

Programação Inteira

Professor : Yuri Frota

www.ic.uff.br/~yuri/pi.html

yuri@ic.uff.br



Solvers



X



Solvers



X



Solvers



X



Solvers



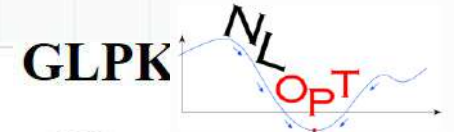
X



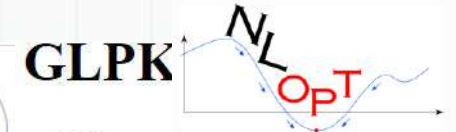
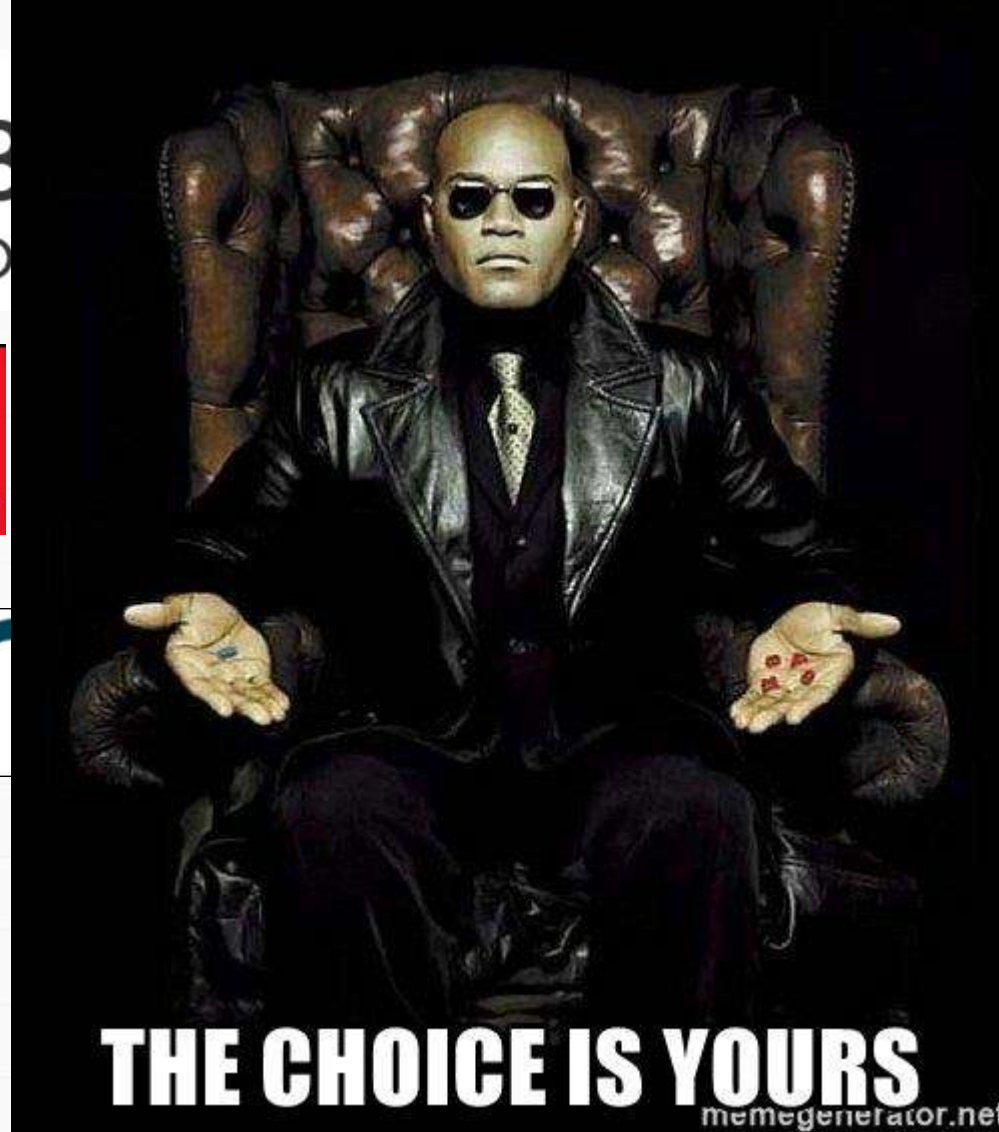
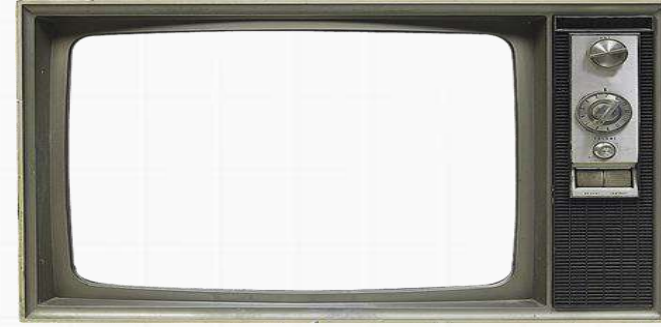
Solvers



X



Solvers



Python-MIP



- No curso vamos ver dois solvers:
 - Python-MIP: 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.

