```
!pip install ortools
```

```
Requirement already satisfied: ortools in /usr/local/lib/python3.7/dist-packages (9.1.9490)
Requirement already satisfied: protobuf>=3.18.0 in /usr/local/lib/python3.7/dist-packages (from ortools) (3.19.0)
Requirement already satisfied: absl-py>=0.13 in /usr/local/lib/python3.7/dist-packages (from ortools) (0.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from absl-py>=0.13->ortools) (1.15.0)
```

Trabalho 1

Este trabalho foi realizado por:

- João Pedro Goulart - A82643
- Tiago Rodrigues - A87952

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,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).

Análise do problema

Condicionantes do problema:

1. Cada sala tem, num slot, um e apenas um projeto associado.
2. Cada colaborador tem, no máximo, um projeto associado.
3. Cada projeto realiza, no máximo, R reuniões por semana.
4. Cada projeto tem, no máximo e, para um determinado slot, uma sala.
5. Cada líder tem, no máximo, um projeto.

Para além disto, o problema tem como limitações mínimas:

1. O líder de cada projeto participa em todas as reuniões do respectivo projeto.
2. Cada reunião relativa a um projeto terá, no mínimo, a presença de 50% dos colaboradores desse mesmo projeto.
3. Cada sala contará com um tempo disponível que estará entre 0 e T.
4. Uma reunião só poderá ser efetuada se a disponibilidade dos colaboradores e do líder o permitir.

Implementação

Começamos por importar a bilioteca de programação linear do OR-Tools e criar uma instância do solver. De seguida inicializamos o solver horário e definimos os valores para as constantes do problema S,D,T,P e C.

```
from ortools.linear_solver import pywraplp

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

S, D, T, P, C = 4, 5, 8, 10, 10

#Onde:
#S - Salas existentes
```

```
#D - Dias da semana
#T - Tempo disponível de sala
#P - Número de projetos
#C - Número de colaboradores por projeto, líder inclusivé

#Número de líderes (equivalnte ao número de projetos existentes)
L = 10

#Número máximo de reuniões semanais para cada projeto
#((T/periodo)*D*S)/P
R = 5

#Número máximo de reuniões por sala no espaço de um dia
#N = T / periodo
```

De seguida, declaramos a matriz de alocação X.

```
x = {}
for p in range(P):
  x[p] = {}
  for c in range(C):
    x[p][c] = {}
    for s in range(S):
      x[p][c][s] = {}
      for t in range(T):
        x[p][c][s][t] = {}
        for d in range(D):
          x[p][c][s][t][d] = horario.BoolVar('x[%i][%i][%i][%i][%i]' %(p,c,s,t,d))


def X(p,c,s,t,d):           # Abreviatura
    return x[p][c][s][t][d]
```

Avançamos para as adaptações relativas às restrições do problema e à introdução das mesmas no *solver*.

Podemos expressar a condição

1. Cada reunião ocupa uma sala durante um slot (tempo,dia)

da seguinte forma:

$$\forall_{p<P} \sum_{s<S,t<T,d<D} x_{p,c,s,t,d} = 1$$

```
for p in range(P):
  horario.Add(sum([X(p,c,s,t,d) for s in range(S) for t in range(T) for d in range(D)]) == 1)
```

Podemos expressar a condição

2. Cada reunião tem associado um projeto e um conjunto de participantes/colaboradores

da seguinte forma:

$$\forall_{p<P} \cdot \sum_{c<C} x_{p,c,s,t,d} = 1$$

```
for p in range(P):
  horario.Add(sum([X(p,c,s,t,d) for c in range(C)]) == 1)
```

Podemos expressar a condição

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.

da seguinte forma:

$$\forall_{p<P} \cdot \sum_{l<L} x_{p,c,s,t,d} = 1$$

$$\forall_{p<P} \cdot \bigwedge_{r<R} \sum x_{p,c,s,t,d} \leq R$$

```
for p in range(P):
  horario.Add(sum([X(p,c,s,t,d) for l in range(L)]) == 1)

for p in range(P):
  horario.Add(sum([X(p,c,s,t,d) for r in range(R)]) <= R)
```

