

UNIVERSIDADE DO ESTADO DE SANTA CATARINA | UDESC

Bacharelado em Sistemas de Informação

Pesquisa Operacional | 6PES004

ATIVIDADE DE EXTENSÃO

Python Pyomo

Programação Linear e Inteira Mista aplicada ao Planejamento de Produção

João Marcos Tischner

São Bento do Sul – SC, maio de 26

1. Manual de uso: Python Pyomo

1.1. O que é o Pyomo?

O Pyomo (Python Optimization Modeling Objects) é uma biblioteca open-source escrita em Python que permite formular e resolver problemas de otimização matemática de forma estruturada e reprodutível. Desenvolvido originalmente pelo Sandia National Laboratories e mantido ativamente pela comunidade, o Pyomo suporta Programação Linear (PL), Programação Inteira Mista (MIP), Programação Não-Linear (NL) e outros paradigmas de otimização.

Por ser construído sobre Python, o Pyomo integra-se naturalmente a todo o ecossistema da linguagem: é possível carregar dados de arquivos CSV com Pandas, visualizar resultados com Matplotlib e versionar modelos com Git. Para profissionais de Sistemas de Informação, essa característica representa um diferencial estratégico em relação a ferramentas fechadas ou de uso exclusivamente manual.

1.2. Instalação e Configuração

Pré-requisito: Python 3.8 ou superior instalado. Recomenda-se criar um ambiente virtual para isolar as dependências do projeto.

Passo 1 - Instalar o Pyomo via pip (gerenciador de pacotes do Python):

pip install pyomo

Passo 2 - Instalar o solver HiGHS (gratuito, open-source, sem necessidade de licença comercial):

pip install highspy

Passo 3 - Verificar a instalação executando no terminal:

python -c "import pyomo.environ as pyo; print('Pyomo OK!')"

O Pyomo não possui interface gráfica. Todo o trabalho é feito por scripts Python (.py) executados via terminal ou IDEs como VS Code, PyCharm ou Jupyter Notebook. Essa abordagem favorece a reprodutibilidade e o versionamento de modelos.

1.3. Estrutura de um Modelo Pyomo

Todo modelo Pyomo segue quatro elementos fundamentais:

- Variáveis de decisão (Var): representam as incógnitas do problema. O domínio define o tipo permitido: NonNegativeReals para valores contínuos, NonNegativeIntegers para inteiros, Binary para variáveis 0/1.
- Função objetivo (Objective): expressão matemática a ser maximizada (lucro, receita) ou minimizada (custo, tempo). Definida com sense=maximize ou sense=minimize.
- Restrições (Constraint): inequações ou equações que delimitam o espaço de soluções viáveis, representando limites físicos, contratuais ou financeiros do problema.
- Solver: motor de cálculo externo acionado via SolverFactory. O Pyomo suporta mais de 20 solvers, entre eles HiGHS, GLPK, CPLEX e Gurobi.

O Pyomo oferece dois paradigmas: ConcreteModel (dados inseridos diretamente no código, ideal para problemas pontuais) e AbstractModel (estrutura separada dos dados, ideal para instancias parametrizáveis). Neste trabalho utilizamos ConcreteModel.

1.4. Exemplo Didático - Problema de Duas Variáveis

Problema fictício: Uma fábrica produz dois produtos, A (x_1) e B (x_2). O lucro unitário é R\$ 5 para A e R\$ 4 para B. Há disponibilidade de 24 horas de produção e 6 horas de embalagem por mês. Objetivo: maximizar o lucro total.

Modelagem matemática:

Maximizar: $Z = 5x_1 + 4x_2$

Sujeito a:

$6x_1 + 4x_2 \leq 24$ (horas de produção)

$x_1 + 2x_2 \leq 6$ (horas de embalagem)

$x_1, x_2 \geq 0$

Código Python Pyomo completo:

```
from pyomo.environ import *

m = ConcreteModel()

# 1. Variáveis de decisão
m.x1 = Var(domain=NonNegativeReals)
m.x2 = Var(domain=NonNegativeReals)

# 2. Função objetivo
m.obj = Objective(expr=5*m.x1 + 4*m.x2, sense=maximize)

# 3. Restrições
m.r1 = Constraint(expr=6*m.x1 + 4*m.x2 <= 24)
m.r2 = Constraint(expr=m.x1 + 2*m.x2 <= 6)

# 4. Resolver
solver = SolverFactory('highs')
result = solver.solve(m, tee=True)

# 5. Exibir resultados
print(f'x1 = {value(m.x1):.2f}')
print(f'x2 = {value(m.x2):.2f}')
print(f'Lucro = R$ {value(m.obj):.2f}')
```

Saída gerada:

x1 = 3.00

x2 = 1.50

Lucro = R\$ 21.00

Interpretação: Produzir 3 unidades de A e 1,5 de B gera o lucro máximo de R\$ 21,00. Ambas as restrições estão ativas (folga = 0), ou seja, todos os recursos são consumidos integralmente na solução ótima.