Túlio A. M. Toffolo

Personal website



Haroldo G. Santos

Personal website



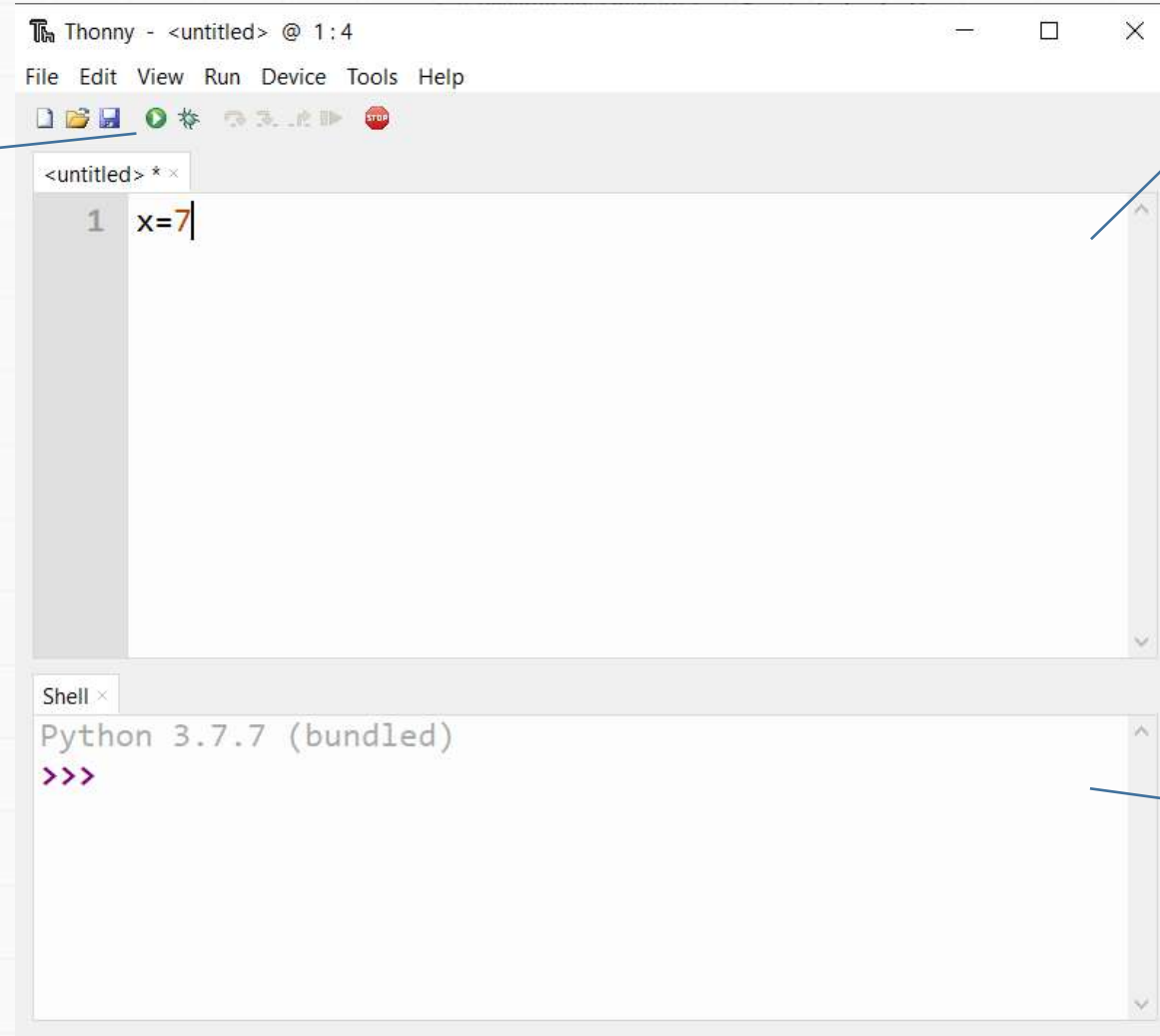
Python-MIP

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



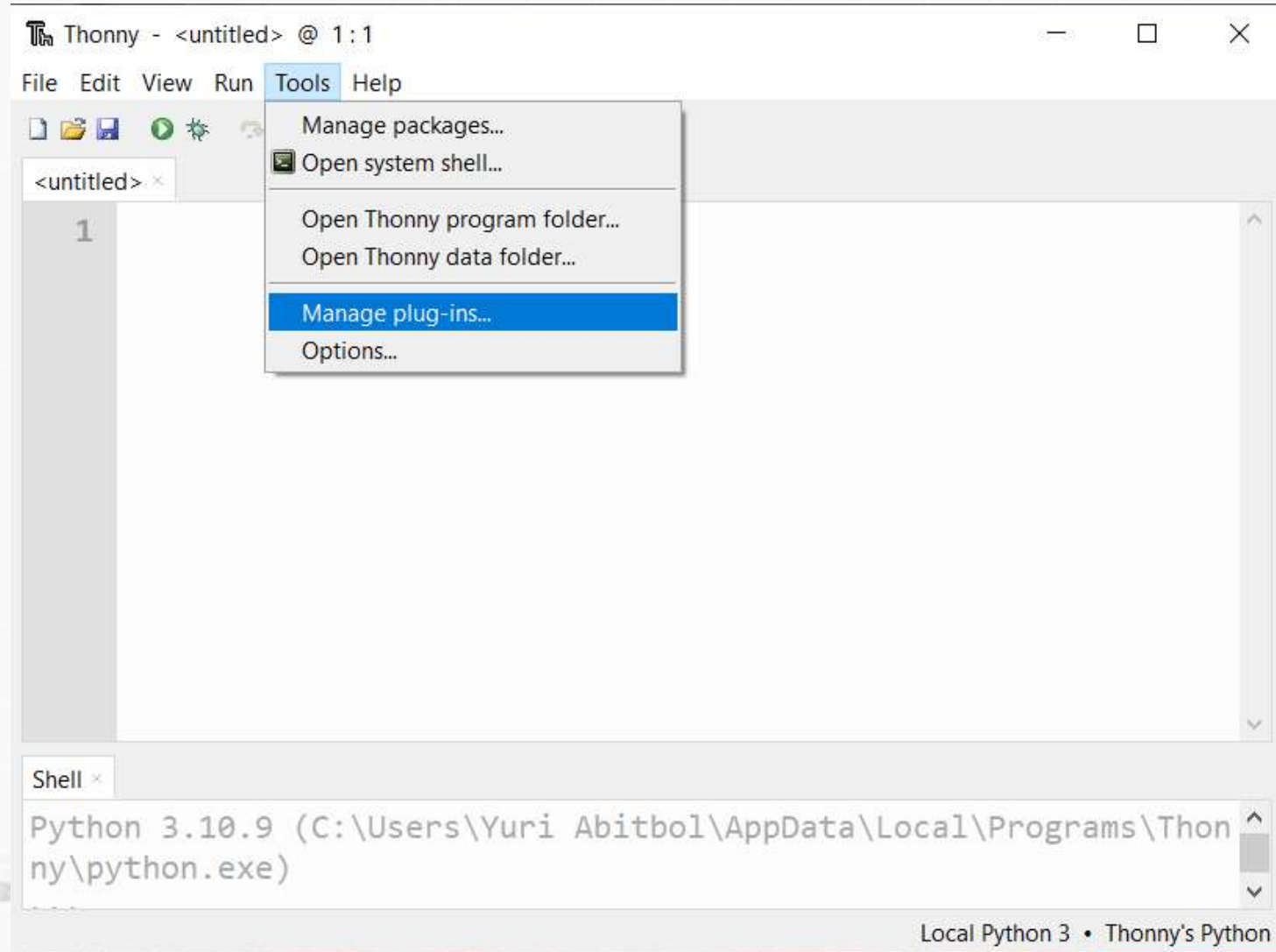
onde o código
é escrito

Thonny
Python IDE for beginners

onde a saída é
mostrada

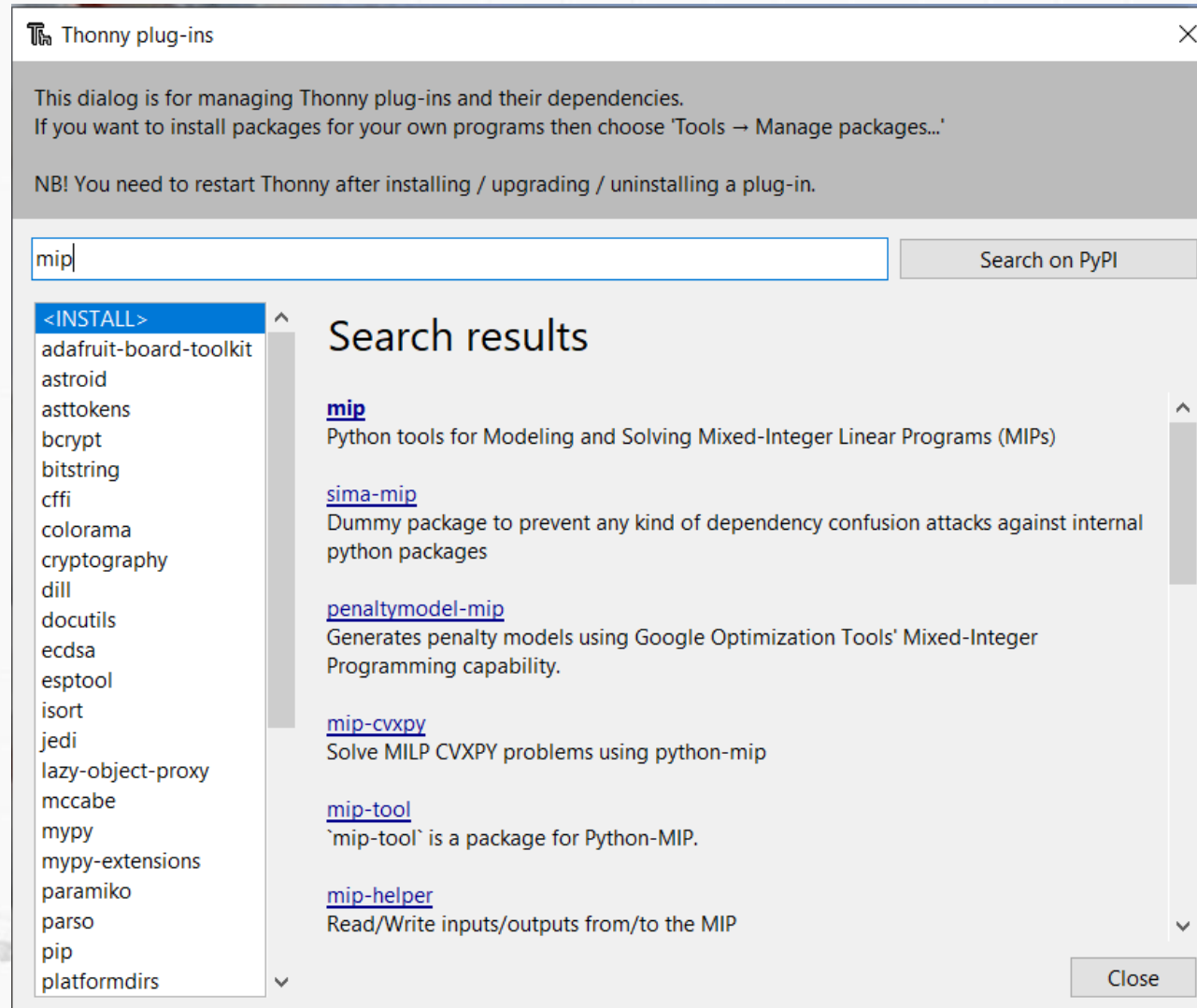
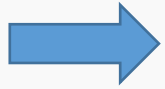
Python-MIP

Instalando a biblioteca Python-MIP



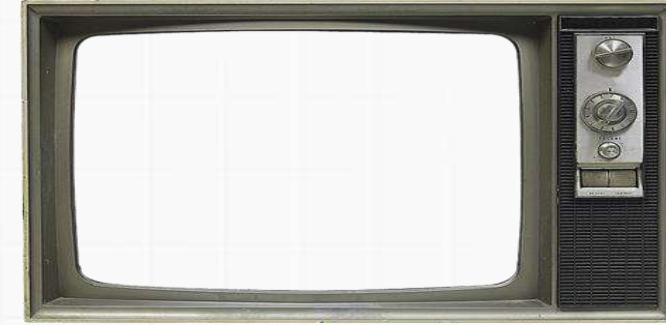
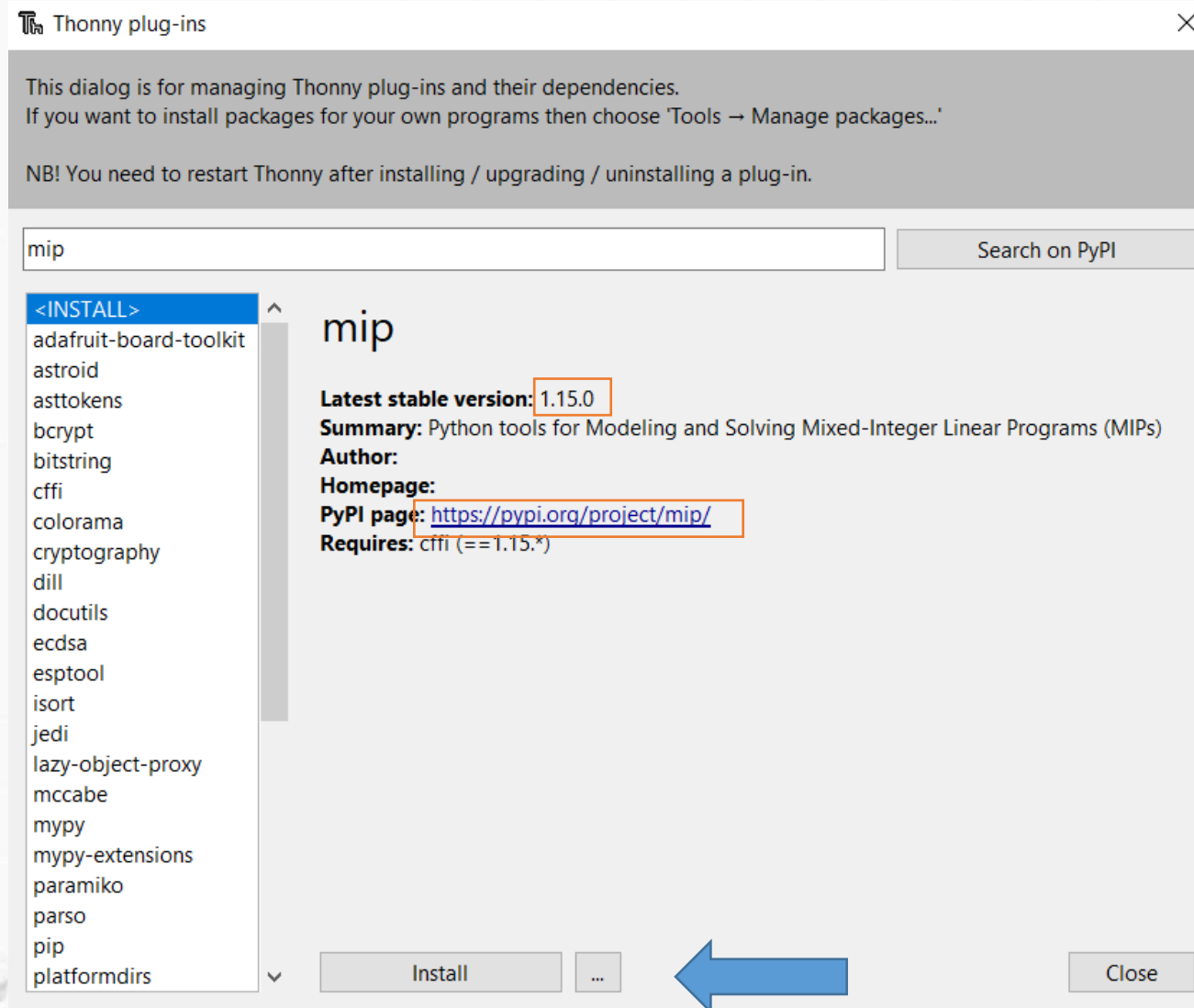
Python-MIP

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

1-python_exemplo_formulacao: 4 exemplos para vocês testarem



Python-MIP

Ex1 (PPL):

MAX $x_0 + 2x_1 + 3x_2$

s.a.

$-x_0 + x_1 + x_2 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$



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()

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

MAX $x_0 + 2x_1 + 3x_2$

s.a.

$-x_0 + x_1 + x_2 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$

- Importa biblioteca mip



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()

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

MAX $x_0 + 2x_1 + 3x_2$

s.a.

$-x_0 + x_1 + x_2 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$

- 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()

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

MAX $x_0 + 2x_1 + 3x_2$

s.a.

$-x_0 + x_1 + x_2 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$

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



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()

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

MAX $x_0 + 2x_1 + 3x_2$

s.a.

$-x_0 + x_1 + x_2 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$

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



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()

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

MAX $x_0 + 2x_1 + 3x_2$

s.a.

$-x_0 + x_1 + x_2 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$

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



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()

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

MAX $x_0 + 2x_1 + 3x_2$

s.a.

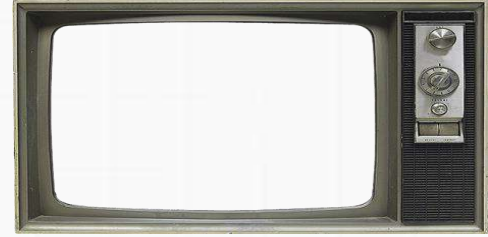
$-x_0 + x_1 + x_2 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$

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



Python-MIP

Ex1 (PPL):

MAX $x_0 + 2x_1 + 3x_2$

s.a.

$-x_0 + x_1 + x_2 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$



```
from mip import *
```

```
# cria model  
model = Model
```

```
# variáveis  
x0 = Model.add_var()  
x1 = Model.add_var()  
x2 = Model.add_var()
```

```
# restrições  
model.add_constraint(-x0 + x1 + x2 <= 20)  
model.add_constraint(x0 - 3x1 + x2 <= 30)  
model.add_constraint(x0 <= 40)
```

```
# função objetivo  
model.set_objective(x0 + 2x1 + 3x2)
```

```
# otimiza  
model.optimize()
```

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

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
```

ência



Python-MIP

MAX $x_0 + 2x_1 + 3x_2$
s.a.



Ex1 (PPL):

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>

vmware®

- Depois instalar linux

<https://ubuntu.com/download/desktop>

ubuntu®



Python-MIP

Ex2 (PPI):

```
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()

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

MAX $x_0 + 2x_1 + 3x_2 + x_3$

s.a.

