Comparing Hybrid Approaches of Ant Colony Algorithms and Reinforcement Learning

Christian Lavoie
April 30, 2003

Abstract

Ever since its introduction in the literature in the mid nineties, the Ant System framework in [4] and its multiple derivatives have set the new standard for flexible approximation of NP-Complete problems. Ever since its introduction in the mid-to-late seventies and its refinement in the nineties, reinforcement learning has set the standard for various control problems. The present report investigates the various approaches to combine both and presents both a theoretical viewpoint and experimental results on the algorithms explored.

1 Introduction

1.1 The approach

In this report, I will expose the results of my findings on the question I asked myself back in March: Is there any way to combine Reinforcement Learning and Ant Systems?, or more to the point, Is there any point in using Reinforcement Learning as an aid in Ant System?. Unsurprisingly, the answers were yes and yes. Surprisingly, very little work needed to be done...

My approach was a two-pronged approach: I decided that I would explore the literature and come up with my own algorithm, and I believed that implementing it would yield experimental results to present here. After the project is over, it appears that combined RL and AS algorithms are already existing and have been researched more extensively than what I expected when I took this project as my own. This report is an explanation of the algorithms I discovered in the literature, and the results my implementations gave.

1.2 Perspective in RL and AS

Reinforcement Learning and Ant Systems are both very popular approaches to solve problems in their respective fields of application. Both have been applied to NP-Complete number problems, and both use a numerical value to qualify the goodness of the found solution. Both reuse the numerical data over time to
explore and exploit the solution space to find new solutions and new paths in the space. Both have been wildly successful.

The fact that AS and RL have so many similarities, is, in retrospect, no surprise whatsoever. As we shall see in this report, RL and AS seem to naturally generalize to the same generic framework. In this report, I shall first explore both RL and AS in some detail, then critically assess the first attempts at combining the two, and then finally explore the AntQ algorithm – an extension to both RL and AS, which seems to bridge the gap between two believed different frameworks. Finally, I shall present some small experimental data I obtained with an implementation of AS and one of AntQ.

We will see that there are quite a few ways to combine RL and AS, and that most of these ways perform quite well in real life – often much better than either RL or AS performs on its own.

2 Reinforcement Learning

As seen in class, Reinforcement Learning is a framework of algorithms based on the idea of learning through the usage of rewards. An agent takes a decision and receives a reward, and updates the probability of taking that decision again using the reward’s value as an indication of the goodness of taking that decision.

In a high level fashion, the RL algorithm we consider in this report is:

1. Initialize $Q(s, a)$
2. Initialize $s$ to the starting state
3. For each episode:
   (a) For each step, unless $s$ is a terminal state:
      i. Pick an action $a$ at state $s$ using values from $Q$
      ii. Perform $a$, observe reward $r$ and resulting state $s'$
      iii. $Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
      iv. $s = s'$

Which is to say we use Sutton and Barto [15] Temporal Difference Off-Policy or Q-Learning algorithm.

Like the Ant System below, the basic RL algorithm is a very flexible algorithm that can be changed in various ways. One of the most important variations of the algorithm is to use an On-Policy algorithm\footnote{Called SARSA} that changes the

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

step to

$$Q(s, a) = Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$$
where \( a' \) is chosen using the \( Q \) values as well – instead of being the best that the algorithm could take, is use the value the algorithm will take.

As we have seen, even more dramatic changes can be performed on the algorithm. In [12], Santamaria, Sutton and Ram change the explore the usage of various representations for \( Q \) values “table” in RL algorithms. While their goal is to use RL in continuous action and state spaces, their approach has been used to practically infinite state spaces\(^2\).

Reinforcement learning’s advantage over other approaches to solving NP-Complete problems are:

- RL does not require extensive previous knowledge of the problem or of possible approaches to the problem beyond the possible states and possible actions to be taken.
- RL (as opposed to other popular forms of machine learning) does not require examples of “good” solutions – a serious advantage in solving NP-Complete problems

Reinforcement Learning is part of the more generic Machine Learning theory.

\(3\) Ant System

Ant Systems are hard to situate in the current body of knowledge. In [16], Taillard, Gambardella, Gendreau and Potvin place it in the same category as Genetic Algorithms, Taboo Search and Scatter Search.

In [11], Roli and Milano place it in the wide body of knowledge about Multi Agent Systems. Surrounding it are the GRASP framework, Iterated Local Search, Memetic Algorithms and Cooperative Search.

