Laboratório de OGP 1: Implementação de modelos

Tasso Augusto Tomaz Pimenta 2021072198

```
from mip import Model, xsum, minimize, CBC, OptimizationStatus, BINARY, maximize
from itertools import product
import matplotlib.pyplot as plt
from math import sqrt
import numpy as np
import time
```

1. Problema de localização de facilidades

https://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/BildeKrarup.html

```
def facilidades(filename: str):
    with open(filename, 'r') as file:
        lines = file.readlines()
    header = lines[1].strip().split()
    nj = int(header[0])
    ni = int(header[1])
    f = []
    c = []
    #print(nj,ni)
    for line in lines[2:]:
        parts = line.strip().split()
        f.append(float(parts[1]))
        c.append([float(cost) for cost in parts[2:]])
    I, J = range(ni), range(nj)
    if len(f) != nj:
```

```
        raise ValueError("Dimensões das listas 'f'' não correspondem ao esperado.")
    if any(len(row) != ni for row in c):
        raise ValueError("Dimensões das listas 'c' não correspondem ao esperado.")

    model = Model('Problema de Localizacao', solver_name=CBC)
    model.verbose = 0

    y = [model.add_var(var_type=BINARY) for j in J]
    x = {(i, j): model.add_var(lb=0.0) for j in J for i in I}

    model.objective = minimize(xsum(f[j] * y[j] for j in J) + xsum(c[j][i] * x[i, j] for (i,

    for i in I:
        model += xsum(x[i, j] for j in J) == 1
    for (i, j) in product(I, J):
        model += x[i, j] <= y[j]

    start_time = time.time()
    status = model.optimize()
    end_time = time.time()

    elapsed_time = end_time - start_time
    if status == OptimizationStatus.OPTIMAL:
        print("Custo total de instalacao: {:12.2f}".format(sum([y[j].x * f[j] for j in J])))
        print("Custo total de transporte: {:12.2f} ".format(sum([x[i, j].x * c[j][i] for (i,
        print("Custo total : {:12.2f}.".format(model.objective_value))
        print("Tempo de resolução: {:.2f} segundos".format(elapsed_time))
        print("facilidades : demanda : clientes ")
        for j in J:
            if y[j].x > 1e-6:
                print("{:11d} : {:7.0f} : ".format(j + 1, sum([x[i, j].x for i in I])), end=
                for i in I:
                    if x[i, j].x > 1e-6:
                        print("{:d}".format(i + 1), end=' ')
                print()
    else: print('fail')
```

```
def open_file(file_list_path: str,path:str):
    with open(file_list_path) as file:
        filenames = file.readlines()
    it = 0
    for filename in filenames:
```

```
        if it == 2:break
        it +=1#ler só os primeiros 2
        filename = f'{path}\\{filename.strip()}'
        if filename:  # Ensure the line is not empty
            facilidades(filename)
open_file(r'BildeKrarup\C\files.lst','BildeKrarup\\C')
# https://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/BildeKrarup.html
```

```
Custo total de instalacao:      7255.00
Custo total de transporte:      9526.00
Custo total :     16781.00.
Tempo de resolução: 8.41 segundos
facilidades : demanda : clientes
          2 :      14 :  11 13 14 25 29 32 44 63 69 70 77 78 82 98
          5 :      24 :  3 7 17 20 22 23 30 40 46 47 53 54 61 67 73 76 83 85 86 87 88 89 90 9
         10 :      13 :  18 34 39 49 50 51 58 62 65 74 93 97 100
         22 :      17 :  2 4 6 10 12 26 27 37 38 43 52 56 64 66 84 94 96
         26 :      16 :  1 5 8 9 16 19 24 35 41 45 55 57 59 60 80 91
         43 :      16 :  15 21 28 31 33 36 42 48 68 71 72 75 79 81 95 99
Custo total de instalacao:      8450.00
Custo total de transporte:      7392.00
Custo total :     15842.00.
Tempo de resolução: 3.86 segundos
facilidades : demanda : clientes
          4 :      11 :  11 29 35 51 55 57 59 61 74 80 99
         20 :      16 :  14 20 21 30 34 50 53 60 73 75 77 83 84 89 93 95
         25 :      13 :  4 5 15 36 39 40 42 45 52 65 69 78 82
         29 :      20 :  3 10 13 17 23 24 26 27 32 56 62 66 68 70 79 88 90 92 97 98
         33 :      10 :  8 25 28 41 63 72 76 81 91 96
         43 :      16 :  1 2 9 12 18 19 33 38 43 46 47 54 58 64 85 87
         49 :      14 :  6 7 16 22 31 37 44 48 49 67 71 86 94 100
```

