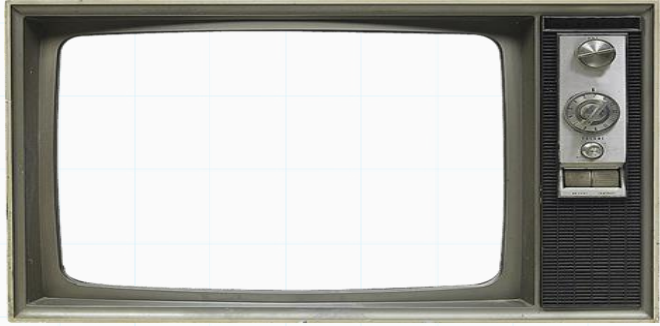
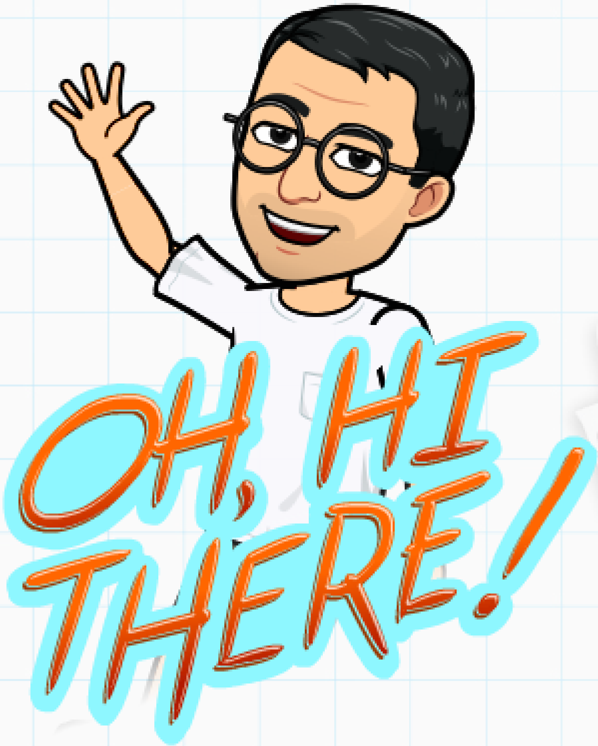
Pesquisa Operacional

