Name: Manan Lohia

5

1.a We have ~~five~~ inequality constraints,

$x_1 + x_2 + \frac{1}{2} x_3 \leq 1$

$x_1, x_2, x_3, x_4 \geq 0$

For extreme points, any 4 of these inequalities must be satisfied at equality (must be active) and all 4 must be linearly independent

$\therefore$ There are 4 extreme points

		Active Constraints
i	$(0,0,0,0)$	$x_1, x_2, x_3, x_4 \geq 0$
ii	$(0,0,3,0)$	$x_1 + x_2 + \frac{1}{3} x_3 \leq 1,\ x_1, x_2, x_4 \geq 0$
iii	$(1,0,0,0)$	$x_1 + x_2 + \frac{1}{3} x_3 \leq 1,\ x_2, x_3, x_4 \geq 0$
iv	$(0,1,0,0)$	$x_1 + x_2 + \frac{1}{3} x_3 \leq 1,\ x_1, x_3, x_4 \geq 0$

b Given 6 inequalities

$x_1 - x_2 \leq 1$
$x_2 - x_3 \leq 1$
$x_3 \leq 1$
$x_1, x_2, x_3 \geq 0$

Following the same process, we get 8 extreme points

		Active constraints
i	$(0,0,0)$	$x_1, x_2, x_3 \geq 0$
ii	$(1,0,0)$	$x_1 - x_2 \leq 1,\ x_2, x_3 \geq 0$
iii	$(1,0,1)$	$x_1 - x_2 \leq 1,\ x_2 \geq 0,\ x_3 \leq 1$
iv	$(0,1,0)$	$x_2 - x_3 \leq 1,\ x_1, x_3 \geq 0$
v	$(0,2,1)$	$x_2 - x_3 \leq 1,\ x_1 \geq 0,\ x_3 \leq 1$
vi	$(2,1,0)$	$x_1 - x_2 \leq 1,\ x_2 - x_3 \leq 1,\ x_3 \geq 0$
vii	$(3,2,1)$	$x_1 - x_2 \leq 1,\ x_2 - x_3 \leq 1,\ x_3 \leq 1$
viii	$(0,0,1)$	$x_1, x_2 \geq 0,\ x_3 \leq 1$

c We have 3 linear equations and 4 variables. The 3 equations are independent and hence all three must be active.

Solution of

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$$

gives us

$x_1 + x_3 = 1,\ x_2 = 1,\ x_4 = 1$

$\therefore$ We get extreme points

		Other active constraints
i	$(1, 1, 0, 1)$	$x_3 \geq 0$
ii	$(0, 1, 1, 1)$	$x_1 \geq 0$

2.a We have the LP problem

$$\max \sum_{i=1}^{100} r_i x_i$$

$$\text{s.t} \sum_{i=1}^{100} x_i \leq b$$

$$x_i \geq 0, \quad i = 1, 2, \ldots 100$$

Let p be the Lagrangian multiplier.
∴ The relaxed problem is given by

$$\max \sum_{i=1}^{100} r_i x_i + p\left(b - \sum_{i=1}^{100} x_i\right)$$

Since $b - \sum_{i=1}^{100} x_i \geq 0$, we penalize the case where $b - \sum_{i=1}^{100} x_i < 0$ by restricting $\underline{\underline{p \geq 0}}$

Rearranging the terms in the relaxed problem, we get

$$\max pb + \sum_{i=1}^{100} (r_i - p) x_i$$

Since $x_i \geq 0 \ \forall i$, this results in

$$r_i - p \leq 0 \ \forall i$$

$$\Rightarrow \underline{\underline{p \geq r_i \ \forall i}}$$

$\therefore$ The dual is given by

$$\begin{aligned} \min \ & pb \\ \text{s.t} \ & p \geq r_i \ \forall i \\ & p \geq 0 \end{aligned}$$

b The optimal solution to the primal is obtained by allotting the entire budget to the stock with the highest return (say the m th stock)

$$\therefore \ x_m = b$$
$$x_i = 0 \ \forall i \neq m$$

And the value of the objective is

$$\underline{\underline{r_m b}}$$

However, incase all r_i are negative, the optimal solution is not investing at all, i.e $x_i = 0 \ \forall i$. $\therefore$ The objective in this case is 0

