# 1 Background material

*Ant colony optimisation* (ACO) is a technique for finding approximate solutions to combinatorial optimisation problems which can be reduced to finding *good* paths through graphs. This paradigm mirrors the pheromone-based communication of biological ants. Artificial *ants* (agents) locate feasible solutions and record the *quality* of each solution (analogous to pheromone trails laid by ants). Ants will use this record of solution quality when locating new feasible solutions.

The *ant system* is an early ant colony optimisation algorithm which is the basis for the extension of *Ant-Q*, and was introduced by Dorigo, Maniezzo, and Colorni [1]. Although introduced with its application to the *symmetric travelling salesman problem* (TSP), it was shown to generalise to the *asymmetric travelling salesman problem* (ATSP), and in fact any *appropriate* graph representation, though a strict definition of what this is in not clear. Thus, in the succeeding sections we will be dealing with the application of ACO to ATSP, but we note its application to any problem that can be reduced to ATSP (of which there are many).

We introduce asymmetric travelling salesman problem (ATSP). Given a set $N$ of $n$ cities and for each $r, s \in N$ a distance $d_{rs}$, ATSP is the problem of finding a minimal length closed tour that visits each city only once. This problem can be realised by an directed weighted graph $(N, E)$ where edge $(r, s) \in E$ has weight $d_{rs}$.

We now introduce the notion of reinforcement learning, and encourage the reader to draw similarities between it and ant colony optimisation.

*Reinforcement learning* concerns how agents make actions in an environment in order to maximise cumulative reward. It involves one or many agents, each having a state from some set of states $S$, and able to take an action from some set of actions $A$. By taking an action, agents may transition to different states, and in doing so is given a reward $r \in \mathbb{R}$. The goal of the agent is to maximise *cumulative reward*; that is, the total reward at the end of its *episode* (when it reaches a terminal state). We may describe an episode of an agent as a sequence of states, actions, and rewards: $e = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_n)$ where $s_n$ is a terminal state.

We now specialise to *Q-learning*, a model-free (by this we mean that we need no model of the environment) reinforcement learning algorithm that quantifies the *quality* of actions an agent may take at a given state.

The core of *Q-learning* is the *Q-function* $Q : S \times A \to \mathbb{R}$ that predicts the quality of an action at a given state (we call the evaluation of the Q-function at a given state-action a *Q-value*). We first initialise $Q$ for each state-action (typically to a fixed value). Agents in a state $s$ will choose action $a_s$ by considering the Q-value, and for it receives a reward $r_t$. The agent will then move into state $s_{t+1}$ and the Q-value for that state-action will be updated by $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\,(r_t + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t))$ where $\alpha \in (0, 1]$ is the *learning rate* (which determines how sceptical the agent is to new information) and $\gamma \in [0, 1]$ is the *discount factor* (which dictates how far-sighted the agent's decisions should be).

# 2 The Ant-Q algorithm, pseudocode

**function** INITIALISE
  **for** each ant $k$ **do**
    pick an initial city for $k$
  **end for**
  **for** $s \in S$ **do**
    **for** $a \in A_s$ **do**
      $\mathrm{AQ}(s, a) \leftarrow \mathrm{AQ}_0$ ▷ initialise AQ values
    **end for**
  **end for**
**end function**
**function** DOEPISODE
  **for** $t \leftarrow 1$ to $n$ **do**
    **for** each ant $k$ **do**
      choose next city $a_t^k \in A_{s_t^k}$ ▷ according to Equation (2)
      $\mathrm{Ep}_k(t) \leftarrow (s_t^k, a_t^k)$ ▷ record ants action choice
    **end for**
    **for** each ant $k$ **do**
      update $\mathrm{AQ}(s_t^k, a_t^k)$ ▷ according to Equation (1), note $\Delta\mathrm{AQ} = 0$
      $s_{t+1} \leftarrow a_t^k$ ▷ sets the state of the ant for the next step
    **end for**
  **end for**

```
end function
function UPDATE
    for each ant k do
        L_k ← length of k's tour
    end for
    for s ∈ S do
        for a ∈ A_s do
            compute ΔAQ(r, s)                    ▷ this is done dependent on the variant
        end for
    end for
    update AQ values                             ▷ according to Equation (1), note ΔAQ ≠ 0
end function
function ANT-Q
    while not endCondition do
        INITIALISE
        DOEPISODE
        UPDATE
    end while
end function
```

# 3 The Ant-Q algorithm, plain English

As discussed we will introduce the Ant-Q algorithm with its application to ATSP. Let $(N, E)$ be the weighted directed graph as discussed with $|N| = n$.

We start with a set of agents (ants) $P$. Our set of states $S$ is just $N$ (at any given point, an ant is at a city). The set of all actions $A$ is also $N$; however, we will later restrict this set to allow only valid tours.

At the start of our algorithm, $t = 1$, each ant $k \in P$ is given an initial state $s_1^k \in S$. Ant $k$ will then choose an action $a_1^k \in A$, perform the action, then move into state $s_2^k = a_1^k \in S$. More generally, at time $t \in \mathbb{N}$, ant $k$ will be in state $s_t^k$. If $t = n$, the simulation is over. Otherwise, ant $k$ will choose an action $a_t^k \in A$ and move into state $s_{t+1}^k = a_t^k \in S$.

We have some restriction to impart before moving forward. In ATSP, we must find a *Hamiltonian cycle*; that is, we can only visit each vertex once and must return to the original vertex. To deal with this, we restrict the choice of actions of the ants. At time $t \in \mathbb{N}$, we denote $A_t^k$ as the set of possible actions an ant $k \in P$ can take. For $t < n$, this is defined by $A_t^k = N \setminus \{s_i^k\}_{i \in \{1,\dots,t\}}$ and $A_n^k = \{s_1^k\}$.

In the above description, we are missing how ant $k$ chooses which action to take. For this, we introduce two constructions.

We define the function $\mathrm{AQ} : S \times A \to \mathbb{R}_{\geq 0}$ such that for a state $s \in S$ and action $a \in A_s$, $\mathrm{AQ}(s, a)$ indicates how *useful* action $a$ is when an ant in state $s$. This is analogous to Q-values in Q-learning, and mirrors similarities to Q-learning in how this value is maintained. When an ant in state $s \in S$ takes the action $a \in A_s$ and moves to state $s' \in S$, we update the AQ value by

$$\mathrm{AQ}(s, a) \leftarrow \mathrm{AQ}(s, a) + \alpha \left( \Delta\mathrm{AQ}(s, a) + \gamma \max_{a' \in A_s} \mathrm{AQ}(s, a') - \mathrm{AQ}(s, a) \right) \tag{1}$$

where $\Delta\mathrm{AQ} : S \times A \to \mathbb{R}$ is a value called *delayed reinforcement*, and is maintained throughout the execution of the algorithm. It is is analogous to *reward* in Q-learning. It is an unspecified parameter for the model, but we will discuss some choices evaluated by Gambardella and Dorigo [3] later.

Let $\mathrm{HE} : S \times A \to \mathbb{R}_{\geq 0}$ be a heuristic function, which is not updated during the runtime of our algorithm. This also indicates the quality of an action given that an ant is in a particular state. For example, we may pick the inverse or negative of the distances between points.

We now describe how an ant $k \in P$ at time $t$ may choose an action $a_t^k$ given that it is in state $s_t^k$. We do this by the *action choice rule*, given by

$$a_t^k = \begin{cases} \arg\max_{a \in A_{s_t^k}} \left(\mathrm{AQ}(s_t^k, a)\right)^\delta \left(\mathrm{HE}(s_t^k, a)\right)^\beta & \text{if } q \leq q_0, \\ S & \text{otherwise,} \end{cases} \tag{2}$$

where $\delta, \beta \in \mathbb{R}$ are parameters which weight the importance of AQ and HE, $q \in [0,1]$ is a value that is chosen uniformly at random, $q_0 \in [0,1]$ is a parameter which dictates how often to take a random action, and $S$ is a random variable selected according to a probability density function given by the AQ and HE values of each possible action. Some choices of these parameters will be later discussed.

# 4 Comparison of various Ant-Q algorithms

The experiments on this section were conducted on grid problems, Oliver30 (30-city TSP), ry48p (48 city ATSP), and random data sets.

## 4.1 The action choice rule

Gambardella and Dorigo [3] presents and compares three different action choice rules; that is, how ants in a given state choose their next action. We let $C : S \to A$ be the action choice function (that is, an ant in state $s$ will choose action $C(s) \in A_s$). In each of the three rules present, $C$ takes the form

$$a = C(s) = \begin{cases} \arg\max_{a \in A_s} \left(\mathrm{AQ}(s,a)\right)^{\delta} \left(\mathrm{HE}(s,a)\right)^{\beta} & \text{if } q \leq q_0, \\ S & \text{otherwise,} \end{cases}$$

where $q_0$ is a parameter to be assigned in the variants, $q$ is a random variable which has uniform distribution over $[0,1]$, and $S$ is are random variable to be assigned in the variants below.

(A1) (Pseudo-random) The random variable $S$ has uniform distribution over $A_s$ and $q_0 \in [0,1]$ is unspecified.

(A2) (Pseudo-random-proportional) The random variable $S$ has the following probability density function

$$p(s,a) = \begin{cases} \dfrac{\left(\mathrm{AQ}(s_t,a)\right)^{\delta} \left(\mathrm{HE}(s_t,a)\right)^{\beta}}{\sum_{a' \in A_s} \left(\mathrm{AQ}(s_t,a')\right)^{\delta} \left(\mathrm{HE}(s_t,a')\right)^{\beta}} & \text{if } a \in S_a, \\ 0 & \text{otherwise} \end{cases}$$

and $q_0 \in [0,1]$ is unspecified.

(A3) (Random-proportional) $S$ is specified as in (A2) but $q_0 = 0$.

It is noted that (A1) closely mimics to pseudo-random action choice rule of Q-learning: agents will choose the best action (based on the Q-values), otherwise it will choose a state at random. (A3) closely mimics the action choice rule as used in the ant system [1]: agents will choose actions at random, weighting actions with higher AQ values. (A2) can be considered a combination of both approaches.

The comparison of the three variants above were tested (using iteration-best delayed reinforcement, discussed below) and can be found in Table 1 of [3]. The values of most parameters were fixed throughout these runs, with the exception of the discount factor which was varied for each action choice function (the reason for this is not made clear). Statistical tests (Kruskal-Wallis ANOVA and Mann-Witney $t$-tests) were used to reaffirm the confidence of the following discussion.

We see that (A2) significantly ($p < 0.001$) performed the best, from which we can deduce that a combination of the methods in Q-learning and ant systems allow for greater performance. The best routes found by (A1) were close to that of (A2), and the standard deviations of (A1) and (A2) routes is low (albeit a little higher for (A1)). (A3) performed much worse than (A1) and (A2), the best solutions were far from that of (A1) and (A2) and the standard deviation was also much higher. From this, we conclude that pseudo-random-proportional action choice combines the methods from Q-learning and ant system effectively.

## 4.2 Delayed reinforcement

Gambardella and Dorigo [3] presents two methods for computing the delayed reinforcement.

(DR1) (Global-best) The delayed reinforcement is calculated by

$$\Delta\mathrm{AQ}(s,a) = \begin{cases} \dfrac{W}{L_{k^*}} & \text{if } k^* \text{ did } (s,a), \\ 0 & \text{otherwise} \end{cases}$$

where $W$ is an unspecified parameter and $k^*$ is the ant who made the globally best tour form the beginning of the trial, and $L_{k^*}$ is the length of its tour.

(DR2) (Iteration-best) The delayed reinforcement is given by the same formula as in (DR1), except $k^*$ is the ant who made the best tour in the current episode (or iteration).

Both (DR1) and (DR2) were shown to have similar performance, the results can be found in Table 2 of [3]. Though Gambardella and Dorigo [3] note that iteration-best has some beneficial properties. Namely, (DR2) was slightly faster in finding the same quality solution as those found by (DR1) and (DR2) was less sensitive to changes in the discount factor $\gamma$ then ($DR1$). We may hyperparameterise discount factor, so having less sensitivity to this parameter allows us to optimise it for other parameters easily using methods such as stochastic gradient descent.

### 4.3 Ant system

There are two major differences between ant system (AS) and Ant-Q. First, all agents in AS contribute towards delayed reinforcement, instead of just the ant with the best tour. Secondly, the AQ update formula is drastically simplified to $\mathrm{AQ}(s,a) \leftarrow (1-\alpha)\,\mathrm{AQ}(r,s)\,\Delta\mathrm{AQ}(r,s)$ and is applied to all edges, not just the one visited by the last agent.

It was found that Ant-Q significantly outperformed AS: both got the same optimum in TSP problems (but AS took much longer) and Ant-Q was able to find a much better solution than AS in the ATSP problem.

## 5 Two properties of Ant-Q

### 5.1 Divergence

Gambardella and Dorigo [3] conducted two experiments of the Ant-Q algorithm, first by examining the lengths of the tours over the iterations and another by examining the $\lambda$-branching factor (defined below) over the iterations.

Figure 2 plots the mean length of the best tour and mean length of all agents tours, as well as the mean length of all agents tours plus and minus the standard deviation of all agent tours (effectively giving us a confidence interval on the mean length of all agents tours).

It was found that ants in the Ant-Q algorithm did not converge to a common path. The monotonic decrease in mean length of the best tour and of all tours diminishes over-time, suggesting convergence. However, the standard deviation of the tour lengths converges to a non-zero value, suggesting that although the ants are reliably finding better paths in each iteration (converging to some best tour), perturbations from this optimal are still present. This suggests that ants always keep exploring, even after many iterations.

We introduce the $\lambda$-branching factor, which gives an indication of the dimension of the search space. We let $\delta(r) = \max_{s\in N\setminus\{r\}} \mathrm{AQ}(r,s) - \min_{s\in N\setminus\{r\}} \mathrm{AQ}(r,s)$ and, given $\lambda \in [0,1]$, define the $\lambda$-branching factor of a node $r$ to be the number of edges $(r,s)$ for which $\mathrm{AQ}(r,s) > \lambda\delta(r) + \min_{s\in N\setminus\{r\}} \mathrm{AQ}(r,s)$. Intuitively, it shows, depending on $\lambda$, how many actions are considered by the random variable $S$ (defined in a preceding section).

It is shown in Figure 3 in [3] that the mean (over all cities) $\lambda$-branching factor (for $\lambda \in \{0.04, 0.06, 0.08, 0.1\}$) monotonically decreases but converge to a non-zero value with each iteration. From this, we can conclude that agents drastically reduce their search space in the first *phase* of the algorithm, but then they withhold a small search space which they explore indefinitely. This property is beneficial as it prevents a *consensus* of an optimal solution forming (even if it is not optimal), and allows us to increase our ant count in order to magnify this effect.

### 5.2 Eventually detrimental heuristic

Gambardella and Dorigo [3] performed two experiments to determine the benefit (or detriment) of the heuristic function. The first experiment examined the tour length over the iteration but by not considering the heuristic in the action choice function when testing (we will refer to this as NO-HE), effectively setting the parameter $\beta = 1$ while testing and back to the original value for more iterations. The second experiment did the same but with the standard Ant-Q algorithm (henceforth HE).

These experiments were plotted and can be found in Figure 4 of [3]. In early iterations ($< 80$), the HE run had better performance then the NO-HE run. After iteration 300 however, NO-HE performed slightly better than HE. In the iteration range 80 to 300, NO-HE and HE performed almost identically (except from a few spurious points).

We can reason why this is simply: at the start of the run the AQ-values are effectively useless (they are all equal to some fixed value at the start). Over time, the AQ values become more useful to the ants, and eventually outperform the heuristic.

Although not stated in the paper, we propose a technique similar to simulated annealing's *entropy*: $\beta$ (the parameter that dictates how much to favour the heuristic) should be decrease every iteration, and this decrement should be chosen in a way that allows the fast early performance which we saw, but then stops the heuristic being detrimental henceforth.

Another suggestion: after a fixed number of iterations, turn off the heuristic in the learning phase. It would be interested to see if, after obtaining beneficial AQ values, the ants are able to learn more without the detriment of the heuristic.

# 6 Ant-Q in comparison to other heuristic algorithms

Ant-Q was compared against four different heuristic algorithms, which we briefly describe below. Note these comparisons were done on the TSP, not ATSP. We encourage the reader to ponder why some of the following methods do not apply for ATSP (for EN and SOM) or how they may be adapted (for SA and FI); economy prohibits further discussion on this here.

**Elastic net (EN)** This algorithm [2] starts with $m$ points with $m \gg n$ lying on a circular ring located at the centre of the cities (this ring can be arbitrarily constructed) embedded on $\mathbb{R}^2$. The rubber band is then gradually elongated until it passes sufficiently near each city define a tour, by two forces: one for minimising the length of the ring (corresponding to the tour length) and one for minimising the distance between cities and points on the ring.

**Simulated annealing (SA)** This algorithm [7], like Ant-Q, takes inspiration from real-world phenomena. It is an iterative process in which there is a running variable, which we will call *entropy*, which is decreased every step. We first pick an initial tour (this can be found using any method). Then, every step we will randomly generate a set of random transformations to the tour, and each perturbation is assigned a *cost difference* (how much better the transformation makes the tour). If the cost difference is positive, the transformation is applied. Otherwise, it stochastically decided whether or not to apply the transformation depending on the entropy (more entropy leads to picking unfavourable transformations more) and the cost difference. If after a step no perturbations are observed, we output the tour.

**Self-organising map (SOM)** This algorithm [4] is similar to the EN algorithm. We pick $m$ *neurons* and form them as a ring on $\mathbb{R}^2$ with the cities embedded. We iteratively select a random city and move the closest neuron closer to that city (the amount moved is dictated by the *learning rate*). We also move the neighbours (identified using a radius around the neuron) of the closest neuron. The radius at which neighbours are identified is gradually reduced with each iteration, as well as the learning rate.

**Farthest insertion (FI)** This algorithm [6] starts by picking the two cities farthest apart, say $s_1$ and $s_2$. We then pick $s_3$ as the city that is farthest away from the edge $(s_1, s_2)$. We then form the mini-tour $T = (s_1, s_2, s_3, s_1)$. Subsequent cities are added as follows: pick the point $s_k$ farthest from any point that has been incorporated into $T$ and note the edge $(s_i, s_j)$ in $T$ which is closest to $s_k$. We replace the edge $(s_i, s_j)$ in $T$ with $(s_i, s_k, s_j)$. This is repeated until all edges are incorporated.

FI improved with local optimisation (FI2 and FI3), SA improved with local optimisation (SA3), and an improved version of SOM (SOM+) was also compared against Ant-Q (algorithms introduced in [5]) and can be seen in Table 5 of [3]. It is also noted that the local optimisation heuristics were also applied to Ant-Q and found no benefit (that is, solutions from Ant-Q are locally optimal with respect to the 2-opt and 3-opt heuristics). Each algorithm (and improvements) were executed on a set of 5 50-city problems. Ant-Q was performed with iteration-best delayed reinforcement and pseudo-random-proportional action choice. Both Ant-Q and SA3 performed equally the best, finding the best runs in 4 of the 5 cities.

It is outlined that, although the experiment is promising, it does not pose a feasible solution to large TSP problems. Ant-Q's iteration complexity is $O(mn^2)$, and there exists better heuristic methods to tackle large problems. Instead, Ant-Q's strengths lie with ATSP problems. ATSP is a much harder problem than TSP, but Ant-Q's iteration complexity does not increase in this case. Ant-Q was compared against some exact algorithms for ry48p and 43X2 (both ATSP problems) and was found to get the optimal route as well as having mean route length close to this optimal. It even managed to compute the solution for 43X2 when the exact algorithm FT-92 was unable after 32 hours of running time.

# References


[1] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996), pp. 29–41.

[2] Richard Durbin and David Willshaw. "An analogue approach to the travelling salesman problem using an elastic net method". In: *Nature* 326.6114 (1987), pp. 689–691.

[3] Luca M Gambardella and Marco Dorigo. "Ant-Q: A reinforcement learning approach to the traveling salesman problem". In: *Machine learning proceedings 1995*. Elsevier, 1995, pp. 252–260.

[4] Teuvo Kohonen. "The self-organizing map". In: *Proceedings of the IEEE* 78.9 (1990), pp. 1464–1480.

[5] Shen Lin. "Computer solutions of the traveling salesman problem". In: *Bell System Technical Journal* 44.10 (1965), pp. 2245–2269.

[6] TAJ Nicholson. "A sequential method for discrete optimization problems and its application to the assignment, travelling salesman, and three machine scheduling problems". In: *IMA Journal of Applied Mathematics* 3.4 (1967), pp. 362–375.

[7] Peter JM Van Laarhoven and Emile HL Aarts. "Simulated annealing". In: *Simulated annealing: Theory and applications*. Springer, 1987, pp. 7–15.