Universidade do Minho
Escola de Ciências

Relatório do trabalho de Investigação Operacional I

Trabalho realizado por:

Eduardo Dias A95467

Maria Cardoso A97169

Mariana Rodrigues A88599

Tomás Santos A91192

Mariana Lopes A96297

Exercício 1

a) Apresente um modelo de programação inteira, explicitando o significado das variáveis de decisão, da função objetivo e de cada grupo de restrições no contexto do problema.

Variáveis de decisão:

$$y_i = \begin{cases} 1, se\ é\ colocado\ um\ veículo\ no\ local\ i;\ i = 0, 1, \dots, n \\ 0, \quad caso\ contrário \end{cases}$$

$$x_{ij} = \begin{cases} 1, se\ o\ veículo\ colocado\ no\ local\ i\ é\ o\ mais\ próximo \\ do\ local\ de\ potencial\ ignição\ j;\ i = 0,1, \dots, n;\ j = n, n+1, \dots, n+m-1 \\ 0, \quad caso\ contrário \end{cases}$$

$$d_{ij} : distância\ entre\ o\ local\ i\ e\ o\ local\ de\ ignição\ j;\ i = 0, 1, \dots, n-1;\ j = n, n+1, \dots, n+m-1$$

Modelo:

$$Min\ Z = \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij}\, x_{ij}$$

O objetivo é minimizar Z que é a distância total dos locais de potencial ignição ao veículo mais próximo.

Sujeito a:

$$\sum_{i=1}^{n} y_i = p$$

O número de locais selecionados tem que ser igual ao número de veículos disponíveis.

$$\sum_{i=1}^{n} x_{ij} = 1;\ \forall j$$

Só pode existir um veículo mais próximo de cada um dos j locais onde podem ocorrer ignições.

$x_{ij} \leq y_i;\ \forall ij$

Um veículo colocado no local i só pode ser considerado o mais próximo se for um dos locais selecionados para colocar um veiculo.

$y_i \in \{0,1\}, \forall i$

$x_{ij} \in \{0,1\}, \forall ij$

b) Implemente o modelo em Python/Gurobi

```python
#1/2/3 b)
import gurobipy as gp
from gurobipy import GRB

import random
import math

import networkx as nx
import matplotlib.pyplot as plt
```

Em primeiro lugar são importadas as bibliotecas.

```python
def getData(num_aluno,n,m,p):

    '''
    Recebe:
    n - número de potenciais localizações para veículos
    m - número de localizações onde podem ocorrer ignições
    p - número de veículos

    Retorna:
    coord - coordenadas de cada localização, indices <n de veículos, índices >=n e <m ignições
    d - distancia entre cada par de potenciais localizações para veículos e ignições
    r - risco de cada local


    '''
    assert n>0
    assert m>0
    assert p>0

    random.seed(num_aluno)

    ###################################
    #random.seed(num_aluno)
    ###################################

    ###################################
    # Coordenadas
    ###################################
    '''
    Os primeiros n locais (índices de 0 a n-1) dizem respeito a veículos.
    Os últimos m locais (índices de n a n+m-1) dizem respeito a ignições.
    '''
    coord={}
    for k in range(n+m):
        coord[k]=(random.random(),random.random())


    ###################################
    # distâncias entre todos os pares
    ###################################
    d = {}
    for i in range(n):
        for j in range(m):
            d[(i,n+j)] = math.sqrt((coord[i][0]-coord[n+j][0])**2 + (coord[i][1]-coord[n+j][1])**2)

    ###################################
    # risco (questão 3)
    ###################################

    r=[random.randrange(5)+1 for i in range (m)]

    return coord, d, r
```

