ARTIFICIAL INTELLIGENCE

PROGRAMMING ASSIGNMENT - 2

BY:
ASHIMA GARG
PhD19003

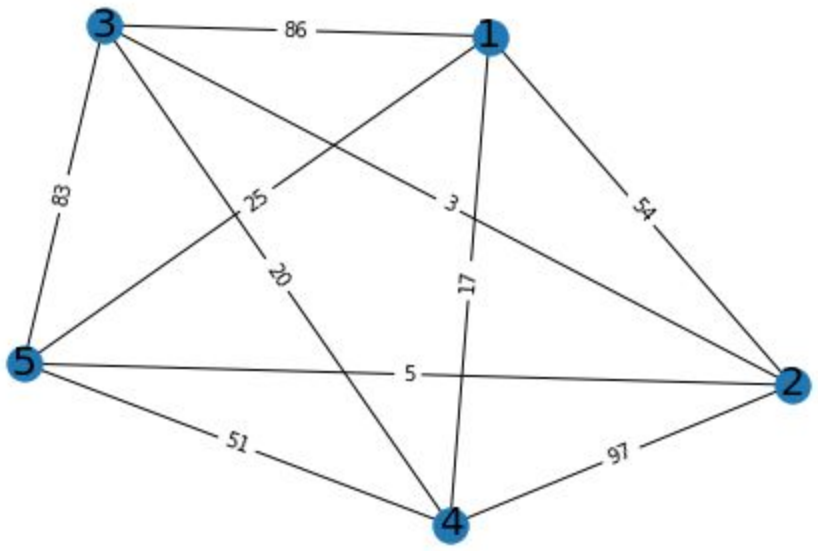
SOLUTION 1: ANT COLONY OPTIMIZATION

Created N*N matrix

Following is the distance matrix plotted using NetworkX library generated randomly for with N=5.

Distance matrix:

999	13	36	66	88
54	999	31	83	81
86	3	999	77	59
17	97	20	999	87
25	5	83	51	999



Assumptions

All ants started from the same initial position in different directions.
Convergence takes place when all the ants started moving in the same path.
Initialized pheromone level to 0.1
Evaporation factor = 0.8

Alpha = 1
Beta = 1

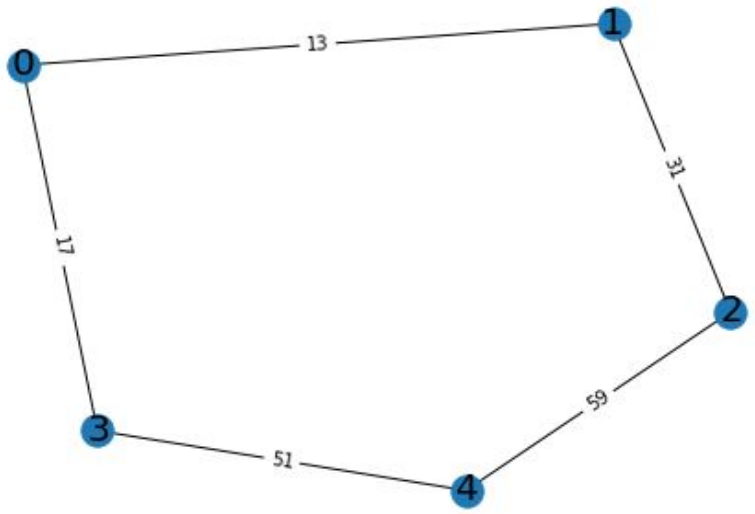
Observations and Inferences

Following observations were made:

- With the increase in value of N, the number of iterations required for all the ants moving the same path increased.
- With the increase in the alpha value i.e. weight given to the pheromone level for each ant, the number of iterations required for convergence decreased.
- With the increase in the beta value i.e. weight given to the pheromone level for each ant, the number of iterations required for convergence decreased, but the convergence is observed more in the case of alpha.
- Within 10 iterations the algorithm converges for N <= 10

Results

Following is the final graph for the above randomly created matrix.



