Ant Q Algorithm

and Electric Vehicle Routing problem

Tan Pham Ngoc

Faculty of Computer Science
University of Information Technology, VNU HCM

March 23, 2021

Table of Contents

Table of Contents

Q-Learning

BIRTH
CHILDHOOD
SCHOOL
UNIVERSITY
WORK
FAMILY
CHILDREN
WORK-HOME
OLD AGE

Q-Learning

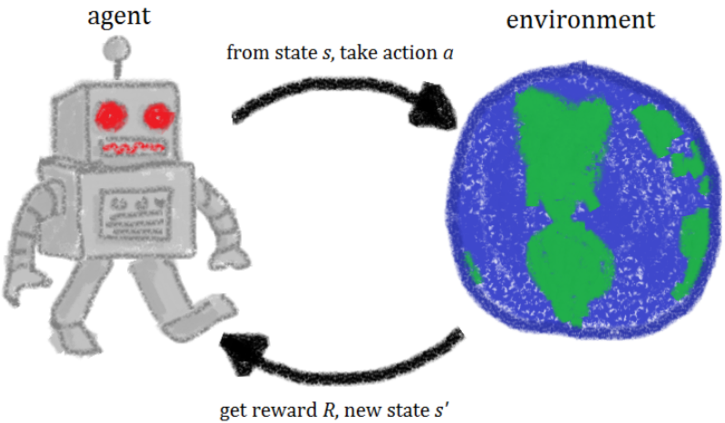


Figure: Q-learning description

Table of Contents

Ant Colonization Optimization

- Marco Dorigo first introduced ACO in the early 90s

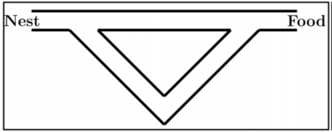
Ant Colonization Optimization

- Marco Dorigo first introduced ACO in the early 90s
- The algorithm's development was inspired by the observation of the ant colonies

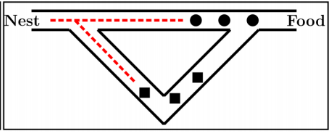
Ant Colonization Optimization

- Marco Dorigo first introduced ACO in the early 90s
- The algorithm's development was inspired by the observation of the ant colonies
- The behavior that provided the inspiration for ACO is the ants' foraging behavior

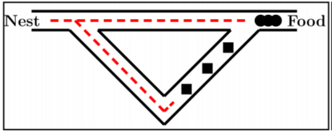
Ant Colonization Optimization



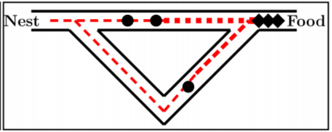
(a) All ants are in the nest. There is no pheromone in the environment.



(b) The foraging starts. In probability, 50% of the ants take the short path (symbolized by circles), and 50% take the long path to the food source (symbolized by rhombs).



(c) The ants that have taken the short path have arrived earlier at the food source. Therefore, when returning, the probability to take again the short path is higher.



(d) The pheromone trail on the short path receives, in probability, a stronger reinforcement, and the probability to take this path grows. Finally, due to the evaporation of the pheromone on the long path, the whole colony will, in probability, use the short path.

Table of Contents

Introduction

- Ant-Q is a family of algorithms which present many similarities with Q-learning

Introduction

- Ant-Q is a family of algorithms which present many similarities with Q-learning
- It was inspired by the work on both the Dorigo et al, 1992, *Ant System* and by Watkins, 1989, *Q-Learning*

Introduction

The core difference between Ant-Q and Q-learning:

Introduction

The core difference between Ant-Q and Q-learning:

- Ant-Q uses *a set of* cooperating agents

Introduction

The core difference between Ant-Q and Q-learning:

- Ant-Q uses *a set of* cooperating agents
- These agents will cooperate exchanging information to each other in the form of *AQ-values*

Introduction

The core difference between Ant-Q and Q-learning:

- Ant-Q uses *a set of* cooperating agents
- These agents will cooperate exchanging information to each other in the form of *AQ-values*
- Over and above that, Ant-Q agents have memory

Table of Contents

Mathematical view on Ant-Q

- A graph (N, E)

