1 Family of Ant-Q algorithms

The section provides an overview of the family of Ant-Q algorithms studied in [1]. This family is defined by the choice of parameters and structural choices, which are outlined below. We assume knowledge of Q-learning and Ant-Q reinforcement learning algorithms.

First we present the parameters of the family of Ant-Q algorithms.

δ The weight of importance of the AQ-value in the action choice rule.

β The weight of importance of the HE-value in the action choice rule.

q_0 The probability that we consider the AQ and HE-values in the action choice rule, instead of the random variable S.

α The learning rate in the range $(0, 1]$, from Q-learning: how sceptical ants are to new information.

γ The discount factor in the range $[0, 1]$, from Q-learning: how far-sighted the agent is.

AQ_0 The initial AQ-value for each state-action.

m The number of ants, a positive integer.

s_0^k The initial city for the ants.

We also have the following structural choices.

Action choice rule For a set of states S and a set of actions A, this is a function $C : S \to A$ such that if an ant is in state s, it will pick action $C(s)$.

Heuristic function The heuristic function as used in the action choice rule.

Delayed reinforcement The value $\Delta\mathrm{AQ}$ as used in the update function for the AQ-values.

The best values for the parameters were determined to be $\delta = 1$, $\beta = 2$, $q_0 = 0.9$, $\alpha = 0.1$, $\gamma = 0.45$, $W = 10$, and $\mathrm{AQ}_0 = (n\overline{L}_e)^{-1}$ where $\overline{L}_e$ denotes the average length of the edges in the induced weighted graph. If not otherwise stated, these are the value the parameters will take in the succeeding sections.

In terms of the structural choices, the best performing were found to be the pseudo-random-proportional action choice rule and the iteration-best delayed reinforcement, and we assume these unless otherwise stated. It is stated that the heuristic function is (at the time) a vital but an optimal is unknown. In the paper, it was chosen to be the inverse of the distances between cities, which is the obvious heuristic and the one we will use.

2 Sensitivity of the Ant-Q algorithms

This section serves to summarise the experimental results of [1]. Three investigations were performed, each using the stated parameters from the last section unless otherwise stated. First, the ant count was investigated on the Oliver30 test-bed, then the behaviour of Ant-Q with respect to α, γ, and q_0 was studied using Oliver30, and finally α, γ, and q_0 are investigated on the ry48p problem.

Each run of the algorithm in this section is split into two parts: the *learning phase* and the *testing phase*. In the learning phase, the standard Ant-Q algorithm is used, but in the testing phase the AQ-values are no longer updated (stopping learning).

2.1 Cooperation

The aim of this experiment is to understand the cooperation characteristic of Ant-Q. An initial experiment was run prior, setting $\delta = 0$ (so ants do not communicate at all). The performance was found to be poor.

The main experiment investigates the number of ants m from 1 to n (the number of cities),

1. first with only 200 iterations per trial; and
2. again with only 6000 tours per trial.

In both runs, each city had at most one ant in the initialisation phase. The reason for the two runs is that in 1 the number of times delayed reinforcement is given is fixed, while in 2 the number of times delayed reinforcement is given is dependent on m (more ants leads to less delayed reinforcement). The results of this can be found in Figure 2 to Figure 5 in [1].

The results of 1 show that, given the same amount of delayed reinforcement, the performance of the algorithm increases with m. It was shown for $m \geq 20$, the AQ-values that were learnt were effectively used by the ants to find short tours.

In the results of 2, the number of ants seemed to be independent of the number of optimal solutions found. Although, the average tour length was large for small values of m, but quickly diminished as m increased.

From this, we see that cooperation is a key characteristic of the algorithm

2.2 The effect of q_0, α, and γ

This investigation was split into two experiments:

1. one in which the q_0 value is varied; and
2. another in which α and γ are varied.

It is noted in 1 that $\gamma = 0.4$.

In the results of 1, it is noted that the best solution found in the testing phase is always equal to or better than that of the learning phase, regardless of the choice of q_0. This suggests that the ants effectively utilise learned AQ-values in order to find optimal solutions. We also note that there was a substantial drop in performance as q_0 is increased from 0.9 to 1, showing that randomness in the ant's behaviour is vital for the performance of this algorithm. This is intuitive though, exploration allows ants to discover routes that might initially give reduced reward, but have eventual pay-off (which is one of the strengths of Q-learning). The optimal value of q_0 was found to be 0.9, though for $q_0 \in [0.6, 0.9]$ the optimal solution was still found.

