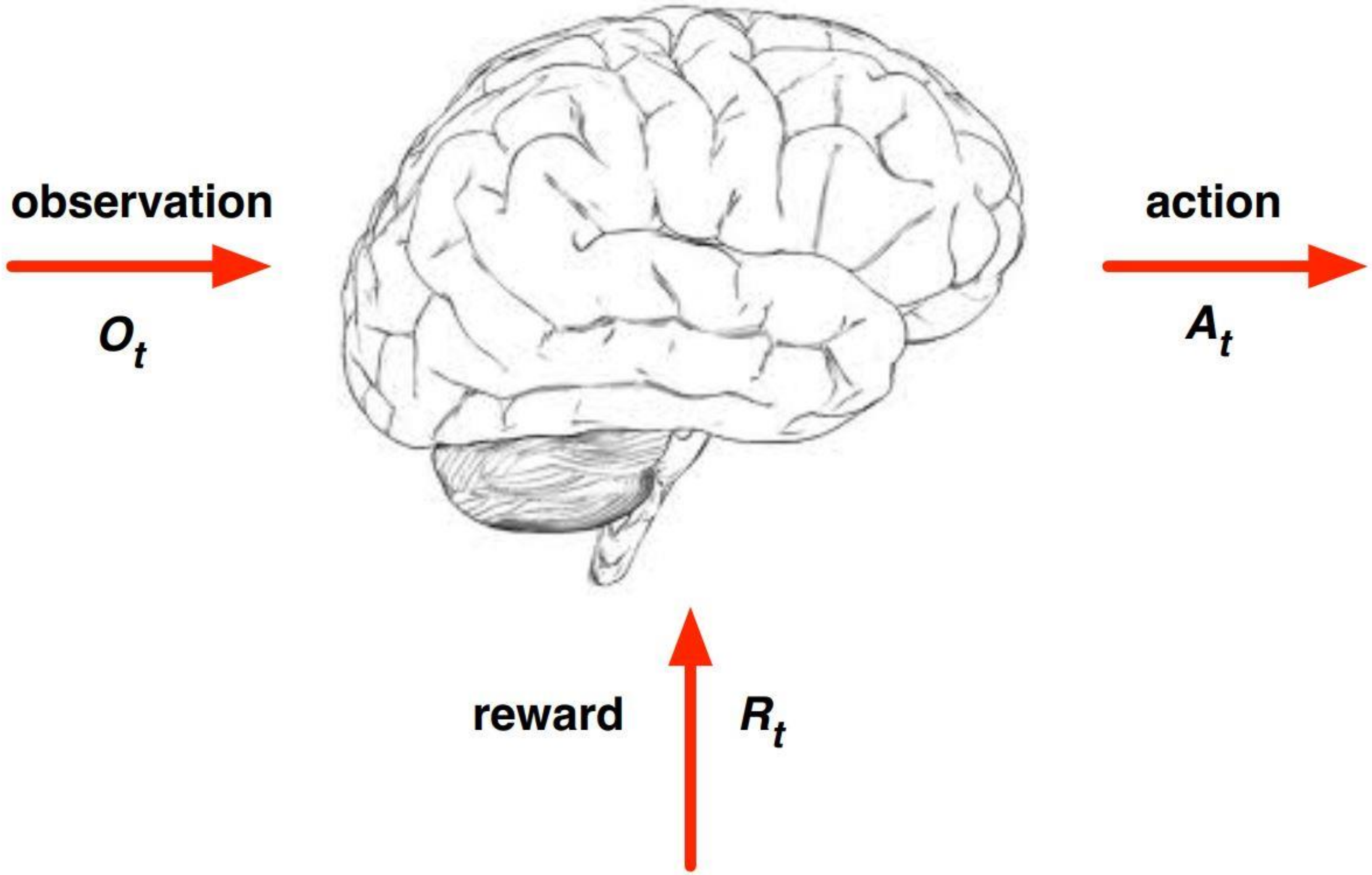
A STUDY OF SOME PROPERTIES OF ANT-Q

Arman Akbari, Arash Akbari

Bio-inspired computation - Dr. BabaAli

Agent and Environment



Definition

The *return* G_t is the total discounted reward from time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k + 1$ time-steps is $\gamma^k R$.
- This values immediate reward above delayed reward.
  - γ close to 0 leads to "myopic" evaluation
  - γ close to 1 leads to "far-sighted" evaluation