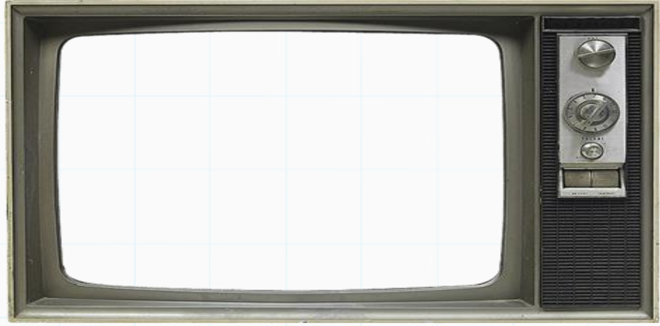


Professor : Yuri Frota

yuri@ic.uff.br



Solvers

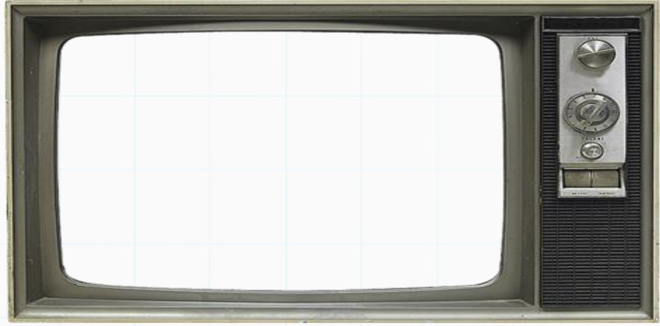


GUROBI OPTIMIZATION

X

IBM ILOG CPLEX

Solvers



GUROBI OPTIMIZATION

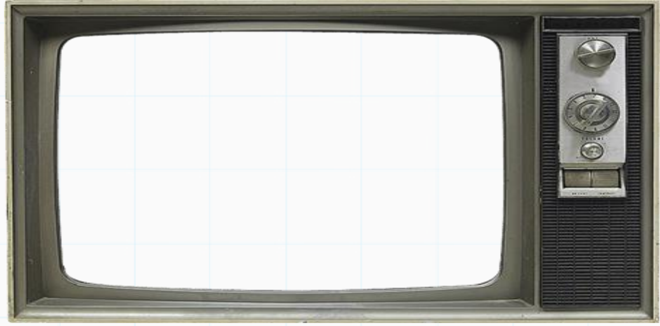
MARVEL

x

IBM ILOG CPLEX

DC

Solvers



GUROBI OPTIMIZATION

X

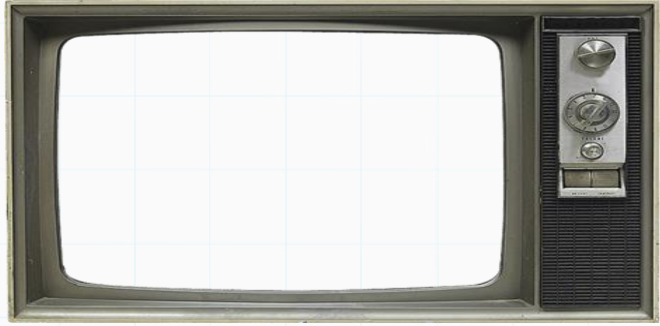
IBM ILOG CPLEX

MARVEL

DC

FICO Xpress

Solvers



GUROBI OPTIMIZATION

X

IBM ILOG CPLEX

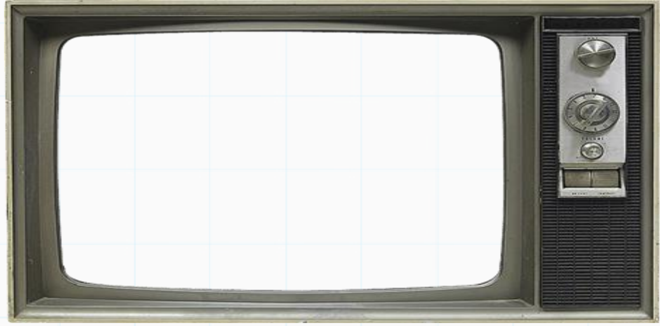
MARVEL

DC

FICO Xpress

DARK HORSE COMICS

Solvers



GUROBI OPTIMIZATION

x

IBM ILOG CPLEX

MARVEL

DC

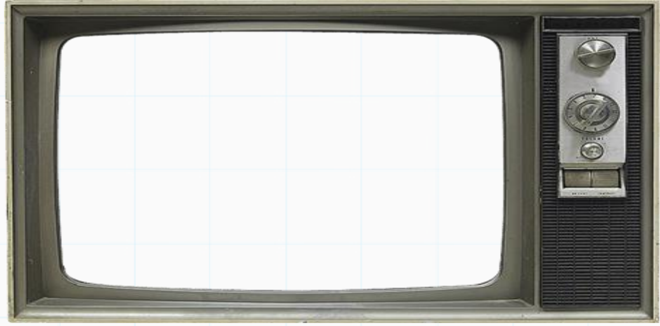
mosek

FICO Xpress

DARK HORSE COMICS

image

Solvers



GUROBI OPTIMIZATION

X

IBM ILOG CPLEX

MARVEL

DC

GLPK

NLOPT

COIN|OR

SCIP

lpsolve

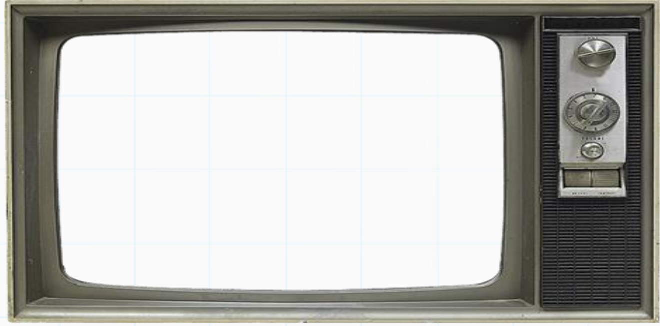
FICO Xpress

DARK HORSE COMICS

mosek

image

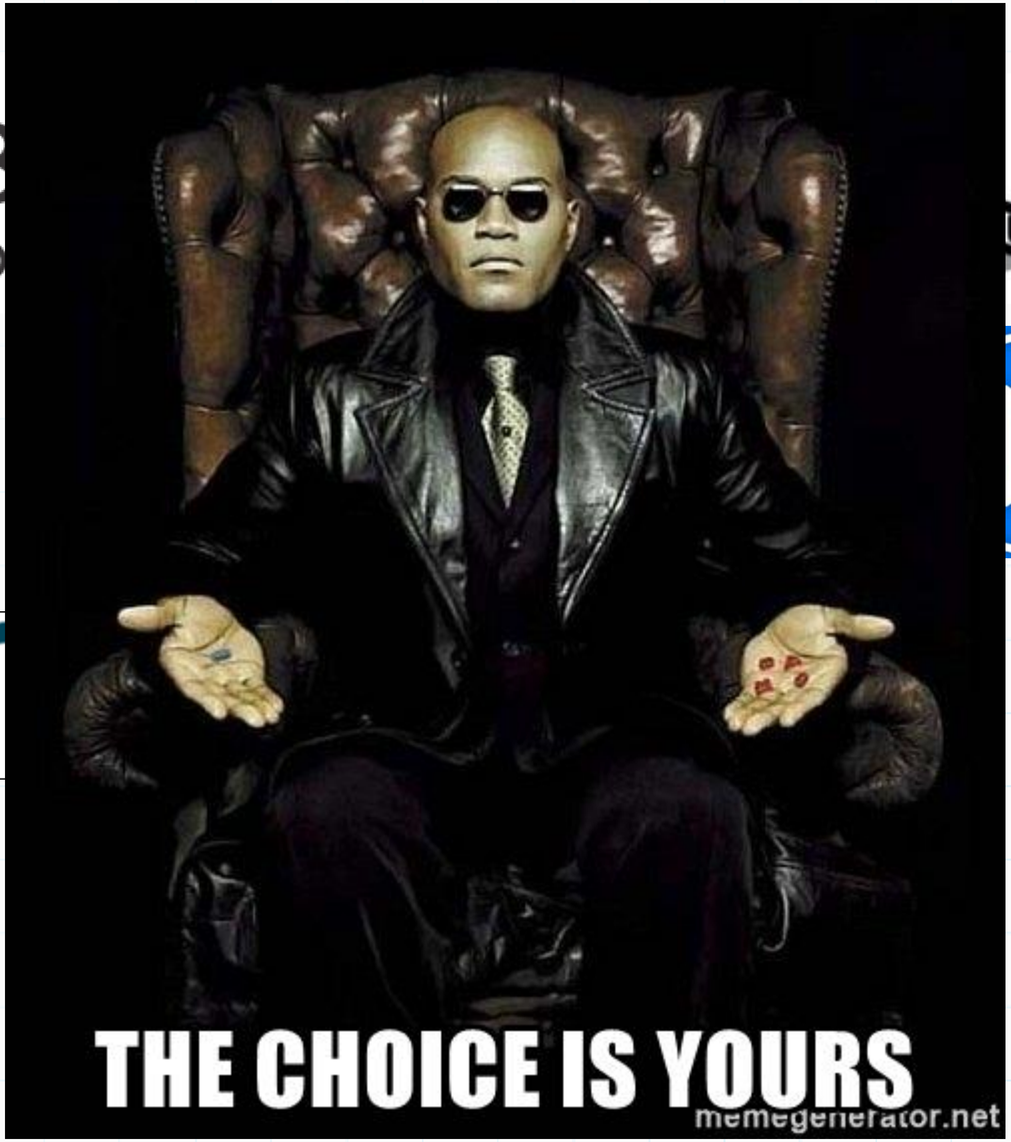
Solvers



GUROB
OPTIMIZATIO

MARVEL

CPLEX



FICO Xpress

DARK HORSE COMICS

COIN|OR

GLPK

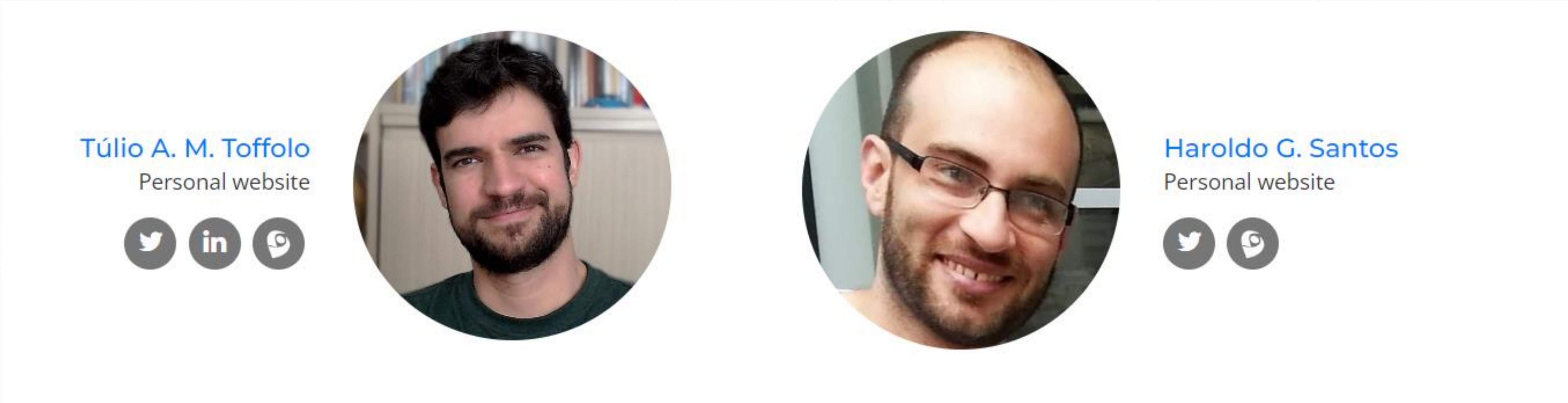
NLOPT

SCIP

lpsolve

Python-MIP

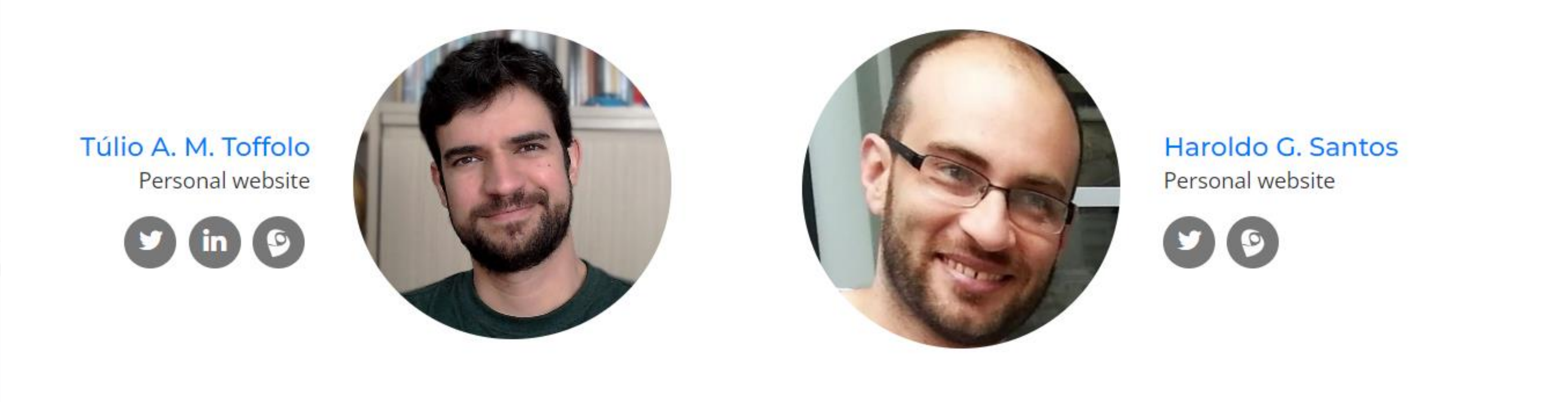
- No curso vamos ver o Python-MIP:
    - Python-MIP: biblioteca do python para problemas PPL e PPI



Python-MIP

- <u>No curso vamos ver o Python-MIP:</u>

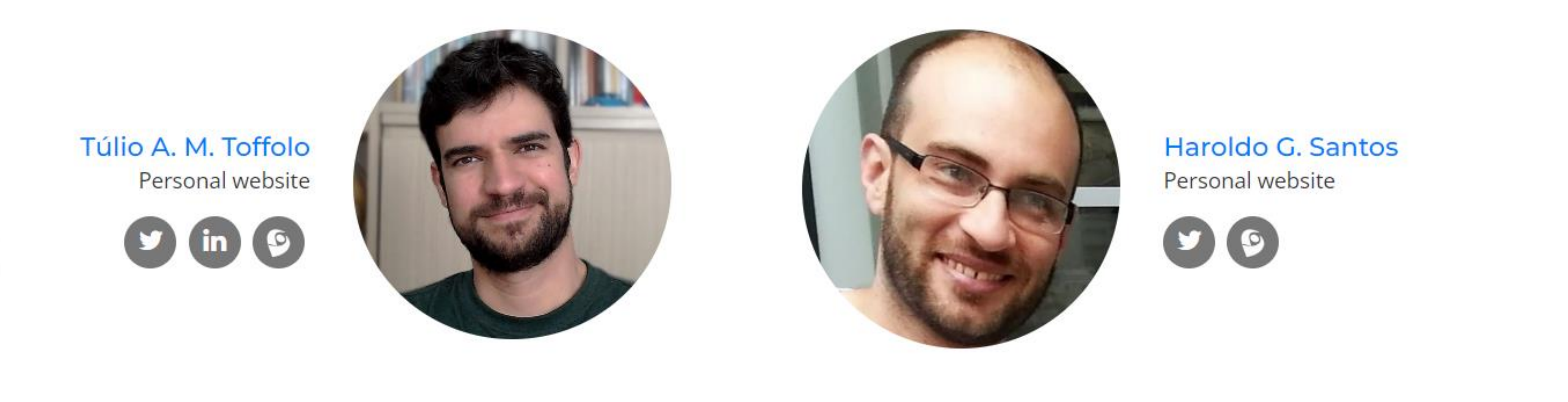
  - <u>Python-MIP</u>: biblioteca do python para problemas PPL e PPI
    - Muito fácil de instalar e usar, já vem com o solver gratuito (COIN-OR)
    - Desempenho lento e manipulação limitada (COIN-OR)
    - Ideal para problemas pequenos, testes e aprendizado (COIN-OR)



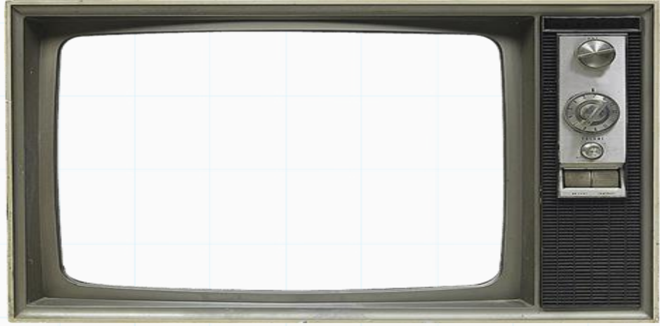
Python-MIP

- <u>No curso vamos ver o Python-MIP:</u>

  - <u>Python-MIP</u>: biblioteca do python para problemas PPL e PPI
    - Muito fácil de instalar e usar, já vem com o solver gratuito (COIN-OR)
    - Desempenho lento e manipulação limitada (COIN-OR)
    - Ideal para problemas pequenos, testes e aprendizado (COIN-OR)
    - Pode ser usado com o GUROBI (se instalado), se tornando uma ferramenta profissional.



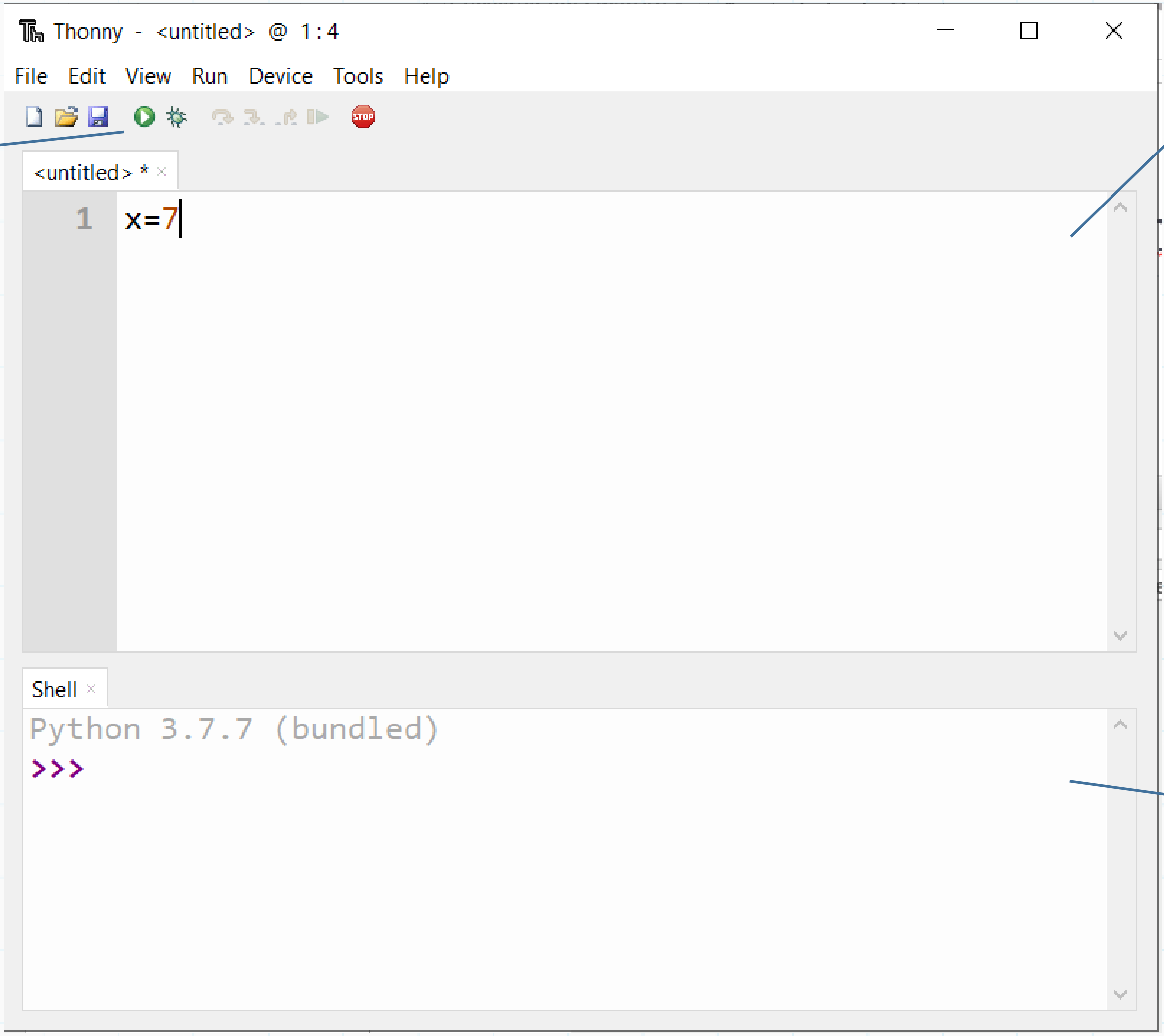
Python-MIP



Instalando o IDE+Compilador

- Usaremos na aula o Thonny (leve e educativo).
- Tem para Windows, MAC e Linux
- https://thonny.org/