Seguidamente temos uma das funções fornecidas pelo docente a qual permite que, indicando o número de potenciais localizações para veículos (n), o número de localizações onde podem ocorrer ignições (m) e o número de veículos (p), nos seja retornado as coordenadas de cada localização, veículo e ignição bem como a distancia entre cada par de potenciais localizações para veículos e ignições (d) e o risco de cada local (r, necessário para o exercício número 3)

```
def visualizeSolution(aSelectedLocations, aNonSelectedLocations, aCoveredPoints,
                      aNonCoveredPoints, aAssignment, aPos):
    '''
    aSelectedLocations - índices das instalações seleccionadas
    aNonSelectedLocations - índices das instalações não seleccionadas
    aCoveredPoints - índices dos pontos seleccionados
    aNonCoveredPoints - índices dos pontos não seleccionados
    aAssignment - lista dos pares (i,j) em que i é uma localização seleccionada e j um ponto por ela coberto.
    aPos  - coordenadas dos locais como retornadas pela função getData()
    '''

    plt.figure()
    G = nx.Graph()
    G.add_nodes_from(aSelectedLocations)
    G.add_nodes_from(aNonSelectedLocations)
    for (i,j) in aAssignment: G.add_edge(i,j)


    nx.draw_networkx_nodes(G, aPos, node_color='green', nodelist=aSelectedLocations, node_size=100)
    nx.draw_networkx_nodes(G, aPos, node_color='grey', nodelist=aNonSelectedLocations, node_size=50)
    nx.draw_networkx_nodes(G, aPos, node_color='blue', nodelist=aCoveredPoints, node_size=10)
    nx.draw_networkx_nodes(G, aPos, node_color='red', nodelist=aNonCoveredPoints, node_size=50)
    nx.draw_networkx_edges(G, aPos, edgelist=aAssignment)


    plt.title("Localização", size=10)
    plt.axis("off")
    plt.savefig("figure.jpg", dpi=300)
    plt.show()
```

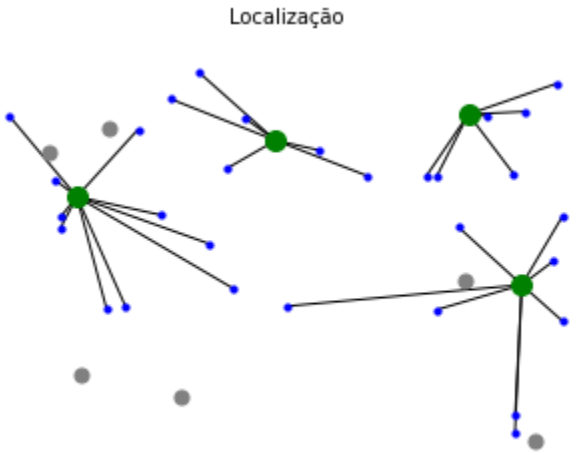
A segunda função fornecida pelo docente (vizualizesolution) permite que, indicando os locais selecionados e não selecionados, os pontos cobertos e não cobertos, a que veículo está associado cada potencial local de ignição e as coordenadas obtidas com a função getdata, se obtenha a representação gráfica da solução obtida.

Todo o código apresentado até agora será utilizado na resolução das três questões, porém, não voltará a ser mostrado na resolução do exercício 2 b) nem na resolução do exercício 3 b).

```python
#1 c)
try:

    num_aluno=97169

    n=10 # número de potenciais localizações para veículos
    m=30 # número de localizações onde podem ocorrer ignições
    p=4 # número de veículos

    # coord - coordenadas de cada local
    # d - distancia entre cada par de potenciais localizações para veículos e ignições
    # r - risco de cada local

    coord, d, r = getData(num_aluno, n,m,p)

    model=gp.Model('trabIO')

    # model.addVars
    y = model.addVars(n, vtype=GRB.BINARY, name='y')
    x = model.addVars(n, m, vtype = GRB.BINARY, name='x')
    # model.setObjective
    model.setObjective(sum(d[i,n+j]*x[i,j] for i in range(n) for j in range(m)), GRB.MINIMIZE)
    # model.addConstrs
    model.addConstr(((sum(y[i] for i in range(n))) == p))
    model.addConstrs((sum(x[i,j] for i in range(n))==1) for j in range(m))
    model.addConstrs((x[i,j] <= y[i]) for i in range(n) for j in range (m))

    model.setParam('TimeLimit', 10) # in seconds
    model.setParam('MIPGap', 1e-4) # default 1e-4

    model.update()

    model.write("modelo.lp")

    model.optimize()

    print(model.Status) #2-optimal

    aSelectedLocations = [i for i in y if y[i].x == 1]
    aNonSelectedLocations = [i for i in y if i not in aSelectedLocations]
    aCoveredPoints = [j+n for j in range(m) if sum(x[i,j].x for i in range(n)) == 1]
    aNonCoveredPoints = [j+n for j in range(m) if j+n not in aCoveredPoints]
    aAssignment = [(i,j+n) for (i,j) in x if x[i,j].x == 1]
    aPos = coord
    visualizeSolution(aSelectedLocations, aNonSelectedLocations, aCoveredPoints, aNonCoveredPoints, aAssignment, aPos)

    print("Optimal value=", model.ObjVal)
    print("Selected facilities:", aSelectedLocations)
    print("Assigment:", aAssignment)

except gp.GurobiError as e:
    print('Error code ' + str(e.errno) + ': ' + str(e))

except AttributeError:
    print('Encountered an attribute error')
```

Aqui é possível observar o código específico da questão 1 c) sendo que basta alterar os valores para n, m e p para se obter também uma solução para a questão 1 d).

c) Para n=10, m=30 e p=4, obtenha uma solução ótima, apresente a sua representação gráfica e discuta a sua razoabilidade.

