StartUp

November 1, 2020

1 StartUp

Grupo 7

- Luís Almeida A84180
- João Pedro Antunes A86813

1.1 Enunciado e Descrição Do Problema:

Foi nos proposta a resolução de um problema cujo objetivo é construir o 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:

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

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

c) 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.

d) 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).

1.1.1 Input

Começamos por definir o input do nosso problema. Iremos ter as variáveis S, D, T, P e C, que representam o número de Salas disponíveis na Startup, o número de Dias de trabalho, o número de Tempos por dia, o número de Projetos e o número total de Colaboradores, respetivamente. Adicionalmente, teremos 2 dicionários:

1) O primeiro dicionário, colPro, estabelecerá uma correspondência entre cada Projeto p e um triplo t cujo conteúdo é o conjunto dos colaboradores que fazem parte do projeto p, o líder do mesmo e o número N de reuniões semanais do projeto.

2) O segundo dicionário, disp, estabelecerá uma correspondência entre cada Colaborador c e uma lista com os slots que determinam a sua disponibilidade.

[67]:
```
from z3 import*
import math

#[Projeto] : (colaboradores, lider, N)
#[Colaborador] : [(tempo, dia)]

S = 2
D = 3
T = 2
P = 3
C = 3
colPro = {1 : ([1,3,2],1,2), 2: ([1,2],2,2), 3:([1,3,2],3,2)}
disp = {1:[(1,1),(1,2),(2,1),(2,2),(3,1),(3,2)], 2: [(2,1),(2,2),(3,1)], 3:
 ↪[(1,1),(1,2),(2,1),(2,2),(3,2)]}
```

1.1.2 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 S,D,T,P e C, podemos criar 2 matrizes de alocação com o seguinte significado:

$$X_{P,S,D,T} == 1 \quad \text{se e só se} \quad \text{o projeto } p \text{ é alocado à sala } s \text{ no dia } d \text{ no tempo } t$$

$$Y_{P,C,D,T} == 1 \quad \text{se e só se} \quad \text{o projeto } p \text{ é alocado ao colaborador } c \text{ no dia } d \text{ no tempo } t$$

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

Limitações

1. Cada sala tem alocada, no máximo, 1 projeto por cada tempo
2. Cada colaborador tem alocado, no máximo, 1 projeto por cada tempo
3. Cada projeto tem alocado, no máximo, 1 sala por cada tempo
4. Um colaborador não pode ser alocado a uma reunião não alocada
5. Um colaborador que não faça parte de um projeto não pode ser alocado a uma reunião desse projeto
6. Um colaborador não pode ser alocado a uma reunião alocada a um slot fora da sua disponibilidade

Obrigações

1. Cada projeto tem de ter alocadas N reuniões
2. Cada líder tem de estar alocado a todas as reuniões do seu projeto
3. Cada reunião tem de ter um Quorum mínimo de 50% do total de colaboradores do projeto

1.2 Implementação

Comecemos por criar as diferentes variáveis necessárias à resolução do problema, e as respetivas matrizes de alocação que as vão conter:

[68]:
```
horario =  Solver()
X = {}
Y = {}

#alocar variaveis binarias para Projetos,Sala,dia,tempo
for p in range(1,P+1):
    for s in range(1,S+1):
        for d in range(1,D+1):
            for t in range(1,T+1):
                X[p,s,d,t] = Int("sal "+str(p)+ ' ' +str(s)+ ' ' +str(d)+ ' '␣
 ↪+str(t))
                horario.add(X[p,s,d,t] >= 0, X[p,s,d,t] <= 1)

#alocar variaveis binarias para Projetos,colaboradores,dia,tempo
for p in range(1,P+1):
    for c in range(1,C+1):
        for d in range(1,D+1):
            for t in range(1,T+1):
                Y[p,c,d,t] = Int("col "+str(p)+ ' ' +str(c)+ ' ' +str(d)+ ' '␣
 ↪+str(t))
                horario.add(Y[p,c,d,t] >= 0, Y[p,c,d,t] <= 1)
```

