_1 = 2, x_2 = 0$), causing cycling.

By imposing an iteration limit, the solver avoided infinite cycling and returned the correct optimal solution. Specifically, the algorithm in `simplexSolve.m` was modified to terminate gracefully by generating a warning when the iteration limit was reached, preventing infinite looping at degenerate solutions. Previously, the algorithm would throw an error upon exceeding the iteration limit.