PO-202 - Programação Linear

Atividade 2

Grupo:

Gabriel Telles Missailidis

Rafael Silva de Oliveira

Samir Nunes da Silva

Professor:

Luiz Leduino Salles Neto

25/08/2022

Instituto Tecnológico de Aeronáutica – ITA

1 Questão 1

As variáveis de decisão são:

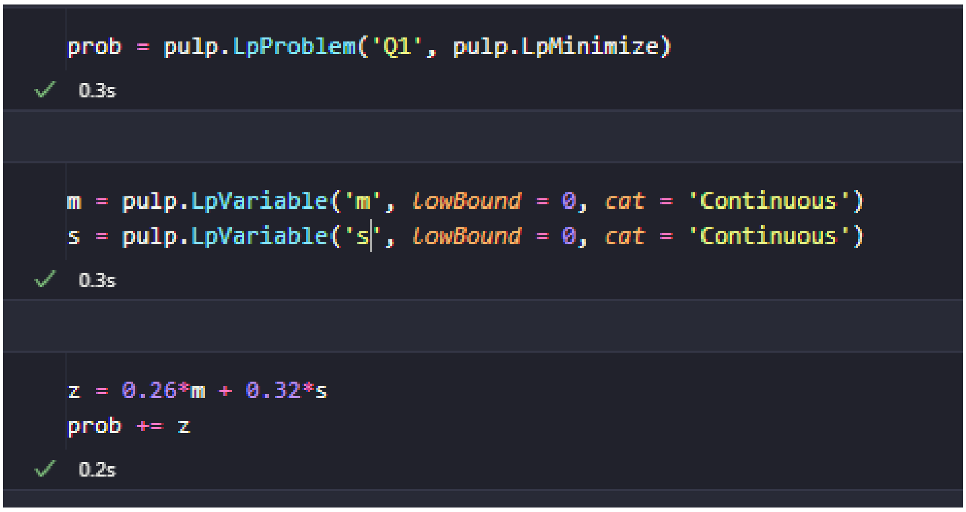
- m: quantidade de ração de milho na ração final
- s: quantidade de ração de farelo de soja na ração final

Ambas as variáveis são contínuas e devem ser maiores ou iguais a zero. Sendo C(m,s) a função de custo, o problema pode ser enunciado da seguinte forma:

$$
\begin{aligned}
Minimizar \quad & C(m,s) = 0,26m + 0,32s \\
sujeito\ a: \quad & 0,07m + 0,21s \geq 0,34 \\
& 0,82m + 0,79s \geq 2,64 \\
& m, s \geq 0
\end{aligned}
$$

sendo a primeira restrição referente à quantidade de proteínas e a segunda à quantidade de carboidratos na ração final.

Utilizando a biblioteca *PuLP* da linguagem *Python*, definimos o problema de otimização, as variáveis e a função objetivo, conforme mostra a Figura 1. Ainda, definiu-se as restrições do problema como na Figura 2.



```
prob = pulp.LpProblem('Q1', pulp.LpMinimize)
✓ 0.3s

m = pulp.LpVariable('m', LowBound = 0, cat = 'Continuous')
s = pulp.LpVariable('s', LowBound = 0, cat = 'Continuous')
✓ 0.3s

z = 0.26*m + 0.32*s
prob += z
✓ 0.2s
```

Figura 1: Modelagem do problema.

```
prot = 0.07*m + 0.21*s
carbo = 0.82*m + 0.79*s
prob += (prot >= 0.34)
prob += (carbo >= 2.64)
```
0.3s

```
prob
```
0.6s

```
Q1:
MINIMIZE
0.26*m + 0.32*s + 0.0
SUBJECT TO
_C1: 0.07 m + 0.21 s >= 0.34

_C2: 0.82 m + 0.79 s >= 2.64

VARIABLES
m Continuous
s Continuous
```

Figura 2: Restrições do problema.

Finalmente, o programa aponta, como otimização, o uso de $2,44kg$ de milho e $0,80kg$ de soja, como indica a Figura 3. Isso representa, aproximadamente, o uso de $75,3\%$ da primeira ração e $24,7\%$ da segunda, resultando num custo final de aproximadamente $R\$0,27$ por kg de ração.

```
optimization_result = prob.solve()
```
0.3s

```
assert optimization_result == pulp.LpStatusOptimal
```
0.2s

```
for var in (m, s):
    print(f'{var.name}:{var.value()}')
```
0.3s

```
m:2.4448246
s:0.80410607
```

Figura 3: Otimização do problema.