Ant System is a relatively new framework that has been used extensively to solve various NP-Complete problems, such as the Traveling Salesman Problem, the Quadratic Assignment Problem, the Edge-Weighted \( k \)-Cardinality Tree Problem, the Job-Shop Scheduling Problem, the Vehicle Routing Problem, the Shortest Common Supersequence Problem, the Graph-Colouring Problem and the Sequential Ordering Problem.

All those problems have been repeatedly approximated in the past, but never had a single framework for approximation been so widely applied and so hugely successful. The advantages of Ant Systems are varied and quite impressive:

- AS can deal with dynamically changing problem instances
- AS is a probabilistic algorithm that does not suffer from instance-specific catastrophic behavior
- AS can scale with problem size in a linear fashion

\(^2\)State spaces where the amount of state exceed what is practically learnable or even practically representable on today’s computers
• AS is a low-order polynomial algorithm

• AS is flexible enough to deal with various mutations and experimentations without losing its most interesting convergence properties

We will go through each of those properties in due time, but first, here is a high level description of the algorithm, in its Traveling Salesman incarnation:

1. Initialize $m$ ants

2. Initialize the pheromona $n \times n$ matrix

3. While not done, have each ant find a tour, and update the pheromona matrix to represent the goodness of the tour found:

4. When done, output the smallest tour found so far.

The framework is described in few details for a good reason: each paper describing and Ant System seems to offer a slight variation, and the description here is the part that’s common to them all. It also seems to be the only required part of an Ant System, all seem to converge more or less well as long as sane parameters and variations are taken.

In more details, step 3 of the algorithm described in [4] is:

\begin{verbatim}
AntSystem(TSP, Ants, $\tau$, $Q$, $\rho$)
1  shortTour = inf
2  for x in Ants
3     do citiesList = cities(TPS)
4        startCity = citiesList[randomInteger()%sizeof(citiesList)]
5        citiesList.remove(startCity)
6        tour = {startCity}
7        while citiesList $\neq \emptyset$
8           do Choose nextCity $\in$ citiesList with probability
9               $p_{ij}^k = \frac{[\tau_{ij}(t)]^\alpha \times [\eta_{ij}]^{\beta_k}}{\sum_{k \in \text{citiesList}} [\tau_{ik}(t)]^\alpha \times [\eta_{ik}]^{\beta_k}}$
10              citiesList.remove(nextCity)
11              tour.append(nextCity)
12              tourCost = cost(tour)
13      for x = 1...sizeof(tour)
14         do y = x + $q$%sizeof(tour)
15           $\tau[tour[x], tour[y]] = \tau[tour[x], tour[y]] + \frac{Q}{tourCost}$
16      Pheromona = Pheromona * $\rho$
\end{verbatim}

and that step is repeated until either some short-enough path has been found, or a maximal number of steps has been executed.

As we can see, the basic algorithm itself is pretty simple, and very few things need to be explained beyond what was said earlier. $\eta_{ij}$ is the heuristic value given to a city – the heuristic chosen is unspecified, but the inverse of

\footnote{An entry ($i, j$) in the matrix represents the edge from city $i$ to city $j$}

\footnote{That is, give more pheromona to edges in shorter tours than in longer tours}
the distance between two cities seems to be a popular heuristic choice. $\rho$ is the evaporation rate – the speed at which pheromone (and thus, past mistakes) are forgotten by the ants. $\tau$ is the matrix of pheromone and $Q$ is the amount of pheromone ants can leave behind.

Experimentally, the Ant System as described here performed very well in Dorigo’s tests, and compared advantageously to other approximation schemes.

3.1 Close variants

Obvious parameters can be changed in the algorithm:

- $\rho$ The evaporation rate of the pheromone trails
- $\eta_{ij}$ The heuristic value
- $p_{ij}^k$ The probability of selection of a given city
- $\text{sizeof(Ants)}$ The quantity of ants used

And quite a few others are available to the determined implementor. Researchers have explored with various random number distributions, with different heuristics, with new update rules and countless other things.

Moreover, many variants on the basic algorithm exist, some of which involve adding steps between two iterations. In particular, [18] relates the addition of Simulated Annealing search to improve the efficiency of Ant System. The idea is to get the best solution found so far, apply SA to it to find better close solutions and update the pheromone matrix with those too (or only with those).

3.2 Hybrid Ant System

As a final testament of the flexibility of the approach, I quickly describe the HAS-QAP algorithm.

The hybrid ant system applied to the quadratic assignment problem, as described in [5] describe a major variation where instead of rebuilding an assignment from scratch at each step, the ants keep their assignment between steps, and at each step pick at random a factory to reassign and use the pheromone matrix to find a better location for the factory to reassign.

