# Ant-Q: A Reinforcement Learning approach to the traveling salesman problem

**Luca M. Gambardella**
IDSIA
Corso Elvezia 36
6900 Lugano
Switzerland
luca@idsia.ch
http://www.idsia.ch

**Marco Dorigo**
IRIDIA, Université Libre de Bruxelles
Avenue Franklin Roosevelt 50
CP 194/6
1050 Bruxelles, Belgium, EU
mdorigo@ulb.ac.be
http://iridia.ulb.ac.be/dorigo/dorigo.html

## Abstract

In this paper we introduce Ant-Q, a family of algorithms which present many similarities with Q-learning (Watkins, 1989), and which we apply to the solution of symmetric and asymmetric instances of the traveling salesman problem (TSP). Ant-Q algorithms were inspired by work on the ant system (AS), a distributed algorithm for combinatorial optimization based on the metaphor of ant colonies which was recently proposed in (Dorigo, 1992; Dorigo, Maniezzo and Colorni, 1996). We show that AS is a particular instance of the Ant-Q family, and that there are instances of this family which perform better than AS. We experimentally investigate the functioning of Ant-Q and we show that the results obtained by Ant-Q on symmetric TSP's are competitive with those obtained by other heuristic approaches based on neural networks or local search. Finally, we apply Ant-Q to some difficult asymmetric TSP's obtaining very good results: Ant-Q was able to find solutions of a quality which usually can be found only by very specialized algorithms.

## 1 INTRODUCTION

Recently a new distributed algorithm for combinatorial optimization has been introduced by Dorigo, Maniezzo and Colorni (Dorigo, 1992; Dorigo, Maniezzo and Colorni 1996; Colorni, Maniezzo and Dorigo, 1991; 1992). The basic idea underlying this algorithm, called ant system (AS), was that of using a colony of cooperating ants to find shortest Hamiltonian tours in a weighted complete graph (the so called traveling salesman problem, TSP). We recently realized that AS can be interpreted as a particular kind of distributed reinforcement learning (RL) technique. In this paper we propose Ant-Q, a family of algorithms which strengthen the connection between RL, in particular Q-learning, and AS. The Ant-Q family of algorithms is presented in Section 2, and a few different members of the family are discussed and experimentally compared in Section 3. Experiments are run on symmetric and asymmetric traveling salesman problems. In Section 4 we experimentally study some properties of Ant-Q. In particular we show that, as computation goes on, agents do not converge toward a common path. Instead, they continue to search the solutions space. Also, we show that Ant-Q learns $AQ$-values which favor the agents discovery of short tours. In Section 5 we present a comparison between results obtained by Ant-Q and those obtained by other heuristic approaches on five sets of standard TSP problems. In this comparison Ant-Q resulted to be, for the majority of data sets, the best performing when compared to the elastic net, simulated annealing, the self organizing map, and farthest insertion. We also applied Ant-Q to some difficult ATSP problems finding very good results. For example, Ant-Q was able to find in 119 iterations[1] (238 seconds on a Pentium PC) the optimal solution for 43X2**,** a 43-city asymmetric problem (Balas, Ceria and Cornuéjols, 1993). The same problem could not be solved to optimality within 32 hours of computation on a workstation by the best published code available for the ATSP based on the Assignment Problem relaxation (Fischetti and Toth, 1992) of the ATSP, and was only very recently solved to optimality by (Fischetti and Toth, 1994) with an algorithm based on polyhedral cuts (branch-and-cut scheme). We conclude in Section 6 briefly discussing related and future work.

## 2 THE ANT-Q FAMILY OF ALGORITHMS

We introduce the Ant-Q algorithm by its application to the traveling salesman problem. Given a set of $n$ cities, and for each pair of cities a distance $d_{rs}$, the TSP is stated

[1] I.e., it visited 5,117 tours. The mean performance, over 15 trials, was 5625 (550 sec).

as the problem of finding a minimal length closed tour that visits each city once. An instance of the TSP is given by a graph (N,E), where N, $|N|=n$, is the set of cities and E is the set of edges between cities (a fully connected graph in the Euclidean TSP). In case $d_{rs} \neq d_{sr}$ we have the more general asymmetric traveling salesman problem (ATSP). Ant-Q algorithms apply indifferently to both problems.