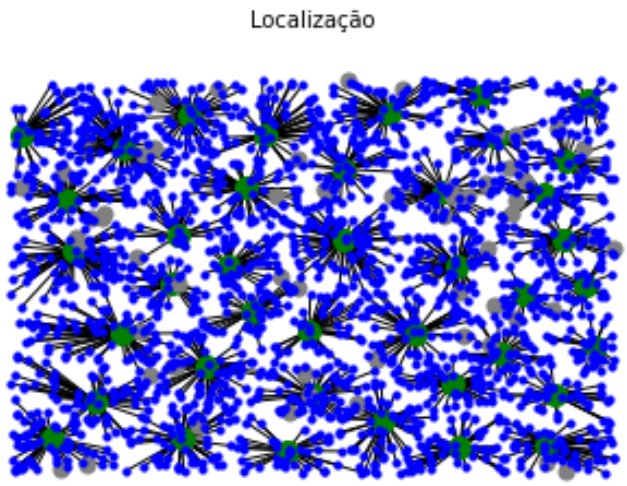


```
Optimal value= 5.631478002119752
Selected facilities: [0, 2, 5, 6]
Assigment: [(0, 12), (0, 13), (0, 19), (0, 32), (0, 35), (0, 39), (2, 10), (2, 17), (2,
22), (2, 23), (2, 24), (2, 25), (2, 27), (2, 30), (5, 20), (5, 29), (5, 31), (5, 36),
(5, 37), (5, 38), (6, 11), (6, 14), (6, 15), (6, 16), (6, 18), (6, 21), (6, 26), (6,
28), (6, 33), (6, 34)]
```

Como se pode observar na imagem e no gráfico, foram selecionados 4 locais dos 10 possíveis para colocar os veículos (0,2,5,6), como pedido no enunciado. Podemos também ver que todos os possíveis pontos de ignição estão devidamente associados ao veículo num dos locais selecionados que se encontra mais perto, comprovando a razoabilidade das restrições.

O objetivo era minimizar a distância total dos locais de potencial ignição ao veículo mais próximo sendo que o valor da solução ótima obtida foi de 5.631478002119752.

d) Para n=100, m=2000 e p=40, obtenha uma solução ótima, apresente a sua representação gráfica e discuta a sua razoabilidade.




```
Optimal value= 127.751519046293
Selected facilities: [3, 8, 10, 11, 14, 15, 16, 17, 18, 20, 21, 22, 24, 29, 32, 34, 36,
38, 39, 45, 50, 52, 55, 56, 57, 62, 63, 65, 67, 68, 69, 79, 80, 81, 82, 83, 86, 91, 96,
97]
```

Observando a imagem e o gráfico podemos ver que foram selecionados 40 locais para se colocar os veículos e os pontos de possível ignição aparentam estar associados aos veículos mais próximo tal como era pedido.

A solução ótima obtida foi 127.751519046293. Como esperado o valor comparado com o da alínea anterior é bem superior pois colocando mais veículos e havendo mais pontos para cobrir é normal que a distância total dos locais de potencial ignição mais próximos aumente.

Exercício 2

a) Apresente um modelo de programação inteira mista, explicitando o significado das variáveis de decisão, da função objetivo e de cada grupo de restrições no contexto do problema.

Variáveis de decisão:

$$y_i = \begin{cases} 1, se\ é\ colocado\ um\ veículo\ no\ local\ i;\ i = 0, 1, \dots, n-1 \\ 0, \quad caso\ contrário \end{cases}$$

$$x_{ij} = \begin{cases} 1, se\ o\ veículo\ colocado\ no\ local\ i\ é\ o\ mais\ próximo \\ do\ local\ de\ potencial\ ignição\ j;\ i = 0, 1, \dots, n-1; \\ j = n, n+1, \dots, n+m-1 \\ 0, \quad caso\ contrário \end{cases}$$

d_{ij} : *distância entre o local* i *e o local de ignição* j; $i = 0, 1, \dots, n-1$; $j = n, n+1 \dots, n+m-1$

Z : *maior distância entre todas as distâncias dos locais de ignição ao veículo que lhe está mais próximo*

Modelo:

$$Min\ Z$$

O objetivo é minimizar a maior distância de entre todas as distâncias dos locais de ignição ao veículo que lhe está mais próximo.