para executar o código



onde o código é escrito

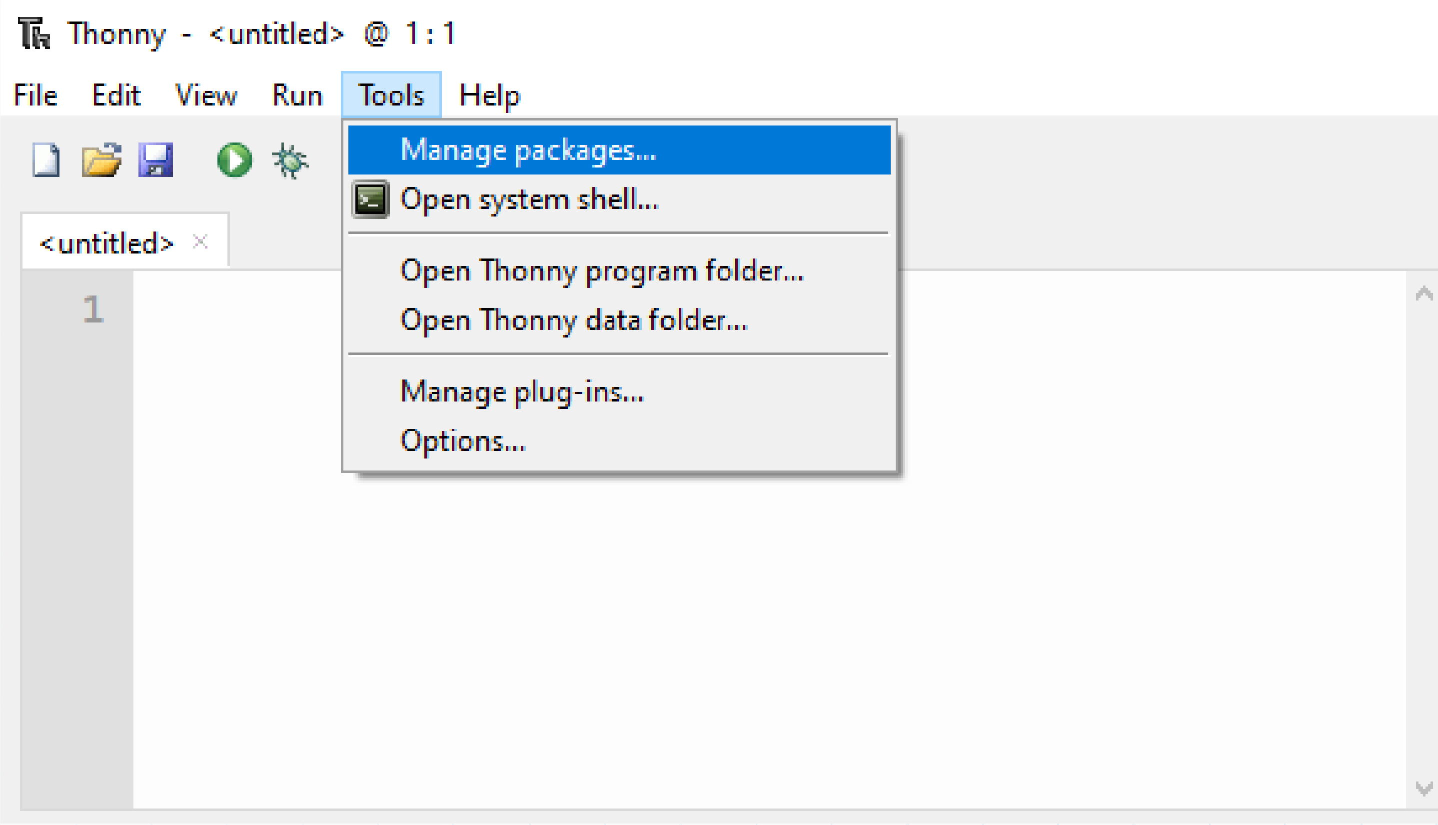
onde a saída é mostrada

Thonny

Python IDE for beginners

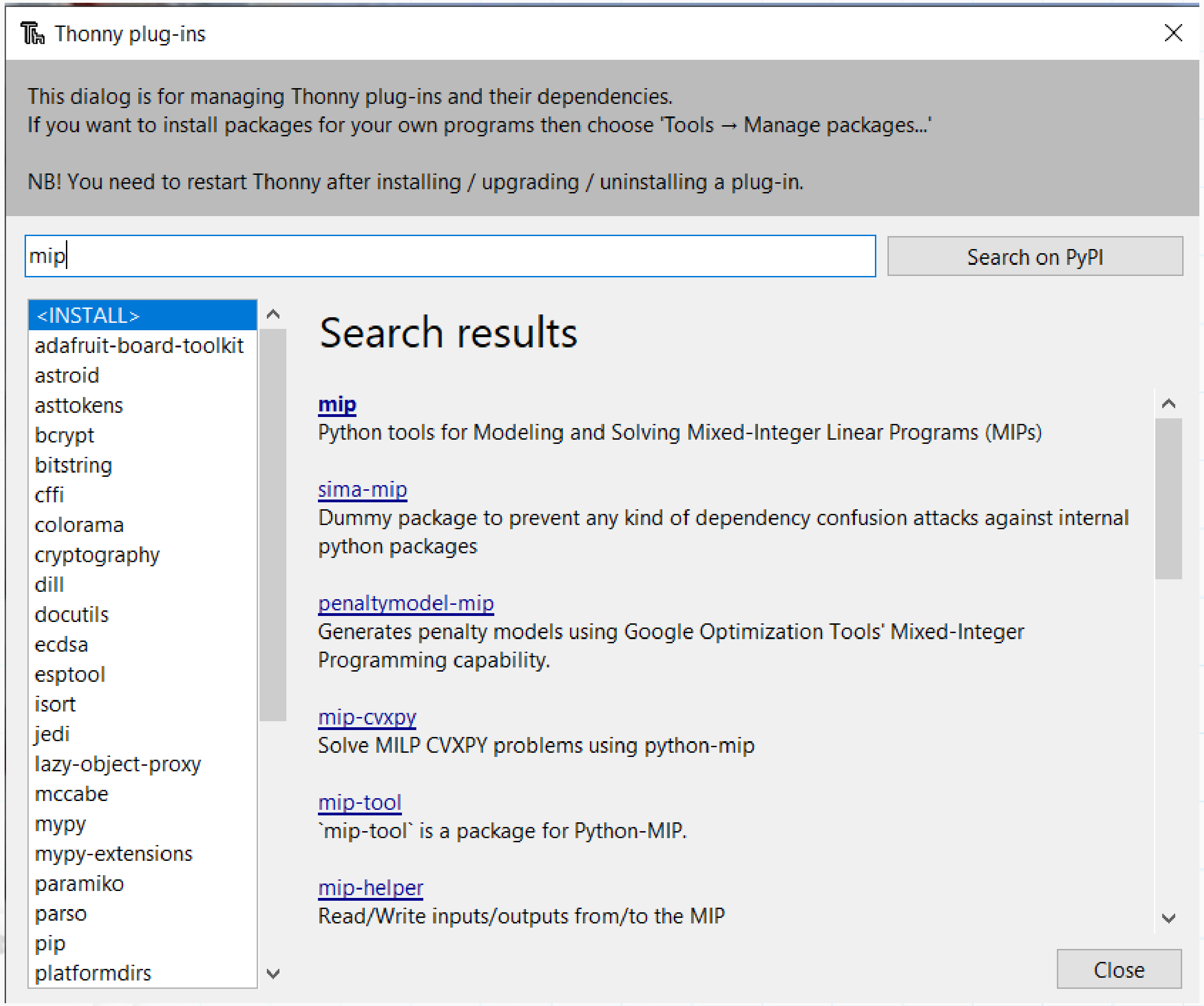
Python-MIP

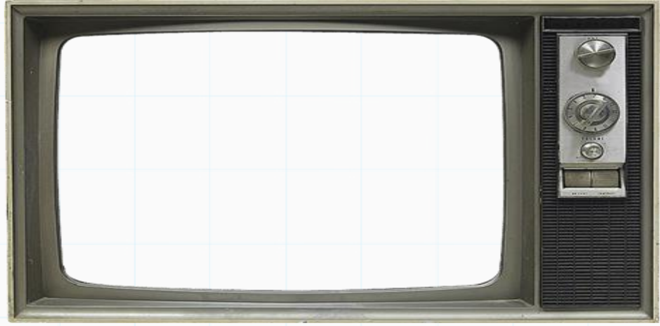
Instalando a biblioteca Python-MIP



Python-MIP

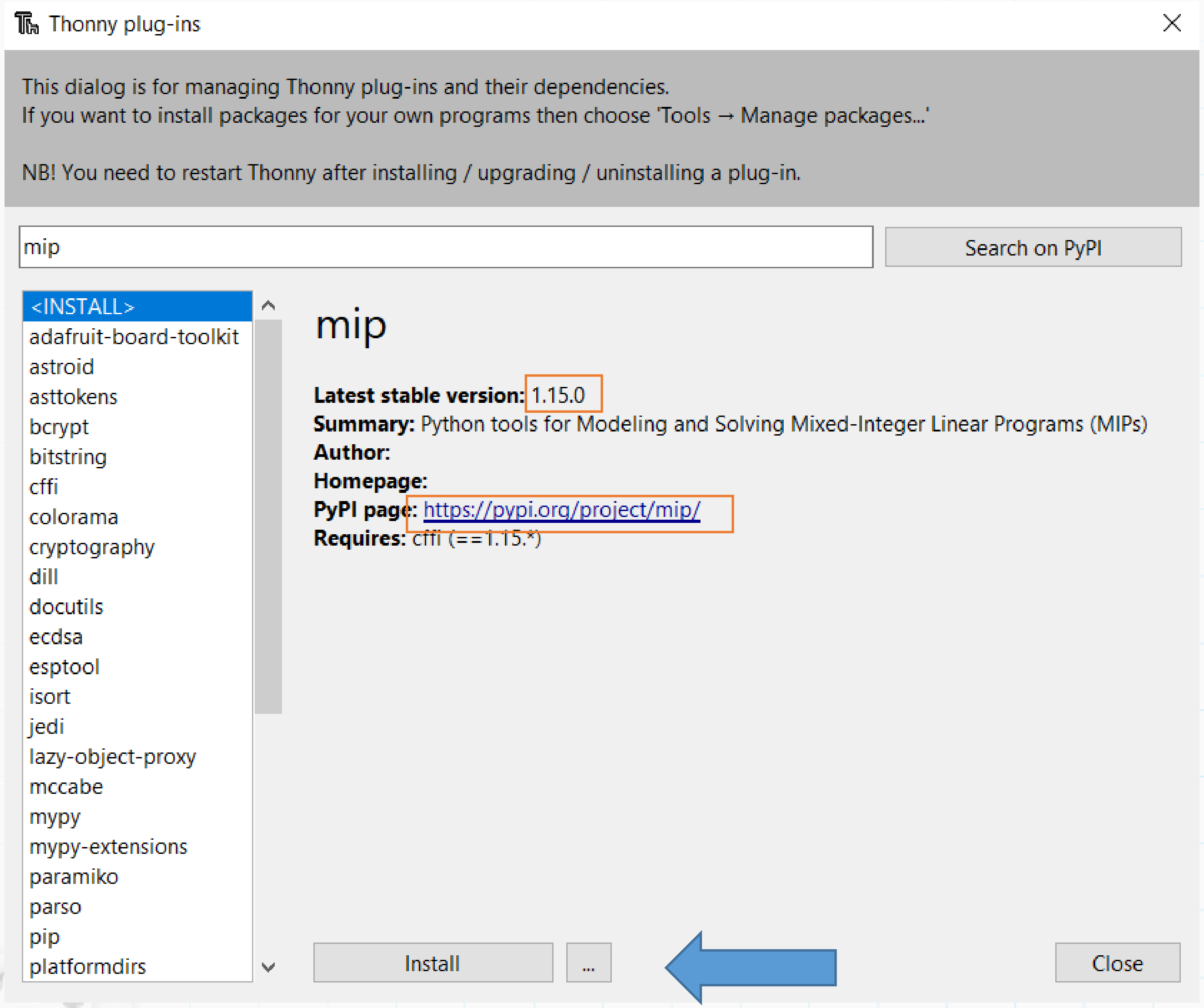
Instalando a biblioteca Python-MIP

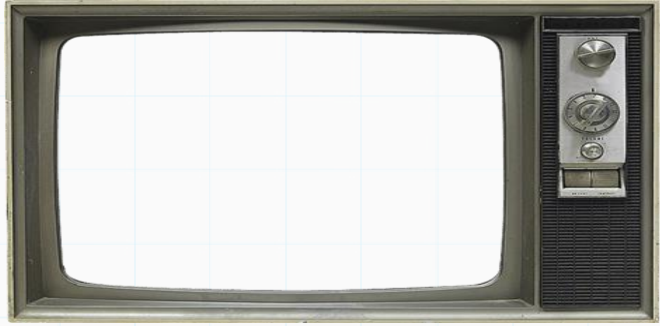




Python-MIP

Instalando a biblioteca Python-MIP





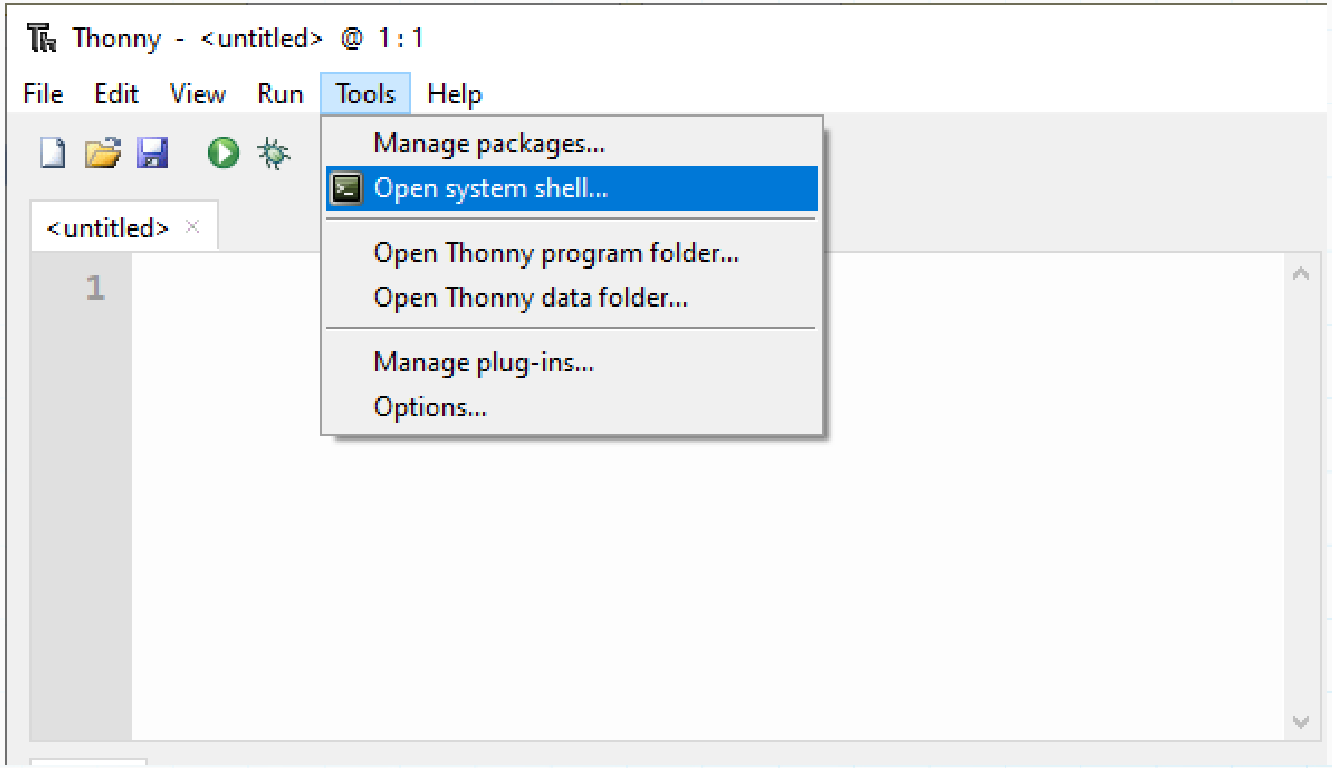
Pronto, acabou toda a instalação e está pronto para usar

NÃO ESQUEÇA DE REINICIAR O THONNY !