Let *AQ(r,s)*, read Ant-Q-value, be a positive real value associated to the edge *(r,s)*. It is the Ant-Q counterpart of Q-learning Q-values, and is intended to indicate how useful it is to make move *s* (i.e., to go to city *s*) when in city *r*. *AQ(r,s)*'s are changed at run time. When Ant-Q is applied to a symmetric TSP, then *AQ(r,s)=AQ(s,r)*. Otherwise, when Ant-Q is applied to an asymmetric TSP, *AQ(r,s)*'s can be different from *AQ(s,r)*'s.

Let *HE(r,s)* be a heuristic value associated to edge *(r,s)* which allows an heuristic evaluation of which moves are better (in the TSP we chose the inverse of the distance).

Let *k* be an agent whose task is to make a tour: visit all the cities and return to the starting one. Associated to *k* there is the list $J_k(r)$ of cities still to be visited, where *r* is the current city. This list implements a kind of memory, and is used to constrain agents to make feasible tours, that is, tours which visit all cities once and only once.

An agent *k* situated in city *r* moves to city *s* using the following rule, called action choice rule (or state transition rule):

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \left\{ [AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta} \right\} & \text{if } q \le q_0 \\ S & \text{otherwise} \end{cases} \quad (1)$$

where $\delta$ and $\beta$ are parameters which weigh the relative importance of the learned *AQ*-values and the heuristic values, *q* is a value chosen randomly with uniform probability in [0,1], $q_0$ ($0 \le q_0 \le 1$) is a parameter such that the higher $q_0$ the smaller the probability to make a random choice, and *S* is a random variable selected according to a probability distribution given by a function of the *AQ(r,u)*'s and *HE(r,u)*'s, with $u \in J_k(r)$.
In equation (1), as it is the case in the following equation (3), we multiply the AQ-value *AQ(r,u)* by the corresponding heuristic value *HE(r,u)*. This choice was meant to favor those AQ-values belonging to shorter edges, and was mainly motivated by our previous work on the ant system. Other composition functions, different from multiplication, are possible and will be the subject of future work.

In Ant-Q *m* agents cooperate to learn *AQ*-values such that they can favor, in probability, the discovery of good TSP solutions. *AQ*-values are updated by the following rule:

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

The update term is composed of a reinforcement term and of the discounted evaluation of the next state. Parameters $\alpha$ and $\gamma$ are the learning step and the discount factor. In AS, and in all the Ant-Q algorithms presented here, the reinforcement $\Delta AQ$ is always zero except after each agent has completed its tour. How this delayed reinforcement $\Delta AQ(r,s)$ is computed will be discussed in Section 3.2.

The update rule of formula (2) is the same as in Q-learning, except for the fact that the set of available actions in state *s*, that is, the set $J_k(s)$, is a function of the previous history of agent *k*.

In Fig.1 we report the generic Ant-Q algorithm. The algorithm is called generic because there are two structural parameters[2] which are not instantiated. These are: (*i*) the action choice rule given by formula (1), and in particular the form of the probability distribution of variable *S*, and (*ii*) the way delayed reinforcement $\Delta AQ(r,s)$ used by formula (2) is computed and distributed. In Section 3 we will study and compare algorithms which result from different instantiations of the above structural parameters.

An iteration of the generic Ant-Q algorithm can be described in words as follows. First, at Step 1 there is an initialization phase in which an initial value is given to *AQ*-values, and each agent *k* is placed on a city $r_{k1}$ chosen according to some policy (discussed in Section 3). Also, the set $J_k(r_{k1})$ of the still to be visited cities is initialized. Then, at Step 2, a cycle, in which each of the *m* agents makes a move and the *AQ(r,s)*'s are updated using only the discounted next state evaluation, is repeated until each agent has finished its tour and is back in the starting city. At Step 3, the length $L_k$ of the tour done by agent *k* is computed, and is used to compute (see discussion in Section 3.2) the delayed reinforcements $\Delta AQ(r,s)$'s. Then *AQ(r,s)*'s are updated using formula (2). Finally, Step 4 checks whether a termination condition is met, and if it is not the case the algorithm returns to Step 2. Usually the termination condition is verified after a fixed number of cycles, or when no improvement is obtained for a fixed number of cycles. (In experiments in which the optimal value was known *a priori* the algorithm was stopped as soon as the optimum was found.)

