A STUDY OF SOME PROPERTIES OF ANT-Q

TR/IRIDIA/1996-4

Université Libre de Bruxelles

Belgium

Marco Dorigo and Luca Maria Gambardella

IDSIA, Corso Elvezia 36, CH-6900 Lugano, Switzerland

dorigo@idsia.ch, luca@idsia.ch

Abstract. Ant-Q is an algorithm belonging to the class of ant colony based methods, that is, of combinatorial optimization methods in which a set of simple agents, called ants, cooperate to find good solutions to combinatorial optimization problems. The main focus of this article is on the experimental study of the sensitivity of the Ant-Q algorithm to its parameters and on the investigation of synergistic effects when using more than a single ant. We conclude comparing Ant-Q with its ancestor Ant System, and with other heuristic algorithms.

Published in the *Proceedings of PPSN IV–Fourth International Conference on Parallel Problem Solving From Nature*, H.–M. Voigt, W. Ebeling, I. Rechenberg and H.–S. Schwefel (Eds.), Springer-Verlag, Berlin, 656–665.

1 Introduction

In this paper we study some properties of Ant-Q, a novel distributed approach to combinatorial optimization based on reinforcement learning. Ant-Q (Gambardella and Dorigo, 1995) finds its ground in one of the authors previous work on the so-called Ant System (Colorni, Dorigo and Maniezzo, 1991; 1992; Dorigo, 1992; Dorigo, Maniezzo and Colorni, 1996), and in the Q-learning algorithm (Watkins, 1989). Ant System (AS), which will not be discussed here, is a distributed algorithm loosely based on the observation of ant colonies behavior, hence its name. It has been applied to various combinatorial optimization problems like the symmetric and asymmetric traveling salesman problems (TSP and ATSP respectively), and the quadratic assignment problem (Maniezzo, Colorni and Dorigo, 1994). Ant-Q is an extension of AS, which is reinterpreted in the light of reinforcement learning and in particular of Q-learning recent literature. This paper is devoted to a study of some of Ant-Q characteristics, and to a comparison of Ant-Q with AS and other heuristics, in which we show Ant-Q superiority. Ant-Q is different from Q-learning in that while typical applications of Q-learning see one single agent exploring the state space, Ant-Q uses a set of cooperating agents. These agents cooperate exchanging information in the form of AQ-values (the analogous of Q-values in Q-learning). Last, Ant-Q agents have some memory: this is necessary, as explained in Section 2, due to the different nature of our application problems (combinatorial optimization) with respect to typical Q-learning applications.

The article is organized as follows. In Section 2, we briefly describe the Ant-Q approach by means of its application to the traveling salesman problem. In Section 3 we discuss a few characteristics of the Ant-Q family of algorithms. In Section 4 we investigate Ant-Q by experimentation on TSP and ATSP problems. A brief conclusion is given in Section 5.

2 The Ant-Q Approach to Combinatorial Optimization

The easiest way to introduce the Ant-Q algorithm is by its application to the traveling salesman problem (TSP) or to the more general asymmetric traveling salesman problem (ATSP). They are defined as follows.

TSP

Let $V = \{v_1, \ldots, v_n\}$ be a set of cities, $A = \{(i,j) : i,j \in V\}$ be the edge set, and $d_{ij} = d_{ji}$ be a cost measure associated with edge $(i,j) \in A$.
The TSP is the problem of finding a minimal length closed tour that visits each city.
In the case cities $v_i \in V$ are given by their coordinates (x_i, y_i) and d_{ij} is the Euclidean distance between i and j, then we have an Euclidean TSP.

ATSP

If $d_{ij} \neq d_{ji}$ for at least some (i,j) then the TSP becomes an ATSP.

In the following of this section we will talk generically of ATSP problems, which include TSP as a special case.