Mathematical view on Ant-Q

- A graph (N, E)
- A distance d_{rs} for each pair of cities

Mathematical view on Ant-Q

- A graph (N, E)
- A distance d_{rs} for each pair of cities
  - Symmetric: $d_{rs} = d_{sr}$
  - Asymmetric: $d_{rs} \neq d_{sr}$

Mathematical view on Ant-Q

Let denote:

Let denote:

- $AQ(r, s)$ as *Ant-Q value*: $AQ(r, s) \in \mathbb{R}^+$

Mathematical view on Ant-Q

Let denote:

- $AQ(r,s)$ as *Ant-Q value*: $AQ(r,s) \in \mathbb{R}^+$

$\rightarrow$ The usefulness to move from city r to city s

Mathematical view on Ant-Q

Let denote:

- $AQ(r, s)$ as *Ant-Q value*: $AQ(r, s) \in \mathbb{R}^+$

$\rightarrow$ The usefulness to move from city r to city s

- $AQ'(r, s)$ as *optimal Ant-Q value*

Mathematical view on Ant-Q

In the event of our graph:

- Symmetric:
- Asymmetric:

Mathematical view on Ant-Q

In the event of our graph:

- Symmetric:

$$AQ(r, s) = AQ'(r, s)$$

- Asymmetric:

Mathematical view on Ant-Q

In the event of our graph:

- Symmetric:

$$AQ(r,s) = AQ'(r,s)$$

- Asymmetric:

$$[AQ(r,s) = AQ'(r,s)] \vee [AQ(r,s) \neq AQ'(r,s)]$$

Mathematical view on Ant-Q

Also, let define:

Also, let define:

- $HE(r, s)$ as *Heuristic values*

Also, let define:

- $HE(r,s)$ as *Heuristic values*

$$HE(r,s) = \frac{1}{distance} = \frac{1}{d_{rs}}$$

Mathematical view on Ant-Q

Also, let define:

- $HE(r, s)$ as *Heuristic values*

$$HE(r, s) = \frac{1}{distance} = \frac{1}{d_{rs}}$$

$\rightarrow$ An heuristic evaluation of which move is better

Mathematical view on Ant-Q

Also, let define:

- $HE(r, s)$ as *Heuristic values*

$$HE(r,s) = \frac{1}{distance} = \frac{1}{d_{rs}}$$

  $\rightarrow$ An heuristic evaluation of which move is better

- k as the *agent*, of which mission is to make tours

Mathematical view on Ant-Q

Also, let define:

- $HE(r,s)$ as *Heuristic values*

$$HE(r,s) = \frac{1}{distance} = \frac{1}{d_{rs}}$$

  $\rightarrow$ An heuristic evaluation of which move is better
- k as the *agent*, of which mission is to make tours
- $J_k(r)$ as a list of cities *to be visited* from the current city r

With all of that, we have our *action choice rule*

Action choice rule

Action choice rule can be understood easily as the rule for an agent k in city r move to city s:

Action choice rule

Action choice rule can be understood easily as the rule for an agent k in city r move to city s:

$$s = \begin{cases} argmax_{u \in J_k(r)} \{[AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta}\} & if \ q \leq q_0 \\ S & otherwise \end{cases}$$

Action choice rule

Action choice rule can be understood easily as the rule for an agent k in city r move to city s:

$$s = \begin{cases} argmax_{u \in J_k(r)} \{[AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta}\} & if \ q \leq q_0 \\ S & otherwise \end{cases}$$

s.t.

- δ, β: parameters weighing the relative importance of learning rate AQ and HE

Action choice rule

Action choice rule can be understood easily as the rule for an agent k in city r move to city s:

$$s = \begin{cases} argmax_{u \in J_k(r)} \{[AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta}\} & if\ \ q \leq q_0 \\ S & otherwise \end{cases}$$

s.t.

- δ, β: parameters weighing the relative importance of learning rate AQ and HE
- q: uniformly distributed value in $[0; 1]$
- q_0: parameter following by the updated rule

Action choice rule

Action choice rule can be understood easily as the rule for an agent k in city r move to city s:

$$s = \begin{cases} argmax_{u \in J_k(r)} \{[AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta}\} & if\ q \leq q_0 \\ S & otherwise \end{cases} \tag{1}$$

s.t.

