Trabalho 1 - Horário

17 de outubro de 2022

André Oliveira Barbosa - A91684 Francisco Antonio Borges Paulino - A91666

Caso de Estudo

Pretende-se construir um horário semanal para o plano de reuniões de projeto de uma “StartUp”. O horário tem S salas que serão ocupadas durante um slot T. Cada reunião terá associado um projeto P, e C colaboradores.

Condições:

As reuniões reuniões funcionam de acordo com as seguintes regras:

```
1.  Cada reunião ocupa uma sala (enumeradas 1...S,) durante um “slot”  1..T, (hor
a,dia).

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

3. Cada projeto tem associado um conjunto de colaboradores, dos quais um  é o líde
r. Cada projeto realiza um dado 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.
```

Análise do Problema:

Deparamo-nos assim com um problema de alocação. Pretende-se alocar colaboradores a salas de reunião, associadas a um projeto, ao longo da semana, em slots.

Existem S salas, que podemos identificar por um índice s ∈ [0..S−1] , às quais podemos atribuir um projeto P que decorre num dado slot T, por um duplo (p,t)∈[0..P−1]×[0..T−1].

Assim, vamos usar uma família x_{s,p,t} de variáveis binárias, com a seguinte semântica:

$$x_{s,p,t} == 1 \quad \text{se e só se} \quad \text{o projeto } p \text{ for alocado à sala } s\text{, no slot } t$$

Existem C colaboradores, que podemos identificar por um índice c ∈ [0..C−1] , às quais podemos atribuir um projeto P que decorre num dado slot T, por um duplo (p,t)∈[0..P−1]×[0..T−1].

Assim, vamos usar uma família y_{c,p,t} de variáveis binárias, com a seguinte semântica:

$$y_{c,p,t} == 1 \quad \text{se e só se} \quad \text{o colaborador } c \text{ for alocado ao projeto } p\text{, no slot } t$$

Variáveis:

```
S - Sala
T - Slot (hora, dia)
P - Projeto
C - Colaboradores
```

Variáveis auxiliares:

x_{s,p,t} - representa o projeto p alocado à sala s, no slot t
y_{c,p,t} - representa o colaborador c alocado ao projeto p, no slot t

Inicialização

Para a resolução deste exercício utilizamos a biblioteca [OR-Tools (https://developers.google.com/optimization)](https://developers.google.com/optimization) que criou uma interface para o SCIP. Esta biblioteca foi instalada com o commando pip install ortools.

In [180]:

```
!pip install ortools
```

```
Requirement already satisfied: ortools in c:\users\andre\anaconda3\envs\logi
ca\lib\site-packages (9.4.1874)
Requirement already satisfied: numpy>=1.13.3 in c:\users\andre\anaconda3\env
s\logica\lib\site-packages (from ortools) (1.23.3)
Requirement already satisfied: protobuf>=3.19.4 in c:\users\andre\anaconda3
\envs\logica\lib\site-packages (from ortools) (4.21.6)
Requirement already satisfied: absl-py>=0.13 in c:\users\andre\anaconda3\env
s\logica\lib\site-packages (from ortools) (1.2.0)
```

Implementação

Começamos por importar a biblioteca de programação linear do OR-Tools e criar uma instância do solver.

Depois inicializamos o solver horario e definir os valores para as constantes P , T, P, C .

In [181]:

```
from ortools.linear_solver import pywraplp
import random

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

In [182]:

```
### Inputs do problema

#Exemplo 1
S, P, T, C = 2, 3, 10, 4
# Número de projeto: (Líder de projeto, Número de reuniões semanais, Lista de colaboradores
projetos = {
    1: (1, 4, [1,2]),
    2: (2, 3, [1]),
    3: (3, 5, [2])
}
# Número do colaborador: Lista de slots
colaboradores = {
1: [1,2,3,4,5,6,7,8,9,10],
2: [1,2,3,4,5,6,7,8,9,10],
3: [1,2,3,4,5,6,7,8,9,10],
4: [1,2,3,4,5,6,7,8,9,10]

