Ant-Q

A Reinforcement Learning approach to the traveling salesman problem

Table of contents

01

Background

The ideas behind the Ant-Q Algorithm

Ant System

- A colony of cooperating ants leaving pheromone trails on the paths to find food.
- Random exploration → following paths with familiar pheromones.
- Pheromones can evaporate as time passes.
- Paths that are shorter will have less pheromone evaporation.

Q-learning

- Reinforcement Learning with states, rewards, and actions.
    - Finite states
    - Finite actions
- Model-free environment - interacting directly with the environment to find optimal policy instead of creating a model.
- Trial-and-error - does many trials and updates its policy as it learns
- Q-values: Q(s, a) - evaluation of the quality of action a in state s.
    - Current estimate of sum of future rewards if we take action a.
- Q-table: gives Q-values for every action in every state.
    - Rows: states
    - Columns: actions
    - Use TDs to update previous Q-values after evaluating current state/actions.

Q-learning

Temporal Difference

The reward received for the action taken in the previous state

The largest Q-value available for any action in the current state (the largest predicted sum of future rewards)

$$TD(s_t, a_t) = r_t + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$$

Temporal difference for the action taken in the previous state

The discount factor (between 0 and 1)

The Q-value for the action taken in the previous state

Bellman Equation

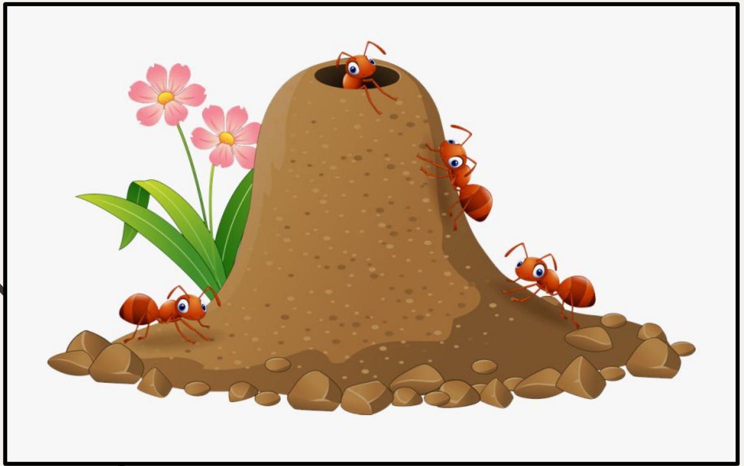
The old Q-value for the action taken in the previous state

The temporal difference for the action taken in the previous state

$$Q^{new}(s_t, a_t) = Q^{old}(s_t, a_t) + \alpha \cdot TD(s_t, a_t)$$

The new Q-value for the action taken in the previous state

The learning rate (between 0 and 1)



02

Ant-Q

The Ant-Q algorithm

Ant-Q Algorithm

Travelling Salesman Problem

- Goal: find a minimal length closed tour that visits each city once
- *n* cities
- Each pair of cities has distance d_{rs}
- Connected graph with (N,E), N = set of n nodes, E = set of edges between cities
- HE(r,s) - heuristic evaluation of edge (r,s) - inverse of the distance
- AQ(r,s) - How useful it is to go to city s when at city r

Ant-Q Algorithm

Action Choice Rule

- Agent k - makes a tour
- Has a list $J_k(r)$ of cities that need to be visited. r = current city

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

Ant-Q Algorithm

AQ-value Updates

- Similar to updating Q-values in Q-learning
- Includes delayed reinforcement value ΔAQ(r,s)

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

Ant-Q Algorithm Steps

Step 1: Initialize

1. AQ-values
2. Multiple agents, each agent is placed on a city
3. $J_k(r_{k1})$ - set of cities that need to be visited

Step 2: Cycle

1. Each agent makes a move
2. AQ(r,s)'s are updated

Step 3: Delayed Reinforcement

1. Length L_k of each agent's tour is computed
2. Use lengths to compute delayed reinforcements
3. AQ(r,s)'s are updated with delayed reinforcements

Step 4: Termination Check

1. Check if termination condition is met
2. If not, return to step 2.

03

Structural Parameters

The Action Choice Rule and Delayed Reinforcement

Action-Choice Rule

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

Delayed Reinforcement

- Global-best
- Iteration-best

The Action-Choice Rule

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

(1)

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

(2)

All use (1) to determine which city to go next.