Shortest Distance found for the above matrix is **171**

SOLUTION 2: Q-Learning in FrozenLake Environment

Assumptions in Settings 1: Following parameters are set:
episodes = 20000
exploration_rate = 1.0
max_epsilon = 1.0
min_epsilon = 0.01
decay_rate = 0.005
learning_rate = 0.5
max_steps = 99

Gamma is varied in the range (0.7, 0.99) with 15 different values and Learning rate is varied in the range (0.2, 0.6) with 10 different values:

Observations

Some observations for different values of gamma and learning rate are:
NOTE: *All the observations are not noted here. These are few in number just to illustrate the changes:*

Gamma	Learning Rate	Score
0.7	0.2	0.212
0.7	0.244	0.19775
0.72	0.2	0.22485
0.72	0.244	0.22515
0.74	0.466	0.2468
0.74	0.511	0.24155
0.76	0.422	0.249
0.76	0.511	0.267
0.845	0.6	0.34865
0.865	0.2	0.3846
0.948	0.2	0.56
0.948	0.24	0.55

0.969	0.2	0.626
0.969	0.244	0.611
0.99	**0.2**	**0.647**
0.99	0.244	0.6433

*Optimal value of gamma and learning rate are found to be **0.99** and **0.2** respectively.*

Assumptions in Settings 2: Following parameters are set:

Here optimum value of gamma and learning rate obtained in Settings 1 are used to find optimal value of decay rate in range (0.002, 0.006) with 10 different values:

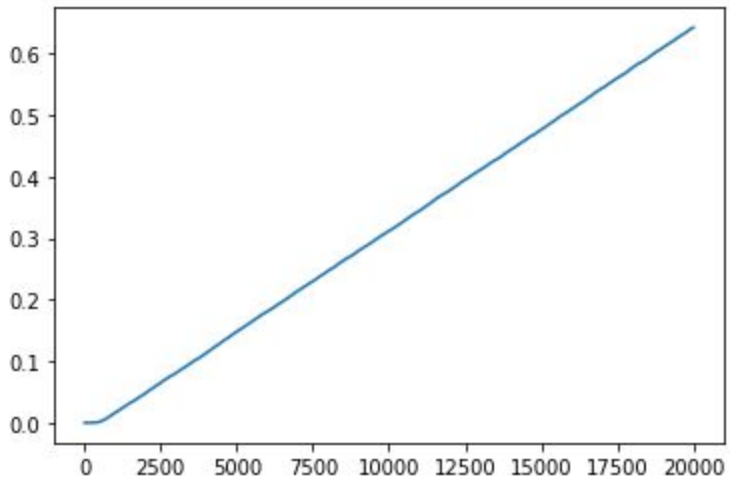
episodes = 20000
gamma = 0.99
learning rate = 0.2
max_steps = 99
exploration_rate = 1.0
max_epsilon = 1.0
min_epsilon = 0.01

Following are the few observations for different values of decay rate and score:

Decay Rate	Score
0.002	0.63135
0.0024	0.6347
0.0028	0.64035
0.0033	0.64105
0.0037	0.64
0.0042	0.6516
0.0051	0.655
0.0055	0.6531
0.006	**0.6536**

Optimal value of decay rate is found to be: ***0.006***

Plot Score vs Episodes



Inferences

- The learning rate is tuned to so as to control the effect of old information in the value of current Q(state, action). Large values of learning rates imply the most recent information will be considered. Small values of learning rates imply the agent will store the previous values only, and it is not learning. Therefore, learning rate of 0.2 achieves a good score as compared to others as the agent learns slowly and do not forget the old information very fast.
- Discount factor is tuned to determine the importance of future rewards. The lower the value of discount factor, the less important future rewards are, and the agent will focus on current reward values which will have high values.
- Decay rate controls the amount of exploration or exploitation need to be done. Initially exploration rate should be high as the agent needs to learn. With the increase in the number of episodes, the exploration rate decreases. Decay rate controls the amount of decay in the exploration rate with the number of episodes and it should be tweaked carefully, as too large or too low values causes abrupt downfall in the average scores received.

SOLUTION 3: TicTacToe Game using Reinforcement Learning

Assumptions

- Initially, Q-Learning algorithm of Reinforcement learning is used to train the two AI Bots against each other by maintaining their respective Q tables. The Q tables learned from the training is used to test against the Human Player.
- Individual Q tables are maintained for the two Bots and are initialized to small positive values i.e. 0.6 and not to 0.
- Reward Values assigned to the two bots is +100 whenever it wins the game and 0 when the other bot wins the game i.e. If Bot1 wins the game, it is assigned +100 reward whereas Bot2 is given 0 reward in this case and vice versa i.e. if Bot2 wins the game, it is assigned +100 reward and Bot1 is given 0 reward.
- In the remaining cases, i.e. when match draws or episode doesn't end, both the bots are assigned 0 rewards.
- Both the Q tables are updated for each action of the two bots with their respective Q values.

Observations and Inferences

Hyperparameter tuning of the parameters: gamma, learning rate, number of episodes to train the algorithm, exploration rate, are used to train the bots.

- Learning rate is varied from 0.1 to 0.6
  Too high or too low values lowers down the performance of the bots.
  Optimal learning rate found is: 0.2

- Gamma Parameter is varied from 0.7 to 0.99
  Optimal gamma found is 0.7

- Episodes: 5000- 50000
  Initially, the total score achieved by the bot increases with the increase in the number of episodes but the performance decays after the number of episodes reaches 20000.
  Optimal value of episodes is found to be 20000

- Exploration Rate: 0.2 to 0.7
  Various settings of exploration rate are observed:
  - Settings 1: When exploration rate is kept fixed.

- Settings 2: When decay rate is used to lower down the value of exploration rate with the increase in the number of episodes by decay rate of 0.05.

Optimal exploration rate of 0.3 is observed in Settings 1.

Game Played against trained BOT and Human Player with optimal parameters:

Move 1:

Available Positions: 9
AI Plays and Output is

```
-------------
|   |   |   |
-------------
|   |   |   |
-------------
|   |   | x |
-------------
```

Move 2:

Available Positions: 8
Human Plays:
Input Row: 0
Input Col: 0
Output is

```
-------------
| o |   |   |
-------------
|   |   |   |
-------------
|   |   | x |
-------------
```

Move 3:

Available Positions: 7
AI Plays and Output is:

```
-------------
| o |   |   |
-------------
|   |   |   |
-------------
| x |   | x |
-------------
```

Move 4:
Available Positions: 6
Human Plays:
Input Row: 2
Input Col: 1
```
-------------
| o |   |   |
-------------
|   |   |   |
-------------
| x | o | x |
-------------
```

Move 5:
Available Positions: 5
AI Plays and Output is:
```
-------------
| o |   | x |
-------------
|   |   |   |
-------------
| x | o | x |
-------------
```

In the above figure it can be noted, trained BOT played intelligently to baffle Human player and now whatever move Human will choose, BOT will win the game.

Move 6:
Available Positions: 4
Input Row: 1
Input Col: 1
```
-------------
| o |   | x |
-------------
|   | o |   |
-------------
| x | o | x |
-------------
```

Move 7:
Available Positions: 3
AI Plays and output is:

| o | | x |

| | o | x |

| x | o | x |

BOT wins the game!

Even the Human player chooses the following move, it loses against the trained AI.
Available Positions: 4
Input Row: 1
Input Col: 2

| o | | x |

| | | o |

| x | o | x |

Available Positions: 3

| o | | x |

| | x | o |

| x | o | x |

BOT wins the game!