Departamento de Engenharia Informática e de Sistemas

Metodologias de Otimização e Apoio à Decisão

Resolução de problemas de PL em Python

- Parte IV -

EXEMPLO 5

Considere o seguinte problema (Anexo 1 do Capítulo II):

Mensalmente um carpinteiro possui 6 peças de madeira e dispõe de 28 horas livres para construir dois modelos diferentes de bancos. Cada banco do modelo I requer 2 peças de madeira e exige 7 horas de trabalho. Cada banco do modelo II requer 1 peça de madeira e exige 8 horas de trabalho. Os lucros unitários obtidos com a venda dos bancos são de, respetivamente, 12 e 8 Unidades Monetárias (U.M.).

O carpinteiro pretende saber quantos bancos de cada modelo deve fabricar por mês, de forma a maximizar o lucro obtido com a venda dos bancos.

Considerando que x_1 e x_2 são o nº de bancos do modelo I e do modelo II a fabricar por mês, respetivamente, e que o objetivo é maximizar o lucro mensal, o modelo matemático pode ser definido como se apresenta em seguida.

$$\text{Max } z = 12\,x_1 + 8\,x_2 \qquad \textit{(Lucro mensal)}$$

sujeito a

$$2\,x_1 + x_2 \leq 6 \qquad \textit{(Disponibilidade de madeira)}$$

$$7\,x_1 + 8\,x_2 \leq 28 \qquad \textit{(Disponibilidade de mão-de-obra)}$$

$$x_1 \geq 0,\ x_2 \geq 0$$

x_1, x_2 inteiros

Este exemplo corresponde a um modelo de programação linear inteira pura (PLIP). Tal como no EXEMPLO 4, vamos inserir os dados do problema num ficheiro EXCEL e depois criar e resolver o modelo em Python importando os dados desse mesmo ficheiro.

O ficheiro EXCEL com os referidos dados designa-se por “Data_EX5.xlsx” e apresenta o conteúdo ilustrado na imagem seguinte.

	A	B	C	D	E	F	G	H
1	**Bench models**		**Wood**	**Hand labour**	**Profits**		**Resources**	**Maximum**
2	1		2	7	12		6	Wood (pieces)
3	2		1	8	8		28	Hand Labour (hours)
4			(pieces)	(hours)	(MU)			
5								

Em seguida apresentam-se duas versões diferentes da implementação em Python: uma primeira semelhante à do EXEMPLO 4, com a criação de vários dicionários relativos às várias colunas do ficheiro e indexados pelo modelo de banco; a segunda envolve a criação de diversas listas cuja indexação é feita pela posição que os elementos ocupam nessas listas.

1ª versão:

```
"""
Carpinteiro / Carpenter
"""
from pandas import read_excel
from pulp import *

# Lê área verde do ficheiro EXCEL / Read green area from EXCEL file
df1 = read_excel("Data_EX5.xlsx",nrows=2,usecols=(['Bench models','Wood',
                                                   'Hand labour','Profits']))
print("==> Dataframe 1\n",df1)

# Cria lista com modelos de bancos/ Create list with models of benches
bench_models = list(df1['Bench models'])
print("==> Bench models\n",bench_models)

# Cria dicionário de quantidade de madeira / Create a dictionary of wood quantity
# indexado por modelo de banco             / indexed by bench model
wood = dict(zip(bench_models,df1['Wood']))
print("==> Wood\n",wood)

# Cria dicionário de tempo de mão-de-obra  / Create a dictionary of hand labour time
# indexado por modelo de banco             / indexed by bench model
hand_labour = dict(zip(bench_models,df1['Hand labour']))
print("==> Hand labour\n",hand_labour)

# Cria dicionário de lucros unitários / Create a dictionary of unitary profits
# indexado por modelo de banco        / indexed by bench model
profits = dict(zip(bench_models,df1['Profits']))
print("==> Profits\n",profits)

# Lê área azul do ficheiro EXCEL / Read blue area from EXCEL file
df2 = read_excel("Data_EX5.xlsx",nrows=2,usecols=(['Resources']))
print("==> Dataframe 2\n",df2)

# Cria lista de recursos / Create a list of resources
resources = list(df2['Resources'])
print("==> Resources\n",resources)

# Cria modelo / Create model
model=LpProblem("Carpinteiro/Carpenter",LpMaximize)

# Cria variáveis de decisão / Create decision variables
x = LpVariable.dicts("x",bench_models,lowBound=0,cat=LpInteger)

# Cria função objetivo / Create objective function
model += lpSum([profits[i]*x[i] for i in bench_models])

# Cria restrições / Create constraints
model += lpSum([wood[i] * x[i] for i in bench_models]) <= resources[0]
model += lpSum([hand_labour[i] * x[i] for i in bench_models]) <= resources[1]

# Resolve modelo / Solve model
model.solve()

# Visualizar resultados / Visualize results
print("-------- Resultados / Results ---------")
print(f"Status = {model.status} <=> {LpStatus[model.status]}")
print(f"z* = {value(model.objective)}")
for var in model.variables():
    print(f"{var.name}* = {var.value()}")
for name,constraint in model.constraints.items():
    print(f"{name}: {constraint.value()}")
```

