

## Modelagem

### Variáveis

$NumCaminhoes_{up,t,tr}$ : representa a quantidade de caminhões da transportadora na UP no dia específico.

$DecisaoTransportadoraFazenda_{f,t,tr}$ : indica se a transportadora ( tr ) está operando na fazenda ( f ) no dia ( t ) (1 para "sim", 0 para "não").

$DecisaoTransportadoraUP_{up,t,tr}$ : indica se a transportadora ( tr ) está transportando a madeira da UP ( up ) no dia ( t ) (1 para "sim", 0 para "não").

$PresencaFazenda_f$ : indica se a operação está ocorrendo na fazenda ( f ) (1 para "sim", 0 para "não"). .

### Parâmetros

$Mapeamento\_UP\_Fazenda$ : Mapeamento das Unidades Produtivas (UP) com as fazendas.

$Demanda\_Min\_Diaria_t$ : Demanda mínima por dia ( t ).

$Demanda\_Max\_Diaria_t$ : Demanda máxima por dia ( t ).

$Tempo\_Ciclo_{tr,up}$ : Tempo de ciclo da transportadora ( tr ) para a UP ( up ).

$Frota\_Max\_Transportadora_{tr}$ : Frota máxima da transportadora ( tr ).

$Frota\_Min\_Transportadora_{tr}$ : Frota mínima da transportadora ( tr ). .

### Função Objetivo

Minimizar a variação diária do BD, ou seja, reduzir as variações diárias na quantidade de madeira (BD) entregue.

$$\text{Minimizar} \left( \max \left( \sum_{up,tr} v_{up,t,tr} \right) - \min \left( \sum_{up,tr} v_{up,t,tr} \right) \right) \forall t$$

### Restrições

#### Restrição de atendimento de demanda mínima e máxima diária

$RestricaoDemanda_t$ : Garante que a quantidade de madeira entregue atende à demanda mínima e máxima da fábrica no dia ( t ).

$$Demanda_{\min,t} \leq \sum_{up,tr} v_{up,t,tr} \leq Demanda_{\max,t} \quad \forall t$$

#### Restrição de frota mínima e máxima diária por transportadora

$RestricaoFrota_{t,tr}$ : Garante que a quantidade de caminhões utilizados pela transportadora ( tr ) está dentro da frota mínima e máxima disponível no dia ( t ).

$$Frota_{\min,tr} \leq \sum_{up} NumCaminhoes_{up,t,tr} \leq Frota_{\max,tr} \quad \forall t, \forall tr$$

#### Restrição de volume das Unidades Produtivas

$RestricaoVolumeUP_{up,tr}$ : Garante que o volume total de madeira transportado da UP ( up ) pela transportadora ( tr ) não excede o volume disponível na UP.

$$\sum_t v_{up,t,tr} \leq Volume_{up} \quad \forall up, \forall tr$$

#### Restrição de qualidade média de RSP

$RestricaoMediaRSP_t$ : Garante que a qualidade média em RSP da madeira entregue está dentro dos limites definidos para o dia ( t ).

$$RSP_{\min,t} \times \sum_{up,tr} v_{up,t,tr} \leq \sum_{up,tr} v_{up,t,tr} \times RSP_{up} \leq RSP_{\max,t} \times \sum_{up,tr} v_{up,t,tr} \quad \forall t$$

#### Restrição de decisão de transporte

*RestricaoDecisaoTransporte<sub>t</sub>*: Garante que cada transportadora ( *tr* ) opera em uma única fazenda por dia.

$$\sum_f d_{f,t,tr} = 1 \quad \forall t, \forall tr$$

#### Restrição de transporte fracionado para UPs menores que 7.000 m³

*RestricaoDecisaoTransporteFrac<sub>t</sub>*: Se uma UP tem menos de 7.000 m³, ela deve ser transportada de uma só vez, sem dividir o transporte em vários dias.

