Assignment 2 - Exercise 1

Bruno Kiyoshi Ynumaru - 201805995

Duas máquinas produzem conjuntamente dois produtos. Cada unidade do primeiro produto requer 3 horas da máquina 1 e 2 horas da máquina 2. Cada unidade do segundo produto requer 2 horas da máquina 1 e 3 horas da máquina 2. A máquina 1 está disponível por 8 horas diárias e a máquina 2 apenas 7 horas diárias. O lucro por unidade vendida á 16 unidades monetárias para o primeiro produto e 10 para o segundo. Ambos os produtos possuem restrição quanto à produção parcial: a quantidade de cada produto produzido por dia precisa ser um múltiplo inteiro de 0.25. Formule um modelo matemático com o objetivo de determinar a quantidade de cada produto a ser produzido por dia que maximize o lucro.

In [1]:

```
import gurobipy as gp
from gurobipy import GRB, Model
```

In [2]:

```
# Create a new model
m = Model("Wyndor_Glass")
```

```
Restricted license - for non-production use only - expires 2022-01-13
```

In [4]:

```
# Create variables
u1 = m.addVar(lb=0, vtype=GRB.INTEGER, name="u1") # lots of product 1
u2 = m.addVar(lb=0, vtype=GRB.INTEGER, name="u2") # lots of product 2
```

In [5]:

```
# Set objective
stdlot = 0.25 # standard lot [units of product / lot]
r1 = 16 * stdlot # revenue per lot of product 1
r2 = 10 * stdlot# revenue per lot of product 2
m.setObjective(r1 * u1 + r2 * u2, GRB.MAXIMIZE)
```

In [6]:

```
# Add constraints
t11 = 3 * stdlot # machine 1 usage time for making 1 lot of product 1
t21 = 2 * stdlot# machine 2 usage time for making 1 lot of  product 1
t12 = 2 * stdlot# machine 1 usage time for making 1 lot of  product 2
t22 = 3 * stdlot# machine 2 usage time for making 1 lot of  product 2
av1 = 8 # machine 1 total available time per day
av2 = 7 # machine 2 total available time per day
m.addConstr(u1 * t11 + u2 * t12 <= av1, 'C0')
m.addConstr(u1 * t21 + u2 * t22 <= av2, 'C1')
m.optimize()
```

```
Gurobi Optimizer version 9.1.1 build v9.1.1rc0 (win64)
Thread count: 2 physical cores, 4 logical processors, using up to 4 thread
s
Optimize a model with 2 rows, 2 columns and 4 nonzeros
Model fingerprint: 0x9b62e511
Variable types: 0 continuous, 2 integer (0 binary)
Coefficient statistics:
  Matrix range     [5e-01, 8e-01]
  Objective range  [3e+00, 4e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [7e+00, 8e+00]
Found heuristic solution: objective 42.5000000
Presolve time: 0.02s
Presolved: 2 rows, 2 columns, 4 nonzeros
Variable types: 0 continuous, 2 integer (0 binary)

Root relaxation: cutoff, 0 iterations, 0.00 seconds

Explored 0 nodes (0 simplex iterations) in 0.06 seconds
Thread count was 4 (of 4 available processors)

Solution count 1: 42.5

Optimal solution found (tolerance 1.00e-04)
Best objective 4.250000000000e+01, best bound 4.250000000000e+01, gap 0.00
00%
```

In [14]:

```
for v in m.getVars():
    print(f'{v.varName}, {v.x} lots, {v.x * stdlot} units')

print(f'Obj: $ {m.objVal}')
```

```
u1, 10.0 lots, 2.5 units
u2, 1.0 lots, 0.25 units
Obj: $ 42.5
```