Horário Semanal de aulas

Grupo 05


```
Eduardo André Silva Cunha
Gonçalo Emanuel Ferreira Magalhães A100084
```


Problema

Pretende-se construir o horário semanal de aulas de uma turma.

1. Existe um conjunto de salas S classificadas em "grandes" e "pequenas".
2. O tempo do horário está organizado em "slots" de uma hora. O total do tempo disponível é 5 horas de manhã e 5 horas às tarde.
3. Existe um conjunto D de disciplinas. Cada disciplina tem um atributo $d \in \{1,2\}$ que classifica a duração de cada sessão (um ou dois "slots") , um atributo $a \in \{2,3\}$ que define o número de sessões semanais e um atributo $s \in \{0,1\}$ que diz se a sessão necessita de uma sala grande ou não.
4. Existe um conjunto $P$$de professores. Cada professor tem associado um conjunto h das disciplinas que está habilitado a lecionar.
5. O horário está organizado em sessões concorrentes onde cada sessão é definido por uma disciplina desce que salas e professores verifiquem as seguintes restrições.
   - Para cada disciplina todas as aulas decorrem na mesma sala e com o mesmo professor.
   - O número total de horas lecionadas por cada professor está num intervalo de $\pm 20\%$ do número médio de horas lecionadas pela totalidade dos professores.
   - Nenhuma sala pode ser ocupada simultaneamente por mais do que uma aula e nenhum professor pode lecionar simultaneamente mais do que uma aula.
   - Em cada disciplina, cada aula é lecionada por um professor habilitado para essa disciplina e ocorre numa sala de tamanho apropriado à disciplina.

   Use o package `ortools` para encontrar uma solução que verifique todas as restrições e maximize o número de partes de dia (manhãs ou tardes) que estão livres de qualquer aula.

```
!pip install ortools

Requirement already satisfied: ortools in c:\users\eduar\anaconda3\
envs\logica\lib\site-packages (9.7.2996)
Requirement already satisfied: absl-py>=0.13 in c:\users\eduar\
anaconda3\envs\logica\lib\site-packages (from ortools) (1.4.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\eduar\
anaconda3\envs\logica\lib\site-packages (from ortools) (1.26.0)
Requirement already satisfied: protobuf>=4.23.3 in c:\users\eduar\
anaconda3\envs\logica\lib\site-packages (from ortools) (4.24.3)
```

Import

Imports usados no código:

```
from ortools.linear_solver import pywraplp
```

Implementação

Definir os valores para os inputs relativos ao problema.

Começamos por definir o input do nosso problema. Iremos ter as variáveis $dias_da_semana$, $salas$, $periodos$, $disciplinas$ e $professores$, que representam o os dias da semana, as salas, os periodos, as disciplinas e os professores, respetivamente, assim como as variáveis implicitas no problema .

Adicionalmente, teremos 2 dicionários:

1) O primeiro dicionário, prof_dict, estabelecerá a correspondência entre cada Professor e as disciplinas que leciona. 2) O segundo dicionário, disp, estabelecerá a correspondência entre cada disciplina, o tempo de cada aula, o numero de dias por semana e se necessita de uma sala grande ou pequena.

```
# dias_da_semana = ["segunda", "terca", "quarta", "quinta", "sexta"]
dias_da_semana = [1, 2, 3, 4, 5]

# Salas - grandes ou pequenas
salas = [1,2,3,4,5,6]

# Periodos - 1h ou 2h, max 5h manha/tarde
#periodos = ["8-9","9-10","10-11","11-12","12-13","14-15","15-16","16-
17","17-18","18-19"]
periodos = [1,2,3,4,5,6,7,8,9,10]

# Disciplinas - duracao de 1h ou 2h, 2 ou 3 aulas semanais, e 0 ou 1
que define se e necessario uma sala grande ou nao
#disciplinas = ["PLC","LC","BD","PA","CC"]
disciplinas = [1,2,3,4,5]

# Professores - cada sore tem o conjunto das disciplinas que leciona
#professores = ["Mario","Abilio","Jota","Januario","Victor"]
professores = [1,2,3,4,5]


#[stores] : [disciplinas que leciona]
prof_dict = {1:[1,2], 2:[3,1], 3:[4,5], 4:[5,3], 5:[2,4]}

#[disciplinas] : (tempodeaula, diasporsemana, salasgrandeoupequena)
#disp_dict = {"PLC":(1,3,"grande"), "LC":(2,1,"pequena"), "BD":
```

```
(2,2,"pequena"), "PA":(2,2,"pequena"), "CC":(1,3,"grande")}
disp_dict = {1: (1, 3, 1), 2: (2, 1, 0), 3: (2, 2, 0), 4: (2, 2, 0),
5: (1, 3, 1)}


#[salas] : "grande"/"pequena" - 1 grande, 0 pequena
#salas_dict = {1:"grande", 2:"pequena", 3:"pequena", 4:"grande",
5:"grande", 6:"pequena"}
salas_dict = {1:1, 2:0, 3:0, 4:1, 5:1, 6:0}
```