~~The solution~~

Since $p \geq r_i \ \forall i$, the solution to the dual is determined by $p = \max(r_1, r_2, \ldots r_m, 0)$ which is the same as $\underline{p = \max(r_m, 0)}$

$\therefore$ If r_m is positive, $p = r_m$ and the objective is $\underline{r_m b}$. However, if all r_i's are negative, then $p = 0$ & the objective is $\underline{0}$

c Case 1

Atleast 1 r_i is positive,

In this case $x_m = r_m$ and $p = r_m$

$\therefore$ Primal solution $= r_m b$
Dual solution $= r_m b$

$\therefore$ By weak duality, since x & p are feasible and the $pb = c'x$, the solutions are optimal

Similarly, for Case 2 [When all r_i's are negative], $x_i = 0$ & $p = 0$ $\Rightarrow$ The objective for both the primal and dual are equal to 0. Since the solutions are feasible and equal, the solution is optimal

3.a Writing the same problem in matrix form,

$$\max \quad c'x$$
$$\text{s.t} \quad Ax \leq b$$
$$x \geq 0$$

where

$$c = \begin{bmatrix} 10 \\ 12 \\ 12 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad b = \begin{bmatrix} 20 \\ 20 \\ 20 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$$

The dual for the same is

$$\min \quad p'b$$
$$\text{s.t} \quad p'A \geq c$$
$$p \geq 0$$

where $p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$ where p_i are the shadow prices and the remaining have the same definition as above

③

b Given optimal solution as

$x_1 = x_2 = x_3 = 4$

$$\therefore \quad x = \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix}$$

As per complimentary slackness conditions,

$$(c - p'A)'x = 0$$

$$\therefore \left(\begin{bmatrix} 10 \\ 12 \\ 12 \end{bmatrix} - \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} \right)' \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix} = 0$$

$$\Rightarrow \left(\begin{bmatrix} 10 \\ 12 \\ 12 \end{bmatrix} - \begin{bmatrix} p_1 + 2p_2 + 2p_3 \\ 2p_1 + p_2 + 2p_3 \\ 2p_1 + 2p_2 + p_3 \end{bmatrix} \right)' \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix} = 0$$

$$\cancel{\Rightarrow 4 \begin{bmatrix} 10 - p_1 - 2p_2 - 2p_3 \\ 12 - 2p_1 \end{bmatrix}}$$

$$\Rightarrow \begin{bmatrix} p_1 + 2p_2 + 2p_3 \\ 2p_1 + p_2 + 2p_3 \\ 2p_1 + 2p_2 + p_4 \end{bmatrix} = \begin{bmatrix} 10 \\ 12 \\ 12 \end{bmatrix}$$ (upon simplifying)

or $\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 10 \\ 12 \\ 12 \end{bmatrix}$

$\Rightarrow \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 18/5 \\ 8/5 \\ 8/5 \end{bmatrix}$

4. a The optimal quantity for the Curries service set is 2 and for the Bluetail service set is 5. For the remaining, the optimal solution is 0.

The total profit is $102 \times 2 + 89 \times 5$

$\therefore$ Total profit is 649

b The sensitivity report suggests that

i Increasing the availability of clay by one unit will increase the profit by 1.429 units. However, this is only valid until we increase the quantity by upto 23.33. The profit decreases by the same amount per unit till we decrease the quantity by 43.75

Similarly, for k_{lin}, the change in profit per unit change in the resource is 20.143 upto an increase of 5.6 or decrease of 3.5 units

For enamel and dry room, no amount of increase will affect the profit. However, decreasing enamel by more than 4 or Dry room by more than 28 will affect the optimal solution and hence the profit

~~The primary method constraint is an active constraint since it is an equality constraint.~~

Similarly, increasing the R.H.S of the last constraint ($P_1 - P_2 = 0$) by 1 corresponds to an increase of 11.429 in the optimal value. However, this is only valid for an increase of upto 3.5.

c Adding 20 lbs of Clay at \$1.1 presents an arbitrage opportunity as the manufacturer is able to make a profit of \$ 0.329 per pound of clay.

d If we change the constraint to $P_1 - P_2 \geq 0$, we expect $P_1 - P_2$ to take the value of 3.5 as that leads to an increase in the value of the objective function. Furthermore, since $P_1, P_2 \geq 0$, the least value P_2 can take is 0 and since $P_1 - P_2 \geq 3.5$, it implies that P_1 will be positive.