2 Questão 2

As variáveis de decisão são:

- x_1: quantidade de depósitos de 1 ano
- x_2: quantidade de depósitos de 2 anos
- x_3: quantidade de depósitos de 3 anos

A fim de linearizar o problema, considerou-se juros simples. Do enunciado, tem-se que o investidor tem R$ 22.000,00 para investir por 5 anos. As possibilidades de investimento são um depósito bancário de 1 ano de duração com retorno de 8%, um depósito de 2 anos com retorno total de 17% ou um título de 3 anos com retorno total de 27%. Note que aplicar primeiro no investimento em x_1 e depois reinvestir o capital inicial em x_2 é exatamente igual a investir primeiro em x_2 e depois em x_1. Ademais, a restrição de só poder investir nos título de 3 anos a partir do segundo ano é irrelevante, pois devido à restrição da quantidade de anos, não é possível investir duas vezes no título de x_3, assim se este for usado basta apenas deixá-lo para o final. (Uma vez que são comutativos!). Note também que a restrição de anos é uma igualdade estrita, pois sempre é vantajoso investir por mais tempo, mesmo que no depósito de apenas 1 ano. Com isso, a função objetiva (lucro) é da forma:

$$f(x_1, x_2, x_3) = 22000 \cdot (0,08x_1 + 0,17x_2 + 0,27x_3)$$

com restrições dadas por:

$$\begin{gathered} x_1 + 2x_2 + 3x_3 = 5 \text{ (em anos)} \\ x_i \geq 0, \ i = 1, 2, 3 \end{gathered}$$

Assim, o problema pode ser enunciado da seguinte forma:

$$\begin{aligned} Maximizar \quad f(x_1, x_2, x_3) = (1,08^{x_1} \cdot 1,17^{x_2} \cdot 1,27^{x_3} - 1) \cdot 22000 \\ sujeito\ a: \qquad x_1 + 2x_2 + 3x_3 = 5 \\ x_1, x_2, x_3 \in Z_+ \end{aligned}$$

conforme pode-se visualizar na Figura 4, que indica a montagem do problema no *Jupyter Notebook* através da biblioteca *PuLP* do *Python*.

```
# definindo o problema como maximização
prob = pulp.LpProblem('Problema_do_investimento', pulp.LpMaximize)
```

```
# definindo as variáveis de decisão
x1 = pulp.LpVariable('Deposito_1', lowBound=0, cat='Integer')
x2 = pulp.LpVariable('Deposito_2', lowBound=0, cat='Integer')
x3 = pulp.LpVariable('Deposito_3', lowBound=0, cat='Integer')
```

```
# definindo a função objetivo
lucro = (0.08*x1 + 0.17*x2 + 0.27*x3) * 22000

# add a função objetivo
prob += lucro
```

```
# restrições
total_anos = x1 + 2*x2 + 3*x3 # ==5
```

```
# add restrições
prob += (total_anos == 5)
```

Figura 4: Modelagem da questão 2 por meio da biblioteca *PuLP* do *Python*

Por fim, resolveu-se o problema através da função *solve()* e escreveu-se os resultados obtidos, como mostra a Figura 5, obtendo-se os valores ótimos de $x_1 = 0$, $x_2 = 1$ e $x_3 = 1$.

$\therefore$ **Resposta:** $(\mathrm{x}_1, x_2, x_3) = (0, 1, 1)$.

```
# verificando se a solução ótima foi encontrada
assert optimization_result == pulp.LpStatusOptimal

# mostrando o resultado
for var in (x1, x2, x3):
    print('A produção ótima deve ser {}: {}'.format(var.name, var.value()))
```

```
A produção ótima deve ser Deposito_1: 0.0
A produção ótima deve ser Deposito_2: 1.0
A produção ótima deve ser Deposito_3: 1.0
```

Figura 5: Obtenção da resposta da questão 2 por meio da biblioteca *PuLP* do *Python*

3 Questão 3

As variáveis de decisão são:

- x: fração empregada da liga 1
- y: fração empregada da liga 2

Ambas as variáveis são contínuas e devem ser maiores ou iguais a 0 e, ao mesmo tempo, menores ou iguais a 1. Sendo C(x,y) a função de custo, o problema pode ser enunciado da seguinte maneira:

$$\begin{aligned} Minimizar \quad & C(m, s) = 190x + 200y \\ sujeito\ a: \quad & x + y = 1 \\ & 3,2 \leq 3x + 4y \leq 3,5 \\ & 1,8 \leq 2x + 1,5y \leq 2,5 \\ & 0,9 \leq x + 1,5y \leq 1,2 \\ & 0 \leq x, y \leq 1 \end{aligned}$$

As restrições do problema são referentes, na ordem em que estão enunciadas, ao percentual de cada liga, à porcentagem de Carbono, à porcentagem de Silicone e à porcentagem de Níquel na liga final. Ainda, note que as restrições $0,9 \leq x + 1.5y$ e $2x + 1,5y \leq 2,5$ são sempre satisfeitas para $0 \leq i \leq 1$, $i = x$ ou y.

Utilizando a biblioteca *PuLP* da linguagem *Python*, definimos o problema de otimização, as variáveis e a função objetivo, conforme mostra a Figura 6. Ainda, definiu-se as restrições do problema como na Figura 7.

```python
# definindo o problema como minimização
prob = pulp.LpProblem('Problema_do_aço', pulp.LpMinimize)
```

```python
# definindo as variáveis de decisão
x = pulp.LpVariable('Liga_1', lowBound=0, upBound=1, cat='Continuous')
y = pulp.LpVariable('Liga_2', lowBound=0, upBound=1, cat='Continuous')
```

```python
# definindo a função objetivo
custo = 190*x+200*y

# add a função objetivo
prob += custo
```

Figura 6: Definição das variáveis e função objetivo.

```python
# restrições
percentual = x + y
carbonoMin = 3*x + 4*y # >= 3.2
carbonoMax = 3*x + 4*y # <= 3.5

niquelMax = x + 1.5*y # <= 1.2
```

```python
# add restrições
prob += (percentual == 1)
prob += (carbonoMin >= 3.2)
prob += (carbonoMax <= 3.5)
prob += (niquelMax <= 1.2)
```

Figura 7: Adicionando as restrições do problema.

Finalmente, a solução ótima apresentada pelo problema é utilizar 80% da liga 1 e 20% da liga 2, obtendo um aço de custo R$192,00 por tonelada, conforme mostra a Figura 8.

```
# escrevendo o prob de otimização linear
print(prob)

# resolvendo o problema
optimization_result = prob.solve()
✓ 0.2s
Problema_do_aço:
MINIMIZE
190*Liga_1 + 200*Liga_2 + 0
SUBJECT TO
_C1: Liga_1 + Liga_2 = 1

_C2: 3 Liga_1 + 4 Liga_2 >= 3.2

_C3: 3 Liga_1 + 4 Liga_2 <= 3.5

_C4: Liga_1 + 1.5 Liga_2 <= 1.2

VARIABLES
Liga_1 <= 1 Continuous
Liga_2 <= 1 Continuous

# verificando se a solução ótima foi encontrada
assert optimization_result == pulp.LpStatusOptimal

# mostrando o resultado
for var in (x, y):
    print('A produção ótima deve ser {}: {}'.format(var.name, var.value()))
✓ 0.3s
A produção ótima deve ser Liga_1: 0.8
A produção ótima deve ser Liga_2: 0.2
```

Figura 8: Resolução ótima do problema, segundo o programa.

4 Questão 4

As variáveis de decisão são:

- c: número de tortas de chocolate
- m: número de tortas de morango

Do enunciado, tem-se que cada torta de chocolate pode ser vendida por R$4,00 e cada torta de morango por R$2,00. Além disso, cada torta de chocolate requer 4 ovos e 20 minutos de forno, e cada torta de morango 1 ovo e 40 minutos de forno, sendo que a Quitutaço tem à disposição 8 horas de forno e 30 ovos. Dessa maneira, a função objetivo é da forma:

$$\mathrm{R(c,m) = 4c + 2m}$$

com restrições dadas por:

$$\begin{gathered}\tfrac{1}{3}\mathrm{c} + \tfrac{2}{3}\mathrm{m} \leq 8 \\ 4\mathrm{c} + \mathrm{m} \leq 30\end{gathered}$$

Assim, o problema pode ser enunciado da seguinte forma:

$$\begin{array}{ll} Maximizar & R(c,m) = 4c + 2m \\ sujeito\ a: & c + 2m \leq 8 \\ & 4c + m \geq 30 \\ & c, m \in Z_+ \end{array}$$

conforme pode-se visualizar na Figura 9, que indica a montagem do problema no *Jupyter Notebook* através da biblioteca *PuLP* do *Python*.