1.5. Elementos da Resposta do Solver

Ao executar `solver.solve(m, tee=True)`, o HiGHS imprime o log completo da resolução. Os principais campos para análise são descritos na tabela abaixo:

Elemento	Significado para Análise
Status: Optimal	Solução ótima encontrada. Qualquer outro status (Infeasible, Unbounded) indica problema na modelagem.
Primal bound	Melhor valor da função objetivo encontrado (resultado final do solver).
Dual bound	Limite teórico superior. Converge para o Primal bound quando Gap = 0%.
Gap	Diferença percentual entre os bounds. Gap = 0% garante que a solução é ótima global.
Nodes	Nós explorados no Branch & Bound. Para problemas lineares contínuos, sempre 0.
Timing	Tempo total de resolução em segundos.
value(m.x)	Função Pyomo que retorna o valor numérico de qualquer variável ou expressão após a resolução.

2. Resolução de um problema complexo utilizando a ferramenta Pyomo

2.1. Descrição do problema

Uma indústria do setor de eletrônicos de consumo deseja **determinar o mix de produção mensal ótimo para cinco linhas de produtos**: Smartphone, Tablet, Notebook, Smartwatch e Fone de Ouvido. **O objetivo é maximizar o lucro bruto total** respeitando os limites operacionais da fábrica, as restritas cotas de demanda de mercado e o orçamento mensal disponível para aquisição de matéria-prima.

O problema é classificado como Programação Inteira Mista (MIP, Mixed Integer Programming), porque as quantidades produzidas são necessariamente valores inteiros - não faz sentido produzir 2,7 notebooks. Essa classificação ativa o algoritmo Branch and Bound no solver, que é uma das funcionalidades avançadas exploradas neste trabalho.

2.2. Modelagem Matemática

As cinco variáveis de decisão representam a quantidade mensal produzida de cada produto (inteiras, não-negativas):

Variável	Produto	Lucro Unit. (R\$)	Custo MP Unit. (R\$)
x_1	Smartphone	800	300
x_2	Tablet	1.200	500
x_3	Notebook	2.000	900
x_4	Smartwatch	400	150
x_5	Fone de Ouvido	150	50

Maximizar: $Z = 800x_1 + 1.200x_2 + 2.000x_3 + 400x_4 + 150x_5$

Sujeito as restrições: *(10 no total, sem contar a de não-negatividade):*

$2x_1 + 3x_2 + 5x_3 + x_4 + 0,5x_5 \leq 1.200$ h/mês (Capacidade montagem)

$x_1 + 1,5x_2 + 2x_3 + 0,5x_4 + 0,25x_5 \leq 600$ h/mês (Capacidade teste QA)

$2x_1 + 3x_2 + 4x_3 + x_4 + x_5 \leq 2.000$ un. (Chips dispon.)

$x_1 \leq 300$ un. (Demanda máx. Smartphone)

$x_2 \leq 150$ un. (Demanda máx. Tablet)

$x_3 \leq 80$ un. (Demanda máx. Notebook)

$x_4 \leq 500$ un. (Demanda máx. Smartwatch)

$x_5 \leq 800$ un. (Demanda máx. Fone)

$x_1 + x_2 + x_3 + x_4 + x_5 \geq 200$ un. (Contrato mínimo)

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

$x_1, x_2, x_3, x_4, x_5 \in \mathbb{Z}^+$ (inteiros não-negativos)

(Problema classificado como MIP - Mixed Integer Programming)

