Exemplos de resolução de problemas lineares utilizando o *solver* do *SciPy*

Sandro Leite Furtado
Mestrado em Computação Aplicada
Universidade de Brasília
Brasília DF, 70910-900, Brasil

Tiago Pereira Vidigal
Mestrado em Computação Aplicada
Universidade de Brasília
Brasília DF, 70910-900, Brasil

William Oliveira Camelo
Mestrado em Computação Aplicada
Universidade de Brasília
Brasília DF, 70910-900, Brasil

***Resumo*—Problemas lineares modelam vários cenários de otimização. O uso de *solvers* permite uma resolução rápida e padronizada por computadores. A função *'scipy.optimize.linprog'* do pacote Python *SciPy* foi usado em alguns exemplos para demonstrar seu uso. Sua flexibilidade e simplicidade permitiu facilmente modelar diversos problemas e sua eficiência se mostrou interessante com até 30 variáveis.**

1. Introdução

Problemas de otimização onde as equações relacionadas são lineares são chamados de problemas lineares. Diversas técnicas existem para sua resolução, como métodos gráficos, analíticos e tabulares [1]. Por serem problemas comuns, *solvers* que automatizam o calculo são amplamente disponíveis.

O *solver 'scipy.optimize.linprog'*, parte da biblioteca *SciPy*, é uma alternativa Python para resolução automática de problemas lineares. A função foi implementada pela *The SciPy community* e permite a modelagem de forma bem flexível. O método de resolução padrão é o *interior-point* de Andersen, mas é possível escolher dentre as opções *simplex* de Dantzig, *revised simplex* de Bartels, *HiGHS* de Huangfu et al (tendo as opções *dual simplex* e *interior-point*). [2]

O objetivo deste trabalho é apresentar alguns exemplos demonstrando o uso do *solver 'scipy.optimize.linprog*. O único método de resolução utilizado é o *interior-point* de Andersen.

2. Desenvolvimento

O uso do solver é avaliado em três exemplos. O objetivo é demonstrar a modelagem de problemas reais utilizando a formatação necessária para seu uso. O resultado de cada modelagem é apresentado na seção 3.

A modelagem do problema para o *solver* espera uma *array* indicando os coeficientes das equações e limites envolvidos. O problema deve ser de minimização e as desigualdades devem estabelecer apenas limites superiores ao problema. Equações de igualdade também podem ser usadas caso se encaixem no problema.

2.1. Exemplo 1: Carteira de Investimentos

O primeiro exemplo é utilizando a questão 3.6-2 [1]. O modelo é:

$$\begin{aligned} \min \quad & -Z = -50x_1 - 20x_2 - 25x_3 \\ \text{unequals} \quad & 9x_1 + 3x_2 + 5x_3 \leq 500 \\ & 5x_1 + 4x_2 + 0x_3 \leq 350 \\ & 3x_1 + 0x_2 + 2x_3 \leq 150 \\ \text{bounds} \quad & 0 \leq x_1 \\ & 0 \leq x_2 \\ & 0 \leq x_3 \leq 20 \end{aligned} \tag{1}$$

O problema, para ser usado no solver, é modelado e resolvido da seguinte forma:

```
c = [-50, -20, -25]
A_ub = [[9, 3, 5],
        [5, 4, 0],
        [3, 0, 2]]
b_ub = [500, 350, 150]
x1_bounds = (0, None)
x2_bounds = (0, None)
x3_bounds = (0, 20)
bounds = [x1_bounds, x2_bounds, x3_bounds]

res = scipy.optimize.linprog(c=c,
        A_ub=A_ub, b_ub=b_ub,
        bounds=bounds)
```

2.2. Exemplo 2: Escala de funcionários

O segundo exemplo é utilizando a questão 3.6-3 [1]. O modelo é:

$$\begin{aligned} \min \quad & Z = 100(6x_1 + 8x_2 + 7x_3 + 4x_4 + 9x_5 + 6x_6) \\ \text{equals} \quad & 1x_1 + 0x_2 + 0x_3 + 1x_4 + 0x_5 + 0x_6 = 300 \\ & 0x_1 + 1x_2 + 0x_3 + 0x_4 + 1x_5 + 0x_6 = 200 \\ & 0x_1 + 0x_2 + 1x_3 + 0x_4 + 0x_5 + 1x_6 = 400 \\ & 1x_1 + 1x_2 + 1x_3 + 0x_4 + 0x_5 + 0x_6 = 400 \\ & 0x_1 + 0x_2 + 0x_3 + 1x_4 + 1x_5 + 1x_6 = 500 \end{aligned} \tag{2}$$