Análise do Problema

Este é um problema de alocação pois temos uma relação entre compromissos e recursos. Como tal, vamos usar Programação Inteira para o modelar e resolver. Utilizando as variáveis I,J,K,S e P, podemos criar 2 matrizes de alocação com o seguinte significado:

$$X_{i,j,k,s} = ¿\, 1 \text{ se e só se disciplina } i \text{ é atribuída ao período } j \text{ no dia da semana } k \text{ e a sala } s$$

$$Y_{i,k,j,p} = ¿\, 1 \text{ se e só se disciplina } i \text{ no dia da semana } k\text{, no período } j \text{ e com o professor } p$$

Vejamos então as *limitações* e as *obrigações* do nosso problema:

Limitações

1. Para cada disciplina todas as aulas decorrem na mesma sala e com o mesmo professor.
2. O número total de horas lecionadas por cada professor está num intervalo de $\pm 20\,\%$ do número médio de horas lecionadas pela totalidade dos professores.
3. Nenhuma sala pode ser ocupada simultaneamente por mais do que uma aula e nenhum professor pode lecionar simultaneamente mais do que uma aula.
4. Em cada disciplina, cada aula é lecionada por um professor habilitado para essa disciplina e ocorre numa sala de tamanho apropriado à disciplina.

Obrigações

1. O número de aulas lecionadas em cada disciplina tem de ser o mesmo que o especificado em disp_dict
2. Apenas uma aula por periodo
3. Uma aula pode ter 1 ou 2 periodos conforme descirto em disp_dict

Na tentativa de resolver o problema usando o solver da pwralp, decidimos executar da seguinte forma.

Inicialização das matrizes de alocação

```
solver = pywraplp.Solver.CreateSolver('SCIP')

x = {}
y = {}

# Variáveis binárias x[i, j, k, s] representando se a disciplina i é
```

```
atribuída ao período j no dia da semana k e a sala s
x = {}
for i in disciplinas:
    for j in periodos:
        for k in dias_da_semana:
            for s in salas:
                x[i, j, k, s] = solver.IntVar(0, 1,
f'x_{i}_{j}_{k}_{s}')

# Variáveis inteiras y[i, k, j, p] representando a disciplina i no dia
da semana k, no período j e com o professor p
y = {}
for i in disciplinas:
    for k in dias_da_semana:
        for j in periodos:
            for p in professores:
                y[i, k, j, p] = solver.IntVar(0, 1,
f'y_{i}_{k}_{j}_{p}')
```

Restrições

1. Para cada disciplina todas as aulas decorrem na mesma sala e com o mesmo professor.

```
#Toda a aula na mesma sala
for i in disciplinas:
    for j in periodos:
        for k in dias_da_semana:
            solver.Add(sum(x[i, j, k, s] for s in salas) == 1)

#Toda a aula com o mesmo professor
for i in disciplinas:
    for k in dias_da_semana:
        for j in periodos:
            solver.Add(sum(y[i, k, j, p] for p in professores) == 1)
```

1. O número total de horas lecionadas por cada professor está num intervalo de $\pm 20\,\%$ do número médio de horas lecionadas pela totalidade dos professores.

```
# Calculo do número médio de horas lecionadas por todos os professores
total_hours = sum(y[i, k, j, p] for i in disciplinas for k in
dias_da_semana for j in periodos for p in professores)
average_hours = total_hours / len(professores)

# Garantir que nenhum professor passa do limite medio +- 20%
for p in professores:
    professor_total_hours = sum(y[i, k, j, p] for i in disciplinas for
k in dias_da_semana for j in periodos)
    solver.Add(professor_total_hours >= 0.8 * average_hours)
    solver.Add(professor_total_hours <= 1.2 * average_hours)
```

1. Nenhuma sala pode ser ocupada simultaneamente por mais do que uma aula e nenhum professor pode lecionar simultaneamente mais do que uma aula.

```
# sala so pra uma aula por periodo:
for k in dias_da_semana:
    for j in periodos:
        for s in salas:
            solver.Add(solver.Sum(x[i, j, k, s] for i in disciplinas)
<= 1)

#professor so pra uma disciplina por periodo:
for k in dias_da_semana:
    for j in periodos:
        for p in professores:
            solver.Add(solver.Sum(y[i, k, j, p] for i in disciplinas)
<= 1)
```