2. Problema de atribuição generalizada

https://people.brunel.ac.uk/~mastjjb/jeb/orlib/gapinfo.html

```
def atribuicao(m:int,n:int,bi:list,aij:list,cij:list):
    M = range(1, m + 1)  # Conjunto de funcionários
    N = range(1, n + 1)  # Conjunto de tarefas
```

```
    model = Model('Alocacao de Tarefas', solver_name=CBC)
    model.verbose = 0
    x = [[model.add_var(var_type=BINARY) for j in N] for i in M]

    model.objective =maximize( xsum(cij[i-1][j-1] * x[i-1][j-1] for i in M for j in N))
    for j in N:
        model += xsum(x[i-1][j-1] for i in M) == 1
    for i in M:
        model += xsum(aij[i-1][j-1] * x[i-1][j-1] for j in N) <= bi[i-1]
    status = model.optimize()

    if status == OptimizationStatus.OPTIMAL:
        print("Custo total de atribuição: {:12.2f}".format(model.objective_value))
        print("Atribuições:")
        for i in M:
            tarefas = []
            for j in N:
                if x[i-1][j-1].x >= 0.99:
                    tarefas.append(j)
            print(f"Funcionário {i} -> Tarefas: {', '.join(map(str, tarefas))}")
    else:
        print("Não foi encontrada uma solução ótima.")
```

```
def open_atri(file_list_path):
    with open(file_list_path) as file:
        filenames = file.readlines()
    for filename in filenames:
        print('\n',filename)
        with open(f'{filename.strip()}.txt', 'r') as file:
            lines = file.readlines()
        p = int(lines[0].strip().split()[0])
        index = 1
        for _ in range(p):
            if _ == 2: break # só mostrar as duas primeiras daa
            m, n = map(int, lines[index].strip().split())
            index += 1
            c_elements = []
            aij_elements = []
            while len(c_elements) < m * n:
                c_elements.extend(map(int, lines[index].strip().split()))
                index += 1
            while len(aij_elements) < n * m:
```

```
                aij_elements.extend(map(int, lines[index].strip().split()))
                index += 1
            c = [c_elements[i * n:(i + 1) * n] for i in range(m)]
            a = [aij_elements[i * n:(i + 1) * n] for i in range(m)]
            b = list(map(int, lines[index].strip().split()))
            index += 1
            atribuicao(m, n, b, a, c)
open_atri(r'gap\files.lst')
```

```
 gap1

Custo total de atribuição:       336.00
Atribuições:
Funcionário 1 -> Tarefas: 5, 7, 9, 13
Funcionário 2 -> Tarefas: 1, 2, 8
Funcionário 3 -> Tarefas: 4, 15
Funcionário 4 -> Tarefas: 3, 10, 11, 12
Funcionário 5 -> Tarefas: 6, 14
Custo total de atribuição:       327.00
Atribuições:
Funcionário 1 -> Tarefas: 2, 3, 6
Funcionário 2 -> Tarefas: 8, 10, 12
Funcionário 3 -> Tarefas: 4, 5, 7, 11
Funcionário 4 -> Tarefas: 1, 9, 13
Funcionário 5 -> Tarefas: 14, 15

 gap2

Custo total de atribuição:       434.00
Atribuições:
Funcionário 1 -> Tarefas: 3, 6, 9, 20
Funcionário 2 -> Tarefas: 7, 10, 16, 19
Funcionário 3 -> Tarefas: 1, 5, 13, 17
Funcionário 4 -> Tarefas: 4, 8, 15, 18
Funcionário 5 -> Tarefas: 2, 11, 12, 14
Custo total de atribuição:       436.00
Atribuições:
Funcionário 1 -> Tarefas: 5, 7, 20
Funcionário 2 -> Tarefas: 10, 11, 12, 13
Funcionário 3 -> Tarefas: 2, 6, 9, 19
Funcionário 4 -> Tarefas: 4, 8, 16, 17, 18
```

```
Funcionário 5 -> Tarefas: 1, 3, 14, 15

 gap3

Custo total de atribuição:       580.00
Atribuições:
Funcionário 1 -> Tarefas: 5, 8, 13, 15, 23
Funcionário 2 -> Tarefas: 10, 11, 20
Funcionário 3 -> Tarefas: 2, 7, 9, 18, 24
Funcionário 4 -> Tarefas: 3, 4, 14, 16, 19, 22
Funcionário 5 -> Tarefas: 1, 6, 12, 17, 21, 25
Custo total de atribuição:       564.00
Atribuições:
Funcionário 1 -> Tarefas: 11, 15, 16, 17, 18, 23
Funcionário 2 -> Tarefas: 2, 5, 7, 9
Funcionário 3 -> Tarefas: 1, 3, 6, 13, 19
Funcionário 4 -> Tarefas: 4, 8, 20, 22, 25
Funcionário 5 -> Tarefas: 10, 12, 14, 21, 24

 gap4

Custo total de atribuição:       656.00
Atribuições:
Funcionário 1 -> Tarefas: 7, 9, 15, 20, 28
Funcionário 2 -> Tarefas: 2, 5, 11, 13, 16, 17, 29
Funcionário 3 -> Tarefas: 1, 8, 18, 19, 24, 27
Funcionário 4 -> Tarefas: 6, 10, 14, 22, 23, 25
Funcionário 5 -> Tarefas: 3, 4, 12, 21, 26, 30
Custo total de atribuição:       644.00
Atribuições:
Funcionário 1 -> Tarefas: 4, 6, 8, 13, 26, 28
Funcionário 2 -> Tarefas: 5, 10, 12, 17, 25, 30
Funcionário 3 -> Tarefas: 1, 2, 7, 11, 20, 23
Funcionário 4 -> Tarefas: 9, 14, 18, 21, 24, 27
Funcionário 5 -> Tarefas: 3, 15, 16, 19, 22, 29

 gap5

Custo total de atribuição:       563.00
Atribuições:
Funcionário 1 -> Tarefas: 1, 2, 21
Funcionário 2 -> Tarefas: 6, 11, 18
Funcionário 3 -> Tarefas: 9, 24
```

```
Funcionário 4 -> Tarefas: 4, 7, 22
Funcionário 5 -> Tarefas: 5, 15, 17, 23
Funcionário 6 -> Tarefas: 3, 10, 16, 20
Funcionário 7 -> Tarefas: 12, 14, 19
Funcionário 8 -> Tarefas: 8, 13
Custo total de atribuição:       558.00
Atribuições:
Funcionário 1 -> Tarefas: 11, 16, 18
Funcionário 2 -> Tarefas: 8, 21, 24
Funcionário 3 -> Tarefas: 7, 10
Funcionário 4 -> Tarefas: 2, 5, 22, 23
Funcionário 5 -> Tarefas: 6, 12, 15
Funcionário 6 -> Tarefas: 1, 3, 13
Funcionário 7 -> Tarefas: 14, 17, 19
Funcionário 8 -> Tarefas: 4, 9, 20
```