$$v_{up,t,tr} = \begin{cases} \text{Volume}_{up} & \text{se } \sum_{t,tr} x_{up,t,tr} = 1 \text{ e } \text{Volume}_{up} < 7000 \\ 0 & \text{caso contrário} \end{cases}$$

#### Restrição sobre a troca de fazendas e UPs após completar o volume

*DecisaoTransportadoraUP<sub>up,t,tr</sub>*: Se uma UP começou a ser transportada em um dia ( *t* ) e não foi totalmente esgotada, ela deve continuar sendo transportada no dia ( *t + 1* ).

$$d_{up,t,tr} \leq d_{up,t+1,tr} \quad \forall t, \forall tr, \forall up$$

#### Restrição sobre a presença de transportadoras em fazendas diferentes ao mesmo tempo

*RestricaoPresenca<sub>f,t,tr</sub>*: Cada transportador só pode estar em uma fazenda a cada momento.

$$\sum_f d_{f,t,tr} = 1 \quad \forall t, \forall tr$$

```

1  # Instalando bibliotecas necessárias
2  # A biblioteca PuLP é uma biblioteca de otimização linear em Python
3  # CPLEX é um solver comercial que pode ser usado com PuLP
4  !pip install pulp
5  !pip install cplex
6
7  # Importando bibliotecas padrão
8  import pandas as pd # Para manipulação de dados tabulares
9  import numpy as np  # Para operações matemáticas
10 import matplotlib.pyplot as plt # Para visualização de dados
11 import sys # Acessar funções específicas do sistema
12 import os # Interface com o sistema operacional
13 import shutil # Operações de arquivo de alto nível
14
15 # Importando bibliotecas para otimização
16 from pulp import (
17     LpProblem, # Classe para criar o problema de otimização
18     LpContinuous, # Classe para definir variáveis contínuas
19     LpVariable, # Classe para criar variáveis de decisão
20     lpSum, # Função para somar variáveis
21     LpMinimize, # Classe para definir a minimização do problema
22     LpInteger, # Tipo de variável inteira
23     LpBinary, # Tipo de variável binária (0 ou 1)
24     LpStatus, # Classe para verificar o status da solução
25     LpConstraint,
26     CPLEX_PY # Solver CPLEX para resolver o problema
27 )
28
29 # Importando bibliotecas para visualização avançada
30 import seaborn as sns # Biblioteca de visualização estatística
31 sns.set(style="darkgrid") # Configurando o estilo dos gráficos
32 from mpl_toolkits import mplot3d
33
34 # Importando a biblioteca Pyomo, uma alternativa ao PuLP para otimização
35 ! pip install pyomo
36 import pyomo.environ as pyo
37
38 import shutil
39 print(shutil.which("cbc"))
40
```

```

Collecting pulp
  Downloading PuLP-2.7.0-py3-none-any.whl (14.3 MB)
  14.3/14.3 MB 79.3 MB/s eta 0:00:00
Installing collected packages: pulp
Successfully installed pulp-2.7.0
Collecting cplex
  Downloading cplex-22.1.1.0-cp310-cp310-manylinux1_x86_64.whl (44.2 MB)
  44.2/44.2 MB 12.9 MB/s eta 0:00:00
Installing collected packages: cplex

```

```

Successfully installed cplex-22.1.1.0
Collecting pyomo
  Downloading Pyomo-6.6.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.7 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.7/12.7 MB 68.5 MB/s eta 0:00:00
Collecting ply (from pyomo)
  Downloading ply-3.11-py2.py3-none-any.whl (49 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.6/49.6 kB 6.5 MB/s eta 0:00:00
Installing collected packages: ply, pyomo
Successfully installed ply-3.11 pyomo-6.6.2
None

```

```

1 %%capture
2 import sys
3 import os
4
5 if 'google.colab' in sys.modules:
6     !pip install idaes-pse --pre
7     !idaes get-extensions --to ./bin
8     os.environ['PATH'] += ':bin'
9 import shutil
10 if not (shutil.which("cbc") or os.path.isfile("cbc")):
11     if "google.colab" in sys.modules:
12         !apt-get install -y -qq coinor-cbc
13     else:
14         try:
15             !conda install -c conda-forge coincbc
16         except:
17             pass
18
19 assert(shutil.which("cbc") or os.path.isfile("cbc"))
20
21 import pyomo.environ as pyo
22