Value function

The value function $v(s)$ gives the long-term value of state s

Definition

The *state value function* $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$

Policies

Definition

A *policy* π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$$

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (not the history)
- i.e. Policies are *stationary* (time-independent),
  $A_t \sim \pi(\cdot|S_t), \forall t > 0$

Definition

The *state-value function* $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s\right]$$

Definition

The *action-value function* $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a\right]$$

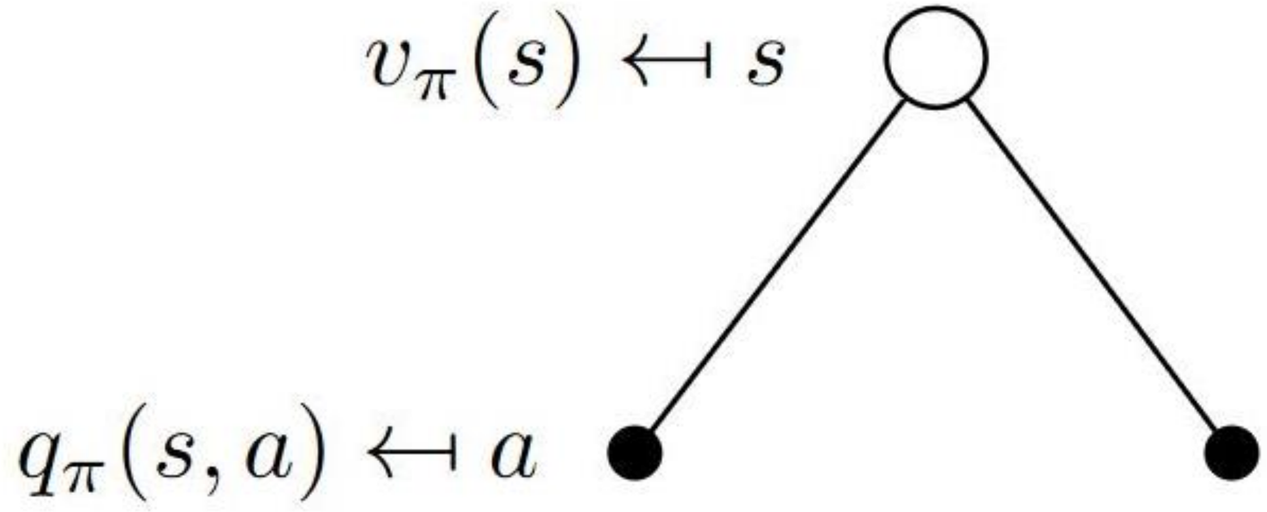
Bellman Expectation Equation

The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right]$$

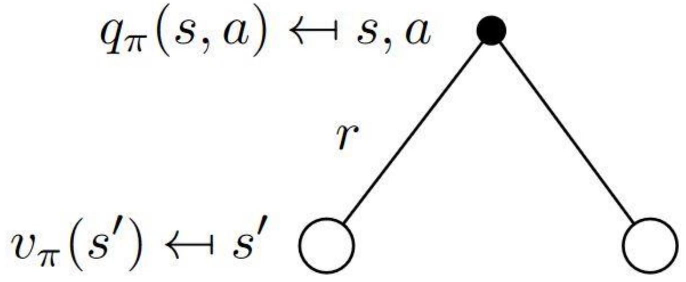
The action-value function can similarly be decomposed,

