# Homework Assignment (Problem Set) 1:

Note, Problem Set 1 directly focuses on Modules 1 and 2; Introduction to Decision Analysis and Formulation and Solving Linear Programs.

***5 questions***

## Rubric:

All questions worth 20 points

20 Points: Answer and solution are fully correct and detailed professionally.

16-19 Points: Answer and solution are deficient in some manner but mostly correct.

11-15 Points: Answer and solution are missing a key element or two.

1-10 Points: Answer and solution are missing multiple elements are significantly deficient/incomprehensible.

0 Points: No answer provided.

## Question 1:

Clocks in Sox is a small company that manufactures wristwatches in two separate workshops, each with a single watch maker (or horologist, as they are called). Each watchmaker works a different number of hours per month to make the three models sold by Clocks in Sox: Model A, Model B, and Model C. Watchmaker 1 works a total of 200 hours per month while Watchmaker 2 works a total of 150 hours per month and the time (in hours) and cost of materials for each watch differ by watchmaker due to their experience and equipment (shown below). Each month, Clocks in Sox must produce a total of 60 Model A watches, 80 Model B watches, and 70 Model C watches. Clearly formulate a linear program (LP) to minimize the cost of manufacturing the desired amount of watches.

Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Workshop | Model A | | Model B | | Model C | |
| Cost ($) | Time | Cost ($) | Time | Cost ($) | Time |
| Watchmaker 1 | 10 | 2 | 11 | 4 | 12 | 3 |
| Watchmaker 2 | 9 | 9 | 10 | 4 | 13 | 7 |

**Solution 1:**

*Decision Variables:*

Let's denote the number of watches of each model made by each watchmaker as follows:

* : number of Model A watches made by Watchmaker 1
* : number of Model A watches made by Watchmaker 2
* : number of Model B watches made by Watchmaker 1
* : number of Model B watches made by Watchmaker 2
* : number of Model C watches made by Watchmaker 1
* : number of Model C watches made by Watchmaker 2

*Objective Function:*

We want to minimize the total cost of manufacturing, so the objective function becomes:

*Minimize*

*Constraints:*

1. Time Constraints:

* The total time spent by Watchmaker 1 is and this cannot exceed 200 hours.
* The total time spent by Watchmaker 2 is and this cannot exceed 150 hours.

1. Production Constraints:

* The total number of Model A watches produced by both watchmakers is and this should be equal to 60.
* The total number of Model B watches produced by both watchmakers is and this should be equal to 80.
* The total number of Model C watches produced by both watchmakers is and this should be equal to 70.

1. Non-negativity Constraints:

So, the complete LP formulation is:

Minimize:

Subject to:

## Question 2:

Consider the following linear program:

Min Z = -9x1 + 18x2

Subject To

-x1 + 5x2 ≥ 5

x1 + 4x2 ≥ 12

x1 + x2 ≥ 5

x1 ≤ 5

x1, x2 ≥ 0

Part A:  Write the LP in standard equality form.

Part B:  Solve the original LP graphically (to scale).  Clearly identify the feasible region and, if one or more exist, the optimal solution(s) (provide exact values for x1, x2, and Z).

**Solution 2 (Part A):**

To write the linear program in standard equality form:

1. All constraints must be equations.

2. The right-hand side (RHS) values of the constraints must be non-negative.

3. All variables must be non-negative, which is already given in the original LP as \( x\_1, x\_2 \geq 0 \).

First, introduce slack (excess) variables to convert the inequalities into equations.

