Pedro Henrique Andrade Trindade

[16]:
```python
import pandas as pd
from collections import defaultdict

def carregar_dados_csv(caminho_csv):
    # Lê o arquivo CSV sem cabeçalho
    df = pd.read_csv(caminho_csv, header=None)

    # Inicializa estruturas de dados
    prof_teoria = defaultdict(list)
    prof_pratica = defaultdict(list)
    monitores = defaultdict(list)
    disponibilidade = {}

    # Mapeamento de colunas para dias/horários
    dias_horarios = [
        ('Segunda', 'Teoria'), ('Segunda', 'Prática'),
        ('Terça', 'Teoria'), ('Terça', 'Prática'),
        ('Quarta', 'Teoria'), ('Quarta', 'Prática'),
        ('Quinta', 'Teoria'), ('Quinta', 'Prática')
    ]

    # Contadores para gerar IDs únicos
    contadores = {
        'P Teoria': defaultdict(int),
        'P Exerc': defaultdict(int),
        'Monitor': defaultdict(int)
    }

    for _, row in df.iterrows():
        disciplina = row[0].strip()
        tipo = row[1].strip()
        disp_raw = row[2:10]

        # Gera ID único
        prefixo = ''
        if tipo == 'P Teoria':
            prefixo = 'T' + disciplina[0].upper()
            contador = contadores[tipo][disciplina] + 1
```

```
            contadores[tipo][disciplina] = contador
            prof_id = f"{prefixo}{contador}"
            prof_teoria[disciplina].append(prof_id)
            if disciplina == "Português": # Prof de português teoria também na␣
 ↪prática
                prof_pratica[disciplina].append("P"+prof_id)
                monitores[disciplina].append("M"+prof_id)
        elif tipo == 'P Exerc':
            prefixo = 'P' + disciplina[0].upper()
            contador = contadores[tipo][disciplina] + 1
            contadores[tipo][disciplina] = contador
            prof_id = f"{prefixo}{contador}"
            prof_pratica[disciplina].append(prof_id)
        elif tipo == 'Monitor':
            prefixo = 'M' + disciplina[0].upper()
            contador = contadores[tipo][disciplina] + 1
            contadores[tipo][disciplina] = contador
            prof_id = f"{prefixo}{contador}"
            monitores[disciplina].append(prof_id)

        # Processa disponibilidade
        disponibilidade[prof_id] = []
        for i, val in enumerate(disp_raw):
            if pd.notna(val) and val == 1:
                dia, horario = dias_horarios[i]
                disponibilidade[prof_id].append((dia, horario))

    return {
        'prof_teoria': dict(prof_teoria),
        'prof_pratica': dict(prof_pratica),
        'monitores': dict(monitores),
        'disponibilidade': disponibilidade
    }

# Exemplo de uso:
if __name__ == "__main__":
    dados = carregar_dados_csv('dados_exato.csv')

    # Agora você pode usar esses dados no seu modelo de otimização
    print("Professores de Teoria:")
    print(dados['prof_teoria'])

    print("\nProfessores de Prática:")
    print(dados['prof_pratica'])

    print("\nMonitores:")
    print(dados['monitores'])
```

```
    print("\nExemplo de disponibilidade:")
    print(dados['disponibilidade']['TM1'])  # Primeiro professor de teoria de␣
 ↪Matemática

    # Usa os dados carregados
    prof_teoria = dados['prof_teoria']
    prof_pratica = dados['prof_pratica']
    monitores = dados['monitores']
    disponibilidade = dados['disponibilidade']
```

```
Professores de Teoria:
{'Matemática': ['TM1', 'TM2'], 'Português': ['TP1', 'TP2'], 'Química': ['TQ1',
'TQ2'], 'Física': ['TF1', 'TF2']}

Professores de Prática:
{'Matemática': ['PM1', 'PM2'], 'Português': ['PTP1', 'PTP2'], 'Química': ['PQ1',
'PQ2'], 'Física': ['PF1', 'PF2']}

Monitores:
{'Matemática': ['MM1', 'MM2', 'MM3', 'MM4', 'MM5', 'MM6', 'MM7', 'MM8', 'MM9',
'MM10', 'MM11', 'MM12', 'MM13'], 'Português': ['MTP1', 'MTP2', 'MP1', 'MP2'],
'Química': ['MQ1', 'MQ2', 'MQ3', 'MQ4', 'MQ5'], 'Física': ['MF1', 'MF2', 'MF3',
'MF4', 'MF5', 'MF6', 'MF7', 'MF8', 'MF9']}

Exemplo de disponibilidade:
[('Segunda', 'Prática'), ('Terça', 'Prática'), ('Quarta', 'Teoria'), ('Quinta',
'Teoria')]
```

[ ]:
```
# Professores disponíveis (aumentamos o número para atender a nova exigência)
prof_teoria = {
    'Física': ['TF1', 'TF2', 'TF3', 'TF4'],
    'Matemática': ['TM1', 'TM2', 'TM3', 'TM4'],
    'Português': ['TP1', 'TP2', 'TP3', 'TP4'],  # 4 professores para permitir 2␣
 ↪por turma
