IE 407 Fundamentals of OR

Spring 2024

Assignment 3

Elif Sena Kuru / 2448678
Muhammed Batuhan Berk / 2309748

Question 1:

a)

Decision Variables:

- c: Number of PCs produced and sold.
    - The quantity of PC CPUs that the company decides to produce and sell in a week.
- t: Number of Ts produced and sold.
    - The quantity of Tablet CPUs that the company decides to produce and sell in a week.
- m: Number of MPs produced.
    - The quantity of microprocessors that the company decides to produce in a week.

Objective Function:

max $Z = 50c + 30t - 2m$

The objective function aims to maximize the profit Z:

- 50c: Revenue from selling c units of PCs, where each PC CPUs sells for $50.
- 30t: Revenue from selling t units of Ts, where each Tablet CPUs sells for $30.
- 2m: Cost of producing m units of microprocessors, where each microprocessor costs $2 to produce.

Constraints:

1. **Production time constraint:**

$m + 5c + 2t <= 120$

This constraint ensures that the total production time for microprocessors, PCs, and Ts does not exceed the available 120 working hours per week.

- m: Each microprocessor requires 1 hour to produce.
- 5c: Each PC requires 5 hours to produce.
- 2t: Each T requires 2 hours to produce.

2. **MP stock constraint:**

$8c + 4t <= m + 80$ which can be written as:

$8c + 4t -m <= 80$

- This constraint ensures that the number of microprocessors used in the production of PCs and Ts does not exceed the available stock. It includes the microprocessors produced this week and the initial stock of 80 units.

- 8c: Each PC requires 8 microprocessors.
- 4t: Each Tablet CPU requires 4 microprocessors.
- m: Subtracting the number of microprocessors produced this week since they are added to the stock.

3. **Non-negativity constraints:**

$c >= 0$

$t >= 0$

$m >= 0$

- These constraints ensure that the number of PCs, Ts, and microprocessors produced cannot be negative, as it is not feasible to produce a negative quantity of any product.

b)

Question 1:

part b) max $z = 50c + 30t - 2m$

$8c + 4t - m \leq 80$ (stock constraint)

$m + 5c + 2t \leq 120$ (time constraint)

	x_1	x_2	x_3	S_1	S_2	RHS	
z	-50	-30	2	0	0	0	
S_1	8	4	-1	1	0	80	$\frac{80}{8} = 10$
S_2	5	2	1	0	1	120	$\frac{120}{5} = 24$

entering variable is x_1 and leaving variable is S_1.
Pivot element is 8.

since -50 is the most negative.

since the ratio is low.

	x_1	x_2	x_3	S_1	S_2	RHS	
z	0	-5	-17/4	25/4	0	500	
x_1	1	1/2	-1/8	1/8	0	10	$\frac{10}{\frac{1}{2}} = 20$
S_2	0	-1/2	13/8	-5/8	1	70	$\frac{70}{-1/2} = -140$

entering variable is x_2 and leaving variable is x_1
Pivot element is 8.

since -5 is the most negative.

since the ratio is low.

	x_1	x_2	x_3	S_1	S_2	RHS	
z	10	0	-11/2	15/2	0	600	
x_2	2	1	-1/4	1/4	0	20	$\frac{20}{-1/4} = -80$
S_2	1	0	3/2	-1/2	1	60	$\frac{60}{3/2} = 40$

entering variable is x_3 and leaving variable is S_2
Pivot element is 8.

since -11/2 is the most negative.

since the ratio is low.

	x_1	x_2	x_3	S_1	S_2	RHS
z	41/3	0	0	17/3	11/3	$\frac{2680}{3}$
x_2	13/6	1	0	1/6	-1/6	$\frac{100}{3}$
x_3	2/3	0	1	-1/3	2/3	$\frac{160}{3}$

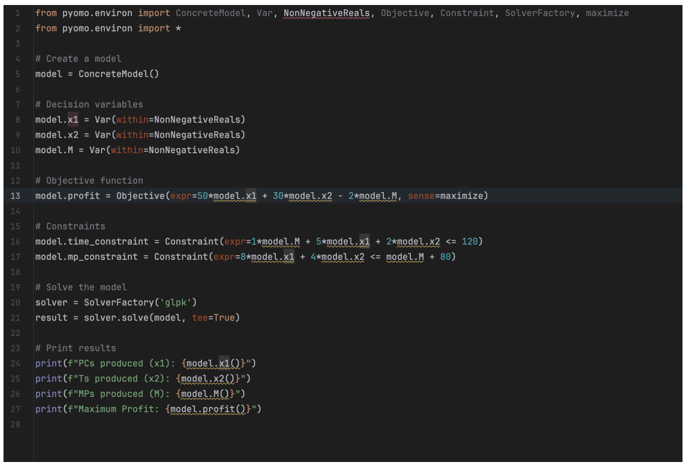
Since there is no negative cell in z row, optimal solution is found.

