b10505005_hw4_readme.pdf

Description

Use the file to plot figures of describing package loss during different traffic loss

Considering SINR from different BS

- 50 random distributed mobile device in cell
- each mobile device has buffer to store data from BS (size = 6e6)
- other BS which cause interference on downlink communication
- three different mode of traffic load (low, medium, high)

Requirement

Python 3.8.10

Main Function

a. basic plot of the coordinates of base station 1-19

```
Xl = 10e3   # Low traffic load (bits/s)
Xm = 25e3  # Medium traffic load (bits/s)
Xh = 500e3  # High traffic load (bits/s)

# Simulation parameters
num_devices = 50
simulation_time = 1000  # Extended simulation time in secon
ds
traffic_buffer_size = 6e6  # 3M bits buffer size
```

```
def generate_points_in_hexagon(num_points, radius):
    x_points = []
    y_points = []

    while len(x_points) < num_points:
```

```
        x = np.random.uniform(-radius * np.sqrt(3) / 2, rad
ius * np.sqrt(3) / 2)
        y = np.random.uniform(-radius, radius)

        if in_hexagon(x, y, radius):
            x_points.append(x)
            y_points.append(y)

    return np.array(x_points), np.array(y_points)
```

```
def hexagon_vertices(side_length):
    return [
        (side_length * np.cos(np.pi / 3 * i), side_length *
np.sin(np.pi / 3 * i))
        for i in range(6)
    ]
```

```
def hexagonal_grid(ISD):
    coords = [(0, 0)]
    for i in range(6):
        angle = -np.pi / 6 + (np.pi / 3 * i)
        x = np.cos(angle) * ISD
        y = np.sin(angle) * ISD
        coords.append((x, y))

    for i in range(6):
        angle = (np.pi / 3 * i)
        x = np.cos(angle) * ISD * np.sqrt(3)
        y = np.sin(angle) * ISD * np.sqrt(3)
        coords.append((x, y))

    for i in range(6):
        angle = (-np.pi / 6 + np.pi / 3 * (i + 1))
        x = np.cos(angle) * 1000
        y = np.sin(angle) * 1000
        coords.append((x, y))
```

```
    return np.array(coords)
```

b. Calculate power from BSs

```
def calculate_interference(x_devices, y_devices, bs_coord
s):
    interference_power = np.zeros(len(x_devices))
    for x_bs, y_bs in bs_coords:
        distances_to_bs = np.sqrt((x_devices - x_bs) ** 2 +
(y_devices - y_bs) ** 2)
        Pr_interference = received_power_two_ray(distances_
to_bs)
        interference_power += Pr_interference
    return interference_power
```

```
def received_power_two_ray(d):
    return Pt * Gr * Gt * (h_device * h_base) ** 2 / (d **
4)
```

c. Generate traffic and Calculate packet loss

In this program, we set up two situation for packet generation: CBR and Poisson, and calculate packet loss probability

```
def received_power_two_ray(d):
    return Pt * Gr * Gt * (h_device * h_base) ** 2 / (d **
4)
```

```
def generate_traffic(CBR, time):
    return CBR * time

def generate_poisson_traffic(lam, simulation_time):
    num_packets = np.random.poisson(lam * simulation_time)
    total_traffic = num_packets
    return total_traffic
```

```
def calculate_packet_loss(buffer_size, generated_traffic):
    if generated_traffic > buffer_size:
        return (generated_traffic - buffer_size) / generated_
    return 0
```

Compile

```
  python 4.1.py
  python 4.2.py
```

Contact：ESOE b10505005 蔣依倢