IDO- Tarea 1

Marcelino Sánchez Rodríguez 191654

2024-04-19

Modelo en python

```python
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt


def construct_graph_and_find_shortest_path(cost_matrix):
    #Creamos el grafo
    G = nx.Graph()

    # Obtenemos el número de nodos
    n = cost_matrix.shape[0]

    for i in range(n):
        #Añade los nodos
        G.add_node(i)

    for i in range(n):
        for j in range(n):
            # Checamos si hay conexión entre los nodos de i y j
            if cost_matrix[i, j] != -1:
                #Añade la arista
                G.add_edge(i, j, weight=cost_matrix[i, j])

                print(f"Weight of {i},{j} edge: {cost_matrix[i, j]}")

    path_length, path = nx.single_source_dijkstra(G, source=0, target=n-1)
    return G, path_length, path
```

Ejemplo de uso 2

```
#Problema 2 de modelado

B = 100.0

p = np.array([20.0, 50.0, 20.0,130.0])

alpha = np.array([ .15, .3, .2,.1 ])

f = np.array([[0.0, 10.0, 20.0, 30.0, 50.0],
            [0.0, 0.0, 5.0, 10.0, 35.0],
            [0.0, 0.0, 0.0, 5.0, 20.0],
            [0.0, 0.0, 0.0, 0.0, 10.0],
            [0.0, 0.0, 0.0, 0.0, 0.0]])

g = np.array([[0.0, 1.0, 2.0, 3.0, 15.0],
            [0.0, 0.0, 1.0, 3.0, 8.0],
            [0.0, 0.0, 0.0, 20.0, 5.0],
            [0.0, 0.0, 0.0, 0.0, 1.0],
            [0.0, 0.0, 0.0, 0.0, 0.0]])

cost_matrix = np.full((5, 5), -1.0)

def funcionB(i):
    biAux = B
    for k in range(0, i):
                biAux = biAux*(1-alpha[k])
    return biAux

# Recorremos la matriz
for i in range(g.shape[0]):  # Recorremos las filas
    for j in range(g.shape[1]):  # Recorremos las columnas
        if g[i][j] != 0.0:
            # Definimos B(i)
            biAux = funcionB(i)
            cost_matrix[i][j] = biAux*(sum(p[i:j]) + g[i][j]) + f[i][j]




G, path_length, path = construct_graph_and_find_shortest_path(cost_matrix)
print(f"Shortest path: {path} with length {path_length}")
```

```
Weight of 0,1 edge: 2110.0
Weight of 0,2 edge: 7220.0
Weight of 0,3 edge: 9330.0
Weight of 0,4 edge: 23550.0
Weight of 1,2 edge: 4340.0
Weight of 1,3 edge: 6215.0
Weight of 1,4 edge: 17715.0
Weight of 2,3 edge: 2384.9999999999995
Weight of 2,4 edge: 9242.499999999998
Weight of 3,4 edge: 6245.599999999999
Shortest path: [0, 1, 3, 4] with length 14570.599999999999
```

Gráfica

```
# Suponiendo que ya tienes el gráfico G y el camino más corto path calculado

# Dibuja el gráfico completo
pos = nx.spring_layout(G)  # Genera posiciones de nodos
nx.draw(G, pos, with_labels=True, node_color='lightblue')

# Dibuja el camino más corto con aristas rojas
edges_in_path = [(path[i], path[i+1]) for i in range(len(path)-1)]
nx.draw_networkx_edges(G, pos, edgelist=edges_in_path, edge_color='r', width=2)

# Opcional: Dibuja las flechas para indicar la dirección del camino
for i in range(len(path)-1):
    start_pos = pos[path[i]]
    end_pos = pos[path[i+1]]
    plt.arrow(start_pos[0], start_pos[1], end_pos[0] - start_pos[0], end_pos[1] - start_po
              head_width=0.05, head_length=0.1, fc='r', ec='r', length_includes_head=True)
```

```
plt.show()
```