```

```

1 # Define o caminho (URL) do arquivo Excel que contém os dados de entrada.
2 # Este arquivo está hospedado no GitHub e é acessível publicamente.
3 path = 'https://raw.githubusercontent.com/SabrinaLameiras/TransformacaoDigital/main/Dados/generic_input_case.xlsx'
4
5 # Função para ler uma planilha específica do arquivo Excel.
6 # Esta função ajuda a evitar a repetição de código e torna o código mais limpo.
7 def read_excel_sheet(sheet_name):
8     """
9     Lê uma planilha específica de um arquivo Excel.
10
11     Parâmetros:
12     - sheet_name (str): Nome da planilha para ler.
13
14     Retorna:
15     - DataFrame: Dados da planilha especificada.
16     """
17     return pd.read_excel(path, sheet_name=sheet_name)
18
19 # Lendo cada planilha do arquivo Excel e armazenando em DataFrames correspondentes.
20
21 # BD_UP contém informações sobre as Unidades de Produção.
22 bd_df = read_excel_sheet('BD_UP')
23
24 # FROTA contém informações sobre a frota de veículos disponível.
25 frota_df = read_excel_sheet('FROTA')
26
27 # HORIZONTE contém informações sobre o horizonte de planejamento.
28 horizonte_df = read_excel_sheet('HORIZONTE')
29
30 # FABRICA contém informações sobre a fábrica e sua demanda.
31 fabrica_df = read_excel_sheet('FABRICA')
32
33 # ROTA contém informações sobre as rotas de transporte.
34 rota_df = read_excel_sheet('ROTA')
35
36 # GRUA contém informações sobre as gruas disponíveis.
37 grua_df = read_excel_sheet('GRUA')

```

```

/usr/local/lib/python3.10/dist-packages/openpyxl/worksheet/_reader.py:329: UserWarning: Data Validation extension is not supported
warn(msg)

```

```

1 # Criando uma lista de DataFrames e seus respectivos nomes para posterior análise.
2 dataframes = [bd_df, frota_df, horizonte_df, fabrica_df, rota_df, grua_df]
3 df_names = ['BD_UP', 'FROTA', 'HORIZONTE', 'FABRICA', 'ROTA', 'GRUA']
4
5 def display_info_and_describe(dfs, df_names):

