Grupo 24

Gabriel Antunes a101101 Guilherme Pinho a105533

Pretende-se construir um horário semanal para o plano de reuniões de projeto de uma "StartUp" de acordo com as seguintes condições:

1. Cada reunião ocupa uma sala (enumeradas 1...S) durante um "slot" (tempo,dia). Assume-se os dias enumerados 1...D e, em cada dia, os tempos enumerados 1...T.

2. Cada reunião tem associado um projeto (enumerados 1...P) e um conjunto de participantes. Os diferentes colaboradores são enumerados 1...C.

3. Cada projeto tem associado um conjunto de colaboradores, dos quais um é o líder. Cada projeto realiza um dado número de reuniões semanais. São "inputs" do problema o conjunto de colaboradores de cada projeto, o seu líder e o número de reuniões semanais.

4. O líder do projeto participa em todas as reuniões do seu projeto; os restantes colaboradores podem ou não participar consoante a sua disponibilidade, num mínimo ("quorum") de 50% do total de colaboradores do projeto. A disponibilidade de cada participante, incluindo o lider, é um conjunto de "slots" ("inputs" do problema).

São os "input" do problema

1. Os parâmetros S, T, P, C
2. O conjunto de colaboradores de cada projeto, o seu líder e o número mínimo de reuniões semanais
3. A disponibilidade de cada participante, incluindo o lider. Essa disponibilidade é um conjunto de "slots" representada numa matriz booleana de acessibilidade com uma linha por cada participante 1...C e uma coluna por "slot" 1...T

Comecemos por definir valores base para que vão servir de inputs para o exercicio:

```
from ortools.linear_solver import pywraplp

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

S = 2
T = (5 , 3) # 5 dias por semana, 3 turnos por dia
C = 5
P = 2

projects = { 1 : { "colaboradores" : [1,3,5], "lider" : 1,
"num_reunioes" : 2},
             2 : { "colaboradores" : [2, 4], "lider" : 2,
"num_reunioes" : 2}}
```

```
disponibilidade = { 1 : [(2,1), (2,2), (2,3), (3,1), (5,2), (5,3)],
                    2 : [(1,2), (2,2), (3,2), (4,2), (5,2)],
                    3 : [(1,1), (2,2), (3,3), (4,1), (5,2)],
                    4 : [(1,1), (1,2), (1,3), (2,1), (2,2), (2,3)],
                    5 : [(1,3), (2,3), (3,3), (4,3), (5,2), (5,3)]  }
```

Vamos criar uma função que vai construir o horário que recebe as Salas, os Turnos e os Dias Este horário vai consistir num dicionário em que as chaves são tuplos no formato "(Turno, Dia, Sala)" e a cada uma destas chaves está associado um valor que é uma lista com as variáveis Bool que representam os projetos possíveis e mais tarde irão ter o propósito de validar se é possivel fazer uma reunião naquela sala para aquele projeto

```
# Função que vai construir o horário com os inputs dados, que vai
associar cada Dia, Turno, Sala a uma lista com os projetos possíveis
# makeSchedule
sch = {}

for day in range(1, T[1] + 1):
    for turn in range(1, T[0] + 1):
        for room in range(S):
            sch[(day, turn, room)] = [solver.BoolVar(f"({day}-{turn}-
{room}: {proj})") for proj in range(1, P + 1)]
```

Depois de ter o horário e as variáveis criadas vamos acrescentar a condição que dita que o lider de um projeto têm de comparecer a todas as reuniões desse mesmo projeto

```
# Função para acrescentar a condição de que o líder tem de comparecer
a todas as reuniões do projeto
# validateLeader

for proj, data in projects.items():
    leader = data['lider']
    for day in range(1, T[1] + 1):
        for turn in range(1, T[0] + 1):
            for room in range(S):
                if (day, turn) in disponibilidade[leader]:
                    solver.Add(sch[(day, turn, room)][proj - 1] <= 1)
                else:
                    solver.Add(sch[(day, turn, room)][proj - 1] == 0)
```

Uma vez definida a condição que certifica que o líder do projeto está em todas as reuniões, é necessário verificar que mais de metade dos restantes colaboradores tem disponibilidade nessas reuniões

```
# Função para acrescentar a condição de que cada reunião deve ter pelo
menos metade dos colaboradores presentes
# validateQuorum
```

```
for proj, data in projects.items():

    num_colabs = len(data["colaboradores"])
    quorum = (num_colabs + 1) // 2

    for day in range(1, T[1] + 1):
        for turn in range(1, T[0] + 1):
            for room in range(S):

                solver.Add(sum(1 for colab in data["colaboradores"] if
(day, turn) in disponibilidade[colab]) >= quorum * sch[(day, turn,
room)][proj - 1])
```