## 3 AN EXPERIMENTAL COMPARISON OF ANT-Q ALGORITHMS

We start this section with an experimental investigation of the Ant-Q performance for a few different instantiations of two structural parameters: the action choice rule, and the way delayed reinforcement $\Delta AQ$ is computed. We conclude with a brief description of the ant system (AS), which first inspired Ant-Q, and with an experimental comparison of AS and Ant-Q.

[2] We call these parameters "structural" because their value changes the form of the algorithm. They differ therefore from other more standard parameters, like $\delta$ and $\beta$ in formula (1), which only need to be numerically optimized.

Algorithm performance was evaluated repeating each trial 15 times. We report means, variances and, when necessary, the significance of comparisons between means was computed by Mann-Whitney t-tests and Kruskal-Wallis ANOVA (Siegel and Castellan, 1956). In tables we report average and best performances. The average performance is computed by taking the best result obtained in each of the 15 trials and computing the mean. The best performance is given by the best result obtained in the 15 trials. In the experiments reported in the following, if not differently indicated, the value of parameters was set to: $\delta$=1, $\beta$=2, $AQ_0$=1/(average_length_of_edges$\cdot n$), $q_0$=0.9, $\alpha$=0.1, $\gamma$=0.3, $W$=10, $m$=$n$. In general, we found that the number $m$ of agents should belong to the range $[0.6 \cdot n, n]$: this range was experimentally found to be a good one (further experiments will be necessary to better understand the role of cooperation, see Dorigo and Gambardella, 1995, for some preliminary results). Regarding their initial positioning, we placed one agent in each city.

```
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
```

Figure 1: The Ant-Q algorithm

Table 1: A comparison of action choice rules. Type of delayed reinforcement: iteration-best. 50-city problems were stopped after 500 iterations. Oliver30 was stopped after 200 iterations and ry48p after 600 iterations. Averaged over 15 trials.

| | 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 |

These values were found to be very good for a set of benchmark problems: grid problems[3], Oliver30 (a 30-city symmetric problem, see for example Whitley, Starkweather and Fuquay, 1989), ry48p (a 48-city asymmetric problem, see TSPLIB, in Reinelt, 1994), and for a set of five 50-city symmetric problems in which cities coordinates were randomly generated (Durbin and Willshaw, 1987).

### 3.1 THE ACTION CHOICE RULE

We tested Ant-Q algorithms with the following action choice rules: *pseudo-random*, *pseudo-random-proportional*, and *random-proportional*. They are all obtained from formula (1) as follows.

- The pseudo-random rule is given by formula (1) in which $S$ is a random variable over the set $J_k(r)$, that is, over the set of cities not yet visited by the agent situated in city $r$, selected according to the uniform distribution. This action choice rule strongly resembles the pseudo-random action choice rule of Q-learning.
- The pseudo-random-proportional rule is given by formula (1) in which $S$ is a random variable over the set $N$, selected according to the distribution given by formula (3) below which gives the probability with which an agent in city $r$ chooses the city $s$ to move to.

$$p_k(r,s) = \begin{cases} \dfrac{[AQ(r,s)]^{\delta} \cdot [HE(r,s)]^{\beta}}{\sum_{u \in J_k(r)} [AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta}} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

- The random-proportional rule is the same as the pseudo-random-proportional in which $q_0$=0. That is, the choice of the next city is always done by using random selection where edges are chosen with a probability distribution given by formula (3). This action choice rule is the same as it was used in the ant system.

Table 1, in which we report the results obtained with the three action choice rules, clearly shows that the pseudo-random-proportional rule is by far the best choice for Ant-Q. The differences between means resulted to be significant (p-value < 0.001) for all problems except Oliver30 for both the Kruskal-Wallis ANOVA and the Mann-Whitney t-tests. These results were obtained using the *iteration-best* type of delayed reinforcement (see next Section 3.2). Very similar results were obtained with the *global-best* type of delayed reinforcement, which is introduce in the next section.

### 3.2 THE DELAYED REINFORCEMENT

We tested two types of delayed reinforcement, called *global-best* and *iteration-best.* These are discussed in the following. All the experiments were run using the pseudo-random-proportional action choice rule.

