**The Traveling Salesman Problem: How The Ants Solve It**

**Gregory S, Justin S, Cory S**

Computer Science Department

Brigham Young University

3361 TMCB, Provo, UT 84602

**Abstract**

The dilemma faced by the traveling salesman wherein he has a finite number of cities to visit and a desire to minimize the total distance traveled by visiting each one once without backtracking is a popular mathematics problem in combinatorial optimization. Our goal is to implement an Ant Colony Optimization (ACO) algorithm so as to improve upon a simple brute force solution of selecting the nearest neighbor.

**Introduction**

Though the origins are unclear, the Traveling Salesman Problem, or TSP, was mathematically formulated by several mathematicians in the 1800s and later studied in depth in the 1930s. Practical applications range from planning/logistics to DNA sequencing to aircraft scheduling (Xu, 2017). It is the most well-known combinatorial optimization problem. It is well-studied, and its existence has even affected pop culture (Lanzone & Lanzone, 2012).

There are many algorithm optimization models used to solve TSP, and a subset of these are derived from observation of ant behavior. These Ant Colony Optimization Algorithms (ACO), or “ant algorithms,” exploit the behavior of ants to coordinate artificial agents to solve computational problems. The ant colony is a distributed system wherein the whole colony is capable of solving complex tasks that are impossible for a single ant to solve.

Ants have limited vision, and some species are completely blind. Their primary means of communication is through stigmergy, which is “a form of indirect communication mediated by modifications to the environment.” (Dorigo & Stützle, 2004) When searching for food, ants achieve this by leaving a trail of special chemical signals. Other ants detect these chemical signals, called pheromones, and can follow the path to a food source. These pheromones slowly break down; consequently, there is a direct correlation between the strength of the pheromone trail and the frequency of travel. In other words, less-traveled paths disappear and optimal paths persist. In absence of any pheromone trails, ants tend to wander randomly in search of food. Even with trails, ants will still deviate on occasion, and this may result in a more optimal path.

**Algorithm Explanation**

As with other implementations of the ACO algorithm, individual ants programmatically and simulate the random choices made as each “ant” travels from one node of a graph to another. In the context of the solving a TSP, once an ant has identified a (semi-)optimal path, this path is marked with pheromones by altering the semi-random decision tree used by each ant when determining the next node. Subsequent iterations of the algorithm cause improvements as future ants avoid less-optimal paths and identify more optimal ones.

An issue that must be solved for any implementation of Ant Colony Optimization is how much weight should be given to distance or pheromones when an ant is deciding which path to take. The decision is probabilistic and weighted according to the shortness of the distance to other nodes as well as the amount of pheromone to other nodes. We found that in our algorithm at first the pheromone scent was too strong and after one ant found a solution, then all other ants followed that same path. Once we added more weight to distance we started seeing more optimal results. Adding more weight to distance and discounting the weight of pheromones allowed ants to better explore new variants of paths being incentivized to new nodes that had short distances.

Another parameter we needed to adjust in our algorithm was how many ants we had exploring concurrently. We experimented with a variety of quantities (5, 10, 20, the number of cities in the problem) but we found that the optimal number of ants exploring at a time was only sending out one ant at a time. By only sending one ant at a time the algorithm could find solutions faster and could better compare solutions and direct ants to more optimal routes.

**Complexity**

The space complexity was very minimal for algorithm. For each city, we stored an array that contained the distances to every other city, an array with the amount of pheromone to every other city, and the probabilities of visiting every other city. For every ant, we stored the path of cities visited and the cost of that path. So our space complexity was O(n3 + a\*n) where n is the number of nodes and a is the number of ants sent in a batch.

Our algorithm generally explained consists of the following: an ant must visit every city and at every city the probability must be calculated for every city. This lends to a time complexity O(an2) again where n is the number of nodes/cities and a is the number of ants sent concurrently. We repeat this m times, where m is the number of iterations until convergence. So Overall the time complexity of our algorithm is O(a\*m\*n2)

**Analysis of Results**

As expected, our version of the ACO algorithm performance difference, when compared with the Branch and Bound algorithm, was negligible at smaller node quantities (and less optimal in some cases). However, as soon as the problem size reached 60 or above, our algorithm performed better than Branch and Bound and solved the problem in far shorter of a time. The longest our algorithm ever took to solve a problem was 12 seconds and that was with problem size 200.