O problema, para ser usado no solver, é modelado e resolvido da seguinte forma:

```
c = [600, 800, 700, 400, 900, 600]
A_eq = [[1, 0, 0, 1, 0, 0],
        [0, 1, 0, 0, 1, 0],
        [0, 0, 1, 0, 0, 1],
        [1, 1, 1, 0, 0, 0],
        [0, 0, 0, 1, 1, 1]]
b_eq = [300, 200, 400, 400, 500]

res = scipy.optimize.linprog(c=c,
        A_eq=A_eq, b_eq=b_eq)
```

2.3. Exemplo 3: Problema de mistura

O terceiro exemplo é utilizando a questão 3.6-4 [1]. O modelo é:

$$
\begin{aligned}
\min \quad & -Z = 25(x_{11}+x_{12}+x_{13}+x_{14}+x_{15}) \\
& \quad + 26(x_{21}+x_{22}+x_{23}+x_{24}+x_{25}) \\
& \quad + 24(x_{31}+x_{32}+x_{33}+x_{34}+x_{35}) \\
& \quad + 23(x_{41}+x_{42}+x_{43}+x_{44}+x_{45}) \\
& \quad + 28(x_{51}+x_{52}+x_{53}+x_{54}+x_{55}) \\
& \quad + 30(x_{61}+x_{62}+x_{63}+x_{64}+x_{65}) \\
\text{unequals} \quad & -1(x_{11}+x_{12}+x_{13}+x_{14}+x_{15}) \leq -8 \\
& -1(x_{21}+x_{22}+x_{23}+x_{24}+x_{25}) \leq -8 \\
& -1(x_{31}+x_{32}+x_{33}+x_{34}+x_{35}) \leq -8 \\
& -1(x_{41}+x_{42}+x_{43}+x_{44}+x_{45}) \leq -8 \\
& -1(x_{51}+x_{52}+x_{53}+x_{54}+x_{55}) \leq -7 \\
& -1(x_{61}+x_{62}+x_{63}+x_{64}+x_{65}) \leq -7 \\
\text{equals} \quad & 1(x_{11}+x_{21}+x_{31}+x_{41}+x_{51}+x_{61}) = 14 \\
& 1(x_{12}+x_{22}+x_{32}+x_{42}+x_{52}+x_{62}) = 14 \\
& 1(x_{13}+x_{23}+x_{33}+x_{43}+x_{53}+x_{63}) = 14 \\
& 1(x_{14}+x_{24}+x_{34}+x_{44}+x_{54}+x_{64}) = 14 \\
& 1(x_{15}+x_{25}+x_{35}+x_{45}+x_{55}+x_{65}) = 14 \\
\text{bounds} \quad & 0 \leq x_{11} \leq 6, \quad x_{12} = 0, \quad 0 \leq x_{13} \leq 6 \\
& x_{14} = 0, \quad 0 \leq x_{15} \leq 6, \quad x_{21} = 0 \\
& 0 \leq x_{22} \leq 6, \quad x_{23} = 0, \quad 0 \leq x_{24} \leq 6 \\
& x_{25} = 0, \quad 0 \leq x_{31} \leq 4, \quad \leq x_{32} \leq 8 \\
& 0 \leq x_{33} \leq 4, \quad x_{34} = 0, \quad 0 \leq x_{35} \leq 4 \\
& 0 \leq x_{41} \leq 5, \quad 0 \leq x_{42} \leq 5, \quad 0 \leq x_{43} \leq 5 \\
& x_{44} = 0, \quad 0 \leq x_{45} \leq 5, \quad 0 \leq x_{51} \leq 3 \\
& x_{52} = 0, \quad 0 \leq x_{53} \leq 3, \quad 0 \leq x_{54} \leq 8 \\
& x_{55} = 0, \quad x_{61} = 0, \quad x_{62} = 0 \\
& x_{63} = 0, \quad 0 \leq x_{64} \leq 6, \quad 0 \leq x_{65} \leq 2
\end{aligned}
\tag{3}
$$