*Global-best.* The delayed reinforcement is computed by the following formula:

$$\Delta AQ(r,s) = \begin{cases} \dfrac{W}{L_{k_{gb}}} & \text{if } (r,s) \in \text{ tour done by agent } k_{gb} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $W$ is a parameter which in all experiments was set to $W$=10 (this value was found to be a good one with AS, and was not optimized for Ant-Q), $k_{gb}$ is the agent who made the globally best tour from the beginning of the trial, and $L_{k_{gb}}$ is its tour length. Formula (4) says that only those $AQ$-values which correspond to edges belonging to the globally best tour will receive reinforcement.

[3] A grid problem is a problem in which cities are evenly distributed on a squared grid. In this paper we present results obtained with a 6x6-city grid.

*Iteration-best.* The delayed reinforcement is computed by the formula:

$$\Delta AQ(r,s) = \begin{cases} \frac{W}{L_{k_{ib}}} & \text{if } (r,s) \in \text{ tour done by agent } k_{ib} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where $k_{ib}$ is the agent who made the best tour in the current iteration of the trial, and $L_{k_{ib}}$ is its tour length.

Results in Table 2 show that, on the average, the two methods give very similar results. Still, there are reasons to prefer the iteration-best type of delayed reinforcement. First, it was slightly faster in finding solutions of the same quality as those found by global-best. Second, and most important, it was less sensitive to changes of the value of parameter γ. We decided therefore to use the iteration-best type of delayed reinforcement in the rest of the paper (except where otherwise stated). In any case, our results on the best way of computing the delayed reinforcement aren't definitive yet, and this subject will require further research.

Table 2: A comparison among different ways to compute delayed reinforcement. 50-city problems were stopped after 500 iterations. Oliver30 was stopped after 200 iterations and ry48p after 600 iterations. Averaged over 15 trials.

| | 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 |

### 3.3 THE ANT SYSTEM

The ant system presents, with respect to Ant-Q, two major differences. First, the delayed reinforcement was computed according to the following formulas:

$$\Delta AQ(r,s) = \sum_{k=1}^{m} \Delta AQ_k(r,s) \quad (7)$$

$$\Delta AQ_k(r,s) = \begin{cases} \frac{W}{L_k} & \text{if } (r,s) \in \text{ tour done by agent k} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where $L_k$ is the length of the tour done by agent *k*, and *m* is the number of agents. In this case therefore the delayed reinforcement is such that edges belonging to shorter tours and chosen by more agents receive more reinforcement than those which belong to longer tours or which were chosen by a smaller number of agents. Therefore, in the ant system, as opposed to Ant-Q algorithms, all agents contribute to delayed reinforcement.

Second, formula (2) is simplified to $AQ(r,s) \leftarrow (1-\alpha) \cdot AQ(r,s) + \Delta AQ(r,s)$ and is applied to all edges, not only to those which were visited by at least one agent. This choice is due to the fact that AS was inspired by the observation of ant colonies and that the *AQ(r,s)* was intended to represent the amount of pheromone on edge *(r,s)*. The *AQ*-values updating formula was meant to simulate the change in the amount of pheromone due to both the addition of new pheromone deposited by agents (ants) on the visited edges, and to pheromone evaporation.

Table 3 indicates that, for the most difficult problems, Ant-Q outperforms the ant system. In particular, similar or better results were obtained in fewer iterations.

Table 3: A comparison among the best Ant-Q algorithm and ant system. The Ant-Q algorithm used pseudo-random-proportional action choice and iteration-best delayed reinforcement. AS experiments were stopped after 3000 iterations (or after the optimal solution was found). Ant-Q experiments were stopped after the optimal solution was found or after 50 iterations for the 6x6 grid, 200 iterations for Oliver30, and 600 iterations for ry48p. Results are averaged over 15 trials.