This is, to my knowledge, the first real use of memory in ant systems beyond the pheromone trails. In particular, it is the first attempt at using local ant memory, as opposed to global and shared ant memory.

---

5 Dorigo suggests in [4] that one ant per city is a good rule of thumb for this particular parameter, that is, $\text{sizeof(Ants)}$ should be $O(\text{sizeof(TSP)})$

6 HAS-QAP
4 Combining Reinforcement Learning and Ant System

There are a few fundamental approaches to combining RL and AS:

1. Give each ant a more extensive local memory and thus tend towards multi-agent systems

2. Treat the global pheromona matrix as a Q values table

The first approach is the approach taken in most of today’s research on parallelization of RL algorithms and usage of multiple agents. In this report, we shall, however, investigate the second approach, typified by the AntQ algorithm.

4.1 AntQ

The AntQ algorithm is first and foremost an Ant System. As such, the high-level interpretation of the algorithm is very similar to the original Ant System:

1. Initialize $m$ ants

2. Initialize the pheromona $n \times n$ matrix

3. While not done, have each ant find a tour, and update the pheromona matrix to represent the goodness of the tour found: 

\[ AQ_{ij}(t) = (1 - \alpha)AQ_{ij}(t) + \alpha[\Delta AQ_{ij} + \gamma Max_{k \in citiesList} AQ_{ik}] \]

where

\[ \Delta AQ_{ij} = \frac{Q}{L_{Best}} \]

if $(r, s)$ is in the best tour, and 0 otherwise, that is, where $\Delta AQ_{ij}$ is the standard Ant System update.

---

7 An entry $(i, j)$ in the matrix represents the edge from city $i$ to city $j$

8 That is, give more pheromona to edges in shorter tours than in longer tours
4. When done, output the smallest tour found so far.

The cross between RL algorithms and AS is obvious: At the city choice point in the algorithm, the decision is either a standard AS one, or a standard RL one. Moreover, the update rule is an RL update rule that has been updated to use the AS “reward”. In the current implementation described in [6] and in [3], the choice between the two is taken randomly with a random \( q \in [0, 1] < q_0 \) for some predefined \( q_0 \).

4.1.1 Relation to Ant System

As mentioned above, the AntQ algorithm is an obvious extension to the Ant System in similar ways to other variations like HAS-QAP or MMAS-TSP. However, the source of this variation is a reinforcement learning algorithm, Watkin’s Q-Learning algorithm[19].

It is interesting to note that AntQ does not break any fundamental rule of Ant System, it is a mere variation.

4.1.2 Relation to Reinforcement Learning

While AntQ looks and behaves exactly like an Ant System algorithm, it is good to remember that its roots also come from the RL approach. Yet, **AntQ is not a Reinforcement Learning Algorithm**. AntQ breaks the fundamental assumption of the **Markov Property**, which is that all states are only dependant on the state before them. Mathematically:

\[
\Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, s_{t-1}, a_{t-1}, \ldots, r_{1}, s_0, a_0\} \\
= \Pr\{s_{t+2} = s', r_{t+1} = r | s_t, a_t\}
\]

the probability that a state and reward is obtained from an action is only dependant on the state preceeding it. In AntQ, notice that the list of possible actions is restricted on the actions taken before, the ant cannot visit a city twice. As such, by visting a city, an ant removes the action of visiting that city from the available actions at subsequent states. **AntQ does not obey the Markov Property**.

However, the Markov property is sometimes broken in RL implementation, and often being “Markov enough” is usually good enough. It definitely seems to be the case here.

5 Experimental results

Comparing the AS to the AntQ algorithm is simple, as AntQ is a subset of the AS algorithm family. AntQ is basically AS with a specific selection rule and a particular update rule.
In my experiments, I ran AS and various AntQ’s to randomly generated TSP instances. The values of the TSP distances between two cities were generated as follows:

\[ d_{ij} = 15 + 50 \cdot \text{randomReal}(0, 1) \]

where \( \text{randomReal}(0, 1) \) is a uniformly pseudo random number generated by a Merseme Twister random number generator.

The results are clear, and the AntQ family outperforms the simplistic AS by a factor of 5 – 15% on average for completely asymmetric TSPs.

Convergence does not appear to be particularly faster, just converges to better solutions overall. It is possible at this point that my implementation is faulty, and unfortunately penalizes the AS algorithm, as the results found in [6] don’t favour AntQ as much as my own do.