Passemos agora à implementação das *limitações*:

1. *Cada sala tem alocada, no máximo, 1 projeto por cada tempo*

Esta restrição pode ser representada pela seguinte fórmula:

$$\forall_s \forall_d \forall_t \quad \sum_{p<P} X_{p,s,d,t} \leq 1$$

[69]:
```
for s in range(1,S+1):
    for d in range(1,D+1):
        for t in range(1,T+1):
            horario.add(Sum([X[p,s,d,t] for p in range(1,P+1)]) <= 1)
```

2. *Cada colaborador tem alocado, no máximo, 1 projeto por cada tempo*

Esta restrição pode ser representada pela seguinte fórmula:

$$\forall_c \forall_d \forall_t \quad \sum_{p<P} Y_{p,c,d,t} \leq 1$$

[70]:
```
for c in range(1,C+1):
    for d in range(1,D+1):
        for t in range(1,T+1):
            horario.add(Sum([Y[p,c,d,t] for p in range(1,P+1)]) <= 1)
```

3. *Cada projeto tem alocado, no máximo, 1 sala por cada tempo*

Esta restrição pode ser representada pela seguinte fórmula:

$$\forall_p \forall_d \forall_t \quad \sum_{s<S} X_{p,s,d,t} \leq 1$$

[71]:
```
for p in range(1,P+1):
    for d in range(1,D+1):
        for t in range(1,T+1):
            horario.add(Sum([X[p,s,d,t] for s in range(1,S+1)]) <= 1)
```

4. *Um colaborador não pode ser alocado a uma reunião não alocada*

Esta restrição pode ser representada pela seguinte fórmula:

$$\forall_p \forall_c \forall_d \forall_t \quad \sum_{s<S} X_{p,s,d,t} - Y_{p,c,d,t} \geq 0$$

Pois caso uma reunião do projeto p esteja alocada para a sala s, a inequação será verdadeira, caso contrário será falsa.

[72]:
```
for p in range(1,P+1):
    for c in range(1,C+1):
        for d in range(1,D+1):
            for t in range(1,T+1):
                horario.add(Sum([X[p,s,d,t] for s in range(1,S+1)]) -␣
↪Y[p,c,d,t] >= 0)
```

5. *Um colaborador que não faça parte de um projeto não pode ser alocado a uma reunião desse projeto*

Esta restrição pode ser representada pela seguinte fórmula:

Seja c um colaborador que não pertence ao projeto p

$$\forall_p \quad \sum_{d<D, t<T} Y_{p,c,d,t} == 0$$

[73]:
```
for p in range(1,P+1):
    aux = set(colPro[p][0])#lista de colaboradores que fazem parte do projeto
    aux.difference_update(range(1,C+1))
    for c in aux: #se o colaborador não fizer parte do projeto
        horario.add(Sum([Y[p,c,d,t] for d in range(1,D+1) for t in␣
↪range(1,T+1)]) == 0)#não pode ser alocado
```

6. *Um colaborador não pode ser alocado a uma reunião alocada a um slot fora da sua disponibilidade*

Esta restrição pode ser representada pela seguinte fórmula:

Seja (d, t) um *slot* que não pertence à disponibilidade do colaborador c

$$\forall_c \quad \sum_{p<P} Y_{p,c,d,t} == 0$$

[74]:
```
for c in disp:
    for d in range(1,D+1):
        for t in range(1,T+1):
            if (d,t) not in disp[c]:
                horario.add(Sum([Y[p,c,d,t] for p in range(1,P+1)]) == 0)
```

Implementação das *obrigações*:

1. *Cada projeto tem de ter alocadas N reuniões*

Esta restrição pode ser representada pela seguinte fórmula:

$$\forall_p \sum_{s<S,d<D,t<T} X_{p,s,d,t} == N$$

[75]:
```
for p in range(1,P+1):
    reunioes = colPro[p][2]
    horario.add(Sum([X[p,s,d,t] for s in range(1,S+1) for d in range(1,D+1) for␣
 ↪t in range(1,T+1)]) == reunioes)
```