Sujeito a:

$$\sum_{i=1}^{n} y_i = p$$

O número de locais selecionados tem que ser igual ao número de veículos disponíveis.

$$\sum_{i=1}^{n} x_{ij} = 1;\ \forall j$$

Só pode existir um veículo mais próximo de cada um dos j locais onde podem ocorrer ignições.

$x_{ij} \leq y_i;\ \forall ij$

Um veículo colocado no local i só pode ser considerado o mais próximo se for um dos locais selecionados para colocar um veiculo.

$Z \geq d_{ij} * x_{ij}; \forall ij$

Z é a maior distância de entre todas as distâncias dos locais de ignição ao veículo que lhe está mais próximo

$Z \geq 0$

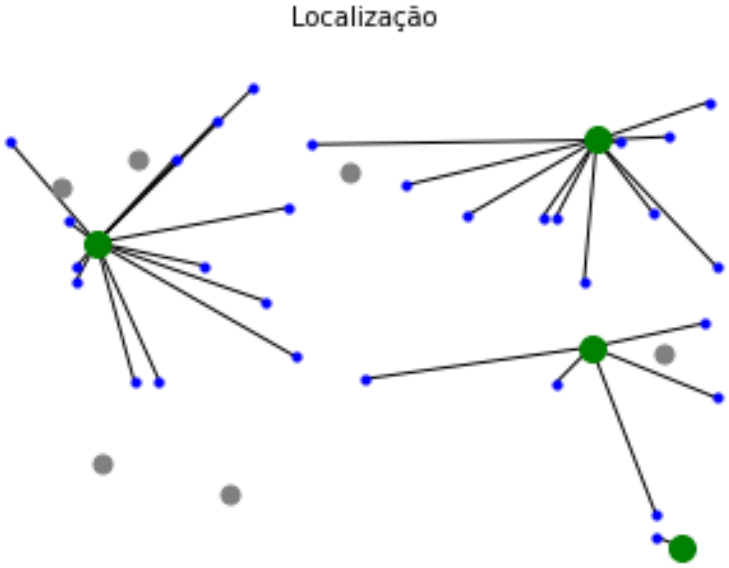
$y_i \in \{0,1\}, \forall i$

$x_{ij} \in \{0,1\}, \forall ij$

b) Implemente o modelo em Python/Gurobi

```python
try:
    num_aluno=97169

    n=10 # número de potenciais localizações para veículos
    m=30 # número de localizações onde podem ocorrer ignições
    p=4 # número de veiculos

    # coord - coordenadas de cada local
    # d - distancia entre cada par de potenciais localizações para veículos e ignições
    # r - risco de cada local

    coord, d, r = getData(num_aluno, n,m,p)

    model=gp.Model('trabIO')

    # model.addVars
    y = model.addVars(n, vtype=GRB.BINARY, name='y')
    x = model.addVars(n, m, vtype = GRB.BINARY, name='x')
    z = model.addVar(vtype = GRB.CONTINUOUS, name='z')
    # model.setObjective
    model.setObjective(z, GRB.MINIMIZE)
    # model.addConstrs
    model.addConstr(((sum(y[i] for i in range(n))) == p))
    model.addConstrs((sum(x[i,j] for i in range(n))==1) for j in range(m))
    model.addConstrs((x[i,j] <= y[i]) for i in range(n) for j in range (m))
    model.addConstrs((z >= d[i,n+j]*x[i,j]) for i in range(n) for j in range(m))

    model.setParam('TimeLimit', 10) # in seconds
    model.setParam('MIPGap', 1e-4) # default 1e-4

    model.update()

    model.write("modelo.lp")

    model.optimize()

    print(model.Status) #2-optimal

    aSelectedLocations = [i for i in y if y[i].x == 1]
    aNonSelectedLocations = [i for i in y if i not in aSelectedLocations]
    aCoveredPoints = [j+n for j in range(m) if sum(x[i,j].x for i in range(n)) == 1]
    aNonCoveredPoints = [j+n for j in range(m) if j+n not in aCoveredPoints]
    aAssignment = [(i,j+n) for (i,j) in x if x[i,j].x == 1]
    aPos = coord
    visualizeSolution(aSelectedLocations, aNonSelectedLocations, aCoveredPoints, aNonCoveredPoints, aAssignment, aPos)

    print("Optimal value=", model.ObjVal)
    print("Selected facilities:", aSelectedLocations)
    print("Assigment:", aAssignment)

except gp.GurobiError as e:
    print('Error code ' + str(e.errno) + ': ' + str(e))

except AttributeError:
    print('Encountered an attribute error')
```