Pseudo-random Rule

- Uniform Distribution

Pseudo-random-proportional Rule

- The distribution in (2)

Random-proportional Rule

- Same as Pseudo-random-proportional, but with q_0 = 0. The choice of the next city is random, chosen with distribution in (2).

The Action-Choice Rule

	Pseudo-random				Pseudo-random-proportional				Random-proportional			
	γ	mean	std dev	best	γ	mean	std dev	best	γ	mean	std dev	best
City Set 1	0.5	6.18	0.06	6.03	0.3	5.87	0.05	5.84	0.9	7.85	0.25	7.40
City Set 2	0.5	6.26	0.04	6.20	0.3	6.06	0.05	5.99	0.9	7.77	0.30	7.43
City Set 3	0.5	5.69	0.07	5.61	0.3	5.57	0.00	5.57	0.9	7.89	0.17	7.75
City Set 4	0.5	5.92	0.05	5.84	0.3	5.76	0.03	5.70	0.9	7.95	0.10	7.85
City Set 5	0.5	6.30	0.04	6.22	0.3	6.18	0.01	6.17	0.9	8.48	0.21	8.10
Oliver30	0.5	425.02	1.22	424.69	0.3	424.44	0.46	423.74	0.9	515.19	10	493.20
ry48p	0.3	15602	440	14848	0.3	14690	175	14422	0.9	19495	797	17921

Delayed Reinforcement

Global-best

- Globally best tour from the beginning of the trial.
- Only the AQ-values for edges in the globally best tour will be reinforced.

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

(1)

Iteration-best

- Best tour in the current iteration of the trial.
- Slightly faster with same quality.
- Less sensitive to changes of discount factor γ.

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

(2)

The Action-Choice Rule

	Ant-Q Global-best			Ant-Q Iteration-best		
	mean	std. dev.	best	mean	std. dev.	best
City Set 1	5.90	0.08	5.84	5.87	0.05	5.84
City Set 2	6.05	0.04	5.99	6.06	0.05	5.99
City Set 3	5.58	0.01	5.57	5.57	0.00	5.57
City Set 4	5.76	0.03	5.70	5.76	0.03	5.70
City Set 5	6.20	0.03	6.17	6.18	0.01	6.17
Oliver 30	424.37	0.43	423.74	424.44	0.46	423.74
ry48p	14697	157	14442	14690	157	14422

Comparisons - Ant System

Delayed Reinforcement

$$\Delta AQ(r,s) = \sum_{k=1}^{m} \Delta AQ_k(r,s)$$

$$\Delta AQ_k(r,s) = \begin{cases} \frac{W}{L_k} & \text{if } (r,s) \in \text{tour done by agent k} \\ 0 & \text{otherwise} \end{cases}$$

AQ-value Updates

$AQ(r,s) \leftarrow (1 - \alpha) \times AQ(r,s) + \Delta AQ(r,s)$

- Applies to all edges
- Simulate pheromones and pheromone evaporation

	Ant-Q			Ant system		
	mean	std. dev.	best	mean	std. dev.	best
6x6 grid	360	0	360	360	0	360
Oliver 30	424.44	0.46	423.74	425.46	0.51	423.74
ry48p	14690	157	14422	14889	223	14803

04

Results

Properties and Comparisons

Observations and Characteristics

Agents do not make the same tour.