$z = \frac{2680}{3} = 893.33$

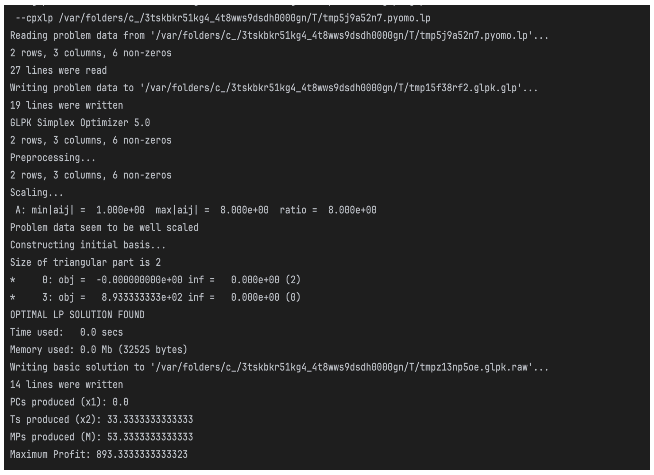
$x_1 = 0$ $\quad x_3 = 160/3 = 53.33$

$x_2 = 100/3 = 33.33$ $\quad S_1 = 0$ $\quad S_2 = 0$

c)



```python
from pyomo.environ import ConcreteModel, Var, NonNegativeReals, Objective, Constraint, SolverFactory, maximize
from pyomo.environ import *

# Create a model
model = ConcreteModel()

# Decision variables
model.x1 = Var(within=NonNegativeReals)
model.x2 = Var(within=NonNegativeReals)
model.M = Var(within=NonNegativeReals)

# Objective function
model.profit = Objective(expr=50*model.x1 + 30*model.x2 - 2*model.M, sense=maximize)

# Constraints
model.time_constraint = Constraint(expr=1*model.M + 5*model.x1 + 2*model.x2 <= 120)
model.mp_constraint = Constraint(expr=8*model.x1 + 4*model.x2 <= model.M + 80)

# Solve the model
solver = SolverFactory('glpk')
result = solver.solve(model, tee=True)

# Print results
print(f"PCs produced (x1): {model.x1()}")
print(f"Ts produced (x2): {model.x2()}")
print(f"MPs produced (M): {model.M()}")
print(f"Maximum Profit: {model.profit()}")
```



```
 --cpxlp /var/folders/c_/3tskbkr51kg4_4t8wws9dsdh0000gn/T/tmp5j9a52n7.pyomo.lp
Reading problem data from '/var/folders/c_/3tskbkr51kg4_4t8wws9dsdh0000gn/T/tmp5j9a52n7.pyomo.lp'...
2 rows, 3 columns, 6 non-zeros
27 lines were read
Writing problem data to '/var/folders/c_/3tskbkr51kg4_4t8wws9dsdh0000gn/T/tmp15f38rf2.glpk.glp'...
19 lines were written
GLPK Simplex Optimizer 5.0
2 rows, 3 columns, 6 non-zeros
Preprocessing...
2 rows, 3 columns, 6 non-zeros
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  8.000e+00  ratio =  8.000e+00
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 2
*     0: obj =  -0.000000000e+00 inf =   0.000e+00 (2)
*     3: obj =   8.933333333e+02 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.0 Mb (32525 bytes)
Writing basic solution to '/var/folders/c_/3tskbkr51kg4_4t8wws9dsdh0000gn/T/tmpz13np5oe.glpk.raw'...
14 lines were written
PCs produced (x1): 0.0
Ts produced (x2): 33.3333333333333
MPs produced (M): 53.3333333333333
Maximum Profit: 893.3333333333323
```

The first screenshot displays the definition and execution of our linear programming model using Pyomo in Python. The model aims to maximize the profit of producing and selling PC CPUs and Tablet CPUs while considering the cost of producing microprocessors (MPs). This code includes the decision variables, objective function and constraints mentioned in part a.

