

Pesquisa Operacional | 6PES004

Python Pyomo

Programação Inteira Mista aplicada ao
Planejamento de Produção Industrial

João Marcos Tischner

São Bento do Sul – SC, maio de 26



O que é o Python Pyomo?

Open-source

Biblioteca Python gratuita, sem licença. Mantida pelo Sandia National Laboratories.

Multi-solver

Integra com HiGHS, GLPK, CPLEX e +20 solvers externos via SolverFactory.

PL, MIP, NL

Suporta Programação Linear, Inteira Mista e Não-Linear no mesmo framework.

Todo modelo tem 4 elementos:

Var → Variáveis

Objective → Função Obj.

Constraint → Restrições

SolverFactory → Resolver



O Problema: Planejamento de Produção

Contexto

Uma fábrica precisa decidir quantas unidades produzir mensalmente de 5 linhas de produtos eletrônicos para maximizar o lucro, respeitando os limites de capacidade fabril, demanda de mercado e orçamento de matéria-prima.

Classificação do modelo

MIP	5 variáveis inteiras
10 restrições	Solver HiGHS

Produtos & Lucro Unitário

Produto	Lucro/un.
Smartphone (x_1)	R\$ 800
Tablet (x_2)	R\$ 1.200
Notebook (x_3)	R\$ 2.000
Smartwatch (x_4)	R\$ 400
Fone (x_5)	R\$ 150

Objetivo: Maximizar $Z = 800x_1 + 1200x_2 + 2000x_3 + 400x_4 + 150x_5$

O Código

```
from pyomo.environ import *
m = ConcreteModel()
# Índices: 0=Smartphone, 1=Tablet, 2=Notebook, 3=Smartwatch, 4=Fone
produtos = [0, 1, 2, 3, 4]
# Variáveis de decisão inteiras não-negativas
m.x = Var(produtos, domain=NonNegativeIntegers)
# Lucro unitário por produto (R$)
lucro = [800, 1200, 2000, 400, 150]
# Função objetivo
m.obj = Objective(
    expr=sum(lucro[i] * m.x[i] for i in produtos),
    sense=maximize
)
# Restrições operacionais
m.r1 = Constraint(expr=2*m.x[0]+3*m.x[1]+5*m.x[2]+m.x[3]+0.5*m.x[4] <=
1200)
m.r2 = Constraint(expr=m.x[0]+1.5*m.x[1]+2*m.x[2]+0.5*m.x[3]+0.25*m.x[4]
<= 600)
m.r3 = Constraint(expr=2*m.x[0]+3*m.x[1]+4*m.x[2]+m.x[3]+m.x[4] <= 2000)# Demandas máximas de mercado
m.r4 = Constraint(expr=m.x[0] <= 300) # Smartphone
m.r5 = Constraint(expr=m.x[1] <= 150) # Tablet
m.r6 = Constraint(expr=m.x[2] <= 80) # Notebook
m.r7 = Constraint(expr=m.x[3] <= 500) # Smartwatch
m.r8 = Constraint(expr=m.x[4] <= 800) # Fone de Ouvido
# Contrato mínimo com varejistas e orçamento de matéria-prima
m.r9 = Constraint(expr=m.x[0]+m.x[1]+m.x[2]+m.x[3]+m.x[4] >= 200)
m.r10 = Constraint(
    expr=300*m.x[0]+500*m.x[1]+900*m.x[2]+150*m.x[3]+50*m.x[4] <=
400000
)
# Resolução
solver = SolverFactory('highs')
result = solver.solve(m, tee=True)
# Impressão dos resultados
nomes = ['Smartphone','Tablet','Notebook','Smartwatch','Fone']
print('\n' + '='*50)
print('SOLUÇÃO ÓTIMA:')
for i in produtos:
    print(f' {nomes[i]}: {int(value(m.x[i]))} unidades')
print(f' Lucro Total: R$ {value(m.obj):.2f}')
print(f' Status: {result.solver.termination_condition}')
```

