Experimenting with Spatial Networks

To the best of my knowledge, there is no good library to work with spatial networks. `networkx` has functions to generate random spatial graphs (called geometric graphs https://networkx.org/documentation/stable/reference/generators.html), but they are not designed to fit deterrence functions, and are not adapted to work with geographical coordinates.

It can be useful to work on a smaller network for experimenting (you can go back to the original graph when your functions are ready) To work on a subnetwork composed of the largest nodes, you can do for instance:

Listing 1: Filtering out low degree nodes

```
to_keep = [n for n,v in g.degree() if v > 10]
g = g.subgraph(to_keep)
```

1. Route Factor/ Accessibility
   (a) Choose an origin airport, and compute the **great circle distance (haversine)** from it to all the other nodes. Check the results. You can use the `haversine` function from `haversine` package. As previously, you can obtain the coordinates of airports with something like:
   `coordinates = {n:(g.nodes[n]["lat"],g.nodes[n]["lon"]) for n in g.nodes}`
   (b) For all edges in the graph, compute its haversine length, and add this information as a parameter to the graph. You can use for instance `nx.set_edge_attributes`
   (c) Compute the Route Distance from the chosen node to all others. You can use `nx.shortest_path_length` (don't forget to choose the right attribute !).
   (d) You can now compute the Route Factor of pairs of nodes, and the Node Accessibility score of the chose node.
   (e) Compare between some nodes (large/small, occupying a central position or not...)
   Check the largest and smallest values, and comment them.
2. Influence of distance in the airport dataset

   When studying a network with spatial information, a first step to check if it can indeed be considered as a spatial network is to compute its deterrence function.

   (a) Plot the distribution of edges length. To get a proper result, you should have bins which are not too wide (missing information) or to narrow (overfit). Is it enough to know if the network has as spatial structure, and the nature of the bias ?
   (b) Plot the distribution of pairs of nodes distance. Is it flat ? why ?
   (c) Draw the spatial graph, for instance using something like:
   `nx.draw_networkx(self.graph,pos=self.coordinates,node_size=5,edge_color="grey",with_labels=False)`
   (d) To compute the bias to observe an edge or not relatively to the distance, you can compute, for each bin, the number of existing edges over the number of possible edges. Plot it. Now you can conclude on the nature of the spatial effect.
   (e) Based on the result of the previous question, generate a spatial graph with a spatial bias similar (approximated at bin resolution) to the observed one. Remember that what we computed in the

previous question can be interpreted as a density, i.e., a probability to observe an edge according to the distance. (You can compare a random value with this probability for each pair of node, for instance, or pick an exact number of edges using `random.choices`.

(f) Plot this graph and compare its properties to the original one (e.g., clustering, average distance...)

3. Going further: Gravity Model

(a) Compute a randomized version of the network using the Gravity Model. Don't forget to check that the resulting network has all the desired properties: same node positions, (approximately) similar degrees, (approximately) similar deterrence function.

(b) Compare network properties with the previous networks

(c) Create a new network by substracting the probability to observe an edge according to the gravity model to the edges in the graph. You can now obtain a much richer view of your network: to each pair of node is associated a value that can be interpreted as the exceptional character of its existence or non-existence. What are the most surprising missing edges, and most suprising edges?