# Recurrent Neural Networks for Optimization

Edward Venator

EECS 484

In this assignment, I created a Hopfield Network—a neural network capable of performing optimization. In this case, the Hopfield network is solving the Traveling Salesman problem for the case of ten cities to be visited in ten days. The network consists of one hundred neurons, each representing a pairing of a city and a day. When a neuron fires, it represents visiting that city on that day. The constraints on the problem, that each city be visited once and only one city be visited on each day, are used to generate the weights between the neurons. Neurons representing the same day inhibit each other and neurons representing the same city inhibit each other. In addition there is a global inhibition and an inhibition representing the cost of the trip. An "excitation current" forces the network to create a solution. An additional integral term decreases the cost inhibition as the network runs, making the constraint inhibitions dominate and forcing a valid solution.

After setting up the network to be basically functional, I began tuning the parameters to try to find a good solution. My method consisted of increasing the data term inhibition to encourage more efficient solutions, then running 1000 iterations (usually enough for a solution to emerge), and if a set of values produced invalid solutions, increased the appropriate constraint inhibition. I also experimented with varying the excitation current, which I decided to set to 20 (twice the product of C and n\_cities). Varying the integration parameters, I found that integrating faster tended to reduce the efficiency of the paths created. I finally settled on this set of parameters:

A = 10;

B = 10;

C = 1;

D = 50000;

Jbias\_factor = 20.0;

lambda = 2000.0;

dt = 0.001;

noise\_scale = 0.1;

To evaluate my results, I ran the code twenty times. Each time, the network was simulated for 100 iterations, then a termination condition was checked. I would then continue to simulated the network in batches of 100 iterations until the termination condition was fulfilled. The termination condition was that the system create a valid plan (ten cities in ten days). The results of simulation with my values was as follows:

Minimum cost: 4.71

Maximum cost: 6.89

Mean cost: 6.00

Minimum run time: 600 iterations

Maximum run time: 2000 iterations

Mean run time: 11.25 iterations

I found that increasing the data inhibition term by a factor of ten resulted (counter intuitively) in a slightly higher mean cost and lower mean run time. Since the network seemed to have little problem finding a legal solution, I did not increase A or B. As mentioned above, the integral term tended to hurt the efficiency of the solutions. I kept C constant, since varying it would only serve to scale the other terms. For the most part, I found that varying the parameters had little effect on the mean cost result, but mainly changed the running time. Of course, the integral feedback was critical to my results. Many of the simulations initially converged to illegal paths, which were then fixed (after many iterations) by the integral term.

These results are, on average, only marginally better than a random choice (6.1484), but they do guarantee an upper bound on cost (6.89) that is lower than a random choice, which could have a cost as high as 8.0657.