$-x_0 + x_1 + x_2 + 10x_3 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_1 - 3.5x_3 = 0$

$x_0 \leq 40$

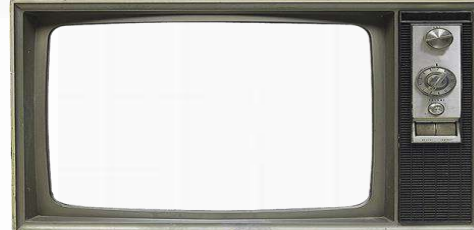
$x_0, x_1, x_2 \geq 0$

$2 \leq x_3 \leq 3$ e inteira



>>> %Run ex2.py

Python-MIP



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
```

Prepoc: vamos
estudar depois

```
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
```

Cortes: vamos
estudar depois

```
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:
vamos estudar
depois

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
```

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: $x_1 + 2 x_2 + 3 x_3 + x_4$
Subject To
c1: $-x_1 + x_2 + x_3 + 10 x_4 \leq 20$
c2: $x_1 - 3 x_2 + x_3 \leq 30$
c3: $x_2 - 3.5 x_4 = 0$
Bounds
 $0 \leq x_1 \leq 40$
 $2 \leq x_4 \leq 3$
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: $x_1 + 2 x_2 + 3 x_3 + x_4$
Subject To
c1: $- x_1 + x_2 + x_3 + 10 x_4 \leq 20$
c2: $x_1 - 3 x_2 + x_3 \leq 30$
c3: $x_2 - 3.5 x_4 = 0$
Bounds
 $0 \leq x_1 \leq 40$
 $2 \leq x_4 \leq 3$
General
 x_4
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: $x_1 + 2 x_2 + 3 x_3 + x_4$
Subject To
c1: $- x_1 + x_2 + x_3 + 10 x_4 \leq 20$
c2: $x_1 - 3 x_2 + x_3 \leq 30$
c3: $x_2 - 3.5 x_4 = 0$
Bounds
 $0 \leq x_1 \leq 40$
 $2 \leq x_4 \leq 3$
General
 x_4
End

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

Python-MIP

tsp (PPI):

Vamos ver como fazer para implementar **um modelo com 2 variáveis**. Para isso, considere o modelo ao lado que é um pedaço de um problema clássico chamado **tsp** (vamos estudar ele semana que vem). Por enquanto vamos apenas considerar um modelo sobre um grafo com 2 classes de restrições.

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

sujeito a:

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

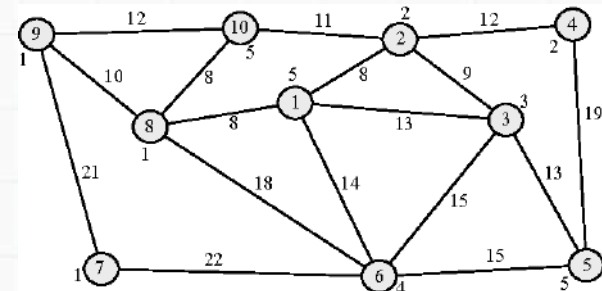
$$\sum_{i \in N \setminus \{j\}} x_{ji} = 1$$

x_{ij} inteiro



$G=(V,A)$

$N(i) \rightarrow$ vizinhos de i



Instancia do problema (matriz de arcos)

```
# instancia
```

```
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]]
```

Python-MIP

tsp (PPI):

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

sujeito a:

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

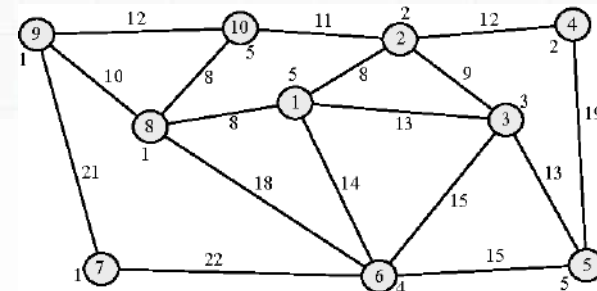
$$\sum_{j \in N} x_{ji} = 1$$

$$x_{ij} \text{ inteiro}$$



$G=(V,A)$

$N(i) \rightarrow$ vizinhos de i



Matriz de variáveis x

```
...  
# tempo  
início = time.process_time() # tempo de CPU  
  
# cria modelo  
model = Model(name="tsp", sense=MINIMIZE, solver_name=CBC)  
  
# variáveis 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

tsp (PPI):

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

sujeito a:

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

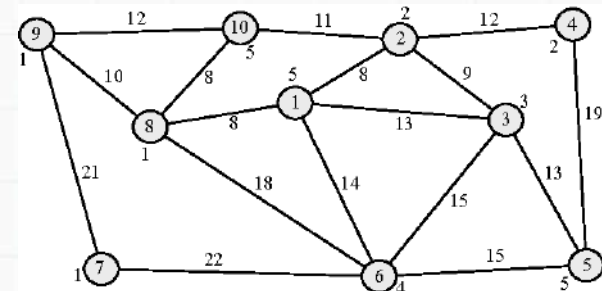
$$\sum_{j \in N \setminus (i)} x_{ji} = 1$$

$$x_{ij} \text{ inteiro}$$



$G=(V,A)$

$N(i) \rightarrow$ 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

tsp (PPI):

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

sujeito a:

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

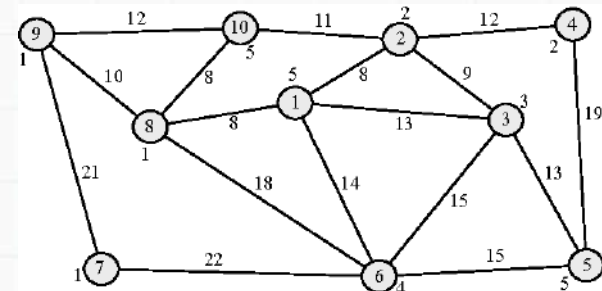
$$\sum_{i \in N} x_{ji} = 1$$

x_{ij} inteiro



$G=(V,A)$

$N(i)$ -> 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
```

```
# saída
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

tsp (PPI):

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

sujeito a:

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

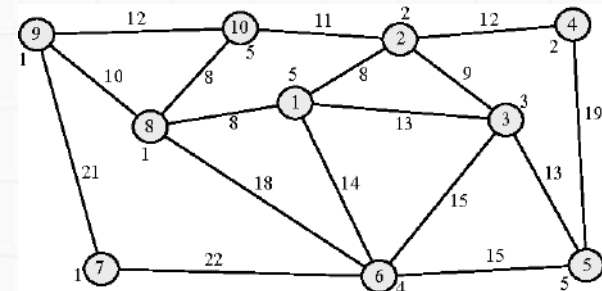
$$\sum_{i \in N} x_{ji} = 1$$

$$x_{ij} \text{ inteiro}$$



$G=(V,A)$

$N(i) \rightarrow$ vizinhos de i



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
```

```
#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
```

```
# saída
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

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

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

sujeito a:

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

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

$$x_{ij} \text{ inteiro}$$



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](#)



CPLEX

- Instalação: <https://www.ibm.com/academic/home>



IBM SkillsBuild Software Downloads

Harness the power of IBM. Get easy no-charge access to the tools you need to develop the next great thing. Enjoy powerful technical and strategic resources from IBM. Jump right in with access to powerful services and the most prominent open-source computer technologies, or take advantage of hands-on resources that will teach you about data science, artificial intelligence, security and more.

[Already registered? Log in](#) [Register now](#)



O IBM SkillsBuild permite instalar softwares e usar para fins acadêmicos



CPLEX

- Instalação: use email da UFF



Enter your academic institution issued email to begin

Only the students and faculty of participating academic institutions are eligible to access this website. Please enter your academic institution issued email below to register.

Your academic institution issued email

Find answers in our [frequently asked questions](#)

Submit

Depois preencha as informações pedidas e finalize seu registro, abaixo tem um link com instruções

<https://github.com/academic-initiative/documentation/blob/main/academic-initiative/how-to/How-to-register-with-the-IBM-Academic-Initiative/readme.md>

CPLEX

- Pronto, estamos logados

The banner features a group of diverse young adults sitting on a curved brick ledge outdoors. They are engaged in learning, with some looking at laptops, tablets, or open books. The background is a light, textured wall.

IBM | SkillsBuild | Topics ▾ | Usage terms ▾ | Additional Resources | FAQ

IBM SkillsBuild Software Downloads

Harness the power of IBM. Get easy no-charge access to the tools you need to develop the next great thing. Enjoy powerful technical and strategic resources from IBM. Jump right in with access to powerful services and the most prominent open-source computer technologies, or take advantage of hands-on resources that will teach you about data science, artificial intelligence, security and more.

CPLEX

- Instalação: desça na página



Most popular topics covered



AI



Capstone



Data Science



IBM Automation



IBM Cloud



IBM Quantum



Red Hat Academy



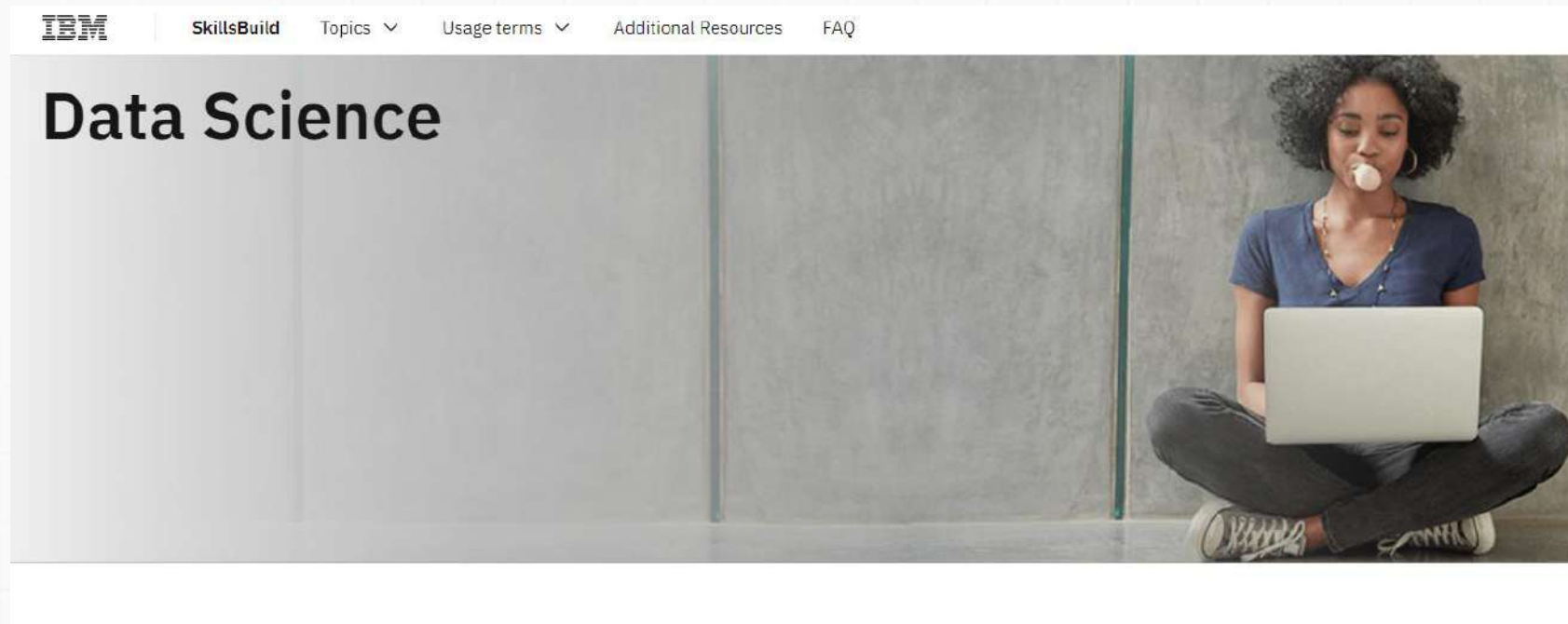
IBM Security



Learn more

CPLEX

- Role para baixo



CPLEX



Courseware

Software

Resources

ILOG CPLEX Optimization Studio

Analytical decision support toolkit for rapid development and deployment of optimization models using mathematical and constraint programming. It combines an integrated development [...]



