Horario

October 24, 2021

1 TP1 - Grupo 4

Pedro Paulo Costa Pereira - A88062

Tiago André Oliveira Leite - A91693

2 Problema 1 - Horário de uma StartUp

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).

2.1 Análise do problema

Pretende-se alocar reuniões de um projecto a salas, ao longo da semana, durante um "slot" (tempo, dia).

Vamos usar uma família $r_{d,t,s,p,c}$ de variáveis binárias em que

$r_{d,t,s,p,c} == 1$ se e só se o colaborador c estará presente numa reunião do projecto p alocada à sala s, no "slot"

Para tal, construíremos uma *matriz de alocação* R com valores $\{0,1\}^{C \times P \times S \times T \times D}$, para representar as $C \times P \times S \times T \times D$ variáveis binárias.

O problema, entretanto, dispõe de algumas limitações que deverão ser tratadas:

1. Um colaborador só pode ir a uma reunião que aconteça durante um "slot" em que esteja disponível.
2. Um colaborador não pode participar de uma reunião de um projecto que não pertença.
3. Não pode haver mais de uma reunião numa sala ao mesmo tempo.
4. Uma reunião só acontece se o líder do projecto está presente.
5. Um colaborador não pode estar em duas reuniões em salas diferentes ao mesmo tempo.

6. Uma reunião de um projecto só acontece se, pelo menos, 50% de todos os colaboradores desse projeto estão presentes.
7. Cada projecto tem um número de reuniões semanais.

2.2 Implementação

Para a resolução deste problema, utilizaremos a biblioteca de programação linear do OR-Tools, o *pywraplp*. Portanto, começaremos por instalar o OR-Tools, importar tal biblioteca e inicializar o *solver*, que chamaremos de *horario*.

[127]:
```
!pip install ortools
```

```
Requirement already satisfied: ortools in /usr/local/lib/python3.8/dist-packages
(9.1.9490)
Requirement already satisfied: absl-py>=0.13 in /usr/local/lib/python3.8/dist-
packages (from ortools) (0.14.1)
Requirement already satisfied: protobuf>=3.18.0 in
/usr/local/lib/python3.8/dist-packages (from ortools) (3.18.1)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from absl-
py>=0.13->ortools) (1.14.0)
```

[65]:
```
from ortools.linear_solver import pywraplp
```

Nesta seção, serão declaradas as constantes do problema (os primeiros "inputs") e algumas variáveis auxiliares.

Exemplo Input 1 (Possivel)

[87]:
```
D, T, S = 5, 8, 2    #5 dias, cada dia com um período de 8 horas; 2 salas

projectos = {}
projectos[1] = (1,[1,2,3],2)
projectos[2] = (4,[1,4,2],1)
projectos[3] = (3,[3,5,2],2)

availability = {}
availability[1] = [(t, d) for t in range(1,5) for d in range(1,6)]   #o␣
↪colaborador 1 está disponível dentre as horas 1 a 4 nos dias 1 a 5.
availability[2] = [(t, d) for t in range(1,9) for d in range(1,6)]
availability[3] = [(t, d) for t in range(6,9) for d in range(1,6)]
availability[4] = [(t, d) for t in range(4,9) for d in range(1,6)]
availability[5] = [(t, d) for t in range(4,9) for d in range(1,6)]
```

Exemplo Input 2 (Impossivel)

[107]:
```
D, T, S = 5, 8, 2

projectos = {}
projectos[1] = (1,[1,2,3],2)
```

```
projectos[2] = (4,[1,4,2],1)
projectos[3] = (3,[3,5,2],2)

availability = {}
availability[1] = [(t, d) for t in range(1,3) for d in range(1,6)]
availability[2] = [(t, d) for t in range(3,5) for d in range(1,6)]
availability[3] = [(t, d) for t in range(5,9) for d in range(1,6)]
availability[4] = [(t, d) for t in range(4,9) for d in range(1,6)]
availability[5] = [(t, d) for t in range(4,9) for d in range(1,6)]
```

Exemplo Input 3 (Possivel)

