Ant Colony Optimization algorithm applied to the Traveling Salesman Problem

Simon Pelletier

Abstract

Ant colonies have shown to exhibit emergent intelligence out in nature. The sub-field of ACO algorithms have found that these natural mechanisms for path-finding can be applied to discrete problems ie. The Traveling Salesman Problem. This paper explores the foundations of this area of research and provides a basic implementation of a few of the systems examined.

Ants out in the real world navigate a complex and dynamic landscape searching for food. When they find a food source they return with it to the nest so that the colony can benefit from this individuals hard fought prize. Scientists have found that each individual ant follows a rather simple set of rules. An ant will wander out across the environment looking for food in a random manner with a preference for following other ants trails. Each ant has the ability to lay down an aromatic scent called pheromone, which Argentinian ants deposit at all times (Deneubourg 1990). The laying down of this scent is what leads to the intelligent path-finding behaviour scientists have witnessed of ant colonies.

Imagine a starting situation in which a collection of ants forage out from the nest at the same time. Lets assume there is only one food source but multiple paths of varying lengths to the food. Say two ants reach the food source at one moment in time; ant A reached it via a shorter path than ant B. They both grab food and head back on their respective pheromone trails. Ant A has a shorter path and will reach the nest before ant B. Now ant A's trail to the food source has a heavier amount of pheromone on it than ant B's. Since ants probabilistically choose paths to follow based on the amount of pheromone deposited, each ant leaving the nest will favour ant A's path to the food source over ant B's. Over a period of time there will be a build up of pheromone on ant A's shorter path. Ants will be driven, on average, along ant A's pheromone heavy trail and this shorter trail will be the one most likely traveled. This feedback loop will cause most ants to use ant A's trail. Once the food source has been depleted the ants will no longer return back upon the same trail and instead search outward, in a random fashion, from the end of ant A's trail. Due to the lack of return travel, evaporation of ant A's pheromone dense trail will eventually lead to ants no longer preferentially favouring that path and instead they will explore new paths to other sources of food (ibid.).

It is in this manner that rather simple behaviour on the part of each individual ant leads to an emergent intelligence (efficient path-finding) of the group as a whole. And it is here that computer scientists have come along and taken these evolved mechanisms of nature and tried to apply them to the more rigidly defined problems of our computationally driven world. The Ant Colony Optimization Algorithm (ACO) was first introduced in 1996 by Marco Dorigo and associates, where they applied an ant-system to the Traveling Salesman Problem and which is exactly what we will do here (M. e. a. Dorigo 1996).

The Traveling Salesman Problem (TSP) is a combinatorial optimization problem and is explained as follows: Take a set of N cities with paths from each and every city to each and every other city. Each of these paths has a distance associated with it. The goal as the salesman is to try and find a route through all these cities where each city is visited once and only once and where

the cumulative distance traveled is the minimum of all the possible solutions. We call this minimal route through the collection of cities a hamiltonian cycle (Blum 2005).We shall define the TSP here as a graph $G = (N, E)$ where N is the collection of nodes pertaining to the cities and E is a set of edges $e_{i,j}$ with each edge e being a distance from $city_i$ to $city_j$ where $i, j \in N$. We will be using a specific TSP called Oliver30 that is a standard comparison metric used in papers exploring combinatorial optimization algorithms (M. e. a. Dorigo 1996).

The type of algorithm we shall develop in this paper is going to follow in the footsteps of the *ant-system* presented in M. e. a. Dorigo (ibid.) paper as well as in other similar implementations (M. Dorigo and Gambardella 1997). The basic approach is going to be a system of individual ants that search the TSP, find a hamiltonian cycle (which we will call a *solution* or a *tour*) and then use these solutions to inform how pheromone is applied to the graph. The system will then iterate on this process, we call it a *cycle*, by resetting the ants and sending them out on the updated problem with its new pheromone levels applied to the edges. Through this iterative process the ACO will, if it's going to be a useful algorithm, find an approximately optimal solution to the TSP in a reasonable amount of computational time – ideally we don't have exponential growth in computation time as the number of cities/nodes increase.