De seguida vamos adicionar a restrição que verifica se os projetos foram alocados o número de vezes suficientes para a sua realização

```
# Função que acrescenta a restrição do número de reuniões
for proj in projects:
    solver.Add(sum(sch[(day, turn, room)][proj - 1] for day in
range(1, T[1] + 1) for turn in range(1, T[0] + 1) for room in
range(S)) == projects[proj]["num_reunioes"])
```

Para terminar a ultima restrição irá verificar se existem mais que uma reunião alocadas num mesmo horário

```
# Função que verifica se não existe mais que 1 projeto alocado num
mesmo horário
for day in range(1, T[1] + 1):
    for turn in range(1, T[0] + 1):
        for room in range(S):
            solver.Add(sum(sch[(day, turn, room)][proj - 1] for proj
in projects) <= 1)
```

Após todas as condições terem sido impostas basta resolver o solver e imprimir os resultados

```
# Resolver o modelo
status = solver.Solve()

# Exibir resultados
if status == pywraplp.Solver.OPTIMAL:
    for proj in projects:
        for day in range(1, T[1] + 1):
            for turn in range(1, T[0] + 1):
                for room in range(S):
                    if sch[(day, turn, room)][proj -
1].solution_value() == 1:
                        print(f"Projeto {proj} alocado para o dia
{day}, no turno {turn} na sala {room}.")
elif status == pywraplp.Solver.INFEASIBLE:
```

```
    print("Não foi possível encontrar uma solução viável.")
else:
    print(f"Solução encontrada com status: {status}.")
```

```
Projeto 1 alocado para o dia 2, no turno 3 na sala 0.
Projeto 1 alocado para o dia 2, no turno 3 na sala 1.
Projeto 2 alocado para o dia 1, no turno 2 na sala 0.
Projeto 2 alocado para o dia 2, no turno 2 na sala 1.
```

Consideremos alguns exemplos de mais casos e os outputs esperados.

EX.1:

```
S = 4  # 4 salas
T = (5, 3)  # 5 dias por semana, 3 turnos por dia
C = 12  # 12 colaboradores
P = 5  # 5 projetos

projects = {
    1: {"colaboradores": [1, 2, 3], "lider": 1, "num_reunioes": 2},
    2: {"colaboradores": [4, 5, 6], "lider": 4, "num_reunioes": 2},
    3: {"colaboradores": [7, 8, 9], "lider": 7, "num_reunioes": 2},
    4: {"colaboradores": [10, 11], "lider": 10, "num_reunioes": 1},
    5: {"colaboradores": [12, 3], "lider": 12, "num_reunioes": 1}
}

disponibilidade = {
    1: [(1, 1), (2, 2), (3, 1), (4, 2)],  # Líder do projeto 1
    2: [(1, 1), (3, 1), (4, 3)],          # Colaborador no projeto 1
    3: [(1, 2), (3, 1), (4, 1)],          # Colaborador no projeto 1 e
projeto 5
    4: [(2, 1), (2, 2), (3, 2)],          # Líder do projeto 2
    5: [(2, 1), (4, 1), (5, 2)],          # Colaborador no projeto 2
    6: [(3, 2), (5, 1), (5, 3)],          # Colaborador no projeto 2
    7: [(1, 3), (2, 3), (4, 2)],          # Líder do projeto 3
    8: [(2, 2), (3, 1), (4, 2)],          # Colaborador no projeto 3
    9: [(1, 3), (3, 2), (5, 2)],          # Colaborador no projeto 3
    10: [(1, 2), (3, 3), (5, 1)],         # Líder do projeto 4
    11: [(2, 3), (3, 3), (5, 3)],         # Colaborador no projeto 4
    12: [(2, 1), (3, 2), (5, 1)]          # Líder do projeto 5
}
```

O output esperado para este exemplo é: Projeto 1 alocado para o dia 1, no turno 1 na sala 0. Projeto 1 alocado para o dia 3, no turno 1 na sala 3. Projeto 2 alocado para o dia 2, no turno 1 na sala 1. Projeto 2 alocado para o dia 3, no turno 2 na sala 3. Projeto 3 alocado para o dia 1, no turno 3 na sala 0. Projeto 3 alocado para o dia 1, no turno 3 na sala 2. Projeto 4 alocado para o dia 1, no turno 2 na sala 0. Projeto 5 alocado para o dia 2, no turno 1 na sala 0. Ex.2:

```
S = 3  # 3 salas
T = (4, 4)  # 4 dias por semana, 4 turnos por dia
C = 10  # 10 colaboradores
P = 4  # 4 projetos

projects = {
    1: {"colaboradores": [1, 2, 3], "lider": 1, "num_reunioes": 3},
    2: {"colaboradores": [4, 5], "lider": 4, "num_reunioes": 2},
    3: {"colaboradores": [6, 7], "lider": 6, "num_reunioes": 2},
    4: {"colaboradores": [8, 9, 10], "lider": 8, "num_reunioes": 2}
}

disponibilidade = {
    1: [(1, 1), (2, 1)],                # Líder do projeto 1
    2: [(2, 3), (3, 2)],                # Colaborador no projeto 1
    3: [(4, 1)],                        # Colaborador no projeto 1
    4: [(1, 4), (2, 2)],                # Líder do projeto 2
    5: [(3, 1), (4, 4)],                # Colaborador no projeto 2
    6: [(1, 2), (4, 3)],                # Líder do projeto 3
    7: [(2, 4)],                        # Colaborador no projeto 3
    8: [(1, 3), (3, 3)],                # Líder do projeto 4
    9: [(1, 2), (3, 4)],                # Colaborador no projeto 4
    10: [(4, 2), (4, 4)]                # Colaborador no projeto 4
}
```