[117]:
```
D, T, S = 5, 8, 2

projectos = {}
projectos[1] = (1,[1,2,3,6,7],2)
projectos[2] = (4,[1,4,2],1)
projectos[3] = (3,[3,5,2],2)


availability = {}
availability[1] = [(t, d) for t in range(1,3) for d in range(1,6)]
availability[2] = [(t, d) for t in range(2,5) for d in range(1,6)]
availability[3] = [(t, d) for t in range(5,9) for d in range(1,6)]
availability[4] = [(t, d) for t in range(4,9) for d in range(1,6)]
availability[5] = [(t, d) for t in range(4,9) for d in range(1,6)]
availability[6] = [(t, d) for t in range(4,9) for d in range(1,6)]
availability[7] = [(t, d) for t in range(1,4) for d in range(1,6)]
```

Agora, declaramos a matriz de alocação R como um dicionário.

[118]:
```
horario = pywraplp.Solver.CreateSolver('SCIP')
P = len(projectos)  #número de projectos
L = [projectos[p][0] for p in range(1,P+1)]  #líderes dos projectos
C = len(availability)  #número de colaboradores
r = {}

for c in range(1,C+1):
  r[c] = {}
  for p in range(1,P+1):
    r[c][p] = {}
    for s in range(1,S+1):
      r[c][p][s] = {}
      for t in range(1,T+1):
        r[c][p][s][t] = {}
        for d in range(1,D+1):
```

```
          r[c][p][s][t][d] = horario.BoolVar('r[%i][%i][%i][%i][%i]' %(c, p, s,␣
 ↪t, d))

#abreviatura
def R(c, p, s, t, d):
  return r[c][p][s][t][d]
```

Precisamos, nesta etapa de modelar as restrições e introduzi-las ao *solver*.

A restrição

1. Um colaborador não pode comparecer a uma reunião que acontece num "slot" em que o mesmo não está disponível.

pode ser expressa como:

$\forall_{c<C} \cdot \forall_{p<P} \cdot \forall_{s<S} \cdot \forall_{t<T} \cdot \forall_{d<D} \cdot r_{c,p,s,t,d} == 0$ se o colaborador c não está disponível no "slot" (t, d).

Ou seja, se (t, d) não está na lista correspondente à chave c do dicionário *availability*, o coordenador c não pode estar presente na reunião.

[119]:
```
for c in range(1,C+1):
  for t in range(1,T+1):
    for d in range(1,D+1):
      if (t, d) not in availability[c]:
        for s in range(1,S+1):
          for p in range(1,P+1):
            horario.Add(R(c, p, s, t, d) == 0)
```

2. Um colaborador não pode participar numa reunião de um projecto que não pertença

pode ser expressa como:

$\forall_{c<C} \cdot \forall_{p<P} \cdot \forall_{s<S} \cdot \forall_{t<T} \cdot \forall_{d<D} \cdot r_{c,p,s,t,d} == 0$ se o colaborador c não faz parte do projeto p.

[120]:
```
for c in range(1,C+1):
  for p in range(1,P+1):
    if c not in projectos[p][1]:
      for s in range(1,S+1):
        for t in range(1,T+1):
          for d in range(1,D+1):
            horario.Add(R(c, p, s, t, d) == 0)
```

3. Não pode haver mais de uma reunião numa sala ao mesmo tempo

é equivalente a dizer que, para toda sala e todo "slot" (t,d), a soma de todas as variáveis $r_{l,p,s,t,d}$ tais que $p < P, l < L$ é menor ou igual a 1, onde L é o conjunto de líderes dos projectos, ou seja:

$$\forall_{s<S} \cdot \forall_{t<T} \cdot \forall_{d<D} \cdot \sum_{p<P, l<L} r_{l,p,s,t,d} \leq 1$$

[121]:
```
for s in range(1,S+1):
  for t in range(1,T+1):
    for d in range(1,D+1):
      horario.Add(sum([R(l, p, s, t, d) for p in range(1,P+1) for l in L]) <= 1)
```

4. Uma reunião de um projecto só acontece quando o líder do mesmo está presente.

Em outras palavras, um colaborador de um projecto nunca estará numa reunião em que o líder do projecto não está, ou seja,

$$\forall_{c<C} \cdot \forall_{p<P} \cdot \forall_{s<S} \cdot \forall_{t<T} \cdot \forall_{d<D} \cdot r_{l,p,s,t,d} \geq r_{c,p,s,t,d}$$

sendo l o líder do projecto p.