Let k be an ant 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 is equivalent to say that ant k remembers already visited cities). An ant k situated in city r moves to city s using the following rule, called *pseudo-random-proportional* 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:

- *AQ(r,s)*, read Ant-Q-value, is a positive real value associated to arc *(r,s)* and is the Ant-Q algorithm counterpart of Q-learning Q-values. *AQ(r,s)*'s are changed at run time and are intended to indicate how useful it is to make move s (i.e., to go to city s) when in state r.
- *HE(r,s),* is a heuristic function which evaluates the goodness of move s when in city r. For example, in the ATSP *HE(r,s)* is the inverse of the distance between cities r and s.
- Parameters δ and β 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], and q_0 ($0 \le q_0 \le 1$) is a parameter: the smaller q_0 the higher the probability to make a random choice.
- S is a random variable selected according to the distribution given by formula (2) which gives the probability with which an ant in city r chooses the city s to move to.

$$p_k(r,s) = \begin{cases} \dfrac{[AQ(r,s)]^{\delta} \cdot [HE(r,s)]^{\beta}}{\sum_{z \in J_k(r)} [AQ(r,z)]^{\delta} \cdot [HE(r,z)]^{\beta}} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

As we said, the goal of Ant-Q is to learn AQ-values such that they can favor, in probability, the discovery of good ATSP solutions. AQ-values are learned by the following rule[1]:

$$AQ(r,s) \leftarrow (1-\alpha) \cdot AQ(r,s) + \alpha \cdot \left(\Delta AQ(r,s) + \gamma \cdot \max_{z \in J(s)} AQ(s,z) \right) \quad (3)$$

The update term is composed of a reinforcement term and of the discounted evaluation of the next state. In general, the reinforcement ΔAQ can be local (immediate) or global (delayed). In the current version of Ant-Q local reinforcement is always zero, while global reinforcement, which is given after all the ants have finished their tour, is computed by the following formula:

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

where L_{Best} is the length of the tour done by the best ant, that is the ant which did the shortest tour in the current iteration, and W is a parameter. The Ant-Q algorithm used in this paper is shown in Fig. 1 (the detailed algorithm can be found in (Gambardella and Dorigo, 1995)).

---

1 The form of this updating rule, which is very similar to the updating rule used by Q-learning, is the most relevant difference between Ant System and Ant-Q. In Ant System formula (3) reduces to $AQ(r,s) \leftarrow (1-\alpha) \cdot AQ(r,s) + \Delta AQ(r,s)$.

```
Ant-Q algorithm
/* Initialization phase */
   Set an initial value for AQ-values
/* Main algorithm */
Loop  /* This loop is an iteration of the algorithm */
 1.   /* Initialization of ants data structures */
      Choose a starting city for ants
 2.   /* In this step ants build tours and locally update AQ-values */
      Each ant applies the state transition rule (1) to choose the city to
      go to, updates the set J_k and applies formula (3) to locally  update
      AQ-values (in formula (3) ΔAQ(r,s)=0)
 3.   /* In this step ants globally update AQ-values */
      The edges belonging to the tour done by the best ant are updated using
      formula (3) where ΔAQ(r,s) is given by formula (4)
Until (End_condition = True)
```

Fig. 1. The Ant-Q algorithm

In words it can be described as follows. First, there is an initialization phase in which an initial value AQ_0 is given to AQ-values. Then comes the main algorithm loop which comprises three steps. In the first step each ant k is placed on a city $r_{k\text{-}initial}$ chosen according to some policy, and the set $J_k(r_{k\text{-}initial})$ of the still to be visited cities is initialized. Then, at Step 2, a cycle, in which each of the m ants makes a move and the $AQ(r,s)$'s are updated using only the discounted next state evaluation, is repeated until each ant has finished its tour and is back in the starting city. Last, in Step 3 the edges belonging to the shortest tour are updated using formulas (3) and (4). The loop is repeated until a termination condition is met. Usually the termination condition is verified after a fixed number of iterations, or when no improvement is obtained for a fixed number of iterations.