SPSS Modeler Premium

SPSS Modeler provides an intuitive graphical interface to help visualize each step in the data mining process. Automatically transforms data into the best format for the most accurate predictive modeling.



SPSS Modeler Premium Concurrent License Request (MAP)

Need to running SPSS Modeler Premium on multiple machines in a computer classroom or lab? SPSS Modeler provides an intuitive graphical interface to help visualize each step in the data [...]



Cognos Analytics

Cognos Analytics on premise, helps enable you to: 1) Find answers, using AI and machine learning; 2) Unearth information that may not be obvious, using pattern detection; 3) Pose questions about your data [...]



View All



CPLEX



ILOG CPLEX Optimization Studio

Analytical decision support toolkit for rapid development and deployment of optimization models using mathematical and constraint programming. It combines an integrated development environment with the powerful Optimization Programming Language and high-performance CPLEX and CP Optimizer solvers.

[Product Information](#) →

Downloads

[Download](#) →

Knowledge base

Communities



CPLEX

Role para baixo



Overview

Search options

Your previous searches

Related Links

Software downloads

Overview

Get easy no-charge access to the tools you need to develop the next great thing. Enjoy powerful technical and strategic resources from IBM.

Jump right in with cloud access to powerful services and the most prominent open-source computer technologies, or take advantage of hands-on resources that will teach you about data and analytics, Internet of Things, and security.

Search options

Text

Brand

Part number

Find by part number

Part numbers

CPLEX



IBM Academic Initiative

Search for software

Popular downloads

Download history

Request assistance

Search options

Your previous searches

Related Links

license terms applicable to the eAssembly. Your acceptance of the license terms applicable to the eAssembly is a precondition to your downloading and using the eAssembly.

<input type="checkbox"/>	Image	Description	Date posted	Size (MB)
<input type="checkbox"/>	M04HML	IBM ILOG CPLEX Optimization Studio V22.1.0 for Linux on System i/p	18/03/2022	258
View details License agreement Download estimate eAssembly →				
<input type="checkbox"/>	M04HKML	IBM ILOG CPLEX Optimization Studio V22.1.0 for Linux x86-64	18/03/2022	625
View details License agreement Download estimate eAssembly →				
<input type="checkbox"/>	M04HMML	IBM ILOG CPLEX Optimization Studio V22.1.0 for OSX	18/03/2022	698
View details License agreement Download estimate eAssembly →				
<input type="checkbox"/>	M04HJML	IBM ILOG CPLEX Optimization Studio V22.1.0 for Windows x86-64	18/03/2022	712
View details License agreement Download estimate eAssembly →				
<input type="checkbox"/>	M04HHML	IBM ILOG CPLEX Optimization Studio V22.1.0 Quick Start Guide	18/03/2022	2
View details License agreement Download estimate eAssembly →				

CPLEX

- Instalação: selecione "I agree" e "Download now"



Search options

Your previous searches

Related Links

View details	License agreement	Download estimate	eAssembly
<input type="checkbox"/>	M04HMML	IBM ILOG CPLEX Optimization Studio V22.1.0 for OSX	18/03/2022 698
View details	License agreement	Download estimate	eAssembly
<input type="checkbox"/>	M04HJML	IBM ILOG CPLEX Optimization Studio V22.1.0 for Windows x86-64	18/03/2022 712
View details	License agreement	Download estimate	eAssembly
<input checked="" type="checkbox"/>	M04HHML	IBM ILOG CPLEX Optimization Studio V22.1.0 Quick Start Guide	18/03/2022 2
View details	License agreement	Download estimate	eAssembly

☒ I agree ☐ I do not agree

By clicking the "I agree" button, you agree that (1) you have had the opportunity to read and understand the above license agreement(s) and multi-product package terms, if any, and (2) terms of the license agreement(s) govern this transaction. If you do not agree with the terms of the agreement(s), you will be unable to download the software.

[Download now](#)

If Download Director does not start [install / re-install Download Director.](#)

Se o download não começar, siga os passos para instalar o downloader da IBM

CPLEX

- Caso tenha Windows, você pode criar uma máquina virtual

<https://www.virtualbox.org/>



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

vmware®

- Depois instalar linux

<https://ubuntu.com/download/desktop>

ubuntu®



CPLEX

Linux



- Torna o arquivo de instalação executável

```
> chmod 777 ./cplex_studio2210.linux_x86_64.bin
```

- Instala como super usuário

```
> sudo ./cplex_studio2210.linux_x86_64.bin
```

Siga as instruções



CPLEX

- CPLEX é um produto desenvolvido pela ILOG (IBM) para resolver:

- LPs
- MIPs
- QPs
- MIQPs
- Network Flow problems
- Constraint programming, e mais...

The CPLEX logo, featuring the word "CPLEX" in a bold, red, sans-serif font, underlined with a thick black line.

Vamos estudar o CPLEX mais a fundo do que o Python-MPI pois ele será nossa principal ferramenta.



CPLEX



- CPLEX é um produto desenvolvido pela ILOG (IBM) para resolver:

- LPs
- MIPs
- QPs
- MIQPs
- Network Flow problems
- Constraint programming, e mais...

The CPLEX logo is displayed in a bold, red, sans-serif font. The word "CPLEX" is underlined with a thick black horizontal line.

- CPLEX é compatível com plataformas:

- Windows
- UNIX (Linux, MAC-OS, Solares, etc)



CPLEX

- CPLEX é um produto desenvolvido pela ILOG (IBM) para resolver:

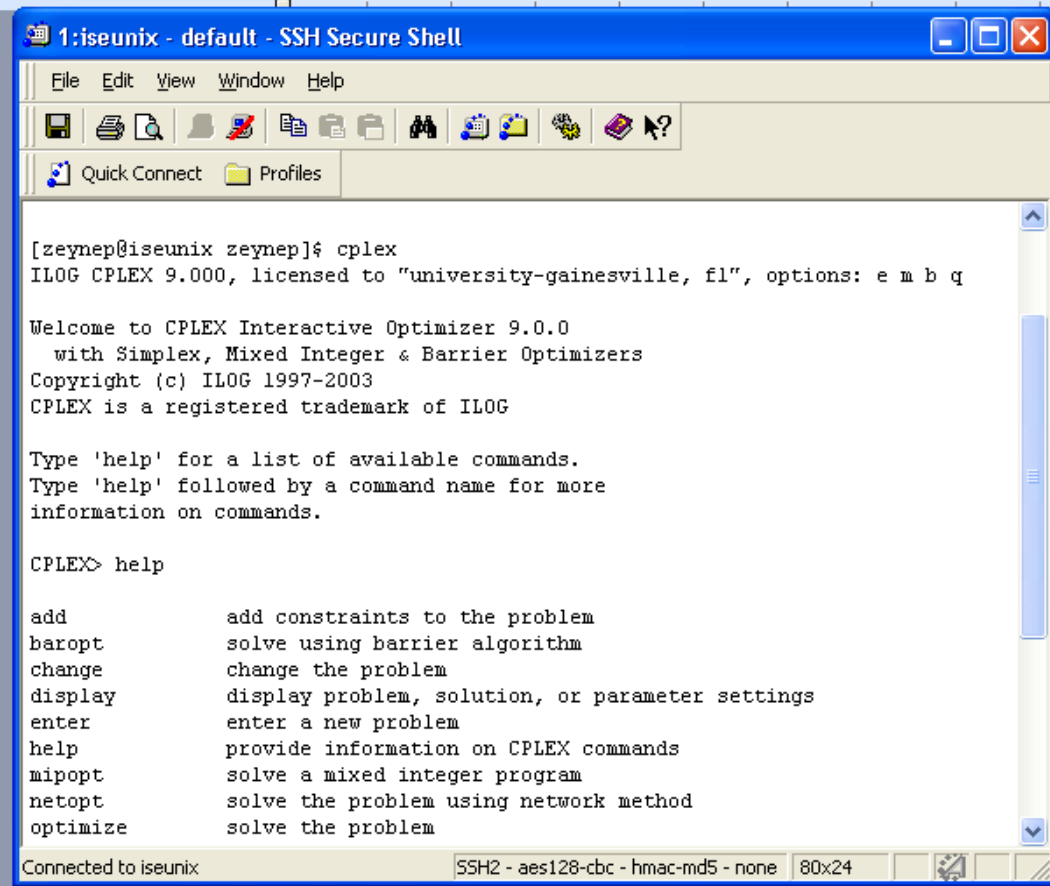
- LPs
- MIPs
- QPs
- MIQPs
- Network Flow problems
- Constraint programming, e mais...

- CPLEX é compatível com plataformas:

- Windows
- UNIX (Linux, MAC-OS, Solares, etc)

- Podemos utilizar CPLEX por:

- CPLEX interactive optimizer (linha de comando)



```
[zeynep@iseunix zeynep]$ cplex
ILOG CPLEX 9.000, licensed to "university-gainesville, fl", options: e m b q

Welcome to CPLEX Interactive Optimizer 9.0.0
  with Simplex, Mixed Integer & Barrier Optimizers
Copyright (c) ILOG 1997-2003
CPLEX is a registered trademark of ILOG

Type 'help' for a list of available commands.
Type 'help' followed by a command name for more
information on commands.

CPLEX> help

add          add constraints to the problem
baropt       solve using barrier algorithm
change       change the problem
display      display problem, solution, or parameter settings
enter        enter a new problem
help         provide information on CPLEX commands
mipopt       solve a mixed integer program
netopt       solve the problem using network method
optimize     solve the problem

Connected to iseunix          SSH2 - aes128-cbc - hmac-md5 - none 80x24
```

CPLEX

- CPLEX interactive optimizer (Exemplo)

CPLEX> enter

Enter name for problem: example

Enter new problem ['end' on a separate line terminates]:

max $4x+6y$

subject to

$x+y<5$

End

CPLEX> optimize

Tried aggregator 1 time.

LP Presolve eliminated 1 rows and 2 columns.

All rows and columns eliminated.

Presolve time = 0.00 sec.

Dual simplex - Optimal: Objective = 3.0000000000e+01

Solution time = 0.00 sec. Iterations = 0 (0)



CPLEX

- CPLEX interactive optimizer (Exemplo)

CPLEX> enter

Enter name for problem: example

Enter new problem ['end' on a separate line terminates]:

max $4x+6y$

subject to

$x+y<5$

End

CPLEX> optimize

Tried aggregator 1 time.

LP Presolve eliminated 1 rows and 2 columns.

All rows and columns eliminated.

Presolve time = 0.00 sec.

Dual simplex - Optimal: Objective = 3.0000000000e+01

Solution time = 0.00 sec. Iterations = 0 (0)



PROS:

- Fácil de usar (carrega modelo e aplica método)

CONS:

- Difícil de manipular e interagir com o método (cortes, colunas e etc)

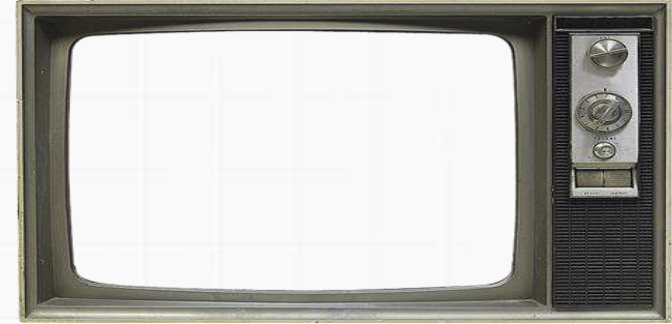
CPLEX



- CPLEX é um produto desenvolvido pela ILOG (IBM) para resolver:
 - LPs
 - MIPs
 - QPs
 - MIQPs
 - Network Flow problems
 - Constraint programming, e mais...
- CPLEX é compatível com plataformas:
 - Windows
 - UNIX (Linux, MAC-OS, Solares, etc)
- Podemos utilizar CPLEX por:
 - CPLEX interactive optimizer (linha de comando)
 - CPLEX Concert Technology (bibliotecas)
 - C++, C#, Java e Python



CPLEX



- CPLEX é um produto desenvolvido pela ILOG (IBM) para resolver:

- LPs
- MIPs
- QPs
- MIQPs
- Network Flow problems
- Constraint programming, e mais...