}
'''
#Exemplo 2
### Inputs do problema
S, P, T, C = 2, 3, 12, 4
# Número de projeto: (Líder de projeto, Número de reuniões semanais, Lista de colaboradores
projetos = {
1: (1, 10, [1,2]),
2: (2, 12,[1]),
3: (3, 10,[2])
}
# Número do colaborador: Lista de slots
colaboradores = {
1: [1,2,3,4,5,6,7,8,9,10,11,12],
2: [1,2,3,4,5,6,7,8,9,10,11,12],
3: [1,2,3,4,5,6,7,8,9,10,11,12]
}
print(colaboradores)
'''
```

Out[182]:

```
'\n#Exemplo 2\n### Inputs do problema\nS, P, T, C = 2, 3, 12, 4\n# Número de
projeto: (Líder de projeto, Número de reuniões semanais, Lista de colaborado
res)\nprojetos = {\n1: (1, 10, [1,2]),\n2: (2, 12,[1]),\n3: (3, 10,[2])\n}\n
# Número do colaborador: Lista de slots\ncolaboradores = {\n1: [1,2,3,4,5,6,
7,8,9,10,11,12],\n2: [1,2,3,4,5,6,7,8,9,10,11,12],\n3: [1,2,3,4,5,6,7,8,9,1
0,11,12]\n}\nprint(colaboradores)\n'
```

Declaração das matrizes de alocação

In [183]:

```
x={}

for s in range(S):
    x[s] = {}
    for p in range(P):
        x[s][p] = {}
        for t in range(T):
            x[s] [p] [t] = horario.BoolVar(f'X[{s}, {p}, {t}]')


def X(s,p,t):
    return x[s][p][t]
#print(colaboradores)
```

In [184]:

```
y={}

for c in range(C):
    y[c] = {}
    for p in range(P):
        y[c][p] = {}
        for t in range(T):
            y[c] [p] [t] = horario.BoolVar(f'Y[{c}, {p}, {t}]')


def Y(c,p,t):
    return y[c][p][t]
#print(colaboradores)
```

In [185]:

```
z={}

for t in range(T):
    z[t] = {}
    for p in range(P):
        z[t][p]= horario.BoolVar(f'Z[{t}, {p}]')

def Z(t,p):
    return z[t][p]
```

Modelação das restrições e introdução do solver

Passamos agora à modelação das restrições e à sua introdução no solver. Para tal, iremos analizar as condições e subdividi-las de forma a facilitar a criação de uma expressão lógica, bem como a sua interpretação.

A restrição

1. Cada reunião ocupa uma sala (enumeradas 1...S,) durante um “slot” 1..T, (hora,dia).

$$\forall_{t<T} . \forall_{s<S} . \sum_{p<P} x_{s,p,t} \leq 1$$

In [186]:

```
for t in range(T):
    for s in range(S):
        horario.Add(sum([X(s,p,t) for p in range(P)]) <= 1)
#print(colaboradores)
```

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

$$\forall_{p<P} . \forall_{t<T} . \sum_{c<C} y_{c,p,t} \leq 1$$

In [187]:

```
for p in range(P):
    for t in range(T):
        horario.Add(sum([Y(c,p,t) for c in range(C)]) <= 1)

#print(colaboradores)
```

3.Cada projeto realiza um dado número de reuniões semanais. ($R_p = projetos[p][1]$)

$$\forall_{p \leq P} \sum_{s \leq S \, t \leq T} x_{s,p,t} = R_p$$

In [188]:

```
for p in range(P):
    reunioes = projetos[p+1][1]
    horario.Add(sum([X(s,p,t) for s in range(S) for t in range(T)]) == reunioes)

#print(colaboradores)
```

4. O líder do projeto participa em todas as reuniões do seu projeto;

$$\forall_{p<P} . \forall_{t<T} . \sum_{s<S} x_{s,p,t} = y_{lider,t,p}$$

In [189]:

```
for p in range(P):
    for t in range(T):
        lider= projetos[p+1][0]
        #print(lider)
        horario.Add(sum([X(s,p,t) for s in range(S)]) == y[lider][p][t])