3 The Family of Ant-Q Algorithms

To apply the Ant-Q algorithm of Fig. 1 a set of somewhat arbitrary choices have to be done. These are: the value of parameters δ, β, and q_0 in formula (1), α and γ in formula (3), the constant W in formula (4), the initial AQ-values AQ_0, and the number m of ants and the choice of their starting city $r_{k\text{-}initial}$. Moreover, some structural characteristics like the state transition rule (1), the chosen heuristic function, and the kind of global reinforcement used, are worth investigating. In this section we discuss some of the most interesting alternative choices.

3.1 Parameters

In the experiments reported in this paper the value of parameters was determined as follows. Each of the parameters was optimized using Ant-Q applied to a set of benchmark problems: grid problems[2], Oliver30 (see for example Whitley, Starkweather and Fuquay, 1989), ry48p (see TSPLIB, in Reinelt, 1994). The experimentally determined best values are δ=1, β=2, q_0=0.9, α=0.1, γ=0.2÷0.6, W=10, AQ_0=1/(average_length_of_edges·n). Regarding the number m of ants and their initial positioning we decided to use m=n ants and to put one ant in each city. Results on the parameters α, γ, q_0, and m are discussed in Section 4.

[2] A grid problem is a problem in which cities are evenly distributed on a squared grid.

3.2 Structural Choices

The Ant-Q approach to combinatorial optimization provides a family of algorithms. Members of the family are characterized by: (*i*) the action choice rule, (*ii*) the heuristic used to bias the choice of the next city, and (*iii*) the kind of global reinforcement provided to the system.

(*i*) The action choice rule we use in our experiments, called *pseudo-random-proportional*, is a compromise between the *pseudo-random* action choice rule typically used in Q-learning (with the pseudo-random rule the chosen action is the best one with probability q_0, and a random one with probability 1-q_0), and the *random-proportional* action choice rule typically used in Ant System[3] (with the random-proportional rule the action is chosen randomly with a probability distribution given by the *AQ*-values). Experiments reported in (Gambardella and Dorigo, 1995) have shown that the pseudo-random-proportional action choice is by far the best choice for Ant-Q algorithms.

(*ii*) A result of our experiments is that the heuristic function *HE* is fundamental in making the algorithm find good solutions in a reasonable time. In fact, if we set β=0 the algorithm does not find any good solution at all. The use of *HE* requires to define its form, which in this paper has been chosen to be the inverse of the distance, and the way in which it is used to direct the search. We chose to multiply *HE* by the corresponding *AQ*-value, as shown in formulas (1) and (2). This choice was suggested by the desire of favoring AQ-values belonging to shorter edges; also, it maintained a continuity with our previous work on Ant System; other ways of composing *HE* and *AQ*-values may be worth investigating.

(*iii*) In the experiments presented in this paper only the best performing ant in the current iteration contributes to the global reinforcement. In previous work on Ant System all the ants contributed to the global reinforcement: the contribution of each ant was proportional to how short was its tour. This and other ways to provide reinforcement are discussed in (Gambardella and Dorigo, 1995).

4 Ant-Q: Experimental Study

In this section we experimentally study the functioning of the Ant-Q algorithm. Tests were run on the following problems: 6x6 grid and Oliver30, two TSP problems, and ry48p, an ATSP problem. Each trial was divided into two phases: a learning phase and a test session. The learning phase follows the algorithm reported in Fig. 1, while in the test session the updating of *AQ*-values is switched off and each ant deterministically chooses the arc with the highest *AQ*-value among those leading to a not yet visited city. The result of the trial is given by the performance of the best ant. Each experiment consisted of at least 30 trials, so to have some statistical information. In all experiments of this section the parameters, except when differently indicated, were set to the values reported in Section 3.1.

This section is divided into three parts. First, we study the synergistic effect generated by the use of more than one ant; here we used Oliver30 as a test-bed. Second, we study the behavior of Ant-Q with respect to α, γ, and q_0 again using Oliver30. Last, we repeat a similar study on the ry48p problem. It is important to note that while the 6x6 grid and Oliver30 are relatively simple problems that are easily solved to optimality by many general purpose (like genetic algorithms) and specialized algorithms, ry48p is much more

[3] A random-proportional rule is also often used in genetic algorithms to select individuals to reproduce.

difficult (Fischetti and Toth, 1992; 1994) and the fact that Ant-Q was able to find the optimal value is remarkable and encouraging.