- CPLEX é compatível com plataformas:

- Windows
- UNIX (Linux, MAC-OS, Solares, etc)

- Podemos utilizar CPLEX por:

- CPLEX interactive optimizer (linha de comando)
- CPLEX Concert Technology (bibliotecas)
 - C++, C#, Java e Python
 - Julia + Jump



CPLEX



- CPLEX é um produto desenvolvido pela ILOG (IBM) para resolver:
 - LPs
 - MIPs
 - QPs
 - MIQPs
 - Network Flow problems
 - Constraint programming, e mais...
- CPLEX é compatível com plataformas:
 - Windows
 - UNIX (Linux, MAC-OS, Solares, etc)
- Podemos utilizar CPLEX por:
 - CPLEX interactive optimizer (linha de comando)
 - CPLEX Concert Technology (bibliotecas)
 - C++, C#, Java e Python
 - Julia + Jump
 - CPLEX Callable Library (bibliotecas)
 - C



CPLEX

- CPLEX é um produto desenvolvido pela ILOG (IBM) para resolver:

- LPs
- MIPs
- QPs
- MIQPs
- Network Flow problems
- Constraint programming, e mais...

- CPLEX é compatível com plataformas:

- Windows
- UNIX (Linux, MAC-OS, Solares, etc)

- Podemos utilizar CPLEX por:

- CPLEX interactive optimizer (linha de comando)
- CPLEX Concert Technology (bibliotecas)
 - C++, C#, Java e Python
 - Julia + Jump
- CPLEX Callable Library (bibliotecas)
 - C
- OPL (linguagem do CPLEX para modelagem, bem próximo a formulação)

```
dvar interval tasks[t in Tasks] size durations[t];
dvar interval opttasks[optTasks] optional;
dvar interval worker[Workers];
```

```
cumulFunction group[g in Groups] =
    sum (w in workers[g]) pulse(worker[w], 1)
    - sum (<t,g> in optTasks) pulse(opttasks[<t,g>], 1);
```

```
execute {
    cp.param.FailLimit = 5000;
}
```

```
minimize max(w in Workers) lengthOf(worker[w]);
```

```
subject to {
    forall(t in Tasks) /* starts of Tasks */
        startOf(tasks[t]) == start[t];
```

```
    forall(t in Tasks)
        alternative(tasks[t], all(<t,g> in optTasks) opttasks[<t,g>]);
```

```
    forall(g in Groups) {
        0 <= group[g];
        group[g] <= maxUnusedWorkers[g];
    }
};
```

```
execute {
    for (var w in Workers)
        writeln(w + " present from " + worker[w].start + " to " + worker[w].end);
}
```

CPLEX

- Dificuldade de uso

- **Model implementation and maintenance**

Hard  **Easy**
Callable Library Concert OPL

- **Problem modifications and hot starts**

Hard  **Easy**
OPL Concert Callable Library

- **Branch-and-bound customization**

Hard  **Easy**
Callable Library Concert

CPLEX

- CPLEX é um produto desenvolvido pela ILOG (IBM) para resolver:

- LPs
- MIPs
- QPs
- MIQPs
- Network Flow problems
- Constraint programming, e mais...

- CPLEX é compatível com plataformas:

- Windows
- UNIX (**Linux**, MAC-OS, Solares, etc)

- Podemos utilizar CPLEX por:

- CPLEX interactive optimizer (linha de comando)
- CPLEX Concert Technology (bibliotecas)
 - C++, C#, Java e Python
 - **Julia + Jump**
- CPLEX Callable Library (bibliotecas)
 - **C**
- OPL (linguagem do CPLEX para modelagem, bem próximo a formulação)

Qual melhor combinação para otimização ?

- lembre-se que podemos ter que implementar métodos (heurísticas e separações) na linguagem de escolha, e nesse ponto o Python-MIP fica para tras



CPLEX

1-cplex_exemplo_formulacao: 4 exemplos para vocês testarem



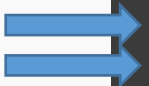
CPLEX



1-cplex_exemplo_formulacao: 4 exemplos para vocês testarem

- Makefile → compila os exemplos

Alterar ?



```
1 #####
2
3 SYSTEM      = x86-64_linux
4 LIBFORMAT   = static_pic
5 CPLEXDIR    = /opt/ibm/ILOG/CPLEX_Studio201/cplex
6 CONCERTDIR  = /opt/ibm/ILOG/CPLEX_Studio201/concert
7
8 #####
9
10 # Compilador
11 CCC = g++ -O0
12
13 # Opcoes de compilacao
14 CCOPT = -m64 -O -fPIC -fno-strict-aliasing -fexceptions -DNDEBUG -DIL_STD -Wno-ignored-attributes
15
16 # Bibliotecas e includes
17 CPLEXBINDIR = $(CPLEXDIR)/bin/$(BINDIST)
18 CPLEXLIBDIR = $(CPLEXDIR)/lib/$(SYSTEM)/$(LIBFORMAT)
19 CONCERTLIBDIR = $(CONCERTDIR)/lib/$(SYSTEM)/$(LIBFORMAT)
20
21 CPLEXBINDIR = $(CPLEXDIR)/bin/$(SYSTEM)
22 CPLEXLIB     = cplex$(dynamic:yes=2010)
23
24 CCLNDIRS = -L$(CPLEXLIBDIR) -L$(CONCERTLIBDIR) $(dynamic:yes=-L$(CPLEXBINDIR))
25 CCLNFLAGS = -lconcert -lilocplex -l$(CPLEXLIB) -lm -lpthread -ldl
26
27 CONCERTINCDIR = $(CONCERTDIR)/include
28 CPLEXINCDIR   = $(CPLEXDIR)/include
```

CPLEX



1-cplex_exemplo_formulacao: 4 exemplos para vocês testarem

- Makefile → compila os exemplos

make



```
32 all : ex1 ex2 ex3 tsp
33
34 ex1: ex1.o
35     $(CCC) $(CCFLAGS) $(CCLNDIRS) -o ex1 ex1.o $(CCLNFLAGS)
36 ex1.o: ex1.cpp
37     $(CCC) -c $(CCFLAGS) ex1.cpp -o ex1.o
38
39 ex2: ex2.o
40     $(CCC) $(CCFLAGS) $(CCLNDIRS) -o ex2 ex2.o $(CCLNFLAGS)
41 ex2.o: ex2.cpp
42     $(CCC) -c $(CCFLAGS) ex2.cpp -o ex2.o
43
44 ex3: ex3.o
45     $(CCC) $(CCFLAGS) $(CCLNDIRS) -o ex3 ex3.o $(CCLNFLAGS)
46 ex3.o: ex3.cpp
47     $(CCC) -c $(CCFLAGS) ex3.cpp -o ex3.o
48
49 tsp: tsp.o data.o
50     $(CCC) $(CCFLAGS) $(CCLNDIRS) tsp.o data.o -o tsp $(CCLNFLAGS)
51 tsp.o: tsp.cpp
52     $(CCC) -c $(CCFLAGS) tsp.cpp -o tsp.o
53 data.o: data.cpp
54     $(CCC) -c $(CCFLAGS) data.cpp -o data.o
55
56 clean:
57     rm --force ex1 ex1.o ex2 ex2.o ex3 ex3.o tsp tsp.o data.o
```

make clean



CPLEX

Ex1 (PPL):

MAX $x_0 + 2x_1 + 3x_2$

s.a.

$-x_0 + x_1 + x_2 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$



CPLEX

Ex1 (PPL):

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ambiente (Env)

Modelo

- Variáveis
- Restrições
- FO, etc..

CPLEX

CPLEX

Ex1 (PPL):

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



CPLEX

Ambiente (Env)

Modelo

- Variáveis
- Restrições
- FO, etc..



Handles de Saída

- Variáveis
- Restrições

CPLEX

Ex1 (PPL):

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



CPLEX

Ambiente (Env)

Modelo

- Variáveis
- Restrições
- FO, etc..

Solver:

- "Solver"
- Heurísticas Primais
- Pré-processamento

Handles de Saída

- Variáveis
- Restrições

CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

Roteiro:

Criar ambiente

Construir modelo

Carregar modelo num algoritmo

Resolver problema

Ver resultados

Terminar o programa

Ambiente (Env)

Modelo

- Variáveis
- Restrições
- FO, etc..

Solver:

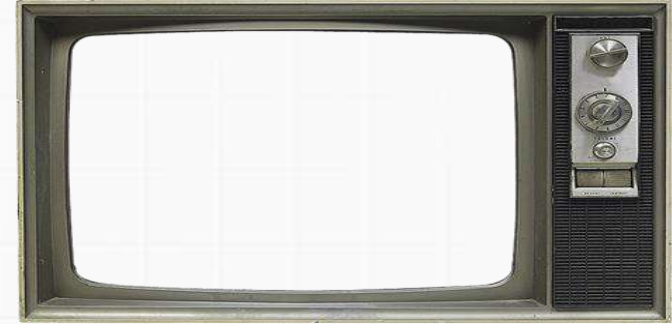
- "Solver"
- Heurísticas Primais
- Pré-processamento

Handles de Saída

- Variáveis
- Restrições

CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel     model(env);

    IloNumVarArray  x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.

Roteiro:

- Criar ambiente
- Construir modelo
- Carregar modelo num algoritmo
- Resolver problema
- Ver resultados
- Terminar o programa



CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$

Ex1 (PPL):

Vamos criar o ambiente

Ambiente (Env)

Modelo

- Variáveis
- Restrições
- FO, etc..

Solver:

- "Solver"
- Heurísticas Primais
- Pré-processamento

Handles de Saída

- Variáveis
- Restrições

CPLEX

Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel     model(env);

    IloNumVarArray x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

MAX $x_0 + 2x_1 + 3x_2$

s.a.

$-x_0 + x_1 + x_2 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$

- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.

Podemos ter vários modelos em um único ambiente

Roteiro:

Criar ambiente

Construir modelo

Carregar modelo num algoritmo

Resolver problema

Ver resultados

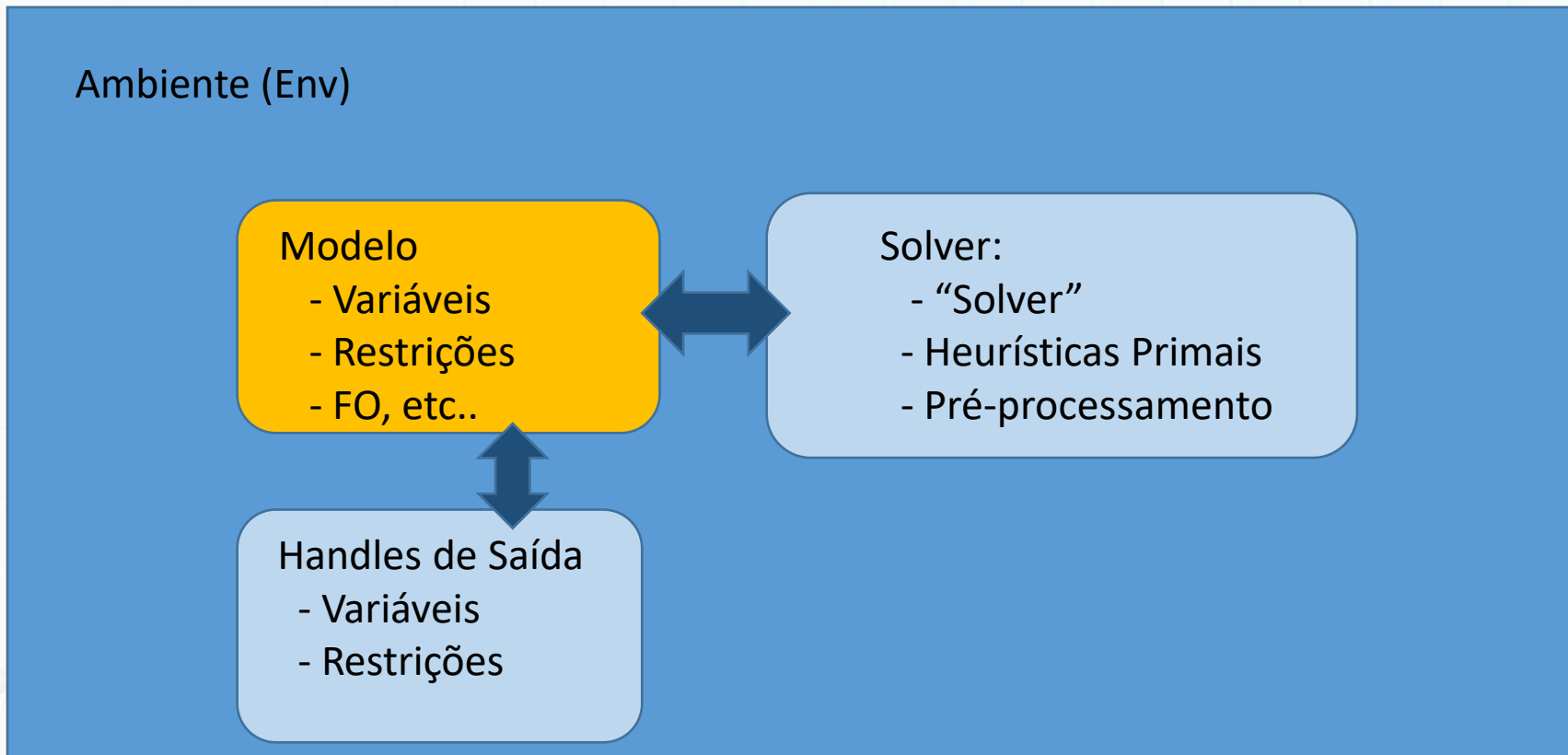
Terminar o programa

CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$

Ex1 (PPL):