c) *Para n=10, m=30 e p=4, obtenha uma solução ótima, apresente a sua representação gráfica e discuta a sua razoabilidade. Identifique o local de ignição que determina o valor da solução ótima. Compare a solução obtida com a solução obtida na questão 1c.*



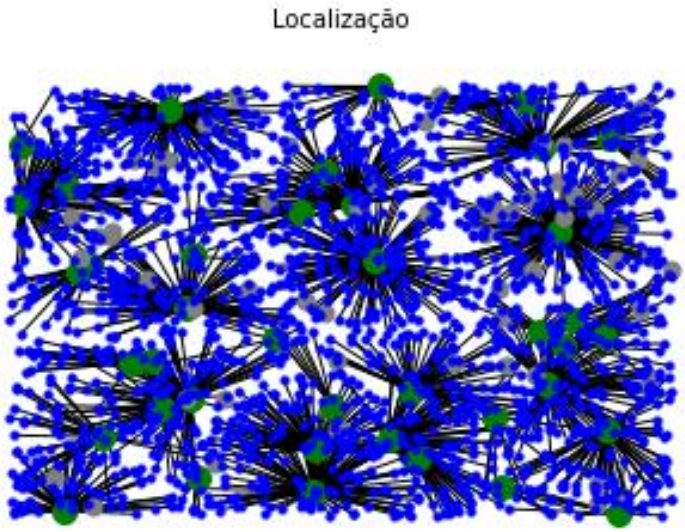

```
Optimal value= 0.38857570248514356
Selected facilities: [0, 1, 6, 7]
Assigment: [(0, 12), (0, 13), (0, 19), (0, 23), (0, 29), (0, 30), (0, 31), (0, 32), (0,
35), (0, 38), (0, 39), (1, 10), (1, 17), (1, 22), (1, 24), (1, 25), (6, 11), (6, 14),
(6, 15), (6, 16), (6, 18), (6, 20), (6, 21), (6, 26), (6, 28), (6, 33), (6, 34), (6,
36), (6, 37), (7, 27)]
```

Como se pode observar na imagem e no gráfico, foram selecionados 4 locais dos 10 possíveis para colocar os veículos, como pedido no enunciado. Podemos também ver que todos os possíveis pontos de ignição estão devidamente associados ao veículo num dos locais selecionados que se encontra mais perto, comprovando a razoabilidade das restrições.

O valor ótimo obtido foi de 0.38857570248514356.

Enquanto que no exercício1 c) os locais escolhidos para colocar os veículos foram o 0,2,5 e 6 neste foram selecionados os locais 0,1,6 e 7 tendo, portanto, sido alterados dois dos locais selecionados. Se observarmos com atenção os gráficos das duas questões podemos verificar que os dois locais que foram selecionados nesta questão em relação à outra permitem minimizar a distância ao ponto de potencial ignição que se encontra no centro do gráfico (ligeiramente para baixo) bem como a distância do ponto azul que se encontra no canto inferior direito o que tem toda a lógica pois nesta questão pretendia-se minimizar a maior distância de entre todas as distâncias dos locais de ignição ao veículo que lhe está mais próximo

d) Para n=100, m=2000 e p=40, obtenha uma solução ótima, apresente a sua representação gráfica e discuta a sua razoabilidade.




```
Optimal value= 0.17642818632958987
Selected facilities: [1, 2, 3, 4, 5, 7, 11, 14, 15, 18, 21, 26, 35, 55, 58, 62, 63, 66,
70, 73, 77, 79, 80, 81, 82, 83, 85, 86, 87, 88, 89, 91, 92, 93, 94, 95, 96, 97, 98, 99]
```

Observando a imagem e o gráfico podemos ver que, tal como requerido, foram selecionados 40 locais para se colocar os veículos.

A solução ótima obtida foi 0.17642818632958987. Comparando o valor c com o da alínea anterior podemos ver que este diminuiu, pois, mesmo havendo 2000 pontos para cobrir os 40 veículos posicionados permitem que a maior distância de entre todas as distâncias dos locais de ignição ao veículo que lhe está mais próximo diminua.