2. *Cada líder tem de estar alocado a todas as reuniões do seu projeto*

Esta restrição pode ser representada pela seguinte fórmula:

Seja l um líder de um projeto p:

$$\forall_p \sum_{t<T,d<D} Y_{p,l,d,t} == N$$

[76]:
```
for p in range(1,P+1):
    l = colPro[p][1]
    reunioes = colPro[p][2]
    horario.add(Sum([Y[p,l,d,t] for d in range(1,D+1) for t in range(1,T+1)])␣
 ↪== reunioes)
```

3. *Cada reunião tem de ter um Quorum mínimo de 50% do total de colaboradores do projeto*

Esta restrição pode ser representada pela seguinte fórmula:

Seja $C(p)$ o conjunto de todos os colaboradores do projeto p e seja l o líder desse projeto. Então, para todo o $c \in C(p)$

$$\forall_p \forall_d \forall_t \quad \sum Y_{p,c,d,t} \geq \frac{\#C(p)}{2} * Y_{p,l,d,t}$$

Devido ao facto de o líder l estar presente em todas as reuniões do seu projeto p, se uma delas não tiver sido alocada no dia d no tempo t, o valor da variável $Y_{p,l,d,t}$ será igual a 0, logo não incorreremos em situações em que forçamos uma reunião que não foi alocada a ter quorum mínimo.

[77]:
```
for p in range(1,P+1):
    for d in range(1,D+1):
        for t in range(1,T+1):
```

```
            l = colPro[p][1]
            horario.add(Sum([Y[p,c,d,t] for c in colPro[p][0]]) >=  math.
↪ceil(len(colPro[p][0])/2)* Y[p,l,d,t])
```

Impressão do resultado

[79]:
```
t = horario.check()
print(t)
if t == sat:
    m = horario.model()
    for p in range(1,P+1):
        for s in range(1,S+1):
            for d in range(1,D+1):
                for t in range(1,T+1):
                    if m[X[p,s,d,t]] == 1:
                        col = []
                        for c in range(1,C+1):
                            if m[Y[p,c,d,t]] == 1:
                                col.append(c)
                        print(f'Projeto: {p} Sala: {s} Dia: {d} Tempo: {t}␣
↪Colaboradores: {col}')
```

```
sat
Projeto: 1 Sala: 1 Dia: 3 Tempo: 2 Colaboradores: [1, 3]
Projeto: 1 Sala: 2 Dia: 2 Tempo: 2 Colaboradores: [1, 3]
Projeto: 2 Sala: 2 Dia: 2 Tempo: 1 Colaboradores: [2]
Projeto: 2 Sala: 2 Dia: 3 Tempo: 1 Colaboradores: [1, 2]
Projeto: 3 Sala: 1 Dia: 1 Tempo: 2 Colaboradores: [1, 3]
Projeto: 3 Sala: 1 Dia: 2 Tempo: 1 Colaboradores: [1, 3]
```

De seguida apresentamos alguns inputs que testam as diferentes restrições:

[ ]:
```
"""
#unsat - Colaborador não pode ser alocado a uma reunião alocada a um slot fora␣
↪da sua disponibilidade
S = 2
D = 2
T = 2
P = 3
C = 3
colPro = {1 : ([1,3,2],1,2), 2: ([1,2],2,2), 3:([1,3,2],3,2)}
disp = {1:[(1,2),(2,1),(2,2)], 2: [(2,1),(2,2)], 3:[(1,1),(1,2),(2,1),(2,2)]}

#unsat - Cada sala só pode ser alocada a um projeto por tempo
S = 1
D = 2
T = 2
P = 3
```

```
C = 3
colPro = {1 : ([1,3,2],1,2), 2: ([1,2],2,2), 3:([1,3,2],3,2)}
disp = {1:[(1,1),(1,2),(2,1),(2,2)], 2: [(1,1),(1,2),(2,1),(2,2)], 3:
 ↪[(1,1),(1,2),(2,1),(2,2)]}

#unsat - Cada projeto tem N reuniões alocadas (tempos insuficientes)
S = 2
D = 2
T = 1
P = 3
C = 3
colPro = {1 : ([1,3,2],1,2), 2: ([1,2],2,2), 3:([1,3,2],3,2)}
disp = {1:[(1,1),(2,1)], 2: [(1,1),(2,1)], 3:[(1,1),(2,1)]}

#unsat - Cada projeto tem N reuniões alocadas (dias insuficientes)
S = 2
D = 1
T = 2
P = 3
C = 3
colPro = {1 : ([1,3,2],1,2), 2: ([1,2],2,2), 3:([1,3,2],3,2)}
disp = {1:[(1,1),(1,2)], 2: [(1,1),(1,2)], 3:[(1,1),(1,2)]}

#unsat - O líder de cada projeto tem de estar presente em todas as reuniões␣
 ↪desse projeto
S = 2
D = 2
T = 2
P = 2
C = 3
colPro = {1 : ([1,3,2],1,2), 2:([1,3,2],3,2)}
disp = {1:[(1,1),(1,2),(2,1),(2,2)], 2: [(1,1),(1,2),(2,1),(2,2)], 3:[(1,1)]}"""
```

1.3 Versão SCIP

Temos também uma versão da nossa resolução do problema usando SCIP, que é uma tradução direta da resolução em Z3:

[38]:
```
from pyscipopt import *

def startUpScip(S,D,T,P,C,colPro,disp):
    horario = Model()
    X = {}
    Y = {}

    #alocar variaveis binarias para Projetos,Sala,dia,tempo
    for p in range(1,P+1):
```

```
        for s in range(1,S+1):
            for d in range(1,D+1):
                for t in range(1,T+1):
                    X[p,s,d,t] = horario.addVar("sal "+str(p)+ ' ' +str(s)+ ' '␣
↪+str(d)+ ' ' +str(t), vtype="INTEGER")
                    horario.addCons(X[p,s,d,t] >= 0)
                    horario.addCons(X[p,s,d,t] <= 1)


    #alocar variaveis binarias para Projetos,colaboradores,dia,tempo
    for p in range(1,P+1):
        for c in range(1,C+1):
            for d in range(1,D+1):
                for t in range(1,T+1):
                    Y[p,c,d,t] = horario.addVar("col "+str(p)+ ' ' +str(c)+ ' '␣
↪+str(d)+ ' ' +str(t), vtype="INTEGER")
                    horario.addCons(Y[p,c,d,t] >= 0)
                    horario.addCons(Y[p,c,d,t] <= 1)


    #cada sala só é ocupada por 1 projeto por dia e tempo
    for s in range(1,S+1):
        for d in range(1,D+1):
            for t in range(1,T+1):
                horario.addCons(sum([X[p,s,d,t] for p in range(1,P+1)]) <= 1)


    #cada colaborador só participa de 1 projeto por dia/tempo
    for c in range(1,C+1):
        for d in range(1,D+1):
            for t in range(1,T+1):