To begin with we place one ant on each of the cities/nodes in the TSP. From here the system will choose the next node for each ant to move to. The choice of the next node is detailed in the probability function p below. There are two components in the function: One is the pheromone level which we call the *trail intensity* ($\tau_{i,j}$). This is the pheromone amount on edge $e_{i,j}$ from city i to city j (M. e. a. Dorigo 1996). The other is an attribute called *visibility* ($\eta_{i,j}$). (ibid.). This is where our artificial ant branches away from reality and we begin to craft an abstract ant-system somewhat divorced from real ants. Our artificial ants have some vision of the problem (they know the distances to different nodes) that real ants do not. The visibility heuristic is a function of the distance from the current node to the next-choice node and it's inversely proportional to the distance: $\eta_{i,j} = 1/e_{i,j}$ where e is distance from city i to city j. We then set each edges probability to:

$$p_{i,j}^k = \frac{[\tau_{i,j}]^\alpha * [\eta_{i,j}]^\beta}{\sum_{l \in N_i^k} [\tau_{i,l}]^\alpha * [\eta_{i,l}]^\beta} \tag{1}$$

where N_i^k is the set of potential cities to move to next for ant k currently at city i (M. Dorigo and Stützle 2004). This is the probability of choosing the edge from city i to city j given a set of choices and where α and β are parameters that determine the relative importance of each attribute in p.

In order to avoid visiting a node more than once we keep a list of cities that have already been visited, called a tabu list (M. Dorigo and Gambardella 1997). For each iteration the probabilities are mapped to each potential edge that each ant can traverse. The system then probabilistically selects an edge and places the new city chosen into the ants tabu list. This is repeated until the tabu list includes all the nodes in the TSP and every ant in the system has returned back to its starting node. The completion of a *cycle*.

In the ant-system approach we will be laying down the pheromone as each edge is traversed by the ant: $\tau_{i,j} \leftarrow \tau_{i,j} + \Delta\tau_{i,j}$. We call this *local update pheromone* and in our implementation there are two ways in which the amount of pheromone deposited is calculated. AntQuantity lays down pheromone inversely proportional to the distance of the particular edge the ant traverses: $\Delta\tau_{i,j} = Q/e_{i,j}$ where Q is a parameter set to 1 in our implementation. AntDensity deposits pheromone in a fixed amount and so it's simply a parameter set at initialization of the algorithm (in our case $\Delta\tau_{i,j} = Q = 0.05$) (M. Dorigo and Stützle 2004). (we probably want to explain how exactly these two local mechs manage to narrow our choices and find an approx. Solution) The other important part of the system is the mechanism by which pheromone is removed from edges. M. e. a. Dorigo (1996) calls this *pheromone evaporation* and it is done after every cycle and is defined by the following equation: $\Delta\tau_{i,j} = (1 - \rho)\tau_{i,j}$ where ρ is the evaporation parameter. This evaporation of pheromone across the graph allows for edges that were explored but didn't result in promising solutions to be ignored where as areas of interest, shorter edges selected by the visibility heuristic, will continue to have pheromone build up, and thus ants will continue to explore these sub regions of the space.

1. ACO ant-system approach with *local update pheromone*

```
for run = 1, 2, ..., numOfRuns do
    init. TSP – pheromone init.
    for cycle = 1, 2, ..., numOfCycles do
        init. ants
        for i = 1, 2, ..., TSP.size do
            for each ant_i^k = 1, 2, ..., antList.size do
                choose edge e_{i,j} based on ρ probability
                update pheromone τ_{i,j} ←τ_{i,j} + Δτ_{i,j}
                add city choice to ant tabu list
            end for
        end for
        tour complete
    end for
end for
```

The ACO system at a high-level view operates on the TSP in an iterative

fashion. The TSP is read into memory, ants initialized and put onto nodes. The ACO is run and tours are found for each ant, then depending on which version of the algorithm we are using, the ACO lays down pheromone either during the tour formation (pseudo code 1.) or at the end of the cycle (pseudo code 2.). These new pheromone values will determine which nodes the next set of ants will probabilistically choose going forward. The next cycle is started and new tours are found and the process repeats until some conditions have been met – the ant solutions have begun to converge, the specified number of cycles has been completed (called a *run*, the choice in this implementation) or some other metric.

The two approaches discussed so far, AntQuantity and AntDensity deposit pheromone as each edge is traversed. There is another subset of algorithms that use a global method for updating pheromone and in M. e. a. Dorigo (1996) they refer to this as the *ant-cycle* system. Pseudo code for this approach is shown below. This approach waits until the cycle has been completed and then lays down pheromone for each ant based on the path it has chosen. The amount of pheromone deposited is proportional to the total length of its tour ($Q/e_{i,j}$) and so over time the paths with shorter tour lengths will end up with more pheromone on them. This ant-cycle system has been shown to find superior minimal tour lengths when compared to AntQuantity or AntDensity and tests on our implementation support this finding (ibid.).