    'Química': ['TQ1', 'TQ2', 'TQ3', 'TQ4']
}

prof_pratica = {
    'Física': ['PF1', 'PF2', 'PF3', 'PF4'],
    'Matemática': ['PM1', 'PM2', 'PM3', 'PM4'],
    'Português': ['PP1', 'PP2', 'PP3', 'PP4'],
    'Química': ['PQ1', 'PQ2', 'PQ3', 'PQ4']
}

monitores = {
    'Física': ['MF1', 'MF2', 'MF3', 'MF4', 'MF5', 'MF6'],
```

```
    'Matemática': ['MM1', 'MM2', 'MM3', 'MM4', 'MM5', 'MM6'],
    'Português': ['MP1', 'MP2', 'MP3', 'MP4'],
    'Química': ['MQ1', 'MQ2', 'MQ3', 'MQ4', 'MQ5', 'MQ6']
}
```

[26]:
```
import cvxpy as cp

# 1. Definindo os dados do problema
disciplinas = ['Física', 'Matemática', 'Português', 'Química']
dias = ['Segunda', 'Terça', 'Quarta', 'Quinta']
horarios = ['Teoria', 'Prática']
turmas = ['Turma A', 'Turma B']

# 2. Criando as variáveis de decisão
variaveis = {}

# Alocação de disciplinas aos dias/turmas
for d in disciplinas:
    for t in turmas:
        for dia in dias:
            variaveis[f'disc_{d}_{t}_{dia}'] = cp.Variable(boolean=True)

# Alocação de professores de teoria
for d in disciplinas:
    for p in prof_teoria[d]:
        for t in turmas:
            for dia in dias:
                variaveis[f'teoria_{p}_{t}_{dia}'] = cp.Variable(boolean=True)

# Alocação de professores de prática
for d in disciplinas:
    for p in prof_pratica[d]:
        for t in turmas:
            for dia in dias:
                variaveis[f'pratica_{p}_{t}_{dia}'] = cp.Variable(boolean=True)

# Alocação de monitores
for d in disciplinas:
    for m in monitores[d]:
        for t in turmas:
            for dia in dias:
                variaveis[f'monitor_{m}_{t}_{dia}'] = cp.Variable(boolean=True)

# 3. Definindo as restrições
constraints = []

# Substitua as linhas problemáticas por:
```

```
# Para professores de teoria
for p in prof_teoria[d]:
    disp_p = disponibilidade[p]
    for t in turmas:
        for dia in dias:
            # Verifica se o professor está disponível para teoria neste dia
            disponivel = any((dia, 'Teoria') in disp_p or (dia, 'Prática') in␣
 ↪disp_p for dia, horario in disp_p)
            if not disponivel:
                constraints.append(variaveis[f'teoria_{p}_{t}_{dia}'] == 0)

# Para professores de prática
for p in prof_pratica[d]:
    disp_p = disponibilidade[p]
    for t in turmas:
        for dia in dias:
            # Verifica se o professor está disponível para prática neste dia
            disponivel = any((dia, 'Prática') in disp_p or (dia, 'Teoria') in␣
 ↪disp_p for dia, horario in disp_p)
            if not disponivel:
                constraints.append(variaveis[f'pratica_{p}_{t}_{dia}'] == 0)

# Para monitores
for m in monitores[d]:
    disp_m = disponibilidade[m]
    for t in turmas:
        for dia in dias:
            # Verifica se o monitor está disponível neste dia (qualquer horário)
            disponivel = any(dia == d for d, h in disp_m)
            if not disponivel:
                constraints.append(variaveis[f'monitor_{m}_{t}_{dia}'] == 0)

# Restrição 1: Cada turma tem uma disciplina por dia
for t in turmas:
    for dia in dias:
        constraints.append(
            sum(variaveis[f'disc_{d}_{t}_{dia}'] for d in disciplinas) == 1
        )

# Restrição 2: Cada turma tem todas as disciplinas (uma por semana)
for t in turmas:
    for d in disciplinas:
        constraints.append(
            sum(variaveis[f'disc_{d}_{t}_{dia}'] for dia in dias) == 1
        )
```

```
# Restrição 3: Professor de teoria alocado apenas quando a disciplina está␣
 ↪programada
for d in disciplinas:
    for p in prof_teoria[d]:
        for t in turmas:
            for dia in dias:
                constraints.append(
                    variaveis[f'teoria_{p}_{t}_{dia}'] <=␣
 ↪variaveis[f'disc_{d}_{t}_{dia}']
                )

# Restrição 4: DOIS professores de teoria por disciplina/turma/dia
for d in disciplinas:
    for t in turmas:
        for dia in dias:
            constraints.append(
                sum(variaveis[f'teoria_{p}_{t}_{dia}'] for p in prof_teoria[d])␣
 ↪==
                2 * variaveis[f'disc_{d}_{t}_{dia}']
            )

# Restrição 5: Professor de prática alocado apenas quando a disciplina está␣
 ↪programada
for d in disciplinas:
    for p in prof_pratica[d]:
        for t in turmas:
            for dia in dias:
                constraints.append(
                    variaveis[f'pratica_{p}_{t}_{dia}'] <=␣
 ↪variaveis[f'disc_{d}_{t}_{dia}']
                )

# Restrição 6: DOIS professores de prática por disciplina/turma/dia
for d in disciplinas:
    for t in turmas:
        for dia in dias:
            constraints.append(
                sum(variaveis[f'pratica_{p}_{t}_{dia}'] for p in␣
 ↪prof_pratica[d]) ==
                2 * variaveis[f'disc_{d}_{t}_{dia}']
            )

# Restrição 7: Monitores alocados apenas para aulas práticas
for d in disciplinas:
    for m in monitores[d]:
        for t in turmas:
```

```
            for dia in dias:
                constraints.append(
                    variaveis[f'monitor_{m}_{t}_{dia}'] <=␣
 ↪variaveis[f'disc_{d}_{t}_{dia}']
                )

# Restrição 8: Distribuição equilibrada de monitores entre turmas
for d in disciplinas:
    total_monitores = len(monitores[d])
    for t in turmas:
        total_para_turma = sum(variaveis[f'monitor_{m}_{t}_{dia}']
                               for m in monitores[d] for dia in dias)
        constraints.append(
            total_para_turma >= (total_monitores // len(turmas)) - 1
        )
        constraints.append(
            total_para_turma <= (total_monitores // len(turmas)) + 1
        )

# Restrição 9: Nas demais disciplinas, cada voluntário atua em no máximo uma␣
 ↪turma
for dia in dias:
    for d in [d for d in disciplinas if d != 'Português']:
        for p in prof_teoria[d]:
            constraints.append(
                sum(variaveis[f'teoria_{p}_{t}_{dia}'] for t in turmas) <= 1
            )
        for p in prof_pratica[d]:
            constraints.append(
                sum(variaveis[f'pratica_{p}_{t}_{dia}'] for t in turmas) <= 1
            )
        for m in monitores[d]:
            constraints.append(
                sum(variaveis[f'monitor_{m}_{t}_{dia}'] for t in turmas) <= 1
            )

# Restrição 10: Caso especial para Português (turmas unidas)
for dia in dias:
    constraints.append(
        variaveis[f'disc_Português_Turma A_{dia}'] ==
        variaveis[f'disc_Português_Turma B_{dia}']
    )

# 4. Função objetivo (minimizar sobrecarga de professores)
objective = cp.Minimize(
    sum(variaveis.values())  # Minimizar o total de alocações
)
```

```python
# 5. Resolver o problema
problem = cp.Problem(objective, constraints)
problem.solve(solver=cp.SCIP)

# 6. Exibir resultados
if problem.status == cp.OPTIMAL:
    print("Solução ótima encontrada!\n")

    # Mostrar grade horária por turma
    for t in turmas:
        print(f"\n{t}:")
        for dia in dias:
            for d in disciplinas:
                if variaveis[f'disc_{d}_{t}_{dia}'].value > 0.5:
                    print(f"\n{dia}: {d}")

                    professores_teoria = [p for p in prof_teoria[d]
                                          if variaveis[f'teoria_{p}_{t}_{dia}'].
 ↪value > 0.5]
                    print(f"  Teoria: {', '.join(professores_teoria)}")

                    professores_pratica = [p for p in prof_pratica[d]
                                           if variaveis[f'pratica_{p}_{t}_{dia}'].
 ↪value > 0.5]
                    print(f"  Prática: {', '.join(professores_pratica)}")

                    monits = [m for m in monitores[d]
                              if variaveis[f'monitor_{m}_{t}_{dia}'].value > 0.5]
                    print(f"  Monitores: {', '.join(monits)}")
else:
    print("Não foi encontrada uma solução ótima.")
    print("Status:", problem.status)
```

```
Solução ótima encontrada!


Turma A:

Segunda: Matemática
  Teoria: TM1, TM2
  Prática: PM1, PM2
  Monitores: MM2, MM5, MM7, MM10, MM13

Terça: Física
  Teoria: TF1, TF2
  Prática: PF1, PF2
```

```
  Monitores: MF2, MF5, MF8

Quarta: Português
  Teoria: TP1, TP2
  Prática: PTP1, PTP2
  Monitores: MTP1

Quinta: Química
  Teoria: TQ1, TQ2
  Prática: PQ1, PQ2
  Monitores: MQ4

Turma B:

Segunda: Química
  Teoria: TQ1, TQ2
  Prática: PQ1, PQ2
  Monitores: MQ5

Terça: Matemática
  Teoria: TM1, TM2
  Prática: PM1, PM2
  Monitores: MM1, MM2, MM4, MM12, MM13

Quarta: Português
  Teoria: TP1, TP2
  Prática: PTP1, PTP2
  Monitores: MTP1

Quinta: Física
  Teoria: TF1, TF2
  Prática: PF1, PF2
  Monitores: MF1, MF5, MF7
```

[ ]: