```
import math
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

random.seed(10)
class Node:
    def __init__(self, x, y, frequency):
        self.frequency = frequency
        self.x = x
        self.y = y
        self.interference = 0

    def update_interference(self, value):
        self.interference = value


class Graph:
    def __init__(self, list_nodes):
        self.adjacency_list = []
        self.list_nodes = list_nodes

        for node in list_nodes:
            # if there are nodes adjacent to the node we are looking at
            hotspot,adj_list = self.get_adjacent_nodes(node)
            self.adjacency_list.append([hotspot,adj_list])


    def get_adjacent_nodes(self, node, radius=350):
        inteference_list = []
        for pos in self.list_nodes:
            if node is not pos and pos is not None and node is not None:
                point1 = [node.x, node.y]
                point2 = [pos.x, pos.y]
                distance_to_neighbour = math.dist(point1, point2)
                if distance_to_neighbour < radius:
                    assert isinstance(pos, object)
                    inteference_list.append(pos)
                    self.update_interference(pos)
                    self.update_interference(node)

        return node,inteference_list

    def change_frequencies(self):
        frequencies = [1, 2, 3, 4, 5]
        freq_adjacent = []
        min_occuring_freq = 1
        for centroid in self.adjacency_list:
            #print(centroid[0], centroid[1])
            freq_adjacent=[node.frequency for node in centroid[1] if len(centroid[1])>0 ]


            for freq in frequencies:
                if (freq_adjacent.count(freq) < freq_adjacent.count(min_occuring_freq)):
                    min_occuring_freq = freq

        # check if all the frequencies are present in centroid and if not the assign centroid to external frequency
            a = set(frequencies).difference(set(freq_adjacent))
            if (len(a) > 0):
                centroid[0].frequency = a.pop()
                centroid[0].interference =0

                index_of_centroid = self.list_nodes.index(centroid[0])
                self.list_nodes[index_of_centroid] = centroid[0]

            else:
                if (centroid[0] is not None):
                    centroid[0].frequency = min_occuring_freq
                    index_of_centroid = self.list_nodes.index(centroid[0])
            if (centroid[0] is not None):
                self.list_nodes[index_of_centroid] = centroid[0]
        print('Done')
    def update_interference(self,node):
        if(node in self.list_nodes):
            index = self.list_nodes.index(node)
            node.update_interference(1)
            self.list_nodes[index] =node
```

```
def get_rand_frequency():
    # returns a frequency between one and five to assign
    return random.randint(1, 5)


def random_xy_coordinates():
    x = random.randint(0, 9999)
    y = random.randint(0, 9999)

    return x, y


def check_neighbours(current_x: int, current_y: int, positions_list, radius=10):
    """
        Get neighbours.

        Parameters
        ----------
        x : int
            The x coordinate.
        y : int
            The y coordinate.

        Returns
        -------
        bool
            whether to place hotspot or not.
        """

    place_hotspot = True
    positions_list = positions_list
    # if there is nothing in the area already we can just place anywhere
    if (len(positions_list) == 0):

        return place_hotspot

    else:
        for pos in positions_list:
            point1 = [current_x, current_y]
            point2 = [pos[0], pos[1]]
            distance_to_neighbour = math.dist(point1, point2)

            if (distance_to_neighbour < radius):
                place_hotspot = False
                break

    return place_hotspot


def initialise_area():
    N = 5000  # number of hotspots
    positions_list = []
    list_nodes = []
    while (N > 0):
        x, y = random_xy_coordinates()
        frequency = get_rand_frequency()

        # print(check_neighbours(x, y, positions_list))
        if (check_neighbours(x, y, positions_list)):
            # if there are other hotspots in the area already and if nothing is within 10m radius of new hotspot
            list_nodes.append(Node(x, y, frequency))

            positions_list.append([x, y])
            # print(x,y)
            N = N - 1

    print(f'Done: {len(list_nodes)} hotspots created')
    return list_nodes


obs = initialise_area()

graph = Graph(obs)
print('changing frequencies')
graph.change_frequencies()

num_interf = [node.interference for node in graph.list_nodes if node is not None and node.interference==1]
percent = sum(num_interf)/5000
print(f'Precentage of interferences: {percent}')
```

```
Done: 5000 hotspots created
changing frequencies
Done
Precentage of interferences: 0.9494
```

```
lst= [[node.x,node.y,node.interference] for node in graph.list_nodes]


# Create DataFrame
df = pd.DataFrame(lst, columns=['X', 'Y', 'interference'])
print(df)
```

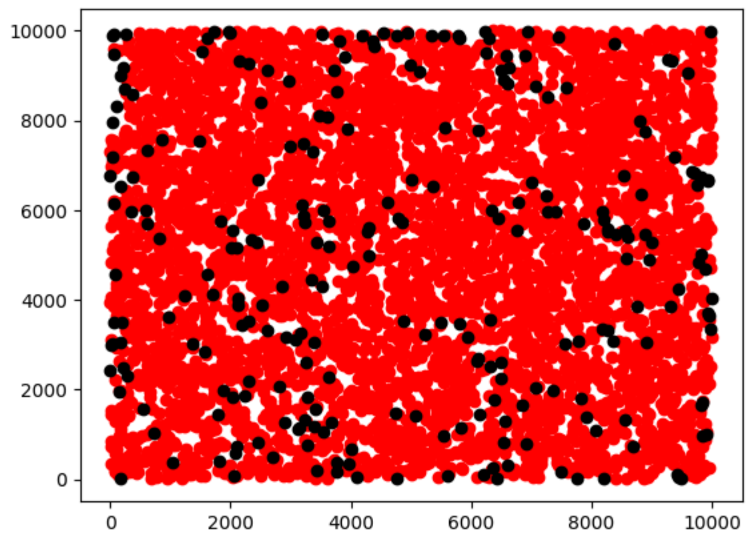
```
         X     Y  interference
0     9361   533             1
1     7906  9471             1
2     3376  7578             1
3     4546  2625             1
4     8530  8029             1
...    ...   ...           ...
4995  1750  6912             1
4996  2603  9710             1
4997  2292  2904             1
4998  5209  9509             1
4999  4241  8466             1

[5000 rows x 3 columns]
```

```
lst1= [[node.x,node.y,node.interference] for node in graph.list_nodes if node.interference ==1]


 # Create DataFrame
df = pd.DataFrame(lst1, columns=['X', 'Y', 'interference'])
plt.scatter(df['X'], df['Y'], c='red')
# second data point
lst2= [[node.x,node.y,node.interference] for node in graph.list_nodes if node.interference ==0]
 # Create DataFrame
df = pd.DataFrame(lst2, columns=['X', 'Y', 'interference'])
# depict second scatted plot
plt.scatter(df['X'], df['Y'], c='black')

# depict illustration
plt.show()
```


10000
8000
6000
4000
2000
0
0
2000
4000
6000
8000
10000