2. ACO ant-cycle approach with *global update pheromone*

```
for run = 1, 2, ..., numOfRuns do
    init. TSP – pheromone init.
    for cycle = 1, 2, ..., numOfCycles do
        init. ants
        for i = 1, 2, ..., TSP.size do
            for each ant_i^k = 1, 2, ..., antList.size do
                choose edge e_{i,j} based on ρ probability
                add city choice to ant tabu list
            end for
        end for
        tour complete
        update pheromone for each ant τ_{i,j} ←τ_{i,j} + Δτ_{i,j}
    end for
end for
```

In order to improve the performance of the ACO there are an additional set of methods that one can implement. The higher level decisions, actions not considered by individual ants, are referred to as *Daemon Actions* and include techniques such as global pheromone updating (as shown above), choosing starting positions of ants and resetting of parameters due to stagnation (Blum 2005).

Here we include two further improvements taking advantage of these Daemon selection mechanisms of the ant-cycle system: *elitist* and *Max-Min*.

The *elitist* system extends the ant-cycle approach but instead of depositing pheromone on the paths of every ant in the cycle, it instead grabs the best-so-far ant (the ant with the shortest tour length from any of the previous cycles) and deposits pheromone on the edges of this best-so-far path. The intention here is that by applying pheromone to only the best solution found, the edges around this cycle will attract more of the ants activity and they will find improvements on the current minimal path (M. e. a. Dorigo 1996). On average the elitist system improves upon the ant-cycle and this is reflected in our tests (ibid.). However, the initial tour found has a great influence on the outcome of further solutions. Since the system lays down pheromone on the best tour, each cycle will typically reinforce the paths of the last cycle and it can be difficult for the ants to explore very far from this local minimum. The elitist approach can struggle to iteratively find better solutions and potentially stagnate (many of the ants traversing the same path repeatedly) very quickly.

The *Max-Min* system improves upon the elitist version, keeping the benefits of the *global update pheromone* while attempting to mitigate the stagnation behaviour that can occur with the elitist strategy (Stutzle and Hoos 1997). It does this in four ways: It uses a *global update pheromone* choice of the cycle-best path. The cycle-best path being the shortest tour found by the set of ants in that specific cycle. The cycle-best allows for more variety where pheromone is laid down due to increased variability of paths chosen — sometimes a cycles shortest tour will be longer than the previous shortest tour. This tends to favour exploration more than the elitists best-so-far choice.

There are two additional parameters that are introduced in this algorithm, *minPheromone* and *maxPheromone*. These are limits on the possible range of pheromone laid down on edges. The *maxPheromone* is the important one and its function is to limit how entrenched a certain solution can get during a run of cycles. Ideally this will limit the amount of early stagnation that can happen in the system by still giving each ant some reasonable chance of exploring an edge without much pheromone.

In this approach the pheromone is initialized in the opposite manner from the other strategies. Here the TSP starts out with all its edges having *max-Pheromone*. The best cycles are continually reinforced each round and the edges of less desired solutions eventually have most of their pheromone evaporated away, leaving only minimal solutions (ibid.).

Finally, the Max-Min has a mechanism for reseting the pheromone values if there is too much convergence of the ants path choices. For example say that more than 50% of the ants in a cycle return the same path length. The

system will recognize this as a lack of exploration and will reset the pheromone levels allowing for ants to begin anew, potentially finding shorter paths. In their paper Stutzle and Hoos (1997) call this *smoothing* and do it in a way that saves some of the pheromone variability that currently exists. This leaves some of the information gained from the system in place while still allowing for more ant activity in previously stagnant areas of the problem space. In our system we do not do this smoothing and instead do a wholesale reset of the pheromone levels with which we achieve reasonably good results.

In our implementation we ran all five ACO's discussed so far: AntQuantity, AntDensity, AntCycle, AntElitist and AntMaxMin are the class names. We did ten runs of two hundred cycles for each of the systems and calculated the averageBestTour from each run as well as the single bestTour found. The numbers for our runs are in the included output.txt file. While each run is random, due to the nature of the ants behaviour, the averages tend to be about the same and one should find similar numbers to what we have found. The elitist and Max-Min had the best averageTours with 424.835 and 424.199 respectively. The Max-Min did on average beat out the elitist system. The ant-cycle came in at 428.098, right in the middle, as expected, and AntQuantity and AntDensity had the longest tours of the systems with values of 432.764 and 430.440. Its important to note that all of the systems were able to find short tours, however, the systems with more Daemon actions and global oversight (elitist, Max-Min, ant-cycle) were much more consistent and were the only ones to approach the optimal value (423.741) for the Oliver30 problem (Dower 2013).