For the constraint , introduce a surplus variable such that:

For the constraint , introduce a surplus variable such that:

For the constraint , introduce a surplus variable such that:

For the constraint , introduce a slack variable such that:

Rewriting the objective function. So, the LP in standard form becomes:

Minimize:

Subject to:

**Solution 2 (Part B):** # Python Code for plotting the graphical solution

# Define the inequality functions

def f1(x1):

    return (5 + x1) / 5.0

def f2(x1):

    return (12 - x1) / 4.0

def f3(x1):

    return 5 - x1

def objective(x1, Z=-45):

    return (Z + 9\*x1) / 18.0

# Create a range for x1

x1 = np.linspace(0, 10, 400)

# Plot each constraint

plt.plot(x1, f1(x1), label=r'$-x\_1 + 5x\_2 \geq 5$')

plt.plot(x1, f2(x1), label=r'$x\_1 + 4x\_2 \geq 12$')

plt.plot(x1, f3(x1), label=r'$x\_1 + x\_2 \geq 5$')

plt.axvline(x=5, color='r', label=r'$x\_1 \leq 5$')

# Plot the objective function

plt.plot(x1, objective(x1), 'k--', label=r'$Z = -9x\_1 + 18x\_2 = -45$')

# Fill the feasible region

y1 = np.minimum(np.minimum(f1(x1), f2(x1)), f3(x1))

plt.fill\_between(x1, y1, where=[(x <= 5) and (y >= -4) for x, y in zip(x1, y1)], color='gray', alpha=0.5)

# Mark the optimal solution

plt.scatter(5, 0, color='magenta', marker='o', s=100, zorder=5, label='Optimal Solution $(x\_1=5, x\_2=0)$')

# Labeling the graph

plt.xlim((0, 10))

plt.ylim((-4, 10))

plt.xlabel(r'$x\_1$')

plt.ylabel(r'$x\_2$')

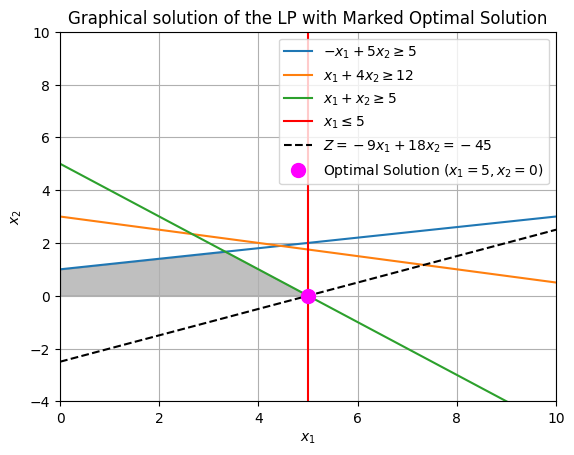
plt.legend()

plt.title('Graphical solution of the LP with Marked Optimal Solution')

plt.grid(True)

plt.show()

Output:



Explaining the graphical solution:

1. Feasible Region: The feasible region is the shaded area on the graph where all the inequalities overlap.
2. Vertices of the Feasible Region: The vertices are the corner points of the feasible region. From the graph, the vertices can be visually identified as the points of intersection of the lines:

* Intersection of and
* Intersection of and
* Intersection of and
* Intersection of and

1. Optimal Solution: The objective function is . The goal is to minimize Z. To do this, we plug in the coordinates of each vertex into the objective function and determine which gives the smallest value:

* Vertex 1 ():
* Vertex 2 ():
* Vertex 3 ():
* Vertex 4 ():

Of these values, the smallest is at the point , which means that this is the optimal solution.

## Question 3:

InvestCo currently has $500 in cash. InvestCo receives revenues at the start of months 1 – 4, after which it pays bills (see Table 2 below). Any money left over may be invested and interest for one month is 0.5%, two months is 2%, three months is 4%, and four months is 8% (total - no compounding). Use linear programming to determine an investment strategy that maximizes cash on hand at the beginning of month 5. Formulate an LP to maximize InvestCo’s profit. Do not solve.

*Hint: What is coming in and what is going out each month?*

Table

|  |  |  |
| --- | --- | --- |
| Month | Revenues ($) | Bills ($) |
| 1 | 600 | 700 |
| 2 | 900 | 400 |
| 3 | 300 | 700 |
| 4 | 500 | 350 |

## Solution 3:

Decision Variables:

Let's denote the cash invested by InvestCo for each month with a variable I , such that:

* I1m: Amount invested for 1 month at the start of month 1
* I1t: Amount invested for 2 months at the start of month 1
* 1th: Amount invested for 3 months at the start of month 1
* I1f: Amount invested for 4 months at the start of month 1

And so on for other months. We'll have similar variables I2m, I2t, I2th, I2f for month 2, I3m, I3t, I3th, I3f for month 3, and I4m, I4t, I4th, I4f for month 4.

Objective Function: We want to maximize the total cash on hand at the start of month 5, so the objective function becomes:

Maximize

Z = 500 + R1 - B1 + I1m(1 + 0.005) + I1t(1 + 0.02) + I1th(1 + 0.04) + I1f(1 + 0.08)

+ R2 - B2 + I2m(1 + 0.005) + I2t(1 + 0.02) + I2th(1 + 0.04)

+ R3 - B3 + I3m(1 + 0.005) + I3t(1 + 0.02)

+ R4 - B4 + I4m(1 + 0.005)

Where:

* Ri: Revenues for month i
* Bi: Bills for month i

Constraints:

1. Cash Constraints:

The cash invested in any month can't be more than the cash available (i.e., cash on hand + revenue for that month - bills for that month).

For month 1:

I1m + I1t + I1th + I1f  <= 500 + R1 - B1

For month 2:

I2m + I2t + I2th + I2f <= I1m + R2 - B2

For month 3:

I3m + I3t + I3th <= I1t + I2m + R3 - B3

For month 4:

I4m + I4t <= I1th + I2t + I3m + R4 - B4

2. Non-negativity Constraints:

Iim, Iit, Iith, Iif >= 0

for all i n {1,2,3,4}.

The above formulation gives the mathematical representation of InvestCo's problem.

## Question 4:

Floor is Java sells premium coffee to restaurants. They sell two roasts which they call (cleverly) Roast 1 and Roast 2, each of which is a blend of Columbian and Arabica coffee beans. Columbian beans cost $20 for a 5 pound box while Arabica beans cost $15 for a 6 pound box. Roast 1 sells for $6 per pound and must be at least 75% Columbian beans, while Roast 2 sells for $5 per pound and must be at least 60% Columbian beans. At most, 40 pounds of Roast 1 and 60 pounds of Roast 2 can be sold each month.

Part A: Formulate an LP to maximize Floor is Java’s profit.

Part B: Solve the LP (provide **exact** values (do not restrict to integer) for all variables and the optimal objective function).

## Solution 4(Part A):

Decision Variables:

: Pounds of Columbian beans used for Roast 1.

: Pounds of Arabica beans used for Roast 1.

: Pounds of Columbian beans used for Roast 2.

: Pounds of Arabica beans used for Roast 2.

Objective Function:

The profit is the selling price of the roasts minus the cost of the beans.

For Roast 1:

Revenue from Roast =

Cost of beans for Roast 1 =

For Roast 2:

Revenue from Roast 2 =

Cost of beans for Roast 2 =

Objective is to maximize the profit:

Maximize

Constraints:

1. The maximum amount that can be sold:
2. Composition constraints for the roasts:

Roast 1 must be at least 75% Columbian beans:

This translates to:

Roast 2 must be at least 60% Columbian beans:

1. Non-negativity constraints:

Formulated LP:

Maximize

Subject to:

This LP will determine the optimal blending strategy to maximize Floor is Java's profit.

## Solution 4(Part B):

# Python Code for solving the linear programming problem

# Initialize the LP problem

lp\_prob = pulp.LpProblem("Floor\_is\_Java\_Profit\_Maximization", pulp.LpMaximize)

# Define the decision variables

x1 = pulp.LpVariable('x1', lowBound=0)  # Pounds of Columbian beans for Roast 1

y1 = pulp.LpVariable('y1', lowBound=0)  # Pounds of Arabica beans for Roast 1

x2 = pulp.LpVariable('x2', lowBound=0)  # Pounds of Columbian beans for Roast 2

y2 = pulp.LpVariable('y2', lowBound=0)  # Pounds of Arabica beans for Roast 2

# Define the objective function

revenue = 6\*(x1 + y1) + 5\*(x2 + y2)

cost = (20 \* x1/5 + 15 \* y1/6) + (20 \* x2/5 + 15 \* y2/6)

lp\_prob += revenue - cost, "Total Profit"

# Define the constraints

lp\_prob += x1 + y1 <= 40, "Roast1\_sale\_constraint"

lp\_prob += x2 + y2 <= 60, "Roast2\_sale\_constraint"

lp\_prob += 0.25\*x1 - 0.75\*y1 >= 0, "Roast1\_composition\_constraint"

lp\_prob += 0.40\*x2 - 0.60\*y2 >= 0, "Roast2\_composition\_constraint"

# Solve the LP

lp\_prob.solve()

# Print the results

print(f"x1 (Columbian beans for Roast 1) = {x1.varValue}")

print(f"y1 (Arabica beans for Roast 1) = {y1.varValue}")

print(f"x2 (Columbian beans for Roast 2) = {x2.varValue}")

print(f"y2 (Arabica beans for Roast 2) = {y2.varValue}")

print(f"Maximum Profit = ${pulp.value(lp\_prob.objective)}")

Output:

x1 (Columbian beans for Roast 1) = 30.0

y1 (Arabica beans for Roast 1) = 10.0

x2 (Columbian beans for Roast 2) = 36.0

y2 (Arabica beans for Roast 2) = 24.0

Maximum Profit = $191.0

## Question 5:

Food Beach, a local grocery store, is building a work schedule for its stockers and has specific requirements over each 24 hour period (shown in the table below). Each stocker must work two consecutive shifts.



Part A: Formulate an LP model to minimize the number of workers required to meet requirements.

Part B: Solve the LP (provide exact values for all variables and the optimal objective function).

## Solution 5 (Part A):

Decision Variables:

* Let be the number of stockers who start their shift at the beginning of time slot i.

Given the table, we have the following time slots:

: Time Slot 1: Midnight - 4am

: Time Slot 2: 4am - 8am

: Time Slot 3: 8am - Noon

: Time Slot 4: Noon - 4 pm

: Time Slot 5: 4pm - 8pm

: Time Slot 6: 8pm - Midnight

Objective Function:

We want to minimize the total number of stockers.

Minimize

Constraints:

1. For each time slot, the number of stockers working during that time should satisfy the requirement:

(Stockers who start at Midnight and 8pm) (for Midnight - 4am)

(for 4am - 8am)

(for 8am - Noon)

(for Noon - 4pm)

(for 4pm - 8pm)

(for 8pm - Midnight)

1. Non-negativity constraint:

LP Formulation:

Minimize

Subject to:

The above LP will determine the optimal number of stockers starting their shift at each time slot to minimize the total number of stockers and still meet the requirements.

## Solution 5 (Part B):

# Python Code for Solving the LP

# Create a LP problem instance

lp\_prob = pulp.LpProblem("Food\_Beach\_Scheduling", pulp.LpMinimize)

# Define the decision variables

x = pulp.LpVariable.dicts("x", [1, 2, 3, 4, 5, 6], 0, None, pulp.LpInteger)

# Define the objective function

lp\_prob += pulp.lpSum([x[i] for i in range(1, 7)])

# Define the constraints

lp\_prob += x[1] + x[6] >= 8

lp\_prob += x[1] + x[2] >= 7

lp\_prob += x[2] + x[3] >= 5

lp\_prob += x[3] + x[4] >= 4

lp\_prob += x[4] + x[5] >= 4

lp\_prob += x[5] + x[6] >= 7

# Solve the LP

lp\_prob.solve()

# Print the results

print(f"Optimal number of stockers starting at:")

for i in range(1, 7):

    print(f"Time Slot {i}: {x[i].varValue} stockers")

print(f"Minimum Total Number of Stockers Required: {pulp.value(lp\_prob.objective)}")

Output:

Optimal number of stockers starting at:

Time Slot 1: 1.0 stockers

Time Slot 2: 6.0 stockers

Time Slot 3: 0.0 stockers

Time Slot 4: 4.0 stockers

Time Slot 5: 0.0 stockers

Time Slot 6: 7.0 stockers

Minimum Total Number of Stockers Required: 18.0