                horario.addCons(sum([Y[p,c,d,t] for p in range(1,P+1)]) <= 1)


    #O mesmo projeto não pode ser alocado simultaneamente em duas salas
    for p in range(1,P+1):
        for d in range(1,D+1):
            for t in range(1,T+1):
                horario.addCons(sum([X[p,s,d,t] for s in range(1,S+1)]) <= 1)


    #o colaborador não pode estar alocado em um projeto não alocado (soma das␣
↪salas alocadas pro projeto - colaborador alocado pro projeto >= 0)
    for p in range(1,P+1):
        for c in range(1,C+1):
            for d in range(1,D+1):
                for t in range(1,T+1):
```

```
                    horario.addCons(sum([X[p,s,d,t] for s in range(1,S+1)]) -␣
↪Y[p,c,d,t] >= 0)


    #o colaborador que não faça parte de um projeto não poderá estar no projeto
    for p in range(1,P+1):
        aux = colPro[p][0] #lista de colaboradores que fazem parte do projeto
        for c in range(1,C+1):
            if c not in aux:
                horario.addCons(sum([Y[p,c,d,t] for d in range(1,D+1) for t in␣
↪range(1,T+1)]) == 0)



    #O lider participa de todas as reuniões de seu projeto
    for p in range(1,P+1):
        lider = colPro[p][1]
        reunioes = colPro[p][2]
        horario.addCons(sum([Y[p,lider,d,t] for d in range(1,D+1) for t in␣
↪range(1,T+1)]) == reunioes)


    #O Projeto tem N reuniões alocadas
    for p in range(1,P+1):
        reunioes = colPro[p][2]
        horario.addCons(sum([X[p,s,d,t] for s in range(1,S+1) for d in␣
↪range(1,D+1) for t in range(1,T+1)]) == reunioes)


    #Quórum mínimo de 50%
    for p in range(1,P+1):
        for d in range(1,D+1):
            for t in range(1,T+1):
                lider = colPro[p][1]
                horario.addCons(sum([Y[p,c,d,t] for c in colPro[p][0]]) >= ␣
↪math.ceil(len(colPro[p][0])/2)* Y[p,lider,d,t])


    #levar em conta a disponibilidade de cada colaborador
    for c in disp:
        for d in range(1,D+1):
            for t in range(1,T+1):
                if (d,t) not in disp[c]:
                    horario.addCons(sum([Y[p,c,d,t] for p in range(1,P+1)]) ==␣
↪0)
```

```
    horario.optimize()
    t = horario.getStatus()
    print(t)


    if t == 'optimal':
        m = horario.getBestSol()

        for p in range(1,P+1):
            for s in range(1,S+1):
                for d in range(1,D+1):
                    for t in range(1,T+1):
                        if(m[X[p,s,d,t]] == 1):
                            col = []
                            for c in range(1,C+1):
                                if m[Y[p,c,d,t]] == 1:
                                    col.append(c)
                            print(f'Projeto: {p} Sala: {s} Dia: {d} Tempo: {t}␣
↪Colaboradores: {col}')
    return
```