```

```
6 """
7 Exibe informações básicas e estatísticas descritivas de cada DataFrame.
8
9 Parâmetros:
10     dfs (list): Lista de DataFrames a serem analisados.
11     df_names (list): Lista com os nomes dos DataFrames para etiquetar a saída.
12
13 Retorna:
14     None. Imprime as informações diretamente no console.
15 """
16 for i, df in enumerate(dfs):
17     print("\n" + "-" * 40)
18     print(f"DataFrame: {df_names[i]}")
19     print("-" * 40)
20
21     # Exibe informações básicas do DataFrame, como tipos de dados e contagem de non-null.
22     print("\nInfo:")
23     print("-" * 5)
24     print(df.info())
25
26     # Exibe estatísticas descritivas, como média, desvio padrão, mínimos e máximos.
27     print("\nDescribe:")
28     print("-" * 9)
29     print(df.describe())
30
31     print("\n\n")
32
33 # Chama a função para exibir as informações.
34 display_info_and_describe(dataframes, df_names)
```

```

1 def detect_anomalies(dfs, df_names):
2     """
3     Detecta anomalias em vários dataframes e retorna os índices das linhas com anomalias.
4     """
5     anomalies = {}
6
7     for i, df in enumerate(dfs):
8         df_name = df_names[i]
9         anomalies[df_name] = {}
10
11         # Detecta valores faltantes
12         missing_data = {}
13         for col in df.columns:
14             missing_rows = df[df[col].isnull()].index.tolist()
15             if missing_rows:
16                 missing_data[col] = {
17                     'count': len(missing_rows),
18                     'rows': missing_rows
19                 }
20         anomalies[df_name]['missing'] = missing_data
21
22         # Detecta duplicatas
23         duplicated_rows = df[df.duplicated()].index.tolist()
24         anomalies[df_name]['duplicated'] = {
25             'count': len(duplicated_rows),
26             'rows': duplicated_rows
27         }
28
29         # Detecta outliers usando o método IQR
30         anomalies[df_name]['outliers'] = {}
31         for col in df.select_dtypes(include=['float64', 'int64']).columns:
32             Q1 = df[col].quantile(0.25)
33             Q3 = df[col].quantile(0.75)
34             IQR = Q3 - Q1
35             outlier_rows = df[(df[col] < (Q1 - 1.5 * IQR)) | (df[col] > (Q3 + 1.5 * IQR))].index.tolist()
36             if outlier_rows:
37                 anomalies[df_name]['outliers'][col] = {
38                     'count': len(outlier_rows),
39                     'rows': outlier_rows
40                 }
41
42     return anomalies
43
44 def display_anomalies(anomalies_result):
45     for df_name, anomalies in anomalies_result.items():
46         print("\n-----")
47         print(f"DataFrame: {df_name}")
48         print("-----")
49
50         # Exibir valores faltantes
51         print("\nValores Faltantes:")
52         print("-----")
53         if anomalies['missing']:
54             for col, missing_data in anomalies['missing'].items():
55                 print(f"{col}: {missing_data['count']} valores faltantes nas linhas {missing_data['rows']}")
56         else:
57             print("Nenhum valor faltante.")
58
59         # Exibir duplicatas
60         print("\nDuplicatas:")
61         print("-----")
62         if anomalies['duplicated']:
63             print(f"{anomalies['duplicated']['count']} registros duplicados nas linhas: {anomalies['duplicated']['rows']}")
64         else:
65             print("Nenhum registro duplicado.")
66
67         # Exibir outliers
68         print("\nOutliers:")
69         print("-----")
70         if anomalies['outliers']:
71             for col, outlier_data in anomalies['outliers'].items():
72                 print(f"{col}: {outlier_data['count']} outliers nas linhas: {outlier_data['rows']}")
73         else:
74             print("Nenhum outlier detectado.")
75
76     print("\n\n")
77
78 # Uso da função de exibição
79 anomalies_result = detect_anomalies(dataframes, df_names)
80 display_anomalies(anomalies_result)