- δ, β: parameters weighing the relative importance of learning rate AQ and HE
- q: uniformly distributed value in $[0; 1]$
- q_0: parameter following by the updated rule
- S: a random variable selected according to a probability distribution given by function $AQ'(r,u)$ and $HE'(r,u)$ with $u \in J_k(r)$

Why multiply ?

Action choice rule

There are 3 main action choice rules:

- Pseudo-random
- Pseudo-random-proportional
- Random-proportional

Action choice rule

There are 3 main action choice rules:

- **Pseudo-random**
- Pseudo-random-proportional
- Random-proportional

Pseudo-random

The formula (1):

$$s = \begin{cases} argmax_{u \in J_k(r)} \{[AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta}\} & if \;\; q \leq q_0 \\ S & otherwise \end{cases}$$

s.t. S, which is a random variable over the set $J_k(r)$, is selected according the uniform distribution

Pseudo-random

The formula (1):

$$s = \begin{cases} argmax_{u \in J_k(r)} \{[AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta}\} & if\ q \leq q_0 \\ S & otherwise \end{cases}$$

s.t. S, which is a random variable over the set $J_k(r)$, is selected according the uniform distribution

$\rightarrow$ **Strongly resembles the action choice rule in Q-Learning**

Action choice rule

There are 3 main action choice rules:

- Pseudo-random
- **Pseudo-random-proportional**
- Random-proportional

Pseudo-random-proportional

In the formula (1), S is a random variable over the set N, selected according to the distribution given below:

$$p_k(r,s) = \begin{cases} \frac{[AQ(r,s)]^{\delta} \cdot [HE(r,s)]^{\beta}}{\sum_{u \in J_k(r)} \{[AQ(r,u)^{\delta} \cdot HE(r,u)^{\beta}]\}} & if \ \ s \in J_k(r) \\ 0 & otherwise \end{cases}$$

$\rightarrow$ The probability with which an agent in city r chooses the city s to move to

Action choice rule

There are 3 main action choice rules:

- Pseudo-random
- Pseudo-random-proportional
- **Random-proportional**

Random-proportional

- Basically, this rule is the same as the pseudo-random-proportional in which $q_0 = 0$
- Specifically, the choice of the next city is always done by using random selection where edges are chosen with a probability distribution given by formula (2)

$$p_k(r,s) = \begin{cases} \frac{[AQ(r,s)]^{\delta} \cdot [HE(r,s)]^{\beta}}{\sum_{u \in J_k(r)} \{[AQ(r,u)^{\delta} \cdot HE(r,u)^{\beta}]\}} & \textit{if } s \in J_k(r) \\ 0 & \textit{otherwise} \end{cases}$$

Random-proportional

- Basically, this rule is the same as the pseudo-random-proportional in which $q_0 = 0$
- Specifically, the choice of the next city is always done by using random selection where edges are chosen with a probability distribution given by formula (2)

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

$\rightarrow$ The same as Ant-System

Agent's update rule

Agent's update rule

If the Q-value's update rule in DQN is shown as:

$$Q_{(s,a)} \leftarrow (1-\alpha)\, Q\,(s,a) + \alpha \left(R_{t+1} + \gamma \max_{a'} Q\,(s',a') \right)$$

Agent's update rule

Then the AQ-value in Ant-Q is updated as:

$$AQ(r,s) \leftarrow (1-\alpha)AQ(r,s) + \alpha \left(\Delta AQ(r,s) + \gamma \max_{z \in J_k(s)} AQ(s,z) \right)$$

Agent's update rule

Then the AQ-value in Ant-Q is updated as:

$$AQ(r,s) \leftarrow (1-\alpha)AQ(r,s) + \alpha\left(\Delta AQ(r,s) + \gamma \max_{z \in J_k(s)} AQ(s,z)\right)$$

in which:

Agent's update rule

Then the AQ-value in Ant-Q is updated as:

$$AQ(r,s) \leftarrow (1-\alpha)AQ(r,s) + \alpha\left(\Delta AQ(r,s) + \gamma \max_{z \in J_k(s)} AQ(s,z)\right)$$

in which:

- α: learning step
- γ: discounted factor

Agent's update rule

Then the AQ-value in Ant-Q is updated as:

$$AQ(r,s) \leftarrow (1-\alpha)AQ(r,s) + \alpha\left(\Delta AQ(r,s) + \gamma \max_{z \in J_k(s)} AQ(s,z)\right)$$

in which:

- α: learning step
- γ: discounted factor
- $\Delta AQ(r,s)$: the delayed reinforcement

Agent's update rule

Then the AQ-value in Ant-Q is updated as:

$$AQ(r,s) \leftarrow (1-\alpha)AQ(r,s) + \alpha\left(\Delta AQ(r,s) + \gamma \max_{z\in J_k(s)} AQ(s,z)\right) \quad (2)$$

in which:

- α: learning step
- γ: discounted factor
- $\Delta AQ(r,s)$: the delayed reinforcement

$\Leftrightarrow$ The update rule (3) is the same as the update rule in Q-learning, but we use the set $J_k(r)$ instead of the set of available actions in state s

What is the delayed reinforcement ?

The delayed reinforcement

- Q-Learning's rewards

The delayed reinforcement

- Q-Learning's rewards $\rightarrow$ **Ant-Q's delayed reinforcement**

The delayed reinforcement

- Q-Learning's rewards $\rightarrow$ **Ant-Q's delayed reinforcement**
- 2 types

The delayed reinforcement

- Q-Learning's rewards $\rightarrow$ **Ant-Q's delayed reinforcement**
- 2 types
  - Global best
  - Iteration best

Global-best

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

Global-best

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

s.t.

- W is often set to 10

Global-best

$$\Delta AQ(r,s) = \begin{cases} \frac{W}{L_{k_{gb}}} & \textit{if } (r,s) \in \text{tour done by agent } k_{gb} \\ 0 & \textit{otherwise} \end{cases} \tag{3}$$

s.t.

- W is often set to 10
- $L_{k_{gb}}$ is the tour made by the agent made the globally best tour from the beginning

Iteration-best

Iteration-best

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

Iteration-best

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

s.t.

- k_{ib} is the agent who made best tour in the current iteration of the trial

Ant-Q vs. Q-Learning

Ant-Q versus Q-Learning

```
1./* Initialization phase */
   For each pair (r,s) AQ(r,s):= AQ0  End-for
     For k:=1 to m do
         Let rk1 be the starting city for agent k
         Jk(rk1):= {1, ..., n} - rk1
         /* Jk(rk1) is the set of yet to be visited cities for agent k in city rk1 */
         rk:=rk1
         /* rk is the city where agent k is located */
     End-for
2. /* This is the step in which agents build their tours. The tour of agent k is stored in
   Tourk.  Given that local reinforcement is always null, only the next state evaluation is used
   to update AQ-values. */
   For i:=1 to n do
    If i≠n
       Then
         For k:=1 to m do
             Choose the next city sk according to formula (1)
             If i≠n-1 Then Jk(sk):= Jk(rk) - sk
             If i=n-1 Then Jk(sk):= Jk(rk) - sk + rk1
             Tourk(i):=(rk,sk)
         End-for
       Else
         For k:=1 to m do  /* In this cycle all the agents go back to the initial city rk1 */
             sk := rk1
             Tourk(i):=(rk,sk)
         End-for
     For k:=1 to m do
         AQ(rk,sk):=(1-α)AQ(rk,sk)+α·γ· Max        AQ(sk,z)
                                       z∈Jk(sk)
         /* This above is formula (2), where the reinforcement ΔAQ(rk,sk) is always null */
         rk := sk  /* New city for agent k */
     End-for
   End-for
3. /* In this step delayed reinforcement is computed and AQ-values are updated using formula
   (2), in which the next state evaluation term γ·Max AQ(rk1,z) is null for all z */
   For k:=1 to m do
     Compute Lk  /*Lk is the length of the tour done by agent k*/
   End-for
   For each edge (r,s)
     Compute the delayed reinforcement ΔAQ(r,s)
   /*The delayed reinforcement ΔAQ(r,s) is a function of Lk's */
   End-for
   Update AQ-values applying a formula (2)
4. If (End_condition = True)
     then Print shortest of Lk
     else goto Step 2
```