O output esperado para este exemplo é: Não foi possível encontrar uma solução viável.

Ex.3:

```
S = 5  # 5 salas
T = (7, 5)  # 7 dias por semana, 5 turnos por dia
C = 20  # 20 colaboradores
P = 6  # 6 projetos

projects = {
    1: {"colaboradores": [1, 2, 3, 4], "lider": 1, "num_reunioes": 3},
    2: {"colaboradores": [5, 6, 7, 8], "lider": 5, "num_reunioes": 3},
    3: {"colaboradores": [9, 10, 11, 12], "lider": 9, "num_reunioes":
2},
    4: {"colaboradores": [13, 14, 15], "lider": 13, "num_reunioes":
2},
    5: {"colaboradores": [16, 17], "lider": 16, "num_reunioes": 2},
    6: {"colaboradores": [18, 19, 20], "lider": 18, "num_reunioes": 1}
}

disponibilidade = {
    1: [(1, 1), (2, 1), (3, 2), (4, 3), (5, 1)],        # Líder do
projeto 1
    2: [(1, 1), (3, 2), (4, 1), (5, 1)],                # Colaborador
no projeto 1
```

```
    3: [(1, 1), (3, 3), (4, 3), (6, 2)],                # Colaborador
no projeto 1
    4: [(1, 1), (4, 2), (5, 1)],                        # Colaborador
no projeto 1
    5: [(1, 2), (3, 1), (4, 1), (6, 3)],                # Líder do
projeto 2
    6: [(1, 2), (2, 4), (3, 1), (5, 2)],                # Colaborador
no projeto 2
    7: [(1, 2), (4, 3), (5, 4), (7, 1)],                # Colaborador
no projeto 2
    8: [(1, 1), (3, 3), (5, 2), (6, 4)],                # Colaborador
no projeto 2
    9: [(1, 2), (3, 4), (4, 1), (6, 2)],                # Líder do
projeto 3
    10: [(1, 2), (2, 3), (4, 4), (5, 2)],               # Colaborador
no projeto 3
    11: [(1, 1), (3, 2), (4, 3), (6, 1)],               # Colaborador
no projeto 3
    12: [(1, 3), (3, 1), (4, 4), (6, 3)],               # Colaborador
no projeto 3
    13: [(2, 1), (3, 2), (4, 1), (5, 3)],               # Líder do
projeto 4
    14: [(2, 2), (3, 3), (4, 2), (6, 1)],               # Colaborador
no projeto 4
    15: [(2, 1), (3, 4), (4, 1), (7, 2)],               # Colaborador
no projeto 4
    16: [(1, 3), (4, 3), (5, 4), (6, 2)],               # Líder do
projeto 5
    17: [(1, 3), (2, 4), (3, 1), (5, 2)],               # Colaborador
no projeto 5
    18: [(1, 1), (2, 1), (3, 2), (4, 4)],               # Líder do
projeto 6
    19: [(1, 1), (3, 3), (5, 3), (7, 2)],               # Colaborador
no projeto 6
    20: [(1, 2), (3, 2), (4, 1), (5, 2)]                # Colaborador
no projeto 6
}
```

O output esperado para este exemplo é: Projeto 1 alocado para o dia 4, no turno 3 na sala 1. Projeto 1 alocado para o dia 4, no turno 3 na sala 2. Projeto 1 alocado para o dia 4, no turno 3 na sala 3. Projeto 2 alocado para o dia 3, no turno 1 na sala 1. Projeto 2 alocado para o dia 3, no turno 1 na sala 3. Projeto 2 alocado para o dia 3, no turno 1 na sala 4. Projeto 3 alocado para o dia 1, no turno 2 na sala 0. Projeto 3 alocado para o dia 1, no turno 2 na sala 3. Projeto 4 alocado para o dia 2, no turno 1 na sala 1. Projeto 4 alocado para o dia 4, no turno 1 na sala 4. Projeto 5 alocado para o dia 4, no turno 3 na sala 0. Projeto 5 alocado para o dia 4, no turno 3 na sala 4. Projeto 6 alocado para o dia 1, no turno 1 na sala 0.