Vamos descrever o modelo



CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel     model(env);

    IloNumVarArray x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.

Roteiro:

- Criar ambiente
- Construir modelo
- Carregar modelo num algoritmo
- Resolver problema
- Ver resultados
- Terminar o programa

CPLEX

MAX $x_0 + 2x_1 + 3x_2$

s.a.

$-x_0 + x_1 + x_2 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel    model(env);

    IloNumVarArray x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.

- Ponteiro para vetor de variáveis numéricas (vazio)

Roteiro:

Criar ambiente

Construir modelo

Carregar modelo num algoritmo

Resolver problema

Ver resultados

Terminar o programa



CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel    model(env);

    IloNumVarArray x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.
- Ponteiro para vetor de variáveis numéricas (vazio)
- adiciona variável numérica ao vetor com limite inferior 0 e superior 40

Roteiro:

- Criar ambiente
- Construir modelo
- Carregar modelo num algoritmo
- Resolver problema
- Ver resultados
- Terminar o programa

CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel    model(env);

    IloNumVarArray x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.
- Ponteiro para vetor de variáveis numéricas (vazio)
- adiciona variável numérica ao vetor com limite inferior 0 e superior 40
- adiciona variável numérica ao vetor com limite inferior 0 e superior + infinito (default)

Roteiro:

- Criar ambiente
- Construir modelo
- Carregar modelo num algoritmo
- Resolver problema
- Ver resultados
- Terminar o programa

CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel     model(env);

    IloNumVarArray x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.
- Ponteiro para vetor de variáveis numéricas (vazio)
- adiciona variável numérica ao vetor com limite inferior 0 e superior 40
- adiciona variável numérica ao vetor com limite inferior 0 e superior + infinito (default)
- restrições

Roteiro:

Criar ambiente
Construir modelo
Carregar modelo num algoritmo
Resolver problema
Ver resultados
Terminar o programa

CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel     model(env);

    IloNumVarArray  x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

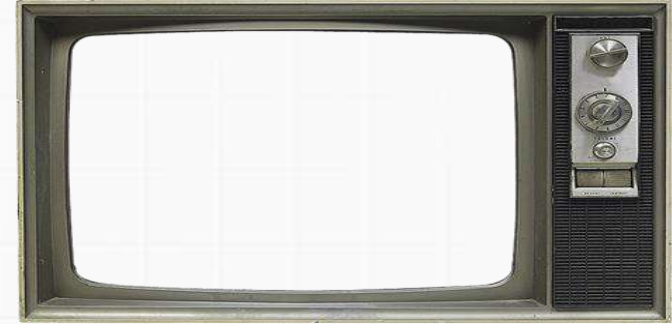
- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.
- Ponteiro para vetor de variáveis numéricas (vazio)
- adiciona variável numérica ao vetor com limite inferior 0 e superior 40
- adiciona variável numérica ao vetor com limite inferior 0 e superior + infinito (default)
- restrições, função objetivo

Roteiro:

- Criar ambiente
- Construir modelo
- Carregar modelo num algoritmo
- Resolver problema
- Ver resultados
- Terminar o programa

CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel    model(env);

    IloNumVarArray x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.
- Ponteiro para vetor de variáveis numéricas (vazio)
- adiciona variável numérica ao vetor com limite inferior 0 e superior 40
- adiciona variável numérica ao vetor com limite inferior 0 e superior + infinito (default)
- restrições, função objetivo

Roteiro:

- Criar ambiente
- Construir modelo
- Carregar modelo num algoritmo
- Resolver problema
- Ver resultados
- Terminar o programa

CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel     model(env);

    IloNumVarArray  x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.
- Ponteiro para vetor de variáveis numéricas (vazio)
- adiciona variável numérica ao vetor com limite inferior 0 e superior 40
- adiciona variável numérica ao vetor com limite inferior 0 e superior + infinito (default)
- restrições, função objetivo, carrega modelo no algoritmo

Roteiro:

- Criar ambiente
- Construir modelo
- Carregar modelo num algoritmo
- Resolver problema
- Ver resultados
- Terminar o programa

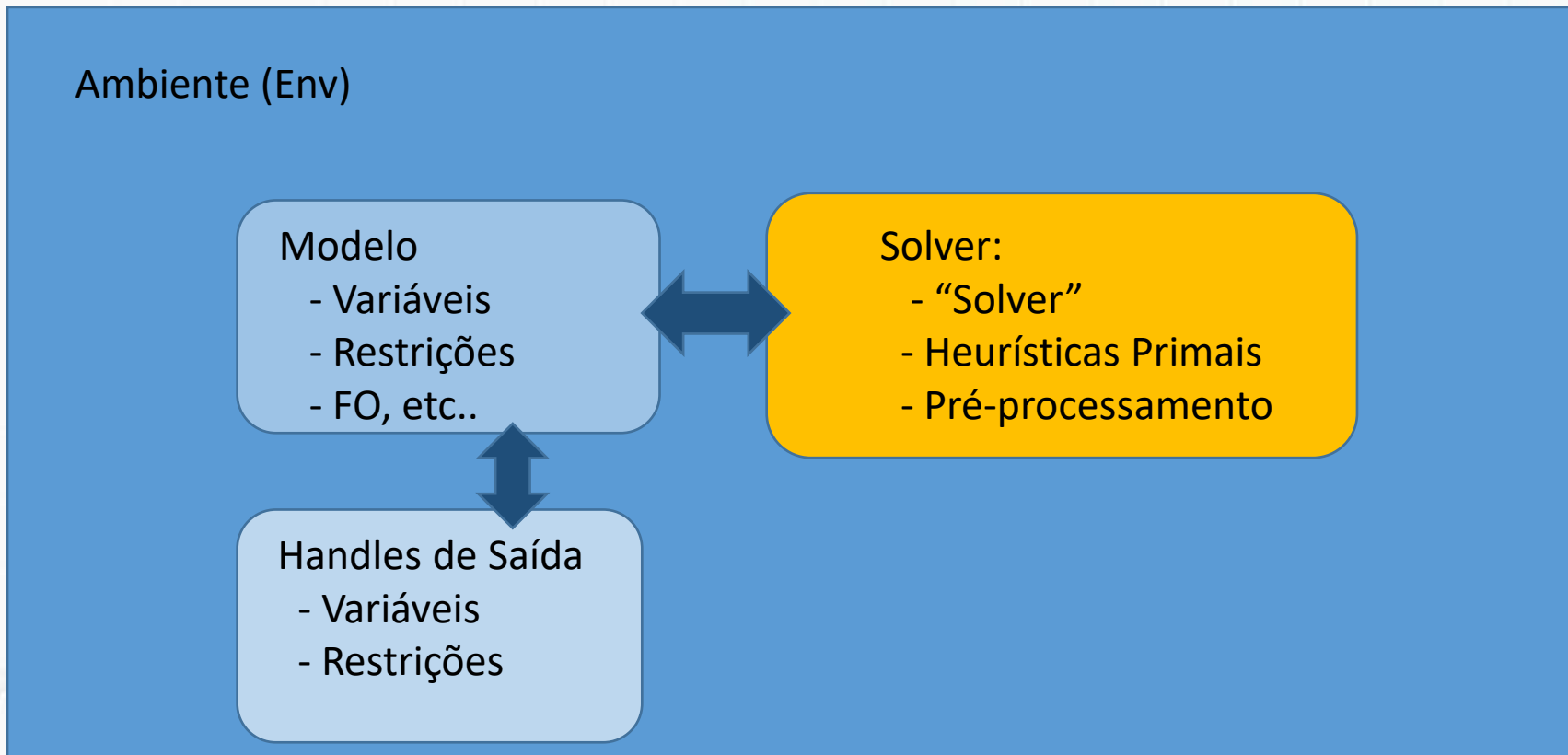
CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

Vamos aplicar método de solução



CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel    model(env);

    IloNumVarArray x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.
- Ponteiro para vetor de variáveis numéricas (vazio)
- adiciona variável numérica ao vetor com limite inferior 0 e superior 40
- adiciona variável numérica ao vetor com limite inferior 0 e superior + infinito (default)
- restrições, função objetivo, carrega modelo no algoritmo
- resolve

Roteiro:

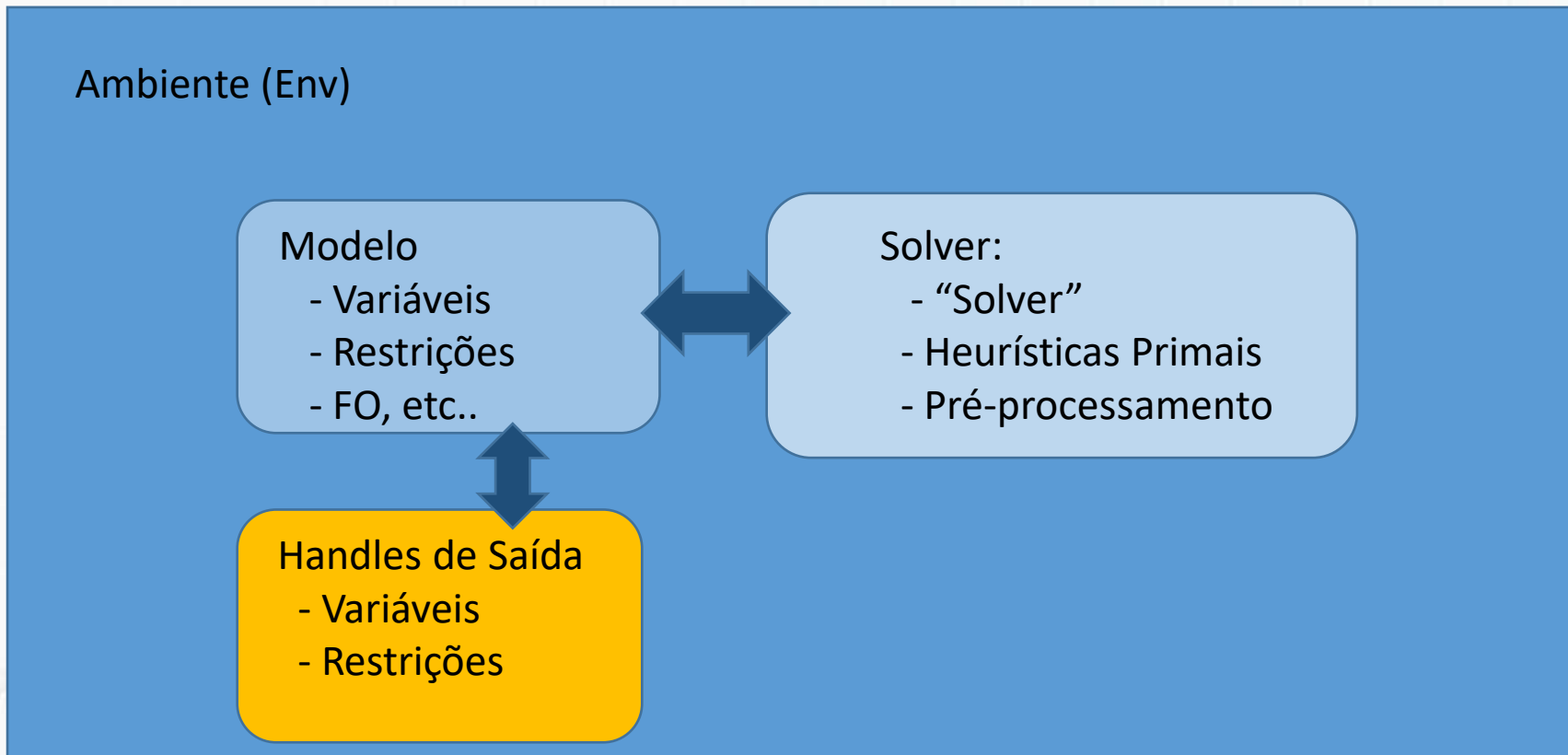
- Criar ambiente
- Construir modelo
- Carregar modelo num algoritmo
- Resolver problema
- Ver resultados
- Terminar o programa

CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$

Ex1 (PPL):

Vamos extrair solução



CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel     model(env);

    IloNumVarArray  x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

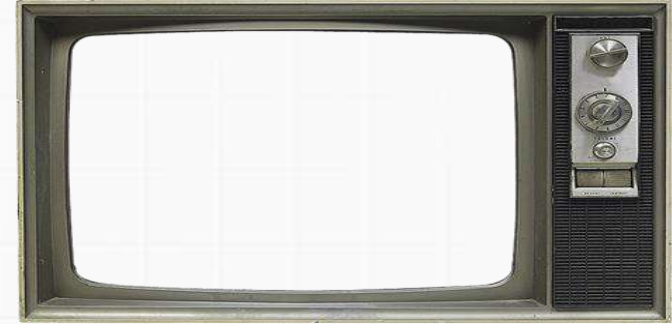
- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.
- Ponteiro para vetor de variáveis numéricas (vazio)
- adiciona variável numérica ao vetor com limite inferior 0 e superior 40
- adiciona variável numérica ao vetor com limite inferior 0 e superior + infinito (default)
- restrições, função objetivo, carrega modelo no algoritmo
- resolve, valor ótimo

Roteiro:

- Criar ambiente
- Construir modelo
- Carregar modelo num algoritmo
- Resolver problema
- Ver resultados
- Terminar o programa

CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel     model(env);

    IloNumVarArray  x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

- ILOSTLBEGIN é uma macro de portabilidade entre os diversos sistemas operacionais.
- Ponteiro para vetor de variáveis numéricas (vazio)
- adiciona variável numérica ao vetor com limite inferior 0 e superior 40
- adiciona variável numérica ao vetor com limite inferior 0 e superior + infinito (default)
- restrições, função objetivo, carrega modelo no algoritmo
- resolve, valor ótimo, finaliza e libera estruturas

Roteiro:

- Criar ambiente
- Construir modelo
- Carregar modelo num algoritmo
- Resolver problema
- Ver resultados
- Terminar o programa



CPLEX

MAX $x_0 + 2x_1 + 3x_2$
s.a.
 $-x_0 + x_1 + x_2 \leq 20$
 $x_0 - 3x_1 + x_2 \leq 30$
 $x_0 \leq 40$
 $x_0, x_1, x_2 \geq 0$



Ex1 (PPL):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {
    IloEnv      env;
    IloModel     model(env);

    IloNumVarArray x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    model.add( - x[0] +      x[1] + x[2] <= 20);
    model.add(  x[0] - 3 * x[1] + x[2] <= 30);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();
}
```

Saída:

```
CPXPARAM_MIP_Strategy_CallbackReducedLP      0
Tried aggregator 1 time.
No LP presolve or aggregator reductions.
Presolve time = 0.00 sec. (0.00 ticks)

Iteration log . . .
Iteration:      1    Dual infeasibility =      0.000000
Iteration:      2    Dual objective      =     202.500000
Max=202.5
```

Por padrão, o CPLEX usa o Dual Simplex para solucionar PPLs.

Outras opções são:

- Primal Simplex
- Network flow Simplex
- Método da Barreira (Pontos Interiores)

CPLEX

Ex2 (PPI):

MAX $x_0 + 2x_1 + 3x_2 + x_3$

s.a.

$-x_0 + x_1 + x_2 + 10x_3 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_1 - 3.5x_3 = 0$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$

$2 \leq x_3 \leq 3$ e inteira



CPLEX

Ex2 (PPI):

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN
int main () {

    IloEnv      env;
    IloModel    model(env);

    IloNumVarArray x(env);

    x.add(IloNumVar(env, 0, 40));
    x.add(IloNumVar(env));
    x.add(IloNumVar(env));

    x.add(IloNumVar(env, 2, 3, ILOINT));

    model.add( - x[0] + x[1] + x[2] + 10 * x[3] <= 20);
    model.add( x[0] - 3 * x[1] + x[2] <= 30);
    model.add( x[1] - 3.5 * x[3] == 0);
    model.add(IloMaximize(env, x[0]+2*x[1]+3*x[2]+x[3]));

    IloCplex cplex(model);
    cplex.solve();

    cout << "Max=" << cplex.getObjValue() << endl;
    env.end();

}
```

MAX $x_0 + 2x_1 + 3x_2 + x_3$

s.a.

$-x_0 + x_1 + x_2 + 10x_3 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_1 - 3.5x_3 = 0$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$

$2 \leq x_3 \leq 3$ e inteira



CPLEX

Ex2 (PPI): Saída

MAX $x_0 + 2x_1 + 3x_2 + x_3$

s.a.

$-x_0 + x_1 + x_2 + 10x_3 \leq 20$

$x_0 - 3x_1 + x_2 \leq 30$

$x_1 - 3.5x_3 = 0$

$x_0 \leq 40$

$x_0, x_1, x_2 \geq 0$

$2 \leq x_3 \leq 3$ e inteira



CPXPARAM MIP_Strategy_CallbackReducedLP 0
Found incumbent of value 46.000000 after 0.00 sec. (0.00 ticks)

...

MIP emphasis: balance optimality and feasibility.

MIP search method: dynamic search.

Parallel mode: deterministic, using up to 8 threads.

Root relaxation solution time = 0.00 sec. (0.00 ticks)

Node	Nodes		Objective	IInf	Best Integer	Cuts/		ItCnt	Gap
	Left	Right				Best	Bound		
*	0+	0			46.0000	163.0000		254.35%	
*	0+	0			122.5000	163.0000		33.06%	
	0	0	125.2083	1	122.5000	125.2083		3	2.21%
	0	0	cutoff		122.5000			3	---

Elapsed time = 0.00 sec. (0.03 ticks, tree = 0.01 MB, solutions = 2)

Root node processing (before b&c):

Real time = 0.00 sec. (0.03 ticks)

Parallel b&c, 8 threads:

Real time = 0.00 sec. (0.00 ticks)

Sync time (average) = 0.00 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 0.00 sec. (0.03 ticks)

Max=122.5

Ex1 → Simplex

Ex2 → B&B



CPLEX

Ex3 (PPI): Arquivo .lp



ex3.lp - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

Maximize

obj: $x_1 + 2x_2 + 3x_3 + x_4$

Subject To

c1: $-x_1 + x_2 + x_3 + 10x_4 \leq 20$

c2: $x_1 - 3x_2 + x_3 \leq 30$

c3: $x_2 - 3.5x_4 = 0$

Bounds

$0 \leq x_1 \leq 40$

$2 \leq x_4 \leq 3$

General

x_4

End

Ex3 (PPI): Arquivo .lp

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN

int main (int argc, char **argv)
{
    IloEnv env;
    try {
        IloModel model(env);
        IloCplex cplex(env);

        IloObjective obj;
        IloNumVarArray var(env);
        IloRangeArray rng(env);

        cplex.importModel(model, argv[1], obj, var, rng);

        cplex.extract(model);
        cplex.solve();

        env.out() << "Solution status = " << cplex.getStatus() << endl;
        env.out() << "Solution value = " << cplex.getObjValue() << endl;

        IloNumArray vals(env);
        cplex.getValues(vals, var);
        env.out() << "Values = " << vals << endl;
    }
    catch (IloException& e) {
        cerr << "Concert exception caught: " << e << endl;
    }
    catch (...) {
        cerr << "Unknown exception caught" << endl;
    }
    env.end();
    return 0;
}
```

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
```

- Importa modelo do arquivo preenchendo estruturas

Ex3 (PPI): Arquivo .lp

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN

int main (int argc, char **argv)
{
    IloEnv env;
    try {
        IloModel model(env);
        IloCplex cplex(env);

        IloObjective obj;
        IloNumVarArray var(env);
        IloRangeArray rng(env);

        cplex.importModel(model, argv[1], obj, var, rng);

        cplex.extract(model);
        cplex.solve();

        env.out() << "Solution status = " << cplex.getStatus() << endl;
        env.out() << "Solution value = " << cplex.getObjValue() << endl;

        IloNumArray vals(env);
        cplex.getValues(vals, var);
        env.out() << "Values = " << vals << endl;
    }
    catch (IloException& e) {
        cerr << "Concert exception caught: " << e << endl;
    }
    catch (...) {
        cerr << "Unknown exception caught" << endl;
    }
    env.end();
    return 0;
}
```

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:

Optimal
Unknow
Infeasible
Unbounded
Error

Ex3 (PPI): Arquivo .lp

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
```

“Handle” de Variáveis para saída da solução

```
#include <ilcplex/ilocplex.h>
ILOSTLBEGIN

int main (int argc, char **argv)
{
    IloEnv env;
    try {
        IloModel model(env);
        IloCplex cplex(env);

        IloObjective obj;
        IloNumVarArray var(env);
        IloRangeArray rng(env);

        cplex.importModel(model, argv[1], obj, var, rng);

        cplex.extract(model);
        cplex.solve();

        env.out() << "Solution status = " << cplex.getStatus() << endl;
        env.out() << "Solution value = " << cplex.getObjValue() << endl;

        IloNumArray vals(env);
        cplex.getValues(vals, var);
        env.out() << "Values      = " << vals << endl;
    }
    catch (IloException& e) {
        cerr << "Concert exception caught: " << e << endl;
    }
    catch (...) {
        cerr << "Unknown exception caught" << endl;
    }
    env.end();
    return 0;
}
```


CPLEX

tsp (PPI): 2 índices

Novamente, para exemplificar como criar variáveis de 2 índices vamos considerar o modelo a cima que é um pedaço de um problema clássico chamado **tsp simétrico** (vamos estudar ele semana que vem). Por enquanto vamos apenas considerar um modelo sobre um grafo com 1 classe de restrições.