Podemos expressar a condição

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).

da seguinte forma:

(Nota: Para implementar esta condição, adicionámos uma família de variáveis binárias

$$y_{c,d}$$

que indicam a disponibilidade dos colaboradores c no dia d, restringindo o número máximo de dias)

$$\forall_{p<P} \cdot \sum_{l<L} x_{p,c,s,t,d} = 1$$

$$\forall_{c<C-1} \cdot \bigwedge_{d<D} \sum y_{c,d} \leq D$$

$$\forall_{p<P} \bigwedge_{c<C} \sum x_{p,c,s,t,d} \geq C/2$$

```
for p in range(P):
  horario.Add(sum([X(p,c,s,t,d) for l in range(L)]) == 1)
```

```
y = {}
for c in range(C):
  y[c] = []
  y[c]= horario.BoolVar('y[%i]' %(c))

for c in range(C-1):
  horario.Add(sum([Y(c,d) for d in range(D)]) <= D)


for p in range(P):
  horario.Add(sum([X(p,c,s,t,d) for c in range(C)]) >= C/2)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-23-dc4e46d1f9d3> in <module>()
      9
     10 for c in range(C-1):
---> 11   horario.Add(sum([Y(c,d) for d in range(D)]) <= D)
     12
     13

<ipython-input-23-dc4e46d1f9d3> in <listcomp>(.0)
      9
     10 for c in range(C-1):
---> 11   horario.Add(sum([Y(c,d) for d in range(D)]) <= D)
     12
     13

NameError: name 'Y' is not defined
```

SEARCH STACK OVERFLOW

2. Da definição do jogo “Sudoku” generalizado para a dimensão N; o problema tradicional corresponde ao caso N=3. O objetivo do Sudoku é preencher uma grelha de N^2xN^2 com inteiros positivos no intervalo 1 até N^2, satisfazendo as seguintes regras:

- Cada inteiro no intervalo 1 até N^2 ocorre só uma vez em cada coluna, linha e secção NxN.
- No início do jogo uma fração

$$0 \leq \alpha < 1$$

das N^4 casas da grelha são preenchidas de forma consistente com a regra anterior.

a) Construir um programa para inicializar a grelha a partir dos parâmetros

$$N \; e \; \alpha$$

```
from ortools.linear_solver import pywraplp

sudoku = pywraplp.Solver.CreateSolver('SCIP')
N = 3
alfa = 0.2
#l - número de linhas
#c - número de colunas
#e - espaço para inserir um número

a = {}
for l in range(N):
  a[l] = {}
  for c in range(N):
    a[l][c] = {}
    for e in range(N):
      a[l][c][e] = sudoku.BoolVar('a[%i][%i][%i]' %(l,c,e))

def A(c,l,e):
  return a[l][c][e]


tabela = [
[4, 0, 0, 0, 0, 5, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 9, 8],
```

```
[3, 0, 0, 0, 8, 2, 4, 0, 0],
[0, 0, 0, 1, 0, 0, 0, 8, 0],
[9, 0, 3, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 3, 0, 6, 7, 0],
[0, 5, 0, 0, 0, 9, 0, 0, 0],
[0, 0, 0, 2, 0, 0, 9, 0, 7],
[6, 4, 0, 3, 0, 0, 0, 0, 0],
]
```

- Cada inteiro no intervalo 1 até N^2 ocorre só uma vez em cada coluna, linha e secção NxN.

$$\forall_{1<x<N^2} \cdot \sum_{l<N, c<N, e<N} a_{l,c,e} = 1$$

```
for x in range(1,N^2):
  sudoku.Add(sum([a(l,c,e) for l in range(N) for c in range(N) for e in range(N)]) ==1)
```

b) Construir soluções do problema para as combinações de parâmetros

$$N \in \{3, 4, 5, 6\}$$

$$e$$

$$\alpha \in \{\ 0.0\ ,\ 0.2\ ,\ 0.4\ ,\ 0.6\ \}$$

0 s concluído à(s) 20:14