As 10 Restrições do Modelo

R1

Cap. Montagem

$$2x_1 + 3x_2 + 5x_3 + x_4 + 0,5x_5 \leq 1.200 \text{ h}$$

R2

Cap. Teste QA

$$x_1 + 1,5x_2 + 2x_3 + 0,5x_4 + 0,25x_5 \leq 600 \text{ h}$$

R3

Chips disponíveis

$$2x_1 + 3x_2 + 4x_3 + x_4 + x_5 \leq 2.000 \text{ un.}$$

R4

Demanda Smartphone

$$x_1 \leq 300 \text{ un.}$$

R5

Demanda Tablet

$$x_2 \leq 150 \text{ un.}$$

R6

Demanda Notebook

$$x_3 \leq 80 \text{ un.}$$

R7

Demanda Smartwatch

$$x_4 \leq 500 \text{ un.}$$

R8

Demanda Fone

$$x_5 \leq 800 \text{ un.}$$

R9

Contrato mínimo

$$x_1 + x_2 + x_3 + x_4 + x_5 \geq 200 \text{ un.}$$

R10

Orçamento MP

$$300x_1 + 500x_2 + 900x_3 + 150x_4 + 50x_5 \leq \text{R\$}400.000$$

Log de saída

Running HiGHS 1.14.0 (git hash: 7df0786): Copyright (c) 2026 under MIT
licence

MIP has 10 rows; 5 cols; 30 nonzeros; 5 integer variables (0 binary)

Coefficient ranges:

Matrix [2e-01, 9e+02] / Cost [2e+02, 2e+03] / RHS [8e+01, 4e+05]

Presolving model

4 rows, 5 cols, 20 nonzeros 0s

Presolve reductions: rows 4(-6); columns 5(-0); nonzeros 20(-10)

Objective function is integral with scale 0.02

Solving MIP model with:

4 rows / 5 cols (5 integer, 0 binary) / 20 nonzeros

Nodes | B&B Tree | Objective Bounds | Work

J 0 0 0 0.00% inf 270000 Large 0 0 0 0 0.0s

T 0 0 0 0.00% 900000 480000 87.50% 0 0 0 1 0.0s

1 0 1 100.00% 480000 480000 0.00% 0 0 0 1 0.0s

Solving report

Status Optimal

Primal bound 480000

Dual bound 480000

Gap 0% (tolerance: 0.01%)

Solution status feasible / 480000 (objective)

int. viol. 0 / bound viol. 0 / row viol. 0

Timing 0.00s / Nodes: 1

=====

SOLUÇÃO ÓTIMA:

Smartphone: 0 unidades

Tablet: 100 unidades

Notebook: 80 unidades

Smartwatch: 500 unidades

Fone: 0 unidades

Lucro Total: R\$ 480.000,00

Status: optimal

Solução Ótima Encontrada

R\$ 480.000

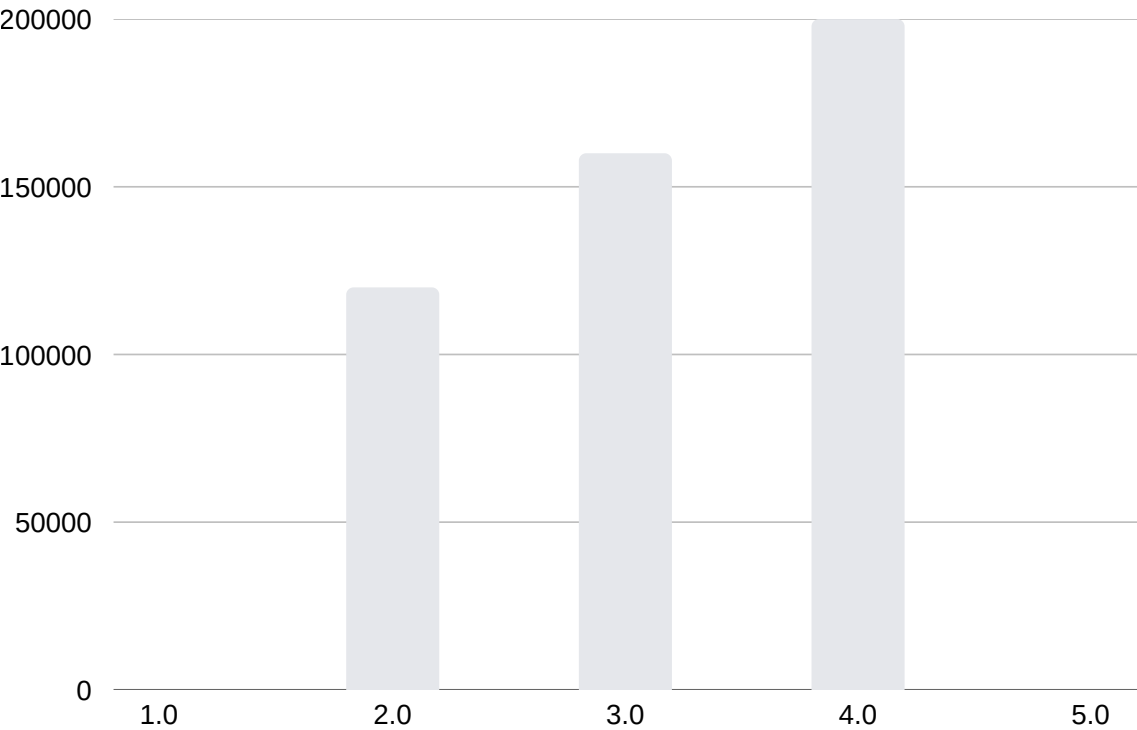
Lucro Total Mensal

680 un.

Total Produzido

0% Gap

Optimalidade Global



Mix de Produção Ótimo

Produto	Qtd.	Lucro
Smartphone	0	R\$ 0
Tablet	100	R\$ 120k
Notebook	80	R\$ 160k
Smartwatch	500	R\$ 200k
Fone	0	R\$ 0

Análise de Gargalos

Gargalos Ativos (folga = 0)

R1 – Horas de Montagem

Usado: 1.200 / 1.200 h

+100h de montagem → +~R\$ 13.200 de lucro (33 Smartwatches extras)

R6 – Demanda Notebook

Usado: 80 / 80 unidades

Produto de maior lucro unitário (R\$ 2.000). Cada unidade extra vale 5 Fones.

R7 – Demanda Smartwatch

Usado: 500 / 500 unidades

41,7% do lucro total. Ampliar cota com varejistas = maior ROI por R\$ investido.

Recursos Subutilizados

Orçamento MP

Folga: 50,75% — R\$ 203.000 livres

Chips

Folga: 44% — 880 un. ociosas

Teste QA

Folga: 6,7% — 40 horas livres

Capital e insumos NÃO são o gargalo: O limitador é operacional e comercial.

Conclusões

01

Resultado verificado

R\$ 480.000 de lucro ótimo · Gap 0% · Tempo < 0,01s · Solução inteira garantida pelo Branch & Bound.

02

Gargalo principal

Capacidade de montagem (R1) limita tudo. Cada hora extra adiciona ~R\$ 132 de lucro.

03

Produtos campeões

Notebook (R\$ 2k/un.) e Smartwatch (volume) são as linhas prioritárias. Smartphones e Fones ficaram fora.

04

Capital não é o problema

R\$ 203.000 do orçamento de MP ficaram livres. O limitador é operacional, não financeiro.

05

Pyomo na prática de SI

Modelos integráveis a pipelines Python (Pandas, Dash) — diferencial direto para o profissional de Sistemas de Informação.