Shaun Howard 11/12/15
EECS484 PS9 smh150

PS9: Solve TSP with Hopfield Network

Summary of Solution:

To begin I refactored the provided starter code. I organized all comments to be above variable assignments or function calls. I rearranged the code in such a way that I could effectively test the program automatically in a loop running for the set maximum number of iterations. I changed u0 to be the negative log function of the number of cities minus one since we already have the first destination set at city one on day one. I kept dU, the random signal noises on inputs, as they were, but I scaled the signal noise to 0.1 of the randomly generated noise as described in Hopfield's paper.

I added a bunch of statistical measurements to compare my results with the provided exhaustive set as well as with other attempts at optimizing the parameters. I completed the weight assignment code, reorganizing and recycling terms as much as possible. The two important cases I created for penalizing the synapse weights for visitation of same city on two different days as well as two cities on the same day are the following:

```
% penalize visitation of same city on two different days
if(X==Y)&&(ix~=jy)
      Tabc(X,ix,Y,jy)=Tabc(X,ix,Y,jy)-A;
end

% penalize visitation of two cities on the same day
if (ix==jy)&&(X~=Y)
      Tabc(X,ix,Y,jy)=Tabc(X,ix,Y,jy)-B;
end
```

I kept the term from equation 12 in Hopfield and Tank's paper: -Uxi/τ. I also used the integral error term provided to achieve efficient and valid solutions. I found that the rest of the network as it was without the integral error term did not satisfy the TSP at least within a reasonable amount of time. I found that many u-dot values were growing at an equal or similar rate, disallowing for the best path to fully emerge across the entire city-day space. It is possible that the proposal of Hopfield and Tank is valid given optimally adjusted parameters, but I found that testing those parameters would not be a good way to spend time since the credibility of the algorithm was already discussed in class. Instead, I focused on experimenting with optimal parameters for the network with the integral-error feedback term included in the computation. I experimented with most tunable parameters such as A, B, C, D, lambda, and time step (dt) in order to find the most efficient and optimal parameters in terms of lowest mean trip cost after 100 iterations of running the TSP computations for each combination I tried. I found that holding j-bias factor at 1 was perfectly fine.

I added code to compute the statistics provided with the solutions of the exhaustive sample set. I did this in an effort to easily compare my solutions to the exhaustive solutions. My experiments and statistical results follow.

Experiments:

The initial adjustable parameter values I used were the following:

A	B	C	D	Lambda	Dt
500	500	200	1000	5000	0.0001

I chose these initial parameters because they are approximately those suggested by Hopfield in his paper on the TSP-solving network. This initial experiment was to test how well his values initially performed. In this experiment the D term was twice of Hopfield's proposed D term.

The network produced the following statistics with the above parameters:

Average iteration count per solution: 174.870000

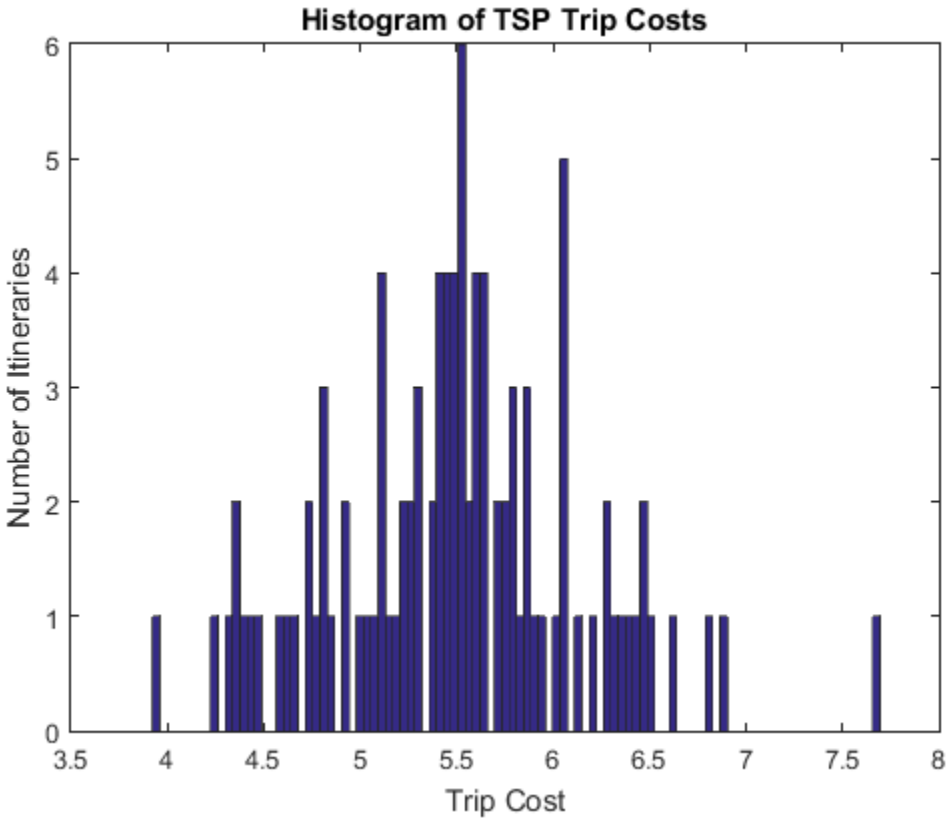
Mean trip cost: 5.495399

Std trip cost: 0.643324

Min trip cost: 3.922910

Max trip cost: 7.695153

The histogram produced based on the number of itineraries found vs. the trip cost for those itineraries follows:



The following is a table to compare the two sets of runs of the different network implementations:

Solution	Avg. Iteration Count	Mean Cost	Std Cost	Min Cost	Max Cost
Exhaustive	unknown	6.15	0.623	3.73	8.07
Initial	175	5.5	0.643	3.92	7.7

Clearly these initial values have already proved to have more optimized statistical measurements than the exhaustive computations. We can see that all but the minimum cost and standard deviation has decreased using approximately the same values suggested by Hopfield in his original paper. We have to consider though the integral-error feedback term being applied when computing u-dot. This term allows us to salvage the good aspects of his paper as far as network design and provide feedback to the network in order to find a best path quicker. This integral-error term 'pats' the less competitive neurons down to 0 per each row while it causes the most competitive neuron in that row to fire at full capacity (of 1). The results are a best path solution to the traveling salesman (n-cities) problem for the given instance.

Despite the goodness of these results, I found that we could do better in terms of reliability and trip cost. I modified the initial parameters to give more penalty for having a higher trip cost per solution by increasing D an order of magnitude. The newer, more concise parameter values I used were the following:

A	B	C	D	Lambda	Dt
500	500	200	10000	5000	0.0001

The reason why I chose to put more influence in penalizing trip cost was to find a lower mean trip cost overall. The statistical results of the network with these parameters were the following:

Average iteration count per solution: 865.650000

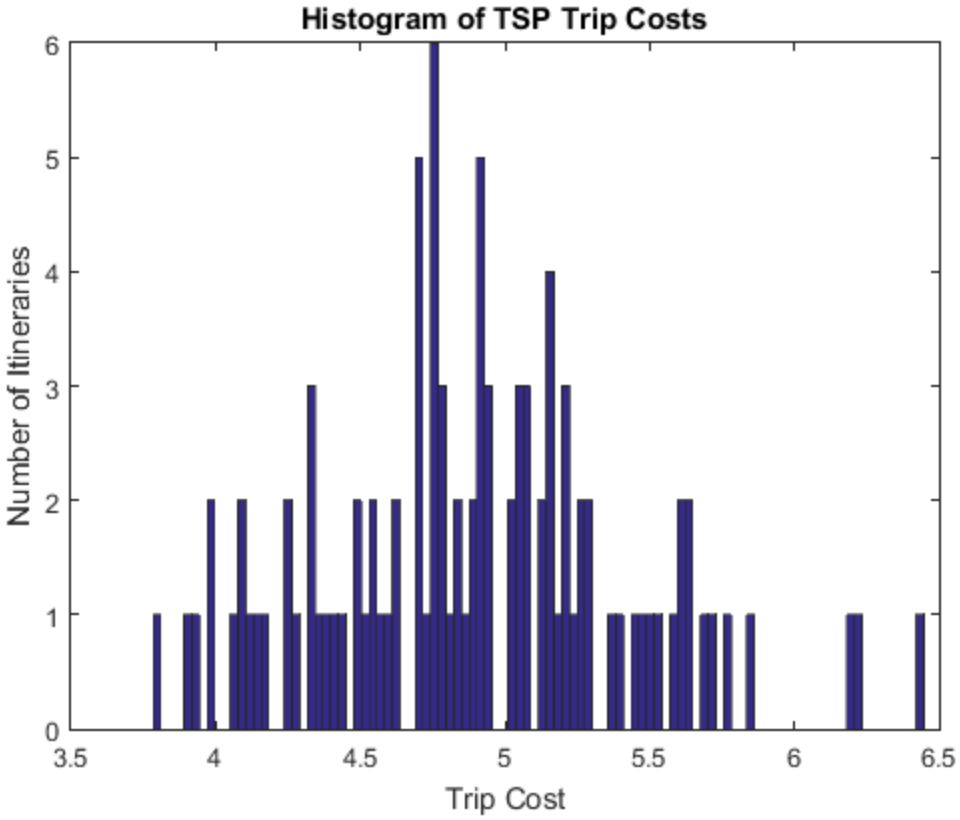
Mean trip cost: 4.893423

Std trip cost: 0.531938

Min trip cost: 3.784153

Max trip cost: 6.448241

The histogram produced based on the number of itineraries found vs. the trip cost for those itineraries follows:



Again we have the table to compare all runs of the network with different parameters so far:

Solution	Avg. Iteration Count	Mean Cost	Std Cost	Min Cost	Max Cost
Exhaustive	unknown	6.15	0.623	3.73	8.07
Initial	175	5.5	0.643	3.92	7.7
Dist. Penalty	866	4.9	0.53	3.78	6.45

As we can see, adding a greater penalty for having a large trip cost has allowed us to strip half a unit off the mean trip cost as compared to our initial parameter set. The standard deviation also diminished by about .1 since the last experiment. The minimum cost trip from our solution has approached the exhaustive solution's minimum cost trip. The maximum cost trip however is noticeably less than the previous experiments. The only downside of this set of parameters is that the network has become less efficient in terms of performance since more iterations were needed to find a proper solution for this network. In order to fix this, I decided to run another experiment with smaller parameters. Hence, for the next experiment I cut all larger parameters down by one order of magnitude. Additionally, I increased the time step by two orders of magnitude to speed up the solution convergence process.

The revised and efficient parameter values I used were the following:

A	B	C	D	Lambda	Dt
50	50	20	1000	500	0.01

The reason why I chose to put cut the size of the variables was in hope of improving performance of the network. The statistical results of the network with these parameters were the following:

Average iteration count per solution: 245.470000

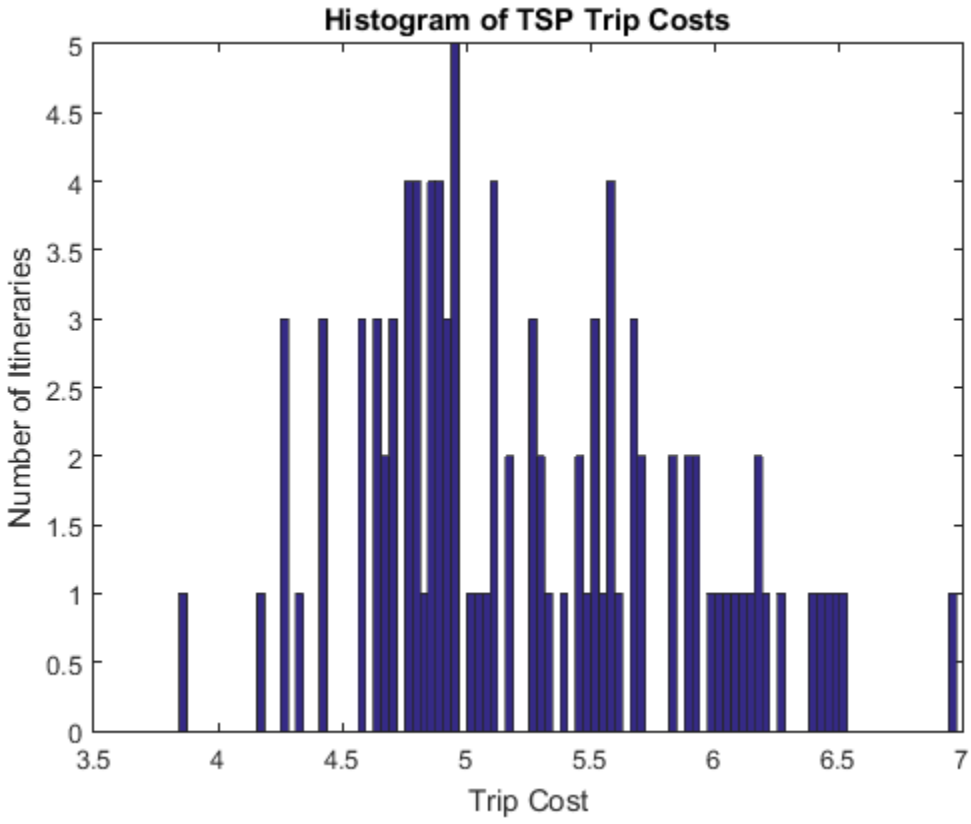
Mean trip cost: 5.251505

Std trip cost: 0.631434

Min trip cost: 3.844137

Max trip cost: 6.977191

The histogram produced based on the number of itineraries found vs. the trip cost for those itineraries follows:



Again we have the table to compare all runs of the network with different parameters so far:

Solution	Avg. Iteration Count	Mean Cost	Std Cost	Min Cost	Max Cost
Exhaustive	unknown	6.15	0.623	3.73	8.07
Initial	175	5.5	0.643	3.92	7.7
Dist. Penalty	866	4.9	0.53	3.78	6.45
Efficiency	245	5.25	0.63	3.84	6.97

The results of this experiment show that my parameter adjustments have led to a more efficient and yet still optimized solution with respect to overall trip cost. We can see that the number of iterations necessary to reach a solution is ~250 as compared to ~870 for the previous set of parameters. The statistics of this run show that these efficient parameters take the best of both worlds of the previous experiments. The mean trip cost is more than the second experiment but less than the first. The standard deviation as gone back up near that of the first experiment, but the minimum cost has not. The maximum cost is also between that of the first and second experiments. Analyzing the histogram for this run, we can notice that the bulk of the values for trip cost are comparable to the mean trip cost for the slower but more precise second experiment. For instance, the median of the trip costs found for this experiment is 5.1 and the mode is 4.26. This shows that our efficient parameter selection is optimal for our Hopfield network design. We can conclude that we have found both efficient and optimal values in terms of solving the traveling salesman problem with a Hopfield neural network.

Discussion/Conclusion:

Overall we can see that all of my experiments using a Hopfield neural network with integral-error feedback have statistically surpassed the exhaustive set of solutions to the traveling salesman problem. We can see that different parameter values affected different aspects of the city-day combinations produced by the network. The first set of values, suggested by Hopfield in his paper, worked fairly well to find a valid solution to the TSP given the effect of integral-error on u-dot. Of course, we had to choose an initial lambda value for this calculation as well. The provided value for lambda was chosen in the long run. It seemed to have enough influence in the overall calculation to efficiently produce a valid path with all each unique city traveled to on non-overlapping days.

The experiments show that penalizing trip distance (cost) more than the other factors results in a smaller mean trip cost produced by the network. They also show that having a larger time step causes the network to converge quicker. This is due to the fact that the change in time affects the integral-error feedback to the network. The more error feedback is provided, the quicker the network will converge to a good solution. I found that having the integral-error term in the solution computation was very beneficial. I could not find a valid solution with Hopfield's proposed values by simply removing the integral-error feedback term.

Obviously finding the shortest path possible is a good feature in a solution, but efficiency should also be considered. A good balance of distance penalty and error feedback were found to provide optimality and efficiency needed to solve a problem like this in finite time. This is a significant improvement over the work of past computer scientists. They have been trying to solve the TSP for ages. The Hopfield network with an integral-error feedback term provides a really good solution to predicting the best path with noisy input values. The parameters used in the computation must be tweaked to meet the desired criteria, but optimal ones can actually be found. This is even an improvement on Hopfield's original work when he thought he solved the problem. We should pat ourselves on the back (mainly Professor Newman).