```

81

```
-----  
DataFrame: FABRICA  
-----
```

```
Valores Faltantes:  
-----  
Nenhum valor faltante.
```

```
Duplicatas:  
-----  
Nenhum registro duplicado.
```

```
Outliers:  
-----  
Nenhum outlier detectado.
```

```
-----  
DataFrame: ROTA  
-----
```

```
Valores Faltantes:  
-----  
Nenhum valor faltante.
```

```
Duplicatas:  
-----  
Nenhum registro duplicado.
```

```
Outliers:  
-----  
Nenhum outlier detectado.
```

```
-----  
DataFrame: GRUA  
-----
```

```
Valores Faltantes:  
-----  
Nenhum valor faltante.
```

```
Duplicatas:  
-----  
Nenhum registro duplicado.
```

```
Outliers:  
-----  
Nenhum outlier detectado.
```

```
1 def exibir_valores_unicos(*dataframes):  
2     for df in dataframes:  
3         print("="*50)  
4         print(f"DataFrame: {df.name if 'name' in dir(df) else 'Unnamed'}") # tenta exibir o nome do DataFrame, se disponível  
5         print("="*50)  
6         for col in df.columns:  
7             print(f"Coluna: {col}")  
8             print(df[col].unique())  
9             print("-"*50)  
10  
11 # identificação na saída  
12 bd_df.name = "BD_UP"  
13 frota_df.name = "FROTA_DF_DF"  
14 horizonte_df.name = "HORIZONTE_DF"  
15 fabrica_df.name = "FABRICA_DF"  
16 rota_df.name = "ROTA_DF"  
17 grua_df.name = "GRUA_DF"  
18  
19 # Usando a função  
20 exibir_valores_unicos(bd_df, frota_df, horizonte_df, fabrica_df, rota_df, grua_df)
```

```

Coluna: DEMANDA_MIN
[6400]
-----
Coluna: DEMANDA_MAX
[8000]
-----
Coluna: RSP_MIN
[1.4]
-----
Coluna: RSP_MAX
[1.49]
-----
DataFrame: ROTA_DF
=====
Coluna: ORIGEM
['S3AX01' 'S3AX02' 'S3AX03' 'S3AX04' 'S3AX06' 'S5AK05' 'S5AK08' 'S5AK09'
'S5AK10' 'S5AW05' 'S5AW09' 'S5AW10' 'S5AW13' 'S5AW14' 'S6BG08' 'S6BG10'
'S6BG11' 'S6BG12' 'S6BG13' 'S6BG14' 'S6BX01' 'S6C297' 'S6C298' 'S6C334'
'S6C335' 'S6C421']
-----
Coluna: DESTINO
['LIM']
-----
Coluna: TRANSPORTADOR
['Rampazo' 'Pastori' 'Tover']
-----
Coluna: CAIXA_CARGA
[66]
-----
Coluna: TEMPO_CICLO
[1.8]
-----
Coluna: CICLO_LENTO
[0]
-----
Coluna: Fazenda
['INDIANA II' 'TURVO III (LEX)' 'PIRACEMA_GLEBA PULADOR - DX' 'SINIMBU'
'SANTO ANTONIO SALIG.' 'FORTALEZA' 'SIRIEMA']
-----
DataFrame: GRUA_DF
=====
Coluna: TRANSPORTADOR
['Pastori' 'Tover' 'Rampazo']
-----
Coluna: QTD_GRUAS
[2]
-----
Coluna: PORCENTAGEM_VEICULOS_MIN
[0.4]
-----

1 horizonte_df = horizonte_df.drop(columns='CICLO_LENTO')
2 frota_df = frota_df.drop(columns='DIA')
3 bd_df = bd_df.drop(columns=['DATA_COLHEITA', 'IDADE_FLORESTA', 'IMA', 'RD', 'CLONE', 'ESPECIE', 'PRECIPITACAO', 'Unnamed: 13'])
4

1 # Mapeamento das Unidades Produtivas (UP) com as fazendas
2 Mapeamento_UP_Fazenda = tuple(zip(bd_df.UP, bd_df.FAZENDA))
3
4 # Demanda mínima por dia
5 Demanda_Min_Diaria = dict(zip(fabrica_df.DIA, fabrica_df.DEMANDA_MIN))
6
7 # Demanda máxima por dia
8 Demanda_Max_Diaria = dict(zip(fabrica_df.DIA, fabrica_df.DEMANDA_MAX))
9
10 # Tempo de ciclo de cada transportadora e sua origem
11 Tempo_Ciclo = dict(zip(tuple(zip(rota_df.TRANSPORTADOR, rota_df.ORIGEM)), rota_df.TEMPO_CICLO))
12
13 # Frota máxima de cada transportadora
14 Frota_Max_Transportadora = dict(zip(frota_df.TRANSPORTADOR, frota_df.FROTA_MAX))
15
16 # Frota mínima de cada transportadora
17 Frota_Min_Transportadora = dict(zip(frota_df.TRANSPORTADOR, frota_df.FROTA_MIN))
18

1 # create a model
2 model = pyo.ConcreteModel()

1 # Conjuntos
2 model.ConjuntoUPFazenda = pyo.Set(initialize=Mapeamento_UP_Fazenda, dimen=2)
3 model.Fazenda = pyo.Set(initialize = list(set([fazenda for (_, fazenda) in model.ConjuntoUPFazenda])))
4 model.TransportadoraParaUP = pyo.Set(initialize=Tempo_Ciclo.keys(), dimen=2)
5 model.Dias = pyo.Set(initialize=list(range(1, 32)), doc='Dias do mês')