[122]:
```
for p in range(1,P+1):
  leader = L[p-1]
  for s in range(1,S+1):
    for t in range(1,T+1):
      for d in range(1,D+1):
        for c in range(1,C+1):
          horario.Add(R(leader, p, s, t, d) >= R(c, p, s, t, d))
```

5. Um colaborador não pode estar em duas reuniões ao mesmo tempo,

ou seja,

$$\forall_{c<C} \cdot \forall_{t<T} \cdot \forall_{d<D} \cdot \sum_{p<P,\, s<S} r_{c,p,s,t,d} \leq 1$$

[123]:
```
for c in range(1,C+1):
  for t in range(1,T+1):
    for d in range(1,D+1):
      horario.Add(sum([R(c, p, s, t, d) for p in range(1,P+1) for s in␣
↪range(1,S+1)]) <= 1)
```

6. Uma reunião de um projecto só acontece se, pelo menos, 50% de todos os colaboradores desse projecto estão presentes.

Para cada projecto p, temos o líder l e um quorum tal que \$quorum = (número de colaboradores do projecto p) / 2\$.

Como as reuniões só acontecem se o líder do projecto está presente, usaremos a variável $r_{l,p,s,t,d}$ para garantir que o quorum seja atingido em "slots" nos quais o líder pode estar presente. Logo, temos:

$$\forall_{p<P} \cdot \forall_{s<S} \cdot \forall_{t<T} \cdot \forall_{d<D} \cdot \sum_{c<C} r_{c,p,s,t,d} \geq (r_{l,p,s,t,d} * quorum)$$

[124]:
```
for p in range(1,P+1):
  l = L[p-1]
  quorum = len(projectos[p][1]) / 2
  for s in range(1,S+1):
```

```
    for t in range(1,T+1):
      for d in range(1,D+1):
        horario.Add(sum([R(c, p, s, t, d) for c in range(1,C+1)]) >= (R(l, p,␣
↪s, t, d) * quorum))
```

7. Cada projecto tem um número de reuniões semanais.

Seja N_p o número de reuniões semanais do projecto p (*input* do problema).

Então, para cada projecto p, temos que garantir que a soma das presenças do líder l em todas as reuniões desse projeto seja igual a N_p, ou seja,

$$\forall_{p<P} \cdot \sum_{s<S,t<T,d<D} r_{l,p,s,t,d} == N_p$$

[125]:
```
for p in range(1,P+1):
  N = projectos[p][2]
  l = projectos[p][0]
  horario.Add(sum([R(l, p, s, t, d) for s in range(1,S+1) for t in range(1,T+1)␣
↪for d in range(1,D+1)]) == N)
```

Com todas as restrições já introduzidas ao *solver*, basta procurarmos por uma solução. Caso exista, será impresso o horário semanal dividido por projeto.

[126]:
```
status = horario.Solve()
if status == pywraplp.Solver.OPTIMAL:
  for p in range(1,P+1):
        print(f'Projeto:{p}')
        l = projectos[p][0]
        for d in range(1,D+1):
            for t in range(1,T+1):
                for s in range(1,S+1):
                    if R(l, p, s, t, d).solution_value() == 1:
                        print(f'Dia: {d} Slot: {t} Sala: {s}')
                        print("Colaboradores:", end= " ")
                        print(f'*{l}', end=" ")
                        for c in range(1,C+1):
                            if R(c, p, s, t, d).solution_value()== 1 and c != l:
                                print(f'{c}', end=" ")
                        print("")
        print("")
else:
  print("Nao é possivel construir o hórario")
```

```
Projeto:1
Dia: 1 Slot: 2 Sala: 2
Colaboradores: *1 2 7
Dia: 4 Slot: 2 Sala: 1
Colaboradores: *1 2 7
```

```
Projeto:2
Dia: 1 Slot: 4 Sala: 1
Colaboradores: *4 2

Projeto:3
Dia: 3 Slot: 5 Sala: 1
Colaboradores: *3 5
Dia: 4 Slot: 5 Sala: 1
Colaboradores: *3 5
```