Q4

```
In [186]: prob = pulp.LpProblem('Q4', pulp.LpMaximize)

In [187]: c = pulp.LpVariable('c', lowBound = 0, cat = 'Integer')
          m = pulp.LpVariable('m', lowBound = 0, cat = 'Integer')

In [188]: R = 4*c + 2*m
          prob += R

In [189]: rest1 = c + 2*m
          rest2 = 4*c + m
          prob += (rest1 <= 24)
          prob += (rest2 <= 30)

In [190]: prob

Out[190]: Q4:
          MAXIMIZE
          4*c + 2*m + 0
          SUBJECT TO
          _C1: c + 2 m <= 24

          _C2: 4 c + m <= 30

          VARIABLES
          0 <= c Integer
          0 <= m Integer
```

Figura 9: Modelagem da questão 4 por meio da biblioteca *PuLP* do *Python*

Por fim, resolveu-se o problema através da função *solve()* e escreveu-se os resultados obtidos, como mostra a Figura 10, obtendo-se os valores ótimos de c = 5 e m = 9.

$\therefore$ **Resposta:** (c,m) = (5,9).

```
In [191]: optimization_result = prob.solve()

In [192]: assert optimization_result == pulp.LpStatusOptimal

In [193]: for var in (c,m):
              print(f'{var.name}:{var.value()}')

          c:5.0
          m:9.0
```

Figura 10: Obtenção da resposta da questão 4 por meio da biblioteca *PuLP* do *Python*

5 Questão 5

As variáveis de decisão são:

- x_1: porcentagem da medida 1 a ser implementada
- x_2: porcentagem da medida 2 a ser implementada
- x_3: porcentagem da medida 3 a ser implementada

Consideraremos que toda medida aplicada a um alto forno é correspondentemente aplicada ao forno aberto. Assim, a medida 1 reduziria em 21 (= 12+9) milhões de toneladas por ano a emissão do contaminante A, a medida 2 reduziria em 45 (= 25+20) milhões de toneladas por ano a emissão do contaminante A, e assim por diante. O objetivo é minimizar o custo dessa operação, satisfazendo as reduções mínimas exigidas para cada tipo de contaminante. Dessa maneira, a função objetivo é da forma:

$$f(x_1, x_2, x_3) = 18x_1 + 13x_2 + 20x_3$$

com restrições dadas por:

$$\begin{gathered}
21x_1 + 45x_2 + 30x_3 \geq 60 \text{ (contaminante A)} \\
77x_1 + 49x_2 + 105x_3 \geq 150 \text{ (contaminante B)} \\
90x_1 + 62x_2 + 49x_3 \geq 125 \text{ (contaminante C)} \\
x_i \geq 0,\ i \in \{1, 2, 3\} \text{ (não-negatividade)} \\
x_i \leq 1,\ i \in \{1, 2, 3\} \text{ (porcentagem)}
\end{gathered}$$

Assim, o problema pode ser enunciado da seguinte forma:

$$\begin{aligned}
Minimizar \quad & f(x_1, x_2, x_3) = 18x_1 + 13x_2 + 20x_3 \\
sujeito\ a: \quad & 21x_1 + 45x_2 + 30x_3 \geq 60 \\
& 77x_1 + 49x_2 + 105x_3 \geq 150 \\
& 90x_1 + 62x_2 + 49x_3 \geq 125 \\
& x_1, x_2, x_3 \in [0, 1]
\end{aligned}$$