O problema, para ser usado no solver, é modelado e resolvido da seguinte forma:

```
c = [25]*5+[26]*5+[24]*5+[23]*5+[28]*5+[30]*5
A_ub = [[-1]*5 +[0]*5 +[0]*5 +[0]*5 +[0]*5 +[0]*5,
        [0]*5+[-1]*5 +[0]*5 +[0]*5 +[0]*5 +[0]*5,
        [0]*5 +[0]*5+[-1]*5 +[0]*5 +[0]*5 +[0]*5,
        [0]*5 +[0]*5 +[0]*5+[-1]*5 +[0]*5 +[0]*5,
        [0]*5 +[0]*5 +[0]*5 +[0]*5+[-1]*5 +[0]*5,
        [0]*5 +[0]*5 +[0]*5 +[0]*5 +[0]*5+[-1]*5]
b_ub = [-8, -8, -8, -8, -7, -7]
A_eq = [[1, 0, 0, 0, 0]*6,
        [0, 1, 0, 0, 0]*6,
        [0, 0, 1, 0, 0]*6,
        [0, 0, 0, 1, 0]*6,
        [0, 0, 0, 0, 1]*6]
b_eq = [14, 14, 14, 14, 14]
x11_bounds = (0, 6); x12_bounds = (0, 0)
x13_bounds = (0, 6); x14_bounds = (0, 0)
x15_bounds = (0, 6); x21_bounds = (0, 0)
x22_bounds = (0, 6); x23_bounds = (0, 0)
x24_bounds = (0, 6); x25_bounds = (0, 0)
x31_bounds = (0, 4); x32_bounds = (0, 8)
x33_bounds = (0, 4); x34_bounds = (0, 0)
x35_bounds = (0, 4); x41_bounds = (0, 5)
x42_bounds = (0, 5); x43_bounds = (0, 5)
x44_bounds = (0, 0); x45_bounds = (0, 5)
x51_bounds = (0, 3); x52_bounds = (0, 0)
x53_bounds = (0, 3); x54_bounds = (0, 8)
x55_bounds = (0, 0); x61_bounds = (0, 0)
x62_bounds = (0, 0); x63_bounds = (0, 0)
x64_bounds = (0, 6); x65_bounds = (0, 2)
bounds = [x11_bounds, x12_bounds, x13_bounds,
          x14_bounds, x15_bounds, x21_bounds,
          x22_bounds, x23_bounds, x24_bounds,
          x25_bounds, x31_bounds, x32_bounds,
          x33_bounds, x34_bounds, x35_bounds,
          x41_bounds, x42_bounds, x43_bounds,
          x44_bounds, x45_bounds, x51_bounds,
          x52_bounds, x53_bounds, x54_bounds,
          x55_bounds, x61_bounds, x62_bounds,
          x63_bounds, x64_bounds, x65_bounds]

res = scipy.optimize.linprog(c=c,
       A_eq=A_eq, b_eq=b_eq, A_ub=A_ub, b_ub=b_ub,
       bounds=bounds)
```

3. Resultados

O resultados dos problemas modelados em 2 são apresentados a seguir. Um resumo do processamento, o valor de Z e os valores de X são listados. O exemplo 1 resultou em:

```
Summary of results:
     con: array([], dtype=float64)
 message: 'Optimization terminated successfully.'
     nit: 5
   slack: array([1.01560659e-06, 7.16238674e-07,
                 3.14285717e+01])
  status: 0
 success: True

Z: 2904.761898856456
X: [26.19047614 54.76190465 19.99999996]
```

O exemplo 2 resultou em:

```
Summary of results:
     con: array([9.24823041e-07, 6.14513795e-07,
                 1.23501030e-06, 1.23185009e-06,
                 1.54249705e-06])
 message: 'Optimization terminated successfully.'
     nit: 5
   slack: array([], dtype=float64)
  status: 0
 success: True

Z: 539999.9983378148
X: [1.39655395e-08 1.99999999e+02 1.99999999e+02
    2.99999999e+02 3.49164400e-09 1.99999999e+02]
```

O exemplo 3 resultou em:

```
Summary of results:
     con: array([2.19345608e-09, 2.41093900e-09,
                 2.19378471e-09, 2.41274023e-09,
                 2.19379359e-09])
 message: 'Optimization terminated successfully.'
     nit: 6
   slack: array([9.99999998e-01, -1.34472167e-09,
                 1.10000000e+01,  1.20000000e+01,
                -9.97330218e-10, -1.25582034e-09])
  status: 0
 success: True

Z: 1754.9999997146087
X:
[[2.78890143 0. 2.78890142 0.         3.42219715]
 [0.         2. 0.         6.         0.        ]
 [4.         7. 4.         0.         4.        ]
 [5.         5. 5.         0.         5.        ]
 [2.21109857 0. 2.21109857 2.57780285 0.        ]
 [0.         0. 0.         5.42219715 1.57780285]]
```

O tempo de resolução de cada exemplo foi calculado pela média de 100 execuções. O valor encontrado, em segundos, foi:

- Exemplo 1: $0.0035194964 \pm 0.0019942994$ s
- Exemplo 2: $0.0039004740 \pm 0.0022250296$ s
- Exemplo 3: $0.0045470143 \pm 0.0022907777$ s

4. Conclusão

O *solver* apresentou os resultados dos problemas em tempo hábil utilizando um notebook pessoal, mesmo para o exemplo 3 com 30 variáveis. A facilidade de representar o problema com *arrays* e a flexibilidade de juntar igualdades, desigualdades e limites demonstrou que é uma ótima alternativa para a resolução de problemas lineares.

Referências

[1] F. Hillier and G. Lieberman, *Introdução à Pesquisa Operacional*, 9th ed. AMGH, 2013. [Online]. Available: https://integrada.minhabiblioteca.com.br/books/9788580551198

[2] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.