$$q_\pi(s, a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right]$$



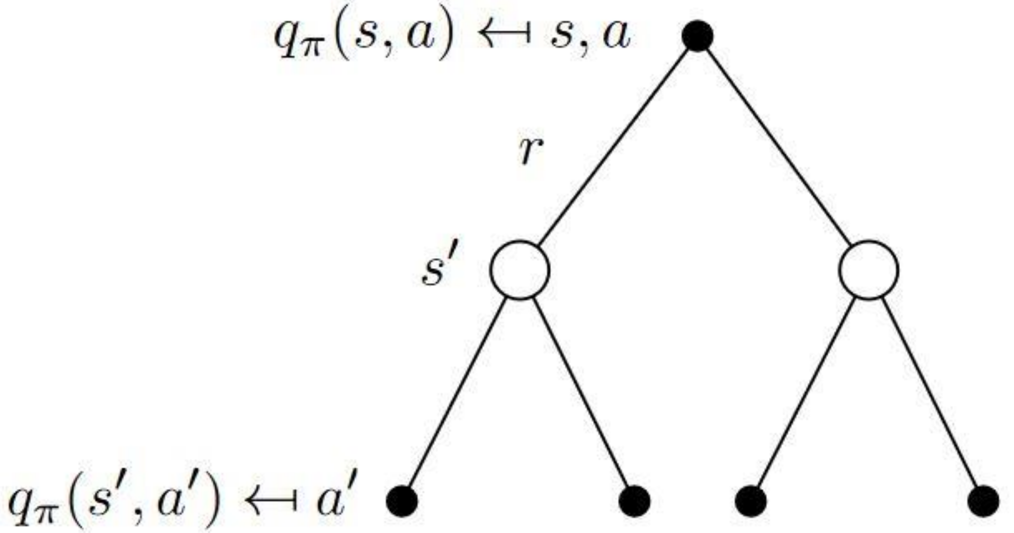
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

Bellman Expectation Equation for q



$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Bellman Expectation Equation for Q



$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

Optimal values

Definition

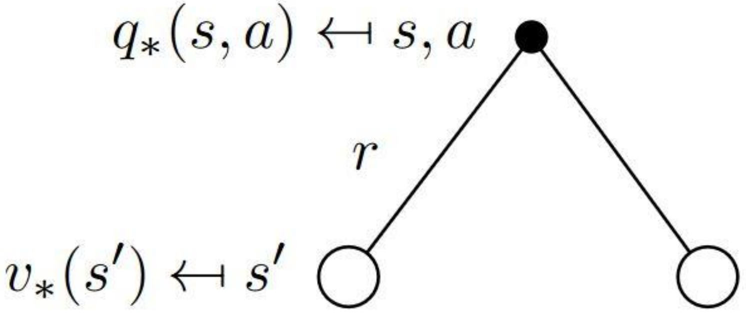
The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

$$v_*(s) = \max_{\pi} v_{\pi}(s)$$

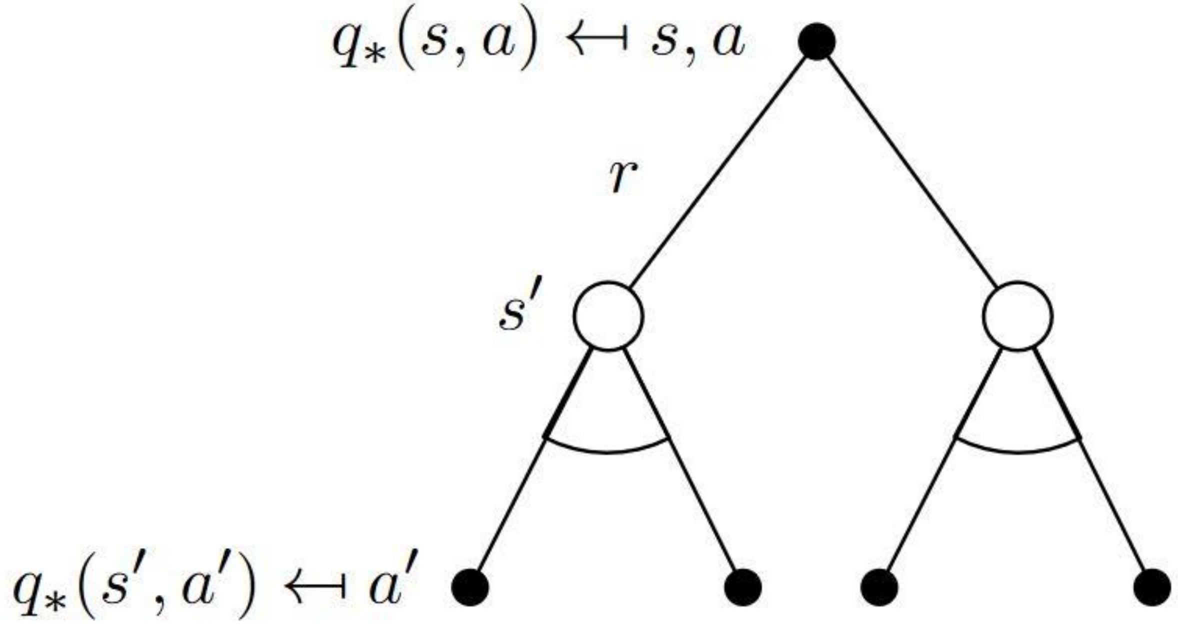
The *optimal action-value function* $q_*(s, a)$ is the maximum action-value function over all policies

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

Bellman Optimality equation for Q



$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$



$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

Off Policy Learning

- Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$
- While following behaviour policy $\mu(a|s)$

$$\{S_1, A_1, R_2, ..., S_T\} \sim \mu$$

- Why is this important?
- Learn from observing humans or other agents
- Re-use experience generated from old policies $\pi_1, \pi_2, ..., \pi_{t-1}$
- Learn about *optimal* policy while following *exploratory* policy
- Learn about *multiple* policies while following *one* policy

Q Learning

- We now consider off-policy learning of action-values $Q(s, a)$
- No importance sampling is required
- Next action is chosen using behaviour policy $A_{t+1} \sim \mu(\cdot|S_t)$
- But we consider alternative successor action $A' \sim \pi(\cdot|S_t)$
- And update $Q(S_t, A_t)$ towards value of alternative action

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t) \right)$$

Off-policy Control with Q learning

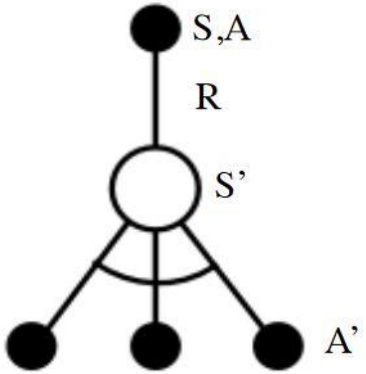
- We now allow both behaviour and target policies to improve
- The target policy π is greedy w.r.t. $Q(s, a)$

$$\pi(S_{t+1}) = \underset{a'}{\operatorname{argmax}}\ Q(S_{t+1}, a')$$

- The behaviour policy μ is e.g. ϵ-greedy w.r.t. $Q(s, a)$
- The Q-learning target then simplifies:

$$\begin{aligned} & R_{t+1} + \gamma Q(S_{t+1}, A') \\ = & R_{t+1} + \gamma Q(S_{t+1}, \underset{a'}{\operatorname{argmax}}\ Q(S_{t+1}, a')) \\ = & R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a') \end{aligned}$$

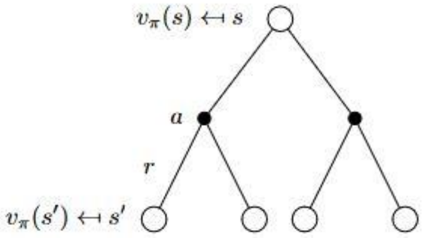
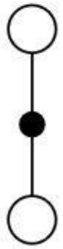
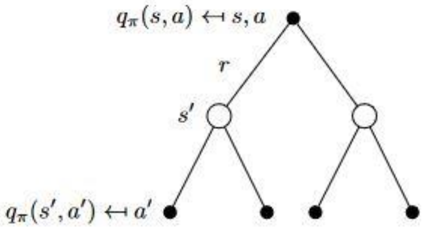
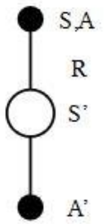
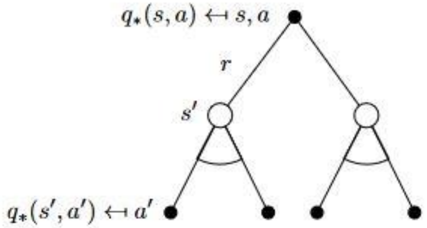
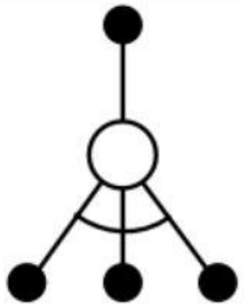
Q Learning Control Algorithm



$$Q(S, A) \leftarrow Q(S, A) + \alpha \left(R + \gamma \max_{a'} Q(S', a') - Q(S, A) \right)$$

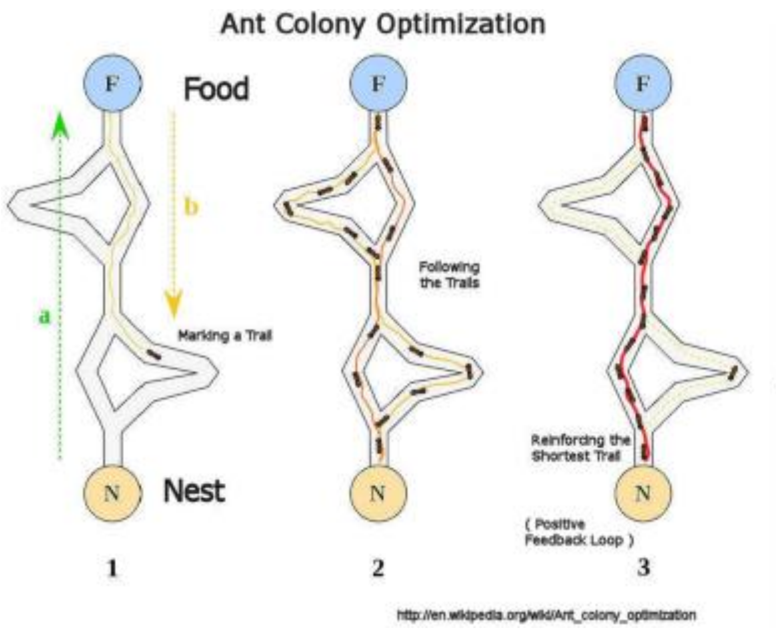
Theorem

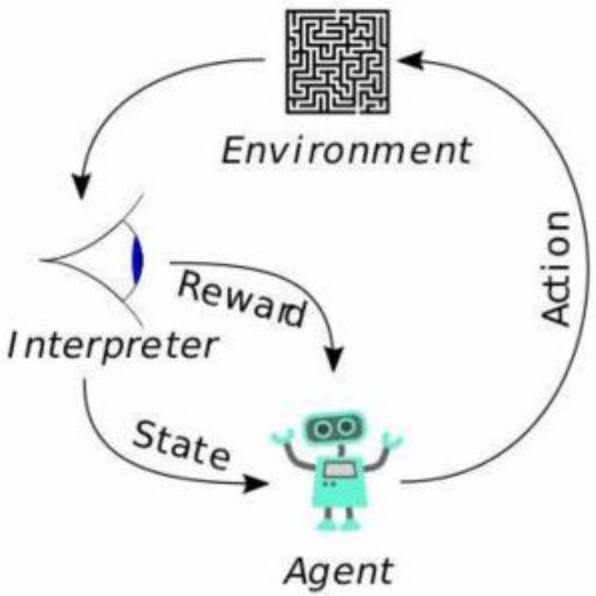
Q-learning control converges to the optimal action-value function, $Q(s, a) \rightarrow q_*(s, a)$

	Full Backup (DP)	*Sample Backup (TD)*
Bellman Expectation Equation for $v_\pi(s)$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $q_\pi(s,a)$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $q_*(s,a)$	 Q-Value Iteration	 Q-Learning

Introduction

- what is Ant-Q?
  - A family of algorithms which strengthen the connection between RL, in particular Q-learning, and AS
  - AS (Ant System)
  - Q-learning algorithm





The Ant-Q Approach to Combinatorial Optimization

- Let *k* be an ant whose task is to make a tour
- Associated to *k* there is the list *J k(r)* of cities still to be visited, where *r* is the current
  city

$$s = \begin{cases} \arg\max\limits_{u \in J_k(r)} \left\{ [AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta} \right\} & \text{if } q \le q_0 \\ S & \text{otherwise} \end{cases}$$