Python-MIP

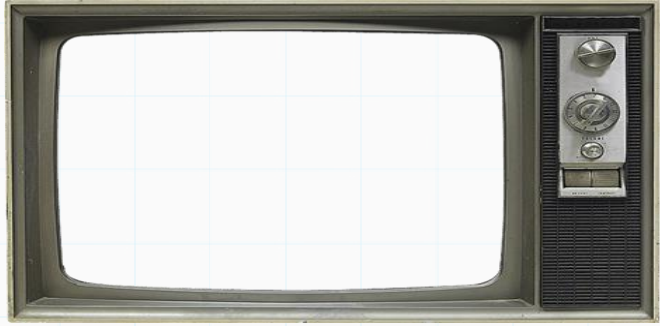
Instalando a biblioteca Python-MIP

Alguns computadores podem ter problema e não conseguir usar o gerenciador de pacotes



Nesse caso podemos instalar por linha de comando

Python-MIP



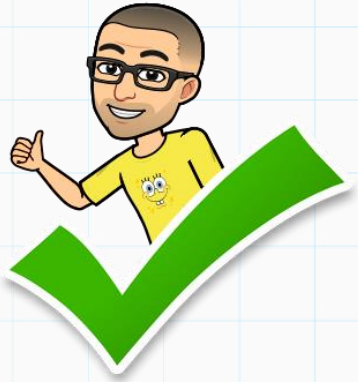
Instalando a biblioteca Python-MIP

Alguns computadores podem ter problema e não conseguir usar o gerenciador de pacotes

```
C:\Windows\system32\cmd.exe
*****************************************************************************
Some Python commands in the PATH of this session:
 - python    == C:\Users\yuri\AppData\Local\Programs\Thonny\python.exe
 - pip       == C:\Users\yuri\AppData\Local\Programs\Thonny\Scripts\pip.bat
 - pip3      == C:\Users\yuri\AppData\Local\Programs\Thonny\Scripts\pip3.bat
 - pip3.7    == C:\Users\yuri\AppData\Local\Programs\Thonny\Scripts\pip3.7.bat

*****************************************************************************

G:\Meu Drive\Cursos\PO\Yuri\13 - LAB Python-MIP\Coloração>pip install mip_
```

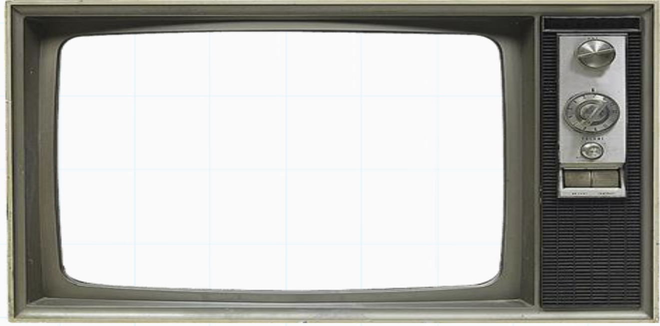
Nesse caso podemos instalar por linha de comando:

`pip install mip`



NÃO ESQUEÇA DE REINICIAR O THONNY !

Python-MIP



1-python_exemplo_formulacao: 4 exemplos para vocês testarem

Python-MIP

Ex1 (PPL):

$$
\begin{aligned}
\text{MAX} \quad & x0 + 2x1 + 3x2 \\
\text{s.a.} \quad & \\
& -x0 + x1 + x2 <= 20 \\
& x0 - 3x1 + x2 <= 30 \\
& x0 <= 40 \\
& x0, x1, x2 \; >= 0
\end{aligned}
$$

Python-MIP

Ex1 (PPL):

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

MAX $x0 + 2x1 + 3x2$

s.a.

$-x0 + x1 + x2 <= 20$

$x0 - 3x1 + x2 <= 30$

$x0 <= 40$

$x0,x1,x2 >=0$

- Importa biblioteca mip

Python-MIP

Ex1 (PPL):

MAX x0 + 2x1 + 3x2

s.a.

-x0 + x1 + x2 <= 20

x0 – 3x1 + x2 <= 30

x0 <= 40

x0,x1,x2 >=0

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

- Importa biblioteca mip
- Cria modelo

Python-MIP

Ex1 (PPL):

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

MAX x0 + 2x1 + 3x2

s.a.

-x0 + x1 + x2 <= 20

x0 – 3x1 + x2 <= 30

x0 <= 40

x0,x1,x2 >=0

- Importa biblioteca mip
- Cria modelo
- Cria variável e retorna referencia

Python-MIP

Ex1 (PPL):

MAX x0 + 2x1 + 3x2

s.a.

-x0 + x1 + x2 <= 20

x0 – 3x1 + x2 <= 30

x0 <= 40

x0,x1,x2 >=0

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

- Importa biblioteca mip
- Cria modelo
- Cria variável e retorna referencia
- Cria restrição

Python-MIP

Ex1 (PPL):

MAX $x0 + 2x1 + 3x2$

s.a.

$-x0 + x1 + x2 <= 20$

$x0 – 3x1 + x2 <= 30$

$x0 <= 40$

$x0,x1,x2 >=0$

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

- Importa biblioteca mip
- Cria modelo
- Cria variável e retorna referencia
- Cria restrição
- Define objetivo

Python-MIP

Ex1 (PPL):

```python
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

MAX $x0 + 2x1 + 3x2$

s.a.

$-x0 + x1 + x2 <= 20$

$x0 - 3x1 + x2 <= 30$

$x0 <= 40$

$x0,x1,x2 >=0$

- Importa biblioteca mip
- Cria modelo
- Cria variável e retorna referencia
- Cria restrição
- Define objetivo
- Otimiza e imprime saída

Python-MIP

Ex1 (PPL):

MAX x0 + 2x1 + 3x2

s.a.

-x0 + x1 + x2 <= 20

x0 – 3x1 + x2 <= 30

x0 <= 40

x0,x1,x2 >=0

```
from mip import *

# cria model
model = Mo

# variáve
x0 = mode
x1 = mode
x2 = mode

# restriç
model.add
model.add

# função
model.obje

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

Importa biblioteca mip

rencia

Saída

```
>>> %Run -c $EDITOR_CONTENT
 Welcome to the CBC MILP Solver
 Version: Trunk
 Build Date: Oct 28 2021

 Starting solution of the Linear programming problem using Primal Simplex

 sol =  202.5
```

Python-MIP

Ex1 (PPL):

MAX x0 + 2x1 + 3x2
s.a.

O CBC não vai funcionar com plataformas 32bits

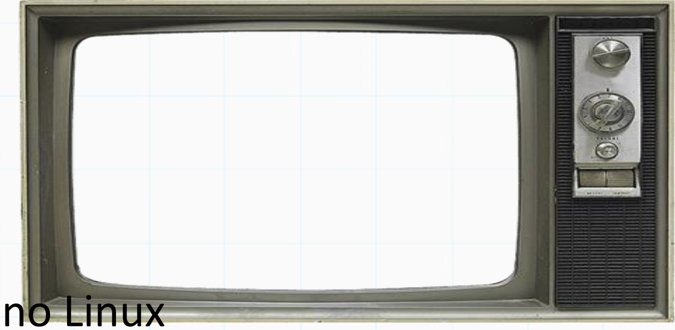
```
>>> %Run ex1.py

 An error occurred while loading the CBC library:      Win32 platform not supported.

 Traceback (most recent call last):
   File "C:\Users\yuri\OneDrive\Desktop\python exemplo formulacao\ex1.py", line 4, in <module>
     model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)
   File "C:\Users\yuri\AppData\Roaming\Python\Python37\site-packages\mip\model.py", line 87, in __init__
     import mip.cbc
   File "C:\Users\yuri\AppData\Roaming\Python\Python37\site-packages\mip\cbc.py", line 603, in <module>
     Osi_getNumCols = cbclib.Osi_getNumCols
 NameError: name 'cbclib' is not defined
```

pode tentar resolver em:

https://docs.python-mip.com/en/latest/install.html#using-your-own-cbc-binaries-optional

Ou tentar outro IDE (Pycharm, etc, ...)

```
# saida
print("sol = ", model.objective_value)
```



Python-MIP

- Caso não consiga rodar em Windows, pode tentar instalar Linux e depois instalar o Thonny no Linux

https://www.virtualbox.org/

https://www.vmware.com/in/products/workstation-player/workstation-player-evaluation.html

- Depois instalar linux

https://ubuntu.com/download/desktop

Python-MIP

Ex2 (PPI):

MAX $x0 + 2x1 + 3x2 + x3$

s.a.

$$-x0 + x1 + x2 + 10x3 <= 20$$
$$x0 - 3x1 + x2 <= 30$$
$$x1 - 3.5x3 = 0$$
$$x0 <= 40$$
$$x0,x1,x2 >= 0$$
$$2 <= x3 <= 3 \text{ e inteira}$$

Python-MIP

Ex2 (PPI):

```python
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)
x3 = model.add_var(name='x3', var_type=INTEGER, lb=2, ub=3)

# restrições
model.add_constr(-x0 + x1 + x2 + 10*x3 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest2')
model.add_constr(x1 - 3.5*x3 == 0, name='rest3')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2 + x3)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

MAX $x0 + 2x1 + 3x2 + x3$

s.a.

$-x0 + x1 + x2 + 10x3 <= 20$

$x0 – 3x1 + x2 <= 30$

$x1 – 3.5x3 = 0$

$x0 <= 40$

$x0,x1,x2 >=0$

$2 <= x3 <= 3$ e inteira

Python-MIP

```
>>> %Run ex2.py
 Welcome to the CBC MILP Solver
 Version: Trunk
 Build Date: Oct 28 2021

 Starting solution of the Linear programming relaxation problem using Primal Simplex

 Coin0506I Presolve 2 (-1) rows, 3 (-1) columns and 6 (-3) elements
 Clp1000I sum of infeasibilities 0 - average 0, 1 fixed columns
 Coin0506I Presolve 2 (0) rows, 2 (-1) columns and 4 (-2) elements
 Clp0029I End of values pass after 2 iterations
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0032I Optimal objective 125.2083333 - 0 iterations time 0.002, Presolve 0.00, Idiot 0.00

 Starting MIP optimization
 Cgl0004I processed model has 2 rows, 3 columns (1 integer (0 of which binary)) and 6 elements
 Coin3009W Conflict graph built in 0.000 seconds, density: 0.000%
 Cgl0015I Clique Strengthening extended 0 cliques, 0 were dominated
 Cbc0045I Nauty did not find any useful orbits in time 0
 Cbc0038I Initial state - 1 integers unsatisfied sum - 0.0833333
 Cbc0038I Pass   1: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass   2: suminf.    0.08333 (1) obj. -125.208 iterations 0

 Cbc0038I Pass  47: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  48: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  sol =  122.5
```

Prepoc

Python-MIP

```
>>> %Run ex2.py
 Welcome to the CBC MILP Solver
 Version: Trunk
 Build Date: Oct 28 2021

 Starting solution of the Linear programming relaxation problem using Primal Simplex

 Coin0506I Presolve 2 (-1) rows, 3 (-1) columns and 6 (-3) elements
 Clp1000I sum of infeasibilities 0 - average 0, 1 fixed columns
 Coin0506I Presolve 2 (0) rows, 2 (-1) columns and 4 (-2) elements
 Clp0029I End of values pass after 2 iterations
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0032I Optimal objective 125.2083333 - 0 iterations time 0.002, Presolve 0.00, Idiot 0.00

 Starting MIP optimization
 Cgl0004I processed model has 2 rows, 3 columns (1 integer (0 of which binary)) and 6 elements
 Coin3009W Conflict graph built in 0.000 seconds, density: 0.000%
 Cgl0015I Clique Strengthening extended 0 cliques, 0 were dominated
 Cbc0045I Nauty did not find any useful orbits in time 0
 Cbc0038I Initial state - 1 integers unsatisfied sum - 0.0833333
 Cbc0038I Pass   1: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass   2: suminf.    0.08333 (1) obj. -125.208 iterations 0

 Cbc0038I Pass  47: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  48: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  sol =  122.5
```

Cortes

Python-MIP

```
>>> %Run ex2.py
 Welcome to the CBC MILP Solver
 Version: Trunk
 Build Date: Oct 28 2021

 Starting solution of the Linear programming relaxation problem using Primal Simplex

 Coin0506I Presolve 2 (-1) rows, 3 (-1) columns and 6 (-3) elements
 Clp1000I sum of infeasibilities 0 - average 0, 1 fixed columns
 Coin0506I Presolve 2 (0) rows, 2 (-1) columns and 4 (-2) elements
 Clp0029I End of values pass after 2 iterations
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0032I Optimal objective 125.2083333 - 0 iterations time 0.002, Presolve 0.00, Idiot 0.00