```

```

6 model.Transportadora = pyo.Set(initialize=list(set([transportadora for (transportadora, _) in model.TransportadoraParaUP])), doc='Tr
7 model.UP = pyo.Set(initialize=list(set([up for (_, up) in model.TransportadoraParaUP])), doc='Unidades Produtivas')
8

1 # Parâmetros
2 model.DemandaMin = pyo.Param(model.Dias, initialize=Demanda_Min_Diaria, doc='Demanda mínima no dia')
3 model.DemandaMax = pyo.Param(model.Dias, initialize=Demanda_Max_Diaria, doc='Demanda máxima no dia')
4 model.CargaCaminhao = pyo.Param(initialize=66, doc = 'Capacidade de carga de cada caminhão')
5 model.RSPMinimo = pyo.Param(model.Dias, initialize=dict(zip(fabrica_df.DIA, fabrica_df.RSP_MIN)), doc='RSP mínimo por dia')
6 model.RSPMaximo = pyo.Param(model.Dias, initialize=dict(zip(fabrica_df.DIA, fabrica_df.RSP_MAX)), doc='RSP máximo por dia')
7 model.QualidadeUP = pyo.Param(model.UP, initialize=dict(zip(bd_df.UP, bd_df.RSP)), doc='Qualidade em RSP de cada UP')
8 model.CicloTransportadora = pyo.Param(model.Transportadora, initialize=Tempo_Ciclo, doc='Tempo de ciclo da transportadora')
9 model.FrotaMinima = pyo.Param(model.Transportadora, initialize=Frota_Min_Transportadora, doc='Frota mínima da transportadora')
10 model.FrotaMaxima = pyo.Param(model.Transportadora, initialize=Frota_Max_Transportadora, doc='Frota máxima da transportadora')
11 model.VolumeUP = pyo.Param(model.UP, initialize=dict(zip(bd_df.UP, bd_df.VOLUME)), doc='Volume disponível na UP')
12 model.RSPUP = pyo.Param(model.UP, initialize=dict(zip(bd_df.UP, bd_df.DB)), doc='RSP em cada UP')
13

1 # Construção do conjunto TransportadoraFazendaUP
2 RelacaoTransportadoraFazendaUP = []
3 for (up, fazenda) in model.ConjuntoUPFazenda:
4     for (transportadora, up_origem) in model.TransportadoraParaUP:
5         if up == up_origem:
6             RelacaoTransportadoraFazendaUP.append(list([transportadora, fazenda, up]))
7 model.ConjuntoTransportadoraFazendaUP = pyo.Set(initialize = RelacaoTransportadoraFazendaUP)
8

1 # Variáveis
2 model.NumCaminhoes = pyo.Var(model.ConjuntoTransportadoraFazendaUP, model.Dias, domain=pyo.NonNegativeReals, doc='Quantidade de cami
3 model.DecisaoTransportadoraFazenda = pyo.Var(model.ConjuntoTransportadoraFazendaUP, model.Dias, domain=pyo.Boolean, doc='Decisão da t
4 model.DecisaoTransportadoraUP = pyo.Var(model.TransportadoraParaUP, model.Dias, domain=pyo.Boolean, doc='Decisão da transportadora de
5 model.PresencaFazenda = pyo.Var(model.Fazenda, domain=pyo.Boolean, doc='Decisão de operar na fazenda específica')
6

1 # Restrição de atendimento de demanda mínima e máxima diária
2 model.RestricaoDemanda = pyo.ConstraintList()
3 for dia in model.Dias:
4     centro = sum(model.NumCaminhoes[transportadora, fazenda, up, dia] * model.CicloTransportadora[transportadora, up] * model.CargaCa
5     minimo = model.DemandaMin[dia]
6     maximo = model.DemandaMax[dia]
7     model.RestricaoDemanda.add(centro >= minimo)
8     model.RestricaoDemanda.add(centro <= maximo)

1 # Restrição de frota mínima e máxima diária por transportadora
2 model.RestricaoFrota = pyo.ConstraintList()
3 for dia in model.Dias:
4     for transportadora in model.Transportadora:
5         centro = sum(model.NumCaminhoes[transportadora, :, :, dia])
6         minimo = model.FrotaMinima[transportadora]
7         maximo = model.FrotaMaxima[transportadora]
8         model.RestricaoFrota.add(centro >= minimo)
9         model.RestricaoFrota.add(centro <= maximo)

1 # Restrição de volume das Unidades Produtivas
2 model.RestricaoVolumeUP = pyo.ConstraintList()
3 for (transportadora, fazenda, up) in model.ConjuntoTransportadoraFazendaUP:
4     centro = sum(model.NumCaminhoes[transportadora, fazenda, up, dia] * model.CicloTransportadora[transportadora, up] * model.CargaCa
5     maximo = model.VolumeUP[up]
6     model.RestricaoVolumeUP.add(centro <= maximo)

1 # Restrição de qualidade média de RSP
2 model.RestricaoMediaRSP = pyo.ConstraintList()
3 for dia in model.Dias:
4     numerador = sum(model.NumCaminhoes[transportadora, fazenda, up, dia] * model.CicloTransportadora[transportadora, up] * model.Carq
5     denominador = sum(model.NumCaminhoes[transportadora, fazenda, up, dia] * model.CicloTransportadora[transportadora, up] * model.Ci
6     minimo = model.RSPMinimo[dia]
7     maximo = model.RSPMaximo[dia]
8     model.RestricaoMediaRSP.add(numerador >= minimo * denominador)
9     model.RestricaoMediaRSP.add(numerador <= maximo * denominador)
10

1 # Restrição de decisão de transporte
2 model.RestricaoDecisaoTransporte = pyo.ConstraintList()
3 for dia in model.Dias:
4     for (transportadora, fazenda, up) in model.ConjuntoTransportadoraFazendaUP:
5         centro = sum(model.DecisaoTransportadoraFazenda[transportadora, :, :, dia])
6         model.RestricaoDecisaoTransporte.add(centro == 1)