É de denotar que a operação é muito demorada, como tal, aumentamos o tempo limite para 10000 ("model.setParam('TimeLimit', 10000)") de forma a conseguir que o status do modelo fosse 2 (modelo resolvido na otimalidade).

Exercício 3

a) Apresente um modelo de programação inteira, explicitando o significado das variáveis de decisão, da função objetivo e de cada grupo de restrições no contexto do problema.

Variáveis de decisão:

$$y_i = \begin{cases} 1, se\ é\ colocado\ um\ veículo\ no\ local\ i;\ i = 0, 1, \dots, n-1 \\ 0, \qquad caso\ contrário \end{cases}$$

$$x_{ij} = \begin{cases} 1, se\ o\ veículo\ colocado\ no\ local\ i\ é\ o\ mais\ próximo \\ do\ local\ de\ potencial\ ignição\ j;\ i = 0, 1, \dots, n-1; \\ j = n, n+1, \dots, n+m-1, \\ 0, \qquad caso\ contrário \end{cases}$$

$$d_{ij} : distância\ entre\ o\ local\ \ i\ e\ o\ local\ de\ ignição\ j;\ i = 0, 1, \dots, n-1;\ j = n, n+1, \dots, n+m-1$$

$$R_j : indíce\ de\ risco\ de\ incêndio\ no\ local\ de\ ignição\ j;\ j = n, n+1, \dots, n+m-1$$

Modelo:

$$Max\ Z = \sum_i^n \sum_j^m R_j * x_{ij}$$

O objetivo é maximizar a soma dos índices de risco dos locais cobertos.

Sujeito a:

$$\sum_{i=1}^{n} y_i = p$$

O número de locais selecionados tem que ser igual ao número de veículos disponíveis.

$$\sum_{i=1}^{n} x_{ij} \leq 1\ ;\ \forall j$$

Só pode existir, no máximo, um veículo mais próximo de cada um dos j locais onde pode ocorrer ignição.

$$d_{ij} * x_{ij} < 0.3; \forall ij$$

O local só está coberto se a distância entre o local i e o local j for menor que 0.3.

$x_{ij} \leq y_i;\ \forall ij$

Um veículo colocado no local i só pode ser considerado o mais próximo se for um dos locais selecionados para colocar um veiculo.