5. a We define the following variables

A : Weight of apples required (in 100s g)
B_a : " " bananas " "
B_1 : " " blueberries " "
D : " " durians " "
T : " " tangerines " "

Hence, the objective is

$$\min \; 0.5A + 0.3B_a + 2.5B_1 + 10D + 0.5T$$

The constraints are

$52A + 89B_a + 57B_1 + 147D + 53T \leq 3000$ [max calories]
$52A + 89B_a + 57B_1 + 147D + 53T \geq 500$ [min calories]

$14A + 23B_a + 14B_1 + 27D + 13T \leq 400$ [max carbohydrates]
$14A + 23B_a + 14B_1 + 27D + 13T \geq 50$ [min carbohydrates]

classmate

Date

Page

$2.4A + 2.6B_A + 2.4B_1 + 3.8D + 1.8T \leq 30$ [max Fibre]

$2.4A + 2.6B_A + 2.4B_1 + 3.8D + 1.8T \geq 20$ [min Fibre]

$54A + 64B_A + 54B_1 + 44D + 681T \leq 3500$ [max Vitamin A]

$54A + 64B_A + 54B_1 + 44D + 681T \geq 2000$ [min Vit A]

$4.6A + 8.7B_A + 9.7B_1 + 19.7D + 26.7T \leq 150$ [max Vit C]

$4.6A + 8.7B_A + 9.7B_1 + 19.7D + 26.7T \geq 75$ [min Vit C]

Furthermore, amount of fruit purchased cannot be negative

$\therefore A, B_A, B_1, D, T \geq 0$

Assignment 2

September 25, 2022

1 Building and Solving the model

Please refer to handwritten solution for the mathematical formulation of the problem

[103]:
```
import pandas as pd
import gurobipy as gp
import numpy as np

# Create dataframe with the nutrition data for each fruit

nutrition = pd.DataFrame(np.array([
    [52,89,57,147,53], #Calories
    [14,23,14,27,13], #Carbohydrates
    [2.4,2.6,2.4,3.8,1.8], #Fibre
    [54,64,54,44,681], #Vitamin A
    [4.6, 8.7, 9.7, 19.7, 26.7] #Vitamin C
]),
     columns = ['Apples', 'Bananas', 'Blueberries', 'Durians', 'Tangerines'],
    index = ['Calories', 'Carbohydrates', 'Fibre', 'Vitamin A', 'Vitamin C'])

nutrition
```

[103]:

	Apples	Bananas	Blueberries	Durians	Tangerines
Calories	52.0	89.0	57.0	147.0	53.0
Carbohydrates	14.0	23.0	14.0	27.0	13.0
Fibre	2.4	2.6	2.4	3.8	1.8
Vitamin A	54.0	64.0	54.0	44.0	681.0
Vitamin C	4.6	8.7	9.7	19.7	26.7

[104]:
```
# Create dataframe with the minimum and max allowed quantities for each nutrient

quantity = pd.DataFrame(np.array([
    [500,3000], #Calories
    [50,400], #Carbohydrates
    [20,30], #Fibre
    [2000,3500], #Vitamin A
    [75,150] #Vitamin C
]),
```

```
    columns = ['Min Reqd', 'Max Allowed'],
    index = ['Calories', 'Carbohydrates', 'Fibre', 'Vitamin A', 'Vitamin C'])

quantity
```

[104]:

	Min Reqd	Max Allowed
Calories	500	3000
Carbohydrates	50	400
Fibre	20	30
Vitamin A	2000	3500
Vitamin C	75	150

[105]:

```
# Create dataframe with the price for each fruit

price = pd.DataFrame(np.transpose(np.array([
    [0.5,0.3,2.5,10,0.5]
    ])),
    columns = ['Price per 100g'],
    index = ['Apples', 'Bananas', 'Blueberries', 'Durians', 'Tangerines'])

price
```

[105]:

	Price per 100g
Apples	0.5
Bananas	0.3
Blueberries	2.5
Durians	10.0
Tangerines	0.5

[106]:

```
# Defining other necessary variables

num_nutrient,num_fruits = nutrition.shape
```

[107]:

```
nutrition.index[0]
```

[107]: 'Calories'

[108]:

```
# Setting up the model

OptimalDiet = gp.Model("Diet")

# Creating the variables
x = OptimalDiet.addVars(num_fruits, name = 'FruitQuantity(in100g)')

# Creating the objective
OptimalDiet.setObjective(gp.quicksum(
```

```
                                (price.iloc[i,0] * x[i]) for i in␣
 ↪range(num_fruits)),
                                 gp.GRB.MINIMIZE)

# Adding Constraints
#Constraints for Max Allowed
for nutrient in range(num_nutrient):
    OptimalDiet.addConstr(gp.quicksum(x[i]*nutrition.iloc[nutrient, i] for i in␣
 ↪range(num_fruits)) <= quantity.iloc[nutrient, 1],
                           nutrition.index[nutrient] + ' max')

#Constraints for Min Allowed
for nutrient in range(num_nutrient):
    OptimalDiet.addConstr(gp.quicksum(x[i]*nutrition.iloc[nutrient, i] for i in␣
 ↪range(num_fruits)) >= quantity.iloc[nutrient, 0],
                          nutrition.index[nutrient] + ' min')

# Solving the model

OptimalDiet.optimize()
```

```
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 10 rows, 5 columns and 50 nonzeros
Model fingerprint: 0xb82df2e5
Coefficient statistics:
  Matrix range     [2e+00, 7e+02]
  Objective range  [3e-01, 1e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+01, 4e+03]
Presolve removed 5 rows and 0 columns
Presolve time: 0.01s
Presolved: 5 rows, 10 columns, 30 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   9.562500e+01   0.000000e+00      0s
       3    2.9998792e+00   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.01 seconds (0.00 work units)
Optimal objective  2.999879183e+00
```

[109]:
```
print("\n Optimal solution:")
for v in OptimalDiet.getVars():
    print(v.VarName, v.x)

print("\n Optimal cost:")
print('Obj:', OptimalDiet.objVal)
```

```
#  Print optimal dual solutions
print("\n Dual solutions:")
for d in OptimalDiet.getConstrs():
    print('%s:\t %g %g %g %g' % (d.ConstrName, d.Pi, d.RHS, d.SARHSLow, d.
 ↪SARHSUp))
```

```
 Optimal solution:
FruitQuantity(in100g)[0] 0.0
FruitQuantity(in100g)[1] 6.052917723812976
FruitQuantity(in100g)[2] 0.0
FruitQuantity(in100g)[3] 0.0
FruitQuantity(in100g)[4] 2.3680077322701463

 Optimal cost:
Obj: 2.9998791832789657

 Dual solutions:
Calories max:    0 3000 664.214 inf
Carbohydrates max:       0 400 170.001 inf
Fibre max:       0 30 20 inf
Vitamin A max:   0 3500 2000 inf
Vitamin C max:   0 150 115.886 inf
Calories min:    0 500 -inf 664.214
Carbohydrates min:       0 50 -inf 170.001
Fibre min:       0.104084 20 15.249 30
Vitamin A min:   0.000459104 2000 741.016 3050.45
Vitamin C min:   0 75 -inf 115.886
```

2 Interpretation of Results

As per the results, the optimal quantity of each of the fruits is
`Apple: 0 g  Bananas: 605.29g  Blueberries: 0g  Durians: 0g  Tangerines: 236.80g`

For these quantities, the cost comes to `$2.99`

This combination of fruits is the cheapest possible combination to meet the given nutrition requirements. Further analysis of the dual solutions suggests that the nutritional value provided by this combination of fruits is:
`Calories: 664.21  Carbohydrates: 170.001g  Fibre: 20g  Vitamin A: 2000IU  Vitamin C: 115.886mg`

Furthermore, we also observe that the binding constraints are the ones for minimum amount of Fibre and minimum amount of Vitamin A, whereas for the other nutrients, the value lies in between the minimum and maximum quantities required. From that, we can interpret the following:

1. Increasing the Max allowed quantity for Calories will not change the solutions, however decreasing it to anything less than 664.214 will lead to a change in the optimal solution

2. Increasing the Max allowed quantity for Carbohydrates will not change the solutions, however decreasing to anything less than 170.001g will lead to a change in the optimal solution
3. Increasing the Max allowed quantity for Fibre will not change the solutions, however decreasing to anything less than 20g will lead to a change in the optimal solution
4. Increasing the Max allowed quantity for Vitamin A will not change the solutions, however decreasing to anything less than 2000 IU will lead to a change in the optimal solution
5. Increasing the Max allowed quantity for Vitamin C will not change the solutions, however decreasing to anything less than 115.886mg will lead to a change in the optimal solution

We notice that the lowest allowed values that would not lead to a change in the solution for the constraints corresponding to the maximum quantity of a particular nutrient is the same as the current value of that nutrient provided by the solutions. This is expected as none of these 5 constraints are binding. Continuing our interpretation,

6. Decreasing the Min required quantity of calories will not change the solution, however, increasing it to anything more than 664.214 will lead to a change in the optimal solution
7. Decreasing the Min required quantity of carbohydrates will not change the solution, however, increasing it to anything more than 170.001g will lead to a change in the optimal solution
8. Decreasing the Min required quantity of Vitamin C will not change the solution, however, increasing it to anything more than 115.886mg will lead to a change in the optimal solution

For these three minimum constraints, we again notice that the highest allowed values without causing a change in the solution are the same as the current nutritional value of the solution. This is expected for these three minimum constraints as they are not binding. However, for the remaining two nutrients, we observe the following

9. For Fibre, we observe that an increase or decrease of 1g in the minimum required quantity causes a corresponding increase or decrease, respectively, of `$0.10` in the cost of the combination of fruits. This suggests that changing the constraint for minimum amount of Fibre leads to a change in the optimal solution, corresponding to `$0.10` worth of fruits. However, this only holds valid for as long as the value of minimum required quantity of Fibre is between 15.249g and 30g

10. For Vitamin A, we observe that an increase or decrease of 1IU in the minimum required quantity causes a corresponding increase or decrease, respectively, of `$0.0004` in the cost of the combination of fruits. This suggests that changing the constraint for minimum amount of Fibre leads to a change in the optimal solution, corresponding to `$0.0004` worth of fruits. However, this only holds valid for as long as the value of minimum required quantity of Vitamin A is between 741.016IU and 3050.45IU

Since the minimum required quantity constraints for Fibre and Vitamin A are binding, this is expected as relaxing or tightening these constraints will lead to a change in the optinal solution, implying a change in both, the cheapest combination of fruits as well as in the cost of the combination. In the case of Vitamin A however, there is a caveat the change corresponding to one unit is very small and hence might not be significant

$\therefore A, B_A, B_1, D, T \geq 0$

5.b With the addition of fat as a nutrient, we have 2 more constraints, namely

$0.2A + 0.3B_A + 0.3B_1 + 5D + 0.3T \leq 10$ [Max fat]

$0.2A + 0.3B_A + 0.3B_1 + 5D + 0.3T \geq 0$ [Min fat]

Under the current solution, the amount of fat in the diet is

$0.3 \times 6.0529 + 0.3 \times 2.3680 = 2.526$

$\therefore$ The current optimal solution still lies within the feasible region. This implies that adding the constraint will <u>not</u> affect our optimal solution.

Question 6

September 25, 2022

1 Building the base model

[23]:
```
from gurobipy import *
import numpy as np

#########Parameters Set-up############

#the vector of prices
price = np.array([60, 54, 48, 36])
#the vector of demands
demand = np.array([125, 162.5, 217.5, 348.8])
#salvage value
s = 25
#total number of inventory
I = 2000
#Time horizon
T = 15
#full price week
full_price_week = 1

#number of price levels
N = len(price)
```