```



```

7
8

1 def FuncaoObjetivo(model):
2     return sum(model.NumCaminhoes[transportadora, fazenda, up, dia] * model.RSPUP[up] for (transportadora, fazenda, up) in model.Conj
3
4 model.FuncaoObjetivo = pyo.Objective(sense=pyo.minimize, rule=FuncaoObjetivo)
5

1 import shutil
2 print(shutil.which("cbc"))
3
4 resultado = pyo.SolverFactory('cbc').solve(model)
5 resultado.write()

bin/cbc
# =====
# = Solver Results =
# =====
# -----
# Problem Information
# -----
Problem:
- Name: unknown
  Lower bound: 810198.08602868
  Upper bound: 810198.08602868
  Number of objectives: 1
  Number of constraints: 443
  Number of variables: 2480
  Number of binary variables: 1240
  Number of integer variables: 1240
  Number of nonzeros: 1240
  Sense: minimize
# -----
# Solver Information
# -----
Solver:
- Status: ok
  User time: -1.0
  System time: 0.13
  Wallclock time: 0.15
  Termination condition: optimal
  Termination message: Model was solved to optimality (subject to tolerances), and an optimal solution is available.
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 0
      Number of created subproblems: 0
    Black box:
      Number of iterations: 0
  Error rc: 0
  Time: 0.22745585441589355
# -----
# Solution Information
# -----
Solution:
- number of solutions: 0
  number of solutions displayed: 0

1 resultados = []
2
3 # Cálculos que não dependem de transportadora ou fazenda
4 RSP_total = sum(pyo.value(model.NumCaminhoes[t, f, u, dia]) * model.CicloTransportadora[t, u] * model.CargaCaminhao * model.Qualidade
5 transporte_diario = sum(pyo.value(model.NumCaminhoes[t, f, u, dia]) * model.CicloTransportadora[t, u] * model.CargaCaminhao for dia :
6
7 for dia in model.Dias:
8     for (transportadora, fazenda, UP) in model.ConjuntoTransportadoraFazendaUP:
9         qtd = pyo.value(model.NumCaminhoes[transportadora, fazenda, UP, dia])
10
11         # Verificar se a variável foi inicializada e é diferente de zero
12         if qtd is not None and qtd != 0:
13             volume = qtd * model.CicloTransportadora[transportadora, UP] * model.CargaCaminhao
14             resultados.append([UP, fazenda, transportadora, dia, pyo.value(model.QualidadeUP[UP]), RSP_total/transporte_diario, qtd,
15
16 # Criando um DataFrame com os resultados
17 df_resultados = pd.DataFrame(resultados, columns=['UP', 'Fazenda', 'Transportadora', 'Dia', 'DB', 'RSP', 'Quantidade', 'Volume'])
18 import matplotlib.pyplot as plt
19 import seaborn as sns
20
21 def plot_green_colors_bar_chart(df_resultados):
22     # Ordenar o dataframe pela quantidade
23     df_sorted = df_resultados.groupby('Fazenda').sum()['Quantidade'].sort_values(ascending=True).reset_index()
24
25     # Calcular o tamanho da figura com base no número de fazendas
26     fig_height = max(6, len(df_sorted) * 0.6)