conforme pode-se visualizar na Figura 11, que indica a montagem do problema no *Jupyter Notebook* através da biblioteca *PuLP* do *Python*.

```
# definindo o problema como minimização
prob = pulp.LpProblem('Problema_da_poluição', pulp.LpMinimize)
```

```
# definindo as variáveis de decisão
x1 = pulp.LpVariable('medida_1', lowBound = 0, upBound=1, cat='Continuous')
x2 = pulp.LpVariable('medida_2', lowBound = 0, upBound=1, cat='Continuous')
x3 = pulp.LpVariable('medida_3', lowBound = 0, upBound=1, cat='Continuous')
```

```
# definindo a função objetivo
custo = 18*x1 + 13*x2 + 20*x3

# add a função objetivo
prob += custo
```

```
# restrições
contaminanteA = 21*x1 + 45*x2 + 30*x3 # >= 60
contaminanteB = 77*x1 + 49*x2 + 105*x3 # >= 150
contaminanteC = 90*x1 + 62*x2 + 49*x3 # >= 125
```

```
# add restrições
prob += (contaminanteA >= 60)
prob += (contaminanteB >= 150)
prob += (contaminanteC >= 125)
```

Figura 11: Modelagem da questão 5 por meio da biblioteca *PuLP* do *Python*

Por fim, resolveu-se o problema através da função *solve()* e escreveu-se os resultados obtidos, como mostra a Figura 12, obtendo-se os valores ótimos de $x_1 = 60.3\%$, $x_2 = 57.2\%$ e $x_3 = 71.9\%$.

$\therefore$ **Resposta:** $(\mathrm{x}_1, x_2, x_3) = (0.603, 0.572, 0.719)$.

```
# escrevendo o prob de otimização linear
print(prob)

# resolvendo o problema
optimization_result = prob.solve()
```

```
Problema_da_poluição:
MINIMIZE
18*medida_1 + 13*medida_2 + 20*medida_3 + 0
SUBJECT TO
_C1: 21 medida_1 + 45 medida_2 + 30 medida_3 >= 60

_C2: 77 medida_1 + 49 medida_2 + 105 medida_3 >= 150

_C3: 90 medida_1 + 62 medida_2 + 49 medida_3 >= 125

VARIABLES
medida_1 <= 1 Continuous
medida_2 <= 1 Continuous
medida_3 <= 1 Continuous

Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /home/gabriel/.local/lib/python3.8/site-packages/pulp/apis/../solverdir/cbc/linux/64/cbc /tmp/90b161f
5af564a48aecc7fd36b1852a8-pulp.mps timeMode elapsed branch printingOptions all solution /tmp/90b161f5af564a48aecc7fd
36b1852a8-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 8 COLUMNS
At line 21 RHS
At line 25 BOUNDS
At line 29 ENDATA
Problem MODEL has 3 rows, 3 columns and 9 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 3 (0) rows, 3 (0) columns and 9 (0) elements
0  Obj 0 Primal inf 4.1507934 (3)
3  Obj 32.680154
Optimal - objective value 32.680154
Optimal objective 32.68015392 - 3 iterations time 0.002
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.00   (Wallclock seconds):       0.00
```

```
# verificando se a solução ótima foi encontrada
assert optimization_result == pulp.LpStatusOptimal

# mostrando o resultado
for var in (x1, x2, x3):
    print('A produção ótima deve ser {}: {}'.format(var.name, var.value()))
```

```
A produção ótima deve ser medida_1: 0.60292844
A produção ótima deve ser medida_2: 0.5724447
A produção ótima deve ser medida_3: 0.71928305
```

Figura 12: Obtenção da resposta da questão 5 por meio da biblioteca *PuLP* do *Python*

6 Questão 6

a) O problema pode ser enunciado da seguinte forma:

$$\begin{aligned} Maximizar \quad & z(x_1, x_2) = x_1 + x_2 \\ sujeito\ a: \quad & x_1 + x_2 \leq 4 \\ & x_1 - x_2 \geq 5 \\ & x_1, x_2 \geq 0 \end{aligned}$$

conforme pode-se visualizar na Figura 13, que indica a montagem do problema no *Jupyter Notebook* através da biblioteca *PuLP* do *Python*.

Q6 - Item a

```
In [194]: prob = pulp.LpProblem('Q6a', pulp.LpMaximize)

In [195]: x1 = pulp.LpVariable('x1', lowBound = 0, cat = 'Continuous')
          x2 = pulp.LpVariable('x2', lowBound = 0, cat = 'Continuous')

In [196]: #Função objetivo
          z = x1 + x2

In [197]: #Add a função objetivo
          prob += z

In [198]: # Restrições
          rest1 = x1 + x2
          rest2 = x1 - x2
          prob += (rest1 <= 4)
          prob += (rest2 >= 5)

In [199]: prob

Out[199]: Q6a:
          MAXIMIZE
          1*x1 + 1*x2 + 0
          SUBJECT TO
          _C1: x1 + x2 <= 4

          _C2: x1 - x2 >= 5

          VARIABLES
          x1 Continuous
          x2 Continuous
```

Figura 13: Modelagem do item a da questão 6 por meio da biblioteca *PuLP* do *Python*.

