Assignment 2 - Exercise 2

Bruno Kiyoshi Ynumaru - 201805995

Um centro de reciclagem industrial usa dois tipos de sucata de alumÍnio, A e B, para produzir uma liga metálica. A sucata A contém 6% alumínio, 3% silício, e 4% carbono. A sucata B contém 3% alumínio, 6% silício e 3% carbono. Os custos por tonelada para a sucata A e B são 100 e 80 unidades monetárias, respectivamente. As especificações da liga metálica requer que (1) o conteúdo de alumínio deve ser pelo menos 3% e no máximo 6%, (2) o conteúdo de silício deve ser pelo menos 3% e no máximo 5%, e (3) o conteúdo de carbono deve ser pelo menos 3% e no máximo 7%. Formule um modelo matemático com o objetivo de determinar a mistura ótima de sucatas A e B para minimizar os custos da produção de 1000 toneladas desta liga metálica.

In [2]:

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

In [3]:

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

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

In [4]:

```
# Create variables
mA = m.addVar(lb=0, vtype=GRB.CONTINUOUS, name="tons A") # [tons of A]
mB = m.addVar(lb=0, vtype=GRB.CONTINUOUS, name="tons B") # [tons of B]
```

In [5]:

```
# Set objective
cA = 100 # [$/ton A]
cB = 80 # [$/ton B]
m.setObjective(cA * mA + cB * cB, GRB.MINIMIZE)
```

In [12]:

```
# Add constraints
total_prod = 1000 # [tons of product]
minAl = 0.03
maxAl = 0.06
minSi = 0.03
maxSi = 0.05
minC = 0.03
maxC = 0.07
Al_A = 0.06
Al_B = 0.03
Si_A = 0.03
Si_B = 0.06
C_A = 0.04
C_B = 0.03

def calculate_concentration(mA, xA, mB, xB, total_prod):
    return (mA * xA + mB * xB) / (total_prod)

m.addConstr(calculate_concentration(mA, Al_A, mB, Al_B, total_prod) >= minAl, 'C0')
m.addConstr(calculate_concentration(mA, Al_A, mB, Al_B, total_prod) <= maxAl, 'C1')
m.addConstr(calculate_concentration(mA, Si_A, mB, Si_B, total_prod) >= minSi, 'C2')
m.addConstr(calculate_concentration(mA, Si_A, mB, Si_B, total_prod) <= maxSi, 'C3')
m.addConstr(calculate_concentration(mA, C_A, mB, C_B, total_prod) >= minC, 'C4')
m.addConstr(calculate_concentration(mA, C_A, mB, C_B, total_prod) <= maxC, 'C5')
m.addConstr(mA + mB == total_prod, 'C6')

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 14 rows, 2 columns and 28 nonzeros
Coefficient statistics:
  Matrix range     [3e-05, 1e+00]
  Objective range  [1e+02, 1e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e-02, 1e+03]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    3.9733333e+04   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds
Optimal objective  3.973333333e+04
```

In [13]:

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

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

```
tons A, 333.33333333333337
tons B, 666.6666666666666
Obj: 39733.333333333336
```

In [ ]:

In [ ]:

```

```

In [ ]:

```

```