Trabalho Prático 1

Grupo 22

Alexis Correia - A102495 João Fonseca - A102512

Exercício 1

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 e 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).

Resolução

Dado que isto é um problema de Programação Inteira ou `Programação Linear`, utilizaremos a ferramenta `pywraplp` do ORTools. Agora, primeira coisa que precisamos fazer, após importar as bibliotecas necessárias, é definir algumas constantes. Além disso, optamos por criar "inputs" aleatórios com auxílio do random para testar o código.

```
# Inicialização

from ortools.linear_solver import pywraplp
import random

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

# Constantes

S, D, T = 5, 5, 8 # num de salas, dias e tempos (por dia)
P, C, cp = 5, 30, 5 # num de projetos, de colaboradores totais e num
```

```
de colaboradores por projeto

# Inputs aleatórios

slots = [(d, t) for d in range(D) for t in range(T)] # Slots, formados
por dia e tempo
aux = set(range(C)) # Dicionário com os colaboradores (ainda) não
atribuídos a nenhum projeto
HC = [random.sample(slots, 20) for _ in range(C)] # Horário com a
disponibilidade de cada Colaborador

Proj = [] # Lista dos projetos, com tuplos formados por: (num de
reuniões, líder, lista de colaboradores)
for _ in range(P):
  cs = random.sample(aux, cp) # Seleciona colaboradores aleatórios
para o projeto
  r = random.randint(1, 5) # Decide o num de reunioes de cada projeto
(de 1 a 5)
  l = random.choice(cs) # Escolhe o líder de cada projeto dentre os
colaboradores do projeto
  Proj.append((r,l, cs)) # Adiciona a lista dos projetos
  aux = aux - set(cs)
    # retira os colaboradores do dicionário, garantindo que cada
colaborador trabalha em apenas um projeto

for i, (r, l, cs) in enumerate(Proj): #Imprime a lsita Proj
  print(f"Projeto {i+1}\nNum de reuniões: {r}  Líder:{l}
Colaboradores: {cs}")
print()
for c in range(30): #Imprime o horário (disponível) de todos os
colaboradores
  HC[c].sort()
  print(f"Colaborador {c+1}: {HC[c]}")
```

Uma vez concluído a geração automática do input, podemos começar a trabalhar com a ferramenta `pywraplp`. O primeiro passo é criar a matriz multi-dimensional.

```
# Criação da matriz multi-dimensional
X = {}
for s in range(S):
    for d in range(D):
        for t in range(T):
            for p in range(P):
                for c in range(C):
                    X[s,d,t,p,c] = solver.BoolVar(f"x[{s},{d},{t},{p},
{c}]")
```

Em seguida, podemos adicionar as restrições que possibilitam a resolução do problema:

1. Cada projeto tem associado um conjunto de colaboradores, dos quais um é o líder. Além disso, o líder do projeto participa de todas as reuniões do seu projeto. Ou seja, tomando L_p como o líder dum projeto p:

$$\forall\, p \in P, \sum_{s \in S, d \in D, t \in T} X_{s,d,t,p,L_p} = R$$

2. Para além do líder, os outros colaboradores podem ou não participar consoante a sua disponibilidade. No estanto, é preciso um mínimo ("quorum") de 50% do total de colaboradores do projeto na reunião.

$$\forall\, p \in P, s \in S, d \in D, t \in T . \sum_{c \in C} X_{s,d,t,p,c} \geq C/2$$

   Do enunciado do problema também podemos tirar que os colaboradores só podem participar das reuniões do projeto em que participam e nos slots compatíveis com suas disponibilidades.

3. $\forall\, s \in S, d \in D, t \in T, p \in P, c \in C$ se o colaborador c não está disponível no slot (d,t) então $X[s,d,t,p,c]=0$

4. $\forall\, s \in S, d \in D, t \in T, p \in P, c \in C$ se o colaborador c não for colaborador no projeto p então $X[s,d,t,p,c]=0$ Por fim, podemos afirmar que cada sala só comporta uma reunião. Ou seja, em cada slot, só pode haver, no máximo, a reunião de um projeto acontecendo numa sala.

5. Primeiro, basta garantir que o líder do projeto (c_L) está em, no máximo, uma sala em um determinado slot. Em outras palavras, garantimos que não aconteça duas reuniões de um mesmo projeto em duas salas diferentes ao mesmo tempo.

$$\forall\, s \in S, d \in D, t \in T . \sum_{p \in P} X_{s,d,t,p,c_L} \leq 1$$

6. E finalmente, obrigamos que cada colaborador só pode estar em, no máximo, uma sala a cada slot.

$$\forall\, d \in D, t \in T, p \in P, c \in C_p . \sum_{s \in S} X_{s,d,t,p,c} \leq 1$$

```
#Restrições:

# R1
for p in range(P):
  solver.Add(sum(X[s, d, t, p, Proj[p][1]] for s in range(S) for d in range(D) for t in range(T)) == Proj[p][0])
# Relembrando que Proj[p][0] é o número de reuniões atribuída a cada projeto p

# R2
for s in range(S):
```

```
  for d in range(D):
    for t in range(T):
      for p in range(P):
        solver.Add(sum(X[s, d, t, p, c] for c in Proj[p][2]) >=
cp*0.5*X[s, d, t, p, Proj[p][1]])
# Relembrando que Proj[p][1] é o líder do projeto p

# R3
for s in range(S):
  for d in range(D):
    for t in range(T):
      for p in range(P):
        for c in range(C):
          if (d, t) not in HC[c]:
            solver.Add(X[s, d, t, p, c] == 0)

# R4
for s in range(S):
  for d in range(D):
    for t in range(T):
      for p in range(P):
        for c in range(C):
          if c not in Proj[p][2]: # Substituímos C por Proj[p][2] com
o intuíto de diminuir o número de iterações.
            solver.Add(X[s, d, t, p, c] == 0)

# R5
for s in range(S):
  for d in range(D):
    for t in range(T):
      solver.Add(sum(X[s, d, t, p, Proj[p][1]] for p in range(P)) <=
1)
      # Relembrando que Proj[p][1] é o líder do projeto p

# R6
for d in range(D):
  for t in range(T):
    for p in range(P):
      for c in Proj[p][2]: # Substituímos C por Proj[p][2] com o
intuíto de diminuir o número de iterações.
        solver.Add(sum(X[s, d, t, p, c]  for s in range(S)) <= 1)
```

Por fim, com as restrições devidamente adicionadas ao solver, podemos resolver o problema.

```
# Fazer o solve - Resolução

status = solver.Solve()

if status == pywraplp.Solver.OPTIMAL:
    for d in range(D):
```

```
        print("Dia ", d+1)
        reuniao = []
        for s in range(S):
            print("S", s+1, end="     ")
        print()
        for t in range(T):
            for s in range(S):
                print("|T", t+1, end='| ')
                h = 0
                for p in range(P):
                    if round(X[s, d, t, p, Proj[p]
[1]].solution_value()) == 1:
                        reuniao.append([c for c in range(C) if
round(X[s, d, t, p, c].solution_value())])
                        print(p+1, end=" ")
                        h = 1
                if h == 0:
                    print("-", end = " ")
            print()
        print(reuniao)
else:
    print("UNSAT")
```