An Introduction to Ant Colony Optimization

A short introduction to the origin, methods, and uses of Ant Colony Optimization

Alexander Lustig
Weber State University
Ogden, Utah, United States
alexanderlustig@mail.weber.edu

ABSTRACT

This project provides a short introduction to, and exploration of, Ant Colony Optimization (ACO) as well as demonstrating a simple implementation of an ACO algorithm. Ant Colony Optimization is a probabilistic meta-heuristic method designed to approximate an optimal solution to problems which can be simplified to certain kinds of graph traversal. Most famously ACO has been used to find near optimal solutions to the Traveling Salesman Problem (TSP). The implementation for this project is simple but nonetheless effective in very quickly approximating optimal solutions for the TSP.

1 Motivation for this project

My motivation for this project was simply curiosity, during this class we never directly touched on heuristic algorithms, although they did appear several times while researching for the later survey papers. I wanted to explore them, and the intuitive nature inspired design of ACO appeared to offer a low bar of entry for that purpose.

2 Introduction to Ant Colony Optimization

Ant Colony Optimization (ACO) arose as a result of the following observation regarding ant colonies: Live ants will deposit trails of pheromones between the colony and objectives such as food. Other ants in the colony will then begin to follow that trail and deposit their own pheromones. However, importantly, the other ants will follow that pheromone trail imperfectly, some finding longer paths and some finding shorter paths. The shorter paths allow ants to repeatedly deposit pheromone trails faster than on longer paths, this stronger pheromone trail draws more ants to the short paths and creates a positive feedback loop. Over time this positive feedback loop causes the distance of the path to either reach or approximate a minimum distance between the colony and the ants objective [1][2][3]. This observation is what led Colomi et al to develop the first published iteration of Ant Colony Optimization in 1991, then known as *Ant system* [1].

The premise of ACO is to mimic the tendency of natural ants to find progressively more optimal paths between two points, and apply that positive feedback loop to path problems which are otherwise very difficult to solve, such as the famous Traveling Salesman Problem.

ACO mimics this by running virtual ants through a graph which represents the problem. Each time an ant finds a new path which is shorter than a previous path, the virtual ant places a pheromone trail along each edge of that path. As the algorithm iterates, future ants are more likely to move along edges with larger pheromone values, since those high pheromone edges are known to lead to an overall shorter path than other edges.

The virtual ants, again borrowing from nature, follow these paths imperfectly, and can use any number of probabilistic methods to decide where they should move to. While traversing the graph and following pheromone trails, they may find a shorter variation of an existing path, which then causes the virtual ant to lay down a pheromone trail along that new shorter path.

Finally, to prevent the virtual ants from being locked into a particular path, after each iteration of the algorithm, the pheromone values for each edge of the graph are reduced by some amount. By reducing the pheromone values along the entire graph, the virtual ants will become more likely to travel along unused edges and find completely new paths through the graph. This step, which mimics the natural evaporation of pheromones, allows the virtual ants to try and find completely new paths through the graph, if variations of existing shortest paths have not been successful. The ACO algorithm then iterates until the desired stop condition, such as time, number of iterations, or minimum distance, is reached.

3 This project's implementation

ACO itself is less an individual algorithm, but more a class of closely related algorithms, all of which leverage the path shortening positive feedback loop displayed by real world ants.

The simple ACO algorithm implemented for this project functions quite similarly to the *ANT-cycle* algorithm described by Colomi et al [1].

In the implementation of an ACO algorithm for this project, the graph is composed of two parts, a list of vertices and a matrix, or 2d array, containing the length and pheromone strength for each edge in the graph. The length value does not necessarily need to represent the euclidean distance between the vertices or be identical for both directions of the edge, however, for simplicity, this was how the graph was implemented. These length values are computed and stored prior to running the algorithm, allowing the ants quick access to those values during operation.

The ants for this implementation contain a list of all vertices which that ant has already traveled to and two weight values, the first weights the ants preference for pheromones, the second weights the ants preference for length.
The random decision making for these virtual ants is provided by the C++ standard libraries
std::discrete_distribution<>() function. This function allows the ants to randomly select an unvisited vertex based on the weight of the edge connecting that unvisited vertex to the ants current vertex. The weight of any unvisited edge *e* for ant *a* is determined by the following formula:

$$\frac{e.pheromoneValue * a.pheromoneWeight}{(e.length * a.lengthWeight)^2}$$

Figure 1: **The implemented edge weight formula**

The exact formula for determining the edge weight will vary between ACO algorithms, the formula in figure 1 is a variation of the *ant-system* edge weight formula:

$$\frac{e.pheromoneValue * a.pheromoneWeight}{e.length * a.lengthWeight}$$

Figure 2: **The *ant-system* edge weight formula [1]**

During testing and experimentation, the formula from figure 1 appeared to cause the ACO algorithm to produce smaller minimum paths, and caused the ants to converge on those minimum paths faster than the figure 2 formula.

After the graph and ants are prepared, the algorithm loops for the specified number of iterations. In each iteration, the algorithm sends a virtual ant through the graph. If the ant returns a path smaller than the path returned by the previous iteration, the algorithm places a trail of strength *s* along the edges of that path. If the ant returns a new shortest path found, the algorithm places a trail of strength *s * s* along that path. The algorithm then reduces the pheromone values over the graph before starting the next iteration.

Once the algorithm has run for the specified number of iterations, the algorithm returns the shortest path found by the ants.

4 Results and conclusions

The simple ACO algorithm implemented for this project is designed to find approximate solutions for the Traveling Salesman Problem. The Traveling Salesman Problem (TSP) is a famous and difficult problem, and as it can be reduced to a graph traversal problem, a prime candidate for the use of an ACO algorithm [1][2][3]. TSP, when all paths are symmetrical, has (n-1)! /2 possible paths, where n is the size of the graph. This superpolynomial growth makes brute force solutions to TSP highly ineffective. For example, a graph with 100 vertices would have approximately 4.666311 x 10^155 possible paths.

Running the implemented simple ACO algorithm however, using a randomized graph of 100 vertices and 20,000 iterations, the algorithm is able to, with varying degrees of success, approximate an optimal solution, such as figure 3, in under 60 seconds.

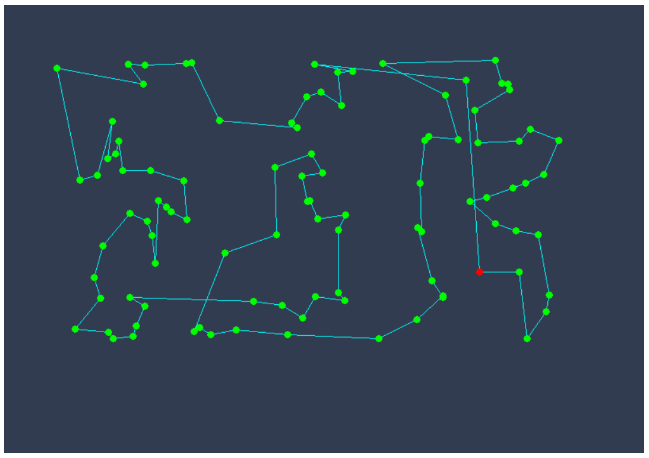

Figure 3: **The solution returned for a 100 vertex graph after 20,000 iterations**

The solution in figure 3, while clearly imperfect, still represents a significant step forward in efficiently finding a usable solution for TSP. This implementation is a very simple one, based on the *ant-system* published in 1991 [1]. Since the publishing of that paper, further innovations and improvements have allowed more advanced implementations of ACO to find great success in its computational niche [2] [3].

REFERENCES

[1] Colorni, A., Dorigo, M., & Maniezzo, V. (1991, December). Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life* (Vol. 142, pp. 134-142).

[2] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," in *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53-66, April 1997, doi: 10.1109/4235.585892.

[3] M. Dorigo, M. Birattari and T. Stutzle, "Ant colony optimization," in *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28-39, Nov. 2006, doi: 10.1109/MCI.2006.329691.