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

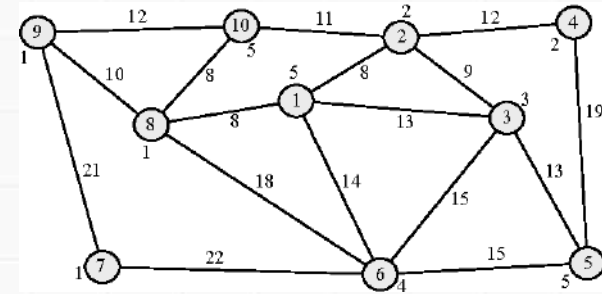
sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 2$$
$$x_{ij} \text{ inteiro}$$



$G=(V,E)$

$j \in N(i) \rightarrow \text{onde } i < j$



CPLEX

tsp (PPI): 2 índices

Novamente, para exemplificar como criar variáveis de 2 índices vamos considerar o modelo a cima que é um pedaço de um problema clássico chamado **tsp simétrico** (vamos estudar ele semana que vem). Por enquanto vamos apenas considerar um modelo sobre um grafo com 1 classe de restrições.

```
#include "data.h"
```

```
int main(int argc, char *argv[])
{
    bool DEPU = true;

    if(argc < 2)
    {
        cout<<"Correct call: ./tsp filename"<<endl;
        exit(1);
    }

    Data data(argc, argv[1]);
    data.readData();

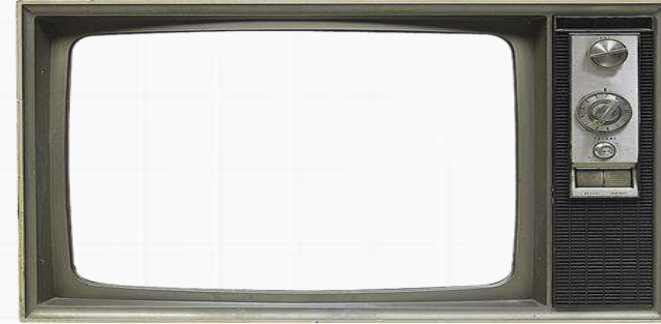
    int dim = data.getDimension();

    // Lê o arquivo da instância
    // dimensao do problema (data.cpp)
```

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

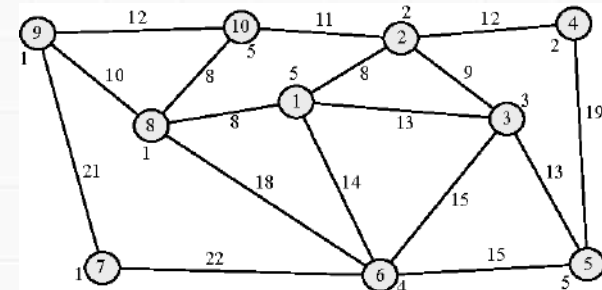
sujeito a:

$$\sum_{j \in N(i)} x_{ij} = 2$$
$$x_{ij} \text{ inteiro}$$



$G=(V,E)$

$j \in N(i) \rightarrow \text{onde } i < j$



A matriz de custo agora é lida de um arquivo "test.tsp" por funções implementadas num pacote específico data.h (não precisa olhar a biblioteca)

CPLEX

tsp (PPI): 2 índices

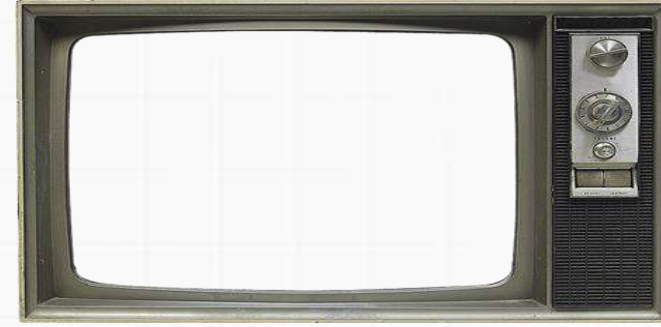
```
...  
IloEnv env;  
IloModel model(env);  
IloArray<IloBoolVarArray> x(env, dim);  
  
char var_name[100];  
  
//variaveis x_uv  
for(int i=0; i < dim - 1; i++)  
{  
    x[i] = IloBoolVarArray(env, dim);  
    for(int j=i+1; j < dim; j++)  
    {  
        sprintf (var_name, "x_%d_%d", (int)i,(int)j);  
        x[i][j] = IloBoolVar(env, var_name);  
        model.add(x[i][j]);  
    }  
}  
...
```

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

sujeito a:

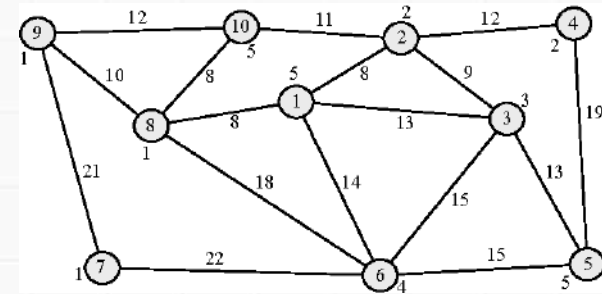
$$\sum_{j \in N(i)} x_{ij} = 2$$

x_{ij} inteiro



$G=(V,E)$

$j \in N(i) \rightarrow$ onde $i < j$



CPLEX

tsp (PPI): 2 índices

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

sujeito a:

$$\sum_{j \in N} x_{ij} = 2$$

x_{ij} inteiro


$$G=(V,E)$$

$j \in N(i) \rightarrow \text{onde } i < j$

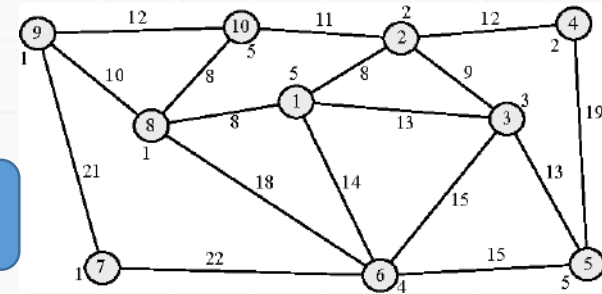
```
IloConstraintArray degree2_constraints(env);
```

```
for(int i=0; i < dim; i++) {
    IloExpr constraint(env);
    for(int j=i+1; j < dim; j++) {
        constraint += x[i][j];
    }
    for(int j=0; j < i; j++) {
        constraint += x[j][i];
    }
    degree2_constraints.add(constraint == 2);
    constraint.end();
}
```

```
model.add(degree2_constraints);
```

vizinhos de i (maiores que i)

vizinhos de i (menores que i)



Cplex

tsp (PPI): 3 índices

$$\text{MINE} \sum_{ijk \in A} c_{ijk} x_{ijk}$$

sujeito a:

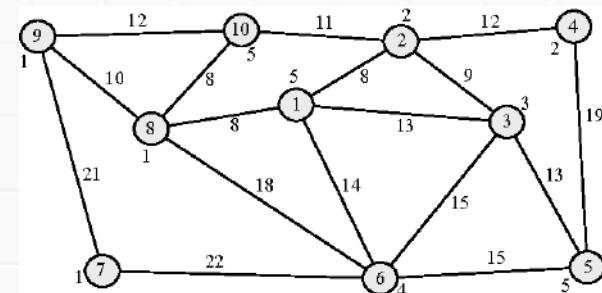
$$\sum_{j,k \in N} x_{ijk} = 2$$


$$G=(V,E)$$

$N(i) \rightarrow$ vizinhos de i

```
char var_name[100];
IloEnv env;
IloModel model(env);
IloArray<IloArray<IloBoolVarArray>> x(env,n1);

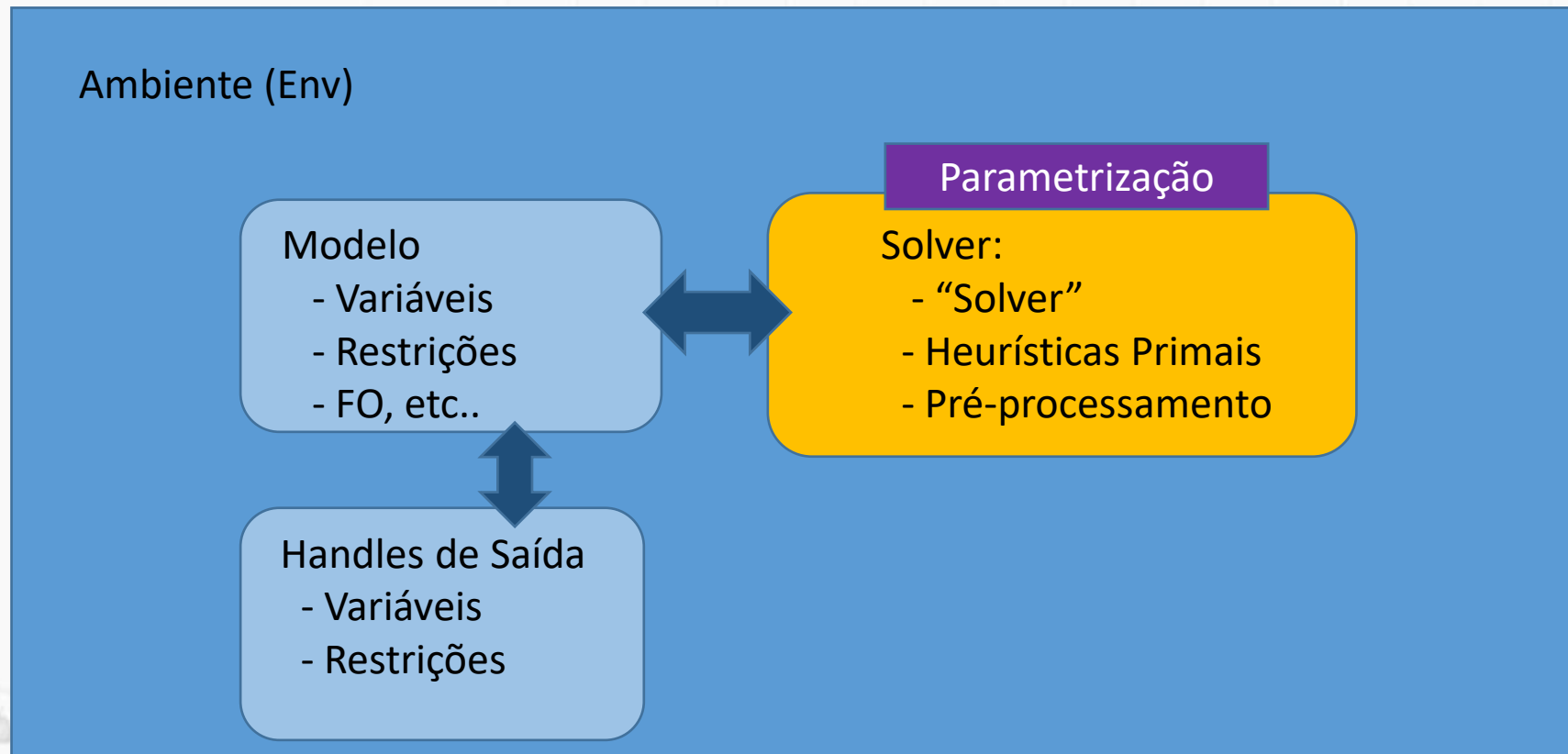
for(int i=0; i < n1; i++)
{
    x[i] = IloArray<IloBoolVarArray>(env, n2);
    for(int j=0; j < n2; j++)
    {
        x[i][j] = IloBoolVarArray(env, n3);
        for(int k=0; k < data._k; k++)
        {
            sprintf (var_name, "x_%d_%d_%d", (i
            x[i][j][k] = IloBoolVar(env, var_na
            model.add(x[i][j][k]);
        }
    }
}
}
```



CPLEX



Parâmetros do método: tudo pode ser configurado



CPLEX



Parâmetros do método: tudo pode ser configurado

- Exemplo:

```
// Parametros do CPLEX
solver.setParam(IloCplex::Param::WorkMem, 1024*2); //tamanho de RAM utilizada maxima
solver.setParam(IloCplex::Param::MIP::Strategy::File, 2); //quando a RAM acaba, 1-guarda nos na memoria e compactado 2-guarda
solver.setParam(IloCplex::Param::TimeLimit, 3600); // tempo limite
solver.setParam(IloCplex::Param::Threads, 1); // Numero de threads
solver.setParam(IloCplex::Param::MIP::Interval, 100); // Log a cada N nos
solver.setOut(env.getNullStream()); // Desabilitando saida do cplex
```

CPLEX

- Manual: [link](#)



Tarefa de Casa

- 1) Baixar e instalar Python-MIP, e executar os 4 exemplos fornecidos.
- 2) Baixar e instalar CPLEX, e executar os 4 exemplos fornecidos



Até a próxima