- Agents do not converge to a common path.
- λ-branching factor - shows the dimension of the search space.
- Number of edges that have an AQ-value that is larger than $\lambda(AQ_{max}(r,s) - AQ_{min}(r,s)) + AQ_{min}(r,s)$.
- $0 \leq \lambda \leq 1$
- The search space is reduced, but agents continue to explore a subset of the search space.

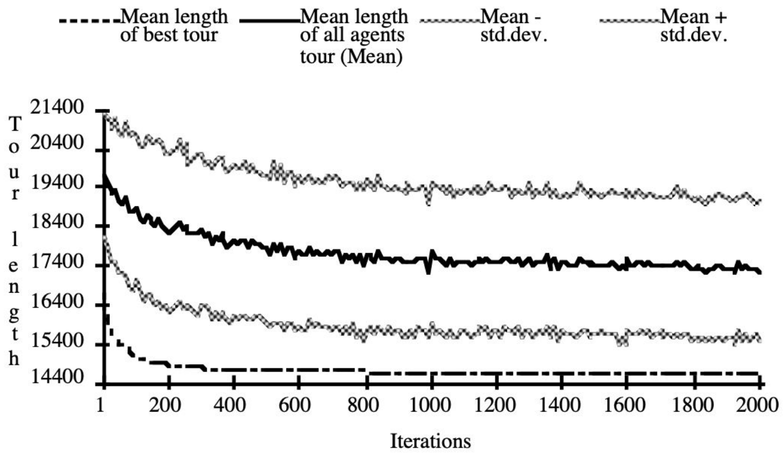


Figure 2: Mean length of best tour, mean length of all agents tour, and its std. dev. Problem: ry48p. Averaged over 15 trials.

Observations and Characteristics

Agents do not make the same tour.

- Agents do not converge to a common path.
- λ-branching factor - shows the dimension of the search space.
- Number of edges that have an AQ-value that is larger than $\lambda(AQ_{max}(r,s) - AQ_{min}(r,s)) + AQ_{min}(r,s)$.
- $0 \le \lambda \le 1$
- The search space is reduced, but agents continue to explore a subset of the search space.

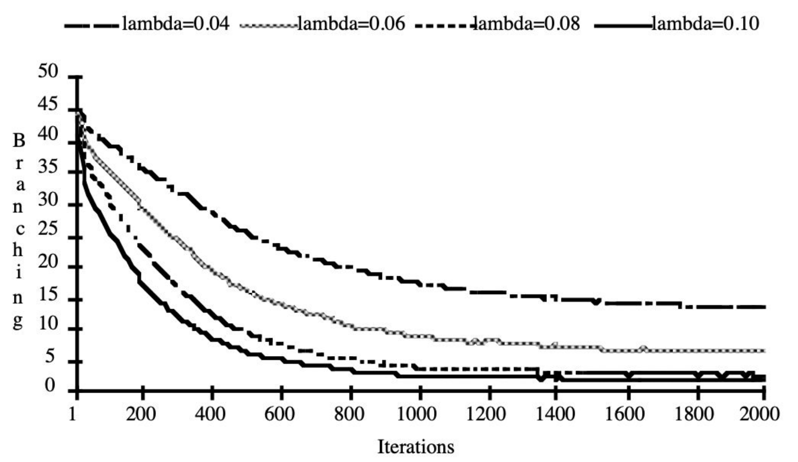


Figure 3: λ–branching factor. Problem: ry48p. Averaged over 15 trials.

Observations and Characteristics

AQ-values are exploited by agents to find short tours.

As iterations increase and good AQ-values are learned:

- AQ-values become more effective in finding good solutions.
- Heuristic values become less effective, even useless.