#print(colaboradores)
```

    5. Os restantes colaboradores podem ou não participar consoante a sua disponibilid
    ade, num mínimo (“quorum”) de 50% do total de colaboradores do projeto.

$$\forall_{p<P} . \forall_{t<T} \sum_{c<C} y_{c,t,p} \leq min * y_{lider,t,p}$$

In [190]:

```
for p in range(P):
    for t in range(T):
        lider= projetos[p+1][0]
        cols = projetos[p+1][2]
        minC = 0.5 * len(cols)
        horario.Add(sum([Y(c,p,t) for c in cols]) <= minC *  y[lider][p][t])

#print(colaboradores)
```

6. Cada colaborador só pode ser colocado num slot em que esteja disponivel

$$\forall_{t<T} . \forall_{p<P} . \forall_{c<C} t \notin colaboradores_c \implies y_{c,p,t} = 0$$

In [191]:

```
for t in range(T):
    for p in range(P):
        for c in range(C):
            if t not in colaboradores[c+1]:
                horario.Add(y[c][p][t] == 0)
```

7. Cada colaborador só pode participar num projeto de cada vez

$$\forall_{t<T} . \forall_{c<C} \sum_{p<P} y_{c,p,t} \leq 1$$

In [192]:

```
for t in range(T):
    for c in range(C):
        horario.Add(sum([Y(c,p,t) for p in range(P)]) <= 1)
```

8. A variavel z_{t,p} tem valor 1 caso haja alguma reuniao do projeto p no slot t e tem valor 0 caso contrario

$$(\forall_{t<T} . \forall_{p<P} \; z_{t,p} \leq (\sum_{s<S,t<T} x_{s,p,t})) \wedge (\forall_{t<T} . \forall_{p<P} . \forall_{s<S} \; z_{d,p} \leq x_{s,p,t})$$

In [193]:

```
for t in range (T):
    for p in range(P):
        horario.Add(z[t][p] <= sum([X(s,p,t) for s in range(S) for t in range(T)]))
        for s in range(S):
            horario.Add(z[t][p] >= x[s][p][t])
```

9. Maximizar o número de dias em que cada projeto tem reuniões

In [194]:

```
horario.Maximize(sum([Z(t,p) for t in range(T) for p in range(P)]))
```

Procura da solução do problema

In [195]:

```
status = horario.Solve()
#print(status)
#print(pywraplp.Solver.OPTIMAL)
if status == pywraplp.Solver.OPTIMAL:
    n = sum(int(X(s,p,t).solution_value())
        for p in range(P)
        for s in range(S)
        for t in range(T)
        )

    print("Solução encontrada:",n)
else:
    print("Solução não encontrada.")
```

```
Solução encontrada: 12
```

Construção e Apresentação dos Horários

In [196]:

```
import math
SlotsDia=T/5
for p in range(P):
    print("Projeto", p+1,":")
    for s in range(S):
        print("------Sala", s+1, "-------")
        for t in range(T):
            if x[s][p][t].solution_value() ==1:
                cols = projetos[p+1][2]
                print("Dia:", math.ceil((t+1)/SlotsDia))
                print(" Slot:",t+1, "Colaboradores :",cols)
            #else:
                #print(x[s][p][t].solution_value())
    print("\n")
```

```
Projeto 1 :
------Sala 1 -------
Dia: 3
 Slot: 5 Colaboradores : [1, 2]
Dia: 3
 Slot: 6 Colaboradores : [1, 2]
Dia: 4
 Slot: 7 Colaboradores : [1, 2]
------Sala 2 -------
Dia: 5
 Slot: 9 Colaboradores : [1, 2]


Projeto 2 :
------Sala 1 -------
Dia: 5
 Slot: 9 Colaboradores : [1]
------Sala 2 -------
Dia: 3
 Slot: 6 Colaboradores : [1]
Dia: 4
 Slot: 7 Colaboradores : [1]


Projeto 3 :
------Sala 1 -------
Dia: 1
 Slot: 2 Colaboradores : [2]
Dia: 2
 Slot: 3 Colaboradores : [2]
Dia: 2
 Slot: 4 Colaboradores : [2]
Dia: 4
 Slot: 8 Colaboradores : [2]
Dia: 5
 Slot: 10 Colaboradores : [2]
------Sala 2 -------
```