2.3. Implementação em Python Pyomo

```
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}')
```

2.4. Log de Execução do Solver

O log a seguir foi gerado pela execução real do código acima com o solver **HiGHS 1.14.0** no ambiente de desenvolvimento do trabalho:

```
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

2.5. Discussão da Solução Ótima

O solver encontrou a solução ótima em menos de 0,01 segundo, explorando apenas 1 nó no Branch & Bound, com Gap = 0%, garantindo que nenhuma outra combinação inteira é capaz de superar R\$ 480.000,00 sob as restrições definidas. A tabela a seguir resume o mix ótimo de produção e a contribuição de cada produto ao lucro total:

Produto	Qtd. (un.)	Lucro unit.	Lucro total	% do lucro
Smartphone	0	R\$ 800	R\$ 0	0,0%
Tablet	100	R\$ 1.200	R\$ 120.000	25,0%
Notebook	80	R\$ 2.000	R\$ 160.000	33,3%
Smartwatch	500	R\$ 400	R\$ 200.000	41,7%
Fone de Ouvido	0	R\$ 150	R\$ 0	0,0%
TOTAL	680 un.	–	R\$ 480.000	100%

Um ponto relevante da solução é que o Smartwatch, produto com o segundo menor lucro unitário (R\$ 400), responde por 41,7% do lucro total. Isso ocorre porque ele consome apenas 1 hora de montagem por unidade, enquanto o Notebook consome 5 horas. Sendo a montagem o recurso mais escasso, o solver maximiza naturalmente o lucro por hora de montagem consumida, e não apenas o lucro unitário isolado.

Smartphones e Fones não entram na solução pelo mesmo motivo: o Smartphone gera R\$ 400/hora de montagem ($800 \div 2h$) e o Fone apenas R\$ 300/hora ($150 \div 0,5h$), índices inferiores ao rendimento do mix ótimo. O solver os descarta em favor de produtos com melhor eficiência no uso do recurso crítico (montagem).

2.6. Análise de Gargalos e Sensibilidade

Restrições ativas (folga = 0) são os gargalos do modelo: qualquer relaxamento nelas, seja por investimento, negociação ou expansão, aumentaria diretamente o lucro ótimo. A tabela a seguir apresenta a situação de cada restrição na solução ótima:

Restrição	Consumido	Limite	Folga	Status
Horas de Montagem	1.200 h	1.200 h	0 h	ATIVO GARGALO
Horas de Teste QA	560 h	600 h	40 h	Folga disponível
Chips Disponíveis	1.120 un.	2.000 un.	880 un.	Folga elevada
Demanda Notebook	80 un.	80 un.	0 un.	ATIVO GARGALO
Demanda Smartwatch	500 un.	500 un.	0 un.	ATIVO GARGALO
Contrato mínimo	680 un.	200 un.	480 un.	Folga elevada
Orçamento MP	R\$ 197.000	R\$ 400.000	R\$203.000	Folga de 50,75%

Três gargalos foram identificados na solução ótima:

- **Horas de Montagem:** Todas as 1.200 horas mensais foram integralmente consumidas. Este é o principal gargalo do modelo, pois limita simultaneamente todos os produtos. Cada hora adicional de capacidade de montagem vale diretamente mais lucro. Se a empresa ampliar a capacidade para 1.300 h/mês (+100h), poderia adicionar aproximadamente 33 Smartwatches extras, gerando mais R\$ 13.200 ao lucro. Alternativas: horas extras, terceirização ou automação de etapas de montagem.
- **Demanda de Notebook:** O teto de 80 unidades foi atingido. O Notebook é o produto de maior lucro unitário (R\$ 2.000), tornando qualquer ampliação do canal de vendas altamente rentável. Cada Notebook adicional vale mais de 5 Fones ou 2,5 Smartwatches em termos de retorno unitário.

- Demanda de Smartwatch: O limite de 500 unidades foi atingido. Com 41,7% do lucro total e custo de matéria-prima baixo (R\$ 150/un.), o Smartwatch é o produto com melhor equilíbrio entre volume, custo e lucro. Ampliar a demanda contratada com varejistas para esta linha seria a ação de maior impacto e menor risco financeiro.

Ao contrário disso, três restrições apresentam folga significativa: o orçamento de matéria-prima utilizou apenas R\$ 197.000 dos R\$ 400.000 disponíveis (folga de 50,75%), os chips tiveram 880 unidades ociosas e o setor de QA encerrou o mês com 40 horas não utilizadas. Isso demonstra que capital e insumos não são os fatores limitantes, mas sim o gargalo central é operacional (capacidade de montagem) e comercial (tetos de demanda).

3. Conclusão

O Python Pyomo demonstrou ser uma ferramenta robusta, precisa e diretamente alinhada ao perfil profissional do bacharel em Sistemas de Informação. Por ser baseado em Python, integra-se naturalmente a pipelines de dados, sistemas de apoio à decisão e dashboards analíticos, o que é um diferencial frente a ferramentas isoladas como planilhas ou IDEs especializados.

O problema modelado simulou um cenário real de planejamento de produção industrial com 5 variáveis inteiras e 10 restrições. A solução ótima de R\$ 480.000,00 foi encontrada em menos de 0,01 segundo com Gap = 0%, garantindo uma solução ótima e integralidade das variáveis de decisão.

A análise crítica revelou que os verdadeiros limitadores do negócio são a capacidade de montagem e os tetos de demanda dos produtos mais rentáveis, enquanto capital e insumos estão amplamente subutilizados. Essa informação demonstra a importância da Pesquisa Operacional aplicada em contextos organizacionais reais, a fim de orientar investimentos e negociações para um potencial crescimento estratégico de uma empresa.

3.1. Referências

BYNUM, M. L. et al. Pyomo – Optimization Modeling in Python. 3. ed. New York: Springer, 2021.

HILLIER, F. S.; LIEBERMAN, G. J. Introdução à Pesquisa Operacional. 9. ed. Porto Alegre: AMGH, 2013.

PYOMO PROJECT. Pyomo Documentation – Pyomo 6.x. Disponível em: <https://pyomo.readthedocs.io>. Acesso em: maio 2026.