gap6

```
Custo total de atribuição:       761.00
Atribuições:
Funcionário 1 -> Tarefas: 6, 15, 17, 19
Funcionário 2 -> Tarefas: 7, 16, 20, 23
Funcionário 3 -> Tarefas: 8, 22
Funcionário 4 -> Tarefas: 1, 5, 32
Funcionário 5 -> Tarefas: 14, 25, 29, 30
Funcionário 6 -> Tarefas: 3, 13, 24, 26, 27
Funcionário 7 -> Tarefas: 2, 4, 18, 28, 31
Funcionário 8 -> Tarefas: 9, 10, 11, 12, 21
Custo total de atribuição:       759.00
Atribuições:
Funcionário 1 -> Tarefas: 7, 25, 29
Funcionário 2 -> Tarefas: 2, 9, 16, 23, 30
Funcionário 3 -> Tarefas: 4, 6, 19, 24, 31
Funcionário 4 -> Tarefas: 12, 20, 22, 26
Funcionário 5 -> Tarefas: 1, 11, 13, 18, 28
Funcionário 6 -> Tarefas: 5, 10, 14, 32
Funcionário 7 -> Tarefas: 15, 21, 27
Funcionário 8 -> Tarefas: 3, 8, 17
```

gap7

```
Custo total de atribuição:       942.00
Atribuições:
Funcionário 1 -> Tarefas: 13, 27, 36
```