Nota-se, por meio da Figura 14, que houve um *AssertionError*, isto é, o resultado da otimização não conseguiu ser definido. Isso indica que o problema não tem solução ótima.

$\therefore$ **Resposta:** $\nexists$ (x_1,x_2).

```
In [10]: optimization_result = prob.solve()

In [11]: assert optimization_result == pulp.LpStatusOptimal

         ---------------------------------------------------------------------------
         AssertionError                            Traceback (most recent call last)
         ~\AppData\Local\Temp/ipykernel_9716/868659970.py in <module>
         ----> 1 assert optimization_result == pulp.LpStatusOptimal

         AssertionError:

In [12]: for var in (x1,x2):
             print(f'{var.name}:{var.value()}')

         x1:5.0
         x2:0.0
```

Figura 14: Obtenção da resposta do item a da questão 6 por meio da biblioteca *PuLP* do *Python*.

b) O problema pode ser enunciado da seguinte forma:

$$
\begin{aligned}
Maximizar \quad & z(x_1, x_2) = 4x_1 + x_2 \\
sujeito\ a: \quad & 8x_1 + 2x_2 \leq 16 \\
& 5x_1 + 2x_2 \leq 12 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

conforme pode-se visualizar na Figura 15, que indica a montagem do problema no *Jupyter Notebook* através da biblioteca *PuLP* do *Python*.

Q6 - Item b

```
In [203]: prob = pulp.LpProblem('Q6b', pulp.LpMaximize)

In [204]: x1 = pulp.LpVariable('x1', lowBound = 0, cat = 'Continuous')
          x2 = pulp.LpVariable('x2', lowBound = 0, cat = 'Continuous')

In [205]: z = 4*x1 + x2
          prob += z

In [206]: rest1 = 8*x1 + 2*x2
          rest2 = 5*x1 + 2*x2
          prob += (rest1 <= 16)
          prob += (rest2 <= 12)

In [207]: prob

Out[207]: Q6b:
          MAXIMIZE
          4*x1 + 1*x2 + 0
          SUBJECT TO
          _C1: 8 x1 + 2 x2 <= 16

          _C2: 5 x1 + 2 x2 <= 12

          VARIABLES
          x1 Continuous
          x2 Continuous
```

Figura 15: Modelagem do item b da questão 6 por meio da biblioteca *PuLP* do *Python*.

Por meio da Figura 16, observa-se que foi obtida a solução $(x_1,x_2) = (2,0)$ pela função *solve()*. Porém, uma observação mais cuidadosa leva à conclusão de que há infinitas soluções para o problema. Basta, por exemplo, observar que z(2,0) = z($\frac{4}{3}$,$\frac{8}{3}$) = 8, ou seja, $(x_1,x_2) = (\frac{4}{3},\frac{8}{3})$ também é solução ótima. Como descobriu-se duas soluções ótimas, devem haver infinitas, o que pode ser visto mais facilmente ao se observar que o gradiente da função z, $\nabla z = (4,1)$, tem direção ortogonal à reta $4x_1 + x_2 = 8$, que compõe uma das restrições do problema.

$\therefore$ **Resposta:** Há infinitas soluções ótimas (x_1,x_2). Uma delas é $(x_1,x_2) = (2,0)$.

```
In [18]: optimization_result = prob.solve()

In [19]: assert optimization_result == pulp.LpStatusOptimal

In [20]: for var in (x1,x2):
             print(f'{var.name}:{var.value()}')

         x1:2.0
         x2:0.0
```

Figura 16: Obtenção da resposta do item b da questão 6 por meio da biblioteca *PuLP* do *Python*.

c) O problema pode ser enunciado da seguinte forma:

$$\begin{aligned} Maximizar \quad & z(x_1,x_2) = -x_1 + 3x_2 \\ sujeito\ a: \quad & x_1 - x_2 \leq 4 \\ & x_1 + 2x_2 \geq 4 \\ & x_1, x_2 \geq 0 \end{aligned}$$

conforme pode-se visualizar na Figura 17, que indica a montagem do problema no *Jupyter Notebook* através da biblioteca *PuLP* do *Python*.