A seguinte função de teste recebe um número N de projetos e gera um input para a função *startUpScip*, testando esse input:

[39]:
```
def teste(N):
    p = N
    c = N
    teste = Solver()
    (S,D,T) = Ints("s d t")
    teste.add(S*D*T == p)
    t = teste.check()

    if t == sat:
        m = teste.model()
        s = m[S].as_long()
        d = m[D].as_long()
        t = m[T].as_long()

        cp = {x :([x],x,1) for x in range(1,c+1)}

        disp = {x: [(i,j) for i in range(1,d+1) for j in range(1,t+1)] for x in␣
↪range(1,c+1)}

    return (s,d,t,p,c,cp,disp)

startUpScip(*teste(50))
```

```
<ipython-input-38-8b9b24598210>:4: UserWarning: linked SCIP 7.0 is not
recommended for this version of PySCIPOpt - use version 7.0.1
  horario = Model()

optimal
Projeto: 1 Sala: 1 Dia: 19 Tempo: 1 Colaboradores: [1]
Projeto: 2 Sala: 1 Dia: 31 Tempo: 1 Colaboradores: [2]
Projeto: 3 Sala: 1 Dia: 27 Tempo: 1 Colaboradores: [3]
Projeto: 4 Sala: 1 Dia: 41 Tempo: 1 Colaboradores: [4]
Projeto: 5 Sala: 1 Dia: 18 Tempo: 1 Colaboradores: [5]
Projeto: 6 Sala: 1 Dia: 48 Tempo: 1 Colaboradores: [6]
Projeto: 7 Sala: 1 Dia: 47 Tempo: 1 Colaboradores: [7]
Projeto: 8 Sala: 1 Dia: 36 Tempo: 1 Colaboradores: [8]
Projeto: 9 Sala: 1 Dia: 20 Tempo: 1 Colaboradores: [9]
Projeto: 10 Sala: 1 Dia: 29 Tempo: 1 Colaboradores: [10]
Projeto: 11 Sala: 1 Dia: 38 Tempo: 1 Colaboradores: [11]
Projeto: 12 Sala: 1 Dia: 45 Tempo: 1 Colaboradores: [12]
Projeto: 13 Sala: 1 Dia: 23 Tempo: 1 Colaboradores: [13]
Projeto: 14 Sala: 1 Dia: 46 Tempo: 1 Colaboradores: [14]
Projeto: 15 Sala: 1 Dia: 22 Tempo: 1 Colaboradores: [15]
Projeto: 16 Sala: 1 Dia: 28 Tempo: 1 Colaboradores: [16]
Projeto: 17 Sala: 1 Dia: 30 Tempo: 1 Colaboradores: [17]
Projeto: 18 Sala: 1 Dia: 34 Tempo: 1 Colaboradores: [18]
Projeto: 19 Sala: 1 Dia: 50 Tempo: 1 Colaboradores: [19]
Projeto: 20 Sala: 1 Dia: 32 Tempo: 1 Colaboradores: [20]
Projeto: 21 Sala: 1 Dia: 1 Tempo: 1 Colaboradores: [21]
Projeto: 22 Sala: 1 Dia: 4 Tempo: 1 Colaboradores: [22]
Projeto: 23 Sala: 1 Dia: 7 Tempo: 1 Colaboradores: [23]
Projeto: 24 Sala: 1 Dia: 12 Tempo: 1 Colaboradores: [24]
Projeto: 25 Sala: 1 Dia: 13 Tempo: 1 Colaboradores: [25]
Projeto: 26 Sala: 1 Dia: 16 Tempo: 1 Colaboradores: [26]
Projeto: 27 Sala: 1 Dia: 21 Tempo: 1 Colaboradores: [27]
Projeto: 28 Sala: 1 Dia: 24 Tempo: 1 Colaboradores: [28]
Projeto: 29 Sala: 1 Dia: 25 Tempo: 1 Colaboradores: [29]
Projeto: 30 Sala: 1 Dia: 26 Tempo: 1 Colaboradores: [30]
Projeto: 31 Sala: 1 Dia: 37 Tempo: 1 Colaboradores: [31]
Projeto: 32 Sala: 1 Dia: 49 Tempo: 1 Colaboradores: [32]
Projeto: 33 Sala: 1 Dia: 33 Tempo: 1 Colaboradores: [33]
Projeto: 34 Sala: 1 Dia: 2 Tempo: 1 Colaboradores: [34]
Projeto: 35 Sala: 1 Dia: 17 Tempo: 1 Colaboradores: [35]
Projeto: 36 Sala: 1 Dia: 35 Tempo: 1 Colaboradores: [36]
Projeto: 37 Sala: 1 Dia: 40 Tempo: 1 Colaboradores: [37]
Projeto: 38 Sala: 1 Dia: 43 Tempo: 1 Colaboradores: [38]
Projeto: 39 Sala: 1 Dia: 44 Tempo: 1 Colaboradores: [39]
Projeto: 40 Sala: 1 Dia: 11 Tempo: 1 Colaboradores: [40]
Projeto: 41 Sala: 1 Dia: 42 Tempo: 1 Colaboradores: [41]
Projeto: 42 Sala: 1 Dia: 39 Tempo: 1 Colaboradores: [42]
Projeto: 43 Sala: 1 Dia: 6 Tempo: 1 Colaboradores: [43]
```

```
Projeto: 44 Sala: 1 Dia: 3 Tempo: 1 Colaboradores: [44]
Projeto: 45 Sala: 1 Dia: 5 Tempo: 1 Colaboradores: [45]
Projeto: 46 Sala: 1 Dia: 8 Tempo: 1 Colaboradores: [46]
Projeto: 47 Sala: 1 Dia: 9 Tempo: 1 Colaboradores: [47]
Projeto: 48 Sala: 1 Dia: 10 Tempo: 1 Colaboradores: [48]
Projeto: 49 Sala: 1 Dia: 14 Tempo: 1 Colaboradores: [49]
Projeto: 50 Sala: 1 Dia: 15 Tempo: 1 Colaboradores: [50]
```