```

```

27
28 # Inicializar layout com tamanho calculado
29 fig, ax = plt.subplots(figsize=(6, fig_height))
30
31 # Adicionar fundo branco ao ax e fig
32 ax.set_facecolor('white')
33 fig.set_facecolor('white')
34
35 # Escolher cores com base na cor verde
36 bar_colors = sns.color_palette("Greens", len(df_sorted))
37
38 # Criar o gráfico com altura ajustada das barras
39 bar_height = 0.5
40 ax.barh(df_sorted['Fazenda'], df_sorted['Quantidade'], color=bar_colors, zorder=2, height=bar_height)
41
42 # Adicionar linhas verticais cinza
43 ax.grid(linestyle='-', alpha=0.5, color='lightgray', axis='x')
44
45 # Título do gráfico com espaçamento e tamanho de fonte ajustados
46 title = 'Distribuição da Quantidade de DB por Fazenda'
47 fig.text(0.03, 1.02, title, fontsize=18, fontweight='bold', ha='left', color='black', family='dejavu sans')
48
49 # Subtítulo do gráfico com espaçamento e tamanho de fonte ajustados
50 subtitle = 'Quantidade acumulada por Fazenda'
51 fig.text(0.03, 0.98, subtitle, fontsize=10, color='black', ha='left', family='dejavu sans')
52
53 # Remover as bordas do gráfico
54 for spine in ax.spines.values():
55     spine.set_visible(False)
56
57 # Configurar posição e etiquetas dos eixos com tamanho de fonte ajustado
58 ax.tick_params(axis='y', labelsize=9, colors='black')
59 ax.tick_params(axis='x', colors='black')
60 ax.xaxis.tick_top()
61
62 # Ajustar margens
63 plt.subplots_adjust(left=0.2, right=0.8, top=0.9, bottom=0.1)
64
65 # Exibir o gráfico
66 plt.show()
67
68 plot_green_colors_bar_chart(df_resultados)
69
70 # Save df_resultados to a CSV file
71 df_resultados.to_csv('resultados.csv', index=False)
72
73

```

<ipython-input-21-18cf797e3edf>:23: FutureWarning: The default value of numeric\_only in DataFrameGroupBy.sum is deprecated. In a fut  
df\_sorted = df\_resultados.groupby('Fazenda').sum()['Quantidade'].sort\_values(ascending=True).reset\_index()