 Starting MIP optimization
 Cgl0004I processed model has 2 rows, 3 columns (1 integer (0 of which binary)) and 6 elements
 Coin3009W Conflict graph built in 0.000 seconds, density: 0.000%
 Cgl0015I Clique Strengthening extended 0 cliques, 0 were dominated
 Cbc0045I Nauty did not find any useful orbits in time 0
 Cbc0038I Initial state - 1 integers unsatisfied sum - 0.0833333
 Cbc0038I Pass   1: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass   2: suminf.    0.08333 (1) obj. -125.208 iterations 0

 Cbc0038I Pass  47: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  48: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  sol =  122.5
```

Árvore de enumeração: B&C

Python-MIP

Ex3 (PPI):

ex3.lp - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
 x4
End
```

Existe um formato específico para escrever modelos (.lp)

Variáveis que não aparecem no Bounds tem limites entre 0 e +inf

General = var. inteiras
Binary = var. binárias

Python-MIP

Ex3 (PPI):

```
from mip import *

# cria modelo
model = Model(name="exemplo3",sense=MAXIMIZE, solver_name=CBC)

# le arquivo .lp
model.read('ex3.lp')
print('modelo tem', model.num_cols, 'variaveis')
print('e ', model.num_rows, ' restrições')

# otimiza
status = model.optimize()
print("\n",status)

# valores das variaveis
variaveis = model.vars

for i in range(len(variaveis)):
    print(variaveis[i].x)

# saida
print("sol = ", model.objective_value)
```

ex3.lp - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
 x4
End
```

-lê modelo, podemos ver também número de variáveis e restrições

Python-MIP

Ex3 (PPI):

```
from mip import *

# cria modelo
model = Model(name="exemplo3",sense=MAXIMIZE, solver_name=CBC)

# le arquivo .lp
model.read('ex3.lp')
print('modelo tem', model.num_cols, 'variaveis')
print('e ', model.num_rows, ' restrições')

# otimiza
status = model.optimize()
print("\n",status)

# valores das variaveis
variaveis = model.vars

for i in range(len(variaveis)):
    print(variaveis[i].x)

# saida
print("sol = ", model.objective_value)
```

ex3.lp - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
 x4
End
```

STATUS:

OptimizationStatus.OPTIMAL

OptimizationStatus.FEASIBLE

OptimizationStatus.NO_SOLUTION_FOUND

Python-MIP

Ex3 (PPI):

```
from mip import *

# cria modelo
model = Model(name="exemplo3",sense=MAXIMIZE, solver_name=CBC)

# le arquivo .lp
model.read('ex3.lp')
print('modelo tem', model.num_cols, 'variaveis')
print('e ', model.num_rows, ' restrições')

# otimiza
status = model.optimize()
print("\n",status)

# valores das variaveis
variaveis = model.vars

for i in range(len(variaveis)):
    print(variaveis[i].x)

# saida
print("sol = ", model.objective_value)
```

ex3.lp - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
 x4
End
```

- Retorna referencia para vetor de variáveis
- Cada variável possui um campo ‘x’ com o valor da variável.

Python-MIP

Ex3 (PPI):

```
from mip import *

# cria modelo
model = Model(name="exemplo3",sense=MAXIMIZE, solver_name=CBC)

# le arquivo .lp
model.read('ex3.lp')
print('modelo tem', model.num_cols, 'variaveis')
print('e ', model.num_rows, ' restrições')

# otimiza
status = model.optimize()
print("\n",status)

# valores das variaveis
variaveis = model.vars

for i in range(len(variaveis)):
    print(variaveis[i].x)

# saida
print("sol = ", model.objective_value)
```

ex3.lp - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
 x4
End
```

```
 OptimizationStatus.OPTIMAL
40.0
10.5
19.5
3.0
sol =  122.5
```

Python-MIP

Ex4 (PPI): Um modelo generalizado

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 1$$

$$\sum_{j \in N(i)} x_{ji} = 1$$

x_{ij} binário

Restrições apenas de conservação

G=(V,A)

N(i) -> vizinhos de i

Ex4 (PPI): Um modelo generalizado

Python-MIP

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 1$$

$$\sum_{j \in N(i)} x_{ji} = 1$$

x_{ij} binário

G=(V,A)

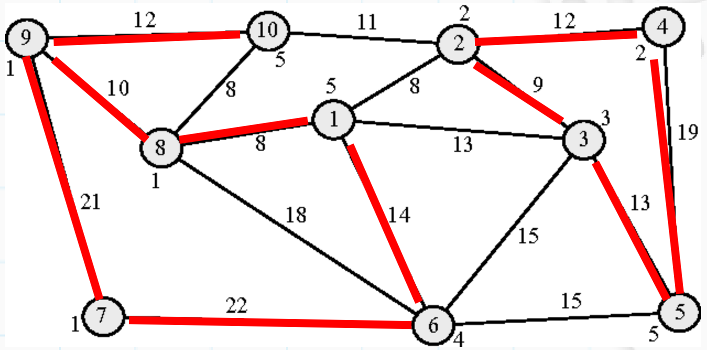
N(i) -> vizinhos de i

Restrições apenas de conservação

matriz de custos do problema já definida no código

```
n = 14
c = [[0, 83, 81, 113, 52, 42, 73, 44, 23, 91, 105, 90, 124, 57],
     [83, 0, 161, 160, 39, 89, 151, 110, 90, 99, 177, 143, 193, 100],
     [81, 161, 0, 90, 125, 82, 13, 57, 71, 123, 38, 72, 59, 82],
     [113, 160, 90, 0, 123, 77, 81, 71, 91, 72, 64, 24, 62, 63],
     [52, 39, 125, 123, 0, 51, 114, 72, 54, 69, 139, 105, 155, 62],
     [42, 89, 82, 77, 51, 0, 70, 25, 22, 52, 90, 56, 105, 16],
     [73, 151, 13, 81, 114, 70, 0, 45, 61, 111, 36, 61, 57, 70],
     [44, 110, 57, 71, 72, 25, 45, 0, 23, 71, 67, 48, 85, 29],
     [23, 90, 71, 91, 54, 22, 61, 23, 0, 74, 89, 69, 107, 36],
     [91, 99, 123, 72, 69, 52, 111, 71, 74, 0, 117, 65, 125, 43],
     [105, 177, 38, 64, 139, 90, 36, 67, 89, 117, 0, 54, 22, 84],
     [90, 143, 72, 24, 105, 56, 61, 48, 69, 65, 54, 0, 60, 44],
     [124, 193, 59, 62, 155, 105, 57, 85, 107, 125, 22, 60, 0, 97],
     [57, 100, 82, 63, 62, 16, 70, 29, 36, 43, 84, 44, 97, 0]]
```

Exemplo de uma solução (outra instancia)



Python-MIP

Ex4 (PPI): Um modelo generalizado

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 1$$

$$\sum_{j \in N(i)} x_{ji} = 1$$

x_{ij} binário

G=(V,A)

N(i) -> vizinhos de i

...

```
# tempo
inicio = time.process_time() # tempo de CPU

# cria modelo
model = Model(name="tsp",sense=MINIMIZE, solver_name=CBC)

# variaveis de rota x
x = [[model.add_var(name='x_'+str(i)+'_'+str(j), var_type=BINARY) for j in range(n)] for i in range(n)]

# função objetivo
exp = 0
for i in range(n):
    for j in range(n):
        exp += c[i][j]*x[i][j]
model.objective = minimize(exp)
```

...

Python-MIP

Ex4 (PPI): Um modelo generalizado

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 1$$

$$\sum_{j \in N(i)} x_{ji} = 1$$

x_{ij} binário

G=(V,A)

N(i) -> vizinhos de i

...

```
# tempo
inicio = time.process_time() # tempo de CPU

# cria modelo
model = Model(name="tsp",sense=MINIMIZE, solver_name=CBC)

# variaveis de rota x
x = [[model.add_var(name='x_'+str(i)+'_'+str(j), var_type=BINARY) for j in range(n)] for i in range(n)]

# função objetivo
exp = 0
for i in range(n):
    for j in range(n):
        exp += c[i][j]*x[i][j]
model.objective = minimize(exp)
```

...

Python-MIP

Ex4 (PPI): Um modelo generalizado

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 1$$

$$\sum_{j \in N(i)} x_{ji} = 1$$

x_{ij} binário

G=(V,A)

N(i) -> vizinhos de i

...

```
# tempo
inicio = time.process_time() # tempo de CPU

# cria modelo
model = Model(name="tsp",sense=MINIMIZE, solver_name=CBC)

# variaveis de rota x
x = [[model.add_var(name='x_'+str(i)+'_'+str(j), var_type=BINARY) for j in range(n)] for i in range(n)]

# função objetivo
exp = 0
for i in range(n):
    for j in range(n):
        exp += c[i][j]*x[i][j]
model.objective = minimize(exp)
```

Veja que estamos considerando grafo completo

...

Python-MIP

Ex4 (PPI): Um modelo generalizado

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N \quad (i)} x_{ij} = 1$$

$$\sum_{j \in N \quad (i)} x_{ji} = 1$$

x_{ij} binário

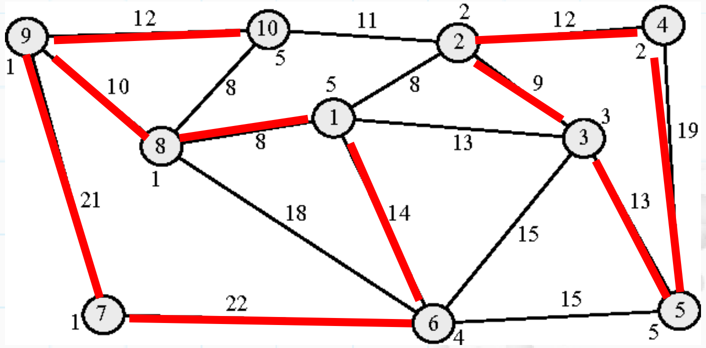
G=(V,A)

N(i) -> vizinhos de i

...

```
for i in range(n):
    exp = 0
    for j in range(n):
        if i != j:
            exp += x[i][j]
    model.add_constr(exp == 1, name='Out_'+str(i))
```

$$\sum_{j \in N^{+}(i)} x_{ij} = 1$$



...

Python-MIP

Ex4 (PPI): Um modelo generalizado

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$\sum_{j \in N\ (i)} x_{ij} = 1$

$\sum_{j \in N\ (i)} x_{ji} = 1$

x_{ij} binário

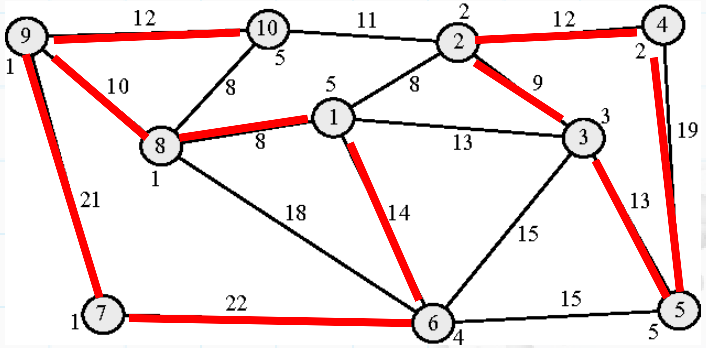
G=(V,A)

N(i) -> vizinhos de i

...

```
for i in range(n):
    exp = 0
    for j in range(n):
        if i != j:
            exp += x[i][j]
    model.add_constr(exp == 1, name='Out_'+str(i))
```

$\sum_{j \in N^+(i)} x_{ij} = 1$




```
for i in range(n):
    exp = 0
    for j in range(n):
        if i != j:
            exp += x[j][i]
    model.add_constr(exp == 1, name='In_'+str(i))
```

$\sum_{j \in N^-(i)} x_{ij} = 1$

...

