File - C:\Users\berkenutku\Desktop\496_hw3\plague2.py

```python
from plague import Plague
import numpy as np
import random
import matplotlib.pyplot as plt


def fuzzy_controller(infected_percentage, effective_infection):
    # Calculating the membership for the current effective infection rate
    effective_memberships = [effective_infection_low(effective_infection),
                             effective_infection_good(effective_infection),
                             effective_infection_high(effective_infection)]

    # Calculating the membership for the infected bots rate
    memberships = [calculate_membership_low_infected(infected_percentage),
                   calculate_membership_good_infected(infected_percentage),
                   calculate_membership_high_infected(infected_percentage)]

    best_member = memberships.index(max(memberships))
    best_effective = effective_memberships.index(max(effective_memberships))

    # According to the highest memberships, evaluate the output control variable by using The Max Criterion method
    # The following conditional statements are the implementation of the Table 1 in the report.
    if best_member == 0:
        if best_effective == 0 or best_effective == 1:
            return output_controller_high_positive(memberships[0])
        else:
            return output_controller_low_positive(memberships[0])
    elif best_member == 1:
        if best_effective == 2:
            return output_controller_high_negative(memberships[1])
        else:
            return output_controller_good(memberships[1])
    else:
        if best_effective == 0 or best_effective == 1:
            return output_controller_high_negative(memberships[2])
        else:
            return output_controller_low_negative(memberships[2])


# Calculate the membership for the current effective infection rate (Low)
def effective_infection_low(x):
    if -0.2 >= x >= -1:
        return 1
    elif 0 >= x >= -0.2:
        value = -5 * x
        return value
    else:
        return 0


# Calculate the membership for the current effective infection rate (High)
def effective_infection_high(x):
    if 1 >= x >= 0.2:
        return 1
    elif 0 <= x <= 0.2:
        value = 5 * x
        return value
    else:
        return 0


# Calculate the membership for the current effective infection rate (Good)
def effective_infection_good(x):
    if 0 >= x >= -0.2:
        return_value = 5 * x + 1
    elif 0.2 >= x >= 0:
        return_value = -5 * x + 1
    else:
        return_value = 0
    return return_value


# Calculate the membership for the infected bots (Low)
def calculate_membership_low_infected(x):
    if x <= 0.4:
        return 1
```

File - C:\Users\berkenutku\Desktop\496_hw3\plague2.py

```
    elif x >= 0.6:
        return 0
    else:
        slope = -(1 / 0.2)
        value = 1 - min(1, slope * (0.4 - x))
        return value


# Calculate the membership for the infected bots (Good)
def calculate_membership_good_infected(x):
    if 0.65 >= x >= 0.6:
        return_value = -10 * x + 7
    elif 0.6 >= x >= 0.55:
        return_value = (10 * x) - 5
    else:
        return_value = 0
    return return_value


# Calculate the membership for the infected bots (High)
def calculate_membership_high_infected(x):
    if x >= 0.8:
        return 1
    elif x <= 0.6:
        return 0
    else:
        slope = (1 / 0.2)
        value = 1 - min(1, slope * (0.8 - x))
        return value


# Calculating the output control variable from membership (Good)
def output_controller_good(x):
    first_point = (x - 1) * 3 / 100
    second_point = -((x - 1) * 3 / 100)
    return_value = random.uniform(first_point, second_point)
    return return_value


# Calculating the output control variable from membership (Low (+))
def output_controller_low_positive(x):
    first_point = 0.03 * x
    second_point = -(x-2) * 0.03
    return_value = random.uniform(first_point, second_point)
    return return_value


# Calculating the output control variable from membership (Low (-))
def output_controller_low_negative(x):
    first_point = ((x - 2) * 3 / 100)
    second_point = first_point + 2 * (-0.03 - first_point)
    return_value = random.uniform(first_point, second_point)
    return return_value


# Calculating the output control variable from membership (High (+))
def output_controller_high_positive(x):
    if x == 1:
        return_value = random.uniform(0.09, 0.15)
    else:
        second_point = (x + 0.5) * 0.06
        return_value = random.uniform(second_point, 0.09)
    return return_value


# Calculating the output control variable from membership (High (-))
def output_controller_high_negative(x):
    if x == 1:
        return_value = random.uniform(-0.15, -0.09)
    else:
        second_point = (x + 0.5) * -0.06
        return_value = random.uniform(-0.09, second_point)
    return return_value


plague = Plague()

```

File - C:\Users\berkenutku\Desktop\496_hw3\plague2.py

```
number_iterations = 200
counter = 0
infected_bots_plot = np.empty(shape=(200,))
effective_infection_rates = np.empty(shape=(200,))
# Loop for spreading virus for 20 days
while counter != number_iterations:
    # Get the rates for the infected bot and effective infection
    infected_bots, effective_infection_rate = plague.checkInfectionStatus()
    # Add them to array for debugging and plotting
    infected_bots_plot[counter] = infected_bots
    effective_infection_rates[counter] = effective_infection_rate
    # Use infected bots rate in order to generate a control variable
    control_variable = fuzzy_controller(infected_bots, effective_infection_rate)
    # Spread virus
    plague.spreadPlague(control_variable)
    # Increase the loop iteration
    counter += 1


# Plotting
plt.ylabel("Infection rate")
plt.xlabel("Steps")
plt.plot(plague.infected_percentage_curve_)
plt.show()

plt.ylabel("Infection cost")
plt.xlabel("Steps")
plt.plot(plague.infection_rate_curve_)
plt.show()

plt.ylabel("effective infection rate")
plt.xlabel("Steps")
plt.plot(effective_infection_rates)
plt.show()

# Computation of the total cost until equilibrium
cost_sum = sum(plague.infected_percentage_curve_[1:77])

plague.viewPlague(77, cost_sum)
```