David Jangdal jangdadm@rose-hulman.edu

The Travelling Salesman Problem
Solved with Ant System

The travelling salesman problem is well known within the field of computer optimizations. It is the task of finding the shortest way between all the nodes on a graph and end on the same node as you started. The story comes from a salesman trying to visit every city and come back to the starting one by travelling the shortest road.

The program consists of the files Main.java Ant.java and Node.java with three classes: Node, Vertex and Ant to solve the TSP. The first attempt for a solution was a random approach. At each node, take a random vertex leading to a city that has not yet been visited. The second attempt chose the vertex with the least cost at each city. They all used random starting location and the third and final attempt was the Ant System, AS

Ant system is inspired by ants in the way that they lay pheromones where they walk, and follow other paths where pheromone is present. When an ant is about to select a vertex for its next destination, the transition rule from figure 1 is used. The parameters alpha and beta can be used for tuning the probability. These values need to be different depending on the size of the graph and the variation of vertex costs.

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [n_{ij}]^{\beta}}{\sum\limits_{l \in J_i^k} [\tau_{il}(t)]^{\alpha} \cdot [n_{il}]^{\beta}}$$

Figure 1. Transition rule to make the ant decide its next destination.
Tau is the amount of pheromone and n is the inversed distance of the vertex

When the ant has visited each city and got back to its origin, the path is compared to the currently best path and updates the best path if the ant's path is better. After all the ants have completed a tour, the pheromone level on each vertex is updated and then another round starts. The pheromone level is increased if the vertex is part of the best tour and for each ant that travelled on the vertex. The amount to increase depends on how well the path performed.

The ant system might seem complex at a first glance but is relatively easy to implement. It is a bit hard to choose all the right tuning parameters, but after a few test runs one will get a feeling for the range of the values.

Results

The first attempt, random, would solve all the small graphs if enough ants and rounds were used. Small graphs are up around 6 nodes. It will probably solve every graph with that condition but it is not doable on larger graphs due to the branching factor.

The second method, take minimal path, did not perform really well. Since it will always choose the path with lowest cost it can never change its path and therefore never improve its result from the first run. It can solve a graph if it is constructed in the right way.

Depending on the tuning parameters, especially alpha and beta, the ant system performs really well. Even on larger graphs, around 30 nodes, it can find the correct solution. On graphs with 100 nodes and over the algorithm is too slow and will take minutes to complete. Figure 2 shows the result of AS on a fully connected 10-node graph by varying the alpha value with a beta value of five. By only looking at the graph, 0.1 seems like a good value for the alpha. But with the low values of alpha it often finds a good solution early and then just runs around on nodes that are not so close to the solution. With higher values on alpha the ants get stuck on the same path after round 3 or so, and no improvements are found later on. So if it didn't find the best path early, it won't find it. This makes the tuning parameters really important and hard to find good values and arbitrary graphs.

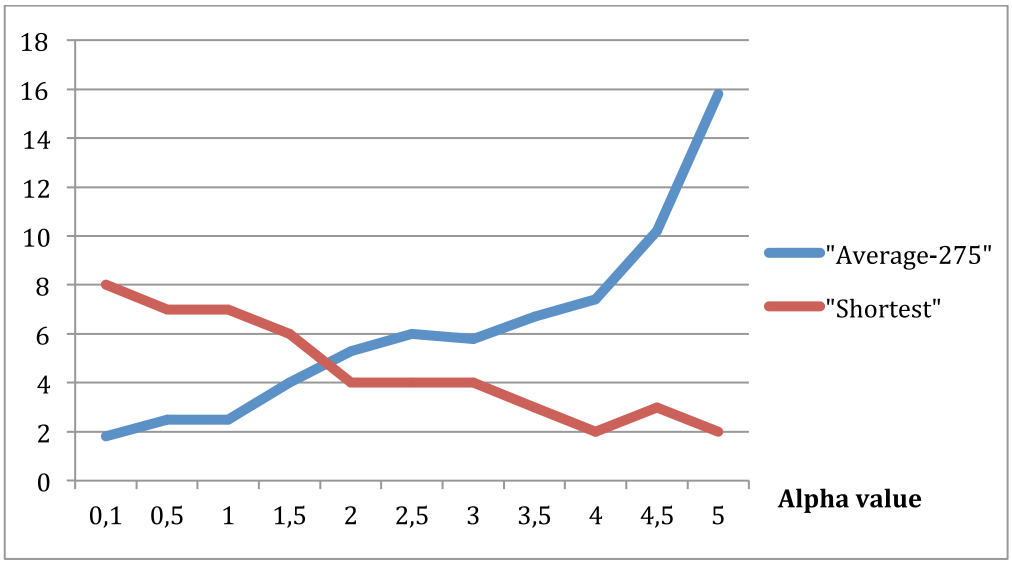

Figure 2. Results from ten test runs of AS on a ten node fully connected graph. The blue line is average travelling cost minus 275, 275 being best path. The red line is how many times it found the best path, out of ten times. X-axis is alpha level on transition rule.

Improvements

The reason why bigger graph takes so long time is the pheromone update part. After each round, it will go over each node N, each vertex V, each ant A and each vertex in the path P of the ant. Since it is a fully connected graph, the amount of vertices to a node is the same amount as nodes minus one, which gives V = N. The amount of ants used is set to the same as number of nodes, A = N. And since we need to visit each node, the path is equal to number of nodes, P = N. This gives a complexity of N*N*N*N, $O(N^4)$, which is really bad. The implementation could instead update the pheromone for each path taken plus the best path. This will instead give a complexity of $O(N^2)$ and hopefully reduce the run time significantly.

David Jangdal jangdadm@rose-hulman.edu

Personal thoughts

The travelling salesman problem is really interesting and a good way to test swarm intelligence. If the project was larger I would like to have implemented a graphical representation of the graph where you can see the ants moving around and the level of pheromone changing over time. In that way it would be easier to see when the ants makes the “wrong” decisions, either following a path too much or too little, and make your own improvements to it. I would also try to automatically predict good values for all the parameters depending on the current graph.

In almost all the other programming assignments, you are given some code and are just supposed to fill out the “blank lines”. The description for this assignment was concise and well defined and you could implement it in whatever way you wanted too and with a language of your choice. It was a good programming assignment.
Perhaps it would be even more fun if it were some kind of a competition within in the class. Who could solve the biggest graph within some time or who can find the best answer the most consecutive times etc.