Python-MIP

Ex4 (PPI): Um modelo generalizado

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 1$$

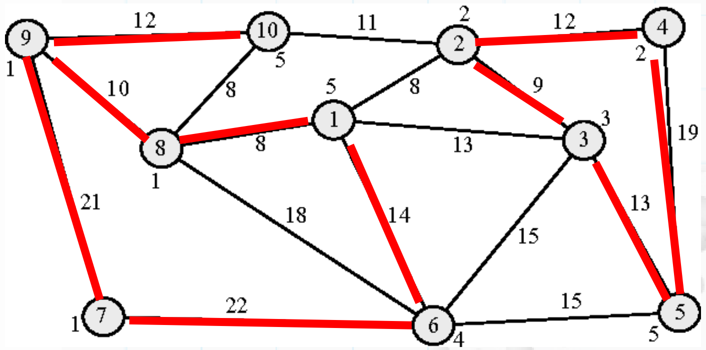
$$\sum_{j \in N(i)} x_{ji} = 1$$

x_{ij} binário

G=(V,A)

N(i) -> vizinhos de i

```python
#escreve o modelo
model.write('model.lp')

# otimiza
model.threads = 1          # -1 quantas tiverem 0 o valor default >0 específico

# otimiza
model.optimize(max_seconds=300) # limite de tempo

# saida
print("melhor solução  = ", model.objective_value)
print("limite inferior = ", model.objective_bound)
print("tempo           = ", time.process_time() - inicio)

for i in range(n):
    for j in range(n):
        if x[i][j].x >= 0.99:
            print(x[i][j].name)
```



Python-MIP

Ex4 (PPI): Um modelo generalizado

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 1$$

$$\sum_{j \in N(i)} x_{ji} = 1$$

x_{ij} binário

G=(V,A)

N(i) -> vizinhos de i

```python
#escreve o modelo
model.write('model.lp')
```

muito importante na hora de escrever o código do modelo para depurar !

```
\Problem name: tsp

Minimize
OBJROW: 83 x_0_1 + 81 x_0_2 + 113 x_0_3 + 52 x_0_4 + 42 x_0_5 + 73 x_0_6 + 44 x_0_7 + 23 x_0_8 + 91 x_0_9 + 105 x_0_10
 + 90 x_0_11 + 124 x_0_12 + 57 x_0_13 + 83 x_1_0 + 161 x_1_2 + 160 x_1_3 + 39 x_1_4 + 89 x_1_5 + 151 x_1_6 + 110 x_1_7
 + 90 x_1_8 + 99 x_1_9 + 177 x_1_10 + 143 x_1_11 + 193 x_1_12 + 100 x_1_13 + 81 x_2_0 + 161 x_2_1 + 90 x_2_3 + 125 x_2_4
 + 82 x_2_5 + 13 x_2_6 + 57 x_2_7 + 71 x_2_8 + 123 x_2_9 + 38 x_2_10 + 72 x_2_11 + 59 x_2_12 + 82 x_2_13 + 113 x_3_0
 + 160 x_3_1 + 90 x_3_2 + 123 x_3_4 + 77 x_3_5 + 81 x_3_6 + 71 x_3_7 + 91 x_3_8 + 72 x_3_9 + 64 x_3_10 + 24 x_3_11
 + 62 x_3_12 + 63 x_3_13 + 52 x_4_0 + 39 x_4_1 + 125 x_4_2 + 123 x_4_3 + 51 x_4_5 + 114 x_4_6 + 72 x_4_7 + 54 x_4_8
 + 69 x_4_9 + 139 x_4_10 + 105 x_4_11 + 155 x_4_12 + 62 x_4_13 + 42 x_5_0 + 89 x_5_1 + 82 x_5_2 + 77 x_5_3 + 51 x_5_4
 + 70 x_5_6 + 25 x_5_7 + 22 x_5_8 + 52 x_5_9 + 90 x_5_10 + 56 x_5_11 + 105 x_5_12 + 16 x_5_13 + 73 x_6_0 + 151 x_6_1
 + 13 x_6_2 + 81 x_6_3 + 114 x_6_4 + 70 x_6_5 + 45 x_6_7 + 61 x_6_8 + 111 x_6_9 + 36 x_6_10 + 61 x_6_11 + 57 x_6_12
 + 70 x_6_13 + 44 x_7_0 + 110 x_7_1 + 57 x_7_2 + 71 x_7_3 + 72 x_7_4 + 25 x_7_5 + 45 x_7_6 + 23 x_7_8 + 71 x_7_9
 + 67 x_7_10 + 48 x_7_11 + 85 x_7_12 + 29 x_7_13 + 23 x_8_0 + 90 x_8_1 + 71 x_8_2 + 91 x_8_3 + 54 x_8_4 + 22 x_8_5
 + 61 x_8_6 + 23 x_8_7 + 74 x_8_9 + 89 x_8_10 + 69 x_8_11 + 107 x_8_12 + 36 x_8_13 + 91 x_9_0 + 99 x_9_1 + 123 x_9_2
 + 72 x_9_3 + 69 x_9_4 + 52 x_9_5 + 111 x_9_6 + 71 x_9_7 + 74 x_9_8 + 117 x_9_10 + 65 x_9_11 + 125 x_9_12 + 43 x_9_13
 + 105 x_10_0 + 177 x_10_1 + 38 x_10_2 + 64 x_10_3 + 139 x_10_4 + 90 x_10_5 + 36 x_10_6 + 67 x_10_7 + 89 x_10_8 + 117 x_10_9
 + 54 x_10_11 + 22 x_10_12 + 84 x_10_13 + 90 x_11_0 + 143 x_11_1 + 72 x_11_2 + 24 x_11_3 + 105 x_11_4 + 56 x_11_5 + 61 x_11_6
 + 48 x_11_7 + 69 x_11_8 + 65 x_11_9 + 54 x_11_10 + 60 x_11_12 + 44 x_11_13 + 124 x_12_0 + 193 x_12_1 + 59 x_12_2 + 62 x_12_3
 + 155 x_12_4 + 105 x_12_5 + 57 x_12_6 + 85 x_12_7 + 107 x_12_8 + 125 x_12_9 + 22 x_12_10 + 60 x_12_11 + 97 x_12_13 + 57 x_13_0
 + 100 x_13_1 + 82 x_13_2 + 63 x_13_3 + 62 x_13_4 + 16 x_13_5 + 70 x_13_6 + 29 x_13_7 + 36 x_13_8 + 43 x_13_9 + 84 x_13_10
 + 44 x_13_11 + 97 x_13_12
Subject To
Out_0:  x_0_1 + x_0_2 + x_0_3 + x_0_4 + x_0_5 + x_0_6 + x_0_7 + x_0_8 + x_0_9 + x_0_10
 + x_0_11 + x_0_12 + x_0_13 = 1
Out_1:  x_1_0 + x_1_2 + x_1_3 + x_1_4 + x_1_5 + x_1_6 + x_1_7 + x_1_8 + x_1_9 + x_1_10
 + x_1_11 + x_1_12 + x_1_13 = 1
Out_2:  x_2_0 + x_2_1 + x_2_3 + x_2_4 + x_2_5 + x_2_6 + x_2_7 + x_2_8 + x_2_9 + x_2_10
 + x_2_11 + x_2_12 + x_2_13 = 1
Out_3:  x_3_0 + x_3_1 + x_3_2 + x_3_4 + x_3_5 + x_3_6 + x_3_7 + x_3_8 + x_3_9 + x_3_10
 + x_3_11 + x_3_12 + x_3_13 = 1
Out_4:  x_4_0 + x_4_1 + x_4_2 + x_4_3 + x_4_5 + x_4_6 + x_4_7 + x_4_8 + x_4_9 + x_4_10
 + x_4_11 + x_4_12 + x_4_13 = 1
Out_5:  x_5_0 + x_5_1 + x_5_2 + x_5_3 + x_5_4 + x_5_6 + x_5_7 + x_5_8 + x_5_9 + x_5_10
 + x_5_11 + x_5_12 + x_5_13 = 1
Out_6:  x_6_0 + x_6_1 + x_6_2 + x_6_3 + x_6_4 + x_6_5 + x_6_7 + x_6_8 + x_6_9 + x_6_10
 + x_6_11 + x_6_12 + x_6_13 = 1
Out_7:  x_7_0 + x_7_1 + x_7_2 + x_7_3 + x_7_4 + x_7_5 + x_7_6 + x_7_8 + x_7_9 + x_7_10
 + x_7_11 + x_7_12 + x_7_13 = 1
Out_8:  x_8_0 + x_8_1 + x_8_2 + x_8_3 + x_8_4 + x_8_5 + x_8_6 + x_8_7 + x_8_9 + x_8_10
 + x_8_11 + x_8_12 + x_8_13 = 1
Out_9:  x_9_0 + x_9_1 + x_9_2 + x_9_3 + x_9_4 + x_9_5 + x_9_6 + x_9_7 + x_9_8 + x_9_10
```

Python-MIP

Ex4 (PPI): Um modelo generalizado

$$\text{MIN} \sum_{ij\in A} c_{ij}x_{ij}$$

sujeito a:

$$\sum_{j\in N\ \ (i)} x_{ij} = 1$$

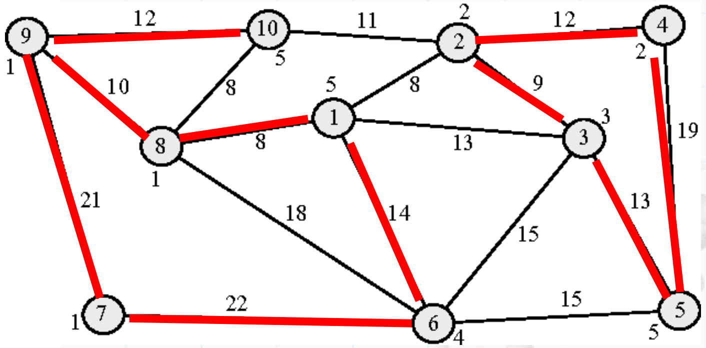
$$\sum_{j\in N\ \ (i)} x_{ji} = 1$$

$$x_{ij} \text{ binário}$$

G=(V,A)

N(i) -> vizinhos de i

```
#escreve o modelo
model.write('model.lp')

# otimiza
model.threads = 1          # -1 quantas tiverem 0 o valor default >0 específico

# otimiza
model.optimize(max_seconds=300) # limite de tempo

# saida
print("melhor solução  = ", model.objective_value)
print("limite inferior = ", model.objective_bound)
print("tempo           = ", time.process_time() - inicio)

for i in range(n):
    for j in range(n):
        if x[i][j].x >= 0.99:
            print(x[i][j].name)
```



Ex4 (PPI): Um modelo generalizado

Python-MIP

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 1$$

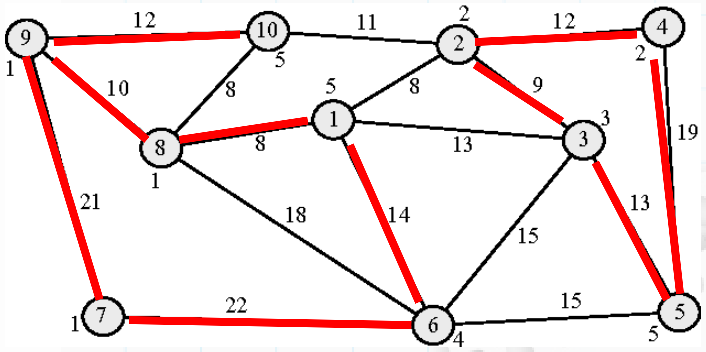
$$\sum_{j \in N(i)} x_{ji} = 1$$

x_{ij} binário

G=(V,A)

N(i) -> vizinhos de i

```
#escreve o modelo
model.write('model.lp')

# otimiza
model.threads = 1          # -1 quantas tiverem 0 o valor default >0 específico

# otimiza
model.optimize(max_seconds=300) # limite de tempo

# saida
print("melhor solução  = ", model.objective_value)
print("limite inferior = ", model.objective_bound)
print("tempo           = ", time.process_time() - inicio)

for i in range(n):
    for j in range(n):
        if x[i][j].x >= 0.99:
            print(x[i][j].name)
```



Ex4 (PPI): Um modelo generalizado

Python-MIP

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 1$$

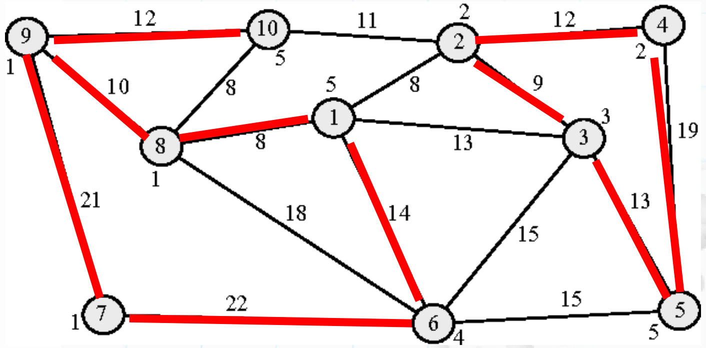
$$\sum_{j \in N(i)} x_{ji} = 1$$

x_{ij} binário

G=(V,A)

N(i) -> vizinhos de i

```
#escreve o modelo
model.write('model.lp')

# otimiza
model.threads = 1          # -1 quantas tiverem 0 o valor default >0 específico

# otimiza
model.optimize(max_seconds=300) # limite de tempo

# saida
print("melhor solução  = ", model.objective_value)
print("limite inferior = ", model.objective_bound)
print("tempo           = ", time.process_time() - inicio)

for i in range(n):
    for j in range(n):
        if x[i][j].x >= 0.99:
            print(x[i][j].name)
```



Python-MIP

Ex4 (PPI): Um modelo generalizado

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 1$$

$$\sum_{j \in N(i)} x_{ji} = 1$$

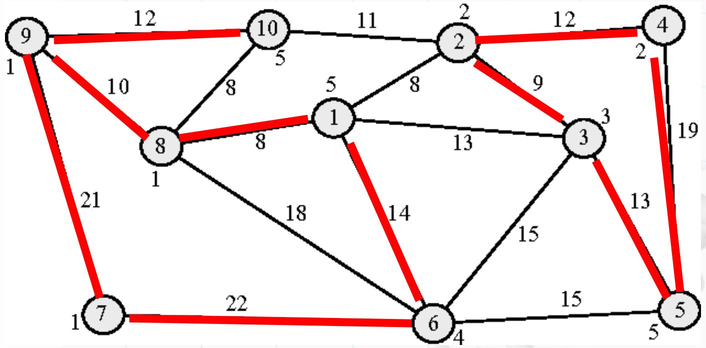
$$x_{ij} \text{ binário}$$

G=(V,A)

N(i) -> vizinhos de i

```
#escreve o modelo
model.write('model.lp')

# otimiza
model.threads = 1          # -1 quantas tiverem 0 o valor default >0 específico

# otimiza
model.optimize(max_seconds=300) # limite de tempo

# saida
print("melhor solução  = ", model.objective_value)
print("limite inferior = ", model.objective_bound)
print("tempo           = ", time.process_time() - inicio)
```

```
for i in range(n):
    for j in range(n):
        if x[i][j].x >= 0.99:
            print(x[i][j].name)
```



Python-MIP

Ex4 (PPI): Um modelo generalizado

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 1$$

$$\sum_{j \in N(i)} x_{ji} = 1$$

x_{ij} binário

G=(V,A)

N(i) -> vizinhos de i

```python
#escreve o modelo
model.write('model.lp')

# otimiza
model.threads = 1         # -1 quantas tiverem 0 o valor default >0 específico

# otimiza
model.optimize(max_seconds=300) # limite de tempo

# saida
print("melhor solução  = ", model.objective_value)
print("limite inferior = ", model.objective_bound)
print("tempo           = ", time.process_time() - inicio)

for i in range(n):
    for j in range(n):
        if x[i][j].x >= 0.99:
            print(x[i][j].name)
```

Saída

```
melhor solução  =  378.0
limite inferior =  378.0
tempo           =  0.09375
x_0_8
x_1_4
x_2_6
x_3_11
x_4_1
x_5_7
x_6_2
x_7_5
x_8_0
x_9_13
x_10_12
x_11_3
x_12_10
x_13_9
```

Python-MIP

... e se a variável tivesse 3 índices, como eu declararia ?

