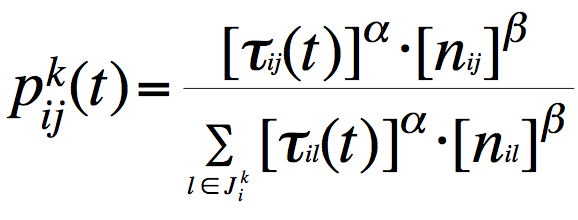
The Travelling Salesman Problem

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.



*Figure 1. Transition rule to make the ant decide its next destination.*

*T 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 the graph if it is constructed in the right way.

*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.*

Because I had single way vertex I had two vertexes for each vertex

For each node

For each vertex connected to the node

For each ant

For each node

For each node

N \* V \* M \* P

How they performed

Did good on small graphs, random with a lot of ants

-graphs with result

-why it doesn’t find the optimal path

-increase alpha same thing as decreace beta

-further improvements

-takes a long time to update pheromone

-ants update vertices