However, my results were pretty constant on the set of TSPs I applied my implementation to, and it is thus my belief at this point that either the TSPs I generated were particularly hard for AS to deal with, asymmetric with completely uniform-random distances, and that moreover, the number of ants I used (50) favoured AntQ. Finally, the evaporation rate was possibly too high, forcing the ants to forget too fast in the AS case, something that did not happen in the AntQ Algorithm, as I disabled the evaporation code.

The parameters used in my runs were the following:

**AS**
- Iterations = 500
- Ants quantity = 50
- TSPsize = 50
- evaporation rate = 0.4
- pheromona quantity = 80.

**AS**
- Iterations = 500
- Ants quantity = 50
- TSPsize = 50
- evaporation rate = 0.4
- pheromona quantity = 80, \( \alpha = 0.5, \gamma = 0.8 \).

The AntQ algorithm returned tours of length 1600, whereas AS returned values of length 1700 to 1800. The variation was much larger in the AS instance, leading me to believe as I said above that the AS couldn’t simply remember enough, and that the results presented here are more akin to raw luck than actual AS convergence.

6 Conclusion

6.1 Literature review

Many researches now have looked into Ant Systems and Reinforcement Learning. A list of particularly interesting papers follows. I have read through the vast majority of these papers at some point exploring the combination of Reinforcement Learning and Ant Systems.
6.1.1 Ant Algorithms, Ant Systems and Ant Colony Systems

In [4] Dorigo exposes for one of the first times the Ant System as a continuation of his work on his Ph. D. thesis.

In [5] Gambardella and Taillard explore the applications of the Ant System to the Quadratic Assignment Problem in more details and in particular introduce the HAS-QAP algorithm.

In [2] Dorigo, Di Caro and Gambardella expose the Ant Colony Algorithm and its applications to various NP-Complete problems. They mention a few variations of the basic Ant System, including various updating moment ones.


In [3] Dorigo explores the various properties of the AntQ algorithm, and compares with the basic AS algorithm.

6.1.2 Situating Ant System

In [16], Taillard, Gambardella, Gendreau and Potvin attempt to situate Ant System in the more generic framework of meta-heuristics for NP-Complete problems.

In [11] Roli and Milano expose their MAGMA framework and include Ant Systems as a specific instance of the framework. Their work on multi agent systems does not seem to have been picked up in the community.

6.1.3 Theoretical Underpinnings

In [1] Blum, Roli and Dorigo present one of the very few attempts at giving strong theoretical underpinnings to the various Ant algorithms, using the hypercube framework to explore various search strategies for ants.

6.1.4 Reinforcement Learning combined with...

In [14] Stone and Veloso expose Distributed Artificial Intelligence and its application to Machine Learning, and describes DAI in terms of its application to a robotic soccer team.


In [13] Schmidhuber compares Reinforcement Learning with Evolutionary Computation, and greatly favours EC over RL.

In [8] Leerink, Schultz and Jabri expose a Reinforcement Learning exploration strategy similar conceptually to the AntQ algorithm, but reverse. Instead of starting from Ant System and picking things from Reinforcement Learning, they started from RL and picked things from Ant System.

9
6.1.5 Multi-agent Reinforcement Learning

In [7] Ghavamzadeh and Mahadevan explore an algorithm that dynamically recombines MDPs in multi-agent Reinforcement Learning problems to speed up learning.

In [9] Makar and Mahadevan explore multi-agent Reinforcement Learning through the MAXQ algorithm, and the benefits of hierarchical learning in terms of speedups to the learning task.

In [17] Tan explores the various implications and costs of doing cooperative vs independent learning agents.

6.2 Future work

In this report, I exposed the AntQ algorithm, its relation to the Ant System, to Reinforcement Learning, and gave a survey of available literature on the subject. Future work on AntQ should involve bringing back properties of the Ant System, in particular the ability to deal with dynamic problem instance. Another area of research that appears forgotten is the seeding of the $Q$ values and pheromona matrices in either the AntQ, the AS or the standard RL algorithm. Finally, the complete lack of any sort of thorough theoretical analysis of Ant algorithms is crippling, while experimental results are breathtaking, no one seemed to have bothered with discovering the theoretical underpinnings of the current success.

The AntQ algorithm framework, and its subset, the AS family of algorithms, is at this point one of the best algorithms out there to deal with NP-Complete problems such as the Traveling Salesman Problem and its extended version the Quadratic Assignment Problem. The sheer quantity of problems to which it has been applied to makes it one of the most important algorithms to study in the short term.

References