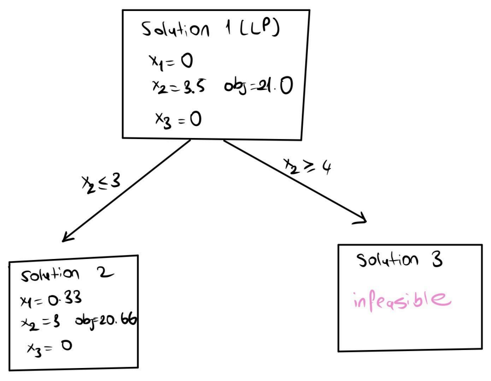
The second screenshot shows the output from the GLPK solver after solving the Pyomo model.

The solver finds the optimal solution for the problem. The output specifies the optimal number of PC CPUs (0 units), Tablet CPUs (33.333 units), and MPs (53.333 units) to be produced. Moreover, the calculated maximum profit from the optimal production plan is 893.33. So, our solution is:

- c : 0
- t : 33.33
- m: 53.33
- Z : 893.33

Question 2:

a)



root

1						
B	x_1	x_2	x_3	S_1	S_2	RHS
S_1	6	4	2	1	0	14
S_2	4	2	4	0	1	22
z	-8	-6	-2	0	0	0

$\frac{14}{6} = 2.33$ leaving variable is S_1

$\frac{22}{4} = 5.5$

entering variable is x_1

2						
B	x_1	x_2	x_3	S_1	S_2	RHS
x_1	1	0.6667	0.33	0.1667	0	2.333
S_2	0	-0.667	2.667	-0.6667	1	12.6667
z	0	-0.6667	0.667	1.333	0	18.6667

$\frac{2.333}{0.6667} = 3.5$ leaving variable is x_1.

entering variable is x_2

3						
B	x_1	x_2	x_3	S_1	S_2	RHS
x_2	1.5	1	0.5	0.25	0	3.5
S_2	1	0	3	-0.5	1	15
Z	1	0	1	1.5	0	21

max z = 21
$x_1 = 0$
$x_2 = 3.5$
$x_3 = 0$

Branch 1

B	x_1	x_2	x_3	S_1	S_2	S_3	RHS
S_1	6	4	2	1	0	0	14
S_2	4	2	4	0	1	0	22
S_3	0	1	0	0	0	1	3
z	-8	-6	-2	0	0	0	0

$\frac{14}{6} = 2.333$

$\frac{22}{4} = 5.5$

entering var → x_1
leaving var → S_1

B	x_1	x_2	x_3	S_1	S_2	S_3	RHS
x_1	1	0.6667	0.333	0.1667	0	0	2.33
S_2	0	-0.6667	2.6667	-0.6667	1	0	12.6667
S_3	0	1	0	0	0	1	3
z	0	-0.6667	0.6667	1.333	0	0	18.6667

$\frac{2.33}{0.6667} = 3.5$

$\frac{3}{1} = 3$

entering var → x_2
leaving var → S_3

B	x_1	x_2	x_3	S_1	S_2	S_3	RHS
x_1	1	0	0.333	0.1667	0	-0.6667	0.333
S_2	0	0	2.6667	-0.6667	1	0.6667	14.6667
x_2	0	1	0	0	0	1	3
z	0	0	0.6667	1.333	0	0.6667	20.6667

optimal solution
$x_1 = 0.333$
$x_2 = 3$
$x_3 = 0$
max z = 20.6667

Branch 2

entering variable

B	x_1	x_2	x_3	S_1	S_2	S_3	A_1	RHS
S_1	6	4	2	1	0	0	0	14
S_2	4	2	4	0	1	0	0	22
A_1	0	1	0	0	0	-1	1	4
z	-8	-M-6	-2	0	0	M	0	-4M