Q6 - Item c

```
In [211]: prob = pulp.LpProblem('Q6c', pulp.LpMaximize)

In [212]: x1 = pulp.LpVariable('x1', lowBound = 0, cat = 'Continuous')
          x2 = pulp.LpVariable('x2', lowBound = 0, cat = 'Continuous')

In [213]: z = -x1 + 3*x2
          prob += z

In [214]: rest1 = x1 - x2
          rest2 = x1 + 2*x2
          prob += (rest1 <= 4)
          prob += (rest2 >= 4)

In [215]: prob

Out[215]: Q6c:
          MAXIMIZE
          -1*x1 + 3*x2 + 0
          SUBJECT TO
          _C1: x1 - x2 <= 4

          _C2: x1 + 2 x2 >= 4

          VARIABLES
          x1 Continuous
          x2 Continuous
```

Figura 17: Modelagem do item c da questão 6 por meio da biblioteca *PuLP* do *Python*.

Nota-se, por meio da Figura 18, que houve um *AssertionError*, isto é, o resultado da otimização não conseguiu ser definido. Isso indica que o problema não tem solução ótima. Observa-se, nesse caso em particular, que pode-se aumentar x_2 indefinidamente, respeitando-se as restrições impostas, para aumentar o valor da função objetivo z infinitamente, justificando o fato de não haver solução ótima para o problema.

$\therefore$ **Resposta:** $\nexists$ $(x_1$,$x_2)$, pois pode-se fazer $x_2 \to \infty$.

```
In [26]: optimization_result = prob.solve()

In [27]: assert optimization_result == pulp.LpStatusOptimal

         ---------------------------------------------------------------------------
         AssertionError                            Traceback (most recent call last)
         ~\AppData\Local\Temp/ipykernel_9716/868659970.py in <module>
         ----> 1 assert optimization_result == pulp.LpStatusOptimal

         AssertionError:

In [28]: for var in (x1,x2):
             print(f'{var.name}:{var.value()}')

         x1:0.0
         x2:0.0
```

Figura 18: Obtenção da resposta do item c da questão 6 por meio da biblioteca *PuLP* do *Python*.

d) O problema pode ser enunciado da seguinte forma:

$$
\begin{aligned}
Maximizar \quad & z(x_1, x_2) = 3x_1 + x_2 \\
sujeito\ a: \quad & 2x_1 + x_2 \leq 6 \\
& x_1 + 3x_2 \leq 9 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

conforme pode-se visualizar na Figura 19, que indica a montagem do problema no *Jupyter Notebook* através da biblioteca *PuLP* do *Python*.

Figura 19: Modelagem do item d da questão 6 por meio da biblioteca *PuLP* do *Python*.

Por fim, resolveu-se o problema através da função *solve()* e escreveu-se os resultados obtidos, conforme mostra a Figura 20, obtendo-se os valores ótimos de $x_1 = 3$ e $x_2 = 0$.

$\therefore$ **Resposta:** $(x_1, x_2) = (3,\ 0)$.

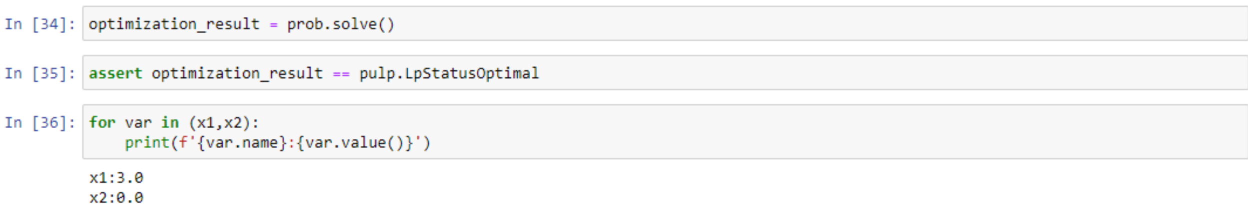


Figura 20: Obtenção da resposta do item d da questão 6 por meio da biblioteca *PuLP* do *Python*.