The reason that the AntQuantity and AntDensity systems don't perform as well as the other ones is due to their inability to properly select tours that are minimal. This is fairly obvious with the AntDensity system as it lays a fixed amount of pheromone on each edge. The AntQuantity does deposit its pheromone proportional to the each edge length, however, this is likely too liberal and too expansive, limiting its ability to narrow the search. The difference between the AntQuantity system and the AntCycle system is that the ants aren't influencing each other as the tour is built. The AntCycle on average returns shorter tours and this seems to suggest that there is a downside to the feedback affect of pheromone laying within a cycle from the AntQuantity approach. It might suggest that randomness is important for search as the AntCycle approach does end up with more randomly allocated pheromone deposit, at least in early stages of a run. The elitist takes the AntCycle approach and narrows the search dramatically by preferring the best solution found across all the runs. This does seem to work as its averages are better than the AntCycle, but there is some amount of stagnation due to ants path lengths converging part way through the runs. The Max-Min is a direct fix to the elitists limit

on exploration. The Max-Min improves on the elitist approach by favouring a little bit more randomness (cycle-best tour) while also not committing fully to a potential local minimum (reset if there is convergence). This ability to unlock convergence while also being able to narrow into a good solution seems to be a superior approach.

The Oliver30 TSP, used as a measuring stick in the literature, is not really a challenging optimization problem for the implementations of the ACO's that have been explored here. There are a host of other types of discrete optimization problems that are a significant challenge in comparison. The Asymmetric Traveling Salesman Problem (ATSP) is a similar problem to the TSP with the only difference being that any particular edge has distances associated with each direction, $e_{i,j} \neq e_{j,i}$. While TSP with thousands of nodes can be solved optimally, ATSP's with only a few dozen nodes can't (M. e. a. Dorigo 1996). Outside of the TSP set there are a variety of other combinatorial optimization problems that the ACO algorithm can be applied to. A short list includes: scheduling problems, routing problems, cell placement in circuit design. Protein folding is one the most computational intense problems in molecular biology and there have been ACO algorithms that have been reported to perform better than the existing efficient methods typically used (Blum 2005).

These problems are a clear avenue for building upon the basic ACO as the requirements force new approaches to be applied. Not surprisingly, there are a whole host of additional ant-systems that have not been examined here. One approach that is an extension of the elitist system is called rank-based ACO. This strategy sorts the tours into a rank ordered list based on each ants tour length and only a select number of the shortest tours are allowed to lay pheromone down on the edges (He 2017). In the Best-Worst ACO only the best and worst tours have pheromone deposited on edges and the pheromone re-initialization process is used if stuck in a local optima (ibid.). While some of these problems require a major redesign of the ACO to fit the specific situation, the core design features remain; it uses individual ants that lay down information about the problem onto the environment, this reinforcement of certain sub-spaces lead to emergent behaviour that can potentially find approximate optimal solutions.

References

Blum, Christian (2005). "Ant colony optimization: Introduction and recent trends." In: *Physics of Life Reviews.* 2 (4), pp. 353–373.

Deneubourg, J.L. et al. (1990). "The Self-Organizing Exploratory Pattern of the Argentine Ant." In: *Journal of insect behavior* 2, pp. 159–168.

Dorigo, M et al. (Feb. 1996). "Ant system: optimization by a colony of cooperating agents." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26 (1), pp. 29–41.

Dorigo, M. and L. M. Gambardella (1997). "Ant colonies for the travelling salesman problem." In: *BioSystems.* 43 (2), pp. 73–81.

Dorigo, M. and T. Stützle (2004). *Ant Colony Optimization.* 1st. Cambridge, Mass: A Bradford Book.

Dower, Steve (2013). "Oliver30 TSP". In: URL: `https://stevedower.id.au/blog/research/oliver-30/`.

He, Jinqiang et al. (2017). "A new pheromone update strategy for ant colony optimization." In: *Journal of Intelligent Fuzzy Systems.[Online]* 32 (5), pp. 3355–3364.

Stutzle, T. and H. Hoos (1997). "MAX-MIN Ant System and local search for the traveling salesman problem." In: *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97). [Online].* Pp. 309–314.