4.1 Multiple Ants: Synergistic Effects

Cooperation between ants is one of the key characteristics of Ant-Q. In this section we show that ants cooperation is essential, and that this cooperation bears a synergistic effect. We show the importance of cooperation by setting $\delta=0$ in formulas (1) and (2): This corresponds to a complete inhibition of communication between ants, which are no longer guided by *AQ*-values. The result is a very low performance level.

To further investigate the existence of synergistic effects, we ran the following experiments: (i) we evaluate the performance of Ant-Q varying the number m of ants from 1 to n, given a fixed number of iterations (200 iterations per trial); (ii) we evaluate the performance of Ant-Q varying the number of ants from 1 to n, given a fixed number of ants tours (6,000 tours per trial). In both experiments in the initialization phase in each city was placed at most one ant. In the first experiment, by fixing the number of iterations we fix the number of times global reinforcement is given, which therefore is independent of the number m of ants; on the contrary, the computational effort changes with m given that each ant causes the application of formula (3) at each state transition. In the second experiment, we fix the total number of ants tours (6,000), and therefore the computational effort is independent of the number m of ants; on the contrary, in experiments with fewer ants (that is, with a small m) the trials last a greater number of iterations (the number of iterations is given by $6{,}000/m$) and therefore the number of times that global reinforcement is provided is greater. Results of the first experiment clearly show that, given the same total amount of global reinforcement, the performance of the algorithm increases with the number of ants. In fact, the number of times Ant-Q finds the optimal solution increases with m, as shown in Fig. 2. Moreover, Fig. 3 shows that for all experiments with $m \geq 20$ (with the only exception of $m=28$) the result of the test session is as good as the best result of the learning phase. This result suggests that for $m \geq 20$ the learnt *AQ*-values are effectively used by the ants to find short tours.

The second experiment, see Figures 4 and 5, shows that, given the same number of ant tours per trial, which implies a higher number of global reinforcements for trials with lower values of m, the number m of ants used is less important. In fact, the number of optimal values found by the ants is largely independent of m. Nevertheless, Fig. 5 shows that good test session performances where achieved only for high values of m. In particular, the test session solution was much worse than the best solution of the learning phase for most values of $m \leq 20$. These results were confirmed by further experiments run on the ry48p problem.

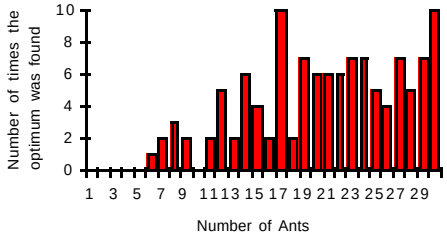

Fig. 2. Ant-Q performance for different values of m. Test problem: Oliver30. Averages on 30 trials, 200 iterations per trial, γ=0.4.

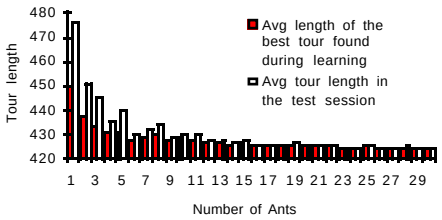

Fig. 3. Ant-Q performance for different values of m. Test problem: Oliver30. Averages on 30 trials, 200 iterations per trial, γ=0.4.

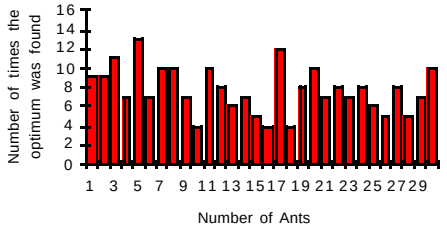

Fig. 4. Ant-Q performance for different values of m. Test problem: Oliver30. Averages on 30 trials, 6000/m iterations per trial, q_0 = 0.9 , α=0.1, γ=0.4.

