Name: Tomas Pickford User-ID: rvkb79

Algorithm A: Ant Colony Optimization

Algorithm B: Basic Greedy Search

Description of enhancement of Algorithm A:

My first enhancement was to implement ASrank. Instead of adding pheromone to all ants' trails, only the top w – 1, plus the best so far tour, receive pheromone at the end of each iteration. The amount of pheromone depends on the ant's rank (compared to the other w ants), as well as the length of its tour as usual.

My second enhancement was to write a 3-opt local search function, taking inspiration from "Ant Colony Optimization" by Marco Dorigo and Thomas Stützle. The function takes only the best tour at each iteration (to save on processing time) and cuts it into three sequences. It then pieces them back together in seven different ways – including by reversing the order of cities within a sequence. It compares the lengths of these tours with the original, and the shortest tour replaces the previous best. This all happens before any pheromone is deposited, so the fact that this tour has the greatest pheromone weighting (due to ASrank) means it is highly likely to be followed by ants on the next iteration, even if some of its edges have never been traversed before. I found that this enhancement is great for most city sets, as local search makes improvements to the best ant's trail at every iteration, and different executions of the algorithm produce consistently small lengths (ants don't get stuck going down one sub-optimal path because too much pheromone was deposited there initially because there's another factor; the local search). For large city sets, there are so many positions in the tour to cut (as the function has cubic time complexity) that the search function leaves too little time to have many ant search iterations, which increases the average tour length.

Description of enhancement of Algorithm B:

I found that the length of tour returned by the Basic Greedy Search varied a lot from execution to execution, because certain starting cities led to better tours. Ideally, the algorithm would be able to try the tour from every starting city and see which gave the least distance, however for large city sets this might take too long. I decided to add the parameter "repeats" to define how many starting cities should be tried out. The starting city is chosen randomly each time.

Basic Greedy Search often chooses a sub-optimal path because it only looks at the nearest immediate cities and doesn't look ahead any further. To address this issue, I modified the code so that it would look through all the possible paths through three cities from the current city, and decide which to take by finding the least total cost of the three moves. It would only move to the first of those cities however, and then repeat the process of looking for all chains of three from there, so as not to miss an opportunity for a shorter route. In the end, I found this took too long for very large city sets, so my submitted enhanced algorithm only looks through possible paths of length 2 to decide which next city to visit (whereas the basic implementation only looks at one).