MIN $\sum_{ij \in A} c_{ij} x_{ij}$

sujeito a:

$$\sum_{j \in N(i)} x_{ijk} = 1$$

$$\sum_{j \in N(i)} x_{jik} = 1$$

x_{ij} binário

Variáveis com 2 índices

```
# variaveis de rota x
x = [[model.add_var(name='x_'+str(i)+'_'+str(j), var_type=BINARY) for j in range(n)] for i in range(n)]
```

Variáveis com 3 índices

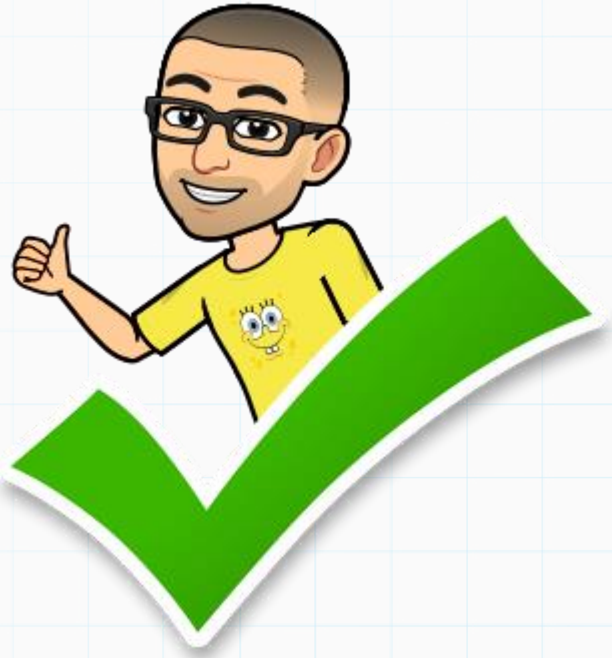
```
# variaveis de rota x
x = [[[model.add_var(name='x_'+str(i)+'_'+str(j)+'_'+str(k), var_type=BINARY) for k in range(n) ] for j in range(n)] for i in range(n)]
```

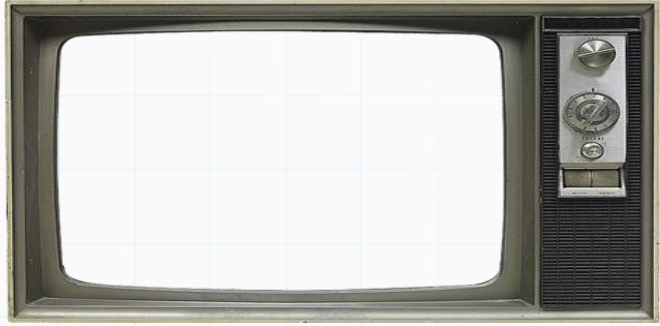
Variáveis com 4 índices

...

Python-MIP

- Documentação: [link]



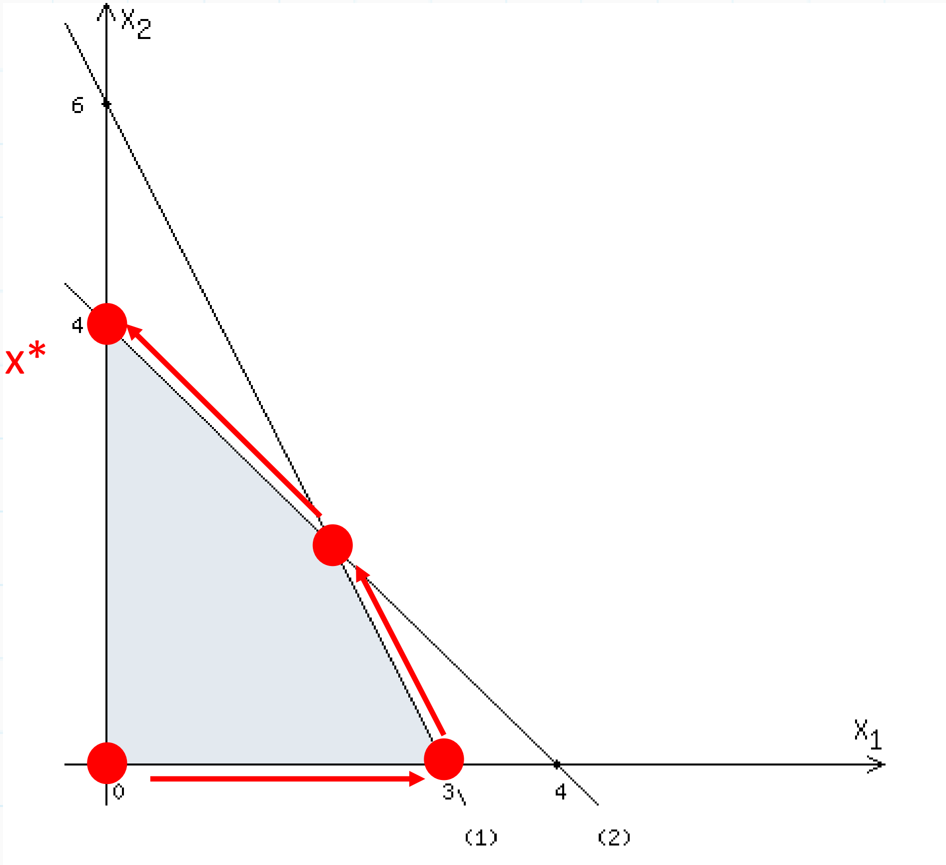


Exercício

PPL1: Vamos ver se conseguimos implementar e resolver esse PPL que já resolvemos com o Simplex anteriormente, para alcançar a mesma solução.

$$\max \; x_1 + 2x_2$$
$$\text{s.a.} \; 2x_1 + x_2 \leq 6$$
$$x_1 + x_2 \leq 4$$
$$x_1, \; x_2 \geq 0$$

$x^*_1=0, x^*_2=4$, com $Z^*=8$

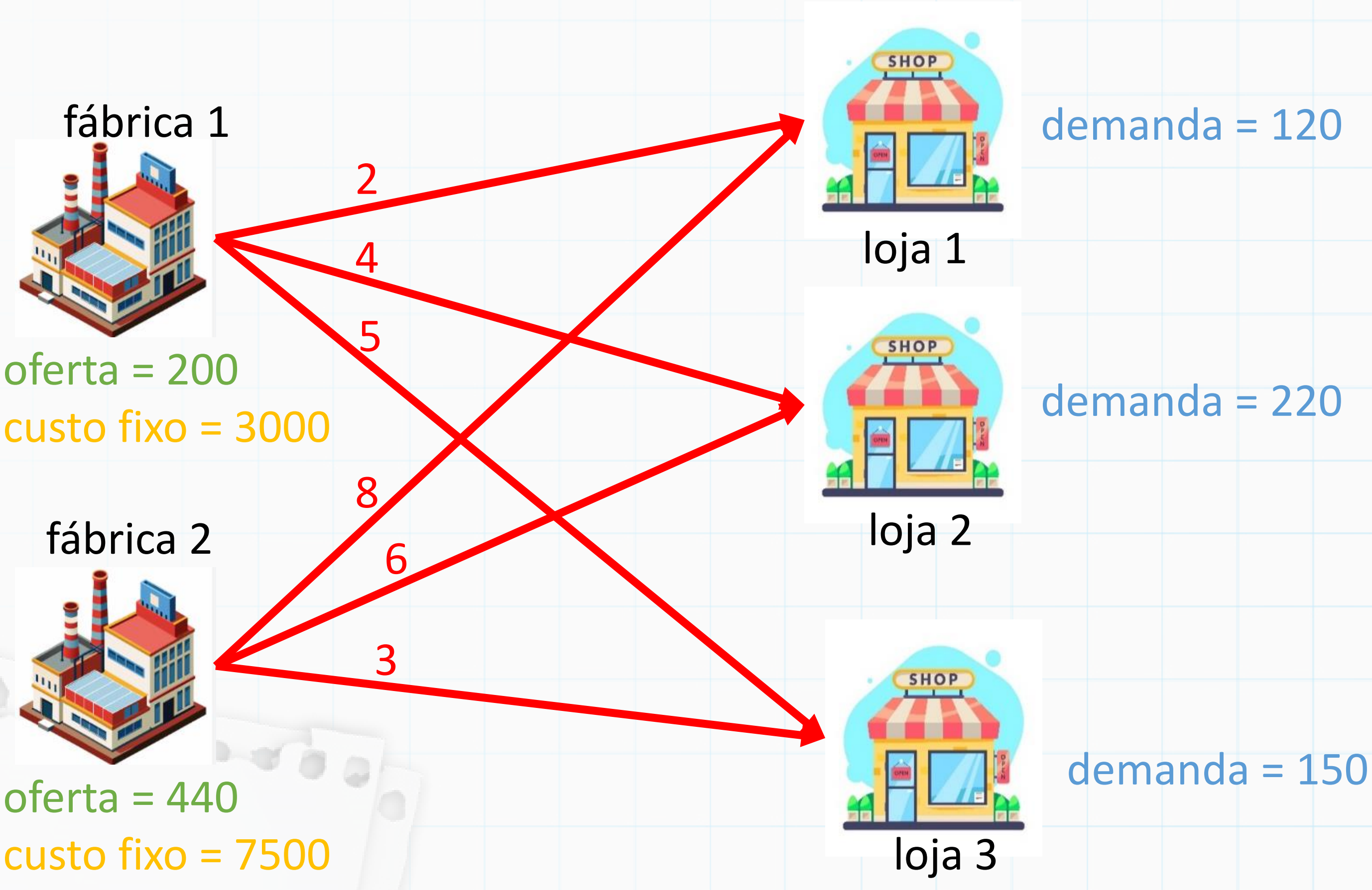


A pergunta é, qual a nova solução se introduzirmos uma nova restrição:

x2 <= 3.5

Exercício

Dolly: A empresa de guaraná Dolly possui 3 lojas para venda de seus produtos. E cada loja possui uma demanda (em azul) especifica (em garrafas). Para atender essa demanda, a empresa possui 2 fábricas, cada uma com uma capacidade de oferta (em verde) de garrafas de Dolly. Além disso, existe um custo unitário de transporte (por garrafa) entre as fábricas e as lojas (em vermelho). Mais ainda, se a fábrica realizar qualquer entrega, existe um custo fixo (em laranja) a ser pago (não importa se entregou 1 ou 1000 garrafas) além do custo por unidade. Queremos ajudar a empresa Dolly a estabelecer as entregas de forma que : (1) toda a demanda das lojas é atendida (2) os limites de oferta das fábricas não são ultrapassados (3) as entregas foram feitas ao menor custo possível.



Modelo a seguir:

xij -> qtd de garrafas enviadas da fabrica i para a loja j (i=1,2 e j=1,2,3)
yi -> se a fabrica i realizou alguma entrega ou não (i=1,2)

Exercício

$$\min \quad 2x_{11} + 4x_{12} + 5x_{13} + 8x_{21} + 6x_{22} + 3x_{23} + 3000y_1 + 7500y_2$$

$$x_{11} + x_{21} = 120$$

$$x_{12} + x_{22} = 220$$

$$x_{13} + x_{23} = 150$$

$$x_{11} + x_{12} + x_{13} \leq 200 \cdot y_1$$

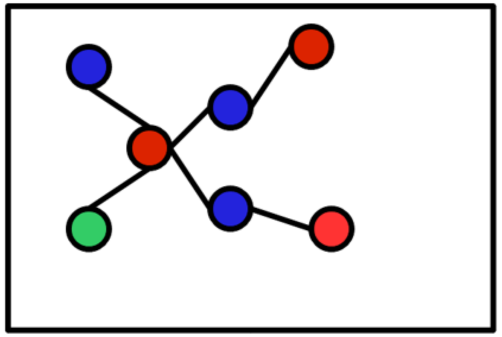
$$x_{21} + x_{22} + x_{23} \leq 440 \cdot y_2$$

$$x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23} \in Z^+ \text{ and } y_1, y_2 \in \{0, 1\}$$

Qual deve ser os valores das variáveis no ponto ótimo ? Sabendo que o valor ótimo é 12350.