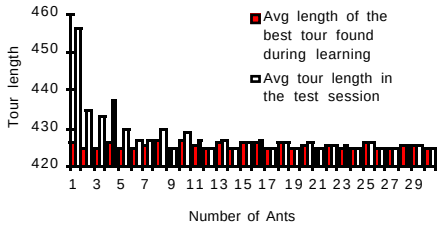

Fig. 5. Ant-Q performance for different values of m. Test problem: Oliver30. Averages on 30 trials, 6000/m iterations per trial, q_0 = 0.9 , α=0.1, γ=0.4.

4.2 The Effect of Parameters q_0, α, and γ on Ant-Q Performance

In this section we report the results of some experiments run to evaluate the best value for the following parameters: q_0, α, and γ. Each experiment consists of 30 trials, and each trial was stopped after 200 iterations of the algorithm. In the experiment to evaluate q_0 we set γ=0.4.

In Fig. 6 we show the average length of the best tours found by Ant-Q during the learning phase and in the test session. It is interesting to note that the result of the test session is always equal to or better than the best found solution during learning. This result suggests that the learnt *AQ*-values are effectively used by the ants to find short tours. Another interesting observation is that the algorithm performance suddenly decreases when q_0 changes from 0.9 to 1. This indicates that exploration is necessary to find good solutions. Ant-Q was able to find the best known solution (423.741) only for $0.6 \leq q_0 \leq 0.9$, and the best result was obtained with q_0=0.9: in this case Ant-Q found the optimal value 10 times out of 30 trials, with an average number of iterations of 96.6. In Fig. 7 we show the average length of the best tour found by Ant-Q and its standard deviation.

In Figures 8 and 9 we show the behavior of Ant-Q when changing α and γ. These bar graphs show that γ is much more important than α. In particular, good values of γ are found in the range 0.2÷0.6, while the value of α seems to be irrelevant. Further analysis of the results, and in particular of the number of times the algorithm was able to find the optimum, suggested that α=0.1 was the best value for this parameter. An interesting observation is that the performance of Ant-Q in the test session is worse than the performance measured in the learning phase only for bad parameter settings: once again,

this indicates that, for good values of the parameters, the learned AQ-values are useful to direct the search for good solutions.

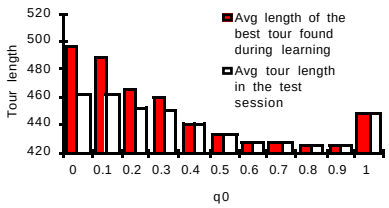

Fig. 6. Ant-Q performance for different values of q_0. Test problem: Oliver30. Avg. on 30 trials, 200 iterations per trial.

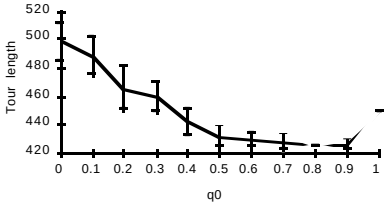

Fig. 7. Ant-Q performance for different values of q_0. Test problem: Oliver30. Avg. and Std. dev. on 30 trials, 200 iterations per trial.

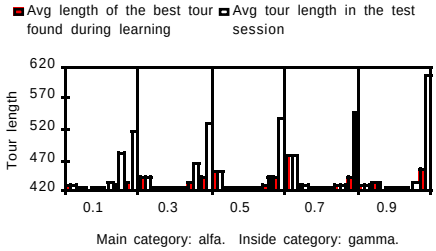

Fig. 8. Performance of Ant-Q for different values of α and γ. The main category is the parameter α. Within each value of α the parameter γ varies from 0 to 1 with steps of 0.2. Test problem: Oliver30. Averages on 30 trials, 200 iterations per trial.