| | 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 |

## 4 TWO INTERESTING PROPERTIES OF ANT-Q

In this section we highlight two characteristics of Ant-Q. In Ant-Q *(i)* agents do not end up making the same tour, and *(ii)* learned *AQ*-values are such that they can be exploited by agents to find short tours. We present results obtained with the Ant-Q algorithm with pseudo-random-proportional action choice and delayed reinforcement com-

puted by the iteration-best method. The test problem is the ry48p asymmetric TSP. Qualitatively analogous results were obtained on other problems. In the experiments reported in this section we computed performance both during learning and in test sessions. The learning performance was computed while running the basic algorithm reported in Fig.1, while in the test sessions the updating of *AQ*-values was switched off and $q_0$ was set to 1 (that is, agents deterministically chose the best edge among those available). In both cases the performance was given by the length of the shortest tour done by the *m* agents in the considered iteration. Test sessions were run every 10 learning iterations. We ran two kinds of test sessions. In the first one, called NO-HE test session, each agent deterministically chose the edge *(r,s)* for which *AQ(r,s)* was the highest among those leading to a not yet visited city. In the second one, called HE test session, each agent deterministically chose the edge *(r,s)* for which the product $[AQ(r,u)]^{\delta}\cdot[HE(r,u)]^{\beta}$ was the highest among those leading to a not yet visited city.

### 4.1 ANT-Q AGENTS DO NOT CONVERGE TO A COMMON PATH

A characteristic of the ant system which we find also in Ant-Q is that agents do not converge to a common path. This was observed by running two experiments. In the first one, see Fig.2, we observed the mean[4] length of all agents tours and its standard deviation. Although the mean tour length diminishes as the algorithm runs, it is clear (see the standard deviation in Fig.2) that agents are not converging towards a common path. Nevertheless, the Ant-Q algorithm continues to improve the best found tour (see mean length of best tour in Fig.2).

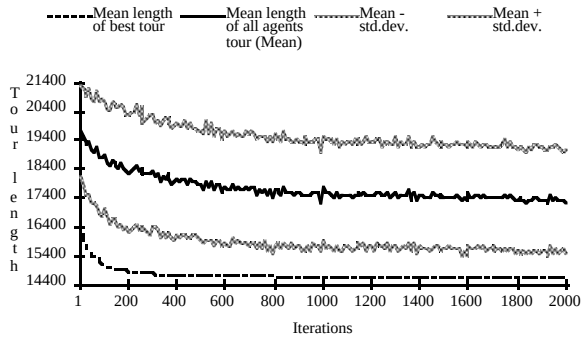


Figure 2: Mean length of best tour, mean length of all agents tour, and its std. dev. Problem: ry48p. Averaged over 15 trials.

In the second experiment we observed how the mean λ-branching factor changes as computation goes on. In this paper the λ-branching factor of a node *r* is defined as follows. Let $AQ_{max}(r,s)$ and $AQ_{min}(r,s)$ be the largest and the smallest respectively of the *AQ*-values on all edges exiting from node *r*, and let $\delta_r=AQ_{max}(r,s)-AQ_{min}(r,s)$. Given a parameter λ, $0\le\lambda\le1$, the λ-branching factor of node *r* is given by the number of edges exiting from *r* which have an associated *AQ*-value greater than $\lambda\cdot\delta_r+AQ_{min}(r,s)$.

The mean λ-branching factor gives an indication of the dimension of the search space. The experiment whose results are reported in Fig.3 showed that the λ-branching factor diminish monotonically as the computation goes on. Ant-Q maintains therefore a property which was typical of AS: the agents reduce drastically the search space (by a reduction of the λ-branching factor) in the first phases of the computation, but they do not converge on a common path. Agents continue to explore a subset of the search space as computation goes on.

This is a desirable property given that if agents explore different paths then there is a higher probability that one of them will find an improving solution than in the case they all make the same tour (which would make useless the use of *m* agents).

### 4.2 THE BEHAVIOR OF THE AGENTS IN THE TEST SESSION

Another interesting thing to do is to observe the behavior of the agents in the test session. Given that in the NO-HE test session at each step agents deterministically choose the edge with the highest *AQ*-value, good tours found in the NO-HE test session indicate that *AQ*-values are exploited by agents to find shortest tours. In Fig.4 we show the HE and NO-HE test session performances.

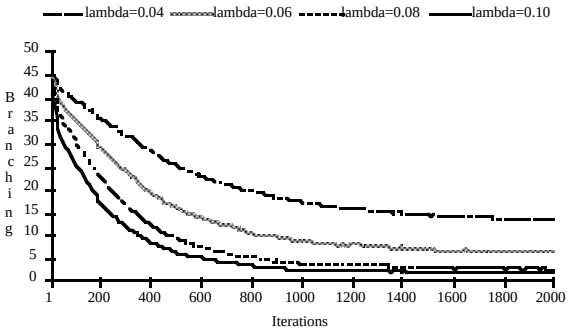


Figure 3: λ–branching factor. Problem: ry48p. Averaged over 15 trials.