Ant-Q-value

Heuristic function

$$s = \begin{cases} \arg\max\limits_{u \in J_k(r)} \left\{ [AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta} \right\} & \text{if } q \le q_0 \\ S & \text{otherwise} \end{cases}$$

A value chosen randomly with uniform probability in [0,1]

A random variable selected according to the distribution given by this formula

$$p_k(r,s) = \begin{cases} \dfrac{[AQ(r,s)]^{\delta} \cdot [HE(r,s)]^{\beta}}{\sum\limits_{z \in J_k(r)} [AQ(r,z)]^{\delta} \cdot [HE(r,z)]^{\beta}} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases}$$

The action choice rule

- The Pseudo-random rule
- The pseudo-random-proportional rule

$$p_k(r,s) = \begin{cases} \dfrac{[AQ(r,s)]^{\delta} \cdot [HE(r,s)]^{\beta}}{\sum\limits_{z \in J_k(r)} [AQ(r,z)]^{\delta} \cdot [HE(r,z)]^{\beta}} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases}$$

- The random-proportional rule (the same as AS) → q_0 = 0

$$AQ(r,s) \leftarrow (1-\alpha)\cdot AQ(r,s) + \alpha \cdot \left(\Delta AQ(r,s) + \gamma \cdot \underset{z \in J(s)}{Max}\ AQ(s,z) \right)$$

the same as in Q-learning, except that the set of available actions in state *s*, *Jk(s)*, is a function of the previous history of agent *k*

Can be <u>local</u> or <u>global</u>:

1. Global-best
2. Iteration-best

the length of the tour done by the best ant

$$\Delta AQ(r,s) = \begin{cases} \dfrac{W}{L_{Best}} & \text{if } (r,s) \in \text{ tour done by the best agent} \\ 0 & \text{otherwise} \end{cases}$$

$$\Delta AQ(r,s) = \begin{cases} \dfrac{W}{L_{k_{ib}}} & \text{if } (r,s) \in \text{ tour done by agent } k_{ib} \\ 0 & \text{otherwise} \end{cases}$$

K_ib is the agent who made the best tour in the current iteration of the trial

```
Ant-Q algorithm
/* Initialization phase */
   Set an initial value for AQ-values
/* Main algorithm */
Loop  /* This loop is an iteration of the algorithm */
  1.    /* Initialization of ants data structures */
        Choose a starting city for ants
  2.    /* In this step ants build tours and locally update AQ-values */
        Each ant applies the state transition rule (1) to choose the city to
        go to, updates the set J_k and applies formula (3) to locally update
        AQ-values (in formula (3) ΔAQ(r,s)=0)
  3.    /* In this step ants globally update AQ-values */
        The edges belonging to the tour done by the best ant are updated using
        formula (3) where ΔAQ(r,s) is given by formula (4)
Until (End_condition = True)
```

Conclusion

Table 4: Comparisons on average result obtained on five 50-city problems. EN = elastic net, SA = simulated annealing, SOM = self organizing map, FI = farthest insertion, FI+2-opt = best solution found by FI and many distinct runs of 2-opt, FI+3-opt = best solution found by FI and many distinct runs of 3-opt. Results on EN, SA, and SOM are from Durbin and Willshaw (1989), and Potvin (1993). FI results are averaged over 15 trials starting from different initial cities. Ant-Q used pseudo-random-proportional action choice and iteration-best delayed reinforcement. It was run for 500 iterations and the results are averaged over 15 trials.

City set	EN	SA	SOM	FI	FI + 2-opt	FI + 3-opt	Ant-Q
1	5.98	5.88	6.06	6.03	5.99	5.90	**5.87**
2	6.03	**6.01**	6.25	6.28	6.20	6.07	6.06
3	5.70	5.65	5.83	5.85	5.80	5.63	**5.57**
4	5.86	5.81	5.87	5.96	5.96	5.81	**5.76**
5	6.49	6.33	6.70	6.71	6.61	6.48	**6.18**