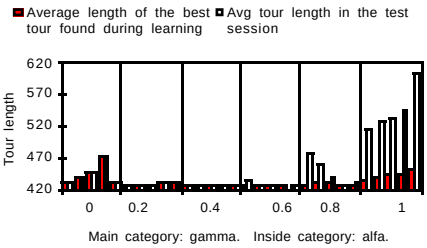

Fig. 9. Performance of Ant-Q for different values of α and γ. The main category is the parameter γ. Within each value of γ the parameter α varies from 0.1 to 0.9 with steps of 0.2. Test problem: Oliver30. Averages on 30 trials, 200 iterations per trial.

4.3 Ant-Q Applied to ry48p

In this last experiment we apply Ant-Q to a rather difficult asymmetric TSP, ry48p. Beside the fact that Ant-Q was able to find the optimal result for this problem, it is interesting to note that the (experimentally obtained) optimal value of most parameters remained the same as for the Oliver30 experiments (this property was found to hold also for a set of experiments run on grid problems whose results are not reported here because of space constraints). In Figures 10 and 11 we report the results of two experiments in which we show the influence that γ and α have on Ant-Q performance. Here the influence of α is higher than in the previous experiment on Oliver30.

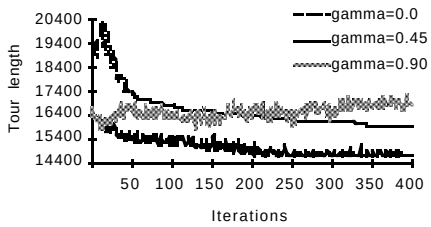

Fig. 10. Performance of Ant-Q for different values of γ. Test problem: ry48p. Averages on 5 trials, 400 iterations per trial.

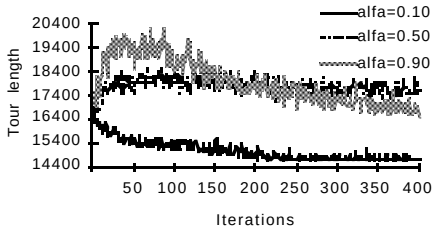

Fig. 11. Performance of Ant-Q for different values of α; γ was set to 0.45. Test problem: ry48p. Avg on 5 trials, 400 iterations per trial.

5 Ant-Q: A Summary of Results

In this section we present a summary of results obtained on some test problems. (Most of the test problems we used can be found in TSPLIB: http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html.) First, we report detailed results obtained with Ant-Q. For each of three test problems (Oliver30, 6x6 grid, and ry48p), in Table 1 we report the value of parameter γ used by Ant-Q (as always in this section, not tested parameters have the values reported in Section 3.1), the optimal solution, the best solution found by Ant-Q, and the number of tours (that is, the number of iterations of the algorithm multiplied by the number *m* of ants) it took to find it.

Second, in Table 2 the same three test problems are used to compare Ant System and Ant-Q. We report the optimum, and the average over five trials of the best-found solution, of the number of tours which were necessary to find it (only in case the algorithm always found the optimal solution), and of the error percentage (computed as 100·((Ant-Q_best_result - Optimum)/Optimum)). The main observations are that Ant-Q always outperformed AS, and that Ant-Q was in the average always very close to the optimal solution.

Third, we compare Ant-Q with other approaches using as test problems Eil50, Eil75, and KroA100. We compared ACS with the genetic algorithm (GA), evolutionary programming (EP), and simulated annealing (SA). In Table 3 we report the best integer tour length, the best real tour length (in parentheses) and the number of tours required to find the best integer tour length (in square brackets). Results using EP are from (Fogel, 1993) and those using GA are from (Bersini, Oury and Dorigo, 1995) for KroA100, and from (Whitley, Starkweather and Fuquay, 1989) for Eil50, and Eil75. Results using SA are from (Lin, Kao and Hsu, 1993). Eil50 and Eil75 are from (Eilon, Watson-Gandy and Christofides, 1969), and are included in TSPLIB with an additional city as Eil51.tsp and Eil76.tsp. KroA100 is also in TSPLIB. Results show that the ACS performance is always better than that of the other algorithms: the only case in which EP found a slightly better solution it took a much higher number of tours.