[4] This mean is the mean over 15 trials of the mean length of the tours made by the *m* agents.

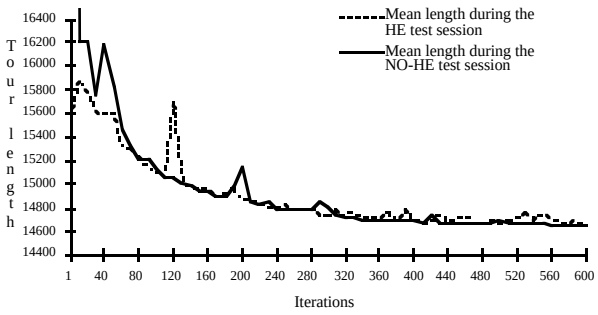


Figure 4: Best tour found during test session using only the AQ-values (NO-HE test session), and using both the AQ-values and the HE heuristic values (HE test session). The test session was run every ten learning iterations. Problem: ry48p. Averaged over 15 trials.

It is interesting to observe that, while in the early stages of the computation the use of the heuristic improves the performance, later on, after about 300 iterations, the NO-HE performance becomes slightly better than the HE performance. This is an indication of the fact that, as computation goes on, *AQ*-values become more and more useful to direct agents in finding good solutions. (At beginning the *AQ*-values have no special meaning; in fact, they all have the same value before the first iteration.) The heuristic function, which is useful in the early stages of the computation, seems to become useless, if not slightly detrimental, once good *AQ*-values have been learned.

## 5 COMPARISONS WITH OTHER HEURISTICS AND SOME RESULTS ON DIFFICULT PROBLEMS

We compared the average behavior of Ant-Q with the following well-known heuristic methods: Elastic Net (EN), Simulated Annealing (SA), Self Organizing Map (SOM), and Farthest Insertion (FI). We also compared the best result obtained by Ant-Q with those obtained using improved versions of the previous heuristics: SOM+, an improved version of SOM consisting of over 4,000 different runs of SOM processing the cities in various orders, and SA and FI plus local optimization by 2-opt and 3-opt (Lin, 1965)[5]. The comparison was run on the set of five 50-city problems of Durbin and Willshaw (1987). Table 4 reports the average results obtained by each heuristic (the best average result found by all heuristics is in bold font). It is interesting to note that Ant-Q was almost always the best performing algorithm. Also, when we applied the 2-opt and 3-opt heuristics to the result produced by Ant-Q these local optimization heuristics were not able to improve the result. That is, solutions produced by Ant-Q resulted to be locally optimal with respect to the 2-opt and 3-opt heuristics.

In Table 5 we compare the best results obtained by Ant-Q, SOM+, SA+3-opt, and farthest insertion plus local optimization by 2-opt and 3-opt. In this case, Ant-Q and SA+3-opt were the two best performing algorithms.

Although the above comparison of Ant-Q with some of the most well-known heuristics gave encouraging results, we believe that Ant-Q's real strength is in solving the asymmetric version of the TSP. In fact, Ant-Q iteration complexity (order of $m \cdot n^2$) makes quickly infeasible its application to big TSP problems, for which there exist good heuristic and exact methods. (Using exact methods optimal solutions can be found for instances of many hundreds of cities; the largest TSP solved optimally has 2392 cities (Padberg and Rinaldi, 1990).) On the other hand, ATSP problems are much more difficult than the TSP (using exact methods optimal solutions have been found for instances of no more than 200 cities), and Ant-Q maintains the same iteration complexity as when applied to the TSP. We run therefore a few experiments to compare Ant-Q with two exact methods[6] proposed in (Fischetti and Toth, 1992; 1994). In Table 6 we report the best result and the time needed to find it by the two exact algorithms and the mean and the best result obtained by Ant-Q on 15 trials of 600 iterations each. The two test problems are available in the TSPLIB[7] (Reinelt, 1994). For the 43X2 problem the FT-92 exact algorithm was not able to find the optimal solution within 32 hours of computation.

