8_avg_rl_true.py

1. <u>For all bots which are currently at a node & idle</u>

Update:

- true idleness of all nodes in graph (true) = +Δt
- Store all the true idleness values at this time stamp
- Expected idleness of nodes at which bots are currently present is calculated as the avg of true idleness while going through a particular edge (now, expected idleness is function of edge not node)

<u>Calculate</u>: here, learning rate (α)= 0.1, discount factor (γ)= 0.95

- Value function all edges where bots are present (Q) =

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t,a_t)}_{\text{old value}}\Bigg)}^{\text{temporal difference}}$$

- Reward (r_t) = true idleness
- Softmax of Value function = value_exp = $\frac{e^{Q_i}}{\sum_{j=i}^{k} e^{Q_j}}$ (summation over all edges)

Set:

- True idleness of nodes where bots are present = 0
- Expected idleness of nodes wrt the corresponding bot = 0

OBSERVATION model: bot will calculate the expected idleness as an average of all the past true idleness it has seen when it last visited the node while travelling **along that particular edge**.

The name 8_avg_rl_true indicates => avg = averaging true idleness to get expected idleness
(OBSERVATION model)
rl = using reinforcement learning (Q-learning algorithm to calculate the value function)

true = since reward is changed to true idleness

2. For a bot deciding the next node to visit

Set:

- True idleness of the node where the bot is present = 0

Decision Making: here, we chose ε=0.1

- With $(1 - \varepsilon)$probability, check all neighbours and visit the one with highest value of = $[expected\ idleness]$x **max** $([value\_exp])$

Here, for each neighbour, we first calculate the maximum value_exp value we can get going to that node. Then we choose the highest value of { expect*max(value_exp) } over all neighbours.

- With ε probability, go to a random node

---END--