2ª versão:

```
"""
Carpinteiro / Carpenter
"""
from pandas import read_excel
from pulp import *

# Lê área verde do ficheiro EXCEL / Read green area from EXCEL file
df1 = read_excel("Data_EX5.xlsx",nrows=2,usecols=(['Wood','Hand labour',
                                                   'Profits']))
print("==> Dataframe 1\n",df1)

# Cria lista da quantidade de madeira / Create a list of wood quantity
wood = list(df1['Wood'])
print("==> Wood\n",wood)

# Cria lista de tempo de mão-de-obra  / Create a list of hand labour time
hand_labour = list(df1['Hand labour'])
print("==> Hand labour\n",hand_labour)

# Cria lista de lucros unitários / Create a list of unitary profits
profits = list(df1['Profits'])
print("==> Profits\n",profits)

# Lê área azul do ficheiro EXCEL / Read blue area from EXCEL file
df2 = read_excel("Data_EX5.xlsx",nrows=2,usecols=(['Resources']))
print("==> Dataframe 2\n",df2)

# Cria lista de recursos / Create a list of resources
resources = list(df2['Resources'])
print("==> Resources\n",resources)

# Cria modelo / Create model
model=LpProblem("Carpinteiro/Carpenter",LpMaximize)

# Cria variáveis de decisão / Create decision variables
x = {j: LpVariable(name=f"x{j}", lowBound=0,cat=LpInteger) for j in range(1,3)}

# Cria função objetivo / Create objective function
model += lpSum([profits[j]*x[j+1] for j in range(2)])

# Cria restrições / Create constraints
model += lpSum([wood[j] * x[j+1] for j in range(2)]) <= resources[0]
model += lpSum([hand_labour[j] * x[j+1] for j in range(2)]) <= resources[1]

# Resolve modelo / Solve model
model.solve()

# Visualizar resultados / Visualize results
print("-------- Resultados / Results ---------")
print(f"Status = {model.status} <=> {LpStatus[model.status]}")
print(f"z* = {value(model.objective)}")
for var in model.variables():
    print(f"{var.name}* = {var.value()}")
for name,constraint in model.constraints.items():
    print(f"{name}: {constraint.value()}")
```

Em qualquer uma das versões, o resultado final surgirá na consola.

```
-------- Resultados / Results ---------
Status = 1 <=> Optimal
z* = 36
x1* = 3
x2* = 0
_C1: 0
_C2: -7
```

Interpretação dos resultados

O carpinteiro deverá fabricar, por mês, 3 bancos do modelo I e nenhum banco do modelo II, conseguindo desta forma um lucro máximo mensal de 36 U.M. O valor que surge na 2ª restrição (_C2) significa que das 28 horas/mês que o carpinteiro dispõe para trabalhar no fabrico dos bancos, 7 não vão ser utilizadas.

Sugestão

Compare os resultados obtidos com os do Anexo 1 do Capítulo II.