Table 1. Best results obtained by Ant-Q on three test problems: 6x6 grid, Oliver30, and ry48p.

Problem	γ	Optimum	Best-found	Mean	Std.dev.	#tours best-found
Oliver30 (a 30-city TSP)	0.46	423.74	423.74	424.44	0.46	180
6x6 grid (a 36-city TSP)	0.40	360	360	360	0	72
ry48p (a 48-city ATSP)	0.45	14,422	14,422	14,690	157	11,136

Table 2. A comparison of Ant-Q and AS on three test problems: 6x6 grid, Oliver30, and ry48p. Averages on 5 trials, γ=0.4.

Problem	Algorithm	Optimum	Avg. best-found	Avg. #tours	%Err
Oliver30 (a 30-city TSP)	Ant-Q	423.74	424.44	N/A	0.09
Grid6x6 (a 36-city TSP)	Ant-Q	360	360	677	0.00
ry48p (a 48-city ATSP)	Ant-Q	14,422	14,690	N/A	1.47
Oliver30 (a 30-city TSP)	AS	423.74	425.46	N/A	0.41
Grid6x6 (a 36-city TSP)	AS	360	360	2160	0.00
ry48p (a 48-city ATSP)	AS	14,422	14,889	N/A	3.30

Table 3. Comparison of Ant-Q with some others "nature inspired" algorithms. We report the best integer tour length, the best real tour length (in parentheses) and the number of tours required to find the best integer tour length (in square brackets). The best result for each problem is in boldface.

Problem name	Ant-Q	GA	EP	SA
Eil50 (50-city problem)	**426** (428.83) **[10,640]**	428 (N/A) [25,000]	**426** **(427.86)** [100,000]	443 (N/A) [68,512]
Eil75 (75-city problem)	**535** **(542.31)** **[8,970]**	545 (N/A) [80,000]	542 (549.18) [325,000]	580 (N/A) [173,250]
KroA100 (100-city problem)	**21282** **(21,285.44)** **[59,150]**	21761 (N/A) [103,000]	N/A (N/A) [N/A]	N/A (N/A) [N/A]

6 Conclusions

In this paper we studied some aspects of the functioning of Ant-Q, an algorithm belonging to the class of ant colony based search methods, and we presented some interesting computational results (e.g., Ant-Q was able to find the optimal solution of a difficult 48-city ATSP). Current research directions include the study of the meaning of *AQ*-values, and the identification of the set of problems that can be efficiently solved by Ant-Q.

Acknowledgments

This research has been funded by the Swiss National Science Fund, contract 21–45653.95 titled "Cooperation and learning for combinatorial optimization". Helpful comments were made by Hugues Bersini.

References

Bersini H., C. Oury and M. Dorigo, 1995. Hybridization of genetic algorithms. *Tech. Rep. No. IRIDIA/95-22*, IRIDIA, Université Libre de Bruxelles, Belgium.

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. (In Italian.)

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–Part B*, 26, 2, 29–41.

Eilon S., C.D.T. Watson-Gandy and N. Christofides, 1969. Distribution management: mathematical modeling and practical analysis. *Operational Research Quarterly*, 20, 37–53.

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. *Tech. Rep. OR-94*, DEIS, Università di Bologna, Italy, April 1994.

Fogel D., 1993. Applying evolutionary programming to selected traveling salesman problems. *Cybernetics and Systems: An International Journal*, 24, 27–36.

Gambardella L. and M. Dorigo, 1995. Ant-Q: A Reinforcement Learning approach to the traveling salesman problem. *Proceedings of ML-95, Twelfth International Conference on Machine Learning*, Tahoe City, CA, A. Prieditis and S. Russell (Eds.), Morgan Kaufmann, 252–260.

Lin F.–T., C.–Y. Kao and C.–C. Hsu, 1993. Applying the genetic approach to simulated annealing in solving some NP-hard problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 23, 1752–1767.

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 Traveling Salesman: the Genetic Edge Recombination Operator. *Proc. of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 133–140.