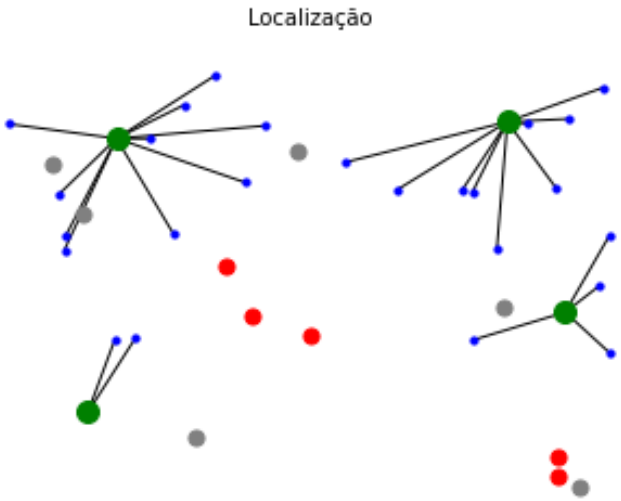
$y_i \in \{0,1\}, \forall i$

$x_{ij} \in \{0,1\}, \forall ij$

b) Implemente o modelo em Python/Gurobi.

```
try:
    num_aluno=97169

    n=10 # número de potenciais localizações para veículos
    m=30 # número de localizações onde podem ocorrer ignições
    p=4 # número de veiculos

    # coord - coordenadas de cada local
    # d - distancia entre cada par de potenciais localizações para veículos e ignições
    # r - risco de cada local

    coord, d, r = getData(num_aluno, n,m,p)

    model=gp.Model('trabIO')

    # model.addVars
    y = model.addVars(n, vtype=GRB.BINARY, name='y')
    x = model.addVars(n, m, vtype = GRB.BINARY, name='x')
    # model.setObjective
    model.setObjective(sum(x[i,j]*r[j] for i in range(n) for j in range(m)), GRB.MAXIMIZE)
    # model.addConstrs
    model.addConstr(((sum(y[i] for i in range(n))) == p))
    model.addConstrs((sum(x[i,j] for i in range(n))<=1) for j in range(m))
    model.addConstrs((x[i,j] <= y[i]) for i in range(n) for j in range (m))
    model.addConstrs((d[i,n+j]*x[i,j]<=0.3) for i in range(n) for j in range(m))

    model.setParam('TimeLimit', 10) # in seconds
    model.setParam('MIPGap', 1e-4) # default 1e-4

    model.update()

    model.write("modelo.lp")

    model.optimize()

    print(model.Status) #2-optimal

    aSelectedLocations = [i for i in y if y[i].x == 1]
    aNonSelectedLocations = [i for i in y if i not in aSelectedLocations]
    aCoveredPoints = [j+n for j in range(m) if sum(x[i,j].x for i in range(n)) == 1]
    aNonCoveredPoints = [j+n for j in range(m) if j+n not in aCoveredPoints]
    aAssignment = [(i,j+n) for (i,j) in x if x[i,j].x == 1]
    aPos = coord
    visualizeSolution(aSelectedLocations, aNonSelectedLocations, aCoveredPoints, aNonCoveredPoints, aAssignment, aPos)

    print("Valor da soma dos indices de risco se todos os locais de potencial ignição se encontrassem cobertos", sum(r))
    print("Optimal value=", model.ObjVal)
    print("Selected facilities:", aSelectedLocations)
    print("Assigment:", aAssignment)

except gp.GurobiError as e:
    print('Error code ' + str(e.errno) + ': ' + str(e))

except AttributeError:
    print('Encountered an attribute error')
```

c) Para n=10, m=30 e p=4, obtenha uma solução ótima, apresente a sua representação gráfica e discuta a sua razoabilidade.



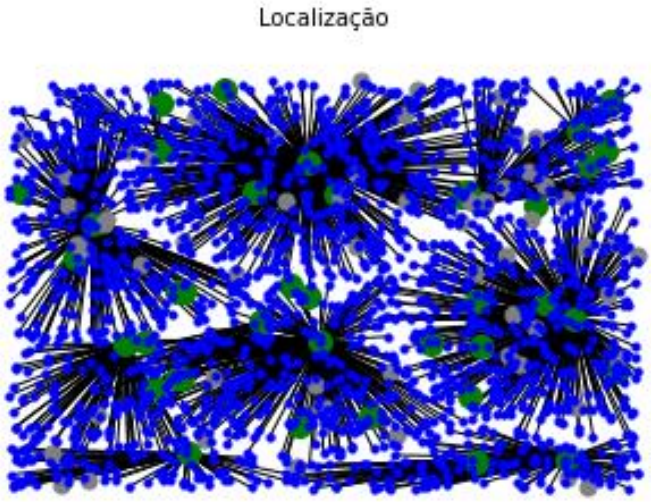

```
Valor da soma dos indices de risco se todos os locais de potencial ignição se
encontrassem cobertos 91
Optimal value= 79.0
Selected facilities: [0, 2, 3, 8]
Assigment: [(0, 12), (0, 13), (0, 19), (0, 23), (0, 29), (0, 31), (0, 32), (0, 35), (0,
39), (2, 17), (2, 22), (2, 25), (2, 30), (3, 18), (3, 26), (8, 11), (8, 15), (8, 16),
(8, 20), (8, 21), (8, 28), (8, 33), (8, 36), (8, 37), (8, 38)]
```

Como se pode observar na imagem e no gráfico, foram selecionados 4 locais dos 10 possíveis para colocar os veículos (0,2,3,8). Podemos também observar que quase todos os pontos de potencial ignição se encontram cobertos (com um veículo a menos de 0.3) sendo que apenas 5 não estão cobertos.

O valor da solução ótima obtida é 79.0. Ou seja, a soma dos índices de risco dos locais cobertos, colocando os veículos nestes locais, é 79 de 91 possíveis (calculado através do comando sum(r))

d) Para n=100, m=2000 e p=40, obtenha uma solução ótima, apresente a sua representação gráfica e discuta a sua razoabilidade.

```
Valor da soma dos indices de risco se todos os locais de potencial ignição se
encontrassem cobertos 5899
Optimal value= 5899.0
Selected facilities: [2, 5, 6, 10, 14, 18, 19, 21, 24, 25, 26, 28, 35, 36, 37, 39, 40,
48, 49, 50, 51, 55, 60, 63, 65, 68, 75, 77, 78, 79, 82, 83, 84, 87, 89, 90, 93, 94, 95,
98]
```



Observando a imagem e o gráfico podemos ver que, tal como requerido, foram selecionados 40 locais para se colocar os veículos.