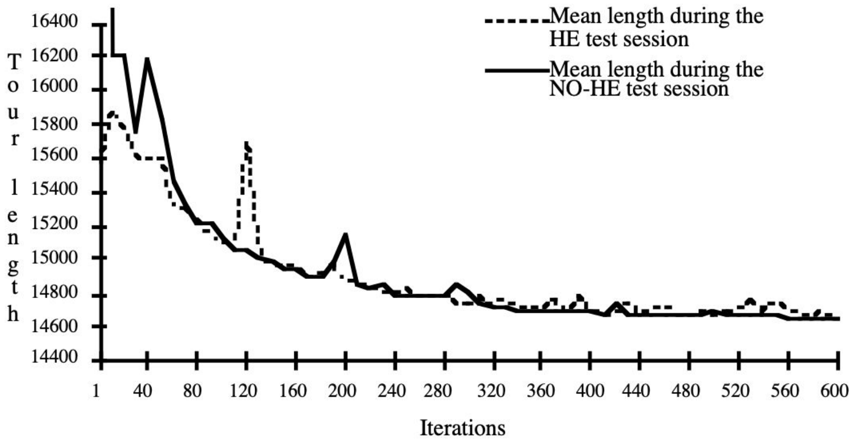


Figure 4: Best tour found during test session using only the AQ-values (NO-HE test session), and using both the AQ-values and the HE heuristic values (HE test session). The test session was run every ten learning iterations. Problem: ry48p. Averaged over 15 trials.

Comparison

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**

Comparison

Table 5: Comparison between the best results obtained by SA+3-opt = best solution found by simulated annealing and many distinct runs of 3-opt, SOM+ = best solution found by SOM over 4,000 different runs (by processing the cities in various orders), FI and its locally optimized versions, and Ant-Q. The 2-opt and 3-opt heuristics used the result of FI as starting configuration for local optimization. Results on SA+3-opt and SOM+ are from Durbin and Willshaw (1989), and Potvin (1993). Ant-Q used pseudo-random-proportional action choice and iteration-best delayed reinforcement. It was run for 500 iterations, and the best result was obtained out of 15 trials.

City set	SA + 3-opt	SOM+	FI	FI + 2-opt	FI + 3-opt	Ant-Q
1	**5.84**	**5.84**	5.89	5.85	5.85	**5.84**
2	**5.99**	6.00	6.02	6.01	**5.99**	**5.99**
3	**5.57**	5.58	**5.57**	**5.57**	**5.57**	**5.57**
4	5.70	**5.60**	5.76	5.76	5.70	5.70
5	**6.17**	6.19	6.50	6.45	6.40	**6.17**

Comparison

Table 6: Comparison between exact methods and Ant-Q for difficult ATSP problems. Numbers in parenthesis are seconds. Type of delayed reinforcement: global-best. For the problem 43X2 we set γ=0.01. Ant-Q was run for 600 iterations, and results were obtained out of 15 trials.

Problem	*FT-92*	*FT-94*	*Ant-Q Mean*	*Ant-Q Best result*
ry48p	14422 (729.6)	14422 (52.8)	14690 (1590)	14422 (696)
43X2	N/A	5620 (492.2)	5625 (550)	5620 (238)

Conclusions

- Making Ant-Q more like Q-learning.
- Extend and apply Ant-Q to other combinatorial optimization problems.

N-Stroll

- Implement delayed reinforcement.
- Try using different action-choice rules.

Difficulties

- Implementing an ‘n’ requirement (could possibly use delayed reinforcement for that)
- Implementing different actions for different states (since each layer of nodes have different actions we can take).
- Whether or not we should implement multiple agents.

Works Cited

Dorigo M., V.Maniezzo and A.Colorni, 1996. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics*, 26, 2, in press.

Gambardella, L. M., & Dorigo, M. (1995). Ant-Q: A Reinforcement Learning approach to the traveling salesman problem. *Machine Learning Proceedings 1995*, 252–260. https://doi.org/10.1016/b978-1-55860-377-6.50039-6

Transparent Hill Png - Anthill Clipart. (n.d.). Clipartkey. https://www.clipartkey.com/mpngs/m/249-2494778_transparent-hill-png-anthill-clipart.png.

Thanks

Do you have any questions?