

```

1 from plague import Plague
2 import numpy as np
3 import random
4 import matplotlib.pyplot as plt
5
6
7 def fuzzy_controller(percentage):
8     # Calculate the memberships of the infected bots rate for 3 different regions
9     memberships = [calculate_membership_low_infected(percentage),
10                    calculate_membership_good_infected(percentage),
11                    calculate_membership_high_infected(percentage)]
12
13     best_member = memberships.index(max(memberships))
14     # According to the highest memberships, evaluate the output control variable by using The Max
Criterion method
15     if best_member == 0:
16         return output_controller_low(memberships[0])
17     elif best_member == 1:
18         return output_controller_good(memberships[1])
19     else:
20         return output_controller_high(memberships[2])
21
22
23 # Calculating the membership for low infected bots
24 def calculate_membership_low_infected(x):
25     if 0.4 >= x > 0:
26         return_value = 1
27     elif 0.55 >= x >= 0.4:
28         return_value = -(x / 0.15) + 3.6666
29     else:
30         return_value = 0
31     return return_value
32
33
34 # Calculating the output control variable from membership (Low)
35 def output_controller_low(x):
36     first_point = (0.15 * x / 2)
37     second_point = -((x - 2) * 0.15 / 2)
38     return_value = random.uniform(first_point, second_point)
39     return return_value
40
41
42 # Calculating the membership for good rate infected bots
43 def calculate_membership_good_infected(x):
44     if 0.7 >= x >= 0.6:
45         return_value = -10 * x + 7
46     elif 0.6 >= x >= 0.5:
47         return_value = (10 * x) - 5
48     else:
49         return_value = 0
50     return return_value
51
52
53 # Calculating the output control variable from membership (Good)
54 def output_controller_good(x):
55     first_point = (x - 1) * 0.15 / 2
56     second_point = -((x - 1) * 0.15 / 2)
57     return_value = random.uniform(first_point, second_point)
58     return return_value
59
60
61 # Calculating the membership for High rate infected bots
62 def calculate_membership_high_infected(x):
63     if 1 >= x >= 0.8:
64         return_value = 1
65     elif 0.8 >= x >= 0.65:
66         return_value = (x / 0.15) - 4.33
67     else:
68         return_value = 0
69     return return_value
70
71
72 # Calculating the output control variable from membership (High)
73 def output_controller_high(x):
74     first_point = ((x - 2) * 0.15 / 2)
75     second_point = -(0.15 * x / 2)
76     return_value = random.uniform(first_point, second_point)

```

```
77     return return_value
78
79
80 plague = Plague()
81
82 number_iterations = 200
83 counter = 0
84 infected_bots_plot = np.empty(shape=(200,))
85 effective_infection_rates = np.empty(shape=(200,))
86 # Loop for spreading virus for 20 days
87 while counter != number_iterations:
88     # Get the rates for the infected bot and effective infection
89     infected_bots, effective_infection_rate = plague.checkInfectionStatus()
90     # Add them to array for debugging and plotting
91     infected_bots_plot[counter] = infected_bots
92     effective_infection_rates[counter] = effective_infection_rate
93     # Use infected bots rate in order to generate a control variable
94     control_variable = fuzzy_controller(infected_bots)
95     # Spread virus
96     plague.spreadPlague(control_variable)
97     # Increase the loop iteration
98     counter += 1
99
100 # Plotting
101 plt.ylabel("Infection rate")
102 plt.xlabel("Steps")
103 plt.plot(plague.infected_percentage_curve_)
104 plt.show()
105
106 plt.ylabel("Infection cost")
107 plt.xlabel("Steps")
108 plt.plot(plague.infection_rate_curve_)
109 plt.show()
110
111 plt.ylabel("effective infection rate")
112 plt.xlabel("Steps")
113 plt.plot(effective_infection_rates)
114 plt.show()
115
116 # Computation of the total cost until equilibrium
117 cost_sum = sum(plague.infected_percentage_curve_[1:100])
118
119 plague.viewPlague(100, cost_sum)
120
```