Exercício

<u>Coloração</u>: Nas aulas nos modelamos o problema de atribuir frequências as antenas de forma que não gerem interferência. Este problema na verdade é o clássico problema de coloração de vértices. Seja G=(V,E) um grafo com n vértices e m arestas. Queremos atribuir uma das F cores para cada vértice, de modo que 2 vértices adjacentes não tenham a mesma cor, de forma que o menor número de cores é usada.



$$\min \sum_{f=1}^{F} w_f$$

$$\sum_{f=1}^{F} x_{if} = 1, \quad \forall i \in V$$

$$x_{if} + x_{jf} \leq 1, \quad \forall ij \in E, \forall f \in \{1...F\}$$

$$x_{if} \leq w_f, \quad \forall i \in V, \forall f \in \{1...F\}$$

$$x_{if} \in \{0,1\}, \quad \forall i \in V, \forall f \in \{1...F\}$$

$$w_f \in \{0,1\}, \quad \forall f \in \{1...F\}$$

Variáveis: x_ij -> se vértice i ∈ V recebe a cor j ∈ F, ou não
w_j -> se a cor j ∈ J é utilizada

Exercício

- resolver a instancia fornecida: TPI_COL_1.txt

Formato do arquivo:

p edge n (numero de vertices) m (numero de arestas) F (número de cores)
e vertice(i) vertice(j) (lista de arestas)

```
p edge 11 20 5
e 1 2
e 1 4
e 1 7
e 1 9
e 2 3
e 2 6
e 2 8
e 3 5
e 3 7
e 3 10
e 4 5
e 4 6
e 4 10
e 5 8
e 5 9
e 6 11
e 7 11
e 8 11
e 9 11
e 10 11
```

TPI_COL_1.txt

Exercício

- resolver a instancia fornecida: TPI_COL_1.txt

Formato do arquivo:

p edge n (numero de vertices) m (numero de arestas) F (número de cores)
e vertice(i) vertice(j) (lista de arestas)

- Dica de Implementação:
  - Use o arquivo base fornecido que já tem a leitura dos dados do arquivo.
  - Escrever o modelo no papel (ou parte dele) já que a instância é pequena.
  - A cada restrição implementada imprima o modelo e cheque a construção da restrição com a que você escreveu no papel, faça isso com as variáveis também.

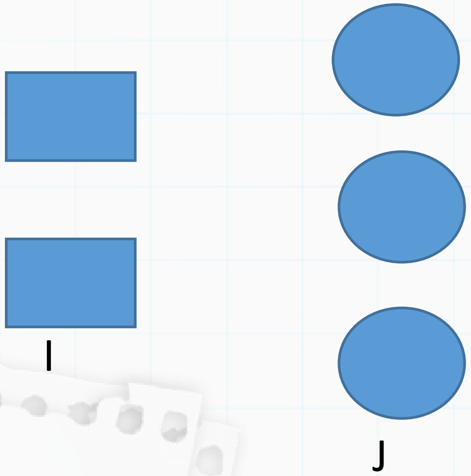
Qual deve ser os valores das variáveis no ponto ótimo ?
Sabendo que o valor mínimo de cores desta instancia é 4.

```
p edge 11 20 5
e 1 2
e 1 4
e 1 7
e 1 9
e 2 3
e 2 6
e 2 8
e 3 5
e 3 7
e 3 10
e 4 5
e 4 6
e 4 10
e 5 8
e 5 9
e 6 11
e 7 11
e 8 11
e 9 11
e 10 11
```

TPI_COL_1.txt

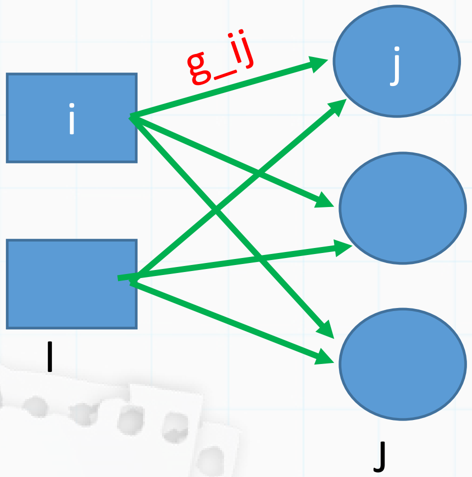
Exercício

Facilidades: Seja o conjunto de clientes J que precisam ser atendidos, e o conjunto I de fábricas que podem atender os clientes.



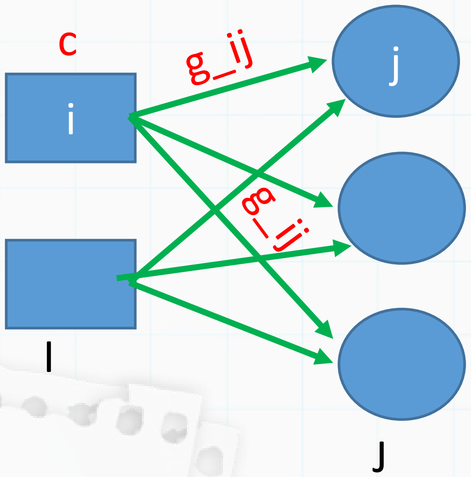
Exercício

<u>Facilidades:</u> Seja o conjunto de clientes J que precisam ser atendidos, e o conjunto I de fábricas que podem atender os clientes. Para cada fábrica $i \in I$ e para cada cliente $j \in J$ chamamos por g_{ij} o custo de atendimento total do cliente pela fábrica.



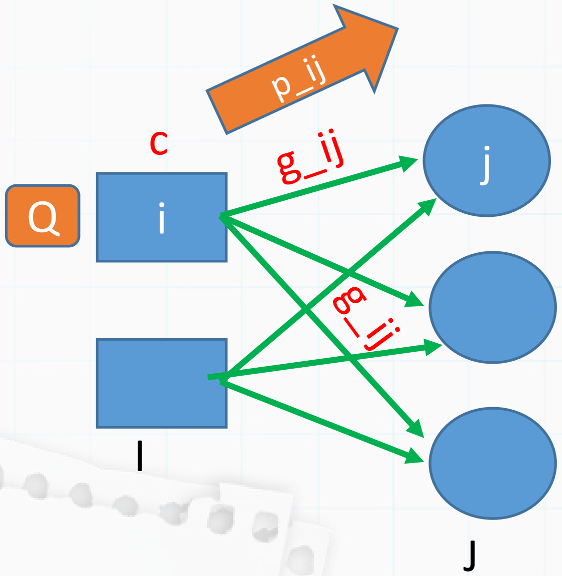
Exercício

<u>Facilidades:</u> Seja o conjunto de clientes J que precisam ser atendidos, e o conjunto I de fábricas que podem atender os clientes. Para cada fábrica $i \in I$ e para cada cliente $j \in J$ chamamos por g_{ij} o custo de atendimento total do cliente pela fábrica. Porém, se a fábrica $i \in I$ atender um ou mais clientes, se tem um custo fixo de atendimento c (o mesmo para todas as fábricas).



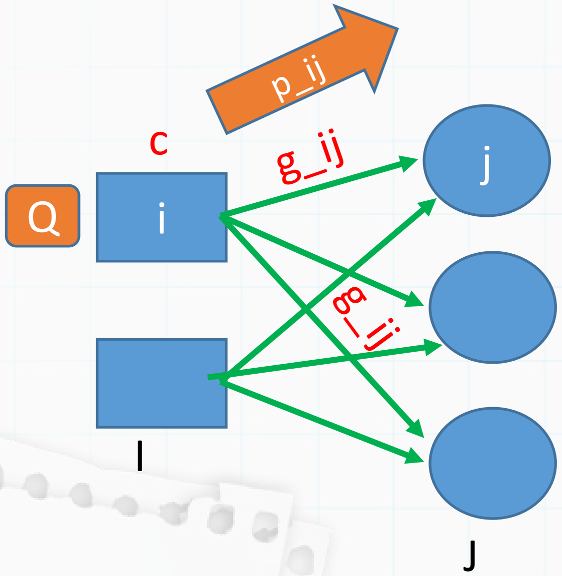
Exercício

Facilidades: Seja o conjunto de clientes J que precisam ser atendidos, e o conjunto I de fábricas que podem atender os clientes. Para cada fábrica i ∈ I e para cada cliente j ∈ J chamamos por g_{ij} o custo de atendimento total do cliente pela fábrica. Porém, se a fábrica i ∈ I atender um ou mais clientes, se tem um custo fixo de atendimento c (o mesmo para todas as fábricas). Além disso, cada fábrica possui uma quantidade de recurso Q para atendimento, porém se a fábrica i ∈ I atender o cliente j ∈ J, este atendimento irá consumir uma quantidade pij de recursos da fábrica i ∈ I.



Exercicio

Facilidades: Seja o conjunto de clientes J que precisam ser atendidos, e o conjunto I de fábricas que podem atender os clientes. Para cada fábrica $i \in I$ e para cada cliente $j \in J$ chamamos por g_{ij} o custo de atendimento total do cliente pela fábrica. Porém, se a fábrica $i \in I$ atender um ou mais clientes, se tem um custo fixo de atendimento c (o mesmo para todas as fábricas). Além disso, cada fábrica possui uma quantidade de recurso Q para atendimento, porém se a fábrica $i \in I$ atender o cliente $j \in J$, este atendimento irá consumir uma quantidade pij de recursos da fábrica $i \in I$. Precisamos definir que fábricas atenderão que clientes, minimizando os custos de atendimento e os custos fixos.



Exercício

Variáveis: x_ij -> se fábrica i ∈ I atende cliente j ∈ J, ou não (custo de atendimento)
y_i -> se fábrica i ∈ I atende alguém, ou não (custo fixo)

$$\min \sum_{i \in I} c y_i + \sum_{(i,j) \in E} g_{ij} x_{ij} \qquad (1)$$

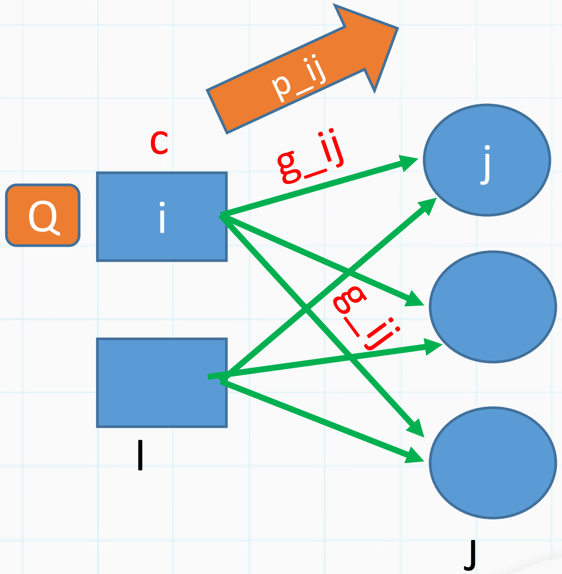
$$\sum_{i \in I} x_{ij} = 1, \qquad \forall j \in J \qquad (2)$$

$$x_{ij} \leq y_i, \qquad \forall (i,j) \in E \qquad (3)$$

$$\sum_{(i,j) \in E} p_{ij} x_{ij} \leq Q y_i, \qquad \forall i \in I \qquad (4)$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E \qquad (5)$$

$$y_i \in \{0,1\}, \qquad \forall i \in I \qquad (6)$$



Exercício

- resolver a instancia fornecida: TPI_F_0.txt

Formato do arquivo:

ni (numero de facilidades) nj (numero de clientes) c (custo de abrir facilidade) Q (quantidade de recursos em cada facilidade)
NL (numero de linhas de facilidades/clientes a seguir)

facilidade(i) cliente(j) g_ij p_ij

```
2 3 50 5 5
1 1 20 2
1 2 15 2
1 3 10 2
2 1 40 2
2 3 30 2
```

TPI_F_0.txt

Exercício

- resolver a instancia fornecida: TPI_F_0.txt

Formato do arquivo:

ni (numero de facilidades) nj (numero de clientes) c (custo de abrir facilidade) Q (quantidade de recursos em cada facilidade)
NL (numero de linhas de facilidades/clientes a seguir)

facilidade(i) cliente(j) g_ij p_ij

- Dica de Implementação:
    - Use o arquivo base fornecido que já tem a leitura dos dados do arquivo.
    - Escrever o modelo no papel (ou parte dele) já que a instância é pequena.
    - A cada restrição implementada imprima o modelo e cheque a construção da restrição com a que você escreveu no papel, faça isso com as variáveis também.

2 3 50 5 5
1 1 20 2
1 2 15 2
1 3 10 2
2 1 40 2
2 3 30 2

TPI_F_0.txt

Qual deve ser os valores das variáveis no ponto ótimo ? Sabendo que o valor ótimo é 165.

Até a próxima