A solução ótima obtida foi 5899.0. Como esperado o valor comparado com o da alínea anterior é bem superior pois colocando mais veículos e havendo mais pontos cobrir, haverá obrigatoriamente mais pontos coberto e consequentemente soma dos índices de risco dos locais cobertos aumenta. Este também é o valor máximo possível para a soma dos índices de risco dos locais cobertos (calculado usando o comando sum (r)), ou seja, todos os locais se encontram cobertos.

e) Compare os resultados das alíneas c) e d) com as soluções obtidas para valores de distância máxima 0.2 e 0.4.

Comparação com a alínea c)

Para 0.2:

```
Valor da soma dos indices de risco se todos os locais de potencial ignição se
encontrassem cobertos 91
Optimal value= 64.0
Selected facilities: [0, 2, 4, 5]
Assigment: [(0, 12), (0, 13), (0, 19), (0, 32), (0, 35), (0, 39), (2, 17), (2, 22), (2,
23), (2, 25), (2, 30), (4, 15), (4, 16), (4, 28), (4, 33), (5, 29), (5, 31), (5, 36),
(5, 38)]
```

Como seria de esperar, diminuindo a distância máxima para a qual um veículo cobre um potencial local de ignição, para além de a solução ser diferente (0,2,4,5 sendo que em c) era 0,2,3,8), há menos pontos cobertos (19 contra 25 na alínea c))e como tal a soma dos índices de risco dos locais cobertos é menor (61 contra 74 na alínea c)).

Para 0.4:

```
Valor da soma dos indices de risco se todos os locais de potencial ignição se
encontrassem cobertos 91
Optimal value= 91.0
Selected facilities: [0, 2, 3, 4]
Assigment: [(0, 13), (0, 29), (0, 35), (0, 39), (2, 12), (2, 17), (2, 19), (2, 22), (2,
23), (2, 24), (2, 25), (2, 27), (2, 30), (2, 31), (2, 32), (3, 10), (3, 14), (3, 18),
(3, 21), (3, 26), (4, 11), (4, 15), (4, 16), (4, 20), (4, 28), (4, 33), (4, 34), (4,
36), (4, 37), (4, 38)]
```

Com a distância máxima para a qual um veículo sobre um potencial local de ignição a ser aumentada para 0.4 é de prever que o número de locais de potencial ignição cobertos aumente. Isto é comprovado na imagem acima onde o valor ótimo para a soma dos índices de risco dos locais cobertos aumentou para 91 que é o valor máximo possível para a soma dos índices de risco dos locais cobertos significando que todos os pontos de potencial ignição se encontram cobertos.

É de denotar também que a solução é diferente da solução da alínea c) (0,2,3,4 sendo que em c) é 0,2,3,8).

Comparação com a alínea d)

Para 0.2:

```
Valor da soma dos indices de risco se todos os locais de potencial ignição se
encontrassem cobertos 5899
Optimal value= 5899.0
Selected facilities: [0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 29, 31, 32, 34, 35, 37, 40, 50, 52, 55, 57, 58, 62, 69, 71]
```

Reduzindo o valor da distância máxima para a cobertura de um local de potencial ignição seria expectável que o valor da solução ótima diminuísse também pois esta alteração provavelmente diminuiria o número de locais de potencial ignição cobertos, porém, mesmo a solução sendo completamente diferente o valor ótimo é exatamente igual: 5899.0 (valor máximo possível).

Para 0.4:

```
Valor da soma dos indices de risco se todos os locais de potencial ignição se
encontrassem cobertos 5899
Optimal value= 5899.0
Selected facilities: [2, 5, 6, 10, 14, 18, 19, 21, 24, 25, 26, 28, 35, 36, 37, 39, 40,
48, 49, 50, 51, 55, 60, 63, 65, 68, 75, 77, 78, 79, 82, 83, 84, 87, 89, 90, 93, 94, 95,
98]
```

Aumentando o valor da distância máxima para a cobertura de um local de potencial ignição é previsível que o valor ótimo se mantenha igual pois se a soma dos índices de risco dos locais cobertos era máxima com o valor de distância máxima de cobertura de um local de potencial ignição igual a 0.3 então passando a 0.4 os pontos continuarão todos coberto como é comprovado pela imagem acima onde vemos que a solução bem como o valor ótimo são exatamente os mesmo da alínea d).