## 6 CONCLUSIONS

In this paper we have presented Ant-Q, a new family of algorithms inspired by both the Q-learning algorithm and by the observation of ant colonies behavior. Ant-Q is, to the authors knowledge, the first and only application of a Q-learning related technique to a combinatorial optimization problem like the traveling salesman problem (TSP). Results obtained with an application to the TSP, in particular to its asymmetric version, have shown that Ant-Q is very effective in finding very good, often optimal solutions to rather hard problem instances. Further work will be in the direction of a better understanding of Ant-Q dynamics. In particular, we suspect that the connection between Ant-Q and Q-learning can be made tighter and that theoretical results obtained for Q-learning can be extended or reformulated for Ant-Q. Also, we plan to extend the realm of Ant-Q applications, applying it to other combinatorial optimization problems.

[5] 2-opt and 3-opt are two well-known r-edge exchange procedures whose purpose is to improve a TSP feasible tour (Hamiltonian tour). In edge exchange procedures, r arcs are removed from the Hamiltonian tour, thus producing r disconnected paths. These paths are reconnected to produce the best possible tour. A tour is r-optimal if no r-exchange produces a tour with a lower length.

[6] These are considered to be the best exact methods for the ATSP.

[7] In the TSPLIB the problem 43X2 is called p43.atsp. They differ in that 43X2 is obtained from p43.atsp by doubling the length of arcs.

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** |

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** |

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 $\gamma$=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) |

**Acknowledgements**

We wish to thank Hugues Bersini for stimulating discussions on the subjects of this paper. This research has been partially supported by an Individual CEC Human Capital and Mobility Programme Fellowship to Marco Dorigo for the years 1994-1996.

**References**

Balas E., S. Ceria and G. Cornuéjols, 1993. A lift-and-project cutting plane algorithm for mixed 0-1 programs, *Mathematical Programming* 58, 295–324.

Colorni A., M. Dorigo and V. Maniezzo, 1991. Distributed Optimization by Ant Colonies. *Proceedings of ECAL91 - European Conference on Artificial Life*, Paris, France, F.Varela and P.Bourgine (Eds.), Elsevier Publishing, 134–142.

Colorni A., M. Dorigo and V. Maniezzo, 1992. An Investigation of some Properties of an Ant Algorithm. *Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92)*, Brussels, Belgium, R.Männer and B.Manderick (Eds.), Elsevier Publishing, 509–520.

Dorigo M., 1992. *Optimization, Learning and Natural Algorithms*. Ph.D.Thesis, Politecnico di Milano, Italy, EU. (In Italian.)

Dorigo M. and L.M. Gambardella, 1995. Ant-Q: A Reinforcement Learning Approach to Combinatorial Optimization. *Tech. Rep. IRIDIA/95-01*, Université Libre de Bruxelles, Belgium.

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.

Durbin R. and D. Willshaw, 1987. An analogue approach to the travelling salesman problem using an elastic net method. *Nature*, 326, 689-691.

Fischetti M. and P.Toth, 1992. An Additive Bounding Procedure for the Asymmetric Travelling Salesman Problem. *Mathematical Programming*, 53, 173–197.

Fischetti M. and P.Toth, 1994. A polyhedral approach for the exact solution of hard ATSP instances. *Technical Report OR-94*, DEIS, Università di Bologna, Italy, April 1994.

Lin S., 1965. Computer solutions of the traveling salesman problem. *Bell Syst. Journal*, 44, 2245–2269.

Padberg M. and G. Rinaldi, 1990. Facet identification for the symmetric traveling salesman problem. *Mathematical programming* 47, 219–257.

Potvin J–Y., 1993. The traveling salesman problem: A neural network Perpective. *ORSA Journal of Computing*, 5, 4, 328–347.

Reinelt G., 1994. *The traveling salesman: Computational solutions for TSP applications*. Springer-Verlag.

Siegel S. and N.J. Castellan, 1956. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill.

Watkins C.J.C.H., 1989. Learning with delayed rewards. *Ph. D. dissertation*, Psychology Department, University of Cambridge, England.

Whitley D., T. Starkweather and D. Fuquay, 1989. Scheduling problems and travelling salesman: the genetic edge recombination operator. *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 133–140.