Channel Quality-Based Greedy User Selection

This code implements a simulation of the Channel Quality-Based Greedy User Selection method. The method involves selecting users based on their channel quality and allocating transmission power among them. The total transmission rate for each simulation is calculated using a specific formula. The code generates histograms to visualize the distribution of total rates per transmission for different channels.

In this method, the term "greedy manner" refers to the iterative selection process used to choose users based on their channel quality. Here's a breakdown of the process:

1. Start: Begin with an empty set of selected users.

2. Initialization: Calculate the channel quality metric (e.g., norm of the channel vector) for all users.

3. Selection: Select the user with the highest channel quality (e.g., the highest norm) and add them to the set of selected users.

4. Iterative Selection: From the remaining users, iteratively choose the user with the next highest channel quality that maximizes the overall system capacity. Add each selected user to the set of selected users.

5. Stopping Condition: The selection process continues until a specific condition is met, such as allocating all available resources or reaching a predefined stopping criterion.

6. End: Once the stopping condition is met, the final set of selected users represents the chosen users based on their channel quality in a greedy manner.

The "greedy manner" refers to the step-by-step process of selecting users based on their channel quality, prioritizing the user with the highest quality and continuing the selection process by adding users with progressively lower quality until the stopping condition is satisfied.

Explanation about Code:

This code simulates the Channel Quality-Based Greedy User Selection method using different channel models. It consists of several functions and a main script.

1. Channel Generation Functions:

    a. **generate_complex_Gaussian_channel(K, t)**:
    This function generates an complex Gaussian channel. It creates a 2D numpy array of shape **(K, t)** where each entry follows a Gaussian distribution with a mean of 0 and a variance of 0.5. The channel represents the connection between each receiving antenna and each of the transmitting antennas.

    b. **generate_chaotic_channel(K, t)**:
    This function generates a chaotic channel model. Each channel (row) is characterized by a complex Gaussian distribution with a mean randomly drawn from a uniform distribution in the range [0,1] and a variance randomly drawn from a uniform distribution in the range [0,3].

    c. **generate_correlative_channel(K, t)**:
    This function generates a complete correlative channel. All antennas have the same randomly drawn complex Gaussian distribution with a mean of 0 and a variance of 1.

2. **Channel Quality-Based Greedy User Selection Function:**

    The core functionality is implemented in the *channel_quality_based_greedy_user_selection(channel_quality, t, P)* function. It takes the channel quality matrix, the number of users to select **t**, and the transmission power **P** as inputs.

    - It initializes an array of users and an empty list to store the selected users.
    - Within a while loop, it selects **t** users with the highest channel quality based on the norm of the channel quality vector.
    - The transmission power is equally allocated among the selected users.
    - The transmission rate for each selected user is calculated using the formula: *rates = np.log2(1 + power_allocation * np.abs(channel_quality[selected_users]) ** 2).*
    - The total transmission rate is obtained by summing up the rates of all selected users.
    - The total rate is returned as the output.

Simulation Function:

The *simulate_channel_quality_based_greedy_user_selection(channel_generator, K, t, P, num_simulations)* function performs simulations for a given channel model.

- It takes a *channel_generator* function, the number of users **K**, the number of users to select **t**, the transmission power **P**, and the number of simulations *num_simulations* as inputs.
- It initializes an empty list to store the total rates obtained in each simulation.
- For each simulation, it generates the channel quality matrix by calling the *channel_generator* function.
- It then calls the *channel_quality_based_greedy_user_selection* function to obtain the total transmission rate for that simulation.
- The total rate is appended to the list of total rates.
- Finally, the list of total rates is returned as the output.

Main Script:

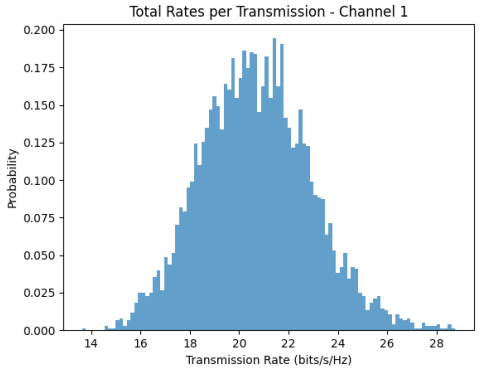
The main script sets the simulation parameters: **K** (number of users), **P** (transmission power in Watts), **t** (number of users to select), and *num_simulations* (number of simulations to run).

- It simulates the channel quality-based greedy user selection for three different channels:
    - Simulate channel 1: *generate_complex_Gaussian_channel* is used to generate the channel quality, and the simulation results are stored in *total_rates_channel1.*
    - Simulate channel 2: *generate_chaotic_channel* is used to generate the channel quality, and the simulation results are stored in *total_rates_channel2.*
    - Simulate channel 3: *generate_correlative_channel* is used to generate the channel quality, and the simulation results are stored in *total_rates_channel3*.
- For each channel simulation, the *plot_total_rates* function is called to generate a histogram of the total transmission rates and save it as a PNG image with the respective channel name.

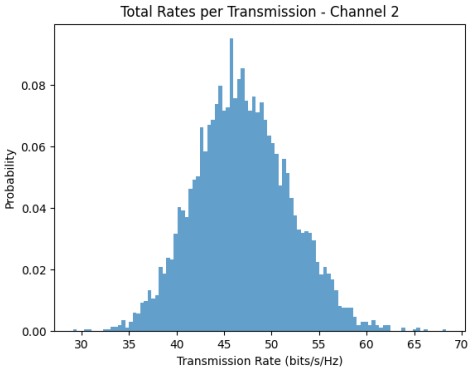
The output of the code is three histograms, each representing the distribution of total transmission rates per simulation for a specific channel. The histograms are saved as PNG images with filenames "Channel 1.png", "Channel 2.png", and "Channel 3.png".

Simluation Graphs:

Gaussian



chaotic



correlative