In the results of 2, the algorithm has much greater sensitivity to changes in γ than to changes in α; that is, how far-sighted the ants are has a greater effect on the performance than how sceptical the ants to new information. It is also noted that performance of the algorithm during testing is worse than the performance during learning only for *bad* parameters, thus AQ-values are useful to direct the ants towards a good solution only for *good* parameters; that is, ants do not get less effective throughout learning. In particular, for $\gamma \in [0.2, 0.6]$ and $\alpha = 0.1$.

2.3 The effect of q_0, α, and γ, ATSP

The same investigation as the last was conducted on ry48p, an ATSP problem. Experiments were ran for $\gamma \in \{0, 0.45, 0.9\}$ with $\alpha = 0.1$ and $\alpha \in \{0.1, 0.5, 0.9\}$ with $\gamma = 0.45$. It was reaffirmed that the best performance was found with $\gamma = 0.45$ and $\alpha = 0.1$.

When $\gamma = 0.9$, it was found that the ants slowly move away from an optimal solution, which is clear as the ants are too far-sighted in their decisions. When $\gamma = 0$, the ants converge to some solution, but it is not optimal, suggesting that they have found a local minimum.

When $\alpha = 0.9$, the ants are not particularly sceptical of new information and initially moves to a much greater tour length than the original; however, it seems to be tending towards a more optimal solution, but would require much more iterations to achieve this solution. When $\alpha = 0.50$, the ants quickly move towards a greater tour length than original, but do not improve on this. The comment to make here is that ants that are too sensitive to reward are not effective at finding solutions; they overestimate the benefit of an action and it takes a long time for AQ values to reach the level of response they require to make good long-term decisions.

Interestingly, the performance of α was shown to have a greater influence that when applied to Oliver30, suggesting that to solve ATSP problems, how sceptical an ant is to new information has a larger affect on the performance of the algorithms. Intuitively, one may agree as the search space is much higher.

3 Comparison of Ant-Q with other algorithms

The section summarises the results on the comparison of the Ant-Q algorithm with other *nature-inspired* algorithms in Dorigo and Gambardella [1].

3.1 Performance of Ant-Q

Ant-Q was executed on the three datasets Oliver30, 6x6 grid, and ry48p (the first two are TSP, the last is ATSP). The γ values were slightly modified to suit the dataset, but for all runs $\gamma \in [0.4, 0.46]$.

Ant-Q found the optimal solution in all three datasets, and the mean tour length at testing closely matched this optimal. In particular, no noticeable decrease was observed in the asymmetrical case.

3.2 Ant-Q vs AS

Ant-Q (with $\gamma = 0.4$) was compared with the original ant-system algorithm (AS) with the same datasets as the last subsection. Both algorithms achieved the optimal route in this experiment, but it is noted that the Ant-Q outperformed AS. The mean best route found by Ant-Q was consistently closer to the optimal than that found by A. The error percentage for Ant-Q was also consistently lower than for AS.

3.3 Ant-Q vs other *nature-inspired* algorithms

Ant-Q was compared against the genetic algorithm (GA), evolutionary programming (EP), and simulated annealing (SA) on the datasets Eil50 (50 cities), Eil75 (75 cities), and KroA100 (100 cities). We briefly outline these algorithms.

Genetic algorithm (GA) Inspired by evolution, genetic algorithms start with an initial population (a set of routes). The *fitness* of this population is calculated (tour lengths), and the best *genes* (edges) are identified, and are *crossed-over* (combined). There is also some mutation introduced to add variation (analogous to the exploration behaviour found in ants).

Evolutionary programming (EP) Evolutionary programming is very similar to the genetic algorithm, but we omit details of cross-over. Intuitively, instead of our population being part of the same *species*, we assume that they are not. Our population asexually reproduces: each parent has one child which has random mutations. Children with favourable characteristics are selected and further mutated and the unfavourable children are discarded.

Simulated annealing This has already been described in the previous part, but I repeat the same here. 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.

It was found that Ant-Q outperforms every other algorithm on every dataset, except for Eil50 where SA found a slightly better solution; however, SA took a much higher number of tours.

References

[1] Marco Dorigo and Luca Maria Gambardella. "A study of some properties of Ant-Q". In: *International Conference on Parallel Problem Solving from Nature*. Springer. 1996, pp. 656–665.