→ $\frac{14}{4} = 3.5$

→ $\frac{22}{2} = 11$

→ 4

leaving variable

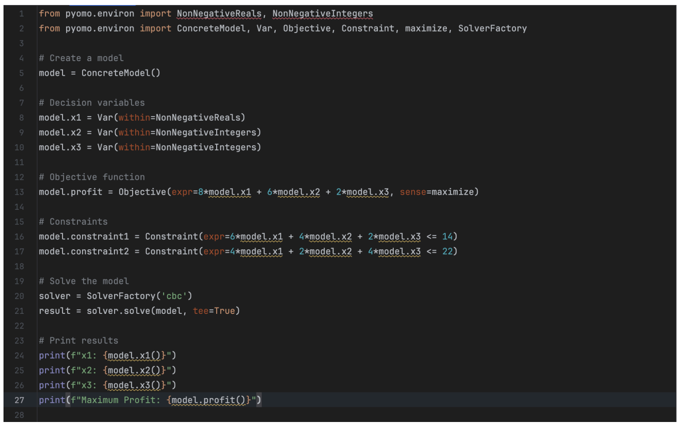
pinks do not matter here!

B	x_1	x_2	x_3	S_1	S_2	S_3	A_1	RHS
x_2	1.5	1	0.5	0.25	0	0	0	3.5
S_2	1	0	3	-0.5	1	0	0	15
A_1	-1.5	0	-0.5	-0.25	0	-1	1	0.5
z	1.5M+1	0	$\frac{1}{2}$M+3	$\frac{1}{4}$M+1.5	0	M	0	0.5M+21

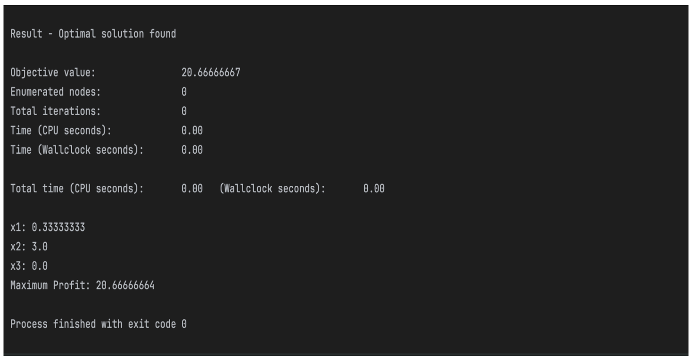
here
$x_1 = 0$
$x_2 = 3.5$
$x_3 = 0$
max z = 21

Since in branch 2 $x_2 \ge 4$, this is infeasible!

b)



```python
from pyomo.environ import NonNegativeReals, NonNegativeIntegers
from pyomo.environ import ConcreteModel, Var, Objective, Constraint, maximize, SolverFactory

# Create a model
model = ConcreteModel()

# Decision variables
model.x1 = Var(within=NonNegativeReals)
model.x2 = Var(within=NonNegativeIntegers)
model.x3 = Var(within=NonNegativeIntegers)

# Objective function
model.profit = Objective(expr=8*model.x1 + 6*model.x2 + 2*model.x3, sense=maximize)

# Constraints
model.constraint1 = Constraint(expr=6*model.x1 + 4*model.x2 + 2*model.x3 <= 14)
model.constraint2 = Constraint(expr=4*model.x1 + 2*model.x2 + 4*model.x3 <= 22)

# Solve the model
solver = SolverFactory('cbc')
result = solver.solve(model, tee=True)

# Print results
print(f"x1: {model.x1()}")
print(f"x2: {model.x2()}")
print(f"x3: {model.x3()}")
print(f"Maximum Profit: {model.profit()}")
```



```
Result - Optimal solution found

Objective value:                20.66666667
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             0.00
Time (Wallclock seconds):       0.00

Total time (CPU seconds):       0.00   (Wallclock seconds):       0.00

x1: 0.33333333
x2: 3.0
x3: 0.0
Maximum Profit: 20.66666664

Process finished with exit code 0
```

The first screenshot displays our Pyomo model for solving the MIP problem. The model, objective function and constraints given in the problem are defined. The model is solved using the CBC solver. Also, the code prints the results.

In the second screenshot, the solver indicates that an optimal solution has been found with an objective value of 20.6666667.

Solution details are:

- x1 = 0.33333333
- x2 = 3.0
- x3 = 0.0

- The maximum profit is 20.6666667.

The solver output provides additional performance metrics, confirming the efficiency of the solution process.