[24]:
```
#########Model Set-up###############
#########Model Set-up Using Function###############
def model_setup():

    m = Model("Retail")

    # number of weeks to offer price level i
    x = m.addVars(N, name = "x")

    # set objective
    m.setObjective( quicksum(price[i]*demand[i]*x[i] for i in range(N)) +
                    s*(I - quicksum(demand[i]*x[i] for i in range(N))), GRB.
 ↪MAXIMIZE)
```

```python
    # capacity constraint:
    m.addConstr( quicksum(demand[i]*x[i] for i in range(N)) <= I , "capacity")

    # time constraint:
    m.addConstr( quicksum(x[i] for i in range(N)) <= T , "time")

    # full price constraint:
    m.addConstr( x[0] >= full_price_week , "full_price")

    return m

m = model_setup()

# Solving the model
m.optimize()

#  Print optimal solutions and optimal value
print("\n Optimal solution:")
for v in m.getVars():
    print(v.VarName, v.x)

print("\n Optimal profit:")
print('Obj:', m.objVal)

print("\n Dual solutions:")
for d in m.getConstrs():
    print('%s %g %g %g %g' % (d.ConstrName, d.Pi, d.RHS, d.SARHSLow, d.SARHSUp))
```

```
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 3 rows, 4 columns and 9 nonzeros
Model fingerprint: 0x3e591d83
Coefficient statistics:
  Matrix range     [1e+00, 3e+02]
  Objective range  [4e+03, 5e+03]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 1 rows and 0 columns
Presolve time: 0.01s
Presolved: 2 rows, 4 columns, 8 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.2000000e+05   1.587200e+01   0.000000e+00      0s
       2    1.1675000e+05   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.01 seconds (0.00 work units)
Optimal objective  1.167500000e+05
```

```
 Optimal solution:
x[0] 11.666666666666666
x[1] 3.333333333333334
x[2] 0.0
x[3] 0.0

 Optimal profit:
Obj: 116750.0

 Dual solutions:
capacity 9 2000 1875 2400
time 3250 15 12.5385 16
full_price 0 1 -inf 11.6667
```

2 Scenario 1: No discount till week 4

[25]:
```python
# Resetting the value of the variable
full_price_week = 4

# Setup the new model
m = model_setup()

# Solving the model
m.optimize()

#  Print optimal solutions and optimal value
print("\n Optimal solution:")
for v in m.getVars():
    print(v.VarName, v.x)

print("\n Optimal profit:")
print('Obj:', m.objVal)
```

```
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 3 rows, 4 columns and 9 nonzeros
Model fingerprint: 0xc74eaf63
Coefficient statistics:
  Matrix range     [1e+00, 3e+02]
  Objective range  [4e+03, 5e+03]
  Bounds range     [0e+00, 0e+00]
  RHS range        [4e+00, 2e+03]
Presolve removed 1 rows and 0 columns
Presolve time: 0.01s
Presolved: 2 rows, 4 columns, 8 nonzeros
```

```
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.2000000e+05   1.587200e+01   0.000000e+00      0s
       2    1.1675000e+05   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.01 seconds (0.00 work units)
Optimal objective  1.167500000e+05

 Optimal solution:
x[0] 11.666666666666666
x[1] 3.333333333333334
x[2] 0.0
x[3] 0.0

 Optimal profit:
Obj: 116750.0
```

6.a) Changing the constraint so that the retailer is allowed to discount only from the 4th week does not affect the optimal solution. This is expected because the optimal solution suggests discounting only after the 11th week.

Hence the retailer will not change his markdown strategy

3 Scenario 2: Low demand due to bad weather

By sticking to his original strategy, the retailer loses out on 100 units worth of revenue in the first week but gain 100 units worth of revenue by salvaging the leftover stock via outlet. Assuming that the demand for the remaining weeks does not change, the revenue of the retailer is his `optimal revenue - 100 * (60-25) = $113250`

We now explore the alternative of identifying a new markdown strategy from week 2.