1. Em cada disciplina, cada aula é lecionada por um professor habilitado para essa disciplina e ocorre numa sala de tamanho apropriado à disciplina.

```
#Professor habilitado para lecionar a disciplina
for i in disciplinas:
    for k in dias_da_semana:
        for j in periodos:
            for p in professores:
                # Se a disciplina i requer o professor p (p está
habilitado para i),
                # então a variável y[i, k, j, p] pode ser 0 ou 1. Mas
caso não seja então tem de ser 0.
                if i in prof_dict[p]:
                    solver.Add(y[i, k, j, p] <= 1)
                else:
                    solver.Add(y[i, k, j, p] == 0)

#Sala apropriada para a disciplina
for i in disciplinas:
    for k in dias_da_semana:
        for j in periodos:
            # Se a disciplina i requer uma sala grande (sala_dict[i]
== 1),
            # então a variável x[i, j, k, s] deve ser igual a 1
somente se a sala s for grande.
            if disp_dict[i][2] == 1:
                for s in salas:
                    if salas_dict[s] == 0:
                        solver.Add(x[i, j, k, s] == 0)
                    # Senão pode ser maior que 0

            # Se a disciplina i requer uma sala pequena (sala_dict[i]
== 0),
```

```
            # então a variável x[i, j, k, s] deve ser igual a 1
somente se a sala s for pequena.
            else:
                for s in salas:
                    if salas_dict[s] == 1:
                        solver.Add(x[i, j, k, s] == 0)
                    # Senão pode ser maior que 0
```

Obrigações:

1. O número de aulas lecionadas em cada disciplina tem de ser o mesmo que o especificado em disp_dict

```
for i in disciplinas:
    total_aulas_disciplina = disp_dict[i][1]
    solver.Add(solver.Sum(x[i, j, k, s] for j in periodos for k in
dias_da_semana for s in salas) == total_aulas_disciplina)
```

1. Apenas uma aula por periodo

```
for k in dias_da_semana:
    for j in periodos:
        solver.Add(solver.Sum(x[i, j, k, s] for i in disciplinas for s
in salas) <= 1)
```

1. Uma aula pode ter 1 ou 2 periodos conforme descirto em disp_dict

```
for i in disciplinas:
    if disp_dict[i][0] == 2:
        for k in dias_da_semana:
            for j in range(1, len(periodos)):
                for s in salas:
                    if x[i, j, k, s] == 1:
                        solver.Add(x[i, j+1, k, s] == 1)
```

Objetivos

O objetivo deste código para a criação de um horário, é maximizar o número de partes de dia (manhãs ou tardes) que estão livres de qualquer aula.

```
manhas = [1, 2, 3, 4, 5]  # Periodos da manha
tardes = [6, 7, 8, 9, 10] # Periodos da tarde

# Variáveis binárias para representar se um período de manhã ou tarde
é livre
manha_livre = {}
tarde_livre = {}

for k in dias_da_semana:
    manha_livre[k] = solver.IntVar(0, 1, f'manha_livre_{k}') # 1
```

```
livre, 0 ocupada
    tarde_livre[k] = solver.IntVar(0, 1, f'tarde_livre_{k}') # 1
livre, 0 ocupada

# Caso seja uma manha livre ou tarde livre então não pode haver aulas
nesse periodo
for k in dias_da_semana:
    manha_livre_expr = solver.Sum(1 - x[i, j, k, s] for i in
disciplinas for j in manhas for s in salas)
    solver.Add(manha_livre[k] == manha_livre_expr)

    tarde_livre_expr = solver.Sum(1 - x[i, j, k, s] for i in
disciplinas for j in tardes for s in salas)
    solver.Add(tarde_livre[k] == tarde_livre_expr)

# Objetivo: Maximizar o número de manhãs ou tardes livres
count_manhas_livres = solver.Sum(manha_livre[k] for k in
dias_da_semana)
count_tardes_livres = solver.Sum(tarde_livre[k] for k in
dias_da_semana)
solver.Maximize(count_manhas_livres + count_tardes_livres)
```

Print do resultado

Impressão do resultado final

```
status = solver.Solve()

if status == pywraplp.Solver.OPTIMAL: # Caso haja resposta ótima
    for k in dias_da_semana:
        for i in disciplinas:
            for j in periodos:
                for sala in salas:
                    for professor in professores:
                        # Verifica se a variável x[i, j, k, sala] é
igual a 1 (aula está alocada na sala)
                        if x[i, j, k, sala].solution_value() == 1:
                            # Verifica se a variável y[i, k, j,
professor] é igual a 1 (professor dá essa aula)
                            if y[i, k, j, professor].solution_value()
== 1:
                                print(f'Disciplina: {i}, Professor:
{professor}, Sala: {sala}, Dia: {k}, Período: {j}')

else: # Caso não
    print('Não foi encontrada uma solução ótima.')
```

```
Não foi encontrada uma solução ótima.
```

Breve Conclusão

Com base nos nossos diversos testes, suspeitamos que as nossas "funções" obrigações e a "função" objetivo possam estar a contribuir para o insucesso do código.

Porém, embora não tenhamos alcançado um resultado concreto, acreditamos que abordamos corretamente o tema da programação linear e acreditamos que estivemos perto da resolução do problema.