# Pesquisa Operacional

Igor M. Coelho

13 de Julho de 2020

Section 1

# Programação Linear Inteira

# Sobre esse material

Esses slides foram possíveis devido a contribuições de diversas pessoas/materiais, em especial:

- Notas do prof. Marcone Jamilson Freitas Souza
- Livro Nelson Maculan e Marcia Fampa
- Livro-texto do curso
- [2] Tutorial ilectures (https://igormcoelho.github.io/ilectures-pandoc/)

# Fundamentos Necessários

Caso não se sintam confiantes nos tópicos abaixo, façam uma revisão antes de aprofundar neste material:

- Simplex e Programação Linear

Section 2

# Programação Inteira

# Introdução

Diversos problemas podem ser modelados por desigualdades lineares, no formato padrão de um Problema de Programação Linear (PPL).

Considere um problema da mochila com capacidade $Q$, na qual um conjunto de itens $i \in \mathcal{I}$ pode ser selecionado, acarretando lucros $p_i \in \mathbb{Z}^+$, mas também incorrendo em pesos $w_i \in \mathbb{Z}^+$.

# Problema da Mochila Fracionária

Note que, no PPL abaixo, as variáveis $x_i$ podem assumir valores fracionários:

$$\textit{maximizar} \sum_{i \in \mathcal{I}} p_i x_i$$

*Sujeito a:*

$$\begin{aligned} \textstyle\sum_{i \in \mathcal{I}} w_i x_i &\leq Q & \\ 0 \leq x_i &\leq 1 & \forall i \in \mathcal{I} \\ x_i &\in \mathbb{R}_{\geq 0} & \forall i \in \mathcal{I} \end{aligned}$$

# Problema da Mochila 0-1

Na prática, pode ser desejável que os itens não possam ser divididos. Em outras palavras, gostaríamos que as variáveis $x_i$ assumam valores no conjunto dos inteiros não-negativos.

$$\textit{maximizar} \sum_{i \in \mathcal{I}} p_i x_i$$

*Sujeito a:*

$$\begin{aligned} \textstyle\sum_{i \in \mathcal{I}} w_i x_i &\leq Q & \\ 0 \leq x_i &\leq 1 & \forall i \in \mathcal{I} \\ x_i &\in \mathbb{Z}_{\geq 0} & \forall i \in \mathcal{I} \end{aligned}$$

# Problemas Inteiros

O Simplex serve para resolver problemas fracionários, porém não resolve problemas inteiros.

Ressaltamos que o *Problema de Programação Inteira* ou *Programação Inteira Mista* (MIP) é NP-Difícil, então não é conhecido um algoritmo que resolva todo tipo de MIP em tempo polinomial.

A abordagem padrão consiste em utilizar uma estratégia de enumeração, chamada Branch&Bound, que utiliza o Simplex para efetuar *podas inteligentes* na árvore de ramificação.

# Slides Complementares

Veja o slide complementar do prof. Marcone, para uma apresentação com exemplo do método Branch&Bound.

Section 3

# Tarefas

# Implementação no Python-MIP

A biblioteca `python-mip` é desenvolvida por pesquisadores da *Universidade Federal de Ouro Preto* (UFOP) e pode ser utilizada para modelar e testar problemas MIP.

O resolvedor integrado Coin CBC é bastante eficiente e de código-aberto, sendo capaz de resolver problemas de pequeno e médio porte. Para problemas maiores (e mais complexos) é recomendada a instalação de *solvers* proprietários como CPLEX e Gurobi, integrando ao `python-mip`.

Para instalar o `python-mip` no python3: `pip install mip`.

Documentação online:
https://python-mip.readthedocs.io/en/latest/examples.html

# Python-MIP exemplo Mochila 0-1

```
from mip import Model, xsum, maximize, BINARY

# dados do problema
p = [10, 13, 9,  31, 7,  15]
w = [11, 15, 10, 35, 10, 33]
Q, I = 47, range(len(w))

# cria um modelo
m = Model("knapsack")
# adiciona variáveis 'x' ao modelo 'm'
x = [m.add_var(var_type=BINARY) for i in I]
# especifica a função objetivo do modelo 'm'
m.objective = maximize(xsum(p[i] * x[i] for i in I))
# adiciona restrições ao modelo 'm'
m += xsum(w[i] * x[i] for i in I) <= Q
```

# Python-MIP exemplo Mochila 0-1

```
# resolve o modelo 'm'
status = m.optimize()

# Imprime a solução final obtida.
#  Note que variáveis da relaxação assumem valores
#  tipo 'float', então precisamos ter certa tolerância
#  para inferir respectivos valores inteiros '0' e '1'
selected = [i for i in I if x[i].x >= 0.99]
print("selected items: {}".format(selected))

# imprime a situação da otimização e valor objetivo
print("status =", status, " obj =", m.objective_value)
# imprime o valor para cada variável na solução
for i in I:
    print("i=", i, "->", x[i].x)
```

# Tarefa Python-MIP com Mochila

O valor ótimo esperado é 41, com $x = (1, 0, 0, 1, 0, 0)$.

**Tarefa:** efetue as seguintes modificações no exemplo anterior, e verifique os valores obtidos na solução ótima:

- substitua o tipo `BINARY` por `INTEGER` nas variáveis (e faça a devida importação no python) e obtenha ótimo 42 com $x = (2, 1, 1, 0, 0, 0)$.
- substitua o tipo `BINARY` por `CONTINUOUS` nas variáveis (e faça a devida importação no python) e obtenha ótimo 42.7273 com $x = (4.273, 0, 0, 0, 0, 0)$.

Analise o significado dessas modificações, bem como os valores de solução obtidos.

# Para a próxima aula

Implemente os modelos do *Problema do Caminho Mínimo* e *Problema do Fluxo Máximo* utilizando o `python-mip`.

**Dica:** observe os modelos existentes no material complementar do curso (apostila do prof. Marcone Jamilson Freitas Souza).

# Lista de Exercícios

A lista de exercícios está disponibilizada no site.