The retailer's revenue in the first week is `25 * 60` = **$1500**

[26]:
```
# Setting new values for variables and resetting the original values for those␣
↪changed in Scenario 2

# New Inventory
I = 1975
# New Time Horizon
T = 14
# Since we are starting from week 2, there is no longer a limit on when we the␣
↪retailer can start marking down prices
full_price_week = 0

# Setup the new model
m = model_setup()

# Solving the model
m.optimize()
```

```python
#  Print optimal solutions and optimal value
print("\n Optimal solution:")
for v in m.getVars():
    print(v.VarName, v.x)

print("\n Optimal profit:")
print('Obj:', m.objVal)
```

```
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 3 rows, 4 columns and 9 nonzeros
Model fingerprint: 0xdbdf791e
Coefficient statistics:
  Matrix range     [1e+00, 3e+02]
  Objective range  [4e+03, 5e+03]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+01, 2e+03]
Presolve removed 1 rows and 0 columns
Presolve time: 0.01s
Presolved: 2 rows, 4 columns, 8 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.1850000e+05   2.868000e+01   0.000000e+00      0s
       2    1.1265000e+05   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.01 seconds (0.00 work units)
Optimal objective  1.126500000e+05

 Optimal solution:
x[0] 8.0
x[1] 6.0
x[2] 0.0
x[3] 0.0

 Optimal profit:
Obj: 112650.0
```

Hence, the new strategy suggests that despite the demand shock in the first week, the retailer should not change the price of the product for the first 9 weeks (instead of 11.67 as per the original strategy). For the remaining 6 weeks, the retailer sells his inventory at the first markdown price (`$54`) and clears out his inventory

Therefore, the total profit in this case is `$112650 + $1500 = $114150`

The improvement in revenue in this case is `$114150 - $113250 = $900`. Therefore, by changing their strategy after the first week of poor sales, the retailer manages to earn an extra `$900`

4 Scenario 3: Change in markdown policy

[28]:
```
# Setting new values for variables and resetting the original values for those␣
↪changed in Scenario 2

#the vector of prices
price = np.array([60, 58, 56, 54, 52, 50, 48, 46, 44, 42, 40, 38, 36])
#the vector of demands
demand = np.array([125, 137.5, 150, 162.5, 180.8, 199.1, 217.5, 239.4, 261.3,␣
↪283.2, 305.1, 327, 348.8])
#total number of inventory
I = 2000
#Time horizon
T = 15
#full price week
full_price_week = 1
#number of price levels
N = len(price)

m = model_setup()

# Solving the model
m.optimize()

#  Print optimal solutions and optimal value
print("\n Optimal solution:")
for v in m.getVars():
    print(v.VarName, v.x)

print("\n Optimal profit:")
print('Obj:', m.objVal)

print("\n Dual solutions:")
for d in m.getConstrs():
    print('%s %g %g %g %g' % (d.ConstrName, d.Pi, d.RHS, d.SARHSLow, d.SARHSUp))
```

```
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 3 rows, 13 columns and 27 nonzeros
Model fingerprint: 0xe0783e89
Coefficient statistics:
  Matrix range     [1e+00, 3e+02]
  Objective range  [4e+03, 5e+03]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 1 rows and 0 columns
Presolve time: 0.01s
```

```
Presolved: 2 rows, 13 columns, 26 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.2000000e+05   1.587200e+01   0.000000e+00      0s
       2    1.1725000e+05   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.02 seconds (0.00 work units)
Optimal objective  1.172500000e+05

 Optimal solution:
x[0] 5.0
x[1] 10.0
x[2] 0.0
x[3] 0.0
x[4] 0.0
x[5] 0.0
x[6] 0.0
x[7] 0.0
x[8] 0.0
x[9] 0.0
x[10] 0.0
x[11] 0.0
x[12] 0.0

 Optimal profit:
Obj: 117250.0

 Dual solutions:
capacity 13 2000 1875 2050
time 2750 15 14.6364 16
full_price 0 1 -inf 5
```

The more flexible markdown policy leads to increase in revenue from `$116750` to `117250`, a net improvement of `$500` over the benchmark

The new markdown policy suggest selling the product at the non discounted price of `$60` for the first 5 weeks, and then sell at the first discounted price of `$58` for the remaining 10 weeks to clear out inventory

The new markdown policy appears to perform better than the benchmark discussed at the beginning. However, an analysis of the Dual solutions shows that the marginal increase in revenue by increasing capacity increases from $9 to $13, however, at the same time, the marginal increase in revenue by increasing the time period by 1 week falls from $3250 to $2750. There are also changes in some of the allowed upper and lower limits of change in the constraints (resources) without changing the optimal solution.